Revista Internacional de Investigación e Innovación Tecnológica

Página principal: www.riiit.com.mx

Diseño de aplicaciones móviles: retos y soluciones actuales

Mobile Applications: Current Challenges and Solutions

Gallardo-López, L.[a], Monroy-Rodríguez, G.[b]

[a] Departamento de Sistemas, Universidad Autónoma Metropolitana, Av. San Pablo No.180, Reynosa Tamaulipas, Azcapotzalco, 02420, CDMX.
[b] Departamento de Computación, CINVESTAV del IPN, Av. Instituto Politécnico Nacional 2508, San Pedro Zacatenco, Gustavo A. Madero, 07360, CDMX.
glizbeth@azc.uam.mx; guillermo.monroy@cinvestav.mx

Innovación tecnológica: Mejoramiento del diseño de aplicaciones móviles.

Área de aplicación industrial: cómputo móvil, cómputo en la nube, entre otras.

Enviado: 30 abril 2020.
Aceptado: 10 agosto 2020.

Abstract

In the creation of the Apps, we should remark on the design stage because software engineers use their knowledge and creativity to suggest a model that solves the problem. The engineers must consider the requirements of the target users, the features of mobile devices and the infrastructure of communication. The creative process implies a set of challenges to solve. In this paper, we will provide a classification given by five challenge-solutions set that the novel developer must observe in the creation Apps process. The methodology used in this analysis consists in three stages: setting of challenge-solutions, search and selection of documents and, lastly, validation of the set of challenges-solutions obtained. The results of our analysis show the problem-solution patterns, i.e. the development of Apps gets to grow enough in a way that while the building paradigm of mobile devices does not change, the improvements in their features will not impact significantly the creation of the Apps. Another important finding is that the novel developer can take the resulting compendium of challenge-solutions and make it his guide to achieve the creation of a quality App.

Keywords: Apps development, challenges and solutions, mobile computing, cloud computing, Software process.

Resumen

En el desarrollo de las aplicaciones móviles, también llamadas Apps, destacamos la etapa de diseño. En esta etapa, los ingenieros de software emplean su conocimiento y su creatividad para proponer un modelo de solución. Estos ingenieros deben considerar los requerimientos de un conjunto de usuarios, las características del dispositivo móvil y la infraestructura de comunicación. Este proceso creativo involucra una serie de retos a resolver. El objetivo de este artículo es proporcionar una clasificación de cinco retos-soluciones que un desarrollador novel debe tener en cuenta en la creación de una App. La metodología empleada en este análisis consiste en tres etapas: planteamiento de retos-soluciones, búsqueda y selección de documentos, y validación de los retos-soluciones. Como resultado de este análisis observamos un patrón donde para cada problema existe al menos una solución. Nosotros consideramos que el desarrollo de Apps y por lo tanto su diseño está alcanzando un nivel de madurez tal, que mientras no se presente un cambio de paradigma en la construcción de dispositivos móviles (Hardware), los retos y las soluciones propuestos hasta este momento, tendrán vigencia. Con el resultado de este análisis, un desarrollador novel puede tomar como guía el compendio presentado para lograr una App de calidad.

Palabras clave: Desarrollo de las Apps, retos y soluciones, cómputo móvil, cómputo en la nube, proceso del *software*.

1. Introducción

Los ingenieros de software crean Apps, empleando un proceso de desarrollo del software. Algunas etapas de este proceso son: análisis de requerimientos, diseño de una solución, implementación de la solución, pruebas sobre la implementación, por mencionar algunas. En el desarrollo de las Apps se destaca la etapa de diseño porque los ingenieros de software emplean su conocimiento y su creatividad para proponer un modelo de solución que considere los requerimientos de un conjunto de usuarios, las características del dispositivo móvil y las características de otros dispositivos, tales como estaciones de trabajo, que intervendrán para que la App funcione correctamente. El objetivo del artículo es proporcionar una clasificación de cinco retos que un desarrollador novel debe tener en cuenta en la creación de una App. Los objetivos específicos son: proporcionar un compendio de retos-soluciones que lo guíen para obtener una App de calidad, e identificar para cada reto-solución uno o varios objetivos cualitativos dirigidos hacia buenas prácticas de diseño e implementación. El término "App de calidad" lo definimos como aquella App que resuelve los retos relacionados con la clasificación siguiente: capacidades del dispositivo móvil, heterogeneidad de plataformas móviles, seguridad sobre la información del usuario, conexión a la red de Internet, usabilidad de las Apps.

Algunas de las soluciones que se han propuesto, son: esquemas de adaptación de contenidos de la interfaz gráfica al tamaño y resolución de la pantalla del dispositivo, herramientas para crear Apps que funcionen en diferentes plataformas, empleo de algoritmos de cifrado para proteger la información, mecanismos de sincronización entre un dispositivo móvil y un servidor remoto, y mecanismos de evaluación de la usabilidad de las Apps. Como resultado de este análisis observamos patrones; entendiendo patrón como una dupla compuesta por un problema y su solución.

El documento está organizado de la manera siguiente: la sección 2. Metodología describe la metodología para la elaboración del compendio de retos y soluciones; la sección 3. Marco Conceptual presenta un breve marco

teórico que pone en contexto el desarrollo de las Apps; la sección 4. Desarrollo de las Apps describe el proceso de desarrollo de las Apps, poniendo énfasis en la etapa de diseño; la sección 5. Clasificación de las Apps presenta una clasificación de las Apps a partir de su arquitectura; la sección 6. Retos en el diseño de Apps describe los retos implicados en el desarrollo de las Apps; la sección 7. Soluciones en el diseño de *Apps* describe algunas soluciones a los retos; finalmente, la sección 8. Resultados y discusión presenta los resultados y su discusión, y la sección 9. Conclusiones y perspectivas presenta las conclusiones y perspectivas sobre el desarrollo de las Apps.

2. Metodología

En el desarrollo de software confluyen dos perspectivas diferentes, la académica y la profesional. Esta visión se muestra en la *Figura 1*. La académica incluye el desarrollo de proyectos de investigación y la formación de recursos humanos calificados. La profesional incluye el desarrollo de proyectos tecnológicos y desarrollo de innovación, producto de la investigación. Posteriormente, identificamos dos perfiles, uno académico que busca aplicar los resultados de la investigación al ámbito profesional, y otro profesional que busca retroalimentar al ámbito académico. Ambos perfiles se encuentran a través de colaboraciones en grupos interdisciplinarios; lectura de reportes técnicos, artículos científicos, documentación de tecnologías, entre otros documentos; reuniones académico-profesionales, congresos, coloquios, seminarios, entre otros. Lo anterior para enfrentar y resolver los retos relacionados con la creación de las Apps, por mencionar el caso que nos interesa en este trabajo.



Figura 1. Ámbitos involucrados en el desarrollo de Apps.

Por un lado, uno de los autores -de este artículo- tiene un perfil académico, cuenta con más de quince años de estudio sobre el ciclo de desarrollo de software, que incluye la dirección de proyectos tecnológicos y de investigación. Estos proyectos concluyen con la implementación y prueba de aplicaciones que resuelven problemas específicos. Por otro lado, otro de los autores tiene un perfil profesional, cuenta con más de ocho años en el desarrollo de múltiples sistemas incluyendo aplicaciones móviles; además ha dirigido equipos de innovación tecnológica en la iniciativa privada. Por lo tanto, consideramos que la experiencia de los dos perfiles de los autores, además de complementar sus visiones sobre el desarrollo de Apps, es suficiente para realizar un análisis sobre sus retos-soluciones.

