UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Desarrollo de aplicaciones para dispositivos con Sistema Operativo Android

Proyecto Final De Carrera

Enrique Ramírez Hernández

21/01/2011

Director:
Juan Vicente Oltra Gutiérrez

etsinf

Demos gracias a los hombres y a las mujeres que nos hacen felices, ellos son los encantadores jardineros que hacen florecer a nuestros espíritus.

Índice

1. Objeto y objetivos 3
2. Introducción 4
3. Metodología y Herramientas 6
3.1. Metodología y tecnologías utilizadas 6
3.1.1 Bases de datos 6
3.1.1.1 SQLite 7
3.1.2 XML 7
3.1.1 Java 8
3.2. Herramientas 9
3.2.1 Eclipse 9
3.2.2 SDK Android 10
3.2.3 Emulador 10
3.2.4 Instalación y configuración de las herramientas 10
4. Android SO 15
4.1. Historia 15
4.2. Características 16
4.3. Arquitectura del sistema 18
4.4 Características de una aplicación 21
4.4.1. Estructura de una aplicación 21
4.4.2. Activity 24
4.4.3. Listeners 24
4.4.4. Content provider 24
4.4.5. Hilos 25

5. Implementación de las aplicaciones 26
5.1. Creación de una aplicación 26
5.2. Finalización de una aplicación 27
5.3. Bloc de notas 28
5.4. Reproductor multimedia 31
5.5. Simón Dice 36
5.6. Piano 39
6. Conclusiones 41
8. Bibliografía 43
7. Índice de anexos 45

1. Objeto y objetivos.

El presente documento describe el trabajo realizado en el proyecto: "Desarrollo de aplicaciones para dispositivos con Sistema Operativo Android." El objeto del mismo es la aprobación del proyecto final de carrera para la obtención del título de Ingeniero Técnico en Informática de Gestión, expedido por la Universidad Politécnica de Valencia.

El objetivo principal de este proyecto es aprender cómo funciona este nuevo sistema operativo móvil, así como aprender a manejar todas las herramientas que se ofrecen en la web para la realización de proyectos con Android. Las tecnologías móviles han tenido un auge muy importante durante los 5 últimos años, desde que Apple con su iPhone irrumpiera en el mercado móvil ha cambiado mucho el uso de estas, tras la llegada del dispositivo de la compañía de Macintosh, Google decidió tomar partido en el pastel que había por repartirse, así como Microsoft, que aunque este ya llevaba unos cuantos años en el mercado sus dispositivos estaban completamente obsoletos, desde hace un par de años dispongo de un terminal de los denominados inteligentes (SmartPhone) y he ido dándome cuenta poco a poco del potencial que esta tecnología tan incipiente puede tener de cara al futuro. Fue así como decidí introducirme en el mundo de los desarrolladores de aplicaciones para Android, y lo que finalmente me ha llevado a realizar este proyecto.

A día de hoy no se imparte ninguna asignatura dedicada exclusivamente a la programación de aplicaciones en dispositivos móviles en toda la facultad de informática, así que, ya que me interesa mucho adquirir estos conocimientos de cara a mi futuro laboral decidí ser autodidacta en cuanto a su aprendizaje, y que mejor forma de hacerlo que dedicando mi proyecto final de carrera a ello.

Me gustaría encauzar mi futuro laboral hacia las aplicaciones multimedia móviles

2. Introducción.

Me gustaría destacar que este Proyecto Final de Carrera no está basado en ningún otro proyecto antecesor de la Universidad Politécnica de Valencia, ya que esta tecnología es tan incipiente que apenas se pueden encontrar 2 o 3 proyectos semejantes en toda la universidad.

Con la realización de este proyecto pretendo adentrarme en el funcionamiento de Android SO así como en las posibilidades que este brinda, empezaré analizando analíticamente el sistema hasta la realización de ciertas aplicaciones para ejemplificar ciertas funciones básicas y otras más complicadas.
La estructura básica del proyecto que acontece cuenta de estas partes:

- **En el primer capítulo** describiré las tecnologías usadas para la realización de las aplicaciones, explicando sus ventajas y desventajas a la hora de usar cada una de ellas, así como ciertos ejemplos que hagan mejor su comprensión.

- **El segundo capítulo** me centraré en la parte principal del proyecto, el sistema operativo Android. Partiré de una explicación acerca del origen e inicios para más tarde explicar el tipo de aplicaciones al que está enfocado así como de los recursos que nos ofrece.

  La última parte la dedicaré a un enfoque práctico mostrando las diferentes aplicaciones creadas, detallando las funciones implementadas y explicando con ejemplos del código el funcionamiento de ellas.

  A grandes rasgos podría dividirlas en dos bloques:

  1. **Aplicaciones funcionales,** dentro de esta sección he realizado un bloc de notas que se centra en el uso del

acelerómetro y los menús contextuales del sistema operativo así como de la utilización de una base de datos para el almacenamiento de las notas. También he realizado un reproductor multimedia que hace uso de la parte multimedia del sistema operativo incluyendo imágenes, videos y música.

2. **Juegos,** en este apartado se encuentran las aplicaciones que más tienen que ver con el control de hilos y eventos del sistema operativo, estas aplicaciones visuales son un juego tipo simón dice, y un piano.

- **El último** capítulo corresponde a los anexos de proyecto, donde se encuentra todo el código que he escrito para la realización de las aplicaciones completamente detallado para lograr un entendimiento claro de las mismas así como del funcionamiento de la programación en Android.

3. Metodología y herramientas.

3.1 Metodología y tecnologías utilizadas.

En esta sección no voy a incluir ninguna parte del código detallada, ni siquiera en los ejemplos, porque el uso de las diferentes tecnologías aquí vistas quedará lo suficientemente claro cuando explique el código detalladamente.

La base utilizada en este proyecto es el sistema operativo Android, pero para que los desarrolladores puedan trabajar rápidamente sin aprender nuevos lenguajes de programación Android se apoya en dos pilares fundamentales de la informática actual, XML para formatear los datos, y java como lenguaje de programación, Android utiliza una serie de tecnologías open source para que las aplicaciones en el cobren sentido, las más importantes son tres, Java, XML y SQLite

3.1.1 Bases de datos.

Una base de datos (BBDD) abarca una sección o conjunto de datos que pertenecen a un mismo contexto estos datos están ordenados y almacenados sistemáticamente para usarlos posteriormente. Digitalmente las BBDD se utilizan para diversas funciones, estas pueden ir desde el uso de un simple listado web donde se muestra información acerca de un catálogo de libros, pasando por cualquiera de los foros existentes en la web, hasta entornos profesionales como pueden ser hospitales o trabajos científicos.

Para poder hacer uso de las BBDD se necesita algún gestor de bases de datos (SGBD) estos gestores se utilizan para acceder y almacenar los datos en las BBDD rápidamente. En concreto el que he usado en una de las aplicaciones es SQLite.

3.1.1.1 SQLite.

SQLite es una versión reducida de SQL del inglés (Structured Query Language o Lenguaje de Consulta Estructurado) se trata de un lenguaje de acceso a BBDD considerado declarativo con el cual se pueden especificar muchos tipos de operaciones sobre las mismas.

Su principal característica es el uso del algebra para la realización de consultas así como del cálculo relacional, pudiendo de esta forma recuperar la información o modificarla de manera sencilla.

Un ejemplo de consulta SQLite sería:

```
Select * from Profesores  where year(fecha) = 1975
```

Fig.1 Detalle código SQLite.

En este caso se estarían seleccionando todos los profesores nacidos en el año 1975.

3.1.2 XML.

XML son las siglas en inglés de eXtensible Markup Language (lenguaje de marcas ampliable), este lenguaje es del tipo metalenguaje y es extensible mediante el uso de etiquetas, fue desarrollado por el World Wide Web Consortium. Nació como una forma de reducir SGML usando para ello la parte que permite definir la gramática de lenguajes específicos.
Se podría decir que XML no es un lenguaje en sí, si no una manera de definir multitud de lenguajes para multitud de funciones. De hecho XML no sólo se usa en internet, si no que se ha convertido en un estándar para el intercambio de información estructurada entre diferentes plataformas.

Su uso hoy en día es importantísimo ya que permite que cualquier aparato adaptado a XML se pueda comunicar con otro para el intercambio de información.

Sus características más importantes son que permite separar el código de la presentación del mismo y que es completamente extensible mediante el uso de nuevas etiquetas creadas por el desarrollador.

Un ejemplo básico de XML sería:

```
<?xml version=" 1.0 " encoding=" UTF-8 " standalone= " yes "?>
 <ficha>
      <nombre> Enrique </nombre>
      <apellido> Ramírez </apellido>
     <direccion> Ángel custodio Nº12 </direccion>
 </ficha>
```

Fig.2 Detalle código XML

3.1.3 Java.

Java es un lenguaje de programación orientado a objetos (OO), fue desarrollado por Sun Microsystems en los primeros años de la década de los 90, de alguna forma se podría considerar que java toma una gran parte de su sintaxis de los lenguajes de programación C y C++ aunque en este caso utiliza un modelo de objetos simplificado y elimina las partes más vulnerables de los anteriores como es el acceso a la memoria o la manipulación de punteros, ya que estas partes son las que más suelen llevarnos a errores.

En 2007 Sun Microsystem procedió a la liberación de la casi totalidad de java, así que, ahora Java es software libre, si bien, la biblioteca de clases de Sun requerida para ejecutar el código sí que es propiedad de Sun.

Los cinco principios fundamentales en los que se basa la programación en Java son:

- Utilización de la metodología de la programación OO
- Ejecución de un mismo programa en multitud de plataformas o SO

- Por defecto se debería poder dar soporte para trabajar en red.
- Uso de lo mejor de otros lenguajes como C++

Un ejemplo básico de código Java sería:

```
// Hola.java
public class Hola {
        public static void main(String[] args) {
            System.out.println("Hola, mundo!");
        }
}
```

Fig.3 Detalle código Java.

3.2 Herramientas.

Las herramientas para la realización de aplicaciones en Android más usuales son la plataforma de desarrollo Eclipse y un pluging proporcionado por Google para la confección de proyectos tipo Android. A parte será necesario tener instalada la última máquina virtual de java.

3.2.1 Eclipse.

Se trata de un entorno de desarrollo integrado multiplataforma y de código abierto que permite desarrollar aplicaciones de escritorio, estas aplicaciones son las que se usan fuera del navegador, un ejemplo de aplicación de escritorio podría ser por ejemplo Microsoft office y un ejemplo de aplicación web sería Google docs. Originalmente Eclipse fue desarrollado por IBM aunque actualmente forma parte de una organización independiente llamada fundación Eclipse siendo un programa libre.

Eclipse consta de un editor de texto con resaltado de sintaxis y realiza la compilación del código en tiempo real pudiendo así mostrar los errores en el código instantáneamente. Dispone además de un apartado de plugins para añadir diferentes controles y forma de realizar los proyectos.

3.2.2 Android SDK.

Android SDK es un kit de desarrollo de software (de las siglas en inglés software development kit) para Android consta de un conjunto de herramientas para el desarrollo que permite a los desarrolladores y programadores confeccionar aplicaciones y juegos para el sistema dicho.

Se trata pues de una interfaz de programación de aplicaciones (API application programing interface) que hace uso del lenguaje de programación Java. Permite su ejecución tanto en sistemas Linux como en Windows o Mac.

Incluye además una gran cantidad de ejemplos así como de notas técnicas de soporte y documentación.

3.2.3 Emulador:

Mediante esta función se puede crear un dispositivo móvil casi completamente funcional dentro del sistema operativo, PC o Mac, para el desarrollo de las aplicaciones.

El emulador de Android está contenido dentro del paquete del SDK, se trata de una herramienta creada por Google para poder probar las aplicaciones móviles sin necesidad de instalarlas en un dispositivo, mediante el proceso de creación de un proyecto en Eclipse haré uso de uno de los terminales emulados creados para probar en cada momento las nuevas funciones o modificaciones de las aplicaciones.

3.2.4 Instalación y configuración de las herramientas.

Los puntos a seguir son:

- La primera parte consistirá en instalar Eclipse, en este caso yo he usado la versión 3.5, llamada Galileo, bajo el sistema operativo Windows XP. La página de descarga es: www.eclipse.org/downloads/
- La versión que he utilizado es exactamente: Eclipse IDE for Java EE Developers(189 MB) aunque también es igual de válida Eclipse IDE for Java Developers (92 MB)

- Después se descomprime en la carpeta en la cual queramos contenerlo, y se ejecuta ecplise.exe. Lo primero que aparece es una pantalla en la que pregunta donde quieres que se cree el directorio de trabajo de Eclipse, donde posteriormente se almacenarán los proyectos que se creen.

Fig. 4 Localización Workspace en Eclipse.

- Después de la carga y de que aparezca la pantalla principal hay que ir a Help > Install New Softare. En la pantalla que aparece hay que presionar el botón Add....
- En la nueva ventana elegimos cualquier nombre, por ejemplo, Android Plugin y añadimos en el campo Location la URL: https://dl-ssl.google.com/android/eclipse/ presionamos ok, y después next hasta que nos pida aceptar los términos de la licencia, los aceptamos y el programa empezará la descarga el plugin.

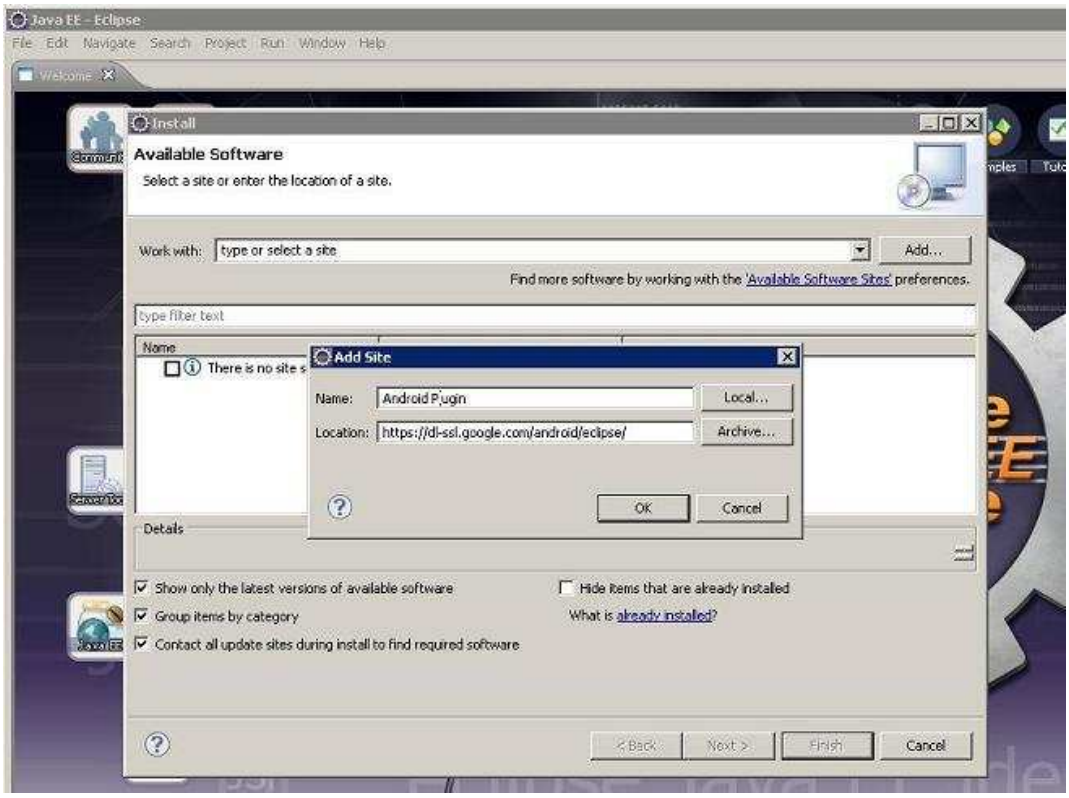

Fig. 5 Instalación del plugin de Android.

- Por último reiniciamos Eclipse y ya tenemos instalado el plugin, aunque la instalación del SDK aun no ha terminado.
- Ahora volviendo a Eclipse hay que seleccionar **Window** > **Preferences,** en el panel de la izquierda que aparecerá hay que seleccionar Android. Donde pone SDK location hay que poner la localización del SDK de Android en el disco duro en el que la tengamos, la descarga del SDK se realiza desde:

  http://developer.android.com/sdk/index.html Actualmente la versión disponible del mismo es la Android 2.3 Platform aunque mis aplicaciones están basadas en la 2.2 puesto que la versión 2.3 no está disponible para mi terminal, no hay ningún problema en descargar la última versión, ya que más tarde cuando explique la parte del emulador, se verá que es muy simple especificar cuál será la versión a usar de Android.

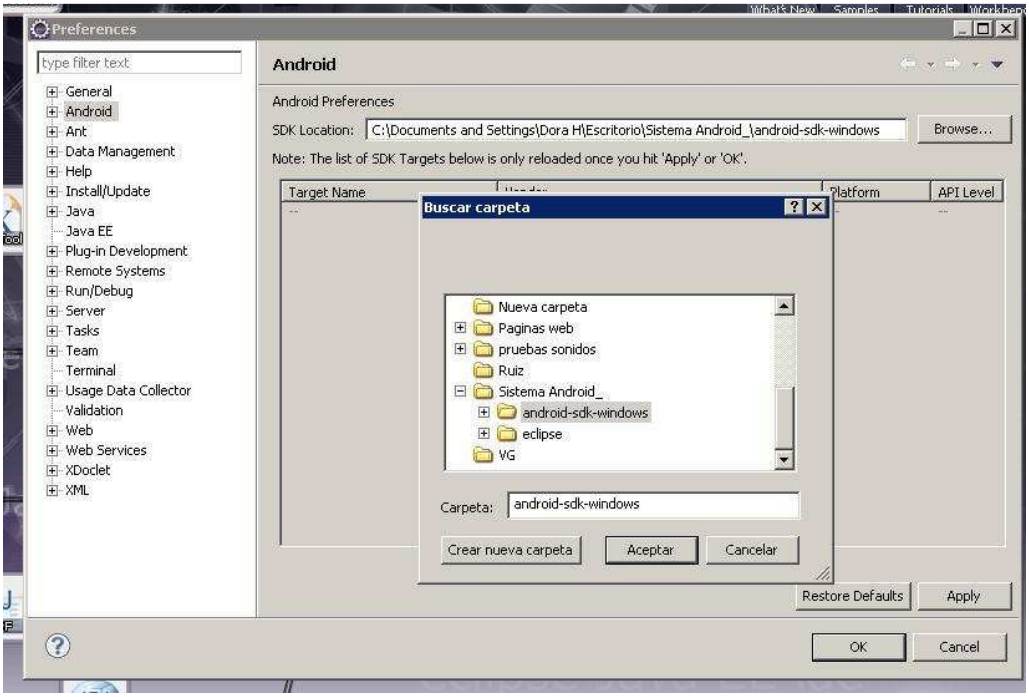

Fig. 6 Localización del SDK de Android.

- Este es el último paso para tener la instalación completada, y se trata de la creación de un emulador Android. Para crearlos simplemente mediante un terminal en Linux y Mac o una ventana de comandos en Windows navegamos hasta la carpeta tool del sdk y ejecutamos: Android create avd --target 8 --name EMULADOR

Al decirle en la creación “target 8” le estoy diciendo que use la versión 2.2 de Android.

Fig.7 Emulador Android.

4.Android SO.

De un tiempo a esta parte los usuarios de móviles han ido requiriendo más opciones para sus terminales y han ido exprimiendo las que estos ofrecían. Las compañías fabricantes de teléfonos ofrecen día a día mejores terminales, llegando al punto de que estos son capaces de realizar muchas funciones destinadas, antiguamente, exclusivamente para PC's. Google no ha desaprovechado la oportunidad de realizar junto con su gran equipo de desarrolladores un sistema operativo moderno y a la altura de las circunstancias.

Se trata de un SO móvil basado en Linux orientado a teléfonos y tabletas.

Fig. 8

4.1 Historia.

Fue desarrollado por Android inc, empresa comprada por Google en 2005. Inicialmente el sistema se desarrolló exclusivamente por Google pero más tarde pasó a formar parte de la Open Handset Alliance, la cual está liderada por Google. Una de las ventajas que contiene el SO es que permite que personas ajenas a Google desarrollen aplicaciones, como es mi propio caso, así consiguen que las aplicaciones basadas en el sistema sean mucho mayores que las de los sistemas cerrados, y esto repercute en la venta de terminales con Android SO.

El primer teléfono que salió al mercado con este dispositivo fue el HTC

G1 o Dream lanzado originalmente en EEUU a finales de 2008, venía con la versión Android 1.0 instalada y rápidamente se convirtió en un éxito de ventas, actualmente la versión 2.3 poco tiene que ver ya con el anticuado sistema 1.0

4.2 Características.

Las características más relevantes son:

- Diseño de dispositivo, Android permite la utilización de casi cualquier resolución móvil disponible hasta día de hoy lo que hace el sistema adaptable desde pantallas de 2,5" hasta tabletas de 10" Además permite el uso de librerías de gráficos en 2D y 3D.

- Almacenamiento, se realiza mediante SQLite como ya he mencionado anteriormente.

- Conectividad Soporte para estas redes de conectividad inalámbrica: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, y WiMAX.

- Mensajería, tanto las formas tradicionales de mensajería en España como son SMS y MMS como los correos electrónicos con sistema push son bien recibidos en Android.

- Navegador web, basado en el motor WebKit, usado por ejemplo en Chrome o Safari y mediante el motor de JavaScript V8 el navegador nativo de Android es uno de los mejores a nivel de SO móviles.

- Soporte de Java, realmente Android no dispone de una maquina virtual de java como tal, debido a que los dispositivos móviles son

de escasos recursos así que se apoya en la maquina virtual de Dalvik para ejecutar el código java.

- Soporte multimedia, Android soporta la mayoría de los formatos multimedia más relevantes de hoy en día, en concreto: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, BMP.61

- Soporte para streaming, El soporte para streaming de descarga en HTML se hace de forma nativa, pero el soporte para adobe flash se hace mediante una aplicación que proporciona Adobe, aunque lo cierto es que son los dispositivos actuales los que está empezando a ejecutar aplicaciones o visualizaciones en Adobe Flash, los terminales con una año de antigüedad no lo soportan.

- Soporte para hardware adicional, el sistema soporta una infinidad de software externo al funcionamiento tradicional de un dispositivo que sirve para hacer llamadas, entre otros encontramos: soporte para cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, magnetómetros, giroscopios, aceleración 2d y 3d, sensores de proximidad y de presión, termómetro.

- Entorno de desarrollo, como ya he explicado anteriormente tiene un gran entorno de desarrollo basado en Eclipse y en el emulador suministrado por Google.

