UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITECNICA SUPERIOR

Ingeniería Técnica de Sistemas de Telecomunicaciones

PROYECTO FIN DE CARRERA

APLICACIÓN ANDROID PARA LA EVALUACIÓN DE SENSORES

OUIDAD BENCHELLAL
OCT 2014

Contenido

0 . AGRADECIMIENTOS

Este proyecto va dedicado a todas las personas que me han apoyado y han confiado en mí en todo momento, especialmente a mis padres que que han sufrido mucho conmigo, y me han brindado la oportunidad de venir a España y estudiar en la Universidad Carlos III, y se han preocupado durante muchos años por mi, mi rendimiento, y todo lo que tenga relación conmigo, a pesar de la distancia, siempre han estado presentes en mi vida día a día.

A todas las personas que he conocido y que me han acogido como una segunda familia.

A mis compañeros de la universidad, de la residencia, de la lucha del estudio y también y de los buenos y malos momentos.

También va dedicado a las personas que no han confiado en mi capacidad para superar mi fobia a la programación, ya que fue una asignatura que me dio mucha guerra a lo largo de la carrera, para decirles aquí estoy yo, y que cuando me propongo algo lo consigo.

A mis profesores que me han dado mucho y me han enseñado mucho hasta el día de la presentación de este proyecto, y espero no dejar de aprender nunca.

"Podrán morir las personas, pero jamás sus ideas"
Autor: Ernesto Che Guevara

1. PLANTEAMIENTO DEL TRABAJO:

En este trabajo vamos a tratar los diferentes sensores de Android, su alcance y sus limitaciones.

La idea de este trabajo surge a fin de acercar a la gente la utilidad de los múltiples sensores de Android, así como las particularidades de cada uno.

Se trata de estudiar los sensores de forma detallada, y ver el alcance que puede tener cada uno de ellos valorando los valores máximos y mínimos en los que se mueven, y las condiciones en los que se alcanzan, haciendo un estudio en profundidad y programando una aplicación, que nos permitirá, entender mejor su funcionamiento y el uso que se le puede dar en aplicaciones futuras.

1.1. PROPUESTA:

El propósito de este proyecto es proporcionar información a los diferentes usuarios de dispositivos móviles con tecnología Android, centrándose en unos componentes que vienen integrados en el sistema llamados sensores (dependiendo de la versión del móvil y la tecnología que tiene integrada puede tener unos sensores u otros).

Particularmente el trabajo persigue analizar el alcance de los distintos sensores de Android en especial, el acelerómetro, el sensor de temperatura, y el magnetómetro con el fin de saber cuál es el alcance de cada uno de ellos.

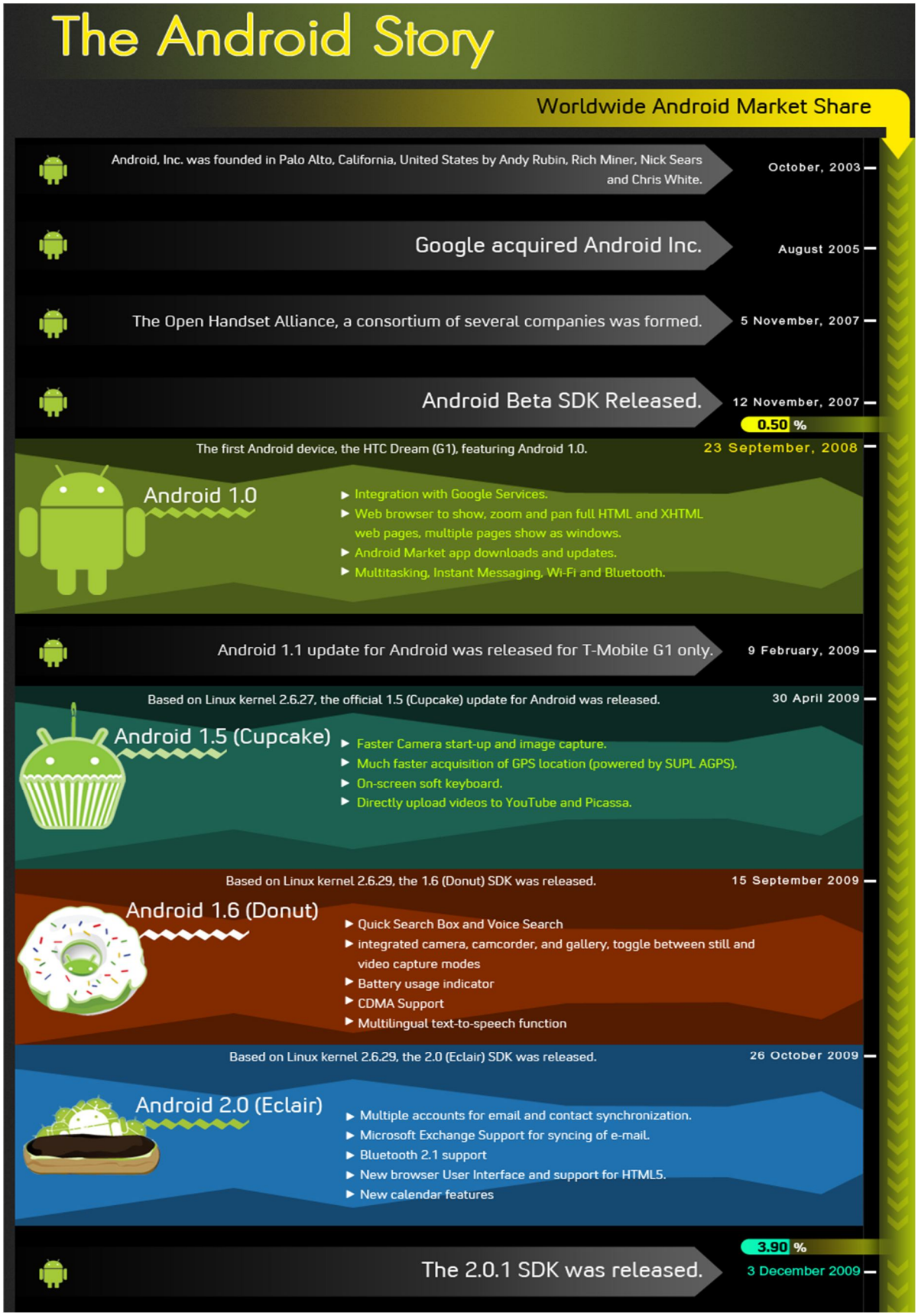
1.2. JUSTIFICACIÓN:

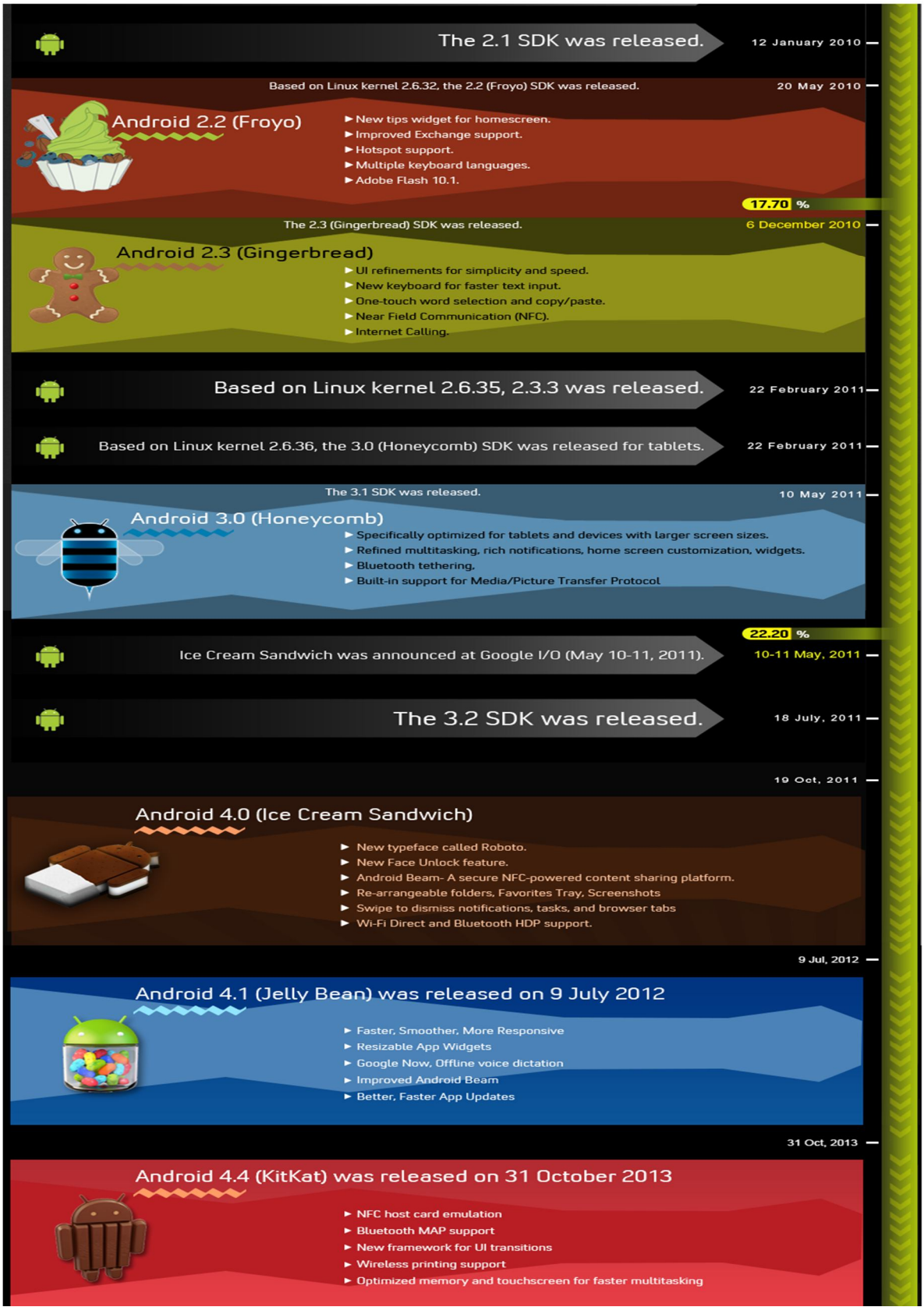
Hoy en día, según la unión internacional de telecomunicaciones, el mundo de la telefonía móvil está en auge “revolucionando el concepto de telefonía de diversas maneras” [*ref. 1*].

El desarrollo de las telecomunicaciones con el despliegue del LTE y el crecimiento de las empresas de desarrollo de sistemas operativos cada vez más potentes y estables como Symbian, Windows Mobile, IOS o Android.

Este último es el sistema operativo en el que vamos a centrar nuestro proyecto. Está basado en Linux y goza de una gran flexibilidad y seguridad. Sin olvidar que Google- la empresa fundadora y desarrolladora del sistema- ha liberado la mayoría del código Android con licencia libre [*ref. 2*].

Esto hace que cada vez tengamos una nueva versión de número mayor siendo esta la versión más moderna *[ref. 3]*, que proporciona mayor prestaciones con un mayor número de aplicaciones compatibles como se puede ver en la siguiente imagen recopilada de un estudio sobre la evolución de Android *[ref. 4]*.


The Android Story
Worldwide Android Market Share
Android, Inc. was founded in Palo Alto, California, United States by Andy Rubin, Rich Miner, Nick Sears and Chris White.
October, 2003
Google acquired Android Inc.
August 2005
The Open Handset Alliance, a consortium of several companies was formed.
5 November, 2007
Android Beta SDK Released.
12 November, 2007
0.50 %
The first Android device, the HTC Dream (G1), featuring Android 1.0.
23 September, 2008
Android 1.0
Integration with Google Services.
Web browser to show, zoom and pan full HTML and XHTML web pages, multiple pages show as windows.
Android Market app downloads and updates.
Multitasking, Instant Messaging, Wi-Fi and Bluetooth.
Android 1.1 update for Android was released for T-Mobile G1 only.
9 February, 2009
Based on Linux kernel 2.6.27, the official 1.5 (Cupcake) update for Android was released.
30 April 2009
Android 1.5 (Cupcake)
Faster Camera start-up and image capture.
Much faster acquisition of GPS location (powered by SUPL AGPS).
On-screen soft keyboard.
Directly upload videos to YouTube and Picassa.
Based on Linux kernel 2.6.29, the 1.6 (Donut) SDK was released.
15 September 2009
Android 1.6 (Donut)
Quick Search Box and Voice Search
integrated camera, camcorder, and gallery, toggle between still and video capture modes
Battery usage indicator
CDMA Support
Multilingual text-to-speech function
Based on Linux kernel 2.6.29, the 2.0 (Eclair) SDK was released.
26 October 2009
Android 2.0 (Eclair)
Multiple accounts for email and contact synchronization.
Microsoft Exchange Support for syncing of e-mail.
Bluetooth 2.1 support
New browser User Interface and support for HTML5.
New calendar features
3.90 %
The 2.0.1 SDK was released.
3 December 2009


The 2.1 SDK was released.
12 January 2010
Based on Linux kernel 2.6.32, the 2.2 (Froyo) SDK was released.
20 May 2010
Android 2.2 (Froyo)
New tips widget for homescreen.
Improved Exchange support.
Hotspot support.
Multiple keyboard languages.
Adobe Flash 10.1.
17.70 %
The 2.3 (Gingerbread) SDK was released.
6 December 2010
Android 2.3 (Gingerbread)
UI refinements for simplicity and speed.
New keyboard for faster text input.
One-touch word selection and copy/paste.
Near Field Communication (NFC).
Internet Calling.
Based on Linux kernel 2.6.35, 2.3.3 was released.
22 February 2011
Based on Linux kernel 2.6.36, the 3.0 (Honeycomb) SDK was released for tablets.
22 February 2011
The 3.1 SDK was released.
10 May 2011
Android 3.0 (Honeycomb)
Specifically optimized for tablets and devices with larger screen sizes.
Refined multitasking, rich notifications, home screen customization, widgets.
Bluetooth tethering.
Built-in support for Media/Picture Transfer Protocol
22.20 %
Ice Cream Sandwich was announced at Google I/O (May 10-11, 2011).
10-11 May, 2011
The 3.2 SDK was released.
18 July, 2011
19 Oct, 2011
Android 4.0 (Ice Cream Sandwich)
New typeface called Roboto.
New Face Unlock feature.
Android Beam- A secure NFC-powered content sharing platform.
Re-arrangeable folders, Favorites Tray, Screenshots
Swipe to dismiss notifications, tasks, and browser tabs
Wi-Fi Direct and Bluetooth HDP support.
9 Jul, 2012
Android 4.1 (Jelly Bean) was released on 9 July 2012
Faster, Smoother, More Responsive
Resizable App Widgets
Google Now, Offline voice dictation
Improved Android Beam
Better, Faster App Updates
31 Oct, 2013
Android 4.4 (KitKat) was released on 31 October 2013
NFC host card emulation
Bluetooth MAP support
New framework for UI transitions
Wireless printing support
Optimized memory and touchscreen for faster multitasking

Por todo esto resulta importante conocer las posibilidades que nos proporciona nuestro dispositivo Android, primero conociendo que sensores posee y para después conocer las limitaciones de cada uno de ellos, las ventajas que proporcionan, su uso, y comprender que no todos los dispositivos llevan integrados todos los sensores ni los mismos, depende del fabricante, que el terminal posea unos sensores u otros, más adelante explicaremos los diferentes sensores ,como saber que sensores posee mi dispositivo ,que usos tiene cada uno y en que ámbitos se usa cada uno.

1.3. OBJETIVOS DEL PROYECTO:

Con este trabajo se pretende profundizar en algunas de las características de la plataforma Android.

Concretamente explorar las posibilidades de los sensores que vienen incorporados en el terminal móvil.

Para ello, primero diseñaremos una aplicación sencilla que detecte los sensores que incluye mi dispositivo y posteriormente se pretende hacer un diseño más complejo en el que tendremos que usar un tipo de base de datos para el correcto funcionamiento de mi aplicación, incluyendo un graficado de estos valores.

De esta forma se pretende acercar a un usuario no necesariamente con conocimientos de Android todo lo que le puede ofrecer su móvil o tableta.

Hoy en día las aplicaciones móviles están al alcance de la mayoría, poder sacar el mayor provecho de nuestro dispositivo es uno de los objetivos de este trabajo.

2. INTRODUCCION:

Este bloque será una presentación resumida de Android, de sus utilidades, y su uso así como de las funciones de unos componentes especiales integrados en los dispositivos móviles llamados sensores.
Lo primero que hace falta saber es que Android está basado en software libre, es decir, que todo el mundo puede ver el código de las aplicaciones realizadas para dicha plataforma. Por el mismo motivo, todos los usuarios que quieran pueden desarrollar sus propias aplicaciones, ya que Google ha puesto a disposición de todos los públicos las herramientas necesarias para programar aplicaciones en Android.
El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y está teniendo una importante aceptación tanto por los usuarios como por la industria. En la actualidad se está convirtiendo en una seria alternativa frente a otras plataformas como Symbian, iPhone o Windows Phone.

2.1. ANDROID:

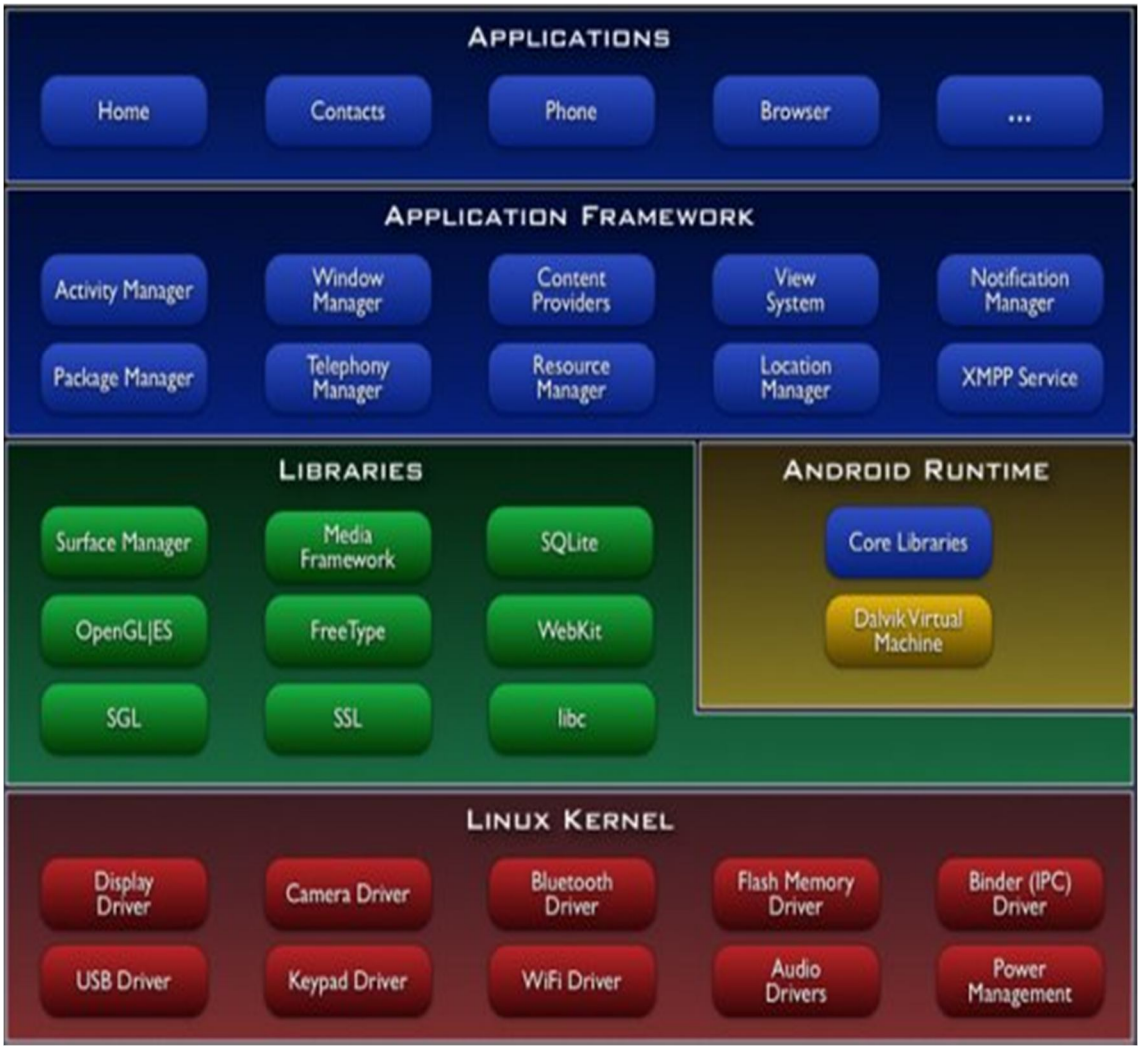
2.1.1. PRINCIPALES CARACTERÍSTICAS DE ANDROID:

Android posee todo un framework -una estructura de trabajo mediante el cual Android puede ser organizado y desarrollado- de aplicaciones que permite la reutilizar y sustituir componentes, típico en los lenguajes de programación Orientados a Objeto como Java, y un lenguaje interpretado para ayudar al desarrollo del proyecto [ref. 5]:

1. Android contiene la máquina virtual Dalvik, un software que emula a un ordenador y puede ejecutar programas como si fuera un ordenador real.
2. Además posee un navegador integrado de código abierto WebKit, que facilita a los desarrolladores incluir gran parte de las funcionalidades de Safari, en sus propias aplicaciones.
3. Gráficos optimizados, con una librería de gráficos 2D y 3D basados en la especificación estándar OpenGL (Librería de Gráficos Abierta).
4. Sistema Gestor de Base de Datos SQLite, para almacenamiento de datos estructurados.
5. Soporte para medios con formatos comunes de audio, vídeo e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
6. Telefonía, común para los dispositivos móviles que usan tecnología digital para conectarse a través de su teléfono con su ordenador y/o tableta e interactuar con él (mediante Bluetooth, EDGE, 3G 4G y WiFi).
7. Cámara, GPS (Sistema de Posicionamiento Global), brújula, acelerómetro y otros sensores que detallare más adelante.
8. Además, posee un ambiente rico de desarrollo incluyendo un emulador de dispositivo, herramientas para depurar, perfiles de memoria, y un Plugin para el Entorno de Desarrollo Integrado (IDE) de Eclipse.

2.1.2. ARQUITECTURA DE UNA APLICACIÓN ANDROID:

A continuación vamos a ver los componentes más significativos del sistema operativo de Android como vemos en la siguiente figura *[ref. 6]*:



Cada sección se describe basándose en el libro “El Gran Libro de Android “en el capítulo 1.4. Arquitectura de Android. A continuación presentamos un resumen completando esta información con las fuentes indicadas en la referencia [ref. 7 y 8]:

Aplicaciones:

9. Es la sección más externa al usuario final (Las aplicaciones están desarrolladas en lenguaje Java que es la base de la programación en Android).
   Contiene las aplicaciones principales para los clientes, tanto las que incluye Android por defecto como aquellas añadidas por el usuario, como correo electrónico, servicio de SMS, calendario, mapas, navegador, contactos, etc.

Aplicaciones del framework:

10. Los desarrolladores de Android tienen pleno acceso a la misma API del framework que se usa para las aplicaciones. Esta arquitectura simplifica la re utilización de componentes, ya que cualquier aplicación que se publique puede aprovechar las capacidades de otras aplicaciones publicadas hasta el momento, siempre adaptándose a los valores de seguridad del framework.

Este mismo mecanismo permite al usuario sustituir componentes que hayan podido quedar obsoletos o

en desuso, de forma fácil y sin tener que modificar código ajeno, cumpliendo así la premisa de 'abierto para la expansión y cerrado para la modificación'.

Esta sección contiene:

- Un conjunto de servicios y sistemas, como un set de vistas para construir aplicaciones con sus listas, botones, cajas de texto, navegador, etc.

- Proveedores de contenido que permiten acceder a los datos de otras aplicaciones, contactos, o incluso compartir sus propios datos.

- Gestión de recursos que facilita el acceso a recursos como cadenas localizadoras, gráficos y archivos de diseño.
  Un administrador de notificaciones que permite mostrar las alertas en la barra de estado.

- Gestión de actividades que controla el ciclo de vida de las aplicaciones.

Librerías:
Android incluye una serie de librerías C/C++ que son usadas por distintos componentes del sistema operativo. Estas librerías están expuestas a los desarrolladores a través del framework, algunas de ellas son:

- Librería libc: Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

- Librería Surface Manager: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- OpenGL/SL y SGL: Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3 y 2D.

- Librería Media Libraries: Proporciona todos los códec necesarios para el soporte multimedia.

- FreeType: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- Librería SSL: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

- Librería SQLite: Creación y gestión de bases de datos relacionales.

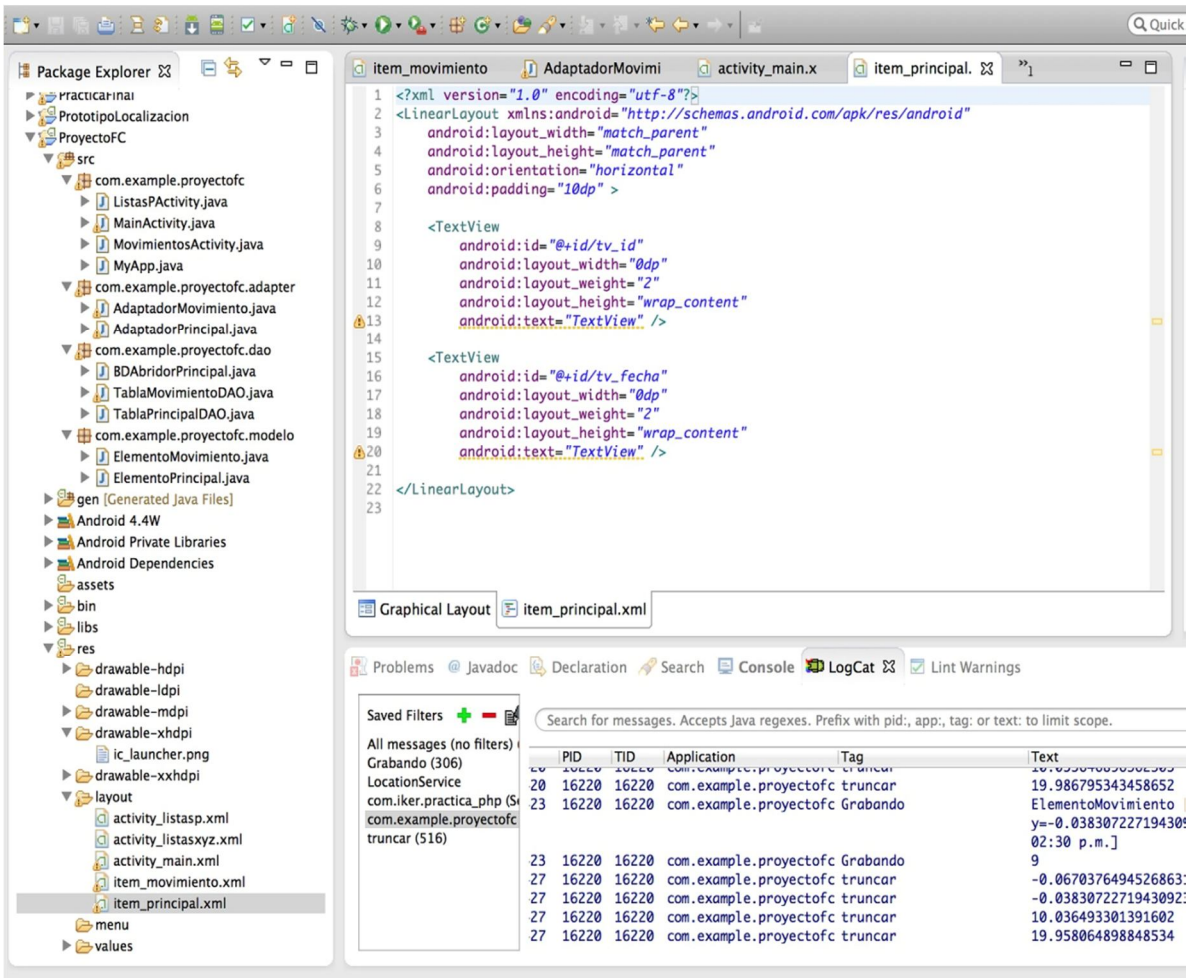
- Librería WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Android Runtime: Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución, incluye el conjunto de librerías que proporciona la mayor parte de la funcionalidad disponible en el núcleo de las librerías de Java, y la máquina virtual Dalvik.

