UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTENDIENDO LAS REDES SOCIALES HACIA UN PLANO FÍSICO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

CAMILO IGNACIO VERGARA ACUÑA

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:

CLAUDIO GUTIERREZ GALLARDO

MARCELA CALDERÓN CORAIL

SANTIAGO DE CHILE
ABRIL 2012

Resumen Ejecutivo

La popularidad y cantidad de redes sociales han aumentado considerablemente en los últimos años. Estas redes mantienen la interacción entre personas en un espacio virtual que es accesible sólo a través de medios computacionales.

Por otra parte, se asume que familiares y amigos buscan encontrarse cara a cara entre ellos, en lugar de mantener comunicaciones por medios digitales. En el caso de contextos laborales, existen tareas que ven incrementada su eficiencia cuando el equipo de trabajo mantiene una reunión física en lugar de virtual.

El desafío abordado en este trabajo de memoria consiste en tratar de llevar la interacción entre los miembros de redes sociales a un plano físico, entendiendo las ventajas que la interacción cara a cara tiene por sobre los medios digitales de comunicación.

La solución desarrollada consiste en un software que promueve las reuniones cara a cara entre personas, mediante la detección de la cercanía física entre ellas. El software permite al usuario importar sus contactos desde redes sociales tradicionales, comunicarse con quienes se encuentren cerca e invitarlos a una reunión presencial.

El software utiliza una red ad hoc móvil para llevar a cabo la detección de contactos que están físicamente cerca. A través del software desarrollado, los usuarios pueden administrar su privacidad dentro de dicha red. Por ejemplo, pueden configurar el nivel de visibilidad que tendrán sus dispositivos para los otros dispositivos que se encuentren en la cercanía. De esa manera un usuario puede estar visible para un grupo de usuarios (por ejemplo para sus amigos), e invisible para sus compañeros de trabajo.

Para llevar a cabo la administración de contactos, el software permite agruparlos (por ejemplo, grupo “familia”, “amigos”, etc.). El nivel de privacidad que un usuario configura para con un cierto grupo es heredado automáticamente por todos sus miembros.

La aplicación desarrollada está completamente funcional; sin embargo, no fue factible probarla con usuarios reales dentro de los plazos establecidos. Pese a lo anterior, la investigación preliminar y el diseño del software permiten esperar que la aplicación sea una herramienta útil para llevar a un plano físico las interacciones entre los usuarios de redes sociales. A futuro sería importante agregarle un *logfile* a la herramienta, para poder registrar las interacciones entre usuarios, y de esa manera determinar qué tan exitosa es en la promoción de las interacciones cara a cara.

Agradecimientos

Mis agradecimientos van para mi familia, por su tremendo cariño y apoyo desde siempre. A mis amigos, por acompañarme durante toda la carrera. A Margarita, por su amor, compañía y paciencia.

Especiales agradecimientos para Leticia, por ser la primera en creer en mí.

Este trabajo de memoria ha sido parcialmente apoyado por los proyectos Fondecyt Nº 1120207 y LACCIR R1210LAC002.

Tabla de Contenidos

Índice de Tablas

Índice de Figuras

1. Introducción

Desde su aparición, las IESN (Internet-enabled social network) han proliferado considerablemente, para dar soporte a grupos sociales de distinta índole, los cuales buscan satisfacer diferentes necesidades de comunicación entre sus miembros (entretención, contactos profesionales, etc.). Además de la meta de una red social, o su público objetivo, existen otros factores que definen la calidad de la interacción entre sus miembros. Entre ellos se encuentran las herramientas brindadas al usuario para contactar a otros miembros de la red social, las alternativas de comunicación entre los miembros, la privacidad de la información dentro y fuera del sitio, y los dispositivos o plataformas que soportan a la red social. Con respecto a esto último, muchas IESN han desarrollado versiones de sus aplicaciones para el uso en dispositivos móviles, como notebooks (sitios Web tradicionales), Palms, teléfonos celulares y smartphones (aplicaciones ad-hoc, o interfaces Web alternativas).

A pesar de que empresas como Facebook, Twitter o Foursquare han logrado llevar sus redes sociales a dichos dispositivos, permitiendo que sus miembros accedan a ellas desde cualquier parte, no se ha modificado el paradigma de interacción entre los miembros. Es decir, todas las interacciones se realizan dentro de un medio virtual (sitio Web).

Frente a todo lo anterior, debe considerarse que muchos de los dispositivos móviles cuentan con herramientas como GPS, antena WIFI, Internet móvil e incluso brújula, los cuales permitirían enriquecer la interacción de los usuarios de una misma red social, obteniendo las ventajas de la comunicación cara-a-cara entre sus miembros [20, 21].

Al tomar en consideración la importancia que tiene la comunicación entre las personas, sobre todo cuando ellas comparten intereses personales o profesionales, es relevante plantear la ventaja que tiene un encuentro cara-a-cara, como el antes mencionado, por sobre otros medios comunes existentes hoy en día, como son las llamadas telefónicas, mensajes de texto, correos electrónicos, o contactos a través de redes sociales. Es en este contexto donde la utilización de los dispositivos móviles por parte de las personas puede brindar gran ayuda, más aún al tener presente las capacidades que ellos presentan en la actualidad, y la probable masificación de los mismos en el futuro próximo.

En el campo de la comunicación y localización entre dispositivos móviles se han desarrollado soluciones sobre redes MANET (Mobile Ad-hoc Network), que consisten en redes ad-hoc

creadas a partir de la cercanía entre los dispositivos. Entre las ventajas de estas redes se encuentra que el rango de alcance es amplio y dinámico, ajustándose a los dispositivos conectados a la red. Además, permiten conocer la ubicación relativa de los dispositivos, y la distancia aproximada existente entre ellos (medida en *nodos de red* de separación). Otra posible solución consiste en conectar los dispositivos a través de una red inalámbrica tradicional, mediante enrutadores. A pesar de que estas redes son muy populares, presentan desventajas para los objetivos deseados en este trabajo, pues al ser la infraestructura fija (routers fijos), el funcionamiento de dicha red está sujeto al espacio abarcado en primera instancia.

Durante el presente trabajo de título se diseñará y desarrollará una aplicación que permita extender las funcionalidades de una red social, de tal manera de sacar mejor provecho a las tecnologías provistas por los dispositivos móviles (en particular, la conexión a una red MANET) para apoyar al usuario en la tarea de concretar encuentros cara-a-cara con los miembros de su red social que él estime conveniente.

1.1. Justificación de la Propuesta

Los avances en las tecnologías ofrecidas por los dispositivos móviles han hecho que el común de las personas los empleen en sus tareas diarias. La principal de ellas es establecer comunicación con otras personas, mediante mensajes de texto, llamadas telefónicas, correos electrónicos, mensajería instantánea o contactos a través de redes sociales. Sin embargo, a pesar de que dichos dispositivos tienen cada vez más capacidades y tecnologías incorporadas, el mecanismo de comunicación entre personas a través de redes sociales no ha cambiado su paradigma, ni ha hecho un buen aprovechamiento de la movilidad de los dispositivos.

Por otra parte, se reconoce la importancia del rol que juega la comunicación cara-a-cara.

En contextos casuales se asume que las personas prefieren mantener reuniones físicas con sus familiares o amigos, en lugar de la comunicación por un medio virtual[1]. Es de esperar que una herramienta que facilite dichos encuentros sea muy exitosa, puesto que acercaría a las personas en escenarios como centros comerciales, recintos deportivos, lugares vacacionales, conciertos, colegios, universidades, etc. En contextos de trabajo existen actividades donde las discusiones cara-a-cara mejoran el rendimiento, mientras que en otras la comunicación por un medio virtual es mejor.

[1] El caso contrario (cuando familiares o amigos no quieren encontrarse cara-a-cara) constituye un asunto de "visibilidad entre usuarios" para la presente propuesta, y será analizado en la sección 3.1

Un análisis de investigaciones en la materia se encuentra en [22], donde se señala que la comunicación por medios virtuales es más lenta que las reuniones físicas (probablemente como resultado de tener que escribir), lo que provoca menor generación de oraciones y mayor tiempo en completar una tarea asignada. Esto produce frustración con el medio de comunicación, lo que se traduce en una peor evaluación de éste último y del grupo de trabajo. Sin embargo, también se señala que genera una comunicación más orientada a completar las tareas, y menos orientada a sociabilizar. El caso opuesto se observa en la comunicación cara-a-cara, donde se tiene una mayor velocidad para completar tareas asignadas, mayor cantidad de oraciones expresadas, y una naturaleza más social en la comunicación. Finalmente, los investigadores observan que las reuniones físicas mejoran el rendimiento al realizar tareas que requieren una elevada interdependencia entre los participantes, mientras que la comunicación por medios virtuales ha demostrado ser un mecanismo más igualitario, con mayor equidad de participación y menores presiones sociales entre sus miembros[2], lo que la hace mejor para tareas del tipo *brainstorming*[22].

La importancia de contar con una aplicación que facilite los encuentros cara-a-cara entre personas se observa en muchas áreas, incluyendo encuentros casuales entre amigos, reuniones profesionales, encuentros entre grupos con intereses comunes, etc. En particular, esta aplicación brindaría una enorme ayuda en contextos como hospitales, u oficinas de emergencia, donde localizar a es un tema crítico.

El problema a abordar presenta desafíos en distintos aspectos. Por ejemplo en la seguridad, donde es importante garantizar que los usuarios sean quienes dicen ser dentro de la MANET, y que la comunicación entre usuarios y la información obtenida y enviada a las redes sociales no pueda ser interceptada o adulterada. Para ello, deberá emplearse un protocolo de comunicación dentro de la MANET que permita abordar el tema de la seguridad. En cuanto a la aplicación en sí misma, será importante que, una vez importada una lista de contactos desde una red social, existan opciones para enriquecer dicha lista, de tal modo que el usuario pueda modificar aspectos de visibilidad dentro de la MANET, o bien crear sub-grupos dentro de su lista de contactos, para, por ejemplo, recibir notificaciones especiales. Además, la aplicación deberá tener una interfaz y un funcionamiento que cumplan con brindar el mejor *awareness* posible, sin comprometer el desarrollo normal de las otras actividades del usuario. Por último, el diseño de la arquitectura debe apuntar a que la aplicación soporte conexión a distintas redes sociales, y sea compatible con una variedad razonable de sistemas operativos de dispositivos móviles.

[2] El caso típico consiste en la inhibición de un miembro del grupo de trabajo para expresar ideas cuando se enfrenta a sus jefes, o a personas más especializadas en el área de trabajo.

1.2. Objetivo General

Extender una red social tradicional, para llevar la interacción entre sus miembros a un plano físico. Para ello se desarrollará un software que promueva las interacciones cara-a-cara entre miembros de una red social, cuando éstos se encuentren en las proximidades uno del otro. Para realizar este trabajo se reutilizarán algunas de las capacidades provistas por la librería HLMP API, que soporta la comunicación entre usuarios móviles en ambientes ad hoc [1].

1.3. Objetivos Específicos

A partir del objetivo general se han definido los siguientes objetivos específicos:

1. Diseñar arquitectura de aplicación, considerando soporte para distintas redes sociales, administración de datos de usuario y configuración personal, lista enriquecida de contactos, conectividad entre dispositivos, etc.
2. Definir protocolo de comunicación entre dispositivos que permita manejar temas de seguridad.
3. Implementación de comunicación entre dispositivos. Implementación de protocolos de mensajería instantánea y asíncrona.
4. Implementación de funcionalidades que importen una lista de contactos desde una red social, para el posterior uso por parte de la aplicación.
5. Implementación de funcionalidades para gestionar datos del usuario, configuración local y lista de contactos.
6. Diseño e implementación de interfaces gráficas y de la funcionalidad del software, para que sean familiares para los usuarios, basándose en aplicaciones similares populares.
7. Diseño e implementación de interfaces gráficas y mecanismos de alerta para cuando la aplicación detecte la proximidad de un contacto.

En la siguiente sección se explica el *marco de trabajo* de la memoria (Servicios de redes sociales, análisis de programas similares, plataformas de desarrollo de aplicaciones, etc.). En la sección 3 se presenta la *concepción de la solución*, con los requisitos y la arquitectura de software preliminar. En la sección 4 se detalla el *diseño arquitectónico* y el *diseño de las interfaces*

gráficas de la aplicación desarrollada. Los *resultados* del trabajo de memoria se describen en la sección 5, mientras que las *conclusiones y el trabajo futuro* se presentan en la sección 6.

2. Marco de Trabajo

A continuación se presenta el conjunto de trabajos relacionados que definen el marco de trabajo del tema de memoria. Dichos trabajos aportan el conocimiento necesario para familiarizarse con las plataformas de desarrollo, el software asociado, y el estado del arte en el tema de redes sociales y comunicaciones entre sus usuarios.

2.1. Servicios de Redes Sociales

Entendemos por "servicio de red social" (o SNS, por su sigla en inglés) un servicio online, sitio Web o plataforma que como mínimo permita a los usuarios (1) construir un perfil personal público o semi-público dentro de parámetros establecidos, (2) administrar una lista de otros usuarios con quienes se mantiene una conexión y, (3) ver y navegar la lista de contactos propia y de otros usuarios del sistema. La naturaleza y nomenclatura para dichas conexiones varía de sitio a sitio (ej., para los contactos dentro de una SNS se emplean los términos "amigo", "contacto", "seguidor", etc.). La facultad de los SNSs para extender la lista de contactos propios mediante la navegación en la lista de los usuarios del sistema constituye la principal diferencia entre estos servicios y aquellos que sólo proveen comunicación entre usuarios; los servicios CMC ("Computer-Mediated Communication") [2].

Las primeras formas de SNSs aparecieron como simples comunidades online, donde los usuarios interactuaban a través de foros, salas de chat, y versiones primitivas de los hoy populares perfiles de usuario y *streams* públicos ("muro", lista de "tweets", etc.). Las versiones más modernas de SNSs, según la definición antes planteada, no aparecieron sino hasta el 2002, con Friendster. A partir de ese momento, las SNSs proliferaron, apuntando a diferentes públicos objetivos, ofreciendo diversas opciones de privacidad o alternativas de comunicación entre sus usuarios. Hoy en día, sitios como Facebook, Twitter, MySpace, Orkut, LinkedIn, Tumblr, Foursquare, Formspring.me e incluso sitios como YouTube, Flickr, o Last.fm están entre los más populares, alcanzando más de 250 millones de visitantes únicos en el mes de diciembre de 2010 [3].

Un factor común entre las SNSs mencionadas es que se han mantenido a la par con las tendencias tecnológicas y sociales que les son contemporáneas, y atendiendo a las necesidades y preferencias de sus usuarios, han alcanzado la popularidad que hoy poseen. Más específicamente, en el ámbito de los dispositivos móviles, las SNSs han desarrollado interfaces especiales para el acceso mediante dichos dispositivos, facilitando que los usuarios se conecten a la plataforma desde cualquier lugar. La plataforma Twitter provee de una interfaz Web diseñada especialmente

para dispositivos móviles[3] , junto con clientes para los sistemas operativos iOs y Android. Además, posee un sistema de interacción con la plataforma a través de mensajes SMS [4]. Al igual que Twitter, Facebook provee de sitios Web para dispositivos móviles[4] , interacción mediante SMS, y clientes para sistemas operativos móviles (Android, iOs, Blackberry, Palm, etc.) [5]. También cuenta con la interacción mediante SMS para actualización de estado, mensajes entre usuarios, búsqueda en el sitio, etc. [6]. En agosto de 2010, Facebook lanzó su servicio Places[5]; un servicio de geo-localización que permite a los usuarios publicar su ubicación actual (o la ubicación de fotos o *estados*) en la plataforma, y conocer la ubicación de sus contactos. También cabe destacar la inclusión de MySpace en las plataformas móviles. Esta SNS provee de un cliente para iPhone, y se encuentra en desarrollo un cliente para Android. Para estos y otros teléfonos, provee de una interfaz Web especial[6] que provee de las mismas funcionalidades de la interfaz principal. Por último, la plataforma Foursquare nació como una plataforma móvil, cuya funcionalidad es permitir que los usuarios se "checkeen" en sitios dentro de la ciudad, para compartir su ubicación con sus contactos, y obtener "puntos" y ofertas a cambio. Foursquare tiene aplicaciones cliente para iPhone, Android, Blackberry, Palm, etc. [7].

Cabe recalcar el caso de Facebook y Foursquare. Mientras que la tendencia de las SNS (y de todos los otros sitios) en cuanto a movilidad se limita a proveer de una interfaz Web apropiada para dispositivos móviles, tanto Foursquare como Facebook han ampliado sus funcionalidades para incluir geolocalización y sistemas de alerta de proximidad de contactos. Es importante notar la amplia gama de oportunidades que ofrecen las aplicaciones señaladas. Los dispositivos móviles modernos cuentan con conectividad a internet, geolocalización mediante GPS, y una elevada capacidad de procesamiento. Mediante la utilización de estas herramientas, una aplicación diseñada para dispositivos móviles puede ayudar en tareas complejas o críticas, como por ejemplo en las inspecciones en construcción, en el área de la salud o en respuesta ante emergencias [8], todo esto en el ámbito de la colaboración móvil.

a. Facebook API

Como parte del desarrollo de la aplicación será necesario importar contactos desde una de SNS, como mínimo. A modo de introducción a este problema, se desarrolló una aplicación en PHP que, usando la API de Facebook (secciones Social Graph y Authentication [14][15]), permitiera realizar esta tarea.

[3] Twitter. URL: mobile.twitter.com. Última visita: Diciembre, 2011.
[4] Facebook. URL: {m.facebook.com, touch.facebook.com, 0.facebook.com}. Última visita: Diciembre, 2011.
[5] Share Where You Are. URL: http://www.facebook.com/places/. Última visita: Diciembre, 2011.
[6] MySpace. URL: http://m.myspace.com. Última visita: Diciembre, 2011.

El primer paso consiste en crear una aplicación de Facebook en su sitio "Create Applicatons"[7]. Como resultado de este paso, la aplicación creada tendrá un "Application ID" y un "Application Secret". Ambos códigos serán usados posteriormente en el proceso de autenticación en el sitio.

La plataforma de desarrollo de Facebook utiliza el protocolo OAuth 2.0 [16] para autenticación y autorización. Este protocolo actúa de intermediario entre el *proveedor del servicio* (o *server*, en este caso, Facebook), el *dueño de los recursos* (en este caso, el *usuario* de Facebook) y el *consumidor* (o *cliente*¸ en este caso, la *aplicación* a desarrollar)[8]. La función del protocolo es permitir que el *usuario*, mediante la *aplicación* acceda a sus recursos dentro del *servidor*, sin compartir sus credenciales (por lo general, "username" y "password") directamente con la *aplicación*. Con esto, el *usuario* tiene seguridad de que la *aplicación* no hará mal uso de sus credenciales.

