Anexo IV – Documentación Técnica de Programación

**Lista de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Autor** |
| 1.1 | 11/10/2011 | Esqueleto inicial | Romina Liuzzi |
| 1.2 | 11/06/2012 | Completar documento | Romina Liuzzi |

Tabla de contenidos

[Introducción 5](#_Toc328163836)

[Documentación de las bibliotecas 6](#_Toc328163837)

[Google APIs [Android 2.1] 7](#_Toc328163838)

[AChartEngine 10](#_Toc328163839)

[JodaTime 11](#_Toc328163840)

[Código fuente 12](#_Toc328163841)

[src/ 13](#_Toc328163842)

[gen/ 14](#_Toc328163843)

[assets/ 14](#_Toc328163844)

[res/ 14](#_Toc328163845)

[res/drawable/ 14](#_Toc328163846)

[res/layout/ 14](#_Toc328163847)

[res/menu/ 14](#_Toc328163848)

[res/values/ 15](#_Toc328163849)

[res/xml/ 15](#_Toc328163850)

[res/raw/ 15](#_Toc328163851)

[AndroidManifest.xml 15](#_Toc328163852)

[R.java 15](#_Toc328163853)

[Manual del programador 16](#_Toc328163854)

[1. Instalar Eclipse 16](#_Toc328163855)

[2. Instalar el ADT 16](#_Toc328163856)

[3. Instalar SDK de Android 17](#_Toc328163857)

[Pruebas 21](#_Toc328163858)

# Introducción

El presente proyecto consiste en el diseño e implementación de una aplicación para móvil que permite calcular el gasto de calorías durante la práctica de ejercicio al aire libre. Durante la creación de ésta aplicación se han sorteado una serie de obstáculos a nivel diseño e implementación, debido en gran parte, a la falta de experiencia.

Algunos de los desafíos enfrentados durante el proceso de desarrollo han consistido en encontrar el equilibrio entre velocidad de respuesta, rendimiento, seguridad y gasto mínimo de recursos por parte de la aplicación a la vez que se cumplían los plazos acordados en la fase de planificación.

Cabe destacar que la aplicación es nativa de Android, es decir, el lenguaje de programación utilizado es Java explotando las librerías del API para Android distribuidas por Google. La plataforma de desarrollo es Eclipse, y la base de datos nativa elegida es SQLite, también con soporte nativo.

Este anexo va a estar formado por la siguiente estructura:

* Documentación de bibliotecas.
* Código fuente: descripción de las directrices que se han utilizado
* Manual del programador: se describe la generación del código compilado a partir de los ficheros fuente
* Pruebas: descripción de las pruebas realizadas sobre el sistema

# Documentación de las bibliotecas

En este proyecto se han utilizado componentes de código ya creados previamente, pertenecientes a fuentes externas. No ha sido necesario modificarlas para adaptarlas a nuestras necesidades. Las librerías o fuentes externas usadas son:

* Google APIs [Android 2.1]: SDK de android versión 7 que incluye el sdk en android.jar y el complemento de google maps en maps.jar.
  + android
  + maps
* AChartEngine: Una librería externa de gráficos.
* Joda Time: Una librería externa con funcionalidades varias sobre tipos Date.

## Google APIs [Android 2.1]

Introducción

La decisión de trabajar sobre este nivel de API viene condicionada por las siguientes circunstancias.

Por un lado, Google Play (la plataforma de comercialización de apps de Google) filtra las aplicaciones que ofrece a un usuario a partir del nivel de SDK definido por el desarrollador en el fichero AndroidManifest.xml.

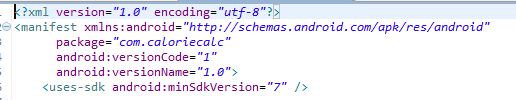


Figura 1. Versión de SDK mínima definida en el AndroidManifest.xml

Es decir que es recomendable utilizar una versión de SDK tan baja como nos sea posible para poder alcanzar el mayor número de usuarios.

