UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

Especialidad Sistemas Electrónicos

TRABAJO FIN DE CARRERA

SISTEMA DE MONITORIZACION DE EQUIPOS REMOTOS MEDIANTE DISPOSITIVOS ANDROID

AUTOR:

MARIO CRUZ SÁNCHEZ

TUTOR:

IGNACIO BRAVO MUÑOZ

Julio 2015

A mis padres,

por haber confiado en mí.

A mi mujer

por apoyarme siempre.

A mis hijas,

para que aprendan que no hay

que dejar las cosas para el último momento.

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

INGENIERO TÉCNICO DE TELECOMUNICACIÓN

Especialidad Sistemas Electrónicos

Trabajo Fin de Carrera

SISTEMA DE MONITORIZACION DE EQUIPOS REMOTOS MEDIANTE DISPOSITIVOS ANDROID

Autor: MARIO CRUZ SÁNCHEZ

Director: IGNACIO BRAVO MUÑOZ

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

CALIFICACIÓN: .....

FECHA: .....

Contenido

Índice de Tablas

Índice de Ilustraciones

1. RESUMEN

1.1 RESUMEN

Este proyecto describe el desarrollo práctico de un sistema de telecontrol y tele medida basado en dispositivos Android. Dispondremos de varios equipos autónomos equipados con un módem GPRS, un receptor GPS y dos entradas analógicas para medir dos magnitudes variables. Estos equipos enviarán datos a un servidor que los almacenará en una base de datos. A través de dispositivos Android se podrán hacer lecturas de las mediciones de cada sensor en tiempo real y mostrar informes de los datos almacenados en el servidor. También se realizará un recorrido por los distintos sistemas operativos móviles, su historia y sus características, profundizando más en Android.

1.2 ABSTRACT

This project shows the developing of a remote-control and telemetry system based in Android devices. We provide several remote device equipped with GPRS modem, GPS receiver and two analog input for measuring of variable magnitudes. Data will be sent by the remote devices to the server, which will store them into a database. By means of Android devices we will be able to read the measurements in real-time and get reports obtained from the data base information. In addition, a review of the history and characteristics of mobile operating systems will be explained, especially focused on Android.

1.3 PALABRAS CLAVE

Tele medida, telecontrol, Android, dispositivos móviles.

1.4 KEYWORDS

Telemetry, remote control, Android, mobile devices.

1.5 RESUMEN EXTENDIDO

Este proyecto describe el diseño y el desarrollo práctico de un sistema de telecontrol y tele medida basado en dispositivos Android. Está pensado para poder telecontrolar y adquirir información de un conjunto de equipos remotos que comunican con un servidor a través de internet. Podrían ser equipos "inalámbricos" dotados de comunicaciones GPRS o Wifi, o bien equipos conectados a través de una red cableada, Ethernet o PLC's. La cuestión es que estos equipos puedan conectar al servidor a través de un puerto TCP, indistintamente del sistema de comunicación empleado.

El servidor tendrá dos funciones principales. Por una parte recogerá la información enviada por los equipos remotos y podrá enviarles respuestas o mensajes de control. Por otra parte atenderá a las aplicaciones cliente, desarrolladas en Android, y les suministrará la información requerida.

Las aplicaciones cliente, desarrolladas en Android podrán gestionar el sistema desde un SmartPhone o una Tablet. Enviarán peticiones al servidor y éste les devolverá la información requerida para presentársela al usuario. Mostrará tanto información en tiempo real como información histórica almacenada en la base de datos del servidor.

Para la ejecución práctica, la aplicación servidor se instalará en un "VPS" (Virtual Private Server) junto a una base de datos MySql. Se utilizarán varios equipos dotados de GPS, GPRS y entradas analógicas para adquirir muestras y enviarlas al servidor. Como ejemplo se trasmitirán, además de los datos capturados por el GPS, una medición de la tensión de entrada (será la de la batería de un coche) y una medición de temperatura gracias a un sensor conectado en otra entrada analógica. Estas muestras se enviarán a intervalos de tiempo prefijados al servidor.

La aplicación servidor se realizará en lenguaje Java, y empleará distintos hilos de proceso para atender a cada cliente.

La aplicación cliente se realizará en Android "nativo" utilizando el SDK de Android sobre Eclipse. Se requerirá el API de Google Maps para mostrar la ubicación de los sensores en el mapa. También manejará una base de datos SQLite para almacenar información necesaria para la aplicación.

2. MEMORIA

2.1 INTRODUCCIÓN

Hoy en día existe una gran variedad de dispositivos móviles en el mercado y su gran accesibilidad y versatilidad los convierten en perfectas herramientas de campo para controlar o monitorizar cualquier tipo de dispositivo.

En la siguiente memoria se desarrollará un sistema de monitorización de dispositivos remotos que capturan información por medio de sensores y la envían a un servidor para almacenarse en una base de datos.

El sistema permitirá el control de los equipos remotos y la lectura y tratamiento de los datos almacenados mediante un teléfono inteligente o tableta con sistema operativo Android, desde cualquier lugar y sin cables (siempre que se disponga de una conexión de datos). La *Ilustración 1* muestra gráficamente el concepto general del proyecto.

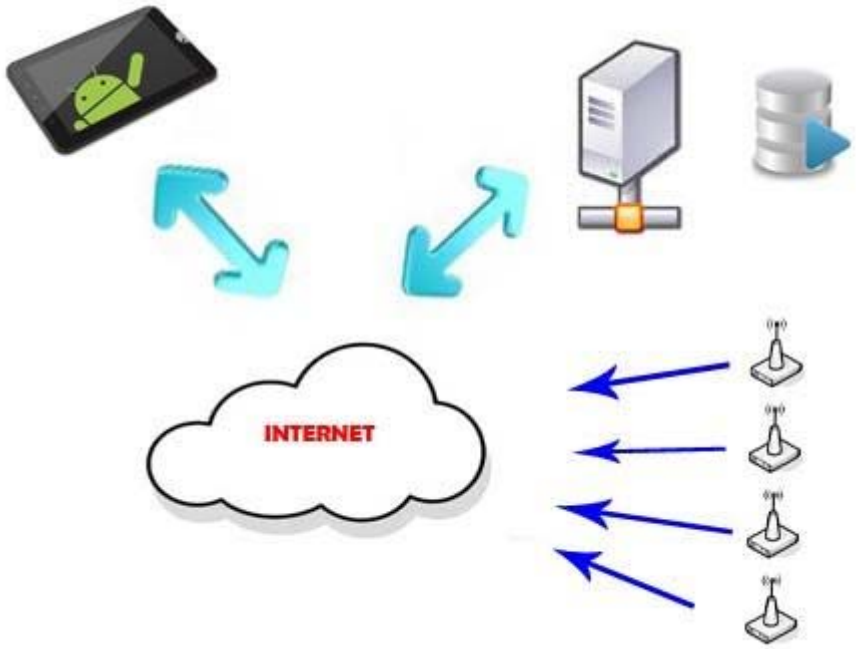

Ilustración 1: Idea general del proyecto

La idea fundamental de este trabajo es combinar las ventajas de los actuales sistemas de control de flotas basados en las tecnologías de comunicación móvil (GPRS, 3G, 4G) y en los receptores GPS, con las distintas posibilidades de adquisición de datos por medio de todo tipo de sensores aplicables a diversos campos de la industria y el transporte, y todo ello potenciado con las capacidades de los actuales dispositivos móviles, como teléfonos o tabletas, que incorporan gran potencia de proceso, un atractivo interfaz de usuario y gran poder de conexión.

Inicialmente se hará un recorrido por la historia y las características de los distintos sistemas operativos móviles más influyentes en los últimos 20 años, desde las primeras PDA Palm, hasta los modernos SmartPhones con Android, IOS y otros novedosos sistemas operativos que están apareciendo recientemente.

Entrando en materia, se tratará en profundidad el sistema operativo Android, sus características, su historia y se realizará una introducción a su entorno de desarrollo y las diversas herramientas y técnicas utilizadas en la actualidad por los desarrolladores.

Después de esta introducción teórica, se abordará la parte práctica del proyecto, explicando detalladamente cada uno de los subsistemas que componen la aplicación propuesta.

2.2 ALCANCE

Este trabajo engloba el software necesario para realizar la adquisición de datos de los equipos remotos y el almacenamiento en un servidor así como la aplicación cliente para poder hacer diversas lecturas de los datos adquiridos desde un dispositivo Android. El diseño y programación de los equipos de adquisición de datos no entra en el alcance de este proyecto. También se realizara una introducción teórica a los sistemas operativos móviles y se profundizará especialmente en Android.

2.3 SISTEMAS OPERATIVOS MOVILES

Los sistemas operativos móviles son similares a los que podrían funcionar en cualquier computador, pero adaptados a las peculiaridades de los terminales para los que están diseñados, que suelen presentar las siguientes características:

- **Se alimentan con baterías:** Por lo que una de las prioridades de estos sistemas es hacer una buena gestión de la energía.
- **Grandes posibilidades de comunicación:** Los dispositivos móviles hoy en día incorporan múltiples interfaces de comunicación, además del sistema GPRS, 3G o 4G propio del teléfono móvil, suelen venir provistos con bluetooth, wifi, NFC, GPS por lo que el sistema operativo debe soportar estos elementos.
- **Potencia y almacenamiento limitados:** aunque cada vez contamos con procesadores multi-nucleo más potentes, y con memorias flash y RAM de más capacidad, los sistemas operativos móviles deben funcionar también en terminales de gama baja con recursos más limitados.

- **Interfaz de usuario:** el interfaz de usuario de los dispositivos móviles es especial, el tamaño de la pantalla es reducido y cada vez encontramos menos botones físicos. La pantalla táctil es el método de entrada por excelencia y cada vez más se puede interactuar con el dispositivo por medio de la voz, imágenes y accesorios inalámbricos. El sistema operativo debe estar diseñado para proporcionar una buena experiencia de uso con estas condiciones.

Podríamos decir que el primer sistema operativo móvil fue el de Palm, que en 1996 lanzó Palm OS con aplicaciones como correo, agenda, bloc de notas… En 1999 aparece el primer terminal BlackBerry y hasta el año 2000 no llegarían PocketPC de Microsoft o Symbian, desarrollado por los principales fabricantes de teléfonos móviles y encabezados por Nokia. De ahí hasta nuestros días, los sistemas operativos móviles han evolucionado y se han hecho muy populares, acercando la tecnología a usuarios de todas las edades y sin conocimientos de informática.

En la *Ilustración 2* se muestra el año aparición de cada uno de los principales sistemas operativos móviles, y en algunos casos, el año de su extinción.

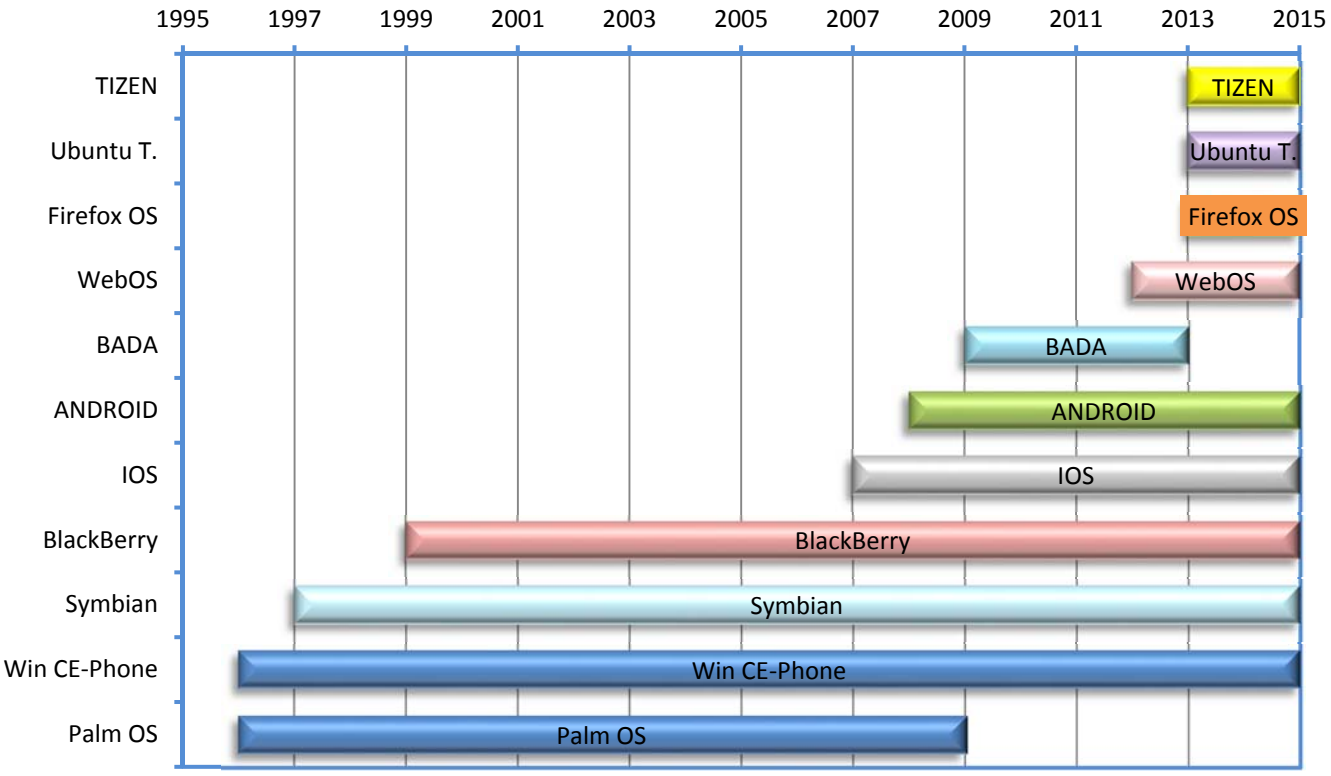

Ilustración 2: Cronograma Sistemas Operativos Móviles

En la *Ilustración 3* podemos observar la evolución en cuota de mercado de los sistemas operativos móviles desde 2009 hasta 2011. En este corto intervalo de tiempo vemos como el lider indiscutible del sector era Symbian, con mas del 40% del total de dispositivos en el mercado. Android e IOS son aun muy recientes, pero en 2010 ya se

igualan con Symbian y en 2011 se cambian las tornas totalmente, pasando a ser Android el sistema operativo móvil más utilizado.

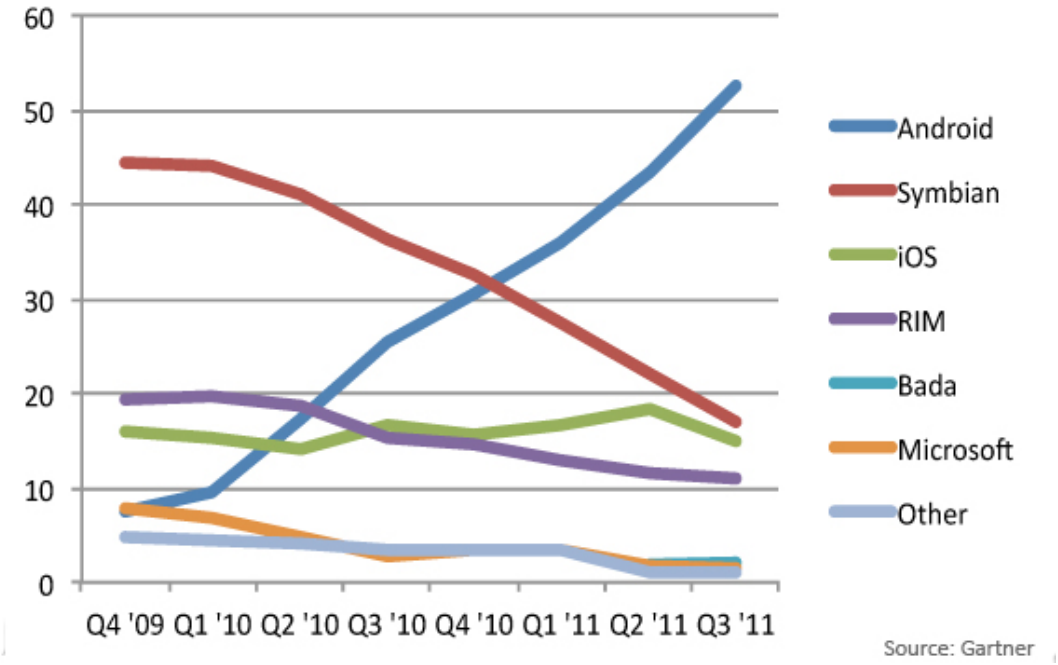

Ilustración 3: Evolución en número de usuarios S.O. móviles hasta 2011

A partir de 2011, la pugna prácticamente está repartida entre Android e IOS.

Android llega a ocupar un 84% de la cuota de mercado en 2014, como podemos apreciar en la *Ilustración 4*.

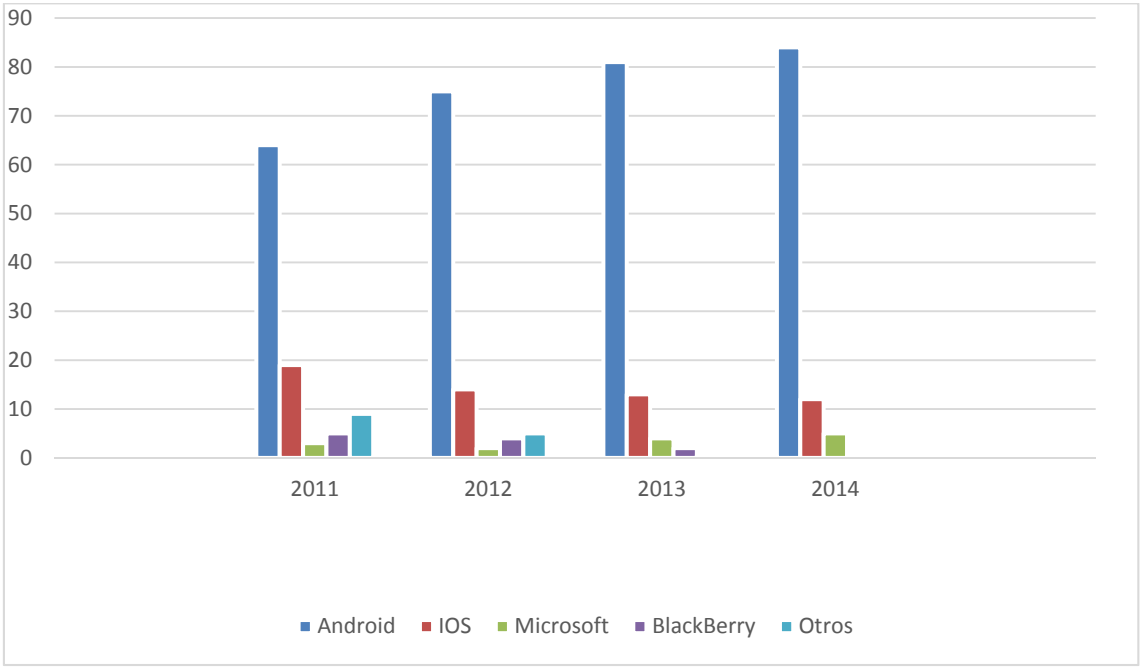

Ilustración 4: Evolución en número de usuarios S.O. móviles a partir de 2011

Pese a ser Android el sistema operativo móvil con mayor número de usuarios, las estadísticas siguen dando la ventaja a IOS en algunos aspectos, como en la actividad en internet, número de aplicaciones en App Store, y sobre todo en ventas de aplicaciones. Android está a la cabeza en cuanto a descargas de aplicaciones gratuitas, pero cuando se trata de aplicaciones de pago, son los usuarios de IOS los que gastan más dinero en éstas, como revela la gráfica de la *Ilustración 5.*

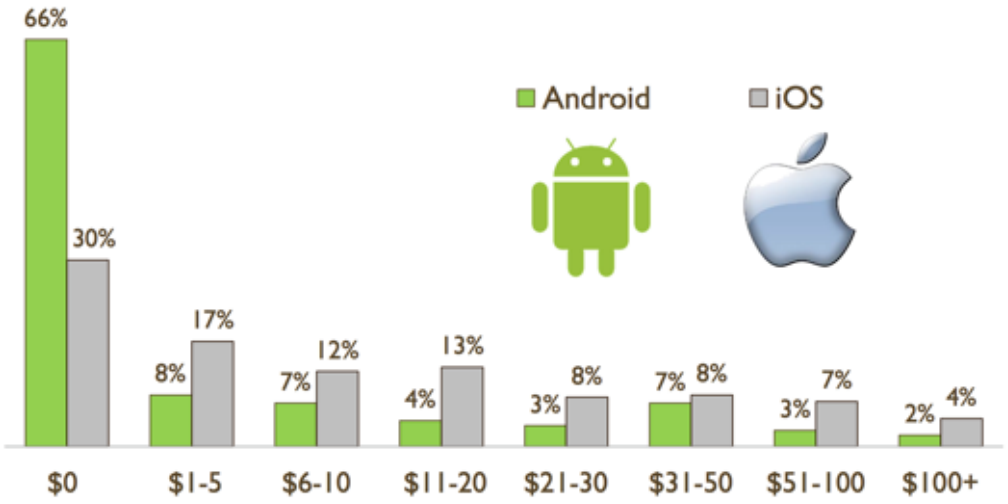

Ilustración 5: Comparativa gasto en aplicaciones.

A continuación se exponen las principales características de los sistemas operativos móviles más relevantes.

2.3.1 Symbian

Nombre/Empresa	SYMBIAN / Nokia
Logotipo	symbian
Inicio - fin	1997 – Actualidad (Fin previsto en 2016)
Última versión	Symbian Os 10.1 (2012)
Núcleo	Propietario (EKA2)
Aplicaciones	Teléfonos móviles
Palataformas HW	ARM, x86

Tabla 1: Características de Symbian

Symbian fue uno de los pioneros en cuanto a sistemas operativos móviles. Data de 1997, desarrollado por una alianza de fabricantes de teléfonos en la que destacaba NOKIA, siendo esta empresa la actual propietaria de Symbian.

En 2011 Nokia anuncia el final de Symbian, comenzando a comercializar terminales con Windows Phone. El soporte a Symbian finalizará en 2016.

Su lenguaje de programación nativo es C++ y su SDK oficial fue QT, aunque también se podían programar aplicaciones con C#, Java e incluso Python.

Ilustración 6: Capturas de pantalla de SmartPhone con Symbian

2.3.2 BADA

Nombre/Empresa	BADA / Samsung
Logotipo	
Inicio - fin	2009 – 2013 (Se integra en TIZEN)
Última versión	2.0.5 (2012)
Núcleo	Linux (gamas altas) o propietario (gamas bajas)
Aplicaciones	Teléfonos móviles
Plataformas HW	ARM

Tabla 2: Características de BADA

BADA, posteriormente reemplazado por TIZEN es un sistema operativo desarrollado por Samsung para sus teléfonos móviles. Se basa en el sistema operativo SHP OS, también propiedad de Samsung. Vio la luz en noviembre de 2009, y en 2010 se lanzó un SDK con el fin de atraer a los desarrolladores a su plataforma. Samsung no se jugó todo a una carta, fabricando terminales con Android, Windows Mobile, y su propio sistema BADA.

En Febrero de 2013 Samsung anuncia oficialmente la desaparición de BADA, después de haber vendido más de 30 millones de terminales con este sistema. BADA terminaría fusionando en un nuevo S.O. llamado TIZEN

Ilustración 7: Captura de pantallas de BADA

2.3.3 Palm-OS

Nombre/Empresa	Palm OS / Palm – U.S. Robotics - ACCESS
Logotipo	
Inicio - fin	1996 – 2009
Última versión	Palm OS Cobalt 6.1 (2009)
Núcleo	Propietario (Palm OS)
Aplicaciones	Teléfonos móviles, ordenador de mano, portátiles, relojes, GPS, videojuegos…
Plataformas HW	ARM/Freescale

Tabla 3: Características de Palm-OS

Palm OS es un sistema operativo móvil con licencia no libre. Diseñado en 1996 para las PDA de Palm, ha sido utilizado posteriormente en gran variedad de dispositivos móviles, incluyendo Smartphones, relojes inteligentes, lectores de código de barras y navegadores GPS.

Las versiones de Palm OS previas a la 5.0 estaban diseñadas para ejecutarse en los microprocesadores DragonBall de Motorola/Freescale. A partir de la versión 5.0 en adelante, el sistema correrá en equipos basados en arquitectura ARM.

El sistema incorporaba desde sus orígenes las aplicaciones típicas de toda PDA, como libreta de direcciones, calculadora, hoja de gastos, bloc de notas, tareas…

Las primeras PALM incorporaban infrarrojos para sincronización con el PC. A Partir de la versión 5.0 se incorporaron las comunicaciones por BlueTooth.

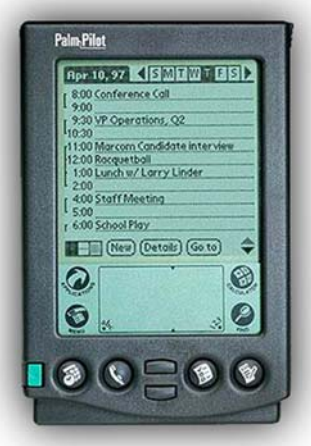

Ilustración 8: Palm Pilot

2.3.4 BlackBerry OS

Nombre/Empresa	BlackBerry / Rim-BlackBerry
Logotipo	
Inicio - fin	1999 – actualidad (2015)
Última versión	BlackBerry OS 10.3.1 (2015)
Núcleo	QNX (adaptación de Unix)
Aplicaciones	Teléfonos móviles, tabletas
Plataformas HW	Dispositivos BlackBerry

Tabla 4: Características de BlackBerry

El BlackBerry OS es un sistema operativo móvil de código cerrado desarrollado por la compañía canadiense RIM (Research In Motion) para los dispositivos BlackBerry. Se trata de un entorno multitarea que soporta multiples métodos de entrada diseñados por RIM para ser usados en terminales de mano, como la trackwheel, trackball, touchpad y pantallas táctiles. Su desarrollo se remonta la aparición de los primeros terminales de mano en 1999.

El SO BlackBerry tuvo un gran éxito para uso profesional como gestor de correo y agenda, debido especialmente a la seguridad y salvaguarda de los datos privados. Esto hizo que otros fabricantes incorporaran el cliente de correo de BlackBerry en sus equipos.

Permite sincronización con Microsoft Exchange Server y además es compatible también con Lotus Notes y Novell GroupWise.

Hoy en día está sufriendo una fuerte caída frente a otros sistemas como IOS o Android, aunque sigue en activo, con la última versión de su sistema operativo BlackBerry 10.

Ilustración 9: Blackberry 8520 Gemini

2.3.5 WebOS

Nombre/Empresa	WebOs / Palm - LG Electronics
Logotipo	webOS
Inicio - fin	2009 – actualidad (2015)
Última versión	BlackBerry OS 10.3.1 (2015)
Núcleo	Linux
Aplicaciones	Teléfonos inteligentes, tabletas, televisores y relojes.
Plataformas HW	ARM

Tabla 5: Características de webOS

Se trata de un sistema operativo basado en Linux y desarrollado inicialmente por Palm para su uso en móviles y tabletas, pero que posteriormente fue adquirido por HP, que termino liberando su código y vendiéndoselo a la coreana LG Electronics. El Palm Pre, lanzado en junio de 2009, fue el primer dispositivo en utilizar esta plataforma.

Actualmente se puede encontrar en multitud de televisores LG, pasando a denominarse LGWebOS.

Ilustración 10: Televisor LG con WebOS

2.3.6 Windows CE, Windows Embedded, Windows Phone y Windows 10

Nombre/Empresa	Windows Phone Windows 10 / Microsoft
Logotipo	Windows Phone Windows 10
Inicio - fin	Windows Phone 2010 – 2015 Windows 10 2015 -
Última versión	Windows Phone 8.1.2 (2015) Windows 10 (lanzamiento en 2015)
Núcleo	Windows NT (Windows CE en versiones anteriores)
Aplicaciones	Windows Phone: Teléfonos inteligentes. Windows 10: Teléfonos inteligentes y tebletas.
Plataformas HW	X86, MIPS, ARM

Tabla 6: Características de Windows Phone

Windows CE es un Sistema operativo desarrollado por Microsoft destinado a sistemas embebidos. Es compatible con procesadores x86, MIPS y ARM. Está diseñado para poder trabajar en sistemas con muy poca RAM (a partir de 1Mb) y no necesita un sistema de almacenamiento externo, ya que puede cargarse desde memoria ROM. Windows CE puede considerarse un RTOS (Sistema operativo en tiempo real). Apareció oficialmente en 1996 y desde entonces ha funcionado en multitud de dispositivos como Pocket-Pc, teléfonos móviles, navegadores de automóviles, GPS, etc.

Windows Embedded, sin embargo, es una adaptación de Windows Xp diseñado para poder ser "embebido" o "encajado" a medida en dispositivos para realizar una función concreta. Este sistema operativo es el que usan los ATM (cajeros automáticos), surtidores, puntos de venta, algunas consolas y máquinas de videojuegos, etc.

Windows Phone es la evolución de Windows Mobile para los actuales SmartPhones, y está basado en el núcleo Windows Embedded CE 6.02. Compite directamente contra Android e iOS. Su última versión es Windows Phone 8.1, lanzado el 14 de abril de 2014.

Windows Phone incorpora con una actualizada interfaz de usuario llamada Modern UI, basada en mosaicos dinámicos que actúan como accesos a aplicaciones y a su vez proporcionan información al usuario de llamadas, mansajes, etc.

Windows 10 es el sucesor de Windows Phone. Microsoft decide unificar el nombre con el sistema operativo para PC, eliminando el "Phone" del nombre, aunque realmente existirán dos versiones distintas, una para móviles y pequeñas tabletas y otra para PC's de escritorio, portátiles y tabletas más grades.

2.3.7 FirefoxOS

Nombre/Empresa	Firefox OS / Mozilla Corporation
Logotipo	
Inicio - fin	2013 – actualidad (2015)
Última versión	2.1 (2014)
Núcleo	Linux
Aplicaciones	Teléfonos inteligentes, tabletas, televisores y relojes.
Plataformas HW	ARM y x86

Tabla 7: Características de FirefoxOS

Firefox OS es un Sistema operativo móvil basado en HML5 sobre núcleo Linux. Ha sido desarrollado por la corporación Mozilla con el apoyo de diversas empresas. Es de código abierto y está mantenido por su comunidad.

Inicialmente estuvo enfocado en los dispositivos móviles, smartphones y tabletas, sobre todo enfocado a las gamas bajas. En julio de 2013, Telefónica comenzó la venta del el ZTE Open, primer terminal con Firefox OS, que fue rápidamente seguido por el teléfono Peak de Geeksphone. También se está tratando de aplicar a otros dispositivos como Raspberry Pi, Smart TV y equipos para conectar al televisor por HDMI.

La arquitectura de Firefox OS se compone de tres elementos fundamentales, que han sido denominados como **Gonk, Gecko y Gaia**.

Gonk es el kernel, basado en Linux y una capa de abstracción del HW. Hereda gran parte del Android y los drivers y dispositivos de las distribuciones Linux.

Gecko es el entorno de ejecución, donde están implementados los estándares de HTML, CSS y JavaScript.

Gaia es la interfaz gráfica del sistema, escrita íntegramente en HTML, CSS y JavaScript.

2.3.8 Ubuntu Touch

Nombre/Empresa	Ubuntu Touch / Canonical LTD
Logotipo	
Inicio - fin	2013 – actualidad (2015)
Última versión	Ubuntu 13.10
Núcleo	Linux
Aplicaciones	Teléfonos inteligentes y tabletas.
Plataformas HW	ARM

Tabla 8: Características de Ubuntu Touch

Ubuntu Touch es un sistema operativo móvil basado en Linux desarrollado por Canonical. De desarrolló con la intención de que una misma interfaz gráfica pudiera utilizarse en ordenadores de sobremesa, portátiles, netbooks, tabletas y teléfonos inteligentes.

Ubuntu Touch utiliza las mismas tecnologías propias del Escritorio de Ubuntu, por lo que cualquier aplicación diseñada para esta plataforma puede ser usada en ambas. Además puede funcionar como un sistema de escritorio completo conectándole un monitor, teclado y ratón externos.

En la actualidad compañías como la española Bq y la china Meizu están "experimentando" lanzando al mercado terminales con Ubuntu Touch, los cuales serán vendidos mundialmente a través de sus respectivas páginas web.

En el caso de BQ ya está disponible el modelo Aquaris E4.5 con este sistema operativo.

2.3.9 TIZEN

Nombre/Empresa	TIZEN / Samgung
Logotipo	
Inicio - fin	2012 – actualidad (2015)
Última versión	3.0 Magnolia
Núcleo	Linux
Aplicaciones	Teléfonos inteligentes, tabletas, televisores, relojes, vehiculos.
Plataformas HW	ARM y x86

Tabla 9: Características de TIZEN

Tizen es una plataforma de software de código abierto. Actualmente, el desarrollo de Tizen está abanderado por Intel, Samsung y la Linux Foundation y cuenta con el respaldo de empresas como eBay, Panasonic, Sharp, ZTE, MacAfee, TrendMicro y Konami .

Las interfaces de desarrollo de Tizen están basadas en HTML5, y están diseñados tanto para teléfonos inteligentes como para tabletas, PC's, Smart TV's y otros dispositivos.

Una característica de este nuevo sistema operativo es que aparte de traer incluidas una serie de aplicaciones nativas y de funcionar con las que se desarrollen con HTML5 (similar al funcionamiento de Firefox OS), también podrá ejecutar aplicaciones Android, aprovechando así el gran número de aplicaciones existentes para este sistema.

2.3.10 APPLE IOS

Nombre/Empresa	IOS / Apple
Logotipo	
Inicio - fin	2007 – actualidad (2015)
Última versión	8.4
Núcleo	XNU (propietario Apple)
Aplicaciones	Teléfonos inteligentes (IPhone), tabletas (IPad), dispositivos multimedia (Ipod), Apple TV
Plataformas HW	ARM

Tabla 10: Características de Apple IOS

IOS es el sistema operativo móvil desarrollado por Apple para sus dispositivos. Su primera aparición oficial fue en junio de 2007, al lanzarse el IPhone. Inicialmente fue llamado IPhone OS, cambiando su denominación a IOS en 2010.