El mecanismo de funcionamiento del protocolo para acceder a recursos en Facebook desde aplicaciones Web es el siguiente:

a) Acceder al diálogo de inicio de sesión provisto por Facebook. En la URL deben ir, como parámetros GET, el valor de "Application ID" mencionado anteriormente, una URL de redirección, y (opcionalmente) el formato de pantalla del dispositivo cliente. La URL completa tendrá el siguiente aspecto:

```
www.graph.facebook.com/oauth/authorize?client_id=...&
                                redirect_uri=redirect_url.com&
                                display=touch
```

Como resultado, el navegador mostrará una interfaz de inicio de sesión a Facebook, donde el usuario deberá ingresar sus credenciales.

b) Si el inicio de sesión se realiza con éxito, el navegador será redirigido a la URL especificada como parámetro, y como parámetro GET de dicha URL se encontrará un *código*, necesario en el siguiente paso. Un ejemplo de la URL de redirección es el siguiente:

```
www.redirect_url.com?code=code_from_facebook
```

[7] Facebook Developers. URL: https://developers.facebook.com/apps. Última visita: Diciembre, 2011.

[8] La terminología de los actores dentro del protocolo se encuentra en la especificación del mismo [16]. Una versión menos extensa de lo mismo se encuentra en "The Authoritative Guide to OAuth 1.0" (http://hueniverse.com/oauth/guide/terminology/). A pesar de que son versiones distintas del protocolo, la terminología se mantiene.

c) El código obtenido se debe intercambiar por un *token* de acceso, que permitirá acceder a los recursos dentro de Facebook. Para ello, debe hacerse un *request* indicando el "Application ID", "Application Secret" y el código obtenido. La URL será como la siguiente:

```
www.graph.facebook.com/oauth/access_token?client_id=...&
                                    client_secret=...&
                                    code=code_from_facebook
```

Como resultado, el navegador será redirigido a una página cuyo único contenido HTML es el *access_token*.

d) Una vez que se posee el *access_token* es posible acceder a los recursos del usuario dentro de Facebook, mediante un *request* que en la URL incluya el token. Por ejemplo, para obtener la lista de amigos del usuario que inició sesión mediante el proceso anterior, debe accederse a la URL `www.graph.facebook.com/me/friends?access_token=...`
El resultado será un objeto JSON, que contiene la lista de amigos, y que la aplicación puede parsear para su futuro uso.

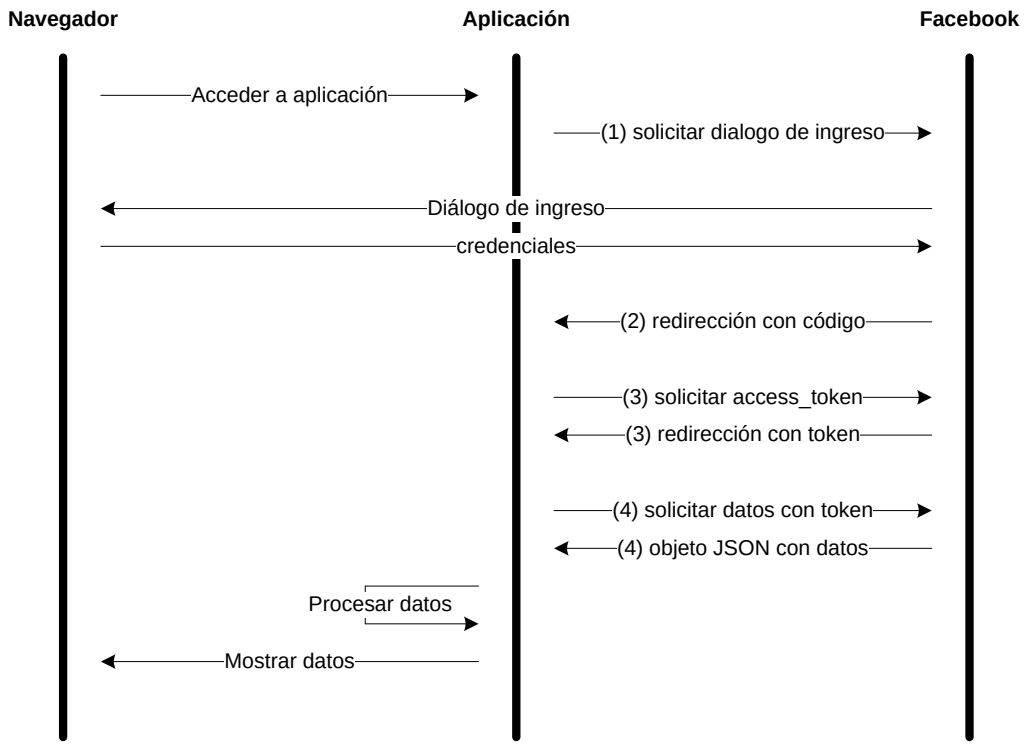

Figura 1. Diagrama de flujo de protocolo OAuth para Facebook

La aplicación de prueba se desarrolló sobre PHP, y no consideró aspectos de diseño visual ni interfaces de usuario apropiadas. Más bien, se orientó completamente a implementar el protocolo descrito, y el resultado fue satisfactorio. Como conclusión, el alumno clarificó los conceptos relacionados con la API de desarrollo de Facebook, y el mecanismo que una aplicación debe ejecutar para obtener los datos necesarios. En este aspecto, se hizo evidente la factibilidad del proyecto, y como trabajo pendiente queda realizar el mismo desarrollo para otras SNS, para así incluirlas en la aplicación final.

b. Análisis de Programas Similares

En el ámbito de las SNS, o software relacionado a mensajería entre contactos de una misma red, hay elementos que se repiten y que han probado ser de utilidad para la interacción entre los usuarios. Estos elementos gráficos o funcionales deberán ser incorporados en el diseño de las interfaces de la aplicación cuando corresponda. El siguiente análisis permitirá destacar los elementos relevantes, para que al hacer uso de ellos la aplicación sea más amigable con el usuario:

a) Facebook: Esta SNS ha logrado una inmensa popularidad a nivel mundial, y ha logrado una gran penetración en Chile, alcanzando 6.5 millones de visitas en enero de 2011 [18]. Como se mencionó anteriormente, Facebook ha desarrollado soluciones para comunicación instantánea, asíncrona, multimedia, e incluso geo-localizada.

i. Mensajería instantánea: Se muestra una lista con los "amigos" de la red social, indicando si se encuentran o no disponibles para conversar mediante colores (un círculo verde, en caso de estar *disponible*). Es posible configurar el propio estado como "*no disponible*" para chatear, tanto para todos los contactos como para grupos de ellos (conocidos, amigos cercanos, etc.). No se guarda historial de las conversaciones de chat.

ii. Notificaciones: Las notificaciones de los eventos en el sitio (nuevo mensaje instantáneo, nuevo mensaje personal, solicitud de amistad, toque, etc.) se muestran de forma visual y auditiva. Como parte del sitio existe una sección de notificaciones, que muestra cuando un nuevo evento ocurre, y para mensajería instantánea se muestra la ventana de conversación, junto con cambios en el título de la página Web y notificaciones sonoras.

iii. Mensajes personales: Se provee de un sistema de mensajes entre miembros de la red social, no necesariamente "amigos" entre ellos. Los

mensajes pueden ser enviados a uno o más destinatarios, notificando instantáneamente a los receptores, o bien cuando se conecten a la red. Se guarda un historial de todos los mensajes enviados, con la opción de borrarlos unilateralmente (no se puede borrar el historial de otro usuario).

iv. **Toque:** Esta herramienta es una forma muy básica de contacto entre dos miembros de la red. Consiste en enviar una notificación a un contacto sin otro mensaje que la notificación en sí misma. Posee características atractivas para mantener el contacto entre personas, pues es personal, privada, eficiente, sencilla y crea una sensación de reciprocidad entre los participantes [9]. En un contexto de comunicación instantánea y geo-localizada, un "toque" puede ser una herramienta valiosa para iniciar una reunión física.

v. **Sugerencia de contactos:** Se provee de una herramienta para sugerir contactos entre los miembros de la red. Mediante este mecanismo un usuario puede sugerirle a un contacto que añada a su red social a un tercer usuario, amigo del primero pero no del segundo.

b) **Gtalk[10]:** Este cliente de mensajería instantánea nació como una herramienta de Gmail, donde los contactos disponibles para chat eran los mismos miembros de la libreta de contactos del correo electrónico. En la actualidad, Google ofrece clientes para Windows y Android, además de una variedad de clientes que implementan la mensajería mediante el protocolo XMPP[11].

i. **Lista de contactos:** Los contactos disponibles para chat se obtienen desde la libreta de contactos de la cuenta de correo electrónico, pero también existe la opción de enviar una invitación de chat a alguien que no exista en dicha libreta. Los estados posibles de cada contacto se grafican con un círculo de color, y pueden ser *disponible*, *ocupado*, *ausente*, *invisible* y *desconectado*.

[9] Facebook Poke – The Most Underestimated Facebook Marketing Tool. Escrito por Stacey Harmon el 4 de Diciembre de 2010. URL: http://www.pixelcoaching.com/public/facebook-poke/. Última visita: Enero, 2012.
[10] Chat de Google: chatea con tus familiares y amigos. URL: http://www.google.com/talk/intl/es/. Última visita: Diciembre, 2011.
[11] The XMPP Standards Foundation. URL: http://xmpp.org/. Última visita: Diciembre, 2011.

ii. **Historial de conversaciones:** Al estar ligado a una cuenta de Gmail, las conversaciones de chat se almacenan en dicha cuenta bajo la etiqueta "Chats". Estas conversaciones se pueden borrar de la cuenta propia, pero permanecerán en las cuentas de los otros participantes de la conversación.

iii. **Mensajes offline:** Gtalk permite enviar mensajes *offline*. En la práctica significa que, para enviar un mensaje a un contacto, puedo optar por el correo electrónico, mensajería instantánea (si el contacto está conectado), o mensajería asíncrona u *offline* (si no está conectado). Éstos últimos serán notificados al contacto en cuanto se conecte a Gtalk.

iv. **Notificaciones:** Para la versión integrada en Gmail, cuando hay una nueva conversación de mensajería instantánea se despliega un pop-up con la conversación. Cuando el mensaje es asíncrono, se muestra el contenido como si fuera un nuevo correo electrónico. Las versiones *stand-alone* del cliente Gtalk ofrecen notificaciones visuales y sonoras para alertar de una nueva conversación de chat (ventana de chat parpadeando, sonido de "ping", íconos que cambian de color, etc.).

c) **Windows Live Messenger[12,13]:** Anteriormente llamado "MSN Messenger", este cliente de mensajería instantánea (IM) no está directamente relacionado a una SNS como se definió en el capítulo 0. Sin embargo, es uno de los clientes de IM más populares, llegando a alcanzar 300 millones de usuarios activos cada mes durante el 2010 [17]. Para usar este software es necesario tener una cuenta Windows Live ID[14].

i. **Lista de contactos:** La lista de contactos permite crear grupos de usuarios, de tal modo de ordenar la lista. Los usuarios pueden estar *desconectados*, *invisibles*, *disponibles*, *ocupados* o *ausentes*; y cada estado se ve reflejado en un círculo de color junto al nombre del usuario (gris, gris, verde, rojo y amarillo, respectivamente). Además de la mensajería instantánea, es posible enviar un correo electrónico a un usuario de la lista.

[12] Messenger: Descarga la mensajería instantánea de Messenger (MI). Url: http://explore.live.com/messenger. Última visita: Diciembre, 2011.
[13] Las funcionalidades descritas corresponden a Microsoft Live Messenger 2011, comp. 15.4.3538.513.
[14] Windows Live ID. URL: https://accountservices.passport.net/ppnetworkhome.srf?vv=1200&mkt=ES-CL&lc=13322. Última visita: Diciembre, 2011.

ii. **Envío de archivos:** Este cliente permite enviar archivos de forma síncrona entre dos contactos de la red. El envío requiere que ambos contactos se encuentren conectados.

iii. **Notificaciones:** El cliente notifica con alertas visuales y sonoras cuando un contacto se conecta a la red, cuando llegan mensajes nuevos o cuando llegan correos nuevos. Los tipos de notificaciones son configurables.

iv. **Historial de conversaciones:** El cliente permite al usuario almacenar localmente un historial de las conversaciones llevadas a cabo en ese dispositivo.

d) **Pidgin[15,16]:** Este cliente de mensajería instantánea implementa los protocolos de chat de una variedad de servicios distintos. Entre ellos, Google Talk, Facebook chat, Live Messenger, ICQ, IRC, etc. Su análisis es importante pues es un ejemplo de cómo presentar y administrar la información personal y grafos sociales de distintas redes dentro de la misma interfaz.

i. **Cuentas y lista de contactos:** Es posible crear una lista de cuentas de mensajería instantánea. Cada cuenta pertenece a un servicio MI, y accede a él mediante las credenciales que sean necesarias (normalmente, *username* y *password*). Luego la lista de contactos se divide en grupos, correspondientes a los contactos de cada una de las cuentas de MI registradas en el programa.

ii. **Grupos:** Además de los grupos iniciales (uno por cada cuenta de MI), se pueden crear grupos adicionales (ej., "casa", "trabajo", etc.). Estos grupos sólo sirven para ordenar la lista de contactos.

iii. **Estados:** En la lista de contactos se muestra el estado de cada uno de ellos en sus respectivos servicios MI, siguiendo una simbología estándar para todos. Sin embargo, el usuario sólo puede cambiar su estado para el conjunto de todos sus contactos, y no es posible definir estados para subgrupos de ellos.

[15] Pidgin, the universal chat client. URL: http://www.pidgin.im/. Última visita: Diciembre, 2011.
[16] Las funcionalidades descritas corresponden a la versión 2.10.1 de Pidgin para Windows.

Como resultado del análisis, se tendrá como objetivo lograr que la aplicación a desarrollar cumpla con los siguientes requisitos:

i. **Cuentas:** Se podrá importar contactos usando distintas cuentas de SNS. Cada vez que se importe un contacto, éste quedará asociado a la cuenta con que fue importado.

ii. **Lista de contactos:** Una vez que se han importado los contactos, quedarán por defecto desagrupados. Los grupos deberán ser creados por el usuario del software, y será su responsabilidad agregar o quitar contactos desde los grupos.

iii. **Estados:** El usuario podrá cambiar su estado para cada grupo que cree. Con esto, los grupos no sólo cumplen un rol de ordenamiento en la lista de contactos, sino que permiten administrar el grado de privacidad que el usuario desee tener con cada grupo de contactos. Los estados serán *disponible* (por defecto), *ocupado*, *invisible*, *ausente* y *desconectado*.

iv. **Notificaciones:** Se buscará brindar notificaciones visuales (pop-up con ventana de conversación, ícono parpadeante en la barra de tareas, etc.) y auditivas para los eventos del software. Se ofrecerán opciones de configuración al respecto (ej., sin notificaciones sonoras).

v. **Historial de conversaciones:** Se ofrecerá la opción de almacenar localmente el historial de conversaciones.

vi. **Toque:** Se ofrecerá la herramienta "toque", dado que en un contexto físico puede aportar a llamar la atención de un contacto.

vii. **Envío de archivos:** Se permitirá que se envíen archivos entre usuarios, siempre y cuando se encuentren ambos conectados.

viii. **Sugerencia de contactos:** Se ofrecerá un mecanismo para sugerencia de contactos. De esta forma, se podrá ampliar el grafo social mediante el uso de la aplicación.

ix. **Mensajes offline:** Se evaluará la factibilidad y usabilidad del envío de mensajes offline entre usuarios (similar a Google Talk), de tal modo que no sea confuso ni perjudique la ejecución de otros procesos.

2.2. Colaboración Móvil

Durante el último tiempo se han observado grandes avances en la tecnología móvil. Este progreso abarca tecnologías inalámbricas de comunicación y transmisión de datos, capacidad de cómputo, peso de los dispositivos, precio, duración de la batería y (muy importante) la disponibilidad y costo de las comunicaciones entre dispositivos. Todos estos avances han contribuido a modificar el comportamiento y las necesidades de los usuarios, quienes comienzan a ser "trabajadores móviles" que desean aprovechar cualquier tiempo libre que tengan, independiente del lugar donde se encuentren.

Este cambio en el comportamiento de los usuarios ha motivado diversas investigaciones, las cuales apuntan a comprender la relación existente entre las soluciones computacionales móviles y la productividad en los trabajadores "nómades". El objetivo ha sido incrementar la eficiencia del trabajo en escenarios como salud, negocios, educación, seguridad, procesos de respuesta ante emergencia y trabajo social.

Los trabajos móviles soportados por computadores incluyen *trabajo desacoplado* (trabajo hecho por trabajadores móviles usando sólo recursos locales), *trabajo poco-acoplado* (trabajadores móviles interactuando on-demand con servidores u otros trabajadores móviles por periodos de tiempo cortos) y *trabajo muy acoplado* (usuarios nómades conectados con otros usuarios o servidores). Sin embargo, en el ámbito de colaboración móvil, los principales desafíos se encuentran cuando los usuarios necesitan interactuar con recursos adicionales a los locales, es decir, existe algún grado de acoplamiento entre ellos. Este problema ha sido abordado con el objetivo de plantear una arquitectura de referencia, debido a que los problemas que se encuentran durante el desarrollo de una solución colaborativa móvil son comunes a los distintos contextos de aplicación (mensajería, información compartida, sincronización, alerta de disponibilidad de usuarios/locaciones, autonomía de datos y servicios, etc.) [8].

2.3. Herramientas de geolocalización

A continuación se presenta un análisis de las principales herramientas de geolocalización relacionadas a redes sociales, o dispositivos móviles.

Google Latitude[17]**:** este servicio, lanzado en febrero de 2009, permite a los usuarios registrados compartir su ubicación con los contactos que desee. La ubicación puede ser introducida de forma manual, o bien ser detectada automáticamente, usando GPS, localización WI-FI o posicionamiento celular (dependiendo del dispositivo). Además, el usuario puede configurar la privacidad de su información, escogiendo compartir su ubicación con todos sus amigos, compartirla solo con algunos, o bien no compartir su ubicación. La información relativa a la ubicación de los contactos de un usuario se muestra en un mapa de Google Maps, mediante íconos que representan a cada contacto localizado, y al usuario mismo.

Facebook (location)[18]**:** la red social Facebook anunció en agosto de 2010 el servicio de geolocalización "Facebook Places", consistente en una herramienta que permitía a sus usuarios hacer "check in" en lugares para que los contactos de la red social conocieran su ubicación. El servicio fue descontinuado en agosto de 2011, siendo reemplazado por un sistema de localización en cada publicación agregada por el usuario a la red social. Así, al subir fotos, videos o actualizaciones de estado, el usuario puede además compartir su ubicación (pasada, presente o futura) con sus contactos. En dispositivos móviles, la herramienta sugiere al usuario una lista de ubicaciones posibles, basándose en los sistemas de geolocalización que el dispositivo tenga disponibles.

