



Universidad de Murcia

Facultad de Informática
2º cuatrimestre 4º de GIITI – 2022/2023
Computación Móvil

PROGRAMACIÓN EN ANDROID

PEDRO NICOLÁS GOMARIZ
pedro.nicolasg@um.es
48753907W

23/03/2023

Índice

1. Introducción.	1
2. Funcionalidad obligatoria.	1
2.1. MainActivity	1
2.2. MapsActivity	5
3. Funcionalidad opcional.	10
3.1. Opcional 1: Comparación con otra tecnología.	10
3.2. Opcional 2: Pantalla de análisis. AnalysisActivity.	11
4. Análisis teórico.	13
4.1. Recorrido 4G (LTE).	14
4.2. Recorrido 2G (GSM).	16
4.3. Comparativa entre recorridos.	17
5. Conclusión.	18
Referencias.	19

1. Introducción.

Este documento tiene como objetivo recoger las principales decisiones de diseño tomadas para la implementación de la aplicación para dispositivos móviles Android denominada **Signal Map** y desarrollada en la parte práctica de la asignatura Computación Móvil.

Dicha aplicación tiene por objetivo recabar información sobre la red móvil y mostrarla en tiempo real sobre un mapa interactivo (Google Map) permitiendo al usuario ver en todo momento, entre otras cosas, el nivel máximo de señal detectado por el terminal en distintos puntos mediante un código de colores para que sea más 'amigable', a que antena se encuentra conectado el dispositivo, el número de celdas que está detectando etc... También tendrá la posibilidad de definir etapas de modo que se pueda agrupar un conjunto de puntos para su posterior tratamiento.

Toda esta información será almacenada en el dispositivo permitiendo posteriormente analizar los resultados.

2. Funcionalidad obligatoria.

En este apartado se van a describir los detalles de implementación de las distintas actividades que cubren la funcionalidad básica requerida como parte de la práctica.

2.1. MainActivity.



Figura 1: MainActivity.

Esta es la pantalla principal de la aplicación y tiene como único propósito dar la bienvenida al usuario y en mi caso mediante dos botones darle a elegir a qué actividad se quiere dirigir (*MapsActivity* ó *AnalysisActivity*).

- **BOTÓN NUEVO RECORRIDO:** Al hacer click, se ejecuta la función *onClick_btn1_mainActivity* (*View v*) que a su vez llama a *createInfoDialog()* la cual se va a encargar de crear un cuadro de dialogo personalizado que tenemos definido en el fichero *info_dialog.xml* y que será lanzado mediante un *AlertDialog.Builder*[1].



Figura 2: Pop-up información recorrido.

```
1 AlertDialog.Builder builder = new AlertDialog.Builder(this);
2 LayoutInflater inflater = this.getLayoutInflater();
3 final View view = inflater.inflate(R.layout.info_dialog, null);
4 builder.setView(view);
5
6 AlertDialog dialog = builder.create();
7 dialog.show();
```

Se ha decidido implementar de esta manera, porque es una buena forma de crear un pop-up personalizable, ya que puedes crear el layout a tu gusto. En mi caso he añadido una imagen con un título, posteriormente un cuadro de texto para que el usuario introduzca el nombre que le quiere dar al recorrido que va a realizar y por último un *RadioGroup* con dos *RadioButton* para que tenga la opción de seleccionar la tecnología sobre la que quiere realizar el recorrido. Esta será la información que se le pasará a la actividad del mapa para en caso del

nombre usarlo como texto descriptivo en la propia actividad y además, a la hora de almacenar los datos del recorrido utilizarlo para darle nombre al fichero. Por otro lado, la tecnología se emplea para tomar los datos relacionados o de la red GSM o LTE, también en función de la tecnología almacenaremos unos datos o otros y emplearemos unas escalas u otras para definir la calidad de la señal, esto se verá más adelante.

Una vez el usuario seleccione los datos del recorrido (se comprueba que el nombre no este vacío y haya seleccionado uno y solo un *RadioButton*) haciendo click en el botón 'NUEVO RECORRIDO' ahora si, se crea la actividad del mapa, se le pasan los datos y se lanza.

```
1 // Map Activity con la info del user
2 Intent intent = new Intent(MainActivity.this, MapsActivity.class);
3 intent.putExtra("nombreRecorrido", nombreRecorrido);
4 intent.putExtra("tecnologia", tecnologia);
5 startActivityForResult(intent);
```

- **BOTÓN ANALIZAR RECORRIDO:** Al hacer click, se ejecuta la función *onClick_btn2_mainActivity (View v)* la cual se encarga de lanzar un File Picker que permitirá al usuario navegar por los distintos directorios del dispositivo y seleccionar uno y solo un fichero con extensión *.json*, este devolverá el path completo del archivo seleccionado, que se le pasará a la actividad de análisis y le servirá para poder leer el contenido del fichero y trabajar con el.

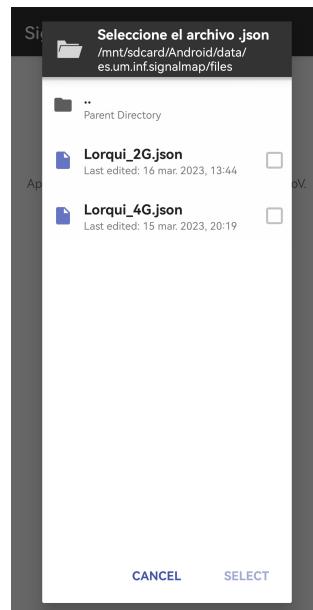


Figura 3: File Picker para AnalysisActivity.

