Departamento de Informática | Universidad de Valladolid

# Tecnología de Programación

# Introducción

Félix Prieto

Curso 2011/12



## Android: Un poco de historia

- Sistema operativo para dispositivos móviles
- Desarrollo inicial de Android Inc.
- En 2005 la empresa fue adquirida por Google
- En 2007 se funda la Open Handset Alliance
- Primer teléfono disponible en octubre de 2008 (HTC Dream)
- Licencias GNU y Apache





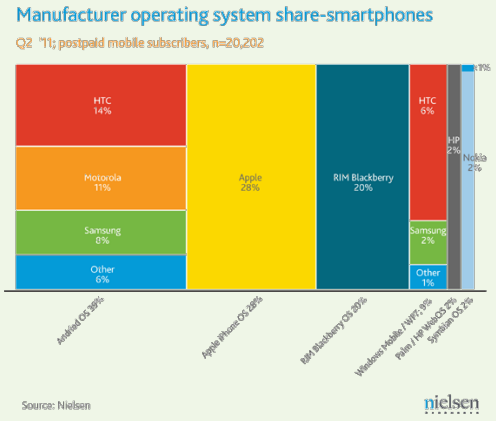
## Otros actores en el mercado

- iOS (Apple)
- BlackBerry (Research In Motion)
- Windows Mobile (Microsoft)
- Palm /HP WebOS (Hewlett-Packard)
- Symbian OS (Nokia?)





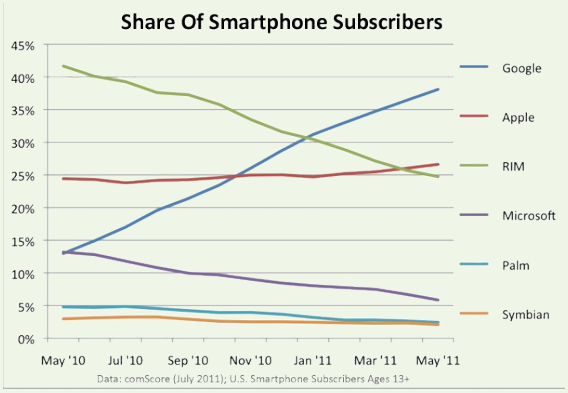
## El reparto de la tarta








## El reparto de la tarta







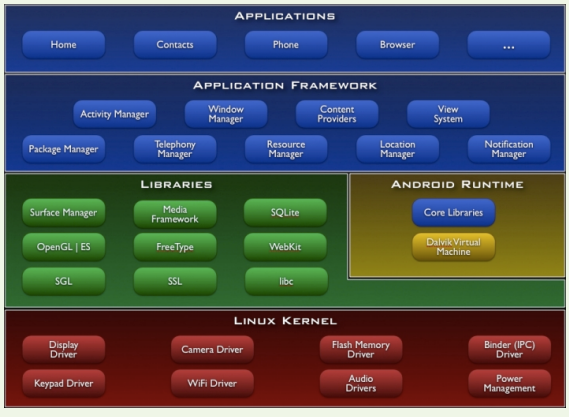

## Algunas sombras

- Guerra de patentes Google/Oracle
- Demandas Apple/Samsung
- Dudas sobre las licencias utilizadas
- Nuevas adquisiciones de Google





## Arquitectura de Android








## Elementos básicos de la arquitectura

- Kernel Linux
- Bibliotecas escritas en C
- Máquina virtual Dalvik
- Framework Java
- Aplicaciones escritas en Java

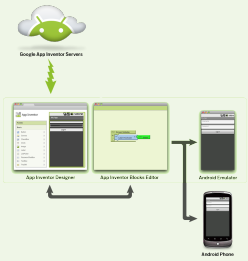
## Herramientas de desarrollo para Android

- App Inventor (http://www.appinventorbeta.com)
- Android SDK (http://developer.android.com/sdk/index.html)
  - Android ADT
- Android NDK (http://developer.android.com/sdk/ndk/overview.html)





## AppInventor



- Editor de la interfaz gráfica en «la nube»
- Control de la lógica mediante «piezas de puzzle»
- Instalación local con simulador de dispositivo y control de dispositivos físicos
- A partir de diciembre liberado como software libre (?)





## Android SDK

- La forma «natural» de programar dispositivos Android
- Un conjunto de bibliotecas
- Herramientas para controlar dispositivos físicos
- Simulador de dispositivos
- Un «plugin» para Eclipse





## Android NDK

- Permite integrar código C y C++ en nuestras aplicaciones
- Permite el acceso directo a las bibliotecas y a los servicios del kernel
- Queda fuera de los objetivos de esta asignatura





## Versiones de Android

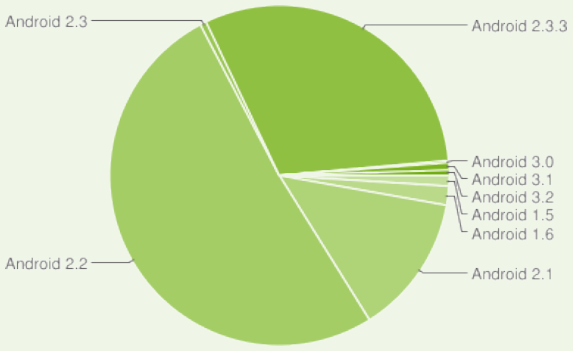
| Plataforma | Código | API Level | Distribución |
|---|---|---|---|
| 1.5 | Cupcake | 3 | 1.0 % |
| 1.6 | Donut | 4 | 1.8 % |
| 2.1 | Eclair | 7 | 13.3 % |
| 2.2 | Froyo | 8 | 51.2 % |
| 2.3/2.3.2 | Gingerbread | 9 | 0.6 % |
| 2.3.3/2.3.4 | | 10 | 30.7 % |
| 3.0 | | 11 | 0.2 % |
| 3.1 | Honeycomb | 12 | 0.7 % |
| 3.2 | | 13 | 0.5 % |

