Universidad de las Ciencias Informáticas

Facultad 1

Título: Proceso para crear personalizaciones del Sistema Operativo Android

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Luis Daniel Sierra Corredera

Tutores: MSc. Dariem Pérez Herrera

Ing. Ernesto Puente Fuentes

La Habana, 2013

Declaración de autoría

Declaramos por este medio ser autores del trabajo final de tesis Proceso para crear personalizaciones del sistema operativo Android, y que autorizamos a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría en la Ciudad de La Habana a los ___ días del mes de _____ del año _____.

_____

Luis Daniel Sierra Corredera

Firma del Autor

_____

MSc. Dariem Pérez Herrera

Firma del Tutor

_____

Ing. Ernesto Puente Fuentes

Firma del Tutor

"(…) la soberanía tecnológica que propugnamos implica la apropiación y dominio de las tecnologías y conocimientos de una forma integral…"

Comandante de la Revolución
Ramiro Valdés Menéndez

Agradecimientos

El más grande agradecimiento es para mi familia, a todos gracias por el apoyo que me han brindado durante el transcurso de mi vida personal y como estudiante. Un agradecimiento muy especial al Juco, la Juca y Tico que me han dado todo lo que está a su alcance, que siempre me motivaron en los estudios lo que me permitió ser un mejor estudiante; ellos son mi guía y mi razón de ser.

Agradecer además a Yudaimy mi “ñiña”, por ser mi compañera en estos últimos tres años de la universidad, por permanecer a mi lado en los momentos buenos y malos dándome su apoyo cuando más lo necesité.

Gracias a los profesores que han contribuido en mi preparación como estudiante y futuro profesional.

Un agradecimiento especial a Ernesto Puente por guiarme desde el primer año de estudio, ser un buen tutor y más que tutor compañero.

Gracias a mis amigos y compañeros con los que he compartido tantos momentos de la vida. Muchas gracias a Ramón y José Miguel por compartir mis alegrías en las competencias de programación como integrantes del equipo NovaTux. Gracias al equipo Orient por permitirme tener uno de mis mayores logros en la UCI durante el Regional Caribeño de la ACM-ICPC de 2012.

Mi más profundo agradecimiento a la Revolución Cubana y todos los que la hicieron posible, por permitirme crecer como un hombre libre y con tantas posibilidades.

De manera general, agradecer a todos los que de una forma u otra me han brindado su compañía y sabiduría.

Dedicatoria

Especialmente dedicado a mi papá, mi mamá y mi hermano, las personas que más quiero en la vida. A ellos debo mi educación y preparación. Sin su apoyo y preocupación nunca hubiese sido posible cumplir con este sueño que hoy se hace realidad.

Resumen

Actualmente el proyecto Nova perteneciente al Centro de Soluciones Libres de la Universidad de las Ciencias Informática se da a la tarea de portar el sistema operativo Android para la arquitectura de computadoras Intel de 32 bits, lo cual permitirá la asimilación e integración de esta tecnología en futuras soluciones nacionales. Sin embargo, luego de varios intentos de crear soluciones a la medida basadas en Android se detecta la ausencia de un mecanismo que oriente el desarrollo de esta actividad y a su vez agrupe el conocimiento necesario para esto. Es por esto que se pretende crear un proceso que guie el desarrollo de soluciones a la medida basadas en Android, a través de la revisión de bibliografías, código fuente y la implementación de casos de estudio. Para dar solución a la problemática planteada se definió el proceso de personalización de Android. Además, se realizó la implementación de casos de estudio, para comprobar la total funcionalidad del proceso definido y demostrar un conjunto de aspectos importantes a la hora de crear aplicaciones para el mismo. Todo esto se llevó a cabo utilizando herramientas y tecnologías establecidas por el proyecto.

Palabras clave: android, aplicaciones, arquitectura, proceso, sistema operativo, tecnología.

Índice

Índice de figuras

Índice de tablas

Introducción

La tecnología móvil es un mercado muy dinámico, donde constantemente hay cambios y nuevos avances tanto en los dispositivos como en las plataformas (1). Entre las tecnologías móviles, el sistema operativo más difundido y con un rápido crecimiento es Android, además tiene una gran cantidad de aplicaciones que pueden ser descargadas por los usuarios desde Google Play.

El creciente desarrollo de esta tecnología, aparejado con la computación en la nube y la miniaturización de los dispositivos informáticos ya permite realizar análisis sobre el destino de las computadoras de escritorio convencionales (1).

Pero Cuba como caso particular, debido al bloqueo económico impuesto por Estados Unidos de América, se ha visto inmersa en un atraso tecnológico. Razón por la cual no puede prescindir de las tecnologías que actualmente posee, ya que sería un proceso muy costoso la adquisición de tecnologías de punta. En cambio, sí se puede aspirar a adaptar los sistemas más actuales a estos dispositivos.

En la Universidad de las Ciencias Informáticas radica el proyecto Nova perteneciente al Centro de Soluciones Libres (CESOL) el cual se dedica fundamentalmente a la creación de la Distribución Cubana de GNU/Linux Nova. Actualmente este proyecto se da a la tarea de portar el sistema operativo Android para la arquitectura de computadoras Intel de 32 bits en lo adelante i386, lo cual permitirá la asimilación e integración de esta tecnología en futuras soluciones nacionales. Sin embargo, luego de varios intentos de crear soluciones a la medida basadas en Android, se detecta la ausencia de un mecanismo que oriente el desarrollo de esta actividad y a su vez, agrupe el conocimiento necesario para esto.

Partiendo de esta problemática se plantea como **problema científico** de la presente investigación: en el proyecto Nova no existe el mecanismo que oriente el desarrollo de soluciones a la medida basadas en Android.

Definiéndose como **objeto de estudio** de la investigación el proceso de construcción de sistemas operativos para la arquitectura de computadoras i386, ubicando como **campo de acción** la personalización del sistema operativo Android para la arquitectura de computadoras i386.

Se plantea como **idea a defender** que la definición de un proceso que guie la creación de soluciones a la medida basadas en Android en el proceso de desarrollo de Nova, podría permitir la asimilación e integración de esta tecnología en futuras soluciones autóctonas. Basados en esta idea se define como **objetivo general** de esta investigación: crear un proceso que guie el desarrollo de soluciones a la medida basadas en Android, a través de la revisión de bibliografías, código fuente y la implementación de casos de estudio.

Para dar solución al mismo se definen los siguientes **objetivos específicos**:

- ✓ Sistematizar la arquitectura de Android.
- ✓ Sistematizar el proceso de construcción de Android.
- ✓ Definir el proceso de personalización de Android.
- ✓ Implementar casos de estudio.

Para dar cumplimiento a los objetivos específicos mencionados con anterioridad se plantean las siguientes **tareas de investigación**:

- ✓ Revisión bibliográfica para analizar la arquitectura y componentes de Android.
- ✓ Estudio del código fuente del núcleo y principales librerías de Android para comprender el funcionamiento e integración de sus componentes.
- ✓ Definición del proceso de personalización de Android.
- ✓ Aplicación de los conocimientos adquiridos en casos de estudio.

En el desarrollo de la presente investigación se utilizarán un conjunto de **métodos científicos** que servirán de guía y facilitarán un mejor entendimiento de lo que está sucediendo, como es el caso de los métodos empíricos y los teóricos.

En el conjunto de métodos empíricos se encuentra la Observación el cual se detalla a continuación.

- ✓ Observación: permitió hacer un análisis del comportamiento de algunas herramientas que apoyan el proceso de construcción de Android y el proceso de compilación de aplicaciones.

Como parte de los métodos teóricos se encuentran el Analítico Sintético y el Histórico Lógico los cuales fueron de mucha utilidad.

- ✓ Analítico Sintético: se realizó un análisis de documentos, libros, artículos. Se dividió el problema en pequeños problemas y se realizó un mejor análisis, para de esta manera interpretar mejor los resultados obtenidos en cada tarea realizada.
- ✓ Histórico Lógico: se analizó la evolución de diferentes teorías y herramientas que apoyan en la construcción de estos sistemas, mediante la búsqueda en diferentes bibliografías.

El presente trabajo está compuesto por un resumen, una introducción, tres capítulos, el contenido de los cuales se describe a continuación.

Capítulo 1: Fundamentación teórica de la investigación, se hace un estudio sobre las características de los sistemas operativos móviles actuales, relevancia y estadísticas de Android sobre los demás sistemas operativos móviles, además se definen los métodos y herramientas a tener en cuenta para el proceso de desarrollo de soluciones basadas en Android.

Capítulo 2: Proceso de configuración y construcción de Android-x86, se define un proceso que guie la creación de soluciones a la medida basadas en Android, además, se describe cada una de las actividades que componen dicho proceso.

Capítulo 3: Implementación de casos de estudio, se describen los resultados de la aplicación del proceso definido a varios casos de estudio para diferentes escenarios.

Capítulo 1: Fundamentación teórica de la investigación

1.1 Introducción

Este capítulo contiene los principales conceptos necesarios para el entendimiento de las etapas por las que transitará el proceso de construcción del sistema. Se aborda también sobre las características del sistema operativo Android y se hace una pequeña reseña sobre el proyecto Android-x86. Además, se especifican los métodos y herramientas con las que se trabajará durante todo el proceso de personalización de Android.

1.2 Sistemas operativos móviles

Un sistema operativo móvil es un sistema operativo específicamente diseñado para controlar dispositivos móviles tales como teléfonos móviles, teléfonos inteligentes, computadoras de tableta y otros dispositivos portátiles o de mano (2). Sin embargo, un sistema operativo móvil es bastante más simple que un sistema operativo convencional para computadoras y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes formas de interacción con los usuarios (3).

En la actualidad son varios los sistemas operativos móviles existentes, algunos libres y otros propietarios. Entre los principales sistemas operativos móviles encontramos Android, IOS, Symbian, RIM, Bada, Windows Phone. A continuación se muestra una tabla con el porcentaje de uso de estos sistemas en los dispositivos móviles actuales, en la cual el lector podrá comprobar la tendencia de Android a resaltar sobre otros sistemas operativos móviles (ver *tabla 1*) (4) (5) (6).

Tabla 1: Uso de los sistemas operativos móviles en el 1er cuarto de año del 2011 y el 2012.

Sistema operativo	1er Cuarto de 2011		1er Cuarto de 2012	
	Miles	%	Miles	%
Android	36 350,1	36,43	81 067,4	56,14
IOS	16 883,2	16,92	33 120,5	22,92
Symbian	27 598,5	27,66	12 466,9	8,63
RIM	13 004,0	13,03	9 939,3	6,88
Bada	1 862,2	1,87	3 842,2	2,66

Windows Phone	2 582,1	2,59	2 712,5	1,88
Otros	1 495,0	1,50	1 242,9	0,86

Para una mejor interpretación visual, la información de la tabla anterior se muestra en la siguiente gráfica de columnas (ver *figura 1*).

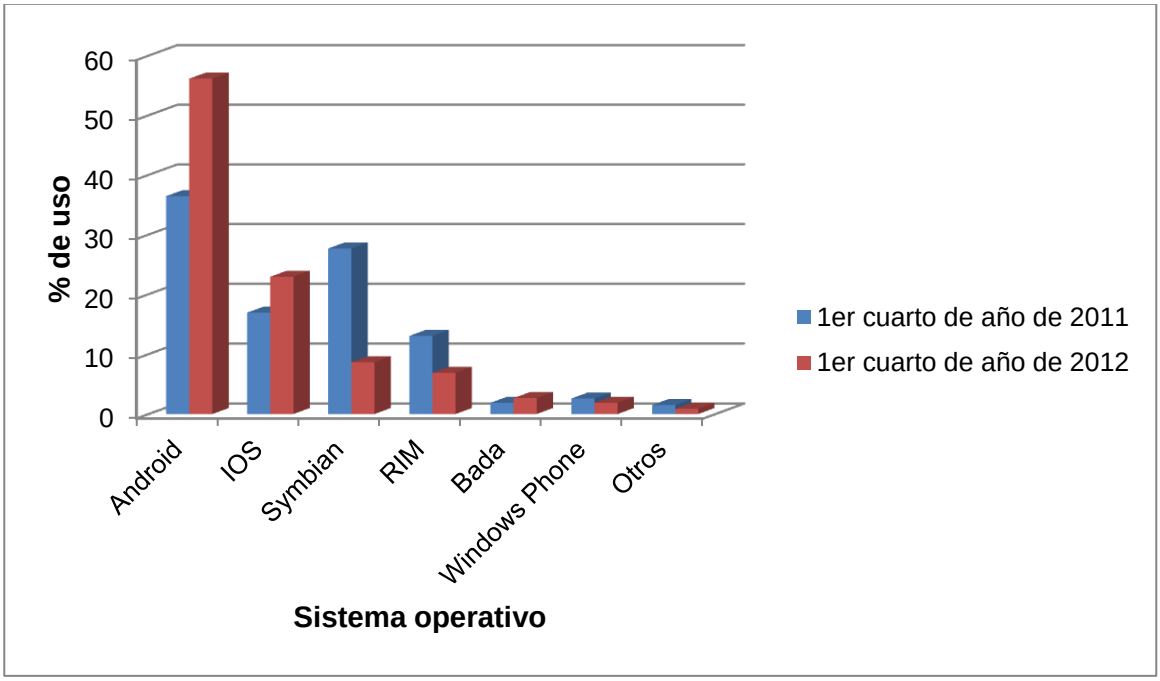


Figura 1: Uso de los sistemas operativos móviles en el 1er cuarto de año del 2011 y el 2012.

1.3 Sistema operativo Android

Android además de una plataforma móvil es un sistema operativo de código abierto basado en el núcleo Linux, diseñado en un principio para dispositivos móviles. Permite controlar dispositivos por medio de bibliotecas desarrolladas o adoptadas por Google mediante el lenguaje de programación Java (7).

Inicialmente Android fue desarrollado por Google Inc. aunque poco después se unió Open Handset Alliance, un consorcio de 84 compañías de hardware, software y telecomunicaciones, las cuales llegaron a un acuerdo para promocionar los estándares de código abierto para dispositivos móviles y acelerar la innovación móvil, ofreciendo a los usuarios una experiencia móvil más rica y barata (8).

Google sin embargo, es quien ha publicado la mayoría del código fuente de Android bajo la licencia de software Apache, una licencia de software libre y de código abierto a cualquier desarrollador (7).

Con esta plataforma móvil se pueden usar todas las aplicaciones de Google conocidas. Además, hay aproximadamente 600 000 aplicaciones y juegos disponibles en Google Play para mantener a los usuarios interesados, junto con millones de canciones y libros, y cientos de películas. Los dispositivos con Android son sistemas inteligentes con características que no se pueden encontrar en ninguna otra plataforma móvil (9).

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un *framework* (marco de trabajo) Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución (10).

La máquina virtual Dalvik se encuentra en la capa de ejecución y ha sido diseñada para optimizar el uso de la memoria y el hardware en el entorno de los dispositivos móviles. Esta máquina virtual ejecuta archivos compilados en el formato *Dalvik Executable* (*.dex), un formato optimizado para el almacenamiento eficiente y ejecución en memoria. El hecho de que corra sobre un núcleo Linux le permite delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel. Otra característica de Dalvik es que ha sido optimizada para que múltiples instancias de ella puedan funcionar al mismo tiempo, esto con el objetivo de proteger las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma las demás (11).

Algunas otras características son:

- ✓ Código abierto.
- ✓ Núcleo basado en Linux.
- ✓ Adaptable a muchas pantallas y resoluciones.
- ✓ Utiliza una base de datos relacional SQlite para el almacenamiento de datos.
- ✓ Ofrece diferentes formas de mensajería.
- ✓ Navegador web basado en el motor de código abierto WebKit.
- ✓ Soporte de Java y muchos otros formatos multimedia (audio, imágenes y video).
- ✓ Máquina virtual Dalvik, con llamadas a instancias muy similar a Java.
- ✓ Soporte de HTML, HTML5, Adobe Flash Player.

- ✓ Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis de rendimiento de software.
- ✓ Catálogo de aplicaciones gratuitas y propietarias que pueden ser descargadas e instaladas directamente desde Google Play.
- ✓ Telefonía GSM dependiente del terminal.
- ✓ Bluetooth, EDGE, 3g y Wi-Fi dependiente del terminal.
- ✓ Cámara, GPS, brújula y acelerómetro dependiente del terminal.
- ✓ Pantalla táctil.
- ✓ Multitarea de aplicaciones.

