Escuela Politécnica Superior de Jaén

Grado en Ingeniería Eléctrica

UNIVERSIDAD DE JAÉN

Nombre del Centro

Trabajo Fin de Grado

DISEÑO DE UNA APLICACIÓN PARA MÓVILES Y TABLETAS PARA TRABAJO EN CAMPO CON LÍNEAS DE ALTA TENSIÓN

Alumno: Yera Díaz, Francisco Javier

Tutor: Prof. D. Francisco José Sánchez Sutil

Dpto: Ingeniería Eléctrica

SANCHEZ SUTIL FRANCISCO JOSE - 26004797P

Firmado digitalmente por SANCHEZ SUTIL FRANCISCO JOSE - 26004797P
Fecha: 2016.08.31 13:52:00 +02'00'

Septiembre, 2016

ÍNDICE

1. INTRODUCCIÓN 5
2. ANTECEDENTES 6
2.1. Breve historia de Android 6
2.2. Ventajas e Inconvenientes 8
2.3. Usos de Andoid 10
2.4. Aplicaciones similares ya existentes 14
2.5. Elección e Instalación del software a utilizar 22
2.6. Lenguaje de Programación 30
2.6.1. Lenguaje JAVA 31
2.6.2. Lenguaje XML 34
2.7. Herramientas utilizadas en la programación 39
2.7.1. Google Maps API 39
2.7.2. Base de datos SQlite 41
2.7.3. JSON 42
2.7.4. Librerías Adicionales 43
2.8. Estructura de una aplicación. Componentes 44
2.8.1. Ventana de escritura de código o de programación. 46
2.8.2. Estructura o “árbol” del proyecto 47
2.8.3. Android monitor. 51
2.8.4. Gradle Console o consola de eventos. 52
2.8.5. Botón Android SDK Manager 52
2.8.6. Botón AVD 53
2.8.7. Botón Run 54
3. OBJETIVOS 56
4. METODOLOGÍA 57
5. RESULTADOS 71
6. DISCUSIÓN 73
7. BIBLIOGRAFÍA 75
8. ANEXOS 78
8.1 MANUAL DE USUARIO 78
8.2 CÓDIGO FUENTE 86

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Versiones de Android. 7
Ilustración 2: Uso de Android en Italia. 11
Ilustración 3: Uso de Android en Gran Bretaña. 11
Ilustración 4: Uso de Android en Alemania. 11
Ilustración 5: Uso de Android en Francia. 12
Ilustración 6: Uso de Android en China. 12
Ilustración 7: Uso de Android en Estados Unidos. 12
Ilustración 8: Uso de Android en España. 13
Ilustración 9: Distribución de las versiones de Android. 13
Ilustración 10: Icono aplicación Topography APP. 15
Ilustración 11: Creación de una "survey". 15
Ilustración 12: Interior de una lista con diversos puntos guardados. 16
Ilustración 13: Exportación de una lista. 16
Ilustración 14: Menú de opciones. 17
Ilustración 15: Icono Aplicación GeoGPS. 18
Ilustración 16: Visualización de un punto. 18
Ilustración 17: Guardado de punto. 19
Ilustración 18: Icono Accurate Altimeter. 19
Ilustración 19: Datos del punto. 20
Ilustración 20: Lista de puntos. 20
Ilustración 21: Icono Tower Topography Lines. 21
Ilustración 22: Lista de puntos. 21
Ilustración 23: Propiedades del punto. 22
Ilustración 24: Versiones de JDK. 25
Ilustración 25: Web de descarga de Android Studio. 26
Ilustración 26: Proceso de instalación de Android Studio. 27
Ilustración 27: Instalación de Android Studio completada. 27
Ilustración 28: Android Studio SDK Manager: Versiones de Android. 28
Ilustración 29: Android Studio SKD Manager: Herramientas Adicionales. 29
Ilustración 30: Definición de un número complejo. 33
Ilustración 31: Estructura del código XML. 35
Ilustración 32: Layouts más comunes. 37
Ilustración 33: Visualización de un Layout simple de forma programática. 38
Ilustración 34: Visualización de un Layout con el panel de diseño. 38
Ilustración 35: Logo de Google Maps. 41
Ilustración 36: Logo Base de datos SQLite. 41
Ilustración 37: Logo del formato JavaScript JSON. 42
Ilustración 38: Logo de la Librería MPAndroidChart. 43
Ilustración 39: Logo del formato de datos CSV. 44
Ilustración 40: Entorno de programación en Android Studio. 45
Ilustración 41: Distinción de los elementos más importantes del entorno. 45
Ilustración 42: Ventana de Código Java. 46
Ilustración 43: Estructura o "árbol" del proyecto. 48
Ilustración 44: Librerías incluidas en el proyecto. 49
Ilustración 45: Archivos Java incluidos en el proyecto. 49

Ilustración 46: Carpeta Res o de recursos. 50
Ilustración 47: Permisos incluidos en el AndroidManifest.xml....51
Ilustración 48: Logcat de Android Monitor....51
Ilustración 49: Muestra de resultados en Grandle Console....52
Ilustración 50: Herramientas añadidas mediante SDK Manager. 53
Ilustración 51: Máquinas Virtuales creadas con AVD. 54
Ilustración 52: Selección de dispositivo para correr la aplicación. 55
Ilustración 53: Creación de una Google Maps activity. 58
Ilustración 54: Mapa Inicial. 59
Ilustración 55: Información acerca del punto....61
Ilustración 56: Icono del botón Archivar. 62
Ilustración 57: Ejemplo de puntos guardados. 63
Ilustración 58: Icono del botón Cambio de Mapa. 63
Ilustración 59: Tipos de mapas. 64
Ilustración 60: Selección del tipo de mapa. 65
Ilustración 61: Icono del botón Borrar. 65
Ilustración 62: Borrar un punto. 66
Ilustración 63: Opción Borrar todos los puntos. 67
Ilustración 64: Icono de botón Exportar datos. 67
Ilustración 65: Selección de aplicación para el envío. 68
Ilustración 66: Archivo adjunto. 69
Ilustración 67: Icono de botón Gráfica. 69
Ilustración 68: Perfil de altitud de los puntos guardados. 70
Ilustración 69: Pantalla principal de la aplicación. 79
Ilustración 70: Función para guardar un punto. 80
Ilustración 71: Almacenamiento de diversos puntos. 81
Ilustración 72: Diferentes tipos de mapas. 82
Ilustración 73: Función de borrado de puntos. 83
Ilustración 74: Selección de aplicación de correo y correo generado. 84
Ilustración 75: Perfil de los puntos almacenados. 85

1. INTRODUCCIÓN

A la hora de realizar una línea de alta tensión se ponen de manifiesto una serie de requerimientos para su trazado que no se deben dejar escapar.

El proyectista ha de pisar el campo donde se van a situar los apoyos para tener la seguridad de que es ahí donde realmente puede existir ese apoyo. Después de hacerlo, ha de desarrollar todos esos puntos en un software para el análisis de la línea y su respectivo perfil longitudinal. Para conseguir estos datos y ayudar al proyectista a la hora de elegir un trazado exacto, habiendo valorado distintas posibilidades, se plantea la creación de una aplicación para móvil o tablet.

Y, ¿Por qué para móvil o tablet? En la actualidad los dispositivos móviles disponen casi en su totalidad de GPS con el cual podemos conocer nuestra ubicación y la de cualquier punto deseado. De hecho, si los comparamos con un GPS convencional podemos observar que no existen grandes diferencias ambos. Si bien es cierto que un GPS de alta gama siempre ofrecerá resultados más exactos. Pero, en la actualidad, está al alcance de cualquiera disponer de un *Smartphone* o Tablet que, además de realizar el trabajo de un GPS con bastante solvencia, nos proporciona también la posibilidad de trabajar con los datos y de poder traspasarlos a un ordenador para seguir trabajando con ellos. Por ello considero la elección totalmente acertada.

Esta aplicación no exime al proyectista de hacer lo antes relatado pero sí facilita el trabajo ya que actualmente el móvil es un elemento indispensable en nuestras vidas y todos acostumbramos a llevarlo con nosotros a diario.

2. ANTECEDENTES

Al estar nuestra aplicación desarrollada en el entorno Android, vamos a comenzar explicando por qué se ha elegido este sistema operativo y no cualquier otro de los disponibles:

2.1. Breve historia de Android

El sistema operativo Android, es más antiguo de lo que nos pueda parecer en un primer momento. Sus orígenes se sitúan allá por 2003, cuando el sistema fue desarrollado a modo particular por la compañía Android Inc, una pequeña empresa ubicada en Alto Palo, California, que en 2005 sería comprada por Google quien continuó el desarrollo del sistema.

Entre 2005 y 2007 Google liberó algunas versiones de Android conocidas con Alpha y Beta. No se trataba de versiones definitivas, de hecho, ni siquiera había dispositivos comerciales que los utilizasen. Dichas versiones sirvieron para que los desarrolladores comenzasen a programar y a probar aplicaciones en ellos.

La primera versión de Android definitiva, versión 1.0, llegó al mercado en 2008 junto con el primer *Smartphone* que la incorporó: el *HTC Dream.*

En ella se encontraron características y funcionalidades que en la actualidad consideramos básicas pero en aquel momento fueron punteras. Dicha versión incluía una tienda de aplicaciones conocida como Android Market (Hoy Google Store), así como compatibilidad con diversas aplicaciones de Google como Maps, Contactos o Youtube, entre otras.

Tras diversas actualizaciones llegamos a un punto de inflexión con la llegada de Android 1.5, conocida como *"Cupcake",* la primera en tener un nombre de dulce. Esta fue la primera gran versión de Android que pasó del desconocimiento a ser una plataforma reconocida y que contó con el apoyo de muchos fabricantes. Se mejoró la interfaz, se introdujeron *widgets* y se mejoraron muchas aplicaciones integradas en el sistema como los contactos o Youtube.

A partir de aquí la plataforma Android fue mejorando y creciendo a lo largo de los años, pasando por diferentes versiones hasta llegar a las que usamos en la actualidad.

Una pequeña lista con las versiones de Android, todas con nombre de dulces, sería la siguiente:

- Android 1.0 “Apple pie”. Lanzada en octubre de 2008.
- Android 1.1 “Banana bread”. Lanzada en febrero de 2009.
- Android 1.5 “Cupcake”. Lanzada en abril de 2009.
- Android 1.6 “Donut”. Lanzada en septiembre de 2009.
- Android 2.0/2.1 “Eclair”. Lanzada en octubre de 2009.
- Android 2.2 “Froyo”. Lanzada en mayo de 2010.
- Android 2.3 “Gingerbread”. Lanzada en diciembre de 2010.
- Android 3.0 “Honeycomb”. Lanzada en febrero de 2011.
- Android 4.0 “Ice Cream Sandwich”. Lanzada en octubre de 2011.
- Android 4.1/4.2/4.3 “Jelly bean”. Lanzada en junio de 2012.
- Android 4.4 “Kit Kat”. Lanzada en octubre de 2013.
- Android 5.0/5.1 “Lollipop”. Lanzada en noviembre de 2014.
- -Android 6.0 “Marshmallow. Lanzado en octubre de 2015.
- -Android 7.0 “N” (En desarrollo)

Ilustración 1: Versiones de Android.

Recuperado en: http://rpp.pe/blog/geek-and-chick/android-n-que-novedades-trae-y-cual-es-su-verdadero-nombre-noticia-944666

Tras todas estas actualizaciones, era de esperar que el entorno de Android cambiara mucho desde su versión inicial. Aquí se puede apreciar que no solo ha cambiado en el entorno, con la última versión de Android (Marshmallow) vienen incluidas una serie de mejoras como: El soporte para el reconocimiento nativo de huellas digitales, que nos servirá no solo para desbloquear nuestro teléfono sino también para realizar transacciones o compras online por ejemplo. También se incluyen los permisos para aplicaciones caso por caso, el sistema "Doze" para ahorrar batería, o el nuevo sistema de carga y transmisión de datos USB "tipo-C" el cual es simétrico y se puede conectar por ambos lados, el cual permitirá una carga hasta 5 veces más rápida

2.2. Ventajas e Inconvenientes

Android resulta ser el sistema operativo más popular del mundo. Esto se debe a que es una plataforma gratuita que, además de disfrutar del apoyo de Google, permite ser utilizado por cualquier fabricante.

A continuación se enumeran algunas de las ventajas por las que se ha vuelto tan exitoso este sistema operativo:

- **Código abierto**: Google liberó Android bajo licencia Apache. Gracias a esto cualquier persona puede realizar una aplicación Android, incluso se pueden realizar sistemas operativos alternativos basados en éste, como es el caso de Fire OS de Amazon para sus Kindle. A estas versiones alternativas de Android se las conoce como ROMs y son muy utilizadas por muchos usuarios. En este sentido se dice que Android es un sistema "Abierto" el cual proporciona prácticamente una total libertad a sus usuarios.

- **Aplicaciones o Apps externas**: A día de hoy existen infinidad de aplicaciones disponibles para teléfonos Android, la mayoría de ellas gratuitas. Además se pueden instalar aplicaciones ajenas a la tienda oficial Google Play, lo que aumenta las posibilidades y libertad del sistema.

- **Personalización**: Android dispone de una interfaz fácilmente modificable que podemos ir retocando para dejarla a nuestro gusto. Esta personalización va más allá de lo puramente estético: permite modificar muchas opciones de tipo funcional, con aplicaciones que nos permiten cambiar la forma en la que manejamos nuestros dispositivos o la forma en la que unas aplicaciones interactúan con otras.

- **Hardware variado**: Android se incluye en cientos de modelos de smartphones, con características y precios muy variados. Contamos con muchas posibilidades de elección.

- **Notificaciones**: Aunque todos los sistemas operativos móviles incluyen este sistema junto con una luz led de notificación, en este apartado Android es el líder teniendo un sistema totalmente personalizable y cómodo para organizarse con él.

- **Versatilidad**: Android soporta infinidad de tecnologías que lo adaptan para que su uso esté más presente cada día. Actualmente podemos encontrar Android no solo en smartphones y tablets, sino también en relojes, televisores, coches, sistemas deportivos, etc.

Además de todo lo anterior, nos permite utilizar un gran número de complementos que lo hacen aún más cómodo y atractivo como son: USB tipo-C, baterías extraíbles e intercambiables, SIM dual, tarjetas de memoria extraíbles y la posibilidad de conectar casi cualquier periférico por puerto USB con un simple adaptador.

Pero no todo son alabanzas hacia este sistema operativo. Android también posee ciertas carencias, algunas de ellas graves, que lleva años arrastrando. No son muchas, pero sí que pueden llegar a resultar importantes para algunos de sus usuarios. Entre ellas se encuentran:

- **Actualizaciones**: Si bien es cierto que existen ciertos dispositivos que reciben directamente las actualizaciones el día que son lanzadas, estos son conocidos como los Nexus, y son comercializados de la mano de Google junto con algunas prestigiosas marcas comerciales. Hay que destacar que para muchos modelos dichas actualizaciones tardan meses en llegar, si es que llegan, ya que a menudo se quedan en el camino.

- **Capas de usuario**: Cualquier usuario que haya probado Android puro puede decir que es realmente atractivo, pero los fabricantes tienen la mala costumbre de modificarlo incluyendo sus propios diseños y una serie de aplicaciones preinstaladas que el usuario no suele desear tener en su teléfono y que pueden ser realmente molestas, ya que no pueden eliminarse y ocupan una valiosa parte de la memoria.

- **Rendimiento**: Android consume más recursos que sus competidores para ofrecer el mismo rendimiento, lo que implica aumentar los costes en los materiales de fabricación, además de una reducción en la duración de la batería.

- **Malware**: Lo cierto es que la seguridad en Android no es perfecta. A veces se cuelan aplicaciones maliciosas incluso en la tienda oficial y resulta difícil parchear las vulnerabilidades.

2.3. Usos de Andoid

Aunque existen países como Japón que se resisten al uso generalizado de Android, la realidad es que este sistema operativo es la plataforma más utilizada en el resto de países, estando en algunos de ellos por encima del 80 o 90% del uso total de smartphones.

A continuación se muestran datos de su uso en ciertos lugares en el primer trimestre de 2016:

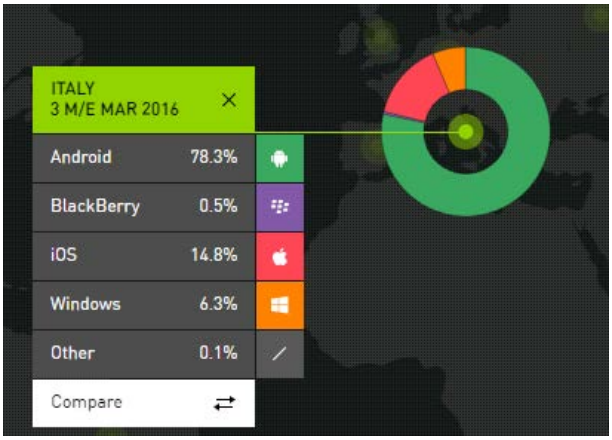

Ilustración 2: Uso de Android en Italia.

Recuperado en: http://www.kantarworldpanel.com/es

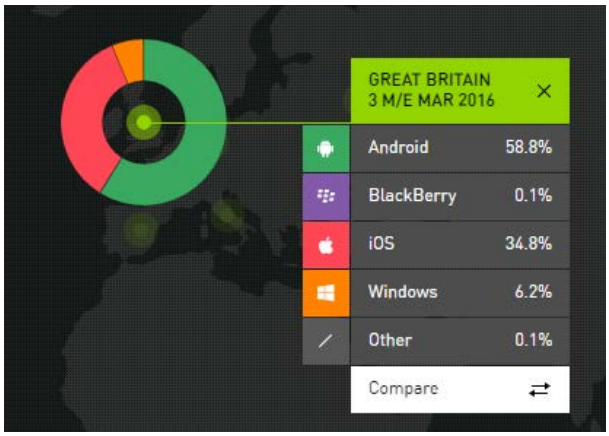

Ilustración 3: Uso de Android en Gran Bretaña.

Recuperado en: http://www.kantarworldpanel.com/es

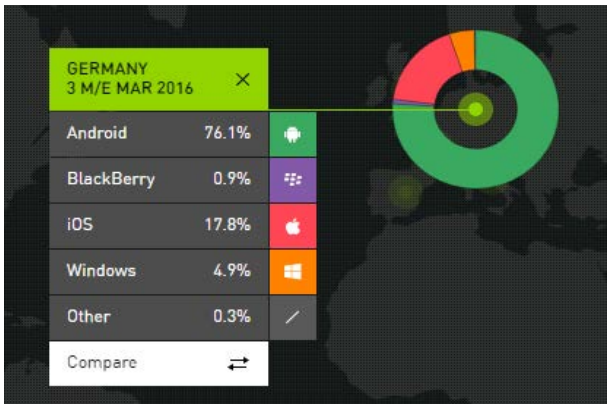

Ilustración 4: Uso de Android en Alemania.

Recuperado en: http://www.kantarworldpanel.com/es

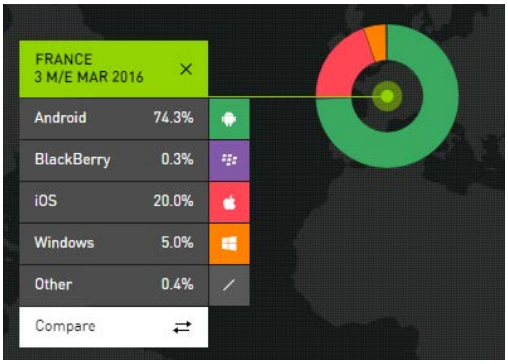

Ilustración 5: Uso de Android en Francia.

Recuperado en: http://www.kantarworldpanel.com/es

Estos son algunos de los datos ofrecidos por países como Italia, Gran Bretaña, Alemania y Francia, donde se puede observar claramente como Android es la plataforma predilecta por todos ellos.

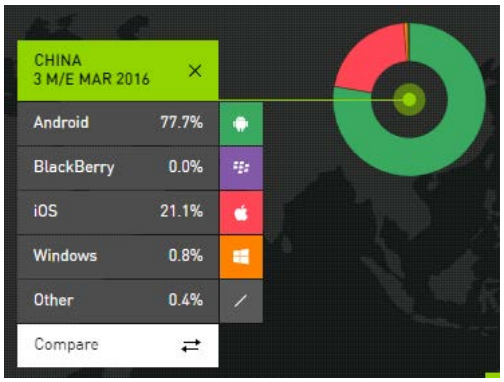

Ilustración 6: Uso de Android en China.

Recuperado en: http://www.kantarworldpanel.com/es

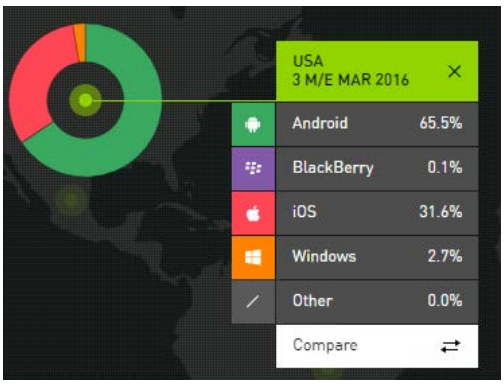

Ilustración 7: Uso de Android en Estados Unidos.

Recuperado en: http://www.kantarworldpanel.com/es

Por otra parte, en los dos grandes mercados fuera de la UE Android sigue dominando. Quizá los datos más abusivos se registren aquí, en nuestro país, España:

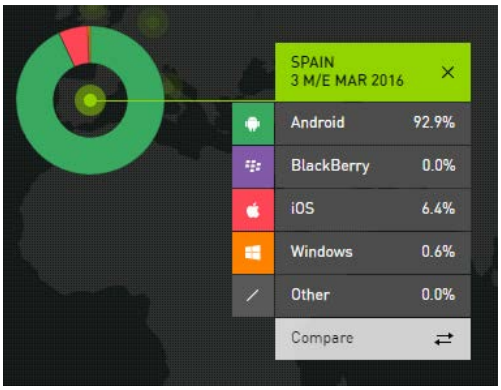

Ilustración 8: Uso de Android en España.

Recuperado en: http://www.kantarworldpanel.com/es

En España, como se puede apreciar, Android domina claramente el mercado con una cuota del 92,9%, es decir, que en nuestro país más de 9 de cada 10 teléfonos utilizan el sistema operativo Android.

Si profundizamos un poco más en el uso de Android en nuestro país, podemos ver el porcentaje de teléfonos que disponen de cada versión de éste:

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%

Data collected during a 7-day period ending on May 2, 2016.
Any versions with less than 0.1% distribution are not shown.

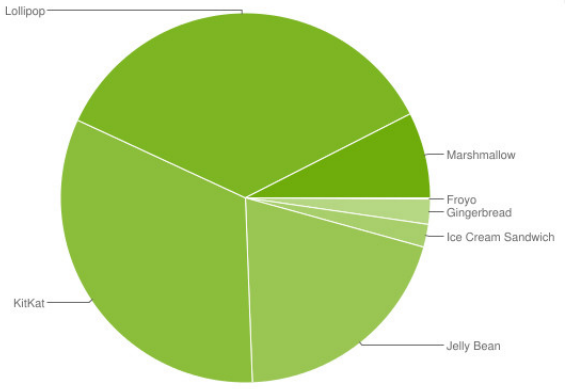

Ilustración 9: Distribución de las versiones de Android.

Recuperado en: http://www.xatakandroid.com/mercado/el-7-5-de-los-dispositivos-android-ya-estan-actualizados-a-marshmallow

Aquí se aprecia que la primera versión que podría considerarse utilizada sería la 2.3, la cual será la primera para la que funcione nuestra aplicación.

La dominante es la 5.0 “Lollipop” seguida muy de cerca por la 4.4 “KitKat” aunque el crecimiento de la 6.0 “Marshmallow” que, solo lleva unos meses en activo, hace pensar que en poco tiempo supere a éstas.

Por tanto nuestra aplicación ha de ser compatible con ésta última actualización de Android, implementando los cambios en el método de programación que esto supone y que se detallarán más adelante.

2.4. Aplicaciones similares ya existentes

Para tener un mejor concepto de la aplicación a realizar es necesario conocer las diversas opciones existentes que podría encontrar un usuario que necesitase un servicio de posicionamiento para el diseño de una línea de alta tensión.

En la actualidad existen numerosas aplicaciones de posicionamiento en las que podemos obtener las coordenadas de nuestra ubicación o las de un punto en concreto. Estas, en su mayoría, son gratuitas.

La mayoría de estas aplicaciones no están enfocadas al trazado de líneas de alta tensión. Tras buscar por el mercado de aplicaciones, se han podido hallar algunas que sí cumplen con dicha función, aunque a opinión propia, ésta tiene ciertas carencias que nuestra aplicación solventa y que serán expuestas más adelante.