- Market, Dispone de un mercado de aplicaciones (Android Market) permitiendo de este modo descargar e instalar aplicaciones sin necesidad de un PC. Actualmente este mercado sobrepasa las 100.000 aplicaciones y sigue aumentando rápidamente.

- Multi-táctil, se incluyen las funciones multitáctiles de forma nativa para todos los dispositivos Android que dispongan de una pantalla con esta capacidad.

- Bluetooth, la versión 2.2 ya da soporte para A2DF y AVRCP, envío de archivos (OPP), exploración del directorio telefónico, marcado por voz y envío de contactos entre teléfonos.

- Videollamada, a partir de la versión 2.3 lo hace de manera nativa, los dispositivos que ofrecen una cámara frontal como el Samsung Galaxy S y esta versión del sistema operativo ya pueden realizarlas sobre IP.

- Multitarea, para todas las aplicaciones y desde su primera versión.

- Tethering, permite al teléfono ser usado como punto de acceso WIFI, está disponible a partir de la versión 2.2

4.3 Arquitectura del sistema.

Desglosando sus partes más importantes, encontramos:

- **Aplicaciones**: Donde se encuentran todas las instaladas en el sistema, algunas de las aplicaciones base son: un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos. Como ya he mencionado están escritas completamente en Java.

- **Marco de trabajo de aplicaciones**, o Framework de aplicaciones o armazón de aplicaciones, cualquier desarrollador puede acceder a ellos para la realización de sus aplicaciones,

está diseñada para simplificar la reutilización de los componentes del sistema.

- **Bibliotecas**: Se incluyen una gran variedad de bibliotecas de C/C++ usadas por ciertas partes sistema. Los desarrolladores pueden hacer uso de ellas, las más importantes son: System C library, bibliotecas de medios, bibliotecas de gráficos, 3D, SQLite…

- **Runtime de Android**: Este incluye una recopilación de bibliotecas base que proporcionan casi la totalidad de las funciones disponibles en las bibliotecas de Java. Cada una de las aplicaciones que se estén ejecutando utilizan su propio proceso con una llamada a la máquina virtual Dalvik. El diseño de ésta máquina permite que se ejecuten concurrentemente varias de ellas sin que haya ningún problema en el dispositivo.

En la siguiente página se puede encontrar un diagrama de estas partes**.**

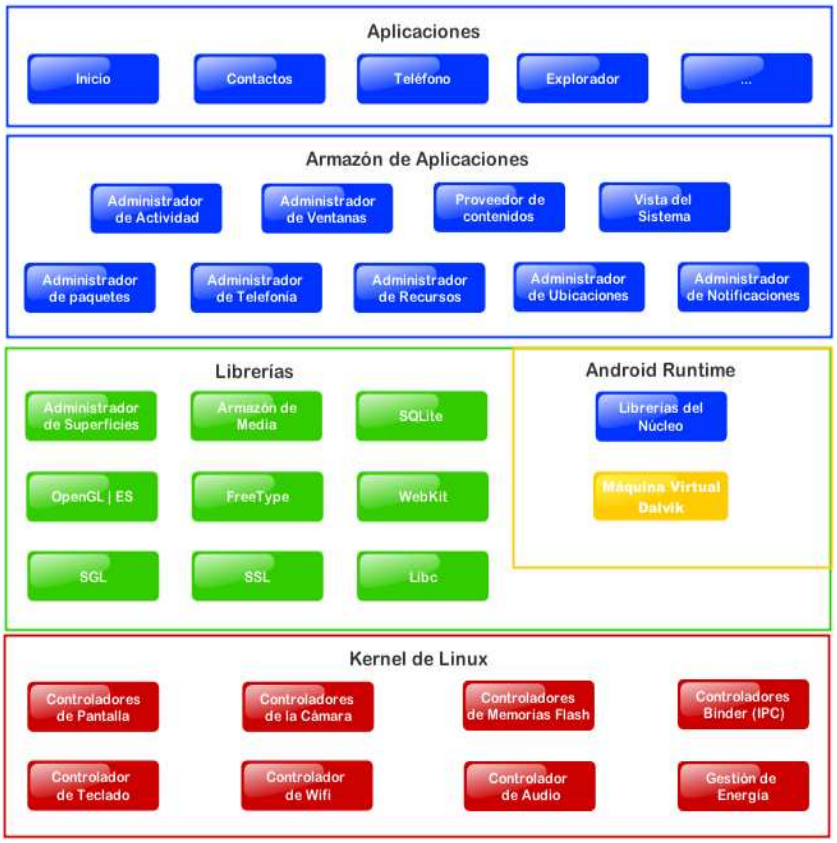

Fig.9 Diagrama de la arquitectura del SO Android.

- **Núcleo Linux**: El núcleo de Android está basado en Linux por lo tanto los servicios base del sistema, la seguridad, la gestión de memoria, la gestión de procesos, la pila de red, y el modelo de controladores está gestionado por las librerías del núcleo de Linux.

4.4 Características de una aplicación.

Para la creación de una aplicación Android Google a creado una estructura la cual voy a detallar a continuación, además dentro de esta existe cuatro componentes muy importantes Activities, Listeners, servicios y Content Provider.

4.4.1 Estructura de una aplicación.

La composición para gestionar la creación de una aplicación Android está dividida en una serie de carpetas y archivos a los que se puede acceder directamente desde el proyecto creado en Eclipse para cada una de las aplicaciones creadas.

Estas carpetas o archivos son:

- **src**, esta carpeta es una de las más importantes, aunque todas son imprescindibles, esta carpeta contiene el código fuente. Aquí es donde se almacenan todas las clases que vamos a desarrollar durante la creación de la aplicación. Como mínimo, aquí tendremos una clase principal que hereda de Activity y sirve para la creación de la aplicación.
- **res**, Aquí se encuentran los archivos necesarios para la visualización correcta de la aplicación, así como los sonidos que se van a reproducir en la misma. A su vez contiene varias carpetas que son:

    - drawable: En esta carpeta están contenidas las imágenes que se usarán. En ella encontraremos como mínimo icon.jpg. que es el icono de la aplicación.

    - layout: Donde encontramos las capas necesarias para mostrar la información, esta información se guarda en formato XML, cuando explique los ejemplos diré que se encuentra dentro de esos XML. Ya que Android es un SO visual nos permite directamente crear

las capas en modo visionado, colocando los objetos directamente en la pantalla donde el desarrollador quiera. A mí, particularmente, me gusta hacerlo "a mano" directamente sobre el código, Android permite también hacerlo de este modo presionando la pestaña de XML. Como mínimo encontraremos una capa que suele ser main.xml la cual nos sirve para mostrar la primera información de la aplicación.

- values: Esta carpeta contiene los valores de los string que vamos a mostrar por pantalla, esta carpeta es especialmente importante para la realización de aplicaciones multilingües puesto que en vez de rehacer el código, creando varios archivos XML con los diferentes idiomas podemos cargar la aplicación en un idioma o en otro. Como mínimo encontraremos un archivo llamado sring.xml donde encontraremos el nombre de la aplicación.

- raw: Esta no es necesario que se encuentre en el proyecto, sirve para almacenar el resto de ficheros, como pueden ser los efectos sonoros.

- **Android 1.6 o 2.2**: Esta carpeta contiene todos los imports necesarios para la ejecución de las aplicaciones, así como los que se añaden directamente en las cabeceras de las clases.

- **gen**, Lo que encontramos en esta carpeta es un archivo llamado R.java autogenerado por el proyecto que se encarga de gestionar toda la información contenida en res para poder ser utilizada en src, por ejemplo si tenemos una imagen en

/res/drawable la cual queremos usar en una clase, la manera de obtenerla sería así:
context.getResources().getDrawable(R.drawable.rojo);.

- **AndroidManifiest.xml,** En este archivo se encuentra toda la información esencial para poder ejecutar la aplicación. Tal como el nombre de la aplicación, el icono de esta y la primera actividad a ser lanzada.

  Ejemplo para la aplicación sonidos (Simón Dice) posteriormente explicada:

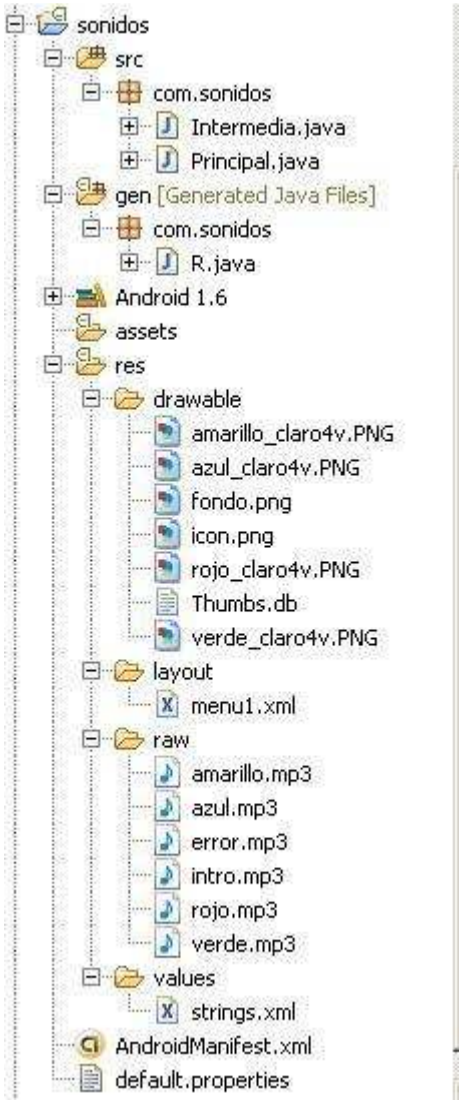

Fig.10 Detalle Eclipse.

4.4.2 Activity.

Son un elemento muy común en Android, hace falta una clase por cada actividad que se quiera implementar, ésta extenderá de una clase base llamada Activity. Para mostrar la interfaz por pantalla se apoya en vitas o View, para mostrar cada una de estas vistas es necesario una clase que herede de Activity por cada una, en el código de las aplicaciones se puede ver claramente como ocurre esto, también está explicado detalladamente.

Mientras estamos en una vista la anterior queda pausada para poder volver a ella rápidamente si fuera necesario pulsando solamente el botón atrás.

La forma en que se comunican las vistas es mediante Intent, dentro de estos Intent se puede almacenar cualquier tipo de información y en la cantidad que sea necesaria para pasársela a la siguiente vista, por ejemplo, en la aplicación reproductor paso la lista de imágenes a la clase que las visualiza para tenerlas todas juntas.

4.4.3 Listeners.

Son muy importantes para el desarrollo de aplicaciones móviles, sobre todo para los que incluyen pantalla táctil, los Listeners se usan para que cuando pase algo en el código que está descrito (un evento) anteriormente, reaccionen mediante una acción determinada, por ejemplo si el Listener en cuestión estuviera “escuchando” cuando se pulsa el botón “OK” cuando sea pulsado realizará las funciones que se le haya dicho que ejecute.

4.4.4 Content provider.

Cuando se trata de guardar información en este sistema operativo se hace mediante bases de datos, pero cuando lo que se quiere es compartir la información para que otras aplicaciones la usen, se utilizan los content provider, esta clase tiene unos métodos

estándar que hace que las aplicaciones puedan consultar, guardar, o modificar la información general de las aplicaciones.

4.4.5 Hilos.

La creación de hilos por el sistema operativo Android está restringida para aumentar la estabilidad del sistema, esto quiere decir que el sistema solo creará un hilo por aplicación que se esté ejecutando, pero permite a los desarrolladores crear todos los hilos que crea necesarios para la ejecución de su aplicación.

5. Implementación de las aplicaciones.

5.1 Creación de una aplicación.

Lo primero que hay que hacer es crear una nueva aplicación Android en el entorno Eclipse para poder comprobar su funcionamiento, para ello, hay que pulsar **File > New > Project** ahora aparecerá una serie de carpetas con todos los tipos de proyectos que se pueden crear en Eclipse, hay que seleccionar la que corresponde a Android (Aparecerá solo si la instalación anterior se ha realizado correctamente) hay que seleccionar Android Project.

En la nueva ventana que aparece hay que proporcionar todos los datos referentes a la aplicación.

- **Project name:** Nombre del proyecto, p.e. HolaMundo
- **Application name:** Nombre de la aplicación, p.e. Hola mundo
- **Package name:** Nombre del package, puede contener una a más aplicaciones, p.e. com.HolaMundo
- **Create Activity:** Nombre de la actividad principal que se creará dentro del proyecto, p.e. HolaMundo
- **Min SDK Versión:** Versión del SDK que se utilizará, 4 para la 1.6, 8 para la 2.2

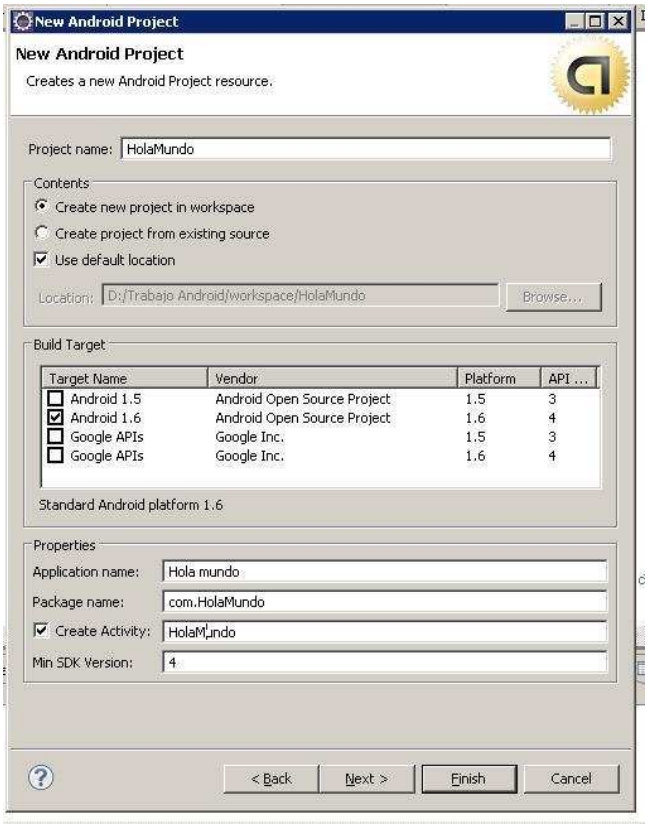

Fig.11 Detalle de la creación de un nuevo proyecto.

Al presionar Finish se creará el nuevo proyecto.

5.2 Finalización de una aplicación.

Una vez tenemos realizado todo el proyecto se procederá a exportarlo. Haciendo clic en el botón derecho encima del nombre del proyecto, entre otras opciones, aparecerá Export, hay que exportarlo como Android Applicattion, el programa solicitará que se cree una contraseña para la posterior creación de aplicaciones mediante el mismo usuario y una caducidad para saber cuándo dejará de funcionar la aplicación. Finalmente obtendremos un archivo file.apk el cual copiándolo en cualquier dispositivo con SO Android nos permitirá instalar y ejecutar la aplicación.

5.3 BLOC DE NOTAS.

Después de haber realizado una gran cantidad de ejemplos y pequeños tutoriales para aprender el manejo básico de las actividades, así como la composición descrita anteriormente, decidí hacer uno de los tutoriales extensos propuestos por Google en la página oficial de Android, este trata sobre la composición de un bloc de notas, el link directo a él es:

http://developer.android.com/guide/tutorials/notepad/index.html una vez finalizado me di cuenta de que ese bloc de notas, en sí, era prácticamente inútil y decidí que mi primera aplicación propia para Android sería un bloc de notas completamente funcional, así que partiendo de esa base empecé a añadirle ciertas funciones que al final dieron como fruto la aplicación que a continuación voy a explicar detenidamente.

Esta aplicación tiene como pantalla principal una lista con todas las notas guardadas hasta el momento las cuales se almacenan en una base de datos del tipo SQLite, de cada una de ellas se muestra su nombre y los 30 primeros caracteres del cuerpo, si los tuviera, si tiene menos, se muestran todos.

Cuando tiene más de 30, después de estos 30 primeros aparecerá "(…)" para indicar que la nota continua, y que si se hace clic en ella se podrá leer en su totalidad.

Foto detalle de la aplicación en la página siguiente:

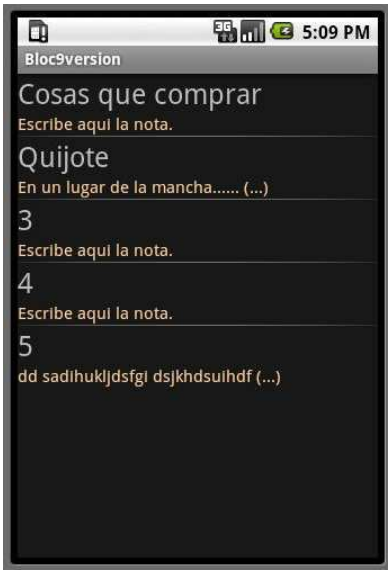

Fig.12 Detalle de la lista de notas.

Presionando la tecla menú se accede a 3 opciones, que son:

Borrar todas:

Si se presiona esta opción aparecerá un mensaje de alerta avisando que se van a borrar todas las notas, si se acepta se borraran.

Salir de la aplicación:

Sirve para finalizar el programa.

Añadir nota:

Si se pulsa sobre esta opción pasaremos a otra capa y a otra clase en la cual aparece una nota con el título: Titulo y el texto del cuerpo: Escriba aquí la nota. El cursor se sitúa sobre el titulo, y al pasar al cuerpo el texto anterior se borrará para dejar al usuario escribir el nuevo. Para guardarla hay que presionar menú y usar la opción de "Guardar". En el caso de que el usuario presionara la opción de salir sin haber guardado la aplicación lanza un mensaje de alerta informando sobre ello, y permite guardar la nota antes de salir.

Fig.13. Detalle de las opciones del menú.

En el menú principal, donde se encuentra la lista de notas, si se presiona sobre cualquier nota de la lista se accede a su contenido en modo visionado, en el caso de que se quisiera editar se presiona menú y se elige la opción Editar nota.

Esto es así porque la aplicación dispone de un sistema en modo visionado, este sistema permite navegar por las diferentes notas simplemente moviendo el móvil, si se mueve hacia la izquierda retrocederá una nota y si se mueve a la derecha avanzará una. Para llevar a cabo esta función he usado el acelerómetro del que dispone mi terminal y la mayoría de los dispositivos Android.

5.4 Reproductor Multimedia.

Esta aplicación consta de 3 partes muy diferenciadas, estas son: la que se encarga de reproducir la música, la de la reproducción de vídeo, y por último la galería de imágenes.

La aplicación parte de una capa principal en la que deja al usuario seleccionar una de las tres características multimedia de las que posee.

Fig.14 Pantalla principal del reproductor.

Al seleccionar cualquiera de estos tres botones la aplicación cambiará a una clase diferente mediante el uso de Intent y este en su método onCreate lo primero que hará es cambiar la vista para que se muestre una de las tres capas secundarias que existen en la aplicación.

En el caso de que la opción seleccionada fuera la de Música, se mostraría un listado de carpetas del directorio raíz, así como una lista de canciones si es que las hubiere en ese directorio, todo esto separado mediante una línea de puntos para diferenciar entre canciones y directorios, además los directorios están desplazados hacia el centro de la pantalla y se muestran entre corchetes, p.e. [El mago de Oz] , también se encuentran en esta pantalla los botones de reproducción, en la parte inferior, y en la parte superior hay un botón para retroceder en el árbol de directorios, así como la ruta actual de en que carpeta se encuentra el usuario.

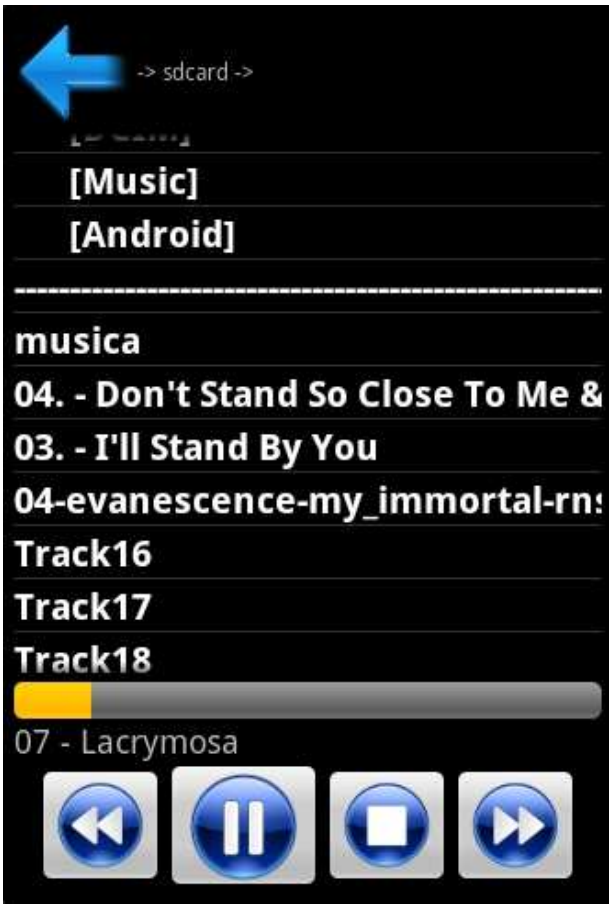

Fig.15 Apartado musical del reproductor.

Cuando se accede por primera vez, al pulsar cualquiera de los botones inferiores no se reproducirá ningún sonido, ya que no hay

cargada ninguna lista de reproducción, en el momento de que el usuario presione una canción todas las que se encuentran en ese directorio pasarán a formar parte de la lista de reproducción, por lo tanto si el usuario presiona avanzar o retroceder una canción sonará la siguiente o la anterior respectivamente.

Además ésta capa de la aplicación también dispone de una barra de progreso para indicarle al usuario el punto de la canción en el que se encuentra en todo momento.

El usuario puede seguir navegando por el árbol de directorios mientras escucha una canción sin ningún tipo de problema.

Si la elección del usuario fuera la visualización de un vídeo se mostrará una nueva capa prácticamente idéntica a la del apartado de música aunque esta vez no existe ningún botón inferior, solo una lista con todos los videos y carpetas cargados del directorio en el que nos encontremos, así como la aparición en la parte superior de un botón para retroceder y la ruta actual por donde se encuentra navegando el usuario.

La visualización de los vídeos se hace mediante una nueva capa, al presionar uno de estos, pasa el control a esta nueva capa donde se muestra el vídeo y dos botones, uno para pausarlo y otro para pararlo, si el vídeo fuera parado automáticamente la aplicación devuelve el control a la capa anterior volviendo a mostrarla tal cual estaba, también aparecerá una barra de progreso que como en el caso anterior mostrará la progresión del video.

Fig.16 Reproducción de un vídeo.

