Universidad Católica "Nuestra Señora de la Asunción"

Facultad de Ciencias y Tecnología (CyT)

Departamento de Electrónica e Informática (DEI)

Proyecto Final de Carrera

Ingeniería Informática

Primera defensa

Propuesta MDD para la persistencia de datos en aplicaciones nativas orientadas a teléfonos móviles inteligentes[1]

Autor:
Manuel Núñez

Tutor:
Ing. Magalí González
Co-tutor:
Ph.D. Luca Cernuzzi
Colaborador:
M.Sc. Nathalie Aquino

24 de mayo de 2016

Asunción, Paraguay

[1] El presente Proyecto Final de Carrera ha sido desarrollado con el apoyo financiero del Consejo Nacional de Ciencia y Tecnología (CONACYT, Paraguay) en el marco del proyecto denominado "Mejorando el proceso de desarrollo de software: propuesta basada en MDD" (14-INV-056)

1. Introducción

El auge de las aplicaciones móviles y la facilidad con la que hoy en día uno puede acceder a un teléfono móvil inteligente[2], o *smartphone* en inglés, promueve a que año tras año vaya creciendo el número de usuarios de éstos dispositivos; y de la misma forma, se produce el aumento del volumen de ventas en el mercado incentivados por la demanda y por la economía que mueve este sector tecnológico. Sin duda, este sector es de gran importancia y no está libre de problemas, sino en constante cambio y evolución[3]. Enfocar nuestro estudio aquí es de fundamental importancia, por su relevancia en la actualidad,en especial al sector informático.

La variedad de teléfonos móviles inteligentes disponibles hoy día en el mercado es muy amplia. El número de sistemas operativos no es tan elevado, pero las diferentes versiones de un mismo sistema operativo incrementan la variabilidad con la que se debe lidiar al momento de desarrollar aplicaciones. Cada sistema operativo móvil presenta un Entorno de Desarrollo Integrado (IDE - *Integrated Development Environment*) propio, con un diseño de interfaz diferente; así también, el manejo de los recursos de hardware y de los datos en las aplicaciones [4] suele realizarse de manera diferente en cada sistema. Este ambiente fragmentado es uno de los principales retos para los desarrolladores de aplicaciones móviles en la actualidad, donde la variedad de plataformas[3] origina este fenómeno conocido como **fragmentación** [5] [6] [7] [8].

Ante tanta variedad de problemas ocasionados por la fragmentación en diferentes aspectos (hardware, interfaz, manejo de datos, entre otros), en el marco de desarrollo de este Proyecto Final de Carrera o PFC, nos centraremos en un aspecto en particular: **la persistencia de datos en los teléfonos móviles**.

Toda aplicación necesita almacenar o retener datos, ya sea de forma permanente o temporal. Las aplicaciones móviles no cuentan con un disco duro ni con una conexión permanente a datos alojados en un servidor, sino que almacenan datos que pueden ser sincronizados luego cuando se cuente con una conexión [9]. Estos datos pueden ser guardados temporal (en caché) o permanentemente en el dispositivo (mediante archivos y bases de datos). Pueden ser de distintos tipos: documentos, respaldo de archivos, media (imágenes, música y vídeo), entre otros. Su origen puede variar, desde una simple configuración de la aplicación que el usuario realiza, a datos precargados en un formulario de la aplicación. De igual forma, los datos pueden provenir de un servidor remoto (por ejemplo, mediante una consulta a un servicio), los cuales pueden ser manipulados localmente [10], guardados en caché, y así estar disponibles aún cuando no se cuente con una conexión a red, pudiendo ser sincronizados luego [11]. Si se desea trabajar con datos que necesitan ser manipulados localmente, o se cuenta con gran cantidad de datos, el mecanismo más utilizado y tradicional, son las bases de datos; pero, existen otros métodos que mencionaremos a continuación.

Como vimos, son varios los escenarios en los que se necesita almacenar datos en las aplicaciones móviles. La fragmentación trae consigo que cada plataforma presente diferentes opciones de persistencia, y a su vez, maneje de forma diferente los distintos mecanismos de persistencia, dificultando esta tarea. Son varias las opciones y mecanismos de almacenamiento, como ser: i) HTML5 (`http://www.w3.org/TR/html5/`) y sus métodos específicos de persistencia dentro de las aplicaciones web; ii) almacenamiento local con archivos y bases de datos integradas como SQLite (`http://www.sqlite.org/`); y, iii) almacenamiento externo con servicios en la nube (como Dropbox y Google Drive), y dispositivos de almacenamiento externo (por ejemplo, tarjetas SD). Hablamos también de mecanismos de almacenamiento temporales, como el almacenamiento tipo pares clave - valor, muy utilizados dentro de la aplicación para guardar datos de configuración, datos de sesión y para pasar datos entre pantallas y aplicaciones [12] [9] [4].

La fragmentación incrementa el tiempo de desarrollo y el costo de mantenimiento de las aplicaciones; tener esto en cuenta al momento de desarrollar una aplicación móvil es muy importante debido a que se debe determinar el alcance que tendrá la aplicación (es decir, si se desea una aplicación para una plataforma específica o para varias) [6]. Para lograr el desarrollo multiplataforma existen varios enfoques: i) el desarrollo web móvil, ahorra esfuerzo de desarrollo, pero con resultados pobres en cuanto rendimiento e interfaz [13] [5]; y, ii) el desarrollo nativo, implica mayor tiempo y esfuerzo de desarrollo, pero arroja mejores resultados en apariencia y aprovechamiento de recursos del dispositivo[13]. Este enfoque es el más afectado por el fenómeno de la fragmentación [6], razón por la cual nos centraremos en los problemas asociados a este enfoque.

Siguiendo el marco investigativo en el que se desarrolla nuestro PFC, analizaremos la adopción del Desarrollo Dirigido por Modelos (MDD - *Model Driven Development*) como herramienta de solución para minimizar la brecha de heterogeneidad que supone la fragmentación, y de esa forma reducir el esfuerzo de desarrollo de las aplicaciones móviles nativas entre plataformas.

La idea que nos presenta MDD es describir un problema en un modelo y generar software a partir de esta representación [14] [15] [16]. Desarrollar aplicaciones que comparten las mismas funcionalidades y comportamiento, pero para diferentes plataformas, constituye un área adecuado para esta metodología [17] [8]. MDD trata el problema de la redundancia de tareas, reduciendo el esfuerzo de programación y los errores de codificación [18]. Así, el uso de metodologías basadas en MDD, frente las problemáticas presentadas previamente, puede suponer facilidades en el desarrollo de aplicaciones para teléfonos inteligentes [8].

Atendiendo a lo dicho anteriormente, la propuesta MoWebA [19] es un enfoque metodológico MDD para el desarrollo de aplicaciones web que adopta el esquema de Arquitectura Dirigido por Modelos (MDA - *Model Driven Architecture*). Esta metodología podría constituirse en una opción adecuada para el entorno móvil, gracias a su estructura por capas bien definida (mediante la cual se logra una clara separación de conceptos) y el modelado centrado en la navegación jerárquica orientada a funciones. En este sentido, esta propuesta proporciona un esquema más adecuado para el diseño de una navegación basada en la interacción del usuario o del contexto [19], al considerar que la manera en la cual la información es organizada y estructurada dentro del sistema no es necesariamente la misma en la que los usuarios acceden a ella. Las aplicaciones móviles son dirigidas por eventos, así que este diseño de navegación sería muy adecuado para el entorno móvil. De la misma forma, MoWebA propone un enriquecimiento de los modelos existentes en orden a considerar aspectos relacionados a la arquitectura final del sistema (por ejemplo RIAs, SOA, REST, entre otros), gracias a su Modelo Específico de la Arquitectura (ASM - *Architecture Specific Modelling*) [19]. Mediante este modelo se podría analizar la posibilidad de representar aplicaciones móviles, utilizando conceptos específicos de la arquitectura y plataforma móvil.

[2] Dispositivo móvil con avanzados recursos computacionales y equipado con tecnologías que facilitan el acceso a Internet, corren aplicaciones, son táctiles, poseen cámara y otros sensores, todos bajo un avanzado sistema operativo [1] [2].

[3] Una plataforma móvil comprende el hardware subyacente del dispositivo, la arquitectura sobre la que está basada, el sistema operativo, el Kit de Desarrollo de Software (SDK - *Software Developer Kit*) del proveedor y las librerías estándares [2]

Como objetivo general nos planteamos desarrollar una propuesta MDD que contemple la persistencia en el desarrollo de aplicaciones móviles nativas. Para ello extenderemos MoWebA al ambiente móvil, realizando los ajustes necesarios para dotarla de la capacidad de considerar aspectos y opciones de persistencia durante el desarrollo de aplicaciones móviles.

El resto del presente documento se distribuye considerando las siguientes secciones: en la sección 2 realizamos una descripción general de lo que implica el desarrollo de aplicaciones para móviles, analizando características, herramientas disponibles y problemas. En la sección 3 presentamos nuestra propuesta de solución, detallando la problemática que atacamos y la solución que planteamos. Así mismo, describimos los trabajos realizados y pendientes, en el marco del avance de este PFC.

2. El desarrollo de aplicaciones móviles

En esta sección se introducen las bases teóricas que fueron utilizadas en el desarrollo del actual PFC, presentando un resumen de la literatura estudiada. Este apartado tiene como objetivo primordial dar a conocer de forma organizada y coherente, una síntesis de los conceptos y el marco teórico de la investigación con el cual pueden abordarse los problemas tratados en la introducción.

Las siguientes subsecciones se dividen en: 2.1) analizamos características de los teléfonos móviles inteligentes; 2.2) describimos los enfoques de desarrollo disponibles para las aplicaciones móviles y comparamos herramientas disponibles actualmente para el desarrollo multiplataforma de aplicaciones; 2.3) nos enfocamos y analizamos la persistencia de datos para el entorno móvil, analizando problemas y opciones de almacenamiento; 2.4) hablamos del desarrollo dirigido por modelos y realizamos un estudio de la literatura con propuestas actuales de solución para el desarrollo móvil; y por último, 2.5) hablamos de la propuesta MoWebA.

2.1. Aplicaciones y sistemas operativos móviles

Los teléfonos móviles inteligentes son caracterizados por una gran habilidad computacional: pueden correr sistemas operativos completos con un número de aplicaciones preinstaladas (propias del sistema operativo). Aplicaciones de terceros pueden ser igualmente descargadas e instaladas desde tiendas virtuales, especialmente diseñadas por las compañías desarrolladoras de los sistemas operativos, para el acceso a contenidos para sus usuarios, así como para la comercialización de las mismas. Como principales tiendas virtuales tenemos al Playstore (`htttp://play.google.com`), exclusivo para usuarios de Android; y al AppStore (`https://itunes.apple.com/es/genre/ios/id36?mt=8`), exclusivo para usuarios de iOS [13].

Las aplicaciones móviles presentan ciertos requerimientos adicionales que no son comúnmente encontrados con las aplicaciones de software tradicionales (por ejemplo, las aplicaciones de escritorio), lo que provoca que el desarrollo sea diferente [17][20]:

- Son dirigidas por eventos y constantemente tienen que reaccionar a entradas de las interfaces de usuarios o sensores, así como a la comunicación vía red. Un criterio de calidad crucial e importante relacionado a esta interacción con la interfaz, es la *responsiveness*[4].
- Interactúan con otras aplicaciones.
- Tienen la posibilidad de manejar sensores (por ejemplo, giróscopo, GPS, movimiento, luz, aceleración, entre otros) y hardware específico del dispositivo móvil (por ejemplo, cámara, batería, bluetooth, NFC, entre otros).
- Generalmente, los teléfonos inteligentes requieren de una conexión a Internet activa para realizar peticiones a servicios web y procesar las respuestas. Además, cuentan con almacenamiento local en caso de que la red no esté disponible. Es importante tener estas consideraciones durante el desarrollo de aplicaciones.
- Trabaja con recursos limitados, como la memoria del dispositivo, la batería y la conectividad, entre otros.

En una aplicación para teléfonos móviles inteligentes podrían considerarse dos aspectos: i) aspectos estructurales, como la interconexión entre módulos y librerías, así como la estructura de la Interfaz Gráfica de Usuario (GUI - *Graphical User Interface*); y ii) aspectos de comportamiento (por ejemplo, reacción a eventos y computación específica de la aplicación) [21].

