UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

SISTEMA DE RECOMENDACIÓN DE IMÁGENES INTEGRADO EN UNA APLICACIÓN DE MENSAJERÍA

Victor Gualdras de la Cruz

Septiembre, 2016

SISTEMA DE RECOMENDACIÓN DE IMÁGENES INTEGRADO EN UNA APLICACIÓN DE MENSAJERÍA

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE RECOMENDACIÓN DE IMÁGENES INTEGRADO EN UNA APLICACIÓN DE MENSAJERÍA

Autor: Victor Gualdras de la Cruz

Director: Dr. Jesús Serrano Guerrero

Septiembre, 2016

Victor Gualdras de la Cruz

Ciudad Real – Spain

E-mail: victor.gualdrasla@alu.uclm.es
Teléfono: 679 532 016

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

Uno de los objetivos de la informática es facilitar la realización de ciertas actividades y tareas a los usuarios, especialmente a aquellos que por una razón u otra tienen mayores dificultades a la hora de desempeñar alguna de estas labores. Hay muchas personas, que por razones como la edad, algún tipo de enfermedad, alguna minusvalía y/o discapacidad, tienen problemas a la hora de leer y escribir, especialmente utilizando los medios electrónicos actuales. El problema se acentúa especialmente en los dispositivos móviles smartphones, donde la pantalla es de un tamaño reducido, así como el teclado disponible para escribir. Este problema de lectura y escritura se traslada también al terreno de las telecomunicaciones. En un mundo en el que la mayor parte de la comunicación electrónica se realiza de manera escrita (obviando las llamadas), ya sea mediante las aplicaciones de mensajería que se utilizan en dispositivos móviles y ahora también en otros dispositivos, como las redes sociales, foros o comentarios, estas personas quedan aisladas y apartadas. Este Trabajo Fin de Grado pretende ayudar a estas personas en el ámbito de la comunicación a través de Internet.

Se propone con este fin el desarrollo de una aplicación de mensajería para la plataforma Android. Esta aplicación, a partir de una entrada que bien sea escrita, pero que en la mayoría de las ocasiones se realizará mediante reconocimiento de voz, propondrá al usuario una serie de imágenes. Se usarán tanto pictogramas creados especialmente para la comunicación de personas que tienen problemas en este área, como imágenes o fotografías de propósito y carácter general que se encuentren en una serie de sitios web que se seleccionarán para este propósito. Estas imágenes se le presentarán o propondrán al sujeto haciendo uso de un sistema de recomendación, el cual, a partir de la entrada proporcionada por el usuario que solicita o para el que se está haciendo la recomendación, las elecciones previas realizadas por este, así como por las elecciones de otros usuarios, recomendará una serie de imágenes.

Abstract

One of the main goals of computer science is to help people in the development of their activities and tasks. This applies particularly to those people that for whatever reason, have problems in the performance of those tasks. There are a lot of people that for reasons such as the age, some illnesses, or some handicap and/or disability, have problems with activities of reading and writing, especially when they have to use the electronic means. This problem is even bigger when using mobile devices such as smartphones, where there is a very small screen and keyboard. The problem of writing and reading is also present in the field of telecommunications. In a world where most of the on-line communication is performed written (except for the phone calls), either by messaging application, social networks, forum or comments in websites, this people are being left out or neglected. That is the purpose of this project, which aims to help this people in the communication throw the Internet.

It is proposed for this purpose the development of a messaging application for the Android platform. This application, starting with the input provided by the user that will be either written, or more probably by voice, will provided the user with a set of images. The images used will be both pictograms especially made for the communication of people with problems in this area, and images or photographs of general purpose located in some web sites selected for this reason. This images will be proposed to the user through a recommender system. This system will use the information available from previous choices of images by the user, and the choices of other users to provide a recommendation.

Agradecimientos

Este documento existe gracias a un gran número de personas que me han ayudado durante el largo camino. En primer lugar y especialmente dar las gracias a mis padres, cuyo apoyo moral y económico siempre ha estado cuando lo he necesitado. También al resto de mi familia, a la que sigue apoyándome y a la que por desgracia ya no puede.

A mis amigos del pueblo, sin los cuales probablemente no hubiese llegado a la universidad.

A los amigos que he hecho en Ciudad Real, tanto de la residencial El Doncel como a mis compañeros de clase y piso, por darme otro hogar al que volver.

A mi pareja Maite, por ser una persona en la que siempre pude apoyarme cuando lo necesité.

A mi director del TFG, por su aydua y por renunciar a parte de sus vacaciones por mi culpa.

A aquellos profesores que se han esforzado porque aprendiese lo necesario para seguir adelante.

Muchas gracias a todos.

Victor

A mi familia y amigos

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

HU	Historias de Usuario
TFG	Trabajo Fin de Grado
SO	Sistema Operativo
SAAC	Sistemas Aumentativos y Alternativos de Comunicación
IDE	Entorno de Desarrollo Integrado
ADT	Android Developments Tools
GCP	Google Cloud Platform
IAAS	Infrastructure as a Service
PAAS	Platform as a Service
SAAS	Software as a Service
ACID	Atomicity Consistency Isolation Durability
CRUD	Create Read Update Delete
RDBMS	Relational DataBase Management System
API	Application Programming Interface
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
ML	Machine Learning
REST	Representational State Transfer
GCM	Google Cloud Messaging
GSE	Google Search Engine
OO	Orientación a Objetos
JWI	MIT Java Wordnet Interface
MIT	Massachusetts Institute of Technology
TF-IDF	Term frequency–inverse document frequency
MAE	Mean Absolute Error
BBDD	Base de Datos

SDK	Software Development Kit
HU	Historia de Usuario

Capítulo 1

Introducción

Existe una gran cantidad de personas que tienen dificultades para comunicarse a través de los medios de comunicación escritos. Las populares aplicaciones de mensajería que se comercializan hoy día, utilizan mayoritariamente este tipo de comunicación, y aunque aportan la posibilidad de enviar otro tipo de archivos o información, no facilitan esta tarea. Se propone para ayudar a estas personas, el desarrollo de una aplicación de mensajería que, a diferencia de las ya actuales, permita a los usuarios utilizar imágenes como principal mecanismo de comunicación, haciendo uso para esto de un sistema de recomendación. Además, este sistema puede ser muy útil para todo tipo de personas, independientemente de si tienen problemas con la comunicación escrita o no, ya que aporta una funcionalidad que muchos usuarios encontrarán útil durante sus conversaciones.

Los sistemas de recomendación están cada vez más presentes en el mundo de la informática. Una gran cantidad de sitios web tienen una cantidad de artículos o elementos superior a la que los usuarios que acceden a sus servicios pueden consultar. Este tipo de sistemas se utilizan en sitios desde portales de venta online como amazon [1], en redes sociales para recomendar contactos como facebook [2] o en sitios de distribución de contenido multimedia como netflix [3].

En este caso, el contenido de la recomendación de este sistema serán imágenes. La razón para haber seleccionado un sistema de este tipo, y no haber recurrido simplemente a un sistema de filtrado o recuperación de información o simplemente un buscador, es debido a que además de presentar una ventaja notoria frente a estos sistemas una vez se tiene suficiente información sobre el usuario, son especialmente adecuados en aplicaciones en las que existen usuarios con perfiles y gustos comunes. Debido a que está orientado a personas que generalmente van a tener una serie de rasgos en común, como por ejemplo personas con síndrome de Down, será más fácil identificar a estas personas dentro de un grupo de usuarios concreto con mayor facilidad que si se tratase de otro tipo de usuarios. Así, si varios usuarios que comparten gustos son activos, y el sistema detecta patrones en común entre ellos, cuando entre un nuevo usuario al sistema, si este detecta que tiene relación en común con estos,

[1] www.amazon.es
[2] www.facebook.com
[3] https://www.netflix.com/

el sistema le recomendará imágenes que con mayor probabilidad cumplirán los requisitos o gustos del usuario en mayor grado que una imagen estándar. Además, debido a que se tratarán de grupos de usuarios muy concretos, este sistema es mucho más útil que un simple buscador de información, el cual devolvería el mismo tipo de resultado a cualquiera, aunque los gustos e intereses de ambos sean completamente distintos.

