PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

UT2. Introducción a la programación de aplicaciones móviles multiplataforma

Departamento de Informática y Comunicaciones

CFGS Desarrollo de Aplicaciones Multiplataforma

Segundo Curso

IES Virrey Morcillo

Villarrobledo (Albacete)

Índice

1. Introducción

La creación de aplicaciones móviles puede ser tan fácil como iniciar el entorno integrado de desarrollo, escribir el código correspondiente a una aplicación, compilar este código, realizar unas pruebas rápidas y publicar el resultado a la Tienda de Aplicaciones o App Store, todo ello en una misma tarde. O puede ser un proceso muy complicado que conlleva el diseño riguroso por adelantado, pruebas de facilidad de uso, pruebas de control de calidad en miles de dispositivos, un ciclo de vida completo de versión beta y la implementación de varias maneras diferentes.

Al plantearse cómo crear aplicaciones para iOS y Android, muchas personas piensan que los lenguajes nativos, Objective-C, Swift, Kotlin y Java, son la única opción. Pero en los últimos años ha surgido un nuevo ecosistema de plataformas para generar aplicaciones móviles.

2. La plataforma Xamarin

La plataforma **Xamarin** es un entorno único, ya que ofrece un solo lenguaje de programación (C#), una biblioteca de clases base (BCL) y un agente de tiempo de ejecución en lenguaje (CLR) que funciona en las plataformas móviles iOS, Android y UWP. Al mismo tiempo que sigue compilando aplicaciones nativas, no interpretadas, con un rendimiento lo bastante bueno incluso para juegos exigentes.

Cada una de estas plataformas tiene un conjunto de características diferente y varía en su capacidad para crear aplicaciones nativas, por ejemplo aplicaciones que se compilan en código nativo y que interoperan con fluidez con el subsistema de Java subyacente. Por ejemplo, algunas plataformas solo permiten compilar aplicaciones en HTML y JavaScript, mientras que otras son de muy bajo nivel y solo permiten código C o C++. Algunas plataformas ni siquiera usan el kit de herramientas de control nativo.

Xamarin es una plataforma única porque combina la potencia de las plataformas nativas y agrega una serie de **características** propias muy eficaces:

- ✓ **Enlaces completos para los SDK subyacentes**: Xamarin contiene enlaces para casi todos los SDK de plataforma subyacentes en iOS y Android. Además, estos enlaces están fuertemente tipados, lo que significa que la navegación y el uso son fáciles y que proporcionan una sólida comprobación de tipos en tiempo de compilación y durante el desarrollo. Como resultado, se producen menos errores en tiempo de ejecución y aplicaciones de mayor calidad.
- ✓ **Interoperabilidad con Objective-C, Java, C y C++**: Xamarin ofrece funciones para invocar directamente las bibliotecas de Objective-C, Java, C y C++, lo que le permite usar diversos tipos de código de terceros ya creado. Esto le permite aprovechar las ventajas de las bibliotecas existentes de iOS y Android escritas en Objective-C, Java, C o C++. Además, Xamarin ofrece proyectos de enlace que le permiten enlazar fácilmente bibliotecas nativas de Objective-C y Java mediante una sintaxis declarativa.
- ✓ **Construcciones de lenguaje moderno**: las aplicaciones de Xamarin se escriben en C#, un lenguaje moderno que incluye mejoras considerables en Objective-C y Java, como características de lenguaje dinámico, construcciones funcionales como lambdas, LINQ, características de programación en paralelo, genéricos sofisticados, etc.
- ✓ **Extensa biblioteca de clases base (BCL)**: las aplicaciones de Xamarin usan la BCL de .NET, una enorme colección de clases con características completas y optimizadas, como una eficaz compatibilidad con XML, bases de datos, serialización, E/S, cadenas y redes, por nombrar algunos. Además, el código C# existente puede compilarse para su uso en una aplicación, lo

que proporciona acceso a miles de bibliotecas que le permitirán hacer cosas que no contempla la BCL.