Actualmente los sistemas operativos más utilizados en los teléfonos móviles inteligentes son **iOS** (desarrollado por Apple), **Android** (desarrollado por Google), y **Windows Phone** (desarrollado por Microsoft). Cada uno de estos sistemas requieren de lenguajes de programación diferentes, diferentes IDE y modelos de programación basados en API (*Application Programming Interface* o Interfaz de Programación de Aplicaciones) propios de cada sistema operativo. Así sabemos que desarrollar aplicaciones para iOS requiere de Swift, Android de Java y Windows Phone de .NET. Además de los ya mencionados, existen otros sistemas operativos presentes en el mercado móvil actual, como FireOS (desarrollado por Amazon), Blackberry OS (desarrollado por Blackberry), Tizen (desarrollado por Samsung) y FirefoxOS (desarrollado por Mozilla) [22].

Según estudios realizados por la International Data Corporation (IDC), en su informe del mercado de los sistemas operativos de teléfonos móviles inteligentes (`http://www.idc.com/prodserv/smartphone-os-market-share.jsp`), la cuota de mercado es bien diferenciada para cada sistema operativo, siendo Android quien domina el mercado con un 82.8 %, seguido por iOS con un 13.9 %, Windows Phone con un 2.6 % y Blackberry OS con un 0.3 %. Otros constituirían el 0.4 % restante.

2.2. Enfoques de desarrollo

La variedad de plataformas disponibles actualmente genera el fenómeno de la fragmentación [5] [6] [7] [8], que incrementa el tiempo de desarrollo y costo de mantenimiento de las aplicaciones. Es determinante tener en cuenta este aspecto al momento de desarrollar una aplicación móvil, puesto que se debe identificar el alcance que tendrá la aplicación y así, optar por el enfoque de desarrollo a utilizar: se puede desarrollar aplicaciones nativas para cada posible plataforma, o desarrollar una vez una aplicación web para todas las plataformas, o bien, desarrollar aplicaciones con lo mejor de cada uno de estos enfoques mencionados. A continuación resaltaremos las principales características, ventajas y desventajas de cada enfoque.

[4] Se refiere a qué tan rápido la aplicación responde a una entrada de la interfaz

2.2.1. Desarrollo de aplicaciones nativas

Desarrollar una aplicación nativa para distintas plataformas se ha dificultado para las compañías. Con la variedad de sistemas operativos, se necesita desarrollar la misma aplicación para cada una de ellos. Puesto que no se puede reutilizar el mismo código fuente para otras plataformas, la misma aplicación debe ser generada desde cero, utilizando lenguajes y SDK específicos, provistos por algunos IDE como Eclipse, Android Studio (IDE oficial de Android, `http://developer.android.com/intl/es/tools/studio/index.html`), Xcode (IDE oficial de Apple, `https://developer.apple.com/xcode/`), y Visual Studio (IDE de desarrollo para Windows Phone, `https://www.visualstudio.com/`). Esto demanda más esfuerzo y tiempo de desarrollo, por la dificultad y la experiencia que requiere trabajar con este tipo de aplicaciones [13] [11].

La aplicación móvil nativa es específica para cada sistema operativo, y presenta las siguientes características [23]:

- **interfaz de usuario acorde al dispositivo:** cada sistema operativo es caracterizado por su propio estilo y modo de interacción, normalmente conocido como el *look and feel* [5];
- **acceso al sistema:** implica el acceso a aplicaciones ya instaladas, a la información contenida en el dispositivo (lista de contactos, galería de fotos, entre otros), y a escuchar los eventos y notificaciones del sistema (evolución del consumo de batería, recibo de un SMS, reanudación del sistema luego de una suspensión, entre otros);
- **acceso al hardware específico:** los teléfonos móviles inteligentes poseen una gran variedad de sensores (por ejemplo, sensores de luz, aceleración, presión, movimiento, elevación, localización, orientación del dispositivo) y recursos de hardware a los que se tiene acceso (por ejemplo, bluetooth, NFC, conectividad, etc.);
- **mejor rendimiento:** el mejor aprovechamiento de recursos de hardware y del sistema, y la apariencia acorde al dispositivo, permite al usuario final tener una mejor Experiencia de Usuario (UX - *User Xperience*)[6]. Además, el código fuente es eficiente, con alto rendimiento, consistente y con total acceso al hardware y datos subyacentes del dispositivo;
- **distribución en tiendas virtuales**, donde son ofrecidas al mercado para su comercialización [13]).

El principal problema con este tipo de aplicaciones es lidiar con las diferentes plataformas disponibles: el desarrollo para cada una ellas incrementa el costo de mantenimiento y solicita cierto nivel de experiencia (implica un costo elevado tener que conocer todo el entorno propio de cada plataforma). Esto significa dificultad y costo en términos de esfuerzo y desarrollo para una única aplicación, estableciéndose así que el reto de la generación multiplataforma se encuentra en el desarrollo de aplicaciones nativas, y no tanto en los demás enfoques. [24].

Para enfrentar estos problemas de desarrollo e interoperabilidad en móviles, una alternativa que también está siendo abordada es el desarrollo multiplataforma, para ambientes web. Aún así, la aplicación nativa es estable, segura y más capaz de acceder a los recursos del dispositivo, y sigue siendo la elección para los desarrolladores de la mayoría de las aplicaciones sociales, juegos y de comunicaciones, [25] brindando mayor Experiencia de Usuario.

2.2.2. Desarrollo de aplicaciones web

Una de las alternativas más utilizadas por las empresas es el desarrollo móvil multiplataforma, el cual permite simplificar el proceso de mantenimiento y despliegue, y ahorrar tiempo de desarrollo y esfuerzo [13]. El desarrollo de aplicaciones web implica usar un sólo código base para todas las plataformas, convirtiéndose en posible alternativa contra el problema de la fragmentación [1].

Estos tipos de aplicaciones no necesitan ser instalados en el dispositivo para poder ejecutarse ya que se ejecutan en un navegador. Se desarrollan usualmente usando PHP, Node.js, ASP.NET, HTML, CSS, o JavaScript [11] [5]. HTML5 es una alternativa prometedora: las aplicaciones basadas en esta tecnología, cuyo objetivo es unificar mediante la utilización de navegadores y componentes web, pueden correr en cualquier plataforma móvil sin ningún esfuerzo extra [5].

Las ventajas al trabajar con este tipo de aplicación son: i) reducción de los conocimientos requeridos para desarrollar aplicaciones, ya que se usa un lenguaje común; de la misma forma, el conocimiento del API propio de cada plataforma se reduce, ya que se manejan las comunes provistas por la herramienta; ii) reducción del código, debido a que se escribe una sola vez y se compila para cada plataforma soportada, lo que conlleva a una reducción del tiempo de desarrollo; iii) ampliamente soportado por la industria, de manera que la mayoría de los dispositivos tienen soporte para utilizarlo (sólo se necesita de un navegador) [13] [5].

Las desventajas, respecto a las aplicaciones nativas, son: i) menor rendimiento, puesto que todo se ejecuta mediante el intérprete Javascript del navegador, cuya potencia es limitada; ii) tanto la interfaz del usuario (no se adapta a las características propias de la plataforma subyacente) como el acceso a recursos de hardware de los dispositivos (los sensores y recursos como cámara, bluetooth, entre otros), son muy limitados; iii) no puede ser distribuida por tiendas virtuales, resultando complicada su comercialización [11].

Con las aplicaciones web se ahorran costos de esfuerzo y tiempo en el desarrollo multiplataforma, pero ofrecen una menor Experiencia de Usuario, y no hacen uso eficiente de los recursos del dispositivo. Ante esto, surgen aplicaciones que aprovechan lo mejor de estos enfoques de desarrollo: aplicaciones web móviles nativas o híbridas. A continuación veremos más detalles.

2.2.3. Desarrollo de aplicaciones web móviles nativas

Propiamente no son aplicaciones web ni tampoco nativas. Se ejecutan en un navegador, o mejor dicho, con un componente nativo que delega en un navegador (conocido como *native wrapper*. En otras palabras, no tienen la potencia de las aplicaciones nativas, simplemente ejecutan código en un navegador embebido (generalmente con HTML5). También conocidas como aplicaciones híbridas [5].

Este tipo de aplicación posee todas las ventajas de las aplicaciones web. Así mismo, en aspectos como instalación y distribución, se las puede considerar como aplicaciones nativas. De la misma manera, posee la mayoría de las desventajas de las aplicaciones web. En cuanto a la Experiencia de Usuario, a pesar de tratarse de una aplicación nativa, requiere de conexión a Internet para poder trabajar, y funciona según el navegador [5].

[5] *Look and feel* es un término usado para describir la interfaz de usuario en términos de apariencia y función [11].

[6] Experiencia de Usuario se refiere al conjunto de factores y elementos relativos a la interacción del usuario, con un entorno o dispositivo concretos, cuyo resultado es la generación de una percepción positiva o negativa de dicho servicio, producto o dispositivo. Fuente Wikipedia: `https://en.wikipedia.org/wiki/User_experience`

2.2.4. Cuadro comparativo de alternativas para el desarrollo de aplicaciones móviles

El desarrollo de aplicaciones móviles cuenta con dos grandes enfoques de desarrollo: enfoque manual y enfoque (semi)automatizado.

Cada sistema operativo móvil cuenta con su IDE, brindando al desarrollador el ambiente, servicios y herramientas necesarios para poder crear aplicaciones, utilizando un lenguaje de programación específico. Ese tipo de desarrollo es totalmente manual, generando código fuente sin ningún nivel de abstracción ni automatización, pero con ayuda del IDE para facilitar el desarrollo (por ejemplo, detección de errores, funcionalidades del API con autocompletado, *debugger*, entre otros). Las aplicaciones resultantes de este tipo de desarrollo pueden ser tanto nativas como web.

Por otra parte, existen otras alternativas que nos permiten desarrollar aplicaciones móviles, adoptando ciertos niveles de abstracción y (semi)automatizando ciertas partes del desarrollo. Hablamos de los ***frameworks*** y **enfoques dirigidos por modelos** (ver sección 2.4)[10].

La mayor ventaja de los *frameworks* es la reutilización de código y la rápida configuración para múltiples plataformas; pero, no reducen la complejidad de la implementación porque no proveen una visión general del sistema a un alto nivel, como sí sucede utilizando un enfoque MDD. En lugar de eso, solo reducen el número de implementaciones requeridas (uno solo para varias plataformas), lo que no implica que sea más simple que usando un SDK nativo, y en algunos casos no producen una aplicación verdaderamente nativa [7]. Los *frameworks* actuales tratan de combinar los aspectos positivos de las aplicaciones web y nativas, mientras que con un enfoque MDD se logra tener una visión más general y abstracciones en todo el conjunto de los subproblemas del dominio, mediante Lenguajes Específicos del Dominio (DSL - *Domain Specific Language*) [10]. El problema con el DSL es su poca documentación, dificultando su aprendizaje. Pero la ventaja es que presenta construcciones dedicadas a un dominio, en nuestro caso sería el desarrollo de aplicaciones móviles; y mediante los modelos, obtenemos un nivel de abstracción más elevado, pudiendo generar a partir de ellos (semi)automáticamente la aplicación [23].

En el **cuadro 1** realizamos una comparación entre los *frameworks* y soluciones MDD disponibles actualmente. Para el estudio de los *frameworks* nos basamos en trabajos como [22] [1] [2] [8] [23], que evalúan y comparan varias de estas herramientas; y de igual forma, investigamos en las páginas oficiales de estos. Por otra parte, las propuestas dirigidas por modelos resultan de un proceso de investigación, que lo mostraremos más adelante, llevado a cabo para estudiar las propuestas MDD disponibles para móviles (ver **sección 2.4.1**).

De los estudios mencionados obtuvimos una lista de *frameworks* y soluciones MDD, a los cuales aplicamos los siguientes criterios de selección, para así poder compararlos: i) que soporte al menos la generación de aplicaciones para dos plataformas (para así poder verificar la generación multiplataforma); y, ii) que se encuentre actualmente activa y en constante mantenimiento (nos interesan solamente las herramientas actuales).

La información recabada de cada herramienta está enfocada en los siguientes puntos: i) el tipo de licencia que contempla; ii) si es código abierto; iii) el estado de la herramienta (si es una herramienta disponible o un trabajo en desarrollo); iv) las plataformas soportadas; v) el lenguaje de programación utilizado; vi) si permite modelado y de qué tipo; vii) el tipo de aplicación que genera (nativo, web o híbrido); viii) si permiten el acceso al API nativo de la plataforma (indica si pueden acceder a recursos del dispositivo); y por último, ix) si se utiliza el esquema MVC de programación.

A continuación analizamos los resultados más relevantes obtenidos.

