Parte II. Computación distribuida en pequeños dispositivos

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

Parte de este material se basa en transparencias de Celeste Campo (celeste@it.uc3m.es), Carlos García Rubio (cgr@it.uc3m.es) y Natividad Martínez Madrid (nati@it.uc3m.es)

Introducción a la programación de dispositivos limitados

Contexto

- **Objetivos**
  - Identificar las características, tipos de clientes y retos de programación de los dispositivos móviles portables
  - Conocer las distintas arquitecturas de desarrollo de aplicaciones para entornos y dispositivos móviles portables
  - Mejorar el desarrollo de aplicaciones para dispositivos móviles

- **Bibliografía**
  - ***Mobile and Wireless Design Essentials***. Mallick, Martyn. Wiley 2003. capítulos 1, 2 y 3.
  - ***Pervasive computing handbook***. Hansmann, Uwe. Springer 2001.

Índice

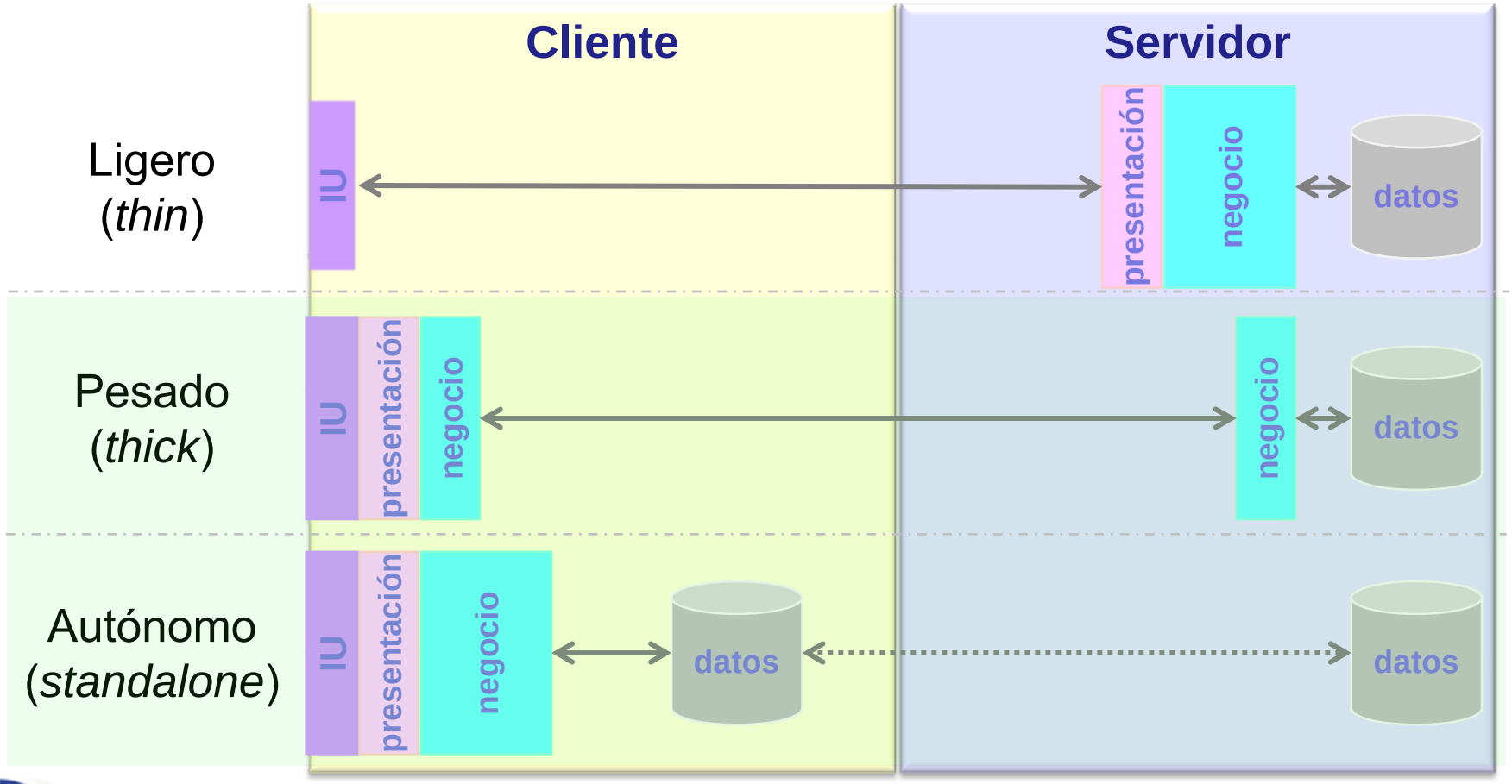
- **Introducción**
- Interfaces de usuario y comunicaciones
- Plataformas de desarrollo
- Herramientas de desarrollo
- Buenas prácticas de programación

