

Don't Panic

GUÍA A LA GALAXIA DE APLICACIONES MÓVILES

PRIMERA VERSIÓN EN ESPAÑOL

GRATIS

12^a
EDICIÓN



publicada por:



Servicios y Herramientas para Todas las Plataformas Móviles

Enough Software GmbH + Co. KG
Sögestrasse 70
28195 Bremen
Alemania
www.enough.de

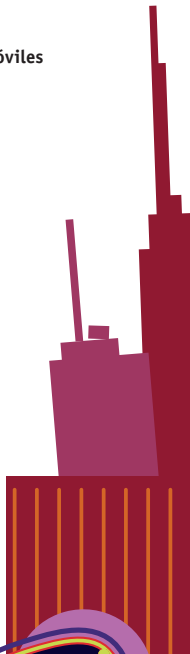
Por favor, envía tu feedback, preguntas
o solicitudes de patrocinio a:
developers@enough.de
Síguenos en Twitter: [@enoughsoftware](https://twitter.com/enoughsoftware)

12ª Edición Febrero 2013

Esta Guía se publica bajo Licencia
Creative Commons Algunos Derechos Reservados.

Dirección Artística y Diseño por
Andrej Balaz
(Enough Software)

Traducción realizada con la colaboración de



Guía a la Galaxía de Aplicaciones Móviles

Contenidos

I **Prólogo**

1 **La Galaxia Móvil: Introducción**

1 Topología: Factores de Formato y Patrones de Uso

2 Formación Estelar: Creando un Servicio Móvil

6 El Universo de los Sistemas Operativos Móviles

13 Sobre Tiempo y Espacio

13 Perdido en el Espacio

15 **Diseño Conceptual para Móviles**

15 Capturando la idea

17 Diseñando la Experiencia de Usuario

23 **Android**

23 El Ecosistema

25 Prerrequisitos

26 Implementación

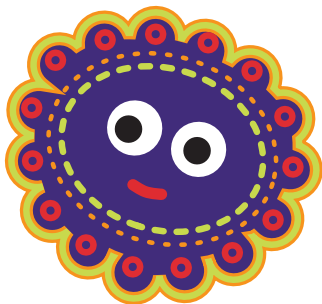
29 Testeo

31 Compilación

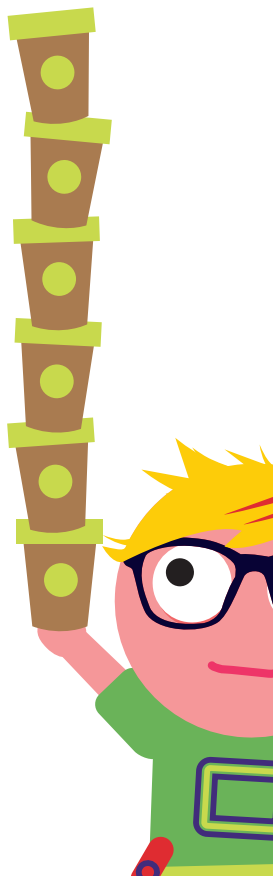
32 Firma

32 Distribución

33 Monetización



35	BlackBerry: Aplicaciones Java
35	El Ecosistema
37	Prerrequisitos
37	Implementación
40	Testeo
41	Firma
41	Distribución
42	Aprende Más
44	BlackBerry 10
44	El Ecosistema
45	Desarrollo
53	Testeo
53	Firma
54	Distribución
56	iOS
56	El Ecosistema
57	Descripción General de la Tecnología
60	Testeo & Debugging
61	Aprende Más
64	Java ME (J2ME)
64	El Ecosistema
65	Prerrequisitos
66	Implementación
70	Testeo
71	Portar
73	Firma
75	Distribución
76	Aprende Más



79	Windows Phone
79	El Ecosistema
80	Implementación
86	Testeo y Analíticas
87	Distribución
88	Aprende Más
91	Windows 8
91	Prerrequisitos
92	Implementación
97	Distribución
98	Aprende Más
100	Hacia Multiplataforma
100	Diferencias Clave Entre Plataformas Móviles
106	Estrategias Multiplataforma
110	Frameworks de Aplicaciones Multiplataforma
115	Motores de Juego Multiplataforma
118	Tecnologías Web
120	HTML5
122	La Fragmentación Requiere Adaptación
126	Testear Tecnologías Web
128	Aprende Más
130	Accesibilidad
132	Aplicaciones Android Accesibles
133	Aplicaciones BlackBerry Accesibles
134	Aplicaciones iOS Accesibles
135	Aplicaciones Symbian/Qt Accesibles
135	Aplicaciones Windows Phone y Windows 8 Accesibles
136	Aplicaciones Web Móvil Accesibles

138	Aplicaciones Corporativas: Estrategia Y Desarrollo
139	Estrategia
141	Gestión de Dispositivos y Aplicaciones en la Empresa
144	Plataformas de Aplicaciones Móviles Corporativas (MEAPs)
145	Seguridad En Aplicaciones Corporativas
148	Analíticas Móviles
148	Poniéndote En Marcha
150	Decidir Qué Medir
150	Definir Cómo Medir
152	Adaptando Tu Código
153	Gestionando Los Resultados
153	Privacidad
155	Aprende Más
157	Implementando Rich Media
158	Streaming versus Almacenamiento Local
160	Descarga Progresiva
161	Conversores de Medios
163	Implementando Servicios Basados en Localización
164	Cómo Obtener Datos de Posición
166	Servicios de Mapas
167	Implementando Soporte a Localización en Diferentes Plataformas
168	Herramientas Para Aplicaciones LBS
171	Comunicaciones de Campo Cercano
173	Modos de operación NFC
175	Usos actuales de NFC

176	Implementación de NFC
178	Implementando Vibración Háptica
178	Consideraciones de Diseño de Vibración Háptica
180	iOS
181	Android
183	BlackBerry 10
183	Windows 8
185	Implementando Realidad Aumentada
186	Escenarios de Uso de RA en Móviles
187	Seguimiento
190	SDKs de Realidad Aumentada
193	Introducción al Desarrollo de RA
198	Seguridad De La Aplicación
200	Amenazas a Tus Aplicaciones
201	Protegiendo Tu Aplicación
205	Mejores Prácticas
206	Herramientas
207	Aprende Más
209	Conclusiones
211	Testeando Tu Aplicación
211	Testeabilidad: El Caballo Ganador
212	Desarrollo Guiado por Pruebas
214	Pruebas Unitarias
215	Testear a Través de Las Cinco Fases del Ciclo de Vida de una Aplicación

220	Testeo Interactivo
223	Automatización de Pruebas
227	Cuidado Con Las Especificaciones
229	Monetización
230	Pago Por Descarga
232	Pagos En Aplicación
234	Publicidad Móvil
235	Participación en Ingresos
236	Indirect Sales
236	Mercado de Componentes
239	Eligiendo tu Modelo de Monetización
240	Tiendas de aplicaciones
244	¿Qué Puedes Ganar?
245	Aprende Más
247	Epílogo
248	Acerca de los Autores

Prólogo

Han pasado exactamente cuatro años desde que se publicó la primera versión de esta guía. En el prólogo de aquella primera edición, dijimos: "Los problemas más importantes para los desarrolladores móviles son sin duda la fragmentación y la distribución". Aunque el paisaje del ecosistema móvil ha experimentado grandes cambios desde entonces, continuamos pensando que esto sigue siendo cierto. Y el paisaje continúa cambiando: probablemente, esta 12ª edición introduce más cambios que las anteriores. Para reflejar los cambios recientes en el mercado, hemos decidido eliminar los capítulos dedicados a Qt y bada y reemplazarlos por nuevos capítulos independientes de plataforma, que cubren tecnologías clave como realidad aumentada, analíticas y prototipado. Además, el capítulo de introducción general ha sido ampliado de manera significativa, y ahora ofrece una exhaustiva visión actualizada sobre el siempre cambiante mundo móvil. Por último, todos los demás capítulos también se han revisado y actualizado como de costumbre. Es casi imposible encontrar una guía más objetiva y actualizada que enseñe cómo empezar como proveedor de aplicaciones o desarrollador, y cómo llevar la oferta móvil existente al siguiente nivel.

También estamos muy contentos de que sea la primera edición disponible en castellano. Muchas gracias por ello a Javier y Eva, de la Universitat Oberta de Catalunya.

¿Qué ocurrió en la industria móvil desde nuestra última edición? Apple lanzó el iPhone 5 con gran éxito, Android se convirtió en el líder mundial indiscutible en sistemas smart-phone, y Microsoft lanzó Windows 8 y Windows Phone 8.

Mark Zuckerberg criticó el enfoque HTML5 del propio Facebook y se encontró con un incremento masivo de su uso tras dar un giro hacia aplicaciones nativas; luego, Canonical presentó Ubuntu para móviles y, por último, BlackBerry nos ha sorprendido a los desarrolladores con canciones de amor y el lanzamiento de BlackBerry OS 10.

A partir de ahora, vamos a establecer un ciclo de publicación semestral. Así que esperamos publicar la 13ª edición alrededor de septiembre de 2013. Si consideráis que le falta contenido, queréis participar de alguna manera, o sabéis de posibles patrocinadores para éste proyecto: Poneos en contacto a través de developers@enough.de. Éste es un proyecto abierto y comunitario e invitamos a todos a impulsarlo para llevarlo aún más lejos.

Estamos deseando saber de vosotros!

Robert + Marco / Enough Software
Bremen, Febrero de 2013



La Galaxia Móvil: Introducción

Bienvenido al mundo del desarrollo móvil, un lugar donde antiguos gigantes se tambalean y nacen nuevas estrellas, aparentemente de forma habitual.

Este libro se centra en el desarrollo de aplicaciones móviles, lo que incluye una serie de fases tales como: planificación y especificación, prototipado y diseño, implementación, testeo interno y despliegue, publicación en una tienda de aplicaciones, descubrimiento por los usuarios, instalación, uso y feedback. En última instancia, queremos que nuestros usuarios disfruten utilizando nuestras aplicaciones y que nos den valoraciones positivas para alentar a otros a hacer lo mismo.

Sigue leyendo para aprender cómo desarrollar aplicaciones para las principales plataformas. Si ésta es la primera vez que has considerado involucrarte en este tema, te recomendamos no demorarlo más ya que el mundo se está enfocando rápidamente hacia el móvil como forma predominante en la informática, y es probable que otros te adelanten si esperas demasiado.

Aunque el desarrollo de aplicaciones móviles tiene algunos puntos en común con el de otro tipo de software, también posee características específicas. Vamos a tratar algunas de ellas a continuación.

Topología: Factores de Formato y Patrones de Uso

Hay que diferenciar entre smartphones, tablets y teléfonos de gama media (en inglés feature phones). Cada factor de forma plantea sus propios retos de usabilidad; por ejemplo, un tablet

exige una navegación diferente a un teléfono. Además, los sistemas de televisión resultan cada vez más atractivos para los desarrolladores móviles como nuevo factor de forma.

Naturalmente, los patrones de uso de Android difieren de los de iOS, que difieren a su vez de los de aplicaciones para Windows Phone, etcétera.

Debes, por tanto, abstenerte de proporcionar la misma experiencia en todos los formatos, o incluso en todos los sistemas en los que funcionará tu aplicación. De lo contrario, corres el riesgo de ofrecer un servicio mediocre a algunos segmentos de tus usuarios.

Formación Estelar: Creando un Servicio Móvil

Hay varias maneras de producir un servicio móvil:

- Aplicación (app)
- Sitio Web
- SMS, USSD¹ y STK²

Aplicaciones

Las aplicaciones se ejecutan directamente en el dispositivo. Se pueden desarrollar como nativas, web o híbridas.

¹ es.wikipedia.org/wiki/USSD

² en.wikipedia.org/wiki/SIM_Application_Toolkit

Aplicaciones Nativas

Una aplicación nativa está programada en un lenguaje específico con APIs propias de la plataforma. Se suele comprar, descargar y actualizar a través de la tienda de aplicaciones específica de la plataforma. Las aplicaciones nativas suelen ofrecer mejor rendimiento, integración más completa y la mejor experiencia de usuario en comparación con otras opciones; sin embargo, el desarrollo nativo suele ser también la opción de desarrollo más compleja.

Aplicaciones Web

Una aplicación web (web-app/web-based) se basa en HTML5, JavaScript y CSS, y no depende de ninguna tienda de aplicaciones. Es un sitio almacenado localmente en el móvil que trata de emular el look-and-feel (aspecto y comportamiento) de una aplicación. Un famoso ejemplo de una aplicación web es la del Financial Times, que abandonó la tienda de aplicaciones con el fin de evitar compartir los ingresos de los suscriptores; justo a la inversa, la aplicación web basada en Facebook iOS fue puesta al día como aplicación nativa para mejorar dramáticamente su rendimiento y usabilidad. Hay frameworks para aplicaciones web que permiten construir un contenedor nativo de manera



que puedan ser publicadas en las tiendas de aplicaciones, por ejemplo Phonegap³.

Aplicaciones Híbridas

Existe una exagerada controversia en torno a si las aplicaciones nativas son el futuro o si lo son las aplicaciones web. Para muchos de los desarrolladores de aplicaciones móviles, esta controversia ya no existe porque la aproximación híbrida al desarrollo de aplicaciones se ha convertido en algo bastante común: una aplicación puede utilizar código nativo para incrementar su rendimiento e integrarse con la plataforma, mientras que utiliza una webview con contenido basado en HTML5 para otras cosas. Una aplicación híbrida hace uso tanto de las tecnologías nativas como las web. Partes de ella se comportan como una aplicación nativa, mientras que otras se ejecutan sobre tecnologías web. Estas partes pueden utilizar la conexión a Internet para ofrecer contenido actualizado. Mientras que esto podría ser visto como un inconveniente, el uso de las tecnologías web permite a los desarrolladores mostrar contenido actualizado sin tener que volver a subir la aplicación a las tiendas de aplicaciones. El desafío clave es combinar las capacidades únicas de tecnologías nativas y web para crear una aplicación verdaderamente fácil de usar y atractiva.

Sitio web

Un sitio web se ejecuta en su mayor parte en tu servidor, pero puedes acceder a varias funciones del dispositivo con JavaScript, por ejemplo para almacenar datos de forma local o solicitar la ubicación actual del dispositivo. A diferencia de las aplicaciones, los sitios web para móviles son inherentemente multiplataforma. Sin embargo, no debes suponer que un navegador móvil siempre estará basado en WebKit, fíjate en

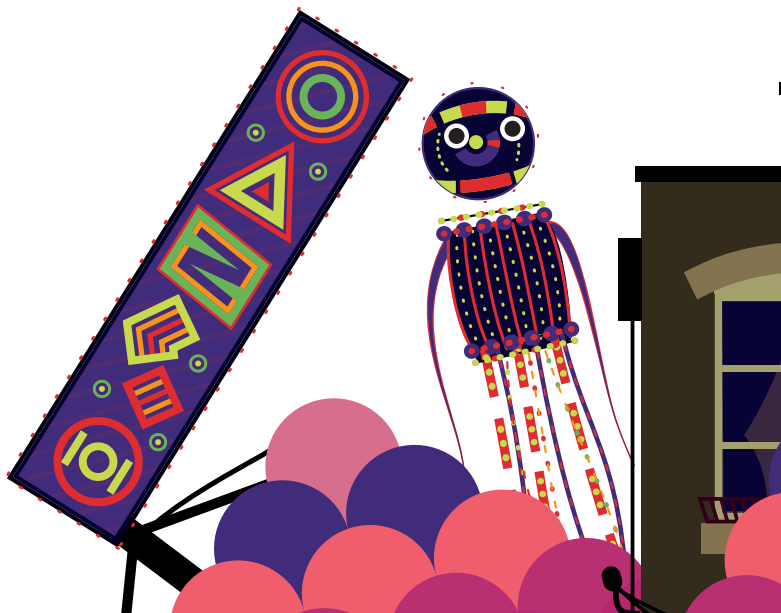
³ www.phonegap.com

la petición de Microsoft a los desarrolladores de webs móviles, para que no creen sus sitios web basándose sólo en WebKit⁴.

SMS, USSD y STK

Los servicios sencillos pueden ser realizados con SMS, USSD o STK. Todo el mundo sabe cómo funciona el sistema de mensajería de texto SMS (Short Message Service), y todos los teléfonos son compatibles con él, pero necesitas convencer a

- 4 blogs.windows.com/windows_phone/b/wpdev/archive/2012/11/15/adapting-your-webkit-optimized-site-for-internet-explorer-10.aspx



los usuarios de que recuerden comandos de texto para servicios más complejos. Algunos operadores ofrecen APIs para servicios de mensajería que funcionan para dispositivos que sólo usan WiFi, tales como las APIs de red de Deutsche Telekom⁵. El USSD (Servicio Suplementario de Datos no Estructurados, del inglés Unstructured Supplementary Service Data) es un protocolo GSM utilizado para enviar sencillos menús basados en texto, cuyas capacidades dependen de la compañía operadora y el dispositivo. El STK (SIM Application Toolkit) permite implementar aplicaciones interactivas de bajo nivel directamente en la tarjeta SIM de un teléfono.

El STK puede parecer irrelevante cuando nos centramos en aplicaciones para smartphones, sin embargo M-Pesa, por ejemplo, es una aplicación STK que está transformando las transacciones financieras en Kenia y otros países.⁶

El Universo de los Sistemas Operativos Móviles

El espacio móvil es mucho más diverso que otras áreas TIC. Cuando estás desarrollando software para ordenadores de sobremesa, tienes básicamente 3 sistemas operativos para elegir, pero en el caso de los móviles hay muchos más. Este libro ofrece una introducción a aquellos más relevantes actualmente, pero hay que ser consciente de que el entorno móvil cambia continuamente a una velocidad que rara vez se observa en otros negocios. Hemos visto muchas tecnologías prometedoras aparecer y desaparecer en breves espacios de tiempo, sin importar lo grandes que eran las compañías que las respaldaban o la relevancia que tuvieran previamente en el mercado.

⁵ www.developergarden.com/apis

⁶ memeburn.com/2012/03/how-m-pesa-disrupts-entire-economies/

Así que sigue leyendo, aprende cómo es el mercado actual y sigue observándolo por tu cuenta (o asegúrate de tener la última edición de nuestra guía a mano).

Quasars: Android e iOS

Cuando la gente habla acerca de las aplicaciones móviles, en su mayoría sólo se refieren a Android e iOS. ¿Por qué? En términos de cuota de mercado, estas dos plataformas combinadas dominan el mercado de smartphones con casi el 90% de cuota en mercados clave, en particular los EE.UU.⁷ (ver la tabla siguiente para los números globales). El estudio Developer Economics 2012⁸ también muestra que iOS y Android están a la cabeza en términos de atención por parte de los desarrolladores, es decir, el porcentaje de desarrolladores utilizando una plataforma, independientemente de la plataforma que ellos consideran su 'primaria'. Android era líder, con un 76% de los desarrolladores trabajando en ella, seguido de iOS con un 66%. Ambas mostraron un aumento de atención en comparación con el año 2011. Otra investigación del 2012, por Appcelerator e IDC⁹ coincide en que los desarrolladores están interesados principalmente en Android e iOS, aunque muestran las cifras invertidas. Este estudio indicó que casi el 90% de los encuestados estaban interesados en iOS, con Android en un porcentaje algo menor, del 80%. A pesar de las diferencias, estos dos informes apuntan a un hecho: si vas a utilizar Android o iOS, tendrás muchísima competencia.

7 blog.nielsen.com/nielsenwire/online_mobile/nielsen-tops-of-2012-digital/

8 www.developereconomics.com

9 www.idc.com/getdoc.jsp?containerId=prUS23619612

Materia Oscura: Plataformas de Móviles de Gama Media

Si bien los smartphones tienen todo el protagonismo, muchas partes del mundo pertenecen al universo del móvil de gama media. A nivel mundial, el 60% de todos los teléfonos vendidos en el tercer trimestre del 2012 han sido teléfonos de gama media¹⁰, con una base instalada muy superior a esa cifra. Los mayores vendedores son Samsung y Nokia. Nokia asegura tener bastante éxito con su Nokia Store ya que hay más de 500 desarrolladores que han tenido más de 1 millón de descargas de sus aplicaciones¹¹. Estudios del 2011 muestran que las plataformas poco promocionadas ofrecen en realidad mejores oportunidades para los desarrolladores: las aplicaciones para teléfonos de gama media en la tienda OVI de Nokia tuvieron 2,5 veces más descargas en comparación con las aplicaciones del Apple App Store¹². Aunque se pueden desarrollar aplicaciones nativas para teléfonos de gama media cuando se tiene estrecha relación con el comercializador, por lo general se desarrollan aplicaciones para estos teléfonos utilizando JavaME o BREW.

Supernovas: Windows 8, Windows Phone, BlackBerry 10 y Aliyun

¿Se convertirán estas plataformas en espectaculares historias de éxito o en desafortunados capítulos de la industria móvil? Nadie lo sabe con seguridad, pero hay señales ambivalentes abiertas a la interpretación.

Mientras que la adopción inicial de Windows 8 fue mayor en números absolutos en comparación con Windows 7, la adopción relativa ha sido más lenta¹³. Las ventas de PCs siguen disminuy-

¹⁰ www.gartner.com/it/page.jsp?id=2237315

¹¹ www.developer.nokia.com/Distribute/Statistics.xhtml

¹² www.research2guidance.com/apps-on-nokia's-ovi-store-had-2.5-times-higher-download-numbers-in-q2-2011-compared-to-apps-on-apple-app-store/

¹³ phonearena.com/news/Wait-so-Windows-8-is-not-outpacing-Windows-7-adoption-rate_id37212

endo, ¿tal vez los consumidores están a la espera de atractivos modelos con pantallas táctiles que destacarían los beneficios de Windows 8? La recepción de Windows 8 varía, por tanto, entre rotundamente hostil y decididamente eufórica.

Windows Phone recibió una cobertura de prensa razonablemente buena con su lanzamiento de WP8, pero todavía tenemos que esperar a ver los números reales de mercado. La adopción global parece estar incrementándose lentamente, y en Italia Windows Phone ha estado incluso rompiendo la barrera del 10%¹⁴. Las aplicaciones se han duplicado prácticamente en el 2012 y las descargas de aplicaciones han aumentado, por lo que aún hay esperanza.

BlackBerry 10 acaba de ser lanzado, y su acogida inicial ha variado entre el escepticismo y el entusiasmo, pero lo más importante es que todos los operadores importantes ofrecerán dispositivos BlackBerry 10.

Aliyun se ha lanzado en un solo dispositivo en China, con una cuota de mercado desconocida. Atrajo publicidad principalmente porque Google presionó a Acer para que no lanzara un dispositivo Aliyun basado en la pertenencia de Acer a la Open Handset Alliance, y en el hecho de que la tienda de aplicaciones de Aliyun promocionaba algunas aplicaciones Google Android que habían sido pirateadas¹⁵. A pesar de que Aliyun afirma estar basado en Linux, el código fuente no ha sido hecho público.

Enanas Blancas: Symbian y bada

Symbian y Samsung bada son sólo sombras de lo que una vez fueron. Aunque bada tuvo una vida muy corta, Samsung anunció que seguirá viviendo dentro de la futura plataforma

¹⁴ guardian.co.uk/technology/2012/oct/02/windows-phone-europe-market

¹⁵ news.cnet.com/8301-1035_3-57513651-94/alibaba-google-just-plain-wrong-about-our-os

Tizen, pero no especificaron de qué manera. Symbian ha sido forzado a entrar en modo mantenimiento, a pesar de que Nokia todavía lanza algunos emblemáticos dispositivos basados en Symbian como el PureView 808, pues su importancia y cuota de mercado continúan cayendo drásticamente a nivel mundial.

Estrellas Recién Nacidas: Firefox OS, Mer, Sailfish, Tizen, Ubuntu

Veamos algunas novedades interesantes del 2013.

La iniciativa Boot-to-Gecko de Mozilla ahora se llama Firefox OS¹⁶. Este sistema operativo está basado en tecnologías web abiertas y ZTE, y Alcatel lanzará dispositivos basados en él en el 2013. Curiosamente, Firefox OS ya ha sido portado a las populares plataformas hacker Raspberry Pi y Pandaboard¹⁷. Firefox OS estará dirigido principalmente a dispositivos de fácil inserción en el mercado, por lo que puede tener un gran impacto sobre las futuras ventas de smartphones en países en desarrollo.

El proyecto Mer¹⁸ continúa la plataforma MeeGo y proporciona la base para Sailfish OS de Jolla¹⁹. Se prevé lanzar Sailfish OS en el 1er trimestre de 2013, y su interfaz de usuario basada en gestos ha sido elogiada por algunos de quienes ya la han podido ver.

Se espera que los dispositivos Tizen²⁰ sean lanzados en el 2013 por Samsung y Lenovo. Aparentemente impulsado con cierta moderación por Samsung e Intel, Tizen tiene como objetivo no sólo los smartphones, sino también televisores, tablets, netbooks y sistemas de infotainment en vehículos.

¹⁶ mozilla.org/firefoxos

¹⁷ rawkes.com/articles/there-is-something-magical-about-firefox-os

¹⁸ merproject.org

¹⁹ jolla.com

²⁰ tizen.org

Por último, pero no menos importante, Canonical presentó Ubuntu²¹ para dispositivos móviles. La idea es llevar la potencia de un PC a los teléfonos.

Cifras Precisas: Cuotas de Mercado de Sistemas Operativos

Echando un vistazo a la cuota de mercado mundial de smart-phones, la imagen parece simple:

Plataforma	Cuota de Mercado a finales de 2012	Ventas en el tercer trimestre de 2012 (millones de unidades)	Base instalada a finales de 2012 (millones de unidades)
Android (Google)	53%	129	710
iOS (Apple)	19%	27	260
Symbian (Nokia)	14%	4	190
BlackBerry (RIM)	8%	8	104
bada (Samsung)	>2%	5	30
Windows Phone (Microsoft)	<2%	4	20

(Fuente: www.tomiahonen.com/ebook/phonebook.html)

Es posible que estés de acuerdo con la mayoría de los desarrolladores en que invertir el tiempo en plataformas diferentes de iOS y Android sería una pérdida de tiempo, pero

²¹ ubuntu.com/devices/phone

puedes estar seguro de que no es tan sencillo. Los propietarios de smartphones son todavía una minoría. Hay más gente con teléfonos de gama media y en muchas regiones del mundo se sigue optando, en el momento de comprar un nuevo dispositivo, por uno de ellos antes que por un smartphone: Por ejemplo, en toda la región MEA (del inglés Middle East & Africa), el 80% de todos los teléfonos vendidos en el cuarto trimestre del 2012 han sido de gama media²². Estos dispositivos ni siquiera aparecen en la tabla anterior, probablemente porque los usuarios de estos teléfonos no suelen utilizar aplicaciones y, por lo tanto, no son de interés para muchos desarrolladores.

Nuevas plataformas tales como BB10 no aparecen en muchas estadísticas porque son nuevas en el mercado. Sin embargo, una de estas plataformas todavía podría ser la mejor opción para tu caso de negocio.

También hay que recordar que se trata de cifras globales; la participación en el mercado regional de cada plataforma es una historia muy diferente. En un mundo donde el contenido geolocalizado está aumentando en importancia, es fundamental conocer los detalles y las características de su mercado de origen. Por ejemplo, China es actualmente el mayor mercado de smartphones y donde Android domina claramente, con una cuota de mercado superior al 90%²³.

Para obtener información sobre la cuota de mercado en tu región objetivo, echa un vistazo a recursos en línea tales como comscore²⁴, StatCounter²⁵, VisionMobile²⁶ o Gartner²⁷.

²² gulfnnews.com/business/technology/apple-microsoft-to-steal-market-share-in-mea-smartphone-sales-1.1112944

²³ www.techinasia.com/android-market-share-china-2012/

²⁴ comscoredatamine.com/category/mobile

²⁵ gs.statcounter.com

²⁶ visionmobile.com

²⁷ gartner.com

Sobre Tiempo y Espacio

Como desarrolladores, tendemos a mostrar una gran pasión por nuestros sistemas predilectos. Sin embargo no debemos olvidar que estas tecnologías son sólo eso, tecnologías que son relevantes en un momento y lugar dado, pero nada más. Sí, algunas flamewars son divertidas pero, en retrospectiva, siempre son pueriles. ¡Que levanten la mano aquellos que participaron en el debate Atari versus Amiga, en los viejos y buenos años 80! Probablemente no muchos de vosotros, pero seguramente, entendéis a los que nos referimos. Iniciativas como FairPhone²⁸ pueden ser más importantes que el sistema operativo o el proveedor que elijas en el futuro.

Perdido en el Espacio

Si estás perdido en la selva del desarrollo móvil, no te preocupes, mantén la calma y sigue leyendo. Explora tus opciones y evalúa el problema que deseas resolver, tu público objetivo y tu know-how. Pon un gran esfuerzo en diseñar la experiencia de tu servicio, concéntrate en el problema en cuestión, y mantenlo simple. Es mejor hacer una cosa bien que hacer "todo" sólo regular. Invierte en el diseño y la usabilidad de la solución. Por último, pero no menos importante, encontrar el nicho de mercado adecuado es a menudo mejor que tratar de copiar algo que ya ha tenido éxito. ¡Esta guía te ayudará a tomar una decisión informada!

²⁸ fairphone.com



Diseño Conceptual para Móviles

Tener una idea es uno de esos maravillosos momentos "¡Ajá!". De repente sabes qué hacer y tienes la confianza de que tu idea va a resolver el problema al que se enfrentan tus usuarios potenciales. Esos momentos son generalmente precedidos de mucha investigación y experiencia, pero el tomar consciencia instantáneamente es lo que probablemente hace decir a la gente que tener ideas es fácil. Más difícil es transformar el concepto en un producto que funcione, sobre todo dentro de las limitaciones del medio en el que deseas expresar tu idea: una aplicación móvil. No sólo aspiras a construir una aplicación estable, también quieres que sea útil y fácil de usar. Antes de entrar en diseño y programación, vale la pena emplear un poco de tiempo en perfeccionar tu idea.

Capturando la idea

Escribe un **resumen** que describa tu concepto de aplicación en unas pocas frases. Trata de explicarla a varias personas (que no sean miembros del equipo) para ver cómo de bien la comprenden y se relacionan con ella.

Define tu **contenido**. Pregúntate cuál es el contenido básico de la aplicación. Dependiendo del tipo de aplicación, podrían ser fotografías (Pinterest), feeds generados por el usuario (Twitter), textos (lectores de libros), etc. Una vez que identificas el valor principal que tu aplicación va a ofrecer, es más fácil enfocar correctamente la interfaz de usuario. Por ejemplo, si estás creando una aplicación de lectura de libros probablemente querrás asegurarte de que la tipografía es de buena calidad.

Describe la **funcionalidad** principal. ¿Qué harán los usuarios a través de tu interfaz? Piensa en ello en términos de verbos y trata de enumerarlos: navegar, compartir, comprar, etc. Te darás cuenta de que algunas actividades están relacionadas. Por ejemplo, si tu aplicación tiene un aspecto social importante, habrá una serie de características que se puedan agrupar (como compartir, comentar, enviar mensajes o seguir a alguien). Este puede ser otro indicio de interfaz de usuario para ti. Presentar funciones relacionadas de manera similar ayuda a los usuarios.

Conoce a tu **audiencia**. ¿Quiénes son las personas para las que estás diseñando? Una técnica útil es la creación de personas (término en inglés original), perfiles genéricos de grupos de usuarios que te ayudarán a comprender las diferentes motivaciones de uso de tu aplicación.

Diseñando experiencias móviles es necesario pensar en el **contexto** en el que tu aplicación se utilizará, y en cómo va a afectar tanto a la interfaz como a los usuarios. ¿Crees que conseguirás captar toda su atención? ¿Es tu aplicación un producto independiente? ¿Se relaciona o depende de otros servicios? ¿Qué ocurrirá si no hay conexión a Internet? ¿Cómo gestionará esta situación la interfaz de usuario?

Después de responder a tantas preguntas deberías tener una idea más clara acerca de la aplicación que quieres crear. Vale la pena pasar algún tiempo investigando. Juega con otras aplicaciones que podrían ser similares a la tuya. Descubre cómo lo hacen: qué piensan los usuarios acerca de ellas. Es una buena manera de conocer el terreno en el que estás entrando.

Cuando avances con el desarrollo de tu concepto mantén en revisión tu lista de preguntas. Son una buena forma de continuar enfocado y ver si sigues en la senda de lo que quieres lograr.

Diseñando la Experiencia de Usuario

La experiencia del usuario (o UX, del inglés User Experience) es cómo los usuarios perciben tu aplicación durante y después de haberla utilizado. Para diseñar y mejorar la experiencia global tienes que pensar en el flujo de uso, funcionalidades, interacciones y diseño visual. ¿Cómo funcionará todo junto en el entorno de tu aplicación? Se trata de pensar acerca de los problemas que los usuarios puedan tener durante el uso de la aplicación y tratar de encontrar soluciones en una fase de prototipo. El diseño no es un proceso lineal, así que hay que iterar y aplicar lo aprendido de las pruebas de usuario tan a menudo como sea posible.

Flujo de uso

Algunas aplicaciones tienen un flujo muy lineal para lograr una tarea sencilla (por ejemplo, una aplicación de cámara). Otras pueden tener flujos más iterativos. Dibuja tu "escenario ideal" donde el usuario comienza en un punto A y, después de una serie de pasos, termina en un punto B. Puedes dibujar diagramas de flujo o mockups para trazar los diferentes flujos. Trata de mantener la navegación lo más sencilla e intuitiva posible.

Wireframes

Los wireframes (maquetas de estructura) o mockups (prototipos que muestran el look-and-feel), son versiones sencillas, esquemáticas, de la interfaz. Su propósito es capturar las funcionalidades y la estructura general de la misma. Se pueden hacer con lápiz y papel o utilizando una de las muchas herramientas de creación de prototipos que existen.

Para una determinada pantalla, tu wireframe tendrá estados diferentes en función de un escenario. Para un error de red, tendrás distintas instancias de la misma pantalla. También para una plataforma diferente el flujo y estructura en pantalla cambiará.

Debes familiarizarte con las directrices para interfaces de usuario del sistema operativo para el que vas a desarrollar. Cada plataforma es un entorno diferente y debes leer las directrices para utilizar las convenciones correctas. Un ejemplo clásico es el botón de vuelta atrás en la cabecera de aplicaciones iOS versus el botón de retroceso de Android que está siempre visible en pantalla independientemente de la aplicación que se esté ejecutando (a veces es un botón físico en el dispositivo). A menos que tengas fuertes razones para hacer lo contrario, sigue las prácticas recomendadas. Investiga y familiarízate con las galerías de patrones disponibles en línea. Encontrarás también enlaces a recursos especializados por plataforma en los respectivos capítulos de este libro.

Dibujar en papel es probablemente la mejor manera de empezar, ya que no es necesario invertir tiempo en aprender un nuevo software, y los dibujos son más fáciles de cambiar y desechar. También es muy divertido hacerlos. Una de las ventajas de las aplicaciones dedicadas a la realización de wireframes y mockups es la capacidad que tienen de hacer que tus wireframes sean clicables en prototipos que pueden probar tus usuarios.

Prototipado

Un prototipo es la mejor manera de visualizar y evaluar las interacciones de tu aplicación. No importa si tienes un gran presupuesto o trabajas en un proyecto personal los fines de semana. Tener un prototipo bastante completo de tu aplicación

es la mejor forma de evaluar su concepto o discutirlo con el cliente. Se lleva a cabo antes de invertir tiempo en el desarrollo del código final y de realizar diseños con precisión a nivel de píxel. Un prototipo validado también es una referencia útil para que los equipos puedan trabajar sin arriesgarse a desviarse demasiado del plan director.

Utiliza la técnica con la que te sientas más cómodo, no hay mejor manera de ensamblar un prototipo. Puede ser una página HTML, un conjunto de esquemas planos enlazados por un flujo clicable, un storyboard en PowerPoint, etc.

Hay muchas herramientas para elegir. Algunas son más especializadas, lo que te permite dibujar o crear wireframes navegables. Algunas te permiten diseñar mockups y compartir y publicar prototipos avanzados. A continuación, una lista de algunas aplicaciones que pueden ser de utilidad:

- **Mockflow:** www.mockflow.com (gratis - descarga & online).
- **Pencil Project** pencil.evolus.vn (gratis - descarga).
- **Axure** www.axure.com (de pago - descarga).
- **Balsamiq** www.balsamiq.com (de pago - descarga).
- **Mockingbird** www.gomockingbird.com (de pago - online).
- **Flowella** www.developer.nokia.com/Resources/Tools_and_downloads/Other/Flowella (gratis - descarga).
- **App in seconds** www.appinseconds.com (de pago - online).

Diseño visual

A menos que estés construyendo una solución que sólo utiliza una entrada de sonido o interfaces hápticas, tu interfaz de usuario se basará en los gráficos. La aplicación del diseño visual no es simplemente colorear wireframes, un diseño visual atractivo mejorará la experiencia de tu aplicación y hará que se destaque de la masa.

La gestión de los espacios y la jerarquía visual mejoran la usabilidad de la interfaz. La estructura gráfica define los detalles del posicionamiento de los elementos en pantalla y su relación entre sí. Después de que los usuarios aprendan la interfaz de usuario, ésta debería permanecer constante en todo el flujo de la aplicación. Por ejemplo, si el botón principal cambia de color de una pantalla a otra eso obliga a los usuarios a volver a aprender las convenciones cada vez que visualizan una pantalla diferente. Si eso es lo que deseas, asegúrese de que lo estás haciendo por una buena razón.

Al igual que el diseño de interacción en el nivel de wireframes, ciertas decisiones de estilo pueden ser informadas por las directrices de la plataforma. Tu aplicación puede parecer muy diferente dependiendo de la plataforma para la que fue diseñada. Asegúrate de que tus diseños siguen las prácticas recomendadas para el uso de fuentes, iconos estándar y convenciones de estructura gráfica. Una vez más, consulta los capítulos relacionados con la plataforma que te interesa en esta guía para encontrar más información y enlaces a recursos específicos online.

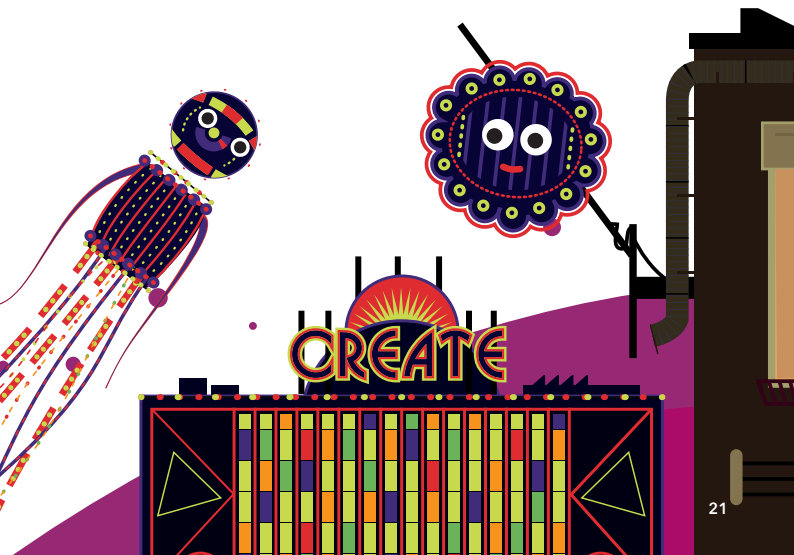
Lo óptimo es que la marca de la empresa esté presente en la interfaz de una manera no obstructiva. Utiliza el fondo, colores de control y, si cabe, ciertas imágenes u opciones de diseño para obtener el aspecto deseado. La pantalla de inicio (si existe) es el lugar donde se pueden mostrar algunos gráficos adicionales.

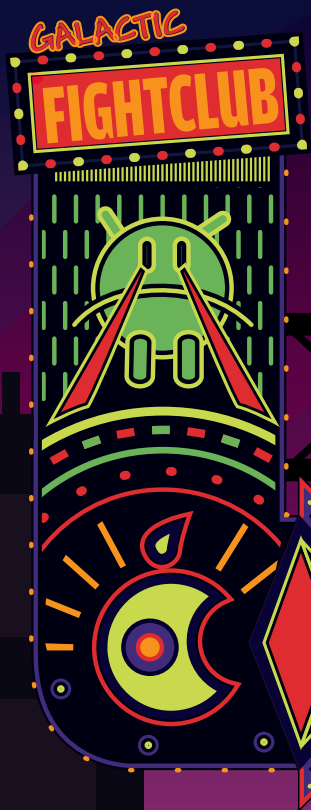
Finalmente, el icono de aplicación es el primer elemento visual con el que tu aplicación va a ser identificada y evaluada. Haz que tenga un buen aspecto. Si estás planeando hacer versiones para múltiples plataformas, comprueba los requisitos de diseño previamente para que puedas crear un diseño fácilmente migrable.

Pruebas de usuario

La mejor manera de validar tu interfaz y concepto es enfrentarlos con usuarios reales, tan pronto como sea posible. No tienes que esperar hasta que haya un producto acabado y pulido. De hecho, no deberías. Es mucho más difícil aceptar comentarios sobre algo que se considera casi terminado que en un prototipo navegable que se puede actualizar con bastante rapidez.

Pide a algunas personas que realicen ciertas tareas con tu prototipo. Si la aplicación que estás diseñando es un reproductor de música, puedes pedirles que pongan una canción. Si no estás seguro de ciertas funciones puede tratar de desviar la atención de los usuarios, pidiéndoles que realicen tareas inversas, como cambiar la pista seleccionada y elegir otra en su lugar. Lo principal es moderar, no guiar. También puedes realizar sesiones de pruebas en otras aplicaciones ya publicadas. Te sorprenderás de lo mucho que perciben los usuarios de tu aplicación, cosas que puede que nunca te hayas parado a pensar.





Android

El Ecosistema

La plataforma Android ha sido desarrollado por la Open Handset Alliance, liderada por Google, y ha estado a disposición del público desde noviembre de 2007. Su uso por muchos fabricantes de hardware la ha convertido en el sistema operativo de smartphones con mayor crecimiento. De acuerdo con IDC¹, el 75% de todos los smartphones vendidos en el tercer trimestre del 2012 en todo el mundo se basa en Android. En otoño de 2012, Google anunció que hay 700.000 aplicaciones disponibles en el Android Market² y que quinientos millones de dispositivos Android han sido activados hasta el momento³. Android también se utiliza en tablets, reproductores multimedia, decodificadores de televisión, teléfonos de sobremesa y sistemas de entretenimiento en automóviles. Algunos dispositivos no-Android también son capaces de ejecutar aplicaciones Android con funcionalidad reducida, como la Playbook de RIM con su BlackBerry Adroid runtime o el nuevo sistema operativo de código abierto Sailfish⁴.

