



# Desarrollo de aplicaciones móviles

Dr. Raúl Valente Ramírez

Ing. Raúl Fuentes Samaniego

Ing. Mirna Mendez



TECNOLOGICO  
DE MONTERREY.



# EL MERCADO

# Nicho de mercado

- Principal consumidor: Mujeres
- Edad de los consumidores mayoritarios: 20-29 años
- Dichos consumidores tienden a tener estudio universitario (Collage) con un ingreso inferior a los \$625 dls mensuales.

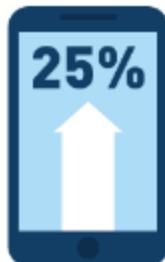


# Nicho de mercado (II)

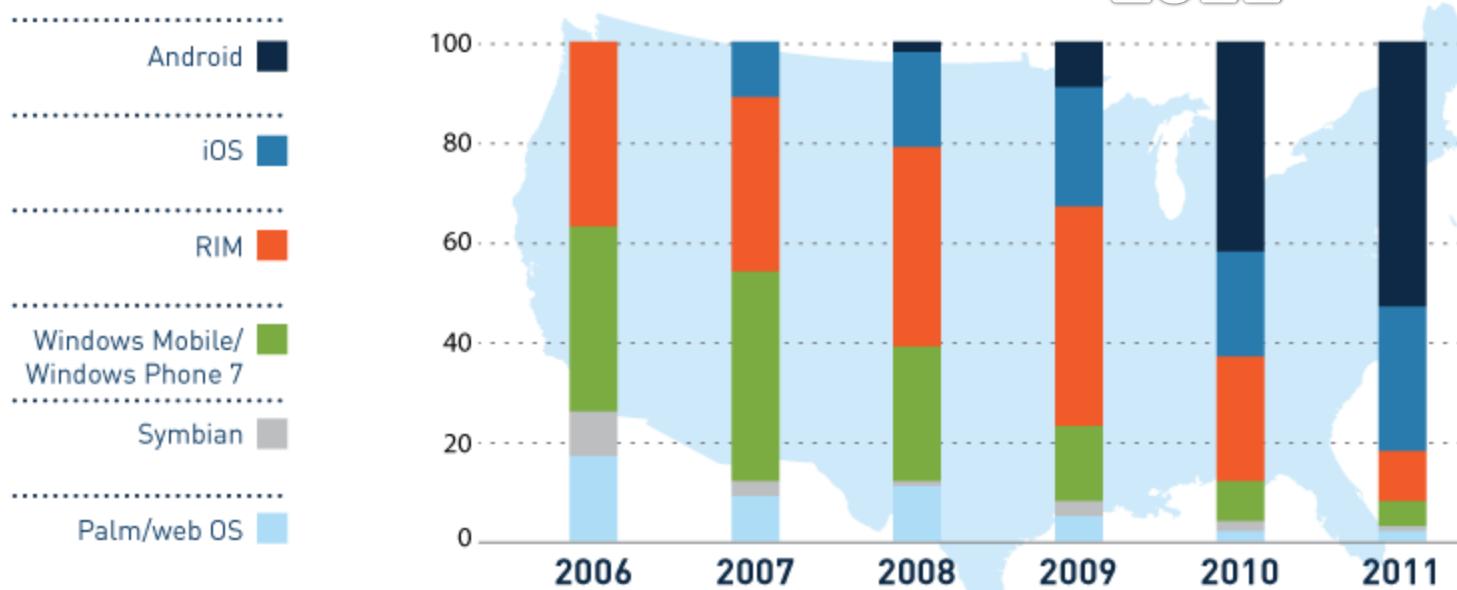
- Características:
  - Mas de 5 años usando smartphones
  - Utilizan principalmente “Built-in apps” (Ya incluidas)
  - Se utiliza principalmente para jugar o entretenerse en horas de oficina y/o clases (“*relaxing and relieving stress*”)



# Mercado (Datos 2012)



Ventas globales de Smartphone se esperan que crezcan 25%. De 472 millones en el 2011 a 630 millones en el 2012



En el 2010 Microsoft cambia de S.O. móvil



En el 2011 RIM quedó incomunicado por mas de 48 horas por un fallo técnico en el backbone

# Mercado II



Solo Android consiguió un mercado superior a 200 mil apps disponibles en Dic 2011. Se espera 58 mil millones para 2016 (27 mas que el esperado para Iphone)



Ciudadanos del G10 Mundial son los principales consumidores de APPS (IOS & Android) PERO el tercer mundo esta adquiriendo rápidamente posición en el mercado.

57 % E.U.A.

2% Hong Kong
2% Australia
3% Canadá
3% R.U.
4% Taiwán
12% China
17% Demás países



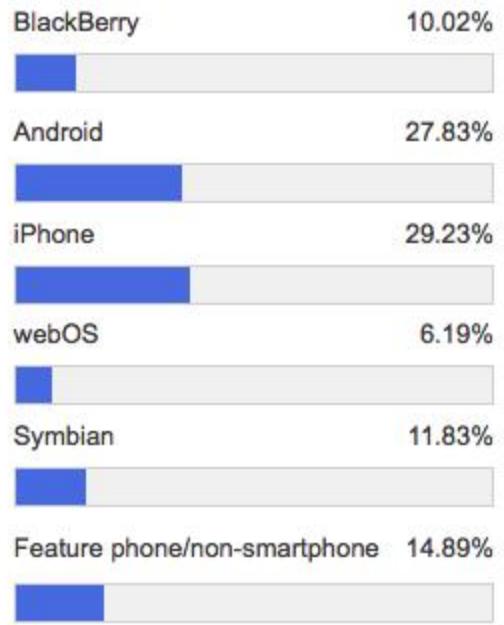
Lugar de origen de descargas de APPs (IOS & Android)



Se espera que China tome Saltos sin precedentes debido a su rápido crecimiento. Solamente, de **Enero 2011 a Octubre 2011**, los usuarios de apps móviles se **incremento un 870%** en dicha región



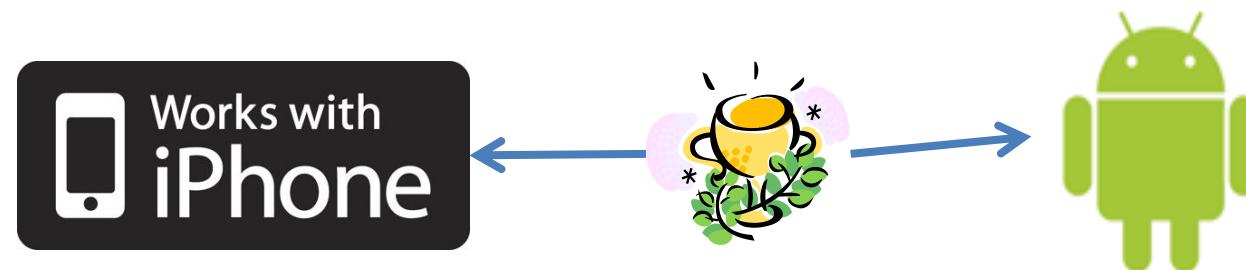
### What did you use prior to Windows Phone?



# ¿Nativo o multi-plataforma?

*"El 2010 fue el año en el que el desarrollo de las aplicaciones móviles comenzaron a cruzar la línea divisoria entre adoptadores tempranos y el mainstream"*

Jeffrey Hammond



- En síntesis cada vez mas se crean aplicaciones nativas con soporte de múltiples plataformas. (Contra deseo de Sys Admin dentro de empresas)
- Tecnologías web móviles como WAP (Wireless Application Protocol), XHTML-Mobile Profile y Java ME (Micro Edition), se “desvanecen rápidamente”

# Aplicaciones móviles

- Apps desarrolladas para dispositivos “de mano (handheld)
- Suelen estar pre-instaladas en dichos dispositivos o ser descargados por los usuarios de plataformas de distribución de software móvil.





# Desarrollo de aplicaciones



open handset alliance  
ANDROID  
**developers**



## CÓDIGOS EJEMPLOS

Existe una gran cantidad de material disponible y gratuito en los sitios oficiales de Android.



## TUTORIALES

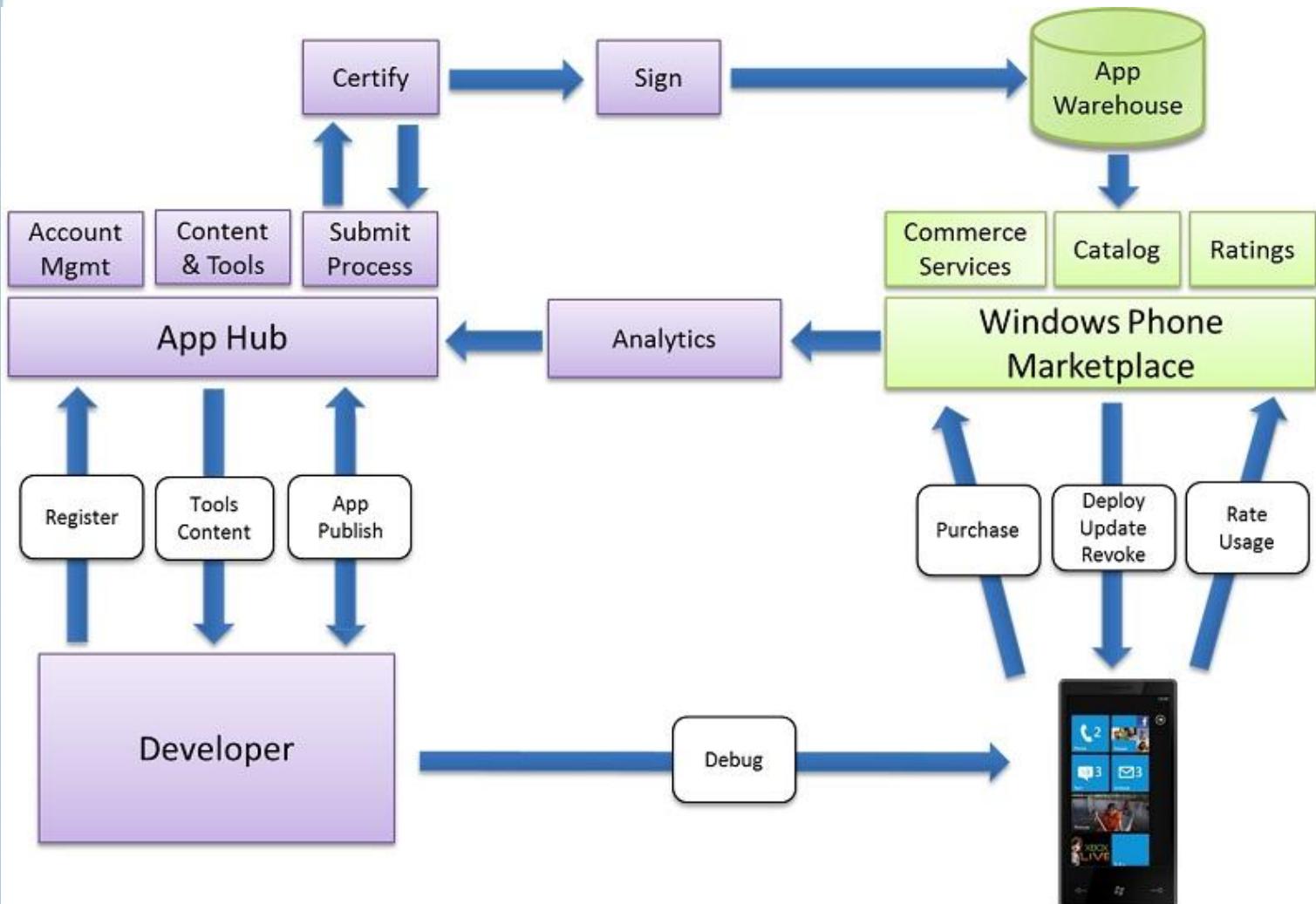


## ARTICULOS

<http://developer.android.com>



# Ciclos de Desarrollo de una App





# Recursos para el desarrollo de Apps

Microsoft®  
**tech·ed**  
Online

Sessions On-Demand & Community

[www.microsoft.com/teched](http://www.microsoft.com/teched)

**Microsoft®** Learning

Microsoft Certification & Training  
Resources

[www.microsoft.com/learning](http://www.microsoft.com/learning)

**Microsoft® TechNet**

Resources for IT Professionals

<http://microsoft.com/technet>

**msdn®**

Resources for Developers

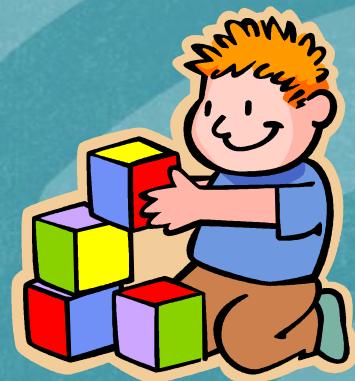
<http://microsoft.com/msdn>

# BB - Plataformas de desarrollo

Plataforma		Descripción
C\C++ Native SDK		Ideal si se tiene una aplicación o juego codificado en C/C++ y se desea exportarlo a las plataformas PlayBook y BB10.
HTML 5 WebWorks		Para aquellos que tienen habilidades con Javascript/CSS/HTML se pueden crear Apps para toda las plataformas actuales de smartphones, PlayBook y el BB10.
Java, BlackBerry Java		Fuertemente vinculado con BB. Permite integrar las App con la experiencia mas fuerte.
Java, Android Runtime		Permite exportar (re-package) y distribuir una App previamente existente para Android a las plataformas BB10 y PlayBook
ActionScript, Adobe AIR		Permite exportar Apps desarrolladas en AIR a Playbook y BB10.
Themes, Theme Studio		Permite la creación de temas personalizados, sean hechos desde cero o mediante templates existentes.



TECNOLOGICO  
DE MONTERREY.



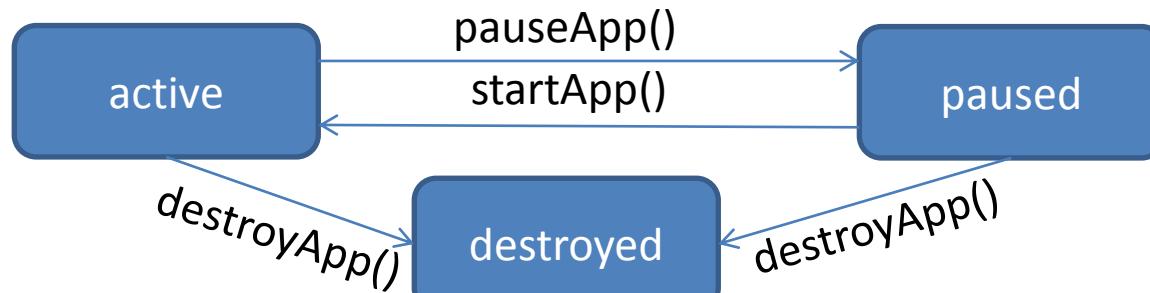
# SISTEMAS OPERATIVOS Y APLICACIONES MÓVILES

# Ciclos de vida de apps

Difieren del ciclo de vida aplicaciones de escritorio.



- Transición entre distintos estados no están etiquetados con acciones abstractas (Ej: *start* o *stop*), si no con nombres de métodos .
- Dichos métodos son métodos de *callback* ejecutados por el S.O. cuando la aplicación cambia de estado. Esto permite que el desarrollador tenga posibilidad de reaccionar ante cambios o eventos
- Bajo este esquema el desarrollador solo necesita realizar un “*override*” al método “*lifecycle*” correspondiente.



# Importancia del manejo adecuado de lifecycles

- Escenario ejemplo:
  1. Usuario escribiendo e-mail
  2. Ocurre una llamada entrante.
  3. El S.O. dispara un callback para detener la App de e-mail que esta enfocada (En pantalla pues) y abre la App de teléfono. Por este callback el pauseAPP() del e-mail es ejecutado.



**MALO:** El desarrollador no usa ese evento para guardar el texto escrito por el usuario, dicho texto se pierde.



**BUENO:** El usuario puede continuar con el texto donde lo dejó.



Cada vez que una app deja de ser enfocada se ejecuta un callback.



Los ciclos de vida de una App suelen ser presentadas como el modelo lifecycle Con su documentación

# S.O. contemporáneos (I)

	iOS	Win Phone	Android	BlacBerry	WebOS
Tipo de Archivo	App	Apx	Apk	Cod, alx	IPK
Elemento clave	Human Interface	Metro	Seguridad	Privacidad	Sencillez Web
¿Multitasking?	Si***	Si	Si	Si	Si
Quien controla los Permisos	Capacidad del App (Auditado)	Capacidad del App (auditado)	22 bloques por el usuario	Internos de RIM	Capacidad del App (auditado)
Fabricante de dispositivos.	Apple	Nokia *	varia	RIM	HP (& Palm)**

\* Microsoft & Nokia tienen convenio pero no es de exclusividad.

\*\* HP esta por liberar WebOS como open-Source por lo tanto puede cambiar.

\*\*\* A partir del iOS 4 fue que se introdujo Multi-tasking.

# S.O. contemporáneos (II)

	iOS	Win Phone	Android	BlackBerry	WebOS
Lenguaje Nativo (Apps)	Objective-C	C#	C/C++	Java	HTML5+CSS+Javascript
Aparición	2007	2010	2008	2004*	2009**
Kernel	Darwin	Windows CE	Linux	Java	Linux
Procesador (Original)	ARM (v6 y v7-A)	ARM	ARM, MIPS, PowerPC	ARM (Xtel)	ARM

\* Aparición del primer modelo de Blackberry del tipo “smartphone”

\*\* WebOS fue liberado en el 2009 por Palm, 2010 por HP y en el 2012 como Open Source



# ANDROID

Desarrollado por



(AKA: Google)

- “Open source” bajo la licencia de Apache
- Apps suelen estar construidas en Java compuestas de actividades y servicios.
- Diseño de múltiples capas para garantizar seguridad y flexibilidad para una plataforma abierta protegiendo al usuario final.
- El enfoque en seguridad fue pensado en desarrolladores y sus creaciones (hacerles la vida fácil) además de darle control y flexibilidad al usuario sobre las aplicaciones.



