Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

Tipo de artículo: Artículo original
Temática: Software libre
Recibido: 15/04/2016 | Aceptado: 05/05/2016

Android para escritorio

Android for desktop

Juan Manuel Fuentes Rodríguez [1*], Allan Pierra Fuentes[1], Abel Fírvida Donestevez [1], Héctor Pérez Baranda[1], Alexis López Zubieta[1], Luis Daniel Sierra Corredera[1]

[1] Universidad de las Ciencias Informáticas. Carretera a San Antonio Km 2 ½, Reparto Torrens, Municipio La Lisa, La Habana Cuba. {jfuentesr, apierra, firvida, hbaranda, azubieta, ldsierra}@uci.cu

[*] Autor para correspondencia: jfuentesr@uci.cu

Resumen

En los últimos cinco años, Android prácticamente ha monopolizado los dispositivos móviles, sin embargo, este sistema operativo no ha logrado profundizar en el área de los equipos de cómputo convencionales a pesar de los esfuerzos de equipos de desarrollo multinacionales como Android x86. Esto se debe en gran medida debido a la sustancial diferencia entre la metáfora de escritorio utilizada en los entornos más consagrados, como Windows, MacOS y las diferentes variantes de ambientes de trabajo de Linux, y la arquitectura de información propuesta y optimizada para dispositivos móviles de Android. El presente trabajo acerca el ambiente de trabajo de Android a la metáfora de escritorio, mediante la incorporación a dicho sistema operativo de un gestor de ventanas desarrollado con la utilización del *framework XPosed*. Esto supone aproximar y aprovechar una plataforma de más de un millón y medio de aplicaciones oficiales en la tienda en línea, una alianza de varias de las mayores potencias de la industria de software y las telecomunicaciones y uno de los productos con mayor gestión de la seguridad a las computadoras de escritorio.

Palabras clave: Android, Metáfora de escritorio, Gestor de Ventanas

Abstract

In the last five years the mobile device market has been monopolized by Android, however the desktops market is still away from its reach despite of the efforts of multiple multinational development teams such as Android X86. This is produced mainly by the huge difference between the Android's information architecture and the traditional desktop

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

metaphor of Windows, MacOS and GNU/Linux distributions. The current work aims to approach the Android's information architecture to a traditional desktop metaphor by the addition of a windows manager developed using the XPosed framework. With this improvement will be possible to use in desktop computers an operative system with better security management and more than a million and a half of applications available in its market and will also get the benefits from an alliance of the major leaders in the software and telecommunications industry.

Keywords: *Android, Desktop Metaphor, Window Manager*

Introducción

Android es un sistema operativo de código abierto basado en Linux y patrocinado por la *Open Handset Alliance* (OHA), una alianza multinacional que comprende ochenta y cuatro empresas líderes en el área de las telecomunicaciones, software y hardware (*OHA, 2015*), que aúna numerosos esfuerzos para su desarrollo y el de los dispositivos en los que se ejecuta. Este ha tenido una evolución exponencial en los últimos cinco años, pues ya cuenta con casi dos millones de aplicaciones en *Google Play* (*The Statics Portal, 2015*), convirtiéndose en el sistema con la mayor tienda en línea (Figura 1), y domina más de la mitad del mercado de los dispositivos móviles (*Net MarketShare, 2015*).