En mi caso he decidido no usar el Gestor de Archivos de android, si no emplear un File

Picker público de GitHub <https://github.com/TutorialsAndroid/FilePicker>[2], en primer lugar porque da la apariencia de estar más integrado en la propia app, y además por la facilidad que este tiene de configuración, ya que el propio repositorio ya provee de un código de ejemplo.

```
1 public void onClick_btn2_mainActivity(View v) {
2     // Configuramos las propiedades del FilePicker
3     // https://github.com/TutorialsAndroid/FilePicker
4     DialogProperties properties = new DialogProperties();
5     properties.selection_mode = DialogConfigs.SINGLE_MODE;
6     properties.selection_type = DialogConfigs.FILE_SELECT;
7     properties.root = new File(DialogConfigs.DEFAULT_DIR);
8     properties.error_dir = new File(DialogConfigs.DEFAULT_DIR);
9     properties.offset = new File(DialogConfigs.DEFAULT_DIR);
10    properties.extensions = new String[]{"json"};
11    properties.show_hidden_files = false;
12    dialogFilePicker = new FilePickerDialog(MainActivity.this, properties);
13    dialogFilePicker.setTitle(getResources().getString(R.string.seleccion_recorrido));
14    dialogFilePicker.setDialogSelectionListener(new DialogSelectionListener() {
15        @Override
16        public void onSelectedFilePaths(String[] files) {
17
18            // Debe de haber uno y solo uno, obligatoriamente.
19            Log.d("DEBUG", "Seleccionado el fichero: " + files[0]);
20
21            // Abrimos la actividad de ANALISIS
22            Intent intent = new Intent(MainActivity.this, AnalysisActivity.class);
23            intent.putExtra("filepath", files[0]);
24            startActivity(intent);
25        }
26    });
27    dialogFilePicker.show();
28 }
```

Cabe recalcar que para poder acceder a los archivos del almacenamiento hay que contar con los permisos de almacenamiento del usuario por lo que hay que redefinir el método *onRequestPermissionsResult()* para solicitar y gestionar dichos permisos.

2.2. MapsActivity.

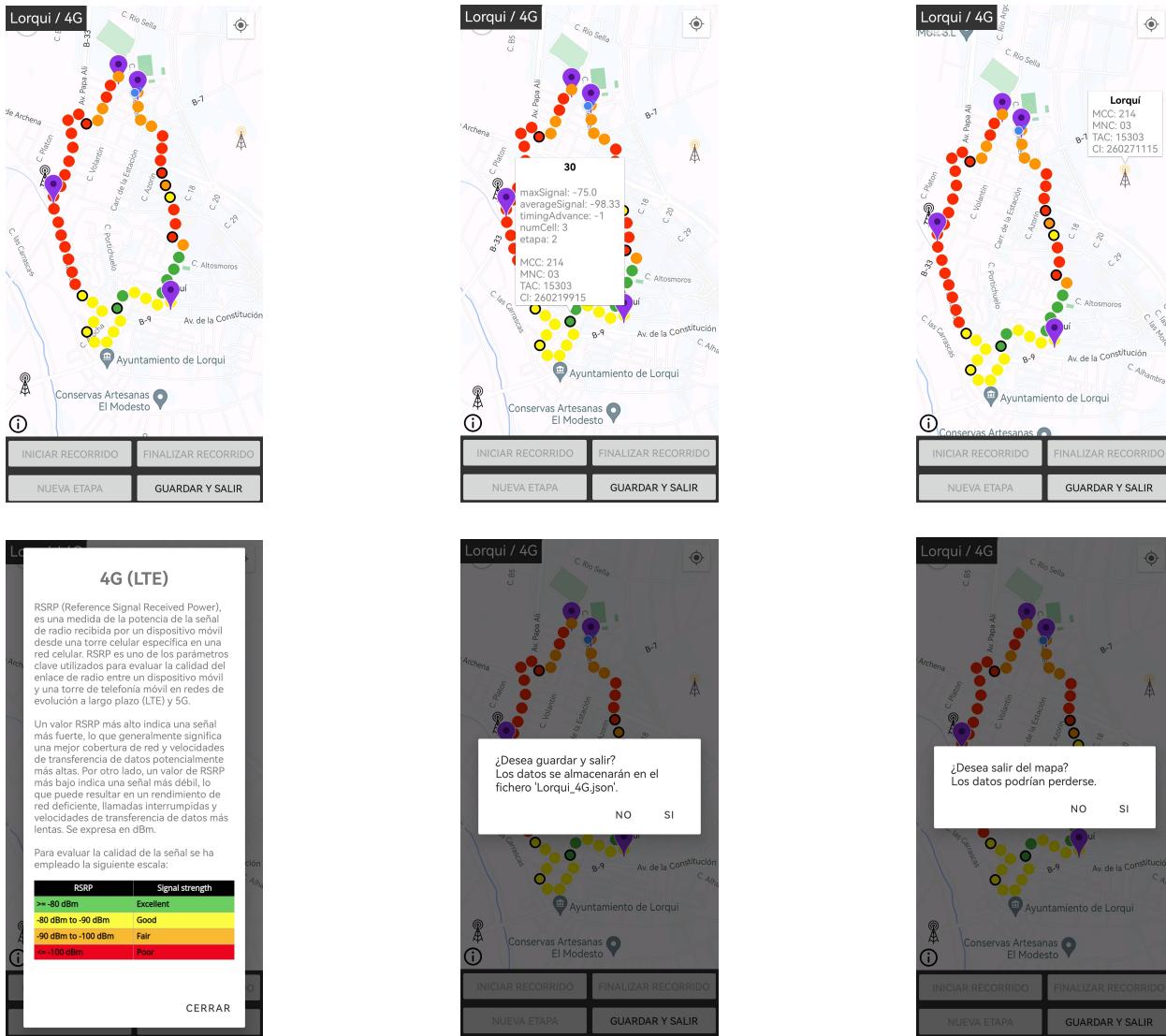


Figura 4: MapsActivity.

Esta es la actividad más importante de la aplicación ya que es la que se encarga de ir recogiendo los datos de la red de telefonía móvil e ir en tiempo real mostrando la información sobre el mapa interactivo.