# Bloques claves de la plataforma Android

Bloque	Descripción
Device Hardware	Android puede correr en una gran variedad de configuraciones de hardware tales como: Smartphones, tabletas y set-top-boxes.
Android O.S.	El cuerpo del S.O. esta construido sobre el kernel de Linux. Todos los recursos del dispositivo son accedidos por medio del S.O.
Android Application Runtime	Las apps en Android suelen estar programadas en lenguaje JAVA y ejecutándose sobre la maquina virtual Dalvik. Pero además existen aplicaciones (incluyendo las principales de android) que son nativas. Sin importar el origen, toda las aplicaciones son ejecutadas sobre un <i>sandbox</i>



# Arquitectura de seguridad

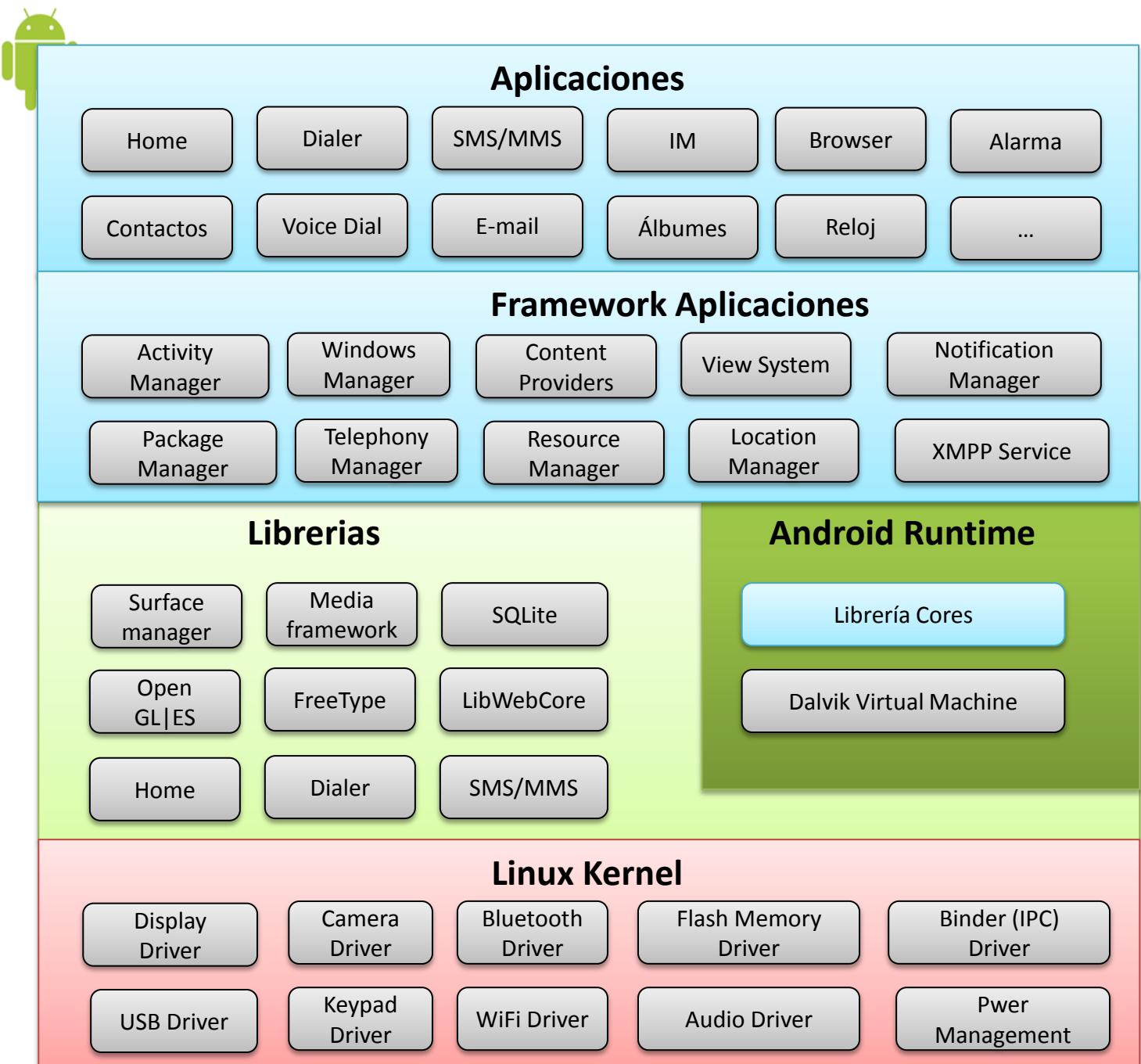
## Objetivos:

- Proteger los datos del usuario.
- Proteger los recursos del sistema.
- Proveer aislamiento de aplicaciones

## Metodología:

- Seguridad robusta a nivel del S.O. por medio del Kernel de Linux
- Sandbox mandatorio para todas las Apps
- Comunicación inter-proceso segura.
- Firmas de aplicación (Application signing)
- Permisos “Application-defined” y dados por el usuario.







# Nivel de seguridad del Sistema y Kernel

## Elementos claves del Kernel de Linux

- Modelo de permisos basados en usuario.
- Aislamiento de procesos
- Mecanismos extendible para IPC seguro.
- Capacidad de remover partes del Kernel
- Seguridad Linux en ambientes multi-usuario



Mengano no puede leer lo de Fulano, tampoco puede tocar sus recursos (Hardware y/o software)



Debido a la implementación del *Sandbox* es igual de seguro usar código nativo o cualquier otro lenguaje para el desarrollo de Apps.

IPC = Inter-Process communication

# Nivel de seguridad del Sistema y Kernel ( Cont... )

## Partición del sistema

- Compuesto por:
  - kernel de Android
  - Librerias del sistema
  - Apps runtime
  - App frameworks
  - Apps
- Read-Only
  - En **Modo Seguro** solo Apps Core estarán disponibles.



Primera versiones de Android guardan el password del dispositivo en texto plano.

## Sistema de Archivos

- Cada App es su propio usuario .
- EXT4 es el mas popular.
- >= Android 3.0 **PUEDE** encriptar el sistema de archivo (y archivos)
  - Utiliza AES128 con CBC & ESSIV:SHA256
  - Key generada del password del dispositivo-
  - Password protegido con algoritmo PBKDF2 (SHA1 + Salt sobre el password)



# Cifrado en la miel...

- El mecanismo de cifrado esta basado en **dm-crypt** el cual es una opción del kernel que trabaja sobre el bloque de la capa de dispositivos.
- En Android activar el cifrado es... truculento debido al fuerte uso de GPL se evito usar elementos no públicos.
- El arranque de un sistema encriptado es algo complejo
  1. Si el proceso de inicialización (init) falla en montar la app o cargar datos asume que esta cifrado y manipula propiedades para volverlo a intentar (considerando ahora el cifrado).
  2. Un framework es inicializado para obtener el password del usuario . Si en este punto hay un error el usuario deberá reiniciar las propiedades por defectos del dispositivo.
  3. Se crea una UI para solicitar el password, en este punto se monta el elemento cifrado . El framework anterior muere.
  4. El UI mata todo los servicios de la clase del framework y vuelve a cargar (esta vez todo listo y con cifrado) y es aquí donde el framework de la apps es cargado y listo para operar.



# Manejo de la memoria

Android SDK, compiladores y S.O. utilizan una variedad de técnicas para proteger la memoria, tales como:

- Address Space Layout Randomization (ASLR) para fijar localizaciones claves en memoria de forma aleatoria.
- Hardware-based Non-Execution (NX) para prevenir ejecución de código en el stack y heaps de apps.
- ProPolice para prevenir stack buffer overruns.
- Extensions to OpenBSD's dlmalloc para prevenir ataques tipo “Chunk consolidation” que son un modo común de explotar corrupciones en heaps.
- Linux mmap\_min\_addr() para mitigar metidas de pata en apuntadores nulos al escalar privilegios.



# Rooting of devices

- Solo el kernel y un sub-grupo de apps core poseen permisos de Root.
- Android NO OFRECE RESISTENCIA a que una Apps (usuario) con privilegio de Root modifique cualquier otra parte del sistema.
- Datos encriptados con una llave guardada en el dispositivo (como el password del dispositivo) NO protegen los datos de la App de usuarios con Root.



“Rooting Android” consiste en explotar vulnerabilidades para obtener “Root” pero esto puede ser potencialmente inseguro ya sea por Apps maliciosas o fallas de diseño en las apps.



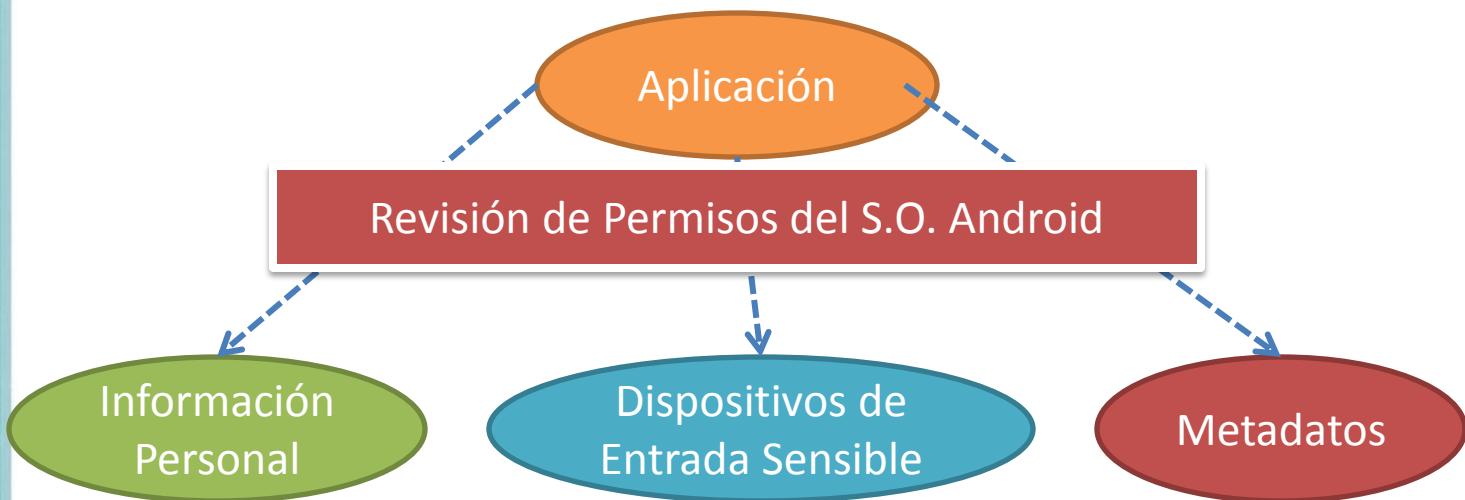
Solo técnicas donde la llave no se encuentre en el dispositivo pueden contrarrestar un poco ese peligro.



# Interprocess Communication & Personal Information

Los procesos se pueden comunicar entre si mediante métodos tradicionales ( Sockets locales, señales, etc...) manteniendo los permisos. Además Android provee nuevos mecanismos:

- **Binder**
- **Services** – (Mediante el uso de Binders)
- **Intents**
- **ContentProvider**



# Diseño del UI

Android provee un Framework específicamente para manipular el aspecto físico de la App para dar la “una experiencia consistente y disfrutable” al usuario final que involucra la experiencia del “*home screen*”, navegación global en el dispositivo y las notificaciones.

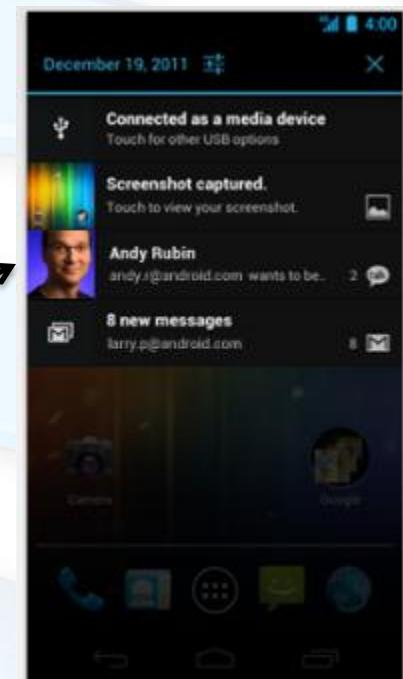
## UI Claves:

- Home, All Apps & Recents
- Barras de sistema
- Notificaciones
- Common App UI



Las UI pueden cambiar al cambiar la versión de Android.

Notificaciones son los mensajes no críticos de una App que no requieren interrumpir al usuario.



# UI: Home & All & Recents



## Home

Controlado por usuario,  
barra de favoritos, accede a  
toda las Apps & Widgets



## All apps

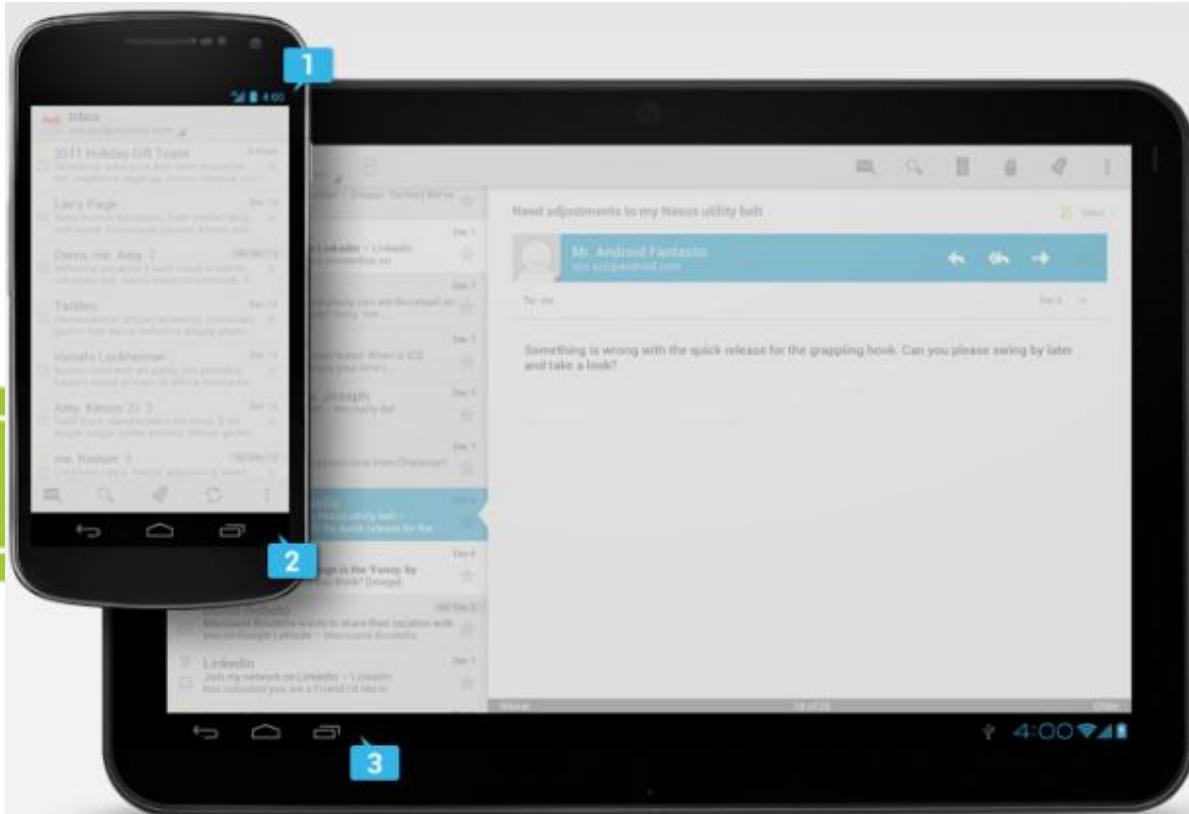
Permite buscar toda las Apps  
& Widgets disponibles.  
Permite transiccion por  
arrastre al Home



## Recent

Permite navegar entre Apps  
recientemente usadas.

# UI: System Bars



## 1 Status Bar

Notificaciones pendientes. Desde aquí se puede acceder (arrastrando) a detalles de notificación.

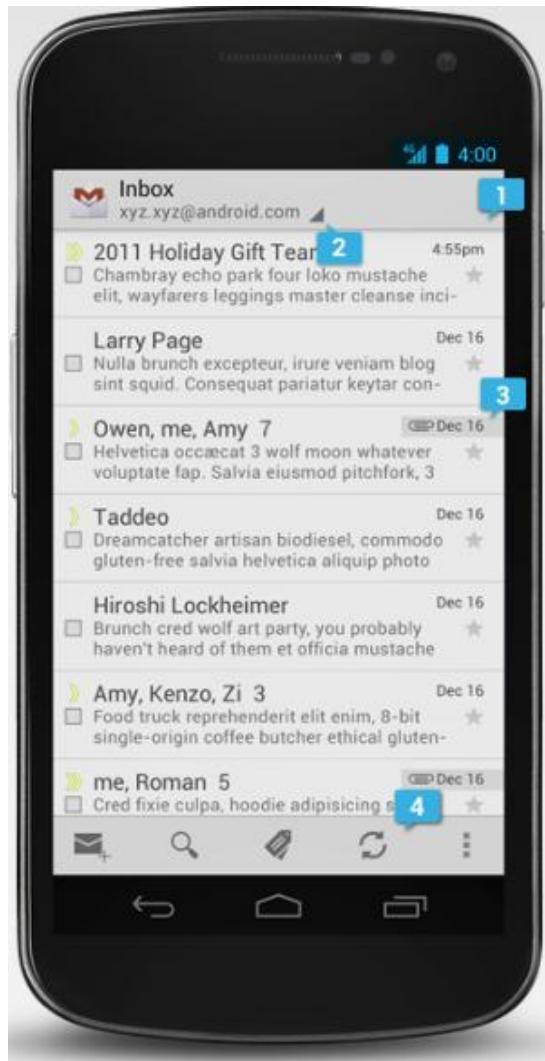
## 2 Navigation Bar (> Android 4)

Solo aparece si el dispositivo no cuenta con botones físicos (Back, Home, Recents)

## 3 Combined Bar

En dispositivos del tipo tableta las dos barras anteriores son combinadas en una sola barra

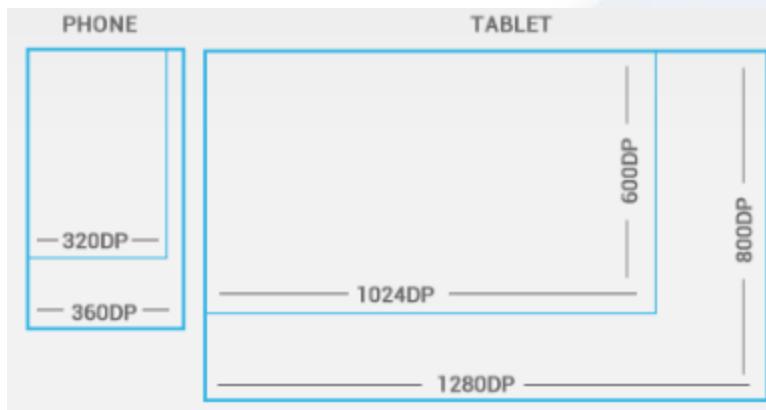
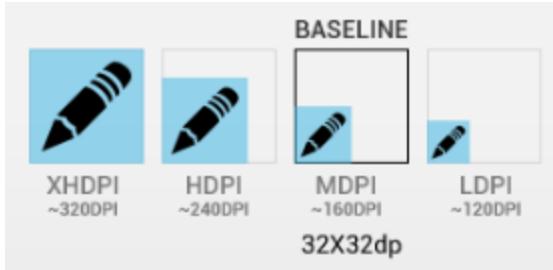
# UI: Apps (Diseño comun)



- 1 Barra de Acción Principal
  - Centro de control y comandos de las Apps. Debe de incluir elementos para poder navegar por la misma (Jerarquía y vistas).
- 2 Control de Vista
  - Permite al usuario cambiar de vistas dadas por la Apps.
- 3 Contenido
  - El espacio donde el contenido de la App es desplegado.
- Barra de “Split Action”
  - Debe de proveer un modo de distribuir las acciones que no pudieron estar en la barra de acción (Por diseño o por conveniencia).



# UI: Otras consideraciones



Text Size Micro	12sp
Text Size Small	14sp
Text Size Medium	16sp
Text Size Large	18sp

- Considerar multiples dispositivos.
- Estilos
- ¡MUCHO ARTE!
- Sinergia de desarrolladores

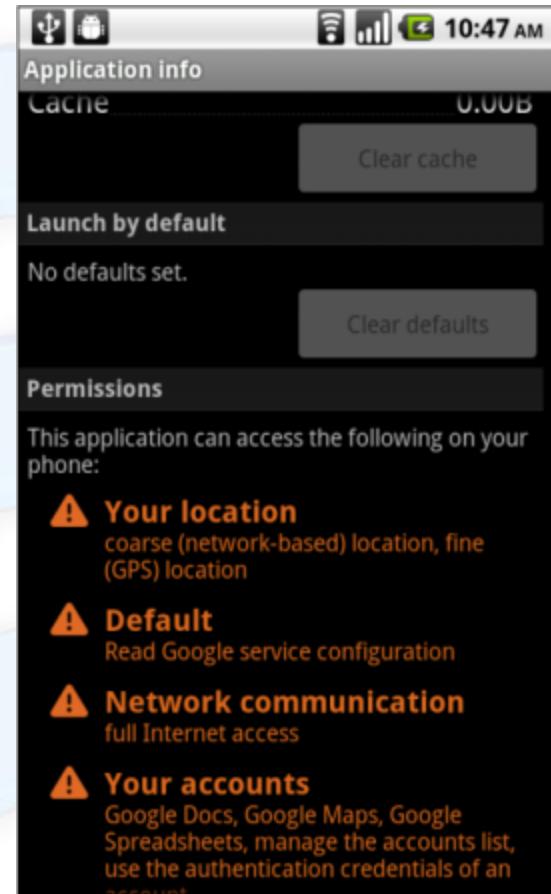
Text Color Primary Dark  
Text Color Secondary Dark

Text Color Primary Light  
Text Color Secondary Light



# Apps: Seguridad

- Los recursos que tienen una App son limitados y están bajo control.
- Un mal manejo de recursos ocasionaría que la experiencia del usuario fuese mala.
- Todo los recursos (API) claves son accedidas solo por el S.O. Por lo tanto cada App debe solicitar permisos al ser instaladas para obtener acceso a tales API's.
- Es posible deshabilitar algunas funciones globales (Wi-Fi, 3G/4G , etc) para toda las Apps.
- El usuario siempre puede ver los permisos dados a una App.
- Actualmente, Existen **22 bloques principales de permisos**



# Apps: Bloques de permisos

|



Nombre	Nivel	Descripción
<b>Servicios con costo</b> (Llamadas y/o SMS/MMS)	Moderado-Alto	Apps como Google voice pueden hacer llamadas o envió de SMS (con o sin informar al usuario)
<b>Almacenamiento</b> (en tarjetas SD)	Alto	Permite Leer/Escribir y borrar CUALQUIER contenido dentro de la tarjeta SD. No hay forma de crear sub-permisos (La App o puede trabajar sobre la SD o no puede).  Apps de respaldo son un excelente ejemplo.
<b>Información personal</b> (Leer datos de contactos)	Alto	Da acceso a los datos del listado de contactos. Típicamente son Apps de redes sociales, typing/note taking, SMS, manejo de contactos, etc...

