UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

DESARROLLO DE UNA APLICACIÓN ANDROID CAPAZ DE RECONOCER Y REGISTRAR AUTOMÁTICAMENTE NUESTRA ACTIVIDAD DIARIA

TRABAJO FIN DE GRADO

febrero - 2020

AUTOR: Vicente Pina Beti

DIRECTOR/ES: César Fernández Peris

AGRADECIMIENTOS.

A mi tutor César Fernández Peris por guiarme a lo largo de este proyecto y a todas las personas que me han ayudado y apoyado a lo largo de mi etapa universitaria.

Índice

Capítulo 1. Introducción 1

1.1. Apps móviles 1

1.2. Actividad física 2

1.3. Objetivo del proyecto 3

1.4. Elección de Android como plataforma 4

Capítulo 2. Servicios en Android 6

2.1. Servicio del reconocimiento de actividad 7

2.1.1. onStartCommand 7

2.1.2. requestActivityUpdatesButtonHandler 8

2.1.3. removeActivityUpdatesButtonHandler 9

2.2. Servicio de la ubicación 10

2.2.1. LocationListener 10

2.2.2. StartTracking y stopTracking 11

2.3. IntentService 13

Capítulo 3. Detección de actividad 15

3.1. HandleUserActivity 15

3.2. Imágenes 19

Capítulo 4. Mapas 21

4.1. Google Maps API 21

4.2. Fragment 23

Capítulo 5. Ubicación 25

5.1. LocationManager 25

Capítulo 6. Base de datos 28

6.1. ListView 30

Capítulo 7. Permisos y Manifest 32

7.1. Permisos 32

7.2. Android Manifest 35
7.2.1. Nombre del paquete e ID de la aplicación 35
7.2.2. Componentes de la aplicación 36
7.2.3. Filtros de intents 36
7.2.4. Iconos y etiquetas 36
7.2.5. Elementos 36
7.2.6. Atributos 37
Capítulo 8. Pantallas y Layouts 38
8.1. Funcionamiento de la aplicación 38
8.2. Layouts 43
8.2.1. Pantalla Inicio 43
8.2.2. Pantalla Principal 46
8.2.3. Pantalla Info 48
Capítulo 9. Conclusiones y futuras mejoras 51
Capítulo 10. Bibliografía 52
Anexo I. AndroidManifest.xml 56
Anexo II. Toast 58
Anexo III. Logcat 59

Índice de figuras

Figura 2.1.1.1. onStartCommand() 7
Figura 2.1.2.1. requestActivityUpdatesButtonHandler() 8
Figura 2.1.3.1. removeActivityUpdatesButtonHandler() 9
Figura 2.1.3.2. startTracking() & stopTracking() 9
Figura 2.2.1.1. LocationListener 11
Figura 2.2.2.1. startTracking & stopTracking 11
Figura 2.3.1. onHandleIntent 13
Figura 2.3.2. broadcastActivity 13
Figura 3.1.1. handleUserActivity 1 16
Figura 3.1.2. handleUserActivity 2 17
Figura 3.1.3. handleUserActivity 3 18
Figura 3.2.1. Andando 19
Figura 3.2.2. Desconocido 19
Figura 3.2.4. En bicicleta 19
Figura 3.2.5. Parado 20
Figura 3.2.6. Corriendo 20
Figura 3.2.7. En vehículo 20
Figura 4.1.1. Servicios Google Play 21
Figura 4.2.1. Layout Fragment 23
Figura 4.2.2. SupportMapFragment 23
Figura 5.1.1. LocationManager 1 26
Figura 5.1.2. LocationManager 2 27
Figura 6.1. DatabaseHelper 29
Figura 6.1.1. AdapterView 30
Figura 6.1.2. viewData() 31
Figura 7.1.1. Permisos AndroidManifest 32
Figura 8.1.1. Funcionamiento Pantallas. 38
Figura 8.1.2. Pantalla Inicio 40
Figura 8.1.3. Pantalla Principal. 41
Figura 8.1.4. Pantalla Info. 42
Figura 8.2.1.1. Layout Inicio. 43
Figura 8.2.1.2. Layout Inicio. 44

Figura 8.2.1.3. Layout Inicio. 45
Figura 8.2.2.1. Layout Principal. 46
Figura 8.2.2.2. Layout Principal. 47
Figura 8.2.3.1. Layout Info. 48
Figura 8.2.3.2. Layout Info. 49
Figura 8.2.3.3. Layout Info. 50
Figura Anexo I. AndroidManifest.xml 56
Figura Anexo I. AndroidManifest.xml 57

Capítulo 1. Introducción

1.1. Apps móviles

Once años después del lanzamiento de las tiendas de aplicaciones de Google y Apple (Google Play y Apple Store) podemos observar como las aplicaciones móviles nos brindan ventajas increíbles y unos estándares de estilo de vida nunca vistos. Esto se debe al uso del poder de procesamiento súper rápido de la tecnología de los Smartphones, altamente evolucionada hoy en día, y su red de soporte. El portal web Statista las confirma como las principales tiendas de aplicaciones pues actualmente albergan la cifra de 6,3 millones de aplicaciones móviles.

Dos factores principales que impulsaron la erupción de las aplicaciones móviles fueron la capacidad del usuario para descargar y usar aplicaciones inteligentes en sus dispositivos y la ventaja de llevar sus teléfonos móviles donde quiera que vayan.

Debido a estos dos factores, la sociedad comenzó a darse cuenta de cómo es posible reemplazar las actividades informáticas sedentarias con una experiencia de aplicación móvil dinámica, interactiva e innovadora.

El desarrollo de software móvil causó una movilidad dinámica en las personas. En lugar de accionar varias características en la versión única del programa en el escritorio, la tecnología que nos brindan los Smartphones ha hecho que las personas realicen tareas informáticas desde sus propios teléfonos móviles.

Además de proporcionar funciones básicas de escritorio como la calculadora, el calendario, etc. Las aplicaciones dieron más poder a los usuarios de los teléfonos móviles, como la edición de fotos y vídeos, blogs y diseño, redes sociales, juegos y mucho más.

Por todo lo comentado anteriormente, se puede decir que el desarrollo de aplicaciones móviles es de los mayores avances para la sociedad. Mejorando el método para mantenerse conectado, las aplicaciones móviles nos han dado una perspectiva completamente nueva de la vida.

1.2. Actividad física

La OMS define el término "*actividad física*" como "todo movimiento corporal producido por músculos esqueléticos que exija un gasto de energía". Además, resalta que no ha de confundirse actividad con ejercicio o deporte.

El sedentarismo es uno de los principales factores de riesgo en lo que respecta a la mortalidad mundial. Además, la inactividad física está relacionada con todo tipo de enfermedades, entre las cuales destaca el cáncer, la diabetes, la obesidad, la depresión, las cardiopatías, entre otras.

La actividad física, además de reducir el riesgo de enfermedades, mejora la capacidad cardíaca y reduce la posibilidad de sufrir ataques al corazón, así como la hipertensión, ayuda contra la depresión e, incluso, nos puede ayudar a mejorar la calidad del sueño. También combate todas las enfermedades asociadas a la diabetes, incluyendo la obesidad, que es la mayor epidemia mundial a día de hoy. La actividad física, incluso para los que no hacen deporte, puede ayudarles a mejorar su calidad de vida, llegando a salvarla.

Además, como se ha demostrado, la actividad física sirve para contrarrestar posibles daños causados por accidentes, problemas de salud y enfermedades, estás últimas normalmente relacionadas con el sedentarismo. También se puede encontrar una mejora en el control muscular y la coordinación, evitando caídas, o mejorando los reflejos. En conclusión, la actividad física no solo es beneficiosa, sino necesaria.

La OMS nos da unas recomendaciones orientadas para cada etapa de nuestras vidas con el fin de mejorar las funciones cardiorrespiratorias y musculares, salud ósea y reducir el riesgo de ENT (Enfermedades No Transmisibles) como infartos de miocardio, cáncer, diabetes y depresión.

Para los adultos que comprenden las edades de 18 a 64 años, la actividad física se basa en realizar actividades recreativas o de ocio, desplazamientos (paseos a pie o en bicicleta),

actividades ocupacionales (trabajo), juegos, tareas domésticas, deportes o ejercicios programados en el contexto de las actividades diarias, comunitarias y familiares.

Entre las recomendaciones que nos propone la OMS, destacan la de dedicar 150 minutos como mínimo semanalmente para practicar cualquier actividad física aeróbica como andar, correr o ciclismo, o realizar 75 minutos, pero con una mayor intensidad. También propone practicar esas actividades en sesiones de 10 minutos como mínimo y que debemos realizar dos veces o más a la semana, actividades de fortalecimiento de los grandes grupos musculares.