Sin más dilación, pasamos a exponer las aplicaciones que más han llamado nuestra atención por su similitud a la hora de cumplir la función deseada en nuestra aplicación:

1- **Topography APP**: Esta APP, originariamente en inglés y ahora también en español, nos permite crear unas listas de puntos que posteriormente pueden ser exportadas según nos interese.

Ilustración 10: Icono aplicación Topography APP.

Recuperado en: https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

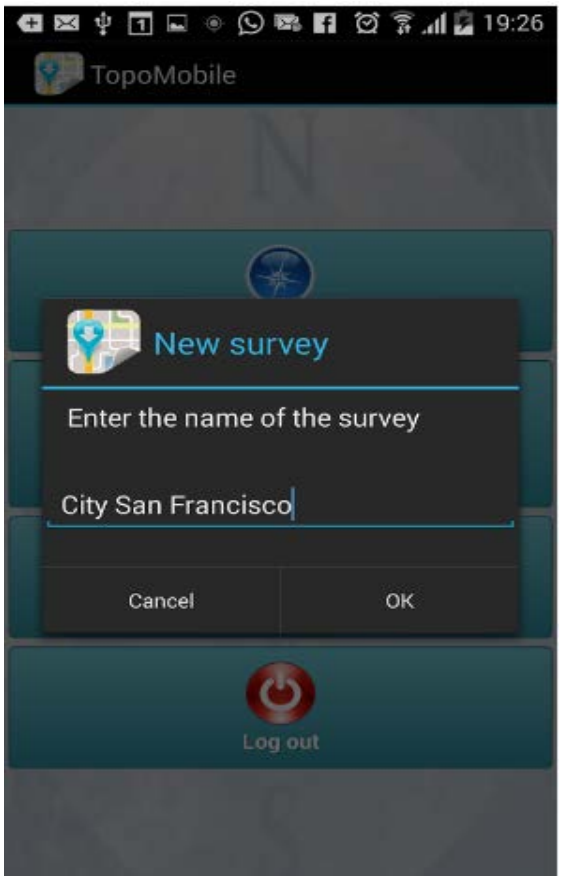

Ilustración 11: Creación de una "survey".

Recuperado en: https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

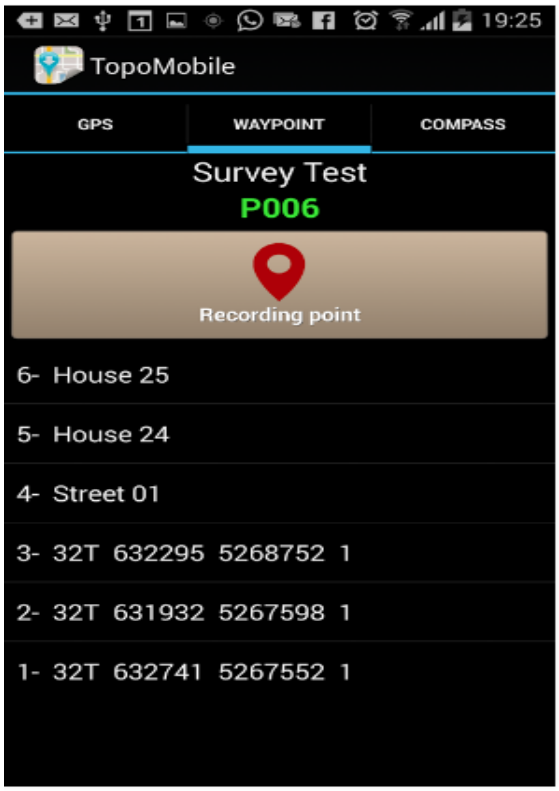

Ilustración 12: Interior de una lista con diversos puntos guardados.

Recuperado en: https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

Ilustración 13: Exportación de una lista.

Recuperado en: https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

Ilustración 14: Menú de opciones.

Recuperado en: https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

Como se aprecia en las imágenes, primero se debe de crear una "survey" o lista, dentro de la cual se incluirán los puntos. Estos puntos sólo nos proporcionarán nuestra ubicación actual incluyendo la altitud. Tras completar la lista de puntos podremos exportarla en 3 tipos diferentes de archivos, lo cual lo hace bastante interesante. Por último destacar la posibilidad de realizar cambios en las unidades de los datos obtenidos mediante una sección de ajustes.

2- **GeoGPS**: Una aplicación similar a la anterior, si bien es más simple, ésta nos ofrece la posibilidad de ver el punto que hemos guardado en el mapa. Permite obtener las coordenadas de un punto así como la altitud y después guardarlo en una lista. Podremos visualizar cada punto el mapa si lo deseamos. La lista con los puntos se puede exportar en formato de texto o ".txt". Esta aplicación solo está disponible en polaco, dificultando así su uso para personas que no estén familiarizados con el idioma.

Ilustración 15: Icono Aplicación GeoGPS.

Recuperado en:

https://play.google.com/store/apps/details?id=aguiar.com.br.point_gps

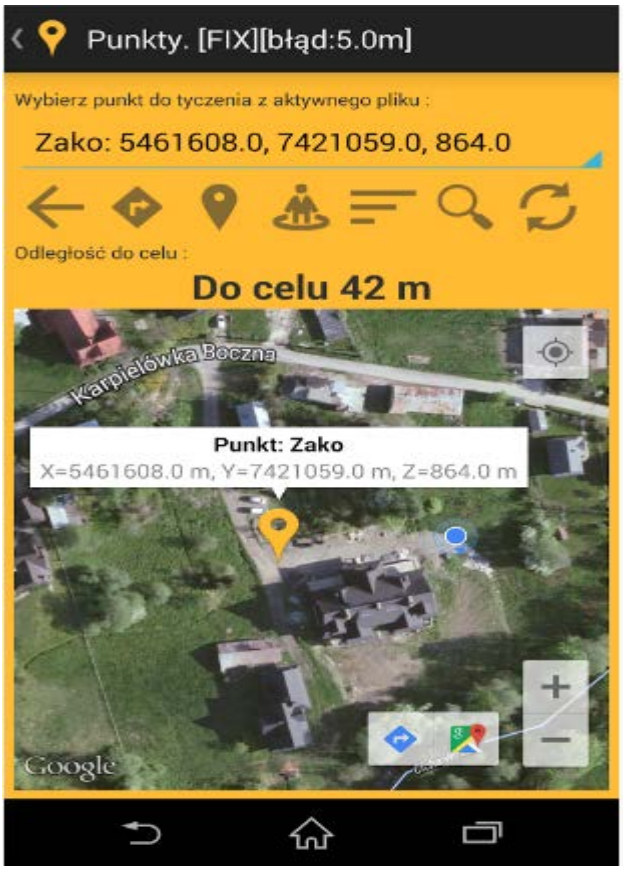

Ilustración 16: Visualización de un punto.

Recuperado en: https://play.google.com/store/apps/details?id=eu.encomit

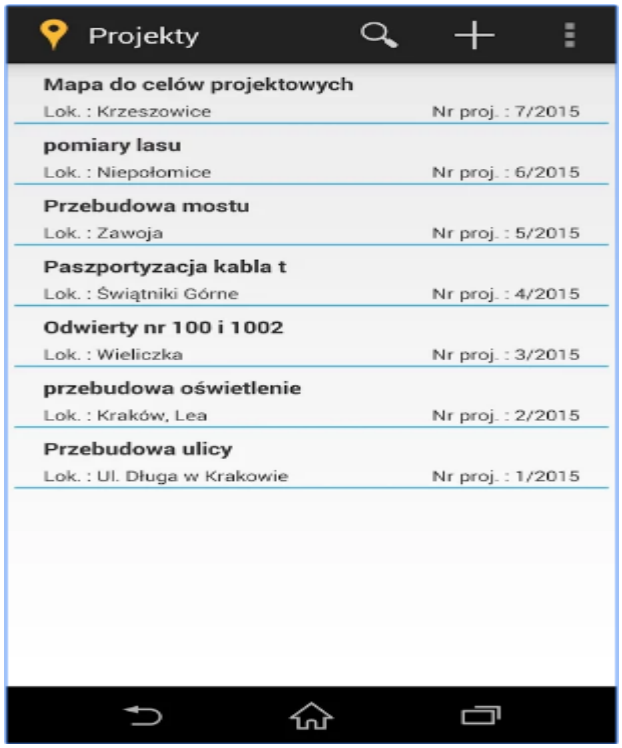

Ilustración 17: Guardado de punto.

Recuperado en: https://play.google.com/store/apps/details?id=eu.encomit

3- **Accurate Altimeter**: Esta aplicación, diseñada inicialmente para indicarnos la altitud de nuestra ubicación, también puede complementarse con nuestros requerimientos. Actualmente permite obtener nuestra ubicación en coordenadas UTM además de nuestra altitud. La posibilidad de guardar cada punto en el que nos hallemos ubicados y exportar todos esos puntos en formato ".txt", ".dxf", y ".kml" hacen también interesante a esta.

Ilustración 18: Icono Accurate Altimeter.

Recuperado en: https://play.google.com/store/apps/details?id=com.thsoft.altitude

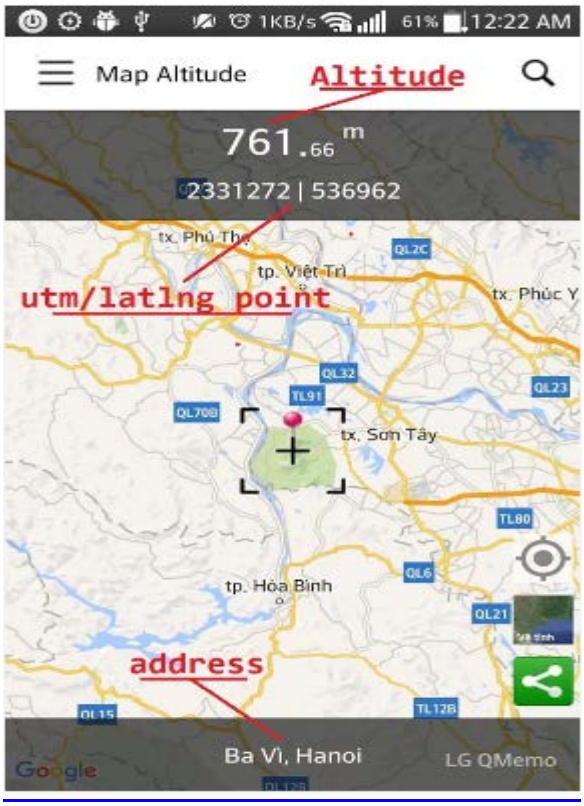

Ilustración 19: Datos del punto.

Recuperado en:

https://play.google.com/store/apps/details?id=com.thsoft.altitude

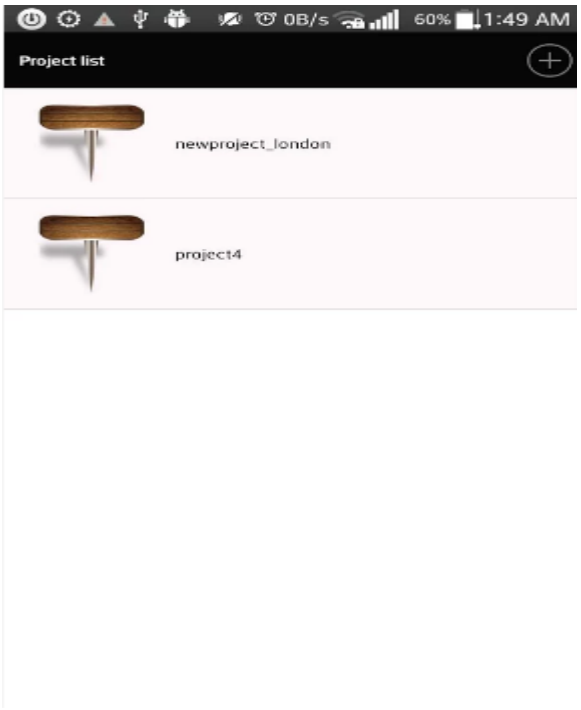

Ilustración 20: Lista de puntos.

Recuperado en:

https://play.google.com/store/apps/details?id=com.thsoft.altitude

4- **Tower Topography Lines**: Por último, una aplicación en la cual se especifica que uno de sus usos es el trazado de líneas de alta tensión. Ésta, al igual que los casos anteriores, permite crear listas con sus respectivos cada uno. En este caso, los datos de ubicación pueden obtenerse mediante GPS o pueden ser rellenados por el usuario. Además pueden asignarse a cada punto una serie de propiedades que facilitan su posterior interpretación. Por último también nos permite exportar las listas en formato ".dxf".

Ilustración 21: Icono Tower Topography Lines.

Recuperado en:http://play.google.com/store/apps/details?id=com.incubatum.app.towertl

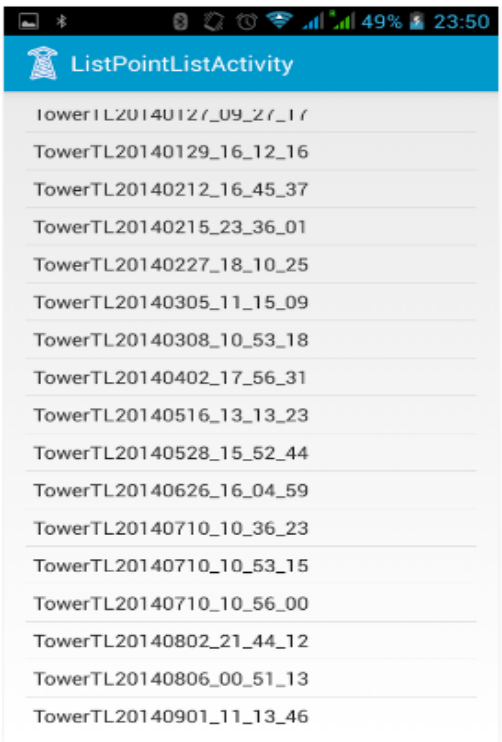

Ilustración 22: Lista de puntos.

Recuperado en:http://play.google.com/store/apps/details?id=com.incubatum.app.towerl

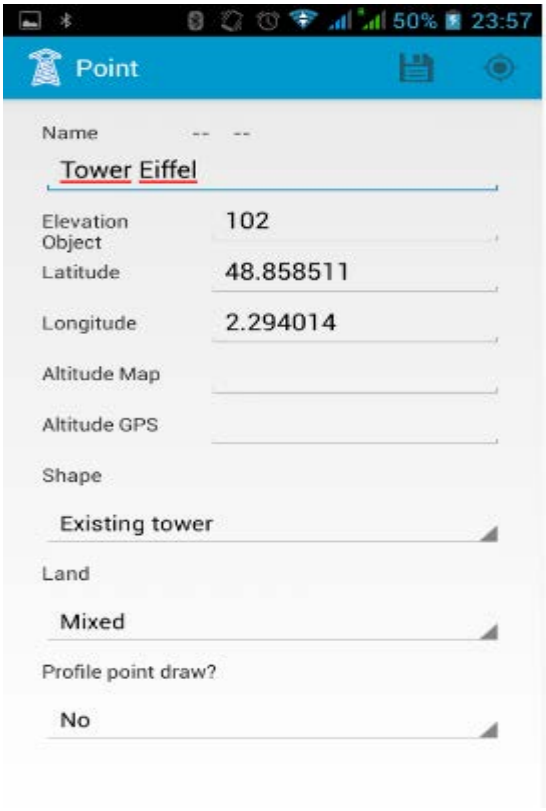

Ilustración 23: Propiedades del punto.

Recuperado en:

https://play.google.com/store/apps/details?id=com.incubatum.app.towertl

2.5. Elección e Instalación del software a utilizar

Como en todo lenguaje de programación, se necesita una herramienta de desarrollo que sea capaz de llevar a cabo todas las instrucciones que consideremos necesarias para realizar una programación lo más sencilla posible y sin fallos.

En el ámbito de Android, existen diversas posibilidades dependiendo también del lenguaje en el que nuestras aplicaciones son programadas.

Google eligió como lenguaje nativo Java y como entorno a Android Studio, pero esto no quiere decir que sea lo único que podemos utilizar.

Iniciarse en Android puede resultar muy complicado si no tenemos conocimiento de Java ni del entorno por eso existen otras herramientas que nos pueden ayudar, aunque quizás no tengan la misma amplitud que las escogidas por Google como nativas.

Las más populares son:

- **Basic 4 Droid:** Es una de las grandes enemigas de Android Studio en la que se utiliza el lenguaje VisualBasic. Posee un entorno más gráfico por lo que se pueden observar los avances conforme se programa.

- **App Inventor:** Una herramienta para quien no tenga conocimientos mínimos de programación, compuesta por un entorno visual en el que no hace falta escribir ni una sola línea de código para desarrollar una app. Uno de los grandes inconvenientes es también su punto fuerte, la simplicidad y rapidez a la hora de crear las aplicaciones

- **LiveCode:** Herramienta que permite programar tanto en Android, como en otras plataformas, por lo que comercializar la aplicación será mucho más sencillo y se conseguirá un mercado mucho más amplio si estás familiarizado con ésta.

- **HTML5:** Un lenguaje de programación basado en la programación de páginas web HTML, permite al usuario crear aplicaciones web con una increíble versatilidad, además de ser un contenido multiplataforma.

- **Appcelerator Titanium:** Es una de las plataformas más activas del momento, con más de medio millón de usuarios activos y más de 20.000 aplicaciones en el mercado. Se trata de una herramienta con la que sepuede aprende a desarrollar fácilmente aplicaciones completamente funcionales. Como referencia de calidad, grandes firmas comerciales como Ebay o PayPal, desarrollan sus aplicaciones móviles con esta herramienta.

- **Eclipse:** Es una gran plataforma de desarrollo. Fue la más utilizada en los inicios de la programación en Android, hasta la aparición de Android Studio. Puede definirse como un entorno genérico al ser compatible con diversos lenguajes de programación y cuyo diseño sigue un patrón de actualización basado en plugins, es decir, en añadir una herramienta (plugin) para la tarea que se quiera realizar. Aunque es una plataforma

que proporciona al programador una experiencia completa a la hora de programar, ha quedado relegada a un segundo plano al estar "desfasada" respecto a la nueva herramienta de Google. No es tan intuitiva ni permite tanta rapidez a la hora de solucionar problemas.

- **NetBeans:** Se trata de un entorno muy similar a Eclipse que cuenta con una gran base de usuarios que forman una comunidad muy amplia en todo el mundo. Ambos poseen una estructura similar, poseen un código abierto y gratuito. Pero, al igual que Eclipse, ha quedado un tanto desfasado y no es tan intuitivo como el moderno Android Studio.

- **Android Studio:** Android Studio es el nuevo entorno de desarrollo creado por Google en mayo de 2013 que permite facilitar la tarea de programación de su sistema operativo y cuya finalidad es unificar a los programadores. A partir de este momento, toda la documentación oficial ofrecida a los desarrolladores que giraba en torno a Eclipse pasa a estar basada en Android Studio. Esta plataforma, que podremos instalar tanto en Windows como Linux y OS, está basada en la idea IntelliJ, el cual supone unos cambios muy importantes para el programador. Sin ir más lejos ésta nos permite:
  - Sistemas de construcción y ayudas para codificación: Nuevo sistema de construcción y empaquetación de un proyecto Android. Mejorado el análisis de código donde el editor resalta las advertencias y los errores de manera inmediata, permitiendo aplicar una solución más rápida.
  - Pre-visualización y generación de recursos: Permite una finalización de código inteligente y un autocompletado más sofisticado y sensible. Situándose sobre una parte visual se puede cambiar rápidamente la densidad o la resolución, entre otras cosas.
  - Herramientas integradas de Android: Posee un diseñador de interfaz de usuario que permite arrastrar y soltar elementos y apoyar a distintos diseños y tipos de pantalla. Incluye diálogos para facilitar la generación de recursos.

- Aumento de productividad: Además de las herramientas comentadas anteriormente, Google ha facilitado el uso de sus recursos tales como Maps, Drive, etc.

En nuestro proyecto, todo lo citado anteriormente resulta fácilmente integrable por el programador, lo que permite obtener aplicaciones más completas y eficaces.

Tras estudiar detenidamente cada una de las plataformas, la elegida para la realización de nuestra aplicación será Android Studio.

La decisión se hace bastante obvia al contemplar las características y el soporte que posee esta, además de la facilidad del usuario para programar con ella.

A continuación se procede a la explicación de la instalación del software Android Studio.

- ***Paso 1: Descarga e instalación de Java***

Primero será necesario poseer en nuestro equipo la herramienta de desarrollo de Java: JDK (Java Development Kit) en una de sus versiones, preferiblemente la más avanzada. Ésta se puede descargar desde la página web de Oracle eligiendo nuestro sistema operativo.

Java SE Development Kit 8u91

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.72 MB	jdk-8u91-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.69 MB	jdk-8u91-linux-arm64-vfp-hflt.tar.gz
Linux x86	154.74 MB	jdk-8u91-linux-i586.rpm
Linux x86	174.92 MB	jdk-8u91-linux-i586.tar.gz
Linux x64	152.74 MB	jdk-8u91-linux-x64.rpm
Linux x64	172.97 MB	jdk-8u91-linux-x64.tar.gz
Mac OS X	227.29 MB	jdk-8u91-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.59 MB	jdk-8u91-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.95 MB	jdk-8u91-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.29 MB	jdk-8u91-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u91-solaris-x64.tar.gz
Windows x86	182.29 MB	jdk-8u91-windows-i586.exe
Windows x64	187.4 MB	jdk-8u91-windows-x64.exe

Ilustración 24: Versiones de JDK.

Recuperado en: www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

- ***Paso 2: Descarga e instalación de Android Studio y el SDK de Android***

A continuación se descargará Android Studio dirigiéndonos a la web de desarrolladores de Android y eligiendo la versión más reciente que coincida con nuestro sistema operativo y aceptando los términos de licencia.

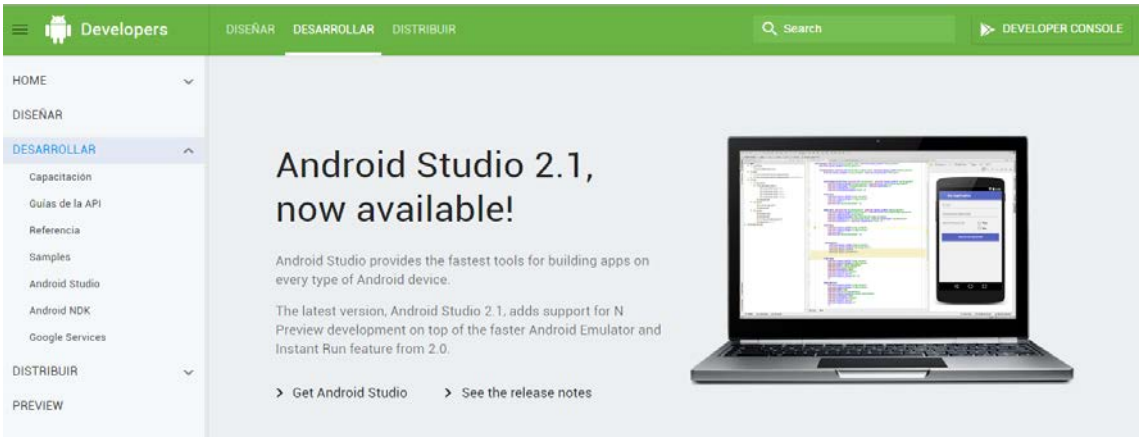

Ilustración 25: Web de descarga de Android Studio.

Recuperado en: http://developer.android.com/studio/index.html

Acto seguido se descargará en nuestro ordenador el ejecutable en el cual iremos siguiendo los pasos para una correcta instalación del software.