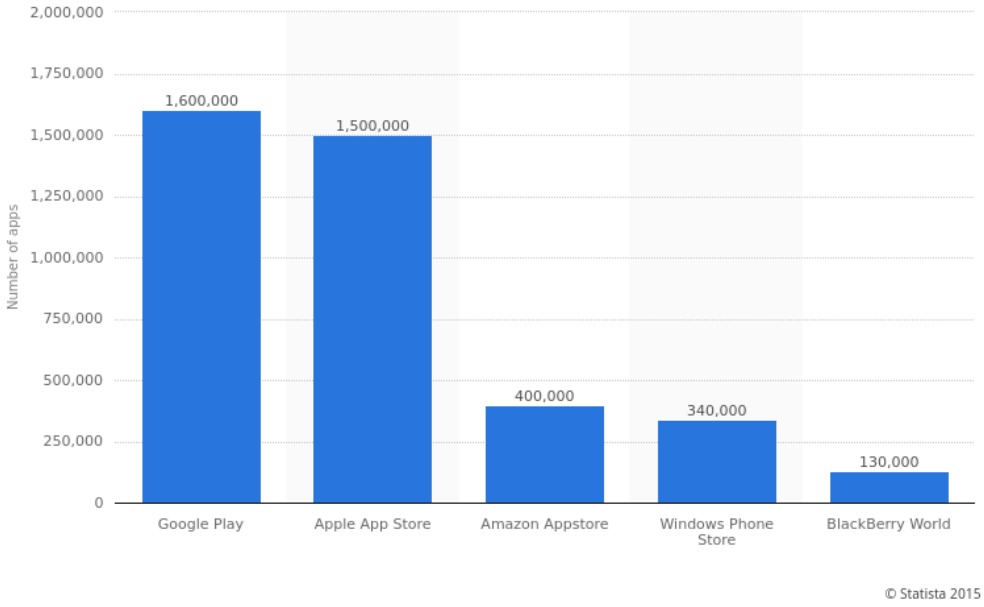

Figura 1. Aplicaciones oficiales disponibles en las diferentes tiendas en línea (*The Statics Portal, 2015*).

Actualmente existe un proyecto llamado Android x86 que se aúna esfuerzos para portar Android a los equipos de cómputo que utilicen dicha plataforma (AOSP, 2015). Este equipo empezó publicando parches para el soporte para x86 del sistema operativo y luego fueron evolucionando hasta mantener su propio rama de un sistema operativo basado en AOSP[1]. Hasta la fecha han realizado catorce lanzamientos oficiales, desde la versión 1.6 hasta la 5.1 de Android.

A pesar de que Android puede ser ejecutado en computadoras convencionales, su interfaz está diseñada para maximizar el uso de la pantalla, pues está orientada a dispositivos de pequeño y mediano tamaño, razón por la que no sigue las metáforas y estándares aplicados en los entornos de computadoras de escritorio, atentando contra la experiencia de usuario en este ambiente. Un ejemplo de lo mencionado es la ausencia de un gestor de ventanas convencional como los de los entornos de escritorio tradicionales que permiten mostrar varias ventanas a la vez, intercambiarlas, redimensionarlas y decorarlas. En vez de esto, Android muestra una única actividad a pantalla completa para la interacción, en caso de que se desee cambiar a alguna de las anteriores solo se puede hacer mediante el botón de las aplicaciones recientes reduciendo la productividad del usuario.

Actualmente existen soluciones que intentan mitigar el problema como *MultiWindow* de Samsung y *Dual Window* de LG que muestran dos actividades al mismo tiempo, permitiendo arrastrar y soltar contenido de una ventana a otra (Figura 2).

Figura 2: Sistema Samsung MultiWindow (S*amsung, 2015*).

[1] AOSP: Proyecto de Software Libre de Android (conocido por AOSP por sus siglas en inglés).

Estas soluciones no resuelven el problema especificado pues solo funcionan en un reducido número de dispositivos y al no ser libres no se pueden portar a Android x86. Además, su concepción sigue estando enfocada a pantallas de medianas dimensiones y no todas las aplicaciones pueden ser ejecutadas en este modo.

El objetivo del trabajo que se presenta es materializar un gestor de ventanas para Android con el cual se consumaría una de las características principales de los entornos de escritorio en el ambiente de usuario de este sistema operativo.

Materiales y métodos

En esta sección se brinda un acercamiento al funcionamiento del sistema operativo Android, así como a su sistema gráfico y a su gestor de ventanas. Además, se explica la solución propuesta y se discuten los resultados obtenidos durante la investigación.

El sistema operativo Android

Android es un paquete de software que tiene como propósito principal crear una plataforma abierta disponible para los fabricantes de equipos y desarrolladores (*AOSP, 2014*). Fue creado por la compañía *Android Inc* y actualmente es patrocinado por la OHA, un consorcio de compañías lideradas por Google (*OHA, 2015*). Android está diseñado primariamente para dispositivos con pantalla táctil, aunque recientemente se han liberado versiones para televisores, carros y dispositivos inteligentes de vestir.

