DESARROLLO DE APLICACIÓN MÓVIL PARA EL ANÁLISIS DE RUTAS DE TRANSPORTE PARTICULAR DEL ÁREA METROPOLITANA DE BUCARAMANGA BASADO EN PLATAFORMAS DE GOOGLE Y ARCGIS.

MARÍA FERNANDA CALDERÓN PÉREZ

FRANK GIOVANNY PABÓN RODRÍGUEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA

2014

DESARROLLO DE APLICACIÓN MÓVIL PARA EL ANÁLISIS DE RUTAS DE TRANSPORTE PARTICULAR DEL ÁREA METROPOLITANA DE BUCARAMANGA BASADO EN PLATAFORMAS DE GOOGLE Y ARCGIS.

MARÍA FERNANDA CALDERÓN PÉREZ
FRANK GIOVANNY PABÓN RODRÍGUEZ

Trabajo de grado para optar al título de
Ingeniero de Sistemas

DIRECTOR:
HERNÁN PORRAS DÍAZ PhD.
INGENIERO EN TELECOMUNICACIONES

CODIRECTOR:
M.SC. JOSÉ LUIS LEAL GÓMEZ
MAGISTER EN INFORMÁTICA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

AGRADECIMIENTOS

Agradecerle a Dios primero que todo, por cada una de las bendiciones que me ha brindado durante mi vida, por darme la fuerza y la sabiduría necesaria para afrontar cada reto.

A mis padres, Nicolás y Adriana, quienes son el motor de mi vida por ser el apoyo incondicional, mi soporte y la mano que siempre ha estado ahí para levantarme en todo momento.

A mis abuelos, Belén y Fernando, quienes han sido como unos padres para mí, siempre pendientes y dispuestos a colaborarme en todo momento, este logro va especialmente dedicado a ellos.

A mis hermanos, Daniel y Nicolh, por ser uno de mis motivos para seguir adelante y darles el mejor ejemplo.

A Frank, quien no solo fue mi compañero de proyecto si no también mi mejor amigo, con su apoyo, inteligencia, paciencia, amistad y alegría hicieron posible la culminación de este proyecto.

A mi director de proyecto, el profesor Hernán Porras, por haber brindado el apoyo y la guía durante todo el desarrollo. A nuestro codirector, José Luis Leal, por haber compartido cada uno de sus conocimientos y el acompañamiento durante la ejecución del proyecto.

A mis amigos, con los que he compartido magníficos momentos llenos de alegrías, por estar ahí dando el apoyo moral, no solo aportaron a mi vida profesional sino también a mi crecimiento como persona.

María Fernanda Calderón Pérez

AGRADECIMIENTOS

Agradecerle a Dios, que me permitió llegar hasta este punto y me ha acompañado a lo largo de toda mi vida.

A mi madre María Antonia, que ha sido mi mayor apoyo, que con sus cuidados y cariño me dio la energía para emprender cada día de mi vida.

A mi padre Florentino, que ha sido mi ejemplo a seguir y el que en mis momentos de mayor debilidad me dio su mano para no desfallecer.

A mis hermanos Irma y Germán, que me han dado su apoyo incondicional sin importar las distancias ni el tiempo.

A Fernanda, mi compañera de proyecto y mejor amiga, que sin su ánimo, optimismo e ideas, este proyecto no hubiera podido ser posible.

A Lina, por llegar y darle un giro muy lindo a mi vida, dándome amor, su amistad y brindándome ganas de salir adelante.

A mi director de proyecto, el profesor Hernán Porras, por habernos guiado y habernos dado su espaldarazo de apoyo en este proyecto. A mi codirector José Luis, por habernos encaminado en la consecución de este trabajo.

A todos mis amigos y personas que conocí haciendo la carrera, a los que estuvieron desde antes, a los que siguen y a los que se fueron, gracias por todas las pequeñas enseñanzas que me han ido forjando como persona.

Frank Giovanny Pabón Rodríguez

CONTENIDO

pág

LISTA DE FIGURAS

pág

LISTA DE ANEXOS

RESUMEN

TÍTULO: DESARROLLO DE APLICACIÓN MÓVIL PARA EL ANÁLISIS DE RUTAS DE TRANSPORTE PARTICULAR DEL ÁREA METROPOLITANA DE BUCARAMANGA BASADO EN PLATAFORMAS DE GOOGLE Y ARCGIS.[1]

AUTORES: María Fernanda Calderón Pérez[2]

Frank Giovanny Pabón Rodríguez[2]

PALABRAS CLAVE: AMB, Área metropolitana de Bucaramanga, Aplicación Móvil, Google, ArcGIS, Android, Ruta.

DESCRIPCIÓN

La implementación de los servicios de mapas de Google Maps y de ArcGIS para la búsqueda de rutas eficientes, dentro de aplicaciones desarrolladas en lenguaje Android tiene como objetivo contribuir al desarrollo de herramientas de tecnologías de información para el área metropolitana de Bucaramanga, así como su desarrollo bajo el concepto de SmartCity. Las aplicaciones que se presentan en este proyecto consisten básicamente de código Android básico unido a código Android que implementa las APIs tanto de Google Maps, como el SDK de ArcGIS.

Este proyecto cuenta con una explicación conceptual básica dentro del desarrollo del marco teórico para posteriormente profundizar en el desarrollo de las aplicaciones, mostrando desde el desarrollo de la malla vial metropolitana, pasando por los procesos de integración de los servicios de mapas con Android, hasta mostrar cómo se ensamblaron finalmente las aplicaciones. Se destaca que dentro de este proyecto se implemento la metodología ágil SCRUM, para obtener un desarrollo rápido y enfocado a las funcionalidades de las aplicaciones.

A modo de resultado, además de las aplicaciones, se desarrolló un informe estadístico del desempeño de las rutas generadas por los servicios de mapas dentro de las aplicaciones teniendo como criterios de evaluación : veracidad de las rutas, tiempos de respuesta al procesamiento de las rutas y distancia de las rutas; sobre los cuales se hizo un análisis detallado, desarrollándose una calificación por puntos para obtener al sobre cuál de los dos servicios de mapas ofrece mejores prestaciones y es más eficiente a la hora de definir rutas dentro del área metropolitana de Bucaramanga.

[1] Trabajo de grado

[2] Facultad de Ingenierías Fisico-Mecanicas. Escuela de Ingeniería de Sistemas. Director: Hernan Porras Díaz PhD. Ingeniero en Telecomunicaciones.

ABSTRACT

TITLE: DEVELOMENT OF MOBILE APLICATION FOR ANALYSIS OF ROUTES FOR METROPOLITAN AREA OF BUCARAMANGA BASED ON PLATAFORMS OF GOOGLE AND ARCGIS.[1]

AUTHORS: María Fernanda Calderón Pérez[2]

Frank Giovanny Pabón Rodríguez[2]

KEYWORDS: AMB, Metropolitan Area of Bucaramanga, Mobile Application, Google, ArcGIS, Android, Route,

DESCRIPTION:

The implementation of the mapping services for Google Maps and ArcGIS for finding of efficient routes, within of applications developed on Android language has as a aim to contribute to the development of information technology tools for the Metropolitan area of Bucaramanga and its development under the concept of SmartCity. The applications presented in this project are basically basic Android code attached to Android code that implements the APIs of both Google Maps, and ArcGIS SDK.

This project has a basic conceptual explanation between the developments of the theoretical framework in order to go deeper in the development of the applications, showing from the design of the metropolitan road network, including the processes of integration of map services with Android, till how applications were finally assembled. It is noted that within this project the SCRUM methodology was used for rapid development and to pay attention to the essential functionalities.

As a result, in addition to the applications, was developed a statistical report about the performance of the routes generated by the mapping services within the applications by measuring values such as truthfulness routes, response times to the processing route and distance routes; for which was made a detailed analysis, developing a points rating on which of the two map services offers better performance and is more efficient defining routes in the metropolitan area of Bucaramanga.

[1] Bachelor Thesis

[2] Facultad de Ingenierías Fisico-Mecanicas. Escuela de Ingeniería de Sistemas. Director: Hernan Porras Díaz PhD. Ingeniero en Telecomunicaciones.

INTRODUCCIÓN

El área metropolitana de Bucaramanga como cualquier ciudad en procesos de desarrollo, presenta problemas de movilidad. Aunque actualmente se cuenta en el mercado con una gran cantidad de aplicaciones especializadas en la búsqueda de rutas, se desea obtener una aplicación focalizada en la movilidad del transporte particular en el área.

Este proyecto consiste en el desarrollo de dos aplicaciones móviles cada una basada en los servicios de mapas de Google o ArcGIS, en donde se permite la búsqueda de rutas a partir de un punto de origen, localizado con GPS o demarcado por el usuario, hacia un punto de destino, demarcado por el usuario. Esta búsqueda puede ser realizada con cualquiera de los dos servicios. A partir de datos generados de los resultados de las búsquedas de una base de datos de puntos de origen y destino, se plantea realizar un estudio estadístico de cada servicio.

El desarrollo de este proyecto tiene como fin principal determinar cuál de los servicios de mapa implementados, puede brindar una mejor solución para de esta manera ser implementado en futuros proyectos con el propósito de entregar al área una aplicación eficiente y eficaz.

1. GENERALIDADES DEL PROYECTO

1.1 DESCRIPCIÓN GENERAL

La movilidad dentro de las ciudades en proceso de desarrollo es uno de los problemas más importantes, no solo para el trasporte público sino también para el particular; cada vez existe mayor congestión vehicular conllevando al aumento de los tiempos para poder movilizarse de un sitio a otro. Debido al boom de las aplicaciones móviles en el mundo, hoy existen infinidad de aplicaciones que buscan dar soluciones a la movilidad de las ciudades, pero aun así estas aplicaciones presentan algunos errores derivados de la actualizaciones de las mallas vehiculares en la ciudades y otros factores.

Con el ánimo de brindar una solución al problema de la movilidad vehicular en el área metropolitana de Bucaramanga se plantea el desarrollo de dos aplicaciones móviles, cada una basada en los servicios de mapas de Google o ArcGIS, para con el determinar cuál de estos brinda mayor eficiencia. Basándonos en dos mallas; una proporcionada por el grupo de investigación Geomática, y la otra la generada por Google, se desea hacer un estudio sobre la eficiencia de cada una para así determinar cuál de estos servicios puede brindar una mejor solución al problema de movilidad.

1.2 DESCRIPCIÓN DEL PROBLEMA

Actualmente, en los países en vías de desarrollo y más específicamente en sus ciudades, se está experimentando una tasa de crecimiento poblacional bastante acelerada, por consiguiente estás ciudades también experimentan un crecimiento que se da sin un control o sin planes de ordenamiento territorial, lo que a corto o largo plazo termina por generar problemas inherentes a ciudades en vías de

desarrollo, como: alta densidad poblacional, contaminación visual, auditiva y del ambiente.

Este proyecto, surge como una alternativa de solución a uno de esos problemas de las ciudades en desarrollo, el cuál es el excesivo tráfico vehicular. El exceso del parque automotor en las ciudades, termina por originar trancones, lo cuales tienen efectos directos sobre las personas, tales como: pérdida de tiempo para desplazarse de un lugar a otro, incomodidad en los desplazamientos, colapso de las vías vehiculares, entre otros.

Hoy en día, este es uno de los mayores problemas del área metropolitana de Bucaramanga, el cual se ha acentuado de forma notoria, debido a que hay un aumento poblacional constante junto con un aumento constante del parque automotor, pero sin aumento de la extensión del territorio ni la creación de nuevas vías vehiculares.

1.3 JUSTIFICACIÓN

El problema de movilidad que presenta el área metropolitana de Bucaramanga cada vez es más latente; la implementación del servicio de transporte masivo, las constantes obras en las vías, el aumento de parque automotor sumado a los desconocimientos de rutas alternativas por parte de los usuarios, hace que este problema crezca. El gobierno ha planteado soluciones como el pico y placa para ayudar al descongestionar las vías, pero aun así no se ven mejoras en la movilidad.

Actualmente se está buscando desarrollar a nivel local una serie de aplicaciones en dispositivos móviles, las cuales puedan convertir a Bucaramanga en una ciudad inteligente bajo el concepto de SmartCity, con el fin mejorar la calidad de vida de las personas que se encuentren dentro del área metropolitana.

El propósito del desarrollo de una aplicación especializada en el análisis de rutas vehiculares, es brindarle a los usuarios soluciones con las cuales puedan tener rutas eficientes para llegar a sus destinos, basadas no solo el parámetros como distancia, si no teniendo en cuenta factores como las horas o el pico el placa. Pero para poder desarrollar una aplicación a este nivel, es necesario conocer cual servicio puede llegar a brindar un mejor rendimiento derivado a sus características propias.

Junto con el apoyo del grupo de investigación Geomática, de la Escuela de Ingeniería Civil, se plantea el desarrollo de dos aplicación móviles nativas para Android basadas en servicios de mapas de las plataformas de ArcGIS o Google, para el análisis de rutas con la cual se puedan obtener datos acerca del rendimiento y con ellos un informe estadístico comparativo sobre los servicio de mapas.

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Ayudar a la mejora de la movilidad del transporte particular en el área metropolitana de Bucaramanga a través del desarrollo de una aplicación móvil en Android con base en los servicios de mapas web de Google Maps y ArcGIS, para generar la búsqueda de rutas y localización de sitios de interés.

2.2 OBJETIVOS ESPECÍFICOS

- Estructurar la malla vial del área metropolitana de Bucaramanga, de forma que pueda ser publicada de manera web en formato ESRI, a partir del dibujo vectorial de la malla vial proporcionada por el grupo de investigación Geomática.

- Desarrollar dos aplicaciones para dispositivos móviles Android, una basada en servicios de mapas de Google Maps y otra en los servicios de mapas de ArcGIS, que genere alternativas de rutas de transporte particular a partir de un origen y un destino.

- Producir un informe estadístico comparativo acerca del rendimiento del uso de los recursos ofrecidos en Google Maps y la malla de ArcGIS construida en relación a la generación de rutas.

- Analizar los resultados del estudio estadístico para generar una fuente de información para la implementación eficiente de los servicios de mapas de Google Maps y ArcGIS.

2.3 ALCANCE

Se desarrollará dos aplicaciones que cuente con los servicios de mapas de Google Maps y ArcGIS, cuyo propósito sea el análisis de rutas sobre la malla vial del área metropolitana de Bucaramanga, proporcionada por parte del grupo de investigación Geomática. El cálculo de rutas tendrá como variable de decisión la distancia, tomando como referencia un punto de partida, indicado por el usuario o georreferenciado por GPS y un punto de destino, seleccionado por el usuario. Mediante este desarrollo se obtendrá información de la eficiencia de cada uno de los servicios de mapas, con el fin de definir cuál de ellos brinda mejores resultados en la búsqueda de rutas. Permitiendo seleccionar el servicio de mapas más adecuado para gestionar una solución acertada al problema de movilidad dentro del área metropolitana para sus tanto habitantes.

3. MARCO TEÓRICO Y METODOLÓGICO

3.1 ANDROID

Sistema operativo basado en el kernel de Linux, se usa en Smartphones, ordenadores portátiles, notebooks, tabletas, Google TV, entre otros. Desarrollado por Android, Inc, inicialmente y respaldado económicamente para ser final comprado en el 2005 por la compañía Google. Android fue presentado en 2007 junto la fundación del Open Handset Alliance, un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++. La plataforma de hardware principal es la arquitectura ARM. Hay soporte para x86 en el proyecto Android-x86,55 y Google TV utiliza una versión especial de Android x86.

Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones. Generalmente cada actualización del sistema operativo Android

es desarrollada bajo un nombre en código de un elemento relacionado con postres en orden alfabético : A, Apple Pie (v1.0); B, Banana Bread (v1.1); C, Cupcake (v1.5); D, Donut (v1.6); E, Éclair (v2.0/v2.1); F, Froyo (v2.2); G, Gingerbread (v2.3); H, Honeycomb (v3.0/v3.1/v3.2); I, Ice Cream Sandwich (v4.0); J, Jelly Bean (v4.1/v4.2/v4.3); K, KitKat (v4.4), esta es la última versión; L, Lime Pie (v4.6-5.0), esta es la próxima versión que se espera sea lanzada en Junio o Julio.

Android cuenta con una tienda oficial de aplicaciones, Google Play, donde los usuarios pueden encontrar aplicaciones gratuitas o pagas, siendo estas las de menor valor en el mercado comparada con el App Store de iOs. En la actualidad existen aproximadamente 1.000.000 de aplicaciones para Android y se estima que 1.500.000 teléfonos móviles se activan diariamente, para el 2013 había ya más de 1000 millones de Smartphones Android.

3.2 LOCALIZACIÓN

3.2.1 ¿Qué es localización?

La palabra localización alude a ubicación espacial (del latín “locus” que indica lugar), término usado en especial en geografía para identificar donde están ubicados ciudades, países, objetos o personas. Mediante de las coordenadas geográficas se identifica un punto de la superficie terrestre, con dos números que indican la latitud y la longitud geográfica. Desde la Edad Antigua se han implementado distintos elementos, científicos o artesanales, para la localización: mapas, brújulas, sextantes, teodolito entre otros; recientemente se ha generalizado el uso de los dispositivos GPS (sistema de posicionamiento global)

3.2.2 GPS

El Sistema de Posicionamiento Global es un sistema de navegación basado en el espacio que provee localización y la hora en todas las condiciones meteorológicas, fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos. El sistema GPS está constituido por 24 satélites en órbita sobre el planeta tierra, a 20.200 km, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra.

Utiliza la triangulación para determinar en todo el globo la posición con una precisión de más o menos metros, en otras palabras el receptor localiza como mínimo tres satélites de la red, los cuales envían señales indicando la identificación y hora del reloj de cada uno de ellos, con base en ellos el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia de cada satélite respecto al punto de medición (Triangulación). Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición.

3.3 SERVICIOS BASADOS EN LOCALIZACIÓN

Los servicios basados en localización o LBS (Location Bases Service) buscan ofrecer un servicio personalizado a los usuarios basándose en la mayoría de situaciones en información de ubicación geográfica , para ello utilizan tecnología de sistemas de información geográfica, ya sea de posicionamiento de lado cliente (GPS) o del lado del servidor (Servicio de posicionamiento suministrado por el operador de red NBL) y tecnología de comunicaciones de redes para la transmisión de la información hacia aplicaciones LBS que puedan procesar y responder a la solicitud.

La aplicaciones LBS tienen como propósito proveer servicios geográficos en tiempo real, un claro ejemplo son los servicios de mapas, enrutamiento y páginas amarillas geográficas. Este tipo de aplicaciones cuentan con dos tipos de escenarios: Usuario necesita información geográfica y centro de administración, necesita rastrear a un usuario en tiempo real

En ambos escenarios, la aplicación obtiene la información de la posición del usuario a través de algún mecanismo, como el GPS de un dispositivo móvil, esta información junto con otros parámetros relevantes se transmite a un centro de procesamiento y almacenada en una base de datos espacial, siendo analizados por una infraestructura apoyada en sistemas de información geográfica y así responder a la solicitud del usuario.

LBS cuenta con dos formas de trabajo, activa y pasiva. Un LBS activo se enfoca en usuarios de móviles particulares para proveerles información de servicios; Un LSB pasivo es diseña para clientes empresariales, donde los requerimientos son administrar los recursos móviles y soportar toma de decisiones.

Algunos ejemplos de aplicaciones LBS en el mercado son:

- **TomTom Mobile:** proporciona información de navegación, mostrando en una pantalla del móvil o PDA un mapa con las instrucciones para llegar a un destino, con tiempo estimado de duración y puntos de interés cercanos.

- **Nokia Sports Tracker:** herramienta de monitoreo por GPS para dispositivos compatibles con Nokia. Realiza un seguimiento de la actividad física, trazando la ruta que recorre el usuario y proporcionando información relativa a la velocidad, la distancia y el tiempo de la actividad. Permite en cualquier momento analizar y compartir en la web todos los datos de la actividad deportiva y las rutas asociadas.

- **Movistar Localízame:** localiza al usuario dentro de la red móvil y comparte esta información con contactos. También permite registrar periódicamente la posición para luego trazar la ruta seguida sobre un mapa.

- **ViaMichelin Web Services:** proveedor de contenidos turísticos. Brinda una solución para los siguientes tipos de problemáticas: localizar establecimientos, seleccionar zonas comerciales, ver en un mapa información local, posicionar un recurso móvil en la

3.4 DEFINICIÓN API

Interfaz de programación de aplicaciones, por sus siglas en inglés “Application Programming Interface”, especifica cómo algunos componentes del software deben interactuar con otros. API representa la capacidad de comunicación entre los componentes de distintos software, siendo un conjunto de librerías que incluyen las especificaciones para las rutinas, la estructura de datos, los objetos, las clases y las variables, por ejemplo el acceso a bases de datos o al hardware (unidades de disco duro o tarjetas de video) y facilitar el trabajo de la programación de la interfaz gráfica de usuario.

3.5 DEFINICIÓN SDK

Kit de Desarrollo de Software o por sus siglas en inglés “Software Development Kit”, es un conjunto de herramientas de desarrollo de software que permiten crear aplicación para un sistema en específico, paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos o plataformas de desarrollo similares. Un SDK puede ser como la aplicación de una o varis en forma de librerías de un lenguaje en particular o incluir hardware para la comunicación con un sistema embebido. Algunas herramientas comunes del SDK

incluyen ayudas de depuración, con frecuencia traen documentación de apoyo como ejemplos de código y notas técnicas.

Los proveedores de los SDK para los sistemas o subsistemas específicos a veces pueden sustituir un término más específico en lugar de software, como lo es en el caso de Microsoft y Apple, ofrecen kits de desarrollo de controladores (DDK) para el desarrollo de controladores de dispositivos.

3.6 GOOGLE MAPS

3.6.1 Definición

Es una aplicación web de servicios de mapas desarrollada por Google, conocido inicialmente como Google Local. Brinda mapas de las calles, rutas entre diferentes puntos, con diferentes tipos de movilización (a pie, carro, bicicleta, o con el transporte público) incluyendo localización de sitios desplazables, imágenes de Google Street View y fotografías por satélite del mundo.

3.6.2 API de Google Maps

El API de Google Maps es un conjunto de diversas API que permite a los usuarios integrar Google Maps en una web o en una aplicación móvil, proporcionando utilidades para manipular los mapas y añadir contenido al mapa mediante diversos servicios. Existe un versión gratuita del API el cual tiene la restricción de que el servicio debe estar disponible para los usuarios finales de forma pública y gratuita. También está la versión “For Business” la cual brinda una asistencia más completa a organizaciones que añaden mapas a sus aplicaciones móviles, sitios web de pago o internos. Por medio del uso de este conjunto de API se pueden crear

aplicaciones basados en la ubicación, visualización de datos geoespaciales, crear mapas para aplicaciones móviles y personalización de mapas.

Para nuestro estudio nos centraremos en la creación de aplicaciones basadas en la ubicación y la creación de mapas para aplicaciones móviles. El desarrollo de aplicaciones basadas en la ubicación se hace más sencillo gracias a la aporte de los diferentes API como: Google Places la cuales brindan el servicio de ubicación de sitios de interés; el API de rutas, se implementa para el cálculo de las rutas; y el API de Matriz de Distancia para el análisis de rutas.

Para la creación de mapas para aplicaciones móviles existen varias API que permiten insertar mapas y servicios de ubicación a cualquier aplicación para dispositivos móviles. Para realizar esto se pueden implementar la versión 2 del API de Google Static Maps y el API de Google Places.

3.6.2.1 Version 2 API de Google Static Maps

El API de rutas de Google es un servicio que utiliza una solicitud HTTP para calcular rutas para llegar de una ubicación a otra. En ella se puede especificar el medio de transporte, como en transporte público, en coche, a pie o en bicicleta. Las rutas pueden especificar los orígenes, los destinos y los hitos como cadenas de texto o como coordenadas de latitud/longitud. El API de rutas puede devolver rutas segmentadas mediante una serie de hitos.

Está API tiene un límite de 2.500 solicitudes de rutas al día. Todas las búsquedas de indicaciones contarán como una única solicitud respecto al límite diario cuando el modo de transporte es en coche, a pie o en bicicleta. La búsqueda de indicaciones en transporte público contará como cuatro. Las solicitudes individuales de indicaciones en coche, a pie o en bicicleta pueden contar como un máximo de ocho hitos intermedios en la solicitud.

3.6.2.2 API de Google Places

El API de Google Places permite encontrar información detallada sobre sitios pertenecientes a un gran número de categorías, por ejemplo establecimientos, sitios de interés destacados, ubicaciones geográficas entre otros; gracias a que hace uso de la misma base de datos de Google Maps muestra más de 95 millones de empresas y puntos de interés, estos son actualizados por contribuciones de los usuarios y verificadas por los propietarios.

Este API cuenta con los siguientes tipos de solicitudes de sitios disponibles:

- **Búsquedas de sitios:** se obtiene una lista de sitios basada en una cadena de búsqueda o en la ubicación del usuario.
- **Detalles de sitios:** se obtiene información más detallada sobre un sitio específico, incluidas las opiniones de los usuarios.
- **Acciones de sitios:** Permite complementar datos con la base de datos de Google, además de crear eventos, añadir o eliminar sitios o ponderar la clasificación de los sitios en función de la actividad de los usuarios con los registros de visitas.
- **Fotos de sitios:** permiten acceder a millones de fotos relacionadas con los sitios almacenados en la base de datos de Google.
- **Autocompletado de sitios:** esta función permite completar automáticamente el nombre y la dirección de un sitio a medida que se escribe.
- **Autocompletado de consultas:** proporcionar un servicio de predicción de consultas para búsquedas geográficas basadas en texto al ofrecer consultas sugeridas a medida que se escribe.