- ✓ **Moderno entorno de desarrollo integrado (IDE)**: Xamarin usa Visual Studio para Mac en Mac OS X y Visual Studio en Windows. Estos IDE modernos incluyen características como la finalización automática de código, un sofisticado sistema de administración de proyectos y soluciones, una biblioteca exhaustiva de plantillas de proyecto, control de código fuente integrado, etc.
- ✓ **Compatibilidad multiplataforma móvil**: Xamarin ofrece una compatibilidad multiplataforma sofisticada con las tres principales plataformas móviles: iOS, Android y UWP. Es posible escribir aplicaciones de modo que compartan hasta el 90% del código, además la biblioteca de Xamarin.Mobile ofrece una API unificada para tener acceso a recursos comunes de las tres plataformas. Esto puede reducir considerablemente los costos de desarrollo y el tiempo de salida al mercado en el caso de los desarrolladores móviles que se centran en las tres plataformas móviles más populares.

3. Funcionamiento de Xamarin

La plataforma Xamarin ofrece dos productos comerciales: **Xamarin.iOS** y **Xamarin.Android**. Ambos se basan en **Mono .NET**, una versión de código abierto de .NET Framework que tiene en cuenta las normas ECMA (European Computer Manufacturers Association) de .NET publicadas. Mono .NET existe desde hace casi tanto tiempo como el propio .NET Framework y puede ejecutarse en prácticamente todas las plataformas imaginables, incluido Linux, Unix, FreeBSD y Mac OS X.

En iOS, el compilador Ahead of time (AOT) de Xamarin compila aplicaciones de Xamarin.iOS directamente en código de ensamblado nativo de la arquitectura ARM, una arquitectura de 32 bits desarrollada en 1983 por la empresa Acorn Computers Ltd para usarse en equipos que manejan un conjunto de instrucciones simple lo que le permite ejecutar tareas con un mínimo consumo de energía.

En Android, el compilador de Xamarin compila en lenguaje intermedio (IL), que es Just-In-Time (JIT) compilado en un ensamblado nativo cuando se inicia la aplicación.

En ambos casos, las aplicaciones de Xamarin usan un agente de tiempo de ejecución en lenguaje común (CLR) que lo controla todo automáticamente, por ejemplo: la asignación de memoria, la recolección de elementos no utilizados, la interoperabilidad de plataforma subyacente, etc.

Las aplicaciones de Xamarin se compilan en un subconjunto de la BCL de .NET conocido como el **perfil móvil de Xamarin** (Xamarin Mobile Profile). Este perfil se ha creado específicamente para aplicaciones móviles y se ha empaquetado en **MonoTouch.dll** y **Mono.Android.dll** para iOS y Android respectivamente.

Además de la BCL, estos archivos .dll incluyen contenedores para casi todo el SDK de iOS y Android, lo que permite invocar las API del SDK subyacente directamente desde C#.

Cuando se compilan aplicaciones de Xamarin, el resultado es un paquete de aplicación, ya sea un archivo .app en iOS o un archivo .apk en Android. Estos archivos no se distinguen de los paquetes de aplicación compilados con los IDE predeterminados de la plataforma y se implementan de la misma manera.

4. Ciclo de vida de desarrollo de software móvil (SDLC)

El ciclo de vida de desarrollo móvil o Software Development Lifecycle (SDLC) es, en gran medida, parecido al ciclo de vida de desarrollo para aplicaciones web o de escritorio.

Normalmente, el proceso de desarrollo de aplicaciones móviles consta de las siguientes fases:

1. Inicio.
2. Diseño.
3. Desarrollo.
4. Estabilización.
5. Despliegue.
6. Distribución o publicación.

4.1. Inicio

La fase de inicio consiste en definir y perfeccionar la idea de la aplicación. Para crear una aplicación correctamente, es importante hacerse algunas preguntas fundamentales. A continuación se plantean algunas cuestiones que se deben tener en cuenta antes de publicar una aplicación en una de las App Store existentes:

- ¿Existen aplicaciones similares?. Si es así, ¿cómo se diferencia esta aplicación de las otras?.
- ¿Con qué infraestructura existente se integrará o cuál extenderá?.
- ¿Qué valor aporta esta aplicación a los usuarios? ¿Cómo la usarán?.
- ¿Cómo funcionará esta aplicación en un entorno móvil? ¿Cómo puedo agregar valor mediante tecnologías móviles, como por ejemplo el reconocimiento de ubicación, la cámara, los contactos, etc.?.

