Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

APLICACIÓN MÓVIL DE MENSAJERÍA INSTANTÁNEA MULTIIDIOMA

Alumno: Manuel Beas Martínez

Tutor: Juan José Jiménez Delgado
Dpto: Informática

Septiembre, 2014

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Juan José Jiménez Delgado, tutor del Proyecto Fin de Carrera titulado: Aplicación móvil de mensajería instantánea multiidioma, que presenta Manuel Beas Martínez, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2014

El alumno:

Manuel Beas Martinez

El tutor:

NOMBRE JIMENEZ DELGADO JUAN JOSE - NIF 26216401N

Firmado digitalmente por NOMBRE JIMENEZ DELGADO JUAN JOSE - NIF 26216401N
Nombre de reconocimiento (DN): c=ES, o=FNMT, ou=FNMT Clase 2 CA, ou=500052716, cn=NOMBRE JIMENEZ DELGADO JUAN JOSE - NIF 26216401N
Fecha: 2014.09.02 10:11:47 +02'00'

Juan José Jiménez Delgado

AGRADECIMIENTOS

Me gustaría agradecer a todas las personas que me han apoyado durante transcurso de mis estudios todos estos años en la Universidad de Jaén.

Principalmente a mis padres, por darme esta oportunidad y no perder la paciencia conmigo en ningún momento.

A mis compañeros de trabajo de El Koala, por acompañarme y apoyarme durante todo este tiempo.

A Natalia, por apoyarme tanto en la realización de este proyecto, como en la adaptación al país donde ha dado comienzo mi vida laboral como Ingeniero Informático. Gracias por estar siempre ahí para echarme una mano.

Índice

ÍNDICE DE FIGURAS

CAPÍTULO 1

PREFACIO

1. Prefacio

1.1. Motivación

Vivimos en un mundo en el que cada día la mensajería instantánea cobra más importancia.

Según un reciente estudio de la AIMC, el uso de la mensajería instantánea se ha disparado durante los últimos años, llegando así en 2014 a superar al uso del email tradicional.

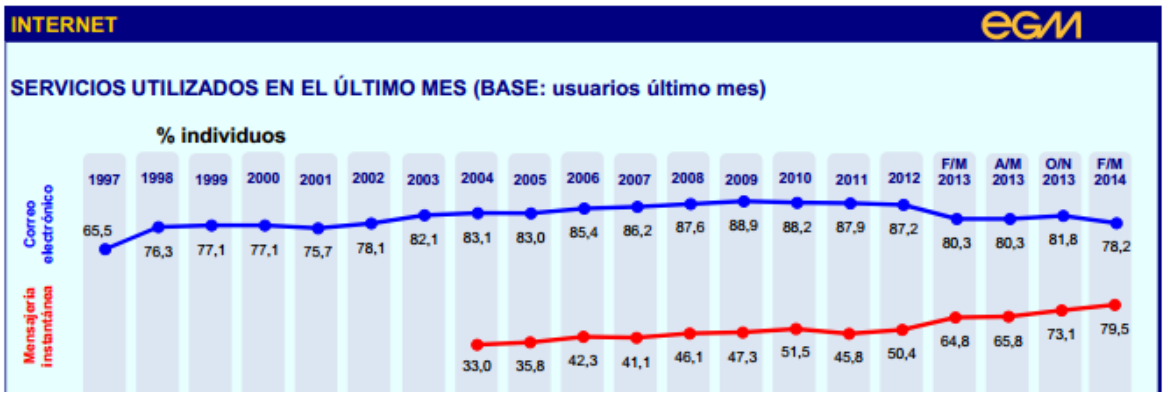


Figura 1.1 Evolución del uso del correo electrónico y de la mensajería instantánea [1]

La mayoría de esos usuarios hacen uso de aplicaciones móviles dirigidas a este fin, con aplicaciones que llegan hasta los 500 millones de usuarios activos.

Durante mucho tiempo los clientes de este tipo de mensajería para PC han ido evolucionando, de forma que facilitan el uso para la comunicación entre individuos de distintos países mediante la traducción automática de mensajes. Muchas son las extensiones que nos permiten activar una rápida traducción integrada en el chat para una comunicación multicultural.

La falta que tienen en común las diferentes alternativas de mensajería instantánea de dispositivos móviles disponibles del momento es la misma: ninguna ofrece un mecanismo de traducción instantánea. Lo que hace tediosa la comunicación con gente en otro idioma debido al proceso de traducción manual que uno debe llevar a cabo tras cada mensaje.

1.2. Objetivos

Diseñar una aplicación de mensajería instantánea en la cual se facilite el proceso de comunicación entre usuarios que hablen diferentes idiomas automatizando la tarea de traducción y soportando diferentes idiomas en la interfaz de usuario.

1.3. Estructura de la memoria

La estructura de esta memoria está dividida en 8 capítulos. A lo largo de estos 8 capítulos recorreremos el proceso de realización de un producto de software real, desde la concepción de la idea hasta las pruebas de control de calidad finales.

A continuación se muestra una breve descripción de que veremos en cada capítulo:

- **Capítulo 1: Prefacio**

Este primer capítulo servirá de introducción para determinar qué queremos hacer y por qué lo queremos hacer, definir los objetivos y el plan de realización será la base sobre la que vamos a trabajar durante el resto del proyecto.

- **Capítulo 2: Entorno actual**

Un tema relevante a la realización de un proyecto es el análisis de antecedentes, la situación actual del mercado y la comparación de lo que queremos hacer con las alternativas actualmente existentes. Durante este capítulo haremos un repaso de la importancia de la mensajería instantánea móvil hoy en día y de la falta de mecanismos de traducción en las aplicaciones más usadas.

- **Capítulo 3: Planificación del proyecto**

Una vez decidida y justificada la aplicación que queremos realizar hablaremos sobre la planificación a lo largo de este cuatrimestre para realizar el proyecto, la planificación de los recursos que necesitaremos y una estimación del presupuesto que se necesitaría para completarlo.

- **Capítulo 4: Tecnologías**

La decisión de las tecnologías a utilizar en un proyecto de software es crucial para determinar las posibilidades que vamos a tener a la hora de diseñar e implementar este software. En este capítulo estudiaremos las alternativas tecnológicas que existen hoy en día para poder llevar a cabo un proyecto de este tipo y elegiremos las que más se ajusten a nuestras necesidades.

- **Capítulo 5: Análisis**

Una de las tareas que más tiempo nos tomará será la de análisis y búsqueda de información.

Antes de actuar es necesario entender la arquitectura necesaria para este proyecto, los requisitos que debemos cumplir y la realización de un estudio de viabilidad.

- **Capítulo 6: Diseño**

En este capítulo hablaremos de los diferentes diseños que serán necesarios para la implementación del proyecto. Diseños para el ciclo de funcionamiento de la aplicación, organización del código en clases, diseño de datos etc. Estos diseños regirán el apartado de diseño.

- **Capítulo 7: Desarrollo**

Se trata de un capítulo más técnico que los demás, con detalles sobre la implementación de los tres proyectos resultantes. Aquí hablaremos de las características más relevantes a la hora de la implementación

- **Capítulo 8: Conclusiones y trabajos futuros**

Último capítulo de la memoria en el cual encontraremos una breve conclusión sobre el desarrollo de este proyecto y las distintas posibilidades de extensión del mismo en un futuro.

CAPÍTULO 2

ENTORNO ACTUAL

2. Entorno actual

2.1. Los Sistemas Operativos móviles de hoy en día

Durante el paso de los últimos años el crecimiento del uso de los Smartphones ha crecido de forma imparable, en concreto los sistemas operativos predominantes alrededor del año 2000 eran principalmente BlackBerry OS y Symbian.

No obstante no fue hasta el año 2007 cuando empezó la verdadera revolución de los Smartphones con el lanzamiento del primer IPhone de Apple.

Más tarde, en 2008, se lanzaría el primer dispositivo con Android 1.0 que, aunque no fue tan bien recibido como el S.O. de Apple, comenzó una revolución en cuanto a sistemas operativos libres para móviles.

Desde entonces existe una interminable batalla entre estos dos S.O. móviles que se reparten la mayoría de la cuota del mercado.

La elección de una plataforma donde desarrollar la aplicación es una fase crucial para comenzar con el diseño del proyecto. Al tratarse de una aplicación de mensajería instantánea, me atrevería a decir que uno de los factores más importantes es el número de usuarios potenciales que podrían hacer uso de la aplicación.

Spain	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	2.9	6.1	3.1
Android	82.5	89.9	7.3
BlackBerry	9.4	0.7	-8.7
Symbian	2.1	0.7	-1.5
Windows	1.7	1.8	0.1
Other	1.4	0.9	-0.5

Figura 2.1 Tabla de reparto de mercado de S.O. móviles en España [2]

Como podemos observar en la figura 2.1, la penetración de Android en el mercado Español en 2013 está cerca del 90% del total de Smartphones frente al 6%

de IOS. Este es un factor muy decisivo dada la predominancia total del S.O. de Google en nuestro país, no obstante, el objetivo de la aplicación es la traducción entre distintos idiomas, por lo tanto debemos cerciorarnos de que este no es un caso aislado y hacer hincapié en un S.O. muy presente en los demás países.

Smartphone OS Sales Share (%)

Germany	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	13.3	11.2	-2.1
Android	73.3	76.8	3.4
BlackBerry	0.6	0.8	0.2
Symbian	5.0	1.4	-3.6
Windows	6.2	8.8	2.6
Other	1.5	1.0	-0.5

GB	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	23.3	31.1	7.8
Android	59.1	55.2	-3.8
BlackBerry	11.0	3.5	-7.5
Symbian	1.7	0.5	-1.2
Windows	4.2	9.2	5.0
Other	0.7	0.4	-0.3

France	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	12.7	17.3	4.6
Android	62.1	62.5	0.4
BlackBerry	9.2	3.7	-5.5
Symbian	2.1	1.5	-0.6
Windows	3.6	11.0	7.4
Other	10.3	4.1	-6.2

Italy	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	14.8	16.2	1.5
Android	58.2	71.0	12.8
BlackBerry	4.1	2.8	-1.2
Symbian	12.4	1.8	-10.5
Windows	8.3	7.8	-0.5
Other	2.3	0.3	-2.0

Spain	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	2.9	6.1	3.1
Android	82.5	89.9	7.3
BlackBerry	9.4	0.7	-8.7
Symbian	2.1	0.7	-1.5
Windows	1.7	1.8	0.1
Other	1.4	0.9	-0.5

USA	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	35.6	43.4	7.8
Android	58.7	51.1	-7.6
BlackBerry	1.9	1.2	-0.6
Symbian	0.0	0.0	0.0
Windows	3.0	3.5	0.5
Other	0.9	0.8	-0.1

China	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	26.3	22.4	-3.9
Android	61.7	70.5	8.8
BlackBerry	0.1	0.1	0.0
Symbian	5.2	2.9	-2.2
Windows	4.6	2.4	-2.2
Other	2.2	1.6	-0.5

Australia	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	28.0	28.1	0.1
Android	62.9	62.0	-0.9
BlackBerry	1.4	0.3	-1.1
Symbian	1.8	1.1	-0.7
Windows	4.6	7.0	2.4
Other	1.3	1.4	0.1

Mexico	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	3.3	9.2	5.8
Android	28.3	60.0	31.7
BlackBerry	34.9	10.0	-25.0
Symbian	24.7	7.3	-17.5
Windows	2.0	12.5	10.5
Other	6.7	1.1	-5.6

EU5	3 m/e July 2012	3 m/e July 2013	% pt. Change
iOS	14.8	17.9	3.1
Android	66.2	69.1	2.9
BlackBerry	6.7	2.4	-4.3
Symbian	4.3	1.1	-3.1
Windows	4.9	8.2	3.3
Other	3.2	1.3	-1.9

Figura 2.2 Tabla de reparto de mercado de S.O. móviles en diversos países [2]

Como podemos apreciar en la figura 2.2, el uso de Android predomina en muchos de los países desarrollados.

Dado el bajo coste de los dispositivos Android podemos dar por hecho que su uso en países con menor poder económico será también mucho más alto que la presencia del S.O. de Apple.

En mi opinión este es el principal motivo para la elección de Android como plataforma donde desarrollar este proyecto, no hay nada más importante en una aplicación de mensajería instantánea que el número de usuarios potenciales.

Una vez decidida la plataforma a la que nos vamos a dirigir es importante determinar a qué versiones de Android vamos a dar soporte, tema que veremos en detalle en el siguiente apartado.

2.2. Fragmentación en Android

Hablar sobre la fragmentación de Android es hablar sobre el pasado, presente y futuro de este sistema operativo.

Desde su primera versión 1.0 en 2008, Android ha sido soportado una innumerable cantidad de dispositivos y, por lo tanto, de diferentes conjuntos de hardware.

Con la evolución del S.O. se han incluido muchas mejoras y funcionalidades, pero todas estas mejoras no podrían llevarse a cabo sin un desarrollo acorde del hardware en el que se ejecutan. Cada fabricante es responsable de la adaptación del sistema operativo al hardware de su dispositivo, y en muchas ocasiones este soporte es bastante breve. De manera que poco tiempo el dispositivo se quedará estancado en una versión antigua sin posibilidad de ser actualizado y de poder disfrutar de las nuevas funcionalidades de Android.

También podemos hablar de fragmentación en el sentido de tamaños de pantalla. En una aplicación de Android habrá que tener en cuenta diferentes tamaños de recursos e incluso de disposición de menús para una mayor eficiencia y una mejor experiencia de usuario.

Estos hechos complican la tarea de desarrollar aplicaciones para esta plataforma, ya que si se utilizan funcionalidades no disponibles en versiones previas, la aplicación no funcionará correctamente en esas versiones.

Debemos encontrar un punto de equilibrio entre la versión mínima que vamos a soportar y la adicción de complejidad que supone soportar versiones muy antiguas en la hora de la implementación.

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	11.4%
4.1.x	Jelly Bean	16	27.8%
4.2.x		17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%

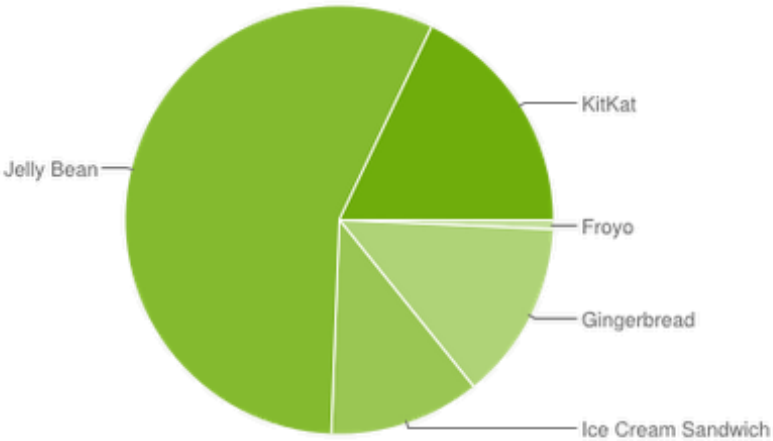


Figura 2.3 Fragmentación en Android. Julio de 2014. [3]

Desde la página web de Android Developers nos facilitan estadísticas de uso de las diferentes versiones cada pocos meses. La figura 2.3 nos muestra una captura de ésta página web, con información de la fragmentación de Android a fecha de Julio de 2014. De esta manera podemos ver con relativa actualidad el estado del sistema operativo.

Sumando porcentajes, apreciamos que un 85,8% de los dispositivos utilizan una versión igual o superior a la 4.0.3 y pasado un tiempo se completará la transición en casi su totalidad.

Desde el punto de vista de desarrollo, la versión 4.0.3 utiliza la API 15. Ofreciendo soporte a un mínimo de la API 15 podríamos soportar todas las funciones necesarias para esta aplicación de forma nativa sin necesidad de hacer uso de librerías de soporte.

Por esta razón y por la facilidad a la hora de implementación, la API 15 será la mínima que vamos a soportar.

2.3. La mensajería instantánea

Como pudimos apreciar en la figura 1.1, un reciente estudio de la Asociación para la Investigación de Medios de Comunicación nos indicaba que durante el año 2014 el uso de la mensajería instantánea ha superado por primera vez al uso del correo electrónico según el porcentaje de individuos en España.

En concreto, en España la aplicación de mensajería instantánea más utilizada es WhatsApp, la cual domina el mercado mundial y tiene un total de 500 millones de usuarios activos (poner algo del link).

WhatsApp es la aplicación de mensajería instantánea con más usuarios activos del momento a nivel mundial, pero no es la predominante en todos los países.

En la siguiente tabla, mostramos una lista de las aplicaciones móviles de mensajería instantánea más usadas actualmente, el país donde más se usa cada una y una aproximación al número de usuarios activos de los que disponen.

Nombre	Logo	País donde más se usa	Usuarios activos
WhatsApp		Estados Unidos	500 millones
Viber		Chipre	200 millones
WeChat		China	396 millones
Line		Japón	400 millones
KakaoTalk		Corea Del Sur	140 millones
Kik Messenger		Canadá	140 millones
Tango		Estados Unidos	70 millones
Nimbuzz		India	150 millones

Figura 2.4 Aplicaciones de mensajería instantánea y usuarios activos.

Viendo los datos de usuarios activos, está claro que el mercado para aplicaciones de mensajería instantánea es inmenso, su uso está en auge y aun en crecimiento, y existe una fragmentación importante según el uso por países.

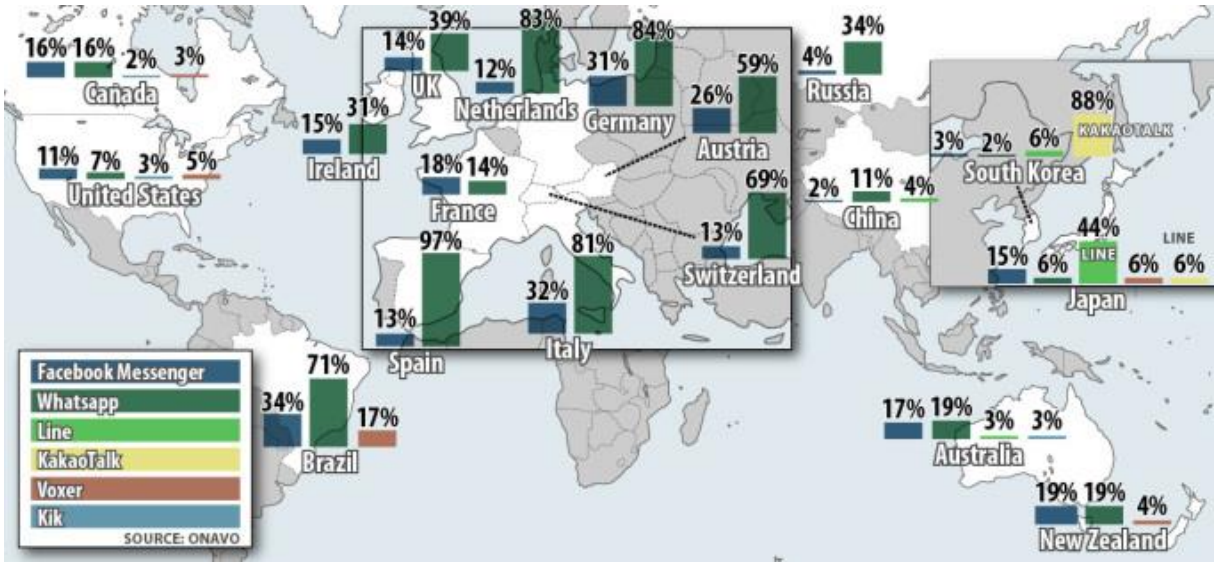


Figura 2.5 Porcentaje de usuarios de Iphone que utilizan mensualmente las aplicaciones de mensajería instantánea por países [4]

Debemos considerar también que los usuarios de estas aplicaciones no son usuarios exclusivos de una aplicación. En la mayoría de casos, un usuario es consumidor de más de una aplicación de mensajería instantánea. ¿Cuáles son los motivos que llevan al usuario a mantener más de una aplicación? ¿Por qué el éxito de unas aplicaciones sobre otras?

Los factores clave que diferencian estas aplicaciones entre ellas y llevan al usuario a mantener varias de ellas son:

- **Usuarios**. Por norma general, los usuarios de una aplicación no serán capaces de comunicarse con los de otra aplicación. Podemos decir que este factor conlleva a un comportamiento como el de una bola de nieve, a medida que avanza se va haciendo más grande, ya que el número de usuarios que se decantarán por esta aplicación dependerá en gran medida de los contactos de los que puedan disponer en la misma
- **Funcionalidades extra**. El intercambio de mensajes simples de texto se ha convertido en algo tan básico que las principales aplicaciones de mensajería han ido evolucionando hasta el punto de ofrecer muchísimas más funcionalidades como envío de archivos, multimedia, audio rápido, localización, compartir estados, etc.

2.4. La traducción

La globalización es un factor que ha contribuido enormemente a la importancia de la traducción desde el siglo XX.

Empresas de todo el mundo se amplían a otros países donde se hablan idiomas diferentes al de su país de origen. Programas de intercambio de estudiantes, el turismo en países exóticos y otros factores conllevan la creación de relaciones y amistades entre las personas de países distintos.

En todos estos casos se requiere un medio de comunicación, el cual no siempre favorece el entendimiento entre diferentes idiomas.

Una aplicación de mensajería instantánea es la herramienta ideal para muchas de estas personas, ya que es de los medios de comunicación a distancia más usados de hoy en día.

2.5. Alternativas

Como hemos visto en la figura 2.4, existen múltiples alternativas de mensajería instantánea en el mercado actual. Dado el gran número de alternativas existentes, nos centraremos en analizar tres de las más importantes para Android: WhatsApp, Line y Wechat.

2.5.1. WhatsApp Messenger

Whatsapp es una aplicación de mensajería instantánea disponible para numerosas plataformas móviles tales como Android, iOS, Windows Phone, BlackBerry OS, Symbian ,etc.

Figura 2.6 Logo de WhatsApp

Fundada en 2009, en tan solo 5 años ha conseguido ser la aplicación de referencia en cuanto a mensajería instantánea, con más de 500 millones de usuarios activos y un valor de más de 19.000 millones de dólares.

Al igual que las demás aplicaciones de mensajería instantánea, con el paso del tiempo ha ido evolucionando hasta el punto de ofrecer muchas funcionalidades añadidas al envío de mensajes

Funcionalidades

- Compartir imágenes
- Compartir videos
- Compartir localización
- Envió de notas de audio
- Compartir estados
- Envió de emoticonos
- Compartir contactos
- Grupos de conversación
- Listas de difusión
- Múltiples opciones de configuración
    - Fondos de pantalla
    - Notificaciones
    - Tamaños de fuente
    - Etc.

El hecho de aparecer en un buen momento cuando aún no había gran competencia le ha permitido expandirse a lo largo del planeta dominando en la mayoría de países desarrollados.

La facilidad de uso es uno de los puntos más importantes de esta aplicación, de manera que permita su uso a personas que no están acostumbradas a las nuevas tecnologías y dispositivos móviles. Es interesante remarcar que su diseño prácticamente no ha cambiado desde su aparición hace 5 años, manteniendo así un diseño intuitivo desde el comienzo y simplificando el uso a los usuarios.

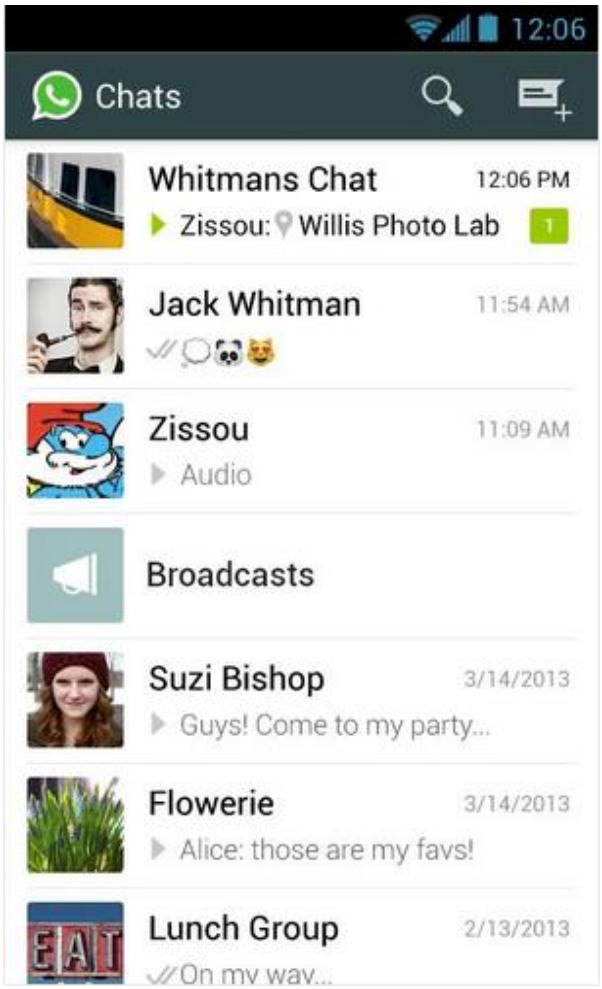


Figura 2.7 Interfaz de WhatsApp

2.5.2. Line

Figura 2.8 Logo de Line

Uno de los competidores de WhatsApp. El servicio Line, utilizado por 70 millones de usuarios desde que se lanzó en su país de origen, Japón, permite no solo enviar mensajes de texto, voz y vídeo sino también realizar llamadas gratuitas.

Está disponible para iOS, Android, Windows Phone, Blackberry, Windows y Mac. Para utilizarlo en un ordenador es necesario registrarse con una dirección de correo electrónico.

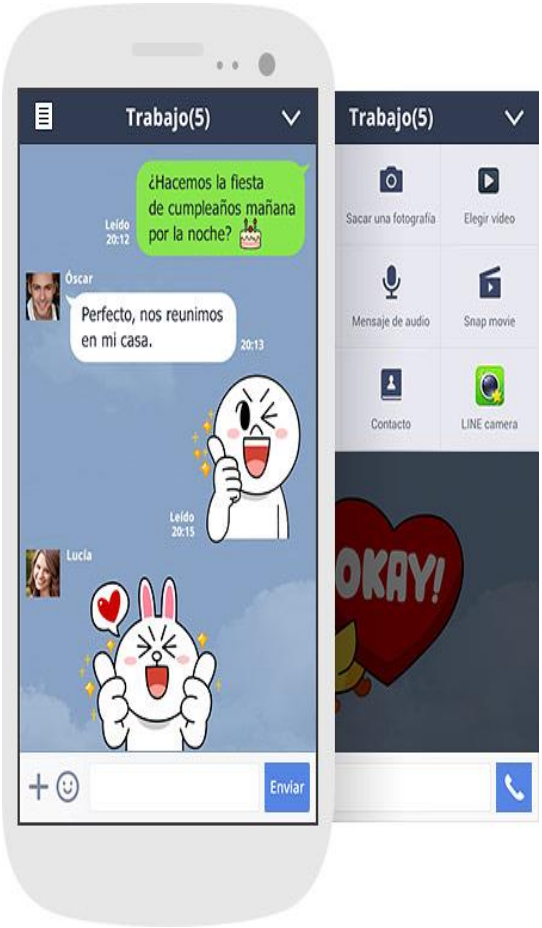
Funciones:

- Videollamadas.
- Llamadas de voz y videollamadas gratis.
- Mensajes enviados y recibidos de forma instantánea
- Timeline - para publicar fotos, texto, películas, stickers e información de la ubicación.
- Stickers con personajes llenos de humor y diversión.