A estos servicios se accede mediante una consulta HTTP, y se obtiene una respuesta JSON o XML. Todas las solicitudes a un sitio deben utilizar el protocolo https:// e incluir tanto una clave como un parámetro sensor.

3.7 ARCGIS

ArcGIS es un sistema compuesto por una serie de programas de software, herramientas profesionales que permiten realizar trabajos SIG, una infraestructura on-line basada en la nube, recursos confiables como plantillas de aplicación, mapas bases listo para utilizar y contenido compartido por la comunidad de usuarios. Gracias a la compatibilidad con las plataformas de servidor y de la nube posibilitan la colaboración y el uso compartido, se logra garantizar la disponibilidad de la información vital para la planificación y toma de decisiones.

Este sistema que permite recopilar, organizar, administrar, analizar, compartir y distribuir información geográfica. Es implementada montar el conocimiento geográfico al servicio de los sectores del gobierno, la empresa, la ciencia, la educación y los medios. Se puede acceder por medio de navegadores Web, dispositivos móviles como smartphones y equipos de escritorio. Una de las ventajas del uso de ArcGIS es que permite realizar actualizaciones mediante los dispositivos móviles en tiempo real siendo posible observarlas desde un equipo de escritorio

ArcGIS permite: crear, compartir y utilizar mapas inteligentes, compilar información geográfica, crear y administrar bases de datos geográficas, resolver problemas con el análisis espacial, crear aplicaciones basadas en mapas, dar a conocer y compartir información mediante la geografía y la visualización

Mediante la creación de aplicaciones ser permite el uso de los mapas, datos y herramientas, siendo posibles implementarlas en la web, equipos de escritorios o dispositivos móviles. Para la creación de aplicaciones móviles ArcGIS proporciona un conjunto de API , para JavaScript , Adobe Flex , Microsoft Silverlight y HTML5 , y kits de desarrollo de software SDK, disponibles para Android, Apple iOS, Windows Phone y Windows Mobile, junto con bibliotecas de muestra de código.

3.7.1 Dataset de Red

Un Dataset de red es implementado para la modelación de redes de transporte, se crean a partir de entidades de origen, incluyendo entidades simples, como líneas y puntos, y giros, y almacenar la conectividad de las entidades de origen. Para realizar análisis sobre ellos se implementa la herramienta ArcGIS Network Analyst.

Para crear un Dataset de Red existen varias opciones:

- A partir de clases de entidad en un dataset de entidades de una geodatabase, debido a que este puede almacenar y comunicarse con facilidad con varias clases de entidad, además este puede admitir varios orígenes y modelar una red de varios modelos, esta forma es considerada la mejor.

- A partir de un shapefile de polilínea que contiene la fuente de red y, opcionalmente, una clase de entidad de giros de shapefile. Este tipo de dataset no admite varios orígenes de borde, ni modelar redes de varios modelos.

3.7.2 ArcGIS Network Analyst

ArcGIS Network Analyst es una extensión de ArcGIS proporciona las herramientas para generar un Dataset de red y realizar análisis sobre él. Esta extensión proporciona análisis espacial basado en la red, tales como enrutamiento, enrutamiento de la flota, direcciones de viaje, ubicación más cercana, área de servicio, y la ubicación y asignación. Mediante su uso se logra modelar dinámicamente condiciones realistas de la red, incluyendo las calles de un solo sentido, girar y restricciones de altura, límites de velocidad y las velocidades de viaje de variables en función del tráfico.

Esta extensión ofrece las siguientes utilidades:

- Encontrar rutas más cortas
- Producir rutas eficientes para la movilización de vehículos con muchos destinos.
- Implementar ventanas de tiempo para generar limitaciones de movilización.
- Permite localizar sitios cercanos a un punto.
- Determinar ubicaciones óptimas para instalaciones mediante análisis de ubicación y asignación.
- Definir áreas de servicio basadas en el tiempo de viaje o distancia.
- Crear redes utilizando datos GIS existentes.
- Generar matrices de gastos de viaje de red desde cada origen a todos los destinos.

3.7.3 ArcGIS Runtime SDK para Android

ArcGIS Runtime SDK para Android permite añadir capacidades de ArcGIS a las aplicaciones programadas en Android. El SDK es un archivo que se expande en el disco. Se incluye una API de base y marco de aplicación de la API bibliotecas, librerías nativas, muestras, doc referencia de la API, y un plugin de Eclipse.

El SDK permite:

- Añadir y el contenido de consulta Organización ,utilizando el Portal de la API
- Añadir capas de ArcGIS Server

- Trabajar sin conexión con mapas base y los datos operativos
- Añadir mapas base y mapas de caché baldosas locales en área de interés almacenados localmente en el dispositivo
- Mapas presentación en todos los soportados proyecciones móviles de tiempo de ejecución
- Utilizar conjunto de tareas que las capacidades de ArcGIS apalancamiento para analizar los mapas y proporcionar información a los usuarios
- Añadir herramientas para editar datos, y sincronizarlas cuando están fuera de línea
- Trabajar con GPS de los dispositivos
- Construir fuera de línea y en línea de enrutamiento aplicaciones
- Realizar operaciones geométricas avanzadas a nivel local
- Ejecutar tareas de geo procesamiento sofisticados y mostrar los resultados
- Búsqueda, consulta, e identificación de características que utilizan criterios espaciales o SQL

3.8 METODOLOGÍA SCRUM

Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental, es un entorno basado en el desarrollo ágil de software. Es la forma en que un equipo puede trabajar conjuntamente en el desarrollo de un producto, proporcionando reglas que crean una estructura de equipo capaz de enfocarse en innovar soluciones. Está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados

pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Se tienen en cuenta tres roles fundamentales: ScrumMaster: mantiene los procesos y trabaja de forma similar al director de proyecto; Dueño del producto: representa a los stakeholders (interesados externos o internos); Equipo: incluye a los desarrolladores.

3.8.1 Pila del Producto

El proceso parte de la Pila del producto, conformada por las historias de usuario, que actúa como plan de proyecto. La pila de producto es una lista priorizadas de requisitos, funcionalidades o historias, que el cliente espera en el proyecto. Cada historia que se genera contiene campos específicos:

- **ID:** Identificador único, auto incrementa.
- **Nombre:** Un descripción corta de la historias.
- **Estimación inicial:** la valoración inicial de trabajo que puede requeriré la historia en realizarse en comparación con las demás. Esta estimación se da en “días persona ideales”.
- **Como Probarlo:** Descripción de alto nivel de cómo se demostrara la historia en la demo final del sprint, es decir el “testing” que debe realizarse para comprobar su funcionamiento.
- **Notas:** información extra de requerimientos o clarificación. Normalmente muy breve. Cada historia a su vez puede tener varias tareas específicas que la conforman

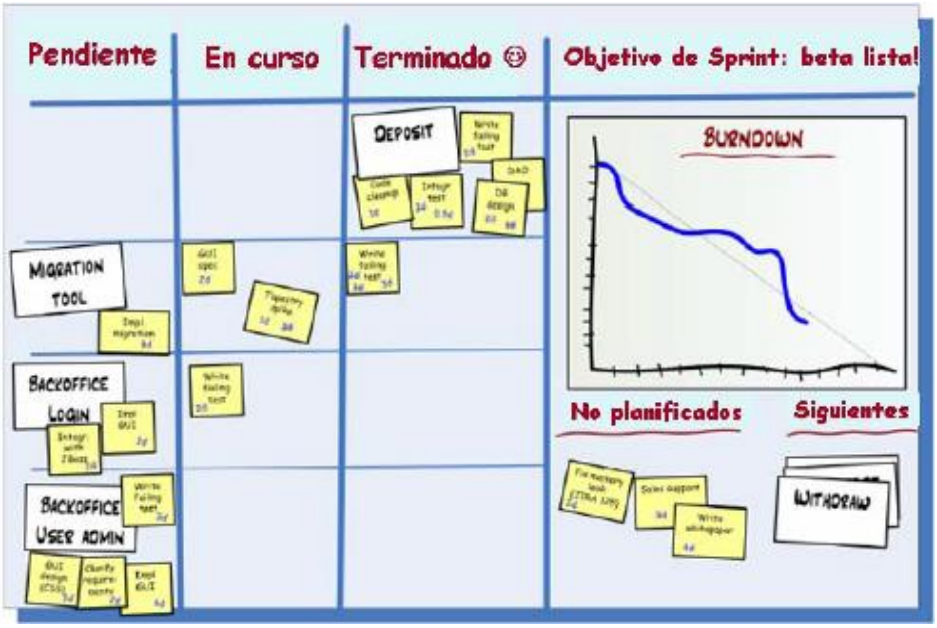
3.8.2 Planificación Sprint

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos, llamado Sprint, que proporcionar un resultado completo funcional de una meta previamente planteada .Cada Sprint se compone de una cantidad de historias de usuario, definidas previamente por: cálculos de velocidad, se decide la velocidad estimada "cantidad de trabajo" y se calculan cuantas historias dependiendo a su estimación inicial pueden entrar al sprint sin sobrepasar la velocidad estimada, o por consenso entre el equipo, el ScrumMaster y el Dueño del producto.

3.8.3 Gestionar la Pila del Sprint Diario

Para gestionar la pila del sprint, se empieza con el desarrollo de las historias, para ello existen varios sistemas que van desde una tabla de tareas hasta paquetes de software. La tabla de tareas consiste en el uso de una pizarras y "pos-it" en la cual inicialmente se trazan cuatro columnas: Pendientes, inicialmente se ubican todas las historias con sus respectivas tareas; En curso, se ubican las tareas de las historias en las cuales se está trabajando; Terminado, se ubican las tareas que se van terminado o la historia ya culminada; Objetivo de Sprint, se encuentra el Burndown , este consiste en un representación gráfica entre el trabajo restante y la fecha del sprint, en la cual cada día dependiendo al trabajo realizado diario se demarca el punto de trabajo restante, esto se realiza con el fin de ir revisando el trabajo del sprint y comprobar si se va allegar a la meta en el tiempo estimado, también se encuentra una pila de historias no planificada y de siguientes por si sobra tiempo en el sprint se puedan realizar. El siguiente grafico puede ilustrar mejor este método.

Figura 1. Tabla de tareas Pila Sprint



Fuente: HENRIK KNIBERG. SCRUM y XP desde las trincheras

4. DESARROLLO APLICACIONES MÓVILES EN ANDROID BASADAS EN SERVICIOS DE MAPAS DE GOOGLE MAPS Y ARCGIS: PASO A PASO

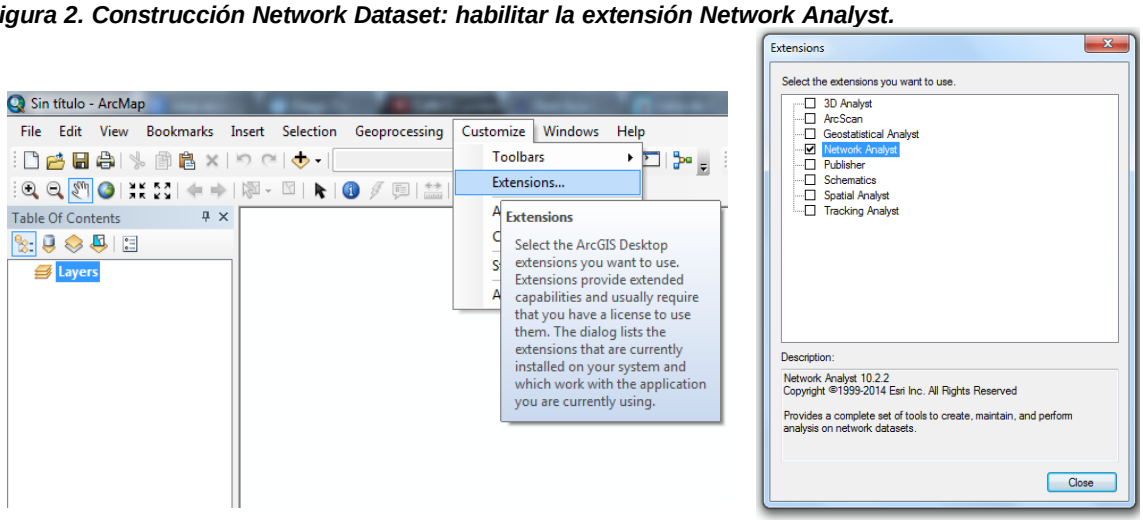
4.1 CONSTRUCCIÓN Y PUBLICACIÓN DEL NETWORK DATASET AMB

Para el desarrollo de la aplicación móvil basada en los servicios de mapas en ArcGIS, es necesario contar con un MapServer y un NAServer, siendo estos la vista del mapa geográfico y el otro la malla vial del mapa. Para ello basándonos en la malla vial proporcionada por el grupo de investigación Geomática se comienza la construcción del Network dataset para su final publicación,.

4.1.1. Construcción Network Dataset AMB

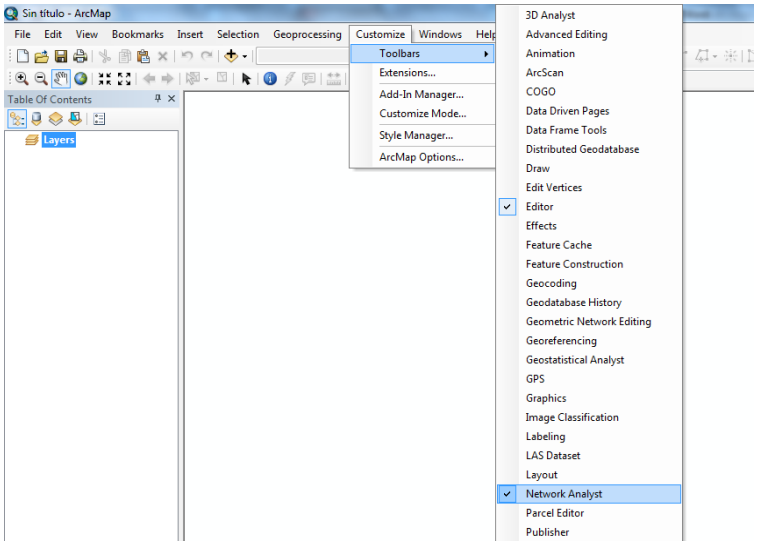
La construcción del Network Dataset se realiza en ArcMap. El primer paso a ejecutar será habilitar la extensión de *Network Analyts* y la barra de herramientas de *Network Analyst.* En la barra de menú principal *Customize>Extensions,* en la ventana emergente *Extensions* seleccionamos *Network Analyst* y se cierra la venta con el botón *Close*,

Figura 2. Construcción Network Dataset: habilitar la extensión Network Analyst.



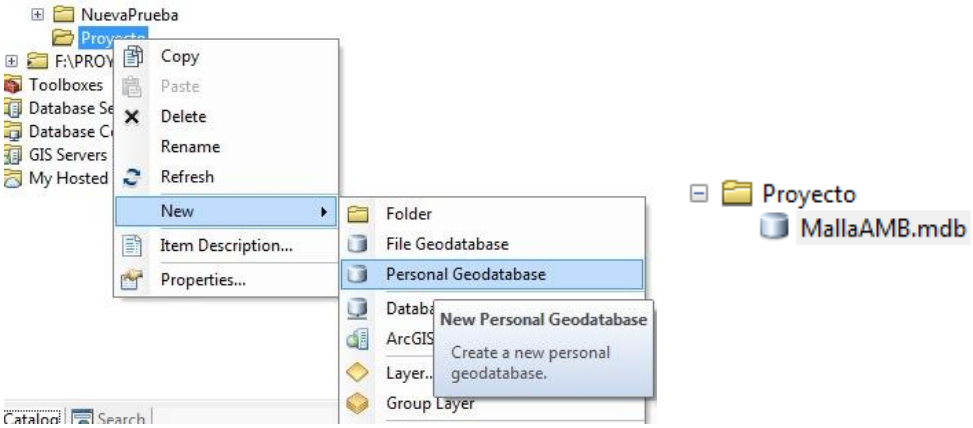
Ahora para habilitar la barra de herramientas, en la barra de menú principal *Customize>Toolbars>Network Analyst.*

Figura 3. Construcción Network Dataset: habilitar la barra de herramientas Network Analyst.



Luego de habilitar la extensión y la barra de herramientas, se debe crear dentro del folder del proyecto una "Personal Geodatabase", esta permitirá que se pueda publicar el servicio. En la sección del *Catálog* ubicándose sobre el folder del proyecto se da clic derecho *New>Personal Geodatabase,* se asigna el nombre, para este caso el nombre será *MallaAMB.*

Figura 4. Construcción Network Dataset: creación Personal geodatabase.

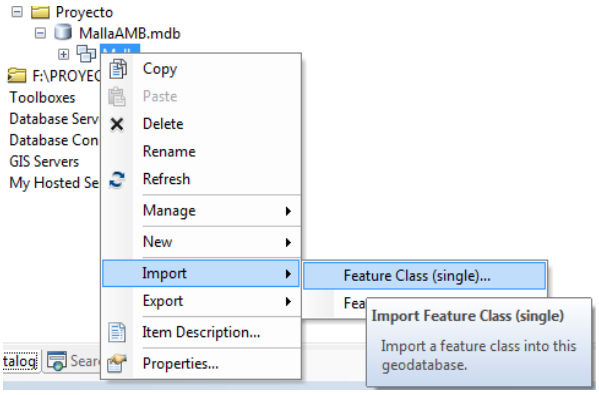


Luego de crear la Personal Geodatabase se creara el Feature Dataset, este contendrá el Feature de Giros y Malla y el Network Dataset. Se da clic derecho sobre la *MallaAMB.mdb, New>Feature Dataset. E*n la venta emergente *New Feature Dataset* se le asigna un nombre al Feature, para este caso *Malla*, clic en *Siguiente* para pasar a la siguiente pantalla donde se asignan el sistema de coordenadas, para este caso WGS_84, clic en *Siguiente>Siguiente>Finish.*

Figura 5. Construcción Network Dataset: creación Feature Dataset.

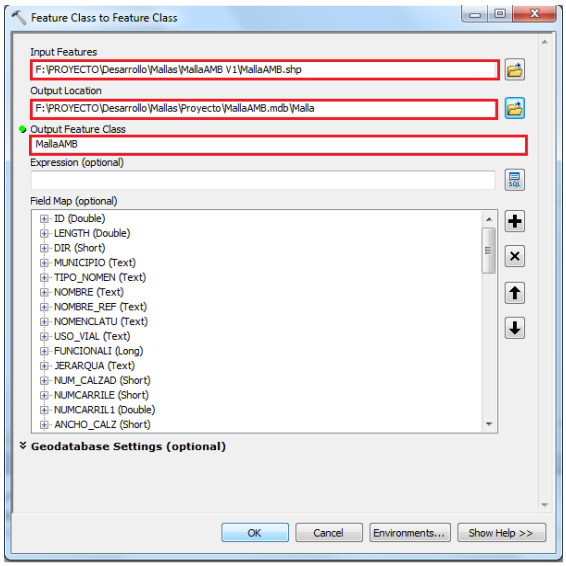
Ahora sobre el Feature Dataset *Malla* se deben agregar los Feature class de la malla vial del AMB y de giros. La malla vial de AMB fue previamente construida por el grupo de investigación Geomática y proporcionada para el proyecto como un archivo tipo shape (*MallaAMB.shp);* debido a que se está trabajando sobre una Personal Geodatabase se debe importar el archivo tipo shape como un Feature Class, esto se puede realizar dando clic derecho sobre *Malla , Import>Feature Class (single).*

Figura 6. Construcción Network Dataset: importar a una Geodatabase un shape como un Feature Class.



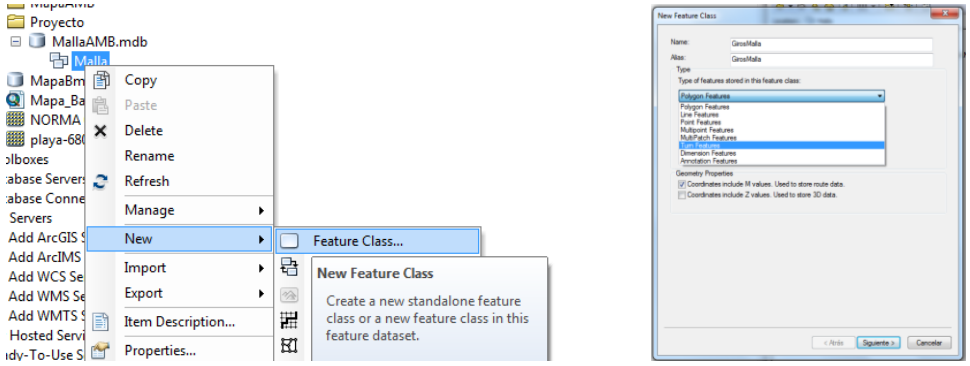
En la ventana generada *Feature Class to Feature Class* , se ingresa en *Input Features* la localización de *la MallaAMB.shp*, en *Output Location* se encontrara por default la dirección del Feauture Dataset y en *Output Feature Class* se le asigna el nombre al nuevo *Feature Class*. Finalmente se da clic en *OK*. Este Feature será de tipo Line Feautures.

Figura 7. Construcción Network Dataset: ventana Feature Class to Feature Class.



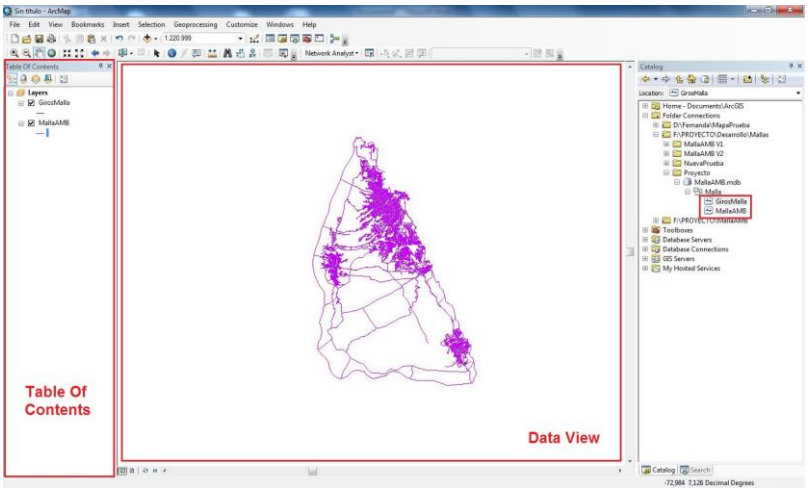
Luego de que se importe la malla vial como un Feature class; se creara el Feature class *Giros,* en la edición de los giros se editara este Feature, para ello nuevamente damos clic derecho en *Malla, New>Feature Class.* En la ventana generada *New Feature Class* se le asigna un nombre y un alias, en este caso será *GirosMalla* para ambos, el tipo será *Turn Features,* el número máximo de giros será 3, se selecciona la opción *Coordinate include M values.Used to Store Data* y por último *Siguiente > Siguiente > Finish.*

Figura 8. Construcción Network Dataset: creación Turn Feature Class.



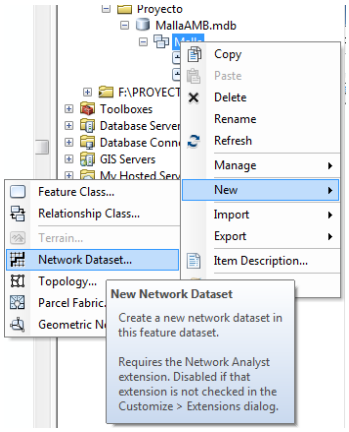
Al terminar de crear el Feature Class *MallaAMB y GirosMalla* se puede generar el Network Dataset . Primero se arrastran los dos Feature class al *Table Of Contents,* de esta manera se puede ver en el *DataView*

Figura 9. Construcción Network Dataset: layout View MallaAMB y GirosMalla.



Al tener los dos Features en *Table Of Contents,* se selección el Feature Dataset *Malla,* clic derecho *New > Network Dataset*

Figura 10. Construcción Network Dataset: crear un Network Dataset.

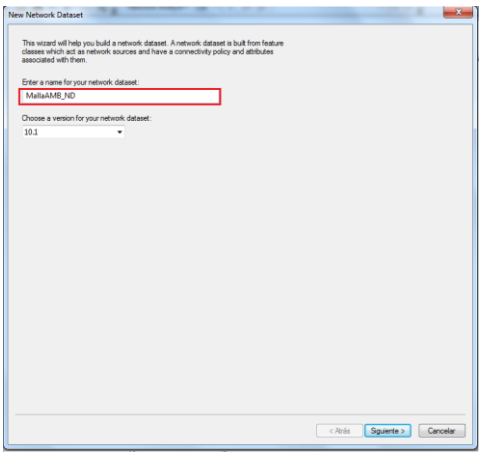


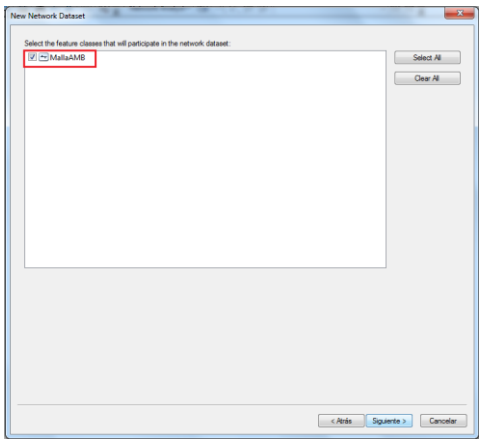
En la ventana *New Network Dataset se van a realizar los siguiente 6 pasos:*

El nombre del network va por defecto, se recomienda no modificarlo, clic en *Siguiente.*

1) Por defecto ya estará seleccionada *MallaAMB,* si existiera más de una Feauture tipo Line Features en *Table Of Contents ,* se seleccionaría sobre el cual se quiera crear en Network Dataset. Clic en *Siguiente*

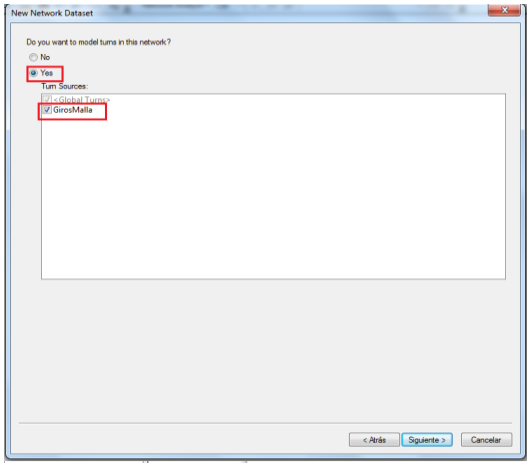
Figura 11. Construcción Network Dataset: creación Network Dataset (paso 1 y 2).





