Programación de Móviles

Publicación 1.0

Oscar Gomez

26 de September de 2014

Índice general

Índice:

CAPÍTULO 1

Análisis de tecnologías para aplicaciones en dispositivos móviles:

1.1 Puntos iniciales

- En primer lugar, en http://10.8.0.253 se puede encontrar un servidor.
- Los apuntes también se podrán encontrar a diario en http://oscarmaestre.github.io
- Existen fotocopias con la programación, criterios, etc... en la mesa del profesor. En cualquier caso, están colgadas en la página del centro http://www.iesmaestredecalatrava.es (buscar el apartado "Presentaciones"
- Si se desea acceder a algún fichero individual de los apuntes puede hacerse en las página siguiente
  - https://github.com/OscarMaestre/Moviles
  - https://github.com/OscarMaestre/ServiciosYProcesos

1.2 Sistemas operativos para dispositivos móviles. Características.

El desarrollo para la telefonía móvil es un campo que se encuentra en plena expansión. El número de teléfonos no deja de crecer y las necesidades de programación de los mismos tampoco. En ese sentido existen diversas plataformas de desarrollo a tener en cuenta al empezar a programar.

- Android: es el más numeroso de lejos. La mayor parte del mercado usa esta plataforma. El hecho de que Google ofrezca *completamente gratis* el sistema operativo para los fabricantes y el entorno para los programadores lo ha hecho crecer hasta desbancar a su competidor. Google solo se ocupa de la venta de apps y ese es su nicho de beneficios.
- iOS: Utiliza una filosofía completamente distinta que es controlar todo el proceso desde el desarrollo hasta la distribución de aplicaciones. Ese control se hace por medio del pago de licencias y de la unicidad de la distribución.

- Windows Phone: es el sistema de Microsoft que ha lanzado más o menos recientemente y que aún está por ver si consigue una cuota de mercado significativa o no.
- FirefoxOS: se centra principalmente en mercados emergentes (con el objetivo a largo plazo de ganar cuota de mercado) su filosofía es la misma del software libre.
- Bada: Es la plataforma de Samsung creada exclusivamente para sus teléfonos. Es muy poco usada fuera de Corea del Sur.
- Symbian: surge de un antiguo sistema creado para las PDA, está prácticamente descatalogado en el desarrollo para telefonía móvil.
- Tizen: una plataforma relativamente nueva. Al igual que Android es gratis y además tiene el respaldo de Intel.
- Jolla: surge en los países nórdicos con una filosofía similar al software libre pero con la salvedad de que, de momento, solo se ejecuta en sus teléfonos (que son de gama alta)

En cuanto a la tecnología hay diferencias sustanciales entre ellas:

- Android: pensado principalmente para ser programado en Java (aunque se puede llegar a usar C++ con un kit aparte).
- iOS: usa Objective-C que lo separa mucho del resto de plataformas. El entorno exige el pago de una licencia, el SO exige una licencia y el poner aplicaciones a la venta exige otra.
- Windows Phone: usa Visual Studio que es una herramienta muy potente y usa la plataforma .NET.
- FirefoxOS: usa HTML y Javascript.
- Symbian y otros también permiten el uso de HTML y JS.
- Tizen permite dos opciones: C++ o HTML/JS
- Jolla/Sailfish: usa C++.
- Bada usa C++.

En este curso se usara Android con Java como lenguaje de desarrollo.

1.3 Limitaciones

Programar un teléfono móvil implica preparar nuestro programa para situaciones que no son de importancia en los ordenadores de escritorio o que incluso no existen.

- Desconexión: un teléfono puede perder el fluido eléctrico sin aviso o perder la conexión de red de forma repentina.
- Seguridad: un teléfono puede ser accesible desde cualquier punto del planeta lo que puede poner en grave riesgo la privacidad del usuario.
- Memoria: la cantidad de memoria de estos dispositivos es mucho más reducida que los equipos de escritorio.

- Consumo batería: la cantidad de código que se ejecuta implica disminuir la cantidad de batería del usuario.
- Almacenamiento: la cantidad de espacio para almacenar ficheros en estos dispositivos es muy variable y a veces prácticamente inexistente.

1.4 Entornos integrados de trabajo.

En este apartado vamos a hablar de los distintos entornos que se pueden usar para programar teléfonos móviles.

- Eclipse.
- XCode para iOS.
- Android Studio.
- NetBeans.
- Visual Studio (para Windows Phone).
- ¿Visual Kaffe?
- Línea de comandos.
- Appcelerator.