Line está iniciando su lanzamiento en Europa y en Estados Unidos y pretende cosechar el mismo éxito que ha conseguido en Japón.

Es una herramienta de comunicación muy interesante cuyo principal problema es la enorme competencia que tiene con otros programas similares, ya que pese a ofrecer buenas prestaciones, su uso no ha cobrado tanta fuerza en el mercado Europeo y Americano.

La interfaz que posee es algo más compleja que las demás aplicaciones, sobrecargando en ocasiones las pantallas con muchas opciones y utlizando colores y animaciones muy llamativas.



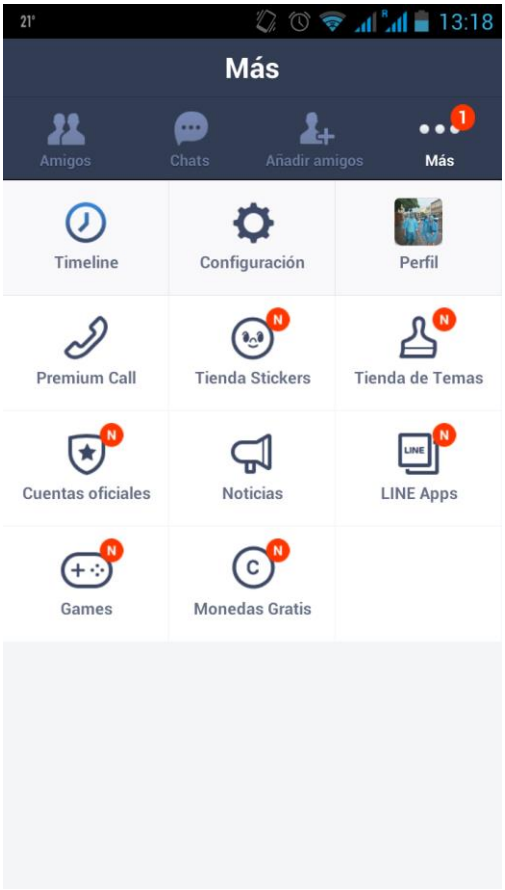


Figura 2.9 Interfaz de Line 1

Figura 2.10 Interfaz de Line 2

2.5.3. WeChat

WeChat es una completa aplicación de comunicación móvil y redes sociales privadas con más de 300 millones de usuarios. Se lanzó por primera vez en enero del 2011 en China.

Figura 2.11 Logo de WeChat

La aplicación está disponible en Android, iPhone, BlackBerry, Windows Phone y las plataformas Symbian s40 y s60.

Es gratuita, multiplataforma y está llena de funciones que la convierten en la mejor manera de mantenerse en contacto con todos.

Funciones:

- Mensajería individual y grupal gratuita entre plataformas con texto, notas de voz, imágenes, videos, datos de ubicación y más.
- Video llamadas HD gratuitas.
- Historial seguro de mensajes almacenados localmente, siempre accesibles sin conexión.
- No tiene la publicidad.
- Red social privada para compartir fotos en grupos.
- Agregar amigos agitando los teléfonos juntos, escanear códigos QR, sincronizar los contactos del teléfono.
- Registro rápido a partir del número de teléfono.
- Uso de WeChat desde el ordenador a través de la página web de WeChat.
- Admite más de 18 idiomas.
- Dispone de una API que permite el uso de aplicaciones de terceros, para compartir contenido por chats o compartir en Momentos.

WeChat cuenta, según cifras oficiales, con más de 300 millones de usuarios, situándose alrededor de 70 de ellos fuera de China. Es una cifra bastante moderada para los usuarios internacionales, centrando su núcleo de usuarios en el continente Asiático.

Su interfaz es algo más sobria que la de Line, utilizando también llamativas animaciones pero abusando menos de los colores. En cuanto a la usabilidad, es relativamente simple de utilizar y añade más funciones en primer plano que WhatsApp.

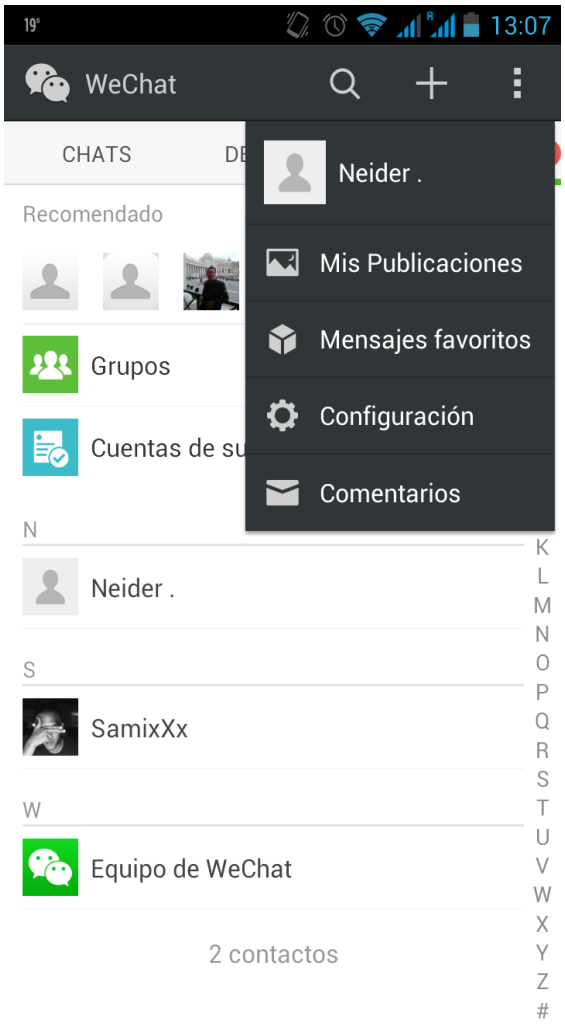


Figura 2.12 Interfaz de WeChat

2.6. Conclusiones

El mercado de aplicaciones de mensajería instantánea se encuentra fragmentado y aun en expansión, aunque la expansión de las mismas depende en gran medida del número de usuarios que ya tienen las aplicaciones actuales.

Las aplicaciones de hoy en día no se limitan a la transmisión de mensajes, sino que ofrecen un gran abanico de posibilidades extra para los usuarios, sobre todo en cuanto a compartición de información y multimedia.

A pesar de la gran cantidad de funcionalidades adicionales, aún podemos encontrar la falta de mecanismos de traducción en la totalidad de aplicaciones disponibles. Es una característica que aún está por explotar.

Es por esto que tenemos la posibilidad de realizar un proyecto relativamente innovador ofreciendo algo que ninguna de las alternativas actuales es capaz de ofrecer.

CAPÍTULO 3

PLANIFICACIÓN DEL PROYECTO

3. Planificación del proyecto

A la hora de comenzar un proyecto es crucial la planificación previa de las actividades a realizar, su definición y asignación de espacio temporal, de manera que marquemos un objetivo a cumplir durante los intervalos definidos.

3.1. Tareas

- **Análisis**

  - Búsqueda de información

Adquisición de conocimiento en relación a la mensajería instantánea, su funcionamiento, importancia de hoy en día y alternativas.

  - Análisis del sistema

Aplicación de la información recopilada en la tarea anterior a las necesidades de nuestro proyecto. Proyección de las posibilidades que existen para realizar la aplicación como un proyecto individual.

- **Diseño**

  - Diseño de la arquitectura

Identificación de elementos necesarios para el funcionamiento de la aplicación, organización en sub-proyectos.

  - Diseño de la aplicación

Diseño del flujo de la aplicación móvil, diseño de la interfaz, organización y relaciones entre clases, diseño de la estructura de las datos.

- **Implementación**

Aplicación del diseño a la producción del código en las plataformas elegidas.

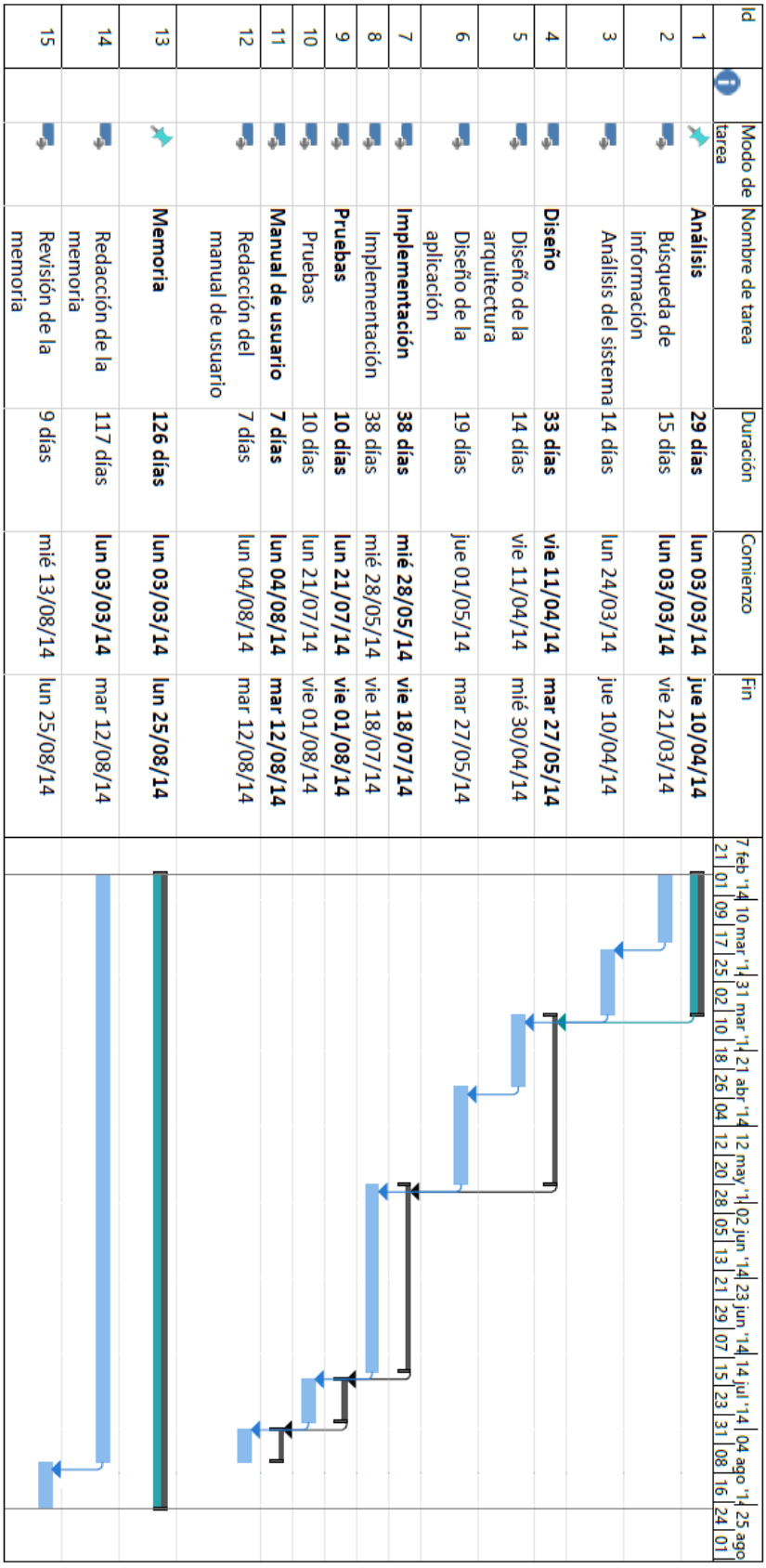
- **Pruebas**

Testeo unitario de las diferentes funciones del sistema. Pruebas funcionales sobre el ciclo de funcionamiento de toda la aplicación móvil.

- **Manual de usuario**

Redacción de un breve manual de usuario con las instrucciones para el uso de la aplicación para el usuario final.

3.2. Planificación temporal



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Análisis**	**29 días**	**lun 03/03/14**	**jue 10/04/14**
2		Búsqueda de información	15 días	**lun 03/03/14**	vie 21/03/14
3		Análisis del sistema	14 días	lun 24/03/14	jue 10/04/14
4		**Diseño**	**33 días**	**vie 11/04/14**	**mar 27/05/14**
5		Diseño de la arquitectura	14 días	vie 11/04/14	mié 30/04/14
6		Diseño de la aplicación	19 días	jue 01/05/14	mar 27/05/14
7		**Implementación**	**38 días**	**mié 28/05/14**	**vie 18/07/14**
8		Implementación	38 días	mié 28/05/14	vie 18/07/14
9		**Pruebas**	**10 días**	**lun 21/07/14**	**vie 01/08/14**
10		Pruebas	10 días	lun 21/07/14	vie 01/08/14
11		**Manual de usuario**	**7 días**	**lun 04/08/14**	**mar 12/08/14**
12		Redacción del manual de usuario	7 días	lun 04/08/14	mar 12/08/14
13		**Memoria**	**126 días**	**lun 03/03/14**	**lun 25/08/14**
14		Redacción de la memoria	117 días	**lun 03/03/14**	mar 12/08/14
15		Revisión de la memoria	9 días	mié 13/08/14	lun 25/08/14

Figura 3.1 Planificación temporal de las tareas a realizar

3.3. Presupuesto

La estimación del presupuesto de un proyecto recae, principalmente, sobre tres factores, el personal necesario, la asignación de las tareas definidas anteriormente a este y gastos materiales.

Para este proyecto en concreto serán necesarios los siguientes perfiles:

Jefe de proyecto: Encargado de supervisar al resto del equipo y garantizar el cumplimiento de los objetivos definidos. Esta persona recopilará toda la información del resto de personal y la plasmará en la memoria.

Analista: Su papel es el de identificar los requisitos impuestos por los objetivos del proyecto y definirlos claramente desde un punto de vista técnico para su posterior diseño e implementación.

Diseñador: Encargado de realizar tanto el diseño de alto nivel de la aplicación como la arquitectura

Programador: La persona que implementará el proyecto bajo el diseño establecido.

Tester: Es la persona encargada de asegurar la calidad de la solución. Debe comprobar una por una las funcionalidades implementadas por el programador y dar el visto bueno antes de la entrega final al cliente.

Además de los recursos humanos nos serán necesarios ciertos recursos materiales, tales como un ordenador (portátil) y al menos dos dispositivos móviles (Android).

Ordenador: Realizaremos el proyecto sobre un equipo portátil de clase media. En concreto las características principales del portátil serán:

- Procesador Intel Core i5.
- 4 GB de Memoria RAM.

- 512 GB de almacenamiento.

Un portátil simple de estas características se puede encontrar actualmente por una cifra alrededor de los 500€.

Si consideramos un tiempo de vida útil de unos 4 años para el equipo y tomamos en cuenta que el proyecto se desarrollará en 6 meses, podemos calcular que el presupuesto necesario para el equipo durante este proyecto será un total de 62,5€

Dispositivos móviles: Serán necesarios al menos dos dispositivos móviles para realizar las pruebas sobre ellos en diferentes situaciones reales de baja conectividad y ancho de banda. En concreto utilizaremos dispositivos móviles de clase media con un presupuesto de 170€ por dispositivo y unas especificaciones aproximadas a:

- S.O. Android 4.0.4
- Procesador de doble núcleo.
- 1GB de RAM
- 4,5” de pantalla
- 720x1280 píxeles.

Estos dispositivos móviles cuentan con, aproximadamente, dos años de vida útil, por lo que el coste un dispositivo amortizado a 2 años sería de 42,5€

Finalmente será necesario incluir en el presupuesto un lugar de trabajo y los costes derivados.

Oficina: Se alquilará una pequeña oficina de una habitación para el desarrollo de este proyecto. Los gastos derivados como luz, agua, internet y climatización están incluidos en el presupuesto de unos 200€.

Recurso	€/hora	Unidades	Tarea	Horas de trabajo	Coste total
Jefe de proyecto	40	-	Memoria	30	1200 €
Analista	30	-	Análisis	87	2610 €
Diseñador	25	-	Diseño	99	2475 €
Programador	18	-	Implementación	228	4104 €
Tester	15	-	Pruebas	50	750 €
Ordenador Portátil		1			62,5 €
Teléfono móvil Android		2			85 €
Alquiler de oficina		6			1200 €
Total				494	12.486,5 €

Figura 3.2 Tabla de presupuesto

CAPÍTULO 4

TECNOLOGÍAS Y ESTUDIO DE SU VIABILIDAD

4. Tecnologías y estudio de su viabilidad

4.1. Android

Android es un sistema operativo móvil basado en el núcleo de Linux long-term support (LTS) de naturaleza open source [7].

Desde el año 2005 cuando Android Inc. fue adquirido por Google, la compañía de Mountain View ha sido la encargada de desarrollar las futuras versiones de la plataforma. El proceso de desarrollo de Google se realiza a puerta cerrada, liberando después cada 6-9 meses las versiones con mejoras más significativas.

El sistema operativo en su nivel más alto está desarrollado en Java, lenguaje que debemos utilizar los desarrolladores de aplicaciones móviles para que funcionen de forma nativa.

No obstante ejecución de código en Android se produce en diferentes máquinas virtuales Dalvik. Una vez compilada la aplicación en Java, se produce la transformación del código en bytes de Java al formato Dalvik Executable (.dex).

Por tanto, los componentes principales de la arquitectura del sistema operativo son:

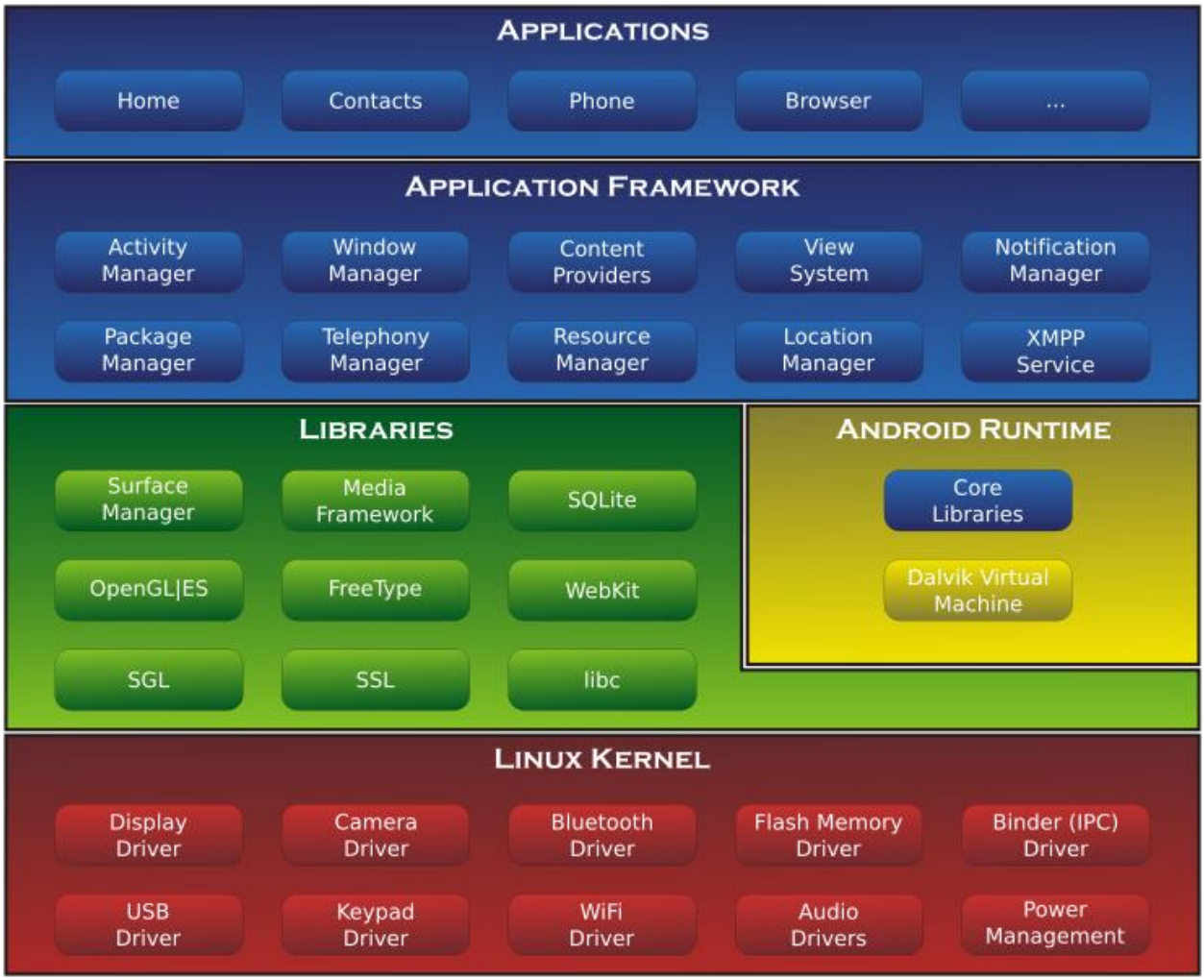


Figura 4.1 Arquitectura de Android

- **Aplicaciones**

Todos los tipos de aplicaciones, tanto las aplicaciones base del sistema (SMS, calendario, contactos) como aplicaciones de terceros tal y como la que vamos a desarrollar.

- **Marco de trabajo de aplicaciones.**

La arquitectura de Android está diseñada para simplificar el acceso y la reutilización de componentes. Cualquier aplicación puede ofrecer sus servicios al dispositivo de forma que otras aplicaciones puedan hacer uso de ellos. Así pues, los desarrolladores disponemos de acceso completo a las mismas Apis que utilizan las aplicaciones base del sistema.

- **Bibliotecas**

Son varias las bibliotecas de C/C++ que están implementadas en Android. Estas bibliotecas ofrecen características como bibliotecas de gráficos, 3D, SQLite, entre otras, que pueden ser utilizadas por los desarrolladores.

- **Runtime de Android**

Cada aplicación Android corre su propio proceso en su propia instancia de la máquina virtual Dalvik. Ésta máquina virtual ofrece la posibilidad de correr múltiples instancias de forma eficiente.

La máquina virtual Dalvik ejecuta ficheros Dalvik Executable, los cuales están optimizados para utilizar la mínima cantidad de memoria necesaria.

- **Núcleo de Linux**

Android depende de Linux para los servicios base de más bajo nivel como el sistema de seguridad, gestión de memoria, gestión de procesos, pila de red etc. El núcleo también actúa como una capa de abstracción entre el hardware y el resto del software.

4.1.1. Entorno de desarrollo integrado

Actualmente el entorno de desarrollo integrado más utilizado para programación en Android es Eclipse.

Google nos provee de la extensión para Eclipse “Android SDK” [6], la cual nos ofrece una serie de librerías API y herramientas de desarrollo para construir, testear y depurar aplicaciones para Android.

Una vez instalado el paquete Android SDK, tan solo es necesario seleccionar en el Android SDK Manager los componentes que queremos utilizar, como por ejemplo las API para las diferentes versiones de Android sobre las que vamos a dirigir nuestra aplicación.

Eclipse no es el único entorno de desarrollo integrado que soporta la programación en Android, IntelliJ IDEA es uno de los IDEs que ha cobrado más fuerza entre los desarrolladores en los últimos años, IDE sobre el que se está basando el desarrollo del nuevo IDE oficial de Google “Android Studio”, que aún se encuentra en fase beta, y por lo tanto no está recomendado para entornos de producción.

4.2. Protocolos de mensajería instantánea

4.2.1. XMPP (Extensible Messaging and Presence Protocol)

El protocolo XMPP [10], traducido al español como Protocolo extensible de mensajería y comunicación de presencia, es un protocolo de comunicación orientado a mensajes basado en XML. Originalmente se conoció como Jabber, ya que fue la compañía que comenzó su desarrollo y el primer cliente de mensajería instantánea en hacer uso de él.

A diferencia de muchos otros, XMPP se define como un protocolo estándar abierto, extensible y bien documentado para su libre implementación. Es por esto que existen múltiples servidores y clientes que utilizan XMPP con diferentes licencias (Apache, Commercial, GPL2, etc.).

Además de intercambio de mensaje de texto, mediante la extensión Jingle se permiten interacciones multimedia como llamadas VoIP o videoconferencia. Esta extensión fue desarrollada por Google para integrarla en su cliente de mensajería instantánea Google Talk, aunque actualmente es soportada por muchos otros clientes.

Podemos considerar a XMPP como el protocolo de mensajería instantánea por excelencia, muchas de las grandes aplicaciones de mensajería hacen uso de él, por ejemplo Google Talk e iChat. Por otro lado WhatsApp utiliza una modificación propia de XMPP que restringe el uso de sus servidores a sus clientes y posee numerosas extensiones. Por último, Facebook Messenger hace uso de XMPP a nivel de interfaz, esto es, aunque implementa MQTT para la comunicación proporciona una interfaz XMPP para que los desarrolladores implementen terceras aplicaciones de forma más sencilla.

A continuación expondremos algunas de sus ventajas y desventajas a destacar [8]:

Ventajas

- Descentralización

Al igual que sucede en la arquitectura de las redes de correo electrónico, XAMPP permite una arquitectura descentralizada, posibilitando a cualquiera establecer su propio servidor sin necesidad de ningún servidor central.

- Estándar abierto

El Grupo de Trabajo de Ingeniería de Internet (IETF) que regula las propuestas y estándares de Internet ha aprobado el uso de XMPP. Esta tecnología no está ligada a ninguna empresa.

- Historia y documentación

El protocolo existe desde 1998 y desde entonces ha sido adoptado por grandes empresas. Existe una extensa documentación por Internet para su uso y desarrollo.

- Seguridad

Al no depender de servidores centralizados, la red donde corre XMPP puede estar aislada de la red pública, además es posible hacer uso de sistemas de seguridad como SASL y TLS.

- Flexibilidad

Es un sistema extensible, por lo que se pueden hacer extensiones a medida y añadir las funcionalidades que sean convenientes. La XMPP Software Foundation gestiona las más comunes para mantener la interoperabilidad de sistemas.

Desventajas

- Sobrecarga de datos de presencia

XMPP fue diseñado para su uso en ordenadores personales con conexiones ilimitadas a Internet. Por esta razón el protocolo no está optimizado para dispositivos móviles, teniendo un mayor tamaño de mensaje que otros protocolos binarios en lugar de XML y una gran redundancia, de hecho cerca del 70% del tráfico son datos de presencia y alrededor de un 60% de estos datos son transmisiones redundantes.

- Escalabilidad

Dados los problemas de redundancia de mensajes, el protocolo puede llegar a tener una difícil escalabilidad en ciertos escenarios como salas de conversación o de suscripción.

- Limitación de envío de datos binarios

El protocolo no está preparado para el envió de datos binarios, aunque se puede simular mediante la codificación de éstos en base64. Generalmente el traspaso de datos binarios se realiza externamente utilizando XMPP como coordinador.

Inconvenientes de XMPP para la realización de este proyecto

Aparentemente XMPP puede resultar idóneo para el desarrollo de nuestra aplicación de mensajería instantánea, pero tras una investigación más profunda y una serie de pruebas este protocolo ha dejado bastante que desear en un entorno móvil.

El problema principal viene dado por la naturaleza del proyecto. Por las limitaciones de tiempo, personal y presupuesto, la implementación desde cero de un servidor XMPP parece imposible, ya que solo esta tarea tiene tal complejidad que requeriría casi la totalidad del tiempo disponible.

La opción restante es, únicamente, la utilización de librerías externas para la implantación de la aplicación móvil y una aplicación servidor de un tercero.

A fecha de 2014, la única librería disponible para la implementación de un cliente XMPP en Android con un nivel de madurez aceptable es aSmack.

Por otro lado, el único software servidor de código abierto, modificable y con una licencia adecuada para su uso en este proyecto es Openfire (licencia Apache) [9].