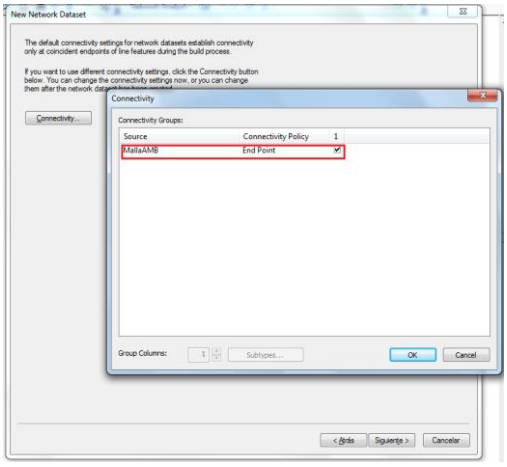
2) En la pantalla siguiente a la pregunta "*Do you want to turn in this network?"* Se selecciona *Yes*, de esta manera se agrega al Network el Feauture *GirosMalla.* Clic en *Siguiente.*

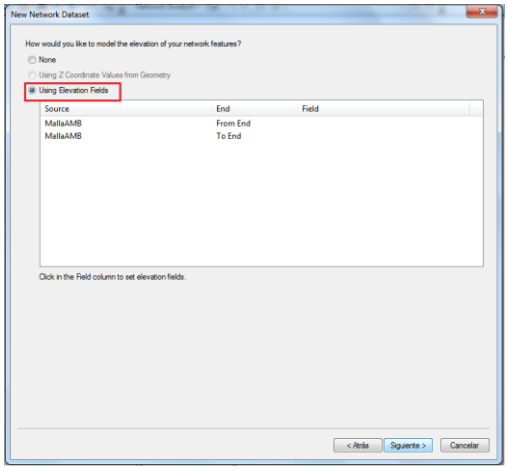
Figura 12. Construcción Network Dataset: creación Network Dataset (paso 3).



3) Clic en *Connectivity,* en la ventana emergente se verifica que la conectividad este en *End Point*. Clic en *OK>Siguiente*. A la pregunta *How would you like to model the elevation of your network features?* Se selecciona *Using Elevation Fields.* Clic en *Siguiente.*

Figura 13 Construcción Network Dataset: creación Network Dataset (paso 4).



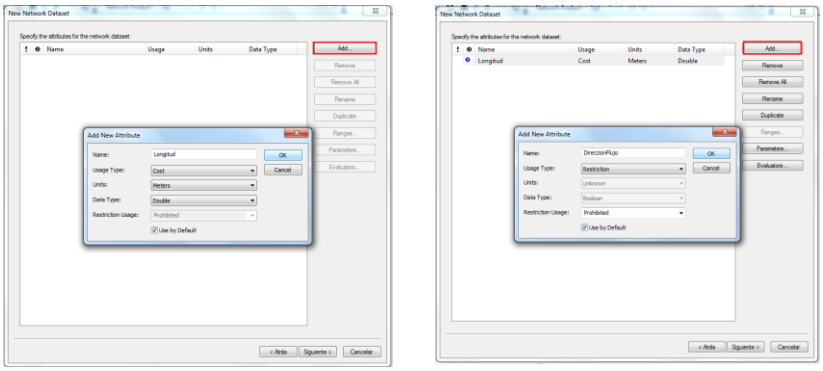


4) En la pantalla siguiente se van a agregar las dos restricciones implementadas, *Longitud* y *DireccionFlujo*. Clic en *Add*; en la ventana emergente *Add New Attribute* se eligen los atributos respectivos indicados en la Imagen

Figura 14. Construcción Network Dataset: creación Network Dataset Tabla atributos restricciones (paso 5).

	Restriccion 1	Restriccion 2
Name	Longitud	DireccionFlujo
Usage Type	Cost	Restriction
Units	Meters	*Default*
Data Type	Double	*Default*
Restriccion Usage	*Default*	Prohibited

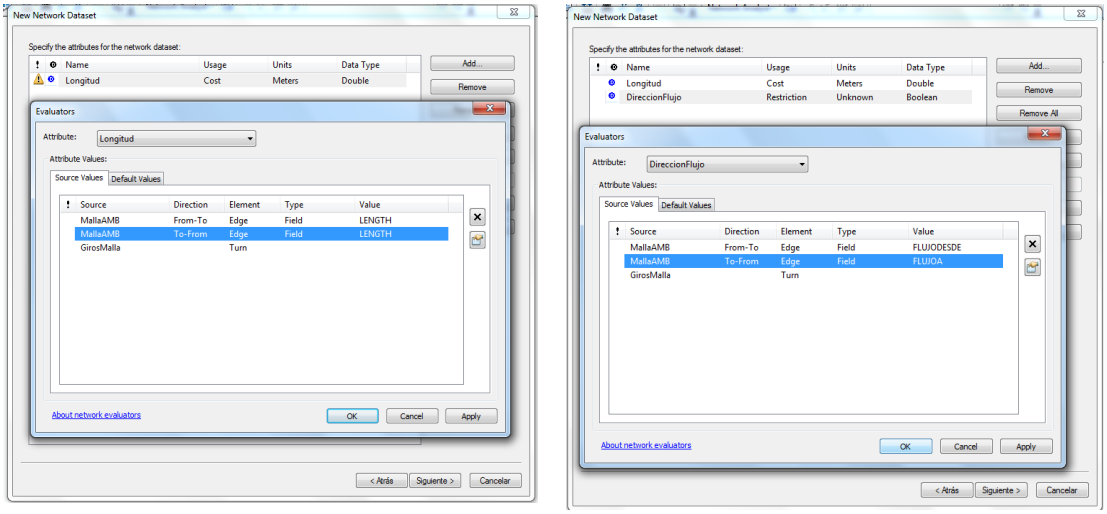
Figura 15. Construcción Network Dataset: creación Network Dataset restricciones (paso 5).



5) Luego de agregar las restricciones. Doble clic sobre la primera restricción *Longtitud, e*n la pestaña *Source Values,* de la ventana *Evaluators*, se seleccionar para las dos entradas de *MallaAMB* en *Type: Field* y en *Value: LENGTH.* Clic en *OK.*

*Doble clic sobre la segunda restricción DireccionFlujo, e*n la pestaña de *Source Values* se selecciona para las dos entradas de *MallaAMB: en Type: Field,* y dependiendo si la *Direction* es *From-To,* el *Value* será *FLUJODESDE y si es To From,* el *Value* será *FLUJOA.* Clic en *OK. Clic en Siguiente*

Figura 16. Construcción Network Dataset: creación Network Dataset (paso 6).



Al finalizar estos 6 pasos se mostrara un barra de proceso mientras se construye el Network; se responde *Sí* a la pregunta "*Do yo also want to add all Feature classes that participate in MallaAMB_ND to the map?"* Con esto se finaliza la construcción del Network y se agrega automáticamente a los *Layers.*

Figura 17. Construcción Network Dataset: creación Network Dataset barra de proceso.

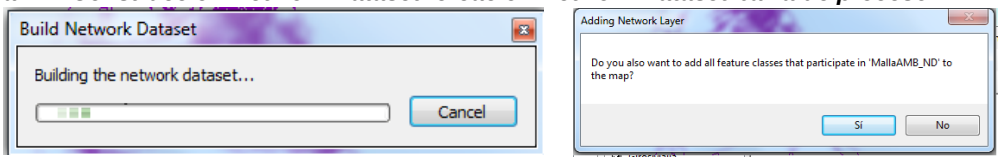
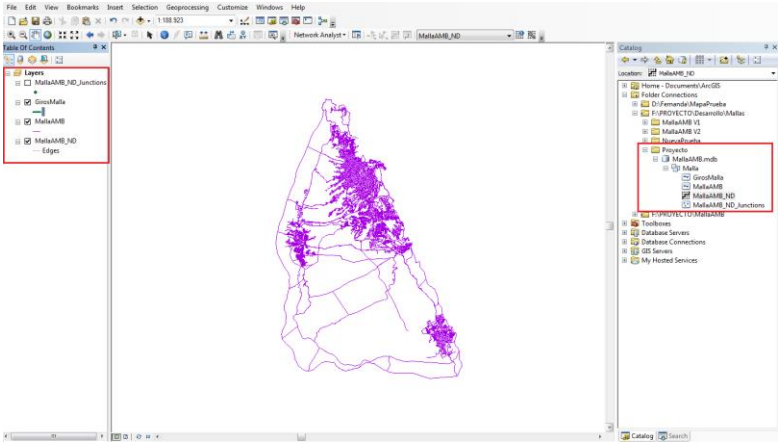


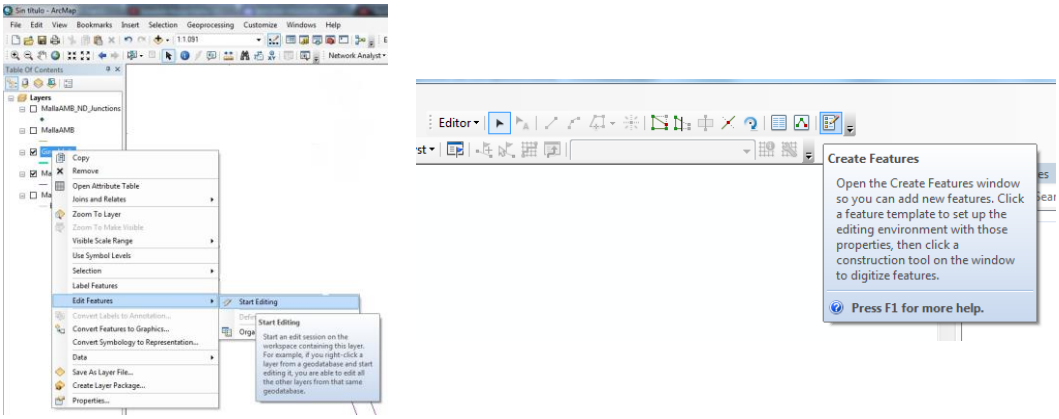
Figura 18. Construcción Network Dataset: resultado creación Network Dataset.



4.1.2 Edición Giros Network Dataset

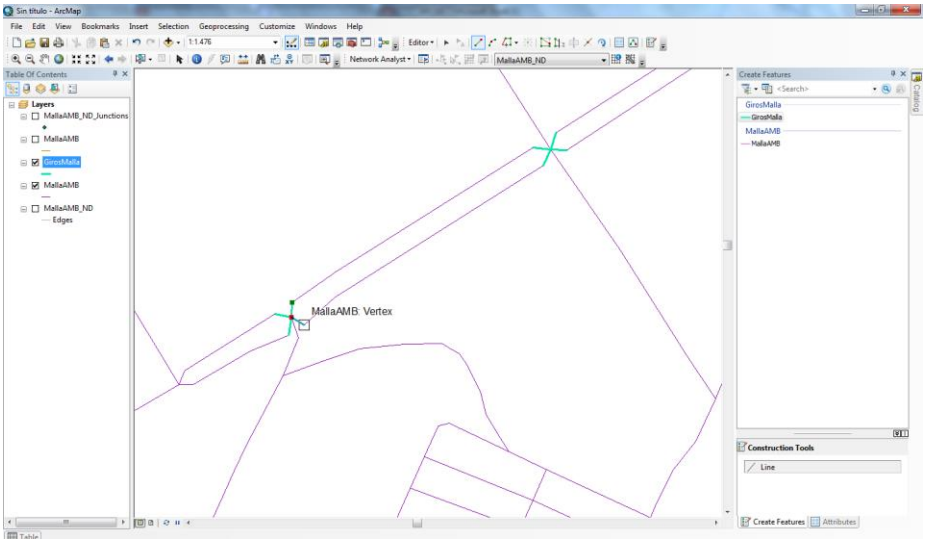
Al terminar la construcción del Network Dataset es momento de editar el Feauture *GiroMalla.* En *Table Of Contents* se deberá revisa que se encuentre solo habilitado el layer *GirosMalla* y *MallaAMB.* Sobre *GirosMalla* clic derecho, *Edit Features>Start Editing.* En la barra de herramientas del Editor, ubicada en la parte superior, clic sobre *Create Feature.*Mediante, se mostrara la pestaña *Create Features,* en la barra lateral izquierda.

Figura 19. Edición giros: Habilitar la edición de GirosMalla.



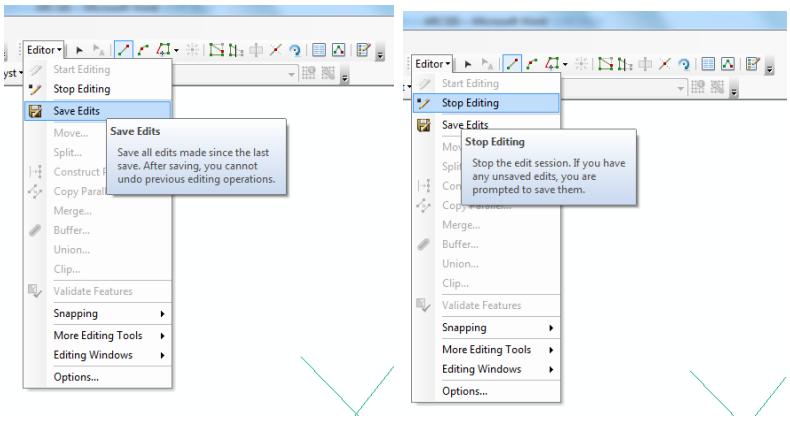
En esa pestaña clic sobre *GirosMalla,* para habilitar *Line* en el menú inferior de la pestaña *Construction Tools.* Clic en *Line.* Este paso habilitara el pincel para comenzar a trazar los giros. Se marca un punto de inicio y final del trazo sobre los vértices de *MallaAMB*, para guardar el trazo *F2,* los giros que se trazaran son los prohibidos. Se recomienda ir guardando paulatinamente, para ello en la barra de herramienta de *Editor* clic en *Editor>Save Edits,* y continuar trabajando.

Figura 20. Edición giros: Habilitar la edición de GirosMalla.



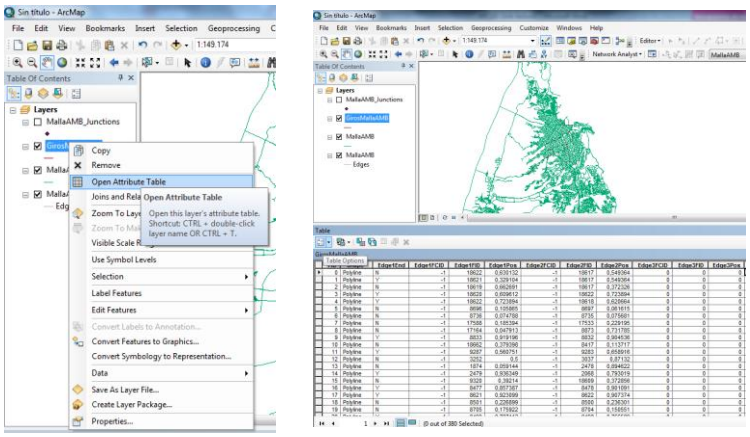
Al finalizar el trazo de todos los giros prohibidos, nuevamente en la barra de herramientas de *Editor,* clic en *Editor>Save Edits>Stop Editing.*

Figura 21. Edición giros: Finalizar la edición de GirosMalla.



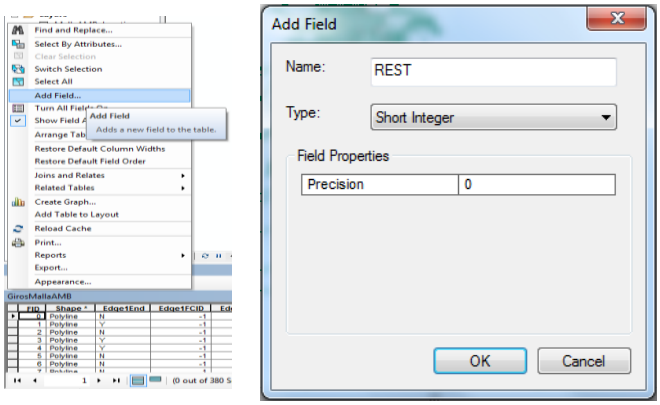
Ahora dentro de la tabla de atributos de *GirosMalla* se crerar un campo llamado REST, este se llenara con el valor de 1 representando de esta manera que el cruce prohibido. Para abrir la tabla de atributos Clic derecho sobre *GirosMalla>Open Attribute Table..*

Figura 22. Edición giros: Tabla de atributos de GirosMalla.



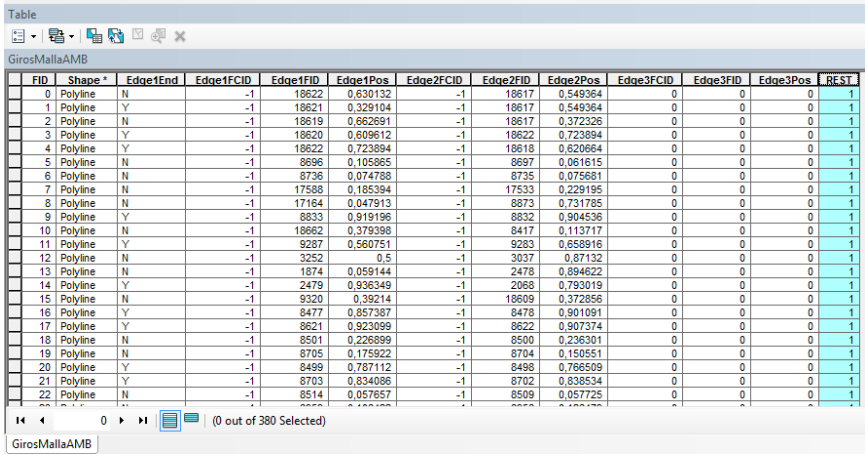
Clic en *Table Options>Add Fields.* En la ventana *Add Field* se le asigna el nombre al campo, en *Type* se selecciona *Short Integer* y finalmente *OK.*

Figura 23. Edición giros: Crear un campo en la Tabla de atributos de GirosMalla.



Para comenzar a llenar el campo *REST,* se debe nuevamente iniciar la edición del Feature *GirosMalla,* clic derecho *GirosMalla>Edit Features>Star Editing.*Al terminar de llenarlo se guarda y se para la edición, de la misma manera como antes se realizó.

Figura 24. Edición giros: Tabla de atributos final de GirosMalla.

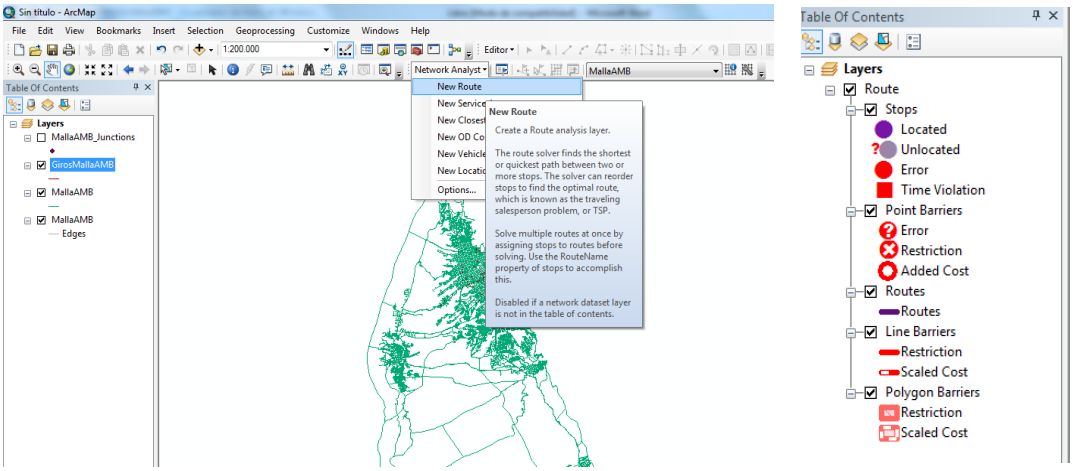


FID	Shape *	Edge1End	Edge1FCID	Edge1FID	Edge1Pos	Edge2FCID	Edge2FID	Edge2Pos	Edge3FCID	Edge3FID	Edge3Pos	REST
0	Polyline	N	-1	18622	0,630132	-1	18617	0,549364	0	0	0	1
1	Polyline	Y	-1	18621	0,329104	-1	18617	0,549364	0	0	0	1
2	Polyline	N	-1	18619	0,662691	-1	18617	0,372326	0	0	0	1
3	Polyline	Y	-1	18620	0,609612	-1	18622	0,723894	0	0	0	1
4	Polyline	Y	-1	18622	0,723894	-1	18618	0,620664	0	0	0	1
5	Polyline	N	-1	8696	0,105865	-1	8697	0,061615	0	0	0	1
6	Polyline	N	-1	8736	0,074788	-1	8735	0,075681	0	0	0	1
7	Polyline	N	-1	17588	0,185394	-1	17533	0,229195	0	0	0	1
8	Polyline	N	-1	17164	0,047913	-1	8873	0,731785	0	0	0	1
9	Polyline	Y	-1	8833	0,919196	-1	8832	0,904536	0	0	0	1
10	Polyline	N	-1	18662	0,379398	-1	8417	0,113717	0	0	0	1
11	Polyline	Y	-1	9287	0,560751	-1	9283	0,658916	0	0	0	1
12	Polyline	N	-1	3252	0,5	-1	3037	0,87132	0	0	0	1
13	Polyline	N	-1	1874	0,059144	-1	2478	0,894622	0	0	0	1
14	Polyline	Y	-1	2479	0,936349	-1	2068	0,793019	0	0	0	1
15	Polyline	N	-1	9320	0,39214	-1	18609	0,372856	0	0	0	1
16	Polyline	Y	-1	8477	0,857387	-1	8478	0,901091	0	0	0	1
17	Polyline	Y	-1	8621	0,923099	-1	8622	0,907374	0	0	0	1
18	Polyline	N	-1	8501	0,226899	-1	8500	0,236301	0	0	0	1
19	Polyline	N	-1	8705	0,175922	-1	8704	0,150551	0	0	0	1
20	Polyline	Y	-1	8499	0,787112	-1	8498	0,766509	0	0	0	1
21	Polyline	Y	-1	8703	0,834086	-1	8702	0,838534	0	0	0	1
22	Polyline	N	-1	8514	0,057657	-1	8509	0,057725	0	0	0	1

4.1.3 Publicación Network Dataset

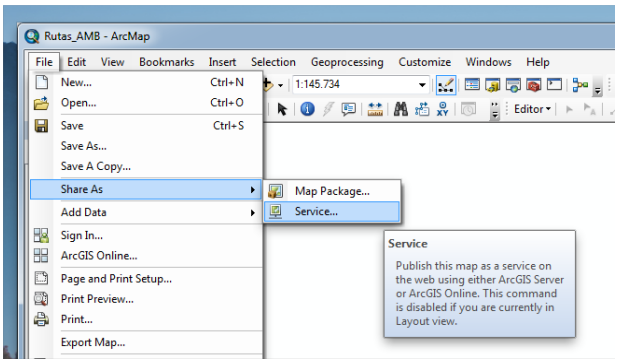
El Network Dataset está listo para publicado en el servidor. Primero en la barra de herramienta *Network Analyst,* clic en *Network Analyst>NewRoute.* En *Table of Contents* se eliminan el resto de layer, de manera que solo quede *Route.*

Figura 25. Publicación Network Dataset: Crear una nueva ruta.



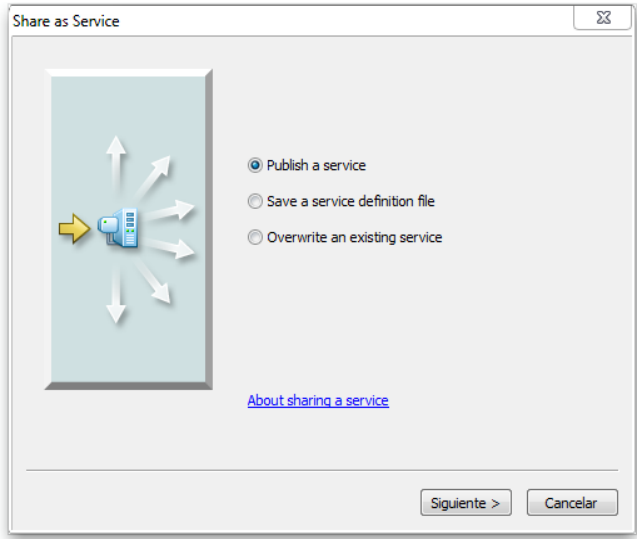
Para comenzar la publicación del Network clic en *File>Share as>Service,* ubicado en el menú principal superior.