Y por último la tercera opción es la de la visualización de la galería, en este caso al presionar el botón imágenes se accederá directamente a la caché de las imágenes que el dispositivo guarda en todo momento, Android realiza un escáner de medios cada vez que se realizan cambios en la memoria externa, analiza toda la memoria en busca de nuevo material multimedia, he querido para esta última opción valerme de esta posibilidad que brinda el dispositivo, así pues, al acceder a esta capa lo primero que se muestra es un cuadrícula con todas las imágenes que el sistema operativo ha encontrado y tiene almacenadas en su caché, es decir, todas las imágenes de la memoria externa.

Realmente lo que se muestra no es la imagen completa ya que el sistema tardaría mucho en mostrar las miniaturas de las imágenes, así que se vale de las imágenes ya reducidas guardadas en la caché, a estas imágenes se les llama thumbnails.

Al seleccionar cualquiera de esas imágenes almacenadas en la caché, se pasará a una nueva capa donde se mostrará la imagen seleccionada, y además se permitirá al usuario recorrer todas las imágenes pulsado la pantalla hacia la izquierda o hacia la derecha para avanzar o retroceder en la lista.

Esto se consigue copiando todas las imágenes en una lista mediante la referencia dada por el thumbnails, así como el número de la imagen actual.

Fig.17 Cuadrícula de imágenes.

5.5 SIMON DICE:

En este juego, muy popular en los años 90, lo que se hace es ir creando una progresión de combinaciones de colores para que el jugador las vaya pulsando, la primera vez que se ejecuta, la aplicación ilumina un color y reproduce un sonido, seguidamente espera a que el usuario presione el mismo color, si es así pasa a reproducir ese y acto seguido uno más, si el usuario los presiona adecuadamente, la aplicación mostrará otro mas y así sucesivamente hasta que el jugador falle.

La primera pantalla que nos encontramos es un mensaje de bienvenida, y al pulsar empezar se muestran las instrucciones del juego.

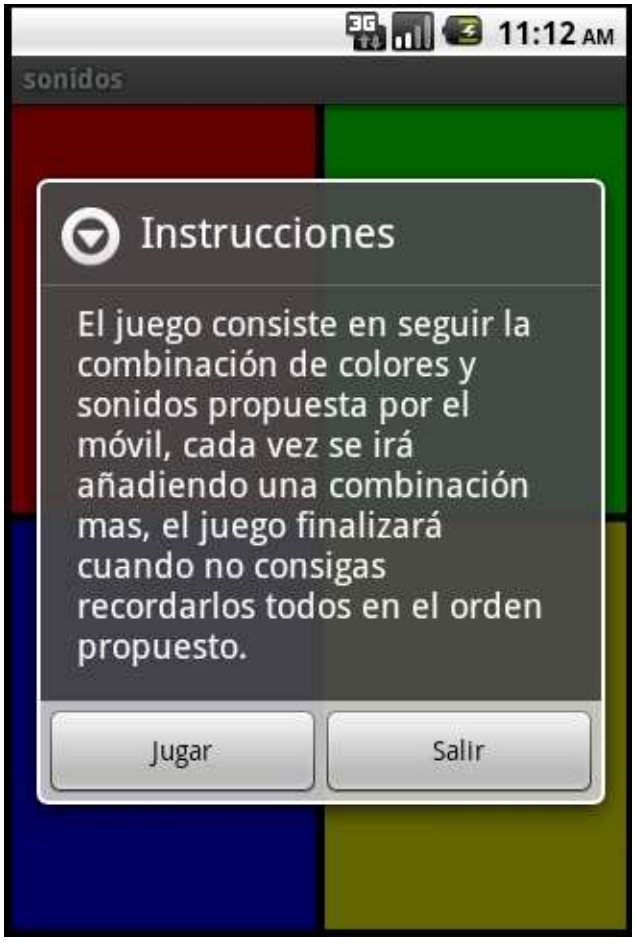

Fig.18 Instrucciones Simón Dice.

Después al pulsar sobre jugar, el juego, empieza reproduciendo un sonido he iluminando una tecla, el jugador podrá salir de la aplicación en cualquier momento pulsando la tecla menú y seleccionando la tecla de Salir. Al finalizar la partida se mostrará una pantalla con la puntuación obtenida y preguntándole al jugador si desea empezar otra vez.

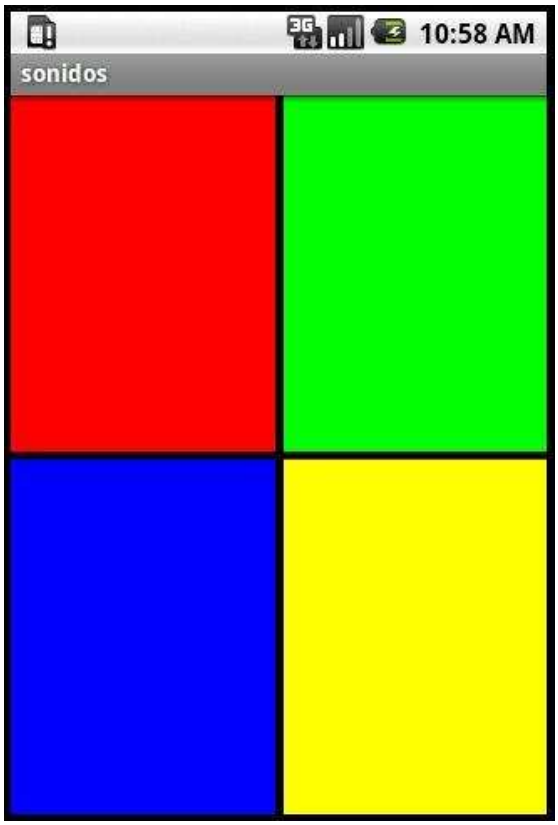

Fig.19 Detalle juego Simón Dice.

A continuación explicaré detenidamente que es lo que realiza cada parte del código, la aplicación está compuesta de dos clases, podría haberse realizado en una sola si no fuera porque las imágenes en Android no se actualizan hasta la finalización de los métodos, y no podía mostrar una sucesión de cambios a no ser que creara un hilo para mostrarlos.

Debido a esto, al volver a la aplicación hay que actualizar la pantalla mediante el método onDraw, este método no se encuentra

en la clase Activity, y la clase por la que empieza el código en Android siempre tiene que heredar de Activity, así que después de crear la primera clase activity automáticamente hay que pasar el control a la nueva clase que hereda de View, esta ultima sí que contiene el método onDraw.

5.6 Piano.

La realización de esta aplicación me ha resultado la más fácil de todas, ya que la he realizado en último lugar, y después de todos los problemas que tuve con las anteriores ya había aprendido los conceptos básicos y algunos avanzados del funcionamiento de Android.

Fig.20 Detalle Piano

Así pues para la realización de este piano me he basado en la utilización de botones, sonidos y eventos, con una ligera diferencia, en vez de manejar los eventos en sí, esta vez he manejado las ocurrencias en la pantalla, para ello en vez de utilizar el método onClick, he usado el método onTouch, lo que he conseguido con ello es saber cuando la pantalla es pulsada y cuando deja de serlo, así cuando una tecla del piano es pulsada cambia de color, y hasta que no se suelta no vuelve a cambiar al sonido original, produciendo el efecto de que la nota está bajando como si de un piano de verdad se tratara.

Las llamadas a los eventos, por lo tanto, también han cambiado, ahora en vez de usar los métodos que “Escuchan” los eventos ocurridos en la pantalla he usado los que “escuchan” cuando la pantalla los ha tocado, y por lo tanto cuando los ha dejado de tocar, un ejemplo de uno de ellos sería, sol.setOnTouchListener(this);

6. Conclusiones.

El futuro de Android está asegurado. En el mes de febrero de 2011 se realizará el congreso mundial de móviles en Barcelona, y casi todos los fabricantes de móviles presentarán terminales con este sistema operativo, demostrando que se ha convertido en su preferencia y produciendo que a corto plazo sea el primer sistema mundial en número de terminales.

Con la realización de este proyecto he conseguido un aporte de conocimientos, extra en lo que a aplicaciones móviles se refiere, al que se imparte en mi facultad, he aprendido a manejar ciertas tecnologías que desconocía por completo así como el uso de herramientas gratuitas con un gran potencial, como es el caso de Eclipse. Gracias al abanico de posibilidades que brinda Android y mis conocimientos adquiridos en la carrera he conseguido realizar 4 aplicaciones, a mi parecer vistosas, y que me han servido para introducirme en este gran mercado emergente.

Valoración personal:

Considero que se han cumplido mis expectativas iniciales a la hora de desarrollar este proyecto, y creo que ha sido así sobre todo porque ya lo tenía instalado en casa y había estado probándolo de algún modo, así que cuando hice la propuesta ya sabía hasta donde podría llegar o hasta donde no.

Después de haber pasado los últimos 4 meses realizando este proyecto, puedo concluir que el potencial del SDK de Google para Android es inmenso, las aplicaciones son muy fáciles de crear, y la ayuda que proporciona Eclipse es esencial para avanzar rápidamente en la creación.

Ahora que ya tengo acabado el proyecto voy a informarme hasta qué punto el juego de simón dice tiene copyright y en el caso de que no me acarreara ningún problema me gustaría colgar la aplicación en la tienda de aplicaciones de Android (Android Market) para que todo el

que quiera pueda disfrutar de ella.

El único aspecto negativo que puedo hacer al respecto es que todavía existe poca información en castellano, si bien, puedo decir que cuando en algún momento no puede avanzar en la creación de alguna de las 4 aplicaciones, escribí en un foro sobre Android llamado Android-spa.com mis problemas, y la gente interesada en este sistema me ayudó a solucionarlos.

Concluyo pues que la realización de este proyecto me ha resultado muy satisfactoria, creo que el futuro de los pequeños desarrolladores, los que ni tienen nombre ni forman parte de ninguna empresa, pasa por la creación de aplicaciones para dispositivos móviles, ya que la inversión económica del desarrollador es mínima y se pueden obtener unos jugosos beneficios colocándola en el mercado de aplicaciones de Android.

Líneas futuras:

A partir de este proyecto se abren varias líneas de investigación o desarrollo, podría servir para aprender cómo realizar aplicaciones y empezar el desarrollo de una gran aplicación con unas bases establecidas muy extensas o bien se podría continuar alguna de las aplicaciones propuestas, el reproductor multimedia es muy funcional, pero dentro de esa funcionalidad se podrían añadir nuevos apartados sin que la simplicidad desaparezca, en cuanto a la aplicación del piano todavía se le podría sacar mucho jugo implementando un juego basado en seguir una melodía con diferentes niveles.

7. Bibliografía.

Wikipedia en español (2002). En línea. Internet. Enero del 2011. Disponible: http://es.wikipedia.org

Wikipedia en inglés (2001). En línea. Internet. EEUU. Enero del 2011. Disponible: http://en.wikipedia.org

Sitio web oficial desarrolladores Android (2008). En línea. Internet. EEUU. Octubre del 2010. Disponible: http://developer.android.com/

Foro oficial Android (2008). En línea. Internet. EEUU. Octubre del 2010. Disponible: http://developer.android.com/community/index.html

Foro Android-SPA (2009). En línea. Internet. Noviembre 2010. Disponible: www.android-spa.com

Foro And.roid (2010). En línea. Internet. Noviembre 2010. Disponible: http://and.roid.es/foro/

Foro HTCMania, Programación y Desarrollo para Android: (2007). En línea. Internet. Diciembre 2010. Disponible: www.htcmania.com/forumdisplay.php?f=153

Eclipse (2004). En línea. Internet. Noviembre 2010. Disponible: www.eclipse.org/

Java (1999 – 2000). En línea. Internet. Noviembre 2010. Disponible: www.java.com

Programación en castellano, java (2007). En línea. Internet. Diciembre 2010. Disponible: www.programacion.com/java

Java hispano (2009). En línea. Diciembre 2010. Disponible: www.javahispano.org

Sitio web oficial de SQLite (2000). En línea. Internet. Noviembre 2010. Disponible: www.sqlite.org/

Manual SQLite (2001). En línea. Internet. Noviembre 2010. Disponible: http://php.net/manual/es/book.sqlite.php

Buscador Google: (1998). En línea. Internet. Enero 2010. Disponible: www.google.es

Sitio web oficial XML (2006). En línea. Internet. Diciembre 2010. Disponible: www.w3.org/XML/

8. Índice de anexos.

Anexo 1. Clase BlocDeNotas.java

Anexo 2. Listar_notas.xml

Anexo 3. EditarNotas.java

Anexo 4. Editar_notas.xml

Anexo 5. Clase AdaptadorBD.java

Anexo 6. Clase CuadriculaImagenes.java

Anexo 7. Clase Imagenes.java

Anexo 8. Clase ListaVideos.java

Anexo 9. Clase Musica.java

Anexo 10. Clase Reproductor.java

Anexo 11. Clase Video.java

Anexo 12. Cuadricula_imagen.xml

Anexo 13. Fila.xml

Anexo 14. Imagen.xml

Anexo 15. Lista_videos.xml

Anexo 16. Main.xml

Anexo 17. Musica.xml

Anexo 18. Video.xml

Anexo 19. AndroidManifest.xml

Anexo 20. Clase Intermedia.java

Anexo 21. Main.xml

Anexo 22. Principal.java

Anexo 23. Clase Piano.java

Anexo 24. Main.xml

Anexo 25. AndroidManifest.xml

Anexo 26. Índice de figuras

Anexo 27. Extracto del tutorial notepad

Anexo 1

Clase BlocDeNotas.java

Es la clase principal, de esta, parte el programa y usa listar_notas.xml como capa para mostrar la lista de las notas. El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.android.BlocDeNotas;
```

Importaciones:

```
import com.android.BlocDeNotas.R;

import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.AdapterView.AdapterContextMenuInfo;
```

Inicio de la clase:

```
public class BlocDeNotas extends ListActivity{
```

Declaración de atributos globales:

```
    //Enteros finales con los valores que se pasarán a las
    nuevas actividades cuando sean creadas.
    private static final int ACTIVITY_CREAR=0;
    private static final int ACTIVITY_EDITAR=1;

    //Enteros finales para la creación del menú.
    private static final int INSERTAR_ID = Menu.FIRST;
    private static final int BORRAR_ID = Menu.FIRST +
    1; private static final int BORRAR_TODAS =
    Menu.FIRST + 2; private static final int SALIR_ID =
    Menu.FIRST + 3;

    //Creación de la base de datos.
    private AdaptadorBD miBD;
```

Método onCreate:

```
//Método onCreate sobrescrito de la clase ListActivity, a
este método se llama una vez se crea la activity.
@Override
public void onCreate(Bundle guardarEstado) {
       super.onCreate(guardarEstado);
        //Utilización de la vista listar_notas.xml
       setContentView(R.layout.listar_notas);
        //Inicialización de la base de datos.
        miBD = new AdaptadorBD(this);
       //apertura de la base de datos.
       miBD.open();
       //Llamada al método llenar datos, que escribirá la
       lista de notas en la pantalla.
       llenarDatos();
       registerForContextMenu(getListView());

}
```

Método llenarDatos:

```
      private void
    llenarDatos() {
  //Creación de un cursor donde se recogen todas las notas.
   Cursor cursorNotas = miBD.recogerTodasNotas();
   //Inicio del cursor.
   startManagingCursor(cursorNota
   s);

  // Creación de un array para especificar los campos que
  hay que    mostrar en la lista,
  // en este caso serán el título, y una pequeña
  introducción de lo que contiene la nota
  // Especificación de donde están contenidos los datos,
  en este caso en la base de datos.
   String[] desde = new String[2];
   desde[0] = AdaptadorBD.KEY_TITULO;
   desde[1] = AdaptadorBD.KEY_INTRO;
  // Y a donde van a ir a parar, que es a los textos 1 y 2,
  que posteriormente se mostraran.
   int[] hasta = new int[2];
   hasta[0]= R.id.texto1;
   hasta[1]= R.id.texto2;

  // Creación de un simple cursor adapter que contendrá
  todos los datos.
  // En el simple cursor adapter hay que especificar el
  contexto, que es el mismo en el que nos encontramos,
  // la capa donde se tienen que mostrar el cursor donde
  se recogen las notas, donde están contenidos los
  datos,y a donde van.

   SimpleCursorAdapter MostrarNotas =
           new SimpleCursorAdapter(this, R.layout.id_notas,
  cursorNotas, desde, hasta);
   // Y finalmente se muestran los datos.
   setListAdapter(MostrarNotas);
```

```
}
```

Método onCreateOptionsMenu:

```
//Método sobrescrito de la clase listactivity que se
encarga de crear el menú.
@Override
 public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);
// Adición de 3 botónes, insertar, borrar todas las
notas, y salir de la aplicación.
// El primer valor del botón es la colocación del mismo en
la pantalla, el segundo es
// el final int creado anteriormente y que después nos
servirá para saber que botón se ha pulsado.
// Por último encontramos el valor del texto del botón,
dado por el valor de los string contenidos en string.xml
// pasados a través del archivo de java autogenerado por
el compilador R.java.
menu.add(0, INSERTAR_ID, 0, R.string.menu_insertar);
menu.add(1,BORRAR_TODAS,0,
R.string.menu_borrar_todas);
menu.add(2,SALIR_ID,0,R.string.menu_salir);
return true;
}
```

Método onMenuItemSelected:

```
// Método sobrescrito de la clase listActivity que pasa a
la acción cuando se pulsa un botón del menú.
@Override
public boolean onMenuItemSelected(int CaracteristicaID,
MenuItem item) {

// Mediante getItemId se obtiene el valor del botón
pulsado.
switch(item.getItemId()) {
// Si el botón pulsado es el insertar, llamada al método
crear nota.
   case INSERTAR_ID:
       crearNota();
       break;
      // Si el botón pulsado es salir, finalización de
      la aplicación.
   case SALIR_ID:
       finish();
       break;
   // Si el botón pulsado es borrar todas:
         case
    BORRAR_TODAS:
      //Creación de una alerta.
      AlertDialog alerta;
      // Inicialización de la alerta.
      alerta = new AlertDialog.Builder(this).create();
      // Titulo de la alerta.
```

```
            alerta.setTitle("Alerta"
            );
            // Mensaje.
            alerta.setMessage("Esto borrará todas las notas");
            // Botones de la alerta:
          // En el caso de que el usuario quiera borrar todas
          las notas:
          alerta.setButton("SI", new
          DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogo, int
            con)
            {
                // Llamada al método borrarTodas de la base
                de datos.
                miBD.BorrarTodas();
                // Vuelta a cargar los datos en la pantalla,
                para refrescarla y que borre las notas antiguas.
                llenarDatos();
                // Refresco de la pantalla.
                registerForContextMenu(getListView()
                );

                return;
             } });
          // En el caso de que el usuario decida no borrar
          las notas, no se hará nada.
          alerta.setButton2("NO", new
          DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialogo, int
                con) {
                      retur
                        n;
                }
           });
            // Se muestra la alerta.
            alerta.show();
          break;

          }

      return super.onMenuItemSelected(CaracteristicaID, item);
    }
```

Método onCreateContextMenu:

```
    // Método sobrescrito de la clase listActivity, este método
    crea el menú
    // que apareará cuando el usuario haga una pulsación
    larga encima de una nota creada con anterioridad.

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
                ContextMenuInfo menuInfo) {
          super.onCreateContextMenu(menu, v, menuInfo);
          // El menú solo tendrá una opción, que es borrar
          la nota.
```

```
        menu.add(0, BORRAR_ID, 0, R.string.menu_borrar);
    }

    // Método que sobrescribe al mismo de la clase
    listActivity y que se encarga de manejar el menú
    contextual.
    @Override
    public boolean onContextItemSelected(MenuItem item) {
    // Mediante getItemId se obtiene el valor de la
    opción pulsada, en este caso única.
    switch(item.getItemId()) {
    case BORRAR_ID:
    // Obtención de la información de la nota sobre la que
    nos encontramos.
    AdapterContextMenuInfo info = (AdapterContextMenuInfo)
    item.getMenuInfo();
    // Acesso al método borrar nota de la base de datos
    pasándole el id de la nota en la que nos encontramos.
            miBD.borrarNota(info.id);
            // Refresco de la pantalla para que la nota se
            borre. llenarDatos();
            return true;
        }
        return super.onContextItemSelected(item);
    }
```

Método crearNota:

```
    // Metodo para crear una nota.
    private void crearNota() {
    // Creación de un nuevo intent diciéndole en que
    contexto estamos y que clase queremos lanzar.
     Intent i = new Intent(this, EditarNotas.class);
     // Inicio de la actividad en modo
     crear. startActivityForResult(i,
     ACTIVITY_CREAR);
    }

    // Método sobrescrito de la clase listActivity,
    // el cual se ejecuta cuando se pulsa algún elemento de
    la lista de notas.
    @Override
    protected void onListItemClick(ListView l, View v, int
    posicion, long id) {
          super.onListItemClick(l, v, posicion, id);
         // Creación de un nuevo inent diciéndole en que
         contexto estamos y que clase queremos lanzar.
         Intent i = new Intent(this, EditarNotas.class);
         // Adición de información extra a la actividad, con
         el id de la nota que queremos visualizar.
         i.putExtra(AdaptadorBD.KEY_IDNOTA, id);
         // Inicio de la actividad en modo editar.
         startActivityForResult(i, ACTIVITY_EDITAR);
     }
```

Anexo 1

Método onActivityResult:

```
      // Este método se ejecutará cuando se regrese de cualquier
      actividad de las arriba mencionadas.
      // Sobrescribe al mismo de la clase listActivity,
      añadiendo una llamada al método llenar datos
      // para que la nueva nota se pueda visualizar.
      @Override
      protected void onActivityResult(int solicitarCodigo, int
      codigoResultado, Intent intent) {
         super.onActivityResult(solicitarCodigo,
         codigoResultado,
intent);
         llenarDatos();
      }
```

Finalización de la clase:

```
      }
```

Anexo 2

Listar_notas.xml

```
//Cabeceras del xml autogeneradas por Eclipse:

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

//Creación de la lista de notas:

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

//Este Texto solo se mostrará cuando no haya ninguna nota en la lista:

      <TextView android:id="@+id/android:empty"
          android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="No hay notas!"
            android:textSize="30px"/></LinearLayout>
```

Anexo 3

Clase EditarNotas.java

Esta es la clase encargada de gestionar tanto el visionado como la edición de las notas, se apoya en editar_notas.xml para mostrarlas adecuadamente, su código es:

Package que contiene al proyecto:

```
package com.android.BlocDeNotas;
```

Importaciones:

```
import com.android.BlocDeNotas.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.database.Cursor;
import android.hardware.SensorListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
```

Inicio de la clase EditarNotas:

```
@SuppressWarnings("deprecation")
public class EditarNotas extends Activity implements
SensorListener {
```

Declaración de atributos globales:

```
    // Textos editables globales para usar entre métodos.
    private EditText textoTitulo;
    private EditText textoCuerpo;
    // Long id de la nota para saber en todo momento en que
    nota se encuentra el programa
    private Long idNota;
    // Creación e la base de datos.
    private AdaptadorBD miBD;
    //Buleano con el estado de guardado de la nota.
    private boolean guardado=true;
    //Enteros finales para la creación del menu.
    private static final int EDITAR_NOTA = Menu.FIRST;
    private static final int BORRAR_TEXTO = Menu.FIRST
    + 1; private static final int SALIR = Menu.FIRST +
    2;
    private static final int GUARDAR = Menu.FIRST + 3;


  //Acelerómetro.....................................
  final String tag = "acelera";
```

```
//SensorManager es la clase que se encarga de manejar
tanto el acelerómetro como la brújula.
SensorManager sm = null;
//.....................................................
```

Método onCreate:

```
// Método onCreate sobrescrito de la clase ListActivity, a
este método se llama una vez se crea la activity.
@Override
        protected void onCreate(Bundle
                guardarEstado) {
        super.onCreate(guardarEstado);


      //Acelerómetro.....................................
       // Inicialización del acelerometro.
      sm = (SensorManager)
      getSystemService(SENSOR_SERVICE);
      //.................................................