Tras una serie de pruebas con un cliente realizado utilizando aSmack y un servidor Openfire los resultados han sido decepcionantes, ya que aunque ambos implementan la base del protocolo XMPP y algunas de sus extensiones, pero los dos carecen de la extensión XEP-0198 (Stream Management) [11], entre otras, sin la cual nos es imposible determinar en tiempo real el estado de conexión y gestión de mensajes ACK (garantía de recibo por parte del otro cliente), de manera que si a un cliente se le envía un mensaje en el momento en que pierde o cambia la conexión, este mensaje se perderá para siempre pues el servidor aun no habrá recibido la información de que el cliente esta desconectado hasta pasados unos segundos determinados por el heartbeat (pulso) establecido.

Por otro lado, si se realiza el envío de un mensaje a un cliente que está completamente desconectado, este mensaje se perderá ya que no es almacenado en ningún lugar. Es posible lograr el almacenamiento de mensajes con la implementación de la extensión XEP-0313 (Message Archive Management), la cual no está disponible en Openfire, y una solución personalizada para el posterior envío al dispositivo móvil cuando éste esté disponible.

En resumen, estos han sido los problemas más relevantes con el uso de XMPP en dispositivos móviles. No podemos crear una solución con baja fiabilidad y sin garantía de recibo de mensajes ni tampoco podemos permitirnos el lujo de dedicar tanto tiempo a la extensión de este protocolo.

4.2.2. MQTT (Message Queue Telemetry Transport)

MQTT es un protocolo de mensajería muy ligero basado en mecanismos de publicación-suscripción.

En 1999 se escribió la especificación de MQTT [12] en busca de un protocolo que permitiese la comunicación entre dispositivos utilizando el mínimo ancho de banda y con garantía de entrega de los mensajes en un entorno con restricciones como:

- Alta latencia.
- Limitaciones en capacidad y potencia del dispositivo.
- Conexiones de red de baja fiabilidad y con grandes costos.

El protocolo se diseñó pensando en la comunicación entre sensores, sistemas de medición y pequeños dispositivos del este tipo, pero actualmente ha sido adaptado a la mensajería móvil.

Al igual que XMPP, MQTT hace uso de un servidor, denominado broker, que redirige los mensajes a los diferentes clientes suscritos al emisor.

Hoy en día la aplicación principal que hace uso de este sistema es Facebook Messenger, la cual adoptó esta solución en el año 2012 debido a los grandes problemas de latencia de los que sufrían sus mensajes. Además de esto, éste protocolo es idóneo para los dispositivos móviles ya que reduce el ancho de banda usado y el uso de batería.

Una vez más, este protocolo parece adecuado para nuestra aplicación, pero al contrario que XMPP, no está tan bien documentado y no ofrece tal nivel de soporte y librerías. El tiempo destinado a implementar este protocolo para una aplicación en Android, la modificación de alguno de los broker disponible en Internet, incluyendo el tiempo necesario para sumergirnos en el entendimiento del mismo y la posterior extensión para notificaciones push a un dispositivo Android, hacen de esta alternativa muy difícil debido al reducido presupuesto y recursos de los que disponemos .

4.2.3. GCM (Google Cloud Messaging)

El servicio GCM no es un protocolo de mensajería instantánea en sí. Concretamente se trata de un servicio de Google para notificaciones push para

Android [13], es completamente gratuito y no establece ninguna restricción en relación a la cantidad de mensajes enviados.

La tecnología push describe un estilo de comunicación sobre Internet donde la petición de la transacción se origina en el servidor, en lugar del cliente.

Esto nos permite enviar los mensajes del servidor al dispositivo móvil y que éste lo reciba en cualquier momento, aun estando bloqueado, sin tener que realizar comprobaciones periódicas de novedades. De esta manera conseguiremos una mejor gestión de recursos de red y de la batería, ya que GCM utiliza el servicio de recepción de Google Play, disponible en la mayoría de dispositivos Android, y no sería necesario utilizar otros servicios de recepción de notificaciones push. Es por esto que uno de los requisitos para hacer uso de GCM es tener instalado Google Play.

Además, el servicio GCM gestiona todos los aspectos relacionados a la cola de envío de mensajes, esto es, si un dispositivo no se encuentra disponible en algún momento, GCM mantendrá una cola con los mensajes que quedan por enviar a ese dispositivo más tarde, cuando él detecte que el dispositivo está disponible estos mensajes serán enviados en el mismo orden, garantizando el recibo del mensaje por parte del cliente. Esta característica es idónea para nuestra aplicación de mensajería instantánea, ya que nos permite externalizar la gestión del envío de mensajes a dispositivos no disponibles y lo que es más, nos garantiza su recepción.

Al igual que los otros protocolos vistos en este capítulo, GCM no permite la transferencia de archivos binarios, de forma que tendrá que complementarse con otro protocolo y actuar de coordinador para el envío de multimedia. Por otro lado, el tamaño máximo permitido para cada uno de los mensajes es de 4 KB (kilo bytes), lo cual nos resulta más que suficiente para una aplicación de mensajería instantánea siempre y cuando establezcamos un máximo de caracteres por mensaje.

La documentación sobre este servicio es amplia en Internet, además Google proporciona APIs para su uso en Android y diversos lenguajes de programación lo cual facilita enormemente su uso.

Todas estas facilidades junto a la baja latencia de envío de mensajes, hacen de GCM un candidato perfecto para utilizarlo en nuestra aplicación, la gestión de colas de mensajes y notificaciones push nos restará una gran carga de trabajo en comparación al uso de otro protocolo de comunicación, por lo tanto será esta la tecnología que usemos para enviar mensajes entre dispositivos.

Para el funcionamiento de GCM será necesario el uso de un servidor que almacenará la información sobre los usuarios que utilizan la aplicación y los identificadores de GCM de cada dispositivo registrado. Este servidor se encargará de re-dirigir los mensajes a estos dispositivos. En el apartado de arquitectura podremos encontrar más detalles sobre este tema y el funcionamiento del sistema completo.

4.3. Servidores en la nube

Los dispositivos móviles de hoy en día no son aparatos que estén siempre conectados ni disponen de conexiones a Internet de gran estabilidad. Si queremos que la comunicación a través de nuestra aplicación sea altamente fiable y de calidad, debemos contar obligatoriamente con servidores.

La aplicación que se ejecute en el servidor será la encargada de mantener un registro con los usuarios de la aplicación móvil, redirigir los mensajes a los dispositivos correspondientes y realizar comprobaciones de contactos para tareas de actualización de contactos principalmente.

Si analizamos estas funciones podemos encontrar una relación exponencial del uso de servidor con el número de usuarios de nuestra aplicación, nuestro servidor sufrirá una gran carga de trabajo a medida que el número de usuarios crezca.

Es por esto que la opción a priori de comprar un servidor exclusivo para el comienzo del proyecto puede ser una mala decisión. Si nos decantamos por un sistema con bajas prestaciones, es muy posible que en un futuro cercano haya que reemplazarlo, y así continuamente. De otro lado, un gran desembolso en una arquitectura de servidores puede ser arriesgado si la aplicación no cuaja en el mercado.

Entonces, ¿Cuál es la mejor solución? ¡La computación en la nube!

Es indiscutible la importancia que los servicios en la nube están cobrando hoy en día. La totalidad de las grandes empresas del sector informático hablan de que es el futuro de la computación. A continuación veremos las ventajas que nos ofrecen estos servicios:

- **Auto escalado**

  Los servicios en la nube nos permiten montar una arquitectura flexible con un fácil auto escalado. De esta manera si nuestras necesidades crecen, podremos aumentar la potencia contratada con tan solo unos clics o incluso mediante una configuración automática. Por el contrario, si el uso de los servidores decrece, se cerrarán automáticamente algunas instancias que se estén ejecutando de forma que solo pagaremos por las necesarias en el momento y sin ningún desembolso inicial importante.

- **Balance de carga**

  Mediante los diferentes configuradores de los distintos proveedores de servicios en la nube, es fácil configurar balanceadores de carga que nos permitan tener una carga de trabajo similar en nuestras instancias, evitando así sobrecargas y ralentizaciones en los servidores más usados.

- **Alta disponibilidad**

  Además de la alta disponibilidad de componentes y la atención 24/7 del data center, la arquitectura en la nube nos permite diseñar entornos dinámicos de alta disponibilidad, de manera que si algo falla se lanzarán automáticamente nuevas instancias para no detener el servicio ni por unos minutos.

- **Recuperación ante desastres**

  Existen múltiples servicios disponibles para la recuperación de información en caso de desastres. Desde sincronización de bases de datos

hasta servidores de copias de seguridad diarias, por los cuales no habrá que pagar más que las horas usadas, en este caso las que hagan falta para subir la copia de seguridad. En caso de desastre, estas instancias con copias de seguridad pueden ser lanzadas en cuestión de segundos para continuar con el correcto funcionamiento del sistema.

- **Gestión de servicios**

Existen muchas empresas dedicadas a la gestión de servicios en la nube. La externalización de tareas de mantenimiento y configuración de servidores puede ser una ventaja en nuestro caso, pues nosotros vamos a centrarnos principalmente en el desarrollo de nuestra aplicación móvil.

SaaS, IaaS o PaaS. ¿Qué son y cuál necesitamos?

Para realizar la elección de proveedor de servicios en la nube debemos tener en cuenta que existen tres conceptos diferentes a los que nos podemos referir, SaaS (Software as a Service), IaaS (Infrastructure as a Service) y PaaS (Platform as a Service) [14] [15].

SaaS es el concepto más extendido de la nube. Se trata de aplicaciones de terceros que funcionan remotamente, sin la necesidad de instalación y configuración de las mismas, como por ejemplo Google Docs u Office 365. Este tipo de servicios no es interesante para nosotros, ya que queremos publicar nuestra aplicación propia.

IaaS nos ofrece servicios en los cuales lo que contratamos es una infraestructura que debemos gestionar y configurar nosotros mismos. Esta infraestructura puede estar formada de potencia de computación, almacenamiento, redes, etc.

Muchas de las ventajas descritas anteriormente están relacionadas a este modelo de servicio, ya que es el más flexible y que más oportunidades nos ofrece. Además, por lo general, los gastos son variables y se suele pagar solo por el tiempo que se use.

Amazon Web Services (AWS) y Microsoft Azure son los proveedores principales de estos servicios. En concreto, para este proyecto se ha evaluado la posibilidad de utilizar AWS y se han podido comprobar las infinitas posibilidades que estos servicios ofrecen. En cualquier caso, no es posible el uso gratuito de AWS y además la complejidad de configuración y arquitectura de la infraestructura es muy superior a las soluciones PaaS.

Finalmente, **PaaS** nos ofrece la contratación de un servidor de aplicaciones, donde se podrán ejecutar nuestras aplicaciones y bases de datos. Esta modalidad está pensada principalmente para el despliegue de aplicaciones web, eliminando casi en su totalidad las configuraciones de la arquitectura del sistema, de manera que todos los parámetros de escalabilidad, alta disponibilidad y demás pueden ser configurados fácilmente desde un panel de opciones en el portal del proveedor.

Una vez claros estos conceptos, podemos concluir que lo que necesitamos es un proveedor de servicios PaaS en el cual desplegar nuestra aplicación de manera que perdamos el menor tiempo posible en configuraciones de hardware, redes etc.

Tras evaluar algunas opciones se ha decidido optar por el servicio **Google App Engine** (GAE) que nos ofrece las siguientes ventajas:

- Uso gratuito de la plataforma hasta unos límites más que suficientes para la realización de este proyecto. Una vez pasados estos límites las tarifas son de las más bajas dentro del mercado.
- Soporte de Java (entre otros), lenguaje de programación sobre el que realizaremos la aplicación en Android. Realizar ambas partes en el mismo lenguaje nos resultará más cómodo a la hora del desarrollo.
- Fácil integración con el resto de tecnologías que utilizaremos en el proyecto. GAE proporciona una extensión para eclipse, para desarrollar aplicaciones web específicas para esta plataforma, lo cual nos ofrece muchas facilidades, además de despliegue automático desde eclipse. Además, al ser producto de Google está muy integrado con otros de sus servicios de los que haremos uso como GCM.

- Muy bien documentado. Al ser uno de los principales proveedores de PaaS es relativamente fácil encontrar información sobre su uso en Internet.

4.4. Servicio de traducción

La característica distintiva de nuestro producto es el soporte a la conversación entre diferentes idiomas, es por esto que el servicio de traducción juega un papel muy importante.

Las cualidades esenciales del servicio a usar para nuestra aplicación son:

- **Alta disponibilidad**

Es imprescindible que el servicio a usar nos garantiza una alta disponibilidad. El funcionamiento de la aplicación dependerá en gran medida de este servicio, de manera que no es posible consentir intervalos de tiempo en los que el servicio de traducción no este activo.

- **Rapidez**

Al tratarse de un cliente de mensajería instantánea, el retraso que sufre un mensaje desde el envío del emisor al receptor es importante. La mayor parte de tiempo que compondrá este retraso se verá determinada por la calidad de conexión y carga del servidor, pero el servicio de traducción jugara también un papel importante. La falta de eficacia del servicio de traducción puede convertirse en un problema claro para la experiencia de usuario.

- **Facilidad de uso**

Dado el número de servicios de traducción disponibles, lo ideal sería optar por un servicio web que nos proporcione una API a la que hacer llamadas con la información esencial: lenguaje de origen, lenguaje de destino y texto a traducir. La facilidad de uso del servicio también jugará su papel en las tareas de diseño e implementación, pudiendo así reducir la carga de trabajo.

- **Soporte de múltiples idiomas**

Uno de los objetivos de la aplicación es llegar al mayor número de público posible. El soporte del mayor número de idiomas posibles sumará puntos a favor, a la hora de captar usuarios que utilicen estos idiomas.

- **Calidad de la traducción**

Existen numerosos servicios en el mercado, pero no todos traducen de la misma manera. Es necesaria una calidad mínima en el servicio de traducción, que nos garantice, que el nivel de entendimiento entre emisor y receptor será aceptable para conseguir establecer una conversación coherente entre las dos partes.

- **Fijación de precio**

Dado que el número de llamadas a este servicio incrementará proporcionalmente al número de usuarios que utilicen la aplicación, la política e precios deberá ser adecuada para el presupuesto disponible para el mantenimiento.

Son varios los servicios de traducción que se han evaluado en este apartado. En la tabla a continuación se muestran los 3 candidatos principales y sus características a tener en cuenta:

	Google Translate	Bing Translator	Yandex translate
Idiomas soportados	80	44	43
Detección de lenguaje automática	Sí	Sí	Sí
Límite de caracteres gratuitos	No	2 Millones de caracteres gratuitos al mes	1 Millón de caracteres al día
Traducción de caracteres de pago	$20 por cada millón de caracteres	Tabla de precios. Descendente en relación $/caracteres	Es necesario contactar al proveedor para determinar el precio.
Calidad de traducción	Buena	Media	Media

Figura 4.2 Tabla comparativa de servicios de traducción

Si el presupuesto no fuera problema, el servicio ganador de los estudiados sería claramente el de Google Translate. Soporte a 80 idiomas y una calidad de traducción superior a las de sus rivales hacen de este servicio el de mayor calidad.

No obstante su precio es prohibitivo para el presupuesto de manteamiento de este trabajo de fin de grado. Para traducir unos 30 millones de caracteres al mes necesitaríamos un presupuesto de aproximadamente 600$ solo para mantener este servicio.

Así pues la decisión entre los dos restantes es fácil, los servicios Bing Translator y Yandex Translate tienen unas características muy similares, con la diferencia de que el servicio de Yandex nos permite la traducción de 1 millón de caracteres gratuitos al día;

La decisión final está tomada, Yandex translate será el servicio de traducción que utilizaremos en nuestra aplicación.

4.5. Git

Git es un software de control de versiones [16], cuya popularidad ha ido creciendo enormemente durante los últimos años.

Aunque un sistema de control de versiones puede realizarse de forma manual, suele ser muy aconsejable la utilización de software que ayude a su gestión, particularmente en proyectos con un gran número de colaboradores.

Este tipo de sistemas nos permiten seguir y controlar los cambios realizados en los ficheros de un proyecto, tanto de código fuente como de otros recursos. Estos proyectos, se almacenan en un repositorio, del cual podremos llevar un historial de cambios y, entre otras funcionalidades, nos ofrecerá la posibilidad de volver a un punto del pasado y comprobar el código en caso de aparición de problemas en las versiones posteriores de alguno de los ficheros del repositorio.

En concreto, la arquitectura de almacenamiento de GIT es de naturaleza distribuida, lo que quiere decir que cada usuario tiene su propio repositorio, lo que le permitirá trabajar de forma independiente. Aun así, GIT también dispone de un repositorio central donde se sincronizaran las contribuciones de los diferentes integrantes del proyecto.

Por tanto la arquitectura de nuestro sistema de versiones tendrá la siguiente disposición.

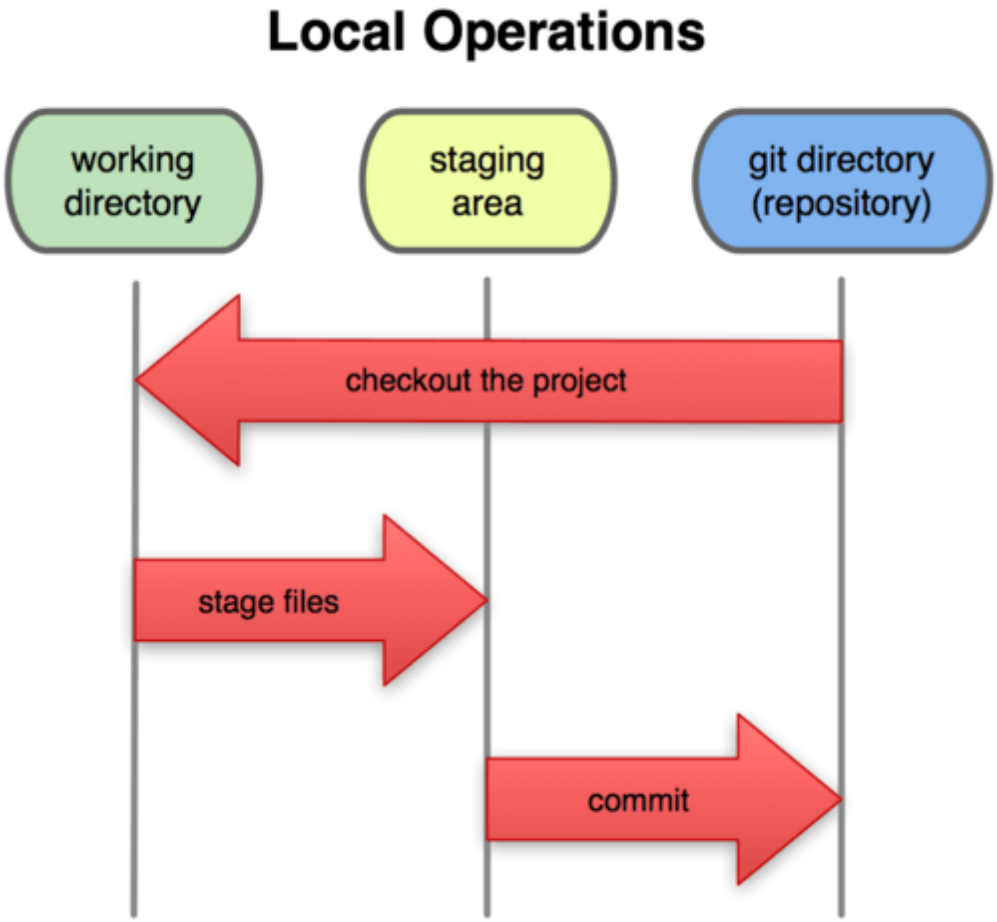


Figura 4.3 Arquitectura de Git [17]

El usuario dispone de su repositorio local (working directory), en el cual puede recibir los proyectos del repositorio y sus cambios, así como hacer las modificaciones correspondientes a cualquiera de sus archivos.

Una vez realizadas las modificaciones o incorporaciones, el usuario guardará los cambios en su repositorio local, en la figura anterior denominado como (staging area). Si los cambios son almacenados en la staging area, estos estarán listos para ser mezclados con el repositorio central en el próximo commit.

Una vez realizado un commit, todos los cambios guardados en la staging area serán sincronizados con el repositorio central. Si sucede algún conflicto con nuevos cambios realizados por otra persona, se nos dará la oportunidad de ver los cambios de ambos y elegir cuales perdurarán.

El servicio web que proporcionará el sistema de versiones GIT para nuestro proyecto será Bitbucket, dado que ofrece la posibilidad de mantener repositorios personales sin ningún coste.

Por razones de simplicidad, utilizaremos la extensión EGit de Eclipse. EGit es la extensión más madura y utilizada para Git en Eclipse. Entre sus ventajas nos encontramos con la facilidad de uso y la integración de una interfaz gráfica en lugar de tradicional consola de comandos.

CAPÍTULO 5

ANÁLISIS

5. Análisis

5.1. Arquitectura del sistema

Dada la reducida disponibilidad de los dispositivos móviles es absolutamente necesario utilizar una arquitectura que nos asegure el envío y recibo de mensajes entre dispositivos en cuanto estén disponibles.

El sistema se compondrá de 4 elementos principales:

- Cliente
- Servidor
- Google Cloud Messaging (GCM).
- Servicio de traducción (Yandex).

En la figura 5.1 podemos ver el funcionamiento del sistema completo de forma simplificada. Como podemos apreciar, en primer lugar se realiza un registro del dispositivo en GCM, este registro es completamente necesario ya que sin un identificador de GCM será imposible el registro en el servidor y el recibo de mensajes.

Tras este primer paso se procede al envío de mensajes directamente entre el dispositivo y el servidor. Estos mensajes no son únicamente mensajes de texto de las conversaciones, sino que se incluyen mensajes de registro, de petición de actualización de contactos etc.

La aplicación del dispositivo móvil dispondrá de un mecanismo de confirmación de recibo de mensajes por parte del servidor, de manera que si alguno de los mensajes enviados no es recibido o no ha devuelto la respuesta esperada (en actualización de contactos por ejemplo) el mensaje será almacenado y se volverá a intentar su envío automáticamente en cuanto haya conexión a Internet disponible. De esta manera garantizamos la no pérdida de mensajes aumentando así la fiabilidad de nuestra aplicación.

Una vez recibido el mensaje por parte del servidor, se realizará un mapeado del número de teléfono de destino a su identificador de GCM y se realizará el envío al servicio de Google.

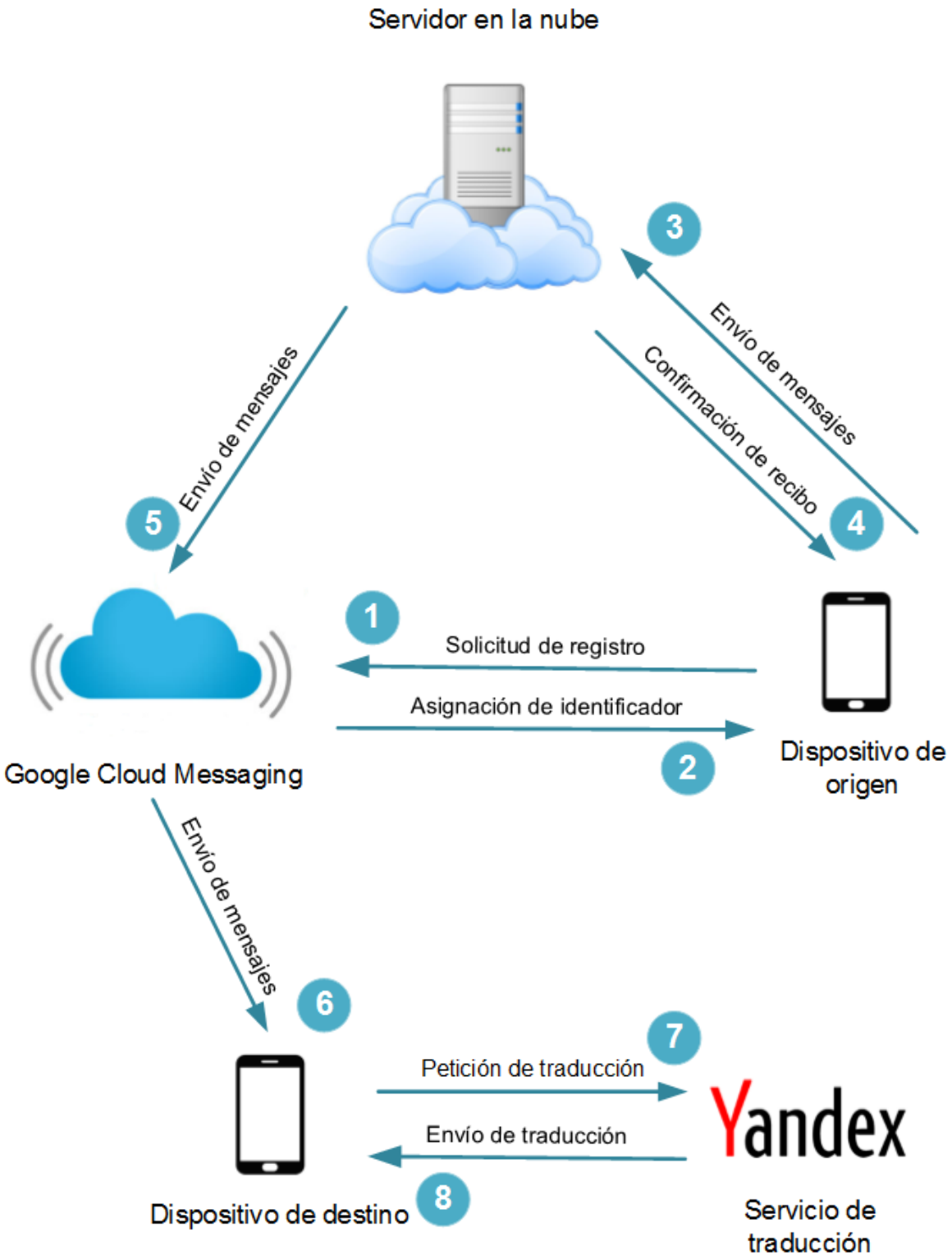


Figura 5.1 Arquitectura del sistema

El servicio GCM se encargará de almacenar el mensaje y enviarlo al dispositivo en cuanto se encuentre disponible.

Una vez recibido el mensaje en el dispositivo de destino se continuará con la traducción si procede. Se realizará una petición de traducción al servicio de traducción Yandex y se esperará a su respuesta.

Finalmente cuando se disponga del mensaje traducido, se mostrarán en pantalla tanto éste como el mensaje original.

5.2. Perfil de usuario

El tipo de usuario al que se dirigirá la aplicación tendrá un gran peso en la determinación del funcionamiento de la aplicación, adicción de opciones y sobre todo en la interfaz.

Los aspectos más importantes a la hora de definir el perfil de usuario en una aplicación son: localización, rango de edades y educación en relación a la capacidad del uso de dispositivos móviles.

Localización

Diferentes países tienen diferentes costumbres y gustos. Esto es algo innegable a la hora de dirigir un producto a un mercado.

La globalización de dispositivos móviles y de aplicaciones en mercados mundiales ha reducido esta diferencia en gran medida, ayudando a unir la acomodación de distintos estilos de vida en un mismo software.

No obstante aún seguimos encontrando rasgos, principalmente en la interfaz, que delatan el destino de algunas aplicaciones.

Un ejemplo claro es la cultura Japonesa. En su cultura es habitual el uso de exuberantes animaciones y colores y sonidos muy llamativos principalmente en contenido audiovisual como programas de televisión o anuncios.