Android que desde sus inicios se pensó para dispositivos con fines específicos y con la característica de ser dispositivos basados en la arquitectura de ARM, ahora también tiene soporte para la arquitectura x86 (32 bits) gracias a la labor del proyecto Android-x86.

1.4 Android-x86

Android-x86 es un proyecto comunitario de código abierto bajo la licencia pública de Apache 2.0; liderado por Chih-Wei Huang y Yi Sun, cuyo objetivo es portar el proyecto de código abierto Android para la plataforma x86. La idea original fue publicar los diferentes parches para dar soporte a Android x86 desde la comunidad de código abierto. Un par de meses después fue creado el proyecto, con el objetivo de proveer primeramente una solución completa en las plataformas Asus Eee PC y entonces extender sus resultados a las plataformas comunes de x86 (12).

Algunos de los resultados más importantes obtenidos por el proyecto, se pueden listar atendiendo a las características soportadas por la última imagen instalable del sistema.

- ✓ Núcleo 3.0.36, *Kernel Module Setting* (KMS) habilitado.
- ✓ Soporte para red inalámbrica con interfaz gráfica de usuario (GUI).
- ✓ Soporte para red cableada configurable mediante GUI.
- ✓ Mejor instalador de disco.
- ✓ Soporte para suspensión/reanudación de energía (modo S3).
- ✓ Monitor del estado de la batería.
- ✓ Software para controlar el cursor del ratón (*mouse*).

- ✓ Software para controlar los eventos del rodillo del ratón (*mouse*).
- ✓ Soporte para audio (ALSA).
- ✓ Soporte para la cámara V4I2.
- ✓ Soporte para monitor externo.
- ✓ Soporte para auto montar los dispositivos de almacenamiento externos.
- ✓ Soporte para teclado externo.
- ✓ Soporte para modo de depuración con busybox.
- ✓ Soporte para Bluetooth.

Además, mencionar que una versión de Android-x86 fue el sistema operativo de Google TV, un televisor inteligente de la plataforma Google desarrollado en conjunto con Intel y Logitech. La invención de estos dispositivos basados en Android y el navegador Google Chrome dio origen a la televisión interactiva (13).

En la actualidad es mucho el trabajo que falta por hacer para lograr un producto de alta calidad. A las características antes mencionadas que constituyen las principales ventajas de este sistema, se contraponen los siguientes problemas actuales:

- ✓ No tiene soporte para varios formatos de video (MPG, AVI, MOV, entre otros).
- ✓ No tiene soporte para varios formatos de audio (MKA, WMA, WAV, entre otros).
- ✓ No tiene aceleración gráfica.
- ✓ No está disponible para una gran variedad de plataformas.

Para Cuba, que posee una gran cantidad de computadoras basadas en la arquitectura x86, el poder utilizar Android como sistema operativo para estos ordenadores supondría el aprovechamiento de las ventajas de un sistema operativo de última generación en equipos obsoletos. Por lo cual, comprender el funcionamiento de Android, así como el proceso para crear soluciones a la medida basadas en este, se convierte en una tarea de primer orden para el proyecto Nova.

1.5 Métodos y herramientas para el desarrollo basado en Android

En este acápite se hace una breve descripción de las principales herramientas y métodos a emplear en la presente investigación, además, se especifica el lenguaje de modelado de procesos a utilizar, así como, los lenguajes de programación a usar en la implementación de los casos de estudio.

1.5.1 Modelado de procesos

A continuación se muestra la definición de proceso, aspecto importante para conocer la relevancia que tiene el modelado de procesos durante el desarrollo de la presente investigación.

Un proceso es una acción o sucesión de acciones continuas regulares, que ocurren o se llevan a cabo de una forma definida, y que llevan al cumplimiento de algún resultado; una operación continua o una serie de operaciones (14).

En el ámbito de la investigación se puede considerar un proceso como un conjunto de actividades interrelacionadas, cuya salida final es la conformación de un bien o un servicio para un cliente que puede ser interno o externo de la organización (15).

Para el modelado del proceso de personalización de Android se utilizará la notación BPMN.

Notación BPMN

La notación BPMN del inglés *Business Process Modeling Notation* (Notación de Modelado de Procesos de Negocio) es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de actividades. Esta notación define un tipo de diagrama de procesos de negocio que se utiliza para modelar procesos de negocio, así como los sub-procesos y tareas que lo componen (16).

Entre sus ventajas se encuentran:

Ventajas:

- ✓ Considera un único diagrama para la representación de los procesos.
- ✓ Permite modelar los procesos de una manera unificada y estandarizada.
- ✓ Está diseñada para modelar procesos manuales, automáticos, físicos o virtuales.
- ✓ Crea un enlace entre la etapa de diseño e implementación.
- ✓ Es capaz de representar procesos complejos.
- ✓ Es independiente de cualquier metodología de modelado de procesos.

Herramienta de modelado de procesos

Visual Paradigm es una herramienta de ingeniería de software asistida por computadoras para apoyar al proceso de desarrollo de software. Soporta estándares de modelado tales como Lenguaje Unificado de Modelado (UML), Diagramas de Flujo de Datos (DFD), Diagramas de Procesos de Negocio utilizando la notación BPMN. Esta herramienta soporta y facilita el trabajo de los equipos de desarrollo durante la captura de requisitos, planeación del software, ingeniería de código, modelamiento de clases, modelamiento de datos (17).

1.5.2 Herramientas de desarrollo

Durante el desarrollo de la investigación se hará uso de diferentes herramientas, necesarias para la compilación de aplicaciones y edición de código fuente. A continuación se hace una breve descripción de las principales.

Android SDK

El SDK del inglés *Software Development Kit* (Paquete de Desarrollo de Software) de Android incluye una variedad de herramientas que ayudan al desarrollo de aplicaciones para la plataforma. Las herramientas son clasificadas en dos grupos: herramientas del SDK y las herramientas de plataformas. Las herramientas del SDK son independientes de la plataforma y son requeridas sin importar para qué plataforma se está desarrollando la aplicación. Por su parte, las herramientas de plataforma están optimizadas para soportar las características de las plataformas existentes de Android (18).

Android SDK incluye:

- ✓ Librerías necesarias.
- ✓ Entorno de depuración.
- ✓ Emulador de dispositivos móviles.
- ✓ Documentación relevante para las APIs de Android.
- ✓ Código fuente de ejemplo.
- ✓ Tutoriales para el sistema operativo Android.

A pesar de que el SDK puede ser utilizado para escribir programas para Android en la línea de comandos, el método más común es usar un entorno de desarrollo integrado (por su sigla en inglés IDE). El IDE recomendado es Eclipse con el módulo (en inglés *plug-in*) *Android Development Tools* (ADT). La mayoría

de los IDEs proveen una interfaz gráfica permitiendo a los desarrolladores realizar tareas de desarrollo en una forma más rápida.

Android NDK

El *Native Development Kit* (Paquete de Desarrollo Nativo) de Android contiene un conjunto de herramientas que permiten a los desarrolladores de aplicaciones para la plataforma Android integrar componentes que hacen uso de código nativo en sus aplicaciones (19).

Las aplicaciones de Android corren sobre la máquina virtual Dalvik, entonces el NDK permite a los desarrolladores implementar parte de sus aplicaciones usando código nativo de lenguajes tales como C/C++. Esto puede proveer beneficios para ciertas clases de aplicaciones, al reusar código existente en algunos casos e incrementar la velocidad.

El NDK de Android provee:

- ✓ Un conjunto de herramientas y archivos para generar librerías de código nativo a partir de código fuente en C/C++.
- ✓ Una forma de embeber las librerías de código nativo dentro de un paquete de aplicación (.apk).
- ✓ Un conjunto de cabeceras y librerías nativas que serán soportadas a partir de Android 1.5.
- ✓ Documentación, ejemplos y tutoriales.

¿Cuándo desarrollar en código nativo?

El NDK no beneficiará a todas las aplicaciones. Un desarrollador debe equilibrar la relación beneficios/inconvenientes, pues utilizar código nativo no se traduce en un aumento automático del rendimiento de la aplicación, pero en general, siempre aumenta la complejidad de desarrollo de las aplicaciones.

Por lo general, una buena oportunidad para utilizar código nativo es cuando la aplicación hace un uso intensivo del CPU tal como el procesamiento de señales, simulaciones físicas, lo que traerá consigo un incremento del rendimiento. Sin embargo, el NDK puede ser una forma efectiva de reusar código existente de los lenguajes de programación C/C++.

El marco de trabajo de Android provee dos formas de usar código nativo:

1. Escribir las aplicaciones utilizando el marco de trabajo de Android y usar JNI para acceder a las APIs provistas por el paquete de desarrollo nativo de Android.
2. Escribir una *native activity* (actividad nativa), la cual permita ejecutar las devoluciones de llamadas utilizando código nativo.

Herramientas de desarrollo contenidas en el NDK

Este paquete de desarrollo nativo de Android incluye un conjunto herramientas de compilación cruzada (compiladores, conectores o *linkers*). Además, proporciona cabeceras del sistema para las APIs nativas estables, entre ellas:

- ✓ Cabeceras de libc (librería de C).
- ✓ Cabeceras de libm (librería de matemática).
- ✓ Cabeceras de la interfaz de JNI.
- ✓ Cabecera de libz (compresión Zlib).
- ✓ Cabecera liblog (registros de Android).
- ✓ Cabeceras de OpenGL ES 1.1 y OpenGL ES 2.0 (librerías de gráficos en 3D).
- ✓ Cabeceras de libjnigraphics (acceso al buffer de pixel).
- ✓ Un conjunto de cabeceras mínimas para el soporte de C++.
- ✓ OpenSL ES (librerías nativas de audio).
- ✓ APIs nativas de Android.

Herramienta de compilación Make

Make es una herramienta que controla la generación de ejecutables y otros archivos que no son fuente de un programa a partir de archivos de código fuente. Con esta herramienta será posible compilar y construir el sistema, así como, el posterior empaquetamiento de este en un disco imagen.

Make obtiene la forma de cómo compilar los programas a partir de un archivo llamado *makefile*, que enumera cada uno de los archivos que no son código fuente y cómo generar estos a partir de otros archivos. Cuando se escribe un programa o aplicación, se debe generar un *makefile* de modo que sea posible usar Make para compilar e instalar dicho programa (20).

Capacidades de la herramienta Make

- ✓ Make permite a los usuarios finales construir e instalar paquetes sin conocimientos en detalles de cómo fue construido.
- ✓ Make selecciona automáticamente cuales son los archivos que deben ser actualizados, basado en cuales archivos fuentes tienen cambios o modificaciones. De esta forma cuando se ejecuta el comando *make*, no se necesita compilar todo el paquete de aplicación, solo aquellos fuentes que dependen de forma directa o indirecta de los archivos modificados.
- ✓ Make no está limitado a ningún lenguaje en particular.

Make no se limita a la construcción de paquetes. También, se puede utilizar para instalar o desinstalar paquetes siempre y cuando se le especifique en el *makefile* la forma de hacerlo.

Herramienta de virtualización

VirtualBox es una poderosa herramienta de virtualización para las arquitecturas x86/AMD64/Intel64. Esta es una herramienta profesional que está libremente disponible como software de código abierto bajo los términos de la licencia GPL del inglés *General Public License* (Licencia Pública General) de GNU en su versión 2 (21).

VirtualBox es un software de virtualización que fue desarrollado originalmente por la empresa alemana Innotek GmbH, pero que pasó a ser propiedad de la empresa Sun Microsystems en febrero de 2008 cuando ésta compró a Innotek. Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como “sistemas invitados”, dentro de otro “sistema operativo anfitrión”, cada uno con su propio ambiente virtual.

En cuanto a la emulación de hardware, los discos duros de los sistemas invitados son almacenados en los sistemas anfitriones como archivos individuales en un contenedor llamado *Virtual Disk Image*. Otra de las funciones que presenta es la de montar imágenes ISO como unidades virtuales de CD/DVD, o como un disco floppy (22).

Ventajas de VirtualBox

Además de las ventajas descritas anteriormente, se pueden mencionar otras que son de suma importancia para los usuarios finales:

- ✓ Está patentizado bajo la licencia GPL.
- ✓ Actualización constante y cada vez más compatible.
- ✓ Virtualiza diferentes sistemas operativos en distintas versiones.
- ✓ Intuitivo para el usuario.
- ✓ Documentación completa.
- ✓ Configuración de red con distintas opciones.
- ✓ Entorno gráfico atractivo.
- ✓ Compatible para distintas arquitecturas.
- ✓ Programa de confianza.
- ✓ Muy estable al momento de virtualizar el Sistema Operativo.

Desventajas de VirtualBox

Al usar VirtualBox se debe de tener presente que se está utilizando el sistema operativo habitual o físico y sobre él se ejecuta otra súper aplicación, como es otro sistema operativo utilizando la herramienta VirtualBox. Esta acción puede afectar de cierta forma el rendimiento del ordenador, esto supondría una desventaja en la utilización de VirtualBox. A continuación se mencionarán algunas desventajas más:

- ✓ Configuración de resolución un poco estática.
- ✓ El rendimiento teórico de una máquina virtual no va a ser igual que el de una instalación en una partición.

En una comparación del rendimiento obtenido de emular Android con la herramienta de virtualización incluida en el SDK y emular con VirtualBox, se comprueba que es menos costoso, pues el tiempo y recursos necesarios para dicha actividad son menores al utilizar VirtualBox. Atendiendo a estas características y la posibilidad de realizar las configuraciones necesarias para el correcto funcionamiento del sistema, es que se decide emplear a VirtualBox como la herramienta para la virtualización de Android durante el proceso de personalización.

Entorno de desarrollo integrado Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas

en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado *Java Development Toolkit* (JDT).

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El entorno de desarrollo integrado Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. La arquitectura *plug-in* permite escribir cualquier extensión e incorporarla al IDE. Se provee soporte para varios lenguajes de programación entre ellos C/C++ y Java (23).

Utilizar el Eclipse como entorno de desarrollo facilitará el trabajo de codificación, búsqueda en el código fuente del sistema, etc. Además de ser una excelente herramienta, sencilla y con muchas funcionalidades necesarias en la investigación, ejemplo de esto es el *plug-in* ADT el cual facilita la construcción de aplicaciones de una forma visual. Por otra parte, ya existe una versión de Eclipse destinada específicamente al desarrollo de aplicaciones para Android.

1.5.3 Métodos de descripción del proceso de construcción

Para la descripción del proceso de desarrollo de la investigación se utilizarán las siguientes reglas.

- ✓ Los textos que aparecen en un rectángulo gris y la letra negrita se pueden teclear exactamente como aparecen, a menos que se indique lo contrario en el texto subyacente. También se utiliza en las secciones explicativas para identificar el comando al que se hace referencia.
- ✓ Los textos escritos en letras cursivas se utilizan para señalar aspectos importantes y palabras con origen en otro idioma.
- ✓ Las líneas de comando a ejecutar que comiencen “#” significa que hay que ejecutarlas como súper usuario.
- ✓ Las líneas de comando a ejecutar que comiencen “$” significa que hay que ejecutarlas como usuario normal del sistema.

1.6 Conclusiones parciales

Con el objetivo de lograr un buen entendimiento de los lectores de la investigación, se presentaron en este primer capítulo los conceptos fundamentales a tener en cuenta durante todo el proceso de personalización de Android. Para el modelado de procesos se utilizará BPMN y la herramienta Visual Paradigm, Además, se selecciona a Eclipse como entorno de desarrollo integrado, VirtualBox como la herramienta de virtualización, Java y C/C++ como lenguajes de programación, entre otras herramientas importantes para la configuración y construcción de Android. Finalmente se hace una descripción de los métodos empleados para la descripción de procesos.

Capítulo 2: Proceso de configuración y construcción de Android-x86

2.1 Introducción

En el presente capítulo se describe el proceso de instalación y configuración de las herramientas necesarias para el desarrollo de la solución. Además, se introducen a los pasos genéricos para la construcción de aplicaciones dirigidas a la plataforma Android y las formas de incluir esta en dicho sistema.

