Programación en otras plataformas móviles

Florina Almenárez Mendoza
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
florina@it.uc3m.es

Contexto

- Conocer generalidades de las plataformas más utilizadas últimamente, como son Android, iOS y RIM, para desarrollar aplicaciones para dispositivos móviles portables

Bibliografía:

- **"Android developer"** disponible en http://developer.android.com/index.html
- **"iOS Developer"** disponible en http://developer.apple.com/library/ios/navigation/
- **Programación de dispositivos móviles portables**, Software de comunicaciones, disponible en http://sites.google.com/site/swcuc3m/home
- **"Blackberry developer: Getting started"** disponible en http://us.blackberry.com/developers/started/

Android

Generalidades

Arquitectura

Modelo de seguridad

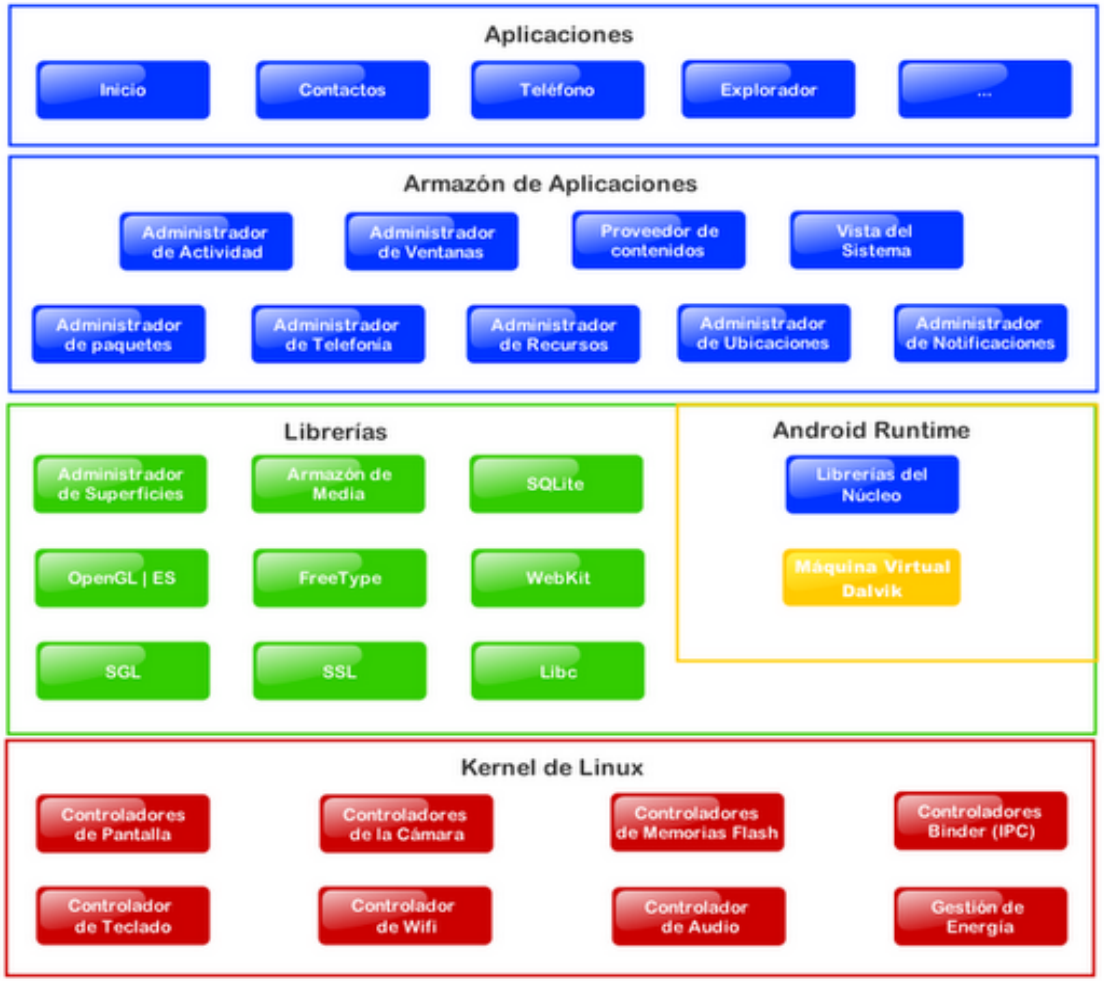
Desarrollo de aplicaciones

Generalidades

- Conjunto de herramientas de software de código abierto creadas por **Google** y **Open Handset Alliance**
- Máquina virtual **Dalvik** optimizada para dispositivos móviles
- **Navegador** integrado basado en **WebKit**
- Biblioteca de **gráficos 2D y 3D** ⇨ OpenGL ES 1.0
- **SQLite** para el almacenamiento de datos estructurados
- Soporte para **audio, vídeo,** y formatos de imagen (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- Interfaces de red: GSM, EDGE, 3G, Bluetooth y Wi-Fi
- **Cámara, GPS, brújula**, y **acelerómetro**
- Entorno de desarrollo ⇨ *plug-in* para el **IDE de Eclipse**

Arquitectura

- Incluye
  - sistema operativo (kernel de **Linux 2.6**)
  - *middleware*
    - **"*framework*" de aplicaciones**
    - **bibliotecas**
    - **entorno de ejecución**
  - **aplicaciones** básicas



Cada capa utiliza servicios ofrecidos por las anteriores, y ofrece los suyos a las capas de niveles superiores

(c) Google

Framework de aplicaciones

- Conjunto de herramientas (APIs) para desarrollar cualquier aplicación
  - *Activity Manager* ⇨ gestiona el ciclo de vida de las aplicaciones
  - *Window Manager* ⇨ gestiona las ventanas de las aplicaciones y utiliza la librería *Surface Manager*
  - *Telephone Manager* ⇨ funcionalidades propias del teléfono (llamadas, mensajes, etc.)
  - *Content Provider* ⇨ compartir datos entre aplicaciones, Por ejemplo, información de contactos, agenda, mensajes, etc., accesible para otras aplicaciones
  - *View System* ⇨ elementos para poder construir y controlar interfaces de usuario (GUI), como listas, mosaicos, botones, "*check-boxes*", etc.