Como vimos en el Capítulo 2, existe una gran demanda de servicios de mensajería instantánea a lo largo de todo el mundo, es por esto que no debemos dirigirnos a una localización determinada o un idioma especifico, trataremos de plantear la aplicación de la forma más global posible con la meta de conseguir un alcance mundial, evitando así elementos demasiado llamativos o la sobrecarga de animaciones.

Rango de edades

Una vez más, la interfaz gráfica va a depender en gran medida del público al que nos dirijamos en cuanto al rango de edades. Los gustos no son los mismos para jóvenes de 15 años como para personas mayores de 60.

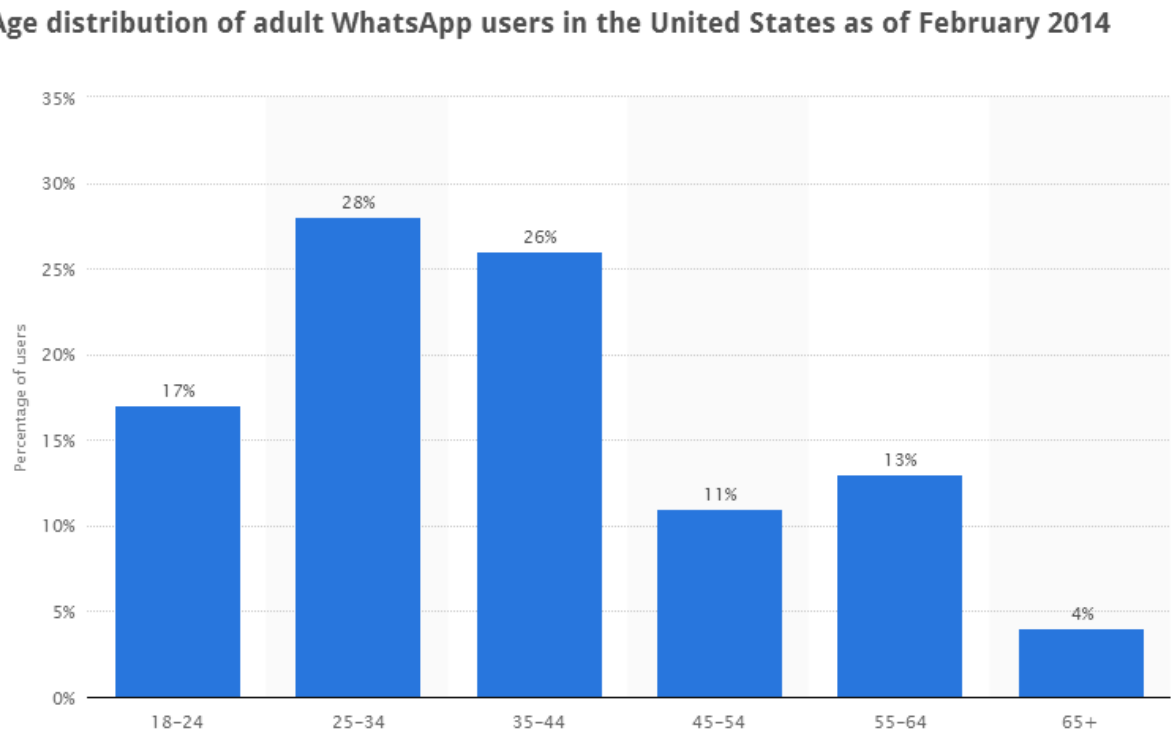


Figura 5.2 Distribución de usuarios de WhatsApp por rangos de edad a fecha de Febrero de 2014. [18]

El estudio presentado en la figura 5.2 nos muestra la distribución de edades de los usuarios de WhatsApp hasta la fecha de Febrero de 2014.

Podemos observar como entre los usuarios de esta aplicación, que recuerdo que es la más usada del mundo, predominan las edades entre 25-44 años con un 54% de usuarios, un rango bastante amplio. Aun así, no debemos olvidarnos del

resto de grupos, como el rango de 18-24 años con un 17% o de 55-64 años, ya que estos pequeños porcentajes representan una gran cantidad de usuarios, en torno a 9 millones de usuarios en el rango de 18-24 años y en torno a 7 millones de usuarios en el rango de 55-64 años de edad.

Este estudio nos lleva a la conclusión de que no podemos centrarnos en un rango de edad específico para una aplicación de este tipo. La distribución es demasiado dispersa y en cualquier rango de edad hay millones de usuarios potenciales.

Educación tecnológica

El grado de educación y soltura de un usuario potencial con las aplicaciones de los nuevos dispositivos móviles es decisivo a la hora de determinar la interfaz y las funcionalidades que una aplicación va a tener.

Usuarios con poca experiencia en este tipo de aplicaciones se pueden ver abrumados con una abundante cantidad de funcionalidades, al añadir complejidad al uso de la aplicación. Por otro lado, usuarios muy exigentes y expertos valoran todo tipo de opciones, personalizaciones y características avanzadas.

Por otro lado una interfaz de usuario moderna, con múltiples gestos y opciones puede suponer una barrera para usuarios con baja educación tecnológica, mientras que a usuarios expertos les puede resultar más atractiva.

Podemos asumir que hoy en día la educación en cuanto a uso de software móvil viene dada en gran medida por la edad del individuo. Por lo general gente mayor tendrá menos habilidad que personas más jóvenes.

Es por esto que, una vez más, nos encontramos ante un parámetro muy generalizado, no podemos centrarnos en un solo tipo de usuario dada lo dispersado que estará nuestro público.

Conclusión

Nos encontramos ante un tipo de aplicación lo más global posible, el alcance será mundial, el rango de edades total, y el grado de educación muy disperso.

Así pues nos decantaremos por una solución acorde a todos los usuarios, un diseño simple que permita el uso de la aplicación para todos los niveles de educación y no resulte demasiado extravagante.

5.3. Análisis de requisitos (funcionales y no funcionales)

En esta fase del proyecto vamos a definir los requisitos funcionales y requisitos no funcionales.

El objetivo será conseguir una lista de tareas a realizar para tener en cuenta a la hora de diseño e implementación.

5.3.1. Requisitos funcionales

Un requisito funcional define una función que el sistema debe permitir llevar a cabo. Estos requisitos generalmente vienen dados fruto de una conversación con el cliente en el cual se declaran cuales son espáticamente las funcionalidades del software a desarrollar.

Al hacer nosotros mismos el papel de cliente, la idea general de nuestros requisitos vendrán dados por los objetivos del proyecto de manera que nos servirá de base para la definición específica de los requisitos individuales.

Los requisitos funcionales de nuestra aplicación serán:

- Un usuario puede ver la lista de conversaciones.
- Un usuario puede ver el historial de conversación.
- Un usuario puede ver la lista de contactos.
- Un usuario puede actualizar su lista de contactos.
- Un usuario puede enviar un mensaje a un contacto.
- Un usuario puede cambiar el idioma de origen de un contacto.
- Un usuario puede abrir el menú de opciones.
- Un usuario puede ver su perfil.
- Un usuario puede activar/desactivar el sonido de las notificaciones.
- Un usuario puede cambiar el idioma de la aplicación.

- Un usuario puede cambiar el idioma de destino de la traducción de forma manual.
- Un usuario puede cambiar el idioma de destino de la traducción con detección automática.
- Un usuario puede copiar un mensaje de una conversación fácilmente.
- Un usuario puede ver si su mensaje ha sido enviado o aún está en espera.

5.3.2. Requisitos no funcionales

Se trata de los requisitos que no describen funciones a realizar por la aplicación.

- Soporte para versiones Android 3.0 o superior.
- Adaptación de la interfaz gráfica a diferentes tamaños de pantalla.
- Compatibilidad con dispositivos en diferentes idiomas.
- Compatibilidad con números de teléfonos de múltiples países.
- Tiempo de envío de mensajes aceptable.
- Tiempo de respuesta del servicio de traducción aceptable.
- Uso mínimo del ancho de banda y consumo de datos.
- Garantía de la no pérdida de mensajes por la no disponibilidad del receptor.
- Garantía de la no pérdida de mensajes por la falta de conectividad del emisor.
- Envío automático de mensajes en cuanto se disponga de conexión.
- Alta disponibilidad del servidor.
- Alta disponibilidad del servicio de traducción.
- Adaptación de la potencia de computación según crezcan o disminuyan las necesidades.
- La totalidad del texto mostrado en la aplicación debe estar traducido a los idiomas soportados.
- Diseño de una interfaz simple y sencilla, que posibilite el uso de la aplicación a usuarios con cualquier nivel de conocimiento sobre el uso de dispositivos móviles.

- Usabilidad. No debe hacer bloqueos de la interfaz de usuario mientras se realizan algunos procesos de la aplicación.
- Realización de pruebas para detección de errores.
- Documento con las especificaciones de las funcionalidades del sistema que permita al usuario descubrir todas las posibilidades de la aplicación. Para esto se redactará un manual de usuario.

5.4. Casos de uso

En éste apartado analizaremos los casos de uso definidos previamente por los requisitos funcionales.

Como podemos observar, en nuestro sistema tan solo participará un actor, el cual hará el papel de usuario.

Este usuario será el que interactúe con el sistema y realice las tareas definidas en los requisitos funcionales.

Para realizar el diagrama general de casos de uso hemos de tener en cuenta que un caso de uso puede incluir o extender a otro caso de uso, esto es.

Incluir un caso de uso dentro de otro significa que el caso de uso que incluye no puede ser realizado sin el caso de uso incluido. Utilizamos esta relación para evitar la repetición del mismo de caso de uso en varios lugares en nuestro esquema, ya que dispondremos de casos de uso como acceder al menú de opciones, que se repiten continuamente para completar ciertos requisitos funcionales.

Extender un caso de uso nos indica que el caso de uso que extiende al original puede ser un paso adicional o opción tras realizar el caso de uso original. En ciertos momentos, como cuando nos encontremos en la pantalla de conversación con un contacto, tendremos varias opciones disponibles a realizar que extenderán del caso de uso de visualizar la conversación.

Tras plasmar todos los requisitos funcionales y establecer las relaciones de extensión e inclusión el diagrama resultante es el siguiente:

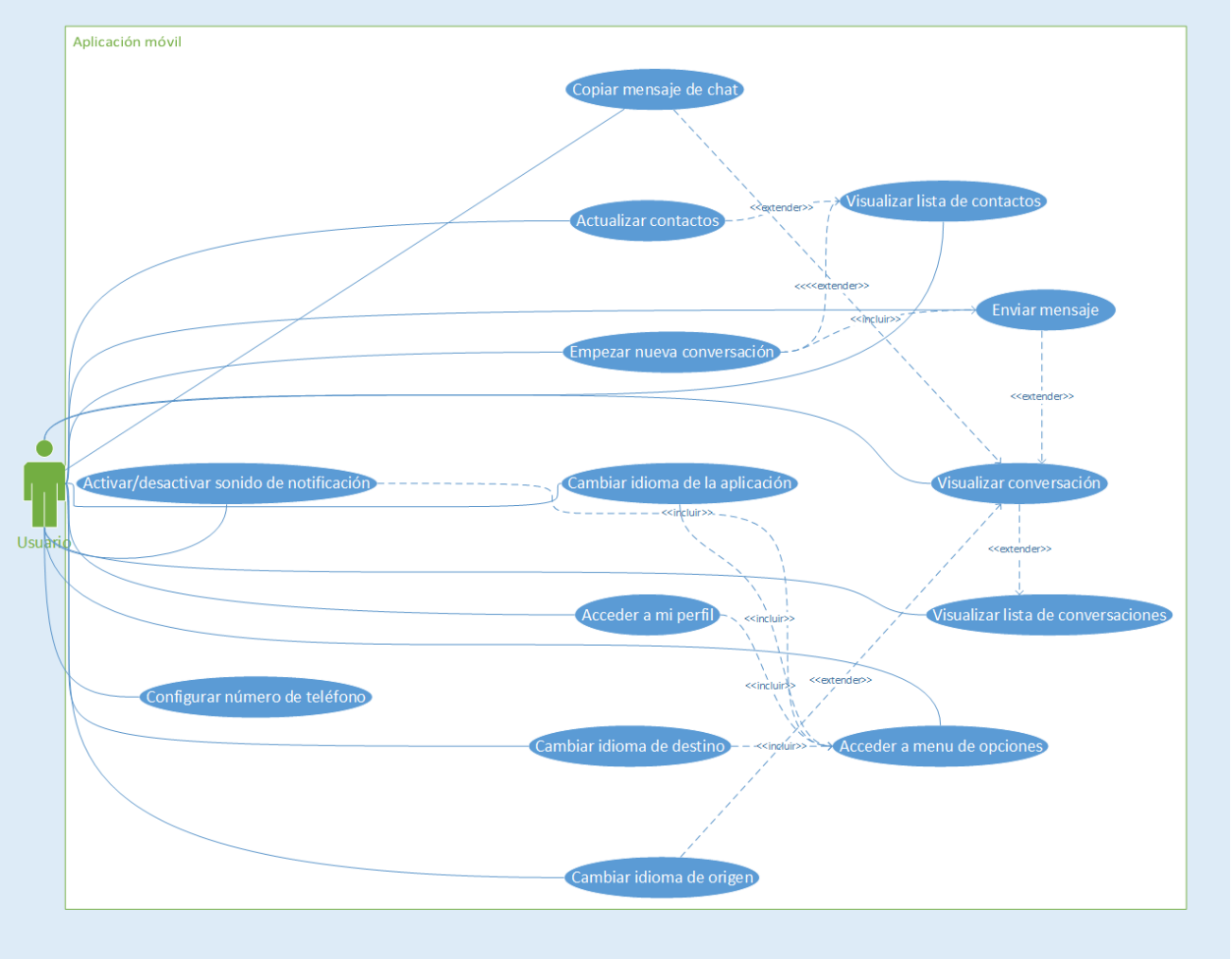


Figura 5.3 Diagrama de casos de uso

Una vez diseñado el diagrama general pasaremos a detallar cada caso de uso. Para esto tendremos en cuenta los siguientes apartados, algunos de ellos optativos:

- **Actor principal.** El individuo que comienza la interacción con el sistema
- **Condiciones de entrada.** Requisitos previos necesarios para empezar.
- **Extiende.** Si extiende algún caso de uso.
- **Incluye.** Si incluye algún caso de uso.
- **Objetivo.** Descripción del caso de uso, qué se quiere conseguir.
- **Camino principal.** Descripción, paso a paso, de las acciones que hay llevar a cabo para la correcta consecución del objetivo.

- **Excepciones.** Casos excepcionales en los que el caso de uso no se cumplirá.
- **Condiciones de salida.** Requisitos necesarios para la finalización del caso de uso

CASO DE USO – 1	
CONFIGURAR NÚMERO DE TELÉFONO	
Actor principal	El usuario.
Condiciones de entrada	Ninguna.
Objetivo	Introducir un número de teléfono para registrarse en el sistema.
Camino principal	1. El usuario inicia la aplicación por primera vez.
	2. El usuario introduce su número de teléfono.
	3. El usuario presiona el botón empezar.
Excepciones	El teléfono debe tener entre 9 y 13 dígitos de longitud.
Condiciones de salida	El usuario presiona el botón empezar y el número introducido tiene entre 9 y 13 dígitos.

Figura 5.4 Caso de uso 1

CASO DE USO – 2	
VISUALIZAR LISTA DE CONVERSACIONES	
Actor principal	El usuario.
Condiciones de entrada	Que exista alguna conversación, de otra manera no se mostrará ningún elemento en la lista.
Objetivo	Mostrar al usuario la lista de conversaciones empezadas para ofrecer nuevas posibilidades posteriormente.
Camino principal	1. Completar el caso de uso 1. “Configurar número de teléfono”
	2. Configurar número de teléfono.
Excepciones	Si no existe ninguna conversación no se mostrará ningún elemento en la lista.

Figura 5.5 Caso de uso 2

CASO DE USO – 3

VISUALIZAR CONVERSACIÓN	
Actor principal	El usuario.
Extiende	2. Visualizar lista de conversaciones.
Condiciones de entrada	Debe existir alguna conversación disponible.
Objetivo	Visualizar los mensajes entre el usuario y el contacto seleccionado.
Camino principal	1. Completar el caso de uso 2. “Visualizar la lista de conversaciones”.
	2. seleccionamos una de las conversaciones en la lista.

Figura 5.6 Caso de uso 3

CASO DE USO – 4

ENVIAR MENSAJE	
Actor principal	El usuario.
Extiende	3. “Visualizar conversación”.
Condiciones de entrada	La ejecución del caso de uso 3. “Visualizar conversación” ha sido satisfactoria.
Objetivo	El usuario introduce un mensaje y lo envía al contacto con el cual está teniendo la conversación.
Camino principal	1. Completar el caso de uso 3. “Visualizar conversación”.
	2. El usuario escribe un mensaje en el cuadro de texto inferior de la pantalla.
	3. El usuario pulsa el botón de enviar mensaje.
Excepciones	El mensaje está vacío o tiene una longitud superior de 300 caracteres. En estos casos el mensaje no se enviará.

Figura 5.7 Caso de uso 4

CASO DE USO – 5

VISUALIZAR LISTA DE CONTACTOS	
Actor principal	El usuario.
Condiciones de entrada	Ninguna.
Objetivo	Mostrar una lista con los contactos que utilizan la aplicación.
Camino principal	1. Localizarse en la pantalla principal de la aplicación. 2. Pulsar el botón de iniciar nueva conversación, situado a la derecha en la barra superior.
Excepciones	Si no existe ningún contacto no se mostrará ninguno.

Figura 5.8 Caso de uso 5

CASO DE USO – 6

ACTUALIZAR CONTACTOS	
Actor principal	El usuario.
Extiende	5. Visualizar lista de contactos.
Condiciones de entrada	Ninguna.
Objetivo	Actualizar la lista de contactos disponibles en la aplicación.
Camino principal	1. Completar el caso de uso 5. “Visualizar lista de contactos”. 2. Pulsar el botón actualizar contactos situado a la derecha en la barra superior.
Excepciones	Ninguna.

Figura 5.9 Caso de uso 6

CASO DE USO – 7

EMPEZAR NUEVA CONVERSACIÓN	
Actor principal	El usuario.
Extiende	Visualizar lista de contactos.
Incluye	3. Visualizar conversación.
	4. Enviar mensaje.
Condiciones de entrada	Se dispone de algún contacto con el que aún no se ha empezado una conversación.
Objetivo	Empezar una conversación con algún contacto con el cual aún no se tiene ninguna, de manera que podamos acceder directamente desde la lista de conversaciones.
Camino principal	1. Completar el caso de uso 5. “Visualizar lista de contactos”.
	2. Seleccionar un contacto de manera que suceda el caso 3. “Visualizar conversación”.
	3. Realizar el caso de uso 4. “Enviar mensaje”.
Excepciones	Ninguna.

Figura 5.10 Caso de uso 7

CASO DE USO – 8

CAMBIAR IDIOMA DE ORIGEN	
Actor principal	El usuario.
Extiende	5. Visualizar conversación.
Condiciones de entrada	Ninguna.
Objetivo	Modificar el idioma de origen del que vamos a traducir los mensajes recibidos de este contacto.
Camino principal	1. Completar el caso de uso 3. “Visualizar conversación”.
	2. Pulsar el botón de traducción situado a la derecha en la barra superior de la conversación.
	3. Elegir el idioma deseado en la lista.

Figura 5.11 Caso de uso 8

CASO DE USO – 9

ACCEDER A MENÚ DE OPCIONES	
Actor principal	El usuario.
Condiciones de entrada	Estar en la pantalla principal.
Objetivo	Mostrar el menú de opciones para posteriormente seleccionar alguna.
Camino principal	1. Hallarse en la pantalla principal. 2. Pulsar el icono de opciones situado a la izquierda en la barra superior.

Figura 5.12 Caso de uso 9

CASO DE USO – 10

ACCEDER A MI PERFIL	
Actor principal	El usuario.
Incluye	9. Acceder al menú de opciones.
Condiciones de entrada	Ninguna.
Objetivo	Mostrar el perfil de usuario.
Camino principal	1. Completar el caso de uso 9. “Acceder al menú de opciones”. 2. Presionar sobre el primer elemento de la lista del menú de opciones, “Perfil”.

Figura 5.13 Caso de uso 10

CASO DE USO – 11

CAMBIAR IDIOMA DE DESTINO	
Actor principal	El usuario.
Incluye	9. Acceder al menú de opciones.
Condiciones de entrada	Ninguna.
Objetivo	Mostrar el perfil de usuario.
Camino principal	1. Completar el caso de uso 9. “Acceder al menú de opciones”.
	2. Presionar sobre el tercer elemento de la lista del menú de opciones, “Notificaciones”.
	3. Activar/Desactivar botón de sonido.

Figura 5.14 Caso de uso 11

CASO DE USO – 12

CAMBIAR IDIOMA DE LA APLICACIÓN	
Actor principal	El usuario.
Incluye	9. Acceder al menú de opciones.
Condiciones de entrada	Ninguna.
Camino principal	1. Completar el caso de uso 9. “Acceder al menú de opciones”.
	2. Presionar sobre el segundo elemento de la lista del menú de opciones, “Idioma”.
	3. Seleccionar uno de los idiomas de la lista mostrada.

Figura 5.15 Caso de uso 12

CAPÍTULO 6:

DISEÑO

6. Diseño

6.1. Diagramas de secuencia

A partir de los diagramas de secuencia obtenemos la posibilidad de modelar la interacción entre los diferentes actores y objetos de un sistema.

Cada uno de estos diagramas está basado en un proceso en un determinado intervalo de tiempo, de manera que nos sea posible dejar constancia del orden de las acciones, en este caso representadas por mensajes, que se llevan a cabo por los objetos del sistema.

Nos ayudaran a entender el proceso de comunicación de los diferentes elementos que componen el sistema como usuarios, aplicaciones y servidores, lo que nos ayudara a comprender el sistema de una forma global.

6.1.1. Diagramas de secuencia de alto nivel

El proceso de comunicación entre los elementos del sistema aún no ha sido definido con exactitud. Al disponer de una escala temporal en este diagrama seremos capaces de apreciar el orden y los elementos involucrados.

Inicio de aplicación

Objetivo:
Mostrar el proceso de comunicación que se produce al iniciar la aplicación por primera vez.

En este diagrama se incluyen también el proceso de registro en el servidor y el proceso de actualización de contactos.

1. El usuario inicia la aplicación por primera vez
2. La aplicación envía la solicitud de registro a GCM
3. GCM devuelve el identificador de registro al dispositivo móvil
4. La aplicación envía la solicitud de registro al servidor de nuestro sistema
5. El servidor responde con la confirmación del registro

6. La aplicación envía la primera solicitud de actualización de contactos
7. El servidor envía la lista de nuestros contactos que utilizan la aplicación.

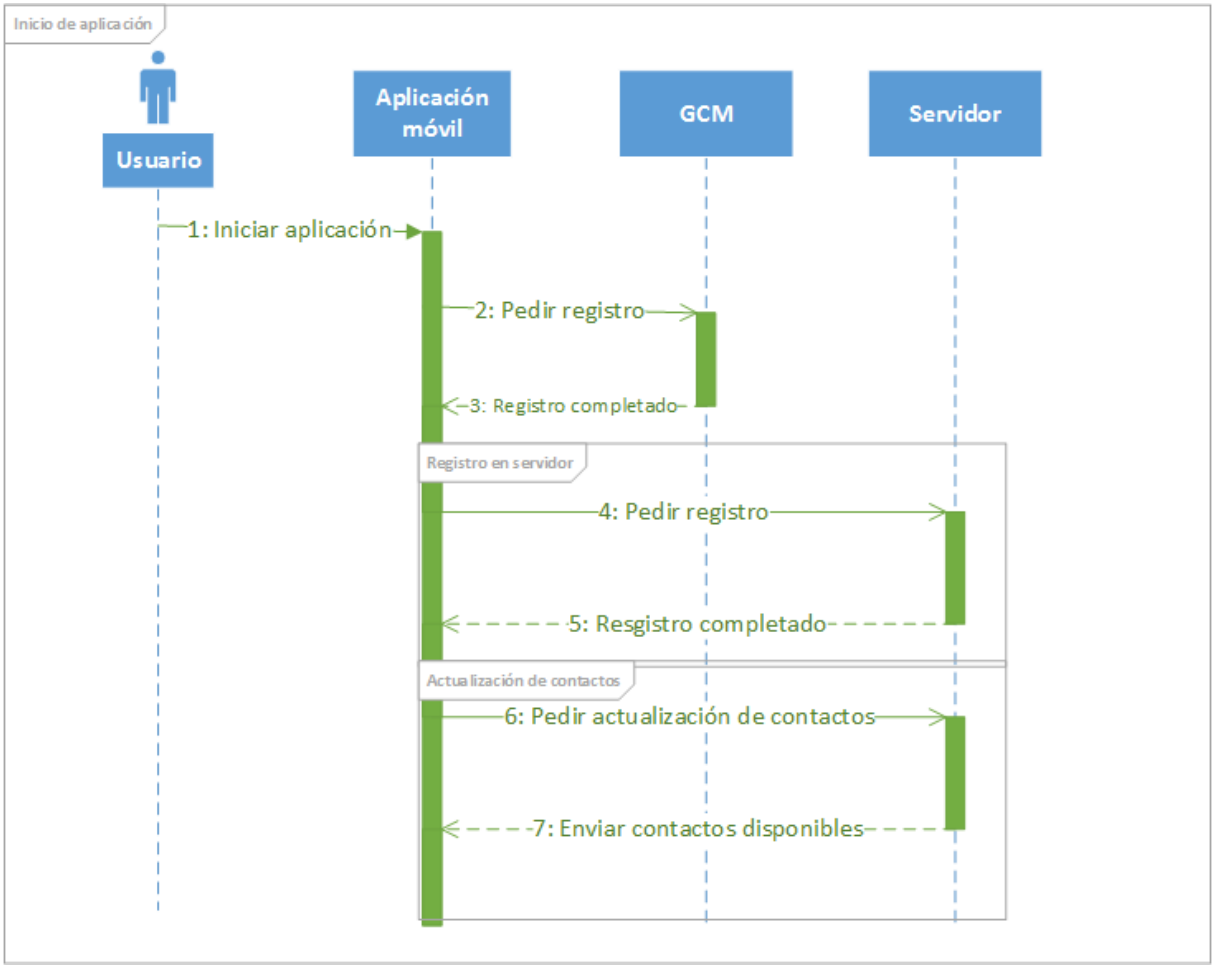


Figura 6.1 Diagrama de secuencia de inicio de la aplicación

Envío de mensaje

Objetivo:

Mostrar el proceso de envío y recepción de un mensaje.

1. El usuario envía un mensaje en la aplicación
2. La aplicación realiza una operación HTTP Post al servidor con el mensaje y la información necesaria

3. El servidor devuelve la confirmación de que el mensaje se ha recibido
4. El servidor envía el mensaje al servicio GCM
5. Cuando el dispositivo de destino esté disponible, GCM enviara el mensaje a la aplicación móvil de destino.

