Parte 1 – Descripción del proyecto

**Lista de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Autor** |
| 1.1 | 11/10/2011 | Esqueleto inicial | Romina Liuzzi |
| 1.2 | 21/04/2012 | Ampliación de apartado de introducción | Romina Liuzzi |

Tabla 1. Lista de cambios

Tabla de contenidos

[Introducción 5](#_Toc322733870)

[Objetivos del proyecto 8](#_Toc322733871)

[Objetivos marcados por los requisitos funcionales 8](#_Toc322733872)

[Objetivos de carácter técnico 8](#_Toc322733873)

[Conceptos teóricos 9](#_Toc322733874)

[Arquitectura Android 9](#_Toc322733875)

[Aplicación 10](#_Toc322733876)

[Actividad 11](#_Toc322733877)

[Ciclo de vida de una aplicación 12](#_Toc322733878)

[Vista 13](#_Toc322733879)

[Estructura de un proyecto Android 15](#_Toc322733880)

[Técnicas y herramientas 19](#_Toc322733881)

[Requisitos del sistema SDK Android 19](#_Toc322733882)

[ADT de Android 21](#_Toc322733883)

[Base de datos integrada SQLite 25](#_Toc322733884)

[Aspectos relevantes del desarrollo del proyecto 29](#_Toc322733885)

[Tendencias 29](#_Toc322733886)

[Sensores 31](#_Toc322733887)

[Trabajos relacionados 32](#_Toc322733888)

[Conclusiones y líneas de trabajo futuras 33](#_Toc322733889)

# Introducción

El presente proyecto comprende dos partes fundamentales. En primer lugar un estudio previo de la plataforma, entorno de desarrollo y técnicas y herramientas necesarias para desarrollar y publicar una aplicación móvil en Android. Y en segundo lugar el diseño e implementación de una aplicación que nos permita reflejar de una manera práctica e intuitiva los resultados obtenidos en el proyecto de investigación. Asimismo la aplicación desarrollada servirá al usuario final para medir los progresos obtenidos a partir de un programa de ejercicios.

En el presente documento se reflejarán ambos apartados, siguiendo la estructura que se define a continuación.

Primera parte. Descripción del proyecto:

Esta sección del documento contendrá las introducciones teóricas necesarias para el correcto acercamiento al proyecto. Se compone de los siguientes apartados:

* Introducción: breve descripción del proyecto realizado y resumen del contenido de la memoria y documentación técnica del mismo.
* Objetivos del proyecto: se abordarán los objetivos que se pretender conseguir con la realización de este proyecto de fin de carrera.
* Conceptos teóricos: Definición de conceptos que abarcan algún campo relacionado con el proyecto.
* Técnicas y Herramientas: se tratarán las técnicas y herramientas empleadas en la elaboración del proyecto, desde la parte de investigación a la de diseño e implementación de la aplicación web, así como herramientas auxiliares empleadas a lo largo de todo el proceso o en la elaboración misma de esta documentación.
* Aspectos relevantes del desarrollo del proyecto: en este apartado se pondrán de relieve aquellos aspectos destacables en la elaboración y desarrollo de este proyecto de fin de carrera.
* Trabajos relacionados: En este apartado se presentarán aplicaciones comerciales con objetivos similares a la solución adoptada por este proyecto.
* Conclusiones y líneas de trabajo futuras: En este último apartado se comentarán las posibles mejoras que se pueden implementar, tanto para ampliar a nivel funcional como a nivel de infraestructura. Así mismo se extraerán aquí las conclusiones derivadas de la consecución del proyecto.

Segunda parte. Documentación técnica:

Esta segunda sección del documento contendrá los apartados específicos del análisis, diseño e implementación de la aplicación, así como los distintos manuales. Su estructura será la siguiente:

* Plan del proyecto: Se comentará el calendario previsto así como los detalles de planificación.
* Especificación de requisitos: En este anexo, se detallarán los requisitos correspondientes a la aplicación Android.
* Especificación de diseño: En este anexo se justifican las decisiones tomadas para implementar la solución al problema descrito en la fase anterior.
* Documentación técnica de programación: apartado dedicado a comentar aspectos relevantes en la implementación de la aplicación, que pueda servir de apoyo a futuras modificaciones en la misma.
* Manuales de usuario: completo manual de la aplicación orientado al usuario final de la misma. Se explicará el proceso de instalación y se tratarán todos los menús y contenidos presentes en la aplicación. También se incluye un apartado para el programador, donde se describe el proceso de desarrollo y depuración de la aplicación.