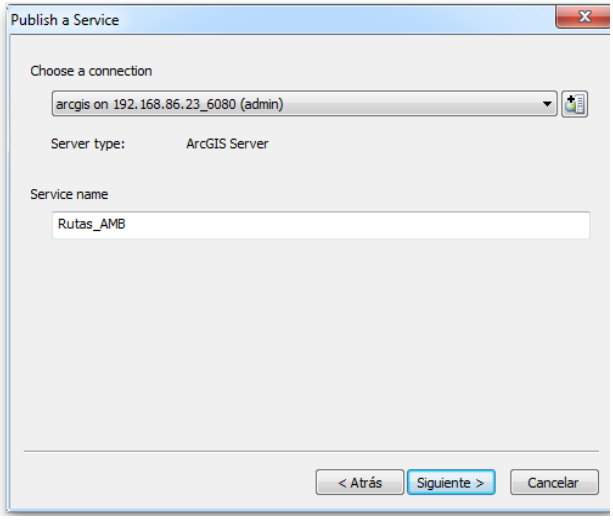
Figura 26. Publicación Network Dataset: Share as Service.



En la ventana emergente *Share as Service,* se selecciona *Publish a service,* clic en *Siguiente.* Ahora en la ventana de *Publish a Service* se escoge el tipo de conexión, en caso *arcgis on 192.168.86.23_6080 (admin),* y se le asigna un nombre al servico*, Rutas_AMB.* Clic en *Siguiente.*

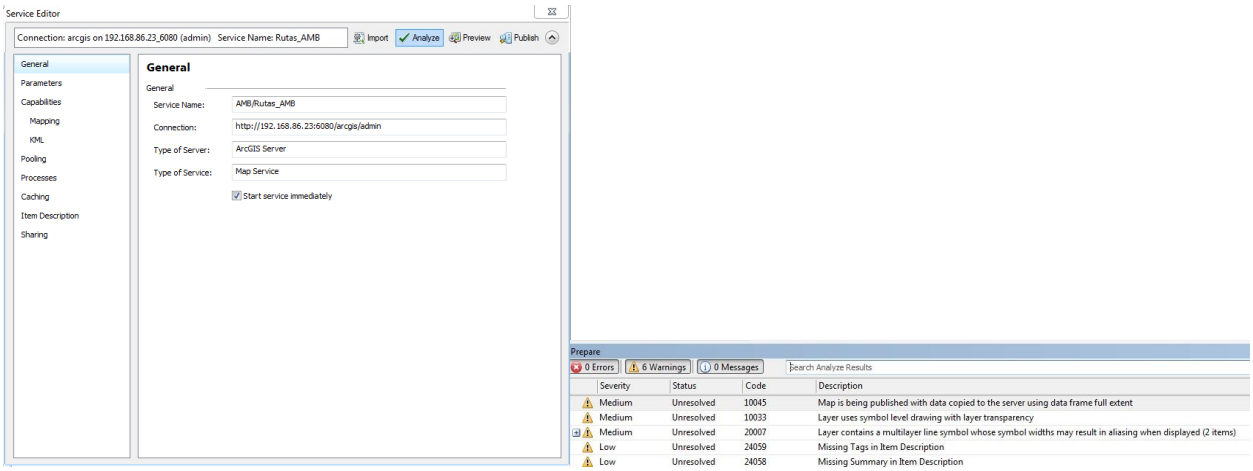
Figura 27. Publicación Network Dataset: Share as Service.





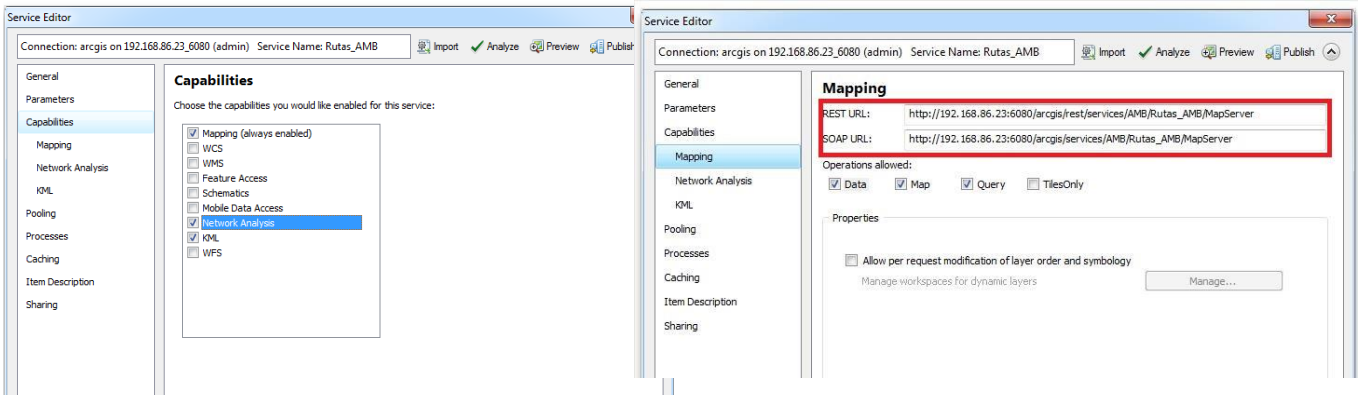
En la ventana *Service Editor,* se muestran los datos generales tales como el nombre del servicio, la conexión, el tipo de servidor y servicio. Primero se debe analizar que no existan errores en el network, clic en *Analyse,* esto mostrara en la barra inferior los errores , advertencias y mensajes.

Figura 28. Publicación Network Dataset: Service Editor Analyse.



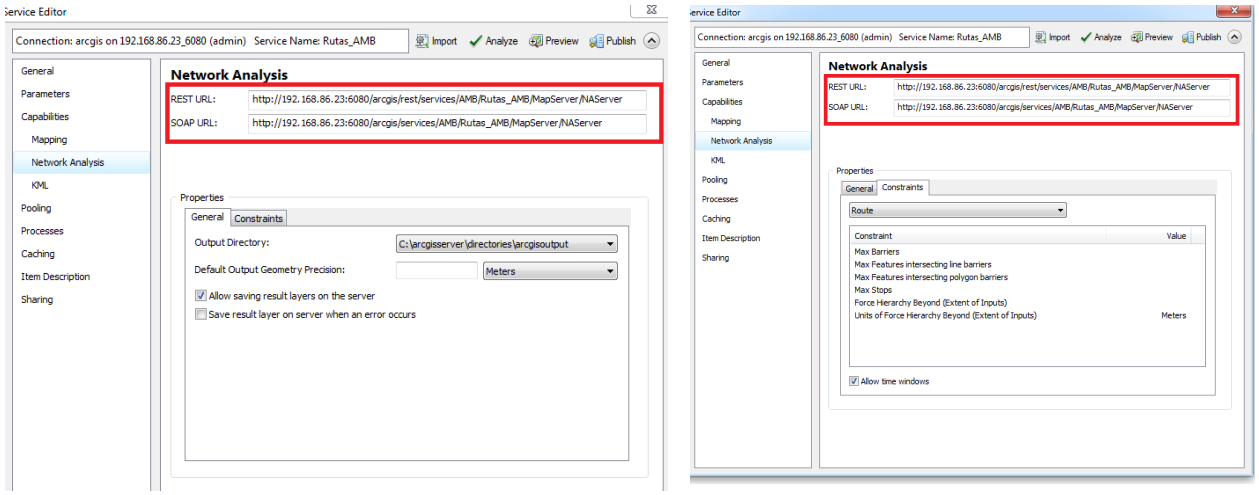
Se selecciona en el menú *Capabilities,* para escogerlas siguientes capacidades*: Mapping(always enabled), Network Analysis y KML.* Ahora se selecciona *Mapping*, esta mostrara la REST URL y la SOAP URL del MapServer.

Figura 29. Publicación Network Dataset: Service Editor Capabilities y Mapping.



Clic en la pestaña *Network Analysis* , esta mostrara la REST URL y la SOAP URL del NAServer además se debe verificar las propiedades generales, tales como la presicion geométrica en metros y que se encuentre seleccionado *Allow saving result layers on the server,* y las propiedades de las restricciones , la cual debe tener seleccionada *Route.*

Figura 30. Publicación Network Dataset: Service Editor Network Analysis.



Finalmente clic en *Publish, esto tomara unos minutos. Al terminar se verifica que este ya publicado el servicio dirigiéndose a las URL de REST del Mapserver y NAServer.*

Figura 31. Publicación Network Dataset: Service Editor Publish.

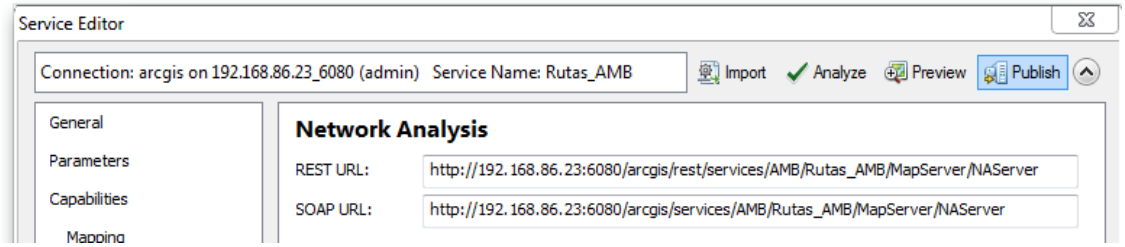
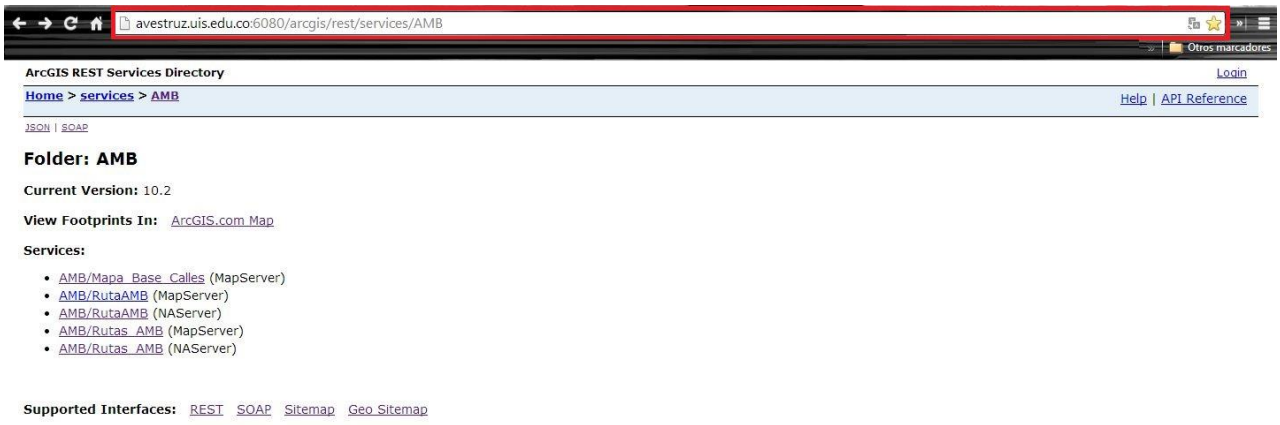


Figura 32. Publicación Network Dataset



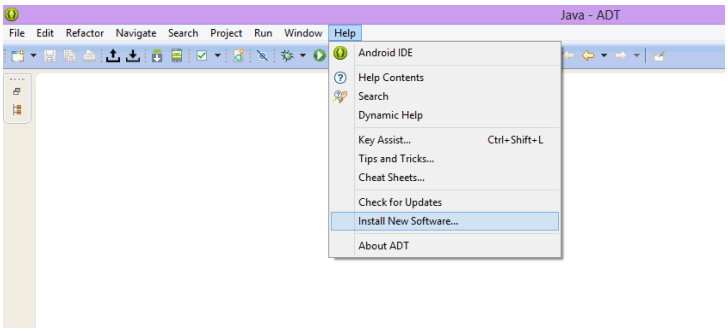
4.2 CÓMO INTEGRAR ARCGIS Y GOOGLE MAPS CON ANDROID

4.2.1 Integración de ArcGIS con Android

Para poder integrar y hacer uso de los servicios ofrecidos por ArcGIS en su SDK for Android, es necesario seguir una serie de pasos con el objetivo de obtener un correcto funcionamiento de la aplicación.

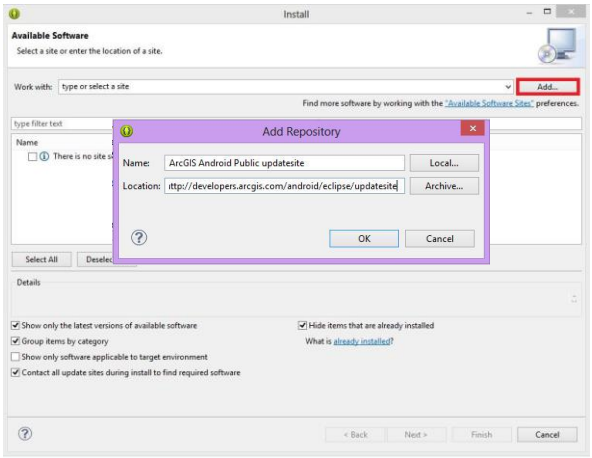
1) Abrir correctamente Eclipse, en el menú *Help>Install New Software.*

Figura 33. Integración ArcGIS con Android: Install new software



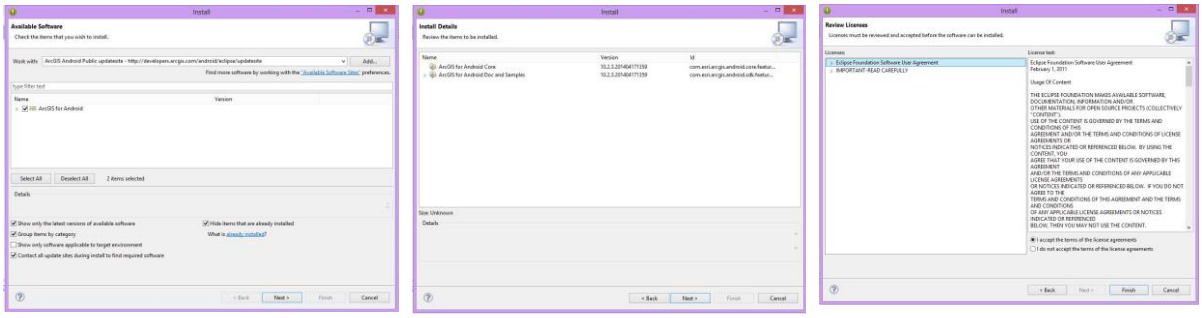
2) En la nueva ventana *Install New Software clic en Add,* se abrirá una la venta de *Add Repository,* sea asigna como nombre "ArcGIS Android Public updatesite" y en localización http://developers.arcgis.com/android/eclipse/updatesite. Clic en *OK.*

Figura 34. Integración ArcGIS con Android: Add Repository



3) Se selecciona el paquete *ArcGIS for Android* , clic en *Next>Next* , se aceptan todos los términos y clic en *Finish.*

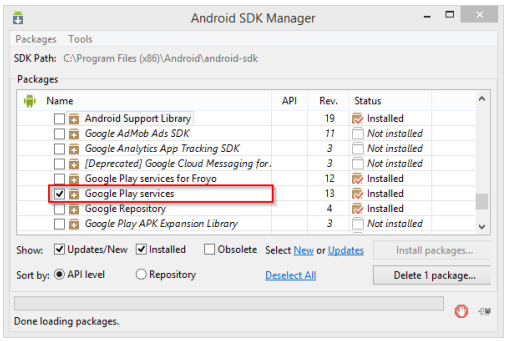
Figura 35. Integración ArcGIS con Android: Install



4.2.2 Integración de Google Maps con Android

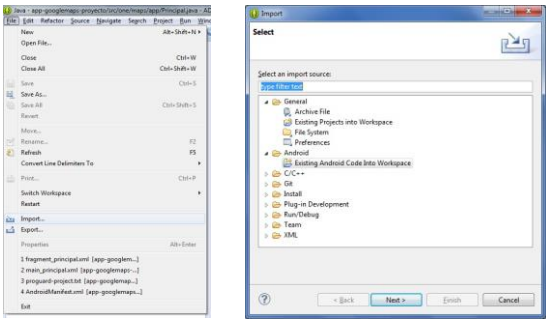
Para poder integrar y hacer uso de los servicios ofrecidos por Google en su API de Google Maps, es necesario seguir una serie de pasos con el objetivo de obtener un correcto funcionamiento de la aplicación. La API de Google Maps, en su última actualización, se incluyó dentro de los llamados Google Play Services los cuales funcionan como una suite de APIs. Cuando queremos hacer uso de cualquiera de las APIs incluidas en los Google Play Services lo primero que tendremos que hacer será importar en Eclipse el proyecto de librería donde se implementa. Todas las APIs se pueden descargar mediante el SDK Manager de Android, accediendo a la sección *Extras* y marcando el paquete llamado Google Play Services.

Figura 36. Integracion Google Maps con Android: Instalación Google Play services



Una vez instalado el paquete *Google Play Services*, se procede a importarlo en Eclipse usando la opción de menú *File>Import.* Y se selecciona la carpeta *Android>Existing Android Code Into Workspace.*

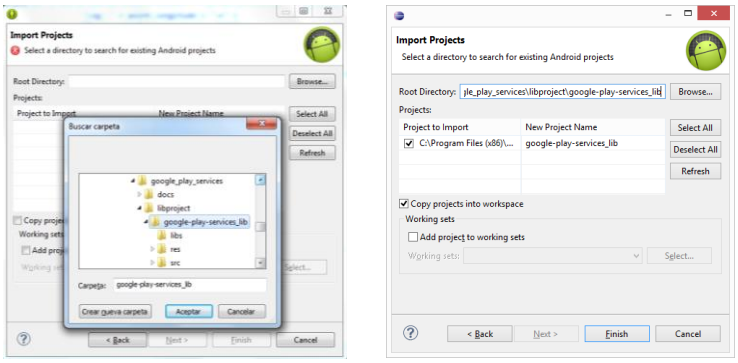
Figura 37. Integración Google Maps con Android: Importar Google Play services.



Damos clic en la opción *Next* y se accede a la sección de importación de proyectos, se pulsa sobre la opción *Browse*, buscando la ubicación del SDK de Android para posteriormente seguir la siguiente ruta: extras\google\google_play_services\libproject\google-play-services_lib.

Seleccionamos el proyecto y damos clic en aceptar. Posteriormente observamos que el proyecto aparece en la lista de *Projects to Import*, nos aseguramos que quede marcado en esta lista y también marcamos la opción *Copy projects into workspace*, pulsamos *Finish* y el proyecto quedará importado en el explorador de paquetes de Eclipse.

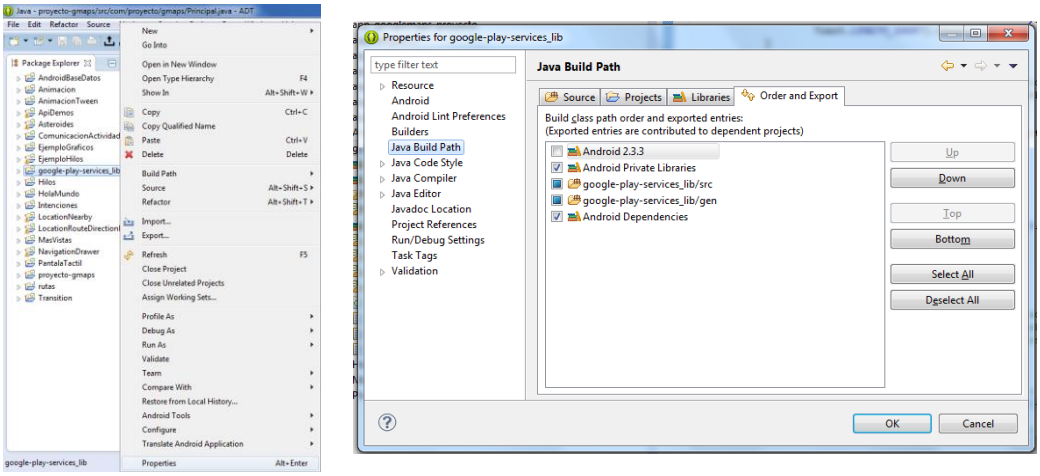
Figura 38. Integración Google Maps con Android: Import Projects



Como paso final, accedemos a las propiedades el proyecto importado *google-play-services_lib* dando clic derecho sobre el mismo y seleccionando *Properties.*

Seleccionamos *Java Build Path* en la columna izquierda, seleccionamos la pestaña *Order and Export* y observamos que la opción *Android Private Libraries* este marcada, si no lo está, la selccionamos y damos clic en *OK*. Con esto ya tenemos integrada la API de Google Maps para Android, lista para poder realizar cualquier aplicación dentro de Eclipse.

Figura 39. Integración Google Maps con Android: Import Projects



4.3 DESARROLLO EN ANDROID

4.3.1 Desarrollo Aplicación basada en los servicios ArcGIS

Después de realizar la Integración de ArcGIS con Android, se puede comenzar el desarrollo en Eclipse.

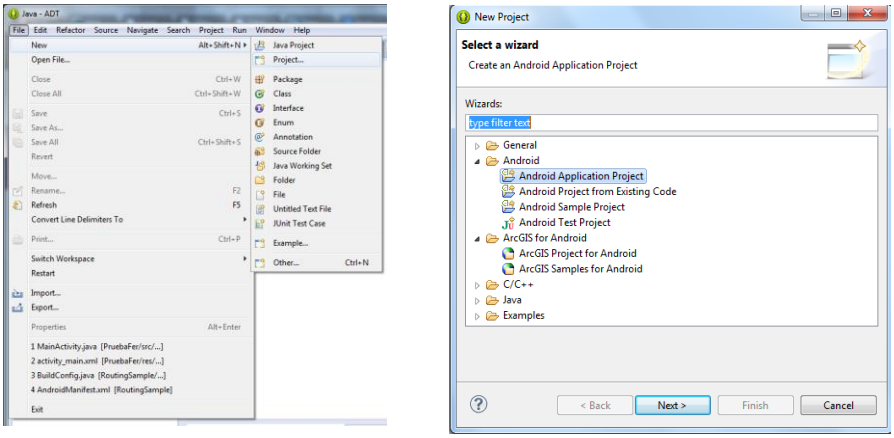
La aplicación tendrá por nombre ArcGISMap donde el usuario podrá visualizar el mapa del área metropolitana, proporcionado por el grupo de investigación Geomática, hacer uso del servicio de geolocalización y hallar rutas entre dos

puntos mediante el uso de los servicios de ArcGIS. Para realizar el desarrollo de todas las funciones anteriormente nombradas, se llevaran a cabo los siguientes pasos:

4.3.1.1 Crear un proyecto: ArcGISMap.

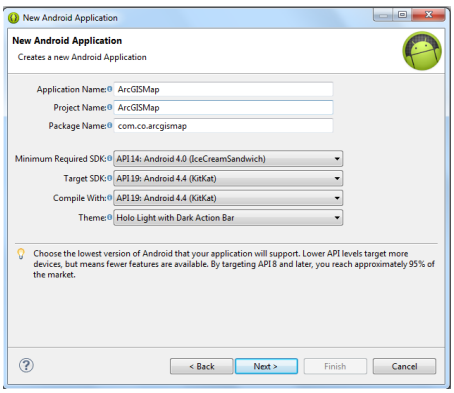
Para crear un nuevo proyecto en Eclipse, clic en *File>New>Project.* En la nueva ventana emergente *New Project* se selecciona *Android>Android Application Project* y clic en *Siguiente.*

Figura 40. Desarrollo aplicación basada en ArcGIS: crear un nuevo proyecto.



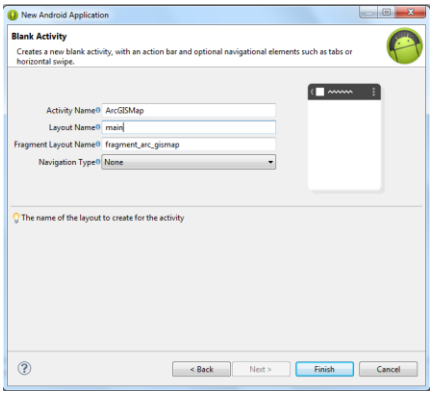
Esto abrirá una nueva ventana emergente, *New Android Application,*en ella se especificaran los detalles del proyecto con los siguientes valores:

Figura 41. Desarrollo aplicación basada en ArcGIS: new Android Application valores



Pulsar *Next* para pasar a la siguiente pantalla, en ella se dejan los valores por defecto y nuevamente *Next.* En la siguiente pantalla se configura el icono del proyecto, si se desea se puede seleccionar un icono si no se puede dejar el por defecto, clic en *Next.* En la siguiente pantalla se deja por defecto el valor de *BlankActivity. C*lic en *Next* para pasar a la última, donde se indica el nombre de la actividad y el layout, se deja el nombre de fragment por defecto. Clic en *Finish* para terminar

Figura 42. Desarrollo aplicación basada en ArcGIS: new Android Application BlankActivity



En la barra lateral izquierda se encontrara la carpeta del proyecto, se debe eliminar el fichero *fragment_arc_gismap.xml,* situado en *ArcGISMap>res>layout.* Luego dentro del fichero *ArcGISMmap.java,* situado en *ArcGISMap>src>com.co.arcgismap,* se dejara solo el fragmento de código que se muestra a continuación.