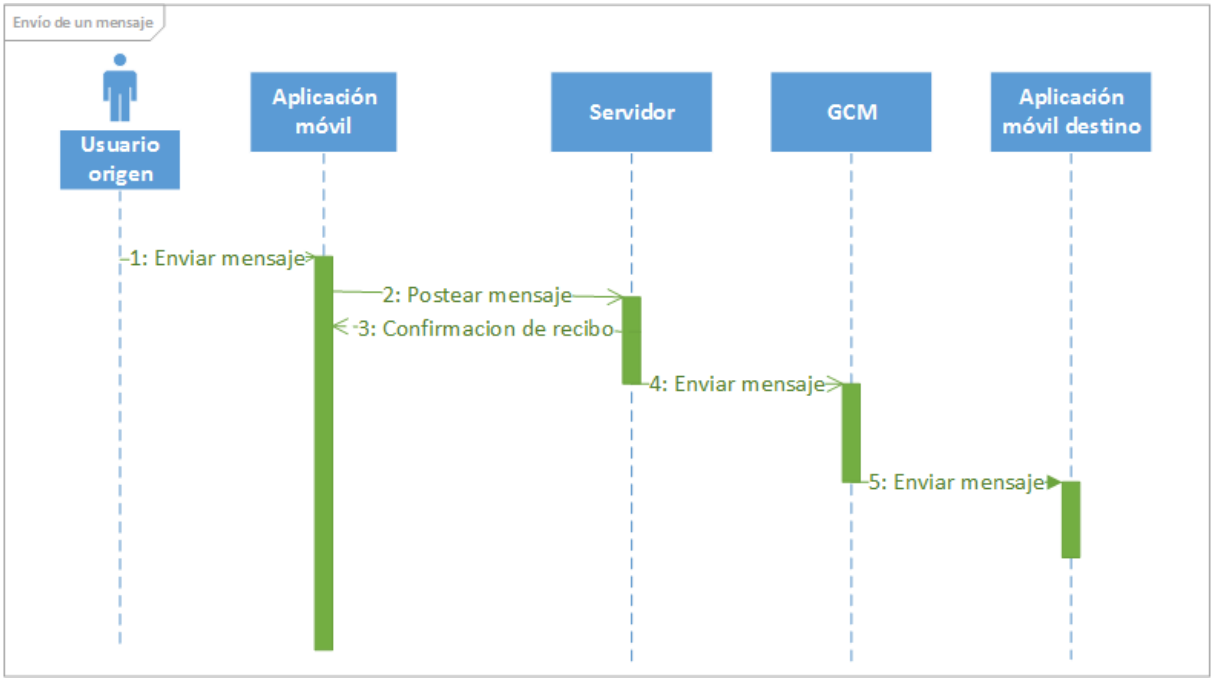


Figura 6.2 Diagrama de secuencia de envío de un mensaje

6.2. Diseño de la interfaz

Una vez definidas las funciones que será capaz de realizar nuestra aplicación, es hora de diseñar la interfaz gráfica con la que el usuario va a interactuar.

Este es un aspecto muy importante para nuestra aplicación, ya que determinara en gran medida el grado de aceptación de los usuarios, así como la facilidad y comodidad de uso de la misma.

Dado el perfil de usuario tan genérico al que va dirigido la aplicación, realizaremos un diseño simple, intuitivo y con el que ya estén familiarizados los usuarios. Los colores a utilizar tampoco serán muy llamativos de manera que no resulte demasiado extravagante a usuarios más conservadores.

En primer lugar realizaremos un boceto rápido a lápiz y papel para aclarar nuestras ideas. En esta memoria estos diseños previos se han realizado con herramientas gráficas para ofrecer un mínimo de calidad de apariencia y lectura.

A continuación definiremos un storyboard con la totalidad de las pantallas de las que dispondrá nuestra aplicación y la relación entre ellas, de manera que quede claro cómo acceder a todas las pantallas disponibles.

Finalmente, estas pantallas serán implementadas en la fase de desarrollo, bajo el marco de trabajo de interfaz gráfica elegido, en este caso con las herramientas de Android y Eclipse.

6.2.1. Pantallas

Queremos mantener la aplicación lo más simple posible, por lo que reduciremos el número de pantallas al mínimo. En el caso de que el proyecto crezca en opciones en el futuro se podrán añadir nuevas pantallas, que de momento son innecesarias.

Dado que nuestro proyecto hace uso de menús desplegables y cuadros de diálogo, consideraremos cada uno de estos como una pantalla más, de modo que veamos claramente la transición entre éstos en el storyboard y su diseño.

Pantalla 1 – Bienvenida.

Se trata de la primera pantalla que veremos nuestra aplicación, una pantalla de bienvenida en la cual deberemos introducir nuestros datos al iniciar la aplicación por primera vez.

Esta pantalla constará del logo de la aplicación y dos campos de texto para introducir el prefijo telefónico del país y el número de teléfono del usuario. Finalmente un botón para comenzar a usar la aplicación.

Figura 6.3 Pantalla 1 – Bienvenida

Pantalla 2 – Lista de conversaciones.

Una vez configurado el número de teléfono, cada vez que iniciemos la aplicación será esta la pantalla principal.

La función principal de la aplicación es el intercambio de mensajes con otros usuarios. Generalmente las conversaciones se repiten con los mismos usuarios, es por esto que pondremos las conversaciones más recientes en nuestra pantalla principal para poder acceder rápidamente a ellas.

El nombre del contacto, imagen, último mensaje y la hora de recibo del mismo será la información que mostraremos en esta pantalla.

A su vez, dispondremos de una barra de acciones superior, en la cual encontramos en el centro el texto identificativo de la pantalla y dos iconos a los lados.

El icono situado a la izquierda nos dará la posibilidad de abrir un panel de opciones deslizante, que también podremos abrir realizando un gesto desde la parte izquierda de la pantalla hacia la derecha. Esta es una característica de Android que vamos a explotar.

En segundo lugar, el icono de la derecha nos servirá para iniciar una conversación con algún contacto con el que aún no hayamos comenzado ninguna. Por esto, al pulsar este icono se nos llevará a la pantalla con la lista de contactos para elegir uno de ellos.

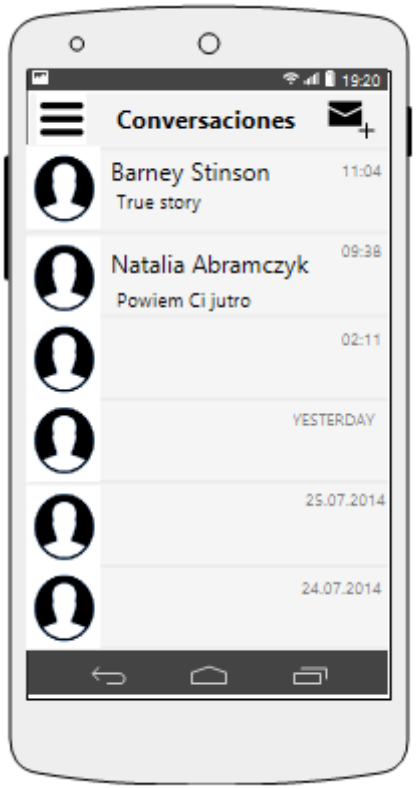


Figura 6.4 Pantalla 2 – Lista de conversaciones

Pantalla 3 - Menú de opciones.

En realidad se trata de la pantalla de conversaciones en trasfondo y un menú desplegable en el frente.

Desde aquí podremos acceder a cuatro funcionalidades: perfil, traducción, notificaciones e idioma.

Además se ha optado por incluir una imagen de encabezado para dar un efecto más visual en el despliegue del menú.

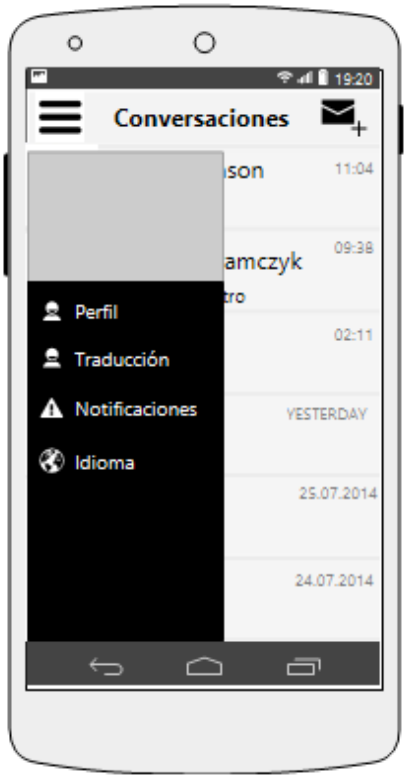


Figura 6.5 Pantalla 3 - Menú de opciones

Pantalla 4 - Perfil.

Pantalla muy simple en la que simplemente podremos visualizar nuestra imagen de contacto y nuestro número de teléfono. Se añade un botón en la parte izquierda de la barra de acciones que nos permite volver a la anterior pantalla.

Figura 6.6 Pantalla 4 - Perfil

Pantalla 5 - Traducción.

Esta pantalla en realidad es un diálogo que aparecerá al pulsar sobre la opción de traducción del menú de opciones, en él encontraremos una serie de idiomas seleccionables a los que la aplicación ofrece soporte para traducir los mensajes que se reciban.

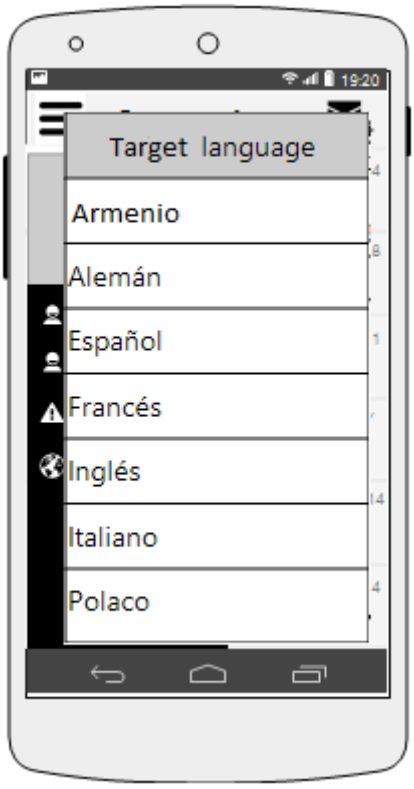


Figura 6.7 Pantalla 5 - Traducción.

Pantalla 6 - Notificaciones.

Al igual que la pantalla anterior, se trata de un diálogo. En esta ocasión nos mostrará la opción de activar o desactivar el sonido de notificaciones.

Constará pues solamente de texto informativo y un botón para realizar esta acción, además de dos botones del diálogo para guardar o cancelar los cambios.

Figura 6.8 Pantalla 6 – Notificaciones

Pantalla 7 - Idioma.

De nuevo un dialogo resultado de presionar una de las opciones del menú de opciones. En este caso se nos presentará una ventana con los idiomas en los que la aplicación está disponible. Una vez seleccionado uno de estos idiomas la aplicación se reiniciará y cambiará el idioma de la interfaz de usuario.

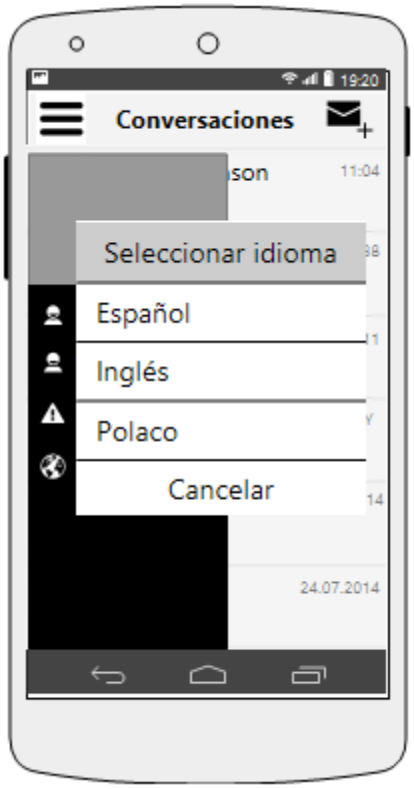


Figura 6.9 Pantalla 7 - Idioma.

Pantalla 8 - Lista de contactos.

En esta pantalla se mostrará la lista de contactos disponibles. La imagen de contacto y su nombre será la información que se presentara de cada uno de ellos.

En la barra de acciones superior, dispondremos de un botón a la izquierda para volver a la pantalla anterior y un botón a la derecha para actualizar los contactos.

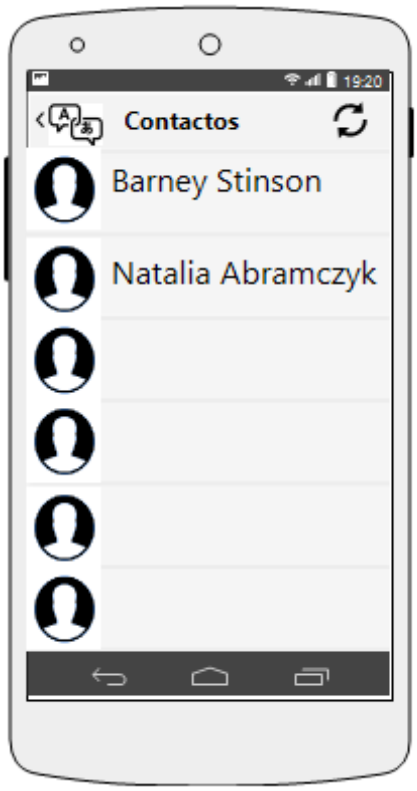


Figura 6.10 Pantalla 8 – Lista de contactos.

Pantalla 9 - Conversación.

En la pantalla de conversación podremos ver los mensajes que hemos intercambiado con el contacto con el que estamos manteniendo la conversación.

En esta lista de mensajes se mostrarán el historial de mensajes. Si el mensaje se origina en nuestro dispositivo o tenemos la traducción desactivada, solamente se mostrará el mensaje, la hora de envío/recibo y un icono mostrando el estado del mensaje, en espera o enviado.

Por otro lado, si el mensaje se ha recibido con la opción de traducción activada, se mostrará el mensaje original con un tamaño de fuente muy pequeño y, bajo él, el mensaje traducido con un tamaño de fuente medio.

En la parte inferior contaremos con un cuadro de texto para escribir nuestros mensajes y un botón de envío.

En la parte superior contaremos con un icono a la izquierda para retroceder a la anterior pantalla y con otro icono a la derecha para acceder a la pantalla de elección de idioma de origen.

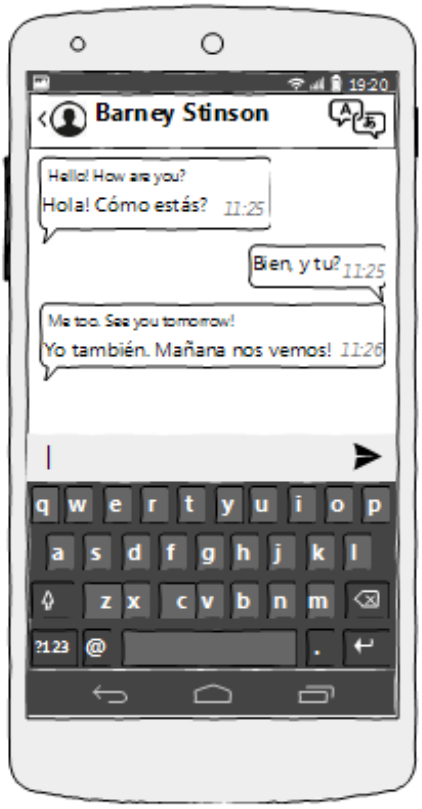


Figura 6.11 Pantalla 9 – Conversación con traducción activada.

Pantalla 10 - Idioma de origen.

Ésta última pantalla se trata de un cuadro de diálogo en el cual se nos mostrará un título y una lista de idiomas soportados por la aplicación a elegir.

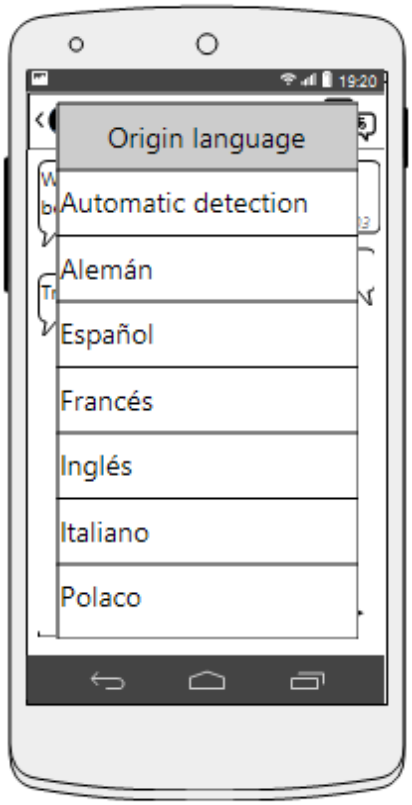


Figura 6.12 Pantalla 10 - Idioma de origen

6.2.2. Storyboard

El storyboard o guion gráfico nos ayudará a entender la estructura de la aplicación con respecto a la interfaz gráfica y el orden de interacción entre las distintas pantallas.

Como podemos ver en la figura 6.13, se ha añadido a cada una de las pantallas un círculo rojo que contiene un número que nos indica cual será la siguiente pantalla a mostrar tras pulsar el botón adyacente al círculo.

Figura 6.13 Storyboard

6.2.3. Colores

Durante la implementación final de las pantallas en Android se han tenido que determinar los detalles aún no establecidos en el diseño de pantallas tales como los iconos a usar y colores de la aplicación.

Se ha decidido usar iconos simples tales como los que se incluyen en el SDK de Android, en concreto se han usado los de la API 20 de color blanco. A estos hay que añadir algún icono editado personalmente siguiendo el mismo estilo que los mencionados.

En cuanto al color se ha decidido utilizar un estilo simple con el juego de colores azul, blanco y negro como se muestra en la figura 6.14.

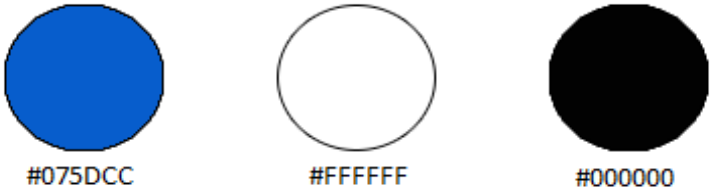


Figura 6.14 Colores principales

A continuación se muestran algunas de las pantallas reales de la aplicación final para ver el resultado conseguido.

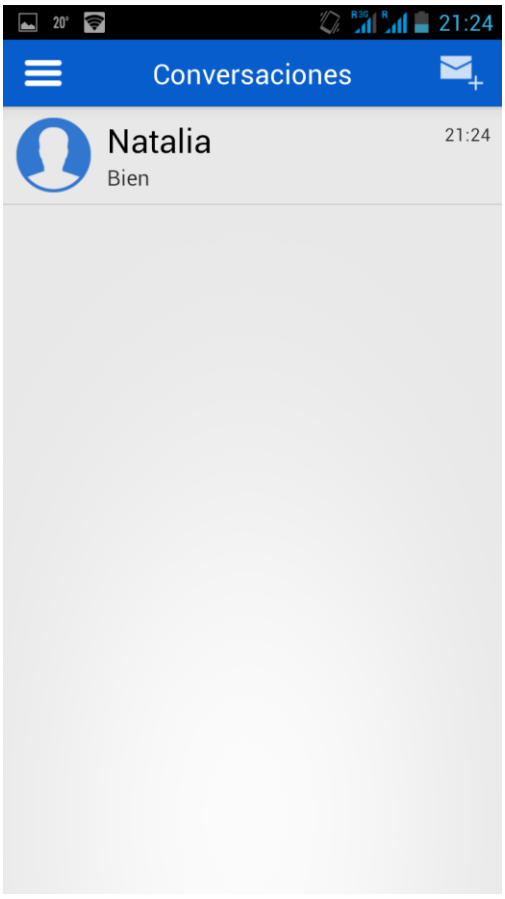
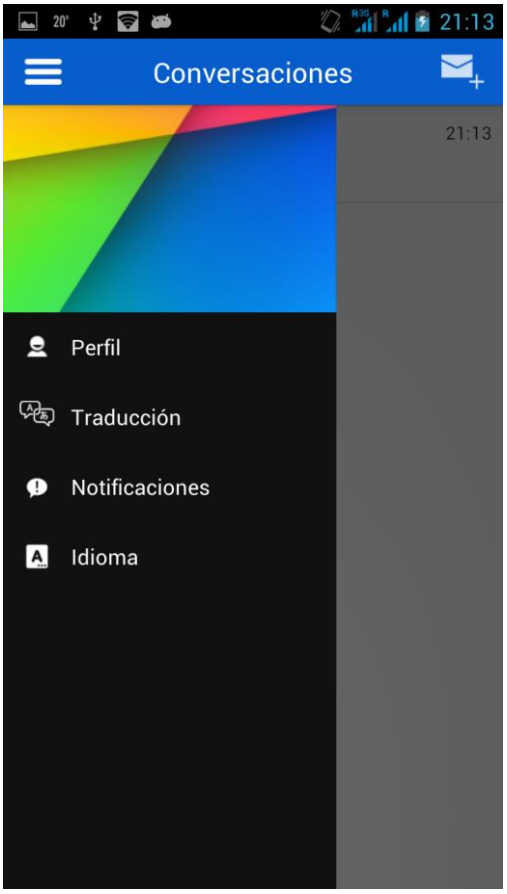


Figura 6.15 Pantalla real de la aplicación 1

Figura 6.16 Pantalla real de la aplicación 2



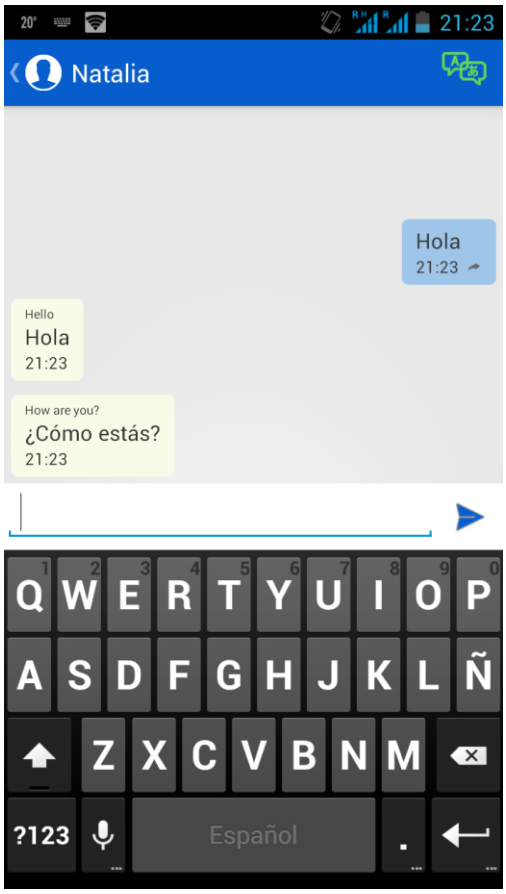


Figura 6.17 Pantalla real de la aplicación 3

Figura 6.18 Pantalla real de la aplicación 4

6.3. Diagrama de clases

El diseño del diagrama de clases definirá las diferentes clases a implementar, sus atributos y métodos. Además nos servirá para identificar las relaciones tales como herencia, composición, agregación, asociación y uso entre ellas.

6.3.1. Proyecto cliente

El proyecto cliente, que se realizará en Android, dispondrá de un gran número de clases. Es por esto que para su mejor visibilidad se mostrarán los diagramas de clases divididos por paquetes.

Esto nos impedirá ver la relación entre clases de diferentes paquetes pero contribuirá favorablemente a la facilidad de lectura en este documento.

- Paquete general

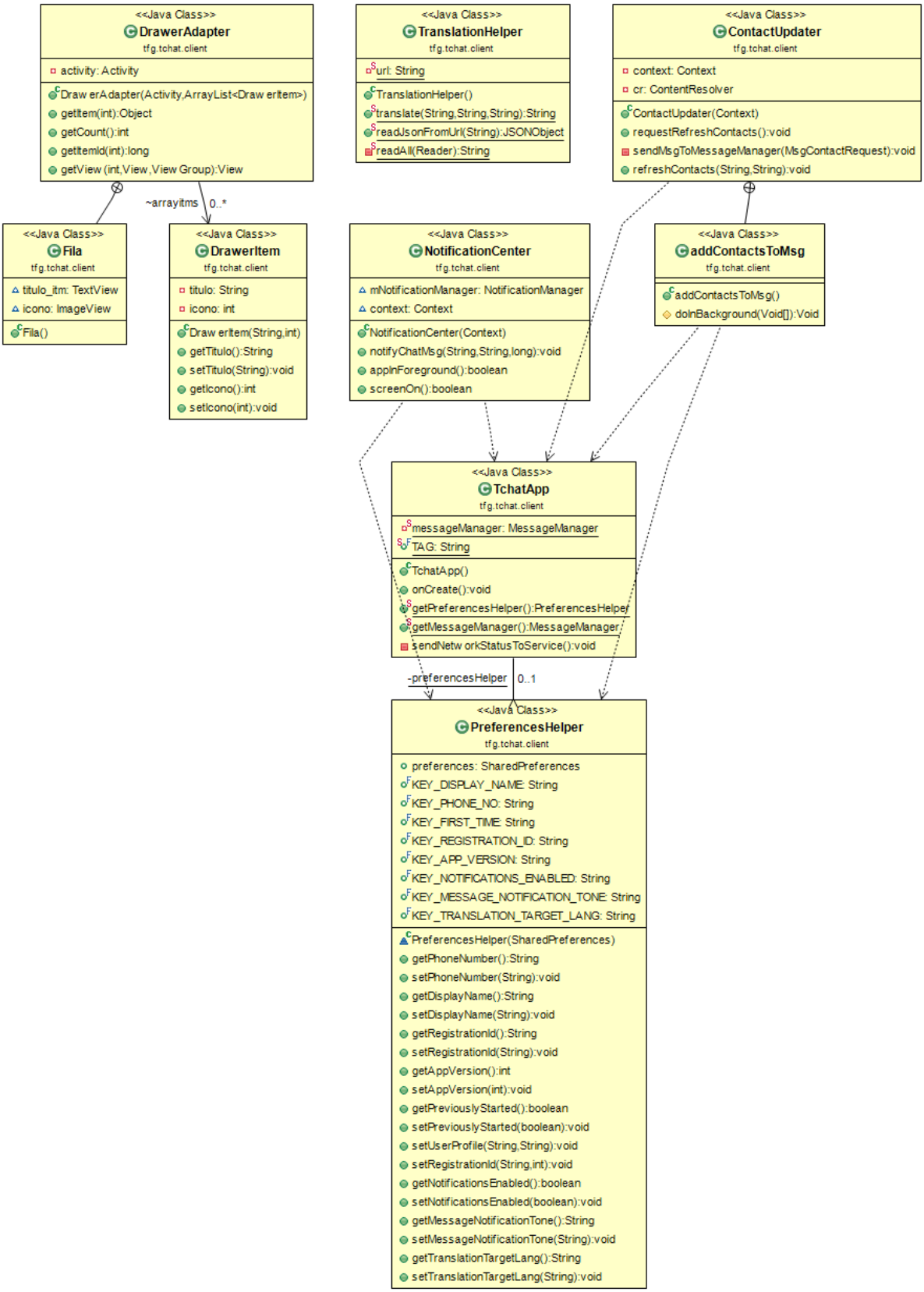


Figura 6.19 Diagrama de clases. Proyecto cliente, paquete general.

