PROGRAMACIÓN MULTIMEDIA
Y
DISPOSITIVOS MÓVILES

UT1. Análisis de tecnologías para aplicaciones en dispositivos móviles

Departamento de Informática y Comunicaciones

CFGS Desarrollo de Aplicaciones Multiplataforma

Segundo Curso

IES Virrey Morcillo

Villarrobledo (Albacete)

Índice

1. Introducción 3
2. Ejecución de aplicaciones en los dispositivos móviles 3
3. Sistemas operativos para dispositivos móviles 4
   3.1. Android 4
   3.2. iOS 5
   3.3. Plataforma Universal de Windows (UWP) 5
4. Tecnologías de desarrollo de aplicaciones para dispositivos móviles 6
   4.1. Aplicaciones Web Móviles 6
   4.2. Aplicaciones Web Progresivas 7
   4.3. Aplicaciones Móviles Nativas 7
   4.4. Aplicaciones Móviles Híbridas Multiplataforma basadas en tecnologías Web 8
   4.5. Aplicaciones Móviles Multiplataforma basadas en un Lenguaje Común 10
5. Entornos Integrados de Desarrollo (IDE) 11
6. Emuladores 13
   6.1. Configuración de Android Emulator 15
   6.2. Editar las propiedades de perfil de un dispositivo virtual Android 17
   6.3. Simulador remoto de iOS para Windows 18
   6.4. Configurar el dispositivo para el desarrollo 18
   6.5. Xamarin Live Player 18
   6.6. Xamarin Hot Reload 19
7. Clasificación de los dispositivos móviles 20
8. Relación entre el dispositivo y la aplicación 22

1. Introducción

El **desarrollo de aplicaciones móviles** es el proceso por el cual se desarrolla un software para dispositivos móviles, como smartphones o tablets. La forma de distribución de estas aplicaciones puede variar, las aplicaciones pueden venir preinstaladas en los teléfonos o pueden ser descargadas por los usuarios desde tiendas de aplicaciones y otras plataformas de distribución de software.

Las principales medidas para el desarrollo de aplicaciones móviles han cambiado mucho y lo seguirán haciendo. Una de ellas es la popularidad de las diferentes plataformas entre los desarrolladores. En los últimos tiempos se ha producido una migración en las preferencias de los desarrolladores, que los ha movido desde la vieja guardia Symbian, BlackBerry y Java hacia los nuevos reyes del sector iOS y Android.

Según algunos estudios, cerca del 60% de los desarrolladores han desarrollado aplicaciones para Android de Google. El iOS de Apple ocupa el segundo lugar con más del 50%, seguido por Java ME de Oracle, que se encuentra en tercera posición.

Durante los últimos años ha habido una explosión de las herramientas y lenguajes de programación para desarrollar aplicaciones sobre dispositivos móviles, así como la creación de nuevas maneras de compartir y vender estas aplicaciones a partir de mercados específicos llamados tienda de aplicaciones o AppStores.

Todo esto ha hecho posible que numerosos programadores estén desarrollando aplicaciones para móviles de una manera rápida, barata y fácilmente comercializable. Nunca ha sido tan fácil crear aplicaciones y poder tener un escaparate de alcance mundial para poder venderlas.

2. Ejecución de aplicaciones en los dispositivos móviles

Una gran cantidad de dispositivos electrónicos se clasifican actualmente como dispositivos móviles, desde teléfonos hasta tablets, pasando por dispositivos como lectores de identificación por radiofrecuencia o RFID. Con tanta tecnología clasificada como móvil, puede resultar complicado determinar cuáles son las características de los dispositivos móviles.

A continuación detallamos las **características esenciales** que tienen los dispositivos móviles en la actualidad:

a) Son aparatos pequeños.

b) La mayoría de estos aparatos se pueden transportar en el bolsillo del propietario o en un pequeño bolso.

c) Tienen capacidad de procesamiento.

d) Tienen conexión permanente a una red.

e) Tienen memoria (RAM, tarjetas MicroSD, flash, etc.).

f) Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, la cual puede adaptarlos a su gusto.

g) Tienen una alta capacidad de interacción mediante la pantalla o el teclado.

En la mayoría de los casos, un dispositivo móvil puede definirse con cuatro características que lo diferencian de otros dispositivos que, aunque pudieran parecer similares, carecen de algunas de las características de los verdaderos dispositivos móviles. Estas cuatro características son:

1) Movilidad.

2) Tamaño reducido.
3) Comunicación inalámbrica.
4) Interacción con las personas.

3. Sistemas operativos para dispositivos móviles

Un **sistema operativo móvil** es un conjunto de programas de bajo nivel que permite la abstracción de las peculiaridades del hardware específico del dispositivo móvil y provee servicios a las aplicaciones móviles, que se ejecutan sobre él.

Al igual que los ordenadores personales que utilizan Windows, Linux o Mac OS, los dispositivos móviles tienen sus sistemas operativos como Android, iOS, UWP de Windows o BlackBerry OS, entre otros.

Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, a los formatos multimedia para móviles y a las diferentes maneras de introducir información en ellos.

3.1. Android

El sistema operativo de Android es de código abierto en su totalidad, lo que significa que cualquiera, incluso la competencia de Android, puede descargar, instalar, modificar y distribuir su código fuente de forma gratuita. En consecuencia, más usuarios que nunca tienen acceso al poder de la tecnología móvil.

Aunque los primeros smartphones eran impresionantes para la época, tenían un precio muy elevado. Los fabricantes de dispositivos tenían que pagar las licencias del sistema operativo o hacerse cargo del coste derivado de desarrollar el suyo propio desde cero. Además, los desarrolladores y fabricantes de terceros no tenían acceso a la mayoría de sistemas operativos. En otras palabras, los smartphones no eran para todos.

En consecuencia, Google se alió en 2007 con el sector de los dispositivos móviles para crear la alianza empresarial Open Handset Alliance (OHA). Su misión era establecer Android como un sistema operativo de código abierto. Esto significaba que todo el mundo podría acceder al código fuente de Android, descargarlo y modificarlo de forma gratuita para poder desarrollar aplicaciones, dispositivos móviles o incluso un sistema operativo competidor.

Las ventajas fueron inmediatas. Las empresas pudieron personalizar Android para desarrollar experiencias únicas para sus clientes, los desarrolladores de aplicaciones pudieron acceder a una audiencia global y, quizás lo más impactante, los fabricantes de dispositivos pudieron instalar Android gratis sin pagar licencias ni desarrollar su propio sistema operativo.

La competencia y la innovación son fundamentales en un ecosistema móvil que ofrece un mayor acceso a la tecnología. El hecho de que el código fuente de Android sea abierto ayuda a impulsar un entorno de este tipo. Todo el mundo puede utilizar Android para crear aplicaciones y desarrollar software competidor sin pagar una cuota o cerrar un acuerdo con Google.

Además, los fabricantes de dispositivos y los operadores de telefonía móvil no tienen la obligación en ningún momento de preinstalar las aplicaciones de Google en sus dispositivos.

Las ventajas sin precedentes que ofrece Android no solo se limitan a los dispositivos. Solo en Google Play Store hay más de 1 millón de aplicaciones disponibles y esa variada gama de aplicaciones abre un nuevo mundo de experiencias.

3.2. iOS

iOS es un sistema operativo móvil de Apple Inc. Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad. No se permite la instalación de iOS en hardware de terceros.

Actualmente es el segundo sistema operativo móvil más utilizado del mundo, detrás de Android, con una cuota de mercado de entre 10-15% al año 2017. La última versión del sistema operativo es iOS12 aparecida en junio de 2018.

Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo, por ejemplo, para el comando deshacer, o rotarlo en tres dimensiones, un resultado común es cambiar de modo vertical al apaisado u horizontal.

iOS comprende el sistema operativo y las tecnologías que se usan para ejecutar aplicaciones de forma nativa en dispositivos como iPad, iPhone y iPod touch. Aunque comparte una herencia común y muchas tecnologías de base con el Mac OS X, iOS se diseñó para satisfacer las necesidades de un entorno móvil, donde las necesidades de los usuarios son ligeramente diferentes. Si se han desarrollado previamente aplicaciones para Mac OS X, se encontrarán muchas tecnologías familiares, pero también tecnologías que solo están disponibles en iOS, como el soporte a interfaz táctil o al acelerómetro.