Para ayudar a diseñar la funcionalidad de una aplicación, puede ser útil definir **actores y casos de uso**. Los actores son roles dentro de una aplicación y, a menudo, son usuarios. Los casos de uso son, normalmente, acciones o intentos.

4.2. Diseño

La fase de diseño consiste en definir la experiencia del usuario (UX) de la aplicación, es decir, definir el diseño general, cómo funciona la aplicación, etc. Así como convertir esa experiencia del usuario en un diseño de interfaz de usuario (UI) adecuado, normalmente con la ayuda de un diseñador gráfico.

La **experiencia del usuario** se efectúa normalmente a través de **prototipos** con uno de los muchos conjuntos de herramientas de diseño. Los prototipos de experiencia de usuario permiten diseñar esta experiencia sin tener que preocuparse por el diseño real de la interfaz de usuario.

Al crear prototipos de experiencia del usuario, es importante tener en cuenta las instrucciones de la interfaz para las diferentes plataformas a las que se dirigirá la aplicación. La aplicación debería "sentirse cómoda" en todas las plataformas. Las instrucciones de diseño oficial para cada plataforma son las siguientes:

- ✓ **Apple**: Directrices de interfaz humana
- ✓ **Android**: Instrucciones de diseño
- ✓ **UWP**: Conceptos básicos de diseño UWP

Por ejemplo, cada aplicación tiene un símbolo para cambiar de sección en una aplicación. iOS usa una barra de pestañas en la parte inferior de la pantalla. En Android usa una barra de pestañas en la parte superior. Por otro lado, UWP usa la vista dinámica o de pestañas.

Además, el propio hardware también impone decisiones de la experiencia del usuario. Por ejemplo, los dispositivos iOS no tienen ningún botón *Atrás* físico y, por tanto, presentan el símbolo de controlador de navegación.

Asimismo, el factor de forma también influye en las decisiones de la experiencia del usuario. Una tablet tiene mucho más espacio y, por tanto, puede mostrar más información. A menudo, lo que necesita varias pantallas en un teléfono, se comprime en una pantalla para una tablet.

Una vez determinada la experiencia del usuario, el siguiente paso es crear el **diseño de la interfaz de usuario**. Mientras que la experiencia del usuario suele componerse de prototipos en blanco y negro, la fase de diseño de la interfaz de usuario es donde se introducen y finalizan los colores, gráficos, etc.

Es importante comprender que cada plataforma tiene su propio lenguaje de diseño, por lo que una aplicación bien diseñada puede tener un aspecto diferente en cada plataforma.

4.3. Desarrollo

Normalmente, es la fase con un uso más intensivo de recursos, en ella se lleva a cabo la creación real de la aplicación.

Normalmente, la fase de desarrollo se inicia muy pronto. De hecho, una vez que una idea ha madurado un poco en la fase conceptual o de inspiración, a menudo se desarrolla un prototipo de trabajo que valida la funcionalidad, las suposiciones y ayuda a comprender el ámbito del trabajo.

4.4. Estabilización

El proceso de estabilización o despliegue es el encargado de solucionar los errores de la aplicación. No solo desde un punto de vista funcional, por ejemplo: "Se bloquea al hacer clic en este botón", sino también de facilidad de uso y rendimiento.

Es mejor empezar la estabilización muy pronto en el proceso de desarrollo para que se puedan realizar correcciones antes de que sean excesivamente costosas.

Normalmente, las aplicaciones pasan por las siguientes fases:

1. **Prototipo**, la aplicación aún está en fase de prueba y solo implementa la funcionalidad principal o determinadas partes de la aplicación.
2. **Alfa**, la aplicación ha alcanzado la funcionalidad principal, pero no está probada por completo.
3. **Beta**, la aplicación ya implementa la mayoría de la funcionalidad y ha pasado al menos una corrección de errores y una prueba ligeras.
4. **Versión candidata para lanzamiento**, la aplicación implementa toda la funcionalidad, está completa y probada, salvo errores nuevos, la aplicación es una versión candidata para lanzamiento.