# Objetivos del proyecto

Se distinguen dos tipos de objetivos; los marcados por los requisitos del software o requisitos funcionales y los objetivos de carácter técnico.

## Objetivos marcados por los requisitos funcionales

Se busca diseñar e implementar una aplicación móvil que permita medir el progreso conseguido en distintas rutinas de ejercicios a lo largo del tiempo. Se pretende distribuir la aplicación a través del Android Market con la idea de ser utilizado con fines recreativos una vez completado su desarrollo.

Un usuario de la aplicación podrá realizar distintos tipos de ejercicios que involucren desplazamientos físicos reales. De tal forma que, mediante los cambios en las coordenadas GPS, recogidos a través de los sensores integrados en el terminal, sea posible calcular: la distancia recorrida, el tiempo, la velocidad y las calorías quemadas al finalizar la actividad.

La aplicación deberá ser capaz de explotar los datos obtenidos a partir de los sensores, persistirlos y utilizarlos para presentar al usuario con gráficas y estadísticas inherentes a su progreso. Será imprescindible que el terminal cuente con un sensor GPS integrado y activo para su correcto funcionamiento. Sera posible habilitar y deshabilitar el sensor desde un menú de la aplicación.

Al ser una aplicación con fines puramente lúdicos, es aceptable que los algoritmos utilizados en los cálculos presenten valores aproximados.

En cambio es importante que las gráficas presentadas en la aplicación representar de forma atractiva los factores claves para motivar al usuario a seguir haciendo ejercicio. Será posible conocer las velocidades máximas alcanzadas, los mínimos y los promedios. De forma parecida se tratarán las calorías quemadas y peso perdido.

Será posible acceder a un resumen histórico de ejercicios. Se pretende también que sea la propia aplicación la que evalúe el consumo de calorías en función de los datos de usuario. Será importante que estos datos sean recientes por lo que se debe asegurar que son válidos antes de comenzar el ejercicio. En caso que los datos no sean válidos se podrá actualizarlos antes de comenzar.

Debe garantizarse el correcto funcionamiento a nivel de seguridad, de rendimiento y a nivel funcional de la aplicación. Para ello se debe definir un proceso de testeo adecuado.

## Objetivos de carácter técnico

Al desarrollo de este proyecto subyace la necesidad de explotar y ampliar los conocimientos adquiridos durante la carrera. De este modo ha sido posible estructurar la aplicación en base al modelo de tres capas: datos, lógica y presentación. Una aproximación al modelo MVC adaptado para Android.

Para la parte del desarrollo se ha usado lenguajes Orientados a Objetos, concretamente: Java. En Android el proceso se ejecuta dentro de una máquina virtual, sobre un Kernel de Linux casi completo. Es necesario interiorizar también estas tecnologías para el correcto despliegue de la aplicación.

Además, es también objetivo del proyecto perfeccionar conocimientos sobre:

* El uso de entornos de desarrollo, cómo Eclipse en sus distintas distribuciones (STS, IDE for Java Developers, etc), que mejora notablemente el desarrollo con la integración de los servidores y facilitando la utilización de librerías externas como parte del propio proyecto.
* La integración de plugins para desarrollo, por ejemplo, para Android: Android ADT que incluye las herramientas del SDK y el AVD, entre muchas otras funcionalidades.
* Las librerías externas que permiten completar las funcionalidades ofrecidas por la aplicación. Ej. AChartEngine, Joda Time, Google Maps API.
* Profundizar sobre opciones y buenas prácticas a la hora de persistir datos en Android.
* Adquirir conocimiento sobre versionado y aprender a utilizar un gestor de versiones distribuido (gitHub).
* Comprender las diferencias, respecto a un entorno web, a la hora de crear una Suite de pruebas unitarias dentro de un proyecto Android.
* Profundizar en el uso de servidores Web. Concretamente Apache Tomcat, pero extrapolable a otras plataformas.
* Aprender sobre la tecnología Java JSP (Java Server Pages), que proporciona de una forma rápida y sencilla la creación de páginas Web con contenido dinámico.
* Realizar una buena definición de las pruebas de aceptación y cumplir con lo pautado.

Finalmente, aunque no menos importante, otro de los objetivos marcados a la hora de abordar el proyecto es intentar utilizar el mayor número de herramientas de libre distribución para demostrar que es posible realizar un proyecto con un desembolso económico mínimo.