El SDK de iOS contiene el código, la información y las herramientas necesarias para desarrollar, probar, ejecutar, depurar errores y adaptar aplicaciones para iOS. Las herramientas Xcode proporcionan el entorno básico para editar, compilar y depurar errores en el código. Xcode también proporciona el punto de lanzamiento para probar las aplicaciones en un dispositivo iOS y en un simulador iOS, una plataforma que imita el entorno básico iOS, pero se ejecuta en un ordenador Macintosh local.

La interfaz de usuario del iOS se basa en el concepto de manipulación directa mediante la utilización de gestos multicontacto. Los elementos de control de la interfaz consisten en deslizadores, interruptores y botones. La respuesta a las peticiones del usuario es inmediata y proporciona una interfaz fluida. La interacción con iOS incluye gestos como "tocar fuerte", "tocar de forma más débil", "sujetar" y "soltar", que tienen definiciones específicas en el contexto del sistema operativo iOS y su interfaz multicontacto.

3.3. Plataforma Universal de Windows (UWP)

La Plataforma universal de Windows (UWP) es una plataforma común de aplicaciones presentes en todos los dispositivos que cuentan con Windows 10 y sus variantes, que se presentó en Windows 8 por primera vez como WinRT (Windows Runtime).

Cuando se publicó Windows Phone 8.1, se adaptó Windows en tiempo de ejecución a Windows Phone 8.1 y a Windows. Esto permitió a los desarrolladores crear aplicaciones universales para Windows 8 destinadas a Windows y a Windows Phone con un código base compartido.

Windows 10 presenta la Plataforma universal de Windows, que continúa el desarrollo del modelo de Windows en tiempo de ejecución y lo incorpora al núcleo unificado de Windows 10. Como parte del núcleo, UWP ahora proporciona una plataforma común de aplicaciones disponible en todos los dispositivos que se ejecutan en Windows 10 en todas sus ediciones.

Las aplicaciones para UWP:

- ✓ Son seguras ya que declaran cuáles son los recursos y datos de dispositivos a los que acceden y el usuario debe autorizar ese acceso.
- ✓ Pueden usar una API común en todos los dispositivos que usan Windows 10.
- ✓ Pueden usar funcionalidades específicas de dispositivo y adaptar la interfaz de usuario a las PPP, resoluciones y tamaños de pantalla de diferentes dispositivos.
- ✓ Están disponibles desde Microsoft Store en todos los dispositivos o solo en aquellos que especifiques que se ejecutan en Windows 10. Microsoft Store proporciona varias maneras de ganar dinero en tu aplicación.
- ✓ Pueden instalarse y desinstalarse sin riesgo para la máquina.
- ✓ Son atractivas ya que usan iconos dinámicos, notificaciones de inserción y actividades del usuario que interactúan con la escala de tiempo de Windows y la función *Continuar donde lo dejé* de Cortana para atraer a los usuarios.
- ✓ Son programables en C#, C++, Visual Basic y Javascript y para la interfaz de usuario usa XAML, HTML o DirectX.

4. Tecnologías de desarrollo de aplicaciones para dispositivos móviles

En el ámbito del desarrollo de software se deben considerar los lenguajes de programación para el desarrollo y las plataformas software como frameworks, librerías del sistema, etc. Las más comunes en la actualidad son Android, iOS y UWP.

Se deben considerar los lenguajes de programación utilizados: Objective-C o Swift para iOS, Java o Kotlin para Android y C# para UWP de Windows.

Por último, también se debe tener en cuenta los entornos de desarrollo típicos de cada plataforma: Xcode para iOS, Android Studio para Android y Microsoft Visual Studio para UWP de Windows.

Dependiendo del tipo de aplicaciones móviles debemos considerar diferentes tecnologías de desarrollo de las mismas.

4.1. Aplicaciones Web Móviles

Una aplicación web es, básicamente, un sitio web específicamente optimizado para un dispositivo móvil. Las características que definen una aplicación web son las siguientes: la interfaz de usuario se construye con tecnologías web estándar, está disponible en una URL pública, privada o protegida por una contraseña y está optimizada para los dispositivos móviles.

Una aplicación web no está instalada en el dispositivo móvil. Se basan en tecnologías web: HTML, CSS, JavaScript. Accedemos a ellas a través del navegador de nuestro móvil.

Las ventajas del desarrollo de aplicaciones web móviles son las siguientes:

- ✓ Hacemos un único desarrollo para todas las plataformas que queramos: Android, iOS, Windows.
- ✓ Los desarrolladores web pueden usar sus propias herramientas.
- ✓ Se pueden usar los conocimientos de diseño y desarrollo web que ya se tengan.
- ✓ La aplicación funcionará en cualquier dispositivo que tenga un navegador web.
- ✓ Se pueden solucionar errores en tiempo real.

- ✓ El ciclo de desarrollo es más rápido.

Los inconvenientes del desarrollo de aplicaciones web móviles son los siguientes:

- × No se puede acceder a todas las características y capacidades del dispositivo móvil.
- × Puede ser difícil conseguir efectos sofisticados en la interfaz de usuario.
- × No se distribuyen a través de las tiendas de aplicaciones.

4.2. Aplicaciones Web Progresivas

Debido a que los navegadores están mejorando poco a poco vamos pudiendo acceder a más capacidades del dispositivo. De esta manera, acercan la experiencia de uso a la de una aplicación nativa.

Las aplicaciones web progresivas son una evolución natural de las aplicaciones web móviles que difuminan la barrera entre la web y las aplicaciones, pudiendo realizar tareas que generalmente sólo las aplicaciones nativas podían llevar a cabo. Algunos ejemplos son las notificaciones, el funcionamiento sin conexión a Internet o la posibilidad de probar una versión más ligera antes de bajarte una aplicación nativa de verdad.

Las aplicaciones web progresivas están a medio camino entre las aplicaciones web móviles y las aplicaciones móviles nativas, es decir, son básicamente páginas web pero mediante el uso de service workers y otras tecnologías se comportan más como aplicaciones normales que como aplicaciones web.

Un service worker consiste en un fichero JavaScript formado por una secuencia de comandos que tu navegador ejecuta en segundo plano, separado de una página web, abriéndoles la puerta a funciones que no necesitan una página web ni interacción de usuario. En la actualidad, ya incorporan funciones como notificaciones de aplicación y sincronización en segundo plano. Mediante los service workers y otras tecnologías, las aplicaciones web progresivas pueden seguir ejecutándose en segundo plano sin tener que vivir dentro del navegador.

En el móvil es posible instalarlas como una aplicación más y también en Windows mediante la mediación de Google Chrome y Mozilla Firefox. Windows 10 añade soporte para aplicaciones web progresivas en la actualización Redstone 4 del mes de abril del 2018.

4.3. Aplicaciones Móviles Nativas

En términos generales, una aplicación móvil nativa es aquella que ha sido específicamente desarrollada para el sistema operativo en el que se ejecuta.

Las aplicaciones móviles nativas son las aplicaciones propias de cada plataforma. Deben ser desarrolladas pensando en la plataforma concreta. No existe ningún tipo de estandarización, ni en las capacidades ni en los entornos de desarrollo, por lo que los desarrollos que pretenden soportar plataformas diferentes suelen necesitar un esfuerzo extra.

Estas aplicaciones son las que mayor potencial tienen, pues aprovechan al máximo los dispositivos y consiguen, de esa manera, una mejor experiencia de usuario.

Existen muchas plataformas, una gran parte de ella ligadas al tipo de dispositivo, aunque también hay plataformas, como Android, que existen para diferentes tipos de dispositivos.

Algunas de las más conocidas son Android, iOS, UWP (antes Windows Phone, Windows Mobile o Windows Ce), Blackberry, bada, Java Me, Symbian, Web OS, Brew, etc. Todas ellas tienen diferentes tipos de dispositivos con una base común entre ellos.

Las ventajas de las aplicaciones móviles nativas son las siguientes:

- ✓ Acceso total al contexto, con todas las posibilidades que eso conlleva. Consigue las mejores experiencias de usuario.
- ✓ Posibilidad de gestión de interrupciones en la aplicación o en las capacidades del dispositivo. Desde saber si tenemos conexión de datos o conexión de localización hasta tener información sobre la batería.
- ✓ Son relativamente fáciles de desarrollar si solo se contempla una plataforma.
- ✓ Se pueden distribuir por los canales conocidos de aplicaciones que permita la plataforma, con lo que se pueden vender más fácilmente.
- ✓ Todas las novedades llegan primero a este tipo de aplicaciones, pues es en este tipo de aplicaciones donde se prueban.