4.5. Distribución

Una vez que la aplicación se ha estabilizado, es el momento de registrarla, publicarla o distribuirla. Para ello existen una serie de opciones de distribución diferentes, dependiendo de la plataforma seleccionada:

- ✓ **iOS**. Las aplicaciones de Xamarin.iOS se distribuyen exactamente del mismo modo:

- **Apple App Store**: el App Store de Apple es un repositorio de aplicaciones en línea disponible de forma global que está integrado en Mac OS X a través de iTunes. Sin duda, es el método de distribución de aplicaciones más popular y permite a los desarrolladores comercializar y distribuir sus aplicaciones en línea con muy poco esfuerzo.
- **Implementación interna**: la implementación interna está pensada para la distribución interna de aplicaciones corporativas que no están disponibles de forma pública a través del App Store.
- **Implementación ad hoc**: la implementación ad hoc está pensada principalmente para desarrollo y pruebas y le permite implementar en un número limitado de dispositivos aprovisionados correctamente. Al implementar en un dispositivo mediante Xcode o Visual Studio para Mac, se conoce como implementación ad hoc.

✓ **Android**. Todas las aplicaciones de Android deben estar firmadas antes de distribuirlas. Los desarrolladores firman sus aplicaciones mediante su propio certificado protegido por una clave privada. Este certificado puede proporcionar una cadena de autenticidad que une a un desarrollador de aplicaciones con las aplicaciones que ha creado y publicado.

Debe tenerse en cuenta que, mientras que un certificado de desarrollo para Android puede estar firmado por una entidad de certificación reconocida, la mayoría de los desarrolladores no optan por usar estos servicios y firman de forma automática sus certificados.

El propósito principal de los certificados es diferenciar entre las distintas aplicaciones y desarrolladores. Android usa esta información para ayudar con el cumplimiento de delegación de permisos entre las aplicaciones y componentes que se ejecutan en el sistema operativo Android.

Android adopta un enfoque muy abierto para la distribución de aplicaciones. Los dispositivos no están bloqueados para usar una única tienda de aplicaciones aprobada. En su lugar, cualquier usuario puede crear una tienda de aplicaciones y la mayoría de teléfonos Android permite que las aplicaciones se instalen desde estas tiendas de terceros.

Google Play Store es la tienda de aplicaciones oficial de Google, pero hay muchas otras. Algunas populares son: AppBrain, Tienda Apps de Amazon para Android, Handango, GetJar, etc.

✓ **UWP**. Las aplicaciones UWP se distribuyen a los usuarios a través de la **Microsoft Store**. Los desarrolladores envían sus aplicaciones para que las aprueben y después aparecen en la tienda.

El primer paso para crear una nueva aplicación en el *panel del centro de desarrollo de Windows* es reservar un nombre de aplicación. Cada nombre reservado, a veces se denomina como título de la aplicación, debe ser único en todo Microsoft Store.

Una vez que hayas creado tu aplicación reservando un nombre, puedes empezar a trabajar en conseguir que se publique. El primer paso es crear un envío.

Puedes iniciar el envío cuando la aplicación está completa y lista para publicar o puedes empezar a escribir información incluso antes de que hayas escrito una sola línea de código. Las actualizaciones que realices en tu envío se guardan, para que puedas volver y trabajar en él cuando estés listo.

5. Desarrollo de aplicaciones multiplataforma

La frase "escribir una vez y ejecutar en todas partes" se utiliza a menudo para ensalzar las virtudes de un solo código base que se ejecuta sin cambios en varias plataformas. Aunque tiene la ventaja de reutilización del código, este método conduce generalmente a aplicaciones que tienen un conjunto de características más bajo y una interfaz de usuario genérica que no se ajusta bien a cualquiera de las plataformas de destino.