Android es un sistema operativo, una colección de aplicaciones preinstaladas y una plataforma de desarrollo de aplicaciones (Dalvik) soportada por un amplio paquete de herramientas. La plataforma sigue evolucionando rápidamente, con la adición de nuevas características regularmente, cada 6 meses aproxima-

¹ www.idc.com

² www.nbcnews.com/technology/gadgetbox/google-says-there-are-700-000-android-activations-day-118326

³ mashable.com/2012/09/12/500-million-android-devices-activated/

⁴ www.sailfishos.org/

damente. Por ejemplo, la versión de Android 4.2, "Jelly Bean", introdujo una funcionalidad llamada "Butter" que básicamente funciona como buffer triple y da como resultado una navegación mucho más suave y una frame rate estable. Además, Google implementó un sistema de notificación mejorada con menús desplegables, iconos flotantes en la pantalla de inicio que automáticamente se reposicionan si mueves iconos hacia ellos y soporte USB-Audio. Hay muchas mejoras y añadidos menores que demuestran que Google ha conseguido convertir Android en una base estable para el futuro.

Uno de los problemas más debatidos cuando se desarrolla para Android es la fragmentación: La multitud de dispositivos diferentes de diversos fabricantes y el rápido progreso de la plataforma en sí generan incertidumbre sobre si una aplicación Android podrá ejecutarse en cualquiera situación. Además, sólo un número muy pequeño de teléfonos y modelos de tablet son compatibles con la última versión del sistema operativo. En cualquier caso, actualmente puedes alcanzar el 96,9% de la base instalada si decides enfocarte a Android 2.2 o superior⁵. Para reducir los problemas de fragmentación causados por las grandes diferencias en el tamaño de pantalla, Android 3.2 introdujo un nuevo recurso descriptor llamado "smallestWidth", que puede ser utilizado para apuntar a teléfonos y tablets con diferentes estructuras gráficas en función de sus dimensiones⁶. Para reforzar una experiencia de usuario sólida y una apariencia consistente en las aplicaciones de Android, Google publica una

⁵ developer.android.com/resources/dashboard/platform-versions.html

⁶ developer.android.com/guide/practices/screens_support.html#NewQualifiers



guía de diseño para aplicaciones Android que está disponible en developer.android.com/design/. También publica una guía de desarrollo más general llamada Training⁷ que ayuda a los desarrolladores a lidiar con las tareas más comunes, como por ejemplo tratar datos persistentes del usuario.

Prerrequisitos

El lenguaje de programación principal para Android es Java. Pero ten cuidado, sólo un subconjunto de las librerías Java son compatibles y hay muchas APIs específicas de la plataforma. Puedes encontrar respuestas online a la preguntas "¿Qué y Por qué?" en la Dev Guide para Android⁸ y a al "¿Cómo?" en la documentación de referencia⁹. Además, Google ha introducido una sección en su documentación llamada "Android Training"¹⁰ dirigida a los nuevos desarrolladores que quieran aprender sobre prácticas recomendadas, conceptos básicos como la navegación y comunicación entre aplicaciones, o funciones más avanzadas como la descarga inteligente de bitmaps y la optimización para reducir el gasto de batería.

Para empezar, necesitas el SDK de Android¹¹, que está disponible para Windows, Mac OS X y Linux. Contiene las herramientas necesarias para crear, probar, depurar y analizar aplicaciones. Es probable que también quieras un buen IDE de Java. Las herramientas de desarrollo de Android (ADT)¹² dan soporte a Eclipse; otros IDEs, como IntelliJ de JetBrains.com, ofrecen buen soporte al desarrollo, despliegue y (muy importante) proyectos de librerías que permiten la compartición de código y recursos entre proyectos.

⁷ developer.android.com/training

⁸ developer.android.com/guide

⁹ developer.android.com/training/index.html

¹⁰ developer.android.com/training/index.html

¹¹ developer.android.com/sdk

¹² developer.android.com/tools/sdk/eclipse-adt.html

Implementación

Arquitectura de la aplicación

Una aplicación Android puede incluir una mezcla de actividades, servicios, receptores de mensajes y proveedores de datos, los cuales tienen que ser declarados en el manifiesto de aplicación.

Una actividad es un bloque de funcionalidad con una interfaz de usuario adjunta. Un servicio se utiliza para tareas que se ejecutan en segundo plano y, por lo tanto, no ligadas directamente a una representación visual. Un receptor de mensajes gestiona los mensajes transmitidos por el sistema u otras aplicaciones. Un proveedor de datos es una interfaz para el contenido de una aplicación que abstrae de los mecanismos de almacenamiento subyacentes.

Una aplicación puede consistir en varios de estos componentes, por ejemplo, una actividad para la interfaz de usuario y un servicio para tareas dilatadas en ejecución. La comunicación entre los componentes se realiza por objetos intent.

Un intent empaqueta datos, tales como la localización del usuario o una URL, con una acción. Estos intents desencadenan acciones en la plataforma y pueden ser utilizados como un sistema de mensajería en tu aplicación. Por ejemplo, el intent para mostrar una página web abrirá el navegador web. La fuerza de esta filosofía de bloques de construcción es que se puede sustituir la funcionalidad por otra aplicación, ya que Android siempre utiliza la aplicación por defecto para un propósito específico. Por ejemplo, el intent para compartir una página web llamada por una aplicación de lectura de noticias puede abrir un cliente de correo electrónico o una aplicación de mensajería de texto, dependiendo de las preferencias del usuario y de las aplicaciones instaladas: se puede utilizar cualquier aplicación que declare un intent tipo share como interfaz.

La interfaz de usuario de una aplicación está separada del código en archivos XML de layout específicos de Android. Diferentes diseños pueden ser creados para diferentes tamaños de pantalla, localización por país y características del dispositivo, sin tocar el código Java. Con este fin, textos e imágenes están organizados en carpetas separadas de recursos. Por supuesto, también puedes definir layouts vía código.

El SDK y los Plug-ins

Para ayudarte en el desarrollo tienes muchas herramientas a tu disposición en el SDK, los más importantes son:

- **android:** Para crear un proyecto o gestionar dispositivos virtuales y versiones del SDK.
- **adb:** Para llamar a los dispositivos, conectarse e interactuar con ellos (así como dispositivos virtuales) moviendo archivos, instalando aplicaciones, etc.
- **emulator:** Para emular las características de un dispositivo virtual. Lleva un tiempo arrancarlo, por lo que hazlo sólo una vez por sesión de trabajo y no para cada compilación.
- **ddms:** Para observar el interior del dispositivo o emulador, ver los mensajes del registro (log) y funciones de control del emulador, como la latencia de red y la posición GPS. También se puede utilizar para monitorizar el consumo de memoria o terminar procesos. Si esta herramienta está en ejecución, también puedes conectar el depurador de Eclipse a un proceso que se ejecuta en el emulador. Aparte, ddms es la única manera (sin acceso root) para realizar capturas de pantalla de las versiones de Android por debajo de la 4.0.

Estas cuatro herramientas y muchas otras se pueden encontrar en el directorio de herramientas del SDK, incluyendo

algunas para el análisis de los registros de método de rastreo, inspeccionar diseños y probar aplicaciones con los eventos al azar.

Los plug-ins IDE están disponibles para ayudar a gestionar estos archivos. La versión 11.x de IntelliJ incluye un editor visual de layouts, por lo que tienes libertad para elegir entre Eclipse e IntelliJ en caso de que quieras hacer prototipado rápido, arrastrando y posicionando elementos de la interfaz en el editor.

Si te enfrentas a problemas, tales como generación de excepciones, asegúrate de revisar el registro de ddms. Te permite comprobar si olvidaste agregar todos los permisos necesarios, tales como `android.permission.INTERNET` en el elemento `uses-permission`¹³.

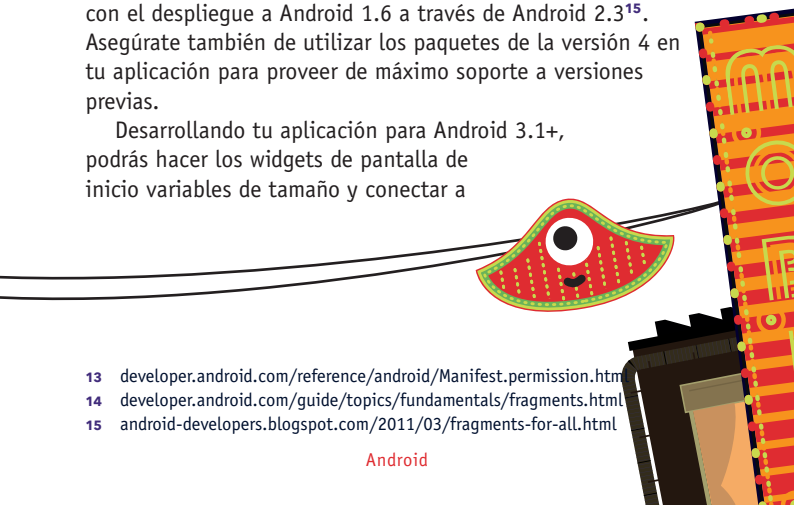
Si estás utilizando las características introducidas después de Android 2.3, tales como Fragments¹⁴ para pantallas grandes, asegúrate de agregar el paquete Android Compatibility de Google. Está disponible a través del SDK & AVD Manager y ayuda a desarrollar para Android 3.0+ sin causar problemas con el despliegue a Android 1.6 a través de Android 2.3¹⁵. Asegúrate también de utilizar los paquetes de la versión 4 en tu aplicación para proveer de máximo soporte a versiones previas.

Desarrollando tu aplicación para Android 3.1+, podrás hacer los widgets de pantalla de inicio variables de tamaño y conectar a

¹³ developer.android.com/reference/android/Manifest.permission.html

¹⁴ developer.android.com/guide/topics/fundamentals/fragments.html

¹⁵ android-developers.blogspot.com/2011/03/fragments-for-all.html



través de USB a otros dispositivos, como cámaras digitales, gamepads y muchos otros.

Android Ice Cream Sandwich (4.0) y Jelly Bean (4.1 y 4.2) introducen interesantes novedades, como notificaciones expandibles, widgets de bloqueo de pantalla y una cámara con detección de rostros. El entorno nativo Renderscript (introducido en Honeycomb 3.1) ha cambiado mucho y no proporciona la capacidad de renderizar gráficos directamente.

Para ofrecer compatibilidad con dispositivos con versiones anteriores de Android, Google comenzó a utilizar el framework Google Play Services¹⁶, que se actualiza a través de la Play Store y añade librerías, como por ejemplo la más reciente de Google Maps. Para aprovechar algunas de las funciones ofrecidas debes autenticar a los usuarios con OAuth 2.0.

Testeo

El primer paso para probar una aplicación es ejecutarla en el emulador o en un dispositivo. Es posible depurarla, si es necesario, a través de la herramienta ddms.

Todas las versiones del sistema operativo Android están preparadas para ejecutarse en dispositivos sin necesidad de modificaciones aunque, sin embargo, algunos fabricantes de hardware pueden realizar cambios en algunos elementos de la plataforma; por tanto, probar la aplicación en varios dispositivos físicos es esencial. La lista de AppBrain¹⁷ puede ayudarte a hacerte una idea de qué dispositivos son los más populares.

¹⁶ <http://developer.android.com/google/play-services/>

¹⁷ www.appbrain.com/stats/top-android-phones

Para automatizar las pruebas, el SDK de Android viene con algunas herramientas muy útiles¹⁸. Los tests se pueden escribir con el formato estándar JUnit, utilizando los objetos simulados de Android (mock objects) que se incluyen en el SDK.

Las clases Instrumentation pueden monitorizar la interfaz de usuario y enviar eventos del sistema tales como pulsaciones de teclas. Tus tests pueden comprobar el estado de tu aplicación tras estos eventos. MonkeyRunner¹⁹ es una potente y extensible herramienta de automatización de pruebas que permite probar toda la aplicación. Estas pruebas se pueden ejecutar tanto en dispositivos virtuales como físicos.

En la revisión 21 del SDK, Google introdujo finalmente un framework de automatización de pruebas de interfaz más eficiente²⁰, que permite realizar pruebas funcionales de interfaz de usuario en Android Jelly Bean y versiones superiores. La herramienta puede ser ejecutada desde el shell con el comando `uiautomatorviewer` y muestra la interfaz capturada, incluyendo información sobre las vistas actuales. Ejecutar las pruebas es relativamente fácil: Una vez que hayas escrito tu test, éste se compila a través de ANT como un archivo JAR. Este archivo tiene que ser subido al dispositivo y ejecutado con el comando `adb shell uiautomator runtest`.

Entornos de prueba de código abierto, como Robotium²¹, pueden complementar otros tests automatizados. Robotium incluso puede ser utilizado para probar archivos binarios apk si el código fuente de la aplicación no está disponible. Roboelectric²² es otra gran herramienta que ejecuta las pruebas directamente sobre tu IDE en tu JVM estándar/de escritorio.

¹⁸ developer.android.com/guide/topics/testing/testing_android.html

¹⁹ developer.android.com/guide/developing/tools/monkeyrunner_concepts.html

²⁰ android-developers.blogspot.de/2012/11/android-sdk-tools-revision-21.html

²¹ code.google.com/p/robotium

²² pivotal.github.com/roboelectric/

Tus pruebas automatizadas se pueden ejecutar en servidores de integración continua como Jenkins o Hudson. Roboelectric se ejecuta en una JVM estándar y no necesita un runtime Android. Muchos otros entornos automatizados de pruebas, incluyendo Robotium, se basan en el entorno de Instrumentación de Android, y requerirán ser ejecutados en la JVM Dalvik. Plug-ins como el Android Emulator Plugin²³ permiten que estos tests sean configurados y ejecutados en Hudson y Jenkins.

Compilación

Aparte de crear tu aplicación directamente en el IDE que prefieras, hay maneras más cómodas de construir aplicaciones Android. El software Gradle²⁴ es actualmente la herramienta de creación automatizada con soporte oficial para Android. También hay un plug-in de Maven²⁵ que cuenta con un gran apoyo en la comunidad de desarrolladores. Ambas herramientas pueden tener dependencias de diferentes repositorios Maven, por ejemplo del Maven Central Repository²⁶.

²³ wiki.hudson-ci.org/display/HUDSON/Android+Emulator+Plugin

²⁴ más información en tools.android.com/tech-docs/new-build-system/using-the-new-build-system

²⁵ code.google.com/p/maven-android-plugin/

²⁶ www.maven.org/



Firma

Tu aplicación tiene que haber sido firmada en el proceso de compilación, ya sea con una firma de depuración o con una de publicación. Puedes utilizar un mecanismo de auto-firma, lo que evita gastos relacionados con la firma (y seguridad).

La misma firma debe ser utilizada para las actualizaciones de la aplicación. Recuerda que puedes utilizar una única clave para todas tus aplicaciones o crear una nueva para cada una de ellas.

Distribución

Una vez hayas creado la próxima aplicación rompedora y la hayas probado, debes publicarla en la tienda de aplicaciones de Android llamada "Play". Este es un buen lugar para llegar a los clientes y vender tus aplicaciones. Android, desde la versión 1.6 en adelante, también es compatible con compras desde la misma aplicación, lo que te permite vender contenido adicional, paquetes de funciones, etcétera, dentro de tu aplicación mediante el uso de la infraestructura de Android Play²⁷.

También se utiliza por otros portales de aplicaciones como fuente de metadatos de aplicaciones. Para subir tu aplicación a Android Play, el punto de partida es play.google.com/apps/publish/.

Estás obligado a inscribirte en el servicio a través de tu cuenta de Google Checkout y pagar una cuota de inscripción de \$25 (USD). Una vez aprobado tu registro, puedes cargar la aplicación, añadir imágenes y descripciones, y publicarla.

²⁷ developer.android.com/guide/google/play/billing/

Asegúrate de que has definido `versionName`, `versionCode`, un icono y una etiqueta en tu `AndroidManifest.xml`. Además, las características declaradas en el manifiesto (nodos `uses-feature`) se utilizan para filtrar aplicaciones para distintos dispositivos.

Como hay una gran cantidad de aplicaciones compitiendo en Android Play, es posible que desees utilizar tiendas alternativas de aplicaciones²⁸. Ofrecen diferentes métodos de pago y pueden dirigirse a grupos específicos de consumidores. Uno de esas tiendas es la Amazon Appstore, que viene preinstalada en la familia de tablets Kindle Fire.

Monetización

Además de vender la aplicación en una de las muchas tiendas de aplicaciones disponibles, hay múltiples maneras diferentes de monetizar una aplicación Android. Una forma viable es mediante el uso de la publicidad, que puede ser basada en clics o en número de visitas, y puede proporcionar un ingreso estable. Aparte, hay diferentes posibilidades para facturar desde la aplicación, como el servicio propio de facturación de Google²⁹, que utiliza la tienda Google Play o pagos mediante la librería Mobile Payments Library de PayPal³⁰. La mayoría de los servicios difieren en las comisiones por transacción y las posibilidades que ofrecen, por ejemplo suscripciones, pagos paralelos o pagos previamente aprobados.

Asegúrate de verificar que el método de pago de tu elección está en armonía con los términos y condiciones de las diferentes tiendas en las que desees publicar la aplicación; en especial vale la pena echarle un vistazo a aquellas que permiten descargas digitales, ya que suelen tener diferentes directrices.

²⁸ wipconnector.com/index.php/appstores/tag/android

²⁹ developer.android.com/google/play/billing/

³⁰ www.x.com/developers/paypal/products/mobile-payment-libraries



BlackBerry: Aplicaciones Java

El Ecosistema

La plataforma BlackBerry es desarrollada por la compañía canadiense Research In Motion (RIM), que el pasado 30 de Enero del 2013 adoptó la marca BlackBerry como nombre comercial de la empresa (aunque sigue cotizando en los índices NASDAQ y TSK como RIMM y RIM, respectivamente)¹. El año de lanzamiento de la plataforma BlackBerry es 1999, haciéndose extremadamente populares sus dispositivos debido a que estaban equipados con un teclado completo para introducir texto cómodamente (que dio lugar a una dolencia llamada Pulgar BlackBerry²), ofrece un robusto servicio de push para email y otros datos, una larga duración de la batería e incluye BlackBerry Messenger, su red social móvil. Suma a esto aplicaciones tipo PDA, como libreta de direcciones, correo electrónico seguro, calendario, tareas y bloc de notas a estas características y comprenderás por qué la plataforma era muy popular tanto en empresas como entre usuarios comunes.

La cuota de mercado global de los teléfonos BlackBerry ha continuado su descenso en 2012³. Para recuperar el terreno perdido en el mercado, BlackBerry decidió tomar medidas radicales e introdujo un sistema operativo completamente nuevo: BlackBerry 10. Los primeros dispositivos BB10 se han lanzado en el primer trimestre del 2013. Este capítulo se centra en el

¹ blackberry.com

² en.wikipedia.org/wiki/Blackberry_thumb

³ gs.statcounter.com/

desarrollo de aplicaciones para los anteriores, los dispositivos BlackBerry no-BB10 actualmente en el mercado; el desarrollo BB10 se explica en un capítulo aparte.

BlackBerry OS es el sistema operativo que se encuentra en todos los smartphones BlackBerry comercializados antes del 2013. Su última versión, lanzada a principios de 2012 (BlackBerry OS 7.1) ofrece algunas mejoras notables respecto a su predecesor: tethering, soporte a llamadas WiFi, soporte a etiqueta NFC y radio FM.

Las incorporaciones más importantes a la API de OS 7.1 son:

- NFC Peer-to-Peer API, que ofrece la capacidad de iniciar transferencias de datos vía NFC, y completar la tarea por Bluetooth
- FM Radio API
- Profiles API, que permite acceso de lectura/escritura al perfil del usuario

Para BlackBerry OS, hay disponibles dos enfoques de desarrollo en función del tipo y la naturaleza del proyecto que estás planificando. Para aplicaciones de mediano a gran tamaño, la mejor opción es el desarrollo nativo en Java, mientras que las pequeñas aplicaciones pueden ser desarrolladas con BlackBerry WebWorks SDK.

A pesar de que se eliminará gradualmente en el futuro, actualmente la API Java de BlackBerry es el método más utilizado para desarrollar aplicaciones BlackBerry. Por ello, este capítulo se centra en el desarrollo con Java.

Prerrequisitos

En cuanto al desarrollo de aplicaciones Java, necesitarás Java SDK⁴ (no el Java Runtime Environment). A continuación, te harán falta Eclipse y el plug-in de BlackBerry⁵. Puedes descargarlos por separado o descargar el paquete que ofrece BlackBerry, que incluye ambos, y que también contiene el SDK y simuladores para el último sistema operativo BlackBerry, con instrucciones sobre cómo descargar los SDKs anteriores disponibles en la página de descargas (necesarios para trabajar en compatibilidad con dispositivos más antiguos). Además, hay simuladores adicionales de dispositivos disponibles para descarga en el sitio web de BlackBerry⁶.

Para implementar la aplicación en un dispositivo de pruebas, debes descargar e instalar el BlackBerry Desktop Manager⁷. Para una implementación más rápida, se puede utilizar una herramienta llamada javaloader que viene con el JDE.

Implementación

El BlackBerry JDE se basa parcialmente en Java ME y algunas de sus extensiones JSR: el estándar MIDP 2.0 está integrado en el SDK, con populares extensiones JSR que proporcionan APIs para la interfaz de usuario, audio, vídeo y servicios de localización, entre otros⁸. Esto significa que las aplicaciones BlackBerry pueden ser creadas utilizando solamente tecnologías Java ME.

⁴ www.oracle.com/technetwork/java

⁵ us.blackberry.com/developers/javaappdev/javaplugin.jsp

⁶ us.blackberry.com/sites/developers/resources/simulators.html

⁷ us.blackberry.com/apps-software/desktop/

⁸ www.blackberry.com/developers/docs/6.0.0api/index.html

Otra opción es el uso de extensiones y frameworks de desarrollo de interfaz de usuario propiedad de BlackBerry, que permiten hacer un uso completo de la plataforma.

El estilo de los componentes nativos de interfaz de usuario puede ser modificado hasta cierto punto, pero heredan su apariencia del tema activo. Esto se puede evitar por código, anulando el método `Field.applyTheme()` para cada componente/campo.

Desde OpenGL-ES a la interacción con la pantalla de inicio y la criptografía, las APIs de BlackBerry proporcionan todo lo necesario para crear aplicaciones atractivas. Además de las herramientas oficiales de BlackBerry, hay extensiones de terceros que te permiten mejorar las aplicaciones, por ejemplo J2ME Polish⁹ o Glaze¹⁰, que te permiten diseñar y animar tu interfaz de usuario mediante CSS.

Servicios

BlackBerry ofrece muchos servicios que pueden ser útiles en el desarrollo de tus aplicaciones, incluyendo publicidad, mapas, pagos y servicios push¹¹.

Un servicio push¹² es útil principalmente en aplicaciones de correo, mensajería o noticias. Su principal ventaja es que el dispositivo espera que el servidor le envíe actualizaciones, en lugar de que el dispositivo esté llamando continuamente al servidor para averiguar si hay actualizaciones disponibles e importarlos al dispositivo desde el servidor. Esto reduce tráfico de red, uso de la batería y, para los usuarios con planes de datos limitados o roaming, reduce también los costes. Funciona de la siguiente manera: tu servidor envía un paquete de datos

⁹ j2mepolish.org

¹⁰ glaze-ui.org

¹¹ developer.blackberry.com/services/#platform

¹² us.blackberry.com/developers/platform/pushapi.jsp

de hasta 8 KB al servicio push de BlackBerry. La infraestructura difunde entonces el mensaje a todos los clientes o a un grupo específico (por ejemplo, el contenido de un informe de noticias), o a un cliente específico (por ejemplo, el contenido de un mensaje de chat). El dispositivo cliente recibe entonces el mensaje a través de la API Push de BlackBerry y puede confirmar la recepción del mensaje de vuelta a la infraestructura. Tu servidor puede entonces comprobar si el mensaje ha sido entregado. BlackBerry ofrece el servicio push gratuitamente de manera limitada, con una extensión premium de pago que permite enviar más mensajes push.

Portabilidad

Portar aplicaciones entre dispositivos BlackBerry es fácil, ya que el sistema operativo ha sido realizado por una sola compañía que ha tenido el cuidado de minimizar los problemas de fragmentación. Sin embargo, esto no elimina totalmente el hecho de tener que enfrentarse a ciertos retos:

- Algunas clases y funciones sólo están disponibles en determinadas versiones del sistema operativo. Por ejemplo, el `FilePicker` que se utiliza para seleccionar un archivo sólo está disponible desde OS 5.0 en adelante.
- Es necesario gestionar diferentes resoluciones de pantalla y modos de orientación (horizontal y vertical).
- Es necesario gestionar dispositivos tanto táctiles como no táctiles. Además, los dispositivos Storm utilizan una pantalla táctil que es físicamente clicable, por lo que hay una diferencia entre un contacto y un clic en estos dispositivos. Los dispositivos táctiles más recientes de BlackBerry ya no utilizan esta tecnología.

Portar a otras plataformas Java tales como Java ME y Android es complicado ya que no es posible trasladar la interfaz de usuario de BlackBerry.

El código escrito para la comunicación con el servidor o el almacenamiento puede ser reutilizado en Java ME y Android si evitas llamadas nativas a la API de BlackBerry. En general, la portabilidad entre plataformas depende en gran medida de la frecuencia con que tu aplicación utilice componentes nativos BlackBerry. Por ejemplo, no es posible reciclar los servicios push de BlackBerry en otras plataformas.

Testeo

BlackBerry ofrece simuladores para varios teléfonos, ya sea incluidos con el plugin de Eclipse o como descargas independientes. Estos simuladores permiten ejecutar una aplicación en un PC de la misma manera que se ejecutaría en un dispositivo. Las capacidades de testeo y depuración de BlackBerry están a la par con las de otras plataformas como Android e iOS: los simuladores permiten a los desarrolladores simular una gran variedad de eventos (llamadas entrantes, los cambios en las coordenadas GPS, los cambios en las condiciones de red, etc.), mientras que la depuración en dispositivos hace que el código sea fácil de probar en hardware real.

Además, los tests automatizados también son posibles, aunque algo limitados y complicados. Puedes utilizar el paquete de herramientas FledgeController¹³ para inyectar eventos programáticamente desde tu ordenador, o puedes utilizar la

¹³ docs.blackberry.com/en/developers/deliverables/15476/Using_the_BBSSmrtphnSmltr_programmatically_607582_11.jsp

clase EventInjector¹⁴ para insertar eventos desde una aplicación BlackBerry en el dispositivo (o simulador). Sin embargo, existe muy poca documentación disponible sobre este tema, por lo que espera tener que hacer algo de hacking y de darle vueltas a la cabeza como parte de tu experiencia con las pruebas automatizadas en BlackBerry.

Firma

Muchas clases de nivel de seguridad crítico y algunas características de la plataforma (tales como trabajo con redes o APIs de archivo) requieren una aplicación para ser firmadas de manera que quien publica pueda ser identificado. Para lograr esto, es necesario obtener una clave de firma directamente desde BlackBerry¹⁵. La firma en sí se lleva a cabo mediante la herramienta rapc, que también empaqueta la aplicación para su distribución.

Distribución

El canal de distribución de BlackBerry se llama App World¹⁶, y es donde puedes publicar tus aplicaciones. Para aplicaciones de pago, como desarrollador obtienes una participación en los beneficios del 70%. Además, GetJar¹⁷ es un sitio web independiente muy conocido que también publica aplicaciones BlackBerry.

¹⁴ <http://www.blackberry.com/developers/docs/4.1api/net/rim/device/api/system/EventInjector.html> www.blackberry.com/developers/docs/4.1api/net/rim/device/api/system/EventInjector.html

¹⁵ www.blackberry.com/SignedKeys/

¹⁶ appworld.blackberry.com/

¹⁷ www.getjar.com/

Aprende Más

Si deseas obtener más información acerca del desarrollo Java para BlackBerry, los siguientes recursos que te pueden ayudar.

Aplicaciones incluidas de ejemplo

Los SDK vienen con una gran selección de aplicaciones de ejemplo, desde una simple aplicación "Hola, Mundo!" a aplicaciones multimedia y de geo-localización complejas.

Online

Un número importante de recursos en línea están disponibles en:

- El microsite oficial de documentación BlackBerry¹⁸
- Los foros de desarrollo de BlackBerry¹⁹

También hay una gran cantidad de conocimiento sobre BlackBerry en Internet, que tratan algunos problemas y temas mucho mejor que la documentación oficial. ¡Los motores de búsqueda son tus amigos!

Libros

Obras impresas que se ocupan del desarrollo Java para BlackBerry:

- **BlackBerry Development Fundamentals**²⁰ por John Wargo
- **Beginning BlackBerry 7 Development 2nd Edition** por Anthony Rizk
- **Advanced BlackBerry 6 Development 2nd Edition** por Chris King

¹⁸ developer.blackberry.com/java/documentation/

¹⁹ supportforums.blackberry.com/t5/Java-Development/bd-p/java_dev

²⁰ bbdevfundamentals.com/



BlackBerry 10

El Ecosistema

La plataforma BlackBerry 10 (BB10) es un relanzamiento global de BlackBerry (marca comercial de Research In Motion, que el pasado 30 de Enero del 2013 adoptó la marca BlackBerry como nombre comercial de la empresa, aunque sigue cotizando en los índices NASDAQ y TSK como RIMM y RIM, respectivamente). Los dispositivos BB10 llegaron al mercado en el primer trimestre del 2013, sin planes de actualización para los dispositivos de la generación anterior. BlackBerry ha adoptado este enfoque con el fin de ponerse al día con los sistemas operativos móviles competidores: iOS, Android y Windows Phone 8. Otro objetivo importante es la armonización de los teléfonos móviles y las tablets de BlackBerry, haciendo que funcionen con el mismo sistema operativo y las mismas aplicaciones.

BlackBerry está sufriendo una gran presión por parte del mercado y está invirtiendo mucho en este relanzamiento, tienen que conseguir que sea un éxito si no quieren perder aún más terreno en el mercado móvil. Esto significa nuevas e interesantes oportunidades para los desarrolladores de aplicaciones que estén dispuestos a desarrollar para la nueva plataforma. Aunque el sistema operativo es completamente nuevo, su núcleo está basado en QNX, un sistema operativo en tiempo real para dispositivos integrados. Las otras partes del ecosistema BlackBerry, como la App World o el servicio push, no han cambiado. Respecto a la seguridad, BB10 ha sido creado con los mismos y elevados estándares por los que es bien conocida la plataforma BlackBerry.

Desarrollo

Con BB10, las aplicaciones se pueden desarrollar utilizando una amplia variedad de tecnologías:

- C Native SDK
- C++ Cascades SDK
- HTML5 (WebWorks SDK)
- Adobe Air
- Android Runtime (capa de compatibilidad Android 2.3.3)
- BlackBerry App Generator

Con el fin de atraer a los desarrolladores a su nuevo sistema operativo, BlackBerry proporciona un amplio conjunto de recursos, incluyendo un simulador, muchos proyectos de ejemplo en GitHub¹ y documentación actualizada frecuentemente².

Un punto importante de descontento, por el que BlackBerry ha recibido muchas críticas, es que la actual API de Java deja de ser compatible. Esto significa que los desarrolladores Java que escriban código para dispositivos BlackBerry deben reorientarse a una de las tecnologías anteriormente mencionadas. Como no todos los desarrolladores están dispuestos a hacer esto, hay cierta preocupación respecto a que un gran número de ellos "abandonen el barco" y se reorienten a las plataformas de la competencia. Además, dado que no hay vía de migración para las aplicaciones de la generación actual, los desarrolladores tendrán que volver a escribir desde cero para la nueva plataforma. Esto es necesario porque el núcleo del nuevo sistema operativo se basa en QNX³, un sistema operativo de tiempo real. Por otra parte, la nueva plataforma ofrece nuevas

¹ github.com/blackberry

² developer.blackberry.com/platforms/bb10

³ www.qnx.com

oportunidades, por ejemplo para los desarrolladores web y Android, que pueden migrar fácilmente sus aplicaciones.

C Native SDK

El BlackBerry NDK es compatible con muchos estándares abiertos que permiten a los desarrolladores llevar sus aplicaciones ya existentes a la plataforma. Para comenzar tienes el sitio web Native Dev⁴. Escribir código con Native SDK permite que tu aplicación esté tan cerca del hardware como sea posible. El BlackBerry 10 Native SDK incluye todo lo necesario para desarrollar programas que se ejecutan en BlackBerry OS 10: un compilador, un enlazador, librerías, y un extenso entorno de desarrollo integrado (IDE). Está disponible para Windows, Mac y Linux.

Los pasos básicos de desarrollo son los siguientes:

- Solicitar una cuenta de firmado y sus claves
- Instalar el SDK nativo⁵
- Instalar y configurar el simulador⁶
- Configurar el entorno para desarrollo y despliegue
- Crear tu primer proyecto
- Ejecutar aplicaciones de ejemplo

Como novedad, BlackBerry ha añadido soporte a Scoreloop⁷ en el NDK. Scoreloop es una tecnología que soporta juegos sociales móviles. Esto permite a los desarrolladores integrar atributos sociales en sus juegos, mientras que preserva el

⁴ developer.blackberry.com/native/beta/

⁵ developer.blackberry.com/native/download

⁶ developer.blackberry.com/native/download

⁷ developer.blackberry.com/native/documentation/bb10/com.qnx.doc.scoreloop.lib_ref/topic/overview.htm

aspecto específico de cada uno de ellos. Algunas de las características disponibles en la actualidad incluyen:

- Perfil de usuario
- Tablas de clasificación
- Retos
- Premios y logros

C++ Cascades SDK

Desarrollar aplicaciones con C++ y Cascades es otra opción. Cascades ha sido diseñado para permitir a los desarrolladores crear una aplicación BlackBerry nativa dando un importante soporte a la fácil implementación de la interfaz de usuario. El marco Cascades separa la lógica de aplicación del motor de renderizado de la interfaz. En la aplicación, los controles de interfaz de usuario declarados, sus propiedades y comportamiento se definen en un lenguaje de marcado llamado Qt Modeling Language (QML)⁸. Cuando se ejecuta la aplicación, el motor de renderizado de interfaz de usuario muestra los controles de la misma y aplica las transiciones y los efectos que hayan sido especificados. El Cascades SDK ofrece las siguientes características:

- Interfaz de usuario Cascades y APIs de plataforma
- Herramientas para desarrollar tu interfaz en C++, QML, o ambos
- Capacidad de beneficiarse de controles básicos de la interfaz y crear nuevos
- Comunicación por redes móviles y WiFi
- Grabación y reproducción de archivos multimedia
- Almacenamiento y retorno de datos
- Gestión de certificados y herramientas criptográficas

⁸ en.wikipedia.org/wiki/QML

El entorno Cascades se basa en el entorno de desarrollo de aplicaciones Qt. Esta arquitectura permite a Cascades aprovechar los modelos Qt de objetos, eventos y threading. Los slots y señales en Qt permiten una potente y flexible comunicación entre objetos. El framework Cascades incorpora características de clases Qt fundamentales (tales como QtCore, QtNetwork, QtXml, QSql, y otras) y compila sobre ellas. Esto permite a los desarrolladores definir las cosas en lugar de programarlas, por ejemplo, sólo tienen que definir la duración y el tipo de una animación, en lugar de programarla. Este enfoque es similar a iOS con Core Animation. Debido a su marcado parecido a JSON, QML puede incluso ser escrito por desarrolladores JavaScript experimentados.

Para ayudar a los desarrolladores con este nuevo enfoque en la construcción de interfaces de usuario, existe una herramienta llamada Cascades Builder. Está incluida en el QNX Momentics IDE y permite a los desarrolladores diseñar una interfaz de usuario mediante una interfaz visual. Cuando se realiza un cambio en el código, se puede ver el efecto inmediato en la vista de diseño. El desarrollador no tiene necesidad de programar un control, puede simplemente arrastrar y soltar.

Si eres un diseñador, el Cascades Exporter⁹ es para tí. Este plug-in de Adobe Photoshop recorta y reescala las imágenes y las empaqueta en un archivo tmz (en recursos de imagen comprimidos, recortados y mejorados con metadatos). Estos archivos de recursos puede ser utilizados fácilmente por un desarrollador con el QNX Momentics IDE.

Para más información está disponible el sitio Cascades Dev¹⁰.

⁹ developer.blackberry.com/cascades/documentation/design/cascades_exporter/

¹⁰ developer.blackberry.com/cascades/

HTML5 WebWorks

Si eres un desarrollador Web/JavaScript, puedes utilizar tus habilidades para escribir aplicaciones para BlackBerry. Hay dos herramientas relevantes que puedes utilizar:

La primera es WebWorks SDK¹¹. Entre otras características, permite escribir páginas web regulares y luego compilarlas como aplicaciones nativas de BlackBerry con facilidad. Si quieres imitar el estilo de la interfaz de usuario BlackBerry en HTML, hay un proyecto en GitHub que te ayudará. Se llama BBUI.js¹². Ofrece un amplio y detallado CSS para hacer que tu página web estándar se visualice como una aplicación nativa BlackBerry. En esta aproximación debes utilizar atributos de datos para mejorar el código HTML.

La segunda herramienta es Ripple Emulator¹³. Se trata de una extensión del navegador Chrome que actúa como un simulador de dispositivos BlackBerry 10 para aplicaciones WebWorks. También emula características específicas de hardware, tales como el acelerómetro y el sensor GPS. Incluso puedes utilizarlo para empaquetar e implementar tu aplicación sin tener que pasar a través de la línea de comandos.

Es bueno saber que BlackBerry ofrece soporte WebGL acelerado por hardware, y que puedes realizar la depuración y perfilado del dispositivo móvil a través del WebInspector como una función integrada.

Para obtener más información sobre el desarrollo con WebWorks hay un micro-site de HTML5 Dev¹⁴.

¹¹ developer.blackberry.com/html5/download/sdk

¹² github.com/blackberry/bbUI.js

¹³ developer.blackberry.com/html5/download/ripple

¹⁴ developer.blackberry.com/html5

Adobe Air

Si eres un desarrollador AIR puedes agregar BB10 como nuevo canal de distribución. Utilizarás el BlackBerry 10 SDK para Adobe AIR para crear aplicaciones para dispositivos BlackBerry.

Puedes utilizar el SDK con las APIs de Adobe ActionScript y Adobe Flex para crear o portar aplicaciones BlackBerry. Estas APIs proporcionan algunos componentes únicos de interfaz de usuario y temas predefinidos, así como listeners de eventos específicos de dispositivos BlackBerry. Utilizando las APIs de Adobe Flash Builder, tu aplicación también puede acceder a funciones únicas de dispositivos móviles, como el acelerómetro y la información de geolocalización. Además, puedes aprovechar las características de BlackBerry SDK Native 10 mediante el desarrollo de extensiones nativas de AIR, ANE (del inglés AIR Native Extensions).

Para comenzar a desarrollar tu aplicación Adobe AIR:

- Descarga e instala VMware Player para Windows or VMware Fusion para Mac
- Descarga el simulador BlackBerry 10 Simulator
- Descarga el BlackBerry 10 SDK para Adobe AIR
- Comienza a desarrollar con Adobe Flash Builder, Powerflasher FDT o herramientas de línea de comandos

Para mayor información, visita el website dedicado a éste tema¹⁵.

Android en Tiempo de Ejecución (Runtime)

Puedes utilizar el BlackBerry Runtime para ejecutar aplicaciones Android 2.3.3 de la plataforma BlackBerry 10. Para utilizar el runtime, primero debes volver a empaquetar tus aplicaciones

¹⁵ developer.blackberry.com/air/

Android en el formato de archivo BAR, que es el necesario para que una aplicación se ejecute en BlackBerry 10.

Como desarrollador, tendrás que utilizar una de las herramientas siguientes para volver a compilar la aplicación. Estas herramientas también comprueban cómo es de compatible tu aplicación para ser ejecutada en BlackBerry 10, ya que algunas de las APIs del SDK de Android pueden no ser compatibles, o pueden serlo sólo parcialmente con la plataforma BlackBerry.

- **Plug-in de recompilación para Eclipse:** La principal ventaja de la utilización de esta herramienta es su capacidad de comprobar niveles de compatibilidad, compilar, depurar y ejecutar aplicaciones en BlackBerry PlayBook, BlackBerry Tablet Simulator, BlackBerry 10 Dev Alpha Simulador y dispositivos BlackBerry 10, todo sin salir de Eclipse. También puedes usar este plug-in para firmar tu aplicación antes de distribuirla. Si deseas probar la aplicación sin firmarla, puedes utilizarlo para crear e instalar un token de depuración en el dispositivo de destino o en el simulador.
- **Compilador online:** La ventaja principal de la BlackBerry Packager para aplicaciones de Android es que se puede utilizar para volver rápidamente a compilar tu aplicación para Android utilizando sólo tu navegador. Puedes probar la compatibilidad de la aplicación, volver a compilarla como un archivo BAR compatible con BlackBerry Tablet OS o BlackBerry 10, y después firmarlo para que pueda ser distribuido a través de la tienda BlackBerry App World.
- **Herramientas recompiladoras de línea de comandos:** Una de las principales ventajas de utilizar el BlackBerry SDK para aplicaciones de Android es que se puede utilizar para recompilar múltiples aplicaciones Android desde el formato de archivo APK al formato de archivo BAR. Además, también puedes utilizar este conjunto de herramientas de

línea de comandos para comprobar la compatibilidad de tus aplicaciones Android, firmarlas, crear tokens de depuración y un certificado de desarrollador.

Si deseas obtener más información acerca de cómo ejecutar aplicaciones Android en BB10, visita el sitio web¹⁶.

Blackberry App Generator

Si no eres un desarrollador, BlackBerry proporciona una manera fácil de generar una aplicación sencilla para BB10 con el BlackBerry App Generator¹⁷. Esa página web genera una aplicación basada en fuentes de información como

- RSS feeds
- Tumbler
- Facebook
- YouTube
- Flickr

y otras. Genera una aplicación master-detail que se puede personalizar con un logotipo y una selección de colores. Para una aplicación de noticias sencilla este enfoque es totalmente correcto, pero no esperes obras maestras tipo CNN.

¹⁶ developer.blackberry.com/android

¹⁷ blackberryappgenerator.com/blackberry/



Testeo

BlackBerry continúa proporcionando un simulador para teléfonos BB10 como una descarga independiente¹⁸. Este simulador permite ejecutar una aplicación en un PC/Mac/Linux de la misma manera que se ejecutaría en un dispositivo BlackBerry real. Para asistirte en tus pruebas, el simulador viene con una aplicación llamada controller. Esta utilidad te permite simular cosas tales como el nivel de la batería, la posición GPS, NFC o la inclinación del dispositivo y, por lo tanto, comprobar cómo reacciona tu aplicación en escenarios reales.

Firma

Muchas clases y características de la plataforma de nivel de seguridad crítico (por ejemplo, la creación de redes o APIs de archivo) requieren que la aplicación esté firmada para que el desarrollador pueda ser identificado. Este último paso en el desarrollo de una aplicación para BlackBerry a menudo es difícil.

Si quieres probar tu aplicación sin firmar en un dispositivo físico, es necesario solicitar un archivo llamado token de depuración. Esta token permite a un dispositivo específico BB10 ejecutar aplicaciones sin firmar. Para realizar este procedimiento de configuración necesitas solicitar un archivo de firma (cliente-PBDT-xxxxx.csj) a través del BlackBerry Orden Key Form¹⁹. Después de recibir el archivo por email podrás instalar un token de depuración con las herramientas de línea de comandos.

