Unidad 1: Análisis de tecnologías para aplicaciones en dispositivos móviles.

Índice:

1. Introducción. ¿Qué es un dispositivo móvil?

Se trata de un aparato de pequeño tamaño (normalmente que quepa en un bolsillo) y de poco peso, con pantalla táctil y en algunos casos teclado, con pequeñas capacidades de procesamiento, memoria limitada y conexión (permanente o no) a una red. Este tipo de dispositivos están diseñados para cumplir algún tipo de función específica (realizar llamadas telefónicas, servir como agendas, jugar, navegación GPS, escuchar música, acceso al correo electrónico, navegar por Internet, etc.) aunque normalmente pueden llevar a cabo también funciones más generales.

Estos aparatos son cada vez más populares especialmente para aquellos entornos en los que llevar consigo un ordenador convencional (incluso un portátil) no es práctico.

1.1.- Evolución, características y tecnologías.

Hemos visto ya que un dispositivo móvil es un elemento portable con ciertas capacidades de procesamiento y de comunicación. Desde la aparición de los dispositivos móviles en el mercado, se ha producido una evolución, o mejor dicho, una revolución de sus características y prestaciones.

Vamos a realizar ahora un repaso de las tecnologías móviles más destacadas desde el inicio de la aparición de estos dispositivos, y las características principales que representan la evolución desde los teléfonos convencionales celulares, hasta los ordenadores de bolsillo con conectividad ilimitada de la actualidad.

TECNOLOGÍAS MÓVILES Y EVOLUCIÓN

SMS

Los primeros teléfonos móviles del mercado solamente permitían la funcionalidad de llamada. En la mayoría de ellos aún no se incluía la posibilidad de envío/recepción de SMS, debido a lo cual se proporcionó como servicio gratuito hasta el momento en que fue implantado de forma masiva en los terminales y empleado por el gran público llegando a ocupar unas cuotas de utilización en torno al 20% con respecto a los ingresos totales de los operadores de telefonía.

WAP

Un siguiente paso para los teléfonos móviles de segunda generación consistió en incluir la posibilidad de navegación. No se puede hablar en estos terminales aún de Internet como tal, ya que los navegadores disponibles en los mismos funcionaban con un protocolo muy inferior a HTTP denominado WAP. Las páginas accesibles por los navegadores WAP estaban escritas en el estándar WML que permitía simplemente la inclusión de texto simple, imágenes e hipervínculos a otras páginas, que representa una

solución muy primitiva y restringida no solamente debido a las capacidades gráficas disponibles en esos móviles, sino también como respuesta a un ancho de banda aún muy pobre.

GPRS

La denominada generación 2.5 comienza con la aparición de la tecnología GPRS, que permite unos anchos de banda mucho mayores a GSM, pero que vio truncada su proyección debido a los altos costes impuestos por los operadores con tarificaciones en función del volumen de datos descargados. En este entorno aparece la versión 2 de WAP basada en XHTML, que es capaz de procesar páginas webs simples. En este entorno, además de los nuevos y mejorados modelos de móviles, irrumpen en el mercado los dispositivos denominados PDA (Personal Digital Assistant - asistentes digitales personales) o PocketPC (PC de bolsillo), que hasta el momento se habían orientado al uso industrial, pero que se adaptaron al usuario final proporcionando mejoras en los aspectos gráficos y de interfaz a través de pantallas táctiles, de navegación por Internet, y, en conjunto, del uso del terminal a través de sistemas operativos cómodos e intuitivos como Windows Mobile como dominante en el mercado europeo.

3G/4G

Después de éstos inicios llega Internet móvil, que permite que un terminal inalámbrico se pueda conectar a cualquier recurso de la misma forma que cualquier otro tipo de equipo de la red, y además a velocidades similares. Todo ello unido a la disponibilidad de tarifas planas de navegación ofrecidas por los operadores y, por supuesto, a la evolución de los teléfonos móviles hacia Smartphones donde su amplia funcionalidad no puede ser concebida sin conexión, ya sea a través del operador o a través de WiFi, que configuran el entorno actual donde los dispositivos móviles representan la conexión, y ésta facilita más que nunca las relaciones entre las personas.

5G

5G es la nueva generación de tecnología de redes móviles, específicamente la quinta. Esta versión mejora ampliamente las prestaciones de acceso a Internet en movilidad como el ancho de banda, la capacidad de dispositivos conectados y la latencia, comparado con las generaciones anteriores (1G, 2G, 3G y 4G).

2.- Limitaciones de las tecnologías móviles.

Antes de comenzar a desarrollar software para alguno de estos dispositivos, es necesario ser conscientes de las limitaciones con las que nos podemos encontrar en los mismos. ¿Cuáles son las restricciones a las que nos vamos a tener que enfrentar?