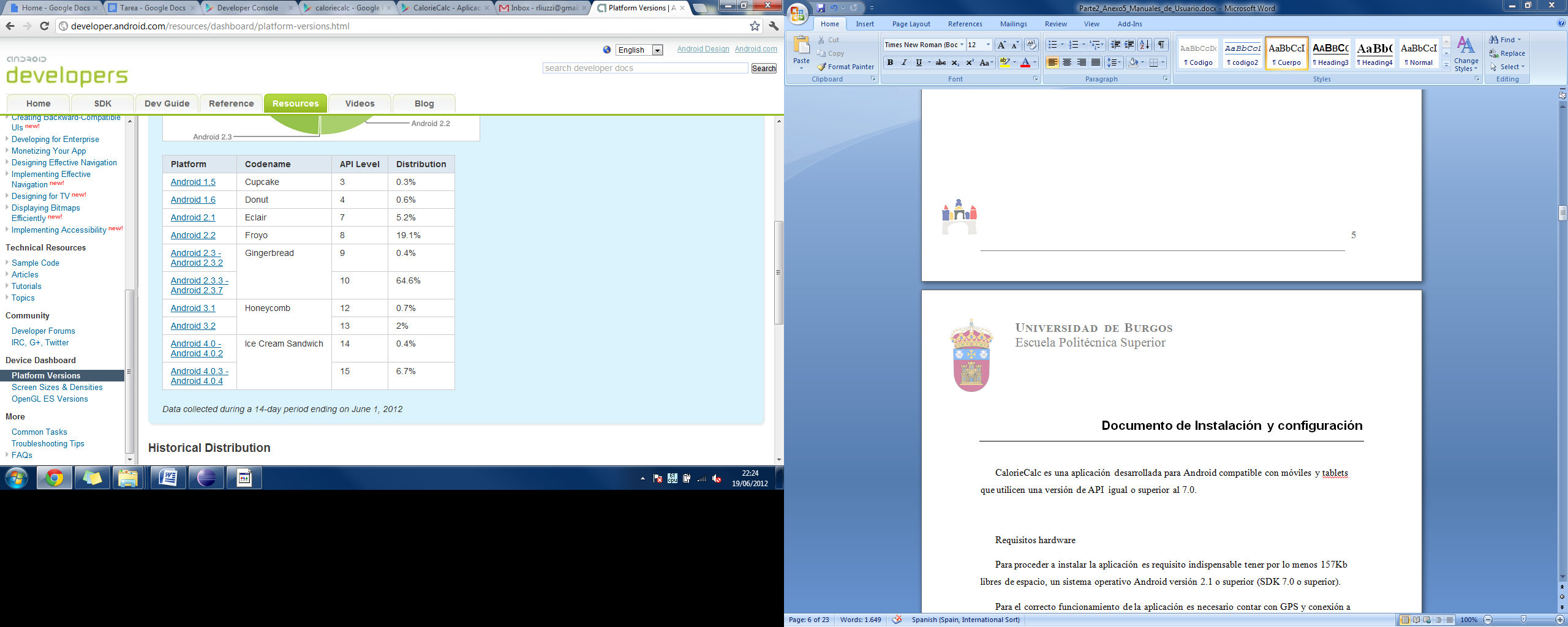


Tabla 1. Distribución SO Android, correspondencia con nivel de API

Si bien, la mayoría de los dispositivos actuales de Android (concretamente el 99.1%) cuentan con una versión de SO igual o superior al 2.1, a nivel de API los cambios incluidos en la versión 8 no nos reportan mayor beneficio de cara al desarrollo de nuestra aplicación.

Que se incluye en el SDK

Esta distribución del sdk incluye dos librerías en una:

* Android.jar
* Maps.jar

Es decir que nos ofrece soporte para desarrollar aplicaciones de android que integran mapas de Google, tal como es el caso de nuestra aplicación.