La metodología empleada en este estudio se muestra en el diagrama de la Figura 2 y a continuación se describe cada una de sus etapas.

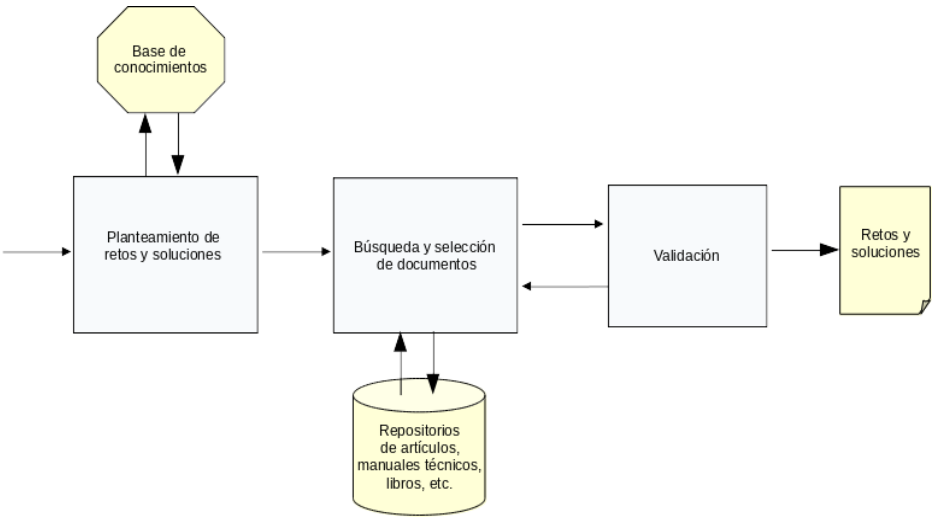


Figura 2. Metodología para la definición de retos y soluciones.

Planteamiento de retos y soluciones. Se realizó un mapa mental con los retos y soluciones que los perfiles hemos identificado a partir de nuestro conocimiento (Base de conocimientos) producto de la experiencia. A partir del mapa mental se generó la versión preliminar del compendio de retos-soluciones en el desarrollo de Apps.

Búsqueda y selección de documentos. Con el propósito de no solo incluir la experiencia de los perfiles, se realizó una búsqueda de artículos, manuales técnicos y libros para validar o descartar los retos-soluciones del compendio. Se emplearon motores de búsqueda tales como: *web of science*, *IEEE Xplore*, *ACM Digital Library*. Los manuales técnicos se obtuvieron directamente de los portales oficiales. Los libros se obtuvieron de la biblioteca digital de la Universidad Autónoma Metropolitana. Los tipos de artículos coleccionados fueron: panorámicas (*surveys*), estados del arte (*state of the art*) y revisiones sistemáticas (*systematic reviews*). Además, se consultaron los portales de las principales plataformas de Apps, tales como: *Google*, *Amazon* y *Apple*.

Validación. A partir del análisis de los documentos, algunos retos-soluciones fueron validados, ampliados o descartados. Como producto de la validación se obtuvo el compendio final de retos-soluciones del desarrollo de Apps.

Las etapas de **búsqueda y selección de documentos** y **validación** se realizaron iterativamente, un total de tres veces.

3. Marco Conceptual

En esta sección se presentan los principales conceptos que ponen en contexto el desarrollo de las Apps, concretamente: aplicación móvil, dispositivo móvil, cómputo móvil y cómputo en la nube.

3.1. Aplicación móvil

Una App es un software aplicativo (*software application*) destinado a un conjunto de usuarios que generalmente no es experto en computación. Un software aplicativo está formado por un conjunto de programas computacionales que tienen como propósito apoyar a un usuario en alguna de sus actividades, a través de un conjunto de funcionalidades. El conjunto de programas computacionales está acompañado de una serie de documentos entre los que destaca el

manual de uso; en particular, las Apps no incluyen un manual de uso porque, en principio, se construyen con las características de usabilidad siguientes: fácil de comprender y fácil de recordar. El conjunto de programas se instala en computadoras, y luego es ejecutado sobre un Sistema Operativo, por ejemplo: iOS, Windows, Linux, UNIX, entre otros. Es conveniente distinguir entre el software aplicativo que opera exclusivamente en la computadora donde es instalado, a éste se le conoce como standalone, y el software aplicativo que opera, inclusive, "en la nube" (cloud), permitiendo que varios usuarios lo usen de manera remota, a través de redes como la de Internet. Algunos tipos de software aplicativo están destinados a apoyar actividades lúdicas, médicas, educativas, bancarias, oficina, entre otras [1][2][3][4]. Las Apps operan en dispositivos móviles, tal y como lo muestra la Figura 3, el software aplicativo y en particular las Apps relacionan dos paradigmas actuales: el cómputo en la nube y el cómputo móvil.

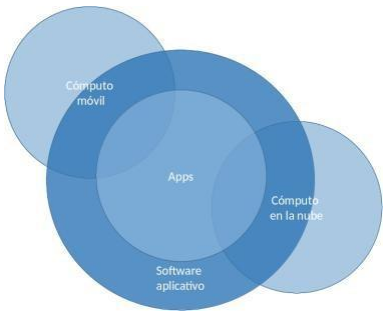


Figura 3. Paradigmas asociados a las Apps.

3.2. Dispositivo móvil

Dispositivo móvil (*mobile*) se define como "equipo electrónico que opera sin cables y cuyo propósito es la comunicación, procesamiento e intercambio de datos. Este tipo de dispositivos pueden ser llevados por el usuario para recibir, enviar o transmitir información en cualquier lugar y en cualquier momento gracias a su movilidad y portabilidad" [1]. Algunos ejemplos de dispositivos móviles son: Navegadores *GPS*, *Personal Device Assistant*, *Smartphone*, *Tablet PC* y *laptop*.

3.3. Cómputo móvil

Hablamos de cómputo móvil (Mobile Computing) cuando, empleando los dispositivos móviles se realiza una comunicación con un servidor centralizado y estático, empleando un canal inalámbrico; dicho dispositivo se convierte en parte del ambiente de cómputo distribuido en una red de computadoras. En consecuencia, los usuarios de dispositivos móviles pueden acceder a la información localizada en una red estática mientras se viaja [5][6][7].

El cómputo móvil permite un tipo de interacción humano-computadora para transmitir voz, video y datos. Los principios del cómputo móvil son:
a) portabilidad, llevar el dispositivo a cualquier lugar; b) ubicuidad, acceder a recursos en cualquier momento y en cualquier lugar; c) conectividad, permanecer conectado a una red con un mínimo de desconexiones; d) interacción social, permitir la colaboración con otros usuarios, a través de transacciones de datos; e) interacción individual, capacidad para acceder a aplicaciones de software que apoyen al usuario en sus actividades cotidianas.

3.4. Cómputo en la nube

Cómputo en la nube (*Cloud computing*) es un paradigma para emplear recursos de software que son puestos a disposición como servicios en la red de Internet. El cómputo en la nube provee poder de cómputo, almacenamiento en base de datos, aplicaciones y otros recursos de tecnologías de la información. La plataforma de servicios está formada por una infraestructura distribuida y en-línea. Algunas de las ventajas que proporciona el

cómputo en la nube son: alta disponibilidad, redundancia, capacidad de concurrencia y escalabilidad [8].