Además de las restricciones de los interfaces de entrada y salida (teclado táctil/pantallas de tamaño reducido) y en cuanto a capacidad de procesamiento, principalmente se pueden mencionar las siguientes:

Desconexión:

Los dispositivos móviles no siempre se encuentran conectados a Internet, incluso aun teniendo capacidad para ello es posible que existan pérdidas de cobertura, conexiones intermitentes o con ancho de banda limitado.

Seguridad:

Aunque poseen capacidades de seguridad de conexiones, cifrado de datos, y certificados digitales, su seguridad se puede ver comprometida debido a la facilidad de pérdida o robo del dispositivo portable. Además, es necesario tomar precauciones con la descarga y permisos de aplicaciones debido a que un uso malicioso puede disparar el consumo de la factura telefónica.

Memoria:

Se trata de dispositivos con capacidades reducidas de memoria para ejecución de aplicaciones, así como para la carga de imágenes o archivos de gran tamaño.

Consumo batería:

El suministro de energía es limitado, y dependiente de la batería del dispositivo, cuyo consumo se hace muy exhaustivo debido al uso prolongado de la pantalla y de la conexión de datos.

Almacenamiento:

Almacenamiento de datos persistente reducido (pequeña memoria flash interna, tarjetas SD, etc.).

Este tipo de restricciones, y algunas otras que dependerán de cada dispositivo en concreto, habrán de ser tenidas muy en cuenta a la hora del análisis y diseño de una aplicación "móvil", pues no podemos pretender, que esa aplicación pueda contener la misma funcionalidad, que la que podemos encontrar habitualmente en un programa que es ejecutado en un ordenador de sobremesa o un portátil.

Por otro lado, no todo va a ser restricciones. También habrá que tener en consideración que esta tecnología va a aportar una serie de ventajas muy importantes: movilidad, poco peso, pequeño tamaño, facilidad para el transporte, conectividad a diversos tipos de redes de comunicaciones (mensajería; voz; Internet; Bluetooth; etc.). Ésas serán las ventajas que podrás explotar en tus aplicaciones.

3.- Tecnologías disponibles.

Cuando vas a desarrollar una aplicación para un dispositivo móvil, algunas de las primeras preguntas que te puedes hacer son:

¿Sobre qué tipos de dispositivos móviles se pueden hacer programas? ¿Sobre qué tipo de hardware se puede programar?

¿Qué sistema operativo puede llevar ese hardware?

¿Qué plataformas de desarrollo existen para desarrollar sobre ese hardware y ese sistema operativo? ¿con qué lenguajes puedo programar? ¿qué herramientas (compiladores, bibliotecas, entornos, etc.) hay disponibles?

Las respuestas a este tipo de preguntas pueden ser múltiples y muy variadas:

Hardware

Respecto al hardware, te puedes encontrar, como has visto ya, con smartphones y tablets de distintos fabricantes: Samsung, HTC, Apple, LG, Sony Ericsson, etc. Cada fabricante produce móviles de distintas gamas y características, pero la idea es crear un software que se pueda ejecutar en el mayor número posible de dispositivos. La gran ventaja que poseemos en este terreno es que no necesitamos conocer detalles del hardware del teléfono ya que las aplicaciones se ejecutan sobre la capa del sistema operativo.

Sistemas Operativos:

En cuanto a los sistemas operativos, comentar que en los inicios de la era móvil se desarrollaban específicamente para cada marca o incluso cada modelo. El sistema operativo Symbian OS surgió como una alternativa más universal que fue adoptado como sistema base de muchos fabricantes. Este mismo planteamiento fue el que posibilitó que Android ganara tanta popularidad entre fabricantes y consumidores, aunque existen otras alternativas muy potentes y conocidas como iOS, Blackberry OS y Windows Phone.

Plataformas de desarrollo:

Si lo que deseas es conocer algo acerca de las plataformas de desarrollo disponibles para cada entorno (hardware y/o sistema operativo), cada plataforma proporciona todo lo necesario para el desarrollo en su plataforma, ya que las apps móviles son un motor muy importante del modelo de negocio de esas compañías.

Lenguajes de programación

Si te refieres a lenguajes de programación, normalmente te encontrarás con lenguajes que son ya viejos conocidos para otras plataformas, como pueden ser las aplicaciones de escritorio para los PCs o las aplicaciones web (Java, C#, C, etc.).

En definitiva, puedes observar que en este nuevo mundo del desarrollo para dispositivos móviles te encuentras con una problemática similar a la que te puedes enfrentar con los ordenadores convencionales: distintos tipos de hardware, distintas opciones de sistemas operativos dependiendo del hardware que los soporte, diferentes lenguajes de programación, plataformas, API y bibliotecas, entornos de desarrollo, etc.

3.1.- Dispositivos.

Los dispositivos principales con los que vamos a trabajar son los Smartphones y las Tablets.

El término SmartPhone se acuñó comercialmente para los terminales de telefonía móvil que proporcionaban unas prestaciones y una funcionalidad mayor que la que podría ofrecer un teléfono móvil normal, sobre todo en cuanto a las características de conectividad a redes inalámbricas y la posibilidad de ejecutar aplicaciones desarrolladas bien por el propio fabricante del terminal, bien por el operador de telefonía móvil, o bien por un tercero (empresa de desarrollo de software).