Kernel de Linux: Sección basada en el Kernel o núcleo de la versión 2.6 de Linux, contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este Kernel de Linux embebido en el propio Android. Éste actúa como una capa de abstracción entre el hardware y el software.

2.1.3. ESTRUCTURA BÁSICA DE UNA APLICACIÓN ANDROID:

Cada vez que creamos un nuevo proyecto de una aplicación Android en Eclipse, se genera automáticamente un sistema de archivos definido para todas las aplicaciones de dicho sistema operativo, aquí se muestra el sistema de clases creado en Eclipse.



Según la versión del SDK, se van haciendo pequeñas modificaciones en la estructura de las aplicaciones en Android. La jerarquía de carpetas y archivos define todos los proyectos en Android construidos a partir de la versión más reciente.

Dentro de la carpeta principal, que lleva el mismo nombre del proyecto en cuestión, se encuentra la estructura de archivos que es la misma siempre *[ref. 9].*

Vamos a definir a continuación las carpetas más importantes, su función y su significado después de hacer un estudio y comparativa a dos artículos y proceder a integrar información de ambos *[ref. 10 y*

11]:

- La carpeta *src: Es* donde se guarda el paquete que contiene las clases necesarias para hacer funcionar la aplicación. Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity) principal de la aplicación, siempre bajo la estructura del paquete java definido.

- La carpeta *bin: Es* donde se guardan las clases compiladas que hemos descrito en el punto anterior, y se genera el .apk, fichero comprimido que contiene la aplicación final lista para instalar.

- La carpeta *gen:* Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en javas dirigidos al control de los recursos de la aplicación. En ella se ubica una clase muy peculiar de las aplicaciones en Android, la clase R.java.que veremos en la imagen a continuación. Esta clase se genera automáticamente cada vez que modificamos los ficheros de recursos y no podemos retocarla manualmente:

Se podría considerar como un nexo de unión entre los recursos de la aplicación y los ficheros *.XML* que se encuentran en el proyecto.

- La carpeta *assets*: Inicialmente vacía y que en general no suele usarse. Contiene los ficheros auxiliares necesarios para la aplicación, y recursos no compilados del proyecto ,a los que no se les asigna un identificador dentro de la aplicación, como por ejemplo ficheros de configuración de datos, etc.

  Se pueden ubicar cualquier tipo de ficheros que represente un origen de datos para la aplicación. Para poder manejar esos datos, en la aplicación se crearía un gestor de assets, o *AssetManager*, que se encargaría de asociar los datos dentro de la propia aplicación.

- La carpeta *res: Se* genera automáticamente para gestionar los recursos de origen de la aplicación. Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.

  Los diferentes tipos de recursos se deberán distribuir entre las siguientes carpetas:

  - /res/drawable/. Contienen las imágenes de la aplicación.
  - /res/layout/. Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica.
  - /res/anim/. Contiene la definición de las animaciones utilizadas por la aplicación.
  - /res/menu/. Contiene la definición de los menús de la aplicación.
  - /res/values/. Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), etc.
  - /res/xml/. Contiene los ficheros XML utilizados por la aplicación.
  - /res/raw/. Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

  El fichero *AndroidManifest.xml* que es indispensable para todas las aplicaciones de Android. Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono,…), sus componentes (pantallas, mensajes,…), o los permisos necesarios para su ejecución, cualquier clase creada debe estar declarada en el manifest antes de ejecutar el programa, en caso contrario la aplicación da error.

2.1.4. COMPONENTES DE UNA APLICACIÓN ANDROID:

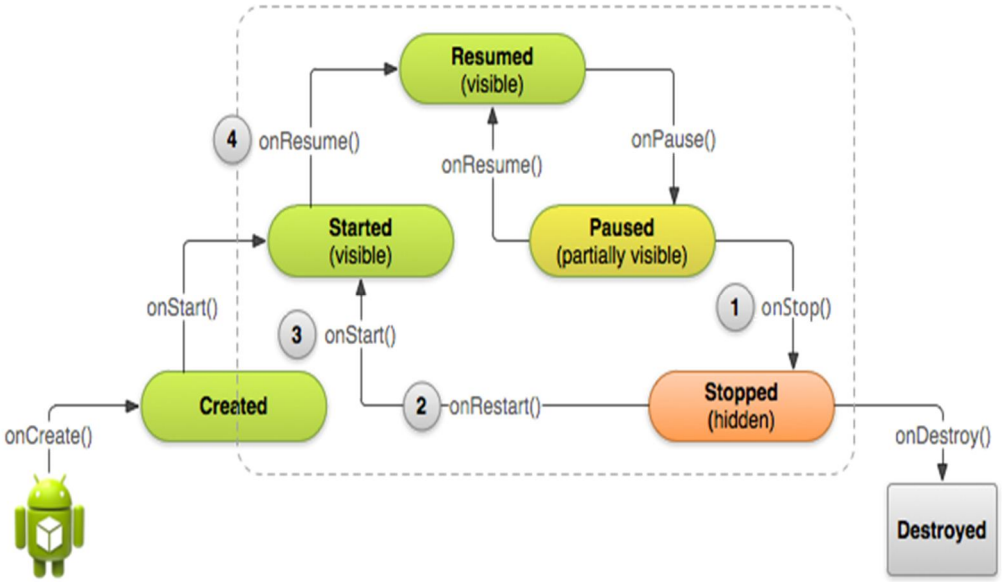
Toda aplicación puede constar de varios componentes que interactúan entre sí, y cada uno de ellos tiene una función determinada a realizar. Los componentes se implementan en las clases que tengamos creadas en la carpeta *src* de nuestro proyecto. Las más importantes de toda aplicación las pasamos a definir a continuación *[ref. 12]*:

Activity: Uno de las componentes principales de una aplicación Android .Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual *[ref. 13]*.

En general, corresponde a una pantalla específica de la aplicación, como por ejemplo la pantalla inicial que representa el punto de entrada de dicha aplicación. Desde ahí se invocan a las principales tareas que el usuario puede llevar a cabo.

Una aplicación tendrá tantas actividades como ventanas distintas tenga.

Las actividades tienen un ciclo de vida diferente al que puede tener una actividad en otros sistemas operativos, es decir, que pasan por una serie de estados, desde que se crean hasta que se destruyen, que conviene conocer para poderlos capturar y actuar sobre ellos en determinadas circunstancias como podemos ver en la siguiente imagen *[ref. 14]:*



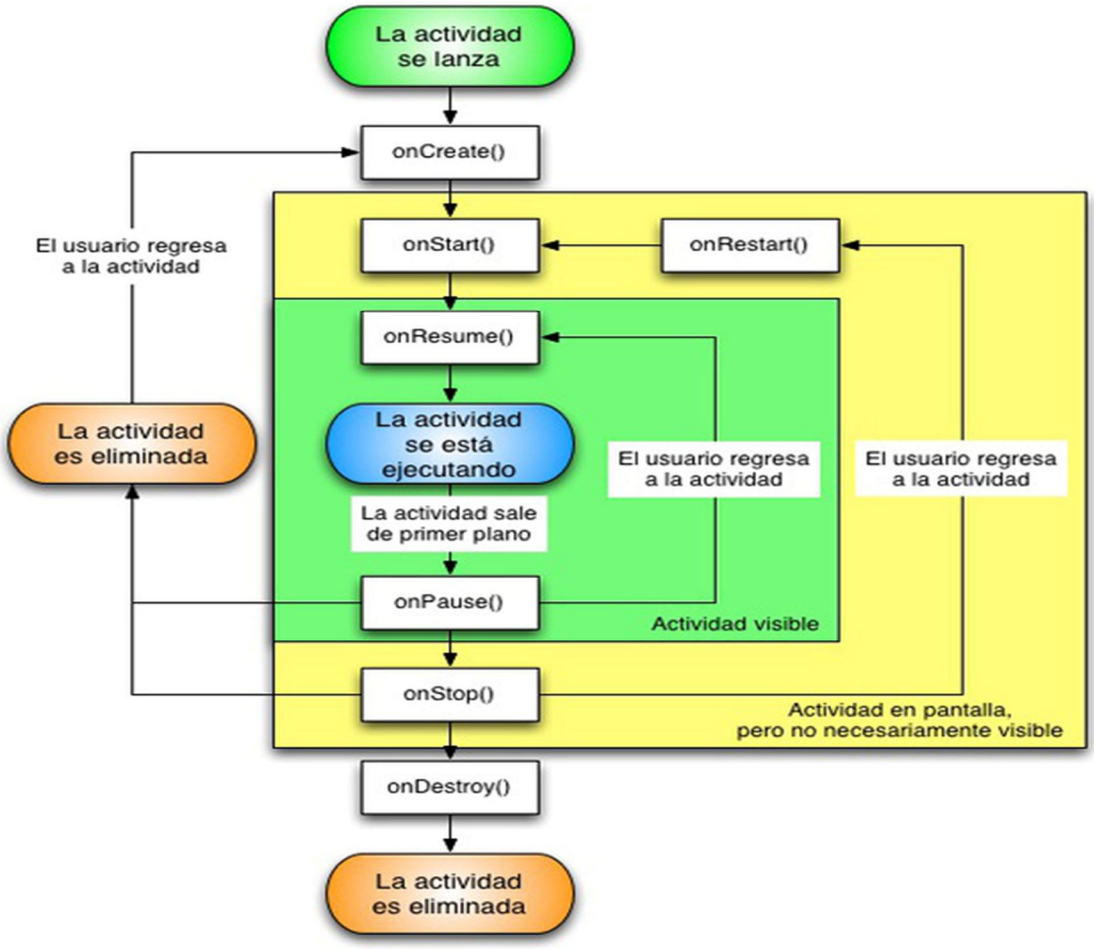
"Una actividad tiene esencialmente 4 estados" *[ref. 15]*:

- Activa (running): Ocurre cuando la actividad está en ejecución, es decir, es la tarea principal [*ref. 16*].

- Visible (paused): Una actividad pausada está completamente viva (que mantiene toda la información del estado y permanece unida al gestor de ventanas) .Se debe guardar la información en este estado para prevenir una posible pérdida de datos en caso de que el sistema decida prescindir de ella para liberar memoria [*ref. 16*].

- Parada (Stopped): La actividad se detiene. Conserva toda la información del estado, pero ya no es visible para el usuario, por lo que su ventana se oculta y a menudo esta actividad será eliminada por el sistema la memoria sea necesaria en cualquier otro sitio [*ref. 16*].

- Destruida (Destroyed): Cuando la actividad termina, al invocarse el método finish (), o es destruida por el sistema Android, sale de la pila de actividades [*ref. 16*].



Cada vez que una actividad cambia de estado se van a producir eventos que podrán ser capturados por ciertos métodos de la actividad que se describen a continuación en esta imagen [*ref. 17*]:

- OnCreate (Bundle savedInstanceState): Es el método que crea la actividad. Recibe un parámetro de tipo Bundle, que contiene el estado anterior de la actividad, para preservar la información que hubiera, en caso de que hubiera sido suspendida, aunque también puede iniciarse con un null si la información anterior no es necesaria o no existe [*ref. 17*].

- OnRestart (): Reinicia una actividad tras haber sido parada (si continúa en la pila de tareas). Se inicia

desde cero [*ref. 17*].

- Onstart (): Inmediatamente después de OnCreate (Bundle savedInstanceState), o de OnRestart () según corresponda. Muestra al usuario la actividad. Si ésta va a estar en un primer plano, el siguiente método debe ser onResume (). Si por el contrario se desarrolla por debajo, el método siguiente será onStop (). Es recomendable llamar al método onRestoreInstanceState () para asegurar la información [*ref. 17*].

- OnResume (): establece el inicio de la interacción entre el usuario y la aplicación. Solo se ejecuta cuando la actividad está en primer plano. Si necesita información previa, el método onRestoreInstanceState () aportará la situación en que estaba la actividad al llamar a onResume (). También puede guardar el estado con onSaveInstanceState () [*ref. 17*].

- OnPause (): se ejecuta cuando una actividad va a dejar de estar en primer plano, para dar paso a otra. Guarda la información, para poder restaurar cuando vuelva a estar activa en el método onSaveInstanceState (). Si la actividad vuelve a primer plano, el siguiente método será onResume (). En caso contrario, será onStop () [*ref. 17*].

- OnStop (): la actividad pasa a un segundo plano por un largo período. Como ya se ha dicho, el sistema puede liberar el espacio que ocupa, en caso de necesidad, o si la actividad lleva parada mucho tiempo [*ref. 17*].

- OnDestroy (): es el método final de la vida de una actividad. Se llama cuando ésta ya no es necesaria, o cuando se ha llamado al método finish () [*ref. 17*].

Además de estos métodos, cabe destacar dos más, que son de vital importancia:

- OnSaveInstanceState (): guarda el estado de una actividad. Es muy útil cuando se va a pausar una actividad para abrir otra [*ref. 17*].

- OnRestoreInstanceState (): restaura los datos guardados en OnSavedInstanceState () al reiniciar una actividad [*ref. 17*].

La vida entera de una actividad ocurre entre la primera llamada a OnCreate () y la única llamada final a OnDestroy (). Una actividad será lanzada por primera vez y, además, establecerá toda la configuración general de su estado en OnCreate (), y liberará todos los recursos que quedan en OnDestroy (). El curso de la vida visible de una actividad ocurre entre una llamada al método onStart () y la correspondiente llamada a onStop (). El OnStart () y onStop () se pueden llamar varias veces, así como la actividad se hace visible y se oculta para el usuario [*ref. 17*].
El tiempo de vida en primer plano de una actividad ocurre entre una llamada a onResume () y la correspondiente llamada a onPause () [*ref. 17*].

2.2. SENSORES EN DISPOSITIVOS MOVILES:

“Se trata de un conjunto de dispositivos con los que podremos obtener información del mundo exterior (en este conjunto no se incluye la cámara, el micrófono o el GPS).

Todos los sensores se manipulan de forma homogénea" [*ref. 18*]. Son los dispositivos de entrada más novedosos que incorpora Android y con ellos podremos implementar formas atractivas de interacción con el usuario.

Los sensores utilizados por una aplicación deberán de estar declarados en el AndroidManifest.xml haciendo uso de la etiqueta <uses-feature> para que la aplicación pueda hacer uso de ellos. Está información es utilizada por el Android Market para saber si una aplicación es compatible con un dispositivo concreto [*ref. 18*].

Existen varios tipos de sensores según vemos en la siguiente tabla [*ref. 18*], aunque los sensores disponibles varían en función del dispositivo utilizado.

Tipo	Utilidad	dim.	desde API
Acelerómetro TYPE_ACCELEROMETER	medir aceleraciones por gravedad y cambios de movimiento	3	3
Campo magnético TYPE_MAGNETIC_FIELD	brújula, detectar campos magnéticos	3	3
Giroscopio TYPE_GYROSCOPE	detectar giros	3	3
Orientación TYPE_ORIENTATION	indicar dirección a la que apunta el dispositivo	3	3
Luz ambiental TYPE_LIGHT	ajustar iluminación pantalla	1	3
Proximidad TYPE_PROXIMITY	si hay un objeto a menos de 5 cm (al hablar por teléfono)	booleano	3
Presión atmosférica TYPE_PRESSURE	altímetro, barómetro	1	3
Temperatura interna TYPE_TEMPERATURE	evitar sobrecalentamientos (obsoleto desde API14)	1	3
Gravedad TYPE_GRAVITY	medir la aceleración debida a la gravedad	3	9
Acelerómetro lineal TYPE_LINEAR_ACCELERATION	medir aceleraciones descontando la gravedad	3	9
Vector de rotación TYPE_ROTATION_VECTOR	detectar giros	3	9
Temperatura ambiental TYPE_AMBIENT_TEMPERATURE	medir la temperatura del aire	1	14
Humedad relativa TYPE_RELATIVE_HUMIDITY	Medir el punto de rocío, humedad absoluta y relativa.	1	

2.2.1. MANEJO DE SENSORES:

Para el manejo de los diferentes sensores disponibles en Android es necesario hacer uso de las siguientes clases e interfaces que están en el paquete Android. Hardware.

Para este análisis de sensores y su clasificación me he basado en la web de Android dedicada a los desarrolladores [*ref. 19*] en el libro de programación avanzada [*ref. 20*]:

- La clase Sensor: Representa el sensor de turno que queramos utilizar. Esta clase proporciona varios métodos que permiten determinar las capacidades de un sensor.

- La clase *SensorManager* : Nos permite acceder a los sensores del dispositivo y a las interfaces .Proporciona varias constantes de sensores que se utilizan para informar de la precisión del sensor, conjunto de tasas de adquisición de datos, y calibrar sensores.

- La clase *SensorEventListener*: Interfaz que registra los cambios hechos en el sensor indicado.

- La clase SensorEvent: El sistema utiliza esta clase para crear un objeto de un evento del sensor, lo que proporciona información acerca de él. Un objeto de evento sensor incluye la siguiente información: los datos del sensor, el tipo de sensor que generó el evento, la exactitud de los datos, y la marca de tiempo para el evento.

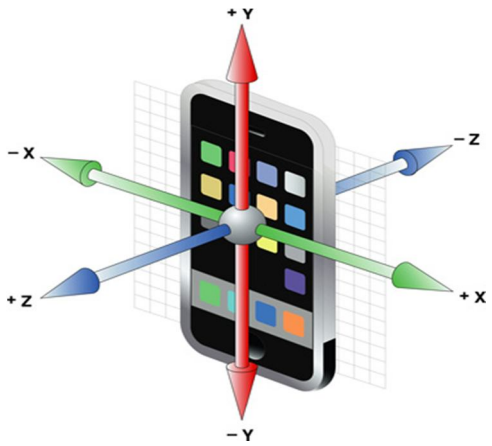
2.2.2. CLASIFICACIÓN DE SENSORES:

Los sensores de Android se clasifican en 3 categorías:

2.2.2.1. SENSORES DE MOVIMIENTO:

Estos sensores miden las fuerzas de aceleración y fuerzas de rotación a lo largo de tres ejes. Esta categoría incluye los acelerómetros, sensores de gravedad, giroscopios y sensores de rotación del vector.

1. Acelerómetro: Es un sensor que mide la aceleración aplicada al dispositivo con respecto a la caída libre. Las medidas son realizadas a lo largo de los tres ejes cartesianos, como podemos ver en la imagen siguiente, por lo que tenemos tres componentes, devueltas en un vector, que corresponden con la fuerza aplicada por el dispositivo sobre los ejes (x, y, z) [*ref. 21*].



Conceptualmente, un sensor de aceleración determina la aceleración que se aplica a un dispositivo (A) mediante la medición de las fuerzas que se aplican sobre el propio sensor (F) usando la siguiente relación:

$A = - \Sigma F / masa.$

Sin embargo, la fuerza de la gravedad siempre está influyendo en la aceleración medida de acuerdo con la siguiente relación:

A =-g - ΣF / masa.

Como sabemos que el módulo de la aceleración de la gravedad es, aproximadamente, 9.8 m/s^2, el módulo percibido por el terminal en reposo debe ser el mismo. Por tanto, valores suficientemente mayores o menores indican que el portador del teléfono comienza a caminar o se detiene. Como sabemos la dirección y el sentido del mayor cambio, podemos estimar la dirección hacia la que se mueve.

Por otro lado, como sabemos la magnitud de la gravedad en esas tres componentes, podemos calcular la inclinación del dispositivo en reposo. Por ejemplo, situando el terminal en posición horizontal, con la parte positiva del eje Z apuntando hacia el techo, tenemos que sólo ese eje se ve afectado por la gravedad. Si comenzamos a inclinarlo alrededor de cualquiera de los otros ejes, veremos que la magnitud del eje Z decae, incrementándose en los otros ejes. Haciendo una medida de proporcionalidad entre estos valores, podemos saber el grado de inclinación del dispositivo.

En general, el acelerómetro es un buen sensor a usar para monitorizar el movimiento del dispositivo. Casi todos los teléfonos Android y tabletas tienen un acelerómetro, que utiliza mucho menos energía en comparación con los otros sensores de movimiento.

Una desventaja es que puede que tenga que aplicar filtros de paso bajo y paso alto para eliminar las fuerzas gravitatorias y reducir el ruido.

2. Acelerómetro lineal: Vector tridimensional que representa la aceleración a lo largo de cada eje del dispositivo, sin tener en cuenta el efecto de la gravedad. Proporciona los datos de aceleración de acuerdo con la siguiente relación:

Aceleración lineal = aceleración - Aceleración de la gravedad

El sistema de coordenadas es el mismo que el utilizado por el acelerómetro, así como las unidades de medida (m/s^2).

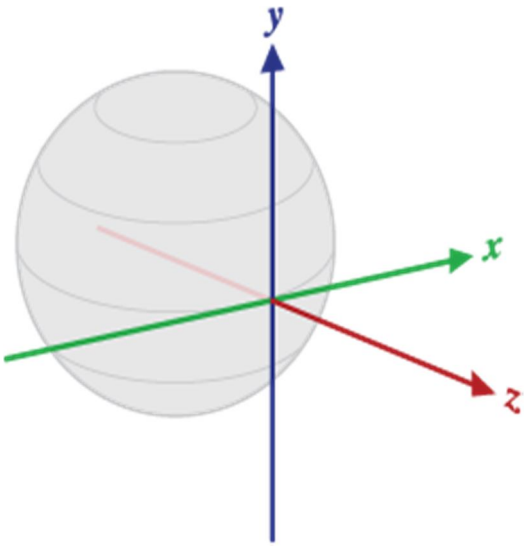
3. Sensores de gravedad: El sensor de la gravedad proporciona un vector tridimensional que indica la dirección y la magnitud de la gravedad.

Cuando un dispositivo está en reposo, la salida del sensor de la gravedad debe ser idéntica a la del acelerómetro (mismo sistema de coordenadas y unidad de medida).

4. Giroscopio: Un giroscopio permite que un teléfono inteligente mida y mantenga la orientación. Pueden monitorizar y controlar posiciones del dispositivo como la orientación, la dirección, el movimiento angular y la rotación. Cuando se aplica a un teléfono inteligente, un sensor giroscópico comúnmente lleva a cabo funciones de reconocimiento de gestos. Además, los giroscopios en los teléfonos inteligentes ayudan a determinar la posición y orientación del teléfono. La tasa o la rotación se mide en (rad /s) alrededor de un dispositivo y tiene el mismo sistema de coordenadas que el acelerómetro.

La rotación es positiva en el sentido contrario a las agujas del reloj, es decir, un observador que mira desde algún lugar positivo en la x, y o eje z en un dispositivo colocado en el origen informará rotación positiva si el dispositivo parecía estar girando en sentido anti horario.

5. Sensor de rotación vectorial: Representa la orientación del dispositivo como una combinación de un ángulo y un eje, en el que el dispositivo ha girado a través de un ángulo θ alrededor de un eje (x, y, o z), como se puede ver en la imagen [*ref. 22*]:



Los tres elementos del vector de giro se expresan como sigue:

X * sin (θ / 2)
Y * sin (θ / 2)
Z * sin (θ / 2)

Los tres elementos del vector de giro son (cos (θ / 2), x * sen (θ / 2), y * sen (θ / 2), z * sen (θ / 2)), y no tienes unidades. Los ejes x, y, z están definidos en la misma forma que el sensor de aceleración. El sistema de coordenadas de referencia se define como una base orto normal directo.

2.2.2.2. Sensores ambientales:

Estos sensores miden diversos parámetros ambientales, como la temperatura del aire ambiente y la presión, la iluminación, y la humedad. Esta categoría incluye barómetros, fotómetros y termómetros.

Los sensores ambientales devuelven un valor único para cada evento, y no suelen requerir ningún filtrado ni procesado de datos.

1. Sensor de luz ambiental: Su función es detectar la intensidad de la luz del ambiente para ajustar el brillo de la pantalla. Así, cuando estemos en un cuarto oscuro, este sensor ajustará la intensidad de la luz de la pantalla para que la lectura sea clara, pero no molesta. En cambio, en condiciones de luz ambiental muy fuerte, el sensor detectará esto para ajustar los niveles y ahorrar batería. En la siguiente imagen se ilustra donde se encuentra este sensor en nuestro dispositivo [ref. 23].

2. Sensor de humedad: Este sensor mide la humedad relativa y el punto del rocío usando la sonda de humedad, de la misma manera que se hace para medir los sensores de luz, presión y temperatura. Sin embargo, si un dispositivo tiene un sensor de humedad TYPE_ RELATIVE_HUMIDITY y un sensor de temperatura TYPE_AMBIENT_TEMPERATURE puede usar estos dos flujos de datos para el cálculo del punto de rocío y la humedad absoluta.

Punto de rocío: El punto de rocío es la temperatura a la que debe enfriarse un volumen dado de aire, a una presión barométrica constante, para el vapor de agua se condense en agua. La ecuación siguiente muestra cómo puede calcular el punto de rocío:

$$Td(t, HR) = Tn \cdot \frac{Ln(RH/100\%) + m \cdot t / (Tn + t)}{m - [\ln(RH/100\%) + m \cdot t / (Tn + t)]}$$

Dónde:

Td= temperatura del punto de rocío en grados C
t = temperatura real en grados C
HR = humedad relativa real en tanto por ciento (%)
m = 17,62
Tn= 243,12

Humedad absoluta: La humedad absoluta es la masa de vapor de agua en un volumen dado de aire seco. La humedad absoluta se mide en gramos/metro3.La ecuación siguiente muestra cómo calcular la humedad absoluta:

$$Dv(t, HR) = 216.7 \cdot \frac{(RH/100\%) \cdot A \cdot \exp(m \cdot t / (Tn + t)}{273.15 + t}$$

Dónde:

Dv= humedad absoluta en gramos/metro3
t = temperatura real en grados C
HR = humedad relativa real en tanto por ciento (%)
m = 17,62
Tn = 243,12 grados C
A = 6.112 ha

3. Sensor de presión: Llamado también barómetro, nos permite medir la presión atmosférica, (la presión que ejerce el aire que nos rodea). La presión atmosférica en un lugar determinado experimenta variaciones asociadas con los cambios meteorológicos y, en un lugar determinado disminuye a medida que se incrementa la altitud.

4. Sensor de temperatura: Los datos en bruto que se adquieren de los sensores de temperatura por lo general no requiere de calibración, filtrado o modificación, lo que los hace parte de los sensores más fáciles de usar.
Hay dos tipos: Los que nos permiten conocer la temperatura ambiente que tenemos a nuestro alrededor, y los que nos dan la temperatura del dispositivo.

2.2.2.3. SENSORES DE POSICIÓN:

Estos sensores miden la posición física de un dispositivo. Esta categoría incluye los sensores de orientación y magnetómetros (sensor campo magnético y sensor de proximidad).

