ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN QUE PERMITA LA CAPTURA, ALMACENAMIENTO, REPRODUCCIÓN, ADMINISTRACIÓN Y ENVÍO DE ARCHIVOS DE VIDEO, AUDIO E IMÁGENES UTILIZANDO TECNOLOGÍA BLUETOOTH, PARA DISPOSITIVOS MÓVILES BASADOS EN LA ARQUITECTURA DEL SISTEMA OPERATIVO ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

DIEGO DAVID ARGÜELLO RIVERA

godiearguello@hotmail.com

DIRECTORA: MSc. SORAYA SINCHE

soraya.sinche@epn.edu.ec

Quito, Septiembre 2012

DECLARACIÓN

Yo, Diego David Argüello Rivera, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Diego David Argüello Rivera

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el señor Diego David Argüello Rivera, bajo mi supervisión.

MSc. Soraya Sinche
DIRECTORA DE PROYECTO

AGRADECIMIENTO

Mami, no me equivoco si digo que usted es la mejor mamá del mundo, gracias por todo su esfuerzo, su apoyo y por la confianza que deposito en mí. Gracias porque siempre, ha estado a mi lado. Le quiero mucho mamita.

Papá, este es un logro que quiero compartir contigo, gracias por ser mi papá y por creer en mí. Quiero que sepas que ocupas un lugar especial.

A mis hermanos que siempre me acompañaron en las buenas y en las malas, más que hermanos son mis mejores amigos Verito, José y Francisco.

A mi esposita preciosa Yelitza que siempre ha confiado en mí.

Por último y no menos importante a MSc. Soraya Sinche por su gran colaboración para sacar adelante este proyecto.

DEDICATORIA

Yelitza Anahí, aunque todavía no puedes leer,

un día vas aprender y por eso te dedico esta tesis,

gracias por alegrarme la vida,

me llenas de emoción cuando me dices papi.

Te amo mi amor.

Diego

CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

ÍNDICE DE TABLAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

RESUMEN

El presente proyecto de titulación describe la forma de programación en el sistema operativo Android que permite manejar su hardware para capturar, almacenar, administrar y enviar vía Bluetooth archivos multimedia.

El desarrollo del proyecto se divide en 4 capítulos descritos de la siguiente manera:

En el Capítulo 1 se presenta el estudio de la arquitectura del sistema operativo Android, se realiza un breve análisis de la situación actual de los sistemas operativos móviles su uso y aplicaciones. A continuación se analiza los componentes y la clasificación de los sistemas operativos luego se enfoca el estudio sólo a sistemas operativos móviles, profundizando en el sistema operativo Android.

El Capítulo 2 incluye el desarrollo de la aplicación multimedia y la comunicación Bluetooth, inicia con una descripción general del ambiente de desarrollo Eclipse y sus ventajas. Luego se describe el desarrollo de la aplicación multimedia utilizando diagramas de flujo, e incorporando secciones del código implementado en la aplicación.

El Capítulo 3 incluye la simulación e instalación de la aplicación en el dispositivo móvil. Se expone la compatibilidad de la aplicación desarrollada con otras versiones del sistema operativo Android. Se presenta la interfaz gráfica de la aplicación y resultados obtenidos en la fase de simulación e instalación, finalizando con el costo aproximado de la aplicación implementada.

En el Capítulo 4 se presentan las Conclusiones y Recomendaciones resultantes del presente proyecto de titulación.

PRESENTACIÓN

En la actualidad el desarrollo de la tecnología crece diariamente a pasos agigantados y el ser humano se adapta constantemente a estos cambios. La necesidad de comunicación de las personas se adapta constantemente a la tecnología, cambiando es el medio de comunicación, utilizando nuevas tecnologías que permiten en un instante comunicarse con alguien que se encuentra del otro lado del mundo.

Debido al uso masivo de los dispositivos móviles a nivel mundial, se han convertido en una herramienta indispensable como plataforma de comunicación y en el desarrollo de sus actividades cotidianas en áreas de entretenimiento o profesionales; por este motivo se desarrolló una aplicación que permita la captura, almacenamiento, administración, reproducción y envío de sus archivos multimedia.

Dada la viabilidad de explotar los recursos que brinda el software libre se utilizó Android OS como plataforma de desarrollo, por el gran potencial de sus herramientas de programación en dispositivos móviles y la facilidad de encontrar, acceder y obtener la información para su estudio.

Se presenta al usuario del sistema operativo Android una aplicación para capturar, almacenar, reproducir, administrar y enviar audio, video e imágenes; fácil y rápidamente, sus archivos multimedia pueden ser transferidos a cualquier otro dispositivo con interfaz Bluetooth, con una presentación amigable y sencilla.

Para satisfacer las necesidades del usuario como capturar momentos especiales, se ha creado la aplicación DroidMedia, un programa versátil, intuitivo y agradable, permitiendo al usuario grabar videos, sonidos y capturar imágenes para posteriormente ser compartidas con sus familiares y amigos.

CAPÍTULO 1

ESTUDIO DE LA ARQUITECTURA DEL SISTEMA OPERATIVO ANDROID

1.1 INTRODUCCIÓN

En la actualidad los dispositivos móviles se han convertido en una herramienta indispensable en las actividades cotidianas del ser humano, satisfaciendo la principal necesidad de comunicación como es el envío y recepción de voz, mensajes de texto, video llamadas, correos electrónicos etc.

El almacenamiento y administración de fotografías y música, es importante para las personas del mundo actual, que han sido atraídas por características del dispositivo, como la definición de su cámara integrada, gran capacidad de almacenamiento, resolución de su pantalla y nuevos procesadores más veloces y poderosos.

Estos dispositivos también satisfacen la necesidad de entretenimiento de todo tipo como ver películas, reproducir música y juegos de video de alta calidad.

Nace la necesidad de estar todo el tiempo en línea para compartir información inmediatamente, en redes sociales, chat y aplicaciones que permiten cargar información a la red.

Las actividades que el usuario puede realizar desde un *Smartphone*[1] son ilimitadas.

[1]Smartphone: Es un teléfono móvil de gran capacidad y productividad posibilita la instalación de nuevas aplicaciones desarrolladas por el fabricante o por terceros.

1.1.1 SITUACIÓN ACTUAL

Para aumentar la producción en diversas empresas de todo tipo, la utilización de dispositivos como teléfonos inteligentes se presenta como una gran opción para agilizar procesos que antes demandaban mayor tiempo.

"Todas las empresas en el mundo de una forma u otra van a tener la necesidad de utilizar estos dispositivos en el futuro. En el pasado hemos hecho la transición de ordenadores de escritorio a laptops y ahora a dispositivos móviles". [**1**]

Existen casos donde las personas en el campo utilizan estos dispositivos móviles para capturar la geografía del terreno y conocer el lugar donde se encuentran con sistemas de GPS embebidos en el dispositivo.

El problema se encuentra en la utilización de diversa cantidad de dispositivos móviles que usan los colaboradores y que existe en el mercado soluciones para empresas que requieren de una plataforma que permita enviar la información a cualquier sistema operativo (Android, iOS, etc.) de forma segura y con inmediatez.

Las empresas que diseñan y producen dispositivos móviles siempre han requerido de un software que administre el dispositivo de forma rápida y efectiva.

Ciertas empresas se han encaminado en un solo sistema operativo propietario, en cambio otras han optado por utilizar diferentes plataformas, lo permite la introducción al mercado de nuevos sistemas operativos, dejando atrás a los más dominantes que se están quedando rezagados en cuanto a diseño y presentación.

Los dispositivos como por ejemplo Nokia en todos sus dispositivos móviles utilizan el sistema operativo Symbian que durante años no ha cambiado su presentación, también se puede mencionar a los que han revolucionado el mercado, como Apple con iPhone OS, Microsoft con Windows Mobile y Google con Android.

1.2 SISTEMAS OPERATIVOS [2]

Un sistema operativo es el programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, permite la normal ejecución de las aplicaciones e interactua con el hardware.

Entre las funciones más relevantes de un sistema operativo se tiene:

- Software encargado de ejercer el control y coordinar el uso del hardware entre diferentes programas de aplicación y los diferentes usuarios.
- Administra los recursos de hardware del sistema que consiste en ofrecer una distribución ordenada y controlada de los procesadores, memorias y dispositivos de E/S entre los diversos programas que compiten por ellos.
- Proveer una máquina virtual, es decir, un ambiente en el cual el usuario pueda ejecutar programas de manera conveniente, protegiéndolo de los detalles y complejidades del hardware y administrar eficientemente los recursos del computador.

El objetivo principal de un Sistema Operativo es, entonces, lograr que el Sistema de computación se use de manera cómoda, y el objetivo secundario es que el hardware del computador se emplee de manera eficiente.

En un sistema de computación intervienen cuatro elementos, dispuestos de la siguiente manera como se observa en la Figura 1.1:

- Hardware.
- Sistema Operativo.
- Programas de aplicación.
- Usuarios.

Figura 1.1: Representación de los elementos de un sistema de computación [3]

El hardware se compone de la unidad central de proceso, memoria y dispositivos de entrada/salida (E/S), proporcionan los recursos básicos de un sistema de computación.

El sistema operativo es la capa intermediaria entre el hardware y las aplicaciones consiste en gestionar los recursos y protección de acceso al hardware utilizado controladores (*Drivers*) para cada elemento de hardware, hecho que alivia a los programadores de aplicaciones que no deben tratar con estos detalles.

Los programas de aplicación resuelven problemas específicos del usuario facilitando su trabajo, como ejemplo un procesador de texto, navegador web o una hoja electrónica etc.

Los usuarios son los que explotan las capacidades del hardware y software para su beneficio.

La mayoría de aparatos electrónicos utilizan microprocesadores, para funcionar llevan incorporado un sistema operativo: teléfonos móviles, reproductores de DVD, computadoras, radios, enrutadores, etc.

1.2.1 COMPONENTES DE UN SISTEMA OPERATIVO

1.2.1.1 Gestión de procesos

Un proceso es simplemente, un programa en ejecución que necesita recursos para realizar su tarea: tiempo de CPU, memoria, archivos y dispositivos de E/S. El OS es el responsable de:

- Crear y destruir los procesos.
- Parar y reanudar los procesos.
- Ofrecer mecanismos para que se comuniquen y sincronicen.

La gestión de procesos podría ser similar al trabajo de oficina. Se puede tener una lista de tareas a realizar y a estas fijarles prioridades alta, media y baja.

Se debe comenzar haciendo las tareas de prioridad alta primero y cuando se terminen seguir con las de prioridad media y después las de baja. Esto puede traer un problema, que las tareas de baja prioridad puede nunca llegar a

ejecutarse y permanezcan en la lista para siempre. Para solucionar esto, se puede asignar alta prioridad a las tareas más antiguas.

1.2.1.2 Gestión de la memoria principal

La memoria es una tabla de palabras o bytes que cada posición es referenciada mediante una dirección única escrita en hexadecimal. Este almacén de datos de rápido acceso, es compartido por la CPU y los dispositivos de E/S; es volátil y pierde su contenido en los fallos del sistema o interrupción de energía. El sistema operativo es el responsable de:

- Conocer qué partes de la memoria están siendo utilizadas y por quién.
- Decidir qué procesos se cargarán en memoria cuando haya espacio disponible.
- Asignar y reclamar espacio de memoria cuando sea necesario.

1.2.1.3 Gestión del Almacenamiento

Un sistema de almacenamiento es necesario, ya que la memoria principal es volátil y además muy pequeña para almacenar todos los programas y datos, es útil mantener los programas que no se encuentra ejecutando en un hardware de almacenamiento no volátil para una posterior ejecución. El sistema operativo se encarga de:

- Establecer tamaño de asignación para cada partición.
- Formato de las particiones.
- Gestionar el espacio libre.
- Direccionar la información en el dispositivo de almacenamiento.
- Verificar que los datos se guarden en orden.

1.2.1.4 Sistema de Entrada y Salida (E/S)

Consiste en un sistema de almacenamiento temporal, un interfaz[2] de manejadores de dispositivos y otro para dispositivos concretos. El sistema operativo debe gestionar el almacenamiento temporal de un *buffer*[3] de E/S.

---

[2] Interfaz: Establece una relación de comunicación entre distintos niveles por ejemplo de distintas capas de un sistema operativo.

[3] *Buffer:* Es un espacio de memoria, en el que se almacenan datos para evitar que el programa o recurso que los requiere, ya sea hardware o software, se quede sin datos durante una transferencia.

Tambien debe atender las interrupciones generadas por los dispositivos de entrada y salida de datos.

1.2.1.5 Sistema de Archivos

Los archivos son colecciones de información relacionada, definidas por sus creadores. Éstos almacenan programas (en código fuente y objeto) y datos, tales como: imágenes, textos, información de bases de datos, etc. El sistema operativo es responsable de:

- Construir y eliminar archivos y directorios.
- Ofrecer funciones para manipular archivos y directorios.
- Establecer la correspondencia entre archivos y unidades de almacenamiento.
- Realizar copias de seguridad de archivos.

Existen diferentes Sistemas de Archivos, es decir, existen diferentes formas de organizar la información en dispositivos de almacenamiento como: memorias portátiles (*flash memory*) y discos duros (HDD). Por ejemplo, existen los siguientes sistemas de archivos FAT, FAT32, EXT3, NTFS, XFS, etc.

Desde el punto de vista del usuario estas diferencias pueden parecer insignificantes a primera vista, sin embargo, existen diferencias muy importantes. Por ejemplo, los sistemas de ficheros FAT32 y NTFS, que se utilizan fundamentalmente en sistemas operativos de Microsoft, tienen una gran diferencia para un usuario que utilice una base de datos con bastante información, ya que el tamaño máximo de un fichero con un sistema de archivos FAT32 está limitado a 4 gigabytes, sin embargo, en un sistema NTFS el tamaño es únicamente limitado por el tamaño de la partición.

1.2.1.6 Sistemas de Protección

Mecanismo que controla el acceso de los programas o los usuarios a los recursos del sistema. El sistema operativo se encarga de:

- Distinguir entre uso autorizado y no autorizado.
- Especificar los controles de seguridad a realizar.
- Forzar el uso de estos mecanismos de protección.

1.2.1.7 Sistema de Comunicaciones

Para mantener las comunicaciones con otros sistemas es necesario poder controlar el envío y recepción de información a través de las interfaces de red. También hay que crear y mantener puntos de comunicación que sirvan a las aplicaciones para enviar y recibir información, crear y mantener conexiones virtuales entre aplicaciones que están ejecutándose localmente y otras que lo hacen remotamente.

1.2.1.8 Programas del Sistema (Aplicaciones)

Son aplicaciones de utilidad que se suministran con el sistema operativo pero no forman parte de él. Ofrecen un entorno útil para el desarrollo y ejecución de programas, algunas de las tareas que realizan son las siguientes:

- Manipulación y modificación de archivos.
- Información del estado del sistema.
- Soporte a lenguajes de programación.
- Comunicaciones.

1.2.1.9 Gestor de Recursos

Como gestor de recursos administra:

- La Unidad Central de Proceso (CPU), donde está alojado el microprocesador.
- Los dispositivos de entrada y salida (E/S).
- La memoria principal para ejecutar procesos (RAM).
- Los discos de almacenamiento (HDD)
- Los procesos de los programas en ejecución.

1.2.2 CLASIFICACIÓN DE LOS SISTEMAS OPERATIVOS

Se los puede clasificar de la siguiente manera:

- Administración de Tareas.
- Administración de Usuarios.
- Manejo de Recursos.
- Tipo de Dispositivos.

1.2.2.1 Administración de Tareas

1.2.2.1.1 Monotarea

Solamente puede ejecutar un proceso (aparte de los procesos del propio sistema operativo) en un momento dado. Una vez que empieza a ejecutar un proceso, continuará haciéndolo hasta su finalización o interrupción. Es una característica de los sistemas operativos más antiguos como MS-DOS pues en la actualidad la mayoría de los sistemas de propósito general son multitarea.

1.2.2.1.2 Multitarea

Es capaz de ejecutar varios procesos al mismo tiempo. Este tipo de sistema operativo normalmente asigna los recursos disponibles (CPU, memoria, periféricos) de forma alternada a los procesos que los solicitan, de manera que el usuario percibe que todos funcionan a la vez, de forma concurrente.

1.2.2.2 Administración de usuarios

1.2.2.2.1 Monousuario

Si sólo permite ejecutar los programas de un usuario al mismo tiempo en el sistema operativo.

1.2.2.2.2 Multiusuario

Si permite que varios usuarios ejecuten simultáneamente sus programas, accediendo a la vez a los recursos de la computadora. Normalmente estos sistemas operativos utilizan métodos de protección de datos, de manera que un programa no pueda usar o cambiar los datos de otro usuario.

1.2.2.3 Manejo de recursos

1.2.2.3.1 Centralizado

Si permite usar los recursos de una sola computadora es decir su CPU, memoria, discos y periféricos

Respecto al hardware se puede decir que se suele tratar de un computador costoso y de gran potencia, con terminales alfanuméricos directamente conectados. Se trata de una computadora de tipo *desktop*, en las cuales es

común encontrar un monitor grande con un teclado y un mouse; además de un CPU para albergar la unidad de procesamiento y los demás componentes.

Los problemas de este modelo se dan cuando la carga de procesamiento aumenta, por lo que se tiene que cambiar el hardware del *Mainframe*[4], lo cual es más costoso que añadir más computadores personales clientes o servidores que aumenten las capacidades.

El otro problema que surgió son las modernas interfaces gráficas de usuario, las cuales conllevan a un gran aumento de tráfico en los medios de comunicación y por consiguiente podían colapsar.

1.2.2.3.2 Distribuido

Un sistema distribuido "Es una colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora". [**4**]

Un sistema distribuido se caracteriza por comportarse frente al usuario como una sola máquina; el usuario desconoce sobre qué procesador se está ejecutando sus procesos y dónde residen sus ficheros.

1.2.2.4 Tipo de Dispositivos

Se clasifican en dos tipos de acuerdo a los dispositivos en los cuales pueden ser instalados:

- Computadores.
- Dispositivos Móviles.

Los sistemas operativos más populares y utilizados para computadores son:

- Windows.
- Mac OS.
- Linux.
- Unix.

Los sistemas operativos móviles más utilizados son:

[4] *Mainframe*: Es una computadora grande, potente y costosa

- Symbian.
- Android.
- iPhone OS.
- Windows Phone.
- BlackBerry OS.
- WebOS evolución del palm.

1.3 SISTEMAS OPERATIVOS MÓVILES [5]

Los sistemas operativos móviles son más simples que los tradicionales pero las funciones que se requiere son similares; están orientados a la conectividad inalámbrica y al bajo consumo de energía.

Los formatos multimedia de imágenes, audio y video que manejan utilizan mayor compresión para ahorrar espacio en memoria, formatos desarrollados específicamente para dispositivos móviles ya que son de características limitadas en cuanto a capacidad de memoria y procesamiento.

Se busca disminuir la tasa de bits[5] para el envío de la información ya que puede resultar costoso trasmitirla utilizando una conexión inalámbrica móvil.

Para adaptarse al tamaño reducido de sus pantallas se utilizan imágenes, audio y video de menor resolución y redefinen el esquema de la interfaz del usuario ya que al administrar la información se utiliza un *touch screen* teclado QWERTY[6] o simplemente sus teclas numéricas.

1.3.1 CAPAS PRINCIPALES DE UN SISTEMA OPERATIVO MÓVIL

Un sistema operativo móvil cumple con todas las actividades de los sistemas tradicionales y se divide en cuatro capas principales:

- Kernel[7].
- Middleware[8].
- Entorno de Ejecución de Aplicaciones.

[5] Tasa de bits: Velocidad de trasferencia de información.
[6] QWERTY: Distribución mas común del teclado alfanumérico.
[7] Kernel: Núcleo del sistema operativo.
[8] Middleware: Capa de abstracción entre el núcleo del sistema operativo y las aplicaciones.

- Interfaz de Usuario.

1.3.1.1 Kernel

Proporciona el acceso a los distintos elementos del hardware del dispositivo. Ofrece distintos servicios a las capas superiores como son los controladores o *drivers* para el *hardware*, la gestión de procesos, el sistema de los archivos y el manejo de la memoria.

1.3.1.2 Middleware

Es el conjunto de módulos que hacen posible la propia existencia de aplicaciones para móviles. Es totalmente transparente para el usuario y ofrece servicios claves como el motor de mensajería y comunicaciones, códec multimedia, intérpretes de páginas web, gestión del dispositivo y seguridad.

1.3.1.3 Entorno de Ejecución de Aplicaciones

Consiste en un gestor de aplicaciones y un conjunto de interfaces, programables por parte de los desarrolladores para facilitar la creación de software.

1.3.1.4 Interfaz de Usuario

Facilita la interacción con el usuario y el diseño de la presentación visual de la aplicación. Los servicios que incluye son el de componentes gráficos (botones, pantallas, listas, visualizador de imágenes, etc.) y el del marco de interacción.

Aparte de estas capas también existe una familia de aplicaciones nativas del teléfono que suelen incluir los menús, agenda telefónica, el marcador de números de teléfono advertencias generales batería baja, llamada entrante etc.

1.3.2 SISTEMAS OPERATIVOS MÓVILES MÁS UTILIZADOS

Se hace un breve análisis del mercado de los sistemas operativos móviles mas utilizados actualmente.

1.3.2.1 Mercado de los Sistemas Operativos móviles [6]

Se muestra una comparación en ventas mundiales por sistema operativo móvil y por marca.

Estadísticas globales para Android, líder en ventas en primer trimestre del 2012, obtiene un 56,1% de la compartición del mercado como se muestra en la Figura 1.2.

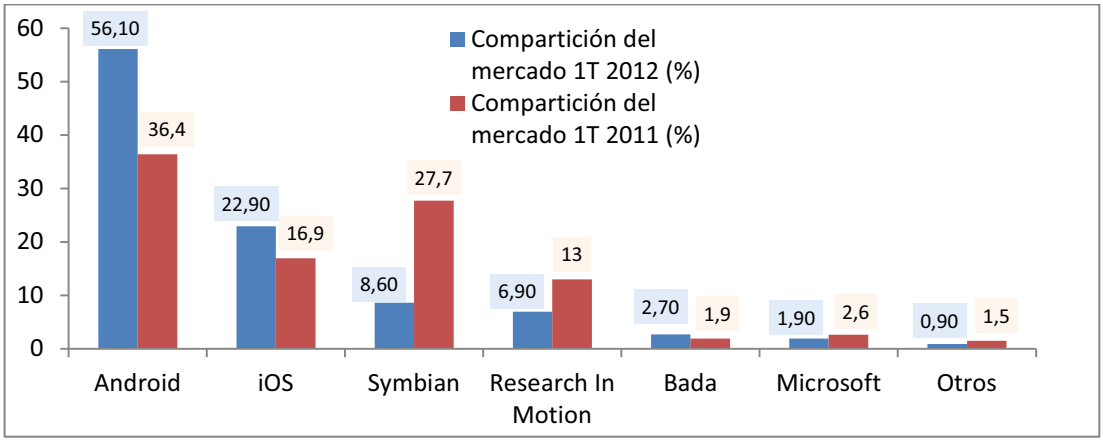


Figura 1.2: Ventas globales a usuarios finales en el año 2012 [6]

Android gana terreno en el mercado de sistemas operativos móviles. En Tabla 1.1 se muestra cómo se reparten las ventas de dispositivos por marca, Nokia ha perdido nivel de mercado con respecto al primer trimestre del 2011 en dispositivos móviles, del 25,1% al 19,8%.

Fabricante	1 Trimestre 2012 miles de unidades	Compartición del mercado 1T 2012 (%)	1 Trimestre 2011 miles de unidades	Compartición del mercado 1T 2011 (%)
Samsung	86567,6	20,7	68782	16,1
Nokia	83162,5	19,8	107556,1	25,1
Apple	33120,5	7,9	16883,2	3,9
ZTE	17439,3	4,2	10788,7	2,5
LG	14720,4	3,5	23997,2	5,6
Huawei Device	10796,1	2,6	7002,9	1,6
Research in Motion	9939,3	2,4	13004	3
Motorola	8368,2	2	8789,7	2,1
Sony Mobile Communications	7898,4	1,9	7919,4	1,9
HTC	7703,4	1,8	9313,5	2,2
Otros	139392,6	33,3	153809	35,9
Total	**419108,3**	**100**	**427845,7**	**100**

Tabla 1.1: Ventas en miles de unidades por Marca [6]

En la Tabla 1.2 se observa las ventas en millones de unidades por sistema operativo utilizado, en el primer trimestre del 2011 la compartición del mercado

para Android era de 36,4% y en el 2012 logra incrementar significativamente ese valor; se puede destacar que está teniendo una gran aceptación por parte de nuevos usuarios.

Sistema Operativo	1 Trimestre 2012 miles de unidades	Compartición del mercado 1T 2012 (%)	1 Trimestre 2011 miles de unidades	Compartición del mercado 1T 2011 (%)
Android	81067,4	56,1	36350,1	36,4
iOS	33120,5	22,9	16883,2	16,9
Symbian	12466,9	8,6	27598,5	27,7
Research In Motion	9939,3	6,9	13004	13
Bada	3842,2	2,7	1862,2	1,9
Microsoft	2712,5	1,9	2582,1	2,6
Otros	1242,9	0,9	1495	1,5
Total	**144391,7**	**100**	**99775**	**100**

Tabla 1.2: Ventas en miles unidades por sistema operativo [6]

A continuación se describen los siguientes sistemas operativos:

- Palm OS.
- Symbian OS.
- Back Berry OS.
- Windows Mobile.
- iPhone OS.
- Android OS.

1.3.2.2 Palm OS (WEB OS)

1.3.2.2.1 Historia

Palm OS es un sistema operativo propietario destinado a dispositivos móviles, más específicamente a PDA (*Personal Digital Assistant*). En la Figura 1.3 se observa la pantalla principal

Figura 1.3: Palm OS [7]

Palm OS comenzó su desarrollo en 1996 y Palm Inc. comenzó a licenciarlo en diciembre de 1997 con sus novedosos aparatos *PalmPilot*.

A partir de ese momento el soporte y el desarrollo de Palm OS se disparó, llegando en enero del 2001 a tener 100.000 personas registradas en su red de desarrolladores trabajando en proyectos para Palm OS.

Palm OS fue uno de los pioneros en el mercado de los dispositivos móviles y por varios años se mantuvo como líder de los sistemas operativos, sobre todas las cosas por ser muy usable y simple. [**8**]

1.3.2.2.2 Características principales del sistema operativo

Se puede mencionar algunas características principales de este sistema operativo:

- Arquitectura basada en procesadores ARM (Advanced RISC Machine).
- Soporte para tamaño de pantalla hasta 320x480.
- Soporte multilenguaje, inglés, japonés y chino simplificado.
- Menos de 300 KBytes en RAM para la ejecución del sistema operativo.
- RAM máxima soportada de 128 MBytes.

1.3.2.3 Symbian OS [9]

1.3.2.3.1 Historia

Es el resultado de una alianza de varias empresas multinacionales tales como Nokia, Sony Ericsson, Samsung, Siemens, Motorola y otras. El objetivo de Symbian fue crear un sistema operativo para terminales móviles que competiría

con Palm o Windows Mobile de Microsoft. Se puede observar la pantalla principal de un dispositivo con Symbian OS en la Figura 1.4.

Figura 1.4: Symbian OS [9]

A continuación se muestra la evolución del sistema operativo a través de sus diferentes versiones:

- 1997, EPOC[9] OS (32 bits).
- 1998, El nombre de Symbian aparece por primera vez.
- 2000, Symbian 6.0.
- 2001, Symbian 6.1.
- 2003, Symbian 7.0.
- 2004, Symbian 8.0.
- 2006, Symbian 9.0.
- 2011, Symbian 9.5.
- 2012, Symbian 10.1 ultima versión.

1.3.2.3.2 Características principales del sistema operativo

Symbian está basado en un micro Kernel. Una mínima porción del sistema tiene privilegios de Kernel, el resto se ejecuta con privilegios de usuario, en modo de servidores. Una de las tareas del Kernel es manejar las interrupciones y prioridades.

El sistema posee componentes que permiten el diseño de aplicaciones multiplataforma, esto es diferentes tamaños de pantalla, color, resolución, teclados, etc. La mayoría de estos componentes han sido diseñados en C++.

[9] EPOC: Nombre que se le dio al primer sistema operativo Symbian.

Soportan paginación bajo demanda, lo que significa que permite un mejor aprovechamiento de la memoria RAM de los dispositivos, ya que solo se carga en memoria la página a ejecutarse.

1.3.2.4 Black Berry OS [8]

1.3.2.4.1 Historia

El BlackBerry OS es un sistema operativo móvil desarrollado por *Research In Motion* (RIM) para sus dispositivos BlackBerry. Un sistema multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la *trackwheel, trackball, touchpad* y pantallas táctiles. Se observa la pantalla principal en la Figura 1.5.

Figura 1.5: Black Berry OS [10]

Su desarrollo se remonta la aparición de los primeros *handheld*[10] en 1999. Estos dispositivos permiten el acceso a correo electrónico, navegación web y sincronización con programas como Microsoft Exchange o Lotus Notes, además de incluir las funciones de un teléfono móvil.

1.3.2.4.2 Características principales del sistema operativo

BlackBerry está claramente orientado a ser utilizado como gestor de correo electrónico y agenda. Desde la versión actual, 7.0 se puede sincronizar el dispositivo con el correo electrónico, el calendario, tareas, notas y contactos de

[10] Handheld: Es un anglicismo que significa en castellano "palmar" y describe a una computadora portátil que se puede llevar en una mano a cualquier parte mientras se utiliza.

Microsoft Exchange Server, además es compatible también con *Lotus Notes* y *Novell GroupWise.*

BlackBerry Enterprise Server (BES) proporciona el acceso y organización del email a grandes compañías identificando a cada usuario con un único BlackBerry PIN[11]. Los usuarios más pequeños cuentan con el software *BlackBerry Internet Servicie* (BIS), programa más sencillo que proporciona acceso a Internet y a correo POP3[12] / IMAP[13] / *Outlook Web Access*.

Al igual que en el sistema operativo Symbian, desarrolladores independientes pueden también crear programas para BlackBerry, pero en el caso de querer tener acceso a ciertas funcionalidades restringidas, necesitan ser firmados digitalmente por una cuenta de desarrollador de RIM.

1.3.2.5 Windows Mobile [11]

1.3.2.5.1 Historia

Windows Mobile es un sistema operativo de la familia Windows CE, desarrollado por Microsoft. A pesar de llevar el nombre Windows, no es un sistema derivado ni es una versión recortada del mismo, sino que es un nuevo sistema diseñado específicamente para dispositivos móviles. La pantalla principal se muestra en la Figura 1.6.

Figura 1.6: Windows Mobile [11]

[11] BB PIN: Es un numero de identificación internacional del equipo, permite su registro en la base de datos de los servidores BlackBerry para usar el servicio de mensajes, correo, aplicaciones de chat, etc.
[12] POP3: Protocolo para obtener los mensajes de correo electrónico almacenados en un servidor remoto.
[13] IMAP: *Internet Message Access Protocol*, o su acrónimo IMAP, es un protocolo de red de acceso a mensajes electrónicos almacenados en un servidor a través de internet; versión mejora de POP3.

Los primeros dispositivos que se comenzaron a fabricar con este sistema operativo datan del año 2000. Para ese entonces, fue lanzado como Pocket PC 2000 y estaba basado en Windows CE 3.0.

1.3.2.5.2 Características principales del sistema operativo

Este sistema, está estrechamente vinculado a otros productos de la misma marca (servicios Live, Office Mobile, Internet Explorer Mobile, etc.) y cuenta con un interfaz gráfica de buena calidad, y muy similar a la de los sistemas operativos Windows. Se destaca las siguientes características:

- Ayuda a disminuir la curva de aprendizaje de los usuarios pues proveen un entorno de trabajo muy similar al que se tiene en el hogar o en la oficina.
- El Kernel unificado de Windows CE puede manejar más de 32000 procesos simultáneos, cada uno con 2GB de memoria virtual compartida.
- El sistema de archivos soporta un direccionamiento de hasta 4GBytes y encriptación de la información.
- Soporta diferentes arquitecturas de procesadores x86[14], ARM[15], SH4[16].

1.3.2.6 IPhone OS [12]

1.3.2.6.1 Historia

La historia del IPhone OS comienza conjuntamente con el nacimiento del conocido IPhone, en el 2007. Aunque, esta aseveración es discutible, ya que este sistema operativo que corre en el IPhone es en realidad una versión adaptada del OS X. Por lo cual, en este sentido, este sistema ya tiene años en el mercado y ha sido puesto a prueba. Se observa la pantalla principal en la Figura 1.7.

[14]Es la denominación genérica dada a ciertos microprocesadores de la familia Intel, sus compatibles y la arquitectura básica a la que estos procesadores pertenecen, por la terminación de sus nombres numéricos: 8086, 80286, 80386, 80486.
[15]Una familia de microprocesadores "*Advanced RISC Machines*".
[16] Microprocesadores creados por Hitachi para altas prestaciones multimedia.

Figura 1.7: iPhone OS 5.0.1 [12**]**

Es una adaptación de OS X para MAC, removiendo todos los componentes que no son críticos para un dispositivo móvil, y se le adicionan funcionalidades que si están relacionadas con el mundo de la telefonía móvil.

1.3.2.6.2 Características principales del sistema operativo

Sobre la versión modificada del Kernel de MAC OS X que corre el iPhone, se encuentran las capas de servicios que componen el teléfono móvil.

Existe una gran preocupación en el desarrollo del interfaz de usuario para que sea lo mas atractiva posible.

Sin duda el iPhone es el SO para dispositivos móviles que brinda una mejor experiencia de usuario, con un modo de manejo revolucionario basado en su *Touch Screen*.

1.3.2.7 Android OS [13]

1.3.2.7.1 Historia

Android OS es el más reciente de los sistemas operativos para móviles del mercado. Android está siendo desarrollado por "*The Open Handset*" Alliance un grupo de más de 84 empresas de tecnología y compañías de servicios móviles es el primer sistema operativo móvil abierto y completo, el principal participante es Google.

1.3.2.7.2 Características principales del sistema operativo

Se trata de un sistema operativo abierto, multitarea permite a los desarrolladores acceder a todas las funcionalidades del dispositivo.

Todas las aplicaciones son remplazables, el sistema operativo no diferencia entre las aplicaciones básicas del teléfono y las aplicaciones de terceros. Cualquier aplicación puede ser remplazada libremente, incluso las que trae por defecto el sistema operativo.

- Navegador Web integrado basado en el motor *WebKit*.
- Soporte para gráfico 2D y 3D basado en la especificación *OpenGL*[17].
- Base de datos *SQLite*.
- Soporte multimedia para audio, video e imágenes en varios formatos.
- Conectividad Bluetooth, EDGE, 3G y Wi-Fi.
- Aplicaciones personalizadas.

Se basa en el Kernel de Linux versión 2.6 para las principales funciones como seguridad, manejo de memoria, manejo de procesos, *networking* y modelo de controladores (*Drivers*).

Android hace público un SDK (*Software Development Kit*) para que los desarrolladores que lo deseen puedan programar aplicaciones que corran en este sistema operativo.

El lenguaje de programación utilizado es Java. Las aplicaciones corren sobre una máquina virtual diseñada para ser usada de forma embebida, denominada Dalvik, la cual se ejecuta sobre un Kernel de Linux.

Cada aplicación en Android corre en su propio proceso con su propia instancia de la máquina virtual Dalvik. La máquina virtual está optimizada para bajo consumo de recursos del sistema.

[17] *OpenGL*: Librería gráfica abierta.

1.3.3 COMPARACIONES ENTRE SISTEMAS OPERATIVOS MÓVILES [14]

Se realizó una comparación de los sistemas operativos móviles en los siguientes aspectos:

- Características Básicas
- Interfaz de usuario
- Funcionamiento
- Soporte para Desarrolladores

1.3.3.1 Características Básicas

Lo esencial e importante en un sistema operativo es el núcleo (Kernel). Android usa un Kernel Linux, con una mezcla especial de Java.

iPhone se basa en OS X, que a su vez es una variante de Unix, uno de los sistemas operativos más estables en el mundo de la informática.

Symbian y Windows Mobile son sistemas operativos muy maduros y estables, aunque la edad no siempre es una ventaja. Por último, RIM usa un Kernel propio. En la Tabla 1.3 se muestra una comparación entre sistemas operativos móviles clasificada por características básicas.

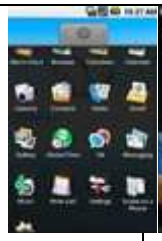
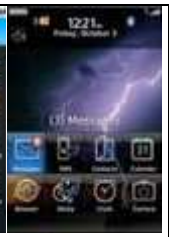
	 **Android OS**	 **BlackBerry OS**	 **iPhone OS**	 **Symbian OS**	 **Palm WebOS**	 **Windows Mobile**
Tipo de núcleo	Linux	Propietario	OS X	Symbian	Linux	Windows CE
Adaptabilidad	Excelente	Buena	Mala	Excelente	Excelente	Excelente
Edad de la plataforma	Joven	Madura	Adolescente	Madura	Joven	Madura
Tecnologías inalámbricas	GSM, WiFi	GSM, CDMA, WiFi	GSM, WiFi	GSM, WiFi	GSM, CDMA, WiFi	GSM, CDMA, WiFi

Tabla 1.3: Comparación características básicas de los sistemas operativos móviles [14]

1.3.3.2 Interfaz de usuario

Una de las opciones más atractivas y prácticas en una interfaz gráfica es la posibilidad de usar gestos en pantalla. En ese campo, el iPhone podría ser

elegido como el ganador, ya que con simples desplazamientos de los dedos, puedes realizar importantes cambios entre aplicaciones.

Android también permite usar gestos[18], pero los movimientos son difíciles de aprender porque son diferentes en cada aplicación. Windows Mobile y BlackBerry OS son fáciles de usar: el primero gracias a su *stylus*, y el otro con las conocidas ruedas o bolitas de los móviles RIM.

Los Symbian se han quedado atrás, y tienen todavía mucho que aprender de la competencia. Peleando por el primer lugar se presenta el Palm OS, que con su nuevo WebOS promete revolucionar la manera en se usa la pantalla del teléfono.

Una de las grandes diferencias en la experiencia del interfaz reside en el tipo de pantalla: capacitiva o resistiva. Mientras que las segundas han sobrevivido el paso del tiempo, las capacitivas son la nueva alternativa, en especial porque no requieren de objetos externos (como un *stylus*[19]) para funcionar. En la Tabla 1.4 se muestra la comparación por interfaces de usuario.

	Android OS	**BlackBerry OS**	**iPhone OS**	**Symbian OS**	**Palm WebOS**	**Windows Mobile**
Gestos	Sí	Sí	Sí	Limitado	Sí	Limitado
Tecnología de la pantalla	Capacitiva	Capacitiva	Capacitiva	Resistiva / Capacitiva	Capacitiva	Resistiva
Multitáctil	Sí	Sí	Sí	Sí	Sí	No
Personalización de pantalla	Sí	Sí	No	Sí	No	Sí
Obtención de información	Teclado virtual, teclado físico	Teclado virtual, teclado físico	Teclado virtual	Teclado virtual, T9, y triple click, teclado físico	Teclado virtual, Teclado físico	Teclado virtual, teclado físico

Tabla 1.4: Comparación de interfaces de usuario [14**]**

[18] Gestos: Movimientos especiales en las pantalla táctiles que permiten realizar funciones adicionales como por ejemplo hacer un acercamiento de una imagen.
[19] *Stylus*: Un pequeño puntero que mejora la precisión en las pantallas táctiles.

1.3.3.3 Funcionamiento

La parte más importante de un sistema operativo es la capacidad de ejecutar múltiples tareas con un solo procesador ahorrando energía.

iPhone OS de Apple ha decidido no permitir la ejecución de más de una aplicación al mismo tiempo. Palm promociona la multitarea como una de las mejores características de su sistema operativo WebOS.

Windows Mobile, implementa múltiple tarea, los sistemas operativos Android, Symbian y BlackBerry también permiten correr múltiples aplicaciones, con mayor o menor eficiencia.

El sistema de notificaciones de Android y WebOS es considerado por los usuarios como eficientes y funcionales.

Windows Mobile y BlackBerry OS usan un sistema de notificación que mezcla sonidos con *pop-ups*[20], que pueden llegar a ser incómodos.

Finalmente, está el iPhone y su implementación de *pop-ups*, se puede observar la Figura 1.8. En la Tabla 1.5 se muestra la comparación de funcionamiento de los diferentes sistemas operativos.