Framework de aplicaciones (II)

- Conjunto de herramientas (APIs) para desarrollar cualquier aplicación
  - *Location Manager* ⇨ información de localización y posicionamiento
  - *Notification Manager* ⇨ las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución, como una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc.
    - si llevan una acción asociada (**`Intent`**) se activa mediante un clic
  - *XMPP Service* ⇨ protocolo de intercambio de mensajes basado en XML

Bibliotecas

- Bibliotecas escritas en C/C++ y proporcionan la mayor parte de las capacidades de Android
- Junto al núcleo basado en Linux, las bibliotecas constituyen el núcleo central de Android
- Entre las más importantes se pueden encontrar:
  - **`libc`** incluye todas las cabeceras y funciones según el estándar del lenguaje C
  - **`Surface Manager`** encargada de componer los diferentes elementos de navegación de pantalla y gestionar las ventanas activas en cada momento
  - **`SGL`** y **`OpenGL/SL`** representan las librerías gráficas: gráficos en 2D y 3D, respectivamente. Permiten utilizar una combinación de ambos tipos de gráficos

Bibliotecas (II)

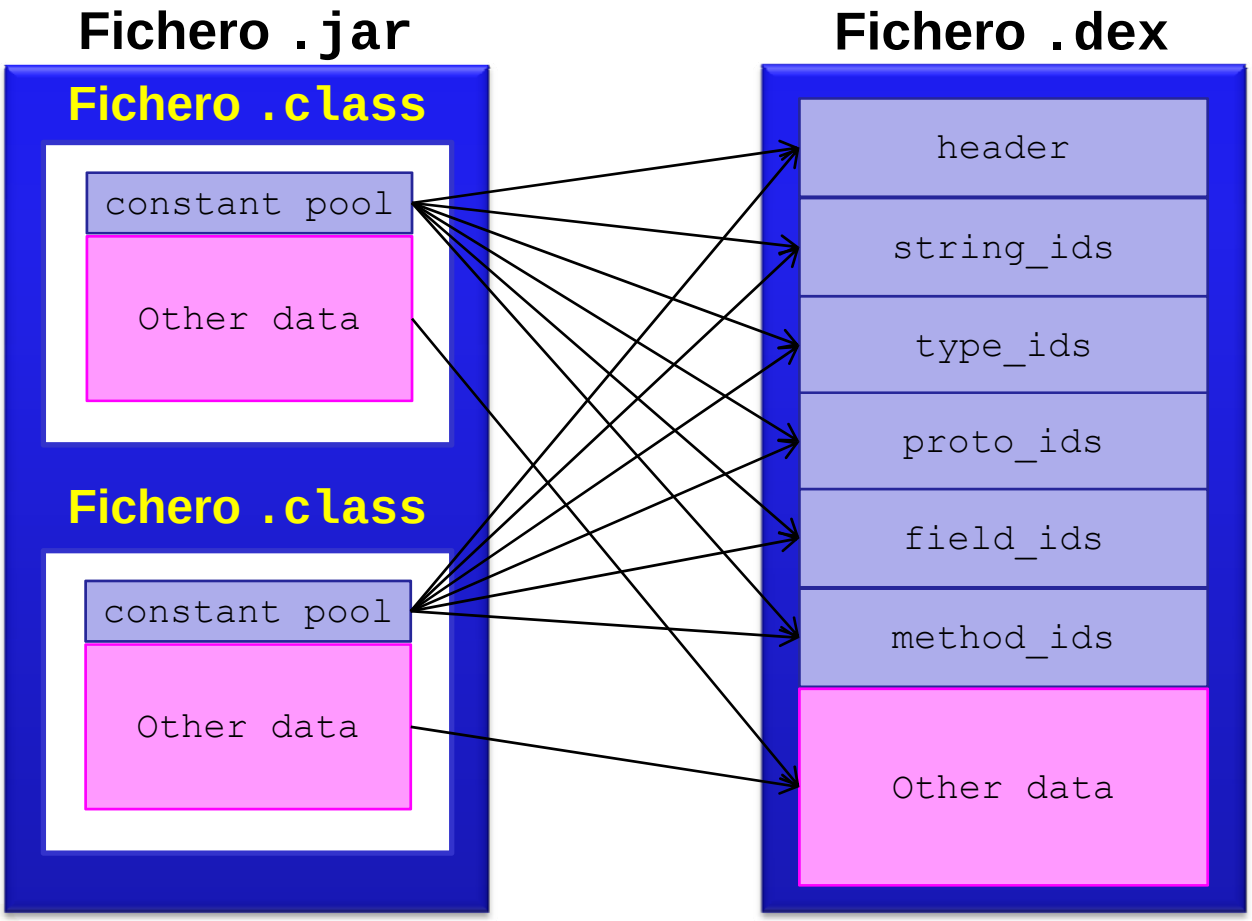
- Entre las más importantes se pueden encontrar:
  - **`Media`** proporciona todos los códec necesarios para el contenido multimedia
  - **`FreeType`** permite trabajar con distintos tipos de fuentes (tipos de letra)
  - **`SSL`** posibilita la utilización de dicho protocolo para establecer comunicaciones seguras
  - **`SQLite`** permite crear y gestionar bases de datos relacionales
  - **`WebKit`** proporciona un motor para las aplicaciones de tipo navegador
    - forma el núcleo del actual navegador incluido por defecto en la plataforma Android

Entorno de ejecución

- El entorno de ejecución está formado por las bibliotecas del núcleo y la máquina virtual Dalvik
  - situado al mismo nivel que las bibliotecas
- **Dalkiv VM** está basada en registro como unidad primaria para almacenamiento (no en pila como la JVM)
  - mejora la recolección de basura
  - no incorpora la característica *Just-in-time* (JIT), de momento
- Dalvik VM es un intérprete que sólo ejecuta los archivos ejecutables con formato .dex (*Dalvik Executable*)
  - delega en el kernel la gestión de hilos, memoria y procesos
  - ejecuta múltiples instancias con muy baja huella
  - herramienta "`dx`" (incluida en el SDK de Android) permite transformar las clases compiladas (`.class`) en formato `.dex`