Los sensores de posición son útiles para determinar la posición física de un dispositivo móvil en relación con el mundo o la tierra. Por ejemplo, el sensor de campo magnético terrestre en combinación con el acelerómetro se puede utilizar para determinar la posición relativa de un dispositivo con respecto al Polo Norte magnético. También se puede utilizar el sensor de orientación (o métodos de orientación basados en sensores similares) para determinar la orientación espacial de un dispositivo (si está en horizontal o vertical) y lanzar mensajes al usuario sobre esto.

1. Sensor de campo magnético terrestre: Está basado en hardware. La mayoría de los fabricantes de teléfonos y tabletas incluyen un sensor de campo magnético terrestre. (Se usa en aplicaciones para detectar metales...etc.).

2. Sensor orientación: Está basado en software y obtiene sus datos del acelerómetro y
   El sensor de campo magnético terrestre. Es un componente no visible, que informa de los siguientes tres valores, en grados suponiendo que el dispositivo móvil está quieto:

   Giro: 0 grados significa que el dispositivo está nivelado, 90 grados, que el dispositivo se inclina sobre su lado izquierdo, y 270 grados, que el dispositivo se inclina sobre su lado derecho.

   Inclinación: 0 grados, significa que el dispositivo está nivelado, 90 grados, que el dispositivo se inclina sobre su parte superior (la del auricular) ,180 grados, cuando se da la media vuelta. Y cuando el dispositivo se inclina sobre su parte inferior (la del micrófono), el ángulo es de 270 grados.

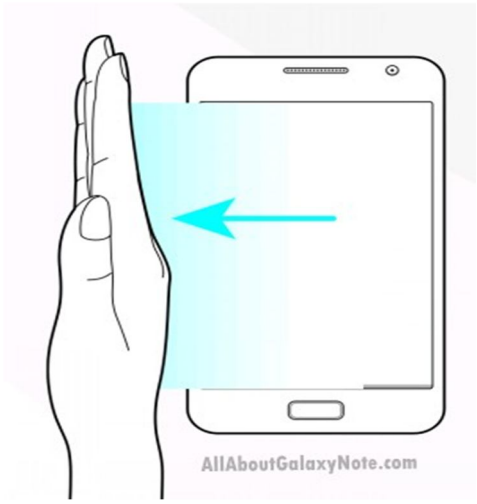
   Azimut:0 grados, significa que la parte superior del dispositivo (la del auricular) está apuntando hacia el norte, 90 grados que se está apuntando al este, 180 grados cuando se está apuntando al sur, 270 grados cuando se apunta al oeste, etc.

3. Sensor proximidad: Sensor proporcionado por Android basado en hardware, es un transductor que permite determinar qué tan cerca del elemento sensor se encuentra una señal o un objeto.

   Existen varios tipos de sensores de proximidad según el principio físico que utilizan. Los más comunes son los interruptores de posición, los detectores capacitivos, los inductivos y los fotoeléctricos (infrarrojos).

   El sensor de proximidad comúnmente más usado es de tipo inductivo, que genera un campo electromagnético, para detectar los objetos de metal que pasan cerca.

Los sensores de proximidad capacitivos trabajan generando un campo electrostático y detectan cambios en dicho campo a causa de un objeto que se aproxima a la superficie de detección, en la siguiente imagen vemos cómo actúa este sensor [ref. 24].



El sensor de proximidad no solo sirve para apagar o encender la pantalla cuando estamos llamando, sino que podemos darle otros muchos usos, como desplazarnos en una web sin necesidad de tocar la pantalla.

3. DESARROLLO DE LA APLICACION:

3.1 VISION GENERICA:

Los Smartphone no solo llevan cada vez procesadores más potentes y con mayor memoria RAM, también se van añadiendo nuevos y mejores sensores [ref. 25]. Por lo general tan solo lo notamos si nos fijamos en la descripción de la caja, o en alguna que otra conversación técnica sobre sensores en dispositivos móviles, pero la gran mayoría de veces pasan desapercibidos, al menos hasta que fallen.

Para saber de qué sensores dispone un terminal se puede descargar cualquiera de las aplicaciones que hay en el Market. Yo he creado una aplicación sencilla cuyo código está basado en un ejercicio encontrado por la red [ref. 26] que ejecutado en el dispositivo, lista los sensores disponibles (Ver *anexos* [*Cód. 1*]).

"La clase comienza indicando el Layout de la actividad y obteniendo el TextView de salida, donde mostraremos los resultados. A continuación vamos a utilizar el método getSystemService para solicitar al sistema servicios específicos. Este método pertenece a la clase Context y será muy utilizado para acceder a gran cantidad de servicios del sistema. Al indicar como parámetro SENSOR_SERVICE, indicamos que queremos utilizar los sensores. Lo haremos a través del objeto SensorManager. En primer lugar llamamos al método getSensorList () del objeto para que nos de la lista de objetos Sensor. La siguiente línea recorre todos los elementos de esta lista parar llamar a su método getName () para mostrar el nombre del sensor" [ref. 26].

Al ejecutar este programa en el Sony Ericsson XPERIA esta es la lista de sensores resultante:

- Sensores BMA150 Accelerometer: Como su nombre lo indica es el sensor que mide la aceleración relativa a la caída libre, como referencia, de mi dispositivo móvil.
- AK8973 Compass: El compás es un sensor que detecta la dirección a la que el dispositivo está orientado.
- AK8973 Compass Raw
- AK8973 Magnetic field: Sensor de medida del campo magnético terrestre.
- APDS 9120 Proximity: Un sensor de proximidad para medir la cercanía del sensor del terminal a algún objeto
- Gravity Sensor: Sensor de medida del campo gravitatorio terrestre.
- Linear Acceleration Sensor: Sensor de medida de la aceleración lineal.
- Rotation Vector Sensor: Indica la rotación de mi móvil con respecto a los ejes.

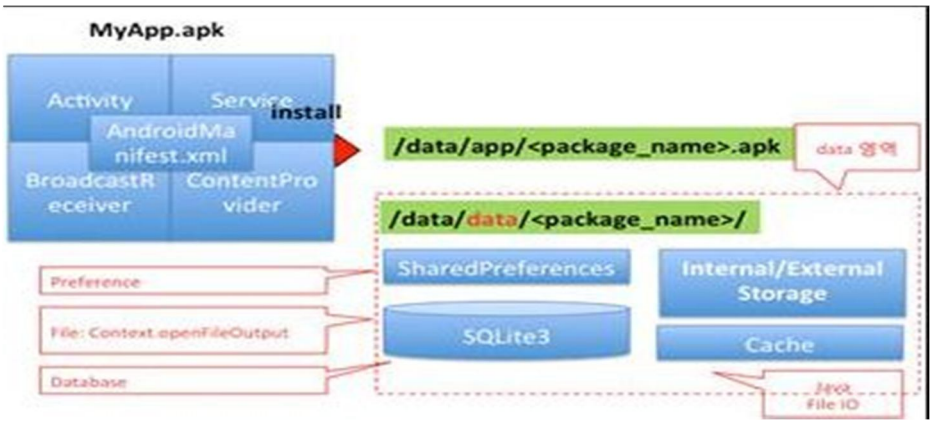
3.2 SENSORES

La aplicación que realizaremos se centra principalmente en el estudio de tres sensores, luminosidad, temperatura y acelerómetro y nos permitirá ver cómo se pueden usar, así como ver sus limitaciones a la hora de trabajar con ellos como un sensor en si o usarlo en una aplicación más compleja.

Se trata de almacenar los valores de estos sensores a la vez que se ejecuta la aplicación haciendo uso de bases de datos. Para ello y tras un estudio de las bases de datos concluimos que Android tiene su propia base de datos que es la más adecuada para usar en una aplicación móvil y que permite el almacenaje persistente de una pequeña cantidad de datos, de manera que aunque se salga de la aplicación los valores registrados quedaran guardados [ref. 27].

3.2.1. BASE DE DATOS SQLITE

SQLite es un gestor de bases de datos que nos permite guardar datos sin necesidad de mucha memoria y con la velocidad necesaria en entornos móviles [ref. 28].



Basándome en la artículo publicado acerca de SQLite [ref. 29] y [ref. 30] se concluye que a diferencia del resto de bases de datos, como Oracle, MySQL, y SQL Server que son procesos de servidor autónomos que se ejecutan independientemente, SQLite es una librería enlazada dentro de las aplicaciones, soporta las características estándar de las bases de datos relacionales como la sintaxis que se basa en SQL, transacciones y la elaboración de consultas.

Por otro lado, la API nativa de SQLite es incompatible con JDBC a pesar del uso de Java como el lenguaje nativo en las aplicaciones Android.

Android viene con SQLite preinstalado, y para poder manejarlo hay que tener una noción básica de las clases Java que interactúan con él.

SQLite soporta datos tipo TEXT (similar a los String en Java), INTEGER (similar a Integer en Java) y REAL (similar a Double en Java). Si usamos cualquier otro tipo de dato, éstos serán convertidos de manera automática para ser compatibles con estos tres tipos de datos.

3.2.2. CLASES DE JAVA SQLITE

- Cursor: Objeto que otorga el acceso a los resultados de una consulta de base de datos, se puede usar directamente a través de *SimpleCursorAdapter* para obtener información por medio de *ListViews* [ref. 29].

- SQLiteDatabase: Esta clase es la verdadera interfaz entre el código de la aplicación y la base de datos SQL. Incluye los métodos *insert (), update ()* y *delete ()* y *execSQL ()* que nos ayuda a ejecutar sentencias SQL directamente. Además de QUERY y RAWQUERY que son los métodos para realizar consultas [ref. 29].

- SQLiteOpenHelper: Es una clase auxiliar indispensable para la creación y modificación de la bases de datos. Debe sobreescribir los métodos onCreate () y onUpgrade () recibiendo un parámetro de tipo SQLiteDatabase [ref. 29].

- Para poder tener acceso de lectura y escritura sobre una base de datos, la clase SQLiteOpenHelper consta de los métodos getReadableDatabase () y getWriteableDatabase () [ref. 29].

3.3 SENSOR ACELEROMETRO, TEMPERATURA Y LUMINOSIDAD

Después de esta presentación de los sensores y de la base de datos SQLite llegamos a la parte en la que se unen ambas componentes de las que he hecho uso para el desarrollo de mi aplicación, y para que a posteriori en el punto 4 de este trabajo veamos los resultados experimentales y el significado de los valores obtenidos. Se procede al desarrollo del acelerómetro por la sencilla razón de que es algo más complejo que el resto al ser éste tridimensional, y requiere de procesado de datos.

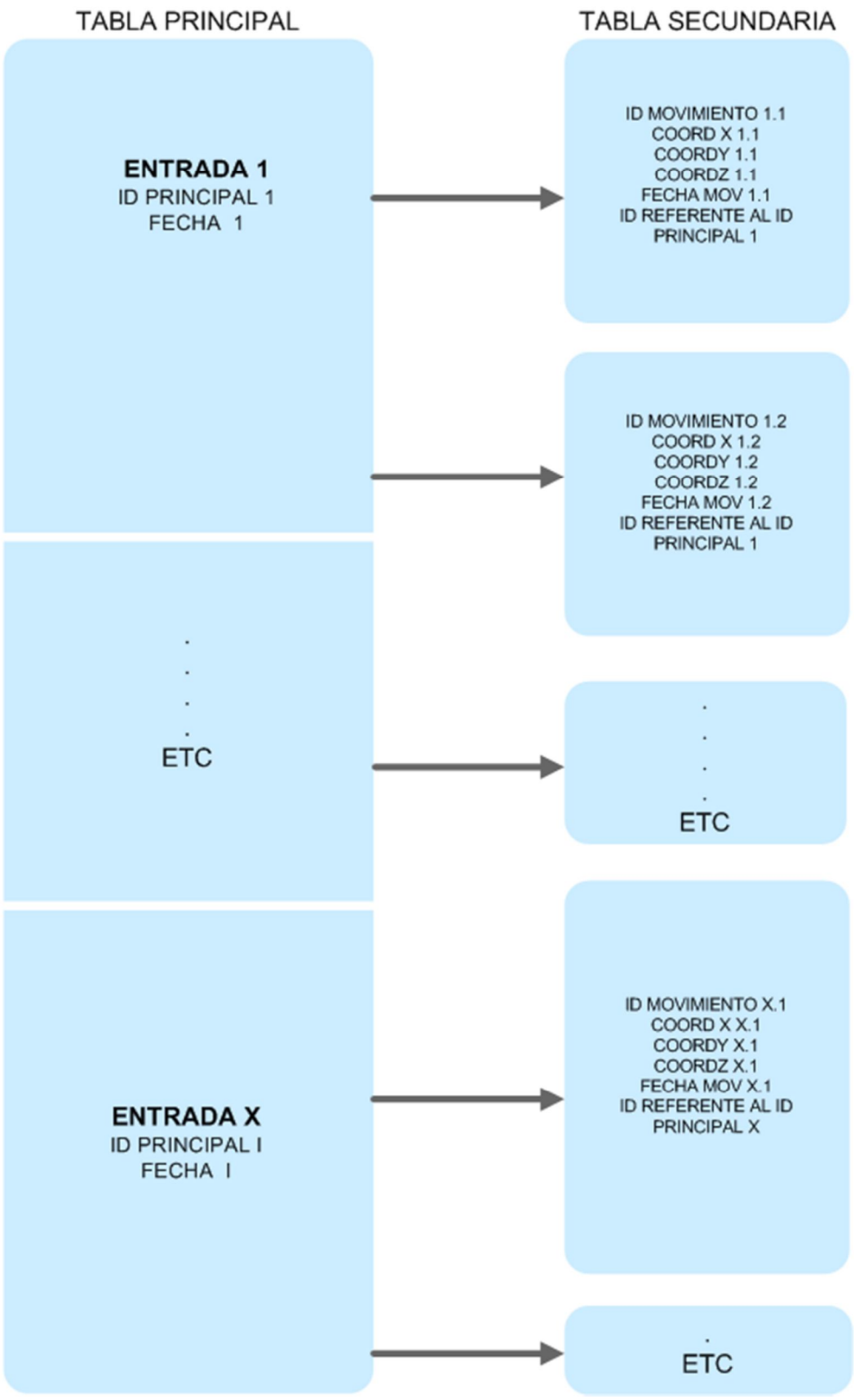
Se trata de que partiendo de la base de que el sensor acelerómetro cambia de valor cada vez que mi dispositivo móvil realiza un movimiento (que se traduce un cambio en las coordenadas tridimensionales (x, y, z)) y sabiendo que ese valor se borra con cada cambio nuevo, vamos a ir guardando en mi base de datos cada uno de estos movimientos para poder sacar conclusiones comparativas.

Vamos a someter el dispositivo a diferentes condiciones de movimiento a fin de averiguar cuáles son los valores a partir de los cuales o por debajo de los cuales este sensor deja de funcionar correctamente, es decir, deja de experimentar cambio en las coordenadas. Y para eso vamos a necesitar dos tablas, una primera tabla que sería la tabla principal y otra secundaria. Cada vez que iniciemos la aplicamos se produce una entrada en la tabla principal, y cada cambio producido en las coordenadas del sensor antes de salir de la aplicación o pararla se traduce en una entrada en la tabla secundaria. Para poder tener una idea más clara de la operación de entrada a la base de datos abajo se ve un diagrama de flujos de mi aplicación en la que se aprecian los atributos que tiene tanto la tabla principal como la secundaria y la forma en la que se relacionan que es mediante el Id principal.

Como vemos en el diagrama de flujo de la tabla secundaria, tenemos dos identificadores, el Id movimiento, un atributo especifico y único para cada acceso a la tabla secundaria, y un Id principal que sería común a todos los movimientos pertenecientes a ese arranque de aplicación, al ser el mismo Id que tenemos en la tabla principal estableciendo así una relación entre las dos tablas.

El funcionamiento es similar para los otros dos sensores usados con la peculiaridad de que el sensor luminosidad cambia con la luz ambiental mientras que el de temperatura varia con los cambios de temperatura que puede sufrir el dispositivo.

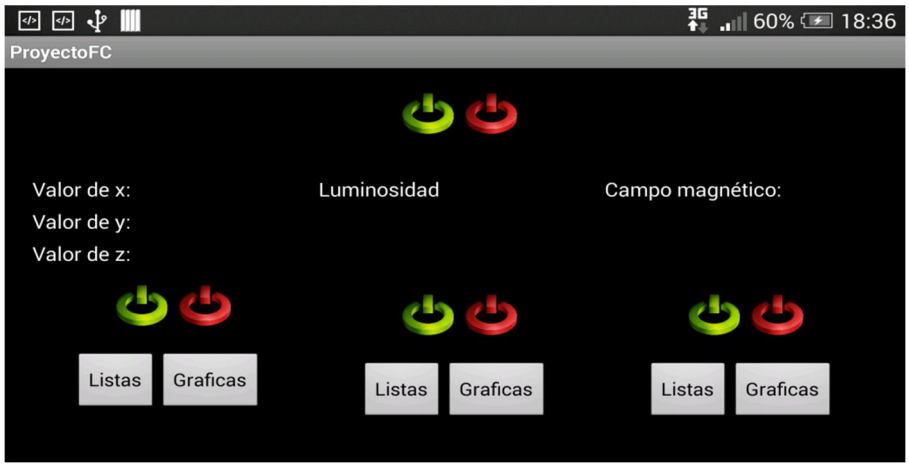
DIAGRAMA DE RELACIÓN ENTRE TABLAS



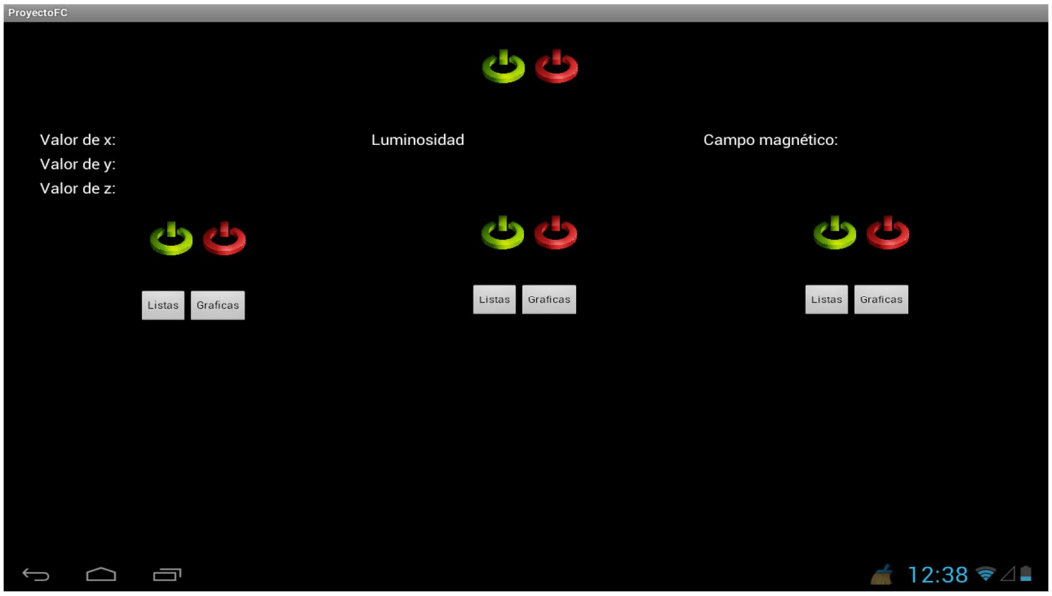
3.3.1. DETALLES DE DESARROLLO:

Con lo que respeta al código podemos dividir su desarrollo en fases (El desarrollo es similar para los tres sensores por lo que nos centraremos en el sensor acelerómetro):

• res->Layout :Lo primero que se hace es crear la parte estética de la aplicación en el Layout según las necesidad de ésta .Aquí abajo podemos ver el aspecto que tomara en el móvil HTC



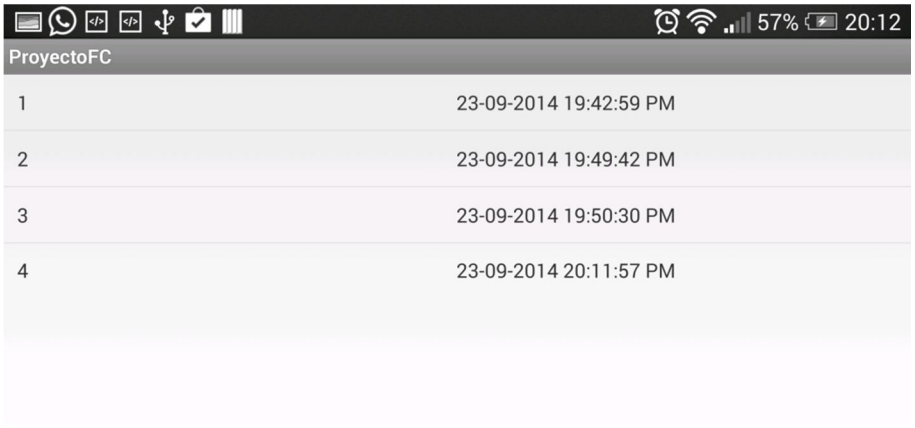
Debido a que la aplicación se ejecutara para su prueba en una tableta, para una mejor visualización se crea el layout-large-land que permite crear una representación vertical más adecuada al dispositivo que vamos a usar. Aquí abajo podemos ver el aspecto que tomara en la tableta.



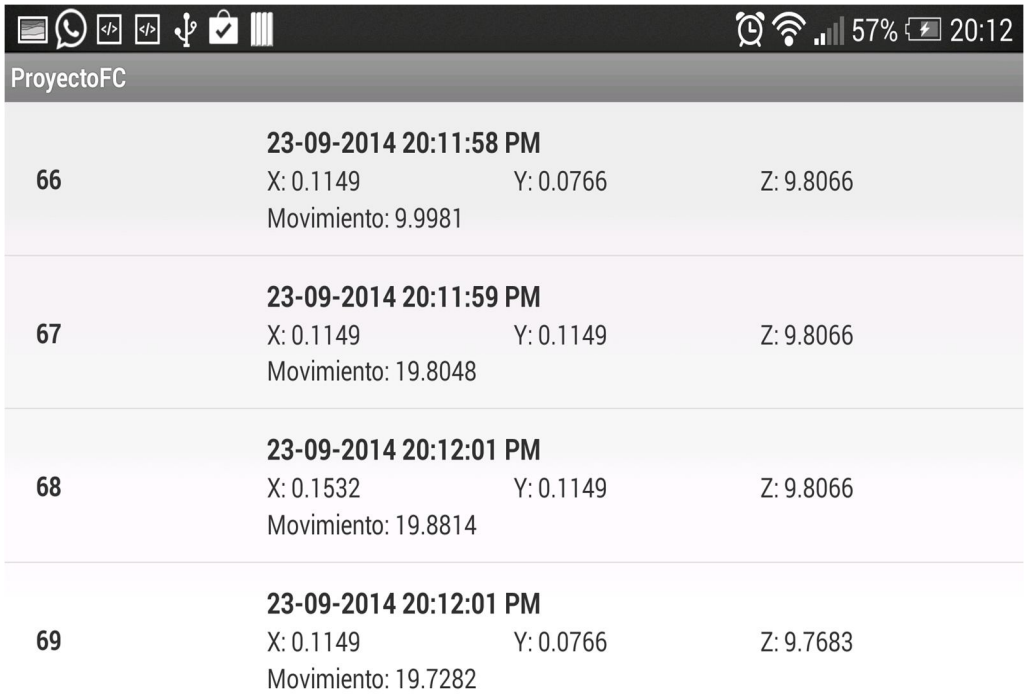
→ Se establecen dos botones genéricos .Un botón empezar para que se sincronicen los tres sensores y empiecen a grabar a la vez, y otro botón parar que hará que se pare la escucha de los cambios sensoriales y por lo consiguiente dejar de grabar en la base de datos.

-Activity_main: Tenemos 3 etiquetas de tipo TextView para mostrar los valores de x, y, z, un botón que verde que al pulsarlo empezaran a almacenarse los valores en nuestra base de datos, y otro botón rojo que al pulsarlo pararía el almacenamiento de datos en esa entrada. Tenemos además otros dos botones: Uno para acceder a las listas creadas (para poder visualizar la base de datos) y otro que al pulsarlo nos permite ver las gráficas (Ver anexos Código 2).

-Ademas de esta vista principal donde se mostrará la aplicación, vamos a tener una segunda vista donde podremos ver que la tabla principal que se crea y que tendrá el siguiente aspecto (Ver anexos Código 3):



Y otra tercera vista donde podremos visualizar los valores x, y, z de cada movimiento con el siguiente aspecto (Ver anexos Código 4):



Esto es posible gracias al uso de listas, por lo que creamos ítem principal, e ítem movimiento, además de crear dos archivos uno para manejar la lista principal que corresponde a la 2 Activity y la lista x, y, z para la 3 Activity (Ver anexos Código 5 y 6).

Por otro lado hay que crear la lista para visualizar las gráficas (Ver anexos Código 7)

Por ultimo para cambiar el icono de la aplicación procedemos a guardar la imagen escogida como símbolo de la aplicación en la carpeta /res/drawable/ con el mismo nombre con el que viene asignada por defecto para no modificar el manifestó (en caso de cambiar el nombre con el que guardamos la imagen habrá que modificarlo en el manifest).

• src->com->example->proyectofc->Lo siguiente es crear la parte funcional de la aplicación. Para ello y para poder tener un orden en el código, se divide el programa en carpetas para poder reutilizar el código y compartirlo de una forma más sencilla.

-Carpeta Adapter: En esta carpeta tenemos las clases adaptadoras necesarias para manejar listas. Cada lista necesita una clase adaptadora para administrar el modelo de datos y los adaptan a cada fila de la lista. En este caso tenemos la clase adaptador principal (Ver anexos Código 10) y adaptador movimiento (Ver anexos Código 11).

Vamos a explicar los métodos más importantes de esta carpeta omitiendo los getters, setters y constructores, y, teniendo en cuenta que se está haciendo un análisis del acelerómetro. (Las clases adaptadoras de los otros sensores tendrían los mismos métodos).

--Clase adaptador principal:

.public View getView (int position, View v, ViewGroup parent):
Devuelve la vista definida como ítem Principal.

--Clase adaptador Movimiento:

.public View getView (int position, View v, ViewGroup parent):
Devuelve la vista definida como ítem Movimiento.

.private String truncar (String valor_real): Trunca el número de decimales en las coordenadas x, y, z a 5.

-Carpeta DAO (Data Access Object): Aquí es donde se definen las conexiones a base de datos para obtener y guardar datos. Se define una clase abridor principal (Ver anexos Código 16) que extiende de SQLiteOpenHelper. Aquí es donde se definen todas las tablas a crear en el proyecto como listas. Por cada sensor se definen dos clases, una para insertar en la tabla principal (Ver anexos Código 17) y otra para insertar en la tabla secundaria (Ver anexos Código 18).

--Clase DAO principal:

.public long insert (): Se instancia un objeto de la clase ContentValues el cual almacenará los datos que introduciremos mediante el método values.put (nombre de la columna, valor). Y llamamos al método db.insert (nombre de la tabla, null, ContentValues) para insertar.

--Clase DAO Movimiento:

- public long insert (Elemento Movimiento elemento): Aquí le pasamos como parámetro un modelo creado de tipo movimiento. Se llama al método values.put (nombre de la columna, valor) tantas veces como parámetros tiene el elemento y se inserta.