Esta compuesta por el layout *activity_maps.xml* el cual contiene por un lado el propio mapa, y sobre el se muestra un *TextView* para mostrar el nombre que le ha dado el usuario al recorrido, y

sobre la tecnología que se va a realizar, por otro lado, abajo a la izquierda contiene un *ImageButton* que al hacer click nos abre otro pop-up personalizable como el que se ha mencionado anteriormente y muestra información de interés al usuario, en mi caso, la métrica que se esta usando para medir la señal, cuales son los valores que puede tomar y también la escala de color que se está usando para clasificar dicha señal (en caso de usar 2G, esta pantalla mostraría los datos correspondientes a dicha tecnología, ya que no coinciden). Por último, contiene un panel con 4 botones los cuales realizan las funcionalidades de 'INICIAR RECORRIDO', 'NUEVA ETAPA', 'FINALIZAR RECORRIDO' y 'GUARDAR Y SALIR'.

El flujo de ejecución que sigue la actividad es el siguiente:

El usuario hace click en el botón '**INICIAR RECORRIDO**', entonces se comienza a recibir actualizaciones de ubicación, cada vez que llega una de estas se llama a la función *addMarcador()* que haciendo uso de la función *getNetworkInfo()* obtiene los datos relacionados con la red móvil en ese momento, a partir de estos datos y según la tecnología y la calidad de la señal añadirá un marcador con el color que identifique dicha calidad, esta función también se encarga de comprobar si se ha producido un cambio de antena con respecto a la última vez que se recibió una actualización, en ese caso pondrá un borde negro al marcador para que el usuario lo pueda ver visualmente.

La función *getNetworkInfo()* recoge toda la información relacionada con la telefonía, la almacena y la devuelve a través de un objeto de la clase *NetworkData* que más adelante analizaremos. También llama a la función *addAntena()* la cual se encarga de hacer una petición HTTP para obtener la ubicación de la antena actual y mostrarla en el mapa.

Si el usuario hace click en el botón '**NUEVA ETAPA**', entonces se añadirá un nuevo marcador que delimita el fin de la etapa anterior y el comienzo de la nueva, también incrementará una variable para llevar la cuenta de las etapas que hay.

Si el usuario hace click en el botón '**FINALIZAR RECORRIDO**', entonces se dejarán de recibir actualizaciones de ubicación, una vez aquí el usuario solo podrá interactuar con el mapa, guardar y salir o salir sin guardar.

Si el usuario hace click en el botón '**GUARDAR Y SALIR**', entonces aparecerá un *AlertDialog* confirmando que desea realizar esta acción, en caso afirmativo se llamará a la función *saveInfo()* que se encargará de almacenar en un fichero toda la información recogida y la actividad finalizará.

Detalles de implementación más relevantes:

- ¿Cómo se obtiene la ubicación del usuario?

En mi caso he decidido que la mejor opción es recibir actualizaciones automáticas de ubicación, y así de esta forma cada vez que esto ocurra de forma automática se recojan los datos de la red móvil, y ocurrirá el proceso descrito anteriormente.

Para recibir actualizaciones automáticas de ubicación he hecho uso de *FusedLocationProviderClient*[3] y de *LocationRequest.Builder*[4] configurándolo para recibir actualizaciones de ubicación en un intervalo de 2 a 3 segundos y siempre y cuando el usuario se haya alejado al menos 50 metros de la última ubicación recibida, así evito estar tomando mediciones en un mismo punto si el usuario se queda parado o dos marcas demasiado cerca. A partir de aquí, en el callback actualizo la variable *currentLatLang* que estaré tomando como ubicación actual en el resto del programa.

- ¿Cómo añado un nuevo marcador al mapa?

La función *addMarcador()* en primer lugar obtiene un objeto de la clase *NetworkData* mediante la función *getNetworkInfo()* que contiene los datos de la red móvil en este momento. Posteriormente según la tecnología y en función de la señal máxima detectada crea el marcador y añade toda la información al snippet para que el usuario la pueda visualizar al hacer click. También haciendo uso de la variable *lastNetworkData* comprueba si se ha realizado un cambio de antena, en ese caso añadirá al marcador un borde negro para distinguir el cambio visualmente en el mapa.

- ¿Cómo obtengo la información de la red móvil?

La función *getNetworkInfo()* a través de la clase *TelephonyManager*[5] concretamente a través del método *getAllCellInfo()* obtiene todas las celdas que está detectando el terminal las cuales se recorren y según la tecnología elegida por el usuario selecciona las que sean GSM o bien LTE, de ellas extraerá toda la información relativa a niveles de señal, información de la antena, etc.

- ¿Cómo almaceno la información recogida sobre la red móvil?

Para almacenar la información relativa a un recorrido se emplea un objeto de la clase *FileContent* que almacenará, el nombre del recorrido, la fecha, la tecnología, el número de etapas, el número de puntos (cada punto representa una toma de datos de la red móvil), el número de antenas a las que se ha conectado el terminal, el número de cambios de antena que se han producido y por último, se almacena una lista de objetos *NetworkData* los cuales almacenan para cada punto los respectivos datos de telefonía.

Una vez el usuario finalice el recorrido podrá guardar toda esta información en el almacenamiento, de esto se encargará la función *saveInfo()* que lo único que hará será convertir el objeto *FileContent* en JSON a través de la librería GSON[6] y con la ayuda de la clase *StorageHelper* proporcionada para esta práctica se almacenará en memoria. El nombre del fichero en memoria tendrá el formato '*nombreDelRecorrido_Tecnología.json*'.

- ¿Qué información almaceno sobre la red móvil?

La clase *NetworkData* es la encargada de para cada punto almacenar la información de telefonía. Esta información es, del conjunto de celdas detectadas, se almacena la información que identifica la antena a la que se está conectado (MCC, MNC, TAC y CI en caso de 4G y MCC, MNC, LAC y CID en caso de 2G). También se almacena el TA(Timing Advance), valor que corresponde al tiempo que tarda la señal en llegar desde el MS a la BS. También se

almacena el nivel de señal máximo detectado en estas celdas y la media. Por último, almacena el número de celdas detectadas, un ID para identificar el punto y el número de la etapa en la que se encuentra dicho punto.