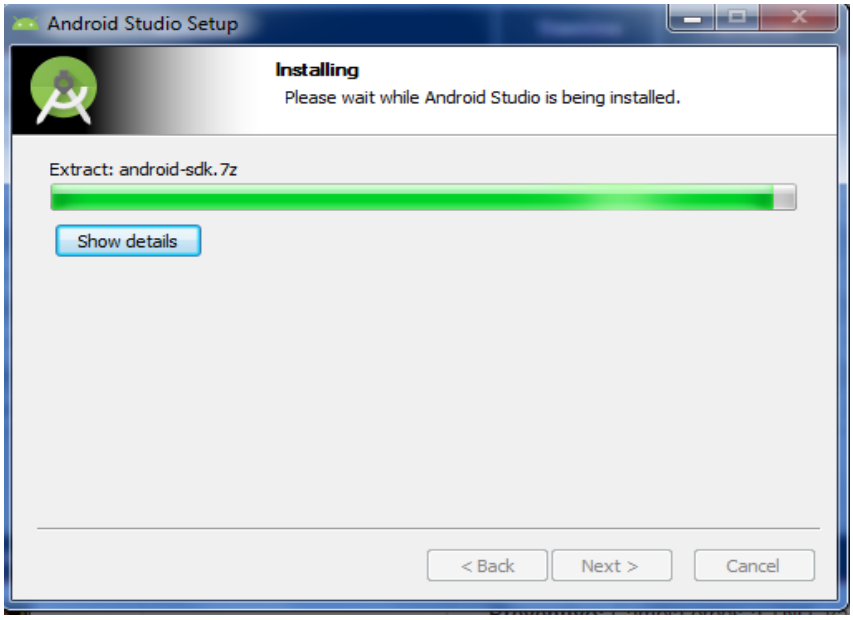

Ilustración 26: Proceso de instalación de Android Studio.

Elaboración Propia

Como se puede observar la instalación se corresponde con una estándar, al menos en Windows.

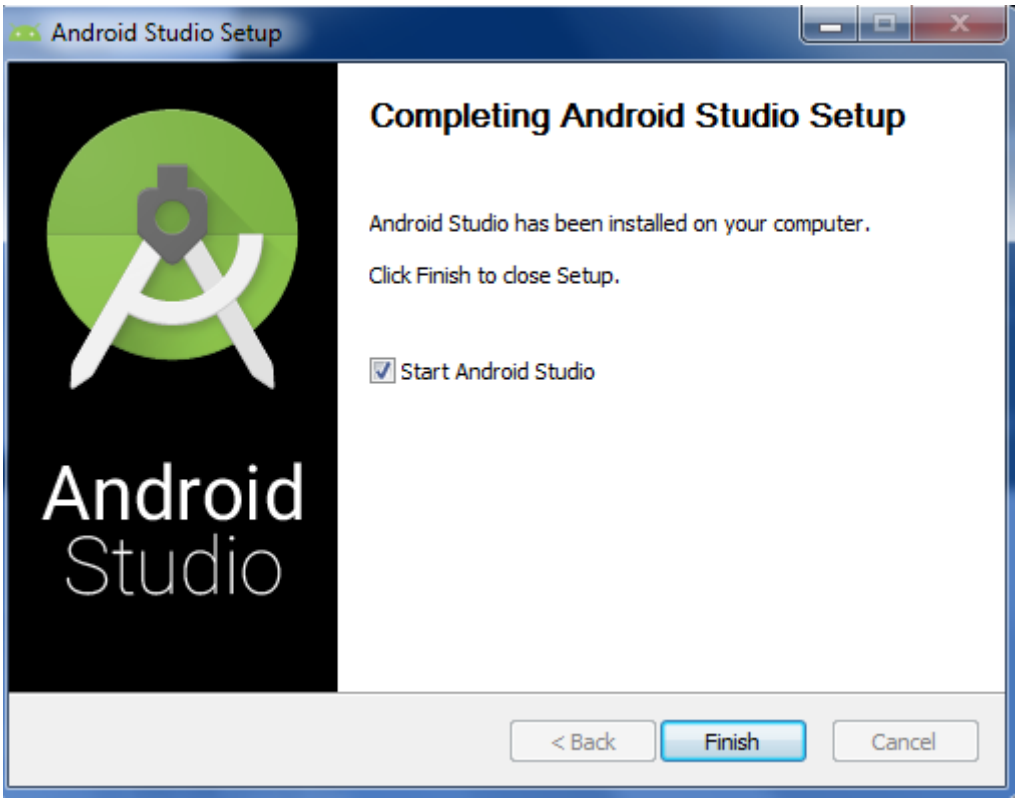

Ilustración 27: Instalación de Android Studio completada.

Elaboración Propia

La primera vez que se inicie en el Entorno de **Android Studio** se nos pedirá revisar una configuración inicial que consistirá básicamente en seleccionar los distintos útiles que queremos descargar para su posterior uso en la programación. Entre éstos se encuentran:

- **SDK de Java**: Las siglas SDK equivalen a Sofware Development Kit (Kit de desarrollo de software). Cada vez que aparece una nueva versión de Android Google libera el código fuente y publica el SDK. Así los desarrolladores podrán adaptar su aplicación a esta nueva versión. Se observa cómo se pueden incluir los SDK de las diferentes versiones en Android Studio, Además de las diversas herramientas de google incluidas en Google Play Services, Google USB driver o Android TV.

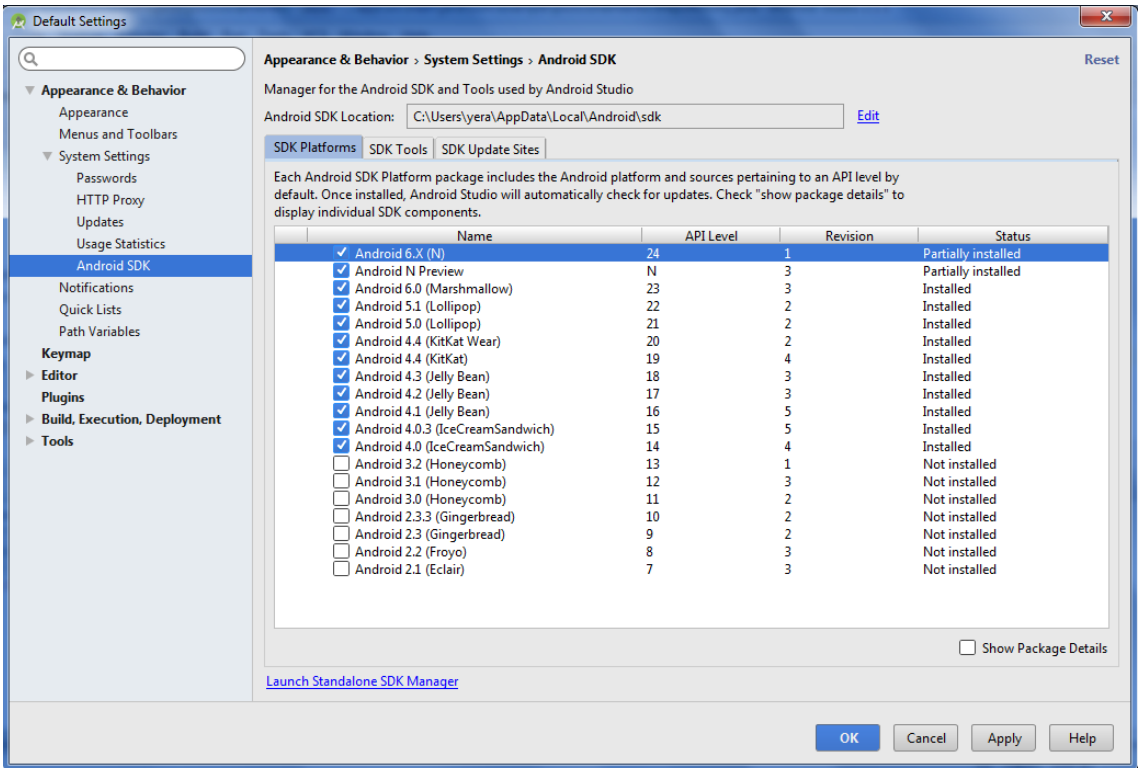

Ilustración 28: Android Studio SDK Manager: Versiones de Android.

Elaboración Propia

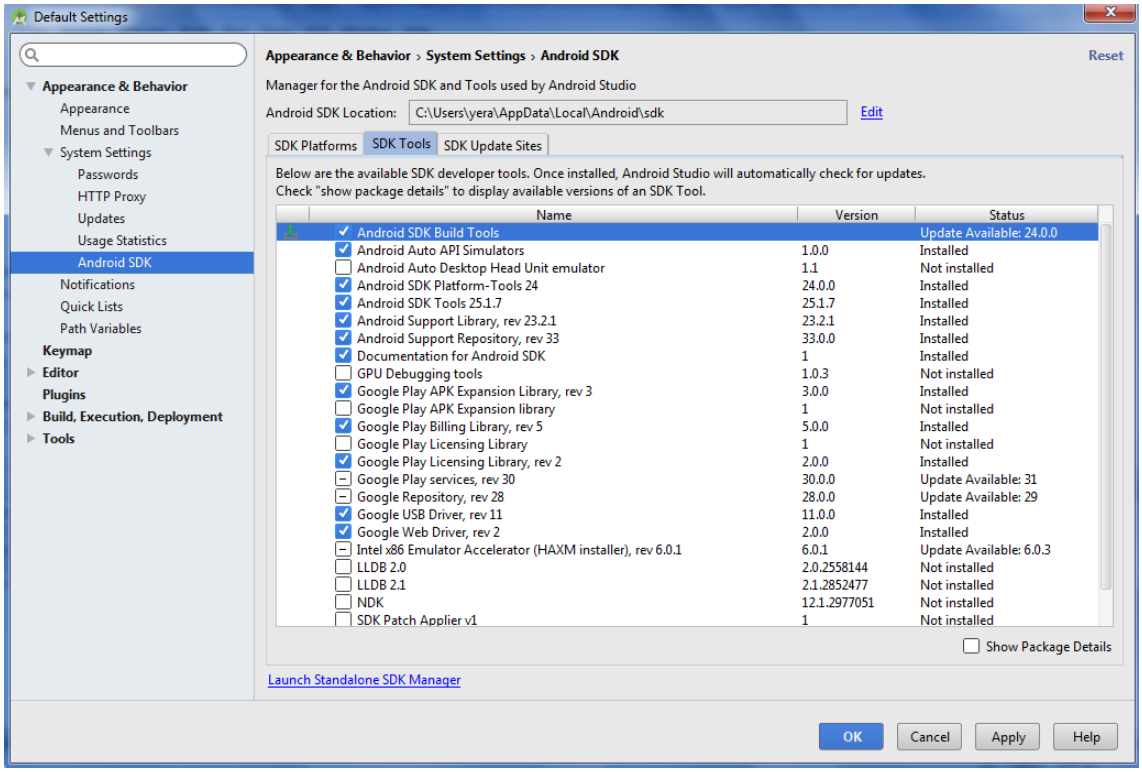

Ilustración 29: Android Studio SKD Manager: Herramientas Adicionales.

Elaboración Propia

Para el desarrollo de una aplicación, hemos de conocer cuáles de estas herramientas serán necesarias para implementarlas en nuestro software. Aunque no las implementemos desde un principio, más tarde podrán ser añadidas, además éstas se han de actualizar de una forma periódica y manual por parte del usuario. Para ello existe un botón en el menú superior.

- **AVD o Smartphone propio:** Lo ideal cuando un usuario desea probar su aplicación para saber si está programada correctamente o no, es crear un archivo .APK para posteriormente transferirlo al teléfono móvil objetivo y ahí comprobar su funcionamiento. El problema surge cuando queremos probar nuestra aplicación cada vez que le realizamos un cambio, por pequeño que sea, y tenemos que realizar este proceso repetidas veces, se vuelve muy tedioso.

Android Studio, propone dos formas de solucionar este problema facilitando así en gran medida el trabajo del programador a la hora de testear su aplicación.

La primera consiste en la creación de un AVD (Android Virtual Device), es decir, una máquina virtual Android. En este caso, podremos simular un Smartphone entre muchos de los modelos elegibles y con la versión de Android que nos plazca, cosa que nos dará mucha ventaja. Podremos comprobar cómo se visualiza nuestra aplicación en distintos tamaños de pantalla, distintas configuraciones del sistema o distintas resoluciones, se trata de una herramienta indispensable para todo programador.

Los AVD se pueden crear al principio como configuración inical o cuando el usuario desee pulsando el botón AVD Manager.

Como segunda opción, Android Studio nos permite trabajar con nuestro propio Smartphone siempre y cuando esté conectado al ordenador de trabajo mediante un cable USB y estén sus pertinentes drivers instalados.

Esta será la opción elegida para este trabajo, ya que al trabajar con ubicaciones el AVD no es muy preciso, y además podemos comprobar de primera mano su funcionamiento en un terminal real. Cabe destacar que para poder trabajar con el teléfono de este modo tiene que estar activado el modo “depuración por usb” o modo desarrollador del mismo, cuya activación se encuentra disponible en el menú de ajustes de cada Smartphone Android.

2.6. Lenguaje de Programación

Para la elaboración de este trabajo se ha estudiado un lenguaje de programación desde cero, a pesar de que se poseen ciertos conocimientos básicos de la programación que han sido de gran ayuda.

Este estudio incluye la realización de dos cursos gratuitos online sin tutorización y la consulta de diversos tutoriales para apartados específicos, como pueden ser la inclusión en la aplicación de Google Maps o de las bases de datos del tipo SQlite.

Posteriormente se realizaron diversas consultas en una comunidad de programadores donde se exponen la mayoría de fallos que puede devolver una aplicación, con el fin de resolver los propios y poder concluir la nuestra.

Aunque las aplicaciones en Android se pueden programar en diversos lenguajes, el más utilizado y a su vez el más simple es el lenguaje Java. Éste

será el elegido para la elaboración de este trabajo ya que, aparte de su simplicidad, en Android Studio solo se puede programar con el lenguaje Java.

A continuación se destacan las características que definen a Java, lo diferencian de otros lenguajes de programación y lo hacen especialmente interesante para el desarrollo de aplicaciones:

2.6.1. Lenguaje JAVA

Java está basado en el lenguaje C, asique resulta muy familiar para los programadores que ya conocen este lenguaje. Además se han eliminado los elementos más confusos de C (punteros, macros, registros…), por lo que se consigue un leguaje de rápido aprendizaje que simplifica la programación y reduce el número de errores. Es soportado por casi la totalidad de plataformas existentes.

Java es un lenguaje de programación orientado a objetos desde su base, habiéndose demostrado que utilizando esta técnica se reduce el tiempo de desarrollo y se obtiene un producto de mayor calidad.

Todo en java son objetos: Cualquier definición de datos ha de serlo, toda función ha de estar dentro de un objeto, las librerías de funciones son objetos, etc.

Esto confiere ciertas ventajas, como por ejemplo:

1- La encapsulación de la información es oculta y modularizada.
2- La herencia de clases, definición de nuevas clases a partir de existentes.
3- Polimorfismo, un objeto puede ser tratado respecto a la clase que pertenece, pero también respecto a la clase de la que desciende.

Este lenguaje posee una gran oferta de APIs y herramientas para poder desarrollar una gran variedad de cosas. Además existen infinidad de librerías de código abierto para disfrute de los usuarios.

Por último, destacar la gran comunidad de desarrolladores, en ésta podemos encontrar desde páginas y foros de ayuda para la gente con problemas, hasta sitios donde buscan ideas y gente imaginativa para poder llevar a cabo nuevos proyectos para Android.

Para entender mejor el lenguaje Java, se propone un ejemplo de programación explicado y desarrollado:

Al ser Java un lenguaje orientado a objetos, resulta vital conocer cómo definir a éstos. Previamente vamos a definir dos conceptos:

- **Objeto**: entidad que dispone de unas propiedades (atributos) y comportamiento (métodos).
- **Clase**: define un tipo de objeto concreto

Una clase en java está formada a su vez por:

- **Atributos**: Propiedades del objeto. Almacenan algún tipo de información de éste. Definen su estado.
- **Constructor**: Método que se utiliza para inicializar un objeto.
- **Métodos**: Utilizados para modificar o consultar el estado de un objeto. Son equivalentes a las funciones en otros lenguajes.

Como concepto de programación en Java básico, se expone el siguiente ejemplo utilizado para definir un número complejo:

```
/** Clase que representa un número complejo. */
class Complejo {

    //declaración de atributos
    private double real, imaginario;

    //declaración de constructor
    public Complejo(double real, double imaginario) {
        this.real= real;
        this.imaginario= imaginario;
    }
    //declaración de métodos
    /** Transcribe el complejo a String.
     * @return string con la parte entera y la imaginaria
     */
    public String toString() {
        return real + "+"+ imaginario + "i";
    }
    /** Suma al complejo de este objeto otro complejo.
     * @param v  el complejo que sumamos*/

    public void suma(Complejo v) {
        real = real + v.real;
        imaginario = imaginario + v.imaginario;
    }
}
```

Ilustración 30: Definición de un número complejo.

Recuperado en*:* *www.androidcurso.com*

Una clase necesita almacenar información que defina al objeto a tratar, esto se hará en los atributos de la clase. En el ejemplo, un número imaginario ha de tener parte real y parte imaginaria.

En el siguiente paso se declara el constructor. Un constructor es un método que tiene el mimo nombre de la clase y sirve para inicializar el objeto.

Los valores separados por coma son los valores que adopta y que posteriormente modifica mediante instrucciones.

Para terminar se declaran los métodos de la clase. Los métodos son equivalentes a las funciones o procedimientos de otros lenguajes y son utilizados para modificar o consultar el estado de nuestro objeto. El método se empieza por una palabra que indica su visibilidad (private, protected o public) y la siguiente indica el tipo de dato que devuelve, en este caso String (cadenad e caracteres) y void (vacío o nada).

Creando un modelo a partir de clases y objetos, y adaptándolo a nuestras necesidades, se puede llegar a programar casi cualquier aplicación. Este es el método adoptado para este trabajo, si bien es cierto que consta de construcciones mucho más complejas que el ejemplo que acabamos de observar, todo puede reducirse a lo anteriormente explicado.

2.6.2. Lenguaje XML

El lenguaje **XML** es otra de las herramientas usadas a la hora de programar en Android. Sus siglas significan eXtensible Markup Lenguage, es decir, lenguaje de marcado extendible. Es un lenguaje basado en etiquetas descriptivas cuya finalidad es la de representar información de texto de manera jerárquica, y a su vez darle formato. Este lenguaje surge para brindarnos la oportunidad de intercambiar información entre dos sistemas con distinto funcionamiento, esto es conocido como interoperabilidad.

Inicialmente se creó para manejar y estructurar opciones de configuración para sistemas operativos o aplicaciones. Posteriormente se impuso como herramienta para intercambiar información entre distintas tecnologías, debido a que establece un solo formato para intercambiar datos. Esto hace mucho más fácil importar y exportar información.

Como cualidades de este lenguaje podemos destacar:

- **Independencia de datos y forma**: Con este lenguaje se puede establecer la estructura de la información y al mismo tiempo determinar la forma en la que se interpretará dependiendo del contexto. Es decir, si tenemos un entorno A y un entorno B que necesitarán los datos en un entorno XML, solo hay que definir los datos y cambiara su forma dependiendo de cada contexto.

- **Extensibilidad**: En cualquier momento se pueden añadir nuevos componentes y atributos a un archivo XML.

- **Interoperabilidad**: La característica más importante de XML. Ya que permite intercambiar información entre aplicaciones, módulos o sistemas sin importar la arquitectura o tecnología usada.

En cuanto a la estructura, un documento XML está formado por datos de caracteres y marcado, el marcado lo forman las etiquetas:

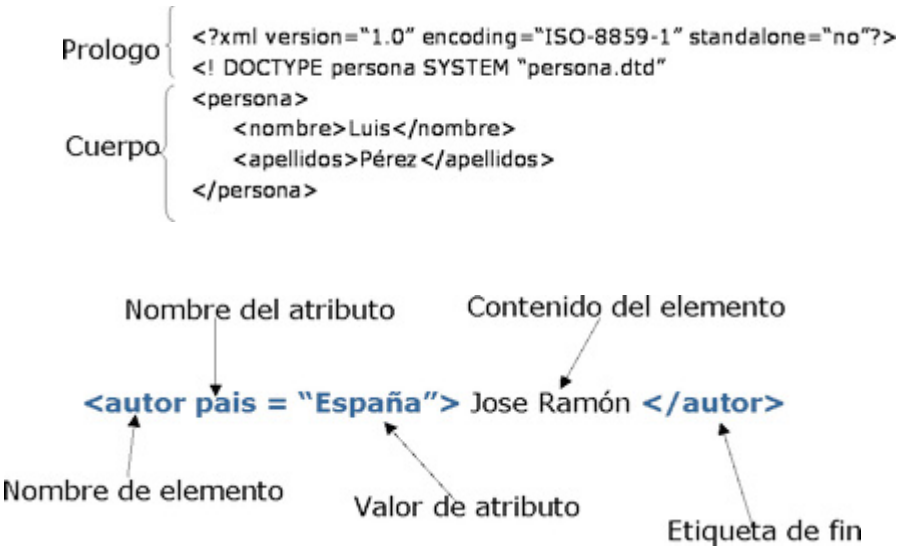

Ilustración 31: Estructura del código XML.

Recuperado en: http://www.mundolinux.info/que-es-xml.htm

Como componentes de un de un documento XML podemos destacar:

- **Elementos**: Parte lógica del marcado, se representa con una cada de texto encerrada entre etiquetas. Pueden existir elementos vacios (<br/>). Los elementos pueden contener atributos.
- **Instrucciones**: Órdenes que serán utilizadas por la aplicación que las procese.
- **Instrucciones de XML**: Órdenes propias del documento, comienzan por < y terminan por />.
- **Comentarios**: Información que no forma parte de la programación del documento, sirven como aclaraciones al programador.
- **Declaraciones de tipo**: especifican alguna información acerca del documento, suelen estar incluidas en el prólogo.

Tras esta introducción, se explica cómo se incluye éste código XML en la programación en Android. Para ello, primero se ha de explicar qué es el "Layout" en Android. El Layout es un objeto que representa el espacio contenedor de todas las vistas (Views) dentro de una actividad. En él se definen el orden y la estructura de los componentes para que el usuario pueda interactuar con la interfaz. Estos elementos se representan como subclases de Java que heredan de la clase ViewGroup.

Para la definición de estos elementos, se escribirá un documento XML que inicialmente contenga algún tipo de Layout para luego incluir en su interior los elementos "hijos". En Android Studio se pueden generar estos elementos sin necesidad de escribir a través del panel de diseño.

Ahora aparece la necesidad de aclarar una cuestión: ¿No se supone que Java es el código con el que programar en Android Studio? La respuesta es sí, Java es el encargado de toda la programación en este aspecto.

La introducción de XML supone a los proyectos Android una forma de reducir la complejidad del diseño de la interfaz, permitiéndonos declarar la estructura en un simple archivo definido por elementos muy intuitivos. Estos archivos son posteriormente analizados por un elemento compilador (librería de Parsing XML) que genera automáticamente todo el código Java necesario. De esta forma utilizamos un estilo sencillo para crear nuestra interfaz, y posteriormente utilizamos el código Java para implementar los elementos sobre la interfaz, teniendo así mucha flexibilidad para programar.

Para continuar se destacan los tipos de Layout más utilizados en Android:

1- **LinearLayout:** El LinearLayout es el más sencillo. Dentro de él los elementos están ubicados en forma linear a través de columnas o filas. Posee un atributo que permite modificar su orientación dependiendo de cómo queramos presentar los elementos vertical u horizontalmente.
2- **RelativeLayout**: Este es el más recomendado a usar, ya que permite distribuir los elementos dentro de él de una forma relativa entre sí.
3- **WebView**: Necesitaremos utilizar este Layout si queremos mostrar contenido en formato web.

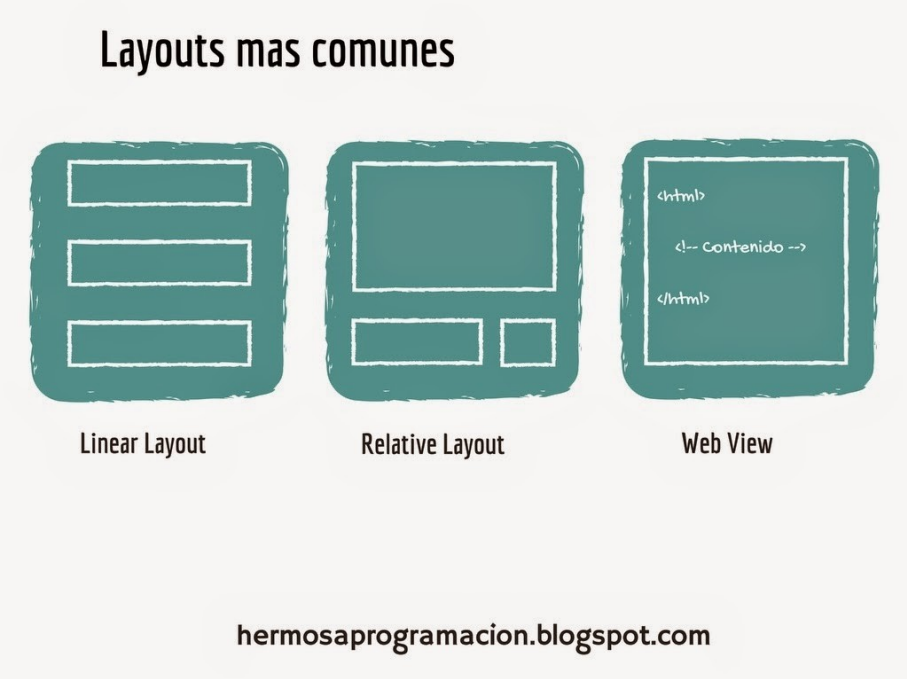

Ilustración 32: Layouts más comunes.