La plataforma Xamarin no es simplemente una plataforma "escribir una vez y ejecutar en todas partes", sino que destaca por su capacidad de implementar las interfaces de usuario nativas específicamente para cada plataforma.

Los puntos más importantes para crear aplicaciones multiplataforma utilizando la plataforma Xamarin son los siguientes:

a. **Usar C# para escribir el código de las aplicaciones**. Este código se puede pasar a iOS, Android o UWP muy fácilmente.

b. **Usar patrones de diseño MVC o MVVM**. Desarrollar la interfaz de usuario de la aplicación mediante el patrón de diseño Modelo – Vista – Controlador (MVC) o en el patrón de diseño Modelo – Vista – Modelo de Vista (MVVM).

c. **Crear interfaces de usuario nativas**. Cada aplicación proporciona una capa de interfaz de usuario diferente, implementada en C#, con la ayuda de herramientas de diseño de interfaz de usuario nativas:

- **En iOS**, usa las APIs UIKit para crear aplicaciones de aspecto nativa, opcionalmente mediante el diseñador de iOS de Xamarin se puede crear visualmente la interfaz de usuario.
- **En Android**, usa Android.Views para crear aplicaciones de aspecto nativa, al aprovechar la posibilidad de diseñador de interfaz de usuario de Xamarin.
- **En UWP**, usa XAML para la capa de presentación, creado en el diseñador de interfaz de usuario de Visual Studio.

Para aumentar la cantidad de código reutilizable dentro de una aplicación multiplataforma, se puede utilizar alguno de los siguientes elementos:

- ✓ SQLite Net, para el almacenamiento local de SQL.
- ✓ Xamarin Plugins, para tener acceso a funciones específicas del dispositivo, por ejemplo la cámara, contactos y la ubicación geográfica.
- ✓ Paquetes de NuGet que son compatibles con los proyectos de Xamarin, como Json.NET.
- ✓ Uso de características de .NET framework para las redes, servicios web, E/S y mucho más.

Es posible maximizar el código compartido entre aplicaciones siguiendo los principios de programación por capas y de independencia de la plataforma, como se muestra en la siguiente figura:

Como ejemplo práctico se propone Tasky Pro donde se implementan los conceptos tratados en este apartado.

Atendiendo al principio de programación por capas, una aplicación típica podría estar formada por las siguientes capas:

- **Capa de datos** – gestiona la persistencia de los datos, es probable que una base de datos de SQLite, pero podría implementarse con archivos XML o cualquier otro mecanismo.
- **Capa de acceso a datos** – es un contenedor alrededor de la capa de datos que proporciona la creación, lectura, actualización, acceso de eliminación (CRUD) a los datos sin exponer los detalles de implementación al llamador. Por ejemplo, puede contener instrucciones de SQL para consultar o actualizar los datos, pero el código de referencia no sería necesario tenerlo en cuenta.
- **Capa de negocio** a veces denominado **capa de lógica empresarial** o **BLL**) – contiene las definiciones de entidad de negocio o modelo y la lógica empresarial.
- **Capa de acceso al servicio** – se usa para los servicios de acceso en la nube: servicios web complejos (REST, JSON, WCF) para la recuperación simple de datos e imágenes desde servidores remotos. Encapsula el comportamiento de la red y proporciona una API sencilla a ser consumida por los niveles de aplicación y la interfaz de usuario.
- **Nivel de aplicación** – contiene el código que normalmente es específico de la plataforma (generalmente no se comparten entre plataformas) o código que es específico de la aplicación (no reutilizable generalmente). Es una buena manera de comprobar si va a colocar código en el nivel de aplicación frente a la capa de interfaz de usuario.
- **Capa de interfaz de usuario** – el nivel de cara al usuario, contiene las pantallas, los widgets y los controladores.

Una aplicación puede no estar formada necesariamente por todas las capas, por ejemplo el nivel de acceso de servicio no existiría en una aplicación que no tiene acceso a recursos de red. Una aplicación muy simple puede combinar la capa de datos y la capa de acceso a datos porque las operaciones son muy básicas.