-Carpeta Modelo: Permite crear instancia de objetos para cada clase DAO. No contiene más que los parámetros que definen cada tabla y sus métodos get y set (Ver anexos Código 23, 24).

-Clase MyApp: Esta clase nos permite coger el contexto (hay que declararla en el manifest) (Ver anexos Código 29).

-Clase tools: Se crea esta clase para poder tratar la excepción Java “Nan” al realizar la operación de truncar (Ver anexos Código 30).

– Carpeta gráficas: [ref. 32] Para poder visualizar las gráficas se hace uso de una librería de Android que brinda la posibilidad de visualizar distintas gráficas. Se hacen unas modificaciones a nivel de código para que se adecue a las necesidades de la aplicación.

Se crea una clase para cada sensor (Ver anexos Código 31).

Se parte de la base de una carpeta de clases para crear gráficas que extienden de la clase AbstarctDemoChart.

-Para cada una de las vistas de la aplicación necesitamos dos clases una clase principal (Ver anexos Código 37) y otra del sensor en cuestión (Ver anexos Código 38), extienden de Activity que es el main donde se crean y se cargan los elementos de nuestra lista.

Ademas de una clase que engloba toda nuestra aplicación y la enlaza .Este a su vez es un main que extiende de la clase Activity (Ver anexos Código 43) y que implementa la clase SensorEventListener para la escucha de los cambios de sensores, y llama a la base de datos para grabar estos valores y pintarlos. Se establece que la escucha sea cada segundo para una mejor apreciación de los cambios en los sensores.

--Clase main principal (Ver anexos Código 43):

.protected void onCreate (Bundle savedInstanceState): En este método vamos a instanciar los sensores a escuchar, se llamará por el sistema cuando comience la ejecución.

.private void acelerómetro (SensorManager manager): Aquí es donde se enlaza la interfaz de usuario con las variables llamando a la función findViewById (), y se establece la acción a realizar al pulsar cualquiera de los botones.

.public void onSensorChanged (SensorEvent event): Define la acción a realizar cuando hay un cambio en el sensor, llamando al metodo onAcelerometerChange ().

4. RESULTADOS EXPERIMENTALES:

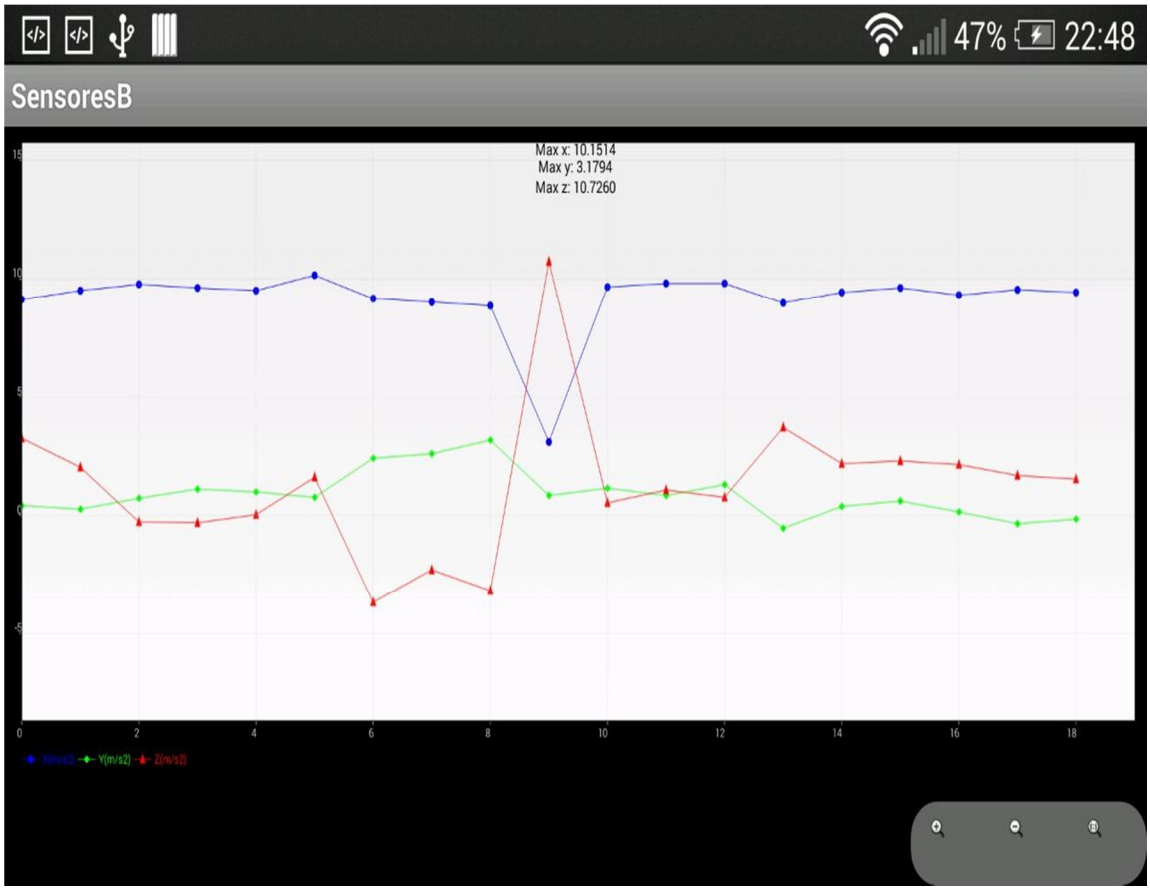
4.1 RESULTADOS PARA EL ACELEROMETRO:

Una vez instalada la aplicación y ejecutada se pueden comentar los resultados obtenidos.

Se observa que en momento de reposo de mi dispositivo (ya sea móvil o tableta) las coordenadas x e y tienen un valor de aproximadamente 0 (este desajuste se debe a la versión de Android que implementa cada dispositivo y a sus características hardware) mientras que la coordenada z marca el valor de la gravedad que también de un dispositivo a otro sufre alteración [ref. 31].

Después de hacer pruebas con este sensor se ve claramente que los valores de las coordenadas oscilan entre 0 y ± 10. Esto es lógico debido a que 9,81 m/s^2 que es el valor que corresponde a la aceleración en caída libre. El signo que adopta la aceleración en cada eje depende del sentido en el que se está realizando el movimiento en los 3 ejes.

Lógicamente los valores extremos se alcanzan cuando sometemos el dispositivo a pruebas limite, es decir cuando se procede a agitarlo o a tirarlo bruscamente para constatar sus limitaciones. En las gráficas abajo queda más claro los valores que registra en cada momento cada una de las coordenadas.



Por otro lado tenemos la gráfica del movimiento. Este parámetro es la velocidad resultante de la raíz cuadrada de la suma de las componentes de la aceleración en las tres componentes cartesianas.

En primer lugar se procede a despejar la velocidad en cada eje. Esto es equivalente a:

$V_x = \int a_x . dt$

$V_y = \int a_y . dt$

$V_z = \int a_z . dt$

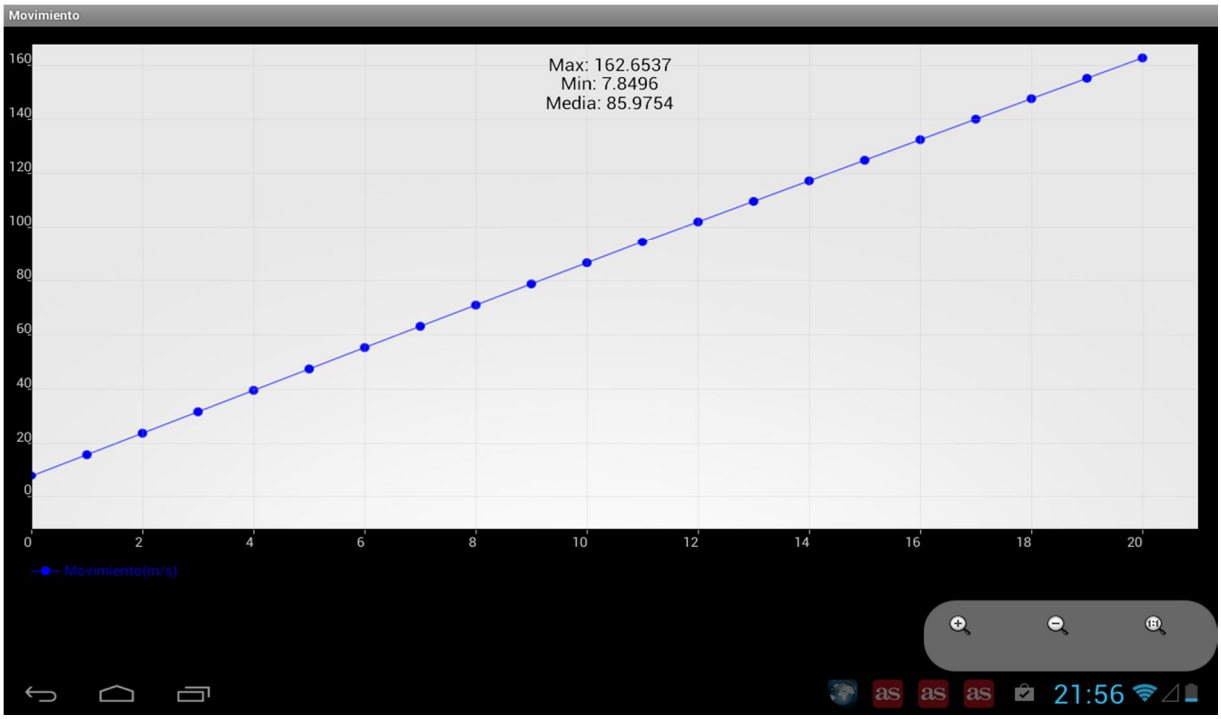
Como se ha fijado el tiempo de captura de datos sabemos que el tiempo siempre va a ser 1.

Y para facilitar el cálculo vamos a considerar la aproximación de que cada una de las integrales se puede expresar como un sumatorio.

Por lo tanto tenemos que el movimiento resultante en un instante i es, en (m/s):

$V_i = \sqrt{(a_x^2 + a_y^2 + a_z^2)}$

Para el siguiente instante la velocidad será la suma de $V_{i+1} + V_i$.



Tal y como se describió antes sabemos que incluso en momentos de reposo la coordenada z tendrá el valor de la aceleración de la gravedad de que oscila entre 0 y ± 10, este efecto se puede anular dando lugar al sensor acelerómetro lineal que en este caso no tiene en cuenta el efecto de la gravedad por lo que:

$a_z = (a_z \pm 9{,}8)\ m/s^2$

>si a_z es > 0 se hace una resta

>si a_z es < 0 se hace una suma.

4.2 RESULTADOS PARA EL SENSOR LUMINOSIDAD:

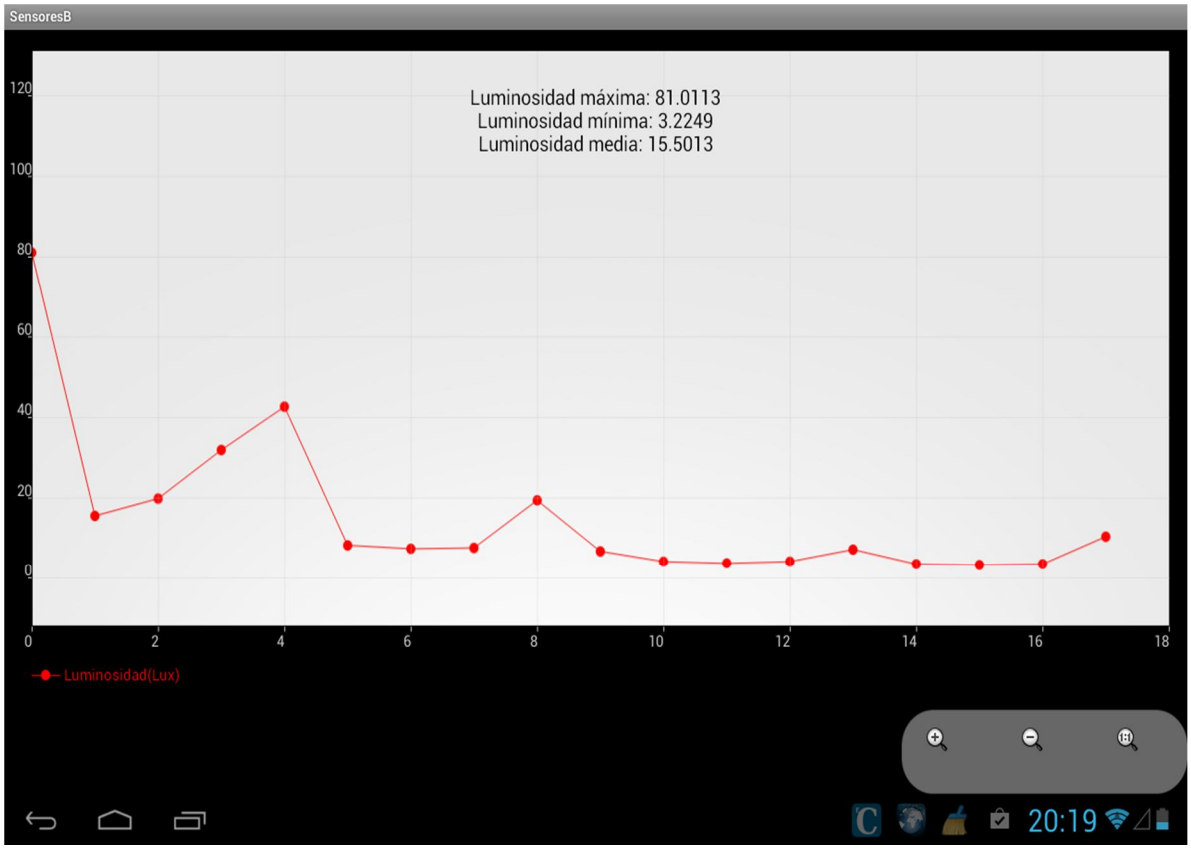
El objetivo de presentar este sensor es su importancia a la hora de controlar el brillo de la pantalla de nuestro dispositivo teniendo en cuenta lo que ello pueda suponer en el ahorro de batería.

Como lo definimos en el capítulo 3 este sensor es el responsable de realizar una adaptación automática al brillo de nuestra pantalla según la luminosidad percibida por el entorno donde nos encontremos en ese momento.

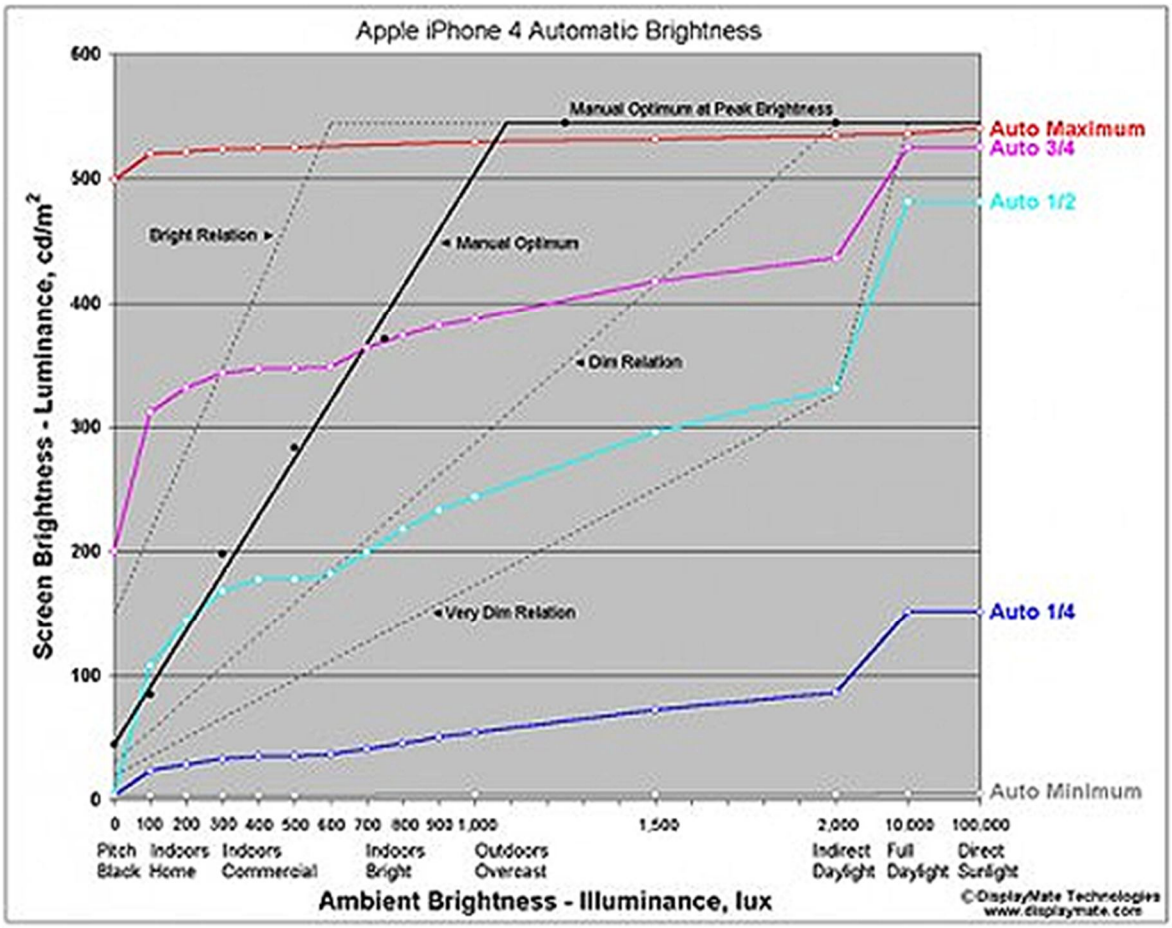
Este sensor puede no funcionar correctamente si el brillo automático está activado en nuestro terminal.

Al hacer pruebas con la aplicación vemos que el valor que toma estando en un lugar oscuro oscilan alrededor del 0 lux., mientras que en un lugar totalmente luminoso toma un valor que asciende a 1000 lux. Las consecuencias directas en ausencia de luz ambiental es que la función de auto-brillo del dispositivo se dispara para poder leer o ver la pantalla lo que supone un gran gasto de batería.

En nuestra aplicación se mostrara de la siguiente forma dependiendo de los valores que haya registrado el sensor:



En la siguiente imagen podemos ver una comparativa genérica entre el nivel de brillo y el nivel de iluminaria de un estudio, que aunque este realizado para iPhone es válido para aclarar lo que ocurre para dispositivos Android, ya que el comportamiento de este sensor aunque, la resolución depende del modelo y el software que tenga instalado cada dispositivo, no variará mucho y podemos formar una idea [ref. 33].



"Medición del brillo de la pantalla en diferentes condiciones de luz. Se determinó manualmente el nivel óptimo de luz mostrada por la línea de puntos negra. Los valores para cinco diferentes condiciones de autobrillo con el iPhone 4 se etiquetaron como AutoMínimo a Automáximo. Los círculos son puntos de información, las líneas guía muestran un amplio rango de relaciones de luz. El gráfico va de 0 a 2,000 lux y salta a 10,000 y 100,000 lux. Las etiquetas van a una obscuridad absoluta a la luz directa del sol y los niveles de Lux están asociados a estas condiciones. El máximo nivel de luminosidad del iPhone fue de 4 a 541 cd/m2 "[ref. 33].

4.3 RESULTADOS PARA EL MAGNETOMETRO:

Este sensor incrementa el coste de los dispositivos por lo que solo se encuentra en móviles de gama alta. Tiene dos funciones principales:

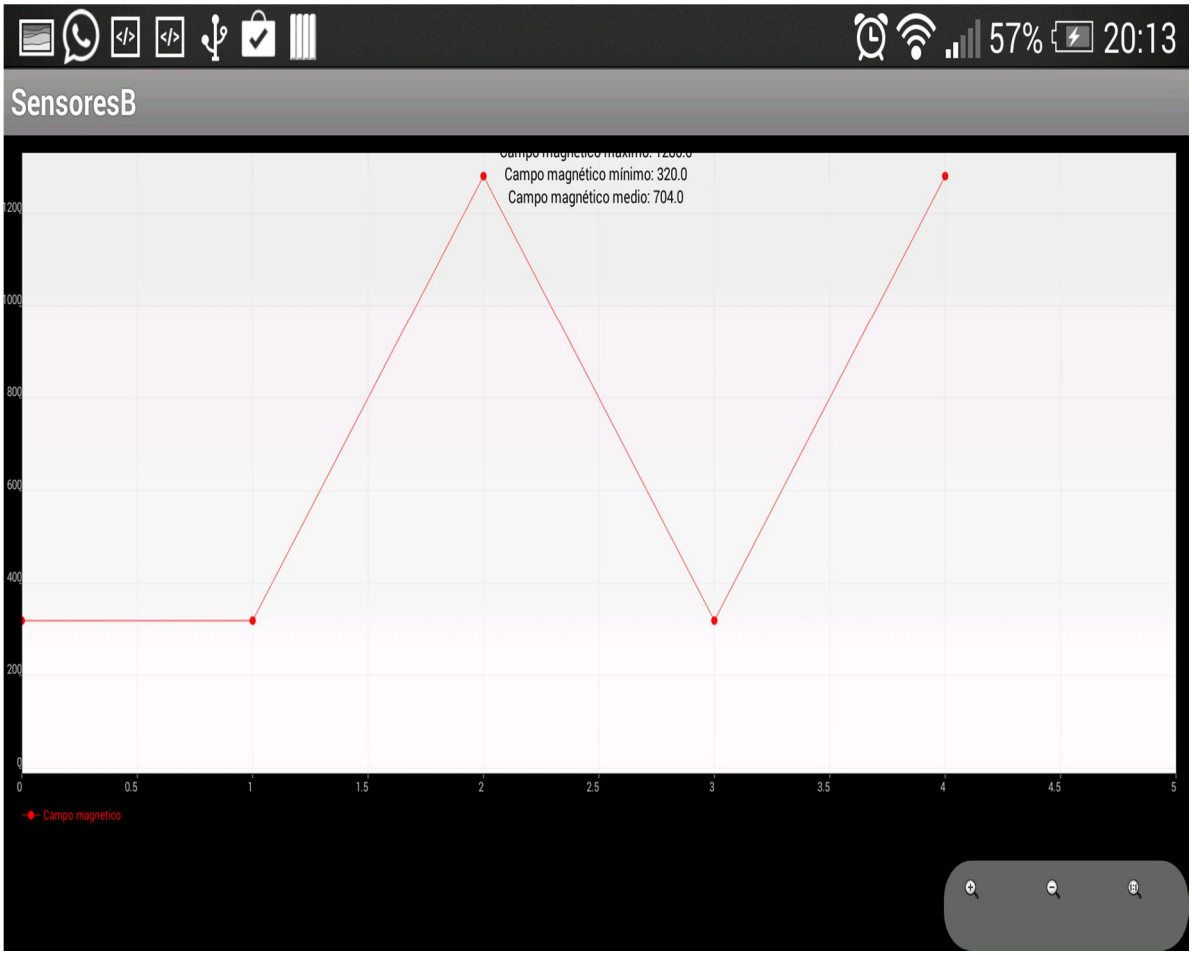
1. Ubicar el dispositivo con respecto a los polos de la tierra (en este caso funciona como brújula).
2. Detector de metales (Es la parte en la que se centra la aplicación).

Se observa que dependiendo de la superficie a la que acerquemos nuestro dispositivo una vez que iniciamos la aplicación, el detector de campo magnético marca un número u otro.

Hay que entender que en realidad lo que este sensor mide es el campo electromagnético, es decir que solo si hay una corriente eléctrica tendremos que este sensor toma valores. En caso de que solo tengamos un metal por el que no circula ninguna corriente este sensor no tiene sentido y dependiendo del voltaje de la corriente generada por el metal tendremos un valor u otro [ref. 34].

“La inducción magnética es la producción de una fuerza electromotriz a través de un conductor cuando se expone a un campo magnético variable. La densidad de flujo magnético, es el flujo magnético que causa una carga de difusión en movimiento por cada unidad de área normal a la dirección del flujo. La unidad de la densidad en el Sistema Internacional de Unidades el tesla.” [ref. 35]. Y esta magnitud es la que el sensor magnetómetro mide.

En la siguiente gráfica se muestra el cambio que refleja el cambio que experimenta el valor de este sensor al ponerlo encima del portátil donde marca un valor alto de 1300 uT, y al alejarlo cae este valor a 320 uT. Teniendo en cuenta que el nivel teórico del campo magnético en la naturaleza esta alrededor de 49uT(1uT=10mG).