¹⁸ developer.blackberry.com/devzone/develop/simulator/

¹⁹ www.blackberry.com/SignedKeys/codesigning.html

Después de realizar esta configuración, también podrás ejecutar aplicaciones sin firmar en tu dispositivo. Ten en cuenta que esto requiere ser hecho en cada dispositivo por separado.

Si quieres publicar tu aplicación en BlackBerry App World necesitas una clave de firma. Las claves de firma se solicitan a través del formulario BlackBerry Orden Key Form²⁰. Para ayudarte con este proceso de configuración, BlackBerry ofrece una guía paso a paso en esta página web²¹ que te guiará en el proceso.

Distribución

Al igual que con todas las versiones anteriores del sistema operativo de BlackBerry, las aplicaciones para BB10 se distribuyen a través de BlackBerry App World²². La necesaria cuenta de proveedor se puede crear en el Portal de Proveedores para BlackBerry App World²³.

En el caso de aplicaciones de pago, los desarrolladores obtener una participación en los ingresos del 70%.

La otra opción es una distribución corporativa. Esto te permite lanzar una aplicación interna en tu organización en lugar de ponerla a disposición pública para cualquier usuario, lo cual es adecuado para aplicaciones B2B. Si deseas obtener más información acerca de la distribución corporativa, por favor visita el sitio web dedicado²⁴.

²⁰ www.blackberry.com/SignedKeys/codesigning.html

²¹ developer.blackberry.com/CodeSigningHelp/codesignhelp.html

²² appworld.blackberry.com/

²³ appworld.blackberry.com/isvportal

²⁴ developer.blackberry.com/distribute/enterprise_application_distribution.html



iOS

El Ecosistema

Breve Historia de iOS

Apple anunció iOS en la MacWorld 2007 (que por aquel entonces se conocía simplemente como OS X, en base a Mac OS X, el sistema operativo de la línea de productos Macintosh), junto con el primer iPhone, el cual fue lanzado, con iPhone OS 1.0, el 29 de junio de 2007. Desde entonces, Apple ha lanzado cada año una nueva generación del iPhone acompañada de una nueva versión de iOS, en algún momento entre junio y octubre. En septiembre de 2012 se hizo pública la última versión disponible hasta la fecha, la 6.0.

Dispositivos Ejecutando iOS

Actualmente, Apple vende múltiples dispositivos (con diversas configuraciones) que funcionan con iOS:

- iPhone
- iPod touch
- iPad
- iPad mini
- Apple TV

Con la excepción de Apple TV, todos estos dispositivos incluyen la App Store y pueden ejecutar aplicaciones de terceros.

La mayoría de los dispositivos ejecutan la versión más reciente de iOS durante dos años o más después de su lanzamiento inicial, por lo que los desarrolladores deben considerar esto cuando crean una aplicación. Utilizar hardware antiguo

habitualmente significa disponer de menos recursos, como ciclos de CPU y RAM, y en algunos casos diferentes tamaños y/o resoluciones de pantalla.

Existe una lista detallada de dispositivos iOS, sus capacidades y las versiones de iOS compatibles en la Wikipedia¹.

Ventas de Dispositivos y Aplicaciones

Según las informaciones de Apple, que suelen ser hitos anunciados en eventos especiales de comunicación, hasta junio de 2012 se han vendido más de 400 millones de dispositivos iOS. Dado que las ventas de dispositivos iOS siguen ganando impulso, incluso después de cinco años, un gran número de estos dispositivos se puede considerar en uso activo y utilizando iOS 5 o iOS 6.

Desde enero de 2013, la App Store ofrece más de 775.000 aplicaciones de terceros desarrolladores, que en conjunto han sido descargadas más de 40 mil millones de veces, aportando más de siete mil millones de dólares a sus desarrolladores, según Apple².

Descripción General de la Tecnología

Frameworks & Lenguaje(s)

Dado que iOS se basa en Mac OS X, utiliza una gran parte de los mismos entornos de desarrollo y tecnologías, a excepción de la capa de Cocoa Touch (que gestiona y renderiza la interfaz de usuario) y algunos pequeños frameworks que son únicos para cada uno de sus sistemas. Esto facilita que un buen número de aplicaciones utilicen una base de código similar y sólo tengan

¹ en.wikipedia.org/wiki/List_of_iOS_devices

² www.apple.com/pr/library/2013/01/07App-Store-Tops-40-Billion-Downloads-with-Almost-Half-in-2012.html

que variar la interfaz de usuario, que debe ser rediseñada en cualquier caso para dispositivos táctiles.

La mayoría de los frameworks para iOS suministrados por Apple están escritos en Objective-C (o sobre APIs Objective-C proporcionadas por otro backend), que es un runtime ligero sobre C inspirado en Smalltalk, con plena compatibilidad sobre C. Pocos entornos suministran APIs de C, la mayoría son utilizados para la programación de audio y video. El sistema también soporta el desarrollo en C++ y Objective-C++, e incluye frameworks estándar para esos lenguajes.

Antes del lanzamiento de iOS, Objective-C llevaba una existencia un tanto sombría con niveles de popularidad tan bajos como del 0,03% en el índice TIOBE³, gracias a su uso casi exclusivo en Mac OS X. En diciembre de 2007 era el lenguaje de programación número 57 en popularidad y desde entonces se ha ido alzando hasta llegar al tercer puesto en el año 2012, justo por detrás de Java y C, tras convertirse en el "Lenguaje de Programación del Año" en 2011. Su popularidad sigue creciendo de manera mucho más rápida que Java y C.

A lo largo de los últimos años, Apple ha realizado numerosas mejoras tanto en el runtime de Objective-C como en el compilador LLVM para agregar nuevas características al lenguaje, como la gestión automática de memoria, blocks (una forma de closures) y propiedades sintetizadas automáticamente, de las cuales la mayoría de los desarrolladores se benefician directamente al tener que escribir menos código.

Apple ofrece una gran cantidad de recursos en su sitio web para desarrolladores iOS⁴, incluyendo descargas de software, videos de aprendizaje, guías de iniciación, documentación, códigos de ejemplo y foros.

³ www.tiobe.com/index.php/content/paperinfo/tpci/index.html

⁴ developer.apple.com/devcenter/ios/

La mayoría de estos recursos contienen información muy valiosa, como las Human Interface Guidelines (Directrices de Interfaz Humana), que todos los desarrolladores deberían haber leído.

Xcode y Sus Alternativas

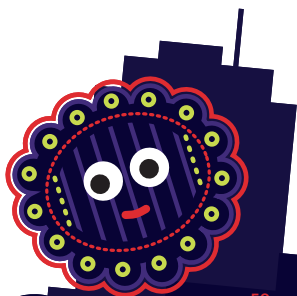
Para el desarrollo en iOS (y Mac OS X), Apple suministra su propia suite de herramientas de desarrollo de forma totalmente gratuita, incluyendo las siguientes aplicaciones:

- **Xcode:** entorno de desarrollo integrado
- **Instruments:** analizador de rendimiento que se ejecuta sobre DTrace
- **Dashcode:** entorno de desarrollo para widgets del Dashboard (Mac OS X) y otro contenido relacionado con HTML
- **iOS Simulator:** simula un entorno iOS para testeo rápido

Una IDE comercial alternativa a Xcode es JetBrains' AppCode⁵, una aplicación Java con algunas características de mayor calado que las que Xcode ofrece. Para aquellos que están tratando de evitar Objective-C, hay entornos completos como MonoTouch⁶, que ofrece incluso soporte multiplataforma.

⁵ www.jetbrains.com/objc/

⁶ xamarin.com/monotouch



Testeo & Debugging

Las herramientas de desarrollo para iOS incluyen soporte para pruebas tanto unitarias como automatizadas sobre la interfaz de usuario a través del framework UIAutomation. A través de la línea de comandos de Xcode, dichas herramientas pueden ser incluidas en sistemas de integración continua para las pruebas de aceptación automática.

Existen numerosas herramientas y frameworks externos de automatización de tests. Algunos son comerciales y propietarios, sin embargo la mayoría actualmente son open source, incluso de empresas que desean vender servicios para que tu testeo automatizado sea más simple y más potente. Muchas de las herramientas externas de automatización de tests requieren que el desarrollador incorpore una librería en una compilación especial de su aplicación. La librería permite a los tests interactuar con la aplicación. Ten cuidado de mantener la compilación especial separada de la destinada a distribución en la tienda de aplicaciones, de lo contrario es posible que sea rechazada cuando la envíes a la App Store.

Xcode incluye tanto gdb y lldb y utilizará automáticamente el apropiado según qué compilador se está utilizando para la aplicación. Aunque el desarrollador mantiene el control total sobre el depurador desde el símbolo del sistema, Xcode ofrece una interfaz de usuario para las acciones frecuentes, como la configuración, edición y eliminación de puntos de interrupción (breakpoints), variables de vista y contenido de la memoria.

Estos instrumentos también ofrecen varias características para ayudar a los desarrolladores a descubrir bugs, cuellos de botella de rendimiento y problemas de gestión de memoria.

Como su nombre indica, el simulador de iOS es sólo eso, un simulador, y como tal tiene diferentes características de tiempo de ejecución respecto a los dispositivos iOS reales, no pudiendo

emular un entorno de iOS completo. Por lo tanto, una serie de cuestiones que surgirán con dispositivos reales simplemente no saldrán a la superficie cuando se prueben las aplicaciones en el simulador. Afortunadamente, todos los tests se pueden ejecutar en dispositivos reales también, y Apple permite a los desarrolladores aprovisionarse de hasta 100 dispositivos iOS para ejecutar sus aplicaciones con fines de prueba y demostración.

Incorporar y gestionar software de pruebas de terceros (si no se dispone de propio) se realiza fácilmente a través de TestFlight⁷ y HockeyApp⁸, los cuales ofrecen diversas funciones útiles, como la firma automática de código, informe de fallos y actualización en la misma aplicación para los beta testers.

Aprende Más

Blogs & Newsletters

Un buen número de desarrolladores publican regularmente información valiosa sobre novedades en sus blogs. Uno muy notable es el de Mike Ash⁹, en el que publica una serie muy interesante de preguntas y respuestas sobre desarrollo en Objective-C y Cocoa. Otros muchos blogs interesantes se pueden encontrar a través de www.planetcocoa.org.

iOS Dev Weekly¹⁰ es un importante boletín que recopila las noticias más interesantes, código open source, herramientas, así como consejos de diseño y marketing en una cómoda lista, y se la envía a los desarrolladores suscritos todos los viernes.

⁷ testflightapp.com/

⁸ hockeyapp.net

⁹ www.mikeash.com/pyblog/

¹⁰ iosdevweekly.com

Conferencias

Debido a la creciente popularidad de iOS, se producen numerosas conferencias centradas en iOS por todo el mundo cada año, demasiadas para enumerarlas aquí. Hay dos de ellas, sin embargo, que merecen especial mención:

- Cada año, en junio, Apple mantiene su Worldwide Developer Conference (WWDC, Conferencia Mundial de Desarrolladores)¹¹. De una semana de duración en San Francisco, incluye muchas conferencias simultáneas sobre Mac OS X e iOS con sesiones de desarrollo de los ingenieros de Apple, así como de laboratorios prácticos, donde los asistentes pueden pedir a los ingenieros de Apple asesoramiento acerca de los problemas que se enfrentan al desarrollar sus aplicaciones.
- La conferencia europea más grande y de mayor éxito en torno a Mac OS X e iOS es la NSConference¹², se celebra cada año alrededor de marzo en Inglaterra.

Ambas conferencias agotan las entradas en cuestión de días, si no horas, así que si estás pensando en ir a cualquiera de ambas, planifícate con mucha anticipación y suscríbete a alertas sobre la salida a venta de tickets.

¹¹ developer.apple.com/wwdc/about/

¹² ideveloper.tv/nsconference/





Java ME (J2ME)

El Ecosistema

J2ME (o Java ME como se le llama oficialmente) es la plataforma de aplicaciones móviles más antigua que aún sigue estando ampliamente utilizada. Desarrollada por Sun Microsystems, empresa que fue comprada posteriormente por Oracle, J2ME está diseñada para funcionar principalmente en teléfonos de gama media. Ha tenido mucho éxito en ese segmento del mercado, con una abrumadora mayoría de teléfonos compatibles. J2ME también es compatible de forma nativa con Symbian y smartphones BlackBerry que no se basen en BB10.

El mayor inconveniente de J2ME es que, debido a su antigüedad y segmento primario de mercado, no resulta especialmente bien parada en comparación a las plataformas modernas de smartphones como Android, iPhone, BlackBerry y Windows Phone: ofrece un conjunto de APIs con inferiores capacidades, a menudo se ejecuta en hardware menos potente y tiende a generar menos ingresos para el desarrollador. Como consecuencia, la popularidad de J2ME en la comunidad de desarrolladores ha disminuido significativamente en los últimos años.

Entonces, ¿por qué querrías desarrollar para J2ME? Sobre todo por una razón: el alcance del mercado. En el segundo trimestre de 2012, las ventas de smartphones aún representaban sólo el 36,7% del total de las ventas de teléfonos móviles en todo el mundo¹. La mayoría de los dispositivos son aún teléfonos de gama media que normalmente soportan Java ME.

¹ gartner.com/it/page.jsp?id=2120015 gartner.com/it/page.jsp?id=2120015
gartner.com/it/page.jsp?id=2120015

Así que si tu modelo de negocio se basa en el acceso a tantos clientes potenciales como sea posible, J2ME podría ser una estupenda opción, especialmente si te diriges a mercados como algunos países de África o la India.

Sin embargo, si su modelo de negocio se basa en las ventas directas de tu aplicación, o si tu aplicación tiene que hacer uso de las funciones y hardware más avanzados, las plataformas para smartphones son la mejor opción.

Prerrequisitos

Para desarrollar una aplicación Java ME, necesitarás::

- El Java SDK² (no el Java Runtime Environment) y un IDE de tu elección, por ejemplo Eclipse Pulsar para Desarrolladores Móviles³, NetBeans⁴ con su plug-in Java ME o IntelliJ⁵. Los principiantes habitualmente eligen NetBeans.
- Un emulador, como Wireless Toolkit⁶, el Micro Emulator⁷ o un SDK o emulador de un proveedor específico.
- Dependiendo de tu configuración, puedes necesitar un ofuscador como ProGuard⁸. Si realizas aplicaciones profesionalmente, probablemente desearás utilizar una herramientas como Maven⁹ o Ant¹⁰.
- Quizás te interese echarle un vistazo a J2ME Polish¹¹, el framework open source para desarrollar tu aplicación para varios dispositivos.

- 2 oracle.com/technetwork/java/javame/downloads
- 3 eclipse.org
- 4 netbeans.org
- 5 jetbrains.com
- 6 oracle.com/technetwork/java/download-135801.html
- 7 microemu.org
- 8 proguard.sourceforge.net
- 9 maven.apache.org
- 10 ant.apache.org
- 11 j2mepolish.org



Completar la instalación y las instrucciones de configuración va más allá del alcance de esta guía, consulta la documentación de las respectivas herramientas.

También puedes descargar y leer los JavaDocs para las tecnologías y APIs más importantes: puedes descargar la mayoría de ellos desde www.jcp.org. Para APIs específicas del fabricante, la documentación suele estar disponible en la página web del vendedor, (por ejemplo, la Nokia UI API¹²).

Implementación

La plataforma Java ME es bastante sencilla: comprende la Connected Limited Device Configuration (CLDC)¹³ y el Mobile Internet Device Profile (MIDP)¹⁴, y ambos son bastante fáciles de entender. Ellos forman la base de cualquier entorno J2ME y proporcionan un conjunto estandarizado de capacidades a todos los dispositivos J2ME. Dado que tanto CLDC y MIDP fueron diseñados hace una década, el conjunto predeterminado de las capacidades que ofrecen es rudimentario según los estándares actuales.

Los fabricantes pueden complementar estas rudimentarias capacidades mediante la implementación de diversas Java Specification Requests opcionales (JSR). Hay JSRs para todo, desde el acceso al calendario, agenda y sistema de archivos (JSR 75), hasta utilizar el GPS (JSR 179) y el NFC (JSR 257). Para obtener una lista completa de JSRs relacionadas con el desarrollo de Java ME, visita el Java Community Process' Lista por JCP Technology¹⁵.

¹² www.developer.nokia.com/Community/Wiki/Nokia_UI_API

¹³ java.sun.com/products/cldc/overview.html

¹⁴ java.sun.com/products/midp/overview.html

¹⁵ jcp.org/en/jsr/tech?listPOR=1&listPORType=platform

Es muy importante saber que los JSR que desees utilizar pueden no estar disponibles para todos los dispositivos, y que las implementaciones varían significativamente entre modelos de teléfonos, por lo que las capacidades disponibles en un dispositivo pueden no estar disponibles en otro, incluso si los dos dispositivos tienen hardware similar.

Runtime Environment

Las aplicaciones J2ME son llamados MIDlets. El ciclo de vida de un MIDlet es muy simple: sólo se puede iniciar, pausar y destruir. En la mayoría de dispositivos un MIDlet se pausa automáticamente cuando se minimiza, no se puede ejecutar en segundo plano. Algunos dispositivos admiten la ejecución concurrente, por lo que es posible que las aplicaciones se ejecuten en segundo plano. Sin embargo, esto generalmente requiere el uso de las APIs específicas del proveedor y/o se apoya en el comportamiento específico del dispositivo, lo que puede causar problemas de fragmentación.

Los MIDlets también se ejecutan de forma aislada unos de otros y están muy limitados en su interacción con el sistema operativo subyacente; estas funciones se proporcionan estrictamente a través de JSRs opcionales (por ejemplo, JSR 75) y APIs específicas del proveedor.

Creando Interfaces de Usuario

Puedes crear la interfaz de usuario de tu aplicación de varias formas:

1. Componente LCDUI de alto nivel: utilizas componentes estándares de la interfaz, tales como formularios y listas
2. Componentes LCDUI de bajo nivel: controlas manualmente cada píxel de tu interfaz utilizando funciones gráficas de bajo nivel
3. SVG: dibujas la interfaz en SVG y usas las APIs de JSR 226¹⁶ o JSR 287¹⁷

Además, encontrarás que algunos fabricantes ofrecen características adicionales de interfaz de usuario. Por ejemplo, la última serie de teléfonos de Nokia (Nokia Asha) utiliza tanto los paradigmas de interfaz Full Touch¹⁸ como Touch and Type¹⁹, dependiendo del modelo del dispositivo. El Nokia UI API se amplió con el fin de permitir a los desarrolladores hacer mejor uso de estos paradigmas en sus aplicaciones. Del mismo modo, Samsung proporciona capacidades pinch zoom en sus últimas APIs Java ME²⁰.

¹⁶ www.jcp.org/en/jsr/detail?id=226

¹⁷ jcp.org/en/jsr/detail?id=287

¹⁸ www.developer.nokia.com/Resources/Library/Full_Touch/

¹⁹ www.developer.nokia.com/Community/Wiki/Nokia_UI_API_1.1b

²⁰ developer.samsung.com/java/technical-docs/Multi-Touch-in-Samsung-Devices

También existen herramientas que te pueden ayudar con el desarrollo de interfaces de usuario. Todas ellas utilizan gráficos de bajo nivel para crear interfaces de usuario más atractivas y más potentes, que son posibles con los componentes estándar de alto nivel LCDUI.

1. J2ME Polish²¹: Esta herramienta separa el diseño en CSS y emplea HTML para la interfaz. Es compatible con el framework de alto nivel LCDUI
2. LWUIT²²: Framework de interfaz inspirado en Swing
3. Mewt²³: Usa XML para definir la interfaz

Un aspecto muy importante a considerar en el diseño de tu interfaz de usuario es la resolución de pantalla típica para dispositivos Java ME. La gran mayoría de dispositivos Java ME tienen una de las siguientes resoluciones: 240x320, 176x208, 176x220, 128x160, 128x128 o 360x640 píxeles. Con mucho, la más popular es 240x320, mientras que 360x640 es una resolución común para la gama alta de dispositivos Java ME (por lo general los que ejecutan Symbian o BlackBerry), y 176x208/220 es una resolución común para los dispositivos de gama baja. También te encontrarás dispositivos que tienen estas resoluciones en disposición horizontal, por ejemplo 320x240 en lugar de 240x320 píxeles.

Manejar tantas resoluciones diferentes puede ser un desafío para muchos. El mejor enfoque es crear diseños de interfaz de usuario que puedan escalar bien en todos ellos, de la misma manera que las páginas web escalan bien a través de diferentes tamaños de ventana de navegador. También se pueden crear interfaces de

²¹ j2mepolish.org

²² lwuit.java.net/

²³ mewt.sourceforge.net



usuario personalizadas para cada resolución, aunque esto no es recomendable, ya que consume mucho tiempo y es costoso y dado a errores.

Otro aspecto que vale la pena considerar es el tamaño de los elementos de tu aplicación (assets), en especial el material gráfico. Siempre que sea posible deben ser optimizados, con el fin de mantener el tamaño de la aplicación tan pequeño como sea posible. Esto se traduce en descargas más baratas para tus usuarios (ya que requieren menos tráfico de datos) y un mayor alcance de mercado (algunos dispositivos tienen un límite en el tamaño máximo de la aplicación). Una gran herramienta gratuita para este tema es PNGGauntlet²⁴, que puede optimizar tus archivos gráficos sin comprometer su calidad.

A pesar de las limitaciones de la plataforma, es bastante factible crear excelentes apariencias gráficas e interfaces Java ME fáciles de usar, especialmente si usas una de las herramientas mencionadas anteriormente.

Testeado

Debido a la fragmentación de las distintas implementaciones de Java ME, probar las aplicaciones es vital. Ponlas a prueba en una combinación de dispositivos tan pronto y tan a menudo como sea posible. Algunos emuladores son bastante buenos, pero hay algunas cosas que tienen que ser probadas en los dispositivos.

Afortunadamente, fabricantes como Nokia²⁵ y Samsung²⁶ proporcionan acceso remoto de pago o incluso gratis a dispositivos seleccionados.

²⁴ pnggauntlet.com

²⁵ forum.nokia.com/rda

²⁶ developer.samsung.com

Testeado automático

Hay varios entornos de pruebas unitarias para Java ME, incluyendo J2MEUnit²⁷, MoMEUnit²⁸ y CLDC Unit²⁹; probar el sistema y la interfaz de usuario es más complejo dado el modelo de seguridad de J2ME, sin embargo JInjector³⁰ es un entorno flexible de inyección de código que soporta dichas pruebas. La cobertura de código también se puede obtener con JInjector.

Portar

Uno de los puntos fuertes del entorno Java para dispositivos móviles es que está respaldado por un estándar, por lo que puede ser implementado por fabricantes competidores. La desventaja es que el estándar ha de ser interpretado, y este proceso puede causar diferencias en las implementaciones individuales. Esto da lugar a todo tipo de bugs y comportamiento no estándar. En las secciones siguientes se esbozan diferentes estrategias para portar tus aplicaciones a todos los teléfonos y plataformas Java ME.

Mínimo Común Denominador

Puedes prevenir muchos problemas de portabilidad si limitas la funcionalidad de tu aplicación al mínimo común denominador. En el mundo J2ME, esto normalmente significa CLDC 1.0 y MIDP 1.0. Si sólo vas a publicar tu aplicación en los países/regiones más desarrollados, puedes centrarte en CLDC 1.1 y MIDP 2.0 como el mínimo común denominador (sin ninguna API adicional ni soporte JSR).

²⁷ j2meunit.sourceforge.net

²⁸ momeunit.sourceforge.net

²⁹ snapshot.pyx4me.com/pyx4me-cldcunit

³⁰ www.code.google.com/p/jinjector

Dependiendo de la región objetivo para la aplicación, también podrías tener que considerar el uso de tecnología Java para la industria inalámbrica (JTWI, JSR 185) o la arquitectura de servicios móviles (MSA, JSR 248) como línea de base. Ambas extensiones están diseñadas para asegurar una implementación común de los JSRs más populares, son apoyadas por muchos dispositivos modernos y ofrecen muchas más capacidades a las aplicaciones. Sin embargo, en algunas regiones como África, América del Sur o la India, debes ser consciente de que el uso de estos estándares puede limitar el número de usuarios potenciales, debido a que los dispositivos más comunes en estas regiones no implementan esas extensiones.

Utilizar el enfoque del mínimo común denominador suele ser fácil: hay menos funcionalidades a considerar. Sin embargo, la experiencia del usuario puede verse afectada si tu aplicación se limita en esta manera, especialmente si deseas portarla más tarde a plataformas smartphones. Así que este enfoque es una buena opción para aplicaciones simples, pero para aplicaciones intensivas, ricas en funcionalidades, no es el camino a seguir.

Frameworks para Portabilidad

Los entornos enfocados a portabilidad pueden ayudarte a lidiar con la fragmentación, adaptando de forma automática tu aplicación a diferentes dispositivos y plataformas. Tales frameworks típicamente cuentan con los siguientes componentes:

- Librerías cliente que simplifican el desarrollo
- Herramientas que convierten código y recursos en paquetes de aplicaciones
- Bases de datos de dispositivos que proveen de información acerca de los mismos
- Compiladores cruzados que portan tu aplicación a diferentes plataformas

Para Java ME, algunas de las opciones que se pueden elegir son: Celsius de Mobile Distillery³¹, que se licencia por mes, y J2ME Polish de Enough Software³², que está disponible tanto bajo la licencia GPL Open Source como bajo licencia comercial. También es posible hacer el camino inverso, de C++ a Java ME, con el software libre MoSync SDK³³.

Para obtener más información acerca del desarrollo multi-plataforma y las herramientas disponibles, consulta el capítulo "Hacia Multiplataforma".

Los buenos frameworks permiten utilizar código específico de plataforma y dispositivo en tus proyectos, de manera que puedas ofrecer la mejor experiencia de usuario. En otras palabras: un buen framework no oculta la fragmentación por dispositivo, pero hace que sea más manejable.

Firma

El estándar de Java para dispositivos móviles diferencia entre las aplicaciones que han sido firmadas y las que no. Algunas funciones de los teléfonos están disponibles sólo para las aplicaciones de confianza. Qué características se ven afectadas y qué sucede si la aplicación no está firmada pero utiliza una de esas características, depende en gran medida de la implementación. En un teléfono, al usuario se le puede pedir que habilite la funcionalidad sólo una vez, en otro se le pedirá cada vez que se utilice esa característica, y en un tercer dispositivo no se podrá utilizar la función en absoluto. La mayoría de las implementaciones

31 mobile-distillery.com

32 enough.de

33 mosync.com



también diferencian entre las autoridades de certificación que han firmado una aplicación.

Las aplicaciones firmadas por el fabricante de un dispositivo disfrutan del nivel más alto de seguridad y pueden acceder a todas las APIs de Java disponibles en el teléfono, y las firmadas por el operador están igualmente consideradas de confianza en aquellos dispositivos que tengan instalado un certificado de clave pública de ese operador.

Las solicitudes firmadas por JavaVerified³⁴, Verisign³⁵ o Thawte³⁶ están en el nivel más bajo de seguridad.

Para empeorar las cosas, no todos los teléfonos tienen todos los certificados raíz necesarios y, en el pasado, algunos fabricantes de dispositivos muy conocidos llegaron a eliminar todos los certificados raíz. El resultado es un buen lío, así que considera firmar tu aplicación sólo cuando sea necesario, o sea, cuando se publique en una tienda de aplicaciones o cuando es absolutamente indispensable el acceso a las funciones de seguridad restringidas. Sin embargo, en algunos casos, una tienda de aplicaciones puede ofrecerse a firmar por tí, como es el caso de Nokia Store.

Otra opción es considerar el uso de un proveedor de servicios de certificación y pruebas, dejando la parte compleja para ellos. Intertek³⁷ es probablemente el mayor proveedor de este tipo.

³⁴ javaverified.com

³⁵ verisign.com

³⁶ thawte.com

³⁷ www.intertek.com/wireless-mobile

Distribución

Las aplicaciones J2ME se pueden instalar directamente en un teléfono de múltiples maneras; los métodos más utilizados son mediante una conexión Bluetooth, a través de una conexión directa por cable o Over-the-Air (OTA). Sin embargo, las tiendas de aplicaciones son probablemente la forma más eficiente de distribuir tus aplicaciones. Ellos manejan los pagos, hospedaje y publicidad, teniendo una participación en los ingresos por estos servicios. Algunas de las tiendas más efectivas son:

- Handmark³⁸ y Mobile Rated³⁹ ofrecen tiendas de aplicaciones independientes de fabricantes y operadoras.
- GetJar⁴⁰ es uno de los distribuidores más antiguos de aplicaciones móviles, y no sólo Java.
- Nokia Store⁴¹ se enfoca a usuarios Nokia en todo el mundo y ofrece un margen de beneficio para el proveedor del 70% si la compra es con tarjeta de crédito, y del 60% si es por operadora.
- Las operadoras también participan, tales como Orange⁴² y O2⁴³.

Básicamente, casi todo el mundo en el ámbito móvil ha lanzado una tienda de aplicaciones. Se puede encontrar una lista de las tiendas de aplicaciones disponibles (no sólo las que venden aplicaciones J2ME) en el WIP App Store Catalogue⁴⁴.

³⁸ store.handmark.com

³⁹ mobilerated.com

⁴⁰ getjar.com

⁴¹ publish.ovi.com

⁴² www.orangepartner.com/distribute

⁴³ mobileapps.o2online.de

⁴⁴ www.wipconnector.com/appstores/

Véase también el capítulo especial sobre tiendas de aplicaciones de esta guía para obtener más información.

Aún mas, hay varios proveedores que ofrecen soluciones para el aprovisionamiento de aplicaciones Java a través de una conexión Bluetooth, incluyendo Waymedia⁴⁵ y Futurlink⁴⁶.

Aprende Más

Si quieres aprender más sobre el desarrollo en Java ME, a continuación tienes algunos recursos que te pueden ayudar.

Online

Como Java ME es una de las plataformas móviles más antiguas que todavía se utilizan, es fácil encontrar tutoriales y recursos relacionados, por ejemplo los disponibles en J2ME Salsa⁴⁷.

Además, el sector J2ME es un rico entorno open source. Se pueden encontrar proyectos interesantes a través del blog opensource.ngphone.com.

También encontrarás proyectos fascinantes en la página de Mobile and Embedded de java.net⁴⁸, por ejemplo, el proyecto sobre Bluetooth Marge⁴⁹.

Libros

Con el paso de los años se han llegado a escribir un buen número de libros de Java ME, por ejemplo:

- **Beginning J2ME: From Novice to Professional** por Jonathan Knudsen y Sing Li

⁴⁵ waymedia.it

⁴⁶ www.futurlink.com

⁴⁷ j2mesalsa.com

⁴⁸ community.java.net/mobileandembedded/

⁴⁹ marge.java.net

- **Pro Java Me Apps: Building Commercial Quality Java ME Apps** por Ovidiu Iliescu
- **Pro J2ME Polish: Open Source Wireless Java Tools Suite** por Robert Virkus, sobre el desarrollo con J2ME Polish
- **LWUIT 1.1 for Java ME Developers** por Biswajit Sarkar, sobre el desarrollo con LWUIT

Desafortunadamente, debido a la decreciente popularidad de Java ME, muy pocos libros se han escrito sobre ella en los últimos años.



Windows Phone

El Ecosistema

Desde su introducción a finales de 2010, la plataforma de Windows Phone ha ganado altos índices de satisfacción entre sus clientes, incluso superando al iPhone¹, pero esa satisfacción no se ha convertido en una cuota de mercado significativa. En una escala global, sólo el 2% de todos los smartphones utilizaban Windows Phone en el tercer trimestre de 2012². Una notable excepción es Italia, con una cuota de mercado de más del 10%³. La anterior cuota de mercado máxima, en Brasil, ha disminuido por debajo del 10% con respecto a 2012, aparentemente debido a algunas decisiones sobre precios OEM.

Los comercializadores de Windows Phone son Nokia, HTC, Samsung y ZTE. LG no ha lanzado un dispositivo con esta plataforma.

En el cuarto trimestre de 2012 Microsoft presentó Windows Phone 8, que comparte núcleo con Windows 8 y Windows RT, los sistemas operativos de sobremesa y tablets, respectivamente. Las aplicaciones existentes de Windows Phone 7 seguirán funcionando en Windows Phone 8, pero no podrán acceder a nuevas características y capacidades de hardware (a menos que utilices alguna inteligente carga dinámica de clases). A partir de Windows Phone 8 también puedes desarrollar aplicaciones utilizando C/C++ y DirectX.

- 1 wmpoweruser.com/q3-2012-survey-finds-windows-phones-outscore-iphone-5-in-customer-satisfaction
- 2 thenextweb.com/2012/11/01/android-grabs-75-0-market-share-in-q3-followed-POR-14-9-for-ios-and-4-3-for-blackberry
- 3 guardian.co.uk/technology/2012/oct/02/windows-phone-europe-market

En 2012 la cantidad de aplicaciones disponibles en Windows Marketplace se duplicó, y el usuario promedio de Windows Phone instala actualmente 54 aplicaciones⁴.

Implementación

El desarrollo de Windows Phone se lleva a cabo en C/C++, C# o VB.NET, usando el Microsoft Visual Studio IDE o el Expression Blend⁵. Las aplicaciones son creadas utilizando Silverlight, principalmente para aplicaciones orientadas a eventos, y DirectX, principalmente para juegos basados en un "bucle de juego", aunque ambas tecnologías pueden ser utilizadas en una sola aplicación. En el caso de Windows Phone 7 también se pueden crear juegos basados en XNA. Aunque este tipo de juegos aún funcionan en Windows Phone 8, sin embargo XNA no es compatible con aplicaciones destinadas a Windows Phone 8 en exclusiva. Además, puedes crear aplicaciones basadas en HTML5 con PhoneGap⁶, sin embargo el desarrollo web no se trata en este capítulo.

⁴ blogs.windows.com/windows_phone/b/wpdev/archive/2012/12/26/reflecting-on-2012-scale-and-opportunity.aspx

⁵ dev.windowsphone.com

⁶ phonegap.com

Metro/Modern UI

La característica más obvia de Windows Phone es la interfaz fácil de usar que se centra en tipografía y contenido. Este paradigma de interfaz de usuario llamado Metro o Modern UI⁷ se ha extendido a la Xbox 360 y también a Windows 8, y contiene los siguientes principios:

- **Content not Chrome** elimina adornos innecesarios y permite que el contenido sea en sí mismo el foco principal principal de atención. También debes abstenerse de utilizar todos los píxeles disponibles, y tener en cuenta que los espacios en blanco dan equilibrio y énfasis al contenido.
- **Alive in motion** añade profundidad a un diseño plano con animaciones ricas.
- **Typography is beautiful** da protagonismo al uso de fuentes en Metro. La fuente Segoe de Windows Phone inspirada en Helvetica encaja con el enfoque moderno de la interfaz.
- **Authentically digital** el diseño no trata de imitar objetos del mundo real, sino que se centra en las interacciones disponibles para soluciones digitales.

Deberás aceptar los modernos principios de diseño Metro/Modern UI en tu aplicación, sobre todo cuando portes aplicaciones ya existentes. Los diseñadores encontrarán inspiración e información en la documentación de diseño de Microsoft⁸. Un aspecto importante del diseño es la alineación de los elementos de tu interfaz de usuario a la rejilla estándar de Windows Phone. Las actuales plantillas de diseño de Silverlight incluyen la rejilla, sólo tienes que quitar los comentarios en el código fuente de la página XAML. También son importantes para la experiencia global los “live tiles”, son widgets pequeños que

⁷ [wikipedia.org/wiki/Metro_\(design_language\)](http://wikipedia.org/wiki/Metro_(design_language))

⁸ dev.windowsphone.com/diseño

residen en la pantalla de inicio. Puedes actualizarlos mediante programación o incluso remotamente mediante notificaciones push.

Nuevas Características

Las nuevas características del Windows Phone 8.0 SDK⁹ incluyen:

- Soporte a C/C++
- Soporte a DirectX 9_3, XAudio2 y DirectXMath
- Interoperabilidad entre DirectX/C++ y XAML/C#
- Texto a voz
- Voz a texto y comandos por voz
- Bloqueo de pantallas personalizados, iconos e imágenes de fondo
- Nuevos tamaños de cuadrícula y plantillas
- Nuevo control de mapas basados en Nokia, con mapas offline y guía a pie y en coche
- Mejor integración de la cámara con aplicaciones Lenses y carga automática de tareas en segundo plano
- Mejor grabación en video y mayor y más preciso control de la cámara
- Integración del monedero
- Geotracking en segundo plano
- Almacenaje de entradas de calendario
- Salvar contactos sin necesidad de confirmación del usuario
- Comunicación aplicación a aplicación o web a aplicación utilizando protocolos personalizados
- Registro de extensiones de archivo
- Acceso a tarjetas SD
- Acceso por Bluetooth en bajo nivel; comunicación aplicación a aplicación y aplicación a dispositivo

⁹ developer.nokia.com/Community/Wiki/What's_new_in_Windows_Phone_8

- Lectura y escritura de etiquetas NFC
- Selección y compartición de etiquetas NFC
- Más resoluciones: 800x480, 1280x720 & 1280x768
- Compra desde aplicaciones
- Mayores opciones de integración para aplicaciones VOIP
- Añadir y eliminar canciones
- Redes sociales: compartir cualquier medio
- Networking: IPv6 y soporte a sockets entrantes
- Distribución de aplicaciones corporativas
- Librerías de clase portables que permiten compartir código entre Windows Phone, Windows 8 y .NET

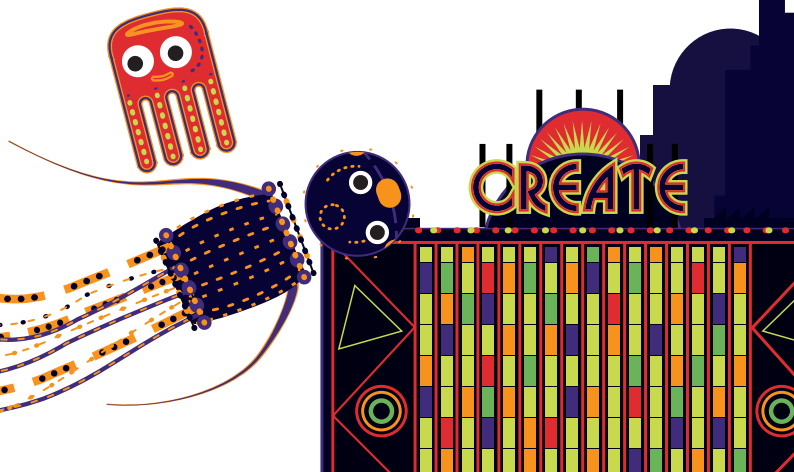
SDK

El SDK de Windows Phone SDK es gratuito e incluye ediciones "Express" tanto de Visual Studio 2012 como de Expression Blend, y requiere Windows 8 Pro para ejecutarse ya que el SDK utiliza 'virtualización'. Mientras que las ediciones Express soportan todo lo necesario para desarrollar para Windows Phone, muchas características adicionales sólo están disponibles en las ediciones comerciales, siendo la más destacada la opción de crear o utilizar las librerías de clases portables. El SDK también incluye un emulador de dispositivos para ejecutar código, que utiliza la aceleración por hardware y cuenta con un panel de control para el control de la ubicación, la simulación del acelerómetro y otros.

Es importante tener en cuenta qué plataforma aprovechar en la construcción de tu aplicación.

Usa C# o VB y Silverlight si...	Usa C++ y DirectX si...
...quieres crear una aplicación orientada a eventos o un juego casual.	...quieres crear un juego 2D o 3D.
...quieres utilizar los controles estándar de Windows Phone.	...quieres gestionar objetos tales como modelos, mallas, sprites, texturas y animaciones.
...quieres enfocarte a Windows Phone y Windows 8; reutilizando mucho código.	...quieres enfocarte a Windows Phone, Windows 7/8, y Xbox 360; reutilizando mucho código.

Mientras que el escenario más común es el uso de Silverlight para aplicaciones y DirectX para juegos, también se pueden crear juegos Silverlight y aplicaciones DirectX, dependiendo de tus necesidades. Asimismo es posible alojar Direct3D dentro de tu aplicación Silverlight. Esto podría ser utilizado para mostrar un modelo 3D dentro de una aplicación Silverlight orientada a eventos, o para crear fácilmente menús atractivos basados en Silverlight sobre un juego DirectX.



Motores de Juego

Hay algunos nuevos motores de juego para Windows Phone 8 con capacidades de aplicación nativa:

- Cocos2d-x¹⁰
- Havok¹¹
- Marmalade¹²
- OGRE¹³
- Unity 3D¹⁴

Servicios

Hay disponibles notificaciones push¹⁵ que pueden actualizar los live tiles de tu aplicación. También puedes considerar el uso del espacio gratuito en nube SkyDrive y la integración con otros servicios de Windows Live¹⁶ para tu aplicación.

Multitarea y Ciclo de Vida de la Aplicación

Windows Phone tiene una capacidad limitada de multitarea que suspende las aplicaciones en segundo plano y permite el cambio rápido entre aplicaciones. Los únicos procesos que pueden ejecutarse en segundo plano, después de que una aplicación se haya abandonado, son la reproducción de audio, el seguimiento de la ubicación y la transferencia de archivos. Las aplicaciones también pueden programar la ejecución de código arbitrario en segundo plano en un intervalo (código que se conoce como

¹⁰ cocos2d-x.org/news/76

¹¹ havok.com/products/havok-windows-ecosystem

¹² madewithmarmalade.com/windows-phone-8

¹³ ogre3d.org/2012/10/30/ogre-now-supports-windows-phone-8

¹⁴ blogs.unity3d.com/2012/10/30/unity-windows-phone-8-demonstrated-at-microsoft-build-conference

¹⁵ msdn.microsoft.com/library/windowsphone/develop/ff402558

¹⁶ msdn.microsoft.com/live

agentes en segundo plano, Background Agents). A dichos agentes se les permite el uso limitado de los recursos y pueden ser detenidos o ignorados si el sistema operativo determina que el teléfono debe conservar los recursos.

Las aplicaciones suspendidas en segundo plano pueden ser cerradas automáticamente si el sistema operativo determina que los recursos son necesarios en otro lugar.

Para crear la apariencia de una aplicación que nunca se cerró, Windows Phone tiene un ciclo de vida de aplicaciones bien documentado llamado Tombstoning¹⁷. Para hacer posible Tombstoning, el framework de Windows Phone proporciona lo necesario para realizar operaciones durante las diferentes etapas del ciclo de vida de la aplicación (por ejemplo, el almacenamiento en caché y la restauración de datos y estados de interfaz de usuario). Con Windows Phone 8 ofrece a los desarrolladores una nueva capacidad de "terminar aplicaciones rápidamente".

Testeo y Analíticas

Puedes hacer pruebas unitarias de las aplicaciones utilizando el Windows Phone Test Framework¹⁸ o el Silverlight Unit Test Framework¹⁹.