Recuperado en: www.hermosaprogramacion.blogspot.com

Existen numerosos elementos que podemos añadir a un Layout en Android, desde un TexView en el cual simplemente nos mostrará un texto, hasta botones desplegables o ventanas emergentes. El encargado de programar todos estos elementos visuales será el código XML.

Para finalizar se adjunta una imagen con la visualización de un Layout simple de forma programática (Texto XML) y otra imagen con el panel de diseño.

```
<LinearLayout  xmlns:android="http://...
      android:layout_height="match_parent"
      android:layout_width="match_parent"
      android:orientation ="vertical">
  <AnalogClock
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
  <CheckBox
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Un checkBox"/>
  <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Un botón"/>
  <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Un texto cualquiera"/>
</LinearLayout>
```

Ilustración 33: Visualización de un Layout simple de forma programática.

Recuperado en: http://www.androidcurso.com/index.php/tutoriales-android/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/114-layouts

Ilustración 34: Visualización de un Layout con el panel de diseño.

Recuperado en: http://hermosaprogramacion.com/2015/08/tutorial-layouts-en-android

2.7. Herramientas utilizadas en la programación

Para la elaboración de este trabajo se han utilizado diversas herramientas que requieren un estudio adicional, aparte del ya estudiado código Java y XML aplicado a Android.

A la hora de programar en Android, contamos de antemano con una amplia diversidad de utensilios que nos facilitan la programación de nuestra aplicación, pero éstos son genéricos. Por ejemplo contamos con: Botones, pestañas o menús desplegables, cuadros de texto, fondo de pantalla, reloj, imágenes sobre pantalla, etc. Todos ellos son personalizables por el usuario, tanto su apariencia como su texto o posición.

Sin embargo, cuando se desea trabajar en un campo más específico, estas herramientas se nos quedan cortas. Se necesitan de unas capaces de trabajar en estos campos con total libertad, que simplifiquen la programación cuando las herramientas convencionales no nos lo permiten. Algunas de estas herramientas adicionales han sido creadas e implementadas por la propia Google, aunque han de ser llamadas para poder operar con ellas ya que no están implementadas desde el principio. Otras en cambio, han sido desarrolladas por la comunidad de programadores y liberadas para uso y disfrute de cualquiera.

En este ámbito, se describen las utilizadas y que han resultado imprescindibles para la realización de este trabajo.

2.7.1. Google Maps API

Google Maps API: Ya que esta aplicación está basada en el posicionamiento, lo esencial es poder contar con un mapa preciso y fiable del terreno para poder trabajar con él. Para esta opción podemos contar con Google Maps ya que, al ser una aplicación de Google, esto facilita mucho las cosas a la hora de integrarlo en nuestro proyecto Android.

Esto no quiere decir que no haya que pasar por ciertas medidas que Google ha establecido para evitar el uso indebido de sus herramientas.

Primero deberemos registrar una cuenta en la web de desarrolladores de Google, indicando nuestros datos personales, correo electrónico y otros opcionales como empresa, página web, etc. Tras esto debemos solicitar una

clave o “key” para nuestro proyecto que consistirá en un código alfanumérico que deberemos introducir más adelante en un apartado específico de nuestro proyecto. Para obtener esta “key” debemos elegir el tipo de herramienta de Google que queremos usar, en nuestro caso la Google Maps API. Tras esto, indicar la finalidad de nuestro proyecto, la plataforma en la que se va a implementar y su nombre. Una vez hecho todo esto tendremos nuestra key lista para ser usada.

Una vez tengamos la key (se indicara más tarde como introducirla en el apartado permisos), ya podemos proceder a cargar el mapa. En nuestro trabajo el mapa supondrá la totalidad de la pantalla exceptuando una serie de botones ubicados en la parte inferior, asique será fundamental. Esta API de Google Maps, nos permite trabajar en nuestra aplicación como si de la propia aplicación de Google se tratara. Podremos desde movernos libremente por el mapa hasta ver edificios en 3D, colocar nuestros markers personalizados, cambiar el tipo de mapa, inclinar la vista con un cierto ángulo o incluso ver planos interiores de edificios. Una herramienta muy completa de la cual Google proporciona todo el código necesario y que, tras haberlo estudiado en profundidad resulta muy práctico para el programador.

Aplicando todo esto a nuestro trabajo, se utilizará esta API para:

1- Obtener el mapa principal con el que trabajar y en el que estará basada la aplicación.
2- Obtener la ubicación del punto deseado del mapa.
3- Fijar un marker una vez sepamos que el punto nos es de utilidad, ya que esto facilitará la visión de cómo transcurrirá la línea eléctrica o lo que se esté trabajando.

Ilustración 35: Logo de Google Maps.

*Recuperado en: /*www.google.com/intx/es/work/mapsearth

2.7.2. Base de datos SQlite

Todas las tareas relacionadas con el almacenamiento de datos de nuestra aplicación serán abarcadas por una base de datos SQLite. Este motor de bases de datos es muy popular en la actualidad por ofrecer características muy interesantes como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y por supuesto, ser de código libre. Podría decirse que SQLite es una librería que está incorporada en Android, así simplifica mucho la tarea del programador cuando desea trabajar con bases de datos.

Ilustración 36: Logo Base de datos SQLite.

Recuperado en: *https://sqlite.org/*

2.7.3. JSON

JavaScript Object Notation (JSON) es un formato ligero para el intercambio de datos. Se trata de un subconjunto del lenguaje Java que se basa en describir los datos con una sintaxis dedicada que nos permite identificar y gestionar dichos datos. Nació como una alternativa a XML y presenta como ventaja que puede ser leído por cualquier lenguaje de programación. Debido a su naturaleza compacta, resulta más sencillo y rápido trabajar con el que con XML.

Si nos centramos en nuestro trabajo, JSON nos permite enviar datos a cierta página web que usa un lenguaje determinado, y recibir la respuesta de ésta, pudiendo obtenerla en el lenguaje propio. Entrando más en detalle, al no poder facilitarnos Google Maps un valor de altitud para el punto deseado, se han de barajar otras opciones.

La opción más sencilla a la hora de conseguir un valor de altitud fiable consiste en introducir las coordenadas del punto que queremos (Latitud, Longitud) en un servicio web proporcionado por Google que nos devolverá la altitud de dicho punto en metros con una excelente precisión. Pues bien, JSON nos permite contactar con este servicio con apenas unas líneas de código. El único inconveniente para el usuario será el de tener que estar conectado a un servicio de internet móvil (ya sea por red telefónica de datos o por wifi) pero el valor de la altitud obtenido será realmente bueno.

Ilustración 37: Logo del formato JavaScript JSON.

Recuperado en: http://www.json.org/

2.7.4. Librerías Adicionales

Librerías adicionales: Antes se ha reseñado cómo añadir herramientas para poder llevar a cabo la programación de manera satisfactoria, éstas venían de la mano de Google. Pero no solo Google crea estas ayudas al programador.

Existen infinidad de comunidades o empresas que crean su propio contenido que puede ser descargado por cualquier usuario. Este suele ser un contenido más especializado que ayuda al programador en una tarea específica.

En este trabajo, se usan dos de éstas librerías, de código abierto y gratuitas. Dichas librerías son:

1- **MP Android Chart:** Poderosa herramienta de creación o visualización de gráficos en Android. Soporta numerosos estilos de gráficos, éstos son totalmente personalizables y se pueden animar o incluir movimientos en éstos. En nuestro caso utilizada para crear un perfil longitudinal de nuestros puntos basándonos en los datos de altitud guardados.

Ilustración 38: Logo de la Librería MPAndroidChart.

Recuperado en: *https://github.com/PhilJay/MPAndroidChart*

2- **Open CSV:** Herramienta que nos permite obtener valores en formato CSV (Coma Separated Values), es decir, valores separados por coma. Muy fácil de usar y muy útil puesto que nos permite obtener un formato universal para la salida de nuestros datos.

En nuestra aplicación nos permitirá convertir una base de datos de formato SQLite. Esta resulta complicada de abrir y leer fuera del formato Android a CSV, el cual podremos abrir fácilmente con Microsoft Excel.

Ilustración 39: Logo del formato de datos CSV.

Recuperado en: https://www.fundrecs.com/blog/csv.html

2.8. Estructura de una aplicación. Componentes

Una vez se tiene el entorno instalado y los conceptos de programación asimilados, procedemos a trabajar en nuestra aplicación. En este apartado se explican los componentes principales de Android Studio y donde hemos de ubicar cada elemento anteriormente descrito, incluyendo muchos ejemplos visuales para un mejor entendimiento de los mismos.

Si se observa el entorno generalizado de Android Studio, se pueden observar muchos componentes y herramientas. Para este trabajo no se ha hecho uso del total, por tanto sólo se procede a la explicación de los que han sido de utilidad o los que se consideran importantes para el desarrollo del mismo.

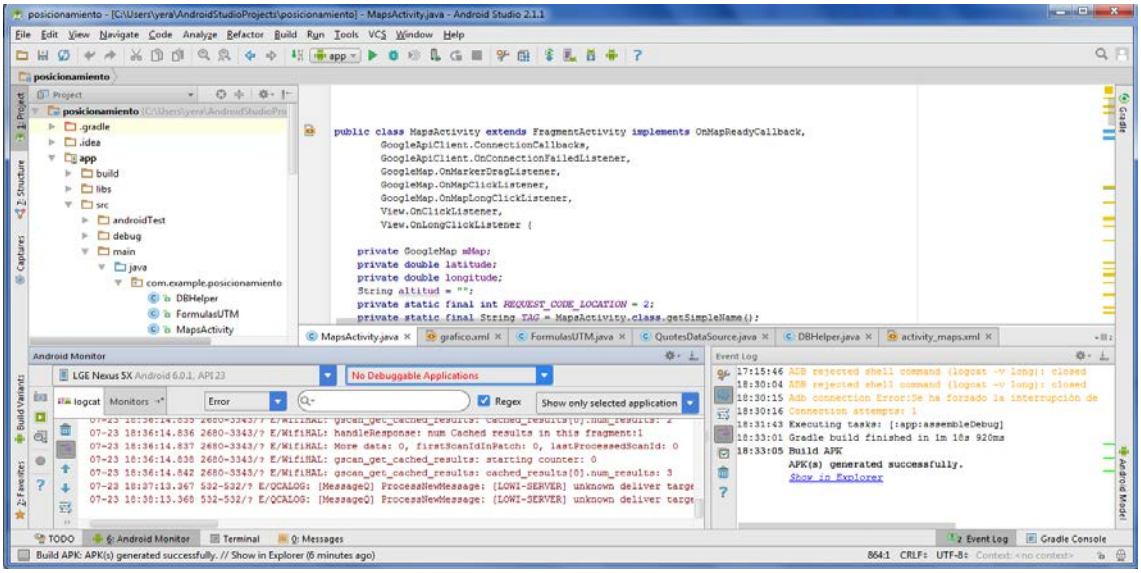

Ilustración 40: Entorno de programación en Android Studio.

Elaboración Propia.

Aunque a priori puede parecer dificultoso y largo el poder familiarizarse con este entorno, con conocer solo ciertas partes importantes se puede trabajar perfectamente y hacer una aplicación como la que en este trabajo se ha creado. Para ello vamos a desglosar esta pantalla en las diferentes partes utilizadas o las que se consideran más importantes para la programación de este trabajo.

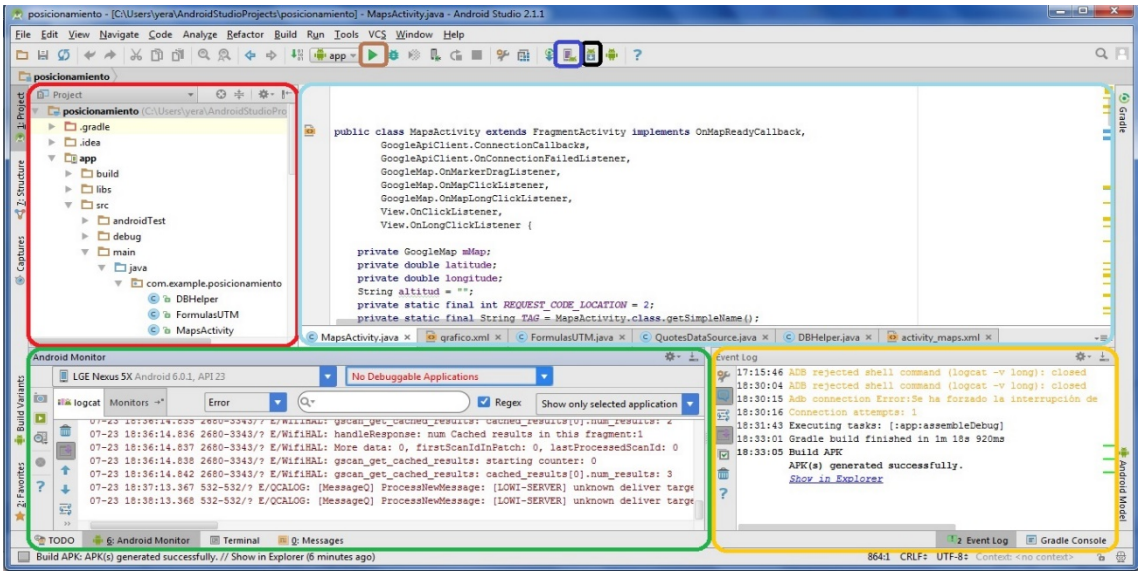

Ilustración 41: Distinción de los elementos más importantes del entorno.

Elaboración Propia.

- Estructura o “árbol” del proyecto..
- Ventana de escritura de código o programación.
- Monitor Android.
- Consola de eventos.
- Botón Android SDK.
- Botón AVD.
- Botón “RUN”.

Con estos elementos se obtiene una programación totalmente satisfactoria de nuestra aplicación. Al explicarlos por separado se consigue una comprensión más rápida de cómo funcionan entre sí dichos elementos.

2.8.1. Ventana de escritura de código o de programación.

Esta ventana es la parte principal de Android Studio, desde aquí se introducirá todo el código y se crearan todas las vistas de nuestra aplicación. Además nos ofrece ayuda con la programación y con los respectivos errores que puedan aparecer. Se trata del único elemento que no se puede ocultar en la ventana principal de Android Studio.

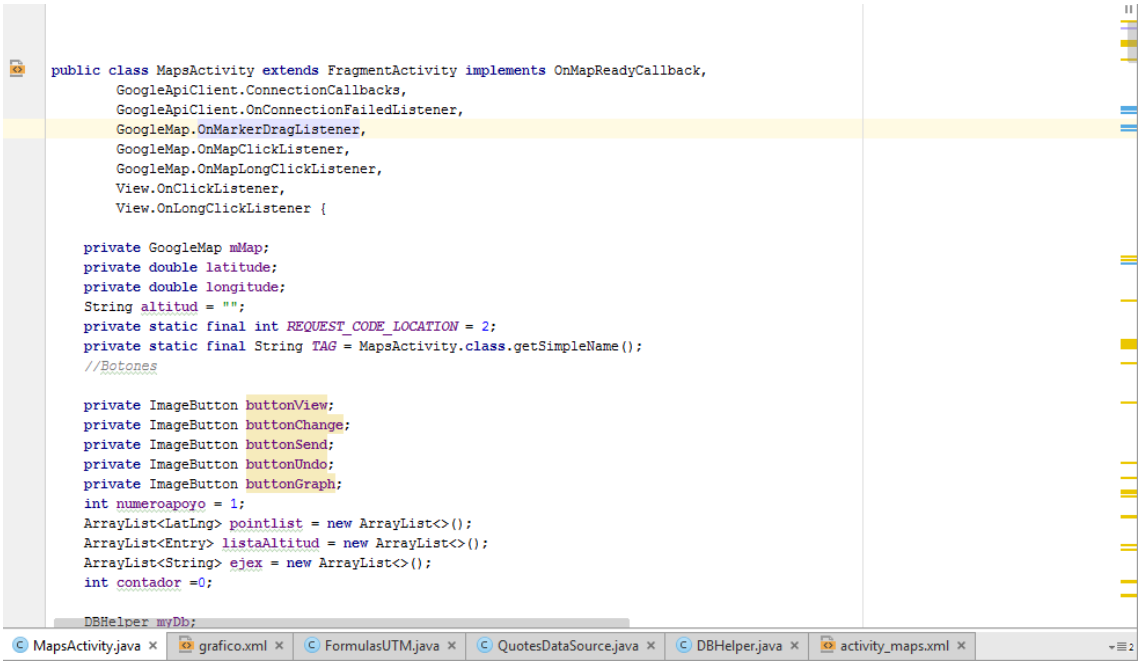

Ilustración 42: Ventana de Código Java.

Elaboración Propia.

Como se observa en la imagen, se pueden tener abiertas diferentes pestañas las cuales pueden corresponder a archivos .java o .xml ya que podemos programar en los dos lenguajes. Las pestañas .java corresponden a pestañas en las cuales el código cumple funciones u órdenes dentro de la aplicación. Las .xml corresponden a las diferentes vistas de la aplicación.

También podemos observar en el "scroll" o barra que nos permite subir o bajar dentro de la pestaña, un rayado de colores. Esta es la forma que tiene Android Studio de hacer varias indicaciones acerca del código. Las de color amarillo indican que, aunque el código es correcto y compilable, se puede mejorar o hacer que éste este menos cargado. Las de color rojo si las hubiere, indican que esa parte de código está mal programado y que Android Studio por tanto no será capaz de compilarlo. Las de color azul por su parte nos indican en que líneas de código se encuentra la variable que estemos seleccionando en este momento.

2.8.2. Estructura o "árbol" del proyecto

En esta parte de Android Studio, se encuentra cómo están distribuidas las distintas carpetas o secciones de las que está compuesta nuestra aplicación. A la hora de programar se han de utilizar ciertas carpetas o archivos que serían difíciles de buscar sin esta herramienta.

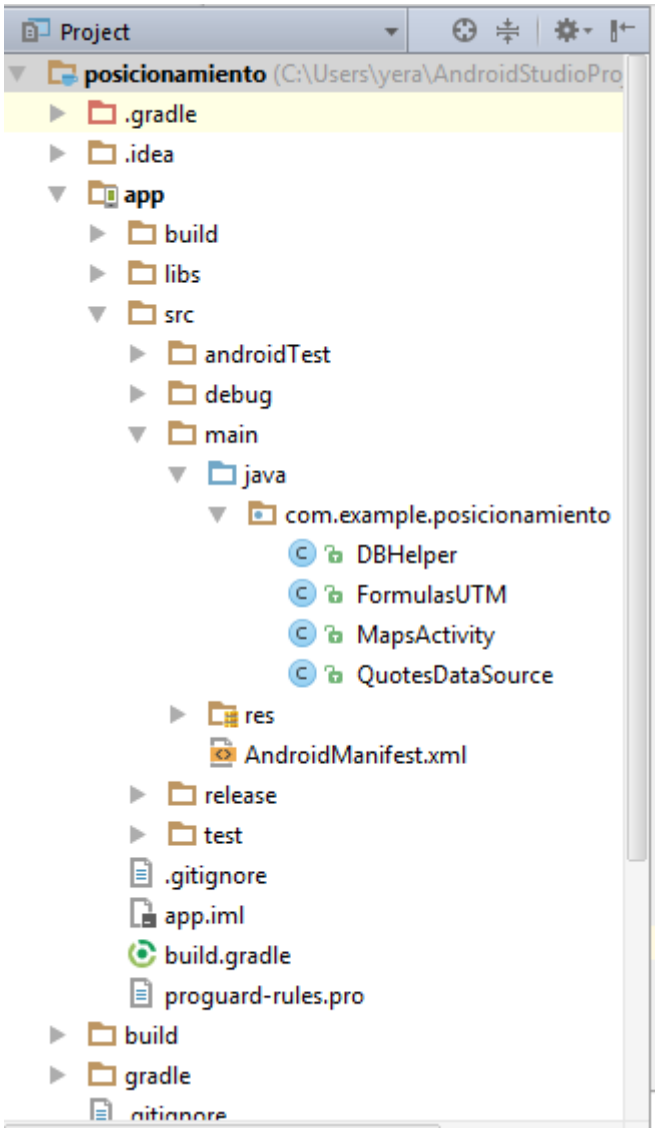

Ilustración 43: Estructura o "árbol" del proyecto.

Elaboración Propia.

Si se observa la imagen podemos apreciar que dentro de la carpeta "app" se encuentra todo, o casi todo, lo referente a nuestra aplicación. Si se entra en detalle, los elementos más utilizados son:

Carpeta Build: El término build se traduce al castellano como "construir". En esta carpeta se alojará el archivo .APK que se creará cuando compilemos nuestra aplicación. Este archivo se utiliza para poder instalar dicha aplicación en un dispositivo Android.

Carpeta Libs: El nombre “libs” viene de libraries o en castellano librerías. Es decir, ésta es la carpeta donde hemos de introducir las librerías externas que se hayan descargado y se quieran añadir a nuestra aplicación.

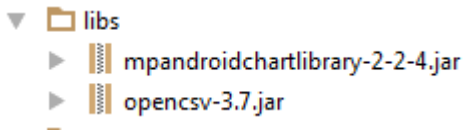

Ilustración 44: Librerías incluidas en el proyecto.

Elaboración Propia.

Carpeta Java: Esta carpeta, junto con otras tantas, se encuentra dentro de la carpeta “src” la cual se traduce como “source” o en castellano “fuente”. Es decir, aquí se incluye todo el código fuente para nuestra aplicación.

En esta carpeta se encuentran todos los archivos de java de nuestra aplicación, separados por módulos que anteriormente hemos creado programando. Estos archivos se pueden abrir con Android Studio u otros programas de programación o incluso lectores de texto.

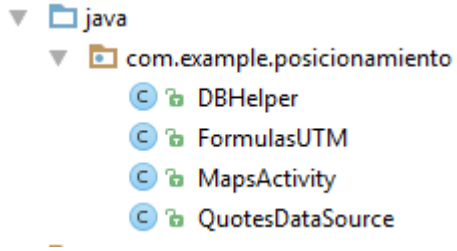

Ilustración 45: Archivos Java incluidos en el proyecto.

Elaboración Propia.

Carpeta Res: En esta carpeta se introducirá todo lo referido a contenido visual de nuestra aplicación. El nombre res viene de “resources” o en castellano “recursos”, estos recursos han de ser introducidos por el usuario para que Android Studio los pueda incorporar a la aplicación.

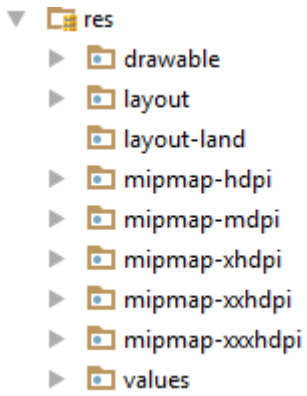

Ilustración 46: Carpeta Res o de recursos.

Elaboración Propia.

Dentro de ésta última se hallan distintas subcarpetas, cada una con una función particular. La subcarpeta “drawable” es la que se encarga de almacenar los ficheros de imagen que posteriormente serán utilizados en nuestra aplicación.

La subcarpeta “layout” contiene los ficheros XML con las distintas vistas de la aplicación.

Las subcarpetas “mipmap” se utilizan para obtener imágenes, las cuales podrían haber sido introducidas en la carpeta “drawable”, a diferentes escalas.

Es decir, si se introduce una imagen en las diferentes carpetas “mipmap” se conseguirá esta misma imagen con diferentes tamaños los cuales suelen ser típicos en Android. Usualmente se utiliza para el icono principal de la aplicación.

La carpeta “values” también contiene ficheros XML los cuales definen propiedades de las vistas, tales como: colores por defecto, márgenes predefinidos y cadenas de caracteres que sean invariables o que puedan aparecer en distintos idiomas.

Android Manifest: El archivo AndroidManifest.xml es un elemento de configuración donde se podrá aplicar las configuraciones básicas de una aplicación. Este fichero describe la aplicación a la que se refiere. Define su nombre, paquete, iconos, estilo, versión mínima de Android, etc. Se indican las actividades, las intenciones, los servicios y los proveedores de contenido de la aplicación. Aquí se declaran los permisos de Android que requerirá la aplicación.