2.2 Arquitectura de Android

Para empezar con el desarrollo de aplicaciones en Android es importante conocer como está estructurado este sistema operativo. A esto se le llama arquitectura y en el caso de Android está formada por varias capas lo que la convierte en una arquitectura N-Capas, facilitando al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de librerías, de esta forma el desarrollador no tiene que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware.

Cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones, es por ello que a este tipo de arquitectura se le conoce también como *pila* (ver *figura 2*) (24).

Figura 2: Arquitectura interna de la plataforma Android, tomado de la ayuda de Android.

Aplicaciones: En esta capa se encuentran las aplicaciones de usuarios. Todas las aplicaciones creadas con la plataforma Android, incluirán como base un cliente de email, calendario, programa de SMS, mapas, navegador, contactos, y algunos otros servicios mínimos. Todas ellas escritas en el lenguaje de programación Java.

Armazón de Aplicaciones: Todos los desarrolladores de aplicaciones para Android, tienen acceso total al código fuente usando las aplicaciones base. Esto ha sido diseñado de esta forma, para que no se generen cientos de componentes de aplicaciones distintas, que respondan a la misma acción, dando la posibilidad de que los programas sean modificados o reemplazados por cualquier usuario sin tener que empezar a programar sus aplicaciones desde el principio.

Librerías: Android incluye un conjunto de librerías C/C++, que son expuestas a todos los desarrolladores a través del armazón de aplicaciones: Android System C library, librerías de medios, librerías de gráficos 3D, SQLite, entre otros.

Android Runtime: Android incorpora un conjunto de librerías que aportan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la Máquina Virtual Dalvik.

Núcleo de Linux: Android depende de Linux para los servicios bases del sistema, tales como seguridad, gestión de memoria, gestión de procesos, entre otros. También actúa como capa de abstracción entre el hardware y el resto software.

2.3 Proceso de personalización de Android

El proceso para la personalización de Android queda definido como se muestra en la *figura 3*. Este diagrama muestra de la forma más sencilla e intuitiva posible, el flujo de actividades que se necesitan realizar para crear una imagen personalizada de Android, así como, las actividades necesarias para construir cualquier aplicación dirigida a dicho sistema.

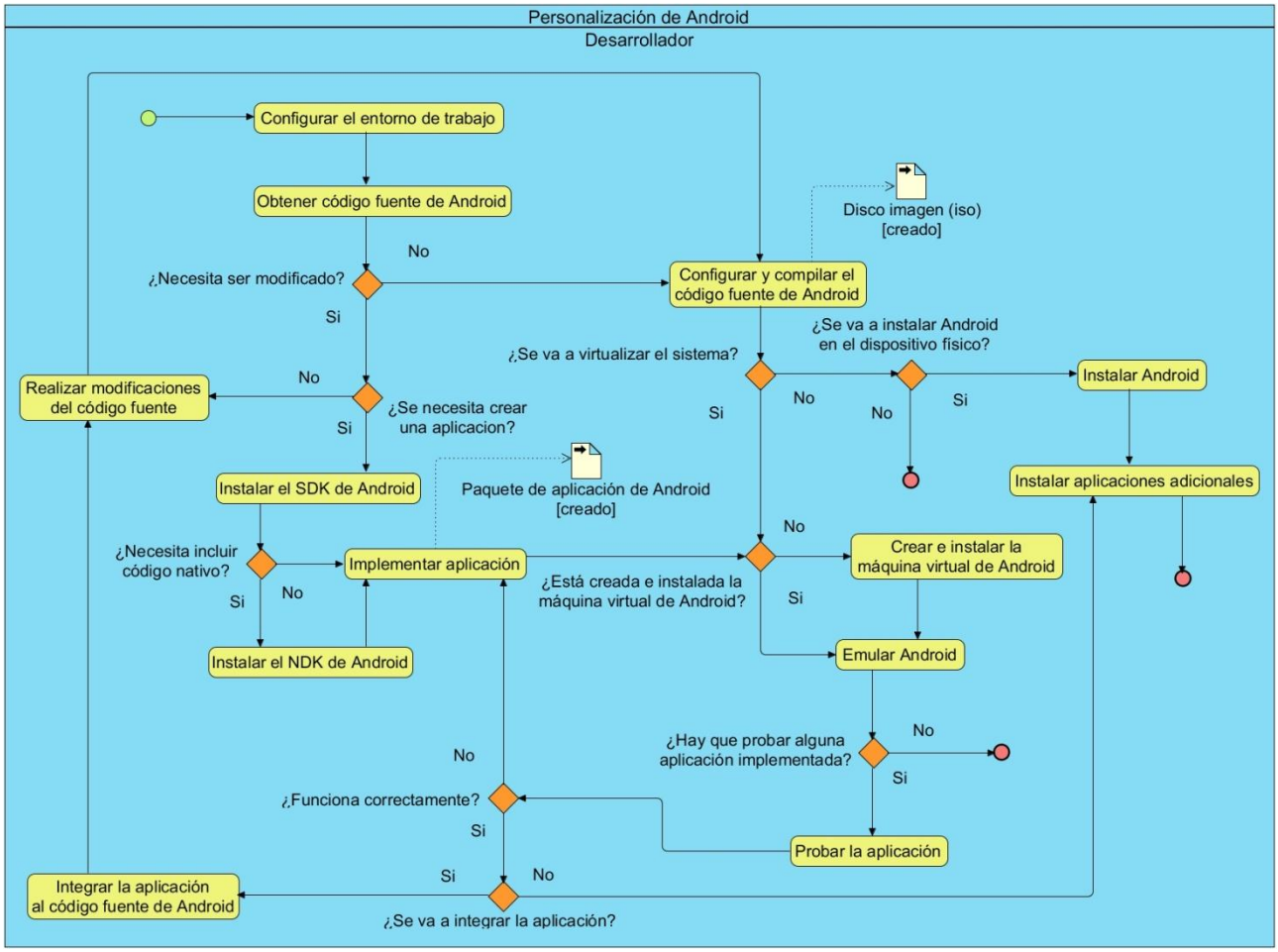


Figura 3: Proceso de personalización de Android.

2.4 Descripción de las actividades del proceso de personalización de Android

A continuación se realiza la descripción de las actividades presentes en el proceso de personalización de Android, con el objetivo de lograr un mejor entendimiento del proceso.

2.4.1 Configurar el entorno de trabajo

En esta actividad se describen las acciones a realizar para instalar y configurar las herramientas necesarias para el trabajo.

Para poder trabajar con Android es necesario tener instalada una versión del JDK del inglés *Java Development Kit* (Paquete de Desarrollo de Java). En la presente investigación se trabajará con el paquete de instalación jdk-6u25-linux-i586.bin. El mismo puede ser descargado directamente desde el servidor FTP de la universidad disponible en ftp://ucistore.uci.cu/software/Desarrollo/Java/JVM/Java%206/jdk-6u25-linux-i586.bin. Una vez descargado el paquete se ejecutan los siguientes comandos para la instalación del JDK en la computadora.

```
$ chmod -x jdk-6u25-linux-i586.bin
$ sh jdk-6u25-linux-i586.bin
$ sudo mv jdk1.6.0_25/ /opt/
$ export PATH=$PATH:/opt/jdk1.6.0_25/bin/
$ export PATH=$PATH:/opt/jdk1.6.0_25/lib/
$ export PATH=$PATH:/opt/jdk1.6.0_25/jre/
$ export PATH=$PATH:/opt/jdk1.6.0_25/include/
```

Luego se procede a instalar un conjunto de paquetes desde el repositorio de Nova con la ejecución de los siguientes comandos en la terminal.

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential curl libc6-dev
$ sudo apt-get install x11proto-core-dev libgl1-mesa-dev g++-multilib mingw32 zip
$ sudo apt-get install openjdk-6-jdk tofrodos python-markdown libxml2-utils xsltproc
$ sudo apt-get install zlib1g-dev:i386 libgl1-mesa-glx:i386 libncurses5-dev:i386
$ sudo apt-get install libreadline6-dev:i386 libx11-dev:i386
```

* Nota: En caso de estar trabajando en un sistema de 32 bits se omite “:i386” al final del nombre del paquete.

Herramienta de virtualización VirtualBox

Además, es necesario instalar y configurar la herramienta de virtualización VirtualBox. Con esta herramienta se probarán las aplicaciones y sistemas generados sin necesidad de instalar Android en un dispositivo físico.

Para la instalación y configuración de VirtualBox es necesario estar conectados a la red y tener configurado correctamente los repositorios de aplicaciones o simplemente tener descargados el paquete de la aplicación y todas sus dependencias.

Ejecutar en la terminal el siguiente comando para instalar VirtualBox y las cabeceras necesarias utilizando la herramienta de gestión de paquetes *apt-get*.

```
$ sudo apt-get install linux-headers-`uname -r`
$ sudo apt-get install virtualbox virtualbox-qt virtualbox-dkms virtualbox-guest-dkms
```

Luego de hacer esto, para que el sistema virtualizado reconozca los dispositivos USB es necesario agregar el usuario con el que se está trabajando al grupo 'vboxusers', para esto se ejecuta en la línea de comandos lo siguiente.

```
$ sudo adduser trabajo vboxusers
```

Ahora se necesita instalar un paquete de extensión de VirtualBox, el cual permitirá agregar nuevos filtros de USB a la máquina virtual. En este caso como la versión que se utiliza para esta investigación es VirtualBox versión 4.1.12, se procede a instalar el paquete *Oracle_VM_VirtualBox_Extension_Pack-4.1.12.vbox-extpack* disponible en la siguiente dirección de descarga http://download.virtualbox.org/virtualbox/4.1.12.

Una vez que se tenga este paquete de extensión descargado se procede a instalar en el VirtualBox de la siguiente forma. En el menú **Archivo** clic en **Preferencias…**, en la parte izquierda clic pestaña **Extensiones**, entonces hacer clic en el ícono **Agregar paquete** de la derecha y luego especificar la **ruta del paquete de extensión de VirtualBox** y finalmente clic en el botón **Aceptar**.

Luego por cada una de las máquinas virtuales que se desee que reconozca los dispositivos USB se procede de la siguiente forma. Seleccionar la máquina virtual y hacer clic en el botón **Configuración**, en la pestaña USB habilitar las casillas **Habilitar controlador USB** y **Habilitar controlador USB 2.0 (EHCI)**, luego clic en el botón **Agregar filtro vacío (Ins)** y finalmente clic en **Aceptar**.

Entorno integrado de desarrollo Eclipse

La versión a utilizar del Eclipse será la destinada específicamente para el desarrollo de aplicaciones de Android, disponible en ftp://10.22/software/Desarrollo/Android/adt-bundle-linux-x86.zip, este comprimido ya trae las principales herramientas a ser utilizadas durante la construcción de aplicaciones. Se descomprime

el paquete y dentro se encuentra el Eclipse con el *plug-in* ADT ya instalado por lo que no hay necesidad de configurarlo.

2.4.2 Obtener código fuente de Android

El código fuente de Android-x86 puede ser descargado desde el repositorio del proyecto Android-x86 con dirección electrónica http://www.android-x86.org, además, se puede encontrar en dicho sitio, mucha información sobre la forma de obtener el código fuente y la configuración del entorno de trabajo para poder construir el sistema.

Hasta el momento el proyecto Android-x86 ha portado las siguientes versiones de Android.

- ✓ donut-x86: Basado en Android 1.6 (Donut).
- ✓ eclair-x86: Basado en Android 2.1 (Eclair).
- ✓ froyo-x86: Basado en Android 2.2 (Froyo).
- ✓ gingerbread-x86: Basado en Android 2.3 (Gingerbread).
- ✓ honeycomb-x86: Basado en Android 3.2 (Honeycomb).
- ✓ ics-x86: Basado en Android 4.0 (Ice Cream Sandwich).
- ✓ jb-x86: Basado en Android 4.2 (Jelly Bean).

De estas personalizaciones se obtiene el código fuente correspondiente a la plataforma *Ice Cream Sandwich (ICS)* para el desarrollo de la investigación.

2.4.3 Realizar modificaciones al código fuente

Está actividad está destinada para actualizar y modificar cualquier archivo fuente o de configuración existente en el código de Android, en caso que se necesite, por ejemplo para hacer uso de alguna librería que se agregue o incluir aplicaciones que se compilarán en conjunto con el sistema.

2.4.4 Configurar y compilar el código fuente de Android

Para la construcción de la imagen es necesario seleccionar una plataforma destino para la cual será configurado y compilado Android-x86. Entre los diferentes tipos de plataformas en correspondencia con las versiones de Android para las que se han creado personalizaciones y teniendo en cuenta que la versión descargada fue la 4.0, se encuentran:

- ✓ generic_x86: para computadoras/netbook genéricas con la arquitectura x86.
- ✓ amd_brazos: para la plataforma AMD Brazos.
- ✓ eeepc: para la familia ASUS EeePC.
- ✓ asus_laptop: para algunas computadoras portátiles Asus.
- ✓ tegav2: para Tegatech Tegav2.

Se selecciona eeepc como plataforma destino para la configuración y compilación, pues entre las disponibles es esta la que más se asemeja al hardware de las computadoras del proyecto Nova, específicamente a las computadoras con placa madre Asus P5LD2-VM. Luego se ejecuta el comando siguiente para empezar con el proceso de construcción de la imagen.

```
$ make -jX iso_img TARGET_PRODUCT=eeepc
```

Donde X es el número de procesadores de la computadora donde se está compilando el sistema más uno. Como en el proceso de compilación se hace un alto uso de la memoria, es recomendable utilizar una memoria SWAP de unos 4 Gb.

La ejecución del comando anterior generará un disco imagen en la ruta **/out/target/product/eeepc** dentro de la carpeta que contiene el fuente de Android-x86.

2.4.5 Instalar Android

Para realizar la instalación de Android se siguen los pasos generales para instalar cualquier tipo de sistema operativo, para realizar esta actividad se puede guiar por el *Anexo I*.

2.4.6 Instalar el SDK de Android

Este paquete puede ser descargado desde la web, se recomienda utilizar la zona de descarga del sitio oficial para desarrolladores de Android, disponible en la dirección http://developer.android.com/sdk/index.html, o puede ser descargado desde el servidor FTP de la Universidad de las Ciencias Informáticas disponible en la dirección ftp://ucistore.uci.cu/software/Desarrollo/Android.

Luego de esto, el paquete debe ser descomprimido en una ubicación física de la computadora de desarrollo, en la investigación se descomprime en la carpeta android bajo la raíz del sistema "**/android/android-sdk"**.

Con el fin de poder utilizar las herramientas disponibles en el paquete de desarrollo de software de Android, es necesario modificar la variable de entorno **PATH** del usuario, con ese fin, se agrega la siguiente línea al final del archivo ***.bashrc*** correspondiente al usuario para el que estarán disponibles dichas herramientas.

```
$ export PATH=${PATH}:/android/android-sdk/tools:/android/android-sdk/platform-tools
```

* Nota: Las direcciones deben ser cambiadas por la dirección donde se encuentre ubicado el *android-sdk*.

2.4.7 Instalar el NDK de Android

Se recomienda utilizar la zona de descarga del sitio oficial para desarrolladores de Android, disponible en la dirección http://developer.android.com/tools/sdk/ndk/index.html, o puede ser descargado desde el servidor FTP de la Universidad de las Ciencias Informáticas disponible en la dirección ftp://ucistore.uci.cu/software/Desarrollo/Android.

Luego de esto, el paquete debe ser descomprimido en una ubicación física de la computadora de desarrollo, en la investigación se descomprime en la carpeta android bajo la raíz del sistema "**/android/android-ndk**".

Para poder utilizar las herramientas disponibles en el paquete de desarrollo de software de Android, se agrega la siguiente línea al final del archivo ***.bashrc*** del usuario, esto modifica la variable de entorno **PATH** del usuario.