# Apps: Bloques de permisos

II

Nombre	Nivel	Descripción
<b>Información personal</b> (Lectura/Escritura de datos del Calendario)	Moderado-Alto	Permite el control del calendario. Una App típica es Google Calendar
<b>Phone Calls</b> (Estado del teléfono e identidad)	Moderado-Alto	Permite conocer el estado del teléfono si esta en medio de una llamada activa) pero además permite obtener el numero único IMEI, el IMSI y un ID de 64 bits de Google para identificar al teléfono. Cabe mencionar que en las primeras versiones de Android se trataba de un permiso que toda App poseía.
<b>Localización (GPS)</b>	Bajo	Permite leer datos del GPS. Apps de realidad aumentada o de mapas suelen requerir el permiso.



# Apps: Bloques de permisos

III

Nombre	Nivel	Descripción
<b>Localización (Coarse netowrk based )</b>	Bajo	Permite leer datos de la red SSID para intentar hacer una ubicación del celular a partir del SSID (o los SSID's cercanos). (MUCHO) Menos preciso que GPS
<b>Comunicación de red (Bluetooth)</b>	Medio	Permite el tener control de Bluetooth para crear conexiones o enviar datos a dispositivos conectados.
<b>Comunicación de red (Acceso a Internet)</b>	ALTO	Similar al anterior, permite el comunicarse con cualquier dispositivo dentro de la red (un permiso clave en casi cualquier tipo de malware) Apps con anuncios publicitarios y Apps de redes sociales suelen utilizarlo.
<b>Comunicación de red (Ver estado de red, ver estado de Wi-Fi)</b>	Bajo	Las Apps pueden deducir si se está teniendo acceso a Internet mediante Wi-Fi o 3G/4G.

# Apps: Bloques de permisos

IV

Nombre	Nivel	Descripción
<b>Herramientas de sistema</b> (Prevenir el dispositivo entre en modo “Sleeping”)	Bajo	Apps como reproductores de video, e-readers y parecidos (Apps donde el usuario no interactúe directamente por tiempos largos)
<b>Herramientas de sistema</b> (Modificar configuración global del sistema)	Moderado	Estas configuraciones son aquellas que se encuentran en la ventana de “settings” del dispositivo. Apps típicas son los widgets de control de volumen, notificaciones y manipulación de widgets.
<b>Herramientas de sistema</b> (Configuración de Read Sync)	Bajo	Permite a la APP saber si se tiene datos sincronizados de fondo habilitados (Tales como Facebook o Gmail)

# Apps: Bloques de permisos

V



Nombre	Nivel	Descripción
<b>Herramientas de Sistema</b> ( Escribir configuración de Access Point)	Bajo	Relacionado a la manipulación de AP para Wi-Fi.
<b>Herramientas de sistema</b> ( Automáticamente iniciar en Boot)	Bajo-Alto	Indica si el S.O. debe arrancar la App al inicializar el sistema (Típicamente después de un reinicio).
<b>Herramientas de sistema</b> ( Reiniciar otras aplicaciones)	Bajo-Alto	Permite una App indicarlo al S.O. que debe de matar a otra App. Sin embargo, la App objetivo debe tener capacidad de reiniciarse a si misma para tal evento.
<b>Herramientas de sistema</b> (Obtener Apps en ejecución)	Medio	Permite a una App identificar que otras Apps están corriendo en el teléfono. Apps contra malware, Apps de control de batería suelen utilizarlo.

# Apps: Bloques de permisos

## VI

Nombre	Nivel	Descripción
<b>Herramientas de sistema</b> (Set preferred Apps)	Moderado	Permite que una App marque cual es la App por defecto para alguna tarea en particular. Ej: Al seleccionar un hyper vinculo se abrirá un navegador de red en particular. Usualmente, Apps que remplazan a una por defecto de Android requieren de este permiso (Firefox, Chrome, etc)
<b>Control de Hardware</b> (Controlar vibración)	Bajo	Permite el control total del sistema de vibración del dispositivo.
<b>Control de Hardware</b> (tomar fotografías)	Bajo	Permite controlar las funciones de cámara (Video/Imagen) del dispositivo
Tus cuentas (discover known accounts)	Bajo	Permite identificar que cuentas están registradas en el dispositivo (Google, Facebook, Twitter, etc...)





# Apps: Composición

- Sin importar el origen del código fuente y desde donde se ejecuta toda App es instalada desde un simple archivo con extensión .apk
- Los bloques de una App Android son los siguientes:
  - **AndroidManifest.xml** – Archivo de control que indica al sistema que hacer con todo los componentes de la capa mas alta, incluyendo los otros bloques de la App.
  - **Actividades** - Usualmente es el código para una sola tarea “*user-focused*” y por lo general involucra una “*User Interface*”.
  - **Servicios** –Código ejecutado de fondo o en el contexto del proceso de otra aplicación (Se ejecuta mientras otra App se ejecuta).
  - **Broadcast Receiver** - Objeto instanciado cuando un mecanismo IPC (Intent) es iniciado por El S.O u otra app. Ej: Cuando una App recibe el aviso de poca batería en el dispositivo podría alterar su funcionamiento.



# Actividades

- La actividad estará en ejecución (o corriendo) si es la aplicación “enfocada” (no siempre es el caso).
- Una actividad está pausada si no se encuentra enfocada por el usuario, aunque en ocasiones puede seguir estando visible.
- Una actividad está detenida, si no se halla visible pero sigue existiendo en el fondo. En este estado, todos sus objetos aún residen en memoria.
- En el estado “shutd down” ningún objeto de la actividad se hallará en memoria.



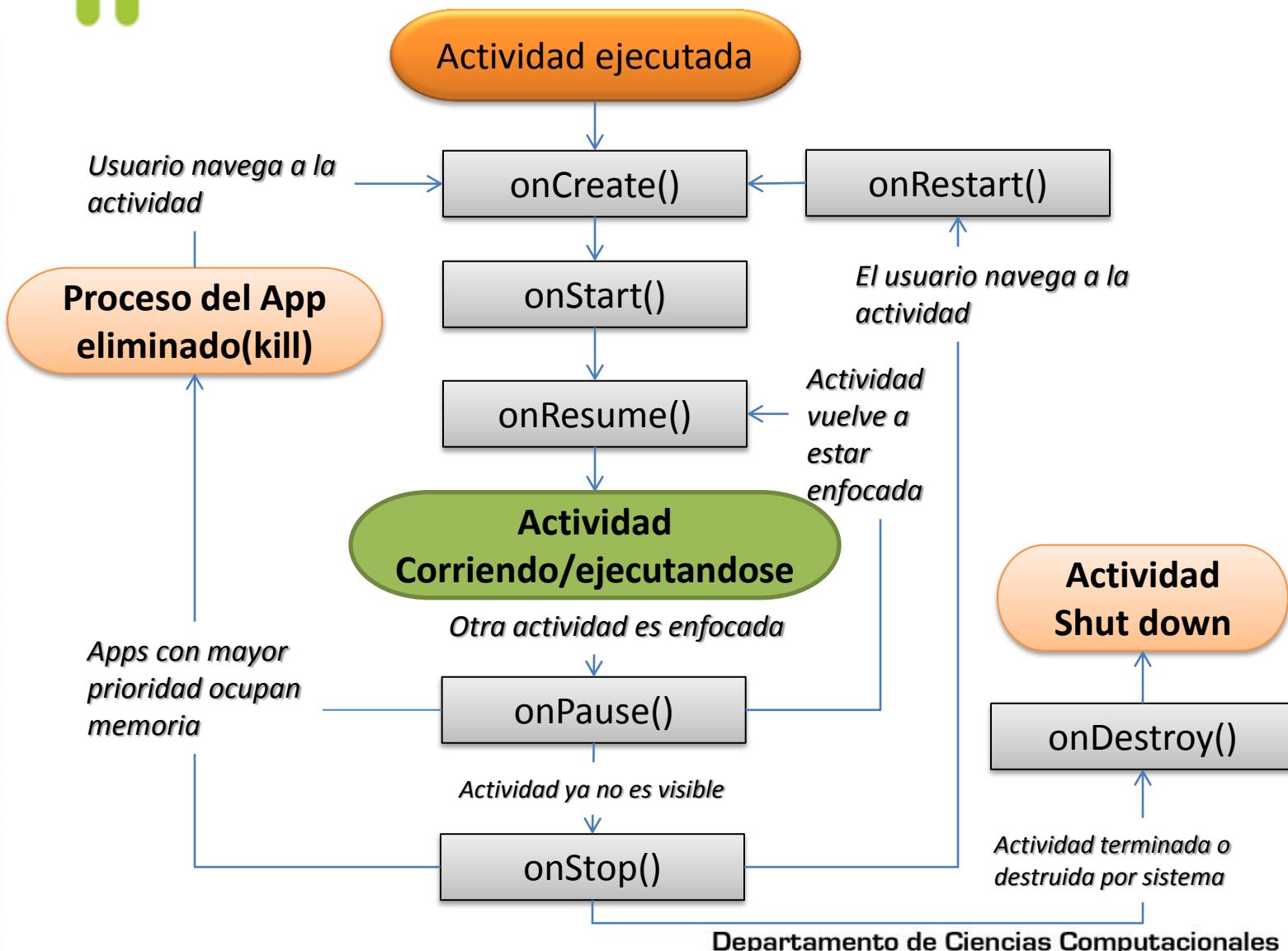
En estados pausado y detenido es posible matar el proceso y liberar recursos.



Los servicios se ejecutan de fondo.



# Lifecycle “Actividades”





# Lifecycles callbacks

```
public class ExampleActivity extends Activity
{
    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }

    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }

    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is
now "resumed").
    }
}
```

```
@Override
protected void onPause() {
    super.onPause();
    // Another activity is taking focus (this
activity is about to be "paused").
}

@Override
protected void onStop() {
    super.onStop();
    // The activity is no longer visible (it is
now "stopped")
}

@Override
protected void onDestroy() {
    super.onDestroy();
    // The activity is about to be destroyed.
}
```

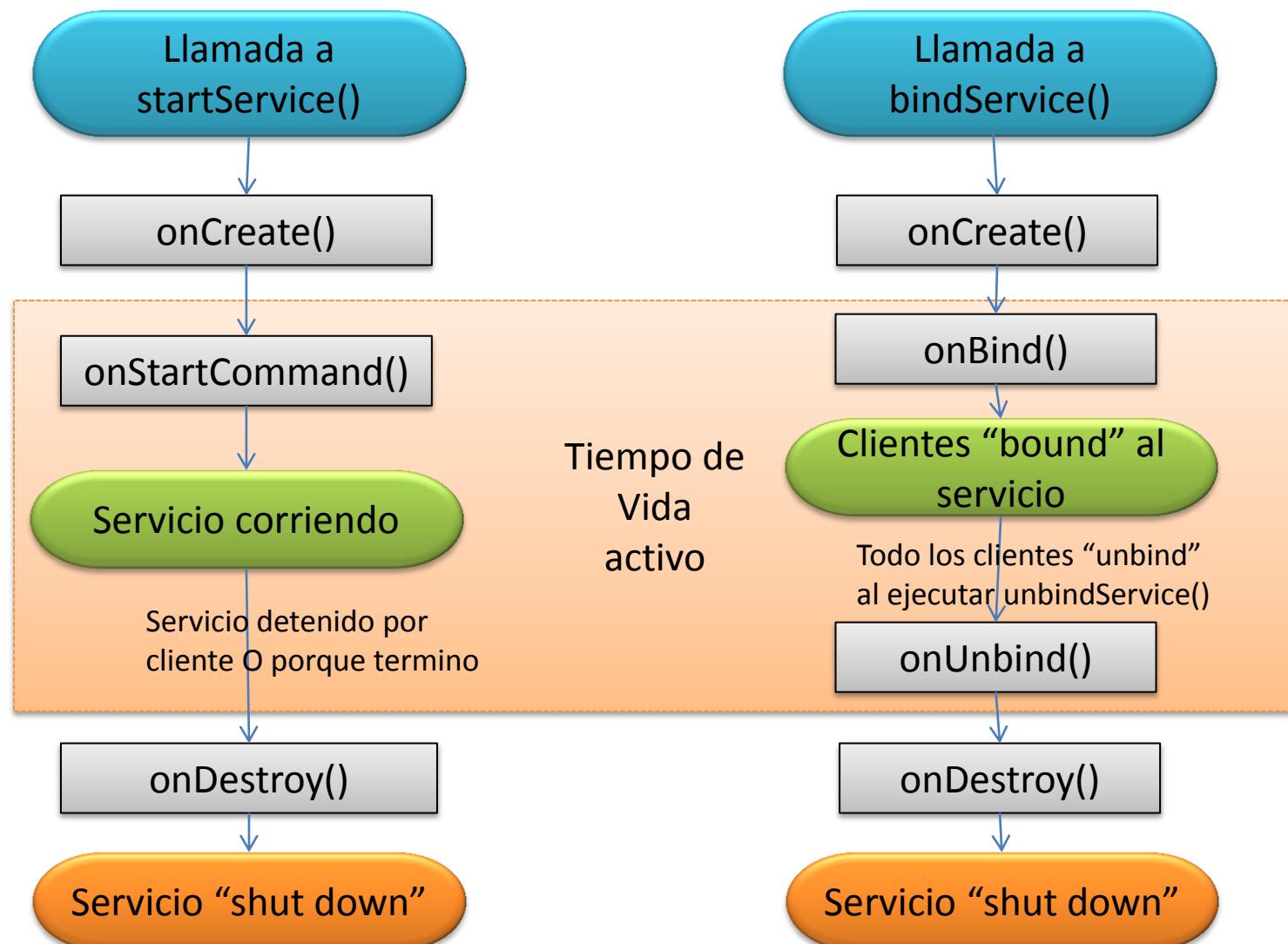


# Servicios

- Componente de una aplicación que corre de fondo sin poseer ninguna GUI.
- Diseñado para ejecutarse por lapsos largos de tiempo y no se ve afectado cuando una *actividad* deja de estar enfocado.
- Posee dos formas:
  - **Iniciado** : Se inicia cuando un componente de una aplicación llama `startService()` , el servicio es independiente al estado de dicho componente “padre”.
  - **Bound** : La atadura ocurre cuando un componente de la app invoca `bindService()`. Existe para dar una interfaz cliente/servidor y multiples componentes pueden usar un mismo servicio “atado”



# Lifecycle de Servicios





# Apps last phase: Signaling

- Firmar el código es el ultimo paso (obligatorio por Google) entre otras cosas permite:
  - Responsabiliza al desarrollador de su app.
  - Valida que la App en el “*Android Market*” no este alterada.
  - Es fundamental para el “*App Sandbox*”
  - Apps pueden ser firmados por OEM, mercados alternos al “*Android Market*” por lo tanto no requieren de un sistema centralizado de firmas digitales.

Debatible





# S.O. Windows Phone



## Windows Phone

- Basado en Win CE, INCOMPATIBLE con antecesores (Por motivos de eficiencia y rendimiento).
- Nombre original: **Windows Phone 7 Series** (Acotado posteriormente)
- Liberado mediados del 2010.
- **NOKIA** se suma al sistema el 11 de Feb 2011
  - Integrando “*OVIS Store*” con “*Windows Phone Store*”
  - Integrando “*Nokia maps*” con “*Bing Maps*”

Windows Phone

“Tango” update



Windows Live

XBOX LIVE

zune



## Windows Mobile

- Basado en Win CE.
- Apareció con el Pocket PC 2000 como un “suite” de Apps desarrollado con Win API
- Enfocado para empresas
- De acuerdo al hardware era el S.O.
  - Windows Mobile Professional (Smartphone + touchscreen)
  - Windows Mobile Standard (Smartphone)
  - Windows Mobile Classic (Pocket PC)





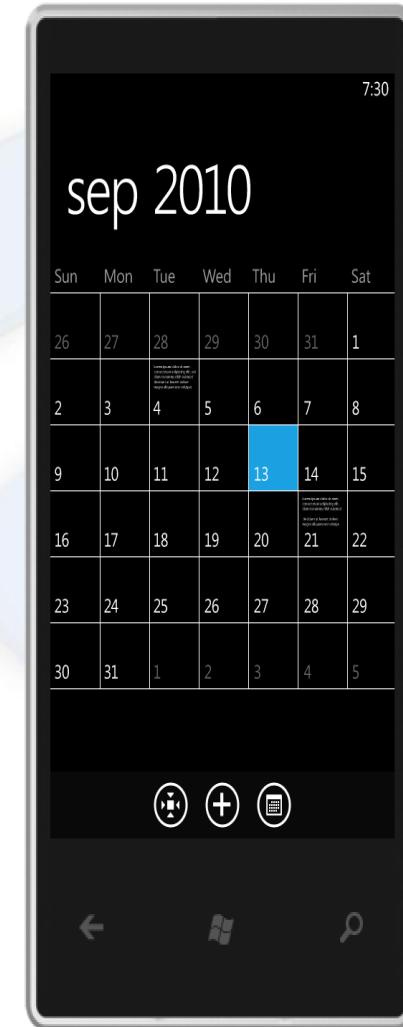
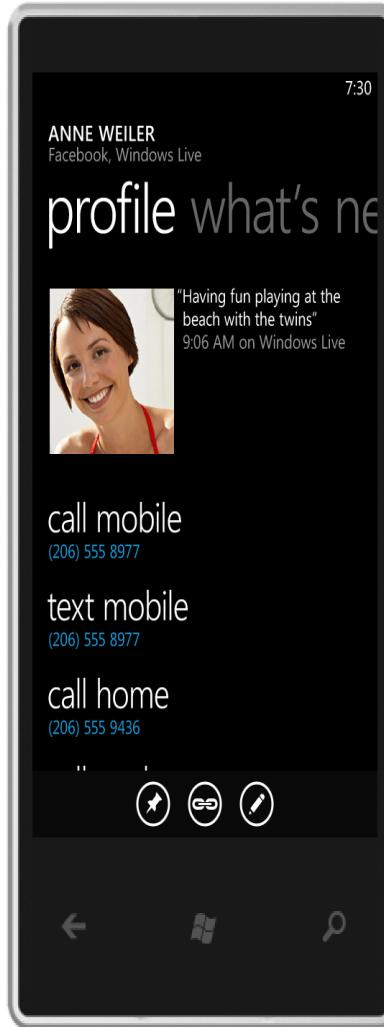
# Bloques claves

<b>Hardware</b>	<b>Hardware Foundation</b> equivalente a <b>HAL</b> de los S.O. de PC.
<b>Frameworks</b>	<b>Silverlight</b> basados en XAML <b>XNA</b> – Para apps basadas en Loops (Videojuegos) <b>Silverlight/XNA</b> – Uso simultaneo de ambos (A partir de <b>Tango</b> )
<b>Apps</b>	Microsoft tiene sus <b>objetivos claves</b> en este <b>bloque</b> . Todo el S.O. gira alrededor de la facilidad para desarrollo y “excelente” experiencia de usuario.
<b>Servicios en la nube</b>	Permite el trabajar con los servicios claves de Microsoft en la nube (Windows Live, XBOX Live, Zune), su objetivo es hacer transparente el servicio al usuario (No importa el smartphone, o tableta , PC, debe tener lo mismo con facilidad).

XAML = Extensible Application Markup Language (de Microsoft)

# Windows Phone Design System

Nombre clave: “Metro”





# METRO

*Metro es uno nombre código para nuestro lenguaje de diseño. Le llamamos "METRO" porque es moderno y limpio. Es rápido y en movimiento. Es sobre contenido y tipografía y es completamente autentico.*

## Microsoft Team

- Metro no solo es “como se ve y se siente” la pantalla si no que también maneja las animaciones en 2D (en la App, entre Apps, por la interfaz, etc) y el sonido emergente del Smartphone.
- Inspirado de las señales de tránsitos en lugares públicos con sus fuertes colores.
- El objetivo es dejar todo fácil, sin interfaces complicadas, con espacio entre cada cosa, que **“El contenido se vuelva la interfaz”**
- Metro puede ser fácilmente integrado a cualquier App (existen templates, sonidos y demás material para ello).