Algunas otras características que suelen tener este tipo de dispositivos son:

- Funcionamiento en multitarea (ejecución concurrente de varios procesos en el sistema operativo).
- Acceso a Internet.
- Conectividad Wi-Fi, Bluetooth, etc.
- Posibilidad de conexión con un ordenador para cargar y descargar información (normalmente con conexión USB o bien una conexión inalámbrica).
- Posibilidad de ampliación de memoria mediante tarjetas externas de memoria (por ejemplo SDCard).
- Pequeñas pantallas pero de alta resolución y/o con millones de colores.
- Pantallas táctiles o incluso multitáctiles (multitouch).

- Sensores (de orientación, de temperatura, de presión, acelerómetros, magnetómetros, etc.).
- Cámaras digitales integradas. Capacidades fotográficas. Grabación de audio y vídeo.
- Receptor GPS.

Puedes observar que aunque el dispositivo es un teléfono móvil, muchas veces su uso principal no va a ser necesariamente el de un teléfono (hacer y recibir llamadas), sino que podrá estar dedicado a muchos otros usos (hacer fotos, navegar por Internet, reproducir archivos de audio, jugar, gestionar la agenda personal, consultar un mapa, usar un diccionario, escuchar la radio, ver una película, trazar una ruta para el navegador por satélite, etc.).

Una Tablet es un dispositivo móvil de mayor tamaño y capacidades, ideado para cumplir con todas las funcionalidades comentadas anteriormente, y que no requiere obligatoriamente de conexión telefónica debido a que su ámbito de uso es más doméstico, donde se presume la capacidad de conexión vía Wi-Fi. A todos los efectos consideraremos las Tablets como versiones superiores de los Smartphones por lo que no se realizará distinción alguna entre ambos tipos de dispositivo a lo largo de los capítulos del curso, excepto en aquellos apartados que sea necesario distinguir algún aspecto relacionado.

3.2.- Sistemas operativos.

Los sistemas operativos más habituales que te puedes encontrar en un dispositivo móvil son:

Android.

Desarrollado inicialmente por Google y basado en el núcleo de Linux. El primer fabricante de móviles que lo incorporó fue HTC.

Android es un sistema operativo completo, libre y abierto basado en Linux y orientado a dispositivos móviles.

Android fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005, y su desarrollo fue continuado por éste y la Open Handset Alliance, un grupo de compañías de tecnologías móviles incluyendo fabricantes, desarrolladores de hardware, software y operadores de telefonía, con el objetivo de acelerar la innovación en movilidad y ofrecer al consumidor una más rica, más barata y mejor experiencia en este tipo de dispositivos.

Al tratarse de una plataforma abierta con kits de desarrollo gratuitos sobre herramientas comunes de desarrollo y utilizando el lenguaje de programación Java, Android dispone de una gran comunidad de desarrolladores creando aplicaciones que permiten extender la funcionalidad de los dispositivos.

Las aplicaciones Android se desarrollan con las APIs Java SE, y se ejecutan sobre una máquina virtual Dalvik. Esta máquina virtual está optimizada para requerir poca memoria y diseñada para permitir ejecutar varias instancias simultáneamente, delegando en Android el soporte de aislamiento de procesos, gestión de memoria e hilos. No se trata de una máquina virtual Java, ya que no opera con bytecodes sino con archivos dex propios.

iOS.

Desarrollado por Apple para el iPhone y usado más tarde también para el iPod Touch y el iPad.

El Sistema Operativo iOS es el sistema operativo móvil de Apple desarrollado originalmente para el iPhone, siendo después usado en otros dispositivos Apple como iPod Touch e iPad.

Se trata de un derivado de Mac OS X, denominado iPhone OS en sus inicios y orientado a la arquitectura ARM.

Existe un SDK integrado en el entorno de desarrollo gratuito de Apple XCode, que permite el desarrollo de aplicaciones utilizando los lenguajes de programación Objective-C o Swift así como la realización de pruebas en el iPhone Simulator. Para poder realizar pruebas en un dispositivo real es necesario suscribirse al iPhone Developer Program y pagar la cuota correspondiente. Este hecho no detuvo a la comunidad mundial de desarrolladores, cuyo amor por este dispositivo provocó una amplia proliferación tanto de terminales como de aplicaciones desarrolladas que solamente se vio truncada por la aparición de Android.

La arquitectura iOS está basada en capas de abstracción, donde las capas más altas contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas más bajas controlan los servicios básicos. La capa de más alto nivel es Cocoa Touch, que contiene los frameworks para desarrollo de aplicaciones UIKit con las clases para interfaz de usuario y Foundation Framework con las clases básicas y de acceso a servicios del sistema operativo. Por debajo de ella se encuentran la capa Media de servicios gráficos y la capa Core Services de servicios comunes a las

aplicaciones. Por último, la capa Core OS contiene todas las características de bajo nivel como acceso a ficheros y memoria.