La idea de este trabajo surge a raíz de estas recomendaciones, ya que para controlar la actividad diaria de las personas es interesante contar con una aplicación móvil que sea capaz de registrar dicha actividad en el dispositivo para poder comprobar en todo momento si se están siguiendo dichas recomendaciones e incluso de esta manera, evitar el sedentarismo.

1.3. Objetivo del proyecto

Este documento tiene como finalidad describir el proyecto realizado por el alumno de Ingeniería Electrónica y Automática Industrial Vicente Pina Beti de la Universidad Miguel Hernández de Elche.

El proyecto consiste en el desarrollo de una aplicación Android capaz de reconocer y registrar automáticamente la actividad diaria del usuario, realizada en la plataforma Android Studio.

La aplicación debe ser capaz de interpretar la actividad física que se está realizando en todo momento, así como la ubicación del dispositivo Android, todo esto mediante unos servicios que trabajan en segundo plano. La actividad física que puede detectar esta aplicación son: caminar, correr, ir en bicicleta, estar parado y en vehículo.

Del mismo modo, la aplicación debe devolver los datos en una lista mediante una consulta en una base de datos SQLite donde se almacenará la información.

1.4. Elección de Android como plataforma

Podemos definir un sistema operativo móvil (SOM) como todo aquel conjunto de programas de bajo nivel que nos permiten aprovechar las propiedades del hardware que posee el teléfono móvil y proporciona unos servicios a las aplicaciones móviles, que se ejecutan sobre él.

Al igual que los ordenadores que utilizan sistemas operativos como Windows o Linux, los teléfonos móviles de hoy en día tienen sus sistemas operativos como Android, iOS, BlackBerry, Windows Phone, etc.

Los SOM son más simples y están bastante más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y a las diferentes maneras de introducir información en ellos.

Android Inc. es la empresa que creó el SOM "*Android*". La compañía se fundó en 2003 y fue adquirida por Google Inc. en el año 2005 y dos años más tarde, en 2007, fue lanzado al mercado. En el año 2007, Google fundó la Open Handset Alliance formada por un total de 78 compañías dedicadas al desarrollo de hardware, software y telecomunicaciones para dispositivos móviles. Estas compañías juntas desarrollaron Android, la primera plataforma móvil completa, abierta y libre.

Algunos de los miembros de la Open Handset Alliance son empresas como Google, Nvidia, HTC, Dell, Intel, Qualcomm, Texas Instruments, Samsung, LG, y Wind River Systems entre muchas otras.

Originalmente, Android estaba pensado para ser un sistema para las cámaras digitales profesionales, pero fue modificado por Google para que fuese utilizado en dispositivos móviles como los *Smartphones* y tabletas.

Android cuenta con el mayor número de instalaciones de Smartphones en todo el mundo y es un sistema basado en el núcleo Linux. Las aplicaciones para dicho sistema se escriben y desarrollan en Java, aunque con unas APIs propias.

Aunque se supone que Android es un sistema operativo de software libre y código abierto, en los dispositivos vendidos, la mayoría del software que se incluye es software propietario y de código cerrado.

En España según el portal web Kantar, el 88.9% de los dispositivos móviles usan Android, con respecto a un 10.8% de dispositivos con iOS, un 0.2% con Windows y otros (entre los que podemos encontrar a BlackBerry, por ejemplo).

Es por estas características que se ha elegido el sistema operativo móvil Android para el desarrollo de la aplicación.

Como se va a ver a lo largo de este trabajo, el proyecto ha sido desarrollado en la plataforma Android Studio, entorno de desarrollo integrado oficial, abreviado comúnmente como IDE, de las aplicaciones para el sistema operativo Android. Este entorno se basa en IntelliJ IDEA. Además de las herramientas que nos proporciona IntelliJ para desarrolladores, Android Studio nos proporciona muchas funciones que aumentan nuestra productividad a la hora de la compilación de las aplicaciones como el sistema basado en Gradle flexible, un emulador con varias funciones, un entorno unificado en el que se puede desarrollar proyectos para todos los dispositivos Android, una función Instant Run para aplicar cambias sin necesidad de compilar de nuevo, etc…

Capítulo 2. Servicios en Android

En este proyecto se han implementado en total tres servicios para el funcionamiento de la aplicación en segundo plano: uno para la detección de la actividad, otro para la detección de la ubicación del dispositivo, y un tercer servicio en el que mediante el método *IntentService* la aplicación nos muestra la detección de la actividad en la pantalla.

Un servicio puede ejecutar operaciones de una duración prolongada en un segundo plano y que no proporciona en sí una interfaz de usuario. Un componente de la aplicación puede iniciar un servicio y éste seguirá ejecutándose en un segundo plano, aunque el usuario cambie de aplicación. También un componente se puede enlazar con un servicio para interactuar con él y realizar una comunicación entre procesos (IPC).

Independientemente de que la aplicación esté iniciada, enlazada o ambas, cualquier componente puede utilizar el servicio, incluso si es desde otra aplicación, de la misma manera que cualquier componente puede utilizar una actividad, iniciándola con un Intent. Sin embargo, se puede declarar el servicio como privado en el manifest, y bloquear el acceso de otras aplicaciones.

2.1. Servicio del reconocimiento de actividad

Este servicio está dedicado al reconocimiento de la actividad que se está realizando.

Para el correcto funcionamiento del servicio se implementarán varios métodos con diversas funciones y características: onStartCommand(), requestActivitiyUpdatesButtonHandler() y removeActivitiyUpdatesButtonHandler().

2.1.1. onStartCommand

El método onStartCommand()es llamado cada vez que el cliente inicializa un servicio. Los parámetros son:

- Intent: Un objeto Intent que se indica en la llamada startService(Intent).
- Flags: es información sobre como comienza la solicitud. Puede ser 0, START_FLAG_REDELIVERY o START_FLAG_RETRY. Un valor distinto de 0 se utiliza para reiniciar un servicio tras detectar algún problema.
- startId: consiste en un número entero único representando la solicitud de arranque específica.
- START_STICKY: el sistema tratará de recrear el servicio cuando sea posible y se realizará una llamada a onStartCommand() pero con intent = null. Esto tiene sentido cuando el servicio puede arrancar sin información adicional.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    super.onStartCommand(intent, flags, startId);
    return START_STICKY;
}
```

Figura 2.1.1.1. onStartCommand()

2.1.2. requestActivityUpdatesButtonHandler

Este método nos ayuda a saber si la actualización del servicio ha funcionado o ha fallado mediante unos avisos que se ha creado a través de un objeto Toast con uno de los métodos makeText. Se habla más detalladamente del objeto Toast en el Anexo II.

```
public void requestActivityUpdatesButtonHandler() {
    Task<Void> task = mActivityRecognitionClient.requestActivityUpdates(
            Constants.DETECTION_INTERVAL_IN_MILLISECONDS,
            mPendingIntent);

    task.addOnSuccessListener((OnSuccessListener) (result) → {
            Toast.makeText(getApplicationContext(),
                    text: "Successfully requested activity updates",
                    Toast.LENGTH_SHORT)
                    .show();
    });

    task.addOnFailureListener((e) → {
            Toast.makeText(getApplicationContext(),
                    text: "Requesting activity updates failed to start",
                    Toast.LENGTH_SHORT)
                    .show();
    });
}
```

Figura 2.1.2.1. requestActivityUpdatesButtonHandler()

2.1.3. removeActivityUpdatesButtonHandler

```
public void removeActivityUpdatesButtonHandler() {
    Task<Void> task = mActivityRecognitionClient.removeActivityUpdates(
            mPendingIntent);
    task.addOnSuccessListener((OnSuccessListener) (result) → {
            Toast.makeText(getApplicationContext(),
                    text: "Removed activity updates successfully!",
                    Toast.LENGTH_SHORT)
                    .show();
    });

    task.addOnFailureListener((e) → {
            Toast.makeText(getApplicationContext(), text: "Failed to remove activity updates!",
                    Toast.LENGTH_SHORT).show();
    });
}
```

Figura 2.1.3.1. removeActivityUpdatesButtonHandler()

De la misma manera que en el método anterior, este código nos sirve para ver si se ha eliminado la actualización de actividades o si ha fallado en su intento de suprimir las actualizaciones. Se usan los mismos tipos de avisos y métodos que en el método anterior.

Una vez hecha la clase, se debe instanciarla en la clase principal donde se inicia y donde se finaliza el servicio. Esto es necesario para cada servicio implementado, así como su correspondiente declaración en el archivo XML manifest que se comenta más adelante.