En cambio, tienen sus inconvenientes:

- × Portar aplicaciones es costoso. En el caso de querer realizar una aplicación para más de una plataforma, se complica el desarrollo, debido a los problemas de la fragmentación.
- × Dependiendo de la plataforma elegida, puede haber fragmentación dentro de cada plataforma, debido a los diferentes tipos de dispositivos o versiones de la plataforma.
- × No existe un estándar, por lo que cada plataforma ofrecerá sus peculiaridades.
- × Normalmente, para desarrollar, distribuir o probar estas aplicaciones en dispositivos reales, es necesario tener una licencia de pago, dependiendo de la plataforma.
- × Las ganancias por estas aplicaciones suelen repartirse entre el creador de la aplicación y la plataforma de distribución.

4.4. Aplicaciones Móviles Híbridas Multiplataforma basadas en tecnologías Web

Para poder soportar todas las plataformas, necesitaríamos saber muchos lenguajes de programación, ya que habría que portar dichas aplicaciones entre plataformas. En concreto, en la actualidad existen al menos siete lenguajes diferentes, que son necesarios para poder realizar aplicaciones sobre las plataformas más actuales: C, C++, Java, Kotlin, C#, Javascript, Objective-C, Swift, además de los diferentes IDE necesarios y sus correspondientes librerías específicas.

Las aplicaciones web móviles nos permiten llegar a muchas plataformas con un mismo código sin necesidad de portar el código, pero sin poder llevar al usuario la misma experiencia que consigue con las aplicaciones móviles nativas. Por lo tanto, si existiera la posibilidad de realizar aplicaciones móviles nativas desde una misma línea de código, tendríamos lo mejor de ambas aproximaciones. Aquí es donde entran en juego las estrategias de aplicaciones móviles híbridas.

Con la llegada de HTML5 y la explosión de nuevos smartphones se han creado nuevas alternativas, que se deben tener en cuenta. Muchas de estas alternativas aprovechan HTML5 como base y construyen a su alrededor diferentes maneras de acceder a las capacidades que HTML5 no da de partida, prácticamente siempre mediante objetos JavaScript. Usar elementos 100% estándar (como HTML5, CCS y JavaScript) ofrece un gran punto a favor, pues se trata de tecnologías muy conocidas.

Las hay que simplemente se quedan en un wrapper o carcasa de la aplicación HTML5 y añaden estos puntos de acceso. En cambio, otras realizan pre-procesamiento para acabar generando aplicaciones 100% nativas. Hay otras alternativas que proporcionan su propia arquitectura y sus

propios lenguajes, y también mediante un sistema de compilación o ejecución vía máquina virtual consiguen tener aplicaciones nativas.

Las ventajas de este tipo aplicaciones son las siguientes:

- ✓ Solo una línea de desarrollo para varias plataformas.
- ✓ Aplicaciones que podemos instalar en nuestros dispositivos, que tienen la posibilidad de ser distribuidas en los mercados o tiendas de aplicaciones.
- ✓ Dependiendo del caso, el resultado son aplicaciones nativas que aprovechan en gran medida el potencial del dispositivo y ofrecen un diseño y experiencia de usuario idéntica a las aplicaciones desarrolladas sólo para una plataforma.
- ✓ Están desarrolladas con una mezcla de tecnologías web y de tecnologías nativas. Esto es posible mediante un wrapper hecho en código nativo que dentro tiene un componente o webview, que incrusta un navegador dentro de nuestra aplicación de forma que dentro de esa carcasa nativa podemos visualizar el código HTML, CSS, JS que hacemos dentro de ese navegador.

En cambio, tienen sus inconvenientes:

- × Al tener un navegador por medio el rendimiento no es tan buena que una aplicación nativa.
- × Pérdida de controles específicos de una plataforma: si tenemos un control de la interfaz gráfica o una funcionalidad concreta que solo existe en una plataforma, no podremos generar de manera única, por nuestro desarrollo multiplataforma.
- × Integración en el escritorio del dispositivo: según la plataforma, las posibilidades de añadir elementos en el escritorio de cada usuario varían. Por ejemplo, en Android o Symbian se pueden crear widgets potentes para mejorar la el uso de nuestra aplicación, mientras que en Windows solo es posible añadir iconos de la misma.
- × Gestión de la multitarea: debido a que se trata de conceptos de bajo nivel de cada plataforma, cada una la trata de manera diferente, con restricciones diferentes, por lo que no será fácil hacer código común para todas sin perder mucha potencia.
- × Consumo de la batería: estas aproximaciones requieren de una capa de abstracción sobre nuestro dispositivo, que provoca problemas como la multitarea. De la misma forma, el control sobre el consumo de batería se hace más difícil cuando no se tienen las capacidades concretas de la plataforma. También afecta el hecho de no tener el control sobre la multitarea, ya que esta es una de las maneras de ahorrar las baterías.
- × Servicios de mensajería asíncrona o push services: sirven para implementar elementos como la mensajería instantánea, pero debido a que cada plataforma los implementa de una manera, se hace complicado atacarlos conjuntamente.

A continuación se presentan algunos de los frameworks o marcos de trabajo más utilizados en la actualidad:

a) **Ionic**. Son aplicaciones para móviles basadas en las tecnologías web HTML, CSS y JavaScript. Éstas se ejecutan en lo que se denomina un web view. Es decir, utiliza el navegador del dispositivo para poder mostrarla.

   Al estar desarrollado sobre el framework Javascript **Angular 2**, contamos con todas las ventajas que nos proporcionará: nuestro código tendrá más calidad y podremos usar muchas de sus utilidades pensadas para un desarrollo más rápido.

b) **Adobe PhoneGap**. Son aplicaciones desarrolladas en HTML5 con objetos JavaScript que permiten el acceso, mediante enlaces, a las funciones nativas para las capacidades que HTML5 no ofrece. Las aplicaciones se ejecutan sobre un componente que contiene un navegador.

   Se trata de una plataforma basada en el proyecto libre **Apache Cordova** que nos permiten acceder a las capacidades internas del dispositivo (gestos, sensores, localización, etc.), además de un conjunto de librerías de terceros.

   Apache Cordova es un framework open source para el desarrollo de aplicaciones móviles usando tecnologías web (HTML, CSS, JavaScript). Apache Cordova fue creado a partir de PhoneGap de Adobe, y proporciona herramientas de valor añadido sobre la plataforma de Cordova, como una aplicación que facilita el desarrollo o un servicio para crear la aplicación final que distribuimos en las tiendas de aplicaciones. Cordova tiene una arquitectura basada en plugins que nos permite acceder a nuevas características del dispositivo desde su envoltorio nativo.

c) **Rhodes.** Son aplicaciones desarrolladas en Ruby que utilizan el patrón de diseño Modelo Vista Controlador (MVC) y que se ejecutan en máquinas virtuales específicas de Ruby para cada plataforma. Por tanto, son aplicaciones nativas. Incluyen sistemas para sincronizar datos de manera sencilla y conseguir, de esa manera, el cambio de "en línea" a "fuera de línea" sin mucho coste.

d) **Appcelerator**. Son aplicaciones desarrolladas en HTML5 que son compiladas en aplicaciones nativas. También son capaces de generar una aplicación de sobremesa clásica y ejecutable sobre Mac, Linux o Windows.

e) **Java ME**. Son aplicaciones desarrolladas en Java, con todas las peculiaridades de los perfiles J2ME, y se ejecutan mediante una máquina virtual, con sus correspondientes restricciones. Actualmente están en casi el ochenta por ciento de los dispositivos móviles del mercado.

4.5. Aplicaciones Móviles Multiplataforma basadas en un Lenguaje Común

Tradicionalmente, cuando hablamos de tecnologías multiplataforma, pensamos en el enfoque del mínimo denominador común: reducir nuestra implementación y desarrollo a un mínimo denominador de todas las plataformas.

Las aplicaciones multiplataforma, como las desarrolladas con **Xamarin**, nos ofrecen otro enfoque. Nos permiten compartir la lógica interna de nuestra aplicación, mientras nos exige que escribamos la interfaz de usuario de forma nativa. Esto nos permite diseñar experiencias óptimas en cada plataforma y nos ayuda a crear una base común de código, usando una sola herramienta y un único lenguaje para nuestro desarrollo.

Podríamos decir que es un paso intermedio entre crear la aplicación totalmente nativa y el desarrollo totalmente multiplataforma.

Las ventajas de este tipo aplicaciones son las siguientes:

- ✓ Podemos crear cualquier tipo de aplicación para las plataformas más usadas: iOS, Android, Mac y Windows.
- ✓ Tenemos una correspondencia del 100% con las APIs nativas de cada plataforma, más las APIs comunes de la plataforma .NET.