Formato .dex

Las clases Java son combinadas en uno o más archivos ejecutables `.dex`



.dex comprimidos en un fichero **.apk** (*Android Package*) en el dispositivo

Aplicaciones

- Una **aplicación** se ejecuta dentro de su **propio proceso** Linux
  - el tiempo y ciclo de vida de una aplicación está controlado por el sistema a partir de una combinación de estados
    - qué aplicaciones están ejecutándose, qué prioridad tienen para el usuario y cuánta memoria queda disponible
- Una aplicación debe declarar todas sus actividades, los puntos de entrada, la comunicación, las capas, los permisos, y las acciones a través de `AndroidManifest.xml`
- Componentes
  - clase **`Activity`** ⇨ refleja una actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario (vistas representadas por subclases `View`)
    - navegar de una ventana a otra implica lanzar una actividad o dormir otra
    - **estados** del ciclo de vida: **activa**, **pausada**, **parada** y **reiniciada**

Aplicaciones (II)

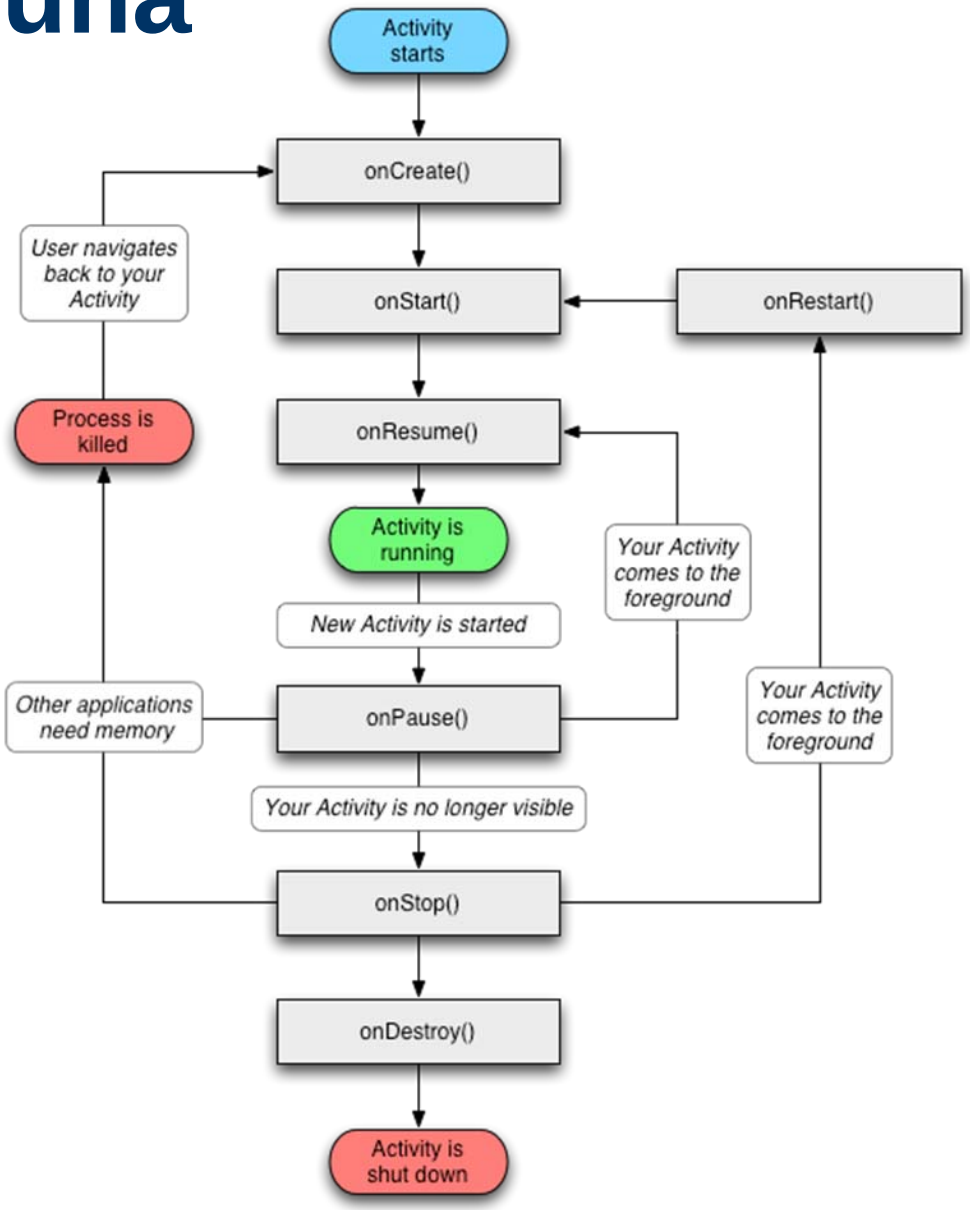
- Componentes
  - **`Intent`** ⇨ realizar una acción, generalmente asociada a datos
    - permiten lanzar actividades
    - delegar el trabajo en otra aplicación, por ejemplo, abrir una URL en algún navegador web
    - están incluidos en el `AndroidManifest` porque describen dónde y cuándo puede comenzar una actividad
  - **`Broadcast Intent Receiver`** ⇨ lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produce
    - generalmente, abrir un componente `Activity`
    - implementado en la clase `BroadcastReceiver`
  - clase **`Service`** ⇨ representa una aplicación ejecutada sin interfaz de usuario

Aplicaciones (III)