Figura 1.8: Pop-ups en iPhone [15**]**

[20] Pop-up: Ventana emergente que se ejecuta automáticamente sin que el usuario lo haya solicitado

	Android OS	BlackBerry OS	iPhone OS	Symbian	Palm WebOS	Windows Mobile
Notificación	Bandeja	Pop-up, fondo	Pop-up	Pop-up	Bandeja	Bandeja, pop-up
Administración de contactos	Google	BES, BIS	Exchange, ActiveSync, Mac OS Address Book	Exchange, Domino, iSync	Synergy	Exchange, Domino, ActiveSync
Multitarea	Sí	Sí	No	Sí	Sí	Sí
Copiar / pegar	Sí	Sí	Sí	Sí	Sí	Sí
Tiendas accesibles	Amazon	iTunes sin DRM	iTunes	Ovi	Amazon	Windows Media Player
Actualización de firmware	OTA[21]	Tethered[22], OTA	Tethered	Tethered, OTA	No Disponible	Tethered, OTA
Motor del navegador	WebKit[23]	Propietario	WebKit	WebKit	WebKit	Internet Explorer
Protocolo para Bluetooth estéreo	Sí	Sí	Sí	Sí	Sí	Sí

Tabla 1.5: Comparación de Funcionamiento [14**]**

1.3.3.4 Soporte para Desarrolladores

Apple tiene la tienda de iTunes es un éxito, y funciona muy bien tanto como para pequeños desarrolladores como para grandes compañías y abrió la puerta para nuevos negocios.

Windows Mobile y Symbian no se quedan muy atrás en lo que a cantidad de aplicaciones se refiere, pero lamentablemente no existe un lugar único para encontrarlas de manera fácil y cómoda.

Una parte fundamental en todos los sistemas operativos móviles es la tienda de aplicaciones. Mientras que Android ya tienen su *Market* disponible.

[21] OTA: (K) actualización con red móvil de datos directamente desde el dispositivo móvil.
[22] *Tethered*: Se denomina anclaje a red o *tethering* al proceso por el cual un dispositivo móvil con conexión a Internet actúa como puente para ofrecer acceso inalámbrico a la red a otros dispositivos.
[23] *WebKit*: Es una plataforma para aplicaciones que funciona como base para el navegador web Safari, Google Chrome. Está basado originalmente en el motor KHTML del navegador web del proyecto KDE.

Los SDK (*Software Development Kit*) deberán ser juzgados por los programadores, y la empresa ganadora será la que ofrezca una plataforma robusta y sencilla de usar. En la Tabla 1.6 se muestra la comparación de soporte para desarrolladores.

	Android	BlackBerry OS	iPhone OS	Symbian	Palm WebOS	Windows Mobile
Disponibilidad de SDK / Soporte	Sí	Sí	Sí	Sí	Sí	Sí
Tienda de aplicaciones	Sí	Sí	Sí	Sí	Sí	Sí
Disponibilidad de aplicaciones	Alta	Mediana	Alta	Mediana	Baja	Alta
Aplicaciones nativas	No	No	Sí	Sí	No	Sí
Administración local de aplicaciones	Excelente	Buena	Excelente	Buena	Excelente	Buena

Tabla 1.6: Comparación de soporte para Desarrolladores [14**]**

1.4 SISTEMA OPERATIVO ANDROID [13]

Un sistema operativo de código completamente abierto, desarrollado principalmente por Google Inc. para competir con las empresas más grandes y poderosas del mercado, utilizado un sistema operativo atractivo, fácil e intuitivo de utilizar.

Se debe conocer la funcionalidad de cada versión del sistema operativo Android, para evaluar, discriminar y determinar un requerimiento mínimo del sistema operativo al instalar una aplicación determinada.

El sistema operativo Android se divide en cinco capas, las cuales determinan su capacidad y funcionalidad en un dispositivo móvil, las aplicaciones se encuentran basadas en un interfaz de programación de aplicaciones API (*Application Programming Interface*).

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un *framework*[24] particular de Java.

Las aplicaciones para Android están orientadas a objetos.

El entorno de ejecución de Android se encuentra el núcleo de las bibliotecas de Java en una máquina virtual Dalvik, la compilación se realiza en tiempo de ejecución.

Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica, un *framework* de código abierto, una base de datos relacional *SQLite*, una API gráfica *OpenGL* 2.0 3D, un motor de navegación WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C.

1.4.1 INTERFAZ DE PROGRAMACIÓN DE APLICACIONES (API) [16]

Es el conjunto de funciones, procedimientos o métodos, que en la programación orientada a objetos, ofrece la posibilidad de abstracción de muchas líneas de comandos en una sola biblioteca

El API es utilizado por otro software de programación, como una capa de abstracción entre el lenguaje de programación y el leguaje propio del sistema operativo, generalmente denominadas “librerías”.

Nivel API es un valor entero que identifica de forma exclusiva la revisión del *Framework* de la API que ofrece una versión de la plataforma Android. Nivel de API permite al sistema determinar correctamente si una aplicación es compatible con el sistema, antes de instalar la aplicación.

La plataforma Android ofrece un marco API que las aplicaciones pueden utilizar para interactuar con la base del sistema Android. El API se compone de:

- Un conjunto de paquetes y clases.
- Un conjunto de elementos y atributos para la declaración de un archivo de configuraciones.

[24] Framework: es una estructura conceptual y tecnológica, normalmente constituido por módulos de software, en base al cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente incluye soporte a leguajes de programación, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

- Un conjunto de elementos y atributos para acceder a los recursos.
- Un conjunto de intenciones que permiten ejecutar procesos.
- Un conjunto de permisos que las aplicaciones pueden solicitar al sistema operativo.
- Cada versión sucesiva de la API incluyen actualizaciones del sistema operativo Android.

Los cambios a la API se han diseñado de manera que la nueva API sigue siendo compatible con versiones anteriores. Es decir, la mayoría de los cambios en la API son aditivos e introducen una nueva funcionalidad.

Como parte de la API se actualizan las partes obsoletas pero no se quitan, por lo que las aplicaciones existentes se pueden seguir utilizando.

En un número muy pequeño de casos, las partes de la API son eliminadas, aunque normalmente estos cambios sólo son necesarios para asegurar la robustez de la API.

La API de referencia que ofrece una plataforma de Android se especifica mediante un identificador entero llamado "nivel de API". Cada versión de la plataforma Android soporta exactamente un nivel de API, aunque existe el apoyo implícito de todos los niveles anteriores de la API (hasta el nivel 1).

El identificador de nivel API desempeña un papel clave para asegurar la mejor experiencia posible para los usuarios y desarrolladores de aplicaciones:

- Permite a la plataforma Android describir el Framework máximo de revisión de la API que soporta.
- Permite a las aplicaciones describir la revisión de la API que requieren.
- Esto permite que el sistema pueda negociar la instalación de aplicaciones en el dispositivo del usuario, de tal manera que si es incompatible la versión de las aplicaciones no serán instaladas.

La Tabla 1.7 especifica el nivel de API con el apoyo de cada versión de la plataforma Android.

VERSIÓN DE LA PLATAFORMA	NIVEL API	CÓDIGO DE LA VERSIÓN
Android 4.0.3	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

Tabla 1.7: Nivel de API en función de la Plataforma Android [16**]**

1.4.2 VERSIONES DEL SISTEMA OPERATIVO ANDROID

Android desde su liberación inicial ha visto numerosas actualizaciones. Estas actualizaciones al sistema operativo base típicamente arreglan *bugs*[25] y agregan nuevas funciones. Generalmente cada actualización del sistema operativo Android es desarrollada bajo un nombre en código de un elemento relacionado con postres desde la versión 1.5.

Los siguientes nombres código del sistema operativo Android están en orden cronológico y alfabético:

- Android (v1.0)
- Android (v1.1)
- C: *Cupcake* (v1.5), Magdalena glaseada.

[25] Bugs: Errores de software.

- D: *Donut* (v1.6), Rosquilla.
- E: *Éclair* (v2.0/v2.1), pastel francés conocido en España como pepito o canuto.
- F: *Froyo* (v2.2), (abreviatura de “frozen yogurt”) Yogur helado.
- G: *Gingerbread* (v2.3), Pan de Jengibre.
- H: *Honeycomb* (v3.0/v3.1/v3.2), Panal.
- I: *Ice Cream Sandwich* (v4.0), Sanduche de Helado.

1.4.2.1 Android versión 1.0 [17]

Figura 1.9: Android versión 1.0 [17]

Android 1.0 (Figura 1.9) es la primera versión comercial del software, lanzada el 23 de septiembre de 2008. El HTC *Dream* fue primer dispositivo Android, incorpora las siguientes características de Android 1.0:

- Android *Market* de descarga de aplicaciones y actualizaciones a través de la aplicación de Mercado.
- Navegador web para mostrar HTML y las páginas web XHTML, varias páginas se muestran como pestañas.
- Soporte de cámara, sin embargo esta versión carecía de la opción de cambiar la resolución, el balance de blancos, calidad, etc.
- Carpetas que permiten la agrupación de una serie de iconos de aplicaciones en un solo icono de la carpeta en la pantalla principal.
- Acceso a servidores de correo electrónico, web, POP3, IMAP4 y SMTP.
- Sincronización del correo Gmail.

- Google *Sync* permite la sincronización de contactos con la aplicación *People* en linea.
- Google *Calendar* la sincronización con la aplicación Calendario.
- Google *Maps* con *Latitude* y *Street View* para ver mapas e imágenes satelitales, así como encontrar negocios, locales comerciales y obtener direcciones y la ruta de conducción a través de GPS.
- Google *Search*, permitiendo a los usuarios buscar en Internet y aplicaciones de teléfono, contactos, calendario, etc.
- Google *Talk* de mensajería instantánea que permite mensajes de texto y MMS
- Media Player, que permite la gestión, la importación y reproducción de archivos multimedia sin embargo, esta versión carece de vídeo y soporte Bluetooth estéreo.
- Las notificaciones aparecen en la barra de estado, con opciones para establecer tono de llamada.
- *Voice Dialer* permite la marcación y colocación de las llamadas telefónicas sin tener que escribir un nombre o número.
- Fondo de pantalla que permite al usuario ajustar una imagen de fondo o una foto detrás de los iconos de la pantalla de inicio y los *widgets*[26]
- YouTube Video Player

1.4.2.2 Android versión 1.1 [17]

Figura 1.10: Android versión 1.1 [17]

[26]*Widget*: Son una nueva categoría de mini aplicaciones; diseñadas para proveer de información o mejorar una aplicación mas compleja.

El 9 de febrero de 2009 se lanza la actualización de Android 1 (Figura 1.10), en un principio para el G1 de T-Mobile. La actualización resuelve los errores, cambió la API y añadió una serie de características nuevas:

- Detalles y comentarios disponible cuando un usuario busca empresas en los mapas
- En la pantalla de llamada predetermina establece un corto tiempo de espera, cuando se utiliza el altavoz para ahorrar batería.
- Capacidad para mostrar / ocultar el teclado de marcación.
- Capacidad de guardar archivos adjuntos en mensajes.

1.4.2.3 Android versión 1.5 (*Cupcake*), Magdalena Glaseada [18]

Figura 1.11: Android versión 1.5 (*Cupcake*) [18]

Liberaron la versión 1.5 (Figura 1.11) el 30 de abril de 2009. Hubo varias características nuevas, actualizaciones en el interfaz de usuario y además de un núcleo basado en el Kernel de Linux 2.6.27.

- Posibilidad de grabar y reproducir videos a través del modo *camcorder*[27]
- Capacidad de subir videos a YouTube.
- Un nuevo teclado con predicción de texto.
- Soporte para Bluetooth A2DP[28] y AVRCP[29].

[27] Camcorder: Dispositivo que soporta la modalidad de cámara y video grabadora
[28] A2DP: (*Advanced Audio Distribution Profile*) Distribución de audio avanzada. Define cómo se puede propagar un *stream* de audio (mono o estéreo) entre dispositivos a través de una conexión Bluetooth.
[29] AVRCP: (*Audio/Video Remote Control Profile*) Control remoto de audio/vídeo. Diseñado para ofrecer una interfaz estándar para el control de televisores y aparatos de música entre otros, de forma que un mando único pueda agrupar todo el control. Puede usarse junto con A2DP

- Capacidad de conexión automática para conectar un auricular Bluetooth a cierta distancia.
- Nuevos *widgets* miniatura y carpetas que se pueden colocar en las pantallas de inicio.
- Transiciones de pantalla animadas.
- Grabar y reproducir videos en formato MPEG-4 y 3GP.

1.4.2.4 Android versión 1.6 (*Donut*), Rosquilla [19]

Figura 1.12: Android versión 1.6 (*Donut*) [19]

El SDK 1.6 (Figura 1.12) fue liberado el 15 de septiembre de 2009. Los cambios que se incluyeron son los siguientes:

- Una experiencia mejorada en el Android *Market*.
- Una interfaz integrada de cámara, filmadora y galería
- La galería ahora permite a los usuarios seleccionar varias fotos para eliminarlas.
- Búsqueda por voz actualizada, con respuesta más rápida y mayor integración con aplicaciones nativas, incluyendo la posibilidad de marcar a contactos.
- Experiencia de búsqueda mejorada que permite utilizar marcadores, historiales, contactos y páginas web desde la pantalla de inicio.
- Actualización de soporte para CDMA/EVDO, 802.1x, VPN y *text-to-speech*
- Soporte para resoluciones de pantalla WVGA[30].
- Framework de gestos y herramienta de desarrollo *GestureBuilder.*
- Navegación gratuita de Google.

[30] WVGA: *Wide Video Graphics Array*, en español "VGA alargado", incrementan la resolución horizontal

1.4.2.5 Android versión 2.0/2.1 (*Éclair)*, Pastel francés [20]

Figura 1.13: Android versión 2.0/2.1 (Éclair) [20]

El SDK 2.0 (Figura 1.13) fue liberado el 26 de octubre de 2009. Los cambios que se incluyeron son los siguientes:

- Velocidad de software optimizada.
- Soporte para más tamaños de pantalla y resoluciones.
- Interfaz de usuario renovada.
- Nuevo interfaz de usuario en el navegador y soporte para HTML5.
- Nuevas listas de contactos.
- Una mejor relación de contraste para los fondos.
- Mejoras en Google *Maps* 3.1.2.
- Soporte para Microsoft Exchange.
- Soporte integrado de flash para la cámara.
- Zoom digital.
- *MotionEvent* mejorado para captura de eventos *multi-touch.*
- Teclado virtual mejorado.
- Bluetooth 2.1.
- Fondos de pantalla animados.
- El SDK 2.0.1 fue liberado el 3 de diciembre de 2009.
- El SDK 2.1 fue liberado el 12 de enero de 2010.

1.4.2.6 Android versión 2.2 (*Froyo*), Yogurt Helado [21]

Figura 1.14: Android versión 2.2 (*Froyo*) [21]

El SDK 2.2 (Figura 1.14) fue liberado el 20 de mayo de 2010. Los cambios que se incluyeron son los siguientes:

- Optimización general del sistema Android, la memoria y el rendimiento
- Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT[31].
- Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser.
- Soporte mejorado de Microsoft Exchange reglas de seguridad, reconocimiento automático, sincronización de calendario, limpieza remota.
- Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y *Browser.*
- Funcionalidad de Wi-Fi *hotspot*[32] y *tethering* por USB.
- Permite desactivar el tráfico de datos a través de la red del operador.
- Actualización del Market con actualizaciones automáticas.
- Cambio rápido entre múltiples idiomas de teclado y sus diccionarios.
- Marcación por voz y compartir contactos por Bluetooth.
- Soporte para contraseñas numéricas y alfanuméricas.
- Soporte para campos de carga de archivos en la aplicación Browser.
- Soporte para la instalación de aplicación en la memoria expandible.

[31] JIT: (*Just in time compilation*) traduce el código de alto nivel (código fuente) a leguaje de maquina o (codificado en bits), acelerando su procesamiento.
[32] Hotspot: Punto de acceso a internet por un medio inalámbrico

- Soporte para Adobe Flash 10.1.
- Soporte para pantallas de alto número de Puntos por pulgada, tales como 4" 720p.

1.4.2.7 Android versión 2.3 (*Gingerbread*), Pan de Jengibre [22]

Figura 1.15: Android versión 2.3 (*Gingerbread*) [22]

El SDK 2.3 (Figura 1.15) fue liberado el 6 de diciembre de 2010. Los cambios que se incluyeron son los siguientes:

- Soporte para dispositivos móviles.
- Actualización del diseño de la interfaz de usuario.
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores.
- Soporte nativo para telefonía VoIP SIP[33].
- Soporte para reproducción de videos WebM/VP8[34] y decodificación de audio AAC[35].
- Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves.
- Soporte para *Near Field Communication* (NFC).
- Funcionalidades de cortar, copiar y pegar disponibles a lo largo del sistema
- Teclado multi-táctil rediseñado.

[33] SIP: Protocolo de Inicio de Sesiones
[34] Web Media Project: Código abierto de los decodificadores de video como VP7, VP8.
[35] AAC: Del inglés (*Advanced Audio Coding*) es un formato informático de señal digital audio basado en un algoritmo de compresión con pérdida, un proceso por el que se eliminan algunos de los datos de audio para poder obtener el mayor grado de compresión posible, resultando en un archivo de salida que suena lo más parecido posible al original

- Soporte mejorado para desarrollo de código nativo.
- Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos.
- Recolección de elementos concurrentes para un mayor rendimiento.
- Soporte nativo para más sensores como giroscopios y barómetros.
- Un administrador de descargas para descargar archivos grandes.
- Administración de la energía mejorada y control de aplicaciones mediante el administrador de tareas.
- Soporte nativo para múltiples cámaras.
- Cambio de sistema de archivos de YAFFS a ext4.

1.4.2.8 Android version 3.1 (*Honeycomp*), Panal [23]

Figura 1.16: Android versión 3.1 (*Honeycomp*) [23]

El SDK 3.0 (Figura 1.16) se liberó en febrero de 2011. Los cambios que se incluyeron son los siguientes:

- Mejor soporte para *Tablet.*
- Escritorio 3D con *widgets* rediseñados.
- Sistema multitarea mejorado.
- Mejoras en el navegador web predeterminado, entre lo que destaca la navegación por pestañas, auto-relleno de formularios, sincronización de favoritos con Google Chrome y navegación privada.
- Soporte para video chat mediante Google *Talk*.
- Mejor soporte para redes Wi-Fi.
- Añade soporte para una gran variedad de periféricos y accesorios con conexión USB: teclados, ratones, dispositivos de juego y cámaras digitales.

Cuando un accesorio está conectado, el sistema busca la aplicación necesaria y ofrece su ejecución.

- Los *widgets* pueden redimensionarse de forma manual sin la limitación del número de cuadros que tenga cada escritorio.
- Se añade soporte opcional para redimensionar correctamente las aplicaciones inicialmente creadas para móvil para que se vean bien en *Tablets*.

1.4.2.9 Android versión 4.0 (*Ice Cream Sandwich),* Sanduche de Helado [24]

Figura 1.17: Android versión 4.0 (*Ice Cream Sandwich*) [24]

El SDK 3.0 (Figura 1.17) fue liberado en Octubre de 2011. Los cambios que se incluyeron son los siguientes:

- Versión que unifica el uso en cualquier dispositivo, tanto en teléfonos, *Tablets*, televisiones, *netbooks* etc.
- Interfaz limpia y moderna con una nueva fuente de texto llamada "Roboto", al estilo de *Honeycomb*.
- Opción de utilizar los botones virtuales en la interfaz de usuario, en lugar de los botones táctiles capacitivos.
- Llega la aceleración por hardware, lo que significa que la interfaz podrá ser manejada y dibujada por la GPU y aumentando notablemente su rapidez, su respuesta y evidentemente, la experiencia de usuario.
- Multitarea mejorada, estilo *Honeycomb*. Añadiendo la posibilidad de finalizar una tarea simplemente desplazándola fuera de la lista.

- Ha añadido un gestor del tráfico de datos de internet. El entorno le permite establecer alertas cuando llegue a una cierta cantidad de uso y desactivación de los datos cuando se pasa de su límite.
- Los *widgets* está en una nueva pestaña, que figuran en una lista similar a las aplicaciones en el menú principal.
- El corrector de texto ha sido rediseñado y mejorado, ofreciendo la opción de tocar en una palabra para que nos aparezca una lista con las diferentes opciones de edición y sugerencias de palabras similares.
- Las notificaciones tiene la posibilidad de descartar las que no son importantes y también desplegar la barra de notificaciones con el dispositivo bloqueado.
- La captura de pantalla se realiza tan solo con pulsar el botón de bajar volumen y el botón de encendido.
- La aplicación de la cámara con nuevas utilidades como es la posibilidad de hacer fotografías panorámicas de forma automática.
- Android *Beam* es la nueva característica que nos permitirá compartir contenido entre teléfonos vía *Near Field Communication* (NFC).
- Reconocimiento de voz del usuario.
- Aplicación de teléfono nuevo con la funcionalidad de buzón de voz visual que le permite adelantarlo o retroceder los mensajes de voz.
- Reconocimiento facial, lo que haría que puedas cambiar la vista.
- Las carpetas son mucho más fáciles de crear, con un estilo de arrastrar y soltar.
- Un único y nuevo *framework* para las aplicaciones.
- El usuario tendrá herramientas para ocultar y controlar las aplicaciones que nos instale la operadora o el fabricante, liberando recursos de (ciclos de ejecución y memoria RAM). No obstante, no se podrán desinstalar.
- Soporte nativo del contenedor MKV[36].
- Soporte nativo para el uso de *Stylus* (lápiz táctil).

[36] MKV: Contenedor de archivos audiovisuales Mp3, Avi, Mpeg en un solo formato.

1.4.3 CAPAS DEL SISTEMA OPERATIVO ANDROID [25]

La disposición de las capas del sistema operativo se observa en la Figura 1.18 y son las siguientes:

- Kernel de Linux.
- Librerías.
- Entorno de ejecución.
- *Framework* de Aplicaciones.
- Aplicaciones.

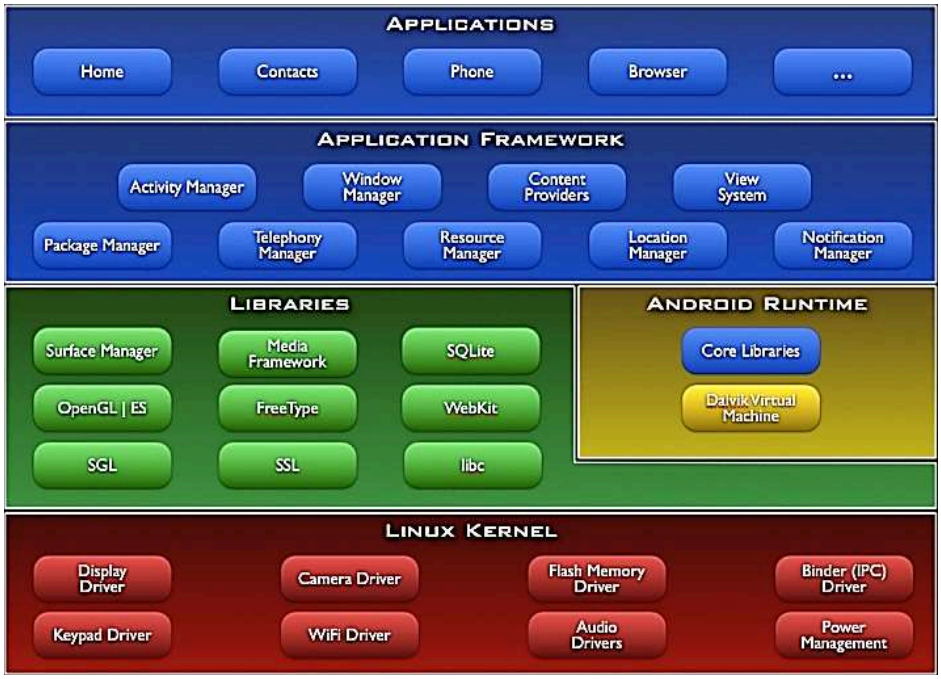


Figura 1.18: Capas del sistema operativo Android [25]

1.4.3.1 Kernel de Linux

El núcleo del sistema operativo Android está basado en el Kernel de Linux versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles (Figura 1.19).

Figura 1.19: Kernel de Linux [25]

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también se evita el hecho de conocer las características precisas de cada dispositivo. Si necesita hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o *driver*) dentro del Kernel que permite utilizarlo desde el software.

El Kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (*networking*), alertas, etc.

1.4.3.2 Librerías

La capa que se sitúa justo sobre el Kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma más eficiente (Figura 1.20).

Figura 1.20: Librerías [25**]**

Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- LIBC: Es una biblioteca personalizada desarrollada por Google para el compilador de C llamado *Bionic*. Incluye todas las cabeceras y funciones según el estándar del lenguaje C.

- *Surface Manager*: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- OpenGL/SL y SGL[37]: Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.
- Media *Libraries*: Proporciona todos los *códec* necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)
- *FreeType*: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- SSL[38]: Protocolo para establecer comunicaciones seguras.
- *SQLite*: Creación y gestión de bases de datos relacionales.
- WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

1.4.3.3 Entorno de Ejecución

Como se puede observar en la Figura 1.21, el entorno de ejecución de Android no se considera una capa independiente, dado que también está formado por librerías. Aquí se encuentra las librerías con la funcionalidad habitual de Java así como otras específicas de Android.

Figura 1.21: Entorno de Ejecución [25**]**

[37] *Open Graphics Library*: Librería abierta para renderización de gráficos 2D y 3D.
[38] SSL: (*Secure Sockets Layer*). Protocolo de encriptación de datos sobre internet.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el *bytecode*[39] Java, se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex que es específico para Dalvik, y por ello no se puede ejecutar aplicaciones Java en Android ni viceversa.

1.4.3.4 Framework de Aplicaciones [26]

Esta capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik. Siguiendo la Figura 1.22 se tiene:

Figura 1.22: Framework de Aplicaciones [25]

1.4.3.4.1 *Activity Manager* (Administrador de Actividades)

Se encarga de administrar la pila de actividades de la aplicación así como su ciclo de vida.

1.4.3.4.2 Windows Manager (Administrador de Ventanas)

Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

[39] Código Fuente escrito en JAVA

1.4.3.4.3 Content Provider (Proveedor de Contenido)

Esta librería crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.

1.4.3.4.4 Views System (Sistemas de Vistas)

En Android, las vistas son elementos que ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google *Maps*.

1.4.3.4.5 Package Manager (Administrador de Paquetes)

Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Paquete se referiere a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

1.4.3.4.6 Telephony Manager (Administrador de Telefonía)

Con esta librería se puede realizar llamadas o enviar y recibir SMS/MMS, No permite remplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

1.4.3.4.7 Resource Manager (Administrador de Recursos)

Con esta librería se puede gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o *Layouts*[40].

1.4.3.4.8 Location Manager (Administrador de Posición)

Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.

1.4.3.4.9 Notification Manager (Administrador de Notificaciones)

Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta

[40] *Layout*: Programa que calcula automáticamente la posición los objetos dentro de una pantalla.

biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los LED del teléfono en caso de tenerlos.

1.4.3.5 Aplicaciones [26]

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no tienen, las aplicaciones nativas (programadas en C o C++), las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

En esta capa se encuentra también la aplicación principal del sistema: Inicio (Home) o lanzador (*launcher*), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso *widgets*, que son también aplicaciones de esta capa.

Los bloques básicos de una aplicación se indican en la Figura 1.23:

Figura 1.23: Bloques básicos de una aplicación [26]

1.4.3.5.1 Activities (Actividades)

Son componentes de la interfaz que corresponde a una pantalla. Puede ser visualizado como una baraja en la que se tiene varias cartas pero solamente una está hasta arriba.

Una aplicación hará una lista de cosas por hacer puede tener una actividad para ingresar las cosas por hacer y otra actividad para mostrar el listado, en conjunto estas actividades conforman la aplicación.

1.4.3.5.2 Intents (Intenciones)

Son mensajes que provocan notificaciones o cambios de estado, que al ser recibidos por actividades o servicios pueden levantar procesos. De esta forma se unen componentes dentro de la misma aplicación o de diferentes aplicaciones.

1.4.3.5.3 Views (Vistas)

Son los componentes de la interfaz de usuario, diferentes vistas pueden agruparse a través de grupos logrando una jerarquía, esto se logra a través de la disposición de los componentes a través de un archivo XML.

1.4.3.5.4 Services (Servicios)

Son componentes que ejecutan operaciones en segundo plano y no tienen una interfaz de usuario. Por ejemplo, al escuchar música, hay un servicio encargado de la reproducción, se ejecuta de fondo y la aplicación que se manipula en ese momento envía mensajes a este servicio diciéndole que se detenga o reproduzca la siguiente canción.

1.4.3.5.5 Content Providers (Proveedores de Contenido)

Representan la abstracción para almacenar y obtener datos permanentes y aplicaciones diferentes. El sistema incluye algunos proveedores de contenido útiles (audio, video, etc.) y además pueden desarrollarse nuevos.

1.4.3.5.6 Adroid.Manifest (Archivo de Configuraciones)

El archivo AndroidManifest.xml es donde se configura la aplicación, se agregan actividades, librerías, configuración de resolución de la pantalla y se asignan permisos.

1.4.3.5.7 Broadcast Receivers (Receptores de Difusión)

Son componentes que responden a avisos y anuncios de difusión (*broadcast*). Estos avisos provienen del sistema (batería baja, una llamada entrante, etc.) y de aplicaciones (pasando avisos de una aplicación a otra). Aunque no muestran una interfaz de usuario algunas veces utilizan barras de progreso para mostrar avances o iconos en la barra de estado. Estos se activan a través de mensajes asincrónicos llamados *intents*.

CAPÍTULO 2

DESARROLLO DE LA APLICACIÓN MULTIMEDIA Y LA COMUNICACIÓN BLUETOOTH

2.1 INTRODUCCIÓN A ECLIPSE IDE Y AL SDK DE ANDROID

Android proporciona acceso a una amplia gama de bibliotecas y herramientas que pueden ser utilizadas para construir aplicaciones variadas. Por ejemplo, Android.Media permite a los desarrolladores administrar los archivos multimedia.

Eclipse IDE es un ambiente de desarrollo integrado, que permite la conexión entre las librerías de Android con el lenguaje de programación JAVA, para facilitar al desarrollador la detección de errores en el momento de la compilación del código.

2.1.1 CONCEPTOS BÁSICOS DE PROGRAMACIÓN EN JAVA [27]

La programación JAVA esta orientada a objetos; significa que se define una plantilla o clase que describe las características y comportamiento de un conjunto de objetos similares

Definición de términos de la programación en JAVA

- **Clase:** Estructura que define como son los objetos, indicando sus atributos y sus acciones. Las clases son declaraciones o abstracciones de objetos, lo que significa que una clase es la definición de un conjunto objetos. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase.
- **Objeto:** En tiempo de ejecución, cuando la máquina virtual de Java encuentra la palabra “new” en el código, utiliza la clase apropiada para crear un objeto, que en sí es una instancia de la clase. Dicho objeto tiene su propio estado y acceso a todos los comportamientos definidos en la clase de la cual se ha instanciado.
- **Variables de instancia o de estado:** Cada objeto (instancia de clase) tiene su propio y único grupo de variables de instancia como fueron

definidas en la clase. De manera colectiva, los valores asignados a cada una de las variables de instancia contenidas en el objeto definirán su estado.

- **Métodos:** Cuando un programador crea una clase, crea a su vez métodos para dicha clase (en lenguaje C sería el equivalente a una función). En los métodos es donde el trabajo real de la clase es realizado, la manipulación de variables y datos para desarrollo de operaciones se realizan dentro de los métodos.
- **Herencia:** Es una propiedad que permite construir nuevos Objetos y Clases a partir de unos ya existentes.
- **Polimorfismo (muchas formas):** Un objeto simple puede ser referenciado a través de muchas formas diferentes, siempre y cuando compartan la misma superclase[41].

2.1.2 HERRAMIENTAS DE DESARROLLO [28]

"Android SDK" incluye una variedad de herramientas especialmente diseñadas para ayudar en el desarrollo de aplicaciones móviles para el sistema operativo Android. Las herramientas más importantes son "Android *Emulator*" y "Android *Development Tools*" *plugin*[42] para Eclipse IDE[43].

SDK también incluye un conjunto de otras herramientas para depuración, empaquetado e instalación de aplicaciones en el dispositivo o emulador. En la Figura 2.1 se observa el SDK *manager* y en la Figura 2.2. AVD *manager*

Se utiliza los siguientes:

- Emulador Android.
- Android *Development Tools* (ADT) extension para Eclipse IDE.
- Dalvik *Debug Monitor Service* (DDMS).
- Android *Debug Bridge.*

[41] Superclase: Posible por la herencia de clases, una superclase es la clase padre de otras clases.
[42] Plugin: Un complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.
[43] IDE: *Integrated Development Environment (Ambiente de Desarrollo Integrado).*

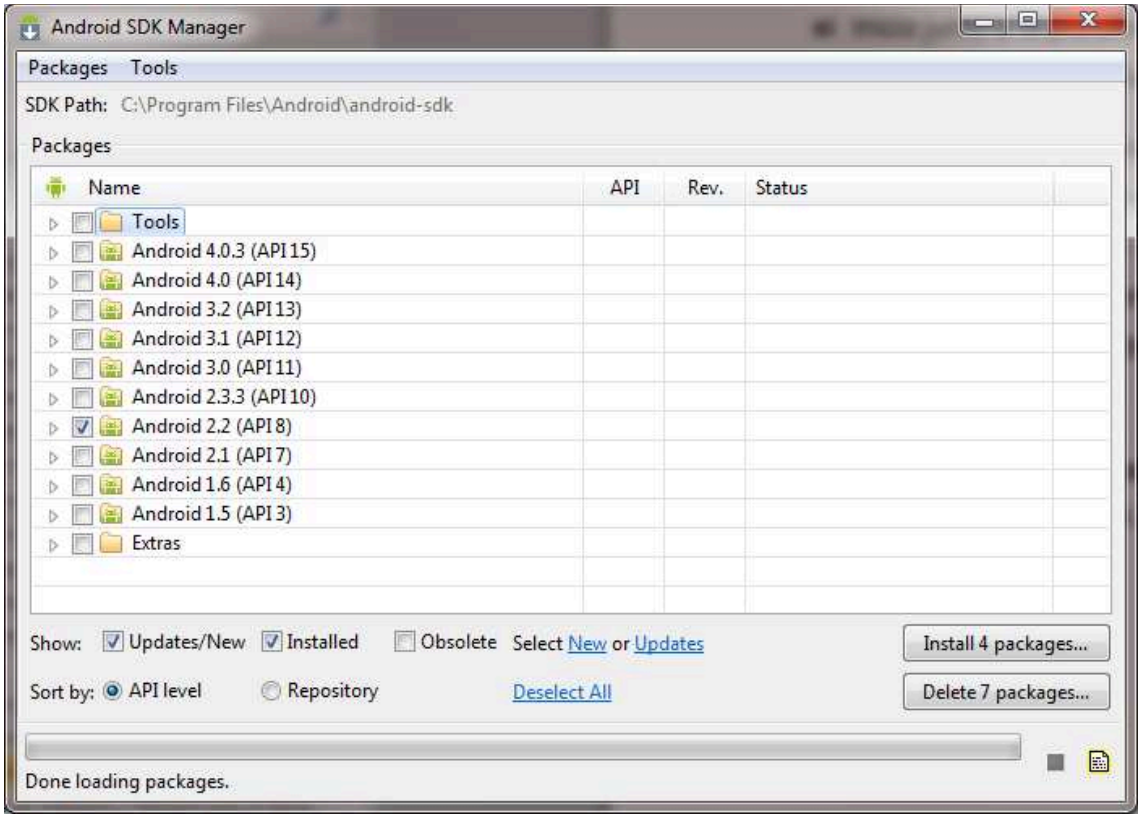


Figura 2.1: SDK *Manager*

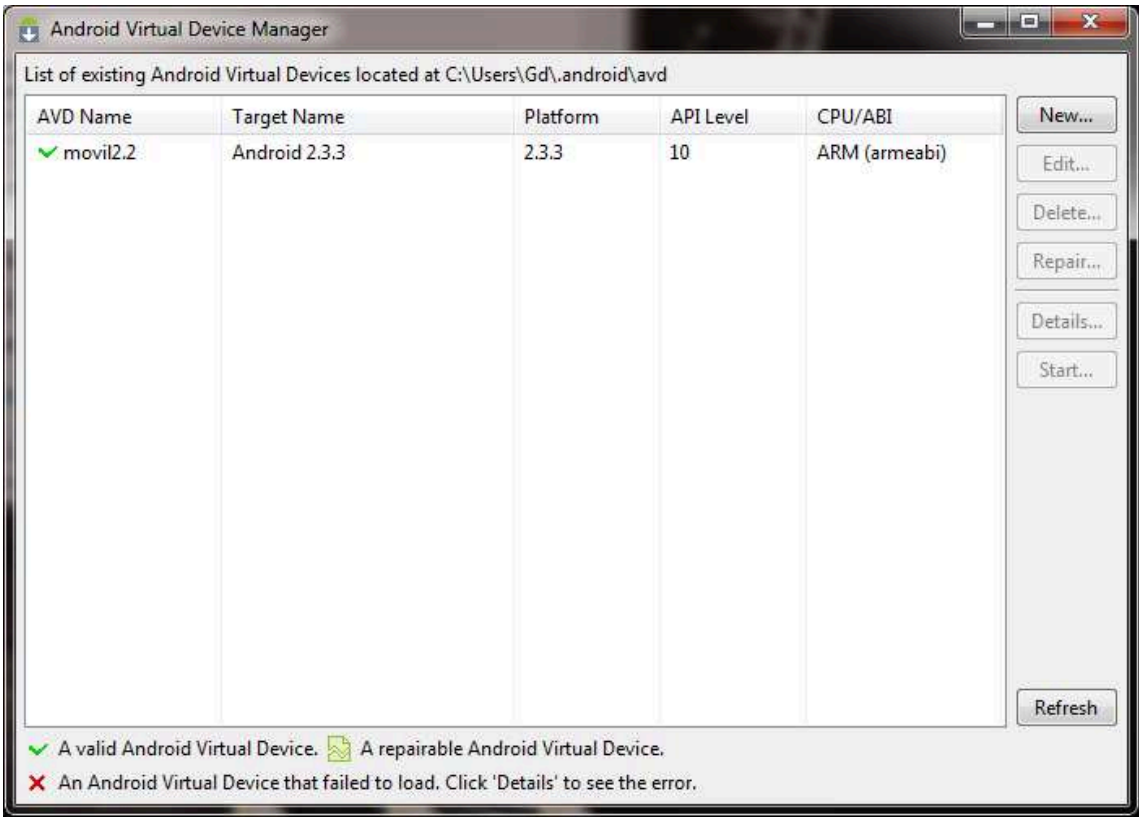


Figura 2.2: AVD *Manager*

2.1.2.1 Emulador de Android

"Android *Emulator*" es un dispositivo virtual que se ejecuta en el computador (Figura 2.3). Este emulador tiene como finalidad ayudar a diseñar y depurar las aplicaciones en un ambiente similar al que existe en un dispositivo real.

Figura 2.3: Emulador de Android

Existen distintas versiones del emulador tanto para Windows, Mac y para Linux.

2.1.2.2 Android *Development Tools* (ADT) extension para Eclipse IDE

El "ADT *plugin*" agrega extensiones al ambiente integrado de Eclipse haciendo que la creación y depuración de las aplicaciones Android sea fácil y rápida.

Si se usa Eclipse, el "ADT *plugin*" ofrece un increíble estímulo para el desarrollo de las aplicaciones Android. A continuación sus características:

- Provee acceso a otras herramientas de desarrollo de Android desde el entorno de Eclipse IDE. Por ejemplo, "ADT" permite el acceso a muchas de las capacidades de la herramienta tales como tomar fotografías de la pantalla, administrar el direccionamiento de puertos, fijar puntos de depuración y examinar la información y procesos directamente dentro de Eclipse.
- Provee un asistente para la creación de proyectos Android, el cual ayuda a estructurar rápidamente todos los directorios y archivos necesarios para

programar la nueva aplicación Android. En la Figura 2.4. se observa los elementos creados inicialmente para un nuevo proyecto.