Respecto a las alternativas disponibles para desarrollar aplicaciones móviles

Del cuadro notamos que contamos con las siguientes 13 (trece) alternativas para desarrollar aplicaciones móviles: 6 (seis) *frameworks* y principalmente, 7 (siete) propuestas dirigidas por modelos. Entre los *frameworks* actualmente disponibles en el mercado, encontramos que los más populares son Apache Cordova (Phonegap)(`https://cordova.apache.org/`) y Titanium (`http://www.appcelerator.com/mobile-app-development-products/`) [10]. De igual forma, encontramos a Rhodes (`http://rhomobile.com/`), Sencha (`https://www.sencha.com/`), IBM Mobile First (`http://www.ibm.com/mobilefirst/us/en/`) y Xamarin (`https://xamarin.com/`). Por otra parte, están las propuestas dirigidas por modelos, en su mayoría DSL como: MD2 [26], Mobia[27], Xis-mobile DSL [7], AXIOM DSL [28], Xmob DSL [23], MobL [29] y plataformas disponibles como WebRatio móvil (`http://www.webratio.com/site/content/es/mobile-app-development`) [30].

Respecto al enfoque de desarrollo utilizado

El desarrollo nativo es una mayoría entre los enfoques de desarrollo que ofrecen las herramientas. Tenemos un total de 9/13 (69 %) herramientas que permiten desarrollar aplicaciones nativas. Entre estos, 4/6 (67 %) constituyen *frameworks* y 5/7 (71 %) son DSL. Entre los *frameworks*, el desarrollo de aplicaciones híbridas constituyen la opción dominante de generación.

Una característica importante del desarrollo nativo de aplicaciones, y que lo diferencia de otros enfoques, es poder acceder a los recursos de cada dispositivo (al API de la plataforma). Entre las propuestas MDD destacamos que todas soportan el acceso a los recursos del dispositivo, excepto Mobia, que no encontramos mayor información al respecto. Por otra parte, entre los *frameworks*, Rhode, Sancha y Phonegap permiten el acceso a ciertos recursos mediante Javascript. Xamarin también permite el acceso mediante C#.

Respecto a los lenguajes de desarrollo utilizados

Ningún lenguaje utilizado en los *frameworks* requiere un esfuerzo importante para su aprendizaje, puesto que son lenguajes muy adoptados para el desarrollo, como C#, .NET y las tecnologías web como HTML5, CSS y Javascript. Entre los lenguajes de desarrollo más utilizados tenemos a HTML5 en un 3/6 (50 %) de los *frameworks*. Entre las propuestas MDD sí existe ese esfuerzo de aprender un lenguaje nuevo de modelado, tenemos por ejemplo a MD2, Xmob, Axiom y Xis-mobile: todos DSL específicos para el dominio móvil. No pasa así con Rhodes y Mobia, que requieren de HTML5, CSS y Javascript para el modelado. Resalta que solamente Xis-Mobile DSL y WebRatio proponen la extensión de un lenguaje de modelado ya existente, en lugar de introducir uno nuevo (Xis y IFML respectivamente).

Respecto a las plataformas soportadas

Android y iOS resultan ser las plataformas más soportadas tanto por los *frameworks* como las propuestas MDD.

Respecto al estado y licencia de la herramienta

Como vimos, existen varias soluciones para el desarrollo multiplataforma. La mayoría son herramientas disponibles, a excepción de MD2, XMOB, AXIOM y Xis-Mobile que están en fase de desarrollo. Entre éstas, solamente Rhodes, Titanium y Apache Cordova son código abierto. Entre las herramientas MDD, no encontramos mucha información al respecto.

	FRAMEWORKS						MDE						
	Rhodes	**Titanium**	**Sencha**	**Apache Cordova (Phonegap)**	**IBM Mobile first**	**Xamarin**	**Mobia**	**mobl**	**md2**	**xmob dsl**	**Axiom DSL**	**XIS-Mobile DSL**	**WebRatio**
Licencia	MIT, Comercial	Apache 2.0	GPLv3 or comercial	Apache 2.0	IBM	Comercial	Comercial	MIT	Apache License 2.0	No especificado	No especificado	No especificado	Versión comunidad, gratuito; versión profesional, pago
Código abierto	Si	Si	No	Si	No	No	No	No especificado	No especificado	No especificado	No especificado	No especificado	No especificado
Plataforma soportada	iOS, Android Blackberry, Windows Phone 8	iOs, Android Blackberry y web con HTML 5	Android, iOS, BlackBerry Kindle, Windows Phone, Tizen	Android, iOS, Blackberry, Symbian, WebOS, Windows Phone, FireOS, Ubuntu OS, Firefox OS	iOS, Web con HTML, integración con *frameworks* PhoneGap, Sencha Touch, jQuery	Android, iOS y Windows Phone	iOS y Android	Navegadores basados en Web-kit, iOS, Android, WebOS, Safari y Chrome	Android y iOS	Android, iOS y .NET	Android y iOS	Android, iOS y Windows Phone	Android y iOS
MVC	Sí	Sí	Sí	No	No especificado	Sí	No	Sí	Sí	Sí	No	No	No especificado
Estado de la herramienta	Herramienta disponible	Herramienta disponible	Herramienta disponible	Herramienta disponible	Herramienta disponible	Herramienta disponible	Herramienta disponible	Herramienta disponible	Trabajo en desarrollo	Trabajo en desarrollo	Trabajo en desarrollo	Trabajo en desarrollo	Herramienta disponible
Modelo	No aplica	No aplica	No aplica	No aplica	No especificado	No aplica	Mobia modeler	DSL	DSL	DSL	DSL	Perfil UML	modelo ER ó modelo IFML
Tipo de modelado permitido	No aplica	No aplica	No aplica	No aplica	No especificado	No aplica	Gráfico	Textual	Textual	Textual	Textual	Gráfico	Gráfico
Lenguajes utilizados	HTML5/ Ruby	Javascript	HTML5, CSS y JavaScript	HTML5, CSS y JavaScript	HTML5	C# y .NET	HTML5, CSS y JavaScript	HTML5,CSS y JavaScript	MD2	Xmob	Axiom	UML	IFML
Tipo de aplicación generada	Nativo	Nativo	Híbrido	Híbrido	Web, Nativo y Híbrido	Nativo	Nativo	Web	Nativo	Nativo	Nativo	Nativo	Híbrido
Acceso a API nativo	Sí, con Javascript/ Ruby	No	Sí, con apache Cordova	Sí, con Javascript	No especificado	Sí	No especificado	Sí, con Javascript	Sí	Sí	Sí	Sí	Sí

Cuadro 1. Diferentes alternativas para el desarrollo de aplicaciones móviles

Discusión

Del análisis resaltamos los siguientes puntos: i) son más las alternativas basadas en modelos que permiten desarrollar aplicaciones verdaderamente nativas, que los *frameworks* (éstos están más enfocados en el desarrollo de aplicaciones híbridas); ii) las propuestas MDD permiten el acceso a API nativo, ya que sus DSL son específicos del dominio móvil; iii) los *frameworks* no requieren el aprendizaje de nuevos lenguajes de programación, ya que utilizan tecnologías y lenguajes web muy utilizadas actualmente (por ejemplo, HTML5, Javascript y CSS); en cambio, la mayoría de las soluciones basadas en modelos proponen lenguajes nuevos que requieren cierto aprendizaje; iv) por lo general, la mayoría de los *frameworks* disponibles ya son herramientas robustas, disponibles y con buena documentación; los DSL en su mayoría, son propuestas aún en desarrollo y con poca documentación (por lo tanto, poco accesibles); v) las plataformas más soportadas son Android y iOS.

Teniendo en cuenta lo dicho anteriormente, para el desarrollo nativo de aplicaciones móviles, creemos que las soluciones MDD podrían constituirse en opción factible y adecuada contra la fragmentación, ya que nos permiten trabajar en el dominio de los móviles, y con una (semi)automatización del proceso de desarrollo; pero claro, atendiendo que implican un esfuerzo de aprendizaje de nuevos lenguajes,y sumado a la poca documentación.

2.3. El problema de la persistencia

En las aplicaciones móviles es de capital importancia implementar técnicas de almacenamiento de datos. Por una parte, los dispositivos móviles no cuentan con un disco duro o una conexión permanente a una base de datos alojada en un servidor, sino que se debe lidiar con limitantes propias del teléfono móvil (como la memoria, batería, conectividad, entre otros) [20] [6] [31]. Es así que las aplicaciones móviles se caracterizan por contar con almacenamiento local en caso que la red no esté disponible, debido a la **naturaleza transitoria** de los teléfonos móviles inteligentes [11]. Esta transitoriedad de la conectividad, se define y manifiesta en dos grandes preocupaciones: i) los dispositivos móviles, en especial los teléfonos móviles inteligentes, **poseen conectividad de red cambiante** (3G, 4G y *wireless*), en constante variación, casi imperceptibles para el usuario; ii) **la pérdida o caída inesperada de conexión**. La conectividad de red juega un papel crucial en aquellas aplicaciones que requieren traspaso de datos entre el dispositivo y un servidor: no tener en cuenta estos aspectos puede resultar en aplicaciones con pobre usabilidad. Para mantener la Experiencia de Usuario y hacer frente a estos problemas, es necesario almacenar datos en caché y guardar información de forma local en el dispositivo [20].

El desarrollo de aplicaciones móviles generalmente involucra guardar datos. Estos datos pueden ser de distintos tipos: desde imágenes provenientes de la cámara, a configuraciones de preferencia del usuario, respaldo de archivos, documentos, entre otros. De la misma manera, estos datos pueden provenir de una fuente externa, como un servidor remoto, los cuales pueden ser manipulados localmente: estas aplicaciones son conocidas como **aplicaciones dirigidas por datos** (o *data driven apps*). El nivel de aplicación varía si los datos se persisten o simplemente se mantienen en la memoria, puesto que los teléfonos móviles tienen limitación de ancho de banda y espacio [6] [31], lo que requiere que ciertos datos deban ser guardados en caché (almacenados localmente pero no permanentemente, con la opción de que puedan ser sincronizados luego). Esto da lugar al modo "sin conexión" que muchas aplicaciones ofrecen, para el cual es necesario guardar en caché ciertos datos necesarios para que la aplicación funcione sin conexión a red [10]. Aquí radica la importancia de contar con bases de datos locales, como SQLite, u otros mecanismos que nos proveen los sistemas operativos. Pero la fragmentación trae consigo que cada plataforma presente diferentes opciones de persistencia, y a su vez, maneje de forma diferente los distintos mecanismos de persistencia, dificultando esta tarea. En las siguientes subsecciones estaremos viendo y analizando cuáles son estas opciones.

La persistencia de datos en los teléfonos móviles inteligentes es una realidad que cada vez más impone su importancia en la fase de desarrollo de aplicaciones, puesto que cada vez es más necesario tratar y tener disponibles datos en diversas situaciones. A nivel código, la cantidad requerida para el manejo de todas las operaciones, constituye una porción significante de la aplicación entera. Alrededor del 21 % del código de la aplicación se destina a las conexiones, a los modelos de datos y a la configuración de las distintas fuentes de datos (tanto para el almacenamiento local como para recibir e

intercambiar datos con fuentes externas). Esta cantidad de código implica una cantidad de esfuerzo para desarrollarlo, y sería costoso realizar cambios posteriores, por lo que tener en cuenta la persistencia en etapas tempranas de desarrollo sería importante [10]. Para esto, es necesario llevar a cabo un análisis del modelo de datos a utilizar en la aplicación, así mismo, conocer y estar al tanto de los distintos mecanismos de persistencia disponibles a fin de utilizarlos adecuadamente.

A continuación se detallan las distintas opciones de almacenamiento disponibles actualmente para los teléfonos móviles inteligentes.

2.3.1. Almacenamiento en teléfonos móviles

Mahmoud et al. [12] y Fotache et al.[9] señalan las opciones de software más comunes para almacenamiento de datos en las aplicaciones móviles, en las distintas plataformas móviles:

- **HTML5**: permite el desarrollo de aplicaciones multiplataforma usando tecnologías web. Provee de un API para almacenamiento, siendo el intérprete Javascript del navegador el intermediario. Algunos métodos de almacenamiento son:
  - ◇ Almacenamiento de objetos por aplicación o por dominio web (*Local Storage* y *Global Storage*). El más utilizado es el *localstorage API*, el cual permite almacenar pares clave – valor, usando cadenas. Con esta opción no es posible almacenar construcciones de datos muy complejas, pero existe la opción de almacenar datos en forma de cadena con el método JSON.stringify ().
  - ◇ IndexedDB, el cual provee una implementación de una base de datos relacional a la aplicación, con la posibilidad de hacer consultas SQL.