```
private void startTracking() {
    Intent intent = new Intent( packageContext: MainActivity.this, BackgroundDetectedActivitiesService.class);
    startService(intent);
}

private void stopTracking() {
    Intent intent = new Intent( packageContext: MainActivity.this, BackgroundDetectedActivitiesService.class);
    stopService(intent);
}
```

Figura 2.1.3.2. startTracking() & stopTracking()

2.2. Servicio de la ubicación

Este servicio se ha utilizado con el fin de ir actualizando la localización del dispositivo Android a utilizar mediante unos intervalos de tiempo constantes. A continuación, se exponen los métodos utilizados más relevantes del servicio para su correcto funcionamiento.

2.2.1. LocationListener

La clase privada LocationListener se ha utilizado con el fin de recibir notificaciones cuando la ubicación del dispositivo ha cambiado. En ella se implementan varios métodos:

- onLocationChanged(): llamado cuando la ubicación ha cambiado.
- onProviderDisabled(): llamado cuando el usuario desactiva el proveedor de la ubicación.
- onProviderEnabled(): llamado cuando el usuario activa el proveedor de la ubicación.
- onStatusChanged(): nos informa cuando el estado cambia. NOTA: actualmente este método está en desuso a partir del API 29.

```
private class LocationListener implements android.location.LocationListener
{
    private Location lastLocation = null;
    private final String TAG = "LocationListener";
    private Location mLastLocation;

    public LocationListener(String provider) { mLastLocation = new Location(provider); }

    @Override
    public void onLocationChanged(Location location)
    {
        mLastLocation = location;
        Log.i(TAG,  msg: "LocationChanged: "+location);
    }

    @Override
    public void onProviderDisabled(String provider)
    {
        Log.e(TAG,  msg: "onProviderDisabled: " + provider);
    }

    @Override
    public void onProviderEnabled(String provider)
    {
        Log.e(TAG,  msg: "onProviderEnabled: " + provider);
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras)
    {
        Log.e(TAG,  msg: "onStatusChanged: " + status);
    }
}
```

Figura 2.2.1.1. LocationListener

2.2.2. StartTracking y stopTracking

```
public void startTracking() {
    initializeLocationManager();
    mLocationListener = new LocationListener(LocationManager.GPS_PROVIDER);

    try {
        mLocationManager.requestLocationUpdates( LocationManager.GPS_PROVIDER, LOCATION_INTERVAL, LOCATION_DISTANCE, mLocationListener );

    } catch (SecurityException ex) {
        Log.i(TAG,  msg: "fail to request location update, ignore", ex);
    } catch (IllegalArgumentException ex) {
        Log.d(TAG,  msg: "gps provider does not exist " + ex.getMessage());
    }

}

public void stopTracking() { this.onDestroy(); }
```

Figura 2.2.2.1. startTracking & stopTracking

El primer método mencionado sirve para empezar a recibir la información de nuestra ubicación y se muestran dos mensajes cuando no se ha podido recibir la actualización de la ubicación o no existe un proveedor GPS.

Se llama al Location Manager que es la clase que nos proporciona acceso al sistema de los servicios de localización. Está clase se explica de manera más detallada en el punto 5.1. De esta manera recibimos nuestro proveedor GPS y los parámetros para obtener nuestra localización.

La herramienta Android Monitor incluye un monitor de logcat en el que se muestran mensajes para la depuración. Se habla de esta herramienta en el Anexo III.

El segundo método llama a onDestroy() para hacer saber al sistema que el servicio no se utiliza más y es eliminado.

2.3. IntentService

IntentService es una clase destinada a servicios que funciona mediante solicitudes asíncronas (expresadas como Intents). Los clientes envían solicitudes a través de unas llamadas Context.startService (Intent); el servicio se inicia según sea oportuno, maneja cada Intent a su vez utilizando un subproceso de trabajo y se detiene cuando se queda sin trabajo.

Este patrón se usa comúnmente para realizar tareas de la clase principal de una aplicación. La clase IntentService existe para simplificar y cuidar la mecánica de la aplicación. Para usarlo, se necesita extender IntentService e implementar onHandleIntent (Figura 2.3.1) (android.content.Intent). IntentService recibirá las los Intents, lanzará un subproceso de trabajo y detendrá el servicio según sea necesario.

Todas las solicitudes funcionan en un solo subproceso de trabajo, pueden tardar el tiempo que sea necesario (y no bloquearán el bucle principal de la aplicación), aunque solo se procesará una solicitud a la vez.

```
protected void onHandleIntent(Intent intent) {
    ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);

    ArrayList<DetectedActivity> detectedActivities = (ArrayList) result.getProbableActivities();

    for (DetectedActivity activity : detectedActivities) {
        Log.e(TAG, msg: "Detected activity: " + activity.getType() + ", " + activity.getConfidence());
        broadcastActivity(activity);
    }
}
```

Figura 2.3.1. onHandleIntent

```
private void broadcastActivity(DetectedActivity activity) {
    Intent intent = new Intent(Constants.BROADCAST_DETECTED_ACTIVITY);
    intent.putExtra( name: "type", activity.getType());
    intent.putExtra( name: "confidence", activity.getConfidence());
    LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
}
```

Figura 2.3.2. broadcastActivity

El método broadcastActivity() (Figura 2.3.2) tiene como objetivo lanzar un Intent con el tipo de la actividad y el nivel de confianza.

Los Broadcast ayudan para registrarse y enviar transmisiones de Intents a objetos locales dentro de su proceso. Esto tiene una serie de ventajas sobre el envío de transmisiones con sendBroadcast (Intent):

- Sabe que los datos que está transmitiendo no abandonarán su aplicación, por lo que no debe preocuparse por la pérdida de datos privados.
- No es posible que otras aplicaciones envíen estas transmisiones a su aplicación, por lo que no debe preocuparse por tener agujeros de seguridad que puedan explotar.
- Es más eficiente que enviar una transmisión global a través del sistema.

Capítulo 3. Detección de actividad

Una vez expuestos los servicios en segundo plano que se van a ejecutar, es momento de comentar el código que ha sido empleado para la detección de la actividad, explicando los métodos utilizados.

3.1. HandleUserActivity

Esta función ha sido utilizada para implementar las actividades con la etiqueta (label) que se utiliza y de esta manera hacerle saber al usuario que actividad de todas se está realizando. Hay bastantes actividades físicas incluidas en este proyecto (Figura 3.1.1), en la práctica se ha observado que las que han resultado ser más exactas a la hora de conocer la actividad que se realiza son STILL (quieto), WALKING (andando), RUNNING (corriendo) y ON_VEHICLE (en vehículo). Aunque también ha resultado exitosa para ON_BICYCLE (en bicicleta) no ha terminado de ser del todo exacta, puesto que si se va a baja velocidad la aplicación puede considerar que vas corriendo, o incluso andando.

En esta función se edita el marcador del mapa para que nos muestre un marcador personalizado con la imagen redimensionada y un texto al pulsar sobre él, que nombra la actividad física que se está realizando en ese momento.

```
private void handleUserActivity(int type, int confidence) {
    String label = "Unknown";
    int icon = R.drawable.ic_still;

    int height = 100;
    int width = 100;

    //still
    BitmapDrawable bitmapdrawstill = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_still);
    Bitmap bs = bitmapdrawstill.getBitmap();
    Bitmap smallMarkerstill = Bitmap.createScaledBitmap(bs, width, height, filter: false);
    //walking
    BitmapDrawable bitmapdrawwalking = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_walking);
    Bitmap bw = bitmapdrawwalking.getBitmap();
    Bitmap smallMarkerwalking = Bitmap.createScaledBitmap(bw, width, height, filter: false);
    //running
    BitmapDrawable bitmapdrawrunning = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_running);
    Bitmap br = bitmapdrawrunning.getBitmap();
    Bitmap smallMarkerrunning = Bitmap.createScaledBitmap(br, width, height, filter: false);
    //on vehicle
    BitmapDrawable bitmapdrawvehicle = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_driving);
    Bitmap bv = bitmapdrawvehicle.getBitmap();
    Bitmap smallMarkervehicle = Bitmap.createScaledBitmap(bv, width, height, filter: false);
    //bike
    BitmapDrawable bitmapdrawbike = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_on_bicycle);
    Bitmap bb = bitmapdrawbike.getBitmap();
    Bitmap smallMarkerbike= Bitmap.createScaledBitmap(bb, width, height, filter: false);
    //unknown
    BitmapDrawable bitmapdrawu = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_unknown);
    Bitmap bu = bitmapdrawu.getBitmap();
    Bitmap smallMarkeru = Bitmap.createScaledBitmap(bu, width, height, filter: false);
    //tilting
    BitmapDrawable bitmapdrawt = (BitmapDrawable)getResources().getDrawable(R.drawable.ic_tilting);
    Bitmap bt = bitmapdrawt.getBitmap();
    Bitmap smallMarkert = Bitmap.createScaledBitmap(bt, width, height, filter: false);
```

Figura 3.1.1. handleUserActivity 1