# Conceptos teóricos

Arquitectura Android

Android es un sistema operativo para dispositivos móviles y tablets propiedad de Google y distribuido bajo el Standard OpenSource.

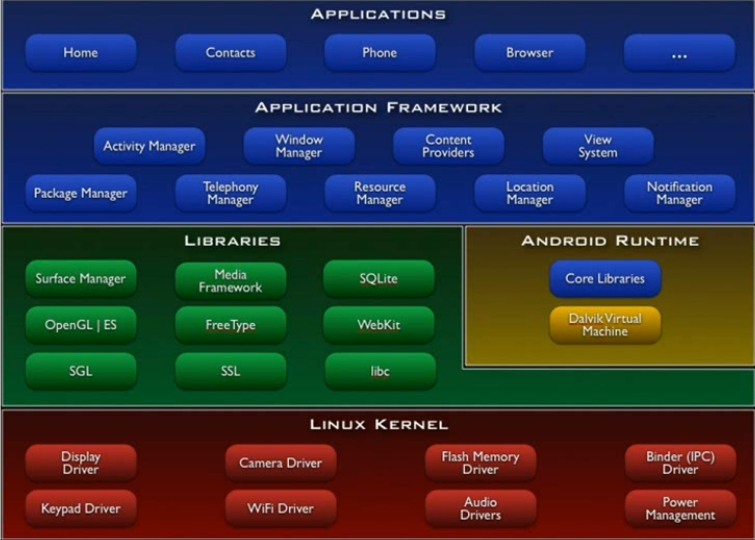


Figura . Arquitectura Android

Como se puede observar en la imagen, el Kernel de Linux cuenta con un gran número de controladores. Es una capa de abstracción con el hardware, ninguna otra capa se encargará de esto. La seguridad del dispositivo también se controla desde aquí.

En la capa de librerías nos encontramos mayormente con librerías escritas en C y C#. Aquí se maneja un gran número de recursos gráficos. Se utiliza un esquema de base de datos relacional SQLite.

A la hora de implementar nos moveremos sobre la capa de aplicaciones, trabajando únicamente sobre el framework de las aplicaciones. El manejo de los niveles de abstracción inferiores nos resultará prácticamente transparente.

Aplicación

Una aplicación es un set de servicios, vistas y recursos unidos dentro de un mismo paquete. En Android lo más natural es escribir las aplicaciones en Java y a partir de las herramientas provistas por el SDK generar nuestros ejecutables “.apk”.

El ejecutable será todo el código que los terminales Android necesiten para instalar la aplicación. Una vez instalada el sistema la ejecutará como un usuario de un sistema Linux multi-usuario. Cada usuario (aplicación) vendrá identificada por un identificador de usuario, conocido solo por el sistema.

Actividad

Una aplicación tal como explicábamos antes, se puede entender como un set de servicios no demasiado ligados entre sí que se distribuyen juntos. Estos servicios pueden ser:

* Actividades - una idea vaga de este concepto es una pantalla
* Servicios – Tal como los servicios de Windows que se ejecutan de fondo
* Broadcast Receivers – Componentes que están a la espera de un cambio para responder
* Content Providers – Proveedores de contenido que a través de una interfaz común básicamente hacen una consulta y devuelven un resultado.

Las aplicaciones de Android deben entenderse como Actividades o pantallas y diseñarse con este concepto en mente. Cada pantalla debe ser lo más independiente posible, una actividad equivale a un módulo independiente.

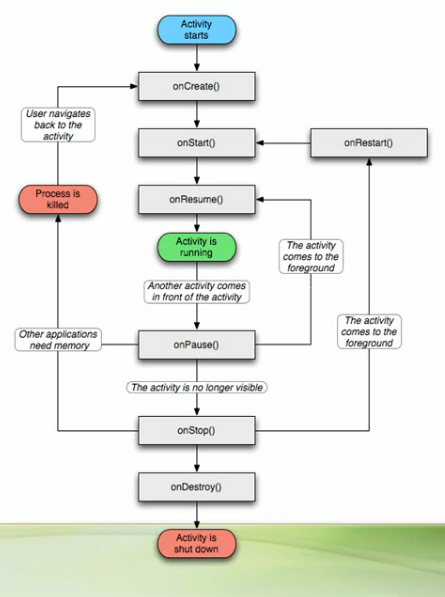
Ciclo de vida de una aplicación

Figura 2. Ciclo de vida de una aplicación

El esquema anterior representa todos los posibles estados que puede atravesar una aplicación durante su vida. Estos métodos están implementados en la clase Activity y son heredados en cada una de nuestras actividades. El funcionamiento de la mayoría de ellos es transparente, de forma que no necesitamos sobrecargar el método del constructor.