Los proyectos creados a partir de Xamarin.Forms admiten todas las plataformas actuales y permiten crear interfaces de usuario comunes con Xamarin.Forms XAML.

5.1. Multitarea

La multitarea, es decir, varias aplicaciones que se ejecutan a la vez, tiene dos retos importantes en un dispositivo móvil.

En primer lugar, dado el espacio real en pantalla limitado, es difícil mostrar varias aplicaciones de forma simultánea. Por tanto, en los dispositivos móviles solo puede haber una aplicación en primer plano de cada vez.

En segundo lugar, tener varias aplicaciones abiertas y realizando tareas puede agotar rápidamente la carga de la batería.

5.2. Factor de forma

Normalmente, los dispositivos móviles se dividen en dos categorías: teléfonos y tabletas, con unos cuantos dispositivos intermedios. El desarrollo para estos factores de forma es generalmente muy similar, en cambio, el diseño de aplicaciones para ellos puede ser muy diferente.

Los teléfonos tienen un espacio de pantalla muy limitado y las tabletas, aunque son más grandes, siguen siendo dispositivos móviles con menos espacio de pantalla que la mayoría de los portátiles. Por este motivo, los controles de interfaz de usuario de la plataforma móvil se han diseñado específicamente para ser efectivos en factores de forma más pequeños.

5.3. Fragmentación de sistema operativo y dispositivo

Es importante tener en cuenta distintos dispositivos en todo el ciclo de vida de desarrollo de software:

1. **Inicio**: hay que tener en cuenta que el hardware y las características varían de un dispositivo a otro, puede que una aplicación que se basa en determinadas características no funcione correctamente en algunos dispositivos. Por ejemplo, no todos los dispositivos tienen cámaras, por lo que, si está creando una aplicación de mensajería de vídeo, puede que algunos dispositivos reproduzcan vídeos, pero no los puedan grabar.
2. **Diseño**: al diseñar la experiencia del usuario (UX) de una aplicación, hay que prestar atención a las diferentes relaciones y tamaños de pantalla de los dispositivos. Además, al diseñar la interfaz de usuario (UI) de una aplicación, se deben tener en cuenta diferentes resoluciones de pantalla.
3. **Desarrollo**: al usar una característica del código, siempre se debe probar primero la presencia de esa característica. Por ejemplo, antes de usar una característica de dispositivo, como una cámara, hay que confirmar primero que el sistema operativo tenga esa característica. Después, al inicializar el dispositivo o característica, hay que asegurarse de solicitar la compatibilidad actual del sistema operativo sobre el dispositivo y después usar esas opciones de configuración.
4. **Pruebas**: es muy importante probar la aplicación al principio y con frecuencia en dispositivos reales. Puede haber incluso dispositivos con las mismas especificaciones de hardware en que varíe mucho su comportamiento.

5.4. Recursos limitados

Los dispositivos móviles son cada vez mejores con el tiempo, pero siguen siendo dispositivos móviles con capacidades limitadas en comparación con los equipos de escritorio o portátiles. Por ejemplo, los desarrolladores de escritorio normalmente no se preocupan por las capacidades de memoria; están acostumbrados a tener memoria virtual y física en grandes cantidades, mientras que en los dispositivos móviles se puede consumir toda la memoria disponible rápidamente solo con cargar algunas imágenes de alta calidad.

Además, las aplicaciones de uso intensivo del procesador, como juegos o reconocimiento de texto, pueden influir realmente en la CPU móvil y afectar negativamente al rendimiento del dispositivo.

Debido a consideraciones como estas, es importante diseñar el código de manera inteligente y realizar la implementación pronto y con frecuencia en dispositivos reales para validar la capacidad de respuesta.

6. Servicios disponibles en dispositivos móviles

Según la RAE un **servicio** se define como “función o prestación desempeñadas por organizaciones de servicio y su personal”.

Por tanto, los servicios en los dispositivos móviles son todas aquellas funciones o prestaciones de que disponen.