- ✓ Existen además frameworks como Unity o MonoGame, que nos ayudan incluso al crear juegos.

Virtualmente no tiene ningún inconveniente:

- × La máxima de Xamarin es que cualquier cosa que puedas hacer con ObjectiveC, Swift o Java, la vas a poder escribir con Xamarin y C#.
- × Además como la interfaz gráfica es nativa, diseñada para cada plataforma, el rendimiento va a ser muy bueno, totalmente comparable a una aplicación nativa.
- × Hace poco Harry Cheung, uno de los ingenieros de Google, escribió un artículo en Medium comparando el rendimiento de Xamarin con Swift, ObjectiveC, Java, RoboVM, RubyMotion y J2ObjC. El resultado fue escoger Xamarin para su proyecto.

5. Entornos Integrados de Desarrollo (IDE)

Un **entorno integrado de desarrollo (IDE)** no es más que un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación empaquetado como un programa o aplicación, que nos provee de un marco de trabajo agradable para la mayoría de los lenguajes de programación. Constan entre sus características básicas con:

- ✓ Editor de código.
- ✓ Compilador.
- ✓ Depurador.
- ✓ Constructor de interfaz gráfica.

La selección de un entorno de desarrollo para la programación multiplataforma debe tener en cuenta los diferentes escenarios, modelos y configuraciones que intervienen en este contexto.

Los principales entornos integrados de trabajo para el desarrollo de aplicaciones móviles son los siguientes:

a) **Eclipse**. Es un entorno integrado de desarrollo compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-Liviano" basadas en navegadores.

   Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse y que son usados también para desarrollar el mismo Eclipse.

b) **Carbide.C++.** Es una herramienta para el desarrollo de software en lenguaje C++ destinado a dispositivos que funcionan bajo Symbian OS. Se usa tanto para desarrollar los teléfonos que incorporan dicho OS como para las aplicaciones que ejecutan estos.

   Está formada por una familia de IDEs desarrollada por Nokia, basada en Eclipse, al que se han incorporado plugins para el desarrollo de Symbian OS. Reemplazó a CodeWarrior como primer entorno de desarrollo para Symbian OS. Desde la versión 2.0, Carbide.c++ es gratuito y se ofrece en tres versiones (Developer, Professional y OEM).

c) **NetBeans**. Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Se trata de un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con más de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Desde la creación de la especificación J2ME (Java 2 Platform, Micro Edition), una versión del entorno de ejecución Java reducido y altamente optimizado, especialmente desarrollado para el mercado de dispositivos electrónicos de consumo se ha producido toda una revolución en lo que a la extensión de Java se refiere.

Es posible encontrar microprocesadores diseñados para ejecutar bytecode Java y software Java para tarjetas inteligentes (JavaCard), teléfonos móviles, buscapersonas, sintonizadores de TV y otros pequeños electrodomésticos.

d) **XCode**. Es el entorno integrado de desarrollo de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interface Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfaces de usuario. Xcode incluye la colección de compiladores del proyecto GNU (GCC), y puede compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript mediante una amplia gama de modelos de programación, incluyendo, pero no limitado a Cocoa, Carbón y Java.

   Otras compañías han añadido soporte para GNU Pascal, Free Pascal, Ada y Perl. Entre las características más apreciadas de Xcode está la tecnología para distribuir el proceso de construcción a partir de código fuente entre varios ordenadores, utilizando Bonjour.

e) **Android Studio**. Es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

   Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS y GNU/Linux.

f) **Microsoft Visual Studio**. Es el entorno de desarrollo integrado oficial para la Plataforma Universal de Windows de Microsoft. El entorno de desarrollo integrado de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación.

   Más allá del editor estándar y el depurador que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.

   Visual Studio está disponible para Windows y Mac. Visual Studio para Mac tiene muchas de las mismas características que Visual Studio y está optimizado para el desarrollo de aplicaciones móviles y multiplataforma.

6. Emuladores

Un emulador es un componente software o programa que permite ejecutar, a su vez, programas en una plataforma diferente de aquella para la cual fueron escritos originalmente. A diferencia de un simulador, que solo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo de manera que éste funcione como si estuviese siendo usado en el equipo original.

Entre los emuladores más destacados en el desarrollo de aplicaciones multiplataforma podemos destacar los siguientes:

1) **Genymotion** dedicado para probar o implementar las aplicaciones de Android. Tiene una excelente calidad de construcción, junto con las características de desarrollador. Puede utilizar la cámara del PC o portátil para videoconferencias con amigos.

   Para desarrolladores está lleno de opciones, probar su sitio web con los diferentes navegadores web para Android y ver la capacidad de respuesta.

   Sus características más relevantes son las siguientes:

   - ✓ Es compatible con todos los SO: Windows, Mac y Linux.
   - ✓ Ventanas de tamaño variable.
   - ✓ API Java soportado.
   - ✓ Equipado con screencasts o grabación digital de la salida por pantalla de la equipo, que puede contener narración de audio.
   - ✓ Un clic con el botón personalizable: IMEI, MEID, ID de Android.
   - ✓ Arrastrar e instalar aplicaciones.
   - ✓ Aceleración OpenGL.
   - ✓ Virtualización de la CPU.

2) **Gorilla Player** es el previsualizador de XAML para Xamarin Forms sin necesidad de compilar el código generado.

   El objetivo principal de Gorilla Player es permitir ver cualquier cambio visual en layouts, estilos o imágenes de forma inmediata sin necesidad de compilar.

   Entre las opciones que ofrece destacan las siguientes:

   - ✓ Add-in para Visual Studio.
   - ✓ Aplicación Player para dispositivos o emuladores.
   - ✓ Posibilidad de ver cualquier cambio de layout, estilos o imágenes al vuelo.
   - ✓ Posibilidad de utilizar datos de prueba.
   - ✓ Posibilidad de sincronizar con múltiples dispositivos de manera simultánea. De esta forma podemos ver el mismo cambio en diferentes dispositivos o plataformas.
   - ✓ Notifica de errores en el marcado (XAML).

3) **BlueStacks** es un destacado emulador de Android para PC que se puede descargar gratis. Actualmente, es usado por más de 130 millones de usuarios.

   Está disponible tanto para máquinas Windows y Mac de forma gratuita. La instalación sencilla de un solo clic y sencilla interfaz de usuario hacen que Bluestacks sea un reproductor de

aplicaciones superior. Para jugar juegos con intensos gráficos, asegúrate de tener una tarjeta gráfica dedicada instalada en tu ordenador.

Sus características más relevantes son las siguientes:

- ✓ 96% de las aplicaciones y los juegos son compatibles.
- ✓ Soporte de pruebas para desarrolladores.
- ✓ Entrada multi-touch
- ✓ Integración del teclado y ratón.
- ✓ Capacidad de sincronización con el escritorio.
- ✓ Soporte gráfico nativo de Windows.
- ✓ Los sensores y la cámara están integrados para una experiencia de Android completa.
- ✓ Soporte de Android-TV.

4) **Remix OS Player** es uno de los emuladores más nuevos de Android para PC. También es el único que funciona con Android 6.0 Marshmallow en lugar Android Lollipop o KitKat. El proceso de instalación es bastante simple y utilizarlo también es bastante sencillo.

   Construido para jugar por lo que tendrá una variedad de opciones a través de la barra lateral para personalizar la experiencia. Es nuevo, por lo que todavía se está resolviendo algunos errores. A pesar de ello, aún funciona mejor que la mayoría y de manera gratuita. La única advertencia es que no es compatible con las CPUs de AMD.

5) **Android x86 – Emulador VirtualBox**. Virtual Box es un software de código abierto que instala los diferentes sistemas operativos como sistema operativo invitado. Desde blog oficial de Android se puede obtener fácilmente la imagen ISO.

   Una vez que haya descargado el archivo ISO se puede instalar fácilmente Android virtualmente en el PC con Virtual Box.

   Características del uso de Android en Virtual Box:

   - ✓ Control por completo de componentes.
   - ✓ Puede realizar todas las acciones como en los mencionados emuladores de Android.
   - ✓ Se pueden asignar memoria RAM para su sistema operativo Android según sus necesidades.
   - ✓ Cada vez que se siente aburrido, puede jugar y probar nuevas aplicaciones.
   - ✓ Es fácil de eliminar / desinstalar el sistema operativo Android en Virtual Box.

6) **Android Emulator**. es el software oficial de Google para que los desarrolladores puedan crear y probar sus aplicaciones y juegos para móviles y tablets.