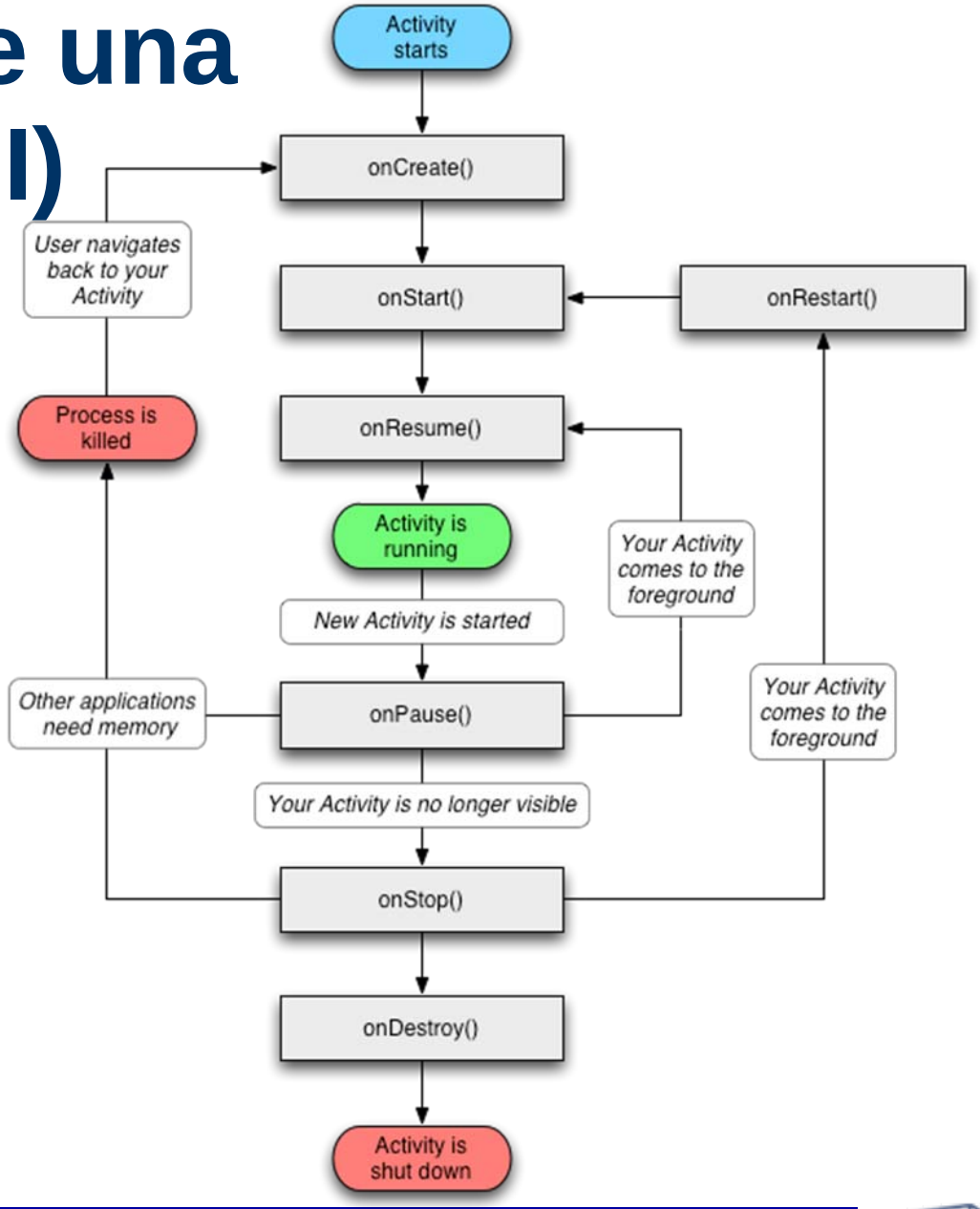
- Componentes
  - **`Service`** ⇨ generalmente en segundo plano mientras otras aplicaciones (con interfaz) están activas en la pantalla
  - **`Content Provider`** ⇨ gestores de contenidos
    - almacenar datos en un fichero, en una base de datos `SQLite` o en cualquier otro formato
    - los datos pueden ser compartidos entre distintas aplicaciones
    - colección de clases para distintos tipos de gestión de datos en el paquete `android.provider`
    - una clase que implemente el componente ⇨ métodos que permiten almacenar, recuperar, actualizar y compartir los datos
- No todas las aplicaciones tienen que tener los cinco componentes
  - cualquier aplicación será una combinación de estos

Ciclo de vida de una `Activity`

- Se lanzan tantos procesos como permitan los recursos del dispositivo
- **`onCreate()`, `onDestroy()`**
  - abarcan todo el ciclo de vida, representan el principio y el fin de la actividad
- **`onStart()`, `onStop()`**
  - representan la parte visible
  - pueden existir otras actividades superpuestas con las que el usuario está interactuando
  - pueden ser llamados múltiples veces



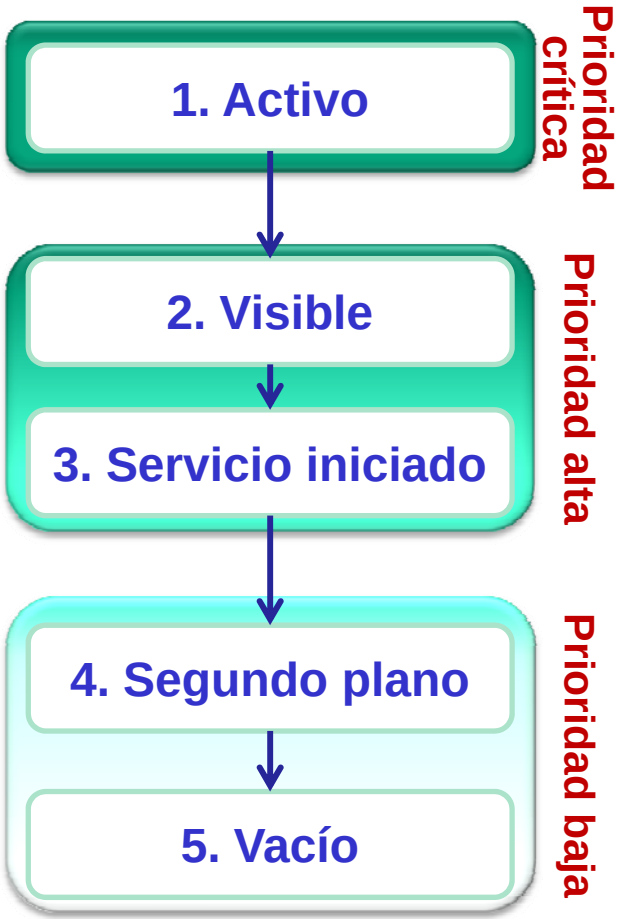
Ciclo de vida de una `Activity` (II)



- **`onResume()`, `onPause()`**
  - delimitan la parte útil del ciclo de vida
  - la actividad tiene el foco de la acción y el usuario puede interactuar con ella
- **`onRestart()`**
  - despertar una actividad que está en segundo plano o invisible