Introducción

- **Diversidad de dispositivos móviles** portables asociados al usuario: agendas electrónicas, teléfonos móviles, buscas, etc.
- Multitud de **nuevos dispositivos con capacidad de cómputo**: electrodomésticos, electrónica de consumo, …
- Dichos dispositivos tienen capacidad de **comunicación** ⇨ nuevos protocolos inalámbricos
  - Bluetooth, WLAN, UMTS, WUSB, WiMax, NFC…
- Se puede acceder a **servicios** tradicionales y a nuevos servicios
  - *mBusiness/m-Commerce*, *mLearning*, *m-Marketing*, *m-Health*, etc.
- Nuevas prestaciones o funcionalidades
  - cámara, reproductor MP3, grabadores de voz, GPS, sensores (acelerómetros, de proximidad, de luz, biométricos, etc.)
  - implementación de aplicaciones novedosas ⇨ dispositivos multi-función

Introducción (II)

- El desarrollo de aplicaciones depende del tipo de cliente



Introducción (III)

- Desarrollo de aplicaciones para **clientes ligeros**
  - El procesamiento se realiza en el servidor
  - El cliente sólo se ocupa de la interfaz de usuario (UI)
    - se compone normalmente de un navegador y páginas Web (***Mobile Web***, W3C)
      - ✓ WML (*Wireless Markup Language*) - WAP
      - ✓ cHTML (*compact* o compatible HTML) - iMode
      - ✓ XHTML-*Mobile Profile* (WAPFORUM) o XHTML *Basic* (W3C)
- Los clientes ligeros **requieren conexión a la red** para descargar las páginas Web y recursos asociados
- Las aplicaciones para clientes ligeros son **fáciles de mantener**, pero su interfaz y funcionalidad se encuentra **limitada** por las posibilidades del **navegador**

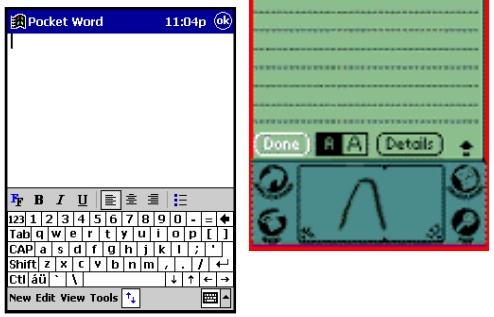
Introducción (IV)

- Desarrollo de aplicaciones para **clientes pesados** o **autónomos**
  - se descargan e instalan en el cliente para su **ejecución en local**
  - pueden trabajar de forma coordinada con un servidor o no
  - **interfaz más flexible** y minimizan el tráfico de red (respecto a aplicaciones de clientes ligeros)
- Presenta nuevos retos, en especial para los dispositivos móviles:
  - **interfaces** con el usuario heterogéneos
  - relativa dificultad para la introducción de datos
  - conectividad **intermitente**, **bajo** ancho de banda, varias interfaces
  - diversas plataformas de desarrollo y entornos de ejecución
  - **limitaciones** en capacidad de cómputo, memoria y almacenamiento
  - **ahorro de energía**, minimizar el uso intensivo de cálculos, ciclos de proceso, gráficos, conexiones inalámbricas, etc.