Figura 43. Desarrollo aplicación basada en ArcGIS: nuevo código ArcGISMap.java

ArcGISMap.java

```
1  package com.co.arcgismap12;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class ArcGISMap extends Activity {
7
8      @Override
9      protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12
13     }
14 }
```

Para terminar en el *AndroidManifest.xml* se deberan agregar una serie de permiso que permitira: el accesoa internet (INTERNET), conocer el estado de la red (ACCESS_NETWORK_STATE), acceder al almacenamiento externo del dispositivo para la caché de mapas (WRITE_EXTERNAL_STORAGE) y lectura de archivos de bajo nivel (READ_LOGS). Dentro del *AndroidManifest.xml* en la pestaña de *Permissions,* clic en *Add* y se selcciona la opccion de *Uses Permissions* en la ventana emergente. En la combobox de name se busca cada el nombre del permiso. Este paso se realiza para cada uno de los permisos.

Figura 44. Desarrollo aplicación basada en ArcGIS: agregar un permiso en el AndroidManifest.

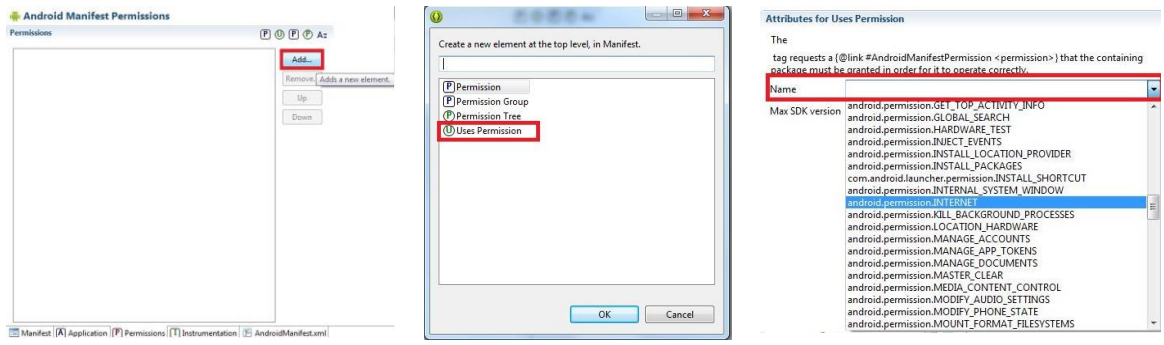
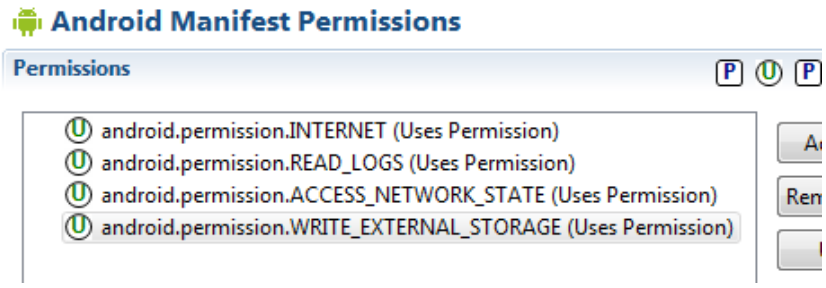


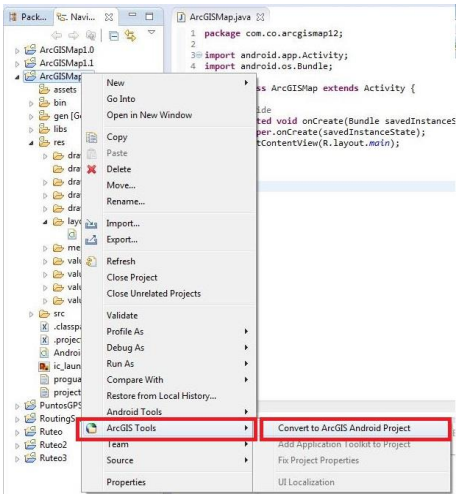
Figura 45. Desarrollo aplicación basada en ArcGIS: permisos en el AndroidManifest.xml.



4.3.1.2 Convertir el proyecto en un Android ArcGIS Project.

Luego de crear el nuevo proyecto ArcGISMap, se debe convertir el proyecto en uno Android ArcGIS Project para hacer uso del SDK. Clic derecho en carpeta del proyecto *ArcGIS Tools>Convert to Android ArcGIS Project.*

Figura 46. Desarrollo aplicación basada en ArcGIS: convertir a Android ArcGIS Project.



- **Crear la vista del mapa**

Cuando la aplicacionse abra en ella se debera visualizar el mapa del área metropolitana de Bucaramanga, para ellos en el fichero *main.xml,* situado en *ArcGISMap>res>layout,* se copiara el siguiente texto, el cual especifica el tipo de layout, Linear Layout, y se agrega un *MapView,* donde se mostrara el mapa identificado con el id *map*, *y un TextView*, identificado con el id *mostrar.*

Figura 47. Desarrollo aplicación basada en ArcGIS: código main.xml

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <com.esri.android.map.MapView
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        initExtent="-19332033.11, -3516.27, -1720941.80, 11737211.28" >
    </com.esri.android.map.MapView>

    <TextView
        android:id="@+id/mostrar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

</LinearLayout>
```

MapView

TextView

Dentro del fichero *ArcGisMap.java* se crearan l tres variable globales, tal cual como se muestra en el código.

Nota: Para importar las librerias automaticamente *Control+Shif+O.*

Figura 48. Desarrollo aplicación basada en ArcGIS: código variables vista mapa ArcGISMap.java.

```
import android.app.Activity;
import android.os.Bundle;

import com.esri.android.map.GraphicsLayer;
import com.esri.android.map.MapView;
import com.esri.android.map.ags.ArcGISTiledMapServiceLayer;

public class ArcGISMap extends Activity {

    MapView map = null;
    ArcGISTiledMapServiceLayer tileLayer;
    GraphicsLayer graphicLayer;
```

- map: es una variable tipo *MapView, esta clase permite contener un* mapa.
- *tiledLayer:* es una variable tipo *ArcGISTiledMapServiceLayer, esta clase* propia de ArcGIS permite trabajar con el recurso de servicio de mapas en caché mediante un URL.
- *graphicLayer:* es una variable tipo *GraphicsLayer, esta clase* propia de ArcGIS representa un layer que contendra uno o más gráficos , como puntos o lineas.

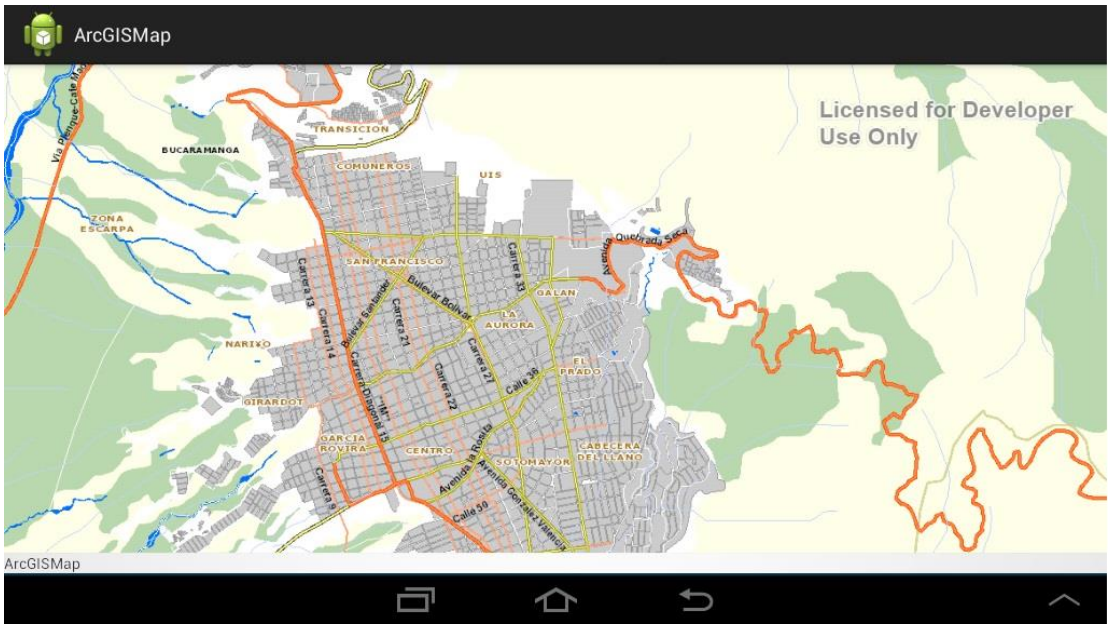
Dentro de la funcion *onCreate* se agregara el siguinete código. En el se inicializan las variables anteriormente creadas: *map* se le asigna el id del *MapView; tiledLayer* se le asigna la URL correspondiente al mapa del AMB *http://avestruz.uis.edu.co/arcgis102/rest/services/AMB/Mapa_Base_Calles/MapServer y graphicLayer* se inializara en modo de renderizacion dinamico, agregue las librería . Tanto como el *tiledLayer* y *graphicLayer* se agregan a la vista map, haciendo posible su visualizacion.Ademas se crearan dos funciones *onPause()* y *onResume().*

Al finalizar se ejecuta la aplicación en un dispositivo virtual o por medio de depuración USB en un dispositivo físico para comprobar la vista del mapa, es importante tener acceso a internet o Wi-Fi.

Figura 49. Desarrollo aplicación basada en ArcGIS: código de la vista mapa ArcGISMap.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    map = (MapView) findViewById(R.id.map);
    tileLayer = new ArcGISTiledMapServiceLayer(
            "http://avestruz.uis.edu.co/arcgis102/rest/services/AMB/Mapa_Base_Calles/MapServer");
    graphicLayer= new GraphicsLayer(RenderingMode.DYNAMIC);
    map.addLayer(tileLayer);
    map.addLayer(graphicLayer);
}

protected void onPause() {
    super.onPause();
    map.pause();
}

protected void onResume() {
    super.onResume();
    map.unpause();
}
```

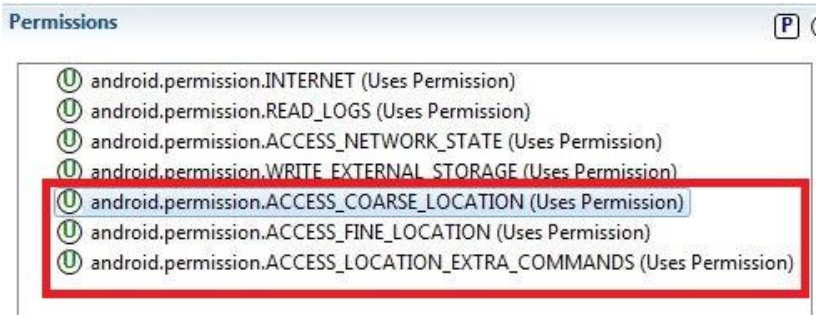
Figura 50. Desarrollo aplicación basada en ArcGIS: vista mapa en el dispositivo móvil.



4.3.1.3 Implementacion servicios de geolocalización.

Una de las funciones del aplicativo es el uso del servicio de geolocalización mediante el GPS. Primero se deben agregar en el *AndroidManifest* los permisos de acceder a una localización aproximada derivada de los recursos de red del móvil (ACCESS_COARSE_LOCATION) , acceder a la localizacion precisa dada del GPS(ACCESS_FINE_LOCATION) y acceder comandos adicionales del provedor de localizacion (ACCESS_LOCATION_EXTRA_COMMANDS). Estos permisos se agregaran de la forma como s mostro anteriormente (4.3.1.1 Crear un proyecto: ArcGISMap pág 41).

Figura 51. Desarrollo aplicación basada en ArcGIS: permisos servicios de geolocalización en AndroidManifest.xml.



En *ArcGISMap.java* se crean 4 variables locales ,*un* método y una clase y se identifica dentro del *onCreate* el *TextView* creado en el *main.xml, c*omo se muestra en el código.Recuerde importar las librerias.

Figura 52. Desarrollo aplicación basada en ArcGIS: código creación variables locales para GPS en ArcGISMap.java.

```
TextView mostrar;
private LocationListener locListener;
private LocationManager locManager;
Point mLocation = null;
PictureMarkerSymbol gps;
```

Figura 53. Desarrollo aplicación basada en ArcGIS: código asignación textView en ArcGISMap.java.

```
mostrar=(TextView)findViewById(R.id.mostrar);
mostrar.setText(" ");
```

Figura 54. Desarrollo aplicación basada en ArcGIS: código método ubicar() en ArcGISMap.java

```
public void ubicar(){
    locManager=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
    locListener=new myLocationListener();
    locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 15, 10, locListener);
    locManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 15, 10, locListener);
}
```

Figura 55. Desarrollo aplicación basada en ArcGIS: código método myLocationListener() en ArcGISMap.java

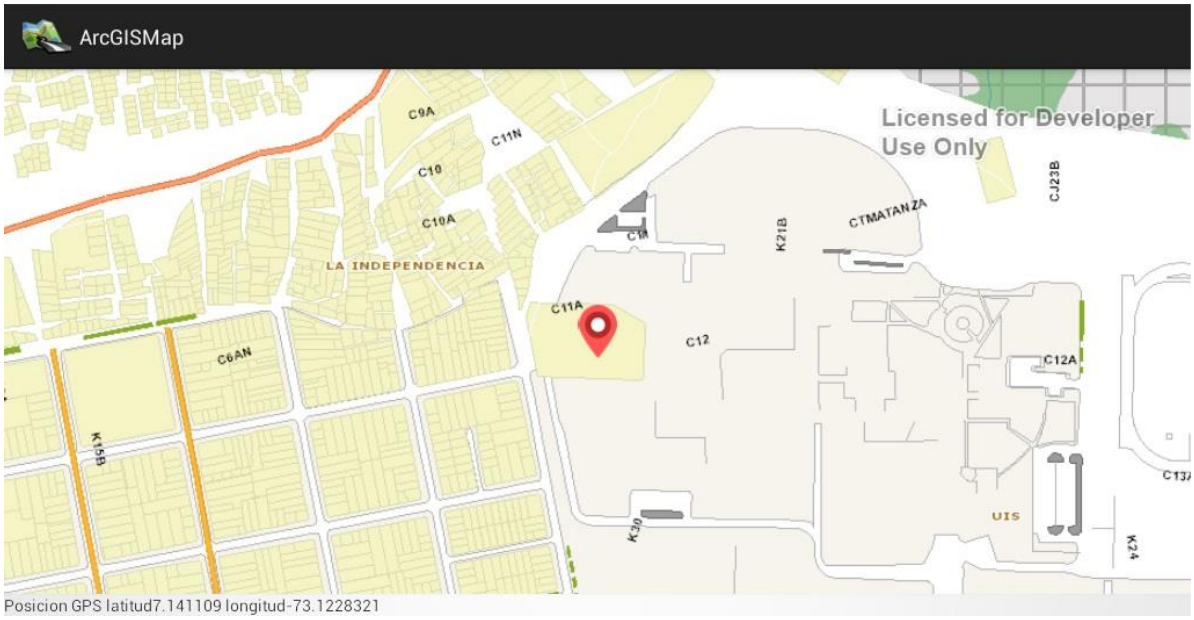
```
private class myLocationListener implements LocationListener{
    public void onLocationChanged(Location location) {
        if(location!=null){
                mLocation = new Point(location.getLongitude(),location.getLatitude());
                gps= new PictureMarkerSymbol(map.getContext(),getResources().getDrawable(R.drawable.gpsicon));
                Graphic graphic = new Graphic(mLocation, gps);
                graphicLayer.addGraphic(graphic);
                map.zoomTo(mLocation, (float) 10.0);
                String m="Posicion GPS "+"latitud"+location.getLatitude()+" longitud"+location.getLongitude();
                mostrar.setText(m); }}

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {
        String m="El GPS ha sido habilitado";
        mostrar.setText(m); }

    public void onProviderDisabled(String provider) {
        String m="El GPS ha sido deshabilitado";
        mostrar.setText(m); }
}
```

- mostrar: es una variable tipo *TextView, esta clase permite mostar texto..*
- *locationListener:* es una variable tipo *LocationListener, se usa para recibir notificaciones del LocationManager cuando la localizacion cambia.*
- *locationManager:* es una variable tipo *LocationManager,* esta clase provee acceso al sistema de servicios de localizacion.
- *mLocation:* es una variable tipo *Point,* esta clase es propia de *ArcGIS* representa un objeto con una localizacion especifica (x,y).
- *ubicar():* este método implementa el *LocationManager* asignadole los servicios de localizacion de la aplicación y usa los servicios de GPS y red para obtener la localizacion y el *LocationListener* llama a la clase *myLocationListener().*

- *myLocationListener(Location location):* esta clase tiene cuatro metodos como los es el *onLocationChanged,* el cual se escarga de darle las corrdenada al punto, graficarlo dentro del mapa, hacer zoom en el y mostrar en el textView las coordenadas, el *onStatusChanged, onProviderEnable y onProviderDisable,* estos dos ultimos informan si el GPS esta o no habilitado.

Se agrega el método *ubicar* al *onCreate*, de manera que apenas se incie la aplicación, localice la posicion del dispositivo móvil. Ejecute la aplicación, debe hacerlo en un dispositivo móvil, y compruebe su funcionamiento.Recuerde activar el GPS y tener conecxion a Wi-Fi.

Figura 56. Desarrollo aplicación basada en ArcGIS: servicio de geolocalización en el dispositivo móvil.



4.3.1.4 Implementacion servicos de ruteo de ArcGIS

La última funcion del palicativo es encontrar rutas dentro del mapa entre dos puntos, para a ellos se crearan nuevas variables locales y dos metodos tal cual como se muestra en el código. Recuerde importar las librerias.

Figura 57. Desarrollo aplicación basada en ArcGIS: código creación variables locales para ruteo en ArcGISMap.java.

```
Point  punto1 =null;
Point punto2=null;
PictureMarkerSymbol p1symbol;
PictureMarkerSymbol p2symbol;
Graphic graphic;

GraphicsLayer routeLayer;
RouteTask mRouteTask = null;
RouteParameters routeParams;
NAFeaturesAsFeature nAsFeature;
RouteResult mResults = null;
Exception mException = null;
```

Figura 58. Desarrollo aplicación basada en ArcGIS: código para ubicar puntos e inicializar RouteTask dentro el método onCreate en ArcGISMap.java

```
routeLayer = new GraphicsLayer();
map.addLayer(routeLayer);
try {
    String routeTaskURL="http://avestruz.uis.edu.co/arcgis102/rest/services/AMB/RutaAMB/NAServer/Route";
    mRouteTask = RouteTask.createOnlineRouteTask(routeTaskURL,null);
} catch (Exception e1) {
        e1.printStackTrace();
}

map.setOnSingleTapListener(new OnSingleTapListener() {
    private static final long serialVersionUID = 1L;
    public void onSingleTap(final float x, final float y) {
        if(punto1==null){
            punto1 = map.toMapPoint(x, y);
            p1symbol= new PictureMarkerSymbol(map.getContext(),getResources().getDrawable(R.drawable.location1));
            graphic = new Graphic(punto1, p1symbol);
            mensaje="Punto 1: "+"x= " + punto1.getX()+"y:" + punto1.getY();
        }
        else{
            if(punto2==null){
            punto2 = map.toMapPoint(x, y);
            p2symbol= new PictureMarkerSymbol(map.getContext(),getResources().getDrawable(R.drawable.location12));
            graphic = new Graphic(punto2, p2symbol);
            mensaje="Punto 2: "+"x= " + punto2.getX()+"y:" + punto2.getY();
            }
            else{
                mensaje="No puede marcar mas punto\nlimpie el mapa si desea hacer nuevos puntos";
            }
        }
        graphicLayer.addGraphic(graphic);
        mostrar.setText(mensaje);
    }});
```

Figura 59. Desarrollo aplicación basada en ArcGIS: código método ruteo() en ArcGISMap.java

```
public void ruteo(){
    Thread t = new Thread() {
    @Override
    public void run() {
        try {
            routeParams = mRouteTask.retrieveDefaultRouteTaskParameters();
            routeParams.setImpedanceAttributeName("Longitud");
            nAsFeature=new NAFeaturesAsFeature();
            StopGraphic startPnt =new StopGraphic(mLocation);
            StopGraphic endPnt=new StopGraphic(punto1);
            nAsFeature.setFeatures(new Graphic[]{startPnt,endPnt});
            routeParams.setStops(nAsFeature);
            mResults=mRouteTask.solve(routeParams);
            List<Route> routes=mResults.getRoutes();
            Route mRoute=routes.get(0);
            final List<RouteDirection> directions = mRoute.getRoutingDirections();
            Double suma=(double) 0;
            for(int i = 0; i < directions.size(); i++) {
               RouteDirection direction = directions.get(i);
               suma= suma + direction.getLength();
            }
            Geometry routeGeom=mRoute.getRouteGraphic().getGeometry();
            Graphic symbolGraphic=new Graphic(routeGeom, new SimpleLineSymbol(Color.BLUE, 3));
            graphicLayer.addGraphic(symbolGraphic);
            distancia=suma*1.609344;
            String fsuma=formato.format(suma);
            mensaje="Distancia: "+fsuma+"Km";
        } catch (Exception e) {
            mException = e;
        }
    }
};
t.start();
}
```

- *punto1y punto2 :*son variable tipo *Point,* donde se almaceran los puntos marcados en el mapa.

- *p1symbol y p2symbol*: son variable tipo *PictureMarkerSymbol,* esta clase permite dibujar puntos con una imagen.

- *routeLayer: es una variable tipo GraphicsLayer*, en ella se guardara el trazo de la ruta

- *mRouteTask:* es una variable tipo *RouteTask* , esta clase se usa para implementar las funciones de ruteo tanto online como offline

- *routeParams: es una variable tipo RouteParameters, esta clase permite especificar los parámetros para realizar el ruteo.*

- *nAsFeature*: es una variable tipo *NAFeaturesAsFeature*, esta clase contiene las paradas, es decir los puntos entre los cuales se realiza el ruteo.

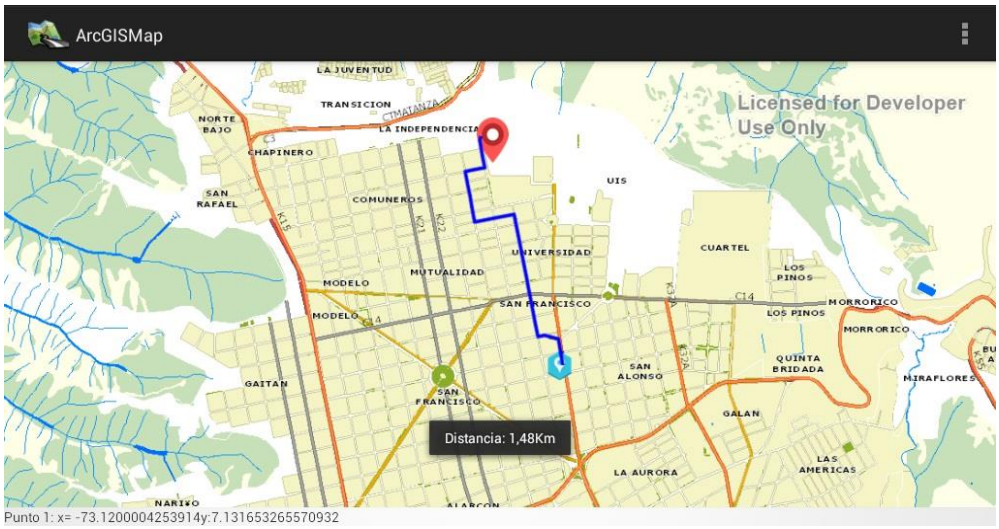
- *mResults:* es una variable tipo *RouteResult,* esta clase contiene la información sobre la ruta halla en la solución del ruteo.

- Dentro del *onCreate* se inicializa el layer donde se mostrara la ruta y el *RouteTask* con la URL del Network Dataset publicado (numeral 4.1.3 Publicación Network Dataset). Además se crea un evento de *setOnSingleTapListener* mediante el cual cuando el usuario realice un tap sobre el mapa se genere un punto con coordenadas geográfica, siendo este el punto de inicio o destino de la ruta.

- *Ruteo():* esta método se encaragara de resolver la ruta entre dos puntos,para ello incializara *routeParams* con el atributo de longitud, establcera la paradas con los puntos, y mediante el metodo *solve* encontrara la ruta entre los puntos. Ademas este método proporcionara la ditancia completa de la ruta y la guardara en el *routeLayer* para ser graficada en el mapa.

Para el desarrollo de la aplicación se implemento el metodo de ruteo() en un menú de opciones, pero tambien se puedo mediante un boton. Luego de implementar el método ejecute la aplicación, debe hacerlo en un dispositivo móvil, y compruebe su funcionamiento.Recuerde activar el GPS y tener conecxion a Wi-Fi.

Figura 60. Desarrollo aplicación basada en ArcGIS: servicio de geolocalización en el dispositivo móvil.



4.3.1.3 Ubicación sitios de interes

Para localización de los distintos sitios de interés, se basó en la información recopilada por Google Maps, donde se identifican las coordenadas de cada sitio .Mediante un *ListView* el usuario seleccionara el tipo de sitio de interés que desea visualizar y este se mostrar en el mapa.