Datos a finales de agosto de 2011





## Versiones de Android








## El Framework de Aplicaciones Android

- Todas las aplicaciones en ejecución usan el Framework
- No podemos escribir un programa Java, rellenar la función main y ejecutarlo
- No tenemos una consola de texto para ejecutar programas
- Los elementos básicos son Activity, Service, Content Provider y Broadcast Receiver
- Objetos `Intent` e `IntentFilter` implementan la petición y oferta de servicios

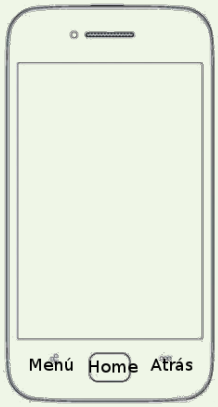




## Activity

- Implementa una tarea específica que requiere la participación del usuario
- Una aplicación suele tener varias Actividades
- Utiliza una o varias «Vistas» (descendientes de `View`) para mostrar información al usuario
- Descendiente de la clase `Activity`, normalmente redefine funciones como `onCreate(Bundle)` y `onPause()` entre otras
- Las actividades pueden quedar en segundo plano
- Las actividades en ejecución son almacenadas en una pila
- El sistema puede finalizar una actividad si no está en primer plano

## Las tres teclas básicas




- Cada dispositivo Android puede tener una configuración de teclado diferente, pero en general todos tienen ciertas teclas
- Home: Nos lleva a la pantalla inicial del dispositivo. La actividad que tenía el foco de ejecución queda en segundo plano
- Atrás: Cierra la actividad que tenía el foco.
- Menú: Opciones elegibles por el usuario en este contexto





## Service

- Permanece en segundo plano
- Proporciona funcionalidad a otros elementos del sistema
- Adecuado para código que debe permanecer largo tiempo en ejecución
- Usado para implementar servicios de geolocalización, reproducción de medios en segundo plano,...





## ContenProvider

- Componente especializada en la gestión de datos
- Los clientes están aislados de la forma de almacenamiento
- Otras componentes acceden a ella mediante un `ContentResolver`
- Accedemos a los datos mediante tablas identificadas mediante una URI





## BroadcastReceiver

- Componente encargada de responder a eventos globales del sistema
- No tiene relación directa con el usuario
- Otra componente dispara uno de estos eventos cuando se cambia de localización, cambia la hora, cambia el día,...
- El receptor tiene que estar suscrito al emisor del evento
- Si el receptor debe realizar tareas complejas, debería delegarlas (en un servicio, por ejemplo) para no bloquear el sistema
- Los widgets son una forma especializada de receptores de eventos





## `Intent` e `IntentFilter`

- Necesitamos un mecanismo de llamada entre los distintos tipos de componentes que podemos utilizar
- La resolución de la llamada en tiempo de compilación es demasiado rígida
- El Framework de Android nos proporciona un mecanismo de llamada resuelto en tiempo de ejecución
- Las llamadas se basan en objetos de tipo `Intent` e `IntentFilter`
- El sistema se encarga de encontrar objetos capaces de realizar la tarea que estamos solicitando





## `Intent`

- Un objeto `Intent` encapsula una petición que deseamos realizar a otro objeto sin conocer su identidad exacta en tiempo de compilación
- Permite iniciar una actividad mediante `startActivity`
- Permite activar un receptor de eventos del sistema mediante la función `broadcastIntent`
- Permite iniciar un servicio mediante `startService` o ligarse a él mediante `bindService`
- Encapsula la acción deseada y los datos necesarios para realizarla





## Dos tipos de `Intent`

- Podemos utilizar dos tipos de objetos `Intent`:
  - *Explícitos:* Llevan especificado de forma explícita el tipo de objeto que debe resolver la petición. Se usan para llamadas internas en la aplicación.
  - *Implícitos:* Sólo expecifican la información necesaria para localizar cualquier objeto capaz de resolver la petición. El sistema se encarga de forma autónoma de determinar qué objeto es el más adecuado para resolver la petición. Se utilizan para llamadas a aplicaciones de las que no concemos detalles concretos

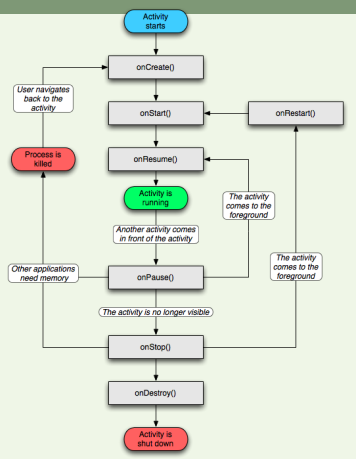




## `IntentFilter`

- Necesito que ciertas componentes describan sus capacidades de modo que el sistema pueda seleccionarlas cuando se solicita una funcionalidad
- Un objeto `IntentFilter` puede encajar con la funcionalidad descrita por un `Intent` de varios modos:
  - Descripción de la acción requerida
  - Categoría de la acción requerida
  - Datos a procesar
- Además incluye una «prioridad» para resolver situaciones en que varios objetos pueden realizar la misma tarea

# Ciclo de vida de las actividades




- Las actividades pueden pasar por varios estados
- El cambio de estado disparan la ejecución de ciertos métodos
- Podemos redefinir los métodos de cambio de estado
- Debemos asegurar que las actividades terminan «adecuadamente»