```
$ export PATH=${PATH}:/android/android-ndk
```

* Nota: La direcciones deben ser cambiadas por la dirección donde se encuentre ubicado el *android-ndk*.

2.4.8 Implementar aplicación

Esta actividad está pensada para la construcción de aplicaciones dirigidas a cualquiera de las capas de la arquitectura del sistema. Son varios los tutoriales para crear aplicaciones que se pueden encontrar en la Web, también se pueden consultar los ejemplos que se encuentran en el SDK de Android, y además, se puede estudiar la documentación disponible en el sitio oficial para desarrolladores de Android en la

siguiente dirección http://developer.android.com/training/basics/firstapp/index.html. Luego de construir la aplicación se compila el proyecto lo que genera un paquete de aplicación de Android (archivo ***.apk***).

2.4.9 Crear e instalar la máquina virtual de Android

Para emular Android es necesario crear la máquina virtual para dicho propósito, por lo que se debe tener un disco imagen de instalación de Android. Ahora se siguen los siguientes pasos para crear la máquina virtual.

Primero se hace clic en el botón **Nueva** y se hace clic en el botón **Siguiente**, en el campo Nombre se escribe un nombre apropiado e identificativo para la máquina virtual que se está creando, en el Tipo de sistema operativo se selecciona Linux versión **Other Linux**, entonces clic en el botón siguiente. Luego de esto se especifica el Tamaño de memoria base en la investigación se recomienda **512 MB**. Ahora se crea un nuevo disco duro de arranque del tipo **VDI** del inglés *VirtualBox Disk Image* (Disco Imagen de VirtualBox) y reservado dinámicamente con un tamaño de 8 GB, finalmente se verifican los parámetros de la máquina virtual y se hace clic en el botón **Crear**.

Ahora se ejecutan los pasos para la instalación de Android en el nuevo disco virtual creado, para ello, en el menú de configuración de la máquina virtual se hace clic en Almacenamiento a la izquierda de la ventana, y en Árbol de almacenamiento clic en el ícono Agregar dispositivo CD/DVD y se selecciona el disco imagen. Luego de estas acciones ya se puede iniciar la máquina virtual haciendo clic en Iniciar. Una vez iniciada la máquina se siguen los pasos que se especifican en el Anexo I para la instalación de Android.

2.4.10 Emular Android

Al emular Android en una máquina virtual es posible probar aplicaciones y el propio sistema en sí, sin necesidad de tener un dispositivo específico para realizar estas tareas. Al hacer esto se ahorra tiempo y recursos de *hardware* y facilita el proceso de pruebas.

Terminada la instalación se puede configurar VirtualBox para que ajuste la resolución de pantalla de la máquina virtual y se asemeje al tamaño de un teléfono, con este objetivo se ejecuta el siguiente comando, donde “Android” corresponde al nombre de la máquina virtual para la cual se está realizando la configuración.

```
$ VBoxManage setextradata "Android" CustomVideoMode1 "320x480x16"chmod
```

Al desarrollar aplicaciones con el IDE Eclipse es posible emular Android con VirtualBox lo cual ahorraría tiempo y una gran cantidad de recursos en máquinas no muy potentes para el desarrollo y emulación de Android directamente con la herramienta de emulación del SDK de Android. Para que sea posible trabajar de esta forma se puede hacer de dos maneras distintas.

La primera de ellas es desde la interfaz gráfica de VirtualBox, hacer clic en el botón de **Configuración** de la máquina virtual, luego en la parte izquierda hacer clic en **Red** y en la parte derecha marcar la casilla de verificación **Habilitar adaptador de red** y seleccionar **NAT** de la lista de opciones, dentro de **Avanzadas** seleccionar como tipo de adaptador **PCnet-FAST III (Am79C973)**, luego habilitar la casilla de verificación **Cable conectado** y hacer clic en el botón Reenvío de puertos, entonces, clic en el ícono **Agregar nueva regla (Ins)** y especificar los parámetros de la *tabla 2* y hacer clic en el botón **Aceptar**.

Tabla 2: Ejemplo de configuración para el reenvío de puertos de VirtualBox.

Nombre	Protocolo	IP anfitrión	Puerto anfitrión	IP invitado	Puerto invitado
adb	tcp		5555		5555

La segunda forma de hacer esta configuración del reenvío de puertos es mediante la ejecución del comando siguiente, donde “Android” corresponde al nombre de la máquina virtual para la cual se está realizando la configuración.

```
$ VBoxManage modifyvm "Android" --natpf1 "adbd,tcp,,5555,,5555"
```

Para iniciar la máquina virtual se hace clic sobre el ícono **Iniciar** en la parte superior de la herramienta de virtualización.

2.4.11 Probar la aplicación

Una vez implementada y compilada la aplicación, se emula Android y se instala el paquete ***.apk*** obtenido en el sistema, luego se realizan pruebas a la aplicación para comprobar su correcto funcionamiento y que cumpla con los requisitos deseados.

2.4.12 Integrar la aplicación al código fuente de Android

En esta actividad se toman las archivos fuentes y recursos necesarios, se copian para una dirección específica en el código fuente de Android, en dependencia de para qué capa de la arquitectura está destinada. Además, se crea el ***Android.mk*** el cual es el archivo que define el cómo se va compilar la aplicación.

2.4.13 Instalar aplicaciones adicionales

Una vez que se tiene la aplicación para la plataforma Android en el formato ***.apk*** es posible instalarla de varias formas.

Una de las vías de instalación es como comúnmente se hace con los paquetes de aplicaciones de Android, se copia o descarga en el dispositivo, y en el explorador de archivos clic sobre el paquete, entonces se ejecuta el asistente para aplicaciones.

Además, se puede recurrir a la herramienta de línea de comandos *Android Debug Bridge* (Puente de Depuración de Android) en lo adelante “ADB”, la cual permite la comunicación con una instancia o dispositivo Android. Haciendo uso de esta herramienta es sencilla la instalación de aplicaciones. Como ejemplo se muestran los comandos siguientes que permiten la conexión con la máquina virtual previamente creada y con los parámetros correctos de red.

```
$ adb connect localhost:5555
$ adb install my_app.apk
```

Para más información sobre la herramienta se puede consultar la dirección electrónica http://developer.android.com/tools/help/adb.html.

2.5 Conclusiones parciales

En este capítulo se describió la arquitectura del sistema, así como el proceso de personalización de Android, además se describieron las actividades que conforman dicho proceso. Con esto se pretende guiar a los desarrolladores del proyecto Nova en el entorno de desarrollo orientado a Android. También, ayuda en la asimilación de esta tecnología y conocimientos para futuras soluciones autóctonas.

Capítulo 3: Implementación de casos de estudio

3.1 Introducción

En el presente capítulo se realizan algunos casos de estudio basados en escenarios, poniendo en práctica el proceso de personalización de Android descrito en el capítulo anterior, lo cual permitirá comprobar la completa funcionalidad del flujo propuesto en el proceso.

3.2 Implementación para la capa de Aplicaciones

En el desarrollo para la capa de Aplicaciones de la arquitectura de Android, es necesario conocer que existen dos tipos de aplicaciones: aplicaciones del sistema y las aplicaciones que no son del sistema.

Una aplicación del sistema es aquella que por su importancia o criticidad para el correcto funcionamiento de la plataforma se instala en el directorio ***/system/app*** dentro del dispositivo. Este directorio tiene la característica de tener la propiedad de *solo-lectura*, esto implica que las aplicaciones en él instaladas, no pueden ser eliminadas o desinstaladas del dispositivo por la simple acción del usuario. Además, este tipo de aplicación tiene acceso a permisos reservados únicamente a las aplicaciones del sistema.

Por otra parte, las aplicaciones que no son del sistema o simplemente aplicaciones de usuario son aquellas que instalan y desinstalan los usuarios comúnmente. Estas en cambio son instaladas en el directorio ***/data/app*** dentro del dispositivo, el mismo tiene la propiedad de *lectura-escritura* lo cual permite a los usuarios eliminar las aplicaciones que allí se encuentran instaladas (25).

Se debe tener en cuenta además, que el sistema operativo Android se basa en el principio de mínimos privilegios, por lo cual a cada aplicación se le otorgan únicamente los permisos que necesita y no más (26). Esto implica que en el desarrollo de aplicaciones se declaren en el ***AndroidManifest.xml***[1] los permisos que deba adquirir la aplicación, los cuales serán concedidos o denegados por el ***Package Manager*** (Manejador de Paquetes) de Android. Si una aplicación hace uso de características propias de la plataforma, entonces debe declararse como aplicación del sistema para que se le concedan los permisos necesarios.

[1] AndroidManifest.xml este archivo presenta la información esencial de cada aplicación para el sistema Android.

Teniendo en cuenta la existencia de estos dos tipos de aplicaciones se explicará el proceso de construcción e instalación para cada una de ellas.

3.2.1 Aplicaciones de usuario

A continuación se detallará el proceso de desarrollo de una aplicación de usuario haciendo uso de código nativo, este estará dirigido por el proceso de personalización de Android descrito con anterioridad. Con esto se pretende demostrar los aspectos a tener en cuenta para realizar esta tarea, así como comprobar la efectividad del flujo de actividades propuesto.

Realizadas las tareas precedentes a la propia actividad de Implementar la aplicación, se procede según los pasos básicos para el desarrollo de aplicaciones haciendo uso de código nativo propuestos en la documentación del NDK instalado, dicha guía puede ser accedida desde la dirección ***$NDK/documentation.html***, donde $NDK corresponde al directorio de instalación del Android NDK. Esta variable de entorno puede ser creada mediante la ejecución del comando siguiente.