Windows Phone.

Desarrollado por Microsoft tanto para smartPhones como para otros dispositivos móviles. Gana en popularidad en el mundo móvil cuando Nokia lo incorpora en su serie Lumia

Windows Phone es el sistema operativo móvil desarrollado por Microsoft para su uso en Smartphones y otros dispositivos móviles.

La versión anterior del sistema operativo fue denominada Windows Mobile, sucesor del Windows CE como sistema operativo destinado a dispositivos móviles con capacidades limitadas como las PDA Pocket PC.

El entorno de desarrollo para Windows Phone es el Microsoft Visual Studio. Existe una versión gratuita para el desarrollo de aplicaciones móviles que incluye los emuladores y una versión Express del IDE, aunque para publicar aplicaciones se requiere una subscripción al Marketplace. Cada vez están proliferando más aplicaciones para Windows Phone, promovido en parte por la alianza estratégica de Microsoft y Nokia que suma adeptos del fabricante líder de los últimos tiempos a la plataforma Windows para dispositivos móviles.

Windows Phone está basado en el núcleo de Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows. Su diseño visual se asemeja al de Windows, proporcionando algunas pantallas con interfaz natural de usuario. Las aplicaciones que corren en este entorno pueden implementarse a través de varias tecnologías: Silverlight para aplicaciones con efectos visuales, .NET para la implementación de aplicaciones convencionales, así como el framework XNA para el desarrollo de videojuegos.

Blackberry OS.

Desarrollado por Research in Motion (RIM) para sus dispositivos Blackberry, con un uso más restringido al ámbito empresarial en la actualidad.

3.3.- Plataformas de desarrollo y lenguajes de programación.

Para poder desarrollar aplicaciones para alguno de los anteriores sistemas operativos es necesario disponer de alguna plataforma de desarrollo que permita generar código ejecutable sobre esos sistemas, o bien sobre alguna máquina virtual, o algún intérprete que esté instalado en el dispositivo y que sea soportado por el sistema operativo.

Existen principalmente dos tipos de alternativas para el desarrollo de aplicaciones:

Aplicaciones nativas:

Las aplicaciones nativas se ejecutan directamente sobre el sistema operativo subyacente, por lo que presentan comportamiento y características óptimas y adaptadas a cada sistema.

La principal desventaja es que no existe ningún tipo de compatibilidad entre plataformas por lo que una app que quiera ejecutarse en todas ellas requiere la construcción de apps independientes, con lenguajes, filosofías de desarrollo y características distintas, normalmente implementadas por distintos equipos de trabajo.

Aplicaciones multiplataforma:

Una aplicación multiplataforma, como su nombre indica, presenta como principal ventaja la posibilidad de ejecución en los distintos sistemas operativos. Por contra, su ejecución suele ser más lenta debido a que se encuentran basadas en motores que interpretan el contenido, que habitualmente se encuentra generado en HTML/CSS/Javascript.

Entre las plataformas de desarrollo nativas más populares se encuentran las que los propios autores de los sistemas operativos ofrecen para trabajar sobre su plataforma. De hecho, ya hemos hablado algo sobre algunas de ellas al describir los sistemas operativos. Hacemos un repaso rápido de las más conocidas:

- Windows Phone. Para desarrollar aplicaciones sobre Windows Phone pueden utilizarse las tecnologías Silverlight, XNA y .NET de Microsoft. Las herramientas que proporciona Visual Studio Express para Windows Phone son ofrecidas gratuitamente por Microsoft. El lenguaje más habitual para el desarrollo es C#, aunque la idea es que se pueda utilizar también otros lenguajes de la plataforma .NET como Visual Basic .NET.
- Android. Google proporciona también de manera gratuita el Android SDK para programar aplicaciones en Java sobre su sistema operativo. El IDE oficial es Android Studio aunque en los inicios lo más utilizado fue Eclipse con el plugin ADT (Android Development Tools). También es posible programar en otros lenguajes como C o C++ gracias al uso del Android NDK (Native Development Kit) que permite generar código nativo (native code), frente al código gestionado (managed code), que se ejecuta sobre una máquina virtual, como es el caso de Java.
- iOS. El SDK también se puede descargar gratis, pero para poder comercializar el software a través de su tienda de aplicaciones hay que registrarse en el programa de desarrollo del iPhone, y renovarlo anualmente. El lenguaje de

programación mayoritario en este caso es Objective-C, aunque en los últimos tiempos Swift ha surgido como una alternativa más sencilla para trabajar con esta plataforma.

- Java ME: Este caso sería una excepción respecto a los demás, puesto que no se trata de una serie de herramientas (bibliotecas, compiladores, IDEs, etc.) asociadas a un sistema operativo, sino de un subconjunto de la plataforma Java orientada para el desarrollo sobre dispositivos móviles. Se programaría en lenguaje Java y sería necesario que hubiera instalada una máquina virtual en el sistema operativo del dispositivo sobre el que se deseara ejecutar la aplicación desarrollada como sucede por ejemplo en sistemas como Symbian o Blackberry OS.