Para desarrollos basados en comportamiento está disponible el Windows Phone Test Framework por Expensify²⁰, pero el proyecto parece haber quedado abandonado.

Hay varias opciones para los desarrolladores que deseen recopilar datos y análisis en tiempo de ejecución. Localytics²¹

¹⁷ msdn.microsoft.com/library/windowsphone/develop/ff817008

¹⁸ wptestlib.codeplex.com

¹⁹ nuget.org/packages/WPToolkitTestFx

²⁰ github.com/Expensify/WindowsPhoneTestFramework/

²¹ localytics.com/docs/windows-phone-7-integration

y Flurry²² proporcionan herramientas de análisis y servicios compatibles con Windows Phone 7. Los desarrolladores también pueden utilizar Silverlight Analytics Framework²³ para conectarse a una variedad de servicios de seguimiento de terceros, como Google Analytics. A partir de la actualización del Windows Phone SDK 7.1, hay disponibles sólidas herramientas de monitorizado de rendimiento en Visual Studio.

Distribución

Las aplicaciones para Windows Phone se distribuyen principalmente a través del servicio Microsoft Marketplace. Aunque el contenido de la aplicación es revisado y restringido de una manera similar a la App Store de Apple, Microsoft proporciona directrices generales para la solicitud de publicación, disponibles en App Hub²⁴. Aunque las herramientas de desarrollo se proporcionan de forma gratuita, se requiere una cuenta de pago de App Hub para implementar una aplicación en dispositivos y distribuirla en el Marketplace. Actualmente, una cuenta de desarrollador cuesta \$99 (USD) por una suscripción anual e incluye 100 solicitudes de publicación de aplicaciones gratuitas e ilimitadas de aplicaciones de pago. La tarifa no se aplica a los estudiantes en el DreamSpark²⁵ ni durante el primer año para desarrolladores Nokia Publish. El Marketplace también ofrece una distribución beta por tiempo limitado y ofrece un hub corporativo para empresas²⁶. Puedes utilizar el Windows Phone Marketplace Test Kit²⁷ para probar la aplicación en local antes de enviarla para su publicación.

²² flurry.com/flurry-analytics.html

²³ msaf.codeplex.com

²⁴ dev.windowsphone.com

²⁵ www.dreamspark.com

²⁶ msdn.microsoft.com/library/windowsphone/develop/jj206943

²⁷ msdn.microsoft.com/library/windowsphone/develop/hh394032

Para aplicaciones de pago, el entorno Windows Phone ofrece la posibilidad de determinar si la aplicación está en "modo de prueba" o no, y en consecuencia limitar su uso. Microsoft recomienda específicamente evitar la limitación por tiempo en las versiones de prueba (por ejemplo, una versión de prueba de treinta minutos) y, en cambio, sugiere limitar las funciones disponibles²⁸.

Para la monetización basada en anuncios hay varias opciones. Microsoft tiene su propio Microsoft Advertising Ad Control²⁹ (actualmente disponible en 18 países), mientras que Nokia³⁰, Smaato³¹, Inneractive³², AdDuplex³³ y Google³⁴ ofrecen soluciones alternativas de publicidad. Para obtener más información acerca de la monetización, consulta el capítulo dedicado en esta guía.

Aprende Más

Visita dev.windowsphone.com para noticias, herramientas de desarrollo y foros.

El equipo de desarrollo publica mensajes en su blog windowsteamblog.com/windows_phone o en su cuenta de Twitter @wpdev. Para encontrar una gran colección de recursos para desarrolladores y diseñadores, visita windowsphonegeek.com y reddit.com/r/wpdev.

Actualmente hay varios controles integrados en el sistema operativo que no están incluidos en el SDK de Windows Phone, tales como menús de contexto, selector de fechas, y otros.

²⁸ msdn.microsoft.com/library/windowsphone/develop/ff967558

²⁹ advertising.microsoft.com/mobile-apps

³⁰ www.developer.nokia.com/Distribute/NAX

³¹ smaato.com

³² inner-active.com

³³ adduplex.com

³⁴ developers.google.com/mobile-ads-sdk/

Estos controles están disponibles como parte de Silverlight Toolkit para Windows Phone, disponible en phone.codeplex.com. Otros proyectos populares de Windows Phone son coding4fun.codeplex.com y mvvmlight.codeplex.com. Para la inspección del árbol visual, bindings y propiedades de las interfaces de usuario basados en XAML en tiempo de ejecución, está disponible xamlspy.com.

Hay varios libros electrónicos publicados de forma gratuita, por ejemplo *Windows Phone Programming in C# (Windows Phone Version 7.5)*³⁵ o *Silverlight for Windows Phone Toolkit In Depth*³⁶.

También se pueden encontrar muchos tutoriales de vídeo en la cobertura de la conferencia Build de Channel 9, en channel9.msdn.com/events/build/2012?T=windows-phone.

³⁵ blogs.msdn.com/b/uk_faculty_connection/archive/2011/11/23/windows-phone-free-ebook-and-demos.aspx

³⁶ windowsphonegeek.com/WPToolkitBook





Windows 8

Windows 8 es primer sistema operativo de Microsoft que se ejecuta en tablets y PCs por igual. Comparte la filosofía de diseño moderno de interfaz de usuario con Windows Phone, y las nuevas aplicaciones Windows Phone se pueden ejecutar también en cualquier sistema Windows 8. Las aplicaciones Windows 8 se pueden desarrollar en C++, C#/VB.NET o JavaScript. Todos ellos son ciudadanos de primera clase en este ecosistema, ya que todos tienen igual acceso a las APIs del Windows Runtime (WinRT).

Mientras la adopción inicial de Windows 8 fue mayor en números absolutos en comparación con Windows 7, la adopción relativa ha sido más lenta¹. Las ventas de PCs han seguido disminuyendo, sin embargo esto podría ser debido a los escasos modelos de pantalla táctil disponibles. Su recepción ha fluctuado entre radicalmente hostil y positivamente eufórica. En 2012, la tienda de Windows 8 creció en 35.000 aplicaciones en tan sólo dos meses, aproximadamente el mismo número de aplicaciones publicadas por la tienda de Windows Phone en su primer año de existencia.

Prerrequisitos

Para desarrollar aplicaciones de estilo Modern UI necesitas Visual Studio 12 y Blend, estando las versiones Express disponibles de forma gratuita. Puedes instalar Windows 8 en una máquina virtual, lado a lado con tu actual sistema operativo o como sistema operativo principal. Tener un monitor táctil habilitado

¹ phonearena.com/news/Wait-so-Windows-8-is-not-outpacing-Windows-7-adoption-rate_id37212

ayuda a pulir la experiencia del usuario para las tablets, pero Windows 8 trabaja igualmente bien cuando se usa un ratón.

Técnicamente, también necesitas una licencia de desarrollador; sin embargo, ésta se adquiere automáticamente y de forma gratuita.

Implementación

Aunque puedes, simplemente, seleccionar el lenguaje que coincida con los conocimientos de tu equipo, es importante entender las diferencias en capacidades ofrecidas por las distintas opciones:

	C/C++	C#/VB.NET	JavaScript
WinRT	sí	sí	sí
Silverlight/ XAML	sí	sí	no
HTML	no	no	sí
DirectX	sí	sí (con SharpDX)	no
Compartición de código	Antiguas aplicaciones Windows nativas, Xbox profesional otras platafor- mas...	Aplicaciones antiguas .NET Windows, indie Xbox, aplicaciones Windows Phone...	Websites, aplicaciones HTML5...

Si deseas utilizar DirectX con C#, puedes usar SharpDX.org, anxframework.codeplex.com o basarte en librerías de juego tales como monogame.codeplex.com. Como Windows Phone 8 y Windows 8 comparten el mismo kernel, es fácil compartir código entre estas plataformas. La manera más sencilla para ello parece ser el uso compartido de librerías .NET entre ellas.

Partes de las Aplicaciones

Cada aplicación Windows 8 comprende varias partes:

- **App tile** representa la aplicación en la pantalla de inicio y puede mostrar contenido relevante al usuario, incluso cuando la aplicación no está en ejecución;
- **Splash screen** es mostrada de manera optativa cuando la aplicación comienza;
- **App bar** contiene acciones y comando relevantes por el contexto;
- **Content area** muestra tu aplicación en diferentes tipos de vista tales como pantalla completa o snapped, véase el apartado “Vistas y Factores de Forma”;
- **Charms** permite al usuario iniciar interacciones con la aplicación, véase el apartado “Contratos en la Aplicación”.

APIs de Windows Runtime

Las APIs de WinRT están documentadas en MSDN², y contienen los temas habituales, como parsear JSON/XML sobre geolocalización, sensores, gestión de medios y las APIs de red. Pero WinRT tiene algunos conceptos bastante más interesantes, por ejemplo:

- `Windows.Security.Authentication.Live`: Usa Windows Live como mecanismo de autenticación con zero click single sign-on³ y comparte datos entre varios dispositivos del usuario, usando SkyDrive⁴
- `Windows.Security.Authentication.Web`: Se integra con servicios web que usan OAuth u OpenID (Facebook, Twitter, etc.)

² msdn.microsoft.com/library/windows/apps/br211369

³ msdn.microsoft.com/library/live/hh826544

⁴ msdn.microsoft.com/library/live/hh826521

- `Windows.Security.Credentials`: Permite el almacenamiento y acceso a contraseñas
- `Windows.ApplicationModel.Contacts`: Accede o provee de datos de contactos

Contratos en la Aplicación

Windows 8 ofrece charms para cada aplicación a los que puedes acceder deslizándote hacia el lado derecho de la pantalla.

Hay cinco charms: buscar, compartir, iniciar, dispositivos y configuración. Mediante la implementación de los contratos (contracts), puedes conectarte a estos charms y compartir información entre aplicaciones. Declara los contratos en el archivo de manifiesto de tu `Package.appx`, y a continuación implementa la funcionalidad requerida.

Puedes encontrar los siguientes contratos⁵:

- **Search** busca en el contenido de tu aplicación. Opcionalmente, puedes dar sugerencias de búsqueda mientras el usuario escribe. La funcionalidad relevante está en el namespace `Windows.ApplicationModel.Search`.
- **Share** proporciona una manera de compartir datos entre una fuente y la aplicación de destino, al declarar el contrato correspondiente. Si deseas que tu aplicación comparta datos, debes ponerla a disposición del mayor número de formatos de datos como sea posible para aumentar el número de aplicaciones potencialmente destinatarias. Puedes utilizar formatos estándar tales como texto, HTML, imágenes o crear tus propios formatos. La aplicación objetivo puede devolver opcionalmente un quicklink que apunta a los datos consumidos. Por ejemplo, puedes compartir una imagen con Facebook o Flickr y obtener un enlace. Las

⁵ msdn.microsoft.com/library/windows/apps/hh464906

clases relevantes están en `Windows.ApplicationModel.DataTransfer.ShareTarget`.

- **Play To** reproduce datos con dispositivos conectados, por ejemplo haciendo streaming de video a un televisor con DLNA. Comienza con el namespace `Windows.ApplicationModel.PlayTo`.
- **Settings** permite al usuario ajustar la configuración dependiente de contexto desde cualquier punto de tu aplicación. Define tu configuración con la ayuda de `Windows.UI.ApplicationSettings`.
- **App to App Picking** permite abrir o salvar archivos de la aplicación desde el interior de otra aplicación. Sus clases están en el namespace `Windows.Storage.Pickers`.

No dupliques la funcionalidad de charms en otras partes de tu aplicación. Eso sólo confundirá a los usuarios. Deberías, por ejemplo, no incluir un campo de búsqueda específico, a menos que la búsqueda sea una tarea relevante de la aplicación.

Windows 8 tiene muchas extensiones y contratos más, la lista completa está disponible en msdn.microsoft.com/library/windows/apps/hh464906.

Vistas y Factores de Forma

Las aplicaciones de Windows 8 pueden funcionar en diferentes modos⁶:

- **full screen** es el modo por defecto, tanto en orientación vertical como horizontal. Tu aplicación utilizará toda la pantalla disponible para sumergir al usuario completamente con `ApplicationLayoutState.FullScreen`.
- **snapped** and **filled** son modos en los que las aplicaciones se muestran lado con lado. Deberías cambiar tu layout con-

⁶ msdn.microsoft.com/library/windows/apps/hh465371

forme a esto, pero manteniendo el estado de tu aplicación y con, al menos, las principales funciones fácilmente accesibles mediante `ApplicationLayoutState.Filled` y `ApplicationLayoutState.Snapped`.

Para ser notificado acerca de los cambios de layout, escucha el evento

`Windows.UI.ViewManagement.ApplicationLayout.GetForCurrentView().LayoutChanged`. Con él puedes incluso cambiar el estado programáticamente: Cuando tu aplicación está en modo snapped y tu usuario selecciona una función que requiere un modo diferente, puedes llamar a `ApplicationLayout.TryUnsnap()`.

Autoescalado

Windows 8 se ejecuta en dispositivos con diferentes resoluciones de pantalla y densidades de píxeles. Dependiendo de la resolución, las aplicaciones se escalan automáticamente a:

- 1366 x 768 (100%)
- 1920 x 1080 (140%)
- 2560 x 1440 (180%)

Los desarrolladores web deberían utilizar gráficos SVG y media queries CSS cuando sea posible. Los desarrolladores XAML pueden utilizar naming schemes para los recursos, de manera que el más adecuado se seleccione automáticamente (como por ejemplo `image.scale-100.jpg`, `image.scale-140.jpg` e `image.scale-180.jpg`). También debes utilizar los recursos con dimensiones múltiples de 5px, para que no se produzcan deformaciones de píxel cuando autoescale.

Push

Puedes enviar datos y hasta imágenes a tus aplicaciones usando el servicio de notificación de Windows Windows Notification Service (WNS)⁷. Esto también te permite actualizar las live tiles de tu aplicación. Usar WNS es gratis, y puedes usar también el Windows Azure Mobile Services⁸ para simplificar la implementación de un servidor push.

Single Sign On

Windows 8 proporciona servicios de administración de credenciales de usuario⁹ y utiliza Microsoft Account para su autenticación de usuario. Puedes aprovechar esta información para proporcionar un inicio de sesión único a tus aplicaciones, lo que te permitiría identificar al usuario directamente sin necesidad de otra autenticación.

Distribución

Las aplicaciones Windows 8 se pueden distribuir solamente a través de Windows Store¹⁰. La participación estándar en los ingresos aumenta del 70% al 80% cuando tu aplicación factura más de \$25.000 (USD). Windows Store funciona en más de 200

⁷ msdn.microsoft.com/library/windows/apps/hh465460

⁸ www.windowsazure.com/en-us/develop/mobile/

⁹ msdn.microsoft.com/library/windows/apps/br211367

¹⁰ msdn.microsoft.com/library/windows/apps/hh694084

países y regiones y más de 100 idiomas, así que puedes tener un alcance global. También puedes distribuir versiones limitadas en funcionalidad o tiempo, utilizar funciones de compra dentro de la aplicación o integrar anuncios, y está permitido el uso de proveedores externos de sistemas de pago.

Las aplicaciones son gestionadas por el cliente, no por el dispositivo. De esta forma un usuario puede utilizar su aplicación en varias plataformas, tales como un PC y un tablet.

Antes de vender aplicaciones, es necesario obtener una cuenta de Windows Store que cuesta \$49 (USD) al año para los particulares y \$99 (USD) para las empresas.

Aprende Más

Tu punto de partida para el desarrollo en Windows 8 es dev.windows.com, y encontrarás consejos y trucos de diseño en design.windows.com.

También puedes debatir sobre problemas de desarrollo en social.msdn.microsoft.com/Forums/en-US/category/windowsapps.

Encontrarás código de ejemplo en code.msdn.microsoft.com/windowsapps, en diversos proyectos en codeplex.com y en los ejemplos disponibles en msdn.microsoft.com/library/windows/apps/br211375. La hoja de ruta para los desarrolladores de aplicaciones ofrece un buen repaso a la planificación, el diseño y el desarrollo de aplicaciones Windows 8, en msdn.microsoft.com/library/windows/apps/xaml/br229583.





Hacia Multiplataforma

Tantas plataformas y tan poco tiempo: Esa frase define perfectamente la situación que tenemos en el mundo móvil. Hay plataformas más que suficientes para elegir: Android, BlackBerry 10, Firefox OS, iOS, Tizen, Windows 8 y Windows Phone están, o probablemente estarán, entre algunas de las plataformas smartphone y tablets más importante, mientras Brew MP y Java ME dominan en el ámbito de los teléfonos de gama media (plataformas listadas no por importancia, sino por orden alfabético).

La mayoría de los promotores de aplicaciones, citando cierta famosa letra de una canción de Queen, le dirán al desarrollador: "Lo quiero todo, lo quiero todo, lo quiero todo... ¡y lo quiero ahora!", así que la elección puede muy bien estar entre invertir en múltiples equipos paralelos de desarrollo, o adoptar una estrategia multiplataforma.

Diferencias Clave Entre Plataformas Móviles

Si deseas hacer disponible tu aplicación a través de diferentes plataformas, tienes que superar algunos obstáculos. Ciertos retos son más fáciles de superar que otros:

Lenguaje de Programación

A estas alturas ya te habrás dado cuenta de que las plataformas móviles publican sus propios SDK, lo que te permite desarrollar aplicaciones en los lenguajes de programación soportados por las plataformas.

Sin embargo, estos lenguajes tienden a pertenecer a alguna de las pocas familias de lenguajes raíz. La tabla siguiente ofrece una descripción general de éstos y de las plataformas que los soportan:

Lenguaje	Compatible de forma nativa en la plataforma, ya sea el idioma primario o uno específico para la creación de aplicaciones	Se admite como opción en la plataforma, se puede utilizar como alternativa al lenguaje nativo, pero generalmente no proporciona el mismo nivel de acceso a características de la plataforma
ActionScript	BlackBerry 10, BlackBerry PlayBook OS (QNX)	ninguno
C, C++	bada, BlackBerry 10, BlackBerry PlayBook OS, Brew MP, Symbian, Windows 8, Windows Phone 8	Android (parcialmente, utilizando el NDK), iOS (parcialmente)
C#	Windows 8, Windows Phone	ninguno
Java	Android, BlackBerry, Java ME devices	Symbian
JavaScript	BlackBerry PlayBook OS, Firefox OS, Tizen, Windows 8	BlackBerry (WebWorks), Nokia (WRT)
Objective-C	iOS	ninguno

- 1 Supported natively POR the platform, for example either the primary or only language for creating applications
- 2 Supported natively POR the platform, for example either the primary or only language for creating applications

Los frameworks multiplataforma pueden solventar las barreras de lenguaje de diferentes maneras:

- **Tecnologías Web**
- **Interpretación**
- **Compilación Cruzada**

La mayoría de los frameworks también proporcionan un conjunto de APIs multiplataforma que permiten acceder de un modo común a determinadas funciones del dispositivo o plataforma, como por ejemplo a las capacidades de geolocalización. Para funciones tales como mensajería SMS también puedes usar APIs de red que son independientes del dispositivo¹.

Versiones de SO

Las plataformas evolucionan y, tarde o temprano, ofrecerán características específicas de versión que desearás aprovechar. Esto añade otra capa de complejidad a tu aplicación y también un desafío para las herramientas multiplataforma: a veces se quedan atrás cuando una nueva versión del sistema operativo es publicada.

Interfaz de Usuario y UX

Un obstáculo difícil para el enfoque de multiplataforma se origina por los diferentes patrones de interfaz de usuario y de experiencia de usuario que prevalecen en las plataformas individuales.

Es relativamente fácil crear una interfaz de usuario agradable que funcione de la misma manera en varias plataformas. Este enfoque, sin embargo, puede olvidar sutilezas importantes que sólo están disponibles en una única plataforma y que podrían mejorar drásticamente la experiencia de usuario. También

¹ www.developergarden.com/apis/

ignoraré las diferencias respecto a la filosofía de diseño de las plataformas: si bien iOS se esfuerza en conseguir un diseño realista en el que las aplicaciones se parecen a sus contrapartes del mundo real, la interfaz Metro de Windows Phone se centra en lograr una experiencia "auténticamente digital", en la que el contenido se enfatiza sin marcos. Otro desafío clave con una interfaz de usuario multiplataforma es que puede comportarse de manera diferente a lo que están acostumbrados los usuarios de la interfaz de usuario nativa, resultando en que la aplicación no "funciona" para los usuarios. Un ejemplo simple es no dar soporte una tecla de hardware correctamente, como la tecla de retroceso en una plataforma determinada. Otro reto es el valle de las sombras resultante de intentar imitar elementos nativos de la interfaz, que se parecen a los originales pero no funcionan de la misma manera. En lugar de imitar controles nativos, deberías usar unos de aspecto no nativo o, simplemente, limitarte a usar los "auténticos".

Al dirigirte a los consumidores finales directamente (B2C), a menudo tienes que tener mucho más en cuenta la experiencia de usuario específica de plataforma que en aquellos casos en que el target es usuarios de negocio (B2B). En cualquier caso, debes tener en cuenta que la personalización y la adaptación de la interfaz de usuario y la UX para cada plataforma puede suponer una gran parte de tu esfuerzo de desarrollo, y que es sin duda el aspecto más desafiante de una estrategia multiplataforma.

Soporte a la Integración en Pantalla de Inicio

La integración de tu aplicación en las pantallas de inicio de los dispositivos varía mucho entre las plataformas. En iOS sólo se puede añadir una etiqueta de identificación con un número al icono de tu aplicación, en Windows Phone puedes crear live tiles que añaden información estructurada, mientras que en

Android y Symbian se puede añadir un widget completo que muestre datos arbitrarios y emplee imágenes.

Utilizar la integración de tu aplicación en este sentido podría mejorar drásticamente su interacción con los usuarios.

Soporte Multitarea

La multitarea permite a servicios en segundo plano y múltiples aplicaciones funcionar al mismo tiempo, y es otra de las características que se gestionan de manera diferente según el sistema operativo. En Android, BlackBerry y Symbian, hay servicios en segundo plano y puedes ejecutar varias aplicaciones al mismo tiempo, pero en Android el usuario no puede terminar las aplicaciones a voluntad, ya que esto se realiza automáticamente por el sistema operativo cuando los recursos disminuyen. En iOS y Windows Phone hay una selección limitada de tareas en segundo plano que pueden seguir funcionando después de salir de la aplicación. Así que, si los servicios en segundo plano pueden mejorar la operativa de tu aplicación, debes evaluar las estrategias multiplataforma cuidadosamente para asegurarte de permitir el acceso más completo a las capacidades del teléfono en este sentido.

Consumo de Batería y Rendimiento

En estrecha relación con la multitarea está el consumo que hace tu aplicación de la batería.

Mientras que la potencia de CPU se duplica aproximadamente cada dos años (la Ley de Moore dice que el número de transistores se duplica cada 18 meses), la capacidad de la batería, en contraste, se duplica sólo una vez cada siete años. Esta es la razón por la que los smartphones pasan tanto tiempo conectados a su cargador. Cuanto más cerca estás de la plataforma en una capa de abstracción multiplataforma, mejor puedes controlar el consumo de la batería y el rendimiento de

tu aplicación. Como regla general, cuanto más tiempo necesita tu aplicación para arrancar y funcionar, menor abstracción puedes conseguir.

Asimismo, algunas plataformas tienen una gran variedad de resultados en rendimiento, sobre todo Android, cuya gama de dispositivos va desde lo dolorosamente lento hasta lo super-rápido.

Servicios Push

Los servicios push son una estupenda manera de aparentar que tu aplicación está activa, incluso cuando no está en funcionamiento. En una aplicación de chat puedes, por ejemplo, enviar mensajes de chat entrantes para el usuario mediante un mecanismo push. Como en el caso anterior, la manera en que los servicios push funcionan y los protocolos que utilizan se emplean de manera diferente en cada plataforma. El tamaño de datos disponible, por ejemplo, oscila entre 256 bytes en iOS y 8 KB de BlackBerry. Los proveedores de servicios como Urban Airship² soportan el envío de datos a través de varias plataformas.

Compra desde Aplicación

Los mecanismos de compra desde aplicaciones permiten vender bienes o servicios dentro de tu aplicación. No hace falta decir que esto funciona de forma diferente en distintas plataformas. Véase el capítulo de monetización para más detalle.

Publicidad en Aplicación

Existen diferentes opciones para la visualización de anuncios en aplicaciones móviles, siendo algunas soluciones de terceros independientes del comercializador. Los servicios de publicidad específicos de plataforma, sin embargo, ofrecen mayores ingresos y una mejor experiencia de usuario. Una vez más, estos

² urbanairship.com/

servicios funcionan de manera diferente según las plataforma. El capítulo de monetización en esta guía proporciona más información sobre este tema.

Estrategias Multiplataforma

En esta sección se describen algunas de las estrategias que puedes emplear para implementar tus aplicaciones en diferentes plataformas.

Soporte Directo

Puedes soportar varias plataformas si cuentas con un equipo especializado para cada plataforma objetivo. Si bien esto puede consumir muchos recursos, es muy probable que te dé mejor integración y experiencia de usuario en cada sistema. Una vía fácil de inicio es comenzar con una plataforma y luego progresar hacia plataformas adicionales una vez que tu aplicación se ha defendido en el mundo real.

Las librerías de componentes pueden ayudarte a acelerar el desarrollo nativo, ejemplos populares se muestran en la siguiente tabla.

Librería de Componentes	Plataformas Objetivo
cocoacontrols.com	iOS
chupamobile.com	Android, iOS
verious.com	Android, iOS, HTML5, Windows Phone
windowsphonegeek.com/Marketplace	Windows Phone

Compartición de Recursos

Cuando mantienes varios equipos para diferentes plataformas puedes ahorrar mucho esfuerzo si compartes algunas estructuras de la aplicación:

- **Concepto y recursos:** Esto se hace habitualmente de forma automática: compartir las ideas y los conceptos de la aplicación, el flujo de interfaz de usuario, input y output y los recursos gráficos de la aplicación (pero debes ser consciente de la necesidad de dar soporte a las estructuras específicas de interfaz de usuario de cada plataforma).
- **Estructuras de datos y algoritmos:** Vé un paso más allá compartiendo estructuras de datos y algoritmos entre plataformas.
- **Compartición de código del modelo de negocio:** Utilizando compiladores de multiplataforma también puedes compartir el modelo de negocio entre las plataformas. Alternativamente, puedes utilizar un intérprete o una máquina virtual y un lenguaje común en un grupo de plataformas.
- **Abstracción completa:** Algunas herramientas multiplataforma te permiten abstraerte completamente del modelo de negocio, las vistas y el control de tu aplicación para diferentes plataformas.

Reproductores y Máquinas Virtuales

Los reproductores suelen proporcionar un conjunto común de APIs sobre distintas plataformas. Ejemplos famosos incluyen Flash, Java ME y Lua. Este enfoque hace que el desarrollo sea muy fácil. Te vuelves dependiente, sin embargo, del proveedor de la plataforma para incorporar nuevas características, por lo que el reto aquí se da cuando las funciones están disponibles

en una única plataforma. A menudo los reproductores tienden a utilizar una aproximación de “mínimo común denominador” respecto a las funciones ofertadas para mantener la uniformidad entre las implementaciones para varias plataformas. Generadores de código como Applause³ llevan el concepto de reproductor un paso más allá, a menudo son específicos del dominio y permiten generar aplicaciones a partir de datos. También suelen carecer de flexibilidad en comparación con las soluciones programables.

Compilación Cruzada

La compilación cruzada permite la programación en un lenguaje que es transformado en el lenguaje específico de una plataforma diferente. En términos de rendimiento a menudo es la mejor solución de multiplataforma, aunque puede haber diferencias en comparación con las aplicaciones nativas. Esto puede darse, por ejemplo, cuando ciertas construcciones de programación no pueden ser traducidas de manera óptima desde el código fuente al lenguaje de destino.

Hay tres enfoques habituales para la compilación cruzada: traducción directa del código fuente, de forma indirecta mediante la traducción del código fuente en un lenguaje intermedio abstracto, y recopilación directa en el formato binario de una plataforma. El enfoque indirecto típicamente produce código

³ applause.github.com



menos legible. Este es un problema potencial cuando se desea continuar con el desarrollo en la plataforma de destino usando el código fuente traducido como punto de partida.

(Híbridas) Aplicaciones Web

En la tabla siguiente se enumeran algunos de los frameworks de aplicaciones web disponibles. Con estos entornos se pueden crear aplicaciones web que se comportan casi como aplicaciones reales, incluyendo capacidades offline. Sin embargo, ten en cuenta que las tecnologías tienen limitaciones cuando se trata de aspectos como la integración en plataforma o rendimiento. Lee el capítulo “Tecnologías Web” para aprender más acerca del desarrollo web móvil.

Solución Web App	Licencia	Plataformas de destino
jQuery Mobile www.jquerymobile.com	MIT y GPL	Android, bada, BlackBerry, iOS, Symbian, webOS, Windows Phone
JQTouch www.jqtouch.com	MIT	iOS
iWebKit iwebkit.net	LGPL	iOS
iUI code.google.com/p/iui	BSD	iOS
Sencha Touch www.sencha.com/products/touch	GPL	Android, iOS
The M Project the-m-project.org	MIT y GPL	Android, BlackBerry, iOS, webOS

Normalmente no tienes acceso a las características de hard-

ware y elementos nativos de interfaz de usuario, por lo que en nuestra opinión no cuentan como soluciones multiplataforma “reales”; estas soluciones, por lo tanto, no aparecen en la tabla al final de este capítulo.

El desarrollo web híbrido implica insertar una webview dentro de una aplicación nativa. Esta aproximación te permite acceder a las funciones nativas desde los componentes web de tus aplicaciones y también utilizar código nativo para mejorar el rendimiento o aspectos críticos de la experiencia de usuario de tu aplicación. Las aplicaciones híbridas permiten reutilizar elementos de desarrollo web en las plataformas que hayas elegido.

ANSI C

Mientras que el HTML y la programación web se crean a partir de un nivel de abstracción muy elevado, también puedes elegir el camino opuesto utilizando ANSI C. Puedes ejecutar código ANSI C en todas las plataformas importantes, como Android, BlackBerry 10, iOS y Windows 8/Windows Phone. El principal problema con esta opción es que no puedes acceder a las APIs específicas de plataforma ni los controles de interfaz de usuario desde dentro de ANSI C. Utilizar C es especialmente adecuado para algoritmos complejos, como los codificadores de audio. Las librerías correspondientes se pueden utilizar en todos los proyectos de aplicaciones para una plataforma.

Frameworks de Aplicaciones Multiplataforma

Hay muchas soluciones multiplataforma disponibles, por lo que es difícil ofrecer una visión completa. Puedes llamarlo fragmentación, nosotros lo llamamos competencia. Una palabra de advertencia: no sabemos todas las opciones, si tienes una solución propia que está disponible públicamente, por

favor háznoslo saber a través de developers@enough.de. Un framework debe soportar al menos dos plataformas móviles para ser listado aquí.

A continuación, algunas preguntas que debes hacer al evaluar herramientas multiplataformas. No todas ellas serán relevantes para tí, así que sopesa las opciones adecuadamente. En primer lugar, ten una visión detallada de tu idea de aplicación, el contenido, su público objetivo y las plataformas de destino. También debes tener en cuenta a la competencia en las diversas plataformas, tu presupuesto de marketing y el know-how de tu equipo de desarrollo.

- ¿Cómo funciona la herramienta multiplataforma? ¿Qué lenguajes de programación y APIs puedo usar?
- ¿Puedo acceder a funciones específicas de la plataforma? ¿En caso afirmativo, cuáles?
- ¿Puedo usar componentes nativos de la interfaz de usuario? ¿En caso afirmativo, cuáles?
- ¿Puedo usar una versión específica de la plataforma como base de mi desarrollo? ¿Cómo es el código fuente traducido/generado?
- ¿Es posible la integración con la pantalla de inicio?
- ¿Puedo controlar la multitarea? ¿Hay algún servicio en segundo plano?
- ¿Cómo funciona la solución con servicios push?
- ¿Cómo puedo usar la compra desde aplicación y la publicidad en aplicación?
- ¿Se mantiene actualizado el framework a medida que evoluciona el sistema operativo?



Solución	Licencia	Input	Output
Application Craft applicationcraft.com	Comercial	HTML, CSS, JavaScript	Android, Black- Berry, iOS, Symbian, Windows Phone, mobile sites
appMobi appmobi.com	Comercial	HTML, CSS, JavaScript	Android, iOS, Kindle Fire, Nook, web
Codename One codenameone.com	Comercial	Java	Android, BlackBerry, iOS, J2ME, Windows Phone
Corona coronalabs.com (Corona Labs)	Comercial	JavaScript	Android, iOS
J2ME Polish j2mepolish.org (Enough Software)	Open Source + Comercial	Java ME, HTML, CSS	Android, BlackBerry, J2ME, PC
Flash Builder adobe.com/devnet/ devices.html (Adobe)	Comercial	Flash	Android, BlackBerry Playbook OS, iOS, PC
Feedhenry feedhenry.com	Comercial	HTML, CSS, JavaScript	Android, BlackBerry, iOS, Windows Phone
Kirin/JS kirinjs.org/	Open Source	JavaScript	Android, iOS
Kony One kony.com/node/4	Comercial	HTML, CSS, Ja- vaScript, RSS	Android, BlackBerry, iOS, J2ME, Symbian, Windows Phone
LiveCode runrev.com (RunRev)	Comercial	English- like	Android, iOS, PC and Web

Solución	Licencia	Input	Output
MobiForms www.mobiforms.com (MobiForms)	Comercial	Drag and Drop + MobiScript	Android, iOS, PC, Windows Mobile
Mono for Android xamarin.com/ monoforandroid (Xamarin)	Comercial	C#	Android (share code with iOS and Windows Phone)
Mono Touch xamarin.com/ monotouch (Xamarin)	Comercial	C#	iOS (share code with Android and Windows Phone)
MoSync mosync.com	Open Source + Comercial	C/C++, HTML5/JS	Android, BlackBerry, iOS, J2ME, Symbian, Windows Phone 7, Windows Mobile
NeoMAD neomades.com	Comercial	Java	Android, bada, BlackBerry, iOS, J2ME, Symbian, Windows Phone 7
PhoneGap/Cordova www.phonegap.com (Adobe/Apache)	Open Source	HTML, CSS, JavaScript	Android, BlackBerry, iOS, Symbian, Windows Phone
Qt qt.digia.com (Digia)	Open Source + Comercial	C++	PC, Symbian, MeeGo and Windows Mobile, desktop Windows, Apple & Linux OS
Rhodes rhomobile.com/products/rhodes (Motorola)	Open Source + Comercial	Ruby, HTML, CSS, JavaScript	Android, BlackBerry, iOS, Symbian, Windows Mobile, Windows Phone
Spot Specific www.spotspecific.com	Comercial	Drag and Drop, JavaScript	Android, iOS

Solución	Licencia	Input	Output
Titanium www.appcelerator.com (Appcelerator)	Open Source	JavaScript	Android, Consoles, iOS, PC
trigger.io trigger.io (Trigger Corp)	Comercial	HTML5, JavaScript	Android, iOS, Windows Phone
Verivo verivo.com	Comercial	(Visual)	Android, BlackBerry, iOS
webMethods Mobile Designer (formerly Me-tismo Bedrock) www.metismo.com (Software AG)	Comercial	Java ME	Android, bada, BlackBerry, brew, Consoles, iOS, PC, Windows Phone, Windows Mobile
XML VM xmlvm.org	Open Source + Comercial	Java, .NET, Ruby	C++, Java, JavaScript, .NET, Objective-C, Python

Motores de Juego Multiplataforma

Los juegos están muy centrados en el contenido y a menudo no necesitan ser integrados en profundidad en la plataforma, así que el desarrollo multiplataforma es a menudo más atractivo para éstos que para las aplicaciones.

Solución	Licencia	Input	Output
Cocos 2D cocos2d-x.org	Open Source	C++, HTML5, JavaScript	Android, BlackBerry, iOS, Windows 8, Windows Phone
Corona coronalabs.com (Corona Labs)	Comercial	Lua	Android, iOS, Kindle, nook
EDGELIB edgelib.com (elements interactive)	Comercial	C++	Android, iOS, PC, Symbian
Esenthel esenthel.com (elements interactive)	Comercial	C++	Android, iOS, PC
GameSalad gamesalad.com	Comercial	Drag and drop	Android, iOS, PC, web
id Tech 5 idsoftware.com (id)	Comercial	C++	Consoles, iOS, PC
Irrlicht irrlicht.sourceforge. net	Open Source	C++	Android & iOS with OpenGL-ES version, PC
IwGame drmpop.com/index. php/iwgame-engine	Open Source	C++	Android, bada, BlackBerry Playbook OS, iOS, PC

Solución	Licencia	Input	Output
Marmalade madewithmarmalade.com (Ideaworks3D)	Comercial	C++, HTML5, JavaScript	Android, bada, BlackBerry 10, BlackBerry PlayBook OS, iOS, LG Smart TV, Windows Phone
Moai getmoai.com (Zipline Games)	Comercial	Lua	Android, iOS, PC, Web
MonoGame monogame.codeplex.com	Open Source	C#, XNA	Android, iOS, PC, Windows 8
Ogre 3D ogre3d.org	Open Source	C++	Windows 8, Window Phone, PC
orx orx-project.org	Open Source	C, C++, Objective-C	Android, iOS, PC
ShiVa 3D stonetrip.com	Comercial	C++	Android, BlackBerry 10, iOS, PC, Consoles
SI02 sio2interactive.com (sio2interactive)	Comercial	C, Lua	Android, bada, iOS, PC
Unigine unigine.com (Unigine corp.)	Comercial	C++, UnigineScript	Android, iOS, PC, PS3
Unity3D unity3d.com (Unity Technologies)	Comercial	C#, JavaScript, Boo	Android, BlackBerry 10, iOS, Windows Phone, PC, consoles, web

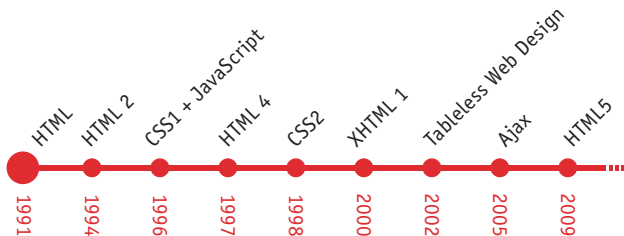


Tecnologías Web

En 2010, Mary Meeker predijo que el tráfico de Internet móvil superaría al de sobremesa en 5 años¹. Vale mucho la pena leer su reciente informe². El año 2012 presencié un crecimiento de tráfico de Internet móvil del 8% al 13%, y en algunos países en particular se ha visto una explosión de este tipo de tráfico. A mediados de 2012, la India fue uno de los primeros países donde el tráfico de Internet móvil superó el uso de Internet en dispositivos de sobremesa³.

El desarrollo continuo de tecnologías web junto con un aumento de los dispositivos compatibles con Internet promete un gran futuro para aquellos que den asistencia a una audiencia web cada vez más móvil.

La línea de tiempo aproximada de las tecnologías web es⁴:



Una gran ventaja de las tecnologías web es que ofrecen la ruta más fácil hacia el desarrollo móvil. Para un desarrollador

- 1 gigaom.com/2010/04/12/mary-meeker-mobile-internet-will-soon-overtake-fixed-internet/
- 2 www.businessinsider.com/mary-meeker-2012-internet-trends-year-end-update-2012-12
- 3 gs.statcounter.com/#mobile_vs_desktop-IN-monthly-201111-201211
- 4 slides.html5rocks.com/#timeline-slide

web, lo móvil es simplemente parte de la web. Tecnologías web como HTML, CSS y JavaScript llevan años en profundo desarrollo; sin embargo, siguen y seguirán siendo las principales bases de desarrollo de sitios móviles. Además, son sin duda más fáciles de aprender que algunos de los complejos lenguajes necesarios para el desarrollo de aplicaciones nativas. Los sitios web para móviles y las aplicaciones web hacen los contenidos accesibles en casi cualquier plataforma con un menor esfuerzo en comparación con el desarrollo nativo para varias plataformas. Esto significa, directamente, que los sitios web para móviles tienen un alcance más amplio. Por consiguiente, el desarrollo web móvil no sólo ahorra tiempo y costes de desarrollo, sino que además proporciona una alternativa ajustada en costes y tiempo también en lo que se refiere a mantenimiento. Y ser independiente de las tiendas de aplicaciones te permite ofrecer cualquier contenido que quieras rápidamente y sin necesidad de suscribir las políticas de un tienda de aplicaciones.

No obstante, existen deficiencias. Las tecnologías web luchan por igualar el nivel de integración en plataforma y el acceso directo a funciones de hardware que ofrece el desarrollo de aplicaciones nativas. Aún más, el rendimiento de las tecnologías web es altamente dependiente de la conectividad, y la monetización de sitios móviles puede resultar difícil, ya que los usuarios esperan tener acceso a las webs de forma gratuita. La herramienta más común para su monetización es la integración de publicidad. Las soluciones de pago para los sitios móviles se encuentran todavía en sus primeras etapas y tienden a ser bastante complejas de implementar. Los actuales instrumentos de monetización de las tiendas de aplicaciones ofrecen, por el contrario, una fácil instalación y un alto nivel de seguridad para el usuario final.

Si la monetización es uno de los requisitos clave, una estrategia híbrida o de aplicación web podría resultar una

buena opción de compromiso. En este caso, el desafío clave es combinar las capacidades únicas de tecnologías nativas y web para crear un producto verdaderamente fácil de usar. En el capítulo sobre el tema multiplataforma de este libro encontrarás una lista de los frameworks disponibles para crear aplicaciones híbridas.

HTML5

La quinta versión del estándar HTML promete la reproducción de características anteriormente sólo disponibles con ayuda de tecnología propietaria. HTML5 es uno de los factores clave que hacen que los desarrolladores sopesen el desarrollo de sitios móviles en lugar de aplicaciones nativas. El look-and-feel similar al de aplicaciones compiladas con un único código base para una serie de dispositivos populares, la capacidad de acceder a hardware de dispositivos tales como la cámara y el micrófono, el almacenamiento de datos en los dispositivos para utilizar sitios móviles sin conexión y la optimización de la página web según en el tamaño de pantalla, hacen de HTML5 una atractiva alternativa para el desarrollo de aplicaciones nativas. Pero HTML5 depende del navegador y es exactamente ése apoyo el que escasea actualmente. Sólo el 60% de los usuarios de Internet tienen navegadores que soporten más del 50% de las características actuales de HTML5⁵. El ex-CTO de Facebook Brent Taylor describe la situación de la siguiente manera: 'Hay una desenfrenada fragmentación de la tecnología en los navegadores móviles, por lo que los desarrolladores no saben qué parte de HTML5 pueden utilizar. HTML5 es promovido como un estándar único, pero viene en diferentes versiones para cada dispositivo móvil. Cuestiones tales como la aceleración de

⁵ gs.statcounter.com/

hardware y la gestión de derechos digitales se implementan de manera inconsistente. Eso hace que sea difícil para los desarrolladores escribir software que funcione en muchas plataformas diferentes y llegue a un público amplio’.