```
$ export NDK='/android/android-ndk'
```

La documentación propone los siguientes pasos generales para el desarrollo de este tipo de aplicaciones.

1. Ubicar el código nativo dentro del directorio ***$PROJECT/jni/…***
2. Implementar el archivo ***$PROJECT/jni/Android.mk*** para describir el código fuente al sistema de construcción del NDK.
3. Opcional: implementar el archivo ***$PROJECT/jni/Application.mk*** para describir el proyecto en más detalles al sistema de construcción.
4. Construir (compilar) el código nativo con la ejecución de ***$NDK/ndk-build*** ubicados en el directorio del proyecto o cualquiera de sus sub-directorios.

En caso de no ocurrir algún error se copiará al directorio raíz del proyecto la librería creada. Luego se genera el ***.apk*** final de la forma común.

A continuación se explica cómo implementar una aplicación de usuario, la cual tiene como requisito la captura de un nombre en un campo de texto y mostrar un mensaje de saludo. Se debe hacer uso de código nativo para obtener la hora del sistema y dar cumplimiento a la funcionalidad.

1. **Crear un nuevo proyecto de aplicación de Android en el Eclipse.**

Se introducen los datos de la nueva aplicación de Android (ver *figura 4*).

***Application Name*:** nombre de la aplicación que la identificará dentro del conjunto de aplicaciones instaladas en el sistema.

***Project Name*:** nombre del proyecto que debe ser único en el *workspace* (espacio de trabajo del Eclipse) en el que se esté trabajando.

***Package Name*:** nombre del paquete el cual debe ser un identificador único de la aplicación. Este no es mostrado a los usuarios; pero debe ser invariable durante toda la línea de vida de la aplicación. Típicamente es el nombre de dominio de la organización en adición de los identificadores del proyecto, y finalmente este tiene que ser un nombre de paquete válido de Java.

***Minimum Required SDK*:** se selecciona la menor versión de Android para la cual brindará soporte la aplicación.

***Target SDK*:** se selecciona la versión de Android para la cual está específicamente destinada la aplicación.

***Compile With*:** se selecciona la API de Android con la cual se compilará el código fuente. Este es típicamente la versión más reciente.

***Theme*:** esta propiedad representa el tema base que usará la aplicación, dígase *look and feel*.

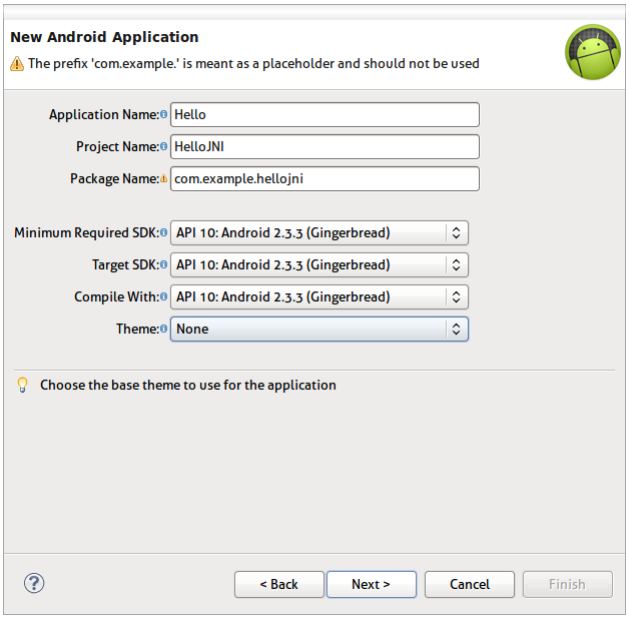


Figura 4: Datos del nuevo proyecto.

En la vista siguiente (ver *figura 5*) se verifica que esté marcada la casilla *"Create activity"*, en caso contrario no se creará la clase Java para implementar la lógica de la aplicación y poder cumplir con los requisitos propuestos.

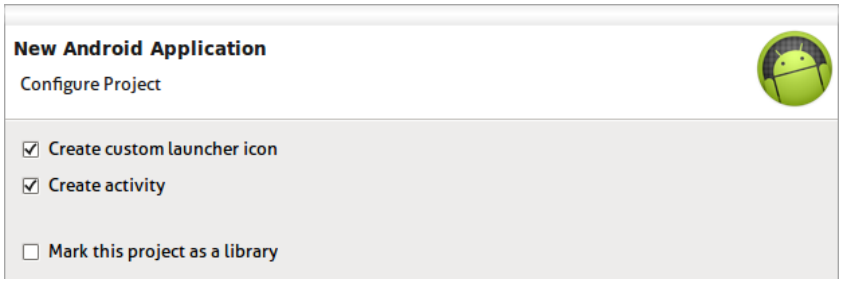


Figura 5: Configuración del proyecto.

En la siguiente vista (ver *figura 6*) se pueden configurar los atributos de los iconos de la aplicación. En el ejemplo se configura *"Additional Padding"* a un 10%, lo cual causa que el icono se muestre más pequeño.

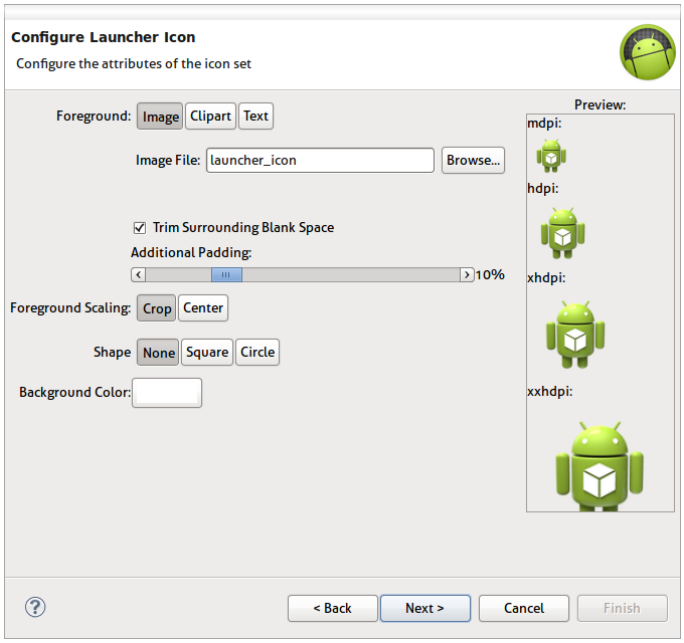


Figura 6: Configuración de los atributos de los iconos.

Luego se selecciona el tipo de actividad entre las predefinidas. En la *figura 7* se muestra un ejemplo utilizando una actividad en blanco.

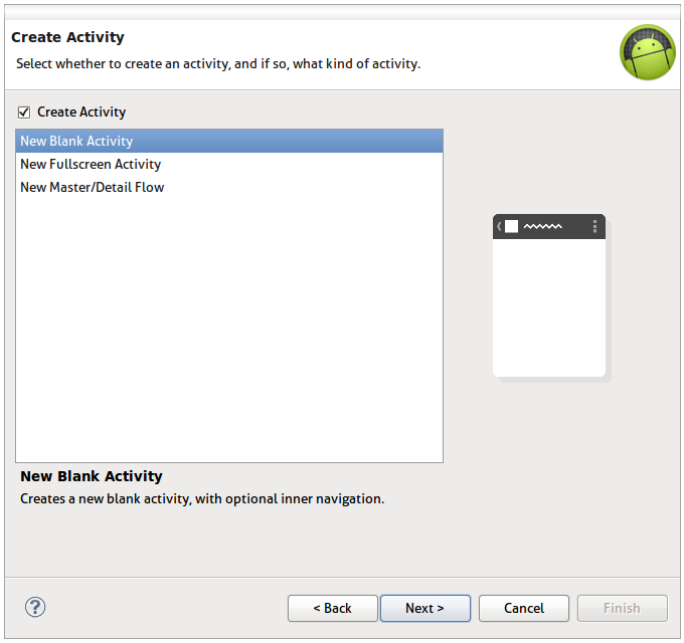


Figura 7: Tipo de actividad a crear.

Después se introducen los atributos de la actividad que se está creado (ver *figura 8*).

***Activity Name*:** nombre de la clase actividad a crear, esto se traduce en una clase Java.

***Layout Name*:** nombre del diseño que se va a crear para la actividad.

***Navigation Type*:** tipo de navegación a usar en la actividad.

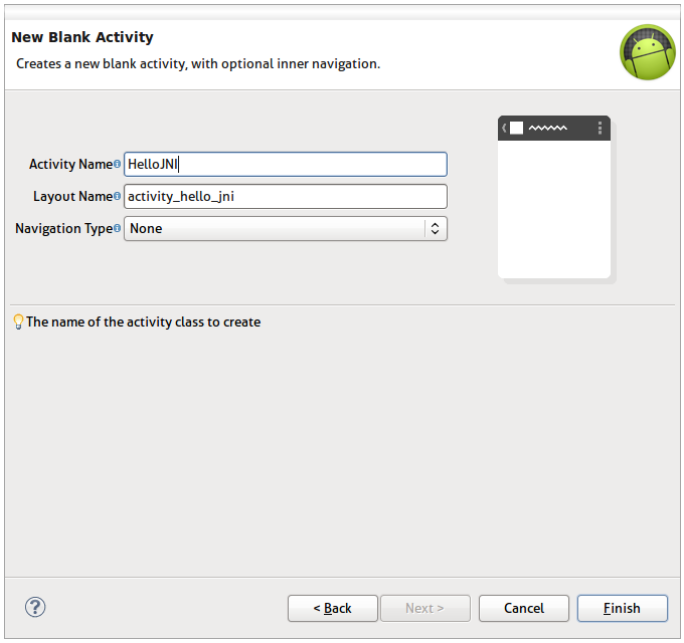


Figura 8: Propiedades de la nueva actividad

Una vez introducidos los datos de la nueva actividad se finaliza la creación de la aplicación.

2. **Crear el diseño de la aplicación.**

Primeramente se elimina el *TextView* con texto *"Hello Word!"* creado por defecto. Luego se arrastran sobre el diseño de la actividad los siguientes elementos: un ***TextView***, un ***Button*** y un ***EditText*** como se muestra en la *figura 9*.

Figura 9: Diseño primario de la aplicación.

Una vez posicionados estos elementos como se muestra en la *figura 9*, se edita el archivo ***HelloJNI/res/values/strings.xml*** para agregar los recursos de texto que serán utilizados en la edición del texto de los elementos que componen el diseño. Para esto se adicionan los recursos identificados por los nombres únicos: *label_text*, *button_text* y *edit_hint* tal y como se muestra en la *figura 10*.

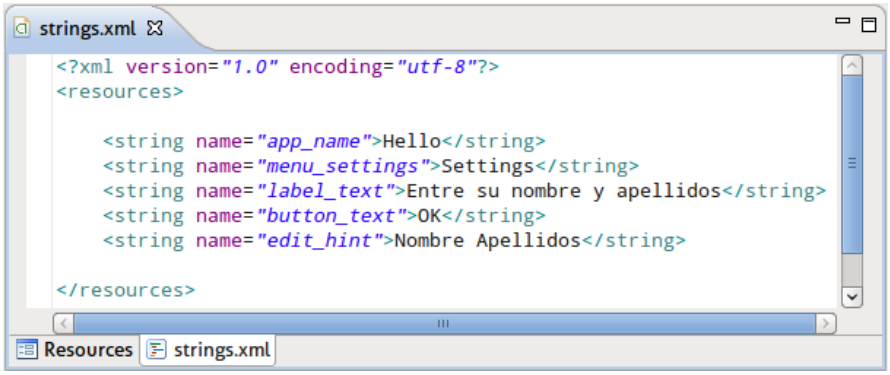


Figura 10: Creación de los recursos de texto.

Luego de adicionar los recursos de texto correspondientes a *label_text*, *button_text* y *edit_hint* en el archivo *strings.xml*, se procede a editar el texto de los componentes en la vista de diseño de la aplicación. Para realizar esta acción se hace ***clic derecho*** sobre cada uno de los elementos y se presiona sobre el

elemento del menú emergente con texto ***Edit text…*** o en el caso del *EditText* se presiona sobre ***Edit Hint…***, entonces se despliega la vista que se muestra en la *figura 11*, donde se selecciona el recurso de texto correspondiente al elemento a editar.

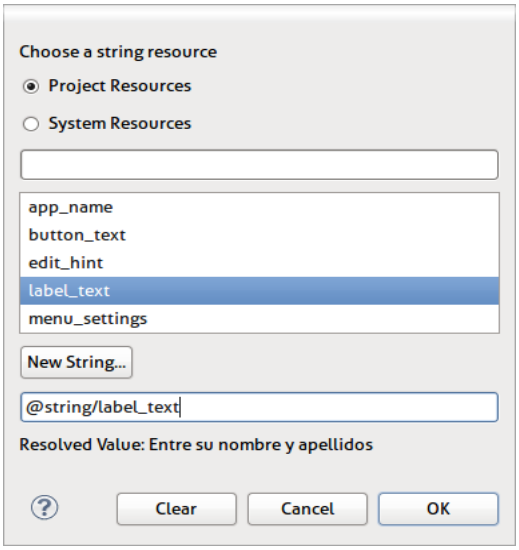


Figura 11: Selección de los recursos de texto.

Es necesario adicionar el evento *onClick* al botón, de modo que al ser presionado, se emita un evento encargado de invocar el método definido por el desarrollador. Para esto, se adiciona la propiedad ***android:onClick="onClickButton_OK"*** al botón en el archivo ***HelloJNI/res/layout/activity_hello_jni.xml*** en la vista del *xml* como se muestra en la *figura 12*, donde *onClickButton_OK* es el método que será invocado al pulsar sobre el botón.

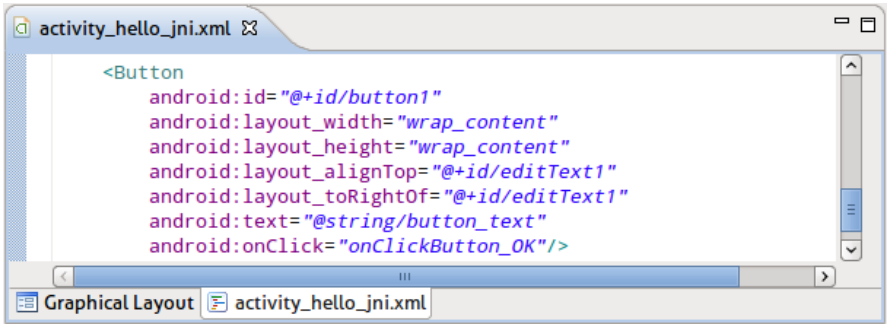


Figura 12: Propiedad *onClick* del botón.

Realizadas estas acciones se obtiene el diseño de la aplicación que se muestra en la *figura 13*. Ahora corresponde implementar la lógica de la aplicación.

Figura 13: Diseño terminado de la aplicación.

3. **Implementar la lógica de la aplicación.**

Para implementar la lógica de la aplicación se edita el archivo ***HelloJNI/src/com/example/hellojni/HelloJNI.java*** que corresponde a la clase principal de la actividad que se creó, tal y como se muestra en la *figura 14*.

Aunque los métodos *onCreate* y *onCreateOptionsMenu* permanecen invariables en la clase *HelloJNI*, se debe informar a dicha clase que cargue la librería *hello-jni* que será implementada luego, lo cual es posible invocando a la función ***System.loadLibrary("hello-jni")***. Además se declara el método *getMessage()* como una función externa implementada en código nativo mediante la palabra reservada *native*, esta declaración se realiza de la siguiente forma ***public native String getMessage(String name)***.

activity_hello_jni.xml | HelloJNI.java

```
package com.example.hellojni;

import android.os.Bundle;

public class HelloJNI extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public void onClickButton_OK(View v) {
        // Se obtiene el EditText del diseño gráfico.
        EditText editText_name = (EditText) findViewById(R.id.editText1);
        // Se obtiene el nombre escrito en el EditText.
        String name = editText_name.getText().toString();
        // Se pasa el nombre por parámetro al método nativo 'getMessage'
        // y retorna el mensaje de saludo adecuado según la hora del día.
        String msg = getMessage(name);
        // Se muestra el mensaje en pantalla.
        Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
    }

    static {
        // Se carga la librería 'hello-jni'.
        System.loadLibrary("hello-jni");
    }
    // Se declara el método nativo 'getMessage'.
    public native String getMessage(String name);
}
```

Figura 14: Implementación de la clase *HelloJNI*.

4. **Ubicar el código nativo dentro del directorio *$PROJECT/jni/…***

Realizadas la implementación en la clase *HelloJNI.java* se acede mediante la consola al directorio ***HelloJNI/bin/classes*** y se ejecuta el comando siguiente para crear la cabecera de la librería *hello-jni* a partir de la implementación de la clase anteriormente mencionada.

```
$ javah -jni com.example.hellojni.HelloJNI
```

La ejecución de este comando genera el archivo ***HelloJNI/bin/classes/com_example_hellojni_HelloJNI.h***. Este archivo se debe mover al directorio ***HelloJNI/jni*** y puede ser renombrado a conveniencia del desarrollador. Con la ejecución del comando siguiente en el directorio raíz del proyecto, se mueve dicho archivo al directorio mencionado, renombrándolo como ***hello-jni.h***.

```
$ mkdir jni
$ mv bin/classes/com_example_hellojni_HelloJNI.h jni/hello-jni.h
```

Una vez ejecutado el comando es necesario convertir el proyecto ***HelloJNI*** en un proyecto de C/C++ nativo. Para esto se hace ***clic derecho*** sobre el proyecto y dentro del menú desplegable ***New*** seleccionar ***Other…***, luego en la vista que se muestra (ver *figura 15*) se selecciona ***Convert to a C/C++ Project (Adds C/C++ Nature)***.

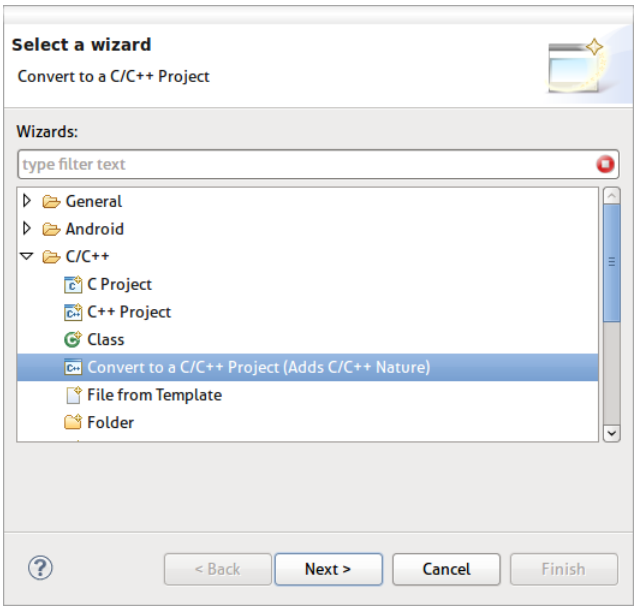


Figura 15: Conversión del proyecto en un proyecto de C/C++.

En la vista siguiente (ver *figura 16*) se selecciona como tipo de proyecto ***Makefile project*** utilizando la herramienta de compilación cruzada ***Android GCC***.

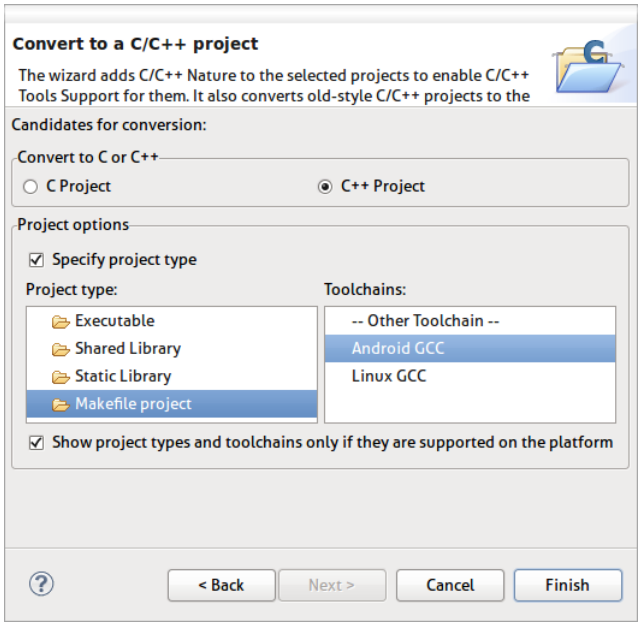


Figura 16: Selección del tipo de proyecto a utilizar.

Una vez finalizadas estas acciones, el desarrollador crea el archivo de código fuente en el lenguaje C: ***HelloJNI/jni/hello-jni.c*** cuya implementación se observa en la *figura 17*. En esta implementación se obtiene la hora del sistema, lo que permite crear el mensaje de saludo en dependencia de la hora del día. Este mensaje será capturado en el código Java y mostrado como una notificación.

hello-jni.h hello-jni.c

```
#include "hello-jni.h"
#include <stdio.h>
#include <time.h>

JNIEXPORT jstring JNICALL Java_com_example_hellojni_HelloJNI_getMessage
(JNIEnv *env, jobject thiz, jstring string)
{
    // Se convierte el jstring a un puntero de char.
    char* name = (char*) (*env)->GetStringUTFChars(env, string, 0);
    char msg[100], hora[3];
    int H;
    // Se obtiene el tiempo actual.
    time_t raw_time = time(0);
    // Se obtiene el timepo local.
    struct tm* tlocal = localtime(&raw_time);
    // Se formatea el tiempo local para sacar la hora actual.
    strftime(hora,100,"%H",tlocal);
    sscanf(hora, "%d", &H);

    // Con 'sprintf' se almacena el mensaje adecuado en la variable msg.
    if(H >= 0 && H < 12)
        sprintf(msg, "Good morning, %s", name);
    else if(H >= 12 && H < 20)
        sprintf(msg, "Good afternoon, %s", name);
    else
        sprintf(msg, "Good night, %s", name);

    // Se convierte el puntero a char en un jstring.
    return (*env)->NewStringUTF(env, msg);
}
```

Figura 17: Implementación de *hello-jni.c*.

5. **Implementar el archivo *$PROJECT/jni/Android.mk*.**

Ahora se procede a crear e implementar el archivo ***HelloJNI/jni/Android.mk***, donde se especifica qué tipo de módulo es el que se va compilar y cuales son los archivos fuentes (ver *figura 18*). En general el *Android.mk* y el *Application.mk* son versiones reducidas de los archivos *makefile.* Se puede obtener información sobre la estructura y funcionamiento de estos dos archivos en ***$NDK/docs/ANDROID-MK.html*** y ***$NDK/docs/APPLICATION-MK.html*** respectivamente.

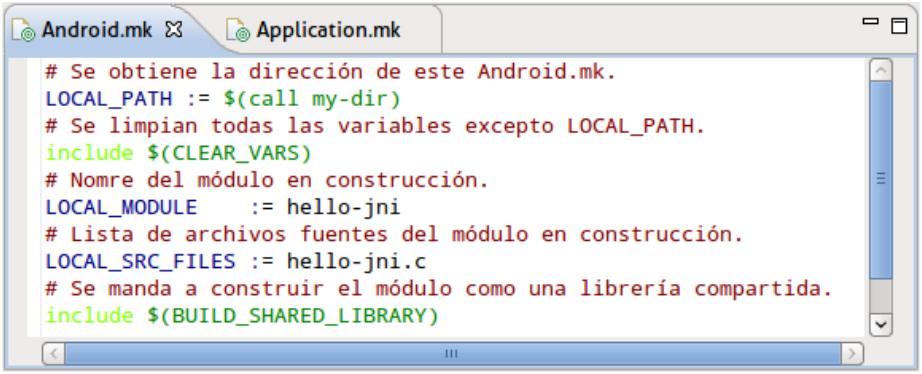


Figura 18: Implementación del *Android.mk*.

6. **Implementar el archivo *$PROJECT/jni/Application.mk*.**

Ahora se crea y edita el archivo ***HelloJNI/jni/Application.mk*** el cual tiene información sobre cuál es la plataforma objetivo de la aplicación y para que arquitectura se compilará (ver *figura 19*).

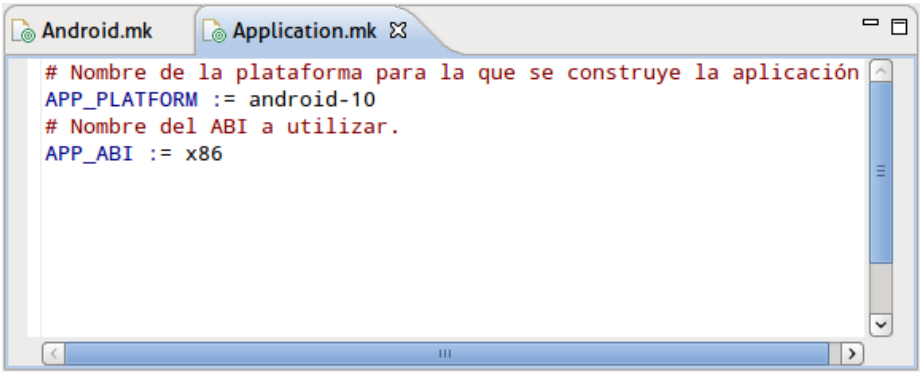


Figura 19: Implementación del *Application.mk*.

7. **Construir el código nativo.**

Para construir el código nativo se ejecuta el comando que se muestra a continuación. Si no hay ningún error, este comando genera la librería ***libhello-jni.so*** en la dirección ***HelloJNI/libs/x86/***.