La arquitectura de Android está compuesta por capas (Figura 3). En la base del sistema está un núcleo Linux con algunas adiciones como un sistema de gestión de la memoria especializado en el ahorro de la misma (*AOSP, 2015*). Luego se encuentra el entorno de ejecución de Android, compuesto principalmente por la máquina virtual *Dalvik*, y las bibliotecas nativas que dotan al sistema operativo de gran parte de sus capacidades (*Brahler, 2010*). Encima se encuentra el marco de trabajo, que provee una Interfaz de Programación de Aplicaciones (API) para varias áreas como la red, multimedia e interfaz de usuario. En este ámbito se proporcionan además los diferentes servicios utilizados por las aplicaciones y sus gestores y es donde se realiza el encapsulamiento de los permisos de cada aplicación (*Brahler, 2010*). La capa superior está formada por el software con el que el usuario interactúa directamente.

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

Figura 3: Arquitectura de Android (*Yaghmour, 2013*).

Un peldaño muy importante en esta arquitectura es *Dalvik*, la máquina virtual de Java para Android. Esta está diseñada para dispositivos embebidos y permite ejecutar códigos de bytes generados a partir de aplicaciones basadas en Java y los propios componentes de Android, proporcionando una vía para interconectarlos con el resto del sistema, incluyendo las bibliotecas nativas (*Yaghmour, 2013*).

Sistema gráfico de Android

El proceso de inicialización del sistema gráfico de Android comienza desde la invocación del *script init.rc* por parte del núcleo. Es aquí donde, entre otras instrucciones, se inicializan las variables de entorno, se montan los sistemas de archivos y se inician los servicios nativos.

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

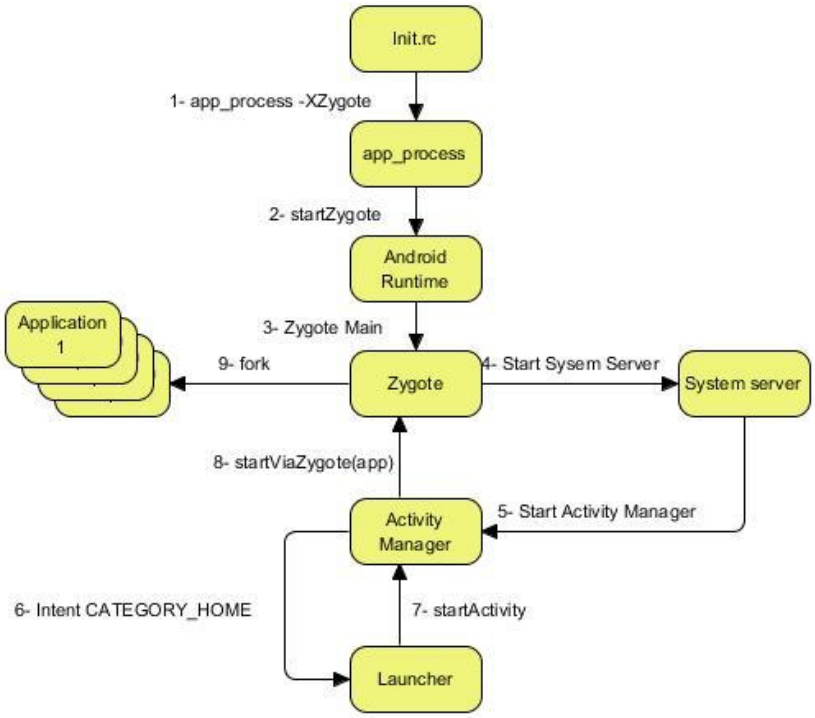

Figura 4: Inicialización y funcionamiento de Zygote.