Como detalle importante aquí también se incluirá la clave solicitada a Google para poder incluir Maps en nuestra aplicación.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Ilustración 47: Permisos incluidos en el AndroidManifest.xml

Elaboración Propia.

2.8.3. Android monitor.

El Android Monitor es una herramienta muy potente la cual nos permite interactuar con el aparato con el que estemos testeando nuestra aplicación, ya sea un dispositivo real o uno simulado con el AVD. Una función consiste en poder "navegar" por los archivos del terminal Android, pudiendo crearlos, modificarlos o simplemente comprobar que se ha creado algún archivo correctamente desde nuestra aplicación. Otra función que posee es que permite conocer en tiempo real todos los datos del teléfono, desde el uso de memoria hasta las aplicaciones que están siendo ejecutadas. Además permite obtener una captura o un vídeo de la pantalla actual. Pero la función por la cual resulta muy interesante, es la conocida por "logcat" o registro en Android Studio. Esta función nos informa de qué errores ocurren cuando ejecutamos nuestra aplicación o en cualquier momento durante la ejecución de la misma. Conociendo dichos errores, el programador puede intentar solucionarlos por su cuenta o con la ayuda de la comunidad, facilitando así la solución de los mismos.

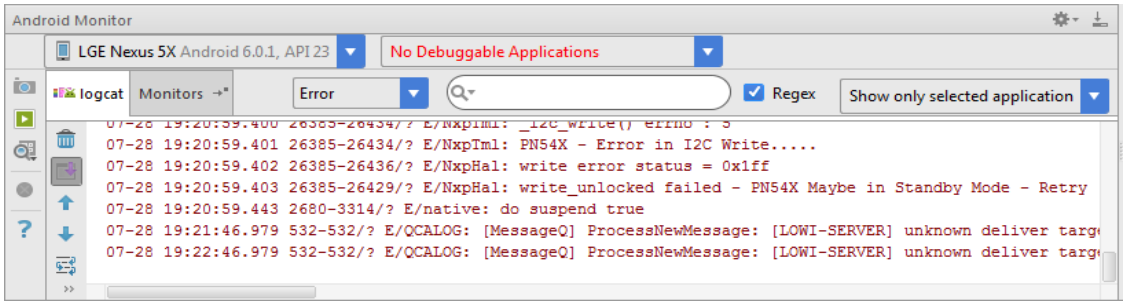

Ilustración 48: Logcat de Android Monitor.

Elaboración Propia.

2.8.4. Gradle Console o consola de eventos.

En esta pequeña consola se mostrará información cuando Android Studio esté trabajando. Por ejemplo, cuando se estén instalando actualizaciones, cuando se carguen datos guardados o simplemente al compilar código. Todos estos procesos serán detallados en este elemento junto con el tiempo de ejecución, lo cual puede resultar muy interesante.

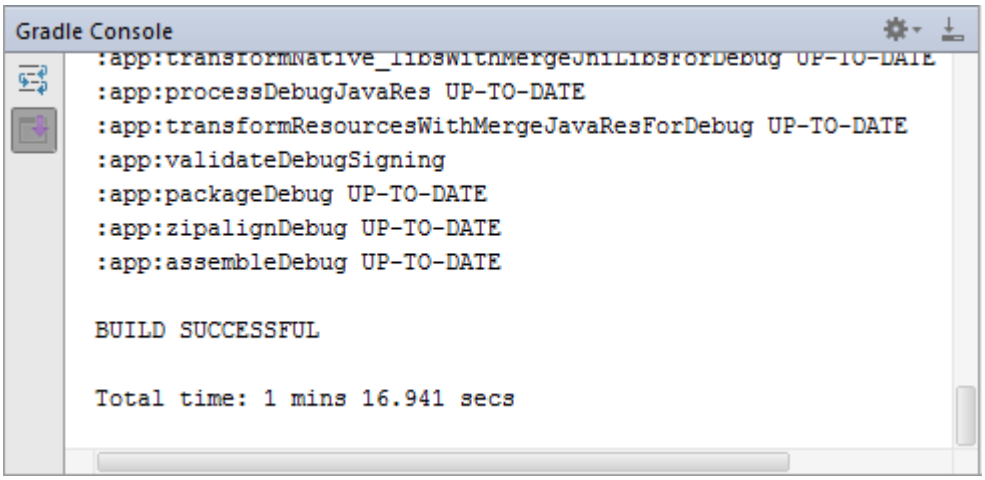

Ilustración 49: Muestra de resultados en Grandle Console.

Elaboración Propia.

Para este trabajo ha resultado de gran utilidad a la hora de crear un archivo .APK para poder instalar la aplicación en otros dispositivos Android y poder probarla en más de uno. Ésta nos indicaba cuándo se ha creado el archivo .APK y su ubicación dentro del PC, además de si había sido satisfactoria dicha creación y su tiempo correspondiente.

2.8.5. Botón Android SDK Manager

Como previamente se ha explicado, este botón se utiliza para abrir el SDK (software development kit) o herramienta de desarrollo de Java. En el añadiremos las herramientas de Google que necesitemos orientadas a nuestra aplicación.

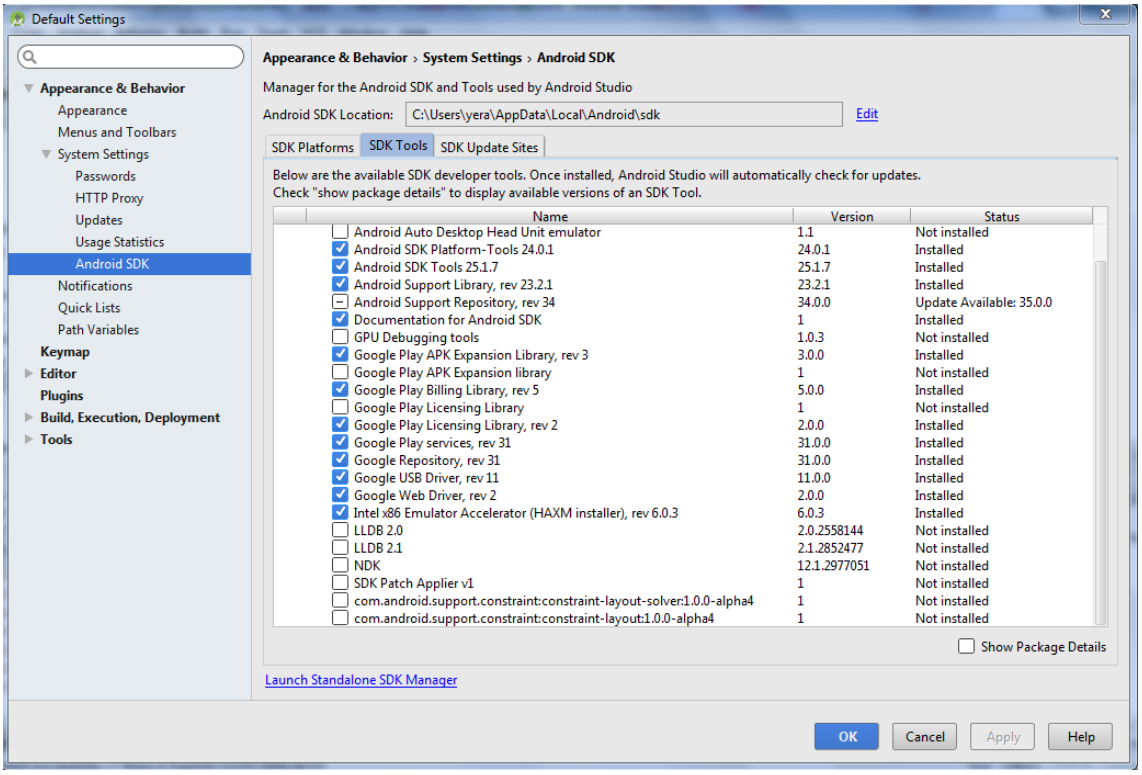

Ilustración 50: Herramientas añadidas mediante SDK Manager.

Elaboración Propia.

Esta herramienta se ha de comprobar periódicamente, ya que pueden ser lanzadas nuevas actualizaciones para los componentes o nuevas versiones de Android y siempre es conveniente trabajar con las últimas versiones para una mejor programación.

2.8.6. Botón AVD

Este botón abre la herramienta que permite gestionar los dispositivos virtuales con los que podremos interactuar para probar nuestra aplicación. Con cada actualización se incluyen los últimos dispositivos Android disponibles no solo limitándose a smartphones, también podremos utilizar dispositivos con Android TV o Android Wear.

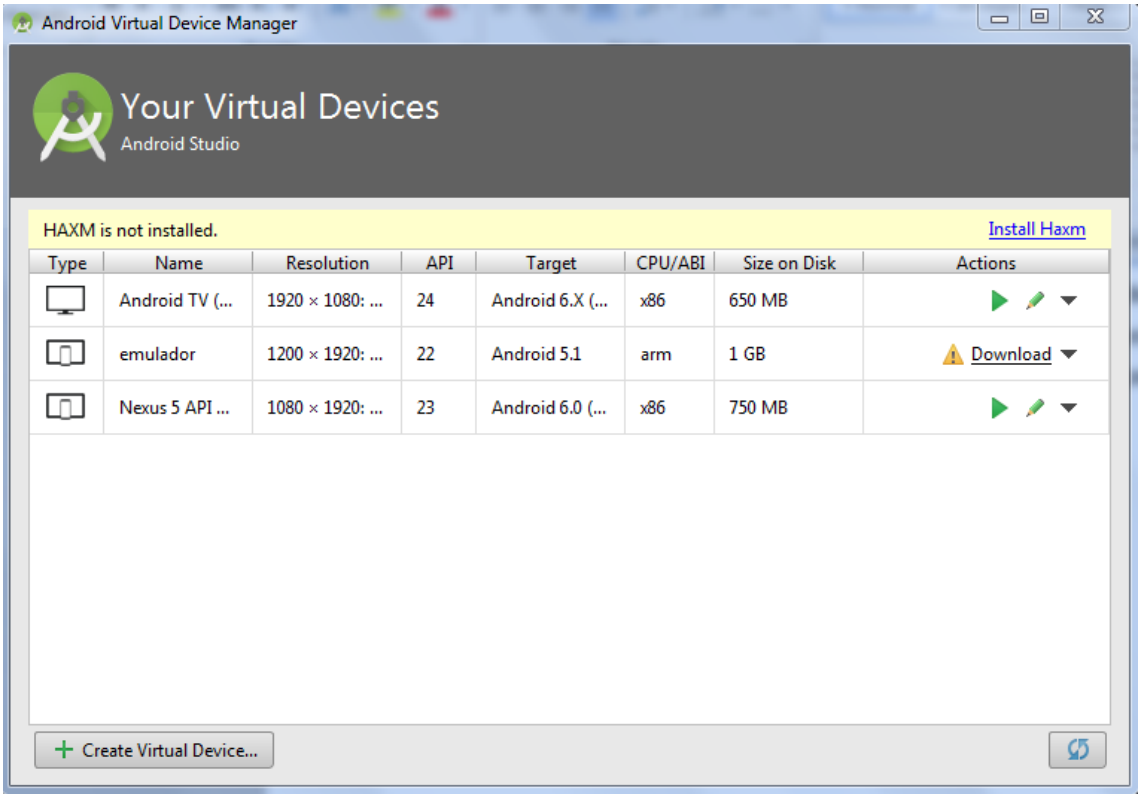

Ilustración 51: Máquinas Virtuales creadas con AVD.

Elaboración Propia.

Al añadir un dispositivo virtual éste podrá simular un dispositivo real seleccionando su nombre comercial o un dispositivo ficticio, en el cual se podrán personalizar todas sus características. Esta características incluyen desde su tamaño de pantalla, su hardware (procesador, ram, etc) hasta su soffware (versión de sistema operativo Android), pudiendo optar a un amplio abanico de testeo para nuestra aplicación.

2.8.7. Botón Run

Es un botón que necesita poca explicación pero que resulta imprescindible a la hora de programar, ya que se encarga de compilar todo el código que se haya escrito.

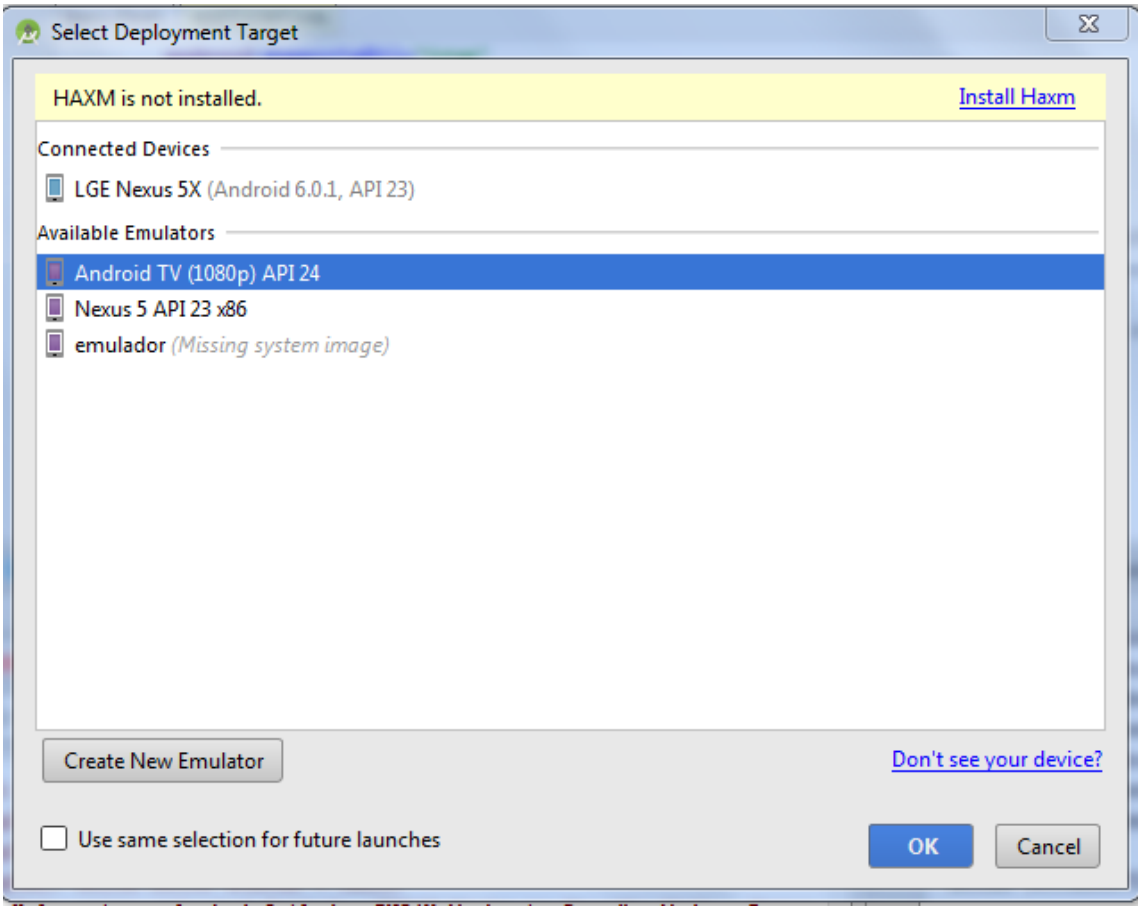

Ilustración 52: Selección de dispositivo para correr la aplicación.

Elaboración Propia.

Al utilizar este botón se pide elegir el dispositivo sobre el cual vamos a "correr" la aplicación.

Una vez elegido, el propio Android Studio se encargará de instalar la aplicación y abrirla en el dispositivo para mayor comodidad del programador.

Todo lo que suceda durante este proceso podrá ser visualizado en el Android Monitor y el tiempo dedicado a ello podrá ser visualizado en la Gradle Console.

3. OBJETIVOS

El objetivo principal de nuestro trabajo de fin de grado consiste en crear una aplicación para Smartphone para su uso en trazados de líneas eléctricas.

Para ello se hace uso del software de desarrollo Android Studio en su versión 2.0 en el cual se llevará a cabo todo el apartado de programación. El apartado de testeo se lleva a cabo en un Smartphone con pantalla de 5,2" y versión Android 6.0.

Al no tener conocimientos en el ámbito de la programación, se propone como objetivo específico el aprendizaje del mismo, que deberá ser correctamente asimilado para una correcta programación de la aplicación.

El aspecto más importante a tener en cuenta será la utilidad junto con la simplicidad. Se intentará que la aplicación sea de utilidad para el usuario y que se convierta en una herramienta usual en su trabajo. A su vez, se intenta que su aprendizaje sea intuitivo y rápido.

Su uso también ha de ser rápido, es decir que en poco tiempo, el usuario deberá ser capaz de abrir la aplicación, almacenar el punto y pasar al siguiente. Solamente tendrá que rellenar unos pocos datos que estarán bien definidos por su recuadro correspondiente.

El aspecto visual también será importante pero no se considerará prioritario para conseguir el objetivo antes descrito. Aun así no se espera una aplicación desagradable a la vista sino más bien poco detallada.

Otro aspecto a tratar será la eliminación completa de errores cuando se está trabajando con la aplicación. Resulta muy molesto que la aplicación se cuelgue o se cierre inesperadamente mientras se trabaja con ella. Se trabajará para la inexistencia de estos errores probando la aplicación en las diferentes versiones disponibles de Android (las más usadas) ya que éstos podrían existir en una versión y no en otra.

4. METODOLOGÍA

Esta aplicación no se crea directamente, sino que requiere de un desarrollo paso a paso donde se pretende cumplir los objetivos fundamentales uno a uno.

Cada paso que se da requiere de una prueba en un dispositivo para asegurar que dicho paso funciona correctamente antes de continuar con el siguiente, pudiendo regresar a un estado anterior para realizar los cambios oportunos conforme se avanza. Seguidamente se enumeran y explican los diferentes pasos que se han dado para el cumplimiento de los objetivos de esta aplicación.

Primer paso: Elegir los criterios básicos para la aplicación.

Una vez instalado el software Android Studio, elegimos la opción de crear un nuevo proyecto. Primero se ha de dar un nombre al proyecto, este nombre no es el definitivo de la aplicación, sino un nombre general que sirva para poder ubicar todo el proyecto en una carpeta.

Después se elige el tipo de dispositivo Android hacia el que va orientada la aplicación: Smartphone y tablet (la elegida en este trabajo), Android TV, Android wear, Auto o Glass.

Tras esto, hemos de elegir la versión de Android objetivo de nuestra aplicación, en este caso la última lanzada (6.0) y la versión mínima de Android para que nuestra aplicación funcione. En este caso la versión mínima escogida es la 4.0, ya que versiones anteriores conllevan problemas de compatibilidad y su utilización es escasa actualmente.

Por último, Android Studio incorpora una serie de plantillas que, en caso de ser elegidas, proporcionan cierto código ya incluido, facilitando así la labor inicial del programador. Este trabajo se beneficia de una de ellas, la llamada *Google Maps Activity,* la cual implementa en nuestro proyecto inicial una serie de instrucciones para conseguir un Mapa como fondo principal de la actividad, lo que nos lleva al siguiente paso.

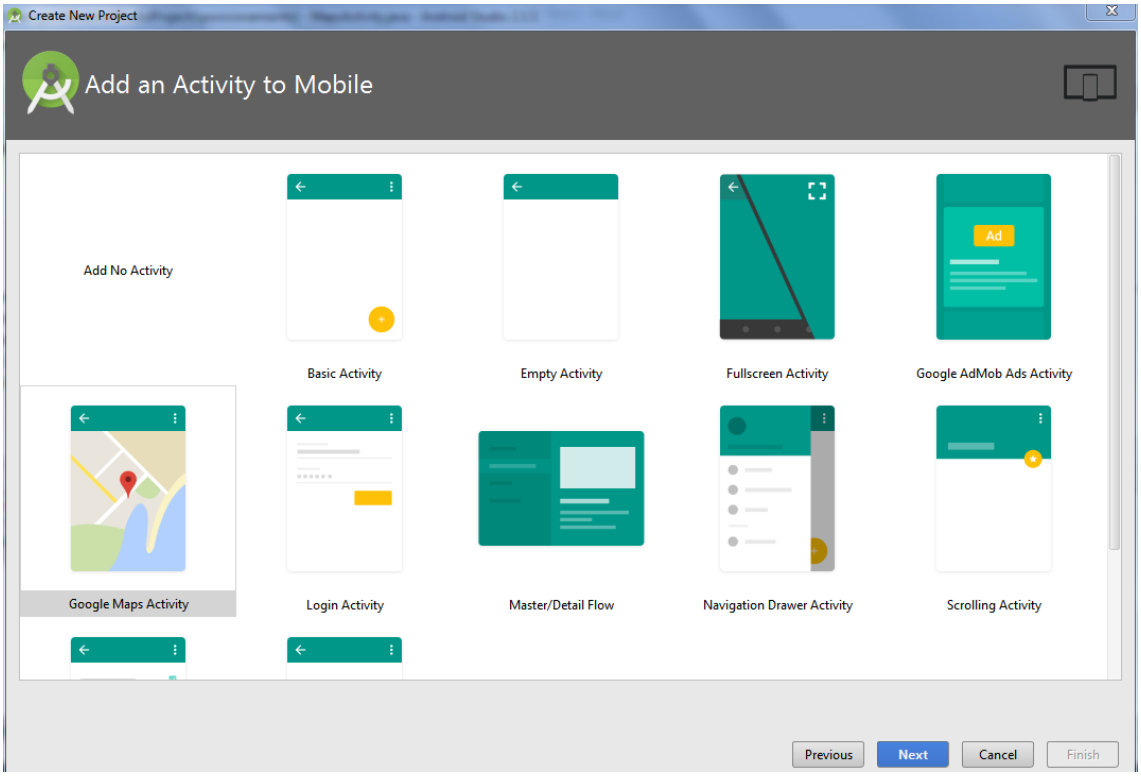

Ilustración 53: Creación de una Google Maps activity.

Elaboración Propia

Segundo paso: Mapa y ubicación.

Esta aplicación está basada en conseguir la ubicación de puntos a determinar por el usuario. La forma de conseguir dichos puntos es visual, es decir, sobre un mapa. Android Studio facilita la tarea de conseguir insertar un mapa en nuestra aplicación, pero no realiza todo el trabajo. La parte que se ha de realizar consiste en obtener una key de Google Maps para nuestro proyecto e introducirla en el archivo AndroidManifest.xml. La obtención de dicha clave está explicada con anterioridad. Una vez introducida se puede compilar el programa y comprobar que la aplicación consiste en un mapa.

Para conseguir activar la ubicación solo habrá que escribir unas líneas de código adicionales aunque al desarrollar la aplicación, también para Android 6.0, se necesita programación adicional. Esto es debido a que los permisos en esta nueva versión de Android no se explican cuando instalamos la aplicación, como ocurría en las versiones anteriores, sino que ahora han de ser solicitados en el momento que se quiera beneficiar de dichos recursos.

Como esta aplicación trabajará con la ubicación del usuario desde el primer momento, el permiso de ubicación para quien trabaje con Android 6.0 o superior será requerido la primera vez que inicie la aplicación.

Ilustración 54: Mapa Inicial.

Recuperado en: http://www.aprendeandroid.com/l6/addMaps2.htm

Tercer paso: Almacenar puntos.

Con el mapa y la ubicación totalmente funcionales, empezamos con la primera funcionalidad de la aplicación: guardar los puntos definidos por el usuario. Tras barajar varias opciones la más sencilla e intuitiva la opción de elegir el punto deseado presionando en el lugar correspondiente del mapa.

Tras esto, se abrirá una ventana emergente superpuesta al mapa la cual nos proporcionará información acerca del punto seleccionado. Como información inicial sólo de disponía de las coordenadas del punto en formato latitud-longitud. Más tarde se añadieron diversas opciones para mayor comodidad del usuario, dichas opciones son:

- **Coordenadas en formato UTM y HUSO**: Mediante la programación de diferentes fórmulas de conversión, se consigue obtener este formato de coordenadas con el cual es mucho más fácil trabajar.

- **Altitud**: Una ardua tarea fue la obtención de un valor de altitud con cierta precisión, Google no facilita esa opción de forma programática y se hubieron de estudiar diferentes alternativas. La escogida consiste en contactar con un servicio web externo (curiosamente facilitado por Google) enviándole las coordenadas del punto y éste devuelve un valor de altitud bastante preciso. El inconveniente que presenta esta opción es que el usuario ha de tener un servicio a internet activo cosa que, a día de hoy, resulta muy común.

- **Nombre**: El usuario puede establecer un nombre para el punto, resultando más fácil encontrarlo posteriormente al estar etiquetado.