Un teléfono móvil es un teléfono portátil que puede hacer y/o recibir **llamadas** a través de una portadora de radiofrecuencia, mientras el usuario se está moviendo dentro de un área de servicio telefónico.

Los teléfonos móviles soportan una variedad de otros servicios, tales como **mensajes de texto** o **SMS**, **mensajes multimedia** o **MMS**, **correo electrónico**, **comunicaciones inalámbricas de corto alcance** (infrarrojos, Bluetooth).

Entre otros rasgos comunes está la función **multitarea**, el **acceso a Internet** vía WiFi o redes 2G, 3G, 4G o 5G, función **multimedia** (cámara y reproductor de videos/mp3), a los programas de **agenda**, administración de **contactos**, **acelerómetros** y algunos programas de **navegación**, así como ocasionalmente **aplicaciones de negocios** con la posibilidad de trabajar con documentos en variedad de formatos como PDF y Microsoft Office.

Los teléfonos móviles se utilizan también para proporcionar servicios de **banca móvil**, que pueden incluir la capacidad de transferir los pagos en efectivo por mensaje de texto SMS seguro.

Los teléfonos móviles actuales se utilizan comúnmente para recopilar datos de **geolocalización**. Mientras que el teléfono está encendido, la ubicación geográfica de un teléfono móvil se puede determinar con facilidad usando una técnica conocida como multilateración para calcular las diferencias en tiempo que una señal viaja desde el teléfono móvil a cada una de varias torres de telefonía móvil cercanas al propietario del teléfono.

No hay que olvidar la capacidad para ejecutar una gran variedad de **juegos**. Actualmente el mercado de los videojuegos para móviles es más grande que cualquier otro mercado de videojuegos portátiles, teniendo cifras de ventas elevadas. Como aplicaciones para el futuro se esperan videojuegos en 3D y videojuegos en red a través de teléfono o WiFi o Bluetooth.

7. Proveedores de contenido

Las nuevas tecnologías avanzan a una velocidad de vértigo y los usuarios se incorporan a la sociedad de la información con fuerza. Cualquier organización que se precie debe plantear la disponibilidad de sus servicios y aplicaciones a través de múltiples canales de forma que asegure la accesibilidad para el usuario y con la imperiosa necesidad de contar con la información dónde y cuándo se necesite.

Los **proveedores de contenido** son todas las personas, empresas o profesionales que publican con regularidad información de cualquier tipo en Internet, ya sea utilizando recursos propios o sirviéndose de los suministrados por un proveedor de acceso.

Los proveedores de contenido más importantes en la actualidad son los siguientes:

- ✓ **Google Play**.
- ✓ **Apple Store**.
- ✓ **Microsoft Store**.
- ✓ **Amazon Appstore**.

El 86% de los teléfonos inteligentes vendidos en todo el mundo, durante el primer trimestre de 2017, fueron Android, según la consultora Gartner. Por eso, no resulta extraño que el sistema operativo dominante también sea el que más propuestas ofrece a sus usuarios, con 2,8 millones de apps disponibles en **Google Play**, en marzo de 2017, seguido de los 2,2 millones existentes en la **App Store** de Apple, según la clasificación elaborada por Statista.

Los usuarios recurren a alternativas a las tiendas oficiales para instalar aplicaciones que aún no han sido lanzadas en su país o que ofrecen promociones.

Un concepto distinto dentro de las alternativas a Google Play es **Aptoide**, una propuesta de origen portugués que nació en 2011 con el objetivo de reinventar la distribución de aplicaciones móviles a través de una plataforma social y colaborativa. Se trata de una comunidad donde cualquiera puede crear y gestionar su propia tienda con archivos APK, de tal modo que actualmente engloba a más de 245.000 establecimientos online que ofrecen cerca de 795.000 apps. Según sus responsables, Aptoide cuenta con más de 100 millones de usuarios que hasta la fecha han realizado unos 3.000 millones de descargas.

Otro ejemplo es el de la española **Uptodown**, que comenzó en 2002 como una web de descargas de software para ordenadores y que, con el auge de la movilidad, fue añadiendo a su oferta propuestas para Android, de tal modo que actualmente cuenta con 30.000 aplicaciones para este sistema operativo. Según sus responsables, el año pasado, 1.200 millones de usuarios utilizaron esta web para descargarse 2.000 aplicaciones para Android.