Windows 8

Metro será aplicada en el nuevo S.O. de Microsoft

# Información == Interfaz

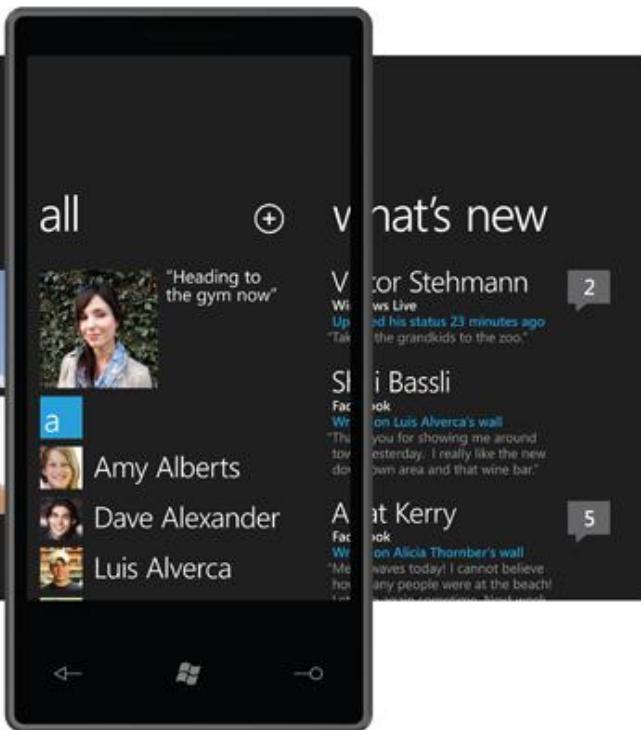
A través de Baldosas en vivo(tiles) se puede mover entre los hubs Me (Usuario), gente, fotos & video, música, juegos y búsqueda. Todos estos hubs estarán entrelazados en un flujo panorámico.

people

recent



Las Apps pueden no estar diseñadas con Metro



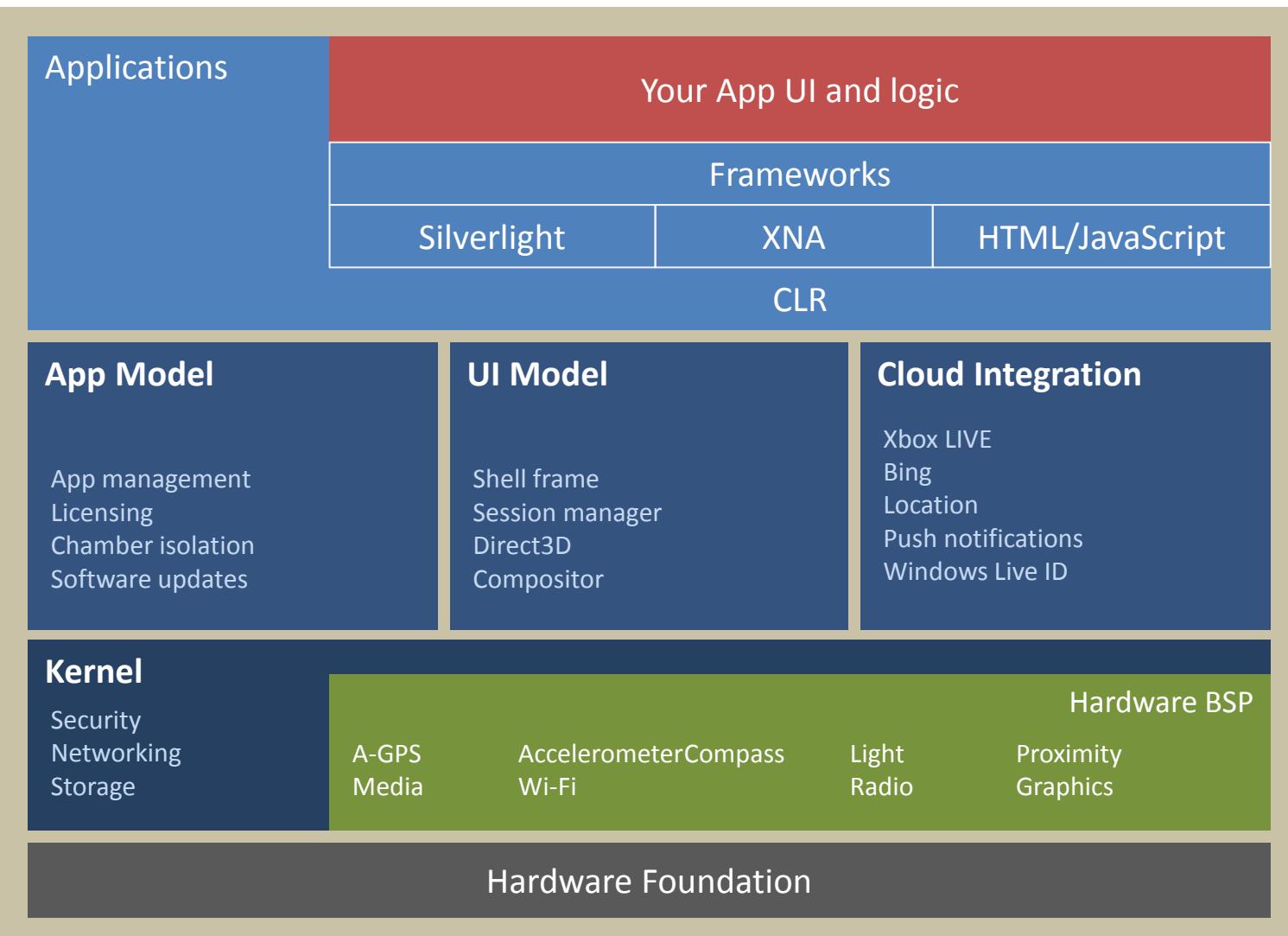
La información está organizada en áreas denominadas HUBS las cuales siguen un flujo de las áreas de intereses del usuario.

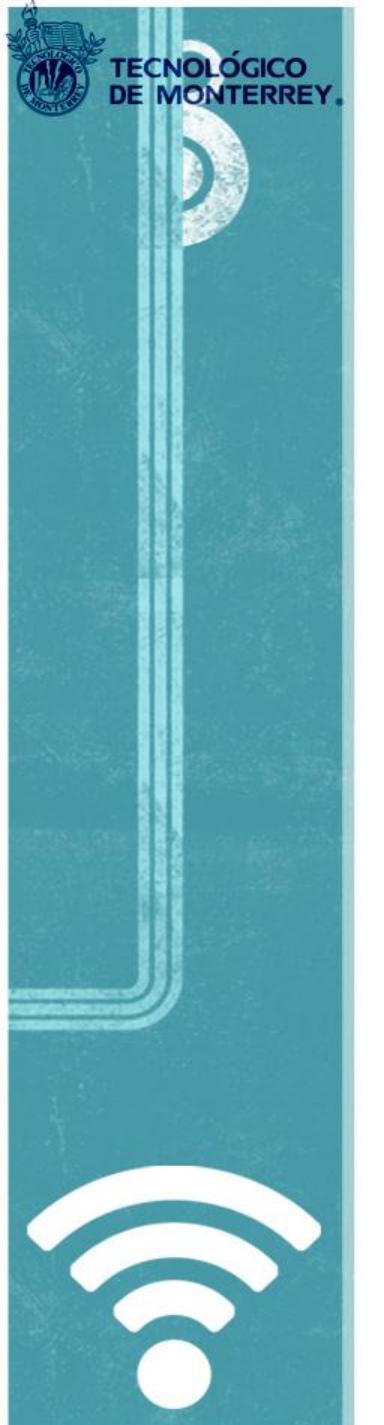


Los hubs están organizados en estos panoramas, colocando grupos de información en columnas de una sola pantalla paisaje (Se desplaza mediante tacto) unidos mediante Metro.



# Arquitectura del S.O.

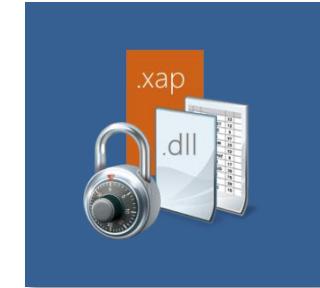
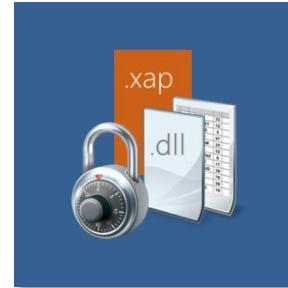




# Conceptos del modelo UI

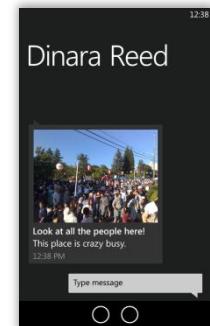
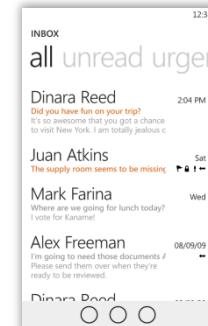
## App

UI y funcionalidades  
lógicas expuestas  
mediante Páginas



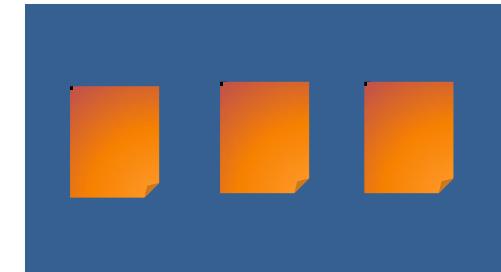
## Página

Una simple pantalla de  
elementos interactivos.



## Sesión

Un flujo de trabajo ordenado  
de interacciones de usuario

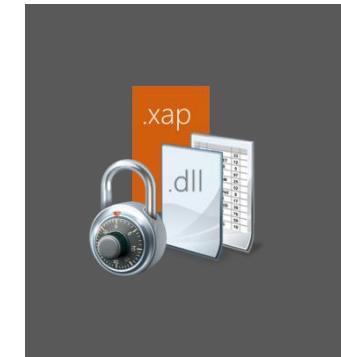




# Modelo UI : Conceptos ya conocidos...

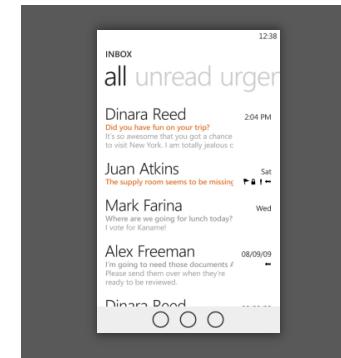
## Phone application

Provee UI representados como paginas XAML conectandos dentro de un flujo de cruce de Apps por URI's.



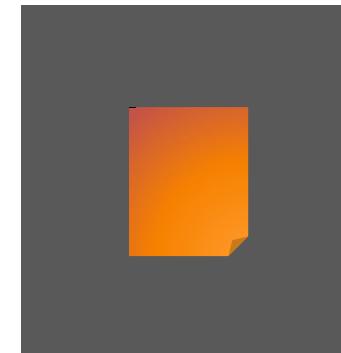
## Shell frame

Solicita pagina, Renderiza UI y maneja la navegación entre Apps.



## Sessions and back stack

Agrupa secuencias de paginas correspondientes a las actividades del usuario en las Apps.



## Web application

Provee medias representados como recursos HTTP vinculadas por URL's.

## Web browser

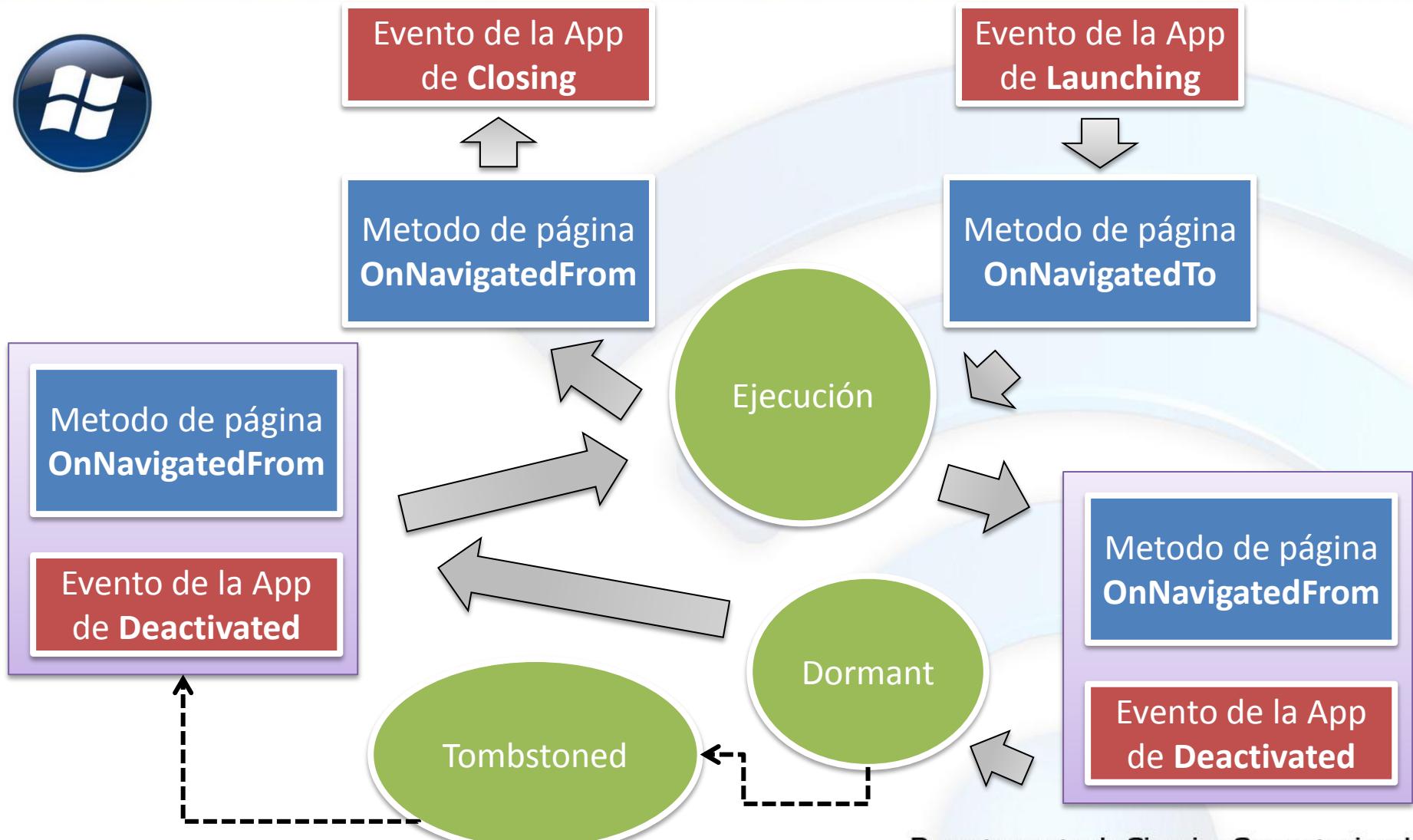
Solicita recursos HTTP, los renderiza y maneja la navegación entre sitios.

## History and tabs

Agrupa secuencias de recursos HTTP correspondientes a las actividades del usuario en los sitios



# Lyfecycle





Vs.



Windows Phone Eventos de Apps	Window Phone Metodos de Páginas	Android
Application_Launching	InitializeComponent()	onCreate()
		onStat(), onResume()
Application_Deactivated	NavigatedFrom	onStop(), onPause()
Application_Closing		onDestroy()

## Dormant

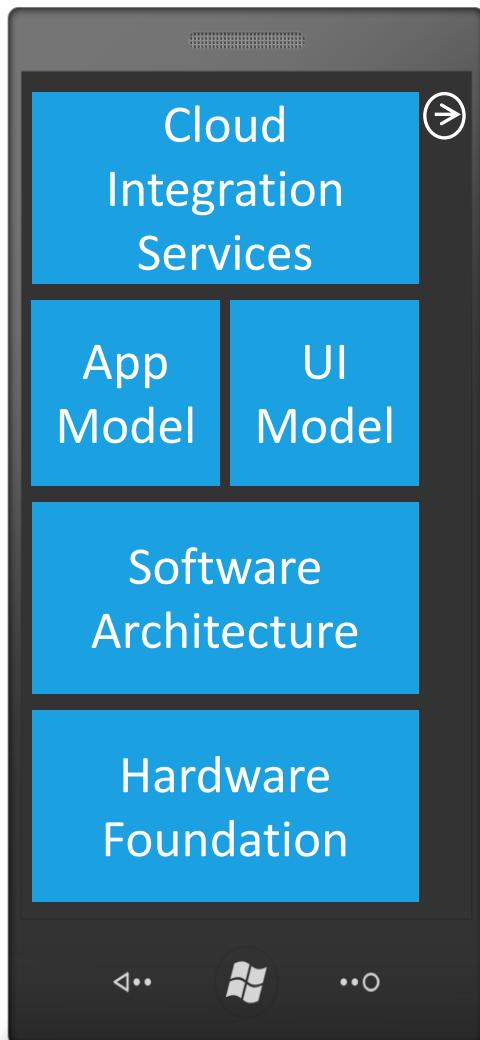
Aquella App que este en ejecución y que por eventos externos (SMS, llamada, Home, etc) es puesta a “dormir”

## Tombstoned

Cuando una App deja de ser el “foco” en el telefono es colocada en este punto hasta que el usuario regrese a ella...



# Apps y la seguridad



Toda las Apps (\*.xap) son descargadas certificadas y seguras



Windows Phone maneja todo los elementos de la instalación de un .xap basado en el manifiesto.

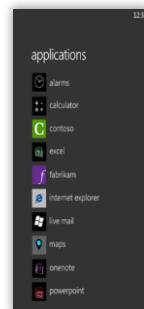
➔ Apps individuales no son capaces de hacer cambios arbitrarios al teléfono durante la instalación.

Los usuarios controlan la instalación, actualización y desinstalación de una App mientras que Windows Phone Marketplace controla la revocación de la misma. ocation

➔ Apps individuales no controlan su propio ciclo de vida en el teléfono.

# App Isolation and Execution

Applications  
and licenses



Application  
install  
folders

Running  
applications



**Windows** Phone  
Marketplace

Windows Phone solo ejecuta Apps que tienen una licencia valida del

Durante instalación y ejecución cada App esta en su propia “Sandbox” y con cuentas de usuarios separados.

Política de localización de recursos mantienen las apps de fondo con capacidad de responder.

Política de manejo de recursos garantiza que el usuario siempre pueda iniciar a ejecutar una nueva App.



# Permisos / Capacidades

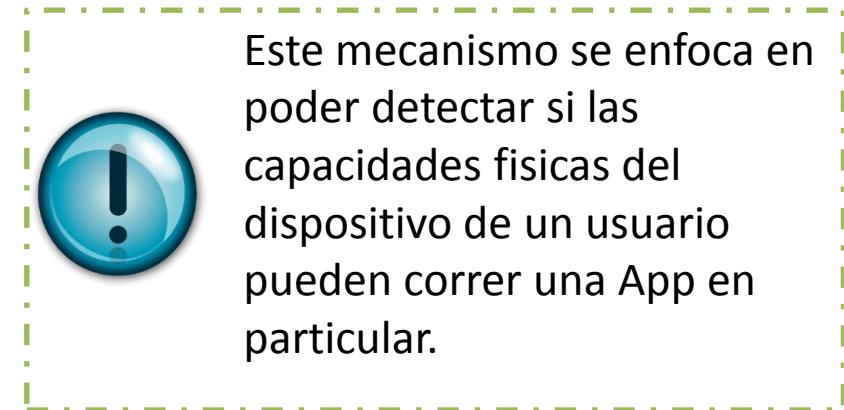


No existen permisos en **Windows Phone**, si no Capacidades de la aplicación, dichas capacidades son definidas en el WMAppManifest.xml

El desarrollador es quien manipula el manifiesto y este es validado en el



**Auditado de forma estricta**



Este mecanismo se enfoca en poder detectar si las capacidades físicas del dispositivo de un usuario pueden correr una App en particular.