   Con Android Studio tienes la posibilidad de emular cualquier dispositivo y versión de Android, no solo móviles y tablets, sino también dispositivos con Android Wear y Android TV. Para ello es necesario que definas un dispositivo virtual a través de Android Virtual Device (AVD), que lo puedes ejecutar desde el menú Tools > Android > AVD Manager del menú del programa.

   El emulador de Android Studio es rápido y potente, y pone a tu disposición infinidad de funciones para que pruebes las aplicaciones. Se puede interactuar con el programa de la misma manera que lo harías con un móvil o tablet, pero con el teclado, el ratón y los controles del emulador.

Es compatible con pantallas táctiles y botones de hardware virtuales, incluidas las operaciones con los dedos.

7) **Visual Studio Emulator para Android**. es el software oficial de Microsoft para que los desarrolladores puedan crear y probar sus aplicaciones y juegos para móviles y tablets. El emulador x86 se inicia y se ejecuta casi a la velocidad de un dispositivo físico, facilitando la depuración en aplicaciones con muchos gráficos que requieren un alto rendimiento del procesador.

   Integra la aplicación en entornos de usuario reales. La gama de sensores, incluidos el acelerómetro, la orientación de la pantalla, la tarjeta SD, la batería, la tecnología multitáctil, el GPS, la cámara, el audio y el teclado, ayuda a reducir el tiempo y el gasto de la funcionalidad de depuración en los dispositivos físicos.

   Los perfiles de dispositivo permiten la orientación a una amplia variedad de hardware de Android. Los dispositivos del mercado presentan un variado conjunto de versiones de Android, tamaños de pantalla y otras propiedades de hardware, lo que complica y encarece las pruebas de las aplicaciones. Contiene un generoso conjunto de perfiles de dispositivo que representan al hardware más popular del mercado, incluidos los dispositivos de Samsung, Motorola, Sony, LG, etc.

   Se adapta perfectamente al entorno de desarrollo Android existente, con una instalación de archivos y APK tan fácil como arrastrar y colocar elementos en la pantalla del emulador.

6.1. Configuración de Android Emulator

Android Emulator se puede ejecutar en varias configuraciones para simular múltiples dispositivos. Cada configuración se denomina dispositivo virtual. Al implementar y probar la aplicación en el emulador, debes seleccionar un dispositivo virtual preconfigurado o personalizado que simule un dispositivo Android físico, como un teléfono Nexus o Pixel.

Android Emulator se ejecuta demasiado despacio si la **aceleración de hardware** no está disponible en el equipo que lo ejecuta. Es posible mejorar drásticamente el rendimiento de Android Emulator mediante el uso de imágenes de dispositivos virtuales especiales que tengan como destino el hardware x86 junto con una de las dos tecnologías de virtualización siguientes:

1. Hyper-V y Hypervisor Platform de Microsoft. Hyper-V es una característica de virtualización de Windows que permite ejecutar sistemas de equipos virtualizados en un equipo físico. Se trata de la tecnología de virtualización recomendada para la aceleración de Android Emulator.
2. Hardware Accelerated Execution Manager (HAXM) de Intel. HAXM es un motor de virtualización para los equipos que ejecutan CPU de Intel. Se trata del motor de virtualización recomendado para los equipos que no pueden ejecutar Hyper-V.

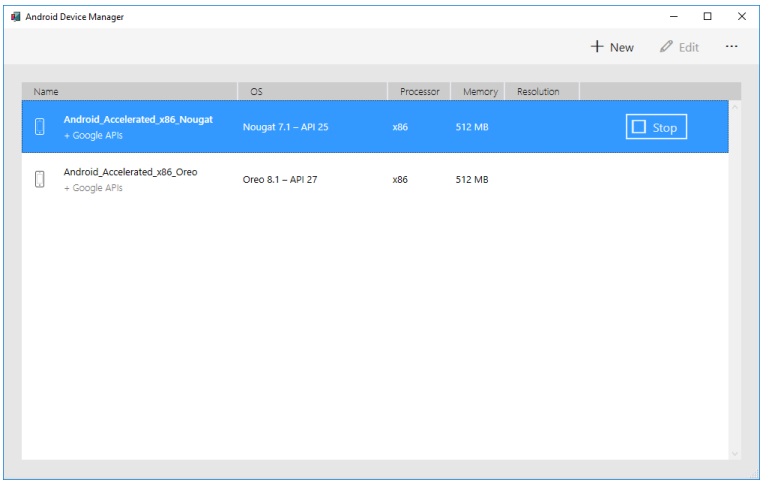
Android Emulator usará automáticamente la aceleración de hardware si se cumplen los criterios siguientes:

a. La aceleración de hardware está disponible y habilitada en el equipo de desarrollo.
b. El emulador ejecuta una imagen del emulador creada específicamente para un dispositivo virtual basado en x86.

Después de comprobar que la aceleración de hardware está habilitada, el siguiente paso es usar **Android Device Manager** para crear dispositivos virtuales que se puedan usar para probar y depurar la aplicación.

Android Device Manager se usa para crear y configurar dispositivos virtuales Android (AVD) que se ejecutan en Android Emulator. Cada AVD es una configuración del emulador que simula un dispositivo Android físico. Esto permite ejecutar y probar la aplicación en diversas configuraciones que simulan diferentes dispositivos Android físicos.

Inicia Android Device Manager desde el menú Herramientas. Para ello, haga clic en Herramientas > Android > Android Device Manager:

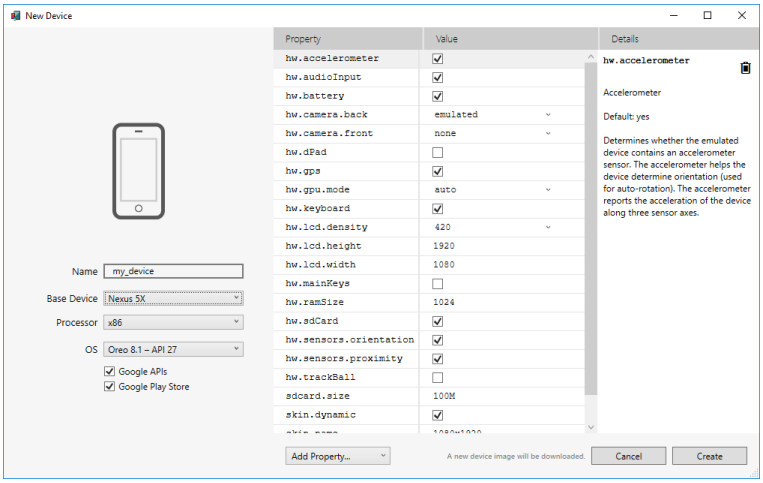


Cuando inicies el Administrador de dispositivos Android por primera vez, aparecerá una pantalla con todos los dispositivos virtuales configurados. Para cada dispositivo, se muestran el nombre, el sistema operativo (nivel de API de Android), la CPU, el tamaño de memoria y la resolución de pantalla:

Al hacer clic en un dispositivo de la lista, aparece el botón Iniciar (Start) a la derecha. Puedes hacer clic en el botón Iniciar para iniciar el emulador con este dispositivo virtual:

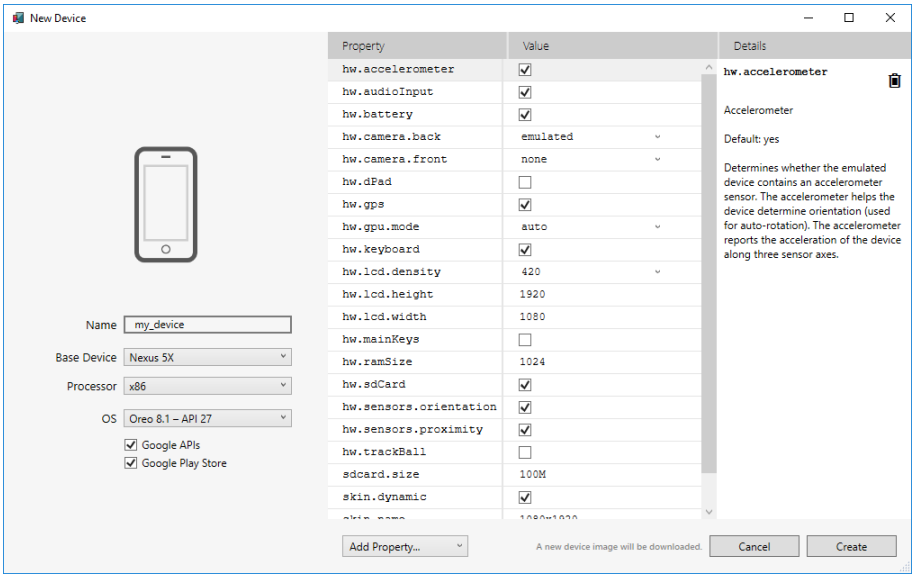
Una vez que el emulador se haya iniciado con el dispositivo virtual seleccionado, el botón Iniciar cambiará al botón Detener (Stop), que puede usar para detener el emulador.