Para cada tipo de sitio de interés se crea un *GraphicLayer* y un método. El siguiente código muestra la creación de cada *GraphicLayer,* como variables locales, y como se inicializan y se agregan a la vista del mapa en el *onCreate.*

Figura 61 Ubicación sitios de interés: declara e iniciar GraphicLayer

```
GraphicsLayer restaurantes;
GraphicsLayer cajeros;
GraphicsLayer bancos;
GraphicsLayer cine;
GraphicsLayer parques;
GraphicsLayer policia;
GraphicsLayer hospitales;
GraphicsLayer bomberos;
GraphicsLayer iglesias;
```

Figura 62. Ubicación sitios de interés: declara e iniciar GraphicLayer

```
restaurantes= new GraphicsLayer(RenderingMode.DYNAMIC);
cajeros= new GraphicsLayer(RenderingMode.DYNAMIC);
bancos= new GraphicsLayer(RenderingMode.DYNAMIC);
cine= new GraphicsLayer(RenderingMode.DYNAMIC);
parques= new GraphicsLayer(RenderingMode.DYNAMIC);
policia= new GraphicsLayer(RenderingMode.DYNAMIC);
hospitales= new GraphicsLayer(RenderingMode.DYNAMIC);
bomberos= new GraphicsLayer(RenderingMode.DYNAMIC);
iglesias= new GraphicsLayer(RenderingMode.DYNAMIC);
map.addLayer(restaurantes);
map.addLayer(cajeros);
map.addLayer(bancos);
map.addLayer(cine);
map.addLayer(parques);
map.addLayer(policia);
map.addLayer(hospitales);
map.addLayer(bomberos);
map.addLayer(iglesias);
```

El método de cada tipo de sitio de interés, contendrá los puntos que se irán agregando al *GraphicLayer* correspondiente. Estos métodos serán llamados según la selección del usuario, utilizando la comunicación entre actividades. El siguiente código muestra el método implementado para el sitio de interés “Cinemas”.

Figura 63. Ubicación sitios de interés: código cines().

```
public void cines(){

    PictureMarkerSymbol cines= new PictureMarkerSymbol(map.getContext(),getResources().getDrawable(R.drawable.cine));
    Point p1=new Point (-73.1078975 , 7.1283431);
    graphic = new Graphic(p1, cines );
    cine.addGraphic(graphic);
    Point p2=new Point (-73.1120174 , 7.1274489);
    graphic = new Graphic(p2, cines );
    cine.addGraphic(graphic);
    Point p3=new Point (-73.1094362 , 7.1155683);
    graphic = new Graphic(p3, cines );
    cine.addGraphic(graphic);
    Point p4=new Point (-73.1067312 , 7.0994376);
    graphic = new Graphic(p4, cines );
    cine.addGraphic(graphic);
    Point p5=new Point (-73.1047965 , 7.0702857);
    graphic = new Graphic(p5, cines );
    cine.addGraphic(graphic);

    mensaje="Cinemas";
    mostrar.setText(mensaje);

}
```

Para poder implementar la comunicación entre actividades, se creara una nueva actividad, llamada *sitiosInteres* y esta tendrá un layer asociado con un *spinner* y un *button.*

Figura 64. Ubicar sitios de interés: código sitiosInteres.java

```
public class sitiosInteres extends Activity {
    Spinner sitios;
    Button botonUbicar;
    String[] tipoSitio=null;
    ArrayAdapter<String> adaptador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sitiosinteres);

        botonUbicar = (Button) findViewById(R.id.ubicar);
        sitios = (Spinner) findViewById(R.id.spinner);
        tipoSitio = getResources().getStringArray(R.array.lugar_tipo_nombre);
        adaptador = new ArrayAdapter<String>(getBaseContext(), android.R.layout.simple_spinner_dropdown_item, tipoSitio);
        adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        sitios.setAdapter(adaptador);
        botonUbicar.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {

                int p = sitios.getSelectedItemPosition();
                String tipo = tipoSitio[p];

                Intent regresar = new Intent();
                regresar.putExtra("respuesta", tipo);
                setResult(RESULT_OK, regresar);
                finish();
            }
        });
    }
}
```

Para finalizar dentro de la actividad principal se crea la funcion que lanzara la intencion y la que recibira la respuesta.

Figura 65. Ubicación sitios de interés: código comunicación entre actividades ArcGISMap.java

```
public void sitiosInteres(){

    Intent i = new Intent(this, sitiosInteres.class);
    startActivityForResult(i, 1234);
}

protected void onActivityResult(int requestcode, int resultcode, Intent regreso){
    if (requestcode==1234 && resultcode==RESULT_OK){
        String respuesta = regreso.getExtras().getString("respuesta");
        Toast.makeText(getBaseContext(),respuesta, Toast.LENGTH_SHORT).show();
        switch(respuesta){
        case "Bancos":
            bancos();
            break;
        case "Estacion de Bomberos":
            bomberos();
            break;
        case "Cajeros":
            cajeros();
            break;
        case "Cinemas":
            cines();
            break;
        case "Iglesias":
            iglesias();
            break;
        case "Hospitales":
            hospitales();
            break;
        case "Parques":
            parques();
            break;
        case "Estaciones de Policia":
            policia();
            break;
        case "Restaurantes":
            restaurantes();
            break;
        }
    }
}
```

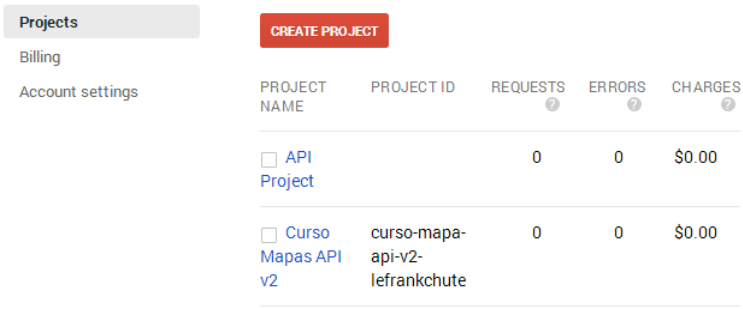
4.3.2 Desarrollo aplicación basada en los servicios Google Maps

4.3.2.1 Obtención de la API Key

Después de haber logrado la integración de Google Maps con Android, se procede a llevar a cabo el desarrollo de la aplicación.

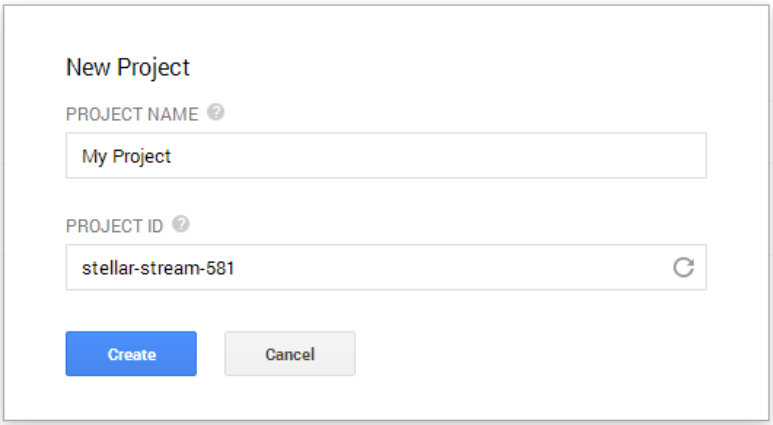
El siguiente paso será obtener una API Key para poder utilizar el servicio de mapas de Google en la aplicación. Para esto se accede a la consola de APIs de Google: *https://console.developers.google.com/project*. Posterior a esto, se debe crear un nuevo proyecto dando clic al botón *CREATE PROJECT*

Figura 66. Desarrollo aplicación basada en los servicios Google Maps: Creación proyecto para API Key



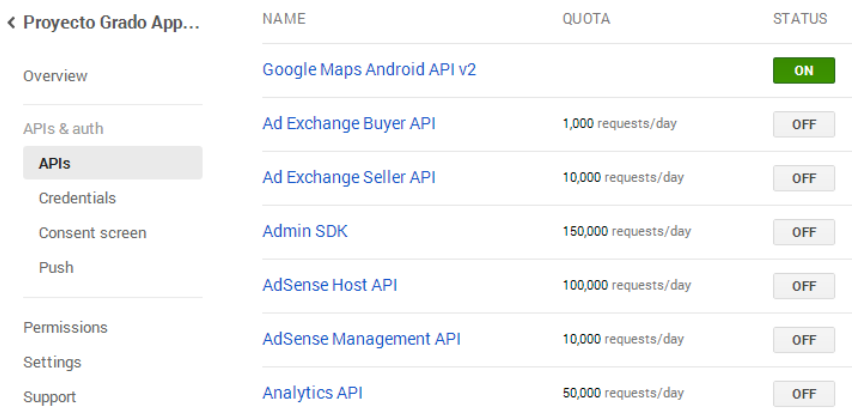
Posteriormente se abrirá una ventana que solicitará el nombre del proyecto y el ID del mismo. Se coloca un nombre descriptivo y un ID único y se pulsa *Create*.

Figura 67. Desarrollo aplicación basada en los servicios Google Maps: Nombre proyecto para API Key



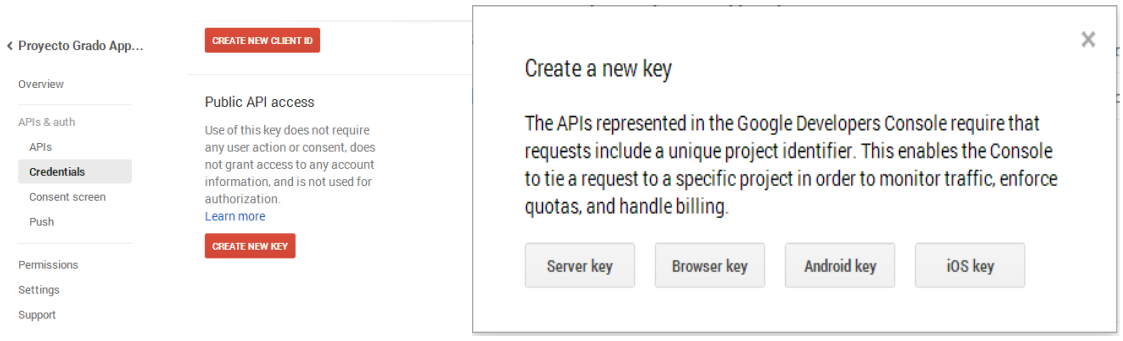
Una vez creado el proyecto, hay que dirigirse a *APIs & Auth* y se selecciona la opción *APIs*. Desde esta ventana se puede activar o desactivar cada una de las *APIs* de Google que se quieran utilizar. Para el caso del proyecto, se empieza por activar la opción llamada *Google Maps Android API v2*. Si hay otras opciones activadas, se desactivan, dejando solo activa la opción anteriormente señalada.

Figura 68. Desarrollo aplicación basada en los servicios Google Maps: Activación Google Maps Android API V2



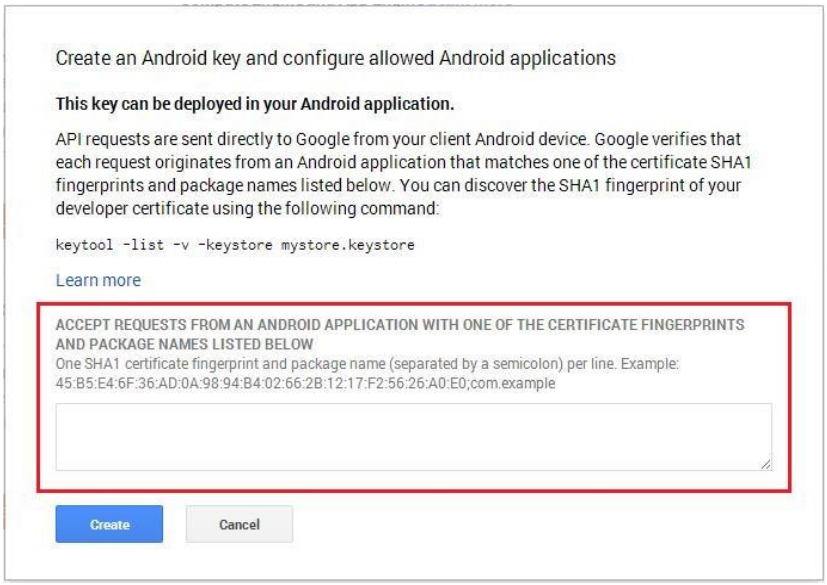
Posteriormente, dirigiendose al submenu *Credentials* ubicado debajo de *APIs & Auth*. Accediendo a este menú, se tendrá la oportunidad de crear la nueva *API Key*, que permita utilizar el servicio de mapas desde la aplicación en particular. Dentro del menú aparecera un subtitulo llamado *Public API access* y debajo del mismo un botón llamado *CREATE NEW KEY*, dando clic en ese botón se dará inicio a la creación de la *API Key*. Después de dar clic en *CREATE NEW KEY* aparece otra ventana.

Figura 69. Desarrollo aplicación basada en los servicios Google Maps: Creación de nueva API Key



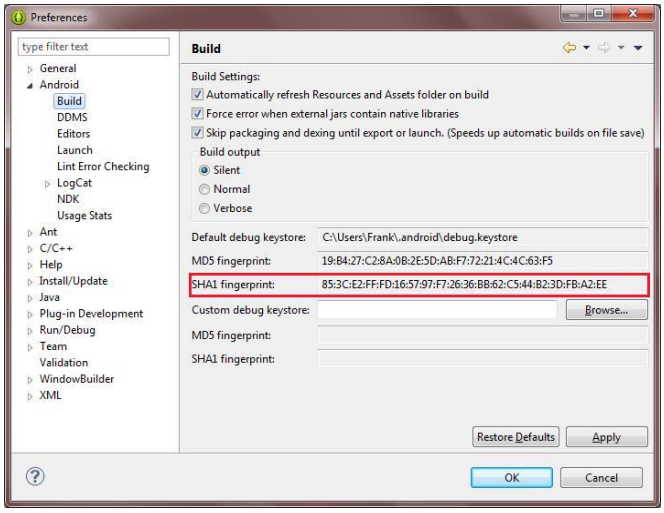
En esta ventana se da clic en *Android Key*, posteriormente aparecerá otra ventana en la que se especificará la huella digital SHA1 del certificado con el que se firma la aplicación y el nombre del paquete java utilizado en la aplicación.

Figura 70. Desarrollo aplicación basada en los servicios Google Maps: Registro huella digital SHA1



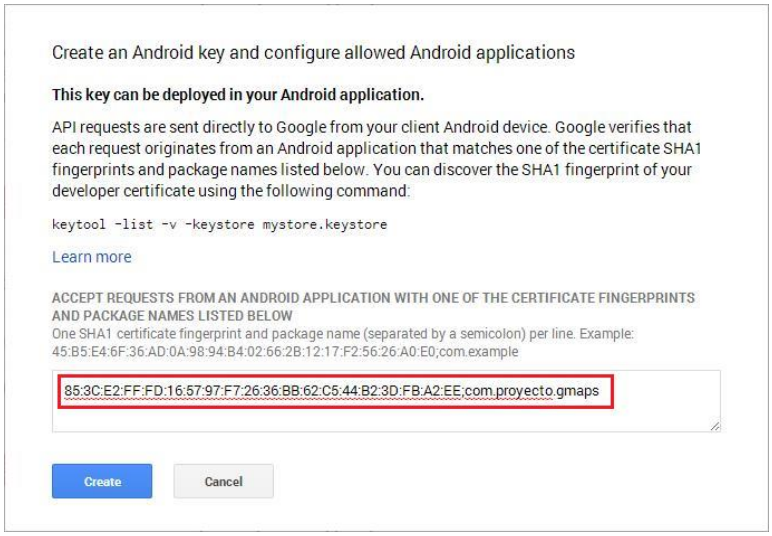
La huella digital SHA1 es una firma digital. Todas las aplicaciones en Android deben ir firmadas para poder ejecutarse tanto en un dispositivo físico como emulado. Este proceso de firma es uno de los pasos que se tienen que hacer siempre antes de distribuir públicamente una aplicación. Para consultar la huella SHA1, en Eclipse se accede al menú *Window* > *Preferences* > *Android* > *Build.* En la ventana que aparece se puede observar la huella SHA1.

Figura 71. Desarrollo aplicación basada en los servicios Google Maps: Ubicación huella digital SHA1 en Eclipse



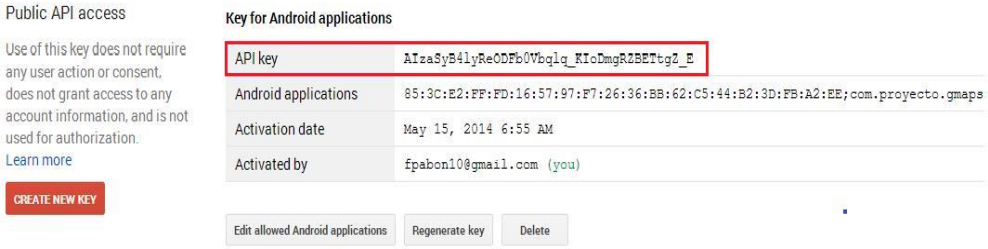
Se copia la clave, se agrega en la ventana donde se va a crear la clave y se coloca punto y coma, acto seguido se escribe el nombre del paquete java de aplicación. El nombre del paquete será: *com.proyecto.gmaps*.

Figura 72. Desarrollo aplicación basada en los servicios Google Maps: Huella digital SHA1 + nombre paquete



Se da clic en *CREATE* e inmediatamente se realizará la creación de la *API Key* para el desarrollo de la aplicación.

Figura 73. Desarrollo aplicación basada en los servicios Google Maps: API Key



4.3.2.2 Creación del proyecto Android de la aplicación

Como paso siguiente se inicia la creación de la aplicación Android del proyecto. Para esto, dentro de Eclipse hay que remitirse a *File* > *New* > *Project* > *Android* > *Android Application Project*.

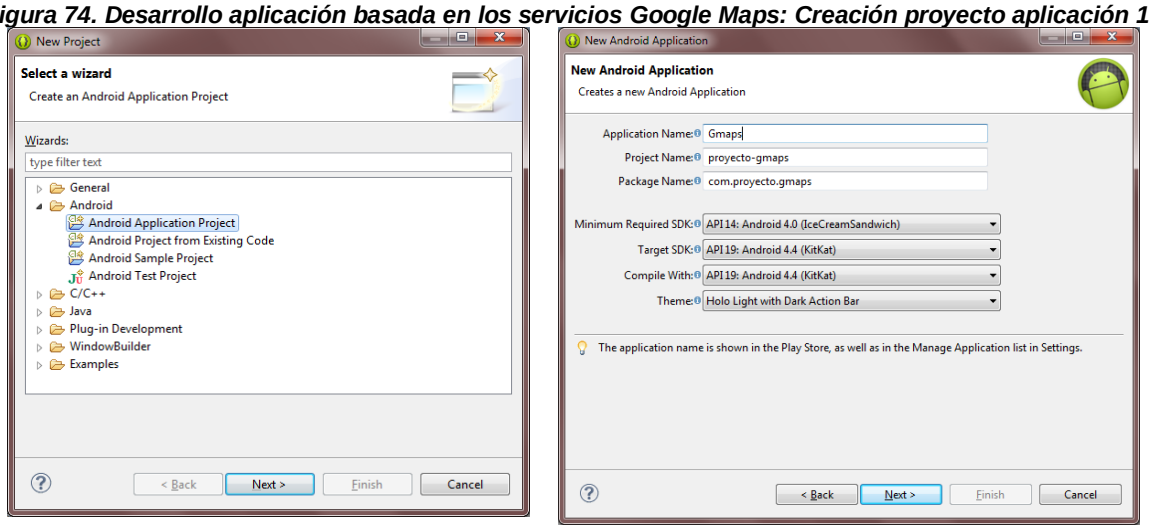
Se da clic en *Next >* y se define el nombre de la aplicación, el nombre del proyecto y el nombre del paquete. En la misma ventana se modifica el valor del mínimo SDK requerido, que especificara la versión mínima de Android a la que la aplicación funcionará. Para el proyecto, la versión mínima se estableció en 4.0. El nombre del paquete DEBE ser el mismo que fue definido durante la obtención de la API Key: *com.proyecto.gmaps*.

Application Name: *Gmaps*

Project Name: *proyecto-gmaps*

Package Name: *com.proyecto.gmaps*

Figura 74. Desarrollo aplicación basada en los servicios Google Maps: Creación proyecto aplicación 1



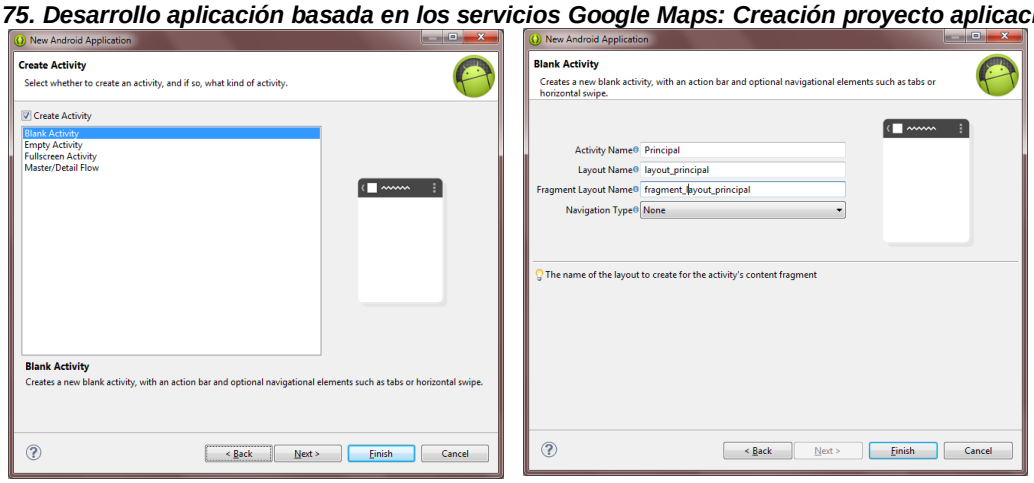
Se da clic en *Next >* hasta que se llegue a la ventana titulada *Create Activity* y se selecciona *Blank Activity*. Se da clic en *Next >* y se define el nombre de la actividad principal, el nombre de su layout asociado, el nombre de su fragment layout asociado.

Activity Name: *Principal*

Layout Name: *layout_principal*

Fragment Layout Name: *fragment_layout_principal*

Figura 75. Desarrollo aplicación basada en los servicios Google Maps: Creación proyecto aplicación 2

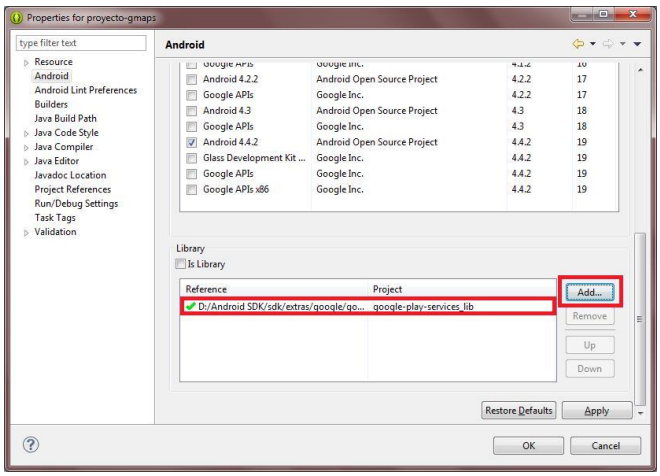


Y finalmente se da *Finish*.

4.3.2.3 Asociación de la aplicación con Google Maps API v2 para android

El siguiente paso es la asociación de la API de Google Maps para android, con la aplicación del proyecto. Para hacer esto nos hay que dirigirse a la carpeta principal del proyecto *proyecto-gmaps,* se da clic derecho y se escoge la opción *Properties*, se entra en la sección *Android*, se baja al final del módulo, se da clic sobre el botón *Add…* se selecciona la carpeta *google-play-services_lib*. Se da Ok en todas las ventanas.

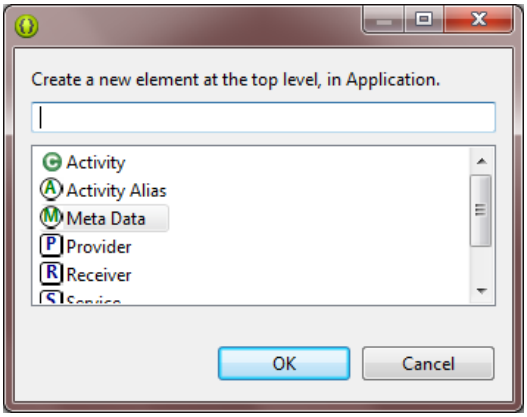
Figura 76. Desarrollo aplicación basada en los servicios Google Maps: Asociación de la aplicación y la API



A continuación, se editará el archivo *AndroidManifest.xml* de la aplicación, añadiendo una clausula *Meta-data* dentro del elemento *Application*

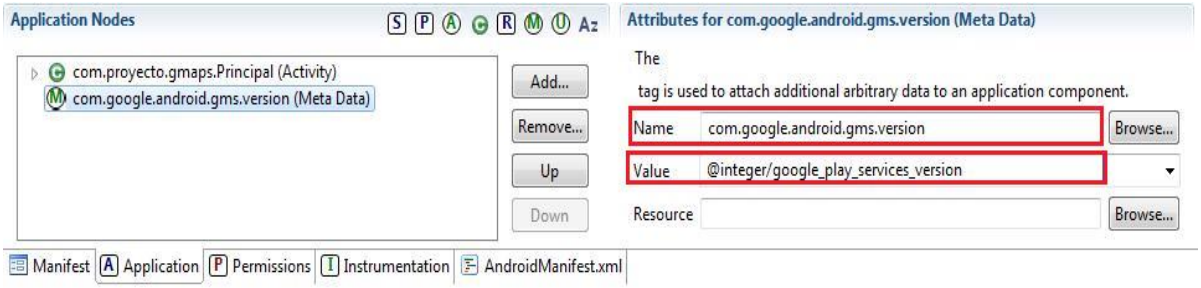
Para esto se selecciona el archivo *AndroidManifest.xml,* en la pestaña *Application*, para luego dirigirse a la parte inferior del módulo y dar clic en el botón *Add…*, donde aparecen algunas opciones a seleccionar, en la cual se esvoge la opción *Meta Data.*

Figura 77. Desarrollo aplicación basada en los servicios Google Maps: Creación de Meta Data 1



Y se digitan los siguientes valores:

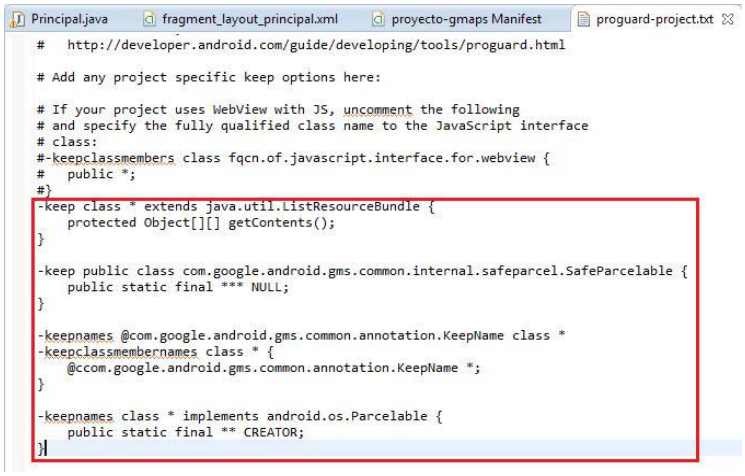
Figura 78. Desarrollo aplicación basada en los servicios Google Maps: Creación de Meta Data 2



Este *Meta-data* permite especificar dentro de la aplicación que se hará uso de la API de *Google Play Services*, la cual contiene la librería necesaria para poder hacer uso del servicio de mapas dentro de la aplicación.