4. Desarrollo de las Apps

La creación de las Apps, es el resultado de un proceso de desarrollo del software formado por varias etapas, algunas de ellas son: análisis de requerimientos, diseño, implementación y pruebas. En la etapa de análisis de requerimientos se especifica el propósito de la App, el alcance de sus funcionalidades y las características de los usuarios que la emplearán. En la etapa de diseño se crea uno o varios modelos de solución para las funcionalidades especificadas en el análisis de requerimientos. Al modelo de solución se le conoce como arquitectura. En la etapa de implementación se hace una traducción del modelo de solución a código, empleando algún lenguaje de programación; el código materializa las funcionalidades. En la etapa de pruebas se inspecciona tanto la arquitectura como el código, y se ejecuta el código empleando casos de prueba diseñados previamente. La corrección de los errores, revelados por las pruebas, permite asegurar que la App cumpla con los requerimientos iniciales, y que sus funcionalidades operen correctamente. De los modelos propuestos para el desarrollo del software, destacan los métodos ágiles para el desarrollo de Apps, tal como *XP-eXtreme Programming*. Los métodos ágiles mantienen la estrategia de construir de forma iterativa e incremental, donde varias etapas del proceso se realizan de forma repetitiva hasta obtener una funcionalidad (un incremento); se realizan tantos incrementos como funcionalidades se hayan especificado para la App, ver Figura 4. Además, en XP los casos de prueba se diseñan antes de la etapa de implementación.

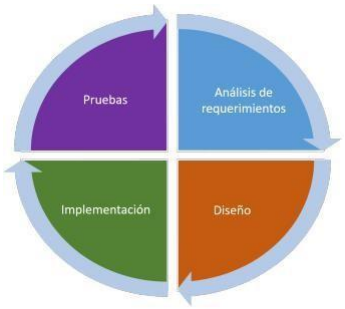


Figura 4. Etapas del desarrollo de una App.

4.1. Arquitectura del software

El diseño de una App es la etapa donde la creatividad de los desarrolladores de software es fundamental porque del modelo de solución para las funcionalidades, que ellos propongan, dependerá la satisfacción del usuario. El software aplicativo, como ya lo mencionamos, está formado por un conjunto articulado de componentes (programas computacionales) que dan origen a las funcionalidades. La estructura y la manera de articular los componentes definen la arquitectura del software. Una de las arquitecturas más usadas en el software aplicativo es la llamada Cliente-Servidor. En esta arquitectura, las tareas se reparten entre componentes de software que proveen recursos o servicios (servidores), y componentes de software que hacen peticiones de los recursos o servicios (clientes). Esta arquitectura también opera sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario y distribuido a través de una red de computadoras [9][10]. Un criterio importante durante el diseño de las Apps es la usabilidad, porque se buscan características como: efectividad, eficiencia y satisfacción.

4.2. Usabilidad de una App

Usabilidad es “la medida en la que un producto puede ser usado por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en un contexto de uso específico” [1]. Para

productos de software, las características se definen de la manera siguiente: Efectividad es la exactitud e integridad con la que los usuarios alcanzan sus objetivos correctamente al usar un software. Por ejemplo, son atributos de efectividad en un software la facilidad de aprendizaje, la ausencia de errores y la facilidad para ser recordado. Eficiencia es la relación entre la exactitud y el grado en el que los usuarios alcanzan sus objetivos de forma completa, y los recursos que se han empleado para lograr esos objetivos. Por ejemplo, son atributos de eficiencia, el tiempo empleado para llevar a cabo las tareas y el tiempo de aprendizaje. Satisfacción se refiere a la comodidad que siente el usuario y al grado en el que demuestra actitudes positivas con el uso de un software.

5. Clasificación de las Apps

Al definir la arquitectura de una App se toman decisiones sobre la conveniencia de emplear únicamente los recursos computacionales (hardware y software) provistos por el dispositivo móvil, o emplear recursos computacionales externos, conectados a una red local, y remotos, conectados a través de la red de Internet. A continuación, se presenta una clasificación de las Apps a partir del empleo de recursos computacionales locales, externos y remotos, a saber: *Native Apps, Based-Ad-Hoc Apps, Web Apps e Hybrid Apps*.

5.1. Native Apps

Se instalan en el dispositivo móvil y no requieren de conexión a la red de Internet para realizar su función principal. Generalmente, este tipo de *Apps* emplean los sensores del dispositivo, por ejemplo: la cámara, el *GPS*, el giroscopio, el acelerómetro, entre otros. Ejemplos de estas *Apps* son: la cámara, la calculadora, los contactos, la agenda, el reloj, el reproductor de video, entre otros. Algunas Apps proponen funcionalidades secundarias (opcionales) que requieren conexión a Internet para solicitar algún servicio; por ejemplo, un gestor de fotos con almacenamiento en la nube.

5.2. Based-Ad-Hoc Apps

Las Apps se instalan en el dispositivo móvil y emplean redes *Ad-Hoc* para comunicarse con otros dispositivos, de forma inalámbrica, siempre y cuando se encuentren en el radio de alcance del medio de conexión *(bluetooth*: tiene un alcance promedio de 100 metros a la redonda). Algunos ejemplos de Apps que emplean redes *Ad-Hoc* son: reproductor y transmisores de audio y video caseros, videojuegos multijugador, monitores de bebés, monitores de actividades deportivas, redes de sensores automotrices, entre otros.

5.3. Web Apps

Son aplicaciones que se ejecutan en servidores *Web (hardware y software)* y son desplegadas en los dispositivos móviles. Proveen una interfaz de comunicación para acceder a su funcionalidad, a través de un navegador Web. Un usuario provisto de algún dispositivo móvil o PC, el cual esté conectado a la red de Internet o a una Intranet podrá emplear una *Web App*. Algunos ejemplos son: MS-*Office online*, *Spotify, Amazon store [11] y Netflix.*

5.4. Hybrid Apps

Las Apps híbridas emplean por lo menos dos de los esquemas antes listados. Estas aplicaciones pueden emplear los servicios Web o de redes *Ad-Hoc* para intercambiar datos a través de protocolos definidos de acuerdo con el tipo de servicio: *HTTP, HTTPS, FTP, SSH, SOAP, SMTP, POP*, entre otros. El cliente solicita uno o varios servicios desde una App instalada en un dispositivo

móvil, o bien desde un navegador. Para emplear los servicios, el dispositivo debe estar conectado a la red de Internet o una red Ad-Hoc. Algunos ejemplos de estas aplicaciones son: *Facebook, Twiter, Whatsapp, Waze*, entre otros.

6. Retos en el diseño de Apps

Algunos retos por considerar durante el diseño de Apps serán descritos en las secciones siguientes: Capacidades del dispositivo móvil, Heterogeneidad de plataformas móviles, Seguridad sobre la información del usuario, Conexión a la red de Internet y Usabilidad de las Apps.

6.1. Capacidades del dispositivo móvil

Las capacidades de los dispositivos móviles junto con los requerimientos determinarán la arquitectura de una *App*. Las capacidades del dispositivo están determinadas por las características siguientes: 1. Tamaño y resolución de la pantalla, 2. Consumo de energía eléctrica, 3. Uso de la unidad de procesamiento central y 4. Uso del espacio de almacenamiento.

6.1.1. Tamaño y resolución de la pantalla

El tamaño y resolución de la pantalla es uno de los principales retos, porque cada fabricante propone diferentes configuraciones. El tamaño en píxeles (Ancho x Alto) varía entre 768x1024 y 1200x1920. El tamaño de la pantalla medido en pulgadas corresponde a su diagonal, y varía entre 7 y 12,9. La resolución es la cantidad de píxeles por pulgada, y varía entre 132x300. Además del tamaño y resolución de la pantalla, se debe considerar las posiciones del dispositivo *portrait* y *landscape*. Es costoso, en cuanto a recursos humanos y por ende recursos económicos, diseñar e implementar una interfaz gráfica para diferentes tamaños y resoluciones de pantalla.