Otras de las tiendas para Android más utilizadas por los que quieren encontrar aplicaciones fuera de Google Play son **SlideMe**, con unas 26.000 apps; el repositorio **APKMirror**, que recopila los archivos APK de las aplicaciones más populares, y la plataforma de código abierto **F-Droid** con unas 2.200 aplicaciones.

Por lo que respecta a Apple, si bien es cierto que existen tiendas diferentes de la **App Store** con aplicaciones para iPhone y iPad, no resulta tan sencillo acceder a ellas como en el caso de las alternativas a Google Play. No en vano, para saltarse las restricciones impuestas en los dispositivos con iOS, es necesario realizar en el terminal lo que se conoce como *jailbreak*, un proceso con el que se modifica el sistema operativo para permitir la instalación de software de terceros. Sin embargo, tal y como recuerda Apple, las modificaciones no autorizadas de iOS no solo derivan en una pérdida de garantía del dispositivo, sino que pueden provocar inestabilidad, vulneraciones de la seguridad, una reducción de la vida de la batería y otros problemas.

8. Seguridad en las aplicaciones móviles

Nuestros móviles pueden revelar muchas cosas sobre nosotros: dónde vivimos y trabajamos; quiénes son nuestros familiares, amigos y conocidos; cómo nos comunicamos con ellos (e incluso qué comunicamos), así como nuestros hábitos personales. Con toda esta información almacenada en los dispositivos, no es de extrañar que los usuarios tomen medidas para proteger su privacidad, como utilizar números de identificación personal o códigos de acceso para desbloquear el teléfono.

Cuando los internautas instalan una nueva aplicación Android o iOS, esta pide permiso al usuario antes de acceder a la información personal. En términos generales, esto es positivo. Además, parte de la información que recogen estas aplicaciones es necesaria para que funcionen correctamente. Por ejemplo, un mapa para móvil sería muchísimo menos útil si no pudiese utilizar los datos del GPS para encontrar una localización.

Pero, una vez que la aplicación tiene permiso para recoger esa información, puede compartir tus datos con quien su creador quiera, permitiendo así que terceras empresas hagan un seguimiento de dónde te encuentras, a qué velocidad te mueves y qué estás haciendo.

Una aplicación no solo recopila datos para utilizarlos en el propio teléfono móvil. Por ejemplo, los mapas mandan tu localización a un servidor gestionado por el creador de la aplicación para que calcule las direcciones desde el punto donde te encuentras hasta el destino deseado.

Asimismo, la aplicación puede mandar datos a cualquier parte. Igual que las páginas web, muchos programas para móviles están escritos combinando diversas funciones, pre-codificadas por otros fabricantes y empresas, en lo que se denomina “bibliotecas de terceros”. Estas bibliotecas ayudan a los fabricantes a rastrear los intereses de los usuarios, conectar con las redes sociales y ganar dinero mostrando anuncios y otros elementos sin tener que escribirlos de cero.

Sin embargo, aparte de su valiosa ayuda, la mayoría de las bibliotecas también recogen datos delicados y los envían a sus servidores o a otra empresa totalmente ajena. Los creadores de bibliotecas más competentes son capaces de elaborar detallados perfiles digitales de los usuarios. Por ejemplo, puede que una persona dé permiso a una aplicación para que sepa cuál es su localización, y que a otra le dé acceso a sus contactos. En principio, ambos son permisos separados, uno para cada aplicación; pero si las dos utilizan la misma biblioteca de terceros y comparten fragmentos de información diferentes, el creador de la biblioteca puede conectar esos fragmentos.

Los usuarios nunca se enterarán, porque las aplicaciones no tienen que informarles de las bibliotecas de programas que utilizan. Además, muy pocas aplicaciones hacen pública su política de privacidad y, en caso de que lo hagan, suele ser mediante extensos documentos legales que una persona normal no lee y muchos menos entiende.