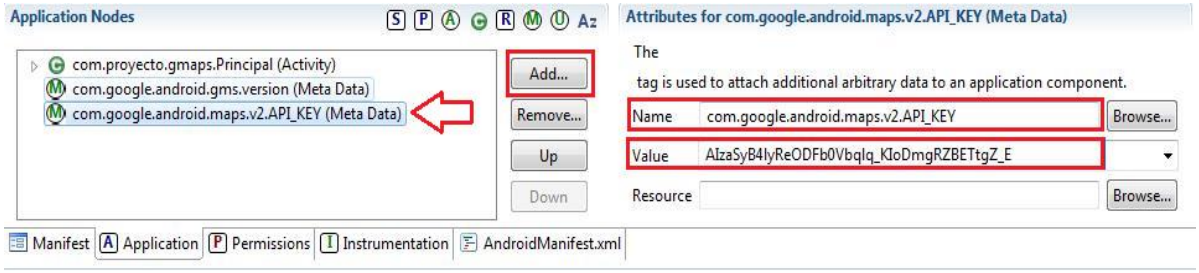
Finalmente, se añadiran al final del fichero *proguard-project.txt* las siguientes lineas para evitar que se eliminen algunas clases.

Figura 79. Desarrollo aplicación basada en los servicios Google Maps: Creación de Meta Data 3



Siguiendo con la configuración de la aplicación, se regresa al archivo *AndroidManifest.xml,* donde se hace clic de nuevo en la pestaña *Application* y se añade otro *Meta-data* con los siguientes valores:

Figura 80. . Desarrollo aplicación basada en los servicios Google Maps: Creación de Meta Data 4

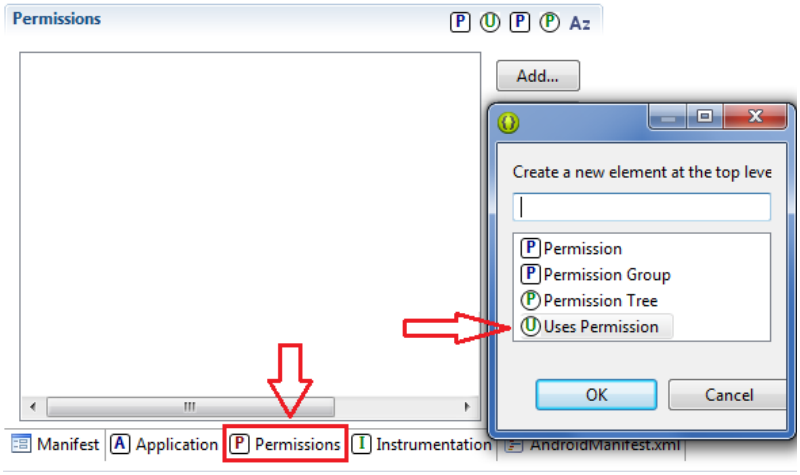


Donde el valor del campo *Value* es la API Key que se generó al principio del desarrollo de la aplicación. Siguiendo con el *AndroidManifest.xml,* también se debe inlcuir una serie de permisos que permiten acceder a internet (INTERNET), conocer el estado de la red (ACCESS_NETWORK_STATE), acceder al almacenamiento externo del dispositivo para la caché de mapas

(WRITE_EXTERNAL_STORAGE) y hacer uso de los servicios web de Google (READ_GSERVICES).

Para incluir estos permisos, nos hay que dirigirse a la pestaña *Permissions* dentro del *AndroidManifest.xml,* se da clic en el botón *Add..* y se selecciona la opción *Uses Permission*, para cada uno de los permisos anteriormente explicados se realiza esta misma acción.

Figura 81. Desarrollo aplicación basada en los servicios Google Maps: Inclusión de permisos



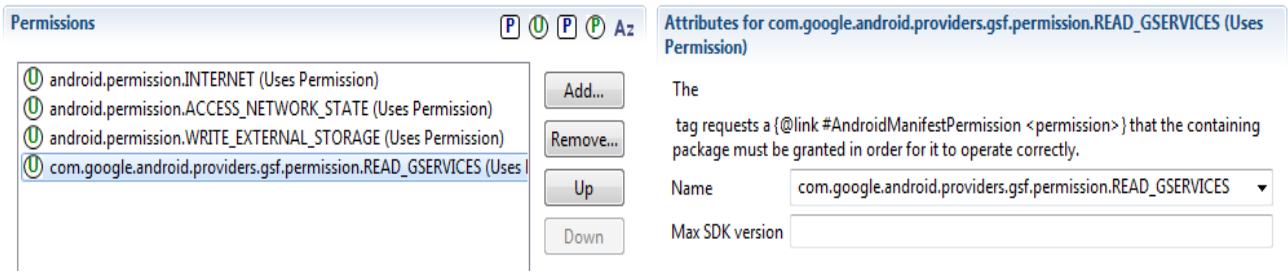
Name: android.permission.INTERNET

Name: android.permission.ACCESS_NETWORK_STATE

Name: android.permission.WRITE_EXTERNAL_STORAGE

Name: com.google.android.providers.gsf.permission.READ_GSERVICES

Figura 82. Desarrollo aplicación basada en los servicios Google Maps: Permisos de la aplicación

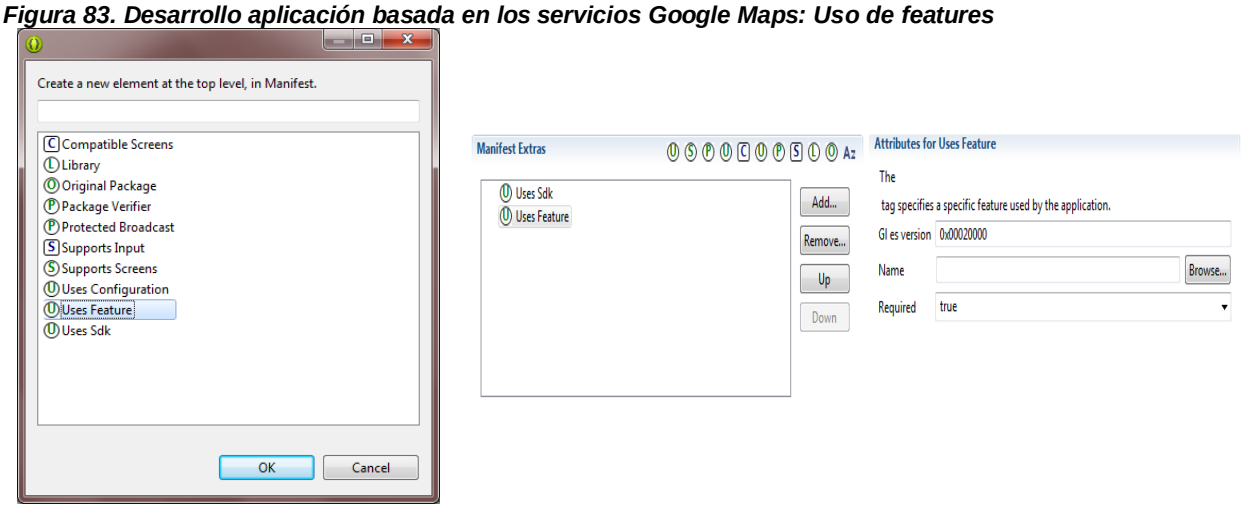


Por último, dado que la API v2 de Google Maps Android utiliza OpenGL ES versión 2, se debe especificar también dicho requisito en el *AndroidManifest.xml* añadiendo un nuevo elemento *Uses-feature*.

Para esto, hay que remitirse al archivo *AndroidManifest.xml* a la pestaña *Manifest*, se pulsa el botón *Add…* y se selecciona la opción *Uses Feature*.

Se rellena con los con los valores:
glEsVersion: *0x00020000*
required: *true*

Figura 83. Desarrollo aplicación basada en los servicios Google Maps: Uso de features



4.3.2.4 Implementación del mapa de Google en la aplicación

Ahora que se han definido las configuraciones base para la aplicación, se puede pasar a escribir el código de la aplicación.

Dentro del layout asociado a la actividad principal, *layout_principal.xml*, se define un control, que se añade en forma de *fragment*, quedando el código XML del layout, de la siguiente forma:

Figura 84. Desarrollo aplicación basada en los servicios Google Maps: Código creación fragment en XML.

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.SupportMapFragment"/>
```

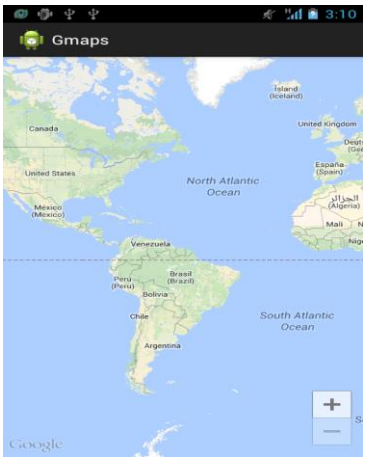
Dado que se utilizan *fragments*, la actividad principal *Principal.java* tendrá que extender a *FragmentAcitivity* (en vez de *Activity,* como es lo normal). Además se debe realizar la importación de la clase asociada a *FragmentAcitivity.*

Figura 85. Desarrollo aplicación basada en los servicios Google Maps: Código creación Activity del mapa.

```
package com.proyecto.gmaps;

import android.app.Fragment;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

public class Principal extends FragmentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
```

Posterior a esto, se podrá ejecutar la aplicación en un dispositivo virtual o por medio de depuración USB en un dispositivo físico.

Figura 86. Desarrollo aplicación basada en los servicios Google Maps: Imagen mapa Gmaps



4.3.2.5 Habilitación y uso del servicio GPS

Para poder hacer uso del servicio GPS dentro de la aplicación, se deben realizar dos cosas: escribir las líneas de código necesarias para habilitar el uso del GPS, en el archivo *Principal.java* y definir la autorización necesaria dentro del documento *AndroidManifest.xml*.

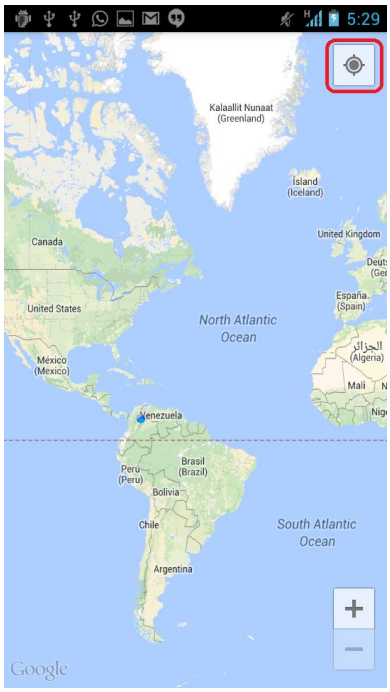
Figura 87. Desarrollo aplicación basada en los servicios Google Maps: Código para habilitar uso del GPS

```
//Obtiene el mapa de la aplicación, asociado al fragment del layout_principal
mapafragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
mapaGmaps = mapafragment.getMap();

//Establece que se puede hacer uso del GPS
mapaGmaps.setMyLocationEnabled(true);
```

Figura 88. Desarrollo aplicación basada en los servicios Google Maps: Permiso para usar el GPS en la aplicación

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

Figura 89. Desarrollo aplicación basada en los servicios Google Maps: Imagen de la aplicación con GPS activo



Posterior a la habilitación del uso del servicio del GPS, se crea un método el cual ubique el mapa en la posición exacta del GPS, haciendo un zoom aumentado del mismo.

Se hace una llamada al método *definirUbicacion()* dentro del método *onCreate()*, el cuál llamará al siguiente código:

Figura 90. Desarrollo aplicación basada en los servicios Google Maps: Código para centrar la ubicación del GPS

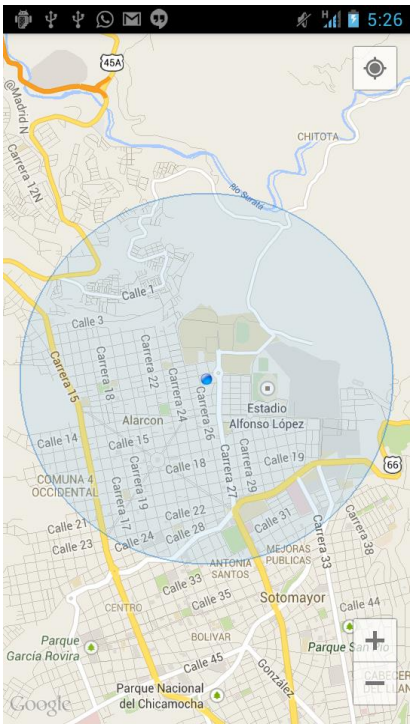
```
/** Define la ubicación actual del GPS */
public void definirUbicacion(){

    gestorlocalizacion=(LocationManager)getSystemService(Context.LOCATION_SERVICE);
    locationListener=new myLocationListener();
    gestorlocalizacion.requestLocationUpdates(LocationManager.GPS_PROVIDER, 60000, 0, locationListener);
    gestorlocalizacion.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 60000, 0, locationListener);
}
```

La variable locationListener, llamará a la clase myLocationListener, la cual está encargada de hacer actualizaciones de la posición actual y hacer zoom al mapa.

Figura 91. Desarrollo aplicación basada en los servicios Google Maps: Actualización posición GPS

```
private class myLocationListener implements LocationListener{

    public void onLocationChanged(Location location) {
        ubicacion=location;
        if(ubicacion!=null){
            double latitud, longitud;
            LatLng puntoactualGPS;
            latitud = ubicacion.getLatitude();
            longitud = ubicacion.getLongitude();
            puntoactualGPS = new LatLng(latitud, longitud);
            mapaGmaps.moveCamera(CameraUpdateFactory.newLatLng(puntoactualGPS));
            mapaGmaps.animateCamera(CameraUpdateFactory.zoomTo(15));
        }else {
            Toast.makeText(getBaseContext(), "no entraaa", Toast.LENGTH_LONG).show();
        }
    }
}
```

Figura 92. Desarrollo aplicación basada en los servicios Google Maps: Imagen aplicación con posición GPS centrada



4.3.2.6 Ubicación de punto de inicio y destino

Para poder realizar la ubicación de los puntos de inicio y destino de la ruta, se tomó ventaja de las facilidades que brinda la API de Googles Maps, la cual permite ubicar posiciones geográficas por medio de marcadores (Markers). Se definen dos marcadores en la declaración de variables de la clase *Principal.java.*

Figura 93. Desarrollo aplicación basada en los servicios Google Maps: definición de las variables marcadores.

```
Marker marcadorInicio = null, marcadorDestino = null;
```

Posteriormente, se define el manejo de los eventos que estarán asociados a estos dos marcadores, estos eventos serán:

- *setOnMapClickListener*: para ubicar los marcadores, permite máximo 2.

Figura 94. Desarrollo aplicación basada en los servicios Google Maps: evento setOnMapClickListener.

```
//OYENYE ONMAPCLICK: Ubica los Marcadores -- MÁXIMO 2
mapaGmaps.setOnMapClickListener(new OnMapClickListener(){
    public void onMapClick(LatLng punto){
        if(existeMI == false && nmarcadores == 0){ //no existe ningun marcador
            añadirMarcadorInicio(punto);
        }
        else if(existeMI == false && nmarcadores == 1){ //existe marcador de destino y no de inicio
            añadirMarcadorInicio(punto);
        }
        else if(existeMD == false && nmarcadores == 1){ //existe marcador de inicio y no de destino
            añadirMarcadorDestino(punto);
        }
        else if(nmarcadores == 2){ //existe marcador de inicio y no de destino
            Toast.makeText(getBaseContext(), "Ya hay dos marcadores, es el límite.", Toast.LENGTH_SHORT).show();
        }else{
            Toast.makeText(getBaseContext(), "ESTADO DESCONOCIDO", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Se definen las restricciones necesarias para realizar las ubicaciones de los marcadores y se realiza la invocación a los métodos definidos para añadir los marcadores:

añadirMarcadorInicio():

Figura 95. Desarrollo aplicación basada en los servicios Google Maps: función añadir marcador inicio

```
public void añadirMarcadorInicio(LatLng punto){
    nmarcadores++;
    existeMI = true;
    marcadorInicio = mapaGmaps.addMarker(
            new MarkerOptions()
                .title("Marcador inicio")
                .position(punto)
                .draggable(true)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));
    Toast.makeText(getBaseContext(), "numero de marcadores: " + nmarcadores, Toast.LENGTH_SHORT).show();
}
```

añadirMarcadorDestino():

Figura 96. Desarrollo aplicación basada en los servicios Google Maps: función añadir marcador destino

```
public void añadirMarcadorDestino(LatLng punto){
    nmarcadores++;
    existeMD = true;
    marcadorDestino = mapaGmaps.addMarker(
            new MarkerOptions()
                .title("Marcador destino")
                .position(punto)
                .draggable(true)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_VIOLET)));
    Toast.makeText(getBaseContext(), "numero de marcadores: " + nmarcadores, Toast.LENGTH_SHORT).show();
}
```

setOnMarkerClickListener: se encarga de borrar el marcador seleccionado.

Figura 97. Desarrollo aplicación basada en los servicios Google Maps: evento setOnMarkerClickListener.

```
//OYENTE ONCLICKMARKER: BORRA EL MARCADOR SELECCIONADO
mapaGmaps.setOnMarkerClickListener(new OnMarkerClickListener(){
    public boolean onMarkerClick(Marker marcador){
        if (marcador.equals(marcadorInicio)){
            mapaGmaps.clear();
            visorTexto.setText("");
            existeMI = false;
            nmarcadores--;
            marcadorInicio.remove();
            marcadorInicio = null;
            if (marcadorDestino != null){
                reañadirMarcadorDestino();
            }
            Toast.makeText(getBaseContext(), "Se borró el marcador de inicio", Toast.LENGTH_SHORT).show();
            Toast.makeText(getBaseContext(), "numero de marcadores: " + nmarcadores, Toast.LENGTH_SHORT).show();
        }
        if (marcador.equals(marcadorDestino)){
            mapaGmaps.clear();
            visorTexto.setText("");
            existeMD = false;
            nmarcadores--;
            marcadorDestino.remove();
            marcadorDestino = null;
            if(marcadorInicio != null){
                reañadirMarcadorInicio();
            }
            Toast.makeText(getBaseContext(), "Se borró el marcador de destino", Toast.LENGTH_SHORT).show();
            Toast.makeText(getBaseContext(), "numero de marcadores: " + nmarcadores, Toast.LENGTH_SHORT).show();
        }
        return true;
    }
});
```

Se definen las restricciones necesarias para realizar las ubicaciones de los marcadores y se realiza la invocación a los métodos definidos:

reañadirMarcadorInicio():

Figura 98. Desarrollo aplicación basada en los servicios Google Maps: función reañadir marcador inicio

```
public void reañadirMarcadorInicio(){
    marcadorInicio = mapaGmaps.addMarker(
            new MarkerOptions()
                .title(marcadorInicio.getTitle())
                .position(marcadorInicio.getPosition())
                .draggable(true)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));
}
```

reañadirMarcadorDestino():

Figura 99. Desarrollo aplicación basada en los servicios Google Maps: función reañadir marcador destino

```
public void reañadirMarcadorDestino(){
    marcadorDestino = mapaGmaps.addMarker(
            new MarkerOptions()
                .title(marcadorDestino.getTitle())
                .position(marcadorDestino.getPosition())
                .draggable(true)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_VIOLET)));
}
```

setOnMarkerDragListener: arrastra el marcador seleccionado a una nueva posición.

Figura 100. Desarrollo aplicación basada en los servicios Google Maps: evento setOnMarkerDragListener.

```
///// OYENTE ONMARKERDRAG: TRASLADA LA UBICACION DEL MARCADOR SELECCIONADO
mapaGmaps.setOnMarkerDragListener(new OnMarkerDragListener(){
    public void onMarkerDrag(Marker marcador){
        //POR EL MOMENTO NADA
    }

    @Override
    public void onMarkerDragEnd(Marker marcador) {
        if (marcador.equals(marcadorInicio)){
            mapaGmaps.clear();
            visorTexto.setText("");
            reañadirMarcadorInicio();
            if (marcadorDestino != null){
                reañadirMarcadorDestino();
            }
            Toast.makeText(getBaseContext(), "Se cambió de posición el marcador de inicio", Toast.LENGTH_SHORT).show();
        }
        if (marcador.equals(marcadorDestino)){
            mapaGmaps.clear();
            visorTexto.setText("");
            reañadirMarcadorDestino();
            if(marcadorInicio != null){
                reañadirMarcadorInicio();
            }
            Toast.makeText(getBaseContext(), "Se cambió de posición el marcador de destino", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onMarkerDragStart(Marker marcador) {
        //POR EL MOMENTO NADA
    }
});
```

4.3.2.7 Establecimiento de la ruta

Para definir una ruta entre dos puntos definidos, dentro de los *Google Play Services*, se encuentra una API llamada *Google Directions*, la cual por medio de consultas de tipo HTTP, recibe coordenadas y devuelve la información sobre la ruta entre las mismas, en formato JSON o XML.

El código utilizado para el establecimiento de las rutas se divide en varios pasos:

- Construcción de la URL para la consulta HTTP

Para comenzar, se define un método llamado *buscarRuta()* el cuál será invocado cada vez que se quiera trazar una ruta entre dos puntos determinados (puntos inicio y destino).

Figura 101. Desarrollo aplicación basada en los servicios Google Maps: función buscarRuta

```
/** Busca trazar la ruta */
public void buscarRuta(){
    try{
        posicionInicio = marcadorInicio.getPosition();
        posicionDestino = marcadorDestino.getPosition();
        String url = getDirectionsUrl(posicionInicio, posicionDestino);
        DownloadTask downloadTask = new DownloadTask();
        downloadTask.execute(url);
    }
    catch(Exception e){
        Toast.makeText(getBaseContext(), "NO HA UBICADO LAS POSICIONES", Toast.LENGTH_SHORT).show();;
    }
}
```

posiciónInicio: define el punto de origen a ser pasado en la URL.
posicionDestino: define el punto de destino a ser pasado en la URL.
getDirectionsURL(): se invoca este método que tiene como parámetros las posiciones. Retorna un String con la URL para la consulta HTTP.

Figura 102. Desarrollo aplicación basada en los servicios Google Maps: función getDirectionsUrl

```
/** Método que obtiene la URL para hacer la consulta de rutas al Google API Directions */
private String getDirectionsUrl(LatLng origin,LatLng dest){

    String str_origin = "origin="+origin.latitude+","+origin.longitude;
    String str_dest = "destination="+dest.latitude+","+dest.longitude;
    String sensor = "sensor=false"; //Sensor habilitado
    String parameters = str_origin+"&"+str_dest+"&"+sensor;
    String output = "json"; //formato de salida
    String url = "https://maps.googleapis.com/maps/api/directions/"+output+"?"+parameters;
    return url;
}
```

downloadTask: es un objeto de la clase *DownloadTask*, el cual extiende de *AsyncTask*, lo cual permitirá hacer la creación de un hilo secundario. Se invoca a su método *execute()* pasándole como parámetro la URL obtenida.

- Descarga de datos JSON desde el URL

La clase *DownloadTask*, al extenderse de la clase *AsyncTask*, funciona en dos hilos diferentes. El hilo principal ejecuta el código que se encuentra dentro del método *onPostExecute()*. El hilo secundario es creado para ejecutar el código que se encuentra dentro del método *doInBackground()*.

doInBackground(): se encarga de descargar los datos JSON desde la URL. Accede al método *downloadUrl()*, el cual devuelve una cadena String.

Figura 103. Desarrollo aplicación basada en los servicios Google Maps: Clase DownloadTask

```
/** Recupera datos del URL pasado(de pasar) */
private class DownloadTask extends AsyncTask<String, Void, String>{

    // Descargando datos en el hilo NO principal
    @Override
    protected String doInBackground(String... url) {

        String data = ""; // Para almacenar datos desde el servicio web
        try{
            data = downloadUrl(url[0]); // Capturando los datos desde el servicio web
        }catch(Exception e){
            Log.d("Tarea en segundo plano",e.toString());
        }
        return data;
    }

    // Ejecuta en el hilo Principal, despues de la ejecución de doInBackground()
    @Override
    protected void onPostExecute(String result) {

        super.onPostExecute(result);
        ParserTaskDirections parserTask = new ParserTaskDirections();
        parserTask.execute(result); // Invoca el hilo para analizar los datos JSON
    }
}
```

downloadUrl(): crea una conexión HTTP para acceder con a la URL, se conecta, lee los datos suministrados por la URL, línea por línea, los guarda y los convierte en una cadena String que es devuelta. Esta cadena es código JSON.