       //Utilización de la vista editar_notas.xml
       setContentView(R.layout.editar_notas);
       // Inicialización de la base de datos.
       miBD = new AdaptadorBD(this);
       // Apertura de la base de datos.
       miBD.open();

      // Tanto textoTitulo como textoCuerpo pasan a guardar
      la información que posteriormente se mostrará por
      pantalla. textoTitulo = (EditText)
      findViewById(R.id.titulo); textoCuerpo = (EditText)
      findViewById(R.id.cuerpo);

      // Llegados a este punto se impide hacer click tanto
      en el titulo como en el cuerpo de la nota,
      // puesto que se supone que estamos en modo de
      visionado, siempre y cuando el titulo de la nota no
      sea nulo,
      // si fuera nulo se procedería a llamar al
      método nombreNota como se verá a posteriori.
      textoTitulo.setClickable(false);
      textoTitulo.setCursorVisible(false);
      textoTitulo.setFocusable(false);
      textoCuerpo.setClickable(false);
      textoCuerpo.setCursorVisible(false);
      textoCuerpo.setFocusable(false);

      // En el caso de que guardarEstado no sea nulo quiere
      decir que acabamos de crear la nota.
      idNota = guardarEstado != null ?
      guardarEstado.getLong(AdaptadorBD.KEY_IDNOTA): null;
      // En el caso de que el idNota sea nulo quiere decir
      que la nota ya está creada
      // y por lo tanto tenemos que asignarle el id
```

```
        almacenado en los extras durante la clase anterior.
         if (idNota == null) {
                   Bundle extras = getIntent().getExtras();
                        idNota = extras != null ?
        extras.getLong(AdaptadorBD.KEY_IDNOTA) : null;

        }

        // Llamada al método que se encarga de rellenar los
        campos en la pantalla.
        rellenarCampos();
        // En el caso de que el texto fuera menor o igual
        que cero quiere decir que es una nota nueva
        if(textoTitulo.getText().toString().length() <= 0)
         {
        //Llamada al método que se encarga de hacer la
        nota clickable para poder escribir en ella.
         nombreNota();
         }
    }
```

Método rellenarCampos:

```
    // Método encargado de rellenar los datos.
    private void rellenarCampos() {
         // Solo rellena los datos si la nota está
        creada correctamente.
        if (idNota != null) {
              // Creación de un cursor donde se recoge la
              nota mediante el
              // método recogerNota de la clase que gestiona
              la base de datos.
              Cursor nota = miBD.recogerNota(idNota);
              // Inicialización del cursor.
              startManagingCursor(nota);

              // Se cambia el texto de textoTitulo y
              textoCuerpo por el que contiene la nota.
              textoTitulo.setText(nota.getString(
              nota.getColumnIndexOrThrow(AdaptadorBD.KEY_TITU
              LO))); ));}
              textoCuerpo.setText(nota.getString(
              nota.getColumnIndexOrThrow(AdaptadorBD.KEY_CUERP
        O)}
```

Método nombreNota:

```
// Método encargado de editar la nota cuando se crea.
private void nombreNota()
{
        // Se avisa de que la nota no esta guardada.
        guardado=false;
        // Inicialmente se coloca un titulo provisional
        hasta que el usuario decida cambiarlo.
        textoTitulo.setText("Titulo");
```

```
        // Se permite hacer click y cambiar el texto al
        titulo.
        textoTitulo.setClickable(true);
        textoTitulo.setCursorVisible(true);
        textoTitulo.setFocusable(true);
        textoTitulo.setFocusableInTouchMode(true);
        // El textoTitulo pasa a tener el foco del programa.
        textoTitulo.requestFocus();
        // Se permite hacer click y cambiar el texto al
        cuerpo de la nota.
        textoCuerpo.setClickable(true);
        textoCuerpo.setCursorVisible(true);
        textoCuerpo.setFocusable(true);
        textoCuerpo.setFocusableInTouchMode(true);

        // Se coloca el texto provisional del cuerpo.
         textoCuerpo.setText("Escribe aqui la
        nota.");

        // Creación de un listener en textoCuerpo,
        // este permitirá borrar el texto provisional de la
        nota una vez el usuario haga click en el.
        textoCuerpo.setOnFocusChangeListener(new
        View.OnFocusChangeListener() {
              // Cuando se pulse sobre el texto:
              @Override
              public void onFocusChange(View v, boolean
              hasFocus) {
                    setResult(RESULT_OK);
                    // El texto pasa a estar vacio.
                    textoCuerpo.setText("");
              }
        }); }
```

Método onSaveInstanceState:

```
  // Siempre que se salga de esta actividad,
  // por ejemplo haciendo uso de la función finish se
  pasará por ese método, sobrescrito de la clase
  activity,
  // esto nos permite después volver a rellenar la lista de
  la pantalla principal con los nuevos datos.
   @Override
  protected void onSaveInstanceState(Bundle estado) {
        super.onSaveInstanceState(estado);
        estado.putLong(AdaptadorBD.KEY_IDNOTA, idNota);
}
```

Método onResume:

```
// Si se inicia la aplicación desde un estado anterior:
@Override
protected void onResume() {
    super.onResume();
```

```
        //Acelerometro.....................................
        // Activación el acelerometro.
        sm.registerListener(this,
        SensorManager.SENSOR_ORIENTATION |
        SensorManager.SENSOR_ACCELEROMETER,
        SensorManager.SENSOR_DELAY_UI);
        //.................................................
        // Se llama al método rellenar datos para refrescar
        la pantalla.
        rellenarCampos();
}
```

Método onCreateOptionsMenu:

```
//Método que sobrescrito de la clase activity que se encarga
de crear el menú.
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
       // En este caso tendrá 4 opciones, edita, borrar
       texto, y salir de la aplicación y guardar.
       // El primer valor del botón es la colocación del
       mismo en la pantalla, el segundo es
       // el final int creado anteriormente y que después
       nos servirá para saber que botón se ha pulsado.
       // Por último encontramos el valor del texto del
       botón,dado por el valor de los string contenidos en
       string.xml pasados a través del archivo de java
       autogenerado por el compilador R.java.
       menu.add(0, EDITAR_NOTA, 0,
       R.string.menu_editar_nota);
       menu.add(1,BORRAR_TEXTO,0,
       R.string.menu_borrar_texto);
       menu.add(2,SALIR,0,R.string.menu_cerrar);
       menu.add(3,GUARDAR,0,R.string.menu_guardar);
       return true;
}
```

Método onMenuItemSelected:

```
// Método sobrescrito de la clase Activity que pasa a la
acción cuando se pulsa un botón del menú.
public boolean onMenuItemSelected(int CaracteristicaID,
MenuItem item) {
       // Mediante getItemId se obtiene el valor del
       botón pulsado.
       switch(item.getItemId()) {

       // Si el botón pulsado es editar:
       case EDITAR_NOTA:
       // La nota pasa al estado de no guardada.
       guardado = false;
       // Se permite hacer click en el titulo y en el
       cuerpo.
       textoTitulo.setClickable(true);
       textoTitulo.setCursorVisible(true);
```

```
textoTitulo.setFocusable(true);
textoTitulo.setFocusableInTouchMode(true);
textoCuerpo.setClickable(true);
textoCuerpo.setCursorVisible(true);
textoCuerpo.setFocusable(true);
textoCuerpo.setFocusableInTouchMode(true);
// Además se pasa el foco al cuerpo de la nota.
textoCuerpo.requestFocus();
 break;

// Si el botón pulsado es salir:
case SALIR:
// En caso de que la nota se encuentre en el estado
de no guardada:
if (guardado==false)
 {
      // Se crea una alerta.
      AlertDialog alerta;
      // Se inicializa con los siguientes valores:
      alerta = new
      AlertDialog.Builder(EditarNotas.this).create
      ();
      // Título.
      alerta.setTitle("Alerta
      ");
      // Mensaje.
      alerta.setMessage("¿Quieres guardar la nota?");
       // En el caso de que queramos guardar la nota:
      alerta.setButton("SI", new
      DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,
            int which) {
            // Llamada al método guardar
            nota.
            guardarNotaEnBD();
            // Finalizamos la clase y pasamos a la
            anterior, es decir a la vista de la lista
            de las notas.
            finish();
            return;
            });
            // En el caso de que no queramos guardar la
            nota:
            alerta.setButton2("NO", new
            DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,
            int which) {
                  // Finalizamos la clase y pasamos a
                  la anterior, es decir a la vista de
                  la lista de las notas.
                 finish();
                return;
                 }
      });
// Se muestra la alerta.
```

```
            alerta.show();
        }
        // En el caso de que la nota se encuentre en el estado
        de guardad, simplemente
        // finalizamos la clase y pasamos a la anterior, es
        decir a la vista de la lista de las notas.
        else
        {
            finish();
        }
         break;
         // Si el botón pulsado es guardar la nota:
         case GUARDAR:
            // Llamada a la clase que se encarga de guardar la
             nota.
            guardarNotaEnBD();
             // El estado pasa a ser guardado.
             guardado = true;
            // Aparece un mensaje automático de tipo pequeño
            avisando de que la nota se ha guardado.
            Toast.makeText(getBaseContext(), "Se han guardado
            los cambios.", Toast.LENGTH_SHORT).show();
            break;
         // Si el boton pulsado es borrar el texto:
         case BORRAR_TEXTO:
              // Creación de una alerta.
              AlertDialog alerta;
              // Se inicializa.
              alerta = new AlertDialog.Builder(this).create();
               // Titulo:
              alerta.setTitle("Alerta");
              // Mensaje:
              alerta.setMessage("¿Seguro que quieres borrar todo
              el
              texto?");
               // Si se quiere borrar el texto:
              alerta.setButton("SI", new
              DialogInterface.OnClickListener
              () {
                    public void onClick(DialogInterface dialogo,
                    int
                    con) {
                         // Se cambia todo el texto por el
                         siguiente:
                         textoCuerpo.setText("Escribe aqui la
                         nota."); return;
                          }
              });
               // En el caso de que no quiera borrar el texto, no
               se hace nada.
               alerta.setButton2("NO", new
               DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialogo,
                   int con) { });
              return;
              } alerta.show();
```

```
            break;
        }

        return super.onMenuItemSelected(CaracteristicaID,
        item);
    }
```

Método guardarNotaEnBD:

```
// Método encargada de guardar la nota en la base de datos.
private void guardarNotaEnBD() {

      //Recogemos el titulo y el cuerpo.
      String titulo = textoTitulo.getText().toString();
      String cuerpo = textoCuerpo.getText().toString();
      // Aquí recogemos el cuerpo otra vez,
      // para a continuación escoger solo los primeros 30
      caracteres en el string intro.
      String aux = textoCuerpo.getText().toString();
      String intro;
      // Comprobamos el tamaño del texto
       int tamanyo = aux.length();
       // Si es menor de 30 intro pasa a ser todo el texto.
       if(tamanyo < 30)
       {
              intro = aux.substring(0,tamanyo);
       }
      // Si es mayor de 30 mediante un substring escogemos solo
      los 30 primeros caracteres mas (...) para indicar que la
      nota continua.
       else
       {
              intro = aux.substring(0, 30) + " (...)";
       }

      // Si el id de la nota es nulo creamos la nota con los
      valores obtenidos anteriormente.
       // Este metodo nos devuelbe el nuevo id.
       if (idNota == null) {
            long id = miBD.crearNota(titulo, cuerpo, intro);
            // Si se crea correctamente ya tenemos un nuevo id
            para la nota.
            if (id > 0) {
                idNota = id;
            }
       }
      // Si el id de la nota no era nulo simplemente
      actualizamos lanota en la que nos encontramos.
       else {
           miBD.actualizarNota(idNota, titulo, cuerpo, intro);
       }
}
```

Método onSensorChanged:

Anexo 3

```
// Este método se ejecuta siempre que el sensor cambie de
posición,
// en este caso servirá para pasar de una nota a la siguiente.
public void onSensorChanged(int sensor, float[] values) {
      synchronized (this) {
      // Aquí obtenemos los valores del sensor para x, y e z.
      Log.d(tag, "onSensorChanged: " + sensor + ", x: " +
      values[0]
      + ", y: " + values[1] + ", z: " + values[2]);
      // Como el sensor sirve tanto para la brújula como
      para el acelerómetro,
      // comprobamos que el sensor que se ha activado es el
      del acelerómetro, en ese caso:
      if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
            // Creación de un identificador de la nueva nota.
             long identificador;
             // Si la x fuera mayor que 8:
                   if( values[0] >= 8)
                   {
                      // El identificador pasa a ser uno mas que
                   el de la id de la nota.
                      identificador = idNota +1;
                      try
                      {
                            idNota = identificador;
                            // Se pasa a la siguiente nota y
                            se rellenan los campos.
                            rellenarCampos();
                      }
                      catch(Exception e)
                      {
                            // En el caso de que no se consiga
                            ir a la siguiente nota
                            // se vulve a colocar el idNota en
                            la actual y lanza una alerta
                            indicando que la nota actual es la
                            última.
                            idNota = identificador -1;
                            AlertDialog aler;
                            aler = new
                            AlertDialog.Builder(this).create();
                            aler.setTitle("Alerta");
                            aler.setMessage("Esta es la
                            ultima nota.");
                            aler.setButton("OK", new
                            DialogInterface.OnClickListener() {
                                  public void
                                  onClick(DialogInterface dialogo,
                                  int con) { return; } });
                            aler.show
                               ();
                      }
            }

            // Si la x fuera mayor que -8, estaríamos en el caso
            contrapuesto al anterior,
```

```
            // por lo tanto hay que ir una nota hacia atrás, así
            que volvemos a hacer lo mismo pero con el valor del
            identificador a una nota menos. En el caso de que
            fuera la primera se lanza un mensaje de aviso.
             if(values[0] <= -8){
                 try{
                       identificador = idNota - 1;
                       idNota = identificador;
                       rellenarCampos();
                  }
                  catch(Exception e)

                        idNota = identificador
                        + 1; AlertDialog aler;
                        aler = new
                        AlertDialog.Builder(this).create();
                        aler.setTitle("Alerta");

                        aler.setMessage("Esta es la primera
                        nota."); aler.setButton("OK", new
                        DialogInterface.OnClickListener() {
                             public void onClick(DialogInterface
                                      dialogo, int con) {
                                           return; } });
                                       aler.show();
                        }
                   }
               }
          }
     }
```

Método onAccuracyChanged:

```
     // Este método sirve para avisar que la precisión a
     cambiado,
     // es conveniente sobrescribirlo en esta aplicación pero
     no es completamente necesario.
     public void onAccuracyChanged(int sensor, int accuracy) {
          Log.d(tag,"onAccuracyChanged: " + sensor + ",
          accuracy: " + accuracy);

     }
```

Método onStop

```
     // En el caso de que se pare la aplicación, dejamos de
     usar el acelerómetro.
     @Override
     protected void onStop() {
         sm.unregisterListener(this);
         super.onStop();
     }
```

Final de la clase:

```
}
```

Editar_notas.xml

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

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

            //Formato y colocación del título de la nota:
            <EditText android:id="@+id/titulo"
              android:layout_weight="1"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"
              android:textStyle="bold"
              android:textSize="30px"
              android:background="#000000"
              android:textColor="#CCCCCC"/>
      </LinearLayout>

            //Formato y colocación del cuerpo:
            <EditText android:id="@+id/cuerpo"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:scrollbars="vertical" />
</LinearLayout>
```

Clase AdaptadorBD.java

Por último nos encontramos con la clase AdaptadorDB encargada de gestionar la base de datos en la que se guardan las notas, esta clase se apoya en SQLite para guardar los datos. Su contenido es:

Package que contiene al proyecto:

```
package com.android.BlocDeNotas;
```

Importaciones:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
```

Inicio de la clase:

```
public class AdaptadorBD {
```

Declaración de atributos globales:

```
    // Strings finales que contienen las cadenas de lo que se
    guardará en la BD
    public static final String KEY_TITULO = "title";
    public static final String KEY_CUERPO =
    "body"; public static final String
    KEY_IDNOTA = "_id"; public static final
    String KEY_INTRO = "intro"; private static
    final String TAG = "AdaptadorBD";
    // Objeto de ayuda para manejar la base de datos.
    private DatabaseHelper BDAyuda;
    // La base de datos en SQLite
    private SQLiteDatabase miBD;
    // Cadena para la creación de la base de
    datos en nomenclatura SQLite.
    private static final String DATABASE_CREAR =
                "create table notes (_id integer primary
    key autoincrement, "
                        + "title text not null, body text
    not null, intro text);";

    // Nombre de la base de datos, de la tabla y versión
    de la base de datos a utilizar.
    private static final String DATABASE_NOMBRE = "data";
    private static final String DATABASE_TABLA = "notes";
```

```
 private static final int DATABASE_VERSION = 2;
 // Creación de un contexto.
 private final Context miContext;
 //Inicio de la clase de ayuda que extiende de SQLite.
 private static class DatabaseHelper extends
SQLiteOpenHelper
{
```

Método DatabaseHelper:

```
// Constructor de la clase de ayuda encargado de crear la
base de datos.
DatabaseHelper(Context context)
{
          super(context, DATABASE_NOMBRE, null,
          DATABASE_VERSION);
}
```

Método DatabaseHelper:

```
//Este método se encarga de inicializar la base de datos,
se ejecuta siempre cuando se crea la clase.
@Override
public void onCreate(SQLiteDatabase BD) {

            BD.execSQL(DATABASE_CREAR);
}
```

Método onUpgrade:

```
//Método usado en el caso de que haga falta actualizar la
versión de la base de datos
@Override
public void onUpgrade(SQLiteDatabase BD, int
versionAntigua,
     int nuevaVersion) {
     Log.w(TAG, "Upgrading database from version " +
      versionAntigua + " to " + nuevaVersion + ", which
      will
      destroy all old data");
     BD.execSQL("DROP TABLE IF EXISTS notes");
           onCreate(BD);
       }
    }
```

Método AdaptadorBD:

```
// Este método toma el contexto para permitir a la base de
datos abrirse y cerrase.
public AdaptadorBD(Context ctx) {
     this.miContext = ctx;
}
```

Método open:

```
// Método para la apertura de la base de datos de las
notas,
public AdaptadorBD open() throws SQLException {
      // si no se puede, se intenta crear una nueva
      instancia a la base de dato.
      // Si no se puede crear, se lanza una
      excepción. BDAyuda = new
      DatabaseHelper(miContext);
      miBD = BDAyuda.getWritableDatabase();
      return this;
}
```

Método close:

```
// Método para cerrar la base de datos.
public void close() {
       BDAyuda.close();
}
```

Método crearNota:

```
// Método para la Creación de una nueva nota usando tanto
el titulo como el cuerpo y la intro que llegan en los
atributos.
public long crearNota(String title, String body, String
intro) {
      ContentValues initialValues = new ContentValues();
      initialValues.put(KEY_TITULO,
      title);
      initialValues.put(KEY_CUERPO,
      body);
      initialValues.put(KEY_INTRO,
      intro);
      // Si la nota esta creada bien, se devuelve el
      identificador de esa nota. si no se devuelve -1
      como error.
       return miBD.insert(DATABASE_TABLA, null,
initialValues);
}
```

Método borrarNota:

```
// Método encargado de borrar una nota.
public boolean borrarNota(long idNota)
{
      //Mediante la id obtenida en el atributo usando
      la tabla, se procede a borrar la nota.
      return miBD.delete(DATABASE_TABLA, KEY_IDNOTA + "=" +
      idNota, null) > 0;
```

```
}
```

Método BorrarTodas:

```
// Método encargado de borrar todas las notas.
public boolean BorrarTodas() {

   return  miBD.delete(DATABASE_TABLA, null, null) >
                              0;

}
```

Método recogerTodasNotas:

```
// Mediante este metodo se devuelven todas las notas.
public Cursor recogerTodasNotas() {
      // Uso de una consulta a la base de datos para
      almacenar todas las notas en un cursor.
      return miBD.query(DATABASE_TABLA, new String[]
      {KEY_IDNOTA, KEY_TITULO,
       KEY_CUERPO, KEY_INTRO}, null, null, null, null,
       null);
}
```

Método recogerNota:

```
// Mediante este método se devuelve una nota.
public Cursor recogerNota(long idNota) throws SQLException
{
      // Uso de una consulta a la base de datos para
      almacenar la nota indicada en la misma,
      // y devolverla, si es que se encuentra.
      Cursor miCursor =
       miBD.query(true, DATABASE_TABLA, new String[]
       {KEY_IDNOTA, KEY_TITULO, KEY_CUERPO,
       KEY_INTRO}, KEY_IDNOTA + "=" + idNota, null,
                   null, null, null, null);
      if (miCursor != null) {
           miCursor.moveToFirst();
       }
   return miCursor;

}
```

Método actualizarNota:

```
// Método encargado de actualizar la nota usando los detalles
suministrados.
public boolean actualizarNota(long idNota, String title,
String
```

```
    body, String intro) {
            ContentValues args = new
            ContentValues(); args.put(KEY_TITULO,
            title); args.put(KEY_CUERPO, body);
            args.put(KEY_INTRO, intro);
            // La nota a ser actualizada es especificada usando
            su identificador.
            return miBD.update(DATABASE_TABLA, args, KEY_IDNOTA +
            "=" + idNota, null) > 0;
     }
```

Final de la clase:

```
}
```

Anexo 6

Clase CuadriculaImagenes.java

Esta clase contiene dos subclases, cuadrícula imágenes y el adaptador de las imágenes. El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones

```
import java.io.IOException;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.Display;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
```

Inicio de la clase principal

```
public class CuadriculaImagenes extends Activity implements
OnItemClickListener {
```

Declaración de atributos globales:

```
    //Cuadrícula que maneja las imágenes
    private GridView imagenesTarjeta;
    //Adaptador de las imágenes a la cuadrícula
    private adaptadorImagenes adaptadorImagenes;
    //Este display se usa para obtener el ancho de la pantalla
    private Display display;
```

onCreate

```
//Método onCreate sobreescrito de la clase Activity, a este
  método se llama una vez se crea la activity.