Sin embargo, este sistema no tiene porque limitarse a aquellos usuarios que presentan dificultades con la comunicación escrita. Puede ser un aliciente y un elemento diferenciador que podría incorporarse a cualquier aplicación de mensajería, aportando una gran experiencia de usuario. Cuánta gente, en un momento determinado durante una conversación, no ha pensado en una imagen que vendría perfecta para esa situación, y ha tenido que bien buscarla entre la gran cantidad de imágenes que tiene en su dispositivo, o recurrir a salir de la aplicación, abrir el navegador y tener que buscarla, descargarla y luego enviarla (quizás ya demasiado tarde), o simplemente ha renunciado a la idea por pura pereza. El sistema utilizado en esta aplicación podría trasladarse fácilmente a cualquier otra, e integrarse con su infraestructura.

También se ha optado por un modelo de computación en la nube, prescindiendo de servidores físicos junto con toda la infraestructura física y tecnológica que implican. De esta manera, se aprovechan todas las ventajas que ofrecen este tipo de sistemas. Para empezar, usando las versiones gratuitas de prueba de las diferentes tecnologías que se emplearán, las cuales son más que suficientes para el desarrollo y pruebas iniciales de este proyecto, se eliminarán los gastos que implicarían la adquisición y el funcionamiento (electricidad necesaria), además del trabajo posterior que implica en esfuerzo humano el mantenimiento de un sistema de este calibre. Este tipo de infraestructura, ofrece también la posibilidad de un escalado fácil y sencillo. De esta manera, si el número de usuarios creciese demasiado, se podría adaptar fácilmente el sistema de manera que este contase con mayor potencia tanto computacional como de memoria. El uso de estas tecnologías, además, aportan gran cantidad de herramientas tanto para el mantenimiento como para ayudar en el desarrollo y el despliegue del servicio. Estas herramientas permitirán ofrecer una mejor experiencia de usuario, debido a su robustez, seguridad y disponibilidad.

Como se ha mencionado anteriormente, este proyecto esta especialmente orientado para personas que presentan dificultades en la comunicación escrita. Existen para este tipo de personas algunos recursos que, aunque no suficientes, si son muy útiles. Estos recursos se denominan Sistemas Aumentativos y Alternativos de Comunicación (SAAC), y están especialmente diseñados para ayudar en el propósito de la comunicación de estas personas. El portal Aragonés de la comunicación aumentativa y alternativa [4], define los SAAC como "formas de expresión distintas al lenguaje hablado, que tienen como objetivo aumentar (aumentativos) y/o compensar (alternativos) las dificultades de comunicación y lenguaje de muchas personas con discapacidad". Se han utilizado recursos de este portal para aportar de una mayor

[4] http://arasaac.org/index.php

funcionalidad al sistema. Estos recursos se adaptan a personas con edades y habilidades motrices, cognitivas y lingüísticas muy dispares. En concreto se han utilizado los pictogramas tanto en color como en blanco y negro.

Es necesario puntualizar que se ha desarrollado el sistema orientándolo a su uso con el idioma inglés. Esto es debido a que algunos de los recursos y herramientas utilizadas funcionan únicamente en inglés, además de que la búsqueda de recursos como imágenes es más rica debido a que gran parte de la gente añade las etiquetas e información necesaria para poder buscar estas imágenes en inglés.

Finalmente, la plataforma móvil escogida para el desarrollo de la aplicación es Android. Esto se debe a que además de ser una de las plataformas más utilizadas, los dispositivos Android son más asequibles que aquellos de otras plataformas como iOS, existiendo en Android un abanico más amplio donde escoger, encontrando desde dispositivos de gama media/baja relativamente asequibles, hasta aquellos de gama alta comparables a aquellos de la plataforma iOS.

Finalmente mencionar que aunque a lo largo del documento aparecerán fragmentos de código correspondientes al proyecto (tanto aplicación como servidor), eestos fragmentos no mostrarán todo el código desarrollado, sino que se limitarán a mostrar aquello que es más relevante y que aporta una mayor información para el tema que se esta tratando. Se adjuntará no obstante una copia de todo el código referente a la aplicación y al servidor en el CD junto con la copia en digital de este documento. En este cd se podrá consultar todo el código sin censuras para quien así lo desee.

1.1 Estructura del documento

A continuación, se muestra la estructura y distribución de este documento, nombrando cada uno de los capítulos así como una breve descripción sobre lo que contiene cada uno de estos.

Capítulo 1:Introducción

Es el presente capítulo, y en el se presenta una breve introducción al proyecto, así como la estructura que compondrá este documento, indicando que se va a encontrar en cada uno de los capítulos.

Capítulo 2:Objetivos

En este capítulo se presenta el principal objetivo que pretende alcanzar este proyecto, así como los subobjetivos en los que se puede dividir este y que permitirán alcanzar su consecución.

Capítulo 1:Antecedentes

Se presentan los fundamentos teóricos sobre los que este proyecto esta basado. En concreto se presenta toda la información referente a los sistemas de recomendación

necesaria para entender posteriormente el por qué y el cómo de su uso.

Capítulo 4:Método de trabajo

En este capítulo se expone la metodología de trabajo que se ha empleado, explicando dicha metodología y justificando su uso, además de presentar los primeros resultados de la aplicación de esta. También se indican las herramientas software y hardware que se han utilizado para la elaboración del proyecto.

Capítulo 5:Arquitectura

Se muestra la arquitectura global del sistema, así como las diferentes fases que se han seguido aplicando la metodología escogida en el capítulo anterior. Se realiza también, una descripción en profundidad sobre el algoritmo para la recomendación de imágenes que se ha empleado, así como una justificación del por qué se ha escogido este algoritmo y de las técnicas propias de los sistemas de recomendación que se aplican.

Capítulo 6:Resultados

En este capítulo se analizan los resultados obtenidos tras la aplicación de las diferentes fases de la metodología, mostrando un ejemplo de ejecución del sistema y analizándolo.

Capítulo 7:Conclusiones y trabajos futuros

Este es el último capítulo, y en el se describen las conclusiones que se pueden extraer tras el desarrollo tanto del proyecto como de la documentación asociada, así como una valoración personal y las posibles ampliaciones que se han planteado para realizar en el futuro.

Capítulo 2

Objetivos

En este capítulo se establecerá el principal objetivo que se pretende alcanzar mediante este proyecto, desglosando este a su vez en objetivos específicos que será necesario alcanzar para su consecución.

2.1 Objetivo general

El objetivo principal que se pretende lograr en este Trabajo Fin de Grado (TFG) es el desarrollo de un sistema de recomendación de imágenes, que integrado en una aplicación de mensajería, sea capaz de a partir de una entrada proporcionada por el usuario, sugerir aquellas imágenes que, en base a ciertos criterios que se especificará a continuación, representen mejor aquello que el usuario desea transmitir. Este sistema estará principalmente orientado para aquellas personas que por un motivo u otro presenten dificultades a la hora de trabajar con las aplicaciones clásicas de mensajería.

Las imágenes a recomendar serán principalmente pictogramas, que resultan más intuitivos que las imágenes tradicionales a la hora de representar conceptos, especialmente para gran parte del público para el que se diseña esta aplicación. Sin embargo, estos pictogramas a veces pueden no ser suficientes, o puede darse el caso de que simplemente no se disponga de un pictograma adecuado para la entrada del usuario. En estos casos será necesario hacer uso de recursos que se encuentren en la red, donde será muy importante hacer uso de fuentes fiables de información.