Índice

- Introducción
- **Interfaces de usuario y comunicaciones**
- Plataformas de desarrollo
- Herramientas de desarrollo
- Buenas prácticas de programación

Interfaces con el usuario
Entrada



- Pantallas sensibles (*touch screen*):
  - Lápiz especial
  - Reconocimiento de escritura o teclado simulado
- Teclado:
  - Más seguro y rápido
  - Teclado reducido o externo (plegable, de goma!)
- Keypad:
  - Datos numéricos y caracteres asignados a números
  - Sistema T9
- Reconocimiento de voz
- Tracking balls, botones, etc.

Interfaces con el usuario
Salida

- Pantalla:
  - Pequeña, pocas líneas.
  - Poca disponibilidad de gráficos, tipos de letra, etc.
- Leds:
  - Actividad de red, estado de la batería.
- Audio
- Vibrador

Interfaces de comunicaciones

- **WAN** (*Wide Area Network*)
  - GSM, GPRS, UMTS
- **LAN** (Local Area Network)
  - Wi-Fi
- **PAN** (Personal Area Network)
  - Bluetooth, IrDA, WUSB
- Incluso con soporte de **WiMax, NFC** (*Near-field Communication*)

- Algunos de estos interfaces vienen integrados en el propio dispositivos y otros se incluyen a través de tarjetas de expansión (CF/SD WiFi – CF/SD Bluetooth)

Índice

- Introducción
- Interfaces de usuario y comunicaciones
- **Plataformas de desarrollo**
- Herramientas de desarrollo
- Buenas prácticas de programación

Plataformas de desarrollo

- Desarrollo de aplicaciones “pesadas” o autónomas
  - ✓ riqueza de contenidos y personalización de servicios proporcionados por terceras partes
- Plataformas de ejecución nativas ⇨ sistemas operativos
  - − cada plataforma tiene su propio lenguaje, herramientas de desarrollo y APIs con las que crear aplicaciones
  - ✓ acceso completo y en tiempo real a todas las funcionalidades del dispositivo
- Desarrollo multiplataforma
  - ✓ portabilidad de aplicaciones entre dispositivos y distintos fabricantes
  - × en general, el acceso no es completo ni de la misma manera a todas las APIs nativas
  - × podrían ser más costosas que ejecutar una aplicación nativa
    - o p.ej. renderizar HTML e interpretar Javascript en HTML 5

Plataformas de los S.O.

- Windows CE (**Windows Phone**)
- **Symbian** (Symbian OS 9.5, **Symbian^3**)
- **Android** (Google)
- MAC **OS X** (iOS)
- **Embedded/Mobile Linux** (Maemo, Meego, Moblin, OpenZaurus, MobiLinux, LiMo platform)
- **HP webOS** (anterior Palm Web OS)
- Otros: **RIM** Blackberry **OS, Hiptop**, propietarios (Bada…)

Plataformas de los S.O. Windows CE

- Plataforma basada en un sistema operativo de 32 bits, modular y de tiempo real
- La primera versión se distribuyó en Noviembre 1996
- Fundamentalmente PDAs y teléfonos móviles de alta gama
  - incluye Windows Phone, Windows Mobile, Pocket PC, Smartphone. Además de “Portable Media Center” para coches
- Utiliza los mismos lenguajes y entornos de desarrollo que se emplean con Windows para PC
  - código nativo: C/C++ (Visual C++)
  - código manejado ("*managed code*"): Visual Basic .NET, C#
  - J2ME, Python, …
- A partir de la versión 4.2 ⇨ Windows Mobile 2003 (6.5)
- Windows Phone 7 series ⇨ Photon (WM7)