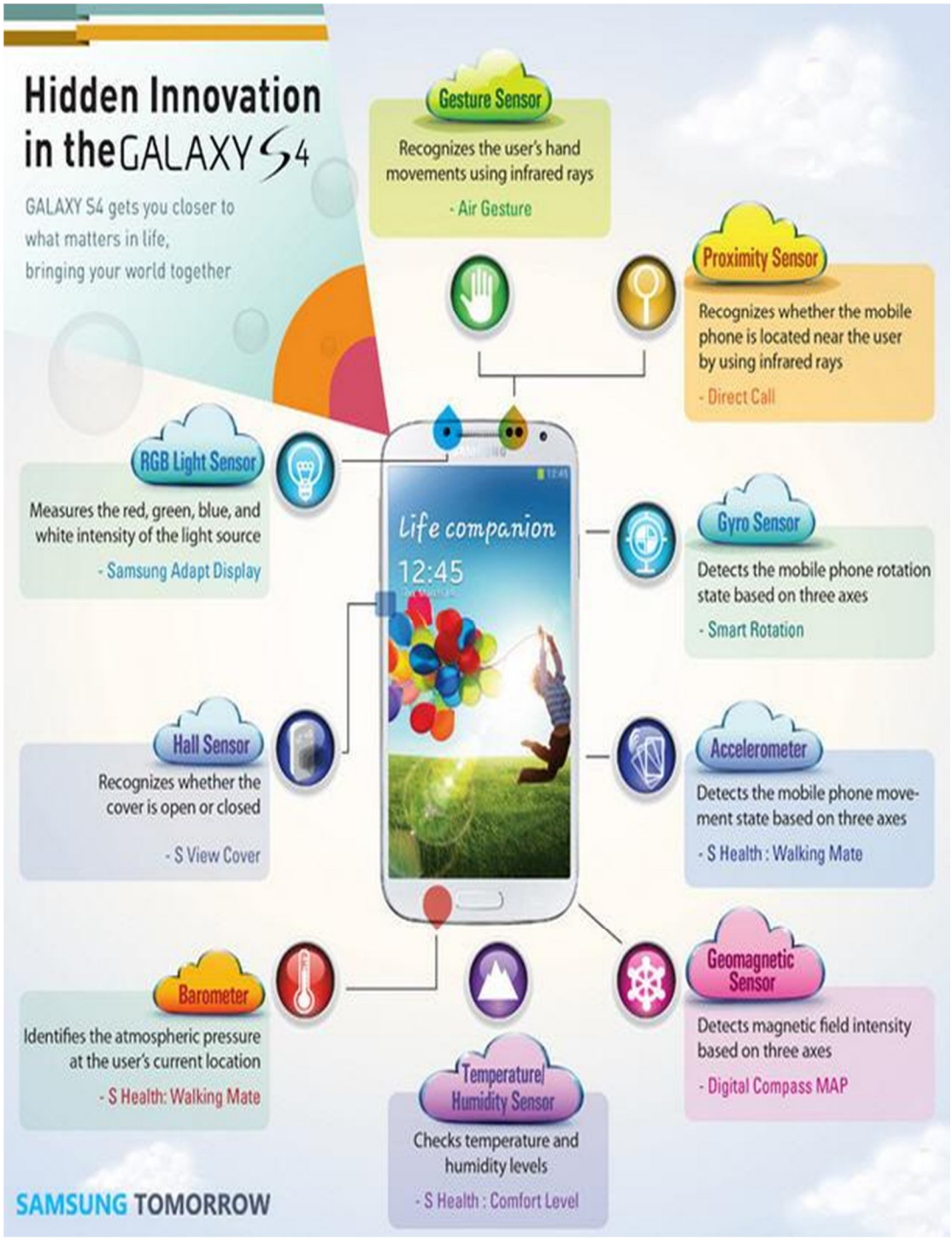

57%
20:13
SensoresB
Campo magnético mínimo: 320.0
Campo magnético medio: 704.0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Campo magnetico

5. CONCLUSIONES Y PROPUESTAS DE FUTURO

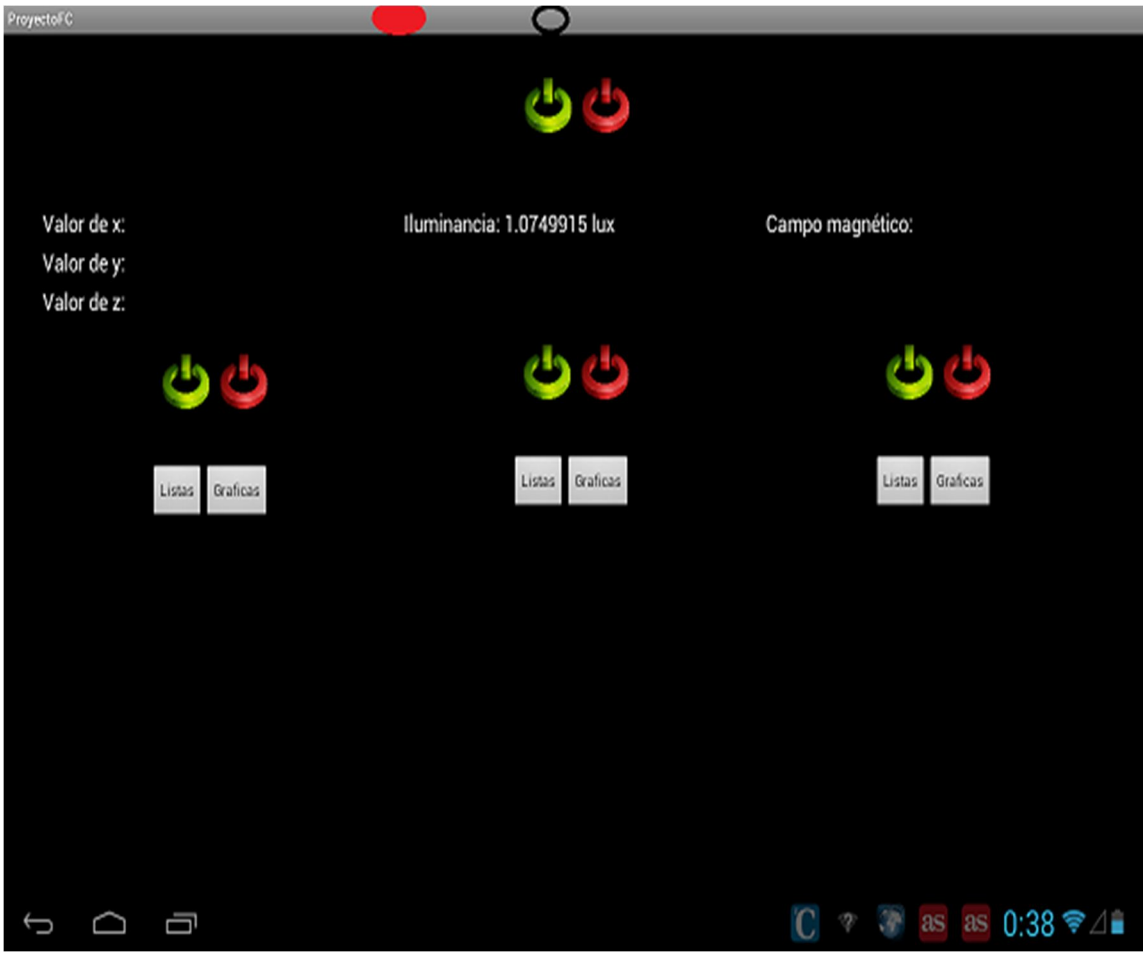
5.1 CONCLUSIONES:

Después de la elaboración de este trabajo y hacer una comparativa de resultados, se pueden sacar las siguientes conclusiones:

1. No todos los dispositivos móviles constan de los mismos sensores. En este caso se han realizado las pruebas en diferentes dispositivos. En la tableta Huwaei MediaPad 10 Link no se ha detectado la existencia de hardware que permite la ejecución del sensor campo magnético, por lo que se han realizado las pruebas en un móvil HTC One (M8) y de allí tenemos la gráfica mostrada en el apartado 4.3 referente a los resultados del sensor campo magnético.
2. Las versiones de Android tampoco son las mismas en los dispositivos, esto depende del momento de fabricación, y precio y repercute en las prestación que puede ofrecer una tecnología u otra [ref. 36]. Generalmente tiene mejores y mayores prestaciones las versiones más nuevas, ya que se intentan corregir errores o imprecisiones cometidas en las versiones anteriores. El móvil al ser nueve y de alta gama tiene instalada la última versión Android 4.4.2 KitKat, mientras que en la tableta tenemos instalada la versión Android 4.2 Jelly Bean, en este caso la diferencia se nota en el tiempo de respuesta que tiene una frente a la otra. Mientras que en el móvil la respuesta de los sensores al ejecutar la aplicación es inmediata, en la tableta se observa que va un poco más lenta.
3. Como consecuencia de los dos puntos anteriores hay una pequeña diferencia de los valores que marcan los sensores. Esto se pudo ver claramente en el capítulo 4 para los resultados del acelerómetro cuando se habló de que en reposo marcaba valores diferentes cuando se ejecutaba en el móvil, y cuando se ejecutaba en la tableta, e incluso ejecutándolo en el mismo dispositivo hay una pequeña variación en el valor numérico en una ejecución y en las siguientes.
4. Cuando se lanza una aplicación el sistema crea un hilo principal (thread) que se encarga de atender a los eventos y ejecutar los métodos onCreate () que incluye [ref. 37]. Por esta sencilla razón al intentar ejecutar la escucha de los tres sensores simultáneamente da la impresión que la aplicación se ha quedado bloqueada, porque internamente se está escuchando un cambio para tres sensores. Pero en realidad lo que se está haciendo son tres procesos que requieren de un tiempo de ejecución sobre todo si se está realizando un cálculo de movimiento para el sensor acelerómetro. Esto se puede subsanar creando hilos secundarios que sería una mejora futura para esta aplicación porque mejorara los tiempos de espera que se producen al ejecutar la aplicación.
5. Cabe aclarar la idea de que a pesar de que el sensor campo magnético es un detector de metales, tiene la limitación de la distancia, es decir que no hay que pensar que podremos encontrar un tesoro al ponerlo en marcha. Solo si se encuentra a una distancia muy cercana y que a su vez cree el metal cree un campo electromagnético se dispara su valor, en caso contrario la variación no será significativa.
6. Cabe destacar que el sensor de luz en los dispositivos móviles no es el foco de cámara sino que según el dispositivo su ubicación física sufre variaciones. En la siguiente imagen encontrada en la web que detalla los sensores del Samsung Galaxy S4 podemos ver donde se ubica tanto este sensor como el resto [ref. 38]:



Y en la siguiente imagen mostramos donde se encuentra en mi tableta Huwaei, que es donde realizo las pruebas teniendo en cuenta que el circulo negro seria el foco mientras que el círculo rojo seria la ubicación física del sensor brillo, donde en un ambiente luminoso el valor del sensor marcaba alrededor de 100 lux y al poner el dedo justo sobre la ubicación del círculo rojo marca 1 lux:



7. Finalmente destacar que la realización de este trabajo fue un reto personal para mí, ya que desconocía por completo el mundo Android como desarrollador. El aprendizaje comenzó desde la instalación del SDK y el Plugin hasta el análisis de los sensores y su funcionamiento pasando por el acceso a base de datos SQLite y el uso de librerías disponibles en Android para pintar las gráficas.

5.2 PROPUESTAS DE FUTURO:

En primer lugar la aplicación se puede mejorar como hemos indicado en el punto anterior introduciendo hilos, esto evitara los tiempos de espera a la hora de ejecutar los 3 sensores a la vez y la aplicación permitirá registrar valores en los tres sensores sin que se note el retraso entre uno y otro.

Por otro lado el sensor movimiento es de gran utilidad a la hora de diseñar aplicaciones de juegos para el móvil, especialmente los juegos basados en movimientos y velocidad, si este no funciona correctamente puede ser un problema ya que la aplicación no funcionara correctamente. En un artículo publicado por el MIT Technology Review [ref. 39] señala que este sensor puede servir para hacer avanzar a las aplicaciones de reconocimiento de gestos o incluso anticipar el estado mental.

"A un nivel muy sencillo, el usuario podría programar un gesto que sirviera de contraseña, un meneo o giro único del teléfono podría convertirse en la clave que desbloqueara la pantalla, una instrucción para llamar a casa o una señal de socorro. Ya hay algunos teléfonos, como el Moto X, que permiten a los usuarios llamar con una firma de movimiento concreta" [ref. 39], "El Moto X de Motorola: una innovadora interfaz pero con curva de aprendizaje" [ref. 40].

Incluso se ha introducido el uso de los sensores en la medicina en una rama que se denomina biosensores [ref. 41]. "La proliferación de biosensores permite medir casi cualquier parámetro fisiológico la tensión, por ejemplo, o la concentración de oxígeno en la sangre y esos datos pueden ser enviados de manera inalámbrica al médico y ser usados por el paciente para mantenerse saludable" [ref. 41]. Y en este ámbito se siguen realizando estudios para minimizar errores en la exactitud de medición de sensores.

Un uso casero y sencillo del sensor campo magnético puede ser útil a la hora de querer hacer un agujero en la pared, y querer saber de antemano si hay cables invisibles por debajo de la estructura, se enciende el interruptor para que pase corriente y se ejecuta el sensor. En caso de que tengamos campo electromagnético esto significa que si hay corriente eléctrica en caso de su ausencia se llega a la conclusión de que no hay cables, o al menos que estos no funcionan.

6. BIBLIOGRAFIA:

6.1 REFERENCIAS:

[ref. 1]: http://www.itu.int/ITU-D/ict/publications/wtdr_99/Spanish/page1-es.html
[ref. 2]: http://es.wikipedia.org/wiki/Google
[ref. 3]: http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/
[ref. 4]: http://www.poderpda.com/plataformas/android/android-evolucion-hasta-ice-cream-sandwich-infografia/
[ref. 5]: El Gran Libro de Android de Jesús Tomás Gironés/ Capitulo 1.1. ¿Qué hace que Android sea especial?
[ref. 6]: http://picandocodigo.net/2007/android-la-nueva-plataforma-de-desarrollos-moviles/
[ref. 7]: http://columna80.wordpress.com/2011/02/17/arquitectura-de-android/
[ref. 8]: http://www.maestrosdelweb.com/curso-android-construir-lector-de-feeds/
[ref. 9]: http://repositorio.bib.upct.es/dspace/bitstream/10317/3424/1/pfc5231.pdf
[ref. 10]: http://elbauldelprogramador.com/programacion-android-hola-mundo/
[ref. 11]: http://www.desarrolloweb.com/articulos/estructura-aplicacion-android-archivos- directorios.html
[ref. 12]: http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion
[ref. 13]: http://www.desarrolloweb.com/articulos/6-componentes-basicos-android.html
[ref. 14]: http://mundogeek.net/android/ciclo-vida-actividad/parando-actividad.htm
[ref. 15]: http://repositorio.bib.upct.es/dspace/bitstream/10317/3424/1/pfc5231.pdf
[ref. 16]: http://androideity.com/2011/09/29/eventos-en-el-ciclo-de-vida-de-actividades-android/ [ref. 17]: http://www.androidsis.com/el-ciclo-de-vida-de-una-aplicacion-de-android/
[ref. 18]: http://www.androidcurso.com/index.php/tutoriales-android/36-unidad-5-entradas- en-Android-teclado-pantalla-tactil-y-sensores/154-los-sensores.
[ref. 19]: http://developer.android.com/guide/topics/sensors/sensors_overview.html#
[ref. 20]: El gran libro de programación avanzada con Android/Capitulo 5 – José Enrique Amaro Soriano
[ref. 21]: http://desarrollojuegosandroid.blogspot.com.es/2011_11_01_archive.html
[ref. 22]: http://www.aprendeandroid.com/l9/sensores_android_movimiento.htm
[ref. 23]: http://www.cristalab.com/tutoriales/adapta-tu-web-a-la-luz-del-ambiente-con- javascript- y-css3-c113413l/
[ref. 24]: http://www.elandroidelibre.com/2013/10/sacale-todo-el-provecho-a-tu-galaxy-note- 3-con-estos-trucos-consejos-y-funciones.html
[ref. 25]: http://www.elandroidelibre.com/2013/04/tu-smartphone-puede-ser-mas-rapido-que- tu-ordenador-la-potencia-y-la-barrera-que-separa-a-los-dos-sistemas.html
[ref. 26]: http://www.androidcurso.com/index.php/tutoriales-android/36-unidad-5-entradas- en-android-teclado-pantalla-tactil-y-sensores/154-los-sensores
[ref. 27]: http://recursos.crfptic.es:9080/jspui/bitstream/recursos/146/20/3.6bis.pdf
[ref. 28]: http://androil.blogspot.com.es/2014/03/preferencias-de-una-app-y-base-de- datos.html
[ref. 29]: http://androideity.com/2011/10/12/manejo-de-bases-de-datos-en-android-i/
[ref. 30]: http://www.sqlite.org/about.html
[ref. 31]: http://developer.android.com/intl/es/reference/android/hardware/SensorEvent.html
[ref. 32]: http://blog.openalfa.com/como-dibujar-graficos-en-android
[ref. 33]: http://www.poderpda.com/noticias/por-que-los-controles-de-brillo-del-iphone-4- y-android-son-inutiles/

[ref. 34]: http://androidayuda.com/2013/04/21/la-app-estupida-de-la-semana-detector-de- metales/
[ref. 35]: http://es.wikipedia.org/wiki/Inducci%C3%B3n_magn%C3%A9tica
[ref. 36]: http://www.elandroidelibre.com/2013/07/android-4-3-todas-las-novedades-del-nuevo-sistema-operativo-de-google.html
[ref. 37]: http://www.elandroidelibre.com/2010/09/aprendiendo-android-vii-los-hilos-threads.html

[ref. 38]: http://www.xatakandroid.com/moviles-android/los-nueve-sensores-del-samsung- galaxy-s4-a-escena
[ref. 39]: http://www.technologyreview.es/read_article.aspx?id=43924
[ref. 40]: http://www.technologyreview.es/read_article.aspx?id=43625
[ref. 41]: http://www.el-nacional.com/tecnologia/Auge-sensores-smartphones-cambiara-medicina_0_193180692.html

6.2 ENLACES GENÉRICOS ANDROID:

http://www.youtube.com/watch?v=SUOWNXGRc6g&list=EC2F07DBCDCC01493A
http://www.youtube.com/watch?v=vgr4l1nsFdU
http://www.youtube.com/user/CornboyzAndroid?feature=watch
http://www.youtube.com/user/thenewboston

Sensor | Android Developers
SensorEvent | Android Developers
USB Host and Accessory | Android Developers
Accessory Development Kit | Android Developers
Build Accessories for Android | Android Open Source
USB Accessory | Android Developers
Building Custom Accessories | Android Open Source
Accessory Development Kit 2012 Guide | Android Developers
ADK 2012 - Aplicaciones de Android en Google Play

SQLITE:

http://androideity.com/2011/10/12/manejo-de-bases-de-datos-en-android-i/
https://www.udemy.com/blog/tutorial-de-android-sqlite-para-principiantes/
http://manuelignasch.wordpress.com/2013/05/22/bibliotecas-para-android-manejo-de-sensores-y-bases-de-datos/
http://www.sqlite.org/datatype3.html
http://es.slideshare.net/mejiaff/ejemplo-base-de-datos-sqlite-android

PARA PODER SIMULAR ELACELEROMETRO EN AVD:

http://rootear.com/android/comandos-adb-basicos
https://code.google.com/p/openintents/wiki/SensorSimulator

7. ANEXOS:

7.1 DESARROLLO DEL CODIGO DE LA APLICACION:

7.1.1. VISION GENERICA:

[*Cód. 1*]: Código para listar sensores:

```
package com.example.sensores;
import android.os.Bundle;
import android.view.Menu;
import java.util.List;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.app.Activity;
import android.widget.TextView;

public class MainActivity extends Activity {
private TextView salida;
      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            salida=(TextView)findViewById(R.id.salida);
            SensorManager sensorM
=(SensorManager)getSystemService(SENSOR_SERVICE);
            List<Sensor> listaSensores=
sensorM.getSensorList(Sensor.TYPE_ALL);
            for(Sensor sensor:listaSensores){
                  log(sensor.getName());
            }
                  }
      private void log(String name) {
            // TODO Auto-generated method stub
            salida.append(name +"\n");
      }
            public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it
is present.
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
      }
}
```

7.2 CODIGO SENSOR:

7.2.1. DETALLES DE DESARROLLO **:**

[*Cod. 2*]: Código para el layout -->Main principal:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="15dp"
    android:background="#000"
    tools:context="com.example.proyectofc.MainActivity" >

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

        <Button
              android:id="@+id/btEmpezar_general"
              android:layout_width="45dp"
              android:layout_height="45dp"
              android:background="@drawable/start" />

           <Button
              android:id="@+id/btFinalizar_general"
              android:layout_width="45dp"
              android:layout_height="45dp"
              android:background="@drawable/stop" />

    </LinearLayout>

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

    <LinearLayout
    android:id="@+id/LinearLayout1"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Valor de x: "
        android:textSize="15sp"
        android:textColor="#FFF"
        android:id="@+id/txtAccX"
```

```
    android:paddingTop="20dp"
    android:paddingBottom="5dp"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Valor de y: "
    android:textColor="#FFF"
    android:id="@+id/txtAccY"
    android:paddingBottom="5dp"
    android:textSize="15sp"
    />

 <TextView

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Valor de z: "
    android:textColor="#FFF"
    android:id="@+id/txtAccZ"
    android:paddingBottom="5dp"
    android:textSize="15sp"
    />


 <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal"
     android:gravity="center"
     android:paddingTop="5dp" >

      <Button
          android:id="@+id/btEmpezar"
          android:layout_width="45dp"
          android:layout_height="45dp"
          android:background="@drawable/start"/>

      <Button
          android:id="@+id/btFinalizar"
          android:layout_width="45dp"
          android:layout_height="45dp"
          android:background="@drawable/stop"/>

 </LinearLayout>


 <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal"
     android:gravity="center"
     android:paddingTop="5dp" >

      <Button
          android:id="@+id/btListas"
          android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="Listas"
            android:layout_gravity="center" />

        <Button
            android:id="@+id/btGraficas"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="Graficas"
            android:layout_gravity="center" />

  </LinearLayout>

  </LinearLayout>

<LinearLayout
android:id="@+id/LinearLayout2"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="5dp" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Luminosidad "
    android:textSize="15sp"
    android:textColor="#FFF"
    android:id="@+id/luz"
    android:paddingTop="20dp"
    android:paddingBottom="63dp"
    />


  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:gravity="center"
      android:paddingTop="5dp" >
       <Button
            android:id="@+id/btEmpezar2"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:background="@drawable/start"/>

       <Button
            android:id="@+id/btFinalizar2"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:background="@drawable/stop" />

  </LinearLayout>
```

```
  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:gravity="center"
      android:paddingTop="5dp" >

       <Button
           android:id="@+id/btListas2"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="5dp"
           android:text="Listas"
           android:layout_gravity="center" />

       <Button
           android:id="@+id/btGraficas2"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="5dp"
           android:text="Graficas"
           android:layout_gravity="center" />

  </LinearLayout>

  </LinearLayout>

<LinearLayout
android:id="@+id/LinearLayout3"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="5dp" >

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Campo magnÃ©tico: "
    android:textSize="15sp"
    android:textColor="#FFF"
    android:id="@+id/magnetometro"
    android:paddingTop="20dp"
    android:paddingBottom="63dp"
    />


  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:gravity="center"
      android:paddingTop="5dp" >
       <Button
           android:id="@+id/btEmpezar3"
           android:layout_width="45dp"
           android:layout_height="45dp"
```

```
                android:background="@drawable/start" />

            <Button
                android:id="@+id/btFinalizar3"
                android:layout_width="45dp"
                android:layout_height="45dp"
                android:background="@drawable/stop" />

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center"
            android:paddingTop="5dp" >

            <Button
                android:id="@+id/btListas3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="Listas"
                android:layout_gravity="center" />

            <Button
                android:id="@+id/btGraficas3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="Graficas"
                android:layout_gravity="center" />

        </LinearLayout>

        </LinearLayout>

</LinearLayout>

</LinearLayout>
```

[*Cod. 3*]:Código para el layout → lista principal :

```
<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" >

    <ListView
        android:id="@+id/lista_principal"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

[*Cod. 4*]:Código para el layout → lista xyz :

```
<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" >

    <ListView
        android:id="@+id/lista_xyz"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

[*Cod. 5*]:Código para el layout → item principal:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="10dp" >

    <TextView
        android:id="@+id/tv_id"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_fecha"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="wrap_content"
        android:text="TextView" />

</LinearLayout>
```

[*Cod. 6*]:Código para el layout → item movimiento:

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

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal"
android:padding="10dp" >

<TextView
    android:id="@+id/tv_id"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:padding="10dp"
    android:textStyle="bold"
    android:text="TextView" />

      <LinearLayout
     android:layout_width="0dp"
    android:layout_weight="3"
     android:layout_height="wrap_content"
     android:orientation="vertical" >

         <TextView
              android:id="@+id/tv_fecha"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:textSize="15sp"
              android:textStyle="bold"
              android:text="TextView" />


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

            <TextView
                    android:id="@+id/tv_x"
                    android:layout_width="0dp"
                    android:layout_weight="2"
                    android:layout_height="wrap_content"
                    android:text="TextView" />

               <TextView
                    android:id="@+id/tv_y"
                    android:layout_width="0dp"
                    android:layout_weight="2"
                    android:layout_height="wrap_content"
                    android:text="TextView" />

               <TextView
                    android:id="@+id/tv_z"
                    android:layout_width="0dp"
                    android:layout_weight="2"
                    android:layout_height="wrap_content"
                    android:text="TextView" />

          </LinearLayout>
```

```
            <TextView
            android:id="@+id/tv_movimiento"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView" />

        </LinearLayout>

</LinearLayout>
```

[*Cod. 7*]:Código para el layout → gráficas :

```
<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="com.example.demoplanoxy.MainActivity" >

    <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView" />

    <LinearLayout android:id="@+id/chart"
android:orientation="horizontal"
                android:layout_width="fill_parent"
android:layout_height="wrap_content" >

    </LinearLayout>

</LinearLayout>
```

[*Cod. 8*]: Código para el layout → item luminosidad:

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

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

```
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_luz_id_principal"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_luz_fecha"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_luz_luminosidad"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

</LinearLayout>
```

[Cod. 9]: Código para el layout → item campo magnético :

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

    <TextView
        android:id="@+id/tv_magnetico_id"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_magnetico_id_principal"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_magnetico_fecha"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
```

```
        android:text="TextView" />

    <TextView
        android:id="@+id/tv_magnetico_magnetismo"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="wrap_content"
        android:text="TextView" />

</LinearLayout>
```

[Cod. 10]: Código para la carpeta adapter → movimiento principal :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoPrincipal;

public class AdaptadorMovimientoPrincipal extends
ArrayAdapter<ElementoPrincipal> {

    private final List<ElementoPrincipal> list;
    private LayoutInflater inflater = null;
    private ElementoPrincipal item;

    public AdaptadorMovimientoPrincipal(Activity activity,
List<ElementoPrincipal> list) {
          super(activity, R.layout.item_principal);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }

    @Override
    public ElementoPrincipal getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
```

```
    }


     @Override
     public View getView(int position, View v, ViewGroup parent) {
          ItemPrincipal viewHolder;

        if (v == null) {

            v = inflater.inflate(R.layout.item_principal, null);

            viewHolder = new ItemPrincipal();
            viewHolder.tv_id = (TextView) v.findViewById(R.id.tv_id);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_fecha);

            v.setTag(viewHolder);

        }
        else {
            viewHolder = (ItemPrincipal) v.getTag();
        }

        item = list.get(position);

        if (item != null) {

            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
        }

        return v;
     }

     static class ItemPrincipal {
        TextView tv_id;
        TextView tv_fecha;
    }
}
```

[Cod. 11]: Código para la carpeta adapter → movimiento :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
```

```
import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoMovimiento;

public class AdaptadorMovimiento extends ArrayAdapter<ElementoMovimiento>
{

    private final List<ElementoMovimiento> list;
    private LayoutInflater inflater = null;
    private ElementoMovimiento item;

    public AdaptadorMovimiento(Activity activity, List<ElementoMovimiento>
list) {
          super(activity, R.layout.item_movimiento);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }

    @Override
    public ElementoMovimiento getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
    }



     @Override
     public View getView(int position, View v, ViewGroup parent) {
          ItemMovimiento viewHolder;

        if (v == null) {

            v = inflater.inflate(R.layout.item_movimiento, null);

            viewHolder = new ItemMovimiento();
            viewHolder.tv_id = (TextView) v.findViewById(R.id.tv_id);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_fecha);
            //viewHolder.tv_id_principal = (TextView)
v.findViewById(R.id.tv_id_principal);
            viewHolder.tv_x = (TextView) v.findViewById(R.id.tv_x);
            viewHolder.tv_y = (TextView) v.findViewById(R.id.tv_y);
            viewHolder.tv_z = (TextView) v.findViewById(R.id.tv_z);
            viewHolder.tv_movimiento = (TextView)
v.findViewById(R.id.tv_movimiento);

            v.setTag(viewHolder);
```

```
        }
        else {
            viewHolder = (ItemMovimiento) v.getTag();
        }

        item = list.get(position);

        if (item != null) {
           Log.d("Grabando", item.toString());
           Log.d("Grabando", ""+position);
            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
                        viewHolder.tv_x.setText("X:
"+truncar(""+item.getX()));
            viewHolder.tv_y.setText("Y: "+truncar(""+item.getY()));
            viewHolder.tv_z.setText("Z: "+truncar(""+item.getZ()));
            viewHolder.tv_movimiento.setText("Movimiento:
"+truncar(""+item.getMovimiento()));
                           }
        }

        return v;
     }

     private String truncar(String valor_real){

           Log.d("truncar",valor_real);
           if("0.0".equals(valor_real)){
                int i = 10;
                i++;
           }
           int punto = valor_real.indexOf('.');
           String pre = valor_real.substring(0, punto);
           String post = valor_real.substring(punto+1,
valor_real.length());

           if(post.length()>4){
                post = post.substring(0, 4);
           }

           return pre+"."+post;

     }

     static class ItemMovimiento {
        TextView tv_id;
        TextView tv_id_principal;
        TextView tv_fecha;

        TextView tv_x;
        TextView tv_y;
        TextView tv_z;
        TextView tv_movimiento;
    }
}
```