Para medir el nivel de señal en 4G hay varios parámetros, en mi caso he optado por usar el RSRP[7] ya que tras investigar parece de los parámetros más correctos para analizar la señal en LTE, este parámetro se mide en dBm. Para obtenerlo, dada una celda LTE se puede hacer mediante el método *getDbm()*[8].

Para medir el nivel de señal en 2G, también hay varios parámetros, en mi caso he optado por usar el RSSI[9] ya que suele ser una de las formas más comunes para pedir la intensidad de la señal en GSM. También se mide en dBm. Para obtenerlo, dada una celda GSM se puede hacer mediante el método *getDbm()*[10].

- ¿Qué escalas de colores uso para clasificar la calidad de la señal máxima detectada?
La escalas de colores nos sirven para clasificar la calidad de la señal en función de su valor, de esta forma podremos decir que se trata de una señal, excelente, buena, suficiente o pobre, y para cada uno de estos términos podremos asignar un color para de forma visual saber de qué tipo se trata.

También hemos visto que para LTE y GSM no se está midiendo el mismo parámetro de señal, por lo que deberemos establecer escalas diferentes.

Tras profundizar sobre esto en numerosos foros[11][12], he llegado a la conclusión de que no hay una escala estandarizada que califique con exactitud el valor de una señal según su intensidad en dBm, pero si es cierto que muchos plantean la misma escala de clasificación, por lo que en mi caso usaré las que he encontrado mayor número de veces y que a su vez me parecen más razonables. A continuación se muestran las mismas:

RSSI	Signal strength	RSRP	Signal strength
$\geq -70 \text{ dBm}$	Excellent	$\geq -80 \text{ dBm}$	Excellent
-70 dBm to -85 dBm	Good	-80 dBm to -90 dBm	Good
-86 dBm to -100 dBm	Fair	-90 dBm to -100 dBm	Fair
$< -100 \text{ dBm}$	Poor	$\leq -100 \text{ dBm}$	Poor

Figura 5: Escala usada en 2G.

Figura 6: Escala usada en 4G.

- ¿Cómo se muestran las antenas en el mapa?
De esta tarea se encarga la función *addAntena()* la cual es llamada por *getNetworkInfo()*,

recibiendo por parámetro los valores necesarios para identificar una antena (MCC, MNC, TAC y CI en caso de 4G y MCC, MNC, LAC y CID en caso de 2G).

Con estos datos se hace una petición HTTP a la API '*mylnikov*'[13] que nos devolverá un JSON donde entre otras cosas encontraremos las coordenadas (latitud, longitud) de la antena.

En mi caso para realizar la petición HTTP empleo *Volley*[14] una biblioteca de Android que permite hacer peticiones de forma muy sencilla.

Además, *addAntena()* se encarga de mantener un registro de las antenas que ya se han reconocido hasta el momento de forma que comprueba si la antena a la que estamos conectados ya se muestra en el mapa. Esto también nos sirve para ir cambiándole el icono, de forma que si se trata de la antena a la que estamos conectados actualmente el icono saldrá de color para que el usuario la pueda identificar, si se trata de una a antena a la que hemos conectados anteriormente el icono será negro. Haciendo click en el mapa sobre la antena el usuario podrá ver el nombre de la localidad donde se encuentra y los datos que la identifican.

- **¿Cómo se gestionan los permisos?**

Para la gestión de permisos hay que redefinir el método *onRequestPermissionsResult()*, este distingue desde donde venimos a través del *requestCode*, es decir, podemos llegar desde la función *startShowingLocation()* solicitando permisos de ubicación para mostrar la posición del usuario en el mapa, podemos llegar desde *getLocationUpdates()* solicitando permisos de ubicación para ir recibiendo las actualizaciones de ubicación del usuario o podemos llegar desde *getNetworkInfo()* solicitando permisos para acceder a los datos de telefonía.

En cualquiera de estas opciones, en caso de que el usuario conceda los permisos el flujo del programa volverá a la función que los solicitó. En caso de que el usuario no los conceda, entonces he decidido que se mostrará un mensaje informativo por pantalla (indicando que son necesarios) y la actividad finalizará. Creo que es lo más oportuno, ya que sin ubicación o telefonía no hay funcionalidad posible en el contexto de lo que debe realizar esta actividad.

- **¿Cómo se añade una nueva etapa?**

La gestión de las etapas es una de las tareas más sencillas en la actividad. Cuando inicia el recorrido se añade al mapa un marcador de color morado delimitando el comiendo de la etapa 1. A través de la variable *numEtapa* llevamos la cuenta de la etapa en la que nos encontramos, esta se incrementa en caso de que el usuario haga click en el botón 'NUEVA ETAPA', en ese caso también se añadirá un marcador al mapa delimitado el fin de la etapa actual y el comienzo de la nueva. Por último, cuando finaliza el recorrido, se añade un último marcador para delimitar el fin de la última etapa.

Como ya se ha comentado anteriormente, de esta variable hace uso la función *getNetworkData()* para saber en qué etapa se encuentra cada punto.

- ¿Cómo se muestra el pop-up de información?

Como ya se ha comentado anteriormente se ha añadido un *ImageButton* que tiene como imagen un ícono de información. En caso de que el usuario haga click aparecerá un pop-up personalizado como los que ya se han comentado anteriormente y el cual tiene como objetivo darle información al usuario sobre el parámetro de señal que se está tomando en función de la tecnología, lo describe brevemente y muestra la escala que se está empleando para clasificar los distintos puntos.

