Escuela Politécnica Superior de Linares
Máster en Ingeniería de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Máster

SERVICIO DE MENSAJERÍA INSTANTÁNEA CON MODERACIÓN Y CONTROL DE TEMÁTICA CON CLIENTE PARA TERMINALES ANDROID

Alumno: **David Miguel Poyatos Reina**

Tutor: Prof. D. Juan Carlos Cuevas Martínez
Depto.: Ingeniería de Telecomunicación

Marzo 2016

RESUMEN

En las últimas décadas se han producido numerosos cambios que han modificado diversos ámbitos de la vida cotidiana de las personas. El auge y el abaratamiento de la tecnología han conllevado a que durante el año 2015 se haya registrado en todo el mundo un mayor número de teléfonos de última generación que cepillos de dientes.

El crecimiento en las ventas de este tipo de dispositivos ha provocado a su vez, que el negocio y el mercado de aplicaciones para estos dispositivos hayan crecido a la par.

Dentro de la gran variedad de aplicaciones existentes, las más populares, y las más utilizadas, son las aplicaciones de comunicación y redes sociales. Por este motivo, en este Trabajo Fin de Master se desarrolla un servicio de mensajería instantánea, aportando ideas y conceptos ya existentes en el mundo de Internet, pero algunos de ellos no empleados hasta ahora en aplicaciones para móviles.

En este documento se describirá cómo se ha desarrollado y cómo funciona el servicio, así como el funcionamiento de la aplicación servidora y la aplicación cliente para terminales Android.

SUMMARY

In recent decades there have been numerous changes that have changed many areas of daily life of people. The rise and cheaper technology has led to that in 2015 have registered worldwide more smartphones than toothbrushes.

The sales growth of these devices has led in turn to the business and applications market for these devices has grown together.

Within the wide range of existing applications, the most used are social and communication applications. Therefore, in this Master's Thesis it has been developed an instant messaging service, providing some ideas and concepts existing in the Internet world, but not commonly seen in mobile applications.

This document describes how the service has been developed and how the application server and client application for Android terminals work.

ÍNDICE GENERAL

1 MEMORIA....1

1.1 INTRODUCCIÓN....1

1.2 OBJETIVOS....2

1.3 ALCANCE....3

1.4 ACTUALIDAD....4

1.5 TECNOLOGÍAS USADAS....5

1.5.1 Android....5

1.5.2 Google Cloud Platform....5

1.5.3 Google App Engine....6

1.5.4 Google Cloud Datastore....7

1.5.5 Google Cloud Storage....7

1.5.6 Google Cloud Endpoints....7

1.5.7 Google Cloud Messaging....8

1.5.8 Seguridad en las comunicaciones....9

1.5.9 Gestor de base de datos SQLite....12

1.5.10 Twilio....13

1.6 DESCRIPCIÓN DEL SISTEMA....14

1.6.1 Arquitectura del sistema....14

1.7 PROTOCOLO DE COMUNICACIONES....16

1.7.1 Encapsulado del mensaje....16

1.7.2 Escenarios de comunicaciones....23

1.8 DESCRIPCIÓN DE LA APLICACIÓN SERVIDORA....29

1.8.1 Estructura de clases....29

1.8.2 Base de datos....29

1.8.3 Envío de mensajes: del cliente al servidor....31

1.8.4 Envío de mensajes: del servidor al cliente....33

1.8.5 Portal web....35

1.9 DESCRIPCIÓN DE LA APLICACIÓN MÓVIL....37

1.9.1 Bibliotecas utilizadas....37

1.9.2 Esquema de clases....44

1.9.3 Permisos de la aplicación *44*
1.9.4 Registro de usuarios *45*
1.9.5 Actualización de perfil de usuario *48*
1.9.6 Actualización de contactos *51*
1.9.7 Conversación con un usuario *51*
1.9.8 Intercambio claves para conversación cifrada extremo a extremo *51*
1.9.9 Envío de archivos *53*
1.9.10 Creación de grupos *53*
1.9.11 Conversación y envío de archivos en un grupo *53*
1.9.12 Envío de invitaciones a contactos que no usan el servicio *54*
1.10 CONCLUSIONES 55
1.11 LÍNEAS FUTURAS 56
1.12 REFERENCIAS 57
1.13 DEFINICIONES Y ABREVIATURAS 60
1.14 LICENCIAS 61
1.14.1 Spongy Castle *61*
1.14.2 MaterialTextField *61*
1.14.3 NineOldAndroids *62*
1.14.4 AndroidProcessButton *62*
1.14.5 MaterialDialog *62*
1.14.6 Picasso *63*
1.14.7 AndroidCrop *63*
1.14.8 AndroidUploadService *64*
1.14.9 CircularReveal *64*
1.14.10 PullRefreshLayout *65*
1.14.11 FilePicker *65*
2 ANEXOS 66
2.1 ANEXO I: MANUAL DE ADMINISTRACIÓN DEL SERVIDOR 66
2.2 ANEXO II: MANUAL DE USUARIO DE LA APLICACIÓN 68
2.2.1 Pantalla principal y contactos *69*
2.2.2 Menú lateral *70*
2.2.3 Pantalla editar perfil *70*
2.2.4 Conversación privada *71*

2.2.5 Envío de notas de audio *72*
2.2.6 Realización y envío de fotografías *72*
2.2.7 Creación de grupos *73*
2.2.8 Conversación en grupo *74*
2.2.9 Invitar a contactos *77*
2.2.10 Permisos *77*
2.3 ANEXO III: CONFIGURACIÓN GOOGLE CLOUD MESSAGING 79
2.3.1 Configuración de la aplicación *81*
2.3.2 Configuración del servidor *85*
2.4 ANEXO IV: CONFIGURACIÓN TWILIO SMS, SERVIDOR Y APLICACIÓN 86
2.4.1 Clase para el envío de SMS desde el servidor *88*
2.4.2 BroadcastReceiver para la recepción de SMS en la aplicación *89*
2.5 ANEXO V: OBJECTIFY API PARA JAVA 91
2.5.1 Definición de una entidad *91*
2.5.2 Operaciones básicas *92*
2.6 ANEXO VI: SUBIDA DE IMÁGENES Y ARCHIVOS A GOOGLE APP ENGINE 93
2.6.1 Imágenes de perfil *93*
2.6.2 Archivos *96*

3 PLIEGO DE CONDICIONES 97
3.1 REQUISITOS PARA EJECUTAR LA APLICACIÓN CLIENTE 97
3.2 ACUERDO DE USUARIO 97
3.3 NORMATIVA 98

4 ESTADO DE LAS MEDICIONES 99

5 PRESUPUESTO 100
5.1 CAPÍTULO 1 - RECURSOS MATERIALES 100
5.2 CAPÍTULO 2 - RECURSOS HUMANOS 100
5.3 CAPÍTULO 3 - COSTES INDIRECTOS 100

ÍNDICE DE FIGURAS

Figura 1.1 - Logo de Wortal 1

Figura 1.2 - Esquema Google Cloud Platform.....6

Figura 1.3 - Arquitectura comunicación aplicación-servidor 8

Figura 1.4 - Esquema RSA (cifrado y firma).....9

Figura 1.5 - Esquema cifrado ECB.....10

Figura 1.6 - Esquema descifrado ECB 10

Figura 1.7 - Esquema modo de cifrado CFB 11

Figura 1.8 - Esquema modo de descifrado CFB 11

Figura 1.9 - Esquema SHA-1 12

Figura 1.10 - Logo de Twilio.....13

Figura 1.11 - Arquitectura del sistema.....15

Figura 1.12 - Concatenación de mensaje.....16

Figura 1.13 – Encriptación AES 17

Figura 1.14 - Concatenación de mensaje de tipo de mensaje y campo extra 17

Figura 1.15 - Cálculo del hash 21

Figura 1.16 - Encapsulación del cuerpo del paquete.....22

Figura 1.17 - Cuerpo del mensaje cifrado 22

Figura 1.18 - Campos cabecera.....22

Figura 1.19 - Cabecera cifrada 22

Figura 1.20 - Paquete enviado por la red 23

Figura 1.21 - Escenario registro usuario 24

Figura 1.22 - Escenario actualización perfil y actualización contactos.....25

Figura 1.23 - Escenario conversación privada 26

Figura 1.24 - Escenario conversación grupo 27

Figura 1.25 - Estructura módulo 'backend' 29

Figura 1.26 - Clase app envío mensaje.....32

Figura 1.27 - Clase servidor recepción mensaje 32
Figura 1.28 - Proceso registro en GCM y envío de mensajes a través de GCM.....33
Figura 1.29 - Clase servidor WortalMessaging.java 34
Figura 1.30 - Servicio recepción mensajes GCM 35
Figura 1.31 - Portal web Wortal.....36
Figura 1.32 – Campo de texto de ejemplo de la biblioteca MaterialTextField 39
Figura 1.33 – Botón de ejemplo de la biblioteca AndroidProcessButton.....39
Figura 1.34 - Diálogos de la biblioteca MaterialDialog.....40
Figura 1.35 - Ejemplo para recortar imagen con la biblioteca AndroidCrop 41
Figura 1.36 - Código de ejemplo AndroidUploadService 42
Figura 1.37 - Pantalla de ejemplo FilePicker.....43
Figura 1.38 - Estructura módulo 'app'.....44
Figura 1.39 - Permisos de la aplicación 45
Figura 1.40 - Inicio proceso registro.....46
Figura 1.41 - Tarea asíncrona proceso registro 47
Figura 1.42 - Respuesta del servidor proceso registro 48
Figura 1.43 - Selección de imagen de perfil 49
Figura 1.44 - Actualización nombre de usuario 50
Figura 1.45 - Actualización estado.....50
Figura 1.46 - Intercambio de claves 52
Figura 2.1 - Pantalla bienvenida Android Studio 66
Figura 2.2 - Interfaz Android Studio 67
Figura 2.3 - Deploy to App Engine 67
Figura 2.4 - Pantalla bienvenida 1.....68
Figura 2.5 - Pantalla bienvenida 2.....68
Figura 2.6 - Pantalla bienvenida 3.....68
Figura 2.7 - SMS verificación 68

Figura 2.8 - Pantalla principal 69
Figura 2.9 - Pantalla de contactos.....69
Figura 2.10 - Menú lateral.....70
Figura 2.11 - Pantalla editar perfil 71
Figura 2.12 - Editar nombre 71
Figura 2.13 - Editar estado.....71
Figura 2.14 - Pantalla de conversación.....71
Figura 2.15 - Menú desplegable adjuntos 71
Figura 2.16 - Explorador de archivos 71
Figura 2.17 - Candado abierto 72
Figura 2.18 - Candado cerrado.....72
Figura 2.19 - Opciones mensaje 72
Figura 2.20 - Botón micrófono.....72
Figura 2.21 - Interfaz grabando audio 72
Figura 2.22 - Opción 'Foto' del menú de Adjuntar 73
Figura 2.23 - Botón cámara de fotos.....73
Figura 2.24 - Interfaz creación grupo 73
Figura 2.25 - Pantalla selección icono grupo.....73
Figura 2.26 - Datos del grupo completados 73
Figura 2.27 - Pantalla elección participantes.....74
Figura 2.28 - Pantalla elección moderador.....74
Figura 2.29 - Pantalla principal con el grupo creado 74
Figura 2.30 - Pantalla conversación grupo, menú adjuntar 75
Figura 2.31 - Pantalla conversación con mensajes 75
Figura 2.32 - Pantalla detalle grupo 75
Figura 2.33 - Menú opciones mensaje grupo 75
Figura 2.34 - Mensaje inapropiado.....76

Figura 2.35 - Cambiar moderador del grupo 77
Figura 2.36 - Cambiar administrador del grupo 77
Figura 2.37 - Pantalla envío invitaciones....77
Figura 2.38 - Invitación recibida 77
Figura 2.39 - Creación Proyecto en Google Developers Console 79
Figura 2.40 - Creación de proyecto Wortal....80
Figura 2.41 - Obtención SENDER ID y API Key....80
Figura 2.42 - Activación API GCM 81
Figura 2.43 - Descarga biblioteca 'Google Play services' 81
Figura 2.44 - Bibliotecas del proyecto 82
Figura 2.45 - Añadir biblioteca 'Google Play services' 82
Figura 2.46 - Permisos para usar GCM....83
Figura 2.47 - Resto de permisos para usar GCM 83
Figura 2.48 - Declaración receptor de mensajes GCM....83
Figura 2.49 - Registro en servidor GCM....84
Figura 2.50 - Clase receptora de mensajes GCM 85
Figura 2.51 - Formulario registro Twilio....86
Figura 2.52 - Búsqueda de números....86
Figura 2.53 - Número de teléfono de Twilio 87
Figura 2.54 - Enviar SMS desde la web 87
Figura 2.55 - Clase servidor envío SMS....89
Figura 2.56 - Servicio recepción SMS aplicación cliente 90
Figura 2.57 - Definición de una entidad con Objectify 91
Figura 2.58 - Operaciones básicas con Objectify 92
Figura 2.59 - Servlet ImgUpload 93
Figura 2.60 - Asynctask subida imagen perfil....94
Figura 2.61 - Método subida imagen perfil 94

Figura 2.62 - Servlet ImgUploaded 95

Figura 2.63 - Servlet FileUpload.....96

ÍNDICE DE TABLAS

Tabla 1.1 - Tipos de mensaje recibidos por el servidor 17

Tabla 1.2 - Tipos de mensaje recibidos por el cliente.....19

Tabla 1.3 - Tabla WortalUser 30

Tabla 1.4 - Tabla WortalGroup.....30

Tabla 1.5 - Tabla de definiciones y abreviaturas 60

Tabla 4.1 - Mediciones ‘Recursos materiales’ 99

Tabla 4.2 - Mediciones ‘Recursos humanos’.....99

Tabla 4.3 - Mediciones ‘Costes indirectos’ 99

Tabla 5.1 - Presupuesto 'Recursos materiales' 100

Tabla 5.2 - Presupuesto 'Recursos humanos' 100

Tabla 5.3 - Presupuesto 'Costes indirectos' 100

1 MEMORIA

1.1 Introducción

El propósito de este Trabajo Fin de Máster es el de crear un servicio de mensajería instantánea. Para ello, se creará una aplicación para terminales con sistema operativo Android que será la herramienta principal para que los usuarios puedan hacer uso del servicio desarrollado.

Conjuntamente se desarrollará una aplicación servidora, en la que se apoyará la aplicación cliente, que funcionará como infraestructura común para los usuarios del servicio.

El servicio se apoya en varias tecnologías:

- Android como sistema operativo para el que se desarrolla la aplicación cliente.
- Google App Engine como plataforma de hospedaje de la aplicación servidora.
- Google Cloud Endpoints como servicio de comunicación entre aplicación y servidor.
- Google Cloud Messaging como servicio de comunicación entre servidor y aplicación.
- RSA, AES y SHA1 para garantizar autenticidad e integridad de los mensajes.
- Google Cloud Datastore como base de datos en el servidor y SQLite en la aplicación móvil.

Para simplificar el nombre del proyecto, en adelante se referirá al servicio con el nombre *Wortal*.

Figura 1.1 - Logo de Wortal

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Máster es crear un servicio de envío y recepción de mensajes instantáneos entre terminales Android apoyándose en la plataforma Google Cloud Messaging (GCM) sobre Google App Engine (GAE).

Además del servicio y la arquitectura necesaria para el intercambio de mensajes, debe desarrollarse una aplicación para terminales Android con las suficientes garantías de seguridad, de forma que se respete la privacidad de los usuarios. Como requisito de implementación, la aplicación deberá ser compatible con el mayor número de terminales posible, siempre y cuando cumplan con los requisitos mínimos que se establezcan.

Las funciones principales del servicio que deben implementarse son las siguientes:

- Registro basado en invitaciones a través de los contactos del móvil.
- Autenticación segura de usuarios.
- Envío de mensajes privados.
- Envío de mensajes a grupos.
- Gestión de grupos de usuarios.
- Envío de archivos de cualquier tipo.
- Rol de moderador de grupo.
- Posibilidad de establecer información adicional a un grupo: mensajes del día, normas, mensajes inapropiados, etc.

1.3 Alcance

Con el desarrollo del proyecto se pretende crear un servicio de mensajería similar a los ya existentes, pero con la diferencia de añadir funcionalidades nuevas que algunos de ellos no poseen, como por ejemplo, la capacidad de enviar cualquier tipo de archivo, o la capacidad de crear grupos y temas de conversación con un funcionamiento similar a los conocidos *foros* de Internet. Entre otras características se encuentran las siguientes:

- Creación de grupos de usuarios estableciendo una temática y unas normas.
- Creación de un moderador del grupo que podrá realizar diversas acciones sobre los mensajes enviados por el grupo al que pertenezca: marcar como inapropiado, marcar como mensaje del día, etc.

A parte de los grupos, un usuario podrá establecer una conversación privada con otro usuario, con la opción de cifrar la conexión extremo a extremo, es decir, el contenido de la conversación sólo podrá ser descifrado por el destinatario. De esta forma, los usuarios del servicio pueden estar seguros de que su comunicación no podrá ser interceptada y descifrada por un 'usuario espía' y falsificar mensajes o realizar cualquier otro tipo de actividades ilícitas.

1.4 Actualidad

En los últimos años se ha producido un crecimiento exponencial en la venta de teléfonos de última generación, también conocidos como *smartphones*, lo que ha provocado que a su vez el mercado de aplicaciones haya crecido de manera considerable, siendo uno de los negocios que se prevé siga creciendo en los próximos años.

Las aplicaciones más descargadas [1] [2] son las aplicaciones sociales y de mensajería instantánea. Dentro de este grupo de aplicaciones, la más popular entre los usuarios es *WhatsApp*®, ya que permite establecer conversaciones individuales y grupales, además de permitir enviar imágenes y archivos de audio. Esta aplicación u otras de similares características están orientadas para usarse para charlar entre amigos, aunque debido a su popularidad han llegado a usarse en otros ámbitos (grupos de trabajo, compañeros de clase, etc.)

El objetivo de Wortal es desarrollar una aplicación que pueda usarse en ámbitos académicos y laborales, de forma que los chats grupales tengan más funciones que faciliten la comunicación y la cooperación de los integrantes del grupo.

1.5 Tecnologías usadas

La gran explosión en los últimos años de la tecnología y la telefonía ha causado que la mayoría de la población posea un *smartphone* con capacidades de cómputo y funciones similares, y a veces superiores a las de un ordenador, con sistemas operativos cada vez más avanzados y parecidos a los de los ordenadores, cerrándose cada vez más la brecha entre ambos tipos de sistemas operativos, por ejemplo, actualmente todos los dispositivos con Windows 10 utilizan el mismo sistema, adaptándose automáticamente al dispositivo en que se utilice: teléfono, tableta u ordenador.

1.5.1 Android

Android es un sistema operativo basado en el *kernel* de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, y también para relojes inteligentes, televisores y automóviles, inicialmente desarrollado por la empresa Android Inc., Google la respaldó económicamente y más tarde la compró en 2005. Android fue presentado en 2007 junto con la fundación Open Handset Alliance, un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles. Es de código abierto, lo que quiere decir que cualquier desarrollador puede crear y desarrollar aplicaciones para este sistema. Su entorno de ejecución está basado en Java.

Está instalado en dispositivos de una gran variedad de marcas. Esto hace que según las características del terminal, podremos instalar una versión u otra del sistema y, por lo tanto, es posible que no se puedan ejecutar ni las mismas aplicaciones, ni con las mismas prestaciones en todos ellos.

1.5.2 Google Cloud Platform

Google Cloud Platform (GCP) [3] es la integración de todos los servicios de Cloud Computing de Google en una misma plataforma, dotándoles de mayor cohesión y homogeneidad. Mediante esta plataforma es posible desarrollar un servidor, también llamado *backend*, para una aplicación desarrollada para *smartphones*. Gracias a la infraestructura de Google no es necesario preocuparse por la escalabilidad, ya que ésta será automática.

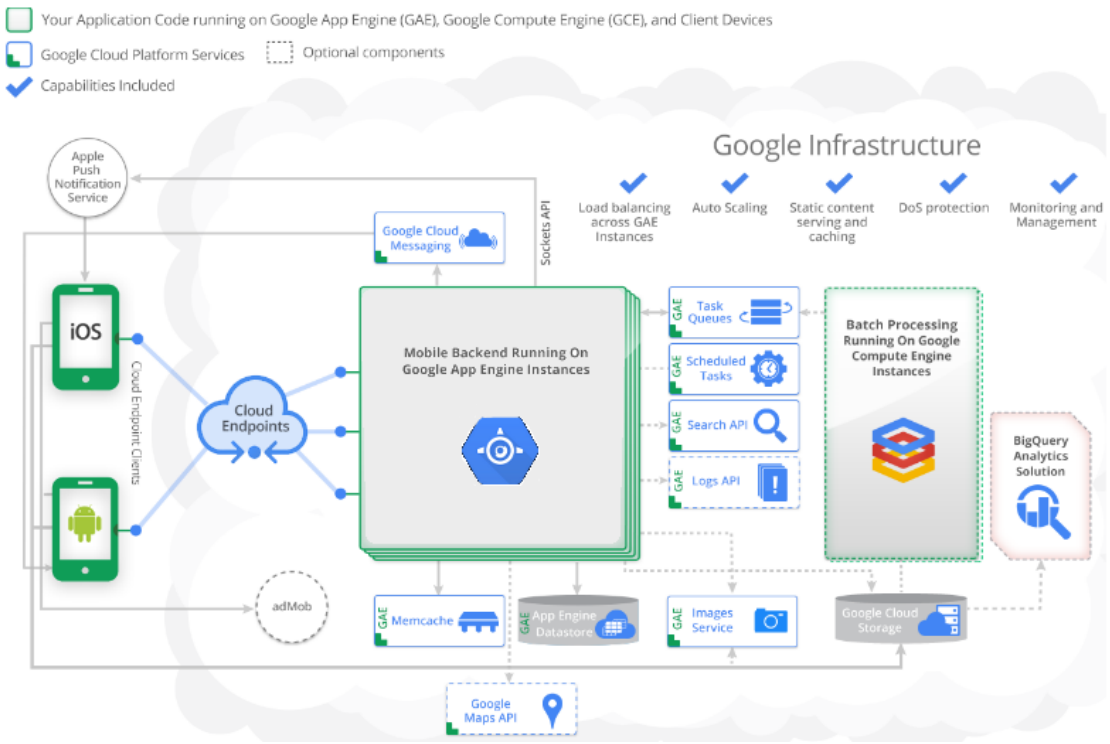


Figura 1.2 - Esquema Google Cloud Platform
Fuente:
(http://static1.1.sqspcdn.com/static/f/923743/24774619/1398262780507/GoogleCloudEndPointsl.pdf)

En la imagen superior pueden verse dos partes claramente diferenciadas: las aplicaciones instaladas en los dispositivos y la nube que identifica a Cloud Platform. En la parte izquierda están las aplicaciones para dispositivos iOS y Android que se comunican a través de Cloud Endpoints con la aplicación servidora desplegada en Google App Engine. Esta aplicación servidora tiene acceso a otros servicios de Cloud Platform (Cloud Messaging, Cloud Datastore, Cloud Storage, etc.).

1.5.3 Google App Engine

Google App Engine (GAE) [4] es uno de los servicios ofrecido por Google dentro de la plataforma Google Cloud Platform. Este servicio es del tipo Plataforma como Servicio el cual permite publicar aplicaciones web en línea sin necesidad de que el desarrollador se preocupe por la infraestructura y así poder enfocarse en el desarrollo de la aplicación con la posibilidad de ejecutarla sobre la infraestructura de Google, es decir, la infraestructura que Google usa para sus propios productos. Las aplicaciones que pueden ejecutarse en GAE pueden ser escritas en varios lenguajes como Java, Python o PHP.

Además, al usar GAE, el desarrollador no debe preocuparse por la estabilidad de la aplicación, ya que cuenta con un balanceador de carga y escalamiento automáticos.

Al ser parte de la familia Google Cloud Platform, el desarrollador tiene la capacidad de integrar GAE con otros productos de la familia.

1.5.4 Google Cloud Datastore

Google Cloud Datastore (GCD) [5] es una tecnología de base de datos NoSQL, ofrecida dentro de la familia Google Cloud Platform, ideada como la opción principal a la hora de almacenar datos de aplicaciones ejecutadas en GAE.

Está basado en transacciones atómicas, que pueden contener más de una operación en la base de datos. Una transacción no finaliza hasta que todos sus procesos hayan concluido, lo cual es muy útil en situaciones en las que se hacen muchas operaciones sobre la misma información al mismo tiempo.

Como limitación, GCD sólo permite almacenar de forma gratuita hasta 1GB y un máximo de 200 índices.

1.5.5 Google Cloud Storage

Google Cloud Storage (GCS) [6] es una tecnología de almacén de datos. Permite configurar de forma sencilla diversos espacios o *buckets* como si fueran carpetas de forma que podamos organizar los archivos almacenados.

Como limitación, GCS sólo permite almacenar de forma gratuita hasta 5GB, previa activación de una cuenta de prueba gratuita durante 60 días. Sin dicha activación no se puede hacer uso del almacenamiento.

1.5.6 Google Cloud Endpoints

La comunicación entre aplicación y servidor se realiza mediante la tecnología Google Cloud Endpoints (GCE) [7], un conjunto de herramientas que permiten al desarrollador de forma sencilla generar las APIs para poder comunicar las aplicaciones clientes (webs y móviles) con una aplicación servidora desplegada en GAE.

Además GCE permite realizar la negociación de seguridad automáticamente, sin que el desarrollador tenga que configurar nada, pudiendo realizar conexiones SSL automáticamente entre clientes y servidor.

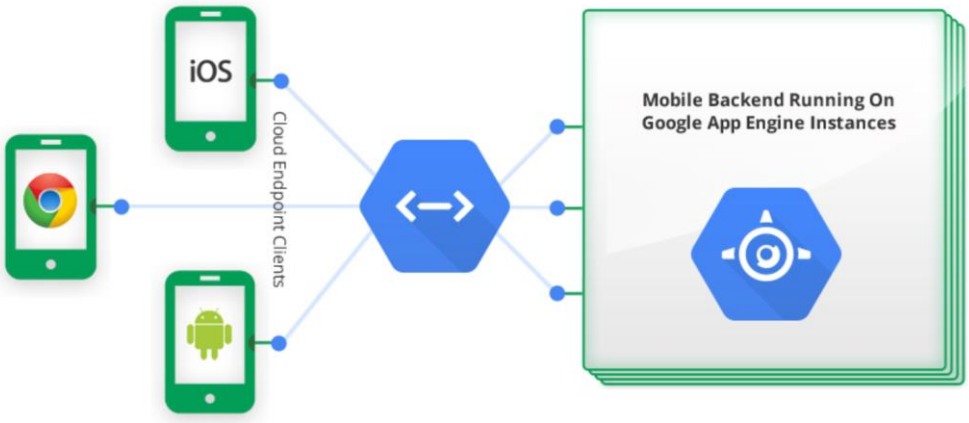


Figura 1.3 - Arquitectura comunicación aplicación-servidor
Fuente:
(http://static1.1.sqspcdn.com/static/f/923743/24774619/1398262780507/GoogleCloudEndPointsl.pdf)

1.5.7 Google Cloud Messaging

La comunicación entre servidor y aplicación se realiza mediante la tecnología Google Cloud Messaging (GCM) [8], un servicio que permite al desarrollador enviar información desde la aplicación servidora a las aplicaciones clientes ejecutadas en dispositivos móviles. El servicio controla todo lo relacionado con el almacenamiento en cola de los mensajes y su entrega a las aplicaciones, es decir, una vez que la aplicación servidora envía el mensaje a retransmitir a la aplicación cliente, el servicio GCM lo almacenará hasta que el cliente pueda recibirlo, facilitando así la tarea del desarrollador a la hora de programar la aplicación servidora. Algunas de sus características básicas son:

- Para que una aplicación cliente reciba un mensaje enviado por GCM, no tiene que estar ejecutándose en primer plano, sino que el sistema la "despertará" cuando llegue el mensaje.
- Requiere dispositivos con Android 2.2 o superior.
- Hace uso de una conexión existente a los servicios de Google, por lo que es necesario que los usuarios tengan configurada una cuenta de Google en el dispositivo. Para dispositivos con Android 4.0.4 o superior no es necesario tener esta cuenta.