Figura 104. Desarrollo aplicación basada en los servicios Google Maps: función downloadUrl

```
/** Método para descargar datos JSON desde URL */
private String downloadUrl(String strUrl) throws IOException{

    String data = "";
    InputStream iStream = null;
    HttpURLConnection urlConnection = null;
    try{
        URL url = new URL(strUrl);
        urlConnection = (HttpURLConnection) url.openConnection(); // Creando una
        urlConnection.connect(); // Conectando a url
        iStream = urlConnection.getInputStream();  // Leyendo datos desde url
        BufferedReader br = new BufferedReader(new InputStreamReader(iStream));
        StringBuffer sb  = new StringBuffer();
        String line = "";
        while( ( line = br.readLine())  != null){
            sb.append(line);
        }
        data = sb.toString();
        br.close();
    }catch(Exception e){
        Log.d("Excepción durante la descarga url", e.toString());
    }finally{
         iStream.close();
         urlConnection.disconnect();
    }
    return data;
}
```

Regresando a la clase *DownloadTask:*

onPostExecute(): crea un nuevo objeto llamado *parserTask* de la clase *ParserTaskDirections*, la cual también es derivada de *AsyncTask.* Se ejecuta el método execute() de parserTask pasándole como parámetro el String devuelto por el método *downloadUrl()*.

- Análisis de código JSON: De JSON a lista de coordenadas

La clase *ParserTaskDirections* al igual que la clase *DownloadTask*, al extenderse de la clase *AsyncTask*, funciona en dos hilos diferentes. También llamados *doInBackground()* y *onPostExecute()*.

Figura 105. Desarrollo aplicación basada en los servicios Google Maps: Clase ParserTaskDirections

```
/** clase para analizar los lugares de Google en formato JSON */
private class ParserTaskDirections extends AsyncTask<String, Integer, List<List<HashMap<String,String>>> >{

    // doInBackground() & onPostExecute()

}
```

Dentro del método *doInBackground()* de la clase *ParserTaskDirections* se encuentra la parte del código encargada de traducir la cadena String devuelta por el método *downloadUrl(),* de datos JSON a coordenadas geográficas.

Figura 106. Desarrollo aplicación basada en los servicios Google Maps: método doInBackground – clase ParserTaskDirections

```
@Override
protected List<List<HashMap<String, String>>> doInBackground(String... jsonData) {

    JSONObject jObject;
    List<List<HashMap<String, String>>> routes = null;
    try{
        jObject = new JSONObject(jsonData[0]);
        DirectionsJSONParser parser = new DirectionsJSONParser();
        routes = parser.parse(jObject); // Inicia el análisis de datos
    }catch(Exception e){
        e.printStackTrace();
    }
    return routes;
}
```

jObject: convierte el String de datos JSON del método *downloadUrl()*, en un JSONObject.

parser: es un objeto de tipo DirectionsJSONParser, esta clase es la que se encarga de transformar el JSONObject recién creado en la ruta entre los puntos de origen y destino.

routes: recibe el resultado del método *parse()* de la clase DirectionsJSONParser, el cual es una lista de los puntos que forman la ruta e información de la misma.

- Graficación de la lista de coordenadas

Dentro del método *onPostExecute()* de la clase *ParserTaskDirections* se encuentra la parte del código encargada de seleccionar uno por uno la lista de los puntos que forman la ruta, para posteriormente ser almacenados en una *PolylineOptions* y ser graficados en el mapa.

Figura 107. Desarrollo aplicación basada en los servicios Google Maps: método onPostExecute – clase ParserTaskDirections

```
@Override
protected void onPostExecute(List<List<HashMap<String, String>>> result) {
    ArrayList<LatLng> points = null;
    String distancia = "";
    //String duracion = "";
    PolylineOptions opcionesLinea = null;
    markerOptions = new MarkerOptions();
    if(result.size()<1){
        Toast.makeText(getBaseContext(), "No Points", Toast.LENGTH_SHORT).show();
        return;
    }
// Recorriendo a través de todas las rutas
    for(int i=0;i<result.size();i++){
        points = new ArrayList<LatLng>();
        opcionesLinea = new PolylineOptions();
    // Obtención de ruta de orden i
        List<HashMap<String, String>> path = result.get(i);
    // Obtención de todos los puntos de ruta de orden i
        for(int j=0;j<path.size();j++){
            HashMap<String,String> point = path.get(j);
            if(j==0){    // Get distance from the list
                distancia = (String)point.get("distance");
                continue;
            }else if(j==1){ // Get duration from the list
                //duracion = (String)point.get("duration");
                continue;}
            double lat = Double.parseDouble(point.get("lat"));
            double lng = Double.parseDouble(point.get("lng"));
            LatLng position = new LatLng(lat, lng);
            points.add(position);}
        opcionesLinea.addAll(points);
        opcionesLinea.width(5);
        opcionesLinea.color(Color.MAGENTA); }
    visorTexto.setText("Distancia: "+ distancia); // Muestra la distancia
    mapaGmaps.addPolyline(opcionesLinea); // Dibujo polilínea en el mapa de Google
}
```

4.3.2.8 Establecimiento de sitios de interés

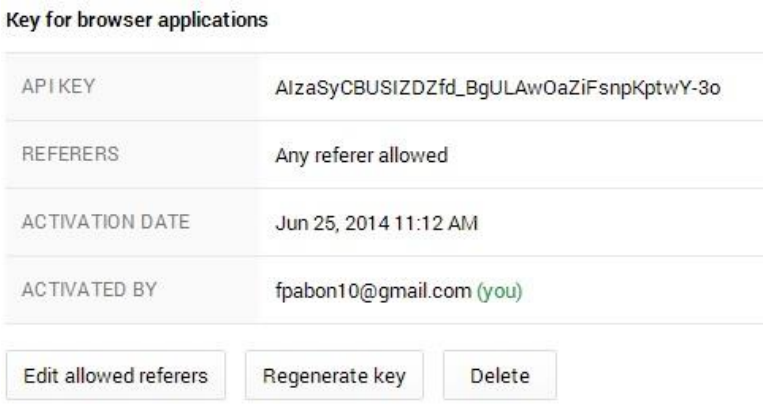
Para definir un sitio de interés, dentro de los *Google Play Services*, se encuentra una API llamada *Google Places*, la cual por medio de consultas de tipo HTTP, recibe una coordenada y devuelve la información sobre los sitios de interés más cercanos, en formato JSON o XML.

El código desarrollado para el establecimiento de los sitios de interés se divide en varios pasos:

- Obtención de la *Browser Key*

Para poder hacer uso de *Google Places*, es necesaria una API Key. Para este caso, se sigue un proceso similar al usado en el apartado *Obtención de la API Key* ubicado al principio de este numeral. Con la diferencia que hay que dirigirse a *APIs & Auth* y al seleccionar *APIs,* se activa la opción *Places API.* Después hay que dirigirse a *Credentials* > *Create new Key* > *Broswer Key* > *Create*. Se generará una clave similar a la siguiente:

Figura 108. Desarrollo aplicación basada en los servicios Google Maps: Browser Key



- Construcción actividad *Lugares.java*

Se crea una nueva actividad llamada *Lugares.java,* un layout xml asociado llamado *lugares.xml*, un archivo xml dentro de la carpeta *values,* llamado

arrays.xml y además se agrega la actividad dentro de la pestaña *Application* del *AndroidManifest.xml.* Con esta actividad se especificarán una lista de sitios de interés para escoger, por medio de una lista desplegable, entre los cuales destacan: cajeros, bancos, iglesias, estaciones de bomberos, hospitales, parques, CAIs, cines y restaurantes.

Figura 109. Desarrollo aplicación basada en los servicios Google Maps: Clase Lugares.java

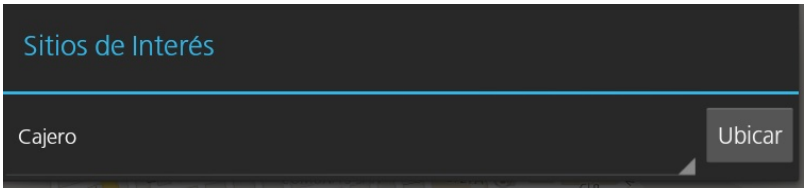
```
public class Lugares extends Activity{

    Spinner lugares;
    Button botonUbicar;
    String[] tipoLugar=null;
    String[] tipoLugarNombre=null;
    ArrayAdapter<String> adaptador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.lugares);

        botonUbicar = (Button) findViewById(R.id.button);
        lugares = (Spinner) findViewById(R.id.spinner);
        tipoLugar = getResources().getStringArray(R.array.lugar_tipo);
        tipoLugarNombre = getResources().getStringArray(R.array.lugar_tipo_nombre);
        adaptador = new ArrayAdapter<String>(getBaseContext(),
                android.R.layout.simple_spinner_dropdown_item, tipoLugarNombre);
        adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        lugares.setAdapter(adaptador);

        botonUbicar.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {

                int posicionSeleccionada = lugares.getSelectedItemPosition();
                String tipo = tipoLugar[posicionSeleccionada];

                Intent regresar = new Intent();
                regresar.putExtra("respuesta", tipo);
                setResult(RESULT_OK, regresar);
                finish();
            }
        });
    }
```

Figura 110. Desarrollo aplicación basada en los servicios Google Maps: archivos lugares.xml y arrays.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="5" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:layout_weight="0"
        android:text="@string/ubicar" />

</LinearLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string-array name="lugar_tipo">
        <item>atm</item>
        <item>bank</item>
        <item>church</item>
        <item>fire_station</item>
        <item>hospital</item>
        <item>park</item>
        <item>police</item>
        <item>movie_theater</item>
        <item>restaurant</item>
    </string-array>

    <string-array name="lugar_tipo_nombre">
        <item>Cajero</item>
        <item>Banco</item>
        <item>Iglesia</item>
        <item>Estación de bomberos</item>
        <item>Hospital</item>
        <item>Parques</item>
        <item>CAIs</item>
        <item>Cine</item>
        <item>Restaurante</item>
    </string-array>
</resources>
```

Figura 111. Desarrollo aplicación basada en los servicios Google Maps: imagen actividad Lugares.java



En la actividad *Principal.java* se escribe el código necesario para lanzar la actividad *Lugares.java,* además del código necesario para recibir respuesta del sitio de interés solicitado dentro de *Lugares.java.*

Figura 112. Desarrollo aplicación basada en los servicios Google Maps: método para lanzar Lugares.java

```java
public void lanzarLugares(View view){
    //Genera animación sonre el botón acerca de...
    if (marcadorInicio != null){
        Intent i = new Intent(this, Lugares.class);
        startActivityForResult(i, 1234);
    }else {
        Toast.makeText(getBaseContext(), "Ubique el marcador de inicio", Toast.LENGTH_SHORT).show();
    }
}
```

lanzarLugares(): se encarga de hacer el lanzamiento de la actividad *Lugares.java*

- Construcción de la URL para la consulta HTTP

Dentro de *Principal.java* también se escribe el método que se encarga de recibir el resultado de parte de la actividad *Lugares.java*, el cual se define en la siguiente imagen:

Figura 113. Desarrollo aplicación basada en los servicios Google Maps: método para obtener resultado de Lugares.Java

```
protected void onActivityResult(int requestcode, int resultcode, Intent regreso){
    if (requestcode==1234 && resultcode==RESULT_OK){
        respuesta = regreso.getExtras().getString("respuesta");
        Toast.makeText(getBaseContext(), "SE SELECCIONÓ: " + respuesta, Toast.LENGTH_SHORT).show();
        String url = getPlacesUrl(marcadorInicio, respuesta);
        PlacesTask placesTask = new PlacesTask();
        placesTask.execute(url);
    }
}
```

onActivityResult(): recibe el sitio de interés escogido dentro de la actividad *Lugares.java* y despliega la ubicación de los sitios de interés. *getPlacesUrl():* obtiene la dirección URL para realizar la consulta de los sitios incluidos en *Google Places.* Tiene como parámetros el marcador de inicio y el sitio de interés escogido. Retorna un String con la URL para la consulta HTTP.

Figura 114. Desarrollo aplicación basada en los servicios Google Maps: función getPlacesUrl

```
/** Método que obtiene la URL para hacer la consulta de lugares al Google API Places */
private String getPlacesUrl(Marker marcador, String tipo){

    StringBuilder cadena = new StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
    cadena.append("location="+ marcador.getPosition().latitude +","+ marcador.getPosition().longitude);
    cadena.append("&radius=5000");
    cadena.append("&types="+tipo);
    cadena.append("&sensor=true");
    cadena.append("&key=AIzaSyAFqM9E-WeEGwsIMSk2f8vuvgVJotuWudw");
    return cadena.toString();
}
```

placesTask: es un objeto de la clase *PlacesTask*. Se invoca a su método *execute()* pasándole como parámetro la URL obtenida.

- Descarga de datos JSON desde el URL

La clase *PlacesTask*, al extenderse de la clase *AsyncTask*, funciona en dos hilos diferentes. El hilo principal ejecuta el método *onPostExecute()*. El hilo secundario ejecuta el método *doInBackground()*.

doInBackground(): se encarga de descargar los datos JSON desde la URL. Accede al método *downloadUrl()*, el cual devuelve una cadena String.

Figura 115. Desarrollo aplicación basada en los servicios Google Maps: clase PlacesTask

```
/** Una clase, para descargar los lugares */
private class PlacesTask extends AsyncTask<String, Integer, String>{

    String data = null;
    // Invocado por el método execute() de este objeto
    @Override
    protected String doInBackground(String... url) {
        try{
            data = downloadUrl(url[0]);
        }catch(Exception e){
             Log.d("Background Task",e.toString());
        }
        return data;
    }

    // Ejecutado despues de la ejecución completa del método doInBackground()
    @Override
    protected void onPostExecute(String result){
        ParserTask parserTask = new ParserTask();
        parserTask.execute(result);
    }
}
```

El método *downloadUrl()* se encuentra especificado en la *Figura 99.*

onPostExecute(): crea un nuevo objeto llamado *parserTask* de la clase *ParserTask* la cual también es derivada de *AsyncTask.* Se ejecuta el método execute() de parserTask pasándole como parámetro el String devuelto por el método *downloadUrl()*.

- Análisis de código JSON: De JSON a lista de coordenadas

La clase *ParserTask* al igual que la clase *DownloadTask*, al extenderse de la clase *AsyncTask*, funciona en dos hilos diferentes. También llamados *doInBackground()* y *onPostExecute()*.

Dentro del método *doInBackground()* de la clase *ParserTask* se encuentra la parte del código encargada de traducir la cadena String devuelta por el método *downloadUrl(),* de datos JSON a coordenadas geográficas.

Figura 116. Desarrollo aplicación basada en los servicios Google Maps: método doInBackground – clase ParserTask

```
protected List<HashMap<String,String>> doInBackground(String... jsonData) {

    List<HashMap<String, String>> places = null;
    PlaceJSONParser placeJsonParser = new PlaceJSONParser();
    try{
        jObject = new JSONObject(jsonData[0]);
        places = placeJsonParser.parse(jObject);  // Obteniendo los datos analizados como una lista

    }catch(Exception e){
            Log.d("Exception",e.toString());
    }
    return places;
}
```

jObject: convierte el String de datos JSON del método *downloadUrl()*, en un JSONObject.
placeJsonParser: es un objeto de tipo *PlaceJsonParser*, esta clase es la que se encarga de transformar el JSONObject recién creado en la lista de sitios.
places: recibe el resultado del método *parse()* de la clase *PlaceJsonParser*, el cual es una lista de los puntos que forman la lista de los sitios de interés y su información.

- Graficación de la lista de lugares

Dentro del método *onPostExecute()* de la clase *ParserTask* se encuentra la parte del código encargada de seleccionar uno por uno la lista de los puntos que forman los sitios de interés encontrados, para posteriormente ser almacenados en marcadores y ser graficados en el mapa.

Figura 117. Desarrollo aplicación basada en los servicios Google Maps: método onPostExecute – clase ParserTask

```
// Ejecutado despues de la ejecución completa del método doInBackground()
@Override
protected void onPostExecute(List<HashMap<String,String>> list){

    // Limpia todos los marcadores existentes
    mapaGmaps.clear();
    visorTexto.setText("");
    reañadirMarcadorInicio();
    if (marcadorDestino != null){
        reañadirMarcadorDestino();
    }
    for(int i=0;i<list.size();i++){
        MarkerOptions markerOptions = new MarkerOptions();
        HashMap<String, String> hmPlace = list.get(i);
        double lat = Double.parseDouble(hmPlace.get("lat"));
        double lng = Double.parseDouble(hmPlace.get("lng"));
        String name = hmPlace.get("place_name");
        String vicinity = hmPlace.get("vicinity");
        LatLng latLng = new LatLng(lat, lng);
        markerOptions.position(latLng);
        markerOptions.title(name + " : " + vicinity);
        mapaGmaps.addMarker(markerOptions);
        visorTexto.setText("Sitio de interés: " +sitioInteres);
    }
}
```

5. SCRUM Y SUS SPRINTS EN EL DESARROLLO DEL PROYECTO

En el marco del desarrollo del proyecto se implementó la metodología SCRUM, mediante la cual se genera una pila de producto que contiene las historias de usuario que compondrán las funcionalidades de la aplicación. A partir de ellas se planifican los sprint para finalmente llevarlos a cabo, al terminar cada uno la aplicación debe ser completamente funcional.

En el Anexo A. Implementación de la metodología SCRUM se encuentra especificados las tablas con la pila del producto, la planificación de los sprint y su ejecución. Como se llevó acabo su implementación.

6. INFORME ESTADÍSTICO DEL DESEMPEÑO DE LOS SERVICIOS DE MAPAS

Posterior al desarrollo de las aplicaciones, se procedió a realizar el informe en el cual se medía el desempeño de las mismas. Para poder comparar el desempeño, se hizo un análisis acerca de los puntos posibles de comparación de los servicios. Al final se definieron 3 ítems de comparación, los cuales son:

- Veracidad de la ruta generada: si la ruta generada si es real y no infringe las restricciones definidas dentro del área metropolitana

- Distancia de la ruta generada: la distancia que hay entre el punto de inicio y el punto de destino.

- Tiempo de retardo de procesamiento de la ruta: el tiempo que demora en procesarse la información de la ruta desde el servidor.

El cuerpo del informe estadístico se encuentra ubicado en el Anexo B, dedicando subpartes del mismo a cada uno de los ítems anteriormente señalados, además de hacer comparaciones ruta a ruta de los servicios, uno contra el otro. Ofreciendo al final una serie de conclusiones acerca del desempeño tanto general como especifico de ArcGIS y Google Maps como servicios de mapas.

7. CONCLUSIONES

- Se logró estructurar la malla vial del área metropolitana de Bucaramanga, de forma que se encuentra publicada como un servicio en formato ESRI en el servidor del grupo de investigación Geomática.

- El desarrollo de las aplicaciones para dispositivos móviles Android basada en servicios de mapas de Google Maps y ArcGIS, permitió al usuario la búsqueda de rutas alternativas entre puntos de origen y destino, indicando a su vez la distancia de las mismas.

- La producción del informe estadístico permitió realizar una comparación acerca del rendimiento del uso de los recursos ofrecidos en Google Maps y la malla de ArcGIS construida, en relación a la generación de rutas, teniendo en cuenta como criterios de evaluación la veracidad de la ruta, la distancia y los tiempos de respuesta.

- Tras analizar los resultados del estudio estadístico basándonos en los criterios de evaluación, se concluye que el servicio de mapas más eficiente para el área metropolitana de Bucaramanga es ArcGIS, con un puntaje total de 461 puntos frente a 260 puntos obtenidos por Google Maps.

8. RECOMENDACIONES

- Para lograr ampliar las capacidades de la aplicación basada en los servicios de ArcGIS en lo referente a proporcionar tiempos de rutas, se sugiere realizar un estudio de tránsito donde se determinen las velocidades de la malla vial del área metropolitana, teniendo en cuenta también los horarios pues varia el flujo vehicular.

- Debido al contexto actual del mundo, donde se evidencia el crecimiento acelerado del uso de las tecnologías de la información, se aconseja que dentro del pensum de la carrera de Ingeniería de Sistemas se profundice en el manejo las tecnologías asociadas a software y dispositivos móviles.

BIBLIOGRAFÍA

[1] Android.[2014] Wikipedia. [Online]. http://es.wikipedia.org/wiki/Android

[2] Android (Operating system).[2014] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Android_(operating_system)

[3] Android Sample Code Home.[2014]. ArcGIS for Developers.[Online]. https://developers.arcgis.com/android/sample-code/

[4] API de Imágenes para Google Maps.[2011] Google Deveopers.[Online]. https://developers.google.com/maps/documentation/staticmaps/?hl=es#overview

[5] API de Google Maps. [2013] Google Developers.[Online]. https://developers.google.com/maps/mobile-apps?hl=es

[6] API de Google Maps. [2013]. W3School. [Online]. http://www.w3schools.com/googleAPI/google_maps_api_key.asp

[7] Application Programming Interface.[2014] Wikipedia-[Online]. http://en.wikipedia.org/wiki/Application_programming_interface

[8] ArcGIS Android 10.2.3 API. [2014]. ArcGIS for Developers.[Online]. https://developers.arcgis.com/android/api-reference/

[9] Crea aplicaciones basadas en la ubicación.[2011]. Google Deveopers.[Online]. https://developers.google.com/maps/location-based-apps?hl=es

[10] Desarrollo con el api de Google. [2014] Desarrollo Web. [Online]. http://www.desarrolloweb.com/manuales/desarrollo-con-api-de-google-maps.html

[11] Dibujar rutas entre dos localizaciones usando la API Google Directions con JSON. [2014]. Knowledge by Experience. [Online]. http://wptrafficanalyzer.in/blog/drawing-driving-route-directions-between-two-locations-using-google-directions-in-google-map-android-api-v2/

[12] Documentación y ejemplos de la API de Google Maps en Android. [2011]. Google Developers. [Online].https://developers.google.com/maps/documentation/android

[13] Formato del algoritmo de polilíneas codificadas. [2011]. Google Developers. [Online]. https://developers.google.com/maps/documentation/utilities/polylinealgorithm?hl=es

[14] GIRONÉS, Jesus Tomás. El gran libro de Android. Tercera edición. Bogotá, Colombia: Alfaomega, 2013, 431p.

[15] Global Positioning System. [2014] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Global_Positioning_System

[16] GMOR: Google Maps para la Optimización de Rutas. [2014] GMOR.[Online]. http://www.goma.ull.es/GMOR/Memoria-GMOR.pdf

[17] Google Places API.[2012]. Google Developers. [Online].

https://developers.google.com/places/?hl=es

[18] Google Play Services: Introducción y preparativos. [2013].Google Play Services: Introducción y Preparativos. [Online]. http://www.sgoliver.net/blog/?p=4116

[19] Guie ArcGIS Runtime SDK for Android.[2014] . ArcGIS for Developers.[Online]. https://developers.arcgis.com/android/guide/welcome-to-the-help-for-arcgis-runtime-sdk-for-android.htm

[20] Interfaz de Programación de aplicaciones.[2014] Wikipedia.[Online]. http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[21] Kit de Desarrollo de Software.[2014] Wikipedia. [Online]. http://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software

[22] LBS, servicios de localización, navegación GPS y mapas.[2011] LBSPro. [Online]. http://lbspro.com/?q=que-son-servicios-localizacion-LBS

[23] Licencia de API de Google Maps.[2011]. Google Developers.[Online]. https://developers.google.com/maps/licensing?hl=es

[24] Localización Geografía. [2014] Wikipedia. [Online]. http://es.wikipedia.org/wiki/Localizaci%C3%B3n_geogr%C3%A1fica

[25] Mapas en Android (Google Maps Android API v2) – I. [2013]. Google Play Services: Introducción y Preparativos. [Online].

http://www.sgoliver.net/blog/?p=3244

[26] Permisos para usar la API Google Maps Android. [2013] Stack Overflow. [Online]. http://stackoverflow.com/questions/20464136/android-google-maps-api-v2-requires-permission-that-it-already-has

[27] Qué es ArcGIS?. [2014] ArcGIS Resources.[Online]. http://resources.arcgis.com/es/help/getting-tarted/articles/026n00000014000000.htm

[28] ¿ Qué es un dataset de red.[2014]. ArcGIS Resurces Center. [Online]. http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#//004700000007000000

[29] Qué es SCRUM ? [2012]. Proyecto Ágiles. [Online]. http://www.proyectosagiles.org/que-es-scrum

[30] SCRUM.[2013]. Wikipedia. [Online]. http://es.wikipedia.org/wiki/Scrum

[31] Servicios basados en localización.[2013] Definición de Wikipedia.[Online http://es.wikipedia.org/wiki/Servicio_basado_en_localizaci%C3%B3n

[32] Sistema de Información Geográfica. [2014] Wikipedia. [Online]. http://es.wikipedia.org/wiki/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica

[33] Software Development Kit.[2014]. Wikipedia.[2014]. http://en.wikipedia.org/wiki/Software_development_kit

[34] Versión 3 del API de JavaScript de Google Maps.[2011] Google Developers.[Online]. https://developers.google.com/maps/documentation/javascript/?hl=es

[35] What is SCRUM.[2014] SCRUM. [Online]. https://www.scrum.org/resources/what-is-scrum/