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
        //Seleccion de la vista que usará la actividad
        setContentView(R.layout.cuadricula_imagenes);
        //Obtención del tamaño de la pantalla.
        display = ((WindowManager)getSystemService
        (Context.WINDOW_SERVICE)).getDefaultDisplay();
        //Inicialización de imagenesTarjeta,
        //dándole como parámetros la cuadrícula creada
        //en la vista cuadricula_imagenes.xml
        //El número de columnas dado mediante el ancho de
           pantalla
        imagenesTarjeta = (GridView)
        findViewById(R.id.sdcard);

        imagenesTarjeta.setNumColumns(display.getWidth()/95);
        //Preparo la cuadrícula para cuando ocurra un veneto.
        imagenesTarjeta.setOnItemClickListener
        (CuadriculaImagenes.this);

        //Inicializacion del adaptador que cargará las
        imágenes.
        adaptadorImagenes = new
        adaptadorImagenes(getApplicationContext());
        //Adjudico el adaptador a la cuadrícula.

        imagenesTarjeta.setAdapter(adaptadorImagenes);
        //Llamada al método que carga las imágenes.
        cargarImagenes();
}
```

Método cargarImagenes

```
private void cargarImagenes()
{
    Bitmap imagen = null;
    Bitmap nuevaImagen = null;
    Uri direccion = null;
    // Creación de un array donde se encuentran todas
    //las identificaciones de los Thumbnails que voy a
       mostrar
    String[] identificadores =
    {MediaStore.Images.Thumbnails._ID};
    // Creacion de un puntero a la tarjeta SD diciendole que
    me devuelva todos los archivos que contengan en una de
    sus columnas los identificadores de los thumbnails, es
    decir, todas las fotos.
    Cursor cursor = managedQuery(
    MediaStore.Images.Thumbnails.EXTERNAL_CONTENT_URI,
         identificadores, null, null, null);
```

```
        //Obtengo los indices de todas las imagenes cargadas en
        el puntero.
        int indiceImagenes = cursor.getColumnIndexOrThrow
        (MediaStore.Images.Thumbnails._ID);
        //Obtengo el numero de imagenes que hay en el puntero
        int tamanyo = cursor.getCount();
        int IDImagen = 0;
        //Para cada imagen en el puntero:
        for (int i = 0; i < tamanyo; i++) {
           //Muevo el puntero a esa posicion y obtengo el
             identificador.
           cursor.moveToPosition(i);
           IDImagen = cursor.getInt(indiceImagenes);
           //obtengo la dirección real que me servirá para crear
           la pequeña imagen.
           direccion = Uri.withAppendedPath
           (MediaStore.Images.Thumbnails.EXTERNAL_CONTENT_URI,
           "" + IDImagen);
           try {
                //Obtengo la imagen mediante la dirección
                imagen =
                BitmapFactory.decodeStream (getContentResolver().
                openInputStream(direccion));
                 //Compruebo que es correcta y creo la nueva
                imagen del tamaño adecuado.
                if (imagen != null) {
                     nuevaImagen = Bitmap.createScaledBitmap
                     (imagen, 80, 80, true);
                     //Libero la memoria de la imagen primera
                     imagen ya que este proceso es tedioso para el
                     sistema y se puede sobrecargar la memoria RAM
                     en moviles con poca cantidad de ella.
                     imagen.recycle();
                     //Si la nueva imagen tambien es correcta la
                     añado a la lista de las que se mostrarán.
                     if (nuevaImagen != null)
                     {
                      adaptadorImagenes.anyadirFoto(nuevaImagen);
                     }
                 }
            } catch (IOException e) {
            //Si hubiera algún error en la imágen,
            simplemente no se muestra.
            }
        }
        //Cierro el puntero.
        cursor.close();
    }
```

Método onItemClick

```
    public void onItemClick(AdapterView<?> padre, View v, int
    posicion, long id) {
```

```
        //System.gc es el garbage collector,
        se encarga de liberar memoria.
        System.gc();
        //Creo un array en el que después estarán todas las
          imágenes
        //para pasárselas a la clase que se encarga de
          mostrarlas a pantalla completa
        ArrayList<String> imagenes = new ArrayList<String>();
        //Obtengo todos los datos de las imágenes.
       String[] datosImagen = { MediaStore.Images.Media.DATA };

       //Creación de un puntero a la tarjeta SD diciéndole que
          Me devuelva todos las imágenes.
        Cursor cursorImagenActual =
        managedQuery(MediaStore.Images.Media.EXTERNAL_CONTENT_UR
        I, datosImagen, null, null, null);
        //Obtengo los índices de todas las imágenes
        cargadas en el puntero.
        int indiceImagenes =
          cursorImagenActual.getColumnIndexOrThrow
        (MediaStore.Images.Media.DATA);

        //Vuelvo a limpiar la memoria por si hubiera
        algún problema de sobre carga.
        System.gc();

        //Añado todas las direcciones de las imágenes al array
          Que después le pasaré a la nueva clase.
        for(int j = 0; j< cursorImagenActual.getCount(); j++){
            cursorImagenActual.moveToPosition(j);
            String ima =
            cursorImagenActual.getString(indiceImagenes);
            imagenes.add(ima);
        }

        //Creo la nueva actividad pasándole la lista actual de
          Todas las direcciones de las imágenes
        //que hay en la memoria SD así como la posición de la
          imagen actual que se ha seleccionado para mostrar
        Intent intent =
        new Intent(getApplicationContext(), Imagenes.class);
        intent.putExtra("lista", imagenes);
        intent.putExtra("posicion", posicion);
        startActivity(intent);
    }
```

Método adaptadorImagenes

```
//Algunos de los métodos creados acontinuación no los uso,
pero la case baseAdapter obliga a crearlos.
class adaptadorImagenes extends BaseAdapter {
```

Atributos de la clase

```
//Contexto de la actividad y un array con las imágenes.
 private Context contexto;
 private ArrayList<Bitmap> fotos = new
    ArrayList<Bitmap>();
```

Constructor del adaptador

```
 public adaptadorImagenes(Context context) {
    contexto = context; }
```

Método anyadirFoto

```
 //Método encargado de añadir las fotos
 public void anyadirFoto(Bitmap newBitmap) {
fotos.add(newBitmap); }
```

Método getCount

```
 //Contador de fotos
 public int getCount() { return fotos.size(); }
```

Método getitem

```
 //Método encargado de devolver una foto mediante su
    Posición en el array.
 public Object getItem(int posicion)
{ return fotos.get(posicion); }
```

Método getItemId

```
 //Método encargado de devolver el ID de la foto
    mediante su posición. En este caso he puesto la misma
    posición, ya que el ID ya no es relevante.
 public long getItemId(int posicion) { return posicion; }
```

Método getView

```
//Método encargado de gestionar como la imagen será
    mostrada.
 public View getView(int posicion, View vista, ViewGroup
vistaAnterior) {

    final ImageView imagenGenerada;
    //Si la vista es nula se creará una imagen vacía.
    if (vista == null) {
          imagenGenerada = new ImageView(contexto);
         //Si es correcta hago un cast a ImageView para
          Poder darle los valores adecuados.
     } else {
          imagenGenerada = (ImageView) vista;
     }
     //Escalo la imagen
    imagenGenerada.setScaleType
    (ImageView.ScaleType.FIT_CENTER);
     //Pongo los márgenes entre imágenes
```

```
            imagenGenerada.setPadding(4, 4, 4, 4);
            //Adjudico la imagen mediante la posición dada.
            imagenGenerada.setImageBitmap(fotos.get(posicion));
            //Y por último la devuelvo.
            return imagenGenerada;
        }
    }
}
```

Clase Imagenes.java

Esta clase contiene dos subclases, cuadrícula imágenes y el adaptador de las imágenes. El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones

```
import java.util.ArrayList;

import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageView;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
```

Inicio de la clase

```
public class Imagenes extends Activity implements
View.OnTouchListener{
```

Declaración de atributos globales:

```
    //Identificador de la imagen que estoy mostrando
    (Posición en el array)
     private int posicion;
     //Array que contiene a todas las imágenes
     private ArrayList<String> imagenes = new
      ArrayList<String>();
     //ImageView que sirve para mostrar la imagen dada
     private ImageView mostrar;
```

Método onCreate

```
    //Metodo onCreate sobrescrito de la clase Activity,
    este método se invoca una vez se crea esta actividad.
     @Override
     public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);

           //Utilizacion de la capa main.xml para mostrar la
           vista.
           setContentView(R.layout.imagenes);
           //Recolector de basura, para limpiar el sistema.
           System.gc();
           //Creo un intent y le asigno los datos que ha pasado
```

```
        la clase anterior.
        Intent i = getIntent();
        //Creo un bundle y extraigo los datos del intent
        Bundle extras = i.getExtras();
        //Inicialización de la ImageView mostrar,
        así como la adjudicación de su vista en la capa main
        mostrar = (ImageView) findViewById(R.id.mostrar);
        //Extraigo los datos tanto de la lista como de la
        posición en la que se encuentra la imagen
        seleccionada.
        imagenes = extras.getStringArrayList("lista");
        posicion = extras.getInt("posicion");
        //Preparo la imagen para cuando sea pulsada que
        ocurra un evento.
        mostrar.setOnTouchListener(this);
        //Por último llamo al método mostrando pasándole la
        posición de la imagen a mostrar.
        mostrando(posicion);
}
```

Método mostrando

```
//Este método se encarga de mostrar la imagen seleccionada.
 public void mostrando(int posicion){
        //Creo un bitmap del tipo BitmapFactory,
        esto sirve para ahorar memoria y tiempo,
        //al no tener que cargar la imagen entera.
        BitmapFactory.Options imagenRenderizada =
        new BitmapFactory.Options();
        imagenRenderizada.inSampleSize = 2;
        //Obtengo la ruta de la imagen con la posición
        adecuada
        String nombreImagen = imagenes.get(posicion);
        //Creo un nuevo bitmap con la imagen renderizada
        Bitmap imagenFinal = BitmapFactory.decodeFile
        (nombreImagen, imagenRenderizada);
        //lo muestro:
        mostrar.setImageBitmap(imagenFinal);
}
```

Método onTouch

```
//onTouch es el manejador de los eventos ocurridos en
Pantalla en direccion recojo que parte de la pantalla ha
sido pulsada.
 public boolean onTouch(View vista, MotionEvent direccion)
{
      //Obtencion de la coordenada x de donde se ha pulsado
      la pantalla
     float x = direccion.getX();




     //En el caso de que la x sea mayor que 160 se procede
     a mostrar la imagen siguiente.
```

Anexo 7

```
            if(x>160)
            {
                  //Si no es la última imagen
                  if(posicion < imagenes.size() - 1){
                        //Aumento la posición y la muestro
                        posicion++;
                        mostrando(posicion);
                  }
            }
            //En el caso de que la x sea menor que 160 se procede
            a mostrar la imagen anterior.
            if(x<160)
            {
                  //Si no es la primera
                  if(posicion > 0 ){
                        //Disminuyo la posición y la muestro.
                        posicion--;
                        mostrando(posicion);
                  }

            }

        return super.onTouchEvent(direccion);
      }


}
```

Anexo 8

Clase ListaVideos.java

Esta clase contendrá la lista de vídeos.

El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones

```
import java.io.File;
import java.io.FilenameFilter;
import java.util.ArrayList;
import java.util.List;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
```

Clase FiltroVideos

```
//Clase que se encarga de filtrar los archivos
//para que solo se muestren los vídeos soportados por el
terminal, los que tienen la extensión 3gp
class FiltroVideos implements FilenameFilter {
    public boolean accept(File dir, String name) {
        return (name.endsWith(".3gp"));
    }
}
```

Inicio de la clase principal:

```
public class ListaVideos extends ListActivity implements
OnClickListener{
```

Declaración de atributos globales:

```
    //Array que contendrá los datos de la lista.
    private List<String> elementos = null;
    //Ruta actual, inicializada con la ruta por defecto.
```

```
    private String rutaActual="/sdcard/";
    //Texto que muestra por pantalla la ruta en la que nos
    encontramos.
    private TextView TextViewRuta;
    //Boton para ir atras en el arbol de directorios.
    private Button atras;
```

Método onCreate

```
//Metodo onCreate sobrescrito de la clase ListActivity, este
método se invoca una vez se crea esta actividad.

    @Override
    public void onCreate(Bundle icicle) {
        try {
           super.onCreate(icicle);

           //Utilizacion de la capa lista_videos.xml
           para mostrar la vista.
           setContentView(R.layout.lista_videos);
            //Inicialización del TextView y del boton,
           //así como la adjudicación de sus
           vistas en la capa lista_videos
           TextViewRuta = (TextView)
           findViewById(R.id.rutaVideos);
           atras = (Button) findViewById(R.id.AtrasVideos);
           //Preparo el boton para cuando un evento ocurra en
           el.
           atras.setOnClickListener(this);
           //Llamo al método rellenar,
           para rellenar la lista con los elementos.
           rellenar();
        } catch (NullPointerException e) {
           //Si no consigue cargar la lista aparecerá
           el nombre de la aplicación.
           Log.v(getString(R.string.app_name), e.getMessage());
        }
    }
```

Método onClick

```
    //Este método es el que se encarga de manejar los eventos,
    //en este caso si se pulsa el boton, llamará al método
      rellenar anterior para que cargue la lista con los datos
      del directorio anterior
    public void onClick(View v) {
       if(v.equals(atras)){
            AntesDeRellenar();
       }
    }
```

Método onListItemClick

```
    //Este método se encarga de manejar los eventos
    que ocurran en la lista.
```

```
@Override
protected void onListItemClick(ListView lista, View vista,
 int posicion, long id) {

   //Recojo la posicion del elemento seleccionado
    int elementoSeleccionado = posicion;
    //Creo un string para identificar cuando se trata
   de una carpeta
    String guion = "      [";
    //Creo un string para identificar
   cuando me encuentro en el separador
    String rayas = "-----------------------------------------
      ---------------------";
    //Creo un string con la ruta del elemento seleccionado.
    String seleccionado =
     elementos.get(elementoSeleccionado);
    //Si no he seleccionado la linea de las rayas:
    if(seleccionado.compareTo(rayas) != 0){
       //Compruebo si se trata de una carpeta si lo es:
          if(seleccionado.charAt(0) == guion.charAt(0))
          {
             //selecciono los datos de la ruta pero sin los
             añadidos para diferenciarlo de un archivo.
             seleccionado =
             rutaActual + seleccionado.substring(7,
             seleccionado.length() - 1) + "/";
             //Se lo paso a ruta actual.
             rutaActual = seleccionado;
             //Se muestra por pantalla pero cambiando
             el guion por -> para hacerlo ams legible
             TextViewRuta.setText(rutaActual.replace
             ("/", " -> "));
             //Relleno la lista con los nuevos
             datos mediante el método rellenar
             rellenar();

          //Si no es un directorio:
          }else{
             //Creo una nueva actividad que se encargará de
             mostrar el vídeo a la que le paso la ruta entera
             del vídeo a mostrar con extensión incluida
             Intent i = new Intent(this, Video.class);
             i.putExtra("1", rutaActual +
             elementos.get(posicion) + ".3gp");
            startActivity(i);
        }
       }
}
```

Método rellenar

```
//Método que se encarga de rellenar la lista
private void rellenar() {
     //Inicializo el array que contendrá los datos de la lista
      elementos = new ArrayList<String>();
```

```
        //Elemento tipo File con la ruta de los elementos a
         mostrar
        File rutaFila = new File(rutaActual);

        //Si hay algún elemento en la ruta de tipo video:
        if (rutaFila.listFiles( new FiltroVideos()).length > 0)
       {
           //Recorro todos los elementos de tipo vídeo:
           for (File file : rutaFila.listFiles( new
           FiltroVideos())) {
                 //Guardo el nombre en un string y le quito la
                 extensión, para que no se muestre por pantalla.
                 String video = file.getName();
                 video = video.substring(0, video.length() - 4);
                 //Añado los videos a la lista de elementos.
                 elementos.add(video);
           }
        }
        //Añado a la lista el separador de videos y directorios.
        elementos.add("------------------------------------
       ------------------------");

        //Recorro todos los elementos de la ruta actual y
        //si hay alguna directorio dentro del directorio actual:
        for( File archivo: rutaFila.listFiles())
        {
            if(archivo.isDirectory())
            //Lo formateo para diferenciarlo de los videos.
           elementos.add("      [" + archivo.getName() + "]");

        }

      //Ya que los datos se van insertando uno a uno
      alfabéticamente, quedarían al revés hay que invertir el
      vector final para que se muestren ordenados
      alfabéticamente:
      int tamanyo=elementos.size();
      String temp;
      //Recorro hasta la mitad del vector y cambio el elemento
      en el que me encuentro por el mismo pero empezando por
      atrás.
      for (int i = 0 ; i < tamanyo/2 ; i++ ){
          temp = elementos.get(i);
          elementos.set(i, elementos.get(tamanyo-1-i));
          elementos.set((tamanyo-1-i), temp);
      }

      //Por último creo un adaptador para mostrar la lista y
      //le paso el array con los elementos tal cual los
       tiene que mostrar
      ArrayAdapter<String> listaArchivos= new
      ArrayAdapter<String>(this, R.layout.fila, elementos);
     setListAdapter(listaArchivos);
}
```

Anexo 8

Método AntesDeRellenar

```
//A este método se le llama cuando hay que cargar la lista
 de nuevo, si estamos en el directorio raiz de la tarjeta no
 haremos nada, pero si estamos en otra carpeta quiere decir
 que el usuario ha ido un paso atrás, y habrá que cambiar la
 ruta
private void AntesDeRellenar() {

   int ultimo=0;
   int penultimo = 0;
   //Compruebo que la ruta no sea el directorio raiz.
   String casa = "/sdcard/";
   if(rutaActual.compareTo(casa) != 0)
   {
        //Localizo la penultima barra en la ruta para
        borrar la última carpeta.
         for ( int i = 0 ; i < rutaActual.length() ; i++)
         {
             String barra = "/";
             if(rutaActual.charAt(i) == barra.charAt(0))
             {
                   penultimo = ultimo;
                   ultimo = i;
             }
          }
         //Creo la nueva ruta eliminando la última carpeta
         rutaActual = rutaActual.substring(0, penultimo+1);
         //muestro en el TextView de la vista la nueva
           ruta cambiando la barra por ->
         TextViewRuta.setText
         (rutaActual.replace("/", " -> "));

 }
 //Ya con la ruta adecuada llamo al método rellenar.
 rellenar();
}
```

Anexo 9

Clase Musica.java

Esta clase se encarga de manejar la parte musical del reproductor, se apoya en la capa música.xml

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones

```
import java.io.File;
import java.io.FilenameFilter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import android.app.ListActivity;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;
```

Clase FiltroMp3

```
//Clase que se encarga de filtrar los archivos
//para que solo se muestren las canciones soportadas por el
terminal, las que tienen la extensión mp3
class FiltroMp3 implements FilenameFilter {
    public boolean accept(File dir, String name) {
        return (name.endsWith(".mp3"));
    }
}
```

Inicio de la clase principal:

```
public class Musica extends ListActivity implements
OnClickListener, Runnable{
```

Declaración de atributos globales:

```
    //MediaPlayer encargado de gestionar el sonido
    private MediaPlayer mp = new MediaPlayer();
```

```
//Barra de progreso
private ProgressBar progreso;
//Posicion actual de la canción, para saber donde se
encuentra  cuando paro la música.
private int posicionActual = 0;
//Botones que se mostrarán
private ImageButton pausa;
private ImageButton detener;
private ImageButton anterior;
private ImageButton posterior;
private Button atras;
//Cancion que está sonando en este momento, -1 quiere decir
ninguna.
private int cancionSonando = -1;

//Array con la lista de canciones (Estará ordenado al
revés)
private ArrayList<String> ListaCanciones = new
ArrayList<String>();
 //Array con la lista de elementos mostrados en la lista
private List<String> elementos = null;
//Ruta actual, por defecto el directorio raiz.
private String rutaActual="/sdcard/";
//Dos TextView para mostrar la ruta donde me encuentro y la
cancion que se está reproduciendo
private TextView TextViewRuta;
private TextView TextViewReproduciendo;
```

Metodo onCreate:

```
//Metodo onCreate sobrescrito de la clase ListActivity, este
método se invoca una vez se crea esta actividad.
@Override
public void onCreate(Bundle icicle) {
    try {
       super.onCreate(icicle);

       //Utilizacion de la capa musica.xml para mostrar la
       vista.
       setContentView(R.layout.musica);

       // Obtengo una referencia a todos los elementos
       Generados en musica.xml
       progreso = (ProgressBar)
       findViewById(R.id.progreso);
       pausa = (ImageButton) findViewById(R.id.pausa);
       detener = (ImageButton) findViewById(R.id.detener);
       anterior = (ImageButton) findViewById(R.id.anterior);
       posterior = (ImageButton)
       findViewById(R.id.posterior);
       TextViewRuta = (TextView)
       findViewById(R.id.rutaMusica);
       TextViewReproduciendo = (TextView)
       findViewById(R.id.reproduciendo);
       atras = (Button) findViewById(R.id.AtrasVideos);
```

```
            // Establezco que la clase actual sea la que maneje
            los clics sobre estos botones
            pausa.setOnClickListener(this);
            detener.setOnClickListener(this);
            anterior.setOnClickListener(this);
            posterior.setOnClickListener(this);
            atras.setOnClickListener(this);
            //Llamo al método rellenar, para rellenar la lista
            con los elementos de la ruta actual.
            rellenar();

        } catch (NullPointerException e) {
            //Si no consigue cargar la lista muestra el nombre de
            la aplicación.
            Log.v(getString(R.string.app_name), e.getMessage());
        }
    }

    //Este método es el que se encarga de manejar los eventos.
    public void onClick(View v) {

     //Si el botón pulsado es posterior retocedo una canción ya
     que el vector está ordenado al revés.
     if(v.equals(posterior)){
           //Si cancionSonando es mayor que cero me aseguro de
           que no es la primera canción
           if(cancionSonando > 0)
           {
                 cancionSonando--;
                 //Llamo al método reproducir.
                 Reproducir();
           }
      }
     //Si el botón pulsado es anterior avanzo una canción ya que
     el vector está ordenado al revés.
     if(v.equals(anterior)){
           //Me aseguro de que no sea la última canción de la
           lista, y de que haya alguna en la lista.
           if(cancionSonando < ListaCanciones.size()-1 &&
           cancionSonando >= 0)
           {
                 cancionSonando++;
                 //Llamo al método reproducir.
                 Reproducir();
           }
      }