Uno de esos servicios nativos es el *Zygote*, el responsable del sistema gráfico. Este es iniciado a través del ejecutable *app_process* luego de levantado el entorno de ejecución de Android y la máquina virtual *Dalvik*. Al nacer, este servicio carga en memoria todos los recursos que un software de Android pueda necesitar con el objetivo de mejorar el proceso de apertura del mismo. A partir de este momento *Zygote* abre un socket *(/dev/socket/zygote*) por el cual se escucha la apertura de nuevas aplicaciones (*Yaghmour, 2013*) y ejecuta, sin necesidad de una petición, el servicio *System Server* que se encarga de inicializar todos los servicios disponibles en el sistema.

Una vez inicializado el ambiente de usuario, al ejecutar una aplicación se realiza una llamada al método *startActivity* de la clase *Activity Manager*, que realizará una petición por el socket utilizando el procedimiento *startViaZygote*. Al recibir una petición *Zygote* ejecuta la solicitud mediante la creación de un proceso hijo, a los cuales les suministra todos los recursos cargados desde el inicio del dispositivo.

Gestores de ventanas de Android

Un gestor de ventanas es un paquete de software que ayuda al usuario a monitorear y controlar diferentes contextos mediante la separación física de sus actividades (comúnmente llamada ventanas) en una o varias pantallas (*Myers*,

1988). Además, permite el encapsulamiento de los métodos de entrada y salida de los diferentes procesos de las actividades y las acciones con las ventanas.

En Android, como se aprecia en la Figura 3, los componentes más importantes para la gestión de las ventanas son *Surface Manager*, *Activity Manager* y *Window Manager*. *Surface Manager* es una biblioteca nativa que provee un compositor que maneja la representación de todas las superficies directamente en el *framebuffer*[2] y permite combinar superficies de dos y tres dimensiones de varias aplicaciones (*Android Engineering Application & Consulting Services Team, 2009*). Este utiliza a *OpenGL ES*[3] y la aceleración de dos dimensiones del propio hardware para realizar el proceso de composición.

Activity Manager es el servicio que se encarga de iniciar los servicios y las aplicaciones través de *Zygote*, gestiona el ciclo de vida de las actividades en Android, emite los *Intents* y resuelve los proveedores de contenido (*Yaghmour, 2013*). Además, interviene en varios de las acciones para las optimizaciones del *kernel* como el controlador de poca memoria.

Window Manager es un servicio del sistema que gestiona la lista de ventanas ordenadas y es el responsable de resolver cuál de estas es visible y cuál es su puesto en la pantalla (*Android Engineering Application & Consulting Services Team, 2009*). Entre sus funcionalidades también se encuentran crear las superficies de las ventanas utilizando a *Surface Manager*, tramitar los eventos de entrada y las transiciones entre las ventanas.

El framework XPosed

El *framework XPosed* o *XPosedBridge* es una biblioteca que permite reemplazar los métodos deseados de las aplicaciones en tiempo de ejecución sin la necesidad de alterar las mismas. Esta funciona gracias a la capacidad de Android de cargar todos los recursos de las aplicaciones antes de que el sistema haya terminado de iniciar (*XPosed, 2015*).

Además, la biblioteca contiene su propio ejecutable del *app_process*, que sitúa al *framework* como una de las dependencias de *Zygote*, por lo que cada proceso hijo de este tendrá a *XPosed* cargado (*XPosed, 2014*). Una vez ejecutada la aplicación, *XPosed* sobrescribirá en tiempo de ejecución las funciones a reemplazar por las definidas en sus módulos.

[2] Frame Buffer: dispositivo gráfico que representa los píxeles como ubicaciones en la memoria.