Foursquare[19]**:** esta herramienta permite a sus usuarios compartir su ubicación con sus contactos mediante hacer "check in" con sus dispositivos móviles. Los contactos pueden buscarse dentro de los usuarios de Foursquare, o bien importarlos desde otras redes sociales. La información de las ubicaciones de los contactos se muestra en un mapa, junto con la información del usuario del sistema e información relativa a lugares de interés (tiendas, cines, restaurantes, etc.). Además, el usuario puede enviar la información de cada "check in" a otras redes sociales, de tal manera de informar a otros contactos acerca de su ubicación.

[17] Google Latitude. URL: https://www.google.com/latitude. Última visita: Abril 2012.
[18] Share Where You Are. URL: https://www.facebook.com/about/location. Última visita: Abril 2012.
[19] foursquare. URL: https://foursquare.com/. Última visita: Abril 2012.

Find my Friends[20]**:** este servicio es similar a Google Latitude. Permite a sus usuarios ingresar a la aplicación accediendo con una Apple ID[21], buscar contactos registrados, compartir la ubicación propia, y visualizar la ubicación de los contactos en un mapa.

2.4. HLMP

Los avances en las comunicaciones inalámbricas y dispositivos móviles han abierto oportunidades para realizar actividades de colaboración móvil en dichos dispositivos. Uno de los aspectos más desafiantes al desarrollar una aplicación de colaboración móvil es la implementación de una infraestructura de comunicaciones entre los dispositivos, debido a la cantidad y dificultad de los requerimientos que esto presenta (seguridad, estabilidad, velocidad, etc.). Abordar este problema desde cero requiere un gran esfuerzo, y en muchos casos amenaza el desarrollo del proyecto completo [1].

Por lo anterior, se han propuesto diversas herramientas de soporte para las aplicaciones en cuestión, herramientas que no siempre son apropiadas para abordar el problema del presente trabajo (debido a que están orientadas a transferencia de archivos, requieren infraestructura fija o dispositivos con alta capacidad de procesamiento, etc.).

HLMP (High Level Manet Protocol) es un protocolo de comunicaciones entre dispositivos mediante redes móviles ad-hoc. Permite conectar dispositivos de forma dinámica, sin depender de infraestructura fija, y brinda mecanismos de alerta para los eventos que ocurren en la red. Además, provee una API que facilita el desarrollo de aplicaciones de colaboración móvil a través del uso de sus funciones y servicios. Debido a estas propiedades, será la herramienta empleada como protocolo de comunicaciones entre dispositivos para el presente trabajo [1].

HLMP API ofrece tres elementos principales, los que serán usados por la aplicación para ofrecer las herramientas de comunicación a sus usuarios:

1. **NetUser**: representa un nodo (dispositivo) en la red. Cada nodo conectado a la red es "visible" por todos los demás nodos conectados, mediante comunicación directa entre ellos o enrutamiento a través de otros nodos.
2. **Message**: representan mensajes entre dispositivo. La aplicación debe extender alguno de los tipos nativos de *message* de HLMP, dependiendo de las necesidades del mensaje a implementar. Por ejemplo, para implementar mensajes de chat entre dos *netUser* se debe

[20] App Store – Find my Friends. URL: http://itunes.apple.com/us/app/find-my-friends/id466122094?mt=8. Última visita: Abril 2012.
[21] Apple – My Apple ID. URL: https://appleid.apple.com/cgi-bin/WebObjects/MyAppleId.woa/. Última visita: Abril 2012.

extender desde *SafeUnicastMessage*, que representa un mensaje de HLMP, con destinatario conocido, que espera un ACK del nodo receptor.

3. **Sub-Protocol**: representa un subprotocolo de HLMP. Cada subprotocolo administra el comportamiento de la aplicación al enviar o recibir cierto tipo de mensajes. El conjunto de tipos de mensajes asignados a cada subprotocolo existente se define al configurar la conexión HLMP dentro de la aplicación.

2.5. Plataforma de Desarrollo

La aplicación deseada deberá funcionar en dispositivos móviles que, como mínimo, tengan antena WIFI (para acceder a una MANET inalámbrica). En el caso de smartphones, se espera que la aplicación corra sobre el sistema operativo Windows Phone 7 (WP7), mientras que para notebooks el sistema operativo será, a priori, Windows 7[22]. En ambos casos se utilizarán las siguientes herramientas de desarrollo:

- Microsoft Visual Studio 2010 Ultimate
- Microsoft Expression Blend 4
- Microsoft Windows SDK for Windows 7 and .NET Framework 4 [12]
- Windows Phone SDK (WP7-SDK) [13]

Para llevar a cabo el desarrollo de la aplicación en ambas versiones (Windows 7 y WP7) se trabajará sobre Windows Presentation Foundation (WPF) y Silverlight, respectivamente. Ambas plataformas son subsistemas del framework .NET. La ventaja de trabajar sobre dichas plataformas es la de unificar el desarrollo de ambas versiones de la aplicación (siempre y cuando sea posible), ya que tanto para WPF como Silverlight se puede trabajar en C#, y las aplicaciones tienen arquitecturas similares (basadas en código C# y archivos XAML[23]). Durante el desarrollo se tendrá como meta plantear una arquitectura que desacople el *core* de la aplicación (manejo de datos, detección de usuarios en la MANET, etc.) con los aspectos particulares de cada plataforma (interfaces de usuario, navegación por la aplicación, notificaciones, etc.)

A continuación se presentan los trabajos de desarrollo de prototipos para WP7 y Windows 7, como parte del entrenamiento y revisión de factibilidad para ambas plataformas.

[22] La aplicación a desarrollar necesita del framework .NET 4.0, provisto en Windows 7 y Windows Vista, y es descargable para Windows Server 2003 SP1 y Windows XP SP2 [10].
[23] Extensible Application Markup Language.

a. Silverlight para WP7

Una vez que se completó el prototipo en PHP descrito en (2.1.a), se procedió a crear una aplicación para WP7 que permitiera al usuario ingresar a su cuenta de Facebook y mostrar una lista de sus contactos, todo mediante el protocolo OAuth.

La plataforma Silverlight está compuesta por un subconjunto del framework .NET y un conjunto de componentes y servicios orientados a la interfaz e interacción con el usuario[24]. Las aplicaciones en Silverlight típicamente tienen las interfaces de usuario declaradas en XAML, mientras que la lógica se expresa en un lenguaje perteneciente a .NET (como VisualBasic.Net, C# o J#). Las versiones de Silverlight presentan pequeñas diferencias dependiendo del ambiente donde será ejecutada la aplicación; en este aspecto es importante recalcar que durante el desarrollo del presente trabajo se empleó la versión de Silverlight para Windows Phone, que incluye librerías específicas para dicho sistema operativo, las que permiten acceder al hardware del dispositivo móvil (pantalla multitouch, acelerómetro, brújula, etc.).

El desarrollo se hizo en Visual Studio 2010 Ultimate y se ejecutó sobre el emulador de Windows Phone provisto por WP7-SDK, el cual se conecta a internet mediante el sistema operativo *host* de la emulación. La aplicación implementó de forma íntegra el protocolo descrito en (2.1.a), usando como navegador Web el control WebBrowser [25] de la plataforma Silverlight. A continuación se presentan capturas de pantalla[26] de la aplicación en funcionamiento, desde que se inicia hasta que se han extraído y procesado los datos desde Facebook:

[24] Arquitectura de Silverlight. URL: http://msdn.microsoft.com/es-es/library/bb404713(v=VS.95).aspx#the_silverlight_platform. Última visita: Junio, 2011.

[25] Dentro de la plataforma, se denota "control" a las componentes de interfaz de usuario de una aplicación. Entre ellas se incluyen Canvas, TextBox, Label, Button, etc.

[26] Como protección a la privacidad de los usuarios importados en el programa en cuestión, la información concerniente a ellos se muestra borrosa, puesto que son usuarios reales de Facebook y no fueron creados exclusivamente para efectos de esta demostración

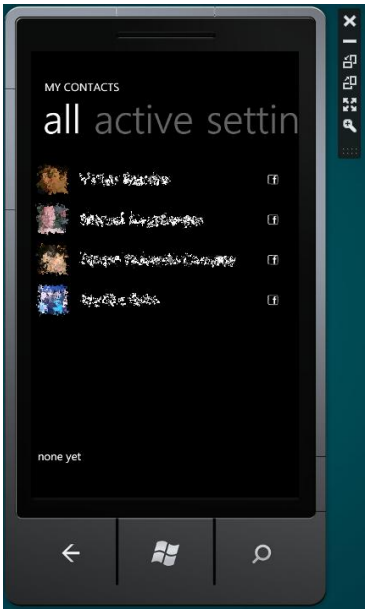

Figura 2. Prototipo WP7, emulador de dispositivo móvil.

En la Figura 2 se muestra el emulador de dispositivo móvil, provisto por el SDK de Windows Phone 7. En el emulador se ejecuta el sistema operativo con las aplicaciones por defecto y las que se estén desarrollando en el momento por el usuario. El funcionamiento de hardware se simula siempre y cuando es posible; en particular, la pantalla *touch* es simulada mediante el uso del *mouse* del computador, y la orientación del dispositivo se cambia mediante los controles del emulador (en la imagen, a la derecha). La conexión a Internet se obtiene desde el sistema operativo host del emulador (en este caso, Windows 7).

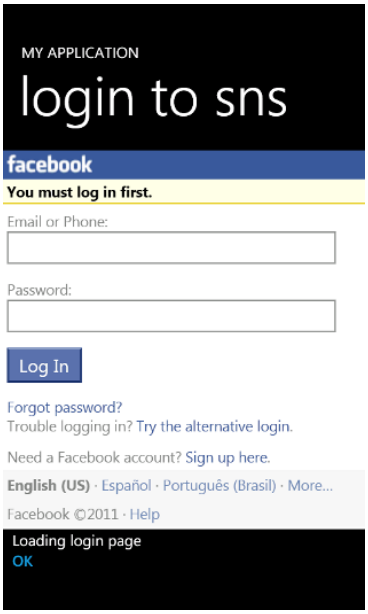

Figura 3. Prototipo WP7, interfaz de acceso a Facebook.

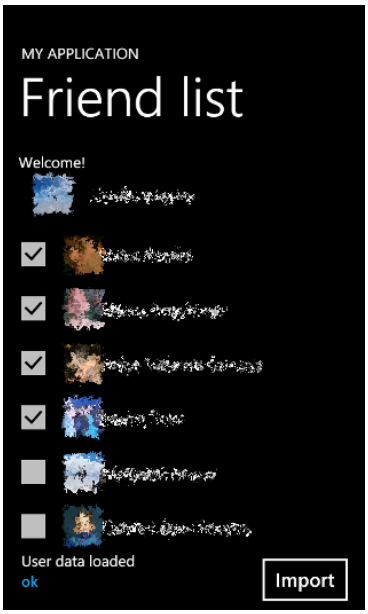

Figura 4. Prototipo WP7, lista de amigos de Facebook.

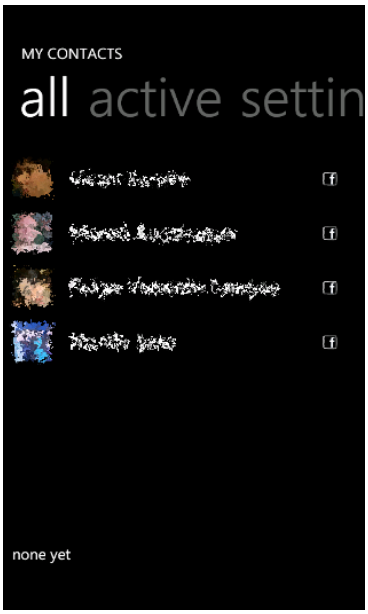

Figura 5. Prototipo WP7, lista con contactos importados.

En la Figura 3 se muestra la pantalla de acceso a Facebook como se presenta en el prototipo desarrollado. Una vez que se ingresan las credenciales correctamente, el prototipo muestra una lista de los contactos de Facebook (Figura 4), desde donde se seleccionan los deseados para posteriormente importarlos al dispositivo. La lista de los contactos importados a la aplicación se muestra en la Figura 5.

Como conclusión de este desarrollo el alumno se familiarizó con la plataforma de desarrollo para dispositivos móviles con WP7, a la vez que integró el prototipo previo (2.1.a) dentro de una aplicación funcional.

b. WPF para Windows 7

Un objetivo deseable de esta memoria es crear versiones de "escritorio" y "móvil" de la aplicación. La investigación previa mostró que mediante el framework .NET y sus componentes WPF y Silverlight existe una gran oportunidad de desarrollar casi simultáneamente ambas versiones. La implementación de muchas funcionalidades de la aplicación se puede hacer independiente de la plataforma final donde se ejecute la aplicación. Por este motivo, se desarrolló un prototipo con las mismas funcionalidades del planteado en (a), pero para ejecutarse sobre Windows 7. A continuación se presentan capturas de pantalla del prototipo desarrollado. Al igual que en (a), la información de los usuarios se muestra borrosa para proteger su privacidad:

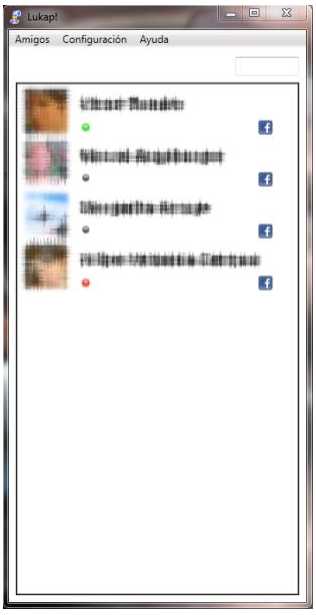

Figura 6. Prototipo para W7, lista de contactos.

En la Figura 6 se muestra la interfaz principal del prototipo para W7, donde se observa la lista de contactos importada con anterioridad.

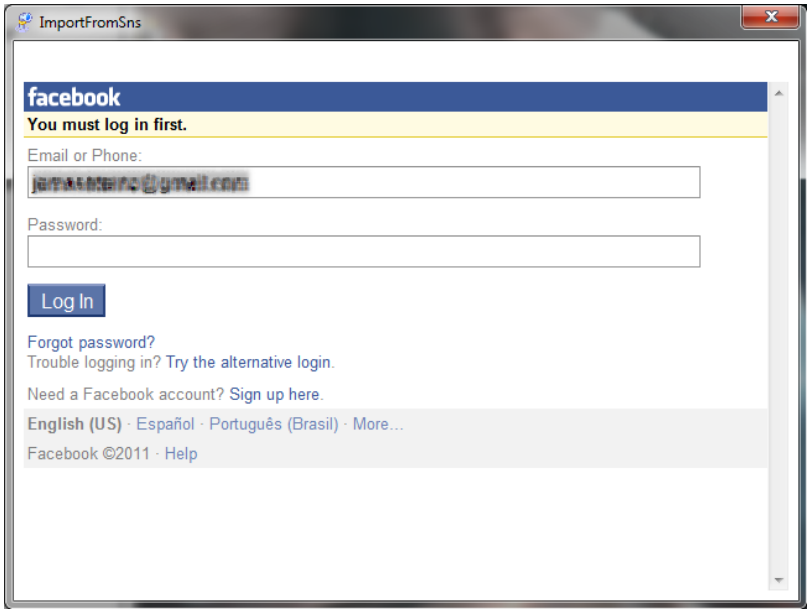

Figura 7. Prototipo W7, interfaz de acceso a Facebook.

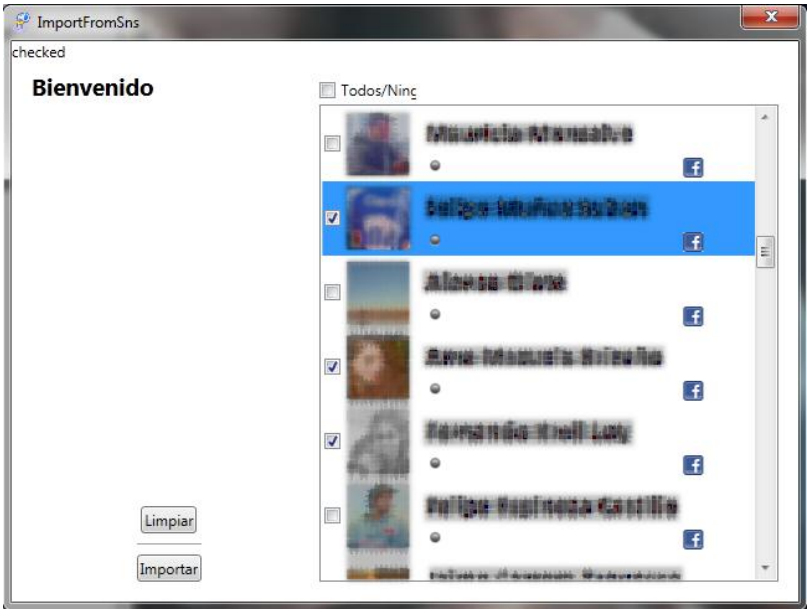

Figura 8. Prototipo W7, lista de amigos de Facebook.

En la Figura 7 se muestra la interfaz de acceso a Facebook a través del prototipo desarrollado. Una vez que se han ingresado las credenciales correctamente se muestra la pantalla presentada en la Figura 8, desde donde se pueden seleccionar los contactos que se desea importar desde la lista de todos los contactos de Facebook.

Figura 9. Prototipo W7, lista con contactos recién importados.

En la Figura 9 se muestra la pantalla principal del prototipo, con la lista de contactos antiguos y los recién importados.

c. HLMP para Windows 7

Como parte de los prototipos desarrollados se trabajó en una aplicación que comunicara dos dispositivos mediante el uso de HLMP. El prototipo se desarrolló sobre WPF para Windows 7, y tuvo como objetivo clarificar el funcionamiento de HLMP, extendiendo los Subprotocolos y Mensajes de dicha librería, y administrando los procesos de conexión y desconexión de la red local ad-hoc.