     //En este caso si se pulsa el boton atras, llamará al
     método rellenar anterior
     //para que cargue la lista con los datos del directorio
     anterior
     if(v.equals(atras)){
           AntesDeRellenar();
      }
```

```
        // Si el usuario hace clic sobre el botón Reproducir.
        if(v.equals(pausa)){

           // Detengo si se estaba reproduciendo algo
            if(mp != null && mp.isPlaying())
            {
                mp.pause();
                //Me guardo el dato de donde se encuentra la
                canción
                posicionActual = mp.getCurrentPosition();
                //Cambio la imagen del boton por la d
                reproducir.
                pausa.setImageDrawable(null);

                pausa.setImageDrawable(getResources().getDrawabl
                e(R.drawable.reproducir));

                //Si no se estaba reproduciendo quiere decir que
                tengo que emprender la reproducción de nuevo
            }else{
                //Llevo el mediaPlayer hasta donde se quedó
                cuando le di a la pausa.
                mp.seekTo(posicionActual);
                //Empiezo a reproducir y cambio la imagen a
                pausa.
                mp.start();
                pausa.setImageDrawable(null);
                pausa.setImageDrawable(getResources().getDrawabl
                e(R.drawable.pausa));

            }

        }

       // Si el usuario ha hecho clic sobre el botón Detener...
       y estamos reproduciendo...
        if(v.equals(detener) && mp!=null){
            // Detengo la reproducción
            mp.stop();

        }
    }
```

Método onListItemClick:

```
    //Este método se encarga de manejar los eventos que ocurran
    en la lista.
    @Override
    protected void onListItemClick(ListView lista, View vista,
    int posicion, long id) {

        //Recojo la posicion del elemento seleccionado
        int elementoSeleccionado = posicion;
       //Creo un string para identificar cuando se trata de una
       carpeta
```

```
String guion = "      [";
//Creo un string para identificar cuando me encuentro en
el separador
String rayas = "----------------------------------------
   ---------------------";
//Creo un string con la ruta del elemento seleccionado.
String seleccionado =
   elementos.get(elementoSeleccionado);
//Si no he seleccionado la linea de las rayas:
if(seleccionado.compareTo(rayas) != 0){
     //Compruebo si se trata de una carpeta si lo es:
     if(seleccionado.charAt(0) == guion.charAt(0))
     {
        //selecciono los datos de la ruta pero sin los
        añadidos para diferenciarlo de un archivo.
        seleccionado =
        rutaActual + seleccionado.substring(7,
        seleccionado.length() - 1) + "/";
        //Se lo paso a ruta actual.
        rutaActual = seleccionado;
        //Se muestra por pantalla pero cambiando el
        guion por -> para hacerlo ams legible
        TextViewRuta.
        setText(rutaActual.replace("/", " -> "));
        //Relleno la lista con los nuevos datos mediante
        el método rellenar
        rellenar();
     //Si no es un directorio ni la raya, es una
     canción, entonces:
     }else{
        //Limpio la lista àra rellenarla de nuevo.
        ListaCanciones.clear();
        //Elemento tipo File con la ruta de los
        elementos a mostrar
        File rutaFila = new File(rutaActual);
        //Si hay algún elemento en la ruta de tipo
        cancion:
        if (rutaFila.listFiles( new FiltroMp3()).length
        > 0) {
        //Recorro todos los elementos de tipo cancion:
        for (File file : rutaFila.listFiles( new
        FiltroMp3())) {
        //Guardo el nombre en un string y le quito la
        extensión, para que no se muestre por pantalla.
        String cancion = file.getName();
        cancion = cancion.substring(0, cancion.length()
        - 4);
        //Añado las canciones a la lista de canciones.
        ListaCanciones.add(cancion);
        }
        }
        for( int i = 0 ; i < ListaCanciones.size() ;
        i++)
        {
              if(ListaCanciones.get(i).
              compareTo(elementos.get(posicion)) == 0)
```

```
                    {
                        cancionSonando = i;
                    }
                }
                //Me aseguro de que el boton es pausa,
                //ya que si he pausado una cancion y le doy a
                otra no se cambiaría.
                pausa.setImageDrawable(null);
                pausa.setImageDrawable(getResources().
                getDrawable(R.drawable.pausa));
                //Por último llamo al m
                Reproducir();
            }
          }
    }

    //Método que se encarga de la reproducción
    private void Reproducir()
    {
        try {
         //Establezco el TextViewReproduciendo al nombre de la
         canción que está sonando

         TextViewReproduciendo.setText(ListaCanciones.get(canc
         ionSonando));
         //Reinicio el mediaPlayer
         mp.reset();
         //Establezco la ruta de la canción que tiene que
         sonar.
         mp.setDataSource(rutaActual +
         ListaCanciones.get(cancionSonando) + ".mp3");
         mp.prepare();
         // Establezco la barra de progreso al principio.
         progreso.setProgress(0);
         // Establezco que el valor final sea el total de
         milisegundos que dura la canción
         progreso.setMax(mp.getDuration());
         //Inicio de la canción
         mp.start();

         //Preparo un evento para cuando se acabe la canción:
         mp.setOnCompletionListener(new OnCompletionListener()
         {
            public void onCompletion(MediaPlayer mp) {
            //Le digo que reteste uno a la variable
            cancionSonando y que cargue de nuevo Reproducir()
            //Esto es así por que el array está ordenado a la
            inversa.
            if(cancionSonando > 0)
              {
                   cancionSonando--;
                  Reproducir();
              }
         }});
```

```
            // Ejecuto un hilo que se encargará de actualizar la
            barra de progreso cada segundo.
            new Thread(this).start();

            } catch(IOException e) {
                  Log.v(getString(R.string.app_name),
                  e.getMessage());
            }
      }

      //Manejador del hilo que se encarga de actulizar la barra.
      public void run() {
            //Creo dos int con la posicionActual y la duración
            total de la canción
            int posicionActual = 0;
            int total = mp.getDuration();

            // Mientras este reproduciendo y no haya llegado al
            final...
            while(mp!=null && posicionActual<total){
                  try {
                        // Espero un segundo...
                        Thread.sleep(1000);
                        // Obtengo la posición por la que se
                        está reproduciendo. (En milisegundos)
                        posicionActual = mp.getCurrentPosition();
                  } catch (InterruptedException e) {
                              return;
                  } catch (Exception e){
                              return;
                  }
            // Hago que la barra de progreso referencie el valor
            actual.
            progreso.setProgress(posicionActual);
            }
       }



       private void rellenar() {

          //Inicializo el array que contendrá los datos de la
          lista
          elementos = new ArrayList<String>();
          //Elemento tipo File con la ruta de los elementos a
          mostrar
          File rutaFila = new File(rutaActual);

          //Si hay algún elemento en la ruta de tipo cancion:
          if (rutaFila.listFiles( new FiltroMp3()).length > 0) {
            //Recorro todos los elementos de tipo cancion:
            for (File file : rutaFila.listFiles( new
            FiltroMp3())) {
              //Guardo el nombre en un string y le quito la
              extensión, para que no se muestre por pantalla.
              String cancion = file.getName();
```

```
            cancion = cancion.substring(0, cancion.length() –
            4);
            //Añado las canciones a la lista de elementos.
            elementos.add(cancion);
           }
        }

        //Añado a la lista el separador de videos y
        directorios.
        elementos.add("----------------------------------------
        ---------------------");

        //Recorro todos los elementos de la ruta actual y
        //si hay alguna directorio dentro del directorio
        actual:
        for( File archivo: rutaFila.listFiles())
        {
          if(archivo.isDirectory())
          //Lo formateo para diferenciarlo de los videos.
          elementos.add("      [" + archivo.getName() + "]");
        }

        //Ya que los datos se van insertando uno a uno
        alfabeticamente, quedarían al reves
        //hay que invertir el vector final para que se muestren
        ordenados alfabeticamente:

        int tamanyo=elementos.size();
        String temp;
        //Recorro hasta la mitad del vector y cambio el
        elemento en el que me encuentro por el mismo pero
        empezando por atrás.
        for (int i = 0 ; i < tamanyo/2 ; i++ ){
          temp = elementos.get(i);
          elementos.set(i, elementos.get(tamanyo-1-i));
          elementos.set((tamanyo-1-i), temp);
        }

        //Por último creo un adaptador para mostrar la lista y
        //le paso el array con los elementos tal cual los tiene
        que mostrar
        ArrayAdapter<String> listaArchivos= new
        ArrayAdapter<String>(this, R.layout.fila, elementos);
        setListAdapter(listaArchivos);
    }


    //A este método se le llama cuando hay que cargar la lista
    de nuevo,
    //si estamos en el directorio raiz de la tarjeta no haremos
    nada, pero si estamos en otra carpeta quiere decir que el
    usuario ha ido un paso atrás, y habrá que cambiar la ruta
    private void AntesDeRellenar() {

          int ultimo=0;
          int penultimo = 0;
```

```
            //Compruebo que la ruta no sea el directorio raiz.
            String casa = "/sdcard/";
            if(rutaActual.compareTo(casa) != 0)
            {
            //Localizo la penultima barra en la ruta para borrar
            la última carpeta.
            for ( int i = 0 ; i < rutaActual.length() ; i++)
            {
                  String barra = "/";
                  if(rutaActual.charAt(i) == barra.charAt(0))
                  {
                        penultimo = ultimo;
                        ultimo = i;
                  }
             }
              //Creo la nueva ruta eliminando la última carpeta
              rutaActual = rutaActual.substring(0, penultimo+1);
            //muestro en el TextView de la vista la nueva ruta
            cambiando la barra por ->
            TextViewRuta.setText
            (rutaActual.replace("/", " -> "));
          }
          //Ya con la ruta adecuada llamo al método rellenar.
          rellenar();
    }
}
```

Clase Reproductor.java

Es la clase principal, de esta, parte el programa y usa main.xml como capa para mostrar la pantalla principal con los tres botones para música, vídeos e imágenes.

El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones:

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

Inicio de la clase principal:

```
public class Reproductor extends Activity implements
OnClickListener {
```

Declaración de atributos globales:

```
    //Botones para controlar que quiere hacer el usuario
    private Button musica;
    private Button videos;
    private Button imagenes;
```

onCreate

```
    //Metodo onCreate sobrescrito de la clase Activity
    este método se invoca una vez se crea esta actividad.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //Utilizacion de la capa main.xml para mostrar la vista.
        setContentView(R.layout.main);


         //Inicialización de los botones, así como la
         adjudicación
         de sus vistas en la capa main
         musica = (Button) findViewById(R.id.musica);
```

```
        videos = (Button) findViewById(R.id.videos);
        imagenes = (Button) findViewById(R.id.imagenes);
        // Establecemos que la clase actual sea la que
        maneje los clics sobre estos botones
        musica.setOnClickListener(this);
        videos.setOnClickListener(this);
        imagenes.setOnClickListener(this);
    }
```

Método onClick

```
    //Este método se encarga de gestionar todos los eventos que
    ocurran al pulsar la pantalla.
    public void onClick(View v) {

        //Si se pulsa el botón música:
        if(v.equals(musica)){

            // Creación de un nuevo inent diciendole en que
            contexto estamos y que clase quiero lanzar,este
            caso música.
            Intent i = new Intent(this, Musica.class);
            // Inicio de la actividad.
            startActivity(i);

        }
        //Si se pulsa el botón videos:
        if(v.equals(videos)){

            // Creación de un nuevo inent diciendole en que
            contexto estamos
            //y que clase queremos lanzar en este caso videos
            Intent i = new Intent(this, ListaVideos.class);
            // Inicio de la actividad.
            startActivity(i);

        }
        //Si se pulsa el botón imágenes:
        if(v.equals(imagenes)){

            // Creación de un nuevo inent diciendole en que
            contexto estamos
            //y que clase queremos lanzar en este caso
            imágenes.
            Intent i = new Intent(this,
            CuadriculaImagenes.class);
            // Inicio de la actividad.
            startActivity(i);

        }
    }
}
```

Clase Video.java

Esta clase mostrará los vídeos. El código java correspondiente a la clase y explicado detenidamente mediante anotaciones es:

Package que contiene al proyecto:

```
package com.reproductor;
```

Importaciones

```
import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.ProgressBar;
```

Inicio de la clase principal:

```
public class Video extends Activity implements
SurfaceHolder.Callback , OnClickListener, Runnable{
```

Atributos globales

```
    //Enteros con el ancho y alto de la parte
    donde se mostrará el vídeo.
    private int anchoVideo;
    private int altoVideo;
    //MediaPlayer que gestionará el vídeo
    private MediaPlayer mp;
    //Superficie donde se mostrará el vídeo y su manejador.
    private SurfaceView vistaVideo;
    private SurfaceHolder manejadorVideo;
    //Ruta del vídeo en la memoria SD
    private String ruta;

    //Creación de los botones y barra de progreso
    que aparecerán en la pantalla
    private ImageButton pausarVideo;
    private ImageButton pararVideo;
    private ProgressBar progresoVideo;


    //Posicion actual del video, para saber donde se encuentra
    cuando lo paro.
    private int posicionActual = 0;
```

Método onCreate

```
 //Método onCreate sobrescrito de la clase Activity,
 este método se invoca una vez se crea esta actividad.

@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

   //Utilizacion de la capa video.xml para mostrar la
   vista.
   setContentView(R.layout.video);

   // Obtengo una referencia a todos los elementos
   generados en video.xml
   vistaVideo = (SurfaceView) findViewById(R.id.surface);
   pausarVideo = (ImageButton)
   findViewById(R.id.pausaVideo);
   pararVideo = (ImageButton)
   findViewById(R.id.pararVideo);
   progresoVideo = (ProgressBar)
   findViewById(R.id.progresoVideo);

   //Establezco los parámetros del manejador.
   manejadorVideo = vistaVideo.getHolder();
   manejadorVideo.addCallback(this);
   manejadorVideo.setType
   (SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

   // Establezco que los botones se mantengan a la espera
   de si son pulsados.
   pausarVideo.setOnClickListener(this);
   pararVideo.setOnClickListener(this);

   //Obtengo la ruta del vídeo, suministrada por la clase
   anterior.
   Bundle extras = getIntent().getExtras();
   ruta= extras.getString("1");
}
```

Método onClick

```
 public void onClick(View v) {

   // Si el usuario hace clic sobre el botón Reproducir.
   if(v.equals(pausarVideo)){

      // Detengo si se estaba reproduciendo algo
      if(mp != null && mp.isPlaying())
      {
           mp.pause();
           //Me guardo el dato de donde se encuentra la
           canción
           posicionActual = mp.getCurrentPosition();
           //Cambio la imagen del boton por la de
```

```
                reproducir,.
                pausarVideo.setImageDrawable(null);
                pausarVideo.setImageDrawable(getResources().
                getDrawable(R.drawable.reproducir));
                //Si no se estaba reproduciendo quiere decir que
                tengo que emprender la reproducción de nuevo
            }else{
                //Llevo el mediaPlayer hasta donde se quedó
                cuando le di a la pausarVideo.
                mp.seekTo(posicionActual);
                //Empiezo a reproducir y cambio la imagen
                a pausarVideo.
                mp.start();
                pausarVideo.setImageDrawable(null);
                pausarVideo.setImageDrawable
                (getResources().getDrawable(R.drawable.pausa));

            }

        }

        // Si el usuario ha hecho clic sobre el botón Detener
         y estaba reproduciendo.
        if(v.equals(pararVideo) && mp!=null){
            // Detengo la reproducción
            mp.stop();
            progresoVideo.setVisibility(ProgressBar.GONE);
            finish();
        }

    }
```

Método run

```
    //Manejador del hilo que se encarga de actulizar la barra.
    public void run() {
      //Creo dos int con la posicionActual y la duración total
      de la canción
      int posicionActual = 0;
      int total = mp.getDuration();
      // Mientras este reproduciendo y no haya llegado al
      final...
      while(mp!=null && posicionActual<total){
         try {
           // Espero un segundo...
           Thread.sleep(1000);
           // Obtengo la posición por la que se está
           reproduciendo. (En milisegundos)
           posicionActual = mp.getCurrentPosition();
           } catch (InterruptedException e) {
             return;
           } catch (Exception e){
                          return;
                     }
```

```
            // Hago que la barra de progreso referencie el
            valor actual.

           progresoVideo.setProgress(posicionActual);
            }
        }

    //método encargado de la preparación del vídeo
    private void prepararVideo(Integer Media) {

       try {
           // Creo el nuevo mediaPlayer con la ruta dada,
           establezco el manejador,
           //preparo el vídeo y por último lo reproduzco
           mp = new MediaPlayer();
           mp.setDataSource(ruta);
           mp.setDisplay(manejadorVideo);
           mp.prepare();
           mp.setAudioStreamType(AudioManager.STREAM_MUSIC);
           empezarVideo();

       } catch (Exception e) {

       }
    }
```

Método empezarVideo

```
    //método encargado de la reproducción
    private void empezarVideo() {

           //Establezco el tamaño de la pantalla
           manejadorVideo.setFixedSize(anchoVideo, altoVideo);

           //Hago visible la barra de progreso
           progresoVideo.setVisibility(ProgressBar.VISIBLE);

           // Establezco su valor actual al principio
           progresoVideo.setProgress(0);

           // Establezco que el valor del final sea el total
           de milisegundos que dura la canción
           progresoVideo.setMax(mp.getDuration());
           //Empieza
           mp.start();

           // Ejecuto un hilo que se encargará de actualizar la
           barra
           de progreso cada segundo...
           new Thread(this).start();
    }
```

Método surfaceCreated

```
//Cuando se crea la superficie donde se visualizará el vídeo
automáticamente llamo a preparar vídeo.
public void surfaceCreated(SurfaceHolder manejadorVideo) {

    prepararVideo(1);
}


//Método obligatorio por si la superficie cambiara.
(No es mi caso)
public void surfaceChanged(SurfaceHolder holder, int format,
       int width, int height) {
       // TODO Auto-generated method stub
 }

 //Método obligatorio por si la superficie desapareciera.
 (No es mi caso)
 public void surfaceDestroyed(SurfaceHolder holder) {
       // TODO Auto-generated method stub
 }
```

Cuadricula_imagenes.xml

```
//Cabeceras:
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

//Cuadrícula:
    <GridView
        android:id="@+id/sdcard"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:numColumns="auto_fit"
        android:verticalSpacing="0dp"
        android:horizontalSpacing="0dp"
        android:columnWidth="20dp"
        android:stretchMode="columnWidth"
        android:gravity="center" />

</FrameLayout>
```

fila.xml

```
//Cabeceras
<?xml version="1.0" encoding="utf-8"?>

//Para cada línea de las listas tanto de los vídeos como de las
canciones la configuración será:

<TextView android:id="@+id/text1"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="1000px"
    android:layout_height="fill_parent"
    android:textStyle="bold"
    android:textSize="20px"
    android:textColor="#ffffff"
    />
```

imagenes.xml

```
//Cabeceras:
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

      //Cada imagen que se mostrará tiene estas características.

      <ImageView android:id="@+id/mostrar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
      </ImageView>
</LinearLayout>
```

Lista_videos.xml

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

      <LinearLayout xmlns:android=
         "http://schemas.android.com/apk/res/android"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
         android:gravity="center">

               //Botón atrás
               <Button
                 android:id="@+id/AtrasVideos"
                 android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:background="@drawable/atras" >
               </Button>
               //Texto ruta vídeos.
               <TextView android:id="@+id/rutaVideos"
                   android:text="-> sdcard ->"
                   android_textSize="12px"
                   android:layout_width="fill_parent"
                   android:layout_height="wrap_content"/>

    </LinearLayout>

    //Lista
    <ListView android:id="@id/android:list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>
    //Texto que aparecerá si no se muestra ninguna canción.
    <TextView android:id="@id/android:empty"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         android:text="No se han encontrado vídeos."/>
</LinearLayout>
```

main.xml

```
//Cabeceras
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android=
      "http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:background="@drawable/fondo"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center">

      //Texto que muestra la palabra música.
      <TextView android:text="Música"
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:textColor="#ffffff"
            android:layout_height="wrap_content"
            android:textStyle="bold">
      </TextView>

      //Botón para acceder a la música.
      <Button android:id="@+id/musica"
            android:background="@drawable/entrar_musica"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content" >
      </Button>

      //Texto que muestra la palabra vídeos
      <TextView android:textStyle="bold"
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
            android:textColor="#ffffff"
            android:layout_height="wrap_content"
            android:text="Vídeos">
      </TextView>

      //Botón para acceder a los vídeos.
      <Button android:id="@+id/videos"
          android:background="@drawable/entrar_videos"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content" >
      </Button>

      //Texto que muestra la palabra Imágenes
      <TextView android:textStyle="bold"
          android:textColor="#ffffff"
          android:id="@+id/TextView03"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Imágenes">
      </TextView>
```

Anexo 16

```
      //Botón para acceder a los vídeos.
      <Button android:id="@+id/imagenes"
            android:background="@drawable/entrar_imagenes"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content">
      </Button>

</LinearLayout>
```

video.xml

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

      <LinearLayout android:orientation="horizontal"
            android:layout_height="350px"
            android:layout_width="320px">

            //Superficie donde se mostrará el vídeo.
            <SurfaceView android:id="@+id/surface"
                  android:layout_width="fill_parent"
                  android:layout_height="fill_parent" />

      </LinearLayout>

      //Botones para el manejo del vídeo.
      <LinearLayout android:orientation="horizontal"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:gravity="center">

            <ImageButton android:id="@+id/pausaVideo"
                  android:text="Pausa"
                  android:src="@drawable/pausa"
                  android:layout_height="wrap_content"
                  android:layout_width="wrap_content" />

            <ImageButton android:id="@+id/pararVideo"
                  android:text="Parar"
                  android:src="@drawable/parar"
                  android:layout_height="wrap_content"
                  android:layout_width="wrap_content" />

  </LinearLayout>

  //Barra de progreso
  <ProgressBar android:id="@+id/progresoVideo"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:max="100"
            android:progress="0"
            android:visibility="gone"/>

  </LinearLayout>
```

AndroidManifest.xml

```
 //Cabeceras.
<?xml version="1.0" encoding="utf-8"?>
<manifest
      xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.reproductor"
      android:versionCode="1"
      android:versionName="1.0">
     //Icono y nombre.
     <application android:icon="@drawable/icon"
        android:label="@string/app_name">

       //Actividad principal.
       <activity
         android:theme=
         @android:style/Theme.NoTitleBar.Fullscreen"
         android:name=".Reproductor"
         android:label="@string/app_name">
         <intent-filter>
             <action
               android:name="android.intent.action.MAIN" />
             <category
               android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
        </activity>

         //Otras actividades.
         <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
            android:name=".Musica"/>

        <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
           android:name=".Video"/>

         <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
            android:name=".ListaVideos"/>

         <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
            android:name=".CuadriculaImagenes"/>
          <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
            android:name=".Imagenes"/>

     </application>

     <uses-sdk android:minSdkVersion="4" />

</manifest>
```

Clase Intermedia.java

Esta es la primera que se crea, y la que hereda de Activity. Utiliza la capa menu1.xml posteriormente explicada.

Package que contiene al proyecto:

```
package com.sonidos;
```

Importaciones:

```
import com.sonidos.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

Inicio de la clase:

```
public class Intermedia extends Activity{
	//Método onCreate sobrescrito de la clase Activity,
	// a este método se llama una vez se crea la activity.
```

Método onCreate:

```
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		//Utilización de la vista menu1.xml
		 setContentView(R.layout.menu1);
	}
```

Método onCreateOptionsMenu:

```
	//Método sobrescrito de la clase Activity que se encarga
	de crear el menú.
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		// Adición de 1 opción, salir de la aplicación.
		// El primer valor de la opción es la colocación del
		mismo en la pantalla, el segundo es
		// el es un entero que después nos servirá para saber
		que opción se ha pulsado.
		// Por último encontramos el valor del texto de la
		opción.
		menu.add(0, 1, 0, "Salir del juego");
		return true; }
```

Método onMenuItemSelected:

```
        // Método sobrescrito de la clase Activity que pasa a la
        acción cuando se pulsa una opción del menú.
        public boolean onMenuItemSelected(int CaracteristicaID,
        MenuItem item) {

                    // Mediante getItemId se obtiene el valor de
                    la opción pulsada.
                    switch(item.getItemId()) {
                         // Si la opción pulsada es el salir,
                         se finaliza la aplicación.
                           case 1:
                             finish();
                             break;
                     }
                    return
                    super.onMenuItemSelected(CaracteristicaID,
                    item);
            }
```

Final de la clase:

```
}
```

menu1.xml

```
//Cabeceras
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:android=
  "http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"

  // Selección del fondo:
  android:background="@drawable/fondo">

// Desde aquí se le pasa el control total de la aplicación a la clase
Principal.java:
<com.sonidos.Principal
          android:id="@+id/Principal"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"/>

</AbsoluteLayout>
```

Clase Principal.java

Esta es la clase que se encarga de realizar todas las operaciones, así como la creación y destrucción de los hilos para mostrar los cambios en los colores del juego.