```
switch (type) {
    case DetectedActivity.IN_VEHICLE: {
        label = "In Vehicle";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkervehicle));
        marker.setTitle("Driving");
        break;
    }
    case DetectedActivity.ON_BICYCLE: {
        label = "On Bicycle";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkerbike));
        marker.setTitle("On Bicycle");
        break;
    }
    case DetectedActivity.ON_FOOT: {
        label = "On Foot";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkerwalking));
        marker.setTitle("On foot");
        break;
    }
    case DetectedActivity.RUNNING: {
        label = "Running";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkerrunning));
        marker.setTitle("Running");
        break;
    }
    case DetectedActivity.STILL: {
        label = "Still";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkerstill));
        marker.setTitle("Still");
        break;
    }
    case DetectedActivity.TILTING: {
        label = "Tilting";
        marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkert));
        marker.setTitle("Tilting");
        break;
    }
```

Figura 3.1.2. handleUserActivity 2

```
        case DetectedActivity.WALKING: {
            label = "walking";
            marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkerwalking));
            marker.setTitle("Walking");
            break;
        }
        case DetectedActivity.UNKNOWN: {
            label = "Unknown";
            marker.setIcon(BitmapDescriptorFactory.fromBitmap(smallMarkeru));
            marker.setTitle("Unknown");
            break;
        }
    }


    Log.e(TAG,  msg: "User activity: " + label + ", Confidence: " + confidence);

    if (confidence > Constants.CONFIDENCE) {
        txtActivity.setText(label);
    }

}
```

Figura 3.1.3. handleUserActivity 3

3.2. Imágenes

La finalidad de este apartado del proyecto consiste en exponer las imágenes que se muestran al usuario de la aplicación para conocer la actividad que está realizando, ya sea andar, correr o ir en un vehículo a motor.

Figura 3.2.1. Andando

Figura 3.2.2. Desconocido

Figura 3.2.4. En bicicleta

Figura 3.2.5. Parado

Figura 3.2.6. Corriendo

Figura 3.2.7. En vehículo

Capítulo 4. Mapas

Este capítulo tiene como objetivo explicar cómo se ha usado la API de Google Maps y la forma de implementarla en el proyecto. El resultado es un fragmento Fragment en el cual se muestra el mapa de Google con un marcador el cual se posiciona en nuestra ubicación.

4.1. Google Maps API

Para añadir mapas en nuestro proyecto tenemos que hacer uso de la API de Google Maps, la cual nos permite incorporar un mapa en una actividad como un fragmento con una simple instrucción XML. Se puede editar el mapa para que tenga vistas en 3D, vista satélite, de terreno e híbrido. En nuestro caso se usará el modelo por defecto de Google Maps. Es posible mediante esta API incorporar marcadores, polilíneas para crear rutas, personalizar la vista y las propiedades de cámara del mapa, añadir Street View, etc.

Para usar Google Maps necesitamos configurar los servicios SDK de Google Play en el proyecto a desarrollar. Los pasos para la configuración son los siguientes:

Debemos añadir los servicios de Google Play a Android Studio mediante la actualización del SDK manager. En nuestro caso, añadimos a nuestro Gradle lo siguiente: Figura 4.1.1.

```
implementation 'com.google.android.gms:play-services:12.0.1'
implementation 'com.google.android.gms:play-services-location:12.0.1'
implementation 'com.google.android.gms:play-services-maps:12.0.1'
```

Figura 4.1.1. Servicios Google Play

De esta manera implementamos los servicios de Google Play y podemos usarlos en nuestro proyecto.

Una tarea importante a realizar es conseguir una clave de Google Maps API. Esto es necesario para que nuestra aplicación pueda acceder a los servidores de Google Maps. La clave la podemos conseguir de manera gratuita (como se ha comentado al principio del documento, Android es una plataforma libre para poder ser desarrollada por todo el mundo).

Para conseguir la clave tendremos que ir a la plataforma en la nube de Google (https://console.cloud.google.com) y seguir las instrucciones para crear el proyecto, crear nuestra clave y una vez conseguida la clave copiarla en el archivo XML de nuestro proyecto.

4.2. Fragment

Para mostrar el mapa haremos uso de SupportMapFragment. Un "fragment", o fragmento, representa una parte de la interfaz de usuario en una actividad. Pueden combinarse varios fragmentos entre sí en una misma actividad para crear un multipanel. Se puede definir un fragmento como una sección modular de una actividad que tiene su propio comportamiento, que recibe eventos de entrada y que se puede agregar o quitar de la actividad mientras esta se esté ejecutando.

Hay que tener en cuenta que, a la hora de incorporar un fragmento a una actividad, su funcionamiento se ve afectado por el funcionamiento de esta, pues si esta pausada, también lo estará el fragmento incorporado.

El fragmento del mapa que usaremos se refleja en el Layout tal como se ve en la Figura 4.2.1.

```
<fragment
    android:id="@+id/Map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    tools:context=".MapsActivity"


    />
```

Figura 4.2.1. Layout Fragment

Por otro lado, será incluido en la clase principal del proyecto como se muestra en la Figura 4.2.2:

```
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.Map);
assert mapFragment != null;
mapFragment.getMapAsync( onMapReadyCallback: this);
mo = new MarkerOptions().position(new LatLng( v: 0,  v1: 0)).title("ESTAS AQUÍ");
```

Figura 4.2.2. SupportMapFragment

Hemos utilizado la clase pública extendida de fragment *SupportMapFragment* que nos permite implementar el mapa en la actividad a realizar y conocer nuestra ubicación mediante un marcador en el que, si pulsamos sobre él, nos aparecerá un cuadro de texto que ponga (“ESTAS AQUÍ”).

Capítulo 5. Ubicación

En este capítulo se va a explicar cómo se ha implementado la función que muestra la ubicación exacta del dispositivo en todo momento. Hay más detalles sobre la forma de obtener la ubicación en el apartado 2.2.

En esta parte del proyecto se crean las clases pertinentes para que en pantalla se muestre la ubicación en tiempo real del dispositivo, actualizándose cada cierto tiempo, situando esa ubicación en el mapa mediante un marcador y haciendo uso de los proveedores de localización necesarios.

5.1. LocationManager

Esta clase nos proporciona acceso al sistema de los servicios de localización. Estos servicios nos permiten obtener unas actualizaciones periódicas de la localización geográfica del dispositivo.

Hay que tener en cuenta que para todos los métodos se requiere el uso de los permisos de localización, de los que se hablará más detalladamente en el capítulo 7.

LocationManager es la clase principal por la que la aplicación podrá acceder a los servicios de localización de Android. Al igual que otros servicios, se puede obtener una referencia llamando al método getSystemService(). Para que reciba la localización la incluimos en el método onCreate().

Además, es necesario verificar que el proveedor de la localización está activado. Algunos proveedores de localización, como el GPS, pueden ser desactivados en el menú de Ajustes del dispositivo. Para poder comprobar si el proveedor de localización deseado está actualmente en activo se hace un llamamiento mediante el método isProviderEnabled().

Podemos fijarnos en nuestro proyecto que se escogen dos proveedores de localización: GPS_PROVIDER y NETWORK_PROVIDER.

La mayoría de los dispositivos que poseen Android pueden recibir actualizaciones de ubicación a través de múltiples funciones, que se resuelven en la aplicación como objetos LocationProvider. Los proveedores de localización pueden llegar a tener diferentes

características de rendimiento en lo referente a precisión. En general, un proveedor de localización con mayor precisión requiere un tiempo de preparación más largo que el basado en la red, lo cual es menos preciso.

Depende de cómo se use la aplicación, se debe elegir un proveedor de localización específico, o múltiples proveedores, en base a algunas compensaciones por similitud. De otra manera, se pueden proporcionar ciertos criterios de entrada como puede ser la precisión, requisito de energía, etc. De esta forma, se deja que Android decida el proveedor de localización más propicio conforme a la situación ofrecida. Se hace saber también, que los criterios pueden llegar a no resolverse con ningún proveedor de localización por lo que puede devolver un valor nulo en ese caso.