  Así mismo, HTML5 provee ahora la modalidad "sin conexión" para la aplicación (utilizado para geolocalización, gráficos canvas y reproducción de audio/vídeo) lo que permite correr las aplicaciones web cuando no hay una conexión de red activa, y para almacenar bases de datos persistentes en el propio navegador web (por ejemplo, Gmail de Google) [11].

  Uno de los aspectos a tener en cuenta con HTML5, al momento de utilizarlo en el desarrollo, es que las diferentes plataformas utilizan diferentes motores en los navegadores para implementar esta tecnología.
- **Almacenamiento en la nube**: es una amplia colección de servicios web como Apple iCloud, Dropbox, Google Drive, Mega, Amazon S3, Ubuntu Cloud, entre otros.
  Estos servicios proveen una gran cantidad de espacio accesible, pero la latencia para su acceso es relativamente alta en comparación con el acceso local. En los dispositivos móviles se utilizan llamadas asíncronas para paliar este problema.
- **SQLite**: es una de las opciones de almacenamiento más populares disponibles en el mercado en la actualidad. Constituye una de las bases de datos para móviles con más atractivo para los desarrolladores: es simple, multiplataforma, fácil de instalar y configurar, no requiere de un servidor, es compacto, portable y de dominio público. Comprende una librería que encapsula funcionalidades SQL. Se instala localmente en el dispositivo en lugar de una conexión a una base de datos remota.
  Muchas aplicaciones en distintas plataformas utilizan SQLite para almacenar datos estructurados en una base de datos privada, desde perfiles de usuario a ajustes de configuraciones. Además, resulta muy adecuado para propósitos de testing y desarrollo.

Las opciones presentadas podemos clasificarlas en:

- **almacenamiento local**: entre los mecanismos de almacenamiento locales más comunes encontramos: i) pares (clave - valor), también conocidos como diccionarios y listas; ii) manejo de archivos; iii) almacenamiento en dispositivos externos (por ejemplo, tarjetas SD); y iv) bases de datos integradas;
- **almacenamiento remoto**: contamos con el almacenamiento en la nube (ya sea en servicios web ofrecidos por empresas, o base de datos alojadas en servidores remotos).

Todas las opciones de almacenamiento mencionadas tienen soporte en las plataformas móviles; pero, es importante notar que en cada una de ellas, existen variaciones en el manejo de estas opciones. A continuación veremos más detalles de estas opciones en algunas plataformas.

En el **cuadro 2**, a modo comparación, resumimos cómo manejan las plataformas más populares las distintas opciones de persistencia[4] [12]. Para más información sobre el manejo de la persistencia en cada plataforma ver: Android (`http://ow.ly/4mUW08`), iOS (`http://ow.ly/4mUVKE`) y Windows Phone (`http://ow.ly/4mUVTw`).

Plataformas	Par (clave, valor)	Manejo de archivos locales	Soporte para almacenamiento externo	Base de datos
Android	SharedPreferences, Bundles	Java Files Stream	Sí	Content Provider, SQLite
iOS	NSUserDefaults, Property lists	NSFileManager	Sí	Core Data, SQLite
Windows Phone	Isolated Storage Settings	Isolated Storage File System	Sí	Isolated Storage, SQLite

Cuadro 2. Mecanismos de persistencia de datos en sistemas operativos móviles

Según la página oficial de desarrollo de iOS (`http://ow.ly/4mWC7j`), *"la persistencia de datos es uno de los problemas más importantes y comunes en el desarrollo de aplicaciones iOS. iOS tiene muchas soluciones de almacenamiento de datos persistente"*. Esto resume lo que vimos anteriormente, y lo mismo sucede con Android, Windows Phone y otras plataformas: ante tanta variedad de soluciones, cada plataforma adopta estos mecanismos según funciones específicas; notándose así, el problema que ocasiona la fragmentación en este aspecto en particular.

2.3.2. Base de datos en móviles

En la actualidad, la mayoría de las aplicaciones y servicios (juegos, *m-commerce* o comercio móvil, *e-commerce* o comercio eléctronico y aplicaciones dirigidas por datos) necesitan tener soporte con bases de datos. Este ambiente de computación móvil a menudo involucra un sistema formado por la comunicación entre una base de datos central y la base

de datos móvil, involucrando tareas como sincronización y consultas en ambas direcciones (de la base de datos central a la base de datos móvil, y viceversa). Las aplicaciones móviles no necesitan estar conectadas a la base de datos central si la información ya está almacenada en el teléfono; de allí radica la importancia de las bases de datos móviles [31].

Las bases de datos móviles integradas constituyen uno de los mecanismos de persistencia de datos más utilizados, son forma de almacenamiento en caché. Estas bases de datos residen localmente en dispositivos como los teléfonos móviles inteligentes, computadoras portátiles, tabletas, entre otros. Están diseñadas para trabajar en un ambiente donde los recursos son limitados (memoria, batería, capacidad de computación) y de constante movilidad y desconexiones. Son capaces de comunicarse con una base de datos central u otros dispositivos móviles remotos. Una de sus principales tareas, además de almacenar datos del usuario y de la aplicación (por ejemplo, datos de un formulario, datos de inicio de sesión, datos de configuración de la aplicación) y guardar multimedia (imágenes, música, vídeo), es permitir el respaldo de archivos y de otras configuraciones, que permiten el manejo de queries de forma local y sin conexión [31] [32].

Las principales ventajas de usar bases de datos móviles son [33]:

- Acceso de datos sin conexión, donde el usuario puede leer y actualizar datos sin necesidad de una conexión de red.
- Supera problemas como pérdida de conexión, bajo ancho de banda, alta latencia y otros problemas comunes con las redes wireless. El no usar conexión de red en todo momento, supone un incremento de la vida de la batería del teléfono, así como una reducción en la tarifa de conexión inalámbrica (servicio de conexión 3G por ejemplo).
- Con la sola necesidad de sincronizar los datos actualizados, se libera una cantidad innecesaria de intercambio de datos entre la base de datos móvil y la central, reduciendo el tráfico y mejorando la velocidad de uso de la aplicación móvil.

Base de datos	Tipo	Tipo de datos almacenados	Licencia	Plataformas soportadas
BerkeleyDB	relacional / NoSQL	objetos, pares clave - valor, documentos	AGPL 3.0	Android, iOS
Couchbase Lite	NoSQL	documentos	Apache 2.0	Android, iOS
LevelDB	NoSQL	pares clave - valor	New BSD	Android, iOS
SQLite	relacional	NULL, Real, Entero, Texto, Blob	Dominio público	Android, iOS, Windows Phone, Blackberry
UnQLite	NoSQL	pares clave - valor, documentos	BSD 2-Clause	Android, iOS, Windows Phone

Cuadro 3. Comparación de bases de datos móviles integradas actuales

Como observamos en el **cuadro 3**, las bases de datos integradas móviles más populares (`http://ow.ly/4mUVz0`) son: Berkeley DB (`http://ow.ly/4mUVs9`), Couchbase Lite (`http://ow.ly/4mUWiC`), LevelDB (`https://github.com/google/leveldb`), SQLite y UnQLite (`http://unqlite.org/`). Entre estas resalta SQLite, el cual es el motor de base de datos más utilizado actualmente (`https://sqlite.org/mostdeployed.html`), compatible y soportado por distintos sistemas operativos móviles como Android, iOS y Windows Phone. Su código fuente es de dominio público.

2.4. Desarrollo Dirigido por Modelos

La Ingeniería de Software Dirigida por Modelos (MDSE - *Model Driven Software Engineering*), puede ser definida como una metodología en la cual se aplican las ventajas del modelado en las actividades de ingeniería de software [16]. El Desarrollo Dirigido por Modelos (MDD) es la evolución natural de MDSE, enriquecida mediante el agregado de transformaciones automáticas entre modelos [15].

MDD es un paradigma de desarrollo de software cada vez más aceptado y usado por la comunidad. Promete mejorar el proceso de construcción de software basándose en un proceso guiado por modelos y soportado por potentes herramientas; además, asigna a los modelos un protagonismo que los hace tan importantes como el código fuente [15]. Permite mejorar las prácticas comunes de desarrollo de software, presentando ventajas tales como: i) **incremento en la productividad**: reduciendo los costos de desarrollo gracias a la generación automática del código y otros artefactos a partir del modelo, incrementando la productividad de los desarrolladores; ii) **adaptación a los cambios tecnológicos**: los modelos de alto nivel están libres de detalles de la implementación, lo que facilita la adaptación a los cambios que pueda sufrir la plataforma tecnológica subyacente o la arquitectura de la implementación; iii) **adaptación a los cambios en los requisitos**: con toda la información capturada en las transformaciones para generar los artefactos de implementación, el adaptarse a un cambio o adición de un requisito utilizando MDD, es una tarea sencilla mediante la re-utilización de dicha información; iv) **consistencia**: mediante la automatización, antes que la generación manual, MDD favorece la consistencia de los artefactos generados; v) **re-utilización**: en MDD la inversión está en el desarrollo de modelos y transformaciones. A medida que éstos se van re-usando, se gana en productividad y en adaptación a cambios de requisitos [14].

Dos propuestas muy relacionadas con los conceptos de MDD son MDA y DSM. El primero tiende a enfocarse en modelado bajo los estándares y guías establecidos por el Grupo de Gestión de Objetos (OMG - *Object Management Group* - `http://www.omg.org/gettingstarted/gettingstartedindex.htm`) . El segundo, sin seguir un estándar específico, permite modelar para un dominio específico.

Con MDA se define una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos, siguiendo el proceso MDD [15] [14]. Acorde con las directivas de la OMG, las dos motivaciones principales para MDA son la **interoperabilidad** (independencia de los fabricantes a través de la estandarización) y la **portabilidad** (independencia de plataforma) de los sistemas de software. Así mismo, promete mejorar la **mantenibilidad** de los sistemas a través de la separación de conceptos [14]. De esta manera, MDA propone una representación independiente de los aspectos del sistema mediante: i) Modelo Independiente de la Computación (CIM - *Computation Independent Model*), también conocido como modelo del dominio, el cual no muestra detalles de la estructura del sistema; ii) Modelo Independiente de la Plataforma (PIM - *Platform Independent Model*), es una vista con un alto nivel de abstracción, independiente de cualquier tecnología o lenguaje de implementación; iii) Modelo Específico de la Plataforma (PSM - *Platform Specific Model*), representa la proyección del PIM en una plataforma específica; y, el iv) Modelo de la Implementación (Código), constituye el último paso en el desarrollo: la transformación del PSM a código fuente [19] [15].

Por otra parte, el Modelado Específico del Dominio (DSM - *Domain Specific Modeling*) propone la idea de crear modelos específicos para un dominio o un área de interés de desarrollo en particular. Para ello se utiliza DSL que permiten

especificar la solución usando directamente conceptos del dominio del problema [15] [8]. Los DSL son utilizados para construir lenguajes de modelado, por lo general son gráficos. La principal diferencia del uso de modelos respecto a MDA, es que éstos no utilizan ningún estándar de la OMG para su infraestructura (no se basan en UML) [15].

La heterogeneidad en las plataformas móviles repercute en el esfuerzo de desarrollo de las aplicaciones móviles para cada plataforma [16] [17]. Desarrollar aplicaciones que comparten las mismas funcionalidades y comportamiento, pero que son destinados a diferentes plataformas, es un área adecuado para MDD [24]. Con MDD, los desarrolladores de aplicaciones, describen sus aplicaciones en un nivel alto de abstracción mediante modelos independientes de detalles específicos de la plataforma, tanto para la GUI, la lógica de negocios y el acceso de datos. Esta especificación es traducida a código para diferentes plataformas. Ésto, siguiendo el enfoque MDA, significa implementar PIM para los diversos aspectos de los móviles (modelos independientes de aspectos tecnológicos), y luego mediante transformaciones sucesivas, llegar al PSM para cada plataforma a la que se desea implementar la aplicación [14] [17] [8] [15]. Así, no tenemos solamente un código base multiplataforma (el modelo), sino también un incremento en el nivel de abstracción y ciclos rápidos de desarrollo.

Existen varias soluciones dirigidas por modelos aplicadas a móviles siguiendo específicamente este enfoque, a continuación veremos cuáles son algunas de ellas y realizaremos un análisis comparativo entre ellas.