[3] OpenGL ES: variante de OpenGL diseñada para dispositivos embebidos

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

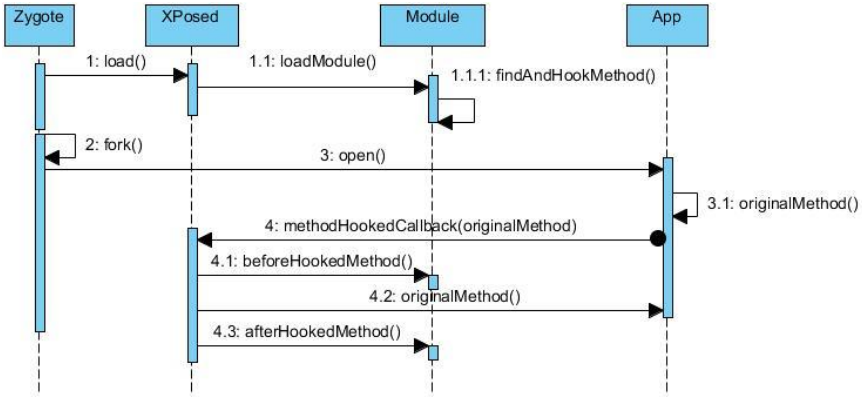

Figura 5: Funcionamiento de XPosed.

Los módulos de esta biblioteca definen los métodos a enlazar o sobrescribir y se identifican mediante los metadatos especiales de *XPosed* en el fichero *AndroidManifest.xml*. Para enlazar un método en específico se necesita conocer de antemano su nombre, sus parámetros y la ruta de la clase que lo contiene (por ejemplo *com.android.systemui*) y transferírselos al procedimiento *findAndHookMethod* junto con una implementación de la clase abstracta *XC_MethodHook*. Esta última provee los métodos *beforeHookedMethod* que es llamado antes del original y *afterHookedMethod* que es llamado luego y ambos reciben como parámetro una instancia de *MethodHookParam* que contiene toda la información del procedimiento original (*XPosed, 2014*). En caso de que se requiera reemplazar completamente se debe emplear *XC_MethodReplacement* como parámetro a *findAndHookMethod*.

El uso de *XPosed* tiene la ventaja de que no existe la necesidad de modificar el código fuente original de Android o de una aplicación para lograr los objetivos deseados, funciona en todas las versiones de AOSP desde el lanzamiento 4.0.3 hasta la 5.1 y la posibilidad de instalar módulos o utilizar los existentes (*XPosed, 2015*). Además, las modificaciones hechas son reversibles con facilidad (con solo desinstalar en módulo y reiniciar basta) y es software libre.

Resultados

El principal resultado de esa investigación es la conceptualización de un gestor de ventanas para el sistema operativo Android, que brinde las funcionalidades básicas de sus homólogos para escritorio. Para lograr este objetivo se desarrolló un módulo de *XPosed* basado en *XHaloFloatingWindow* (*XHaloFloatingWindow, 2015*) que sirviera como prueba de concepto del sistema propuesto.

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

La Figura 6 muestra el diagrama de clases del módulo desarrollado, dentro de los que sobresalen *MainXposed*, *HaloFloting* y *MovableWindow* pues en estos se realiza prácticamente toda la modificación al sistema para que sirva como gestor de ventanas.