```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

if (!isPermissionGranted()) {
    requestPermissions(PERMISSIONS, PERMISSION_ALL);
} else requestLocation();
if (!isLocationEnabled())
    showAlert();

txtLat = findViewById(R.id.textView1);

locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this,
        Manifest.permission.ACCESS_COARSE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
    // TODO: Consider calling
    //    ActivityCompat#requestPermissions

    return;
}
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTime: 0,  minDistance: 10,  listener: this);
```

Figura 5.1.1. LocationManager 1

```
private boolean isPermissionGranted() {
    if (checkSelfPermission(android.Manifest.permission.ACCESS_COARSE_LOCATION)
            == PackageManager.PERMISSION_GRANTED || checkSelfPermission(android.Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
        Log.v( tag: "mylog",  msg: "Permission is granted");
        return true;
    } else {
        Log.v( tag: "mylog",  msg: "Permission not granted");
        return false;
    }
}

private void requestLocation() {
    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.ACCURACY_FINE);
    criteria.setPowerRequirement(Criteria.POWER_HIGH);
    String provider = locationManager.getBestProvider(criteria,  enabledOnly: true);
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this,
            Manifest.permission.ACCESS_COARSE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions

        return;
    }
    locationManager.requestLocationUpdates(provider,  minTime: 1000,  minDistance: 10,  listener: this);
}
private boolean isLocationEnabled() {
    return locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
            locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
}
```

Figura 5.1.2. LocationManager 2

Capítulo 6. Base de datos

Expuesto cómo vamos a utilizar los servicios, la detección de la actividad que se está realizando y el posicionamiento de nuestra ubicación en el mapa de Google, necesitamos un lugar donde guardar la información. Es por esto que se ha creado una base de datos mediante SQLite para este proyecto, guardando de esta manera nuestra ubicación en una lista ListView.

SQLite es un motor de base de datos SQL de código abierto, ligero, de configuración simple y sin servidor, que se caracteriza por almacenar información persistente de forma sencilla. Gracias a estas características se diferencia de otros gestores de bases de datos. Ya de por sí, SQLite implementa una serie de librerías, que son las encargadas de la gestión, y no ejecuta procesos para administrar la información. Al ser código abierto, los archivos de compilación y las instrucciones de escalabilidad, se encuentran disponibles para toda la comunidad de desarrolladores.

Para reducir el tiempo que tarda la señal en comenzar a funcionar (latencia), se cuenta con una biblioteca SQLite que es llamada dinámicamente a través de funciones simples y que los programas de la aplicación utilizan. De la misma forma implementa el estándar SQL-92 y usa un sistema poco común para sistemas de administración de bases de datos que son compatibles con SQL.

SQLite es perfecto a la hora de consultar y almacenar datos de una forma estructurada. La aplicación, en lugar de leer el archivo en su totalidad, solo tendrá que cargar los datos necesarios y mantener un análisis en la memoria. Por lo tanto, el tiempo de inicio y el consumo de la memoria se reducen.

El contenido se va actualizando de manera continua para que no se pierda nada en caso de un fallo de energía o algún bloqueo. A este contenido se puede acceder mediante consultas SQL, por lo que se reduce la complejidad del código de la aplicación.

Debido a todo lo comentado anteriormente, SQLite permite que el desarrollo sea más simple, convirtiéndolo en una tecnología muy práctica para los dispositivos móviles.

Para la creación de nuestra base de datos se ha añadido una clase auxiliar para gestionar asi las bases de datos y las versiones. Esta clase cuenta con los métodos onCreate para la creación de la tabla, onUpgrade para el control de las versiones, insertData para añadir los datos a la base de datos y por último viewData para poder visualizarlos en pantalla.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_LATITUD = "Latitud.db";
    private static final String DB_TABLE = "Latitud_Table";

    private static final String ID = "ID";
    private static final String LATITUD = "LATITUD";


    private static final String CREATE_TABLE = "CREATE TABLE "+ DB_TABLE+" (" +
            ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "+
            LATITUD+ " TEXT "+ ")";

    public DatabaseHelper(Context context) { super(context, DB_LATITUD,  factory: null, version: 2); }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) { sqLiteDatabase.execSQL(CREATE_TABLE); }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {

        if (oldVersion == 1 && newVersion == 2) {
            sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
        }
        onCreate(sqLiteDatabase);
    }

    //create method to insert data
    public boolean insertData(String txtLat){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(LATITUD , txtLat);

        long result = db.insert(DB_TABLE,  nullColumnHack: null,contentValues);
        return result != -1;
    }

    //create method to view data
    public Cursor viewData(){
        SQLiteDatabase db = this.getReadableDatabase();
        String query = "Select * from "+ DB_TABLE;
        Cursor cursor = db.rawQuery(query,  selectionArgs: null);

        return cursor;
    }
```

Figura 6.1. DatabaseHelper

6.1. ListView

Una ListView es una subclase de la clase AdapterView, la cual muestra una lista vertical de elementos. La ventaja de utilizar esta clase es que se puede añadir, modificar o eliminar elementos de la lista de forma sencilla. En nuestro caso, cada vez que la aplicación detecta un cambio en la ubicación (actividad) ésta incorpora esos datos a la tabla. Los AdapterView requieren un objeto que sirva como puente de los adapterviews y la información que se quiere mostrar a los que llamaremos simplemente adapters. El adapter da acceso a los archivos de información y es responsable de mostrar una vista (o view) para cada ítem de información.

El código más relevante del proyecto, con respecto a la base de datos, lo encontramos en las siguientes figuras: Figura 6.1.1 y Figura 6.1.2, en las que podemos observar cómo son llamados los métodos y cómo son utilizados para recoger los datos e incorporarlos a la ListView.

```
db = new DatabaseHelper( context: this);
listItem = new ArrayList<>();
userlist =findViewById(R.id.users_list);

viewData();

userlist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        String text = userlist.getItemAtPosition(i).toString();
        Toast.makeText( context: MainActivity.this, text: ""+text, Toast.LENGTH_SHORT).show();

    }
});
```

Figura 6.1.1. AdapterView

```
private void viewData() {
    Cursor cursor = db.viewData();

    if (cursor.getCount() == 0){
        Toast.makeText( context: MainActivity.this, text: "no data to view", Toast.LENGTH_SHORT).show();
    } else {
        while (cursor.moveToNext()){
            listItem.add(cursor.getString( columnIndex: 1));
        }
        adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, listItem);
        userlist.setAdapter(adapter);
    }

}
```

Figura 6.1.2. viewData()

Capítulo 7. Permisos y Manifest

7.1. Permisos

El objetivo de un permiso es proteger la privacidad del usuario del dispositivo Android. Todas las aplicaciones Android deben pedir una serie de permisos para acceder a datos sensibles para el usuario como los contactos, internet o la ubicación. Dependiendo de cómo este configurado, el sistema puede conceder el permiso automáticamente mandar una petición al usuario para que se conceda.

El rasgo principal de diseño en la arquitectura de seguridad de Android consiste en que, por defecto, ninguna aplicación tiene permiso para realizar ninguna operación que pueda impactar en otra aplicación, sistema o al mismo usuario. Esto incluye leer datos privados del usuario o leer y escribir en los archivos de otra aplicación, entre otros.