- Paquete Db

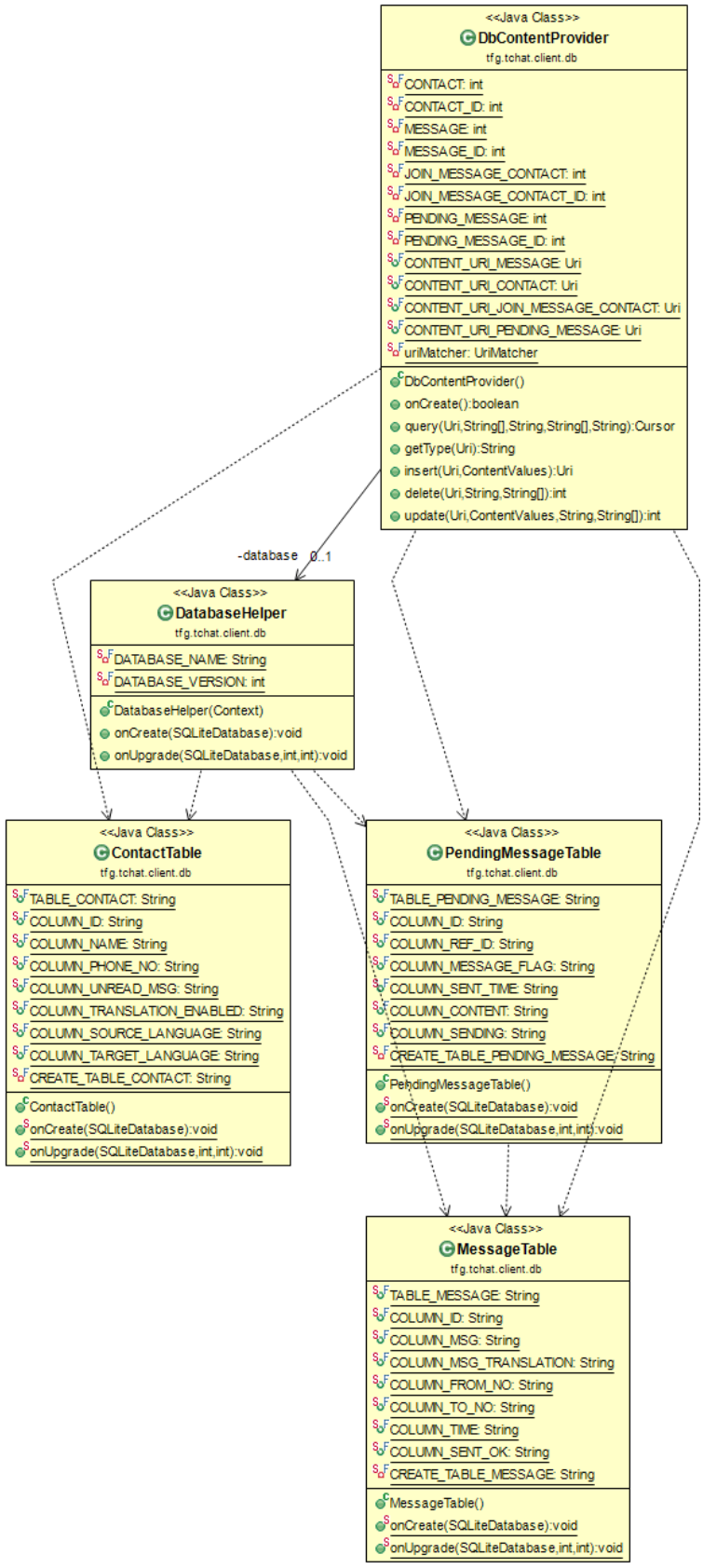


Figura 6.20 Diagrama de clases. Proyecto cliente paquete Db

- Paquete message

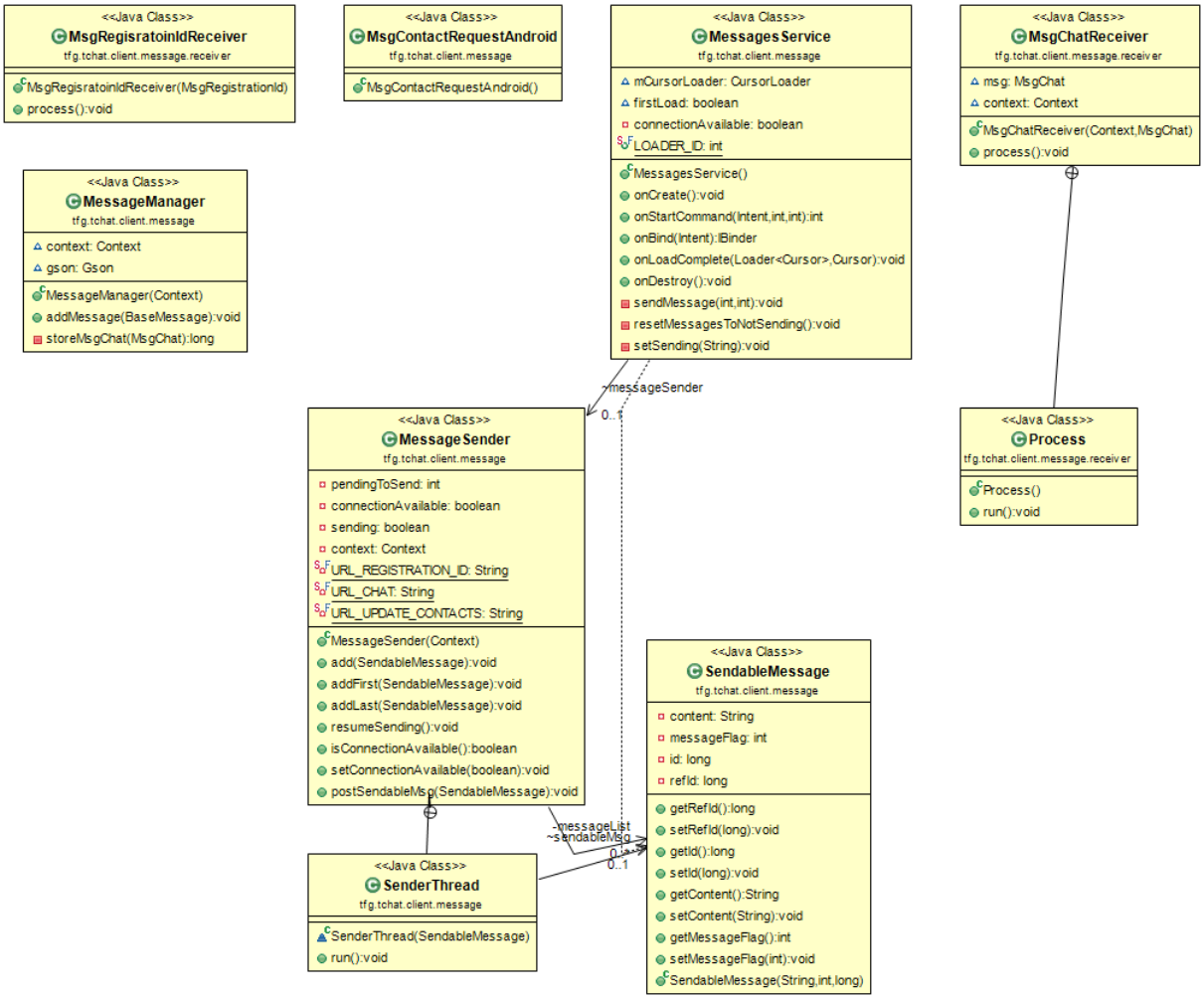


Figura 6.21 Diagrama de clases. Proyecto cliente, paquete message

- Paquete ui.activities

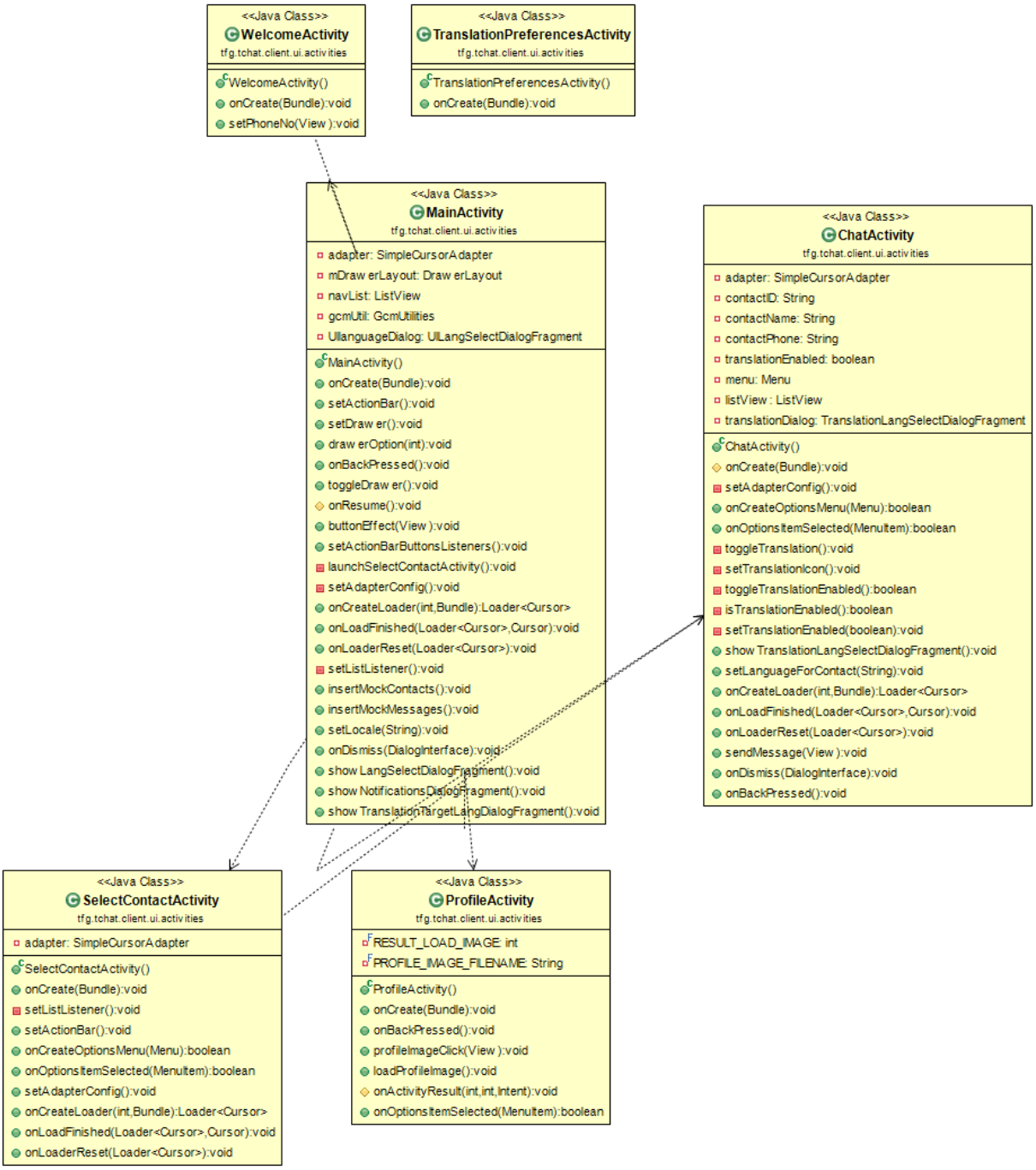


Figura 6.22 Diagrama de clases. Proyecto cliente, paquete activities

- Paquete ui.fragments

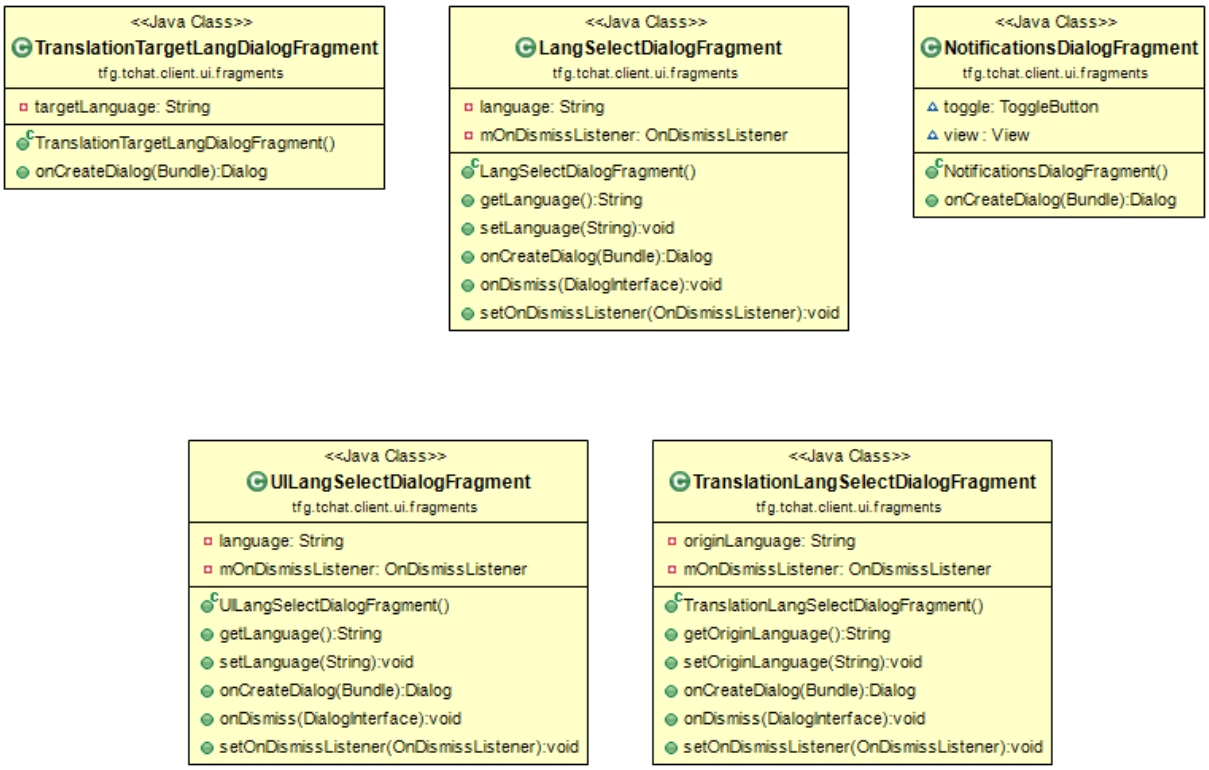


Figura 6.23 Diagrama de clases. Proyecto cliente, paquete fragments

- Paquete utilities

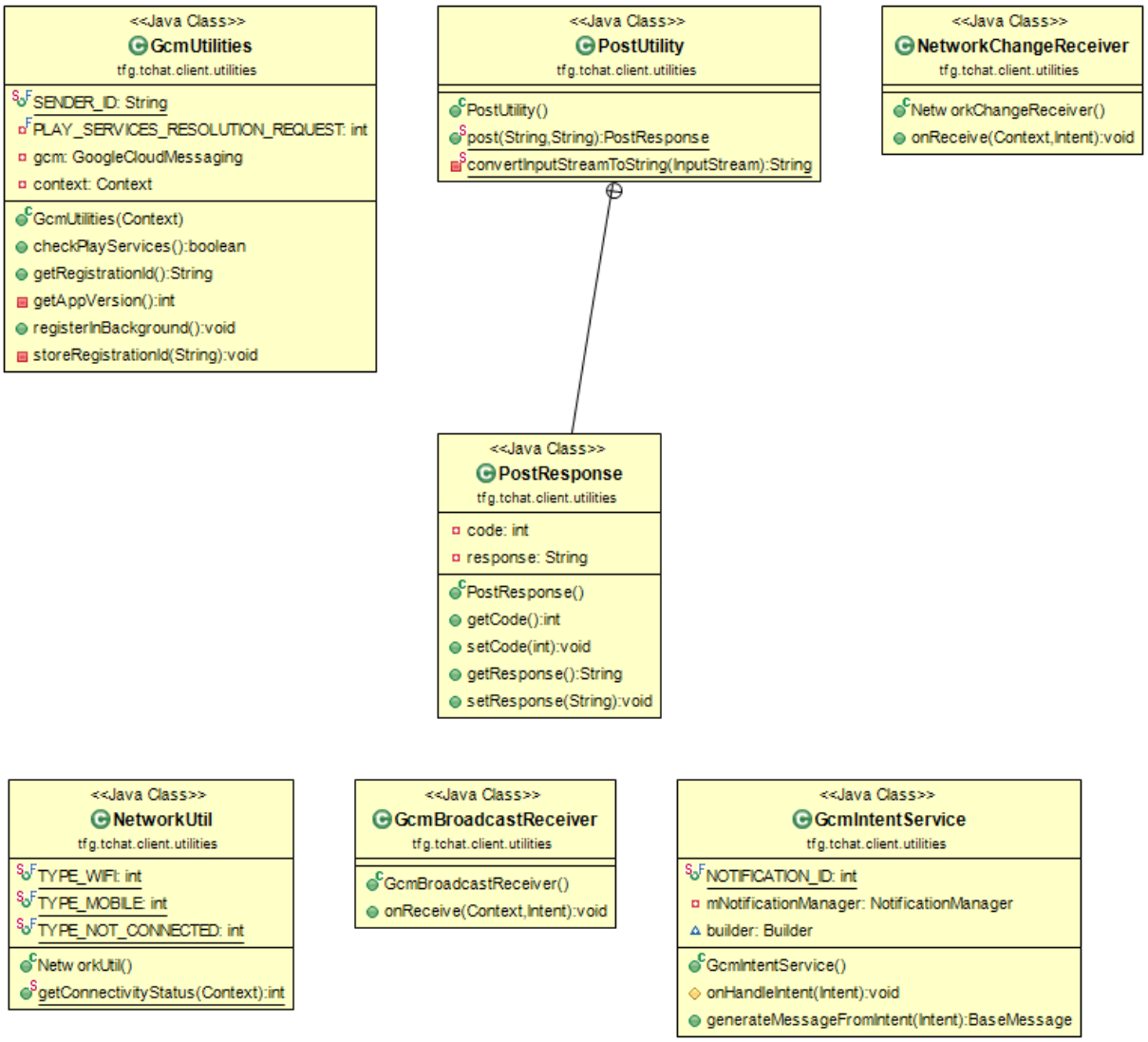


Figura 6.24 Diagrama de clases. Proyecto cliente, paquete utilities

6.3.2. Proyecto compartido

Este Proyecto al ser más reducido se mostrará en un solo diagrama.

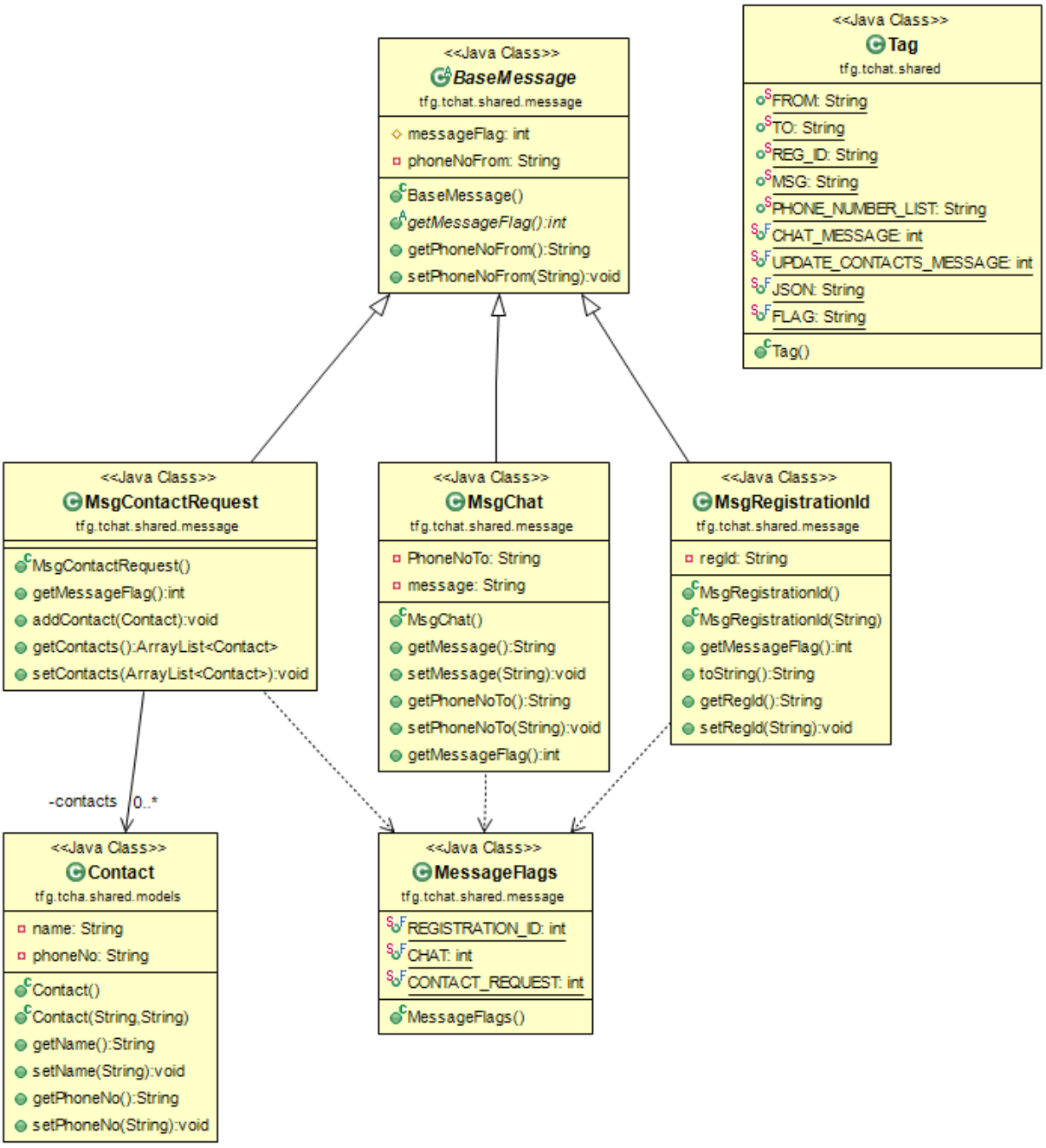


Figura 6.25 Diagrama de clases. Proyecto compartido

6.3.3. Proyecto servidor

Al igual que el proyecto compartido, el proyecto servidor tiene un tamaño reducido y podremos mostrarlo en tan sólo un diagrama.

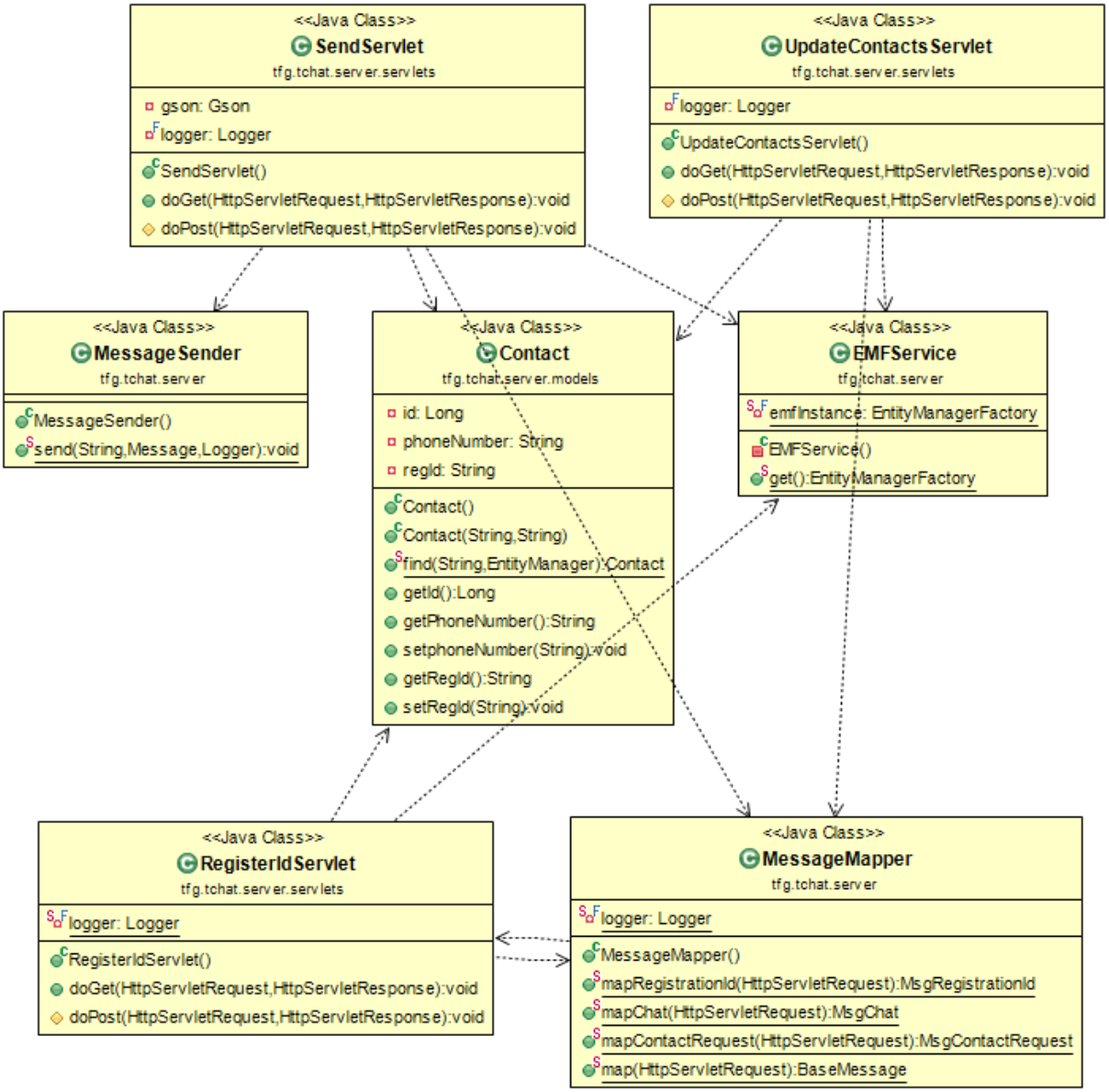


Figura 6.26 Diagrama de clases. Proyecto servidor

6.4. Diseño de datos

La fase del diseño de datos consiste en descubrir y definir la estructura de los datos que poseen los elementos de nuestro sistema.

Si el diseño de datos es bueno, el acceso a los datos de la aplicación será rápido y fácil de mantener, además podrá aceptar sin problemas las futuras mejoras de los datos que puedan ser necesarias en futuras extensiones de la aplicación

Este proceso generalmente se lleva a cabo mediante la realización de un diseño conceptual de la base de datos y posteriormente la extracción de las tablas de este modelo. No obstante es importante tener en cuenta el entorno en el que nos encontramos. Al tener una arquitectura cliente-servidor tendremos diferentes necesidades para cada proyecto. Los mecanismos de persistencia de datos en una aplicación Android son diferentes a los de un servicio web Java.

6.4.1. Persistencia en Android

En el sistema operativo Android nos encontramos con diferentes opciones para almacenar nuestros datos, estas son:

- SQLite Databases (Bases de datos SQLite)
- Shared Preferences (Preferencias compartidas)
- Internal Storage (Almacenamiento interno)
- External Storage (Almacenamiento externo)
- Network Connection (Conexión de red)

En mi caso me centraré en las 3 primeras, ya que son las 3 que utilizaré en mi aplicación.

SQLite Databases

En primer lugar es necesario identificar los datos que nuestra aplicación va a necesitar almacenar en la base de datos, estos son:

- Contactos

Cada uno de los contactos que figuran en nuestra agenda del móvil y que tengan la aplicación Tchat activa deberán ser almacenados para un rápido acceso.

- Mensajes de chat.

Cada uno de los mensajes enviados o recibidos en el chat deben ser almacenados para su posterior consulta.

- Mensajes pendientes

En condiciones de conectividad limitada o nula, es importante la característica de almacenar la información que ha de ser enviada al servicio web para su posterior envió cuando haya una conexión disponible. Estos mensajes no son solamente mensajes de chat, también se incluyen mensajes de control, peticiones para actualizar los contactos y demás mensajes que podrán surgir en el futuro de la aplicación.