En el apartado 1.8.4 Envío de mensajes: del servidor al cliente se detalla cómo se usa el servicio Google Cloud Messaging.

1.5.8 Seguridad en las comunicaciones

Para proporcionar seguridad a las comunicaciones entre aplicación y servidor, se hace uso de varias tecnologías de cifrado y resumen. La forma concreta en que se aplican según el tipo de mensaje se verá en el apartado 1.7 Protocolo de comunicaciones.

- RSA [9]: es un algoritmo de cifrado asimétrico el cual utiliza clave pública (distribuida públicamente) y otra privada para el cifrado de la información (se debe mantener bien almacenada en secreto). Al enviar un mensaje, el emisor utiliza la clave pública del destinatario. El destinatario descifrará el mensaje con su clave privada. Al utilizar este algoritmo se garantiza que el mensaje sólo puede procesarlo el destinatario del mensaje.

Debido a problemas con la API de Android, muchas funciones no están totalmente soportadas por lo que era imposible realizar este tipo de cifrado. Para solucionarlo se utiliza una biblioteca adicional que sí permite realizar este tipo de cifrado, se conoce como *SpongyCastle* [10]. Para evitar posibles incompatibilidades, esta biblioteca se utiliza tanto en la aplicación móvil como en el servidor.

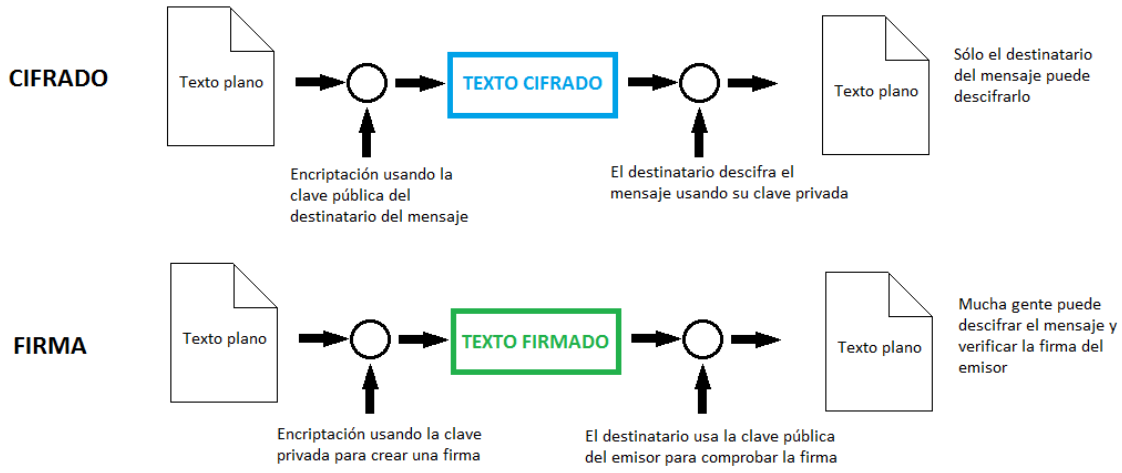


Figura 1.4 - Esquema RSA (cifrado y firma)

Como método de operación de cifrado de bloque se utiliza ECB (Electronic CodeBook), de forma que el mensaje se divide en bloques y se encripta por separado. Es la forma más fácil de operación, no necesita vector de inicialización, y su principal desventaja es que siempre la misma entrada va a tener la misma salida.

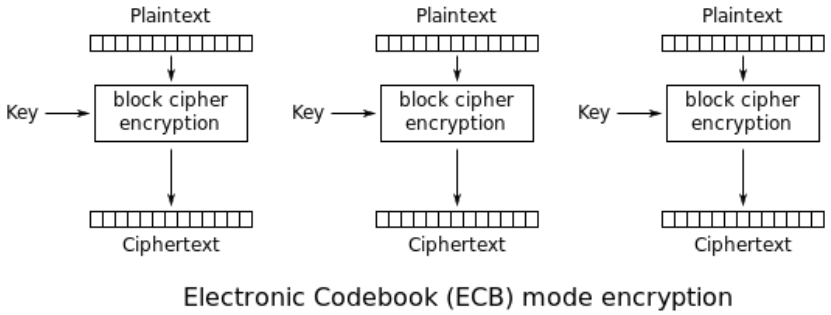


Figura 1.5 - Esquema cifrado ECB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

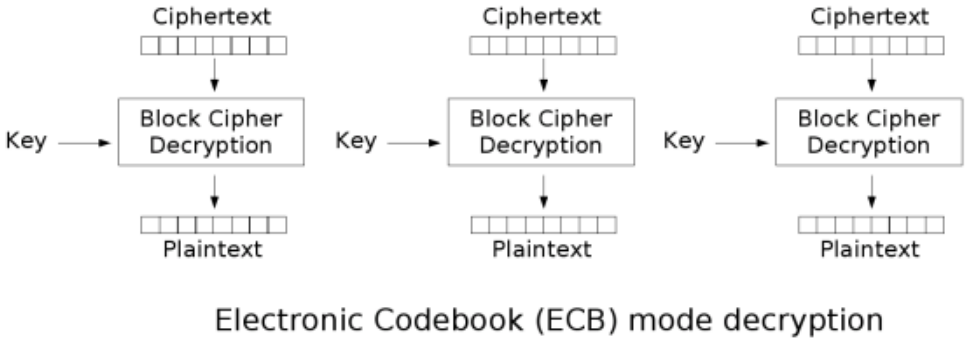


Figura 1.6 - Esquema descifrado ECB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Obligatoriamente al utilizar el algoritmo RSA, hay que utilizar un estándar de *padding* (relleno), ya que si no se utiliza no se garantiza total seguridad. Se utiliza PKCS1 [11], el primero de la familia de estándares conocida como *Public-Key Cryptography Standards*. Define las propiedades matemáticas de las claves públicas y privadas, operaciones primitivas para encriptar y firmar, esquemas seguros de criptografía, etc.

Este algoritmo se utiliza para garantizar que el servidor Wortal es el único que puede procesar la información codificada.

- Advanced Encryption Standard (AES) [12]: basado en varias sustituciones, permutaciones y transformaciones lineales, ejecutadas en bloques de datos de 16 bytes. AES es uno de los algoritmos más seguros y más utilizados hoy en día. Usado por la Agencia de Seguridad Nacional – National Security Agency (NSA) – de Estados Unidos para garantizar el nivel de seguridad más alto a la información calificada de "*Top Secret*". Con AES se utiliza una misma clave para el cifrado y para el descifrado, de esta forma

se garantiza la confidencialidad, cifrando los datos enviados/recibidos con una clave que comparten los usuarios que intervienen en la comunicación. El modo de operación utilizado es CFB (Cipher FeedBack), el cual utiliza un vector de inicialización para 'aleatorizar' la salida del cifrador. En este caso no se utiliza relleno.

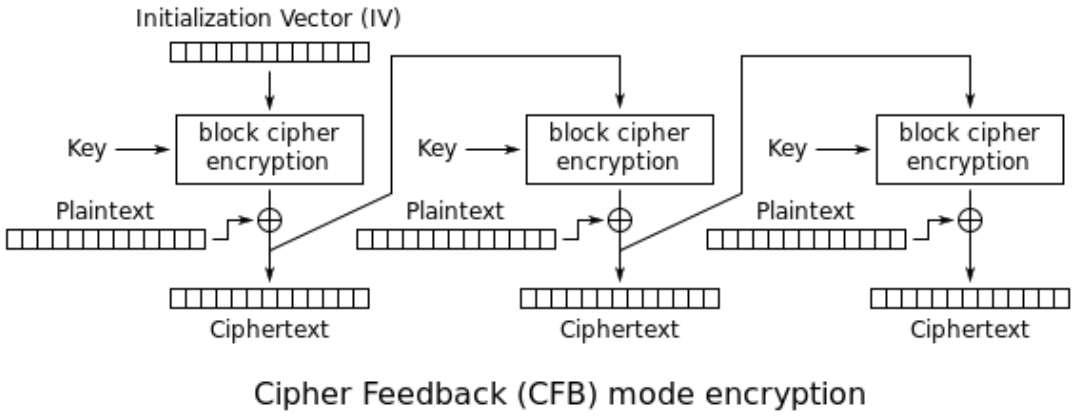


Figura 1.7 - Esquema modo de cifrado CFB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

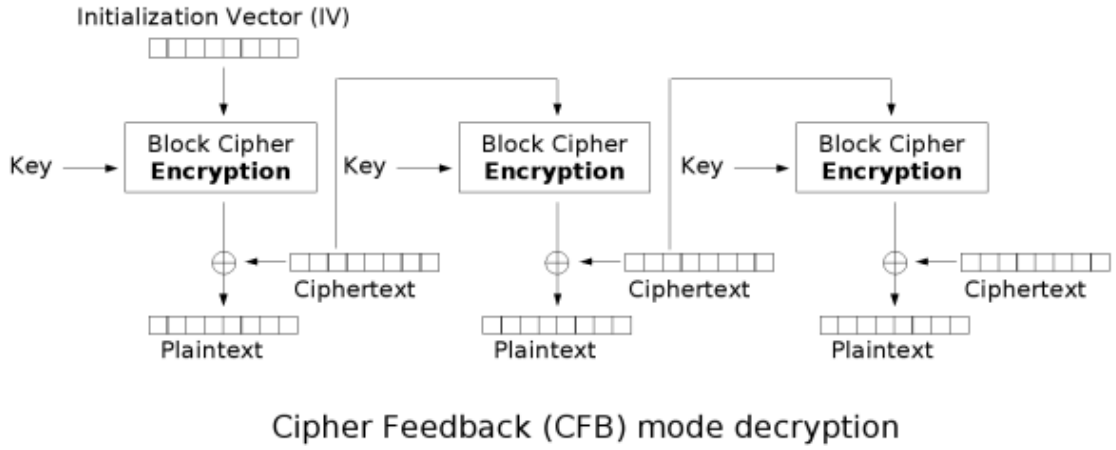


Figura 1.8 - Esquema modo de descifrado CFB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

- Secure Hash Algorithm-1 (SHA-1) [13]: es una de las integrantes de la familia de funciones hash SHA. Se utiliza para garantizar que la información intercambiada no ha sufrido ningún cambio. El emisor calculará el resumen del mensaje a enviar por la red y lo adjuntará a dicho mensaje. Cuando el destinatario reciba el mensaje, calculará el resumen de los datos recibidos y si el resumen recibido como el calculado son distintos, significará que se ha producido algún error en la transmisión.

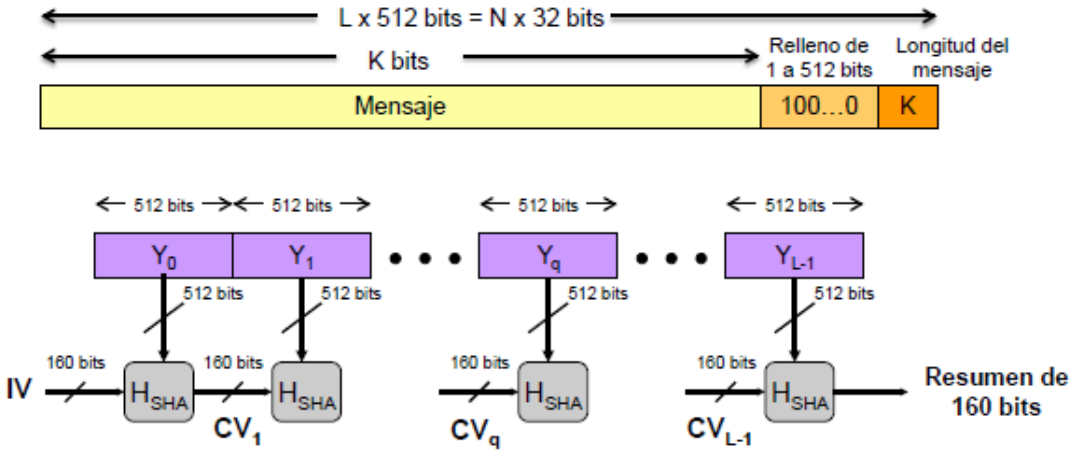


Figura 1.9 - Esquema SHA-1
Fuente: Diapositivas asignatura optativa Aplicaciones Telemáticas para la Administración – Grado en Ingeniería Telemática – Profesor: Juan Carlos Cuevas Martínez

1.5.9 Gestor de base de datos SQLite

SQLite [14] es una base de dato Open Source similar a MySQL, muy popular en muchos dispositivos pequeños o embebidos, como pueden ser los terminales con sistema operativo Android.

Las ventajas que presenta utilizar SQLite es que no requiere configuración, no tiene un servidor de base de datos ejecutándose en un proceso separado y es relativamente simple su empleo.

Algunas de las características principales de SQLite:

- La base de datos completa se encuentra en un solo archivo.
- Puede funcionar enteramente en memoria, lo que la hace muy rápida.
- No necesita bibliotecas adicionales para funcionar.
- Cuenta con bibliotecas para muchos lenguajes de programación.
- Soporta texto en formato UTF-8, UTF-16 y datos numéricos de 64 bits.
- Soporta funciones SQL definidas por el usuario (UDF).
- El código fuente es de dominio público.

Esta base de datos se utiliza para guardar los datos de información del usuario, información de grupos, datos de contactos y mensajes enviados/recibidos.

1.5.10 Twilio

Twilio [15] es un servicio que permite a los desarrolladores enviar SMS a números de teléfono móvil desde las aplicaciones servidor utilizando la API que ofrecen para diversos lenguajes de programación.

Figura 1.10 - Logo de Twilio

En el Anexo IV se detalla cómo se ha configurado el servicio para el envío de SMS desde el servidor, así como la programación necesaria en la aplicación cliente para la lectura de los SMS recibidos.

1.6 Descripción del sistema

Para crear el servicio de mensajería instantánea objeto de este TFM, se han desarrollado desde cero dos aplicaciones:

- Aplicación cliente para terminales Android: desarrollada desde cero con lenguaje de programación JAVA. Es la herramienta necesaria para hacer uso del servicio, ya que realiza las funciones de envío y recepción de mensajes y archivos entre los usuarios del servicio. Para el envío de mensajes se utiliza la tecnología GCE y para la recepción GCM. La base de datos diseñada se basa en la tecnología SQLite mencionada anteriormente.
- Aplicación servidora: desarrollada desde cero con lenguaje de programación JAVA y hospedada en App Engine apoyándose en la base de datos Cloud Datastore. Realiza las funciones de comprobación y reenvío de mensajes provenientes de la aplicación cliente, además de almacenar la información de perfil de cada usuario, y la información sobre los grupos de usuarios. Para la recepción de mensajes se utiliza la tecnología GCE y para el envío GCM. La base de datos diseñada se basa en GCD mencionada anteriormente.

1.6.1 Arquitectura del sistema

La infraestructura necesaria para el despliegue del sistema consta de los siguientes elementos:

- Aplicación Android.
- Aplicación Servidora con base de datos GCD.
- Conexión a Internet.
- Servicio GCM: en el apartado '1.8.4 Envío de mensajes: del servidor al cliente' se detalla el funcionamiento de este sistema.

En la siguiente Figura se muestra la arquitectura del sistema. En ella aparecen varios terminales con la aplicación Android que se comunican con el servidor a través de GCE. Una vez que la aplicación servidora recibe un mensaje de un cliente, lo procesará de la forma adecuada, realizando las comprobaciones necesarias en la base de datos GCD. Una vez procesado, la respuesta se enviará, de una forma u otra dependiendo del tipo de mensaje, a través del servicio GCM hacia sus respectivos destinatarios.

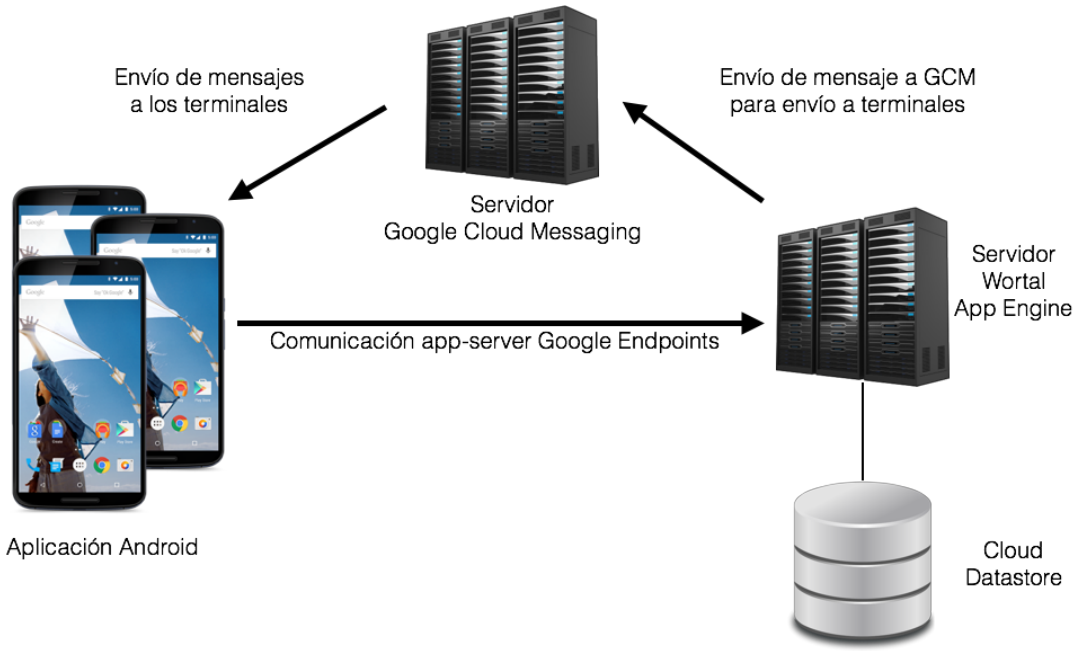


Figura 1.11 - Arquitectura del sistema

En los siguientes apartados, se describirá en primer lugar, el protocolo de comunicaciones desarrollado para el envío seguro de información a través de la red y posteriormente, las aplicaciones cliente y servidora desarrolladas.

1.7 Protocolo de comunicaciones

Uno de los requisitos en el desarrollo del sistema, es que debe implementarse con las suficientes garantías de seguridad. Por este motivo se ha desarrollado desde cero un protocolo de comunicaciones, que haciendo uso de diferentes algoritmos de seguridad, los mensajes enviados por los usuarios del sistema se encapsulan de una forma determinada para asegurar así su intimidad ante posibles 'espías' de la red. No obstante, también se ofrece al usuario el cifrado extremo a extremo, de forma que ni el servidor alojado en GAE tendrá conocimiento del contenido de las conversaciones de los usuarios.

1.7.1 Encapsulado del mensaje

Al mensaje que se enviará se añadirá al principio el identificador del destinatario del mensaje. En mensajes de actualización de perfil, el destinatario de la respuesta del mensaje será el propio emisor, mientras que en los mensajes enviados en una conversación, el identificador se corresponde con el destinatario del mensaje.

Como delimitador entre un campo y otro se utiliza el símbolo *#*, de esta forma, utilizando, tanto en la aplicación como el servidor, la función *split(#)*, se podrán obtener de forma separada cada uno de los campos que forma la cadena de texto.

Figura 1.12 - Concatenación de mensaje

En los mensajes cifrados extremo a extremo el campo MENSAJE irá cifrado con la clave AES establecida en el intercambio de claves previo al establecimiento de la conexión. En el Apartado 1.9.7 se detalla cómo se produce dicho intercambio de claves.

1.7.1.1 Primer cifrado AES

El siguiente paso es el cifrado del identificador del destinatario y del mensaje utilizando la clave AES correspondiente que se generó durante el proceso de registro.

En dicho proceso se crearon dos claves:

- Serveraeskey: clave utilizada por el servidor cuando envía un mensaje a un cliente.
- Clientaeskey: clave utilizada por el cliente cuando envía un mensaje al servidor.

Figura 1.13 – Encriptación AES

1.7.1.2 Concatenación de tipo de mensaje y campo extra

El siguiente paso, una vez cifrado el identificador de destinatario y el mensaje, es añadir el tipo de mensaje y un campo extra. Este campo extra se utiliza sólo en algunos mensajes, mientras que en otros este campo tendrá texto aleatorio.

Figura 1.14 - Concatenación de mensaje de tipo de mensaje y campo extra

1.7.1.3 Tipos de mensaje

Para procesar de la forma adecuada cada tipo de mensaje, se han creado unos identificadores, una cadena de texto con caracteres concretos, que añadidos en la cabecera, permiten al servidor y a la aplicación diferenciar cada uno de los mensajes.

Los tipos de mensaje que el servidor puede recibir (enviados por la aplicación cliente) según identificador son los siguientes:

Tabla 1.1 - Tipos de mensaje recibidos por el servidor

TIPO	COMANDO	DESCRIPCIÓN
WUR	Wortal User Register	Utilizado para indicar el registro de usuario.
WUV	Wortal User Verify	Utilizado para verificar la identidad del usuario en el registro.
WUD	Wortal User Delete	Utilizado para dar de baja a un usuario del servicio.
CPN	Change Profile Name	Utilizado para cambiar el nombre de usuario.
CPS	Change Profile Status	Utilizado para cambiar el estado de usuario.

CPP	Change Profile Photo	Utilizado para cambiar la imagen de perfil de usuario.
NCM	Normal Chat Message	Utilizado en el envío de un mensaje de una conversación privada
CPC_SYN	Create Private Chat SYNcrhonize	Utilizado para iniciar el intercambio de claves para establecer una conversación privada cifrada extremo a extremo.
CPC_SYN_ACK	Create Private Chat SYNchronize ACKnowledgement	Utilizado para indicar la recepción del mensaje de inicio de intercambio de claves CPC_SYN.
CPC_ACK	Create Private Chat ACKnowledgement	Utilizado para indicar que la conversación privada extremo a extremo se ha completado.
CPC_FIN	Create Private Chat FINish	Utilizado para terminar una conversación privada extremo a extremo.
PCM	Private Chat Message	Utilizado en el envío de un mensaje cifrado extremo a extremo de una conversación privada.
CTG	Create Topic Group	Utilizado para crear un nuevo grupo.
CTG_ACK	Create Topic Group ACKnowledgement	Utilizado para aceptar la invitación de participación en un grupo.
CTG_DEL	Create Topic Group DELete	Utilizado para eliminar un participante de un grupo.
DTG	Delete Topic Group	Utilizado para borrar un grupo.
TGM	Topic Group Message	Utilizado en el envío de mensajes de grupo.
TGD	Topic Group Day	Utilizado por el moderador para establecer el 'Mensaje del día' de un grupo.
TGNM	Topic Group New Message	Utilizado para cambiar el moderador de un grupo.
TGNA	Topic Group New Administrator	Utilizado para cambiar el administrador de un grupo.

TGEG	Topic Group Exit Group	Utilizado por un participante para abandonar un grupo.
TGCG	Topic Group Close Group	Utilizado por el administrador y/o moderador para cerrar un grupo.
TGI	Topic Group Inappropiate	Utilizado por el moderador para marcar un mensaje como 'inapropiado'.
INV	INVite	Utilizado para el envío de invitaciones a usuarios que aún no se han registrado en el servicio.

Los tipos de mensaje recibidos que pueden ser recibidos por la aplicación cliente (enviados por la aplicación servidora) son los siguientes

Tabla 1.2 - Tipos de mensaje recibidos por el cliente

TIPO	**COMANDO**	**DESCRIPCIÓN**
REG_OK	REGister OK	Utilizado para indicar que el registro se ha completado exitosamente.
REG_ER	REGister ERror	Utilizado para indicar que se ha producido un error en el registro.
CHN_OK	CHange Name OK	Utilizado para indicar que se ha cambiado el nombre de usuario correctamente.
CHS_OK	CHange Status OK	Utilizado para indicar que se ha cambiado el estado de usuario correctamente.
NCM	Normal Chat Message	Utilizado en el envío/recepción de un mensaje de una conversación privada.
ACM	Ack Chat Message	Utilizado para indicar a la aplicación que un mensaje privado ha llegado al servidor.
PTT	Push To Talk	Utilizado para indicar un mensaje entrante con una nota de audio.
PHT	PHoTo	Utilizado para indicar un mensaje entrante con una imagen.
PHT_ACM	PHoTo Ack Chat Message	Utilizado para indicar que una imagen ha sido enviada correctamente.

FIL	FILe	Utilizado para indicar un mensaje entrante con un archivo.
FIL_ACM	FILe Ack Chat Message	Utilizado para indicar que un archivo ha sido enviado correctamente.
CPC_SYN	Create Private Chat SYNcrhonize	Utilizado para iniciar el intercambio de claves para establecer una conversación privada cifrada extremo a extremo.
CPC_SYN_ACK	Create Private Chat SYNchronize ACKnowledgement	Utilizado para indicar la recepción del mensaje de inicio de intercambio de claves CPC_SYN.
CPC_ACK	Create Private Chat ACKnowledgement	Utilizado para indicar que la conversación privada extremo a extremo se ha completado.
CPC_FIN	Create Private Chat FINish	Utilizado para terminar una conversación privada extremo a extremo.
PCM	Private Chat Message	Utilizado en el envío/recepción de un mensaje cifrado extremo a extremo de una conversación privada.
CGT_ADD	Chat Group Topic ADD member	Utilizado para indicar que el usuario ha sido añadido a un grupo.
CGT_ADD_ADMIN	Chat Group Topic ADD ADMINistrator	Utilizado para indicar al creador de un grupo la creación del grupo.
CGT_PRT	Chat Group Topic PaRTicipants update	Utilizado para actualizar la lista de participantes de un grupo.
CGT_MSG	Chat Group Topic MeSsaGe	Utilizado para la recepción de un mensaje de grupo.
CGT_DEL	Chat Group Topic DELete	Utilizado para indicar el cierre de un grupo.
CGT_PHT	Chat Group Topic PHoTo	Utilizado para indicar mensaje de grupo con una imagen.
CGT_FIL	Chat Group Topic FILe	Utilizado para indicar mensaje de grupo con un archivo.