Respecto al procedimiento a seguir por parte del sistema, lo primero que será necesario es preprocesar la entrada, de manera que la búsqueda de imágenes no quede limitada a la palabra o palabras empleadas por el usuario. A continuación, será necesario establecer qué imágenes tienen más posibilidades de representar aquello que el usuario pretendía transmitir. Será necesario establecer perfiles entre los diferentes usuarios de la aplicación. Estos perfiles permitirán relacionar a usuarios con gustos similares de manera que sea más fácil sugerir imágenes que ya han sido previamente seleccionadas por otro usuarios con un perfil similar. De esta manera, el sistema deberá mejorar con la cantidad de usuarios y el uso que se haga de este.

Es necesario incidir en que el objetivo principal del proyecto es el de desarrollar un sis-

tema de recomendación, y no una aplicación de mensajería, por lo que la mayor parte del esfuerzo se centrará en el desarrollo de este sistema, careciendo la aplicación de algunas de las funcionalidades que caracterizan a las populares aplicaciones de mensajería ya existentes en el mercado.

2.2 Objetivos específicos

El objetivo discutido anteriormente puede ser desglosado en los siguientes objetivos específicos.

2.2.1 Desarrollo de una interfaz para la comunicación entre los usuarios

Será necesario desarrollar una aplicación de mensajería. Esta aplicación deberá presentar las características básicas comunes a toda aplicación de mensajería, como una lista con los contactos del usuario que tengan la aplicación. También, contará con un chat que permita la comunicación tanto mediante texto como mediante imágenes, y que permita al usuario realizar la entrada de manera oral y/o escrita. Para el almacenamiento de mensajes intercambiados por el usuario con otros se dispondrá de una base de datos.

2.2.2 Diseñar un protocolo de comunicación entre dispositivos

Se desarrollará un protocolo de comunicaciones, el cual proporcionará una estructura que permita la comunicación entre los diferentes dispositivos y usuarios que hagan uso de la aplicación. Para ello contará con los siguientes componentes. Por un lado, un almacén de datos situado en la red, el cual contendrá a todos los usuarios de la aplicación junto con los datos de estos, permitiendo a los distintos usuarios descubrir a aquellos de entre sus contactos que también disponen de la aplicación. Por otro lado, se deberá establecer un mecanismo que se encargue de hacer llegar los mensajes que envía un determinado usuario de la aplicación a otro permitiendo la comunicación entre estos.

2.2.3 Implementar un repositorio flexible de imágenes

Será necesario desarrollar un sistema capaz de almacenar y gestionar la información relativa a estas imágenes. Deberá permitir la posibilidad de añadir nuevas imágenes junto con información relativa de estas, además de poder modificar información de imágenes ya almacenadas.

2.2.4 Desarrollo de un mecanismo para la obtención y análisis de nuevas imágenes

Se deberá diseñar un mecanismo capaz de realizar búsquedas de imágenes en sitios de terceros cuando las imágenes propias sean insuficientes. Debe existir la posibilidad de poder configurar los sitios de terceros sobre los que se realizan esas búsquedas. También será necesario poder realizar un análisis de estas nuevas imágenes, con el fin de extraer la máxima

información posible de estas.

2.2.5 Diseño de un algoritmo matemático para la recomendación de imágenes

Se deberán determinar aquellas variables que resultan más determinantes a la hora de recomendar las imágenes. Para ello se valorará si estas variables dependen de características propias de las imágenes únicamente, o si también influyen en ellas características propias de los usuarios. Con todos estos datos será necesario establecer un algoritmo que, valorando todos estos datos en el grado adecuado, sea capaz de determinar qué imágenes podrán satisfacer en mayor grado las necesidades del usuario y proceder así a su recomendación.

Capítulo 3

Antecedentes

En este capítulo se discutirán algunos de los conceptos que se han empleado para la elaboración de este proyecto. En concreto, se hablará sobre los sistemas de recomendación, explicando en que consisten y profundizando especialmente en los sistemas de recomendación híbridos.

3.1 Sistemas de recomendación

En la vida diaria, las personas recurren a las recomendaciones o críticas por parte de otras cuando sus experiencias personales o conocimientos sobre el tema son insuficientes. Los sistemas de recomendación actuales desempeñan un papel similar a este proceso. En 1997 Resnick y Varian [RV97] definieron por primera vez los sistemas de recomendación como sistemas en los que "la gente proporciona recomendaciones como entrada al sistema, el cual luego se encarga de agregar y redirigir estas a los destinatarios adecuados". Posteriormente, se ha ampliado esta definición [Bur02], considerando sistemas de recomendación aquellos que generan recomendaciones personalizadas o que son capaces de guiar al usuario hacia elementos que sean de su interés dentro de un amplio abanico de posibilidades. Es esta característica de individualización o recomendación personalizada lo que distingue a estos sistemas de otros como los buscadores simples y los sistemas de recuperación de información.

Estos sistemas son claves hoy en día, donde toda la información que se encuentra en la red en prácticamente cualquier portal trasciende la capacidad de los usuarios para buscar y seleccionar de entre todos los elementos por sí mismo. Un ejemplo de sistema de recomendación es el sistema de la compañía Netflix, que proporciona recomendaciones de películas en base a las anteriores visualizaciones de los usuarios. Esta compañía realizó en 2006 una competición en la que retó a la comunidad, a desarrollar un sistema de recomendación que fuese capaz de derrotar al de la compañía *Cinematch* [BEL+07]. Para ello, ofreció al público una pequeña fracción de sus datos en la que se encontraban valoraciones anónimas de usuarios sobre películas. Se inscribieron 20.000 equipos, de los cuales 2.000 presentaron al menos una solución. En 2009 se concedió un premio de 1.000.000 de dolares a un equipo que mejoró la precisión de Cinematch en un 10 %. Todos estos números pueden servir para

hacerse una idea del potencial que ofrecen los sistemas de recomendación hoy en día.

3.1.1 Evolución histórica de los sistemas de recomendación

El primer sistema de recomendación que se desarrollo fue Tapestry [GNOT92] a comienzos de los 90, y sus desarrolladores lo denominaron con el término *Filtro colaborativo*, el cual fue adoptado por otros desarrolladores, pero que en la actualidad ha quedado en desuso debido principalmente a que los actuales sistemas no solo filtran aquellos elementos o productos que no son deseables, sino que tratan de sugerir aquellos que puedan aportar mayor valor, además de que como se verá más adelante, algunos de los tipos de sistemas de recomendación que fueron apareciendo posteriormente, no tenía en cuenta las opiniones de otros usuarios, pudiendo, por tanto, no ser colaborativos. Este primer sistema pionero, tenía como propósito el de filtrar el correo electrónico, así como artículos de noticias online. Es especialmente en este campo del filtrado de noticias donde los primeros sistemas de recomendación tuvieron mayor auge.

Pese a que este sistema es el primero considerado de recomendación, algunos años antes ya se propusieron trabajos y teorías sobre sistemas para el filtrado de elementos, como el trabajo de Housman y Kaskela [HK70] que buscaba mantener a los científicos informados de los nuevos documentos mediante un matching entre las palabras clave que estos establecían de su interés, y el contenido de los nuevos artículos. Sin embargo, esta estrategia no logró los resultados deseados. Otra aproximación fue la de la creación de modelos de usuario que aparece en el trabajo de Allen [All90]. Por otra parte, el sistema *The information Lens* [MGT86] planteó un enfoque diferente hacia los sistemas de recomendación, estableciendo reglas que se apoyaban en la estructura que tienen la mayoría de los mensajes de correo electrónico y en palabras clave de estos, permitiendo a los usuarios utilizar estas estructuras como plantillas de manera que fuese más fácil para estos establecer los mecanismos de filtro. Todos estos sistemas fueron dando forma a los diferentes tipos y categorías de sistemas de recomendación que existen hoy en día.