Para crear un dispositivo, haz clic en el botón Nuevo (New) situado en la parte superior derecha de la pantalla:



6.2. Editar las propiedades de perfil de un dispositivo virtual Android

Android Device Manager es compatible con la edición de propiedades de perfil de dispositivos virtuales Android individuales. En las pantallas Nuevo dispositivo y Edición de dispositivos se muestran las propiedades del dispositivo virtual en la primera columna, con los valores correspondientes de cada propiedad en la segunda columna, como se ve en este ejemplo:



Cuando se selecciona una propiedad, aparece una descripción detallada de dicha propiedad a la derecha.

Puedes modificar las propiedades de perfil de hardware y las propiedades de AVD. Las propiedades de perfil de hardware (hw.ramSize y hw.accelerometer) describen las características físicas del dispositivo emulado. Estas características incluyen el tamaño de pantalla, la cantidad de memoria RAM disponible y si hay un acelerómetro o no.

Las propiedades de AVD especifican el funcionamiento del AVD durante su ejecución. Por ejemplo, se pueden configurar las propiedades de AVD para especificar la forma en que el AVD usa la tarjeta gráfica del equipo de desarrollo para la representación.

6.3. Simulador remoto de iOS para Windows

El simulador remoto de iOS para Windows nos permite probar nuestras aplicaciones en un emulador de iOS sobre Windows utilizando Visual Studio.

Esta aplicación se instala de forma automática como una parte de Xamarin en Visual Studio. Para utilizarla es necesario seguir los siguientes pasos:

1. Emparejar Visual con un equipo Mac.
2. En Visual Studio, iniciar la depuración del proyecto sobre iOS. El simulador remoto de iOS para Windows mostrará la ventana correspondiente en nuestro equipo con Windows.

6.4. Configurar el dispositivo para el desarrollo

En este apartado vamos a ver cómo configurar un dispositivo Android real y cómo conectarlo a un equipo de modo que el dispositivo pueda usarse para ejecutar y depurar aplicaciones utilizando Microsoft Visual Studio.

A estas alturas, probablemente ya has ejecutado tu nueva aplicación en el emulador de Android y quieres ver cómo se ejecuta en su dispositivo Android. Estos son los pasos que debes llevar a cabo para conectar un dispositivo a un equipo para la depuración:

1. **Habilita la depuración en el dispositivo**: de forma predeterminada, no es posible depurar aplicaciones en un dispositivo Android.

   A partir de Android 4.2 y versiones posteriores, las Opciones del desarrollador están ocultas de forma predeterminada. Para que estén disponibles, ve a Ajustes > Acerca del dispositivo y pulsa el elemento Número de compilación siete veces para que se muestre la pestaña Opciones de desarrollador.

   Una vez que la pestaña Opciones del desarrollador está disponible en Ajustes > Sistema, ábrela para mostrar las opciones del desarrollador. Desde aquí puedes habilitar las opciones del desarrollador, como la depuración de USB y el modo Permanecer activo.

2. **Instala controladores USB**: este paso no es necesario para los equipos OS X. Los equipos de Windows podrían requerir la instalación de controladores USB.
3. **Conecta el dispositivo al equipo**: el último paso consiste en conectar el dispositivo al equipo mediante USB o WiFi:
   a. USB: esta es la manera más fácil y más común. Basta con que conectes el cable USB al dispositivo y, después, al equipo.
   b. WiFi: es posible conectar un dispositivo Android a un equipo a través de WiFi, sin usar un cable USB. Esta técnica requiere un mayor esfuerzo, pero puede resultar útil si no tiene un cable USB o si el dispositivo se encuentra demasiado lejos para usar un cable USB.

6.5. Xamarin Live Player

Xamarin Live Player es un emulador que ayuda a comenzar a trabajar con el desarrollo de aplicaciones multiplataforma con Xamarin, de manera que se pueden crear y probar aplicaciones móviles en un dispositivo real con **Android** e **iOS**.

Esta aplicación permite modificar el código de la aplicación y reflejar los cambios en el dispositivo en directo. El código se ejecuta dentro de la aplicación Xamarin Live Player, por lo que no es necesario configurar ningún emulador ni usar ningún cable para implementar su aplicación.

Los pasos para configurar este emulador son los siguientes:

1) Instala la aplicación Xamarin Live Player en el dispositivo real con Android disponible mediante Google Play.
2) Inicia la aplicación Xamarin Live Player en el dispositivo real.
3) Conecta el dispositivo real con Android mediante un cable USB al equipo con Visual Studio.
4) Inicia Visual Studio.
5) Sólo en el caso del primer uso de Xamarin Live Player debes seguir estos pasos:
    a) Clic en Herramientas > Opciones...
    b) Clic en Xamarin > otros.
    c) Selecciona Habilitar Xamarin Live Player.
    d) Clic en Aceptar.
6) Crea o abre un proyecto de Xamarin.
7) Selecciona Live Player en la lista de dispositivos.
8) Si ya has emparejado un dispositivo real, éste estará disponible como una opción, en caso contrario, empareja el dispositivo escaneando el código QR mostrado en la pantalla del equipo con Visual Studio con la aplicación instalada en el dispositivo real.
9) Clic en el botón Ejecutar o selecciona una de las siguientes opciones del menú Ejecutar o del menú contextual:
    a) Iniciar sin depurar: puede editar la aplicación y ver los cambios que se producen en el dispositivo, la aplicación se reinicia cuando se realizan cambios y guarda el archivo.
    b) Iniciar depuración: puede establecer puntos de interrupción e inspeccionar las variables, pero no se puede editar el código.

    Como alternativa, selecciona Herramientas > Xamarin Live Player > Ejecutar vista actual en vivo, lo cual permite editar la aplicación y ver los cambios que se producen en el dispositivo. Se muestra la vista actual en lugar de la pantalla principal de la aplicación.

6.6. Xamarin Hot Reload

El objetivo principal de **Xamarin Hot Reload** es permitir ver cualquier cambio relacionado con la interfaz de usuario de forma rápida y sencilla, soportando cambios al vuelo sin necesidad de compilar y desplegar. Cualquier cambio en XAML será reflejado de forma automática manteniendo los cambios.

La recarga activa de XAML se conecta al flujo de trabajo existente para aumentar su productividad y ahorrar tiempo. Sin la recarga activa de XAML, debe compilar e implementar la aplicación cada vez que quiera ver un cambio de XAML. Con la recarga activa, al guardar el archivo XAML, los cambios se reflejan en directo en la aplicación en ejecución. Además, se mantendrán el estado y los datos de navegación, lo que le permitirá iterar rápidamente en la interfaz de usuario sin perder su lugar en la aplicación.

Por lo tanto, con la recarga activa de XAML, pasará menos tiempo recompilando e implementando sus aplicaciones para validar los cambios de la interfaz de usuario.

Requisitos del sistema:

- ✓ Visual Studio 2019 16,3 o superior
- ✓ Visual Studio 2019 para Mac 8,3 o superior
- ✓ Xamarin.Forms 4,1 o superior

No se requiere ninguna instalación o configuración adicional para usar la recarga activa de XAML. Está integrado en Visual Studio y puede habilitarse en la configuración del IDE. Una vez habilitada, puede empezar a usar la recarga activa de XAML mediante la depuración de la aplicación en un emulador, un simulador o un dispositivo físico. Actualmente, la recarga activa de XAML solo funciona durante la depuración en iOS o Android.

En Windows, la recarga en caliente de XAML se puede habilitar activando la casilla Habilitar la recarga activa de Xamarin en herramientas > Opciones > Xamarin > Hot Reload.

Si se realiza un cambio que no se puede volver a cargar la recarga activa de XAML, se mostrará un mensaje de error con IntelliSense. Estos cambios, conocidos como ediciones forzadas, incluyen escribir indirectamente el código XAML o conectar un control a un controlador de eventos que no existe. Incluso con una edición forzada, se puede seguir recargando sin reiniciar la aplicación: realiza otro cambio en otra parte del archivo XAML y presiona guardar. La edición forzada no se recargará, pero se seguirán aplicando los demás cambios.