6.1.2. Consumo de energía eléctrica

Los dispositivos móviles son alimentados eléctricamente con una batería. Algunos tipos de batería empleados actualmente son: iones de litio Li-ion y polímeros de litio Li-Po. La batería tiene que alimentar el procesador, el procesador de gráficos, a los dispositivos de entrada (audífonos, memoria externa, entre otros) y a los distintos sensores que forman parte del dispositivo. A continuación, se presenta en una lista ordenada, de mayor a menor, a los principales consumidores de energía en un *smartphone*: 1) llamadas telefónicas, 2) pantalla encendida, 3) *Wi-fi* durante la transmisión/recepción, 4) empleo del *GPS* y 5) empleo del procesador.

6.1.3. Uso de la unidad de procesamiento central

Algunos dispositivos móviles tienen más de una unidad de procesamiento central (*CPU-Central Processing Unit*); a esto se le conoce como multiprocesamiento. Un circuito integrado (*chip*) con varias *CPU* se le denomina procesador multi-núcleo; destacan los *dual-core* y *quad-core*. Algunos también cuentan con una unidad de procesamiento gráfico (*Graphics Processing Unit - GPU*). La *GPU* es un coprocesador dedicado al procesamiento de gráficos y operaciones de coma flotante. Con el *GPU* se aligera la carga de la *CPU* durante la ejecución de videojuegos o aplicaciones *3D* interactivas. Un reto durante el diseño, sobre todo de videojuegos, es emplear de manera efectiva los recursos.

6.1.4. Uso del espacio de memoria

Un dispositivo móvil tiene tres tipos de memoria: *RAM-Random Access Memory*, memoria interna y memoria externa; la tabla 1 muestra una comparación, en cuanto sus tamaños y sus velocidades de transmisión.

RAM, el reto principal tiene que ver con su tamaño, el cual determinará junto con el tipo

de procesador, la capacidad del dispositivo para soportar la ejecución, en segundo plano, de varias tareas (por ejemplo, *Apps*). Por lo tanto, la *RAM* debe corresponder con el tipo de procesador. Un procesador potente podría tener cuellos de botella con una *RAM* de poca capacidad de almacenamiento y de baja velocidad de acceso a los datos. La *RAM* de dispositivos de alta gama va de 4GB a 8GB y con tecnologías relacionadas a la velocidad *DDR3* (*Double Data Rate 3th Generation*) con una velocidad de transmisión de hasta 160MB/s y *DDR4* con velocidades de hasta 350Mb/s [12].

Memoria interna, el reto principal tiene que ver con su tamaño, el cual determinará el número de Apps que podrán ser instaladas en el dispositivo móvil, así como la cantidad de datos que podrá almacenar. El tamaño de la memoria interna varía de los 32GB y hasta 256GB, en dispositivos de alta gama. La velocidad que puede alcanzar es de 350Mb/s en la lectura y 150Mb/s en la escritura.

Memoria externa, el reto principal tiene que ver con la velocidad de acceso a los datos. El acceso a los datos, en la memoria externa, es más lento (velocidad máxima de 80MB/s para la lectura y 90MB/s para la escritura) comparado con la memoria interna (velocidad máxima 350MB/s en la lectura y 150MB/s en la escritura). Aunque la memoria externa cuenta con mayor espacio de almacenamiento (hasta 2TB) que la memoria interna (hasta 256GB) [12][13]. En la **Tabla *1*.** Tipos de memoria en dispositivos móviles. se muestra un resumen.

Tabla 1. *Tipos de memoria en dispositivos móviles.*

Memoria	Tamaño	Velocidad
RAM	4GB a 8GB	160MB/s a 350MB/s
Interna	32GB a 256GB	350MB/s y 150MB/s
Externa	64GB a 2TB	80MB/s y 90MB/s

6.2. Heterogeneidad de plataformas móviles

Existen diferentes plataformas para dispositivos móviles en el mercado, a saber: *Android, iOS, Windows Mobile, Amazon Kindle OS, Ubuntu Touch y BlackBerry OS*, entre otros. La heterogeneidad de plataformas impacta en el diseño e implementación de *Native Apps*. Por ejemplo, la forma de acceder a los sensores y la forma de acceder a otras Apps es distinta en cada plataforma. Por si esto fuera poco, cada plataforma emplea tecnologías diferentes e incompatibles para el desarrollo. Por ejemplo, para Android se emplea un lenguaje de programación basado en *Java* y *Android SDK* (*Software Development Kit*); mientras que para *iOS* se emplea el lenguaje *Objective-C y Swift* [14] [15].

6.3. Seguridad sobre la información del usuario

Todo dispositivo móvil conectado a la red de Internet está expuesto a ataques provenientes de programas maliciosos (*malwares*). Estos programas pueden alterar el funcionamiento de las *Apps* o los datos que manipulan. Las consecuencias son nefastas, los *malwares* pueden comprometer los datos del usuario y dar lugar a crímenes como el robo de información o el robo de identidad, así como el espionaje [16]. Los desarrolladores de *Apps* deben proteger del espionaje la información del usuario y el acceso a los recursos del dispositivo móvil. El espionaje sobre la información del usuario incluye la recopilación de sus documentos y datos de búsqueda en páginas web. El espionaje sobre los recursos del dispositivo incluye la activación del *GPS* para conocer la ubicación geográfica del usuario, la activación del micrófono para escuchar sus conversaciones, y la activación de la cámara para videograbarlo.

6.4. Conexión a la red de Internet

Los dispositivos móviles deben acceder a la red de Internet para considerarse como tales, este acceso se realiza a través de dos vías: la red celular y el *Wi-Fi*; sin embargo, no siempre es posible acceder a la red de *Internet*; por lo tanto, las *Apps* basadas en *Web* y las híbridas deben crear mecanismos que soporten las funcionalidades de las *Apps offline* (fuera de línea) así como esquemas de sincronización de datos entre el almacenamiento local y el remoto [17][18][19].

6.5. Usabilidad de las Apps

La usabilidad es un factor cualitativo del software aplicativo, el cual debe ser considerado por los desarrolladores durante el proceso de desarrollo para que las Apps logren satisfacer a los usuarios. Las *Apps* cuyo propósito es el mismo, compiten por el gusto de los usuarios. Por lo tanto, la usabilidad es fundamental porque la percepción del usuario sobre las funcionalidades, la interfaz gráfica y la forma de interacción provistas por la *App* serán determinantes para que él la adopte o no [6].

7. Soluciones en el diseño de *Apps*

Algunas soluciones a los retos presentes en el diseño de las Apps serán descritas en las secciones siguientes: Capacidades del dispositivo móvil, Heterogeneidad de plataformas móviles, Seguridad sobre la información del usuario, Conexión a la red de Internet y Usabilidad de las Apps. Las tablas 2 y 3 muestran una síntesis de las soluciones.

7.1. Capacidades del dispositivo móvil

Retomando las características del dispositivo móvil: 1. tamaño y resolución de la pantalla, 2. consumo de energía eléctrica, 3. uso de la unidad de procesamiento central y 4. uso del espacio de almacenamiento.

7.1.1. Tamaño y resolución de la pantalla

Son dos las propuestas que describiremos a continuación, *Responsive Design* y *Adaptative Web Design*.