Para llevar a cabo el diseño de datos en la base de datos SQLite de Android realizaré un modelo conceptual del tipo Modelo Entidad-Relación ya que es el modelo más extendido y por ende podríamos decir que es considerado un estándar.

El modelo entidad-relación nos permite representar las **entidades** relevantes de nuestra aplicación así como sus **relaciones** y **atributos**. A continuación una introducción a cada uno de estos elementos:

- **Entidad**

Representa un objeto del cual se quiere tener su información. Este objeto puede representar entes concretos o abstractos del dominio, además debe ser fácilmente diferenciable de los demás.

Este elemento lo representamos gráficamente mediante un rectángulo que contiene el nombre de la entidad, tal como se muestra en la figura 6.27.

Figura 6.27 Representación de una entidad

- **Relación**

Describe la dependencia o relación entre entidades. Una entidad puede relacionarse consigo misma o con una o más entidades.

Las relaciones son representadas mediante un rombo etiquetado en su interior con un verbo, que describe brevemente la naturaleza de la misma.

Figura 6.28 Representación de una relación

- **Atributo**

Los atributos son las características que definen una entidad. Una entidad puede contener muchas de estas características de forma que almacenen toda la información pertinente y diferencien a unas entidades de otras.

Por lo general una entidad suele disponer de uno o más atributos identificativos que son aquellos que permiten diferenciar una instancia de la entidad a otra diferente. Estos atributos se los conoce comúnmente como clave.

Los atributos se representan mediante una elipse con su nombre en el interior, en el caso de ser un atributo identificativo el nombre deberá estar subrayado.

Figura 6.29 Representación de un atributo

Por lo tanto nuestro esquema conceptual quedaría como se puede ver en la figura 6.30.

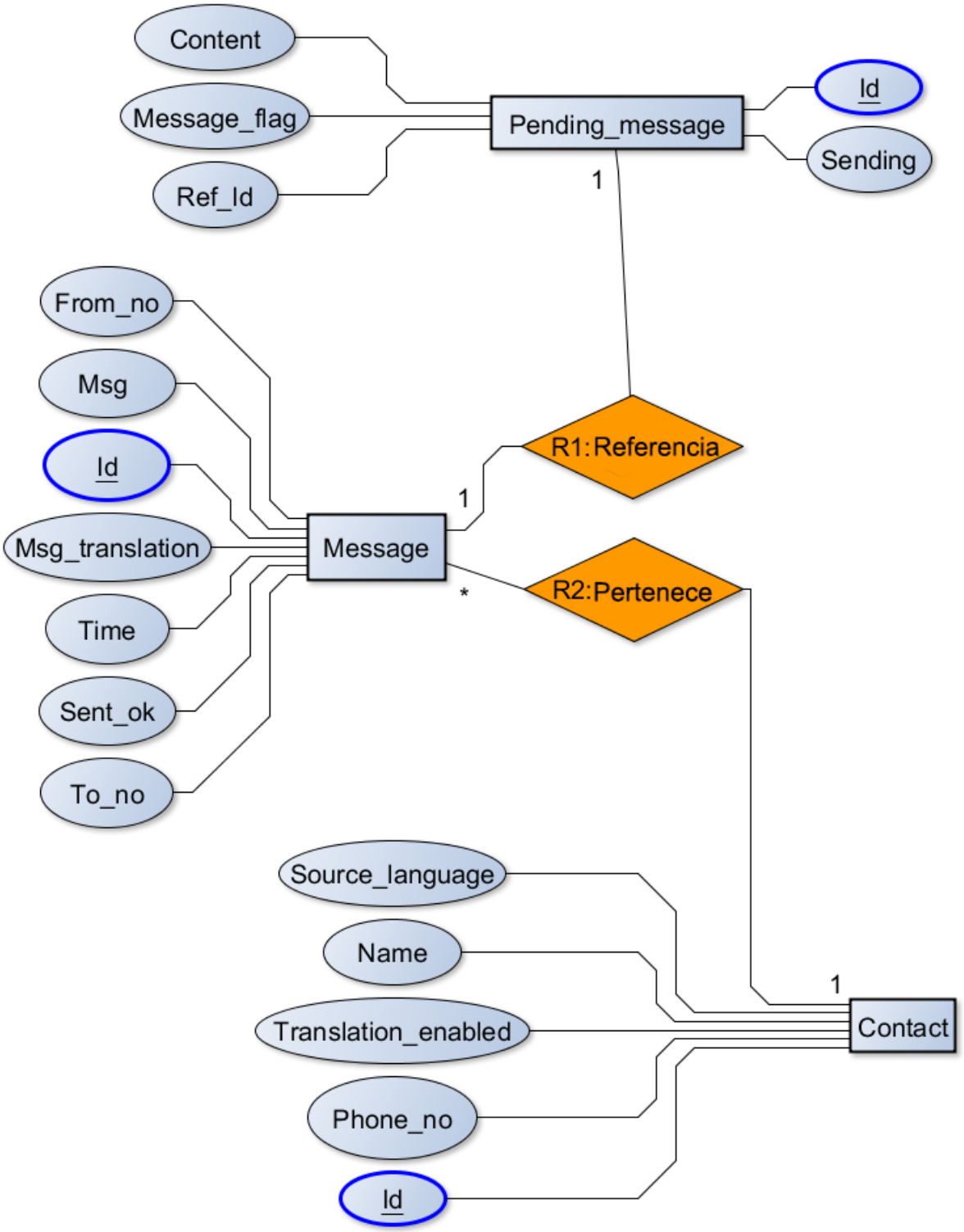


Figura 6.30 Diagrama entidad-relación de la aplicación Android

A continuación una explicación detallada de cada uno de los elementos de este diagrama:

Entidad Pending_message

Se trata de la tabla de mensajes pendientes de enviar, no solo se tratan de mensajes de chat, sino también de registro al servidor, actualización de contactos y demás mensajes de comunicación con el servidor. Esta tabla será utilizada de buffer entre la orden de envío del mensaje de usuario y el envío real cuando haya una conexión a internet disponible. Sus atributos son:

- **Id** – Entero, clave primaria y autoincremento.

  Identificador de la tabla.

- **Message_flag** – Entero no nulo.

  Bandera del tipo de mensaje almacenado.

- **Content** – Texto no nulo.

  Contenido del mensaje en formato JSON. Este es el mensaje que se enviará al servidor cuando exista conexión a internet disponible.

- **Sending** – Entero, por defecto 0.

  Valor entero que representará una variable booleana, 1 si el mensaje se está intentando enviar en estos instantes, 0 si el mensaje no está en la cola de mensajes a enviar.

Entidad Message

Es la tabla que contendrá los mensajes recibidos y enviados de las conversaciones.

- **Id** – Entero, clave primaria y autoincremento.

Identificador de la tabla.

- **Msg** – Texto no nulo.

Mensaje a enviar o recibido en una conversación.

- **Msg_translation** – Texto.

En el caso de recibir un mensaje con la traducción activa para el contacto que lo envía, el mensaje traducido se almacenará aquí.

- **From_no** – Texto.

Número de teléfono de origen del mensaje.

- **To_no** – Texto.

Número de teléfono de destino del mensaje.

- **Time** – %s.%f, Segundos desde 1970-01-01 seguido de los milisegundos, por defecto coge el momento de creación del registro.

Momento en el que se ordenó el envío del mensaje o se recibió el mensaje.

- **Sent_ok** – Entero, por defecto 0.

Valor entero que representará una variable booleana, 1 si el mensaje se envió al servidor correctamente, 0 si el mensaje aún no ha sido recibido por el servidor.

Entidad Contact

Esta entidad contiene información referente a los contactos que tenemos en la aplicación.

- **Id** – Entero, clave primaria y autoincremento.

Identificador de la tabla.

- **Phone_no** – Texto no nulo.

  Número de teléfono del contacto.

- **Name** – Texto no nulo.

  Nombre a mostrar del contacto.

- **Source_language** – Texto, por defecto ‘auto’.

  Idioma de origen de los mensajes de este contacto, este campo se tendrá en cuenta si la traducción para este contacto está activada.

- **Translation_enabled** – Entero, por defecto 0.

  Valor entero que representará una variable booleana, 1 si la traducción está activada para este contacto, 0 si está desactivada.

Shared Preferences

Esta característica de Android permite la persistencia de pares clave/valor en el dispositivo que permanecerá almacenados hasta que el usuario desinstale la aplicación.

Algunas de las ventajas del uso de las preferencias compartidas son:

Persistencia de datos hasta la desinstalación de la aplicación o hasta el borrado manual de datos de la aplicación desde los ajustes del sistema.

Eliminación de la necesidad de creación de tablas extra en la base de datos, o de guardar manualmente la información en un sistema de ficheros.

Fácil accesibilidad, tanto para lectura como escritura

Es por esto que algunos de los valores que necesitaremos constantemente en nuestra aplicación los almacenaremos de esta manera, principalmente por la facilidad de acceso y de uso.

Estos valores son:

- Phone_No (String)

Aquí se almacenará el número de teléfono introducido en la inicialización de la aplicación, o lo que es lo mismo, nuestro número de teléfono.

- First_Time (Booleano)

Por defecto este campo será verdadero, una vez iniciada la aplicación por primera vez e inicializada, este campo se fijará a falso, de modo que no se nos vuelva a pedir configurar nuestro número de teléfono de nuevo.

- Registration_Id (String)

Se trata de la identificación de registro en los servicios de GCM. Si no se dispone de ninguna identificación ésta será solicitada al inicio de la aplicación.

- App_Version (Entero)

La versión de la aplicación nos será necesaria para el uso de GCM, cada vez que la aplicación sea actualizada se solicitará una identificación de registro nueva.

- Notiifcations_Enabled (Booleano)

Este valor determina la preferencia de reproducción de sonidos de notificación. Desde el menú de opciones podremos modificar este valor para activar/desactivar el sonido de notificaciones.

- Translation_Target (String)

Idioma al que se traducirán los mensajes en las conversaciones con traducción activa. En concreto aquí se guardará el código del idioma en formato ISO 639-1.

Internal Storage

Android nos ofrece la posibilidad de guardar archivos fácilmente en el almacenamiento interno, de forma privada.

Esto es, tan solo nuestra aplicación será capaz de leer/escribir estos archivos.

En nuestro caso concreto, nuestra aplicación podría hacer uso del almacenamiento interno para el almacenamiento de las imágenes de contactos.

6.4.2. Persistencia en GAE

El mecanismo de persistencia de datos en Google App Engine (GAE) es denominado por Google Datastore. El Datastore es una base de datos NoSQL, lo cual aumenta la optimización para grandes sistemas en la nube con gran escalabilidad.

Nuestra aplicación en GAE será muy sencilla, y solo almacenará los contactos que hacen uso de nuestra aplicación, por lo que utilizaremos anotaciones JPA directamente en el código para definir la sencilla estructura de la base de datos:

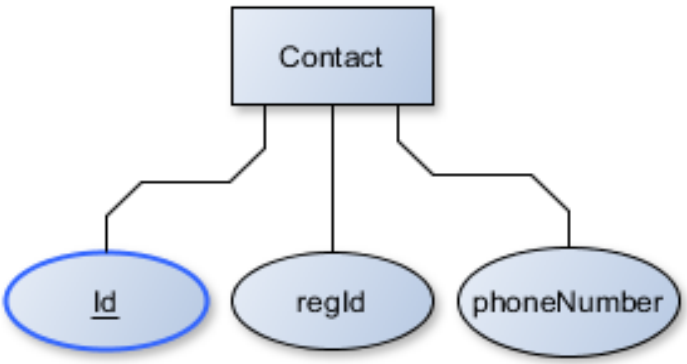


Figura 6.31 Diagrama entidad-relación de la aplicación web

Entidad Contact

Contiene información sobre los contactos que usan nuestra aplicación. Principalmente se usará para relacionar un número de teléfono a un identificador de GCM.

- **Id** – Entero, clave primaria y autoincremento.

  Identificador de la tabla.

- **regId** – String.

  Identificación del dispositivo en GCM

- **phoneNumber** – String.

  Número de teléfono del contacto.

CAPÍTULO 7

DESARROLLO

7. Desarrollo

El desarrollo del sistema se ha dividido en tres proyectos:

- Proyecto cliente
- Proyecto compartido
- Proyecto servidor

Cada uno de ellos tendrá su propósito específico en el sistema.

7.1. Proyecto cliente

Se trata de la aplicación para Android, el proyecto se ha llamado Tchat. Como ya dijimos anteriormente, se ha realizado utilizando Eclipse, en concreto la versión 4.2.1 y el SDK de Android.

La API mínima soportada será la 11 para Android 3.0 HoneyComb, la API de destino sobre la que se realizarán las pruebas será la API 19 para Android 4.4 KitKat.

A continuación describiremos los diferentes componentes que forman esta aplicación.

7.1.1. Permisos

Al realizar una aplicación de Android es necesario declarar una serie de permisos que son necesarios para acceder a ciertas características del sistema tales como recursos de red, contactos del teléfono, envío de SMS etc.

Los permisos que hemos utilizado para esta aplicación son los siguientes:

- tfg.tchat.client.permission.C2D_MESSAGE
- tfg.tchat.client.C2D_MESSAGE
- com.google.android.c2dm.permission.RECEIVE

Estos tres permisos son necesarios para la recepción de mensajes de GCM.

- android.permission.INTERNET

  Permite hacer uso de la conexión a internet.

- android.permission.WAKE_LOCK

  Nos permite usar el teléfono aunque este bloqueado para que cuando recibamos un mensaje éste pueda ser procesado..

- android.permission.GET_TASKS

  Este permiso es necesario para conseguir una lista de las aplicaciones que están funcionando en el dispositivo. Esto será necesario para determinar si nuestra aplicación se está ejecutando y si está en primer plano, para mostrar o no las notificaciones.

- android.permission.READ_CONTACTS

  Permite obtener la lista de contactos almacenados en el teléfono y sus detalles. Es necesario para comprobar qué contactos del teléfono usan nuestra aplicación y añadirlos a la misma.

- android.permission.ACCESS_NETWORK_STATE

Permite obtener el estado de la red. Esto es necesario para comprobar cuando estamos conectados a internet para poder enviar los mensajes que tengamos en cola.

7.1.2. Aplicación, Actividades y fragmentos

El objeto Application(Aplicación) mantiene el estado global de la aplicación y su contexto. Tras su inicialización se lanza la actividad lanzadera.

Las actividades en Android serán la base de cada pantalla a mostrar en el dispositivo, en ellas podremos mostrar un diseño gráfico (layout) o fragmentos.

TchatApp

Es el objeto de tipo Application y será el primero en iniciarse. En esta clase podemos inicializar variables u ofrecer métodos estáticos a recursos necesarios para toda la aplicación, como por ejemplo el gestor de preferencias o el gestor de mensajes desarrollados.

WelcomeActivity

Esta será la actividad para la pantalla principal de la aplicación. En ella se muestra el mensaje de bienvenida y se hace la petición del número de teléfono del usuario. Mientras que la aplicación no esté configurada ésta será la pantalla principal que se muestre siempre. Una vez introducido el número de teléfono esta pantalla no se volverá a mostrar nunca más.

MainActivity

Actividad principal donde se mostrará la lista de conversaciones disponibles, el menú y el botón para iniciar nueva conversación.

Es importante el uso de un adaptador personalizado para la lista de conversaciones con una consulta al proveedor de contenido adecuado.

Además se hará uso de la interfaz LoaderManager que nos ayudará a ofrecer una interfaz más fluida, ya que realizará la carga de los elementos en segundo plano sin bloquear la interfaz de usuario y mostrará los cambios tan pronto como se modifiquen los datos en proveedor de contenidos.

Adicionalmente, esta actividad permitirá el lanzamiento de los cuadros de diálogo para modificar las preferencias del usuario.

SelectContactActivity

En esta actividad se muestran los contactos disponibles en nuestra aplicación y un botón para actualizar esta lista.

Al igual que en MainActivity, se hará uso de la interfaz LoaderManager para la carga de contactos.

ChatActivity

Desde esta actividad se nos permitirá el envío de mensajes a otros contactos. Se accede mediante el click en una conversación en MainActivity o un contacto en SelectContactActivity, que enviarán la información del contacto seleccionado a esta actividad.

De manera que la ventana será personalizada para cada contacto, mostrando su nombre, el historial de conversación que tenemos con él y las opciones de traducción pertinentes.

Una vez haremos uso de la interfaz LoaderManager, la conversación será una lista de elementos definidos en los recursos de la aplicación, con una serie de campos y un fondo dibujable (drawable). A la hora de cargar los mensajes se irá aplicando el formato, el icono (enviado o en espera), el lado en que mostrarse, el color etc.

Desde aquí se podrá acceder al cuadro de diálogo para activar y desactivar la traducción con el contacto y seleccionar el idioma de origen.

ProfileActivity

Actividad muy sencilla en la cual se mostrará el número de teléfono registrado en la aplicación y la imagen de perfil.

Fragmentos

Todos los fragmentos utilizados en la aplicación extienden la clase DialogFragment, ya que todos son cuadros de diálogo que mostramos para cambiar algunas opciones.

En concreto son

- NotificationsDialogFragment

Muestra la opción para activar o desactivar el sonido de las notificaciones.

- TranslationLangSelectDialogFragment

En este dialogo podemos elegir el idioma en el que nos escribe un contacto para afinar el servicio de traducción.

- TranslationTargetLangDialogFragment

Mostrará la lista de idiomas disponibles a los que podemos traducir los mensajes recibidos.

- UILangSelectDialogFragment

Mostrará la lista de idiomas disponibles para la interfaz de usuario.

7.1.3. Proveedor de contenido

Los proveedores de contenido en Android gestionan el acceso a datos estructurados como por ejemplo la base de datos que hemos diseñado para esta aplicación.

Para su uso debemos declararlo en el manifiesto de la aplicación, también indicaremos que no queremos exportarlo y que solo será accesible desde nuestra aplicación, pues una de las características de los proveedores de contenido es que se pueden compartir para el libre acceso de otras aplicaciones, pero este no es el caso.

Una vez declarado, será necesario establecer una serie de URIs (identificador de recursos uniforme) para acceder a diferentes tablas o conjuntos de tablas que utilicemos comúnmente.

Finalmente es necesario definir los métodos insert, delete, query y update para cada uno de estos URI.

El desarrollo de una aplicación con proveedores de contenido puede resultar complicado al principio dadas las diferencias con los mecanismos de acceso a bases de datos tradicionales, pero una vez terminada esta parte, el acceso de datos desde otras clases, la carga en listas o el refresco automático de datos resulta mucho más cómodo, ya que los elementos de Android y su SDK están altamente integrados con éstos.

7.1.4. Servicios

Un servicio es, a groso modo, un componente que se ejecuta en segundo plano, realiza tareas y no interactúa con el usuario.

La característica más importante para nosotros es que un servicio se puede estar ejecutando continuamente en el dispositivo móvil, incluso aunque la aplicación no se esté ejecutando.

En nuestro proyecto contaremos con dos servicios:

GcmIntentService

Una vez recibido un mensaje por GCM en el BroadcastReceiver, se llamará a este servicio para que se ocupe del mensaje.

Este servicio recibirá el contenido del mensaje en formato JSON y lo transformará a la clase adecuada para después ser procesado. En el caso de un mensaje de conversación, comprobará el lenguaje al que es necesario traducirlo, realizará la traducción y lo almacenará en el proveedor de contenido. Finalmente comprobará si es necesario mostrar una notificación, en caso afirmativo delega el trabajo a la clase NotificationCenter donde se harán las comprobaciones pertinentes para las diferentes opciones de notificación, como si está activo o no el sonido de las mismas.

MessagesService

La ejecución de este servicio será permanente en segundo plano.

Cada vez que se envíe un mensaje desde la aplicación, el servicio recibirá los cambios en la tabla de la base de datos con mensajes pendientes y lo marcará como mensaje que se está enviando, mientras intenta enviarlo al servidor. Una vez obtenida la confirmación de recibo del servidor, el servicio elimina de la tabla de mensajes pendientes de enviar el mensaje enviado.

El servicio recibirá actualizaciones del estado de conexión, permitiendo el envío de mensajes solo cuando tengamos una conexión disponible, y almacenándolos en una cola cuando no tengamos conexión a internet.

Para disminuir las operaciones de lectura y escritura de disco, que son las más costosas, el servicio guarda en memoria la cola de mensajes pendientes de enviar, de manera que si disponemos de un instante de conectividad, el servicio comenzará de forma automática el envío rápido de los mensajes ya cargados.

7.1.5. BroadcastReceivers

Los últimos componentes que nos quedan por registrar son los BroadcastReceivers. Básicamente son unas clases encargadas de la recepción de ciertos intents.

En el manifiesto de la aplicación indicaremos cual es la clase que usaremos par a esta recepción, que debe extender a la clase BroadcastReceiver, y los filtros que va a aceptar para la recepción.

En nuestra aplicación contamos con dos BroadcastReceivers:

GcmBroadcastReceiver

Este BroadcastReceiver se encargará de la recepción de mensajes relacionados con el servicio GCM. En concreto los filtros que hemos indicado en el

manifiesto de Android son com.google.android.c2dm.intent.RECEIVE y com.google.android.c2dm.intent.REGISTRATION, los cuales nos permiten la recepción de mensajes y de la solicitud de registro de GCM.

La funcionalidad de este recepto es muy sencilla, se limita a obtener el mensaje e iniciar el servicio GCMIntentService para que lo procese.

NetworkChangeReceiver

Los filtros registrados en el manifiesto de Android son android.net.conn.CONNECTIVITY_CHANGE y android.net.wifi.WIFI_STATE_CHANGED, los cuales no permiten obtener los cambios de conectividad, tales como si hemos cambiado a red Wi-fi o a Datos o no tenemos ninguna conectividad.

Una vez obtenido el estado de la red, este receptor enviará un intent al servicio MessagesService indicando los cambios para que el servicio lleve a cabo las operaciones necesarias como reanudar o detener el envío de mensajes.

7.2. Proyecto compartido

Se ha optado por la realización de un proyecto compartido, llamado Tchat-Shared, ya que tanto el cliente como el servidor se han realizado en java y existen múltiples elementos comunes. Este es un proyecto de Java estándar.

La idea inicial de realizar el proyecto compartido surgió a la hora del envío de mensajes por parte del cliente y de recepción por parte del servidor. Existen algunas etiquetas (Tags) que se han usado en ambos proyectos y deben ser iguales en los dos. Es por esto que resulta más sencillo la definición de estas etiquetas en un proyecto aparte y la inclusión de este recurso en los otros dos proyectos.

De esta manera, siempre y cuando ambos proyectos tengan incluida la misma versión de este proyecto compartido tendrán las mismas etiquetas.

Así mismo, se ha implementado la base de los mensajes en este proyecto, ya que tanto en el cliente como en el servidor deben ser los mismos y es necesario realizar un mapeado del objeto a JSON y de JSON al objeto.

Algunas de las clases de mensajes implementadas en este proyecto son BaseMessage, MsgChat, MsgContactRequest y MsgRegistrationId.

7.3. Proyecto servidor

Finalmente necesitamos una aplicación web como ya indicamos anteriormente. Este será un Google Web Application Project que hemos llamado Tchat-WebService. Para la creación de un proyecto de este tipo es necesario instalar la extensión para eclipse Google Plugin for Eclipse. Una vez instalada, además de la posibilidad de crear este tipo de proyectos, tendremos la opción de desplegar la aplicación web a Google App Engine directamente desde Eclipse con tán solo unos clicks.

La aplicación web será muy sencilla. Contará en primer lugar con un modelo para la base de datos “Contacts” como ya indicamos en el apartado de diseño de datos, donde declararemos las variables necesarias y las anotaciones para la persistencia con JPA. Más tarde haremos uso de los datos en GAE mediante el EntityManager declarado en la clase EMFService.

Aparte de este modelo, dispondremos de tres Servlets que responderán a peticiones HTTPPost y de los que hará uso nuestra aplicación:

RegisterIdServlet

Este servlet recibirá un mensaje de tipo MsgRegistrationId, una vez mapeado el JSON al objeto, se comprobará si ya existe este contacto en la base de datos. En caso de no existir se creará la nueva entrada del contacto con el identificador de registro de GCM.

Por último se devolverá un valor para confirmar que la petición ha sido recibida y procesada.

SendServlet

Este es el servlet que se usará con más frecuencia, ya que será el encargado de recibir las peticiones de envío de mensajes.

Una vez mapeado el contenido JSON a un objeto MsgChat, se buscará el identificador GCM relacionado al número de teléfono al que se desea enviar el mensaje, y se enviará el contenido a ese identificador mediante la librería de GCM que nos proporciona Google.

UpdateContactsServlet

Este servlet recibirá un mensaje de tipo MsgContactRequest el cual contendrá la lista de contactos del dispositivo del usuario. Se comprobarán los números de teléfono para ver si ya existen en nuestro servidor y en caso afirmativo se marcará como contacto disponible.

Tras comprobar todos los contactos se devuelve una respuesta con los contactos encontrados para que el cliente pueda procesarla y agregarlos a la aplicación.

7.4. Casos de test y resultados

Una vez terminada la implementación del Proyecto es hora de realizar las pruebas para comprobar si el Sistema funciona correctamente.

Se ha decidido realizar una serie de pruebas utilizando dos dispositivos Android reales y bajo diferentes circunstancias como situaciones de conectividad limitada en la carretera.

TEST 1	
Requisito	Configurar la aplicación por primera vez.
Acción	El usuario introduce el prefijo telefónico de su país y su número de teléfono, después pulsa en hecho.
Salida esperada 1	La aplicación muestra la pantalla principal.

Figura 7.1 Test 1

TEST 2	
Requisito	Mostrar perfil.
Acción	El usuario pulsa el icono de opciones, situado en la parte izquierda de la barra de acciones de la pantalla principal.
Salida esperada 1	El sistema muestra el menú principal a la izquierda de la pantalla.
Acción	El usuario pulsa el elemento de la lista llamado "Perfil".
Salida esperada 2	El sistema muestra la pantalla de perfil.

Figura 7.2 Test 2

TEST 3	
Requisito	Mostrar lista de contactos.
Acción	El usuario pulsa el icono de iniciar nueva conversación, situado en la parte derecha de la barra de acciones de la pantalla principal.
Salida esperada 1	El sistema muestra la lista con los contactos disponibles.

Figura 7.3 Test 3

TEST 4	
Requisito	Actualizar contactos.
Acción	El usuario pulsa el icono de actualizar, situado en la parte derecha de la barra de acciones de la pantalla de contactos.
Salida esperada 1	El sistema muestra un mensaje indicando que la actualización se está llevando a cabo, tras unos segundos la lista de contactos es actualizada.

Figura 7.4 Test 4

TEST 5	
Requisito	El usuario crea una nueva conversación con un contacto.
Acción	El usuario pulsa sobre un contacto en la lista de la pantalla de contactos.
Salida esperada 1	El sistema muestra la pantalla de conversación.
Acción	El usuario escribe un mensaje en la caja para editar texto situada en la parte inferior de la pantalla y pulsa el botón enviar.
Salida esperada 2	El sistema borra el mensaje de la caja de texto, muestra el mensaje en la lista de mensajes de la ventana de chat.
Acción	El usuario pulsa dos veces el botón atrás, la conversación.
Salida esperada 3	El sistema muestra primero la pantalla de contactos y después la pantalla de conversaciones, donde debe estar la nueva conversación.