1.5 Android Studio

Android Studio es el entorno oficial de programación de Google. Como tal, es probablemente la herramienta del futuro si bien tiene diversas desventajas.

- Aún está en fase beta y según los documentos de instalación aún puede cambiar mucho sin previo aviso.
- Requiere una máquina más potente que los otros: de no ser así el proceso de edición-compilación-ejecución se vuelve demasiado lento.

1.6 Eclipse

Eclipse ha sido desde los comienzos la herramienta ofrecida por Google para programar en Android. De hecho, en su página se ofrece el "Android Developer Tools Bundle" que contiene absolutamente todo lo necesario para trabajar.

En este manual será la herramienta que se utilizará para los ejemplos.

1.7 La línea de comandos

La línea de comandos es el entorno más ligero. Además ofrece grandes ventajas en cuanto a la automatización de tareas, y de hecho Google ofrece el kit de desarrollo adaptado a la línea de comandos. El inconveniente principal es que algunos desarrolladores no están muy acostumbrados a ella.

1.8 El primer proyecto

Cuando se instala el Android Developer Bundle y se arranca Eclipse podremos utilizar un pequeño asistente para crear la primera aplicación. Para ello, en el menú `File-New` elegiremos la opción `Android Application Project`, mostrándonos una ventana que debería ser parecida a la siguiente figura.