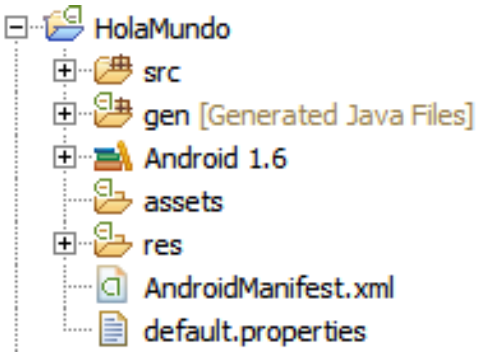


Figura 2.4: Elementos de un proyecto nuevo de Android

- Automatiza y simplifica el proceso de construcción de una aplicación.
- Provee un editor de código Android, ayuda a escribir en el lenguaje de programación XML, utilizado el archivo "AndroidManifest.xml", donde se establecen los permisos de la aplicación.

2.1.2.2.1 Instalación y configuración del ADT plugin en Eclipse IDE

Pasos para la instalación del ADT en Eclipse IDE:

- **Paso 1:** En la barra de herramientas de Eclipse se selecciona Help->Install New Software (Figura 2.5).

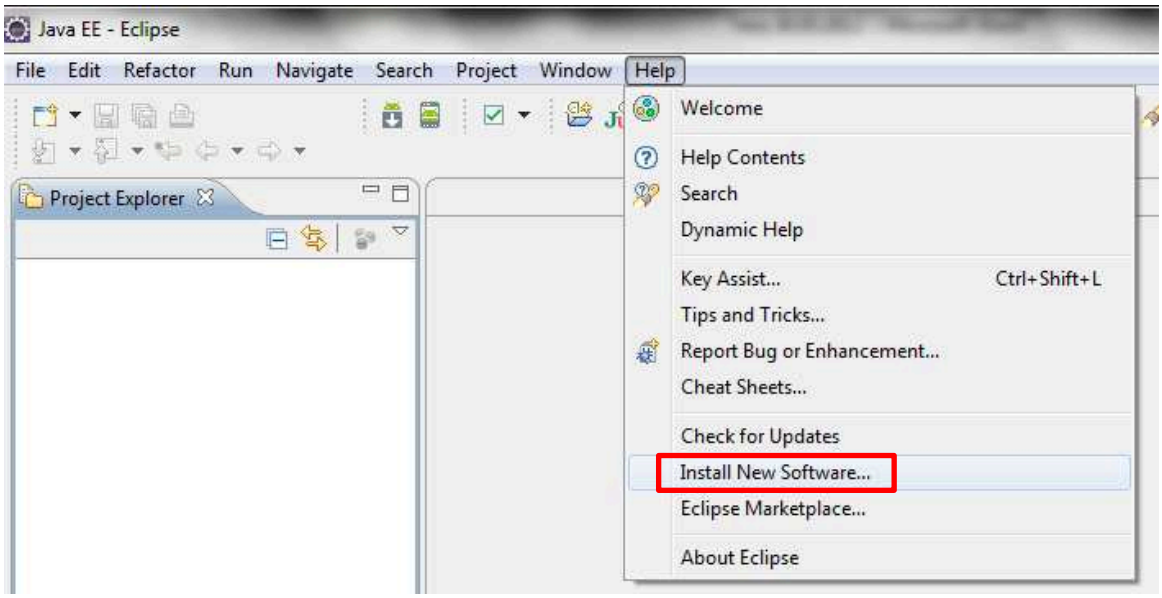


Figura 2.5: Eclipse IDE *Install New Software*

- **Paso 2:** Se introduce la dirección del repositorio donde se puede descargar automáticamente el ADT http://dl-ssl.google.com/android/eclipse/, seleccionamos *Developer Tools* y presionamos *Next>* (Figura 2.6).

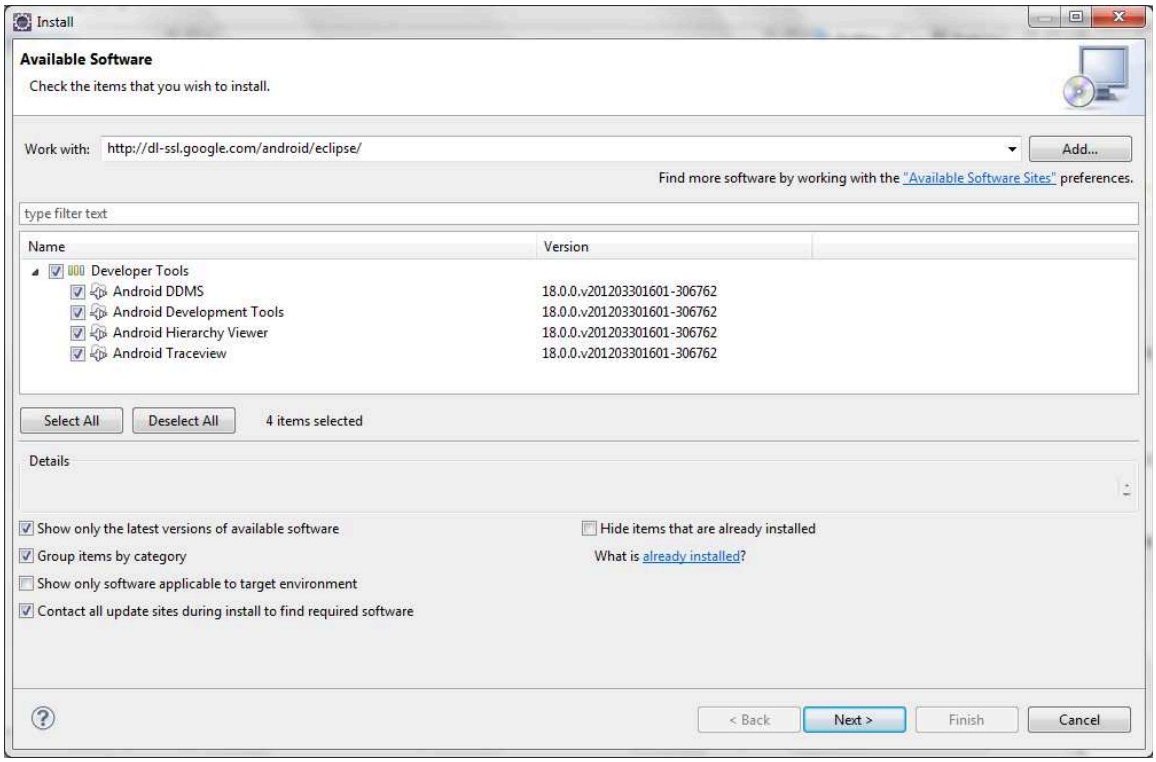


Figura 2.6: Ventana de instalación Eclipse IDE

- **Paso 3:** Informe de los detalles de instalación de ADT (Figura 2.7), aceptar los términos de la licencia (Figura 2.8).

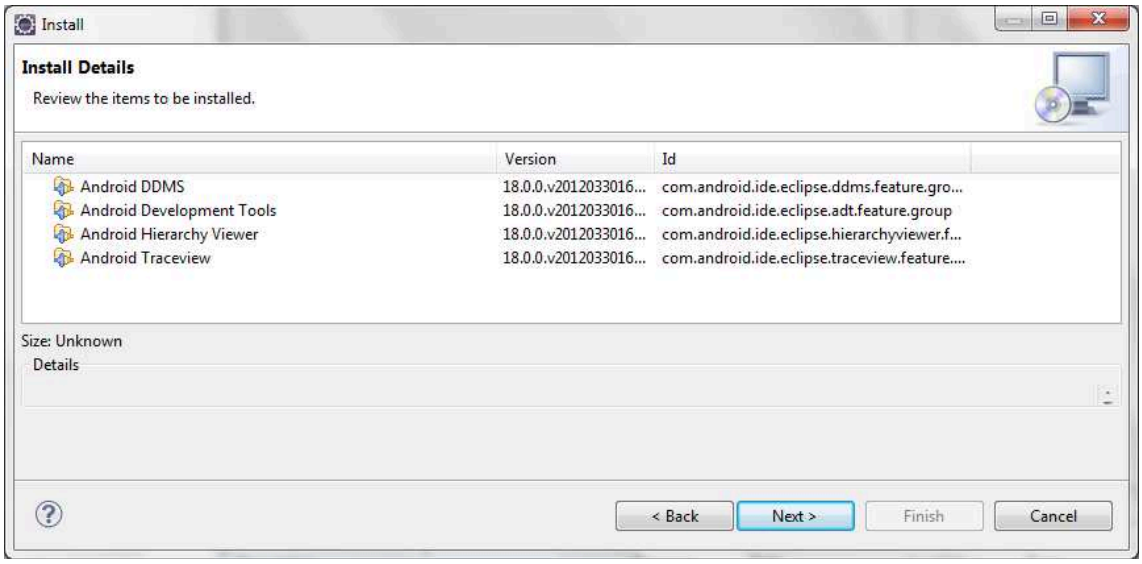


Figura 2.7: Detalles de la instalación de ADT

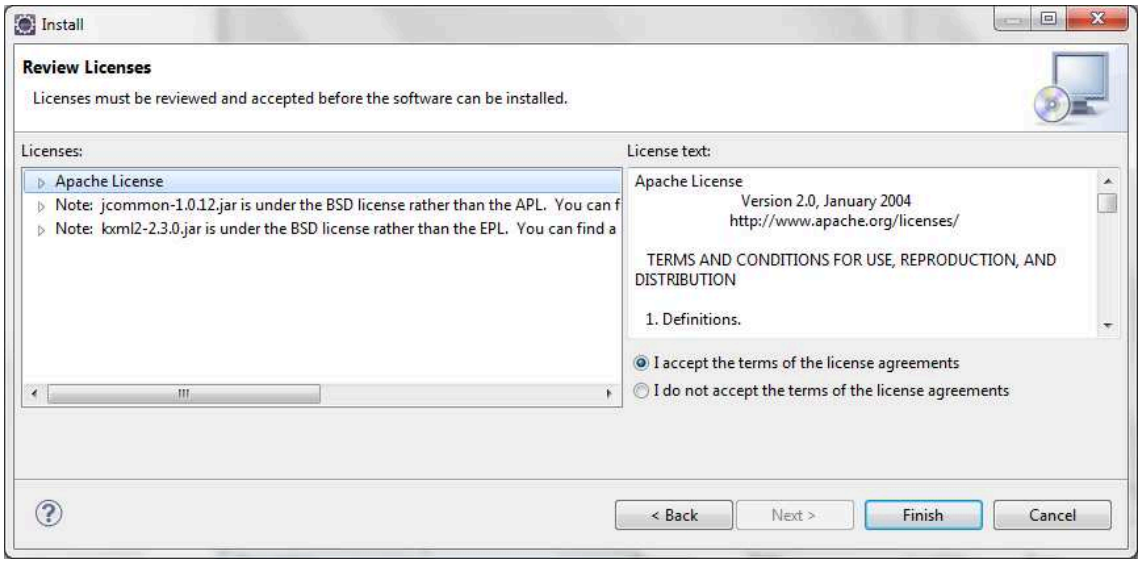


Figura 2.8: Aceptar términos de la Licencia

- Paso 4: Configurar la ruta del SDK de Android en las preferencias de Eclipse IDE (Figura 2.9).

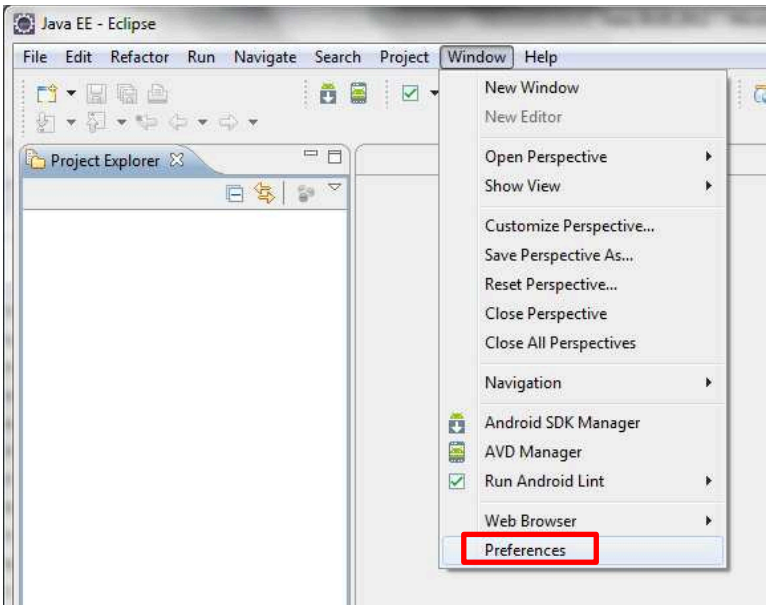


Figura 2.9: Opciones de *Window* en Eclipse IDE

Paso 5: Seleccionar en la lista desplegable Android y se ubica la dirección donde se instaló el SDK de Android, después de esto aparecerán las plataformas instaladas, lo que indica que se realizo correctamente la instalación (Figura 2.10).

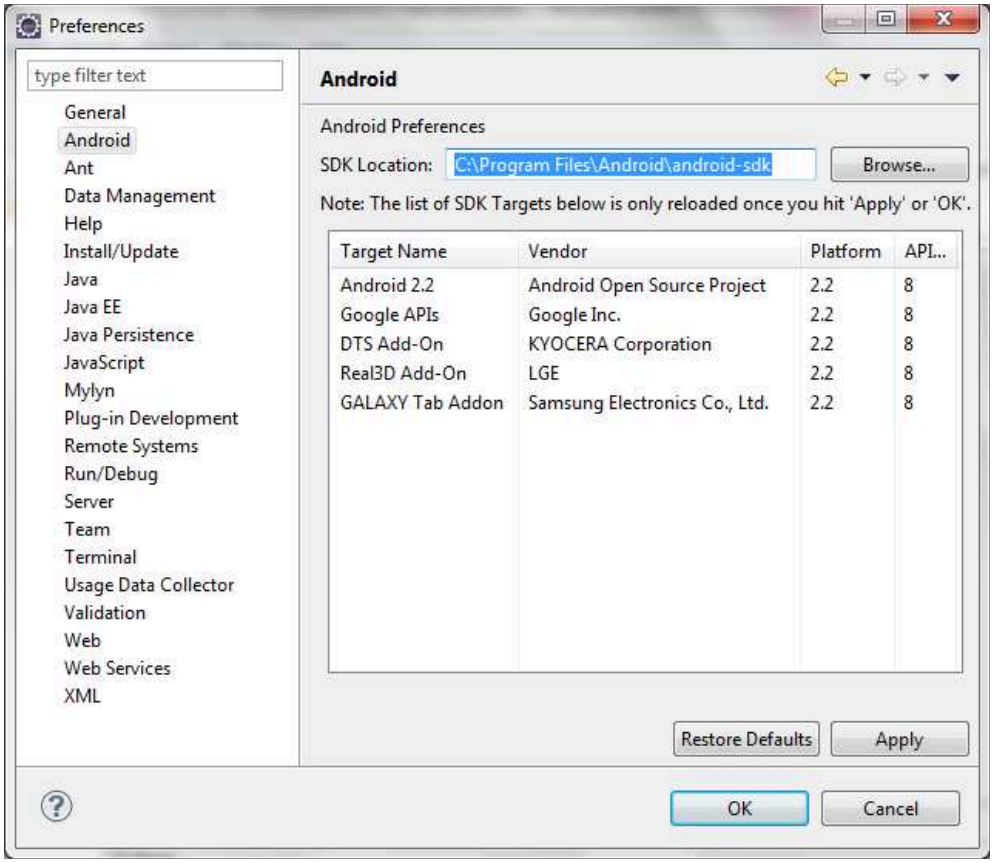


Figura 2.10: Ventana de preferencias de Eclipse IDE

2.1.2.3 *Dalvik Debug Monitor Service* (DDMS)

"*Dalvik Debug Monitor Service*" es una herramienta integrada con la "Dalvik Virtual Machine", y permite administrar los procesos que corren en una instancia de emulador o dispositivo, además de asistir en la depuración de estos.

Se puede usar esta herramienta para:

- Terminar la ejecución de un proceso.
- Seleccionar un determinado proceso y depurar.
- Generar reportes a partir de información de bitácoras (LOG).

2.1.2.4 *Android Debug Bridge* (ADB)

Esta herramienta establece la conexión para la depuración y permite instalar las aplicaciones (archivos ".apk") directamente en el emulador o en el dispositivo.

2.1.3 REQUERIMIENTOS DE HARDWARE Y SOFTWARE DEL COMPUTADOR UTILIZADO EN LA PROGRAMACIÓN DE LA APLICACIÓN [29]

Se determina los requerimientos mínimos de hardware y software del computador utilizado en el desarrollo de la aplicación para el sistema operativo Android.

2.1.3.1 Requerimientos de Hardware

Para utilizar SDK de Android se especifica un espacio mínimo disponible en el disco duro de 560 MBytes, para la plataforma sus ejemplos y la documentación fuera de línea (Tabla 2.1), adicionalmente se necesita 300 MBytes para la instalación de Eclipse IDE y 85 MBytes para Java JDK.

En total se requiere un espacio mínimo disponible en el disco duro de 946 MBytes

El software requerido se puede descargar del siguiente *link* http://developer.android.com/sdk/index.html.

Componente	Tamaño Aproximado	Observación
SDK Tools	35 MB	Requerido
SDK *Platform-tools*	6 MB	Requerido
Android *Platform* (Cada una)	150 MB	Por lo menos una plataforma requerida.
SDK Add-on (Cada una)	100 MB	Opcional.
USB Driver para Windows	10 MB	Requerido.
Ejemplos (por Plataforma)	10MB	Opcional.
Documentación fuera de línea	250 MB	Opcional.

Tabla 2.1: Requerimientos de Disco Duro (HDD) para el SDK Android [29]

Se especifica el hardware mínimo requerido en cuanto a procesador, memoria y puertos, para el adecuado desempeño del computador durante el desarrollo de la aplicación.

- Procesador doble núcleo.
- Memoria RAM 1GB.
- Puerto USB para la conexión del dispositivo

Los requerimientos mencionados se determinan en base a las necesidades requeridas por Android *Emulator*, es la maquina virtual de Android que emula las características de un dispositivo móvil Android y requiere como mínimo 256 MBytes de memoria RAM libre para su uso reservado.

El procesador doble núcleo es requerido para soporte de la ejecución simultánea de Android *Emulator* y Eclipse IDE.

2.1.3.2 Requerimientos de Software

Android funciona bajo los siguientes sistemas operativos:

- Windows XP, Vista y Win7 (32 o 64 bits).
- Mac OS X 10.4.8 o posterior (solo X86).
- Linux en sus diferentes distribuciones (Ubuntu, Dapper y Drake).

Se puede utilizar el entorno de desarrollo de su preferencia:

- Eclipse: El requisito necesario para usar este IDE es tener instalado el paquete JRE (Java *Runtime Enviroment*). Sin embargo, como Eclipse es una herramienta para desarrolladores, se recomienda descargar el paquete JDK (Java *Develpment Kit*), que también incluye JRE.
- Netbeans.
- Apache Ant 1.6.5 para Linux y Mac o Apache Ant 1.7 para Windows.

Se seleccionó Eclipse IDE por ser la herramienta recomendada por Android *Developers* para desarrollar aplicaciones y se encuentra en el siguiente *link* http://www.eclipse.org/downloads/.

El requerimiento para utilizar Eclipse IDE es instalar Java JRE que se encuentra en el *link* http://www.oracle.com/technetwork/java/javase/downloads/jdk-7u3-download-1501626.html.

2.1.4 CREACIÓN DE UN NUEVO PROYECTO EN ANDROID

El nuevo proyecto se crea en el entorno de desarrollo Eclipse IDE, con las herramientas de Android previamente instaladas, facilitando al programador crear la estructura necesaria para desarrollar una la aplicación.

Pasos para la creación de un proyecto Android:

- **Paso 1:** Se selecciona en la barra de herramientas *File -> New -> Other…*(Figura 2.11)

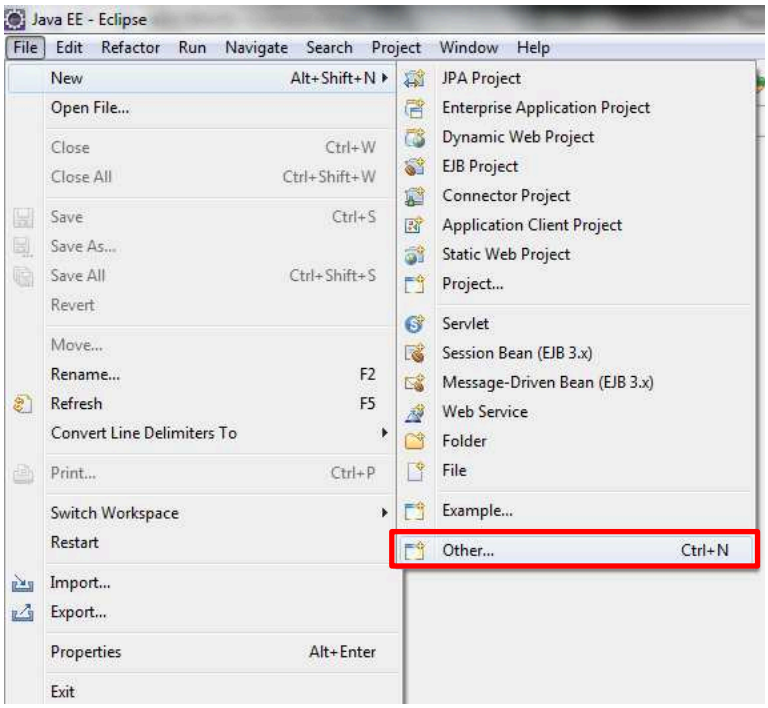


Figura 2.11: Eclipse nuevo proyecto

- **Paso 2:** Se selecciona de lista Android -> Android Project (Figura 2.12).

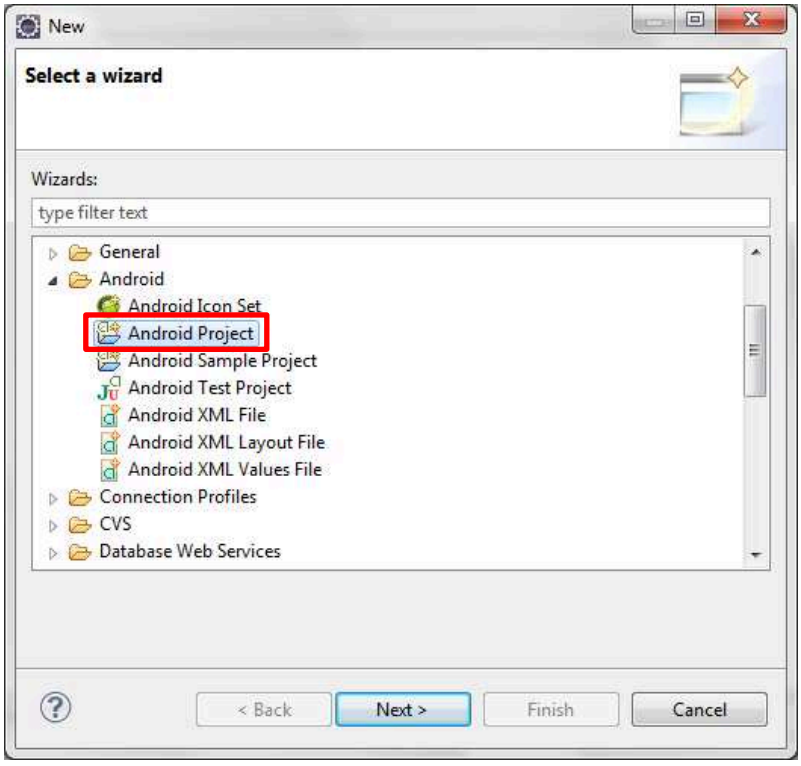


Figura 2.12: Nuevo proyecto Android

- **Paso 3:** Se establece nombre al Proyecto como “HelloAndroid” (Figura 2.13).

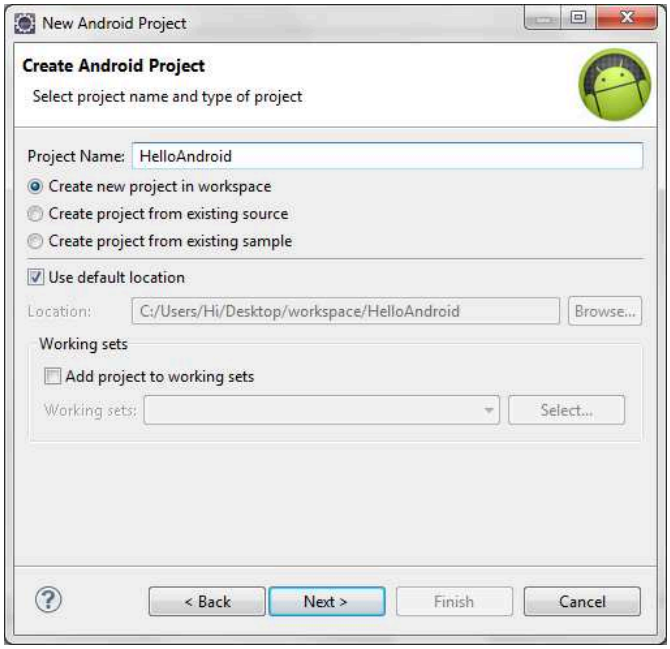


Figura 2.13: Nombre del proyecto

- **Paso 4:** Se selecciona una plataforma instalada “Android 2.2” (Figura 2.14)

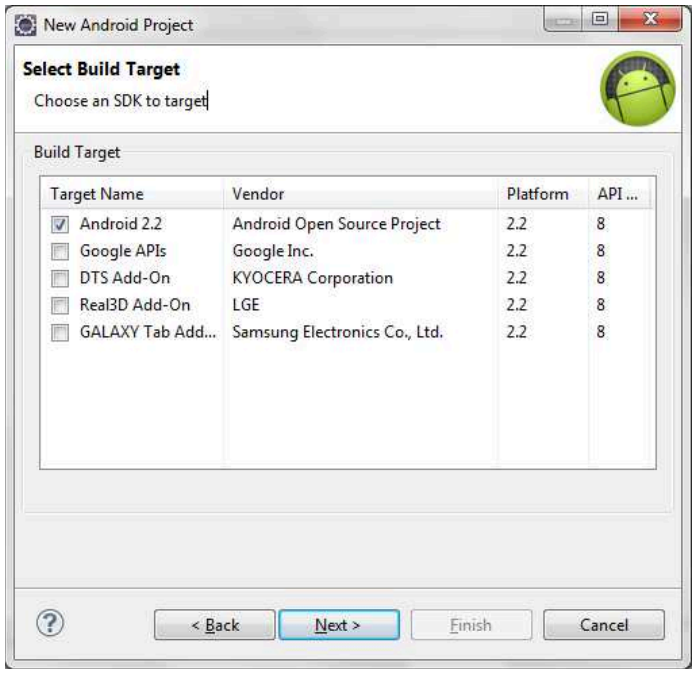


Figura 2.14: Selección de la plataforma Android

- **Paso 5:** Establecer el nombre del paquete “prueba.proyecto” y crea la actividad por defecto con el nombre “HelloAndroidActivity” (Figura 2.15).

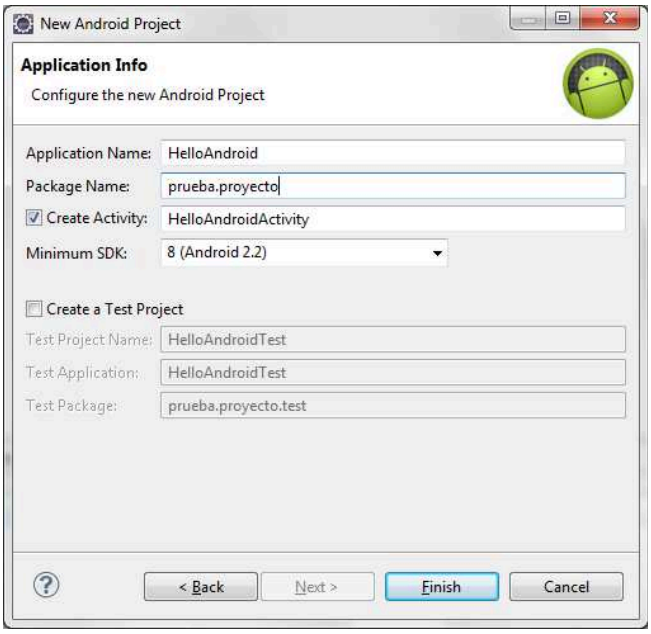


Figura 2.15: Nombre del paquete del nuevo proyecto

Finalmente se presiona *Finish* y el nuevo proyecto ha sido creado.

2.1.5 DESCRIPCIÓN DEL PROYECTO EN ANDROID

Se explica la estructura de un proyecto Android y el contenido de sus carpetas en la Tabla 2.2.

Carpeta	Descripción
/res/drawable/	Contienen las imágenes de la aplicación. Para utilizar diferentes recursos dependiendo de la resolución del dispositivo se divide en varias subcarpetas: • */drawable-ldpi* • */drawable-mdpi* • */drawable-hdpi*
/res/layout/	Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Para definir distintos *Layouts* dependiendo de la orientación del dispositivo se puede dividir en dos subcarpetas: • */layout* • */layout-land*
/res/values/	Contiene otros recursos de la aplicación como por ejemplo cadenas de texto (*strings.xml*), estilos (*styles.xml*), colores (*colors.xml*), etc.
/res/xml/	Contiene los ficheros XML utilizados por la aplicación.
/scr/	Contiene los archivos fuente codificados en java.
/Android X.X	Identifica el API de la librería que se está utilizando.
/AndoridManifest.xml	Se definen los permisos y clases de la aplicación.

Tabla 2.2: Elementos de un proyecto Android [30**]**

La carpeta /src/ contiene el código fuente de la aplicación y de las clases auxiliares. Inicialmente Eclipse creará automáticamente el código por defecto para cualquier aplicación Android (Figura 2.16)

Figura 2.16: Contenido de la carpeta /src

Código creado por defecto en un nuevo proyecto de Android

```
public class HelloAndroidActivity extends Activity { //Hereda todas las
                                                      propiedades de la
                                                      librería Activity.
    public void onCreate(Bundle savedInstanceState) { // Código default para
                                                       crear una pantalla que va
                                                       ha ser utilizada por la
                                                       actividad.
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main); //Define la interfaz gráfica ubicada en
                                        res/layout como "main"
```

La carpeta /res contiene los recursos gráficos, diseños de pantalla y las variables estáticas, creadas inicialmente por el asistente de Android (Figura 2.17)

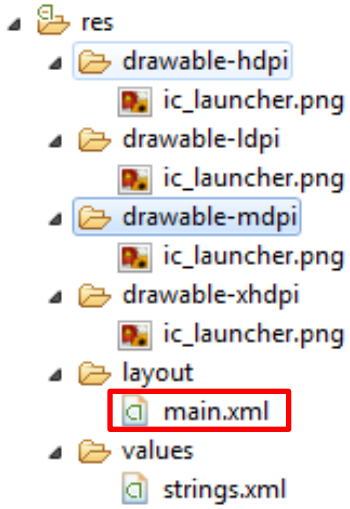


Figura 2.17: Contenido de la carpeta /res

En la raíz del directorio del proyecto encontramos la librería definida por el API seleccionado en la creación del proyecto y el archivo de configuraciones AndroidManifest.xml (Figura 2.18).

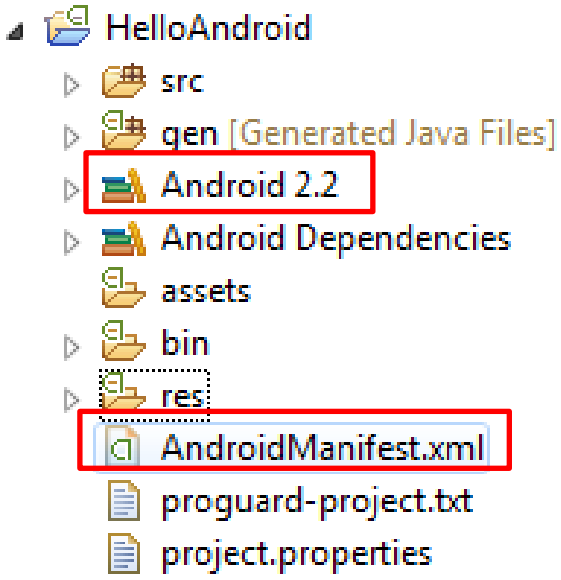


Figura 2.18 Contenido del proyecto

La carpeta /gen y /bin son generadas automáticamente por el compilador de java no se tiene que realizar ningún cambio sobre los archivos presentes en dichas carpetas.

2.2 ANÁLISIS DE REQUERIMIENTO DEL SISTEMA OPERATIVO ANDROID (NIVEL API)

Este análisis se hace a partir de las librerías que serán utilizadas por la aplicación, determinando el nivel API del sistema operativo Android, necesario para el correcto funcionamiento y desempeño de la misma.

2.2.1 DESCRIPCIÓN DE LAS LIBRERÍAS [31]

A cada librería le corresponde un nivel de API se detalla cada una con su correspondiente funcionamiento.

Las siguientes librerías son utilizadas en el desarrollo de la aplicación:

- Librería java.io.File
- Librería java.io.IOException
- Librería java.text.SimpleDateFormat
- Librería java.util.Date
- Librería android.app.Activity
- Librería android.app.AlertDialog
- Librería android.content.Context
- Librería android.content.DialogInterface
- Librería android.content.Intent
- Librería android.database.Cursor
- Librería android.media.MediaPlayer
- Librería android.media.MediaRecorder
- Librería android.media.MediaScannerConnection
- Librería android.provider.MediaStore
- Librería android.widget.Toast
- Librería android.os.Environment

2.2.1.1 Librería java.io.File

Permite la representación de una entidad del sistema de archivos identificado por un nombre de ruta. La ruta puede ser absoluta (relativa al directorio raíz del sistema de archivos) o relativa al directorio actual en el que se ejecuta el programa. Se incluye desde el API nivel 1.

Se incluye en la aplicación para el manejo de archivos multimedia.

2.2.1.2 Librería java.io.IOException

Son señales generadas automáticamente por el programa sin interrumpir la ejecución, relacionadas con el error. Se incluye desde el API nivel 1.

Se tiene en cuenta que también hay varias subclases de esta librería, más específicas tales como FileNotFoundException o EOFException cuando se utiliza archivos.

En la aplicación permite controlar los errores generados por la librería java.io.File.

2.2.1.3 Librería java.text.SimpleDateFormat

Una clase concreta para dar formato y analizar las fechas de una manera sensible. Convierte una fecha en una cadena de caracteres y viceversa. Se incluye desde el API nivel 1.

Se utiliza en la aplicación para generar el nombre los archivos multimedia con fecha y hora.

2.2.1.4 Librería java.util.Date

Representa un momento específico en el tiempo con una precisión de milisegundos. Se incluye desde el API nivel 1.

Se utiliza para adquirir la fecha y la hora del sistema en un determinado momento.

2.2.1.5 Librería android.app.Activity

Esta librería administra el orden de ejecución de las actividades. Presenta al usuario la interfaz de pantalla previamente diseñada. Puede ser una actividad principal o secundaria, informa oportunamente al sistema operativo sobre las acciones requeridas por el usuario. Se incluye desde el API nivel 1.

Si bien las actividades se presentan al usuario como ventanas de pantalla completa, también se puede usar de otras maneras: como ventanas flotantes o incrustadas en el interior de otra actividad.

En el sistema operativo Android las actividades se gestionan como una pila (*Stack*). Cuando una nueva actividad se ha iniciado, se coloca en la parte superior de la pila y se convierte en la actividad principal; la actividad anterior siempre permanece por debajo de ella, y no llegará a la parte frontal hasta que finalice la actividad que se encuentra ejecutando.

Todas las actividades tienen un ciclo de vida como se muestra en la Figura 2.19

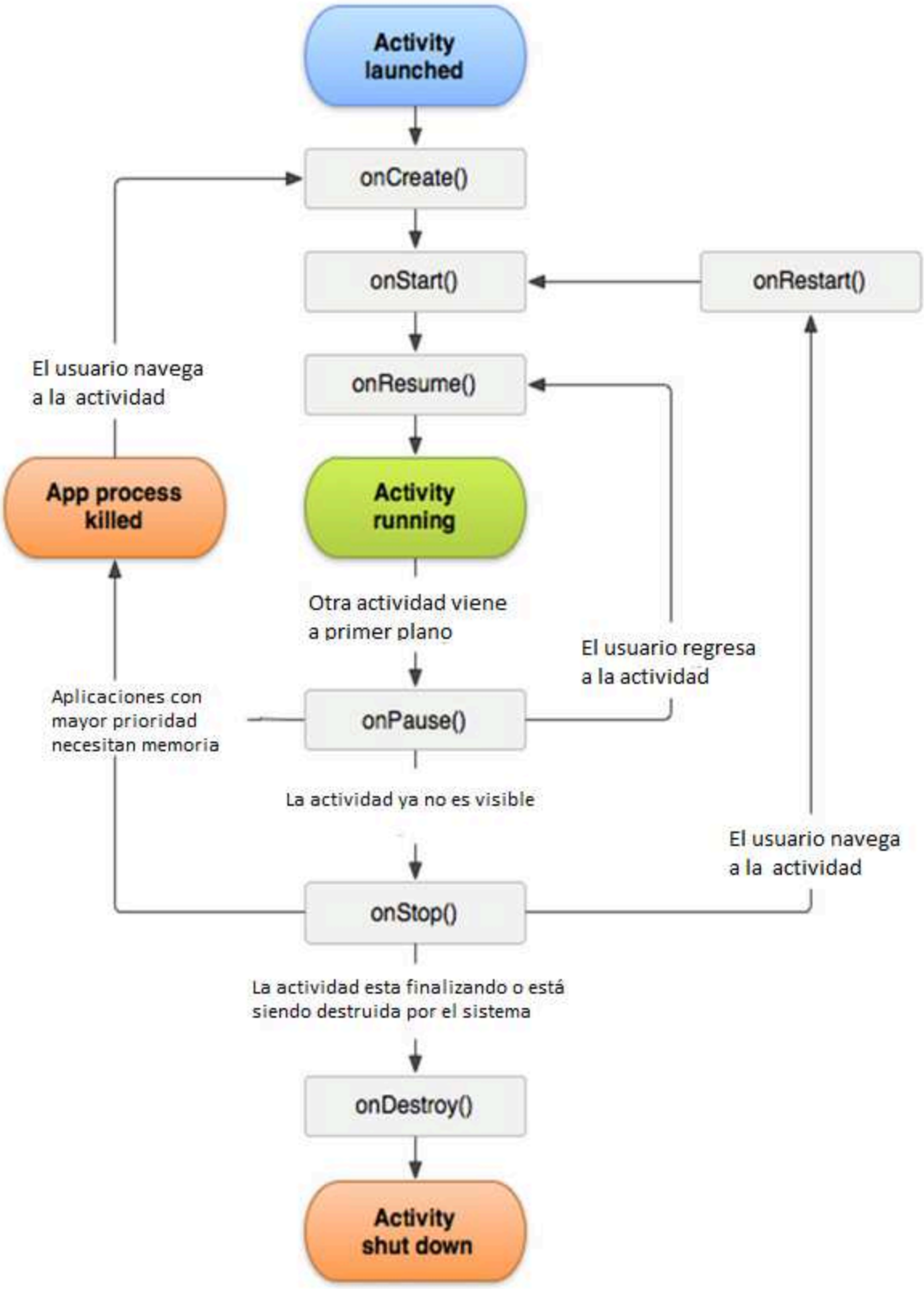


Figura 2.19: Ciclo de vida de una Actividad [31]

Descripción general de los métodos del ciclo de vida de una actividad (Tabla 2.3).