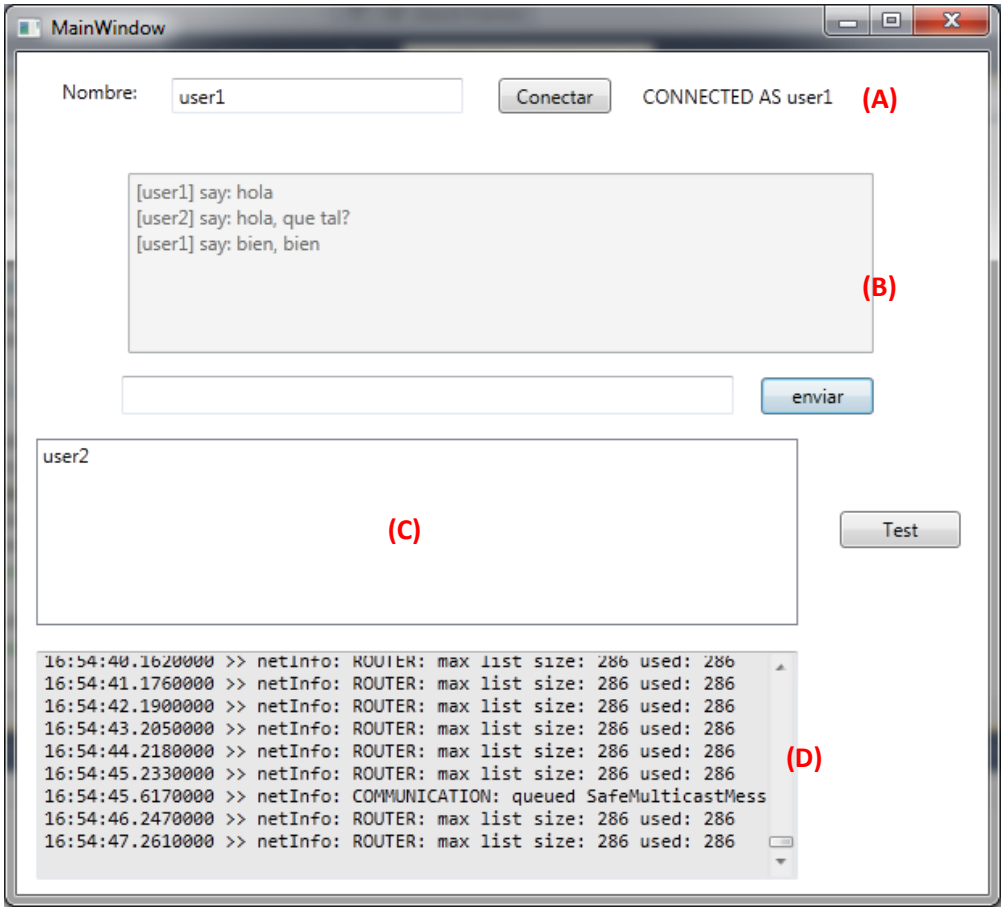

Figura 10. Prototipo de uso de HLMP en funcionamiento.

En la Figura 10 se muestra el prototipo en funcionamiento. En la parte superior de la ventana (A) se encuentra el campo para ingresar un *username*, el cual servirá para identificar al usuario en la red. A continuación se encuentran los controles de chat (B), donde el usuario ingresa los mensajes y observa el contenido de la conversación hasta el momento. Luego se muestra una lista con los otros usuarios en la red local (C), y finalmente se muestran los eventos del sistema en forma de *log* (D).

2.6. Resumen de Trabajos Relacionados

Como resultado del desarrollo de los prototipos (a, b, c) se logró lo siguiente:

- *Familiaridad con plataformas*: el desarrollo de ambos prototipos fue de gran ayuda para adquirir experiencia en las plataformas Silverlight y WPF. El alumno se entrenó en lenguajes y frameworks que no conocía, y aminoró el nivel de incertidumbre de su trabajo de memoria. Debido a este entrenamiento el planteamiento de una arquitectura

para el sistema y la cuantificación del tiempo y dificultad de la tarea a completar se efectuarán de mejor manera.

- *Familiaridad con HLMP*: el desarrollo de un prototipo que usara la librería HLMP para comunicar dispositivos, permitió comprender a cabalidad el uso de dicha librería, para así plantear de mejor manera el enfoque de trabajo al momento de desarrollar la aplicación final.
- *Interfaces y funciones*: el análisis de aplicaciones similares aporta con lineamientos generales de lo que se busca lograr con el desarrollo de la aplicación. Se usará la experiencia entregada por estos otros programas para ofrecer las funcionalidades de forma óptima.
- *Factibilidad de proyecto*: el alumno comprobó la factibilidad de realizar la aplicación sobre las plataformas deseadas, y clarificó la posibilidad de realizar un desarrollo lo más unificado posible para ambas versiones de la aplicación.

3. Concepción de la Solución

Como se revisó previamente, existe una oportunidad de desarrollar herramientas que extiendan la comunicación entre contactos de redes sociales hacia el plano físico, permitiéndoles concretar reuniones de forma casual mediante el uso de sus dispositivos móviles. Para ello, se desarrollará un software para Windows 7, usando el trabajo realizado en el capítulo 2 como base.

3.1. Requisitos de la Solución

Se desarrollará una solución de software que permita a los usuarios conectarse a través de sus dispositivos móviles a una MANET e importar una lista de contactos desde una o más SNS, notificando al usuario cuando se encuentre cerca de uno de sus contactos o reciba un mensaje, y permitiéndole emplear herramientas de comunicación entre ellos. Los requisitos de la solución son:

- **Seguridad y privacidad:** la solución debe proveer de un protocolo de comunicaciones seguro entre los usuarios, de tal modo que la información compartida entre ellos no sea visible para observadores no autorizados. De la misma forma, la información personal relacionada a los usuarios debe cumplir con las opciones de privacidad que ellos determinen.

- **Identidad de usuarios:** es deseable que la aplicación solicite credenciales de identificación a los usuarios, para garantizar que el dispositivo está siendo empleado por el dueño de las listas de contactos. Además, es deseable que la aplicación permita que un dispositivo pueda ser usado por más de una persona, separando la información de cada usuario de manera de garantizar la privacidad de la información personal.
- **Detección de usuarios:** la aplicación debe permitir detectar otros usuarios dentro de la red, y brindar las notificaciones respectivas cuando ocurran estos eventos. Además, debe permitirse al usuario ajustar su estado de visibilidad dentro del sistema (ej., ser "invisible" a un grupo de usuarios, y "disponible" para los demás).
- **Localización de usuarios:** la solución debe indicar la distancia relativa que lo separa de los miembros de su lista de contactos que se encuentren conectados a la MANET.
- **Soporte online y offline:** considerando la movilidad de los dispositivos, la aplicación debe soportar comunicaciones síncronas y asíncronas.
- **Canales de comunicación:** la aplicación debe proveer de canales síncronos y asíncronos de comunicación. Como mínimo, debe contarse con chat, mensajes de texto, solicitudes de reuniones físicas, y transferencia de archivos (según las necesidades del contexto de uso de la aplicación).
- **Notificaciones:** el sistema debe notificar al usuario mediante sonidos o alertas gráficas cuando ocurran eventos que ameriten la alerta. Entre estos eventos están la cercanía de otros usuarios, la recepción de mensajes u otras alertas propias de un contexto dado.
- **Compatibilidad con SNS:** la arquitectura debe ser extensible para soportar la interacción con distintas SNS. La aplicación debe soportar extensiones en cuanto a importar contactos desde diversas SNS, como alimentar dichas SNS con información.

Los requisitos mencionados serán completados mediante el uso de HLMP, que satisface varios de ellos (como Detección de usuarios, o algunos aspectos de Seguridad y Privacidad) o bien mediante el diseño e implementación del software. La aplicación llevará como nombre **Lukap** (basado en "look up"), y tendrá como plataforma de ejecución el framework .NET 4.

3.2. Diseño Arquitectónico

A continuación se presenta el diseño arquitectónico del software. Esta sección se divide entre el Diseño Arquitectónico Preliminar 4.a) y el Definitivo (4.b). El primero muestra las componentes necesarias para desarrollar una aplicación MESN (MANET-Enabled Social Network), según el

trabajo de Juan Rodriguez-Covili et al. [9]. En el segundo se plantea la arquitectura de la aplicación, considerando la plataforma de desarrollo y patrones de diseño adecuados.

a. Diseño Arquitectónico Preliminar

Se considerará como arquitectura referencial el trabajo propuesto por Juan Rodriguez-Covili et al. [9]. En dicho trabajo se presentan los antecedentes para extender las redes sociales tradicionales (IESN) a un plano físico, con una infraestructura soportada por una MANET. Finalmente, se plantea una arquitectura de referencia para el desarrollo de aplicaciones MESN.

A continuación se presenta la arquitectura de referencia (Figura 11), junto con la descripción de sus componentes:

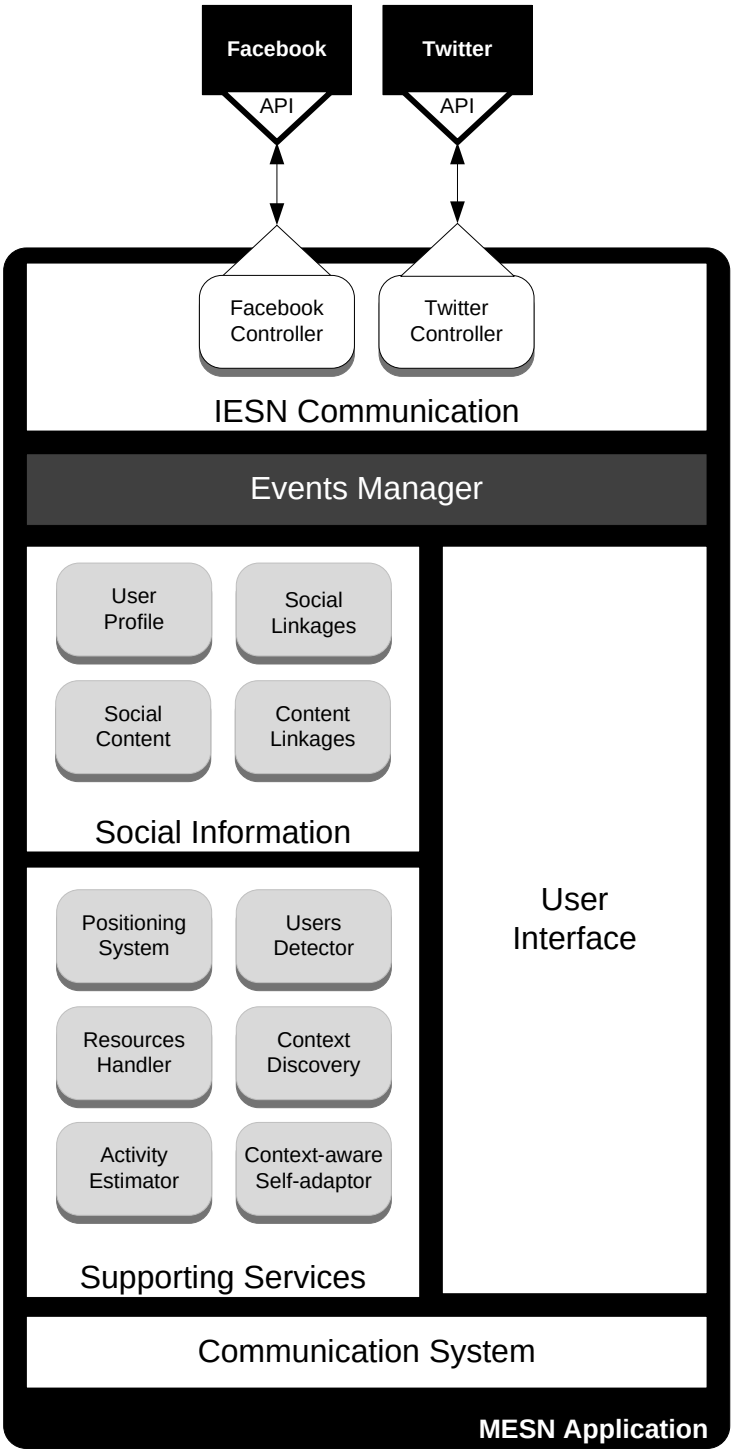

Figura 11. Arquitectura de referencia para una MESN

a) *IESN Communication*; esta capa está compuesta por distintos controladores, cada uno específicamente diseñado para interactuar con una IESN a través de su API.

b) *Events Manager*; capa que actúa como intermediaria entre los controladores de IESN y los servicios de la aplicación MESN. Es una interfaz única para administrar los *input* y *output* de las llamadas a los servicios.

c) *Communication System*; esta capa es la responsable de crear comunicación entre dispositivos a través de una MANET de forma apropiada. Esto implica proveer de mecanismos de comunicación entre dispositivos, formar redes MANET de forma automática y mantener la topología de la red dinámicamente. Para abordar este problema se empleará un trabajo previo de Rodríguez-Covili et al. [1].

d) *User Interface*; esta capa expone al usuario los servicios y alertas del sistema a través de las interfaces gráficas, auditivas o mecánicas que los dispositivos móviles provean.

e) *Supporting Services*; tiene funcionalidades que serán usados por otros servicios. Dichas funcionalidades son: identificar y notificar cambios de contexto a otros servicios de la aplicación, auto-adaptar la aplicación cuando dicho cambio sea considerable, inferir la actividad realizada por el usuario para ofrecerle servicios que pudieran ser requeridos, administrar los recursos privados y públicos de cada nodo (dispositivo), notificar si un usuario se encuentra o no en alcance para una interacción (junto con el estado de los usuarios dentro de la MANET), y ubicar a un dispositivo dentro de un escenario físico.

f) *Social Information*; está encargado de mantener y administrar la información relativa al usuario del sistema, sus contactos, vínculos sociales, etc. Además, es deseable que provea funcionalidades para autenticación de usuarios, permitiendo que un dispositivo almacene información de distintos usuarios.

b. Diseño Arquitectónico

Siguiendo los lineamientos presentados por la arquitectura preliminar, se plantea una arquitectura de software basada en las recomendaciones de desarrollo de Microsoft para WPF [19]. El modelo planteado se llama Model-View-ViewModel (MVVM), y se caracteriza por separar la aplicación en capas de Interfaz de usuario (*view*), Lógica de presentación (*viewmodel*) y Lógica de Negocios y Datos (*model*), todo esto usando herramientas propias de C# y del framework WPF.

Dichas herramientas propias facilitan la limpieza y el orden en el software, mientras que permite administrar fácilmente la visibilidad de objetos y servicios dentro del mismo, lo cual presenta ventajas desde el punto de vista de WPF. Sin embargo, para el desarrollo de la aplicación Lukap es necesario que los objetos que representan información personal,

lista de contactos, etc. sean visibles tanto para las interfaces como para HLMP, quien ejerce roles de comunicación entre dispositivos. La arquitectura planteada busca satisfacer ambas demandas de la mejor manera posible.

Varias de las componentes presentadas en (0) se encuentran incorporadas en HLMP, mientras que otras deben ser implementadas desde cero. A continuación se presenta la arquitectura de software de Lukap, indicando la correspondencia que cada componente tiene con los elementos de la arquitectura preliminar:

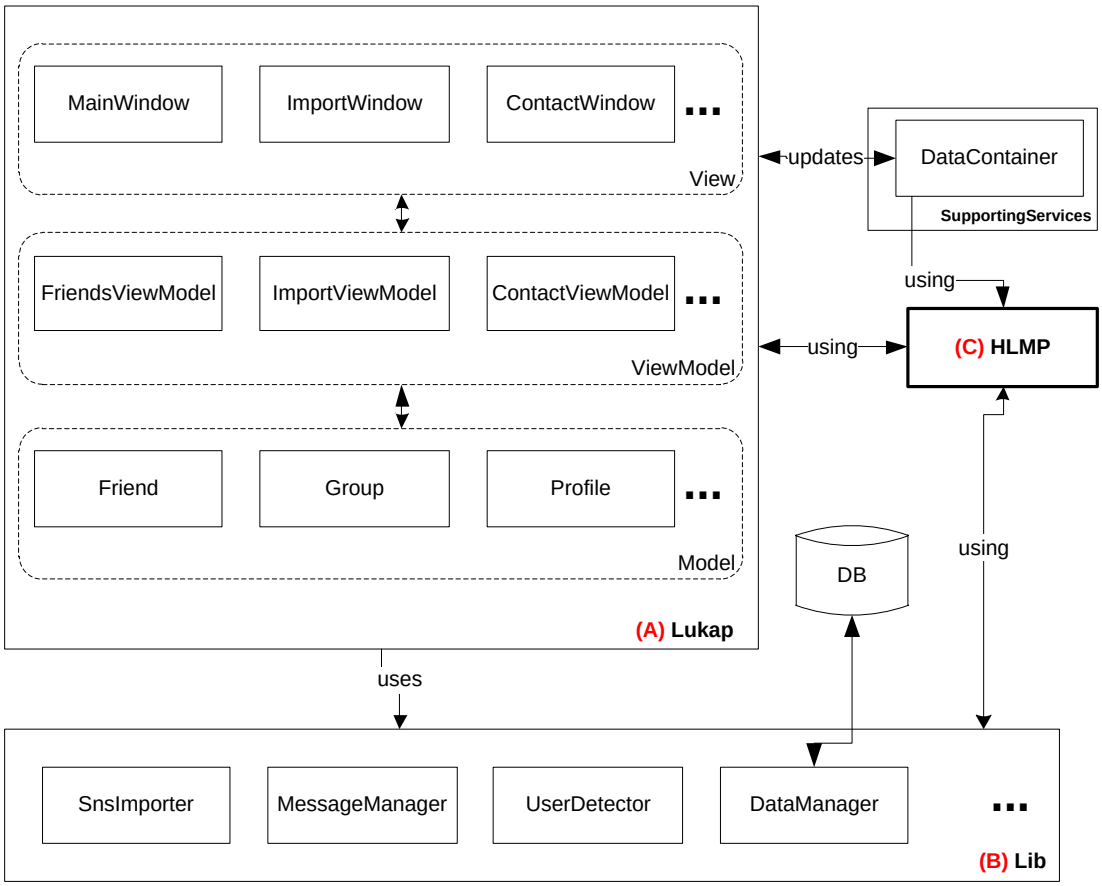

Figura 12. Arquitectura de software.

La Figura 12. Arquitectura de software. muestra la arquitectura de Lukap. La aplicación principal (A) sigue el patrón MVVM, donde cada interfaz gráfica existente en *View* (ej. *ImportWindow*) está asociada a una componente de *ViewModel* (ej. *ImportViewModel*). El framework WPF plantea que, mediante este mecanismo, la componente *View* procesa eventos gatillados por el usuario, mientras que en *ViewModel* se procesa la obtención y preparación de los datos a mostrar en la interfaz. La capa *Model* representa los objetos que las otras dos capas usarán para trabajar con los datos de la aplicación. En la implementación de Lukap, los datos se obtienen a través del módulo *DataManager*.

El esquema recién planteado optimiza el desarrollo de aplicaciones en WPF, pero tiene el inconveniente de que los objetos no persistentes de la aplicación se encuentran dispersos en varias componentes (ej., la lista de contactos con sus respetivos estados, mensajes, etc.). La componente *DataContainer*, perteneciente a *SupportingServices*, se encarga de recopilar todos estos objetos, y así facilitar la publicación y modificación por parte de las otras componentes. De esta forma, los eventos gatillados en la capa *View* o *ViewModel* se deben encargar de actualizar la información existente en *DataContainer*, y visceversa.