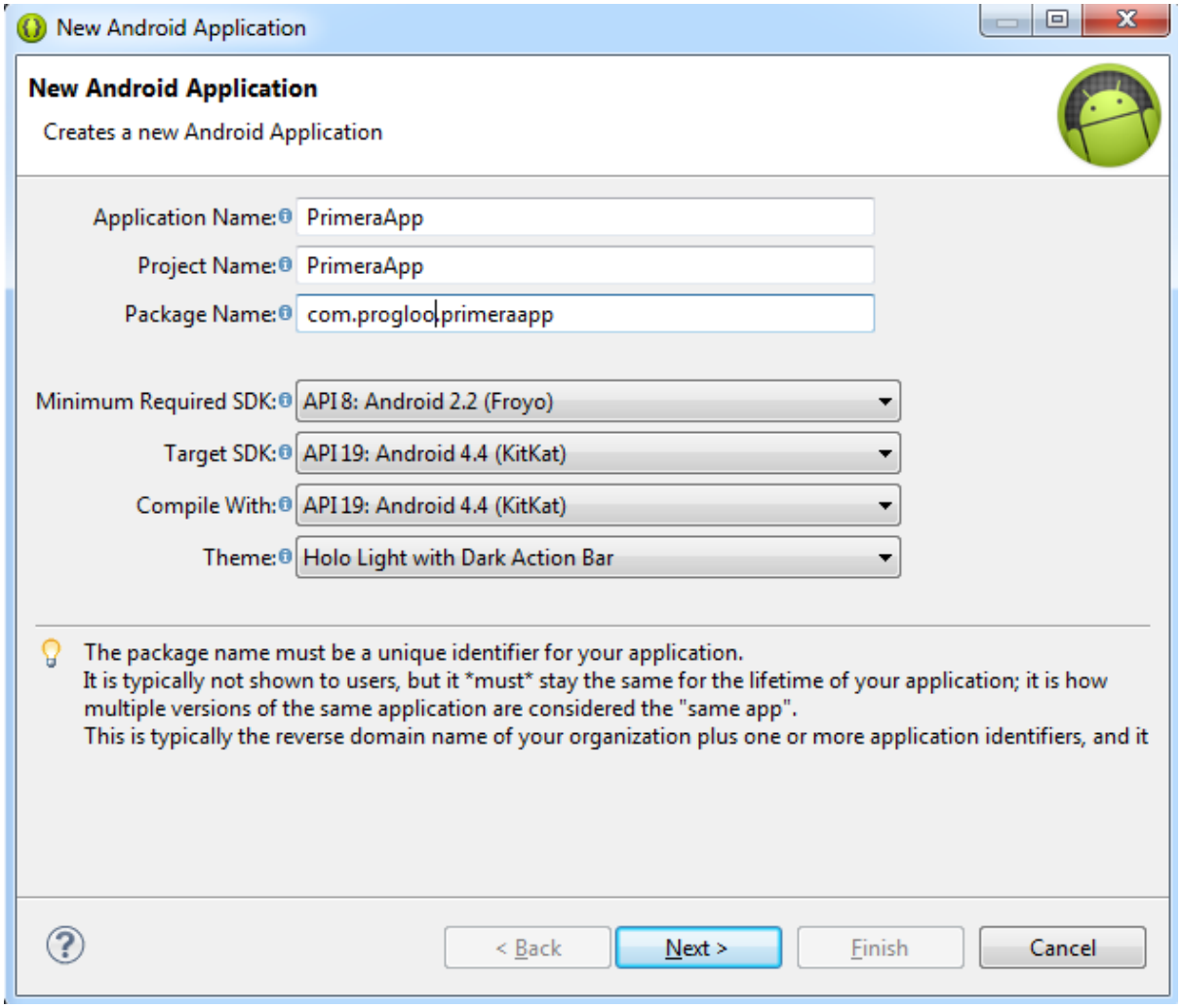

Figura 1.1: Datos iniciales de la aplicación Android

En ella deberemos prestar atención a los siguientes elementos:

- *Minimum required SDK* : es la versión de Android mínima que necesitará en su móvil/tablet quien desee instalar la aplicación. Si se tiene la tentación de poner la versión 1.0 se debe tener en cuenta que también se dispondrán de menos clases y métodos para construir la app. La versión 8 (Android 2.1) es un valor razonable a día de hoy.
- *Target SDK* : es la versión de Android para la cual hemos optimizado la aplicación. En todo este manual se usará la versión 19 de Android (o Android 4.4)
- *Compile with* : Android tiene varias versiones y podemos utilizar una versión posterior para optimizar una aplicación orientada a un Android más antiguo. Sin embargo, normalmente no lo haremos y usaremos la misma versión que en el Target SDK, es decir, la 19.
- *Theme* : las aplicaciones pueden tener diversos temas o "skins". Google ofrece algunos estilos predeterminados, pero no haremos especial hincapie en el diseño, solo en la programación. Usaremos el estilo por defecto "Holo Light".

Despues de haber rellenado estos datos podremos ver algo como esto:

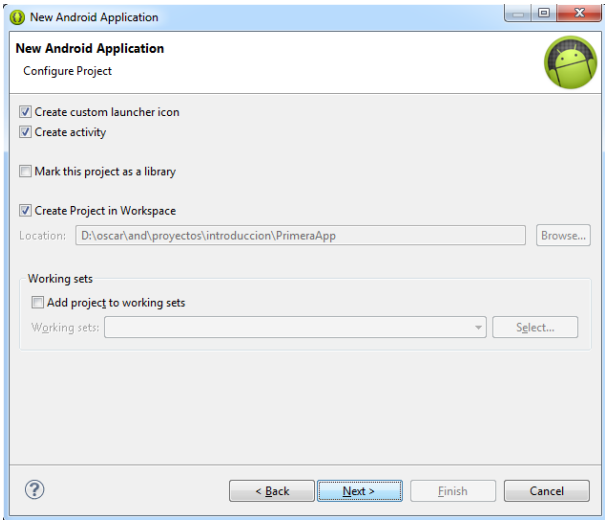

Figura 1.2: Opciones específicas del proyecto

Aquí podremos indicar si queremos crear una biblioteca en lugar de una aplicación, si deseamos que se cree una actividad en blanco y si queremos ponerlo en el directorio de trabajo predeterminado. Se dejarán las opciones por defecto.

1.9 Descargando plataformas

Una vez hecho esto se debería instalar alguna versión del kit de desarrollo Android para empezar a programar. Para ello, se debe arrancar el gestor de plataformas Android mediante el menú de Eclipse `Window->SDK Manager`

El SDK Manager hace unas cuantas recomendaciones bastante prácticas: normalmente intentará instalar la última versión de Android más algunas herramientas útiles.

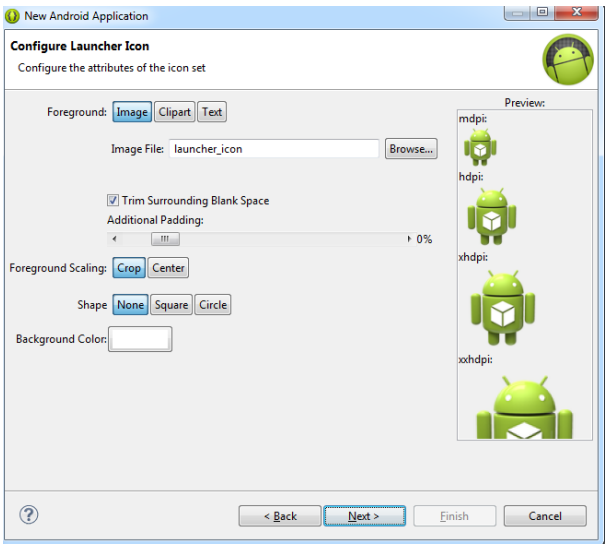

Figura 1.3: Personalizando el icono

Esta ventana permite elegir algunas opciones sobre el icono de la aplicación:

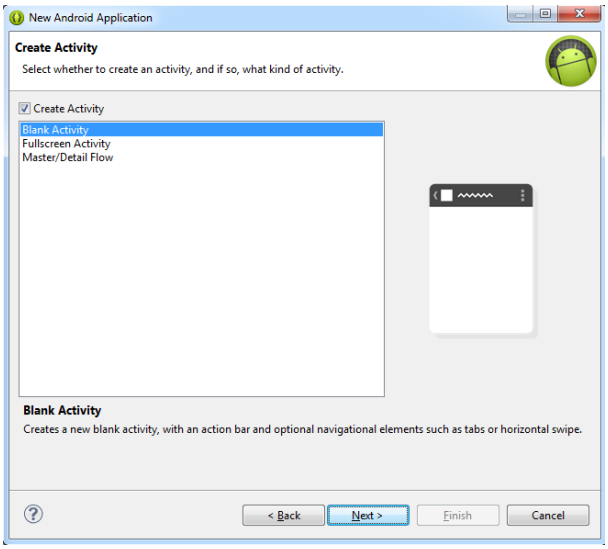

Figura 1.4: Tipo de actividad

Aquí se puede elegir que tipo de actividad se desea. En general, usaremos una actividad en blanco.

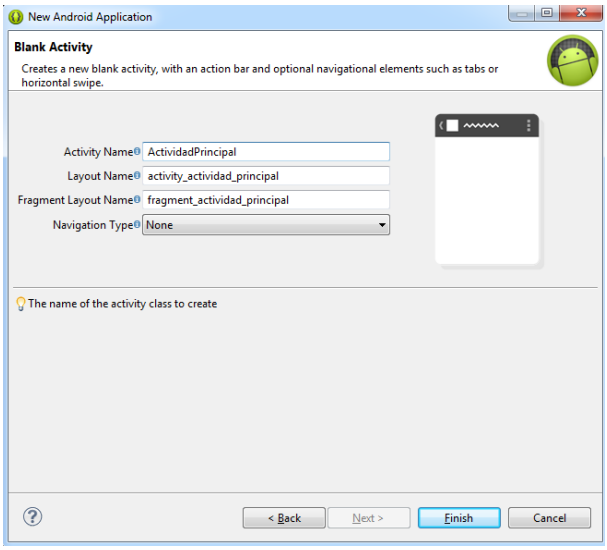

Figura 1.5: Datos de la actividad

En este último paso se indicará el nombre de la clase que contendrá la actividad principal de la aplicación. Usaremos el nombre `ActividadPrincipal` y terminaremos el asistente.

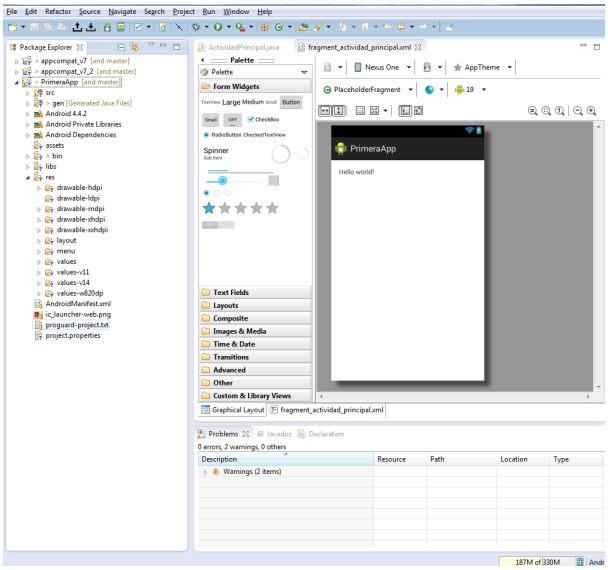

Figura 1.6: Un proyecto vacío de Android

El asistente terminará y se nos mostrará el entorno de Eclipse.