Método	Descripción	Finalizar Proceso (*Killable*?)	Siguiente Método
onCreate()	Se le llama cuando la actividad se crea por primera vez. Aquí es donde se establece el conjunto estático de: Pantallas, los datos de las listas, variables, etc. Este método también ofrece un paquete que contiene la actividad previamente congelada onPause().	No	onStart()
onRestart()	Es llamado después de que su actividad se ha detenido para reanudarla, se activa por petición del usuario.	No	onStart()
onStart()	Se ejecuta automáticamente cuando la actividad se hace visible para el usuario.	No	onResume() ó onStop()
onResume()	Se ejecuta cuando la actividad va a empezar a interactuar con el usuario. En este punto su actividad esta en la parte superior de la pila de actividades.	No	onPause()
onPause()	Se ejecuta cuando el sistema está a punto de comenzar o reanudar una nueva actividad. Esto es generalmente utilizado para liberar memoria y CPU finalizando animaciones, procesos, etc. Se ejecuta onResume() si la actividad vuelve a ser mostrada en pantalla, o OnStop() si desea que sea invisible para el usuario.	Si	onResume() ó onStop()
onStop()	Se ejecuta cuando la actividad ya no es visible para el usuario, debido a que otra actividad se ha reanudado o también ocurre cuando una nueva actividad se está iniciando. Se ejecuta onRestart () si esta actividad va a volver a interactuar con el usuario y OnDestroy () si esta actividad va a ser finalizada.	Si	onRestart() ó onDestroy()
onDestroy()	Se ejecuta cuando la actividad esta siendo finalizada por el sistema. Esto ocurre cuando la actividad está terminando por el comando finish(), o porque el sistema se encuentra temporalmente destruyendo esta instancia de la actividad para ahorrar espacio de memoria.	Si	Ninguno

Tabla 2.3: Descripción de los métodos del ciclo de vida de una actividad [31**]**

Existen dos métodos que se incluyen en la mayoría de clases de una aplicación Android, son indispensables porque permiten establecer el diseño de la interfaz gráfica y la conexión con el código fuente y son los siguientes:

- Método onCreate()
- Método onPause()

2.2.1.5.1 Metodo onCreate

Es donde se inicia la actividad, normalmente se llama a setContentView(int) con un recurso de diseño grafico previamente generado (interfaz de usuario), y se usa findViewById(int) para identificar los elementos de la interfaz que interactúan con la programación.

2.2.1.5.2 Metodo onPause

El método se ejecuta automáticamente cuando el usuario abandona la aplicación sin finalizarla. Es importante capturar los datos de la memoria RAM para no perder información que allí se encontraba.

Para hacer uso de varias actividades en la misma aplicación, deben ser declaradas explícitamente en el archivo de configuración AndroidManifest.xml.

2.2.1.6 Librería android.app.AlertDialog

Una subclase de diálogo que puede mostrar uno, dos o tres botones. Si desea mostrar una cadena de caracteres en este cuadro de diálogo, se puede utilizar el método setMessage(). Se incluye desde el API nivel 1.

En la aplicación se incluye en el método eliminación de archivos, permitiendo al usuario seleccionar una respuesta positiva o negativa.

2.2.1.7 Librería android.content.Context

Permite obtener la información global acerca de un entorno de aplicación (Clase). Esta es una clase abstracta, cuya aplicación está prevista por el sistema Android. Permite el acceso a recursos específicos de la aplicación, así como las llamadas para las operaciones de lanzamiento, difusión y recepción de las intenciones, etc. Se incluye desde el API nivel 1.

En la aplicación se utiliza para obtener información referente a elementos de la clase en ejecución.

2.2.1.8 Librería android.content.DialogInterface

Constructor de los mensajes de alerta y métodos públicos utilizados por AlertDialog().

2.2.1.9 Librería android.content.Intent

Contiene un conjunto de métodos que permiten configurar e iniciar actividades, provee los comandos startActivity (), startActivityForResult (). Se incluye desde el API nivel 1.

2.2.1.10 Librería android.database.Cursor

Proporciona acceso de lectura-escritura al conjunto de resultados obtenidos de una consulta a la base de datos. Se incluye desde el API nivel 1.

En la aplicación se utiliza para el manejo de tablas, obtenidas de la biblioteca multimedia.

2.2.1.11 Librería android.media.MediaPlayer

Se utiliza para controlar la reproducción de archivos de audio, vídeo y streaming multimedia. Se incluye desde el API nivel 1.

2.2.1.12 Librería android.media.MediaRecorder

Se utiliza para grabar video y audio, se encuentra incluida en el sistema operativo Android desde el API nivel 1 hasta el nivel 8.

2.2.1.13 Librería android.media.MediaScannerConnection

Proporciona una forma de registrar un archivo creado o descargado, al servicio lector de los medios multimedia. El servicio de lector de los medios va a leer los metadatos[44] del archivo y agregar al proveedor de contenidos. Se incluye desde el API nivel 1.

El MediaScannerConnectionClient proporciona una interfaz para el servicio de lector de los medios de comunicación, para crear el URI[45] de un nuevo archivo escaneado en el cliente de la clase MediaScannerConnection.

La aplicación registra en la base de datos todos los archivos multimedia creados o eliminados en el trascurso de su ejecución.

---

[44] Metadatos: Información adicional como artista, álbum, genero etc.

[45] URI: (*Uniform Resource Identifier*) Es una cadena compacta de caracteres para la identificación de un recurso abstracto o físico.

2.2.1.14 Librería android.provider.MediaStore

El proveedor de medios multimedia contiene metadatos para todos los medios disponibles en dispositivos de almacenamiento internos y externos.

La aplicación solicita información sobre los archivos multimedia almacenados en el dispositivo. Se incluye desde API nivel 1 hasta el nivel 8.

2.2.1.15 Librería android.widget.Toast

Es una vista que contiene un mensaje rápido y corto para el usuario. La clase toast ayuda a crear y mostrar los mismos. Se incluye desde el API nivel 1.

La aplicación despliega mensajes emergentes informativos utilizando esta librería.

2.2.1.16 Librería android.os.Environment

Provee el acceso a las variables de entorno del sistema operativo como por ejemplo el estado de la memoria extraíble, directorio de almacenamiento de archivos multimedia, etc. Se incluye desde el API nivel 1.

Como se ha descrito las librerías requieren un API nivel 1 hasta el nivel 8, porque se han modificado aumentando su funcionalidad y determinan el nivel API mínimo requerido para la aplicación en desarrollo.

2.2.2 MÉTODOS Y CLASES DE LAS LIBRERÍAS QUE DETERMINAN EL REQUERIMIENTO API DE LA APLICACIÓN

Se describe los métodos, clases y configuraciones que son utilizados para ejecutar la aplicación multimedia

A continuación se describe el método y la configuración que determinan el nivel de API:

- Método getExternalStoragePublicDirectory(Environment.DIRECTORY)
- Configuración *CamcorderProfile*

2.2.2.1 Método getExternalStoragePublicDirectory(Environment.DIRECTORY) [32]

Este método devuelve la ubicación estándar y común donde es recomendado almacenar archivos de video, audio e imágenes. El directorio es compartido públicamente, por lo que otras aplicaciones pueden descubrir, leer, modificar y

borrar los archivos guardados en este lugar. Este método está disponible en Android 2.2 (Nivel API 8).

Los campos admitidos para este método son:

- DIRECTORY_ALARMS, no es utilizado.
- DIRECTORY_DCIM, utilizado para almacenar las imágenes generadas por la aplicación.
- DIRECTORY_DOWNLOADS, no es utilizado.
- DIRECTORY_MOVIES, utilizado para almacenar los archivos de video generados por la aplicación
- DIRECTORY_MUSIC, utilizado para almacenar los archivos de audio generados por la aplicación
- DIRECTORY_NOTIFICATIONS, no es utilizado.
- DIRECTORY_PICTURES, no es utilizado.
- DIRECTORY_PODCASTS, no es utilizado.
- DIRECTORY_RINGTONES, no es utilizado.

Es fácil reconocer el uso de cada carpeta, ya que su nombre corresponde a su contenido.

2.2.2.2 Configuración *CamcorderProfile* [33]

Recupera los ajustes de la videocámara predefinidos en el perfil para las aplicaciones de cámara de vídeo. Estos valores son de sólo lectura.

Cada perfil especifica el siguiente conjunto de parámetros:

- El formato del archivo de salida.
- Codificador de vídeo (Códec).
- Codificador de audio (Códec).
- Tasa de bits de video en Bps.
- Velocidad de fotogramas del vídeo en Fps.
- Tamaño de vídeo ancho y alto.
- Tasa de bits de audio en Bps
- Frecuencia de muestreo de audio.

- Número de canales de audio para la grabación.

Esta clase está disponible en Android 2.2 (Nivel API 8).

Se define el interfaz de programación de aplicaciones (API) para el desarrollo de la aplicación cumpliendo con los objetivos del actual proyecto en **"Android 2.2 FROYO (NIVEL API 8)"**.

2.3 MEDIA RECORDER Y MEDIA PLAYER DEL SISTEMA OPERATIVO ANDROID

Para el desarrollo de la aplicación multimedia se describe el funcionamiento del MediaRecorder y del MediaPlayer, por medio de su respectivo diagrama de estados.

2.3.1 DESCRIPCIÓN MEDIARECORDER [34]

Se utiliza para grabar audio y video en el dispositivo, configurando los parámetros de salida como: formato, calidad, tiempo máximo de duración y el hardware que va ser utilizado por el MediaRecorder.

Código fuente ejemplo para la implementación de un simple MediaRecorder utilizando el formato de menor calidad y tamaño de video para dispositivos móviles.

```
MediaRecorder recorder = new MediaRecorder(); // Creacion del Objeto
      recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
      recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
      recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
      recorder.setOutputFile(Path de destino);
      recorder.prepare();
      recorder.start(); // Comienza la grabación
      recorder.stop();// Detiene la grabación
      recorder.reset();
      recorder.release(); // Se libera el objeto
```

El formato 3gp es un formato de video, creado específicamente para dispositivos de capacidades limitadas, pero hoy en día la tecnología permite capturar en resoluciones muy altas, por lo que se han desarrollado nuevos formatos como mpeg4 de alta definición mejorando la calidad de video.

Se presenta el diagrama de estados del MediaRecorder en la Figura 2.20.

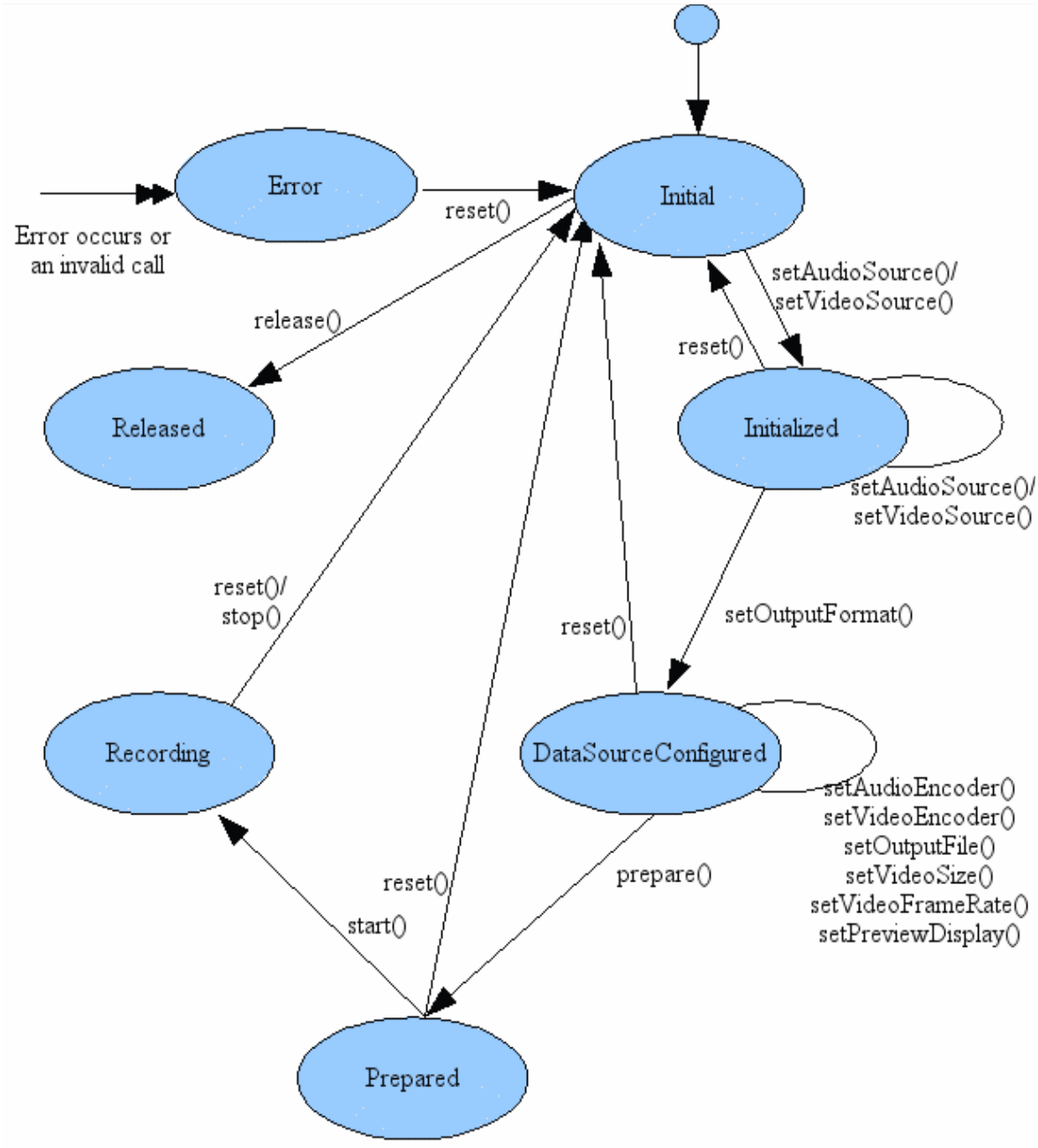


Figura 2.20: Diagrama de estados MediaRecorder [34]

Los estados del MediaRecorder son:

- Estado *Initial* (Inicial).
- Estado *Initialized* (Inicializado).
- Estado *DataSourceConfigured* (Fuente de datos configurada).
- Estado *Prepared* (Preparado).

- Estado *Recording* (Grabando).
- Estado Released (Liberado).
- Estado *Error.*

Descripción de estados del MediaRecorder considerando que la ejecución es secuencial y si el estado anterior no está correctamente configurado, no existe la posibilidad de avanzar al siguiente estado según el diagrama de la Figura 2.20.

2.3.1.1 Estado *Initial* (Inicial)

El MediaRecorder se encuentra en este estado justo después de su creación dentro del código como un objeto.

Los métodos disponibles en el estado son:

- Método **setAudioSource()**, tiene como opciones:
  - Opción DEFAULT, escoge el dispositivo de entrada de audio por defecto del sistema operativo.
  - Opción MIC, escoge el micrófono del dispositivo como entrada de audio predeterminada.
- Método **setVideoSource()**, tiene como opciones:
  - Opción DEFAULT, escoge el dispositivo de entrada de video por defecto del sistema operativo.
  - Opción CAMERA, escoge la cámara del dispositivo como entrada de video predeterminada.
- Método **Released()**, libera el objeto MediaRecorder creado anteriormente.

2.3.1.2 Estado *Initialized* (Inicializado)

El MediaRecorder se encuentra en este estado cuando se han configurado los parámetros de los dispositivos de captura de audio y video.

Los métodos disponibles en el estado son:

- Método **setAudioSource()**, tiene como opciones:
  - Opción DEFAULT, escoge el dispositivo de entrada de audio por defecto del sistema operativo.

- Opción MIC, escoge el micrófono del dispositivo como entrada de audio predeterminada.

- Método **setVideoSource()**, tiene como opciones:
  - Opción DEFAULT, escoge el dispositivo de entrada de video por defecto del sistema operativo.
  - Opción CAMERA, escoge la cámara del dispositivo como entrada de video predeterminada.
- Método **setOutputFormat()**, define la extensión que identifica al archivo de salida del MediaRecorder y tiene las siguientes opciones:
  - Opción AMR, para archivos de audio
  - Opción ThreeGpp, para archivos de video
  - Opción Mpg_4, para archivos de video de mejor calidad
- Método **Reset(),** borra todas las configuraciones de los métodos anteriores y regresa al estado inicial.

2.3.1.3 Estado *DataSourceConfigured* (Fuente de datos configurada).

Los métodos permitidos para este estado son:

- Método **setAudioEncoder()** define el codificador para captura de audio con las siguientes opciones:
  - Opción AMR_NB, (*Adaptive Multi-Rate Narrow Band*) Codificador de audio optimizado para la trasmisión de datos GSM con ancho de banda de (50-4000)Hz
  - Opción AMR_WB, (*Adaptive Multi-Rate Wide Band*) Corresponde al ARM extendido con mejor calidad y una acho de banda (50-7000)Hz
  - Opción AAC, (*Advanced Audio Coding*) Desarrollado para mejorar las características del mp3 soportando múltiples canales de audio y un ancho de banda de 48Khz
- Método **setVideoEncoder()** define el codificador para captura de video con las siguientes opciones:
  - Opción H263, formato de alta compresión de video y baja calidad diseñado para dispositivos móviles la extensión es .3gp
  - Opción H264+, Formato de video de alta calidad, la extensión es .mp4 y se incluye desde Android 3.0

  - Opción MPEG-4_SP, Formato de video de calidad media, es una adaptación del estándar MPG4 para dispositivos móviles con capacidad de almacenamiento limitada.

- Método **setOutputFile()**, indica la dirección (PATH) del archivo de salida donde se va almacenar la información generada por el MediaRecorder.
- Método **setVideoSize()**, se indica el tamaño del video capturado en pixeles Largo por Ancho.
- Método **setPreviewDisplay()**, indiaca al MediaRecorder, el espacio de pantalla que puede usar, para mostrar el video que esta siendo capturado.
- Método **prepare()**, indiaca al MediaRecorder que todos los parámetros para la captura de video o audio esta completa; en caso de ejecutar este comando con parámetros erróneos el sistema operativo avisara oportunamente mediante el método IOExeption.
- Método **Reset()**borra todas las configuraciones de los métodos anteriores y regresa al estado inicial.

2.3.1.4 Estado *Prepared* (Preparado)

Aquí el MediaRecorder esta listo para grabar y pude utilizar los siguientes métodos:

- Método **start()**, indica la MediaRecorder que empiece la captura.
- Método **Reset(),** borra todas las configuraciones de los métodos anteriores y regresa al estado inicial.

2.3.1.5 Estado *Recording* (Grabando)

El MediaRecorder se encuentra grabado y puede utilizar los siguientes métodos:

- Método **stop()**, indica la MediaRecorder que detenga la captura.
- Método **Reset()**, detiene la grabación y borra todas las configuraciones de los métodos anteriores y regresa al estado inicial.

2.3.1.6 Estado Released (Liberado)

Este estado solo ocurre cuando se ha liberado el objeto MediaRecorder por el método **release()**.

2.3.2 DESCRIPCIÓN MEDIAPLAYER [35]

Su función es reproducir audio y video en el dispositivo móvil, configurando los parámetros de entrada tales como: dirección del archivo, superficie para la reproducción de video.

Se presenta el diagrama de estados del MediaPlayer en la Figura 2.21.

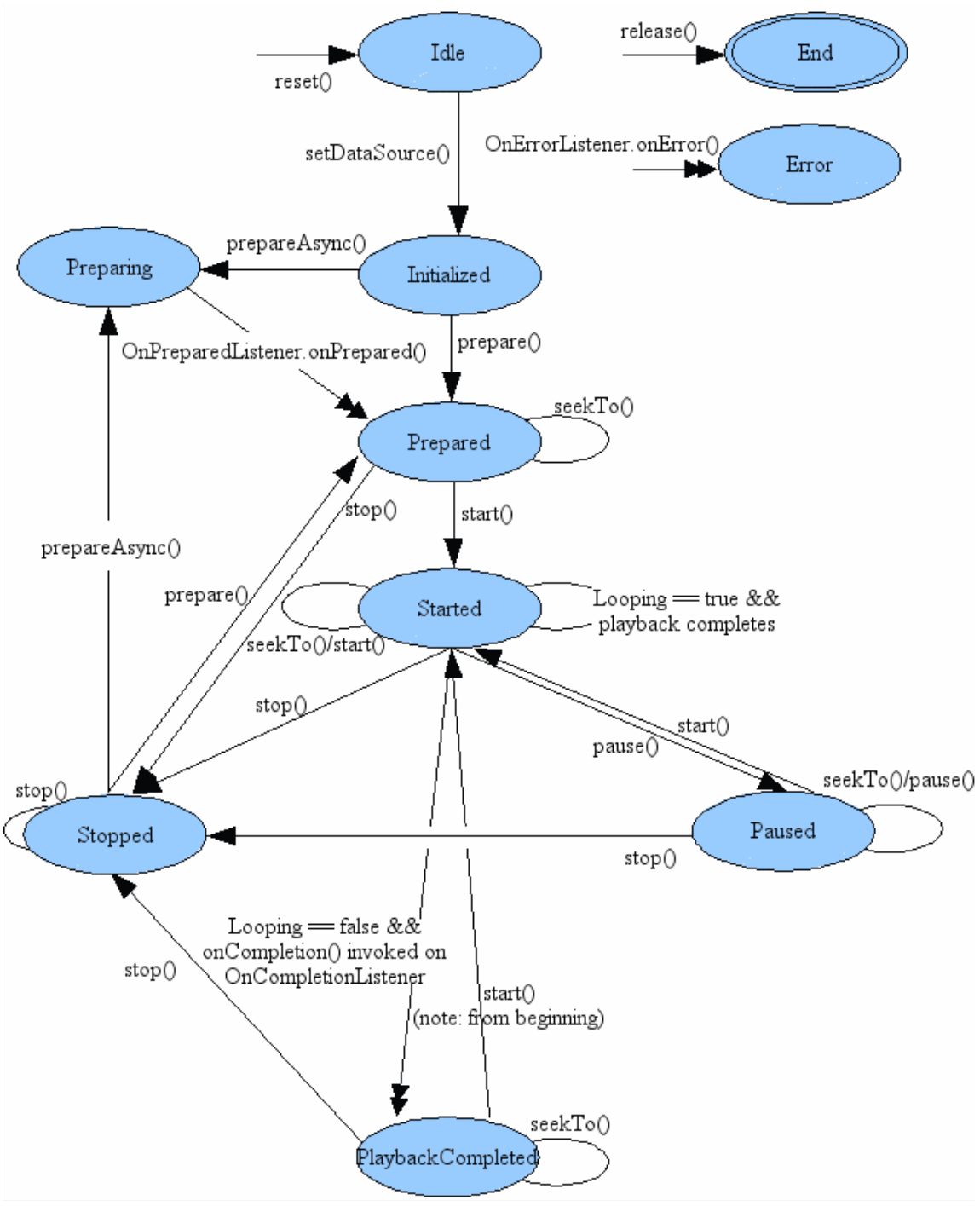


Figura 2.21: Diagrama de estados MediaPlayer [35]

Los estados del MediaPlayer son:

- Estado *Idle* (En espera).
- Estado *Initialized* (Iniciado).
- Estado *Prepared* (Preparado).
- Estado *Started* (Reproducción en curso).
- Estado *Paused* (Reproducción en pausa).
- Estado *Stopped* (Reproducción detenida).
- Estado *PaybackCompleted* (Reproducción finalizada).

Descripción de los estados del MediaPlayer considerando la ejecución secuencial.

2.3.2.1 Estado *Idle* (En espera)

El MediaPlayer se encuentra en este estado justo después de su creación dentro del código como un objeto.

Los métodos disponibles en el estado son:

- Método **setDataSource()** se indica la ruta de acceso (PATH) del archivo de audio o video
- Método **Reset()** borra todas las configuraciones de los métodos anteriores y regresa al estado inicial.

2.3.2.2 Estado *Initialized* (Iniciado)

Los métodos permitidos para este estado son:

- Método **prepare()** tiene que ser ejecutado obligatoriamente para cualquier reproducción de forma sincrónica; se utiliza en la mayoría de casos porque no genera alertas adicionales y simplemente indiaca al MediaPlayer que esta preparado para reproducir.
- Método **prepareAsync()** se ejecuta cuando el fichero multimedia que se desea reproducir es un streaming de datos no almacenados en el dispositivo. Genera la señal onPrepared() que indica que el MediaPlayer esta preparado.

2.3.2.3 Estado *Prepared* (Preparado)

Los métodos permitidos para este estado son:

- Método **start()** indica la MediaPlayer que comience la reproducción.
- Método **stop()** detiene el reproductor y permanece en el estado stopped.
- Método **seekto()** busca una posición de tiempo en milisegundos dentro del archivo multimedia seleccionado.

2.3.2.4 Estado *Started* (Reproducción en curso)

Los métodos permitidos mientras el MediaPlayer se encuentra en reproducción son:

- Método **seekto()** busca una posición de tiempo en milisegundos dentro del archivo multimedia seleccionado.
- Método **Looping()=True** indica al reproductor la reproducción debe ser en bucle. Apenas finalice la reproducción empiece otra vez.

2.3.2.5 Estado *Paused* (Reproducción en pausa)

Los métodos permitidos para este estado son:

- Método **start()** indica la MediaPlayer que comience la reproducción.
- Método **stop()** detiene el reproductor y permanece en el estado stopped.
- Método **seekto()** busca una posición de tiempo en milisegundos dentro del archivo multimedia seleccionado.

Se tiene los mismos métodos que el estado de preparado con la diferencia de que se almacena la posición en milisegundos de la última reproducción.

2.3.2.6 Estado *Stopped* (Reproducción Detenida)

Los métodos permitidos para este estado son:

- Método **prepare()** tiene que ser ejecutado obligatoriamente para cualquier reproducción de forma sincrónica; se utiliza en la mayoría de casos porque no genera alertas adicionales y simplemente indiaca al MediaPlayer que esta preparado para reproducir.
- Método **prepareAsync()** se ejecuta cuando el fichero multimedia que se desea reproducir es un streaming de datos no almacenados en el dispositivo. Genera la señal onPrepared() que indica que el MediaPlayer esta preparado.

2.3.2.7 Estado *PaybackCompleted* (Reproducción finalizada)

Indica al MediaPlayer que concluyo la reproducción del archivo multimedia y dependiendo de la configuración se reproducirá otra vez desde el principio o permanecerá detenido.

2.3.3 FORMATOS MULTIMEDIA SOPORTADOS EN ANDROID

La Tabla 2.4 describe los formatos multimedia incluidos en el sistema operativo Android. En los formatos que no se indica la versión, se considera que está disponible en todas las versiones.

Tipo	Códec	Codificador	Decodificador	Formatos
Audio	AAC LC/LTP	SI	SI	• 3GPP (.3gp)
	AAC+	NO	SI	• MPEG-4 (.mp4, .m4a)
	HE-AACv2 (Ampliado AAC+)	NO	Si	• MPEG-TS
	AMR-NB	Si	Si	3GPP (.3gp),(AMR)
	AMR-WB	Si	Si	3GPP (.3gp)
	MP3	NO	Si	MP3 (.mp3)
	MIDI	NO	SI	• Type 0 and 1 (.mid, .xmf, .mxmf)
				• RTTTL/RTX (.rtttl, .rtx)
				• OTA (.ota)
				• iMelody (.imy)
	Vorbis	NO	SI	• Ogg (.ogg)
				• Matroska (.mkv, Android 4.0+)
	PCM/WAVE	NO	SI	WAVE (.wav)
Imagen	JPEG	SI	SI	JPEG (.jpg)
	GIF	NO	SI	GIF (.gif)
	PNG	SI	SI	PNG (.png)
	BMP	NO	SI	BMP (.bmp)
	WEBP	SI (Android 4.0+)	SI (Android 4.0+)	WebP (.webp)
Video	H.263	SI	SI	• 3GPP (.3gp)
				• MPEG-4 (.mp4)
	H.264 AVC	SI	SI	• 3GPP (.3gp)
		SI(Android 3.0+)		• MPEG-4 (.mp4)
		NO		• MPEG-TS
	MPEG-4	NO	Si	(.3gp .mp4)

Tabla 2.4: Formatos Multimedia Soportados en Android [36**]**

2.3.4 PERMISOS PARA UTILIZAR MEDIAPLAYER Y MEDIARECORDER

Los permisos que debe adquirir la aplicación para utilizar el hardware son incluidos en el archivo de configuración AndroidManifest.xml del proyecto generado inicialmente y se utilizará los siguientes:

- Permiso para utilizar la cámara.

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Permiso para grabar audio.

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

- Permiso para grabar video.

```
<uses-permission android:name="android.permission.RECORD_VIDEO" />
```

- Permiso para grabar en la memoria externa.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

2.4 DESARROLLO DE LOS MÓDULOS Y MÉTODOS DE LA APLICACIÓN MULTIMEDIA

La aplicación permite la captura, almacenamiento, reproducción, administración y envío de archivos de video, audio e imágenes utilizando tecnología Bluetooth por lo tanto se ha dividido de la siguiente manera:

- Cuatro módulos principales que son:
  - **Módulo de Audio:** Permite la grabación, almacenamiento, reproducción, administración y envío de archivos de Audio.
  - **Módulo de Imágenes:** Permite la captura, almacenamiento, reproducción, administración y envío de archivos de imágenes.
  - **Módulo de Video:** Permite la grabación, almacenamiento, reproducción, administración y envío de archivos de video.
  - **Módulo del Menú Principal:** Permine la unión de los modulos de audio, imágenes y video en una interfaz gráfica única.

- Ocho métodos generales incluidos en los módulos principales que son:
  - **Método de la Comunicación Bluetooth:** Su función es enviar los archivos multimedia vía Bluetooth.

  - **Método Generador de Archivo y Carpeta de Almacenamiento:** Su función es crear la carpeta de almacenamiento y el nombre del archivo multimedia.

  - **Método Mensajes Emergentes:** Su función es desplegar mensajes rápidos e informativos en la pantalla del dispositivo.

  - **Método Eliminar Archivos:** Permite la eliminación del archivo seleccionado.

  - **Método Botones Presionados:** Detecta el botón que se ha presionado.

  - **Método Ruta de Acceso de Archivos:** Recupera el PATH absoluto de un archivo multimedia.

  - **Actividad por Resultados para video y audio:** Recupera la información generada por la respectiva actividad utilizando un provedor de contenidos.

  - **Método Barra de búsqueda (*SeekBar*):** Ubica al MediaPlayer en un determinada posición en la línea del tiempo

Se describe cada uno de los módulos y métodos incluidos en la aplicación multimedia, con su diagrama de flujo, objetivo, parámetros de entrada, y el código fuente implementado.

2.4.1 DESARROLLO DEL MÓDULO DE AUDIO

Se define el codificador de audio en ARM_NB porque permite captura en formato ARM y no en 3GP. El problema del formato 3gp es que la biblioteca de audio no lo reconoce y no es tomado en cuenta.

Se define el dispositivo de captura de audio en *DEFAULT* del sistema operativo Android porque los dispositivos móviles normalmente no poseen más de un micrófono.

Se determina la funcionalidad de este módulo con las siguientes características:

- Permitir la grabación de audio en formato ARM_NB.
- Registrar en la base de datos del sistema operativo los archivos generados.
- Almacenar los archivos generados en la memoria externa del dispositivo.
- Reproducir archivos de sonido almacenados en la memoria
- Reproducir los formatos mp3, arm, wav y midi.
- Permitir enviar vía Bluetooth archivos seleccionados previamente.
- Buscar una posición de tiempo en el audio que se está reproduciendo.

2.4.1.1 Diseño esquemático del módulo de audio

El diagrama de flujo de la Figura 2.22, permite observar la funcionalidad de cada botón de la interfaz gráfica y las variables globales del módulo descritas así:

- **Name:** Se almacena la ruta de acceso absoluto (PATH) en la variable “Name” de los archivos de auidio grabados o previamente almacenados en el dispositivo.
- **Recording:** Permite conocer el estado del MediaRecorder con una variable tipo *Boolean* (Verdadero o Falso) que indica si se encuentra grabando en un momento determinado de la ejecución de la aplicación.
- Se instancian los objetos **MediaRecorder** y **MediaPlayer** para su posterior uso.

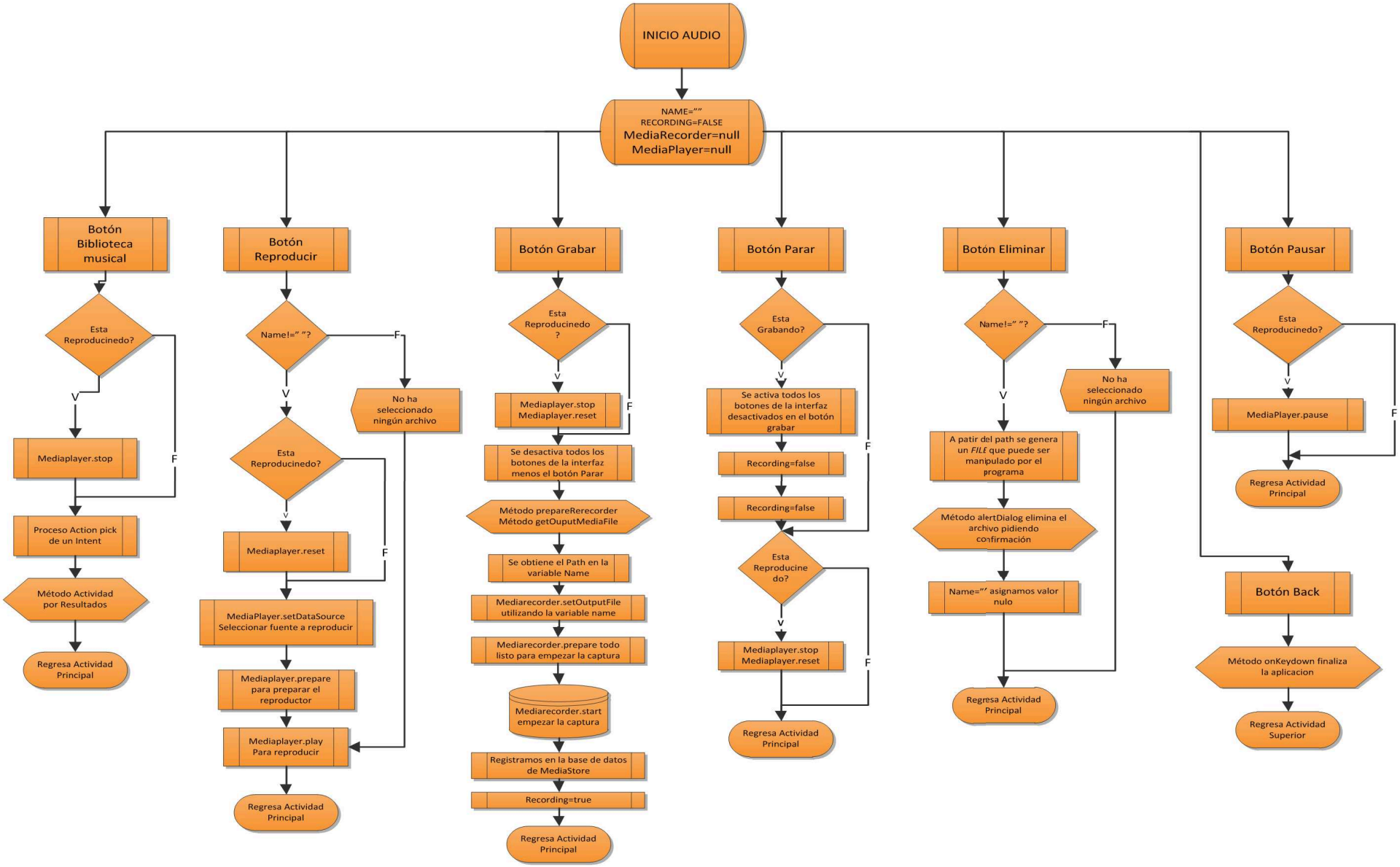


Figura 2.22: Módulo de Audio

2.4.1.2 Desarrollo de los métodos incluidos en el módulo de audio

A continuación se describen los métodos propios del módulo de audio que son:

- Biblioteca Musical.
- Reproducir.
- Grabar.
- Detener.
- Pausar.
- Eliminar.
- Preparar la grabadora de audio.

2.4.1.2.1 Biblioteca Musical

Permite obtener un archivo de audio almacenado previamente en el dispositivo.

- Los parámetros necesarios para este método son:
  - Establecer la biblioteca a ser utilizada por el proceso.
  - Establecer la localización de los archivos de audio internos o externos.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){
      mediaPlayer.stop();
}
Intent intent = new Intent(Intent.ACTION_PICK, android.provider.
MediaStore.Audio  //Se indica que se debe usar la biblioteca de Audio
.Media.EXTERNAL_CONTENT_URI); //Localización de archivos en la memoria externa
startActivityForResult(intent,1);//inicia  la  actividad  con  los  parámetros
                                 configurados
```

2.4.1.2.2 Reproducir

Configura correctamente el MediaPlayer de Android para reproducir el archivo de audio previamente seleccionado.

- Los parámetros necesarios para este método son:
  - ➢ Configurar la ruta de acceso del archivo (PATH) de audio en la variable “Name”.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){
      try { //Control de errores de java
            mediaPlayer.setDataSource(Name); //Se establece el PATH
            mediaPlayer.prepare();
```

```
            } catch (IllegalStateException e) {
            } catch (IOException e) {
      }
      if(mediaPlayer.isPlaying()){
            return;
      }
      mediaPlayer.start(); //Reproduce la canción seleccionada
}else{
      mensaje("No ha seleccionado ningún archivo de Sonido!");
}
```

2.4.1.2.3 Grabar

Configura correctamente el MediaRecorder de Android para grabar un archivo de audio.

- Los parámetros necesarios para este método son:
  - Definir el nombre del archivo mediante el método generador de archivo y carpeta
  - Definir la ruta de almacenamiento de los archivos
  - Se establece los parámetros de grabación; dispositivos de entrada, formato de salida, codificador del *stream*[46] de datos.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){ //Control de reproducción
      mediaPlayer.reset();
      Name="";
}
prepareRecorder();
File file=getOutputMediaFile();//Método generador de archivo y carpeta
Name=file.getAbsolutePath();
displayTxt(Name);
mediaRecorder.setOutputFile(Name); //Se establece el PATH para el MediaRecorder
try { //Control de errores de java
      mediaRecorder.prepare();
} catch (IllegalStateException e) {
} catch (IOException e) {
}
mediaRecorder.start(); //Empieza la grabación
// Registra el archivo creado en la base de datos multimedia
new MediaScannerConnectionClient() {
      private MediaScannerConnection msc = null; {
      msc = new MediaScannerConnection(getApplicationContext(), this);
      msc.connect();
}
public void onMediaScannerConnected() {
      msc.scanFile(Name, null);
}
public void onScanCompleted(String path, Uri uri) {
      msc.disconnect();
```

[46] Stream de datos: Flujo continuo de informacion

```
}
// Fin del registro
recording = true; //Variable global “recording” se establece como verdadero
                  para informar a todo el módulo que se esta grabando
```

2.4.1.2.4 Detener

Permite detener el MediaPlayer o el MediaRecorder si se encuentra grabando o reproduciendo regresando a valores por defecto (*Default*).

- Los parámetros necesarios para este método son:
  - Se evalúa la variable *recording* establecida en métodos anteriores para controlar el estado del MediaRecorder.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (recording) { // Se ejecuta si la variable recording es igual a verdadero
       recording = false;
       mediaRecorder.stop();
       mediaRecorder.reset();
       iv1.setVisibility(4); // Icono que se muestra en pantalla cuando se esta
                             grabando deshabilitado
       // Habilita los botones
       btnRec.setEnabled(true);
       btnStop.setEnabled(true);
       btnPlay.setEnabled(true);
       btnSend.setEnabled(true);
       btnPause.setEnabled(true);
       btnDel.setEnabled(true);
       btnGallery.setEnabled(true);
       // Fin Habilitar
} else if (mediaPlayer.isPlaying()) {
       mediaPlayer.stop();
       mediaPlayer.reset();
}
```

2.4.1.2.5 Pausar

Permite congelar el reproductor sin cambiar los valores establecidos.