7. Clasificación de los dispositivos móviles

Si pensamos en dispositivos móviles, lo primero que nos viene a la cabeza es un teléfono móvil. Pero en la actualidad son varios los dispositivos móviles disponibles en el mercado: Smartphones, PC portátiles, PocketPC, Tabletas, etc.

Esta diversidad comporta una importante problemática para quien debe programarlos, ya que cada uno tiene unas características particulares: dispone de una memoria determinada o ha de soportar un lenguaje y un entorno específicos.

Los nuevos dispositivos nos aportan mucha información sobre nuestro entorno. Por ejemplo, la más clara y conocida es la posición geográfica actual, que nos permite realizar aplicaciones basadas en la localización. Además, existen otras informaciones como, por ejemplo, la orientación, la presión, la luz, etc. La posibilidad de grabar imágenes, vídeos y audio también nos aporta más información sobre el contexto, las aplicaciones que reaccionan al habla o las de realidad aumentada son ejemplos de aplicaciones que aprovechan este tipo de contexto. Existen capacidades más obvias como la de saber la hora actual, el idioma, o la zona horaria que ayudan a realizar aplicaciones más contextualizadas. Otro aspecto que ha hecho mejorar la situación es, sin duda, la mejora de las comunicaciones inalámbricas a partir de la aparición del 4G, del 5G o del UMTS, o la mejoras de Bluetooth o nuevos estándares de WiFi.

Muchas de las aplicaciones aprovechan varias capacidades. Por ejemplo, una aplicación de realidad aumentada aprovecha varias de ellas:

- ✓ GPS, para conocer la posición geográfica y saber qué se debe mostrar.
- ✓ Brújula, para saber la orientación actual
- ✓ Acelerómetro, para saber cuál es la orientación exacta de nuestro dispositivo y superponer las capas.
- ✓ Cámara, para poder captar nuestro alrededor y así ampliar la información (en ocasiones, incluso, varias cámaras).

- ✓ Conexión a Internet, para poder obtener la información para ampliar nuestra realidad. Esta conectividad puede venir mediante Internet móvil o WiFi, entre otros.
- ✓ Capacidad de procesamiento gráfico muy mejorada, con chips de aceleración gráfica potentes.

Además, están apareciendo nuevos protocolos o capacidades que ayudarán a mejorar las aplicaciones móviles permitiendo realizar pagos con el móvil, como es el caso de NFC (Comunicación de Campo Cercano) que es una tecnología de comunicación inalámbrica de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de comunicación y formatos de intercambio de datos, y están basados en ISO 14443 (RFID, radio-frequency identification) y FeliCa. Los estándares incluyen ISO/IEC 18092 y los definidos por el Foro NFC (NFC Forum), fundado en 2004, por Nokia, Philips y Sony, y que hoy suma más de 170 miembros.

Atendiendo a la utilidad para el usuario del propio aparato, se podría proponer las siguientes categorías de dispositivos móviles:

a. **Trabajo**. Dentro de este campo, están las tablets y teléfonos inteligentes de mayor avance operativo, los mismos se caracterizan por el desarrollo generalizado de actividades como también de concreción de funciones. Éstas se constituyen como el medio de trabajo de una gran cantidad de personas, donde por medio de los mismos, se lleva a cabo un total de actividades que pueden generar gran productividad.

b. **Entretenimiento**. Dentro de estos se incluyen todos y cada uno de los equipos que sirven para brindar una gran diversión a las personas, bien sea por medio de la activación y desarrollo de juegos.

c. **Control**. Son aquellos que permiten de una forma u otra potenciar las comunicaciones, en efecto, estamos en presencia de teléfonos que resultan útiles para desarrollar todos y cada uno de los sistemas operativos y comunicativos a través de diversas aplicaciones.

d. **Especializados**. Incluye una amplia gama de equipos entre los cuales se pueden presentar desde teléfonos, relojes hasta los propios cajeros automáticos. Todos estos equipos permiten y facilitan el intercambio de información. Los más predilectos de todos y que forman parte de los más avanzados sistemas vienen conformados por los equipos con inteligencia artificial empleado en las instituciones, oficinas o bien en las zonas de acceso educativo. Estos equipos presentan un alto grado tecnológico pero más allá de ello, los mismos representan un alto nivel de ayuda para las personas, al poder sincronizar en sí mismos una serie de funciones que permiten llevar a cabo un sinfín de actividades.

Atendiendo a otra clasificación, a grandes rasgos, y dependiendo del tamaño los dispositivos móviles se pueden dividir en las siguientes clases:

a. **Teléfonos**. Son los más pequeños de la casa, y por tanto los más ligeros y más transportables. En general, también son los más baratos, aunque un teléfono de gama alta puede superar en precio a muchos de sus hermanos mayores, las PDAs. Su función primordial era clara: recibir y realizar llamadas; aunque parece que dentro de poco va a comenzar a ser complicado encontrar teléfono que sirvan para eso. Funcionalidades propias de ordenadores, o de dispositivos de otro tipo, como la grabación y edición de vídeo, realización de fotografías, lectura de documentos, localización en mapas, navegación por Internet, y muchas cosas más, son no sólo habituales, sino esperadas en cualquier teléfono moderno.

b. **PDAs**. Son organizadores electrónicos u ordenadores de mano. Su nombre (PDA) significa Personal Digital Assistant (asistente personal digital), un término acuñado en sus primeros años de historia, pero que resume bien su funcionalidad principal, que es servir como organizadores, con agenda, calendario, gestión de contactos, y que posteriormente han ido creciendo, de forma que actualmente sirven tanto como aparatos en los que leer un libro como en los que encontrarse en un mapa. La línea que los separa de los teléfonos es cada vez más difusa.

c. **Consolas**. En realidad esta categoría debería llamarse "dispositivos orientados a jugar", porque son más que simples consolas. Los ejemplos actualmente en el mercado son la PlayStation de Sony, la Xbox de Microsoft y la Switch de Nintendo, que no sólo sirven para jugar, sino que integran algunas de las funcionalidades típicas de una PDA, como reproducción de archivos multimedia, integración con agenda y calendario, o navegador de Internet.

d. **Wareables**. Hacen referencia al conjunto de equipos y dispositivos electrónicos que se incorporan en alguna parte de nuestro cuerpo interactuando de forma continua con el usuario y con otros dispositivos con la finalidad de realizar alguna función concreta. Relojes inteligentes o smartwatchs, zapatillas de deportes con GPS incorporado y pulseras que controlan nuestro estado de salud son ejemplos entre otros muchos de este género tecnológico que se halla poco a poco más presente en nuestras vidas.

8. Relación entre el dispositivo y la aplicación

Hay dos limitaciones fundamentales a tener en cuenta a la hora de desarrollar una aplicación para ser ejecutada en un dispositivo móvil y son:

1. Las particularidades de hardware.
2. Las particularidades de la conexión.

Aunque los dispositivos a nivel del **hardware** cada vez tienen más memoria y más capacidad de procesamiento, siguen siendo hermanos pequeños de los ordenadores de escritorio. Por tanto, debe tenerse especial cuidado con no realizar demasiadas animaciones, ni gastar ciclos de proceso en procedimientos que no sean estrictamente necesarios. La pantalla es pequeña, mal iluminada, y se suele mirar en movimiento, como ya hemos visto con anterioridad, por lo que podemos aprovechar eso en nuestro beneficio.

Por ello la **entrada de datos** debe ser lo más sencilla posible. En el caso de las PDAs, se realizará escribiendo en la pantalla con un lápiz, para que sea el dispositivo el que realice el reconocimiento de la escritura. Aunque ya no hace falta escribir de forma especial, como ocurría con el sistema de reconocimiento de escritura de las primeras Pilot, llamado Graffiti, para que el dispositivo entienda la letra, tampoco se puede ser demasiado descuidado con la caligrafía.

En los móviles, sin embargo, la entrada de datos se va a hacer, lo más probablemente, con una sola mano, y en condiciones no muy apropiadas. El teclado, en la mayoría de los casos, tiene sólo 10 teclas útiles, que se van a utilizar en movimiento.