3.1.2 Clasificación de los sistemas de recomendación

Desde su aparición, se han realizado distintas clasificaciones de los sistemas de recomendación. En este caso se va a mostrar la clasificación que realizaron Resnick y Varian [RV97]. De este modo, se establecen una serie de características según las cuales se podría realizar la clasificación de estos sistemas.

En primer lugar se establecen las siguientes características técnicas:

- El contenido de la recomendación, es decir, la valoración a los distintos elementos a recomendar. Esta valoración puede ser tan simple como un valor binario (recomendado o no) o algo más complejo como un un porcentaje o un texto.
- El modo en el que se realizan o recogen las recomendaciones. Las recomendaciones

pueden ser realizadas de manera explícita, pero también pueden ser recogidas de manera implícita por el sistema, por ejemplo, analizando las preferencias de los usuarios, las búsquedas y visitas de estos a diferentes portales o las compras previas de estos. Existen mecanismos muy útiles para poder recoger estas recomendaciones implícitas como las cookies en los sistemas web.

- La identidad de los recomendadores. Esta puede ser la identidad real, un pseudónimo o bien una identidad anónima. Como se verá más adelante, los usuarios prefieren conservar su privacidad en numerosas ocasiones, y pueden ser reacios a compartir información de carácter sensible aun cuando esta información puede ser crucial para el sistema de recomendación, por esta razón es necesario ofrecer mecanismos que les permitan conservar dicha privacidad.
- Las técnicas de recomendación, es decir, la manera en la que se relacionan las recomendaciones con aquellos que buscan recomendación. Esta es una de las características que permiten mayor flexibilidad en este tipo de sistemas y será analizada en profundidad más adelante.
- La finalidad de las evaluaciones. Uno de los usos es el de descartar o sugerir elementos. Otra finalidad puede ser la de ordenar los elementos recomendados según un peso, o mostrar para cada elemento el nivel de recomendación.

Además de las características técnicas, otro de los elementos que caracterizan un sistema de recomendación es su dominio, es decir, los elementos o ítems que se recomiendan, y el público que realiza o recibe las distintas recomendaciones.

En lo referente a los elementos sobre los que se aplican las recomendaciones es importante, en primer lugar, definir el tipo de los elementos que se están recomendando. En el Cuadro 3.1 se muestran algunos ejemplos de sitios y los elementos que recomiendan. Otro factor importante es el volumen de los elementos que se recomiendan, así como la frecuencia con la que se generan y desaparecen. Es necesario conocer y tener en cuenta estos parámetros ya que se deben de tratar de manera muy distinta unos elementos que se generan con gran frecuencia y tienen un tiempo de vida corto, como pueden ser las noticias de un medio electrónico, en el que es muy importante poder recomendar dichas noticias en un tiempo acotado, a la necesidad de recomendar por ejemplo películas o libros. Es aquí donde asumen gran importancia las técnicas de recomendación.

Por otro lado, se encuentran los costes que implican el proceso de recomendación y los resultados. Es posible, que el coste de fallar en la recomendación como por ejemplo al no recomendar un buen ítem, sea más alto que el coste de recomendar un ítem incorrecto o viceversa. También es muy relevante el tiempo que conlleva realizar una recomendación. Pueden existir sistemas de recomendación que deban realizar la recomendación en un tiempo crítico, y por tanto, el coste de un análisis intensivo sea mayor que el de obviar buenos ítems

o recomendar algunos que no sean los adecuados. Este factor, es altamente dependiente de la configuración de los características técnicas mencionadas anteriormente.

Sitio	Elementos recomendado
Amazon	Libros/otros productos
Facebook	Amigos
WeFollow	Amigos
MovieLnes	Películas
Nanocrowd	Películas
Jinni	Películas
Findory	Noticias
Digg	Noticias
Zite	Noticias
Meehive	Noticias
Netflix	DVDs
CDNOW	CDs/DVDs
eHarmony	Citas
Chemistry	Citas
True.com	Citas
Perfectmatch	Citas
CareerBuilder	Trabajos
Monster	Trabajos
Pandora	Música
Muffin	Música
StumbleUpon	Páginas Web

Cuadro 3.1: Sitios web y elementos que recomiendan (RESNICK [LMY+12])

En el caso de los usuarios involucrados en el proceso de las recomendaciones, tanto aquellos que las realizan como los que las consumen, es necesario conocer los perfiles de estos. Por ejemplo, se debe saber si los usuarios tienden a realizar recomendaciones de numerosos elementos similares, o por el contrario, evalúan solo elementos muy específicos dando lugar a diferentes conjuntos de recomendaciones. También es importante conocer la cantidad de usuarios que componen o compondrían el sistema, y la variedad respecto a los gustos de los usuarios. El sistema y las técnicas a aplicar varían de manera muy notable si existe una gran cantidad de usuarios con gustos similares, o por el contrario existen muy pocos usuarios con perfiles muy especializados y concretos.

3.1.3 Técnicas de recomendación

Como se ha mencionado anteriormente existen varias técnicas de recomendación que permiten adaptarse a la situación en cuestión. Algunos autores prefieren realizar una clasificación sobre estas técnicas de recomendación más acotada que la que se va a utilizar en este documento[BS97], diferenciando entre tres grandes grupos de sistemas de recomendación. Por un lado, distinguen principalmente si son sistemas en los que influyen las valoraciones

u opiniones de otros usuarios, a los que se consideran sistemas colaborativos. Si por el contrario lo único que afecta a la recomendación es la valoración previa del usuario sobre los elementos que se recomiendan y las relaciones entre estos elementos, el sistema se considera basado en contenido. Además, si son sistemas que combinan estrategias de los dos anteriores, estos se consideran híbridos. Si bien esta es una clasificación perfectamente válida, y abarca las técnicas más importantes y utilizadas, en este documento se han querido nombrar también otras técnicas utilizadas.

A la hora de elegir una de estas técnicas, es necesario tener en cuenta principalmente tres factores. Por un lado, se encuentran los datos y la información de la que disponemos antes de comenzar el proceso de recomendación, como pueden ser las valoraciones de ciertos usuarios de los ítems, o información sobre dichos ítems. Por otro lado, está la entrada que realiza el usuario, es decir, la información que este comunica al sistema con el propósito de obtener una recomendación. Finalmente, es necesario disponer mecanismo o algoritmo que sea capaz de combinar los dos elementos anteriores para poder llevar a cabo la recomendación. Mediante estos tres factores, se pueden definir un total de cinco técnicas de recomendación. Considerando **U** como el conjunto de los usuarios de los que se conocen las preferencias, **I** el conjunto de los ítems sobre los que existen valoraciones, **u** el usuario para el que realizar la recomendación e **i** el elemento en cuestión que se está considerando para ser recomendado, en el cuadro 3.2 se muestra a grandes rasgos como funcionan estas técnicas en función de los factores anteriores.

A continuación se muestran en más detalle las diferentes técnicas.

Colaborativa

La técnica colaborativa es probablemente la más usada dentro de este tipo de sistemas. En los sistemas que usan dicha técnica, cada usuario tiene un perfil en el que se encuentran las evaluaciones que ha realizado este de cada elemento. El funcionamiento de esta técnica, consistiría en comparar al usuario, o en este caso su perfil, con el de otros usuarios a los que se puede considerar sus vecinos, para encontrar las similitudes entre ellos. Usando estas similitudes, se puede extrapolar la información sobre las evaluaciones u opiniones de estos usuarios sobre el ítem que se pretende evaluar o recomendar. De esta manera, si el ítem en cuestión ha sido positivamente valorado por aquellos usuarios que se consideran vecinos del usuario para el que se quiere realizar la recomendación, este ítem será también recomendado a dicho usuario.