- **Tipo**: Para una etiquetación más exacta, se puede escoger el tipo de punto. Este elemento consiste en un menú desplegable que muestra diversas opciones. Como nuestro trabajo está orientado al trazado de líneas de alta tensión, las opciones consisten en punto de traza y punto auxiliar.

- **Descripción**: Por último, si se desea añadir alguna observación concreta para el punto se dispone de esta opción.

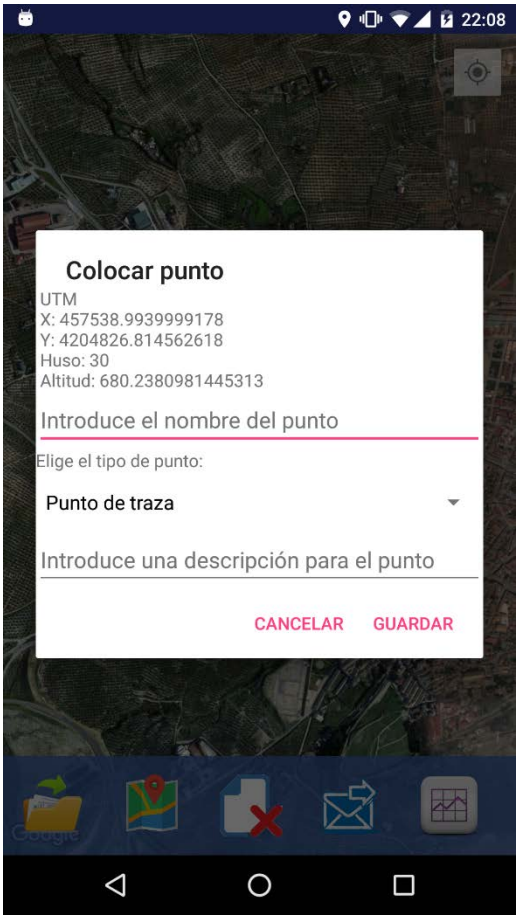

Ilustración 55: Información acerca del punto.

Elaboración propia

Una vez el usuario considere válidos los datos a introducir puede pulsar el botón guardar para almacenar dichos datos o puede desistir de ellos pulsando el botón cancelar.

Para el almacenamiento existen diferentes opciones a considerar. Una cosiste en guardar los datos en variables, pero si se almacenan muchos de estos datos la carga en la memoria puede llegar a ser un problema.

La otra opción y elegida para este trabajo consiste en la creación de una base de datos. Para ello se utiliza una herramienta integrada en el código Android conocida como SQLite Database. Esta base de datos será similar a una tabla organizada en columnas para cada apartado y cada vez que se guarde un punto se rellenará una fila con la información de dicho punto.

Destacar que el usuario no tiene acceso a esta base de datos, es un archivo interno al cual sólo se puede acceder desde la aplicación.

Por tanto, al comprobar la aplicación, se podrá observar cómo queda el cuadro para rellenar los datos del punto, pero no se podrá comprobar si los datos están bien guardados. No obstante, se puede comprobar si la base de datos está bien creada y guarda los datos, al no recibir ningún aviso de error desde el logcat.

Cuarto paso: Lectura de los datos almacenados.

Ahora que ya se dispone de una base de datos y un método para almacenar valores en la misma, se hace necesario poder leer estos datos. Para ello, se valoran dos opciones: leer los datos dentro de la misma aplicación o exportarlos para poder leerlos fuera. Se considera necesario implementar las dos opciones ya que la primera permite comprobar los datos guardados por si hubiese algún error y la segunda permite utilizar estos datos con algún programa externo para aprovechamiento de los mismos.

En este paso se trata de solventar la primera opción. Para ello se implementa la línea de botones, que consiste en diferenciar en la parte inferior de la pantalla de la aplicación un espacio horizontal superpuesto al mapa para la inclusión de los botones necesarios en esta aplicación. Se decide crear de esta manera ya que confiere a la aplicación una interfaz más visual e intuitiva.

Ilustración 56: Icono del botón Archivar.

Recuperado en*: http://www.freeiconspng.com/*

El primer botón será el llamado “botón leer”, el cual permitirá al usuario leer la base de datos con los datos guardados. Este botón tiene implementada una función que lee todos los elementos de la base de datos y los muestra ordenadamente en pantalla a través de una ventana emergente. Aquí se muestra cómo a cada punto se le asigna un número identificador único.

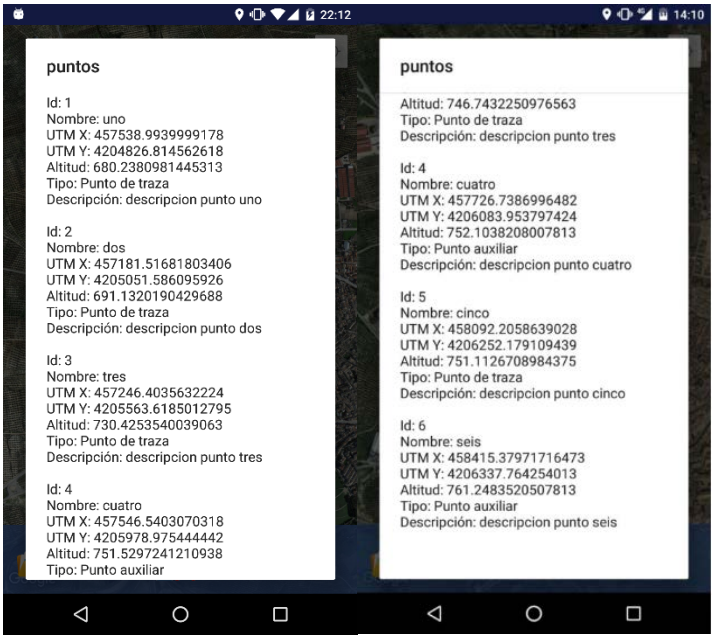

Ilustración 57: Ejemplo de puntos guardados.

Elaboración propia.

Se pueden guardar cuantos puntos se deseen, pero no todos van a caber en la misma pantalla. Para poder ver el punto deseado en caso de no ser mostrado, el usuario simplemente ha de deslizar el dedo verticalmente hasta encontrarlo.

Quinto paso: Tipo de mapa.

Esta aplicación tiene definida por defecto el tipo de mapa satélite el cual podrá visualizar el usuario nada más iniciar la misma. Pero puede pasar que un usuario se sienta más cómodo o prefiera utilizar alguna de las variaciones de mapa que incluye Google Maps.

Para ser más precisos existen cuatro variaciones, las cuales se implementarán en la aplicación. Esto se hará mediante un botón que al ser pulsado nos mostrará una ventana emergente con las cuatro opciones de mapa disponibles.

Ilustración 58: Icono del botón Cambio de Mapa.

Recuperado en: www.iconfinder.com

Al seleccionar una opción, se cambiará el tipo de mapa al de la opción elegida pudiendo elegir entre:

- **Mapa normal:** Muestra la vista como un mapa de carreteras. Éste es el mapa que viene por defecto en la aplicación Android de Google Maps. Aquí se muestran nombres de calles, edificios importantes, etc.
- **Mapa satélite**: Este mapa es el incluido por defecto en la aplicación. Permite ver el mapa con imágenes obtenidas por satélite.
- **Mapa híbrido**: Muestra una combinación de los mapas normal y satélite.
- **Mapa terreno**: Muestra un mapa físico basado en información terrestre.

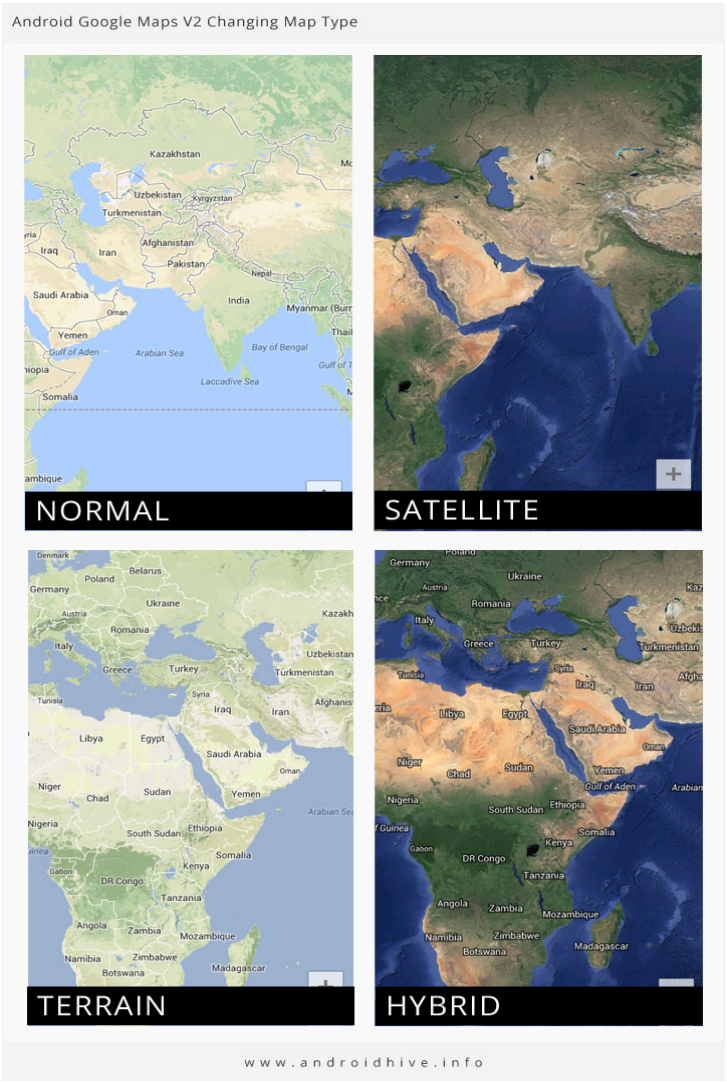

Ilustración 59: Tipos de mapas.

Recuperado en: http://www.androidhive.info/2013/08/android-working-with-google-maps-v2

Cabe destacar que el usuario podrá cambiar el tipo de mapa cuando desee y cuantas veces desee. El menú de selección de mapa es del tipo seleccionable, es decir, cuando se selecciona una opción esta aparecerá marcada la próxima vez que se abra este menú.

Ilustración 60: Selección del tipo de mapa.

Elaboración Propia.

Sexto paso: Eliminar datos.

Tras un tiempo manejando la aplicación, el usuario puede necesitar eliminar ciertos puntos almacenados o crear un nuevo proyecto eliminándolos todos. Para tener esta posibilidad se incluye el botón borrar.

Ilustración 61: Icono del botón Borrar.

Recuperado en: http://www.freeiconspng.com/

Cuando se pulsa el botón borrar, se abre una ventana emergente la cual muestra dos opciones: borrar un punto o borrar todos los puntos. Como se observa en el cuarto paso: lectura de los datos almacenados, la base de datos asigna un identificador único a cada punto, el cual se muestra como “id”.

Para borrar un punto el usuario solamente deberá introducir el número identificador de dicho punto y pulsar el botón correspondiente.

Ilustración 62: Borrar un punto.

Elaboración Propia.

Aunque no se ha comentado antes, la base de datos interna de la aplicación que es la encargada de guardar toda la información, no es formateada al cerrar la aplicación ni siquiera al reinstalarla. El usuario puede cerrarla o apagar el móvil y cuando vuelva a abrirla los datos seguirán ahí.

Pero no siempre se va a trabajar con el mismo proyecto, por eso se incluye la función: Borrar todos los puntos. Esta función tiene la tarea de formatear la base de datos para poder empezar con un nuevo proyecto.

Al testear la aplicación se comprueba que funciona correctamente y se añade un mensaje temporal que nos indica cuando hemos pulsado el botón.

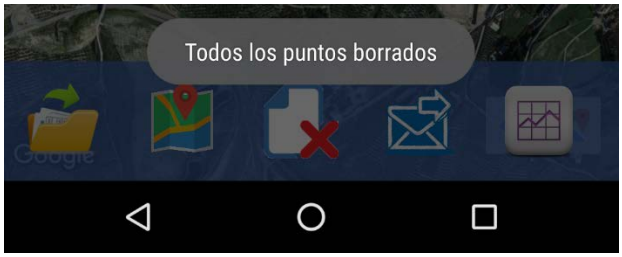

Ilustración 63: Opción Borrar todos los puntos.

Elaboración Propia.

Séptimo paso: Exportar datos

Para finalizar con la composición de puntos realizada por el usuario, éste puede exportarla mediante el envío de un archivo incluido en un correo electrónico. Para ello se programa el botón enviar.

Ilustración 64: Icono de botón Exportar datos.

Recuperado en*: www.iconfinder.com*

Como ya se ha comentado con anterioridad, la base de datos con todos los puntos guardados no puede ser exportada directamente ya que es un elemento interno de la aplicación. Al pulsar este botón, la aplicación lee toda la información guardada en la base de datos y genera con ella un archivo de extensión .CSV. Este tipo de archivo se puede leer fácilmente con programas como Microsoft Excel® o similares.

Tras esto se invita al usuario a elegir una aplicación para poder enviar el correo. Preferiblemente, al estar trabajando en la plataforma Android, ésta será la aplicación para Android de Gmail.

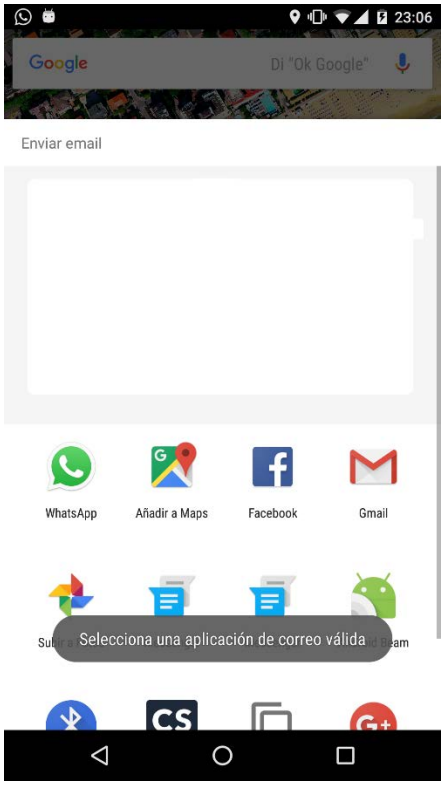

Ilustración 65: Selección de aplicación para el envío.

Elaboración Propia.

Una vez seleccionada se abrirá dicha aplicación incluyendo un breve mensaje informando acerca de los puntos guardados y de la inclusión de un archivo CSV.

Este archivo aparecerá automáticamente como adjunto, por tanto el usuario solo necesitará añadir una dirección de correo válida para poder exportarlo.

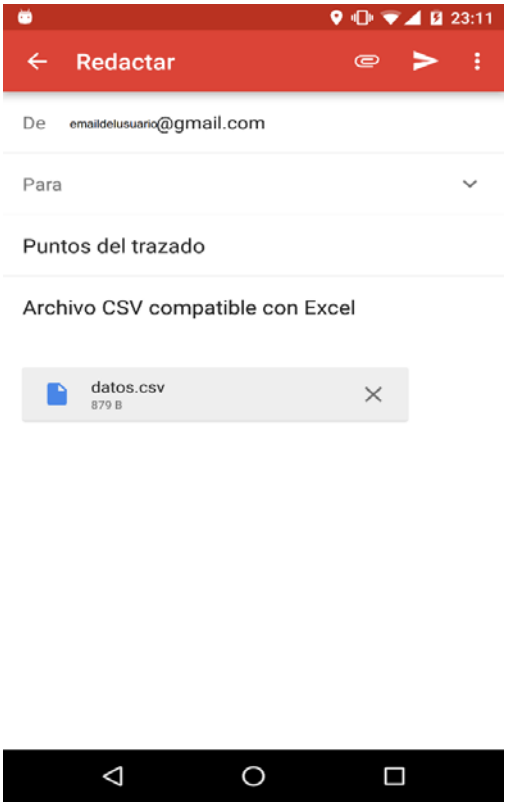

Ilustración 66: Archivo adjunto.

Elaboración Propia.

El correo tendrá como originario la dirección del usuario y el destinatario podrá ser el mismo. Una vez enviado el usuario podrá descargar la información en el mismo dispositivo o en cualquier otro, como un PC para trabajar sobre los datos obtenidos.

Octavo paso: Trazar un perfil longitudinal

Como opción adicional el usuario puede tomar la información de altitud de todos los puntos guardados para dibujar una gráfica con los datos de dichos puntos. Para ello se implementa el botón “gráfica”.

Ilustración 67: Icono de botón Gráfica.

Recuperado en: http://www.icons.mysistemyway.com

Al utilizar este botón, se abrirá una ventana emergente en la cual simplemente se mostrará el perfil de altitud de los puntos guardados para que el usuario posea una aproximación visual.

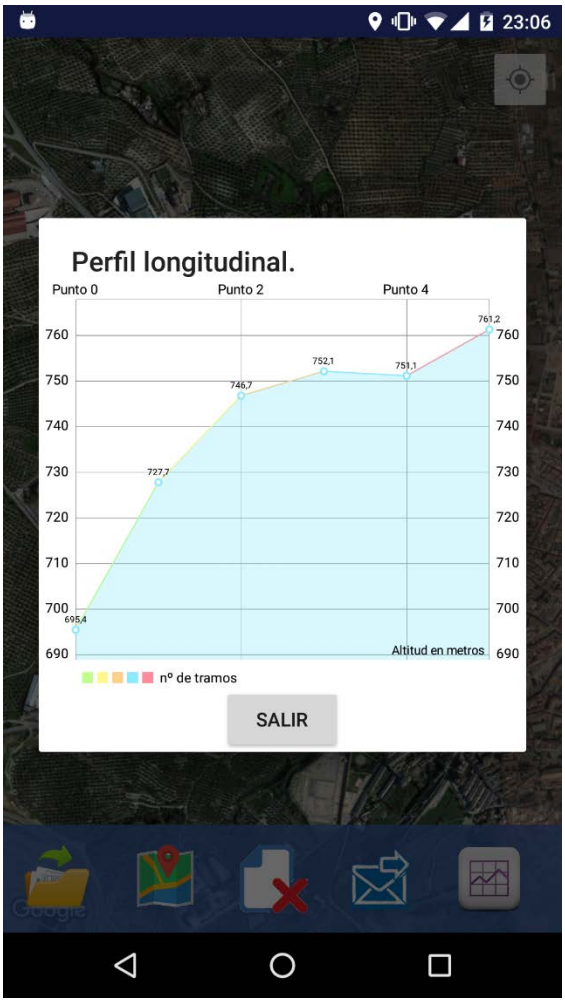

Ilustración 68: Perfil de altitud de los puntos guardados.

Elaboración Propia.

En ella aparecen todos los tramos compuestos por dos puntos y la altitud de dichos puntos representada sobre cada uno en metros.

Para concluirla se añade un botón que nos devuelve a la pantalla principal.

5. RESULTADOS

Entre los resultados de nuestro trabajo podemos encontrar que ofrece una gran comodidad y utilidad para el proyectista poder conocer con exactitud los puntos por los que debe trascurrir un trazado a la hora de colocar una línea de alta tensión. Utilizando simplemente su Smartphone, el cual casi siempre se suele llevar a mano, permite obtener una información rápida de los puntos y si le parece correcto almacenarlos.

Esta aplicación muestra una información tan valiosa como es la ubicación y la altitud para poder conocer el desnivel entre los diversos puntos, además de poder añadir información propia para consultarla en el momento que sea necesario.

Una vez completada la lista con los puntos deseados esta podrá ser exportada a un ordenador de trabajo donde se dispondrá de los puntos guardados para su posterior elaboración con un software de cálculo, diseño de planos, etc.

La ventaja más significativa que tiene esta aplicación respecto a otras ya existentes en el mercado radica en su diseño, puesto que trabaja siempre sobre un mapa para una correcta ubicación y una utilización más sencilla.

Por el contrario, en otras aplicaciones el usuario se encarga de crear una lista de puntos introduciendo las coordenadas de su ubicación, lo que limita su uso y no proporciona una correcta visualización del trazado que se crea.

Cabe destacar, además, todo lo aprendido acerca de la programación, poseyendo escasos conocimientos básicos en este aspecto. A pesar de este inconveniente, se ha podido llegar a crear una aplicación que, aunque de carácter básico, es plenamente funcional y de utilidad para aquellos usuarios especializados.

Todo esto ha sido posible gracias a la documentación compartida por otros usuarios en Google (que es de gran ayuda para iniciarse) y también gracias a la comunidad de programadores, muy útil para resolver errores una vez empiezan a surgir éstos durante la programación.

Respecto al testeo, esta aplicación se ha probado en diversos smartphones con diversas versiones de Android. En concreto se ha probado en la versión 4.4, la 5.0 y la 6.1 obteniendo para todas ellas resultados satisfactorios. También en una tablet con la versión 5.0 instalada funcionando correctamente.

Además, se ha probado a incluir un gran número de puntos en la aplicación, llegando a tener un total de 250 puntos añadidos. La aplicación responde rápidamente y sin problemas, pudiendo leerlos, seguir añadiendo o exportarlos sin problemas aunque el tamaño del archivo CSV aumenta proporcionalmente dependiendo de la cantidad de puntos.

6. DISCUSIÓN

El objetivo de este estudio ha sido crear una aplicación para Smartphone Android que fuera de utilidad para la realización de una línea de alta tensión o similar. Dicha utilidad se basa en la ayuda que esta aplicación puede brindar al proyectista a la hora de elegir los diversos puntos por los que deberá discurrir el trazado.

Se puede decir que en este trabajo de fin de grado se crea una primera versión de la aplicación, totalmente funcional, en la que se cumplen una serie de objetivos básicos.

Aun siendo funcional, todavía permite numerosas mejoras. A continuación se exponen una serie de mejoras que podrían ser implementadas en un futuro:

- La primera y más importante será la de incluir esta aplicación en el mercado Google Play™. Aquí podrá ser descargada por cualquier persona que lo desee y así el usuario podrá probar su utilidad de primera mano y ofrecer su opinión, tanto de las cosas positivas como de las negativas, para ayudar a mejorarla.

- La segunda consistiría en mejora en la interfaz gráfica.
  Con la ayuda de una persona especializada podría hacerse que la aplicación tuviera un acabado visual más profesional ya que actualmente solo se han utilizado elementos básicos.

- Por otra parte podría mejorarse la Versión de funcionamiento. Android es un sistema operativo que dispone de continuas actualizaciones. Por lo que debería programarse una tarea de mantenimiento para estudiar y adaptar la aplicación a la nueva versión, evitando de este modo comportamientos erróneos dentro de una futura versión.

- En cuanto al funcionamiento de la aplicación, podrían incluirse ciertas mejoras que, si bien no modifican su comportamiento sí lo harían más cómodo. Por ejemplo:

- Añadir líneas que permitan unir los diferentes puntos señalados en el mapa para una mejor visualización de la línea.

- Poder editar un punto desde la aplicación una vez guardado. Por el momento sólo puede borrase y volverse a crear.

- Poder crear varios proyectos a la vez o, simplemente no tener que eliminar el proyecto actual para empezar uno nuevo.

- Un mejor sistema de descarga del archivo CSV, pudiendo ser descargado directamente en el Smartphone sin necesidad de intermediar con una aplicación de correo.

- Por último se podría desarrollar esta aplicación para diferentes plataformas móviles en cuyo caso la primera sería IOS, al ser la más utilizada por detrás de Android.

7. BIBLIOGRAFÍA

- Cosmos, J. (2016). El 7,5% de los dispositivos Android ya están actualizados a Marshmallow**.** Recuperado el 20 de Mayo de 2016 en: http://www.xatakandroid.com/mercado/el-7-5-de-los-dispositivos-android-ya-estan-actualizados-a-marshmallow

- Developer.android.com. (2016). *Android Developers*. [online] Recuperado el 12 de Febrero de 2016 en https://developer.android.com/index.html

- Esquiva, A. (2013). JSON 1- ¿Qué es y para qué sirve JSON? Recuperado el 12 de Febrero de 2016 en: https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/

- García, F., Montiel, J.F. & Pro, J.L. (2016). Introducción a la programación en Android. Recuperado el 12 de Febrero de 2016 en: https://mileto.cica.es/cursos/node/114?codigo=03-ANDROID

- Gironés, J.T. (2015). Android: Introducción a la programación. Universidad Politécnica de Valencia. Recuperado el 15 de Diciembre de 2015 en: www.androidcurso.com.

- Gironés, J.T. (2015). El Gran Libro de Android 5ª Edición. España: Maracombo. S.A.