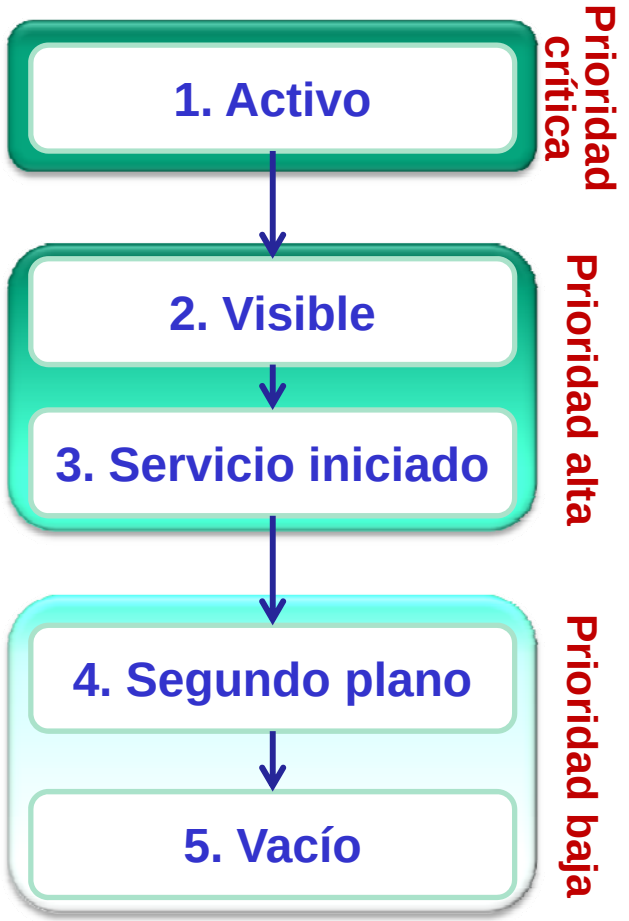
Estados de los procesos

- Cada proceso se sitúa en una jerarquía de "importancia" basada en estados
  1. **Procesos en primer plano:** aloja una `Activity` en la pantalla, con la que el usuario interactúa
     - serán eliminados como último recurso si el sistema necesitase memoria
  2. **Procesos visibles**: aloja una `Activity` pero no está en primer plano
     - la aplicación muestra un cuadro de diálogo para interactuar con el usuario
     - no será eliminado en caso que sea necesaria la memoria para mantener a todos los procesos del primer plano corriendo



Estados de los procesos (II)

3. **Procesos de servicio iniciado**: aloja un `Service` que ha sido iniciado con el método `startService()`
   - no son visibles y suelen ser importantes para el usuario, p.ej. conexiones con servidores
4. **Procesos en segundo plano**: aloja una `Activity` que no está actualmente visible
   - la eliminación de éstos no supone un gran impacto para la actividad del usuario
     - **LRU** (*Least Recently Used*)
5. **Procesos vacíos**: no aloja ningún componente
   - tener una caché disponible de la aplicación para su próxima activación
   - se eliminan con frecuencia para obtener memoria disponible



Modelo de seguridad

- La mayoría de medidas de seguridad entre el sistema y las aplicaciones derivan de los estándares de Linux 2.6
- Ninguna aplicación tiene permiso para realizar operaciones que impacten de forma negativa en la ejecución de otras aplicaciones o del sistema mismo
  - por ejemplo, leer o escribir ficheros privados del usuario no están permitidas
  - declaración explícita de permisos (`<uses-permission>`) para autorizar determinadas acciones ⇨ `<manifest xmlns:android>`
    - o clase `android.Manifest.Permission`
  - ejecución dentro de procesos separados (ID de usuario único)
- Abarca desde el despliegue hasta la ejecución de la aplicación
- Implementación ⇨ aplicaciones firmadas digitalmente para poder ser instaladas en el dispositivo

Entorno de desarrollo

- El entorno por defecto se encuentra integrado por el **SDK** de Android y **Eclipse Classic** versión 3.5 o 3.6
  - http://developer.android.com/sdk/index.html
    - a partir de Windows XP, MAC OS X, Linux
  - ***plug-in* ADT** (*Android Development Tools*)
  - guía de instalación del entorno y desarrollo de aplicaciones
- También disponible en NetBeans
  - instalación de varios complementos que proporcionan toda la funcionalidad necesaria
- **Android NDK** ⇨ desarrollar con código nativo (C/C++)
  - herramientas para generar bibliotecas de código nativo
  - embeber las bibliotecas nativas en un archivo `.apk`
  - cabeceras y bibliotecas nativas del sistema desde Android 1.5

Herramientas de desarrollo

- El emulador permite probar las aplicaciones, emulando tanto hardware como software
  - configuraciones **AVD** (*Android Virtual Devices*)
  - basado en QEMU
- **DDMS** (*Dalvik Debug Monitor Server*)
  - monitor de depuración de la máquina virtual
  - se conecta al **ADB** (*Android Debugger Bridge*) y monitoriza una instancia de máquina virtual
  - información del estado de una instancia Android
  - proporciona servicios de redireccionamiento de puertos, captura de pantallas, captura y listado de información en el dispositivo, hacer *logcats*, ver los procesos y la información del estado de radio, simular llamadas entrantes o SMS, e incluso simular una ubicación, etc.

Hola Mundo Android

R.java | Hello.java

```
package uc3m.it.swc.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Hello extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hola Mundo");
        setContentView(tv);
    }
}
```

iOS

Generalidades

Arquitectura

Modelo de seguridad

Desarrollo de aplicaciones

Generalidades

- Sistema operativo, conocido también como iPhone, OSX, o iPhone OS X, desarrollado y comercializado por **Apple Inc**.
- La interfaz de usuario se basa en el concepto de manipulación directa
  - uso de gestos multi-táctil: "*swiping*", "*tapping*", "*pinching*", y "*reverse pinching*"
  - deslizadores, interruptores y botones
- Soporte de acelerómetros internos para responder a movimientos del teléfono o rotación en 3D
- No existe el concepto de empezar o salir de aplicaciones de Apple
  - las aplicaciones de terceros se cierran cuando se sale de ellas
  - intercambio de datos desde una aplicación a otra

Arquitectura

- La arquitectura está formada por **cuatro** capas de abstracción:
  - **sistema operativo** ⇨ controla el sistema de memoria virtual, los hilos, los ficheros, la red, etc. Incluye el entorno del kernel, los controladores y las interfaces básicas.
  - **Servicios "*Core*"** ⇨ proporciona acceso a los servicios principales del sistema operativo como son el acceso a ficheros, los datos de bajo nivel, servicios de descubrimiento, de red, etc.