2.4.1. Propuestas dirigidas por modelos para el desarrollo de aplicaciones móviles

Con el fin de analizar las propuestas MDD para el desarrollo de aplicaciones móviles, llevamos a cabo un estudio de la literatura disponible siguiendo algunas de las pautas propuestas para realizar mapeos sistemáticos de la literatura [34], sin seguir estrictamente ese método:

- La búsqueda fue realizada en bibliotecas digitales como IEEEXplore (`http://ieeexplore.ieee.org/Xplore/home.jsp`), ACM (`http://dl.acm.org/dl.cfm`), SpringerLink (`http://link.springer.com/`) y ResearchGate (`https://www.researchgate.net/`). De la misma forma, utilizamos el buscador de Google destinado a la búsqueda de literatura científica-académica, Google Scholar (`https://scholar.google.com/`).
- Nuestras cadenas de búsqueda estuvieron basadas en aspectos de nuestro interés: MDD, móviles, persistencia.
- Los criterios de exclusión aplicados a los trabajos revisados fueron: i) no tienen nada que ver con el proceso MDD y/o aplicado a móviles; ii) año de publicación a partir del 2005.

El principal objetivo del estudio realizado fue obtener una visión global de MDD en el marco del desarrollo de las aplicaciones móviles. Nos enfocamos en el desarrollo de aplicaciones móviles nativas, por ser éstas las principales víctimas de la fragmentación. Dentro de la variedad de aspectos que afecta la fragmentación en los teléfonos móviles, nos enfocamos en aspectos que nos interesan: la persistencia de datos. Como resultado del estudio realizado, en el **cuadro 4** encontramos la lista de 18 soluciones MDD para el desarrollo de aplicaciones móviles y sus respectivos aportes.

El **cuadro 5** compara cada propuesta estudiada, realizando una descripción breve, en base a diversos aspectos que nos parecen interesantes:

- A fin de analizar el enfoque MDD utilizado por cada propuesta, recopilamos los siguientes datos: i) qué aspectos de la aplicación móvil fueron tenidos en cuenta por cada propuesta en su desarrollo (GUI, lógica de negocios, datos, entre otros); ii) en qué se basaron para el modelado (por ejemplo, en perfiles UML, en un DSL, entre otros); iii) qué lenguaje de modelado fue utilizado (por ejemplo, UML, DSL, y otros); iv) qué tipo de lenguaje modelado representa (gráfico, textual); y por último, v) si se basaron en estándares de la OMG (MDA, UML y IFML).
- Aspectos del desarrollo para móviles fueron analizados mediante: i) cuáles fueron las plataformas de destino elegidas por cada herramienta y/o propuesta (por ejemplo, Android, iOS); ii) si se utilizó MVC como patrón de arquitectura; iii) si generaban aplicaciones nativas; iv) si estas aplicaciones generadas eran completas[7]; y por último, v) de qué tipo eran las aplicaciones generadas (por ejemplo, juegos, aplicaciones de negocio, entre otros).
- Respecto a la persistencia, consideramos los siguientes puntos, que son aspectos mencionados entre las propuestas estudiadas, y creemos que nos ayudarán a tener un panorama general del tema en cuestión: i) si se consideraron aspectos de persistencia en el modelado; ii) si soporta la modalidad "sin conexión"; y por último, iii) si contempla la conexión con alguna base de datos externa a la aplicación.
- Como analizamos propuestas de investigación al igual que nuestro PFC, nos pareció interesante averiguar: los métodos de evaluación utilizados, así como validaciones, y saber qué resultados obtuvieron en el desarrollo de sus propuestas.

Teniendo en cuenta estos criterios, a continuación analizaremos los resultados más relevantes obtenidos.

Respecto a la persistencia en móviles, qué propuestas de solución encontramos.

Considerando los criterios descritos para el análisis de este aspecto, mencionados anteriormente, a continuación analizamos los resultados encontrados.

i) Este punto nos brinda una visión del grado de detalle y consideración que tuvieron las propuestas respecto a la persistencia. Hablamos anteriormente de la importancia que representa tener en cuenta este aspecto en las aplicaciones móviles. Nos interesa saber qué tipo de técnicas u opciones de almacenamiento fueron utilizadas por las distintas propuestas.

- Encontramos que 6/18 propuestas (33 %) no especifican, según nuestro estudio, ni hacen mención o consideran la persistencia al momento de modelar; a su vez, 5/18 propuestas (28 %) no lo aplican, puesto que abarcan solamente aspectos de interfaz. Pero encontramos 7/18 propuestas (39 %) que de alguna u otra forma sí tienen en cuenta en el modelado algún aspecto de persistencia.
  Resumiendo algunas formas de persistencia encontradas, encontramos que P1, en el perfil UML móvil desarrollado (independiente de alguna plataforma), nombra un estereotipo *"Persistence"*, junto con un atributo *"Persistencetype"* para indicar el tipo de persistencia. Ese atributo es clave puesto que puede contemplar distintos tipos de persistencia, pero no habla de cuáles son esos tipos, ni hasta qué punto los tiene en cuenta. Por otra parte, P17 presenta en su perfil UML (específico para Windows) el estereotipo *"IsolateStorage"*, tratándose del almacenamiento específico de Windows Phone, pero no encontramos más información al respecto. Lo mismo pasa con P18, donde se tiene en cuenta

[7] Consideramos como completa una aplicación que al ser generada es totalmente funcional, es decir, en su desarrollo se tuvo en cuenta más aspectos que solamente la GUI, como por ejemplo lógica de negocios y datos.

el *ContentProvider* (específico de Android), indicando la forma en que se almacenan los datos. Como posibles valores tiene al *SharedPreferences*, *InternalStorage*, entre otros. P8, a su vez, hace uso de un campo *"Persistent"* en sus entidades para indicar que esa entidad debe ser persistida. Resulta muy interesante, como método de selección, qué campos o qué datos deben ser almacenados. En P13 encontramos un modelo dedicado a datos, donde se detalla la conexión con una fuente de datos (o *datasource*) local (se especifica el tipo de datos que tiene de entrada y de salida) o remoto (se especifica la URL de conexión). De la misma forma, encontramos que P14 y P15 especifican con un *"providerType"* este tipo de detalle de la fuente de datos, indicando que el almacenamiento local lo realiza por medio de archivos y el remoto, mediante conexión a un *backend*.

ii) Una conexión con una base de datos externa implica el intercambio de datos entre el dispositivo y una base de datos, lo que implicaría la necesidad de guardar datos localmente (el uso de la técnica de guardar datos en caché podría ser interesante como opción en caso que no haya conexión).

- Tenemos 5/18 propuestas (28 %) que contemplan la conexión con una base de datos externa, incluyendo en el modelo las configuraciones de conexión con la misma. Pero sin dar detalle de qué hace con la información recibida.

iii) La modalidad "sin conexión" puede realizarse con algún mecanismo que permita persistir datos en el teléfono (como base de datos y archivos), a fin de poder utilizarlos en caso que no haya conexión de red.

- Considerando las propuestas del punto ii), solamente P6 permite el soporte para la modalidad "sin conexión" y permite conexión externa a una base de datos (P6 sería la opción más robusta puesto que asegura que aún se podría contar con datos en el teléfono cuando no haya conexión). No pasa así con P7, P9, P11 y P13, los cuales no especifican si soportan ésta modalidad. Finalmente, P4 soporta la modalidad sin conexión, pero no especifica si permite o no conexión con bases de datos externas (aunque lo más razonable es pensar que sí).

En general fue muy poco lo que pudimos encontrar. El manejo de la persistencia es un tema no muy claro ni muy especificado entre las propuestas analizadas. Encontramos indicios de algunas formas de implementación de la persistencia, analizando los metamodelos y perfiles (si los presentaban). Pero somos conscientes de que capaz muchas propuestas abarcaban aspectos de los móviles a nivel PIM, por lo que la ausencia de detalles de persistencia a ese nivel tendría sentido, pero en base a lo poco recopilado, no podemos decir más.

Destacamos que P1 es el único que menciona la persistencia y da a entender que podría manejar distintos tipos de persistencia (por los estereotipos que encontramos en su perfil). Pero este no indica cuáles podrían ser estos tipos ni da más detalles al respecto. En este sentido encontramos una oportunidad de investigación: presentar un nivel de detalle de la persistencia, donde sean tenido en cuenta las distintas opciones y mecanismos de persistencia disponibles y utilizadas por los sistemas operativos.

Respecto a la utilización de estándares de la OMG.

Entre todas las propuestas con enfoque dirigido por modelos, tenemos que 6/18 propuestas (33 %) adoptan o se basan en estándares MDA, utilizando separación de conceptos y diseño mediante PIM y PSM; todas éstas utilizan UML como lenguaje de modelado, a excepción de P12, que utiliza lenguaje textual llamado AXIOM DSL. También, encontramos que P1, P17 y P18 utilizan UML como estándar. Por otra parte, existen propuestas basadas en la interfaz de las aplicaciones, las cuales utilizan el estándar IFML de la OMG para interfaces de sistemas, incluido móviles. Tenemos que P5 y P6 utilizan este estándar.

Respecto a las plataformas móviles, cuáles son las más utilizadas como plataforma destino.

La plataforma más utilizada por las propuestas es Android, utilizada por la mayoría para ser destino de ejemplos, casos de estudio y de evaluaciones. Como el desarrollo es multiplataforma, muchos casos de estudio generan a su vez para iOS, específicamente encontramos 7/18 propuestas (39 %) que generan para Android y iOS. Cabe resaltar que Windows Phone es también una plataforma muy utilizada, apareciendo en 4/18 propuestas (22 %) como plataforma de prueba.

Respecto a las aplicaciones generadas.

En este punto resaltaremos tres criterios: i) si la aplicación generada es nativa; ii) si la generación es completa; y iii) el tipo de aplicación resultante.

Respecto al punto i), casi todas las aplicaciones generadas son nativas, por lo que se puede ver una tendencia hacia este tipo de aplicación. En total tenemos a 9/18 propuestas (50 %) que son nativas; de las restantes, 3/18 (17 %) serían híbridas y los demás, no especificaron.

Para el punto ii), es muy importante notar las estrategias de generación que encontramos. Tenemos que 11/18 propuestas (61 %) sí generan aplicaciones completas. Entre estas, solamente P14 y P15 generan la aplicación en su totalidad, es decir, generan las estructuras del proyecto y la aplicación final, **sin necesidad de una compilación previa en un IDE**, como las demás. Esta propuesta es MD2, y afirma que el trabajo más costoso fue la generación para iOS, puesto que debían generar incluso archivos temporales para emular lo que la herramienta Xcode hace para poder compilar la aplicación.

Por último, en el punto iii) es importante aclarar algo que todas las propuestas decían: los DSL desarrollados tienen la capacidad y soporte para generar todo tipo de aplicaciones; pero, la complejidad de la aplicación es proporcional a la dificultad y el esfuerzo en el modelado y en la generación de código. Para la mayoría de los casos de estudio y evaluaciones, se optaron por aplicaciones que no requerían muchos gráficos ni mucha interacción con el usuario. Es así que tenemos que las aplicaciones orientadas por datos fue la más elegida, por un 6/18 propuestas (33 %).

Respecto a las evaluaciones llevadas a cabo en cada propuesta

En general todos los trabajos proponen un método de evaluación para sus propuestas, y todas con resultados positivos; solamente P6 y P13 no dan más detalles de la evaluación utilizada. Entre los métodos de evaluación utilizados tenemos:

- implementación de una misma aplicación en varias plataformas, verificando así la generación multiplataforma del modelo desarrollado; es el caso de P2, P10, P12 y P16.

- Otro muy utilizado es simplemente un caso de estudio demostrando la aplicabilidad de la propuesta en una aplicación; 11/18 propuestas (61 %) lo llevaron a cabo. Entre estas, P1 y P10 compararon la aplicación generada con una ya existente.
- Varias propuestas hicieron experimento con un grupo de personas. P5 trabajo con un grupo de desarrolladores, haciendo comparación de esfuerzo. P9 hizo lo mismo, pero fueron desarrolladores que probaron la herramienta quienes dieron sus evaluaciones positivas sobre ella. P11 llevó a cabo un estudio de aplicabilidad del DSL desarrollado, donde varios grupos de trabajo desarrollaron una aplicación de prueba, luego respondieron ciertas preguntas. Al final, la hipótesis de adopción del lenguaje fue aceptada tras los comentarios positivos de los usuarios.
- Resalta que P15 lleva a cabo una comparación por líneas de código de la aplicación generada. No nos parece una buena métrica de comparación, puesto que el esfuerzo es menor porque se generó código automáticamente, no porque haya menos líneas. Además, una aplicación desarrollada tradicionalmente, varía la cantidad de líneas según la habilidad del desarrollador.
- Las métricas más utilizadas fueron: comparación de esfuerzo (utilizando o no la propuesta), comparación de la aplicación generada con una ya existente, generación multiplataforma, experimento con un grupo de gente (tanto desarrolladores como gente ajena al área) utilizando la herramienta y recibiendo comentarios.