```
$ $NDK/ndk-build
```

8. **Emular Android.**

Para emular Android se decide utilizar una máquina virtual del sistema creada en la herramienta de virtualización VirtualBox, para su configuración ver *Capítulo 2* de la presente investigación. Primero se inicia la máquina virtual y luego se ejecuta el siguiente comando para conectar dicha máquina con la herramienta ADB.

```
$ adb connect localhost:5555
```

Con esta conexión es posible ejecutar la aplicación desde el entorno de desarrollo Eclipse.

9. **Construir y ejecutar la aplicación en el Eclipse.**

Para construir y ejecutar la aplicación se hace clic en el botón ***Run HelloJNI*** en la barra de herramientas del Eclipse, permitiendo que se construya la aplicación y su final empaquetamiento en el archivo ***HelloJNI.apk*** ubicado en la carpeta ***HelloJNI/bin/***. Luego esta aplicación se instala y se ejecuta de forma automática en el sistema emulado.

10. **Firmar la aplicación en modo liberación.**

Después que una aplicación ha sido desarrollada y probadas todas sus funcionalidades, se debe firmar esta con una clave. Android requiere que todas las aplicaciones instaladas sean firmadas digitalmente con un certificado cuya clave privada debe ser mantenida por el desarrollador de la aplicación.

Los puntos fundamentales para comprender por qué es importante que las aplicaciones de Android sean firmadas, son:

- ✓ Todas las aplicaciones deben ser firmadas. El sistema no instalará una aplicación en un emulador o dispositivo si esta no está firmada.
- ✓ Para probar y depurar la aplicación, la herramienta de construcción firma esta con una llave especial para depurar creada por la herramienta de construcción del SDK de Android.
- ✓ Cuando la aplicación está lista para ser liberada a los usuarios finales debe ser firmada con una llave privada adecuada.
- ✓ Se pueden usar las herramientas *Keytool* y *Javasigner* para generar llaves y formar los archivos de aplicación.

- ✓ Luego que una aplicación es firmada para su liberación, se recomienda usar la herramienta *Zipalign* para optimizar el paquete final de la aplicación.

A continuación se describen los pasos para crear un paquete de aplicación de Android firmado y alineado en el Eclipse.

1. Seleccionar el proyecto en el explorador de proyectos y seleccionar la opción ***File > Export***.
2. Desplegar la carpeta Android y seleccionar la opción ***Export Android Application***, luego hacer clic en ***Next***.
3. Luego se muestra el asistente para exportar las aplicaciones de Android, el cual guiará el proceso de firmado de la aplicación, incluyendo los pasos para seleccionar la llave privada con la que se va a firmar el *apk* (o crear una nueva si es necesario).
4. Finalmente se tendrá la aplicación compilada, firmada, alineada y lista para la distribución.

Una vez implementada, probada y firmada la aplicación, ya se puede tomar el archivo ***HelloJNI.apk*** generado durante el proceso e instalarlo de la forma común en Android o integrarlo al código fuente del sistema para que se compile junto con este.

11. Integrar la aplicación al código fuente de Android.

Para integrar esta aplicación de usuario al código fuente del sistema, se debe copiar el paquete de aplicación en el directorio ***/android/x86/device/common/app***. Al copiar el archivo ***HelloJNI.apk*** en este directorio, el ***Android.mk*** existente en el mismo se encargará de mandar a incluirlo como un paquete pre-construido dentro del sistema. Lo cual permitirá que al obtener el disco imagen la aplicación se encuentre instalada por defecto.

3.2.2 Aplicaciones del sistema

A continuación se implementará una aplicación del sistema, teniendo como requisito funcional que la pantalla sea bloqueada al presionar un botón.

Para implementar una aplicación del sistema se realizan los dos primeros pasos descritos en el epígrafe anterior para el desarrollo de aplicaciones de usuarios. Una vez realizadas estas actividades quedaría diseñada la actividad principal la aplicación del sistema como se muestra en la *figura 20*.

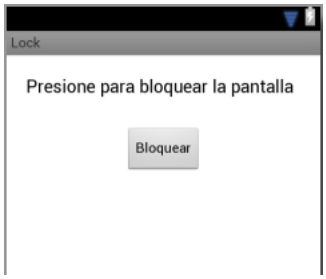


Figura 20: Diseño de la aplicación del sistema.

Al hacer uso de funcionalidades que modifican el estado del dispositivo es necesario declarar que permisos mínimos son requeridos. Además, como cada aplicación representa un usuario distinto dentro del sistema operativo, se hace necesario declararla como una aplicación del sistema para que se le concedan los permisos que se requieren.

Los cambios necesarios en el ***AndroidManifest.xml*** se muestran en la *figura 21*. Dentro del ***manifest*** se agrega la línea ***android:sharedUserId="android.uid.system"*** para asignar el usuario de las aplicaciones del sistema y pueda ganar los permisos solicitados. Además, se agrega la línea ***<uses-permission android:name="android.permission.DEVICE_POWER" />*** lo cual permite hacer usar el ***PowerManager*** (Administrador de energía) y las funciones que este implementa.

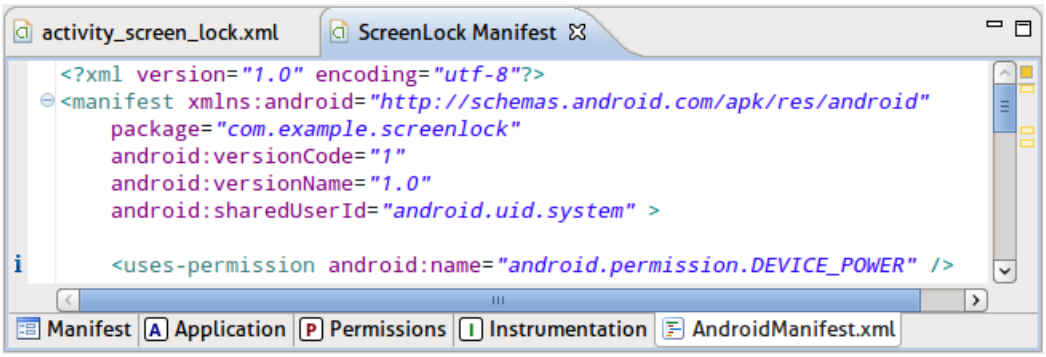


Figura 21: *AndroidManifest.xml* de la aplicación del sistema.

Luego de declarar estos aspectos, se procede a implementar la lógica de la aplicación la cual se muestra en la *figura 22*.

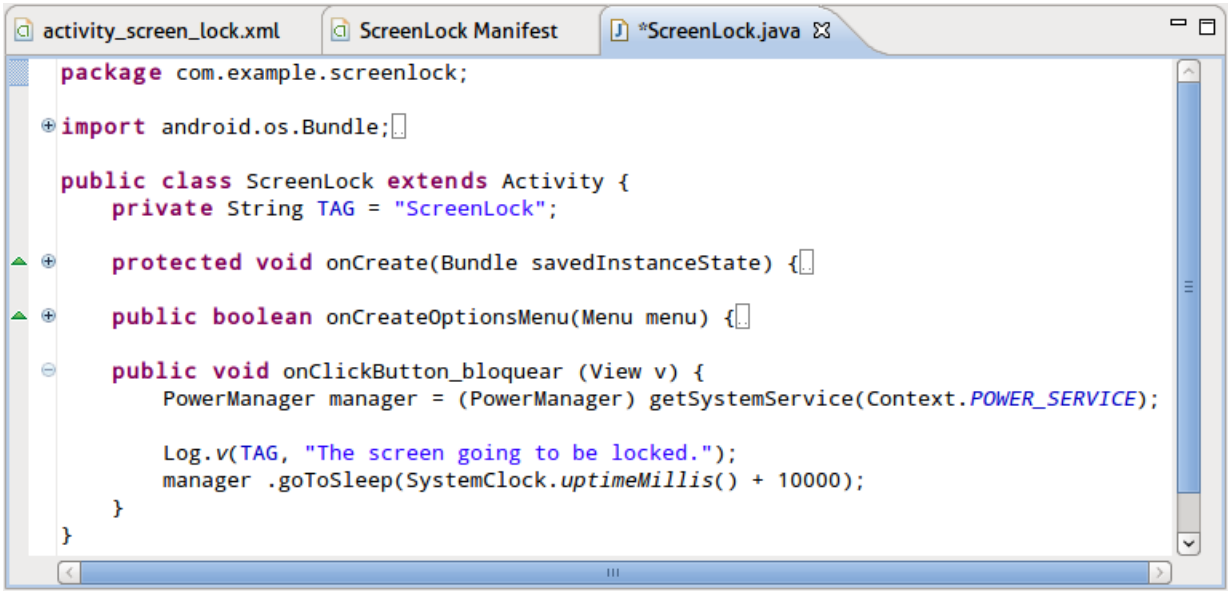


Figura 22: Implementación de la clase *ScreenLock.java.*

Para poder comprobar el correcto funcionamiento de la aplicación es necesario firmarla con la llave de las aplicaciones del sistema de la plataforma. Para esto se seguirá un flujo de actividades distinto al usado en la aplicación anterior.

A continuación se describen los pasos para crear un paquete de aplicación sin firmar en el Eclipse.

- ✓ Seleccionar el proyecto en el explorador de proyectos y seleccionar la opción ***Android Tools > Export Unsigned Application Package...*** del menú emergente al hacer clic derecho sobre el proyecto.
- ✓ Especificar la dirección y nombre del paquete sin firmar que se obtendrá.

Ahora, haciendo uso de la herramienta *signapk.jar* se firma la el paquete el cual se podrá instalar como se mostrará a continuación.

```
$ java -jar /android/x86/out/host/linux-x86/framework/signapk.jar \
/android/x86/build/target/product/security/platform.x509.pem \
/android/x86/build/target/product/security/platform.pk8 \
ScreenLock-unsigned.apk ScreenLock-signed.apk
```

Para instalar esta aplicación del sistema en una máquina virtual y poder comprobar su buen funcionamiento, se ejecuta el siguiente comando.

```
$ adb install ScreenLock-signed.apk
```

Realizadas estas acciones se puede ejecutar la aplicación. Para comprobar que efectivamente se esté ejecutando como una aplicación del sistema se puede ejecutar el comando.

```
$ adb shell ps
```

Este comando muestra una tabla dónde se relacionan los procesos activos del sistema y con qué usuario están ejecutándose. En el caso de esta aplicación el usuario debe ser “system”.

Integrar la aplicación al código fuente de Android

Para integrar la aplicación con el código fuente de Android se copia el proyecto en la dirección ***/android/x86/packages/apps***. Hecho esto se crea el archivo ***Android.mk*** en la raíz del proyecto. Dicho archivo queda definido como se muestra en la *figura 23*.

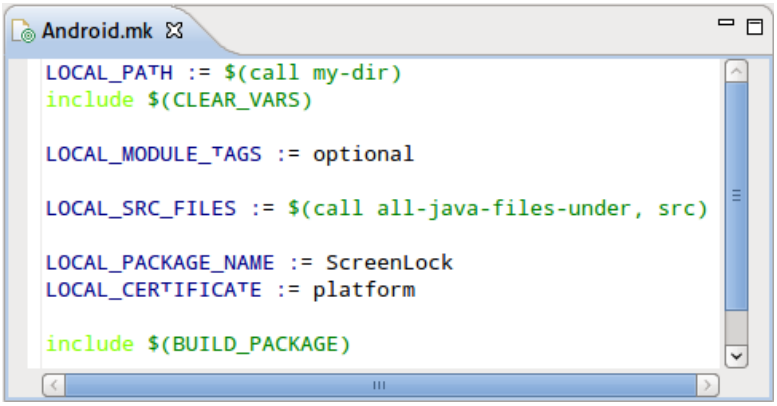


Figura 23: *Android.mk* de la aplicación del sistema.

En este punto es necesario editar el archivo **/android/x86/build/target/product/generic.mk**, en el cual se agrega la línea ***PRODUCT_PACKAGES := ScreenLock***. Esto posibilita que durante el proceso de compilación del sistema se construya la aplicación.

Configurar y compilar el código fuente de Android

Para configurar y compilar el código fuente de Android se ejecuta el siguiente comando.