El lanzamiento del IPhone supuso el nacimiento del concepto de Smart Phone que conocemos hoy en día. Previamente ya existían computadores de mano (PDA) con teléfono incorporado, en los que habitualmente había que usar un puntero para accionar la pantalla táctil con precisión. Apple quería romper con esa tendencia, y para ello lo que hizo fue evolucionar el sistema operativo de su dispositivo multimedia (Ipod) y dotarle de servicio de telefonía y datos. Lo que consiguió fue lo que ahora todos entendemos como Smart Phone.

El sistema operativo IOS puede ser utilizado en los dispositivos IPod, IPad e IPhone, todos de Apple, y no se permite su utilización en dispositivos de otros fabricantes.

Su principal característica, que fue lo que le hizo destacar y ser imitado por el resto, es su sencillez de uso a través de la pantalla táctil. La sensación del usuario es de fluidez y respuesta inmediata a las interacciones.

Las aplicaciones se muestran en la pantalla principal, reservando una zona en la parte inferior para las más usadas, de forma que estén siempre accesibles.

En la parte superior de la pantalla encontramos la barra de estado, que muestra la hora, el nivel de cobertura y el estado de la batería.

Las aplicaciones pueden ser fácilmente agrupadas en carpetas para clasificarlas y acceder a ellas rápidamente.

El sistema operativo IOS proviene de MAC OS X (el sistema operativo de los pc de sobremesa de Apple), que a su vez fue desarrollado a partir un sistema operativo Unix

Al igual que la mayoría de sistemas operativos, IOS está estructurado en capas.

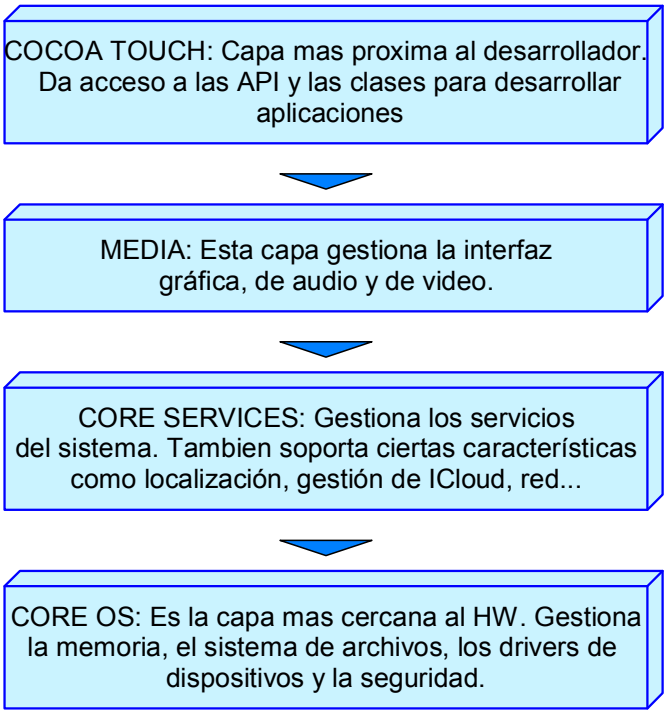

Ilustración 11: Estructura de capas IOS

IOS fue pionero en muchos aspectos, pero algunos de los más destacados son:

- **Multitarea:** A partir de IOS 4, es capaz de mantener múltiples aplicaciones funcionando al mismo tiempo, dejando una en primer plano y el resto en segundo plano. No se trata de multitarea real, ya que las aplicaciones que quedan en segundo plano realmente se cierran y luego se vuelve a abrir en el mismo estado en que estaban. Solo se produce multitarea real, por ejemplo, si se está reproduciendo un archivo de audio a la vez que se usa otra aplicación. En IOS 8 si se anuncia multitarea real, pudiendo tener múltiples aplicaciones abiertas a la vez y en primer plano, compartiendo la pantalla en dispositivos grandes como IPad.

- **Multi Touch:** El sistema Multi Touch, que detecta pulsaciones simultaneas en la pantalla y es capaz de identificar patrones y gestos, hacen que la interfaz de usuario sea tan ágil y fluida.

- **Tienda de APP's:** Desde el terminal se accede directamente a la tienda de aplicaciones "ITunes", desde donde se pueden descargar las aplicaciones al teléfono, unas de pago y otras no. Apple realiza un control exhaustivo de la calidad y la seguridad de las aplicaciones que ofrece. En "ITunes" también están disponibles las actualizaciones del sistema. Hasta la fecha, se considera la App Store con más aplicaciones disponibles.

- **Sincronización "en la nube":** Los teléfonos con IOS (a partir de IOS 5) están sincronizados con la nube de Apple, ICloud. Esto permite a las aplicaciones guardar ficheros y datos de configuración en los servidores de Apple y ser así accesibles desde otros dispositivos del mismo usuario, o como copia de respaldo.

2.3.11 ANDROID

Nombre/Empresa	Android / Google
Logotipo	
Inicio – fin	2008 – actualidad (2015)
Última versión	5.1 LollyPop
Núcleo	Linux
Aplicaciones	Teléfonos inteligentes, tabletas, televisores, relojes, vehículos.
Plataformas HW	ARM y x86

Tabla 11:Características de Android

El sistema operativo Android fue diseñado inicialmente por Android Inc. con el apoyo de Google, pero en 2005 terminaría siendo absorbida por el gigante de Mountain view.

El primer Android vio la luz oficialmente el 22 de Octubre de 2008 cuando el terminal T-Mobile G1 fue lanzado en Estados Unidos. En ese momento no existían muchas de las características de las que hoy no podríamos prescindir pero ya contenía muchos aspectos que han ido perdurando versión tras versión.

Android es un sistema operativo basado en el *kernel* de Linux. Está pensado para trabajar principalmente sobre dispositivos con pantalla táctil, como smartphones o tabletas, pero que cada vez se está aplicando a mas equipos, como televisores, relojes e incluso automóviles

La arquitectura de Android está concebida en orden a las siguientes capas:

- KERNEL LINUX: Es la capa inferior, el núcleo del sistema operativo. Basado en un Linux 2.6, con un centenar de parches, provee al sistema de las funcionalidades más básicas, como la gestión de la memoria, servicios y dispositivos.

- LIBRERIAS: Sobre el Kernel de Linux existen un conjunto de librerías escritas en C y C++ que dotan al sistema de múltiples características esenciales, como el WebKit (motor de navegación web) SQLite (motor de bases de datos), librerías de reproducción de audio y de video, librerías SSL para seguridad en internet, etc…

- ENTORNO DE EJECUCIÓN: Esta es la tercera capa de la arquitectura, y es accesible por las dos siguientes. Está compuesta por un elemento clave en el sistema, como es la “Máquina Virtual”. En las primeras versiones se utilizaba la máquina virtual DavLink, y a partir de Android 5, la maquina ART. Es un tipo especial de máquina virtual de Java específicamente diseñada para Android y establece la capa de abstracción entre el sistema y las aplicaciones, permitiendo la ejecución del código Java en un proceso independiente para cada aplicación, administrando los recursos de sistema.
  El entorno de ejecución también aporta las librerías necesarias para poder desarrollar aplicaciones usando un lenguaje Java estándar.

- FRAMEWORK DE APLICACIONES: Representa el conjunto de herramientas de desarrollo comunes empleadas por todas las aplicaciones del nivel superior. Se trata de librerías de alto nivel en forma de clases JAVA.

- APLICACIONES: Representa la capa de más alto nivel, donde se encuentran las aplicaciones incluidas en el sistema operativo y las instalada posteriormente por

el usuario. Estas aplicaciones utilizan las API, las librerías y los servicios de los niveles inferiores.

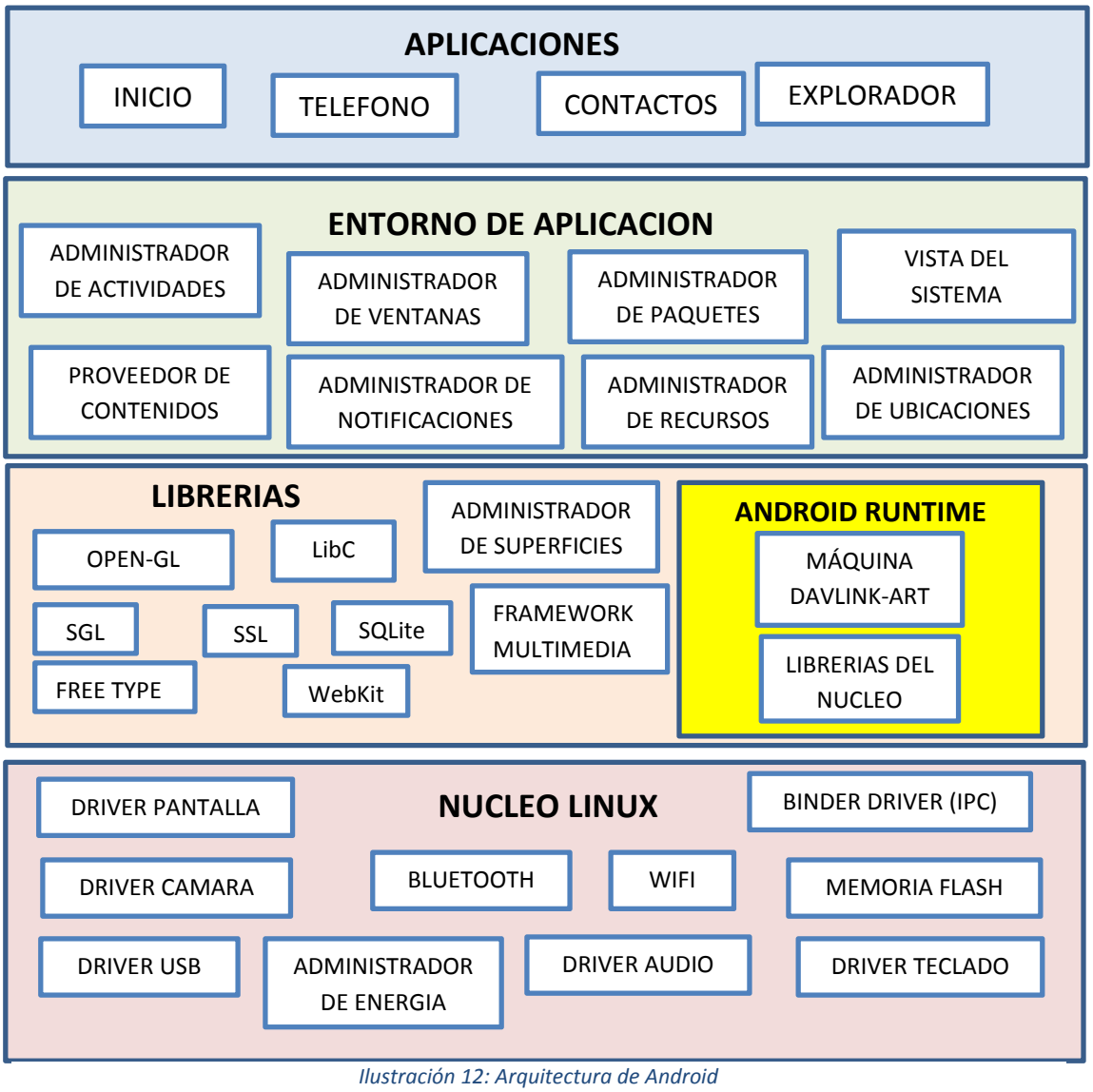

Ilustración 12: Arquitectura de Android

Algunas de las principales características a destacar en Android son las siguientes:

- Adaptable a todo tipo de pantallas, de cualquier resolución y tamaño. Utiliza el estándar OpenGL para representar gráficos en 2D y 3D
- Multitáctil, ya que incorpora soporte para este tipo de pantallas táctiles capacitivas que detectan múltiples puntos de contacto simultáneos.
- En cuanto a conectividad, soporta los sistemas : GSM-EDGE, GPRS DCMA, UMTS, IDEN, EV-DO, Bluetooth, Wi-Fi, HSDPA, HSPA+, HSDPA+.NFC y WIMAX
- Navegador basado en el motor de renderizado WebKit, de código abierto.
- Soporte para multimedia y “streaming” en todos los formatos más utilizados.

- Multitarea REAL. Las aplicaciones que están en segundo plano continúan ejecutándose.
- Búsqueda por voz, incluida desde la primera versión.
- Google Play, tienda de aplicaciones y contenido multimedia, gratuito y de pago, para acceder a los contenidos y descargarlos directamente desde el teléfono.
- Identificación por medio de cuenta Google que da acceso a todos los servicios integrados, como correo Gmail, almacenamiento en la nube con Google Drive, a la red social Google+, copias de seguridad en la nube, mensajería instantánea y video llamada con Hangouts, etc.
- Soporte para multitud de sensores y dispositivos como acelerómetros, giróscopos, sensores de proximidad, sensores de presión, cámara de fotos, magnetómetro, sensores fotoeléctricos, tarjetas de memoria, etc.
- Soporte USB-OTG para dispositivos USB externos, como dispositivos de almacenamiento, teclados, ratones, mandos de juego, etc.

2.3.12 COMPARATIVA ENTRE LOS S.O. MOVILES MÁS USADOS

A continuación se reflejan diversas comparativas entre los tres sistemas operativos móviles más extendidos en la actualidad: Android, IOS y Windows Phone.

CARACTERISTICAS	ANDROID	IOS	WINDOWS PHONE
Desarrollado por	Google	Apple	Microsoft
Núcleo	Linux	XNU	Windows7.net
Lenguajes nativos	C, C++, Java	C, C++,Objective C	C, C++, C#
Código abierto	SI	NO	NO
Plataforma HW	ARM,x86, MIPS POWER	ARM	ARM, x86
Multitarea	SI	Pseudo	SI
SDK	Android SDK	iPhone SDK	Visual Studio
Tienda de Apps	Google Play 1,43 millones de aplicaciones (2014)	I-tunes 1,22 millones de aplicaciones (2014)	Store 122.000 aplicaciones
Coste SDK	SDK gratuito. Cuota única de 20€ para publicar en Google Play de por vida	SDK gratuito (limitado) y de pago 80€ anuales para poder desarrollar.	SDK gratuito, cuanta de desarrollador por 14€ anuales
Soporte en la nube	Google Drive	I-Cloud	SkyDrive
Customización	SI	NO (salvo jailbreak)	NO

Tabla 12: Comparación entre Android, IOS y Windows Phone

2.4 EVOLUCIÓN DE "ANDROID"

Desde su primera versión en 2008 hasta nuestros días la Android ha ido evolucionando recortando distancias con su principal competidor (IOS de Apple) hasta sobrepasarle y doblarle en cuanto a cuota de mercado. Las distintas versiones que han ido apareciendo han ido siempre asociadas a un nombre clave, siempre relacionado con nombres de postres o dulces y en orden alfabético.

2.4.1 VERSIONES DE ANDROID

- **Android 1.0: Apple Pie**

La primera versión de Android (1.0 Apple Pie) fue lanzada el 22 de octubre de 2008, sobre un dispositivo HTC Dream. Esta versión ya contaba con la mayoría de las principales características que conocemos hoy en día, como Android Market, un Navegador Web, soporte para mensajes de texto SMS y MMS, marcador para llamadas, así como las aplicaciones de Google más conocidas; Google Maps con Street View, Google Sync para sincronizar el correo de Gmail, búsqueda de Google integrada, calendario y Contactos, Google Talk (servicio de mensajería de Google que ahora conocemos como Hangouts) y YouTube.

Por ultimo cabe destacar que Android 1.0 ofreció desde sus inicios el soporte para cámara fotográfica, WiFi y Bluetooth, y los característicos iconos de notificaciones que se muestran en la barra de estado, ofreciendo también la posibilidad de configurar alertas acústicas, luminosas o vibraciones.

Ilustración 14: Logotipo Android 1.0 Apple Pie

- ### Android 1.1: Banana Bread

Lanzada el *9 de febrero de 2009* la actualización de Android 1.1 llegó solo para los dispositivos T-Mobile G1. La actualización solventó algunos errores de la versión anterior y mejoró la API, además de añadir algunas funcionalidades nuevas en Google Maps y en la pantalla de llamadas telefónicas.

Ilustración 15: Logotipo Android 1.1 Banana Bread

- ### Android 1.5: Cupcake

Lanzada en abril de 2009, esta versión fue la primera actualización importante del sistema, introduciendo multitud de mejoras visuales así como el teclado interactivo en pantalla ya que los anteriores aun funcionaban con teclado físico. También se introducen por primera vez los Widgets (vistas en miniatura de aplicaciones en el escritorio).

Ilustración 16: Logotipo Android 1.5 CupCake

- ### Android 1.6: Donut

Android 1.6, que apareció en septiembre de 2009, trajo pequeñas mejoras en varios aspectos, como la barra de búsquedas, la cámara, un renovado Android Market, soporte para resoluciones de pantalla mayores, etc.

Ilustración 17: Logotipo Android 1.6 Donut

- ## Android 2.0, 2.1: Éclair

La versión 2.0 fue lanzada el 26 de octubre del 2009, dando paso al crecimiento exponencial del sistema. Como novedades mejoraba la sincronización con redes sociales, como Facebook o Twitter, mejoras considerables en la interfaz gráfica y el reconocimiento de voz. La versión 2.1 fue una actualización menor sobre la 2.0, mejorando aspectos visuales e incorporando soporte para pantallas "multitouch".

Ilustración 18: Android 2.0 Eclair

- ## Android 2.2: Froyo

En mayo de 2010 Froyo trajo principalmente mejoras de rendimiento y velocidad. Además incorporó por primera vez la posibilidad de compartir la conexión 3G a través del Wifi, actuando como un router inalámbrico. También permitió la instalación de aplicaciones en la tarjeta de memoria externa.

Ilustración 19: Logotipo Android 2.2 Froyo

- Android 2.3: Gingerbread

Gingerbread incorporó una gran cantidad de novedades a nivel estético. Fue lanzada el 6 de diciembre de 2010 renovando profundamente el interfaz de usuario e introduciendo mejoras en velocidad y facilidad de uso.

Gingerbread también incorporó soporte para sistema de archivos EXT4, optimizándose así para la aparición de los terminales multi-nucleo. Otra de las principales mejoras en esta versión fue la capacidad de gestión de pantallas de mayor tamaño y resolución. Como importante novedad, también incorpora por primera vez soporte para NFC (Near Field Communication).

Ilustración 20: Logotipo Android 2.3 GingerBread

- Android 3.0: Honeycomb

Android 3.0 llegó el 23 de febrero de 2011, especialmente pensado para tablets, con diversas mejoras para adaptar el sistema a pantallas más grandes. Se plantea como un sistema paralelo al ya existente para teléfonos, pero fracasa, dada la dificultad de mantener dos plataformas independientes simultáneamente.

Ilustración 21: Logotipo Android 3.0 HoneyComb

- Android 4.0: Ice Cream Sandwich

El 19 de octubre de 2011 llega Android 4.0 Ice Cream Sandwich, significando un importante paso en la evolución de Android, no solo renovando profundamente su interfaz de usuario con el nuevo diseño Holo, sino también integrando el sistema operativo en su versión para Tablets y Smartphones. Con Ice Cream Sandwich desaparecen los botones hardware de menú, inicio y atrás, reemplazándose por botones táctiles integrados en la pantalla.

Ilustración 22: Logotipo Android 4.0 Ice Cream Sandwich

- Android 4.1, 4.2, 4.3: Jelly Bean

Jelly Bean fue lanzado en Julio de 2013. Uno de los principales propósitos de esta versión fue dotar al sistema de una interfaz realmente ágil y rápida. Para conseguirlo se partió de un núcleo Linux 3.0.31 y se aplicaron una serie de mejoras que permitió aumentar hasta 60 frames por segundo las transiciones del interfaz gráfico, aportando una gran sensación de fluidez en el manejo.

Otra de las grandes novedades de esta versión fue la incorporación de Google Now, un asistente con funciones predictivas para competir con el Siri de Apple.

En Android 4.2 se introducen nuevas funciones como PhotoSphere, aplicación que permite tomar imágenes esféricas y Gesture Typing, un nuevo sistema para interactuar con el teclado por medio de trazos.

Ilustración 23: Logotipo Android 4.X Jelly Bean

- Android 4.4: KitKat

Android 4.4 (octubre de 3013) incorpora un Google Now mejorado, una nueva aplicación de llamadas, gestión de impresoras e impresión de documentos y una función "podómetro" para contar los pasos al caminar. También incorpora numerosas mejoras en estética.

Ilustración 24: Logotipo Android 4.4 KitKat

- Android 5.0: Lollipop

Lolipop aparece en noviembre de 2014. Trae consigo una de las mayores mejoras en la interfaz de usuario que se hayan hecho, con la incorporación de Material Design, un nuevo lenguaje de diseño desarrollado por Google. Material Design se adapta a los distintos dispositivos en una apariencia de capas planas, sombras de unas sobre otras, y un movimiento ahora más fluido que nunca. Permite la creación de usuarios en el equipo, para proteger los datos personales en caso de compartir el dispositivo e incorpora numerosas mejoras para facilitar el acceso a la configuración de los dispositivos desde la barra de notificaciones.

Aparte de las mejoras visuales, incorpora soporte para procesadores de 64 bits, **tanto en núcleos ARM como x86 o MIPS**, manteniéndose compatible también con los procesadores de 32 bits. Incluye la nueva máquina virtual de Android, ART, que promete grandes mejoras en el rendimiento.

Ilustración 25: Logotipo Android 5.0 Lollipop

- **Android 5.2 M ¿?:**

La siguiente versión de Android, que aún no tiene nombre, (solo sabemos que empezará por M, y se rumorea que se identificará con Muffin o M&M's) será lanzada a finales de 2015. Entre otras cosas pretende mejorar características como el pago con móvil, la interactuación entre aplicaciones, navegación por pestañas personalizadas, lectura de huellas dactilares, USB 3.1 Type-C y mejor gestión de la batería.

El primer dispositivo que recibirá esta actualización será el Google Nexus 5 fabricado por la coreana LG.

2.5 HERRAMIENTAS DE DESARROLLO PARA APLICACIONES MOVILES

Hoy en día existen multitud de posibilidades para desarrollar aplicaciones móviles, cada una tiene sus ventajas y sus inconvenientes.

Para empezar, cabe distinguir dos grandes grupos de "sistemas" de desarrollo de aplicaciones móviles, los "nativos" y los basados en tecnología WEB. También podemos hablar de aplicaciones hibridas, que combinan ambas técnicas.

2.5.1 APLICACIONES NATIVAS

El software nativo, **está desarrollado en el lenguaje de programación nativo del S.O**. En el caso de IOS, su lenguaje nativo es Objective-C, el de Android es JAVA y C# para los sistemas operativos de Microsoft. Programar en el leguaje nativo consigue que la aplicación pueda obtener mejor rendimiento del equipo y pueda acceder a todos los recursos del mismo, tanto a los de alto nivel, como a los de más bajo nivel.

- **Ventajas**
  - Mayor rendimiento, aplicaciones más rápidas.
  - Acceso completo al hardware del terminal (cámara, sensores, lista de contactos, etc.)
  - Mantenimiento más sencillo
  - Aplicaciones más seguras. Llevan firmas de seguridad.
- **Inconvenientes**
  - Mayor dificultad para desarrollo multiplataforma

- Requieren un SDK específico para cada plataforma.

2.5.2 APLICACIONES WEB

El software "no nativo" está desarrollado con las técnicas actuales de desarrollo de aplicaciones y páginas WEB, como HTML5, CSS y JavaScript. Al ser aplicaciones "interpretadas" por un navegador WEB pueden funcionar con independencia de la plataforma, al igual que vemos cualquier página web en cualquier plataforma. El aspecto más desfavorable de este tipo de aplicaciones es que no podrán aprovechar al máximo las posibilidades del dispositivo.

En el caso de Firefox-OS, el sistema "nativo" de programación es mediante HTML5, CSS y Javascript, por lo que no da lugar a hacer este tipo de distinción.

- **Ventajas**
  - Multiplataforma, fácilmente adaptable de un sistema a otro.
  - Lenguajes de programación estándar y ampliamente conocidos
  - Fácil mantenimiento

- **Inconvenientes**
  - Menor rendimiento, menor velocidad.
  - Acceso limitado al hardware del terminal.
  - Menor seguridad.

2.5.3 APLICACIONES HIBRIDAS

Las aplicaciones hibridas, por una parte contienen una aplicación Web, que corre dentro de una navegador "nativo" y aparte pueden incluir otros elementos nativos que permitan acceso al hardware y a la plena potencia del dispositivo. Normalmente el interfaz de usuario esta realizado en HTML5 y las capas más bajas de la aplicación en lenguaje nativo.

Existen herramientas como Cordova o PhoneGap que permiten encapsular aplicaciones web dentro de una aplicación nativa, dando la sensación de tratarse de aplicaciones nativas. Con este tipo de herramientas es sencillo realizar una aplicación para Android, IOS y Windows con apenas cambiar alguna configuración.

- **Ventajas**
  - Aprovecha las ventajas de ambas técnicas

  - Diseño ágil y sencillo de la interfaz de usuario sin renunciar al control del HW de más bajo nivel.
- **Inconvenientes**
  - No es tan fácil de portar entre plataformas, al tener partes en lenguaje "nativo"
  - Más difíciles de mantener que las que son puramente WEB.

2.5.4 ANDROID SDK

Android SDK es el conjunto de herramientas oficial proporcionado por Google. Incluye un depurador, un emulador de dispositivos Android, todas las librerías, ejemplos y documentación. Está disponible para Linux, Mac OS y Windows.

El IDE (Entorno de desarrollo integrado) oficialmente soportado era Eclipse, pero actualmente Google apuesta por el nuevo entorno "Android Studio" desarrollado por "InteliJ IDEA". Aun así a día de hoy la integración con Eclipse sigue siendo plenamente funcional.

El SDK, además de lo ya mencionado, proporciona las siguientes herramientas:

- NDK: "Native Development Kit". Se trata de un compilador y librerías en C, y C++ para desarrollo de partes de aplicaciones a bajo nivel. Es muy útil para optimizar procesos que requieran gran velocidad, como tratamiento de gráficos, simulaciones físicas, tratamiento de señal, etc. Esto es debido a que estas rutinas se ejecutan directamente en el procesador sin tener que ser "interpretadas" por la máquina virtual de Java. Las librerías escritas en C y C++ conectan con el Java a través del interfaz JNI (Java Native Interface)

- ADB: "Android Debug Bridge". Es una herramienta de depuración y comunicación con el dispositivo Android basada en línea de comando.

- FastBoot: Fastboot o arranque rápido es un protocolo utilizado para acceder al sistema de ficheros del terminal en su memoria flash desde el PC a través del puerto USB. Para ello debemos arrancar el dispositivo pulsando una secuencia de botones (normalmente con los botones de volumen y encendido) para dejarlo

en modo “boot”. De esta forma se puede actulizar o sustituir el firmware del terminal.

- Android Open Accessory Devlelopment Kit: Este sistema da soporte para que un terminal Android pueda comportarse como “host”, manejando a periféricos USB o bien como “accesorio” comportándose como esclavo de otro dispositivo “host”.

2.5.5 ANDROID STUDIO

Actualmente Android Studio ha reemplazado a Eclipse como IDE oficial para el desarrollo de aplicaciones nativas. Está desarrollado por InteliJ IDEA, e incorpora numerosas mejoras en el editor de código y en el diseño de GUI específicamente creadas para el desarrollo de aplicaciones para Android. El principal problema que presenta frente a el uso de Eclipse como IDE, es que Android Studio tiene un comportamiento más lento, necesitando una maquina más potente y con más RAM que Eclipse para poder desarrollar con soltura.

El IDE es inteligente y predictivo facilitando enormemente la escritura de código. Permite insertar plantillas de los elementos más utilizados y se integra con GITHUB para poder importar ejemplos desde allí.

Android Studio permite crear emuladores a medida, para cualquier tamaño de pantalla.

Incorpora GRADLE como motor para crear los paquetes compilados.

2.5.5.1 INSTALACION DE ANDROID STUDIO

La instalación del IDE no puede ser más sencilla. Simplemente hay que entrar en la página oficial, http://developer.android.com y pulsar en GET SDK. Descargaremos la versión correspondiente a nuestro sistema operativo y se instalará.

Una vez instalado el IDE, automáticamente se descargará todos los API necesarios, pudiendo elegir para que versiones de Android queremos desarrollar.

La descarga de todos los elementos lleva bastante tiempo y ocupa en torno a 1,5 Gb de espacio en el disco duro.

2.5.5.2 PRIMEROS PASOS EN ANDROID STUDIO

Lo primero que haremos al empezar con Android Studio es ir a File->New Project…

Nos aparecerá una ventana donde podremos elegir el nombre de la aplicación, el nombre de dominio y la carpeta donde se creará el proyeto.

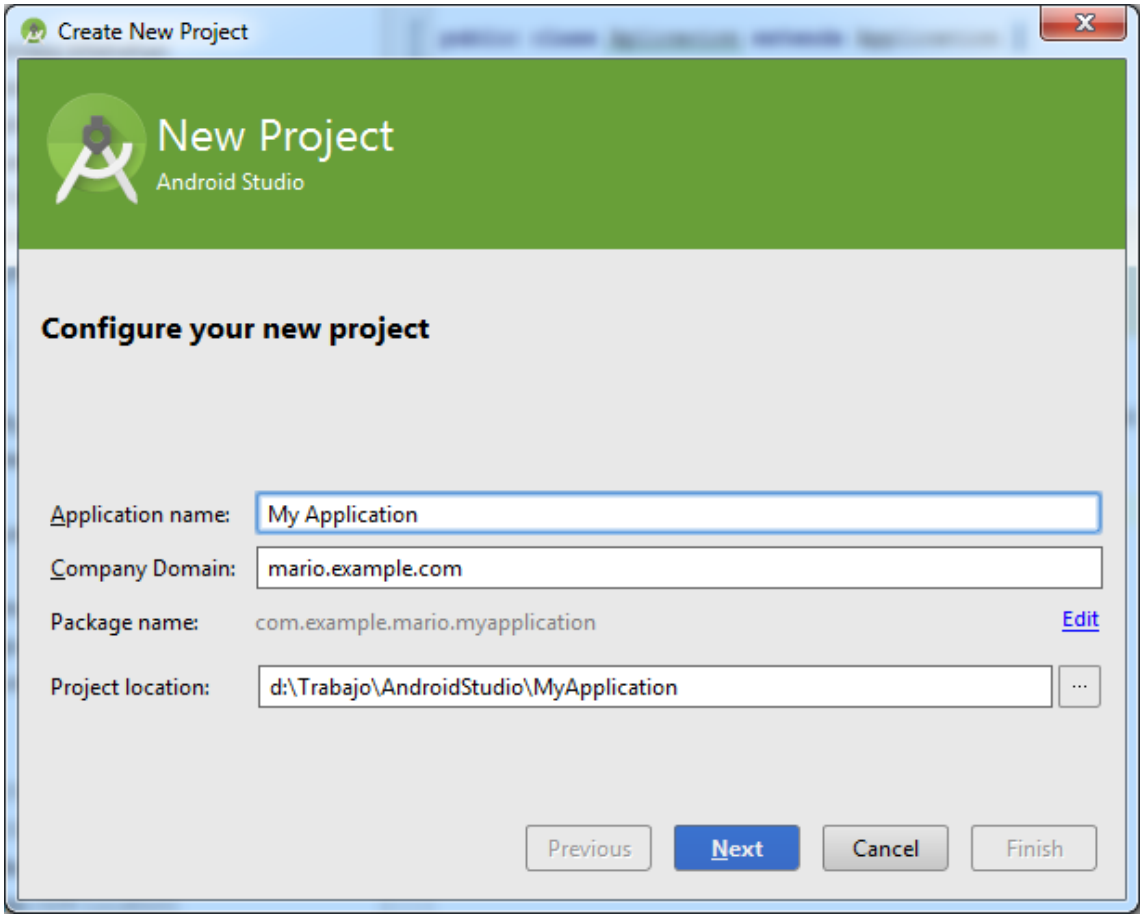

Ilustración 26: Creando el proyecto, paso 1

En la siguiente ventana, especificamos el nivel de api mínimo con el que queremos trabajar. Con API's más antiguas, nuestra aplicación será compatible con mayor número de dispositivos, pero no podremos aprovechar las características de las API mas modernas.