Un ejemplo simple de como sería un perfil de usuario en estos sistemas consiste en una estructura de datos en el que se encuentran los elementos evaluados junto con las valoraciones realizadas por el usuario. Los sistemas que emplean esta técnica están pensados para generar relaciones a largo plazo, ya que por norma general mejoran su funcionamiento conforme aumentan las recomendaciones. Es por esto, que en algunos casos es recomendable

Técnica	Información previa	Entrada	Algoritmo
Colaborativo	Evaluaciones de **U** de los elementos en **I**.	Evaluación de **u** de los elementos en **I**.	Identifica a los usuarios en **U** con perfiles similares a **u**, y extrapola sus valoraciones de **i**.
Basada en contenido	Características de los elementos en **I**.	Las valoraciones de **u** de los elementos en **I**.	Generar un clasificador que adapte el comportamiento respecto a las valoracioes de **u** y lo use en **i**.
Demográfica	Información demográfica sobre **U** y sus evaluaciones de los elementos en **I**.	Información demográfica sobre **u**.	Identificar aquellos usuarios que son demográficamente similares a **u** y extrapolar sus evaluaciones de **i**.
Basada en utilidad	Características de los elementos en **I**.	Una función de utilidad sobre los elementos en **I** que describa las preferencias de **u**.	Aplicar la función sobre los elementos y determinar la clasificación de **i**.
Basada en conocimiento	Características de los elementos en **I**. Conocimiento de como estos elementos cumplen las necesidades del usuario.	Una descripción de las necesidades o intereses de **u**	Inferir la afinidad del item **i** con las necesidades de **u**.

Cuadro 3.2: Técnicas de recomendación (BURKE [Bur02])

establecer mecanismos que sean capaces de modificar estos datos en función del tiempo de manera ajena al usuario. Esto es debido a que el usuario puede haber variado sus gustos u opiniones a lo largo de este tiempo, y las evaluaciones pueden no ser fiables.

Respecto a los algoritmos y medidas utilizados en este tipo de sistemas de recomendación destacan especialmente dos tipos, por una parte, se encuentran los algoritmos basados en memoria[YST+04]. En este tipo de algoritmos, se tienen en cuenta todas las valoraciones realizadas por un usuario. Debido a esto, este tipo de sistemas presentan problemas de escalabilidad en aquellos escenarios en los que existen gran cantidad de usuarios, gran cantidad de evaluaciones por parte de los usuarios o ambos [Lee01]. Por otra parte, se encuentran los sistemas basados en modelos, en los que se crean modelos probabilísticos a partir de las valoraciones de los usuarios. Existen numerosas técnicas que han sido empleadas para la generación de estos modelos como redes bayesianas, o técnicas de aprendizaje automático como clústers o redes neuronales.

Jonathan Lee Herlocker, en su tesis[Her00], realizó un estudio de esta técnica, comparando diferentes algoritmos para evaluar la similitud entre los distintos perfiles de usuarios, así como diferentes métricas para evaluar este tipo de sistemas. Tras analizar las métricas para la evaluación de la precisión de las evaluaciones, este llegó a la conclusión de que todas las métricas propuestas desempeñan una ejecución similar, y propone adoptar como estándar la métrica del error medio absoluto o Mean Absolute Error (MAE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

Donde *e* es la diferencia para cada estimación *i* entre el valor estimado y el valor real, y *n* es el total de las estimaciones.

Basada en contenido

Esta técnica es una extensión de los estudios en el filtrado de información[HSS01]. A diferencia de la técnica colaborativa en la que se establecían relaciones entre los usuarios, en esta se establecen relaciones entre los los elementos a recomendar y los elementos que ya han sido valorados por el usuario. Se comparan aquellas características o atributos que se encuentran en los diferentes ítems, estableciendo el grado de similitud entre los valores de estas. Usando esta información, es posible sugerir aquellos elementos que tienen características en común con aquellos que el usuario ha valorado positivamente.

De esta manera, el perfil de usuario se establece basándose en aquellos elementos que han gustado, o dicho de otra manera que el usuario ha valorado positivamente. Es esta característica de poseer un perfil de usuario en el que se encuentran los gustos o intereses de los usuarios, lo que diferencia este tipo de sistemas de los sistemas de filtrado de información

convencionales.

Esta técnica puede ser muy útil en un sistema de recomendación por ejemplo de películas, donde encontramos elementos que comparten las mismas características como el reparto, la temática o la fecha de lanzamiento. Además, debido a su relación con los sistemas de recuperación y filtrado de la información, y al trabajo previo desarrollado en este tipo de sistemas, este tipo de técnica es especialmente útil en aquellos casos en los que los elementos a recomendar son descritos de manera textual, o contienen características textuales [MR00].

Una de las **medidas** más usadas para el cálculo de las relaciones entre los diferentes elementos, y su relación con los perfiles de usuario, es la de la frecuencia de término - frecuencia inversa de documento TF-IDF[Rob04]. Esta es una medida numérica que busca identificar las palabras más relevantes en un documento o texto. Para ello, contrasta el número de veces que aparece una palabra en el texto en cuestión(tf), frente al número de veces que aparece en el conjunto de todos los textos (idf). Así pues, si una palabra aparece frecuentemente en un texto pero no así en el conjunto de estos, se dice que esa palabra es representativa de ese texto en concreto.

Esta técnica suele ser combinada con un sistema vectorial[SWY75] en el que se representan para cada elemento las palabras o características más relevantes junto con su relevancia o peso. Se construyen así los perfiles para cada elemento de los que se pueden recomendar. De la misma manera, es necesario establecer aquellos términos o características que al usuario le parecen interesantes, haciendo una ponderación de los distintos ítems que este ha valorado positivamente, y de los conceptos relevantes que aparecen en estos. Finalmente, se utiliza alguna heurística de puntuación para establecer la relación que existe entre los elementos a recomendar y los perfiles de usuario, o dicho de otra manera, la probabilidad de que ese elemento satisfaga las necesidades o los deseos del usuario. Esta heurística deberá tratar generalmente con vectores.

Otras medidas también usadas en este tipo de sistemas de recomendación son las técnicas de aprendizaje automático[Lan95]. Estas técnicas, a diferencia de las anteriores que están basadas en los estudios de la recuperación de información, usan modelos aprendidos de los datos mediante técnicas de aprendizaje en lugar de heurísticas, para calcular los elementos a recomendar. También se pueden utilizar clasificadores bayesianos para estas recomendaciones[PB97].

Demográfica

En este tipo de recomendadores, es necesario establecer una serie de perfiles o clases definidos de manera previa. Posteriormente, se obtiene la información personal de los usuarios y se usa esta para categorizarlos en alguna de las clases demográficas definidas anteriormente. Los mecanismos para obtener estos datos pueden ser variados, usando mecanismos para obtenerlos de manera explícita como puede ser un test, u otros mecanismos que extraigan

esta información de manera implícita. Esta técnica es similar a la colaborativa en el uso de los perfiles de los usuarios para sus recomendaciones, sin embargo la ventaja de esta sobre la anterior es que no necesita evaluaciones previas de los ítems por parte de los usuarios.

Basada en utilidad

Los sistemas que emplean esta técnica realizan recomendaciones a partir de un cálculo sobre la utilidad que tiene cada elemento para satisfacer las necesidades del usuario. Es necesario por tanto, establecer una función de utilidad para cada usuario, siendo esta función la que componga su perfil. Este tipo de técnica presenta la ventaja de poder incluir en sus cálculos y recomendaciones atributos que no son propios del elemento que se quiere recomendar como tal, como puede ser por ejemplo, la confianza en el vendedor de dicho producto en casos de comercio en línea, o el tiempo de entrega. Esto permite que estos sistemas sean capaces de valorar y negociar de acuerdo a distintas características, por ejemplo negociando el precio en función del tiempo de entrega.

Basada en conocimiento

Esta técnica realiza las recomendaciones a parir del conocimiento de como un determinado ítem satisface las necesidades o deseos de un usuario. El perfil de usuario en este tipo de recomendadores será cualquier estructura que sea capaz de representar las necesidades de los usuarios. Del mismo modo, el conocimiento sobre los elementos podrá ser representado y almacenado de diversas formas. Al igual que los recomendadores basados en utilidad, los basados en conocimiento no están diseñados para mejorar su funcionamiento con la adición de usuarios.