4.- Desarrollo de aplicaciones para dispositivos móviles.

En nuestro caso, Java puede ser una buena elección dado que es el lenguaje con el que aprendiste a programar en el módulo de Programación y por tanto no tendrás que aprender un nuevo lenguaje y podrías dedicarte de lleno al aprendizaje de las características específicas para el desarrollo de aplicaciones para dispositivos móviles. Por otro lado, un lenguaje multiplataforma como Java es también muy adecuado para un ciclo con un nombre como "Desarrollo de Aplicaciones Multiplataforma". Pero en cualquier caso no debes olvidar que se trata de una de las muchas opciones con las que te puedes encontrar en el mercado, y que estas alternativas irán apareciendo y desapareciendo constantemente en función del éxito que vayan obteniendo.

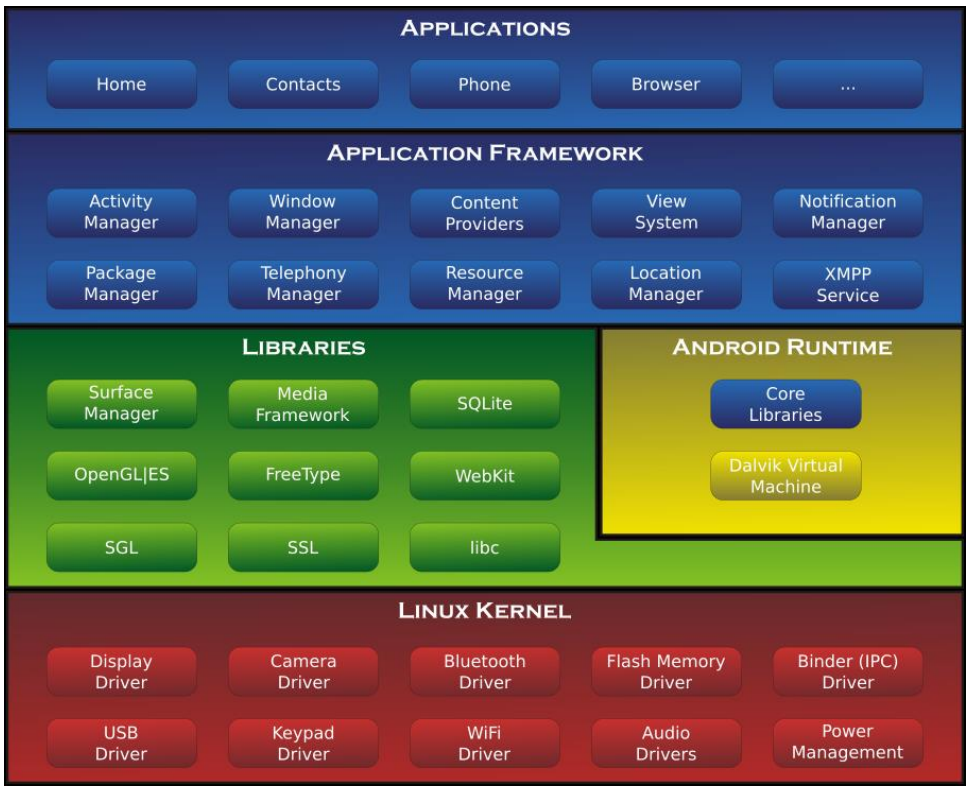
Respecto a la plataforma, Android posee su propio SDK para el desarrollo de aplicaciones que cuenta con ciertas características que debemos aprender para poder iniciarnos en el desarrollo de aplicaciones móviles en este sistema operativo.

En cuanto al entorno de desarrollo, se va a emplear Android Studio ya que se trata del IDE oficial de Google. Aunque también existe la posibilidad de emplear Eclipse junto con el plugin ADT, que citamos anteriormente.

Recapitulando, para facilitarte el aprendizaje vamos a trabajar con:

4.1.- Módulos para el desarrollo de aplicaciones móviles.

Para poder desarrollar con Android, es imprescindible disponer del kit de desarrollo de Android (Android SDK). Este conjunto de módulos incluye todo lo necesario para el desarrollo de aplicaciones destinadas a Smartphones, Tablets y otros dispositivos, y se encuentra incluido dentro del entorno Android Studio, sobre el cual hablaremos ampliamente en el siguiente capítulo.



El SDK incluye las herramientas necesarias para el desarrollo, como son los compiladores o los emuladores, así como la documentación completa y ejemplos de aplicaciones Android. El SDK Manager se encarga de mantener actualizado el entorno

de desarrollo, presentando la gran ventaja de poder descargar la última versión de módulos de programación disponibles para cada versión del sistema operativo directamente desde los servidores de desarrollo de Android.