Para declarar en la aplicación cualquier permiso se tiene que incluir el elemento <uses-permission> en el archivo manifest de la aplicación y se debe poner como elemento secundario del elemento <manifest>. Esto es necesario para cualquier versión de Android.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.vicentepina.reconocimientoactividad">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"/>
```

Figura 7.1.1. Permisos AndroidManifest

Como observamos en la Figura 7.1.1 nuestro proyecto precisa de 6 permisos.

- ACCESS_FINE_LOCATION: permite a la API usar el WiFi, los datos del dispositivo o ambos para determinar la localización de este.
- INTERNET: este permiso se usa para que la aplicación tenga acceso a Internet.

- WRITE_EXTERNAL_STORAGE: este permiso se usa para poder realizar operaciones con los datos de la base de datos a utilizar en el proyecto.
- FOREGROUND_SERVICE: sirve para poder empezar un servicio en primer plano que es usado para el reconocimiento de la actividad.
- ACCESS_COARSE_LOCATION: este permiso sirve para dar una localización aproximada, a diferencia del primer permiso comentado que lo hace de manera precisa.
- ACTIVITY_RECOGNITION: este permiso es necesario para poder saber que actividad se está realizando.

Si en la aplicación es necesario el uso de un permiso peligroso, debe verificarse si tiene ese permiso cada vez que se quiera realizar la operación que lo requiera. A partir de la versión Android 6.0 se pueden revocar estos permisos desde cualquier otra aplicación aunque esté orientada a un nivel de API más bajo.

Para comprobar que la aplicación tiene un permiso, se llama al método ContextCompat.checkSelfPermission() y si tiene el permiso, este método muestra "PERMISSION_GRANTED" (permiso concedido). Si por el contrario no lo tiene, se muestra "PERMISSION_DENIED" (permiso denegado) y la aplicación debe solicitarlo al usuario de forma explícita.

En el caso de que la aplicación reciba "PERMISSION_DENIED" de checkSelfPermission(), debe solicitarse permiso al usuario y Android proporciona formas de hacerlo, por ejemplo mediante requestPermission().

Si la aplicación no precisa del permiso necesario, esta deberá llamar a uno de los métodos requestPermissions() para solicitar los permisos que correspondan. La aplicación transmite el permiso y junto a él, un entero de código de solicitud que debe especificarse con el fin de identificar dicha solicitud del permiso. Esta solicitud se muestra de manera inmediata y una vez que se responda a la solicitud, el sistema llamará al método de devolución de llamada de la aplicación con el resultado y transmitirá el mismo código del que se ha comentado anteriormente.

Cuando es respondida esta solicitud de permiso por el usuario, el sistema llama al método onRequestPermissionsResult() de la aplicación y transmite la respuesta del usuario. Dicha respuesta tiene como objetivo anular el método para saber si se concedió el permiso solicitado. Lo que se devuelve de la llamada; es recibido por el mismo código de solicitud que se transmite a requestPermissions()

7.2. Android Manifest

Cualquier proyecto de Android debe tener un archivo llamado específicamente AndroidManifest.xml en la raíz del proyecto. Este archivo proporciona la información esencial de la aplicación para el uso de las herramientas de Android Studio, Google Play y el propio sistema operativo.

En el manifest se debe declarar el nombre del paquete de la aplicación para que sea determinada la ubicación de las entidades del código a la hora de la compilación del proyecto. Deben declararse también los componentes de la aplicación que incluyen a las actividades, servicios, receptores y proveedores de contenido. Además, es necesario definir las propiedades básicas de los componentes, tales como el nombre de la clase Java, las configuraciones de dispositivos que se pueden manejar e intents que describen como se inicia el componente. También se añadirán al manifest los permisos necesarios que se han comentado con anterioridad.

Este archivo es generado automáticamente por Android Studio y la mayoría de elementos esenciales se agregarán a medida que se vaya compilando la aplicación.

Es importante volver a señalar que la clave para el uso de la API de Google Maps se encuentra declarada en este archivo.

7.2.1. Nombre del paquete e ID de la aplicación

Este atributo es requerido por el elemento raíz del manifest para el nombre del paquete de la aplicación. Suele coincidir con la estructura del directorio del proyecto. En nuestro proyecto se muestra: "com.example.vicentepina.reconocimientoactividad".

A la hora de la compilación se utiliza el atributo package con una doble finalidad. Una es aplicar ese nombre como espacio de nombre para la clase R (R.java) que se usa para acceder a los recursos de la aplicación. La segunda es para poder resolver cualquier nombre de clase relativo en el archivo manifest.

El nombre del paquete tiene que coincidir con el del paquete básico del proyecto, en el que se pueden encontrar las actividades y cualquier otro tipo de código utilizado de la

aplicación. Es posible tener otros subpaquetes en el proyecto, pero estos deberán importar la clase R.java con el espacio de nombres del paquete principal.

7.2.2. Componentes de la aplicación

Se necesitan declarar todos los componentes de la aplicación que sean creados como elementos XML en el manifest. Si no se declaran todos los componentes, el sistema no lo podrá iniciar. La manera de declararlos es la siguiente:

- <activity>: para la subclase de Activity.
- <service>: para la subclase de Service.
- <receiver>: para la subclase de BroadcastReceiver.
- <provider>: para la subclase de ContentProvider.

7.2.3. Filtros de intents

Un componente puede tener más de un filtro de intents y cada uno de ellos pueden describir una capacidad diferente que posee ese componente.

7.2.4. Iconos y etiquetas

Estos atributos son utilizados para mostrar un pequeño icono y etiqueta de texto. Los que son establecidos en un elemento superior se convierten en los predeterminados para los inferiores, por lo que si se lo añadimos en el elemento <application> serán los predeterminados para toda la aplicación, incluyéndose las actividades. Son establecidos por un <intent-filter> y son mostrados al usuario si ese componente se presenta para cumplir el intent.

7.2.5. Elementos

Los elementos <manifest> y <application> son los únicos requeridos en el archivo AndroidManifest.xml y solo deben aparecer una vez. El resto de elementos pueden aparecer más de una vez o ninguna. Los elementos se pueden alternar al gusto del desarrollador, pues no tiene un orden concreto a la hora de organizarse, aunque existen dos excepciones. El elemento <application> debe ser el último dentro del <manifest> y el elemento <activity-alias> obligatoriamente seguirá a <activity>.

7.2.6. Atributos

A excepción de algunos atributos, todos los nombres de atributos deben comenzar con el prefijo “android:”. Ningún atributo es obligatorio técnicamente, aunque varios deberán ser especificados para que un elemento cumpla con su propósito.

Capítulo 8. Pantallas y Layouts

En este capítulo se mostrarán las pantallas creadas para la aplicación, así como sus respectivos layouts.

8.1. Funcionamiento de la aplicación

La aplicación, al abrirse, muestra una pantalla de inicio, como se muestra en la *Figura 8.1.2*, la cual tiene el título del proyecto, el logo de la Universidad Miguel Hernández de Elche y dos botones que nos llevan a la pantalla principal y a una pantalla de información sobre la aplicación.

Las transiciones de la pantalla vienen mostradas por la *Figura 8.1.1*:

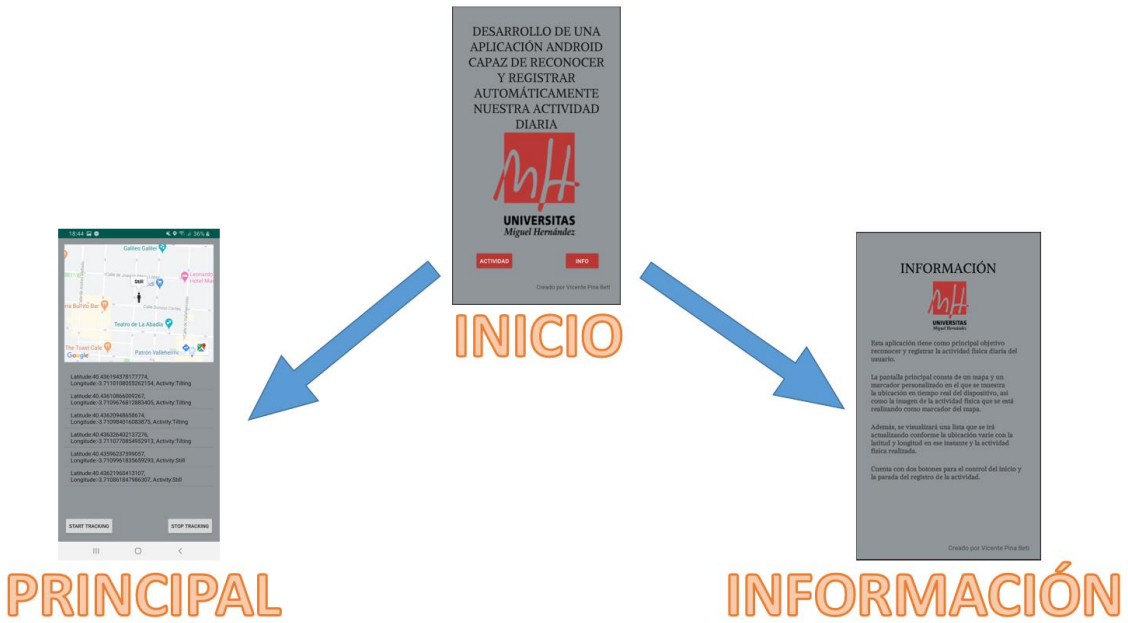

Figura 8.1.1. Funcionamiento Pantallas.

Una vez en la pantalla de *Inicio*, al presionar sobre el botón *Actividad* nos lleva a la pantalla principal, la cual se puede ver con detalle en la *Figura 8.1.3*. La pantalla consta de un fragmento de mapa, el cual posee un marcador personalizado con la imagen de la actividad física que se realiza. La pantalla contiene a su vez un listado con el registro de la localización del dispositivo móvil y de un texto nombrando la actividad que se está realizando en ese momento.

Por último, si desde la pantalla de *Inicio* presionamos en el botón *Info*, nos lleva a una pantalla de información donde se explica el funcionamiento de la aplicación. Esta pantalla se puede ver con más detalle en la *Figura 8.1.4*.

Figura 8.1.2. Pantalla Inicio.

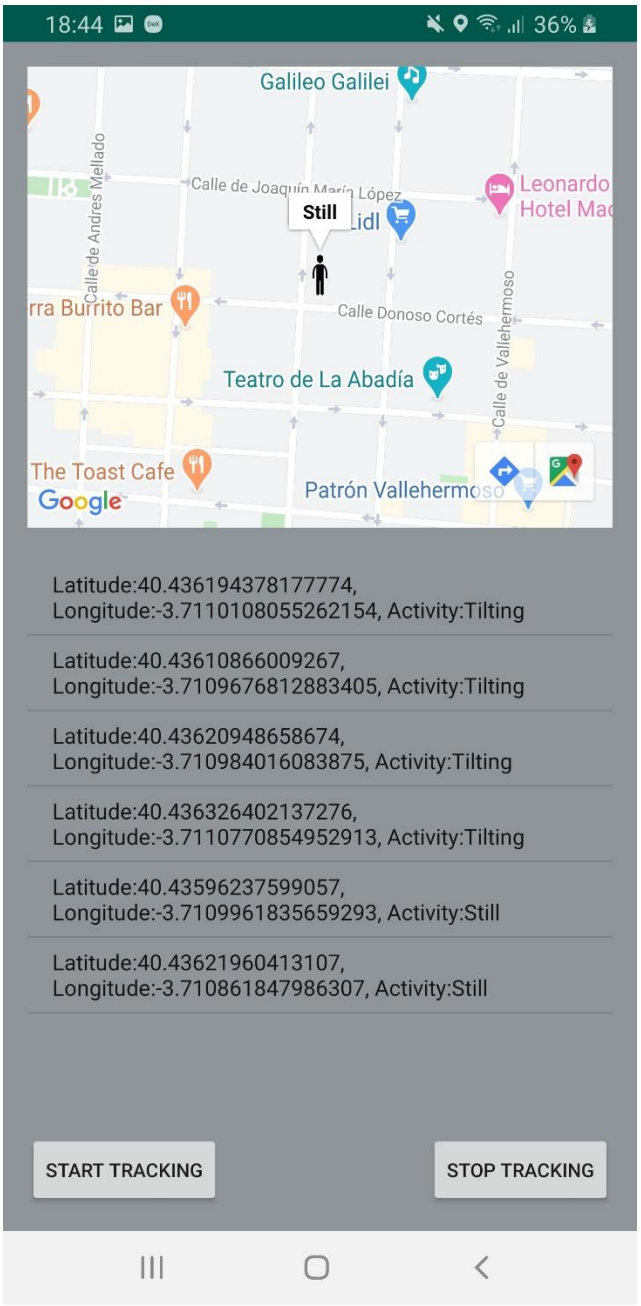

Figura 8.1.3. Pantalla Principal.

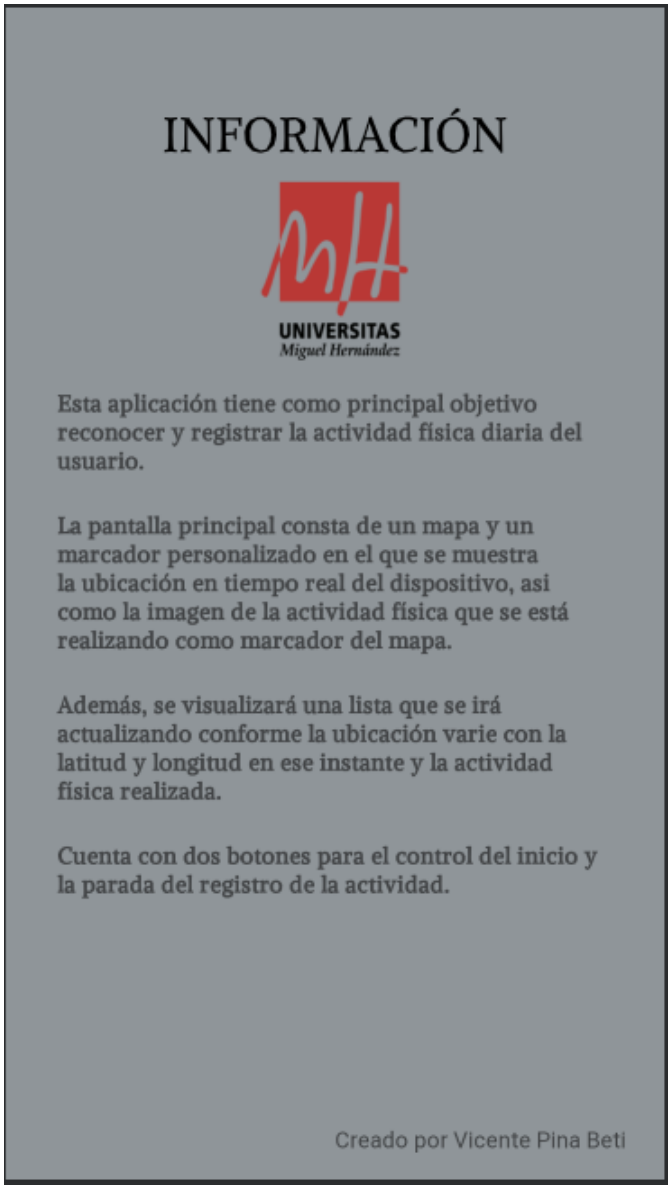

Figura 8.1.4. Pantalla Info.

8.2. Layouts

A continuación se muestran los layouts de las pantallas para una vista más específica a ojos del desarrollador.

8.2.1. Pantalla Inicio

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#8F9599"
    android:padding="16dp"
    tools:context=".Inicio">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/img_activity"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:tint="#606060" />

    </LinearLayout>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:fontFamily="@font/alike_angular"
            android:text="DESARROLLO DE UNA APLICACIÓN ANDROID CAPAZ DE RECONOCER Y REGISTRAR AUTOMÁ
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textSize="30sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.031"
            tools:layout_conversion_absoluteHeight="34dp"
            tools:layout_conversion_absoluteWidth="326dp" />
```