3.1.4 Dilemas de los sistemas de recomendación

Los sistemas de recomendación presentan principalmente dos tipos de problemas, uno que se deriva de la técnica utilizada y que va ligado a esta debido a su funcionamiento, y que por tanto tiene carácter técnico, y otro tipo de problema que depende en mayor medida del dominio, es decir, de los elementos que se recomiendan y los usuarios que hacen uso del sistema.

Problemas del dominio

Uno de los problemas en los que los usuarios tienen un papel protagonista es la **privacidad**, que aunque no es exclusivo de estos sistemas adquiere gran importancia. En estos sistemas suele ser común que la información más valiosa sea también aquella que el usuario es más reacio a compartir. En algunos sistemas, como hemos visto anteriormente, es posible la participación en las recomendaciones de manera anónima o bajo un pseudónimo[RIS+94], sin embargo, en muchas ocasiones esto puede no ser posible cuando, por ejemplo, los usuarios quieren reconocimiento al realizar una recomendación, pero no dar a conocer todos

los detalles de su información personal. Por esto es necesario en algunos sistemas el proporcionar mecanismos que presenten un grado intermedio de privacidad, ajustándose a las necesidades y deseos de los usuarios.

Otro problema que es común a todas las técnicas es el de la **creación del perfil** de usuario. Muchos usuarios al principio pueden no ver el potencial que puede llegar a tener este sistema, y por tanto, considerar la inversión de tiempo necesaria para formar su perfil demasiado alta respecto con el beneficio inmediato que le puede aportar. Es por esto que se requiere que los sistemas presenten incentivos para que estos usuarios contribuyan en la creación de los perfiles, o que por otro lado, estos sistemas de recomendación sean capaces de recopilar estas preferencias o necesidades de los usuarios sin necesidad de intervención explicita de estos. En la figura 3.1 se puede ver como la red social twitter incentiva a los nuevos usuarios a identificar aquellos campos en los que este está interesado, de manera que este pueda recomendar cuentas con ese perfil temático a los usuarios.

Figura 3.1: Cuestionario de la red social twitter al crear una nueva cuenta (Junio 2016). Fuente: https://www.twitter.com/

Cuando las recomendaciones dependen de las opiniones de los usuarios, como en el caso de las técnicas colaborativas, surge otro problema conocido como "vote early and often" o votar pronto y a menudo en español. Este fenómeno, consiste en **manipular** de alguna manera los votos o en este caso las recomendaciones. En el caso de que cualquiera pueda realizar todas las recomendaciones que desee, los creadores de contenido o proveedores de distintos productos o servicios recurrirán a votar en favor de sus elementos y en detrimento de sus competidores.

Comparación de técnicas

Existen problemas van ligados a la técnica escogida, y todas las técnicas presentan una serie de ventajas y desventajas entre sí.

Uno de los problemas más conocidos, es el problema del *comienzo frío* (cold start) o cuesta arriba (ramp-up) [Lee01]. Este problema se puede dar en dos casos, cuando aparece un nuevo usuario[RAC+02] y cuando aparece un nuevo ítem. Cuando aparece un nuevo usuario no se tiene información de él, y en el caso de ser necesario relacionarlo con algún otro usuario o con algún perfil concreto resulta complicado. El caso de un nuevo ítem es similar. Cuando aparece un nuevo ítem y apenas existen recomendaciones sobre este, aparecen problemas a la hora de relacionar este nuevo ítem con otros. En ambos casos, si el sistema requiere de recomendaciones, es necesario que este presente algún tipo de incentivo para que los usuarios realicen valoraciones sobre los elementos.

En el caso de los sistemas **colaborativos**, estos se ven afectados por los problemas tanto de nuevo usuario como de nuevo ítem, lo cual provoca que estos sistemas no puedan comenzar a funcionar hasta tener información tanto de usuarios como de ítems lo suficientemente amplia como para poder realizar recomendaciones.

Debido a sus características, este sistema presenta problemas en aquellos entornos en los que existen pocos usuarios y estos evalúan continuamente los mismos ítems. problema conocido como **dispersión** o problema de la matriz dispersa[HCZ04], o bien los usuarios tienen gustos muy diferentes. En definitiva, este tipo de sistemas resulta útil cuando existe un número suficientemente amplio de usuarios y con una variedad de gustos aceptable dentro del dominio del problema. En [Lee01], además del problema clásico del *comienzo frío*, se tratan también estos problemas relativos a la escasa cantidad de recomendaciones.

Debido al enfoque de este tipo de sistemas en los que se depende enormemente de los usuarios, su actitud y predisposición, estos son especialmente sensibles a los problemas que se han analizado en la sección 3.1.4. Es especialmente necesario en estos sistemas el incentivar a los usuarios a realizar las valoraciones de los productos, para evitar tanto los problemas del *comienzo frío* como los de la dispersión. También, es en estos sistemas donde existe una mayor problemática con la privacidad, debido a que se utiliza la información de estos en las recomendaciones a otros usuarios desconocidos.

Una de las ventajas que presentan estos sistemas respecto a otros es principalmente su capacidad para poder recomendar elementos que pese a que no estén dentro del mismo grupo, pueden estar relacionados de alguna manera. Por ejemplo, puede darse la situación de que a todos aquellos a los que les gusta la música Jazz les guste también el mismo grupo de Rock, y este sistema sería capaz de recomendar dicho grupo, mientras que otros como por ejemplo el basado en contenido no podría. Otra gran ventaja de esta técnica es la de ser independiente de la representación a nivel de computador de los elementos que recomienda. Debido a esto es frecuentemente usado en la recomendación de elementos u objetos complejos como libros, música, imágenes o elementos multimedia en general.

Los sistemas **basados en contenido** sufren de manera menor del problema de la dispersión

debido a que solo tienen en cuenta las valoraciones del propio usuario y no de todos los usuarios. A pesar de esto, tienen el mismo problema de comienzo en frío que tenían los sistemas colaborativos cuando aparece un nuevo usuario. Es con la aparición de un nuevo ítem con lo que este tipo de sistemas no tiene problema. A pesar de que esta aparición de nuevos elementos no presenta un problema para estos sistemas, lo hace por otra parte el formato que deben presentar estos elementos. Debido a que están fuertemente ligados con las técnicas de filtrado y recuperación de información, presentan el inconveniente de funcionar de manera muy pobre en aquellos escenarios en los que los sistemas a recomendar no se encuentran representados de manera textual. Además, las características y valores de estos elementos deben presentar una estructura común, de manera que puedan consultarse y recuperarse estos atributos de manera automatizada.

Otro problema que tiene en común con los colaborativos, surge como consecuencia de que estos sistemas generalmente no deben recomendar un objeto que el usuario ya ha valorado, y que por tanto, se entiende que dispone de él o lo conoce lo suficiente. Aunque esta condición no es obligatoria, ya que el puede ser útil recomendar algo que el usuario ya haya seleccionado anteriormente, sí puede no ser deseada. El problema que se presenta aquí es el de distinguir cuando dos elementos son lo suficientemente parecidos para ser considerados iguales y no ser recomendados al mismo usuario cuando este ya ha consumido o valorado uno de ellos.

La sobre especialización es otro problema común. Este tipo de sistemas presentan carencias a la hora de sugerir novedades a los usuarios que les puedan interesar, y además presentan dificultades a la hora de adaptarse a los cambios de gustos o intereses de los usuarios. Es necesario pues, que en algunos casos, se establezcan mecanismos que se encarguen de introducir este factor de aleatoriedad, de manera que se sugieran nuevos elementos para así ampliar el abanico de gustos e intereses del usuario sin estancarse en un solo campo.