[Cod. 12]: Código para la carpeta adapter → luz principal :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoLuzPrincipal;

public class AdaptadorLuzPrincipal extends
ArrayAdapter<ElementoLuzPrincipal> {

    private final List<ElementoLuzPrincipal> list;
    private LayoutInflater inflater = null;
    private ElementoLuzPrincipal item;

    public AdaptadorLuzPrincipal(Activity activity,
List<ElementoLuzPrincipal> list) {
          super(activity, R.layout.item_principal);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }

    @Override
    public ElementoLuzPrincipal getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
    }



     @Override
     public View getView(int position, View v, ViewGroup parent) {
          ItemPrincipal viewHolder;

       if (v == null) {

           v = inflater.inflate(R.layout.item_principal, null);

           viewHolder = new ItemPrincipal();
```

```
            viewHolder.tv_id = (TextView) v.findViewById(R.id.tv_id);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_fecha);

            v.setTag(viewHolder);

        }
        else {
            viewHolder = (ItemPrincipal) v.getTag();
        }

        item = list.get(position);

        if (item != null) {

            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
        }

        return v;
     }

     static class ItemPrincipal {
        TextView tv_id;
        TextView tv_fecha;
    }
}
```

[Cod. 13]: Código para la carpeta adapter → luz :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
```

```
import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoLuminosidad;

public class AdaptadorLuz extends ArrayAdapter<ElementoLuminosidad> {

    private final List<ElementoLuminosidad> list;
    private LayoutInflater inflater = null;
    private ElementoLuminosidad item;

    public AdaptadorLuz(Activity activity, List<ElementoLuminosidad> list)
{
          super(activity, R.layout.item_luminosidad);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }

    @Override
    public ElementoLuminosidad getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
    }



     @Override
     public View getView(int position, View v, ViewGroup parent) {
          ItemLuz viewHolder;

        if (v == null) {

            v = inflater.inflate(R.layout.item_luminosidad, null);

            viewHolder = new ItemLuz();
            viewHolder.tv_id = (TextView) v.findViewById(R.id.tv_luz_id);
            viewHolder.tv_id_principal = (TextView)
v.findViewById(R.id.tv_luz_id_principal);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_luz_fecha);
            viewHolder.tv_luminosidad = (TextView)
v.findViewById(R.id.tv_luz_luminosidad);


            v.setTag(viewHolder);

        }
        else {
```

```
            viewHolder = (ItemLuz) v.getTag();
        }

        item = list.get(position);

        if (item != null) {
            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
            viewHolder.tv_id_principal.setText(""+item.getId_principal());
            viewHolder.tv_luminosidad.setText("Luminosidad:
"+truncar(""+item.getLuminosidad())+" uT");
        }

        return v;
     }


     static class ItemLuz {
        TextView tv_id;
        TextView tv_id_principal;
        TextView tv_fecha;
        TextView tv_luminosidad;
    }

     private String truncar(String valor_real){

           Log.d("truncar",valor_real);
           if("0.0".equals(valor_real)){
                int i = 10;
                i++;
           }
           int punto = valor_real.indexOf('.');
           String pre = valor_real.substring(0, punto);
           String post = valor_real.substring(punto+1,
valor_real.length());

           if(post.length()>4){
                post = post.substring(0, 4);
           }

           return pre+"."+post;

     }
}
```

[Cod. 14]: Código para la carpeta adapter → magnético principal :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoCmagneticoPrincipal;
import com.example.proyectofc.modelo.ElementoLuzPrincipal;

public class AdaptadorMagneticoPrincipal extends
ArrayAdapter<ElementoCmagneticoPrincipal> {

    private final List<ElementoCmagneticoPrincipal> list;
    private LayoutInflater inflater = null;
    private ElementoCmagneticoPrincipal item;

    public AdaptadorMagneticoPrincipal(Activity activity,
List<ElementoCmagneticoPrincipal> list) {
          super(activity, R.layout.item_principal);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }

    @Override
    public ElementoCmagneticoPrincipal getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
    }



     @Override
     public View getView(int position, View v, ViewGroup parent) {
           ItemPrincipal viewHolder;

        if (v == null) {

            v = inflater.inflate(R.layout.item_principal, null);

            viewHolder = new ItemPrincipal();
            viewHolder.tv_id = (TextView) v.findViewById(R.id.tv_id);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_fecha);

            v.setTag(viewHolder);

        }
        else {
            viewHolder = (ItemPrincipal) v.getTag();
```

```
        }

        item = list.get(position);

        if (item != null) {

            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
        }

        return v;
     }

     static class ItemPrincipal {
        TextView tv_id;
        TextView tv_fecha;
    }
}
```

[Cod. 15]: Código para la carpeta adapter → magnético :

```
package com.example.proyectofc.adapter;

import java.util.List;

import android.app.Activity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.proyectofc.R;
import com.example.proyectofc.modelo.ElementoCampoMagnetico;

public class AdaptadorMagnetico extends
ArrayAdapter<ElementoCampoMagnetico> {

    private final List<ElementoCampoMagnetico> list;
    private LayoutInflater inflater = null;
    private ElementoCampoMagnetico item;

    public AdaptadorMagnetico(Activity activity,
List<ElementoCampoMagnetico> list) {
          super(activity, R.layout.item_magnetismo);
          this.list = list;
          inflater = activity.getLayoutInflater();

    }

    @Override
    public int getCount() {
          return list.size();
    }
```

```
    @Override
    public ElementoCampoMagnetico getItem(int arg0) {
          return list.get(arg0);
    }

    @Override
    public long getItemId(int position) {
          return list.get(position).getId();
    }



     @Override
     public View getView(int position, View v, ViewGroup parent) {
          ItemLuz viewHolder;

        if (v == null) {

            v = inflater.inflate(R.layout.item_magnetismo, null);

            viewHolder = new ItemLuz();
            viewHolder.tv_id = (TextView)
v.findViewById(R.id.tv_magnetico_id);
            viewHolder.tv_id_principal = (TextView)
v.findViewById(R.id.tv_magnetico_id_principal);
            viewHolder.tv_fecha = (TextView)
v.findViewById(R.id.tv_magnetico_fecha);
            viewHolder.tv_magnetismo = (TextView)
v.findViewById(R.id.tv_magnetico_magnetismo);


            v.setTag(viewHolder);

        }
        else {
            viewHolder = (ItemLuz) v.getTag();
        }

        item = list.get(position);

        if (item != null) {
            viewHolder.tv_id.setText(""+item.getId());
            viewHolder.tv_fecha.setText(item.getFecha());
            viewHolder.tv_id_principal.setText(""+item.getId_principal());
            viewHolder.tv_magnetismo.setText("Campo magnÃ©tico:
"+truncar(""+item.getMagnetismo())+" uT");
        }

        return v;
     }


     static class ItemLuz {
        TextView tv_id;
        TextView tv_id_principal;
        TextView tv_fecha;
        TextView tv_magnetismo;
```

```
    }

     private String truncar(String valor_real){

          Log.d("truncar",valor_real);
          if("0.0".equals(valor_real)){
               int i = 10;
               i++;
          }
          int punto = valor_real.indexOf('.');
          String pre = valor_real.substring(0, punto);
          String post = valor_real.substring(punto+1,
valor_real.length());

          if(post.length()>4){
               post = post.substring(0, 4);
          }

          return pre+"."+post;

     }
}
```

[Cod. 16]: Código para la carpeta DAO → bases de datos :

```
package com.example.proyectofc.dao;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.example.proyectofc.MyApp;

public class BDAbridorPrincipal extends SQLiteOpenHelper {

     private static int version = 1;
     private static final String DATABASE_NAME = "proyectofc.db";


     public BDAbridorPrincipal() {
          super(MyApp.getContext(), DATABASE_NAME, null, version);
     }

     @Override
     public void onCreate(SQLiteDatabase db) {
          // Crear tablas de la base de datos
          //crear la base de datos con inicio y fin de movimiento
```

```
                String consultaCrearTabla = "CREATE TABLE tablaPrincipal
("
                           + "FechaHora text, "
                           + "IDPrincipal INTEGER PRIMARY KEY
AUTOINCREMENT"
                           + ");";
                //EJECUTA LA SENTENCIA DE CREAR LA TABLA

                db.execSQL(consultaCrearTabla);


                consultaCrearTabla = "CREATE TABLE tablaMovimiento ("
                           + "Movimiento DOUBLE, "
                           + "CoordX DOUBLE, "
                           + "CoordY DOUBLE, "
                           + "CoordZ DOUBLE, "
                           + "FechaHora text, "

                            /* SQLlite no tiene boolean : 0:FALSE, 1:TRUE
*/

                           + "IDPrincipal INTEGER, "
                           + "IDMovimiento INTEGER PRIMARY KEY
AUTOINCREMENT, "
                           + "FOREIGN KEY(IDPrincipal) REFERENCES
tablaPrincipal(IDPrincipal)"
                           + ");";
                db.execSQL(consultaCrearTabla);



                String consultaCrearTablaLuz = "CREATE TABLE
"+TablaLuzPrincipalDAO.TABLA+" ("
                           + TablaLuzPrincipalDAO.FECHA+" text, "
                           + TablaLuzPrincipalDAO.ID+" INTEGER PRIMARY
KEY AUTOINCREMENT"
                           + ");";

                db.execSQL(consultaCrearTablaLuz);


                consultaCrearTabla = "CREATE TABLE "+TablaLuzDAO.TABLA+"
("
                           + TablaLuzDAO.LUMINOSIDAD+" DOUBLE, "
                           + TablaLuzDAO.FECHAHORA+" text, "
                           + TablaLuzDAO.ID_PRINCIPAL+" INTEGER, "
                           + TablaLuzDAO.ID+" INTEGER PRIMARY KEY
AUTOINCREMENT, "
                           + "FOREIGN KEY("+TablaLuzDAO.ID_PRINCIPAL+")
REFERENCES "+TablaLuzPrincipalDAO.TABLA+"("+TablaLuzPrincipalDAO.ID+")"
                           + ");";
                db.execSQL(consultaCrearTabla);
```

```
                String consultaCrearTablaCM = "CREATE TABLE
"+TablaMagneticoPrincipalDAO.TABLA+" ("
                            + TablaMagneticoPrincipalDAO.FECHA+" text, "
                            + TablaMagneticoPrincipalDAO.ID+" INTEGER
PRIMARY KEY AUTOINCREMENT"
                            + ");";

                db.execSQL(consultaCrearTablaCM);


                consultaCrearTabla = "CREATE TABLE
"+TablaMagneticoDAO.TABLA+" ("
                            + TablaMagneticoDAO.MAGNETISMO+" DOUBLE, "
                            + TablaMagneticoDAO.FECHAHORA+" text, "
                            + TablaMagneticoDAO.ID_PRINCIPAL+" INTEGER, "
                            + TablaMagneticoDAO.ID+" INTEGER PRIMARY KEY
AUTOINCREMENT, "
                            + "FOREIGN
KEY("+TablaMagneticoDAO.ID_PRINCIPAL+") REFERENCES
"+TablaMagneticoPrincipalDAO.TABLA+"("+TablaMagneticoDAO.ID+")"
                            + ");";
                db.execSQL(consultaCrearTabla);


     }

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

          // Actualizar de la base de datos de la vieja versiï¿½n a la
nueva
          String consultaBorrarTabla = "DROP TABLE IF EXISTS
tablaPrincipal";
          db.execSQL(consultaBorrarTabla);

          consultaBorrarTabla = "DROP TABLE IF EXISTS tablaMovimiento";
          db.execSQL(consultaBorrarTabla);

          onCreate(db);
          // En este ejemplo se borra el contenido y se vuelve a crear
     }

}
```

[Cod. 17]: Código para la carpeta DAO → movimiento principal :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
```

```
import com.example.proyectofc.modelo.ElementoPrincipal;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class TablaPrincipalDAO {


     private final String TABLA = "tablaPrincipal";
     private final String FECHA = "FechaHora";
     private final String ID = "IDPrincipal";

     public List<ElementoPrincipal> getListas() {

          List<ElementoPrincipal> list = new
ArrayList<ElementoPrincipal>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA, null);

          ElementoPrincipal valueObject;
          if(cursor.moveToFirst()){

               do{
                    int id = cursor.getInt(cursor.getColumnIndex(ID));
                    String fecha =
cursor.getString(cursor.getColumnIndex(FECHA));
                    valueObject = new ElementoPrincipal(id, fecha);
                    list.add(valueObject);
               }while(cursor.moveToNext());
          }

          cursor.close();
          db.close();

          return list;
     }

     public long insert() {

        SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
        ContentValues values = new ContentValues();
        Calendar ahora = Calendar.getInstance();
        SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
          String f1 = df3.format(ahora.getTime());
        values.put(FECHA, f1);

        long id = db.insert(TABLA, null, values);
        return id;

     }


}
```

[Cod. 18]: Código para la carpeta DAO → movimiento :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.example.proyectofc.modelo.ElementoMovimiento;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

public class TablaMovimientoDAO {


     private final String TABLA = "tablaMovimiento";
     private final String MOVIMIENTO = "Movimiento";
     private final String COORX = "CoordX";
     private final String COORY = "CoordY";
     private final String COORZ = "CoordZ";
     private final String FECHAHORA = "FechaHora";
     private final String ID_PRINCIPAL = "IDPrincipal";
     private final String ID = "IDMovimiento";


     public List<ElementoMovimiento> getListas(int id_principal) {

          List<ElementoMovimiento> list = new
ArrayList<ElementoMovimiento>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          String[] datos = {""+id_principal};
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA+ " WHERE
"+ID_PRINCIPAL+"=?", datos);

          ElementoMovimiento valueObject;
          if(cursor.moveToFirst()){

               do{
                    int id = cursor.getInt(cursor.getColumnIndex(ID));
                    double x =
cursor.getDouble(cursor.getColumnIndex(COORX));
                    double y =
cursor.getDouble(cursor.getColumnIndex(COORY));
                    double z =
cursor.getDouble(cursor.getColumnIndex(COORZ));

                    String fecha =
cursor.getString(cursor.getColumnIndex(FECHAHORA));
                    valueObject = new ElementoMovimiento(id,
id_principal, x, y, z, fecha);
```

```
                list.add(valueObject);
            }while(cursor.moveToNext());
        }

        cursor.close();
        db.close();

        return list;
    }

    public long insert(ElementoMovimiento elemento) {

       SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
       ContentValues values = new ContentValues();
       Calendar ahora = Calendar.getInstance();
       SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
          String f1 = df3.format(ahora.getTime());
       values.put(ID_PRINCIPAL, elemento.getId_principal());
       values.put(FECHAHORA, f1);
       values.put(COORX, elemento.getX());
       values.put(COORY, elemento.getY());
       values.put(COORZ, elemento.getZ());

       long id = db.insert(TABLA, null, values);
       return id;

    }


}
```

[Cod. 19]: Código para la carpeta DAO → luz principal :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

import com.example.proyectofc.modelo.ElementoLuzPrincipal;

public class TablaLuzPrincipalDAO {


    public static final String TABLA = "tabla_luz";
    public static final String FECHA = "FechaHora";
    public static final String ID = "IDPrincipal";
```

```
     public List<ElementoLuzPrincipal> getListas() {

          List<ElementoLuzPrincipal> list = new
ArrayList<ElementoLuzPrincipal>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA, null);

          ElementoLuzPrincipal valueObject;
          if(cursor.moveToFirst()){

               do{
                    int id = cursor.getInt(cursor.getColumnIndex(ID));
                    String fecha =
cursor.getString(cursor.getColumnIndex(FECHA));
                    valueObject = new ElementoLuzPrincipal(id, fecha);
                    list.add(valueObject);
               }while(cursor.moveToNext());
          }

          cursor.close();
          db.close();

          return list;
     }

     public long insert() {

        SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
        ContentValues values = new ContentValues();
        Calendar ahora = Calendar.getInstance();
        SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
           String f1 = df3.format(ahora.getTime());
        values.put(FECHA, f1);

        long id = db.insert(TABLA, null, values);
        return id;

     }


}
```

[Cod. 20]: Código para la carpeta DAO → luz :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import android.content.ContentValues;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

import com.example.proyectofc.modelo.ElementoLuminosidad;

public class TablaLuzDAO {


     public static final String TABLA = "tabla_luminosidad";
     public static final String LUMINOSIDAD = "luminosidad";
     public static final String FECHAHORA = "FechaHora";
     public static final String ID_PRINCIPAL = "IDPrincipal";
     public static final String ID = "IDLuz";


     public List<ElementoLuminosidad> getListas(int id_principal) {

          List<ElementoLuminosidad> list = new
ArrayList<ElementoLuminosidad>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          String[] datos = {""+id_principal};
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA+ " WHERE
"+ID_PRINCIPAL+"=?", datos);

          ElementoLuminosidad valueObject;
          if(cursor.moveToFirst()){

               do{
                    int id = cursor.getInt(cursor.getColumnIndex(ID));
                    double x =
cursor.getDouble(cursor.getColumnIndex(LUMINOSIDAD));

                    String fecha =
cursor.getString(cursor.getColumnIndex(FECHAHORA));
                    valueObject = new ElementoLuminosidad(id,
id_principal, x, fecha);
                    list.add(valueObject);
               }while(cursor.moveToNext());
          }

          cursor.close();
          db.close();

          return list;
     }

     public long insert(ElementoLuminosidad elemento) {

        SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
        ContentValues values = new ContentValues();
        Calendar ahora = Calendar.getInstance();
        SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
          String f1 = df3.format(ahora.getTime());
        values.put(ID_PRINCIPAL, elemento.getId_principal());
```

```
        values.put(FECHAHORA, f1);
        values.put(LUMINOSIDAD, elemento.getLuminosidad());

        long id = db.insert(TABLA, null, values);
        return id;

     }


}
```

[Cod. 21]: Código para la carpeta DAO → magnético principal :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

import com.example.proyectofc.modelo.ElementoCmagneticoPrincipal;

public class TablaMagneticoPrincipalDAO {


     public static final String TABLA = "tabla_magnetica";
     public static final String FECHA = "FechaHora";
     public static final String ID = "IDPrincipal";

     public List<ElementoCmagneticoPrincipal> getListas() {

          List<ElementoCmagneticoPrincipal> list = new
ArrayList<ElementoCmagneticoPrincipal>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA, null);

          ElementoCmagneticoPrincipal valueObject;
          if(cursor.moveToFirst()){
```

```
                do{
                     int id = cursor.getInt(cursor.getColumnIndex(ID));
                     String fecha =
cursor.getString(cursor.getColumnIndex(FECHA));
                     valueObject = new ElementoCmagneticoPrincipal(id,
fecha);
                     list.add(valueObject);
                }while(cursor.moveToNext());
           }

           cursor.close();
           db.close();

           return list;
     }

     public long insert() {

        SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
        ContentValues values = new ContentValues();
        Calendar ahora = Calendar.getInstance();
        SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
           String f1 = df3.format(ahora.getTime());
        values.put(FECHA, f1);

        long id = db.insert(TABLA, null, values);
        return id;

     }


}
```

[Cod. 22]: Código para la carpeta DAO → magnético :

```
package com.example.proyectofc.dao;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

import com.example.proyectofc.modelo.ElementoCampoMagnetico;
import com.example.proyectofc.modelo.ElementoLuminosidad;

public class TablaMagneticoDAO {
```

```
     public static final String TABLA = "tabla_magnetismo_singular";
     public static final String MAGNETISMO = "magnetismo";
     public static final String FECHAHORA = "FechaHora";
     public static final String ID_PRINCIPAL = "IDPrincipal";
     public static final String ID = "IDLuz";


     public List<ElementoCampoMagnetico> getListas(int id_principal) {

          List<ElementoCampoMagnetico> list = new
ArrayList<ElementoCampoMagnetico>();
          SQLiteDatabase db = new
BDAbridorPrincipal().getReadableDatabase();
          String[] datos = {""+id_principal};
          Cursor cursor = db.rawQuery("SELECT * FROM "+TABLA+ " WHERE
"+ID_PRINCIPAL+"=?", datos);

          ElementoCampoMagnetico valueObject;
          if(cursor.moveToFirst()){

               do{
                    int id = cursor.getInt(cursor.getColumnIndex(ID));
                    double x =
cursor.getDouble(cursor.getColumnIndex(MAGNETISMO));

                    String fecha =
cursor.getString(cursor.getColumnIndex(FECHAHORA));
                    valueObject = new ElementoCampoMagnetico(id,
id_principal, x, fecha);
                    list.add(valueObject);
               }while(cursor.moveToNext());
          }

          cursor.close();
          db.close();

          return list;
     }

     public long insert(ElementoCampoMagnetico elemento) {

        SQLiteDatabase db = new
BDAbridorPrincipal().getWritableDatabase();
        ContentValues values = new ContentValues();
        Calendar ahora = Calendar.getInstance();
        SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss
a");
          String f1 = df3.format(ahora.getTime());
        values.put(ID_PRINCIPAL, elemento.getId_principal());
        values.put(FECHAHORA, f1);
        values.put(MAGNETISMO, elemento.getMagnetismo());

        long id = db.insert(TABLA, null, values);
        return id;

     }
```

```
}
```

[Cod. 23]: Código para la carpeta modelo → movimiento principal :

```
package com.example.proyectofc.modelo;

public class ElementoPrincipal {

	private int id;
	private String Fecha;


	public ElementoPrincipal(int id, String fecha) {
		this.id = id;
		Fecha = fecha;
	}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getFecha() {
		return Fecha;
	}
	public void setFecha(String fecha) {
		Fecha = fecha;
	}

	@Override
	public String toString() {
		return "ElementoPrincipal [id=" + id + ", Fecha=" + Fecha +
"]";
	}



}
```

[Cod. 24]: Código para la carpeta modelo → movimiento :

```
package com.example.proyectofc.modelo;

public class ElementoMovimiento {

	private int id;
	private int id_principal;
	private double x, y, z;
	private String fecha;
```

```
	public ElementoMovimiento(int id, int id_principal, double x,
double y,
			double z, String fecha) {
		this.id = id;
		this.id_principal = id_principal;
		this.x = x;
		this.y = y;
		this.z = z;
		this.fecha = fecha;
	}

	// public double getMovimiento(double prevX, double prevY, double
prevZ ){
	// double tiempo_dif = 1;
	// double mov = Math.abs((this.x +this.y + this.z) - (prevX - prevY
-
	// prevZ)) / tiempo_dif;
	//
	// return mov;
	// }

	public double getMovimiento() {

		double sum = Math.pow(this.x, 2) + Math.pow(this.y, 2) +
Math.pow(this.z, 2);
		double mov = Math.sqrt(sum);

		return mov;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public int getId_principal() {
		return id_principal;
	}

	public void setId_principal(int id_principal) {
		this.id_principal = id_principal;
	}

	public double getX() {
		return x;
	}

	public void setX(double x) {
		this.x = x;
	}

	public double getY() {
		return y;
```

```
	}

	public void setY(double y) {
		this.y = y;
	}

	public double getZ() {
		return z;
	}

	public void setZ(double z) {
		this.z = z;
	}

	public String getFecha() {
		return fecha;
	}

	public void setFecha(String fecha) {
		this.fecha = fecha;
	}

	@Override
	public String toString() {
		return "ElementoMovimiento [id=" + id + ", id_principal="
				+ id_principal + ", x=" + x + ", y=" + y + ", z=" +
z
				+ ", fecha=" + fecha + "]";
	}

}
```

[Cod. 25]: Código para la carpeta modelo → luz principal :

```
package com.example.proyectofc.modelo;

public class ElementoLuzPrincipal {

	private int id;
	private String Fecha;


	public ElementoLuzPrincipal(int id, String fecha) {
		this.id = id;
		Fecha = fecha;
	}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getFecha() {
		return Fecha;
	}
```

```
	public void setFecha(String fecha) {
		Fecha = fecha;
	}

	@Override
	public String toString() {
		return "ElementoPrincipal [id=" + id + ", Fecha=" + Fecha +
"]";
	}




}
```

[Cod. 26]: Código para la carpeta modelo → luz :

```
package com.example.proyectofc.modelo;

public class ElementoLuminosidad {

	private int id;
	private int id_principal;
	private double luminosidad;
	private String fecha;

	public ElementoLuminosidad(int id, int id_principal, double
luminosidad, String fecha) {
		this.id = id;
		this.id_principal = id_principal;
		this.luminosidad = luminosidad;
		this.fecha = fecha;
	}


	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public int getId_principal() {
		return id_principal;
	}

	public void setId_principal(int id_principal) {
		this.id_principal = id_principal;
	}

	public double getLuminosidad() {
		return luminosidad;
	}

	public void setLuminosidad(double x) {
```

```
		this.luminosidad = x;
	}

	public String getFecha() {
		return fecha;
	}

	public void setFecha(String fecha) {
		this.fecha = fecha;
	}
	}
```

[Cod. 27]: Código para la carpeta modelo → magnético principal :

```
package com.example.proyectofc.modelo;

public class ElementoCmagneticoPrincipal {

	private int id;
	private String Fecha;


	public ElementoCmagneticoPrincipal(int id, String fecha) {
		this.id = id;
		Fecha = fecha;
	}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getFecha() {
		return Fecha;
	}
	public void setFecha(String fecha) {
		Fecha = fecha;
	}

	@Override
	public String toString() {
		return "ElementoCmagneticoPrincipal [id=" + id + ", Fecha=" +
Fecha + "]";
	}



}
```

[Cod. 28]: Código para la carpeta modelo → magnético :

```
package com.example.proyectofc.modelo;

public class ElementoCampoMagnetico {

	private int id;
	private int id_principal;
	private double magnetismo;
	private String fecha;

	public ElementoCampoMagnetico(int id, int id_principal, double
magnetismo, String fecha) {
		this.id = id;
		this.id_principal = id_principal;
		this.magnetismo = magnetismo;
		this.fecha = fecha;
	}


	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public int getId_principal() {
		return id_principal;
	}

	public void setId_principal(int id_principal) {
		this.id_principal = id_principal;
	}

	public double getMagnetismo() {
		return magnetismo;
	}

	public void setMagnetismo(double x) {
		this.magnetismo = x;
	}

	public String getFecha() {
		return fecha;
	}

	public void setFecha(String fecha) {
		this.fecha = fecha;
	}



}
```

[Cod. 29]: Código para la clase MyApp :

```
package com.example.proyectofc;

import android.app.Application;
import android.content.Context;

public class MyApp extends Application {

	private static MyApp INSTANCE;

	public static Context getContext() {
		return INSTANCE.getApplicationContext();
	}

	@Override
	public void onCreate() {
		super.onCreate();
		INSTANCE = this;
	}

}
```

[Cod. 30]: Código para la clase tools :

```
package com.example.proyectofc.tools;

import android.util.Log;

public class Tools {


	public static String truncar(String valor_real){


		if(valor_real==null || "".equals(valor_real) ||
"NaN".equals(valor_real)){
			return "";
		}

		Log.d("truncar",valor_real);
		if("0.0".equals(valor_real)){
			int i = 10;
			i++;
		}
		int punto = valor_real.indexOf('.');
		String pre = valor_real.substring(0, punto);
		String post = valor_real.substring(punto+1,
valor_real.length());

		if(post.length()>4){
			post = post.substring(0, 4);
		}

		return pre+"."+post;

	}
```

```
}
```