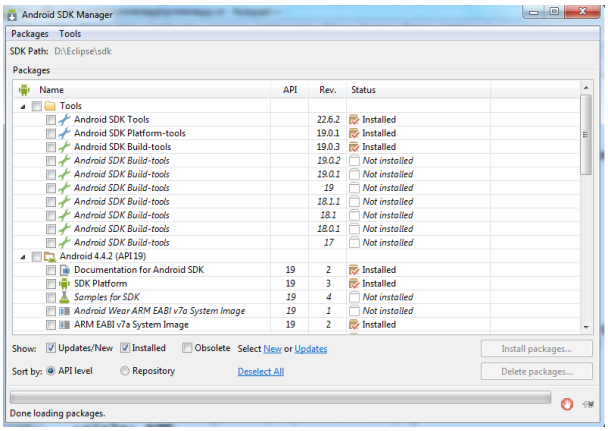

Figura 1.7: Administrador de plataformas Android

Advertencia: Una de las herramientas que se descargará es *Intel x86 Emulator Accelerator* o HAXM. Esta herramienta de Intel permite acelerar la ejecución del emulador de Android en microprocesadores Intel que tengan activada en su BIOS la opción de aceleración. Puede ser necesario habilitar esta opción en la BIOS (probablemente en alguna opción con el nombre `Enable Intel VT-x` o similar). El uso de HAXM es **MUY RECOMENDABLE**. Por otro lado, el SDK Manager descarga, pero no descomprime ni instala HAXM. Se debe buscar el ZIP en el directorio de instalación y ejecutarlo.

En líneas generales se necesitarán:

- Todos los archivos de la última plataforma
- El driver USB, que permitirá ejecutar nuestros programas en un móvil/tablet conectado por USB al equipo
- El driver HAXM
- La biblioteca de soporte de Android: permite que programas con una versión moderna se ejecuten en algunas plataformas más antiguas, entre otras cosas.
- Las *build-tools* o herramientas de compilación.
- Las *platform-tools* o herramientas específicas de la plataforma.
- Las *Android tools*, herramientas específicas de Android

1.10 Creando emuladores

Cuando se haya completado el paso anterior, se podrán crear *Android Virtual Devices* o AVDs o emuladores. Se pueden crear dispositivos con diferentes características como se muestra a continuación.

En primer lugar, se debe elegir la opción `Window-Android Virtual Device Manager`, con lo que ser verá una herramienta que permite crear emuladores.