Los sistemas **demográficos** también sufren el problema del *comienzo frío*, pero solo respecto a la aparición de nuevos ítems. También comparten otro de los problemas de los colaborativos que es el de identificar a usuarios que se salen de lo común, es decir, que o bien no tienen características en común con ninguno de los actuales grupos, o presentan algunas características de cada grupo, pero no las suficientes como para categorizarlos en uno concreto. Pese a no tener el problema de *comienzo frío* con los nuevos usuarios, necesitan recopilar información sobre los distintos perfiles antes de comenzar a operar. Esta recopilación y formación de perfiles es su principal diferencia respecto a los sistemas colaborativos.

Pese a que estos sistemas mencionados anteriormente, que están basados en el aprendizaje durante la ejecución, parecen tener muchos problemas especialmente al inició, presentan la ventaja de ser más flexibles y adaptativos que las dos técnicas que se verán a continuación. Tanto los sistemas basados en utilidad como los basados en conocimiento carecen de los problemas de *comienzo frío* y de dispersión ya que no aprenden durante su ejecución. Sin

embargo, en ambos casos es necesario un gran trabajo previo.

En el caso de los sistemas **basados en utilidad**, pese a carecer del problema del *comienzo frío*, necesitan crear la función de utilidad considerando todas las características del objeto. Esto supone un gran trabajo al inicio y mucha interacción con el usuario, lo cual supone una desventaja para usuarios inexpertos o usuarios que no quieren invertir demasiado tiempo en esto, pero que facilita en gran medida el filtro y selección de aquellos usuarios expertos o que buscan elementos con unas características muy específicas. Como se mencionó en la sección anterior, la gran ventaja de estos sistemas es poder considerar aquellas propiedades que no son intrínsecas al objeto, como el tiempo o el formato del envío. Esto los hace especialmente útiles en sistemas de comercio online.

Por otra parte, los sistemas **basados en conocimiento** comparten el problema de la recolección de datos o conocimiento con los sistemas clásicos basados en el conocimiento, en los que se necesita de un experto del que extraer información y de un ingeniero del conocimiento que sea capaz de extraer dicha información. Respecto a los tipos de conocimiento que estos sistemas requieren se pueden categorizar en tres. Por una parte encontramos el conocimiento sobre los ítems o elementos que se van a recomendar. También encontramos el conocimiento sobre el usuario y las necesidades de este. Finalmente está el conocimiento funcional, que permite establecer relación entre las necesidades del usuario y las características de los objetos.

3.1.5 Sistemas de recomendación híbridos

En la sección anterior se ha visto que cada tipo de sistema de recomendación tiene unas fortalezas y debilidades, las cuales, en muchos casos, son complementarias. Es decir, algunas técnicas proveen de aquello que les falta a otras y viceversa. Siendo esto así, surge una idea, la de combinar varios de estas técnicas, tratando de buscar sacar el máximo potencial posible de cada una. En general la mayoría de estos sistemas tratan de solventar los problemas de los sistemas colaborativos del nuevo usuario y el nuevo item combinándolo con alguna otra técnica.

Existen varios mecanismos a la hora de combinar las técnicas vistas anteriormente. Estos mecanismos permiten combinar no solo dos de las técnicas anteriores, sino tantas como se desee. Además, algunos de estos mecanismos permiten combinar las salidas de otros entre sí o con otras técnicas de recomendación, ofreciendo de esta manera gran posibilidad de personalización y adaptación al problema en cuestión. En [Bur07] se realiza un análisis de sistemas que emplean algunas de estrategias de hibridación, comparando estos sistemas y analizando que tipo de técnicas funcionan mejor según que estrategias o métodos se empleen de los que se muestran a continuación.

A continuación se muestran los mecanismos disponibles:

Ponderado

Los sistemas ponderados son aquellos en los que la utilidad o puntuación de un determinado elemento para un usuario, es calculado a partir de la puntuación proporcionada por las diferentes técnicas utilizadas. Se pueden usar diferentes aproximaciones a la hora de combinar estas puntuaciones, desde una simple media ponderada que puede ir variando conforme varíen la información disponible, a sistemas de consenso[Paz99].

Conmutado

En un sistema conmutado se alterna entre diferentes técnicas de recomendación dependiendo de alguna condición [TC00]. De esta manera, puede usarse un sistema basado en contenido mientras no existan suficientes valoraciones para relacionar usuarios, y más adelante, conforme se comiencen a recoger más y más valoraciones pasar a utilizar técnicas colaborativas. También puede emplearse para recurrir a otra técnica cuando la primera falle o no genere un resultado lo suficientemente bueno.

Combinado

Este tipo de sistema, combina las recomendaciones proporcionadas por varios tipos de técnicas, y se las ofrece al usuario[CS00]. De esta manera, se pueden utilizar varias técnicas que usen información diferente, de manera que se superen problemas como el del estancamiento o sobre especialización que traen consigo los sistemas basados en contenido y problemas por ejemplo de nuevo item de los sistemas colaborativos.

Combinación de características

Este tipo de mecanismo busca combinar las características de varios tipos de técnicas en una sola. Por ejemplo, se puede enriquecer la información de los sistemas basados en contenido a partir de las relaciones existentes entre los usuarios que han valorado dichos elementos, las cuales son extrapoladas de la aplicación de las técnicas colaborativas o demográficas.

Cascada

En los sistemas que emplean este tipo de mecanismos, se realizan filtros progresivos en los que se van seleccionando y filtrando mediante varias técnicas aquellos elementos que se recomendarán al usuario. La recomendación se realiza mediante etapas, donde cada técnica realiza una selección del conjunto de elementos que le proporciona la anterior, o en caso de ser la primera de todo el conjunto disponible. Esto puede servir para liberar de carga computacional a aquellas técnicas que sean más costosas, utilizando un primer filtro que elimine aquellos elementos que con total seguridad no van a ser deseados por el usuario.

Aumento de las características

La salida proporcionada por la aplicación de alguna técnica de recomendación se convierte en la entrada de otra en este tipo de sistemas[SKB+98]. De esta manera, la valoración producida por una técnica se incorpora al proceso de recomendación de la segunda. Difiere del mecanismo en cascada en que esta técnica de cascada no usa las salidas producidas por estas en su valoración, sino que simplemente la segunda técnica opera sobre aquellos elementos que han pasado el primer filtro, sin tener ningún conocimiento del proceso seguido ni de lo que existía anteriormente. Tampoco se tiene acceso en el mecanismo de cascada a la valoración proporcionada por el mecanismo anterior.

Meta-nivel

En este tipo de sistemas[BS97], lo que se pasa como entrada a la siguiente técnica es el modelo generado por la primera, y no la salida como ocurría en el método de aumento de las características. Es especialmente útil para que el segundo método trabaje con información más compacta que si trabajase sobre el total de la información, evitando de esta manera el problema de trabajar con gran cantidad de datos. Además, al trabajar con información más compacta también se reduce el problema de la matriz dispersa que se mencionaba anteriormente, ya que en el modelo que se pasa como resultado, se eliminan muchas dimensiones y se presentan modelos que agrupan diversas valoraciones de elementos.

Capítulo 4

Método de trabajo

A lo largo de este capítulo se planteará la metodología escogida para el desarrollo de este proyecto, justificando su elección y explicando sus principales características así como el primer planteamiento para afrontar el problema. También se mencionarán las herramientas software y hardware empleadas en la realización tanto del proyecto como de esta documentación.

4.1 Metodología de desarrollo

Para la realización de este proyecto se ha escogido una metodología ágil [IIS04], en concreto se va a aplicar el marco Scrum [Sch04]. Las metodologías ágiles se caracterizan por el desarrollo del proyecto mediante interacciones incrementales que finalizan con un entregable, el cual aporta una determinada funcionalidad de manera que pueda ser evaluado por el usuario. Mediante estos entregables tempranos se pretende reducir el riesgo de desarrollar un producto que no se ajuste a las necesidades del usuario, permitiendo añadir cambios y adaptarse a los requisitos de este durante su desarrollo.