    Se utilizan los "*frameworks*" de esta capa si las superiores no cubren las necesidades de las aplicaciones.

Arquitectura (II)

- ***Media*** ⇨ proporciona las tecnologías de gestión multimedia que permiten construir rápidamente aplicaciones con vistas llamativas y sonido

- **"*Cocoa Touch*"** ⇨ permite implementar una aplicación gráfica orientada a eventos.

  MVC (*Model-View-Controller*)

  Lenguaje *Objective-C*

  Esta API proporciona acceso a las funciones claves del dispositivo:
  - multitarea
  - animación
  - reconocimiento de gestos
  - datos: contactos, agenda, …

Frameworks

- Un ***framework*** es un directorio que contiene una biblioteca dinámica compartida y los recursos de soporte necesarios
- *Frameworks* de la capa de **sistema operativo**
  - ***Accelerate*** ⇨ interfaces para cálculos matemáticos, procesamiento de imágenes, etc.
  - ***Bluetooth*** ⇨ interactuar con accesorios "LE" (*Low-Energy*) Bluetooth
  - ***External Accessory*** ⇨ proporciona soporte para comunicación con los accesorios hardware del dispositivo
  - ***Generic Security Services (GSS)*** ⇨ conjunto estándar de servicios relacionados con seguridad (RFC 2743, RFC 4401)
  - ***Security*** ⇨ características de seguridad implícitas y explícitas para proteger los datos: claves, certificados, políticas, etc.

Frameworks de la capa de servicios "*core*"

- ***Accounts*** ⇨ modelo de *single sign-on (SSO)* para ciertas cuentas de usuario
- ***Address Book*** ⇨ acceso programático a los contactos almacenados
- ***CFNetwork*** ⇨ conjunto de interfaces basadas en C que utiliza abstracciones orientadas a objeto para trabajar con protocolos de red, como FTP, HTTP, DNS, SSL/TLS, sockets BSD, etc.
- ***Core Data*** ⇨ gestionar el modelo de datos de una aplicación MVC
- ***Core Foundation*** ⇨ conjunto de interfaces basadas en C que proporcionan gestión de datos básicos (arrays, string, date/time, …)
- ***Core Location*** ⇨ información de localización y posicionamiento para las aplicaciones (GPS, celda, o radio Wi-Fi)
- ***Core Media*** ⇨ proporciona tipos multimedia de bajo nivel

Frameworks de la capa de servicios "*core*" (II)

- ***Core Telephony*** ⇨ interfaces para interactuar con información de telefonía
- ***Event Kit*** ⇨ interfaces para acceso a los eventos del calendario
- ***Foundation*** ⇨ proporciona "envoltorios" (*wrappers*) para muchas de las características encontradas en el *Core Foundation framework*
- ***Mobile Core Services*** ⇨ define los tipos de bajo nivel utilizados en los identificador de tipo uniforme (UTIs, *Uniform Type Identifiers*)
- ***Newsstand Kit*** ⇨ lugar central para usuarios que leen revistas y periódicos
- ***Quick Look*** ⇨ interfaz directa de previsualización del contenido de los ficheros
- ***Store Kit*** ⇨ soporte para la compra de contenido y servicios
- ***System Configuration*** ⇨ permite determinar la configuración de red

Frameworks de la capa *Media*

- ***Assets Library*** ⇨ interfaz basada en consulta para recuperar fotos y vídeos
- ***AV Foundation*** ⇨ clases Objetive-C para reproducir contenido audio y video
- ***Core Audio*** ⇨ soporte nativo para audio (tipos de datos, reproducción y grabación, y procesamiento)
- ***Core Graphics*** ⇨ interfaces para la API de dibujo Quartz 2D
- ***Core Image*** ⇨ conjunto de filtros para manipulación de video e imágenes sin movimiento
- ***Core MIDI*** ⇨ comunicación estándar con dispositivos MIDI
- ***Core Text*** ⇨ conjunto de interfaces basadas en C para trazado de fuentes y manejo de fuentes

Frameworks de la capa *Media* (II)

- ***Core Video*** ⇨ "*buffer*" y *pool* de ellos para *Core Media Framework*
- ***Image I/O*** ⇨ interfaces para importar y exportar imágenes y sus metadatos
- ***GLKit*** ⇨ conjunto de clases de utilidad para simplificar el esfuerzo requerido para crear una aplicación con OpenGL ES 2.0
- ***Media Player*** ⇨ soporte de alto nivel para reproducción de audio y video
- ***OpenAL (Audio Library)*** ⇨ interfaz para entrega de audio en juegos
- ***OpenGL ES*** ⇨ herramientas de dibujo 2D y 3D
- ***Quartz Core*** ⇨ interfaces de animación avanzada ("**Core Animation**") y tecnología de composición

Frameworks de la capa "Cocoa Touch"

- ***Address Book UI*** ⇨ interfaz de programación Objective-C para crear nuevos contactos, y editar y seleccionar contactos existentes
- ***Event Kit UI*** ⇨ proporciona controladores visuales para presentar interfaces estándares para visualizar y editar eventos relacionados con el calendario
- ***Game Kit*** ⇨ permite añadir capacidades de red "peer-to-peer" a las aplicaciones, juegos multijugador
- ***iAd*** ⇨ permite proporcionar anuncios basados en "banner" desde las aplicaciones
- ***Map Kit*** ⇨ proporciona una interfaz de mapa desplazable integrables en las vistas
- ***Message UI*** ⇨ soporte para componer y encolar mensajes de correo electrónico en la carpeta de salida del usuario

Frameworks de la capa "Cocoa Touch" (II)