Figura 8.2.1.1. Layout Inicio.

```
<Button
    android:id="@+id/button2"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:backgroundTint="#B83835"
    android:text="ACTIVIDAD"
    android:textColorHighlight="#00E9DFDF"
    android:textColorLink="#00B43333"
    android:textCursorDrawable="#B83835"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.112"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.908"
    tools:layout_conversion_absoluteHeight="48dp"
    tools:layout_conversion_absoluteWidth="379dp" />

<Button
    android:id="@+id/button3"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:backgroundTint="#B83835"
    android:text="INFO"
    android:textColorHighlight="#00E9DFDF"
    android:textColorLink="#00B43333"
    android:textCursorDrawable="@color/colorAccent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.91"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.908"
    tools:layout_conversion_absoluteHeight="48dp"
    tools:layout_conversion_absoluteWidth="379dp" />
```

Figura 8.2.1.2. Layout Inicio.

```
        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:text="Creado por Vicente Pina Beti"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="1.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.987" />

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="490dp"
            android:layout_height="482dp"
            app:srcCompat="@mipmap/ic_launcher_foreground_umh"
            tools:layout_editor_absoluteX="-56dp"
            tools:layout_editor_absoluteY="182dp"
            tools:ignore="MissingConstraints" />

    </android.support.constraint.ConstraintLayout>

</RelativeLayout>
```

Figura 8.2.1.3. Layout Inicio.

8.2.2. Pantalla Principal

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#8F9599"
    android:padding="16dp"
    tools:context=".MainActivity">
    <fragment
        android:id="@+id/Map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        tools:context=".MapsActivity" />
    <ListView
        android:id="@+id/users_list"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_alignParentTop="true"
        android:layout_marginTop="323dp" />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:orientation="vertical">
        <TextView
            android:id="@+id/txt_activity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="500dp"
            android:textAllCaps="true"
            android:textColor="@color/colorPrimary"
            android:textSize="18sp"
            android:textStyle="bold" />
    </LinearLayout>
    <Button
        android:id="@+id/btn_start_tracking"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:text="Start Tracking" />
    <Button
```

Figura 8.2.2.1. Layout Principal.

```
    <Button
        android:id="@+id/btn_stop_tracking"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:text="Stop Tracking" />
</RelativeLayout>
```

Figura 8.2.2.2. Layout Principal.

8.2.3. Pantalla Info

```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#8F9599"
    android:padding="16dp"
    tools:context=".Info">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginVertical="40dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="66dp"
        android:layout_marginEnd="16dp"
        android:background="#8F9599"
        android:fontFamily="@font/alike_angular"
        android:lineSpacingExtra="8sp"
        android:text="INFORMACIÓN"
        android:textAlignment="center"
        android:textAllCaps="true"
        android:textColor="#000000"
        android:textSize="30sp"
        android:typeface="sans"
        tools:background="#8F9599" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="16dp"
        android:layout_marginTop="220dp"
        android:layout_marginEnd="16dp"
        android:fontFamily="@font/alike_angular"
        android:text="Esta aplicación tiene como principal objetivo reconocer y registrar la actividad
        android:textStyle="bold" />
```

Figura 8.2.3.1. Layout Info.