- Los parámetros necesarios para este método son:
  - Evalúa si se encuentra reproduciendo el MediaPlayer.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){ //Solo pausara la reproducción si se encuentra
                              activa.
       mediaPlayer.pause();
}
else{
       mensaje("No se encuentra Reproduciendo ningún elemento");

}
```

2.4.1.2.6 Eliminar

Permite eliminar un archivo previamente seleccionado.

- Los parámetros necesarios para este método son:
  - Se indica el PATH del archivo que se desea eliminar en la variable "Name".
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){ //Ejecuta el método de alerta
	alerta();
}
else{
	mensaje("No ha seleccionado ningún archivo de Sonido!");
}
```

2.4.1.2.7 Preparar la grabadora de audio

Objetivo: Establecer los valores de inicio para capturar audio

- Los parámetros necesarios para este método son:
  - Establecer el micrófono DEFAULT como dispositivo de entrada.
  - Establecer el codificador de audio en ARM_NB
  - Indicar el formado de salida del codificador .arm
- El funcionamiento se determina a través del código que ejecuta el método.

```
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

2.4.2 DESARROLLO DEL MÓDULO DE IMÁGENES

Se define el codificador de imágenes en JPEG porque es un formato de buena calidad y ocupa poco espacio de almacenamiento en el dispositivo.

Se determina la funcionalidad de este módulo con las siguientes características:

- Permitir la captura de imágenes en formato JPEG.
- Almacenar los archivos generados en la memoria externa del dispositivo.
- Presentar las imágenes capturadas en la interfaz principal.
- Permitir mostrar archivos almacenados en el dispositivo.
- Permitir enviar vía Bluetooth archivos seleccionados previamente.

2.4.2.1 Diseño esquemático del módulo de imágenes

El diagrama de flujo de la Figura 2.23, permite observar la funcionalidad de cada botón de la interfaz gráfica y las variables globales del módulo descritas así:

- Utiliza la variable “Name” para almacenar el PATH de la imagen manipulada.

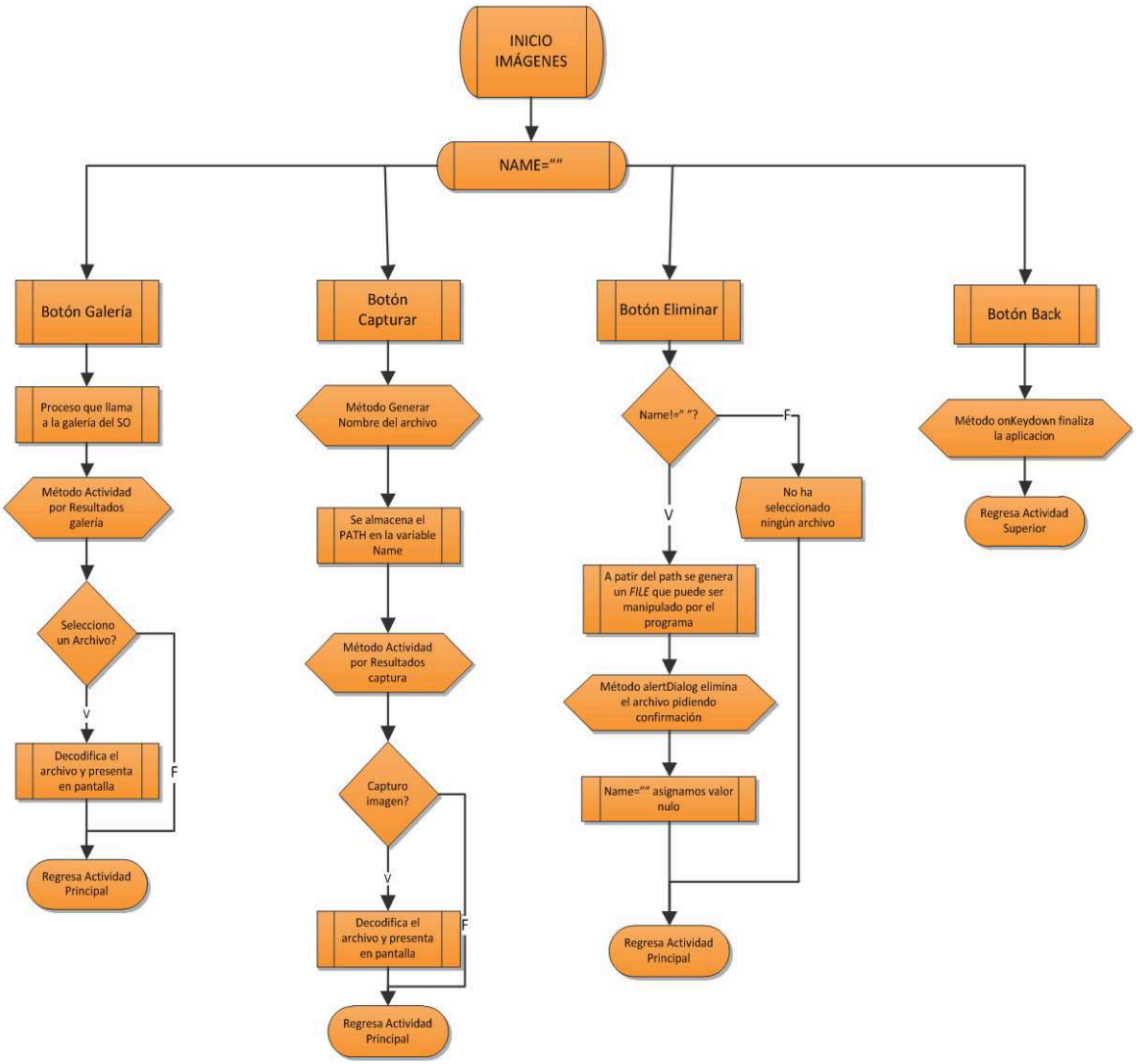


Figura 2.23: Módulo de Imágenes

2.4.2.2 Desarrollo de los métodos incluidos en el módulo de imágenes

A continuación se describen los métodos propios del módulo de imágenes que son:

- Galería de imágenes.

- Capturar Imagen.
- Eliminar.
- Actividad por resultados para imágenes.

2.4.2.2.1 Galería de Imágenes

Permite obtener y mostrar una imagen previamente almacenada en el dispositivo.

- Los parámetros necesarios para este método son:
  - Establecer la biblioteca a ser utilizada por el proceso.
  - Establecer el filtro de tipo de archivos.
- El funcionamiento se determina a través del código que ejecuta el método.

```
Intent intent = new Intent(Intent.ACTION_PICK, //Acción inherente a seleccionar
                                               un elemento.
      android.provider.
      MediaStore.Images.Media.EXTERNAL_CONTENT_URI);   //   Seleccionamos   las
                                               imágenes   almacenadas   en   la
                                               memoria externa.
      intent.setType("image/*"); //Se establece que la galería solo muestre
                                 imágenes
startActivityForResult(intent,1); //Ejecuta el método actividad por resultados
                                   con el código 1.
```

2.4.2.2.2 Capturar Imagen

Objetivo: Obtener desde la cámara del dispositivo una imagen y almacenarla.

- Los parámetros necesarios para este método son:
  - Establece el nombre del archivo
  - Indicar la ruta de almacenamiento
- El funcionamiento se determina a través del código que ejecuta el método.

```
File file=getOutputMediaFile(); //Crea el archivo de imagen.
Name=file.getAbsolutePath(); // Extrae el PATH de archivo.
displayTxt(Name); // Muestra en pantalla el nombre del archivo
Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);//Solicita al sistema
                                 operativo capturar una imagen.
Uri output = Uri.fromFile(file);
intent.putExtra(MediaStore.EXTRA_OUTPUT, output);
startActivityForResult(intent,2); //Ejecuta el método actividad por resultados
                                   con el código 2.
```

2.4.2.2.3 Eliminar

Permite eliminar un archivo previamente seleccionado.

- Los parámetros necesarios para este método son:

- Se indica el PATH del archivo que se desea eliminar en la variable “Name”.

- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){
	alerta(); //Ejecuta el método de borrado si ha seleccionado un archivo.
}
else{
	mensaje("No ha seleccionado ninguna Imagen!");
}
```

2.4.2.2.4 Actividad por Resultados para imágenes

Permite capturar los datos generados por otras actividades en el momento que finalizan su ejecución.

- Los parámetros necesarios para este método son:
  - Código requerido en la variable “RequestCode” auto generada por el método Actividad por resultados.
  - Datos generados en la variable “data” y “resultCode”.
- El funcionamiento se determina a través del código que ejecuta el método.

RequestCode=1, Indica que la imagen fue obtenida desde la galería.

RequestCode=2, Indica que la imagen fue obtenida desde la cámara del dispositivo.

```
if (requestCode ==1 && data!=null  ){ //Verifica el código requestCode=1.
	Uri fileU = data.getData();//Extrae los datos almacenados en “data”.
	Name =getRealPathFromURI(fileU);//Obtiene el PATH absoluto del archivo.
	final ImageView imgV =(ImageView)findViewById(R.id.imgV);
	imgV.destroyDrawingCache(); //Libera espacio en la memoria cache
	imgV.setImageDrawable(null); //Libera la superficie.
	imgV.setImageBitmap(BitmapFactory.decodeFile(Name));//Muestra la imagen.
	displayTxt(Name); //Muestra el nombre de la imagen seleccionada.
}
if(requestCode ==2&& && resultCode==RESULT_OK){
	final ImageView imgV =(ImageView)findViewById(R.id.imgV);
	imgV.destroyDrawingCache();//Libera espacio en la memoria cache
	imgV.setImageDrawable(null); //Libera la superficie.
	imgV.setImageBitmap(BitmapFactory.decodeFile(Name)); //Muestra la imagen.
	// Registra el archivo generado en la biblioteca de imágenes
	new MediaScannerConnectionClient() {
		private MediaScannerConnection msc = null; {
		msc = new MediaScannerConnection(getApplicationContext(), this);
		msc.connect();
	}
	public void onMediaScannerConnected() {
		msc.scanFile(Name, null);
```

```
        }
        public void onScanCompleted(String path, Uri uri) {
                msc.disconnect();
        }
}
```

Se muestra el diagrama de flujo de actividad por resultados en la Figura 2.24.

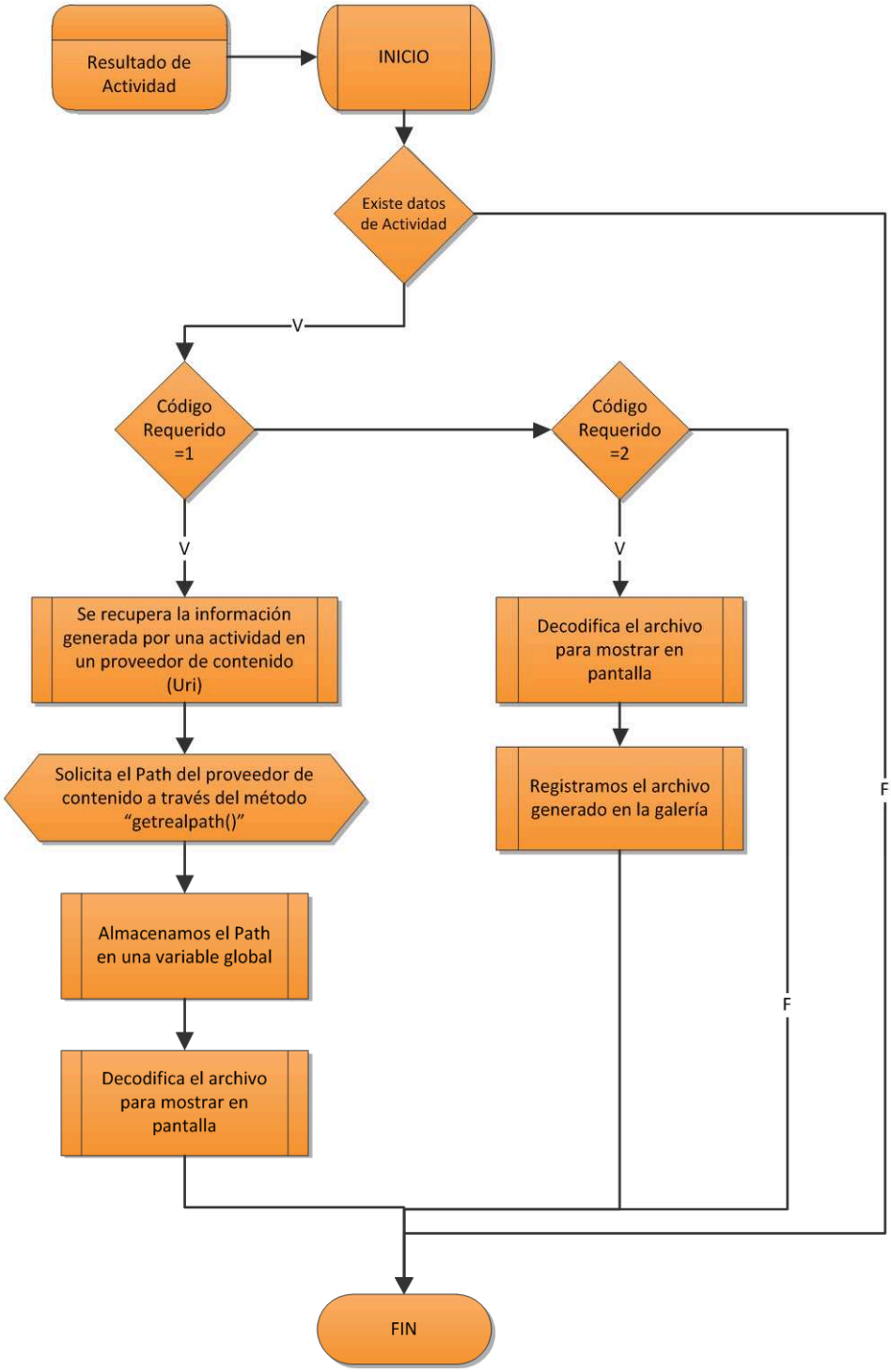


Figura 2.24: Actividad por resultados Imágenes

2.4.3 DESARROLLO DEL MÓDULO DE VIDEO

Se define el codificador de video MPEG_SP, porque asegura compatibilidad con la mayoría de dispositivos móviles. Es una adaptación del estándar MPEG4 que reduce la tasa de bits reduciendo el tamaño en almacenamiento.

Se define la cámara principal y el micrófono principal del dispositivo con la opción “DEFAULT” del MediaRecorder.

Se determina la funcionalidad de este módulo con las siguientes características:

- Permitir la grabación de video en formato MPEG4
- Registrar en la base de datos del sistema operativo los archivos de video generados por la actividad de captura.
- Almacenar los archivos generados en la memoria SD externa del dispositivo.
- Reproducir archivos de video almacenados en la memoria
- Buscar una posición de tiempo en el video que se está reproduciendo
- Presentar el video en pantalla completa según desee el usuario.
- Permitir enviar vía Bluetooth archivos seleccionados previamente.
- Permite seleccionar la calidad de captura del video al usuario, según sus propósitos; en normal y alta calidad (La máxima soportada por el dispositivo)

2.4.3.1 Diseño esquemático del módulo de Video

El diagrama de flujo de la Figura 2.25, permite observar la funcionalidad de cada botón de la interfaz gráfica y las variables globales del módulo descritas así:

- **Name:** Se almacena la ruta de acceso (PATH) de los archivos creados o previamente almacenados.
- **Recording:** Permite conocer el estado del MediaRecorder con una variable tipo *Boolean* (Verdadero o Falso).
- Se instancian los objetos **MediaRecorder** y **MediaPlayer** para su posterior uso.
- Se implementa un surfaceHolder() para soportar la reproducción de video.

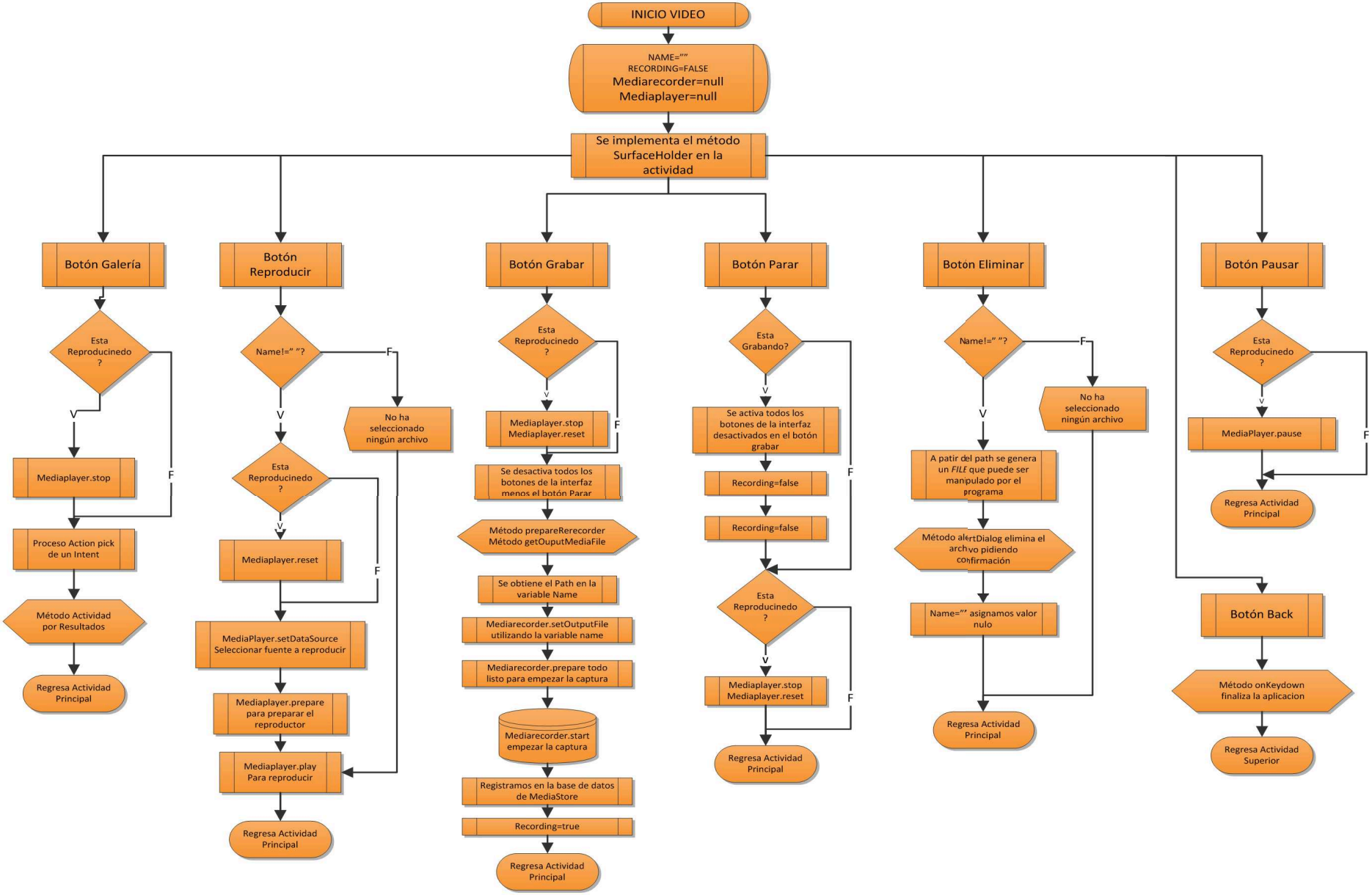


Figura 2.25: Módulo de Video

2.4.3.2 Desarrollo de los métodos incluidos en el módulo de Video

A continuación se describen todos los métodos propios del módulo de video que son:

- Galería de videos.
- Reproducir.
- Grabar.
- Detener.
- Pausar.
- Eliminar.
- Preparar la grabadora de video.
- Método pantalla completa.

2.4.3.2.1 Galería de videos

Permite obtener un archivo de video almacenado previamente en el dispositivo

- Los parámetros necesarios para este método son:
  - Establecer la galería a ser utilizada por el proceso.
  - Establecer el filtro de tipo de archivos.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){ // Evalúa el estado actual del MediaPlayer.
      mediaPlayer.stop();
}
Intent intent = new Intent(Intent.ACTION_PICK, //Acción inherente a seleccionar
                                                un elemento
      android.provider.MediaStore.Video.
      Media.EXTERNAL_CONTENT_URI);
intent.setType("video/*");//Se establece que la galería solo muestre videos
startActivityForResult(intent,1);
```

2.4.3.2.2 Reproducir

Configura correctamente el MediaPlayer de Android para reproducir el archivo de video previamente seleccionado.

- Los parámetros necesarios para este método son:
  - Configurar la ruta de acceso del archivo (PATH) de video en la variable “Name”.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){
	try { //Control de errores de java
		mediaPlayer.setDataSource(Name); //Se establece el PATH
		mediaPlayer.prepare();
		} catch (IllegalStateException e) {
		} catch (IOException e) {
	}
	if(mediaPlayer.isPlaying()){
		return;
	}
	mediaPlayer.start(); //Reproduce la canción seleccionada
}else{
	mensaje("No ha seleccionado ningún archivo de Video!");
}
```

2.4.3.2.3 Grabar

Configura correctamente el MediaRecorder de Android para grabar un archivo de video.

- Los parámetros necesarios para este método son:
  - Definir el nombre del archivo mediante el método generador de archivo y carpeta
  - Definir la ruta de almacenamiento de los archivos
  - Se establece los parámetros de grabación; dispositivos de entrada, formato de salida, codificador del *stream*[47] de datos.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){ //Control de reproducción
	mediaPlayer.reset();
	Name="";
}
prepareRecorder();
File file=getOutputMediaFile();//Método generador de archivo y carpeta
Name=file.getAbsolutePath();
displayTxt(Name);
mediaRecorder.setOutputFile(Name); //Se establece el PATH para el MediaRecorder
try {
	mediaRecorder.prepare();
} catch (IllegalStateException e) {
} catch (IOException e) {
}
mediaRecorder.start(); //Empieza la grabación
// Registra el archivo creado en la base de datos multimedia
new MediaScannerConnectionClient() {
	private MediaScannerConnection msc = null; {
	msc = new MediaScannerConnection(getApplicationContext(), this);
	msc.connect();
}
public void onMediaScannerConnected() {
```

[47] Stream de datos: Flujo continuo de informacion

```
        msc.scanFile(Name, null);
}
public void onScanCompleted(String path, Uri uri) {
        msc.disconnect();
}
// Fin del registro
recording = true; //Variable global “recording” se establece como verdadero
                  para informar a todo el módulo que se esta grabando
```

2.4.3.2.4 Detener

Permite detener el MediaPlayer o el MediaRecorder si se encuentra grabando o reproduciendo regresando a valores por defecto (*Default*).

- Los parámetros necesarios para este método son:
  - Se evalúa la variable *recording* establecida en métodos anteriores para controlar el estado del MediaRecorder.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (recording) { // Se ejecuta si la variable “recording” es igual a verdadero
        recording = false;
        mediaRecorder.stop();
        mediaRecorder.reset();
        iv1.setVisibility(4); // Icono que se muestra en pantalla cuando se esta
                              grabando deshabilitado
        // Habilita los botones
        btnRec.setEnabled(true);
        btnStop.setEnabled(true);
        btnPlay.setEnabled(true);
        btnSend.setEnabled(true);
        btnPause.setEnabled(true);
        btnDel.setEnabled(true);
        btnGallery.setEnabled(true);
        // Fin Habilitar
} else if (mediaPlayer.isPlaying()) {
        mediaPlayer.stop();
        mediaPlayer.reset();
}
```

2.4.3.2.5 Pausar

Permite congelar el reproductor sin cambiar los valores establecidos.

- Los parámetros necesarios para este método son:
  - Evalúa si se encuentra reproduciendo el MediaPlayer.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if(mediaPlayer.isPlaying()){ //Solo pausara la reproducción si se encuentra
                              activa.
        mediaPlayer.pause();
}
```

```
else{
	mensaje("No se encuentra Reproduciendo ningún elemento");

}
```

2.4.3.2.6 Eliminar

Permite eliminar un archivo previamente seleccionado

- Los parámetros necesarios para este método son:
  - Se indica el PATH del archivo que se desea eliminar en la variable "Name".
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){ //Ejecuta el método de alerta
	alerta();
}
else{
	mensaje("No ha seleccionado ningún archivo de video!");
}
```

2.4.3.2.7 Preparar la grabadora de video

Establece las configuraciones para capturar video según los requerimientos establecidos para este módulo.

- Los parámetros necesarios para este método son:
  - Establecer el micrófono como dispositivo de entrada.
  - Establecer la cámara como dispositivo de entrada.
  - Establecer el codificador de audio
  - Establecer el codificador de video
  - Indicar el formado de salida del codificador de video
- El funcionamiento se determina a través del código que ejecuta el método.

```
//Se configuran los parámetros establecidos para este módulo.
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
mediaRecorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
CheckBox chkHd=(CheckBox)findViewById(R.id.chkHd);
if (chkHd.isChecked()) {
	mediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
}else{
	mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
	mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
	mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);
```

2.4.3.2.8 Método pantalla completa

Objetivo: Reproducir el video en curso en pantalla completa

- Los parámetros necesarios para este método son:
  - Nombre del video en curso.
  - Posición del MediaPlayer.
- El funcionamiento se determina a través del código que ejecuta el método.

```
if (Name!=""){ // Evalúa si ha seleccionado un archivo de video.
      Intent intent = newIntent(VideoActivity.this,fullScreen.class);//
Configura el proceso que llamara a la clase "fullScreen"
      Bundle bundle = new Bundle(); // Se crea un contenedor independiente de
                                       la clase.
      int progress=seekBar.getProgress();//Envía el tiempo de progreso del
                                          video actual.
      bundle.putString("Archivo", Name);//Envía el PATH del archivo en
                                          reproducción
      bundle.putInt("Progress",progress);
      intent.putExtras(bundle);
      startActivity(intent);
}
```

Se envía los datos a la clase pantalla completa a través de un Bundle[48]

2.4.4 DESARROLLO DEL MÓDULO DEL MENÚ PRINCIPAL

Este módulo permite la unificación de todos métodos, módulos, comandos y permisos ya descritos, en una sola interfaz.

Además se debe incluir los permisos en AndroidManifest.xml de cada módulo a ser incluido en la actividad principal de la siguiente manera:

Para el módulo de Video se incluye los siguientes permisos:

```
<activity android:name=".VideoActivity"
android:screenOrientation="landscape"></activity>
<activity android:name=".fullScreen"
android:screenOrientation="landscape"></activity>
```

Para el módulo de Audio se incluye el siguiente permiso

```
<activity android:name=".AudioActivity"
android:screenOrientation="landscape"></activity>
```

[48] Bundle: Contenedor de datos creado por el programador para almacenar variables y utilizarlas en diferentes Clases

Para el módulo de Imágenes se incluye el siguiente permiso

```
<activity android:name=".ImgActivity"
android:screenOrientation="landscape"></activity>
```

2.4.4.1 Diseño esquemático del menú principal

El diagrama de flujo de la Figura 2.26, permite observar la funcionalidad de cada botón de la interfaz gráfica y muestra el funcionamiento completo de la aplicación multimedia.

Se unifica los tres módulos desarrollados por separados logrando un interfaz principal, que permite el acceso a los métodos particulares de cada uno de la siguiente manera:

- Audio
  - Grabar.
  - Reproducir.
  - Eliminar.
  - Envío vía Bluetooth.
- Imágenes
  - Capturar
  - Importar
  - Eliminar
  - Envío vía Bluetooth.
- Video
  - Grabar.
  - Reproducir.
  - Eliminar.
  - Envío vía Bluetooth.

Su objetivo es facilitar el acceso a los diferentes módulos de la aplicación.

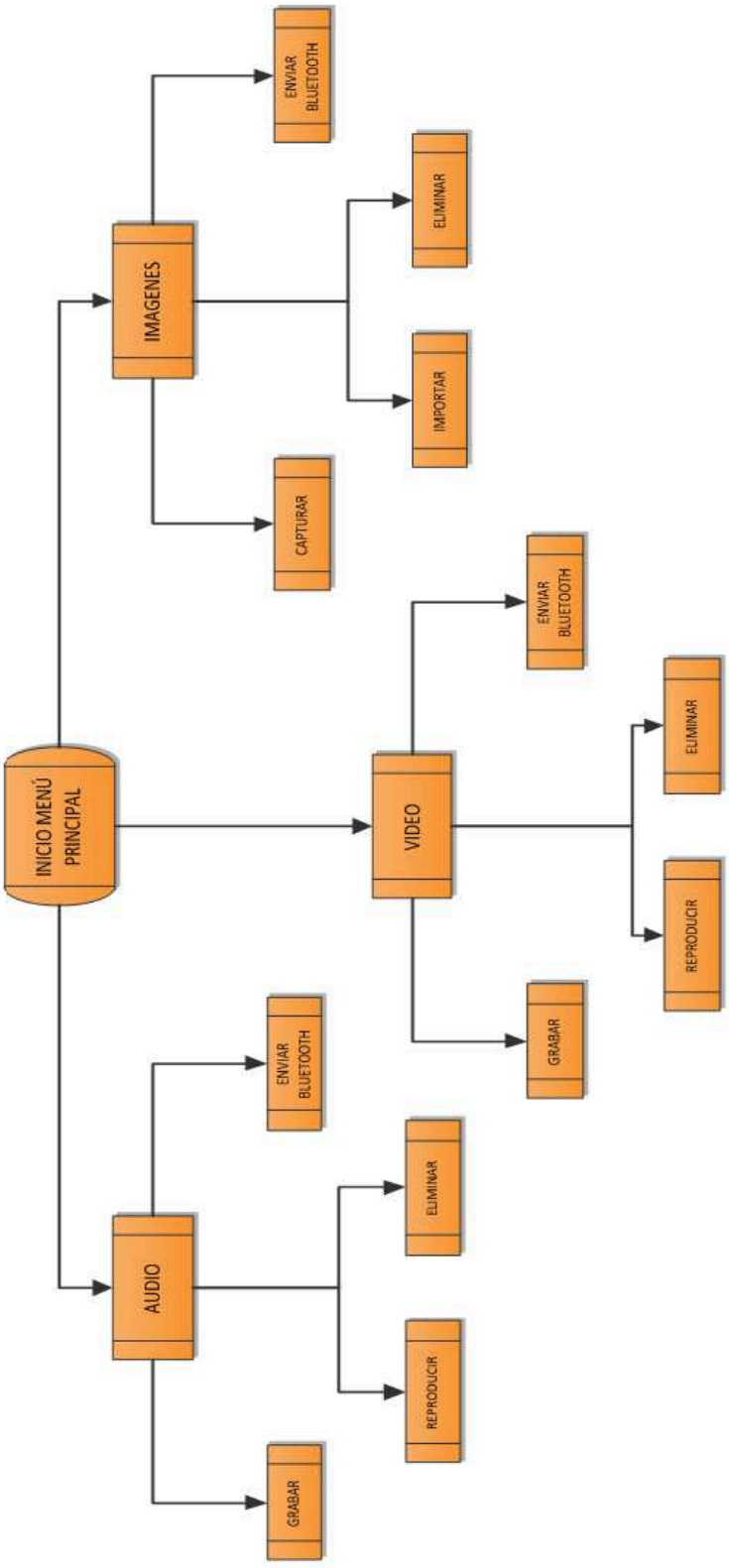


Figura 2.26: Diagrama de Flujo del Menú Principal

2.4.4.2 Código implementado en el Menú Principal

El funcionamiento se determina a través del código que ejecuta el método.

```
// Botón Imágenes
        btnImg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
                Intent intent = new Intent(DroidMediaActivity.this,
                ImgActivity.class);//Configura la actividad que llamara a la clase
                                   de Imágenes.
                startActivity(intent);
                }
        });
// Botón Video
        btnVideo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
                Intent intent = new Intent(DroidMediaActivity.this,
                VideoActivity.class); //Configura la actividad que llamara a la
                                      clase de Video.
                startActivity(intent);
                }
        });
// Botón Audio
       btnAudio.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
                Intent intent = new Intent(DroidMediaActivity.this,
                AudioActivity.class); //Configura la actividad que llamara a la
                                      clase de Audio.
                startActivity(intent);
                }
        });
// Método de mensajes en pantalla
        public void mensaje(CharSequence text){
                Context context = getApplicationContext();
                int duration = Toast.LENGTH_LONG;
                Toast toast = Toast.makeText(context, text, duration);
                toast.setGravity(0, 0, 150);
                toast.show();
        }
// Método teclas presionadas solo incluye Keyback

        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
                if(keyCode==KeyEvent.KEYCODE_BACK)
                {
                this.finish(); //Ejecuta el comando de finalización de actividades.
                Android.os.Process.killProcess(android.os.Process.myPid());//Termin
                a cualquier proceso adicional iniciado durante la ejecución del
                programa
                }
        return true;
```

2.4.5 MÉTODOS GENERALES PARA LOS MÓDULOS DE AUDIO, VIDEO E IMÁGENES

Los métodos descritos a continuación son incluidos en los tres módulos principales de Audio, Video e Imágenes. Complementan el código principal de cada módulo, para agregar funcionalidad, cumpliendo los objetivos plateados en el proyecto de titulación

Cada uno tiene una función específica y son los siguientes:

- **Método de la Comunicación Bluetooth:** Su función es enviar los archivos multimedia vía Bluetooth.
- **Método Generador de Archivo y Carpeta de Almacenamiento:** Su función es crear la carpeta de almacenamiento y el nombre del archivo multimedia.
- **Método Mensajes Emergentes:** Su función es desplegar mensajes rápidos e informativos en la pantalla del dispositivo.
- **Método Eliminar Archivos:** Permite la eliminación del archivo seleccionado.
- **Método Botones Presionados:** Detecta el botón que se ha presionado.
- **Método Ruta de Acceso de Archivos:** Recupera el PATH absoluto de un archivo multimedia.
- **Actividad por Resultados para video y audio:** Recupera la información generada por la respectiva actividad utilizando un provedor de contenidos.
- **Método Barra de búsqueda (*SeekBar*):** Ubica al MediaPlayer en un determinada posición en la línea del tiempo

Los diagramas de flujo incluidos en cada uno de los métodos, representan gráficamente el algoritmo implementado, permitiendo comprender rápidamente el código fuente.

2.4.5.1 Método de la Comunicación Bluetooth

Permite enviar archivos multimedia vía Bluetooth.

- Se determina la funcionalidad de este método con las siguientes características:
  - Envió de archivos multimedia audio, video e imágenes utilizando tecnología Bluetooth.
  - Permitir el envío a cualquier dispositivo que incluya Bluetooth.
  - No se requiere una versión específica de Bluetooth para su funcionamiento.

El diagrama de flujo del método se observa en la Figura 2.27

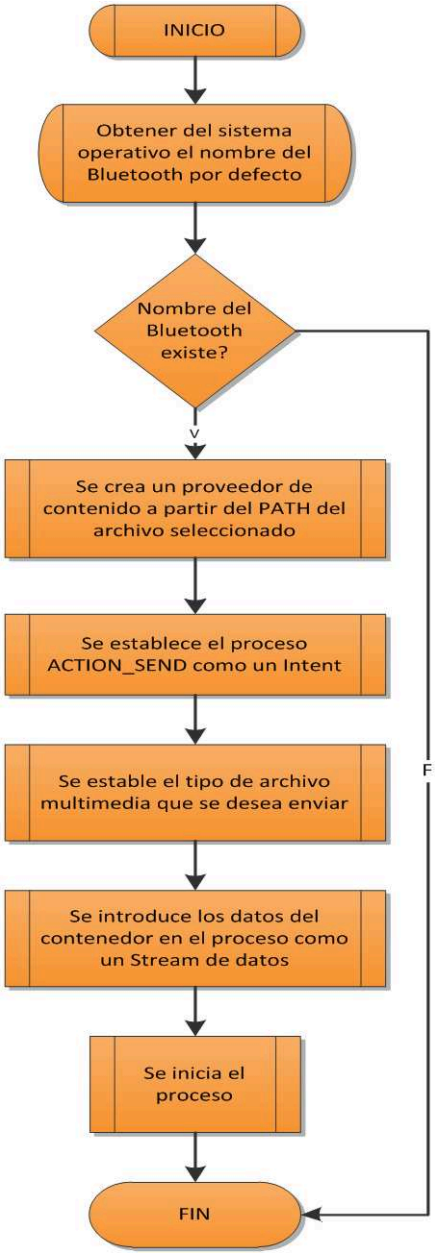


Figura 2.27: Método Comunicación Bluetooth

- Los parámetros necesarios para este método son:
  - Configurar la ruta de acceso del archivo multimedia (PATH) en la variable “Name”.
  - Seleccionar el tipo de archivo multimedia y el formato se define en cualquiera con “*” asterisco.
    - ✓ Para Audio (“audio/*”)
    - ✓ Para Imágenes (“image/*”)
    - ✓ Para Videos (“video/*”)
- El funcionamiento se determina a través del código que ejecuta el método.

```
btnSend.setOnClickListener(new View.OnClickListener() {
	@Override
	public void onClick(View arg0) {
	mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter(); // Adquiere el
                                    nombre del Bluetooth del dispositivo
	if (mBluetoothAdapter != null) { //Comprueba la existencia de Bluetooth
		if (Name!=""){ //Comprueba que haya seleccionado un archivo
		Uri fileUri = Uri.fromFile(new File(Name));// Transforma el PATH en
                                        un URI para el stream de datos
		Intent intent1 = new Intent();
		intent1.setAction(Intent.ACTION_SEND); //Ejecuta el proceso de
                                                envío el proceso
		intent1.setType("*/*"); // Se define el tipo de archivo multimedia
                                    que se desea enviar
		intent1.putExtra(Intent.EXTRA_STREAM,fileUri);//Se configura los
                                                        datos de entrada
		startActivity(intent1); //Procede al envío
		}else {
			mensaje("No ha seleccionado ningún archivo!");	}
	}else{
		mensaje("Bluetooth no disponible");
		}
	}
});
```

Adicionalmente el método aquí descrito, permite el envío de los archivos multimedia, al correo electrónico o publicarlos en redes sociales.

2.4.5.2 Método Generador de Archivo y Carpeta de Almacenamiento

El objetivo es crear la carpeta de destino para almacenar los archivos multimedia en la memoria extraíble e indicar el nombre del archivo.

Se representa el funcionamiento del método generador de archivo y carpeta mediante el diagrama de flujo en la Figura 2.28.

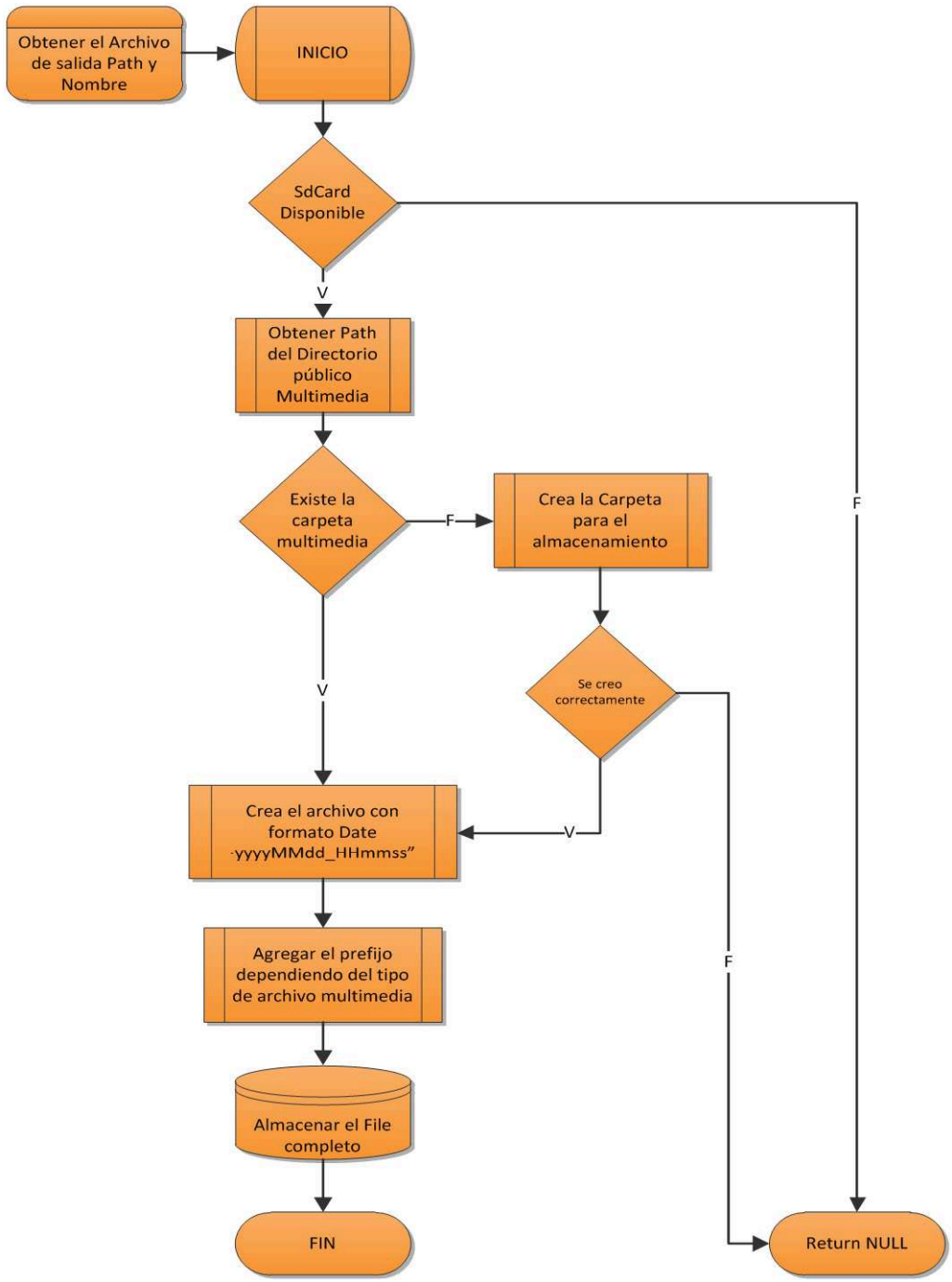


Figura 2.28: Método Generador de Archivo y Carpeta

- Los parámetros necesarios para este método son:
  - Indicar tipo de archivo Imagen, video o audio.
  - Formatos de salida .arm, .jpg y .mp4.
  - Establecer la extensión del archivo correspondiente a su contenido.
  - Define el nombre utilizando la fecha y la hora establecida en el sistema operativo.