- Se redefine el método *onBackPressed()* para que cuando se haga click en el botón de retroceder aparezca un *AlertDialog*[1] y haya que confirmar la salida, de forma que no se pueda salir por error de la actividad y se pierdan los datos recogidos.
- Se redefine el método *onPause()* para dejar de recibir las actualizaciones de ubicación en caso de que la actividad pase a segundo plano.
- Se redefine el método *onResume()* para volver a activar las actualizaciones de ubicación en el caso de que el recorrido ya este empezado. Esto puede pasar por ejemplo, si el usuario estaba realizando su recorrido, salió de la aplicación dejándola en segundo plano y regresa.
- Se redefine el método *onDestroy()* para restaurar el comportamiento normal de la pantalla ya que en el método *onCreate()* se activa el flag '*FLAG_KEEP_SCREEN_ON*' para que la pantalla del dispositivo no se apague mientras se realiza el recorrido. Aunque esto pueda suponer un coste en energía, he decidido que puede ser una buena opción para evitar que la pantalla se apague y se pausen las actualizaciones de ubicación y se deje durante un tiempo de recoger datos de la red móvil.
- Se añade la propiedad *android:screenOrientation='portrait'* a la actividad *MapsActivity* en el *AndroidManifest.xml* para evitar que la pantalla no se pueda girar evitando así resets.
- Mediante el archivo *strings.xml* se realiza la traducción para que la aplicación sea multilingüe. El idioma por defecto es el Español y está traducida para que se pueda usar en Inglés.

3. Funcionalidad opcional.

En este apartado se van a describir los detalles de implementación de los distintos aspectos que se han implementado como parte opcional de la práctica.

3.1. Opcional 1: Comparación con otra tecnología.

Tal y como se ha ido introduciendo a lo largo del documento se ha implementado la posibilidad de que el usuario pueda realizar un recorrido para realizar un estudio de la señal tanto 4G como 2G.

Para esto, el usuario tiene la posibilidad de seleccionar entre ambas opciones mediante un *RadioButton* a la hora de iniciar el recorrido. Esta información será pasada y almacenada en la clase *MapsActivity* a través de una variable de tipo string que nos servirá para a la hora de obtener los

datos de la red móvil seleccionar en función de esta las celdas LTE o bien GSM, por otro lado, a la hora de mostrar el marcador para usar una escala de colores o otra, y por último, para mostrar un contenido o otro en el pop-up de información según la tecnología.

Más adelante en la sección '*Análisis teórico*' realizaremos un recorrido 2G y realizaremos una comparativa con el 4G.

3.2. Opcional 2: Pantalla de análisis. AnalysisActivity.

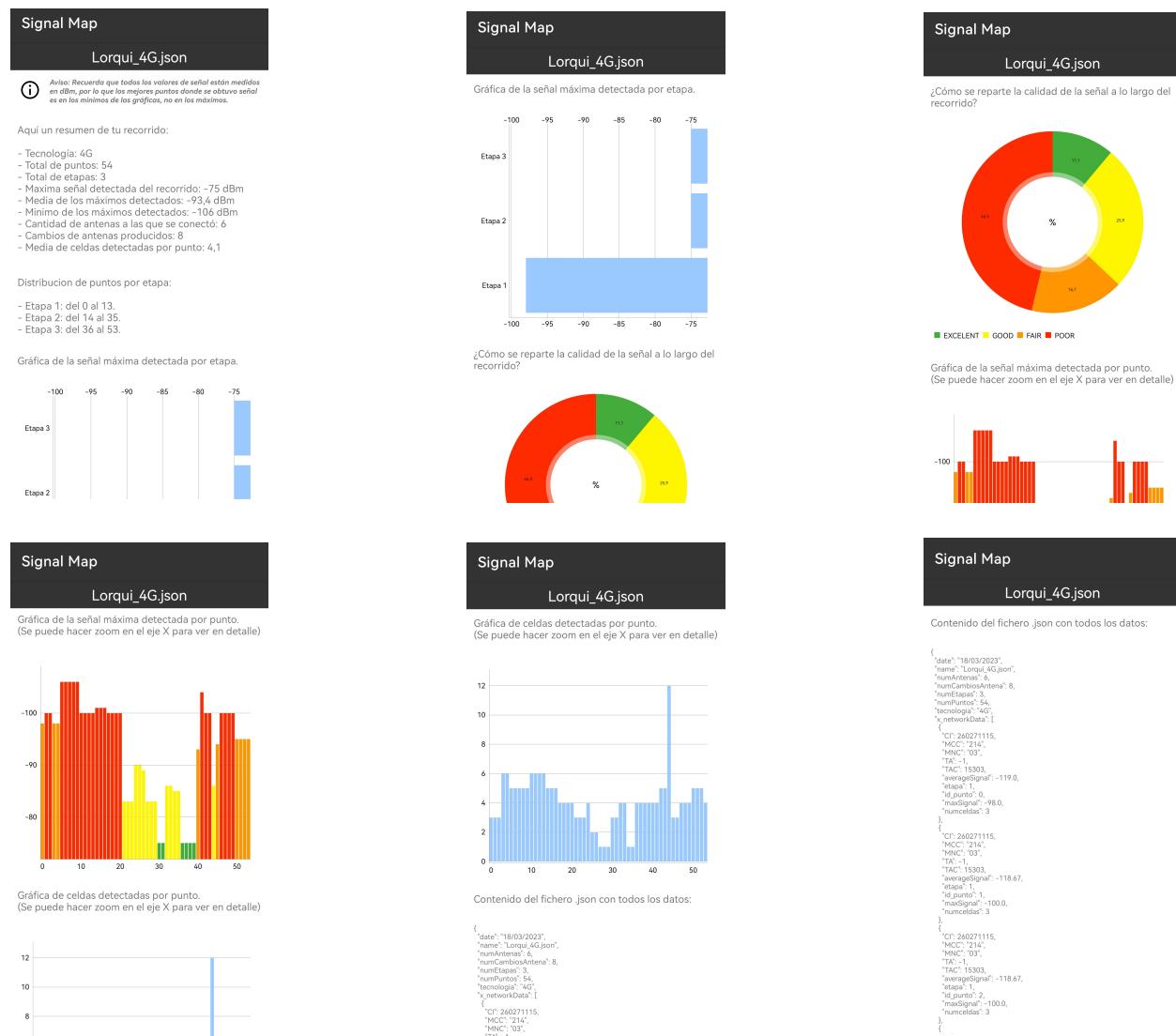


Figura 7: AnalysisActivity.