En cualquiera de los dos casos, hay que intentar, por tanto, formatear las pantallas de entrada de datos para que se pueda realizar el mayor número de operaciones o bien punteando encima de botones, en el caso de las PDAs, o bien navegando con el joystick y confirmando con alguno de los botones principales en el caso de los teléfonos; evitando en lo posible que el usuario tenga que dirigirse al teclado. Siempre es preferible que haya, por ejemplo, un desplegable en el que elegir entre

"Calle", "Vía", "Avenida" y "Paseo", a que haya un campo de texto en el que se tenga que escribir, letra a letra, "Calle".

Además, no hay que olvidar que los interfaces de los distintos dispositivos son desesperantemente heterogéneos. Además, ni el tamaño de las pantallas, ni el tamaño ni la ubicación de las teclas de ayuda a la hora de introducir datos por el usuario. Por eso, siempre hay que intentar desarrollar los interfaces de las aplicaciones de forma que no sean necesarias muchas pulsaciones de teclas, e intentar que, cuando sea necesario pulsar alguna tecla sea alguna de las softkeys, o teclas especiales. En el caso de los dispositivos móviles, más que en cualquier otro, menos es más.

No sólo menos es más, sino que la colocación de los elementos de ese menos son los que van a marcar la diferencia. En realidad, no es sólo una cuestión de colocación en pantalla, sino de organización de la información a presentar, de lo que se llama arquitectura de la información.

La premisa es bastante sencilla: dadas las dificultades para la introducción de datos, hay que intentar simplificarla al máximo. Para ello, es muy importante organizar el interfaz de la aplicación de forma que los procesos de entrada de datos sean lo más naturales posibles, y que el usuario nunca pase por pantallas cuya presentación no sea estrictamente necesaria.

Pero en realidad esas precauciones no hay que tomarlas sólo a la hora de introducir datos, sino también a la hora de presentar información, y a la hora de estructurar la navegación por la aplicación.

Normalmente es mucho más efectivo presentar la información agrupada de forma jerárquica, para de ese modo discriminar cuanto antes qué es lo que va a necesitar ver el usuario, y no mostrarle nada que no le resulte estrictamente necesario.

En cuanto a las **conexiones**, lo primero que hay que tener siempre en cuenta es que, al contrario que en las aplicaciones web estándar, no están siempre disponibles. Por tanto, nunca se puede suponer que el dato que es necesario para poder seguir adelante en la ejecución de la aplicación, va a poderse obtener a través de una conexión a Internet.

Además, debe tenerse en cuenta que antes de realizar una conexión a cualquier servicio tarificable, el dispositivo va a solicitar confirmación al usuario, y que éste puede denegarla, lo cual es perfectamente posible, dado el precio de las conexiones, que son facturables por peso, por cantidad de bytes enviados y descargados, no por tiempo, por lo que siempre hay que optimizar todo lo posible lo que se vaya a enviar en uno u otro sentido

Por si no fueran dificultades suficientes, la latencia de la red, es decir, el tiempo de espera entre una petición de datos y el comienzo de la llegada de la respuesta, es mucho mayor que la de cualquier aplicación web normal. Por tanto, hay que ser muy cuidadosos a la hora de informar al usuario sobre lo que está ocurriendo en todo momento, si se está en espera de datos, si ya se están recibiendo, dando siempre, si es posible, estimaciones sobre el tiempo que resta para la terminación de todas las tareas.

Tampoco se puede suponer que la ejecución de la aplicación sea lineal y completa. La función básica de muchos dispositivos es emitir y recibir llamadas telefónicas. Por tanto, nuestra aplicación debe ser capaz de comportarse adecuadamente al recibir una llamada, entre otras cosas porque la función de teléfono tiene asignada la prioridad más alta de todas las del dispositivo, es decir, se va a parar la ejecución de cualquier aplicación cuando entre una llamada nueva. Se debe tener prevista una forma de congelar el estado de la aplicación justo antes de recibir la llamada, para que cuando ésta termine, y tras la confirmación del usuario, se pueda volver a ese estado. Por tanto, si es un juego, todos los elementos del mismo deben permanecer estáticos hasta que la llamada termine y el usuario confirme que desea continuar con la aplicación.

En general, hay que intentar probar las aplicaciones en el dispositivo para el que se vayan a desarrollar cuanto antes. De hecho, hay que probar mucho, y comenzar a probar lo antes posible.

Nunca hay que fiarse de emuladores, entre otras cosas, porque se ejecutan en hardware cuya capacidad de procesamiento es superior a la del dispositivo en varios órdenes de magnitud.

Por tanto, al desarrollar para dispositivos móviles, es muy importante comenzar a probar pronto, y probar a menudo, a fin de evitar sorpresas desagradables.

La **generación de perfiles** y las **herramientas de diagnóstico** ayudan a diagnosticar el uso de CPU, memoria y otros problemas a nivel de aplicación. Con estas herramientas, puede acumular datos (como valores de variables, llamadas a funciones y eventos) durante el tiempo que se ejecute la aplicación en el depurador. Se puede ver el estado de la aplicación en distintos momentos de la ejecución del código.

Las herramientas de generación de perfiles de Visual Studio proporcionan varios mecanismos para recopilar y analizar los datos de rendimiento. Sin embargo, en muchos casos, el mejor modo de comenzar es usar los valores predeterminados del Asistente para rendimiento. Con el Asistente para rendimiento, se pueden recopilar estadísticas de la aplicación que pueden indicar los problemas de rendimiento del código mediante:

- ✓ Las **advertencias de rendimiento** notifican los problemas de codificación más comunes. Las advertencias se muestran en la ventana Errores de Visual Studio. Se puede navegar desde la advertencia al código fuente y a temas detallados de la Ayuda que ayudarán a escribir código más eficaz.
- ✓ Los **informes de las herramientas de generación de perfiles** proporcionan vistas de los diferentes niveles de la estructura de la aplicación, desde las líneas de código fuente hasta los procesos, y de los datos de ejecución del programa, desde las funciones que realizan o reciben la llamada de una determinada función hasta el árbol de llamadas de toda la aplicación.

Las herramientas de **generación de perfiles** de Visual Studio 2017 proporcionan cinco métodos para recopilar datos de rendimiento:

1. **Muestreo**. Recopila datos estadísticos sobre el trabajo realizado por una aplicación.
2. **Instrumentación**. Recopila información de tiempo detallada sobre cada llamada a una función.
3. **Simultaneidad**. Recopila información detallada sobre las aplicaciones multiproceso.
4. **Memoria de .NET**. Recopila información detallada sobre la asignación de memoria de .NET y la recolección de elementos no utilizados.
5. **Interacción de capas**. Recopila información sobre las llamadas de funciones ADO.NET sincrónicas a una base de datos de SQL Server.

Dentro del entorno de desarrollo integrado Visual Studio Community 2017, se puede obtener acceso a las herramientas de generación de perfiles mediante *Depurar* > *Windows* > *Mostrar herramientas de diagnóstico* para usar las herramientas durante la sesión de depuración, o mediante *Depurar* > *Generador de perfiles de rendimiento...* para realizar un análisis de rendimiento preciso.

Las diferentes **herramientas de rendimiento** que están disponibles en el entorno de desarrollo integrado Visual Studio Community 2017 son las siguientes:

a. **Uso de memoria**. Busca pérdidas de memoria y memoria ineficaz durante la depuración de la aplicación.

b. **Uso de CPU**. Muestra el tiempo que la CPU dedica a ejecutar el código C#.

c. **Explorador de rendimiento**. Permite usar muchas herramientas diferentes, incluidas Muestreo de la CPU, Instrumentación, Asignación de memoria de .NET, y Contención de recursos.

d. **Uso de GPU**. Ayuda a comprender mejor el uso de hardware de alto nivel de la aplicación Direct3D.

e. **Escala de tiempo de la aplicación**. Ayuda a mejorar el rendimiento de las aplicaciones XAML proporcionando una vista detallada del consumo de recursos.

f. **Sugerencias de rendimiento**. Cuando el depurador detiene la ejecución en un punto de interrupción o en una operación de ejecución paso a paso, el tiempo transcurrido entre la interrupción y el punto de interrupción anterior aparece como una sugerencia en la ventana del editor.

g. **IntelliTrace**. Permite registrar eventos específicos, examinar los datos en la ventana Locales durante los eventos de depurador y llamadas a funciones, y depurar errores que son difíciles de reproducir. Se trata principalmente de una herramienta de depuración, pero también proporciona información que puede usar para investigaciones de rendimiento. Solo se puede usar esta herramienta en Visual Studio Enterprise.

h. **Generación de perfiles en producción**. El método recomendado para la generación de perfiles en producción es generar el perfil desde la línea de comandos mediante vsperf.exe para recopilar un perfil de la CPU.