Plataformas de los S.O. Symbian

- Empresa fundada por Nokia, Motorola, Ericsson y Psion
  - versión 6 de EPOC, 1998
  - Psion ⇨ en 1989 comenzó a desarrollar EPOC (para PDAs)
- **Objetivo**: crear un sistema operativo para dispositivos inalámbricos, especialmente teléfonos móviles
- Sistema operativo de 32 bits con características de tiempo real y multitarea
  - Series 40 (pantallas más pequeñas), 60 y 80 (*Communicators*)
- Desarrollo de aplicaciones:
  - Código nativo: C/C++
  - OPL (< v8), **Python**, Visual Basic, Simkin, Perl, J2ME
- Symbian C ⇨ actualización Symbian Belle, Anna, Symbian^3
- Herramienta **Qt** ⇨ unificará Meego y Symbian

symbian

Plataformas de los S.O. Embedded Linux

- Mismo software que en el PC o servidor pero en un dispositivo limitado
  - Linux empotrado puede ocupar aproximadamente 2 MB
- Ventajas de ser software libre: disponibilidad de fuentes, modificación y adaptación del sistema operativo (a medida)
- Qtopia, Meego, OpenMoko, MobiLinux, …
- Se comercializan PDAs y móviles con Linux y también existen distribuciones para instalar sobre otros sistemas
  - Motorola presentó su primer teléfono basado en Linux en 2003
- Librerías compactas de `glibc` and `gcc`
- **ELCPS**, *Embedded Linux Consortium Platform Specification*

Plataformas de los S.O. Android

- Plataforma de software que incluye un SO basado en Linux y desarrollado por Google y Open Handset Alliance
- Características
  - núcleo monolítico
  - pantalla táctil
  - teclado QWERTY
  - OpenGL, SQLite, OpenSSL, …
- Desarrollo oficial de aplicaciones
  - principalmente en Java ⇨ APIs propietarios y VM **Dalvik**
  - herramienta para desarrollo nativo (NDK, *Native Development Kit*)
  - HTML 5
- Primer dispositivo (2008) ⇨ **T-Mobile G1/HTC Dream**
- Utilizado tanto en *smartphones* como en *tablets*

Plataformas de los S.O.
iOS

- Sistema operativo móvil de Apple desarrollado originalmente para iPhone
  - siendo usado además en iPod touch , iPad y Apple TV
- Derivado de **Mac OS X**, optimizado para procesadores ARM
- 4 capas de abstracción
  - núcleo del SO
  - servicios principales
  - Media
  - cocoa *touch* (interfaz gráfica de usuario multi-táctil)
- Desarrollo de aplicaciones
  - C, Objective-C (orientado a objetos), Pascal, Java
  - HTML5
  - iPhone SDK (Xcode)

Plataformas de los S.O.
RIM OS

- **RIM** (*Research In Motion*) **OS** para Blackberry
  - última versión Blackberry OS 7 (Agosto, 2011)
- Arquitectura orientada a eventos, multitarea y con soporte especializado de dispositivos de entrada (*trackball*, *trackpad*, pantalla táctil, etc.)
- Desarrollo de aplicaciones
  - C, (Visual) C++, Java (JME)
  - HTML 5
- Plataforma basada en **QNX** para BlackBerry Tablet
  - sistema operativo similar a UNIX
  - arquitectura micro-kernel de tiempo real
    - procesamiento distribuido ⇨ intercambio de mensajes entre procesos
  - posible reemplazo de BlackBerry 8

Plataformas de los S.O. WebOS

- Jeff Hawkins desarrolló la primera versión de Palm, 1996
  - pensado exclusivamente para PDAs
- Características (Garnet OS)
  - mono-tarea, sistema de ficheros ⇨ Utiliza base de datos para representar archivos ejecutables y datos
  - Necesidades asequibles de potencia (16-33 Mhz)
- Desarrollo de aplicaciones
  - Código nativo: C/C++
  - Visual Basic, J2ME, Python, …
  - aplicaciones ejecutables ⇨ archivos con extensión PRC
- Palm ALP (Access Linux Platform), 2006/2007
- Palm WebOS (Enero, 2009) ⇨ basado en Linux
  - HP Web OS 2.0 (2010) con futuro incierto