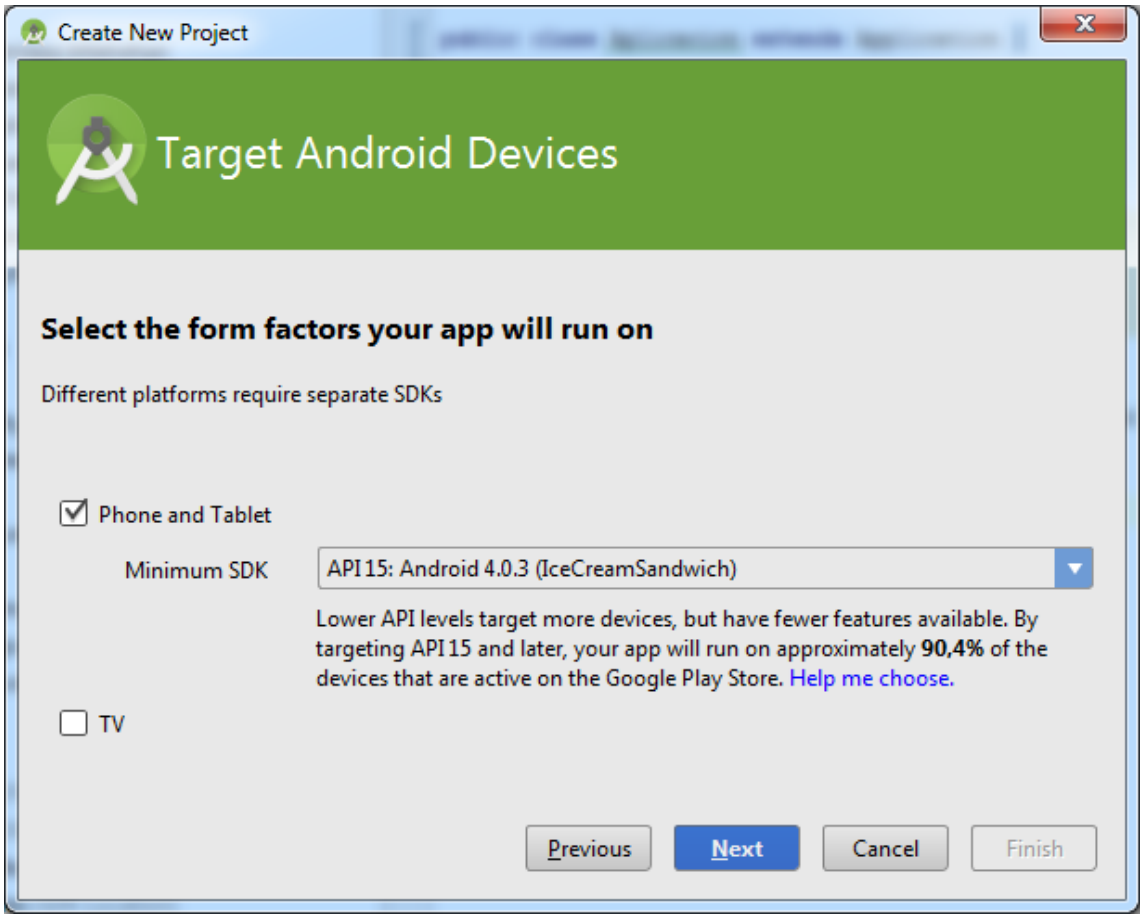

Ilustración 27: Creando el proyecto, paso 2

Después de elegir el API y pulsar Next, el asistente nos permite elegir la plantilla por defecto de nuestra Actividad principal. Podemos elegir entre una pantalla en blanco vacia hasta una pantalla con mapa, con menus de distintos tipos, con diversos controles ya preestablecidos…

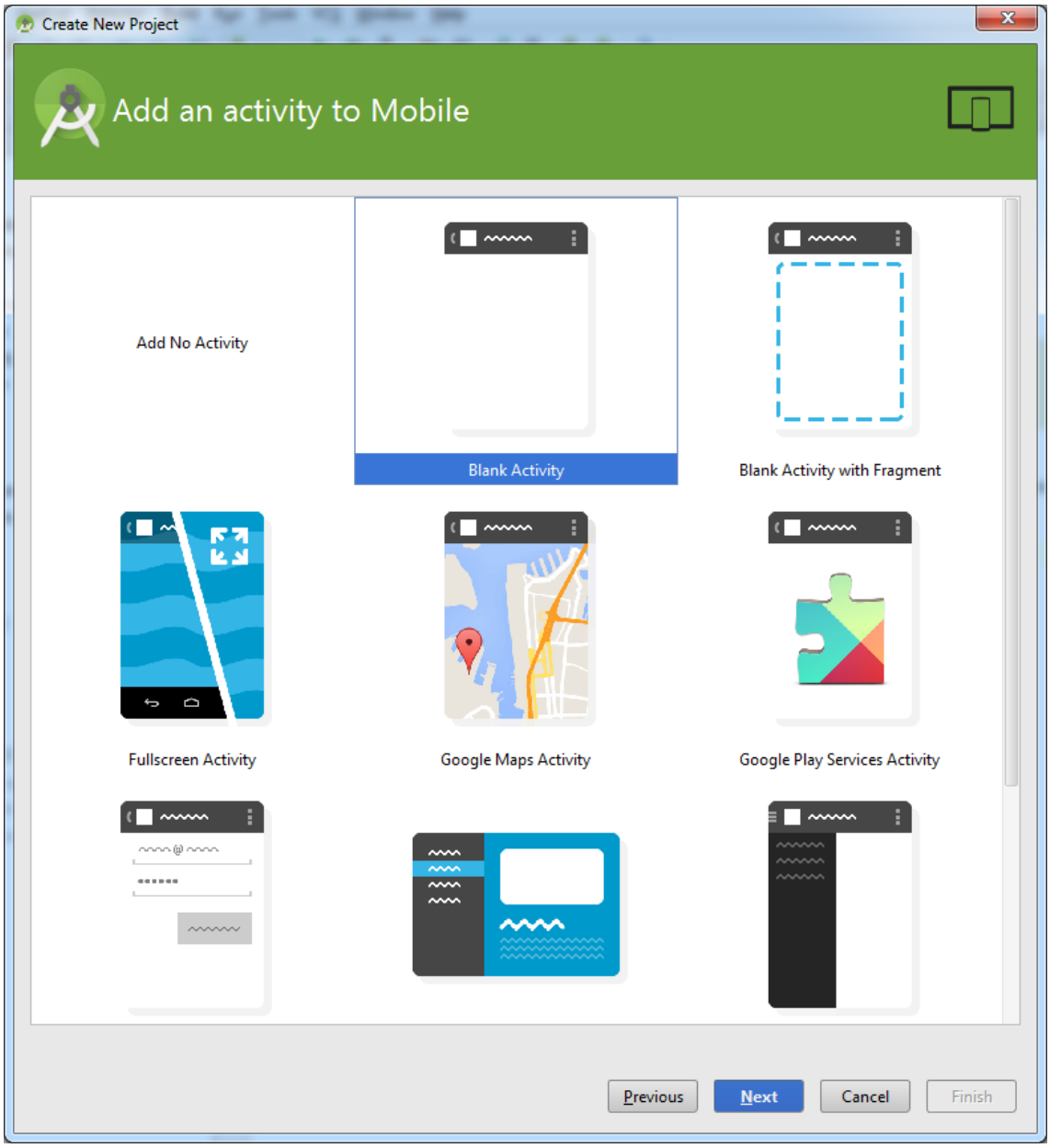

Ilustración 28: Creando el proyecto, paso 3

Por último, le damos nombre a cada uno de los elementos de la Actividad, como la clase Java principal, el layout (fichero XML que define el GUI), el título a mostrar y el nombre del menú principal.

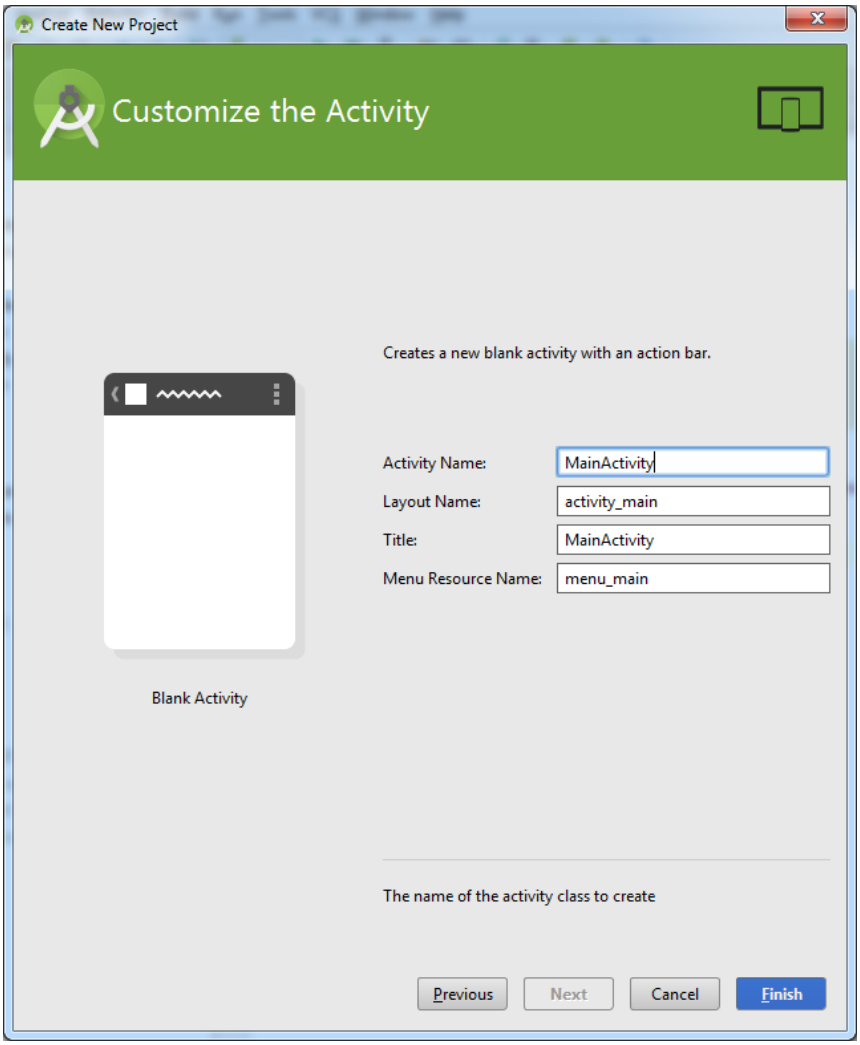

Ilustración 29: Creando el proyecto, paso 4.

Pulsamos “Finish” y esperamos un momento a que Gradle haga su trabajo

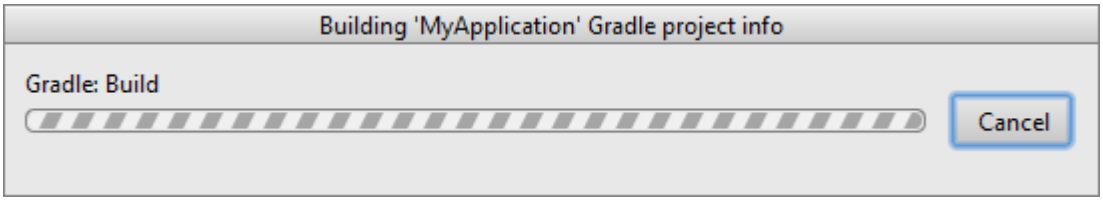

Ilustración 30: Esperando a que Gradle componga el entorno de trabajo.

Ya tenemos listo el entorno de desarrollo del proyecto. Con las pestañas superiores podemos cambiar el documento a mostrar. En la imagen inferior vemos el editor “Visual” del GUI, con el que podemos añadir y organizar los distintos elementos del interfaz de usuario de una forma visual. También se puede trabajar sobre el GUI en

modo texto, ya que todo él está definido por medio de un fichero XML. En las pestañas inferiores cambiamos de un modo a otro.

En el bloque de la izquierda vemos la estructura del proyecto, con sus archivos, y a la derecha, en este caso tenemos la estructura de la interfaz gráfica, que se organiza por medio de "layouts".

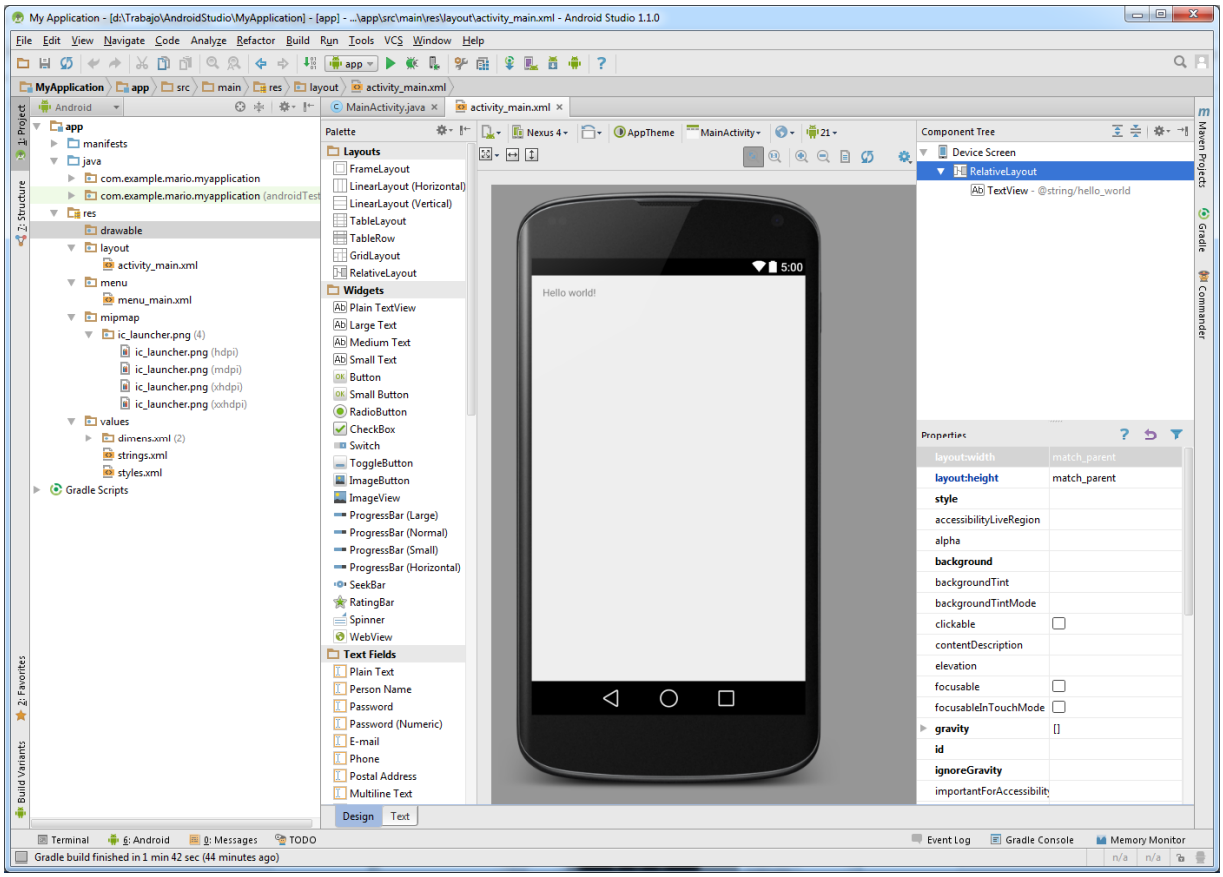

Ilustración 31: Entorno de trabajo. Vista de Diseño

2.5.5.3 ESTRUCTURA DE UN PROYECTO

En toda aplicación de Android se establece una estructura general del proyecto que consta de varios grupos:

- **Manifiesto:** Esta carpeta contiene el archivo AndroidManifest.xml. En él se declaran todos los aspectos generales de la aplicación, como el API mínimo, los permisos que requiere la aplicación, etc…

- **Clases de Java**. Contiene todo el código en Java del proyecto.

- **Recursos:** La carpeta "RES" contiene todos los recursos necesarios para el funcionamiento de la aplicación. A su vez se divide en varias subcarpetas:

    - Drawable: Contiene elementos gráficos.
    - Layout: Contiene los XML que definen la estructura de los distintos interfaces gráficos de la aplicación.
    - Menu: Contiene la definición de los menús, en formato XML.
    - Mipmap: Carpeta que alberga los iconos de la aplicación en distintas resoluciones.
    - Values: Esta carpeta contiene diversos ficheros XML que almacenan valores constantes, ya sean textos, estilos, colores, dimensiones, ect…

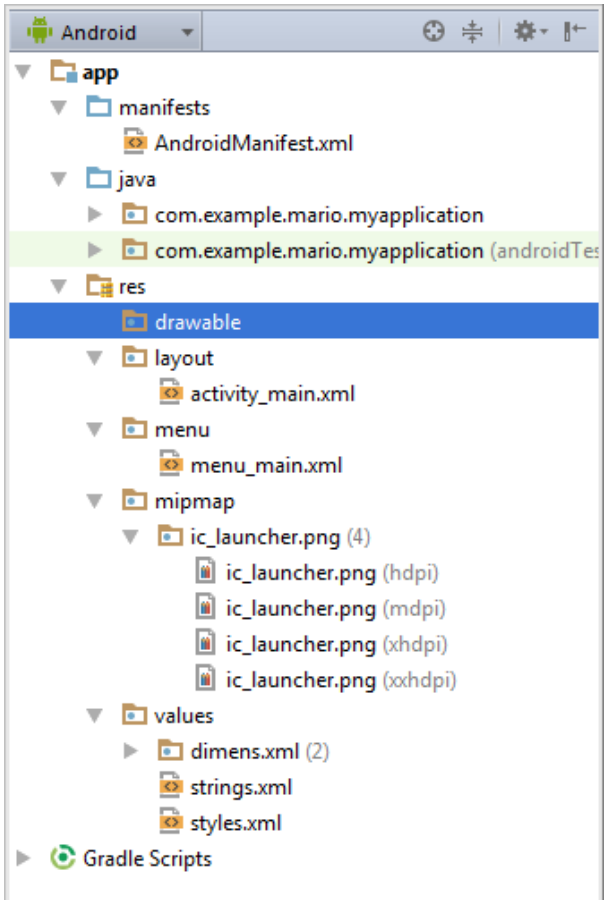

Ilustración 32: estructura de un proyecto

2.6 TRABAJOS PREVIOS

Existen numerosos sistemas comerciales de localización y gestión de flotas que se ofrecen tanto a empresas como a particulares para tener un control de sus vehículos. Partiendo de esta idea, y haciéndolo extensivo al ámbito industrial o agrícola, surge la posibilidad de añadir distintos sensores o captadores que aprovechan la comunicación GPRS de los equipos de localización para enviar cualquier tipo de medidas al servidor, junto con la información proporcionada por el GPS.

En cuanto a la interfaz de usuario, la gran mayoría de aplicaciones existentes suelen tener una interfaz de usuario consistente en una aplicación para PC o una página web.

Existen trabajos previos con la misma filosofía, pero al no tratarse de aplicaciones de consumo, sino para un entorno industrial y profesional, cada uno cuenta con características que le hacen singular. Este trabajo pretende ser una demostración práctica de un sistema flexible y aplicable a cualquier tipo de entorno simplemente utilizando los sensores adecuados.

Como referentes de gestión de flotas o localización de vehículos podemos destacar los siguientes productos:

- **TomTom Telematics:** Se trata de un sistema de control de flotas profesional, que entre otras características, incorpora un control del comportamiento del conductor basado en acelerómetros.
  http://business.tomtom.com/es_es/fleet-management/control-de-flotas-gps/
- **Detector:** Uno de los pioneros en localización de vehículos robados, ahora también ofrece servicios de gestión de flotas personalizados.
  http://www.detectorseguridad.com/gestion-y-localizacion-de-flotas/

En cuanto a otros sistemas de telemetría, orientados a otros sectores podríamos poner como ejemplos:

- **General Electric Healthcare:** Se trata de un sistema de telemetría médico que permite tener monitorizados a los pacientes sin necesidad de que estén en un lugar concreto.
  http://www3.gehealthcare.es/es-es/productos/categorias/monitores_de_paciente/telemetria/apexpro
- **GeoAgris:** Proporciona sistemas de telemetría y telecontrol para explotaciones agrícolas.
  http://geoagris.com/soluciones/telemetria-agriexplorer/agriexplorer-riego/

2.7 APLICACIÓN SERVIDOR "SERVI-SENSOR"

2.7.1 DESCRIPCIÓN GENERAL

La aplicación que corre en el servidor es la encargada de:

- Recibir la información enviada por los equipos móviles
- Almacenar la información recibida en la base de datos
- Recibir las peticiones de las aplicaciones Android
- Procesar las estas peticiones y devolver los datos solicitados.

El servidor teóricamente podrá gestionar simultáneamente hasta 64K conexiones, que se corresponden al máximo número de clientes que soporta un puerto TCP. En la práctica, el máximo número de conexiones habría que determinarlo en función de la cantidad de RAM disponible y de la potencia del procesador. Por ello elegiremos un servidor virtual que permita escalabilidad, para poder asignar recursos en caso de ser necesarios.

También se debería tener en cuenta la cantidad de información almacenada en la base de datos. Se crearan "scripts" en el servidor para que archiven los datos más antiguos en otros medios cuando se alcance un límite de almacenamiento establecido.

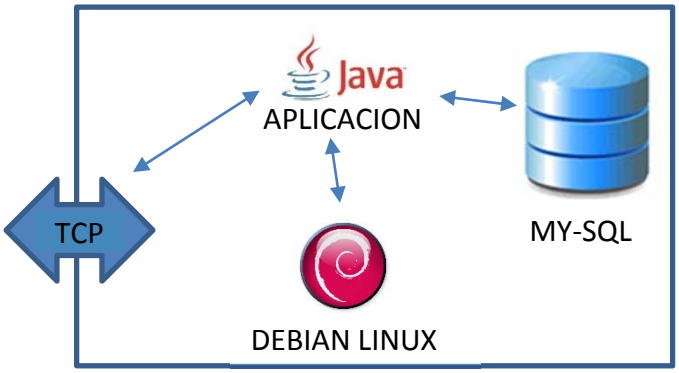

Ilustración 33: Elementos del servidor

2.7.2 COMPORTAMIENTO DE LOS EQUIPOS REMOTOS

Los equipos remotos utilizados para desarrollar este proyecto son dispositivos autónomos, dotados de procesador, memoria, modem GPRS y receptor GPS. Los dispositivos llevan una aplicación embebida que permite configurar los distintos

parámetros de conexión a la red GPRS, la dirección IP y el puerto del servidor, los tiempos a los que debe enviar información, etc…

Los equipos tratarán de mantener continuamente una conexión con un puerto TCP del servidor, y enviarán en los intervalos de tiempo establecidos un paquete de datos con la información obtenida del GPS y los sensores, que es la siguiente:

- Cabecera o identificador de trama.
- IMEI del modem (se usa como identificador único del dispositivo)
- Alias o nombre "amigable" del equipo
- Fecha y hora UTC del momento en que se toma la muestra.
- Latitud y longitud.
- Altitud sobre el nivel del mar.
- Velocidad en km/h
- Rumbo
- Estado señal GPS
- Valor sensores (mV).

Un ejemplo de una trama enviada podría ser el siguiente:

```
#SENSOR#353234025641513#TRACKER_001#2013/06/18,15:59:22,40.2518133,N,0
03.6946650,W,00624,001,245.67,3,13450,00224
```

2.7.3 SCRIPTS Y CONFIGURACION DEL SERVIDOR

Antes de describir el funcionamiento de la aplicación en sí misma, paso a detallar algunos procesos de instalación y configuración previos necesarios para que el servidor funcione con los elementos requeridos.

2.7.3.1 INSTALACION DE MYSQL Y JAVA

Para que la aplicación pueda ser ejecutada en el servidor, previamente se debe instalar el motor de MySQL y el JAVA JRE (Java Run Time Environment) para poder ejecutar las aplicaciones en Java. Trabajaremos sobre un servidor virtual con Debian Linux.

Para instalar MySQL abrimos una sesión en el servidor y ejecutamos:

```
apt-get install mysql-server mysql-client
```

Este comando descarga el MySql del repositorio y lo instala automáticamente.

Para instalar JAVA debemos descargar la versión adecuada para nuestro servidor de la página de ORACLE y a continuación seguir los siguientes pasos.

- Crear el directorio donde se instalará Java en /opt

```
$ sudo mkdir -pv /opt/java_se/java/
```

- Descomprimir java en el directorio creado. Usaremos la versión "jre-7u5-linux-i586.tar.gz" en este caso, pero deberá descargase la más actualizada en el momento de la instalación.

```
$ sudo tar xzvf jre-7u5-linux-i586.tar.gz -C /opt/java_se/java/
```

- Crear un enlace simbólico llamado "JAVA-ORACLE" al contenido del directorio de java. Esto facilitará la actualización de Java en el futuro.

```
$ sudo ln -sv /opt/java_se/java/* /opt/java_se/JAVA-ORACLE
```

- Crear el enlace simbólico al ejecutable de java en update-alternative a través de JAVA-ORACLE

```
$ sudo update-alternatives --install /usr/bin/java java
/opt/java_se/JAVA-ORACLE/bin/java 1200
```

- Elegimos en “update-alternatives” nuestra versión de java (en nuestro caso elegimos la que viene por defecto)

```
$ sudo update-alternatives --config java
```

- Aparecerá un texto para escoger la versión de Java

```
Existen 3 opciones para la alternativa java (que prove /usr/bin/java).
Selección Ruta Prioridad Estado
 ------------------------------------------------------------
 * 0   /opt/java_se/JAVA-ORACLE/bin/java       1200    modo automático
   1   /opt/java_se/JAVA-ORACLE/bin/java     1200    modo manual
   2   /usr/lib/jvm/java-6-openjdk/jre/bin/java    1061    modo manual
   3   /usr/lib/jvm/java-6-sun/jre/bin/java          63    modo manual
```

```
 Pulse <Intro> para mantener el valor por omisión [*] o pulse un
número de selección: <intro>
```

En este caso únicamente hay que presionar <intro>. Si no, el número que corresponda la version de JAVA-ORACLE que se desee.

- Añadir java a la variable PATH, preferiblemente para todos los usuarios
  Para agregar java al PATH de TODOS los usuarios editamos el archivo /etc/bash.bashrc con un editor de textos, por ejemplo nano.

```
$ sudo nano /etc/bash.bashrc
```

Hay que añadir la siguiente línea al final del archivo.

```
export PATH=$PATH:/opt/java_se/JAVA-ORACLE/bin
```

Después cerrar el archivo guardando los cambios. (CRTL+X, Y en caso de usar nano)

- Verificación de la versión de java instalada

```
$ java -version
```

El resultado será:

```
java version "1.7.0_05"
Java(TM) SE Runtime Environment (build 1.7.0_05-b05)
Java HotSpot(TM) Client VM (build 23.1-b03, mixed mode)
```

2.7.3.2 SCRIPTS PARA INICIAR Y DETENER EL SERVICIO

Al compilar el código de la aplicación servidor obtendremos un fichero ServiSensor.jar, que dejaremos en una carpeta del servidor, como por ejemplo:

```
/home/ServiSensor/
```

Para iniciar la aplicación java como un servicio crearemos un script como el siguiente, llamándolo start.sh y dejándolo en /home/ServiSensor/

```
#! /bin/sh
```

```
now=$(date +"%Y%m%d")
nohup java -jar /home/ServiSensor/ServiSensor.jar >>
/home/ServiSensor/log$now.txt 2>>/home/ServiSensor/err$now.txt <
/dev/null &
PID=$!
echo $PID >/home/ServiSensor/pid.txt
```

Lo que hace el script es lanzar la aplicación java, redirigiendo la salida Stdio y stdError a sendos ficheros llamados logAAAAMMDD.txt y errAAAAMMDD.txt. La aplicación se lanza en segundo plano.

La línea se lanza con “nohup” para que no se cierre la ejecución al cerrar el terminal, y con el & al final, para que se ejecute en segundo plano.

Además, el PID del proceso se guarda en un fichero pid.txt para poder matarlo directamente desde otro script en caso de ser necesario.

Para ello crearemos el script stop.sh de esta forma:

```
#! /bin/sh
PID=$(cat /home/ServiSensor/pid.txt)
kill $PID
```

Simplemente lee el PID del fichero y hace un KILL deteniendo así el proceso.

Tambien crearemos el script restart.sh que es la suma de los dos anteriores, primero detiene el proceso y luego lo vuelve a lanzar:

```
#! /bin/sh
PID=$(cat /home/ServiSensor/pid.txt)
kill $PID
now=$(date +"%Y%m%d")
nohup java -jar /home/ServiSensor/ServiSensor.jar >>
/home/ServiSensor/log$now.txt 2>>/home/ServiSensor/err$now.txt <
/dev/null &
PID=$!
echo $PID >/home/ServiSensor/pid.txt
```

Para que la aplicación se lance cada vez que arranque el servidor, editamos el fichero /etc/rc.local y añadimos la siguiente línea:

```
/home/ServiSensor/start.sh
```

2.7.4 DISEÑO DE LA BASE DE DATOS

Para que la aplicación funcione hay que crear un esquema en MySQL llamado "ServiSensor" con las siguientes tablas: (se pueden crear con PHPMyAdmin si el servidor dispone de ello)

2.7.4.1 TABLA USUARIOS

La tabla usuarios contiene la relación de usuarios dados de alta en el sistema, sus claves y su nivel de permisos dentro del sistema.

CAMPO	TIPO	DESCRIPCIÓN
User	varchar(40) NOT NULL	Nombre de usuario
Password	varchar(12) NOT NULL	Password
Level	int(11) NOT NULL	Nivel de permisos del usuario
Alias	varchar(40) NULL	Alias
usuarioID	int(11) NOT NULL	Id numérico asociado

Tabla 13: Diseño tabla USUARIOS

2.7.4.2 TABLA USEREQUIPO

La tabla "USEREQUIPO" relaciona cada usuario con los distintos posibles equipos que puede monitorizar.

CAMPO	TIPO	DESCRIPCIÓN
ID	int(11) NOT NULL	Id auto numérico
usuario	varchar(30) NOT NULL	Nombre usuario
alias	varchar(30) NOT NULL	Alias
usuarioID	int(11) NOT NULL	Id numérico del usuario
IMEI	varchar(20) NOT NULL	Imei del equipo asociado al usuario

Tabla 14: Diseño tabla USEREQUIPO

2.7.4.3 TABLA EQUIPOS

La tabla “equipos” contiene la información más reciente disponible por cada dispositivo. De esta tabla se obtendrá la información de seguimiento en tiempo real. La información será enviada periódicamente por los equipos remotos. La aplicación “Servisensor” será la encargada de procesar dicha información y rellenar las tablas.

CAMPO	TIPO	DESCRIPCIÓN
IMEI	varchar(20) NOT NULL	Código IMEI del modem
ALIAS	varchar(20) NULL	Nombre amigable del equipo
ult_conexion	datetime NULL	Fecha y hora ultima conexión
DATETIME	datetime NOT NULL	Fecha y hora ultima muestra
LAT	float NULL	Latitud en grados
LON	float NULL	Longitud en grados
SPEED	float NULL	Velocidad instantánea Km/h
SPEED_AVG	float NULL	Velocidad media en intervalo
SPEED_MAX	float NULL	Velocidad máxima en intervalo
DISTANCIA	int(11) NULL	Metros recorridos desde el punto anterior
COURSE	float NULL	Rumbo
ALTITUDE	int(11) NOT NULL	Altitud instantánea
LBS	varchar(8) NOT NULL	Identificador celda GSM
ICON	varchar(16) NULL	Icono asociado al equipo
MV_1	int(11) NULL	Lectura ADC: Temperatura
MV_2	int(11) NULL	Lectura ADC: V in
SesionID	int(10) unsigned NOT NULL	RFU
VBAT	int(11) NULL	Tension batería interna
CARGADOR	int(11) NULL	0 Desconectado / 1 Conectado Sin Cargar / 2 Conectado Cargando
NIVEL_BAT	int(11) NULL	% batería (cuando el cargador está desconectado)
VAL	int(1) NOT NULL	GPS valido: 0=NO 2=2D 3=3D

Tabla 15: Diseño tabla EQUIPOS

2.7.4.4 TABLA TRACKING

La tabla "tracking" contiene la información histórica de cada dispositivo. De esta tabla obtendremos los informes relativos a un intervalo de tiempo. La información será enviada periódicamente por los equipos remotos equipados con sensores y GPS.

CAMPO	TIPO	DESCRIPCIÓN
IMEI	decimal(20,0) NOT NULL	Código IMEI del modem
DATETIME	datetime NOT NULL	Fecha y hora de la muestra
LAT	float NULL	Latitud en grados
LON	float NULL	Longitud en grados
COURSE	float NULL	Rumbo
ALTITUDE	int(11) NOT NULL	Altitud instantánea
SPEED	float NULL	Velocidad instantánea
SPEED_AVG	float NULL	Velocidad media en intervalo
SPEED_MAX	float NULL	Velocidad máxima en intervalo
LBS	varchar(8) NOT NULL	Identificador de celda
MV_1	int(11) NULL	Temperatura
MV_2	int(11) NULL	Tensión Batería IN
VAL	int(1) NOT NULL	0 Señal GPS mala / <>0 GPS OK
SesionID	int(11) NOT NULL	RFU
DISTANCIA	int(11) NULL	Metros recorridos desde el punto anterior
SECUENCIA	int(10) NULL	
VBAT	int(11) NOT NULL	Tensión Batería interna
CARGADOR	int(11) NOT NULL	0 Desconectado / 1 Conectado Sin Cargar / 2 Conectado Cargando
NIVEL_BAT	int(11) NOT NULL	% batería (cuando el cargador está desconectado)
ALTITUDE_INC_POS	int(11) NULL	Altitud ganada (RFU)
ALTITUDE_INC_NEG	int(11) NULL	Altitud perdida (RFU)

Tabla 16: Diseño tabla TRACKING

2.7.5 DISEÑO DE LA APLICACIÓN SEVIDOR "SERVI-SENSOR"

A continuación se describe la aplicación desarrollada en Java que gestiona la información recibida y las peticiones a través del socket TCP y maneja la base de datos.

Esta aplicación consta de un hilo principal que permanece a la escucha del puerto, y múltiples hilos secundarios que realizan los procesos derivados de las peticiones recibidas.

2.7.5.1 HILO PRINCIPAL

La aplicación "ServiSensor", nada más arrancar, establece conexión con la base de datos y se abre un puerto TCP en modo servidor (a la espera de conexiones entrantes). La aplicación entra en un bucle en el que va chequeando si llega algún dato por el puerto. Cuando llega un paquete de datos se distingue si se trata de una trama proveniente de un equipo de sensores remoto, o es una petición de una aplicación móvil Android.

En el primer caso, se procesa directamente la información recibida y se almacena en la base de datos. En el segundo caso, se lanza un proceso para atender a la petición del dispositivo móvil y devolverle la información solicitada.

A continuación se muestra un diagrama de flujo describiendo el funcionamiento global de la aplicación servidor.