```
$ make -j2 iso_img TARGET_PRODUCT=eeepc
```

Como se dijo con anterioridad, al finalizar el proceso de compilación del sistema se generó un disco imagen en la dirección ***/android/x86/out/target/product/eeepc***. Este *iso* puede ser usado tanto para virtualizar un sistema, como para instalarlo en un dispositivo de la arquitectura i386.

3.3 Conclusiones parciales

El proceso de personalización de Android demostrado en los casos de estudio en este capítulo, favorece a la documentación necesaria para dirigir el proceso a otras soluciones. Algunos aspectos que se conocieron fueron: los dos tipos de aplicaciones existentes para la primera de las capas de la arquitectura y los pasos para el firmado de aplicaciones. Además, se demostraron las actividades para la integración de aplicaciones al código fuente del sistema.

Conclusiones

A partir de la importancia que tiene la definición de un proceso para la personalización de Android dirigida al personal del proyecto Nova se arrojaron los siguientes resultados.

- ✓ Se presentaron los conceptos fundamentales a tener en cuenta durante todo el proceso de personalización de Android, así como las tecnologías, herramientas y métodos utilizados.
- ✓ Se sistematizó la arquitectura de la plataforma, lo que evidenció que este sistema presenta una arquitectura N-Capas, en la que cada una utiliza elementos de la capa inferior para realizar sus funciones.
- ✓ Se definió el proceso de personalización de Android que guiará a los desarrolladores en la implementación de soluciones a la medida.
- ✓ Al aplicar el proceso en la implementación de dos casos de estudio, los principales resultados obtenidos son:
  - ➢ Se identificaron los dos tipos de aplicaciones dirigidas a la capa más externa de la arquitectura y las características de cada una de ellas.
  - ➢ Se conoció el modo de firmado de las aplicaciones para su distribución a los usuarios finales.
  - ➢ Se demostraron los pasos para la integración de aplicaciones al código fuente del sistema.
  - ➢ Se evidenció la superioridad de la herramienta de virtualización VirtualBox sobre su análoga incluida en el SDK de Android, durante la realización de las actividades incluidas en el proceso definido.
  - ➢ Se comprobó la validez del proceso de personalización de Android definido durante la investigación.

Recomendaciones

Se recomienda:

1. Continuar la investigación del proceso de construcción de Android-x86 para distintos dispositivos.
2. Analizar a profundidad el proceso de desarrollo para las distintas capas de la arquitectura de Android-x86.

Bibliografía referenciada

1. **ICIO2.** Tendencias para los dispositivos móviles 2010-2013. [En línea] 30 de abril de 2011. [Consultado el: 15 de enero de 2013]. Disponible en: http://cxo-community.com/articulos/blogs/blogs-management/3939-tendencias-para-los-dispositivos-moviles-2010-2013.html.

2. **Webopedia.** What is mobile operating system? [En línea] [Consultado el: 08 de febrero de 2013]. Disponible en: http://www.webopedia.com/TERM/M/mobile_operating_system.html.

3. **Morales, Diego.** Sistemas operativos para celulares. [En línea] 18 de enero de 2012. [Consultado el: 08 de febrero de 2013]. Disponible en: http://www.slideshare.net/diego_morales/sistemas-operativos-para-celulares-11136511.

4. **Gartner.** Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009. [En línea] 16 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.gartner.com/newsroom/id/2017015.

5. **Evaluamos.** Sistemas operativos en móviles inteligentes en 1Q de 2012. [En línea] 29 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.evaluamos.com/internal.php?load=detail&id=13283.

6. **Haselton, Todd.** Android Has 56.1% Of Global OS Market Share, Gartner Says. [En línea] 16 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.technobuffalo.com/2012/05/16/android-has-56-1-of-global-os-market-share-gartner-says.

7. **Vilchez, Angel.** Qué es Android: Características y Aplicaciones. [En línea] 02 de abril de 2009. [Consultado el: 15 de enero de 2013]. Disponible en: http://www.configurarequipos.com/doc1107.html.

8. **Google.** Open Handset Alliance. [En línea] [Consultado el: 12 de febrero de 2013]. Disponible en: http://www.openhandsetalliance.com.

9. **Android.** Discover Android. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://www.android.com/about.

10. **Alumnos del ITS Villada.** Android OS – Android OS 0.1 documentation. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: https://androidos.readthedocs.org/en/latest.

11. **AndroidDevMX.** Java y la máquina virtual Dalvik . [En línea] 15 de noviembre de 2011. [Consultado el: 15 de enero de 2013]. Disponible en: http://www.androiddevmx.net/java-y-la-maquina-virtual-dalvik.

12. **Proyecto Android-x86.** Android-x86 - Porting Android to x86. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://www.android-x86.org.

13. **Google TV.** Información sobre Google TV en español. [En línea] [Consultado el: 21 de febrero de 2013]. Disponible en: http://www.googletelevision.es.

14. **Diccionario de la lengua española -Vigésima segunda edición.** Real Academia Española. [En línea] [Consultado el: 07 de abril de 2013]. Disponible en: http://buscon.rae.es/draeI/.

15. **Pozo Rodríguez, José Manuel y Rodríguez Cotilla, Zoe.** Procesos y operaciones. [En línea] febrero de 2005. [Consultado el: 5 de abril de 2013]. Disponible en: http://www.gestiopolis.com/recursos4/docs/ger/consite.htm.

16. **Andrés, Rolando Rodríguez.** Lenguajes, notaciones y herramientas para el modelado y análisis de procesos. [En línea] 16 de junio de 2008. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

17. **Visual Paradigm.** Visual Paradigm for UML. [En línea] 2013. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.visual-paradigm.com.

18. **Android Developers.** Tools Help. [En línea] [Consultado el: 06 de abril de 2013]. Disponible en: http://developer.android.com/tools/help/index.html.

19. **Android Developers.** Android NDK. [En línea] [Consultado el: 03 de abril de 2013]. Disponible en: http://developer.android.com/tools/sdk/ndk/index.html.

20. **Free Software Fundation.** GNU Make. [En línea] 3 de septiembre de 2010. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.gnu.org/software/make.

21. **Oracle.** VirtualBox. [En línea] [Consultado el: 22 de febrero de 2013]. Disponible en: https://www.virtualbox.org.

22. **Lashon StudioWordPress.** Virtual Box .es. [En línea] [Consultado el: 22 de febrero de 2013]. Disponible en: http://virtualbox.es.

23. **Martinez, Chus.** IDE Eclipse. [En línea] 29 de septiembre de 2010. [Consultado el: 22 de febrero de 2013]. Disponible en: https://sites.google.com/site/picandojavacom/mundo-java/ide-eclipse.

24. **Rodríguez, Aurora.** Arquitectura de Android. [En línea] 04 de julio de 2011. [Consultado el: 05 de abril de 2013]. Disponible en: http://androideity.com/2011/07/04/arquitectura-de-android.

25. **Baldacchino, Sergio.** Explaining the behavior of an Android application: System apps vs Non-System apps. [En línea] 10 de abril de 2013. [Consultado el: 26 de abril de 2013]. Disponible en: http://ricston.com/blog/?p=3177.

26. **Meneses, Daniela y Lipa, Jorge.** Fundamentos de android. [En línea] 2012. [Consultado el: 26 de abril de 2013]. Disponible en: https://sites.google.com/site/grafcomputacional/manejo-de-graficos-en-android/fundamentos-de-android.

Bibliografía consultada

Alumnos del ITS Villada. Android OS – Android OS 0.1 documentation. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: https://androidos.readthedocs.org/en/latest.

Android Developers. Android Developers. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://developer.android.com.

Android Developers. Android NDK. [En línea] [Consultado el: 03 de abril de 2013]. Disponible en: http://developer.android.com/tools/sdk/ndk/index.html.

Android Developers. Android, the world's most popular mobile platform. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://developer.android.com/about/index.html.

Android Developers. Android.mk file syntax specification. [En línea] [Consultado el: 08 de marzo de 2013]. Disponible en: http://www.kandroid.org/ndk/docs/ANDROID-MK.html.

Android Developers. Get the Android SDK. [En línea] [Consultado el: 03 de abril de 2013]. Disponible en: http://developer.android.com/sdk/index.html.

Android Developers. Tools Help. [En línea] [Consultado el: 06 de abril de 2013]. Disponible en: http://developer.android.com/tools/help/index.html.

Android Developers. What is the NDK? [En línea] [Consultado el: 03 de abril de 2013]. Disponible en: http://developer.android.com/tools/sdk/ndk/overview.html.

Android-X86. Android for x86 from scratch. [En línea] [Consultado el: 08 de marzo de 2013]. Disponible en: https://sites.google.com/site/lapndaandroid/android-x86.

Android. Discover Android. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://www.android.com/about.

AndroidDevMX. Java y la máquina virtual Dalvik . [En línea] 15 de noviembre de 2011. [Consultado el: 15 de enero de 2013]. Disponible en: http://www.androiddevmx.net/java-y-la-maquina-virtual-dalvik.

Rodríguez Andrés, Rolando. Lenguajes, notaciones y herramientas para el modelado y análisis de procesos. [En línea] 16 de junio de 2008. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

Baldacchino, Sergio. Explaining the behavior of an Android application: System apps vs Non-System apps. [En línea] 10 de abril de 2013. [Consultado el: 26 de abril de 2013]. Disponible en: http://ricston.com/blog/?p=3177.

Beal, Vangie. Mobile Operating Systems (Mobile OS) Explained. [En línea] 08 de octubre de 2011. [Consultado el: 08 de febrero de 2013]. Disponible en: http://www.webopedia.com/DidYouKnow/Hardware_Software/mobile-operating-systems-mobile-os-explained.html.

Dergarabedian, Cesar. Los 12 modelos de celulares que llegaron para renovar la oferta y mantener "caliente" el mercado de telefonía móvil. [En línea] 25 de junio de 2012. [Consultado el: 09 de marzo de 2013]. Disponible en: http://www.iprofesional.com/notas/139135-Los-12-modelos-de-celulares-que-llegaron-para-renovar-la-oferta-y-mantener-caliente-el-mercado-de-telefona-mvil-.

Diccionario de la lengua española -Vigésima segunda edición. Real Academia Española. [En línea] [Consultado el: 07 de abril de 2013]. Disponible en: http://buscon.rae.es/draeI.

Evaluamos. Sistemas operativos en móviles inteligentes en 1Q de 2012. [En línea] 29 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.evaluamos.com/internal.php?load=detail&id=13283.

Fernández, Jaime. iSCSI en la práctica. [En línea] 01 de octubre de 2005. [Consultado el: 27 de febrero de 2013]. Disponible en: http://www.networkworld.es/iSCSI-en-la-practica/seccion-/articulo-170841.

Free Software Fundation. GNU Make. [En línea] 3 de septiembre de 2010. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.gnu.org/software/make.

Gargenta, Marko. Remixing Android - Marakana. [En línea]. 13 de febrero de 2012. [Consultado el: 23 de abril de 2013]. Disponible en: http://marakana.com/s/post/1044/remixing_android.

Gartner. Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009. [En línea] 16 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.gartner.com/newsroom/id/2017015.

González, M. Bases de datos. [En línea] 2004. [Consultado el: 12 de febrero de 2013]. Disponible en: http://www.monografias.com/trabajos27/bases-datos/bases-datos.shtml.