Windows Phone provides a capabilities-driven security model where a user must opt-in to certain functionality within the application

Windows Phone operating system grants security permissions to the application according to the capabilities listed in that application manifest file

# Listado de Capacidades (I)

Valor	Descripción
ID_CAP_APPOINTMENTS	Applications that access appointment data.
ID_CAP_CAMERA	Apps que utilicen capacidades de la camara. Esta capacidad es controlada <b>solamente</b> por los Operadores Móviles y los manufacturadores originales del dispositivo. Desarrolladores de Apps deben usar ID_CAP_ISV_CAMERA en su lugar.
ID_CAP_CONTACTS	Apps que accedan a los datos de contactos.
ID_CAP_GAMERSERVICES	Las aplicaciones que pueden interactuar con las API de Xbox LIVE. Esto debe ser divulgada debido a cuestiones privacidad, ya que los datos son compartidos con la Xbox.
ID_CAP_IDENTITY_DEVICE	Las aplicaciones que utilizan información específica del dispositivo, como un ID de dispositivo único, el nombre del fabricante, o nombre del modelo.



# Listado de Capacidades (II)

Valor	Descripción
ID_CAP_ISV_CAMERA	Apps que utilizan la cámara principal o la frontal.
ID_CAP_LOCATION	Apps con acceso a servicios de localización
ID_CAP_MEDIALIB	Apps que pueden acceder a la librería de medios.
ID_CAP_MICROPHONE	Apps que utilicen el microfono. No es requerido dar una indicación visual de su uso.
ID_CAP_NETWORKING	Apps con acceso a servicios de redes. Esto debe ser indicado debido a que tales servicios pueden incurrir en cargos económicos.
ID_CAP_PHONEDIALER	Apps que pueden colocar llamadas telefónicas. No es requerido dar una indicación visual cuando se realiza.



# Listado de Capacidades (III)

Valor	Descripción
ID_CAP_PUSH_NOTIFICATION	Las aplicaciones que pueden recibir notificaciones/eventos <i>push</i> desde un servicio de Internet. Esto debe ser divulgada por el uso podría incurrir en cargos por <i>roaming</i> .
ID_CAP_WEBBROWSERCOMPONENT	Las aplicaciones que utilizan el componente de navegador web. Hay riesgos de seguridad debido al <i>scripting</i> .
ID_HW_FRONTCAMERA	Las aplicaciones que tienen características que requieren de la cámara frontal. Si el dispositivo no posee una el usuario recibe un mensaje de advertencia.



# Listado de Capacidades (IV)

Valor	Descripción
ID_CAP_PHONEDIALER	Applications that can place phone calls. This may happen without a visual indication for the end user.
ID_CAP_SENSORS	Applications that use the Windows Phone sensors.
ID_CAP_IDENTITY_USER	Las aplicaciones que utilizan el Live ID anónimos para identificar al usuario de forma anónima.

La protección contra Malware viene “embebida” en los mecanismos para obtener los manifiestos y poder colocar la App en el **mercado** oficial del **Windows Phone**, por lo mismo el usuario final “recibe” información clara y “fidedigna” en el momento de instalar dicha App.

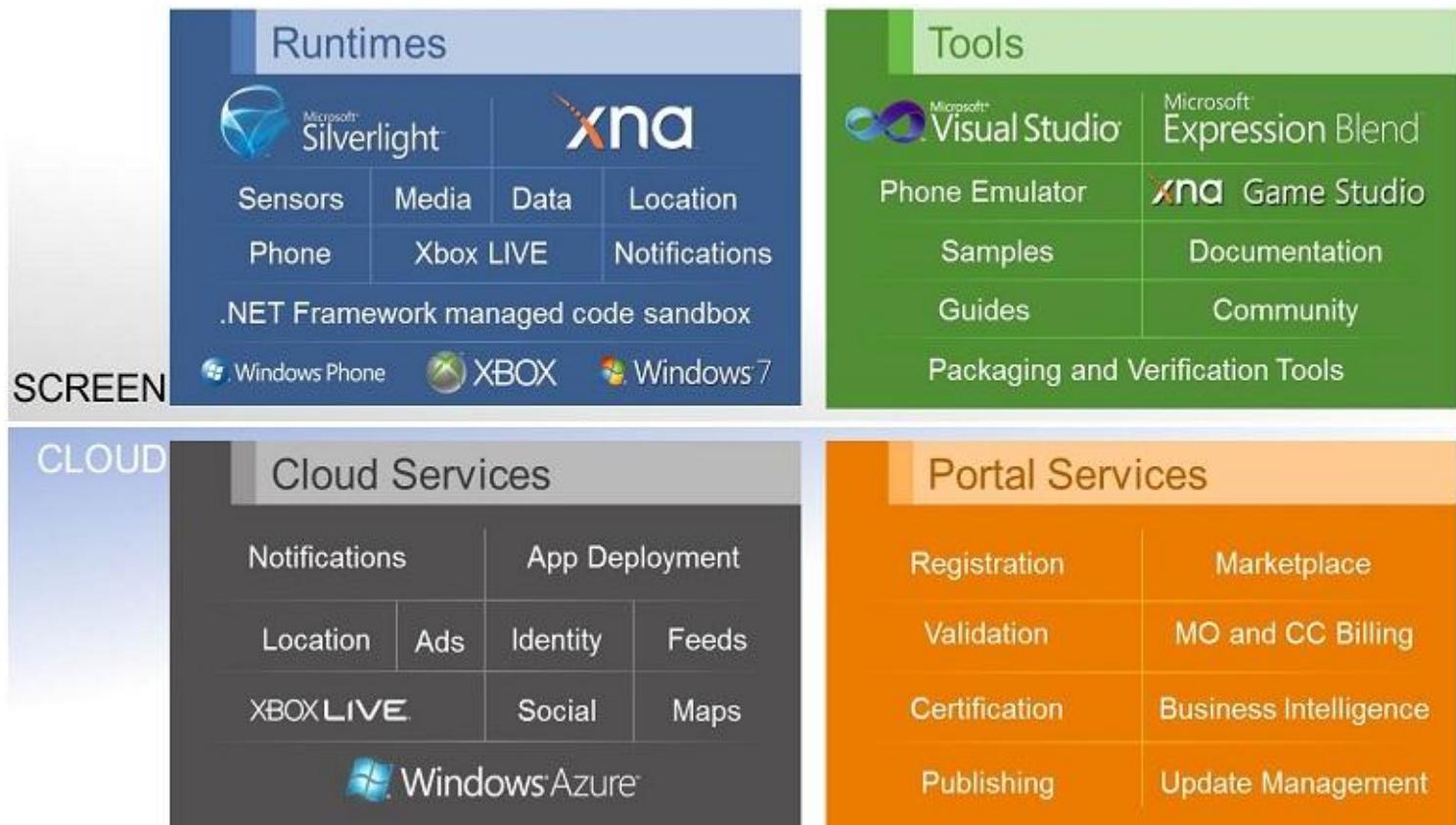


**Windows  
Phone  
Marketplace**



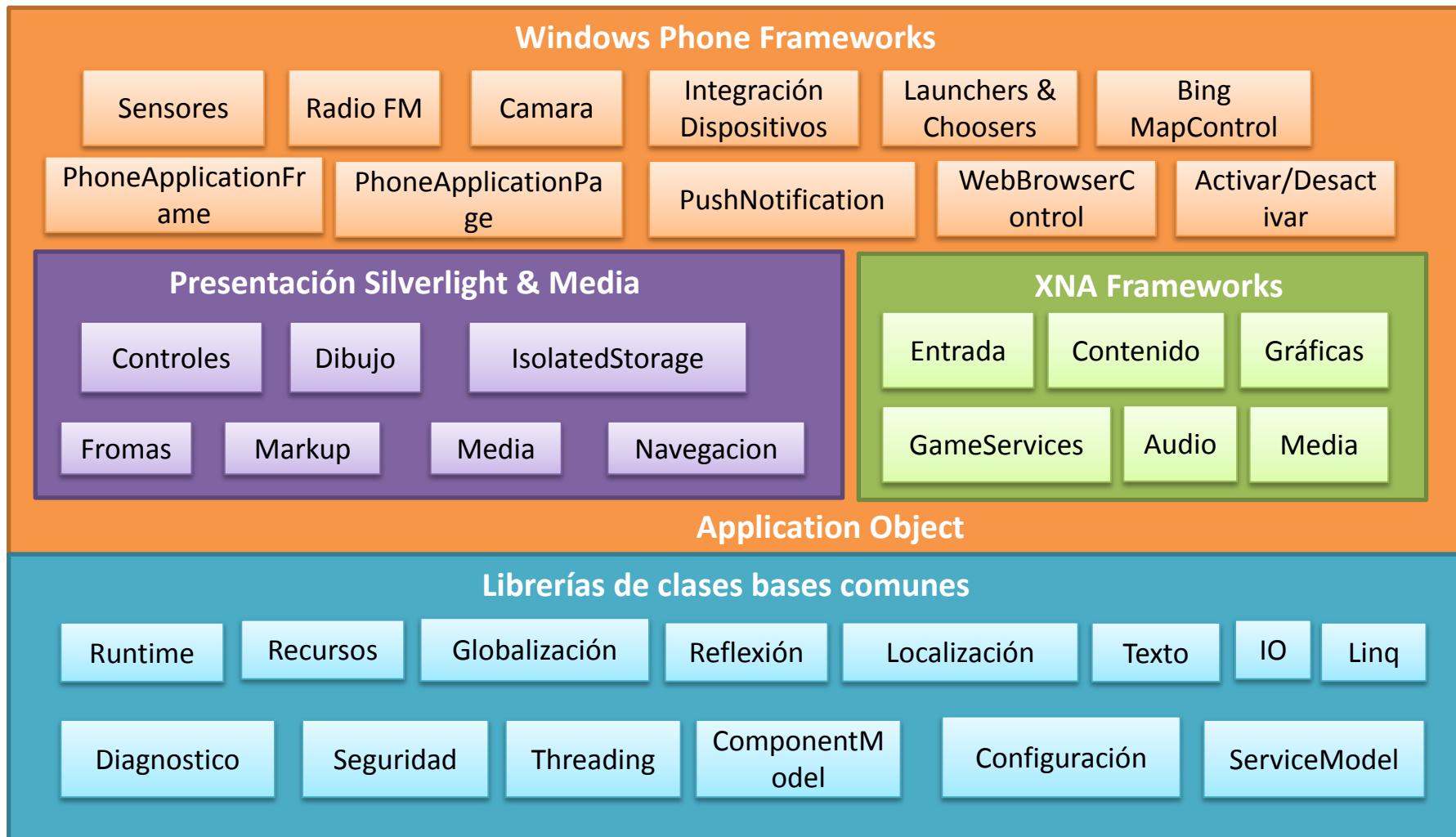


# Arquitectura de la Plataforma de Aplicaciones





# Runtimes (I)





# Runtimes (II)

Silverlight & XNA Framework junto a las opciones específicas del Windows Phone combina un ambiente maduro para la creación de Apps seguras y enriquecidas en gráficos.

## Silverlight

- Rich Internet App-style UI
- Las UI son expuestas en páginas
- Visual Studio o Expression
- El diseño (Style) de las páginas controladas por el S.O.
- Bien sirven para crear las interfaces basadas en XAML.
- Permite el uso de los controles de Windows Phone.
- Permite embeder videos y/o HTML Browser Control.

## XNA

- Software, Servicios y recursos enfocados para diseño de videojuego en plataformas Microsoft.
- Involucra un grupo de API's para tal elemento. (2d, 3d, rotaciones, luz, etc)
- Ideal para juegos de alto desempeño.

## Combo

- XAML+Videojuego de alto desempeño.
- Videojuego+ UI de Internet.
- Rendering de texto (internacional) en vez de Sprites.
- Etc...

## Datos

IsolatedStorage permite crear sandbox de directorios (ninguna App puede alcanzar los archivos claves del S.O.)



# Herramientas (I)



## Windows Phone SDK

Visual Studio 2010 (ó la versión expres para Windows Phone) es el IDE para crear Apps para Windows Phone. (XNA o Silverlight)

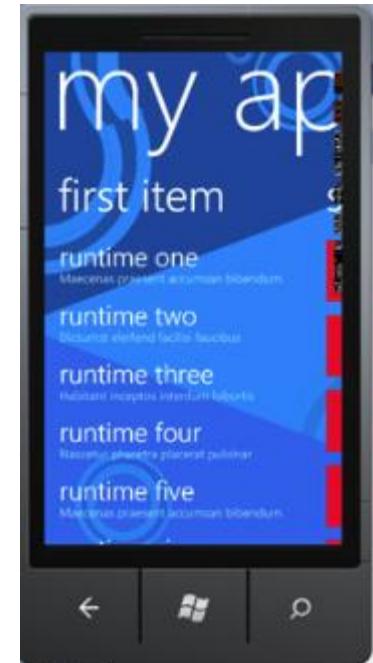


Permite la creación de interfaces basadas en XAML (Silverlight) cuyo comportamiento puede ser implementado mediante Visual Studio.



# Herramientas (II)

*Visual Studio* y *Expressión Blend* incluye emuladores del Windows Phone para las pruebas de Apps. Tiene soporte completo (incluso emula hasta cierto punto el Dispositivo objetivo) para el desarrollo, debugging y ejecución de las Apps.



Ambiente de diseño para desarrollar juegos para Windows, Xbox360, Zune y Phone. Permite el soporte del framework XNA en Visual Studio además de herramientas para gráficos y audio.

 | Game Studio



# Servicios en la nube y Servicios de Portales

Microsoft permite el uso de Azure o servicios web de terceros para importar datos al teléfono. Las API disponibles sirven para:

- Usar de forma eficiente (Uso de bateria) las notificaciones.
- Facilitar el uso de herramientas tales como localización fisica (GPS, Assited-GPS) servicios de mapa identidad, etc (Facilitar el acceso de Apps a estos recursos).
- El uso de la plataforma Windows Azure sea como un conjunto o solo ciertas partes.
- Manejo de anuncios publicitarios con “*Microsoft Advertisers on the Microsoft Advertising Platform for Windows Phone*”

**Windows Phone MarketPlace** provee un mecanismo centralizado para certificar y colocar Apps con objetivo que el usuario final solo requiera de un único punto para comprar o actualizar las Apps.



Windows Phone  
Marketplace



# S.O. BlackBerry

- ❖ Propietario de 
- ❖ BlackBerry apareció originalmente en 1999 para sistemas empresariales en el 2004 entra al comercial.
- ❖ En el 2010 declino y en el 2011 sufre un colapso.
- ❖ El kernel esta basado en Java (A partir de los modelos 5000 y 6000).
- ❖ La arquitectura de hardware parece no cambiar mas que para su tablet.
- ❖ Tiene un gran soporte multiplataforma para sus Apps.



# Arquitectura (Hardware)

- El hardware es una arquitectura ARM (del tipo RISC) con un procesador Intel XSCale.
- El firmware es RedBoot
- El S.O. de acuerdo a RIM ofrece Multitasking.
- El S.O. permite la interacción con el hardware mediante Java. (OS 10 usara QNX acabando con Java)
- **Todo lo demás es un secreto de RIM**

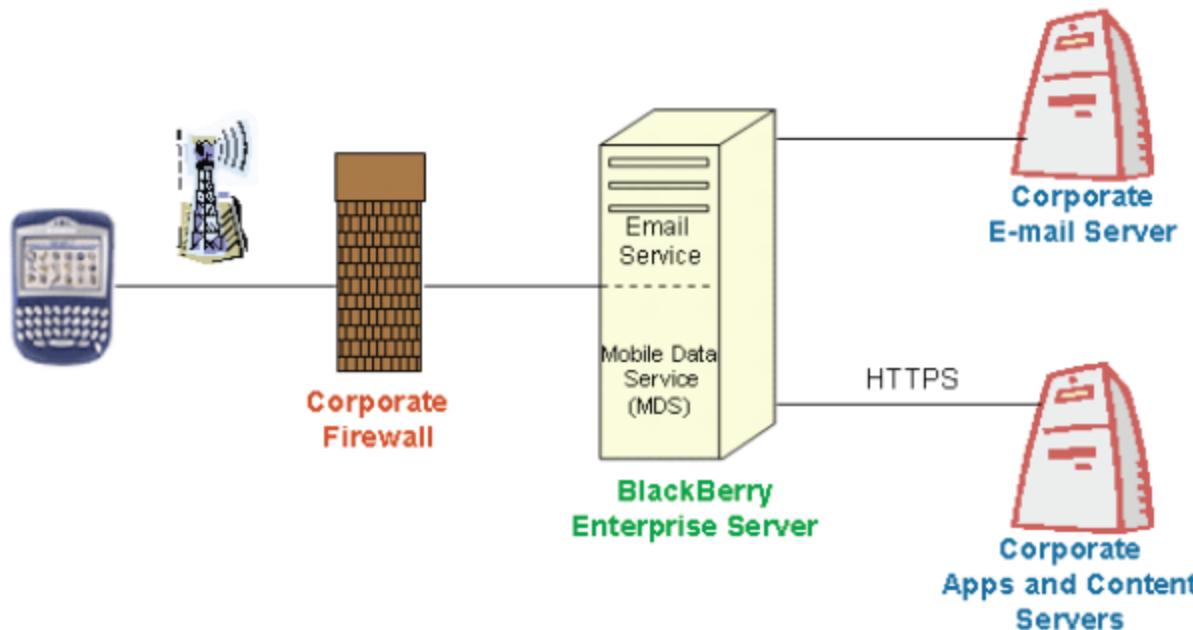




# Arquitectura de Red en



RIM posee su propia red privada antes de dar acceso a Internet (El envío de mensajes tiene capacidad de cifrado).





# Gran variedad de soporte para Apps



C/C++  
Native SDK



HTML5  
WebWorks



Java  
BlackBerry Java



Java  
Android Runtime



Action Script  
Adobe AIR



Themes  
Theme Studio

- “Plataforma abierta” con variedad de lenguajes de desarrollo y runtimes de acuerdo a las habilidades del desarrollador.
- Gran soporte para cada uno.
- El mercado de Apps de Blackberry es el mas rentable



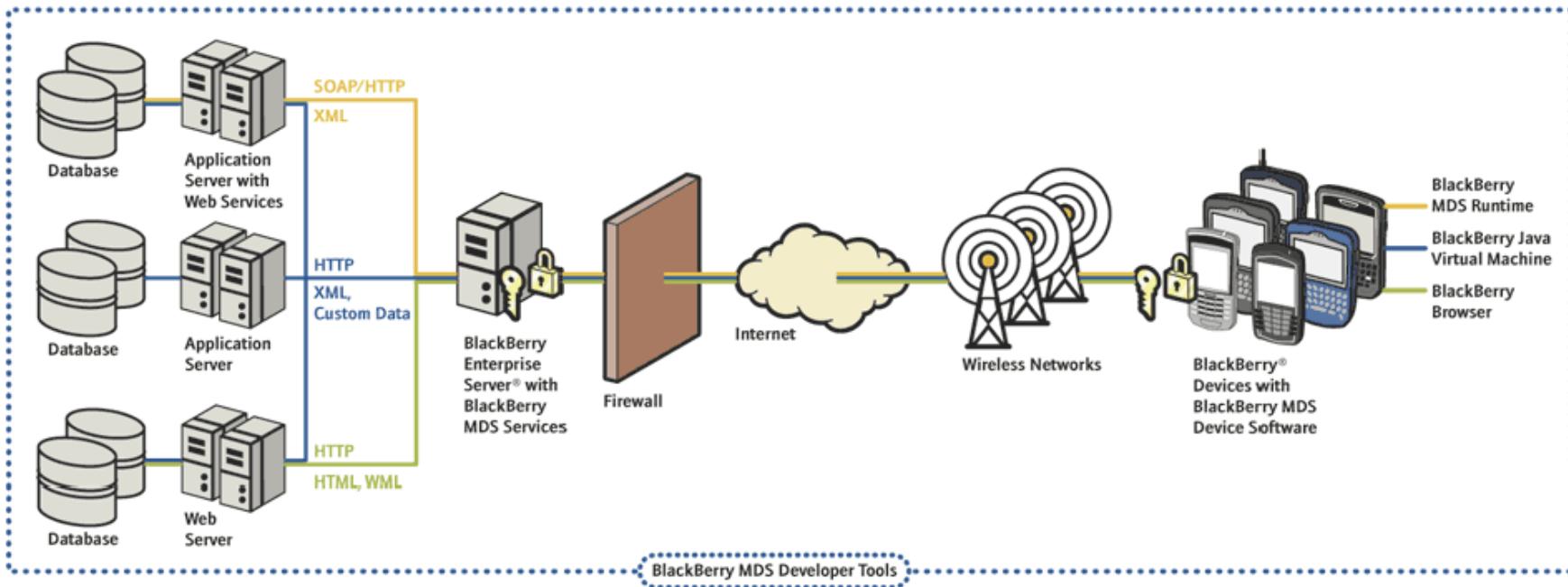
Posición 15/50 de los grandes inventos del os últimos 50 años



# Apps & Ambiente de ejecución

- Existen dos modelos importantes
  - JVM en el que corren principalmente CLDC, MIDLets, RIMlets
  - Mobile Data Service (MDS) runtime
- Aunque se usa JavaME los archivos .jar y .jad son convertidos a archivos propietarios .cod.
- Existe ademas archivos .alx (Cargar Apps mediante BB Desktop Manager) y estan basados en XML
- Los .cod pueden estar firmados

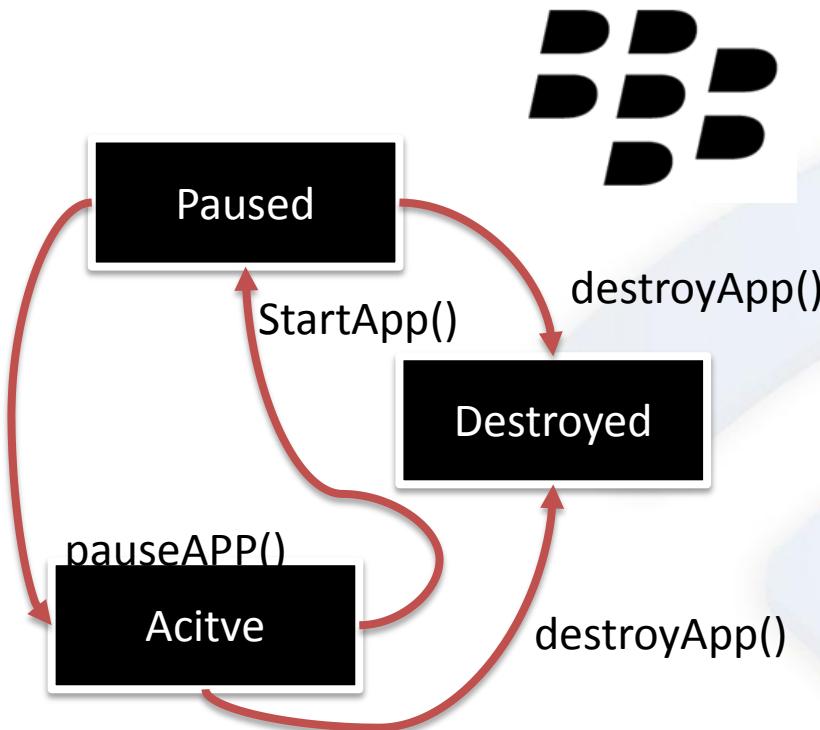
# MDS



- MDS se enfoca principalmente en servicios WEB y empresariales
- Por lo mismo se consideran “Basados en Buscadores”
- Definen esquemas WSDL o SQL-DB
- Se utiliza encripción y compresión de datos.