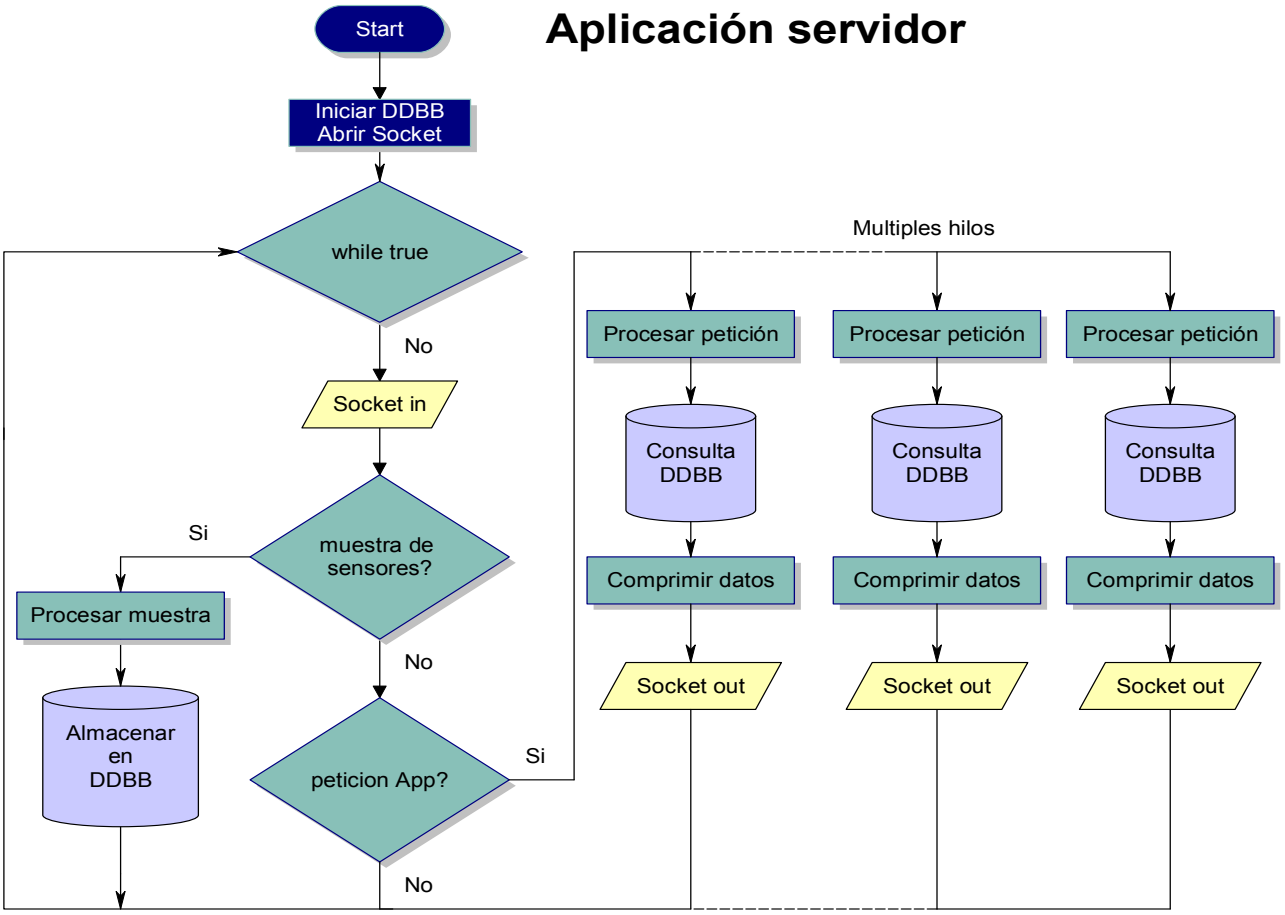

Ilustración 34: Diagrama de flujo Aplicación Servidor

2.7.5.2 HILOS SECUNDARIOS

Cada vez que llega una petición de una aplicación Android, se crea un nuevo hilo asociado al cliente que hace la petición. Este hilo procesara el comando y generara la respuesta que será entregada al cliente. El motivo de crear un hilo por cada petición es que puede tomar bastante tiempo en realizar la consulta y empaquetar los datos. Al correr en un hilo secundario, no detenemos al hilo principal, evitando desatender a otras peticiones o a información entrante de los sensores.

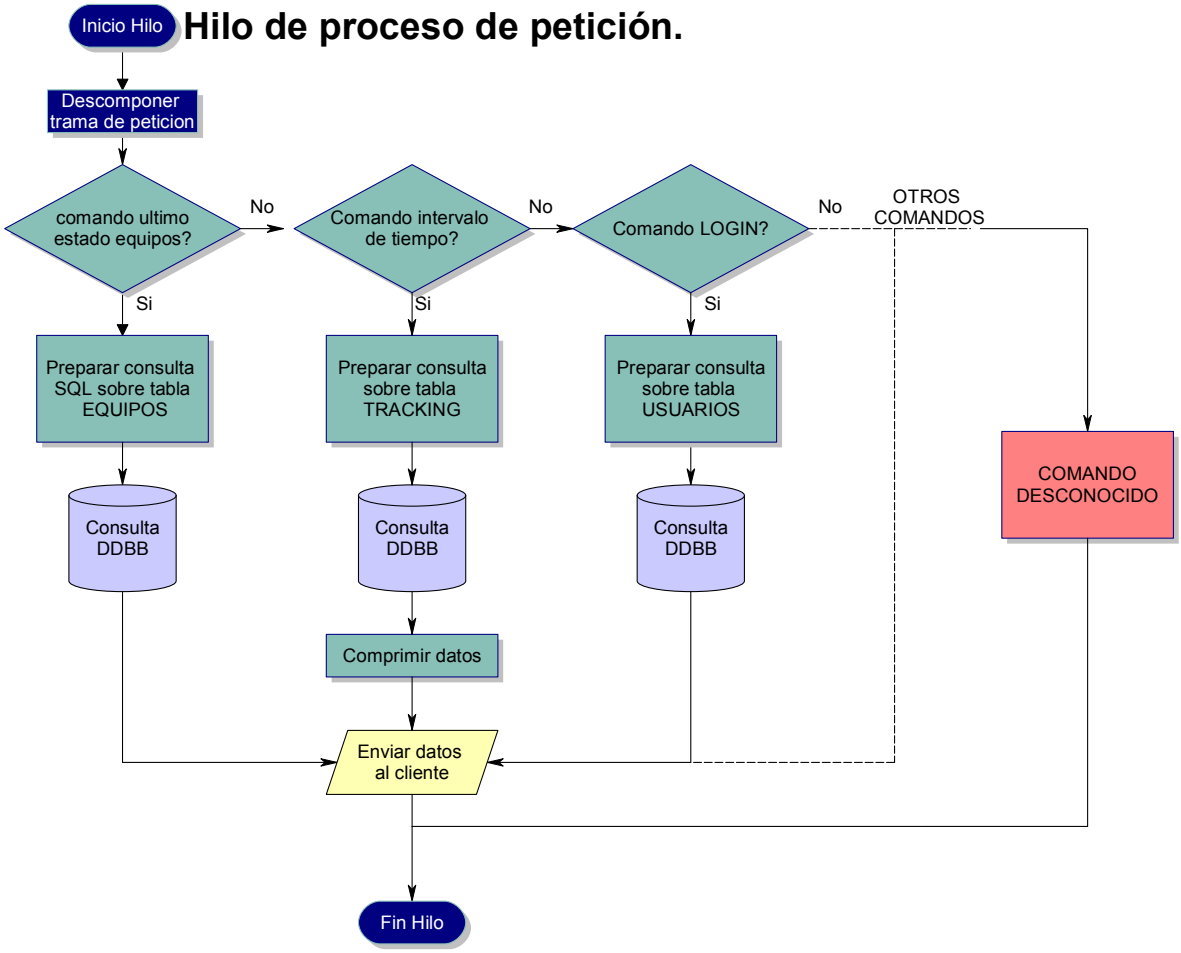

Ilustración 35: Diagrama de flujo de cada hilo secundario.

2.8 APLICACIÓN CLIENTE "SENSOR-TRACKING"

La aplicación cliente, totalmente desarrollada en Java con Android SDK, es la que se encargará de hacer las peticiones necesarias al servidor para mostrar la información captada y enviada por los dispositivos remotos.

La aplicación contiene cuatro "Actividades":

- **Login:** Actividad que se lanza al arrancar la aplicación para identificar al usuario con el servidor y permitir acceder al resto de actividades de la aplicación.
- **Mapa:** Es la actividad principal que está basada en un mapa de Google y es la que nos muestra la localización y los recorridos realizados por los localizadores.
- **Selección de Fechas-Horas:** Actividad que nos permite elegir un localizador y un intervalo de tiempo sobre el que realizar una consulta, ya sea de recorrido a

mostrar en el mapa o de valores de las mediciones para representar gráficamente.

- **Graficas:** Esta actividad nos mostrará los datos captados por los sensores mediante gráficas.

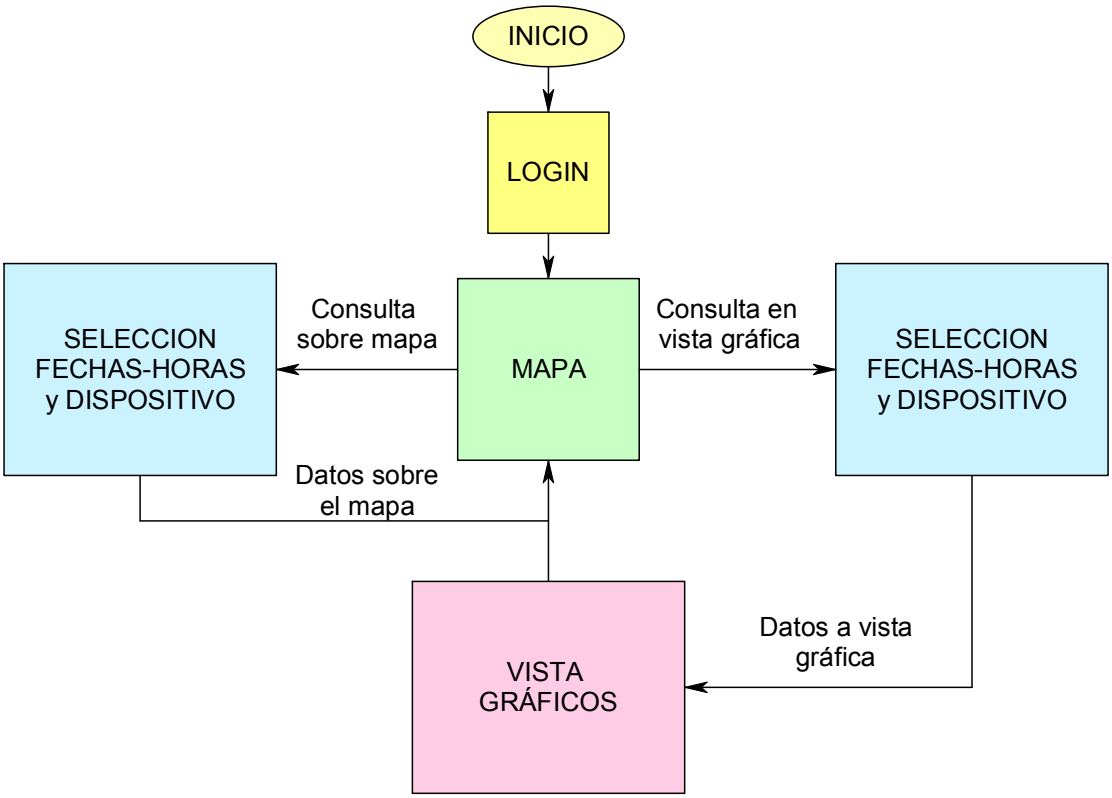

Ilustración 36: Estructura de la aplicación.

2.8.1 ACTIVIDADES DE LA APLICACIÓN

2.8.1.1 ACTIVIDAD "LOGIN"

LA actividad LOGIN es la primera que se muestra al iniciar la aplicación. En esta actividad se introduce el nombre de usuario y la clave para identificar al usuario. La aplicación envía una solicitud al servidor con los datos introducidos, y este verifica la validez y devuelve el resultado a la aplicación. Si la respuesta es positiva la aplicación lanza la actividad principal MAPA. Si no, muestra un error y da opción a volver a intentarlo.

Ilustración 37: Captura de pantalla de la actividad LOGIN

2.8.1.2 ACTIVIDAD "MAPA"

Es la actividad principal de la aplicación. En ella se muestran continuamente los equipos en tiempo real. Tocando cualquiera de ellos se muestra un desplegable con información del equipo en ese momento.

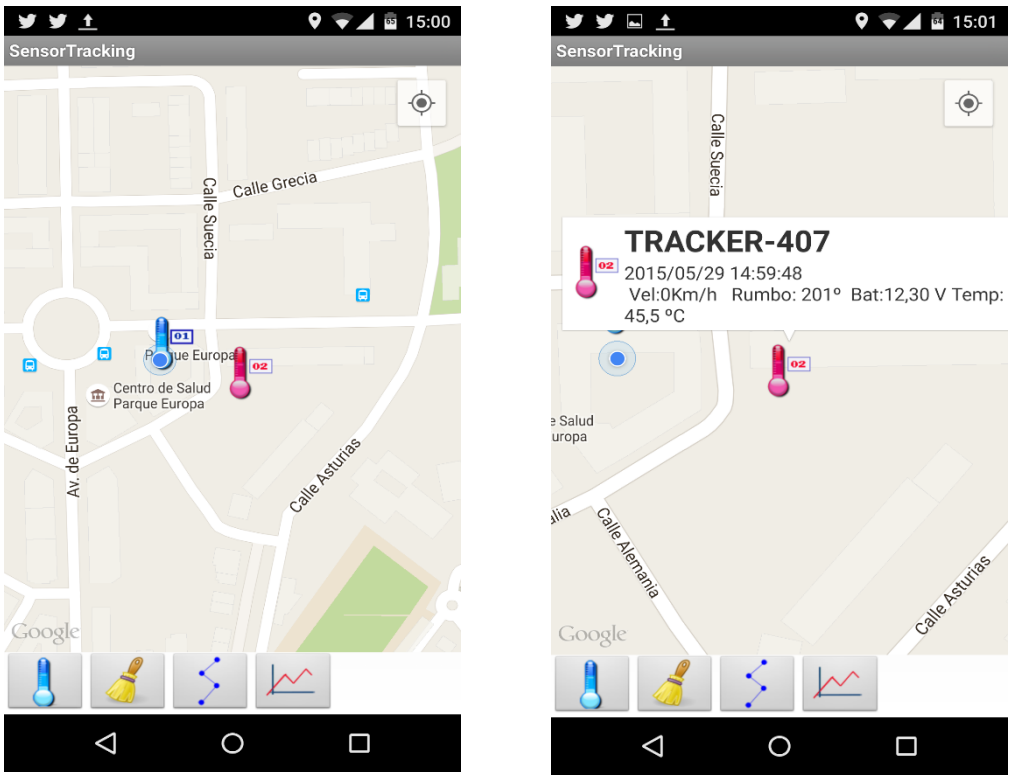

Ilustración 38: Capturas de la actividad MAPA

Desde esta actividad, accionando los botones de la parte inferior, se accede a las funciones para visualizar la información, representada en el mapa o representada en gráficas.

El primer botón nos permite seleccionar un equipo y localizarle el en mapa. Para ello debemos dejar pulsado el botón unos segundos y aparecerá una lista desplegable donde seleccionarle. Si hacemos una pulsación corta en el botón, el mapa ajustara su nivel de zoom para encuadrar a todos los dispositivos disponibles.

El segundo botón limpia el mapa de información que hayamos podido mostrar.

El tercer botón genera una consulta y la muestra sobre el mapa, mostrando los puntos capturados de un dispositivo durante un intervalo de tiempo. Pulsando en cualquiera de esos puntos se muestra un desplegable con la información de los sensores en el momento que fue capturada la muestra.

El cuarto botón muestra una representación gráfica de los parámetros medidos por el dispositivo. Estos son temperatura, velocidad y tensión de la batería.

Tanto el tercer botón como el cuarto dan paso a otra actividad para capturar los datos necesarios para la consulta, es decir, el dispositivo a y las fechas y horas de inicio y fin del intervalo a consultar.

En caso de ser una consulta sobre el mapa, al recibir los datos del servidor la aplicación vuelve a la actividad "mapa" donde representará los puntos en función de las coordenadas de cada muestra. Si es una consulta en modo de gráficas, se lanzará la actividad "Gráficas" donde se mostraran los datos gráficamente.

2.8.1.2.1 INTEGRACION DEL API DE GOOGLE MAPS

Para que nuestro proyecto pueda funcionar con el mapa de Google Maps, debemos especificarlo debidamente en la configuración del proyecto. Para ello debemos de elegir un proyecto tipo "Google API" y añadir la librería "google_play_services_lib" como se muestra en la imagen a continuación (para Eclipse)

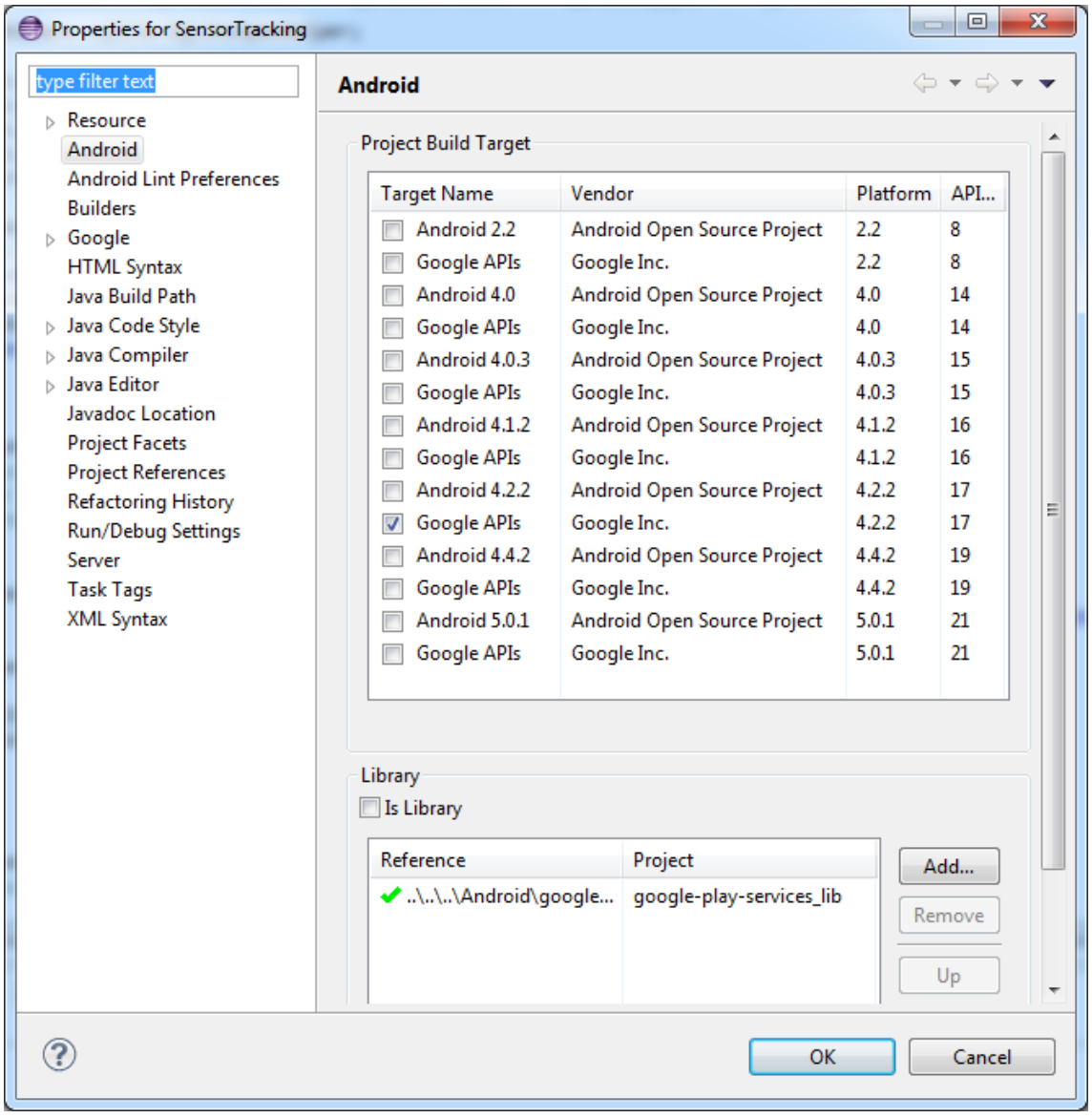

Ilustración 39: Configuración del proyecto para uso de Google Maps

Para poder utilizar el API de GoogleMaps, debemos obtener una clave desde la consola de desarrolladores de Google "Google Developers Console". Esta es una página web de google donde, previamente registrados y logeados con una cuenta Gmail, podemos gestionar los permisos y las claves de distintos API's y servicios que ofrece Google para nuestras aplicaciones (no sólo para Android). Aparte de el acceso a Google Maps, nos ofrecen multitud de servicios, como mensajería en la nube (Google Cloud Messaging), almacenamiento en la nube, redes sociales, traducción, etc.

Para registrar nuestra aplicación y que pueda "consumir" los servicios de Google, hay que hacer lo siguiente.

- Entramos a Google Developers Console, iniciando sesión con nuestra cuenta de Google.
- En el Apartado API, buscamos las relativas a Google Maps y accecemos a "Crecenciales"
- Debemos pulsar en "Crear clave nueva", y cuando nos pregunte que tipo de clave queremos crear, seleccionaremos "Clave de Android"
- Como las solicitudes a la API se envían directamente desde cada dispositivo Android, debemos identificar al proyecto para que nuestra aplicación sea identificada. Para ello debemos obtener un código SHA1 de nuestro "KeyStore". Para ello podemos hacerlo mediante línea de comando:

  ```
  keytool -list -v -keystore mystore.keystore
  ```

  indicándole cual es nuestro almacén de claves o Keystore para Android.

  También podemos encontrar esa clave en nuestro IDE de desarrollo. Por ejemplo, en Eclipse está en Window->Preferences, y en la ventana de preferencias, en Android->Build, como muestra la siguiente imagen. Ahí vemos la SHA1 fingerprint, que la debemos copiar para llevarla a la Consola de Google.

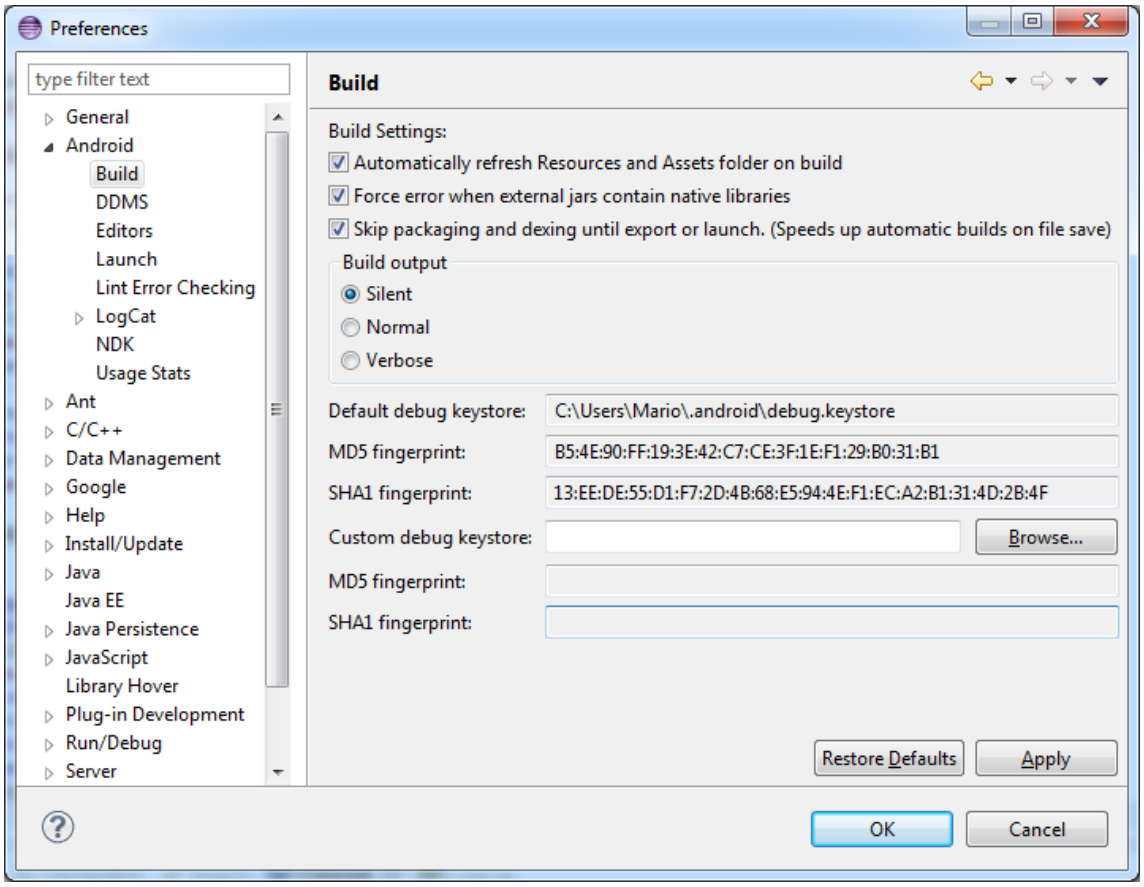

Ilustración 40: Obtención de la firma SHA1

Para obtener la clave SHA1 en Android Studio, solo es necesario crear un proyecto cuya actividad principal sea tipo Mapa de Google. Al abrir el entorno de la aplicación directamente accederemos a un xml con la clave.

```
<resources>
    <!--
   TODO: Before you run your application, you need a Google Maps API
key.
    To get one, follow this link, follow the directions and press
"Create" at the end:
https://console.developers.google.com/flows/enableapi?apiid=maps_andro
id_backend&keyType=CLIENT_SIDE_ANDROID&r=13:EE:DE:55:D1:F7:2D:4B:68:E5
:94:4E:F1:EC:A2:B1:31:4D:2B:4F%3Bcom.example.mario.myapplication
    You can also add your credentials to an existing key, using this
line:
```

```
13:EE:DE:55:D1:F7:2D:4B:68:E5:94:4E:F1:EC:A2:B1:31:4D:2B:4F;com.exampl
e.mario.myapplication
    Once you have your key (it starts with "AIza"), replace the
"google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" translatable="false"
templateMergeStrategy="preserve">
        YOUR_KEY_HERE
    </string>
</resources>
```

Hasta aquí hemos obtenido la clave SHA1, pero esta clave no identifica a nuestra aplicación. Para ello debemos añadirle al final el nombre completo de la misma, como nos indican en el xml…

```
13:EE:DE:55:D1:F7:2D:4B:68:E5:94:4E:F1:EC:A2:B1:31:4D:2B:4F;com.exampl
e.mario.sensortracking
```

Esta seria la clave completa que tenemos que introducir en la consola de Google. Una vez introducida, la pagina de Google generara una firma que tendremos que añadir a nuestra aplicación (texto resaltado)

```
Clave para las aplicaciones Android
Clave de la API
AIzaSyCw_u0-GP7HC4aBf8q_frFwfdAHKrElfe0
Aplicaciones Android
13:EE:DE:55:D1:F7:2D:4B:68:E5:94:4E:F1:EC:A2:B1:31:4D:2B:4F;com.exampl
e.mario.myapplication
Fecha de activación
19 jun. 2015 10:27:00
Activado por  micorreo@gmail.com (tú)
```

- El último paso consiste en agregar esta clave al manifiesto de nuestra aplicación. Para ello añadimos este bloque al fichero AndroidManifest.xml del proyecto dentro del bloque <Aplicación>

```
<meta-data
```

```
android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyC03i-M_GHNGs3WyzTxDMhA6jMMJMC4cRo" />
```

2.8.1.3 ACTIVIDAD "SELECCIÓN"

Esta actividad permite seleccionar el dispositivo sobre el que queremos consultar. Pulsando sobre el nombre del dispositivo aparecerá una lista desplegable donde podremos seleccionar el que nos interese.

También podremos fijar los límites del intervalo de tiempo sobre el que se realizara la consulta. Una vez seleccionados todos los parámetros, pulsando el botón Aceptar se lanzara la consulta al servidor y se recogerán los datos proporcionados.

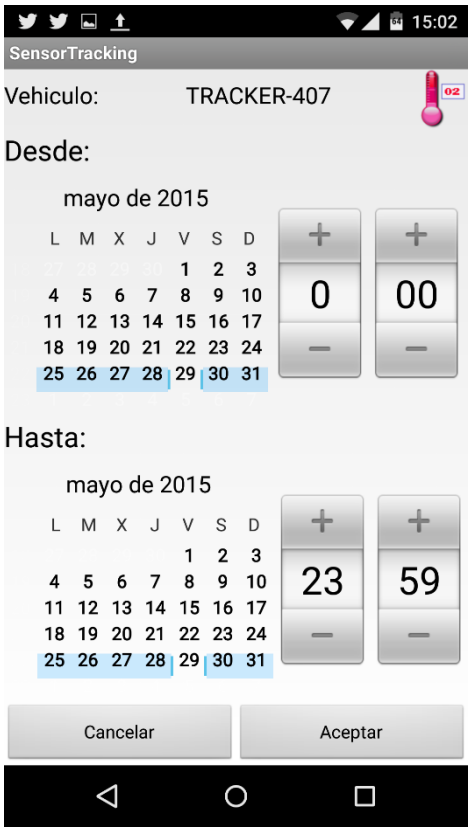

Ilustración 41: Captura de la actividad de SELECCION

2.8.1.4 ACTIVIDAD "GRÁFICAS"

Esta actividad muestra los datos recibidos del servidor en forma de gráfica lineal, siendo el eje X el intervalo de tiempo y el eje Y la magnitud representada. Con los botones de la parte inferior podemos seleccionar la magnitud que se representará.

Después de acotar el intervalo de tiempo y elegir el vehículo a consultar, la aplicación lanza la consulta al servidor, obteniendo la lista de puntos con sus respectivas medidas. Esta información es almacenada en RAM, y representada en una gráfica.

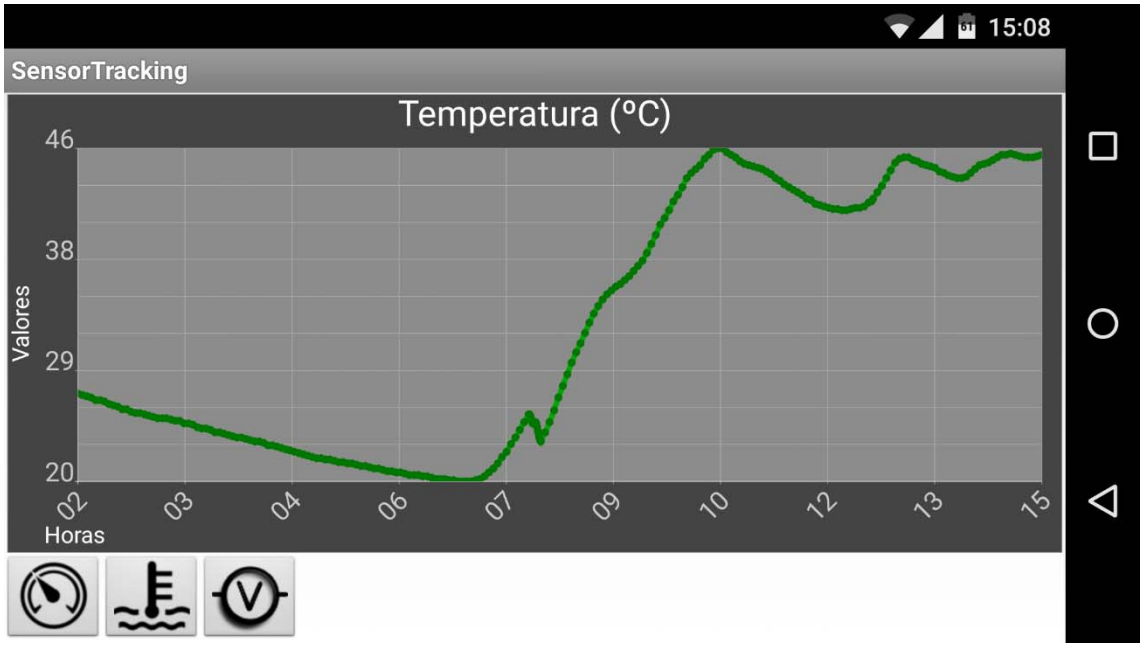

Ilustración 42: Captura de la actividad de GRAFICAS

2.8.2 BASE DE DATOS SQLITE

Android incorpora SQLITE "de serie". SQLITE es un motor de bases de datos transaccional muy ligero y de fácil manejo. Además, si le sumamos el hecho de ser de código libre, se convierte en uno de los motores de bases de datos más populares para aplicaciones de no muy gran envergadura.

En el SDK disponemos de todas API's necesarias para el manejo de SQLITE, siendo frecuentemente utilizada por muchas aplicaciones para almacenar datos no volátiles y poder acceder a ellos con facilidad.

En nuestro caso, vamos a crear dos tablas, una para almacenar parámetros de configuración y otra para guardar la última información recibida de los equipos móviles.

La tabla de configuración únicamente cuenta con dos campos por registro, un campo "clave" que contendrá el nombre que identificará al parámetro, y el campo "valor", que contendrá el valor del parámetro.

Los parámetros que se almacenaran con este método de momento son:

- Usuario (para que se recuerde en el login)
- Password (para que se recuerde en el login)

CAMPO	TIPO	DESCRIPCIÓN
Clave	varchar	Nombre de la clave
Valor	varchar	Valor asignado

La tabla de equipos (similar a la del servidor) almacena la última posición recibida de cada equipo. Se utiliza para mantener una última posición que mostrar en caso de perder la comunicación con el servidor.

CAMPO	TIPO	DESCRIPCIÓN
Fecha_hora	datetime	Fecha y hora de la muestra
Lat	float	Latitud
Lon	float	Longitud
Speed	Int	Velocidad instantánea
Course	Int	Rumbo
Altitud	Int	Altitud
LBS	Varchar	Código de celda GSM
Icon	Varchar	Nombre del fichero con el icono
Mv1	Int	Tensión sensor 1
Mv2	Int	Tensión sensor 2

2.8.3 INTERCAMBIO DE MENSAJES ENTRE CLIENTE Y SERVIDOR

El intercambio de información entre la aplicación cliente y el servidor se realiza mediante mensajes enviados por TCP. El cliente lanza una petición con una palabra clave y una serie de parámetros asociados. El servidor recibe la trama, la procesa y envía una o varias tramas con la información solicitada. Hasta este momento, y por tratarse de un proyecto puramente demostrativo, la información de la trama va en claro, pero debería cifrarse para un uso comercial o profesional, ya que puede contener información sensible, como usuarios, contraseñas o la ubicación de alguien.

Los comandos utilizados son:

LOGIN: El cliente envía al servidor una trama con el usuario y la clave, el servidor verifica que ese usuario esté dado de alta en la base de datos y responde con un código positivo o negativo al cliente. Mientras no se reciba una confirmación del servidor, la aplicación cliente no pasa de la pantalla de Login.