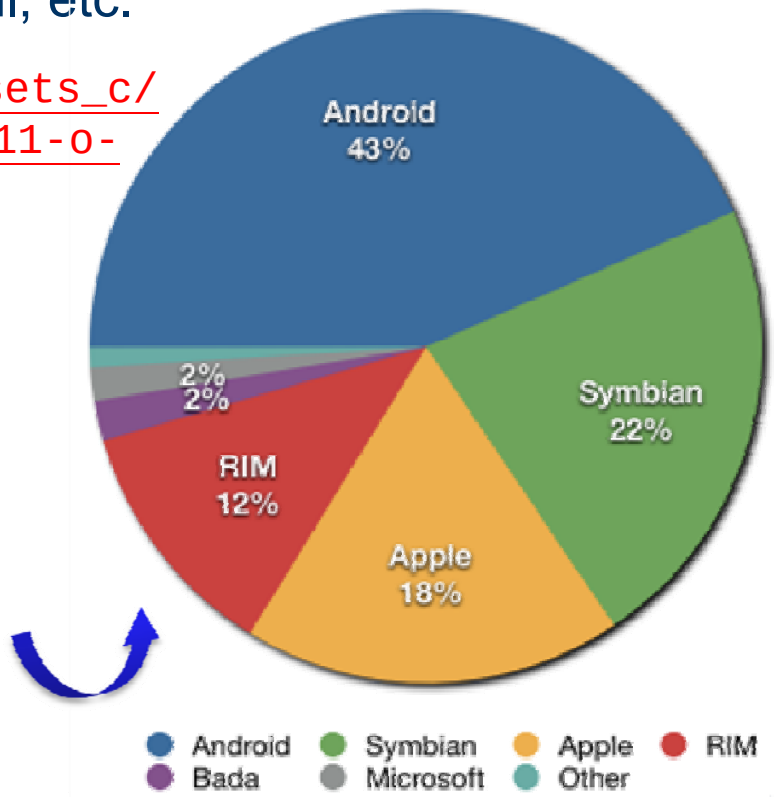
Plataformas S.O.
Cuotas de mercado

En el siguiente enlace se muestra como **iOS** (Apple) ocupa el primer lugar en norte América, Europa, Japón y Australia, mientras que **Symbian** lo ocupa en China, India, Brasil, etc.

http://www.readwriteweb.com/mobile/assets_c/2011/02/global-os-marketshare-feb.2011-o-27904.php

Estudio de Febrero de 2011, InfoGraphic

No obstante, en un estudio de la consultora **Gartner** sobre las ventas de "*smartphones*" a usuarios finales en el segundo cuarto de 2011, se muestra que **Android** ocupa el primer puesto



Plataformas de los S.O. Dispositivos

Windows CE

iPhone

Android

BlackBerry

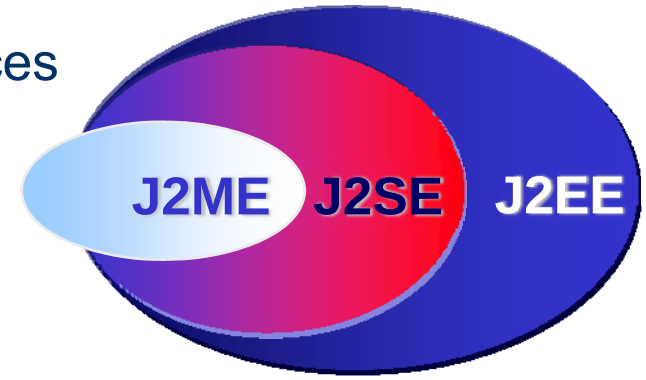
Symbian

Embedded Linux

Otros

Multiplataforma Mobile Java

- Java Micro Edition (ME) o **J2ME**
- Plataforma estándar para la programación de aplicaciones Java en dispositivos limitados
  - versión muy simplificada de J2SE
- Abarca un gran tipo de dispositivos limitados, no sólo teléfonos móviles
  - de consumo y embebidos
  - PDAs, buscas, electrodomésticos inteligentes, etc.
- Capacidades gráficas para diseñar interfaces de interacción con el usuario
- Historia
  - **PersonalJava** (1997), JDK 1.1.8
  - **EmbeddedJava** (1998)