Package que contiene al proyecto:

```
package com.sonidos;
```

Importaciones:

```
import java.util.ArrayList;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.media.MediaPlayer;
import android.os.Handler;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
```

Inicio de la clase:

```
public class Principal extends View {
```

Declaración de atributos globales:

```
    //Contadores necesarios para la ejecución del juego.
    public int contadorColores = 0;
    public int contadorJugada = 0;
    public int contadorFinal = 0;
    public int contadorMostrar = 0;

    //Lista con todos los colores ordenados que se han
    mostrado hasta el momento.
    public ArrayList<String> lista = null;

    //Colores que se iluminarán cada vez que se pulse uno
    de ellos, o cuando el móvil los reproduzca desde la
    lista.
    public Drawable rojo = null;
    public Drawable azul = null;
    public Drawable verde = null;
    public Drawable amarillo = null;
    public Drawable fondo = null;

   //Sonidos a reproducir a la vez que se muestran los
   colores.
```

```
public MediaPlayer MPRojo = null;
public MediaPlayer MPAzul = null;
public MediaPlayer MPVerde = null;
public MediaPlayer MPAmarillo = null;
public MediaPlayer MPError = null;
public MediaPlayer MPIntro = null;

//Manejador de hilos.
final Handler miManejador = new Handler();
```

Constructor Principal:

```
//Método constructor de la clase View
public Principal(Context context, AttributeSet attrs) {
      super(context, attrs);

      //Inicialización de los colores, los recogemos de
      la carpeta drawable.
      rojo = context.getResources().
      getDrawable(R.drawable.rojo_claro4v);

      azul = context.getResources().
      getDrawable(R.drawable.azul_claro4v);

      verde = context.getResources().
      getDrawable(R.drawable.verde_clar o4v);

      amarillo = context.getResources().
      getDrawable(R.drawable.amarillo_c laro4v);

      fondo = context.getResources().
      getDrawable(R.drawable.fondo);

      //Inicialización de los sonidos.
      MPRojo = MediaPlayer.create
      (this.getContext(),R.raw.rojo);
      MPAzul = MediaPlayer.create
      (this.getContext(),R.raw.azul);
      MPVerde = MediaPlayer.create
      (this.getContext(),R.raw.verde);
      MPAmarillo = MediaPlayer.create
      (this.getContext(),R.raw.amarillo);
      MPError = MediaPlayer.create
      (this.getContext(),R.raw.error);
      MPIntro = MediaPlayer.create
      (this.getContext(),R.raw.intro);

       //Reporducción de la canción de introducción.
       MPIntro.start();
      //Mensaje de introducción, creación de un objeto del
      tipo alerta.
      AlertDialog alerta;
      //Inicialización de alerta.
      alerta = new AlertDialog.Builder
      (this.getContext()).create();
      //Titulo de la alerta.
```

```
        alerta.setTitle("Bienvenido ");
        //Botones de la alerta, en este caso solo
        uno. alerta.setButton("Empezar", new
        DialogInterface.OnClickListener() {
              //Al hacer click en el boton llamamos al
              método empezamos().
              public void onClick(DialogInterface dialogo, int
              con) {
                          empezamos();
                          return;
                    }
        });

        //Mostramos la alerta anteriormente creada.
        alerta.show();

  }
```

Método empezamos:

```
  //Este método es necesario porque no se puede crear una
  alerta dentro de otra,
  //así que hay que crear un método que se encargue de
  manejar esta.
  public void empezamos()
  {
        //Mensaje de instrucciones, creación de un objeto
        del tipo alerta.
        AlertDialog alerta2;
        //Inicialización de alerta. alerta2 = new
        AlertDialog.Builder(this.getContext()).create(
        );
        //Titulo de la alerta.
        alerta2.setTitle("Instrucciones ");
         //Mensaje de la alerta.
        alerta2.setMessage("El juego consiste en seguir
        la combinación de colores y sonidos propuesta" +
        " por el móvil, cada vez se irá añadiendo
        una combinación mas, " +
        "el juego finalizará cuando no consigas
        recordarlos todos en el orden propuesto.");

         //Botones de la alerta, en este caso dos.

        //Primer botón:
        alerta2.setButton("Jugar", new
        DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialogo, int
              con) {
                    //Parada de la canción de introducción, si
                    es que todavía está sonando.
                    MPIntro.stop();
                    //Inicialización de los
                    contadores.
                    contadorColores = 0;
                    contadorJugada = 0;
```

```
                    contadorFinal = 0;
                    contadorMostrar = 0;
                    //Inicialización de la lista.
                    lista = new ArrayList<String>();
                    //Llamada al método generador
                    de combinaciones.
                    Generador();
              }});

        //Segundo botón:
        alerta2.setButton2("Salir", new
        DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialogo, int
              con) {
                    //Llamada al método saliendo() para cerrar
                    la aplicación.
                    saliendo();
              }});

      //Se muestra la alerta.
      alerta2.show();
}
```

Método generador:

```
//Método generador.
public void Generador()
{
      //Creación de un string inicializado a cadena vacía,
      //el cual servirá para almacenar provisionalmente
      el color seleccionado aleatoriamente.
      String es = "";
      //El contador final pasa a tener un valor
      mas. contadorFinal= contadorFinal + 1;
      //El contador de la jugada pasa a valer lo
      mismo que el contador final,
      //es decir el numero de colores que se han
      mostrado hasta el momento,
      //Mas tarde este contador nos servirá para saber
      el numero de veces que el jugador tiene que
      pulsar la pantalla
      //Y así cuando el contador jugada llegue a 0 volver a
      mostrar todos los colores desde el principio
      añadiendo una mas a la lista.
      contadorJugada=contadorFinal;

      //Obtenemos un número aleatorio del 0 al 3 y
      lo guardamos en la variable x.
      int x = (int)( Math.random() * 4);
      //Mediante este switch dependiendo del valor
      obtenido aleatoriamente se asignará al string es.
      switch(x)
      {
         case 0:
               es = "rojo";
```

```
                break;
            case 1:
                es = "azul";
                break;
            case 2:
                es = "verde";
                break;
            case 3:
                es = "amarillo";
                break;
        }
        //Se añade el valor obtenido a la
        lista. lista.add(es);
        //Llamada al método mostrador, que como su propio
        nombre indica se encargará de
        //mostrar los diferentes colores apoyado en
        otros métodos.
        Mostrador();
}
```

Método Mostrador:

```
//Método mostrador:
public void Mostrador()
{
        //Creación de un hilo secundario para mostrar
        los diferentes colores,
        //Es necesaria la creación de un hilo en este
        punto por que si no cuando
        //se mostrara el mismo color dos veces seguidas solo
        se vería una
        //debido a que el refresco de las imágenes en
        android solo se puede hacer a la salida de los
        métodos.
        Thread t = new Thread() {
            //El hilo usa el manejador con pasoMostrador.
             public void run() {
                 miManejador.post(pasoMostrador);
             }
         };
        //Empezamos el
        hilo. t.start();
}
```

Runnable pasoMostrador:

```
//Creación y ejecución de pasoMostrador
final Runnable pasoMostrador = new Runnable() {
        public void run() {
        //Creación de un string donde posteriormente
        se almacenará un color.
        String s;
        //Si la talla de la lista es mayor que el contador
```

```
        de colores
        //quiere decir que todavía queda algun color
        que reproducir.
        if(lista.size() > contadorColores ){
              //Obtenemos el color
              correspondiente. s =
              lista.get(contadorColores);
              //Dependiendo del color obtenido llamaremos a
              los
              métodos
              //que se encargan de manejar los diferentes
              hilos para cada color.
              if(s == "rojo")
              {
                    hiloRojoGenerado();
              }
              if(s == "azul")
              {
                        hiloAzulGenerado();
              }
              if(s == "verde")
              {
                    hiloVerdeGenerado();
              }
              if(s == "amarillo")
              {
                    hiloAmarilloGenerado();
              }
              //En el caso de que se haya llegado a la talla
              máxima
              //reinicializamos el contador de los colores a 0
              para volver a contarlos la próxima vez.
              }else{contadorColores=0;}
        }
};
```

Método hiloRojoGenerado:

```
//El manejo de estos hilos es igual para los cuatro
colores,
//así que lo explicaré para uno solo, se entiende que
los demás hacen lo mismo, pero con diferente color.

//Método hiloRojoGenerado.
protected void hiloRojoGenerado()
{
      //Creación del hilo.
      Thread t = new Thread()
      {
            public void run() {
            //El manejador se encargará de
            ejecutar refrescarRojoGenerado.
              miManejador.post(refrescarRojo);
          }
      };
```

```
        //Inicio del
        hilo. t.start();
}
```

Runnable refrescarRojo:

```
//Creación y ejecución de refrescarRojo.
final Runnable refrescarRojo = new Runnable() {
       public void run() {
            //Inicialización del sonido correspondiente
            al color rojo.
            MPRojo.start();
            //Cambio del color del fondo por otro en el que
            se
            resalta el color rojo.
            setBackgroundDrawable(rojo);
            //Suspensión del hilo el tiempo suficiente
            para que se pueda ver el color con claridad,
            //y para que finalice el sonido
            try { Thread.sleep(300);
                 }catch (InterruptedException e) {
                 e.printStackTrace(
                         );
            }
            //Adición de una unidad al contador de
            los colores,
            //para que al volver al método
            Mostrador() nos muestre el siguiente
            color en la lista
             contadorColores+
            +; Mostrador();
                 }
       };
```

Método hiloAzulGenerado:

```
//Idem hiloRojo.
protected void hiloAzulGenerado()
{
       Thread t = new Thread() {
            public void run() {
              miManejador.post(refrescarAzul);
            }
       };
       t.start();
}
```

Runnable refrescarAzul:

```
//Idem refrescarRojo.
final Runnable refrescarAzul = new Runnable() {
       public void run() {
            MPAzul.start();
```

```
                setBackgroundDrawable(azul
                ); try {
                            Thread.sleep(300);
                      } catch (InterruptedException e) {
                            e.printStackTrace();
                      }
                contadorColores+
                +; Mostrador();
           }
      };
```

Método hiloVerdeGenerado:

```
      //Idem hiloRojo.
      protected void hiloVerdeGenerado()
     {
         Thread t = new Thread() {
               public void run() {
                     miManejador.post(refrescarVerde);
                }
          };
               t.start();
          }
```

Runnable refrescarVerde:

```
       //Idem refrescarRojo.
        final Runnable refrescarVerde = new Runnable() {
              public void run() {
                   MPVerde.start();
                   setBackgroundDrawable(verde
                   ); try {
                               Thread.sleep(300);
                         } catch (InterruptedException e) {
                               // TODO Auto-generated catch
                               block e.printStackTrace();
                         }
                   contadorColores+
                   +; Mostrador();

              }
          };
```

Método hiloAmarilloGenerado:

```
     //Idem hiloRojo.
     protected void hiloAmarilloGenerado()
     {
           Thread t = new Thread() {
                 public void run() {
                       miManejador.post(refrescarAmarillo);
                 }
```

```
        };
        t.start();
    }
```

Runnable refrescarAmarillo:

```
    //Idem refrescarRojo.
    final Runnable refrescarAmarillo = new Runnable() {
        public void run() {
            MPAmarillo.start();
            setBackgroundDrawable(amarillo
            ); try {
                Thread.sleep(300);
                } catch (InterruptedException e) {
                // TODO Auto-generated catch
                block e.printStackTrace();
                }
             contadorColores++;
             Mostrador();
        }
    };
```

Método onTouchEvent:

```
    //Este método está sobrescrito de la clase View,
    //y es el que se encarga de capturar los eventos ocurridos
    en la pantalla mediante el contacto con ella.
    //Se ejecuta cuando se toca la pantalla.
    public boolean onTouchEvent(MotionEvent event) {

        //Obtención de las coordenadas x e y de la pulsación
        en la pantalla.
        float x = event.getX();
        float y = event.getY();

        //String que recoge el color correspondiente de
        la lista.
        String ColorValido;

        //ROJO
        //En el caso de que la x sea menor que 160 y la y
        sea menor que 240 nos encontramos en el área del
        rojo. if(x<160 && y<240)
        {
            //Obtención del color correspondiente al
            contador mostrar en la lista,
            //el cual va aumentando cada vez que se muestra
            un color.
            ColorValido = lista.get(contadorMostrar);
            //Si el color obtenido es el rojo quiere decir
            que
            el jugador ha pulsado correctamente.
            if(ColorValido == "rojo")
            {
```

```
                //Lanzamiento del hilo que permitirá
                el cambio del color y la
                reproducción del sonido.
                hiloRojoPulsado();
                //Disminución del contador de jugadas,
                puesto que ya queda un color menos.
                contadorJugada--;
                //Aumento del contador mostrar
                //para que la próxima vez que se pulse
                la pantalla obtenga el color siguiente
                de la lista.
                contadorMostrar++;
        }
        //En el caso de que el color pulsado no se
        corresponda con el que en ese momento tocaba en
        la lista
        //Se realiza una llamada al método fin.
        else
        {
                fin();
        }

        }
        //AZUL
        //Idem rojo pero cambiando los valores de
        la pulsación de la pantalla y la llamada al
        hilo correspondiente.
        if(x<160 && y > 240)
        {
                ColorValido = lista.get(contadorMostrar);
                if(ColorValido == "azul")
                {
                        hiloAzulPulsado();
                        contadorJugada--;
                        contadorMostrar++;
                }else{
                fin();
                }
        }
        //VERDE
        //Ídem rojo pero cambiando los valores de
        la pulsación de la pantalla y la llamada al
        hilo correspondiente.
        if( x > 160 && y < 240 )
        {
                ColorValido = lista.get(contadorMostrar);
                if(ColorValido == "verde")
                {
                        hiloVerdePulsado();
                        contadorJugada--;
                        contadorMostrar++;
                }else{
                        fin();
                }
        }
```

```
                //AMARILLO
                //Ídem rojo pero cambiando los valores de
                la pulsacion de la pantalla y la llamada al
                hilo correspondiente.
                if( x > 160 && y > 240 )
                {
                      ColorValido = lista.get(contadorMostrar);
                      if(ColorValido == "amarillo")
                      {
                            hiloAmarilloPulsado
                            (); contadorJugada-
                            -;
                            contadorMostrar++;
                      }else{
                            fin();
                      }
                }
                return super.onTouchEvent(event);
          }
```

Método hiloRojoPulsado:

```
    //El código tanto de estos cuatro métodos como de sus
    manejadores podría ser compartido con los hilos generados,
    //si no fuera porque al finalizar estos hay que realizar
    una llamada al método ultima pos.

    //Método hiloRojoPulsado.
    protected void hiloRojoPulsado()
    {
          //Creación del hilo.
          Thread t = new Thread()
          {
                public void run() {
                //El manejador se encargará de
                ejecutar refrescarRojoPulsado.
                  miManejador.post(refrescarRojoPulsado);
                }
           };
           //Inicio del
           hilo. t.start();
    }
```

Runnable refrescarRojoPulsado:

```
    //Creación y ejecución de refrescarRojoPulsado.
    final Runnable refrescarRojoPulsado = new Runnable() {
          public void run() {
                //Inicialización del sonido correspondiente
     al color rojo.
                 MPRojo.start();
                //Cambio del color del fondo por otro en el que
                se resalta el color rojo.
```

```
                setBackgroundDrawable(rojo);
              //Suspensión del hilo el tiempo suficiente para
              que se pueda ver el color con claridad,
              //y para que finalize el sonido. Esta vez la
              suspensión es de solo 20 milésimas de segundo,
              //he incluso se podría llegar a omitir, ya que
              ese tiempo es mucho menor de que puede tardar el
              jugador
              //en llegar hasta el siguiente color a
              pulsar, pero he comprobado que poniendo la
              suspensión la estabilidad
               //del juego es mayor.
               try {
                   Thread.sleep(20);
                   } catch (InterruptedException e) {
                   e.printStackTrace();
                   }
                   //Llamada al método
                   ultimaPos. ultimaPos();
          }
    };
```

Método hiloAzulPulsado:

```
//Idem hiloRojoPulsado.
 protected void hiloAzulPulsado()
{
      Thread t = new Thread() {
            public void run() {
                  miManejador.post(refrescarAzulPulsado);
             }
       };
       t.start();
}
```

Runnable refrescarAzulPulsado:

```
final Runnable refrescarAzulPulsado = new Runnable() {
      public void run()
               {
            setBackgroundDrawable(azul
             ); MPAzul.start();
             try
              {
                  Thread.sleep(20);
                  } catch (InterruptedException e) {
                  // TODO Auto-generated catch
                  block e.printStackTrace();
            }
            ultimaPos();
      }
};
```

Método hiloVerdePulsado:

```
//Idem hiloRojoPulsado.
protected void hiloVerdePulsado()
{
      Thread t = new Thread() { public void run()
            {
            miManejador.post(refrescarVerdePulsado
            );
                  }
      };
       t.start();
}
```

Runnable refrescarVerdePulsado:

```
final Runnable refrescarVerdePulsado = new Runnable() {
      public void run() {
                  setBackgroundDrawable(verde
                  ); MPVerde.start();
                  try {
                  Thread.sleep(20);
                  } catch (InterruptedException e) {
                        // TODO Auto-generated catch
                        block e.printStackTrace();
            }
            ultimaPos();
       }
};
```

Método hiloAmarilloPulsado:

```
//Idem hiloRojoPulsado.
protected void hiloAmarilloPulsado()
{
       Thread t = new Thread() {
             public void run() {
                   miManejador.post(refrescarAmarilloPulsado
                   );
                  }
      };
      t.start();
}
```

Runnable refrescarAmarilloPulsado:

```
final Runnable refrescarAmarilloPulsado = new Runnable() {
      public void run() {
            setBackgroundDrawable(amarillo
            ); MPAmarillo.start();
            try {
                  Thread.sleep(20);
                  } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
                ultimaPos();
        }
};
```

Método ultimaPos:

```
//Método ultimaPos:
//Para la ejecucion del codigo que tendría que ir en
este método tambien hace falta
//la creación de un hilo, porque si no el último colo
pulsado no se mostaría cuando lo pulsa,
//si no cuando va a empezar otra vez a generar los
colores, es decir, cuando acaba el método.
public void ultimaPos()
{
      //Creación del hilo.
      Thread t = new
      Thread() {
            public void run() {
                  //El manejador se encargará de
                  ejecutar pasoGenerador.
                  miManejador.post(pasoGenerador);
            }
      };
      //Comienza el
      Hilo. t.start();

}
```

Runnable pasoGenerador:

```
final Runnable pasoGenerador = new Runnable()
                        {
      public void run() {
            //Si el contador jugada ha llegado a 0
            quiere decir que no quedan colores por
            pulsar
            //y por lo tanto se puede generar uno nuevo,
            añadirlo a la lista y volver a mostrarla.
             if(contadorJugada <= 0)
             {
                  //Espera de medio segundo para no
                  confundir al jugador.
                  try {
                        Thread.sleep(500
                              );
                          } catch (InterruptedException e) {
                              e.printStackTrace();
                   }
                  //Reseteo del contador mostrar para que
                  la próxima vez empiece a mostrar
                   //la lista otra vez desde la posición 0.
                   contadorMostrar =
```

```
                        0;
                    //Llamada al método
                generador. Generador();
            }
        }
};
```

Método onDraw:

```
//Este método se sobrescribe de la clase View, y lo que
hace
es que cada vez que hay un cambio en la pantalla se
ejecuta.
//En este caso, al ejecutarse, lo que hace es volver a
poner el fondo como estaba al principio.
@Override
protected void onDraw(Canvas canvas)
{
      super.onDraw(canvas);
      setBackgroundDrawable(fondo);
}
```

Método fin:

```
//Método fin.
//Como su propio nombre indica, se ejecuta al finalizar
el juego.
public void fin()
{
      //Ejecución de un sonido de
      error. MPError.start();
      //Entero donde se guardará la puntuación para
      después mostrarla.
      int
      puntuacion;

      //Si no se ha acertado ningún color la puntuación es
      0,
      //en caso contrario es el contador final
      multiplicado por 4
      if(contadorFinal==1)
      puntuacion=0;
      else
      puntuación = contadorFinal*4;

      //Creación de una alerta:
      AlertDialog alerta;
      alerta = new
      AlertDialog.Builder(this.getContext()).create();
      alerta.setTitle("Fin del juego");
      //En el mensaje de la alerta se muestra la
      puntuación obtenida por el jugador.
      alerta.setMessage("Te has equivocado! \nHas obtenido
      una puntuacion de: "+ puntuacion +" puntos. \n¿Deseas
      volver a jugar?");
```

```
            //Si el jugador decide volver a jugar, se lleva a
            cabo una reinicialización de los contadores y
            //de la lista posteriormente empieza el juego otra
            vez con el método generar.
             alerta.setButton("SI", new
             DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialogo,
                   int con) {
                         contadorColores = 0;
                         contadorJugada = 0;
                         contadorFinal = 0; contadorMostrar = 0;
                         lista = new ArrayList<String>();
                         Generador();
                         return;
                   }
             });

             //Si el jugador ya no quiere jugar mas, se llama al
             método saliendo, para cerrar la
             aplicación. alerta.setButton2("NO, salir
             del juego", new
             DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialogo,
                   int con) { saliendo();}
                   });
            //Se muestra la alerta.
            alerta.show();
      }
```

Método saliendo:

```
      //Método saliendo, cierra la aplicación.
      public void saliendo()
      {
            System.exit(0);
      }
```

Final de la clase:

```
}
```

Clase Piano.java

Esta clase contiene toda la información sobre el piano.