# Apps basadas en MIDP y CLDC

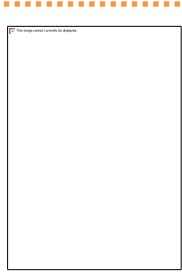


- **MIDP**

- Java ME o en ocasiones referida como MIDP se refiere principalmente a Java ME CLDC. Las apps desarrolladas en esta plataforma se le denominan MIDlets, un nombre analogo a Applets o Servlets.
- Sus ciclos de vida estan por lo mismo definidos en la parte de JavaME (o CLDC o MIDP).

- **BlackBerry CLDC**

- BB pueden ejecutar por completo MIDP pero anexan sus propios frameworks para trabajar con el hardware del dispositivo. RIM se refiere a las Apps hechas con CLDC como “*CLDC Application*”



El nuevo S.O. 10 (BB10) dejara de utilizar Java ME como plataforma. Por lo tanto muchas Apps tendrán que ser re-escritas (incluyendo oficiales )



# webOS™

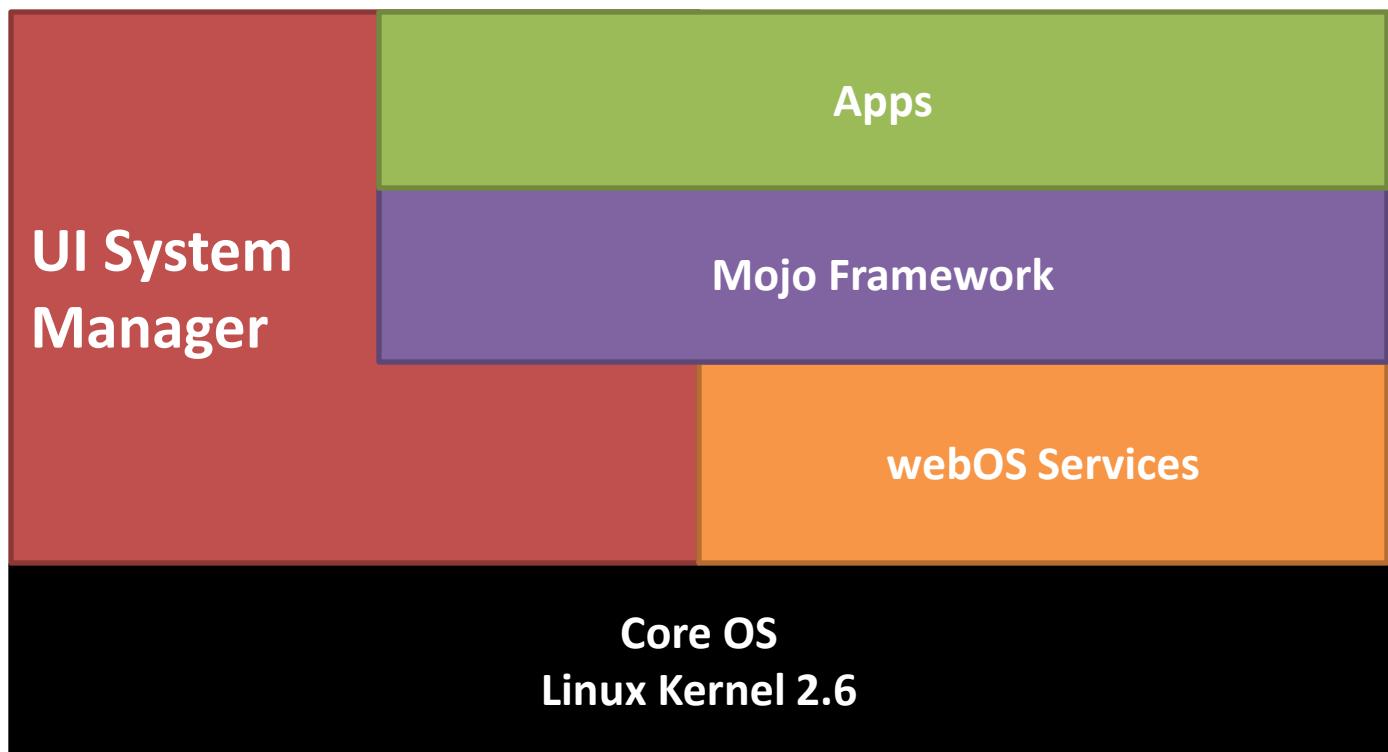
- Originalmente desarrollado por PALM para los Palm OS en el 2009.
- Aquerido por HP al comprar Palm
- Liberado como Código Abierto en el 2011 . 
- HP le apuesta a webOS como la plataforma para todo sus futuros productos (Excepto PC por el momento).
- webOS o WebOS es un sistema Linux embebido.
- Embebe un S.O., De ventanas con la simplicidad de un navegador web permitiendo que las Apps estén hechas en tecnologías y lenguajes estándares de la Web (Html5, CSS, javascript) esto da como resultado un S.O. Extremadamente ágil para el multitasking.
- Si tienes experiencia con AJAX... dominas WebOS

**HTML**





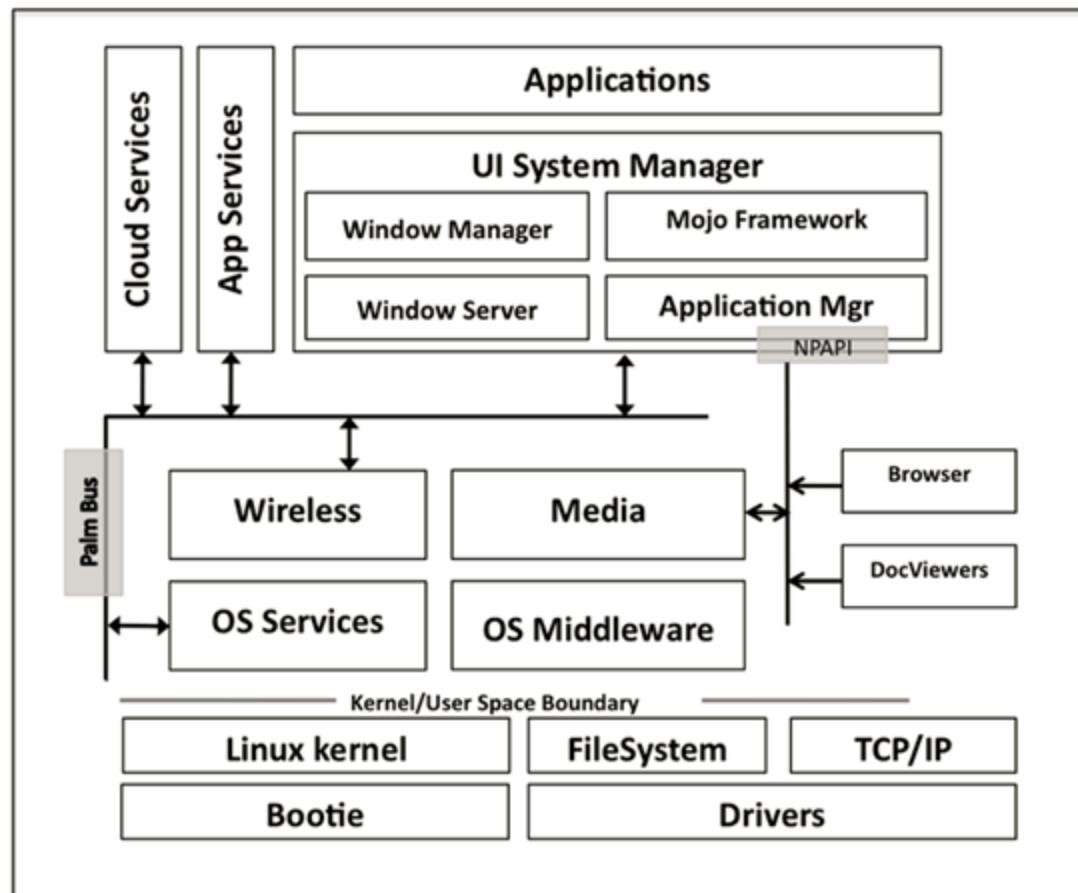
# Arquitectura del S.O. ( I )



El Core OS soporta ext3 para las particiones (privadas) internas y Fat32 para las particiones de los medios externos (vía USB).



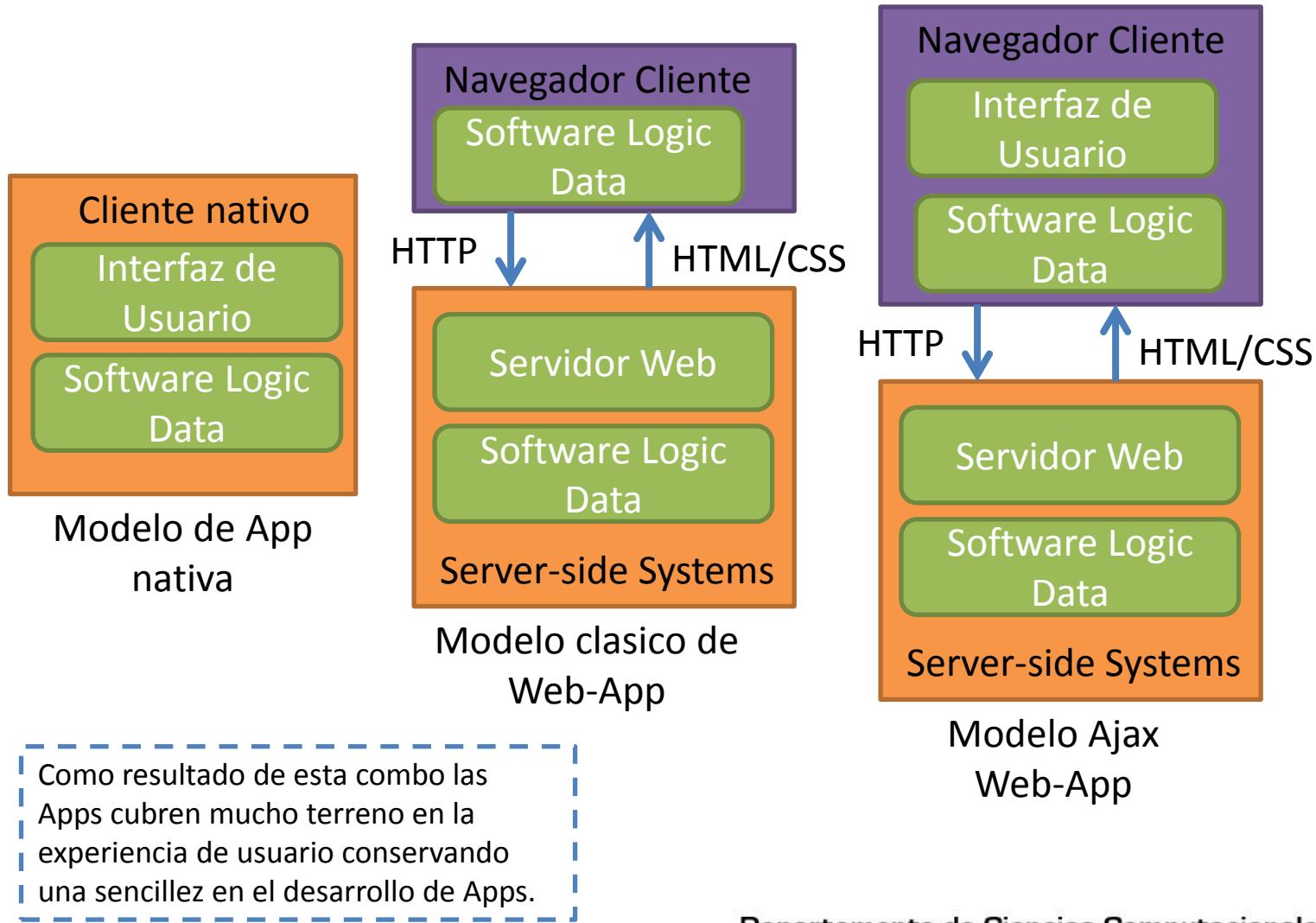
# Arquitectura del S.O. ( II )



El UI System Manager (UI SysMgr) es elemento clave para todo lo que es visible al usuario.



# Arquitectura de las Apps





# Framework

El framework de Apps permite:

- Embeber Widgets UI con capacidades avanzadas.
- Ejecutar interfaces de usuarios avanzados.
- Manejador de eventos
- Servicios de notificación
- Modelo multi-tasking (Permitiendo Apps corriendo de fondo y manejo de datos por separado)
- Se puede manejar BD, datos de contactos y calendario mediante funciones de almacenamiento de HTML5.
- La arquitectura del OS es un GNU/Linux con un Manejador de Sistema UI personalizado y construido en tecnología de navegador estándar.
- Apps Core por defecto: Contactos, Calendario, Tareas, Memos, Teléfono, navegador, e-mail, messaging.



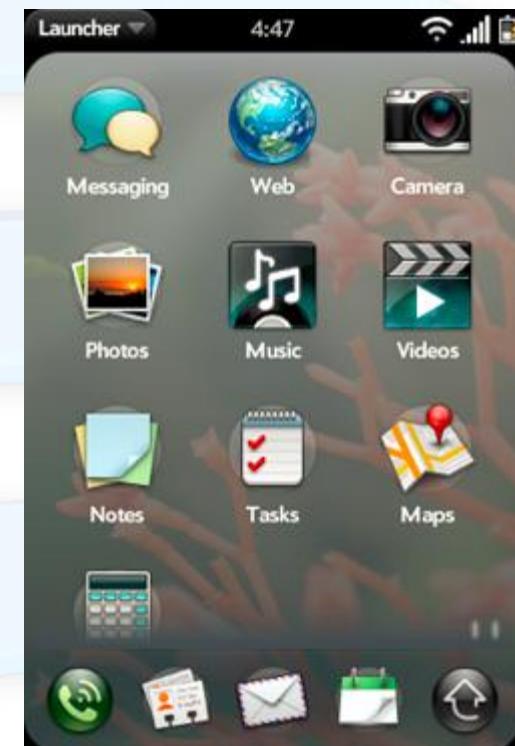
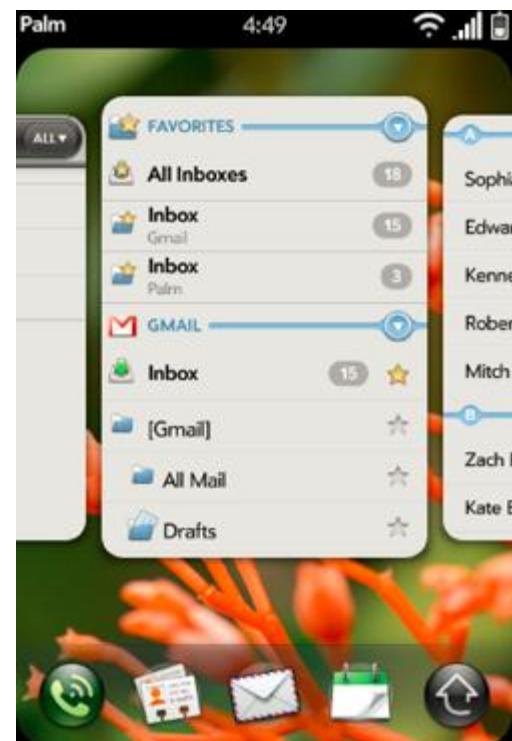
open source



ENYO

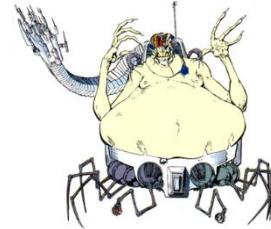
# Interfaz de usuario & Navegación

- UI hecho para distintos dispositivos móviles principalmente aquellos con pantalla táctil.
- Pantalla principal posee una barra de estatus, botones de ejecución y demás elementos parecidos a otros smartphones.
- Maneja “Cardview” para fácil cambio entre Apps y sus interfaces (Por ejemplo cada correo tendrá su propia tarjeta además de la tarjeta de la App).