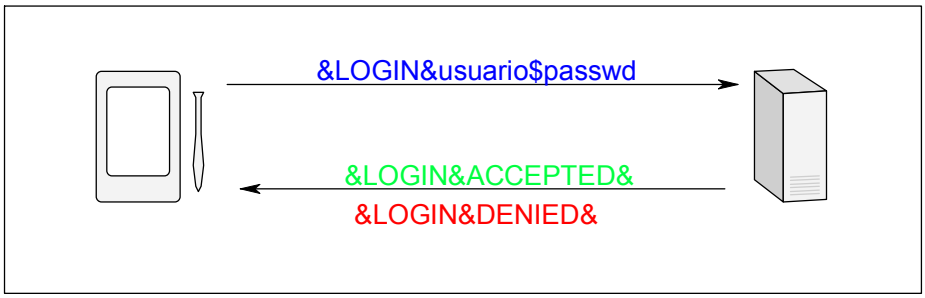

Ilustración 43: Comando LOGIN

POS_USR: Este comando solicita la información en tiempo real de todos los dispositivos remotos asociados a un determinado usuario. Cuando el cliente recibe la respuesta, actualiza su base de datos interna con la información recibida y la representa en el mapa.

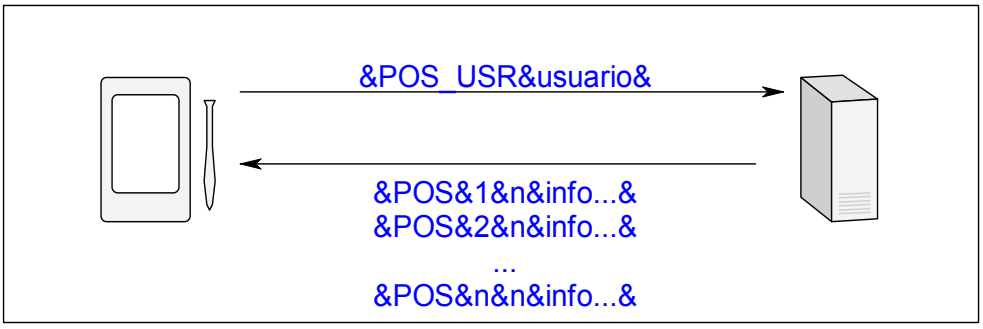

Ilustración 44: Comando POS_USR

TRACK: El comando TRACK solicita todos los datos de un dispositivo durante un intervalo de tiempo. El servidor en este caso comprime la respuesta, ya que puede ser información de varios días, para agilizar en envío. El servidor responde con una trama con identificador TRACKSTART donde informa al cliente del número total de registros

obtenidos, el tamaño total del fichero comprimido y el tamaño de cada paquete. A continuación comienza a enviar los paquetes. A cada paquete el cliente responde con un ACK.

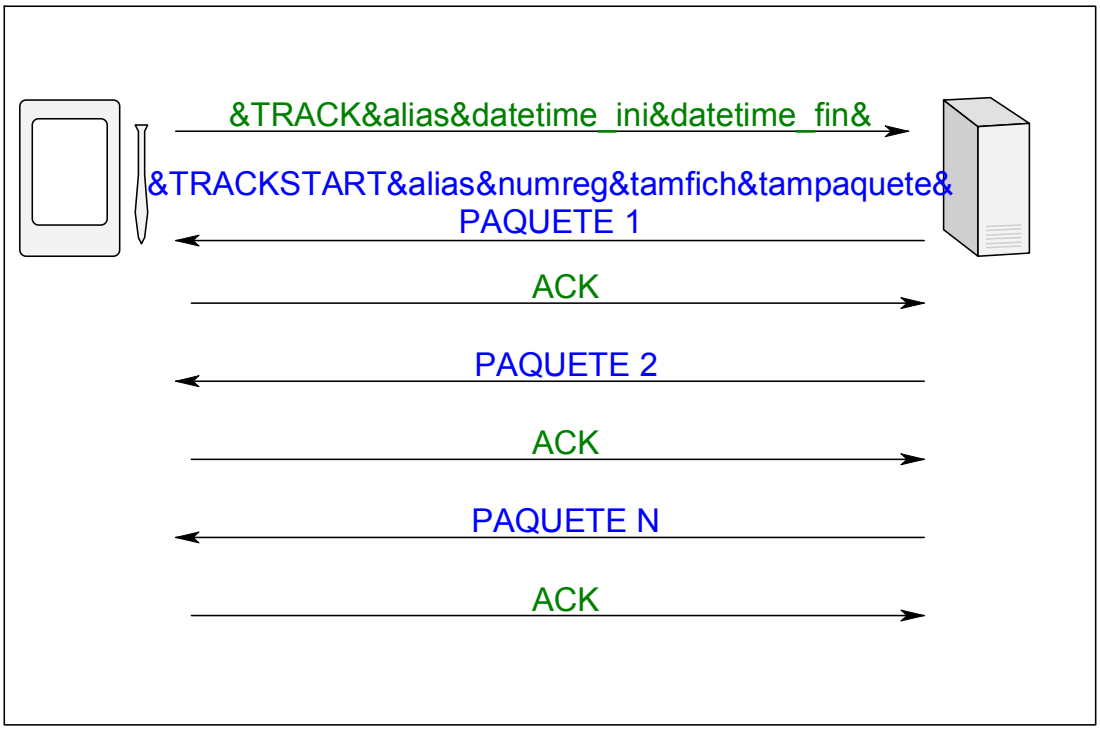

Ilustración 45: Comando TRACK

2.8.4 INTERFAZ DE USUARIO

Al diseñar aplicaciones para Android, tenemos una significativa separación entre el código y el interfaz de usuario. El código lo tenemos en la carpeta "src" del árbol de la aplicación y está compuesto por clases de Java, mientras que el interfaz de usuario lo encontramos en la carpeta res, que a su vez se divide en varias carpetas con diversos contenidos.

La definición de la estructura del interfaz de usuario de cada actividad está definido en un fichero XML dentro de la carpeta res\layout. Cada uno de los elementos parte de una clase fundamental llamada View. Cualquier elemento visual, como un botón, un desplegable, los radio button, etc, heredan de la clase View (vistas). La organización de los elementos dentro de la pantalla de la actividad se hace mediante estructuras denominadas layouts. Android nos ofrece múltiples tipos de Layouts :

- Linear Layouts: Los elementos se organizan linealmente, ya sea en filas o en columnas, por lo que podemos definirlos como verticales u horizontales.
- Relative Layouts: Los elementos se organizan mediante vínculos o referencias de posición entre ellos.
- Absolute Layouts: Los elementos se anclan en posiciones absolutas, teniendo en cuenta que el origen de coordenadas es la esquina superior izquierda de la pantalla.
- Grid Layouts: Los elementos se disponen en función de una rejilla predefinida.
- Frame Layouts: este tipo ubica a todos sus elementos alineados en su esquina superior izquierda, superpuestos entre sí. Suele servir para reemplazar un elemento por otro en la misma posición, ajustando la visibilidad de uno u otro.
- Fragments: Los framgment son layouts que pueden ser creados dinámicamente y reemplazados unos por otros en tiempo de ejecución. Mediante el uso de fragments podemos hacer en una sola actividad lo que en varias, cambiando el interfaz tantas veces como se necesite y adaptándolo al tamaño de la pantalla del dispositivo.

A continuación se describe la estructura de la interfaz gráfica de cada una de las actividades de la aplicación.

2.8.4.1 DISEÑO ACTIVIDAD LOGIN

El interfaz de usuario de la actividad LOGIN está basado en un Linear Layout vertical principal, en el que se ordenan de arriba abajo los siguientes elementos:

- textView1: Es una vista de texto con un texto estático. Contiene el título.
- imageView1: Contiene una imagen estática.
- T_logerror: Vista de texto que mostrara mensajes dinámicos en fincion del resultado del proceso de login.
- Linear Layot horizontal, que a su vez contiene dos elementos:
    - TexView con el literal “Email”
    - EditText para introducir el email que es el identificador para el login.
- Linear Layout horizontal, que contiene:
    - TexView con el literal “Password”
    - EditText para introducir la contraseña. Este último oculta los caracteres sustituyéndolos por asteriscos.
- Linear Layout horizontal conteniendo dos botones:
    - b_salir : Boton para salir sin logear.
    - B_aceptar: Botón para proceder con el login.

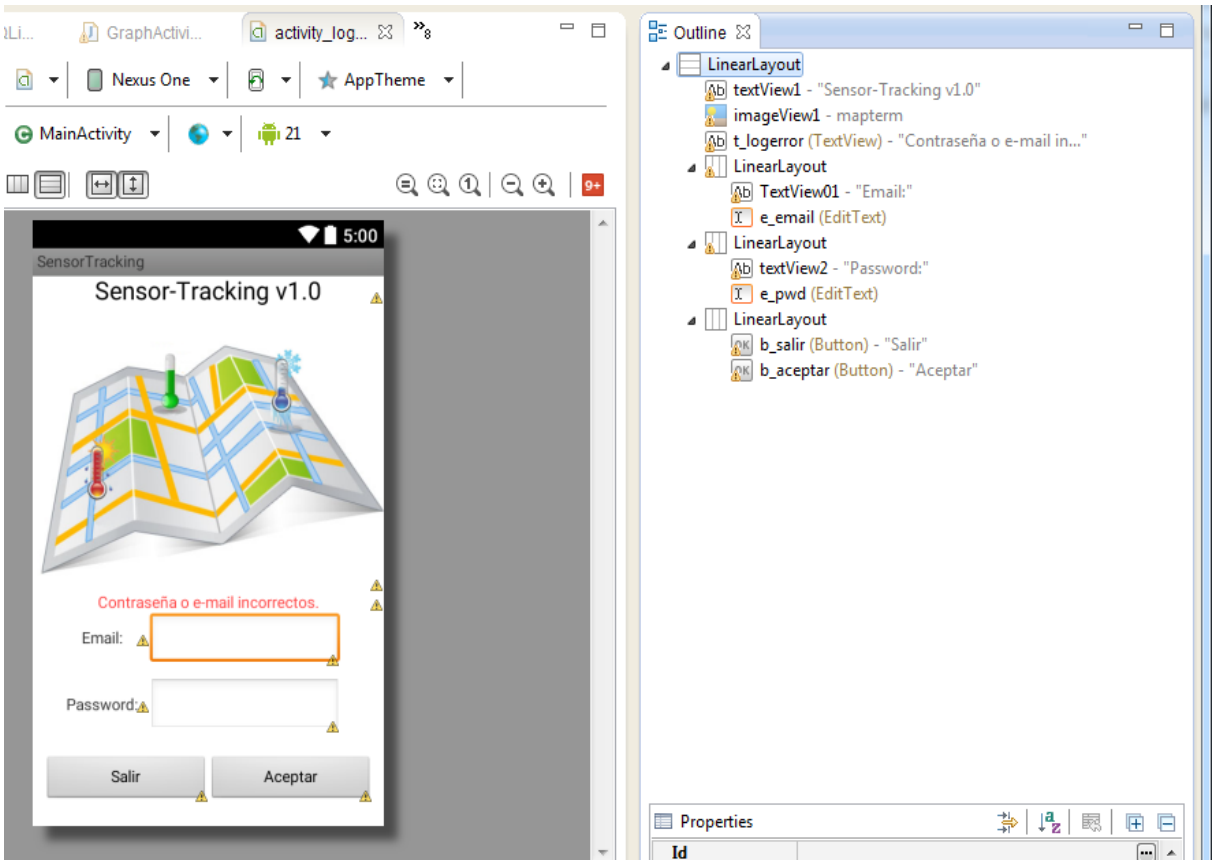

Ilustración 46: Diseño de GUI, actividad LOGIN

2.8.4.2 DISEÑO ACTIVIDAD MAPA

El interfaz grafico de la actividad “MAPA” tiene la siguiente estructura:

- Linear layout vertical, que contiene los siguientes elementos:
    - Fragmet MAPA. Es una vista tipo fragment que contiene el mapa.
    - Linear layout horizontal, que contiene los siguientes botones:
        - Botón “centrar”, que tiene dos funciones: si se hace una pulsación corta, ajusta el zoom y la posición del mapa para encuadrar a todos los vehículos. Si se hace una pulsación larga, se muestra una lista desplegable con todos los vehículos para poder centrar el foco en uno de ellos.
        - Botón “limpiar”, para limpiar elementos mostrados en el mapa.

- Botón “track”. Nos da acceso a la actividad ce selección y posteriormente muestra en el mapa la información capturada en el intervalo de tiempo seleccionado.

- Botón “grafico”. Nos da acceso a la actividad ce selección y posteriormente muestra en un gráfico la información capturada en el intervalo de tiempo seleccionado.

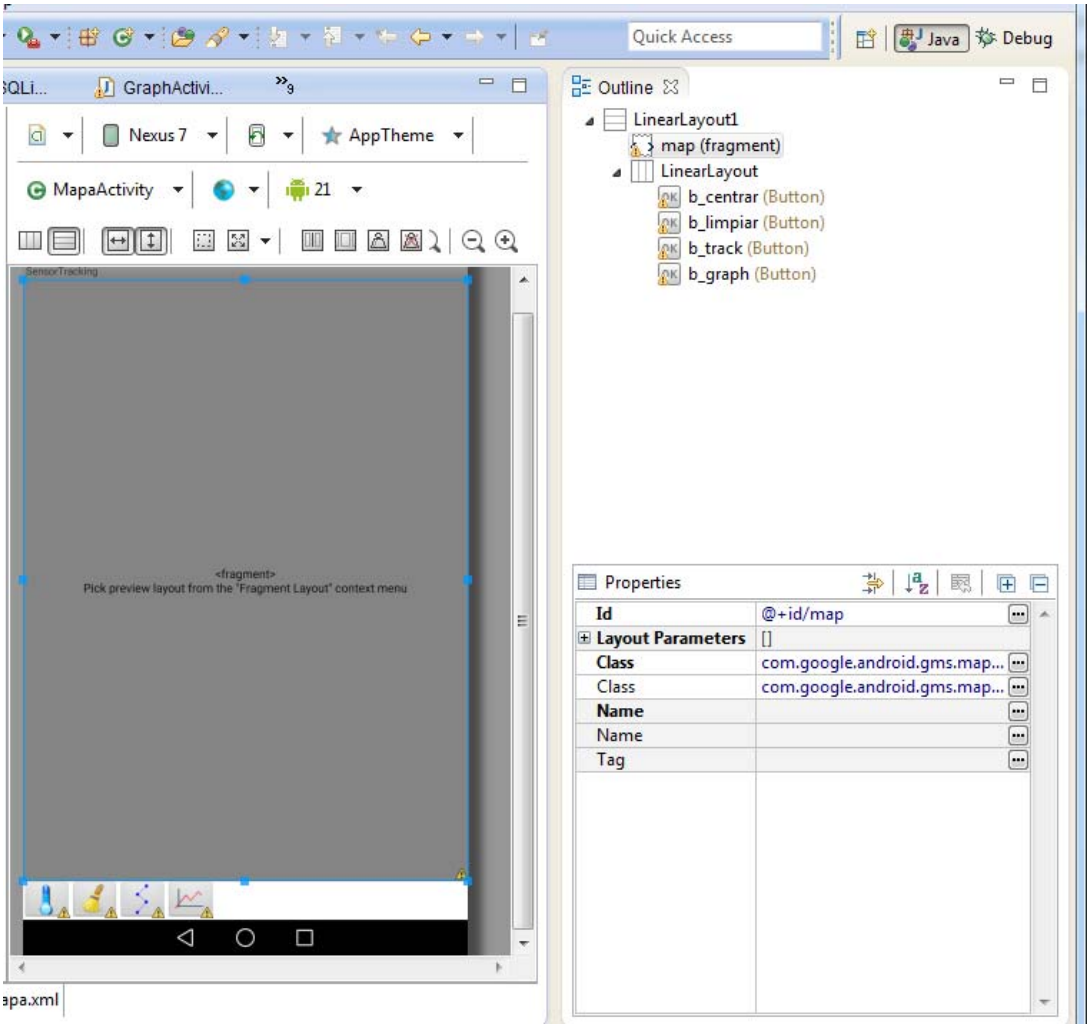

Ilustración 47: Diseño de interfaz de actividad "MAPA"

2.8.4.3 DISEÑO ACTIVIDAD SELECCIÓN

La estructura de la actividad de “selección” está basada en múltiples layouts siguiendo la siguiente estructura:

- LinearLayout Vertical, que contiene al resto de Layouts:

    - LinearLayout Horizontal que contiene los elementos para la selección del vehículo. Son los siguientes:
        - TexView3 con el texto fijo “Vehiculo”
        - TexView4 que mostrará el nombre del vehículo seleccionado.
        - Botón b_tracker. Al pulsarlo se muestra un desplegable para elegir el vehículo.

    - TexView1, con el texto fijo “Desde:”

    - LinearLayout Horizontal, que contiene los elementos que utilizaremos para seleccionar la fecha y la hora de inicio:
        - calendarView1: Es una vista que permite seleccionar una fecha
        - timePicker1: Es una vista que permite seleccionar una hora

    - TexView2, con el texto fijo “Hasta:”

    - LienarLayour Horizontal, que contiene los elementos que utilizaremos para seleccionar la fecha y la hora de fin:
        - calendarView2: Es una vista que permite seleccionar una fecha
        - timePicker2: Es una vista que permite seleccionar una hora

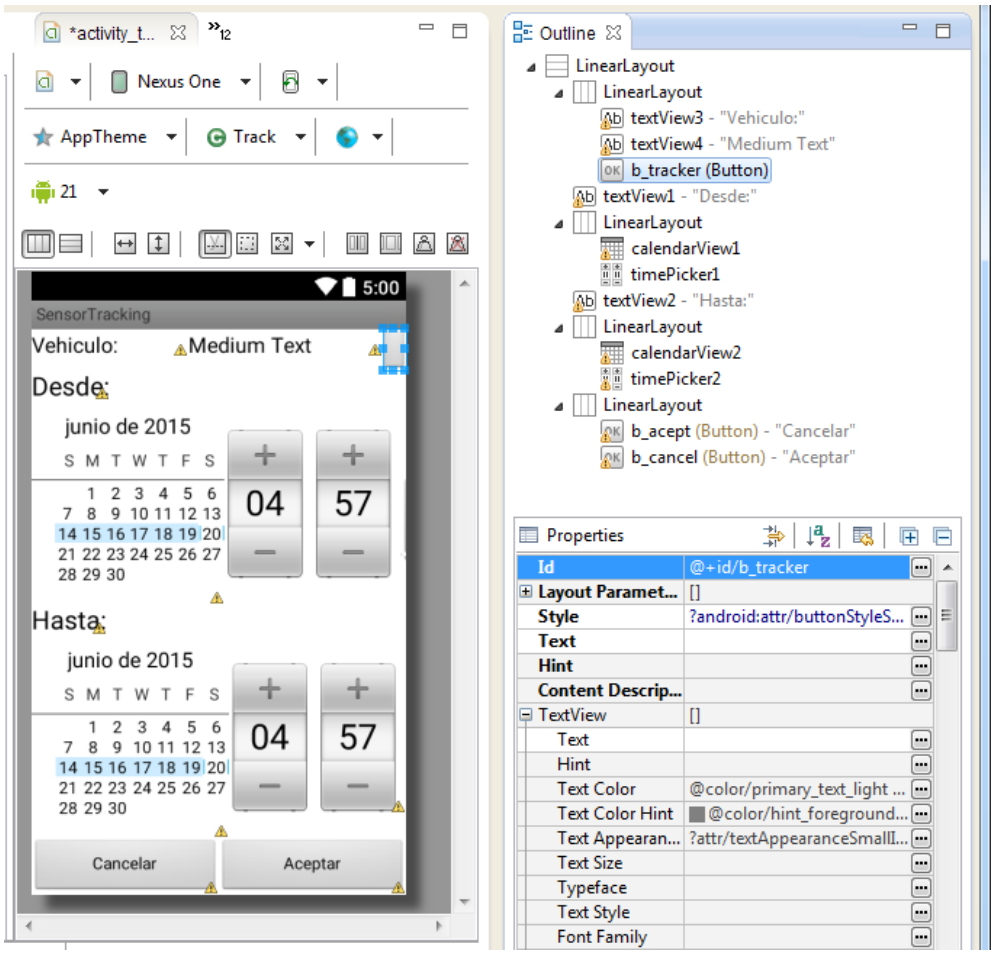

Ilustración 48: Diseño del interfaz de la actividad "Selección"

2.8.4.4 DISEÑO ACTIVIDAD GRÁFICOS

La actividad "Gráficos" simplemente contiene la vista de gráfico y tres botones para seleccionar la medición a mostrar. La estructura es simple:

- Linear Layout Vertical, que contiene:
  - mysimpleXYplot, que es una vista para representación de graficas en el plano XY
  - Linear layout horizontal, que a su vez contiene los tres botones:
    - B_vel, para trazar el gráfico de velocidad
    - B_tracker, que muestra el gráfico de temperaturas
    - B_volt, que muestra el gráfico de tensión de batería.

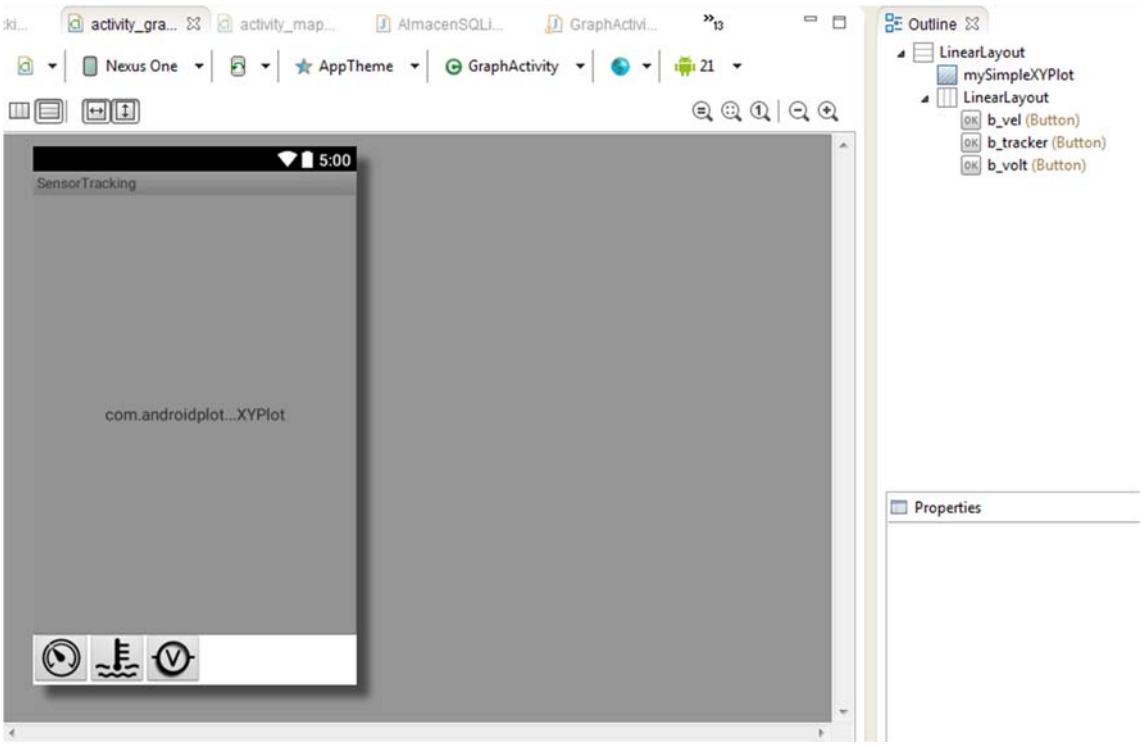

Ilustración 49: Diseño de actividad "gráficos"

2.8.5 FUNCIONAMIENTO DE LA APLICACIÓN

Como en toda aplicación de Android, una de las reglas generales es que nunca se debe detener el hilo principal de la aplicación, que es la que gestiona la interfaz gráfica. En caso de hacerlo, se producirá una excepción y la aplicación finalizará.

Por ello, cuando se realizan operaciones que requieren esperas, como en nuestro caso, las consultas al servidor, siempre se deben realizar en hilos independientes. Todos los eventos del hilo principal son asíncronos, y se gestionan por medio de escuchadores.

Podemos definir escuchadores para atender a los botones, para saber cuándo una tarea ha finalizado o para recibir mensajes de otros hilos.

El hilo principal será pues el que se encargue de gestión la interfaz gráfica, es decir, capturará los eventos de los botones y gestionará la información sobre el mapa.

2.8.5.1 FUNCIONAMIENTO DE LA ACTIVIDAD LOGIN

La actividad LOGIN es la primera que se lanza al iniciar la aplicación. Muestra los campos para introducir el usuario y la contraseña, y los botones de aceptar y cancelar.

Al introducir los datos y pulsar aceptar, se monta un trama tipo "LOGIN" (ver *capítulo 2.8.3*). Esta trama es enviada al servidor mediante un hilo auxiliar, ya que el hilo principal no se puede detener. Este nuevo hilo espera la respuesta del servidor, y al recibirla envía un mensaje al hilo principal. Si la respuesta es negativa, la aplicación finaliza. Si la respuesta es positiva, se da paso a la actividad "MAPA"

Ilustración 50: Proceso de Login

Los datos de usuario y clave son almacenados en la base de datos SQLITE y se volverán a cargar automáticamente la próxima vez que se lance la activad para no tener que volver a introducirlos cada vez.

2.8.5.2 FUNCIONAMIENTO DE LA ACTIVIDAD MAPA

La actividad "MAPA" es la principal dentro de la aplicación y es la que contiene prácticamente toda la lógica.

En esta actividad utilizaremos un hilo secundario para gestionar la comunicación con el servidor y mantener actualizada la posición de los "trackers" en tiempo real. Este hilo se encargará de hacer una consulta al servidor cada 10 segundos usando el comando "POS_USR" (ver capítulo 2.8.3). Al recibir respuesta del servidor, el hilo secundario guarda la información en la base de datos y envía un mensaje al hilo principal. Cuando el hilo principal recibe este mensaje, lee la base de datos y actualiza los "marcadores" del mapa con la nueva información. La *Ilustración 51* muestra el diagrama de flujo de los hilos.

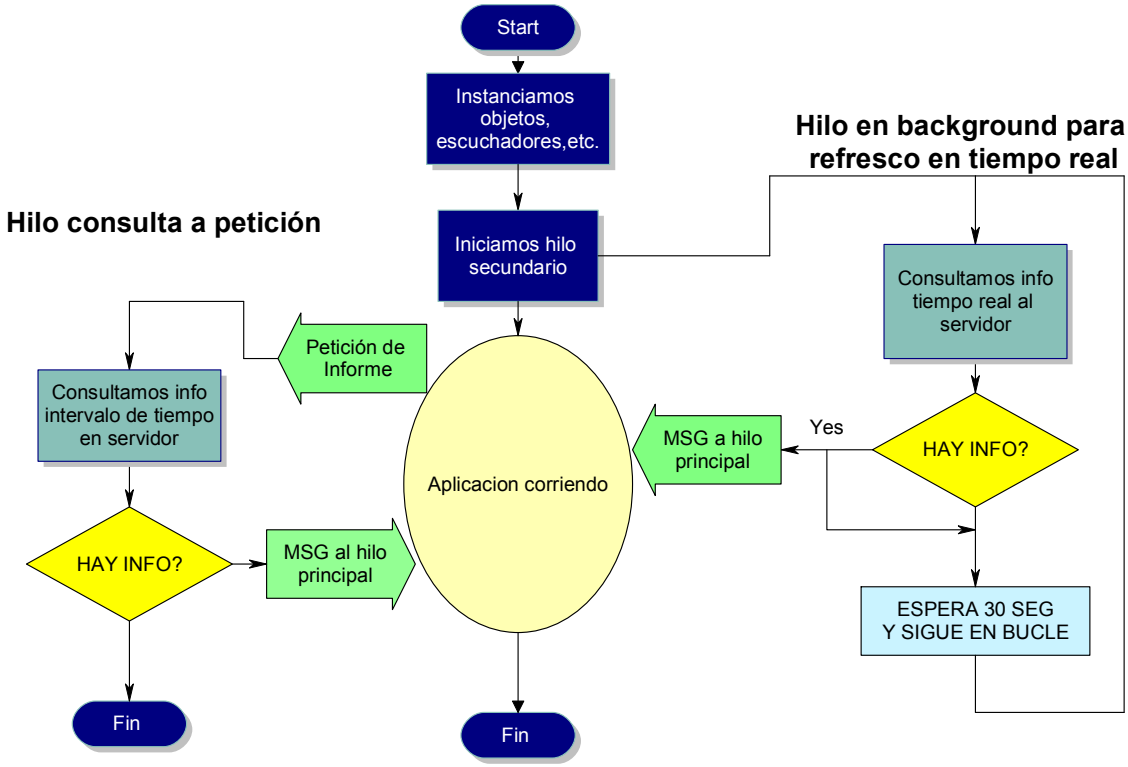

Ilustración 51: Hilos en la actividad principal

También se utilizará otro hilo independiente del anterior cuando se realice una consulta de datos, ya sea para visualizar el en mapa o en forma de gráfica.

Cuando el usuario pulsa uno los botones para solicitar una consulta, la aplicación lanza la actividad de "Selección" donde el usuario seleccionará el vehículo y los límites de tiempo deseados. Al confirmar, la actividad "Selección" finaliza, y la actividad Mapa

recoge los valores establecidos. Con estos valores se crea una trama del tipo "TRACK" (ver 2.8.3) y se envía al servidor mediante el hilo de "consulta a petición" como se muestra en la *Ilustración 51.*

Cuando el hilo finaliza, envía un mensaje a la actividad principal, que contiene una lista con toda la información obtenida. En función del tipo de representación deseada, la actividad principal convierte estos datos en "marcadores", en caso de que se desee representar sobre el mapa, o se crean tres tablas tiempo-valor, una para temperatura, otra para velocidad y otra para tensión, y se pasan como parámetro a la actividad "Gráficos" en caso de esperar una representación gráfica.

2.8.5.3 FUNCIONAMIENTO DE LA ACTIVIDAD "GRAFICOS"

La actividad "GRAFICOS" no tiene ninguna lógica implementada. Simplemente se utiliza una librería de código abierto llamada "AndroidPlot" a la que se le pasan unos arrays con los datos a mostrar.

La actividad principal, después de realizar la consulta al servidor y obtener los datos, actualiza unas listas definidos en la clase Aplicación, que es una clase que no está asociada a ninguna actividad, pero es accesible desde todas, por lo que se utiliza para compartir objetos "globales" entre todas las actividades.

Por lo tanto, cuando la actividad "GRAFICOS" es lanzada, ya tiene la información necesaria en estas listas globales, una por cada magnitud a mostrar. Lo único que queda por hacer es ajustar las leyendas de los ejes en función del tipo de magnitud a mostrar y el intervalo de tiempo seleccionado.

Al pulsar los botones inferiores se vuelve a redibujar la gráfica cambiando la lista de datos por la que corresponda para la información deseada.

2.9 CONCLUSIONES Y FUTUROS TRABAJOS

Este trabajo ha sido realizado para demostrar las aplicaciones de las nuevas tecnologías móviles en diversos campos, de manera genérica. Las mediciones obtenidas y mostradas no tenían mayor interpretación que el hecho de gestionarlas y mostrarlas a través de un dispositivo móvil, pero podría ser fácilmente aplicable a múltiples campos, como la industria, la agricultura, el transporte y la logística, actividades deportivas, etc.

2.9.1 PRUEBAS REALIZADAS

Para realizar las pruebas se han instalado dos equipos localizadores en dos vehículos y han estado funcionando durante más de un año. Durante este tiempo la información enviada ha sido recogida por la aplicación ServiSensor que corre en un servidor virtual VPS alojado en el proveedor OVH. En este servidor se han instalado tanto el MySql como el Java Runtime Environment, y se han aplicado las configuraciones y scripts que se detallan en esta memoria. Toda la información recogida se encuentra almacenada en la base de datos del servidor.

Ocasionalmente también han funcionado otros equipos en diversos vehículos durante pequeños intervalos de tiempo.

Durante el periodo de pruebas se han recogido más de 400.000 muestras, y podemos acceder a ellas desde la aplicación móvil realizando consultas en mapa o gráficos.

La aplicación Android se ha probado en varios terminales, tanto teléfonos como tabletas, con distintas versiones de Android, dando resultados satisfactorios en general, aunque cuando en volumen de datos de la consulta generada es demasiado grande, hay ocasiones en las que producen cortes en el envío de los datos desde el servidor que habría que depurar.

2.9.2 FUTURAS MEJORAS

Como futuras mejoras, podríamos plantear las siguientes.

- Adaptación de los interfaces gráficos, para adecuarlos a la estética y funcionalidad de las nuevas versiones de Android (L, M…). Se deberían sustituir los botones de la activad MAPA por un menú desplegable lateral o superior.

- Encriptación de las comunicaciones. Durante el desarrollo de este trabajo se ha obviado la seguridad. Todos los datos se transfieren en claro. Sería conveniente enviar los datos cifrados para evitar acciones malintencionadas o robos de información.

- Añadir un menú de configuración, que permita asignar distintos parámetros a cada equipo remoto y configurar parámetros generales de la aplicación.

- Avisos y alarmas mediante notificaciones. Se podrían establecer límites configurables, de forma que al rebasarse alguno de ellos se emitiera un aviso por medio de una notificación. Podrían ser, por ejemplo, excesos de velocidad, temperaturas demasiado altas o bajas, etcétera, en función de las magnitudes a tratar.

- Descarga de los datos obtenidos en forma de ficheros, que puedan ser compartidos, almacenados o exportados a otras aplicaciones.

- Realizar la aplicación en tecnología “WEB” para poder ser migrada fácilmente a IOS y Windows Phone.

- Realizar un portal WEB para poder acceder a la información y controlar el sistema desde un navegador.

3. DIAGRAMAS

En esta sección se adjunta el código fuente de ambas aplicaciones, la que corre en el servidor y la desarrollada para los dispositivos Android, junto con los xml que describen el interfaz gráfico.

3.1 CÓDIGO FUENTE APLICACIÓN SERVIDOR

3.1.1 CLASE "ServiSensor"

Esta es la clase principal de la actividad servidor. Se encarga de conectar con la base de datos, abrir el socket TCP y quedarse a la espera de datos entrantes. Según van llegando los datos al socket, los va procesando y realiza las acciones necesarias en función del tipo de trama recibida.