Mark Zuckerberg llegó incluso más allá, definiendo a la aplicación HTML5 de Facebook como ‘uno de los mayores errores, si no el mayor error estratégico’ que habían cometido⁶.

Sea como fuere, tanto Taylor como Zuckerberg creen en HTML5 a largo plazo. Facebook también ha lanzado ringmark⁷, que pone a prueba los navegadores web en 3 anillos, o niveles de soporte de características de HTML5, lo cual ayuda a los desarrolladores a comprobar rápidamente el nivel de soporte de múltiples navegadores web móviles (y desktop).

ABI Research estima que mientras que en 2010 sólo 109 millones de dispositivos ofrecían un navegador con algún tipo de soporte a HTML5, se espera que esta cifra suba a 2.1 billones de dispositivos móviles para el 2016 (unidad de billón americana, 2.100 millones europeos)⁸. Además, el Worldwide Web Consortium (W3C) ha declarado finalmente a HTML5 como completo en funcionalidad y prevé que HTML5 sea un estándar oficial web para el 2014⁹.



6 news.cnet.com/8301-1023_3-57511142-93/html5-is-dead-long-live-html5/

7 rng.io/

8 www.abiresearch.com/press/21-billion-html5-browsers-on-mobile-devices-POR-201

9 www.w3.org/

La Fragmentación Requiere Adaptación

El mayor reto del desarrollo web móvil es la fragmentación. En teoría, todos los dispositivos con conexión a Internet pueden acceder a cualquier sitio móvil a través de un navegador. La realidad, sin embargo, es que los desarrolladores necesitan atender al creciente número de navegadores y dispositivos con diferentes niveles de capacidades de software y hardware para adaptar y optimizar el contenido web móvil.

En términos generales, existen dos enfoques para optimizar el contenido para dispositivos móviles: adaptación del lado del cliente o del lado del servidor:

- La adaptación del lado cliente hace uso de una combinación de CSS y JavaScript ejecutándose en el dispositivo para ofrecer una experiencia ‘mobile-friendly’.
- La adaptación en el lado del servidor utiliza al servidor para que ejecute lógica antes de que se suministre el contenido al cliente.

La siguiente sección ofrece una revisión de técnicas tanto del lado cliente como del lado servidor para hacer los sitios webs accesibles a la mayoría de dispositivos actuales con acceso a Internet, así como a aquellos futuros.

Adaptación del Lado Cliente

Diseño Web Adaptativo (Responsive)

El diseño web adaptativo ha sido un término de moda entre vendedores y desarrolladores web por igual. En su formato más simple, consiste en una rejilla flexible, imágenes flexibles y CSS media queries para comportarse adaptativamente a una serie de resoluciones de pantalla o tipos de dispositivos. Desafortunadamente, este diseño sólo puede proporcionar una experiencia

sensible a dispositivo en un rango limitado de ellos y no puede adaptarse a contenido sofisticado. El mismo contenido se sirve a todos los dispositivos. No se recomienda como técnica para producir sitios web complejos para navegadores desktop o móviles.

Pros

- Una adaptación completa en el lado cliente asegura que no haya impacto en la infraestructura existente.
- Ajuste automático de contenido y diseño.

Contras

- El mismo contenido disponible en el sitio web estará disponible también en la versión móvil (visible o no).
- El peso del sitio tendrá un impacto significativo en términos de rendimiento en los dispositivos móviles.
- Es una aproximación general más que una optimización real a un entorno móvil.

Mejoras Progresivas

Las mejoras progresivas tienen la capacidad de responder a toda la gama de dispositivos móviles. Se envía una sola página HTML a todos los dispositivos. Se utiliza código JavaScript de manera adicional para construir progresivamente la funcionalidad a un nivel óptimo para el dispositivo en particular. Ya que es una solución exclusiva móvil, el principal inconveniente es el rendimiento. La construcción progresiva conlleva tiempo de ejecución y varía de acuerdo con el dispositivo y la red. Como solución de desktop y móvil, su principal inconveniente es que un único documento HTML se envía a todos los dispositivos. Un

entorno muy conocido que hace uso de la mejora progresiva es jQuery Mobile¹⁰.

Pros

- Una adaptación completa en el lado cliente asegura que no haya impacto en la infraestructura existente.
- Ajuste automático de contenido, funcionalidad y diseño.

Contras

- Pérdida de control, pues la detección es gestionada por el navegador.
- La detección de navegador dista mucho de ser perfecta.
- La detección realizada en el lado cliente tiene impacto sobre el rendimiento global del site.
- El mismo HTML es servido a todos los dispositivos.

Adaptación del Lado Servidor

Bases de Datos de Dispositivos

Las bases de datos de dispositivos detectan a cada dispositivo que accede a la página web y devuelven una lista de las capacidades del dispositivo al servidor. Esta información se utiliza entonces para servir un sitio móvil acorde con las capacidades del dispositivo. La adaptación del lado del servidor es una de las soluciones más antiguas y fiables. Algunas populares bases de datos de dispositivos son WURFL¹¹ y DeviceAtlas¹². El principal inconveniente de las bases de datos de dispositivo es que la mayoría sólo están disponibles bajo licencia comercial.

¹⁰ jquerymobile.com

¹¹ wurfl.sourceforge.net/

¹² deviceatlas.com/

Pros

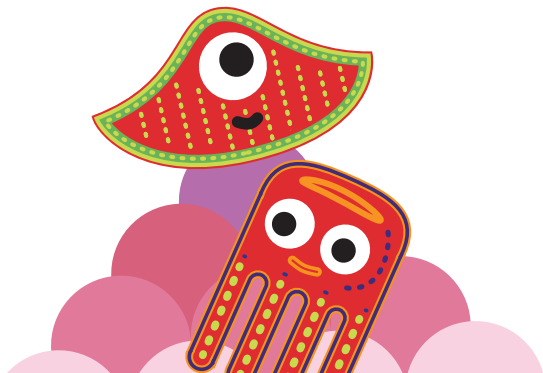
- La solución más habitual (Google, Facebook, Amazon...).
- Máximo control.
- Es posible la optimización según dispositivo (por ejemplo, iPhone, Samsung Galaxy III, ...).

Contras

- Los Repositorios de Descripción de Dispositivos (en inglés Device Description Repositories) están orientados a hardware.
- Aparte de los datos, es necesario un sistema de detección (un simple 'User-Agent' no es suficiente).

Adaptación Híbrida

Realmente lo mejor de ambos mundos, la combinación de la adaptación en cliente y servidor asegura un alto rendimiento gracias a la adaptación del lado del servidor y garantiza que las capacidades obtenidas se pueden utilizar para enriquecer la experiencia móvil para las visitas siguientes. Las soluciones híbridas de adaptación están disponibles comercialmente de



compañías como Sevenval¹³ o Netbiscuits¹⁴, o como soluciones en la nube respaldadas por la comunidad, por ejemplo FITML¹⁵.

Mejor Input de Datos

Con pequeñas teclados, a menudo en pantalla, introducir texto puede resultar engorroso y lento, especialmente si el usuario tiene que introducir números, direcciones de correo electrónico, etcétera. Afortunadamente, los desarrolladores pueden especificar el tipo esperado de datos y los smartphones mostrarán entonces el teclado más apropiado en pantalla. mobileinput-types.com proporciona varios ejemplos claros y concisos.

Testear Tecnologías Web

Se puede probar cómo las tecnologías web funcionan en varios teléfonos móviles de múltiples maneras. La más simple es probar el sitio o aplicación web en una variedad de navegadores en dispositivos móviles. Estos deberían incluir una combinación de los navegadores móviles más populares, por ejemplo aquellos basados en datos públicos gs.statcounter.com/#mobile_browser-ww-monthly-201111-201211. El conjunto de dispositivos se puede refinar mediante el análisis de datos de los registros web existentes, etcétera. Además, las pruebas en varios factores de forma ayuda a descubrir problemas de diseño y formato.

¹³ www.sevenval.com

¹⁴ www.netbiscuits.com

¹⁵ www.fitml.com

En cuanto a las pruebas automatizadas, WebDriver¹⁶ es el entorno predominante. Existen dos enfoques complementarios:

1. Testeo automático utilizando controles WebView en Android e iOS.
2. Falseo del User-Agent (spoofing) utilizando Google Chrome o Mozilla Firefox configurados para emular a varios navegadores web móvil.

Ambas aproximaciones tienen pros y contras:

- WebViews incrustadas se ejecutan en la plataforma de destino. Es probable encontrar en ellas muchos errores de comportamiento. Sin embargo, la configuración tiene mucho que ver y algunas plataformas no están soportadas.
- El falseo puede engañar a los servidores web para que traten al navegador como si accediera mediante una gran variedad de navegadores, incluyendo navegadores móviles no disponibles con la WebView incrustada, por ejemplo el teléfono Nokia Asha 201. Aún así, el comportamiento y renderizado no son realistas, así que muchos errores permanecerán latentes, mientras que habrá falsos positivos que no se darán en los dispositivos reales.

¹⁶ seleniumhq.org/projects/webdriver/

Aprende Más

Online

- **HTML5 Rocks** (excelente recurso sobre HTML5 que incluye tutoriales, presentaciones, artículos, etc.):
www.html5rocks.com/en/
- **Breaking the Mobile Web** (Max Firtman, el autor de múltiples libros acerca de programación web móvil, ofrece noticias actualizadas en su blog dedicado a móviles):
www.mobilexweb.com
- **Mobi Thinking** (el recurso de DotMobi para vendedores, con análisis y opiniones de expertos en marketing móvil):
www.mobithinking.com
- **Testing (Mobile) Web Apps:** docs.webplatform.org/wiki/tutorials/Testing_web_apps
- **WHATWG** (la página de la comunidad HTML):
www.whatwg.org/
- **Word Wide Web Consortium** (la organización que define los estándares web): *www.w3.org/*

Libros

- **Mobile First** por Luke Wroblewski
- **Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement** por Aaron Gustafson y Jeffrey Zeldman
- **Responsive Web Design** por Ethan Marcotte
- **Programming the Mobile Web** por Max Firtman
- **jQuery Mobile Up and Running** por Max Firtman



Accesibilidad

Independientemente de la tecnología que elijas para desarrollar tus aplicaciones, querrás asegurarte de que tu aplicación puede ser utilizada por tanta gente y en tantos mercados como sea posible.

Muchos de tus usuarios potenciales pueden tener una discapacidad que hace que sea más difícil para ellos utilizar la tecnología móvil. Estas discapacidades incluyen, pero no se limitan a, diversos niveles de deterioro visual o auditivo, discapacidades cognitivas, problemas de destreza, tecnofobia y similares. Muchos de estos usuarios confían en aplicaciones de terceros, tales como TalkBack en la plataforma Android o Talks de Nuance para la plataforma Symbian, que ofrece lectura de pantalla y funciones de lupa. iOS incluye ahora VoiceOver¹, que es la favorita en términos de proporcionar una interfaz accesible en teléfonos móviles.

Para hacer que tu software sea accesible para los usuarios con discapacidad, debes seguir algunas pautas generales. Si te atienes a ellas, también le darás a tu aplicación mayores posibilidades de interoperar con cualquier software de acceso de terceros que el usuario pueda estar ejecutando en conjunción con tu software:

- Descubre las capacidades y APIs de accesibilidad que tu plataforma ofrece y sigue las prácticas recomendadas para aprovechar dichas APIs, si existieran.
- Usa elementos de interfaz estándares en vez de los personalizados siempre que sea posible. De esta manera te aseguras de que si la plataforma tiene una infraestructura de accesibilidad, o la adquiere en el futuro, tu aplicación

¹ www.apple.com/accessibility/iphone/vision.html

tendrá muchas probabilidades de ser renderizada de manera accesible para tus usuarios.

- Sigue las directrices para interfaces estándares de la plataforma. Esto refuerza la consistencia y puede implicar un diseño más accesible.
- Etiqueta todas las imágenes con una descripción corta de su contenido, por ejemplo “Play” para un botón de reproducción.
- Evita utilizar colores como único método de diferenciar una acción. Por ejemplo, un usuario ciego al color no será capaz de identificar errores si se le solicita corregir los campos de un formulario que están marcados en rojo.
- Asegúrate de tener un buen contraste de tonos de color en toda la aplicación.
- Usa la API de accesibilidad en tu plataforma, si la hay. Esto te permitirá hacer interfaces de usuario personalizadas más accesibles y significará menos carga de trabajo para tí.
- Da soporte a la navegación programática de tu interfaz de usuario. Esto no sólo permitirá que tus aplicaciones sean utilizadas con un teclado externo, sino que también mejorará la accesibilidad de tu aplicación en plataformas tales como Android, donde la navegación puede ser realizada por un trackball o un d-pad virtual.
- Prueba tu aplicación en el dispositivo de destino con tecnología de asistencia, como por ejemplo VoiceOver en el iPhone.

Puedes encontrar una lista de directrices más completa online².

² www.slideshare.net/berryaccess/designing-accessible-usable-application-user-interfaces-for-mobile-phones

Recientemente, Apple y Google han aumentado la importancia de su apoyo a este área mediante una interfaz de accesibilidad para mejorar sus entornos de automatización de pruebas de interfaz gráfica. Esto proporciona un incentivo para que los desarrolladores consideren el diseño de sus aplicaciones de manera más accesible, lo que es muy positivo.

Mirando las diferentes plataformas móviles con más detenimiento, se hace evidente que difieren ampliamente en cuanto a sus características y APIs de accesibilidad.

Aplicaciones Android Accesibles

La última distribución de Android, la versión 4, trae una serie de mejoras de accesibilidad. Estas incluyen el foco de accesibilidad, soporte Braille y otros. La documentación de desarrollo también se ha mejorado. Sin embargo, para maximizar el alcance de tu aplicación hacia aquellos que utilizan versiones anteriores de Android, debes utilizar los controles estándar de interfaz de usuario siempre que sea posible y asegurarte de que los usuarios pueden navegar por tu aplicación a través de un trackball o d-pad. Esto le dará a tu aplicación la mejor oportunidad de ser renderizada de manera accesible por tecnologías tales como Talkback, u otras aplicaciones de asistencia.

Para obtener detalles sobre cómo utilizar la API de accesibilidad de Android, y los datos de las mejores prácticas en materia



de accesibilidad de la plataforma, consulta el documento de Google titulado Creando Aplicaciones Accesibles³.

También encontrarás más ejemplos en el área de aprendizaje de la documentación para desarrolladores, en una sección titulada Implementado la Accesibilidad⁴. Testear la accesibilidad también está contemplado online en⁵.

Para obtener más información acerca de accesibilidad Android, incluyendo cómo utilizar la API de texto a voz, consulta el proyecto Eyes-Free⁶.

Aplicaciones BlackBerry Accesibles

BlackBerry también proporciona buena y detallada información sobre el uso de su API de accesibilidad, y muchos consejos sobre el diseño de interfaces de usuario accesibles en su sitio web para desarrolladores⁷.

En mayo de 2012, BlackBerry lanzó BlackBerry Screen Reader⁸ para varios BlackBerry® Curve™ recientes. Está disponible en descarga gratuita y puede que te interese utilizarla para las pruebas de accesibilidad de tus aplicaciones.

3 developer.android.com/guide/topics/ui/accessibility/apps.html

4 developer.android.com/training/accessibility/index.html

5 developer.android.com/tools/testing/testing_accessibility.html

6 code.google.com/p/eyes-free

7 developer.blackberry.com/java/documentation/intro_accessibility_1984611_11.html

8 www.blackberry.com/screenreader

Aplicaciones iOS Accesibles

iOS tiene un buen soporte en accesibilidad. Por ejemplo, los dispositivos iOS incluyen:

- **VoiceOver** es un lector de pantalla que menciona los objetos y el texto en pantalla, permitiendo a tu aplicación ser utilizada por aquellos que no pueden ver la pantalla con claridad.
- **Zoom** amplía los contenidos en pantalla.
- **Blanco sobre Negro** invierte los colores, lo que ayuda a muchas personas que necesitan el contraste del blanco y negro pero que encuentran el fondo blanco demasiado brillante.
- **Subtitulado** para personas con problemas auditivos.
- **Alertas auditivas, visuales y de vibración** permiten a los usuarios elegir las que mejor se adaptan a sus necesidades.
- **Control por voz y Siri** permite a los usuarios realizar llamadas de teléfono y operar varias otras capacidades de su dispositivo utilizando comandos de voz.

Si estás trabajando en iOS, asegúrate de seguir las directrices de Apple respecto a accesibilidad⁹. Estas directrices detallan la API y proporcionan una excelente fuente de consejos y sugerencias para maximizar la experiencia del usuario con tus aplicaciones.

⁹ developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/iPhoneAccessibility

Aplicaciones Symbian/Qt Accesibles

En el momento de escribir esta guía, no hay una "API de accesibilidad" para la plataforma Symbian; sin embargo, hay varias aplicaciones de terceros que proporcionan un buen acceso a muchos teléfonos Symbian y a muchas de las aplicaciones que utilizan.

En el desarrollo de aplicaciones nativas para Symbian, tu mejor oportunidad de desarrollar una aplicación accesible es utilizar los controles de interfaz de usuario estándar siempre que sea posible. Si vas a desarrollar con Qt, visita la web para conocer más detalles sobre su API de accesibilidad¹⁰.

Aplicaciones Windows Phone y Windows 8 Accesibles

Al igual que con el resto de plataformas móviles más importantes, las características de accesibilidad de Windows Phone se están mejorando en cada nueva versión.

Como probablemente ya aprendiste en los capítulos sobre Windows Phone y Windows 8, tienes esencialmente dos opciones al respecto para crear aplicaciones para esa plataforma.

Si tu aplicación está escrita en C#, C++ o Visual Basic, encontrarás información completa sobre cómo hacer tu aplicación accesible en el documento Accessibility in Metro style apps using C++, C#, or Visual Basic¹¹.

Si has optado por utilizar HTML5 y JavaScript, entonces tendrás que consultar el documento Accessibility in Metro style apps using JavaScript¹².

¹⁰ doc.qt.nokia.com/qq/qq24-accessibility.html

¹¹ msdn.microsoft.com/en-us/library/windows/apps/xaml/hh452680.aspx

¹² msdn.microsoft.com/en-us/library/windows/apps/hh452702.aspx

Una vez que hayas probado la accesibilidad de tu aplicación¹³, Microsoft únicamente te permitirá declarar tu aplicación como accesible en la tienda Windows¹⁴, que le permitirá ser descubierta por los que están filtrado por accesibilidad en sus búsquedas.

Aplicaciones Web Móvil Accesibles

Mucho se ha escrito sobre el tema de la accesibilidad web; sin embargo, en el momento de escribir esta guía, no existe una norma que incorpore las mejores prácticas de desarrollo para una web móvil accesible.

Si tu aplicación está destinada a simular el look-and-feel de una aplicación nativa, entonces debes seguir las pautas ya mencionadas en este capítulo.

Si eres un desarrollador de contenidos web, entonces debes echar un vistazo a la Web Content Accessibility Guidelines (WCAG) Overview¹⁵.

Como el soporte a HTML5 se está incrementando en las plataformas móviles, puede que te resulte útil echar un vistazo al documento titulado Mobile Web Application Best Practices¹⁶, ya que es probable que forme parte de cualquier estándar de accesibilidad de aplicaciones web móviles que emerja en el futuro.

También es un recurso útil el documento Relationship between Mobile Web Best Practices (MWBP) and Web Content Accessibility Guidelines (WCAG)¹⁷.

¹³ msdn.microsoft.com/en-us/library/windows/apps/xaml/hh994937.aspx

¹⁴ msdn.microsoft.com/en-us/library/windows/apps/xaml/jj161016.aspx

¹⁵ w3.org/WAI/intro/wcag

¹⁶ w3.org/TR/mwabp

¹⁷ www.w3.org/TR/mwbp-wcag/



Aplicaciones Corporativas: Estrategia Y Desarrollo

Los responsables de decisiones corporativas ven actualmente las aplicaciones empresariales móviles como un factor estratégico, una necesidad, más que como una mera entrada en una hoja de cálculo de contabilidad. Las aplicaciones empresariales internas son capaces de reducir la latencia de la transferencia de información dentro de una organización, incrementando la agilidad del trabajador al hacer disponibles datos de carácter competitivo en cualquier momento y en cualquier lugar. Las aplicaciones también pueden permitir a las empresas comprometerse con sus clientes, proveedores y consumidores finales, etc. Ejemplos de aplicaciones empresariales incluyen software para puntos y personal de ventas, de respuesta a emergencias, gestión de inventario o cadena de suministro, pero también de marketing B2C.

Puede parecer una obviedad decirlo, pero el riesgo más importante en este momento es **no** tener una estrategia móvil corporativa. Las empresas están tomando el enfoque **Móvil para Todo** en contraste a limitar su uso a la alta dirección, como ocurría en el pasado. Para conseguir esto, el enfoque tradicional de TIC de compra y distribución de dispositivos en la estructura de gestión ya no es la única estrategia a utilizar. El concepto Bring Your Own Device (BYOD, en castellano Trae Tu Propio Dispositivo) se está afianzando, permitiendo al personal utilizar sus dispositivos personales para conectarse a la infraestructura TIC, descargar contenido seguro y usar aplicaciones empresariales. Con la llegada de BYOD, una empresa se expone a riesgos que tradicionalmente no se consideraban en la estrategia TIC corporativa. La adopción temprana en la empresa

de una estrategia móvil bien pensada y ejecutada es clave para asegurarse de que los datos están protegidos en todo momento.

Puntos clave para las Aplicaciones Móviles en la definición de nuevas iniciativas empresariales:

- Reducción de costes comparados con los sistemas existentes
- Racionalización de los procesos de negocio
- Ventaja competitiva con el acceso directo a datos actualizados de manera inmediata
- Incremento de la satisfacción y efectividad del empleado
- Rápida respuesta en comparación con procesos existentes

Estrategia

Hoy en día, muchas empresas tienen un Chief Mobile Officer (CMOO) o un puesto equivalente. El trabajo del CMOO es coordinar las tendencias móviles y sus directrices, y hacer de puente entre el negocio y TIC. Dependiendo del tamaño y el enfoque de la empresa, su trabajo es también construir un equipo interno de desarrollo de software móvil o coordinar la cooperación con una agencia de desarrollo externa. Para asegurarse de que el software móvil ofrece lo que los empleados/usuarios quieren, que sea técnicamente factible y que todo se ajuste a la estrategia global de la empresa, el CMOO podría considerar la creación de un Consejo de Innovación Móvil (MIC, en inglés Mobile Innovation Council). El MIC debe incluir a miembros clave tales como: representantes capacitados del equipo de desarrollo de aplicaciones móviles, las partes interesadas en tema móvil en la empresa y, lo más importante, los usuarios finales de varios departamentos con experiencia en los procesos de negocio más relevantes.

Los tópicos en los que el CMOO debe centrarse, en sincronía con el MIC incluyen:

- **Estrategia:** Visión y dirección de la estrategia móvil global y las aplicaciones.
- **Políticas de gobernanza:** Bring Your Own Device (BYOD) versus Chose Your Own Device (CYOD), que básicamente es la diferencia entre una política de Gestión de Aplicaciones Móviles (MAM, del inglés Mobile Application Management) y una de Gestión de Dispositivos y Seguridad Móviles (CYOD, del inglés Mobile Device Management & Security).
- **Especificaciones de las aplicaciones.**
- **Libro de ruta de las aplicaciones.**
- **Planificación presupuestaria.**
- **Visto bueno:** aprobación de las aplicaciones para producción.
- **Despliegue de aplicaciones:** feedback temprano de demos y prototipos, testeo y despliegue masivo.
- **Incentivos:** Promocionar la adopción de entornos móviles.

En cuanto a la adopción comercial, el desarrollo de aplicaciones corporativas se encuentra todavía en sus inicios, y como tal uno de los principales obstáculos que debe enfrentar una empresa escribiendo aplicaciones corporativas para terceros, o un manager de proyecto deseoso de adoptar una estrategia interna corporativa, es resolver una necesidad de negocio. La pregunta más probable es "Todo esto suena muy bien, pero ¿por qué lo necesitamos?", por lo que tienes que estar preparado para dar razones de peso en base a las que una empresa debería adoptar una estrategia móvil.

Puntos clave en el modelado de un caso de negocio para Aplicaciones Corporativas Móviles:

- Crear un Plan de Visión para nuevas aplicaciones móviles, y cómo ayudarán y darán forma a tu empresa.
- Crear un Manifiesto de Definición de Aplicación (ADS, del inglés Application Definition Statement) para cada aplicación, especificando propósito y audiencia objetivo.
- Crear un Presupuesto para dispositivos.
- Crear un plan para una Infraestructura de Aplicaciones, Gestión de Dispositivos y Seguridad.
- Crear un plan para en equipo de desarrollo utilizando una plataforma de desarrollo fiable a futuro, con una Plataforma de Desarrollo de Aplicaciones Corporativas (MEAP, del inglés Mobile Enterprise Application Platform).

Gestión de Dispositivos y Aplicaciones en la Empresa

Al desarrollar una aplicación empresarial, siempre hay que tener en cuenta que el hardware que contiene datos confidenciales de la empresa se puede extraviar o robar. En la actualidad, hay dos enfoques para asegurar dispositivos, contenidos y aplicaciones: Gestión de Dispositivos Móviles (MDM, del inglés Mobile Device Management) y Gestión de Aplicaciones Móviles (MAM, del inglés Mobile Application Management).

MDM ofrece un control total corporativo sobre un dispositivo, por lo que cuando un dispositivo se pierde, se roba o un empleado se marcha, llevándose el dispositivo, la empresa puede limpiar el dispositivo y, esencialmente, éste deja de funcionar. Este enfoque general se toma cuando la empresa es propietaria del dispositivo y por tanto todos los datos y aplicaciones en el dispositivo son propiedad suya: todos los

datos personales almacenados en el dispositivo se almacenan en él a riesgo del empleado.

MAM permite a una empresa adoptar BYOD, ya que así la empresa puede asegurar aplicaciones y contenidos descargados a un dispositivo sin quitarle el control del mismo al propietario. Cuando un empleado deja la empresa, llevándose consigo su dispositivo, la organización puede deshabilitar las aplicaciones empresariales y limpiar cualquier contenido descargado en el dispositivo sin afectar a los datos personales, como fotografías y aplicaciones de consumo compradas por el empleado. La mayoría de soluciones de MDM y MAM son multiplataforma, siendo compatibles con dispositivos Apple, Android, Windows y BlackBerry, y esto siempre debe tenerse en cuenta a la hora de decidirse por un proveedor de MDM o MAM.

Hay varias funciones de seguridad disponibles a través de estas dos soluciones de gestión, incluyendo:

- Monitorización de dispositivo
- Control de licencias
- Distribución vía solución interna Over-The-Air (OTA)
- Inventariado de software
- Control de recursos
- Control remoto
- Gestión de la conexión
- Soporte y distribución de aplicaciones

Las medidas de seguridad incluyen:

- Protección de contraseñas
- Encriptación de datos en el dispositivo
- Encriptación de datos OTA
- Bloqueo remoto de dispositivos
- Limpieza de datos remota

- Reprovisionamiento de dispositivos
- Copia de seguridad de datos en los dispositivos

Aunque las soluciones MDM y MAM ofrecen la gestión y el envío de aplicaciones a los dispositivos, proporcionando una gestión multiplataforma, también existen métodos específicos del sistema operativo para algunos aspectos de la gestión de aplicaciones, así como su envío y despliegue.

- Android ofrece la Google Play Store para Empresas, que permite a una compañía distribuir y gestionar aplicaciones Android a través de una Google Play Store personalizada.
- Apple cuenta con el Acuerdo de Compras por Volumen, que permite a las empresas realizar compras en bloque de aplicaciones públicas de la App Store para ser distribuidas entre sus empleados de manera gratuita vía descarga, hasta agotar el número de compras realizadas.
- Microsoft ofrece el Microsoft System Center 2012 (anteriormente System Center Mobile Device Manager), con MDM para dispositivos Windows.
- RIM BlackBerry cuenta con el Business Enterprise Server (BES), permitiendo un detallado MDM para dispositivos BlackBerry.

Plataformas de Aplicaciones Móviles Corporativas (MEAPs)

Por lo general, un elemento clave de las aplicaciones empresariales es la sincronización de datos. Los dispositivos móviles tienen que recibir información actualizada o relevante de los servidores corporativos y los datos actualizados o recogidos tienen que ser enviado de vuelta. El enfoque de acceso a los datos está determinado por las responsabilidades del usuario, así como por la política de confidencialidad. En cualquier caso, la sincronización tiene que ser segura, ya que los datos de las empresas son uno de sus activos más preciados. Aún más, una aplicación integrada en toda la empresa debe ser multiplataforma. Para compensar las deficiencias de los SDK nativos así como de las soluciones multiplataforma más comunes en este aspecto, es posible que desees considerar la evaluación de soluciones tipo Plataformas de Aplicaciones Móviles Corporativas (MEAP, del inglés Mobile Enterprise Application Platform). Las MEAPs son entornos móviles que proporcionan el middleware y las herramientas para desarrollar, probar, implementar y administrar aplicaciones empresariales que se ejecutan en múltiples plataformas móviles con distintas fuentes de datos en el back-end. Su objetivo es simplificar el desarrollo y reducir sus costes, en entornos donde las habilidades técnicas se deben mantener para múltiples plataformas, herramientas y complejidades, como la sincronización de datos y autenticación. Algunas soluciones disponibles son:

- Amp Chroma por Antenna¹
- Kony²

¹ <http://www.antennasoftware.com>

² <http://www.kony.com>

- SpringWireless³
- Sybase Unwired Platform⁴
- Syclo, adquirida recientemente por SAP⁵

Seguridad En Aplicaciones Corporativas

Una de las funciones principales de cualquier departamento de TIC es asegurarse de que todos los aspectos de la infraestructura de la empresa están asegurados contra ataques, de modo que no haya fugas de datos y éstos no se vean comprometidos o robados. Dado que los dispositivos móviles son una extensión de la infraestructura TIC de una empresa, todas las aplicaciones empresariales deben ser diseñados para asegurarse de que no se pueden utilizar para obtener acceso ilegal a la red interna corporativa. Como desarrollador de aplicaciones empresariales, por lo general se te pedirá que te ajustes a las normas que la empresa haya establecido en sus políticas de seguridad, así que debes estar preparado para responder a preguntas acerca de la seguridad de tu aplicación, como el cifrado de datos, la comunicación en red y en cómo lidias con dispositivos jailbreakeados o rooteados.

Puntos clave para asegurar aplicaciones empresariales:

- Cuando almacenes un dato en el dispositivo, asegúrate de que está encriptado.
- Cuando comuniques con un servicio web, usa siempre https.
- Además de lo anterior, asegúrate de que realizas verificaciones de puntos finales tanto en la aplicación como en el

³ <http://www.springwireless.com>

⁴ <http://www.sybase.com/products/mobileenterprise/sybaseunwiredplatform>

⁵ <http://www.syclo.com>

servicio web, para confirmar que tanto el servidor como el dispositivo son válidos.

- Siempre verifica que los ajustes del dispositivo cuentan con una checksum para asegurarte de que los valores no pueden ser cambiados una vez instalada la aplicación en el dispositivo.
- No permitas que la aplicación se ejecute en dispositivos jailbreakados o rooteados.
- Ten un método para deshabilitar la aplicación si ésta detecta que ha sido comprometida.
- Asegúrate de que el uso de encriptación es conforme a normativas de exportación y legislación relevantes para la región donde la aplicación es utilizada.





Analíticas Móviles

Nuestras aplicaciones son utilizadas por personas a las que probablemente nunca conoceremos, así que las analíticas para móviles te pueden ayudar a descubrir cómo tu aplicación se está utilizando para poder mejorar las futuras versiones de la aplicación. Más de la mitad de las mejores aplicaciones móviles ya incluyen analíticas¹.

Hay muchos productos comerciales que dicen que ofrecen potentes y fáciles soluciones para añadir analíticas a nuestras aplicaciones, incluso hay algunos de código abierto. Podríamos simplemente elegir una de estas soluciones y esperar que nos funcione; sin embargo, corremos el riesgo de ahogarnos en un turbulento mar de datos cuyo significado desconecemos y del que no podremos escapar fácilmente. En este capítulo se incluyen algunos consejos y orientación para ayudarte a entender cómo las analíticas móviles pueden ayudarte a comprender cómo está siendo utilizada tu aplicación. Descubrirás cómo elegir una solución adecuada e implementarla.

Poniéndote En Marcha

Muchos de los proveedores de soluciones de analíticas para móviles incluyen un apartado 'Getting Started' donde se aprende cómo comenzar a utilizar sus productos. Algunos ejemplos son Flurry² y KISSmetrics³.

En general, es necesario registrarse antes de que puedas utilizar cualquiera de los productos de manera útil, ya que

¹ blog.velti.com/mobclix-index-the-when-where-what-of-apps, static.usenix.org/event/sec11/tech/slides/enck.pdf

² support.flurry.com

³ support.kissmetrics.com/getting-started/overview

necesitan ser configurados con una única 'clave' para tu aplicación.

Considera algunas de las posibles soluciones antes de comprometerte con ninguna de ellas. Lee la documentación y el código de ejemplo para ver lo fácil que es ponerla en práctica en tu aplicación, y comprueba los acuerdos legales, incluyendo la privacidad. A continuación, elije al menos una de ellas para poder experimentar con esas analíticas en tu aplicación. Probablemente, integrando su código aprenderás mucho más acerca de lo que te gustaría lograr mediante el uso de analíticas móviles en tu aplicación, y de cómo funcionan en la práctica.

Debes ser consciente de que la mayoría de proveedores de soluciones de analítica móvil también están extrayendo y utilizando los datos comunicados por tu aplicación, y que pueden suministrarlos y venderlos a otros. Puede que controlen la vida de esos datos, lo que significa que podrían hacerlos inaccesibles a ti o, por el contrario, podrían conservarlos y usarlos mucho tiempo después de que hayas retirado de circulación tu aplicación. Además, si hay información de identificación personal en los datos también pueden darse otras implicaciones legales. Así que vale la pena considerar cómo terceros utilizarán y compartirán los datos comunicados a través de su software y APIs.



Decidir Qué Medir

¿Qué te gustaría a medir, comprender, acerca de cómo se está utilizando la aplicación? Algunas sugerencias:

- **Eventos clave de uso:** ¿Cuáles son las características clave más utilizadas de tu aplicación?
- **Eventos centrados en el negocio:** Cualquier interacción generada por el usuario que suponga un beneficio para tí. ¿Con qué frecuencia tus usuarios compran la versión premium de tu aplicación y otros ítems ofrecidos en su interior? ¿Cuándo cancelan pedidos o descartan un carrito de compra antes de realizar el pago?
- **Métricas de usabilidad:** ¿Cuándo se quedan bloqueados tus usuarios en el flujo de uso? ¿Alcanzan rápidamente su objetivo cuando utilizan tu software?

Una vez definidas las principales áreas de interés tendrás que diseñar las métricas de análisis, por ejemplo, cuáles elementos de los datos deberían ser reportados.

Definir Cómo Medir

Crea nombres significativos para tus eventos de interacción, en lugar de limitarte a numerarlos, para que puedas recordar fácil y correctamente lo que miden. Para cada evento que desees grabar, debes decidir cuáles son los elementos que necesitan ser incluidos. Considera cómo serán utilizados los datos una vez que se hayan obtenido; por ejemplo, esboza informes y diagramas estándar y mapea cómo los diversos datos serán procesados para generar cada informe o gráfico.

Asimismo, recuerda abordar temas sobre globalización, tales como la fecha y hora de cada elemento. ¿La aplicación detectará el momento de un evento de acuerdo a la ubicación

del dispositivo, su configuración o utilizando un tiempo global como el UTC⁴?

Algunas de las soluciones de analítica móvil grabarán y reportarán automáticamente los elementos de datos al servidor. Vale la pena comprobar cuáles son estos elementos, cómo y cuándo se presentan, y cómo están formateados. Entonces podrás decidir si deseas utilizar y confiar en estos elementos reportados automáticamente.

Las etiquetas personalizadas de eventos enriquecen aquellos predeterminados, y muchas de las soluciones de analítica móvil proporcionan medios para que tu aplicación los genere. Es posible que necesites formatear los mensajes de eventos personalizados. Si es así, pon atención a la codificación de los elementos y los separadores, es posible, por ejemplo, que deban pasar por codificación URL⁵ cuando se envían mensajes como REST⁶.

También puede que desees considerar la frecuencia con que la aplicación debe informar sobre eventos para minimizar el riesgo de saturar la capacidad disponible del sistema de analíticas, lo que podría afectar a la fiabilidad y exactitud de los datos suministrados. Un método para reducir el volumen

4 http://en.wikipedia.org/wiki/Coordinated_Universal_Time en.wikipedia.org/wiki/Coordinated_Universal_Time]

5 en.wikipedia.org/wiki/Percent-encoding

6 msdn.microsoft.com/en-us/library/live/hh243648



de datos procesados es el *sampling*. Adán Cassar ha publicado un interesante post sobre este tema en www.periscopix.co.uk/blog/should-you-be-worried-about-sampling.

Adaptando Tu Código

Puede que tengas que declarar capacidades adicionales necesarias para que los análisis móviles funcionen correctamente cuando se integran con tu aplicación.

Para Android, a éstos se les conoce como permisos. Probablemente las analíticas necesitarán permisos de Internet de modo que los eventos puedan ser enviados online, y permisos centrados en la localización en el caso de que la solución registre la ubicación del teléfono. Si tu aplicación ya está utilizando esos permisos, no es necesario especificar su uso de nuevo.

Para iOS, `UIRequiredDeviceCapabilities` indica a iTunes y a la App Store qué características de dispositivo requiere la aplicación. Se implementa como un diccionario donde los elementos se especifican utilizando claves, que incluyen WiFi, servicios de localización y GPS.

En Windows Phone, se utilizan capacidades para decidir lo que la aplicación utiliza. Localytics tiene online una guía de arranque rápido⁷ que incluye un ejemplo de configuración de la capacidad `ID_CAP_IDENTITY_DEVICE`.

⁷ <http://www.localytics.com/docs/windows-phone-7-integration/> www.localytics.com/docs/windows-phone-7-integration/

Gestionando Los Resultados

Ten en cuenta que hay un lapso de tiempo entre que una aplicación envía un evento de analíticas y la información es procesada y puesta a tu disposición. El retardo, o latencia, varía entre muy cerca de 'tiempo real' y un día o más. Tú, y tus promotores, debéis decidir cuánto tiempo podéis permitir os retrasar el conocimiento sobre acontecimientos en tiempo real.

Algunas soluciones de análisis también proporcionan una API para acceder a los datos. Esto te permite realizar copias de seguridad y crear informes personalizados.

Para evaluar la calidad de los resultados, algunas empresas invierten esfuerzo adicional en incorporar múltiples soluciones de análisis en su aplicación y cruzar después los resultados. Sin embargo, dos resultados contradictorios no se reconcilian fácilmente, así que puede ser necesario el uso de tres bloques de resultados para diagnosticar las diferencias por triangulación⁸. Si has decidido trabajar con KISSmetrics, echa un vistazo a su artículo sobre maneras de evaluar tus métricas en support. kissmetrics.com/getting-started/testing-km.

Privacidad

Recuerda que debes explicar a los usuarios finales que la aplicación está diseñada para registrar y compartir información acerca de cómo la aplicación se está utilizando, a ser posible en los términos y condiciones que desees. Puede que necesites, o te interese, que los usuarios tengan capacidad de decidir si permiten un seguimiento a su uso de la aplicación. Si es así, facilítale al usuario el control sobre los ajustes; además,

⁸ [en.wikipedia.org/wiki/Triangulation_\(social_science\)](https://en.wikipedia.org/wiki/Triangulation_(social_science))

considera la posibilidad de proveer al usuario de una forma de acceder a los datos registrados, borrarlos, o ponerse en contacto con el proveedor de soluciones de analíticas.

Los proveedores de librerías de terceros parecen tener variadas aproximaciones a la privacidad. Algunos declaran que la privacidad de los usuarios es de suma importancia y resaltan la relevancia de no realizarles seguimiento. Google Analytics prohíbe de manera fehaciente trazar información personal que permita la identificación en sus términos de servicio⁹. Otros proveen de ejemplos, incluidos fragmentos de código, que demuestran cómo grabar datos de identificación personal. Por ejemplo, KISSmetrics provee el siguiente código `sharedAPI] identify:@"name@email.com"];`¹⁰, y Mixpanel ofrece un ejemplo de cómo actualizar el registro de analíticas personales de un usuario¹¹.

Hay varios lugares para aprender más sobre privacidad y la ética de trabajar con datos relativos a los usuarios, por ejemplo:

- El post en el blog de Jeff Northrop sobre analíticas móviles¹²
- El libro de Kord Davis "Ethics of Big Data"

⁹ www.google.com/analytics/terms/us.html

¹⁰ support.kissmetrics.com/apis/objective-c

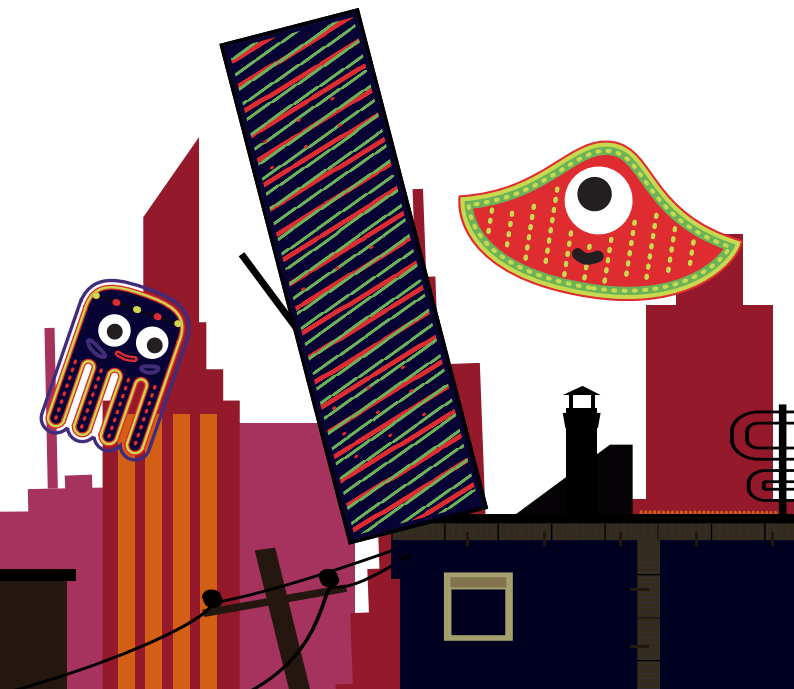
¹¹ mixpanel.com/docs/people-analytics/android

¹² <http://jnorthrop.me/2012/07/2/privacy-considerations-mixpanel-people-analytics/>

Aprende Más

Esperamos que este capítulo haya despertado tu curiosidad por saber más acerca de las analíticas móviles. A continuación, algunos lugares para comenzar tu investigación:

- **The Mobile Developer's Guide to the Parallel Universe**, un libro gemelo a éste, que trata analíticas móviles desde la perspectiva del marketing. Disponible en descarga PDF desde www.wipconnector.com.
- **Mobile Analytics** por Jesús Mena, disponible como libro Kindle vía Amazon.