Package que contiene al proyecto:

```
package com. Piano;
```

Importaciones

```
import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageButton;
```

Inicio de la clase

```
public class Piano extends Activity implements
View.OnTouchListener{
```

Declaración de atributos globales:

```
    //MediaPlayer que se encargará de la reprodución de los
    diferentes sonidos.
     private MediaPlayer mp;

     //ImageButtons que dan forma a las diferentes notas,
     //como la palabra do está reservada,
     la primera nota se llama do_primero
     private ImageButton do_primero ,re, mi, fa, sol, la, si,
     do_octava, do_sostenido;
     private ImageButton re_sostenido, fa_sostenido,
      sol_sostenido, la_sostenido;
```

Metodo onCreate

```
    //Metodo onCreate sobrescrito de la clase Activity, este
      método se invoca una vez se crea esta actividad.
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        //Utilizacion de la capa main.xml para mostrar la vista.
        setContentView(R.layout.main);
        //Inicialización del mediaPlayer
        mp = new MediaPlayer();

        //Inicialización de todos los botones,
         así como la adjudicación de sus vistas en la capa main
        do_primero = (ImageButton)
           findViewById(R.id.do_primero);
        re = (ImageButton) findViewById(R.id.re);
```

```
mi = (ImageButton) findViewById(R.id.mi);
fa = (ImageButton) findViewById(R.id.fa);
sol = (ImageButton) findViewById(R.id.sol);
la = (ImageButton) findViewById(R.id.la);
si = (ImageButton) findViewById(R.id.si);
do_octava = (ImageButton)
   findViewById(R.id.do_octava);
do_sostenido = (ImageButton)
   findViewById(R.id.do_sostenido);
re_sostenido = (ImageButton)
   findViewById(R.id.re_sostenido);
fa_sostenido = (ImageButton)
   findViewById(R.id.fa_sostenido);
sol_sostenido = (ImageButton)
   findViewById(R.id.sol_sostenido);
la_sostenido = (ImageButton)
   findViewById(R.id.la_sostenido);

//Almacenamiento de los diferentes valores que tiene
   cada botón, que son: 1º el sonido,
//2º la imagen cuando está pulsado,
3º la imagens cuando no está pulsado.
do_primero.setTag(new int[] {R.raw.do_primero,
   R.drawable.do_p,R.drawable.do_primero});
re.setTag(new int[]
{R.raw.re, R.drawable.re_p, R.drawable.re});
mi.setTag(new int[]
{R.raw.mi, R.drawable.mi_p, R.drawable.mi});
fa.setTag(new int[]
{R.raw.fa, R.drawable.fa_p, R.drawable.fa});
sol.setTag(new int[]
{R.raw.sol, R.drawable.sol_p, R.drawable.sol});
la.setTag(new int[]
{R.raw.la, R.drawable.la_p, R.drawable.la});
si.setTag(new int[]
{R.raw.si, R.drawable.si_p, R.drawable.si});

do_octava.setTag(new int[]{R.raw.do_octava,
   R.drawable.do_octava_p, R.drawable.do_octava});
do_sostenido.setTag(new int[]{R.raw.do_sostenido,
   R.drawable.do_sostenido, R.drawable.do_sostenido});
re_sostenido.setTag(new int[]{R.raw.re_sostenido,
   R.drawable.re_sostenido, R.drawable.re_sostenido});
fa_sostenido.setTag(new int[]{R.raw.fa_sostenido,
   R.drawable.fa_sostenido, R.drawable.fa_sostenido});
sol_sostenido.setTag(new int[]{R.raw.sol_sostenido,
   R.drawable.sol_sostenido, R.drawable.sol_sostenido});
la_sostenido.setTag(new int[]{R.raw.la_sostenido,
   R.drawable.la_sostenido, R.drawable.la_sostenido});

//Preparo los diferentes botones para que cuando sean
   Pulsados ocurra un envento.
do_primero.setOnTouchListener(this);
re.setOnTouchListener(this);
mi.setOnTouchListener(this);
fa.setOnTouchListener(this);
```

```
        sol.setOnTouchListener(this);
        la.setOnTouchListener(this);
        si.setOnTouchListener(this);
        do_octava.setOnTouchListener(this);
        do_sostenido.setOnTouchListener(this);
        re_sostenido.setOnTouchListener(this);
        fa_sostenido.setOnTouchListener(this);
        sol_sostenido.setOnTouchListener(this);
        la_sostenido.setOnTouchListener(this);

    }
```

Método onTouch

```
    //onTouch es el manejador de los eventos creados con
      anterioridad en vista recojo quien ha pulsado el boton, y
      en movimiento como ha sido pulsado.
    public boolean  onTouch(View vista, MotionEvent movimiento)
    {
        //recojo la acción: pulsado, soltado...
        int accion = movimiento.getAction();
        //El método devolverá false si la acción no se ha
           realizado correctamente.
        boolean devuelvo = false;

        //Si vista es un ImageButton entonces:
        if (vista instanceof ImageButton) {
            //Hago un cast a ImageButton de vista
            ImageButton boton = (ImageButton) vista;
            //Y se lo paso junto con la accion al
                manejador del piano
            devuelvo = manejadorPiano(boton, accion);
        }
        return devuelvo;
    }
```

Método manejadorPiano

```
    //El manejador se encarga de reproducir el sonido y de
      cambiar el color de la nota:
    private boolean manejadorPiano(ImageButton boton, int
accion) {
        //El método devolverá false si la acción no se ha
           realizado correctamente.
        boolean devuelvo = false;
        //Guardo en un objeto los datos correspondientes a ese
           boton, es decir, sonido, imagen pulsado y sin pulsar.
        Object ob = boton.getTag();
        //Si el objeto no está vacío, y es un vector de
           intentonces:
        if (ob != null) {
            if (ob instanceof int[]) {
                //Hago un cast del objeto a vector de int
                int[] tag = (int[]) ob;
                //Si la acción que ha ocurrido es que la tecla
```

```
                ha sido pulsada:
                if (accion == MotionEvent.ACTION_DOWN) {
                  //reproduzco el sonido.
                  mp.release();
                  mp = MediaPlayer.create(Piano.this, tag[0]);
                  mp.start();
                  //Cambio la imagen a pulsado
                  boton.setImageResource(tag[1]);
                  //Si la acción es que la tecla ha sido
                  soltada:
                } else if (accion == MotionEvent.ACTION_UP) {
                //Vuelvo a poner la imagen normal.
                boton.setImageResource(tag[2]);
                }
            }
        }
        return devuelvo;
    }
```

main.xml

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

    <ImageView android:id="@+id/iv"
               android:layout_width="480px"
               android:layout_height="46px"
               android:src="@drawable/header"
               android:layout_x="0px"
               android:layout_y="0px"/>

     //Botones de tipo imagen que contienen cada una de las
     notas:

    <ImageButton android:id="@+id/do_primero"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:src="@drawable/do_primero"
            android:layout_x="0px"
            android:layout_y="46px"/>

    <ImageButton android:id="@+id/re"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:src="@drawable/re"
            android:layout_x="57px"
            android:layout_y="46px"/>

    <ImageButton android:id="@+id/mi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:src="@drawable/mi"
            android:layout_x="117px"
            android:layout_y="46px"/>

    <ImageButton android:id="@+id/fa"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:src="@drawable/fa"
            android:layout_x="177px"
            android:layout_y="46px"/>
```

Anexo 23

```
<ImageButton android:id="@+id/sol"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:src="@drawable/sol"
        android:layout_x="236px"
        android:layout_y="46px"/>

<ImageButton android:id="@+id/la"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:src="@drawable/la"
        android:layout_x="295px"
        android:layout_y="46px"/>

<ImageButton android:id="@+id/si"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:src="@drawable/si"
        android:layout_x="354px"
        android:layout_y="46px"/>

<ImageButton android:id="@+id/do_octava"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:src="@drawable/do_octava"
        android:layout_x="413px"
        android:layout_y="46px"/>

<ImageButton android:id="@+id/do_sostenido"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:layout_x="32px"
        android:layout_y="46px"
        android:src="@drawable/do_sostenido"/>

<ImageButton android:id="@+id/re_sostenido"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="0px"
        android:layout_margin="0px"
        android:layout_x="105px"
        android:layout_y="46px"
        android:src="@drawable/re_sostenido"/>

<ImageButton android:id="@+id/fa_sostenido"
        android:layout_width="wrap_content"
```

Anexo 23

```
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:layout_x="209px"
            android:layout_y="46px"
            android:src="@drawable/fa_sostenido"/>

    <ImageButton android:id="@+id/sol_sostenido"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:layout_x="278px"
            android:layout_y="46px"
            android:src="@drawable/sol_sostenido"/>

    <ImageButton android:id="@+id/la_sostenido"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="0px"
            android:layout_margin="0px"
            android:layout_x="344px"
            android:layout_y="46px"
            android:src="@drawable/la_sostenido"/>

</AbsoluteLayout>
```

AndroidManifest.xml

```
//Cabeceras.
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
      "http://schemas.android.com/apk/res/android"
       android:versionCode="1"
       android:versionName="1.0" package="com.Piano">
       //Inicio de la aplicación.
      <application android:icon="@drawable/icon"
         android:label="@string/app_name">
         //Actividad principal.
         <activity android:theme=
            "@android:style/Theme.NoTitleBar.Fullscreen"
            android:screenOrientation="landscape"
            android:configChanges="orientation|keyboardHidden"
            android:name=".Piano"
            android:label="@string/app_name">
            <intent-filter>
                <action
                   android:name="android.intent.action.MAIN" />
                 <category android:name=
                   "android.intent.category.LAUNCHER" />
             </intent-filter>

         </activity>

    </application>
    <uses-sdk android:minSdkVersion="8" />

</manifest>
```

Extracto del tutorial notepad.

Step 1

Create a new Android project using the sources from `Notepadv2` under the `NotepadCodeLab` folder, just like you did for the first exercise. If you see an error about `AndroidManifest.xml`, or some problems related to an `android.zip` file, right click on the project and select **Android Tools** > **Fix Project Properties**.

Open the `Notepadv2` project and take a look around:

- Open and look at the `strings.xml` file under `res/values` — there are several new strings which we will use for our new functionality
- Also, open and take a look at the top of the `Notepadv2` class, you will notice several new constants have been defined along with a new `mNotesCursor` field used to hold the cursor we are using.
- Note also that the `fillData()` method has a few more comments and now uses the new field to store the notes Cursor. The `onCreate()` method is unchanged from the first exercise. Also notice that the member field used to store the notes Cursor is now called `mNotesCursor`. The `m` denotes a member field and is part of the Android coding style standards.
- There are also a couple of new overridden methods (`onCreateContextMenu()`, `onContextItemSelected()`, `onListItemClick()` and `onActivityResult()`) which we will be filling in below.

Step 2

Context menus should always be used when performing actions upon specific elements in the UI. When you register a View to a context menu, the context menu is revealed by performing a "long-click" on the UI component (press and hold the touchscreen or highlight and hold down the selection key for about two seconds).

First, let's create the context menu that will allow users to delete individual notes. Open the Notepadv2 class.

1. In order for each list item in the ListView to register for the context menu, we call `registerForContextMenu()` and pass it our ListView. So, at the very end of the `onCreate()` method add this line:

   ```
   registerForContextMenu(getListView());
   ```

   Because our Activity extends the ListActivity class, `getListView()` will return us the local ListView object for the Activity. Now, each list item in this ListView will activate the context menu.

2. Now fill in the `onCreateContextMenu()` method. This callback is similar to the other menu callback used for the options menu. Here, we add just one line, which will add a menu item to delete a note. Call `menu.add()` like so:

```
public boolean onCreateContextMenu(Menu menu, View v
        ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, DELETE_ID, 0, R.string.menu_delete);
}
```

   The `onCreateContextMenu()` callback passes some other information in addition to the Menu object, such as the View that has been triggered for the menu and an extra object that may contain additional information about the object selected. However, we don't care about these here, because we only have one kind of object in the Activity that uses context menus. In the next step, we'll handle the menu item selection.

Step 3

Now that the we've registered our ListView for a context menu and defined our context menu item, we need to handle the callback when it is selected. For this, we need to identify the list ID of the selected item, then delete it. So fill in the `onContextItemSelected()` method like this:

```
public boolean onContextItemSelected(MenuItem item) {
    switch(item.getItemId()) {
    case DELETE_ID:
        AdapterContextMenuInfo info = (AdapterContextMenuInfo)
item.getMenuInfo();
        mDbHelper.deleteNote(info.id);
        fillData();
        return true;
    }
    return super.onContextItemSelected(item);
}
```

Here, we retrieve the `AdapterContextMenuInfo` with `getMenuInfo()`. The *id* field of this object tells us the position of the item in the ListView. We then pass this to the `deleteNote()` method of our NotesDbAdapter and the note is deleted. That's it for the context menu — notes can now be deleted.

Step 4

Starting Other Activities

In this example our Intent uses a class name specifically. As well as starting intents in classes we already know about, be they in our own application or another application, we can also create Intents without knowing exactly which application will handle it.

For example, we might want to open a page in a browser, and for this we still use an Intent. But instead of specifying a class to handle it, we use a predefined Intent constant,

and a content URI that describes what we want to do. See `android.content.Intent` for more information.

Fill in the body of the `createNote()` method:

Create a new `Intent` to create a note (`ACTIVITY_CREATE`) using the `NoteEdit` class. Then fire the Intent using the `startActivityForResult()` method call:

```
Intent i = new Intent(this, NoteEdit.class);
startActivityForResult(i, ACTIVITY_CREATE);
```

This form of the Intent call targets a specific class in our Activity, in this case `NoteEdit`. Since the Intent class will need to communicate with the Android operating system to route requests, we also have to provide a Context (`this`).

The `startActivityForResult()` method fires the Intent in a way that causes a method in our Activity to be called when the new Activity is completed. The method in our Activity that receives the callback is called `onActivityResult()` and we will implement it in a later step. The other way to call an Activity is using `startActivity()` but this is a "fire-and-forget" way of calling it — in this manner, our Activity is not informed when the Activity is completed, and there is no way to return result information from the called Activity with `startActivity()`.

Don't worry about the fact that `NoteEdit` doesn't exist yet, we will fix that soon.

Step 5

Fill in the body of the `onListItemClick()` override.

`onListItemClick()` is a callback method that we'll override. It is called when the user selects an item from the list. It is passed four parameters: the `ListView` object it was invoked from, the `View` inside the `ListView` that was clicked on, the `position` in the list that was clicked, and the `mRowId` of the item that was clicked. In this instance we can ignore the first two parameters (we only have one `ListView` it could be), and we ignore the `mRowId` as well. All we are interested in is the `position` that the user selected. We use this to get the data from the correct row, and bundle it up to send to the `NoteEdit` Activity.

In our implementation of the callback, the method creates an `Intent` to edit the note using the `NoteEdit` class. It then adds data into the extras Bundle of the Intent, which will be passed to the called Activity. We use it to pass in the title and body text, and the `mRowId` for the note we are editing. Finally, it will fire the Intent using the `startActivityForResult()` method call. Here's the code that belongs in `onListItemClick()`:

```
super.onListItemClick(l, v, position, id);
Cursor c = mNotesCursor;
c.moveToPosition(position);
Intent i = new Intent(this, NoteEdit.class);
i.putExtra(NotesDbAdapter.KEY_ROWID, id);
```

```
i.putExtra(NotesDbAdapter.KEY_TITLE, c.getString(
        c.getColumnIndexOrThrow(NotesDbAdapter.KEY_TITLE)));
i.putExtra(NotesDbAdapter.KEY_BODY, c.getString(
        c.getColumnIndexOrThrow(NotesDbAdapter.KEY_BODY)));
startActivityForResult(i, ACTIVITY_EDIT);
```

- `putExtra()` is the method to add items into the extras Bundle to pass in to intent invocations. Here, we are using the Bundle to pass in the title, body and mRowId of the note we want to edit.
- The details of the note are pulled out from our query Cursor, which we move to the proper position for the element that was selected in the list, with the `moveToPosition()` method.
- With the extras added to the Intent, we invoke the Intent on the `NoteEdit` class by passing `startActivityForResult()` the Intent and the request code. (The request code will be returned to `onActivityResult` as the `requestCode` parameter.)

Note: We assign the mNotesCursor field to a local variable at the start of the method. This is done as an optimization of the Android code. Accessing a local variable is much more efficient than accessing a field in the Dalvik VM, so by doing this we make only one access to the field, and five accesses to the local variable, making the routine much more efficient. It is recommended that you use this optimization when possible.

Step 6

The above `createNote()` and `onListItemClick()` methods use an asynchronous Intent invocation. We need a handler for the callback, so here we fill in the body of the `onActivityResult()`.

`onActivityResult()` is the overridden method which will be called when an Activity returns with a result. (Remember, an Activity will only return a result if launched with `startActivityForResult`.) The parameters provided to the callback are:

- `requestCode` — the original request code specified in the Intent invocation (either `ACTIVITY_CREATE` or `ACTIVITY_EDIT` for us).
- `resultCode` — the result (or error code) of the call, this should be zero if everything was OK, but may have a non-zero code indicating that something failed. There are standard result codes available, and you can also create your own constants to indicate specific problems.
- `intent` — this is an Intent created by the Activity returning results. It can be used to return data in the Intent "extras."

The combination of `startActivityForResult()` and `onActivityResult()` can be thought of as an asynchronous RPC (remote procedure call) and forms the recommended way for an Activity to invoke another and share services.

Here's the code that belongs in your `onActivityResult()`:

```
super.onActivityResult(requestCode, resultCode, intent);
Bundle extras = intent.getExtras();

switch(requestCode) {
case ACTIVITY_CREATE:
```

```
    String title = extras.getString(NotesDbAdapter.KEY_TITLE);
    String body = extras.getString(NotesDbAdapter.KEY_BODY);
    mDbHelper.createNote(title, body);
    fillData();
    break;
case ACTIVITY_EDIT:
    Long mRowId = extras.getLong(NotesDbAdapter.KEY_ROWID);
    if (mRowId != null) {
        String editTitle = extras.getString(NotesDbAdapter.KEY_TITLE);
        String editBody = extras.getString(NotesDbAdapter.KEY_BODY);
        mDbHelper.updateNote(mRowId, editTitle, editBody);
    }
    fillData();
    break;
}
```

- We are handling both the `ACTIVITY_CREATE` and `ACTIVITY_EDIT` activity results in this method.
- In the case of a create, we pull the title and body from the extras (retrieved from the returned Intent) and use them to create a new note.
- In the case of an edit, we pull the mRowId as well, and use that to update the note in the database.
- `fillData()` at the end ensures everything is up to date .

Step 7

The Art of Layout

The provided note_edit.xml layout file is the most sophisticated one in the application we will be building, but that doesn't mean it is even close to the kind of sophistication you will be likely to want in real Android applications.

Creating a good UI is part art and part science, and the rest is work. Mastery of Declaring Layout is an essential part of creating a good looking Android application.

Take a look at the Hello Views for some example layouts and how to use them. The ApiDemos sample project is also a great resource from which to learn how to create different layouts.

Open the file `note_edit.xml` that has been provided and take a look at it. This is the UI code for the Note Editor.

This is the most sophisticated UI we have dealt with yet. The file is given to you to avoid problems that may sneak in when typing the code. (The XML is very strict about case sensitivity and structure, mistakes in these are the usual cause of problems with layout.)

There is a new parameter used here that we haven't seen before:
`android:layout_weight` (in this case set to use the value 1 in each case).

`layout_weight` is used in LinearLayouts to assign "importance" to Views within the layout. All Views have a default `layout_weight` of zero, meaning they take up only as

much room on the screen as they need to be displayed. Assigning a value higher than zero will split up the rest of the available space in the parent View, according to the value of each View's `layout_weight` and its ratio to the overall `layout_weight` specified in the current layout for this and other View elements.

To give an example: let's say we have a text label and two text edit elements in a horizontal row. The label has no `layout_weight` specified, so it takes up the minimum space required to render. If the `layout_weight` of each of the two text edit elements is set to 1, the remaining width in the parent layout will be split equally between them (because we claim they are equally important). If the first one has a `layout_weight` of 1 and the second has a `layout_weight` of 2, then one third of the remaining space will be given to the first, and two thirds to the second (because we claim the second one is more important).

This layout also demonstrates how to nest multiple layouts inside each other to achieve a more complex and pleasant layout. In this example, a horizontal linear layout is nested inside the vertical one to allow the title label and text field to be alongside each other, horizontally.

Índice de figuras.

Fig.1 Detalle código SQLite 7

Fig.2. Detalle código XML 8

Fig.3. Detalle código Java 9

Fig.4. Localización Workspace en Eclipse 11

Fig.5. Instalación del plugin de Android 12

Fig.6. Localiazción del SDK de Android..... 13

Fig.7. Emulador Android 14

Fig.8. Logo Android 15

Fig.9. Diagrama de la arquitectura del SO Android 20

Fig.10. Detalle Eclipse 23

Fig.11. Detalle de la creación de un nuevo proyecto 27

Fig.12. Detalle de la lista de notas 29

Fig.13. Detalle de las opciones del menú 30

Fig.14. Pantalla principal del reproductor..... 55

Fig.15. Apartado musical del reproductor 56

Fig.16. Reproducción de un vídeo 58

Fig.17. Cuadrícula de imágenes 59

Fig.18. Instrucciones Simón Dice 100

Fig.19. Detalle juego Simón Dice 101

Fig.20. Detalle Piano..... 123