Responsive Design propuesto por Ethan Marcotte [20] para el diseño de aplicaciones Web. Consiste en que el contenido responda a los cambios en el tamaño del (browser), ajustando la posición y el tamaño de los elementos, y habilitando o deshabilitando una barra de navegación.

Adaptative Web Design propuesto por Aaron Gustafson [21] para el diseño de aplicaciones Web, pero también empleado en Native Apps. En tiempo de ejecución, la App seleccionará un layout acorde con el tamaño de la pantalla del dispositivo. A diferencia del esquema anterior, si se intenta redimensionar el tamaño del browser no habrá cambio en la posición de los elementos. Generalmente se realizan seis diseños para seis de los tamaños de pantalla típicos: 320, 480, 760, 960, 1200, y 1600 pixeles.

7.1.2. Consumo de energía eléctrica

Encontramos cuatro propuestas de solución: Disminuir el empleo de la pantalla, emplear esquemas de color adecuados, emplear la *CPU* el menor tiempo posible, y disminuir las conexiones a la red de Internet.

Disminuir el empleo de la pantalla Consiste en desarrollar *Apps* con capacidad multimodal; es decir, que alternen su modo de operación, de visual a auditivo. Por ejemplo, la *App Waze* apaga la pantalla y cambia a su modo de operación auditivo; las notificaciones se realizan a través de sonidos e instrucciones vocales para el conductor del automóvil.

Emplear esquemas de color adecuados Las pantallas de algunos dispositivos móviles

emplean la tecnología *OLED-Organic Light-Emitting Diode*. Esta tecnología mejora el consumo de energía eléctrica con respecto a las pantallas de generaciones anteriores. Sin embargo, el empleo de colores sigue siendo un reto en el desarrollo de las Apps. Los colores claros, como por ejemplo el blanco, requieren más energía que los colores oscuros, como por ejemplo el negro. Chameleon [22][23] propone invertir los colores; es decir, aquello que esté en color claro cambia a color oscuro y viceversa. Esta propuesta es sencilla y práctica, sobre todo porque muchas *Apps* tienen un fondo claro y textos en color oscuro. Otra propuesta consiste en crear esquemas de color alternativos y ahorradores de energía para las Apps. Chameleon [22] propone que el mismo browser encuentre y aplique, para la App que está por desplegar, un esquema de color adecuado. Ding Li et al. [23] proponen reescribir de manera automática el código *html* de la *App* para modificar el esquema de color.

Emplear la *CPU* el menor tiempo posible para emplear la *CPU* el menor tiempo posible, es necesario evaluar la conveniencia de emplear una arquitectura *Native, Web o Hybrid*. Por ejemplo, si el volumen de datos que empleará la App es pequeño y el cálculo es mínimo, el empleo de la *CPU* del dispositivo móvil será lo mejor; por lo tanto, una arquitectura *Native* es adecuada. Pero si el volumen de datos a emplear es grande y se requiere realizar un cálculo considerable, es preferible recurrir al cómputo en la nube; por lo tanto, una arquitectura *Web* o *Hybrid* serán las más adecuadas. Otra manera de reducir el tiempo de empleo de la *CPU* consiste en mejorar u optimizar los algoritmos que darán origen a las funcionalidades de la App. Existen diferentes tipos de patrones de solución, entre otros: patrones de diseño de software [24], patrones de arquitectura [25] [26], patrones de integración [27] [28]. Por ejemplo, en una *Hybrid App* cuya arquitectura es de tipo *Thin-client*, el grueso del código se encuentra en un servidor; en este caso conviene evaluar el empleo de técnicas como programación concurrente, con el propósito de que algunas tareas, involucradas en las funcionalidades, se ejecuten, en el servidor, de forma paralela.

Disminuir las conexiones a la red de *Internet.* En las *Apps* basadas en *Web*, se emplea la técnica de memoria *cach*e que consiste reservar un espacio de la memoria interna o externa del dispositivo móvil para almacenar la información que de manera recurrente está consultando el usuario. Al implementar la técnica de memoria *cache* se reduce sustancialmente el número de conexiones al servicio Web. Los resultados de la investigación de Park et al. [29] revelan un ahorro del 45% de la batería del dispositivo móvil.

7.1.3. Uso de los procesadores CPU y GPU

Para mejorar la efectividad de los procesadores se promueve entre los desarrolladores de software el empleo de mejores prácticas de programación y el empleo del cómputo en la nube (*offloading*). Empleo de mejores prácticas de programación específicamente mencionaremos mejores prácticas durante las etapas de diseño e implementación. En estas etapas es importante que los desarrolladores tengan amplio conocimiento sobre el paradigma de programación, y que empleen, siempre que sea posible, patrones para solucionar el problema; por ejemplo, patrones de diseño de software, patrones de arquitectura, patrones de integración y patrones de diseño de interfaces gráficas, actualmente disponibles en la literatura, inclusive para tecnologías específicas [30].

Empleo del cómputo en la nube también conocido como cómputo (*offloading*), en donde el dispositivo móvil se convierte en un *thin-client* que solicita la ejecución de una

tarea a un servicio Web o a un servicio en la nube. La solicitud se hace de forma asíncrona para no bloquear a la plataforma. Cuando el servicio *Web* responde, la *Native App* continúa con el procesamiento de su flujo programático [31].

7.1.4. Uso del espacio de memoria

Se promueve entre los desarrolladores el evaluar el empleo de memoria interna o externa para la *App* que se está diseñando, y el empleo de la técnica de memoria *cache*.

Memoria interna o externa para el diseño de una *App* que requiera el empleo de bases de datos o manipulación de multimedios, a saber: música, fotos y vídeo, de manera local, se deberá evaluar y decidir sobre el empleo de la memoria interna o de la externa, a partir de la velocidad de acceso y de la capacidad de almacenamiento de la memoria interna contra la externa.

Empleo de memoria *cache* Una forma de mejorar la efectividad de la *RAM* es el uso de la memoria *cache*. La información se almacena temporalmente con el propósito de reducir el procesamiento y las peticiones a servicios locales o remotos [32] [33].

7.2. Heterogeneidad de plataformas móviles

Existen *frameworks* para construir, al mismo tiempo, una *Native App* para distintas plataformas. Estas *frameworks* son capaces de traducir los componentes que son comunes a todas las plataformas de su propio meta-lenguaje a un código intermedio; luego, el *framework* traduce el código del meta-lenguaje al lenguaje nativo de la plataforma deseada. Posterior a esto, es posible refinar el código (*refactoring*), generado por el *framework* para optimizar el uso de recursos en tareas específicas. Es importante mencionar que los componentes que son exclusivos de cada plataforma deberán ser implementados en la tecnología correspondiente. Finalmente, para cada App se obtendrá su paquete de instalación nativo; por ejemplo .apk o .dmg. Algunas de estas *frameworks* son: *Xamarin* [34], *Apache Cordova* [35], *Titanium Appcelerator SDK* [36], por mencionar solo algunos.

7.3. Seguridad sobre la información del usuario

Se promueve entre los desarrolladores el empleo de: 1. autenticación, 2. anonimato y 3. cifrado.

7.3.1. Autenticación

Para evitar el robo de identidad se recurre a la autenticación de *n* factores, a saber: 2F A o 3F A de seguridad. Los factores de seguridad son: conocimiento, posesión y biométricos.

- Factor de conocimiento: contraseña o respuesta a una pregunta.
- Factor de posesión: el usuario posee un *token* de algún tipo; por ejemplo, una calculadora de claves, o una cuenta con una clave.
- Factor biométrico: algún parámetro físico de la persona; por ejemplo, huella digital, patrón de retina o reconocimiento facial [37] [38].