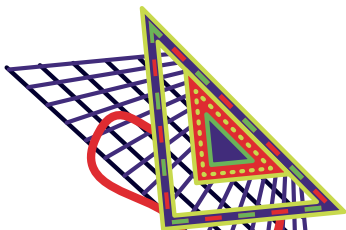


Implementando Rich Media

Decir “hay tantos estándares como teléfonos” es un auténtico tópico cuando se habla de la lista de formatos multimedia soportados en teléfonos móviles. A diferencia de los ordenadores personales, donde la mayoría de formatos de audio y vídeo son compatibles o se puede instalar fácilmente un códec para soportar alguno, los dispositivos móviles son una historia diferente. Los formatos y protocolos específicos para estas plataformas se han desarrollado durante los últimos años para permitir la optimización respecto a tamaño de pantalla y ancho de banda. Pequeñas variaciones en la resolución, tasa de bits, contenedor, protocolo o códec puede causar fácilmente un fallo de reproducción, por lo que se recomienda probar en dispositivos reales.

Dicho esto, la mayoría de los smartphones actuales soportan MP4 h.264 320x240 AAC-LC, aunque múltiples variaciones son posibles de un dispositivo a otro, incluso entre fabricantes o versiones de firmware. Nuevos formatos se añaden a diario aún hoy en día, tales como WebM/vp8¹, un estándar abierto de vídeo sobre Android 4+, que aspira a convertirse en el estándar para HTML5 (aunque no está soportado por Apple aún). A continuación están los formatos de pantalla completa recomendados para una óptima compatibilidad:

¹ <http://en.wikipedia.org/wiki/VP8> en.wikipedia.org/wiki/VP8]



Contenedor	mp4, 3gp, avi (sólo BlackBerry), wmv (sólo Windows Phone + BB10)
Protocolo	HTTP (progresivo o descarga) or RTSP (streaming)
Vídeo	H.264, H.263
Audio	AAC-LC, MP3, AAC+
Resoluciones Clásicas	176x144 (teléfonos antiguos), 320x240, 480x320 (J2ME)
Resoluciones Comunes	480x800, 640x480 (Blackberry), 960x640 (iPhone), 1024x768 (iPad 1+2), 2048x1536 (iPad 3+4)
Resoluciones HD	1280x720 (BB10, Samsung, Windows Phone 8), 1136x640 (iPhone 5)

Streaming versus Almacenamiento Local

Hay dos opciones para llevar contenido multimedia a dispositivos móviles: reproducirlo de forma local o en streaming en tiempo real desde un servidor.

Para realizar streaming de contenido a través de redes móviles relativamente inestables, se desarrolló un protocolo específico denominado RTSP que resuelve los problemas de latencia y buffering. Las frame rate típicas son 15 fps para MP4 y 25 fps para 3gp, con velocidad de datos de hasta 48 kbps para GPRS (sólo audio), 200 kbps para Edge, 300 kbps para 3G/UMTS/WCMA y 500 kbps para WiFi y 4G. HD-vídeo comienza en 2Mbps y no se recomienda para el streaming (todavía).

El servidor de código abierto de Apple, Darwin Streaming Server², puede servir streaming de vídeo y audio con el máximo nivel de compatibilidad y fiabilidad RTSP combinado con FFM-PEG³, que siempre es una buena opción para realizar streaming de archivos 3gp o mp4.

Cuando el objetivo es Windows Mobile/Phone, Windows Media Server⁴ es el preferido para soportar streaming HTTP. Android 3.0 y superiores también soportan streaming HTTP. Ten en cuenta que se requiere atomic hinting (ver Descarga Progresiva) y que los archivos mp4 son muy estrictos en la codificación (utilizan 15 fps H.264 AAC-LC estéreo de 48 kHz). Sólo los dispositivos HTC Android y Android 4.0 son menos estrictos en formatos de streaming y reproducirán muchas más variaciones de codificación que otras marcas.

Cuando el modo streaming no está disponible en el teléfono, bloqueado por la operadora o en el caso de que desees que el usuario pueda ver el contenido sin establecer una conexión cada vez que lo abra, por supuesto puedes simplemente conectar y descargar el archivo. Esto es tan fácil como el enlace a una descarga en una web estándar, pero los teléfonos móviles pueden ser más estrictos a la hora de comprobar los tipos MIME correctos. Utiliza audio/3gp o video/3gp para archivos 3GP y video/mp4 para archivos MP4.

Algunos teléfonos sólo tienen que utilizar las extensiones de archivo para la detección del tipo de datos, por lo que cuando se utiliza una secuencia de comandos, como `download.php`, un truco muy conocido es añadir un parámetro como `download.php?dummy=.3gp` para garantizar el correcto procesamiento del contenido. Algunos teléfonos no pueden reproducir audio 3gp sin video, pero una solución es incluir

² dss.macosforge.org

³ www.ffmpeg.org

⁴ www.microsoft.com/windows/windowsmedia/forpros/server/server.aspx

una pista de vídeo vacía en el archivo o una imagen fija de la portada del álbum.

Dependiendo de la extensión y el protocolo, los diferentes reproductores podrán manejar la petición. En algunos teléfonos, como los Android, si hay múltiples reproductores multimedia disponibles se abre una ventana popup para permitir al usuario seleccionar uno.

Finalmente, puedes simplemente incluir archivos multimedia en tu aplicación móvil como recursos. En dispositivos Android, asegúrate de dar soporte a archivos localizados en tarjetas SD (Android 3.1 y superiores), lo que requiere el permiso `android.permission.READ_EXTERNAL_STORAGE`.

Descarga Progresiva

Para evitar tener que configurar un servidor de streaming, una buena alternativa es ofrecer descargas progresivas para que los archivos multimedia puedan ser distribuidos desde cualquier servidor web. Para hacer esto, tienes que hacer “hint” en tus archivos. “Hinting” es el proceso de marcar varios lugares en el archivo, de manera que un reproductor móvil pueda comenzar a reproducirlo tan pronto como se haya descargado una pequeña parte de él (por lo general los primeros 15 segundos). Nota: un archivo mp3 no necesita, ni permite, este procesamiento.

Probablemente, el software de hinting de código abierto más fiable es Mp4box⁵.

⁵ gpac.wp.institut-telecom.fr/mp4box/

Conversores de Medios

Para convertir una amplia variedad de medios existentes a formatos compatibles con teléfonos móviles, el conversor de formatos FFMPEG (open source) es básico. Puede ajustar a la vez frame rate, bit rate y canales. Asegúrate de incluir soporte a los codificadores H263, H264, AAC y AMR. Hay buenos conversores disponibles en FFMPEG, tales como “Super” de eRightSoft⁶. Para usuarios Mac, QuickTime pro (versión de pago) es una buena alternativa para codificar y realizar hints sobre archivos 3go y mp4. Si buscas una solución completa de servidor con base Java/opensource, echa un vistazo a Alembik⁷.

⁶ www.erightsoft.com/super

⁷ www.alembik.sourceforge.net





Implementando Servicios Basados en Localización

Los servicios basados en localización son uno de los temas candentes en las aplicaciones móviles. Aunque nadie ha demostrado aún que la información de localización sea lucrativa en sí misma, las aplicaciones que contienen un componente geográfico suelen dirigir a servicios más relevantes, que a su vez pueden contribuir a unos mayores ingresos. Conocer la ubicación de un usuario significa que puedes enviarle información más relevante, ayudándole a encontrar un restaurante cercano, teniendo en cuenta el pronóstico local del tiempo, encontrando a sus amigos, o facilitándole el descubrimiento de rutas pintorescas para bicicleta que hayan sido recomendadas por otros ciclistas. Sin embargo, obtener datos de localización es sólo la mitad de la historia, ofrecer al usuario una representación con significado es un factor clave en muchas de las aplicaciones, lo que normalmente significa mostrar una representación gráfica superpuesta con rutas, puntos de interés, etcétera. Aún así, una lista completa de recursos seleccionados por proximidad puede ser muchas veces más fructífera que una lenta vista de mapa con scroll. En resumen, los servicios de localización suelen trabajar silenciosamente en segundo plano y no siempre son transparentes para el usuario final.

Cómo Obtener Datos de Posición

Las aplicaciones basadas en localización pueden adquirir información de ubicación de varias fuentes: a través de una de las conexiones de red disponibles, satélites GPS, sistemas de corto alcance basado en etiquetas visibles o radio de alcance local, o introduciendo datos a través de la pantalla o el teclado, a la manera antigua.

- **Posicionamiento por red:** Cada estación base GSM o UMTS contiene un identificador único, que contiene su código de país, ID de red, cinco dígitos de Location Area y dos dígitos de Routing Area. Las coordenadas de una estación de base pueden obtenerse consultando la declaración de la operadora en una base de datos. Esta información no es particularmente exacta para señalar nuestra ubicación exacta y depende del tamaño de la célula (cobertura de la estación base): Las células se colocan más densamente en las áreas urbanas, que proporcionan una mayor precisión que en las zonas rurales. Técnicas, tales como la medición de la diferencia en el tiempo de llegada de las señales de varias estaciones base cercanas (conocido como multilateración) pueden ayudar a mejorar la precisión, mientras que los proveedores de telefonía cobran por estos servicios premium de red. En los teléfonos con capacidades WiFi, las listas de puntos conocidos de acceso LAN inalámbrico son utilizados por las empresas, incluido Google.
- **Posicionamiento GPS:** Un módulo de GPS a bordo (o uno externo) normalmente da una precisión del 50%, que va oscilando entre 5 y 50 metros, dependiendo de la calidad del hardware y el número de los satélites que el módulo GPS encuentra en el cielo en un momento dado. La precisión se ve afectada también por el terreno, el follaje o los

materiales de los muros; cualquiera de ellos puede ocultar las señales satélite: En las ciudades, cañones urbanos creados por grupos de edificios altos pueden distorsionar la señal, dando lecturas falsas o imprecisas. La combinación de GPS con el posicionamiento de la red es cada vez más común: el GPS asistido, o A-GPS, utiliza un intermediario llamado servidor de asistencia con el fin de minimizar el retardo en la identificación de la primera posición GPS. El servidor utiliza los datos orbitales, el tiempo exacto de red y análisis de la información GPS. Sin embargo, el A-GPS no significa obligatoriamente una posición más precisa, sino más bien un resultado más rápido cuando el GPS está activado inicialmente, o cuando la cobertura del satélite GPS es pobre. Esto acorta el tiempo necesario para la fijación de la ubicación. Nota: la mayoría de las soluciones A-GPS requieren una conexión red activa en el móvil.

- **Posicionamiento de Corto Alcance:** Los sistemas basados en sensores, NFC (del inglés Near Field Communication), Bluetooth y otros sistemas de etiquetado basados en radio, utilizan sensores activos o pasivos en la proximidad de los puntos de interés, como exposiciones en museos o tiendas en un centro comercial. Soluciones tecnológicas de bajo nivel incluyen los códigos de barra y otras etiquetas visuales (como los códigos QR) que pueden ser fotografiados y analizados en un servidor o localmente en el teléfono; esas etiquetas pueden contener un identificador desde el que se puede buscar una posición. El usuario puede especificar su posición mediante la selección de una ubicación en un mapa, ingresar un código de área o una dirección física. Esta opción se utiliza típicamente para las aplicaciones en los teléfonos de gama media, que pueden carecer de otros medios para determinar una ubicación.

Servicios de Mapas

Poca gente se perdió la bien poco exitosa presentación de los mapas de Apple en la segunda mitad de 2012, donde ciudades, monumentos y calles estaban desaparecidos o eran incorrectos, ni tampoco el lanzamiento de Google de una aplicación de mapas para iOS justo antes de Navidad.

En general, un servicio de mapas toma una posición y un puñado de metadatos como parámetros de entrada y devuelve un mapa, con capas adicionales de metadatos contextuales. El propio mapa puede estar en la forma de uno o más bitmaps, vectores o una combinación de ambos. Los datos en vector tienen varias ventajas sobre los bitmaps: las representaciones vectoriales consumen menos ancho de banda y permiten hacer zoom arbitrariamente desde el espacio hasta tu cara. Sin embargo requiere más procesamiento en el lado del cliente. Los bitmaps a menudo se proporcionan en niveles de zoom discretos, cada una con un nivel de aumento fijo, nombrados según sus coordenadas.

Mapas gratuitos, tanto bitmaps como vectores, son Open Street Map¹ o CloudMade². Los comerciales incluyen mapas de Garmin³ o los recursos Bing de Microsoft⁴, entre otros.

Algunas soluciones, como Google Maps⁵, son gratuitas cuando la aplicación está disponible sin coste alguno, pero requieren que obtengas una clave de mapa. Otros servicios de mapas, como los mapas estáticos de Google, se limitan a mostrar una serie de secciones de imagen (tiles) para una clave de mapa o dirección IP. Muchos de estos recursos comparten formatos similares formatos de mapa y son por tanto intercambiables.

¹ wiki.openstreetmap.org/wiki/Software

² www.cloudmade.com

³ garmin.com

⁴ www.microsoft.com/maps/developers

⁵ www.code.google.com/apis/maps

Implementando Soporte a Localización en Diferentes Plataformas

La API de localización para Java ME ofrece detalles tales como latitud y longitud, precisión, tiempo de respuesta, y la altitud derivada del GPS de a bordo, así como la velocidad basada en la realización de lecturas consecutivas.

iOS tiene integrado soporte a la ubicación, pero con restricciones acerca de cómo los datos de localización pueden ser generados por sus funciones. En la actualidad, también hay un debate en curso sobre cómo los datos de ubicación se registran y almacenan en los dispositivos iOS y en cómo Apple planea utilizar tales datos para sus propios fines. Los desarrolladores de Android también tienen acceso a las librerías de alto nivel, y esos dispositivos son más liberales con la elección de las fuentes de mapas, aunque usan por defecto la API de mapas de Google. En dispositivos Symbian, Nokia Maps se puede utilizar de forma gratuita, incluso para uso comercial.

Desde iOS 3.x y Android 2.0, los desarrolladores de aplicaciones web han sido capaces de acceder a información geográfica a través de la interfaz `navigator.geolocation`, por ejemplo llamando a `navigator.geolocation`

```
.getCurrentPosition(my_geo_handle) puedes capturar  
themy_geo_handle.coords  
.latitude y la my_geo_handle.coords.latitude, siem-  
pre que el usuario lo autorice y los satélites están disponibles.  
A través de programación avanzada, esta acción se puede  
combinar con fallbacks a nuevas búsquedas de red.
```

Los datos geográficos a menudo son presentados con información adicional, disponible en un número de formatos. Uno de los estándares ampliamente aceptados se llama geoRSS, y tiene este aspecto para un único punto de interés:

```
<entry>
  <title>Byviken's fortress</title>
  <description>Swedish 1900-century army
    installation, w. deep mote
  </description>
  <georss:point>18.425 59.401</georss:point>
</entry>
```

Existen otros formatos de datos geográficos, pero la idea básica es similar, armonizando flujos de datos y servicios web, se pueden crear mashups sólidos para ser ejecutados sin problemas en diversos contextos. Otros formatos importantes para incluir geoinformación son el Geography Markup Language (GML), una codificación XML específica para la transmisión y almacenamiento de información geográfica, así como KML, un elaborado geoformato que se utiliza en Google Earth y servicios web relacionados.

Herramientas Para Aplicaciones LBS

Varias compañías ofrecen herramientas y APIs cómodas para desarrolladores como un servicio de valor añadido; utilizarlas acelera drásticamente el desarrollo y despliegue de servicios basados en localización. Cada herramienta se enfoca normalmente a una o una pequeña selección de las plataformas móviles.

Empresas de publicidad como AdMob ofrecen a los desarrolladores un programa de publicidad independiente y sensible a localización para orientar mejor sus ofertas, aunque no se visualizan interfaces de mapas, sólo las coordenadas.

A continuación, más enlaces a mapas y recursos basados en la localización de servicios:

- **Garmin Mobile XT SDK:** *developer.garmin.com*
- **Android offline maps project:** *code.google.com/p/big-planet-tracks/*
- **TomTom:** *developer.tomtom.com*
- **Nutiteq:** *www.nutiteq.com*
- **RIM:** *us.blackberry.com/developers/* (buscar por “map api”)
- **Nokia Maps:** *developer.here.net*
- **Nokia Windows 8 Mapexplorer:**
projects.developer.nokia.com/mapexplorer
- **Windows Phone:** *msdn.microsoft.com/en-us/library/windowsphone/develop/ff431803*
- **Recursos de Google Maps:**
developers.google.com/maps/mobile-apps



Comunicaciones de Campo Cercano

A pesar de que en los últimos tres años ha aumentado el interés en las Comunicaciones de Campo Cercano (o NFC, del inglés Near Field Communications), NFC no es en realidad una tecnología nueva. NFC es una evolución de la identificación por radiofrecuencia (RFID), que ha estado en funcionamiento desde principios de los 90¹. NFC extiende las capacidades de RFID y, al mismo tiempo, mantiene la compatibilidad con las tecnologías más antiguas. Durante años, la tecnología RFID se ha utilizado principalmente para el seguimiento de objetos y el pago por contacto. La tecnología NFC, sin embargo, permite algo más que hacer pagos. NFC es una interfaz y protocolo construido sobre RFID y que está dirigido a los dispositivos móviles. La tecnología funciona sobre banda ISM sin licencia (del inglés Industry Scientific Medical) de 13,56 MHz que limita el rango de operación a 3 cm o contacto directo, y proporciona a los dispositivos móviles un medio de comunicación seguro sin ninguna configuración de red. Esto es una ventaja añadida respecto a Bluetooth, que requiere aparear los dispositivos para establecer comunicación.

¹ www.nearfieldcommunication.org/history-nfc.html

Ventajas de NFC sobre Bluetooth²:

1. **Seguridad:** El corto campo de operación de NFC es en realidad una ventaja. Algunas implementaciones en smartphones requieren que la pantalla esté activa y que se haya introducido el PIN para obtener permiso de acceso a hardware NFC.
2. **Bajo consumo de energía:** Debido al pequeño radio de operación, sólo es necesario generar un campo electromagnético muy débil para escribir o leer etiquetas NFC.
3. **Pareado rápido:** Los dispositivos NFC pueden conectarse en una décima de segundo.

Breve resumen del funcionamiento de NFC

NFC trabaja por inducción electromagnética. Cuando las ondas electromagnéticas en una bobina cambian, se induce un voltaje. Desde la perspectiva del hardware, una serie de voltajes altos y bajos representan bits. Así es como se envían los paquetes entre dispositivos. En 2006, el NFC Forum, que supervisa el ecosistema NFC, definió las especificaciones NFC. El Foro estableció un estándar para el formato de intercambio de datos en NFC (NDEF, del inglés NFC Data Exchange Format), un mensaje de bajo peso en formato binario que se encarga de encapsular los datos. Por ejemplo, para codificar una URI tiene una cabecera de 5 bytes y puede ser tan breve como aproximadamente 12 bytes para transmitir una URL corta. Se puede encontrar más información online³.

2 www.nearfieldcommunication.org/bluetooth.html

3 [www.developer.nokia.com/Community/Wiki/Understanding_NFC_Data_Exchange_Format_\(NDEF\)_messages](http://www.developer.nokia.com/Community/Wiki/Understanding_NFC_Data_Exchange_Format_(NDEF)_messages) y www.nfc-forum.org/specs/

Modos de operación NFC⁴

1. **Modo Lectura/Escritura:** En este modo, el smartphone actúa como un lector, generando campos electromagnéticos para leer las etiquetas NFC. Sólo es posible escribir en un mensaje NDEF en una etiqueta NFC si la etiqueta no es de sólo lectura.
2. **Modo Punto-a-Punto (Peer-to-Peer):** Se trata de una importante ampliación de la tecnología RFID. En el modo P2P, dos smartphones son capaces de intercambiar pequeñas cantidades de información, tales como vCards, URLs o iniciar una conexión Bluetooth para la transferencia de datos de gran tamaño. Android Beam normalmente funciona en modo P2P de manera que el pareado de un “usuario invisible” se realiza con NFC, y la transferencia de datos se ejecuta a través de la conexión Bluetooth con mayor rapidez.
3. **Modo Emulación de Tarjeta:** Con mucho, la forma más interesante de operar. En este modo, el smartphone actúa como un etiqueta NFC pasiva. El modo de emulación de tarjeta es crucial para fines de pago en los que se almacena información sensible en un elemento seguro (ver más adelante). Google Wallet y Microsoft Wallet son ejemplos de aplicaciones que permiten a un teléfono inteligente equipado con NFC ser utilizado para realizar pagos por contacto basándose en la emulación de la tarjeta. Actualmente, la última versión disponible de Android (Jelly Bean) no tiene APIs públicas para la emulación de tarjeta, principalmente porque todavía no existe un estándar para el ecosistema NFC (ver la sección Problemas actuales).

⁴ www.nfc.cc/technology/nfc/

El Elemento Seguro

En el corazón de cualquier dispositivo de tarjeta emulada se encuentra el elemento seguro (o SE, del inglés Secure Element)⁵ que contiene los datos asegurados (información de tarjetas de crédito, entre otros⁶). Hay tres posibles lugares en los que el SE puede ser almacenado:

1. En la SIM/UICC (vía un protocolo de cable único, del inglés Single Wire Protocol, una especificación que permite la conexión entre la tarjeta SIM y el chip NFC)
2. En el interior del chip NFC
3. En una tarjeta SD

Como nota al margen, Google Wallet⁷ sólo almacena las credenciales de una tarjeta de prepago Google en el elemento seguro del chip NFC. Sólo los números de las tarjetas de Google se pasan a los vendedores reales, mientras que los números de tarjeta de crédito reales se almacenan en servidores seguros de Google. Microsoft Wallet, por el contrario, almacena elementos sensibles en el elemento seguro de la tarjeta SIM. Según Microsoft, este método permite a las personas mover su cartera de un teléfono a otro.

En todo el mundo, un número creciente de entidades de crédito y operadoras de redes móviles (MNOs, del inglés Mobile Network Operator) están cooperando para implementar métodos de pago que utilicen NFC. MasterCard PayPass y Visa PayWave son ejemplos de tales soluciones implementadas. Operadoras de redes móviles tales como A1 (Austria), Orange Francia y China Mobilee están emitiendo tarjetas NFC SIM.

⁵ nearfieldcommunication.com/developers/nfc-architecture/

⁶ www.smartcardalliance.org/

⁷ www.google.com/wallet/faq.html

Usos actuales de NFC

1. **Modo Lectura/Escritura:** Funcionalidad RFID básica que incluye una etiqueta NFC. Las etiquetas pueden ser utilizadas para múltiples propósitos (por ejemplo, definir perfiles para un smartphone, guardar una pequeña cantidad de datos tales como una URL o información de contacto, entre otros).
2. **P2P,** para dos smartphones equipados con NFC: Pareado, Intercambio de datos.
3. **Emulación de tarjeta:** Tickets, Pagos, Operaciones de conmutación (por ejemplo apertura de puertas), reemplazo de targetas (sanitarias, de crédito o de conducción, entre otras).

Dificultades actuales

NFC es una tecnología fascinante que atraerá transacciones económicas. Sin embargo, antes de que veamos un amplio despliegue de medios de pago mediante NFC, son necesarias su plena comprensión y la cooperación entre todas las entidades bancarias, fabricantes de hardware, operadores de redes móviles y desarrolladores de sistemas operativos.

Además, algunos fabricantes de teléfonos importantes no han adoptado aún la tecnología NFC. Los dispositivos actuales de Apple, por ejemplo, no son compatibles con NFC. La compañía ha decidido confiar en la aplicación Passbook, que tiene un modo completamente diferente de operar respecto a las aplicaciones centradas en NFC.

Aún más, el elemento seguro tiene un espacio de almacenamiento limitado, y no está claro cómo este espacio debe ser compartido entre todos los implicados. Por último, debido a la falta de estándares aceptados, algunos bancos han desplegado

sus propias soluciones y quieren convencer a los comerciantes de aceptar su nuevo sistema de pago.

Implementación de NFC

Windows Phone 8: El WP8.0 SDK ofrece los paquetes Proximity, que proporcionan un conjunto de clases con las APIs necesarias para habilitar el intercambio de datos P2P entre aplicaciones WP8. También es posible transferir pequeños paquetes de datos desde un dispositivo Android a un dispositivo WP8 y viceversa. En la actualidad, su implementación se encuentra todavía en sus inicios y no es posible transferir gran cantidad de datos.

Puedes encontrar más detalles online sobre la implementación en⁸.

Android: Desde la API nivel 9 (Gingerbread 2.3), Android provee un conjunto de APIs de alto nivel que facilitan su uso. Más información en la página de Android para desarrolladores⁹.

BlackBerry: Su último SDK provee de APIs de alto nivel para NFC. Ejemplos de código para implementar un lector y grabador de etiquetas NFC¹⁰, e información adicional en los sitios web de BlackBerry sobre cómo usar el modo emulación en BlackBerry¹¹.

⁸ msdn.microsoft.com/en-us/library/windowsphone/develop/jj207060

⁹ developer.android.com/reference/android/nfc/package-summary.html

¹⁰ docs.blackberry.com/en/developers/deliverables/34480/Near_Field_Communication_1631111_11.jsp

¹¹ supportforums.blackberry.com/t5/Java-Development/NFC-Card-Emulation-Primer/ta-p/1596893



Implementando Vibración Háptica

Consideraciones de Diseño de Vibración Háptica

¿Por qué deberías usar efectos hápticos de vibración? Tu aplicación va a funcionar igual de bien sin la retroalimentación táctil, ¿verdad? Sí, tal vez, pero perderá el único elemento sensorial que hace su entorno virtual más realista y convincente. Margaret Atwood escribió una vez: "El tacto va antes que la vista, antes que hablar. Se trata de la primera lengua y la última, y siempre dice la verdad". Es este sentido de retroalimentación táctil, más que la vista o el oído, el que nos enseña qué esperar de las interacciones en el mundo real.

Son nuestras experiencias en el mundo real las que definen las expectativas de un usuario en el mundo virtual que se encuentra dentro de tus aplicaciones.

Toma, como ejemplo, pulsar un botón. Una pulsación real de botón es una experiencia muy táctil. Tiene un comienzo y un final. Sientes una satisfactoria confirmación de tu acción. En comparación, pulsar un botón virtual se percibe hueco, sin efecto háptico para simular esa misma confirmación de la acción. Aún más, sin una confirmación táctil háptica obligas al usuario a confiar en las señales visuales o auditivas, que son más estresantes que el simple uso del sentido del tacto.

La retroalimentación háptica es aún más importante en los videojuegos móviles; lo sabemos por nuestra experiencia con los juegos de consola. ¿Recuerdas cuando Sony lanzó la PS3 sin "DualShock" rumble pad? Los jugadores expresaron su

descontento y, poco después, Sony incorporó la retroalimentación DualShock a PS3. La misma satisfacción respecto a la retroalimentación háptica se aplica a los juegos móviles. El uso de efectos hápticos en tus juegos te ayudará a dar a tus usuarios móviles lo que ya esperan de las consolas y, si diseñas bien tus juegos, tus usuarios los percibirán como más realistas y convincentes.

Al diseñar una experiencia háptica, ten en cuenta la última experiencia del usuario. Dedicar algún tiempo a la planificación antes de iniciar tu implantación háptica. Una vez que el proyecto está definido y tomando forma en tu mente, considera las siguientes pautas:

- Las sensaciones simples son a menudo las más efectivas. A veces es sorprendente darse cuenta de que algo como una muy simple sensación Pop o Click puede mejorar las interacciones del menú y aumentar la confianza del usuario en la aplicación.
- Sensaciones sincronizados con los eventos de audio e imagen, como un sencillo evento de click sobre un botón, hacen que el total sea mayor que la suma de sus partes. Ver, oír, y sentir un objeto o actividad favorece la armonía sensorial de una manera superior a sólo viendo y escuchando.
- Es malo molestar al usuario. Las sensaciones táctiles mal elegidas pueden ser molestas y contraproducentes. Mientras que un zumbido agudo puede ser muy eficaz como parte de una alerta, si es continuo o recurrente puede causar que el usuario abandone la aplicación irritado.
- Es malo confundir y abrumar al usuario. Así como reproducir demasiados sonidos bonitos simultáneamente se convierte en una cacofonía, demasiadas sensaciones convincentes ejecutadas a la vez o demasiado seguidas en

tiempo y espacio pueden llegar a ser confusas y agobi-antes.

- La familiaridad facilita la experiencia del usuario. Los efectos hápticos pueden transmitir información importante al usuario que puede no ser posible o práctico proporcionar a través de gráficos o de sonido. Normalización y consistencia son importantes. La limitación del lenguaje de efectos hápticos a un rango de sensaciones manejable hace que el proceso de aprendizaje del usuario sea más fácil porque hay menos efectos hápticos que reconocer.

Casi todas las plataformas móviles permiten alguna forma de control de retroalimentación háptica por vibración. Esta sección será tu recurso para la comprensión de las clases y métodos en ellas.

iOS

iOS es probablemente la plataforma con menor documentación sobre vibración para desarrolladores, ya que Apple permite a los desarrolladores muy poco control sobre la vibración de los dispositivos. El método de vibración iOS indicado a continuación se aplica sólo a los iPhones. iPads e iPods no tienen soporte a las vibraciones actualmente.

Utiliza la clase `SysSoundViewController`¹ con la función `AudioServicesPlaySystemSound` y la constante `kSystemSoundID_Vibrate` para iniciar la vibración en el iPhone. La llamada a esta constante mantendrá el motor activo durante unos 2 segundos.

¹ developer.apple.com/search/index.php?q=SysSoundViewController

Android

Android es único para controlar las vibraciones. Ofrece soporte nativo y tiene más control de la vibración que iOS. Además, hay maneras de extender este control de vibración para crear experiencias de retroalimentación al estilo de consolas como X-Box o PlayStation. Pero, tanto si utilizas los métodos básicos como los extendidos indicados a continuación, por favor ten en cuenta que un usuario puede tener habilitados los efectos hápticos con fines de accesibilidad. Por ejemplo, el Servicio de Accesibilidad KickBack proporciona retroalimentación háptica y está disponible como parte del proyecto de código abierto eyes-free². Por lo tanto, ten en cuenta cómo los efectos hápticos generados por tu aplicación puede interactuar con, o perturbar, tales servicios.

Para un control básico de la vibración en Android, primero debes habilitar el permiso `android.permission.VIBRATE` para permitir a tu aplicación vibrar. A continuación, usa la clase `Vibrator`³ con la función `getSystemService` y `Context.Vibrator_Service` para llamar al servicio de vibración.

En el método anterior se puede variar la duración del evento en milisegundos y establecer patrones de vibración mediante la creación de tantos eventos de inicio y reposo como desees. El método básico de control de vibración Android sólo permite controlar la duración de los eventos de vibración.

² code.google.com/p/eyes-free

³ developer.android.com/index.html?q=Vibrator

Control Extendido de Vibración Android

Debido a que la plataforma Android es de código abierto, hay al menos una empresa que ofrece métodos gratuitos para extender el control de la vibración en ella. Haptic SDK⁴ de Immersion Corporation permite un control total de la duración del motor de vibración, su amplitud y frecuencia pulsante con una librería de más de 120 efectos predefinidos de retroalimentación háptica de vibración. Con este tipo de control, los desarrolladores de aplicaciones tienen la capacidad de diseñar efectos de vibración que rivalizan con los de consolas de videojuegos, al mismo tiempo que ahorran batería.

Para los desarrolladores Android, Unity3D Pro ofrece ese mismo método de inmersión prolongada a través de un plug-in que también se puede encontrar en la página web del SDK. Los desarrolladores interesados en este control de vibración prolongada pueden descargar la guía de empresa Quick Start Guide⁵, que explica cómo configurar tu entorno Eclipse y utilizar el método `Launcher` para llamar a los efectos hápticos de la librería predefinida. En Google Play también se puede descargar una aplicación de demostración para probar los efectos hápticos prediseñados antes de usarlos en el código. La aplicación se llama "Haptic Effect Preview".

Además de tener efectos hápticos prediseñados para desarrolladores, hay una capa de abstracción de hardware en la librería Immersion que compensa las diferencias entre tipos de motor de fabricantes de hardware.

⁴ immersion.com/haptic/sdk

⁵ immersion.com/haptic/guide

BlackBerry 10

BlackBerry ofrece el mismo control básico de vibración que Android, pero sin extensiones. Utiliza la clase `VibrationController`⁶ con `startVibrate(int duration)` y `stopVibrate(int duration)`.

Además, ahora BlackBerry incluye un parámetro de intensidad (1-100) a disposición de los desarrolladores.

Windows 8

Windows ofrece un método básico de control de vibración, pero actualmente sin extensiones de método. Usa la clase

`VibrateController`⁷ con los métodos `Start` & `Stop` para hacer vibrar el motor del dispositivo de 0 a 5 segundos. Para un control más afinado de la duración, necesitarás añadir un método `TimeSpan` para poder usar valores de milisegundos.

La clase mencionada de Windows 8 es la misma que la clase previa en Windows 7.

⁶ developer.blackberry.com/search/?search=VibrationController

⁷ social.msdn.microsoft.com/search/en-us?query=VibrateController



Implementando Realidad Aumentada

La realidad aumentada (RA, del acrónimo inglés AR a partir del término *Augmented Reality*) es una tecnología que amplía el mundo real con elementos virtuales. Componentes reales y virtuales se combinan en tiempo real para aumentar la percepción de la realidad, añadiendo información adicional a un entorno del mundo físico. Aunque la realidad aumentada visual es su forma más común, esta tecnología puede apelar a todos los sentidos. La información añadida puede incluir un objeto tridimensional que se funde con el entorno real, una superposición de dos dimensiones que contiene texto, o simplemente un archivo de audio.

El pronóstico para el 2013 es de 200 millones de usuarios de realidad aumentada, mientras que para el 2020 se estima que el número total de consumidores que utilicen aplicaciones de RA suba hasta los mil millones. Según 'Research and Markets', el crecimiento del mercado de RA es exponencial. Con un crecimiento de ventas de 181 millones de dólares (USD) en 2011, se espera un incremento a una tasa de crecimiento del 95,35%, hasta alcanzar 5.155 millones en el año 2016. Datos disponibles en línea.¹

¹ www.researchandmarkets.com/reports/1963197/

Escenarios de Uso de RA en Móviles

La RA Móvil se utiliza en situaciones en que la información adicional puede aumentar la eficiencia, la efectividad y la satisfacción mientras se está utilizando. Es especialmente adecuada para aquellas aplicaciones en las que las personas se enfrentan con una gran cantidad de datos o una carga elevada de información y necesitan procesarlos en un corto período de tiempo. A través de la inserción de información virtual en la transmisión en tiempo real en pantalla, la atención del usuario ya no necesita cambiar entre la pantalla del dispositivo móvil y el entorno. Las soluciones móviles de Realidad Aumentada tienen un muchas áreas de aplicación, tales como la industria, el marketing, la educación o simplemente para entretenimiento. Éstos son sólo algunos ejemplos:

- **AR Jump'N'Run:** AR Jump'N'Run² es una aplicación basada en localización para smartphones Android desarrollados con DroidAR³. Se puede jugar en interiores y exteriores, ya sea usando el sistema de posicionamiento global (GPS), detección de pasos, o ambos. El jugador camina a través de un mundo enriquecido virtualmente utilizando su dispositivo Android como visor y recogiendo objetos en 3D; estos pueden ser hostiles o amigables, y tienen diversas consecuencias para la partida. El juego también incluye un editor de mapas que permite a los jugadores crearlos y modificarlos directamente en sus dispositivos.
- **Navegadores de Realidad Aumentada:** Navegadores-RA como Layar y Wikitude pueden sobreimprimir la visión en directo del mundo físico real que te rodea con datos basados en la localización. La ubicación del usuario se

² bitstars.com/?page_id=119

³ github.com/bitstars/droidar

determina por GPS y la información sobre los PDIs (puntos de interés o POIs, acrónimo del término inglés Point Of Interest) cercanos se muestra en la pantalla del smartphone. En la mayoría de casos, la información se muestra como iconos seleccionables o fragmentos de texto. Wikitude, además, ofrece una conexión a la Wikipedia para más información. Puedes descargar y probar las aplicaciones en la página web de la aplicación de Wikitude⁴ y la página web de Layar⁵.

- **Aplicación de Catálogo IKEA:** La aplicación IKEA de RA se ha realizado por y con Metaio⁶. En lugar de utilizar códigos QR, se basa en el software de reconocimiento de imágenes de Metaio. Al escanear con el dispositivo móvil las páginas del catálogo de Ikea que muestran un icono de "desbloqueo de RA", se muestra información adicional, como las opciones de personalización y más imágenes de producto. El usuario también puede ver cómo los muebles de IKEA se verán en su hogar mediante la colocación de los modelos 3D en la cámara del dispositivo. Más información sobre el conjunto de características y un vídeo de presentación en el blog de desarrolladores de Metaio⁷.

Seguimiento

Determinar la ubicación es una parte esencial de los sistemas modernos de realidad aumentada. Es tan necesario determinar la posición del usuario como colocar los objetos virtuales

⁴ www.wikitude.com/app

⁵ www.layar.com/features/#feature-layarapp

⁶ www.metaio.com

⁷ augmentedblog.wordpress.com/2012/07/24/ikea-2013-catalog-has-augmented-reality/

correctamente en una habitación tridimensional. Tecnologías comunes de seguimiento incluyen GPS, sensores ópticos, brújula, acelerómetro, giróscopo y detección de pasos. Otro concepto importante es el uso de sistemas de seguimiento con o sin marcadores.

Seguimiento basado en marcadores

Los marcadores son una solución simple y económica para identificar objetos. Aún más, la precisión es muy alta siempre y cuando no haya una gran distancia entre el marcador y la cámara. El seguimiento con marcadores funciona con el principio de reconocimiento de patrones. Los marcadores están con frecuencia en blanco y negro, y con forma cuadrada para facilitar la captura y el procesamiento de la orientación a realizar por el sistema de procesamiento de imagen. Este sistema de procesamiento puede funcionar en rangos de luz normal o infrarroja. Al procesarse la información acerca del tamaño real del marcador, la distancia entre el dispositivo y el marcador puede ser calculada. También es posible el uso de reflectores esféricos que reflejan la luz incidente en la dirección de la que procede.

Seguimiento sin marcadores

Un escenario de implementación similar, pero ligeramente diferente, utiliza los objetos normales en lugar de marcadores. Estos objetos pueden ser patrones de dos dimensiones (por ejemplo, carteles publicitarios) o incluso aquellos en tres dimensiones en los alrededores (por ejemplo, edificios). Las imágenes grabadas se comparan con



una base de datos para detectar coincidencias, lo que requiere algoritmos complejos y gran potencia de procesamiento. Los cambios en las condiciones de iluminación y el movimiento del objeto pueden hacer difícil de encontrar la combinación correcta. Otro desafío es el reconocimiento de objetos cuya forma no es fija.

Seguimiento híbrido

La tecnología de seguimiento híbrido combina las diferentes fuentes de datos de posición, tales como el GPS, la detección de objetos en 3D, la detección de marcadores y la detección de pasos. Esto permite una mayor precisión en posicionamiento y detección de movimiento.



SDKs de Realidad Aumentada

Wikitude

Wikitude es un navegador de puntos de interés basado en la localización. Un escenario de uso clásico en el que Wikitude ofrece una solución adecuada es la búsqueda de tales puntos (por ejemplo, "¿Dónde está la oficina de correos más cercana?"). Wikitude está diseñado para contenido estático y no permite escenarios interactivos. Si utilizas las versiones más recientes del SDK para construir tu propia aplicación Wikitude, también puedes utilizar HTML5, Titanium o Phonegap. Incluso BlackBerry 10 es compatible.

www.wikitude.com/developer

Vuforia

El SDK Vuforia de Qualcomm sólo admite la extracción de elementos en 2D y el reconocimiento de imágenes especiales almacenadas localmente. Vuforia no se puede utilizar para crear aplicaciones basadas en la localización que utilicen datos de movimiento o geo-referencias.

www.vuforia.com

metaio

El SDK metaio es una de las herramientas de desarrollo RA más avanzadas y está centrada en las imágenes 2D, por ejemplo en revistas o catálogos (como la aplicación de Ikea mencionada antes). La versión Pro del SDK también soporta el reconocimiento de objetos 3D, como el embalaje de un producto, una estatua o la fachada de un edificio. Como aspecto negativo, el SDK ofrece muy pocas posibilidades de implementar escenarios de interacción individuales. El desarrollador sólo puede decidir entre la captura de elementos 2D o 3D y objetos basados en localización.

www.metaio.com

Layar

En sus inicios, Layar era una plataforma RA basada en la localización, con capas de fundido de entrada y muy limitadas posibilidades de interacción para el usuario. Actualmente, Layar cambiado su enfoque de capturar de elementos 2D a aumentar imágenes. El nuevo SDK y las herramientas afines (llamadas creadores de capa) están especialmente diseñados para extender la sobreimpresión de contenido multimedia. El Layar Player SDK para iOS permite construir aplicaciones que no necesitan el navegador Layar.

<http://www.layar.com>

DroidAR

DroidAR SDK se creó para el desarrollo de aplicaciones de RA interactivas basadas en la localización. DroidAR proporciona un seguimiento híbrido y está basado en marcadores, pero no aumenta imágenes 2D. Su fuerza reside en construir aplicaciones que se utilizan en movimiento, por ejemplo, juegos de RA o aplicaciones de compra de productos. Por el momento, DroidAR sólo es compatible con dispositivos Android.

github.com/bitstars/droidar

D'Fusion







































































El SDK D'Fusion de Total Immersion ofrece captura de elementos 2D para aumentar imágenes y es muy similar a otros SDKs como el de metaio, Qualcomm y Layar. En algunos paquetes del SDK se ofrece rastreo de la cara humana y librerías de detección de movimiento.

www.t-immersion.com

ARToolwork

ARToolworks ofrece SDKs tales como NyARToolkit. Sus SDKs sólo ofrecen realidad aumentada basada en marcadores y se distribuyen bajo un modelo de licencia equitativo.

www.artoolworks.com/products/mobile/

SDKs de Realidad Aumentada	Wikitude SDK	Vuforia	Qualcomm Metaio	Layar	DroidAR	D'Fusion Total Immersion	ARToolworks
RA basada en localización							
iOS y Android							
SDK para RA interactiva							
Seguimiento híbrido							
Detección de marcadores							
Aumentado de imagen 2D							
Potencial de escalado							
Detección de pasos							
Forum de desarrolladores							
Costes de licencia							

Introducción al Desarrollo de RA

Esta sección te dará una introducción general a los conceptos clave necesarios para crear aplicaciones de RA. Una vez que entienda estos conceptos, deberías ser capaz de elegir el framework adecuado para tu proyecto.