Si se trata de una trama de datos de un equipo móvil, se almacenan los datos directamente en la base de datos. Si por el contrario se tratara de una petición de un dispositivo Android, se lanza un nuevo hilo para atender a dicha petición.

```
package ServiSensor;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.nio.charset.Charset;
import java.nio.charset.CharsetDecoder;
import java.nio.charset.CharsetEncoder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.Locale;
import java.util.Set;
import java.util.TimeZone;
import java.util.Vector;
```

```
public class ServiSensor{

    /**
     * @param args
     */
    static Connection conn = null;
    static ServerSocketChannel sk = null;
    static Charset charset = Charset.forName("UTF-8");
    static CharsetEncoder encoder = charset.newEncoder();
    static CharsetDecoder decoder = charset.newDecoder();
    static ServerSocketChannel server = null;
    static Selector selector = null;
    static Iterator i;

    static public MiCliente miCliente;
    static public Vector<MiCliente> misClientes;
    static MisTrackers misTrackers;



    public static void main(String[] args) throws InterruptedException
{
        boolean reabrir=false;
        String comando="";

        misClientes=new Vector <MiCliente>();


        misTrackers=new MisTrackers();
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader (isr);
        int ch_in;


        String datos = "";


        //Abrimos el socket del servidor
        AbrirSocket();
        //Conectamos con la base de datos
        AbrirBaseDatos();
        //Bucle principal. Chequeamos datos entrantes por el SOCKET
        while (true) {
            try{
            while(br.ready()){
                ch_in=br.read();
                if(ch_in==0x0d){
                    System.out.println("COMANDO
RECIBIDO:"+comando+"\n");
                }
                else{
                  comando+=(char)ch_in;
                }

            }
            }catch(Exception e){

            }
            try {
```

```
                    // Esperando eventos

                    selector.select();
                    // obtenendo entradas
                    Set keys = selector.selectedKeys();
                    i= keys.iterator();
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }

                    //para cada entrada...
                    while (i.hasNext()) {
                        SelectionKey key = (SelectionKey) i.next();

                        //eliminamos la entrada actual
                        i.remove();
                        if (!key.isValid()) {
                            continue;
                        }

                        // if isAccetable = true
                        // un cliente solicita conexion
                        if (key.isAcceptable()) {
                            // obtenemos el socket del cliente
                            SocketChannel client = server.accept();
                            // Sin bloquar I/O
                            client.configureBlocking(false);
                            // registramos el cliente
                            client.register(selector,
SelectionKey.OP_READ);
                            continue;
                        }

                        // if isReadable = true
                        // hay datos que leer
                        if (key.isReadable()) {

                            SocketChannel client = (SocketChannel)
                            key.channel();

                             // lectura del buffer proveniente del client
                                int BUFFER_SIZE = 4000;
                                int lng;
                                 ByteBuffer buffer =
                            ByteBuffer.allocate(BUFFER_SIZE);
                                try {
                                    lng = client.read(buffer);
                                } catch (Exception e) {
                                  //el cliente ya no esta activo
                                  try{
                                        i.remove();
                                    }
                                    catch(Exception j){
                                       System.out.println("ERROR EN
      i.remove()");
                                        continue;
                                    }

                                    e.printStackTrace();
```

```
                                continue;
                            }
                            if (lng <= 0)
                                continue;
                            // Show bytes on the console
                            buffer.flip();
                        //
System.out.println(client.getRemoteAddress().toString() + " --> " +
lng+ " Bytes recibidos");
                            // Charset charset=Charset.forName("ISO-
8859-1");
                            // CharsetDecoder decoder =
charset.newDecoder();
                            CharBuffer charBuffer =
decoder.decode(buffer);
                            datos = charBuffer.toString();
                            if (datos.length() > 0) {

System.out.println(client.getRemoteAddress().toString() + " --> " +
datos);
                                ProcesarDatos(client, datos);
                            }
                            continue;
                        }
                    }
                }
             catch (Exception e) {
                e.printStackTrace();
                System.out.println("SE HA PRODUCIDO UN ERROR: SE
REINICIARAN LOS SERVICIOS\n ");
                reabrir=true;
            }
            if(reabrir==true){

                try{
                    reabrir=false;
                    System.out.println("REABRIENDO SERVICIOS\n ");

                    misClientes.clear();
                    selector.close();
                    server.close();
                    conn.close();
                    AbrirSocket();
                    AbrirBaseDatos();
                }
                catch(Exception e){
                    System.out.println("ERROR REABRIENDO SERVICIOS\n
");
                    e.printStackTrace();
                    //System.exit(-1);
                }


            }
        }

    }
// FUNCION: AbrirBaseDato()
// Esta funcion conecta con la base de datos

    public static int AbrirBaseDatos()
```

```
    {
      String login = "+++++++";
      String password = "????????";
      String url = "jdbc:mysql://localhost/ServiSensor";
      int resp=-1;
      try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            conn = DriverManager.getConnection(url, login, password);
            if (conn != null) {
                 System.out.println("Conexion con DDBB
establecid\r\n”);

                return 1;
                // conn.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex);
        } catch (ClassNotFoundException ex) {
            System.out.println(ex);
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return resp;
    }
    // FUNCION: AbrirSocket()
    // funcion que abre un socket tcp en modo servidor non-blocking
    public static void AbrirSocket()
    {
        try {
            server = ServerSocketChannel.open();
            // nonblocking I/O
            server.configureBlocking(false);
            // host-port 8000

            server.socket().bind(new InetSocketAddress(4444));
            System.out.println("Servidor  activo puerto 4444");

            // Create the selector
            selector = Selector.open();
            // Recording server to selector (type OP_ACCEPT)
            server.register(selector, SelectionKey.OP_ACCEPT);
        } catch (Exception e) {
            e.printStackTrace();

        }
    }

    public static int buscaCliente(SocketChannel cliente){
        MiCliente mc=new MiCliente();
        if(misClientes.size()==0) return -1;
        for(int i=0;i<misClientes.size();i++){
            mc=misClientes.elementAt(i);
            if(mc.cliente==cliente) return i;
        }
        return -1;

    }
```

```
    public static int limpiaClientes(){
        MiCliente mc;
        System.out.println("Limpiando Clientes");
        if(misClientes.size()==0) return -1;
        for(int i=0;i<misClientes.size();i++){
            mc=misClientes.elementAt(i);
            System.out.println("Cliente: "+mc.cliente.toString()+"
hilo activo: "+mc.hilo.getId()+" "+mc.hilo.isAlive());
            if(!mc.hilo.isAlive()){
                System.out.println("Eliminando
cliente:"+mc.cliente.toString()+" por inactividad. Quedan
"+misClientes.size());
                misClientes.remove(i);
                i=0;
            }
        }
        return -1;

    }
    public static void ProcesarDatos(SocketChannel cliente, String
datos_in) {
        String[] campos = new String[100];
        String datos;
        int indice_ini=0;
        int indice_fin=0;

        do{
             indice_fin=datos_in.indexOf(0x0d,indice_ini);
             if(indice_fin==-1){
                 datos=datos_in.substring(indice_ini);
             }
             else{
                datos=datos_in.substring(indice_ini,indice_fin);
             }
             indice_ini=indice_fin+1;
             datos=datos.trim();
             if(datos.equals(""))break;

             try {
                 if (datos.charAt(0) == '#') {

                        campos = datos.split("#");
                        if(campos[1].equals("CAMPER")){
                            procesaTrackerCinterion(campos,cliente);

                        }
                        if ((campos[1].length() ==
15)&&(campos.length>2)) {
                            procesaTrackerChino(campos,cliente);
                        }
                        else {
                            campos[1]=datos;

                        }

                        // ResultSet res = stmt.executeQuery("INSERT
INTO equipos
                        // (imei,date,time,lat,lon,speed,course)
VALUES (,,,),(4,5,6) ON
                        // DUPLICATE KEY UPDATE
```

```
                        // ON DUPLICATE KEY UPDATE
c=VALUES(a)+VALUES(b);
                    }
                else{
                    if (datos.charAt(0) == '&') { // peticion de un
cliente
                        campos = datos.split("&");
                    }
                    else{
                        campos[1]=datos;

                    }
                    MiCliente miCliente=new MiCliente();
                    int idx=buscaCliente(cliente);
                    if(idx>=0){ //ya esta el cliente en la lista

                            miCliente=misClientes.elementAt(idx);
                            if(miCliente.hilo.isAlive()){
                               miCliente.hilo.campos=campos;
                               System.out.println("hilo aun activo:"+
miCliente.hilo.getName());

                               return;
                            }
                            else{
                                System.out.println("hilo no activo: "+
miCliente.hilo.getName()+ " Eliminando...");
                                misClientes.remove(idx);
                            }
                    }


                    miCliente.cliente=cliente;
                    ProcesarComando hilo =new
ProcesarComando(cliente,conn, campos,misTrackers);
                    miCliente.hilo=hilo;

                    misClientes.add(miCliente);
                    System.out.println("Nuevo hilo creado. Toral:"+
misClientes.size());
                    hilo.start();
                    limpiaClientes();//limpia la lista de hilos
quitando los no activos

                }


            } catch (Exception e) {
                e.printStackTrace();

            }
        }while(indice_fin!=-1);

    }
    static int procesaTrackerChino(String[] campos,SocketChannel
cliente)
    {
        String[] gps = new String[6];
        String imei, url = "No disponible", alias = "";
        Statement stmt=null;
        int numtramas = 0;
```

```
        double lon = 0, lat = 0, kmh = 0, rumbo = 0, min = 0;
        String fecha, hora, aux,lbs="";
        MiTracker tracker=new MiTracker();
        try {
            stmt = conn.createStatement();

            imei = campos[1];
            alias = campos[2];
            if (alias.equals(""))
                alias = "DESCONOCIDO";
            tracker.alias=alias;
            tracker.imei=imei;
            tracker.cliente=cliente;
             java.util.Date date= new java.util.Date();
            tracker.tiempo=new Timestamp(date.getTime());
            misTrackers.setTracker(tracker);

            numtramas = Integer.parseInt(campos[6]);
            for (int i = 0; i < numtramas; i++) {
                if (campos[7 + (4 * i)].equals("V")) { // es GPS
valido
                    gps = campos[8 + (4 * i)].split(",");
                    lon = Double.parseDouble(gps[0].substring(0, 3));
                    min = Double.parseDouble(gps[0].substring(3)) /
60;
                    lon = lon + min;
                    if (gps[1].equals("W"))
                        lon = lon * -1;
                    lat = Double.parseDouble(gps[2].substring(0, 2));
                    min = Double.parseDouble(gps[2].substring(2)) /
60;
                    lat = lat + min;
                    if (gps[3].equals("S"))
                        lon = lon * -1;
                    kmh = Double.parseDouble(gps[4]);
                    kmh = kmh * 1.852;
                    rumbo = Double.parseDouble(gps[5]);
                    url = String.format(
                            "http://maps.google.es/?q=%.5f%%20%.5f",
                            lat, lon);
                    url = url.replace(',', '.');
                    // System.out.println(url);
                    aux = campos[9 + (4 * i)];
                    fecha = aux.substring(4, 6) + aux.substring(2, 4)
                            + aux.substring(0, 2);
                    hora = campos[10 + (4 * i)];
                    String fecha_hora = "20" + fecha + hora;
                    String query = String
                            .format(Locale.ENGLISH,
                                    "INSERT INTO equipos
(imei,alias,datetime,lat,lon,speed,course,url) VALUES
('%s','%s','%s','%.5f','%.5f','%.1f','%.1f','%s')",
                                    imei, alias, fecha_hora, lat, lon,
                                    kmh, rumbo, url);
                    String query2 = String
                            .format(Locale.ENGLISH,
                                    " ON DUPLICATE KEY UPDATE
alias='%s',datetime='%s',lat='%.5f',lon='%.5f',speed='%.1f',course='%.
1f',url='%s'",
                                    alias,fecha_hora, lat, lon, kmh,
rumbo,
```

```
                                    url);
                    query += query2;
                    //System.out.println(query);
                    int res = stmt.executeUpdate(query);

                    query = String
                            .format(Locale.ENGLISH,
                                    "INSERT INTO tracking
(imei,datetime,lat,lon,speed,course) VALUES
('%s','%s',%.5f,%.5f,%.1f,%.1f)",
                                    imei, fecha_hora, lat, lon, kmh,
                                    rumbo);
                    //System.out.println(query);
                    res = stmt.executeUpdate(query);

                } else {

                    // El GPS no es valido

//#356823033534475#TRACKER_001#0#6020#AUT#1#0B0613CF#0,,0,,,####
                    lbs=campos[7 + (4 * i)]; // no es GPS valido,
tenemos el LBS
                    aux="";
                    if(campos.length>9){
                      aux = campos[9 + (4 * i)];
                    }
                    String fecha_hora="";

                    //if(aux.equals("")){
                        //fecha_hora=new SimpleDateFormat("yyyyMMdd
HHmmss").format(Calendar.getInstance(TimeZone.getTimeZone("UTC")).getT
ime());
                        Calendar cal =
Calendar.getInstance(TimeZone.getTimeZone("GMT"));
                        Date currentLocalTime = cal.getTime();

                        DateFormat df = new SimpleDateFormat("yyyy-MM-
dd HH:mm:ss");
                        df.setTimeZone(TimeZone.getTimeZone("GMT"));

                        fecha_hora = df.format(currentLocalTime);
                    //} // SI EL GPS ESTA MAL NO NOS FIAMOS DE LA HORA
QUE NOS DA, porque a veces es incorrecta
                    /*else{
                        fecha = aux.substring(4, 6) + aux.substring(2,
4)
                                + aux.substring(0, 2);
                        hora = campos[10 + (4 * i)];
                         fecha_hora = "20" + fecha + hora;
                    }*/
                    String query = String
                            .format(Locale.ENGLISH,
                                    "INSERT INTO equipos
(imei,alias,datetime,lat,lon,speed,course,url,lbs) VALUES
('%s','%s','%s','%.5f','%.5f','%.1f','%.1f','%s','%s')",
                                    imei, alias, fecha_hora, lat, lon,
                                    kmh, rumbo, url,lbs);
                    String query2 = String
                            .format(Locale.ENGLISH,
                                    " ON DUPLICATE KEY UPDATE
alias='%s',datetime='%s',url='%s',lbs='%s'",
```

```
                                        alias,fecha_hora, url,lbs);
                    //String query2 = String
                    //      .format(Locale.ENGLISH,
                //                  " ON DUPLICATE KEY UPDATE
alias='%s',datetime='%s',lat='%.5f',lon='%.5f',speed='%.1f',course='%.
1f',url='%s',lbs='%s'",
                    //              alias,fecha_hora, lat, lon, kmh,
rumbo, url,lbs);
                    query += query2;
                //  System.out.println(query);
                    int res = stmt.executeUpdate(query);

                    query = String
                            .format(Locale.ENGLISH,
                                    "INSERT INTO tracking
(imei,datetime,lat,lon,speed,course,lbs) VALUES
('%s','%s',%.5f,%.5f,%.1f,%.1f,'%s')",
                                    imei, fecha_hora, lat, lon,
kmh,rumbo,lbs);
                //  System.out.println(query);
                    res = stmt.executeUpdate(query);



                }

            }
            stmt.close();
            return 1;

        } catch (SQLException e) {
            try {
                stmt.close();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                return -2;


            }
            // TODO Auto-generated catch block
            e.printStackTrace();
            return -1;
        }

        }
        static int procesaTrackerCinterion(String[] campos,SocketChannel
cliente)
        {
            String[] gps = new String[20];
            String imei, url = "No disponible", alias = "";
            Statement stmt=null;
            int numtramas = 0,altitud;
            double lon = 0, lat = 0, kmh = 0, rumbo = 0, min = 0;
            String fecha, hora, aux;
            MiTracker tracker = new MiTracker();
            try {
                stmt = conn.createStatement();

                imei = campos[2];
```

```
            alias = campos[3];
            if (alias.equals(""))
                alias = "DESCONOCIDO";

            tracker.alias=alias;

            tracker.imei=imei;
            tracker.cliente=cliente;
             java.util.Date date= new java.util.Date();
            tracker.tiempo=new Timestamp(date.getTime());
            misTrackers.setTracker(tracker);


            gps = campos[4].split(",");
// Procesamos los campos basicos de GPS de la trama, que van separados
por comas
//#SENSOR#353234025641513#TRACKER_001#2013/06/18,15:59:22,40.2518133,N
,003.6946650,W,00624,001,245.67,3
            //2013/06/18  15:59:22 40.2518133 N  003.6946650  W  00624
001   245.67  3
             //   0          1       2       3   4           5    6
7      8     9
            //  if (gps[9].equals("3")) { // es GPS valido

                    lon = Double.parseDouble(gps[4]);

                    if (gps[5].equals("W"))
                        lon = lon * -1;
                    lat = Double.parseDouble(gps[2]);

                    if (gps[3].equals("S"))
                        lon = lon * -1;

                    kmh = Double.parseDouble(gps[7]);
                    //kmh = kmh * 1.852;
                    rumbo = Double.parseDouble(gps[8]);
                    url =
String.format("http://maps.google.es/?q=%.5f%%20%.5f",lat, lon);
                    url = url.replace(',', '.');
                    altitud=Integer.parseInt(gps[6]);
                    // System.out.println(url);

                    fecha = gps[0];
                    hora = gps[1];
                    String fecha_hora = fecha +" "+ hora;
                    if(!gps[9].equals("0")){
                        String query = String
                                .format(Locale.ENGLISH,
                                        "INSERT INTO equipos
(imei,alias,datetime,lat,lon,speed,course,altitude,url) VALUES
('%s','%s','%s','%.5f','%.5f','%.1f','%.1f','%d','%s')",
                                        imei, alias, fecha_hora, lat,
lon,
                                        kmh, rumbo,altitud, url);
                        String query2 = String
                                .format(Locale.ENGLISH,
                                        " ON DUPLICATE KEY UPDATE
alias='%s',datetime='%s',lat='%.5f',lon='%.5f',speed='%.1f',course='%.
1f',altitude='%d',url='%s'",
                                        alias,fecha_hora, lat, lon,
kmh, rumbo,altitud,
```

```
                                        url);
                        query += query2;
                        //System.out.println(query);
                        int res = stmt.executeUpdate(query);
                        query = String
                                .format(Locale.ENGLISH,
                                        "INSERT INTO tracking
(imei,datetime,lat,lon,speed,course,altitude) VALUES
(%s,'%s',%.5f,%.5f,%.1f,%.1f,%d)",
                                        imei, fecha_hora, lat, lon,
kmh,
                                        rumbo,altitud);
                    //  System.out.println(query);
                        res = stmt.executeUpdate(query);
                    }
                    else{


                        String query2 = String
                                .format(Locale.ENGLISH,
                                        "UPDATE equipos SET
alias='%s',datetime='%s'WHERE imei LIKE '%s'",
                                        alias,fecha_hora,imei);
                        //query += query2;
                        //System.out.println(query);
                        int res = stmt.executeUpdate(query2);


                    }

                //}


        stmt.close();
        return 1;

        } catch (SQLException e) {
            try {
                stmt.close();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                return -2;

            }
            // TODO Auto-generated catch block
            e.printStackTrace();
            return -1;
        }
    }



}
```

3.1.2 CLASE “ProcesarComando”

La clase “ProcesarComando” es la que trata cada peticion de un dispositivo Android al servidor. Esta clase “hereda” de la clase Thread, ya que va a funcionar en un hilo propio. Así el bucle principal de la aplicación puede seguir trabajando mientras que cada petición se procesa en su propio hilo.

```
package ServiSensor;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.channels.SocketChannel;
import java.nio.charset.CharacterCodingException;
import java.nio.charset.Charset;
import java.nio.charset.CharsetDecoder;
import java.nio.charset.CharsetEncoder;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Vector;
import java.util.zip.GZIPOutputStream;

public class ProcesarComando extends Thread {

    private String datos;
    private SocketChannel cliente;
    private Connection conn;
    static Charset charset = Charset.forName("UTF-8");
    static CharsetEncoder encoder = charset.newEncoder();
    static CharsetDecoder decoder = charset.newDecoder();
    static String[] campos = new String[100];
    static MisTrackers misTrackers;
    public ProcesarComando(SocketChannel in_cliente, Connection
in_conn,String[] in_datos,MisTrackers mt) {
        cliente = in_cliente;
        campos = in_datos;
        conn = in_conn;
        misTrackers=mt;


    }

    public void run() {


        String output = "", imei = "", alias = "", lat = "", lon = "",
speed = "", url = "", datetime = "", rumbo = "", start =
"",lbs="",altitud="";

        try {
            Statement stmt = conn.createStatement();
            if (campos[1].equals("WAIT")) {
                int espera= Integer.parseInt(campos[2]);
                for(int i=espera;i>0;i--){
                  Thread.sleep(1000);

System.out.println("Hilo:"+Thread.currentThread().toString()+"
esperando: "+i);
```

```
                }
            }
            else if (campos[1].equals("AT")) {
                alias=campos[2];
                String comando=campos[3];
                output=ProcesaComandoAT(alias,comando);

cliente.write(encoder.encode(CharBuffer.wrap(output)));

            }
            else if (campos[1].equals("CONN")) {
                //lista de equipos conectados
                int size=misTrackers.size();
                for(int i=0;i<size;i++){
                    MiTracker tr= misTrackers.get(i);

output="&CONN&"+(i+1)+"&"+size+"&"+tr.imei+"&"+tr.alias+"&"+tr.cliente
.toString()+"&"+tr.tiempo.toLocaleString()+"\r\n";

cliente.write(encoder.encode(CharBuffer.wrap(output)));
                }
            }
            else if (campos[1].equals("POS")) {
                ResultSet res = stmt.executeQuery("SELECT * FROM
equipos WHERE alias LIKE '"+ campos[2] + "'");
                // System.out.println("\nIMEI \t\t BUS \t\t\t
VELOCIDAD \n");
                while (res.next()) {
                    imei = res.getString("imei");
                    alias = res.getString("alias");
                    lat = res.getString("lat");
                    lon = res.getString("lon");
                    datetime = res.getString("datetime");
                    speed = res.getString("speed");
                    url = res.getString("url");
                    rumbo = res.getString("course");
                    // System.out.println(nombre + " \t " + lat + " \t
"+ lon);
                }
                res.close();
                if (alias != "") {

                    output = "&POS&" + alias + "&" + datetime + "&" +
lat + "&"
                            + lon + "&" + speed + "&" + rumbo + '&' +
url
                            + "&\r\n";


cliente.write(encoder.encode(CharBuffer.wrap(output)));
                    System.out.println("Enviando:" + output);

                }
            }
            else if (campos[1].equals("POS_USR")) {
                ResultSet res = stmt.executeQuery("SELECT * FROM
equipos WHERE user LIKE '"+ campos[2] + "'");
                // System.out.println("\nIMEI \t\t BUS \t\t\t
VELOCIDAD \n");
                if (res.last()) {
                    int rows = res.getRow();
```

```
                    int i = 0;
                    // Move to beginning
                    res.beforeFirst();
                    output = "";
                    while (res.next()) {
                        imei = res.getString("imei");
                        alias = res.getString("alias");
                        lat = res.getString("lat");
                        lon = res.getString("lon");
                        datetime = res.getString("datetime");
                        speed = res.getString("speed");
                        url = res.getString("url");
                        rumbo = res.getString("course");
                        altitud = res.getString("altitude");
                        lbs=res.getString("lbs");
                        // System.out.println(nombre + " \t " + lat +
" \t "+ lon);

                        if (alias != "") {

                            output += "&POS&"+(i+1)+"&"+rows+"&" +
alias + "&" + datetime + "&" + lat + "&"
                                    + lon + "&" + speed + "&" + rumbo
+ '&' + altitud +'&'+ lbs + '&'+ url
                                    + "&\n";
                        }
                        i++;
                    }

cliente.write(encoder.encode(CharBuffer.wrap(output)));
                    System.out.println("Enviando:\n" + output);
                    res.close();
                }
            }
            else if (campos[1].equals("TRACK")) {
                alias = campos[2];
                // OBTENEMOS EL IMEI A PARTIR DEL ALIAS
                ResultSet res = stmt
                        .executeQuery("SELECT * FROM equipos WHERE
alias LIKE '"
                                + alias + "'");
                while (res.next()) {
                    imei = res.getString("imei");
                }
                res.close();
                // Obtenemos los puntos del intervalo
                String dt1,dt2;
                if(campos[3].compareTo(campos[4])>0){
                    dt1=campos[4];
                    dt2=campos[3];
                }
                else{
                    dt1=campos[3];
                    dt2=campos[4];


                }
                res = stmt
                        .executeQuery("SELECT * FROM tracking WHERE
imei LIKE '"
                                + imei
```

```
                                    + "' AND DATETIME>='"
                                    + dt1
                                    + "' AND DATETIME <= '" + dt2 + "'"
                                    +" AND lat !=0 and lon != 0"
                                    );
                    // System.out.println("\nIMEI \t\t BUS \t\t\t
VELOCIDAD \n");
                    if (res.last()) {
                        int rows = res.getRow();
                        int i = 0;
                        // Move to beginning
                        res.beforeFirst();

                        output = "";

                        while (res.next()) {

                            imei = res.getString("imei");
                            lat = res.getString("lat");
                            lon = res.getString("lon");
                            datetime = res.getString("datetime");
                            speed = res.getString("speed");
                            rumbo = res.getString("course");

                            // System.out.println(nombre + " \t " + lat +
" \t "+
                            // lon);

                            if (imei != "") {
                                output += "&TRACK&" + i + "&" + alias +
"&"
                                        + datetime + '&' + lat + "&" + lon
+ "&"
                                        + speed + "&" + rumbo + "&\r\n";

                                // System.out.println("Enviando:" +
output);

                            }
                            i++;

                        }
                        // String comprimida;
                        // comprimida = compress(output);
                        byte[] comprimida;
                        comprimida = compressString(output);
                        int tam_paquete=4096;
                        start = "&TRACKSTART&" + alias + "&" + rows + "&"
                                + comprimida.length +
"&"+tam_paquete+"&\r\n";

cliente.write(encoder.encode(CharBuffer.wrap(start)));
                        System.out.println("Enviando:" + start);
                        int paquetes, resto;
                        paquetes = comprimida.length / tam_paquete;
                        resto = comprimida.length % tam_paquete;

                        int resp = -1;
                        int reintentos = 0;
                        boolean salir=false;
                        ByteBuffer bb = ByteBuffer.allocate(10);
```

```
                    for (i = 0; i < paquetes; i++) {
                        System.out.println("Enviando paquete:" + (i +
1)
                                + " de " + (paquetes + 1));
                        cliente.write(ByteBuffer.wrap(comprimida, i *
tam_paquete,tam_paquete));//
                        //
cliente.write(encoder.encode(CharBuffer.wrap(comprimida.substring(i
                        // * 1024, (i + 1) * 1024))));

                        while (!salir) {

//System.out.println("RESPUESTA:"+campos[1]+":"+campos[2]);
                                if(campos[1].equals("ACK")){
                                   i=Integer.parseInt(campos[2]);
                                   campos[1]="";
                                   reintentos=0;
                                   break;

                                }
                                else{
                                    reintentos++;
                                    if(reintentos>30){
                                       salir=true;
                                       break;
                                    }
                                    Thread.sleep(200);

                                }
                        }
                        if(salir==true)break;

                    }
                    if (resto > 0 && salir==false) {
                        //
cliente.write(encoder.encode(CharBuffer.wrap(comprimida.substring(paqu
etes
                        // * 1024))));
                        cliente.write(ByteBuffer.wrap(comprimida, i *
tam_paquete,
                                resto));//
                        System.out.println("Enviando paquete:" + (i +
1)
                                + " de " + (paquetes + 1));
                    }
                }
                res.close();


            }

        } catch (Exception e) {
            e.printStackTrace();

        }
        //System.out.println("Finalizando el hilo:"+this.getName());


    }
    public static String ProcesaComandoAT(String alias, String
comando){
```

```
        MiTracker mt;
        boolean salir=false;

        int reintentos=0;
        mt=misTrackers.getTrackerByAlias(alias);
        campos[1]="";
        campos[2]="";
        try {
            System.out.println("Enviando comando:"+comando);

mt.cliente.write(encoder.encode(CharBuffer.wrap(comando)));
        } catch (CharacterCodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        while (!salir) {

//System.out.println("RESPUESTA:"+campos[1]+":"+campos[2]);
                if(!campos[1].equals("")){
                   return campos[1];
                }
                else if(!campos[2].equals("")){
                    return campos[2];
                }
                else{
                    reintentos++;
                    if(reintentos>30){
                       salir=true;
                       break;
                    }
                    try {
                        Thread.sleep(200);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
        }
        return "NO RESP";



    }

    public static byte[] compressString(String str) {

        byte[] compressedStr = null;

        if (str != null && str != "") {

            GZIPOutputStream gzos = null;

            try {

                ByteArrayOutputStream baos = new
ByteArrayOutputStream();
```

```
                gzos = new GZIPOutputStream(baos);
                gzos.write(str.getBytes());
                gzos.flush();
                gzos.close();
                compressedStr = baos.toByteArray();
            } catch (IOException ioe) {
                System.out.println("Error I/O en la compresion");
                try {
                    gzos.close();
                } catch (IOException io) {
                    System.out.println("Error I/O al cerrar el
stream");
                }
            }
        }
        return compressedStr;
    }
}
```

3.1.3 CLASE "MiTracker"

La clase MiTracker describe un tipo de objeto para registrar a cada uno de los dispositivos que se conectan.

```
package ServiSensor;

import java.nio.channels.SocketChannel;
import java.sql.Timestamp;

public class MiTracker {
     public MiTracker(){
       };//constructor sin parametros
       public String alias;
       public String imei;
       public SocketChannel cliente;
       public Timestamp tiempo;

}
```

3.1.4 CLASE "MisTrackers"

La clase MiTrackers contiene una colección de objetos MiTraker para tener registrados todos los equipos conectados. Contiene además métodos para buscar un equipo por el Alias, para añadir nuevos dispositivos en la lista y para obtener uno en concreto.

```
package ServiSensor;
import java.io.IOException;
import java.nio.channels.SocketChannel;
import java.util.ArrayList;

public class MisTrackers {
     private ArrayList<MiTracker> misTrackers;

     public MisTrackers(){
        misTrackers=new ArrayList<MiTracker>();
     }

    public MiTracker getTrackerByAlias(String alias) {
        MiTracker miTracker;
        for(int i=0; i<misTrackers.size(); i++){
            miTracker=misTrackers.get(i);
            if(alias.equals(miTracker.alias))return miTracker;
        }
        return null;

    }
    public MiTracker get(int i) {
        if(i>=misTrackers.size()){
            return null;
        }
        MiTracker miTracker;
        miTracker=misTrackers.get(i);

        return miTracker;

    }

    public void setTracker(MiTracker tracker) {
        //Compruebo si existia previamente
        MiTracker tr1, tr2;
        boolean encontrado=false;
        //System.out.println("A□endo: imei:"+tracker.imei+"
alias:"+tracker.alias);
        for(int i=0; i<misTrackers.size(); i++){
            tr1=misTrackers.get(i);
            //System.out.println("Pos:"+i+" imei:"+tr1.imei+"
alias:"+tr1.alias+ "timestamp:"+tr1.tiempo.toString());
            if(tracker.imei.equals(tr1.imei)){
                //comprobamos si es el mismo socket o es distinto
                if(tr1.cliente!=tracker.cliente){
                    //compruebo que ese cliente no lo use otro tracker
                    for(int j=0;j<misTrackers.size();j++){
                        tr2=misTrackers.get(j);
                        if((tr2.cliente==tr1.cliente) &&( i!=j)){
```

```
                            encontrado=true;
                            break;
                        }
                    }
                    if(encontrado==false){
                        System.out.println("Cerrando socket:
"+tr1.cliente.toString());
                        try {
                            tr1.cliente.close();
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                }
                //System.out.println("Eliminando tracker:"+tr1.alias);
                misTrackers.remove(i);
                break;
            }
        }


        //Si no existe se a
        misTrackers.add(tracker);
        //System.out.println("A endo tracker:"+tracker.alias);
        //System.out.println("Total sockets:"+ misTrackers.size());
    }

    public int size(){
        return misTrackers.size();
    }

}
```

3.1.5 CLASE "MiCliente"

La clase "MiCliente" describe un objeto para gestionar a cada cliente (aplicación Android) que hace una petición al servidor. Se creara una lista de estos objetos para poder gestionar a varios clientes al mismo tiempo. Cada objeto cliente contendrá un alias, el objeto Socket asociado y el hilo del proceso que lo trata.

```
package ServiSensor;

import java.nio.channels.SocketChannel;

public class MiCliente{
       public MiCliente(){
       };//constructor sin parametros
       String alias;
       SocketChannel cliente;
       ProcesarComando hilo;

}
```

3.2 CODIGO FUENTE APLICACION CLIENTE "SensorTracking"

3.2.1 CLASE "Aplicacion"

La clase "Aplicación" extiende el tipo "Application" y es una clase "base" utilizada para mantener un entorno global de aplicación entre varias "Actividades". Desde cualquier actividad de la aplicación se podrá acceder a los elementos contenidos en esta clase, por lo que se usará pana contener ciertas variables globales.

```
package com.mariocruz.sensortracking;

import java.util.Vector;

import android.app.Application;

import com.google.android.gms.maps.model.MarkerOptions;

public class Aplicacion extends Application {


      public Vector<MarkerOptions> trackItems;
      public Vector<MarkerOptions> graphItems;
      public Number[] serie_temp=new Number[10000];
      public Number[] serie_volt=new Number[10000];
      public Number[] serie_kmh=new Number[10000];
      public Number[] serie_time=new Number[10000];
      public int numItems;


    @Override
    public void onCreate() {
          trackItems = new Vector<MarkerOptions>();
          graphItems= new Vector<MarkerOptions>();
          numItems=0;
    }

}
```

3.2.2 CLASE "MainActivity"

La clase MainActivity es la primera en ejecutarse al iniciar la aplicación. Va asociada al formulario descrito en "activity_login.xml".