7.3.2. Anonimato

Un esquema de seguridad para los datos personales es el anonimato en las peticiones a servicios *Web* y el anonimato durante la transmisión de información sensible (por ejemplo, transacciones bancarias) a través de la red de Internet. Youngseok [39] propone emplear un esquema de autenticación anónimo sin enlace (*unlinkability*). Este esquema consiste en disociar la información funcional de la información personal; con este esquema nadie sabrá que un conjunto de sesiones particulares tiene su origen en un mismo usuario, tampoco sabrá quién es el usuario.

7.3.3. Cifrado

El objetivo del cifrado es restringir el acceso a la información a un conjunto de usuarios no autorizados. De acuerdo con Mollin [40] [41], el cifrado es un proceso que consiste en convertir *Plain Text en Cipher Text*. Donde *Plain Text* es un mensaje escrito de tal forma que es comprensible por el humano, y *Cipher Text* es el mismo mensaje escrito de tal forma que es incomprensible para el humano. El cifrado se utiliza para el almacenamiento de información o para la transmisión de la información a través de una red insegura, como la Internet. Para acceder a la información original es necesario realizar el proceso inverso (denominado descifrado), que consiste en convertir *Cipher Text* en *Plain Text*. Existen dos tipos de cifrado: simétrico y asimétrico.

Cifrado simétrico formado por una llave secreta (*secret key*), un algoritmo de cifrado y un algoritmo de descifrado. En este caso, tanto el emisor como el receptor conocen la llave secreta. La distribución de la llave secreta debe realizarse a través de un canal seguro, como lo es persona a persona. El algoritmo de cifrado recibe como entradas el mensaje original y la llave secreta, y como salida proporciona el mensaje cifrado. El algoritmo de descifrado recibe como entradas el mensaje cifrado y la llave secreta, y como salida proporciona el mensaje original. Algunos algoritmos de cifrado y descifrado son: *AES*, *DES* e *IDEA*. Este tipo de cifrado es ampliamente usado en estrategias para el almacenamiento de la información; por ejemplo, cifrar la información antes de ser almacenada en una base de datos, cifrar la información contenida en un archivo y cifrar el sistema de archivos. En los dispositivos móviles se debe evaluar la conveniencia de emplear técnicas cifrado para el almacenamiento de información, porque consumen gran cantidad de recursos, a saber: tiempo de *CPU*, espacio de memoria y consumo de energía eléctrica.

Cifrado asimétrico formado por una llave pública (*public key*) una llave privada (*private key*), un algoritmo de cifrado y un algoritmo de descifrado. En este caso la llave pública es distribuida libremente por el propietario; mientras que la llave privada debe mantenerla en secreto. El algoritmo de cifrado recibe como entradas el mensaje original y la llave pública del receptor, y como salida proporciona el mensaje cifrado. El algoritmo de descifrado recibe como entradas el mensaje cifrado y la llave privada del receptor, y como salida proporciona el mensaje original. Algunos algoritmos de cifrado y descifrado son: RSA y ECC. Este tipo de cifrado es ampliamente usado por las App Store, la cuales proporcionan una llave pública (*Certificate*) y una llave privada (*PKI-Public Key Infrastructure*) para firmar el *.apk* o el *.dmg*; con este mecanismo, el sistema operativo del dispositivo reconoce el origen y la confiabilidad de la App.

7.4. Conexión a la red de Internet

Una solución a la desconexión o falta de conectividad del dispositivo móvil a la red de Internet es la implementación del esquema de sincronización de datos. Cuando una Web App se encuentra fuera de línea (*offline*), se realiza una parte del procesamiento en el dispositivo móvil; se almacenan los datos localmente y de manera temporal, hasta que logre establecer nuevamente comunicación con el servidor remoto. Una vez establecida la comunicación, el servicio Web recibe los datos; posteriormente, compara la fecha de estos con la fecha de los datos que tiene almacenados localmente; si la fecha de los datos recibidos es superior, entonces compara los datos, y si son diferentes actualiza los datos localmente; por ejemplo, datos añadidos, modificados o eliminados. Si la fecha de los datos recibidos es inferior a la

fecha de los datos almacenados por el servidor, entonces se envía una notificación al dispositivo móvil para iniciar el proceso de actualización en él. Por lo tanto, este esquema de sincronización es bidireccional. Un ejemplo de sincronización de datos es Store and Forward [42].

7.5. Usabilidad

Grupos de desarrolladores, diseñadores de interfaces gráficas y expertos en *HCI* se han dado a la tarea de definir: 1. lineamientos, 2. guías de diseño, patrones de diseño de interfaces gráficas, 4. modelos y sistemas de evaluación. Los lineamientos para crear Apps dependen de cada tienda, *Apple Store* [43], *PlayStore* [44] y *Amazon AppStore* [11]. Algunas guías de diseño para crear Apps son *Material Design* [45], *Foundations* [16] y *Polymer* [46]. Los patrones de diseño de interfaces gráficas son *navigation*, *search form*, *signin*, *basic table*, entre muchos otros [47]. Los modelos de evaluación heurísticos están basados en breves cuestionarios para expertos o para los usuarios. También se realizan *benchmarks* para comparar el rendimiento de las Apps en cuanto a facilidad de uso y facilidad para ser aprendida.

8. Resultados y discusión

Los resultados de este análisis son: i) una clasificación de cinco retos que un desarrollador novel debe tener en cuenta en la creación de una App, ii) un compendio de retos-soluciones que puede guiarlo para obtener una App de calidad, y iii) un conjunto de objetivos cualitativos dirigidos hacia buenas prácticas de diseño e implementación. La **Tabla *2*** presenta en su primera columna el reto, y en la segunda columna presenta los objetivos.

Tabla 2. *Objetivos para cada aspecto*.

Reto	Objetivos
Capacidades de hardware del dispositivo	Maximizar el uso los recursos de hardware de los dispositivos móviles
Heterogeneidad de plataformas móviles	Fomentar el uso de *frameworks*, tal que haga posible emplear las soluciones en la mayoría de las plataformas móviles
Seguridad sobre la información del usuario	Proveer un nivel de seguridad aceptable para garantizar la confidencialidad de los datos sensibles de los usuarios
Conectividad a la red de Internet	Garantizar el uso de estándares para el intercambio de datos de forma segura e integral
Usabilidad de las aplicaciones móviles	Proveer una experiencia de usuario tal que las aplicaciones sean atractivas, fáciles de aprender y fáciles de emplear, sin dejar atrás la calidad de las funcionalidades

Gracias a la dupla reto-solución es posible medir el grado de cumplimiento de estos objetivos, por parte de una App, haciendo un listado de los retos-soluciones, e indicando si se cumple o no cada uno de ellos. El resultado de contar y ponderar las respuestas

determinará el nivel de calidad de la App. Por lo tanto, los objetivos encontrados en este artículo pueden guiar sin sesgo, al desarrollador novel, en creación y mejora de las Apps.

La clasificación de los retos y soluciones se resume en la **Tabla *3*** y la **Tabla *4***.