Esta actividad tiene como objetivo mostrar de forma visual información sobre un recorrido y algunas gráficas de forma que se puedan analizar los distintos parámetros medidos a lo largo del mismo, realizar comparaciones y sacar conclusiones.

Para realizar esto, la actividad recibe la ruta absoluta del fichero que el usuario quiere analizar (previamente seleccionado mediante el *File Picker*) para posteriormente de nuevo usando la librería *GSON*[6] obtener el objeto *FileContent* que contiene toda la información relativa al recorrido a analizar.

Para leer el contenido del fichero en memoria de nuevo se hace uso de la clase *StorageHelper* facilitada para esta práctica, pero con una pequeña modificación, ya que antes recibía una ruta relativa y ahora recibe una ruta absoluta, así damos la posibilidad de si el usuario a movido el archivo de carpeta que lo pueda abrir de igual forma.

Esta actividad muestra parte de la información en forma de gráficas, de forma que se pueda analizar de forma más visual. En mi caso, para crear y mostrar las mismas he empleado la librería *MPAndroidChart*[15] que permite dibujar una gran variedad de gráficas y todas ellas configurables, en cuanto a tamaño, forma, color, tipos de datos, leyendas, etc. Por esto mismo he decidido usar esta librería, además es bastante común por lo que es sencillo encontrar ejemplos y documentación, lo que hace que sea fácil de usar.

El contenido del layout de esta actividad contiene lo siguiente:

1. Un pequeño panel donde se muestra un recordatorio al usuario y de nuevo un *ImageButton*, exactamente igual al que hemos comentado en el *MapsActivity*.
2. Un *TextView* que muestra un breve resumen de los datos más relevantes del recorrido (tecnología, número de puntos, número de etapas, niveles de señal detectados, máximo, mínimo y medio, número de antenas a las que nos hemos conectado, número de cambios de antenas producidos, media de celdas detectadas por etapa y distribución de puntos por etapa, es decir, de qué punto a qué punto va la etapa 1, y así para el resto).
3. Un *HorizontalBarChart* (gráfico de barras horizontales) donde se muestra para cada etapa el nivel máximo de señal detectada en la misma.
4. Un *PieChart* (gráfico circular o de pastel) donde se muestra cómo se reparte la calidad de la señal a lo largo del recorrido, es decir, suponiendo que el total de puntos es el 100% del gráfico, nos muestra en qué porcentaje de puntos se detectó un nivel de señal máximo excelente, bueno, suficiente o pobre. Esto permite al usuario saber cual de ella predominó a lo largo del recorrido.
5. Un *BarChart* (gráfico de barras) que nos muestra la calidad de la señal máxima detectada a lo largo del recorrido, es decir, para cada punto que se registró, se mostrará una barra del color correspondiente según la calidad (aquí también se tiene en cuenta la tecnología para usar una

escala u otra). Esto permitirá ver cómo evoluciona la calidad de la señal a lo largo del tiempo y recorrido. Puesto que este gráfico puede contener muchas barras, ofrece la posibilidad de hacer zoom en el eje X por si hay alguna zona que se quiera ver más en detalle.

6. Un *BarChart* (gráfico de barras) igual que el anterior, pero en este se representa el número de celdas detectadas en cada punto a lo largo del recorrido. Esto puede servir al usuario para ver como evoluciona, y para compararlo con el gráfico anterior y establecer posibles relaciones.
7. Un *TextView* que muestra el contenido del fichero 'a pelo', a priori se puede convertir en una sección muy larga pero su única finalidad es que si el usuario lo desea pueda ir y consultar los datos específicos de algún punto en concreto.

* Nota: siempre que hablamos de un valor de señal en específico nos estamos refiriendo al nivel máximo de señal detectado en ese punto en concreto.

4. Análisis teórico.

Para la realización de este apartado se seleccionará un recorrido de unos 4,5km que atraviese distintos entornos para poder analizar qué ocurre con la señal y los distintos parámetros de red recogidos a lo largo del mismo.

Este recorrido se realizará en primer lugar para la tecnología 4G (LTE) y posteriormente para 2G (GSM), una vez analizados ambos recorridos, se realizará una comparativa entre los resultados de ambas tecnologías.

4.1. Recorrido 4G (LTE).



Para la realización de este apartado se realiza el recorrido que se muestra a través de las imágenes anteriores por distintos puntos de la localidad donde resido (Lorquí). El recorrido consta de un total de 137 puntos (repartidos en 7 etapas) donde se han recogido los valores máximos de señal, entre otras cosas, detectados en cada uno de ellos.

- Etapa 1: Es una etapa corta, ya que representa la salida de una zona residencial y que está alejada del centro de la localidad por lo que es razonable que la intensidad de la señal sea suficiente y pobre.
- Etapa 2: Esta etapa es a lo largo de las afueras del municipio, pero el entorno es abierto, no hay obstáculos por lo que aunque sigamos lejos del centro de la localidad, en gran parte de la etapa la calidad mejora, siendo buena.

- Etapa 3: En esta etapa volvemos a ver un empeoramiento de la calidad, es debido a que entramos en una zona de huerta.
- Etapa 4: Al comienzo de esta etapa seguimos teniendo mala cobertura debido a que seguimos por un tramo de huerta, pero en el momento que subimos hasta la orilla del río vemos una clara mejora del nivel de la señal, esto es debido a que estamos a una altura superior (evitando apantallamientos). Además, el río hace de separador entre dos localidades que tiene a cada lado, por lo que es razonable que sea un punto donde deba haber buena señal.
- Etapa 5: En esta etapa mejora drásticamente la cobertura debido a que salimos de la zona del río y pasamos a un entorno abierto a lo largo de un carril bici que se dispone entre las localidades de Lorquí y Ceutí.
- Etapa 6: Esta etapa se desenvuelve en un espacio más cerrado ya que es a través de las calles de la localidad, al principio la calidad sigue siendo bastante aceptable, pero al final del recorrido es un entorno con calles algo estrechas, por lo que es razonable que debido a fenómenos como la refracción, difracción o dispersión, que hacen que las ondas cambien su comportamiento al chocar con objetos, se dé un empeoramiento de la calidad de la señal.
- Etapa 7: Esta etapa posee las mismas características que la primera etapa.