CGT_PTT	Chat Group Topic Push To Talk	Utilizado para indicar mensaje de grupo con nota de audio.
TGNA	Topic Group New Admin	Utilizado para indicar que el administrador de un grupo ha sido modificado.
TGNM	Topic Group New Moderator	Utilizado para indicar que el moderador de un grupo ha sido modificado.
TGI	Topic Group message Inappropiate	Utilizado para indicar que un mensaje de un grupo ha sido marcado como inapropiado.
TGMC	Topic Group iMage Change	Utilizado para actualizar la imagen de un grupo.
CGT_ACM	Chat Group Topic Ack Chat Message	Utilizado para indicar a la aplicación que un mensaje de grupo ha llegado al servidor.
INV_SNT	INVite SeNT	Utilizado para indicar a la aplicación que las invitaciones han sido enviadas a sus destinatarios.
CON_TC	CONtacts update	Utilizado para actualizar la lista de contactos.
CON_GR	CONtacts GRoup	Utilizado para comprobar la lista de participantes de un grupo con la lista de contactos local del usuario.

1.7.1.4 Cálculo del hash

En este punto, se calcula el hash SHA1 de los campos concatenados hasta este momento. De esta forma, en caso de error en la transmisión o intento de alteración del mensaje, con el hash calculado se podrá detectar cualquier alteración.

Figura 1.15 - Cálculo del hash

Una vez calculado el hash, se concatenará al final, quedando como aparece en la imagen inferior:

Figura 1.16 - Encapsulación del cuerpo del paquete

1.7.1.5 Segundo cifrado AES

Una vez que se tiene el cuerpo del mensaje, se cifrará con AES utilizando, en este caso, una clave AES generada aleatoriamente en cada envío. Posteriormente, esta clave se indicará al destinatario en la cabecera del mensaje.

Figura 1.17 - Cuerpo del mensaje cifrado

1.7.1.6 Cabecera del paquete

En la cabecera del mensaje se enviará el identificador del emisor del mensaje, la fecha en que se envía el mensaje y la clave AES generada aleatoriamente para cifrar el cuerpo del mensaje.

Figura 1.18 - Campos cabecera

Posteriormente, para asegurar la privacidad en el envío, la cabecera se cifrará utilizando la clave pública RSA del destinatario, de forma que sea él el único capaz de descifrar la cabecera, y así obtener la clave de cifrado del cuerpo del mensaje y posteriormente obtener los datos contenidos en él.

Figura 1.19 - Cabecera cifrada

1.7.1.7 Paquete enviado por la red

El paquete resultante que se enviará por la red, compuesto por la cabecera y el cuerpo detallados anteriormente tendrá la siguiente forma:

Figura 1.20 - Paquete enviado por la red

1.7.2 Escenarios de comunicaciones

Dependiendo del tipo de mensaje se procesará y se enviarán unos mensajes de respuesta de distinta forma. En este apartado se describen los escenarios que pueden darse.

1.7.2.1 Registro en el sistema

El primero de los escenarios posibles en el sistema es el dado en el proceso de registro de un usuario. Los pasos mostrados en la figura son los siguientes:

1. La aplicación envía la información de registro del usuario.
2. El servidor comprueba en la base de datos si el usuario ya existe y realiza las inserciones o actualizaciones oportunas. Se genera un código de verificación y se amacena en la base de datos.
3. El servidor llama API de Twilio para enviar un SMS al número de teléfono indicado por el usuario con el código de verificación.
4. Twilio envía el SMS al número de teléfono indicado por el usuario.
5. La aplicación lee el SMS, obtiene el código de verificación y lo comprueba con el servidor.
6. El servidor comprueba el código recibido con el almacenado.
7. El servidor envía la respuesta del proceso de verificación a través de GCM.
8. El servidor de GCM envía el mensaje al terminal.

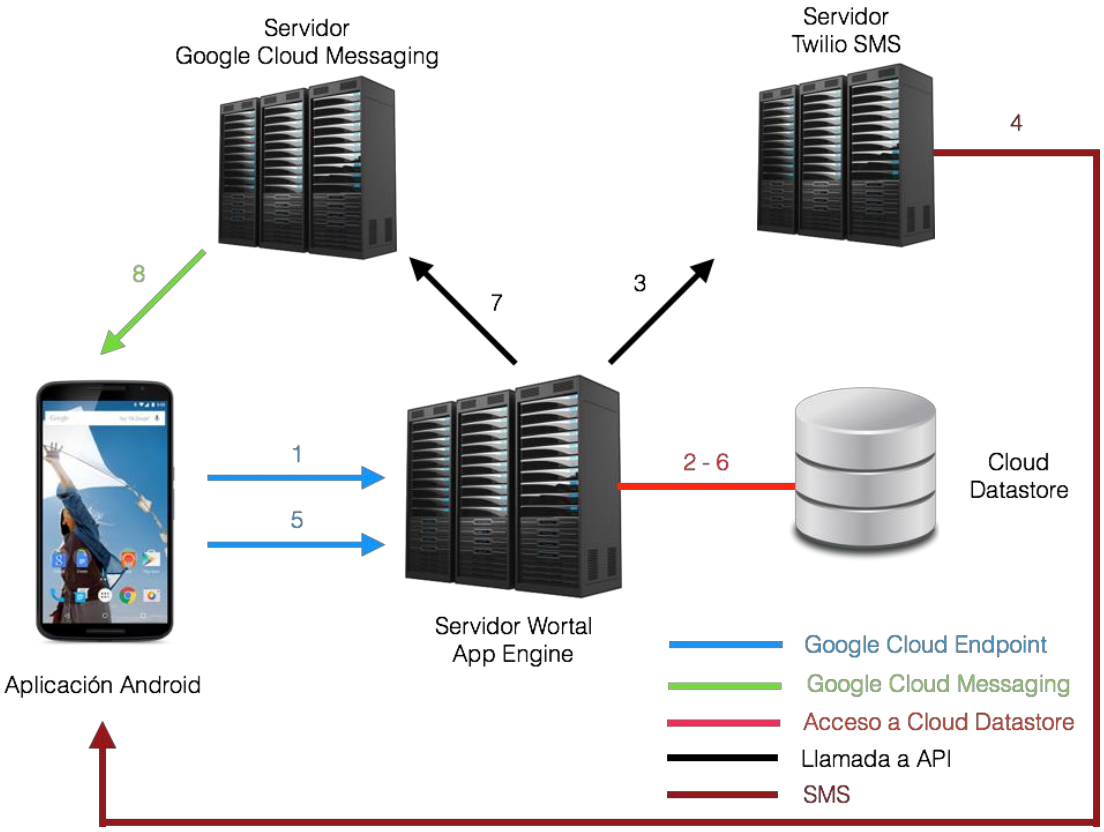


Figura 1.21 - Escenario registro usuario

Si todo el proceso se realiza correctamente, a partir del paso 8, el usuario ya podrá empezar a usar el servicio sin ningún tipo de problema. En caso de que el SMS no llegue al terminal (paso 4) desde donde se generó la petición de registro, no se podrá usar el servicio.

1.7.2.2 Actualización de perfil y Actualización de contactos

Otro de los escenarios que se puede dar en el servicio es el mostrado en la siguiente figura. Se presenta en los siguientes tipos de mensajes:

- Actualización de imagen de perfil.
- Actualización de nombre.
- Actualización de estado.
- Actualización de contactos.
- Envío de invitaciones a registrarse en el servicio: adicionalmente se llamaría a la API de Twilio que enviaría un SMS a los contactos seleccionados.

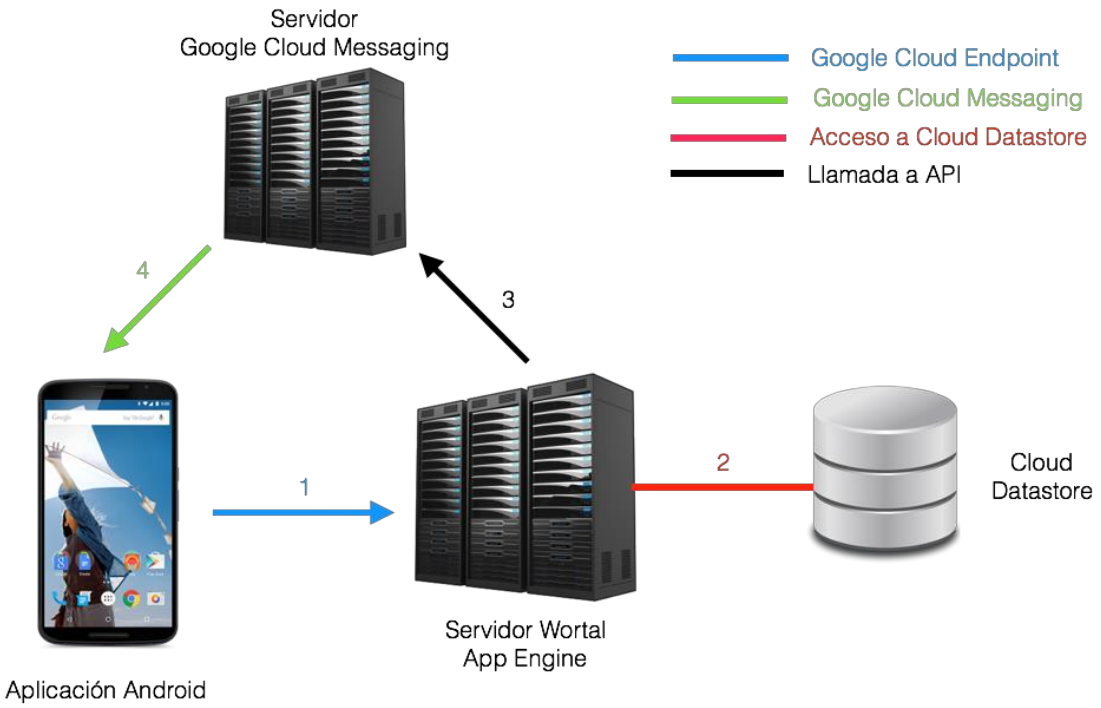


Figura 1.22 - Escenario actualización perfil y actualización contactos

1. La aplicación cliente envía al servidor, utilizando GCE, el mensaje.
2. El servidor, en función del tipo de mensaje indicado, realizará las acciones necesarias en la base de datos y realizará el procesado correspondiente.
3. El servidor llamará a la API de GCM para enviar la respuesta a la aplicación cliente a través de GCM.
4. Si el terminal está conectado a Internet, GCM enviará el mensaje, sino lo almacenará hasta que el terminal se conecte a Internet.

1.7.2.3 Conversación con un contacto

En este caso el servidor, además de enviar la respuesta al emisor del mensaje, reenviará el mensaje al destinatario correspondiente. Se presenta en los siguientes tipos de mensajes:

- Envío de mensaje en conversación privada.
- Envío de mensaje en conversación privada cifrada extremo a extremo.
- Envío de imagen en conversación privada.
- Envío de archivo en conversación privada.
- Envío de nota de audio en conversación privada.

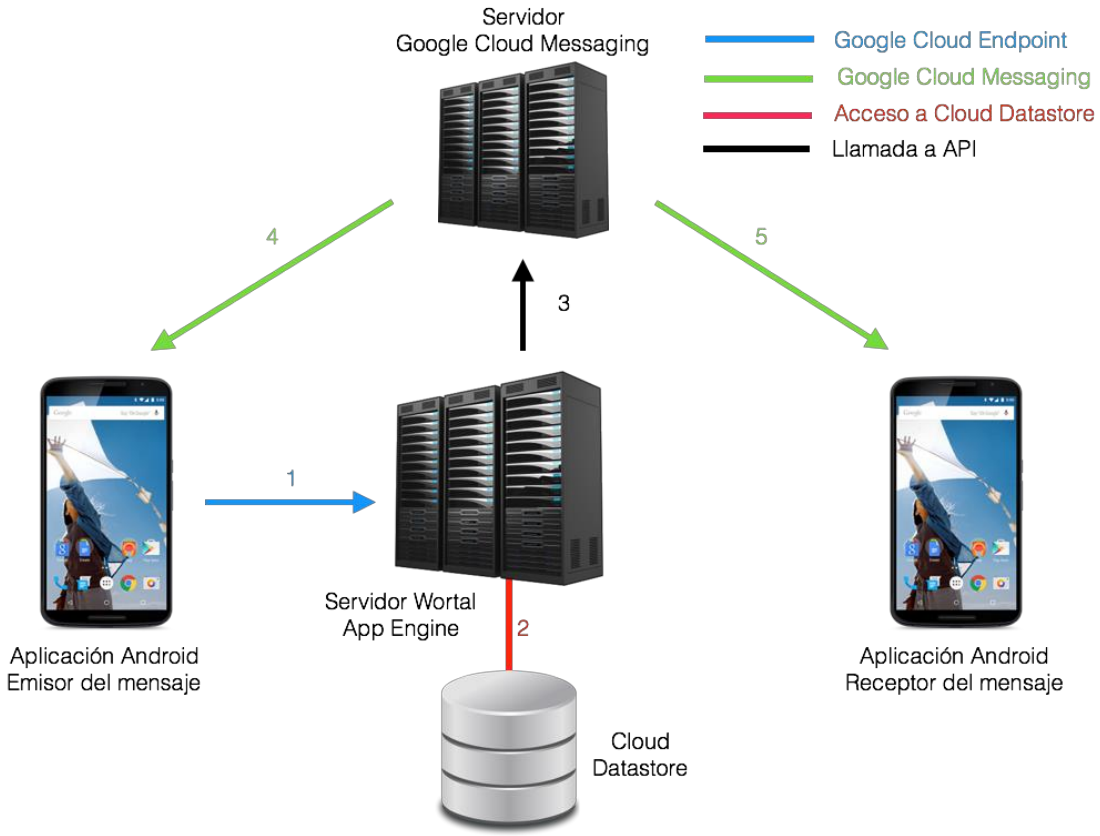


Figura 1.23 - Escenario conversación privada

1. La aplicación cliente envía al servidor, utilizando GCE, el mensaje.
2. El servidor, en función del tipo de mensaje indicado, realizará las acciones necesarias en la base de datos y realizará el procesado correspondiente.
3. El servidor llamará a la API de GCM para enviar la respuesta a la aplicación cliente a través de GCM.
4. Si el terminal está conectado a Internet, GCM enviará el mensaje de respuesta al emisor, sino lo almacenará hasta que el terminal se conecte a Internet.
5. Si el terminal está conectado a Internet, GCM reenviará el mensaje al destinatario, sino lo almacenará hasta que el terminal se conecte a Internet.

1.7.2.4 Conversación en grupo

El tercero de los escenarios es el mostrado en la siguiente figura, se da en todos los mensajes relacionados con grupos. En este caso el servidor, además de enviar la respuesta al emisor del mensaje, reenviará el mensaje al destinatario. Se presenta en los siguientes tipos de mensajes:

- Envío de mensajes en un grupo.

- Envío de archivos en un grupo.
- Envío de imágenes en un grupo.
- Envío de notas de audio en un grupo.
- Mensajes de control del grupo: mensajes del día, mensajes inapropiados, actualización de normas, actualización de imagen de perfil, actualización de participantes, actualización de rol de participantes.

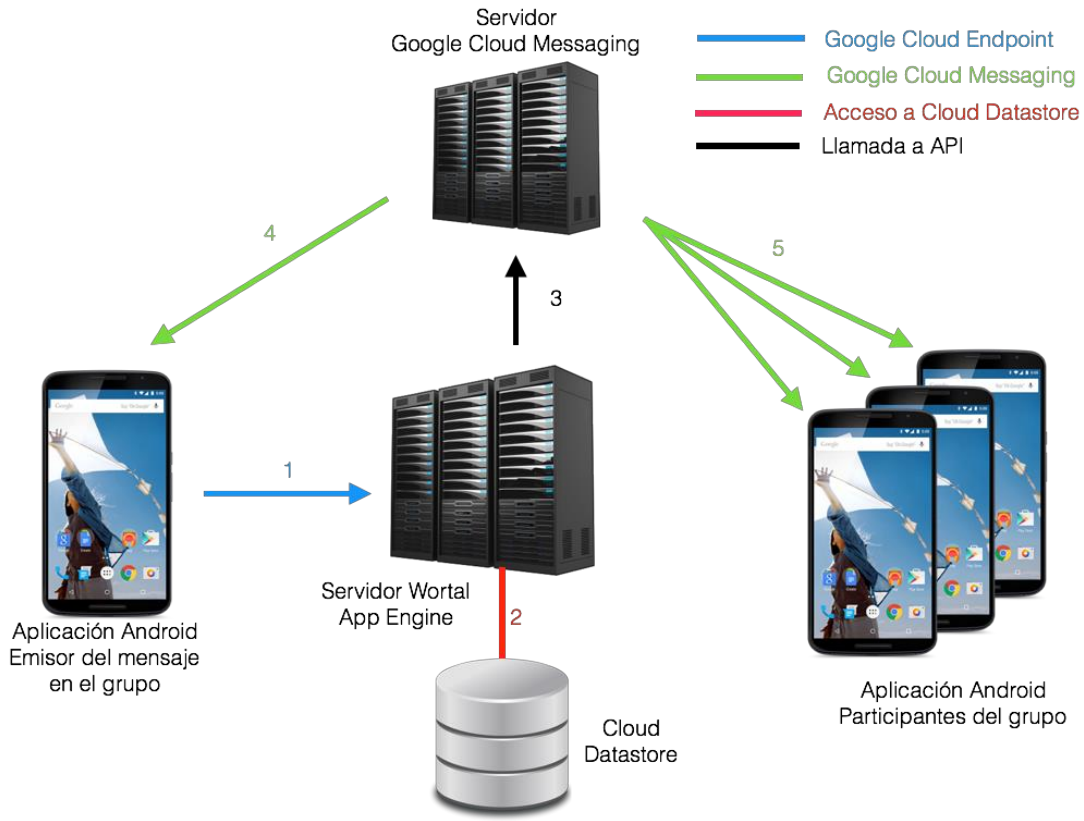


Figura 1.24 - Escenario conversación grupo

1. La aplicación cliente envía al servidor, utilizando GCE, el mensaje.
2. El servidor, en función del tipo de mensaje indicado, realizará las acciones necesarias en la base de datos y realizará el procesado correspondiente.
3. El servidor llamará a la API de GCM para enviar la respuesta a la aplicación cliente a través de GCM.
4. Si el terminal está conectado a Internet, GCM enviará el mensaje de respuesta al emisor, si no, lo almacenará hasta que el terminal se conecte a Internet.

5. Si los terminales están conectados a Internet, GCM reenviará el mensaje a todos los participantes del grupo, si no, lo almacenará hasta que se conecten a Internet.

1.8 Descripción de la aplicación servidora

En este apartado se describirán los aspectos más relevantes de la aplicación servidora hospedada y ejecutada en Google App Engine.

1.8.1 Estructura de clases

Los paquetes y clases son los mostrados en la figura inferior. En la carpeta *libs* se han añadido las bibliotecas SpongyCastle en formato *jar* para el correcto funcionamiento del cifrado RSA.

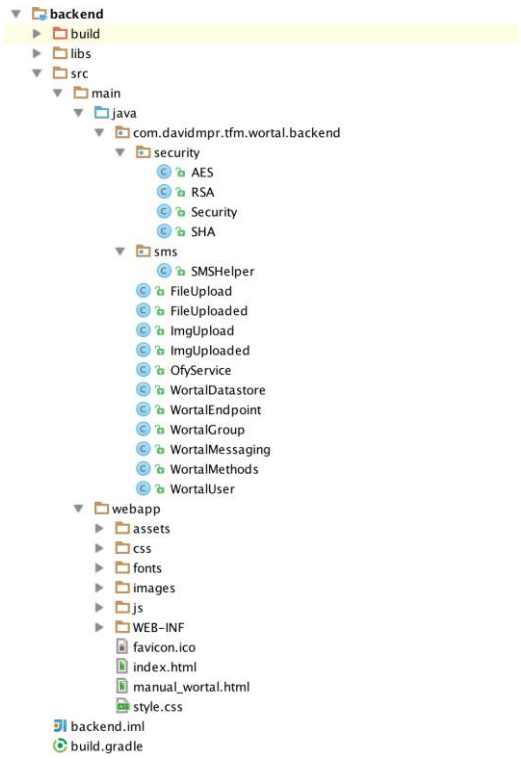


Figura 1.25 - Estructura módulo 'backend'

El módulo '*backend*' contiene el código de la aplicación servidora que se hospedará en GAE.

El paquete *java* contiene todos los paquetes y clases con el código necesario para el funcionamiento del servicio. Las clases más importantes son:

- Security: posee el método para encapsular y cifrar un mensaje con destino la aplicación cliente.
- WortalEndpoint: punto de entrada con el método que invocará la aplicación cliente a través de GCE.
- WortalMethods: clase con los métodos que procesarán la información en función del tipo de mensaje.

El paquete *webapp* contiene los archivos desarrollados para la visualización del portal web de presentación del servicio. El portal puede accederse en pinchando en la URL: http://wortalapp.appspot.com/, proporcionado automáticamente por GAE al crear el proyecto.

1.8.2 Base de datos

La base de datos diseñada para ser usada por la aplicación servidora utilizando la tecnología Cloud Datastore consta de dos tablas. Utiliza la API Objectify, diseñada por Google específicamente para usar GCD en GAE. En el Anexo V se explica con detalle cómo funciona la API. Las tablas con sus respectivas columnas son las siguientes:

- WortalUser: almacena los datos personales de los profesores.

Tabla 1.3 - Tabla WortalUser

ID	Identificador único de usuario
clientaeskey	Clave AES usada por el cliente en el cifrado de mensajes
fecha	Fecha de registro del usuario
hash	Hash SHA1 del número de teléfono
img	URL para obtener imagen de perfil del usuario
imgkey	Identificador de la imagen, utilizada para borrar tras cambio
mfecha	Última fecha de modificación de perfil
phonenumber	Número de teléfono registrado por el usuario
publickey	Clave pública del cliente para cifrado RSA
regId	Identificador utilizado para el envío de mensajes con GCM
serveraeskey	Clave AES usada por el servidor en el cifrado de mensajes
status	Estado configurado por el usuario
verifycode	Código de verificación usado en el registro

- WortalGroup: almacena información de los grupos de usuarios creados.

Tabla 1.4 - Tabla WortalGroup

ID	Identificador único del grupo
name	Nombre del grupo
info	Información adicional del grupo
admin	Identificador del creador del grupo
moderator	Identificador del moderador del grupo
image	URL con la imagen del grupo
date	Fecha de creación del grupo
participants	Identificador de los participantes del grupo
participants_temp	Cadena con todos los participantes añadidos por el administrador. Cuando acepten la invitación al grupo se añadirán a la columna 'participants'

1.8.3 Envío de mensajes: del cliente al servidor

Como se ha mencionado anteriormente, la aplicación cliente se comunica con el servidor a través de la tecnología GCE.

En la siguiente imagen se muestra el código de la clase CommunicationAsyncTask[1] creada para manejar las comunicaciones de toda la aplicación.

El primer paso es crear el *builder* indicando la URL raíz donde se encuentra la aplicación servidora.

El siguiente paso es crear y encapsular el mensaje de acuerdo al protocolo llamando al método *encode* de la clase *Security* con los parámetros correspondientes.

Por último, se ejecutará el envío de la información utilizando el método del servidor que recibirá el mensaje, en este caso el método se llama *endpoint(String mensaje).*

En la clase de recepción de mensajes en el servidor es muy importante utilizar correctamente las "anotaciones" (indicaciones precedidas del símbolo @), ya que será la forma en que el entorno de desarrollo podrá compilar correctamente los métodos y podremos llamar al método "endpoint" del servidor desde el código de la aplicación cliente.

[1] Las clases que heredan de AsyncTask [16] permiten realizar tareas en segundo plano de forma que no bloqueen el hilo principal.

```java
public class CommunicationAsynctask extends AsyncTask<String, Void, String> {
    private Wortalendpoint cservice = null;
    private Context context;
    private String ms = null;

    private static final String SENDER_ID = "452030613331";

    public CommunicationAsynctask(Context co) {
        context = co;
    }

    @Override
    protected String doInBackground(String... arg0) {
        final String msg = arg0[0];
        final String msgtype = arg0[1];
        final String extrafield = arg0[2];

        cservice = null;
        Wortalendpoint.Builder builder = new Wortalendpoint.Builder(
                AndroidHttp.newCompatibleTransport(), new AndroidJsonFactory(), null)
                .setRootUrl("https://wortalapp.appspot.com/_ah/api/");
        cservice = builder.build();

        try {
            final SharedPreferences prefs = context
                    .getSharedPreferences("WRTLPRFS", Context.MODE_PRIVATE);
            String id = prefs.getString("ID", "id");
            String ph = SHA.hash(prefs.getString("NUMBER", "hash"));
            String aeskey = prefs.getString("CLIENT_AES_KEY","aeskey");
            Security s = new Security();
            ms = s.encode(id,ph,msg,aeskey,msgtype,extrafield,context);
            cservice.endpoint(ms).execute();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return msg;
    }

    @Override
    protected void onPostExecute(String msg) {

    }
}
```

Figura 1.26 - Clase app envío mensaje

```java
@Api(
        name = "wortalendpoint",
        version = "v1",
        namespace = @ApiNamespace(
                ownerDomain = "backend.wortal.tfm.davidmpr.com",
                ownerName = "backend.wortal.tfm.davidmpr.com",
                packagePath = ""
        )
)
public class WortalEndpoint {

    private static final Logger log = Logger.getLogger(WortalEndpoint.class.getName());

    @ApiMethod(name = "endpoint")
    /**
     * Point of entry of all messages from devices
     *
     * @param msg Coded message
     */
    public void endpoint(@Named("msg") String msg) throws Exception {
        WortalDatastore wd = new WortalDatastore();
        WortalMethods wm = new WortalMethods();

        //The first thing we have to do is obtain the header and the body of the coded message
        String[] m = msg.split("_");
        String header = m[0];
        String body = m[1];
```

Figura 1.27 - Clase servidor recepción mensaje

1.8.4 Envío de mensajes: del servidor al cliente

Cuando un usuario envíe un mensaje a otro usuario, a un grupo o un mensaje de actualización de perfil, el servidor generará unos mensajes de respuesta que se enviarán a sus destinatarios a través del servicio GCM. Para ello es necesario obtener previamente un código identificador asociado a cada uno de los terminales. Este código es proporcionado por el servidor de GCM y deberá ser comunicado al servidor del servicio Wortal para su posterior almacenamiento durante el proceso de registro de usuario.