Tabla 3. *Resumen de retos y soluciones I.*

Retos	Soluciones
Capacidades del dispositivo	
Diferentes tamaños y resoluciones de la pantalla implican diseñar e implementar diferentes interfaces gráficas.	Emplear uno de los siguientes mecanismos: *Responsive Design* y *Adaptive Web Design* para la adaptación de contenidos de la interfaz gráfica al tamaño y resolución de la pantalla del dispositivo móvil.
La batería es consumida, principalmente por: llamadas telefónicas, pantalla encendida, *Wi-fi*, *GPS* y *CPU*.	Disminuir el empleo de la pantalla recurriendo al modo de audio. Emplear esquemas de color que reduzcan el consumo de energía. Optimizar los algoritmos que dan origen a las funcionalidades de la App para que empleen el procesador el menor tiempo posible. Disminuir la cantidad de conexiones a la red de Internet, empleando la técnica de memoria *cache*.
Uso efectivo de la unidad de procesamiento central (*CPU*).	Escoger adecuadamente la arquitectura de la App a desarrollar. Emplear buenas prácticas durante el diseño e implementación de algoritmos. Emplear patrones de diseño de software.
Uso efectivo de la RAM, la memoria interna y la memoria externa.	Evaluar el empleo de memoria interna o externa, teniendo en cuenta los requerimientos de la App, en cuanto al espacio de memoria y a la velocidad de acceso a los datos. Emplear la técnica de memoria cache.
Heterogeneidad de plataformas móviles	
Tecnologías diferentes e incompatibles para el desarrollo de las Apps	Emplear herramientas que permiten construir *Native Apps* para distintas plataformas, al mismo tiempo.

Tabla 4. *Resumen de retos y soluciones II.*

Retos	Soluciones
Seguridad sobre la información del usuario	
Los dispositivos móviles conectados a la red de Internet están expuestos a ataques provenientes de programas maliciosos.	Evitar el robo de identidad recurriendo al mecanismo de autenticación de *n* factores de seguridad (2FA o 3FA). Emplear un esquema de seguridad basado en el anonimato durante las peticiones y la transmisión de datos personales a través de la red de Internet. Garantizar la seguridad de las comunicaciones entre un dispositivo móvil y un servidor remoto, empleando algoritmos de cifrado.
Conexión a la red de Internet	
Desconexión de los dispositivos móviles a la red de Internet.	Emplear un esquema de sincronización bidireccional, tal que Store and Forward
Usabilidad	
Las Apps compiten por el gusto de los usuarios cuando varias de ellas tienen el mismo propósito.	Emplear lineamientos, guías de diseño, patrones de diseño de interfaces gráficas y sistemas de evaluación de las Apps. Emplear durante el proceso de desarrollo, paradigmas como el diseño centrado en el usuario (UCD), o experiencia del usuario (UX).

Tal y como lo hemos argumentado en la sección 6.1, los recursos de hardware de los dispositivos móviles están limitados. Las Apps compiten por el empleo de estos recursos; por lo tanto, una App de calidad, bien diseñada e implementada, será aquella que emplee los recursos mínimos necesarios. Las Apps de calidad tendrán una escalabilidad en generaciones futuras de dispositivos móviles. El usuario las percibe como rápidas y eficientes; rápidas en los tiempos de respuesta de las tareas encomendadas, y eficientes en el consumo energético. Además, las Apps de calidad desincentivan el paradigma de la obsolescencia programada de los dispositivos móviles.

Según lo descrito en la sección 6.2, existen diferentes plataformas móviles en el mercado de Apps. Para cada producto de software, que busca solucionar un problema en plataformas móviles, surge la necesidad de definir si implementar la App para una sola plataforma o para varias de éstas. Normalmente se requiere un gran esfuerzo para realizar la misma App para diferentes plataformas. Es obvio que hay una carrera por desarrollar Apps que tengan penetración en los mercados para atraer la atención de los usuarios que emplean distintas plataformas. Por lo tanto, el desarrollador debe saber que existen *frameworks* que posibilitan diseñar e implementar Apps para diferentes plataformas al mismo tiempo. Para que la App sea consistente con ser de calidad, es posible refinar el código generado por el *framework*, *refactoring*, para optimizar el uso de recursos en tareas específicas.

Según lo descrito en la sección 6.3, garantizar la seguridad de los datos sensibles del usuario es un tema fundamental de cualquier desarrollo de software; cobra especial atención en las App porque muchas tareas se realizan a través de ellas, por ejemplo: transacciones bancarias, ofertas de trabajo, redes sociales, entre otras. Las Apps envían datos sensibles del usuario a servicios en la nube; por lo tanto, el desarrollador debe conocer los esquemas de seguridad disponibles, los protocolos de comunicación y mecanismos de cifrado para que las transacciones con datos sensibles no sean robadas por atacantes malintencionados a través de *malware*, por mencionar uno de los posibles casos. Es importante saber que las operaciones de cifrado son costosas en recursos de hardware, así que se debe lograr un equilibrio entre la seguridad de los datos sensibles, el desempeño de la App y el consumo de energía.

Tal y como se mencionó en la sección 6.4, los recursos computacionales de los dispositivos móviles, para algunas tareas, se ven superados, y es cuando se recurre a delegar algunas tareas a un servicio en la nube, o a servicios disponibles en servidores remotos, *offloading*. Por un lado, el *offloading* implica transmitir datos mediante la red de Internet y, por otro lado, las Apps se enfrentan a la falta de cobertura de la red. Por lo tanto, los desarrolladores deben conocer los esquemas de sincronización que les permitan garantizar el intercambio de datos de forma segura e integral a pesar de la falta de cobertura de la red.

A partir de la sección 6.5, podemos afirmar que los desarrolladores pueden satisfacer los requerimientos funcionales de las Apps; sin embargo, algo recurrente en los desarrolladores es que no observan el aspecto de la usabilidad durante el diseño de las Apps. Los desarrolladores deben saber que existen guías de estilo, patrones de diseño de interfaces gráficas, métodos de desarrollo centrado en el usuario y métricas de la experiencia del usuario al emplear las Apps. El ejercicio de este conocimiento mejora las Apps, haciéndolas atractivas, fáciles de aprender y fáciles de emplear, generando así una sensación de satisfacción en el usuario.

9. Conclusiones y perspectivas

El objetivo principal del presente artículo es proporcionar al lector novel en el desarrollo de aplicaciones móviles, un compendio de retos-soluciones que lo guíen para lograr un equilibrio en los cinco retos implicados en la creación de Apps, sabiendo de antemano que la App “perfecta” o libre de “errores” no existe. El análisis se centró en cinco retos: 1) Capacidades del dispositivo móvil, 2) Heterogeneidad de plataformas móviles, 3) La seguridad sobre la información del usuario, 4) Conexión a la red de Internet y 5) Usabilidad de la App. A partir de este análisis pudimos observar un patrón en donde para cada problema existe, al menos, una solución. Las capacidades de los dispositivos móviles condicionan el diseño e implementación de las Apps; sin embargo, el desarrollo de Apps está alcanzando un nivel de madurez, gracias a las soluciones propuestas, y a la mejora continua de éstas. Se observa una tendencia hacia la convergencia de las tecnologías tanto en el hardware como en el software de los dispositivos móviles.

Por un lado, los fabricantes emplean componentes de hardware similares y, por otro lado, los ingenieros de software están proponiendo *frameworks* multiplataforma para el desarrollo de Apps. Otra tendencia observada es la creación de herramientas que agilizan el desarrollo de las Apps como son las herramientas de meta-programación y plataformas de desarrollo cruzado. Además, es cada vez más frecuente encontrar equipos de desarrollo multidisciplinarios trabajando

en el diseño e implementación de una App. El equipo multidisciplinario resuelve de manera integral el problema, atendiendo los requerimientos funcionales y no-funcionales. De tal manera que la App responda a las características cualitativas demandadas por el sector al que va dirigida, sin dejar de lado la usabilidad. Por último, se observa que la dupla reto-solución sirve, como métrica (se cumple o no), para conducir el diseño de una App hacia lograr los objetivos cualitativos relacionados. La métrica también permitirá medir el grado de calidad, por parte de una App existente, con el propósito de mejorarla. En ambos casos se promueve buenas prácticas en la creación y mejora de las Apps.