Nro	Título del documento	Año de publicación	Tipo de documento	Descripción
P1	A Model driven Approach to Generate Mobile Applications for Multiple Platforms [35]	2014	INPROCEEDINGS	Perfil UML para desarrollo de móviles multiplataforma. El enfoque se centra y permite generar código de lógica de negocios para varias plataformas. Desarrollo de prototipo de herramienta para la generación de código llamada MAG (Generador de Aplicaciones Móviles).
P2	Modelling and generating the user interface of mobile devices and web development with DSL [36]	2015	ARTICLE	Enfoque para el desarrollo de interfaz de usuario para aplicaciones móviles, aplicado a Android y a Java Server Faces Framework. Se define un lenguaje para desarrollo de interfaces gráficas, Technology Neutral DSL, con la intención de generar código nativo para varias plataformas.
P3	Generating Android graphical user interfaces using an MDA approach [37]	2014	INPROCEEDINGS	Enfoque para el desarrollo de interfaz de usuario para aplicaciones móviles, aplicado a Android. Se define un lenguaje para desarrollo de interfaces gráficas, Technology Neutral DSL, con la intención de generar código nativo para varias plataformas.
P4	Model Driven Development of Mobile Applications Allowing Role Driven Variants [38]	2014	INBOOK	Lenguaje de modelado para el desarrollo MDD de aplicaciones móviles según roles de usuario, basado en EMF. Prototipo de infraestructura MDD para implementación del lenguaje presentado, basado en GMF (editor gráfico) y generador de código.
P5	Extending the Interaction Flow Modeling Language IFML for Model Driven Development of Mobile Aplications Front End [39]	2014	INBOOK	Enfoque dirigido por modelos para el desarrollo de aplicaciones móviles, basadas en el estándar IFML. Se propone una extensión de IFML para móviles. Se presenta el desarrollo del prototipo de editor de modelos en Eclipse y el generador de código multiplataforma.
P6	Model-Driven Development Based on OMG's IFML with WebRatio Web and Mobile Platform [30]	2015	INBOOK	Propuesta de plataforma WebRatio para el desarrollo MDD de aplicaciones web y móviles basado en IFML.
P7	MobDSL A Domain Specific Language for multiple mobile platform deployment [40]	2010	INPROCEEDINGS	MobDSL para el desarrollo multiplataforma de aplicaciones móviles, basado en el lenguaje calculus.
P8	XIS-Mobile A DSL for Mobile Applications [7]	2014	INPROCEEDINGS	Especificación de aplicaciones móviles independientes de la plataforma, mediante lenguaje XIS-Mobile y framework, basados en perfiles UML
P9	Bootstrapping Mobile App Development [41]	2015	INPROCEEDINGS	Desarrollo de RAPPT, una herramienta que genera el esqueleto de aplicación móvil para Android, especificado mediante un DSL. Su objetivo es brindar mayor soporte para los desarrolladores y reducir la redundancia de tareas que se presentan utilizando los IDEs actuales.
P10	Model-Driven Design for the Development of Multi-Platform Smartphone Applications [21]	2013	ARTICLE	Metodología de Diseño Dirigido por Modelos para el desarrollo de aplicaciones móviles independientes de la plataforma específica. Utiliza perfil UML2 para el PIM.
P11	A GUI Modeling Language for Mobile Applications [18]	2015	INPROCEEDINGS	Método para modelar interfaces móviles, como parte de un futuro proyecto de desarrollo MDD para aplicaciones móviles. Se presenta el lenguaje desarrollado: MIM (Modelado de Interfaz Móvil), y se evalúa la factibilidad de aplicación de la propuesta.
P12	The AXIOM Model Framework Transforming Requirements to Native Code for Cross-platform Mobile Applications [28]	2014	INPROCEEDINGS	Enfoque dirigido por modelos para el desarrollo multiplataforma de aplicaciones móviles que usa Axiom DSL para definir aplicaciones independientes de la plataforma. Desarrollo de prototipo Axiom para la generación de aplicaciones.
P13	Yet Another DSL for Cross-Platforms Mobile Development [23]	2013	INPROCEEDINGS	Presentación de XMOB DSL (Technology Neutral DSL), enfocado en aplicaciones móviles y basado en MDA.
P14	Model-Driven Cross-Platform Apps Towards Business Practicability [26]	2015	INPROCEEDINGS	MD2 como una solución para cumplir requisitos típicos de aplicaciones empresariales.
P15	Cross-Platform Development of Business Apps with MD2 [22]	2013	INBOOK	MD2 es un *framework* para el desarrollo MDD multiplataforma. Consiste en un DSL para describir aplicaciones de negocio y a partir de ahí, mediante generadores, crear automáticamente aplicaciones iOS y Android.
P16	A MDA-Based Model-Driven Approach to Generate GUI for Mobile Applications [42]	2013	ARTICLE	Enfoque MDA para modelado de GUI móvil. Presenta una forma de diseñar utilizando UML.
P17	A UML Metamodel for Smart Device Application Modeling based on Windows Phone 7 Platform [43]	2011	INPROCEEDINGS	Metamodelo UML extendido para el desarrollo de aplicaciones para Windows Phone 7.
P18	Extending UML Metamodel for Android Application [44]	2012	INPROCEEDINGS	Metamodelo UML extendido para el desarrollo de aplicaciones para Android

Cuadro 4. Propuestas recopiladas para el desarrollo de aplicaciones móviles siguiendo el enfoque MDD.

Puntos resaltantes obtenidos:

- Sin dudar, la persistencia es un aspecto muy poco tenido en cuenta, así como lo estuvimos analizando anteriormente. Son pocos los trabajos que llegan a considerar la persistencia en el modelado. Consideramos la posibilidad de que

algunos trabajos capaz hayan presentado sus metamodelos a nivel PIM (siguiendo un enfoque MDA), por lo que tiene sentido no considerar aspectos de persistencia allí. Pero fue muy poca la información que encontramos. No encontramos propuestas que hagan mención de los distintos mecanismos de persistencia: crear de una base de datos local, manejo de archivos, pares clave - valor. Sí encontramos que algunas propuestas tienen en cuenta una posible conexión con bases de datos externas, pero sin hacer mención de cómo manejar esos datos locales. Rescatando algunas referencias de utilización de persistencia entre las propuestas, encontramos: indicar si se almacenamiento es local o remoto; o si una entidad o clase debía ser persistente o no. Pero como se puede notar no es suficiente el grado de detalle ni se indica los diferentes tipos de almacenamiento existentes.

- Otro punto interesante es el hecho de que Android y iOS constituyen las plataformas destino más elegidas para los casos de estudio y evaluaciones. Como tercera opción está Windows Phone.
- Las aplicaciones generadas son en gran mayoría nativas y son del tipo orientadas a datos.
- Además, casi todas las propuestas generan la aplicación teniendo en cuenta la estructura del proyecto según el IDE correspondiente a la plataforma elegida; mediante esto se puede realizar una compilación con el SDK respectivo y así obtener la aplicación ejecutable.
- Por último, el tipo de evaluación más visto es la de realizar una ilustración o caso de estudio de la propuesta presentada.

2.5. MoWebA

El Enfoque de Desarrollo Web Orientado en Modelos (MoWebA - *Model Oriented Web Approach*) se presenta como una metodología enfocada al desarrollo de aplicaciones web centrado en roles. Puede ser clasificado como una metodología orientada a hipertexto, la cual adopta los estándares que propone el enfoque MDA en cada fase. La separación de conceptos (contenido, navegación, procesos y presentación) ya identificados en otras propuestas, es considerada fuertemente y reforzada, añadiendo a los usuarios como un nuevo concepto de separación [19].

El enfoque MDA adoptado por MoWebA presenta tres abstracciones diferentes para el modelado: i) **el espacio del problema**, cubierto por los CIM y PIM; ii) **el espacio de la solución**, el cual es cubierto por los ASM y PSM; iii) **definición de código fuente**, cubierto por el Modelo Específico de la Implementación (ISM - *Implementation Specific Model*) y código manual.

Los principales **aportes** de MoWebA como metodología pueden resumirse en [19]: i) proveer un enfoque original de navegación más orientado a funciones que orientado a datos, por lo cual no presenta una conexión directamente con el modelo de datos subyacente; ii) el modelo de datos/conceptual subyacente ya no constituye el punto de partida para el proceso de modelado, en su lugar se utiliza el modelo navegacional con un enfoque más bien orientado al comportamiento; iii) la utilización del ASM como fase en el proceso de modelado y transformación que propone MoWebA.

Respecto a los puntos i) y ii), son varios los métodos web que definen la estructura navegacional como grafos, basados en el modelo conceptual, lo que implica: j) el nivel de granularidad de los elementos navegacionales están relacionados directamente a la estructura de los elementos (por ejemplo, clases); jj) la navegación es obtenida considerando la forma en que la información es estructurada (relaciones de clases, por ejemplo), no de la forma en que son accedidas. En cambio, en MoWebA la estructura de navegación es jerárquica (en forma de árbol), lo que permite modelar la estructura navegacional orientada a funciones, y generar niveles de exploración (menús y submenús). Esta navegación es definida considerando las unidades funcionales (casos de uso) como el nivel de granularidad, y los caminos de navegación son definidos considerando las relaciones entre los casos de uso, haciendo que la definición de la navegación se obtenga de la forma en que el usuario interactúa con el sistema [19].

Respecto al punto iii), ASM y PSM son dos modelos generados en forma semiautomática. ASM enriquece los modelos previos con información adicional relacionada a la arquitectura del sistema. PSM es orientada a refinar los modelos con información relacionada a la plataforma y al lenguaje seleccionado para el sistema final (Java, .NET, PostgreSQL, etc.). Ambos permiten la independencia de plataformas, favoreciendo la portabilidad [19].

A continuación analizaremos el proceso de modelado y transformación utilizado por esta metodología.

2.5.1. Proceso de modelado y transformación

El proceso de modelado consiste en la utilización y la especificación sistematizada de los modelos CIM, PIM y PSM descritas por MDA, y del modelo específico propuesto por MoWebA, ASM. Se establecen siete etapas: etapas del 1 al 6, donde se definen el CIM y PIM; y la etapa 7 donde se definen el ASM y PSM. A continuación haremos una breve descripción de cada una de las etapas.

En la **etapa 1** se realiza el análisis de requerimientos. Aquí los potenciales usuarios son identificados. Por cada potencial usuario se lista una serie de requerimientos funcionales, navegacionales y de usabilidad. Constituye el punto de partida para la definición del Árbol Navegacional. El diagrama producido en esta etapa es el Diagrama de Casos de Uso. La **etapa 2** significa la definición del Diagrama de Entidades, donde se organiza la información estructuralmente. Aquí se define el Árbol Navegacional. Los diagramas de Rol y Zonas son utilizados, considerando los usuarios potenciales de la etapa 1. Tenemos así que en esta etapa son producidos los diagramas de Entidades, el Árbol Navegacional y los diagramas de Rol y Zona. En la **etapa 3** se define el comportamiento de cada nodo del árbol navegacional a través del Diagrama de Nodos (representado por Diagrama de Estados de UML). La **etapa 4** se encarga de la presentación donde se definen qué elementos van a ser mostrados en cada página de presentación usando el Diagrama de Contenido. En esta etapa, las estructuras de las páginas (posiciones de la cabecera, menús, pie de página, etc.) son definidas a través del Diagrama de Estructuras. La composición estructural de los procesos de negocios y los procedimientos transaccionales también son definidos es ésta etapa con el Diagrama Lógico. De esta forma, los diagramas producidos son el de Contenido, de Estructura y el Lógico. Para la **etapa 5**, la personalización de los modelos es realizada a través del Modelo de Adaptación. En esta etapa, MoWebA propone definir los diagramas de Fuente y Reglas. En la **etapa 6** se propone una definición detallada de cada servicio o acción identificada en el Diagrama Lógico y de Contenido usando el Diagrama de Servicios. Por último, en la **etapa 7** se generan los diagramas ASM y PSM. Aquí se propone un enriquecimiento de los modelos existentes en orden a considerar aspectos relacionados a la arquitectura final del sistema y agregando información específica de la plataforma.

El proceso de transformación es basado en metamodelado (transformación de PIM a PSM/ASM). La fase PIM a ASM/PSM es hecho de manera (semi)automática, de hecho, la información a ser agregada algunas veces requiere intervención humana. La fase ASM a PSM-ISM es hecho automáticamente con herramientas *opensource* (por ejemplo, Acceleo y androMDA).