Paquetes interesantes

## AChartEngine

## JodaTime

# Código fuente

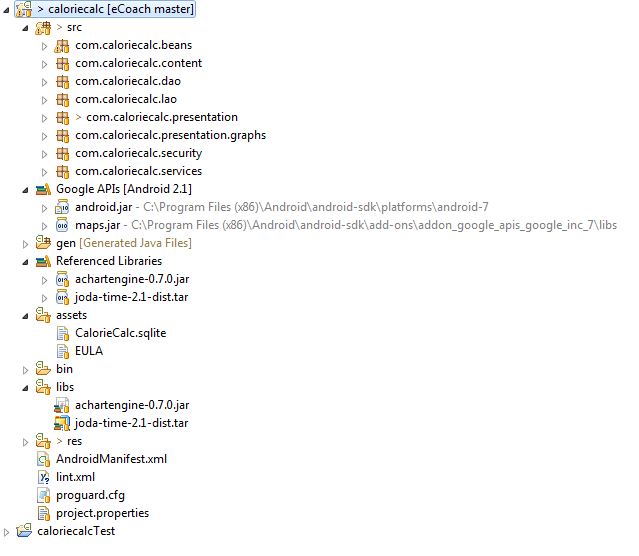
En este apartado se explica la estructura de ficheros del presente proyecto, sus directorios y ficheros más relevantes.

Figura 2. Estructura Básica del Proyecto

## src/

Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Distribuidos en los paquetes definidos en nuestra aplicación. En nuestro caso:

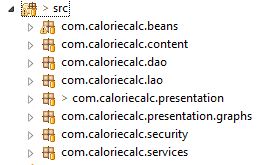


Figura 3. Estructura de paquetes dentro del directorio src

## 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: Se podrá acceder a estos ficheros especificando su ruta como a cualquier otro fichero del sistema.*

## res/

Contiene 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.

### 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.

# Manual del programador

Para el desarrollo y testeo de la parte móvil fue preciso contar con una versión de Eclipse soportada tanto por el SDK de Android como por el respectivo ADT (plugin) de Android para Eclipse.

## Instalar Eclipse

La distribución de Eclipse utilizada para este proyecto fue:

**Eclipse IDE for Java Developers**

**Version: Indigo Service Release 1**

**Plataforma de Eclipse: 3.7.1.r37**

Se puede descargar directamente de la página oficial de [Eclipse](http://www.eclipse.org/downloads/), se incluye el fichero comprimido en formato .zip para Windows de 64-bits. Su instalación consiste en descomprimir el fichero y a abrir la aplicación.

## Instalar el ADT

Por último, se procede a instalar el ADT en eclipse:

Desde Eclipse navegar a: Help > Install New Software.

Agregar repositorio con los valores:

ADT Plugin

https://dl-ssl.google.com/android/eclipse/

## Instalar SDK de Android

Por otra parte el SDK de Android permite disponer de varias plataformas de desarrollo, es posible descargar el ejecutable desde el [Android Developers Site](http://developer.android.com/sdk/index.html), se incluye una versión para Windows del ejecutable en el CD presentado con las fuentes de la aplicación.

A la hora de instalarlo es importante tener en cuenta que para su correcto funcionamiento, la versión de nuestro Java JDK debe ser cumplir con los requisitos mínimos especificados por el SDK de Android. Si se nos presenta el error ilustrado en la siguiente imagen será preciso actualizar la versión de nuestro JDK.