- ***Twitter*** ⇨ soporte para envío de peticiones Twitter en representación del usuario y para componer y enviar "*tweets*"
- **UIKit** ⇨ infraestructura clave para implementar aplicaciones manejadas por eventos y gráficas
  - gestión de interfaz de usuario, soporte gráfico y de ventanas
  - soporte multitarea y de impresión
  - soporte para personalización de la apariencia de controles
  - soporte para manejar **eventos táctiles** y basados en **movimiento**
  - soporte para texto y contenido Web
  - cortar, copiar y pegar
  - soporte para características específicas del dispositivo: acelerómetros, cámara, biblioteca de fotos, estado de la batería, modelo del dispositivo, sensores de proximida, etc.
  - …

Objective-C

- Lenguaje de programación orientado a objetos
  - encapsulación, herencia, y polimorfismo están presentes
- Extiende el estándar ANSI del lenguaje C ⇨ superconjunto de C
  - misma sintaxis para definir clases, métodos, así como otras estructuras que promueven la extensión dinámica de clases
  - ficheros de cabeceras (`.h`) y de código fuente (`.m` o `.mm`) para separar las declaraciones públicas
- La especificación de una clase requiere de dos piezas: interfaz e implementación
  - **interfaz** ⇨ declaración de la clase y define las variables de instancia y los métodos asociados, se encuentra en un fichero `.h`
  - **implementación** ⇨ código para los métodos de la clase, se encuentra en un fichero `.m`
- Dos tipos de métodos: de instancia (−) y de clase (+)
  - llamar a un método ⇨ pasando mensajes (`[ msg ]`) a un objeto

Modelo de seguridad

- Incluye un demonio llamado el **Servidor de seguridad** que implementa varios protocolos de seguridad, como acceso a objetos `Keychain` y administración de certificados de confianza raíz
- El servidor de seguridad tiene APIs no públicas
- Los servicios de seguridad no proporcionan una interfaz de autenticación
  - no hay necesidad para los servicios de seguridad de tener una interfaz de usuario
- Firma digital de las aplicaciones ⇨ certificados y empaquetado desde el IDE

Entorno de desarrollo

- SDK ⇨ herramientas y recursos para crear aplicaciones nativas y Web
- Permite almacenar datos en el sistema de archivos local e incluso comunicarse con otras aplicaciones instaladas a través de esquemas de URL personalizada
- IDE se denomina **Xcode**
  - sólo disponible para Mac OS
  - el depurador se conecta al dispositivo en tiempo real, dejando al dispositivo el control de la gestión de puntos de interrupción
  - **interfaz *Builder*** es un editor gráfico para el diseño de cada aspecto de la interfaz gráfica de la aplicación
  - el simulador ejecuta la aplicación de la misma forma que lo haría en un iPhone real

Otras características de *Xcode*

- El editor de código fuente permite autocompletado de código, plegado de código, sintaxis resaltada, y mensajes emergentes
- Puede construir, instalar, ejecutar y depurar aplicaciones basadas en “Cocoa Touch”
- Conjunto completo de compiladores de código abierto para C, C++ y Objective-C optimizados por Apple
- Depurador muestra directamente los valores de las variables en en la interfaz
- Realiza análisis estático ya que encuentra errores en el código antes de ejecutar la aplicación
  - errores potenciales que podrían haber permanecido ocultos o casi imposibles de reproducir
- Desarrollo de aplicaciones

Hola Mundo Objetive-C

```
// main.m
#import <UIKit/UIKit.h>
int main(int argc, char *argv[]) {
  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
  int retVal = UIApplicationMain(argc, argv, nil, nil);
  [pool release];
  return retVal;
}

// HelloWorldAppDelegate.h
#import <UIKit/UIKit.h>
@interface HelloWorldAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;

@end
```

Hola Mundo Objetive-C (II)

```
// HelloWorldAppDelegate.m
#import "HelloWorldAppDelegate.h"
#import "MyView.h"

@implementation HelloWorldAppDelegate
@synthesize window;
 - (void)applicationDidFinishLaunching:(UIApplication *)application {
    // Override point for customization after application launch
    MyView *view = [[MyView alloc] initWithFrame:[window frame]];
    [window addSubview:view];
    [view release];
    [window makeKeyAndVisible];
 }
 - (void)dealloc {
    [window release];
    [super dealloc];
 }
@end
```

Hola Mundo Objective-C (III)

```
// MyView.h
#import <UIKit/UIKit.h>
@interface MyView : UIView {}
@end

// MyView.m
#import "MyView.h"
@implementation MyView
 - (id)initWithFrame:(CGRect)frame {
     if (self = [super initWithFrame:frame]) { // Initialization code  }
     return self;
 }
 - (void)drawRect:(CGRect)rect {
     NSString *hello = @"Hello, World!";
     CGPoint location = CGPointMake(10, 20);
     UIFont *font = [UIFont systemFontOfSize:24];
     [[UIColor whiteColor] set];
     [hello drawAtPoint:location withFont:font];
 }
 - (void)dealloc { [super dealloc]; }
@end
```

Hola mundo en C

```
//main.m
#import <stdio.h>
int main( int argc, const char *argv[] ) {
    printf( "hello world\n" );
    return 0;
}
```