Propuestas basadas en MDD para el desarrollo de aplicaciones móviles															
Propuesta	**Aspectos considerados**	**En qué se basa para el modelado**	**Lenguaje de modelado**	**Tipo de lenguaje de modelado**	**Utiliza estándar OMG**	**Plataformas posibles de destino**	**Utiliza MVC**	**Generación de código nativo**	**Generación completa de la aplicación**	**Tipo de aplicación generada**	**Presenta evaluación de su propuesta**	**Sus evaluaciones dieron resultado positivos**	**Tiene en cuenta la persistencia en el modelado**	**Soporte modalidad "sin conexión"**	**Permite conexión con bases de datos**
P1	Estructural y lógica de negocios	Perfil UML	UML	Gráfico	UML	Android y Windows Phone[8]	No	Sí	Sí	Soporta el desarrollo de todo tipo de aplicaciones	Sí	Sí	Sí[9]	No especificado	No especificado
P2 y P3	GUI	DSL	UML y Technology Neutral DSL	Textual	MDA	Android[8]	No	Sí	No	No especificado	Sí	Sí	No aplica	No aplica	No aplica
P4	GUI y lógica de negocios	EMF	EMF	Gráfico	No	Android y iOS	Sí	Sí	Sí	Aplicaciones de negocio dirigidas por datos	Sí	Sí	No especificado	Sí	No especificado
P5	GUI y lógica de negocios	IFML	IFML para móvil	Gráfico	IFML	Android y iOS[8]	No	No, *native wrapper*	Sí	Aplicaciones móviles basadas en tecnologías web[10]	Sí	Sí	No aplica	No aplica	No especificado
P6	Estructural, lógica de negocios y GUI	IFML	IFML para móvil (interfaz) y Flujo Visual (*backend*)	Gráfico	IFML	Android y iOS	No	No, *native wrapper*	Sí	Aplicaciones móviles basadas en tecnologías web[10]	No	No aplica	No especificado	Sí	Sí
P7	No especificado	λ calculus	MobDSL	Textual	No	Android y iOS	No especificado	No, interpreta el código a través de la máquina virtual	Sí	Aplicaciones dirigidas por datos	Sí	Sí	No especificado	No especificado	Sí[11]
P8	Estructural, lógica de negocios y GUI	Perfil UML llamado Xis	Xis-Mobile DSL	Gráfico	No	Android, iOS y Windows Phone[8]	No	Sí	Sí	Aplicaciones dirigidas por datos	Sí	Sí	Sí[12]	No especificado	No especificado
P9	Datos, estructural, lógica de negocios, y GUI	DSL	RAPPT DSL	Gráfico/Textual[13]	No	Android	Sí	Sí	Sí	Aplicaciones dirigidas por datos	Sí	Sí	No especificado	No especificado	Sí
P10	Estructural, lógica de negocios y GUI	Perfil UML	UML	Gráfico	MDA	Android y Windows Phone.	Sí	Sí	Sí	Aplicaciones sencillas, que no requieran gráficos avanzados	Sí	Sí	No especificado	No especificado	No especificado
P11	GUI	MIM	MIM DSL	Gráfico	No	No especificado	No	No especificado	No[14]	Aplicaciones dirigidas por datos	Sí	Sí	No aplica	No especificado	Sí
P12	Requerimientos y lógica de negocios, y GUI	DSL	AXIOM DSL	Textual	MDA	Android y iOS[8]	Sí	Sí	Sí	Soporta el desarrollo de todo tipo de aplicaciones	Sí	Sí	No especificado	No especificado	No especificado
P13	Datos, GUI y lógica de negocios	DSL	Xmob DSL(PIM); perfil UML(PSM)	Textual	MDA	Android, iOS y .NET	Sí	Sí	Sí	Aplicaciones dirigidas por datos	No	No aplica	Sí[15]	No especificado	Sí
P14 y P15	Datos, GUI y lógica de negocios	DSL	MD2 DSL	Textual	No	Android y iOS	Sí	Sí	Sí[16]	Aplicaciones dirigidas por datos	Sí	Sí	Sí[17]	No especificado	No especificado
P16	GUI	UML	UML	Gráfico	MDA	Android y Blackberry	No	No especificado	No[14]	No especificado	Sí	Sí	No aplica	No aplica	No aplica
P17	Datos, GUI, lógica de negocios y recursos de hardware	Perfil UML	UML	Gráfico	UML	Windows Phone 7	Sí	No especificado	No especificado	No aplica	Sí	Sí	Sí[18]	No especificado	No especificado
P18	Datos, GUI, lógica de negocios y recursos de hardware	Perfil UML	UML	Gráfico	UML	Android	Sí	No especificado	No especificado	No aplica	Sí	Sí	Sí[19]	No especificado	No especificado

Cuadro 5. Cuadro comparativo de las propuestas MDD resultantes del estudio de la literatura

[8] El lenguaje soporta cualquier plataforma, pero la herramienta desarrollada tiene como objetivo desarrollar solo estas plataformas.
[9] Se nombra un estereotipo *"Persistence"* en el perfil UML. Se usa un campo *"Persistencetype"* para indicar el tipo de persistencia. Se usa como ej. para modelar la clase *DBHandler* o manejador de la base de datos.
[10] Basadas en HTML5, CSS y Javascript. Optimizado para el framework Apache Cordova.
[11] El caso de estudio se realiza con el desarrollo de una aplicación con soporte para bases de datos SQLite. Como los resultados fueron positivos, creemos que es factible la utilización de bases de datos.
[12] Las entidades poseen un campo boolean *"Persistent"*, que indica si deben o no ser persistentes.
[13] Gráfico, modelado de la navegación de pantallas de la aplicación. Textual, para la descripción y diseño de cada pantalla.
[14] Generación parcial de la aplicación, solo la GUI.
[15] Con el sublenguaje XMOB-data tiene en cuenta el mecanismo de datos a utilizar, especificamente si se trata o no de consumir datos de un servicio. Se especifica el *bean* y *datasource*, y el tipo de dato de retorno.
[16] Genera totalmente la aplicación, sin necesidad de compilar el código generado en un IDE (genera los archivos temporales necesarios para emular los creados durante la compilación en el IDE).
[17] Utiliza el contentProvider / ORM como capa de acceso a los datos. Se realizan acciones CRUD para persistir datos. Se especifica con *"providerType"* el tipo de almacenado, ya sea local (archivos) o remoto (*backend*).
[18] En el metamodelo se contempla el *"Isolate Storage"* el cual es espacio independiente de almacenamiento para almacenar una aplicación ya sea temporalmente o para persistirlo.
[19] En el metamodelo se tiene en cuenta el *ContentProvider*, específico de Android, indicando la forma en que se almacenan los datos. Como posibles valores tiene al *SharedPreferences*, *InternalStorage*, entre otros.

3. Propuesta

En esta sección hablaremos sobre la propuesta que desarrollaremos en base a la problemática y a todo el estudio realizado en las secciones anteriores. Haremos un breve resumen de la problemática descrita a lo largo del documento y plantearemos una hipótesis de solución para ello. Así mismo, estableceremos los límites de este PFC e indicaremos las actividades realizadas y las pendientes.

3.1. Problemática

Cuando se desea desarrollar aplicaciones para varias plataformas, la fragmentación supone un incremento en el tiempo de desarrollo y en el costo de mantenimiento de las aplicaciones. Como vimos, entre los enfoques de desarrollo, es el desarrollo nativo de aplicaciones móviles el principal afectado por este fenómeno.

La fragmentación puede afectar distintos aspectos del desarrollo de la aplicación: la GUI, que se ve afectada ante tanta variedad de tamaño de pantallas, de resoluciones de los teléfonos móviles inteligentes; y, la persistencia de datos, ante las opciones de almacenamiento disponibles, diferentes en cada plataforma.

Como vimos, la naturaleza transitoria de los teléfonos móviles inteligentes obliga a que la aplicación cuente con algún método de persistencia en caso que no se cuente con conexión de red. Además, existen aplicaciones que requieren de almacenamiento de datos y de conexión con datos remotos (por ejemplo, las aplicaciones dirigidas por datos). La persistencia de datos puede darse de distintas formas, y a su vez, con distintos mecanismos dependiendo del sistema operativo (**sección 2.3.1**).

Gracias a la investigación realizada en este trabajo (**sección 2.4.1**), hemos podido notar que la persistencia de datos no es abordada o no tiene el suficiente nivel de detalle. Siguiendo el enfoque MDA, puede considerarse la posibilidad de que en el estudio realizado, muchos trabajos hayan presentado sus propuestas a nivel PIM, por lo que no tener en cuenta aspectos de persistencia a este nivel, tiene sentido. Pero por lo poco que pudimos recabar, no podemos dar más que suposiciones al respecto.

El enfoque dirigido por modelos fue presentado como posible herramienta de solución a estas problemáticas. La necesidad de aprendizaje de un nuevo lenguaje de modelado (en caso de los DSL) resultó ser uno de sus principales incovenientes.

Con nuestra propuesta pretendemos abarcar las problemáticas mencionadas. A continuación describiremos nuestra propuesta de solución.

3.2. Propuesta de la solución

Planteamos desarrollar una propuesta MDD que permita considerar aspectos de la persistencia de datos en el desarrollo de aplicaciones móviles nativas. Para ello, utilizaremos la propuesta MoWebA. Creemos que esta metodología podría constituirse en una opción adecuada para el entorno móvil, gracias a su estructura por capas bien definida (mediante la cual se logra una clara separación de conceptos) y el modelado centrado en la navegación jerárquica orientada a funciones. Además, MoWebA está basado en UML, evitándonos la necesidad de aprender un lenguaje de modelado nuevo. En este sentido, MoWebA nos proporciona:

- un esquema más adecuado para el diseño de una navegación basada en la interacción del usuario o del contexto [19], al considerar que la manera en la cual la información es organizada y estructurada dentro del sistema no es necesariamente la misma en la que los usuarios acceden a ella. Como vimos, las aplicaciones móviles son dirigidas por eventos, así que este diseño de navegación encajaría muy bien con nuestra propuesta.
- un enriquecimiento de los modelos existentes en orden a considerar aspectos relacionados a la arquitectura final del sistema, gracias a su ASM. Mediante esta capacidad se podría pensar en la posibilidad de representar aplicaciones móviles, permitiendo utilizar conceptos específicos de la arquitectura y plataformas móviles [19].

Atendiendo a lo dicho anteriormente, proponemos la extensión de MoWebA para que contemple el desarrollo de aplicaciones móviles considerando los aspectos necesarios para tener en cuenta el diseño de la persistencia de datos en teléfonos móviles inteligentes.

Utilizaremos metamodelos y perfiles que nos permitan definir un ASM para móviles, la cual nos permitirá modelar las nuevas funcionalidades y mediante reglas de transformación generar código funcional (de ser necesario se realizaran ajustes manuales para lograr la aplicación móvil final).

Siguiendo con el estándar MDA, adoptado por MoWebA, para los metamodelos utilizaremos lenguaje de metamodelado MOF y los perfiles se expresarán mediante UML. Las reglas de transformación serán elaboradas utilizando la herramienta Acceleo.

En base a estos puntos, creemos que el manejo de la persistencia puede ser adecuado abarcando puntos analizados en nuestro estudio de propuestas. A continuación especificaremos el alcance de nuestro PFC.

3.3. Alcance del PFC

En el desarrollo de nuestra propuesta tendríamos en cuenta los siguientes aspectos de persistencia: i) posibilidad de utilizar las distintas opciones y mecanismos de persistencia analizados; ii) posibilidad de modelar la base de datos local para el teléfono móvil; iii) configuración de la *datasource* de conexión a una base de datos externa para poder tratar de forma local los datos recibidos (almacenamiento local de datos); y relacionado al punto anterior, iv) brindar la posibilidad de utilizar la aplicación en modo "sin conexión".

Para lograr los puntos mencionados, todos requieren de la extensión de MoWebA para su representación. Este MoWebA móvil lograría diseñar y tener una visión global de la aplicación a desarrollar, considerando aspectos de persistencia ya desde nivel ASM.

Para el punto i), mencionado anteriormente, tendríamos en cuenta los mecanismos de persistencia resultantes del estudio comparativo del manejo de datos en plataformas como Android, iOS y Windows Phone: pares clave-valor, manejo de archivos locales, soporte para almacenamiento externo y base de datos. Estas opciones lo integraríamos a nivel ASM.