Una excepción es el método onCreate() que debemos implementar siempre, es llamado cuando se inicia la actividad.

Dentro de éste diagrama de flujo es importante notar que existen tres posibles estados en los que se puede “terminar/iniciar” una actividad.

onPause() 🡪 onResume()

onStop() 🡪 onRestart() 🡪 onStart()

onDestroy() 🡪 onCreate()

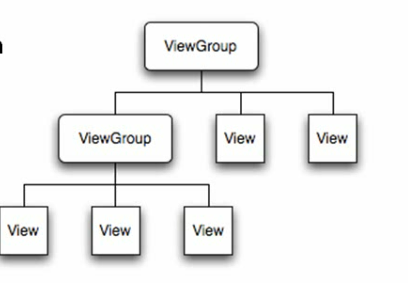


Figura . Vistas

Vista

El concepto de vista está sobrecargado, de forma que se utiliza tanto para referirse a la interfaz de usuario, a los elementos dentro de la interfaz de usuario, o a un layout.

Siguen el Patrón Compuesto (Composite Pattern). La raíz de un GrupoVistas puede contener a su vez más GrupoVistas. En cierta forma el patrón se parece un poco al modelo vista controlador, donde la actividad es una especie de controlador.

Estructura de un proyecto Android

En este apartado se explica la estructura de ficheros de un proyecto Android, sus drectorios y ficheros más relevantes.

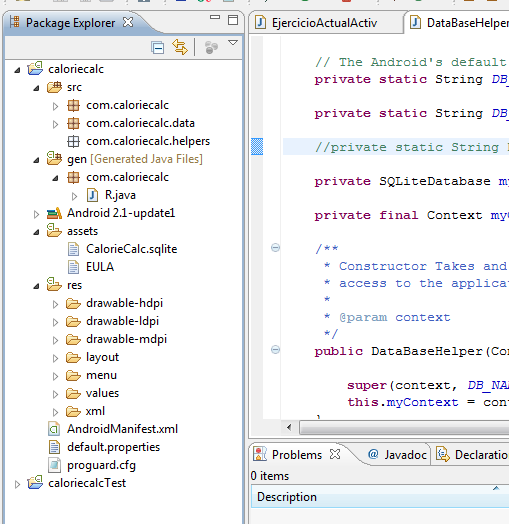


Figura 4. Estructura Proyecto Android

src/

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.

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 Java dirigidos al control de los recursos de la aplicación.

assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, de datos, etc.

*NOTA: La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Usaremos uno u otro según las necesidades de nuestra aplicación.*

res/

Contiente todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:

res/drawable/

Contienen las imágenes de la aplicación. Se puede dividir en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi para utilizar diferentes recursos dependiendo de la resolución del dispositivo.

res/layout/

Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land para definir distintos layouts dependiendo de la orientación del dispositivo.

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.

AndroidManifest.xml

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.

R.java

Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a ellos.

# Técnicas y herramientas

Requisitos del sistema SDK Android

Las secciones descritas a continuación describen el sistema y los requisitos del sistema para el desarrollo de aplicaciones Android utilizando el SDK de Android.

Sistemas operativos soportados:

• Windows XP (32-bit), Vista (32- o 64-bit), o Windows 7 (32- o 64-bit)

• Mac OS X 10.5.8 o posterior (solo x86)

• Linux (probado en Ubuntu Linux, Lucid Lynx)

o Se requiere librería GNU C (glibc) 2.7 o posterior.

o En Linux Ubuntu se requiere tener instalada la versión 8.04 o posterior.

o Las distribuciones de 64-bit deben ser capaces de ejecutar aplicaciones de 32-bit. Para obtener información sobre cómo soportar aplicaciones de 32-bit se puede visitar “Ubuntu Linux installation notes”.

Entornos de Desarrollo Soportados:

Eclipse IDE

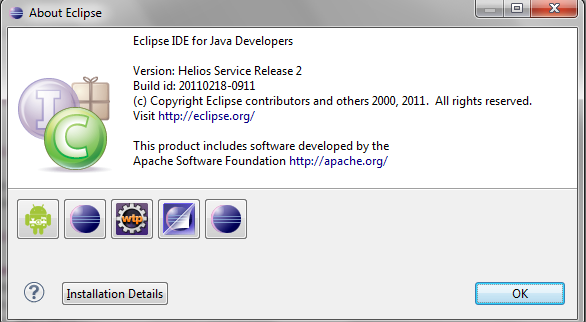
• Eclipse 3.5 (Galileo) o posterior.