Figura 7.5 Test 5

TEST 6	
Requisito	Recibir notificación de nuevo mensaje con el dispositivo bloqueado.
Acción	Un usuario externo envía un mensaje al usuario.
Salida esperada 1	El dispositivo emite el sonido de notificaciones
Acción	El usuario desbloquea el dispositivo.
Salida esperada 2	El sistema Android muestra la notificación en la barra de notificaciones.

Figura 7.6 Test 6

TEST 7	
Requisito	Pulsar sobre la notificación y que se abra la app.
Acción	El usuario extiende la barra de notificaciones de Android y pulsa sobre la notificación de Tchat.
Salida esperada 1	El sistema muestra la conversación con el contacto el cual ha enviado el mensaje.

Figura 7.7 Test 7

TEST 8	
Requisito	Cambiar idioma de origen.
Acción	El usuario pulsa el icono de traducción, situado en la parte derecha de la barra de acciones de una pantalla de conversación.
Salida esperada 1	El sistema muestra una lista de idiomas para elegir.
Acción	El usuario selecciona un idioma.
Salida esperada 2	El sistema quita la lista de idiomas y muestra el icono de traducción activado.
Acción	El usuario recibe un mensaje del contacto en el idioma indicado.
Salida esperada 3	El sistema añade el mensaje a la lista de mensajes de la conversación, el mensaje aparece en el idioma original y traducido al idioma de destino del usuario.

Figura 7.8 Test 8

TEST 9	
Requisito	Cambiar idioma de destino.
Acción	El usuario pulsa el icono de opciones, situado en la parte izquierda de la barra de acciones de la pantalla principal.
Salida esperada 1	El sistema muestra el menú principal a la izquierda de la pantalla.
Acción	El usuario pulsa el elemento de la lista llamado "Traducción".
Salida esperada 2	El sistema muestra una lista de idiomas.
Acción	El usuario selecciona un idioma.
Salida esperada 3	El sistema quita la lista de idiomas de la pantalla.
Acción	El usuario recibe un mensaje de un contacto con el que tenga la traducción activada y abre esta conversación.
Salida esperada 4	El sistema muestra el mensaje en el idioma de origen y traducido al nuevo idioma de destino.

Figura 7.9 Test 9

TEST 10	
Requisito	Desactivar sonido a las notificaciones.
Acción	El usuario pulsa el icono de opciones, situado en la parte izquierda de la barra de acciones de la pantalla principal.
Salida esperada 1	El sistema muestra el menú principal a la izquierda de la pantalla.
Acción	El usuario pulsa el elemento de la lista llamado "Notificaciones".
Salida esperada 2	El sistema muestra una ventana con las opciones de notificaciones.
Acción	El usuario pulsa sobre el interruptor de sonido de notificaciones.
Salida esperada 3	El sistema marca como "No" el sonido de notificaciones.
Acción	El usuario recibe un mensaje con el dispositivo bloqueado y sonido activado.
Salida esperada 4	El sistema muestra la notificación en pantalla cuando ésta se desbloquee, no reproduce ningún sonido de notificación.

Figura 7.10 Test 10

TEST 11	
Requisito	Cambiar idioma de la app.
Acción	El usuario pulsa el icono de opciones, situado en la parte izquierda de la barra de acciones de la pantalla principal.
Salida esperada 1	El sistema muestra el menú principal a la izquierda de la pantalla.
Acción	El usuario pulsa el elemento de la lista llamado "Idioma".
Salida esperada 2	El sistema muestra una ventana con la lista de idiomas.
Acción	El usuario pulsa sobre un idioma de la lista.
Salida esperada 3	El sistema reinicia la aplicación y muestra el texto de la misma en el idioma que se ha seleccionado.

Figura 7.11 Test 11

TEST	RESULTADO
Test 1	
Salida esperada 1	OK
Test 2	
Salida esperada 1	OK
Salida esperada 2	OK
Test 3	
Salida esperada 1	OK
Test 4	
Salida esperada 1	OK
Test 5	
Salida esperada 1	OK
Salida esperada 2	OK
Salida esperada 3	OK
Test 6	
Salida esperada 1	OK
Salida esperada 2	OK
Test 7	
Salida esperada 1	OK
Test 8	
Salida esperada 1	OK
Salida esperada 2	OK
Salida esperada 3	OK
Test 9	
Salida esperada 1	OK
Salida esperada 2	OK
Salida esperada 3	OK
Salida esperada 4	OK
Test 10	

Salida esperada 1	OK
Salida esperada 2	OK
Salida esperada 3	OK
Salida esperada 4	OK
Test 11	
Salida esperada 1	OK
Salida esperada 2	OK
Salida esperada 3	OK

Figura 7.12 Resultados de tests

CAPÍTULO 8

CONCLUSIONES Y TRABAJOS FUTUROS

8. Conclusiones y trabajos futuros

8.1. Conclusiones

La elección de realizar este trabajo vino dada, en primer lugar, por la necesidad de satisfacer la falta de una aplicación de este tipo. Viviendo en un país extranjero, han sido múltiples las ocasiones en las que he echado en falta un sistema de traducción incorporado en las aplicaciones de mensajería instantánea que utilizo habitualmente.

Se trata de un proyecto que a primera vista pudo parecer relativamente sencillo, pues todos estamos acostumbrados a chatear día a día y nos parece algo básico en nuestra vida cotidiana. Sin embargo, las tareas de investigación han ocupado gran parte del tiempo disponible dada la falta de conocimiento inicial sobre el funcionamiento y arquitectura de este tipo de sistemas para ofrecer un alcance instantáneo, móvil y fácilmente escalable para el soporte de múltiples usuarios.

El hecho de que se trate de un sistema móvil, abierto a múltiples factores como la no disponibilidad del dispositivo, pérdidas de conexión o ejecución de otras tareas, ha complicado en gran medida el desarrollo del proyecto. Teniendo que llevar a cabo mecanismos complejos con los cuales asegurar el envío y recibo de mensajes.

Durante su realización, se han barajado distintos protocolos de mensajería instantánea, aún en fases poco maduras para el uso en dispositivos móviles. Finalmente la elección de GCM nos ha permitido cumplir con los objetivos propuestos al principio de este proyecto de una forma simple sin la necesidad de entrar en el desarrollo y ampliación de otro de estos protocolos.

También ha sido necesario el estudio de tecnologías como Android, JPA o el uso de Google Cloud Engine, entre otras, lo cual me ha ofrecido una experiencia muy satisfactoria ya que me ha ayudado a expandir mis conocimientos pudiendo aplicarlos en el desarrollo de un proyecto real.

Finalmente, me complace afirmar que el proyecto ha sido un éxito, tanto en la definición de los objetivos, fases de análisis, implementación y en los resultados

obtenidos en los test finales. Más allá de estos test se encuentra el uso casi diario que le estoy dando en el país donde resido actualmente.

8.2. Trabajos futuros

Como ya hemos dicho en varias ocasiones, las aplicaciones de mensajería instantánea para dispositivos móviles actuales nos brindan una gran cantidad de funcionalidades adicionales al envío de mensajes,

Debido a las limitaciones de tiempo y medios de la realización de este trabajo de fin de grado, estas funcionalidades han sido reducidas a las más básicas, asi como el protocolo de comunicación utilizado.

Por esta razón este apartado de trabajos futuros podría ser tan extenso como quisiéramos, añadiendo todo tipo de opciones hasta un gran nivel de detalle.

Las principales mejoras que se pueden realizar sobre la aplicación realizada son:

- Modificación del protocolo de mensajería instantánea. Existen protocolos de mensajería instantánea que optimizan el consumo de recursos el apartado de servidor y nos ofrecen características como monitorización de estado. La adaptación del proyecto a uno de ellos sería un gran paso adelante para utilizar este sistema en un entorno de producción.
- Envío de emoticonos. Cualquier aplicación de este tipo incluye soporte a diferentes emoticonos para favorecer una conversación más llamativa y visual.
- Envío de imágenes y videos. Compartir imágenes y videos es una funcionalidad básica que ha cobrado mucha importancia desde la llegada de los sistemas de mensajería instantánea a los dispositivos. Permitiendo comentar las imágenes a la vez que se comparten en tiempo real.
- Modificación de imagen de perfil. Compartir una imagen de perfil es otra de las funcionalidades que con el tiempo se ha convertido en algo común para estos clientes.
- Visibilidad del estado de conexión de la otra persona. Es interesante ver si la persona con la que estás hablando se conectó recientemente o por el

contrario lleva mucho tiempo sin conectarse, y por ende hay baja probabilidad de recibir una respuesta temprana. Indicar si el contacto está escribiendo un mensaje sería también interesante.

- Confirmación de la llegada del mensaje al móvil de destino. Imprescindible característica para asegurarnos de que nuestro mensaje llego al contacto deseado y de que existe la posibilidad de que este se ha leído.
- Un largo etcétera como compartición de archivos, de localización, múltiples opciones de notificaciones, de privacidad, etc.

CAPÍTULO 9

ANEXOS

9. Anexos

9.1. Anexo I: Manual de usuario

Este manual de usuario describe las funciones principales de la aplicación Tchat.

Configuración

Al iniciar por primera vez la aplicación se mostrará la pantalla de bienvenida.

Deberá introducir el prefijo telefónico del país de procedencia de la tarjeta sim, sin ceros ni signos, en la primera casilla disponible. Para España este código es 34.

En la segunda casilla disponible será necesario introducir el número de teléfono, que en España consta de 9 dígitos.

Figura 9.1 Pantalla de inicio vacía

Figura 9.2 Pantalla de inicio con datos introducidos

Una vez introducidos los datos requeridos tan solo será necesario pulsar el botón ¡Hecho! para avanzar a la siguiente pantalla y comenzar a usar la aplicación.

La siguiente pantalla será la pantalla principal de la aplicación, en la que se muestran las conversaciones.

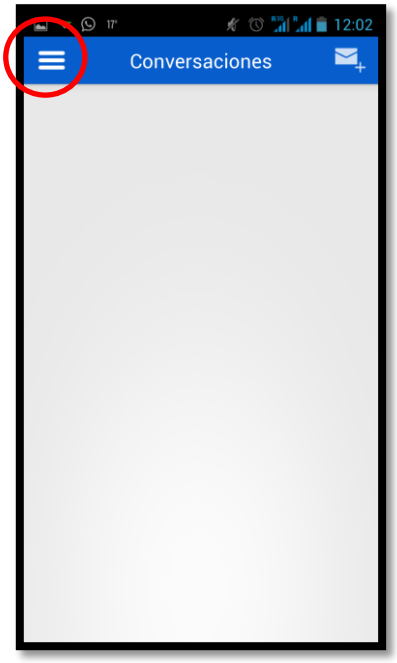


Figura 9.3 Pantalla principal

En la pantalla de conversaciones podrá acceder al menú deslizante pulsando sobre el icono destacado en la figura 9.3.

Tras pulsar este botón podrá acceder a las siguientes opciones: perfil, traducción, notificaciones e idioma.

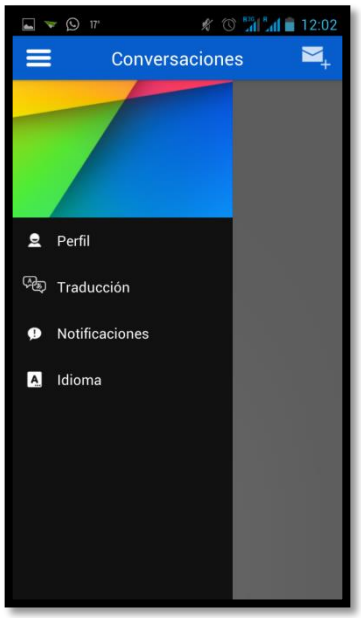


Figura 9.4 Menú de opciones

Pulsando sobre **Perfil** se accederá al perfil de usuario, en el cual podrá ver la imagen de perfil y el número de teléfono registrado en la aplicación.

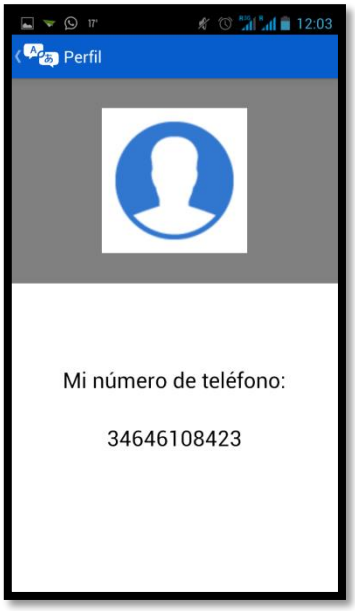


Figura 9.5 Perfil de usuario

Pulsando sobre **Traducción** tendrá la posibilidad de configurar al idioma en el cual desee traducir los mensajes recibidos cuando la traducción esté activa.

Se mostrará un listado con los idiomas disponibles y posteriormente deberá pulsar sobre uno de ellos.

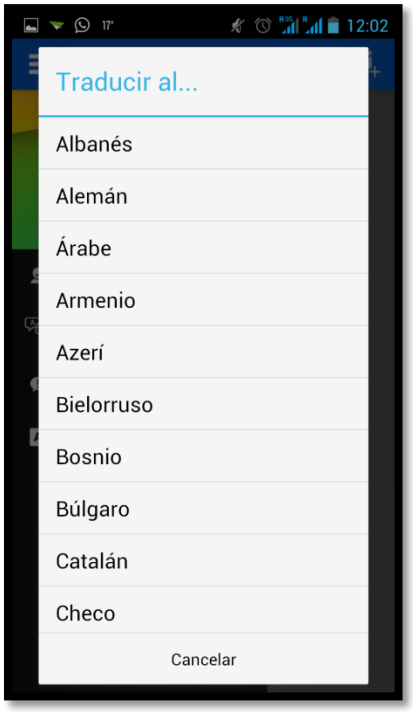


Figura 9.6 Lista de idiomas de destino

Pulsando sobre **Notificaciones** se mostrarán las opciones relacionadas con las notificaciones.

En concreto tendrá la posibilidad de activar y desactivar el sonido de notificaciones. Si desea guardar los cambios pulse en guardar, si desea descartarlos pulse cancelar.

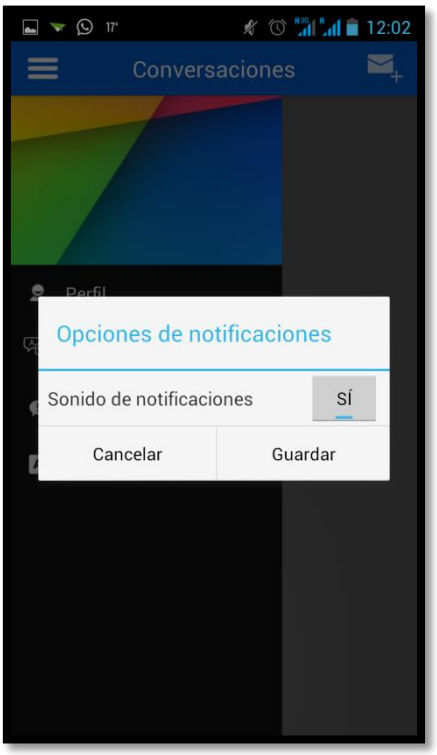


Figura 9.7 Opciones de notificaciones

Pulsando sobre la última opción del menú, **Idioma**, podrá modificar el idioma de la interfaz de usuario. Por defecto, el idioma de la aplicación será el mismo que su sistema Android si está disponible o inglés si no está disponible.

Un diálogo con los idiomas disponibles, actualmente Español, Inglés y Polaco, se mostrará en pantalla. Aquí podrá pulsar sobre el idioma que desee.

Figura 9.8 Idioma de interfaz de usuario.

Inicio de una conversación y envío de mensajes.

Una vez configurada la aplicación a su gusto podrá comenzar con el envío de mensajes.

Si desea empezar una nueva conversación, deberá situarse en la pantalla principal y desde ahí pulsar en el botón de nuevo mensaje, situado en la esquina superior derecha tal y como se destaca en la figura 9.9.

Figura 9.9 Pantalla principal, inicio de conversación.

Tras esto podrá ver la lista de contactos almacenados en su dispositivo que hacen uso de ésta aplicación. Ésta lista se actualizará por primera vez en la primera ejecución de la aplicación. Si desea actualizarla en un futuro pulse sobre el botón de refresco en la esquina superior derecho como se indica en la figura 9.10.

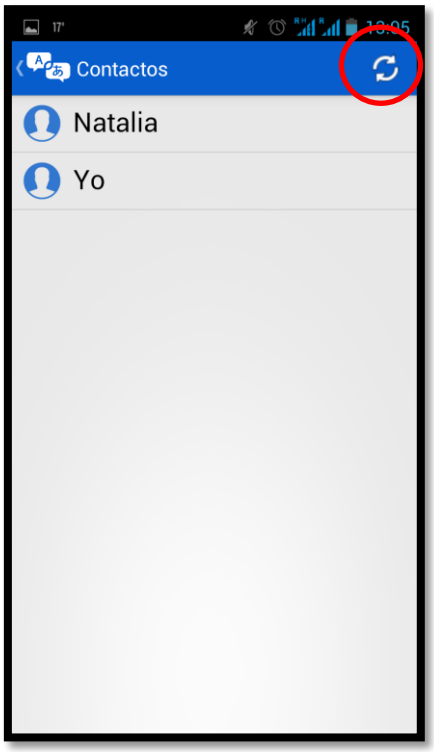
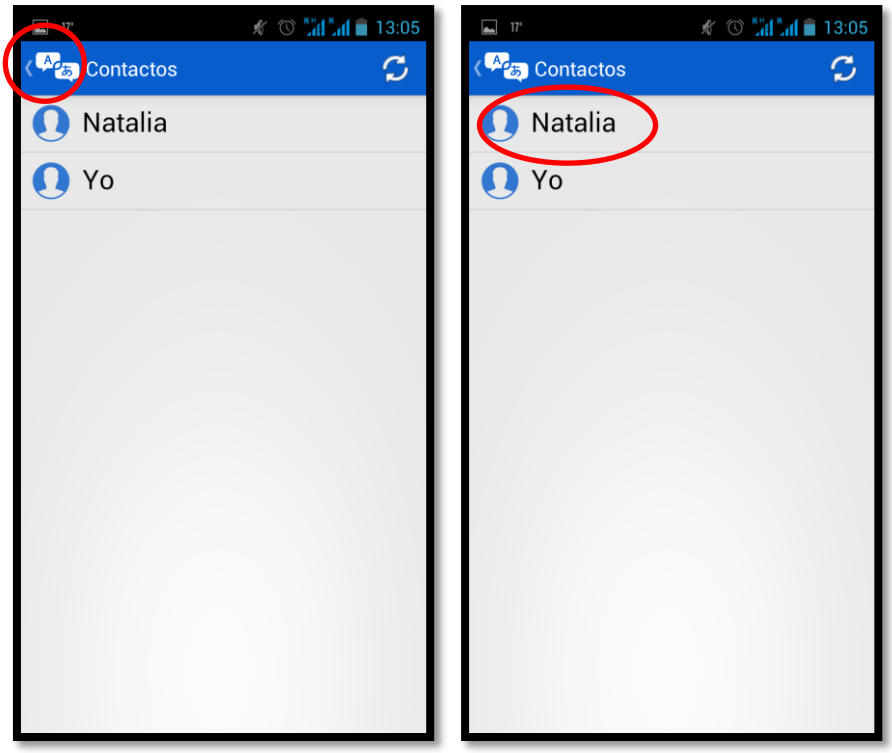


Figura 9.10 Lista de contactos.

Usted podrá volver a la lista de conversaciones pulsando el botón situado en la esquina superior izquierda como se indica en la figura 9.11.

Desde esta pantalla también tendrá la posibilidad de iniciar una nueva conversación con otro usuario. Para hacerlo deberá pulsar sobre el nombre del usuario con quien desee chatear.



.

Figura 9.11 Lista de contactos. Botón atrás.

Figura 9.12 Lista de contactos. Iniciar conversación.

Una vez en la pantalla de conversación con un usuario, podrá enviar nuevos mensajes escribiendo el mensaje deseado en el recuadro de texto inferior y pulsando el botón a su derecha con forma de flecha.

Si desea activar la traducción para los mensajes recibidos deberá pulsar en el botón situado en la esquina superior derecha como se muestra en la figura 9.13.

Una vez pulsado, se mostrará la lista de idiomas disponibles de los que traducir. Si usted sabe el idioma en que están escritos los mensajes que recibe seleccione el idioma de la lista, ya que la traducción será mucho más afín.

Si por el contrario desconoce el idioma en el que está recibiendo los mensajes, pulse sobre el botón de detección automática situado al final de la lista como se muestra en la figura 9.14.

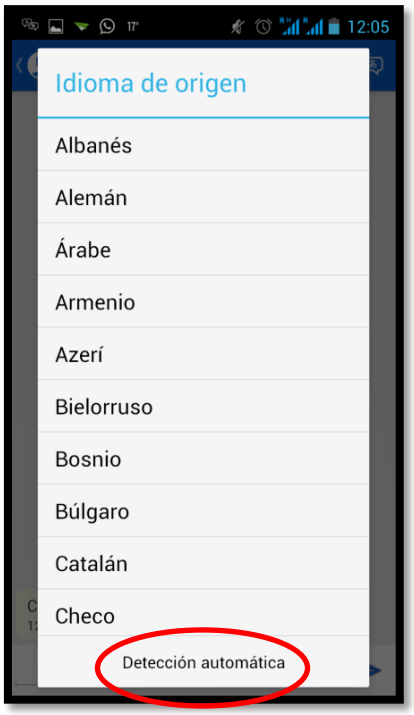


Figura 9.13 Pantalla de conversación. Botón de traducción.

Figura 9.14 Selección de idioma de origen.

Si la traducción se ha activado con éxito, podrá apreciar que el icono de traducción estará de color verde.

A partir de ahora cuando reciba un mensaje de este contacto recibirá el mensaje en el idioma original en un tamaño reducido y el mensaje traducido bajo el original en tamaño mediano como podrá apreciar en la figura 9.16.

Para desactivar la traducción automática pulse de nuevo sobre este botón.

Si desea copiar un mensaje enviado o recibido deberá realizar una pulsación larga sobre el mensaje y éste será copiado automáticamente al portapapeles.

Finalmente podrá apreciar que cuando usted envía un mensaje se mostrará un pequeño icono de un reloj junto a su mensaje. Esto significa que su mensaje aún no se ha enviado. Si este icono perdura varios segundos es posible que sufra problemas de conectividad.

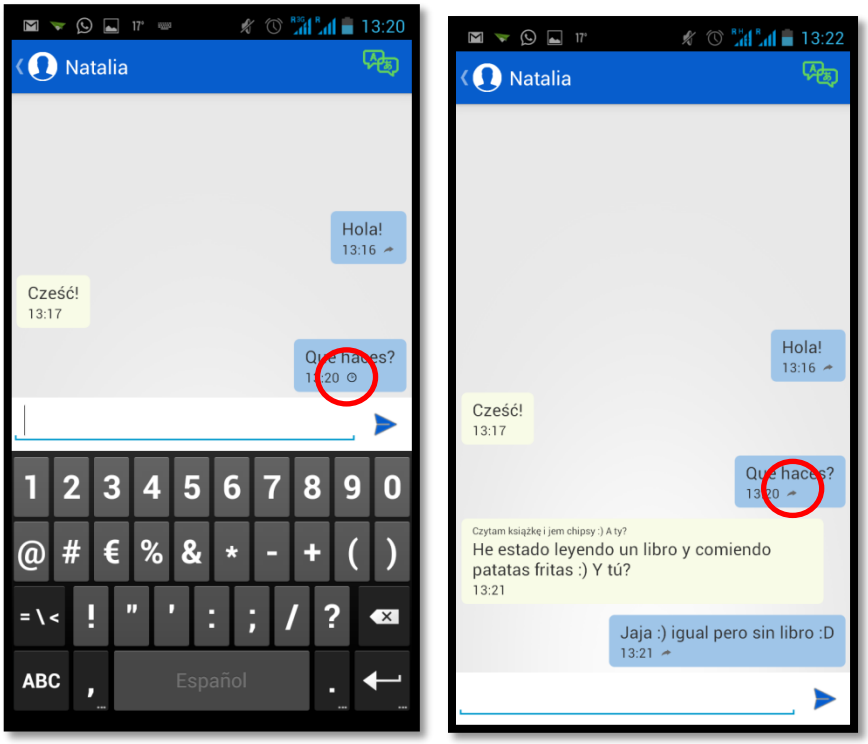


Figura 9.15 Icono de estado de mensaje en espera.

Figura 9.16 Icono de estado de mensaje enviado.

Bibliografía

[1] AIMC EGM 1º Ola 2014 Febrero/Marzo.
http://www.aimc.es/spip.php?action=acceder_document&arg=2538&cle=a25b72a23d55035bcd34827fc8004a0d74241b97&file=pdf%2Finternet114.pdf (Última fecha de acceso 01/09/2014)

[2] Smartphone OS Sales Share – Kantar WorldPanel.
http://www.kantarworldpanel.com/dwl.php?sn=news_downloads&id=282 (Última fecha de acceso 01/09/2014)

[3] Android Developers Dashboards.
https://developer.android.com/about/dashboards/index.html (Última fecha de acceso 28/08/2014)

[4] TechCrunch, Global Messaging App Market.
http://techcrunch.com/2012/12/04/global-messaging-market/ (Última fecha de acceso 01/09/2014)

[5] How Many People Use Top Chat Apps.
http://expandedramblings.com/index.php/how-many-people-use-chat-apps/#.VATDBPl_s3l (Última fecha de acceso 01/09/2014)

[6] Android SDK
http://developer.android.com/sdk/index.html (Última fecha de acceso 01/09/2014)

[7] Android operating system, Wikipedia.
http://en.wikipedia.org/wiki/Android_(operating_system) (Última fecha de acceso 01/09/2014)

[8] XMPP, Wikipedia.
http://en.wikipedia.org/wiki/XMPP (Última fecha de acceso 01/09/2014)

[9] Comparison of XMPP server software, Wikipedia.
http://en.wikipedia.org/wiki/Comparison_of_XMPP_server_software (Última fecha de acceso 01/09/2014)

[10] XMPP.
http://xmpp.org/ (Última fecha de acceso 01/09/2014)

[11] XMPP Extensions.
http://xmpp.org/xmpp-protocols/xmpp-extensions/ (Última fecha de acceso 01/09/2014)

[12] Why Facebook is using MQTT on mobile
https://www.ibm.com/developerworks/community/blogs/mobileblog/entry/why_facebook_is_using_mqtt_on_mobile?lang=en (Última fecha de acceso 01/09/2014)

[13] Google Cloud Messaging for Android.
http://developer.android.com/google/gcm/index.html (Última fecha de acceso 01/09/2014)

[14] IaaS, PaaS, SaasS (Explained and Compared).
http://apprenda.com/library/paas/iaas-paas-saas-explained-compared/ (Última fecha de acceso 01/09/2014)

[15] Cuando hablamos de la nube: ¿qué es IaaS, PaaS, SaaS?
http://www.xatakaon.com/almacenamiento-en-la-nube/cuando-hablamos-de-la-nube-que-es-iaas-paas-saas (Última fecha de acceso 01/09/2014)

[16] Control de versiones, Wikipedia.
http://es.wikipedia.org/wiki/Control_de_versiones (Última fecha de acceso 01/09/2014)

[17] Git
http://git-scm.com/ (Última fecha de acceso 01/09/2014)

[18] Age distribution of adult WhatsApp users in the United States as of February 2014.
http://www.statista.com/statistics/290447/age-distribution-of-us-whatsapp-users/ (Última fecha de acceso 01/09/2014)

[19] StackOverflow. (Última fecha de acceso 01/09/2014)
http://stackoverflow.com/