Echando un vistazo a la actividad de análisis podemos observar en la gráfica circular como con un 40,1% predomina a lo largo del recorrido una calidad de señal buena, seguido por un 25,5% suficiente, un 20,4% pobre y un 13,9% excelente.

Esta actividad también nos indica que el valor máximo de señal detectado a lo largo del recorrido es de -69 dBm, valor que si observamos en el gráfico de barras podemos ver que pertenece al punto 57, y si observamos el de barras horizontales, podemos ver que dicho punto pertenece a la etapa 4, justo en el momento que comentábamos anteriormente subiendo a la orilla del río. A lo largo del recorrido, el valor medio de señal detectada es de -90,1 dBm. El punto donde peor señal se detectó a lo largo del recorrido fue en el punto 31 (etapa 3) con un valor de -109 dBm, este fue justo el momento donde se produce la transición a la zona de huerta.

Si seguimos observando la pantalla de análisis vemos que el terminal se conectó a un total de 7 antenas, produciéndose un total de 26 cambios entre ellas, lo que se puede considerar un número no muy alto teniendo en cuenta la amplitud que abarca el recorrido. Esto puede deberse a que 4G busca mantener la conexión con el dispositivo con la misma antena durante el mayor tiempo posible siempre y cuando la señal sea suficientemente fuerte y estable, reduciendo así cambios de antena lo que mejora la calidad de la conexión.

También podemos observar que la media de celdas detectadas por punto es de 4.1, siendo 13 el mayor número de celdas detectadas en un punto (punto 15) y siendo 1 el número mínimo de celdas detectadas en numerosos puntos. Para este recorrido no se aprecia de forma tan radical, pero para otros de prueba que he realizado se puede apreciar comparando el gráfico de barras de señal

máxima con el de número de celdas detectadas que a menor número de celdas detectadas suele haber mejor cobertura, o lo que es lo mismo, en las zonas donde el terminal está detectando más celdas, hay peor calidad de la señal, quizás esto podría deberse a que se produzcan interferencias entre ellas.

4.2. Recorrido 2G (GSM).

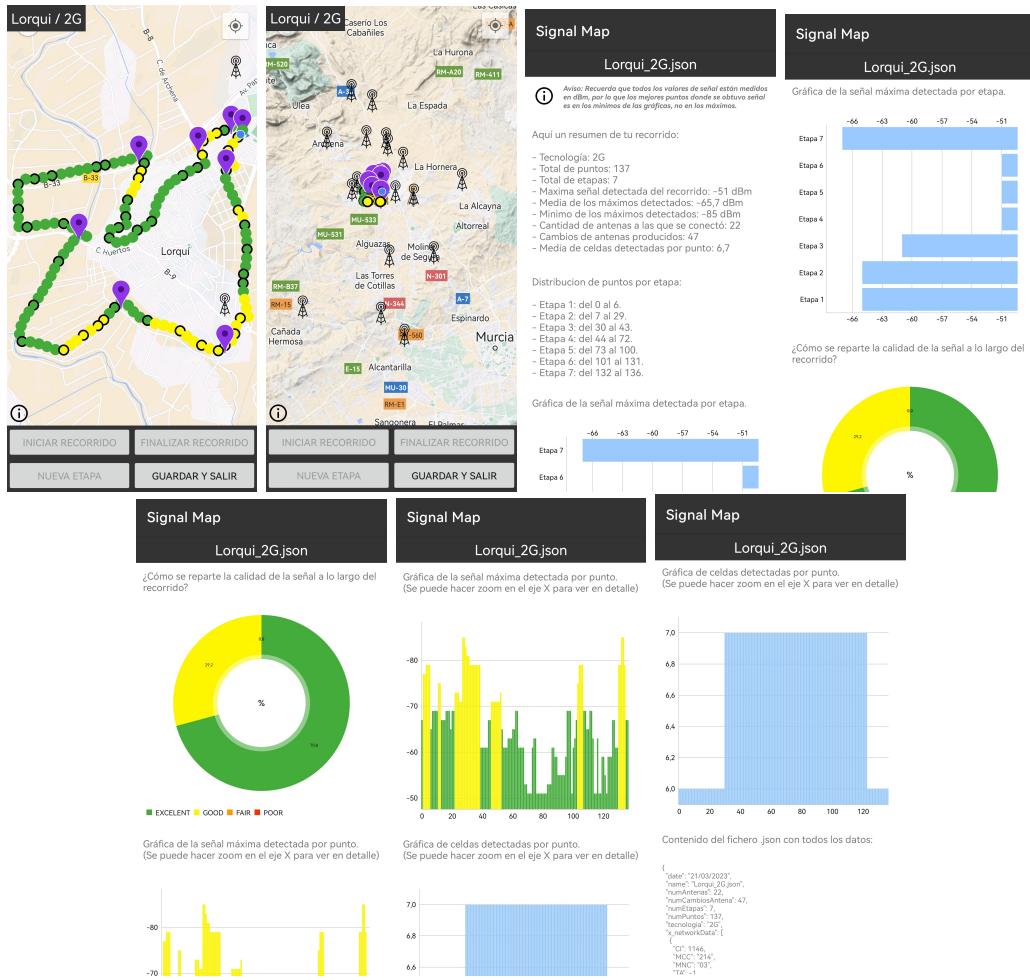


Figura 9: Recorrido 2G (GSM). (hacer zoom para ver en detalle)

Para la realización de este apartado se repite exactamente el mismo recorrido descrito en el apartado anterior, observando de la misma forma, como se repiten las zonas donde la calidad de la señal es un poco más baja.