Como hemos comentado, entre otros muchos elementos, Android SDK se compone de las herramientas necesarias para el desarrollo:

SDK TOOLS

Contiene las herramientas para el desarrollo, depuración y testeo del código y UI de la aplicación.

SDK PLATFORM TOOLS

Dependencias de las plataformas para el desarrollo.

SDK PLATFORMS

Plataformas Android descargadas, con los componentes para el desarrollo y despliegue en dispositivos Android de una versión de sistema operativo concreto. Es necesario disponer de al menos una plataforma que permita la generación del código así como la creación de emuladores en caso necesario.

USB DRIVER

Driver para aquellos dispositivos que no disponen de uno propio (la mayoría de los fabricantes dispone de drivers específicos). Es necesario un driver para que el entorno de desarrollo pueda testear aplicaciones en dispositivos físicos.

GOOGLE APIs

Módulos propios de Google para componentes específicos como mapas, compras integradas o análisis de apps.

SAMPLES

Contiene el código de ejemplo y aplicaciones disponibles para cada plataforma de desarrollo de Android.

DOCUMENTATION

Copia local de la documentación de los componentes para el desarrollo y las APIs disponibles para cada plataforma.

4.2.- El entorno de ejecución.

El entorno oficial de desarrollo Android es el Android Studio. Este entorno integrado orientado exclusivamente al desarrollo móvil incluye todas las herramientas y facilidades para la construcción, depuración, testeo y distribución de aplicaciones Android, integrando componentes como:

Android SDK, para la utilización de herramientas y plataformas.

AVD Manager, para la gestión de emuladores.

DDMS, para la depuración y conexión con dispositivos.

El desarrollo Android no se tiene que realizar exclusivamente en esta plataforma, aunque en la actualidad es el entorno oficial que dispone de las actualizaciones más recientes. Otras posibilidades pueden ser: ECLIPSE + ADT

El IDE Eclipse es uno de los más empleados en el mundo de la programación, en los inicios de la programación Android fue el entorno elegido junto con el plugin ADT que permitía la integración con las herramientas propias del desarrollo móvil.

ANDROID NDK

Para el desarrollo de aplicaciones nativas. Se restringe su utilización al desarrollo de procesos que requieran una alta eficiencia o interacción muy exhaustiva con el sistema operativo subyacente.

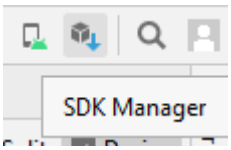
DESARROLLO CON ANDROID STUDIO

- INSTALAR JAVA: En versiones anteriores era necesario instalarlo, ahora lo incluye el instalador de Android Studio.

- INSTALAR ANDROID STUDIO: Descargar e instalar Android Studio resulta muy sencillo. https://developer.android.com/studio

- CONFIGURAR ANDROID SDK: Descargar las herramientas de Android SDK imprescindibles para poder comenzar a crear aplicaciones.

Desde la propia aplicación añades emuladores, y puedes descargarte todo lo necesario para que te funcionen, desde:



5.- Ciclo de vida de una aplicación.

La clase principal en las aplicaciones Android es Activity. Una Activity es un componente que representa una pantalla para interactuar con el usuario.

Una aplicación Android normalmente se compone de varias Activity, de forma que en cada momento solamente existe una de ellas activa. La Activity principal (main) se muestra cuando el usuario inicia la aplicación, y ésta se encargará de iniciar otras Activity en función de las necesidades de la aplicación.

Existe un mecanismo en Android denominado Back Stack utilizado para almacenar la secuencia de llamadas a Activity, de forma que cada vez que una nueva Activity comienza, la anterior es parada y almacenada en una estructura de pila. Si el usuario pulsa el botón Atrás, la Activity anterior se extrae de la pila y se muestra. Los usuarios de Android estamos acostumbrados a ver cómo al pulsar Atrás nos encontramos con la app que habíamos abierto anteriormente a la app actual.

5.1.- Perfil de desarrollo y requerimientos.

El sistema operativo será el encargado de gestionar el ciclo de instalación, eliminación, ejecución y actualización de las distintas versiones que se puedan ir generando de una misma app a través de la tienda de aplicaciones.

Una aplicación Android se puede distribuir libremente, pero lo más convencional será realizarlo a través de Google Play, de esta forma el usuario final obtiene una mayor seguridad acerca del contenido que va a descargar, así como información adicional, opiniones de otros usuarios y valoraciones.

El único requerimiento para distribuir aplicaciones Android dentro de la tienda es convertirse en desarrollador mediante el pago de una cuota inicial, y seguir los acuerdos y directrices de Google para la distribución a través de su plataforma.

Google Play permite la distribución no solamente de apps para Smartphones, Tablets y otros dispositivos que ejecutan Android, sino también es habitual encontrar libros o música de diversidad de autores.

5.2.- Dispositivos y configuraciones soportadas.