- Gómez, S. (2011). Bases de Datos en Android (I): Primeros pasos. Recuperado el 12 de Febrero de 2016 de: http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/

- Gómez, S. (2014). Estructura de un proyecto Android (Android Studio). Recuperado el 12 de Febrero de 2016 de: http://www.sgoliver.net/blog/estructura-de-un-proyecto-android-android-studio/

- López, D. (2014). IDE: Entornos Integrados de Desarrollo para Android. Recuperado el 6 de Abril de 2016 en http://academiaandroid.com/ide-entornos-integrados-de-desarrollo-para-android/

- Mocholí, A. (2015). 10 Entornos de programación para desarrollar apps Android sin Java. Recuperado el 6 de Abril de 2016 en https://www.yeeply.com/blog/entornos-programacion-desarrollar-apps-android/

- Phillips, B., Stewart, C., Hardy, B. & Marsicano, K. (2015). Programación Con Android - Edición 2016 (Títulos Especiales). España: ANAYA.

- Regueira, M. (2015). Comparativa: iOS vs Andoid, fortalezas y debilidades. Recuperado el 12 de Febrero de 2016 en: http://www.elgrupoinformatico.com/comparativa-ios-android-fortalezas-debilidades-t26834.html

- Revelo, J. (2014). Tutorial básico del lenguaje Xml. Recuperado el 12 de Febrero de 2016 en: http://www.hermosaprogramacion.com/2014/08/xml-lenguaje/

- Revelo, J. (2015). Tutorial De Layouts En Android. Recuperado el 6 de Abril de 2016 de: http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/

- Revelo, J. (2015). Tutorial De Android Studio: Navegación En La Interfaz. Recuperado el 6 de Abril de 2016 de: http://www.hermosaprogramacion.com/2015/09/tutorial-de-android-studio-navegacion-en-la-interfaz/

- Sanz, J. (2016). Android sigue liderando el mercado en España y Europa en 2015 mientras iOS se estanca. Recuperado el 16 de febrero de 2016

en: http://www.movilzona.es/2016/01/27/android-recupera-cuota-de-mercado-en-espana-y-europa-en-2015-mientras-ios-se-estanca/

- Stackoverflow.com. (2016). *'android' tag wiki*. Recuperado el 20 de Febrero de 2016 en: http://stackoverflow.com/tags/android/info

- Subirats, J. (2014). Qué es y para qué sirve el SDK. Recuperado el 20 de Mayo de 2016 de http://www.fandroides.com/que-es-y-para-que-sirve-el-sdk/

- Universidad a distancia de Madrid. Área de Programación y Desarrollo. Manual de XML. Recuperado el 18 de Abril de 2016 en http://www.mundolinux.info/que-es-xml.htm

8. ANEXOS

8.1 MANUAL DE USUARIO

Este manual de usuario explica cómo utilizar la aplicación “TrazadoMapa” para la plataforma Android. Dicha aplicación está pensada para obtener un trazado para líneas de alta tensión o similares.

Con ella el usuario podrá crear líneas de puntos colocados convenientemente conociendo sus características de ubicación y pudiendo añadir información propia.

Tras esto, el usuario puede comprobar la lista de puntos guardados pudiendo incluso borrar los que no desee. Una vez este conforme, esta lista se podrá enviar vía correo electrónico en un archivo de formato. CSV. Como añadido se podrá mostrar un perfil longitudinal con las alturas de los diferentes puntos guardados.

En este manual se indica paso a paso el funcionamiento de la aplicación para un entendimiento completo, pudiendo así trabajar correctamente con ella.

Primer paso: Instalación.

La instalación se corresponde con cualquier puesta en marcha de una aplicación en el sistema operativo Android. En este apartado es importante hacer hincapié en la sección de permisos que necesitará la app para poder funcionar. Si el usuario trabaja con una versión Android inferior a la 6.0, se le informará de estos permisos durante la instalación. En cambio, si la versión es 6.0 o superior, estos permisos se solicitarán dentro de la aplicación (una vez instalada) según sean necesarios.

Permiso de ubicación: Esta aplicación funciona principalmente con un mapa de fondo sobre el cual trabajaremos, obtendremos información y nos moveremos a través de él. En dicho mapa estará siempre presente la ubicación del usuario para trabajar con sitios cercanos con más facilidad.

Permiso de almacenamiento: Cuando el usuario desee exportar la lista de puntos que ha confeccionado, se generará un archivo CSV que posteriormente será añadido como archivo adicional a un correo generado por la propia

aplicación. Este archivo necesita ser previamente almacenado en la memoria del teléfono para que pueda adjuntarse al citado correo.

Segundo paso: Visualización del entorno y botones.

Una vez instalada la aplicación, el usuario se encontrará con la pantalla principal que mostrará un mapa y cinco botones.

No se ha añadido texto alguno pensando en componer una aplicación más visual e intuitiva.

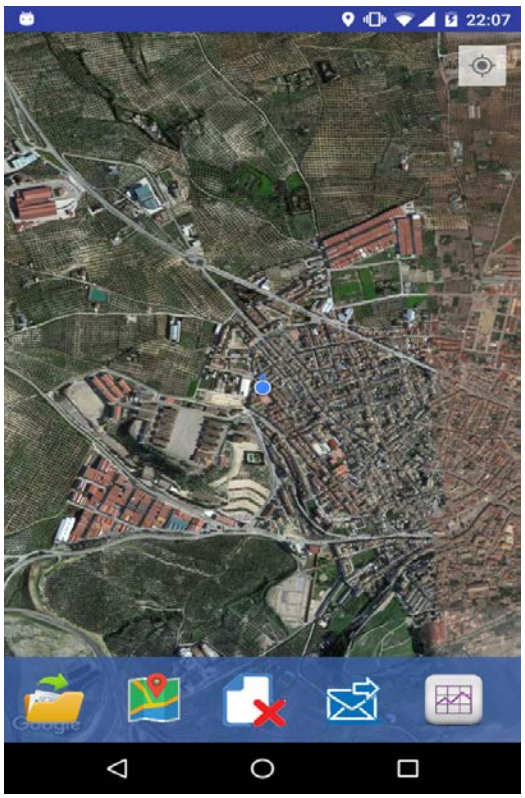

Ilustración 69: Pantalla principal de la aplicación.

Elaboración Propia.

En todo momento, excepto al enviar la información por correo, se trabaja sobre esta base y las distintas funciones son ventanas emergentes que aparecen sobre la misma. Por orden, de izquierda a derecha, los botones se corresponden con:

- Botón leer.
- Botón cambiar tipo de mapa.
- Botón borrar.

- Botón enviar.
- Botón perfil o gráfica.

Tercer paso: Añadir o guardar un punto.

Esta aplicación está basada en archivar puntos elegidos por el usuario. Para elegir un punto simplemente se ha de pichar sobre el mapa en la ubicación deseada y se mostrará la siguiente ventana emergente:

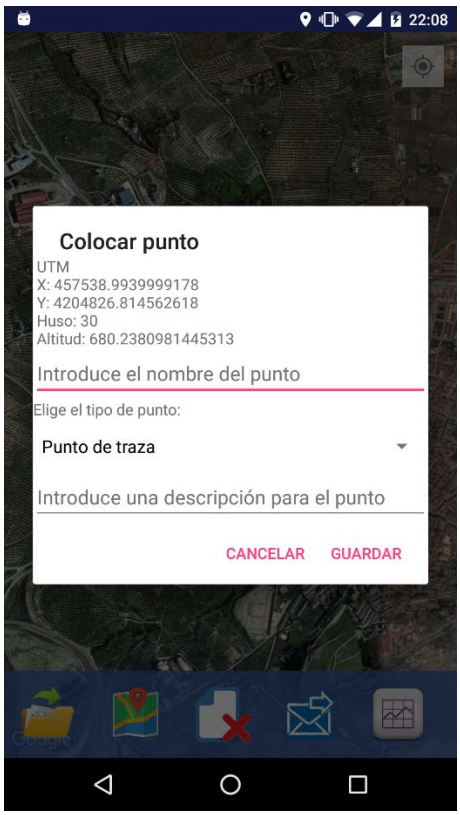

Ilustración 70: Función para guardar un punto.

Elaboración Propia.

Aquí se muestran las características del punto donde se ha pichado: coordenadas UTM, huso y la altitud. Además permite al usuario añadir información acerca del punto como puede ser: nombre, tipo de punto e información adicional.

Si se está conforme se pulsa sobre guardar y el punto queda almacenado, si no se está se pulsa sobre cancelar y no se almacenará dicha información.

Cuando se almacena un punto, se muestra un marcador en la pantalla indicando dónde se encuentra en el mapa.

Cuarto paso: Leer la información guardada.

Una vez hayamos almacenado al menos un punto, podremos leer dicha información pulsando sobre el botón leer. Este botón nos muestra la información en una lista punto por punto.

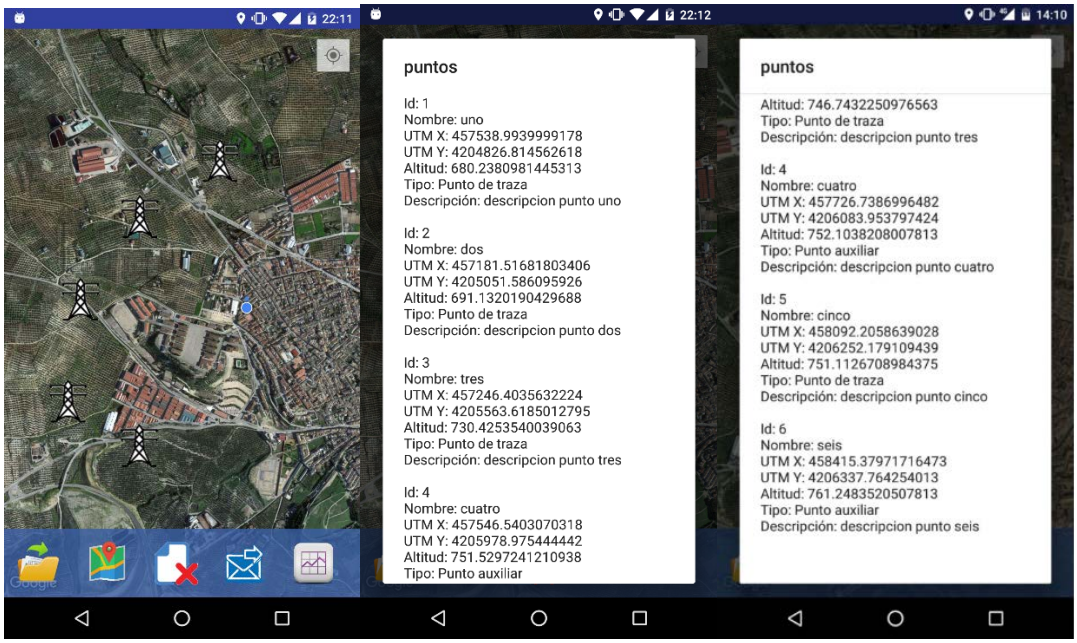

Ilustración 71: Almacenamiento de diversos puntos.

Elaboración Propia.

En el ejemplo anterior se han añadido seis puntos a la lista. Cómo no se pueden visualizar todos dentro de la pantalla, el usuario podrá deslizar la lista hacia arriba y hacia abajo para moverse dentro de ella.

Quinto paso: Cambiar el tipo de mapa.

Puede que al usuario no le guste trabajar con el mapa que se incluye por defecto (Satélite) y prefiera trabajar con una de las opciones que incluye Google Maps. Para ello se ha de pulsar el botón cambiar tipo de mapa y aparecerá el siguiente menú desplegable:

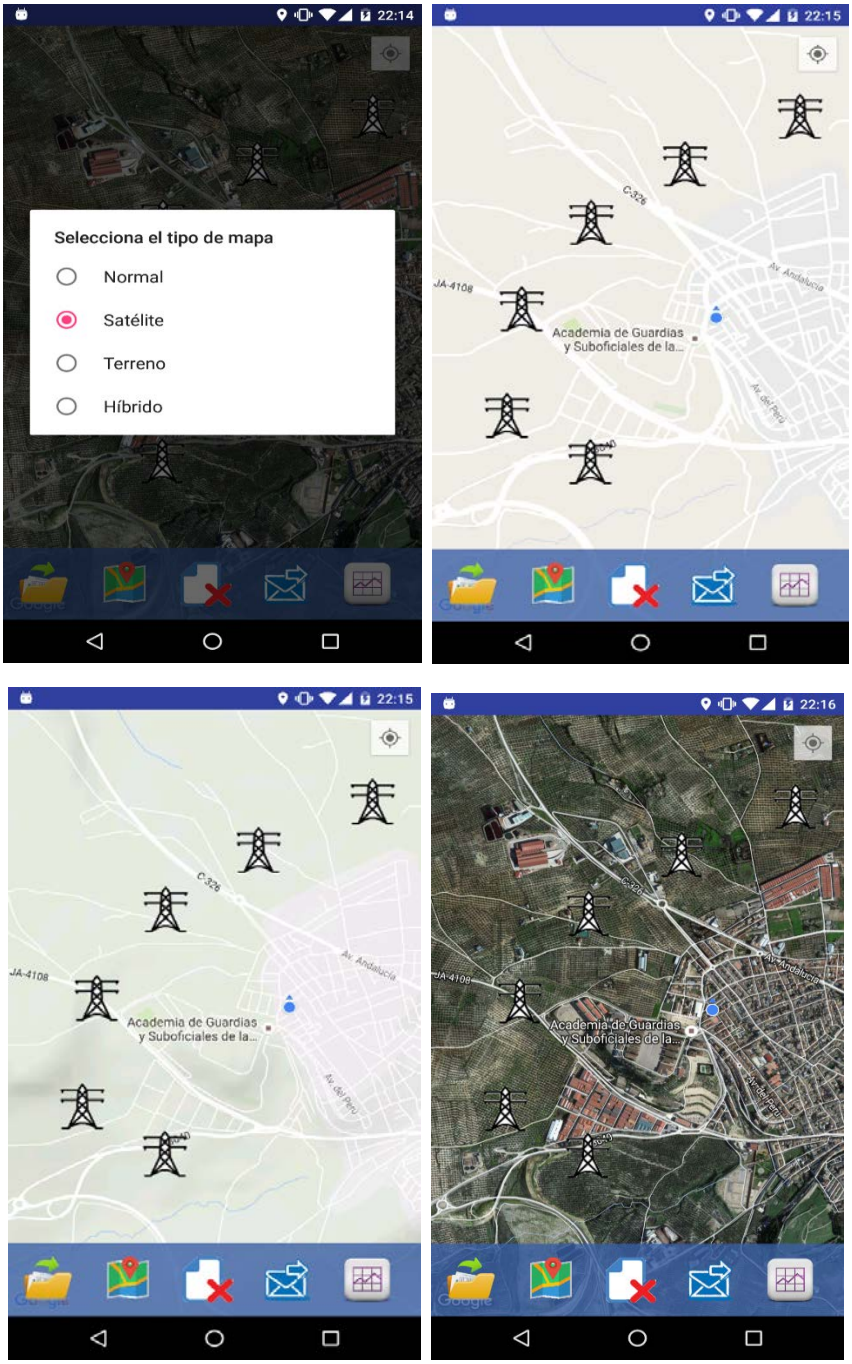

Ilustración 72: Diferentes tipos de mapas.

Elaboración Propia.

Imágenes correspondientes al tipo normal, terreno e híbrido respectivamente. Para el tipo satélite ver el apartado anterior.

Sexto paso: Borrar un punto.

Si se ha añadido información errónea o simplemente se desean modificar los datos existe una función para esto. Pulsando el botón borrar se puede realizar esta tarea:

Ilustración 73: Función de borrado de puntos.

Elaboración Propia.

Si se observa el quinto paso, cada punto tiene una id única que aquí se necesita para elegir qué punto borrar. Simplemente se introduce el valor de la id que queramos eliminar y se pincha en el botón borra este punto.

Se ha añadido la opción de borrar todos los puntos ya que la aplicación por sí sola no elimina los puntos guardados aunque la cerremos. Por tanto, cuando se quiera empezar un nuevo proyecto se ha de recurrir a este botón.

Séptimo paso: Enviar la lista de puntos por correo.

Cuando se termine de añadir los puntos deseados, se podrá obtener la lista con todos los puntos guardados pulsando el botón enviar. Una vez pulsado, se generará un archivo .CSV con los datos y se pedirá seleccionar una aplicación de correo válida.

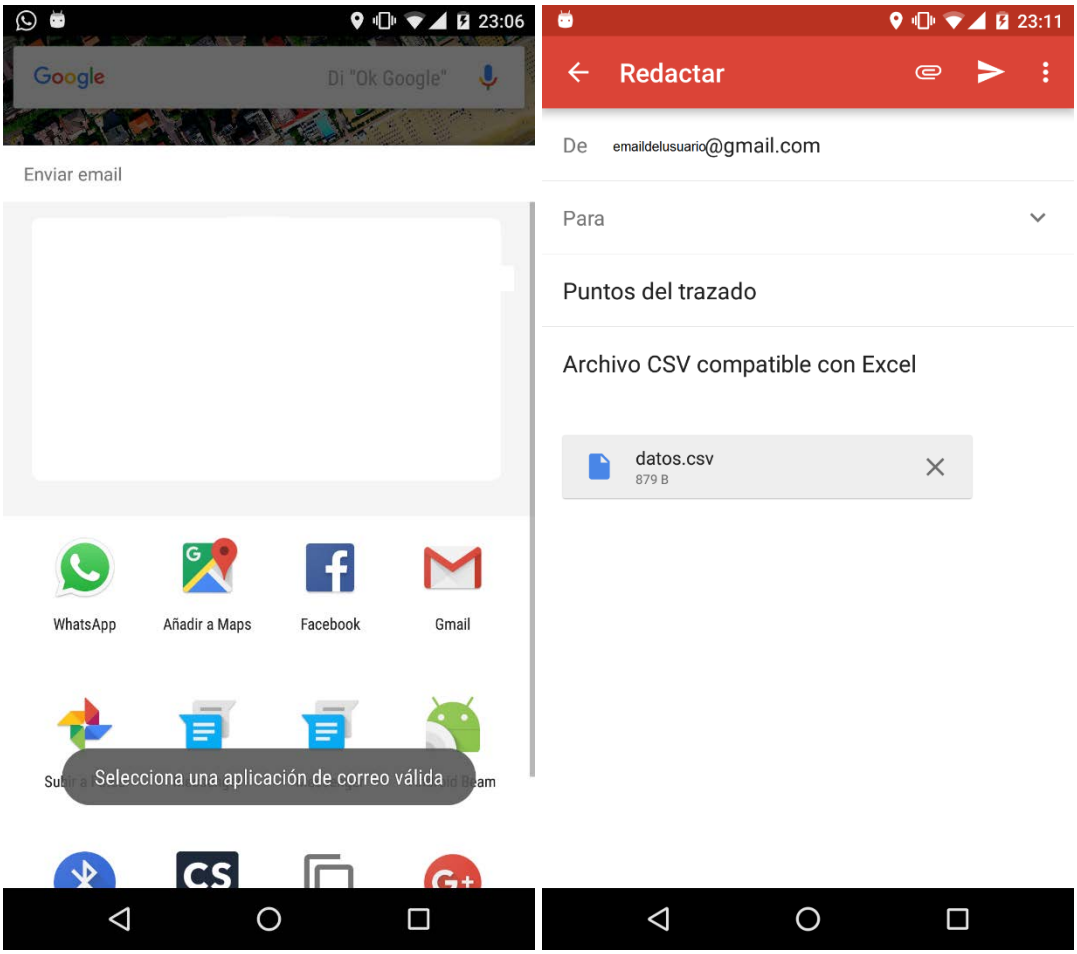

Ilustración 74: Selección de aplicación de correo y correo generado.

Elaboración Propia.

Una vez seleccionada, se mostrará un email con el archivo adjunto para simplemente añadir una dirección de destinatario y poder descargar el documento.

Octavo paso: Obtención del perfil.

Como añadido, la aplicación puede dibujar el perfil o gráfico de altitud de los puntos guardados. Esta se mostrará al pulsar el botón “gráfica”.

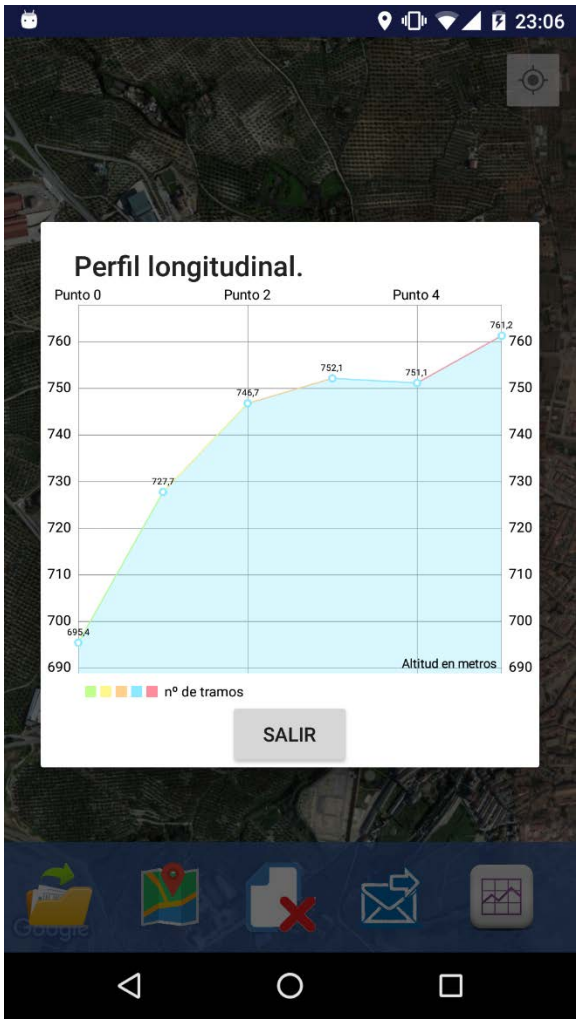

Ilustración 75: Perfil de los puntos almacenados.

Elaboración Propia.

8.2 CÓDIGO FUENTE.

En este anexo se muestra el código usado para la creación de la aplicación objetivo de este trabajo separando cada clase y sus módulos para una mayor claridad.

La clase principal: clase MapsActivity.

Esta clase es la principal de la aplicación y donde se incluye la mayoría del código. En ella primero se declara la clase y los atributos, después las variables globales. Tras esto vendrán los diferentes módulos que serán explicados por separado.

```
public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        GoogleMap.OnMarkerDragListener,
        GoogleMap.OnMapClickListener,
        GoogleMap.OnMapLongClickListener,
        View.OnClickListener,
        View.OnLongClickListener {

    //Variables globales
    private GoogleMap mMap;
    private double latitude;
    private double longitude;
    String altitud = "";
    private static final int REQUEST_CODE_LOCATION = 2;
    private static final int REQUEST_WRITE_STORAGE = 112;
    public static final int REQUEST_ID_MULTIPLE_PERMISSIONS = 1;
    private static final String TAG =
MapsActivity.class.getSimpleName();

    //Botones
    private ImageButton buttonView;
    private ImageButton buttonChange;
    private ImageButton buttonSend;
    private ImageButton buttonUndo;
    private ImageButton buttonGraph;
    int numeroapoyo = 1;
    ArrayList<LatLng> pointlist = new ArrayList<>();
    ArrayList<Entry> listaAltitud = new ArrayList<>();
    ArrayList<String> ejex = new ArrayList<>();
    int contador = 0;

    DBHelper myDb;

    /**
     * ATTENTION: This was auto-generated to implement the App
Indexing API.
     * See https://g.co/AppIndexing/AndroidStudio for more
information.
     */
    private GoogleApiClient client;
    private QuotesDataSource dataSource;
```

```
    private FormulasUTM formulasUTM;
    String edittext;
    String edittext1;
    Uri uri1;
```

Módulo onCreate.

El primer módulo definido, en él se incluirán las funciones que se inicien cuando iniciamos la aplicación. Se utiliza sobre todo para inicializar variables y elementos visuales.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is
ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
            .findFragmentById(R.id.map);
    if (savedInstanceState == null) {
        mapFragment.setRetainInstance(true);
    }
    mapFragment.getMapAsync(this);
    // ATTENTION: This was auto-generated to implement the App
Indexing API.
    // See https://g.co/AppIndexing/AndroidStudio for more
information.
    client = new GoogleApiClient.Builder(this).addApi(AppIndex.API)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API).build();


    //Iniciamos botones y agregamos onclicklistener

    buttonView = (ImageButton) findViewById(R.id.buttonView);
    buttonChange = (ImageButton) findViewById(R.id.buttonChange);
    buttonSend = (ImageButton) findViewById(R.id.buttonSend);
    buttonUndo = (ImageButton) findViewById(R.id.buttonUndo);
    buttonGraph = (ImageButton) findViewById(R.id.buttonGraph);
    buttonView.setOnClickListener(this);
    buttonChange.setOnClickListener(this);
    buttonSend.setOnClickListener(this);
    buttonGraph.setOnClickListener(this);
    buttonUndo.setOnClickListener(this);
    dataSource = new QuotesDataSource(this);
    myDb = new DBHelper(this);
```

Módulos onStart y onStop.

Son módulos auto-generados por Android Studio que se utilizan cuando la aplicación tiene que detener o reanudar su funcionamiento, en este caso solo presentan información autocompletada.