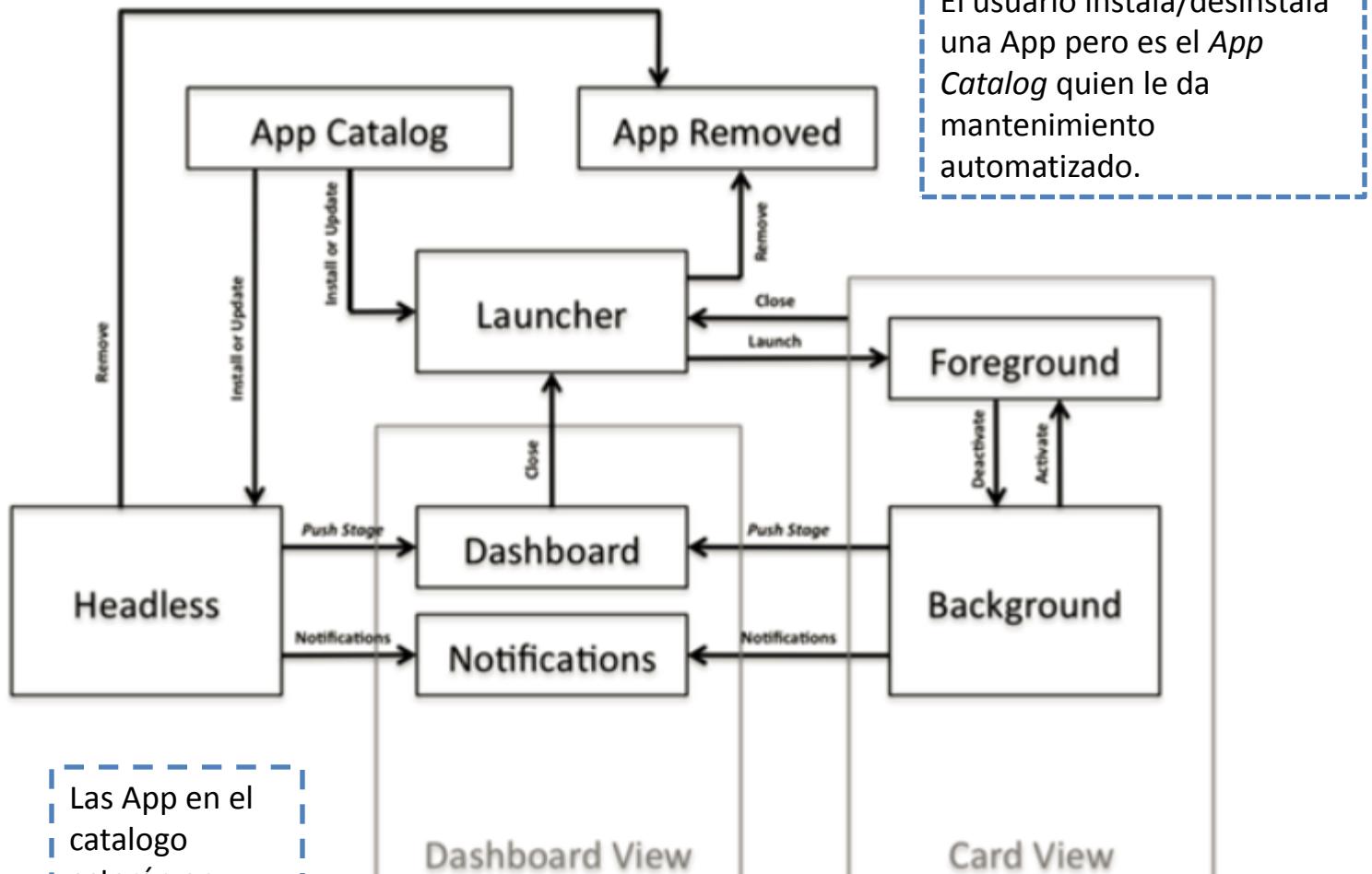
# Framework : MOJO



- Los Ciclos de vida de las Apps son distintas a los típicos de Web-Apps.
- Las Apps son ejecutadas dentro del Sistema Manejador del UI.
- Dichas Apps son entregadas como frameworks de javascript denominado MOJO los cuales soportan una gran variedad de funciones pero pueden incrementar sus opciones utilizando HTML5 para elementos tales como video, audio y DB.
- Mojo utiliza una arquitectura basada en MVC (Model View Controller) para incrementar sencillez.



# Ciclos de vida de una App





# Servicios

- Los servicios son el elemento clave de para completar la plataforma webOS.
- Permiten la interacción con los componentes físicos del dispositivo.
- Se tratan de servidores “*on-device*” para cualquier recurso, dato o configuración que es expuesto a través del framework para el uso interno de una App.
- Pueden ser accedidos por una el OS (Apps nativas), una App o incluso un servidor en la nube.
- Son invocados atreves de un solo controlador cuya función es serviceRequest la cual pasa un objeto JSON.



# ¿Seguridad?



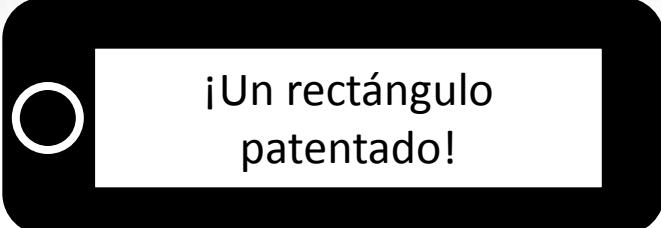
- De forma similar a Windows Phone, solo las Apps que pasan por el Mercado oficial de WebOS reciben una auditoria de seguridad
- El único enfoque de seguridad que audita HP en una App es el - Respeto de la App por las preferencias de nivel de sistema PRE- Existentes (no las altera) y colocadas por el usuario.
- Sin embargo parece haber algún tipo de seguridad bajo ocultamiento en el sitio (aunque si lo hay no es divulgado).
- Finalmente, las Apps son en servidores Web, por lo tanto se debe de asumir los riesgos de seguridad en tales plataformas.
- Una limitante es que un App no puede acceder a datos claves del usuario mediante terceros (Debe solicitar permiso en vuelo)



# iOS

- Originalmente llamado: “*Iphone Operating System*”, posteriormente compro “iOS” de Cisco.
- Propietario de Apple Inc.
- El elemento tan cerrado de Apple Inc es una apuesta intencional para el desarrollo “optimo” de las Apps.
- Al igual que Microsoft, iOS apunta a integrar todo los servicios que ofrece Apple Inc dentro de su plataforma (Apple Tv, iStorage) aunque el uso de iTunes es obligatorio..
- Apple se reserva mucha información de la arquitectura del S.O.

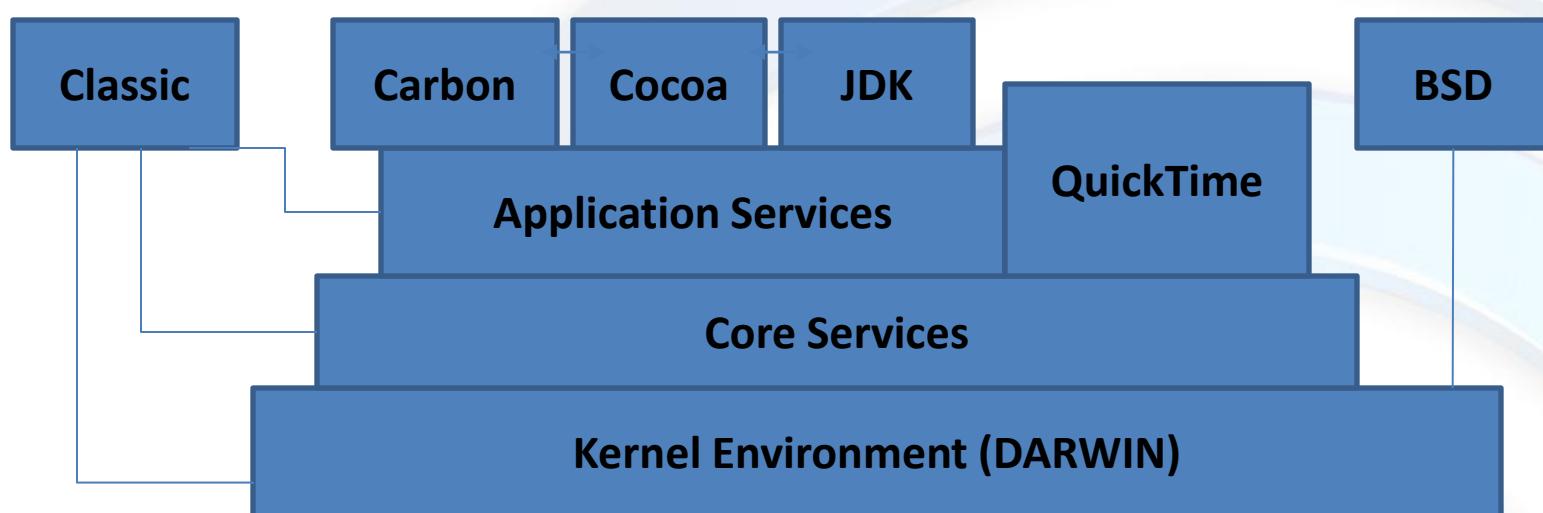
# Introducción...



- iPhone, iPods e iPad poseen dicho S.O.
- El S.O. del iOS esta basado fuertemente en el S.O. Mac OS X.
- Las apps no pueden interactuar directamente con el hardware si no que lo hacen mediante el iOS.
- Solo ciertos dispositivos con iOS4 o superior pueden ejecutar Multitasking (los anteriores tenían que efectivamente matar temporalmente una aplicación fuera de foco).
- Toda las Apps poseen sus “Preferencias” que son configurables desde “Settings”, aunque la mayoría no hará uso de ellas.
- Las Apps se dividen en **iOS App** y **Web Apps** siendo estas ultimas servidores Web (en la red) cuyas paginas simulan ser Apps (la GUI).
- **Safari** en **iOS** juega un papel clave y por lo mismo difiere a la versión **para Mac OS X** (No Adobe y adaptación del punto anterior)

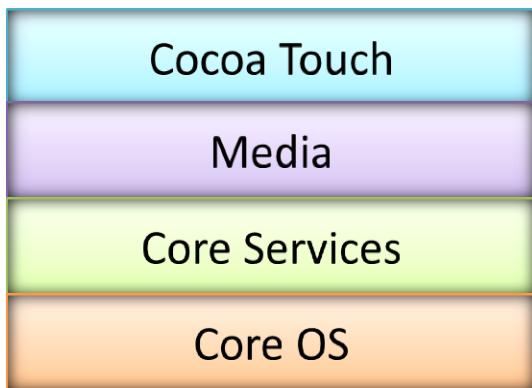
# Mac OS X (Arquitectura)

Basado en Unix





# S.O. Arquitectura



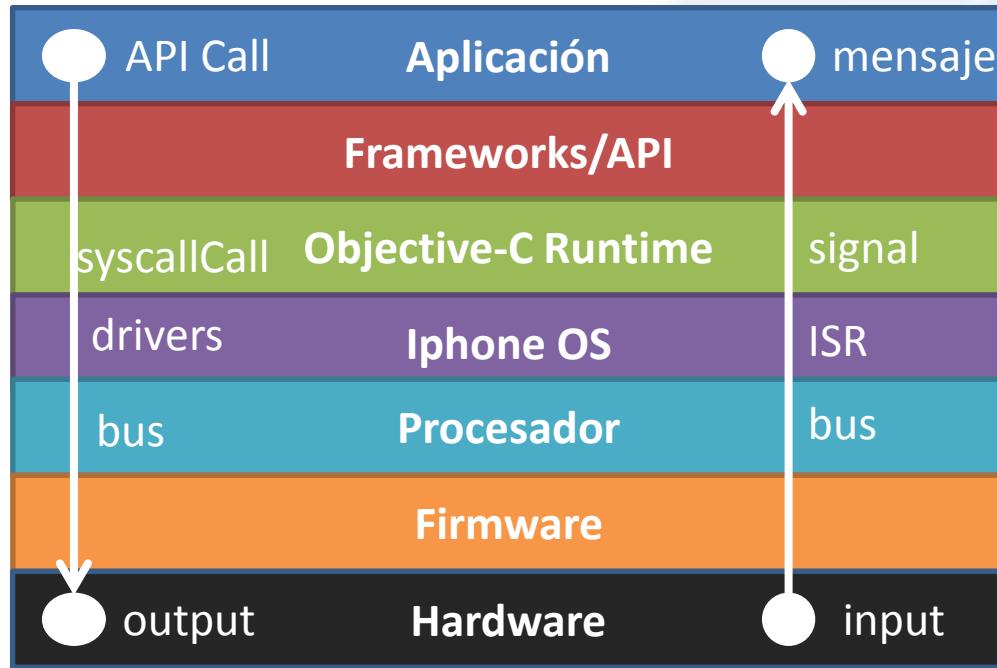
En síntesis toda la interfaz humano-dispositivo se halla en **Cocoa**, además de servicios de impresión, compartición de archivos y encriptación.

**La capa de Medios** contiene todos los recursos que el desarrollador ocupara para su App, manejo de gráficos incluyendo acceso a OpenGL ES y control de audio y de Video, cabe aclarar que estos dos ultimos usa fuertemente HTML5 (NO Adobe).

Los servicios de “Core” manejan el control de la iCloud (servicio de nube de Apple) como es la parte de cifrado y credenciales, control de objetos Objective-C, C, SQL, XML entre otros.

La ultima capa al igual que Android el manejo de hardware proviene de esta capa y la inferior.

# Arquitectura del iPhone



La App es adquerida mediante la AppSotre compilada en código nativo y vinculada con Objective-C runtime & Librería C. Se mantiene en el sandbox.

Invocaciones de Cocoa Touch, Upper-level Open GLL. Mayor parte de los frameworks escritos en Objective-C

Librerias Objective-C y librerias C (que crean el ambiente para el Objective-C runtime)

Kernel, drivers & servicios del iOS

Arquitectura ARM ( ensamblador)

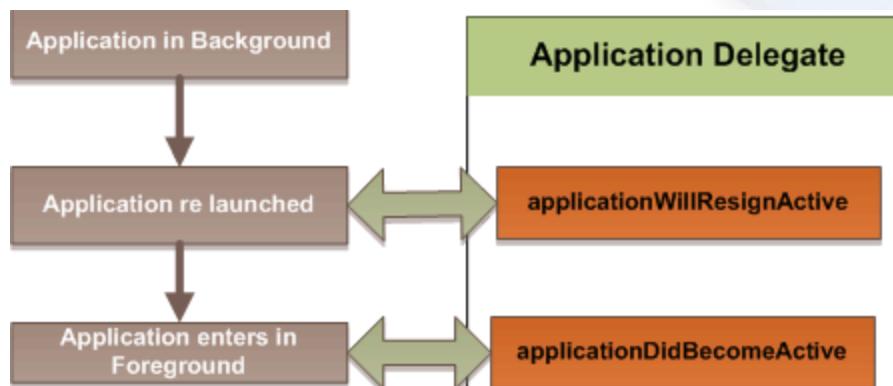


# Human Interface (Tap Away)

- Diseño basado en el usuario y sus necesidades operacionales (Elemento revolucionario en su momento).
- 100% Táctil (el dedo es el nuevo apuntador).
- “Human Interface” es el término de Apple para UI y se refiere al diseño para “dedos” haciéndolo lo mas fácil para tu audiencia objetivo. (En síntesis no es darle máximo control si no solo lo necesario).
- “Con un iPhone lo ultimo que harás es llamar”

# Lifecycles – Conceptos/Fases

- **Not running** – La app todavía no es lanza (inicializada, ejecutada, etc)
- **Inactive** – La App esta corriendo de fondo pero no esta recibiendo eventos.
- **Active** – La App esta corriendo de fondo y recibiendo eventos.
- **Background** –La mayoría de las Apps entran a esta fase brevemente cuando van encaminadas a ser suspendidas, sin embargo una App que solicita tiempo extra de ejecución puede permanecer en esta fase por un periodo de tiempo. Además una App inicializada para funcionar de fondo entrara a esta fase sin pasar por “inactive”
- **Suspend (iOS>3)** - La App se encuentra de fondo pero no se esta ejecutando código alguno. Técnicamente esta congelado y solo es removido cuando se tiene poca memoria disponible