- El funcionamiento se determina a través del código que ejecuta el método.

```
public File getOutputMediaFile(){ //Nombre del método.
String state = Environment.getExternalStorageState(); //Almacena el estado de
                                                      la memoria externa
if (Environment.MEDIA_MOUNTED.equals(state)) {
       File mediaStorageDir = new
       File(Environment.getExternalStoragePublicDirectory(
       Environment.DIRECTORY),"DroidMedia"); // Crea la capeta correspondiente
       al módulo en que se encuentra ejecutado el método.
       if (! mediaStorageDir.exists()){
              if (! mediaStorageDir.mkdirs()){
              Log.d("DroidMediaImagenes", "No se pudo crear el directorio");
             return null;
       }
}
// Genera el nombre del archivo multimedia
String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
File mediaFile; //Obtiene la fecha y la hora del sistema operativo.
mediaFile = new File(mediaStorageDir.getPath() + File.separator +
"I"+ timeStamp + ".JPG"); //Crea el nombre del archivo como ejemplo el de
                           imágenes.
return mediaFile;
}
mensaje("Inserte SD para grabar!");//En caso de no disponer memoria SD mostrara
                                   el siguiente mensaje.
return null;
}
```

El método crea las carpetas de la siguiente manera:

- Para Audio se amacena en DroidMediaAudio.
- Para Video se almacena en DroidMediaVideo
- Para Imágenes se almacena en DroidMediaImagenes.

El método crea el nombre de los archivos con la siguiente estructura:

- Para Audio A"yyyyMMdd_HHmmss".arm
- Para Video V"yyyyMMdd_HHmmss".mp4
- Para Imágenes I"yyyyMMdd_HHmmss".jpg

2.4.5.3 Método Mensajes Emergentes

Permite desplegar en pantalla mensajes rápidos e importantes para el usuario.

Se representa el funcionamiento de mensajes emergentes mediante el diagrama de flujo de la Figura 2.15.

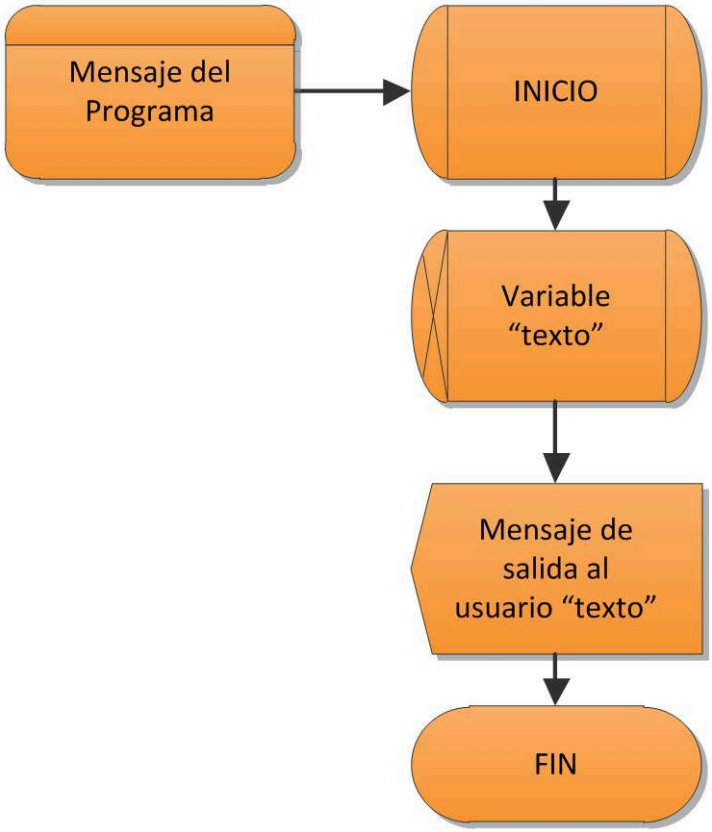


Figura 2.29: Método Mensajes Emergentes

- Los parámetros necesarios para este método son:
  - Se ingresa el texto que será mostrado en pantalla.
- El funcionamiento se determina a través del código que ejecuta el método.

```
public void mensaje(CharSequence text){ //Nombre del método.
      Context context = getApplicationContext();
      int duration = Toast.LENGTH_SHORT;//Permanece un tiempo corto en
                                        pantalla.
      Toast toast = Toast.makeText(context, text, duration);//Configura el
      método con el texto de entrada almacenado en la variable text.
      toast.show();
}
```

2.4.5.4 Método Eliminar Archivos

Permite eliminar un archivo previamente seleccionado utilizando un mensaje de confirmación.

Se representa el funcionamiento del método eliminación mediante el diagrama de flujo de la Figura 2.30.

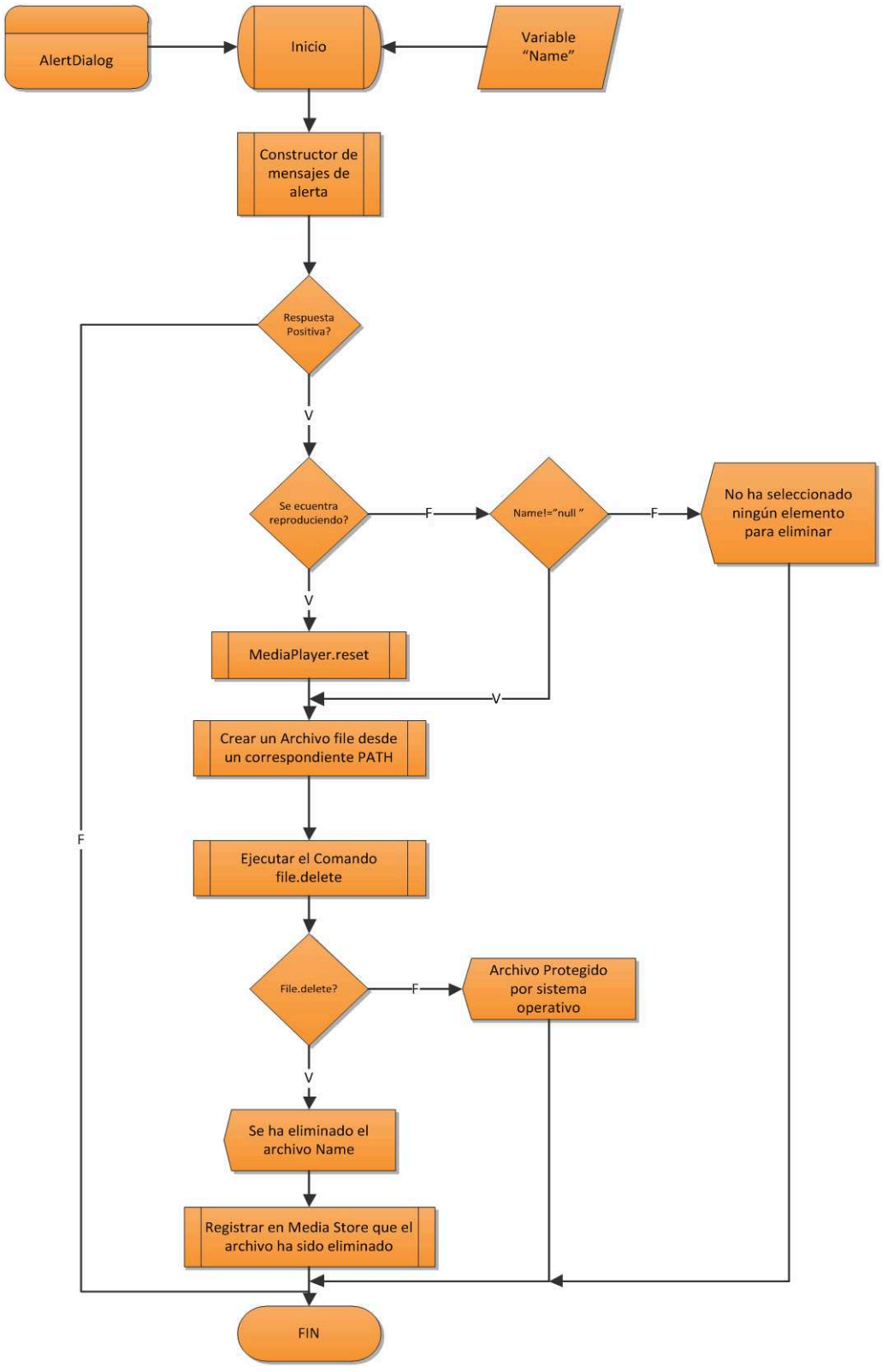


Figura 2.30: Método Eliminar Archivos

- Los parámetros necesarios para este método son:
  - Se indica el PATH del archivo a ser eliminado en la variable “Name”.
- El funcionamiento se determina a través del código que ejecuta el método.

Primero construye el mensaje de alerta, luego elimina si recibe una confirmación positiva.

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Esta seguro que desea borrar este archivo?")
.setCancelable(false)
.setPositiveButton("Si", new DialogInterface.OnClickListener(){//Respuesta
                                                                positiva.
      public void onClick(DialogInterface dialog, int id) {
      if(mediaPlayer.isPlaying()){
            mediaPlayer.reset(); //Reinicia el MediaPlayer si esta
                                 reproduciendo.
      }
      File file=new File(Name);
      boolean deleted = file.delete(); //Comando que elimina el archivo.
      if (deleted){
            mensaje("Archivo Eliminado"+Name); //Muestra un mensaje confirmado
                                               el archivo que ha sido eliminado.
            //Elimina el registro de la biblioteca multimedia.
            sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED,
            Uri.parse(Environment.getExternalStoragePublicDirectory());
            Name="";
            displayTxt(Name);
      }else { //En caso de que el archivo este protegido muestra el mensaje
            mensaje("Archivo protegido por el sistema");
      }
      })
//Repuesta negativa.
.setNegativeButton("No", new DialogInterface.OnClickListener(){
      public void onClick(DialogInterface dialog, int id) {
      dialog.cancel(); //Cancela la ventana de dialogo.
      }
});
AlertDialog alert = builder.create();
alert.show();
```

2.4.5.5 Método Botones Presionados (*Button Listener*)

Permite identificar el botón que ha sido presionado en el dispositivo y que no pertenece a la interfaz gráfica de la aplicación desarrollada.

Se representa el funcionamiento del método mediante el diagrama de flujo de la Figura 2.31

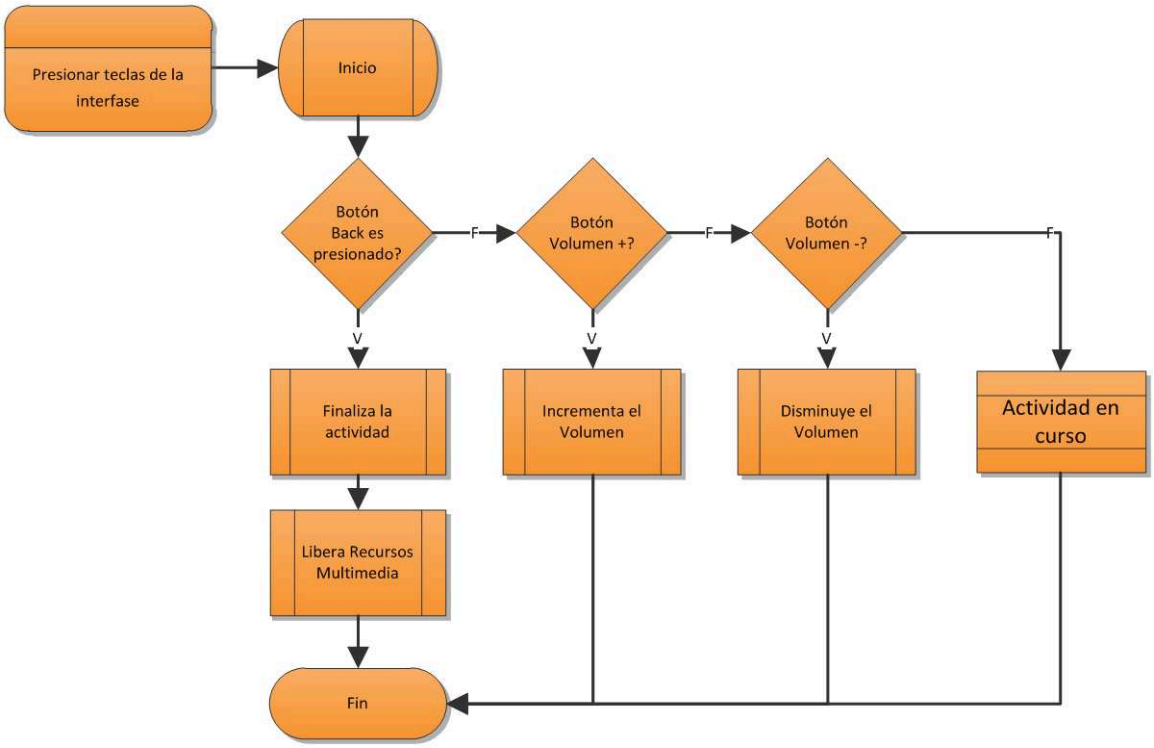


Figura 2.31: Método botones presionados

- Los parámetros necesarios para este método son:
  - Botones presionados (*keyEvent*)
- El funcionamiento se determina a través del código que ejecuta el método.

```
switch (keyCode) {
case KeyEvent.KEYCODE_VOLUME_UP: //Incrementa el volumen.
       audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
       AudioManager.ADJUST_RAISE, AudioManager.FLAG_SHOW_UI);
       return true;
case KeyEvent.KEYCODE_VOLUME_DOWN: //Disminuye el volumen.
       audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
       AudioManager.ADJUST_LOWER, AudioManager.FLAG_SHOW_UI);
       return true;
case KeyEvent.KEYCODE_BACK: //Finaliza todos los procesos y regresa a la
                            actividad anterior.
       Thread.setDefaultUncaughtExceptionHandler(null);
       this.finish();
       mediaPlayer.release();
       mediaRecorder.release();
       return true;
default:
return false;
}
```

2.4.5.6 Método Ruta de Acceso de archivos (PATH)

El objetivo es resolver la ruta absoluta del PATH relativo, que proporciona la librería Android.MediaStore.

Se representa el funcionamiento mediante el diagrama de flujo de la Figura 2.32.

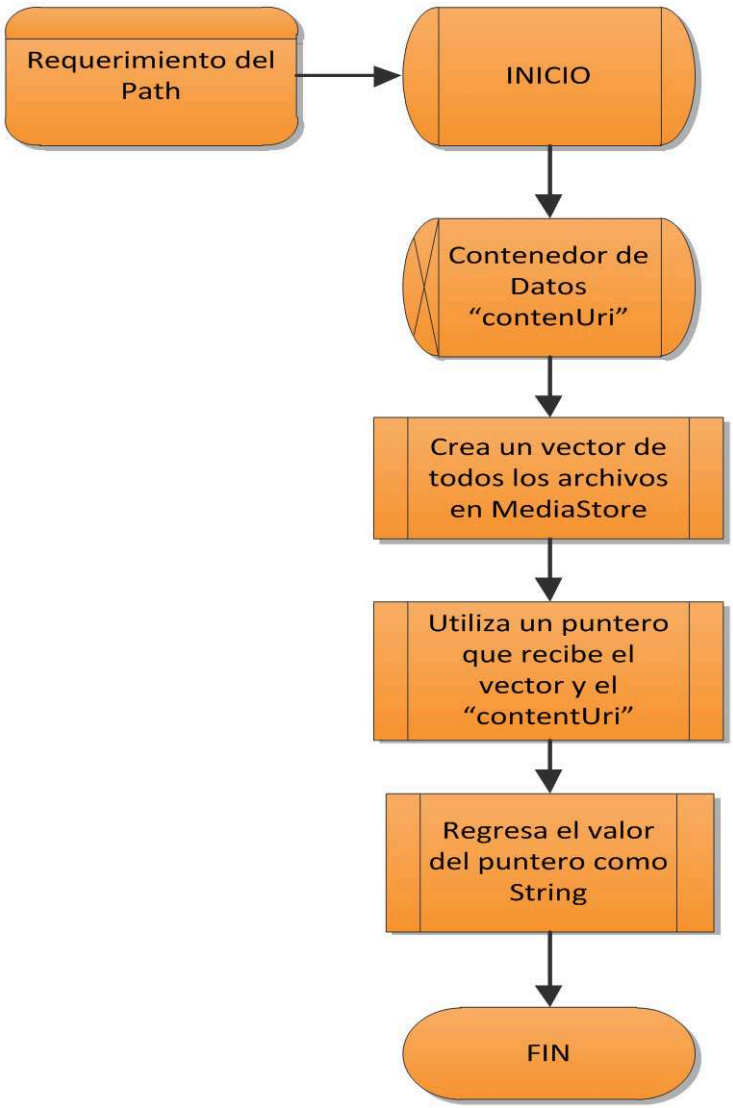


Figura 2.32: Método Ruta de Acceso

- Los parámetros necesarios para este método son:
  - Ingresa la ruta relativa referida al Content:// de MediaStore
- El funcionamiento se determina a través del código que ejecuta el método.

```
Public String getRealPathFromURI(Uri contentUri) { //Nombre del método.
      //Crea un cursor de la base de datos multimedia en blanco.
      Cursor cursor = managedQuery(contentUri, proj, null, null, null);
      int column_index = cursor.getColumnIndexOrThrow
      (MediaStore.Video.Media.DATA);//Se apunta el cursor al archivo
                                    seleccionado.
      cursor.moveToFirst();//Se ubica en cursor en la primera posición.
return cursor.getString(column_index);
```

2.4.5.7 Actividad por Resultados para el módulo de Video y Audio

Permite capturar los datos generados por la galería multimedia, en el momento de finalizar la ejecución y recupera el PATH absoluto en la variable “Name”.

Se representa el funcionamiento del método mediante el diagrama de flujo de la Figura 2.33.

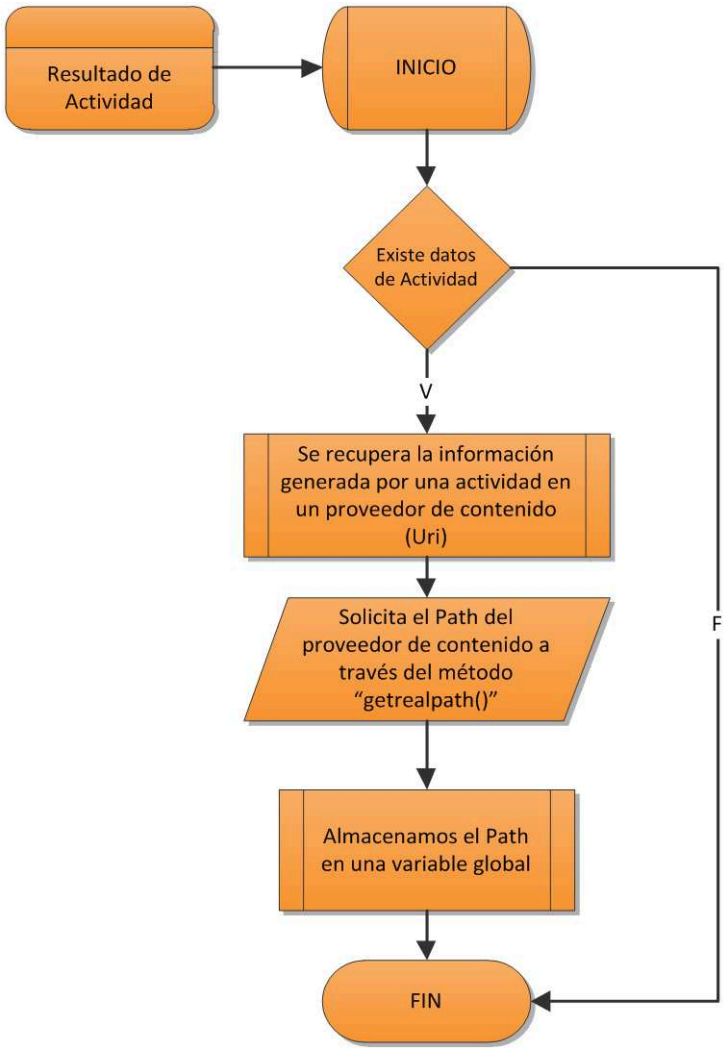


Figura 2.33: Actividad por resultados

- Los parámetros necesarios para este método son:
  - Código requerido en la variable “requestCode” se especifica cuando se ejecuta el comando como parámetro de entrada.
  - Los datos generados se almacenan en la variable “data”.

- El funcionamiento se determina a través del código que ejecuta el método.

```
if (requestCode ==1 && data!=null  ){
      Uri fileUri = data.getData();//
      Name =getRealPathFromURI(fileUri);//
      displayTxt(Name);
}
```

2.4.5.8 Método Barra de Búsqueda (SeekBar) para Audio y Video

Permite adelantar o retroceder un archivo de video o audio mientras se encuentra reproduciendo.

Este método consta de dos partes:

- La primera sincroniza la barra mientras el MediaPlayer esta activo.
- La segunda cambia la posición en la línea de tiempo del reproductor.

Se representa el funcionamiento de la sincronización de la barra mediante el diagrama de flujo de la Figura 2.34.

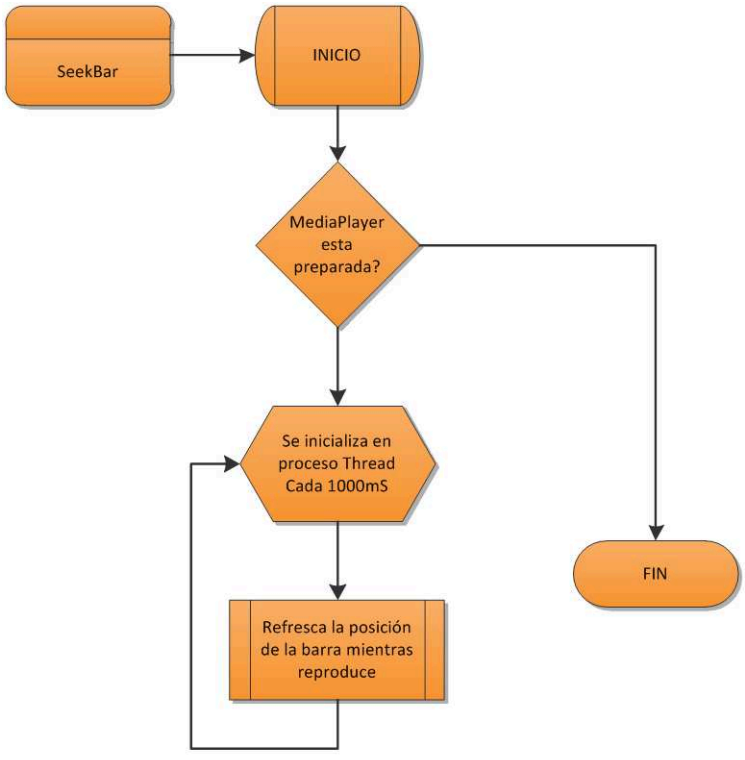


Figura 2.34: SeekBar Sincronización

Se representa el funcionamiento del método para adelantar y retroceder mientras reproduce un archivo de video o audio, mediate el diagrama de flujo de la Figura 2.35.

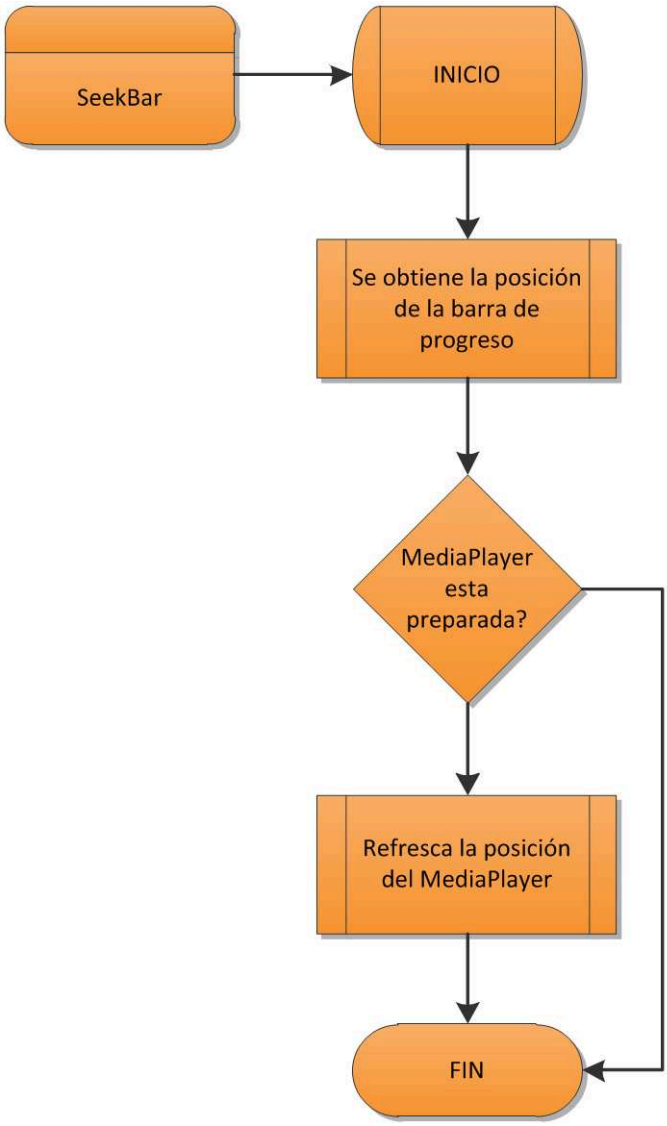


Figura 2.35: SeekBar Adelantar o Retroceder

- Los parámetros necesarios para este método son:
  - Se establece el temporizador de actualización de la barra, en 1000mseg significa que la barra será sincronizada cada segundo porque no es necesario un tiempo menor.
  - Se obtiene el progreso de la interfaz gráfica que interactúa con el usuario.

- El funcionamiento se determina a través del código que ejecuta el método.
  - Para la sincronización de la barra de búsqueda.

```
while(mp!=null && mp.getCurrentPosition()<mp.getDuration()){
      seekBar.setProgress(mp.getCurrentPosition()); //Consulta al MediaPlayer
                                                    la posición de tiempo actual
                                                    en milisegundos.
      try {
            Thread.sleep(1000);//Se indica el tiempo de espera en milisegundos.
      }
      //Sentencias que controlan los errores, solo si es necesario.
      catch (InterruptedException e) {
      return;
      } catch (Exception e) {
      return;
       }
}).start();
```

  - Para adelantar o retroceder mientras reproduce audio o video.