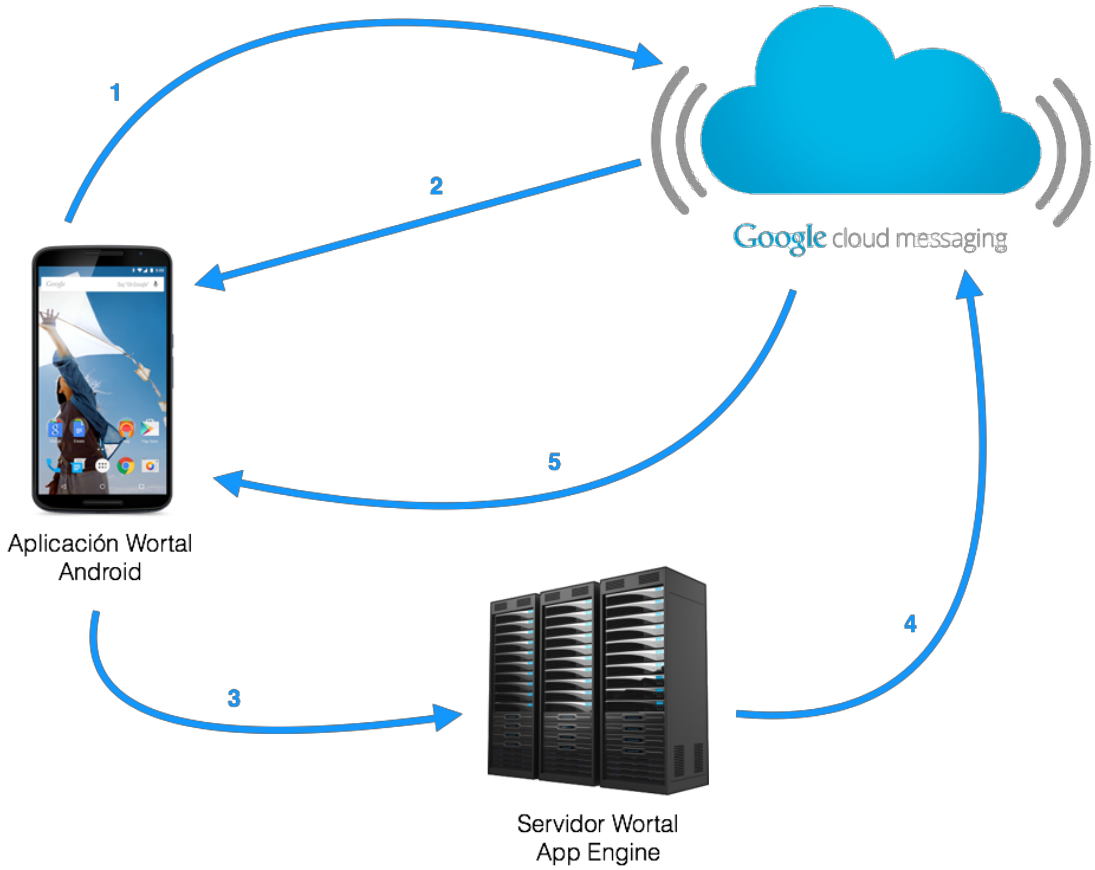


Figura 1.28 - Proceso registro en GCM y envío de mensajes a través de GCM

1. La aplicación se registra en los servidores de GCM enviando los valores 'Sender ID' (proporcionado por Google al activar el API) y 'App ID' (nombre del paquete de la aplicación).
2. Si el registro es satisfactorio se devuelve a la aplicación un 'Registration ID' (identificador único del dispositivo móvil).
3. La aplicación envía al servidor el 'Registration ID' en el proceso de registro de usuario.

4. Cuando haya algún mensaje, el servidor Wortal envía al servidor GCM el mensaje, los 'Registration ID' de los móviles a los que va a enviar el mensaje y el identificador de 'Sender', conocida como 'API Key'. El servidor GCM mantiene el mensaje en cola hasta que el terminal pueda recibirlo.
5. El mensaje es enviado a cada dispositivo.

```
public class WortalMessaging {

    private static final Logger log = Logger.getLogger(MessagingEndpoint.class.getName());
    private static final String API_KEY = System.getProperty("gcm.api.key");


    public void sendMessage(String message, String to) throws IOException {
        if (message == null || message.trim().length() == 0) {
            log.warning("Not sending message because it is empty");
            return;
        }

        Sender sender = new Sender(API_KEY);
        Message msg = new Message.Builder().addData("message", message).build();

        Result result = sender.send(msg, to, 5);
        if (result.getMessageId() != null) {
            log.info("Message sent to " + to);
            String canonicalRegId = result.getCanonicalRegistrationId();
            if (canonicalRegId != null) {
                // if the regId changed, we have to update the datastore
                log.info("Registration Id changed for " + to + " updating to " + canonicalRegId);
                WortalDatastore wd = new WortalDatastore();
                WortalUser u = wd.findUserByRegId(to);
                u.setRegId(canonicalRegId);
                ofy().save().entity(u).now();
            }
        } else {
            String error = result.getErrorCodeName();
            if (error.equals(Constants.ERROR_NOT_REGISTERED)) {
                // if the device is no longer registered with Gcm, remove it from the datastore
                WortalDatastore wd = new WortalDatastore();
                WortalUser u = wd.findUserByRegId(to);
                ofy().delete().entity(u).now();
            } else {
                log.warning("Error when sending message : " + error);
            }
        }
    }
}
```

Figura 1.29 - Clase servidor WortalMessaging.java

En la aplicación servidora se ha creado una clase llamada *WortalMessaging.java* (mostrado en la figura superior) con el método necesario para enviar el correspondiente mensaje a su correspondiente destinatario.

En la aplicación cliente se ha creado un servicio[2] que será el encargado de recibir los mensajes enviados por el servidor GCM.

Después de realizar ciertas comprobaciones se llamará al método *processIncomingMessage(message)* y se descifrará el mensaje siguiendo el protocolo de seguridad establecido. Si todo el proceso se realiza correctamente, el mensaje se

[2] Un servicio [17] es un componente de una aplicación que se ejecuta en segundo plano sin necesidad de tener ninguna interfaz. Puede realizar las mismas acciones que una Actividad.

procesará de la forma que corresponda según el tipo de mensaje indicado en el campo correspondiente.

```
public class GCMIntentService extends IntentService {
    public static final int NOTIFICATION_ID = 1;
    public static String nn,nm;
    public static boolean nc;
    private NotificationManager mNotificationManager;
    NotificationCompat.Builder builder;
    private int counter;

    public GCMIntentService() { super("GcmIntentService"); }


    @Override
    protected void onHandleIntent(Intent intent) {
        Bundle extras = intent.getExtras();
        GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(this);
        // The getMessageType() intent parameter must be the intent you received
        // in your BroadcastReceiver.
        String messageType = gcm.getMessageType(intent);

        if (extras != null && !extras.isEmpty()) {  // has effect of unparcelling Bundle
            // Since we're not using two way messaging, this is all we really to check for
            if (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equals(messageType)) {
                String message = extras.getString("message");
                try {
                    //Process the incoming message in function of message type
                    processIncomingMessage(message);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        GCMBroadcastReceiver.completeWakefulIntent(intent);
    }
```

Figura 1.30 - Servicio recepción mensajes GCM

En el Anexo III se explica con más detalle los pasos necesarios para desplegar el servicio Google Cloud Messaging desde cero.

1.8.5 Portal web

Una de las características de GAE, es que al crear un proyecto, asigna una URL desde la cual puede accederse para crear una *landing page responsive* utilizando Bootstrap [19], por lo que permite la correcta visualización en cualquier tipo de dispositivo, ya que Bootstrap se basa en el 'mobile-first', es decir, diseñar pensando primero en los terminales móviles. El portal puede accederse desde la URL: wortalapp.appspot.com

En la imagen inferior se muestra el aspecto de dicho portal web.

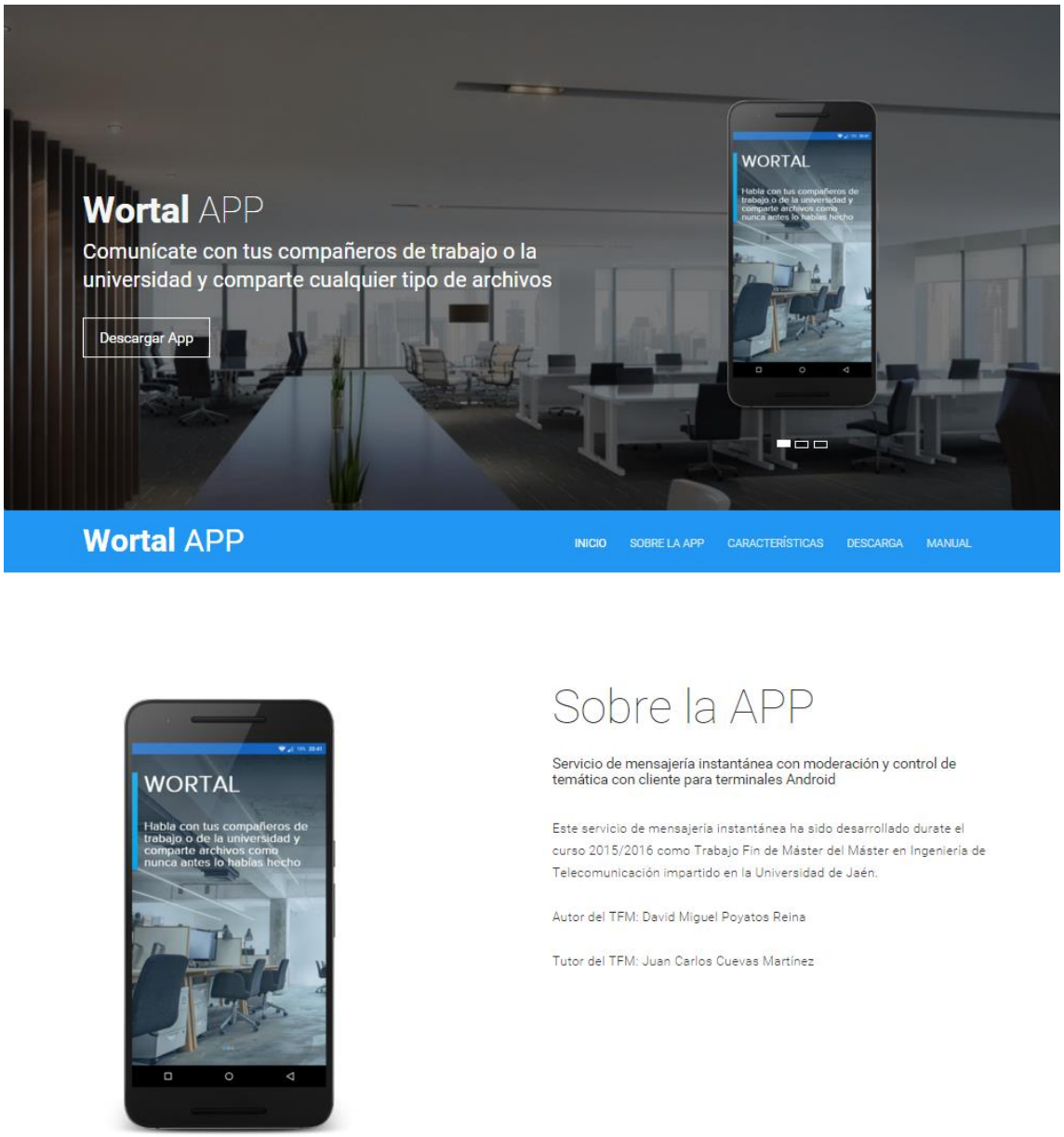


Figura 1.31 - Portal web Wortal

1.9 Descripción de la aplicación móvil.

La aplicación móvil es la herramienta necesaria para que los usuarios puedan usar el servicio de mensajería instantánea. Las funciones implementadas son las siguientes:

- Registro en el servicio.
- Actualización de perfil (imagen, nombre y estado mostrado a los contactos).
- Comprobación de la lista de contactos.
- Conversación con un contacto.
- Conversación con un contacto cifrada extremo a extremo (previo intercambio seguro de claves).
- Envío de archivos de cualquier tipo.
- Realización y envío de fotografías.
- Realización y envío de notas de audio.
- Creación de grupos de usuarios (se define el nombre de grupo, asunto, imagen, participantes y moderador).
- Conversación en grupo.
- Envío de archivos, fotografías y notas de audio en grupo.
- Funciones de moderación para el moderador seleccionado para el grupo (establecimiento de mensaje del día, mensajes inapropiados, cambio de normas, cambio de imagen de grupo y cierre del grupo).
- Realización de ajustes en la aplicación.
- Envío de invitaciones a usuarios que aún no usan el servicio.

Además, la aplicación está diseñada de forma que su uso sea fácil e intuitivo, para ello se ha desarrollado siguiendo los patrones de diseño de *Material Design* [18] utilizando bibliotecas de soporte para asegurar su compatibilidad y el correcto funcionamiento en el mayor número de dispositivos.

1.9.1 Bibliotecas utilizadas

En los siguientes puntos se mencionarán las bibliotecas utilizadas en el desarrollo de la aplicación y su función.

1.9.1.1 Bibliotecas oficiales de Google

Se hace uso de las bibliotecas oficiales que Google pone a disposición de los desarrolladores para hacer compatibles los últimos elementos visuales (introducidos en

Material Design) en versiones antiguas y para hacer uso de servicios como el de mensajería Cloud Messaging. Entre ellas se encuentran:

- support-v4: biblioteca de soporte que permite utilizar algunas de las APIs más recientes de Android en aplicaciones que se instalen en dispositivos con versiones antiguas.
- appcompat-v7: esta biblioteca añade soporte para mostrar un Action Bar introducido en la versión 3.0 o Toolbar introducido en la versión 5.0. Esta biblioteca depende de la anterior, support-v4.
- design: contiene los elementos de Material Design de forma que puedan usarse en versiones antiguas de Android.
- cardview: contiene el elemento 'cardview', compatible con versiones antiguas de Android.
- recyclerview: contiene un nuevo elemento introducido en las últimas versiones de Android para la creación de listas. Sustituye a 'ListView'.
- play-services-gcm: biblioteca utilizada para poder usar el servicio GCM.

1.9.1.2 Biblioteca de seguridad

Para poder cifrar correctamente usando RSA, se debe usar una biblioteca adicional, ya que la biblioteca que se usa por defecto en Android genera una serie de problemas, por lo que para evitar estos problemas, existe una biblioteca desarrollada/empaquetada para su uso en Android por rtyley llamada "Spongy Castle", basada en la existente "Bouncy Castle", una API de bajo nivel optimizada para gestionar eficientemente los algoritmos criptográficos de forma que se puedan usar en entornos de bajos recursos.

1.9.1.3 Bibliotecas HTTP

Para poder hacer uso de los Google Cloud Endpoints en la comunicación aplicación-servidor es necesario hacer uso de una serie de bibliotecas http de Apache debido a que en la última versión de Android, las bibliotecas que en versiones anteriores si venían por defecto, ahora han sido eliminadas, por lo que si el desarrollador utiliza como versión de compilación la última versión, deberá indicar explícitamente que va a usar las bibliotecas antiguas incorporándolas al proyecto.

1.9.1.4 MaterialTextField

Esta biblioteca [20] adicional se utiliza para poder incluir un campo de texto con la apariencia de los campos de texto de Material Design. El campo de texto posee la característica de que el texto de ayuda que se muestra en su interior, al pulsarse para poder escribir, se desplazará hasta la parte superior del campo con un elegante efecto.

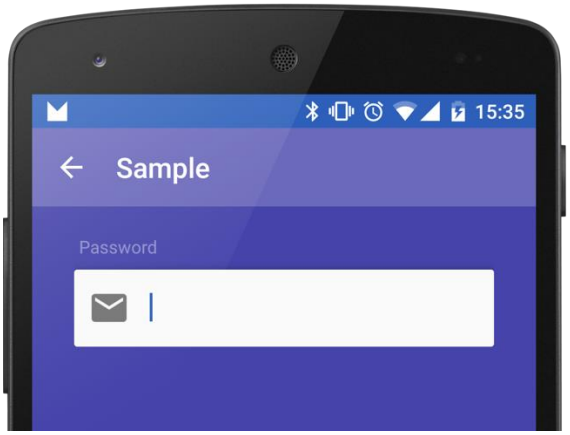


Figura 1.32 – Campo de texto de ejemplo de la biblioteca MaterialTextField

1.9.1.5 NineOldAndroids

Esta biblioteca [21] adicional se usa para obtener la API de animación incluida en Android 3.0 en versiones anteriores. Aunque no se usa de forma explícita, ha sido necesaria su incorporación al proyecto por requisito de otra biblioteca.

1.9.1.6 AndroidProcessButton

Esta biblioteca [22] adicional se usa para obtener un botón, que se usa en la pantalla de registro, el cual muestra un indicador de progreso de forma elegante. La barra de progreso puede configurarse de forma que vaya de 0 a 100 % o sea sin fin, es decir, la barra de progreso no terminará su animación hasta que se le indique.

Figura 1.33 – Botón de ejemplo de la biblioteca AndroidProcessButton

1.9.1.7 MaterialDialog

Esta biblioteca [23] adicional se usa para obtener los diálogos con la apariencia de Material Design de forma que sean compatibles con todas las versiones de Android.

De esta forma, el desarrollador puede asegurarse que la apariencia de los diálogos que se muestren en la aplicación será exactamente igual en todas las versiones y en todos los terminales Android.

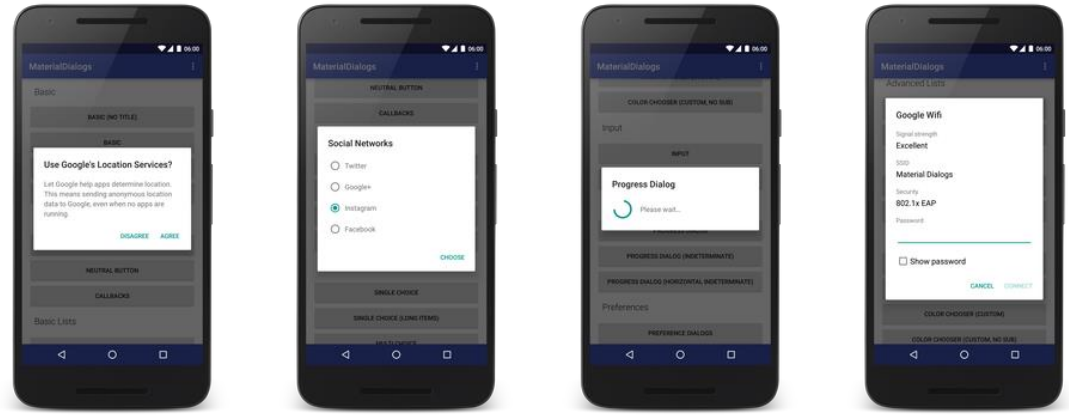


Figura 1.34 - Diálogos de la biblioteca MaterialDialog

1.9.1.8 Picasso

Esta biblioteca [24] adicional se usa para mostrar imágenes de la forma más eficiente posible, de forma que la memoria no se sature y se generen problemas que causen cierres de la aplicación, tal y como sucedería si las imágenes se cargasen de forma habitual. El uso de esta biblioteca (u otras similares) es aconsejable y muy recomendable cuando se trabajen con muchas imágenes de forma simultánea, como sucede cuando se carga la lista de conversaciones o la lista de contactos, ya que en cada uno de los elementos se muestra la imagen de perfil establecida.

Además, aunque no se usa en la aplicación, la biblioteca permite cargar las imágenes directamente desde Internet, usando su URL, sin que sea necesario su descarga, aunque no obstante, la velocidad de carga va a ser inferior que si la imagen se encontrase descargada ya en el dispositivo.

Otra característica importante es que en caso de que la imagen configurada no se encontrase, se podrá establecer otra imagen que se mostrará en caso de error, de esta forma la vista donde se mostrará la imagen nunca se quedará en blanco.

1.9.1.9 AndroidCrop

Esta biblioteca [25] adicional se usa para establecer la imagen de perfil del usuario. Cuando escoja una imagen de su galería, ésta se recortará según seleccione el usuario de la forma en que se muestra en la imagen inferior:

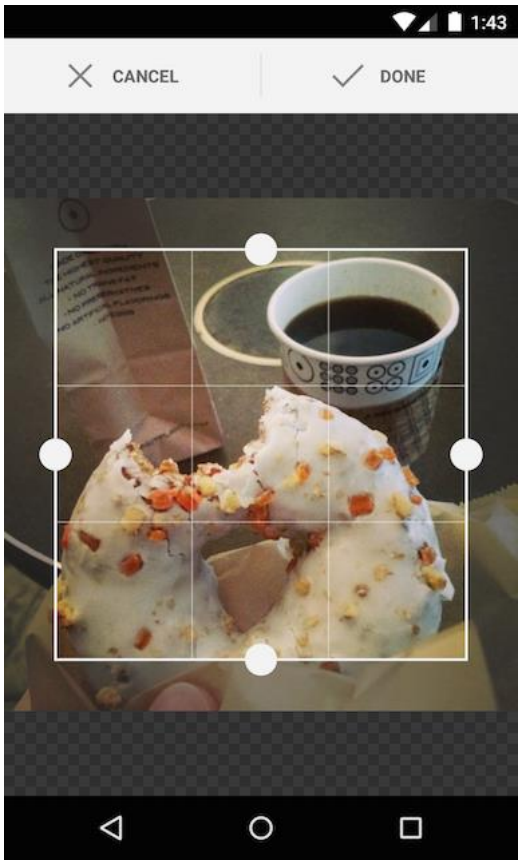


Figura 1.35 - Ejemplo para recortar imagen con la biblioteca AndroidCrop

1.9.1.10 AndroidUploadService

Esta biblioteca [26] adicional se usa para subir archivos al servidor. En la imagen inferior se muestra un código de ejemplo para subir un archivo. Algunos de los parámetros a indicar son:

- Identificador de carga: es un número para identificar diversas cargas en paralelo.
- URL de subida: URL donde se enviará el archivo.
- URI del archivo: localización del archivo en el dispositivo.
- Nombre de parámetro: nombre que se utilizará para obtener el archivo en el servidor.
- Cabecera/parámetros adicionales: pueden añadirse más parámetros a la carga.

```
public void uploadMultipart(final Context context) {

    final String uploadID = UUID.randomUUID().toString();
    final String serverUrlString = "http://www.yoursite.com/yourscript";

    try {
        new MultipartUploadRequest(context, uploadID, serverUrlString)
            .addFileToUpload("/absolute/path/to/your/file", "your-param-name")
            .addHeader("your-custom-header-name", "your-custom-value")
            .addParameter("your-param-name", "your-param-value")
            .setNotificationConfig(new UploadNotificationConfig())
            .setMaxRetries(2)
            .startUpload();
    } catch (Exception exc) {
        Log.e("AndroidUploadService", exc.getMessage(), exc);
    }
}
```

Figura 1.36 - Código de ejemplo AndroidUploadService

1.9.1.11 CircularReveal

Esta biblioteca [27] adicional se usa para obtener un efecto de aparición circular introducido en Material Design a la hora de abrir un menú desplegable. Se utilizar para desplegar el menú para enviar archivos, contactos o la localización actual del usuario.

1.9.1.12 PullRefreshLayout

Esta biblioteca [28] adicional se usa para actualizar una lista simplemente deslizando desde la parte superior de la pantalla. Automáticamente se mostrará un indicador de progreso circular indicando al usuario que la actualización está en progreso. Se utiliza para actualizar la lista de contactos.

1.9.1.13 FilePicker

Esta biblioteca [29] adicional se usa para permitir al usuario seleccionar un archivo dentro del directorio de archivos de su terminal. Permite elegir cualquier tipo de archivo, independientemente de su extensión o tamaño.

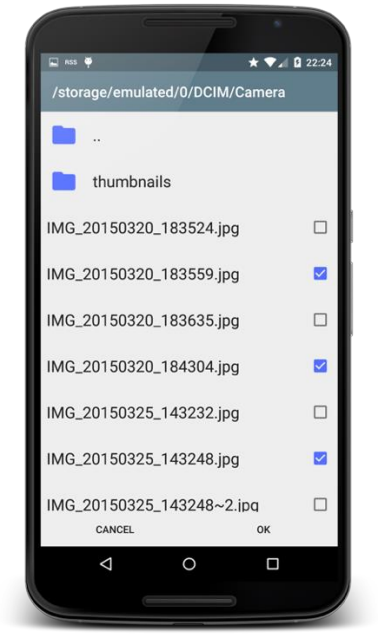


Figura 1.37 - Pantalla de ejemplo FilePicker

Para más información sobre las bibliotecas consultar la documentación *Javadoc* situada en la carpeta: DOCUMENTACIÓN/Javadoc_Wortal_app/index.html

1.9.2 Esquema de clases

En la siguiente figura se muestra el esquema del módulo *app* con el código fuente utilizado para desarrollar la aplicación cliente.

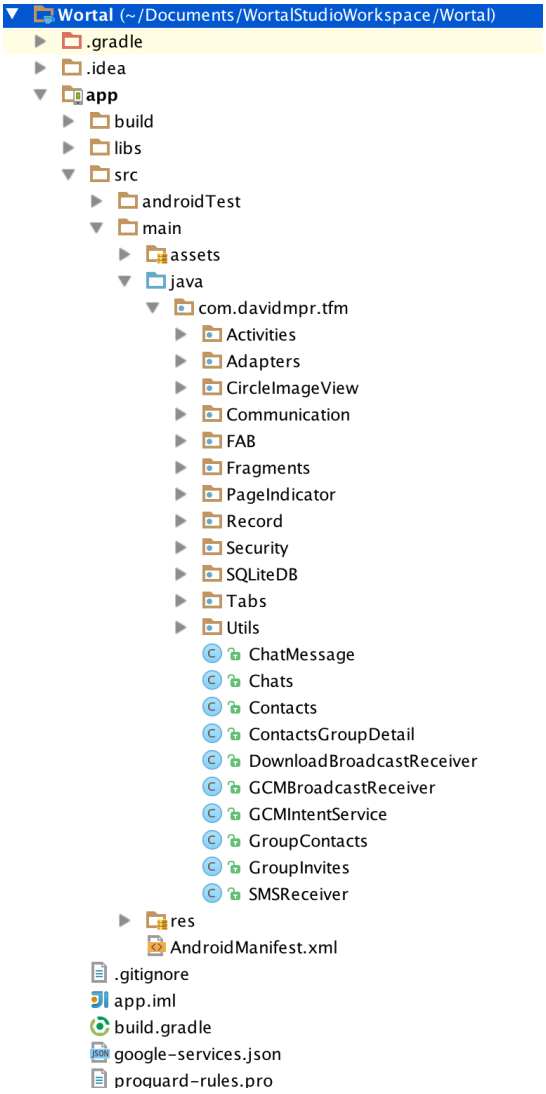


Figura 1.38 - Estructura módulo 'app'

El paquete **Activities** contiene las clases con todas las actividades usadas en la aplicación.

El paquete **Adapters** contiene las clases con los adaptadores utilizados para representar las listas de conversaciones, listas de mensajes y listas de contactos.

El paquete **Communication** contiene las dos clases utilizadas para establecer la comunicación a través de GCE con la aplicación servidora.

El paquete **Record** contiene las clases utilizadas para la grabación de notas de audio.

El paquete **Security** contiene las clases necesarias para la encapsulación y cifrado de mensajes enviados y recibidos del servidor.

El paquete **SQLiteDB** contiene la clase necesaria para la correcta configuración de la base de datos local de la aplicación.

El paquete **Utils** contiene varias clases usadas en otras partes del código.

Otras clases de interés son:

-GCMIntentService: clase que recibe los mensajes provenientes del servidor.

-SMSReceiver: clase que recibe el SMS de verificación enviado en el proceso de registro.