Esta clase permite autenticar al usuario con el servidor. Después de producirse una identificación satisfactoria, se lanza la siguiente actividad.

```
package com.mariocruz.sensortracking;


import java.io.BufferedInputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity{

  static AlmacenSQLite miDB;
  EditText e_email,e_passwd;
  TextView t_logerror;
  Button b_salir,b_aceptar;
  static Activity actividad;
  static String Servidor = "campertracking.com";
  //static String Servidor = "192.168.0.9";
  static String CMD_out,CMD_in;

  Thread background;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      String UsuarioDB,PasswDB;
      super.onCreate(savedInstanceState);
      actividad=this;
      requestWindowFeature(Window.FEATURE_NO_TITLE);
      setContentView(R.layout.activity_login);
      e_email=(EditText)findViewById(R.id.e_email);
      e_passwd=(EditText)findViewById(R.id.e_pwd);
      b_salir=(Button)findViewById(R.id.b_salir);
      b_aceptar=(Button)findViewById(R.id.b_aceptar);
      t_logerror=(TextView)findViewById(R.id.t_logerror);
      t_logerror.setText("Introduzca su e-mail y su contraseña.");
      t_logerror.setTextColor(Color.BLACK);
      miDB=new AlmacenSQLite(this);
      UsuarioDB=miDB.getConfig("usuario");
      PasswDB=miDB.getConfig("passwd");
      if(UsuarioDB!=null&&PasswDB!=null){
            e_email.setText(UsuarioDB);
            e_passwd.setText(PasswDB);
```

```
	}

	b_aceptar.setOnClickListener(new OnClickListener() {
		@Override
		public void onClick(View view) {

		Login(e_email.getText().toString(),e_passwd.getText().toStr
		ing());
		}
	});
	b_salir.setOnClickListener(new OnClickListener() {
		@Override
		public void onClick(View view) {
			miDB.close();
			actividad.finish();
		});

	}
	private int Login(String usr,String pwd){
		t_logerror.setText("Conectando con el servidor...");
		t_logerror.setTextColor(Color.BLUE);
		EnvioComando("&LOGIN&"+usr+"&"+pwd+"&\n");

		return 0;
	}
	final Handler handler = new Handler() {
		@Override
		public void handleMessage(Message msg) {

		if(CMD_in.contains("ACCEPTED")){

		    t_logerror.setTextColor(Color.GREEN);
			t_logerror.setText("Conexión Aceptada");

				miDB.setConfig("usuario",e_email.getText().toSt
				ring());

			miDB.setConfig("passwd",e_passwd.getText().toString()
			);
			  miDB.close();
			                    Intent i = new Intent(actividad,
			MapaActivity.class);
			  Log.i("MAPA","LOGIN ACEPTADO");
			  startActivityForResult(i,1);
			}
			else{
				t_logerror.setText("Contraseña o e-mail
			incorrectos.");
				t_logerror.setTextColor(Color.RED);
				Log.i("MAPA","LOGIN DENEGADO");

			}

		}

	};
	@Override
	protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
	    // Check which request we're responding to
	    if (requestCode == 1) {
```

```
            actividad.finish();
        }
    }
    private void EnvioComando(String comando)
    {

        CMD_out=comando;
        background = new Thread(new Runnable() {
            @Override
            public void run() {
              try {
                try {
                    byte[] buf = new byte[2000];
                    int i=0;
                    String[] campos=new String[30];
                    String datos_in="";
                    Socket sk = new Socket(Servidor, 4444);
                    BufferedInputStream entrada = new
                    BufferedInputStream(sk.getInputStream(),
                    300000);
                    //BufferedReader entrada = new
BufferedReader(new InputStreamReader(sk.getInputStream()));
                    PrintWriter salida = new PrintWriter(new
OutputStreamWriter(sk.getOutputStream()),true);
                    salida.println(CMD_out);
                    int resp;
                    boolean primero=true;
                    char ch;
                    int intentos=0;
                    while (true) {
                        if(intentos>10){
                          Log.w("DATOS","ERROR TIME OUT
POS_USR");                          break;
                        }
                    resp=-1;
                    if (entrada.available() > 0) {
                        resp = entrada.read();
                    }
                    if (resp == -1) {
                        Thread.sleep(300);
                        intentos++;
                    continue;
                }
                ch = (char) resp;
                if(primero){
                    if(ch!='&'){
                      continue;
                    }
                    primero=false;
                }
                if(ch==0x0A){
                    Log.i("DATOS","Recibido:"+datos_in);
                    CMD_in = datos_in;
                    break;
                }
                datos_in += ch;
                intentos=0;
            }
            sk.close();
        } catch (Exception e) {
            Log.e("CamperTracking", e.toString(), e);
```

```
        }
        Log.i("MAPA","Enviando mensaje");
        handler.sendMessage(handler.obtainMessage());
} catch (Throwable t) {
}
Log.i("MAPA","Fin del tread:");
}
      });
            background.start();

      }

}
```

3.2.3 CLASE "MapaActivity"

La clase MapaActivity es la clase principal de la aplicación. Contiene el mapa y los botones con los que se accederá a todas las funciones y es la que gestiona la comunicación con el servidor.

```
package com.mariocruz.sensortracking;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;
import java.util.Vector;
import java.util.zip.GZIPInputStream;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint.Style;
import android.graphics.Path;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.ShapeDrawable;
import android.graphics.drawable.shapes.PathShape;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
```

```
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.util.TypedValue;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Button;
import android.widget.ListView;


import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import
com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListener;
import com.google.android.gms.maps.GoogleMap.OnMapClickListener;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;



public class MapaActivity extends FragmentActivity  implements
            OnMapClickListener, OnInfoWindowClickListener,
OnItemClickListener {

      private LatLng MiCoche = new LatLng(39.481106, -0.340987);

      static private GoogleMap mapa;
      //static private MapFragment mapa;
      MarkerOptions mi_tracker_o;
      Marker mi_tracker = null;
      static String Servidor = "campertracking.com";
      //static String Servidor = "192.168.0.9";
      static boolean isRunning = false;
      String[] Campos = new String[20];
      boolean salir = false;
      boolean espera = false;
      Thread background;
      Marker tracker = null;
      static boolean primera_vez = true;

      private String TrackString;
      int n;
      int numreg;
      Runnable changeMessage;
      String mensajeProgreso;
      static ArrayList <MarkerOptions> Marcadores;
      static private ArrayList <Marker> MisTrackers;
      static float mi_zoom=16;
      static View vista;
      ListView mListview;
      AlertDialog mAlertDialog;
      static Activity actividad;
      static Context context;
      static AlmacenSQLite miDB;
```

```
	static String User;
	static int ModoTrack;
	Aplicacion aplicacion;



	Handler progreso = new Handler() {
		@Override
		public void handleMessage(Message msg) {

			changeMessage.run();

		}

	};
	 Handler handler = new Handler() {
		@Override
		public void handleMessage(Message msg) {
			if(!isRunning)return;
		ArrayList <MarkerOptions> marcadores;
		Log.i("MAPA",mapa.toString());
		if(mapa==null){
			Log.i("MAPA","El mapa no esta listo");
			return;

		}

		marcadores=miDB.getEquipos();
		Marcadores=marcadores;
		Marker marcador = null;
		try{
			Log.i("MAPA", "Mensaje Recibido");

			for(int i=0;i<marcadores.size();i++){
				if (primera_vez) {

	marcador=mapa.addMarker(marcadores.get(i));
					MisTrackers.add(marcador);
					if(aplicacion.trackItems.size()>0){
						pintarTrack();
					}
				}
				else{
					for(i=0;i<MisTrackers.size();i++){
					   marcador=MisTrackers.get(i);
					   marcador.remove();

marcador=mapa.addMarker(marcadores.get(i));

//marcador.setPosition(marcadores.get(i).getPosition());
					   MisTrackers.set(i,marcador);
					}
				}
			}
			if(primera_vez){
			  moveCamera(vista);
			  primera_vez = false;
			}
		}catch(Exception e){
			primera_vez=true;
			e.printStackTrace();
```

```
            }


            }

      };

      Handler handler_track = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                  if(!isRunning)return;
                  aplicacion.trackItems = (Vector<MarkerOptions>)
msg.obj;
                  if(aplicacion.numItems>0){
                     switch (ModoTrack){
                           case 1:
                              pintarTrack();
                              break;
                           case 2:
                                pintarGraph();
                                break;
                     }
                  }
            }

      };
      public void pintarTrack(){
            Log.d("MAPA", "Pintando markers ");
            for (int i = 0; i < aplicacion.trackItems.size(); i++) {
                  mapa.addMarker(aplicacion.trackItems.get(i));
            }
            moveCamera(aplicacion.trackItems);

      }
      public void onAttach(Activity activity){
            Log.i("MAPA","On Attach. mapa cargado:"+mapa.toString());
            mapa = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
      }
////////////////////////////////////////////////////////////////////////
//////////////////////////////
////////////////////////////////////////   ON CREATE
//////////////////////////////////////////////
      @Override
      protected void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);
            aplicacion=(Aplicacion)getApplication();
        setContentView(R.layout.activity_mapa);
            salir = true;
            espera=true; //para que el timer espere a que cargue el
mapa
            primera_vez=true;


            //trackItems = new Vector<MarkerOptions>();

            Log.i("MAPA","On create. Cargando el mapa");
            mapa = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
```

```
            //mapa= ((MapFragment)
getFragmentManager().findFragmentById(R.id.map)) .getMap();

            Log.i("MAPA","mapa cargado:"+mapa.toString());
            mapa.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            Log.i("MAPA","antes de set my location enabled:");
            mapa.setMyLocationEnabled(true);
            Log.i("MAPA","despues de set my location enabled:");
            mapa.getUiSettings().setZoomControlsEnabled(false);
            mapa.getUiSettings().setCompassEnabled(true);
            Log.i("MAPA","despues de set compass enabled:");
            miDB=new AlmacenSQLite(this);
            mListview = new ListView(this);
              //Adapter Use in ListView
            ArrayList <String[]> mis_trackers;
            mis_trackers=miDB.getAliasIconos();
            User=miDB.getConfig("usuario");
            actividad=this;
            context=this.getBaseContext();
            MiAdaptador adaptador = new
MiAdaptador(this,mis_trackers,context);
            mListview.setOnItemClickListener(this);
            mListview.setAdapter(adaptador);
            prepareAlertDialog();


            mapa.setInfoWindowAdapter(new
PopupAdapter(getLayoutInflater(),mis_trackers,context));
            Log.i("MAPA","despues de set Info window");
            mapa.setOnInfoWindowClickListener(this);
            Log.i("MAPA","despues de setOnInfoWindowClickListener");
            MisTrackers = new ArrayList <Marker>();
            mapa.setOnMapClickListener(this);
            Log.i("MAPA","despues de setOnMapClickListener");
            Button b_mistracks=(Button)findViewById(R.id.b_centrar);
            b_mistracks.setOnLongClickListener(new
OnLongClickListener() {
              @Override
              public boolean onLongClick(View v) {
                  // TODO Auto-generated method stub
                    //SimpleAlertDialog();
                   ArrayList <String[]> mis_trackers;
                   mis_trackers=miDB.getAliasIconos();
                   MiAdaptador adaptador = new
MiAdaptador(actividad,mis_trackers,context);
                   mListview.setAdapter(adaptador);
                    mAlertDialog.show();
                  return true;
              }
          });
            espera=false;

            Log.i("MAPA","fin de OnCreate");



      }

      public void prepareAlertDialog() {
              WindowManager.LayoutParams lp = new
WindowManager.LayoutParams();
```

```
              AlertDialog.Builder mBuider = new
AlertDialog.Builder(actividad);
              mBuider.setTitle("Mis Trackers");
              mBuider.setPositiveButton("Salir", new
DialogInterface.OnClickListener() {
               @Override
               public void onClick(DialogInterface dialog, int which) {
                     dialog.cancel();

                // Do what you like on Ok Button
               }
              });
              /*
              mBuider.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {

                 @Override
                 public void onClick(DialogInterface dialog, int which)
{
                  dialog.cancel();
                 }
                });
                */

              mBuider.setView(mListview);
              mAlertDialog = mBuider.create();

              lp.copyFrom(mAlertDialog.getWindow().getAttributes());

            }
          @Override
           public void onItemClick(AdapterView<?> adapter, View view,
int position,long log) {
                    // Do what you want on List Item Click

              Marker m=MisTrackers.get(position);

mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(m.getPosition(),
17));
                    mAlertDialog.cancel();
             }




            //This Method For Create Simple AlertDialog In android

            public void SimpleAlertDialog(){
              AlertDialog.Builder builder = new
AlertDialog.Builder(this);
              builder.setMessage("Are you sure you want to exit?")
                     .setCancelable(false)
                     .setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
                          @Override
                               public void onClick(DialogInterface
dialog, int id) {
                          actividad.finish();
                          }
                     })
```

```
                    .setNegativeButton("No", new
DialogInterface.OnClickListener() {
                        @Override
                            public void onClick(DialogInterface
dialog, int id) {
                             dialog.cancel();
                        }
                    });
              AlertDialog alert = builder.create();
              alert.show();
             }

      @Override
      public void onPause() {
            super.onPause();
            Log.i("MAPA","OnPause");
            isRunning = false;
            try {
                  background.join(1000,1);
            } catch (InterruptedException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

      }

      @Override
      public void onResume() {
            super.onResume();
            Log.i("MAPA","OnResume");
            isRunning = true;

      }
      @Override
      public void onStop() {
            super.onStop();
            Log.i("MAPA","OnStop");
            isRunning = false;
      }

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

            Log.i("MAPA","OnStart");
            //if(1==1)return; //para evitar que arranque

            background = new Thread(new Runnable() {
                  @Override
                  public void run() {
                        try {
                              while (isRunning) {
                                    while(espera){
                                          Thread.sleep(500);
                                          Log.i("MAPA","en espera...");
                                    }


                                    try {
                                          byte[] buf = new byte[2000];
                                          int i=0;
```

```
                                        String[] campos=new
String[30];
                                        String datos_in="";


                                        Socket sk = new
Socket(Servidor, 4444);
                                        BufferedInputStream entrada =
new BufferedInputStream(sk.getInputStream(), 300000);
                                        //BufferedReader entrada =
new BufferedReader(new InputStreamReader(sk.getInputStream()));
                                        PrintWriter salida = new
PrintWriter(new OutputStreamWriter(sk.getOutputStream()),true);

      salida.println("&POS_USR&"+User+"&");
      int resp;
      boolean primero=true;
      char ch;
      int intentos=0;
      while (true) {
          if(intentos>10){
                  Log.w("DATOS","ERROR TIME OUT POS_USR");
                  break;
          }
          resp=-1;
          if (entrada.available() > 0) {
             resp = entrada.read();
          }
          if (resp == -1) {
              Thread.sleep(300);
              intentos++;

      continue;
                                                    }
                                                    ch = (char) resp;
                                                    if(primero){


      if(ch!='&'){

continue;
                                                          }

      primero=false;
                                                    }
                                                    if(ch==0x0A){


      Log.i("DATOS","Recibido:"+datos_in);
                                                          campos =
datos_in.split("&");

miDB.insertarEquipo(campos);

if(campos[2].equals(campos[3]))break;
                                                      datos_in="";

                                                    }
                                                    datos_in += ch;
                                                    intentos=0;
                                              }
                                        sk.close();
```

```
                                    } catch (Exception e) {
                                          Log.e("CamperTracking",
e.toString(), e);
                                    }
                                    Log.i("MAPA","Enviando mensaje");

      handler.sendMessage(handler.obtainMessage());
                                    for (int i = 0; i < 20; i++) {
                                          Thread.sleep(500);

      //if(salir==false)progreso.sendMessageAtFrontOfQueue(progreso.ob
tainMessage());
                                          if (!isRunning)
                                                break;
                                    }


                              }
                        } catch (Throwable t) {

                        }
                        Log.i("MAPA","Fin del tread:");
                  }
            });


            isRunning = true;
            background.start();

            try {
                  Thread.sleep(200);
            } catch (Exception e) {
            }

      //mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(MiCoche,
15));
      }

      public static void moveCamera(View view) {
            try{
                  LatLngBounds.Builder builder = new
LatLngBounds.Builder();
                  for (Marker m:MisTrackers) {
                      builder.include(m.getPosition());
                  }

                  LatLngBounds bounds = builder.build();
                  int padding = 100; // offset from edges of the map in
pixels
                  CameraUpdate cu =
CameraUpdateFactory.newLatLngBounds(bounds, padding);
                  mapa.animateCamera(cu);
            }
            catch(Exception e){
                  e.printStackTrace();
            }


      //mapa.animateCamera(CameraUpdateFactory.newLatLngZoom(MisTracke
rs.get(0).getPosition(),mi_zoom));
      }
```

```
	public static void moveCamera(Vector <MarkerOptions> mo) {
		LatLngBounds.Builder builder = new LatLngBounds.Builder();
		for (MarkerOptions m:mo) {
		    builder.include(m.getPosition());
		}
		LatLngBounds bounds = builder.build();
		int padding = 100; // offset from edges of the map in
pixels
		CameraUpdate cu =
CameraUpdateFactory.newLatLngBounds(bounds, padding);
		mapa.animateCamera(cu);


	//mapa.animateCamera(CameraUpdateFactory.newLatLngZoom(MisTracke
rs.get(0).getPosition(),mi_zoom));
	}

	public void borrarMapa(View view) {
		mapa.clear();
		 ArrayList <MarkerOptions> marcadores;
	   marcadores=miDB.getEquipos();
		 for(int i=0;i<marcadores.size();i++){
			Marker marcador;
			marcador=mapa.addMarker(marcadores.get(i));
			MisTrackers.add(marcador);

	//mapa.animateCamera(CameraUpdateFactory.newLatLngZoom(marcador.
getPosition(),mi_zoom));

		}
	}

	public void getTrack() {
		final ProgressDialog myDialog;
		salir = false;
		myDialog = new ProgressDialog(this);
		myDialog.setTitle("CamperTracking");
		myDialog.setMessage("Descargando datos...");
		myDialog.setCancelable(false);
		mensajeProgreso="Conectando...";
		myDialog.setButton(DialogInterface.BUTTON_NEGATIVE,
"Cancelar",
				new DialogInterface.OnClickListener() {
					@Override
					public void onClick(DialogInterface
dialog, int which) {
						dialog.dismiss();
						salir = true;
					}
				});
		myDialog.show();
		changeMessage = new Runnable() {
			@Override
			public void run() {
				String msg;
				msg=mensajeProgreso;
				// Log.v(TAG, strCharacters);
				myDialog.setMessage(msg);
			}
		};
```

```
            Thread background2 = new Thread(new Runnable() {
                  @Override
                  public void run() {
                        try {

                              try {
                                    String[] Campos2 = new String[20];

                                    char[] buf = new char[200];
                                    int i;
                                    int numbytes;
                                    int tam_paquete;
                                    int resp;
                                    char ch;

                                    LatLng miCoche;
                                    MarkerOptions marcador;
                              espera=true;
                                    String cadena;

                                    Path pathFlecha;
                                    Vector<MarkerOptions> track = new
Vector<MarkerOptions>();
                                    Socket sk = new Socket(Servidor,
4444);

                                    pathFlecha = new Path();
                                    pathFlecha.moveTo((float) 0.0, 1);
                                    pathFlecha.lineTo((float) 0.3,
(float) 0.0);

                                    pathFlecha.lineTo((float) 0.6, 1);
                                    pathFlecha.lineTo((float) 0.3,
(float) 0.7);

                                    pathFlecha.lineTo((float) 0.0, 1);
                                    /*
                                     Paint p = new Paint();
                                     p.setColor(Color.RED);
                                        p.setStyle(Paint.Style.FILL);
                                        p.setAntiAlias(true);
                                    */



                                    sk.setReceiveBufferSize(300000);
                                    BufferedInputStream entrada = new
BufferedInputStream(sk.getInputStream(), 300000);
                                    //BufferedOutputStream output = new
BufferedOutputStream(sk.getOutputStream());

                                    PrintWriter salida = new
PrintWriter(new OutputStreamWriter(sk.getOutputStream()),true);
                              cadena=TrackString;
                              Log.d("CamperTracking","cadena");
                                    salida.println(cadena);

                                    String respuesta = "";
                                    int reintentos = 0;
                                    aplicacion.numItems=0;
                                    boolean  primero=true;
                                    while (true) {
                                          resp=-1;
```

```
                                        if (entrada.available() > 0)
{
                                              resp = entrada.read();


                                        }
                                        if (resp == -1) {
                                              // hemos llegado al
final;
                                              if(reintentos>500){
                                                break;
                                              }
                                              reintentos++;
                                              Thread.sleep(300);
                                              continue;
                                        }
                                        ch = (char) resp;
                                        if(primero){
                                              if(ch!='&'){
                                                continue;
                                              }
                                              primero=false;
                                        }

                                        if (ch != 0x0D && ch != 0x0A)
{
                                              respuesta += ch;
                                              reintentos=0;
                                        } else {
                                              break;
                                        }

                                  }

                                  if (respuesta.length() > 0) {

      Log.d("CamperTracking",respuesta);
                                        if (respuesta.charAt(0) !=
'&') {
                                              Log.e("CamperTracking",
                                                    "Error:
respuesta de servidor incorrecta");
                                        }

                                        Campos2 =
respuesta.split("&");

                                        if
(Campos2[1].equals("TRACKSTART")) {

                                              numreg =
Integer.parseInt(Campos2[3]);
                                              numbytes =
Integer.parseInt(Campos2[4]);
      tam_paquete=Integer.parseInt(Campos2[5]);
                                              //
mapa.addMarker(mi_tracker);
                                              Log.d("TRACK",
"Numregs:" + numreg);
```

```
      aplicacion.trackItems.clear();
                                                  respuesta = "";
                                                  String comprimida = "";
                                                reintentos=0;
                                                  salir = false;
                                                  int nu = 0;
                                                  int n=0;

                                                  resp =
entrada.read();//Para quitar el 0x0a que se queda
                                                  byte comprimido[] = new
byte[numbytes];

      mensajeProgreso="Descargando 0 de "+numbytes+" bytes";

      progreso.sendMessageAtFrontOfQueue(progreso.obtainMessage());
                                                  byte[] buffer=new
byte[tam_paquete];

                                                  while (nu < numbytes &&
salir == false) {
                                                        resp=-1;

      //Thread.sleep(50);
                                                        if
(entrada.available() > 0) {
                                                              //resp =
entrada.read();

      resp=entrada.read(buffer);

      Log.d("SOCKET","Recibidos: "+resp);


                                                        }
                                                        if (resp < 0) {
                                                              // hemos
llegado al final;
                                                              if
(reintentos >30) {

      Log.d("ERROR SOCKET","No se recibieron todos los bytes");
                                                                    salir
= true;

                                                              } else {

      reintentos++;


      Log.d("ERROR SOCKET","Reintentando:"+reintentos);


      Thread.sleep(500);

      continue;
                                                              }
                                                              break;
                                                        }
                                                        reintentos = 0;
```

```
                                                    //ch = (char)
resp;
                                                    //comprimida +=
ch;
      //comprimido[nu]=(byte)resp;
                                                    //nu++;
      cadena="&ACK&"+n+"&";
      salida.println(cadena);

      System.arraycopy(buffer, 0, comprimido, nu,resp);
                                                    nu+=resp;
                                                    n++;
                                                    //if(nu%1000==0){
mensajeProgreso="Descargando "+nu+" de "+numbytes+" bytes";
progreso.sendMessageAtFrontOfQueue(progreso.obtainMessage());
Log.d("SOCKET",mensajeProgreso);
                                                    //}
                                                }
      mensajeProgreso="Descargando "+numbytes+" de "+numbytes+"
bytes";

      progreso.sendMessageAtFrontOfQueue(progreso.obtainMessage());

                                                String descomp
=uncompressString(comprimido);
                                    n=descomp.charAt(0);
                                                salir = false;
                                                //Una vez
descomprimido, procesamos el String
                                                int indice=0;
                                                int end;
                                                n=0;
      mensajeProgreso="Procesando "+n+" de "+numreg+" puntos";
                                                for(i=0;i<numreg;i++){

end=descomp.indexOf("\r", indice);
respuesta=descomp.substring(indice, end);
                                                  try {
                                                        Campos2 =
respuesta.split("&");
                                                        respuesta =
"";
                                                        int
val=Integer.parseInt(Campos2[13]);
```

```
      //
      if(ModoTrack==1&&val==0)continue; //Nos saltamos los que no
tengan GPS valido

      aplicacion.numItems++;
                                                              miCoche =
new LatLng(Float

      .parseFloat(Campos2[5]), Float

      .parseFloat(Campos2[6]));
                                                              cadena =
"";
                                                              Double
vtemp=Double.parseDouble(Campos2[11]);
                                                              Double
volt=Double.parseDouble(Campos2[12])/1000;
                                                            String temp;
                                                            vtemp=(vtemp-
5)/10;

if(vtemp>900){
                                                               temp="
Temp: ?? °C";
                                                            }else{
                                                               temp="
Temp:"+String.format("%.1f °C",vtemp);
                                                            }


      SimpleDateFormat inputFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

inputFormat.setTimeZone(TimeZone.getTimeZone("Etc/UTC"));

SimpleDateFormat out = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
                                                                 Date
date = inputFormat.parse(Campos2[4]);
                                                                 cadena
= out.format(date) + "\n Vel:"+ Campos2[7]
                                                                   + "
Rumbo: "    + Campos2[8] + "°"
                                                                   + "
Bat:"+String.format("%.2f V",volt)+ temp;


                              // +"<a href=\""+Campos[7]+"\">Google
Maps</a>";

                                                              int vel =
Integer.parseInt(Campos2[7]);

      if(ModoTrack==2){ //Si se trata de graficos

aplicacion.serie_kmh[i]=vel;

aplicacion.serie_time[i]=date.getTime();

aplicacion.serie_volt[i]=volt;

aplicacion.serie_temp[i]=vtemp;
```

```
                                                                    //No
seguimos procesando el Marker
                                                                }
                                                                else{

      Bitmap bmp;

      Drawable drawableFlecha;

      ShapeDrawable dFlecha = new ShapeDrawable(new
PathShape(pathFlecha, 1, 1));

      if(vel<20){

      dFlecha.getPaint().setColor(Color.CYAN);
                                                                }
                                                                else
if(vel>=20&&vel<60){

      dFlecha.getPaint().setColor(Color.GREEN);
                                                                }
                                                                else
if(vel>=60&&vel<90){

      dFlecha.getPaint().setColor(Color.YELLOW);
                                                                }
                                                                else
if(vel>=90&&vel<120){

      dFlecha.getPaint().setColor(0xFFF7B447);//ORANGE

                                                                }
                                                                else
if(vel>=120&&vel<150){

      dFlecha.getPaint().setColor(Color.RED);

                                                                }
                                                                else
if(vel>=150){

      dFlecha.getPaint().setColor(Color.BLACK);
                                                                }

      dFlecha.getPaint().setStyle(Style.FILL);

      dFlecha.setIntrinsicWidth(50);

      dFlecha.setIntrinsicHeight(50);


      drawableFlecha = dFlecha;

      bmp=fromDrawable(drawableFlecha,24,24,Float.parseFloat(Campos2[8
]));



      marcador = new MarkerOptions()
```

```
        .position(miCoche)
        .title(Campos2[3])
        .snippet(cadena)
        .icon(BitmapDescriptorFactory.fromBitmap(bmp))
        .anchor(0.3f, 0.3f);
        track.add(marcador);
                                                        }
                                                    } catch
(Exception e) {
        e.printStackTrace();
                                                    }
                                                    indice=end+1;
                                                    n++;
                                                    if(n%100==0){
        mensajeProgreso="Procesando "+n+" de "+numreg+" puntos";
        progreso.sendMessageAtFrontOfQueue(progreso.obtainMessage());
                                                    }
                                                }
        mensajeProgreso="Procesando "+n+" de "+numreg+" puntos";
        progreso.sendMessageAtFrontOfQueue(progreso.obtainMessage());
                                                Message msg =
handler_track.obtainMessage();
                                                msg.obj = track;
        handler_track.sendMessage(msg);
                                                Log.d("TRACK",
track.size()
                                                        + " Markers
añadidos");
                                            }
                                        }
                                        else{
                                            Log.e("CamperTracking","NO
HAY RESPUESTA A LA PETICION");
                                        }
                                        sk.close();
                                    }
                                    catch (Exception e) {
                                    }
                                }
                                catch (Throwable t) {
                                }
```

```
                        myDialog.dismiss();
                        salir=true;
                        espera=false;
                    }
                });
                background2.start();
                changeMessage.run();

        }
        protected Bitmap fromDrawable(final Drawable drawable, final int
height, final int width,final float rotation) {
            final int widthDip = (int) TypedValue.applyDimension(1, width,
getResources()
                    .getDisplayMetrics());
             final int heightDip = (int) TypedValue.applyDimension(1,
width, getResources().getDisplayMetrics());
            final Bitmap bitmap = Bitmap.createBitmap(width, height,
Config.ARGB_8888);
            final Canvas canvas = new Canvas(bitmap);
            drawable.setBounds(0, 0, canvas.getWidth(),
canvas.getHeight());
            drawable.draw(canvas);
            Matrix matrix = new Matrix();
                matrix.postRotate(rotation,width/2,height/2);  // La
rotación debe ser decimal (float o double)
                Bitmap rotatedBitmap = Bitmap.createBitmap(bitmap, 0, 0,
bitmap.getWidth(), bitmap.getHeight(), matrix, true);
            return rotatedBitmap;
        }
        public static String uncompressString(byte[] compressedStr) {

                String str = null;

                if (compressedStr != null && compressedStr.length > 0) {

                    GZIPInputStream gzis = null;

                    try {

                        ByteArrayInputStream bais = new
ByteArrayInputStream(
                                compressedStr);

                        gzis = new GZIPInputStream(bais);

                        ByteArrayOutputStream baos = new
ByteArrayOutputStream();

                        byte[] buffer = new byte[1024];

                        int size = 0;

                        while ((size = gzis.read(buffer)) > 0) {

                            baos.write(buffer, 0, size);

                        }

                        gzis.close();

                        baos.close();
```

```
                        str = new String(baos.toByteArray());
                  } catch (IOException ioe) {
                        System.out.println("Error I/O en la
descompresion");
                        try {
                              gzis.close();
                        } catch (IOException io) {
                              System.out.println("Error I/O al cerrar
el stream");
                        }
                  }
            }
            return str;
      }
    public void pintarGraph(){
      Intent i = new Intent(this, GraphActivity.class);
          startActivityForResult(i, 3);
    }
      public void addTrack(View view) {
            Intent i = new Intent(this, Track.class);
            startActivityForResult(i, 1);
      }
      public void addGraph(View view) {
            Intent i = new Intent(this, Track.class);
            startActivityForResult(i, 2);
      }
      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
            if (requestCode == 1||requestCode == 2) {
                  if (resultCode == RESULT_OK) {
                        TrackString = data.getStringExtra("TRACK");
                        ModoTrack=requestCode;
                        getTrack();
                  }
                  if (resultCode == RESULT_CANCELED) {
                        // Write your code if there's no result
                  }
            }
      }// onActivityResult
```

```
	@Override
	public void onMapClick(LatLng puntoPulsado) {
		mapa.addMarker(new
MarkerOptions().position(puntoPulsado).icon(
				BitmapDescriptorFactory

	.defaultMarker(BitmapDescriptorFactory.HUE_YELLOW)));
	}

	@Override
	public void onInfoWindowClick(Marker arg0) {
		// TODO Auto-generated method stub

	}
}
```

3.2.4 Clase "GraphActivity"

La clase GraphActivity utiliza la libreria AndroidPlot para dibujar las graficas a partir de los datos almacenados en listas dentro de la clase "Aplicacion".

```
package com.mariocruz.sensortracking;

import java.text.DecimalFormat;
import java.text.FieldPosition;
import java.text.Format;
import java.text.ParseException;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;

import com.androidplot.xy.LineAndPointFormatter;
import com.androidplot.xy.PointLabelFormatter;
import com.androidplot.xy.SimpleXYSeries;
import com.androidplot.xy.XYPlot;
import com.androidplot.xy.XYSeries;
import com.google.android.gms.maps.model.MarkerOptions;

public class GraphActivity extends Activity
{

    private XYPlot plot;
    private Aplicacion aplicacion;
    static long Intervalo;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        aplicacion=(Aplicacion)getApplication();


        setContentView(R.layout.activity_graph);

        // inicializamos la referencia a XYPlot:
        plot = (XYPlot) findViewById(R.id.mySimpleXYPlot);
        plot.getLegendWidget().setVisible(false);
        for(int i=aplicacion.numItems;i<10000;i++){
              aplicacion.serie_temp[i]=null;
              aplicacion.serie_time[i]=null;
              aplicacion.serie_volt[i]=null;
              aplicacion.serie_kmh[i]=null;
        }


pintaGrafico(aplicacion.serie_temp,aplicacion.serie_time,"Temperatura
(C)");

    }
    public void grafTemp(View view) {
        plot.clear();

pintaGrafico(aplicacion.serie_temp,aplicacion.serie_time,"Temperatura
(C)");
        plot.redraw();
    }
    public void grafVolt(View view) {
       plot.clear();

pintaGrafico(aplicacion.serie_volt,aplicacion.serie_time,"Tensi□ateria
(V)");
       plot.redraw();
    }
    public void grafVel(View view) {
      plot.clear();

pintaGrafico(aplicacion.serie_kmh,aplicacion.serie_time,"Velocidad
(Km/h)");
      plot.redraw();
   }
    private void pintaGrafico(Number[] serie,Number[] tiempo,String
texto){
        // convertimos los arrays en XYSeries':

        XYSeries series1 = new SimpleXYSeries(
                Arrays.asList(tiempo),
                Arrays.asList(serie),      // SimpleXYSeries toma una
List como parametro
                texto);                   // establece el titulo de la
serie


        // Creamos y formateamos una serie usando LineAndPointRenderer
```

```
        // se configura desde el xml:
        LineAndPointFormatter series1Format = new
LineAndPointFormatter();
        series1Format.setPointLabelFormatter(new
PointLabelFormatter());
        series1Format.configure(getApplicationContext(),
                R.xml.line_point_formatter_with_plf1);

        // a□mos nueva serie al xyplot:
        plot.addSeries(series1, series1Format);
        plot.setTitle(texto);
        // reducimos el numero de etiquetas
        plot.setTicksPerRangeLabel(3);
        plot.getGraphWidget().setDomainLabelOrientation(-45);
        plot.getGraphWidget().setRangeValueFormat(new
DecimalFormat("0"));
        Intervalo=(tiempo[aplicacion.numItems-1].longValue()-
tiempo[0].longValue())/3600000;



        plot.setDomainValueFormat(new Format() {


            private SimpleDateFormat dateFormat;
            @Override
            public StringBuffer format(Object obj, StringBuffer
toAppendTo, FieldPosition pos) {
                if( Intervalo<=2){
                      dateFormat    = new SimpleDateFormat("mm");
                      plot.setDomainLabel("Minutos");
                }
                if( Intervalo<=72){
                  dateFormat    = new SimpleDateFormat("HH");
                  plot.setDomainLabel("Horas");
                }
                else if(Intervalo>72&&Intervalo<744){
                     dateFormat = new SimpleDateFormat("dd");
                    plot.setDomainLabel("Dias");
                }
                else {
                    dateFormat  = new SimpleDateFormat("MM");
                    plot.setDomainLabel("Meses");
                }

                long timestamp = ((Number) obj).longValue();// * 1000;
                Date date = new Date(timestamp);
                return dateFormat.format(date, toAppendTo, pos);
            }

            @Override
            public Object parseObject(String source, ParsePosition
pos) {
                return null;

            }

        });


    }
```

```
}
```