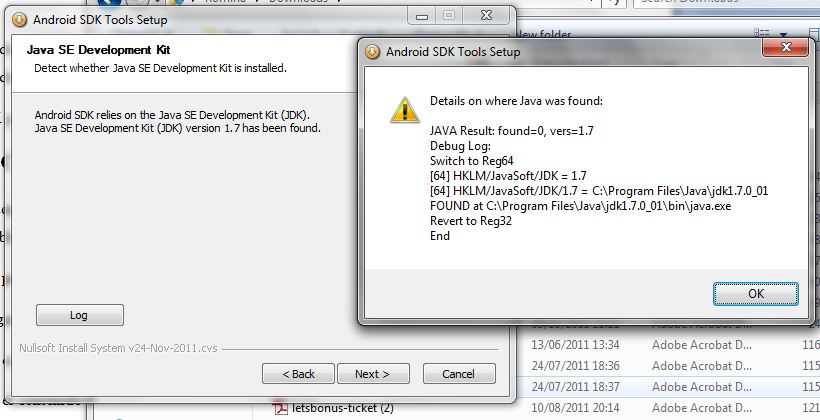


Figura 4. Es preciso tener instalada una versión del JDK compatible con la del SDK de Android.

Otra consideración a tener en cuenta es que para bajar los SDK a través de eclipse es necesario contar con una conexión operativa a internet. Finalmente, cabe tener en cuenta que si se ejecuta desde un entorno de Windows es preciso ejecutar el programa como administrador o se producirán conflictos a la hora de guardar los paquetes descargados en disco.

El SDK Manager de Android viene incluido en el ADT y se puede acceder a él fácilmente tal como ilustra la siguiente imagen.