Multiplataforma
JavaFX Mobile

- Nueva familia de tecnologías Java para desarrollo de aplicaciones multimedia (*Rich Internet Applications*, RIAs)
  - anunciada por primera vez en mayo de 2007
  - lenguaje de script que forma parte de la tecnología JavaFX
- Implementaciones de Java ME y Java SE corriendo sobre un kernel Linux
  - mejorar la portabilidad y rendimiento de las aplicaciones
  - competidor de SilverLight, Flash o AJAX
- Compatibilidad y futuro incierto en móviles de gama media y alta
  - posiblemente móviles LG y Sony Ericsson
  - dispositivos con soporte Java, p.ej. Nokia, HTC (Windows Mobile)
- A partir de la v1.1 del SDK incluye emulador para móviles

Multiplataforma
Mobile Web

- W3C ha desarrollado tecnologías Web que explícitamente tienen en cuenta las especificaciones de los dispositivos móviles
  - **CSS (*Cascading Style Sheet*) Mobile** ⇨ lenguaje para estilo de páginas Web
  - **SVG (*Scalable Vector Graphics*) Tiny** ⇨ un perfil del formato de gráficos vectoriales escalables
  - **XHTML *For Mobile*** ⇨ define un subconjunto de XHTML
- La última generación de navegadores móviles son capaces de utilizar tecnologías Web más avanzadas ⇨ no sólo clientes ligeros
  - **HTML5**
  - CSS 2.1 y 3
  - APIs JavaScript ⇨ desarrollo de *widgets*
- Validador ⇨ mobileOK

Multiplataforma HTML5

- Con HTML5 la filosofía de las aplicaciones Web ha cambiado
  - permite desde documentos estáticos hasta aplicaciones dinámicas
  - APIs para almacenamiento persistente en local con bases de datos o almacenamiento de objetos (soporte "*offline*")
  - posibilidades de animación, visualización de contenidos multimedia, y otras funcionalidades hasta ahora exclusivas de los lenguajes nativos
- HTML5 ⇨ JavaScript/CSS/JSON/XML
- Adaptando el estilo (CSS3) por cada dispositivo se puede rediseñar la navegación
- Se encuentra en modo experimental
- Soportado por los navegadores de los sistemas operativos: iOS, Android 2.0, Qt de Nokia, Windows Phone 7, Blackberry OS 7, WebOS 2.1

Índice

- Introducción
- Interfaces de usuario y comunicaciones
- Plataformas de desarrollo
- **Herramientas de desarrollo**
- Buenas prácticas de programación

Herramientas de desarrollo

Plataforma S.O.	Lenguaje	IDE	Plataforma de despliegue	Emulador disponible	Costo
Windows Mobile	C, C++	Visual Studio 2010, 2008, 2005, eMbedded VC++ (gratuito), Satellite Forms	Windows Mobile, Windows CE 5.2	con el IDE	eVC++ gratuito, comerciales
Windows Phone	C#	Visual Studio 2010	Windows Phone WCE 7	con el IDE	Comercial
Symbian	C++	Carbide, opciones dependiendo de la serie Qt SDK	Depende de la serie	Gratuitos, con los SDKs	Herramientas gratuitas y comerciales
Embedded Linux	C, C++	Depende de la distribución	Depende del dispositivo		
Android	Java, con porciones de código que pueden estar en C, C++	Eclipse (Android SDK), Plug-in Netbeans, NDK (desarrollo nativo)	Android	Multiplataforma y gratuito en el SDK	Gratuito

Herramientas de desarrollo (II)