Nota: Eclipse 3.4 (Ganymede) ya no se soporta en la última versión del ADT.

• Plugin JDT de Eclipse JDT (se incluye en la mayoría de paquetes de IDE Eclipse)

Para desarrollar aplicaciones de Android se encuentran en nuestras manos numerosas opciones a la hora de desarrollar

Tenemos a nuestra disposición un elevado número de paquetes de Eclipse para cada plataforma, para nuestro caso particular hemos adoptado:



Siendo preciso instalar además el JDK de Java versión jdk1.6.0\_24.

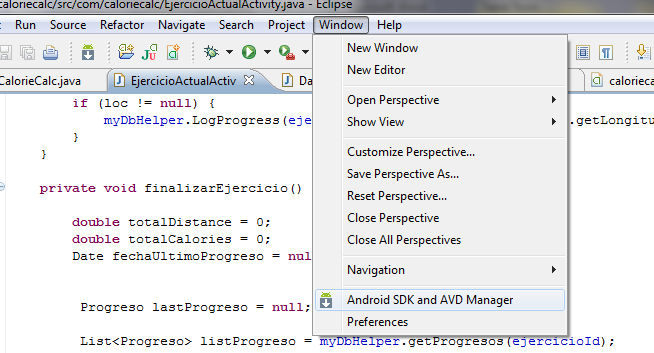
**Requisitos a nivel Hardware:**

El SDK de Android requiere guardar datos para todos los componentes que se decidan instalar, a continuación se provee una idea robusta de los requisitos en cuanto a cuota de disco necesaria basada en nuestras opciones.

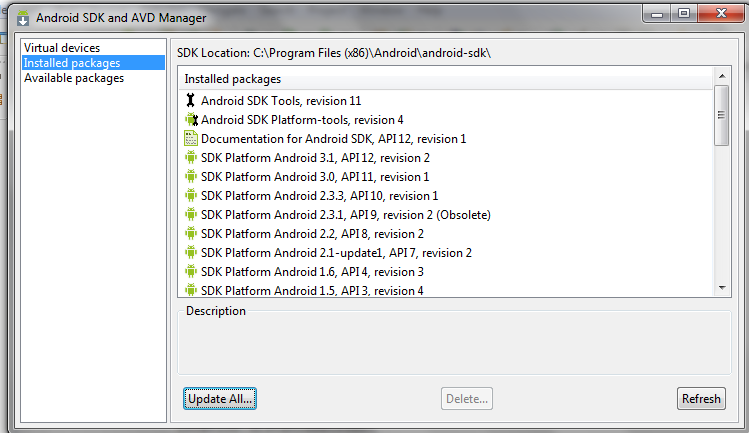
|  |  |  |
| --- | --- | --- |
| **Tipo de Componente** | **Tamaño Aproximado** | **Comentarios** |
| SDK Tools | 35 MB | Requerido |
| SDK Platform-tools | 6 MB | Requerido |
| Android platform (each) | 150 MB | Al menos una |
| SDK Add-on (each) | 100 MB | Opcional |
| USB Driver for Windows | 10 MB | Opcional y solo para Windows |
| Samples (per platform) | 10M | Opcional |
| Offline documentation | 250 MB | Opcional |

ADT de Android

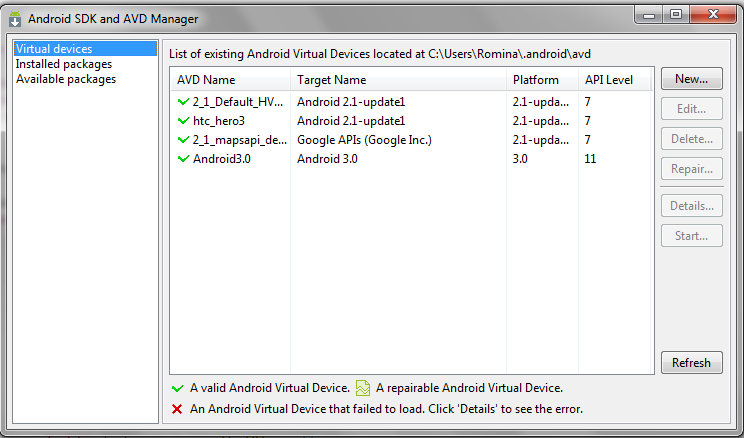
El plugin de Android para eclipse nos permite instalar todos los sdk que necesitemos asi como paquetes adicionales desde el mismo IDE.