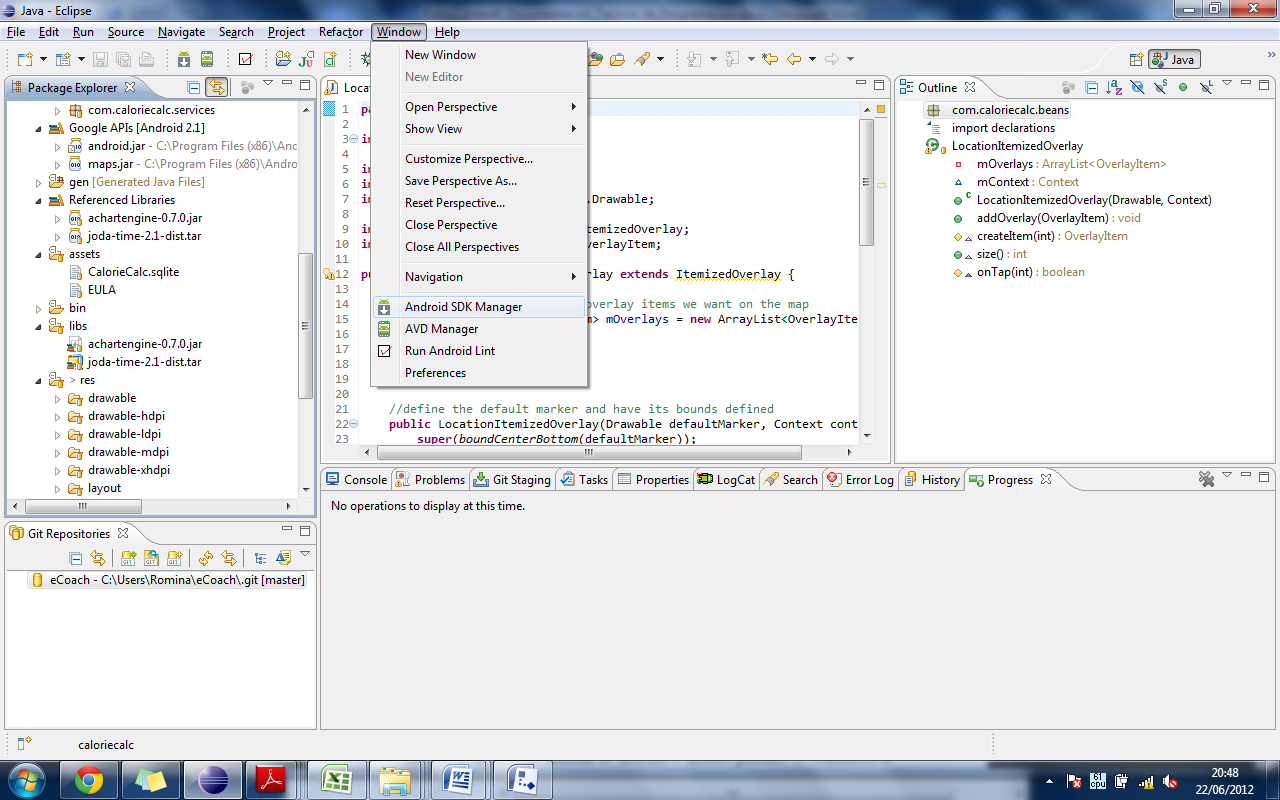


Figura 5. Acceso al Android SDK Manager desde Eclipse

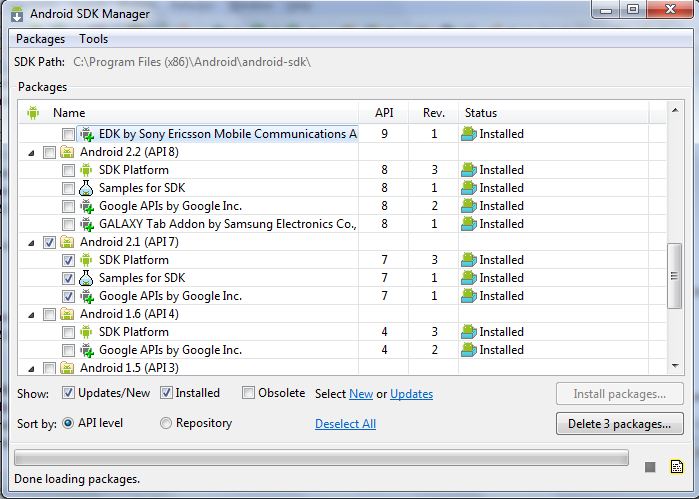
Éste nos permite seleccionar las plataformas de Android que nos interesa incorporar. En nuestro caso en concreto nos basta con tener las últimas versiones de Google APIs [Android 2.1].

Figura 6. Descargar SDK desde Android SDK Manager

## Configurar emulador o terminal físico

Para poder ejecutar la aplicación es necesario contar con una de las dos opciones presentadas a continuación junto con los pasos a seguir para su correcta configuración.

1. Un emulador donde ejecutar la aplicación
   1. Desde eclipse desplegar la pestaña “Window”
   2. Del desplegable seleccionar la opción “AVD Manager”
   3. Clicar “New” para abrir el wizard de configuración de emuladores
   4. Completar el campo nombre con un nombre coherente
   5. Elegir el de las opciones el target correspondiente (Google API 2.1)
   6. Es recomendable contar con suficiente espacio, por lo que se recomienda completar el tamaño de la tarjeta SD con 1024MiB
   7. Para que el terminal arranque más rápido marcar la selección “snapshot enabled”
   8. Las siguientes configuraciones son opcionales, pero se recomienda habilitar “GPS Support”, “SD Card Support” y un “Device RAM size” de por lo menos 1024.
   9. Guardar el emulador.
   10. Desde la pantalla principal de AVD Manager seleccionar “Start” para arrancarlo
2. Un terminal físico conectado al ordenador en modo de desarrollo
   1. Conectar el terminal al ordenador via cable USB
   2. Desde el terminal acceder a “Ajustes”
   3. Navegar a “Aplicaciones”
   4. Seleccionar “Desarrollo”
   5. Marcar la casilla para permitir utilizar el terminal en modo “Depuración USB”, con “Pantalla activa” y permitiendo también aceptar “Ubicaciones simuladas”.

## Generar .apk a partir del código fuente

Desde eclipse, importar el proyecto existente en el espacio de trabajo, el proyecto de eclipse se encuentra incluido en el CD presentado, el nombre del directorio es CalorieCalc.

# Pruebas

Pruebas de stress sobre la aplicación:

<http://developer.android.com/guide/developing/tools/monkey.html>