Plataforma S.O.	Lenguaje	IDE	Plataforma de despliegue	Emulador disponible	Costo
iOS	Objective-C	iOS SDK (Xcode)	iPhone, iPad, iPod Touch	Con el iPhone SDK, integrado en Xcode. iPhoney	Gratuito para ordenadores Mac. Pruebas en simulación gratuitas, pero la instalación en el dispositivo requiere pago licencia de desarrollador
RIM Blackberry	Java	Eclipse	Blackberry	Si	Gratuito
WebOS	JavaScript, CSS, HTML, C and C++	Eclipse	WebOS, Palm	Con el SDK	Licencia de desarrollo gratuitas; licencias de despliegue gratuitas y comerciales
Palm OS	C, C++, Pascal	Palm OS SDK (Eclipse), CodeWarrior, PocketStudio, HB++, Satellite Forms	Palm OS, o Windows Mobile con emulador StyleTap	v1.0 - 4.1: PalmSource (Access); v5.0: - 5.4 Palm (palmOne)	Gratuitos (p.ej. Palm SDK) y comerciales (p.ej. CodeWarrior)

Herramientas de desarrollo (III)

Plataforma	Lenguaje	IDE	Plataforma de despliegue	Emulador disponible	Costo
Java ME	Java	Java ME SDK, plug-ins Eclipse, Netbeans	Windows CE, Symbian, Linux, Palm OS (.rpc) Algunas especificas del dispositivo	Sun Wireless Toolkit, emuladores propietarios (p.ej. Nokia)	Gratuito
JavaFX Mobile	Java	JavaFX SDK	Windows Mobile, Symbian, Linux	Integrado con el SDK	Gratuito
HTML5	Javascript, CSS3, JSON, XML	Varias opciones, p.ej. Google Web Toolkit/Eclipse, dreamweaver, Netbeans, Phonegap	iOS, Android 2.0, Qt de Nokia, Windows Phone 7, Blackberry 7, WebOS 2.1	Con SDKs, Ripple Emulator	Licencias comerciales y gratuitas

Herramientas de pruebas de aplicaciones móviles

- Herramientas que permitan realizar pruebas de:
  - funcionalidad, rendimiento, compatibilidad, interoperabilidad, usabilidad, seguridad y consumo
- Emuladores y validadores (Web) permiten llevar a cabo algunas pruebas de forma manual
- Otras herramientas gratuitas
  - FoneMonkey ⇨ iPhone (pruebas funcionales automatizadas)
  - iPhoney ⇨ iPhone (pruebas de compatibilidad)
  - Robotium ⇨ Android (pruebas GUI automatizadas)
  - WTK 2.5 ⇨ MIDP (pruebas funcionales y rendimiento)
    - *Memory monitor, network monitor, profiling*
  - Nokia *Energy profiler application* ⇨ Symbian (pruebas de consumo)
- Blog *Mobile Application Testing*

Índice

- Introducción
- Interfaces de usuario y comunicaciones
- Plataformas de desarrollo
- Herramientas de desarrollo
- **Buenas prácticas de programación**

Buenas prácticas de programación

- Al desarrollar software para dispositivos limitados es muy importante realizar optimizaciones siempre que “sea posible”
  - capacidades limitadas, tamaños pequeños de pantalla, …
  - optimización puede influir en el tiempo de desarrollo y la portabilidad de las aplicaciones
- Importante realizar pruebas de las aplicaciones
  - a través de diversas plataformas ⇨ compatibilidad y rendimiento
  - interfaces, uso de recursos, consumo de batería, etc.
- Algunos consejos de programación en lenguajes **orientados a objetos** se centran en:
  - uso de memoria
  - uso de hilos
- Buenas prácticas de programación Web ⇨ Recomendación W3C