Google TV. Información sobre Google TV en español. [En línea] [Consultado el: 21 de febrero de 2013]. Disponible en: http://www.googletelevision.es.

Google. Android. [En línea] [Consultado el: 12 de febrero de 2013]. Disponible en: http://www.android.com.

Google. Open Handset Alliance. [En línea] [Consultado el: 12 de febrero de 2013]. Disponible en: http://www.openhandsetalliance.com.

Haselton, Todd. Android Has 56.1% Of Global OS Market Share, Gartner Says. [En línea] 16 de mayo de 2012. [Consultado el: 18 de enero de 2013]. Disponible en: http://www.technobuffalo.com/2012/05/16/android-has-56-1-of-global-os-market-share-gartner-says.

Hernández, Jon. Android SDK, ya disponible. [En línea] 12 de noviembre de 2007. [Consultado el: 03 de abril de 2013]. Disponible en: http://www.visualbeta.es/647/movil/android-sdk-ya-disponible.

ICIO2. Tendencias para los dispositivos móviles 2010-2013. [En línea] 30 de abril de 2011. [Consultado el: 15 de enero de 2013]. Disponible en: http://cxo-community.com/articulos/blogs/blogs-management/3939-tendencias-para-los-dispositivos-moviles-2010-2013.html.

Janssen, Cory. Android SDK. [En línea] [Consultado el: 13 de abril de 2013]. Disponible en: http://www.techopedia.com/definition/4220/android-sdk.

Lashon StudioWordPress. Virtual Box .es. [En línea] [Consultado el: 22 de febrero de 2013]. Disponible en: http://virtualbox.es.

Martinez, Chus. IDE Eclipse. [En línea] 29 de septiembre de 2010. [Consultado el: 22 de febrero de 2013]. Disponible en: https://sites.google.com/site/picandojavacom/mundo-java/ide-eclipse.

Meneses, Daniela y Lipa, Jorge. Fundamentos de android - Computación gráfica. [En línea]. 18 de junio de 2012. [Consultado el: 20 de mayo de 2013]. Disponible en: https://sites.google.com/site/grafcomputacional/manejo-de-graficos-en-ndroid/fundamentos-de-android.

Morales, Diego. Sistemas operativos para celulares. [En línea] 18 de enero de 2012. [Consultado el: 08 de febrero de 2013]. Disponible en: http://www.slideshare.net/diego_morales/sistemas-operativos-para-celulares-11136511.

Nicola, Thierry. Install ANDROID Development Tools (ADT) on Ubuntu 10.04 (and Eclipse). [En línea] 19 de julio de 2010. [Consultado el: 03 de abril de 2013]. Disponible en: http://www.williambrownstreet.net/blog/?p=266.

Oracle. VirtualBox. [En línea] [Consultado el: 22 de febrero de 2013]. Disponible en: https://www.virtualbox.org.

Pozo Rodríguez, José Manuel y Rodríguez Cotilla, Zoe. Procesos y operaciones. [En línea] febrero de 2005. [Consultado el: 5 de abril de 2013]. Disponible en: http://www.gestiopolis.com/recursos4/docs/ger/consite.htm.

Proyecto Android-x86. Android-x86 - Porting Android to x86. [En línea] [Consultado el: 15 de enero de 2013]. Disponible en: http://www.android-x86.org.

Rodríguez Andrés, Rolando. Lenguajes, notaciones y herramientas para el modelado y análisis de procesos [En línea]. 16 de junio de 2008. [Consultado el: 20 de mayo de 2013]. Disponible en: http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

Rodríguez, Aurora. Arquitectura de Android. [En línea] 04 de julio de 2011. [Consultado el: 05 de abril de 2013]. Disponible en: http://androideity.com/2011/07/04/arquitectura-de-android.

Rodríguez, Aurora. La máquina virtual Dalvik. [En línea] 07 de julio de 2011. [Consultado el: 12 de febrero de 2013]. Disponible en: http://androideity.com/2011/07/07/la-maquina-virtual-dalvik.

S. Khorate, Swapnilkumar. My First Hello World in Android on Linux(Ubuntu) Platform. [En línea] 30 de junio de 2011. [Consultado el: 03 de abril de 2013]. Disponible en: http://swapnilkumarkhorate.wordpress.com/2011/06/30/my-first-hello-world-in-android-on-linuxubuntu-platform-2.

Sanchez, Xavier. Qué es iSCSI y porqué iSCSI. [En línea] 13 de diciembre de 2009. [Consultado el: 27 de febrero de 2013]. Disponible en: http://www.ctxdom.com/index.php?option=com_content&view=article&id=338:que-es-iscsi-y-porque-iscsi&catid=31:general&Itemid=72.

Vilchez, Angel. Qué es Android: Características y Aplicaciones. [En línea] 02 de abril de 2009. [Consultado el: 15 de enero de 2013]. Disponible en: http://www.configurarequipos.com/doc1107.html.

Visual Paradigm. Visual Paradigm for UML. [En línea] 2013. [Consultado el: 22 de febrero de 2013]. Disponible en: http://www.visual-paradigm.com.

Webopedia. What is mobile operating system? [En línea] [Consultado el: 08 de febrero de 2013]. Disponible en: http://www.webopedia.com/TERM/M/mobile_operating_system.html.

Willis, Nathan. Running Android on x86. [En línea] 14 de marzo de 2012. [Consultado el: 28 de febrero de 2013]. Disponible en: http://lwn.net/Articles/486383.

Glosario de términos

ALSA (Advanced Linux Sound Architecture): Término inglés que significa Arquitectura Avanzada de Sonido para Linux, provee un API para los controladores de dispositivo de la tarjeta de sonido.

API (Application Programming Interface): Término inglés que significa Interfaz de Programación de Aplicaciones.

APK (Android Application Package): Término inglés que significa Paquete de Aplicación de Android, es un formato de archivo usado para distribuir e instalar las aplicaciones de Android.

Bluetooth: Sistema de comunicación inalámbrica que permite la interconexión de diferentes dispositivos electrónicos.

Busybox: Es un programa que combina muchas utilidades estándares de Unix en un solo ejecutable, se le denomina como *"la navaja suiza de los sistemas embebidos"*.

Compilación cruzada: Es el proceso mediante el cual se realiza la compilación de aplicaciones en una computadora, denominada huésped, para ser ejecutado sobre otra arquitectura de hardware, denominado objetivo.

CPU (Central Processing Unit): Término que significa Unidad Central de Procesamiento, es el componente principal del ordenador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

Dispositivo: Cualquier elemento que se pueda unir al sistema mediante algún medio de conexión.

Emular: Simular el hardware o software de otra computadora.

Framework: Plataformas o herramientas del mundo de la informática que le proveen a los programadores un grupo de facilidades en el ámbito para la cual han sido creadas.

FTP (File Transfer Protocol): Término inglés que significa Protocolo de Transferencia de Ficheros, es un sistema de transferencia de ficheros en el Internet.

Imagen ISO: Es un archivo donde se almacena una copia o imagen exacta de un sistema de ficheros.

JNI (Java Native Interface): Término inglés que significa Interfaz Nativa de Java, es una interfaz de programación que permite al código Java inter-operar con funciones que son escritas en otros lenguajes de programación.

Librería: Conocido también como biblioteca, es un conjunto de subprogramas utilizados para desarrollar software.

Núcleo: Es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro a los dispositivos físicos de la computadora. Es el encargado de gestionar recursos, a través de servicios de llamadas al sistema.

Placa madre: También conocido como placa base es un dispositivo que funciona como plataforma o circuito principal de un ordenador, permite integrar y coordinar los demás componentes de la computadora.

Plataforma: Sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

RISC (Reduced Instruction Set Computer): Es una filosofía para la construcción de procesadores, se define además como una Arquitectura de Computadora.

Sistema Operativo: Conjunto de programas y utilidades básicas que hacen funcionar la computadora.

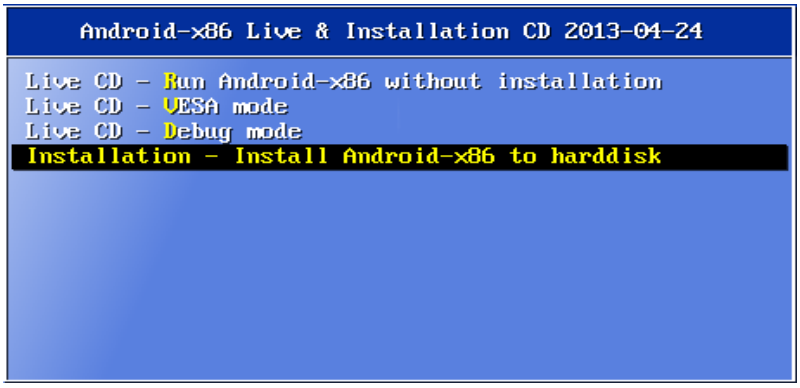
Software: Término genérico que designa al conjunto de programas operativos que posibilitan el uso del ordenador. Aplicación o programa de cómputo.

USB (Universal Serial Bus): Interfaz que permite la comunicación entre la computadora y los dispositivos periféricos.

X86: Es una familia de procesadores diseñada por Intel que comparten un mismo conjunto de instrucciones.

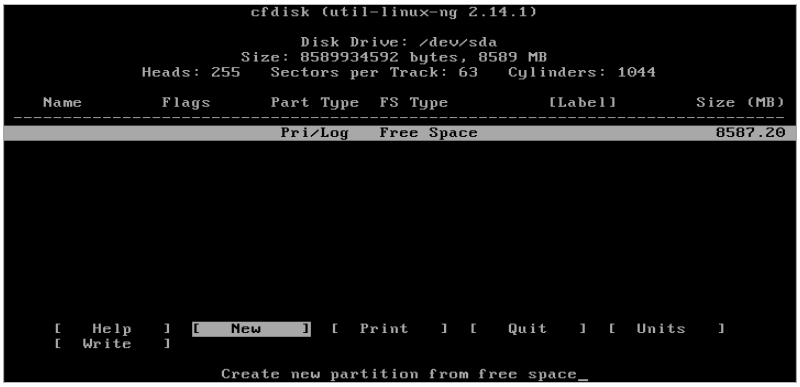
XML (Extensible Markup Language): Término inglés que significa Lenguaje de Marcas Extensible, es un lenguaje de marcas desarrollado por Word Wide Web Consortium (W3C).

Anexo I: Instalación de Android

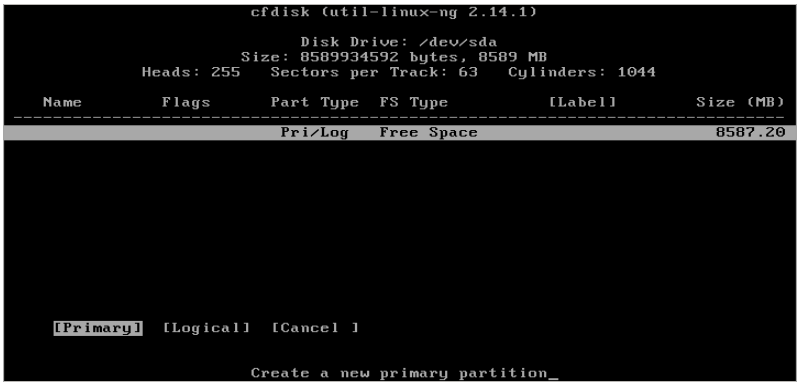


Seleccionar en el grub la línea de inicio "**Installation – Install Android-x86 to harddisk**" y pulsar Enter.

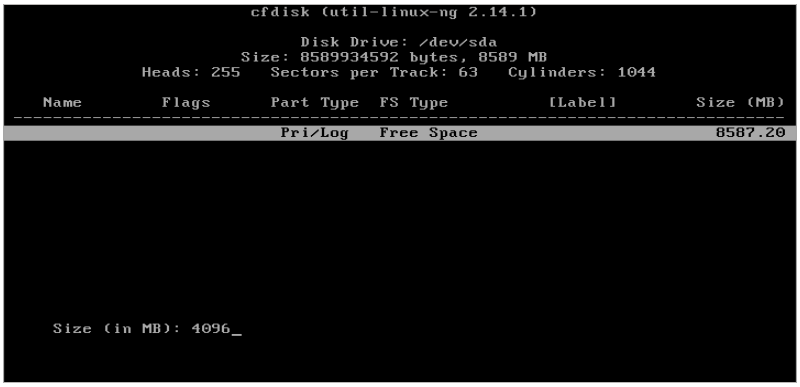
En el menú de selección de la partición para la instalación seleccionar "**Create/Modify partitions**" y pulsar Enter sobre "OK".



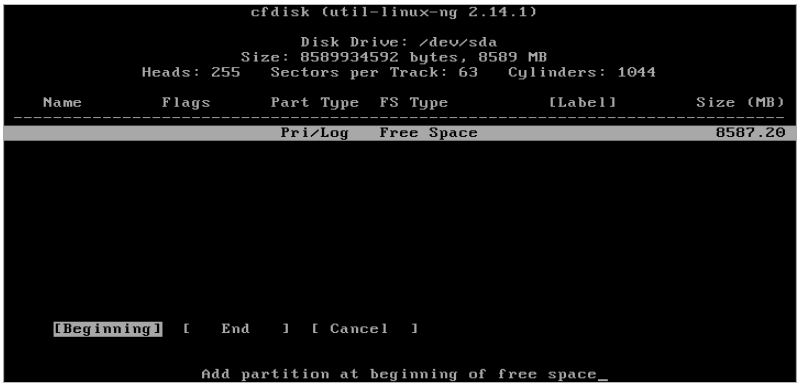
En la utilidad para la creación y modificación de particiones presionar Enter sobre "**New**".



En la utilidad para la creación y modificación de particiones presionar Enter sobre "**Primary**".

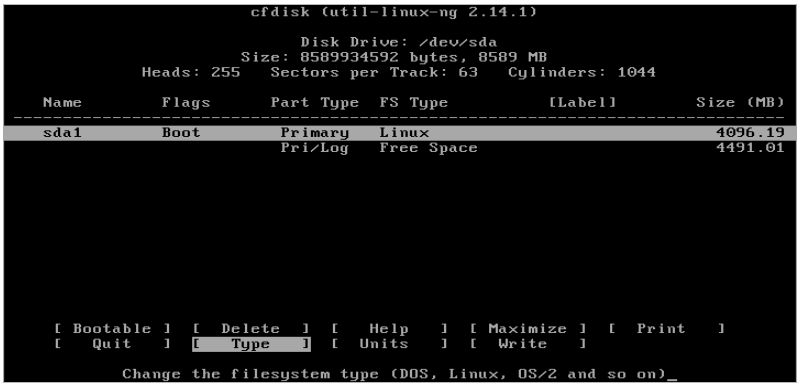


En la utilidad para la creación y modificación de particiones escribir un valor adecuado para el tamaño de la partición donde se instalará Android (en el ejemplo 4096 MB) y presionar Enter.

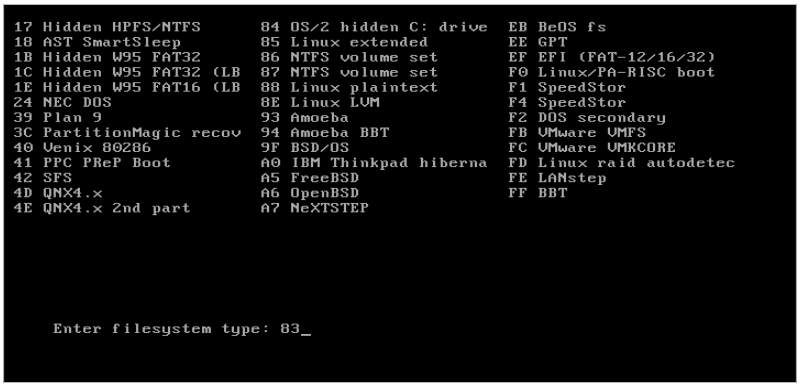


En la utilidad para la creación y modificación de particiones presionar Enter sobre "**Beginning**".

En la utilidad para la creación y modificación de particiones estando ubicado sobre la nueva partición que se creó (en el ejemplo sda1) pulsar Enter sobre "**Bootable**".

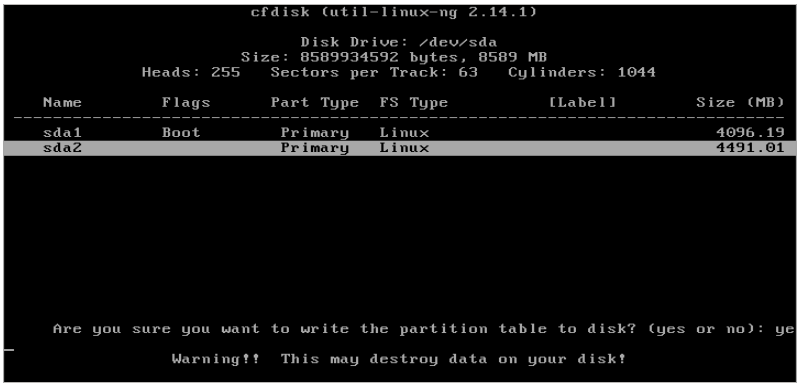


En la utilidad para la creación y modificación de particiones estando ubicado sobre la nueva partición que se creó (en el ejemplo sda1) pulsar Enter sobre "**Type**".

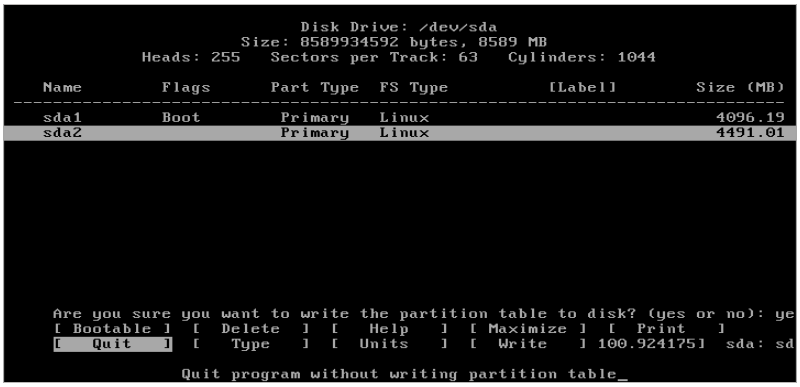


En la utilidad para la creación y modificación de particiones se especifica como sistema de ficheros el número "**83**" que corresponde con el sistema de ficheros de Linux.

De forma análoga se crean todas las particiones que se desee, teniendo en cuenta que no es necesario habilitar la opción de “bootable”. Una vez que se definan todas las particiones a utilizar se presiona Enter sobre “**Write**”.

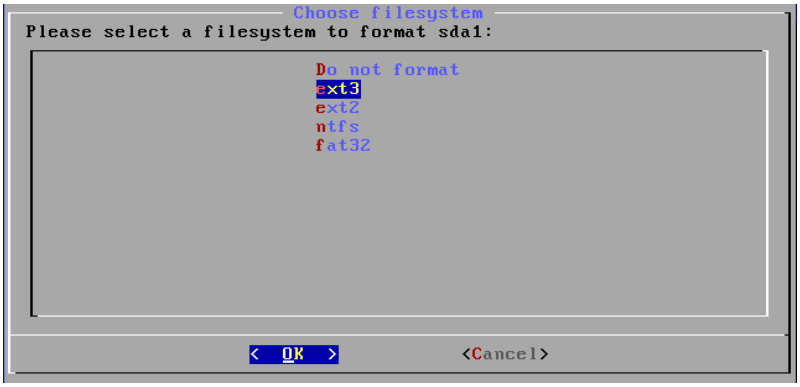


En la utilidad para la creación y modificación de particiones se deben aceptar las modificaciones hechas escribiendo “**yes**” y presionando Enter.

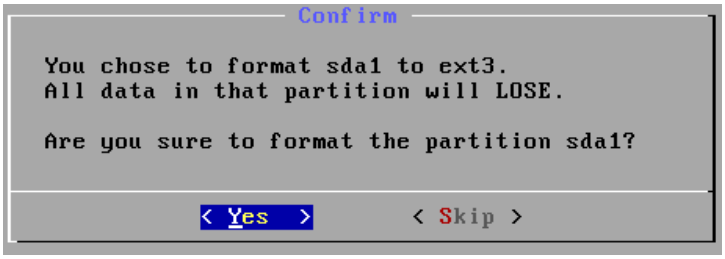


Para salir del asistente para la creación y modificación de particiones se presiona Enter sobre “**Quit**”.

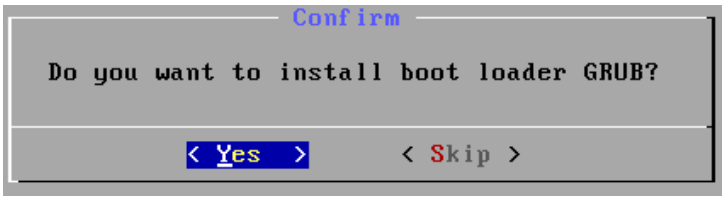
Ahora ya se puede seleccionar la partición para instalar Android (en el ejemplo se instala en sda1). Se ubica sobre la partición a utilizar y se pulsa Enter sobre "**OK**".



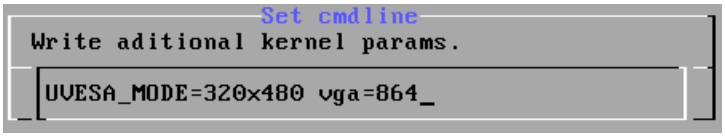
Se selecciona como sistema de archivos "ext3" y se pulsa Enter sobre "**OK**".



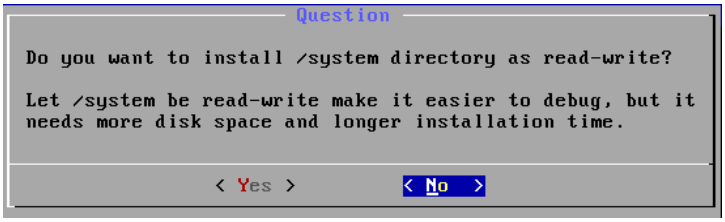
Se confirma la advertencia que se muestra pulsando Enter sobre "**Yes**".



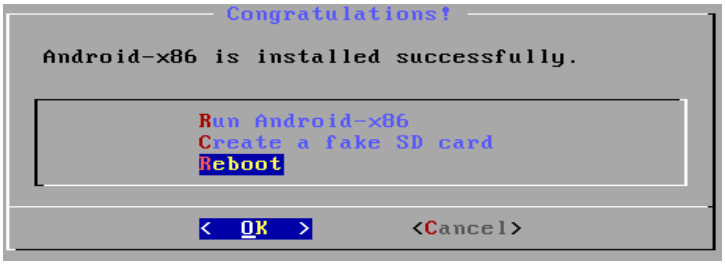
Se acepta la opción de instalar el Grub pulsando Enter sobre "**Yes**".



Se escriben ahora las opciones adicionales del Kernel para que levante con una resolución de pantalla similar a los dispositivos móviles escribiendo en el diálogo "**UVESA_MODE=320x480 vga=864**" y aceptando al pulsar Enter.



Se responde No a la pregunta de instalar el directorio del sistema con las propiedades de lectura/escritura pulsando Enter sobre "**No**".



Realizadas estas acciones ya se puede iniciar la máquina virtual pulsando Enter estando ubicado en la opción "**Reboot**" sobre **OK**.