Para el punto ii), teniendo en cuenta las bases de datos, brindaríamos la posibilidad de modelar una. Específicamente nos interesan las bases de datos móviles integradas, siendo SQLite nuestro objetivo, puesto que es una de las bases de datos más populares y utilizadas en la actualidad. Es así, que utilizaremos el Modelo de Entidades de MoWebA para poder representar la estructura de datos a utilizar (quedaría verificar si serán necesarios hacer unas extensiones, pero lo más seguro que no). Obtendríamos el código SQLite con la estructura de la base de datos, ya en la plataforma correspondiente, mediante reglas de transformación.

Para el punto iii) necesitamos considerar unos aspectos de configuración para poder consumir datos de una fuente de datos, en este caso una base de datos remota. Tendríamos que especificar la URL y los datos a obtener. El objetivo es obtener estos datos de forma local, almacenándolos o simplemente utilizándolos para mostrar datos en una pantalla en particular. En caso de almacenarlos, estaríamos guardando en caché. Esto último da lugar a nuestro último punto iv), pues si podemos retener datos provenientes de una conexión a una base de datos remota, estaríamos permitiendo que la aplicación trabaje con los datos guardados y sin conexión, utilizando algún mecanismo de persistencia mencionado.

A partir de los modelos generados, pretendemos generar código funcional utilizando reglas de transformación, teniendo en cuenta los siguientes aspectos: j) generación de código nativo para al menos dos plataformas, Android y Windows Phone; jj) generación de aplicaciones orientadas a datos; y jjj) generar aplicaciones completas mediante la integración del código generado (abarcando solamente los aspectos de la persistencia), con la interfaz de la aplicación y el resto de la lógica que se necesite **generados con alguna herramienta o manualmente**.

Por últimos, validaríamos nuestra propuesta mediante una ilustración, que permita desarrollar aplicaciones orientadas a datos (aplicación recolectora de datos o aplicación de manejo de compra productos online), considerando las distintas opciones de persistencia para varias plataformas; o bien, mediante una experiencia empírica. Las métricas de valoración de uno u otro método aún quedan por ser definidas.

4. Avances

En el cuadro 6 listamos las principales actividades planificadas para el desarrollo del presente PFC, junto con el estado de avance en que se encuentra cada una de ellas. A continuación analizaremos el cuadro, detallando las actividades involucradas en cada aspecto.

Nro	Actividad	Estado
A1	Revisión bibliográfica y recopilación de información.	Finalizado
A2	Evaluación y análisis de las propuestas MDD para el desarrollo de aplicaciones móviles.	Finalizado
A3	Estudio y análisis de los diagramas de MoWebA para la identificación de aquellos a utilizar y extender.	En progreso
A4	Estudio de lenguajes de metamodelado y reglas de transformación.	En progreso
A5	Desarrollo de metamodelos y perfiles.	En progreso
A6	Definición y validación de las reglas de transformación.	No iniciado
A7	Validación empírica y análisis de resultados.	No iniciado
A8	Elaboración del libro de tesis	En progreso

Cuadro 6. Estado de las actividades planificadas para este PFC

Actividades realizadas

Para A1 fue necesario un análisis preliminar del ambiente móvil en búsqueda de aspectos problemáticos. Se identificaron dos posibles aspectos que requerían de mejoras: interfaz y persistencia de datos. Como parte del PFC, se elaboró la propuesta para el PFC enfocados en la persistencia de datos como aspecto a analizar. En A2 se llevaron a cabo estudios del estado del arte sobre las aplicaciones móviles, donde analizamos: por una parte, las herramientas de desarrollo de aplicaciones móviles existentes; y luego, enfocados en las propuestas MDD, observamos la persistencia de datos en el desarrollo de aplicaciones móviles.

Actividades en progreso

A modo de definir nuestra propuesta, se llevaron a cabo A3, A4 y A5. Queda pendiente un análisis más detallado de los diagramas a extender y presentar una versión pulida de los metamodelos y perfiles, a utilizar posteriormente en el desarrollo de las reglas de transformación. De la misma forma, A7 fue iniciado, utilizando los documentos e investigaciones ya realizadas, para la preparación de las primeros capítulos del libro.

Actividades no iniciadas

Una vez que tengamos una versión más robusta y terminada de nuestros metamodelos y perfiles, llevaremos a cabo A6. Con las reglas de transformación terminadas, pasaremos a A7, donde realizaremos nuestras validaciones y experiencias empíricas.

Referencias

1. Redda, Y.A.: Cross platform mobile applications development. Norwegian University of Science and Technology, Norwegian University of Science and Technology, Norwegian University of Science and Technology, Master in Information Systems (2012)
2. Sommer, A., Krusche, S.: Evaluation of cross-platform frameworks for mobile applications. In: Software Engineering (Workshops). (2013) 363–376
3. eMarketer: Global media intelligence report executive summary. `https://www.emarketer.com/public_media/docs/GMI-2015-ExecutiveSummary.pdf` (2015)
4. Grønli, T.M., Hansen, J., Ghinea, G., Younas, M.: Mobile application platform heterogeneity: Android vs windows phone vs ios vs firefox os. In: Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, IEEE (2014) 635–641

5. Vique, R.R.: Métodos para el desarrollo de aplicaciones móviles. PID_00176755 (2012)
6. Marinho, E.H., Resende, R.F.: Native and multiple targeted mobile applications. In: Computational Science and Its Applications–ICCSA 2015. Springer (2015) 544–558
7. Ribeiro, A., da Silva, A.R.: Xis-mobile: A dsl for mobile applications. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing. SAC '14, New York, NY, USA, ACM (2014) 1316–1323
8. Ribeiro, A., da Silva, A.R.: Survey on cross-platforms and languages for mobile apps. In: Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the, IEEE (2012) 255–260
9. Fotache, M., Cogean, D., et al.: Nosql and sql databases for mobile applications. case study: Mongodb versus postgresql. Informatica Economica **17**(2) (2013) 41–58
10. Melman, C.: A generative approach for data synchronization between web and mobile applications. Master thesis, Electrical Engineering, Mathematics and Computer Science (2014)
11. Ferreira, J.A.L., da Silva, A.R.: Mobile cloud computing. Open Journal of Mobile Computing and Cloud Computing **1**(2) (2014)
12. Mahmoud, Q.H., Zanin, S., Ngo, T.: Integrating mobile storage into database systems courses. In: Proceedings of the 13th Annual Conference on Information Technology Education. SIGITE '12, New York, NY, USA, ACM (2012) 165–170
13. Xanthopoulos, S., Xinogalos, S.: A comparative analysis of cross-platform development approaches for mobile applications. In: Proceedings of the 6th Balkan Conference in Informatics. BCI '13, New York, NY, USA, ACM (2013) 213–220
14. Stahl, T., Voelter, M., Czarnecki, K.: Model-driven software development: technology, engineering, management. John Wiley & Sons (2006)
15. Pons, C., Giandini, R., Perez, G.: Desarrollo de software dirigido por modelos : conceptos teoricos y su aplicacion practica. 1era edn. Mc Graw Hill Educacion (2010)
16. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers (2012)
17. Balagtas-Fernandez, F.T., Hussmann, H.: Model-driven development of mobile applications. In: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering. ASE '08, Washington, DC, USA, IEEE Computer Society (2008) 509–512
18. Geiger-Prat, S., MarThín, B., España, S., Giachetti, G.: A gui modeling language for mobile applications. In: 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS). (2015) 76–87
19. González, M., Cernuzzi, L., Pastor, O.: A Navigational Role-Centric Model Oriented Web Aproach - MoWebA. International Journal of Web Engineering and Technology (IJWET) **11**(1) (2016) (in press).
20. Barnett, S., Vasa, R., Tang, A.: A conceptual model for architecting mobile applications. In: Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on. (2015) 105–114
21. Botturi, G., Ebeid, E., Fummi, F., Quaglia, D.: Model-driven design for the development of multi-platform smartphone applications. In: Specification Design Languages (FDL), 2013 Forum on. (2013) 1–8
22. Heitkötter, H., Majchrzak, T.A., Kuchen, H.: Cross-platform model-driven development of mobile applications with md 2. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM (2013) 526–533
23. Le Goaer, O., Waltham, S.: Yet another dsl for cross-platforms mobile development. In: Proceedings of the First Workshop on the Globalization of Domain Specific Languages. GlobalDSL '13, New York, NY, USA, ACM (2013) 28–33
24. Heitkötter, H., Majchrzak, T.A.: Cross-Platform Development of Business Apps with MD2. In: Design Science at the Intersection of Physical and Virtual Design: 8th International Conference, DESRIST 2013, Helsinki, Finland, June 11-12, 2013. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2013) 405–411
25. Alamri, H.S., Mustafa, B.A.: Software engineering challenges in multi platform mobile application development. Advanced Science Letters **20**(10-12) (2014) 2115–2118
26. Majchrzak, T.A., Ernsting, J., Kuchen, H.: Model-driven cross-platform apps: Towards business practicability. In: Proc. of the 27th Conference on Advanced Information Systems Engineering (CAiSE) Forum. CEUR. (2015)
27. Balagtas-Fernandez, F., Tafelmayer, M., Hussmann, H.: Mobia modeler: easing the creation process of mobile applications for non-technical users. In: Proceedings of the 15th international conference on Intelligent user interfaces, ACM (2010) 269–272
28. Jones, C., Jia, X.: The axiom model framework: Transforming requirements to native code for cross-platform mobile applications. In: Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on. (2014) 1–12
29. Hemel, Z., Visser, E.: Declaratively programming the mobile web with Mobl. Volume 46. ACM (2011)
30. Acerbis, R., Bongio, A., Brambilla, M., Butti, S.: Model-Driven Development Based on OMG's IFML with WebRatio Web and Mobile Platform. In: Engineering the Web in the Big Data Era: 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings. Springer International Publishing, Cham (2015) 605–608
31. IbikunleF.A, A.A, A.: Management issues and challenges in mobile database system. International Journal of Engineering Sciences & Emerging Technologies **5** (2013) 6
32. Nori, A.K.: Mobile and embedded databases, sigmod 2007. In: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Beijing, China, disponible en: http://citeseerx. ist. psu. edu/viewdoc/summary. (2007)
33. Yu, W., Amjad, T., Goel, H., Talawat, T.: An approach of mobile database design methodology for mobile software solutions. In: Grid and Pervasive Computing Workshops, 2008. GPC Workshops '08. The 3rd International Conference on. (2008) 138–144
34. Keele, S.: Guidelines for performing systematic literature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE. (2007)
35. Usman, M., Iqbal, M., Khan, M.: A model-driven approach to generate mobile applications for multiple platforms. In: Software Engineering Conference (APSEC), 2014 21st Asia-Pacific. Volume 1. (2014) 111–118
36. LACHGAR, M., ABDALI, A.: Modeling and generating the user interface of mobile devices and web development with dsl. Journal of Theoretical & Applied Information Technology **72**(1) (2015)
37. Lachgar, M., Abdali, A.: Generating android graphical user interfaces using an mda approach. In: 2014 Third IEEE International Colloquium in Information Science and Technology (CIST). (2014) 80–85
38. Vaupel, S., Taentzer, G., Harries, J.P., Stroh, R., Gerlach, R., Guckert, M.: Model-Driven Development of Mobile Applications Allowing Role-Driven Variants. In: Model-Driven Engineering Languages and Systems: 17th International Conference, MODELS 2014, Valencia, Spain, September 28 – October 3, 2014. Proceedings. Springer International Publishing, Cham (2014) 1–17
39. Brambilla, M., Mauri, A., Umuhoza, E.: Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End. In: Mobile Web Information Systems: 11th International Conference, MobiWIS 2014, Barcelona, Spain, August 27-29, 2014. Proceedings. Springer International Publishing, Cham (2014) 176–191
40. Kramer, D., Clark, T., Oussena, S.: Mobdsl: A domain specific language for multiple mobile platform deployment. In: Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on. (2010) 1–7
41. Barnett, S., Vasa, R., Grundy, J.: Bootstrapping mobile app development. In: Proceedings of the 37th International Conference on Software Engineering - Volume 2. ICSE '15, Piscataway, NJ, USA, IEEE Press (2015) 657–660
42. Sabraoui, A., El Koutbi, M., Khriss, I.: A mda-based model-driven approach to generate gui for mobile applications. International Review on Computers and Software (IRECOS) **8**(3) (2013) 845–852
43. Min, B.K., Ko, M., Seo, Y., Kuk, S., Kim, H.S.: A uml metamodel for smart device application modeling based on windows phone 7 platform. In: TENCON 2011 - 2011 IEEE Region 10 Conference. (2011) 201–205
44. Ko, M., Seo, Y.J., Min, B.K., Kuk, S., Kim, H.S.: Extending uml meta-model for android application. In: Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on. (2012) 669–674