```
<TextView
    android:id="@+id/textView7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView6"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="16dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="16dp"
    android:fontFamily="@font/alike_angular"
    android:text="La pantalla principal consta de un mapa y un marcador personalizado en el que
    android:textStyle="bold" />

<TextView
    android:id="@+id/textView8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView7"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="16dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="16dp"
    android:fontFamily="@font/alike_angular"
    android:text="Además, se visualizará una lista que se irá actualizando conforme la ubicación
    android:textStyle="bold" />

<TextView
    android:id="@+id/textView9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView8"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
    android:layout_marginStart="16dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="16dp"
    android:fontFamily="@font/alike_angular"
    android:text="Cuenta con dos botones para el control del inicio y la parada del registro de
    android:textStyle="bold" />
```

Figura 8.2.3.2. Layout Info.

```
    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="232dp"
        android:layout_height="207dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="75dp"
        android:layout_marginTop="45dp"
        android:layout_marginEnd="75dp"
        app:srcCompat="@mipmap/ic_launcher_foreground_umh" />

    <TextView
        android:id="@+id/textView10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8.dp"
        android:text="Creado por Vicente Pina Beti" />
</RelativeLayout>
```

Figura 8.2.3.3. Layout Info.

Capítulo 9. Conclusiones y futuras mejoras

La primera conclusión a la que se llega con este proyecto es que la aplicación creada resulta muy útil, especialmente para el desarrollador. Esto se debe a que la aplicación muestra y notifica al usuario lo que ocurre en cada momento en un primer plano cómo la muestra continua de la posición en latitud y longitud del dispositivo mientras se realiza la actividad o si se añaden datos a la tabla o no mediante unos mensajes emergentes.

En segundo lugar, la aplicación creada siguiendo el método descrito a lo largo de este trabajo es capaz de reconocer la actividad del usuario y de mostrar la localización del dispositivo a la vez que tiene implementados los servicios que hacen que trabaje en segundo plano. Esta característica hace la aplicación muy práctica a la hora de poder usar otras aplicaciones mientras se realiza la actividad. Por ejemplo, es posible que la aplicación creada esté abierta y funcionando a la vez que se está, mediante otras aplicaciones, escuchando música, mandando mensajes, o navegando por internet.

Por último, cabe señalar que la aplicación creada en este trabajo de fin grado puede mejorarse a partir de la estructura creada en este proyecto y de todas las funciones que es capaz de realizar en este momento. Para futuros trabajos, sería interesante que la aplicación fuera capaz de crear rutas efectuadas durante el día por el usuario, mostrarlas en una tabla y que al pulsar sobre cualquier ruta se muestre una pantalla con el mapa, los puntos iniciales y finales y la ruta realizada mediante una polilínea. También resultaría atractivo cambiar la interfaz, creando animaciones para las actividades, para los cambios de pantalla.

Capítulo 10. Bibliografía

Almacenamiento de Datos (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/training/data-storage/room#java

Fragments (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/guide/components/fragments?hl=es-419

Obtener la localización (en Stack Overflow. Foro para desarrolladores).

URL: https://stackoverflow.com/questions/33415033/getting-current-location-in-android-studio-app

Almacenar y buscar datos (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/training/search/search

Coding with Mitch.

URL: https://codingwithmitch.com/

Organización mundial de la salud. Web oficial.

URL: https://www.who.int/dietphysicalactivity/factsheet_recommendations/es/

Vitónica. Sección sobre estilo de vida de Xataka.

URL: https://www.vitonica.com/wellness/muevete-tu-dia-a-dia-asi-impacta-actividad-fisica-diaria-beneficio-tu-salud

Nexora. Empresa de Diseño y Desarrollo Web en Toledo.

URL: https://nexora.es/las-apps-nos-cambiado-la-vida/

Código Espagueti. Portal de noticias.

URL: https://codigoespagueti.com/noticias/mapa-plataformas-moviles-mas-usadas/

Portal Kantar.

URL: https://www.kantarworldpanel.com/global/smartphone-os-market-share/intro

Introducción Android Studio (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/studio/intro

Servicios (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/guide/components/services?hl=es-419

Devolver localización (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/training/location/retrieve-current

Recibir actualización de ubiacación (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/training/location/receive-location-updates

Transiciones localización (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/guide/topics/location/transitions

BroadcastManager (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager

Maps (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/training/maps

Medium Magazine.

URL: https://medium.com/@debuggingisfun/android-pie-foreground-service-340b98e9f198

Master en Desarrollo de Aplicaciones Android. Web de la Universidad Politécnica de Valencia.

URL: http://www.androidcurso.com/index.php

Location Manager (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/reference/android/location/LocationManager

El Español. Sección “El androide Libre”.

URL: https://elandroidelibre.elespanol.com/2013/11/desarrollando-android-3-centrando-la-localizacin-y-marcadores.html

Toasts (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/guide/topics/ui/notifiers/toasts

Android Manifest (en Android Developers. Web oficial de Android para desarrolladores).

URL: https://developer.android.com/guide/topics/manifest/manifest-intro

Anexo I. AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.vicentepina.reconocimientoactividad">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="TFG"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="GoogleAppIndexingWarning">
        <activity android:name=".Inicio"/>

        <uses-library
            android:name="org.apache.http.legacy"
            android:required="false" />

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBvKjPlNnIzl94KgvnXhiXRNYGQ6TsYjvQ" />

        <activity
            android:name=".MainActivity"
            android:label="TFG"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MapsActivity"
            android:label="Map">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
```

Figura Anexo I. AndroidManifest.xml

```
        <service
            android:name=".DetectedActivitiesIntentService"
            android:exported="false" />
        <service android:name=".BackgroundDetectedActivitiesService" />
        <service android:name=".BackgroundService" />
    </application>

</manifest>
```

Figura Anexo I. AndroidManifest.xml

Anexo II. Toast

Este método toma tres parámetros: el valor Context de la aplicación, el mensaje de texto y la duración del aviso. Devuelve un objeto de aviso correctamente inicializado. Se muestra el aviso con show(). Una notificación de aviso normal, aparece en la parte inferior de la pantalla del teléfono móvil, centrada horizontalmente. Se puede cambiar esta localización con el método setGravity(int, int, int). Este método acepta tres parámetros: una constante Gravity, un eje de desplazamiento x y un eje de desplazamiento y.

Anexo III. Logcat

La herramienta Android Monitor incluye un monitor de logcat en el que se muestran mensajes para la depuración. En él se presentan mensajes del sistema, como, por ejemplo, mensajes que se pueden agregar a la aplicación mediante la clase Log. Se muestran mensajes en tiempo real y también se conserva un historial para que se puedan ver mensajes antiguos.

Con el fin de ofrecer solo información de interés, pueden crearse filtros, modificar la cantidad de información que se muestra en los mensajes, establecer niveles de prioridad, etc. De forma predeterminada, el monitor muestra el resultado de registros solo en relación con la aplicación que se haya ejecutado más recientemente.

Cuando se produce una excepción en la aplicación, el monitor de logcat hace aparecer un mensaje y luego un seguimiento asociado que contiene vínculos con el código. Esta función puede ayudar a solucionar errores y mejorar la operación de la aplicación.

A partir de la versión de Android Studio 2.2, la ventana “Run” también muestra mensajes de registro para la aplicación que está en ejecución. Hay que tener en cuenta que se puede configurar la visualización del monitor de logcat, pero no la ventana “Run”.

El monitor de logcat nos proporciona unos niveles distintos para el log, pueden ser ERROR (E), WARNING (W), INFORMATIVE (I), DEBUG (D), VERBOSE (V) o ASSERT (A).

- ERROR: se muestran problemas que generan errores. Este nivel debe usarse cuando ocurra algo que tendrá consecuencias visibles para el usuario y que no se podrá recuperar sin eliminar explícitamente algunos datos, desinstalar aplicaciones o reseteando el dispositivo por completo. Este nivel se muestra siempre.
- WARNING: se muestran posibles problemas que no son considerados por el momento errores. Este nivel trabaja como el anterior a excepción de que es

probable que pueda recuperarse sin implicar la perdida de datos realizando alguna acción como puede ser reiniciar el dispositivo.

- INFORMATIVE: se muestran mensajes de registro esperados para un uso regular. Debe usarse para indicar que ha pasado algo que puede resultar interesante a la mayoría de la Esta condición debe ser mostrada sólo por un módulo que sea considerado el más apropiado. Este nivel se muestra siempre.
- DEBUG: se muestran mensajes de registro de depuración que son útiles únicamente durante el desarrollo. Se debe usar para dar a conocer qué eventos están ocurriendo en el dispositivo que puedan ser relevantes para investigar y depurar comportamientos de la aplicación inesperados. Se mostrará únicamente lo necesario para obtener la información suficiente sobre qué está pasando.
- VERBOSE: Se muestran todos los mensajes de registro. Debe usarse para cualquier otra cosa. Solo se mostrará en las aplicaciones destinadas a depuración y debe estar dentro de un bloque if para permitir su desactivación.