3.2.5 Clase "AlmacenSQLite"

Esta clase gestiona una base de datos SQLite para almacenar datos persistentes en la aplicación. Almacena la última posición y valores leídos de cada equipo remoto y algunos parámetros de configuración.

```
package com.mariocruz.sensortracking;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.mariocruz.sensortracking.R;

public class AlmacenSQLite extends SQLiteOpenHelper {
    // SQLiteOpenHelper
    Context context;
    public AlmacenSQLite(Context context) {
        super(context, "campertracking", null, 1);
        this.context=context;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // db.execSQL("CREATE TABLE puntuaciones ("
        // + "_id INTEGER PRIMARY KEY AUTOINCREMENT, "
        // + "puntos INTEGER, nombre TEXT, fecha LONG)");

        try {
            db.execSQL("CREATE TABLE equipos ("
                    + "alias TEXT PRIMARY KEY ,fecha_hora DATETIME,lat
DOUBLE, lon DOUBLE,speed DOUBLE,course DOUBLE,altitud DOUBLE,lbs
TEXT,icon TEXT,mv1 DOUBLE,mv2 DOUBLE)");

            db.execSQL("CREATE TABLE config (_id INTEGER PRIMARY KEY
AUTOINCREMENT,clave TEXT,valor TEXT)");

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // En caso de una nueva version hay que actualizar las tablas

    }

    // M
    public int insertarEquipo(String[] campos) {
        SQLiteDatabase db = getWritableDatabase();

        //Enviando:&POS&1&3&TRACKER-407&2013-06-20
11:20:11.0&40.2518&-
3.69459&0&96.2&&&http://maps.google.es/?q=40.25182%20-3.69459&


        try {
            String consulta = "INSERT OR IGNORE INTO equipos VALUES
('"
                    + campos[4] + "','" + campos[5] + "','" +
campos[6] + "','"
                    + campos[7] + "','" + campos[8] + "','" +
campos[9] + "','"
                    + campos[10] + "','" + campos[11] +
"','"+campos[13]+"','"+campos[14]+"','"+campos[15]+"')";
            db.execSQL(consulta);
            consulta = "UPDATE equipos SET fecha_hora = '" + campos[5]
                    + "',lat=" + campos[6] + ",lon=" + campos[7] + ",
speed="
                    + campos[8] + ",course=" + campos[9] + ",altitud="
                    + campos[10] + ",lbs='" + campos[11] + "',icon='"
+ campos[13]+ "',mv1='" + campos[14]+ "',mv2='" + campos[15]+ "' WHERE
alias LIKE '"
                    + campos[4]+"'";
            db.execSQL(consulta);
        } catch (Exception e) {
            e.printStackTrace();
            return -1;
        }

        return 1;
        // + nombre + "', " + fecha + ")");
    }
    public int configEquipo(String alias,String icono){
        SQLiteDatabase db = getWritableDatabase();
        String consulta = "UPDATE equipos SET icon = '"+icono+"' WHERE
alias LIKE '"+alias+"'";
        db.execSQL(consulta);
        return 1;
    }

    public int borrarEquipo(String alias) {
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM equipos WHERE alias LIKE '" + alias +
"'");
        return 1;
    }

    public ArrayList<MarkerOptions> getEquipos() {
```

```
        ArrayList<MarkerOptions> lista = new
ArrayList<MarkerOptions>();
        SQLiteDatabase db = getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT * FROM equipos", null);
        while (cursor.moveToNext()) {
        int Id=
context.getResources().getIdentifier(cursor.getString(8),
"drawable",context.getPackageName());
        if(Id==0)Id=R.drawable.ico01;
        //Id=R.drawable.ico01;
            BitmapDescriptor icono =
BitmapDescriptorFactory.fromResource(Id);
            float lat=Float.parseFloat(cursor.getString(2));
            float lon=Float.parseFloat(cursor.getString(3));
            LatLng miCoche = new LatLng(lat,lon);
            String cadena = "";
            SimpleDateFormat inputFormat = new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss",Locale.ROOT);
            inputFormat.setTimeZone(TimeZone.getTimeZone("Etc/UTC"));
            SimpleDateFormat out = new SimpleDateFormat("yyyy/MM/dd
HH:mm:ss");
            Date date;
            Double vtemp=cursor.getDouble(9);
            String temp;
            if(vtemp>900){
               temp=" Temp: ?? C";
            }else{
               temp=" Temp:"+String.format("%.1f
C",(cursor.getDouble(9)-5)/10);
            }
            try {
                date = inputFormat.parse(cursor.getString(1));
                cadena = out.format(date) + "\n Vel:" +
cursor.getString(4)
                        + "Km/h   Rumbo: " + cursor.getString(5) + "
"+" Bat:"+String.format("%.2f V",cursor.getDouble(10)/1000)
                        +temp;// +"\n"
                MarkerOptions marcador = new
MarkerOptions().position(miCoche)

.title(cursor.getString(0)).snippet(cadena).icon(icono)
                        .anchor(0.3f, 0.3f);
                lista.add(marcador);
            } catch (ParseException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }



        }
        cursor.close();
        return lista;
    }
    public ArrayList<String[]> getAliasIconos() {

        ArrayList<String[]> lista = new ArrayList<String[]>();

        SQLiteDatabase db = getReadableDatabase();
```

```
        Cursor cursor = db.rawQuery("SELECT * FROM equipos", null);
        while (cursor.moveToNext()) {
            String[] campos= new String[2];
            campos[0]=cursor.getString(0);
            campos[1]=cursor.getString(8);

            lista.add(campos);

        }
        cursor.close();
        return lista;
    }
    public String getConfig(String clave) {
        String result;
        SQLiteDatabase db = getReadableDatabase();
        Cursor cursor = db.rawQuery("SELECT valor FROM config WHERE
clave LIKE '"+clave+"'",null);
        if(cursor.moveToNext()==false)return (null);
        result=cursor.getString(0);
        cursor.close();
        return result;
    }
    public int setConfig(String clave,String valor) {


        SQLiteDatabase db = getWritableDatabase();
        try {
        db.execSQL("INSERT OR IGNORE INTO config VALUES
(null,'"+clave+"','"+valor+"')");
        db.execSQL("UPDATE config SET valor='" + valor
                + "' WHERE clave LIKE '"+clave+"'");
        }
        catch(Exception e){
            return -1;
        }
        return 1;
    }

}
```

3.3 INTERFAZ DE USUARIO DE LA APLICACION CLIENTE

3.3.1 DEFINICION XML ACTIVIDAD "LOGIN"

Este es el fichero xml que describe a la actividad LOGIN. Su contenido se detalló en la sección 2.8.1.3.

```
<?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:gravity="top"
```

```
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="Sensor-Tracking v1.0"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="242dp"
        android:layout_marginTop="10dp"
        android:src="@drawable/mapterm" />

    <TextView
        android:id="@+id/t_logerror"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:gravity="center_horizontal"
        android:text="Contraseña o e-mail incorrectos."
        android:textColor="#FF4444" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="40dp" >

        <TextView
            android:id="@+id/TextView01"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="4"
            android:gravity="center"
            android:text="Email:" />

        <EditText
            android:id="@+id/e_email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:ems="10"
            android:inputType="textEmailAddress|textAutoComplete" >

            <requestFocus />
        </EditText>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="40dp" >

        <TextView
            android:id="@+id/textView2"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="4"
            android:gravity="center"
            android:text="Password:" />

        <EditText
            android:id="@+id/e_pwd"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:ems="10"
            android:inputType="textPassword" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="top"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="20dp"
        android:gravity="bottom|center_vertical|top" >

        <Button
            android:id="@+id/b_salir"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.02"
            android:text="Salir" />

        <Button
            android:id="@+id/b_aceptar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0.02"
            android:text="Aceptar" />

    </LinearLayout>

</LinearLayout>
```

3.3.2 DEFINICION XML ACTIVIDAD “MAPA”

Este es el fichero xml que describe a la actividad MAPA. Su contenido se detalló en la sección 2.8.1.32.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    tools:context=".MapaActivity" >

    <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        class="com.google.android.gms.maps.SupportMapFragment" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="52dp" >

        <Button
            android:id="@+id/b_centrar"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:onClick="moveCamera"
            android:drawableLeft="@drawable/ico09"
         />

        <Button
            android:id="@+id/b_limpiar"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:onClick="borrarMapa"
            android:drawableLeft="@drawable/clear"

         />

        <Button
            android:id="@+id/b_track"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:drawableLeft="@drawable/track"
            android:onClick="addTrack" />

        <Button
            android:id="@+id/b_graph"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:drawableLeft="@drawable/graph"
            android:onClick="addGraph" />

    </LinearLayout>

</LinearLayout>
```

3.3.3 DEFINICION XML ACTIVIDAD "SELECCIÓN"

Este es el fichero xml que describe a la actividad SELECCIÓN. Su contenido se detalló en la sección 2.8.1.3.

```
<?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:layout_gravity="top"
    android:gravity="bottom"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="top"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Vehiculo:"

android:textAppearance="?android:attr/textAppearanceMedium" />

        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Medium Text"

android:textAppearance="?android:attr/textAppearanceMedium" />

        <Button
            android:id="@+id/b_tracker"
            style="?android:attr/buttonStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="" />

    </LinearLayout>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Desde:"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:orientation="horizontal" >

        <CalendarView
            android:id="@+id/calendarView1"
            style="@style/Widget.CalendarView.Custom"
            android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:layout_weight="1"

        />

        <TimePicker
            android:id="@+id/timePicker1"
            android:layout_width="155dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_margin="00dp">
            <requestFocus/>
        </TimePicker>


    </LinearLayout>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hasta:"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:orientation="horizontal" >

        <CalendarView
            android:id="@+id/calendarView2"
            style="@style/Widget.CalendarView.Custom"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1" />

        <TimePicker
            android:id="@+id/timePicker2"
            android:layout_width="151dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:textSize="10sp">
            <requestFocus />
        </TimePicker>


    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom" >

        <Button
            android:id="@+id/b_acept"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_weight="1"
            android:text="Cancelar" />
```

```
        <Button
            android:id="@+id/b_cancel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_weight="1"
            android:text="Aceptar" />
    </LinearLayout>

</LinearLayout>
```

3.3.4 OTROS FICHEROS XML IMPORTANTES

3.3.4.1 ELEMENTO LISTA

Este layout se utiliza como formato de cada elemento de las listas desplegables. Contiene una imagen y un texto.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:alpha="1"
    android:background="#FFFFFFFF"
    android:showDividers="beginning" >

    <ImageView
        android:id="@+id/icono"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <TextView
        android:id="@+id/alias"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:text="TRACKER-MARIO" />

</LinearLayout>
```

3.3.4.2 POPUP

Este fichero define el format de las viñetas popup que se muestran al pulsar sobre un elemento del mapa. Contiene una imagen, un título y un texto.

```
<?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="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:padding="2dip"
        android:src="@drawable/ico09"
        android:contentDescription="@string/icon"/>

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

        <TextView
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="25sp"
            android:textStyle="bold"/>

        <TextView
            android:id="@+id/snippet"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15sp"/>
    </LinearLayout>

</LinearLayout>
```

3.4 ANDROID MANIFEST

Androidmanifest.xml es un fichero xml que contiene información esencial de la aplicación para el sistema Android. Entre otras cosas, especifica:

- El nombre de paquete de la aplicación.
- Los componentes de la aplicación, actividades, servicios, "broadcast receivers", etc.
- Los permisos que la aplicación necesita y el usuario debe aprobar.
- El mínimo nivel de API necesario.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mariocruz.sensortracking"
    android:versionCode="1"
    android:versionName="1.0"
    >
```

```
    <permission

android:name="com.mariocruz.sensortracking.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="com.mariocruz.sensortracking.permission.MAPS_RECEIVE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVIC
ES" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="18" />
    <uses-feature android:glEsVersion="0x00020000"
android:required="true"/>

    <application
         android:name="Aplicacion"
        android:allowBackup="true"
        android:icon="@drawable/icomap"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyC03i-M_GHNGs3WyzTxDMhA6jMMJMC4cRo" />
        <activity
            android:name="com.mariocruz.sensortracking.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
android:name="com.mariocruz.sensortracking.MapaActivity">

        </activity>
         <uses-library android:name="com.google.android.maps" />
         <activity android:name="Track"></activity>
         <activity android:name="GraphActivity"></activity>

    </application>


</manifest>
```

4. PLIEGO DE CONDICIONES

Tanto para el pliego de condiciones como para el presupuesto se va a suponer la instalación del sistema de tele medida para supervisar a una flota imaginaria de diez camiones frigoríficos.

4.1 CONDICIONES GENERALES

El sistema comprende tres elementos físicos independientes, que serán adquiridos a terceros:

- Equipos remotos
- Servidor
- Equipos Android

El proyecto en sí se basa en el software a realizar más que en los propios equipos, pero se deberán respetar las condiciones de uso y de eliminación de residuos establecidas por cada fabricante, así como el cumplimiento de las normativas aplicables y la legislación vigente.

4.2 CONDICIONES DE MATRIALES Y EQUIPOS

- EQUIPOS REMOTOS: Los equipos remotos serán los suministrados por la empresa IngeniApp, modelo Itrk-101. La instalación de los equipos en los vehículos correrá a cuenta del cliente. Los equipos se entregarán al cliente pre-configurados para conectarse al servidor del proyecto. Los equipos remotos requerirán un SIM de telefonía móvil con servicio de datos activos. Estos SIM los deberá contratar el cliente con la operadora de telefonía que desee, informando de la elección con antelación para poder pre-configurar los equipos con los parámetros propios del operador.

- SERVIDOR: El servidor que se utilizará para albergar la aplicación y la base de datos general será un VPS suministrado por la empresa OVH. Para las dimensiones de este proyecto es suficiente con el plan más económico, (VPS CLASSIC)

Este servidor correrá Debian Linux, y tendrá estos requisitos mínimos.

- Procesador 1gCore
- Ram 1GByte
- HDD 10 Gbytes RAID 10

  - Ancho de banda 100Mbps
  - Acceso “root” completo.

En caso de ampliar el sistema, el VPS es fácilmente escalable contratando un plan superior. Toda la instalación del software y la puesta en marcha del servidor serán realizadas por el contratista.

- Equipos Android. Se suministrarán dos tabletas modelo Nexus 7 con GPS y GPRS para los operarios del sistema. Aparte de estos equipos suministrados, el cliente podrá instalar la aplicación en cualquier dispositivo Android, pero se recomiendan estas características mínimas.
  - Procesador doble núcleo
  - 1Gb de RAM
  - Android 4.0 o superior

5. PRESUPUESTO

Como en el apartado anterior, se partirá del supuesto de la instalación del sistema para una flota de diez camiones frigoríficos.

5.1 COSTE DEL MATERIAL

5.1.1 EQUIPOS SUMINISTRADOS AL CLIENTE

Aquí se detalla el coste de los equipos que se suministrarán al cliente.

Concepto	Precio Unitario	QTY.	TOTAL
Localizador con sensores Itrk-101	240€	10	2400€
Tablet Nexus 7	280€	2	560€
TOTAL			2960€

5.1.2 EQUIPOS Y SOFTWARE EMPLEADOS EN EL DESARROLLO

En este apartado se incluye la parte proporcional al uso de equipos informáticos y software durante el desarrollo del proyecto teniendo en cuenta un periodo de amortización de 4 años por equipo.

Concepto	Precio	Tiempo de uso	TOTAL
PC Intel	800€	12 meses	200€
Tablet Nexus 7	280€	12 meses	70€
Localizadores iTrk-101 (3)	720€	12 meses	180€
Servidor VPS Classic OVH	2€/mes	12 meses	24€
Tarjetas SIM datos (3)	3€/mes	12 meses	36€
Windows 7 Pro	140€	12 meses	35€
Material de oficina			50€
Encuadernación			80€
TOTAL			**675€**

Concepto	TOTAL
Equipos Suministrados	2960€
Equipos Utilizados	675€
TOTAL COSTE MATERIAL	3635€

5.2 COSTE DE MANO DE OBRA

En este apartado se contabilizan las horas de trabajo del ingeniero que realizó el análisis, especificaciones, programación, pruebas y documentación del proyecto.

Concepto	Precio Unitario	QTY.	TOTAL
Ingeniero técnico	50€	900	45000€
TOTAL MANO DE OBRA			45000€

5.3 PRESUPUESTO DE EJECUCION MATERIAL

El presupuesto de ejecución material es la suma del coste de los materiales y el de la mano de obra.

Concepto	TOTAL
COSTE MATERIAL	3635€
MANO DE OBRA	45000€
TOTAL PRESUPUESTO EJECUCION MATERIAL	**48635€**

5.4 IMPORTE DE EJECUCION POR CONTRATA

En este apartado se aplica un recargo del 22% al importe del presupuesto de ejecución material. Este recargo comprende los gastos derivados del uso de instalaciones, gastos de administración, servicios, cargas fiscales y gastos financieros. También comprende el beneficio industrial del proyecto.

Concepto	TOTAL
EJECUCION MATERIAL	**48635,00€**
GASTOS Y BENEFICIO INDUSTRIAL 22%	10709,70€
TOTAL EJECUCION POR CONTRATA	**59334,70€**

5.5 TOTAL PRESUPUESTO

El presupuesto total lo obtenemos a partir del importe de ejecución por contrata, añadiendo los honorarios del ingeniero y el IVA.

Concepto	TOTAL
TOTAL EJECUCION POR CONTRATA	**59334,70€**
HONORARIOS INGENIERO 7%	4153,42€
SUBTOTAL	**63488,12€**
IVA 21%	13332,50€
TOTAL PRESUPUESTO	**76820,62**

El presupuesto total asciende a **SETENTA Y SEIS MIL OCHOCIENTOS VEINTE EUROS CON SESENTA Y DOS CENTIMOS**

En Pinto, a 15 de Julio de 20015

Fdo.

Mario Cruz Sánchez

Ingeniero Técnico de Telecomunicación

6. MANUAL DE USUARIO

6.1 INTRODUCCION

La aplicación SensorTracking es una herramienta diseñada controlar desde un dispositivo Android a una serie de equipos remotos y poder obtener información de los mismos.

Este manual pretende ser una guía simple y práctica para el usuario de la aplicación.

6.2 INSTALACION DE LA APLICACIÓN

Para instalar la aplicación, accedemos al fichero APK que previamente habremos descargado de internet. Al ejecutar el APK se lanzará el Instalador de Paquetes de Android, que solicitará la aprobación de los permisos requeridos por la aplicación.

Ilustración 52: Pantalla de instalación del APK

Pulsando INSTALAR acepta que la aplicación acceda a los elementos mostrados en la ilustración anterior y la aplicación se instala en el dispositivo.

6.3 ACCESO A LA APLICACIÓN

Para lanzar la aplicación, simplemente hay que pulsar el icono de acceso directo que se habrá creado en la lista de aplicaciones del terminal.

El acceso a la aplicación se realiza mediante una pantalla de registro que aparece nada más arrancar la aplicación. En ella se debe introducir el usuario y la clave que le haya facilitado el administrador del sistema.

Ilustración 53: Pantalla de acceso a la aplicación.

Para acceder a la aplicación, introduzca su nombre de usuario (email) y contraseña y pulse Aceptar. Si el usuario y la contraseña están registrados en el servicio, la aplicación le dará paso a la pantalla principal.

6.3 VISTA EN TIEMPO REAL

Después de conseguir acceso a la aplicación accederá directamente a la pantalla principal con el mapa. En esta pantalla se mostrarán todos los vehículos asignados al usuario que inició la sesión con un nivel de zoom tal que todos queden encuadrados.

Al pulsar sobre cualquiera de los vehículos, se mostrará una viñeta con información detallada del estado del mismo, como muestra la siguiente imagen.

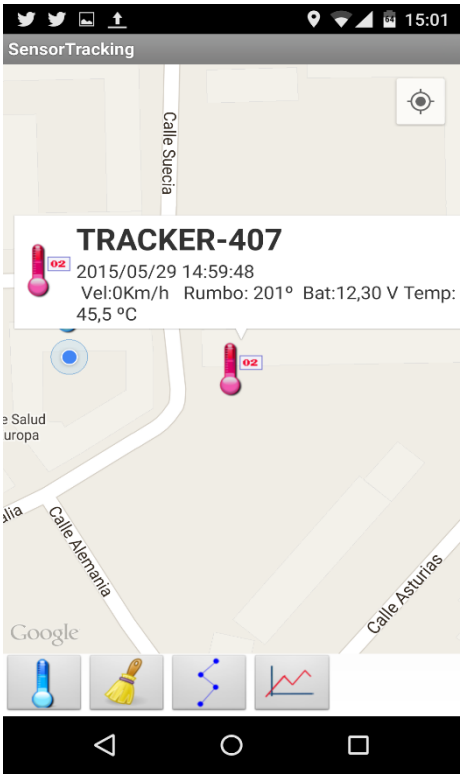

Ilustración 54: Información detallada en tiempo real

En esta pantalla se puede acceder a distintas funciones pulsando en los botones de la parte inferior. La función de cada botón es la que se indica en la Ilustración 50.

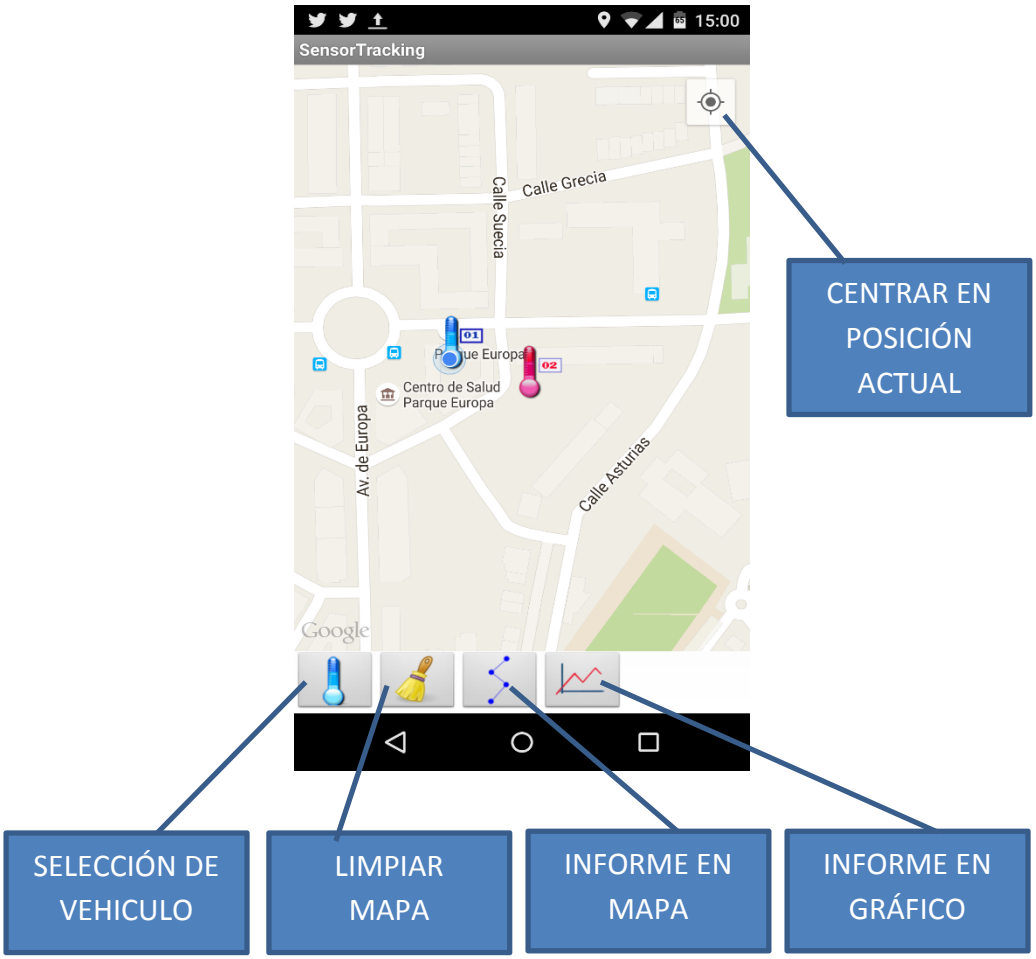

Ilustración 56: Funcionalidad de los botones de la pantalla principal.

- **SELECCIÓN DE VEHICULO:** Pulsando este botón de forma normal se vuelve a ajustar el zoom y la posición del mapa de forma que se encuadren todos los vehículos asignados al usuario. Si se mantiene pulsado durante dos segundos aparece un listado de vehículos para seleccionar uno. En este caso, el zoom y la posición del mapa se centrarán en el vehículo que se haya seleccionado.

  También existe un botón en la parte superior derecha para centrar en mapa en la posición donde se encuentre el usuario.

- **LIMPIAR MAPA:** Este botón elimina cualquier información del mapa excepto la posición en tiempo real de los vehículos.
- **INFORME EN MAPA:** Con este botón se obtiene un informe de un determinado vehículo durante un intervalo de tiempo mostrando la información sobre el mapa.
- **INFORME EN GRÁFICO:** Con este botón se obtiene un informe de un determinado vehículo durante un intervalo de tiempo mostrando la información en forma de grafico de líneas.

6.4 INFORME SOBRE MAPA

Para obtener un informe sobre mapa, después de pulsar el botón que da acceso a la funcionalidad, debe seleccionar el vehículo y el intervalo de tiempo de su interés.

Esto se realizara desde una pantalla como la que se muestra a continuación.

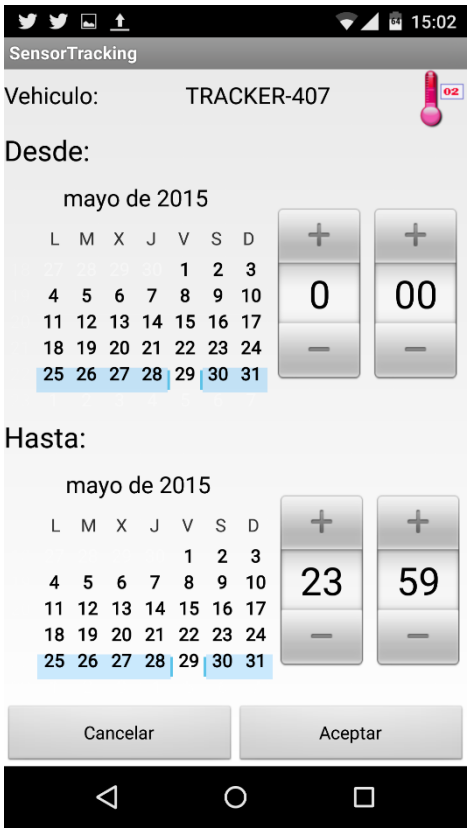

Ilustración 57: Pantalla de selección de parámetros de consulta

Seleccione el vehículo en la parte superior y los intervalos de fecha y hora en los controles “Desde” y “Hasta” y pulse en “Aceptar”. La aplicación tomará un tiempo para obtener los datos deseados y a continuación los mostrará en el mapa como una secuencia de puntos.

Cada punto tendrá forma de flecha, indicando el rumbo del vehículo y el color representará la velocidad instantánea en el momento de la captura. Pulsando en cualquiera de los puntos se puede obtener toda la información relativa a la muestra mediante una viñeta.

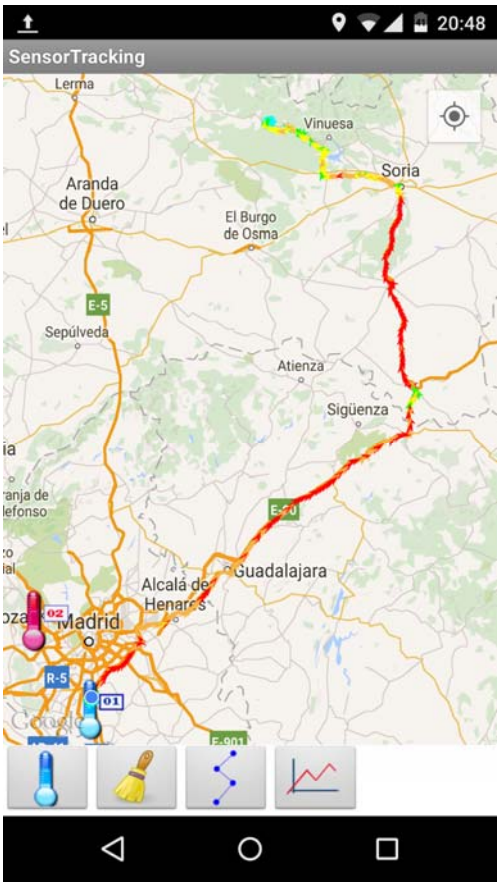

Ilustración 58: Representación de informe sobre el mapa.

6.5 INFORME SOBRE GRAFICA

Para obtener un informe en forma de grafico hay que pulsar el botón correspondiente y realizar los mismos pasos que en caso del informe en mapa para acotar los límites. El resultado se mostrara en una pantalla con tres botones, con los que podremos seleccionar la magnitud que se representará en el gráfico.

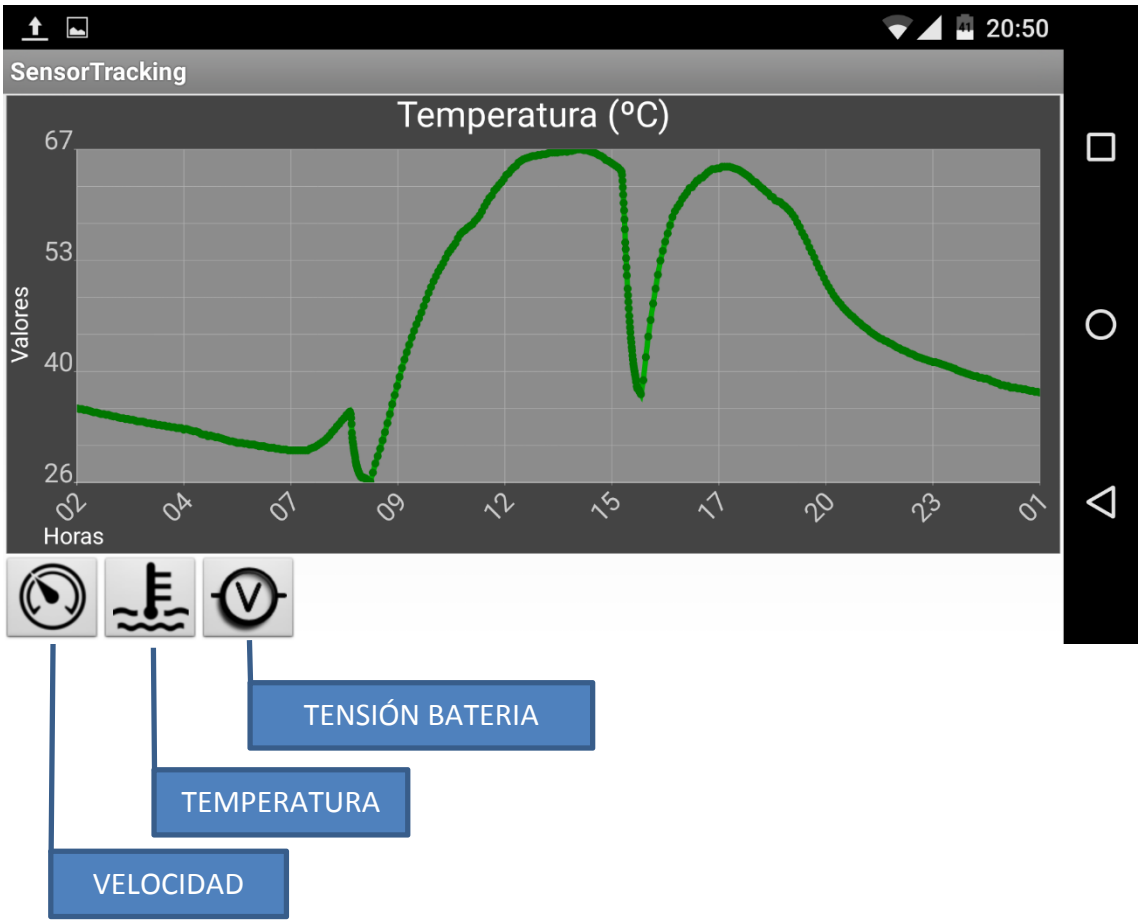

Ilustración 59: Informe sobre gráfico.

7. BIBLIOGRAFIA

- Curso Básico de Android, por la Universidad Politécnica de Valencia
  - https://www.miriadax.net/web/android_programacion

- Curso Avanzado de Android: Universidad Politécnica de Valencia
  - http://www.androidcurso.com/

- El Gran Libro de Android Avanzado
  - Jesús Tomás, Vicente Carbonell, Carsten Vogt, Miguel García, Jordi Bataller, Daniel Ferri

- Guía oficial de desarrollo de Android SDK
  - http://developer.android.com/guide/index.html

- Google Developers. Guía de google para desarrolladores. Información sobre la API de Google Maps.
  - https://developers.google.com/maps/?hl=es

- Wikipedia, varios artículos.
  - https://es.wikipedia.org/wiki/Android
  - https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

- Stack Overflow. Foro de referencia para cualquier consulta de programación en cualquier lenguaje. http://stackoverflow.com/

- El Debian de Pepe: Como instalar JRE en Debian (instalación de Java)
  - https://eldebiandepepe.wordpress.com/2012/09/09/como-instalar-jre-x64-64-bits-en-debian
- AndroidPlot: Pagina de la librería utilizada para dibujar las gráficas.
  - http://androidplot.com/