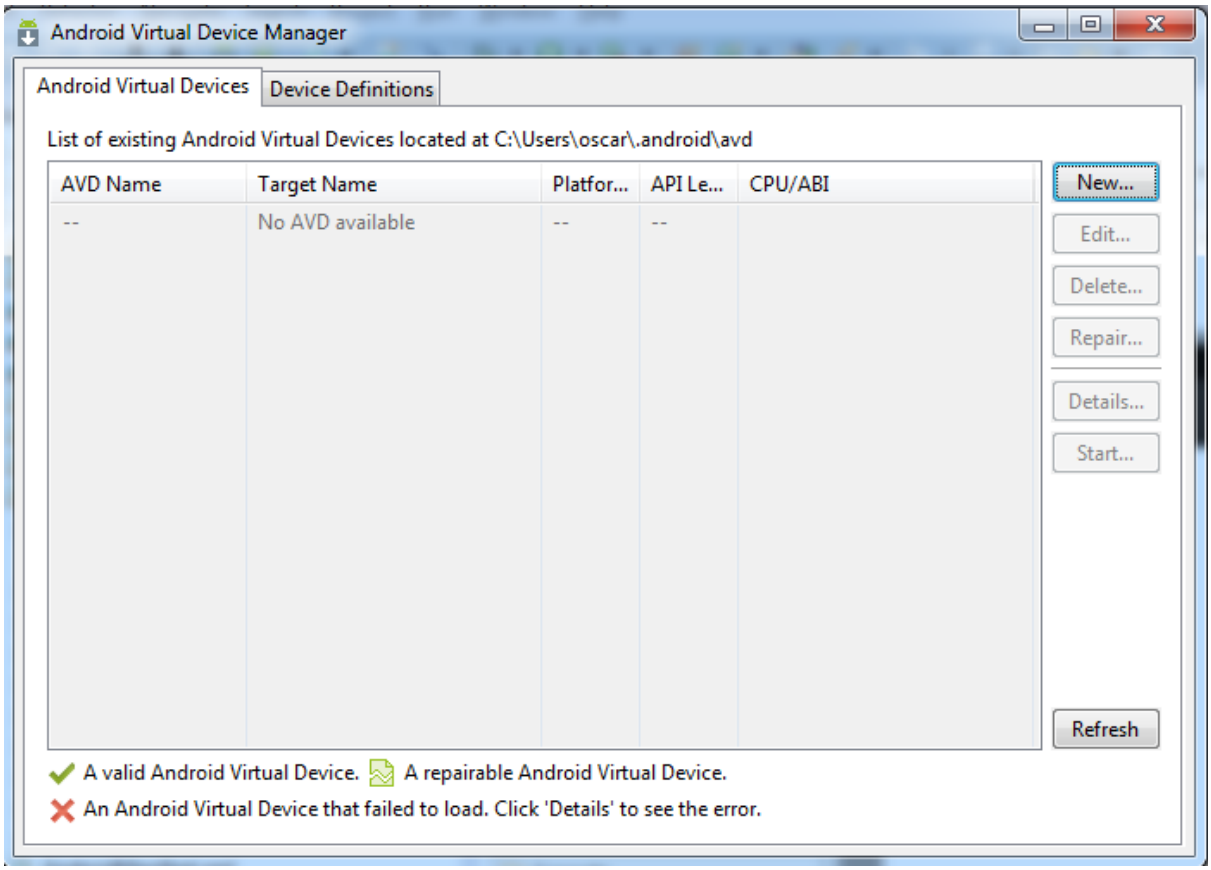

Figura 1.8: El Android Virtual Device Manager

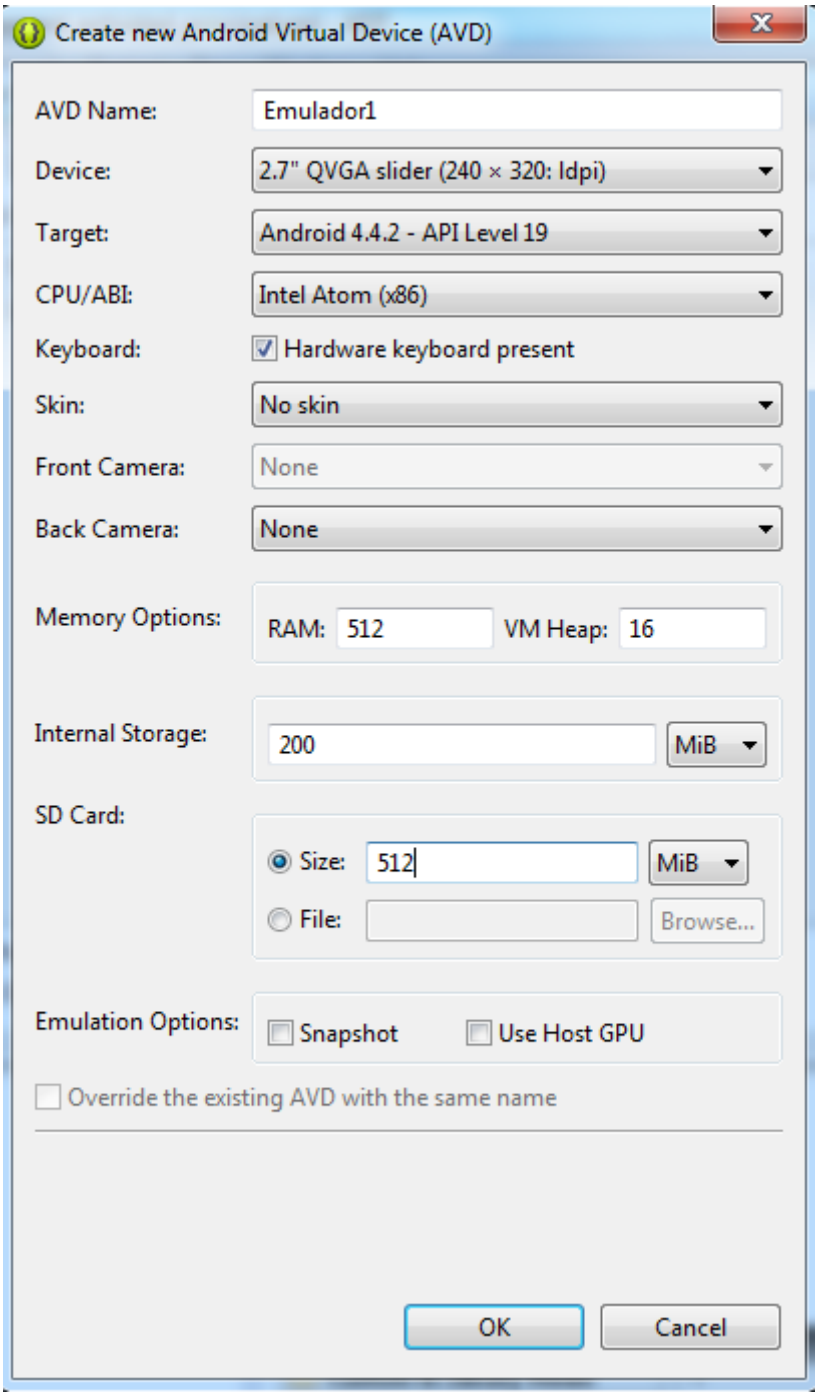

Figura 1.9: Creando dispositivos

Una vez arrancado se podrá crear un nuevo dispositivo con el botón *New*. Se recomienda mantener estas opciones.

1.11 Arrancando el programa

Una vez que se tiene el emulador creado, se puede arrancar con el botón Start, y despues arrancar el proyecto vacío Android de Eclipse. Para ello, una posibilidad es hacer click con el botón derecho en el proyecto que vemos a la izquierda de Eclipse y elegir el menu `Run As-Android Application`. Debería arrancarse la aplicación en el emulador y ver el resultado.

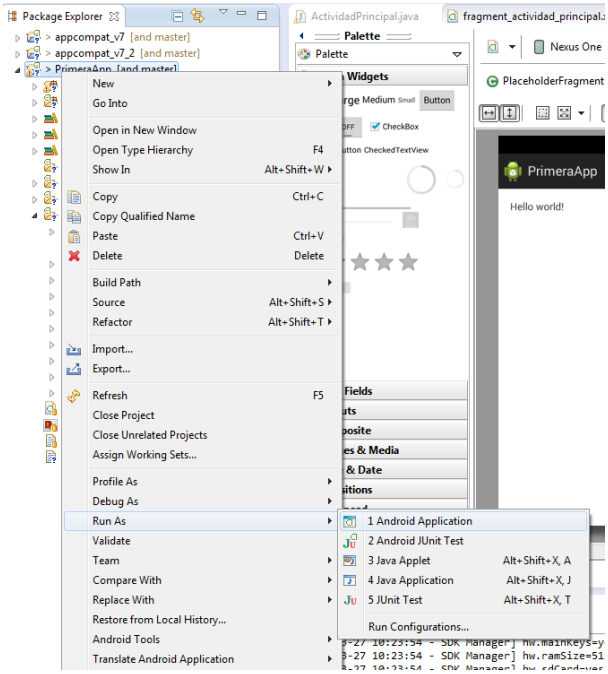

Figura 1.10: Ejecutando el primer proyecto

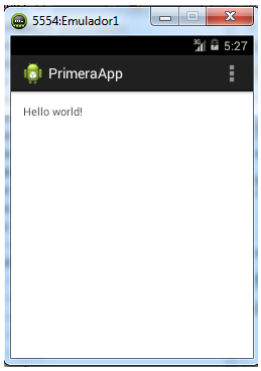

Figura 1.11: Emulador ejecutando la primera app

1.12 Módulos para el desarrollo de aplicaciones móviles.

En este curso, realmente solo necesitaremos Java para crear apps. Sin embargo, existen un montón de bibliotecas que permiten acelerar el desarrollo.

- Unity para desarrollar juegos.
- JQuery para Javascript.
- Bibliotecas para tareas muy específicas como la seguridad SSL y similares.

1.13 Emuladores.

A la hora de probar un app suele ser posible utilizar un emulador cargado en el sistema operativo que facilite la tarea de depurar la aplicación.

En Android, Google proporciona un sistema completo de emulación basado en máquinas virtuales (no usa VirtualBox sino un programa similar llamado QEMU).

El sistema de emulación permite crear dispositivos de características muy variadas para probar nuestra app en distintos entornos. Google denomina a estos dispositivos Android Virtual Devices (o AVDs)

- Se puede modificar el tamaño y la resolución.
- La memoria RAM y espacio en tarjeta SD.
- Se puede poner o quitar cámara.
- Existen dispositivos predefinidos por Google que permiten crear emuladores muy rápidamente.
- También se pueden clonar dispositivos para hacer solo una modificación de forma rápida.
- Una característica de interés es que *si se dispone de una tarjeta gráfica con aceleración* se puede activar una casilla llamada "Host GPU" que permite acelerar la emulación.
- Se puede obligar al emulador a que "recuerde" el estado en que se quedó para así continuar donde nos hubiésemos quedado el último día. Esta opción se llama instantánea o *snapshot*.