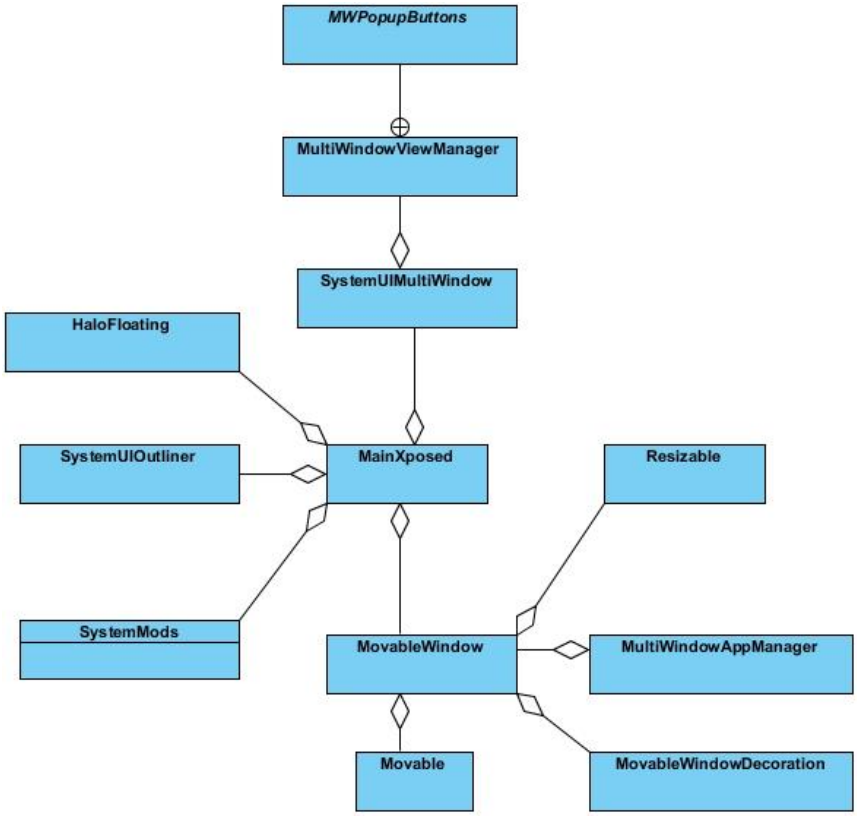

Figura 6: Diagrama de clases del módulo de XPosed.

El gestor de ventanas basa su funcionamiento en la bandera *0x00002000* de los *Intents*, que permite abrir las ventanas en modo diálogo flotante. Conociendo el hecho de que todos estos son ejecutados e interpretados desde la clase *ActivityRecord* del servicio *Activity Manager*, se sobrescribió el constructor de esta clase para que todas las aplicaciones se ejecutaran en modo flotante (con excepción de las ventanas del *Launcher* y el *SystemUI* que tienen que ser obligatoriamente a pantalla completa para el correcto funcionamiento del ambiente de usuario).

Revista Cubana de Ciencias Informáticas
Vol. 10, No. Especial Informática 2016, Mes mayo 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 82-93

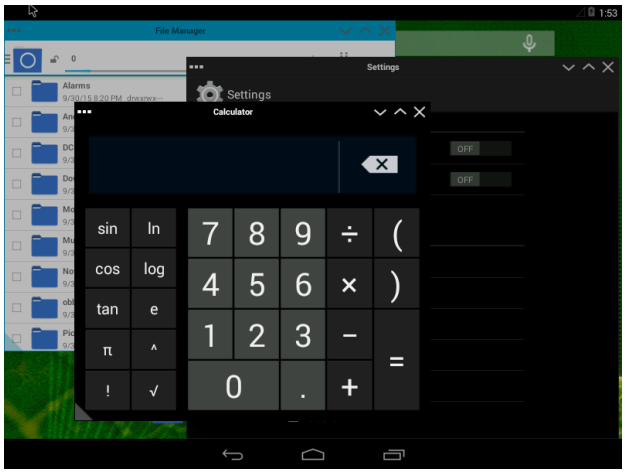

Figura 7: Ventanas flotantes de Android.

La decoración de las ventanas se realiza mediante la modificación a la clase *Window*, para que el componente raíz de una ventana sea un marco compuesto por un borde tradicional y la vista raíz en donde las actividades colocarían sus propios componentes.

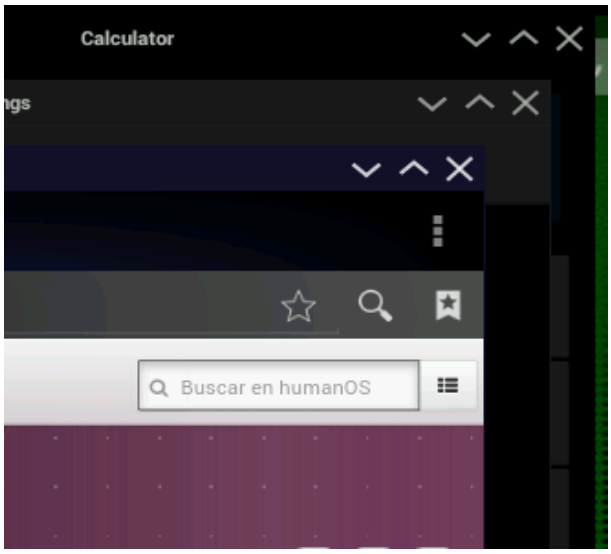

Figura 8: Border de las ventanas flotantes.