Blackberry OS

Generalidades

Aplicaciones

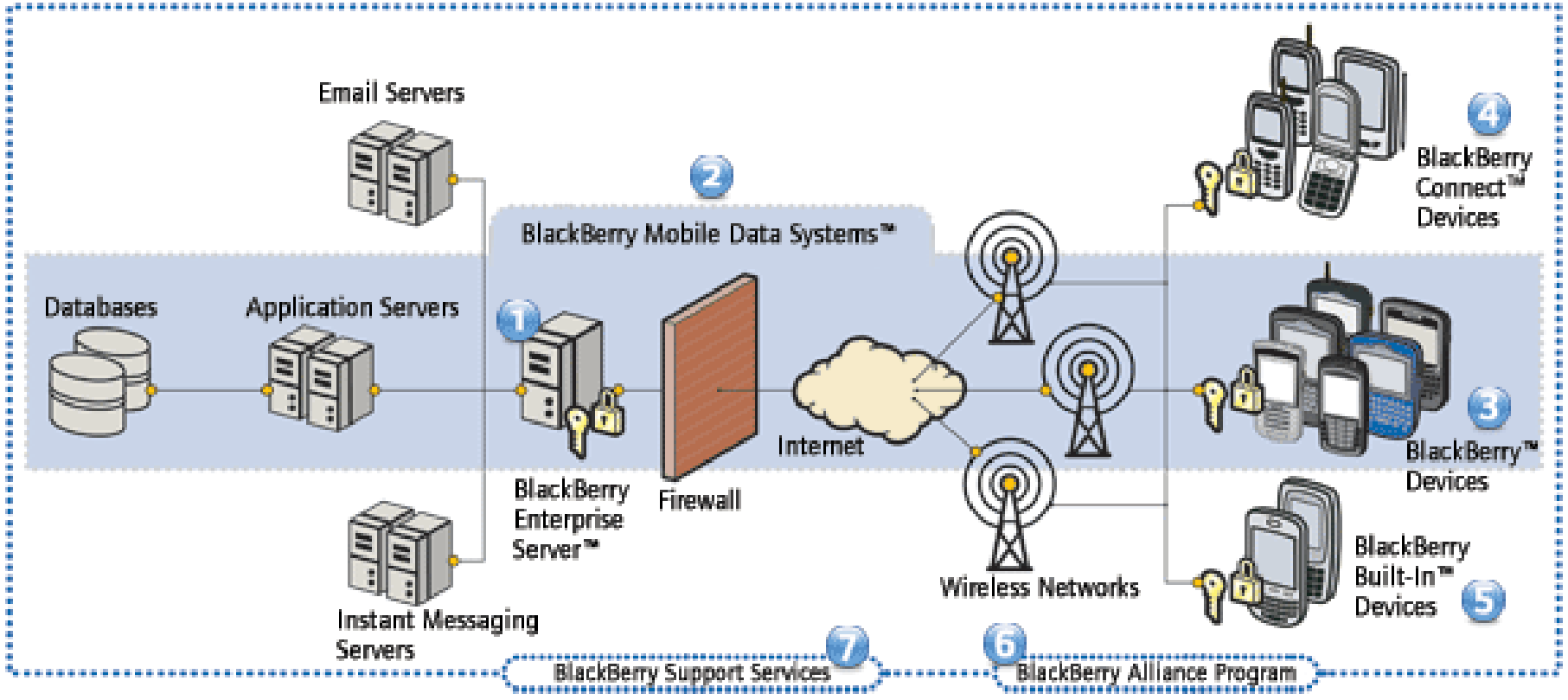
Modelo de seguridad

Desarrollo de aplicaciones

Generalidades

- Desarrollado por Research In Motion (**RIM**)
- Multitarea con soporte de telefonía, mensajería, fax por Internet, correo electrónico “push”, y navegación Web
- Conocido por ser una solución móvil que mantiene a las personas comunicadas en cualquier lugar
- Interfaz con el usuario ⇨ *trackwheel*, *trackball*, trackpad y pantallas táctiles
- Desarrollo de aplicaciones en Java ⇨ Blackberry JVM
  - CLDC, MIDlets, RIMlets
- A partir de la versión 6 incorpora
  - navegador con tecnología WebKit
  - posibilidad de ejecutar juegos 3D
  - soporte para Wi-Fi LBS
  - reconocimiento de rostro en la cámara, etc.

Arquitectura solución móvil



© http://crackberry.com

Aplicaciones

- Aplicaciones MIDP contiene ficheros `.jar` y `.jad`
  - convertidas a ficheros propietarios `.cod` ⇨ herramienta `RAPC`
  - ficheros `.alx` usados para cargar aplicaciones via el software de gestión de escritorio de BB (*Desktop Manager software*)
  - descriptor de la aplicación basado en XML
- RIM API extiende MIDP y con "*look-and-feel*" BB
  - similar a `Swing`
- RIM API (RIMlets)
  - `net.rim.system.Application` (*background*)
  - `net.rim.system.UiApplication` (UI)
  - método `main()` es el punto de inicio
  - método `pushScreen()` visualiza la UI

APIs

- `net.rim.blackberry.api`
  - `browser`: crear páginas HTML
  - `invoke`: llama aplicaciones BB (correo, tareas, etc.)
  - `mail`: leer, escribir, enviar correo electrónico
  - `mail.event`: escuchadores de eventos
  - `phone`: eventos y operaciones de telefonía
  - `pdap`: aplicaciones PDA (tareas, calendario, agenda)
    - o `javax.microedition.pim`
- API del dispositivo
  - bluetooth, batería, compresión, LDAP, UI, HTTP, math, GPS, multimedia, RMS
- API criptográfico
  - `RIMKeyStore`: clases para almacenar y usar claves
  - `TrustedStore`: operaciones inalámbricas

Modelo de seguridad

- Ficheros de clase verificados para conformidad de las interfaces
- Conjunto de APIs limitadas (CLDC)
- Descarga y gestión dentro de la JVM
  - verifica firmas de código válidas
  - desarrollador sólo envía “hashes” de código a un servicio Web
- No cargardores de clases definidos por el usuario
- No JNI o extensiones de usuario
- Clases del sistema no pueden ser anuladas
- Uso de APIs de BB y RIM restringido

Entorno de desarrollo

- Blackberry JDE (*Java Development Environment*)
  - *plug-in* para Eclipse ⇨ JRE y JDK 1.6
  - contiene:
    - API, simulador de teléfono, y simulador MDS
    - cargador Java, *Java Debug Wire Protocol*, RAPC
    - herramienta de firma
  - preverifica ficheros `.cod`
- Herramientas de desarrollo rápido de aplicaciones
  - MDS (*Mobile Data System*) Studio 2.0
  - Blackberry MDS *Runtime*
- *Plug-in* para Microsoft Visual Studio
- Netbeans

Hola Mundo BB

```
class HelloWorld extends net.rim.device.api.ui.UiApplication {

  public static void main(String[] args) {
    HelloWorld appInstance = new HelloWorld();
    appInstance.enterEventDispatcher();
  }

  public HelloWorld() {
    pushScreen(new HelloWorldScreen());
  }

}
```

Hola Mundo BB (II)

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;

class HelloWorldScreen extends MainScreen {

 public HelloWorldScreen(){
   super();
   setTitle("Hello World Title");
   RichTextField hWTextField = new RichTextField("Hello World!");
   add(hWTextField);
 }

 public void onClose() {
   Dialog.alert("GoodBye!");
   System.exit(0);
 }
}
```