BPP: uso de memoria

1. Eliminar clases que no son necesarias
2. Disminuir los niveles de jerarquía de clases:
   - Compromiso entre modularidad de código y tamaño.
3. Utilizar nombre cortos para los paquetes, clases, métodos y miembros de clase.
   - Intentando no perder la legibilidad del código.
4. Evitar inicializaciones de arrays en el código
5. Liberar objetos (poniendo la referencia a `null`)
6. Tener en cuenta la complejidad de las operaciones matemáticas
7. Utilizar tipos primitivos en lugar de objetos
8. Intentar pasar tan pocos parámetros como sea posible en métodos que son frecuentemente invocados

BPP: uso de hilos

- Un hilo es un flujo de control secuencial individual dentro de un programa
- Los hilos permiten a la aplicación realizar múltiples actividades simultáneamente:
  - El interfaz de usuario se mantiene activo mientras se realizan operaciones largas (comunicación por red o cálculos complicados)
- Evitar el ***deadlock***:
  - Dos hilos están en "*deadlock*" si cada uno intenta bloquear datos que ya han sido bloqueados por el otro.
  - El hilo A bloquea el objeto X y el hilo B el objeto Y. A intenta acceder al Y al mismo tiempo que el hilo B intenta acceder al X
- Por ejemplo, en Java ME implementar **`Runnable`** es más eficiente que utilizar directamente `Thread`
- Implementar **espera suspendida** que consume menos recursos

BPP: Aplicaciones Web móvil

- Uso conservador de recursos
  - Compresión para transmitir datos ⇨ *HTTP 1.1 compression*
    - balance entre eficiencia y uso de la batería y tiempo para descompresión
  - Minimizar el tamaño de los datos y de la aplicación
    - procesar archivos HTML, JavaScript y CSS para eliminar espacios en blanco y comentarios
    - optimización a través de la sustitución global de tokens (variables, nombre de métodos, etc.) con nombres alternativos más cortos
  - evitar redirecciones
  - optimizar peticiones de red
    - procesamiento por lotes, conocimiento de la conectividad, monitorizar actividad del usuario para tomar decisiones en períodos de inactividad
  - minimizar el uso de recursos externos

BPP: Aplicaciones Web móvil (II)

- Uso conservador de recursos
  - agregar imágenes estáticas en un solo recurso compuesto (sprites)
    - p.ej. imágenes de fondo pueden ser codificadas usando el esquema URI: `url(' [data])`
  - almacenar recursos dinámicos identificándolos con una URI que incluye un “hash” del contenido del recurso (cache)
  - los datos accesibles desde peticiones AJAX desde el cliente deben ser “cacheados” como los contenidos principales
  - no enviar información de cookies innecesariamente
  - mantener el tamaño del modelo de objetos del documento (DOM) razonable
    - p. ej. usando paginación

BPP: Aplicaciones Web móvil (III)

- Datos de aplicación (usuario y personalización)
  - utilizar pocas cookies, o incluso ser funcional si las cookies no están disponibles
  - utilizar tecnologías de almacenamiento apropiadas del lado del cliente
    - p.ej. APIs de BONDI, HTML5, Opera Widgets
  - replicar datos locales en un servidor si es necesario
- Seguridad y privacidad ⇨ OWASP
- Control del usuario
  - el usuario debe estar informado acerca del uso de la información personal y del dispositivo
  - habilitar registro de usuario automático
- Experiencia de usuario
  - diseñar varios métodos de interacción
    - basado en cursor ("*focus*"), puntero o táctil

BPP: Aplicaciones Web móvil (IV)

- Experiencia de usuario
  - optimizar el tiempo de arranque inicial de la aplicación
    - tecnologías *offline* (p.ej. AppCache), dividir scripts grandes (modular), almacenamiento local, minimizar el número de consultas al almacenamiento local
  - Minimizar la latencia percibida
    - mantener al usuario informado, evitar recarga de páginas, precargar las posibles siguientes páginas, …
  - Usar medidas relativas y porcentajes, **no** absolutas o en píxeles
  - Considerar tecnologías específicas para móviles a la hora de inicializar aplicaciones Web
- Considerar variaciones en el contexto de entrega
- Considerar el uso del elemento `Canvas` o gráficos SVG
- **- - -**