1.9.3 Permisos de la aplicación

Para poder ejecutar la aplicación y que todas las funciones implementadas puedan ser ejecutadas correctamente, la aplicación necesita una serie de permisos, por lo que al usuario deberá aceptar la utilización de dichos permisos al instalar la aplicación. Los permisos son los mostrados en la siguiente figura:

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="com.davidmpr.tfm.permission.C2D_MESSAGE" />
```

Figura 1.39 - Permisos de la aplicación

1.9.4 Registro de usuarios

El primer paso para hacer uso del servicio es el registro en el servidor. Para ello, la aplicación requerirá al usuario su número de teléfono móvil. El servidor comprobará si el número existe ya o no en la base de datos y en función del resultado de la comprobación enviará al usuario un SMS con un código de verificación. Esta verificación se realiza para comprobar que el número de teléfono introducido se corresponde con el terminal desde el que se realiza el registro. Una vez que el SMS es recibido por el terminal, la aplicación Wortal obtendrá el código que aparece en el SMS y lo comprobará con el servidor. Si la comprobación es satisfactoria se enviará un mensaje de respuesta con el tipo de mensaje *REG_OK* y el usuario podrá empezar a usar la aplicación.

En este proceso se generan las claves AES y RSA que se utilizarán para el futuro intercambio de mensajes con el servidor.

En la imagen inferior se muestra el inicio de proceso de registro que se realiza cuando el usuario pulsa el botón para registrarse. Al pulsar el botón se llama al método *enter(), donde,* en primer lugar, se comprueba que el campo de número de teléfono no esté vació, en caso de no estarlo, se sigue con el proceso, obteniendo las parejas de claves RSA e iniciando la tarea asíncrona *GcmRegistrationAsyncTask*.

```
public void enter (View view){
    number = et2.getText().toString();
    if(number.equals("")){
        b1.setProgress(0);
        b1.setText("ESCRIBE TU NÚMERO");
        return;
    }
    final SharedPreferences prefs = getSharedPreferences("WRTLPRFS", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putString("NUMBER",number);
    editor.commit();

    b1.setMode(ActionProcessButton.Mode.ENDLESS);
    b1.setProgress(1);

    String registrationId = prefs.getString("REG_ID", "");
    int reg = prefs.getInt("REG", 0);
    String regenv = prefs.getString("REGENV","0");

    if (registrationId.isEmpty()) {
        if(regenv.equals("0")) {
            //Generate RSA Keypair with client keys -> The public key will be sent to the serve
            RSA rsa = new RSA();
            Security sec = new Security();
            KeyPair kp = rsa.generate();
            Preferences.init(context);
            sec.writeClientPrivateKeyToPreferences(kp);
            publickey = sec.writeClientPublicKeyToPreferences(kp);

            new GcmRegistrationAsyncTask(this).execute();
            editor.putString("REGENV", "1");
            editor.commit();
            startService(new Intent(this, SMSReceiver.class));
        }
    } else{
        if(reg==1) {
            b1.setProgress(100);
            stopService(new Intent(this, SMSReceiver.class));
            Intent j = new Intent(WelcomeActivity.this, MainActivity.class);
            this.finish();
            startActivity(j);
        }
    };

}
```

Figura 1.40 - Inicio proceso registro

```
class GcmRegistrationAsyncTask extends AsyncTask<Void, Void, String> {
    private static final String SENDER_ID = "452030613331";
    public GcmRegistrationAsyncTask(Context context) { this.context = context; }
    private GoogleCloudMessaging gcm;
    private Context context;
    String clientaeskey = "";


    @Override
    protected String doInBackground(Void... params) {
        String msg = "";
        String SENDER_ID = "452030613331";
        Bitmap icon = BitmapFactory.decodeResource(
                context.getResources(), R.drawable.default_original_profile_pic);
        String img = IMGUtils.BitmapToString(icon);
        try {
            if (gcm == null) {
                gcm = GoogleCloudMessaging.getInstance(context);
            }
            //Generate client AES Key
            String elegibleChars = "ABCDEFGHJKLMPQRSTUVWXYabcdefhjkmnpqrstuvwxy23456789";
            char[] chars = elegibleChars.toCharArray();
            StringBuffer finalString = new StringBuffer();

            for ( int ij = 0; ij < 16; ij++ ) {
                double randomValue = Math.random();
                int randomIndex = (int) Math.round(randomValue * (chars.length - 1));
                char characterToShow = chars[randomIndex];
                finalString.append(characterToShow);
            }
            clientaeskey = finalString.toString();

            SharedPreferences prefs = getSharedPreferences("WRTLPRFS", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("CLIENT_AES_KEY", clientaeskey);
            editor.commit();
            //Obtain Google GCM ID and save it
            String regId = gcm.register(SENDER_ID);
            storeRegistrationId(context, regId, name, number, img, clientaeskey);
            //Get actual date
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
            String fecha = sdf.format(new Date());

            msg = regId+"@"+number+"@"+fecha+"@default@"+getString(R.string.default_status)
                    +"@"+publickey;
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        return msg;
    }

    @Override
    protected void onPostExecute(String msg) {
        new CommunicationAsynctask(context).execute(msg, "WUR", clientaeskey);
    }
}
```

Figura 1.41 - Tarea asíncrona proceso registro

En la clase *GcmRegistrationAsyncTask (mostrada en la figura superior)*, en primer lugar, se crea la instancia de Google Cloud Messaging para posteriormente registrarse en el servidor GCM y obtener el identificador (*String regId*).

En segundo lugar, se genera la clave AES que utilizará la aplicación para comunicarse con el servidor: 16 caracteres elegidos aleatoriamente de entre los disponibles dentro del *String elegibleChars*.

Posteriormente, se obtiene y se almacena el identificador GCM junto con otros datos como el número de teléfono y la clave AES generada.

Para finalizar, se obtiene la fecha actual y se forma el mensaje (tipo de mensaje WUR) que será enviado al servidor a través de la tare asíncrona *CommunicationAsynctask*.

```
//Response from server indicating register succesful
if (s1.equals("REG_OK") == true) {
    new Handler(Looper.getMainLooper()).post(() → {
            final SharedPreferences prefs = getSharedPreferences("WRTLPRFS",
                    Context.MODE_PRIVATE);
            final SharedPreferences.Editor editor = prefs.edit();
            editor.putString("SERVER_AES_KEY", s2);
            editor.putString("ID", s3);
            editor.putInt("REG", 1);
            editor.commit();
            WelcomeActivity.b1.setProgress(100);
            WelcomeActivity.b1.setText("CONTINUAR");
    });

}
//Response from server indicating register error
if (s1.equals("REG_ER") == true) {
    showToast("ERROR: " + s2 + " " + s3);
    WelcomeActivity.b1.setProgress(-1);
}
```

Figura 1.42 - Respuesta del servidor proceso registro

Si la respuesta del servidor es correcta, se almacenará la clave AES generada por el servidor y el identificador de usuario asignado por el servidor.

En caso de error, mensaje *REG_ER* se mostrará al usuario un mensaje de error.

1.9.5 Actualización de perfil de usuario

El usuario puede cambiar en cualquier momento tres parámetros de su perfil de usuario. Estos parámetros son:

- Imagen de perfil.
- Nombre (nombre que se mostrará a aquellos usuarios que no lo tengan en su agenda).
- Estado (breve frase que puede utilizar el usuario para indicar a los usuarios del servicio de cualquier eventualidad)

Para poder ver esta información y cambiarla, se ha creado un apartado en el cual el usuario puede realizar dichas actualizaciones de forma fácil e intuitiva pulsando ciertos botones claramente indicados.

1.9.5.1 *Actualización de la imagen de perfil*

Al pulsar el botón correspondiente para actualizar la imagen de perfil, se abrirá la pantalla para seleccionar imágenes que el dispositivo tenga por defecto. Una vez que el usuario haya seleccionado la imagen apropiada, se abrirá otra pantalla en la que se dará al usuario la opción de recortar la imagen de forma que se visualice correctamente en todos los dispositivos de todos los usuarios con la proporción adecuada.

```
edit_img.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);

        intent.putExtra("return-data", true);
        startActivityForResult(Intent.createChooser(intent, "Elige imagen"), SELECT_PHOTO);
    }
});
```

Figura 1.43 - Selección de imagen de perfil

Una vez recortada la imagen, la aplicación solicitará al servidor una URL de subida. Cuando el servidor responda con la URL, se iniciará el proceso de subida.

Ver ANEXO VI donde se detalla en profundidad todo el proceso.

1.9.5.2 *Actualización del nombre de usuario*

Al pulsar el botón correspondiente para actualizar el nombre de perfil, se abrirá un desplegable donde se invita al usuario a introducir su nombre. Una vez introducido el nombre y pulsado el botón de aceptación correspondiente se iniciará el proceso de comunicación (tipo de mensaje CPN) con el servidor a través de la clase asíncrona *CommunicationAsynctask*.

Si el cambio se produce correctamente se enviará un mensaje CHN_OK.

1.9.5.3 *Actualización del estado*

La actualización de estado se realiza de la misma forma que la actualización de nombre, con el cambio del tipo de mensaje y las indicaciones que se realizan al usuario en el cuadro desplegable.

```
edit_name.setOnClickListener((view) → {
        MaterialDialog builder = new MaterialDialog.Builder(ProfileActivity.this)
                .title("Introduce tu nombre")
                .widgetColor(getResources().getColor(R.color.colorPrimary))
                .inputMaxLength(30, getApplicationContext().getResources()
                        .getColor(R.color.colorPrimary))
                .inputType(InputType.TYPE_CLASS_TEXT)
                .input("Nombre", "", (dialog, input) → {
                        String ename = input.toString();

                        dialogg = new MaterialDialog.Builder(ProfileActivity.this)
                                .title("Cambiando nombre")
                                .content("Espere...")
                                .progress(true, 0)
                                .progressIndeterminateStyle(true)
                                .show();

                        String msg = ename;

                        new CommunicationAsynctask(ProfileActivity.this)
                                .execute(msg,"CPN","extrafield");


                }).negativeText("Cancelar").show();
});
```

Figura 1.44 - Actualización nombre de usuario

```
edit_status.setOnClickListener((view) → {
        MaterialDialog builder = new MaterialDialog.Builder(ProfileActivity.this)
                .title("Introduce tu estado")
                .widgetColor(getResources().getColor(R.color.colorPrimary))
                .inputMaxLength(30, getApplicationContext().getResources()
                        .getColor(R.color.colorPrimary))
                .inputType(InputType.TYPE_CLASS_TEXT)
                .input("Estado", "", (dialog, input) → {
                        String estatus = input.toString();

                        dialogg = new MaterialDialog.Builder(ProfileActivity.this)
                                .title("Cambiando estado")
                                .content("Espere...")
                                .progress(true, 0)
                                .progressIndeterminateStyle(true)
                                .show();

                        String msg = estatus;

                        new CommunicationAsynctask(ProfileActivity.this)
                                .execute(msg,"CPS","extrafield");

                }).negativeText("Cancelar").show();
});
```

Figura 1.45 - Actualización estado

1.9.6 Actualización de contactos

El proceso seguido para mostrar al usuario los contactos de su agenda que usan el servicio de Wortal es el siguiente:

1. Se almacenan/actualizan todos los contactos de la agenda en la base de datos de la aplicación (en caso de que ya existiera se comprueba si el nombre ha cambiado y se actualiza).
2. Se crea un mensaje con el hash del número de teléfono y la última fecha de actualización. Este mensaje se crea en bloques de 30 contactos, para evitar crear cadenas de texto excesivamente largas. El formato es el siguiente:

```
Hash1_Fecha1#Hash2_Fecha2#Hash3_Fecha3#.....#HashN_FechaN
```

3. Cuando el servidor reciba la cadena de texto, la dividirá e irá comprobando si el Hash indicado se corresponde con algún usuario registrado en la base de datos. En caso positivo, se enviará la información de perfil actualizada (imagen, estado y fecha de última actualización) encadenando la información de la misma forma en que se recibió. En caso de que el hash no se corresponda con el usuario no se enviará nada.
4. La base de datos de contactos es actualizada con los datos proporcionados por el servidor y las nuevas imágenes de perfil son reemplazadas.

1.9.7 Conversación con un usuario

Para empezar una conversación con un usuario, basta con que aparezca en la lista de contactos. Al seleccionarlo se abrirá una pantalla en la que se podrá introducir un mensaje en un campo de texto y al pulsar el botón se enviará al servidor a través de GCE. Cuando el mensaje llegue al servidor enviará al emisor un mensaje de confirmación ACM notificando que el servidor ha procesado el mensaje y enviará al destinatario el mensaje a través de GCM.

1.9.8 Intercambio claves para conversación cifrada extremo a extremo

Como seguridad añadida, se da al usuario la opción de cifrar la conexión extremo a extremo, de forma que solamente emisor y destinatario conozcan el contenido del mensaje.

Para ello, ambos participantes deben intercambiar las claves AES que utilizarán para cifrar los mensajes enviados.

Se utilizan tres tipos de mensajes:

- CPC_SYN: enviado por el usuario que quiere iniciar el establecimiento de una conexión cifrada. En ese mensaje el usuario envía cifrada con la clave pública del destinatario la clave AES con la que el destinatario debe cifrar los mensajes.
- CPC_SYN_ACK: enviado por el usuario que recibe el mensaje CPC_SYN. Como respuesta envía con la clave AES recibida en el mensaje anterior, la clave AES con la que deben cifrar los mensajes que le envíen.
- CPC_ACK: enviado en respuesta del mensaje CPC_SYN_ACK. Envía cifrado con la clave AES un mensaje de asentimiento con el que indica el establecimiento correcto de la conexión.

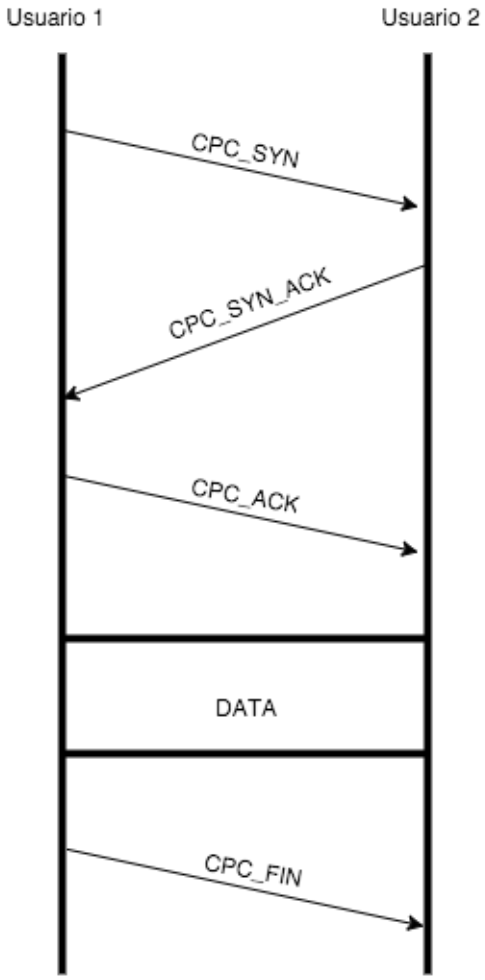


Figura 1.46 - Intercambio de claves

Cuando uno de los dos usuarios quiera terminal la conexión cifrada enviará un mensaje CPC_FIN y a partir de ese momento la conexión ya no estará cifrada extremo a extremo.

1.9.9 Envío de archivos

Para dar al usuario la opción de cualquier tipo de archivo, se utiliza una biblioteca adicional que permite incorporar dentro de la aplicación un explorador de archivos. Seleccionando el archivo que se quiera enviar, se obtendrá la URI local del archivo y será posible subirlo al servidor y así compartir el recurso con los usuarios con los que se quiera compartir.

El funcionamiento es similar al descrito para la actualización de imagen de perfil. Una vez obtenida la URI local, se solicita al servidor una URL de subida. Con la URL de subida y la información asociada al emisor y los destinatarios, el archivo se guarda en el almacenamiento del servidor y la URL asociada con el archivo es enviada a los destinatarios.

Ver ANEXO VI donde se detalla en profundidad todo el proceso.

1.9.10 Creación de grupos

Los usuarios pueden crear grupos de usuarios en cualquier momento, sin número límite de usuarios. El administrador del grupo siempre será el creador del grupo. Dicho administrador será quien elija en el momento de la creación del grupo la imagen identificativa del grupo, el nombre del grupo y una descripción sobre el asunto de discusión del grupo. Posteriormente se mostrará la lista de contactos y el administrador podrá seleccionar qué usuarios forman parte del grupo. Por último, se mostrará una pantalla con los participantes seleccionados y deberá seleccionar quién será el moderador del grupo. Toda esta información se enviará al servidor y se almacenará en la tabla WortalGroup.

Hecho todo esto, se enviará una invitación para participar en el grupo. Los receptores de la invitación podrán aceptarla y participar en el grupo o declinarla.

1.9.11 Conversación y envío de archivos en un grupo

El funcionamiento es igual que una conversación o un envío de archivo en una conversación privada, con la salvedad de que al cambiar el tipo de mensaje, se consultará la tabla WortalGroup para enviar el mensaje o el archivo a todos los participantes del grupo. Debido a la complejidad que supondría el intercambio de claves, no se permite el cifrado extremo a extremo en conversaciones grupales.

1.9.12 Envío de invitaciones a contactos que no usan el servicio

Para invitar a usuarios que no usan el servicio, se ofrece la posibilidad de enviar una invitación a los contactos de la agenda que no estén registrados en el servidor. Para ello se mostrará una lista con los contactos y se podrá seleccionar a qué usuarios enviar la invitación. Cuando el servidor reciba el mensaje, enviará un SMS invitando a los usuarios a descargar la aplicación.

1.10 Conclusiones

Una vez terminado el desarrollo del Trabajo Fin de Máster, se han cumplido todos los objetivos de forma satisfactoria:

- Se ha creado un servicio de envío de mensajes entre terminales Android utilizando la plataforma de Google.
- Se ha creado una aplicación Android que permite el envío y la recepción de mensajes entre los usuarios del servicio.
- Se ha creado un protocolo de seguridad que haciendo uso de diversos algoritmos, se garantiza la integridad y autenticidad en la comunicación.
- Se han implementado de forma satisfactoria todas las funciones indicadas en la propuesta del TFM:
    - Registro basado en invitaciones a través de los contactos del móvil.
    - Autenticación segura de usuarios.
    - Envío de mensajes privados.
    - Envío de mensajes a grupos.
    - Gestión de grupos de usuarios.
    - Envío de archivos de cualquier tipo.
    - Rol de moderador de grupo.
    - Posibilidad de establecer información adicional a un grupo: mensajes del día, mensajes inapropiados, normas, etc.
- Al utilizar bibliotecas de soporte, se asegura que la aplicación podrá ser empleada en el mayor número posible de terminales siempre y cuando cumplan los requisitos mínimos establecidos en el apartado '3. Pliego de Condiciones'.

1.11 Líneas futuras

El sistema queda abierto para futuras mejoras o cambios que se puedan realizar, con el objetivo de mejorar lo presente.

En primer lugar, la aplicación se podría desarrollar para el resto de sistemas operativos móviles, ya que aunque Android ocupa casi el 90% [30] del mercado, los usuarios utilizan otras plataformas móviles como iOS y Windows.

En segundo lugar se podrían añadir más funciones al servicio, como por ejemplo la posibilidad de establecer una llamada con un contacto o con un grupo a través de VoIP, utilizando el servicio Twilio, ya utilizado para el envío de SMS.

En tercer lugar, se podría desarrollar un cliente de escritorio para poder utilizar el servicio desde un ordenador. De esta forma se facilitaría la cooperación y la comunicación en ámbitos laborales y profesionales, pero en este caso, habría que hacer cambios también en el servidor, ya que en este caso habría que guardar los mensajes en el servidor, para que estos pudiesen ser consultados desde cualquier dispositivo.

En cuarto lugar, se podría añadir más funciones como la posibilidad de crear y enviar encuestas a los contactos o a grupos; posibilidad de interaccionar con ILIAS, pizarra virtual, etc.

1.12 Referencias

[1] Apps para *smartphones* más descargadas en España 2014 (última vez consultada Febrero 2016): http://es.statista.com/estadisticas/485899/aplicaciones-mas-populares-en-usuarios-espanoles-de-smartphone/

[2] Apps más descargadas en 2015 (última vez consultada Febrero 2016): http://blogthinkbig.com/cuales-son-las-apps-mas-descargadas/

[3] Google Cloud Platform (última vez consultada Febrero 2016): https://cloud.google.com/

[4] Google App Engine (última vez consultada Febrero 2016): https://platzi.com/blog/google-app-engine/

[5] Google Cloud Datastore (última vez consultada Febrero 2016): https://cloud.google.com/datastore/

[6] Google Cloud Storage(última vez consultada Febrero 2016): https://cloud.google.com/storage/

[7] Google Cloud Endpoint (última vez consultada Febrero 2016): https://cloud.google.com/endpoints/

[8] Google Cloud Messaging (última vez consultada Febrero 2016): https://developers.google.com/cloud-messaging/

[9] Rivest Shamir Adleman (RSA) (última vez consultada Febrero 2016): http://es.wikipedia.org/wiki/RSA
http://stackoverflow.com/questions/12471999/rsa-encryption-decryption-in-android

[10] SpongyCastle (última vez consultada Febrero 2016): http://rtyley.github.io/spongycastle/

[11] PKCS1 (última vez consultada Febrero 2016): https://tools.ietf.org/html/rfc3447
https://en.wikipedia.org/wiki/PKCS_1

[12] Advanced Encryption Standard (AES) (última vez consultada Febrero 2016): http://iamvijayakumar.blogspot.com.es/2013/10/android-example-for-encrypt-and-decrypt.html

[13] Secure Hash Algorithm 1 (SHA-1) (última vez consultada Febrero 2016): http://es.wikipedia.org/wiki/Secure_Hash_Algorithm

[14] SQLite (última vez consultada Febrero 2016): http://sg.com.mx/revista/17/sqlite-la-base-datos-embebida#.VtCFTpPhDq5

[15] Twilio (última vez consultada Febrero 2016): https://www.twilio.com/docs/java/install

[16] AsyncTask Android (última vez consultada Febrero 2016): http://developer.android.com/reference/android/os/AsyncTask.html

[17] Crear un servicio en Android (última vez consultada Febrero 2016): http://www.hermosaprogramacion.com/2015/07/tutorial-para-crear-un-servicio-en-android/

[18] Material Design (última vez consultada Enero 2016): https://www.google.com/design/spec/material-design/introduction.html

[19] Bootstrap (última vez consultada Marzo 2016):

http://getbootstrap.com/

[20] MaterialTextField de florent37 (última vez consultada Enero 2016): https://github.com/florent37/MaterialTextField

[21] NineOldAndroids de de Jake Wharton (última vez consultada Enero 2016): https://github.com/JakeWharton/NineOldAndroids

[22] AndroidProcessButton de dmytrondanylyk (última vez consultada Enero 2016): https://github.com/dmytrodanylyk/android-process-button

[23] MaterialDialog de afollestad (última vez consultada Enero 2016): https://github.com/afollestad/material-dialogs

[24] Picasso de square (última vez consultada Enero 2016): https://github.com/square/picasso

[25] AndroidCrop de soundcloud (última vez consultada Enero 2016): https://github.com/jdamcd/android-crop

[26] AndroidUploadService de alexbbb (última vez consultada Enero 2016): https://github.com/alexbbb/android-upload-service

[27] CircularReveal de ozodrukh (última vez consultada Enero 2016): https://github.com/ozodrukh/CircularReveal

[28] PullRefreshLayout de baoyongzhang (última vez consultada Enero 2016):

https://github.com/baoyongzhang/android-PullRefreshLayout

[29] FilePicker de nononsenseapps (última vez consultada Enero 2016): https://github.com/spacecowboy/NoNonsense-FilePicker/blob/master/LICENSE

[30] Cuota mercado Android segundo trimestre 2015 (consultado en Enero 2016): http://hipertextual.com/2015/07/cuota-de-mercado-de-android-europa

1.13 Definiciones y abreviaturas

Tabla 1.5 - Tabla de definiciones y abreviaturas

AES	Advanced Encryption Standard (Estándar avanzado de encriptación)
API	Application Programming Interface (Interfaz de programación de aplicaciones)
GAE	Google App Engine
GCD	Google Cloud Datastore
GCE	Google Cloud Endpoint
GCM	Google Cloud Messaging (Sistema de mensajería en la nube de Google)
GCP	Google Cloud Platform
HTML	HyperText Markup Language (Lenguaje de marcado de hipertexto)
HTTP	HyperText Transfer Protocolo (Protocolo de transferencia de hipertexto)
Kernel	Software que constituye una parte fundamental del sistema operativo.
Linux	Combinación de kernel libre similar a Unix con el sistema GNU.
OHA	Open Handset Alliance (Alianza comercial formada por Google entre otros)
PKCS	Public Key Criptography Standard (Estándar criptográfico de clave pública)
RSA	Rivest Shamir Adleman (Iniciales de los apellidos de los creadores de la empresa y del algoritmo de encriptación RSA)
SDK	Software Development Kit (Kit de desarrollo software)
SHA	Secure Hash Algorithm (Algoritmo de *Hash* Seguro)
SSL	Secure Socket Layer
TCP	Transmission Control Protocol (Protocolo de control de transmisión)

1.14 Licencias

En los siguientes puntos de este apartado se incluyen todas las licencias de las bibliotecas adicionales de terceros que han sido utilizadas para el desarrollo de la aplicación para Android.

1.14.1 Spongy Castle

Copyright (c) 2000 - 2015 The Legion of the Bouncy Castle Inc. (http://www.bouncycastle.org)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.14.2 MaterialTextField

Copyright 2015 florent37, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.14.3 NineOldAndroids

Copyright 2012 Jake Wharton

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.14.4 AndroidProcessButton

Copyright (c) 2014 Danylyk Dmytro

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.14.5 MaterialDialog

The MIT License (MIT)

Copyright (c) 2015 Aidan Michael Follestad

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to dealing the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.14.6 Picasso

Copyright 2013 Square, Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.14.7 AndroidCrop

Copyright 2015 SoundCloud

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS

OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.14.8 AndroidUploadService

Copyright (C) 2013-2015 Aleksandar Gotev

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1.14.9 CircularReveal

The MIT License (MIT)

Copyright (c) 2014 Abdullaev Ozodrukh

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.14.10 PullRefreshLayout

The MIT License (MIT)

Copyright (c) 2014 baoyongzhang

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THESOFTWARE.

1.14.11 FilePicker

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this program, see COPYING.LESSER and COPYING. If not, see <http://www.gnu.org/licenses/>.

2 ANEXOS

2.1 Anexo I: Manual de administración del servidor

Suponiendo un escenario inicial sin servidor alguno, los pasos a seguir para conseguir un servidor totalmente operativo son:

- Ordenador conectado a Internet (con requisitos hardware necesarios para ejecutar Google Chrome y Android Studio).
- Máquina virtual Java instalada y configurada en el ordenador.

Una vez cumplidos los anteriores requisitos, ya se puede importar el paquete con el proyecto del servidor en Android Studio. Para ello en la pantalla de bienvenida se debe pulsar sobre la opción marcada en la imagen inferior *Import project (Eclipse ADT, Gradle, etc.).* En la ventana que se abrirá a continuación, se elige la ruta donde está el archivo *build.gradle* ó *settings.gradle* del proyecto.

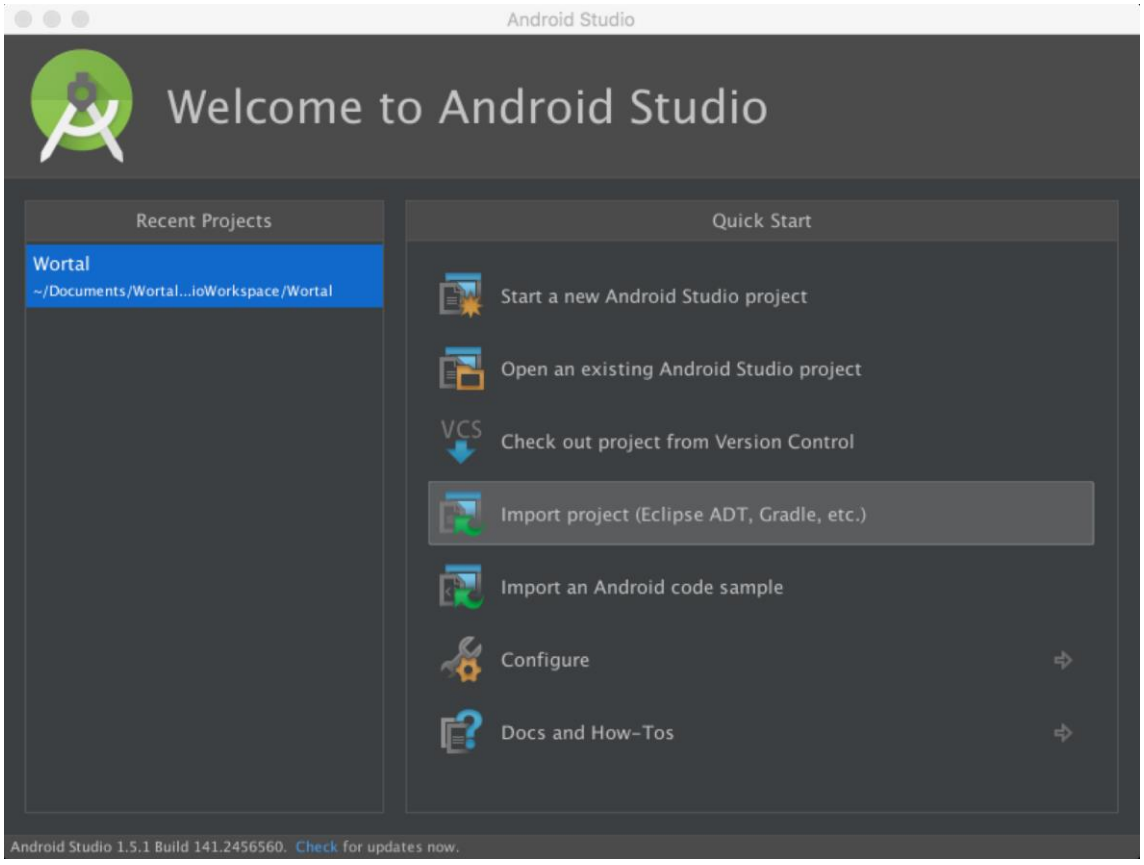


Figura 2.1 - Pantalla bienvenida Android Studio

Hecho esto, los módulos con la aplicación cliente (módulo app) y con la aplicación servidora (módulo backend) estarán correctamente importados en el entorno de desarrollo.

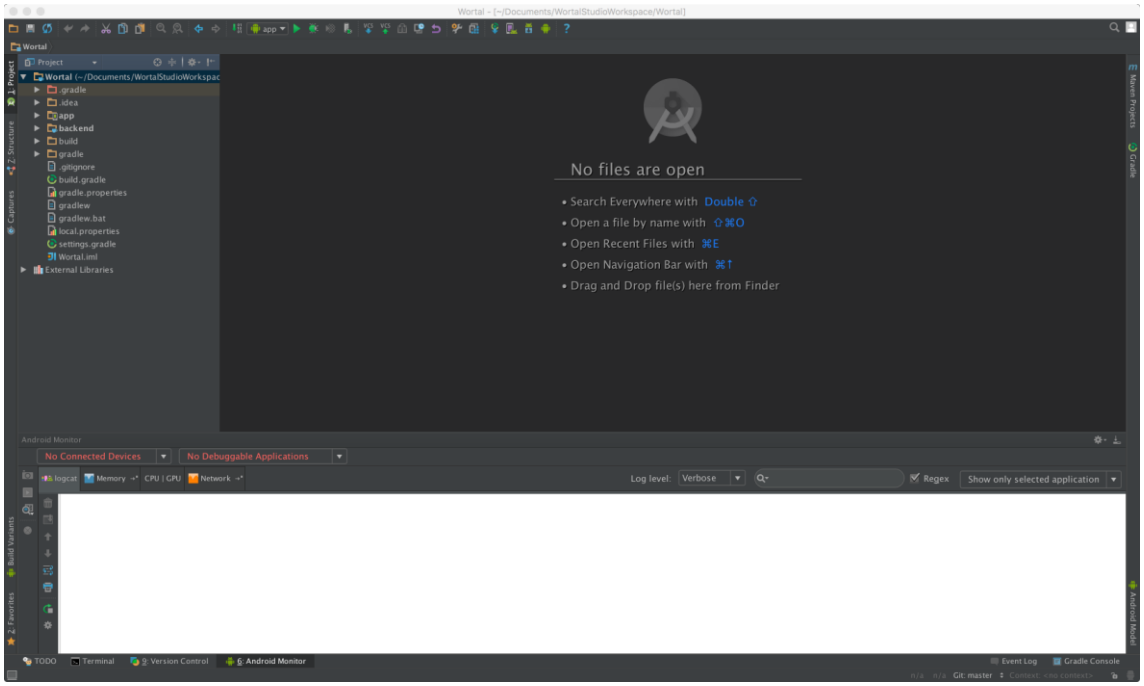


Figura 2.2 - Interfaz Android Studio

Para enviar el código del servidor a GAE, será necesario buscar la opción *Deploy module to App Engine* (previamente hay que haber iniciado sesión en Android Studio con el correo electrónico utilizado en la creación del servidor).

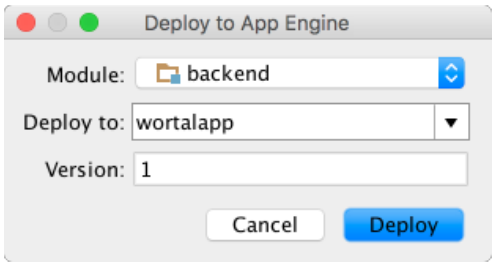


Figura 2.3 - Deploy to App Engine

Al pulsar el botón 'Deploy', en pocos segundos, el proyecto se subirá a GAE y se ejecutará tal y como esté programado.

Una vez que el servidor se está ejecutando en GAE, no se puede hacer ningún cambio en caliente, si es necesario realizar algún cambio en el código, será necesario volver a realizar los pasos anteriores.

2.2 Anexo II: Manual de usuario de la aplicación

Una vez que el usuario ha instalado la aplicación en su terminal, al ejecutarla por primera vez se mostrarán las pantallas de presentación de la aplicación. El usuario podrá navegar entre ellas deslizando a derecha y/o a izquierda.

Figura 2.4 - Pantalla bienvenida 1

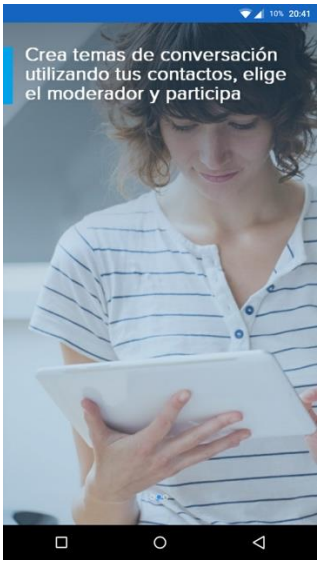


Figura 2.5 - Pantalla bienvenida 2

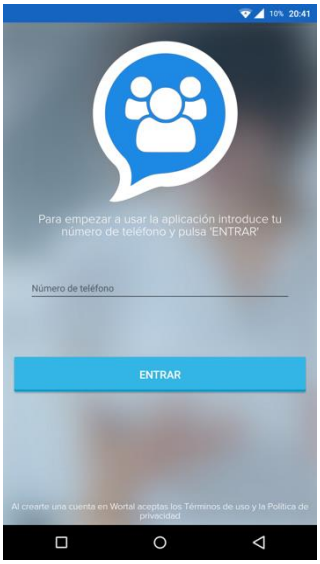


Figura 2.6 - Pantalla bienvenida 3

En la tercera pantalla se muestra un cuadro de texto donde el usuario debe introducir su número de teléfono. Una vez introducido, la pulsar en el botón 'ENTRAR', la aplicación se comunicará con el servidor para iniciar el proceso de registro. Si no hay ningún error, el servidor enviará un SMS de verificación.

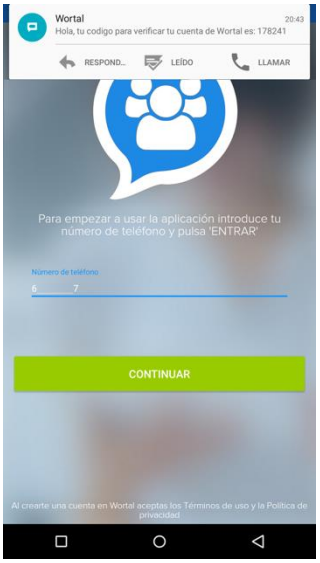


Figura 2.7 - SMS verificación

En este momento la aplicación obtendrá el código de verificación que aparece en el SMS y procederá a la comprobación con el servidor.

Si el resultado es satisfactorio el botón se pondrá de color verde y aparecerá el texto “CONTINUAR”. Al pulsar se abrirá la pantalla principal de la aplicación.

2.2.1 Pantalla principal y contactos

La primera vez que se ejecuta la aplicación, lógicamente, no hay ninguna conversación activa por lo que se le indicará al usuario que para iniciar una conversación pulse el botón para abrir la pantalla de ‘Contactos’. La primera vez la pantalla de contactos estará vacía por lo que para actualizar la lista deberá pulsar el primer botón superior o deslizar el dedo desde la parte superior de la pantalla hacia abajo. De las dos formas puede iniciarse el proceso de actualización de contactos.

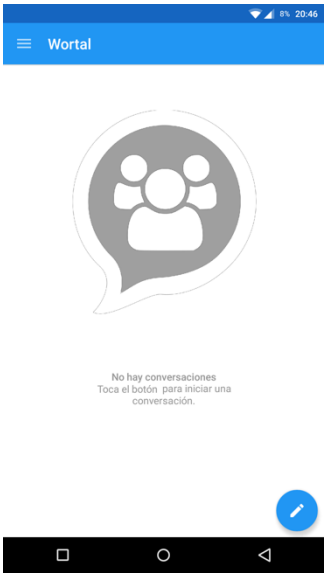


Figura 2.8 - Pantalla principal

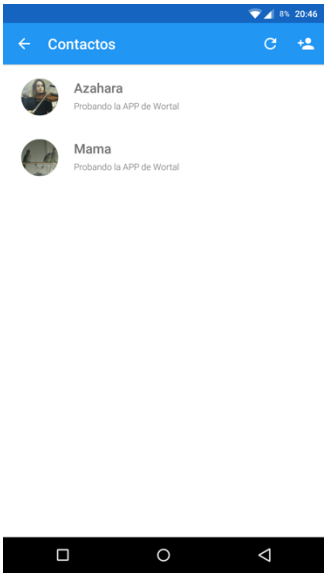


Figura 2.9 - Pantalla de contactos

Para empezar una conversación basta con seleccionar un contacto y se abrirá la pantalla donde se mostrarán todos los mensajes que se intercambien.

2.2.2 Menú lateral

Para acceder a otras opciones de la aplicación, se puede desplegar el menú lateral deslizando en la pantalla principal de izquierda a derecha o pulsando el icono de tres líneas horizontales.

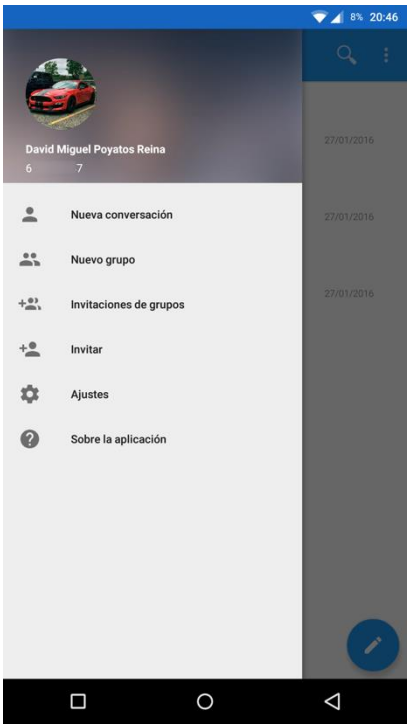


Figura 2.10 - Menú lateral

El menú consta de 5 opciones:

- Nueva conversación: se abre la pantalla de contactos y seleccionando uno se puede abrir una conversación nueva o ya existente.
- Nuevo grupo: se abre la pantalla que permite crear un nuevo grupo.
- Invitar: se abre la pantalla que permite seleccionar de entre los contactos del usuario a quiénes quiere enviar una invitación para descargar la aplicación.
- Ajustes: se abre la pantalla donde el usuario puede configurar algunos ajustes de la aplicación.
- Sobre la aplicación: se abre una pantalla donde se muestra información sobre la aplicación, sobre las licencias de uso y sobre el desarrollador.

Pulsando sobre la cabecera, donde aparece la foto de perfil, el nombre y el número de teléfono, se abrirá la pantalla de edición de perfil.

2.2.3 Pantalla editar perfil

En esta pantalla se muestra la imagen de perfil el nombre y el estado actuales. Pulsando sobre el icono de lápiz que aparece junto a cada elemento, se podrá proceder a editarlos.

Todos estos cambios se pueden realizar siempre y cuando el dispositivo tenga conexión a Internet, de lo contrario aparecerá un mensaje indicándole al usuario de la falta de conexión e invitándole a intentarlo más tarde.

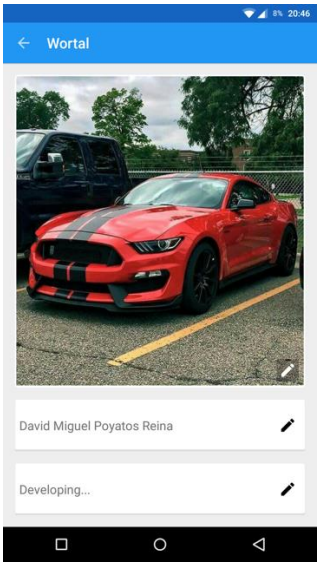


Figura 2.11 - Pantalla editar perfil

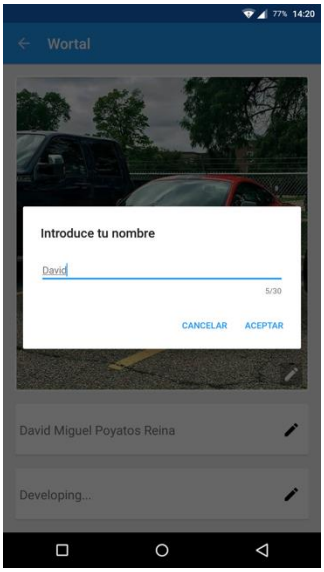


Figura 2.12 - Editar nombre

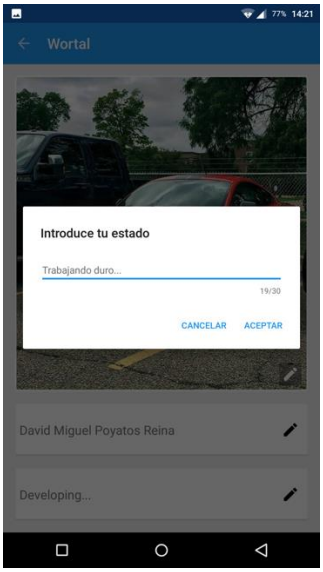


Figura 2.13 - Editar estado

2.2.4 Conversación privada

En las figuras siguientes se muestra la interfaz de conversación privada con algunos mensajes y archivos intercambiados.

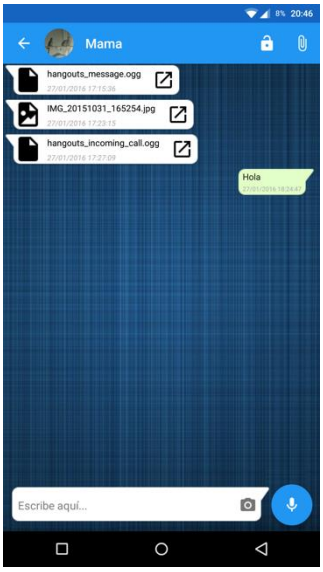


Figura 2.14 - Pantalla de conversación

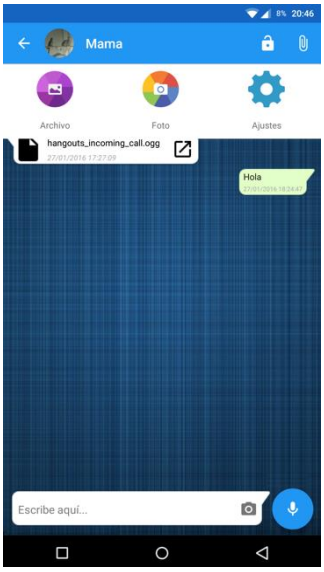


Figura 2.15 - Menú desplegable adjuntos

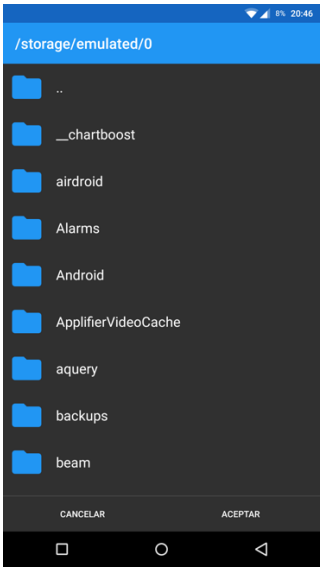


Figura 2.16 - Explorador de archivos

Para seleccionar la opción para enviar un archivo hay que pulsar el botón del 'clip' en la parte superior derecha. Pulsando sobre el primer de los elementos que aparece se abrirá el navegador de archivos que permitirá seleccionar el archivo a enviar.

Como indicador del tipo de conexión establecida (extremo a extremo o extremo-servidor) se muestra un candado en la parte superior de la pantalla de conversación.

Si el candado está abierto la conexión no está cifrada extremo a extremo. Pulsando sobre él se iniciará el intercambio de claves y cuando termine el proceso, el candado se cerrará y la conexión estará cifrada extremo a extremo. Si el usuario quiere terminar la conexión cifrada, pulsando sobre el candado, la conexión se liberará y se indicará al usuario mostrando el candado abierto.

Figura 2.17 - Candado abierto

Figura 2.18 - Candado cerrado

Manteniendo pulsado durante unos segundos un mensaje de la conversación se desplegará un menú flotante que permitirá al usuario copiar o borrar el mensaje.

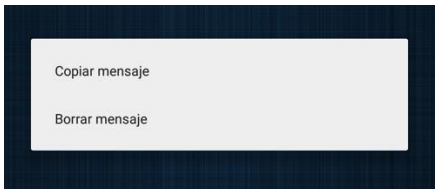


Figura 2.19 - Opciones mensaje

2.2.5 Envío de notas de audio

Una función añadida al servicio es la posibilidad de enviar notas de audio de forma rápida e intuitiva.

Figura 2.20 - Botón micrófono

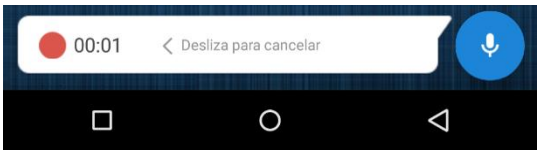


Figura 2.21 - Interfaz grabando audio

Para ello el usuario sólo tendrá que mantener pulsado el botón con el icono del micrófono y cuando termine de realizar la grabación deberá dejar de mantener pulsado el botón y el mensaje de audio se enviará a su destinatario. En caso de que el usuario desee cancelar la grabación, deberá deslizar el dedo hacia la parte izquierda de la pantalla.

2.2.6 Realización y envío de fotografías

Otra función añadida al servicio es la posibilidad de realizar rápidamente una fotografía y enviarla sin necesidad de abrir el navegador de archivos. Para ello puede seleccionarse la opción ‘Foto’ del menú desplegable de ‘Adjuntar’ (clip situado en la

parte superior derecha) o bien, pulsando sobre el icono de la cámara de fotos mostrado en la parte inferior de la pantalla, dentro del campo para introducir un mensaje.

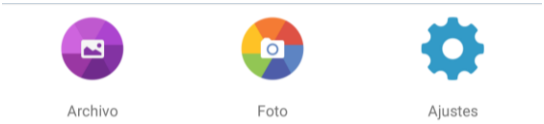


Figura 2.22 - Opción 'Foto' del menú de Adjuntar

Figura 2.23 - Botón cámara de fotos

2.2.7 Creación de grupos

Pulsando en la opción correspondiente del menú lateral, se da la opción al usuario de crear un grupo nuevo.

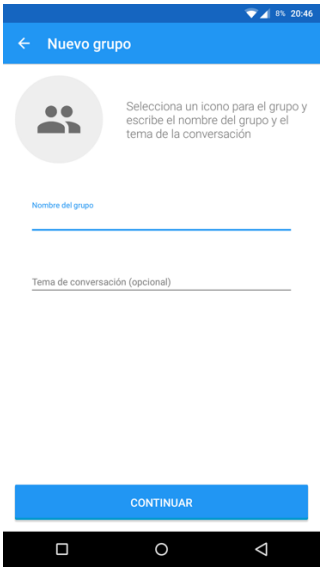


Figura 2.24 - Interfaz creación grupo

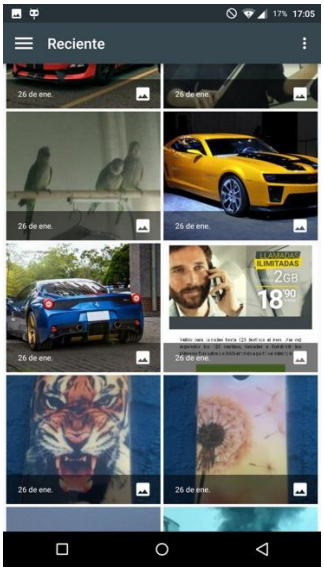


Figura 2.25 - Pantalla selección icono grupo

Figura 2.26 - Datos del grupo completados

En las figuras superiores se muestra, en primer lugar, la interfaz para introducir el nombre de grupo, la descripción sobre el tema de conversación (opcional) y pulsando sobre el icono se abre la galería de imágenes que permite la selección de una imagen para establecerla como icono del grupo.

Una vez completados los datos del grupo, el usuario debe pulsar el botón 'CONTINUAR', donde se mostrará a continuación la lista de contactos que participarán en el grupo.

Una vez seleccionados los participantes, se mostrará otra lista con los participantes seleccionados incluyendo al mismo usuario, el siguiente paso es la elección del moderador del grupo.

Una vez seleccionado, pulsando el botón 'CONTINUAR' la aplicación se comunicará con el servidor y se procederá al registro en la base de datos del grupo y al envío de las invitaciones de participación al grupo.

Si el resultado es satisfactorio, el botón cambiará a color verde y aparecerá el mensaje 'GRUPO CREADO CORRECTAMENTE'.

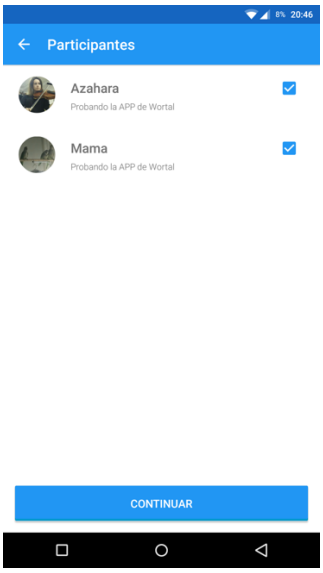


Figura 2.27 - Pantalla elección participantes

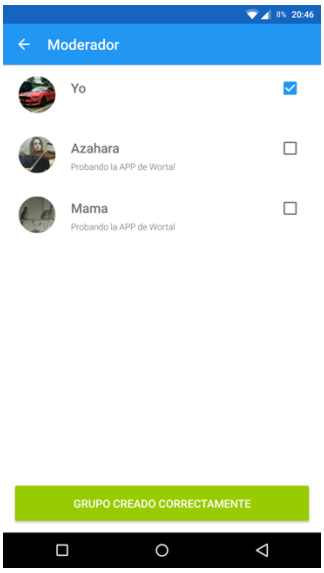


Figura 2.28 - Pantalla elección moderador

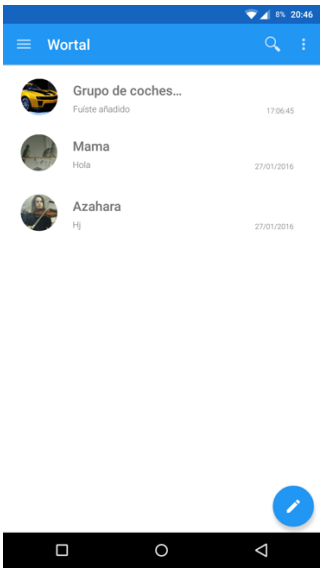


Figura 2.29 - Pantalla principal con el grupo creado

2.2.8 Conversación en grupo

La interfaz de conversación de grupo es similar a la conversación privada, con la diferencia que el candado para establecer la conexión cifrada extremo a extremo no aparece, ya que en conversaciones grupales no se da la opción de realizar el cifrado extremo a extremo.

Figura 2.30 - Pantalla conversación grupo, menú adjuntar

Figura 2.31 - Pantalla conversación con mensajes

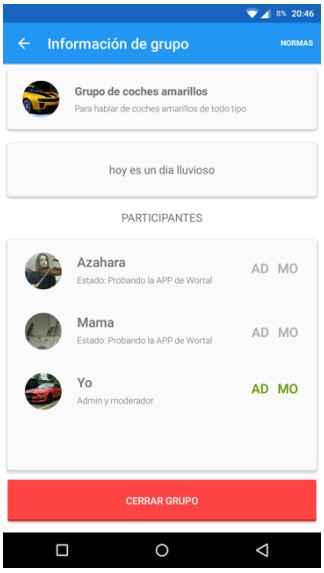


Figura 2.32 - Pantalla detalle grupo

En la figura 74, puede verse una conversación en la que en la parte superior de la lista de mensajes se muestra el mensaje del día establecido. Además puede verse que se ha enviado una Nota de audio, dos imágenes y en el momento de hacer la captura se estaba grabando una nota de audio (dejando pulsado el botón con el símbolo del micrófono).

En la pantalla de la conversación, realizando una pulsación larga sobre un mensaje se mostrará un menú flotante con 4 opciones:

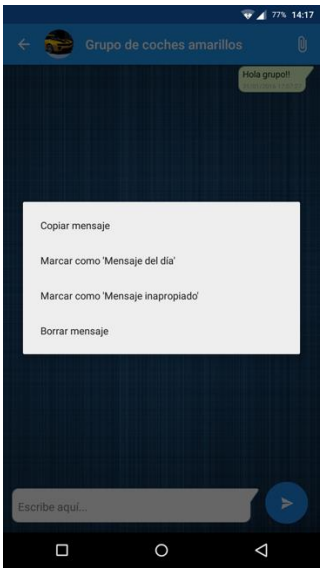


Figura 2.33 - Menú opciones mensaje grupo

Las opciones mostradas son las siguientes:

- Copiar mensaje: copia el contenido del mensaje en el portapapeles del dispositivo, pudiendo pegar el mensaje en cualquier otra aplicación.

- Marcar como 'Mensaje del día': establece el mensaje seleccionado como mensaje del día y se mostrará en el apartado de detalles de grupo.

- Marcar como 'Mensaje inapropiado': sólo el moderador puede realizar esta acción si el mensaje no cumple con las normas del grupo. El mensaje se ocultará y será el usuario el que decidirá su lectura o no.

- Borrar mensaje: borra el mensaje seleccionado de la base de datos del dispositivo.

Los mensajes marcados como inapropiados se mostrarán como aparece en la siguiente figura. En vez de aparecer el texto del mensaje, se mostrará un mensaje informativo y para ver el contenido del mensaje será necesario tocar sobre él y se mostrará en un mensaje flotante que desaparecerá a los pocos segundos.

Figura 2.34 - Mensaje inapropiado

Pulsando sobre el nombre del grupo, en la parte superior de la pantalla, se abrirá la pantalla de detalle de grupo, mostrada en la figura inferior. Desde ahí se podrán ver los detalles del grupo: imagen de grupo, nombre de grupo, descripción de grupo, normas, mensaje del día y participantes.

El administrador y el moderador podrán cambiar en cualquier momento la imagen de perfil, pulsando sobre la imagen se abrirá la galería de imágenes y podrá seleccionar la imagen deseada. Pulsando sobre el mensaje del día, se abrirá un cuadro de texto desde donde se podrá introducir el mensaje del día adecuado. Pulsando sobre el botón 'Normas', al igual que en el caso de modificar el mensaje del día, se desplegará un cuadro de texto desde donde se podrán modificar las normas. Adicionalmente, en la lista de participantes, se identifica al moderador y al administrador, marcando con color verde las letras correspondientes (AD para administrador y MO para moderador). El administrador y el moderador podrán cambiar su rol y otorgárselo a otro participante, para ello deberá pulsar en el símbolo correspondiente del participante seleccionado.

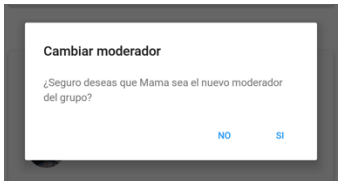


Figura 2.35 - Cambiar moderador del grupo

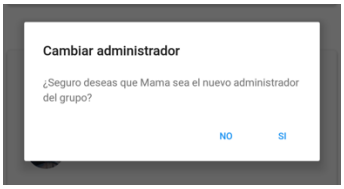


Figura 2.36 - Cambiar administrador del grupo

2.2.9 Invitar a contactos

Seleccionando esta opción en el menú lateral, se mostrará la lista de contactos que aún no son usuarios del servicio. Seleccionando a qué usuarios enviar la invitación y pulsando el botón 'INVITAR' el servidor enviará un SMS invitando a los usuarios a descargar y a usar la aplicación.

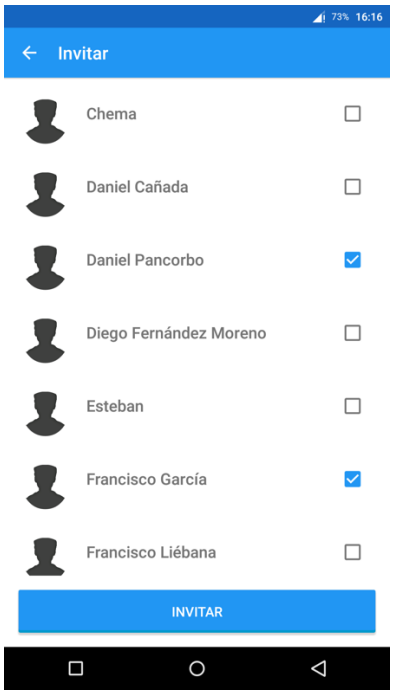


Figura 2.37 - Pantalla envío invitaciones

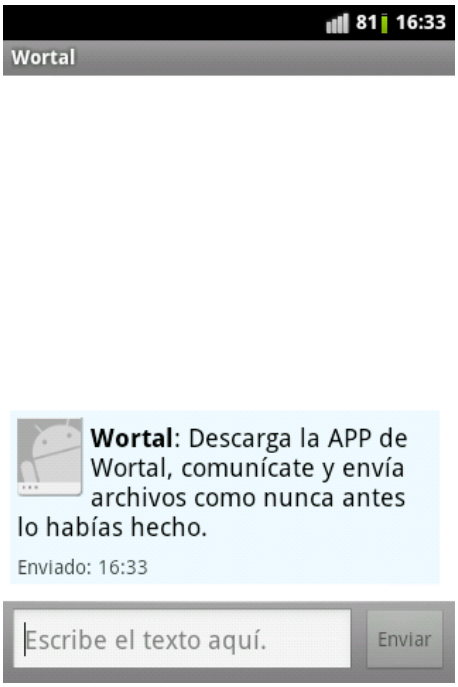


Figura 2.38 - Invitación recibida

2.2.10 Permisos

Para ejecutar correctamente la aplicación, el usuario debe aceptar los permisos necesarios al instalar la aplicación. En caso de que no esté de acuerdo, deberá eliminar la aplicación de su terminal.

Los permisos son los siguientes:

- ACCESS_NETWORK_STATE: necesario para acceder a la información sobre conectividad a red de datos.
- ACCESS_WIFI_STATE: necesario para acceder a la información sobre conectividad a red inalámbrica WiFi.

- INTERNET: necesario para poder establecer comunicación con Internet.
- READ_PHONE_STATE: necesario para poder obtener información sobre el estado del teléfono.
- WAKE_LOCK
- READ_CONTACTS: necesario para obtener los contactos de la agenda del usuario.
- RECEIVE_SMS: necesario para poder recibir SMS.
- READ_SMS: necesario para leer el SMS con el código de verificación en el proceso de registro.
- RECORD_AUDIO: necesario para poder enviar notas de audio.
- VIBRATE: necesario para que el terminal vibre cuando se muestre una notificación o cuando empieza la grabación de una nota de audio.
- CAMERA: necesario para poder realizar una fotografía.
- WRITE_EXTERNAL_STORAGE: necesario para poder escribir en el almacenamiento del terminal.
- READ_EXTERNAL_STORAGE: necesario para poder leer del almacenamiento del terminal.
- GET_ACCOUNTS: necesario para poder registrarse con los servidores de GCM.
- DOWNLOAD_WITHOUT_NOTIFICATION: necesario para poder descargar en segundo plano los archivos enviados en las conversaciones y descargar las imágenes de perfil.
- com.google.android.c2dm.permission.RECEIVE: necesario para poder recibir mensajes a través de GCM.
- com.davidmpr.tfm.permission.C2D_MESSAGE: necesario para procesar los mensajes recibidos a través de GCM.

2.3 Anexo III: Configuración Google Cloud Messaging

Google Cloud Messaging (GCM) para Android es un servicio gratuito que permite enviar información desde un servidor de aplicación a un dispositivo Android. El servicio GCM controla el almacenamiento en cola de los mensajes y la entrega a las aplicaciones.

Características básicas:

- Para que una aplicación reciba mensajes, no es necesario que ésta se encuentre ejecutándose, sino que el sistema la despertará cuando el mensaje llegue.
- GCM pasa la información recibida directamente a la aplicación, la cual tiene control total sobre ella.
- Requiere dispositivos con Android 2.2 o superior que tengan instalada la aplicación de Google Play Store.
- Hace uso de una conexión existente a los servicios de Google. Es necesario que los usuarios tengan configurada su cuenta de Google en el dispositivo.

Pasos para activar el servicio:

1. Crear un nuevo proyecto en la consola de Google APIs (https://code.google.com/apis/console) pulsando en 'Create Project'.

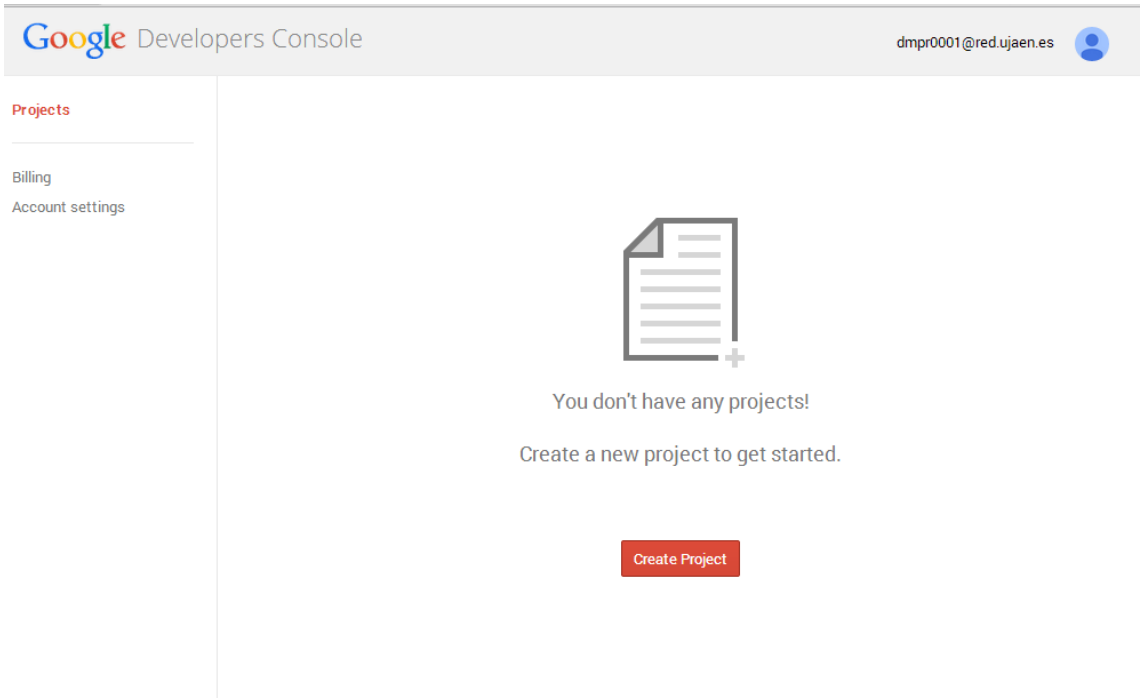


Figura 2.39 - Creación Proyecto en Google Developers Console

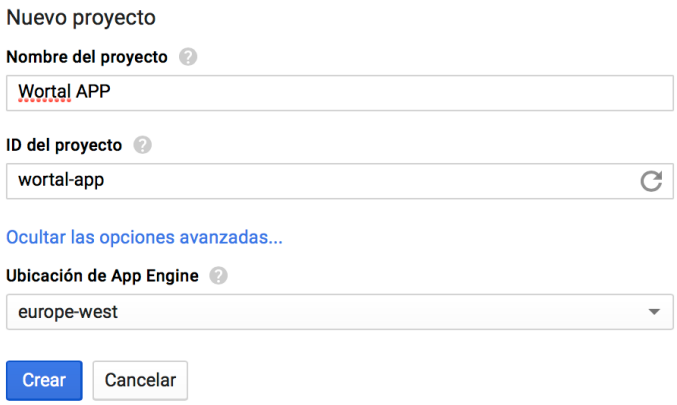


Figura 2.40 - Creación de proyecto Wortal

2. Una vez creado el proyecto se pincha en él, y en el menú 'APIS & AUTH' / 'Credentials', se obtiene el valor de 'SENDER ID' y se crea la Key ('Create new Key') para 'browser aplications' y se obtiene la 'API Key'.

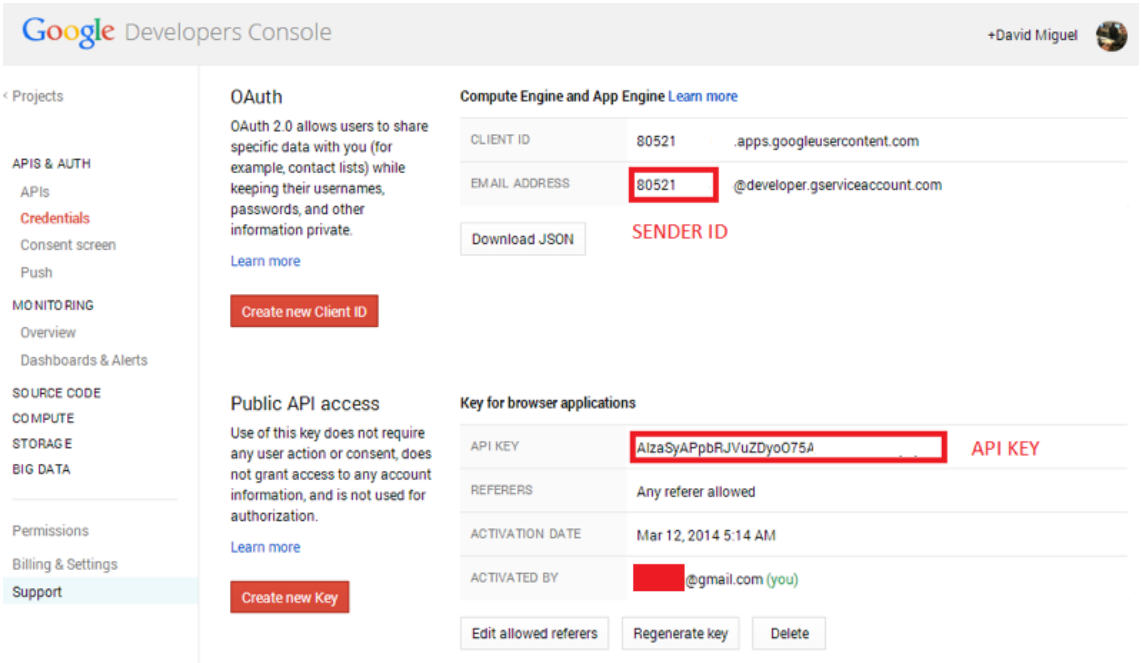


Figura 2.41 - Obtención SENDER ID y API Key

3. En el menú 'APIS & AUTH' / 'APIs' se debe activar la API para 'Google Cloud Messaging for Android'.

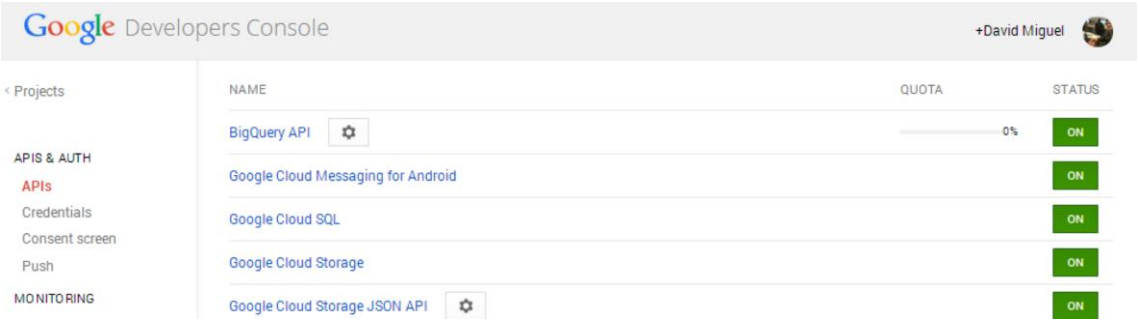


Figura 2.42 - Activación API GCM

Una vez terminado este proceso ya está activado el servicio, el siguiente paso será implementar las funciones necesarias en la aplicación cliente y en la aplicación servidora.

2.3.1 Configuración de la aplicación

Para configurar GCM en la aplicación, el primer paso es instalar la biblioteca de GCM, en concreto el paquete “Google Play services”.

1. Para instalar la biblioteca en el entorno de desarrollo Eclipse, en primer lugar hay que importar al espacio de trabajo dicha biblioteca, antes debe estar descargada mediante el SDK de Android. Para ello, abrir el programa ‘SDK Manager’ y en la sección ‘Extras’ instalar el paquete ‘Google Play services’.

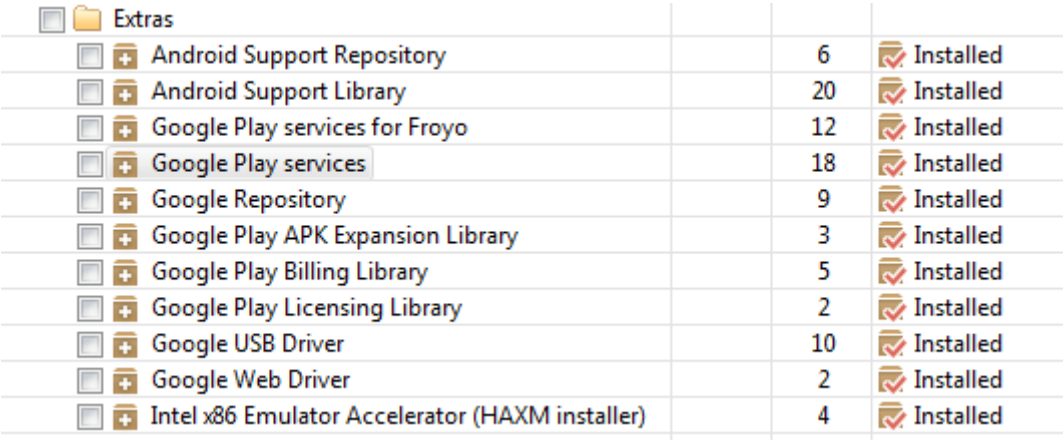


Figura 2.43 - Descarga biblioteca 'Google Play services'

Una vez que el paquete está instalado, se puede importar al espacio de trabajo del entorno de desarrollo.

File → Import → General → Existing Projects into Workspace → Next → Select root directory (Path SDK/extras/google/google_play_services) → Marcar ‘Copy projects into workspace’ → Finish

El siguiente paso es importar la biblioteca al proyecto. Para ello se pulsa con el botón derecho del ratón sobre la carpeta del proyecto → Properties (Alt + Enter).

Seleccionando el segundo elemento de la tabla de la izquierda 'Android', se añaden las bibliotecas de las que depende el proyecto.

En la parte inferior, en el apartado 'Library' aparecen las bibliotecas ya añadidas y pulsando el botón 'Add' se pueden seleccionar las bibliotecas disponibles para importar al proyecto.

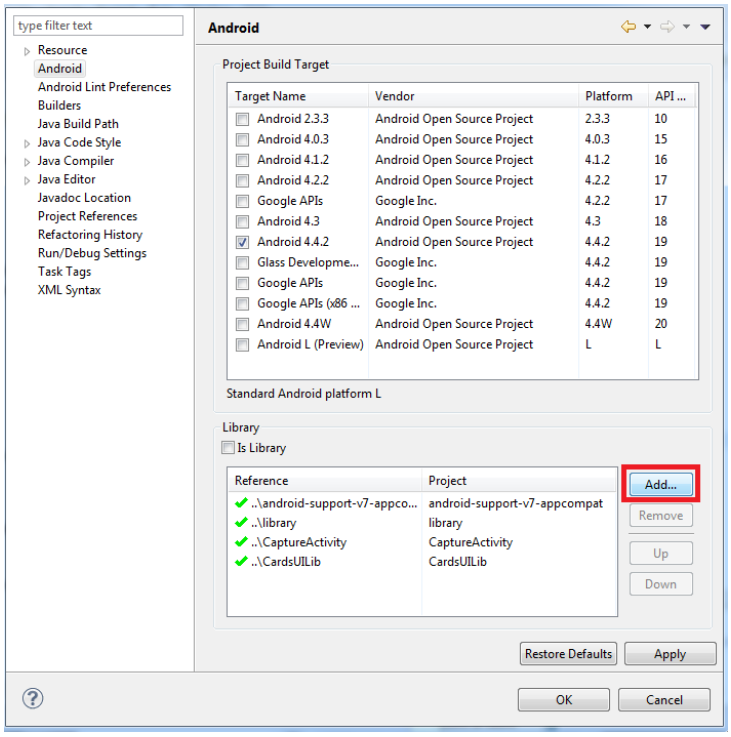


Figura 2.44 - Bibliotecas del proyecto

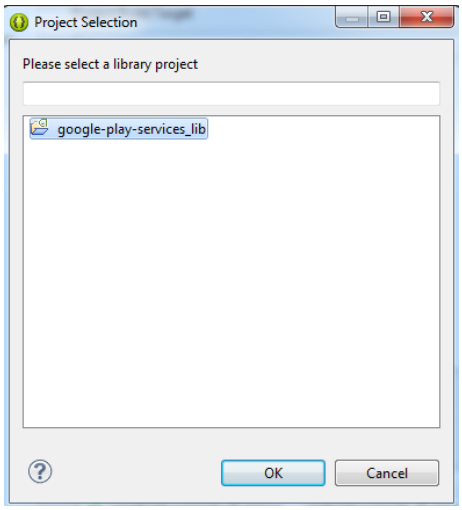


Figura 2.45 - Añadir biblioteca 'Google Play services'

En caso de utilizar Android Studio, lo único que habría que hacer es añadir al archivo *build.gradle* las siguientes líneas y automáticamente se descargará la biblioteca y se incorporará al proyecto:

```
dependencies{
   compile 'com.google.android.gms:play-services-gcm:7.5.0'
}
```

2. El segundo paso es añadir los permisos necesarios al manifiesto de la aplicación, archivo "AndroidManifest.xml".

Se añaden permisos para evitar que otras aplicaciones se registren y reciban los mensajes de nuestra aplicación. Se debe especificar el mismo nombre que el del paquete de la aplicación (que se encuentra también indicado en el propio *manifest*).

```
<permission android:name="com.tfg.tutoguia.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
<uses-permission android:name="com.tfg.tutoguia.permission.C2D_MESSAGE" />
```

Figura 2.46 - Permisos para usar GCM

Se añade el resto de permisos:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

Figura 2.47 - Resto de permisos para usar GCM

"GET_ACCOUNTS" permite acceder a la cuenta de Google del dispositivo.

Dentro de la etiqueta "application" se añade el receptor o cliente de los mensajes. Son 2 *intents* los que pueden recibirse del GCM: "REGISTRATION" y "RECEIVE". Gracias a que se está usando la biblioteca de GCM para Android, no hay que implementar ninguna clase propia cliente, sino que se usa la que la biblioteca proporciona *com.google.android.gcm.GCMBroadcastReceiver*.

```
<receiver
    android:name="com.google.android.gcm.GCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.tfg.tutoguia" />
    </intent-filter>
</receiver>
```

Figura 2.48 - Declaración receptor de mensajes GCM

Se debe declarar de esta forma, y no mediante programación, para que los mensajes sean recibidos incluso si la aplicación no se está ejecutando.

El permiso "SEND" se incluye para que tan sólo se reciban *intents* provenientes de los servidores GCM.

Por último, se añade un servicio cuyo nombre debe ser "*GCMIntentService*". Este nombre es al que se llama desde el receptor anterior. Este servicio sí hay que implementarlo con acciones propias.

```
<service android:name=".GCMIntentService" />
```

3. El tercer paso es implementar el código para registrarse al servicio: en la actividad de la aplicación, en el método '*onCreate*' hay que añadir el código para que la aplicación se registre en los servidores de GCM y así obtener el identificador necesario para el envío de notificaciones.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ctx=this;

    // Check device for Play Services APK.
    if(isOnline()==true){
        if (checkPlayServices()) {
            // If this check succeeds, proceed with normal processing.
            // Otherwise, prompt user to get valid Play Services APK.

        try{
           GCMRegistrar.checkDevice(this);
           GCMRegistrar.checkManifest(this);

           final String regId = GCMRegistrar.getRegistrationId(this);
           if (regId.equals("")) {
              GCMRegistrar.register(this, SENDER_ID);
           }
        } catch (UnsupportedOperationException e) {
           Toast.makeText(this, "El dispositivo no soporta GCM.", Toast.LENGTH_LONG).show();
        } catch (IllegalStateException e) {
            Toast.makeText(this, "El manifest no está bien configurado.", Toast.LENGTH_LONG).show();
        }
        }
    }
```

Figura 2.49 - Registro en servidor GCM

4. El cuarto paso es implementar el código del servicio que va a recibir los *intents.*

Cada vez que el dispositivo reciba un mensaje de los servidores de GCM, lo hará a través de un servicio que se ejecuta aunque la aplicación no lo esté. Este servicio consta de cuatro métodos que se encargan de procesar cuatro tipos de mensajes: error, mensaje recibido, registro correcto, baja correcta.

```
import com.google.android.gcm.GCMBaseIntentService;

public class GCMIntentService extends GCMBaseIntentService {

   @Override
   protected void onError(Context context, String errorId){
      // Error en el registro: tratamiento del error
   }

   @Override
   protected void onMessage(Context context, Intent intent){
      // Notificación recibida: informo al usuario u otra acción
   }

   @Override
   protected void onRegistered(Context context, String regId){
      // Registro correcto: envío el regId a mi servidor
   }

   @Override
   protected void onUnregistered(Context context, String regId){
      // Borrado correcto: informo a mi servidor
   }
}
```

Figura 2.50 - Clase receptora de mensajes GCM

2.3.2 Configuración del servidor

En este caso la configuración es más simple, ya que no hay que hacer uso de bibliotecas, ni crear servicios. Tan solo hay que crear una clase con los métodos adecuados para enviar al servidor de GCM los mensajes que se reenviarán a los teléfonos.

Para más detalle de esta clase (WortalMessaging.java) véase la documentación *javadoc*: DOCUMENTACIÓN/Javadoc_Wortal_backend/index.html

2.4 Anexo IV: Configuración Twilio SMS, servidor y aplicación

El primer paso para poder usar el servicio de envío de SMS de Twilio es registrarse en la web http://www.twilio.com/ rellenando un formulario como el de la imagen inferior, indicando que estamos construyendo un sistema de envío de alertas SMS y el lenguaje es Java.

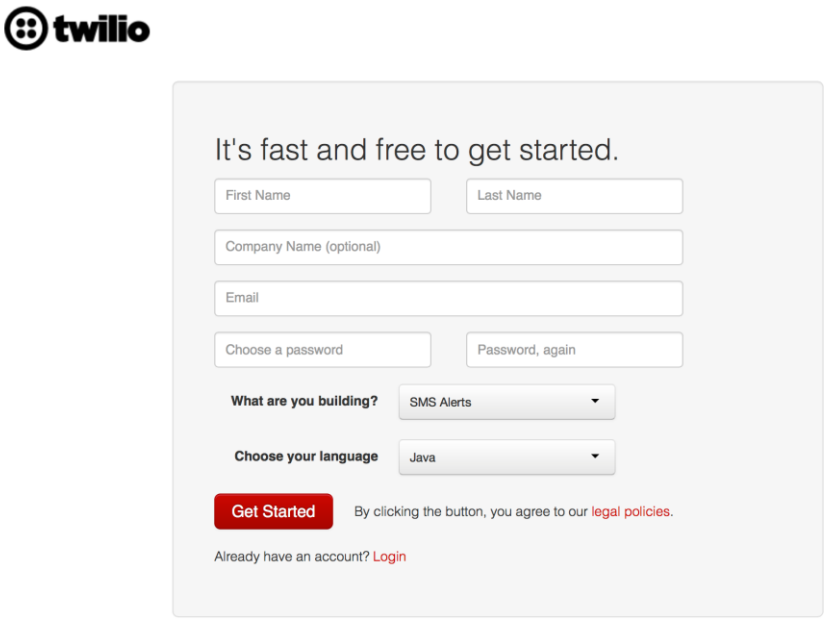


Figura 2.51 - Formulario registro Twilio

El primer paso para poder empezar a hacer uso del servicio es asociar a la cuenta un número de teléfono proporcionado por Twilio que permita el envío de SMS (con Twilio también pueden realizarse llamadas y en función del número elegido se podrá utilizar un servicio u otro o ambos). Este número de teléfono es el que, a priori, aparecerá como remitente en los terminales que reciban los SMS.