Un elemento indispensable en el desarrollo de aplicaciones Android, que determina la configuración de la misma y los permisos y características requeridos por parte de la app, es el archivo AndroidManifest.

Este fichero principal de la aplicación Android denominado AndroidManifest.xml, posee la información de todos los elementos que la componen. Toda aplicación tiene que poseer un fichero llamado de esta forma en su carpeta raíz, ya que contiene información básica para informar al sistema sobre la aplicación.

Cuando creamos un proyecto con Android Studio este fichero se genera de forma automática, pero es necesario conocer los elementos que lo componen y para qué se utilizan. A continuación, tienes un ejemplo de archivo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.hubiqus.p1t1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="23" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name="es.hubiqus.p1t1.MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
    </activity>
</application>
```

Las funciones principales del AndroidManifest son las siguientes:

REFERENCIA PAQUETE BASE

Toda app Android dispone de un nombre de paquete base que referencia a la misma de forma única tanto a nivel global en Google Play como en el dispositivo, es decir, en un móvil no se pueden instalar dos apps con el mismo nombre de paquete, ya que Android lo consideraría una actualización de la app instalada.

El concepto de paquete en Android es el mismo que en Java.

Se define en la etiqueta manifest principal del archivo con un atributo package.

COMPONENTES

Se definen los componentes de la aplicación y las clases que los implementan. Hasta ahora conocemos solamente un tipo de componente, la Activity, pero en una app pueden existir otros elementos (Services, Receivers, ...).

En cada manifest existe una etiqueta application que define los aspectos generales de la aplicación a través de atributos tales como el icono (icon - imagen que se mostrará en el menú del móvil), título (label - nombre que tendrá la app en el menú del móvil) o tema (theme - define aspectos generales de diseño), así como configuraciones más avanzadas como la orientación, el comportamiento ante cambios de giro o bloqueo o si la app se puede instalar en la SDCard del dispositivo.

Dentro de la etiqueta application encontraremos las distintas Activity que componen la app. Existirá al menos una etiqueta activity definida por los atributos name que indica el nombre de la clase Java que ejecuta dicha Activity y label, que indica el título de la pantalla. La Activity principal de la app, es decir, la que se inicia al ejecutarla, debe estar marcada con un intent-filter determinado. PERMISOS

Cuando la app requiere de elementos que pueden comprometer la seguridad, privacidad o la incursión en costes para el usuario, Android obliga a establecer explícitamente en el Manifest cuáles son los permisos necesarios de la app. La lista de permisos de la app representa la aceptación de las condiciones de la misma por parte del usuario a la hora de instalarla, de forma que éste sea consciente de los usos y acciones que la app pueda ejecutar mediante interacción directa o en segundo plano.

Los permisos se suelen colocar al inicio del manifest, con etiquetas uses-permission.

SDK MÍNIMO

Cuando estamos desarrollando una app requeriremos ciertas funcionalidades y APIs que determinan las clases y métodos Java que se encuentran disponibles a la hora de programar cada Activity y elementos asociados. Si en nuestra app utilizamos características del sistema que son avanzadas en cuanto a la versión del mismo, la app no será compatible con las versiones anteriores que no disponen de dichos avances. Para determinar en el Manifest cuál será la versión mínima soportada por la app, se utiliza el atributo minSdkVersion.

Por suerte en Android existen una serie de APIs de compatibilidad que permiten desarrollar para versiones anteriores del sistema operativo sin perder funcionalidades que hayan surgido posteriormente.

VERSIÓN

En el manifest se incluyen dos tipos de elementos para manejar las distintas versiones de la app, versionName que representa la etiqueta con el nombre de versión que se muestra a la hora de buscar en la tienda e instalar la app, y que se define con alguno de los formatos habituales para las versiones de software (como por ejemplo 1.0.3) y versionCode que se trata de un número que se irá incrementando en función de las distintas actualizaciones que se vayan obteniendo de la app y subiendo a la tienda.

Google Play también emplea el número de versión para gestionar la descarga y actualización automática de versiones de una app.

Como hemos visto, el fichero AndroidManifest indica al sistema toda la configuración de la app que se va a instalar en el sistema, y está compuesto por una serie de elementos, de los cuales hay dos que determinan si la app es compatible con el dispositivo en el cual se va a instalar, de forma que si éste no cumple con los requerimientos mínimos o el usuario no acepta las condiciones del mismo, no será instalado: los permisos y el SDK mínimo. La importancia de estos requerimientos mínimos para conocer los dispositivos que soportan la app es tal que si no existe compatibilidad por parte del móvil o tablet del usuario de alguna de las características de la app, ésta ni siquiera aparecerá disponible para el usuario en la tienda.

6.- Utilización del entorno.

Un proyecto Android se compone principalmente de dos partes diferenciadas:

Código fuente:

Se trata de clases Java de cualquier naturaleza, ya sea clases simples con atributos y métodos convencionales, clases que hereden o utilicen comportamientos de clases

Java SE habituales (exceptuando al paquete java.sql) o clases que usen parte de la Android SDK, ya sea para interacción con el interfaz gráfico o para la realización de tareas propias de la programación móvil (mensajería, conexiones, localización, ...).

En cualquiera de los casos, una clase Java de un proyecto Android puede hacer referencia a los elementos del interfaz gráfico ya que el objetivo de cualquier app móvil es permitir alta interacción con el usuario. Estos elementos del interfaz gráfico se definen de forma independiente a través de ficheros de layout dentro de la carpeta de recursos del sistema.

Recursos de la app

Es posible utilizar en una aplicación Android ficheros XML para especificar distintos tipos de recursos. Estos ficheros deben encontrarse en la ubicación /res de la aplicación, organizados en función del tipo de recurso que representan subcarpetas y cuyo nombre indica qué elementos proporcionan a la aplicación, como por ejemplo:

drawable: contiene las imágenes y recursos gráficos empleados en las pantallas.

layout: para la definición de pantallas y elementos de las mismas.

menu: definición de los menús de la app.

raw: archivos de tipo variado.

values: carpeta con valores empleados dentro de otros ficheros, como colores, estilos, cadenas, ...

Dentro de la carpeta values se pueden encontrar ficheros de distintos tipos, como por ejemplo, los ficheros de recursos String que contienen las cadenas de caracteres que se usan en la aplicación. Es una práctica muy recomendable utilizar ficheros de recursos para los títulos, etiquetas, mensajes y cualquier otro tipo de elemento textual de la aplicación, de forma que facilite el mantenimiento en caso de modificación de los mismos, la reutilización en distintos elementos gráficos, así como la traducción en caso de aplicaciones multi-idioma.

6.1.- Emuladores y pruebas.

Normalmente las herramientas de desarrollo integradas suelen incorporar emuladores que servirán para poder probar las aplicaciones antes de instalarlas en el dispositivo para el cual se han programado. Así podrás observar si la aplicación funciona correctamente y si tiene más o menos el aspecto que tenías pensado. A la hora de definir el emulador que se va a utilizar a través del AVD Manager, se pueden establecer

una serie de parámetros que permiten ajustar lo máximo posible las pruebas como si se tratara de un entorno real:

Además de emplear emuladores, una de las grandes ventajas a la hora de desarrollar con Android, es que se pueden emplear dispositivos para el testeo y depuración de la misma manera que los emuladores, incluso alcanzando un rendimiento mucho mayor, al depurar sobre dispositivos directamente y tratarse de lo más recomendado.

Cuando ejecutes la app desde Android Studio directamente te dará la opción de hacerlo sobre emulador o sobre dispositivo físico, siempre que el dispositivo se encuentre correctamente configurado: el PC de desarrollo contiene los drivers correspondientes y se ha activado la opción de depuración USB en el teléfono. Cada marca y modelo se configura de forma distinta para este objetivo.

6.2.- Depuración.

Android Studio permite, ya sea en emulador o en dispositivo físico, la ejecución en modo depuración de la app que se está construyendo. De esta forma, podrás establecer puntos de interrupción dentro del código para saber por qué "instrucciones" pasa la "ejecución" de la app, el valor de las variables en un momento determinado, examinar expresiones...en definitiva, todas las acciones que se pueden realizar en cualquier otro tipo de entorno de desarrollo.

Además de ésto, en Android es una práctica habitual realizar una traza de los puntos destacados o conflictivos de la app a través del sistema de logging.

El sistema logging de Android proporciona un mecanismo para la captura y visualización de la salida del sistema de depuración interno. Los registros (logs) de diferentes aplicaciones y de partes del sistema se recogen y muestran en la vista logcat del Android Monitor dentro de Android Studio.

La ventana logcat imprime los mensajes que se producen dentro del código y que hayan sido notificados a través de la clase Log. Dentro de logcat se pueden filtrar los mensajes según su nivel de importancia:

NIVELES DE LOG

Verbose

Para el caso de mostrar información muy exhaustiva sobre la traza.

Posee el nivel de prioridad más bajo.

Debug

Información de depuración.

Indicado para expresar valores de variables y condiciones.

Info

Información general, como entrada o salida de métodos.

Warning

Advertencias.

Error

Errores y excepciones.

En el siguiente fragmento de código podemos observar cómo se realiza el logging. Simplemente hay que invocar al método correspondiente (por ejemplo "i" para nivel info) de la clase Log, pasando una etiqueta (TAG) indicativa del elemento o clase que se está ejecutando, y una descripción textual del mensaje o error que se produce al pasar por esa etiqueta.

```
private static final String TAG = "es.curso.android.CicloDeVidaActivity";
…
protected void onStop() {
System.out.println("onStop...");
Log.i(TAG, "onStop..."); //Informar del paso por el método onStop
super.onStop();
}
```

La salida por la pantalla del logcat tendrá un formato similar al siguiente:

I/ es.curso.android.CicloDeVidaActivity (1557): onStop...