[Cod. 31]: Código para la clase gráficas : Coord xyz

```
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.ArrayList;
import java.util.List;

import org.achartengine.ChartFactory;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint.Align;
import android.widget.Toast;

import com.example.proyectofc.MyApp;
import com.example.proyectofc.dao.TablaMovimientoDAO;
import com.example.proyectofc.modelo.ElementoMovimiento;
import com.example.proyectofc.tools.Tools;

/**
 * Average temperature demo chart.
 */
public class AcelerometroChart extends AbstractDemoChart {
      /**
       * Returns the chart name.
       *
       * @return the chart name
       */
      public String getName() {
            return "Average temperature";
      }

      /**
       * Returns the chart description.
       *
```

```
	 * @return the chart description
	 */
	public String getDesc() {
		return "The average temperature in 4 Greek islands (line
chart)";
	}

	/**
	 * Executes the chart demo.
	 *
	 * @param context
	 *            the context
	 * @return the built intent
	 */
	public Intent execute(Context context, int id) {

		TablaMovimientoDAO dao = new TablaMovimientoDAO();
		List<ElementoMovimiento> elementos = dao.getListas(id);

		String[] titles = new String[] { "X(m/s2)", "Y(m/s2)",
"Z(m/s2)" };

		//dominio
		List<double[]> x = new ArrayList<double[]>();
		for (int i = 0; i < titles.length; i++) {
			double[] dominio = new double[elementos.size()];
		       //x.add(new double[] { 1, 2, 3, 4, 5, 6, 7,7.5, 8, 9, 10,
11 });
			for (int j = 0; j < elementos.size(); j++) {
				dominio[j]=j;
			}
			x.add(dominio);
		}



		List<double[]> values = new ArrayList<double[]>();

		//f(t)=x
		double[] fx = new double[elementos.size()];
		double[] fy = new double[elementos.size()];
		double[] fz = new double[elementos.size()];


		double max_mov = Double.MIN_VALUE;
		double min_mov = Double.MAX_VALUE;
		double total_mov = 0;

		double max_x = Double.MIN_VALUE;
		double min_x = Double.MAX_VALUE;
		double total_x = 0;

		double max_y = Double.MIN_VALUE;
		double min_y = Double.MAX_VALUE;
		double total_y = 0;

		double max_z = Double.MIN_VALUE;
```

```
            double min_z = Double.MAX_VALUE;
            double total_z = 0;

            for (int i=0; i<elementos.size(); i++) {
                ElementoMovimiento elementoMovimiento = elementos.get(i);
                 fx[i] = elementoMovimiento.getX();
                 fy[i] = elementoMovimiento.getY();
                 fz[i] = elementoMovimiento.getZ();
                 //fmov[i] = elementoMovimiento.getMovimiento(prevX,
prevY, prevZ);
                 //fmov[i] = elementoMovimiento.getMovimiento();

                 //Totales
                 //total_mov = total_mov + fmov[i];
                 total_x = total_x + fx[i];
                 total_y = total_y + fy[i];
                 total_z = total_z + fz[i];

                 //Maximos
//               if(fmov[i]>max_mov){
//                    max_mov = fmov[i];
//               }
                 if(fx[i]>max_x){
                      max_x = fx[i];
                 }
                 if(fy[i]>max_y){
                      max_y = fy[i];
                 }
                 if(fz[i]>max_z){
                      max_z = fz[i];
                 }

                 //Minimos
//               if(fmov[i]<min_mov){
//                    min_mov = fmov[i];
//               }
                 if(fx[i]<min_x){
                      min_x = fx[i];
                 }
                 if(fy[i]<min_y){
                      min_y = fy[i];
                 }
                 if(fz[i]<min_z){
                      min_z = fz[i];
                 }

            }
            values.add(fx);
            values.add(fy);
            values.add(fz);
            //values.add(fmov);

            //Medias
            double media_mov = total_mov/(double)elementos.size();
            double media_x = total_x/(double)elementos.size();
            double media_y = total_y/(double)elementos.size();
            double media_z = total_z/(double)elementos.size();
```

```
          String mensaje = "";

          mensaje =  "\n\n\nMax x: "+Tools.truncar(""+max_x)+"\n"+
                          "Max y: "+Tools.truncar(""+max_y)+"\n"+
                          "Max z: "+Tools.truncar(""+max_z)+"\n";

          Toast.makeText(MyApp.getContext(), mensaje,
Toast.LENGTH_LONG).show();

          int[] colors = new int[] { Color.BLUE, Color.GREEN,
Color.RED };

          PointStyle[] styles = new PointStyle[] { PointStyle.CIRCLE,
     PointStyle.DIAMOND, PointStyle.TRIANGLE};

          XYMultipleSeriesRenderer renderer = buildRenderer(colors,
styles);

          int length = renderer.getSeriesRendererCount();
          for (int i = 0; i < length; i++) {
               ((XYSeriesRenderer)
renderer.getSeriesRendererAt(i)).setFillPoints(true);
          }

          double max_max = Math.max(max_x, Math.max(max_y,
Math.max(max_z, max_mov)))+5d;
          //setChartSettings(renderer,
"SensoresA\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal",
"Tiempo","R", 0, elementos.size(), -12, 12, Color.LTGRAY, Color.LTGRAY);
          setChartSettings(renderer, mensaje, "Tiempo","R", 0,
elementos.size(), 0, max_max, Color.BLACK, Color.BLACK);
          renderer.setXLabels(12);
          renderer.setYLabels(10);
          renderer.setShowGrid(true);
          renderer.setXLabelsAlign(Align.RIGHT);
          renderer.setYLabelsAlign(Align.RIGHT);
          renderer.setZoomButtonsVisible(true);
          renderer.setPanLimits(new double[] { 0, 20, 0, 40 });
          renderer.setZoomLimits(new double[] {0, 20, 0, 40 });

          XYMultipleSeriesDataset dataset = buildDataset(titles, x,
values);
          XYSeries series = dataset.getSeriesAt(0);
          //series.addAnnotation("Vacation", 6, 30);
          Intent intent = ChartFactory.getLineChartIntent(context,
dataset,renderer, "SensoresB");
          return intent;
     }}
```

[Cod. 32]: Código para la clase gráficas : movimiento

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
```

```
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.ArrayList;
import java.util.List;

import org.achartengine.ChartFactory;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint.Align;
import android.widget.Toast;

import com.example.proyectofc.MyApp;
import com.example.proyectofc.dao.TablaMovimientoDAO;
import com.example.proyectofc.modelo.ElementoMovimiento;
import com.example.proyectofc.tools.Tools;

/**
 * Average temperature demo chart.
 */
public class MovimientosChart extends AbstractDemoChart {
      /**
       * Returns the chart name.
       *
       * @return the chart name
       */
      public String getName() {
            return "Average temperature";
      }

      /**
       * Returns the chart description.
       *
       * @return the chart description
       */
      public String getDesc() {
            return "The average temperature in 4 Greek islands (line
chart)";
      }
```

```
     /**
      * Executes the chart demo.
      *
      * @param context
      *            the context
      * @return the built intent
      */
     public Intent execute(Context context, int id) {

          TablaMovimientoDAO dao = new TablaMovimientoDAO();
          List<ElementoMovimiento> elementos = dao.getListas(id);

          String[] titles = new String[] {"Movimiento(m/s)" };

          //dominio
          List<double[]> x = new ArrayList<double[]>();
          for (int i = 0; i < titles.length; i++) {
               double[] dominio = new double[elementos.size()];
              //x.add(new double[] { 1, 2, 3, 4, 5, 6, 7,7.5, 8, 9, 10,
11 });
               for (int j = 0; j < elementos.size(); j++) {
                    dominio[j]=j;
               }
               x.add(dominio);
          }



          List<double[]> values = new ArrayList<double[]>();

          double[] fmov = new double[elementos.size()];


          double max_mov = Double.MIN_VALUE;
          double min_mov = Double.MAX_VALUE;
          double total_mov = 0;

           double suma_movimientos = 0d;
        List<ElementoMovimiento> anteriores = new
ArrayList<ElementoMovimiento>();

          for (int i=0; i<elementos.size(); i++) {
              ElementoMovimiento elementoMovimiento = elementos.get(i);

               fmov[i] =
elementoMovimiento.getMovimiento(anteriores,suma_movimientos);
               suma_movimientos =
elementoMovimiento.getMovimiento(anteriores,suma_movimientos);
               anteriores.add(elementoMovimiento);

               //Totales
               total_mov = total_mov + fmov[i];

               //Maximos
               if(fmov[i]>max_mov){
                    max_mov = fmov[i];
```

```
                }
                //Minimos
                if(fmov[i]<min_mov){
                      min_mov = fmov[i];
                }

          }

          values.add(fmov);

          //Medias
          double media_mov = total_mov/(double)elementos.size();

          String mensaje = "";

          mensaje =  "\nMax: "+Tools.truncar(""+max_mov)+"\n"+
                         "Min: "+Tools.truncar(""+min_mov)+"\n"+
                         "Media: "+Tools.truncar(""+media_mov);

          Toast.makeText(MyApp.getContext(), mensaje,
Toast.LENGTH_LONG).show();

          int[] colors = new int[] { Color.BLUE};

          PointStyle[] styles = new PointStyle[] { PointStyle.CIRCLE };

          XYMultipleSeriesRenderer renderer = buildRenderer(colors,
styles);

          int length = renderer.getSeriesRendererCount();
          for (int i = 0; i < length; i++) {
               ((XYSeriesRenderer)
renderer.getSeriesRendererAt(i)).setFillPoints(true);
          }

          double max_max = max_mov+5d;
          setChartSettings(renderer, mensaje, "Tiempo","R", 0,
elementos.size(), -12, max_max, Color.BLACK, Color.BLACK);
          renderer.setXLabels(12);
          renderer.setYLabels(10);
          renderer.setShowGrid(true);
          renderer.setXLabelsAlign(Align.RIGHT);
          renderer.setYLabelsAlign(Align.RIGHT);
          renderer.setZoomButtonsVisible(true);
          renderer.setPanLimits(new double[] { -10, 20, -10, 40 });
          renderer.setZoomLimits(new double[] { -10, 20, -10, 40 });


          XYMultipleSeriesDataset dataset = buildDataset(titles, x,
values);
          XYSeries series = dataset.getSeriesAt(0);
          //series.addAnnotation("Vacation", 6, 30);
          Intent intent = ChartFactory.getLineChartIntent(context,
dataset,renderer, "Movimiento");
          return intent;
     }
```

```
}
```

[Cod. 33]: Código para la clase gráficas : AbstractDemoChart

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.Date;
import java.util.List;

import org.achartengine.chart.PointStyle;
import org.achartengine.model.CategorySeries;
import org.achartengine.model.MultipleCategorySeries;
import org.achartengine.model.TimeSeries;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.DefaultRenderer;
import org.achartengine.renderer.SimpleSeriesRenderer;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

/**
 * An abstract class for the demo charts to extend. It contains some
methods for
 * building datasets and renderers.
 */
public abstract class AbstractDemoChart implements IDemoChart {

  /**
   * Builds an XY multiple dataset using the provided values.
   *
   * @param titles the series titles
   * @param xValues the values for the X axis
   * @param yValues the values for the Y axis
   * @return the XY multiple dataset
   */
  protected XYMultipleSeriesDataset buildDataset(String[] titles,
List<double[]> xValues,
      List<double[]> yValues) {
    XYMultipleSeriesDataset dataset = new XYMultipleSeriesDataset();
```

```
    addXYSeries(dataset, titles, xValues, yValues, 0);
    return dataset;
  }

  public void addXYSeries(XYMultipleSeriesDataset dataset, String[]
titles, List<double[]> xValues,
      List<double[]> yValues, int scale) {
    int length = titles.length;
    for (int i = 0; i < length; i++) {
      XYSeries series = new XYSeries(titles[i], scale);
      double[] xV = xValues.get(i);
      double[] yV = yValues.get(i);
      int seriesLength = xV.length;
      for (int k = 0; k < seriesLength; k++) {
        series.add(xV[k], yV[k]);
      }
      dataset.addSeries(series);
    }
  }

  /**
   * Builds an XY multiple series renderer.
   *
   * @param colors the series rendering colors
   * @param styles the series point styles
   * @return the XY multiple series renderers
   */
  protected XYMultipleSeriesRenderer buildRenderer(int[] colors,
PointStyle[] styles) {
    XYMultipleSeriesRenderer renderer = new XYMultipleSeriesRenderer();
    setRenderer(renderer, colors, styles);
    return renderer;
  }

  protected void setRenderer(XYMultipleSeriesRenderer renderer, int[]
colors, PointStyle[] styles) {
    renderer.setAxisTitleTextSize(16);
    renderer.setChartTitleTextSize(20);
    renderer.setLabelsTextSize(15);
    renderer.setLegendTextSize(15);
    renderer.setPointSize(5f);
    renderer.setMargins(new int[] { 20, 30, 15, 20 });
    int length = colors.length;
    for (int i = 0; i < length; i++) {
      XYSeriesRenderer r = new XYSeriesRenderer();
      r.setColor(colors[i]);
      r.setPointStyle(styles[i]);
      renderer.addSeriesRenderer(r);
    }
  }

  /**
   * Sets a few of the series renderer settings.
   *
   * @param renderer the renderer to set the properties to
   * @param title the chart title
   * @param xTitle the title for the X axis
```

```
   * @param yTitle the title for the Y axis
   * @param xMin the minimum value on the X axis
   * @param xMax the maximum value on the X axis
   * @param yMin the minimum value on the Y axis
   * @param yMax the maximum value on the Y axis
   * @param axesColor the axes color
   * @param labelsColor the labels color
   */
  protected void setChartSettings(XYMultipleSeriesRenderer renderer,
String title, String xTitle,
      String yTitle, double xMin, double xMax, double yMin, double yMax,
int axesColor,
      int labelsColor) {
    renderer.setChartTitle(title);
    renderer.setXTitle(xTitle);
    renderer.setYTitle(yTitle);
    renderer.setXAxisMin(xMin);
    renderer.setXAxisMax(xMax);
    renderer.setYAxisMin(yMin);
    renderer.setYAxisMax(yMax);
    renderer.setAxesColor(axesColor);
    renderer.setLabelsColor(labelsColor);
  }

  /**
   * Builds an XY multiple time dataset using the provided values.
   *
   * @param titles the series titles
   * @param xValues the values for the X axis
   * @param yValues the values for the Y axis
   * @return the XY multiple time dataset
   */
  protected XYMultipleSeriesDataset buildDateDataset(String[] titles,
List<Date[]> xValues,
      List<double[]> yValues) {
    XYMultipleSeriesDataset dataset = new XYMultipleSeriesDataset();
    int length = titles.length;
    for (int i = 0; i < length; i++) {
      TimeSeries series = new TimeSeries(titles[i]);
      Date[] xV = xValues.get(i);
      double[] yV = yValues.get(i);
      int seriesLength = xV.length;
      for (int k = 0; k < seriesLength; k++) {
        series.add(xV[k], yV[k]);
      }
      dataset.addSeries(series);
    }
    return dataset;
  }

  /**
   * Builds a category series using the provided values.
   *
   * @param titles the series titles
   * @param values the values
   * @return the category series
   */
```

```
  protected CategorySeries buildCategoryDataset(String title, double[]
values) {
    CategorySeries series = new CategorySeries(title);
    int k = 0;
    for (double value : values) {
      series.add("Project " + ++k, value);
    }

    return series;
  }

  /**
   * Builds a multiple category series using the provided values.
   *
   * @param titles the series titles
   * @param values the values
   * @return the category series
   */
  protected MultipleCategorySeries buildMultipleCategoryDataset(String
title,
      List<String[]> titles, List<double[]> values) {
    MultipleCategorySeries series = new MultipleCategorySeries(title);
    int k = 0;
    for (double[] value : values) {
      series.add(2007 + k + "", titles.get(k), value);
      k++;
    }
    return series;
  }

  /**
   * Builds a category renderer to use the provided colors.
   *
   * @param colors the colors
   * @return the category renderer
   */
  protected DefaultRenderer buildCategoryRenderer(int[] colors) {
    DefaultRenderer renderer = new DefaultRenderer();
    renderer.setLabelsTextSize(15);
    renderer.setLegendTextSize(15);
    renderer.setMargins(new int[] { 20, 30, 15, 0 });
    for (int color : colors) {
      SimpleSeriesRenderer r = new SimpleSeriesRenderer();
      r.setColor(color);
      renderer.addSeriesRenderer(r);
    }
    return renderer;
  }

  /**
   * Builds a bar multiple series dataset using the provided values.
   *
   * @param titles the series titles
   * @param values the values
   * @return the XY multiple bar dataset
   */
  protected XYMultipleSeriesDataset buildBarDataset(String[] titles,
```

```
List<double[]> values) {
    XYMultipleSeriesDataset dataset = new XYMultipleSeriesDataset();
    int length = titles.length;
    for (int i = 0; i < length; i++) {
      CategorySeries series = new CategorySeries(titles[i]);
      double[] v = values.get(i);
      int seriesLength = v.length;
      for (int k = 0; k < seriesLength; k++) {
        series.add(v[k]);
      }
      dataset.addSeries(series.toXYSeries());
    }
    return dataset;
  }

  /**
   * Builds a bar multiple series renderer to use the provided colors.
   *
   * @param colors the series renderers colors
   * @return the bar multiple series renderer
   */
  protected XYMultipleSeriesRenderer buildBarRenderer(int[] colors) {
    XYMultipleSeriesRenderer renderer = new XYMultipleSeriesRenderer();
    renderer.setAxisTitleTextSize(16);
    renderer.setChartTitleTextSize(20);
    renderer.setLabelsTextSize(15);
    renderer.setLegendTextSize(15);
    int length = colors.length;
    for (int i = 0; i < length; i++) {
      SimpleSeriesRenderer r = new SimpleSeriesRenderer();
      r.setColor(colors[i]);
      renderer.addSeriesRenderer(r);
    }
    return renderer;
  }

}
```

[Cod. 34]: Código para la clase gráficas : IDemoChart

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
```

```
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import android.content.Context;
import android.content.Intent;

/**
 * Defines the demo charts.
 */
public interface IDemoChart {
  /** A constant for the name field in a list activity. */
  String NAME = "name";
  /** A constant for the description field in a list activity. */
  String DESC = "desc";

  /**
   * Returns the chart name.
   *
   * @return the chart name
   */
  String getName();

  /**
   * Returns the chart description.
   *
   * @return the chart description
   */
  String getDesc();

  /**
   * Executes the chart demo.
   *
   * @param context the context
   * @return the built intent
   */
  Intent execute(Context context, int id);
}
```

[Cod. 34]: Código para la clase gráficas : XYChartBuilder

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
```

```
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.List;

import org.achartengine.ChartFactory;
import org.achartengine.GraphicalView;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.SeriesSelection;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.example.proyectofc.R;
import com.example.proyectofc.dao.TablaMovimientoDAO;
import com.example.proyectofc.modelo.ElementoMovimiento;

public class XYChartBuilder extends Activity {
      /** The main dataset that includes all the series that go into a
chart. */
      private XYMultipleSeriesDataset mDataset = new
XYMultipleSeriesDataset();
      /** The main renderer that includes all the renderers customizing a
chart. */
      private XYMultipleSeriesRenderer mRenderer = new
XYMultipleSeriesRenderer();
      /** The most recently added series. */
      private XYSeries mCurrentSeries;
      /** The most recently created renderer, customizing the current
series. */
      private XYSeriesRenderer mCurrentRenderer;

      /** The chart view that displays the data. */
      private GraphicalView mChartView;

      private int id;

      @Override
      protected void onSaveInstanceState(Bundle outState) {
            super.onSaveInstanceState(outState);
            // save the current data, for instance when changing screen
orientation
            outState.putSerializable("dataset", mDataset);
            outState.putSerializable("renderer", mRenderer);
```

```
            outState.putSerializable("current_series", mCurrentSeries);
            outState.putSerializable("current_renderer", mCurrentRenderer);
      }

      @Override
      protected void onRestoreInstanceState(Bundle savedState) {
            super.onRestoreInstanceState(savedState);
            // restore the current data, for instance when changing the
screen
            // orientation
            mDataset = (XYMultipleSeriesDataset) savedState
                        .getSerializable("dataset");
            mRenderer = (XYMultipleSeriesRenderer) savedState
                        .getSerializable("renderer");
            mCurrentSeries = (XYSeries) savedState
                        .getSerializable("current_series");
            mCurrentRenderer = (XYSeriesRenderer) savedState
                        .getSerializable("current_renderer");
      }

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

            // set some properties on the main renderer
            mRenderer.setApplyBackgroundColor(true);
            mRenderer.setBackgroundColor(Color.argb(100, 50, 50, 50));
            mRenderer.setAxisTitleTextSize(16);
            mRenderer.setChartTitleTextSize(20);
            mRenderer.setLabelsTextSize(15);
            mRenderer.setLegendTextSize(15);
            mRenderer.setMargins(new int[] { 20, 30, 15, 0 });
            mRenderer.setZoomButtonsVisible(true);
            mRenderer.setPointSize(5);

            this.id =
Integer.parseInt(getIntent().getStringExtra("id_principal"));

      }

      private void addNewSeries() {
            String seriesTitle = "Series " + (mDataset.getSeriesCount() +
1);
            // create a new series of data
            XYSeries series = new XYSeries(seriesTitle);
            mDataset.addSeries(series);
            mCurrentSeries = series;
            // create a new renderer for the new series
            XYSeriesRenderer renderer = new XYSeriesRenderer();
            mRenderer.addSeriesRenderer(renderer);
            // set some renderer properties
            renderer.setPointStyle(PointStyle.CIRCLE);
            renderer.setFillPoints(true);
            renderer.setDisplayChartValues(true);
            renderer.setDisplayChartValuesDistance(10);
            mCurrentRenderer = renderer;
```

```
            // mChartView.repaint();
      }

      /**
       * Add data to current series
       *
       * @param x
       * @param y
       */
      private void addNewPoint(double x, double y) {
            mCurrentSeries.add(x, y);
            // mChartView.repaint();
      }

      @Override
      protected void onResume() {
            super.onResume();
            if (mChartView == null) {
                  LinearLayout layout = (LinearLayout)
findViewById(R.id.chart);
                  mChartView = ChartFactory.getLineChartView(this,
mDataset,
                              mRenderer);
                  // enable the chart click events
                  mRenderer.setClickEnabled(true);
                  mRenderer.setSelectableBuffer(10);
                  mChartView.setOnClickListener(new View.OnClickListener()
{
                        public void onClick(View v) {
                              // handle the click event on the chart
                              SeriesSelection seriesSelection = mChartView
                                          .getCurrentSeriesAndPoint();
                              if (seriesSelection == null) {
                                    Toast.makeText(XYChartBuilder.this, "No
chart element",
                                                Toast.LENGTH_SHORT).show();
                              } else {
                                    // display information of the clicked
point
                                    Toast.makeText(
                                                XYChartBuilder.this,
                                                "Chart element in series index
"
                                                            +
seriesSelection.getSeriesIndex()
                                                            + " data point
index "
                                                            +
seriesSelection.getPointIndex()
                                                            + " was clicked"
                                                            + " closest point
value X="
                                                            +
seriesSelection.getXValue() + ", Y="
                                                            +
seriesSelection.getValue(),
                                                Toast.LENGTH_SHORT).show();
```

```
                              }
                         }
                    });
                    layout.addView(mChartView, new LayoutParams(
                              LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));

                    init();
               } else {
                    mChartView.repaint();
               }

     }

     private void init() {

          TablaMovimientoDAO dao = new TablaMovimientoDAO();
          List<ElementoMovimiento> elementos = dao.getListas(this.id);

          //f(t)=x
          addNewSeries();
          for (int i=0; i<elementos.size(); i++) {
              ElementoMovimiento elementoMovimiento = elementos.get(i);
               addNewPoint(i,elementoMovimiento.getX());
          }

          //f(t)=y
          addNewSeries();
          for (int i=0; i<elementos.size(); i++) {
              ElementoMovimiento elementoMovimiento = elementos.get(i);
               addNewPoint(i,elementoMovimiento.getY());
          }

          //f(t)=z
          addNewSeries();
          for (int i=0; i<elementos.size(); i++) {
              ElementoMovimiento elementoMovimiento = elementos.get(i);
               addNewPoint(i,elementoMovimiento.getZ());
          }

     }

}
```

[Cod. 35]: Código para la clase gráficas : luz

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
```

```
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.ArrayList;
import java.util.List;

import org.achartengine.ChartFactory;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint.Align;
import android.widget.Toast;

import com.example.proyectofc.MyApp;
import com.example.proyectofc.dao.TablaLuzDAO;
import com.example.proyectofc.modelo.ElementoLuminosidad;
import com.example.proyectofc.tools.Tools;

public class LuzChart extends AbstractDemoChart {
      /**
       * Returns the chart name.
       *
       * @return the chart name
       */
      public String getName() {
            return "Luminosidad";
      }

      /**
       * Returns the chart description.
       *
       * @return the chart description
       */
      public String getDesc() {
            return "Luminosidad";
      }

      /**
       * Executes the chart demo.
       *
       * @param context
       *            the context
       * @return the built intent
       */
      public Intent execute(Context context, int id) {

            TablaLuzDAO dao = new TablaLuzDAO();
```

```
            List<ElementoLuminosidad> elementos = dao.getListas(id);

            String[] titles = new String[] { "Luminosidad(Lux)" };

            //dominio(tiempo)
            List<double[]> x = new ArrayList<double[]>();
            for (int i = 0; i < titles.length; i++) {
                 double[] dominio = new double[elementos.size()];
                //x.add(new double[] { 1, 2, 3, 4, 5, 6, 7,7.5, 8, 9, 10,
11 });
                 for (int j = 0; j < elementos.size(); j++) {
                      dominio[j]=j;
                 }
                 x.add(dominio);
            }



            List<double[]> values = new ArrayList<double[]>();

            //f(t)=x
            double[] fx = new double[elementos.size()];


            double max_x = Double.MIN_VALUE;
            double min_x = Double.MAX_VALUE;
            double total_x = 0;


            for (int i=0; i<elementos.size(); i++) {
                 ElementoLuminosidad elementoMovimiento =
elementos.get(i);
                 fx[i] = elementoMovimiento.getLuminosidad();

                 //Totales
                 total_x = total_x + fx[i];

                 //Maximos
                 if(fx[i]>max_x){
                      max_x = fx[i];
                 }

                 //Minimos
                 if(fx[i]<min_x){
                      min_x = fx[i];
                 }


            }
            values.add(fx);


            //Medias
            double media_x = total_x/(double)elementos.size();

            String mensaje = "";
```

```
            mensaje =  "\n\n\nLuminosidad mÃ¡xima:
"+Tools.truncar(""+max_x)+"\n"+
                       "Luminosidad mÃnima:
"+Tools.truncar(""+min_x)+"\n"+
                       "Luminosidad media:
"+Tools.truncar(""+media_x)+"\n";


            Toast.makeText(MyApp.getContext(), mensaje,
Toast.LENGTH_LONG).show();

            int[] colors = new int[] {Color.RED };

            PointStyle[] styles = new PointStyle[] { PointStyle.CIRCLE};

            XYMultipleSeriesRenderer renderer = buildRenderer(colors,
styles);

            int length = renderer.getSeriesRendererCount();
            for (int i = 0; i < length; i++) {
                 ((XYSeriesRenderer)
renderer.getSeriesRendererAt(i)).setFillPoints(true);
            }

            double max_max = max_x+50d;
            //setChartSettings(renderer,
"SensoresA\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal",
"Tiempo","R", 0, elementos.size(), -12, 12, Color.LTGRAY, Color.LTGRAY);
            setChartSettings(renderer, mensaje, "Tiempo","R", 0,
elementos.size(), -12, max_max, Color.BLACK, Color.BLACK);
            renderer.setXLabels(12);
            renderer.setYLabels(10);
            renderer.setShowGrid(true);
            renderer.setXLabelsAlign(Align.RIGHT);
            renderer.setYLabelsAlign(Align.RIGHT);
            renderer.setZoomButtonsVisible(true);
            renderer.setPanLimits(new double[] { -10, 20, -10, 40 });
            renderer.setZoomLimits(new double[] { -10, 20, -10, 40 });
            XYMultipleSeriesDataset dataset = buildDataset(titles, x,
values);
            XYSeries series = dataset.getSeriesAt(0);
            //series.addAnnotation("Vacation", 6, 30);
            Intent intent = ChartFactory.getLineChartIntent(context,
dataset,renderer, "SensoresB");
            return intent;
      }