Si el equipo de escritorio es un Intel se puede instalar el Hardware Accelerated eXecution Manager o HAXM que permite acelerar la emulación. En el directorio `sdk/extras/intel` se puede encontrar un archivo ZIP que contiene un EXE que instala el HAXM. Se recomienda encarecidamente instalarlo en casa y, si es necesario, habilitar la tecnología VT en la BIOS.

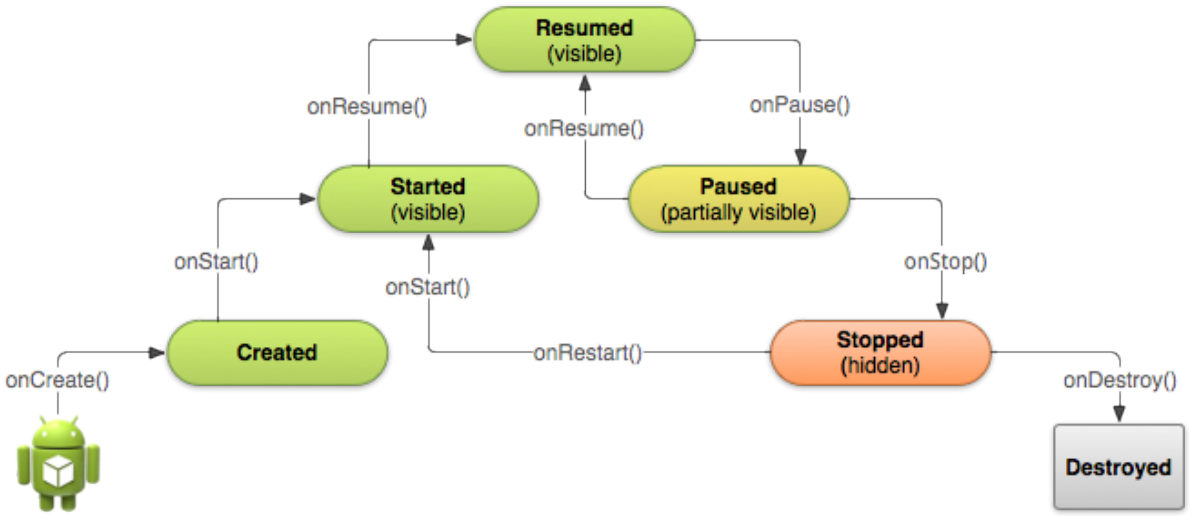

Figura 1.12: Pasos en la ejecución de una app (imagen tomada de Google).

1.14 Ciclo de vida

1.15 Configuraciones y perfiles

1.16 Modificación de aplicaciones existentes.

1.17 Utilización del entorno de ejecución del administrador de aplicaciones.

CAPÍTULO 2

Programación de aplicaciones para dispositivos móviles

CAPÍTULO 3

Utilización de librerías multimedia integradas

CAPÍTULO 4

Análisis de motores de juegos

4.1 Animación 2D y 3D.

4.2 Arquitectura del juego. Componentes.

4.3 Motores de juegos: Tipos y utilización.

4.4 Áreas de especialización, librerías utilizadas y lenguajes de programación

4.5 Componentes de un motor de juegos.

4.6 Librerías que proporcionan las funciones básicas de un Motor 2D/3D.

4.7 APIs gráficos 3D.

4.8 Estudio de juegos existentes.

4.9 Aplicación de modificaciones sobre juegos existentes.

CAPÍTULO 5

Desarrollo de juegos 2D y 3D

5.1 Entornos de desarrollo para juegos.

5.2 Integración del motor de juegos en entornos de desarrollo.

5.3 Conceptos avanzados de programación 3D.

5.4 Fases de desarrollo:

5.5 Propiedades de los objetos: luz, texturas, reflejos, sombras.

5.6 Aplicación de las funciones del motor gráfico. Renderización.

5.7 Aplicación de las funciones del grafo de escena.

5.8 Tipos de nodos y su utilización.

5.9 Asociación de sonidos a los eventos del juego.

5.10 Análisis de ejecución. Optimización del código.

5.11 Documentación de la fase de diseño y de desarrollo.

CAPÍTULO 6

Sistemas basados en localización

6.1 Tecnologías de localización (GPS, A-GPS,...).

6.2 Servicios de localización, mapas y geocodificación.

6.3 Emuladores para simular las ubicaciones.

6.4 Visualización la información geolocalizada.