Bibliografía

[1] Bose, I., Ping, W., Shan, M. W., Shing, W. K., Shing, Y. Y., Tin, C. L., and Wai, S. K. "Databases for Mobile Applications". *Encyclopedia of Database Technologies and Applications*. IGI Global, 2005. 162-169.

[2] Khedkar, S., Thube, S., Estate, W. I., and Naka, C., "Real time databases for applications", *Int. Res. J. Eng. Technol. IRJET*, vol. 4, Núm. 06, pp. 2078–2082, 2017.

[3] Barday, K. A., *Data processing systems and methods for operationalizing privacy compliance via integrated mobile applications*. Google Patents, 2018.

[4] Akowuah, F., Ahlawat, A., and Du W., "Protecting Sensitive Data in Android SQLite Databases Using TrustZone", in *Proceedings of the International Conference on Security and Management (SAM)*, 2018, pp. 227–233.

[5] Barbará D., "Mobile computing and databases-a survey», *IEEE Trans. Knowl. Data Eng.*, vol. 11, Núm.1, pp. 108–117, 1999.

[6] Nayebi F., Desharnais, J.-M., and Abran A., "The state of the art of mobile application usability evaluation". In *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, 2012, pp. 1–4.

[7] Waluyo, A. B., Zhu, F., Taniar, D., and Srinivasan B., "Design and implementation of a mobile broadcast system". In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, 2014, pp. 449–456.

[8] "AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube." https://aws.amazon.com/es/ec2/ (accessed Aug. 19, 2019).

[9] Sadoski, D., "Client/Server Software Architectures–An Overview, Software Technology Roadmap", 1997.

[10] Sharma, A. Kumar, M., and Agarwal, S., "A complete survey on software architectural styles and patterns", *Procedia Comput. Sci.*, vol. 70, pp. 16–28, 2015.

[11] "Amazon Appstore Presubmission Checklist | App Submission and Testing." https://developer.amazon.com/es-mx/docs/app-submission/presubmission-checklist.html (accessed Aug. 19, 2019).

[12] J. S. S. T. Association, *JEDEC Standard: Low Power Double Data Rate 3*. 2014.

[13] Elle, C. E. P. T., and Sharusvens, P., "Les Bases de données mobiles et réparties", 2016.

[14] "IDC-Smartphone Market Share - OS", 2017 Q1. https://www.idc.com/promo/smartphone-market-share/os (accessed Aug. 19, 2019).

[15] Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., and Cavallaro, L., "The

evolution of android malware and android analysis techniques", *ACM Comput. Surv. CSUR*, vol. 49, Núm. 4, pp. 1–41, 2017.

[16] "The most advanced responsive front-end framework in the world. | Foundation." https://get.foundation/ (accessed Aug. 19, 2019).

[17] Nicholson, A. J., and Noble, B. D., "Breadcrumbs: forecasting mobile connectivity". In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 46–57.

[18] Suh, D., Ko, H., and Pack, S., "Efficiency analysis of WiFi offloading techniques", *IEEE Trans. Veh. Technol.*, vol. 65, Núm. 5, pp. 3813–3817, 2015.

[19] Poularakis, K., and Tassiulas, L., "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks", *IEEE Trans. Mob. Comput.*, vol. 16, nº. 3, pp. 675–687, 2016.

[20] Marcotte, E., *Responsive web design: A book apart n. 4*. Editions Eyrolles, 2017.

[21] Gustafson, A., *Adaptive web design: crafting rich experiences with progressive enhancement*. New Riders, 2015.

[22] Dong, M., and Zhong, L., "Chameleon: a color-adaptive web browser for mobile OLED displays". In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011, pp. 85–98.

[23] Li, D., Tran, A. H., and Halfond, W. G., "Making web applications more energy efficient for OLED smartphones". In *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 527–538.

[24] Larman, C., *Applying UML and patterns: an introduction to object oriented analysis and design and iterative development*. Pearson Education India, 2012.

[25] Schmidt, D. C., Stal, M., Rohnert, H., and Buschmann, F., *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects*, vol. 2. John Wiley & Sons, 2013.

[26] Alreshidi, A., Ahmad, A., Altamimi, A. B., Sultan, K., and Mehmood, R., "Software Architecture for Mobile Cloud Computing Systems", *Future Internet*, vol. 11, Núm. 11, p. 238, 2019.

[27] Hohpe, G., and Woolf, B., *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[28] Ritter, D., May, N., and Rinderle-Ma, S., "Patterns for emerging application integration scenarios: A survey", *Inf. Syst.*, vol. 67, pp. 36–57, 2017.

[29] Park, J., Adeli, H., Park, N., and Woungang, I., *Mobile, Ubiquitous, and Intelligent Computing*. Springer, 2014.

[30] Wasserman, A. I., "Software engineering issues for mobile application development". In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 397–400.

[31] Barbera, M. V., Kosta, S., Mei, A., Perta, V. C., and Stefa, J., "Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system". In *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 2355–2363.

[32] Wu, K.-L., Yu, P. S., and Chen, M.-S., "Energy-efficient caching for wireless mobile computing". In *Data*

Engineering, 1996. Proceedings of the Twelfth International Conference on, 1996, pp. 336–343.

[33] Bžoch, P., and Šafařík, J., "Algorithms for maintaining consistency of cached data for mobile clients in distributed file systems", *Int. J. Distrib. Syst. Technol. IJDST*, vol. 8, Núm. 1, pp. 17–33, 2017.

[34] "Xamarin | Open-source mobile app platform for .NET." https://dotnet.microsoft.com/apps/xamarin (accessed Aug. 19, 2019).

[35] "Apache Cordova." https://cordova.apache.org/ (accessed Aug. 19, 2019).

[36] Titanium, A., *Titanium Mobile Application Development Appcelerator Inc*. 2018.

[37] Labrou, Y., and Agre, J. R., *Authentication services using mobile devices*. Google Patents, 2009.

[38] Dill, M., Narayan, P. L., Powell, G., Sheets, J., and Carpenter, A., *Systems and methods for communicating token attributes associated with a token vault*. Google Patents, 2018.

[39] Chung, Y., Choi, S., and Won, D., "Lightweight anonymous authentication scheme with unlinkability in global mobility networks", vol. 4, no. 4, p. 7, 2013.

[40] Mollin, R. A., *An introduction to cryptography*. Chapman and Hall/CRC, 2006.

[41] Hussain, I., Alqahtani, A., and Gondal, M. A., "An efficient method for secure communication of biometric information based on chaos", *3D Res.*, vol. 6, Núm. 2, p. 15, 2015.

[42] Singh, U. K., and Narayanan, A. K., *System and method for a generic mobile synchronization framework*. Google Patents, 2014.

[43] "App Store Review Guidelines - Apple Developer." https://developer.apple.com/app-store/review/guidelines/ (accessed Aug. 19, 2019).

[44] "Lineamientos de marca | Google Play | Android Developers." https://developer.android.com/distribute/marketing-tools/brand-guidelines (accessed Aug. 19, 2019).

[45] "Design - Material Design." https://material.io/design (accessed Aug. 19, 2019).

[46] "Polymer Project." https://www.polymer-project.org/ (accessed Aug. 19, 2019).

[47] Neil, T., *Mobile design pattern gallery: UI patterns for smartphone apps*. O'Reilly Media, Inc., 2014.