El Mundo Real Y El Mundo Virtual

RA significa colocar objetos artificiales en el mundo real, de alguna manera, utilizando una capa virtual. Esta capa virtual, o mundo virtual, y su sistema de coordenadas están ligados al mundo real por puntos de referencia. Una referencia puede ser una posición de GPS en el caso de una aplicación de guía turística que proporciona información basada en la ubicación sobre PDIs, o también puede ser un marcador visual, por ejemplo un juego impreso en una caja de cereales de desayuno, que se podría jugar en cualquier parte del mundo y sólo necesitaría la referencia relativa a esa caja.

Mapeando Entre Dos Mundos

Para aplicaciones RA basadas en localización se requiere un mapeo mutuo entre la posición en constante cambio en el mundo real y la posición en el mundo virtual. Motores de renderizado como OpenGL v1 y v2 reducen la complejidad de este proceso y aumentan velocidad y precisión en tiempo real. El motor también se encarga de hacer coincidir las posiciones en el mundo virtual y real de la cámara, y garantiza transiciones fluidas en la cámara cuando estas posiciones cambian. Para hacerte la vida más fácil, como desarrollador puedes utilizar motores extendidos como gameplay⁸ o Unity⁹.

⁸ www.gameplay3d.org/

⁹ unity3d.com

Como elemento central, los datos de la cámara también se pueden transmitir a otros componentes. Por ejemplo, un componente de colisión puede calcular fácilmente la distancia entre los objetos virtuales y la imagen capturada por la cámara.

Creando Elementos Virtuales

Los elementos virtuales se representan como objetos 2D y 3D que pueden responder a diferentes lógicas de comportamiento: algunos objetos pueden coleccionarse, otros siguen al usuario y algunos pueden permanecer estáticos y no permitir interacción alguna.

Para crear este tipo de elementos virtuales necesitas una técnica llamada Localización y Mapeado Simultáneos (o SLAM, acrónimo del término inglés Simultaneous Localization And Mapping). Esta es una técnica muy potente de visión por ordenador basada en las secciones identificables de manera inequívoca en las características de una imagen. Estas características se combinan con el mundo real en un espacio 3D en el que el dispositivo se puede mover. La nube 3D creada también se puede utilizar para la detección de un objeto del mundo real o su forma, y posteriormente aumentarlos. Otra parte de la visión por ordenador es el reconocimiento de imágenes. La posición de la imagen reconocida puede ser utilizada como posición de referencia para el mundo virtual (ver el ejemplo anterior de la caja de cereales de desayuno). SLAM y la técnica de visión por ordenador requieren de una inmensa capacidad computacional para ser ejecutadas en tiempo real, así que las limitaciones y capacidades del hardware tienen que ser tomadas en cuenta.

Combinando Capas de Aplicación

Un importante elemento de la RA visual es dibujar algo sobre la imagen recibida desde la cámara. Dependiendo de los requisitos de tu aplicación, tendrás que colocar gráficos en 2D o 3D en esta superficie y utilizar las APIs correspondientes.

Una superposición 2D es suficiente para navegadores PDI simples. Te recomendamos encarecidamente un framework de representación como OpenGL, en lugar de intentar volver a inventar la rueda. Dicho marco utiliza la posición del usuario, orientación del dispositivo, otra información del sensor, o los datos de análisis de la imagen y los traduce para mostrar tu contenido de manera acorde. Si el componente de representación está desacoplado del resto del código de la aplicación podrá ser intercambiado en el futuro, por ejemplo para cambiar a soluciones de renderización más avanzadas.

Tal vez quieras agregar elementos tales como una pequeña interfaz radar para visualizar la posición de los objetos virtuales o algunos botones sencillos para interactuar con el mundo virtual. En ese caso, asegúrate de ceñirte a las pautas específicas de la plataforma y sus diseños, e implementar estos elementos en una capa separada para mantenerla flexible.

Construyendo Un Mundo Virtual Con Múltiples Capas

La mejor manera de componer un mundo virtual es utilizar una estructura de árbol y colocar objetos virtuales en diferentes capas: una capa para objetos fijos “de fondo” que no necesitan ser actualizados o que no interactúan con el usuario, y otras capas para objetos movibles o elementos de interfaz de usuario.

Sólo deberías actualizar y renderizar los objetos próximos al usuario y hacer uso de una estructura de árbol quad. Un árbol quad es una estructura de datos que permite obtener todos los objetos en un cuadrante delimitador de una manera eficiente. Dependiendo del hardware del dispositivo, se puede utilizar

un radio de vista diferente para mantener el rendimiento de la aplicación.

También recomendamos el uso de un mecanismo de actualización que lance las actualizaciones y llame a los métodos, por ejemplo, cada 20 ms. Los nodos en el árbol de objetos deciden individualmente a qué hijos deberían ser enviadas estas llamadas de actualización. Un árbol quad, por ejemplo, sólo actualizará los objetos cerca del usuario para mantener el proceso de actualización eficiente. Una estructura de lista básica actualizaría todos sus hijos, por lo que resulta más adecuada para elementos que no tienen una posición virtual (como estadísticas de juego) o que tienen que ser actualizados a toda costa. El concepto exacto de composición de objetos depende del escenario de aplicación y no se puede definir de una forma universal.





Seguridad De La Aplicación

Los lectores de esta guía saben cómo de extendidos están los smartphones y cómo de útiles pueden ser las aplicaciones móviles. Usamos potentes móviles conectados para una multitud de cosas todos los días. Los dispositivos móviles, además, son también mucho más personales que lo que jamás lo han sido los ordenadores. La gente se despierta con sus teléfonos, permanecen cerca de ellos todo el día, y duermen a su lado por las noches. Con el tiempo se convierten en nuestros ‘colaboradores de confianza’.

Actualmente, las empresas están desarrollando aplicaciones que se aprovechan de tal cercanía y confianza. Por ejemplo, tu teléfono puede ser tratado como un eslabón en el proceso de autenticación para acceder a tu cuenta bancaria, o tu tableta puede tener acceso directo a las películas online que has comprado. Tu dispositivo podría incluso contener una cartera con dinero real para realizar pagos con NFC.

Es evidente que las aplicaciones móviles van a atraer la atención de hackers y ladrones cuyos intereses se extienden mucho más allá de conseguir gratis una aplicación de 99 céntimos. Los datos históricos de la red y las defensas de punto final (como antivirus) no son suficientes. Incorporar medidas de seguridad en una aplicación móvil es un tema crítico.

La arquitectura de las aplicaciones móviles sigue evolucionando. Algunas aplicaciones son sólo nativas, y requieren de bases de código muy diferentes para cada sistema operativo móvil. Otras son webviews, poco más que la URL de un sitio web envuelta en un icono. Algunas son híbridas, una combinación de funcionalidad de aplicación nativa con webviews. La mayoría de aplicaciones móviles necesitan conectarse con los servicios de back-end utilizando tecnologías web para buscar

o actualizar información. Al igual que las aplicaciones web, la seguridad para aplicaciones clásica necesita ser aplicada a las aplicaciones móviles. La entrada de datos debe ser validada por tamaño, tipología y valores permitidos. El manejo de errores debe proporcionar mensajes de error útiles para los usuarios a la vez que evitar el filtrado de información sensible. No deben volcarse ratreos de pila o diagnósticos del sistema que los hackers pueden aprovechar para penetrar en mayor profundidad. Las pruebas de penetración en aplicaciones son necesarias para asegurarse que los controles de identificación, autenticación y autorización no pueden ser superados. El almacenamiento en los dispositivos debe ser inspeccionado y probado para asegurar que los datos sensibles y las claves de cifrado no se almacenan en texto plano. Los archivos de registro no deben capturar contraseñas u otra información sensible. La configuración SSL debe ser probada.

Los usuarios desean utilizar sus aplicaciones de forma segura y no quieren sorpresas desagradables. Su dispositivo móvil puede exponerlos a mayores vulnerabilidades, por ejemplo, su ubicación podría ser rastreada, potencialmente, utilizando el GPS incorporado, la cámara y el micrófono podrían ser utilizados para capturar información que preferirían mantener privada, y así sucesivamente. Las aplicaciones también pueden ser escritas para acceder a información confidencial, como por ejemplo los contactos, o hacer llamadas telefónicas de manera encubierta y enviar mensajes SMS a números de tarificación adicional.

El desarrollador de la aplicación puede estar preocupado por su reputación, pérdida de ingresos y de propiedad intelectual, mientras que las empresas quieren proteger los datos empresariales a los que los usuarios pueden acceder desde sus dispositivos, posiblemente utilizando tu aplicación. ¿Pueden sus datos mantenerse separados y protegidos de todo aquello que el usuario ha instalado?

Amenazas a Tus Aplicaciones

En algunas plataformas (iOS y Android en particular), desactivar la verificación de firmas es una práctica bastante común.

Necesitas considerar si es relevante o no el que alguien pueda modificar tu código y ejecutarlo en un dispositivo jailbrokeado o ruteado. Un punto obviamente clave sería la eliminación de una verificación de licencia, lo que podría dar lugar a que tu aplicación sea robada y usada de forma gratuita. Menos evidente, pero más grave, es la amenaza de inserción de código malintencionado (o malware) que puede robar información de los usuarios y destruir la reputación de tu marca.

La ingeniería inversa sobre tu aplicación puede dar a hackers el acceso a una gran cantidad de datos sensibles, tales como las claves de cifrado para películas protegidas con DRM, el protocolo secreto para hablar con tu servidor de juegos online, o la manera de acceder al crédito almacenados en el teléfono para tu sistema de pago móvil. Sólo se requiere un hacker y un teléfono jailbrokeado para explotar cualquiera de estas amenazas.

Si tu aplicación maneja dinero real o contenido valioso es necesario tomar todas las medidas posibles para protegerla de ataques Man-At-The-End (o MATE, en el que el atacante tiene acceso físico al dispositivo). Y, si estás implementando un estándar DRM, tendrás que seguir las reglas de robustez que hacen obligatorias las medidas de auto-protección.

Protegiendo Tu Aplicación

Escondiendo el Plano de Tu Código

Algunas plataformas móviles se programan utilizando código administrado (Java o NET.), integrado por los códigos de bytes ejecutados por una máquina virtual, en lugar de hacerlo directamente en la CPU. Los formatos binarios para estas plataformas incluyen metadatos que establecen la jerarquía de clases y dan el nombre y tipo de cada clase, variable, método y parámetro. Los metadatos ayudan a la máquina virtual a implementar algunas de las características del lenguaje (por ejemplo, la reflexión). Sin embargo, los metadatos también son muy útiles para un hacker que esté tratando de realizar ingeniería inversa del código. Hay programas de descompilación, de libre acceso, que regeneran el código fuente a partir de los códigos de bytes, y hacen ingeniería inversa fácilmente.

La plataforma Android tiene la opción de usar la Interfaz Nativa Java (JNI, del inglés Java Native Interface) para acceder a funciones escritas en C y compilar en código nativo. Es mucho más difícil que el código nativo sea objeto de ingeniería inversa en comparación con Java, por lo que se recomienda para cualquier parte de la aplicación donde la seguridad es de suma importancia.

"gcc" es el compilador utilizado normalmente para generar código nativo para Android, mientras que su gemelo "clang" se utiliza para iOS. La configuración predeterminada para estos compiladores prepara cada función para ser exportada desde un objeto compartido, y la agrega a la tabla de símbolos dinámicos en el binario. La tabla dinámica de símbolos es diferente a la tabla de símbolos utilizada para la depuración y es mucho más difícil de dismantelar después de la compilación. El volcado de los símbolos dinámicos puede dar a un hacker un índice muy útil de cada función en el código nativo. El uso correcto del

conmutador del compilador `-f visibility`¹ es una manera fácil de hacer que sea más complicado comprender el código.

El código compilado en Objective-C contiene el código máquina y una gran cantidad de metadatos que pueden proveer a un atacante de información sobre nombres y la estructura de llamada de la aplicación. En la actualidad, existen herramientas y scripts para leer estos metadatos y guiar a los hackers, pero no hay herramientas para ocultarlos. La manera más común de construir una interfaz gráfica para iOS es el uso de Objective-C, pero el método más seguro consiste en minimizar su uso y cambiar a C o C++ para todo lo que no sea la interfaz gráfica de usuario.

Escondiendo el Control de Flujo

Incluso si todos los nombres están ocultos, un buen hacker todavía puede averiguar cómo funciona el software. Hay herramientas comerciales de protección de código administrado que pueden ocultar deliberadamente el camino a través del código recodificando las operaciones y rompiendo los bloques de instrucciones, lo que hace la decompilación mucho más difícil. Con una buena herramienta de protección bien ubicada, un intento de decompilar un binario protegido terminará o bien en un fallo de decompilador o en un código fuente no válido.

Decompilar código nativo es mucho más difícil, pero se puede conseguir. Incluso sin una herramienta, no se necesita mucha práctica para ser capaz de seguir el control de flujo en el código ensamblador generado por un compilador. Las aplicaciones con un fuerte requisito de seguridad necesitarán una herramienta de ofuscación para el código nativo, así como para el código administrado.

¹ gcc.gnu.org/wiki/Visibility

Protegiendo las Comunicaciones de Red

Las comunicaciones de red también son vulnerables, sobre todo cuando las aplicaciones se pueden instalar en emuladores o simuladores, donde los analizadores de red son de libre acceso y pueden monitorear e interceptar el tráfico de red. Considera la posibilidad de proteger las comunicaciones sensibles en red, por ejemplo mediante el uso de SSL para el tráfico HTTP entre tu aplicación y los servidores. Aún así los ataques MATE, especialmente a través de redes WiFi, pueden desvelar datos sensibles.

Protegerse Contra la Manipulación de Datos (Tampering)

Puedes proteger el código base aún más detectando los intentos de manipulación en la aplicación y respondiendo a esos ataques. El código criptográfico siempre debe utilizar algoritmos de cifrado estándar relativamente seguros (por ejemplo, AES, RSA, ECC) pero, ¿qué pasa si un atacante puede encontrar las claves de cifrado en tu archivo binario o en memoria en tiempo de ejecución? Esto podría resultar en que el



atacante tenga acceso a contenido valioso. Incluso si utilizas criptografía de clave pública y sólo se expone una parte del par de claves, todavía tienes que considerar lo que ocurriría si un atacante intercambiara esa clave por la otra mitad que ya conocía. Se necesita una técnica para detectar cuándo el código ha sido manipulado. Hay herramientas disponibles que encriptan/desenscriptan código sobre la marcha, ejecutan las sumas de verificación contra el código para detectar su manipulación, y reaccionan cuando el código ha cambiado.

Las comunicaciones entre la aplicación móvil y los servicios de backend pueden ser monitorizadas y pirateadas. Incluso cuando se utiliza SSL, un proxy web interceptor (como Paros) puede ser configurado con una conexión WiFi que inspeccionará el tráfico SSL. Los atacantes pueden entonces manipular los datos en tránsito, con fines de lucro o entretenimiento. Así que, si se están enviando datos realmente sensibles a través de HTTPS, considera encriptar/desenscriptar datos en la aplicación móvil y en el servidor, de modo que los sniffers de red sólo puedan ver datos cifrados.

Protegiendo Algoritmos Criptográficos

Una herramienta activa contra la manipulación puede ayudar a detectar o prevenir algunos ataques a claves criptográficas, pero no permitirá que las claves permanezcan siempre ocultas. La criptografía de caja blanca tiene por objeto aplicar los algoritmos de cifrado estándar de manera que las claves permanecen ocultas. Algunas versiones de criptografía de caja blanca utilizan complejos enfoques matemáticos para obtener los mismos resultados numéricos de una manera que es difícil de deducir por ingeniería inversa. Otras insertan claves en tablas de consulta y máquinas de estados que es difícil que sean objeto de ingeniería inversa. La criptografía de caja blanca sin duda será necesaria si vas a escribir código DRM o necesitas almacenamiento de datos de alta seguridad.

Mejores Prácticas

No Almacenes Secretos o Información Privada

Minimiza la cantidad de información sensible almacenada en el dispositivo. No almacenes las credenciales o claves de cifrado, a menos que se utilice un almacenamiento seguro protegido por una contraseña compleja. En su lugar, guarda los tokens de autenticación que tienen tiempo de vida y funcionalidad limitados.

Los archivos de registro son útiles para diagnosticar errores en el sistema y hacer un seguimiento del uso de las aplicaciones, pero vigila no violar la privacidad de los usuarios mediante el almacenamiento de información de ubicación, o registrar información personalmente identificable de los mismos. Algunos países tienen leyes que restringen la información de seguimiento que se puede recopilar, así que asegúrate de comprobar las leyes al respecto de los países en los que será utilizada tu aplicación.

No Confíes En El Dispositivo

Cuando diseñas una aplicación, supón que el dispositivo será propiedad de un atacante tratando de abusar de la aplicación. Realiza el mismo ciclo de desarrollo de software seguro en la creación de aplicaciones móviles que emplearías con los servicios de backend. No confíes ni siquiera en las bases de datos que creas para tus aplicaciones móviles, un hacker puede cambiar su esquema. No confíes en el sistema operativo para obtener protección, la mayoría de esas protecciones pueden ser anuladas de manera trivial por un jailbreak al dispositivo. No confíes en que los almacenes nativos de claves mantendrán los datos protegidos, se puede irrumpir en los almacenes de claves por fuerza bruta, a menos que el usuario proteja el dispositivo con una contraseña compleja y de gran longitud.

Minimiza Los Permisos

Android tiene el concepto de permisos, mientras que iOS tiene derechos, que permiten el acceso de las aplicaciones a sensores tales como el GPS y contenido sensible. En Android estos permisos deben ser especificados como parte de la creación de la aplicación, en el archivo `AndroidManifest.xml`. Éstos son mostrados al usuario cuando opta por instalar la aplicación en su dispositivo.

Cada permiso aumenta el potencial de que tu aplicación haga cosas nefastas, y eso puede disuadir a algunos usuarios incluso de descargar la aplicación. Así que intenta minimizar el número de permisos o de características que tu aplicación necesita.

Herramientas

Protección

El renombrado de código básico de Java puede ser realizado mediante Proguard², una herramienta de código abierto, y GuardIT de Arxan³.

Dos comercializadores de herramientas de protección de código gestionado (Java y .NET) son Arxan Technologies⁴ y PreEmptive Solutions⁵.

La mayoría de vendedores de herramientas de protección de código nativo y librerías criptográficas de caja blanca son Arxan e Irdeto⁶.

Las técnicas para proteger Android frente a manipulación de datos están documentadas en androidcracking.blogspot.com.

² www.proguard.sourceforge.net

³ arxan.com

⁴ arxan.com

⁵ preemptive.com

⁶ www.irdeto.com

EnsureIT de Arxan te permite insertar código adicional en tiempo de compilación para detectar depuradores, usar sumas de validación para detectar cambios y permitir al código ser desenscriptado o reparado sobre la marcha.

Sniffing

Una herramienta web proxy estándar es Paros⁷, y una herramienta estándar de sniffing de red disponible en las plataformas habituales es Wireshark⁸.

Decompilado

Véase el decompilador Hex Rays⁹.

Aprende Más

A continuación, algunos recursos y referencias útiles que pueden ayudarte:

- Apple ofrece una guía general de seguridad de software¹⁰. También incluye múltiples enlaces a temas más detallados de su plataforma.
- Hay cursos comerciales de entrenamiento disponibles para iOS y Android¹¹, y el Lancelot Institute¹² ofrece cursos de programación segura que cubren iOS y Android.
- O'Reilly (2011) ha publicado un libro sobre seguridad en Android, Jeff Six: Application Security For The Android Platform. Processes, Permissions and Other Safeguards (Dec

⁷ sourceforge.net/projects/paros

⁸ sourceforge.net/projects/wireshark

⁹ www.hex-rays.com

¹⁰ developer.apple.com/library/mac/navigation/#section=Topics&topic=Security

¹¹ marakana.com/training/android/android_security.html

¹² www.lancelotinstitute.com

2011)¹³ y otro para iOS, Jonathan Zdziarski: Hacking and Securing iOS Applications¹⁴.

- Charlie Miller et al. (2012) publicaron iOS Hackers Handbook¹⁵, que demuestra cómo de fácil es robar código y datos de dispositivos iOS.
- Investigadores académicos han demostrado cuánta información puede ser obtenida de aplicaciones públicas Android en USENIX 2011¹⁶.
- Qualsys Labs¹⁷ ofrece una herramienta de testeo SSL.
- Extensas guías y herramientas de testeo gratuitas de aplicaciones son ofrecidas por OWASP¹⁸, incluyendo el OWASP Mobile Security Project¹⁹.
- AT&T ofrece una herramienta de código abierto de monitorizado de rendimiento de aplicaciones Android llamada AT&T's Application Resource Optimization tool²⁰.

¹³ shop.oreilly.com/product/0636920022596.do

¹⁴ shop.oreilly.com/product/0636920023234.do

¹⁵ www.wiley.com/WileyCDA/WileyTitle/productCd-1118204123.html

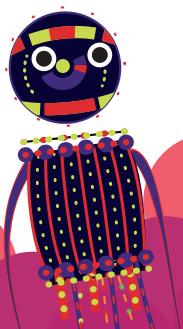
¹⁶ <http://static.usenix.org/event/sec11/tech/slides/enck.pdf> static.usenix.org/event/sec11/tech/slides/enck.pdf

¹⁷ www.ssllabs.com/ssltest/

¹⁸ www.owasp.org

¹⁹ www.owasp.org/index.php/OWASP_Mobile_Security_Project

²⁰ developer.att.com



Conclusiones

Cada vez hay mayor confianza en las aplicaciones móviles, pero están expuestas a mucha gente que quisiera aprovecharse de tal confianza. El nivel adecuado de seguridad de una aplicación es algo que debe considerarse para cada caso en particular. Al final, tu aplicación va a estar ahí fuera, sola ante el peligro, y tendrá que defenderse de los piratas informáticos y otras amenazas maliciosas, donde quiera que vaya.

Invierte tiempo en aprender acerca de las características y capacidades de seguridad de las plataformas móviles que tienes como objetivo. Usa técnicas como el modelado de amenazas para identificar los peligros potenciales relacionados con su aplicación. Realiza revisiones del código y elimina métodos no esenciales de registro y depuración. Considera cómo un hacker podría analizar tu código y, a continuación, utiliza técnicas similares contra la aplicación en un ambiente seguro para descubrir vulnerabilidades y mitigarlas antes de publicar tu aplicación.



Testeando Tu Aplicación

Después de todo ese duro trabajo creando tu aplicación, ¿qué tal probarla antes de publicarla? Probar aplicaciones móviles solía ser una tarea casi enteramente manual; afortunadamente, las pruebas automatizadas son viables actualmente para muchas de las plataformas móviles. Un gran número de las más importantes incluyen la automatización de pruebas en sus herramientas básicas, incluyendo iOS y Android.

Hay disponibles herramientas de automatización de pruebas multiplataforma para las plataformas más populares, algunas son gratuitas y de código abierto mientras que otras son de pago.

En este capítulo se tratan los temas generales, las pruebas para plataformas específicas se explican en sus capítulos correspondientes. Vamos a empezar por cubrir varios conceptos clave que deben tenerse en cuenta antes de que el código sea creado, ya que afectan a la forma en que se escribirá, y que incluyen:

- Testeabilidad
- Desarrollo Guiado por Pruebas
- Pruebas unitarias

Testeabilidad: El Caballo Ganador

Si quieres encontrar la manera de probar tu aplicación eficaz y eficientemente, comienza por diseñar y emplear medidas para testearla; esto se aplica especialmente en el caso de las pruebas automatizadas. Por ejemplo, utilizar técnicas como la inyección de dependencias en el código te permite reemplazar servidores reales (técnica lenta y precaria) con servidores simu-

lados (controlable y rápida). Usa identificadores únicos y claros para los elementos clave de la interfaz de usuario. Si mantienes sin cambios tus identificadores, tus pruebas requerirán menos mantenimiento.

Separa el código en módulos testeables. Hace años, cuando los dispositivos móviles y herramientas de software eran muy limitados, los desarrolladores optaban por "optimizar" su código móvil en monolíticas gotas de código; sin embargo, los actuales dispositivos y plataformas móviles implican que esta forma de "optimización" es innecesaria e incluso contraproducente.

Ofrece maneras de consultar el estado de la aplicación, posiblemente a través de una interfaz de depuración personalizada. De lo contrario, tú o tu equipo de pruebas podríais pasar mucho tiempo tratando de entender cuáles son los problemas que se producen cuando la aplicación no funciona como se esperaba.

Desarrollo Guiado por Pruebas

Hay varias maneras de diseñar e implementar software. El Desarrollo Guiado por Pruebas (TDD, acrónimo del término inglés Test-Driven Development) es un enfoque donde los desarrolladores escriben pruebas automatizadas en paralelo a la escritura del código principal de la aplicación. Las pruebas automatizadas incluyen pruebas unitarias, que se tratarán en la siguiente sección.

TDD es a la vez una actitud y una práctica. Requiere una cierta disciplina para escribir las pruebas, incluso cuando las cosas se ponen difíciles, pero mediante la diligente práctica de TDD, los desarrolladores tienden a escribir código de mejor calidad, más simple y limpio, que también es más fácil de

mantener en el futuro (ya que están protegidos y apoyados por un conjunto de pruebas automatizadas que se pueden ejecutar durante el mantenimiento y la revisión del código fuente de la aplicación).

El enfoque puro se da cuando las pruebas se escriben primero, y se ejecutan, antes de que se escriba nuevo código de la aplicación. Se espera que las nuevas pruebas fallen, es decir, que reporten fallos en el comportamiento de la aplicación. Dichos fallos expresan la discrepancia entre lo que la aplicación necesita hacer y lo que hace actualmente. Ahora, el desarrollador tiene una forma simple, automatizada, para probar sus modificaciones en el código fuente de la aplicación. Una vez que se ha escrito suficiente software para realizar todos los tests a pasar, tenemos la confianza de que la aplicación cumple con los requisitos especificados por esas pruebas.

Si bien es posible que hayan cumplido con los requisitos de negocio, es posible que decidamos que nuestro trabajo no está finalizado todavía. Por ejemplo, puede haber duplicaciones, complejidad innecesaria, y otros defectos conocidos en la implementación. Ahora tenemos la oportunidad de revisar y mejorar el código fuente a través de un proceso conocido como "refactoring". Es en el refactoring donde los desarrolladores mejoran la implementación, y en él las pruebas automatizadas siguen siendo ejecutadas contra el código mejorado.

A veces tenemos que modificar o incluso eliminar pruebas automatizadas cuando los deseados cambios 'rompen' el código existente. El trabajo extra a realizar en los tests para cambiar rápidamente de bases de código puede ser percibido como una carga innecesaria. Los equipos deben decidir y comprometerse a revisar sus pruebas automatizadas en algún momento antes de que terminen sus cambios, de lo contrario muchos de los beneficios a largo plazo de estas pruebas se perderían.

TDD se ha vuelto popular y su uso se ha extendido en las comunidades de desarrollo, especialmente cuando se emplean prácticas de desarrollo Agile.

A pesar de que TDD es toda una batalla cuando se utilizan las herramientas actuales de testeo automático móvil, muchas personas han aportado ejemplos del uso de TDD con éxito, por ejemplo, el libro de Graham Lee Test-Driven iOS Development¹. También puedes considerar usar TDD para los aspectos genéricos de la aplicación.

Pruebas Unitarias

Las pruebas unitarias consisten en escribir pruebas automatizadas que prueban pequeñas porciones de código, por lo general sólo unas pocas líneas de código fuente. El código fuente puede ser la aplicación de un método o función, por ejemplo, que comprueba si un dato tiene el formato correcto o no. Las pruebas unitarias tienen una larga tradición en el desarrollo de software, en el que JUnit² ha generado frameworks similares para prácticamente todos los lenguajes de programación utilizados para desarrollar aplicaciones móviles.

Las pruebas unitarias pueden ser escritas por cualquier persona que pueda escribir software, sin embargo para obtener el máximo beneficio deberían ser escritas por el mismo desarrollador que escribe el código fuente de la aplicación.

Las pruebas unitarias son sólo un aspecto de las pruebas automatizadas, por lo que no son suficientes en sí mismas para

¹ www.informit.com/store/product.aspx?isbn=0321774183

² en.wikipedia.org/wiki/JUnit



acreditar el funcionamiento de una aplicación. Ayudan a los desarrolladores a entender lo que se espera que hagan los bloques individuales del software. Pruebas adicionales, incluyendo otras formas de pruebas automatizadas, pueden ayudar a aumentar la confianza en la aplicación.

Testear a Través de Las Cinco Fases del Ciclo de Vida de una Aplicación

El ciclo de vida completo de una aplicación móvil se define en 5 fases: implementación, verificación, lanzamiento, vinculación y validación, y se testea en cada una de ellas. Algunas de las decisiones tomadas en las primeras etapas pueden afectar nuestras pruebas en etapas posteriores. Por ejemplo, si decidimos que queremos pruebas automatizadas del sistema en la primera fase, serán más fáciles de aplicar en las fases posteriores.

Fase 1: Implementación

Esto incluye diseño, código, pruebas unitarias, y creación de tareas. Tradicionalmente, los probadores no están involucrados en estas tareas; sin embargo, un buen testeo aquí puede mejorar significativamente la calidad y el éxito de la aplicación al ayudarnos a asegurarnos de que nuestra implementación se hace correctamente.

En términos de diseño, deberíamos decidir sobre las siguientes cuestiones:

- ¿Utilizar TDD?
- ¿Escribir pruebas unitarias incluso si no estamos utilizando TDD?
- ¿Tendremos sistemas automatizados de pruebas? En caso afirmativo, ¿cómo facilitaremos dichas pruebas? Por

ejemplo, añadiendo etiquetas adecuadas a objetos clave en la interfaz de usuario.

- ¿Cómo validaremos nuestras aplicaciones? ¿Por ejemplo, a través del uso de analíticas móviles, informes de errores o feedback de usuarios?

Cuestiona el diseño. Queremos asegurarnos de que cumple con los fines previstos, y también evitar cometer errores graves. El documento de Phillip Armour 'Five Orders Of Ignorance'³ es un gran recurso para ayudar a estructurar tu enfoque.

Ten en cuenta también la forma de mejorar la testeabilidad de su aplicación en esta etapa para que puedas hacer tu aplicación más fácil de probar con eficacia y eficiencia. Prácticas tales como pruebas unitarias y TDD se aplican en la fase de implementación. Recuerda probar tu proceso de construcción y los build scripts para asegurarte de que sean eficaces, fiables y eficientes; de lo contrario, es probable que sufras los efectos de un desarrollo deficiente a lo largo de la vida de la aplicación.

Fase 2: Verificación

Esto incluye la revisión de las pruebas unitarias, instalación interna y pruebas de sistema.

Revisa las pruebas unitarias y evalúa su potencia: ¿son realmente útiles y dignas de confianza? Nota: también deben ser revisadas como parte de la fase de implementación, sin embargo éste es un buen momento para abordar las deficiencias materiales antes de que el desarrollo se considere "completo" para la base de código actual.

Para las aplicaciones que requieren instalación, necesitamos maneras de llevarlas a dispositivos específicos para las pruebas de pre-lanzamiento. Para algunas plataformas (incluyendo

³ www.plan.cs.colorado.edu/diwan/3308-07/p17-armour.pdf

Android, iOS y Windows Phone), los teléfonos deben estar configurados para que las aplicaciones se puedan instalar. También tenemos que decidir en qué teléfonos probar la aplicación. Por ejemplo, es aconsejable probar la aplicación en cada versión apropiada de la plataforma móvil. Para iOS esta sólo puede incluir las últimas versiones. En Android es bastante diferente ya que los dispositivos de gama baja se siguen vendiendo con la versión 2.x de Android y nunca se actualizarán a 4.x

También queremos probar diferentes factores de forma de los dispositivos, por ejemplo cuando la relación de las dimensiones de pantalla difiere. Las nuevas dimensiones de pantalla del iPhone 5 han expuesto un montón de bugs de interfaz de usuario. Los desarrolladores de Android son muy conscientes de los muchos problemas que diferentes tamaños de pantalla pueden desencadenar.

Las pruebas del sistema se realizan a menudo de forma interactiva, por los probadores. Considera la posibilidad de evaluar las herramientas de automatización de pruebas y los entornos para algunas de las pruebas del sistema. Entraremos en más detalle más adelante en esta sección.

También queremos considerar cómo sabremos que la aplicación responde bien en:

- Usabilidad, experiencia de usuario y requisitos estéticos
- Rendimiento, particularmente tal y como es percibido por los usuarios finales
- Pruebas de internacionalización y localización

Phase 3: Lanzamiento

Esto incluye pre-publicación y publicación.

Aquellos de vosotros que aún no habéis trabajado con las principales tiendas de aplicaciones, estad preparados para una desafiante experiencia donde la mayoría de los aspectos están fuera de tu control, incluyendo los plazos para la aprobación de tu aplicación. Además, en algunas tiendas de aplicaciones, no es posible revertir a una versión anterior, así que si tu versión actual tiene defectos importantes, tienes que crear una nueva versión que corrija los defectos y, luego, esperar hasta que haya sido aprobada por la tienda de aplicaciones antes de que tus usuarios puedan recibir una versión operativa de la aplicación.

Dadas estas limitaciones es útil ampliar tus pruebas para incluir verificaciones de pre-publicación, por ejemplo respecto a si es adecuada para el conjunto de los dispositivos de destino. Los proveedores de las principales plataformas publican actualmente pautas para ayudarte a verificar que tu aplicación responde a sus criterios de presentación. Estas directrices pueden ayudarte incluso si tu objetivo son otras tiendas de aplicaciones.

Apple	https://developer.apple.com/appstore/resources/approval/guidelines.html (Apple account needed for access)
Android	http://developer.android.com/distribute/googleplay/publish/preparing.html#core-app-quality
Windows Phone	http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh394032(v=vs.105).aspx
BlackBerry	http://developer.blackberry.com/devzone/appworld/tips_for_app_approval.html

Fase 4: Vinculación

Esto incluye búsqueda, confianza, descarga e instalación. Una vez que tu aplicación es de disponibilidad pública, los usuarios necesitan encontrarla, confiar en ella, descargarla e instalarla. Podemos probar cada aspecto de esta fase. Intenta buscar tu aplicación en la tienda de aplicaciones pertinente y en los motores de búsqueda principales. ¿De cuántas maneras diferentes puede ser encontrada por los usuarios de destino? ¿Qué pasa con los usuarios fuera de los grupos objetivo, quieres que la encuentren? ¿Cómo van a confiar los usuarios en tu aplicación lo suficiente como para descargarla y probarla? ¿Tu aplicación realmente necesita tantos permisos? ¿Cómo de grande es la descarga, es práctico descargarla mediante la red móvil? ¿Cabrá en el teléfono del usuario, especialmente si hay poco espacio de almacenamiento disponible en su dispositivo? ¿Se instala correctamente la aplicación? Puede que haya problemas de firmado que causen que sea rechazada por algunos dispositivos.

Fase 5: Validación

Esto incluye el pago, el uso y la retroalimentación. Como seguramente ya sabes, es poco probable que una aplicación móvil con poco feedback tenga éxito. Además, muchas de las aplicaciones tienen una vida muy corta en el teléfono de un usuario. Si la aplicación no es de su agrado ni le involucra en pocos minutos, es probable que sea descartada o ignorada. Y, si estás buscando obtener pagos, vale la pena probar las diferentes formas de pago, sobre todo para aquellos integrados en la aplicación.

Considera encontrar maneras de probar los siguientes puntos prácticos:

- Detección y reporting de problemas. Pueden incluir tu propio código, utilidades de terceros y servicios online.

- Analíticas móviles. ¿Tienen sentido los datos recabados? ¿Qué anomalías hay en los datos reportados? ¿Cuál es la latencia en la recepción de resultados?

Testeo Interactivo

La variedad y el movimiento pueden ayudar a exponer los errores que permanecen latentes cuando se prueba la aplicación sobre un pequeño conjunto de dispositivos en un lugar fijo, como tu lugar de trabajo. Aprende de tus usuarios, ¿cómo usan tu aplicación, o aplicaciones similares? Entonces, diseña pruebas que imiten las maneras que tienen de utilizar aplicaciones y dispositivos.

Vale la pena considerar las directrices en [appqualityalliance](#). org al diseñar casos de prueba. Por ejemplo, se incluyen pruebas de la aplicación para ver lo que sucede cuando se recibe una llamada entrante, y cuando el usuario cambia el teléfono a "modo de vuelo".

Las próximas secciones describen tres aproximaciones diferentes al testeo interactivo.

- **Dispositivos físicos:** por qué las pruebas con teléfonos reales son importantes.
- **Control remoto:** utilizando teléfonos que no están en tus manos físicamente, ya estén a cientos o miles de kilómetros e incluso en otro continente.
- ***Testeo "crowd sourced":** en el que otros probadores realizan tests en tu nombre.

Dispositivos Físicos

A pesar de que los emuladores y simuladores pueden proporcionar pruebas primitivas pero efectivas de tus aplicaciones, e incluso permitir que las pruebas sean totalmente automatizadas en algunos casos, en última instancia tu software necesita funcionar en teléfonos reales, como los utilizados por los usuarios objetivo. Las características de rendimiento de diferentes modelos de teléfono varían enormemente de unos a otros y del dispositivo virtual en tu ordenador, así que debes comprar, mendigar, pedir prestados, varios teléfonos sobre los que probar. Un buen comienzo es elegir una combinación de los populares, los más recientes y los modelos que incluyen características específicas o características como: pantalla táctil, teclado físico, resolución de pantalla, chipset de redes, etcétera. Prueba el software en al menos un dispositivo de gama baja o antiguo si quieres que los usuarios con esos dispositivos también estén contentos.

Algunos ejemplos de áreas a probar en dispositivos físicos:

- **Navegación de la interfaz de usuario:** por ejemplo, ¿pueden los usuarios utilizar la aplicación con una sola mano? Efectos de las diferentes condiciones de iluminación: la experiencia de la interfaz de usuario puede ser diferente bajo luz solar real cuando se está en la calle. Se trata de un dispositivo móvil, la mayoría de usuarios estarán en movimiento. Gira la pantalla y asegúrate de que la aplicación sigue siendo atractiva y funcional.
- **Localización:** si usas información de localización en tu aplicación, muévete, tanto rápido como despacio. Ve a lugares con cobertura de red y GPS irregular y mira cómo se comporta tu aplicación.
- **Multimedia:** el soporte a audio, la reproducción de vídeo y la facilidad de grabación pueden diferir dramáticamente entre dispositivos y sus respectivos emuladores.

- **Conectividad a Internet:** establecer una conexión a Internet puede llevar una cantidad increíble de tiempo. El retardo en la conexión y el ancho de banda dependen de la red, su fuerza y el número de conexiones simultáneas. Prueba los efectos de conectividad intermitente y en cómo la aplicación responde.

Control Remoto

Si no tienes dispositivos físicos a mano o si necesitas probar la aplicación en otras redes, especialmente en el extranjero y diversas localizaciones, a continuación, un "servicio de dispositivos remotos" podría ayudarte. Ellos pueden ayudar a ampliar el alcance y la profundidad de tus pruebas a bajo o ningún coste.

Muchos fabricantes ofrecen este servicio gratuitamente para un subconjunto de sus modelos de teléfono a los desarrolladores de software registrados. Tanto Nokia⁴ (para MeeGo y Symbian) y Samsung⁵ (para Android y Bada) proporcionan acceso restringido, pero diario y gratuito.

También puedes utilizar servicios comerciales de empresas como PerfectoMobile⁶ o DeviceAnywhere⁷ para pruebas similares en una amplia gama de dispositivos y plataformas. Algunos fabricantes promueven estos servicios y crean marca, sin embargo a menudo hay que pagar por ellos después de un corto periodo de prueba. Algunos de los servicios comerciales proporcionan APIs que permiten crear pruebas automatizadas.

Puedes incluso crear un repositorio privado de dispositivos

⁴ apu.ndhub.net/devices

⁵ rtl.innovator.samsungmobile.com/

⁶ www.perfectomobile.com

⁷ www.deviceanywhere.com

remotos, por ejemplo alojándolos en oficinas y ubicaciones remotas.

Ten cuidado con los temas de privacidad y confidencialidad cuando usas estos servicios compartidos.

Testeo “Crowd-Sourced”

Hay miles de millones de usuarios con teléfonos móviles en todo el mundo.

Algunos de ellos son probadores profesionales de software y, de éstos, unos cuantos trabajan para compañías de servicios de pruebas externalizadas como uTest y mob4hire. Ellos pueden probar tu aplicación de forma rápida y relativamente económica, en comparación con el mantenimiento de un amplio equipo especializado en pruebas de software.

Estos servicios pueden mejorar tus otros testeos, por lo que no recomendamos su uso como única prueba formal. Para obtener buenos resultados deberás dedicar parte de tu tiempo y esfuerzo a la definición de las pruebas que quieres ejecutar, trabajar con la empresa para revisar los resultados, etcétera.

Automatización de Pruebas

Las pruebas automatizadas pueden ayudarte a mantener y mejorar tu celeridad, tu velocidad produciendo capacidades, proporcionando feedback temprano de problemas. Para ello, tienen que estar bien diseñadas e implementadas. De lo contrario, corres el riesgo de duplicar la carga de trabajo para mantener un embrollo de pruebas automatizadas averiadas y poco fiables, así como una aplicación en esas mismas condiciones. Las buenas pruebas automatizadas imitan las buenas prácticas de desarrollo de software, por ejemplo utilizando

Patrones de Diseño⁸, modularidad, realización de revisiones de código, etcétera.

Es importante evaluar la longevidad y vitalidad de las herramientas de automatización de pruebas que vayas a utilizar, de lo contrario es posible que se carguen de código de automatización de pruebas no soportado. Las herramientas de automatización de pruebas proporcionadas como parte del SDK de desarrollo son dignas de consideración. Por lo general son gratis, inherentemente disponibles para la plataforma en particular, y soportadas por las grandes empresas.

Automatización de Tests de BDD

BDD es el acrónimo original de Desarrollo Guiado por Comportamiento, en inglés Behavior-Driven Development⁹, en el que el comportamiento es descrito en archivos de texto formateado que pueden ser ejecutados como tests automatizados. El formato de los tests está pensado para ser legible y comprensible por cualquier persona implicada en el proyecto de software. Pueden estar escritos virtualmente en cualquier lenguaje humano, por ejemplo Japonés¹⁰, y utilizan una estructura consistente, simple, con declaraciones tales como Dado, Cuando, Entonces para estructurar los scripts.

Hay varios frameworks BDD disponibles para testear aplicaciones móviles. Incluyen:

- **Calabash para Android e iOS:** <http://github.com/calabash>
- **Frank para iOS:** www.testingwithfrank.com
- **RoboGerk para Android:**
<http://github.com/leandog/RoboGherk>
- **Zucchini para iOS:** www.zucchiniframework.org

⁸ en.wikipedia.org/wiki/Design_Patterns

⁹ en.wikipedia.org/wiki/Behavior-driven_development

¹⁰ github.com/cucumber/cucumber/tree/master/examples/i18n/ja

y varias implementaciones que se integran con Selenium-WebDriver para probar aplicaciones web, incluyendo aquellas en iOS y Android.

A menudo, definiciones de pasos personalizadas (o 'step-definitions' en inglés, que son pequeños scripts que interactúan con la aplicación que es testeada) necesitan ser escritas por alguien con habilidades de programación.