```
//En método se ejecuta automáticamente cuando se cambia la posición de la barra
de búsqueda.
public void onStopTrackingTouch(SeekBar arg0) {
      int milis=arg0.getProgress();
      if (Name!=""){
      mediaPlayer.seekTo(milis);//Indica al MediaPlayer que se ubique en la
                  posición almacenada en la variable milis que es generada
                  por la posición actúan de la barra de búsqueda
      }
}
```

CAPÍTULO 3

SIMULACIÓN E INSTALACIÓN DE LA APLICACIÓN EN EL DISPOSITIVO MÓVIL

3.1 PRESENTACIÓN DE LA APLICACIÓN

A la aplicación desarrollada por sus características y por la plataforma utilizada como núcleo de sus funciones, se le ha otorgado el nombre de DROID MEDIA (Figura 3.1).

Figura 3.1: Logotipo Droid Media

En la Figura 3.2 se presenta el icono de la aplicación Droid Media que se muestra en el escritorio del dispositivo.

Figura 3.2: Icono Droid Media

3.2 DESCRIPCIÓN DE LA INTERFAZ GRÁFICA

En la aplicación Droid Media se tiene diferentes interfaces de pantalla diseñadas para que el usuario maneje sus archivos multimedia con la mayor comodidad posible. A continuación se detalla la interfaz gráfica de cada módulo:

- Pantalla Principal.
- Módulo de Audio.
- Módulo de Imágenes.
- Módulo de Video.

3.2.1 PANTALLA PRINCIPAL

En la Figura 3.3 se presenta la pantalla principal en el momento se ejecuta la aplicación la misma que contiene el acceso a los módulos de Audio, Video e Imágenes.

Figura 3.3: Pantalla Principal

Descripción de los iconos de la pantalla principal.

- Ejecuta el módulo de Audio (Figura 3.4).

Figura 3.4: Icono del módulo de Audio

- Ejecuta el módulo de Imágenes (Figura 3.5).

Figura 3.5: Icono del módulo de Imágenes

- Ejecuta el módulo de Video (Figura 3.6).

Figura 3.6: Icono del módulo de Imágenes

3.2.2 MÓDULO DE AUDIO

La Figura 3.7 presenta la pantalla en la cual, se desarrolla la funcionalidad del módulo de Audio

Figura 3.7: Pantalla módulo de Audio

Descripción de los iconos de la pantalla del módulo de Audio

- Ejecuta la biblioteca musical para seleccionar un archivo de audio almacenado previamente en el dispositivo (Figura 3.8).

Figura 3.8: Icono Biblioteca Musical

- Despliega en pantalla el volumen actual del dispositivo (Figura 3.9).

Figura 3.9: Icono Indicador de Volumen

- Reproduce la canción seleccionada (Figura 3.10).

Figura 3.10: Icono *Play Music*

- Pone en pausa al MediaPlayer (Figura 3.11).

Figura 3.11: Icono Pausa

- Detiene la reproducción y resetea el MediaPlayer (Figura 3.12).

Figura 3.12: Icono Detener

- Graba audio directamente del micrófono del dispositivo (Figura 3.13).

Figura 3.13: Icono Grabar Audio

- Permite eliminar el archivo seleccionado (Figura 3.14).

Figura 3.14: Icono Borrar

- Permite enviar los archivos multimedia vía Bluetooth y también es posible enviarlos como un adjunto de un *e-mail* o publicar en las redes sociales disponibles en el dispositivo (Figura 3.15).

Figura 3.15: Icono Enviar

3.2.3 MÓDULO DE IMÁGENES

La Figura 3.16 presenta la pantalla en la cual se desarrolla la funcionalidad del módulo de imágenes.

Figura 3.16: Pantalla módulo de Imágenes

Descripción de los iconos de la pantalla del módulo de Imágenes. No se describen los iconos ya mencionados en el literal (3.2.2)

- Ejecuta la Galería multimedia, permitiendo seleccionar una imagen previamente almacenada en el dispositivo (Figura 3.17).

Figura 3.17: Icono Galería de Imágenes

- Permite captura imágenes desde la cámara del dispositivo (Figura 3.18).

Figura 3.18: Icono Capturar Fotografía

- Indica la posición relativa donde se mostrará la imagen capturada o seleccionada de la galería de imágenes (Figura 3.19).

Figura 3.19: Indicador de posición relativa de las imágenes

3.2.4 MÓDULO DE VIDEO

La Figura 3.20 presenta la pantalla en la cual se desarrolla la funcionalidad del módulo de Video.

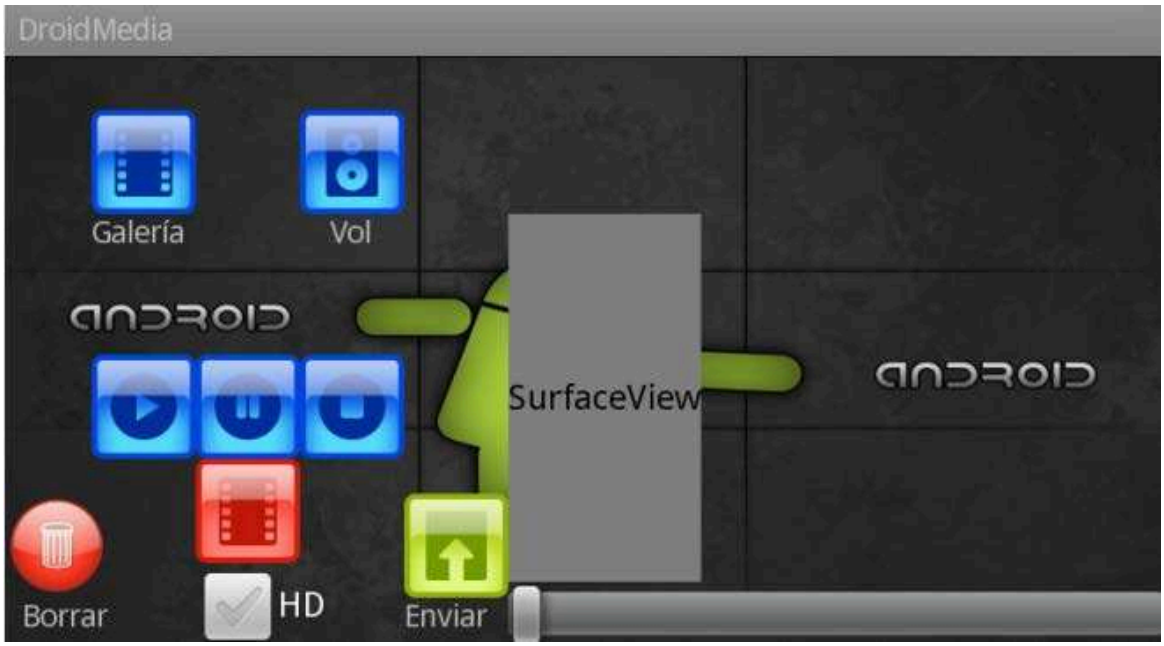


Figura 3.20: Pantalla del módulo de Video

Descripción de los iconos de la pantalla del módulo de Video. No se describen los iconos ya mencionados en el literal (3.2.2)

- Ejecuta la galería de Videos permitiendo seleccionar un video previamente almacenado en el dispositivo (Figura 3.21).

Figura 3.21: Icono Galería de Videos

- Permite la captura de video (Figura 3.22).

Figura 3.22: Icono Captura de Videos

- CheckBox HD establece el perfil de captura de video a la mayor calidad posible (Figura 3.23).

Figura 3.23: CheckBox HD

- SurfaceView es la superficie que permite la reproducción de video adaptándose al tamaño de pantalla (Figura 3.4).

Figura 3.24: *SurfaceView*

3.3 COMPATIBILIDAD DE LA APLICACIÓN

Se determinó el nivel API de la aplicación en Android 2.2 (Froyo) de nivel API 8, esto permite que la aplicación sea instalada en el API correspondiente o superior, lo que representa más del 90% de dispositivos Android registrados en Google *Market.*

La Tabla 3.1 se basa en el número de dispositivos Android que han accedido a Google *Market* en un lapso de 14 días hasta el 5 de marzo del 2012, se muestran los siguientes valores.

Plataforma	Nombre	Nivel API	Distribución
Android 1.5	*Cupcake*	3	0.4%
Android 1.6	*Donut*	4	0.8%
Android 2.1	*Eclair*	7	6.6%
Android 2.2	*Froyo*	8	25.3%
Android 2.3 - Android 2.3.2	*Gingerbread*	9	0.5%
Android 2.3.3 - Android 2.3.7		10	61.5%
Android 3.0	*Honeycomb*	11	0.1%
Android 3.1		12	1.1%
Android 3.2		13	2.1%
Android 4.0 - Android 4.0.2	*Ice Cream Sandwich*	14	0.4%
Android 4.0.3		15	1.2%

Tabla 3.1: Dispositivos Android en el mundo [37]

La Figura 3.25 representa en número de dispositivos Android registrados en Google *Market*.

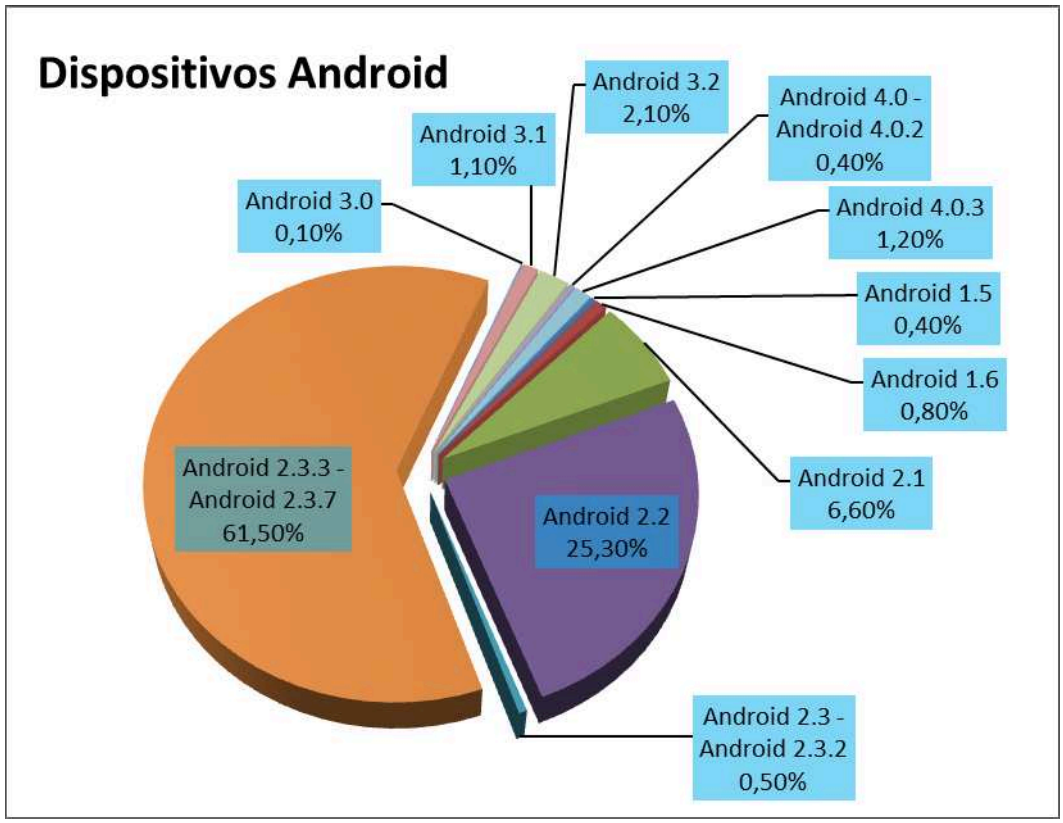


Figura 3.25: Dispositivos Android en el mundo [37**]**

La arquitectura del sistema operativo Android permite que las aplicaciones puedan ser instaladas en versiones superiores para la cual fueron desarrolladas siendo completamente compatibles sin generar errores.

3.4 SIMULACIÓN Y PRUEBAS DE LA APLICACIÓN EN LA MÁQUINA VIRTUAL DE ANDROID

Para empezar con la simulación primero se debe crear una máquina con los parámetros requeridos por la aplicación.

3.4.1 CREACIÓN Y COFIGURACIÓN DE UNA MÁQUINA VIRTUAL

Se utiliza el Android *Virtual Device* (AVD) para crear la máquina virtual y se configura los siguientes parámetros:

- **Nombre:** Se establece el nombre de la máquina virtual
- ***Target*:** Nivel API del emulador.

- **CPU/ABI:** Tipo de procesador tiene uno solo por defecto.
- ***Sd Card*:** Se establece la capacidad de la memoria externa en Megabytes o a su vez en una imagen desde un archivo con extensión .iso.
- **Skin:** Configura el tamaño y resolución de la pantalla del dispositivo.
- **Hardware:** Dispositivos incluidos para la simulación como capara y memoria externa.

En la Figura 3.26 se indica la configuración del AVD

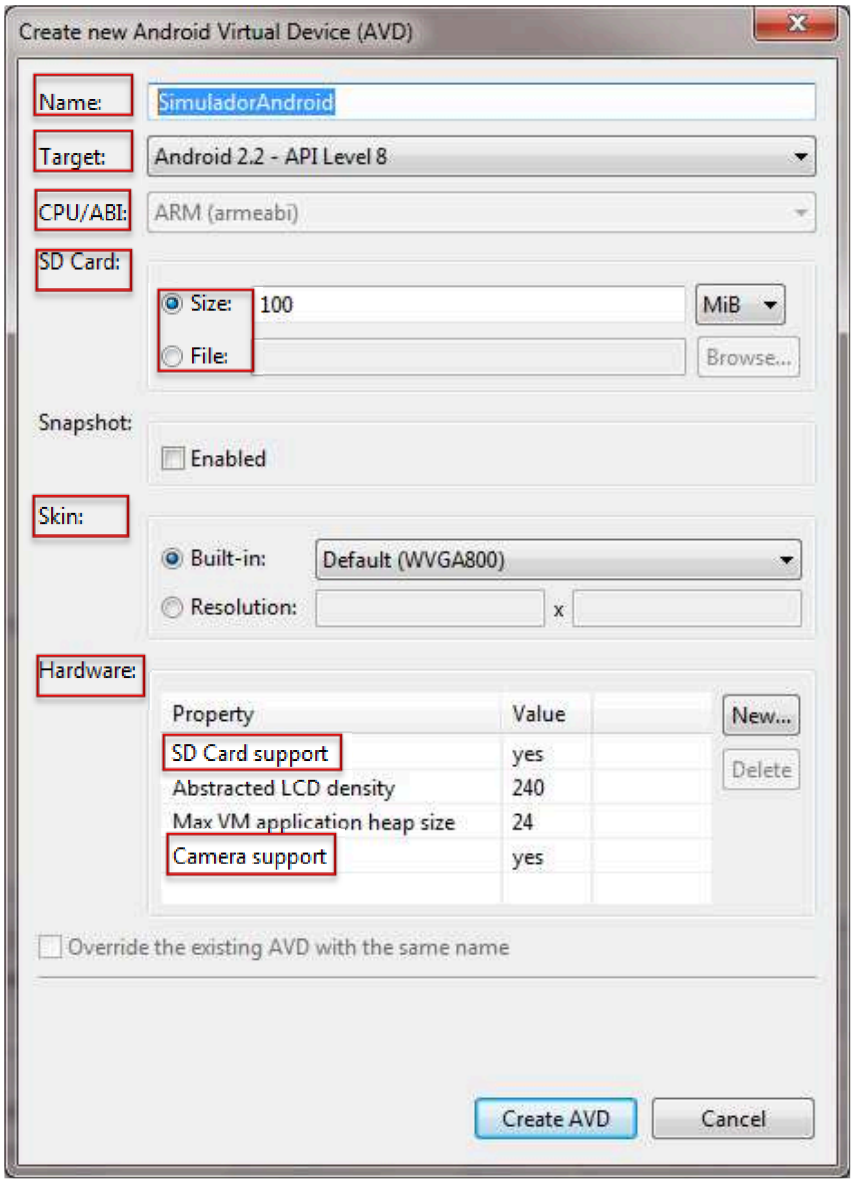


Figura 3.26: Ventana de configuración (AVD)

Para finalizar se presiona "*Create* AVD."

3.4.2 INSTALACIÓN DE LA APLICACIÓN EN LA MÁQUINA VIRTUAL

La instalación de la aplicación en la máquina virtual se llevará a cabo por medio del Android *Debug Bridge* (ADB) que permite la conexión directa del entorno de programación Eclipse y la máquina virtual de Android.

3.4.2.1 Pasos para la instalación en el emulador

Se observa en la Figura 3.27, la ventana de configuración de ejecución con los pasos de instalación.

- **Paso 1:** seleccionamos la aplicación que va ser instalada.
- **Paso 2:** Seleccionemos la máquina virtual previamente instalada.
- **Paso 3:** Click en el botón *Run*.

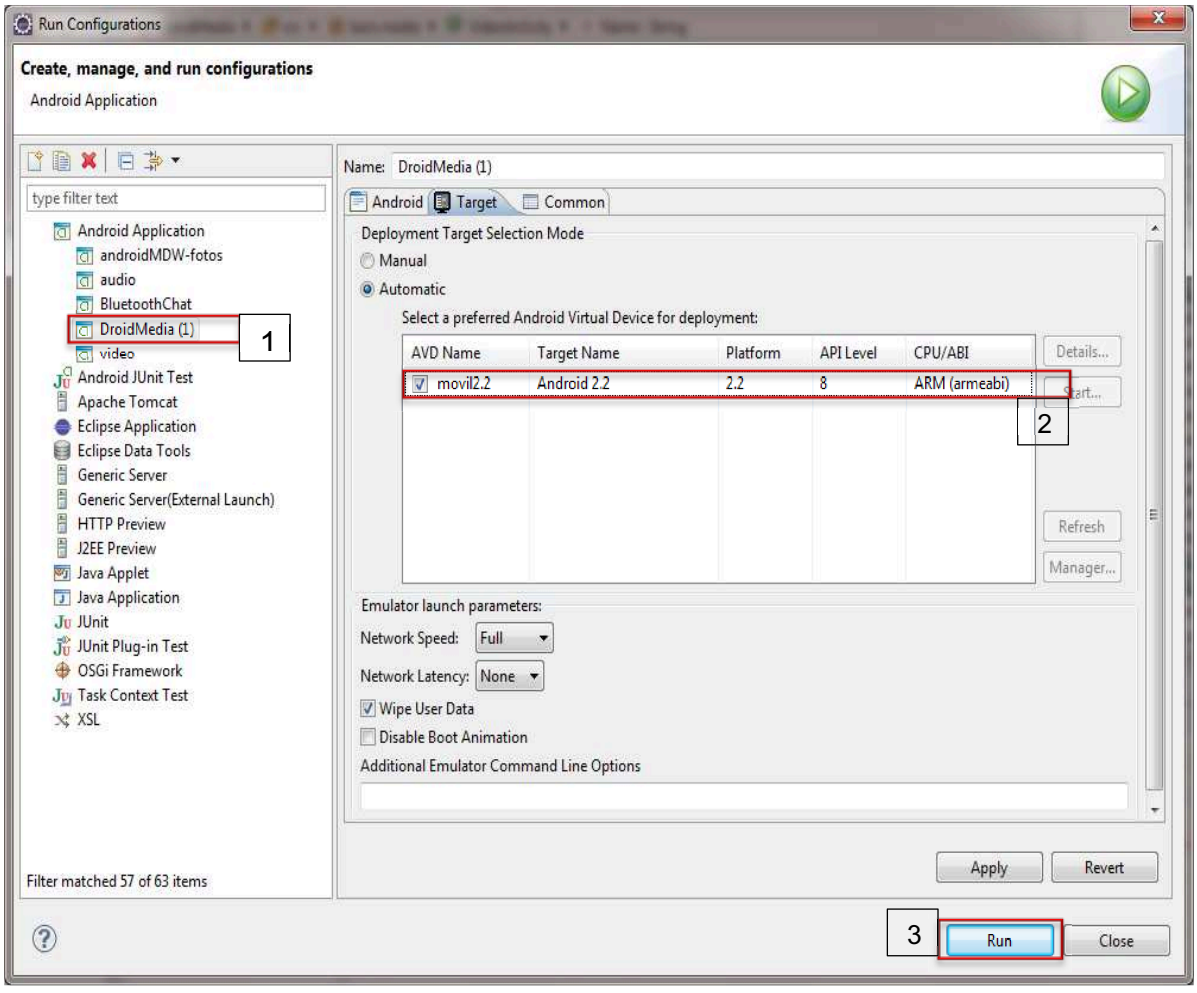


Figura 3.27: Run Cofigurations Eclipse IDE

- **Paso 4**: Emulador iniciado en la Figura 3.28.

Figura 3.28: Emulador Android 2.2

- **Paso 5**: Seleccionemos la aplicación Droid media como se observa en la Figura 3.29

Figura 3.29: Menú Emulador Android 2.2

3.4.3 RESULTADOS DE LAS PRUEBAS EN EL EMULADOR DE ANDROID

Se ejecutan pruebas de funcionamiento comprobando que cumpla con los requerimientos descritos en el Capítulo 2 para cada módulo.

3.4.3.1 Análisis de resultados en el emulador de Android para el módulo de Audio

Se presentan los resultados en la Tabla 3.2.

Requerimientos	Observación
Permitir la grabación de audio en formato ARM_NB.	Correcto
Registrar en la base de datos del sistema operativo los archivos generados.	Correcto
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Reproducir archivos de sonido almacenados en la memoria desde la biblioteca de audio.	Correcto
Reproducir los formatos mp3, arm, wav y midi.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	El Emulador no dispone de Bluetooth
Buscar una posición de tiempo en el audio que se está reproduciendo.	Correcto

Tabla 3.2: Resultados de la simulación del módulo de Audio

3.4.3.2 Análisis de resultados en el emulador de Android para el módulo de Imágenes

Se presentan los resultados en la Tabla 3.3.

Requerimientos	Observación
Permitir la captura de imágenes en formato JPEG.	Emula una cámara virtual
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Presentar las imágenes capturadas en la interfaz principal.	Correcto
Permitir mostrar archivos almacenados en el dispositivo.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	El emulador no dispone de Bluetooth

Tabla 3.3: Resultados de la simulación del módulo de Imágenes

3.4.3.3 Análisis de resultados en el emulador de Android para el módulo de Video

Se presentan los resultados en la Tabla 3.4.

Requerimiento	Observación
Permitir la grabación de video en formato MPEG4	Emula una cámara virtual
Registrar en la base de datos del sistema operativo los archivos de video generados por la actividad de captura.	Correcto
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Reproducir archivos de video almacenados en la memoria	Correcto
Reproducir los formatos.3gp y .mp4	Correcto
Buscar una posición de tiempo en el video que se está reproduciendo	Correcto
Presentar el video en pantalla completa según desee el usuario.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	El emulador no dispone de Bluetooth
Permite seleccionar la calidad de video al usuario según sus propósitos; en baja y alta calidad (HD)	Correcto

Tabla 3.4: Resultados de la simulación del módulo de Video

3.5 INSTALACIÓN Y PRUEBAS DE FUNCIONAMIENTO DE LA APLICACIÓN EN EL DISPOSITIVO ANDROID

A continuación se realiza la instalación y las pruebas de funcionamiento de la aplicación Droid Media en el dispositivo Android.

3.5.1 INSTALACIÓN DE LA APLICACIÓN EN EL DISPOSITIVO ANDROID

Los pasos para la instalación de la aplicación Droid Media en un dispositivo Android son los siguientes:

Paso 1: Copiar la aplicación DroidMedia.Apk en la memoria SD del dispositivo como se muestra en la Figura 3.30

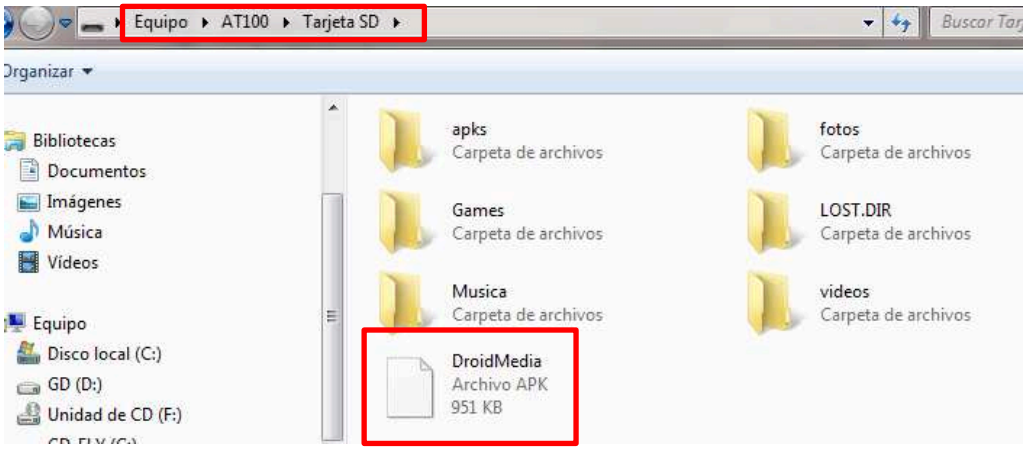


Figura 3.30: Directorio de la memoria SD del dispositivo

Paso 2: Instalar la aplicación con el gestor de archivos del sistema operativo Android seleccionando la acción “Instalador de paquetes” y aceptando el mensaje de confirmación con el detalle de los permisos garantizados para la aplicación como se muestra en las Figuras 3.31, 3.32 y 3.33.

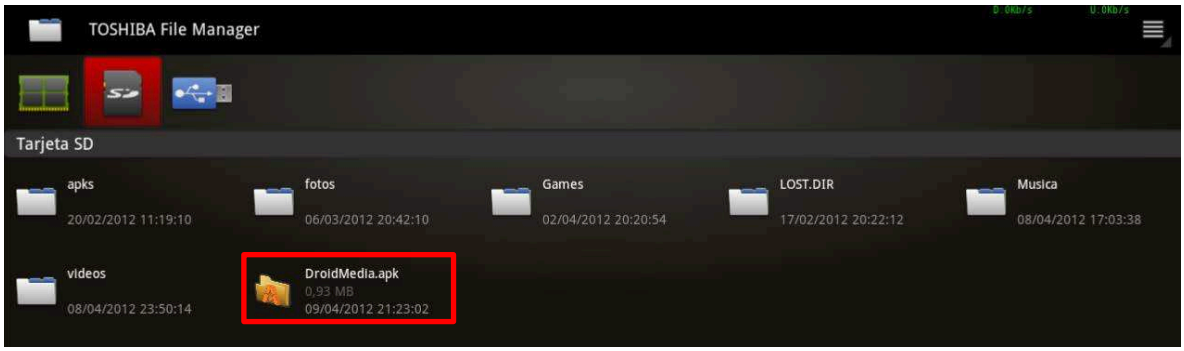


Figura 3.31: Gestor de archivos dispositivo Android

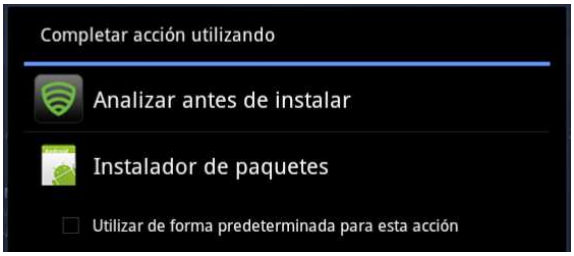


Figura 3.32: Instalador de Paquetes

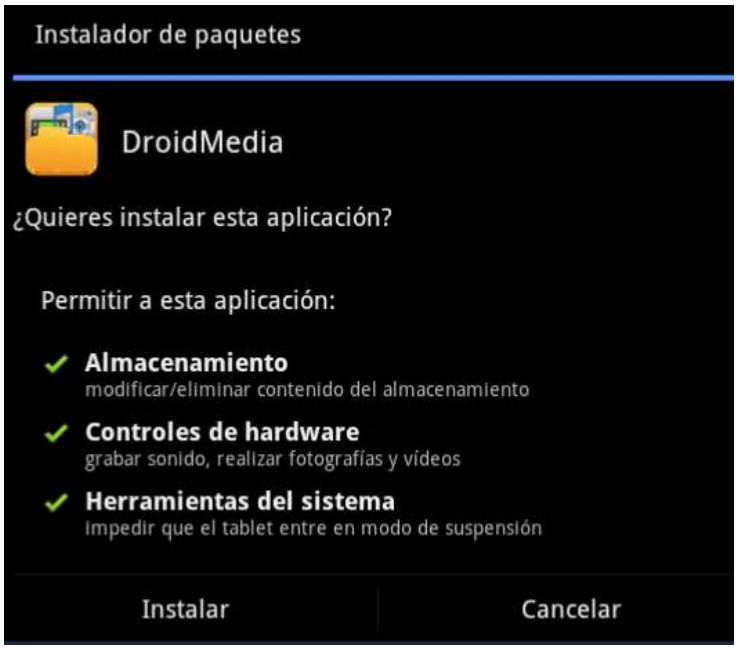


Figura 3.33: Mensaje de confirmación

Paso 3: Ejecutar la aplicación instalada Figura 3.34

Figura 3.34: Aplicaciones instaladas en el dispositivo

El resultado de la instalación se puede observar en la Figura 3.35.

Figura 3.35: Pantalla principal Droid Media en el dispositivo

3.5.2 RESULTADO DE LAS PRUEBAS DE FUNCIONAMIENTO EN EL DISPOSITIVO ANDROID

Se ejecutaron todas las pruebas ya indicadas en el literal 3.4.3 y se encontraron dos errores señalados en el literal 3.5.2.4, además se debe considerar que el dispositivo si dispone Cámara y Bluetooth como muestran las Figuras 3.36 y 3.37.

Figura 3.36: Captura de video en la aplicación Droid Media

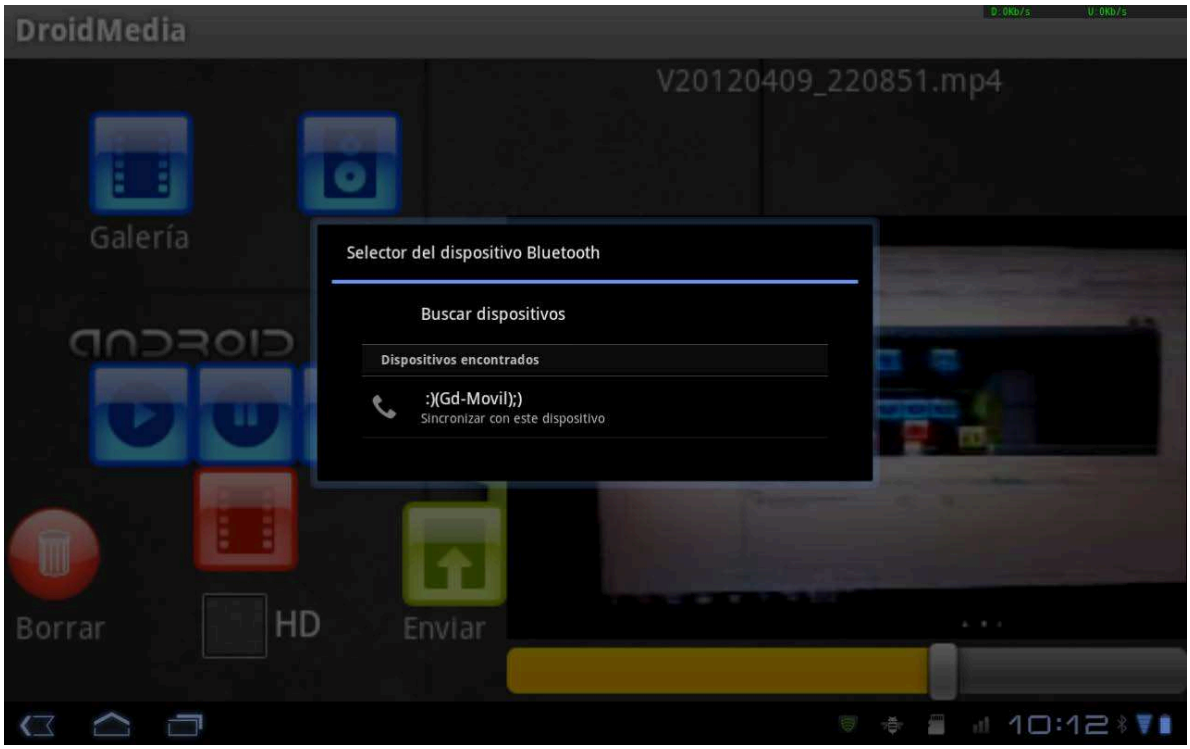


Figura 3.37: Captura enviando un archivo vía Bluetooth

3.5.2.1 Análisis de resultados en el dispositivo Android para el módulo de Audio

Se presentan los resultados en la Tabla 3.5.

Requerimientos	Observación
Permitir la grabación de audio en formato ARM_NB.	Correcto
Registrar en la base de datos del sistema operativo los archivos generados.	Correcto
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Reproducir archivos de sonido almacenados en la memoria desde la biblioteca de audio	Error, Corregido
Reproducir los formatos mp3, arm, wav y midi.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	Correcto
Buscar una posición de tiempo en el audio que se está reproduciendo.	Correcto

Tabla 3.5: Resultados de las pruebas del módulo de Audio

3.5.2.2 Análisis de resultados en el emulador de Android para el módulo de Imágenes

Se presentan los resultados en la Tabla 3.6.

Requerimientos	Observación
Permitir la captura de imágenes en formato JPEG.	Correcto
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Presentar las imágenes capturadas en la interfaz principal.	Error, Corregido
Permitir mostrar archivos almacenados en el dispositivo.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	Correcto

Tabla 3.6: Resultados de las pruebas del módulo de Imágenes

3.5.2.3 Análisis de resultados en el emulador de Android para el módulo de Video

Se presentan los resultados en la Tabla 3.7.

Requerimiento	Observación
Permitir la grabación de video en formato MPEG4	Correcto
Registrar en la base de datos del sistema operativo los archivos de video generados por la actividad de captura.	Correcto
Almacenar los archivos generados en la memoria externa del dispositivo.	Correcto
Reproducir archivos de video almacenados en la memoria	Correcto
Reproducir los formatos.3gp y .mp4	Correcto
Buscar una posición de tiempo en el video que se está reproduciendo	Correcto
Presentar el video en pantalla completa según desee el usuario.	Correcto
Permitir enviar vía Bluetooth archivos seleccionados previamente.	Correcto
Permite seleccionar la calidad de video al usuario según sus propósitos; en baja y alta calidad (HD)	Correcto

Tabla 3.7: Resultados de las pruebas del módulo de Video

3.5.2.4 Corrección de los errores que se presentaron duran la ejecución de la aplicación en el dispositivo Android

Se presentaron dos errores, uno en el módulo de audio y otro en el módulo de imágenes

3.5.2.4.1 Error en el módulo de Audio

El error es generado por el método de la biblioteca musical los comandos utilizados especifican que se use directamente la biblioteca, pero no es valido en todos los dispositivos, los comandos que fueron remplazados son:

```
Intent intent = new Intent(Intent.ACTION_PICK, android.provider.
MediaStore.Audio  //Se indica que se debe usar la biblioteca de Audio el mismo
que producía el error
.Media.EXTERNAL_CONTENT_URI); //Localización de archivos en la memoria externa
startActivityForResult(intent,1);//inicia la actividad con los parámetros
                                 configurados
```

Se soluciona permitiendo al usuario seleccionar la biblioteca musical que desee, utilizando los siguientes comandos.

```
String path = ""; //Se crea una variable tipo string en blanco para almacenar
                  el path
path = Environment.getExternalStorageDirectory().getAbsolutePath();//Ruta de
los archivos de audio
Intent intent1= new Intent(path); //Se crea un nuevo proceso
intent1.setType("audio/*");//Tipo de acción
intent1.setAction(Intent.ACTION_GET_CONTENT); //Configuración de la acción
seleccionado
intent1.addCategory(Intent.CATEGORY_OPENABLE); //Configuración de la categoría
startActivityForResult(Intent.createChooser(intent1, "Select Music"), 1); //Se
inicia el proceso
```

3.5.2.4.2 Error en el módulo de Imágenes

El error se genera cuando se quiere usar la memoria del *buffer* de datos que no se encuentra disponible en dicho momento, generalmente ocurre cuando se despliega diferentes imágenes en pantalla sin eliminar la imagen anterior.

Se soluciona introduciendo los siguientes comandos en el método de actividad por resutados para liberar la memoria.

```
imgV.destroyDrawingCache(); //imgV hace referencia al objeto imageview que se
utiliza para desplegar imágenes
imgV.setImageDrawable(null);
```

3.5.2.5 Pruebas de trasferencia de archivos vía Bluetooth

Se utilizó Bluetooth versión 3.0 de clase 2 incorporado en el dispositivo Android.

Se ejecutan pruebas de transferencia de archivos variando la distancia con dos dispositivos adicionales en un ambiente *indoor* como se muestra en la Tabla 3.8 y Tabla 3.9.

- **Dispositivo 1**

Las características del Bluetooth son: Versión 2.0 y Clase 2.

Distancia (m)	Tamaño (MBytes)	Tiempo (Segundos)	Estado del envío	Velocidad (Mbits)
0	4,27	40,8	Correcto	0,837
2	4,27	41,3	Correcto	0,827
4	4,27	60,1	Correcto	0,568
6	4,27	83,8	Correcto	0,408
8	4,27	113,1	Correcto	0,302
10	4,27	115,2	Correcto	0,297
12	4,27	300	Error	0,114

Tabla 3.8: Resultados de trasferencia Bluetooth dispositivo 1

- **Dispositivo 2**

Las características del Bluetooth son: Versión 3.0 y Clase 2.

Distancia (m)	Tamaño (MBytes)	Tiempo (Segundos)	Estado del envío	Velocidad (Mbits)
0	4,27	30,6	Correcto	1,116
2	4,27	33,2	Correcto	1,029
4	4,27	46,4	Correcto	0,736
6	4,27	46,4	Correcto	0,736
8	4,27	42,7	Correcto	0,800
10	4,27	46,7	Correcto	0,731
12	4,27	45,5	Correcto	0,751
14	4,27	59,8	Correcto	0,571
16	4,27	N/A	Error	N/A

Tabla 3.9: Resultados de trasferencia Bluetooth dispositivo 2

En las pruebas de transferencia, se observa que se cumple con los parámetros de operación y se comprueba que la velocidad de transmisión depende de la distancia.

Nuevas versiones de Bluetooth mejoran la velocidad de transferencia y el alcance.

3.6 COSTO APROXIMADO DE IMPLEMENTACIÓN APLICACIÓN

Se presenta el costo referencial de la aplicación Tabla 3.10 en base a las horas requeridas para el diseño, desarrollo y depuración de errores; ya que la plataforma de desarrollo y licencias son de tipo libre.

Se utiliza los siguientes valores referenciales para el cálculo del costo por hora[49]

- Salario para personal de diseño $850 por lo tanto se invertiría $5,32 c/hora de diseño.
- Salario para el personal de desarrollo y depuración $1150 por lo tanto se invertiría $7,19 c/hora de desarrollo y depuración.

Costo de Implementación			
Actividad	**Horas**	**Costo por Hora**	**Total**
Diseño	60	$ 5,31	$ 318,75
Desarrollo	90	$ 7,19	$ 646,88
Depuración	40	$ 7,19	$ 287,50
		Total	$ 1.253,13

Tabla 3.10: Costo de Implementación

Se debe tomar en cuenta que la aplicación no tendrá costo para el usuario y se recuperaría la inversión, por medio de la venta de servicios de publicidad en Android.

[49] Los valores de los salarios utilizados para el calculo, fueron obtenidos del Gerente de desarrollo del Banco Solidario

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El sistema operativo Android permite al programador utilizar todos los recursos del dispositivo sin limitaciones; pone a disposición todo el hardware como procesador, memoria, pantalla, altavoces, botones, Wi-Fi, Bluetooth. Permitiendo desarrollar aplicaciones que cumplen los requerimientos y altas expectativas del usuario final de hoy en día. Que por generalmente es el único que juzga el atractivo, funcionalidad y comodidad de uso de la aplicación.

- El lenguaje de programación Java, utilizado en el desarrollo de los módulos de la aplicación, es muy versátil para incorporar nuevos métodos y facilita la visualización de cada uno, de manera ordenada, secuencial y clasificados de manera jerárquica en orden de ejecución, contribuyendo a la visión general del programador para poder depurar una aplicación más rápidamente.

- La aplicación Droid Media tiene la ventaja de incluir todas las tareas relacionas con archivos multimedia en una sola interfaz. La misma que al ser desarrollada para un sistema operativo libre no limita su distribución y abarata costos por no necesitar licencias. Se asegura compatibilidad para versiones posteriores del sistema operativo, solo se debe cumplir un requisito mínimo de API para ser instalada y disfrutar de sus prestaciones.

- Este tipo de aplicaciones pueden ser actualizadas frente a posibles competidores con la colaboración de la retroalimentación que realiza el

usuario después de ser instalada en su dispositivo. Facilidad de ser publicada en el mercado mundial a través de Google Play, el costo de la licencia de publicación es bajo y asequible a cualquier desarrollador independiente.

- El desarrollo del código y la generación de algoritmos de la aplicación en lenguaje java, se facilita con el apoyo gráfico que brinda los diagramas de flujo diseñados previamente. Comprenden los métodos, condiciones, funciones y la respectiva secuencia ordenada de cada proceso. Respetando las condiciones lógicas que establece del sistema operativo Android.

- La simulación de la aplicación, permite observar el resultado de la programación de manera gráfica, detectando posibles excepciones y errores no contemplados en el momento del desarrollo; los mismos que se pudieran generar en un dispositivo. Sin embargo el emulador no contempla errores del uso excesivo de memoria RAM y del Buffer del dispositivo, situaciones que se presentan cuando la aplicación es instalada y probada en el dispositivo.

- Se utilizaron 1287 líneas de código en java, el mismo que permite el uso de librerías embebidas del sistema operativo Android, lo cual simplificó la programación requerida para resolver el actual proyecto de titulación.

- Se obtuvo los resultados deseados de la aplicación desarrollada, se verificó el cumplimiento de los requerimientos:
    - Permitir al usuario la captura, reproducción, almacenamiento de archivos multimedia.

- Permitir al usuario el envío de archivos multimedia mediante una conexión Bluetooth.
- Permitir al usuario la administración de archivos multimedia lo que comprende uso de la galería de imagen, videos, y la biblioteca musical del sistema operativo. Además permitiendo una posible eliminación si el usuario lo requiere.

- Se presentaron los siguientes problemas:
  - En el módulo de Audio el comando ActionPick del método de la "Biblioteca Musical" genero un error que termina la ejecución del programa porque no todos los dispositivos Android utilizan la misma aplicación para manejar los archivos de audio; se soluciona permitiendo al usuario que escoja la aplicación que desee.
  - En el módulo de imágenes, se generaba un desbordamiento del *buffer* de la memoria cache cuando se decodificaban las imágenes; se solucionó eliminado los datos anteriores almacenados en el cache antes de representar una nueva imagen.
  - Se debe tomar en cuenta que los problemas se presentaron cuando la aplicación fue ejecutada en el dispositivo y no en el emulador de Android.

4.2 RECOMENDACIONES

- Para el desarrollo de aplicaciones en Android se recomienda el uso del IDE Eclipse, por el acoplamiento directo que tiene con el *plugin* de Android (*Android Development Tools)*, que facilita la creación de nuevos proyectos estructurando el diseño básico sobre el cual se trabaja la aplicación en desarrollo, también hace un control de comandos indicando las respectivas opciones para completar el código y si es necesario invoca nuevas librerías del sistema operativo.

- Se puede optimizar el código para dispositivos de menor capacidad que no poseen gran cantidad de memoria RAM disminuyendo la calidad gráfica de la interfaz implementada, ya que se desarrolló para dispositivos HDp (*High Density pixel*), con una resolución no menor 480x320 pixles en pantallas de 3.7 pulgadas. Aunque la tendencia en los dispositivos de ahora es crecer en capacidad, prestaciones y resolución.

- Se recomienda depurar una aplicación directamente en el dispositivo Android porque el emulador representa un caso ideal en condiciones aisladas, que no genera errores ni excepciones. En el dispositivo se encuentra casos como:
  - Se ha excedido el límite de memoria.
  - Variable no contiene los valores respectivos.
  - Proveedor de contenidos no Disponible.
  - Caso de procesos no secuenciales que terminan la ejecución del programa.
  - Configuración errónea de parámetros de uso del hardware.
  - Procesos recursivos que generan bucles infinitos que en emulador no son evidentes.

- Conocer los alcances del sistema operativo como plataforma de desarrollo, para determinar el nivel API correcto y el mínimo posible, incrementando el rango de dispositivos que pueden utilizar dicha aplicación.

- Se recomienda usar librerías incluidas en el sistema operativo Android para el manejo del hardware ya que estas no producen conflictos en la ejecución de la aplicación.

REFERENCIAS

Sitios Web

[1] Ciencia y Tecnología.

Obtenido de:

http://www.rpp.com.pe/2011-06-17-conozca-el-papel-de-los-dispositivos-moviles-en-el-trabajo-noticia_376271.html

(Último acceso:12/01/2012)

[2] Los Sistemas Operativos: su historia y su concepto.

Obtenido de:

http://www.monografias.com/trabajos19/sistemas-operativos/sistemas-operativos.shtml.

(Último acceso:26/12/2011)

[3] Operative System Placement.

Obtenido de:

http://es.wikipedia.org/wiki/Archivo:Operating_system_placement-es.svg

(Último acceso:26/11/2011)

[4] Sistemas Operativos Centralizados y Distribuidos.

Obtenido de:

http://es.scribd.com/doc/17180457/SO-Centralizados-Distribuidos

(Último acceso:26/12/2011)

[5] Sistema Operativo Móvil

Obtenido de:

http://es.wikipedia.org/wiki/Sistema_operativo_m%C3%B3vil

(Último acceso:26/12/2011)

[6] Gartner Newsroom

Obtenido de:

http://www.gartner.com/it/page.jsp?id=2017015

(Último acceso:31/07/2012)

[7] PDA expertos. [Online].

Obtenido de:

http://www.pdaexpertos.com/foros/viewtopic.php?t=71539

(Último acceso:26/12/2011)

[9] Todo Symbian

Obtenido de:

http://www.todosymbian.com/secart29.html.

(Último acceso:15/11/2011)

[10] Archivo: BlackBerry-OS-6.0.jpg. [Online].

Obtenido de:

http://es.wikipedia.org/wiki/Archivo:BlackBerry-OS-6.0.jpg

(Último acceso:28/12/2011)

[11] Centro de dispositivos de Windows Mobile.

Obtenido de:

http://es.wikipedia.org/wiki/Archivo:BlackBerry-OS-6.0.jpg

(Último acceso:16/11/2011)

[12] IOS Sistema Operativo.

Obtenido de:

http://en.wikipedia.org/wiki/IOS

(Último acceso:16/11/2011)

[13] Android Resumen. [Online].

Obtenido de:

http://www.openhandsetalliance.com/android_overview.html

(Último acceso:17/11/2011)

[14] Comparación de los Sistemas Operativos móviles.

Obtenido de:

http://es.engadget.com/2009/03/19/la-gran-comparacion-de-los-sistemas-operativos-moviles/

(Último acceso:30/12/2011)

[15] Noticias Movíles.

Obtenido de:

http://www.moviles.com3.es/2011/05/02/

(Último acceso:2/02/2012)

[16] Android API levels.

Obtenido de:

http://developer.android.com/guide/appendix/api-levels.html

(Último acceso:24/01/2012)

[17] Android 1.1 Version Notes.

Obtenido de:

http://developer.android.com/sdk/android-1.1.html

(Último acceso:18/01/2012)

[18] Android 1.5 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-1.5.html

(Último acceso:18/01/2012)

[19] Android 1.6 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-1.6.html

(Último acceso:18/01/2012)

[20] Android 2.0 Release 1.

Obtenido de:

http://developer.android.com/sdk/android-2.0.html

(Último acceso:18/01/2012)

[21] Android 2.2 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-2.2.html

(Último acceso:24/01/2012)

[22] Android 2.3.3 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-2.3.3.html

(Último acceso:24/01/2012)

[23] Android 3.0 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-3.0.html

(Último acceso:24/01/2012)

[24] Android 4.0 Plataform.

Obtenido de:

http://developer.android.com/sdk/android-4.0-highlights.html

(Último acceso:24/01/2012)

[25] What is Android?.

Obtenido de:

http://developer.android.com/guide/basics/what-is-android.html

(Último acceso:13/01/2012)

[26] Application Fundamentals.

Obtenido de:

http://developer.android.com/guide/topics/fundamentals.html

(Último acceso:24/01/2012)

[27] El universo y la tecnologia conviviendo en completa armonia.

Obtenido de:

http://monillo007.blogspot.com/2007/12/conceptos-bsicos-de-programacin-en-java.html

(Último acceso:25/01/2012)

[29] Android Developers, Sdk Requirements.

Obtenido de:

http://developer.android.com/sdk/requirements.html

(Último acceso:9/03/2012)

[30] CURSO ANDROID.

Obtenido de:

http://www.sgoliver.net

(Último acceso:12/03/2012)

[31] Android Developers, Activity Lifecycle.

Obtenido de:

http://developer.android.com/reference/android/app/Activity.html

(Último acceso:15/03/2012)

[32] Android Developers, PublicDirectory.

Obtenido de:

http://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory

(Último acceso:15/03/2012)

[33] Android Developers, CamcorderProfile.

Obtenido de:

http://developer.android.com/reference/android/media/CamcorderProfile.html

(Último acceso:19/03/2012)

[34] Android Developers, MediaRecorder.

Obtenido de:

http://developer.android.com/reference/android/media/MediaRecorder.html

(Último acceso:19/03/2012)

[35] Android Developers, MediaPlayer.

Obtenido de:

http://developer.android.com/reference/android/media/MediaPlayer.html

(Último acceso:19/03/2012)

[36] Android Developers.

Obtenido de:

http://developer.android.com/guide/appendix/media-formats.html

(Último acceso:19/03/2012)

[37] Android Developers, Platform Versions.

Obtenido de:

http://developer.android.com/resources/dashboard/platform-versions.htm

(Último acceso:19/03/2012)

DOCUMENTOS EN LÍNEA

[8] Sistemas Operativos para Dispositivos Móviles.

Obtenido de:

http://miriammeza.files.wordpress.com/2010/09/presentacion.pdf

(Último acceso:15/11/2011)

[28] Historia de ANDROID

Obtenido de:

http://www.utm.mx/~caff/poo/AndroidIntro.pdf

(Último acceso:25/01/2012)

ANEXO A

MANUAL DE INSTALACIÓN

Los pasos para la instalación de la aplicación Droid Media en un dispositivo Android son los siguientes:

Paso 1: Copiar la aplicación DroidMedia.Apk en la memoria SD del dispositivo como se muestra en la Figura A.1

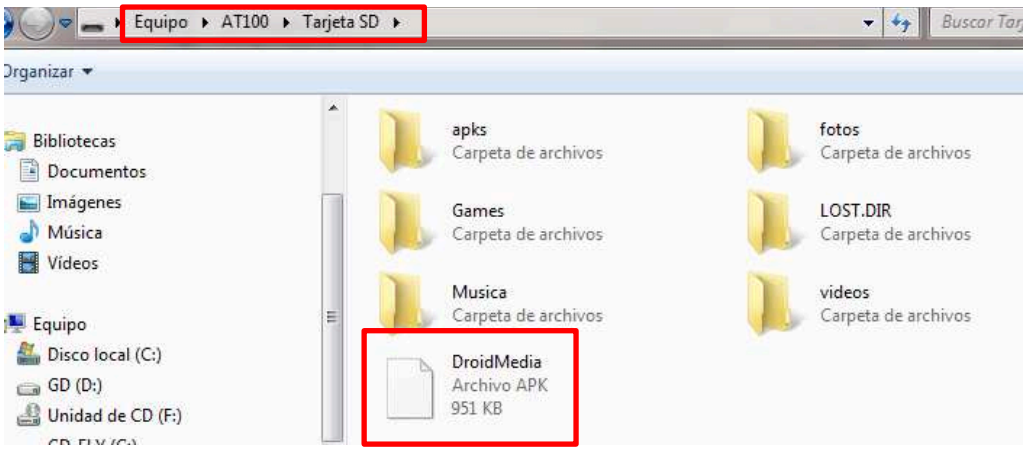


FIGURA A.1: Directorio de la memoria SD del dispositivo

Paso 2: Instalar la aplicación con el gestor de archivos del sistema operativo Android seleccionando la acción “Instalador de paquetes” y aceptando el mensaje de confirmación con el detalle de los permisos garantizados para la aplicación como se muestra en las Figuras A.2, A.3 y A.4.

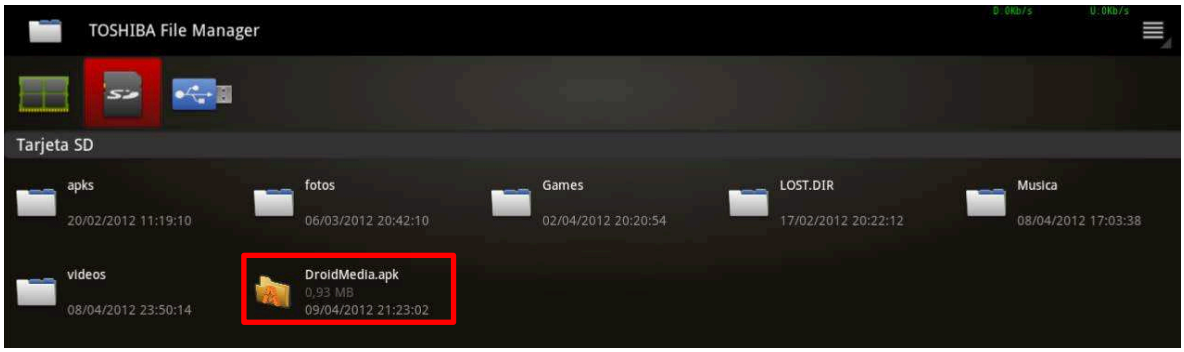


Figura A.2: Gestor de archivos dispositivo Android

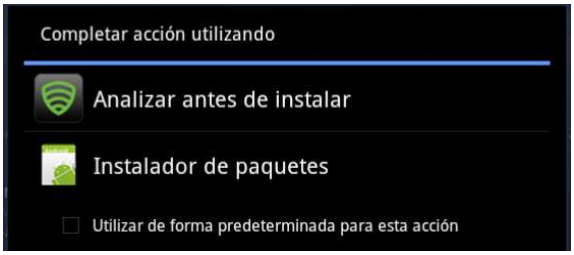


Figura A.3: Instalador de Paquetes

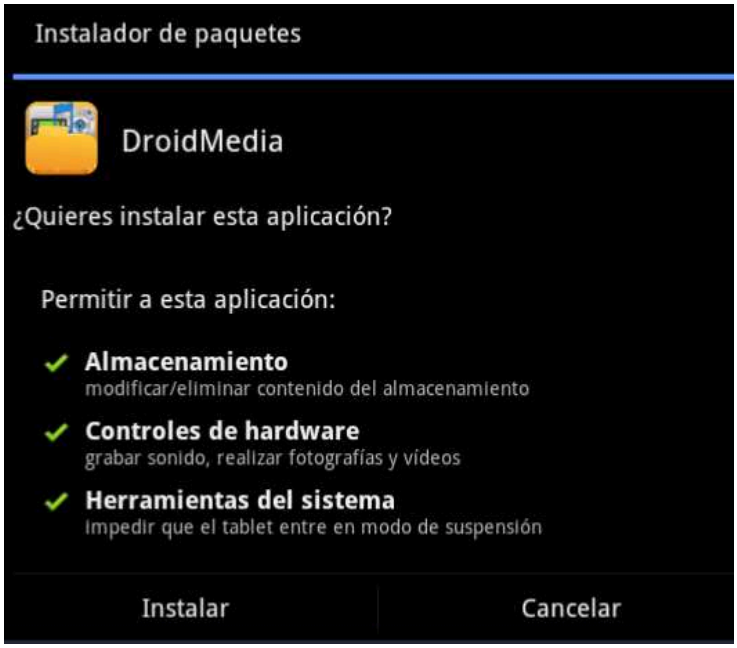


Figura A.4: Mensaje de confirmación

Paso 3: Ejecutar la aplicación instalada Figura A.5

Figura A.5: Aplicaciones instaladas en el dispositivo

MANUAL DE USUARIO

Droid Media es una aplicación intuitiva y de fácil manejo, a continuación se ha elaborado el respectivo manual de usuario para cada módulo con su interfaz gráfica.

Pantalla Principal

Figura A.6: Pantalla Principal

Descripción de los iconos de la pantalla principal (Figura A.6).

En la pantalla principal se encuentra los siguientes iconos:

- Ejecuta el módulo de Audio

- Ejecuta el módulo de Video

- Ejecuta el módulo Imágenes

Módulo de Audio

Figura A.7: Módulo de Audio

Descripción de los iconos de la pantalla del módulo de Audio (Figura A.7).

- Ejecuta la biblioteca musical para seleccionar un archivo de audio almacenado previamente en el dispositivo.

- Despliega en pantalla el volumen actual del dispositivo.

- Reproduce la canción seleccionada.

- Pone en pausa al MediaPlayer.

- Detiene la reproducción y resetea el MediaPlayer.

- Graba audio directamente del micrófono del dispositivo

- Permite eliminar el archivo seleccionado.

- Permite enviar los archivos multimedia vía Bluetooth y también es posible enviarlos como un adjunto de un *e-mail* o publicar en las redes sociales disponibles en el dispositivo.

- SeekBar permite adelantar o retroceder el archivo multimedia en curso.

Módulo de Imágenes

Figura A.8: Módulo de Imágenes

Descripción de los iconos de la pantalla del módulo de imágenes (Figura A.8).

- Ejecuta la Galería multimedia, permitiendo seleccionar una imagen previamente almacenada en el dispositivo.

- Permite captura imágenes desde la cámara del dispositivo.

- Indica la posición relativa donde se mostrará la imagen capturada o seleccionada de la galería de imágenes.

- Permite eliminar el archivo seleccionado.

- Permite enviar los archivos multimedia vía Bluetooth y también es posible enviarlos como un adjunto de un *e-mail* o publicar en las redes sociales disponibles en el dispositivo.

Módulo de Video

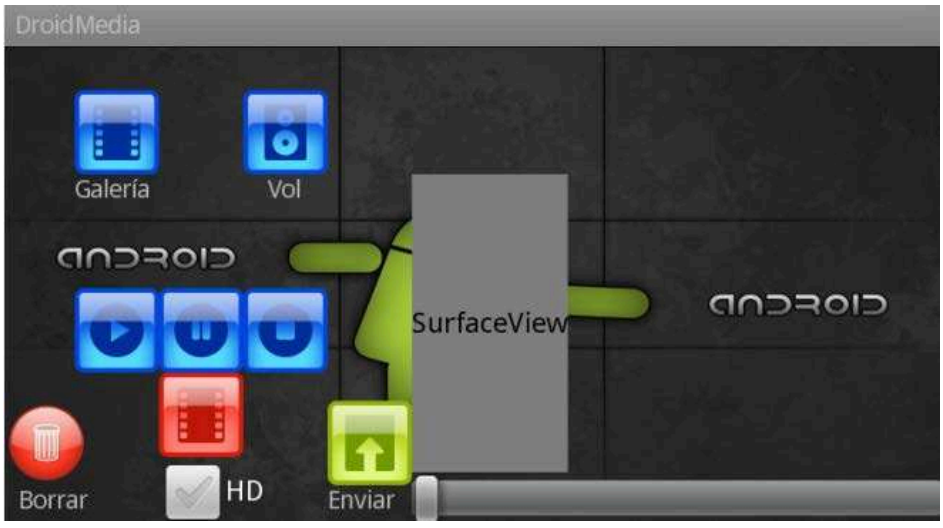


Figura A.9: Modulo de Video

Descripción de los iconos de la pantalla del módulo de video (Figura A.9).

- Ejecuta la galería de Videos permitiendo seleccionar un video previamente almacenado en el dispositivo.

- Permite la captura de video.

- CheckBox HD establece el perfil de captura de video a la mayor calidad posible

- SurfaceView es la superficie que permite la reproducción de video adaptándose al tamaño de pantalla y además al ser presionada ejecuta la pantalla completa.

- Despliega en pantalla el volumen actual del dispositivo.

- Reproduce la canción seleccionada.

- Pone en pausa al MediaPlayer.

- Detiene la reproducción y resetea el MediaPlayer.

- Permite eliminar el archivo seleccionado.

- Permite enviar los archivos multimedia vía Bluetooth y también es posible enviarlos como un adjunto de un *e-mail* o publicar en las redes sociales disponibles en el dispositivo.

- SeekBar permite adelantar o retroceder el archivo multimedia en curso.

ANEXO B

CÓDIGO FUENTE DE LA APLICACIÓN

Código fuente para el módulo de Audio