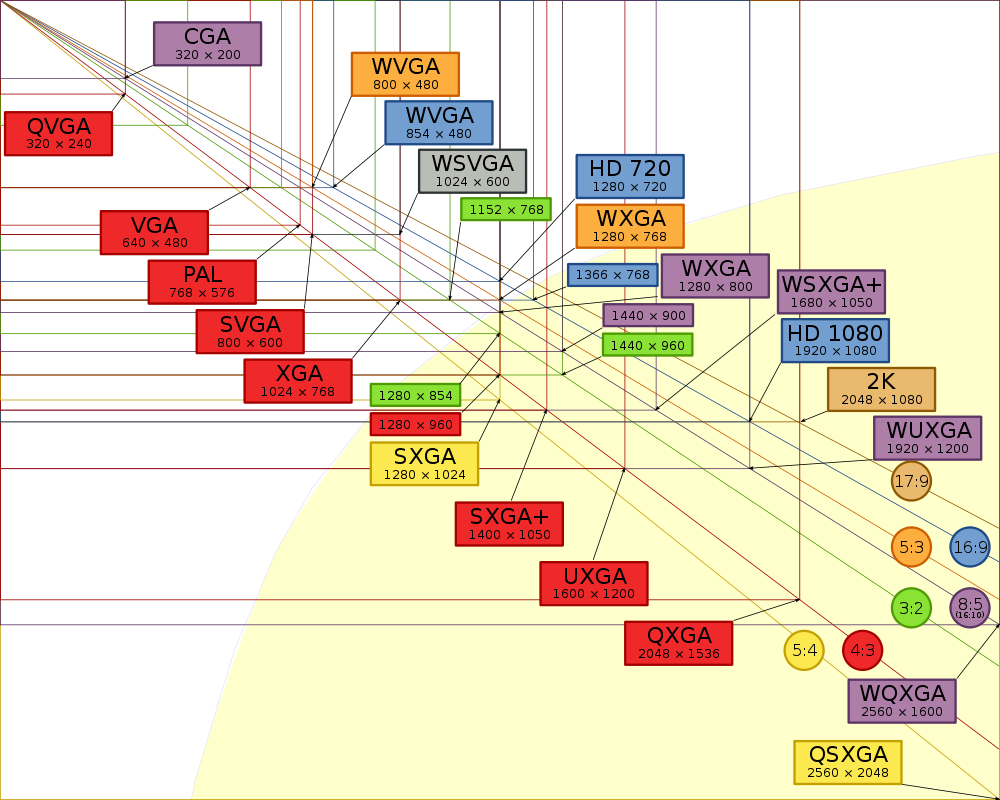
Resulta casi trivial mantener actualizados nuestros paquetes con los ultimos updates.



Es realmente sencillo crear un emulador que represente de forma feaciente nuestro modelo de móvil.



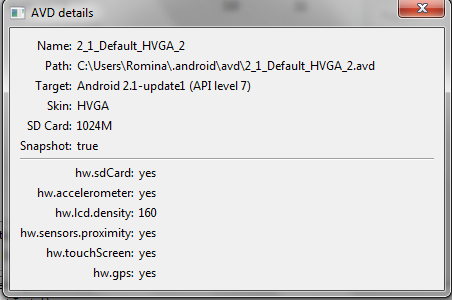
Nos basta con revisar las especificaciones técnicas del fabricante donde se detallan todos los datos técnicos que podamos necesitar. Una de las características más importantes a tener en cuenta durante el desarrollo consiste en la piel, los tamaños y configuraciones se pueden revisar en el esquema que se adjunta. De esta manera podemos validar que nuestro diseño de las vistas se adapta correctamente tanto a distintos tamaños de pantalla, interfaces gráficas, cambios de orientación, etc.