Para adicionar la funcionalidad de mover y redimensionar las ventanas, el gestor de ventanas aprovecha la capacidad del servicio *Window Manager* de Android de gestionar la posición de las actividades en la pantalla y solo fue necesario la implementación de un evento de arrastrar y soltar sobre la barra de título para que cambie la posición de la actividad y sobre los bordes para cambiar su tamaño.

Conclusiones

En la presente investigación se realizó la conceptualización de un gestor de ventanas tradicional para Android. Eso tiene gran importancia pues mejora la usabilidad del sistema en equipos con pantallas de gran tamaño, permitiendo al usuario organizar las actividades a su preferencia. El resultado impacta directamente el socioadaptabilidad del sistema pues brinda a los usuarios un ambiente similar a entornos tradicionales como el de Windows.

La solución propuesta supera a las mencionadas anteriormente pues permite que todas las actividades sean ejecutadas en modo ventana y no solo las definidas por los fabricantes, no posee un límite de aplicaciones ejecutadas al mismo tiempo y realiza la decoración de las ventanas.

Con esta incorporación de este resultado, y futuras investigaciones como un entorno de escritorio tradicional, se acerca el ambiente de usuario de Android a las metáforas y estándares utilizados en los sistemas operativos para computadoras de escritorio.

Agradecimientos

A la Universidad de las Ciencias Informáticas y al equipo de trabajo de Nova.

Referencias

1. **Open Handset Alliance.** Open Handset Alliance. [En línea] 2015. [Citado el: 10 de Octubre de 2015]. Disponible en: http://www.openhandsetalliance.com/.

2. **The Statistics Portal.** Number of apps available in leading app stores as of July 2015. [En línea] Julio de 2015. [Citado el: 30 de Septiembre de 2015]. Disponible en: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

3. **Net Market Share.** Market share for mobile, browsers, operating systems and search engines. [En línea] Septiembre de 2015. [Citado el: 10 de Octubre de 2015]. Disponible en: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1.

4. **Android Open Source Project.** The Android Source Code. [En línea] 2015. [Citado el: 10 de Octubre de 2015]. Disponible en: https://source.android.com/source/index.html.

5. **Samsung.** Nueva función multiwindow. [En línea] 2015. [Citado el: 12 de octubre de 2015]. Disponible en: http://www.samsung.com/es/consumer/flagship/tutorial_galaxy_note3/note3/tutorial/new_multiwin_index.html.

6. **Yaghmour, Karim.** *Embedded Android.* Sebastopol: O'Reilly Media, Inc, 2013. 978-1-449-30829-2.

7. **Android Open Source Project.** Android Interfaces and Architecture. [En línea] 2015. [Citado el: 10 de Octubre de 2015]. Disponible en: https://source.android.com/devices/index.html.

8. **Brahler, Stefan.** *Analysis of the Android Architecture.* Karlsruhe: s.n., 2010.

9. **Myers, Brad A.** *A Taxonomy of Window Manager User Interfaces.* 5, s.l.: IEEE Computer Graphics and Applications, 1988, Vol. 8.

10. **Android Engineering Application & Consulting Services Team.** Android Anatomy and Physiology. [En línea] Noviembre de 2009. [Citado el: 15 de Septiembre de 2015]. Disponible en: http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf.

11. **XPosed.** XPosed Insaller | XPosed Module Repository. [En línea] 2015. [Citado el: 15 de Septiembre de 2015]. Disponible en: http://repo.xposed.info/module/de.robv.android.xposed.installer.

12. **Xposed.** Development tutorial . [En línea] 12 de Agosto de 2014. [Citado el: 1 de Octubre de 2015]. Disponible en: https://github.com/rovo89/XposedBridge/wiki/Development-tutorial.

13. **XHaloFloatingWindow.** XPosed Module Repository. [En línea] 21 de Julio de 2014. [Citado el: 3 de octubre de 2015]. Disponible en: http://repo.xposed.info/module/com.zst.xposed.halo.floatingwindow