}
```

[Cod. 36]: Código para la clase gráficas : campo magnético

```
/**
 * Copyright (C) 2009 - 2013 SC 4ViewSoft SRL
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
```

```
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.example.proyectofc.graficas;

import java.util.ArrayList;
import java.util.List;

import org.achartengine.ChartFactory;
import org.achartengine.chart.PointStyle;
import org.achartengine.model.XYMultipleSeriesDataset;
import org.achartengine.model.XYSeries;
import org.achartengine.renderer.XYMultipleSeriesRenderer;
import org.achartengine.renderer.XYSeriesRenderer;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.Paint.Align;
import android.widget.Toast;

import com.example.proyectofc.MyApp;
import com.example.proyectofc.dao.TablaLuzDAO;
import com.example.proyectofc.dao.TablaMagneticoDAO;
import com.example.proyectofc.modelo.ElementoCampoMagnetico;
import com.example.proyectofc.modelo.ElementoLuminosidad;
import com.example.proyectofc.tools.Tools;

/**
 * Average temperature demo chart.
 */
public class CampoMagneticoChart extends AbstractDemoChart {
     /**
      * Returns the chart name.
      *
      * @return the chart name
      */
     public String getName() {
          return "Average temperature";
     }

     /**
      * Returns the chart description.
      *
      * @return the chart description
      */
     public String getDesc() {
          return "The average temperature in 4 Greek islands (line
```

```
chart)";
      }

      /**
       * Executes the chart demo.
       *
       * @param context
       *            the context
       * @return the built intent
       */
      public Intent execute(Context context, int id) {

            TablaMagneticoDAO dao = new TablaMagneticoDAO();
            List<ElementoCampoMagnetico> elementos = dao.getListas(id);

            String[] titles = new String[] { "Campo magnetico(uT)" };

            //dominio(tiempo)
            List<double[]> x = new ArrayList<double[]>();
            for (int i = 0; i < titles.length; i++) {
                  double[] dominio = new double[elementos.size()];
                 //x.add(new double[] { 1, 2, 3, 4, 5, 6, 7,7.5, 8, 9, 10,
11 });
                  for (int j = 0; j < elementos.size(); j++) {
                        dominio[j]=j;
                  }
                  x.add(dominio);
            }



            List<double[]> values = new ArrayList<double[]>();

            //f(t)=x
            double[] fx = new double[elementos.size()];


            double max_x = Double.MIN_VALUE;
            double min_x = Double.MAX_VALUE;
            double total_x = 0;


            for (int i=0; i<elementos.size(); i++) {
                  ElementoCampoMagnetico elementoMovimiento =
elementos.get(i);
                  fx[i] = elementoMovimiento.getMagnetismo();

                  //Totales
                  total_x = total_x + fx[i];

                  //Maximos
                  if(fx[i]>max_x){
                        max_x = fx[i];
                  }

                  //Minimos
                  if(fx[i]<min_x){
```

```
                    min_x = fx[i];
               }


          }
          values.add(fx);


          //Medias
          double media_x = total_x/(double)elementos.size();

          String mensaje = "";

          mensaje =  "\n\n\nCampo magnÃ©tico mÃ¡ximo:
"+Tools.truncar(""+max_x)+"\n"+
                    "Campo magnÃ©tico mÃnimo:
"+Tools.truncar(""+min_x)+"\n"+
                    "Campo magnÃ©tico medio:
"+Tools.truncar(""+media_x)+"\n";


          Toast.makeText(MyApp.getContext(), mensaje,
Toast.LENGTH_LONG).show();

          int[] colors = new int[] {Color.RED };

          PointStyle[] styles = new PointStyle[] { PointStyle.CIRCLE};

          XYMultipleSeriesRenderer renderer = buildRenderer(colors,
styles);

          int length = renderer.getSeriesRendererCount();
          for (int i = 0; i < length; i++) {
               ((XYSeriesRenderer)
renderer.getSeriesRendererAt(i)).setFillPoints(true);
          }

          double max_max = max_x+50d;
          double min_min = min_x-50d;
          //setChartSettings(renderer,
"SensoresA\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal\nlalal",
"Tiempo","R", 0, elementos.size(), -12, 12, Color.LTGRAY, Color.LTGRAY);
          setChartSettings(renderer, mensaje, "Tiempo","R", 0,
elementos.size(), min_min, max_max, Color.BLACK, Color.BLACK);
          renderer.setXLabels(12);
          renderer.setYLabels(10);
          renderer.setShowGrid(true);
          renderer.setXLabelsAlign(Align.RIGHT);
          renderer.setYLabelsAlign(Align.RIGHT);
          renderer.setZoomButtonsVisible(true);
          renderer.setPanLimits(new double[] { -10, 20, -10, 40 });
          renderer.setZoomLimits(new double[] { -10, 20, -10, 40 });
//        renderer.setPanLimits(new double[] { 0, 20, -12, 12 });
//        renderer.setZoomLimits(new double[] { 0, 20, -12, 12 });

          XYMultipleSeriesDataset dataset = buildDataset(titles, x,
values);
```

```
            XYSeries series = dataset.getSeriesAt(0);
            //series.addAnnotation("Vacation", 6, 30);
            Intent intent = ChartFactory.getLineChartIntent(context,
dataset,renderer, "SensoresB");
            return intent;
      }

      }
```

[Cod. 37]: Código main Activity : lista principal movimiento

```
package com.example.proyectofc;

import java.util.List;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;

import com.example.proyectofc.adapter.AdaptadorMovimientoPrincipal;
import com.example.proyectofc.dao.TablaPrincipalDAO;
import com.example.proyectofc.graficas.AcelerometroChart;
import com.example.proyectofc.graficas.MovimientosChart;
import com.example.proyectofc.modelo.ElementoPrincipal;

public class ListasPrincipalActivity extends Activity {


      ListView lista;
      String modo;


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

            this.modo = getIntent().getStringExtra("MODO");

            lista = (ListView) findViewById(R.id.lista_principal);

            lista.setOnItemClickListener(new OnItemClickListener() {

                  //id, es la id de la fila seleccionada de la tabla
principal
                  @Override
                  public void onItemClick(AdapterView<?> parent, View
view,int position, final long id) {
                        String modo_ejecucion =
ListasPrincipalActivity.this.modo;
```

```
                        if(modo_ejecucion.equals("Lista")){
                              Intent intent = new
Intent(ListasPrincipalActivity.this, MovimientosActivity.class);
                              intent.putExtra("id_principal", ""+id);
                              startActivity(intent);
                        }
                        else{


                              DialogInterface.OnClickListener
dialogClickListener = new DialogInterface.OnClickListener() {
                              public void onClick(DialogInterface dialog,
int which) {
                                    switch (which) {
                                    case DialogInterface.BUTTON_POSITIVE:
                                       Intent intent = new
AcelerometroChart().execute(ListasPrincipalActivity.this,(int)id);
                                       startActivity(intent);
                                       break;

                                    case DialogInterface.BUTTON_NEGATIVE:
                                       Intent intent2 = new
MovimientosChart().execute(ListasPrincipalActivity.this,(int)id);
                                       startActivity(intent2);
                                        break;
                                    }
                              }
                        };

                        AlertDialog.Builder builder = new
AlertDialog.Builder(ListasPrincipalActivity.this);
                        builder.setMessage("Â¿Que deseas consultar?")
                                    .setPositiveButton("Coordenadas x,y,z",
dialogClickListener)
                                    .setNegativeButton("Movimiento",
dialogClickListener).show();


                        }


                  }
            });

      }
      /**
       * Aqui se carga los elementos de lista principal
       */
      @Override
     public void onResume(){

       TablaPrincipalDAO dao = new TablaPrincipalDAO();
       List<ElementoPrincipal> lista_principal = dao.getListas();

       AdaptadorMovimientoPrincipal adapter = null;
```

```
     if(lista.getAdapter()==null){
          adapter = new AdaptadorMovimientoPrincipal(this,
lista_principal);
          lista.setAdapter(adapter);
     }
     else{
          adapter = (AdaptadorMovimientoPrincipal) lista.getAdapter();
     }

     adapter.clear();
     adapter.addAll(lista_principal);
     adapter.notifyDataSetChanged();
     super.onResume();
    }


}
```

[Cod. 38]: Código main Activity: lista movimiento

```
package com.example.proyectofc;

import java.util.List;

import com.example.proyectofc.adapter.AdaptadorMovimiento;
import com.example.proyectofc.dao.TablaMovimientoDAO;
import com.example.proyectofc.modelo.ElementoMovimiento;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class MovimientosActivity extends Activity{


     private ListView lista;
     private int id_principal;

     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_listasxyz);

          lista = (ListView) findViewById(R.id.lista_xyz);
          id_principal =
Integer.parseInt(getIntent().getStringExtra("id_principal"));
     }

     @Override
    public void onResume(){

          //Recuperamos los datos de la BBDD
          TablaMovimientoDAO dao = new TablaMovimientoDAO();
     List<ElementoMovimiento> lista_principal =
dao.getListas(id_principal);
```

```
	AdaptadorMovimiento adapter = null;

	if(lista.getAdapter()==null){
		adapter = new AdaptadorMovimiento(this, lista_principal);
		lista.setAdapter(adapter);
	}
	else{
		adapter = (AdaptadorMovimiento) lista.getAdapter();
	}

	adapter.clear();
	adapter.addAll(lista_principal);
	adapter.notifyDataSetChanged();
		super.onResume();
	}
}
```

[Cod. 39]: Código main Activity: principal luz

```
package com.example.proyectofc;

import java.util.List;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.Toast;

import com.example.proyectofc.adapter.AdaptadorLuzPrincipal;
import com.example.proyectofc.dao.TablaLuzDAO;
import com.example.proyectofc.dao.TablaLuzPrincipalDAO;
import com.example.proyectofc.graficas.LuzChart;
import com.example.proyectofc.modelo.ElementoLuminosidad;
import com.example.proyectofc.modelo.ElementoLuzPrincipal;

public class ListasLuzPActivity extends Activity {

	ListView lista;
	String modo;

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

		this.modo = getIntent().getStringExtra("MODO");

		lista = (ListView) findViewById(R.id.lista_principal);

		lista.setOnItemClickListener(new OnItemClickListener() {
```

```
                //id, es la id de la fila seleccionada de la tabla
principal
                @Override
                public void onItemClick(AdapterView<?> parent, View
view,int position, long id) {

                     String modo_ejecucion =
ListasLuzPActivity.this.modo;

                     if(modo_ejecucion.equals("Lista")){
                          Intent intent = new
Intent(ListasLuzPActivity.this, LucesActivity.class);
                          intent.putExtra("id_principal", ""+id);
                          startActivity(intent);
                     }
                     else{

                          TablaLuzDAO dao = new TablaLuzDAO();
                          List<ElementoLuminosidad> elementos =
dao.getListas((int)id);

                          if(elementos.size()>0){
                               Intent intent = new
LuzChart().execute(ListasLuzPActivity.this,(int)id);
                               startActivity(intent);
                          }
                          else{
                               Toast.makeText(ListasLuzPActivity.this,
"No hay entradas.", Toast.LENGTH_SHORT).show();
                          }

                     }

                }
           });

     }
     /**
      * Aqui se carga los elementos de lista principal
      */
     @Override
    public void onResume(){

     TablaLuzPrincipalDAO dao = new TablaLuzPrincipalDAO();
     List<ElementoLuzPrincipal> lista_principal = dao.getListas();

     AdaptadorLuzPrincipal adapter = null;

     if(lista.getAdapter()==null){
          adapter = new AdaptadorLuzPrincipal(this, lista_principal);
          lista.setAdapter(adapter);
     }
     else{
          adapter = (AdaptadorLuzPrincipal) lista.getAdapter();
     }

     adapter.clear();
```

```
      adapter.addAll(lista_principal);
      adapter.notifyDataSetChanged();
      super.onResume();
     }


}
```

[Cod. 40]: Código main Activity: luz

```
package com.example.proyectofc;

import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

import com.example.proyectofc.adapter.AdaptadorLuz;
import com.example.proyectofc.dao.TablaLuzDAO;
import com.example.proyectofc.modelo.ElementoLuminosidad;

public class LucesActivity extends Activity{


      private ListView lista;
      private int id_principal;

      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_listasxyz);

            lista = (ListView) findViewById(R.id.lista_xyz);
            id_principal =
Integer.parseInt(getIntent().getStringExtra("id_principal"));
      }

      @Override
     public void onResume(){

            //Recuperamos los datos de la BBDD
            TablaLuzDAO dao = new TablaLuzDAO();
      List<ElementoLuminosidad> lista_principal =
dao.getListas(id_principal);

      AdaptadorLuz adapter = null;

      if(lista.getAdapter()==null){
            adapter = new AdaptadorLuz(this, lista_principal);
            lista.setAdapter(adapter);
      }
      else{
            adapter = (AdaptadorLuz) lista.getAdapter();
      }

      adapter.clear();
      adapter.addAll(lista_principal);
```

```
	adapter.notifyDataSetChanged();
		super.onResume();
	}
}
```

[Cod. 41]: Código main Activity: principal magnetismo

```
package com.example.proyectofc;

import java.util.List;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.Toast;

import com.example.proyectofc.adapter.AdaptadorMagneticoPrincipal;
import com.example.proyectofc.dao.TablaMagneticoDAO;
import com.example.proyectofc.dao.TablaMagneticoPrincipalDAO;
import com.example.proyectofc.graficas.CampoMagneticoChart;
import com.example.proyectofc.modelo.ElementoCampoMagnetico;
import com.example.proyectofc.modelo.ElementoCmagneticoPrincipal;

public class ListasMagnetismoPActivity extends Activity {

	ListView lista;
	String modo;

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

		this.modo = getIntent().getStringExtra("MODO");

		lista = (ListView) findViewById(R.id.lista_principal);

		lista.setOnItemClickListener(new OnItemClickListener() {

			//id, es la id de la fila seleccionada de la tabla
principal
			@Override
			public void onItemClick(AdapterView<?> parent, View
view,int position, long id) {

				String modo_ejecucion =
ListasMagnetismoPActivity.this.modo;

				if(modo_ejecucion.equals("Lista")){
					Intent intent = new
Intent(ListasMagnetismoPActivity.this, MagnetismosActivity.class);
```

```
                              intent.putExtra("id_principal", ""+id);
                              startActivity(intent);
                         }
                         else{
                              TablaMagneticoDAO dao = new
TablaMagneticoDAO();
                              List<ElementoCampoMagnetico> elementos =
dao.getListas((int)id);

                              if(elementos.size()>0){

                                   Intent intent = new
CampoMagneticoChart().execute(ListasMagnetismoPActivity.this,(int)id);
                                   startActivity(intent);
                              }
                              else{

     Toast.makeText(ListasMagnetismoPActivity.this, "No hay entradas.",
Toast.LENGTH_SHORT).show();
                              }
                         }

                    }
               });

     }
     /**
      * Aqui se carga los elementos de lista principal
      */
     @Override
    public void onResume(){

          TablaMagneticoPrincipalDAO dao = new
TablaMagneticoPrincipalDAO();
     List<ElementoCmagneticoPrincipal> lista_principal = dao.getListas();

     AdaptadorMagneticoPrincipal adapter = null;

     if(lista.getAdapter()==null){
          adapter = new AdaptadorMagneticoPrincipal(this,
lista_principal);
          lista.setAdapter(adapter);
     }
     else{
          adapter = (AdaptadorMagneticoPrincipal) lista.getAdapter();
     }

     adapter.clear();
     adapter.addAll(lista_principal);
     adapter.notifyDataSetChanged();
     super.onResume();
    }}
```

[Cod. 42]: Código main Activity: magnetismo

```
package com.example.proyectofc;

import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

import com.example.proyectofc.adapter.AdaptadorMagnetico;
import com.example.proyectofc.dao.TablaMagneticoDAO;
import com.example.proyectofc.modelo.ElementoCampoMagnetico;

public class MagnetismosActivity extends Activity{


      private ListView lista;
      private int id_principal;

      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_listasxyz);

            lista = (ListView) findViewById(R.id.lista_xyz);
            id_principal =
Integer.parseInt(getIntent().getStringExtra("id_principal"));
      }

      @Override
     public void onResume(){

            //Recuperamos los datos de la BBDD
            TablaMagneticoDAO dao = new TablaMagneticoDAO();
      List<ElementoCampoMagnetico> lista_principal =
dao.getListas(id_principal);

      AdaptadorMagnetico adapter = null;

      if(lista.getAdapter()==null){
            adapter = new AdaptadorMagnetico(this, lista_principal);
            lista.setAdapter(adapter);
      }
      else{
            adapter = (AdaptadorMagnetico) lista.getAdapter();
      }

      adapter.clear();
      adapter.addAll(lista_principal);
      adapter.notifyDataSetChanged();
            super.onResume();
      }}
```

[Cod. 43]: Código main Activity Principal:

```
package com.example.proyectofc;

import java.text.SimpleDateFormat;
import java.util.Calendar;
```

```
import java.util.List;

import android.app.Activity;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

import com.example.proyectofc.dao.TablaLuzDAO;
import com.example.proyectofc.dao.TablaLuzPrincipalDAO;
import com.example.proyectofc.dao.TablaMagneticoDAO;
import com.example.proyectofc.dao.TablaMagneticoPrincipalDAO;
import com.example.proyectofc.dao.TablaMovimientoDAO;
import com.example.proyectofc.dao.TablaPrincipalDAO;
import com.example.proyectofc.modelo.ElementoCampoMagnetico;
import com.example.proyectofc.modelo.ElementoLuminosidad;
import com.example.proyectofc.modelo.ElementoMovimiento;

public class MainActivity extends Activity implements SensorEventListener
{

     private TextView tv_x,tv_y,tv_z;
     private TextView tv_luz;
     private TextView tv_magnetic_field;
     private Button bt_empezar, bt_finalizar, bt_listas,
bt_graficas,bt_graficas2,bt_graficas3;
     private Button bt_empezar2, bt_finalizar2, bt_listas2;
     private Button bt_empezar3, bt_finalizar3, bt_listas3;
     private boolean grabando1;
     private boolean grabando2;
     private boolean grabando3;

     private long tiempo;
     private long tiempo_anterior;
     private long id_principal1,id_principal2,id_principal3;

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

          grabando1 = grabando2= grabando3 = false;

          SensorManager sensorManager =
(SensorManager)getSystemService(SENSOR_SERVICE);

          acelerometro(sensorManager);
          luz(sensorManager);
          magneticField(sensorManager);
```

```
            Button bt_empezar_general =
(Button)findViewById(R.id.btEmpezar_general);
            bt_empezar_general.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        grabando1 = grabando2 = grabando3 = true;

                        tiempo = Calendar.getInstance().getTimeInMillis();
                        tiempo_anterior = tiempo;

                        TablaPrincipalDAO dao = new TablaPrincipalDAO();
                        id_principal1 = dao.insert();

                        TablaLuzPrincipalDAO dao2 = new
TablaLuzPrincipalDAO();
                        id_principal2 = dao2.insert();

                        TablaMagneticoPrincipalDAO dao3 = new
TablaMagneticoPrincipalDAO();
                        id_principal3 = dao3.insert();

                  }
            });

            Button bt_finalizar_general =
(Button)findViewById(R.id.btFinalizar_general);
            bt_finalizar_general.setOnClickListener(new OnClickListener()
{

                  @Override
                  public void onClick(View v) {

                        grabando1 = grabando2 = grabando3 = false;

                  }
            });

      }

      private void acelerometro(SensorManager manager){

            tv_x = (TextView)findViewById(R.id.txtAccX);
            tv_y = (TextView)findViewById(R.id.txtAccY);
            tv_z = (TextView)findViewById(R.id.txtAccZ);

            List<Sensor> listaSensores =
manager.getSensorList(Sensor.TYPE_ACCELEROMETER);
          Sensor acelerometerSensor = listaSensores.get(0);
          manager.registerListener(this, acelerometerSensor,
          SensorManager.SENSOR_DELAY_UI);

            bt_empezar = (Button)findViewById(R.id.btEmpezar);
            bt_finalizar = (Button)findViewById(R.id.btFinalizar);
            bt_listas = (Button)findViewById(R.id.btListas);
            bt_graficas = (Button)findViewById(R.id.btGraficas);
```

```
            bt_empezar.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        grabando1 = true;
                        //Se coge el tiempo actual
                        tiempo = Calendar.getInstance().getTimeInMillis();
                        tiempo_anterior = tiempo;

                        TablaPrincipalDAO dao = new TablaPrincipalDAO();
                        id_principal1 = dao.insert();
                  }
            });

            bt_finalizar.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        grabando1 = false;


                  }
            });

            bt_listas.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        Intent intent = new Intent(MainActivity.this,
ListasPrincipalActivity.class);
                        intent.putExtra("MODO", "Lista");
                        startActivity(intent);

                  }
            });

            bt_graficas.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        Intent intent = new Intent(MainActivity.this,
ListasPrincipalActivity.class);
                        intent.putExtra("MODO", "Grafica");
                        startActivity(intent);

                  }
            });
      }

      private void luz(SensorManager manager){

            List<Sensor> listaSensores =
manager.getSensorList(Sensor.TYPE_LIGHT);
            //List<Sensor> listaSensores =
manager.getSensorList(Sensor.TYPE_TEMPERATURE);

            tv_luz = (TextView)findViewById(R.id.luz);
```

```
            if (!listaSensores.isEmpty()) {
              Sensor temperatureSensor = listaSensores.get(0);
              manager.registerListener(this, temperatureSensor,
SensorManager.SENSOR_DELAY_UI);
            }


            bt_empezar2 = (Button)findViewById(R.id.btEmpezar2);
            bt_finalizar2 = (Button)findViewById(R.id.btFinalizar2);
            bt_listas2 = (Button)findViewById(R.id.btListas2);
            bt_graficas2 = (Button)findViewById(R.id.btGraficas2);

            bt_empezar2.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        grabando2 = true;
                        //Se coge el tiempo actual
                        tiempo = Calendar.getInstance().getTimeInMillis();
                        tiempo_anterior = tiempo;

                        TablaLuzPrincipalDAO dao = new
TablaLuzPrincipalDAO();
                        id_principal2 = dao.insert();
                  }
            });

            bt_finalizar2.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        grabando2 = false;
                  }
            });

            bt_listas2.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        Intent intent = new Intent(MainActivity.this,
ListasLuzPActivity.class);
                        intent.putExtra("MODO", "Lista");
                        startActivity(intent);

                  }
            });

            bt_graficas2.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {
                        Intent intent = new Intent(MainActivity.this,
ListasLuzPActivity.class);
                        intent.putExtra("MODO", "Grafica");
                        startActivity(intent);
```

```
            }
        });

    }

    private void magneticField(SensorManager manager){

        List<Sensor> listaSensores =
manager.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);

        tv_magnetic_field = (TextView)findViewById(R.id.magnetometro);

        if (!listaSensores.isEmpty()) {
          Sensor temperatureSensor = listaSensores.get(0);
          manager.registerListener(this, temperatureSensor,
SensorManager.SENSOR_DELAY_UI);
        }


        bt_empezar3 = (Button)findViewById(R.id.btEmpezar3);
        bt_finalizar3 = (Button)findViewById(R.id.btFinalizar3);
        bt_listas3 = (Button)findViewById(R.id.btListas3);
        bt_graficas3 = (Button)findViewById(R.id.btGraficas3);

        bt_empezar3.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                grabando3 = true;
                //Se coge el tiempo actual
                tiempo = Calendar.getInstance().getTimeInMillis();
                tiempo_anterior = tiempo;

                TablaMagneticoPrincipalDAO dao = new
TablaMagneticoPrincipalDAO();
                id_principal3 = dao.insert();
            }
        });

        bt_finalizar3.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                grabando3 = false;
            }
        });

        bt_listas3.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                //TODO cambiar ventana
                Intent intent = new Intent(MainActivity.this,
ListasMagnetismoPActivity.class);
                intent.putExtra("MODO", "Lista");
                startActivity(intent);
```

```
            }
        });

        bt_graficas3.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                //TODO cambiar ventana
                Intent intent = new Intent(MainActivity.this,
ListasMagnetismoPActivity.class);
                intent.putExtra("MODO", "Grafica");
                startActivity(intent);

            }
        });


    }

    @Override
    public void onSensorChanged(SensorEvent event) {


        if( ( tiempo >= tiempo_anterior +1000) &&
(grabando1||grabando2||grabando3)){
            synchronized (this) {
              switch(event.sensor.getType()) {
                  case Sensor.TYPE_ACCELEROMETER:
                      if(grabando1)
                          onAcelerometerChange(event);
                      break;
                  case Sensor.TYPE_LIGHT:
                      if(grabando2)
                          onLightChange(event);
                      break;
                  default:
                      if(grabando3)
                          onMagneticFieldChange(event);
                      break;
              }
            }
        }
        tiempo = Calendar.getInstance().getTimeInMillis();


    }

    private void onAcelerometerChange(SensorEvent event){

        tv_x.setText("Valor de X: "+event.values[0]);
    tv_y.setText("Valor de Y: "+event.values[1]);
    tv_z.setText("Valor de Z: "+event.values[2]);

    double x = event.values[0];
    double y = event.values[1];
    double z = event.values[2];
```

```
      //data access object

      if(id_principal1 != -1){
            Calendar ahora = Calendar.getInstance();
            SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy
HH:mm:ss a");
            String f1 = df3.format(ahora.getTime());
            ElementoMovimiento movimiento = new ElementoMovimiento(-1,
(int)id_principal1, x, y, z, f1);

            TablaMovimientoDAO dao_movimiento = new TablaMovimientoDAO();
            dao_movimiento.insert(movimiento);
      }

      tiempo_anterior = tiempo;


      }

      private void onLightChange(SensorEvent event){

            tv_luz.setText("Iluminancia: "+event.values[0]+" lux");

            double luz = event.values[0];

            if(id_principal2 != -1){
            Calendar ahora = Calendar.getInstance();
            SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy
HH:mm:ss a");
            String f1 = df3.format(ahora.getTime());
            ElementoLuminosidad movimiento = new ElementoLuminosidad(-1,
(int)id_principal2, luz, f1);

            TablaLuzDAO dao_luz = new TablaLuzDAO();
            dao_luz.insert(movimiento);
      }

      tiempo_anterior = tiempo;

      }

      private void onMagneticFieldChange(SensorEvent event){
            tv_magnetic_field.setText("Campo magnetico:
"+event.values[0]+" uT");

            double magnetismo = event.values[0];

            if(id_principal3 != -1){
            Calendar ahora = Calendar.getInstance();
            SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy
HH:mm:ss a");
            String f1 = df3.format(ahora.getTime());
            ElementoCampoMagnetico movimiento = new
ElementoCampoMagnetico(-1, (int)id_principal3, magnetismo, f1);

            TablaMagneticoDAO dao_mag = new TablaMagneticoDAO();
            dao_mag.insert(movimiento);
```

```
	}

	tiempo_anterior = tiempo;

	}

	@Override
	public void onAccuracyChanged(Sensor sensor, int accuracy) {
		// TODO Auto-generated method stub

	}


}
```