Automatización de Tests de GUI

La automatización de tests de la interfaz gráfica de usuario (o GUI, del término inglés Graphical User Interface) es donde las pruebas automatizadas interactúan con la aplicación a través de la GUI. Es uno de los elixires de la industria de las pruebas, muchos lo han intentado, pero pocos han tenido éxito en la creación de pruebas automatizadas de GUI que para aplicaciones móviles sean útiles y viables. Una de las razones principales por las que la automatización de tests de GUI es tan retadora es que la interfaz de usuario está sujeta a cambios significativos que pueden romper la forma en que las pruebas automatizadas interactúan con la aplicación.

Para que los tests sean efectivos a largo plazo, y conforme cambie la aplicación, los desarrolladores necesitan diseñar, implementar y dar soporte a etiquetas y otros elementos usados por los tests automatizados de GUI. Tanto Apple, con UI Automation¹¹, como más recientemente Android¹² usan la etiqueta 'Accessibility' asignada a elementos de la interfaz de usuario como interfaz de facto para la automatización de la misma.

El año 2012 ha visto muchísimos nuevos contendientes en herramientas y servicios de automatización. Algunas empresas comerciales tienen sus herramientas en código abierto, tales

¹¹ developer.apple.com/library/ios/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/UsingtheAutomationInstrument

¹² developer.android.com/tools/testing/testing_ui.html

como GorillaLogic's MonkeyTalk¹³ y LessPainful's Calabash¹⁴. Estas herramientas se enfocan a proveer de soporte multiplataforma a Android e iOS particularmente. Las empresas cobran por consultoría y otros servicios, mientras que el software es de uso gratuito.

Desafortunadamente, muchos proyectos de código abierto parecen estar parados, incluyendo muchos que hemos mencionado en ediciones anteriores. A otros, entre los que se incluyen Robotium¹⁵ y Frank¹⁶, les va bien y puede que sean incorporados en otras herramientas de automatización de pruebas.

Cientes Sin Cabeza

La interfaz de usuario de una aplicación móvil moderna puede constituir más del 50% del código completo. Si tu testeo se limita a usar la interfaz gráfica de usuario diseñada para usuarios, puedes complicar innecesariamente las pruebas y los esfuerzos de depuración. Un enfoque es crear una interfaz de usuario muy básica que sea un fino envoltorio alrededor del resto del código del núcleo (normalmente esto incluye la conexión en red y las capas de negocio). Este cliente "sin cabeza" puede ayudarte a aislar e identificar rápidamente errores, por ejemplo aquellos relacionados con el dispositivo, la operadora, y otros problemas del entorno.

Otro de los beneficios de la creación de un cliente sin cabeza es que puede ser más sencillo para automatizar algunos de los tests, por ejemplo para probar todas las funciones clave del negocio y/o para automatizar la captura y comunicación de los resultados de las pruebas.

También puedes considerar la creación de programas de

¹³ www.gorillalogic.com/testing-tools/monkeytalk

¹⁴ <https://github.com/calabash> github.com/calabash

¹⁵ code.google.com/p/robotium/

¹⁶ testingwithfrank.com/

esqueleto que "sondan" las características y capacidades básicas a través de una gama de modelos de teléfono, por ejemplo, para que una aplicación J2ME ponga a prueba la gestión de archivos, donde al usuario se le puede pedir (múltiples veces) autorización para permitir las operaciones de archivos (IO). Dada la fragmentación y peculiaridades de las plataformas maduras, tales sondas pueden retornar rápidamente la inversión que realices para crearlas y ejecutarlas.

Cuidado Con Las Especificaciones

Plataformas, redes, dispositivos, e incluso firmware, todos son especificaciones. Cualquiera de ellos puede causar problemas para tus aplicaciones. Pruébalos primero manualmente, suponiendo que tengas el tiempo y presupuesto necesarios para hacerlo rápidamente y obtener feedback temprano.

Muchas aplicaciones móviles incluyen algoritmos y otros elementos no vinculados a una tecnología móvil. Este código genérico debería ser separado del código específico de la plataforma. Por ejemplo, en Android o J2ME la lógica de negocio normalmente puede ser codificada como estándares Java, para después escribir, y ejecutar, pruebas unitarias automáticas en tu entorno de desarrollo estándar utilizando JUnit. Considera pruebas automatizadas específicas de plataforma una vez que el código genérico tenga buenos tests automatizados.



Monetización

Finalmente, has terminado tu aplicación o sitio web para móviles y la has pulido como resultado del feedback de beta testing. Suponiendo que no estás desarrollando como hobby, para la promoción de marcas, etcétera, es el momento de hacer algo de dinero. Pero, ¿cómo se hace eso y cuáles son tus opciones?

En general, tienes las siguientes opciones de monetización:

1. **Pago por descarga:** Vende tu aplicación por descarga
2. **Pago en aplicación:** Añade opciones de pago dentro de tu aplicación
3. **Publicidad móvil:** Gana dinero por publicidad
4. **Participación en ingresos:** Obtén ingresos por servicios del operador que se originen en tu aplicación
5. **Ventas indirectas:** Afiliados, venta de datos y bienes físicos, entre otros
6. **Mercado de componentes:** Vende componentes o una versión marca blanca de tu aplicación a otros desarrolladores

Cuando planifiques tu desarrollo, la determinación de la monetización en tu modelo de negocio debería ser uno de los elementos clave de su diseño inicial, ya que podría afectar al comportamiento técnico y funcional de la aplicación.

Pago Por Descarga

Usando el pago por descarga (PPD, del inglés Pay Per Download), tu aplicación se vende una vez para cada usuario al descargarla e instalarla en su teléfono. El pago puede ser gestionado por una tienda de aplicaciones, operador móvil o mediante otro mecanismo configurado por tí mismo.

Cuando tu aplicación se distribuye en una tienda de aplicaciones, en la mayoría de los casos será ofrecida por el propietario de la plataforma de destino, tales como Apple, Google, BlackBerry, Microsoft o Nokia, y la tienda se encargará del mecanismo de pago por tí. A cambio, la tienda cobra una comisión por todas las ventas (habitualmente el 30%). En la mayoría de los casos, las tiendas ofrecen una matriz de precios fijos para elegir por país y moneda (\$0,99, 0,7 EUR, \$3, etc.) a la hora de fijar el precio de tu aplicación.

La facturación vía operador permite a tus clientes pagar por tu aplicación con sólo confirmar que la venta se cargará a su factura de teléfono móvil o mediante el envío de un SMS Premium. Los SMS Premium siguen siendo muy populares para aplicaciones web móviles, juegos Java, fondos de pantalla y tonos de llamada.

Las APIs de otro operador te permiten incluir en tu aplicación características tales como MMS, devolución de llamada y conferencias multimedia, y obtener ingresos por su uso. Sin embargo, la facturación del operador resulta muy difícil de manejar, sobre todo si quieres vender en varios países, ya que es necesario firmar contratos con un operador en cada país.

En 2011, 57 de los principales operadores de redes de telefonía móvil y fabricantes de móviles del mundo fundaron la Wholesale Applications Community (WAC)¹, una organización sin ánimo de lucro que ayuda a normalizar el ecosistema de

aplicaciones móviles. En 2012, WAC se integró en GSMA. Uno de sus productos es la clave WAC OneAPI, que permite a los desarrolladores interactuar fácilmente con todos los operadores conectados. OneAPI proporciona una interfaz de servidor basada en REST para la facturación de operador y SMS Premium.

Cada operador tendrá una cuota de ingresos típicamente 45% a 65% del precio de venta, pero algunos operadores pueden quedarse hasta con el 95% del precio de venta (y, si los usas, los mediadores también se quedarán con una cuota). Seguridad (cómo evitar la copia de tu aplicación) y gestión son problemas comunes con PPD, pero para algunos dispositivos podría ser la única opción.

A partir de Android 4, Google decidió solicitar los datos de la tarjeta de crédito en el momento de inscripción, algo que Apple ya exige desde el año 2008, lo que según los analistas es el diferenciador clave para una mayor ganancia mensual por aplicación. Además de los clientes que tienen planes de pago por contrato, los clientes de prepago pueden utilizar sus créditos para comprar aplicaciones; como es el caso de la API de pago en aplicación de BlueVia, esta capacidad es particularmente importante para los desarrolladores dedicados a mercados emergentes, donde el número de tarjetas de crédito es bajo.

Vale la pena señalar que la mayoría de los propietarios de tiendas de aplicaciones están llevando a cabo acuerdos de facturación vía operador, teniendo Nokia Store, con mucho, la mejor cobertura con capacidad de facturación vía operador disponible en 46 países. Además, Nokia está aportando su expertise a Microsoft Windows Phone Marketplace para expandir rápidamente su cobertura de facturación vía operador. Google y BlackBerry también están reclutando activamente a los operadores. La razón principal de ésto es que normalmente, cuando los usuarios tienen la opción de comprar con tarjeta de crédito o con métodos de facturación vía operador, muestran

una preferencia significativa hacia la facturación vía operador. Nokia, por lo menos, también aísla a los desarrolladores de las variaciones de la cuota de operador, ofreciendo a los desarrolladores el 70% de los ingresos por facturación.

La última opción es crear tu propio sitio web y poner en práctica un mecanismo de pago a través de él, como PayPal móvil, la iniciativa de los Países Bajos eM! Payment², marcación a números fijos premium³ y otros.

El uso de PPD se puede implementar sin un diseño o requisitos de codificación especiales para tu aplicación. Para comenzar, recomendamos utilizar las opciones de facturación de las tienda de aplicaciones, ya que implican un coste mínimo de instalación y gastos administrativos de menor importancia.

Pagos En Aplicación

El pago dentro de la aplicación es una forma de cobrar por acciones o contenidos específicos dentro de tu aplicación. Un uso muy básico podría permitir la compra unitaria de tu aplicación (sin suscripciones ni cargos recurrentes) después de un período de prueba, que puede generar más ventas que PPD si crees que las características de tu aplicación justifican un precio más alto. Alternativamente, puedes ofrecer las características básicas de tu aplicación de forma gratuita, pero cobrar por contenido premium (vídeos, créditos virtuales, información premium, características adicionales, eliminación de los anuncios y similares). La mayoría de las tiendas de aplicaciones ofrecen una opción de compra en aplicación, o puedes implementar tu propio mecanismo de pago. Si quieres algo

² empayment.com

³ daopay.com

más que una "licencia completa" por pago, tienes que pensar cuidadosamente acerca de cómo, cuándo y qué están dispuestos a pagar tus usuarios, y diseñar tu aplicación en consecuencia.

Este tipo de pago es especialmente popular en los juegos (para funciones como la compra de energía extra, niveles adicionales, créditos virtuales y demás) y puede ayudar a lograr una mayor base instalada ya que puedes ofrecer la aplicación básica de forma gratuita. Ten en cuenta, sin embargo, que algunas tiendas de aplicaciones no permiten implementar opciones de pago vía terceros dentro de tu aplicación. Esto se hace para evitar que se utilice la tienda de aplicaciones para una libre distribución evitando el pago a la tienda de la comisión por ingresos.

También debería ser obvio que necesitarás diseñar y desarrollar tu aplicación de manera que incorpore el método de pago en aplicación. Si la aplicación se ejecuta en varias plataformas, puede ser necesario implementar un mecanismo diferente para cada una de ellas.

Al igual que con PPD, te recomendamos que comiences con el mecanismo de compra en aplicación que ofrece una tienda de aplicaciones, especialmente porque algunos de estos servicios pueden incrementar la facturación vía operador, o con el pago en aplicación ofrecido directamente por los operadores. En Alemania, Deutsche Telekom, Vodafone y Telefónica/O2 se convirtieron en los primeros operadores en lanzar APIs de pago en aplicación que funcionan transversalmente entre operadores y permiten facturar directamente a la cuenta del teléfono del usuario. Desde la perspectiva del usuario, esta es la manera más fácil y más conveniente de pago (uno o dos clics, sin necesidad de introducir números de tarjetas de crédito, nombres de usuario u otras credenciales), por lo que los desarrolladores pueden esperar una gran aceptación por parte del usuario y elevadas tasas de conversión.

Publicidad Móvil

Como es común en los sitios web, puedes decidir ganar dinero mostrando anuncios. Hay una serie de entidades que ofrecen herramientas para mostrar anuncios para móviles, siendo ésta la forma más fácil de ganar dinero en aplicaciones de navegador móvil. Admob.com, Buzzcity.com y inmobi.com son algunas de ellas. Sin embargo, debido a la amplia gama de dispositivos, países y características, en la actualidad hay más de 50 grandes redes de publicidad móvil. Cada red ofrece enfoques ligeramente diferentes y encontrar el que mejor monetice la audiencia de tu aplicación puede no ser tarea simple. No hay una regla de oro, es posible que tengas que experimentar con algunas para encontrar la que mejor te funcione. Sin embargo, para un comienzo rápido es posible considerar el uso de un agregador de publicidad móvil, como Smaato⁴, Madgic⁵ o Inneractive⁶, ya que tienden a dar mejores resultados al combinar y optimizar los anuncios de más de 30 redes móviles de publicidad. La mayoría de las redes de publicidad tienen una participación del 30% al 50% en los ingresos por publicidad y los agregadores de otro 15% o 20% adicional.

Si tu aplicación está dando buenos resultados y tiene un gran volumen en un determinado país, podrías considerar vender publicidad directamente a agencias de publicidad o marcas (publicidad Premium) o contratar a una agencia de medios para hacer eso por ti.

También aquí, muchos de los fabricantes de dispositivos ofrecen servicios de publicidad móvil como parte de las funciones de su tienda de aplicaciones, e igualmente vale la pena explorar estos mecanismos. En algunos casos puede que tengas

⁴ smaato.net

⁵ madgic.com

⁶ inner-active.com

que utilizar los servicios de publicidad del proveedor para poder incluir tu aplicación en su tienda.

La publicidad en aplicación requiere que diseñes y programes tu aplicación con cuidado. No sólo debes considerar con cuidado la ubicación de los anuncios en tu aplicación, también las variaciones y el mecanismo de exclusión (opt-out). Si los anuncios se vuelven demasiado intrusivos, puede que los usuarios abandonen tu aplicación, al tiempo que la publicidad demasiado sutil significará pocos o nulos ingresos.

Puedes necesitar experimentar un poco para encontrar el nivel y las posiciones adecuadas para colocar los anuncios.

Participación en Ingresos

Compartir los ingresos con el operador móvil para servicios integrados en las aplicaciones es una nueva oportunidad para los desarrolladores, y vale la pena seguirle la pista. Este método de monetización permite a los desarrolladores construir servicios como SMS, MMS, localización, publicidad, perfil del cliente y facturación del operador en sus aplicaciones. Con APIs bien documentadas de uso gratuito, los ingresos generados se reparten de forma transparente entre el operador y el propietario de la aplicación.

Aunque BlueVia es actualmente la única comunidad de desarrolladores dedicada a este modelo, si su adopción temprana continúa creciendo puede convertirse en un modelo de negocio reconocido por los operadores móviles.

Indirect Sales

Otra opción es utilizar tu aplicación para dirigir las ventas donde desees.

Habitualmente ofreces tu aplicación o website de manera gratuita y entonces emplearías mecanismos tales como:

1. **Programas de afiliación:** Promover aplicaciones de pago propias o de terceros en el interior de una gratuita. Ver también MobPartner⁷. Esto puede ser considerado una variación de publicidad móvil.
2. **Venta de datos:** Seguimiento del comportamiento y venta de los datos a partes interesadas. Ten en cuenta que, por razones de privacidad, no debe revelar ninguna información personal, y que debes asegurarte de que todos los datos son proporcionados en informes anónimos y consolidados.
3. **Mundo real versus virtual:** Usa tu aplicación como herramienta de marketing para vender bienes en el mundo real. Ejemplos típicos son aplicaciones para vehículos, revistas y marcas tales como McDonald's y Starbucks. También las aplicaciones de cupones utilizan este modelo de negocio.

No hay nada que te impida combinar esta opción con cualquiera de las otras de monetización si lo deseas, pero ten la precaución de no dar la impresión de realizar promociones extremadamente intrusivas.

Mercado de Componentes

Un mercado de componentes o CMP (del inglés Component Marketplace) proporciona otra oportunidad para los desarrolladores de monetizar sus productos mediante la venta a otros

⁷ mobpartner.com

desarrolladores de componentes de software o aplicaciones de marca blanca. Un componente es una pieza de construcción de software que ofrece una funcionalidad definida para ser utilizado por el software de nivel superior.

La típica pregunta que surge en este punto es cómo los CMPs conviven con el código abierto. Como usuario, el código abierto es a menudo gratuito, por lo que el código fuente debe ser proporcionado y los usuarios tienen el derecho de modificar el código fuente y distribuir la obra derivada.

Algunos proveedores de componentes requieren una cuota de licencia. Ellos pueden proporcionar el código fuente completo que permite al desarrollador depurar código en el nivel más bajo. Algunos CMP soportan todos los modelos: componentes de pago así como componentes libres, ambos con o sin el código fuente.

Si eres un desarrollador en busca de un componente, los CMP ofrecen dos ventajas principales: En primer lugar, no tienes que abrir su código fuente sólo porque uses componentes de software. Todo código abierto viene con una licencia. Algunas licencias como la Apache están comercialmente compatibles, mientras que otras, como AGPL y OSL, requieren que abras el código fuente tuyo que se integrará con el suyo. Es posible que no quieras esto. En segundo lugar, los CMPs proporcionan una forma fácil de encontrar y descargar componentes. Puedes pasar días mirando los repositorios de código abierto para encontrar el componente adecuado.

Los mercados de componentes han existido durante décadas. El mercado más importante es de componentes para .NET y Visual Basic en la comunidad y Windows. Mercados como componentOne y proveedores como Infragistics son bien conocidos.



En cambio, la idea de los mercados de componentes dentro del ámbito móvil es bastante nueva. Recientemente, Developer Garden y Verious se introdujeron en este campo⁸. Además, la idea de abrir el mercado a los proveedores semi-profesionales es nueva. En Developer Garden puede registrarte y ofrecer tu componente al público de forma gratuita o como un producto de pago.

Ahora imagina que has construido una pieza de código que podría ser visto como uno o incluso varios componentes. Los siguientes pasos incluirían proveer de una documentación extensa y detallada y, a continuación, podrías agregar también varios ejemplos "hola mundo" utilizando el componente. Combínalo en código fuente, o precompilado, con la documentación y los ejemplos, y ya está listo para ser ofertado en el mercado de componentes. Debido a que tus clientes probablemente te harán preguntas (a mejor documentación, menos preguntas recibirás), a menudo también tiene sentido ofrecer una FAQ.

8 www.developergarden.com/component-marketplace/



Eligiendo tu Modelo de Monetización

Así que, con todas estas opciones, ¿cuál debería ser tu estrategia? Depende de tus objetivos, veamos algunos:

- ¿Quieres una gran base de usuarios? Considera distribuir tu aplicación gratuitamente en primer término, y posteriormente añadir publicidad móvil o dividirla en versiones gratuitas y de pago, cuando tengas más de cien mil usuarios a nivel mundial.
- ¿Estás seguro de que los usuarios desearán comprar tu aplicación inmediatamente? Entonces véndela como PPD a 0.89€, pero ten en cuenta que, aunque puede que ingreses varios cientos de euros por día, fácilmente pueden no ser más que un par de cientos por semana si la valoración de tu aplicación es poco apropiada o la competencia es feroz.
- ¿Estás ofreciendo capacidades premium a precio premium? Considera una versión limitada en tiempo o capacidades y usa la compra en aplicación para permitir la compra de una versión completa, ya sea permanentemente o por un período de tiempo.
- ¿Estás desarrollando un juego? Considera ofrecerlo de manera gratuita con publicidad en la aplicación, o una versión básica que desbloquee mediante compra en la aplicación nuevas capacidades, niveles, vehículos o cualquier equipamiento de juego (freemium).
- ¿Es tu aplicación móvil una extensión de una tienda web o física tuya? Ofrece la aplicación gratis y gana ingresos de los productos y servicios que comercializas en el mundo real.

El estudio Developer Economics 2012⁹ identificó los modelos de monetización más lucrativos preguntando a más de 1.500 desarrolladores sus experiencias y estrategias preferidas. La siguiente tabla presenta el porcentaje de desarrolladores que utilizan cada modelo, así como el promedio de ingresos mensual por aplicación para cada uno.

Modelo de ingresos	Porcentaje (%) de desarrolladores usando el modelo	Beneficio medio mensual por aplicación
Subscripción	12%	\$3.683
Compras en aplicación	19%	\$3.033
Pago por descarga	34%	\$2.451
Freemium	18%	\$1.865
Publicidad	33%	\$1.498

Tiendas de aplicaciones

La otra cara de la generación de ingresos es el marketing y la publicidad. La necesidad puede ser obvia si vendes tu aplicación a través de tu propio sitio web, pero resulta igualmente importante cuando se usa la tienda de aplicaciones de un vendedor. Esas tiendas son a la vez la maldición y la bendición de los desarrolladores móviles. En el lado positivo, dan a los desarrolladores largo alcance y exposición potencial de ventas que de otra manera sería muy difícil de lograr. En el lado negativo, las más populares contienen actualmente cientos de miles de aplicaciones, disminuyendo el potencial para destacar entre la multitud y tener éxito, llevando a muchos a comparar

la posibilidad de éxito en una tienda de aplicaciones a la probabilidad de ganar la lotería.

Dicho esto, aquí están algunos consejos y trucos para ayudarte a incrementar tus probabilidades.

Estrategias Básicas para Llegar Alto

Lo más importante a comprender acerca de las tiendas de aplicaciones es que son canales de distribución y máquinas de marketing. Esto significa que aunque son una gran manera de conseguir introducir tu aplicación en los dispositivos de los usuarios, no van a promocionar tu aplicación por tí (a menos que compres posicionamiento premium, ya sea a través de banners o posiciones de lista). No puedes confiar en las tiendas de aplicaciones para incrementar tus descargas, a menos que aparezcas en una lista top-ten. Pero no hay que jugar a la lotería con tus aplicaciones, ten una estrategia y un plan para comercializar tu aplicación.

Hemos preguntado a muchos desarrolladores acerca de las tácticas que les han proporcionado la mayor atención y un mejor ranking en las tiendas de aplicaciones.

Nos dieron muchas respuestas y salió a la luz un tema común: no hay ninguna fórmula mágica, ¡tienes que atacar por todos los frentes! Sin embargo, te ayudará tratar de mantener en cuenta lo siguiente:

- Necesitas una aplicación rompedora: debería ser entretenida, fácil de usar y estar libre de bugs. Asegúrate de ponerla en las manos de usuarios antes de en las tiendas.
- Pule tus iconos e imágenes para la tienda de aplicaciones, trabaja en la descripción de tu aplicación, y elige cuidadosamente palabras clave y categoría. Si estás inseguro o insatisfecho con los resultados, experimenta.
- Obtener reseñas de bloggers y revistas es una de las me-

jores maneras de recibir atención. Como contraprestación, algunos te pedirán dinero, otros exclusividad y otros acceso anticipado a tus aplicaciones.

- Consigue críticas positivas tan pronto como sea posible. Llama a tus amigos y pide a tus usuarios de manera regular una reseña.
- Si vas a llevar a cabo cualquier tipo de publicidad, hazlo en avalancha durante un par de días. Esto es mucho más efectivo que gastar la misma cantidad de dinero durante dos semanas, ya que te ayudará a alcanzar un elevado pico de audiencia, más que una subida lenta y gradual.
- No confíes en el tráfico generado por personas que exploran la tienda de aplicaciones, cerciórte de conducir tráfico a tu aplicación a través de tu sitio web, SEO y social media.

Tiendas Múltiples versus Tienda Única

Con más de 120 tiendas de aplicaciones disponible para los desarrolladores, evidentemente hay muchas opciones para distribuir una aplicación. Pero los 20 minutos necesarios en promedio para presentar una aplicación en una tienda significa que podrías pasar mucho tiempo publicando aplicaciones en tiendas lóbregas que consiguen pocas descargas. Esta es la razón por la que la mayoría de los desarrolladores se adhieren a sólo una o dos tiendas, perdiendo una oportunidad potencialmente enorme, ¡pero consiguiendo mucho más tiempo para cosas importantes, como la programación! Así que, ¿debes ir a múltiples tiendas o no?

Múltiples tiendas	Tienda única
Las tiendas de aplicaciones de las principales plataformas pueden tener serias limitaciones, tales como mecanismos de pago, penetración en determinados países y directrices de contenido.	Más del 90% de los usuarios sólo utilizan una única tienda de aplicaciones, que suele ser la que viene por defecto incluida con el teléfono.
Las tiendas más pequeñas te dan más opciones de visibilidad (aplicación destacada).	Tu propio sitio web te puede aportar más tráfico que las tiendas (especialmente si tienes una marca conocida).
Las tiendas más pequeñas tienen un entorno social más amistoso que las grandes.	Muchas tiendas pequeñas arañan datos de las grandes, así que puede que tu aplicación también esté en ellas.
Las tiendas de los operadores tienen directrices para contenidos notoriamente estrictas y puede ser difícil entrar, particularmente para algunos tipos de aplicaciones.	Para contenidos sin nicho, la tienda del operador o plataforma pueden ofrecer suficiente exposición para no justificar el esfuerzo extra de una estrategia multi-tienda.
Las tiendas más pequeñas ofrecen un amplio rango de opciones de pago y modelos de negocio, o están disponibles en muchos países.	Algunas tiendas de operadores tienen procesos de facturación más fáciles, tales como facturación directa a la cuenta de teléfono del usuario, conduciendo a un incremento en la ratio de conversión.
Algunos desarrolladores reportan que el 50% de sus beneficios en Android vienen de fuera de Android Market.	Los desarrolladores para iPhone sólo necesitan una tienda.

¿Qué Puedes Ganar?

Una de las preguntas más comunes de un desarrollador es acerca de cuánto dinero se puede ganar con una aplicación móvil. Está claro que algunas aplicaciones han hecho millonarios a sus desarrolladores, mientras que otros no renunciarán a su puesto de trabajo en el corto plazo. De acuerdo con una investigación del 2012 por App-Promo.com¹⁰, el 59% de todos los desarrolladores de aplicaciones no están generando suficientes ingresos para pagar los costes de desarrollo, y el 80% confirmó que los ingresos generados por su aplicación de más éxito no resularon suficientes para apoyar un negocio independiente.

En última instancia, lo que puedes ganar depende de satisfacer una necesidad y de un marketing eficaz. La experiencia sugiere que las aplicaciones que ahorran dinero o tiempo al usuario son más atractivas (descuentos, cupones de hotel, música gratuita y similares), seguidas por los juegos (basta con ver el éxito de Angry Birds) y las herramientas de negocio (los visores de documentos de oficina, herramientas de sincronización, herramientas de copia de seguridad y otras), pero a menudo el éxito (económico) de una sola aplicación no se puede predecir. El éxito, por lo general, viene con un bagaje en experimentación y mucha perseverancia.

Cuando se trata de plataformas, en realidad son BlackBerry e iOS las que encabezan la tabla de ingresos mensuales por aplicación, de acuerdo a la investigación Developer Economics de VisionMobile. Android es tercero por detrás de ellos, con Windows Phone en la retaguardia. A continuación, se presenta el promedio de ingresos mensuales por aplicación para cada plataforma de acuerdo a la investigación antes mencionada.

¹⁰ app-promo.com/press-release-app-developers-get-a-wake-up-call-from-results-of-app-promos-first-annual-developer-survey/

Plataforma	Ingresos al mês por aplicación (USD)
Android	\$2.735
BlackBerry	\$3.853
iOS	\$3.693
Windows Phone	\$1.234

Aprende Más

Si quieres ahondar más en el tema de marketing de aplicaciones, lee la "Mobile Developer's Guide To The Parallel Universe" publicada por WIP. La tercera edición acaba de ser publicada y está disponible tanto en sus eventos como en su sitio web¹¹.

¹¹ wipconnector.com



Epílogo

Gracias por leer esta 12ª edición de nuestra Guía del Desarrollador Móvil. Esperamos que hayas disfrutado de su lectura y que te hayamos ayudado a clarificar tus opciones. Tal vez ahora estás listo para participar en el desarrollo de una aplicación móvil. Esperamos que sí. Por favor, involúcrate también con la comunidad y comparte tus experiencias e ideas con nosotros y el resto de compañeros.

Si quieres contribuir a esta guía o patrocinar próximas ediciones, por favor envía tu feedback a **developers@enough.de**.

Si estás usando Twitter, te invitamos a seguirnos en **twitter.com/enoughsoftware** y pasar la voz acerca de este proyecto utilizando el hashtag **#mdgg**

Finalmente, para esta edición, puedes descargar esta publicación en PDF desde nuestro website:
www.enough.de/products/mobile-developers-guide.

También puedes pedirla impresa a través de esa página.
¿Y qué tal enviar el enlace a amigos y colegas también?

Acerca de los Autores

Mostafa Akbari / bitstars

Mo ha trabajado en ingeniería de software e investigación de la interacción humana en los últimos años. Está involucrado en proyectos de movilidad verde. Actualmente, Mo ha arrancado una spin-off con Simón Heinen en la Universidad RWTH Aachen para la investigación y el desarrollo de la realidad aumentada. Se centra en las interacciones personales de RA con datos basados en la localización y en visión por ordenador. Su pasión por los juegos de realidad mixta tiene origen en su afición por los juegos de mesa y el geocaching.

 [@mosworld](https://twitter.com/mosworld) www.bitstars.com

Anna Alfut

Anna comenzó su vida profesional como diseñadora creativa. Después de descubrir su pasión por el diseño de interfaces, se convirtió en la co-autora de una aplicación para iOS y Android y consultora en múltiples proyectos, tanto en el lado agencia como en el de cliente. Actualmente trabaja desde casa como diseñadora de interfaces y UX para productos de consumo en dispositivos móviles y de sobremesa. Aparte de dibujar y pensar a través de pantallas también hace ilustraciones y disfruta de vivir en Londres.

 www.alfutka.net

Andrej Balaz / Enough Software

Como graduado de la Universidad de Bellas Artes de Bremen, Andrej se centra en interfaces de usuario, UX, diseño visual para aplicaciones móviles y otras tecnologías interactivas. También es el encargado de la distribución y el diseño de esta guía. Cuando no está involucrado en algo móvil, le encanta experimentar con el arte digital y la ilustración.

 [@abiozzUI](https://twitter.com/abiozzUI) www.enough.de

Davoc Bradley / Redware

Davoc ha estado trabajando como ingeniero de software desde 1999, especializándose en arquitectura y diseño de webs y sistemas móviles de alta usabilidad. Más recientemente, en Redware, se puso a los mandos de la arquitectura, el diseño y el desarrollo de la galardonada plataforma de gestión de aplicaciones móviles Redsource. Davoc también es un músico entusiasta, fan ávido del cricket y le encanta viajar.

 [@davocbradley](https://twitter.com/davocbradley) www.redware.co.uk

Richard Bloor / Sherpa Consulting Ltd

Richard ha estado escribiendo sobre el desarrollo de aplicaciones móviles desde el año 2000. Ha contribuido a sitios web populares, como AllAboutSymbian.com, pero ahora se centra en ayudar a las empresas en la creación de recursos para desarrolladores. Richard aporta una sólida formación técnica a su trabajo, habiendo administrado el desarrollo y testeo de una serie de proyectos TIC importantes, entre ellos el de Land Information NZ, integrando información sobre la propiedad del territorio y un sistema de inspección. Cuando Richard no está escribiendo sobre desarrollo móvil, se le puede encontrar regenerando el monte autóctono de su propiedad al norte de Wellington, Nueva Zelanda.

Eva Casado de Amezua / Universitat Oberta de Catalunya

Eva es diseñadora, desarrolladora y formadora en multimedia, así como la traductora al castellano de esta edición de la guía. Además de trabajar con plataformas web en GDSK, tutoriza proyectos de final de grado y enseña teoría de los nuevos medios en la UOC. Eva colabora asimismo con otras actividades e iniciativas, entre las que se encuentra Zentralized.com. Actualmente centra su atención en aplicaciones y webs móviles, y la fotografía creativa.

 **@ellaing** **www.ellaing.com**

Dean Churchill / AT&T

Dean trabaja en el diseño de seguridad, desarrollo y prueba de aplicaciones de AT&T. Desde hace años está centrado en impulsar los requisitos de seguridad en las aplicaciones móviles, tanto para aquellas de consumo como para internas de AT&T. También ha apoyado las líneas emergentes de producto de AT&T en Salud Móvil y Vida Digital. Vive en el área de Seattle y disfruta haciendo esquí alpino y pesca con mosca.

Julian Harty / Commerctest

Julian fue contratado por Google en 2006 como su primer ingeniero de pruebas fuera de EE.UU. responsable de probar las aplicaciones móviles de Google. Ayudó a otros, tanto dentro como fuera de Google, a aprender a hacer lo mismo, y terminó escribiendo el primer libro sobre ese tema. Posteriormente trabajó para eBay, donde su misión era reformar las pruebas a nivel mundial. Actualmente trabaja de forma independiente, escribe aplicaciones móviles y herramientas de automatización de pruebas, y ayuda a otros a mejorar sus aplicaciones móviles. También está escribiendo un nuevo libro sobre testeo y automatización de pruebas para aplicaciones móviles.

 **@julianharty**

Bob Heubel / Immersion Corp.

Bob es un evangelista de la tecnología háptica en Immersion Corporation, especializado en ayudar a los desarrolladores a implementar lo que se conoce como efectos de retroalimentación por fuerza, táctil o con estruendo. Ha pasado más de diez años trabajando con desarrolladores, operadores, OEMs y ODMs para diseñar y poner en práctica estas sensaciones cuya finalidad es mejorar los juegos y las experiencias de interacción de usuario. Bob se graduó en la Universidad de California en Berkeley el año 1989 con una licenciatura en Literatura Inglesa.

 **@bobheubel** **www.immersion.com**

Ovidiu Iliescu / Enough Software

Tras desarrollar aplicaciones de sobremesa y web durante varios años, Ovidiu decidió que el software móvil era más de su agrado. Está involucrado en desarrollos Java ME y BlackBerry para Enough Software desde 2009. Se emociona con cualquier cosa relacionada con una programación eficaz, algoritmos y gráficos por ordenador.

 **@ovvyblabla** **www.enough.de** **www.oviduiiliescu.com**

Gary Johnson / Sharkfist Software, LLC

Gary está actualmente activo en desarrollo móvil en todos los ámbitos, Windows Phone, iOS y Android. Su experiencia previa incluye una gran conocimiento de .NET, Silverlight y WPF. Tiene una fuerte pasión por todas lo móvil, así como por la creación de excelentes interfaces de usuario y UX.

Alex Jonsson / MoSync

A Alex le gusta todo lo móvil, tanto aplicaciones como tecnologías web, así como conectar inteligentemente objetos físicos a otros digitales. Tiene un Doctorado en Ciencias de la Computación y sigue dando clases de vez en cuando. Alex tiene una necesidad ecléctica de crear nuevos valores mediante la búsqueda de nuevas combinaciones de cosas, transfiriendo conocimientos entre disciplinas, y explotando aspectos de la comunicación y los medios con el objetivo de unir a las personas. Alex es Director de Tecnología de MoSync Inc.

 [@dr_alexj](https://twitter.com/dr_alexj) www.mosync.com

Matos Kapetanakis / VisionMobile

Como Director de Marketing de VisionMobile, sus actividades incluyen la gestión del sitio web y el blog, así como dar con los conceptos y marcoms para las ilustraciones e infografías publicadas por la compañía. Matos también es el director del proyecto de la línea de investigación Developer Economics, así como de otros proyectos de investigación sobre desarrollo.

www.visionmobile.com

Michael Koch / Enough Software

Michael viene desarrollando software desde 1988, uniéndose al equipo de desarrollo de Enough Software en 2005. Ocupa el cargo de Director de Tecnología. Ha dirigido numerosos proyectos de desarrollo de aplicaciones móviles (principalmente para Java ME, Windows Mobile y BlackBerry), y también es un experto en tecnologías de servidor. Michael es un entusiasta del código abierto involucrado en muchos proyectos libres, como GNU classpath.

 [@linux_penguin](https://twitter.com/linux_penguin) www.enough.de

Daniel Kranz / Sevenval

Daniel es un estratega multicanal con experiencia en consultoría, agencias y tecnología. Anteriormente director de proyecto en una de las principales agencias de publicidad, ahora trabaja como consultor de soluciones móviles aconsejando a marcas sobre cómo integrar lo móvil como parte de su estrategia digital global.

www.sevenval.com www.fitml.com

Javier Melenchón / Universitat Oberta de Catalunya

Javier Melenchón se dedica a realizar actividades formativas asociadas a la creación multimedia desde el ámbito universitario, inicialmente en la Universitat Ramon Llull y actualmente en la Universitat Oberta de Catalunya. Forma parte de la primera promoción de titulados universitarios en ingeniería multimedia y se doctoró en ese mismo ámbito. Es experto en todo tipo de tratamiento de señales de audio, imagen y vídeo y en cómo formar a los profesionales para que sepan manejar este tipo de información, de crucial importancia en el ámbito móvil actual. Colabora con la revista digital Mosaic.

LinkedIn: melenchon mosaic.uoc.edu multimedia.uoc.edu

Tim Messerschmidt / PayPal

Tim ha estado desarrollando aplicaciones Android desde 2008. Después de estudiar informática de negocios, se unió a Neofo nie Mobile con sede en Berlín como desarrollador de software móvil en 2011, siendo consultor para Samsung Alemania como Developer Advocate para Android y bada desde 2010. En 2012 se trasladó a PayPal como un Developer Evangelist. Es un apasionado de los pagos por móvil, las interfaces de usuario, UX y el desarrollo en Android en general. Además, le encanta hablar en conferencias, escribir artículos y participar en todo tipo de social media.

 **@seraandroid & @PayPalEuroDev**
www.timmesserschmidt.de

Gary Readfern-Gray / RNIB

Gary es un especialista en accesibilidad que trabaja para el Royal National Institute of the Blind. Situado en la Unidad de Innovación, tiene pasión por el espacio móvil y, en particular, en el desarrollo de aplicaciones accesibles a través de una amplia gama de plataformas mediante la colaboración con las comunidades de desarrolladores.

www.rnib.org.uk

Alexander Repty

Alexander ha estado desarrollando software para Mac OS X desde 2004. Cuando el iPhone SDK fue lanzado en 2008, fue uno de los primeros desarrolladores registrados en el programa. Como empleado de Enough Software trabajó en una serie de aplicaciones, una de las cuales fue destacada en un anuncio de Apple TV. Ha escrito una serie de artículos sobre desarrollo en iPhone. En abril de 2011, comenzó su propio negocio como desarrollador de software independiente y contratista.

 **@arepty** **www.alexrepty.com**

Marcus Ross

Marcus es un desarrollador y formador independiente. Después de 10 años de estar empleado en varias empresas, ahora está haciendo proyectos SQL-BI y todo tipo de actividades móviles multiplataforma. Es un autor regular en la revista alemana "mobileWebDeveloper". En su tiempo libre se le ve a menudo en conferencias, hablando sobre temas móviles y JavaScript. También escribe artículos, libros y tweets sobre desarrollo móvil.

 [@zahlenhelfer](https://twitter.com/zahlenhelfer) www.zahlenhelfer-consulting.de

Thibaut Rouffineau / WIP

Constructor de comunidades y pasión con vertiente móvil, durante los últimos 5 años Thibaut ha estado trabajando para impulsar la comunidad de desarrolladores móviles hacia una mayor apertura e intercambio. Thibaut es Vicepresidente de Developer Partnerships en WIP, la entidad organizadora de Droidcon Londres. Es un orador entusiasta sobre temas móviles y ha realizado comunicaciones por todo el mundo.

 [@sthibautr](https://twitter.com/sthibautr) www.wipconnector.com

Michel Shuqair / AppValley

Tras comenzar con aplicaciones WAP en blanco y negro, iMode y juegos SMS en la década de los 90, Michel pasó a liderar la red social m.wauwee.com. Sirviendo a casi un millón de miembros, Michel fue apoyado por un equipo de especialistas en Symbian, iPhone, BlackBerry y Android en su sede en Amsterdam. m.wauwee.com fue adquirida posteriormente por MobiLuck.
www.appvalley.nl

Shailen Sobhee / Enough Software

Shailen es un reciente graduado en Ingeniería Electrónica y Ciencias de la Computación por la Universidad Jacobs de Bremen. Actualmente se encuentra cursando un Máster en Ciencias e Ingeniería de la Computación en la Universidad Técnica de Munich. Shailen se unió a Enough Software en julio de 2012 en prácticas y desde entonces ha estado desarrollando software Android de vanguardia. Con una formación informática fuerte, Shailen encuentra interés en nuevas tecnologías que potencialmente pueden tener un impacto muy positivo en el futuro. Actualmente está desarrollando e investigando en nuevas soluciones basadas en tecnologías NFC.

www.enough.de

Marco Tabor / Enough Software

Marco es el responsable de relaciones públicas, ventas y mucho más en Enough Software. Es coordinador de este proyecto, teniendo además la responsabilidad de encontrar patrocinadores y fusionar las aportaciones hechas por la comunidad móvil.

 **[@enoughmarco](https://twitter.com/enoughmarco)** **www.enough.de**

Ian Thain / SAP

Ian es un Mobile Evangelist de SAP, aunque comenzó hace 12 años con Sybase Inc. Se dirige regularmente a público de todo el mundo, proporcionando conocimiento móvil y experiencia corporativa. También escribe artículos, blogs y tweets en movilidad empresarial y es un apasionado de la experiencia en desarrollo móvil en el mundo empresarial.

 **[@ithain](https://twitter.com/ithain)** **ianthain.ulitzer.com** **www.sap.com**

Robert Virkus / Enough Software

Robert ha estado trabajando en el espacio móvil desde 1998. Experimentó la fragmentación de Java de primera mano cuando desarrolló y portó un cliente móvil en el Siemens SL42i, el primera teléfono para el mercado de masas con una máquina virtual Java. Después de esta experiencia puso en marcha el proyecto de código abierto J2ME Polish en 2004. J2ME Polish ayuda a los desarrolladores a superar la fragmentación por dispositivo. Es el fundador y CEO de Enough Software, la compañía detrás de J2ME Polish, muchas aplicaciones móviles, y este libro.

 [@robert_virkus](https://twitter.com/robert_virkus) www.enough.de www.j2mepolish.org

una iniciativa de:



patrocinadores:



**UNA REVISIÓN NO COMERCIAL DE LAS
TECNOLOGÍAS MÓVILES PARA DESARROLLADORES
Y TOMADORES DE DECISIONES, REALIZADA POR
LA PROPIA COMUNIDAD.**

Daniel Hudson, www.webtechman.com

¡Un trabajo espectacular! Te sorprenderá lo increíblemente rápido que puedes introducirte en el mercado móvil con los sencillos pasos explicados en esta guía.

Monika Lischke, Community Manager, Intel AppUp developer program

Unos contenidos extremadamente útiles, también para los no desarrolladores. ¡Y con un diseño fantástico!