Por último, se la implementación del sistema de comunicaciones entre dispositivos (C), desarrollado sobre la librería HLMP. A continuación se presenta la correspondencia entre las componentes de la arquitectura de referencia (Figura 11. Arquitectura de referencia para una MESN y la arquitectura de software (Figura 12. Arquitectura de software.).

a) *IESN Communication*; La librería *SnsImporter* permite importar contactos desde las SNS. Para cada SNS existe una clase que hereda de SnsImporter (ej., *FacebookImporter*, *TwitterImporter*, etc.), y utiliza la API específica de la SNS para autenticar al usuario y luego importar su grafo social.

b) *Events Manager*; los datos importados por los controladores de SNS se almacenan directamente en la base de datos local. Para ello se usa la librería *DataAccessLayer*, que encapsula la lectura y escritura de la información social en métodos públicos.

c) *Communication System*; las funcionalidades de la capa de comunicaciones se encuentran satisfechas por HLMP.

d) *User Interface*; las interfaces de usuarios están a cargo de las componentes MVVM de la aplicación Lukap (Figura 12. Arquitectura de software.

e) *Supporting Services*; estas funcionalidades se encuentran satisfechas por HLMP

f) *Social Information*; la información social se almacena localmente en una base de datos. La administración de estos datos se hace mediante la componente *DataAccessLayer*, quien encapsula cada escritura y lectura de información en métodos públicos para la aplicación. La información no persistente se reúne en la componente *DataContainer*, quien la hace pública para los sistemas de comunicaciones (31c) y los servicios de soporte (e), así como también para las interfaces del software (Figura 12, A).

4. Diseño Detallado de la Solución

En el siguiente capítulo se presentará en detalle la solución desarrollada. La descripción se separará en el Diseño Detallado del Software, donde se profundizará en la arquitectura de software planteada y en los procesos implementados para cumplir con los requisitos funcionales de la solución; y en el Diseño Detallado de las Interfaces Gráficas, donde se explicará el funcionamiento de la aplicación desde el punto de vista del usuario.

4.1. Diseño Detallado del Software

Tal como se mencionó en la sección 3.2, la arquitectura del software deberá cumplir con incluir los módulos planteados en la arquitectura de referencia, lo que permitirá que la aplicación satisfaga los requisitos planteados.

A continuación se presentará el diseño del software en forma detallada, separado en tres aspectos complementarios: Procesos o mecanismos involucrados, Modelo de Datos, y Estructura de clases.

a) Procesos o Mecanismos:

Siguiendo los lineamientos de las redes sociales estudiadas en la sección 2.1.b, se plantearon mecanismos internos de comunicación entre cada instancia del software, los cuales sirvieron como base para la implementación de los requisitos funcionales de la solución. Estos procesos son: Identificación de Usuarios, Estatus de Usuario, e Invitación y Aceptación de Invitaciones.

- **Identificación de usuarios:** la aplicación debe permitir que el usuario importe[27] contactos desde distintas SNS, o bien accediendo con distintas cuentas a una misma SNS. Como consecuencia de lo anterior, cada usuario tendrá una "identidad" dentro de la red local, compuesta por un conjunto de *Perfiles* de SNS (los mismos que usó para importar contactos). La identificación directa mediante el uso de la librería HLMP no es factible, pues los "nodos" de la red (objetos *NetUser*) no son apropiados para encapsular y transmitir el conjunto de Perfiles de un usuario. Para solucionar este problema, se ideó un mecanismo que funciona en base al envío de mensajes de HLMP cada vez que: (a) la aplicación detecte un cambio en el conjunto de Perfiles del usuario, (b) la aplicación detecte un nuevo nodo en la red, o (c) la aplicación se desconecta de la red. En cada uno de estos casos se envía un mensaje a uno o todos los nodos de la red, notificándoles el nuevo conjunto de Perfiles del usuario. De esta forma, en todo momento cada nodo de la red conoce los Perfiles existentes

[27] La información de los contactos se descarga desde la SNS usando la API pública que ella provea, y luego se almacena localmente.

en cada uno de los otros nodos. Esta información será utilizada con posterioridad por otros módulos de la aplicación, por ejemplo, cuando sea necesario enviar un mensaje de chat a un Perfil en particular.

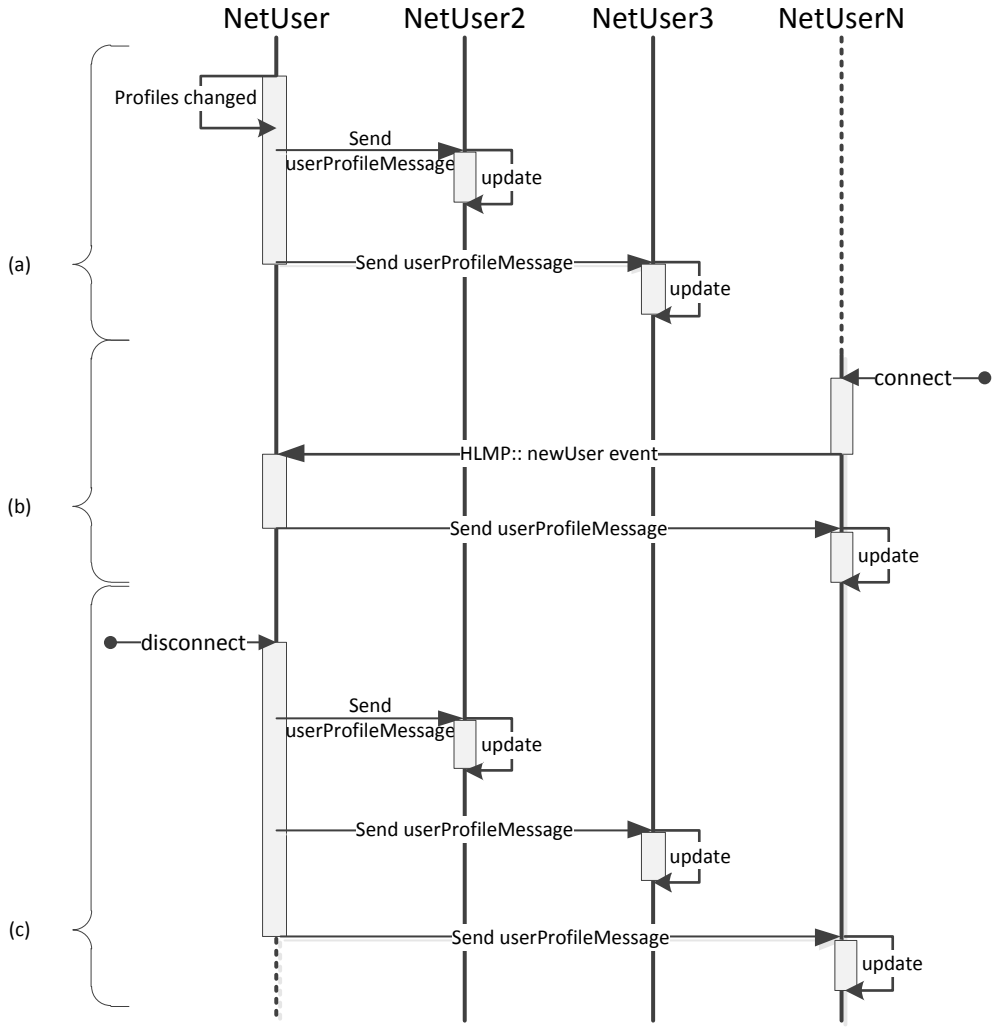

Figura 13. Diagrama de secuencia del mecanismo de identificación de usuarios dentro de la MANET.

- **Estatus de usuario:** la aplicación deberá permitir que los usuarios organicen sus contactos en grupos, donde a cada grupo podrá serle asignada una "visibilidad" desde las definidas en 2.1.b.iii (por ejemplo, asignar la visibilidad *disponible* al grupo "Familia", *ocupado* a "Colegas", e *invisible* al resto de los contactos). Debido a esto, el "estado" o "visibilidad" de un usuario en la red es un objeto complejo, que contiene la información de cada grupo, con su visibilidad particular, y de cada contacto en el grupo, con el Perfil que se usó para importarlo. Existen diversas formas de serializar toda esta información, teniendo presente que como resultado se debe obtener un objeto que sea transferible de un nodo a otro usando HLMP.

Para publicar el "estado" a los contactos en la red, se enviará un mensaje de HLMP a uno o más nodos cuando ocurran ciertos eventos en la aplicación. El mensaje tendrá la información necesaria y relevante (llamada "estado compuesto", o *ComposedStatus*) para que cada nodo actualice su lista de contactos. Los eventos son:

- Usuario se conecta o desconecta de la red local: se crea un *ComposedStatus* completo (i.e., con la información de todos los contactos). El mensaje se envía a todos los usuarios en la red.
- Usuario agrega o quita un contacto de un grupo: se crea un *ComposedStatus* con la información relevante para el contacto en cuestión. Se le envía el mensaje sólo a él.
- Usuario modifica visibilidad de un grupo: se crea un *ComposedStatus* con la información relevante para los miembros del grupo en cuestión. El mensaje se envía uno por uno a dichos miembros.
- Usuario elimina un grupo: similar a modificar visibilidad de un grupo, pero los miembros del grupo se desagrupan (quedan libres), y su visibilidad corresponde a la que el usuario configuró para sus contactos desagrupados.

A continuación se presenta un ejemplo del mecanismo de envío de estatus:

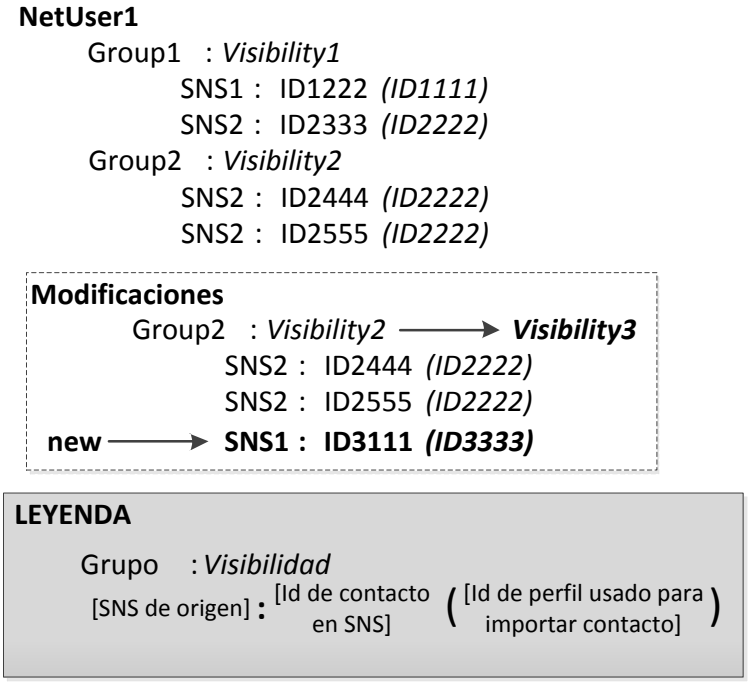

Figura 14. Ejemplo de lista de contactos de un usuario de Lukap, junto con un ejemplo de modificaciones a la lista.

Tabla 1. Ejemplo de eventos en la lista de usuarios y las acciones que ejecuta la aplicación para cada evento, en relación a los estatus de usuario.

Evento	Acción	Estado Compuesto
Conexión o desconexión de la red	Envío de *composedStatus* a todos los usuarios de la red	{SNS1:(ID1222,ID1111),SNS2:(ID2333,ID2222)}Visibility1, {SNS2:(ID2444,ID2222),SNS2:(ID2555,ID2222)}Visibility2
Agregar contacto a grupo (recuadro en Figura 14)	Envio de *composedStatus* a usuario(s) correspondiente(s) (*NetUser[SNS1,ID3111]*)	{SNS1:(ID3111,ID3333)}Visibility2
Cambiar visibilidad de grupo (recuadro en Figura 14)	Envío de *composedStatus* a miembros del grupo (*NetUser[SNS,SNS_ID]* para cada miembro de *Group2*)	{SNS2:(ID2444,ID2222),SNS2:(ID2555,ID2222), SNS1:(ID3111,ID3333)}Visibility3

- **Invitación y aceptación de invitaciones:** siguiendo los lineamientos de las SNS y programas estudiados en 2.1.b, se implementó un mecanismo para invitar contactos a formar parte del grupo de amigos, y para poder aceptar (o rechazar) las invitaciones recibidas. Con esto, al importar contactos, no aparecerán directamente en la lista de amigos, sino que quedarán "pendientes de aceptación". Dado que la aplicación conoce los Perfiles de cada nodo conectado a la red local, podrá enviar la invitación al destinatario correspondiente. Con esta acción, ambos contactos quedarán marcados como "pendientes" en la aplicación del otro.

  Si el destinatario de la invitación la acepta, se le indicará al remitente mediante un mensaje HLMP, y ambos usuarios podrán comunicarse a través de la red local. Por otra parte, si el destinatario rechaza la invitación, la información del remitente se eliminará de su aplicación, mientras que el remitente seguirá viendo al contacto como "pendiente".

  Por último, si un usuario se arrepiente de haber importado un contacto, podrá eliminarlo desde la lista de contactos "pendientes", independiente de si el sistema ya envió la invitación a dicho usuario a través de la red local.

b) Modelo de Datos:

A continuación se presenta el modelo de datos de la aplicación, seguido de una descripción de cada Tabla y de cada Campo de ellas.

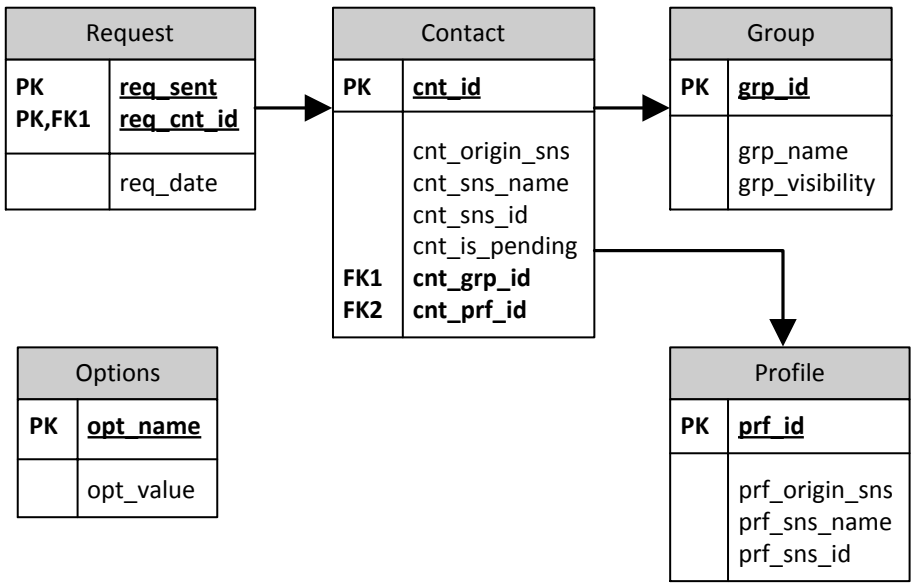

Figura 15. Modelo de datos de la aplicación.

- **Contact:** almacena la información de los contactos del usuario de la aplicación.

Tabla 2. Detalle de los campos de la tabla Contact.

Campo	Significado
cnt_id	identificador único de *Contact* dentro de la base de datos
cnt_origin_sns	red social de origen del *Contact*
cnt_sns_name	nombre del *Contact* dentro de su SNS de origen
cnt_sns_id	identificador único del *Contact* dentro de su SNS de origen
cnt_grp_id	identificador del *Group* al que pertenece el *Contact*.
cnt_prf_id	identificador del *Profile* usado para importar el *Contact*, o del *Profile* que recibió una invitación del *Contact*.
cnt_is_pending	indica si el *Contact* está o no pendiente de aceptación.

- **Group:** almacena la información de los grupos creados por el usuario para organizar sus contactos.

Tabla 3. Detalle de los campos de la tabla Group.

Campo	Significado
grp_id	identificador único de *Group* dentro de la base de datos
grp_name	nombre asignado por el usuario al *Group*
grp_visibility	visibilidad que el usuario asigna al *Group*

- **Profile:** almacena la información de los perfiles usados por el usuario para importar contactos o recibir invitaciones.

Tabla 4. Detalle de los campos de la tabla Profile.

Campo	Significado
prf_id	identificador único de *Profile* dentro de la base de datos
prf_origin_sns	red social de origen del *Profile*
prf_sns_name	nombre del *Profile* dentro de su SNS de origen
prf_sns_id	identificador único del *Profile* dentro de su SNS de origen

- **Request:** almacena la información relacionada a las invitaciones enviadas o recibidas por el usuario.

Tabla 5. Detalle de los campos de la tabla Request.

Campo	Significado
req_cnt_id	identificador único del *Contact* que envió o recibió la invitación
req_date	fecha cuando se envió o recibió la invitación
req_sent	indica si la invitación fue enviada o recibida por el usuario de la aplicación

- **Options:** almacena las preferencias de programa del usuario.

Tabla 6. Detalle de los campos de la tabla Options.

Campo	Significado
opt_name	nombre del parámetro de opción
opt_value	valor del parámetro de opción

c) Estructura de clases:

Según lo mostrado en 3.2.b, la aplicación se divide en cuatro componentes principales: *LukapGUI*, *Lib*, *SupportingServices* y *HLMP*.

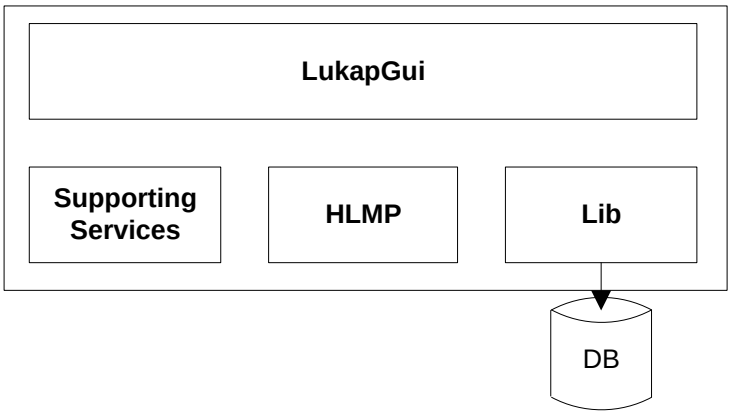

Figura 16. Componentes principales de la Arquitectura de Software

A continuación se describirá en detalle cada una de estas componentes, y la forma en que se comunican entre ellas.

i) LukapGUI: esta componente implementa la aplicación como tal, mediante el uso del framework WPF. Su arquitectura sigue un patrón de tres capas, llamado MVVM (*Model-View-ViewModel*). De forma general, cada Interfaz de usuario se implementa creando una Vista (*View*), que define las interfaces gráficas y sus funcionalidades; y un Controlador (*ViewModel*), que procesa los datos y los entrega a la Vista en el formato que se requiera. El Modelo (*Model*) cumple el rol de definir los objetos (y atributos) que serán usados por las Vistas y Controladores. Este planteamiento permite que la componente LukapGUI sea fácil de reutilizar en Windows Phone 7, de ser necesario.