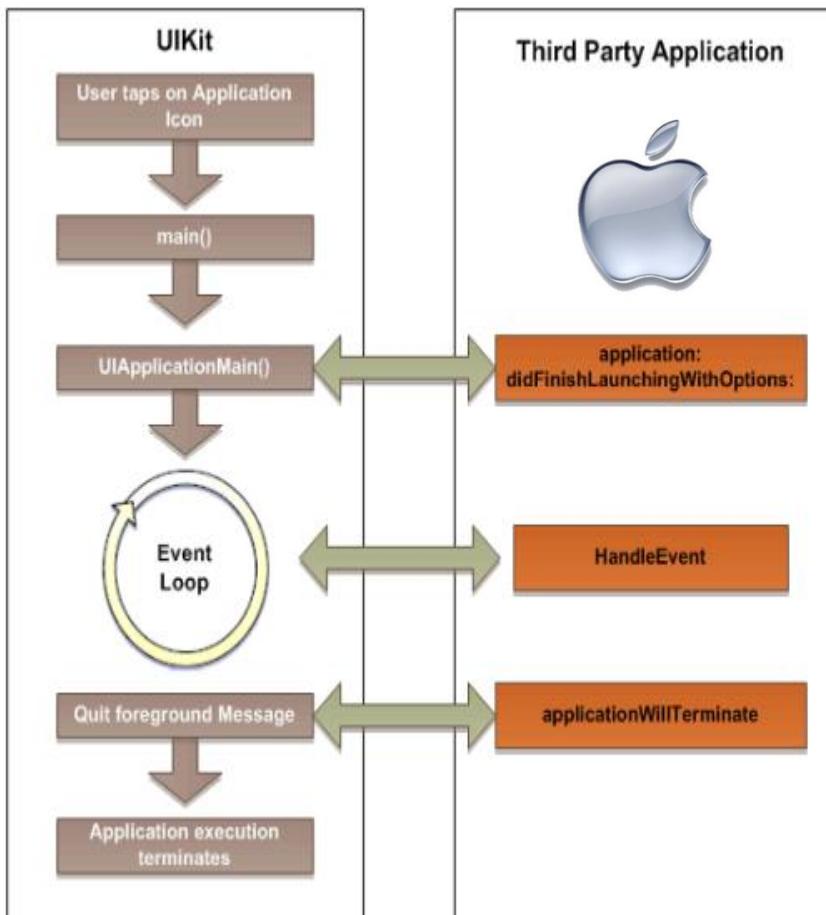
Transiciones de Background a activo

`applicationWillEnterForeground`  
`applicationDidBecomeActive`

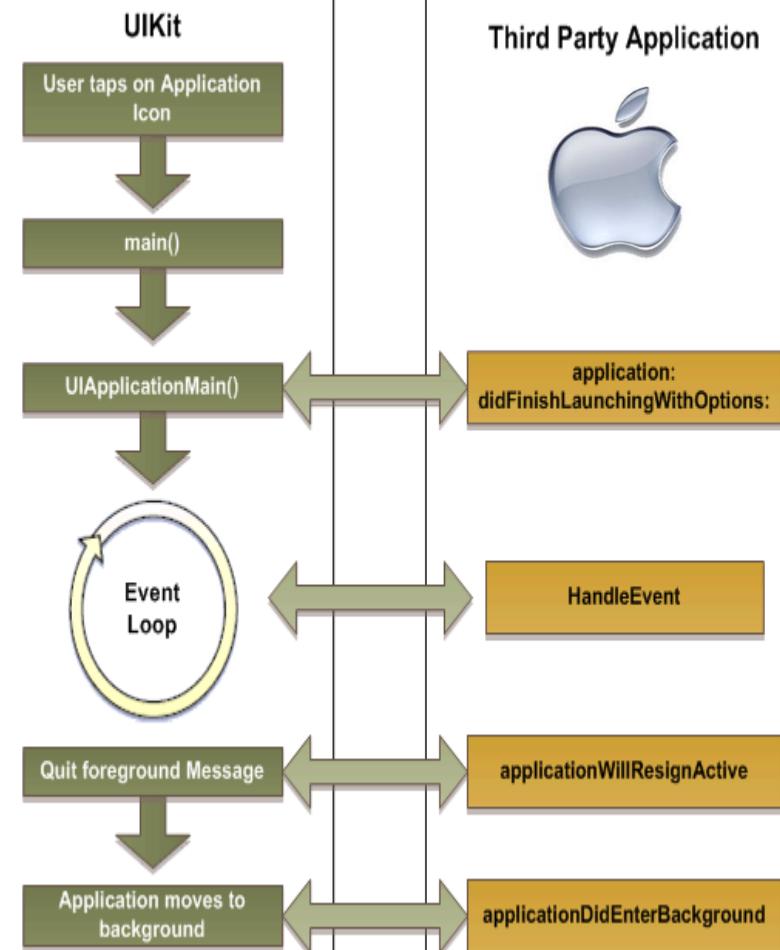


# Lifecycles

iOS &lt; 4



iOS &gt;= 4





# Seguridad

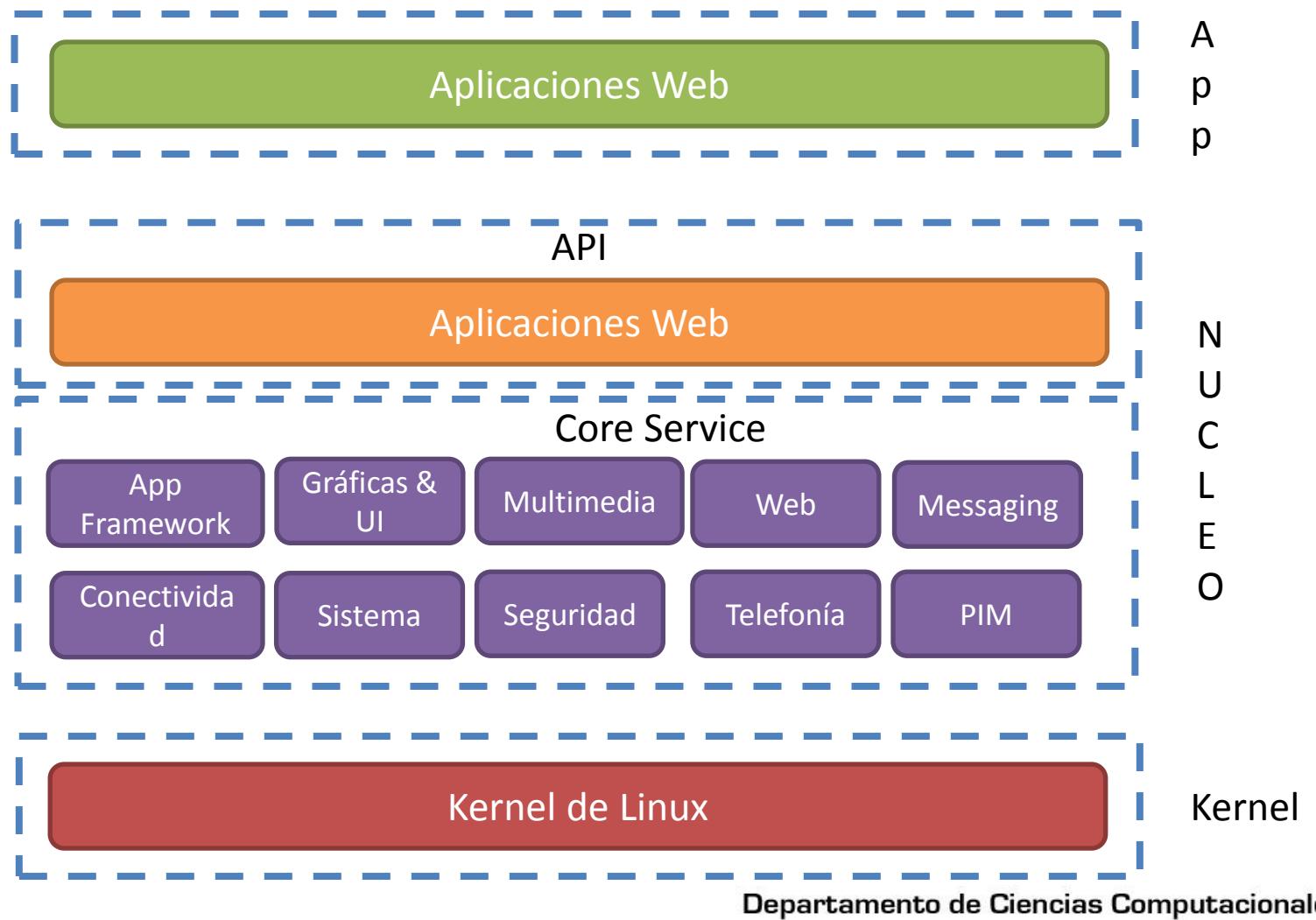
- **Code Signing**
- **Sandbox** para las Apps
- **Security Server** (securityd) protocolos para logar acceso a certificaciones mediante formas autorizadas, implementados en Mac OS X e iOS.
- **Root-Access deshabilitado** (Aunque es posible ejecutar “jailbreaking”)
- **Keychain** ( Información importante es guardada cifrada)
- Conexiones de red seguras en base a SSL o TLS.
- **Mecanismos de autentificación** para Wifi y **manejo de certificados e identidades**, además de **soporte para VPN** (Cisco IPSec, L2TP over Ipsec, PPTP VN).
- **Remote wipe** – Capacidad de borrar el contenido del dispositivo via remoto (para casos de robos o extravíos)



- Originalmente existía el S.O. Meego desarrollado por Intel & Samsung, con el apoyo de “Linux Fundation” se finiquito Meego en el 2011 y en su lugar nace Tizen.
- Enfocado para un S.O. para la industria en variedad de dispositivos móviles (tabletas, teléfonos, tv, etc...)
- Tizen todavía no es liberado (Marzo, 2012) y permanece en **BETA**.
- Originalmente se dijo que Apps de Meego serían completamente exportable a Tizen... Eso quedo descartado.



# Diseño de la plataforma





TECNOLOGICO  
DE MONTERREY.



# Seguridad y Ética

# Encuesta de seguridad ISACA

- 2012 Study on Application Security:
  - A Survey of IT Security and Developers

## Percepciones de seguridad de aplicaciones según :



VS.



Desarrolladores de  
aplicaciones para  
compañías (No  
independientes).

Personal de seguridad encargados de  
realizar auditorias a sistemas  
completos, sean miembros internos de  
empresas o terceros.

Equipo rojo:  
+ 14 mil encuestados  
Muestra de 567 (3.8%)

Equipo azul:  
+ 6 mil encuestados  
Muestra de 256(3.7%)

# Aplicación de seguridad en Aplicaciones (Generales)

¿A que le dan prioridad las empresas? ¿Qué impacto tiene?

79% de los desarrolladores tienen procesos ad-hoc, o no tienen ninguno, enfocado a la construcción de seguridad dentro de Apps

64% del personal de seguridad tienen procesos ad-hoc, o no tienen ninguno, enfocado a la construcción de seguridad dentro de Apps

71% de los desarrolladores sienten que la seguridad no esta manejada en el SDLC (*System development life Cycle*)



51% del personal de seguridad consideran que la seguridad no es manejada en el SDLC (*System development life Cycle*)

30% de los desarrolladores crean (build) la seguridad en la fase de “post-launch”



13% del personal de seguridad consideran que las amenazas inducidas por el codigo (code-induced threats)

# Manejo de fallas y parches de aplicaciones criticas

Las organizaciones no identifican un punto de partida y suelen buscar en otras organizaciones para realizarlo.

47% afirman que no existe un mandato formal y activo para remediar código vulnerable de las aplicaciones.

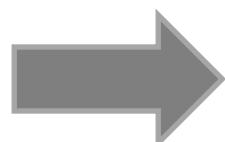
29% afirman que no existe un mandato formal y activo para remediar código vulnerable de las apps.

51% no tienen entrenamiento en seguridad de aplicaciones.



51% no tienen entrenamiento en seguridad de aplicaciones.

54% sienten que reparar bugs y/o parchar aplicaciones es un desperdicio de tiempo y recurso para la compañía.



46% menciona que la metodología de ataque mas utilizada en los últimos 24 meses es “SQL Injection”

# Y esto añadido a Apps

Y a la hora de los golpes, el escenario relacionado a Apps móviles es...

47% menciona que la amenaza emergente mas seria relativa a seguridad de aplicaciones es Web 2.0 o aplicaciones de medios sociales.

29% mencionan que las apps de Web 2.0 o medios sociales son la segunda causa de brechas en datos superados solo por SQL Injection.

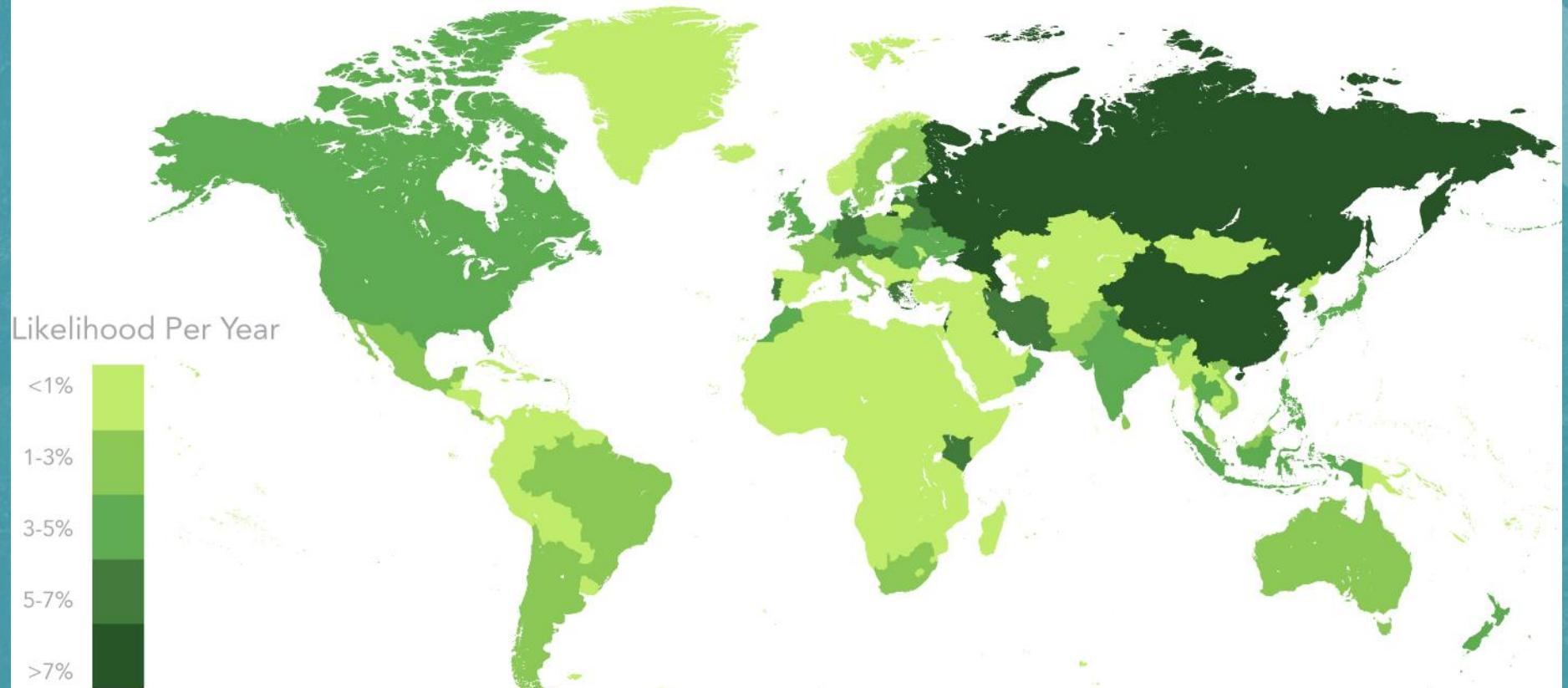
65% no prueban las apps móviles en las fases de Producción, desarrollo o Q/A del SDLC.

46% menciona que la amenaza emergente mas seria relativa a seguridad de aplicaciones es Web 2.0 o aplicaciones de medios sociales.

24% mencionan que las apps de Web 2.0 o medios sociales son la segunda causa de brechas en datos superados solo por SQL Injection.

60% no prueban las apps móviles en las fases de Producción, desarrollo o Q/A del SDLC.

## Annual Mobile Malware Infection Likelihood 2011



\*As of November 2011

# Malware

Malware	Descripción
<b>Mobile Pickpocketing (SMS/call fraud)</b>	Muchos dispositivos móviles tienen la capacidad de hacer cargos a tu cuenta del móvil (Phone bill) vía llamadas y “SMS billing”. Con este método el dinero solo esta a una distancia de un CLIC para una transacción fraudulenta. Es mas fácil que el clásico método por PC. Algunos conocidos: GGTracker, RuFraud, sin olvidar “horoscopos”, fondos de pantalla para Angrybirds, etc.
<b>Botnets</b>	Miles de dispositivos móviles funcionan como Botnet. Algunos botnets conocidos: DroidDrem, Geimini, solamente en el 2011 fueron descubiertas 10 nuevas familias del tipo Botnet.

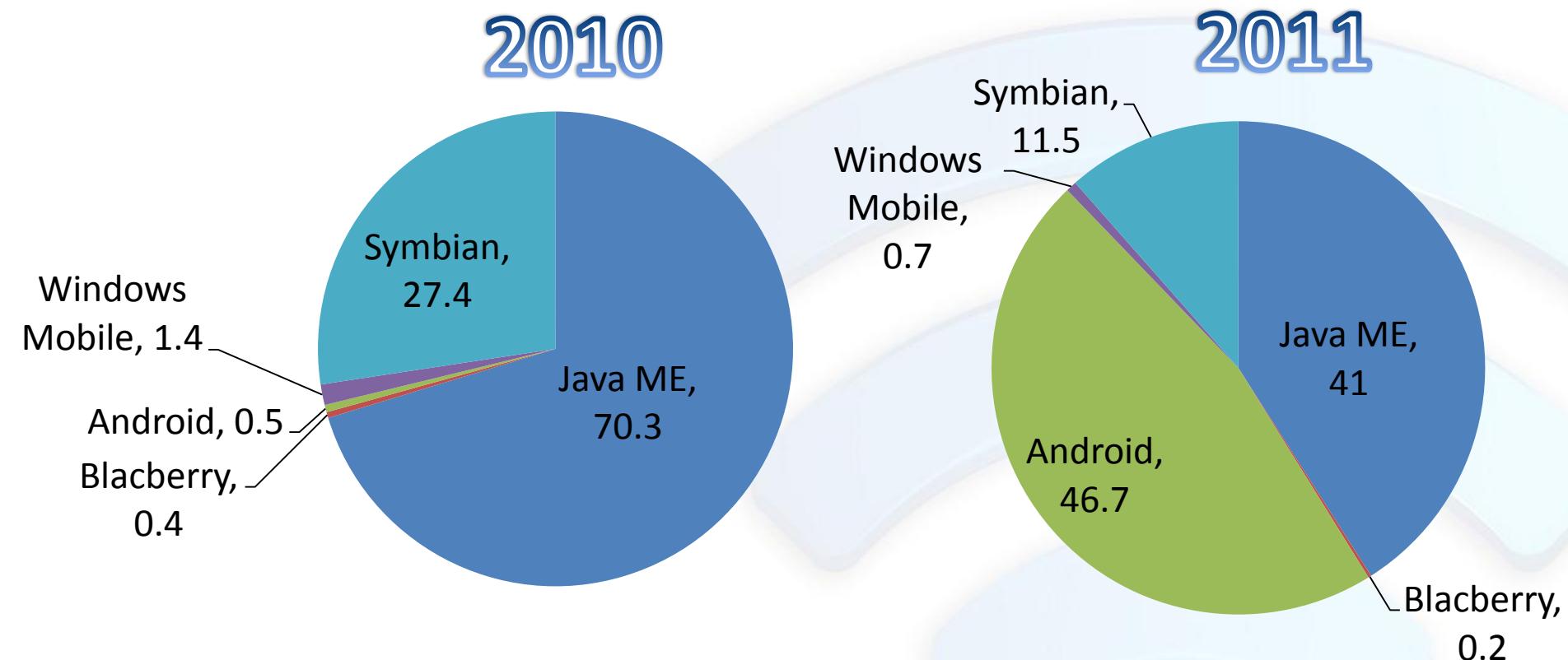
# Malware II

Malware	Descripción
<b>Teléfonos vulnerables (Software)</b>	Una vulnerabilidad de software, sin importar la plataforma, siempre es una puerta a inyección de Malware. DroidDream aprovechaba dos vulnerabilidades de Sistemas Android. En dispositivos móviles , actualizar el software es mas difícil.
<b>Automated Repackaging</b>	Creadores de malware han tenido bastante éxito infectando usuarios con las versiones “repackaged” de las aplicaciones. Se espera que el escenario se vuelve mas critico donde existan aplicaciones que harán el embebido de las aplicaciones originales añadiéndoles malware y subiendo a los mercados

# Malware III

Malware	Descripción
<b>Malversting</b>	Avisos (comerciales) que parecen genuinos pero que envían a sitios fraudulentos. Este método resulta bastante eficiente. Troyanos como GGTracker
<b>Browser attacks</b>	Esto involucra: E-mail, mensajes de textos y sitios web fraudulentos. Se espera un fuerte incremento en phising móvil y mensajes que vinculen a sitios web con descarga automática de malware a los dispositivos. Todo los dispositivos móviles pueden ser victima de estos tipos de ataques y claro, esto significa que sitios legítimos que sean adulterados pueden servir para tal propósito.

# Malware (versiones únicas) para móviles detectados por S.O.



Datos de Juniper Network , reporte “Amenazas Móviles para 2011”

Basado en exámenes a mas de 790,000 aplicaciones y vulnerabilidades de los principales S.O. Móviles

<http://www.juniper.net/us/en/security/>



# App de la muerte (CVE-2011-3918)

**Versiones afectadas:** Android 4 y anteriores.

**Daño:** Negación de servicio del dispositivo.

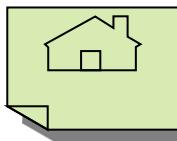
La vulnerabilidad residiría en el proceso 'Zygote' encargado de la compartición de código entre procesos Android y más concretamente en la gestión de las llamadas mediante sus sockets asociados.

Zygote escucha en un determinado socket a la espera de recibir comandos y genera un nuevo proceso bifurcándose ('fork') como un proceso de la capa Linux, fuera de las capas Android.

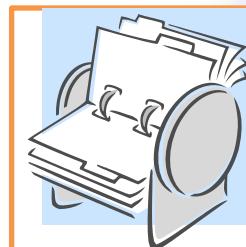
El ataque se da cuando se invocan Zygotes y se dejan en estado Zombie incapaces de ser eliminados por el S.O. y de esta forma agotar los recursos del dispositivo

# Ética

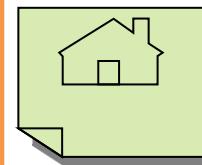
- La aplicación solo debe de hacer lo que dice que hacen
  - Investigadores de Lookout escanearon más de 300 mil apps móviles y un tercio de ellas fueron revisadas a fondo
  - Resultados revelan que mucho desarrolladores no ponen de manifiesto el comportamiento de recolección de datos de una aplicación en sus descripciones.
  - Los desarrolladores utilizan software de tercero y no validan lo que estos recopilan o hacen...



Un tercio  
(Iphone)recopilaba  
info. De donde  
vives



8% Android, 14%  
Iphone (Gratis)  
intentan acceder a  
tu lista de  
contactos.



29% de Apps Android  
(Gratis)recopilan  
donde vives



Los desarrolladores de apps Aunque  
conocen los conceptos de seguridad  
No conocen los detalles y no evalúan a  
fondo Las apps y frameworks de  
terceros.



# Apple & ética

- En el caso donde las Apps son auditadas por las compañías antes de estar disponibles (App Store, Windows Market, Blackberrt Store) pueden ocurrir conflictos si se manejan criterios no claros.
- App Store tiene capacidad de censurar la App si esta muestra material “ofensivo” para ellos.
- ¿Qué ocurre si la App se trata de la App móvil de un diario? ¿Opinión pública debe ser “Aceptable” para Apple?



# Bibliografías

- **Uses and gratifications of mobile application users , ICEIE, 2010 International Conference On, Vol 1, Pag V1-315 – V1-319**
- **Malwarenomics: 2012 Mobile Threat Predictions, Dic 13, 2011**
- 2012 Study on Application Security: A Survey of IT Security and Developers, Ed Adams, Dr. Larry Ponemon, 2012
- Android Developers , <http://developer.android.com>
- Smartphone and APP Grwoth Soar; Lookout; 2012
- Use Permissions to Secure Your Private Data from Android Apps, Raju PP, <http://techpp.com/2010/07/30/android-apps-permissions-secure-private-data/>



# Bibliografías II

- [Application Platform Overview for Windows Phone](#), Microsoft MSDN,
- [Windows Phone 7 Series Architecture Deep Dive](#) . Istvan Cseri. Distinguished Engineer. **Windows Phone**. istvanc@microsoft.com . Please view slides in show mode
- [Inside Windows Phone #08](#): Taking a look inside **Windows Phone Programming Model Architecture** ...
- [Metro, Windows Phone 7 design language – a guest post](#) ,Andrew Spoone, msdn uk team blog.
- [Android to WP7-Chapter 5: A Comparison of Application Life Cycles in Windows Phone and Android; Windows Phone Interoperability](#)



# Bibliografías III

- Security for Windows Phone; MSDN; Marzo 2012;
- Would you Mind Forking this process? A denial of service attack on Android (and some countermeasures); Alessandro Armando, Mauor Migliardi, Luca Verderame;
- The 50 Greatest Gadgets of the Past 50 Years; Dan Tynan; 2005; Pcworld;
- Blackberry OS Report 2; Josh Schiffman; Department of Computer Science and Engineering Pennsylvania State University;
- Overview of HP WebOS; developer HP Palm WebOs
- iOS Developer Library; Apple Developer
- iPhone OS Technology Overview; Appel ; 2008
- Application's Life cycle in iOS4; Adi Saxena; 29 Oct 2010; codeproject.com