```
  @Override
public void onStart() {
```

```
    super.onStart();

    // ATTENTION: This was auto-generated to implement the App
Indexing API.
    // See https://g.co/AppIndexing/AndroidStudio for more
information.
    client.connect();
    Action viewAction = Action.newAction(
            Action.TYPE_VIEW, // TODO: choose an action type.
            "Maps Page", // TODO: Define a title for the content
shown.
            // TODO: If you have web page content that matches this
app activity's content,
            // make sure this auto-generated web page URL is correct.
            // Otherwise, set the URL to null.
            Uri.parse("http://host/path"),
            // TODO: Make sure this auto-generated app deep link URI
is correct.
            Uri.parse("android-
app://com.example.posicionamiento/http/host/path")
    );
    AppIndex.AppIndexApi.start(client, viewAction);
}

@Override
public void onStop() {
    super.onStop();

    // ATTENTION: This was auto-generated to implement the App
Indexing API.
    // See https://g.co/AppIndexing/AndroidStudio for more
information.
    Action viewAction = Action.newAction(
            Action.TYPE_VIEW, // TODO: choose an action type.
            "Maps Page", // TODO: Define a title for the content
shown.
            // TODO: If you have web page content that matches this
app activity's content,
            // make sure this auto-generated web page URL is correct.
            // Otherwise, set the URL to null.
            Uri.parse("http://host/path"),
            // TODO: Make sure this auto-generated app deep link URI
is correct.
            Uri.parse("android-
app://com.example.posicionamiento/http/host/path")
    );
    AppIndex.AppIndexApi.end(client, viewAction);
    client.disconnect();
}
```

Módulo onMapReady.

Como la aplicación está basada en un mapa sobre el cual se incluyen unos botones y se trabaja, este módulo será esencial para su funcionamiento. En él, se inicializa el mapa y sus componentes: markers, ubicación, tipo de mapa… Además se incluye la petición de permisos de ubicación y de almacenamiento, necesarios para el correcto funcionamiento de la aplicación.

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
    //Para conocer el movimiento del marker
    mMap.setOnMarkerDragListener(this);
    //Pulsar sobre el mapa
    mMap.setOnMapClickListener(this);
    //Para pulsación largar sobre el mapa
    mMap.setOnMapLongClickListener(this);
    if(checkAndRequestPermissions()) {
        if (ActivityCompat.checkSelfPermission(this,
permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

            return;
        }
        mMap.setMyLocationEnabled(true);
        getCurrentLocation();
        moveMap();
    }
}
```

Método para la solicitud de permisos:

```
private boolean checkAndRequestPermissions() {
    int permissionSendMessage =
ContextCompat.checkSelfPermission(this,
            permission.WRITE_EXTERNAL_STORAGE);
    int locationPermission = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
    List<String> listPermissionsNeeded = new ArrayList<>();
    if (locationPermission != PackageManager.PERMISSION_GRANTED) {

listPermissionsNeeded.add(Manifest.permission.ACCESS_FINE_LOCATION);
    }
    if (permissionSendMessage != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(permission.WRITE_EXTERNAL_STORAGE);
    }
    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this,
listPermissionsNeeded.toArray(new
String[listPermissionsNeeded.size()]),
REQUEST_ID_MULTIPLE_PERMISSIONS);
        return false;
    }
    return true;
}
```

Dentro de éste, se crean unos métodos para trabajar sobre el mapa como puede ser mover a una ubicación determinada o fijar un marker en dicha posición.

```
  //Metodo para obtener mi ubicación actual.
private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this,
permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
```

```
permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;
    }
    Location location =
LocationServices.FusedLocationApi.getLastLocation(client);
    if (location != null) {
        //Obtenemos latitud y longitud
        latitude = location.getLatitude();
        longitude = location.getLongitude();

    }
}

//Método para mover el mapa.
public void moveMap() {
    LatLng latLng = new LatLng(latitude, longitude);
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));


}

//Métoco para fijar un marker.
private void fixMarker2(LatLng punto) {
    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(punto);
    markerOptions.draggable(true);

markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.apo
yo));
    markerOptions.title("Apoyo numero:" + numeroapoyo);
    LatLng latLng = markerOptions.getPosition();
    latitude = latLng.latitude;
    longitude = latLng.longitude;
    mMap.addMarker(markerOptions);

}
```

Módulo onMapClick.

Cuando se pulsa se llama a este módulo cuya función es leer las coordenadas del punto y enviarlas a través de la red de datos para obtener la altitud. Despues se abre la ventana que nos indica estos datos del punto y nos pide otros para completar y guardar.

```
public void onMapClick(LatLng punto) {
    latitude = punto.latitude;
    longitude = punto.longitude;
    new
JSONTask().execute("https://maps.googleapis.com/maps/api/elevation/jso
n?locations=" + String.valueOf(latitude) + "," +
String.valueOf(longitude));
   }
```

Tras esto se ejecuta la parte de JSON que envía los datos a cierta página web para obtener la altitud. Esta será una nueva clase llamada JSONTask y esta codificada de la siguiente forma:

```
public class JSONTask extends AsyncTask<String, String, String> {


    @Override
```

```
    protected String doInBackground(String... params) {
        HttpsURLConnection connection = null;
        BufferedReader reader = null;
        try {
            URL url = new URL(params[0]);
            connection = (HttpsURLConnection) url.openConnection();
            connection.connect();
            InputStream stream = connection.getInputStream();
            reader = new BufferedReader(new
InputStreamReader(stream));
            StringBuffer buffer1 = new StringBuffer();
            String line = "";
            while ((line = reader.readLine()) != null) {
                buffer1.append(line);
            }
            String finalJson = buffer1.toString();
            JSONObject parentObject = new JSONObject(finalJson);
            JSONArray parentArray =
parentObject.getJSONArray("results");
            JSONObject finalObject = parentArray.getJSONObject(0);
            double altitude = finalObject.getDouble("elevation");

            return String.valueOf(altitude);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        } finally {
            if (connection != null) {
                connection.disconnect();
            }
            try {
                if (reader != null) {
                    reader.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        return null;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        altitud = s;
        openDialog();
    }
}
```

Todo esto se realiza sin la apreciación del usuario, a él le aparecerá una ventana desplegable con los datos de posición del punto y una serie de cuadros de diálogo para rellenar, aparte de un botón para almacenar y otro para cancelar. Esto está codificado de la siguiente manera:

```
private void openDialog() {
    //Creamos el dialog
    final AlertDialog.Builder alert = new AlertDialog.Builder(this);
```

```
    LayoutInflater inflater = (LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERVICE);
    final View v = inflater.inflate(R.layout.dialog, null);
    final EditText nombrepunto = (EditText)
v.findViewById(R.id.nombrepunto);
    alert.setView(v);
    //Titulo del dialog.
    alert.setTitle("Colocar punto");
    //Texview del dialog
    TextView coordenadas = (TextView)
v.findViewById(R.id.coordenadas);
    formulasUTM = new FormulasUTM();
    formulasUTM.cambioUTM(latitude,longitude);
    final double latitudUTM = formulasUTM.returnUTMesteX();
    final double longitudUTM = formulasUTM.returnUTMnorteY();
    final int huso = formulasUTM.returnhuso();

    coordenadas.setText(" UTM" + "\n" + " X: " +
String.valueOf(latitudUTM) + "\n" + " Y: " +
String.valueOf(longitudUTM) +"\n"+" Huso: "+ String.valueOf(huso) +
"\n"+ " Altitud: "+ altitud);

    TextView tituloSpinner = (TextView)
v.findViewById(R.id.tituloSpinner);
    tituloSpinner.setText("Elige el tipo de punto:");
    //Spinner del dialog
    final Spinner spinner = (Spinner) v.findViewById(R.id.spinner);
    String[] valores = {"Punto de traza", "Punto auxiliar"};
    spinner.setAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item, valores));
    spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {

        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }

    });
    final EditText descripcion = (EditText)
v.findViewById(R.id.descripcion);


    alert.setPositiveButton("GUARDAR", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            //Guardar los datos
            edittext1 = descripcion.getText().toString();
            edittext = nombrepunto.getText().toString();
            String latitud = Double.toString(latitude);
            String longitud = Double.toString(longitude);
            String tipo = spinner.getSelectedItem().toString();
            dataSource.saveQuoteRow(edittext,
String.valueOf(latitudUTM), String.valueOf(longitudUTM),
String.valueOf(huso), altitud, tipo, edittext1);
            LatLng latLng = new LatLng(latitude, longitude);
            fixMarker2(latLng);
```

```
        }
    });
    alert.setNegativeButton("CANCELAR", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            // Canceled.
            dialog.cancel();
        }
    });
    AlertDialog alertDialog = alert.create();
    alertDialog.show();

}
```

Clases DBHelper y QuotesDataSource.

Para poder almacenar los puntos, se crean dos clases relacionadas con la base de datos. La primera, la clase DBHelper es la encargada de crear la base de datos y estableces sus características.

```
public class DBHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "DATABASE.db";
    public static final int DATABASE_VERSION = 4;

    public DBHelper(Context context) {

        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(QuotesDataSource.CREATE_QUOTES_SCRIPT);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL(QuotesDataSource.CREATE_QUOTES_SCRIPT);


        db.execSQL("DROP TABLE IF EXISTS " +
QuotesDataSource.QUOTES_TABLE_NAME);
    }

}
```

La segunda, llamada QuotesDataSource, es la encargada de establecer las variables que se van a almacenar, su orden y la forma en que serán almacenadas.

```
public class QuotesDataSource {
    public static final String QUOTES_TABLE_NAME = "Puntos";
    public static final String STRING_TYPE = "text";
    public static final String INT_TYPE = "integer";

    //Campos de la tabla Quotes
    public static class ColumnQuotes{
        public static final String ID_PUNTOS = BaseColumns._ID;
        public static final String NOMBRE_PUNTOS = "nombre";
```

```
        public static final String LATITUD_PUNTOS = "UTM_X";
        public static final String LONGITUD_PUNTOS = "UTM_Y";
        public static final String HUSO_PUNTOS = "Huso";
        public static final String ALTITUD_PUNTOS = "altitud";
        public static final String TIPO_PUNTOS = "tipo";
        public static final String DESCRIPCION_PUNTOS = "descripción";
    }

    //Script de Creación de la tabla Quotes
    public static final String CREATE_QUOTES_SCRIPT =
            "create table "+ QUOTES_TABLE_NAME + "(" +
                    ColumnQuotes.ID_PUNTOS + " " + INT_TYPE + "
primary key autoincrement," +
                    ColumnQuotes.NOMBRE_PUNTOS + " " + STRING_TYPE + "
not null," +
                    ColumnQuotes.LATITUD_PUNTOS + " " + STRING_TYPE +
" not null," +
                    ColumnQuotes.LONGITUD_PUNTOS + " " + STRING_TYPE +
" not null," +
                    ColumnQuotes.HUSO_PUNTOS + " " + STRING_TYPE + "
not null," +
                    ColumnQuotes.ALTITUD_PUNTOS + " " + STRING_TYPE +
" not null," +
                    ColumnQuotes.TIPO_PUNTOS + " " + STRING_TYPE + "
not null," +
                    ColumnQuotes.DESCRIPCION_PUNTOS + " " +
STRING_TYPE+ " not null)";



    //Variables para manipulación de datos
    private DBHelper openHelper;
    private SQLiteDatabase database;

    public QuotesDataSource(Context context) {
        //Creando una instancia hacia la base de datos
        openHelper = new DBHelper(context);
        database = openHelper.getWritableDatabase();
    }

    public void saveQuoteRow(String nombre,String latitud, String
longitud,String huso, String altitud, String tipo, String
descripcion){
        //Nuestro contenedor de valores
        ContentValues values = new ContentValues();

        //Seteando body y author

values.put(QuotesDataSource.ColumnQuotes.NOMBRE_PUNTOS,nombre);

values.put(QuotesDataSource.ColumnQuotes.LATITUD_PUNTOS,latitud);

values.put(QuotesDataSource.ColumnQuotes.LONGITUD_PUNTOS,longitud);
        values.put(QuotesDataSource.ColumnQuotes.HUSO_PUNTOS,huso);

values.put(QuotesDataSource.ColumnQuotes.ALTITUD_PUNTOS,altitud);
        values.put(QuotesDataSource.ColumnQuotes.TIPO_PUNTOS,tipo);

values.put(QuotesDataSource.ColumnQuotes.DESCRIPCION_PUNTOS,descripcio
n);
```

```
        //Insertando en la base de datos
        database.insert(QUOTES_TABLE_NAME,null,values);
    }

    public Cursor getAllQuotes(){
        //Seleccionamos todas las filas de la tabla Quotes
        return database.rawQuery(
                "select * from " + QUOTES_TABLE_NAME, null);
    }
    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    public void deleteAll(){
        database.delete(QUOTES_TABLE_NAME,null,null);
        database.execSQL("DELETE FROM SQLITE_SEQUENCE WHERE NAME = '"
+ QUOTES_TABLE_NAME + "'");

    }

    public void deletePunto(String id) {
        database.delete(QUOTES_TABLE_NAME, "_ID = ?", new
String[]{id});
    }

}
```

Módulo onClick.

Con este módulo, se indica la función a realizar cuando uno de los botones es pulsado. En este caso los cinco botones principales de la aplicación se han incluido dentro separados por un case – break respectivamente.

- **Boton leer.**
  El botón encargado de leer los datos almacenados mediante la creación de una ventana emergente o “dialog”.

```
case R.id.buttonView:


    Cursor query = dataSource.getAllQuotes();
    if (query.getCount() == 0) {
        setMessage("error", "no se encontraron datos");
    }
    StringBuffer buffer = new StringBuffer();
    while (query.moveToNext()) {
        buffer.append("Id: " + query.getString(0) + "\n");
        buffer.append("Nombre: " + query.getString(1) + "\n");
        buffer.append("UTM X: " + query.getString(2) + "\n");
        buffer.append("UTM Y: " + query.getString(3) + "\n");
        buffer.append("Huso: " + query.getString(4) + "\n");
        buffer.append("Altitud: "+ query.getString(5)+ "\n");
        buffer.append("Tipo: " + query.getString(6) + "\n");
        buffer.append("Descripción: "+query.getString(7)+ "\n\n");
    }
    setMessage("puntos", buffer.toString());


    break;
```

Y para crear el dialog:

```
public void setMessage(String title, String message) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setCancelable(true);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.show();
```

- **Botón cambio de mapa.**
  Para la función de este botón se utiliza una ventana emergente con varias opciones seleccionables.

```
//Boton para iniciar la ventana de cambio de mapa.
case R.id.buttonChange:
    showMapTypeSelectorDialog();
    break;
```

Para ello se crean las siguientes funciónes:

```
//Métodos para mostrar un selector de mapa.
private static final CharSequence[] MAP_TYPE_ITEMS =
        {"Normal", "Satélite", "Terreno", "Híbrido"};

private void showMapTypeSelectorDialog() {
    // Preparamos el dialog con un builder
    final String fDialogTitle = "Selecciona el tipo de mapa";
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(fDialogTitle);

    // Encontramos el tipo de mapa actual para que salga check por
defecto.
    int checkItem = mMap.getMapType() - 1;

    // Añadimos on click al dialog
    builder.setSingleChoiceItems(
            MAP_TYPE_ITEMS,
            checkItem,
            new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int item)
{

                    switch (item) {
                        case 1:

mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
                            break;
                        case 2:

mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
                            break;
                        case 3:

mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
                            break;
                        default:

mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                    }
                    dialog.dismiss();
                }
            }
```

```
    );

    // Crea el dialog y lo muestra.
    AlertDialog fMapTypeDialog = builder.create();
    fMapTypeDialog.setCanceledOnTouchOutside(true);
    fMapTypeDialog.show();
}
```

- **Botón borrar.**
  Este botón será el encargado de abrir un dialog que nos dará dos opciones: borrar un punto de los guardados o borrarlos todos.

```
//Boton para borrar uno o todos los puntos de la base de datos.
    case R.id.buttonUndo:
        LayoutInflater layoutInflater = LayoutInflater.from(this);
        View promptView =
layoutInflater.inflate(R.layout.dialogborrar, null);
        final AlertDialog alertD = new
AlertDialog.Builder(this).create();
        alertD.setTitle("Borrar: ");

        final EditText userInput = (EditText)
promptView.findViewById(R.id.idpunto);
        Button btnAdd1 = (Button)
promptView.findViewById(R.id.buttonDeleteid);
        Button btnAdd2 = (Button)
promptView.findViewById(R.id.buttonDelete);

        btnAdd1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String borraid = userInput.getText().toString();
                dataSource.deletePunto(borraid);
                Toast.makeText(MapsActivity.this, "Punto borrado",
Toast.LENGTH_SHORT).show();
            }
        });
        btnAdd2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                dataSource.deleteAll();
                mMap.clear();
                numeroapoyo = 1;
                Toast.makeText(MapsActivity.this, "Todos los puntos
borrados", Toast.LENGTH_SHORT).show();
            }
        });
        alertD.setView(promptView);
        alertD.show();
}
```

- **Botón enviar.**
  Con este botón se ha de crear un correo para poder enviar nuestra lista de puntos. Este proceso se divide en varios métodos, el primero el de pulsado del botón.

```
//Boton para convertir la BD a CSV y enviar un email.
case R.id.buttonSend:
    exportDocument();
    Toast.makeText(this, "Selecciona una aplicación de correo válida",
Toast.LENGTH_LONG).show();
    String[] to = {""};
    enviar(to, "Puntos del trazado", "Archivo CSV compatible con
```

```
Excel", uri1);
    break;
```

Para su funcionamiento, se crea un método capaz de crear un archivo CSV con los datos de la base de datos, para ello se utiliza una librería específica.

```
//Método para convertir la BD a CSV
private void exportDocument() {
    DBHelper dbhelper = new DBHelper(getApplicationContext());
    File exportDir = new
File(Environment.getExternalStorageDirectory(), "");
    if (!exportDir.exists()) {
        exportDir.mkdirs();
    }

    File file = new File(exportDir, "datos.csv");
    try {
        file.createNewFile();
        CSVWriter csvWrite = new CSVWriter(new FileWriter(file));
        SQLiteDatabase db = dbhelper.getReadableDatabase();
        Cursor curCSV = db.rawQuery("SELECT * FROM Puntos", null);
        csvWrite.writeNext(curCSV.getColumnNames());
        while (curCSV.moveToNext()) {
            //Which column you want to exprort
            String arrStr[] = {curCSV.getString(0),
curCSV.getString(1), curCSV.getString(2),
curCSV.getString(3),curCSV.getString(4),
curCSV.getString(5),curCSV.getString(6)};
            csvWrite.writeNext(arrStr);
        }
        csvWrite.close();
        curCSV.close();
    } catch (Exception sqlEx) {
        Log.e("MainActivity", sqlEx.getMessage(), sqlEx);
    }
    uri1 = Uri.fromFile(file);

}
```

Y posteriormente un método para crear el correo en el cual se incluirá el archivo.

```
private void enviar(String[] to, String asunto, String mensaje, Uri
uri1) {
    Intent emailIntent = new Intent(Intent.ACTION_SEND);
    emailIntent.setType("*/*");
    emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
    emailIntent.putExtra(Intent.EXTRA_SUBJECT, asunto);
    emailIntent.putExtra(Intent.EXTRA_TEXT, mensaje);
    emailIntent.putExtra(Intent.EXTRA_STREAM, uri1);
    try {
        startActivity(Intent.createChooser(emailIntent, "Enviar
email"));
        finish();
        Log.i("Finished sending email", "");
    } catch (android.content.ActivityNotFoundException ex) {
        Toast.makeText(MapsActivity.this,
                "There is no email client installed.",
Toast.LENGTH_SHORT).show();
    }
}
```

- **Botón gráfica.**
  Con este botón se abre una ventana emergente que muestra una gráfica con la altitud de todos los puntos guardados para ello se ha utilizado una librería externa que posee un código propio.

```
//Botón para abrir la ventana de gráfico.
case R.id.buttonGraph:
    Cursor query1 = dataSource.getAllQuotes();
    while (query1.moveToNext()){
        String uname =
query1.getString(query1.getColumnIndex("altitud"));
        float alti = Float.parseFloat(uname);
        listaAltitud.add(new Entry(alti,contador));
        ejex.add("Punto "+contador);
        contador++;
    }
    LayoutInflater layoutInflater1 = LayoutInflater.from(this);
    View vista = layoutInflater1.inflate(R.layout.grafico, null);
    LineDataSet dataSet =  new LineDataSet(listaAltitud,"nº de
tramos");
    LineData lineData = new LineData(ejex,dataSet);
    dataSet.setColors(ColorTemplate.VORDIPLOM_COLORS);
    dataSet.setDrawFilled(true);

   LineChart chart = (LineChart) vista.findViewById(R.id.chart);
    chart.setData(lineData);
    chart.setNoDataTextDescription("No hay datos de altitud
disponibles.");
    chart.animateX(3000);
    chart.setDescription("Altitud en metros");
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    final AlertDialog alert = builder.create();

    alert.setTitle("Perfil longitudinal.");
    alert.setView(vista);
    Button btnAdd3 = (Button)
vista.findViewById(R.id.buttonDeleteGraph);

    btnAdd3.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            listaAltitud.clear();
            ejex.clear();
            alert.dismiss();
            contador = 0;
        }
    });

    alert.show();

    break;
```

Clase Fórmulas UTM.
La aplicación para Android de Google Maps, solo trabaja con el tipo de coordenadas latitud – longitud. Como nosotros deseamos tener estas coordenadas en formato UTM, se crea una clase con todas las fórmulas que implementan este cambio.

```
public class FormulasUTM {
    //Elegimos el elipsoide WGS84
```

```
    private static final double aSemiejeMayor = 6378137;
    private static final double bSemiejeMenor = 6356752.314;
    private static final double excentricidad = 0.081819191;
    private static final double excentricidad2 = 0.082094438;
    private static final double ecuadrado = 0.006739497;
    private static final double cRadioPolarCurvatura = 6399593.626;
    double UTMesteX;
    double UTMnorteY;
    int huso;


    //Formulas para la conversión.

    public void cambioUTM (double latitud, double longitud){
        //Cambio a radianes.
        double latitudradianes = latitud * Math.PI / 180.0;
        double longitudradianes = longitud * Math.PI / 180.0;
        double huso1 = (longitud / 6) + 31;
        huso = (int) huso1;
        int meridianohuso = 6 * huso - 183;
        double deltalambda = longitudradianes - ((meridianohuso *
Math.PI) / 180);
        double aMayus = Math.cos(latitudradianes) *
Math.sin(deltalambda);
        double xi = 0.5 * Math.log((1 + aMayus) / (1 - aMayus));
        double eta = Math.atan(Math.tan(latitudradianes) /
Math.cos(deltalambda)) - latitudradianes;
        double ni = cRadioPolarCurvatura / Math.sqrt(1 + ecuadrado *
Math.cos(latitudradianes) * Math.cos(latitudradianes)) * 0.9996;
        double zeta = ecuadrado / 2 * xi * xi *
Math.cos(latitudradianes) * Math.cos(latitudradianes);
        double a1 = Math.sin(2 * latitudradianes);
        double a2 = a1 * Math.cos(latitudradianes) *
Math.cos(latitudradianes);
        double j2 = latitudradianes + a1 / 2;
        double j4 = (( 3 * j2) + a2) / 4;
        double j6 = (5 * j4 + a2 * Math.cos(latitudradianes) *
Math.cos(latitudradianes)) / 3;
        double alfa = 3 * ecuadrado / 4;
        double beta = 5 * alfa * alfa / 3;
        double gamma = 35 * alfa * alfa * alfa / 27;
        double bfi = 0.9996 * cRadioPolarCurvatura * (latitudradianes
-(alfa * j2) + (beta * j4) - (gamma * j6));
        UTMesteX = xi * ni * (1 + zeta / 3) + 500000;
        UTMnorteY = eta * ni * (1 + zeta) + bfi;

    }
    public double returnUTMesteX(){
        return UTMesteX;

    }
    public double returnUTMnorteY(){
        return UTMnorteY;
    }
    public int returnhuso(){
        return huso;
    }

    }
```