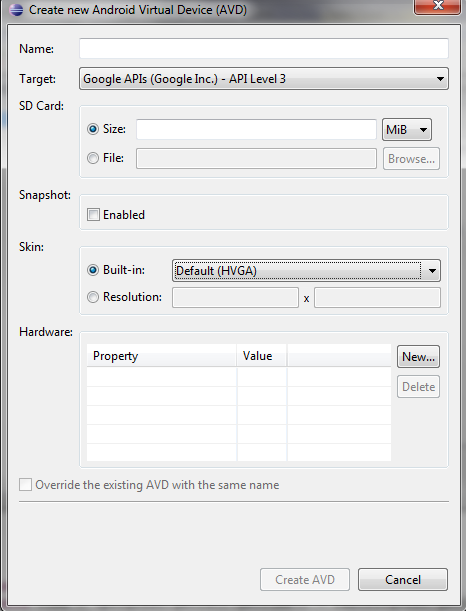


*Esquema de tamaños relativos estandarizados.*

El emulador que utilizamos durante el desarrollo de este proyecto así como las características técnica predefinidas durante su creación.







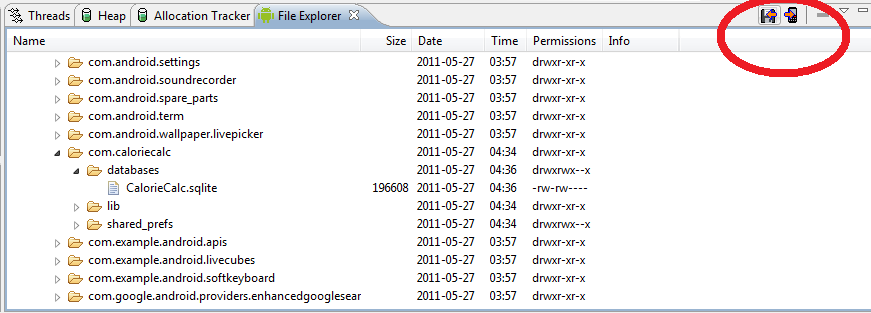
Base de datos integrada SQLite

Android ofrece diferentes opciones a la hora de decidir el tipo de almacenamiento que utilizaremos para nuestra aplicación. Una de las opciones presentadas es la creación de un esquema de base de datos relacional SQLite. Todas las operaciones relacionadas con la base de datos se recomienda sean llevadas a cabo desde una clase DBHelper que hace de interfaz entre la base de datos y la aplicación. Una vez se entiende la dinámica, no es difícil entender su funcionamiento.

La base de datos de la aplicación se crea al instalar por primera vez y se actualiza cada vez que lo deseemos. En la primera creación, se creará el esquema bajo el directorio:

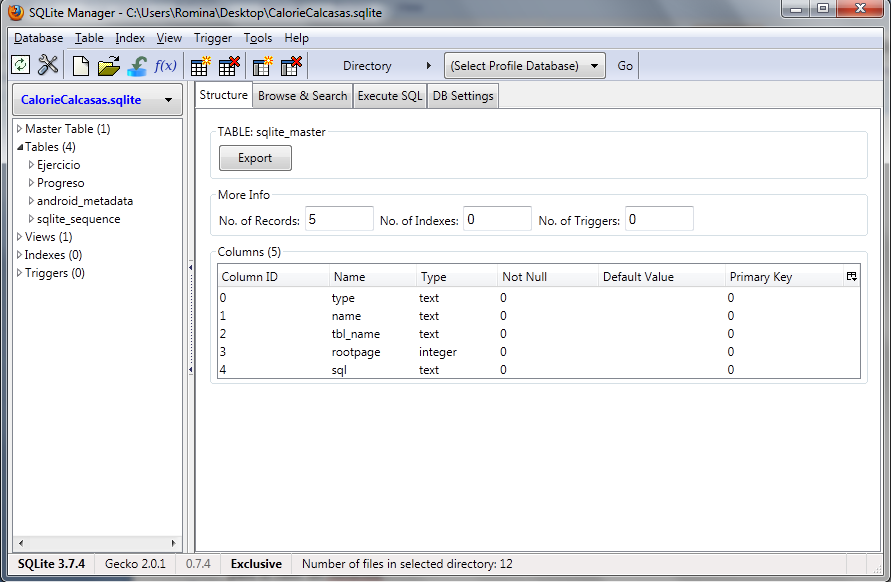
Data/data/<nombre del paquete>/databases

Y podemos monitorizarlo a través del File Explorer:

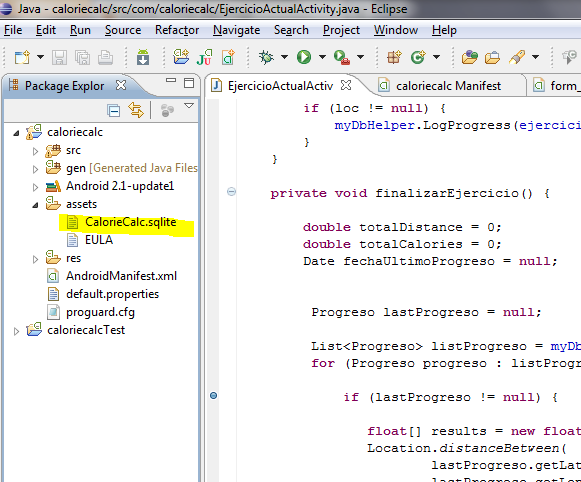


Tenemos la opción de bajar datos desde el terminal a nuestro PC desde esta pestaña, también es posible realizar el camino inverso.

Durante este proyecto se utilizó un **plugin para Firefox** llamado **SQLite Manager**, que permite realizar las operaciones más comunes sobre la base de datos, desde realizar SELECTS, crear VIEWS e incluso crear las bases de datos desde aquí. En la especificación técnica se explora una alternativa a esta opción.



Una vez tenemos terminado el esquema es posible arrastrarlo dentro del contexto de nuestro espacio de trabajo en la carpeta Assets, de tal forma que al instalar el .apk de nuestra aplicación se cree dentro del registro correspondiente (data/data…).



Shorcuts Eclipse

Ctrl + espacio: Autocompletar

Ctrol + Shift + F: Auto formatear

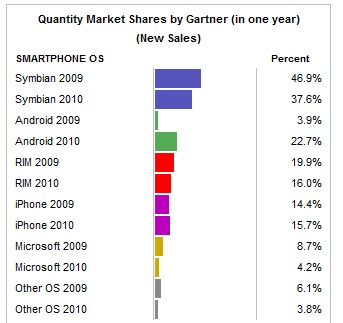
Ctrl + Shift + O: Auto importar paquetes

Alt + Shift + R: Renombrar y actualizar referencias

Ctrl + F12: Rotar la pantalla del emulador

Ctrl + O: Buscar métodos dentro de la clase.

# Aspectos relevantes del desarrollo del proyecto

Tendencias

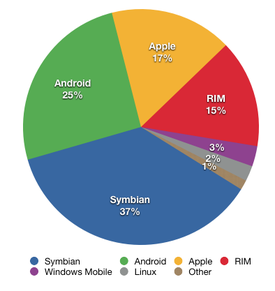
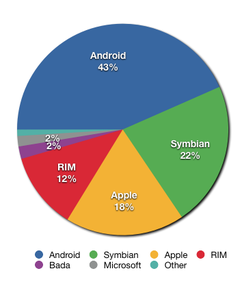
Ventas a nivel mundial de Smartphones a usuarios finales clasificadas por sistema operativo.

Figure 5. Ventas Q2 2011

Figura 6. Ventas Q3 2010

Sensores

Lo más importante para cualquier proyecto de esta índole es la capacidad de recoger data de los sensores a fin de estudiarlos.

* Accelerómetro de 3 ejes (También lo llaman G-Sensor)
* Sensor de luz ambiental
* Sensor de proximidad
* GPS, brújula digital
* WiFi
* Giroscopio

El acelerómetro de 3 ejes, mide es la aceleración lineal cuando el dispositivo se encuentra derecho, si el ángulo de giro cambia puede confundir la influencia de la gravedad con aceleración lineal.

El giroscopio mide el ángulo de giro, con lo cual las medidas son claras y rápidas.

Si se combinan ambos se obtiene un sistema inercial de medida de 6 ejes, ya que el acelerómetro utiliza la información medida por el giroscopio y luego puede medir sin interferencias la aceleración lineal en cualquier ángulo.

*Nota: estos sensores ya se están usando en mandos de consolas como Wii Sport para traducir movimientos complejos. Algunos móviles ya están empezando a incorporarlo (ej. HTC Evo 4G).*

# Trabajos relacionados

MyTracks:

Esta aplicación permite al usuario guardar sus rutas de deporte, compartirla vía Google maps, generar estadísticas a partir de ellas, así como mostrar por pantalla una serie de datos calculados a partir de la actividad y datos de sensores.

Si bien se trata de aplicaciones muy completas, es posible ir agregando funcionalidades de forma iterativa al modelo ofrecido en este PFC como prototipo inicial.

# Conclusiones y líneas de trabajo futuras