El acceso a los datos por parte de los Controladores se hace a través de una Capa de Acceso a la Base de Datos, perteneciente a la macrocomponente **Lib**, que se revisará más adelante.

El detalle de cada módulo de *LukapGUI* se explicará a continuación, mientras que las componentes gráficas de cada interfaz se detallarán en la sección Diseño de Interfaces Gráficas (4.2):

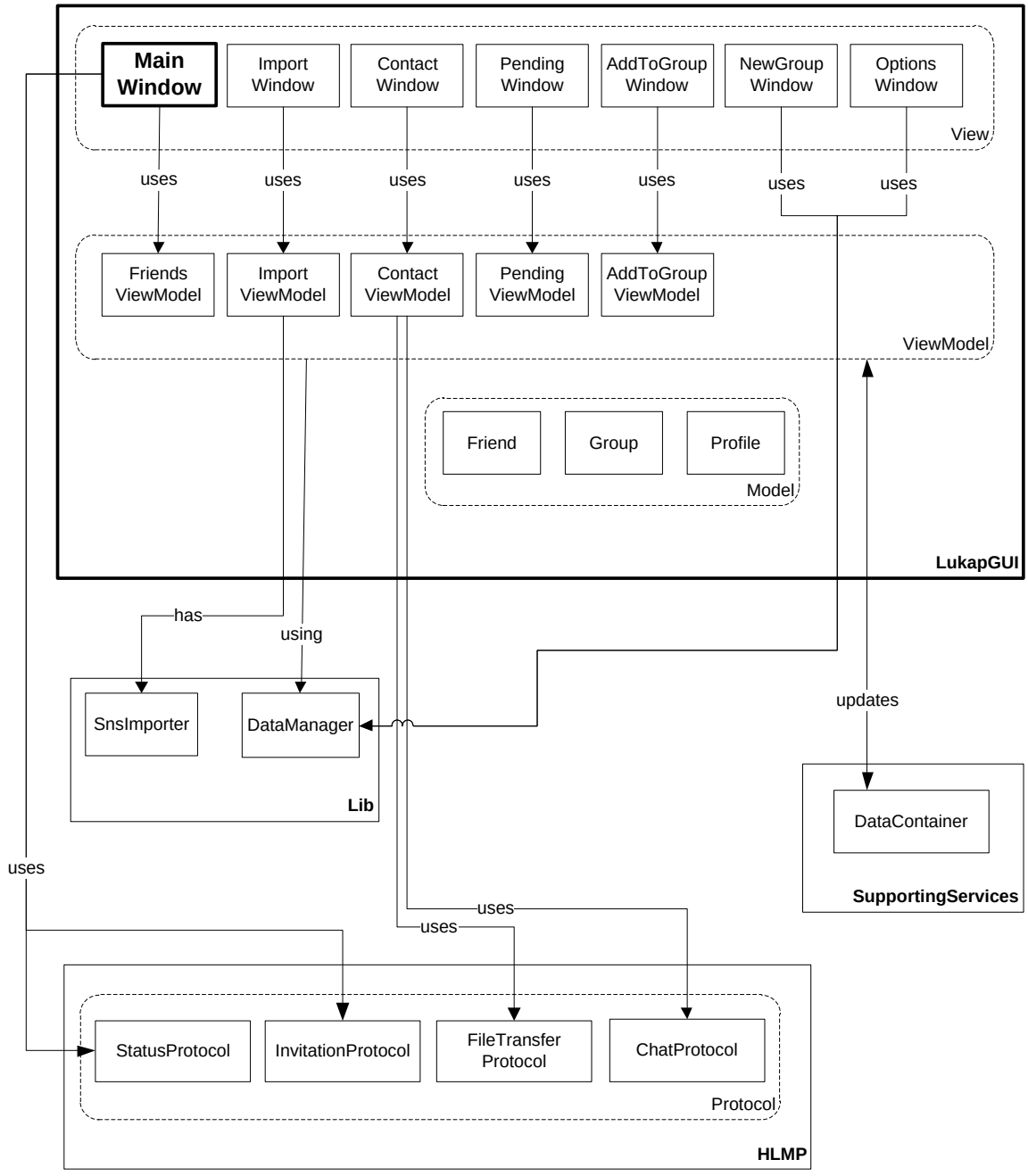

Figura 17. Arquitectura detallada de *LukapGUI*.

En la Figura 17 se observa la componente LukapGUI con todos sus módulos, las relaciones entre ellos y las relaciones entre ellos y las otras componentes.

- **MainWindow** es la ventana principal del programa. Su controlador es **FriendsViewModel**, el cual obtiene los datos usando *DataManager*, de la componente *Lib*. **MainWindow** es también el punto de partida del programa, y es el responsable de crear las otras ventanas y de inicializar la conexión a la Manet, usando HLMP. Por último, es el módulo responsable de enviar las Actualizaciones de Estado e Invitaciones de Usuario, usando *StatusProtocol* e *InvitationProtocol* respectivamente (ambos de la componente *HLMP*).
- **ImportWindow** es la ventana que permite al usuario ingresar a una SNS y luego importar los contactos que desee al programa. Su controlador es **ImportViewModel**, el cual posee un objeto *SnsImporter* (de la componente *Lib*). Mediante este objeto se realiza el acceso a los contenidos de las SNS.
- **ContactWindow** corresponde a cada una de las ventanas de chat entre usuarios. Su controlador es **ContactViewModel**, e implementa el patrón de diseño "Singleton", lo que permite que para cada par Usuario-Contacto pueda haber solo una ventana **ContactWindow** abierta. El controlador usa los protocolos *FileTransferProtocol* y *ChatProtocol* (ambos pertenecientes a la componente HLMP) para establecer comunicaciones con otros usuarios.
- **PendingWindow** corresponde a la ventana que muestra las invitaciones enviadas y recibidas. **AddToGroup** es la ventana que permite agregar un Contacto a un Grupo existente. Los controladores de ambas ventanas son **PendingViewModel** y **AddToGroupViewModel**, respectivamente.
- **NewGroupWindow** es la ventana que permite crear nuevos Grupos en la lista de contactos. **OptionsWindow** corresponde a la interfaz de opciones de programa, donde el usuario configura la aplicación según sus preferencias. Ambas ventanas cumplen con tareas específicas y sencillas, por lo que usan controladores.
- **Friend**, **Group** y **Profile** representan a los Contactos, Grupos y Perfiles de usuario, respectivamente. Además, poseen atributos adicionales a los obtenidos desde las SNS, o de la Base de Datos, que son propios del funcionamiento de la aplicación (p. ej.: estado de un usuario). Las tres clases son usadas por todas las interfaces y controladores.

ii) HLMP: esta componente implementa los protocolos y mensajes usados en la comunicación entre dispositivos, basándose en la librería HLMP. La Figura 18 muestra los protocolos (*Protocol*), junto con el conjunto de mensajes (*Messages*) administrado por

cada uno de ellos, y la clase interfaz que les permite manejar los mensajes recibidos (*handlerInterface*).

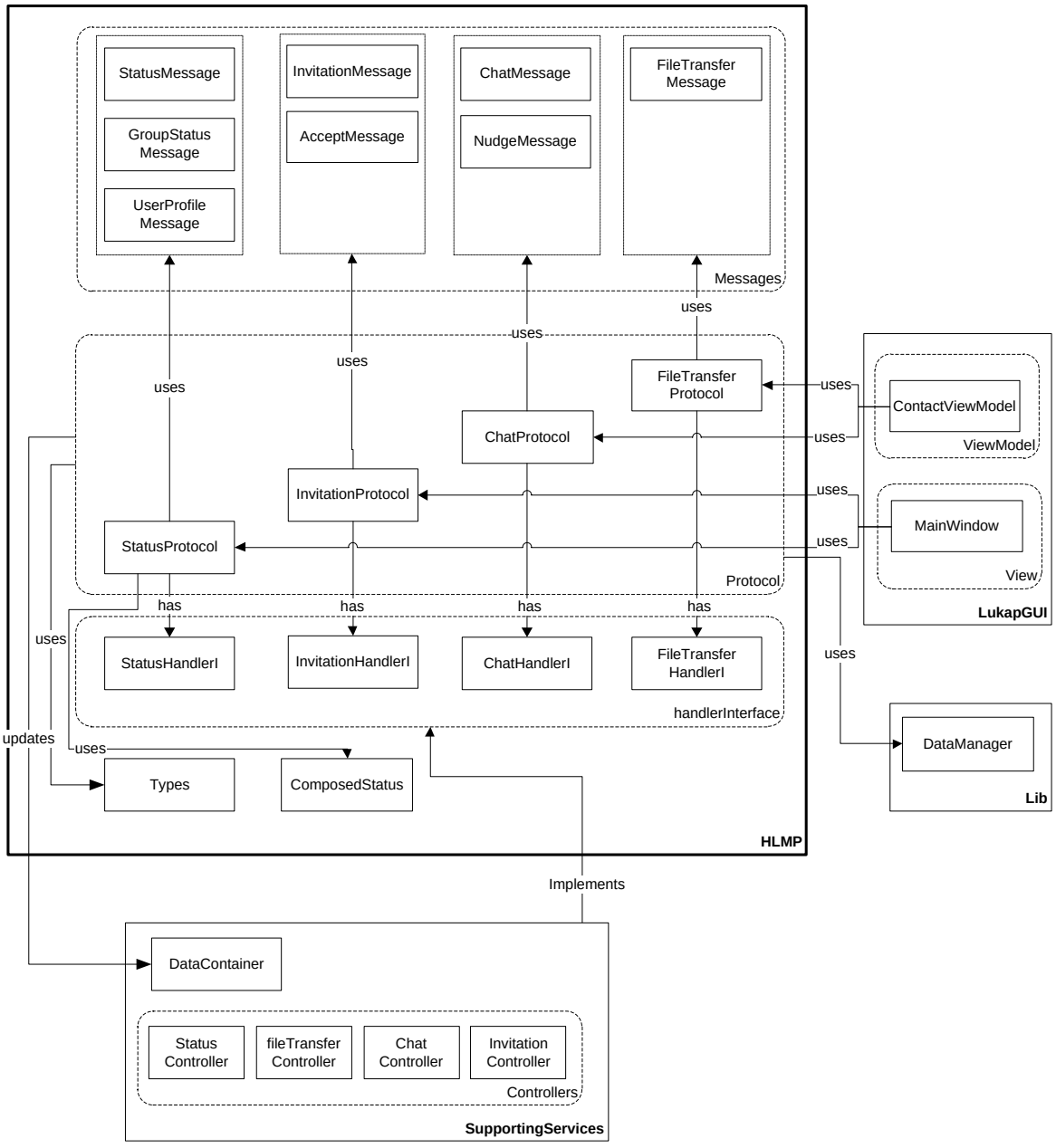

Figura 18. Arquitectura detallada de *HLMP*.

- **StatusProtocol:** protocolo encargado de enviar y recibir mensajes que permitan actualizar la lista de Perfiles existentes en cada dispositivo, y el Estado del usuario para cada Grupo. Es usado por *MainWindow*, de la componente *LukapGUI*.

  (a) StatusMessage y GroupStatusMessage: clases que implementan un mensaje de actualización de "estado" (o "visibilidad") destinado para enviarse a un usuario en particular o a todos los usuarios de la red, respectivamente. Contienen toda la

información relativa a los estados del usuario remitente, codificada por el módulo **ComposedStatus**.

(b) UserProfileMessage: clase que implementa un mensaje de actualización de Perfiles de Usuario (es decir, el conjunto de Perfiles que representan la identidad de un usuario en la Manet), destinado para enviarse a un usuario en particular.

(c) StatusHandlerI: interfaz cuyos métodos abstractos manejan las acciones a seguir cuando el protocolo **StatusProtocol** recibe un mensaje de otro dispositivo. Es implementada por *StatusController*, del módulo *Controllers*, de *SupportingServices*.

- **InvitationProtocol:** protocolo encargado de enviar y recibir mensajes que permitan a un usuario invitar a otros a formar parte de su Lista de Contactos, y que permitan aceptar dichas invitaciones. Es usado por *MainWindow*, de la componente *LukapGUI*.

(a) InvitationMessage: clase que implementa un mensaje de Invitación de Contacto. Se envía a un usuario en particular, indicando la SNS e identificador del remitente y destinatario de la invitación.

(b) AcceptMessage: clase que implementa un mensaje de Aceptación de Invitación. Se envía a un usuario en particular, indicando la SNS e identificador del remitente y destinatario de la invitación que fue aceptada.

(c) StatusHandlerI: interfaz cuyos métodos abstractos manejan las acciones a seguir cuando el protocolo **InvitationProtocol** recibe un mensaje de otro dispositivo. Es implementada por *InvitationController*, del módulo *Controllers*, de *SupportingServices.*

- **ChatProtocol:** protocolo encargado de enviar y recibir mensajes que permitan la comunicación instantánea o asíncrona entre dispositivos, ya sea por mensajes de texto, "toques", o bien Peticiones de Reunión cara-a-cara. Es usado por *ContactViewModel*, de la componente *LukapGUI*.

(a) ChatMessage: clase que implementa un mensaje de texto, destinado a un usuario en particular. El mensaje se envía de forma instantánea, si el destinatario está conectado a la red, o se encola hasta que el destinatario se conecte.

(b) NudgeMessage: clase que implementa un mensaje de "toque" o Petición de reunión cara-a-cara, dependiendo de los parámetros de creación del mensaje. Se envía a un usuario en particular.

(c) ChatHandlerI: interfaz cuyos métodos abstractos manejan las acciones a seguir cuando el protocolo **ChatProtocol** recibe un mensaje de otro dispositivo. Es implementada por *ChatController*, del módulo *Controllers*, de *SupportingServices.*

- **FileTransferProtocol:** protocolo encargado de enviar y recibir mensajes que permitan el envío de archivos entre dispositivos. Es usado por *ContactViewModel*, de la componente *LukapGUI*.
  (a) FileTransferMessage: clase que implementa un mensaje que contiene un archivo serializado. Se envía a un usuario en particular.
  (b) FileTransferHandlerI: interfaz cuyos métodos abstractos manejan las acciones a seguir cuando el protocolo **FileTransferProtocol** recibe un mensaje de otro dispositivo. Es implementada por *FileTransferController*, del módulo *Controllers*, de *SupportingServices.*

- **Types:** clase que contiene valores constantes, usados por los Protocolos para procesar los mensajes recibidos.

- **ComposedStatus:** clase que serializa la información concerniente al "estado" o "visibilidad" de un usuario en la red. Dependiendo de si el mensaje a enviar es **StatusMessage** o **GroupStatusMessage**, la clase procesa la información de la lista de contactos completa, o bien de un subconjunto de ella.

iii) Lib: ofrece librerías usadas por las otras componentes del sistema. En la Figura 19 se muestran los módulos pertenecientes a **Lib**, las relaciones entre ellos y los usos que reciben por parte del resto del software.

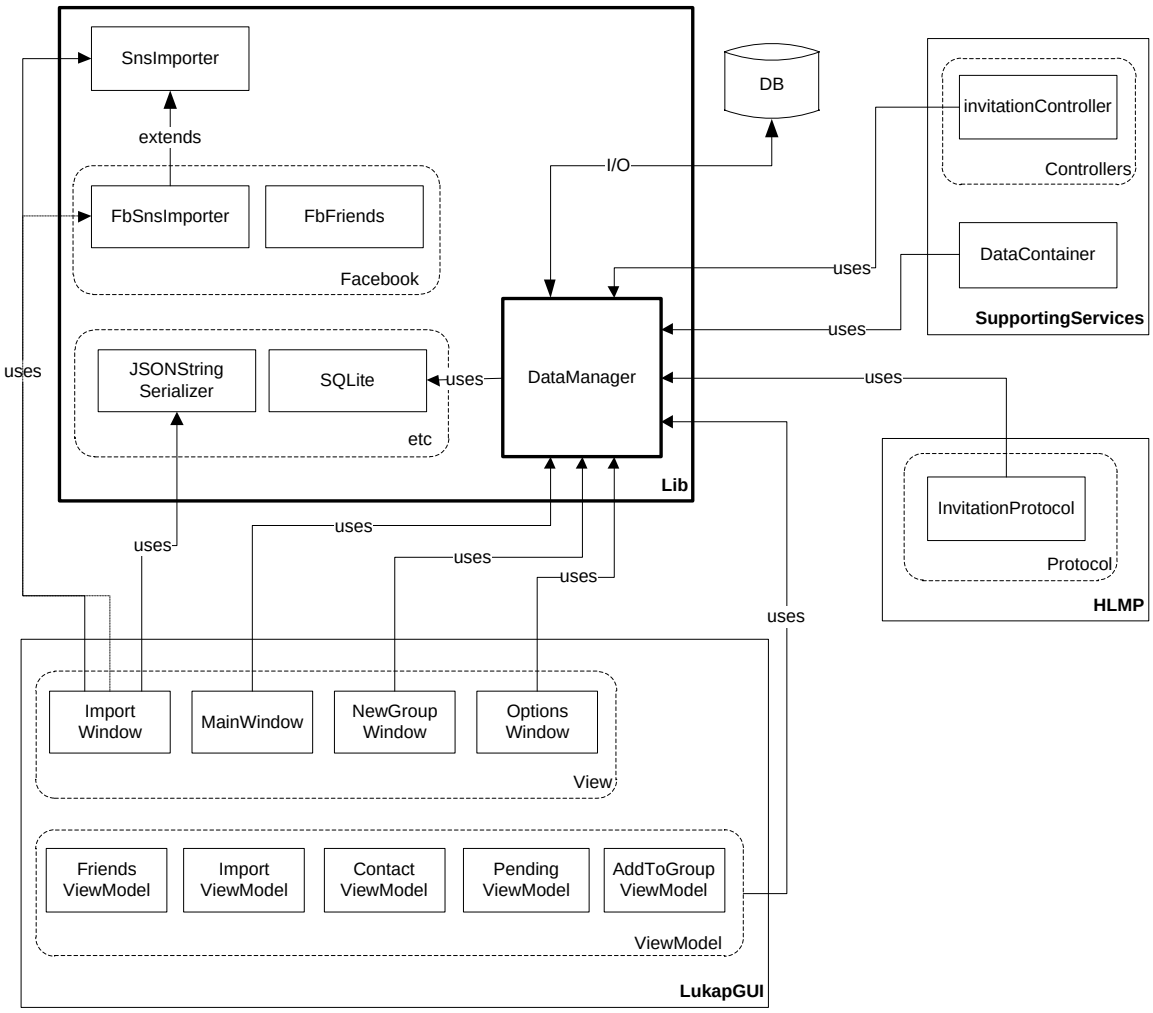

Figura 19. Arquitectura detallada de *Lib.*

- **DataManager**: esta clase actúa como capa de acceso a la Base de Datos por parte del resto de la aplicación. Encapsula en varias funciones todas las lecturas y escrituras a la Base de Datos que el software necesita realizar, y permite que el mecanismo de almacenamiento le sea transparente. En la implementación actual, se usa *SQLite* como motor de base de datos, y se emplea la librería de C# **SQLite** para acceder a ella.
- **JSONStringSerializer**: librería usada por *ImportWindow* (perteneciente a *LukapGUI*) que permite *parsear* los datos enviados por la API de Facebook, de tal modo de importar los datos del usuario correctamente.
- **SnsImporter:** interfaz cuyos métodos abstractos y atributos permiten completar la obtención de datos desde una SNS. Implementa el acceso mediante el protocolo OAuth 2.0. Las clases **FbSnsImporter** y **FbFriends** implementan a **SnsImporter** para obtener datos desde Facebook.

iv) **SupportingServices:** esta componente ofrece servicios usados para establecer comunicación entre las otras componentes del sistema. En la Figura 20 se muestran los módulos de **SupportingServices**, las relaciones entre éstos y las relaciones con los otros módulos del software.