Como se puede observar para esta tecnología tenemos mejor calidad de la señal, si observamos la gráfica circular podemos ver que predomina una calidad excelente en un 70.8 % seguido de un

29.2 % para una calidad buena.

También podemos ver el máximo de nivel de señal detectado en -51 dBm en numerosos puntos a lo largo de las etapas 4, 5 y 6, justo coincidiendo con los tramos que recorren la orilla del río, carril bici y el centro de la localidad. A lo largo del recorrido, el valor medio de señal detectada es de -65.7 dBm. El punto donde peor señal se detectó a lo largo del recorrido fue en los puntos 132 y 133 (etapa 7) con un valor de -85 dBm, este fue el momento que nos encontrábamos a las afueras de la localidad y entrando en una zona residencial.

Si seguimos observando la pantalla de análisis podemos observar que el terminal realizó 47 cambios de antena, entre un total de 22 antenas diferentes y detectando una media de 6.7 celdas por punto. Esto es debido a que las celdas en GSM son de gran tamaño, por ello el teléfono detecta varias de ellas en todo momento y al no tener una tecnología avanzada en la gestión de handovers hace que en esta tecnología los cambios de antena sean bastante comunes.

4.3. Comparativa entre recorridos.

Si realizamos una comparativa entre los recorridos LTE y GSM, podemos ver en primer lugar como la calidad de la señal es superior en 2G, eso sí, a pesar de ello 4G sigue siendo mucho mejor ya que:

1. La velocidad de transmisión de 4G es mucho más rápida que la de 2G, gracias a tecnologías como OFDM y MIMO 4G permite velocidades de descarga de hasta varios cientos de Mbps mientras que 2G está limitado a unos pocos Mbps.
2. El ancho de banda 4G es mucho más amplio, de varios MHz, mientras que el de 2G de unos pocos KHz.
3. La latencia de 4G es típicamente de unos 50 ms, mientras que la de 2G es superior a 500 ms, lo que hace que la experiencia del usuario en 4G sea más rápida y fluida.

Un motivo por el que la señal en 2G puede ser mejor que en 4G es porque 2G opera en bandas de frecuencia más bajas que 4G, y tal y como hemos visto en teoría frecuencias más bajas tienen un mayor alcance y son más robustas, por lo que penetran mejor en los objetos.

Otra curiosidad a destacar es que si comparamos ambos recorridos rápidamente se puede observar que coinciden las zonas donde hay una mejor y peor cobertura a lo largo del recorrido, es por ello que podemos concluir que se produce por las distintas características del entorno comentadas anteriormente.

Otro aspecto muy interesante a destacar es con respecto a las antenas, como se puede observar para el recorrido 2G, tenemos un gran número de antenas y a una gran distancia del terminal, en cambio si lo comparamos con 4G vemos un número de antenas mucho menor y mucho más cercanas al terminal. Esto confirma que efectivamente como hemos visto en teoría, conforme van

evolucionando las tecnologías, en concreto la parte radio, siempre se trata de acercarla más al terminal, y también localizandolo cada vez en áreas más pequeñas para facilitar el proceso de paginación, lo que también coincide con la reducción de celdas detectadas de 2G a 4G.

5. Conclusión.

Tras realizar la práctica la valoración es positiva, ya que por un lado he tenido la oportunidad de por primera vez desarrollar una app para un terminal móvil, cosa que satisface cuando finalizas, ves el resultado y compruebas que funciona.

Por otro lado, también me ha parecido interesante y curioso el poder ver cómo se comporta la señal en mi localidad y gracias a la comparativa entre tecnologías llegar a las mismas conclusiones que se han visto en la parte teórica de la asignatura con respecto a las tecnologías GSM y LTE.

Recalcar que quise recabar información sobre el Time Advance (TA) en cada punto, para también después analizarlo, pero finalmente no se porque mi terminal no devuelve ningún valor para esta métrica, si no que siempre devuelve *UNAVAILABLE*[16], por lo que no puede analizarlo.

Es cierto que al decir no realizarla en pareja y realizarla en un breve periodo de tiempo, eso sí dedicando bastantes horas diarias durante varias semanas, la última parte y la elaboración de esta memoria se ha hecho un poco cuesta arriba, pero esto es una decisión que yo tomé, igualmente experiencia satisfactoria.

Referencias

- [1] <https://developer.android.com/guide/topics/ui/dialogs?hl=es-419>
- [2] <https://github.com/TutorialsAndroid/FilePicker>
- [3] <https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient>
- [4] <https://developer.android.com/reference/android/location/LocationRequest.Builder>
- [5] <https://developer.android.com/reference/android/telephony/TelephonyManager>
- [6] <https://javadoc.io/doc/com.google.code.gson/gson/latest/com.google.gson/com.google/gson/package-summary.html>
- [7] <https://forum.huawei.com/enterprise/es/funcionamiento-del-rsrp/thread/1058450-100275>
- [8] [https://developer.android.com/reference/android/telephony/CellSignalStrengthLte#getDbm\(\)](https://developer.android.com/reference/android/telephony/CellSignalStrengthLte#getDbm())
- [9] <https://norfipc.com/redes/intensidad-nivel-senal-redes-moviles-2g-3g-4g.php>
- [10] [https://developer.android.com/reference/android/telephony/CellSignalStrengthGsm#getDbm\(\)](https://developer.android.com/reference/android/telephony/CellSignalStrengthGsm#getDbm())
- [11] <https://poynting.tech/articles/signal-strength-measure-rsrp-rsrq-and-sinr-reference-for-lte>
- [12] https://wiki.teltonika-networks.com/view/Mobile_Signal_Strength_Recommendations
- [13] <https://www.mylnikov.org/archives/1059>
- [14] <https://developer.android.com/training/volley/simple?hl=es-419>
- [15] <https://github.com/PhilJay/MPAndroidChart>
- [16] <https://developer.android.com/reference/android/telephony/CellInfo#UNAVAILABLE>