```
package tesis.media;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.provider.MediaStore;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

public class AudioActivity extends Activity implements SeekBar.OnSeekBarChangeListener {
    /** Called when the activity is first created. */
         private String Name="";
         private MediaRecorder mediaRecorder = null;
         private MediaPlayer mediaPlayer = null;
         private boolean recording=false;
         private AudioManager audio;
         private BluetoothAdapter mBluetoothAdapter = null;
//       private int milis1=0;

         @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.audio);
//                Objetos heredados del sistema operativo Android
        mediaPlayer = new MediaPlayer();
        mediaRecorder = new MediaRecorder();
        audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
//      variables de la pantalla
        final Button btnGallery =(Button)findViewById(R.id.btnGallery);
        final Button btnSend =(Button)findViewById(R.id.btnSend);
        final Button btnPlay =(Button)findViewById(R.id.btnPlay);
        final Button btnRec =(Button)findViewById(R.id.btnRec);
        final Button btnStop =(Button)findViewById(R.id.btnStop);
        final Button btnDel =(Button)findViewById(R.id.btnDel);
        final ImageView iv1 =(ImageView)findViewById(R.id.iv1);
        final Button btnPause =(Button)findViewById(R.id.btnPause);
        final Button btnVol =(Button)findViewById(R.id.btnVol);

        final PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
        final PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "My Tag");
        wl.acquire();

        final SeekBar seekBar = (SeekBar) findViewById(R.id.seekBar);
        seekBar.setOnSeekBarChangeListener(this);

        mediaPlayer.setOnCompletionListener(new OnCompletionListener() {
                        @Override
                        public void onCompletion(MediaPlayer mp) {
```

```
//                                  mensaje("Fin de Reprodución");

                                }
                        });

//      Aciones ON CLIK LISTENER
//      Boton Volumen
        btnVol.setOnClickListener(new View.OnClickListener() {
                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub
                                        audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                                        AudioManager.ADJUST_SAME, AudioManager.FLAG_SHOW_UI);
                                }
                        });
//      Boton Pausar
        btnPause.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // TODO Auto-generated method stub
                                        if(mediaPlayer.isPlaying()){
                                                mediaPlayer.pause();
                                        }
                                        else{
                                                mensaje("No se encuentra Reproduciendo ningún elemento");
                                        }
                                }
                        });
//      Boton Eliminar
        btnDel.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // TODO Auto-generated method stub
                                        if (Name!=""){
                                                alerta();

                                        }
                                        else{
                                                mensaje("No ha seleccionado ningún archivo de Sonido!");
                                        }
                                }
                        });

//      Boton Stop
        btnStop.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub
                                        if (recording) {
                                recording = false;
                                mediaRecorder.stop();
                                mediaRecorder.reset();
                                iv1.setVisibility(4);   //Invisible icono de grabando
                                btnRec.setEnabled(true);
                                btnStop.setEnabled(true);
                                btnPlay.setEnabled(true);
                                btnSend.setEnabled(true);
                                btnPause.setEnabled(true);
                                btnDel.setEnabled(true);
                                btnGallery.setEnabled(true);
                                /**
                                 * Si se esta reproduciendo, detenemos la reproduccion y reiniciamos la configuracion
                                 */
                        } else if (mediaPlayer.isPlaying()) {
                                mediaPlayer.stop();
                                mediaPlayer.reset();
                        }
                                }
                        });

//      boton  Grabar
        btnRec.setOnClickListener(new View.OnClickListener() {
                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub
                                        if(mediaPlayer.isPlaying()){
                                                mediaPlayer.reset();
                                                Name="";
                                        }
```

```
                                btnRec.setEnabled(false);
                        btnStop.setEnabled(true);
                        btnPlay.setEnabled(false);
                        btnSend.setEnabled(false);
                        btnPause.setEnabled(false);
                        btnDel.setEnabled(false);
                        btnGallery.setEnabled(false);
                                prepareRecorder();
                                File file=getOutputMediaFile();
                                Name=file.getAbsolutePath();
                                displayTxt(Name);
                                mediaRecorder.setOutputFile(Name);
                                try {
                        /**
                         * Una vez configurado todo llamamos al metodo prepare que deja todo listo
                         * para iniciar la grabacion
                         */
                                mediaRecorder.prepare();
                        } catch (IllegalStateException e) {
                        } catch (IOException e) {
                        }

                        mediaRecorder.start();
                        iv1.setVisibility(0);
                        new MediaScannerConnectionClient() {
                                private MediaScannerConnection msc = null; {
                                        msc = new MediaScannerConnection(getApplicationContext(), this);
msc.connect();
                                }
                                public void onMediaScannerConnected() {

                                        msc.scanFile(Name, null);

                                }
                                public void onScanCompleted(String path, Uri uri) {
                                        msc.disconnect();
                                }
                        };
                        recording = true;
                        }
                });

//        Boton Reproducir
        btnPlay.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                if (Name!=""){
                                        try {

                                mediaPlayer.setDataSource(Name);
                                mediaPlayer.prepare();
                        } catch (IllegalStateException e) {
                        } catch (IOException e) {
                        }
                                        if(mediaPlayer.isPlaying()){
                                                return;
                                        }

                                        mediaPlayer.start();
                                }else{
                                        mensaje("No ha seleccionado ningún archivo de Sonido!");

                                }

                        }
                });

//       Boton Enviar
        btnSend.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
                                if (mBluetoothAdapter != null) {
                                if (Name!="" ){
                                Uri audioUri = Uri.fromFile(new File(Name));
                                 Intent intent1 = new Intent();
                                intent1.setAction(Intent.ACTION_SEND);
                                intent1.setType("audio/*");
                                intent1.putExtra(Intent.EXTRA_STREAM,audioUri);
                                startActivity(intent1);
                                 }else {
```

```
                                        mensaje("No ha seleccionado ningún archivo de Sonido!");
                                }
                                }else{
                                        mensaje("Bluetooth no disponible");
                                }
                        }
                });

// boton Galeria
        btnGallery.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                if(mediaPlayer.isPlaying()){
                                        mediaPlayer.stop();
                                }
                                String path = "";
              path = Environment.getExternalStorageDirectory().getAbsolutePath();
              Intent intent1= new Intent(path);
              intent1.setType("audio/*");
              intent1.setAction(Intent.ACTION_GET_CONTENT);
              intent1.addCategory(Intent.CATEGORY_OPENABLE);
              startActivityForResult(Intent.createChooser(intent1, "Select Music"), 1);

                        // TODO Add extras or a data URI to this intent as appropriate.

                        }
                });

         mediaPlayer.setOnPreparedListener(new OnPreparedListener()
        {
          @Override
          public void onPrepared(final MediaPlayer mp)
          {
           if (Name!=""){
              seekBar.setMax(mp.getDuration());
                 new Thread(new Runnable() {
                         @Override
                         public void run() {
                                 while(mp!=null && mp.getCurrentPosition()<mp.getDuration())
                                 {
//
                                     seekBar.setProgress(mp.getCurrentPosition());

                                     try {
                                         Thread.sleep(1000);
                                     }
                                     catch (InterruptedException e) {
                                        e.printStackTrace();
                                     }


                                 }
                         }

                 }).start();

           }
          }
      });

    }

//       Metodos

          @Override
          public void onProgressChanged(SeekBar arg0, int arg1, boolean arg2) {

          }

          @Override
                  public void onStartTrackingTouch(SeekBar arg0) {


          }

          @Override
          public void onStopTrackingTouch(SeekBar arg0) {
                 // TODO Auto-generated method stub
                   int milis=arg0.getProgress();
                   if (Name!="")
                   mediaPlayer.seekTo(milis);
          }
```

```
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                  super.onActivityResult(requestCode, resultCode, data);
                 if (requestCode ==1 && data!=null  ){
                          Uri audioU = data.getData();
                 Name =getRealPathFromURI(audioU);
                 displayTxt(Name);
                 mediaPlayer.reset();
                 try {
                          mediaPlayer.setDataSource(Name);
                          mediaPlayer.prepare();
                 } catch (IllegalStateException e) {
                 } catch (IOException e) {
                 }
                          mediaPlayer.start();
                 }
        }
        /**
                  *
                  */
        public void prepareRecorder(){

                          mediaRecorder.setAudioSource(MediaRecorder.AudioSource.DEFAULT);
                          mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
                          mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
                 }
        public void mensaje(CharSequence text){
                          Context context = getApplicationContext();
                          int duration = Toast.LENGTH_SHORT;
                          Toast toast = Toast.makeText(context, text, duration);
                          toast.show();
                 }
        public String getRealPathFromURI(Uri contentUri) {
                 String[] proj = { MediaStore.Audio.Media.DATA };
                 Cursor cursor = managedQuery(contentUri, proj, null, null, null);
                 int column_index = cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DATA);
                 cursor.moveToFirst();
                 return cursor.getString(column_index);
             }
        public File getOutputMediaFile(){

                  String state = Environment.getExternalStorageState();
             if (Environment.MEDIA_MOUNTED.equals(state)) {
             File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC), "DroidMediaAudio");

             if (! mediaStorageDir.exists()){
                 if (! mediaStorageDir.mkdirs()){
                     Log.d("DroidMediaAudio", "No se pudo crear el directorio");
                     return null;
                 }
             }

             // Create a media file name
             //       Date date =Date.valueOf("yyyyMMdd_HHmmss");
             String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
             File mediaFile;
             mediaFile = new File(mediaStorageDir.getPath() + File.separator +"A"+ timeStamp + ".AMR");
             return mediaFile;
             }
             mensaje("Inserte SD para grabar!");
             return null;
        }
        // Indica el nombre del archivo seleccionado
        public void displayTxt(String Name){
                 File file=new File(Name);
                 final TextView txtFile=(TextView)findViewById(R.id.txtFile);
                 txtFile.setText(file.getName());
        }

        public void alerta(){
                 AlertDialog.Builder builder = new AlertDialog.Builder(this);
                 builder.setMessage("Esta seguro que desea borrar este archivo?")
                        .setCancelable(false)
                        .setPositiveButton("Si", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                                 if(mediaPlayer.isPlaying()){
                                                          mediaPlayer.reset();
                                                 }
                                File file=new File(Name);
                                boolean deleted = file.delete();
```

```
                                                    if (deleted){
                                                             mensaje("Archivo Eliminado"+Name);
                                                             sendBroadcast(new
Intent(Intent.ACTION_MEDIA_MOUNTED,
                                                           Uri.parse("file://" +
Environment.getExternalStorageDirectory())));
                                                    Name="";
                                                    displayTxt(Name);

                                                    }else {
                                                    mensaje("Archivo protegido por el sistema");
                                            }
                                                    }

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

//Metodo para detectar teclas  presionadas
            @Override
            public boolean onKeyDown(int keyCode, KeyEvent event)
            {
                     switch (keyCode) {
                case KeyEvent.KEYCODE_VOLUME_UP:
                    audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                            AudioManager.ADJUST_RAISE, AudioManager.FLAG_SHOW_UI);
                    return true;
                case KeyEvent.KEYCODE_VOLUME_DOWN:
                    audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                              AudioManager.ADJUST_LOWER, AudioManager.FLAG_SHOW_UI);
                    return true;
                case KeyEvent.KEYCODE_BACK:
                     Thread.setDefaultUncaughtExceptionHandler(null);
                     this.finish();
                              mediaPlayer.release();
                              mediaRecorder.release();
                              return true;
                default:
                    return false;
                     }
            }
}
```

Código fuente para el módulo de imágenes

```
package tesis.media;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.BitmapFactory;
import android.media.MediaScannerConnection;
import android.media.MediaScannerConnection.MediaScannerConnectionClient;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.provider.MediaStore;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

public class ImgActivity extends Activity {
    /** Called when the activity is first created. */
         private String Name="";
```

```
         private BluetoothAdapter mBluetoothAdapter = null;
         @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.img);
//                Objetos heredados del sistema operativo Android

//      variables
        final Button btnGallery =(Button)findViewById(R.id.btnGallery);
        final Button btnSend =(Button)findViewById(R.id.btnSend);
        final Button btnRec =(Button)findViewById(R.id.btnRec);
        final Button btnDel =(Button)findViewById(R.id.btnDel);

        final PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
        final PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "My Tag");
        wl.acquire();
//      Aciones ON CLIK LISTENER
//      Boton Eliminar
        btnDel.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                if (Name!=""){
                                        alerta();
                                }
                                else{
                                        mensaje("No ha seleccionado ninguna Imagen!");
                                }
                        }
                });

//      boton  Capturar
        btnRec.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub

                                File file=getOutputMediaFile();
                                Name=file.getAbsolutePath();
                                displayTxt(Name);
                                Intent intent =  new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                                Uri output = Uri.fromFile(file);
                        intent.putExtra(MediaStore.EXTRA_OUTPUT, output);
                        startActivityForResult(intent,2);

                        }
                });


//       Boton Enviar
        btnSend.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
                                if (mBluetoothAdapter != null) {
                                if (Name!=""){
                                Uri fileUri = Uri.fromFile(new File(Name));
                                Intent intent1 = new Intent();
                               intent1.setAction(Intent.ACTION_SEND);
                               intent1.setType("image/*");
                               intent1.putExtra(Intent.EXTRA_STREAM,fileUri);
                               startActivity(intent1);
                                }else {
                                        mensaje("No ha seleccionado ninguna Imagen!");
                                }
                                }else{
                                        mensaje("Bluetooth no disponible");
                                }

                        }
                });

//                Boton Galeria
        btnGallery.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                           Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                           intent.setType("image/*");
```

```
                                startActivityForResult(intent,1);

                        }
                });
    }

        /**
         * Metodos
         */
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                 super.onActivityResult(requestCode, resultCode, data);
                if (requestCode ==1 && data!=null  ){
                        Uri fileU = data.getData();
                Name =getRealPathFromURI(fileU);
                final ImageView imgV =(ImageView)findViewById(R.id.imgV);
                imgV.destroyDrawingCache();
//                      imgV.setBackgroundResource(R.drawable.img);
                imgV.setImageDrawable(null);
                        imgV.setImageBitmap(BitmapFactory.decodeFile(Name));
                displayTxt(Name);
                }
                else if(requestCode ==2 && data!=null){
                        File file=new File(Name);
                        file.delete();
                        sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED,Uri.parse("file://" +
Environment.getExternalStorageDirectory())));
                        Name="";
                        final ImageView imgV =(ImageView)findViewById(R.id.imgV);
                        imgV.setImageDrawable(null);
                        imgV.setBackgroundResource(R.drawable.img);
                        displayTxt(Name);

                        }
                else if(requestCode ==2&& data==null&& resultCode==RESULT_OK){
                        final ImageView imgV =(ImageView)findViewById(R.id.imgV);
                        imgV.destroyDrawingCache();
//                      imgV.setBackgroundResource(R.drawable.img);
                        imgV.setImageDrawable(null);
                        imgV.setImageBitmap(BitmapFactory.decodeFile(Name));

                                new MediaScannerConnectionClient() {
                                        private MediaScannerConnection msc = null; {
                                                msc = new MediaScannerConnection(getApplicationContext(),
this); msc.connect();
                                        }
                                        public void onMediaScannerConnected() {
                                                msc.scanFile(Name, null);
                                        }
                                        public void onScanCompleted(String path, Uri uri) {
                                                msc.disconnect();
                                        }
                                };
                 }
                }
        public void mensaje(CharSequence text){
                        Context context = getApplicationContext();
                        int duration = Toast.LENGTH_SHORT;
                        Toast toast = Toast.makeText(context, text, duration);
                        toast.show();
                }
//      Extraer el nombre del archivo desde la galeria
        public String getRealPathFromURI(Uri contentUri) {
                String[] proj = { MediaStore.Images.Media.DATA };
                Cursor cursor = managedQuery(contentUri, proj, null, null, null);
                int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
                cursor.moveToFirst();
                return cursor.getString(column_index);
            }
        public File getOutputMediaFile(){

                 String state = Environment.getExternalStorageState();
            if (Environment.MEDIA_MOUNTED.equals(state)) {
            File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), "DroidMediaImagenes");


            if (! mediaStorageDir.exists()){
                if (! mediaStorageDir.mkdirs()){
                    Log.d("DroidMediaImagenes", "No se pudo crear el directorio");
                    return null;
                }
            }
```

```
            String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
            File mediaFile;
            mediaFile = new File(mediaStorageDir.getPath() + File.separator +"I"+ timeStamp + ".JPG");
            return mediaFile;
            }
            mensaje("Inserte SD para grabar!");
            return null;
        }
        // Indica el nombre del archivo seleccionado
        public void displayTxt(String Name){
                File file=new File(Name);
                final TextView txtFile=(TextView)findViewById(R.id.txtFile);
                txtFile.setText(file.getName());
        }

        public void alerta(){
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setMessage("Esta seguro que desea borrar este archivo?")
                       .setCancelable(false)
                       .setPositiveButton("Si", new DialogInterface.OnClickListener() {
                           public void onClick(DialogInterface dialog, int id) {
                             File file=new File(Name);
                             boolean deleted = file.delete();
                                                    if (deleted){
                                                            mensaje("Archivo Eliminado"+Name);

                                                            final ImageView imgV
=(ImageView)findViewById(R.id.imgV);

                                                            imgV.setImageDrawable(null);
                                                            sendBroadcast(new
Intent(Intent.ACTION_MEDIA_MOUNTED,
                                                                  Uri.parse("file://" +
Environment.getExternalStorageDirectory())));
                                                    Name="";
                                                    displayTxt(Name);
                                                        }else {
                                                    mensaje("Archivo protegido por el sistema");
                                                    }
                                                    }

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

//Metodo para detectar teclas  presionadas
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
                switch (keyCode) {

            case KeyEvent.KEYCODE_BACK:
                 this.finish();
                         return true;
            default:
                return false;
                 }
        }
}
```

Código fuente para el modulo de video

```
package tesis.media;

import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
```

```
import android.database.Cursor;
import android.media.AudioManager;
import android.media.CamcorderProfile;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaRecorder;
import android.media.MediaScannerConnection;
import android.media.MediaScannerConnection.MediaScannerConnectionClient;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.PowerManager;
import android.provider.MediaStore;
import android.util.Log;
import android.view.KeyEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

public class VideoActivity extends Activity implements SurfaceHolder.Callback,SeekBar.OnSeekBarChangeListener {
    /** Called when the activity is first created. */
         private String Name="";
         private int milis1=0;
         private MediaRecorder mediaRecorder = null;
         private MediaPlayer mediaPlayer = null;
         private boolean recording=false;
         private AudioManager audio;
         private BluetoothAdapter mBluetoothAdapter = null;

         @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.video);
//                Objetos heredados del sistema operativo Android
        mediaPlayer = new MediaPlayer();
        mediaRecorder = new MediaRecorder();
        audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
//      variables

        final PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
        final PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "My Tag");
        wl.acquire();

        final SurfaceView surface = (SurfaceView)findViewById(R.id.surface);
        final SurfaceHolder holder = surface.getHolder();
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        holder.addCallback(this);

        mediaRecorder.setPreviewDisplay(holder.getSurface());
        mediaPlayer.setDisplay(holder);

//      Inicializacion de los Botones
        final Button btnGallery =(Button)findViewById(R.id.btnGallery);
        final Button btnSend =(Button)findViewById(R.id.btnSend);
        final Button btnPlay =(Button)findViewById(R.id.btnPlay);
        final Button btnRec =(Button)findViewById(R.id.btnRec);
        final Button btnStop =(Button)findViewById(R.id.btnStop);
        final Button btnDel =(Button)findViewById(R.id.btnDel);
        final ImageView iv1 =(ImageView)findViewById(R.id.iv1);
        final Button btnPause =(Button)findViewById(R.id.btnPause);
        final Button btnVol =(Button)findViewById(R.id.btnVol);

        final SeekBar seekBar = (SeekBar) findViewById(R.id.seekBar);
        seekBar.setOnSeekBarChangeListener(this); //Para utilizar metodos de seek bar de esta clase

// Entrar en full screen
        surface.setOnClickListener(new View.OnClickListener() {
                                    @Override
                                    public void onClick(View arg0) {
                                            if (Name!="" &&!recording){
                                            Intent intent = new Intent(VideoActivity.this,fullScreen.class);
                                            Bundle bundle = new Bundle();
                                            int progress=seekBar.getProgress();
                                            bundle.putString("Archivo", Name);
                                            bundle.putInt("Progress",progress);
                                            intent.putExtras(bundle);
```

```
                                        Thread.setDefaultUncaughtExceptionHandler(null);
                                        mediaPlayer.release();
                                        startActivityForResult(intent,2);

                                        }
                                }
                        });

//        Detectar cuando ha finalizado la reproduccion
        mediaPlayer.setOnCompletionListener(new OnCompletionListener() {
                                @Override
                                public void onCompletion(MediaPlayer mp) {
//                                      mensaje("Fin de Reprodución");
//                                      mp.reset();

                                }
                        });

        mediaPlayer.setOnPreparedListener(new OnPreparedListener()
        {
          @Override
          public void onPrepared(final MediaPlayer mp)
          {
           if (Name!=""){
              seekBar.setMax(mp.getDuration());
                 new Thread(new Runnable() {

                        @Override
                        public void run() {
                                while(mp!=null && mp.getCurrentPosition()<mp.getDuration())
                                {
                                    seekBar.setProgress(mp.getCurrentPosition());
//
                                     try {
                                         Thread.sleep(1000);
                                     }
                                     catch (InterruptedException e) {
                                      return;
                                     } catch (Exception e) {
                                      return;
                                     }
                                }
                        }
                }).start();
           }
          }
      });

//      Aciones ON CLIK LISTENER
//      Boton Volumen
        btnVol.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub
                                        audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                                        AudioManager.ADJUST_SAME, AudioManager.FLAG_SHOW_UI);
                                }
                        });

//      Boton Pausar
        btnPause.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // TODO Auto-generated method stub
                                        if(mediaPlayer.isPlaying()){
                                                mediaPlayer.pause();

                                        }
                                        else{
                                                mensaje("No se encuentra Reproduciendo ningún elemento");
                                        }
                                }
                        });


//      Boton Eliminar
        btnDel.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View v) {
                                        // TODO Auto-generated method stub
```

```
                                        if (Name!=""){
                                                alerta();

                                        }
                                        else{
                                                mensaje("No ha seleccionado ningún archivo de Video!");
                                        }
                                }
                        });

//      Boton Stop
        btnStop.setOnClickListener(new View.OnClickListener() {

                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub

                                        if (recording) {
                                recording = false;
                                mediaRecorder.stop();
                                mediaRecorder.reset();
                                iv1.setVisibility(4);
                                btnRec.setEnabled(true);
                                btnStop.setEnabled(true);
                                btnPlay.setEnabled(true);
                                btnSend.setEnabled(true);
                                btnPause.setEnabled(true);
                                btnDel.setEnabled(true);
                                btnGallery.setEnabled(true);
                                /**
                                 * Si se esta reproduciendo, detenemos la reproduccion y reiniciamos la configuracion
                                 */
                        } else if (mediaPlayer.isPlaying()) {
                                mediaPlayer.stop();
                                mediaPlayer.reset();
                        }
                                }
                        });

//      boton  Grabar
        btnRec.setOnClickListener(new View.OnClickListener() {
                                @Override
                                public void onClick(View arg0) {
                                        // TODO Auto-generated method stub
                                        if(mediaPlayer.isPlaying()){
                                                mediaPlayer.reset();
                                                Name="";
                                        }

                                        btnRec.setEnabled(false);
                                btnStop.setEnabled(true);
                                btnPlay.setEnabled(false);
                                btnSend.setEnabled(false);
                                btnPause.setEnabled(false);
                                btnDel.setEnabled(false);
                                btnGallery.setEnabled(false);
                                        prepareRecorder();
                                        File file=getOutputMediaFile();
                                        Name=file.getAbsolutePath();
                                        displayTxt(Name);
                                        mediaRecorder.setOutputFile(Name);
                                        try {
                                        mediaRecorder.prepare();
                                } catch (IllegalStateException e) {
                                } catch (IOException e) {
                                }

                                mediaRecorder.start();
                                iv1.setVisibility(0);
                                new MediaScannerConnectionClient() {
                                        private MediaScannerConnection msc = null; {
                                                msc = new MediaScannerConnection(getApplicationContext(), this);
msc.connect();
                                        }
                                        public void onMediaScannerConnected() {

                                                msc.scanFile(Name, null);

                                        }
                                        public void onScanCompleted(String path, Uri uri) {
                                                msc.disconnect();
                                        }
```

```
                        };
                        recording = true;
                        }
                });

//        Boton Reproducir
        btnPlay.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                if (Name!=""){

                                        try {
                                mediaPlayer.setDataSource(Name);
                                mediaPlayer.prepare();
                        } catch (IllegalStateException e) {
                        } catch (IOException e) {
                        }
//                                      if(mediaPlayer.isPlaying()){
//                                              return;
//                                      }

                                      mediaPlayer.start();
                                }else{
                                      mensaje("No ha seleccionado ningún archivo de Video!");

                                }

                        }
                });

//       Boton Enviar
        btnSend.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
                                if (mBluetoothAdapter != null) {
                                if (Name!=""){
                                Uri fileUri = Uri.fromFile(new File(Name));
                                Intent intent1 = new Intent();
                               intent1.setAction(Intent.ACTION_SEND);
                               intent1.setType("video/*");
                               intent1.putExtra(Intent.EXTRA_STREAM,fileUri);
                               startActivity(intent1);
                                }else {
                                      mensaje("No ha seleccionado ningún archivo de Video!");
                                }
                                }else{
                                      mensaje("Bluetooth no disponible");
                                }
                        }
                });

// boton Galeria
        btnGallery.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                if(mediaPlayer.isPlaying()){
                                      mediaPlayer.stop();
                                }
                           Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
                           intent.setType("video/*");
                           startActivityForResult(intent,1);

                        // TODO Add extras or a data URI to this intent as appropriate.

                        }
                });

    }

          @Override
          public void onProgressChanged(SeekBar arg0, int arg1, boolean arg2) {

          }

          @Override
                  public void onStartTrackingTouch(SeekBar arg0) {
                 // TODO Auto-generated method stub
          }
```

```
         @Override
         public void onStopTrackingTouch(SeekBar arg0) {
                // TODO Auto-generated method stub
                  int milis=arg0.getProgress();
                  if (Name!=""){
                  mediaPlayer.seekTo(milis);}
         }

        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                  super.onActivityResult(requestCode, resultCode, data);
                 if (requestCode ==1 && data!=null  ){
                          Uri fileUri = data.getData();
                 Name =getRealPathFromURI(fileUri);
                 displayTxt(Name);
                 }else if (requestCode ==2 && data!=null){
//                          Name=data.getStringExtra("Archivo");
//                          int milis1=data.getIntExtra("Progreso",milis);
//                          mediaPlayer.seekTo(milis1);
                          Bundle bundle = data.getExtras();
                          Name=bundle.getString("Archivo");
                          milis1=bundle.getInt("Progreso");



//                Name=bundle.getString("Archivo");
                 }
         }


         /**
                  *
                  */
         public void prepareRecorder(){
                  mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                  mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
                  CheckBox chkHd=(CheckBox)findViewById(R.id.chkHd);
                  if (chkHd.isChecked()) {
                          mediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH));
        }else{

                  mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
                  mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
                  mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
        }
         }
         public void mensaje(CharSequence text){
                          Context context = getApplicationContext();
//                        CharSequence text = "No ha seleccionado ningún archivo de Sonido!";
                          int duration = Toast.LENGTH_SHORT;
                          Toast toast = Toast.makeText(context, text, duration);
                          toast.show();
                 }
         public String getRealPathFromURI(Uri contentUri) {
                 String[] proj = { MediaStore.Video.Media.DATA };
                 Cursor cursor = managedQuery(contentUri, proj, null, null, null);
                 int column_index = cursor.getColumnIndexOrThrow(MediaStore.Video.Media.DATA);
                 cursor.moveToFirst();
                 return cursor.getString(column_index);
             }
         public File getOutputMediaFile(){

                  String state = Environment.getExternalStorageState();
             if (Environment.MEDIA_MOUNTED.equals(state)) {
             File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MOVIES), "DroidMediaVideos");

             if (! mediaStorageDir.exists()){
                 if (! mediaStorageDir.mkdirs()){
                     Log.d("DroidMediaVideos", "No se pudo crear el directorio");
                     return null;
                 }
             }

             // Create a media file name
             //       Date date =Date.valueOf("yyyyMMdd_HHmmss");
             String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
             File mediaFile;
             mediaFile = new File(mediaStorageDir.getPath() + File.separator +"V"+ timeStamp + ".mp4");
             return mediaFile;
             }
             mensaje("Inserte SD para grabar!");
```

```
            return null;
        }
        // Indica el nombre del archivo seleccionado
        public void displayTxt(String Name){
                File file=new File(Name);
                final TextView txtFile=(TextView)findViewById(R.id.txtFile);
                txtFile.setText(file.getName());
        }

        public void alerta(){
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setMessage("Esta seguro que desea borrar este archivo?")
                       .setCancelable(false)
                       .setPositiveButton("Si", new DialogInterface.OnClickListener() {
                           public void onClick(DialogInterface dialog, int id) {

                             File file=new File(Name);
                                                 if (file.delete()){
                                                         if(mediaPlayer.isPlaying()){
                                                                 mediaPlayer.reset();
                                                         }
                                                         mensaje("Archivo Eliminado"+Name);

                                                         sendBroadcast(new
Intent(Intent.ACTION_MEDIA_MOUNTED,
                                                               Uri.parse("file://" +
Environment.getExternalStorageDirectory())));
                                                         Name="";
                                                 displayTxt(Name);
                                                 }else {
                                                         mensaje("Archivo protegido por el sistema");
                                                         }
                                                 }

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

//Metodo para detectar teclas  presionadas
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
                 switch (keyCode) {
            case KeyEvent.KEYCODE_VOLUME_UP:
                audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                        AudioManager.ADJUST_RAISE, AudioManager.FLAG_SHOW_UI);
                return true;
            case KeyEvent.KEYCODE_VOLUME_DOWN:
                audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                           AudioManager.ADJUST_LOWER, AudioManager.FLAG_SHOW_UI);
                return true;
            case KeyEvent.KEYCODE_BACK:
                 Thread.setDefaultUncaughtExceptionHandler(null);
                           mediaPlayer.release();
                           mediaRecorder.release();

                           this.finish();
                           return true;
            default:
                return false;
                 }
        }

        @Override
        public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

        }
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
                 if (mediaRecorder == null) {
                           mediaRecorder = new MediaRecorder();
                           mediaRecorder.setPreviewDisplay(holder.getSurface());
                 }

                 if (mediaPlayer == null) {
                           mediaPlayer = new MediaPlayer();
                           mediaPlayer.setDisplay(holder);
```

```
                }

                if (Name!=""){
                        mediaPlayer.setDisplay(holder);
                        displayTxt(Name);
                        mediaPlayer.reset();
                try {
                        mediaPlayer.setDataSource(Name);
                        mediaPlayer.prepare();

                } catch (IllegalStateException e) {
                } catch (IOException e) {
                }
                mediaPlayer.seekTo(milis1);
                mediaPlayer.start();
                }
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder arg0) {

        }
}
```

Codigo fuente para el Menu Principal

```
package tesis.media;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.PowerManager;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

public class DroidMediaActivity extends Activity {
    /** Called when the activity is first created. */


        @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);


    final Button btnAudio =(Button)findViewById(R.id.btnAudio);
    final Button btnVideo =(Button)findViewById(R.id.btnVideo);
    final Button btnImg =(Button)findViewById(R.id.btnImg);
    final ImageView imageView1 =(ImageView)findViewById(R.id.imageView1);

    final PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
    final PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "My Tag");
    wl.acquire();

//      Créditos
    imageView1.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        mensaje("CREADO POR: DIEGO ARGÜELLO; ESCUELA POLITÉCNICA NACIONAL QUITO-ECUADOR
2012");
                }
        });


//    Boton Imagenes

    btnImg.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                                                Intent intent = new
Intent(DroidMediaActivity.this,ImgActivity.class);
                                                startActivity(intent);
                }
```

```
        });


// Boton Video
    btnVideo.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        Intent intent = new Intent(DroidMediaActivity.this,VideoActivity.class);
                        startActivity(intent);
                }
        });


//    Boton Audio
    btnAudio.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        // TODO Auto-generated method stub
                        Intent intent = new Intent(DroidMediaActivity.this,AudioActivity.class);
                        startActivity(intent);
                }
        });

    }
//      Metodo de mensajes en pantalla
        public void mensaje(CharSequence text){
                Context context = getApplicationContext();
                int duration = Toast.LENGTH_SHORT;
                Toast toast = Toast.makeText(context, text, duration);
                toast.setGravity(0, 0, 150);
                toast.show();

        }

//      Metodo Teclas presionadas

        public boolean onKeyDown(int keyCode, KeyEvent event)
        {
                if(keyCode==KeyEvent.KEYCODE_BACK)
                {
                        this.finish();
                        android.os.Process.killProcess(android.os.Process.myPid());
                }

        return true;

        }
```

ANEXO C

INSTALADOR DE LA APLICACIÓN PARA DISPOSITIVOS ANDROID

(El contenido del anexo se presenta en el CD adjunto)

ANEXO D

MANUALES DE PROGRAMACIÓN PARA ANDROID

(El contenido del anexo se presenta en el CD adjunto)