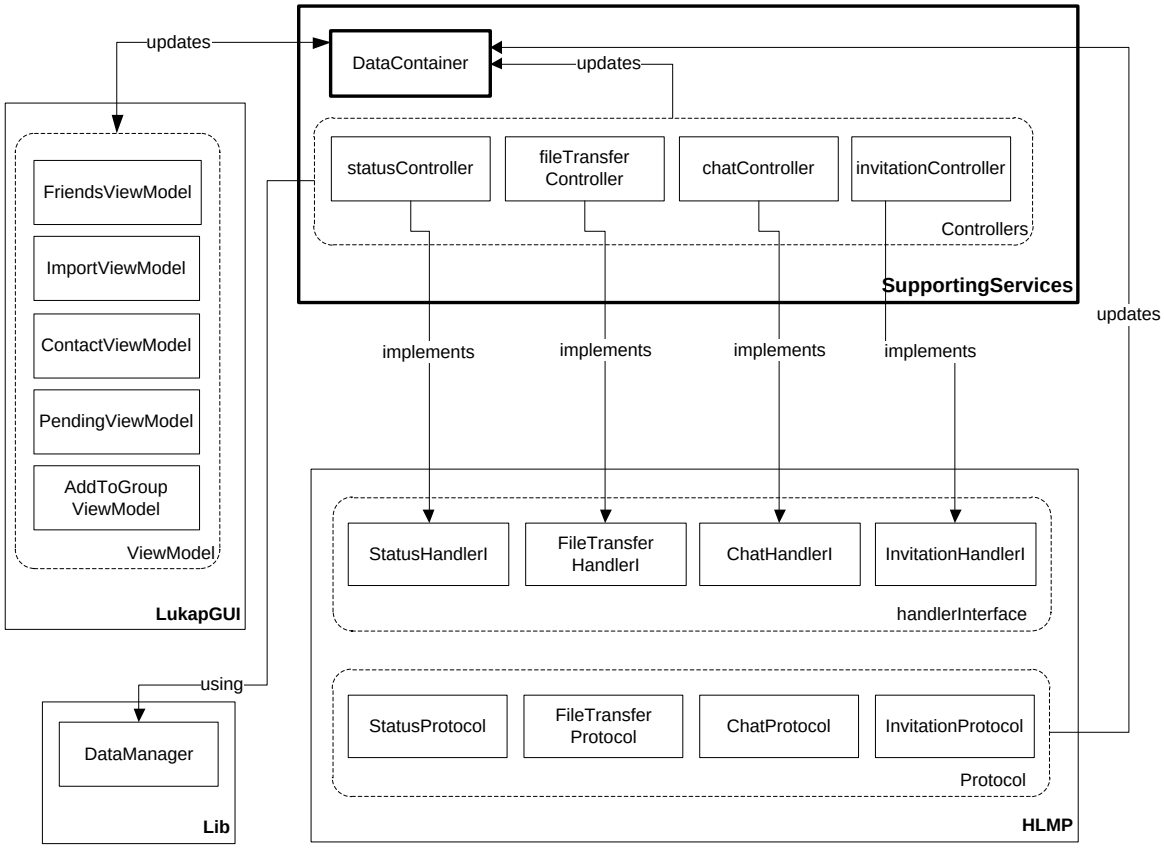

Figura 20. Arquitectura detallada de *SupportingServices*.

- **DataContainer:** esta clase recopila las variables relativas a la Lista de Contactos del usuario (visibilidad, grupos, mensajería, ventanas de chat abiertas, etc.). Permite que tanto el usuario, mediante las interfaces gráficas, como la aplicación misma puedan modificar dichas variables, y que los cambios realizados se propaguen a través del sistema, gatillando notificaciones o envíos de mensajes. Este módulo es fundamental en la coordinación interna de la aplicación, y permite combinar el patrón de diseño de *LukapGUI* (patrón MVVM) con la arquitectura planteada en la componente *HLMP*.
- **Controllers:** los módulos **statusController**, **fileTransferController**, **chatController** e **invitationController** implementan las interfaces *StatusHandlerI*, *FileTransferHandlerI*, *ChatHandlerI* e *InvitationHandlerI*, respectivamente. Procesan los mensajes recibidos por sus Protocolos respectivos, y generan cambios en **DataContainer** (ej.: llega un *StatusMessage*, es procesado por **statusController**, quien

modifica el "estado" del contacto correspondiente en la lista de contactos de **DataContainer**).

4.2. Diseño de Interfaces Gráficas

A continuación se presentarán las interfaces gráficas de Lukap, describiendo las partes que conforman cada una de ellas. Cabe mencionar que la información concerniente a los usuarios se ocultará, para proteger sus privacidades.

1) **MainWindow:** ventana principal de la aplicación. Muestra la lista de contactos del usuario.

Figura 21. Ventana principal de Lukap.

Los elementos que se observan en la Figura 21 son:

A) Cantidad de invitaciones recibidas que no han sido aceptadas.

B) Buscador de contactos: permite filtrar la lista de contactos con la palabra ingresada.

C) Indicador de conexión a la Manet: muestra si la aplicación está Conectada, Conectándose o Desconectada de la red local.

D) Lista de Contactos: muestra los contactos que han sido agregados exitosamente a la aplicación. Cada contacto muestra una foto de perfil, el nombre dentro de la SNS, el "estado" (círculo de color), la distancia relativa al usuario y la Red Social de origen.

E) Visibilidad de grupo: corresponde a la visibilidad que el usuario asigna a todos los miembros del grupo.

F) Estado de contacto: el color del círculo indica el estado (o visibilidad) del contacto. El símbolo adyacente indica la cercanía relativa (medida en "saltos") del contacto.

En la barra de menú de **MainWindow** se encuentra lo siguiente:

- *Conexión*: Permite conectarse o desconectarse de la Manet.
  - Conectar
  - Desconectar
- *Amigos*:
  - Invitaciones Pendientes: abre la ventana de Contactos Pendientes, es decir, a quienes el usuario ha invitado, o de quienes ha recibido invitaciones.
  - Filtro: muestra opciones de filtro para la lista de contactos. Los filtros permiten mostrar los miembros por Grupo, por Estado o por Red Social.
- *Configuración*:
  - Crear nuevo grupo.
  - Importar Contactos: abre la ventana para Importar Contactos desde una SNS.
  - Opciones: abre la ventana de Opciones de programa.
- *Ayuda*:
  - Ayuda.
  - Acerca de.

2) **ImportWindow:** interfaz que permite al usuario conectarse a una SNS, y luego seleccionar un subconjunto de sus contactos para importarlos a la aplicación.

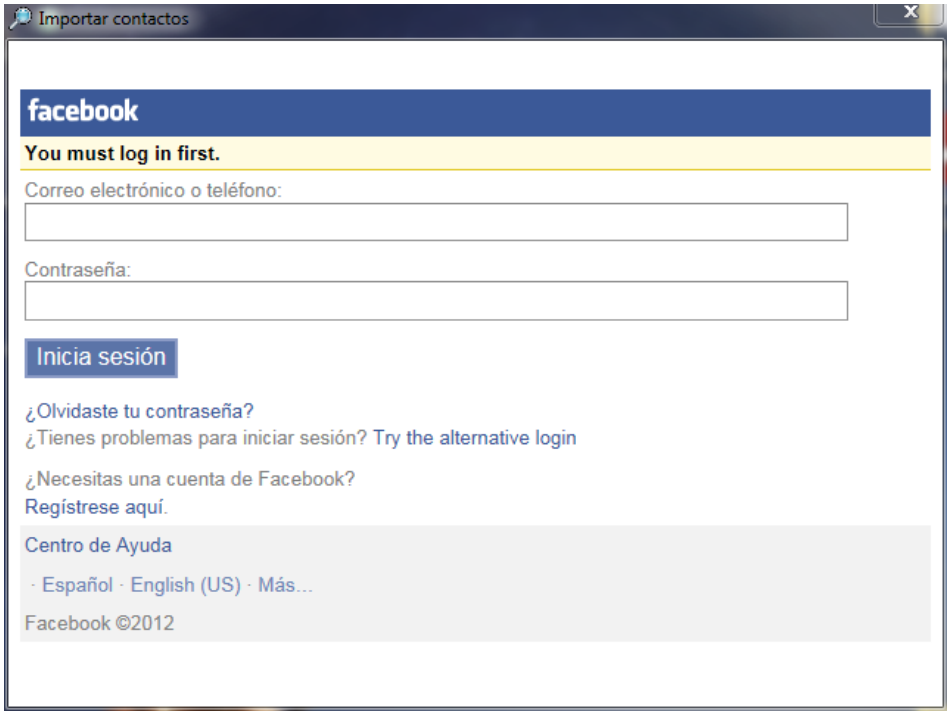

Figura 22. interfaz de ImportWindow para conectarse a la SNS.

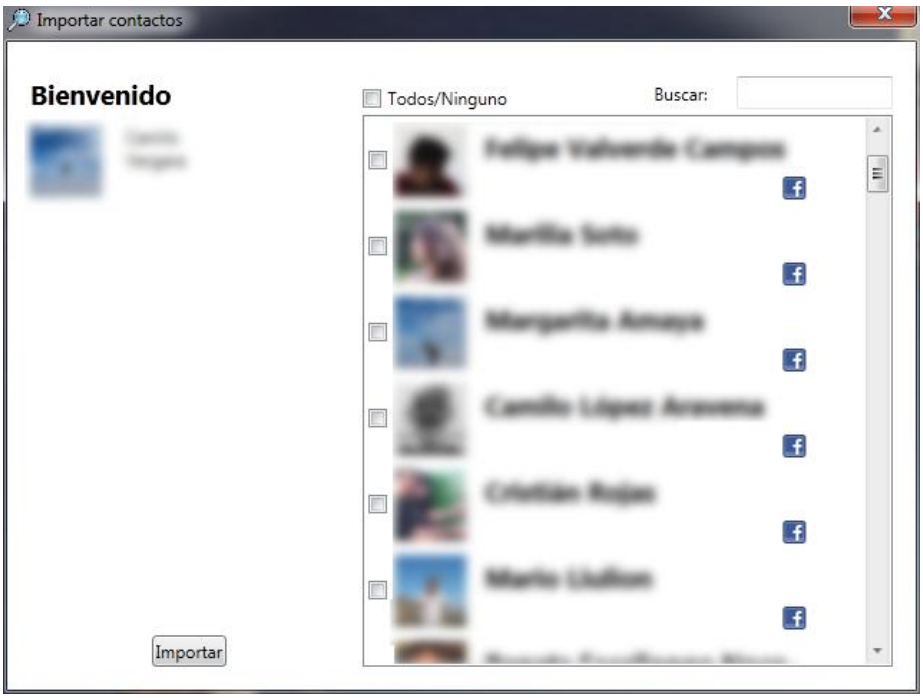

Figura 23. Interfaz de ImportWindow para seleccionar contactos a importar.

En la Figura 22 se observa la interfaz mostrando el diálogo de autorización de la SNS. Una vez que el usuario ingresa sus credenciales, ImportWindow muestra el contenido de la Figura 23. En ese momento, el usuario puede seleccionar los contactos que desee, para

luego presionar el botón “Importar”. Con esto, los contactos quedan almacenados como “pendientes”, a la espera de que la aplicación envíe las invitaciones.

3) **PendingWindow:** esta interfaz muestra los contactos “pendientes” del usuario. Permite revisar los contactos importados, y (si se desea) eliminarlos mientras siguen pendientes. También permite revisar la lista de invitaciones recibidas, y aceptarlas, rechazarlas, o no hacer nada con ellas.

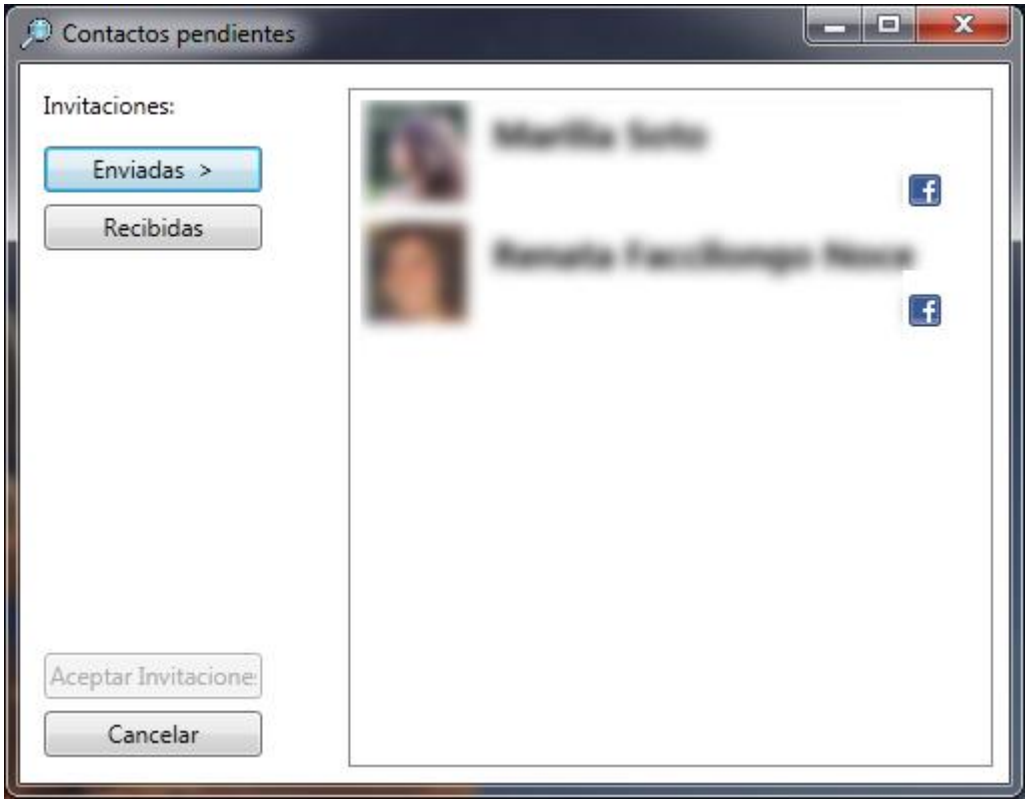

Figura 24. interfaz PendingWindow mostrando los contactos importados.

En la Figura 24 se observa la lista de los contactos importados por el usuario. Es posible eliminar cada contacto abriendo el menú contextual sobre su ítem correspondiente de la lista, lo que mostrará la opción “Eliminar contacto”. Esta acción se puede realizar independientemente de si Lukap ya envió la invitación a dicho usuario.

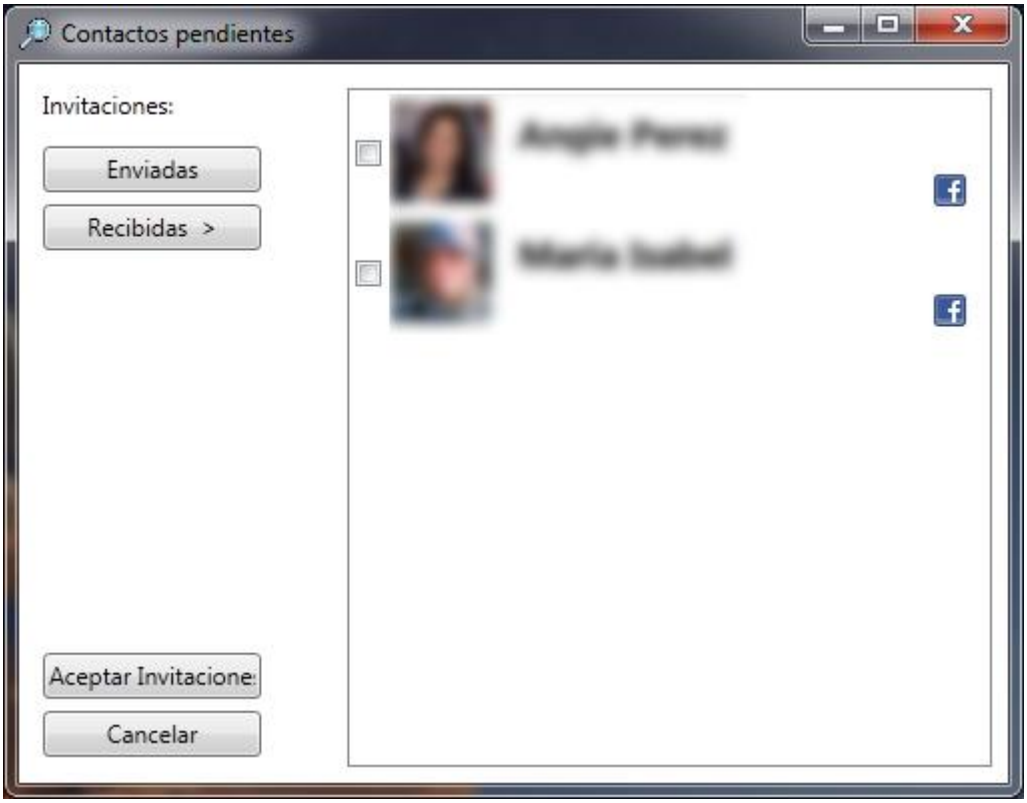

Figura 25. interfaz PendingWindow, mostrando las invitaciones recibidas.

En la Figura 25 se observa a la ventana PendingWindow mostrando la lista de invitaciones recibidas por el usuario. Al igual que con la lista anteriormente descrita, es posible eliminar estas invitaciones mediante el menú contextual de la lista. Además, es posible aceptar las invitaciones, seleccionando sus casillas, y luego presionando el botón "Aceptar Invitaciones".