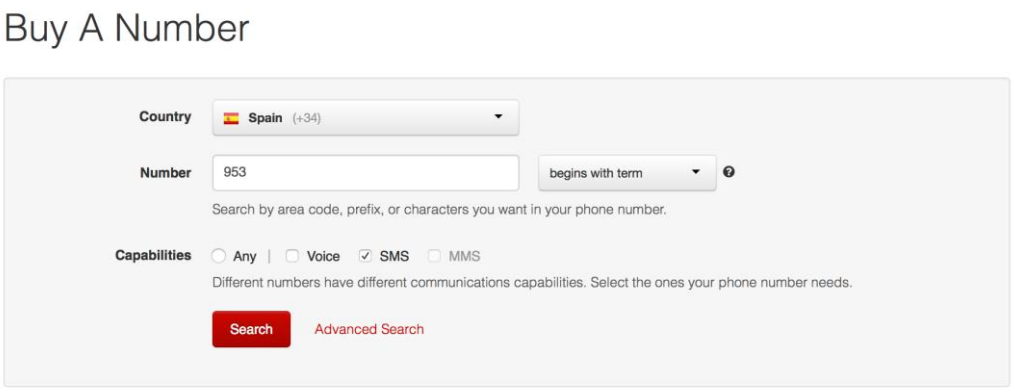


Figura 2.52 - Búsqueda de números

En la imagen inferior se muestra el número de teléfono asociado a la cuenta (pueden añadirse más pulsando en el botón *Buy a number* mostrado en la figura anterior). Como puede verse en la parte de *Capabilities*, sólo está activada la casilla SMS, ya que es el único servicio indispensable a utilizar. El número de teléfono es de la provincia de Jaén (utilizando el buscador de números se ha escogido uno de la provincia que permita SMS), y el nombre asignado es el del servicio desarrollado: *Wortal.*

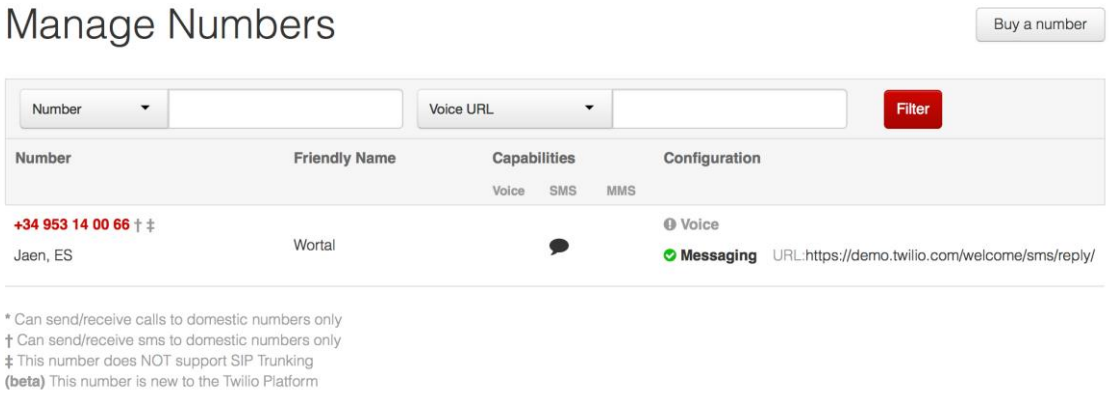


Figura 2.53 - Número de teléfono de Twilio

Al utilizar un *Friendly Name*, cuando un teléfono reciba un SMS, ya no aparecerá el número de teléfono, sino que aparecerá el nombre asignado, en este caso Wortal.

El coste por cada número de teléfono asociado es de 1$ mensual.

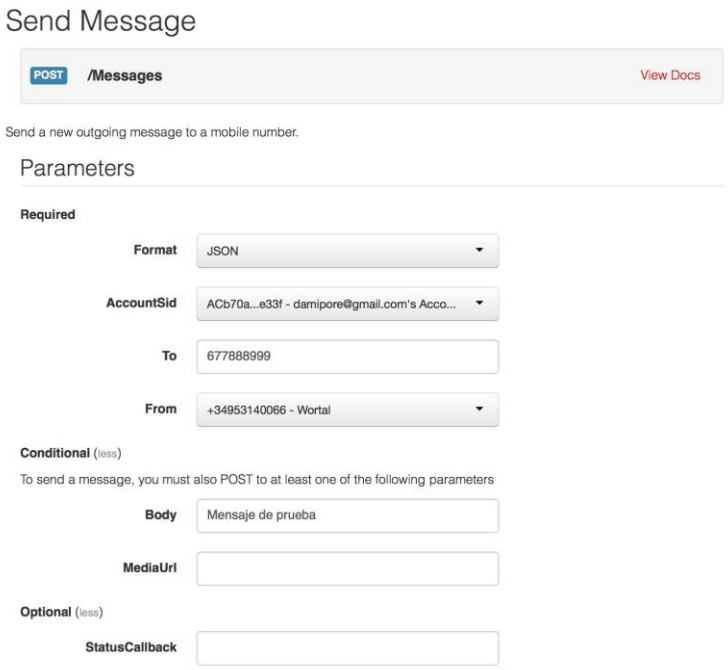
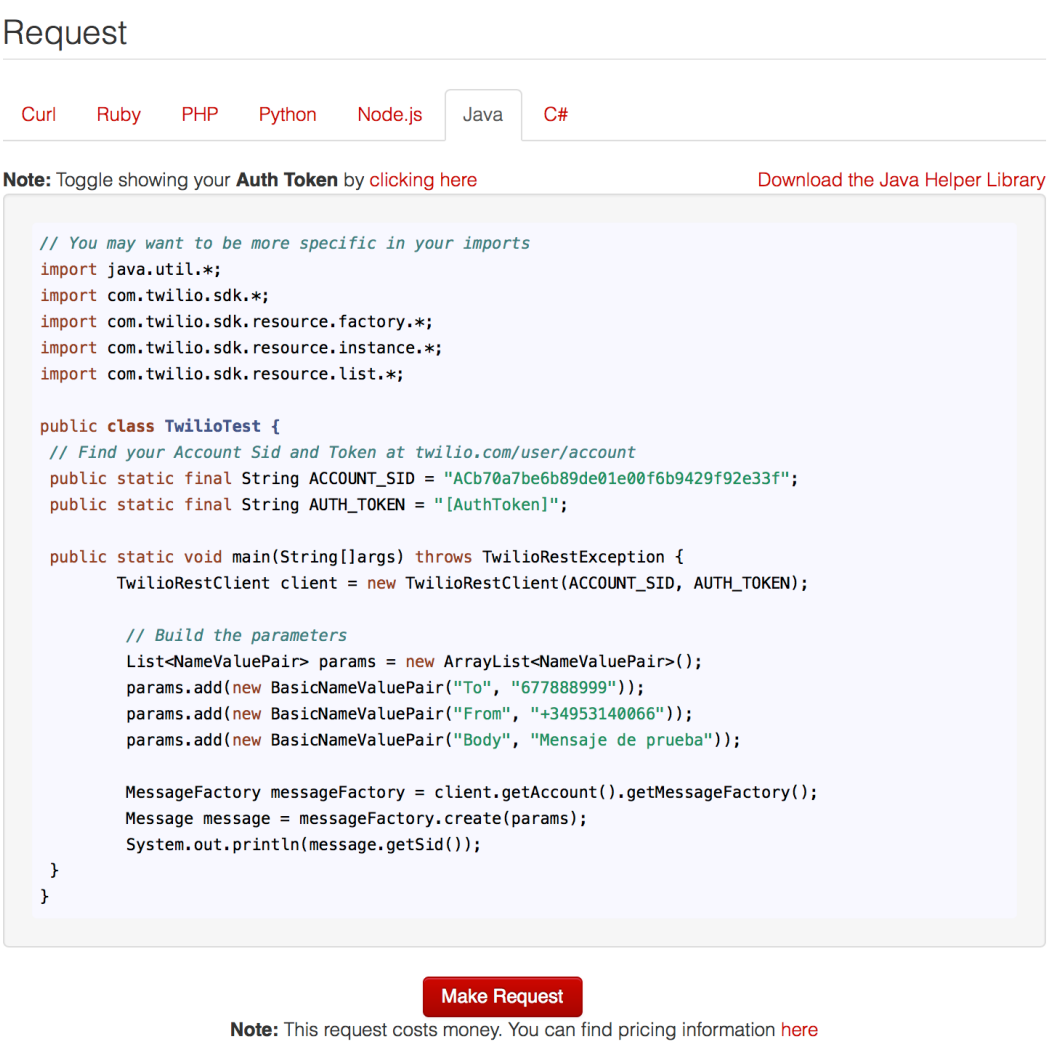


Figura 2.54 - Enviar SMS desde la web

En este punto, ya es posible empezar a enviar SMS con la herramienta para desarrolladores que ofrece Twilio, mostrada en la figura anterior. Los únicos campos que hay que rellenar es el número de teléfono del destinatario y el cuerpo del mensaje, el texto del SMS.

Automáticamente, la herramienta mostrará el código, en varios lenguajes de programación, necesario para enviar un SMS desde la aplicación en la que trabaje el desarrollador.



2.4.1 Clase para el envío de SMS desde el servidor

Para hacer uso del servicio en la aplicación servidora, se ha creado una clase llamada *SMSHelper* que contiene la función necesaria para enviar un SMS utilizando el código de ejemplo mostrado por la herramienta.

Los *ACCOUNT_SID*, y *AUTH_TOKEN* son identificadores de cada una de las cuentas registradas en el servicio. Si estos identificadores no son correctos, el SMS no se enviará.

```
public class SMSHelper {
    public static final String ACCOUNT_SID = "ACb70a7be6b89de01e00f6b9429f92e3__";
    public static final String AUTH_TOKEN = "addecd0a5723942d6026f36bac6065__";

    public static void sendSMS(String to, String body) throws TwilioRestException{
        TwilioRestClient client = new TwilioRestClient(ACCOUNT_SID, AUTH_TOKEN);

        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("Body", body));
        params.add(new BasicNameValuePair("To", "+34"+to));
        params.add(new BasicNameValuePair("From", "Wortal"));

        MessageFactory messageFactory = client.getAccount().getMessageFactory();
        Message message = messageFactory.create(params);
    }
}
```

Figura 2.55 - Clase servidor envío SMS

2.4.2 BroadcastReceiver para la recepción de SMS en la aplicación

Para que la aplicación cliente pueda leer el contenido del SMS recibido, es necesario crear un *BroadcastReceiver* que será notificado siempre que produzca un evento en el terminal, en este caso, la recepción de un SMS. Una vez notificado, se analizará el SMS y en caso de que el emisor se corresponda con *Wortal* se seguirá con el proceso.

En la imagen siguiente se muestra el código utilizado para la lectura de los SMS entrantes.

```
public class SMSReceiver extends BroadcastReceiver {


    static final String ACTION = "android.provider.Telephony.SMS_RECEIVED";

    @Override
    public void onReceive(Context context, Intent intent) {

        if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")){
            Bundle bundle = intent.getExtras();
            SmsMessage[] msgs = null;
            if (bundle != null){
                try{
                    Object[] pdus = (Object[]) bundle.get("pdus");
                    msgs = new SmsMessage[pdus.length];
                    for(int i=0; i<msgs.length; i++){
                        msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
                        String msg_from = msgs[i].getOriginatingAddress();
                        String msg_body = msgs[i].getMessageBody();

                        if (msg_from.equals("Wortal")==true){
                            WelcomeActivity.verifyPhone(msg_body, context);
                        }
                    }
                }catch(Exception e){
                        Log.d("Exception caught",e.getMessage());
                }
            }
        }
    }
}
```

Figura 2.56 - Servicio recepción SMS aplicación cliente

2.5 Anexo V: Objectify API para Java

Objectify es una API de acceso a datos desarrollada en lenguaje JAVA diseñada específicamente para funcionar en Cloud Datastore con App Engine. Algunos de los objetivos con los que se diseñó son:

- Fácil de entender y aprender.
- Soportar todas las capacidades del almacén de datos de forma intuitiva.
- Aumentar el rendimiento y reducir el coste a través del almacenamiento en caché de transacciones.

2.5.1 Definición de una entidad

La forma de definir una entidad utilizando Objectify es prácticamente igual que la creación de una clase en Java, con la diferencia de que hay que añadir unas *anotaciones* para que el sistema sepa que estamos definiendo una entidad (@Entity) y el tipo de cada uno de los parámetros (@Id, @Index).

En la imagen inferior puede verse un ejemplo para definir una entidad:

```
import com.googlecode.objectify.annotation.Entity;
import com.googlecode.objectify.annotation.Id;
import com.googlecode.objectify.annotation.Index;

@Entity
public class Car {
    @Id Long id;
    @Index String license;
    int color;
}
```

Figura 2.57 - Definición de una entidad con Objectify

Antes de crear la clase, se escribe la anotación para indicar que la clase que se va a definir es una entidad de la base de datos.

Posteriormente a cada uno de los parámetros de la clase se escribe la anotación que define la “clave primaria” con @Id, y @Index para indicar al sistema de forma explícita que indexe el parámetro que se le indique.

2.5.2 Operaciones básicas

Con Objectify pueden hacerse operaciones de forma similar a como se haría en otros sistemas de base de datos.

En la imagen inferior pueden verse algunos ejemplos de operaciones básicas:

```
import static com.googlecode.objectify.ObjectifyService.ofy;

Car porsche = new Car("2FAST", RED);
ofy().save().entity(porsche).now();    // async without the now()

assert porsche.id != null;    // id was autogenerated

// Get it back
Result<Car> result = ofy().load().key(Key.create(Car.class, porsche.id));  // Result is
Car fetched1 = result.now();    // Materialize the async value

// More likely this is what you will type
Car fetched2 = ofy().load().type(Car.class).id(porsche.id).now();

// Or you can issue a query
Car fetched3 = ofy().load().type(Car.class).filter("license", "2FAST").first().now();

// Change some data and write it
porsche.color = BLUE;
ofy().save().entity(porsche).now();    // async without the now()

// Delete it
ofy().delete().entity(porsche).now();    // async without the now()
```

Figura 2.58 - Operaciones básicas con Objectify

Para más información sobre los métodos utilizados en la implementación del servicio Wortal, consultar la documentación avadoc de las clases WortalUser.java, WortalGroup.java y WortalDatastore.java.

2.6 Anexo VI: Subida de imágenes y archivos a Google App Engine

La forma utilizada para actualizar la imagen de perfil de usuario y grupo, para enviar imágenes y para enviar archivos y notas de audio dentro de una conversación es muy similar.

2.6.1 Imágenes de perfil

El primer paso una vez que se tiene la URI de la imagen a enviar es solicitar al servidor una URL de subida a través del servlet *imgupload*.

```
public class ImgUpload extends HttpServlet{

    BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();


    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

        //"imguploaded" is another servlet which will send UploadUrl and blobkey to android client
        String blobUploadUrl = blobstoreService.createUploadUrl("/imguploaded");

        resp.setStatus(HttpServletResponse.SC_OK);
        resp.setContentType("text/plain");

        PrintWriter out = resp.getWriter();
        out.print(blobUploadUrl);
    }

    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        doGet(req, resp);
    }


}
```

Figura 2.59 - Servlet ImgUpload

El servlet genera una URL de subida apuntando al servlet “imguploaded”, que será el encargado de procesar la imagen de perfil y realizar las acciones correspondientes en la base de datos.

En las imágenes siguientes se muestra el código utilizado para solicitar la URL y una vez obtenida la forma en que se obtiene la URI local de la imagen de perfil y la llamada al método *upload* de la biblioteca UploadService para subir un fichero utilizando los parámetros indicados.

```
static class getUploadURLAsyncTask extends AsyncTask<String, Void, String> {
    private Context context;


    public getUploadURLAsyncTask(Context context) { this.context = context; }

    @Override
    protected String doInBackground(String... arg0) {
        String msg = "";

        HttpClient httpClient = new DefaultHttpClient();
        //This will invoke "ImgUpload servlet
        HttpGet httpGet = new HttpGet("https://wortalapp.appspot.com/getbloburl");
        HttpResponse response;

        try {
            response = httpClient.execute(httpGet);
            HttpEntity urlEntity = response.getEntity();
            InputStream in = urlEntity.getContent();
            String str = "";
            StringWriter writer = new StringWriter();
            String encoding = "UTF-8";
            IOUtils.copy(in, writer, encoding);
            msg = writer.toString();

        } catch (IOException ex) {
            ex.printStackTrace();
            msg = "Error: " + ex.getMessage();
        }
        return msg;
    }

    @Override
    protected void onPostExecute(String msg) {
        String url = msg;
        Random randomGenerator = new Random();
        int randomInt = randomGenerator.nextInt(100);

        File filesDir = new File(Environment.getExternalStorageDirectory().toString() + "/Wortal
        File imgPath = new File(filesDir, imgname + ".jpg");

        String uri = imgPath.toString();

        upload(context, randomInt+"", url, uri, imgname + ".jpg", "image/jpeg", imgname);
    }
}
```

Figura 2.60 - Asynctask subida imagen perfil

```
public static void upload(final Context context, String upload_id, String URL, String pathFile,
                          String nameFile, String type, String ID) {
    final UploadRequest request = new UploadRequest(context, upload_id, URL);

    request.addFileToUpload(pathFile, "file", nameFile, type);
    request.addParameter("ID", ID);
    request.addParameter("MODE","PROFILE");
    request.setNotificationConfig(android.R.drawable.ic_menu_upload,
            "Cambiando imagen de perfil de Wortal",
            "Subiendo imagen",
            "Imagen cambiada correctamente",
            "Error",
            false);

    request.setCustomUserAgent("UploadServiceDemo/1.0");
    request.setNotificationClickIntent(new Intent(context, ProfileActivity.class));
    request.setMaxRetries(2);

    try {
        //Start upload service and display the notification
        UploadService.startUpload(request);
    } catch (Exception exc) {
        Log.e("AndroidUploadService", exc.getLocalizedMessage(), exc);
    }
}
```

Figura 2.61 - Método subida imagen perfil

```
public class ImgUploaded extends HttpServlet{

    BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();

    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

        try {
            List<BlobKey> blobs = blobstoreService.getUploads(req).get("file");
            BlobKey blobKey = blobs.get(0);


            String id = req.getParameter("ID");
            String md = req.getParameter("MODE");

            //Store the image
            ImagesService imagesService = ImagesServiceFactory.getImagesService();
            ServingUrlOptions servingOptions = ServingUrlOptions.Builder.withBlobKey(blobKey);

            //Generate URL to get the image
            String servingUrl = imagesService.getServingUrl(servingOptions);

            if(md.equals("PROFILE")) {
                //Save the generated URL for the user
                List<WortalUser> records = ofy().load().type(WortalUser.class).list();
                for (WortalUser record : records) {
                    String idl = record.getPhonenumber();
                    if (idl.equals(id) == true) {
                        record.setIMG(servingUrl);
                        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
                        String fecha = sdf.format(new Date());
                        record.setMFecha(fecha);
                        record.setIMGKEY(blobKey.getKeyString());
                        ofy().save().entity(record).now();

                        //Send response to client with JSONObject
                        resp.setStatus(HttpServletResponse.SC_OK);
                        resp.setContentType("application/json");

                        JSONObject json = new JSONObject();
                        json.put("status", "OK");
                        json.put("servingUrl", servingUrl);
                        json.put("blobKey", blobKey.getKeyString());

                        PrintWriter out = resp.getWriter();
                        out.print(json.toString());
                        out.flush();
                        out.close();
                    }
                }
            }
```

Figura 2.62 - Servlet ImgUploaded

Como puede verse se generan dos parámetros:

- servingUrl: será la que se utiliza para acceder a la imagen.
- blobKey: será utilizada por el servidor para para eliminar la imagen en futuras actualizaciones.

2.6.2 Archivos

El primer paso una vez que se tiene la URI del archivo a enviar es solicitar al servidor una URL de subida a través del servlet correspondiente *fileupload*.

```
public class FileUpload extends HttpServlet{

    BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();


    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

        //"fileuploaded" is another servlet which will send UploadUrl and blobkey to android client
        String blobUploadUrl = blobstoreService.createUploadUrl("/fileuploaded");

        resp.setStatus(HttpServletResponse.SC_OK);
        resp.setContentType("text/plain");

        PrintWriter out = resp.getWriter();
        out.print(blobUploadUrl);
    }

    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        doGet(req, resp);
    }


}
```

Figura 2.63 - Servlet FileUpload

El proceso de subida es muy similar al seguido para la actualización de la imagen de perfil, la diferencia viene en el procesado que se realiza en el servlet *fileuploaded*, ya que en este caso no se utiliza el almacén de blobs de GAE, sino que se utiliza el llamado Google Cloud Storage, un espacio de almacenamiento similar a otros servicios en la nube.

Por todo lo demás el procesamiento es igual, ya que se generará una URL para poder acceder al archivo.

3 PLIEGO DE CONDICIONES

En este apartado se describen las condiciones técnicas para la correcta ejecución de la aplicación, incluyendo, tanto recursos hardware, como software, así como la normativa aplicable y el contrato de usuario.

3.1 Requisitos para ejecutar la aplicación cliente

Requisitos mínimos para ejecutar la aplicación cliente en terminal Android:

- Versión mínima: Android 2.3 (Gingerbread API 8)
- Memoria RAM: 512 Mb
- Procesador: Singlecore 1 GHz
- Almacenamiento disponible: 50 Mb

Requisitos recomendados para ejecutar la aplicación en terminal Android:

- Versión: Android 5.0 (Lollipop API 23)
- Memoria RAM: 2 Gb
- Procesador: Quadcore 2 GHz
- Almacenamiento disponible: 200 Mb

3.2 Acuerdo de usuario

Al utilizar el servicio Wortal, el usuario debe aceptar compartir el uso de los siguientes datos:

- Número de teléfono móvil.
- Acceso a Internet.
- Código GCM del dispositivo móvil.

En cumplimiento de la Ley Orgánica 15/1999, de 13 de Diciembre de Protección de Datos de Carácter Personal (LOPD), se informa de que los datos solicitados quedarán incorporados en una base de datos cuya finalidad es el correcto uso de la aplicación Wortal. Los campos anteriormente nombrados son de cumplimentación obligatoria, siendo imposible realizar la finalidad expresada si no se aportan estos datos. Queda igualmente informado de la posibilidad de ejercitar los derechos de acceso, rectificación, cancelación y oposición, de sus datos personales ante la entidad, organismo o empresa que pudiera ofrecer el servicio soportado por la aplicación Wortal en un futuro.

3.3 Normativa

Acuerdos o licencias aceptados para el desarrollo del trabajo.

• Google App Engine: Al utilizar la plataforma Google App Engine como desarrollador, se aceptan los términos de uso que aparecen en https://cloud.google.com/terms.

• Android: Al desarrollar una aplicación usando el sistema operativo de código abierto Android, se aceptan los términos de uso que aparecen en http://developer.android.com/legal.html.

4 ESTADO DE LAS MEDICIONES

En este apartado se definen y determinan las unidades de cada partida que configuran la totalidad del Trabajo Fin de Máster.

CAPÍTULO 1 – RECURSOS MATERIALES
Hardware y software utilizado.

Tabla 4.1 - Mediciones 'Recursos materiales'

CÓDIGO	RESUMEN	UNIDADES
1.01	Ordenador Portátil	1
1.02	Microsoft Windows 10	1
1.03	Microsoft Office 2016	1
1.04	Android Studio	1
1.05	Java (JRE + JDK)	1
1.08	Samsung Galaxy Ace 3	1
1.09	OnePlus One	1

CAPÍTULO 2 – RECURSOS HUMANOS
Horas empeladas para cada una de las fases de desarrollo.

Tabla 4.2 - Mediciones 'Recursos humanos'

CÓDIGO	RESUMEN	UNIDADES
2.01	Análisis del entorno	40
2.02	Análisis del sistema	40
2.03	Diseño del sistema	40
2.04	Implementación	500
2.05	Pruebas	50
2.06	Documentación	80

CAPÍTULO 3 – COSTES INDIRECTOS
Costes indirectos aplicados al trabajo.

Tabla 4.3 - Mediciones 'Costes indirectos'

CÓDIGO	RESUMEN
3.01	Conexión a Internet
3.02	Luz

5 PRESUPUESTO

En este documento se determinará el coste económico de cada capítulo y del Trabajo Fin de Máster en su totalidad.

5.1 Capítulo 1 - Recursos materiales

Tabla 5.1 - Presupuesto 'Recursos materiales'

Unidades	Concepto	Coste (€)	Tiempo de uso	Período de amortización	Subtotal (€)
1	Ordenador Portátil	1.649,00	4 meses	5 años	110,00
1	Microsoft Windows 10	0,00	4 meses		0,00
1	Microsoft Office 2016	0,00	4 meses		0,00
1	Android Studio	0,00	4 meses		0,00
1	Java (JRE + JDK)	0,00	4 meses		0,00
1	Samsung Galaxy Ace 3	200,00	4 meses	2 años	66,66
1	OnePlus One	350,00	4 meses	2 años	58,33
				Total	**234,99**

5.2 Capítulo 2 - Recursos humanos

Tabla 5.2 - Presupuesto 'Recursos humanos'

Unidades	Concepto	Precio/hora(€)	Subtotal (€)
40	Análisis del entorno	50,00	2.000,00
40	Análisis del sistema	50,00	2.000,00
40	Diseño del sistema	50,00	2.000,00
500	Implementación	50,00	25.000,00
50	Pruebas	50,00	2.500,00
80	Documentación	50,00	4.000,00
750		**Total**	**37.500,00**

5.3 Capítulo 3 - Costes indirectos

Tabla 5.3 - Presupuesto 'Costes indirectos'

Concepto	Precio (€)/mes	Tiempo de uso	Subtotal (€)
Conexión a Internet	50,00	4	200,00
Luz	60,00	4	480,00
		Total	**680,00**

SUBTOTAL = CAPÍTULO 1 + CAPÍTULO 2 + CAPÍTULO 3 **SUBTOTAL: 38.414,99 €**

IVA (21%): 8.067,15 €

TOTAL: 46.482,14 €

Asciende el presente Presupuesto de Trabajo Fin de Máster "*Servicio de mensajería instantánea con moderación y control de temática con cliente para terminales Android*" a la cantidad de cuarenta y seis mil cuatrocientos ochenta y dos euros con catorce céntimos.

En Linares, a 7 de marzo de 2016

NOMBRE POYATOS REINA DAVID MIGUEL - NIF 53598409Z

Firmado digitalmente por NOMBRE POYATOS REINA DAVID MIGUEL - NIF 53598409Z
Fecha: 2016.03.06 16:53:45 +01'00'

Fdo.: David Miguel Poyatos Reina