4) **AddToGroupWindow** y **CreateGroupWindow:** estas interfaces permiten agregar a un contacto a un grupo y crear un nuevo grupo, respectivamente.

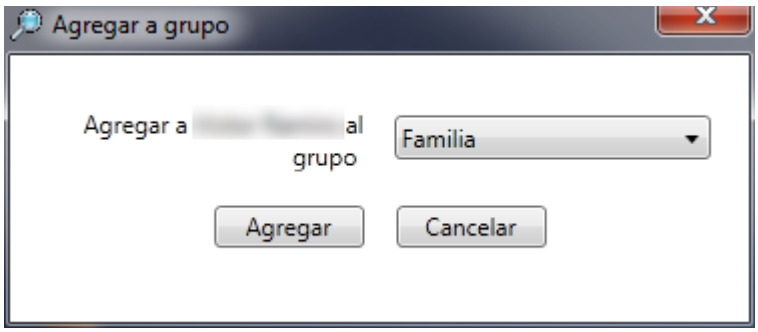

Figura 26. interfaz AddToGroupWindow.

En la Figura 26 se observa la ventana que permite agregar un contacto a un grupo existente. La Figura 27 muestra la ventana que permite al usuario crear un nuevo grupo.

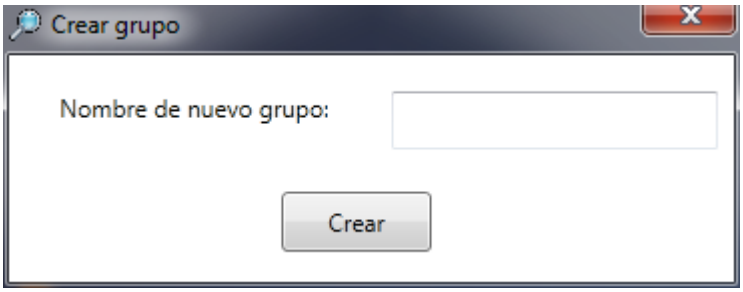

Figura 27. interfaz CreateGroupWindow.

5) **ContactWindow:** esta interfaz corresponde a la ventana de chat entre dos usuarios de la red. Además, posee opciones de administración del contacto con quien se mantiene la ventana de chat, y permite enviar los distintos tipos de mensajes implementados en Lukap.

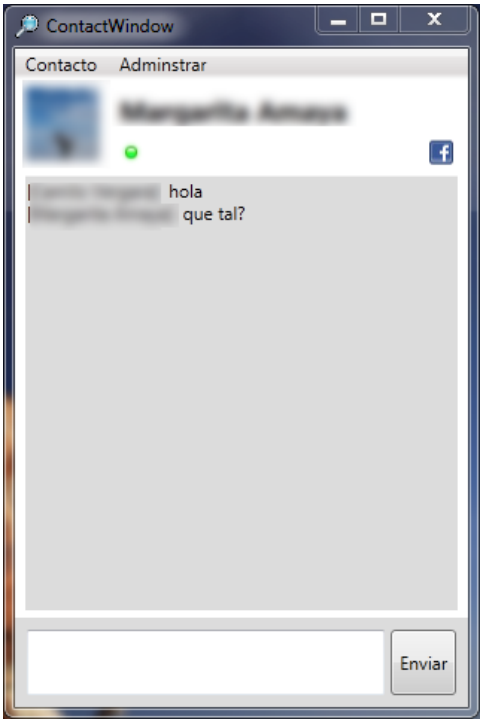

Figura 28. interfaz ContactWindow.

En la Figura 28 se observa la información del contacto con quien se mantiene la conversación, los mensajes enviados hasta el momento entre ambos contactos, y el campo de texto que permite ingresar mensajes nuevos.

En la barra de menú de **ContactWindow** se encuentra lo siguiente:

- *Contacto*: lista de posibles mensajes para enviar al contacto.
  - Enviar mensaje: permite enviar un mensaje de texto asíncrono, en caso de que el contacto se encuentre desconectado.
  - Enviar toque: permite enviar un "toque" al contacto. El "toque" gatilla una notificación al recibirse, y no tiene más contenido que ése.
  - Enviar petición de reunión: similar al "toque", pero su contenido invita al contacto a concretar una reunión cara-a-cara.
  - Enviar archivo: abre diálogo de selección de archivo, para luego enviarlo a través de la MANET.
- *Administrar*: opciones de administración del contacto.
  - Ver historial de chat: muestra el historial de conversaciones de chat mantenidas entre ambos contactos (sólo si el usuario tiene habilitada la opción de Almacenar Historiales de Chat).
  - Agregar a grupo / Quitar de grupo: ambas opciones son excluyentes. Sólo se muestra la que corresponda, dependiendo de si el usuario pertenece o no a un grupo.
  - Eliminar Contacto: elimina al contacto de la lista de contactos del usuario.

5. Evaluación Preliminar

La aplicación Lukap, desarrollada en el trabajo de memoria, se ejecuta sobre el sistema operativo Windows 7, habiendo instalado el framework *.Net 4*, versión completa. El software permite a los usuarios importar los contactos que desee desde Redes Sociales, para luego comunicarse con ellos a través del mismo software, mediante cualquiera de los canales de comunicación que éste provee. El objetivo principal de la aplicación es notificar a los usuarios de la cercanía física de los contactos que ha importado, para así poder concretar reuniones cara-a-cara con ellos.

Luego de una revisión técnica de las funcionalidades desarrolladas, se puede asegurar que Lukap (implementado conforme a la arquitectura de software planteada en el capítulo 4), cumple con los requisitos de software listados en la concepción de la solución:

- **Seguridad y privacidad:** El empleo de HLMP como librería de comunicaciones garantiza que la mensajería entre usuarios es segura, y que la información personal de cada usuario no

es accesible a través de la red. Los mecanismos de Identificación de Usuarios y Actualización de Estados fueron diseñados para cumplir con el mismo nivel de seguridad y privacidad.

- **Detección de usuarios:** La detección de dispositivos en la red se realiza mediante HLMP, y la detección de contactos se realiza con el Mecanismo de Identificación de Usuarios. La implementación de arquitectura planteada permite gatillar notificaciones cada vez que se detecta un cambio en los contactos conectados a la red.
- **Localización de usuarios:** La librería HLMP permite conocer un rango relativo de distancia entre los usuarios mediante la cantidad de "saltos" existentes entre ellos. Un "salto" representa que dos usuarios no se encuentran en rango de comunicaciones, pero si pueden hacerlo si existe al menos otro usuario que sirva como puente comunicacional entre sus dispositivos.
- **Soporte online y offline:** La aplicación permite enviar mensajes a usuarios de forma asíncrona, es decir, cuando uno o ambos usuarios no se encuentran conectados. En el caso en que ambos se encuentren conectados, las comunicaciones se realizan de forma instantánea.
- **Canales de comunicación:** La aplicación provee de todos los canales de comunicación planteados como requisitos (mensajería instantánea, mensajes de texto asíncronos, solicitud de reunión, y transferencia de archivos).
- **Notificaciones:** El diseño de las interfaces de usuario y el análisis de programas similares tuvieron como resultado encontrar maneras apropiadas y oportunas para notificar de todos los eventos relevantes del sistema al usuario.
- **Compatibilidad con SNS:** La arquitectura se planteó para que fuera sencillo dar soporte a nuevas SNS, cuando sea requerido. La información relativa al origen de cada contacto importado se muestra de forma clara en la interfaz gráfica.

Como resultado del trabajo realizado se espera que la aplicación sea una herramienta útil para concretar reuniones físicas, en contextos laborales o casuales, donde la cercanía entre personas no sea fácil de detectar sin algún tipo de asistencia. Además, el software debiera ser útil como medio para mantener conversaciones de chat, enviar mensajes asíncronos de texto (similares a correos electrónicos), y enviar archivos entre dispositivos.

Por último, se espera que los usuarios sean capaces de usar la aplicación sin entrenamiento previo en ésta, dado que las interfaces y funcionalidades se diseñaron siguiendo los lineamientos planteados por Redes Sociales y Aplicaciones Similares que gozan de gran popularidad, y que de una u otra forma han definido componentes estándares con los cuales la mayoría de los usuarios se sienten familiarizados.

6. Conclusiones y Trabajo a Futuro

El objetivo del trabajo de memoria era construir una herramienta que permitiera interactuar cara-a-cara a personas que ya interactuaban de forma virtual a través de Redes Sociales. Para ello se desarrolló una aplicación para Windows 7, llamada Lukap, que permitiera a sus usuarios (a) importar un subconjunto de sus contactos de Redes Sociales, (b) comunicarse con ellos a través de distintos canales (chat, mensajes offline, envío de archivos) y (c) enterarse de la cercanía de sus contactos, para así concretar reuniones físicas con ellos.

El desarrollo de Lukap se abordó desde diferentes ángulos, los cuales constituyeron la etapa investigativa del trabajo de memoria. Entre ellos destacan dos, por haber presentado mayor complejidad.

El primer aspecto en cuestión se refiere a la implementación de aplicaciones para los sistemas operativos considerados desde un inicio (Windows 7 y Windows Phone 7). Se desarrollaron prototipos funcionales para ambas plataformas (descritos en 2.5), con lo que se alcanzó el conocimiento requerido para el desarrollo de la aplicación final. También se estudiaron las posibilidades de patrones de diseño en ambas plataformas, con el fin de diseñar una arquitectura modificable, mantenible y extensible para la aplicación final.

El segundo aspecto consistió en obtener familiaridad con el empleo de la librería HLMP. Para ello se consultó el trabajo de Rodríguez-Covili [1], y luego se desarrolló un prototipo funcional que usara la librería (descrito en 2.5.c). Más adelante, al diseñar el software, se hizo evidente la necesidad de implementar nuevos mecanismos de detección de usuarios que funcionaran en base a los provistos por HLMP (detallados en 4.1.a)), lo que aumentó la dificultad de establecer la comunicación necesaria entre los dispositivos que usaran la aplicación final.

Al completar las investigaciones mencionadas, el desarrollo de la aplicación se limitó al sistema operativo Windows 7, puesto que por el momento HLMP no es compatible con Windows Phone 7. De todas maneras, la arquitectura fue planteada para ser altamente compatible con dicho sistema operativo, aprovechando el hecho de que Silverlight (el framework de desarrollo de WP7) es un subconjunto de WPF (el framework de desarrollo de Windows 7).

Cabe destacar que la versión actual de HLMP desconecta el dispositivo de internet (o de cualquier red inalámbrica en uso) para conectarse a la Manet. El desarrollo de la aplicación consideró esto como válido, y todos los mecanismos lo asumen (por ejemplo, no es posible importar contactos desde una SNS si la aplicación está conectada a la Manet). En este aspecto, las modificaciones a la aplicación Lukap quedan sujetas a las futuras versiones de HLMP.

El software se desarrolló de acuerdo a la arquitectura planteada, y resulta oportuno decir que se cumplieron tanto el objetivo general como los objetivos específicos del trabajo de memoria. Lamentablemente, no fue factible realizar pruebas con usuarios reales dentro de los plazos establecidos, sin embargo, el cumplimiento de los objetivos hace esperar que la aplicación resulte sencilla de usar sin que sus usuarios hayan recibido entrenamiento previo en ésta, y logre ser una herramienta útil para llevar a un plano físico las interacciones entre los usuarios de redes sociales.

Por último, durante el desarrollo del tema de memoria surgieron distintas ideas que finalmente no se implementaron en la versión final del software, por el tiempo requerido o por no ser críticas para el cumplimiento de los objetivos. A pesar de esto, el diseño de la arquitectura se realizó considerando estos temas, para permitir cambios o extensiones en el futuro. Los temas mencionados son:

- *Soporte para otras SNS*: la arquitectura actual del software implementa el protocolo OAuth 2.0 para autorizar el acceso a contenidos de las SNS, y en particular, implementa el acceso a Facebook mediante dicho protocolo. Con la arquitectura planteada se hace sencillo agregar soporte para otras SNS que usen el protocolo OAuth 2.0, como Foursquare, Live Messenger o LinkedIn. Queda pendiente implementar el soporte para estas SNS, y también implementar otros protocolos de autorización, como OpenId[28] u otras versiones de OAuth.
- *Agregar más opciones de programa*: el análisis de programas similares arrojó una larga lista de ideas de Opciones de Programa, las cuales no se implementaron pues de ninguna forma era crítico para cumplir los objetivos del tema de memoria. Sin embargo, proveer de más opciones de programa podrá permitir que el usuario personalice de mejor manera el comportamiento de la aplicación. Al enriquecer la interacción del usuario con el software será menos probable que dejen de usarlo debido a incomodidades o frustraciones provocadas por el mismo programa.
- *Enriquecer lista de contactos*: existen dos maneras principales de enriquecer la lista de contactos. Una consiste en agregar funcionalidades propias de programas de mensajería instantánea, tales como cambiar nombres de contactos, bloquear temporalmente a los contactos, chats grupales, etc. Muchas de estas caben en la categoría de "opciones de programa" ya comentadas. El otro modo se relaciona con interacción cara-a-cara entre contactos, y dichas funcionalidades deberán diseñarse e implementarse en base a las necesidades o requerimientos que los usuarios del software hagan a lo largo del tiempo (ejemplo: notificaciones personalizadas para cada usuario, calendario de reuniones, etc.)
- *Estados de usuario funcionales*: la versión actual de Lukap permite configurar la "visibilidad" o "estado" para cada grupo, pero esto es meramente informativo. En versiones

[28] OpenId Foundation website. URL: http://openid.net/. Última visita: Marzo, 2012.

futuras se espera que el usuario pueda configurar las notificaciones, o respuestas automáticas, para cada estado existente. Por ejemplo, si para un grupo la visibilidad es "ocupado", los mensajes de esos contactos no generarán notificaciones sonoras.

- *Nuevas versiones de HLMP*: la librería HLMP es fundamental para el software, pues provee los mecanismos de conexión a la red local, detección de usuarios y comunicación entre dispositivos. Para versiones futuras de la librería se espera:
  - Posibilidad de conectarse a la red local sin tener que desconectarse de Internet.
  - Compatibilidad de la librería con Windows Phone 7, lo que permitirá reusar gran parte del software al implementar Lukap para Silverlight.
  - Compatibilidad de HLMP con otros sistemas operativos. Con implementaciones apropiadas de Lukap para cada sistema operativo involucrado (que permitan comunicaciones entre ellas independiente del S.O. donde se ejecuten) aumentará el conjunto de contextos y usos posibles de la aplicación.

7. Bibliografía y Referencias

[1] Rodríguez-Covili, J.F. Ochoa, S.F., Pino, J.A., Messeguer, R., Medina, E., Royo, D. "A Communication Infrastructure to Ease the Development of Mobile Collaborative Applications". Journal of Network and Computer Applications. 34(6), pp. 1883-1893, 2011.

[2] Boyd, D. M. and Ellison, N. B. "Social Network Sites: Definition, History, and Scholarship". Journal of Computer-Mediated Communication, 13(1), pp. 210–230, 2008.

[3] The 2010 U.S. Digital Year in Review, February 7, 2011 http://www.comscore.com/Press_Events/Presentations_Whitepapers/2011/2010_US_Digital_Year_in_Review. Última Visita: Marzo, 2012.

[4] Twitter Help Center. Cómo obtener notificaciones, @menciones, MDs y más, vía SMS. URL: http://support.twitter.com/groups/34-mobile/topics/123-getting-started/articles/361700-c-xf3-mo-obtener-notificaciones-menciones-mds-y-m-xe1-s-v-xed-a-sms. Última Visita: Marzo, 2012.

[5] Facebook Mobile. URL: http://www.facebook.com/mobile/. Última Visita: Marzo, 2012.

[6] Facebook Text Messages. URL: http://www.facebook.com/mobile/?texts. Última Visita: Marzo, 2012.

[7] Foursquare. URL: http://foursquare.com/. Última Visita: Marzo, 2012.

[8] Rodríguez-Covili, J.F. Ochoa, S.F., Pino, J.A., Herskovic, V., Favela, J., Mejía, D., Morán, "Towards a reference architecture for the design of mobile shared Workspaces". Future Gener. Comput. Syst. 27 (1), pp. 109-118. Enero 2011.

[9] Rodríguez-Covili, J.F. Ochoa, S.F., Aliaga, R. "Extending Internet-Enabled Social Networks". Under Review in the 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2012).

[10] Microsoft .NET Framework 4 (instalador Web). URL: http://www.microsoft.com/downloads/es-es/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992. Última Visita: Marzo, 2012.

[11] Video Training – WindowsClient.net. URL: http://windowsclient.net/learn/video.aspx?v=315275. Última Visita: Marzo, 2012.

[12] Download the Windows Phone SDK (Windows Phone Developer Tools). URL: http://download.microsoft.com/download/1/7/7/177D6AF8-17FA-40E7-AB53-00B7CED31729/vm_web.exe. Última Visita: Marzo, 2012.

[13] Microsoft Download - Microsoft Windows SDK for Windows 7 and .NET Framework 4. URL: http://www.microsoft.com/downloads/dlx/en-us/listdetailsview.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b. Última Visita: Marzo, 2012.

[14] Graph API – Facebook Developers. URL: https://developers.facebook.com/docs/reference/api/. Última Visita: Marzo, 2012.

[15] Authentication – Facebook Developers. URL: http://developers.facebook.com/docs/authentication/. Última Visita: Marzo, 2012.

[16] Hammer-Lahav, et al. The OAuth 2.0 Authorization Protocol. URL: http://tools.ietf.org/pdf/draft-ietf-oauth-v2-12.pdf. Última Visita: Marzo, 2012.

[17] Windows Live Messenger – a short history. URL: http://windowsteamblog.com/windows_live/b/windowslive/archive/2010/02/09/windows-live-messenger-a-short-history.aspx. Última Visita: Marzo, 2012.

[18] Social Networking Accounts for 1 of Every 4 Minutes Spent Online in Argentina and Chile. URL: http://www.comscore.com/Press_Events/Press_Releases/2011/3/Social_Networking_Accounts_for_1_of_Every_4_Minutes_Spent_Online_in_Argentina_and_Chile. Última Visita: Marzo, 2012.

[19] Chapter 5: Implementing the MVVM Pattern. URL: http://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx. Última Visita: Marzo, 2012.

[20] Giles, L.C., Glonek, G.F., Luszcz, M.A., Andrews, G.R. “Effect of social networks on 10 year survival in very old Australians: the Australian longitudinal study of aging,” Journal of Epidemiology and Community Health 59(7), pp. 574-579, 2005.

[21] Joinson, A.N. “Looking up’ or ‘Keeping up with’ people? Motives and Uses of Facebook”. Proc. of the 26th Annual ACM SIGCHI’08. ACM Press, pp. 1027–1036. New York, USA, 2008.

[22] Bordia, P. “Face-to-Face Versus Computer-Mediated Communication: A Synthesis of the Experimental Literature.” Journal of Business Communication 34.1 (1997) : 99-118.