4.1.1 Scrum

Scrum es un marco de desarrollo ágil basado en ciclos de desarrollo iterativos e incrementales para la gestión y desarrollo de proyectos. Scrum establece un conjunto de prácticas y roles que se deberán emplear en el desarrollo de un proyecto con el objetivo de minimizar los riesgos que conlleva este desarrollo. Una de las principales características de este modelo, es que se busca el desarrollo colaborativo, en el que todo el equipo trabaja de manera autoorganizada y sin la intervención de agentes externos al proyecto o producto que se desea desarrollar. El equipo de desarrollo de Scrum debe trabajar de manera muy cercana, comunicándose entre sí continuamente, y realizando revisiones de todo lo que se ha desarrollado con gran frecuencia para asegurar la mayor calidad posible del producto. Debido a su capacidad de adaptabilidad a las necesidades del usuario y a los cambios, es especialmente útil en entornos donde existe volatilidad en las necesidades de los usuarios, y donde se necesitan obtener resultados tangibles en un tiempo corto.

Como se ha mencionado anteriormente *"Scrum no es un proceso o una técnica para la*

creación de productos; si no un marco dentro del cual puedes emplear diferentes técnicas y procesos" [1]. Scrum está fundamentalmente basado en el enfoque empírico, revisando continuamente el trabajo que se ha llevado a cabo para mejorar. Está basado principalmente en tres pilares. Por un lado encontramos la **transparencia**, que implica que todos los aspectos importantes del proyecto deben ser conocidos por todos los integrantes del equipo Scrum, y que además deben existir estándares conocidos y seguidos por los implicados, de manera que todos los integrantes entiendan aquello que se está representando. Por otro lado debe existir una **inspección**, que pese a que no debe ser tan frecuente que se interponga en el desarrollo del trabajo, debe ser suficiente como para asegurar que el producto se desarrolla con la calidad adecuada. Finalmente, el marco Scrum y especialmente el equipo que implementa dicho marco, deben ser capaces de **adaptarse** a los cambios que tras las inspecciones se consideren necesarios para asegurar el correcto desarrollo del producto.

Roles en Scrum

Como se ha mencionado anteriormente, solo aquellos miembros que se encuentren dentro del proyecto serán los encargados de tomar decisiones que afecten a dicho proyecto, sin intervención de terceras personas ajenas a este. Así pues, el equipo de Scrum es auto-suficiente y auto-organizado, y es este equipo el encargado de tomar todas las decisiones que afecten a este desarrollo. Scrum define tres grandes roles dentro de su estructura, aunque luego pueda existir una distribución más amplia dentro de estos.

- **Product Owner (Propietario del producto):** Es la persona que representa al cliente y a los stakeholders (o personas interesadas en el producto). Es el encargado de maximizar el valor del producto, asegurándose de que las necesidades se cumplen de la manera más exitosa posible. Para se encarga de redactar las Historias de usuario, dotarlas de prioridad y añadirlas al Product Backlog.
- **Scrum Master (Facilitador):** Es el encargado de facilitar la realización del proyecto por parte del equipo. Debido a la característica autoorganizativa de Scrum no representa el papel clásico de líder de proyecto, sino que actúa como facilitador del equipo, eliminando o minimizando aquellos obstáculos que puedan perjudicar al desarrollo del proyecto. También es el encargado de hacer cumplir el marco de Scrum, así como de liderar o dirigir las reuniones para asegurarse de que son productivas y se desarrollan mediante la idelogía Scrum. Ayuda al Product Owner en la realización del Product Backlog, de manera que se tenga claro que elemento es necesario desarrollar a continuación para que el equipo pueda seguir trabajando continuamente.
- **Development Team (Equipo de desarrollo):** Este equipo es el encargado de materializar las historias de usuario y las necesidades del proyecto en entregables iterativos ya finalizados para que sean evaluados. Dentro de este equipo debe haber personas con

[1] http://www.scrumguides.org/scrum-guide.html

las capacidades adecuadas para el desarrollo de todas las funcionalidades necesarias en cada incremento. Debido a esto los equipos se componen de personal con capacidades variadas y complementarias. Estos equipos deben ser de un tamaño reducido de manera que la autoorganización pueda funcionar, y pueda existir una gran comunicación entre el equipo.

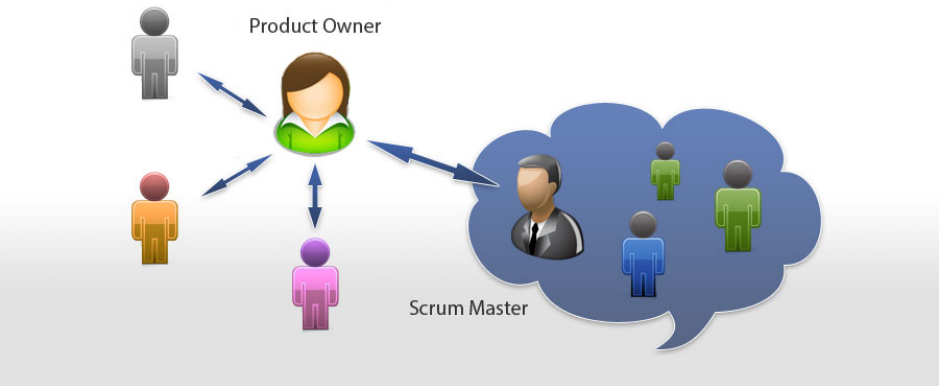

Figura 4.1: Roles en Scrum. Fuente: https://www.scrum.as/

Sprint

El Sprint es la piedra angular en Scrum. Representa un bloque de tiempo de una a cuatro semanas, que debe finalizar con la entrega de un elemento incremental que aporte funcionalidad y que pueda ser evaluado.

Eventos

Del concepto de Sprint se derivan una serie de eventos relacionados con este. En la figura 4.2 se puede observar el esquema que se seguiría para cada Sprint.

- **Sprint Planning Meeting (Reunión de planificación del Sprint):** Esta reunión se lleva a cabo al comienzo de cada Sprint, y en ella se decide como se va a organizar el Sprint, estableciendo que elementos del Product Backlog se van a desarrollar, así como periodos y plazos. La duración máxima es de unas cuatro horas para los Sprint de 2 semanas y ocho para los Sprint de un mes.
- **Daily Scrum (Scrum Diario):** Durante esta reunión diaria de unos quince minutos aproximadamente, se evalúa el trabajo realizado desde el ultimo Daily Scrum, se plantea el objetivo para ese día, y se evalúan los posibles problemas o dificultades que puedan afectar a la consecución del objetivo.
- **Sprint Review (Revisión del Sprint):** Este evento se realiza cada vez que se finaliza un Sprint, y durante esta revisión se evalúa el trabajo realizado y se presentan los avances al cliente o stakeholders.

- **Sprint Retrospective (Retrospectiva de Sprint):** Durante este evento el equipo evalúa aquello que se hizo mal o puede mejorarse, y se ponen en marcha las medidas necesarias para solventar los errores o implementar las mejoras necesarias. Es una revisión que se realiza tras el Sprint Review y antes de que se comience con el siguiente Sprint.

Figura 4.2: Esquema de la estructura un Sprint. Fuente: http://www.i2btech.com/

Artefactos

Los artefactos en Scrum ayudan al equipo a la consecución de los objetivos y a la planificación y estructuración de estos, permitiendo proporcionar la transparencia necesaria para que todos puedan tener una imagen de lo que se necesita, y de la situación del proyecto, permitiendo una adaptación y mejora continua.

- **Product Backlog (Lista de Producto):** El Product Backlog contiene todos los elementos necesarios bien sean características como fallos por resolver, requisitos no funcionales, mejoras o cualquier otro elemento que sea necesario llevar a cabo para lograr llevar a cabo el proyecto. Es visible a todo el equipo, pero solo podrá ser cambiado con el consentimiento del Product Owner. Los items que conforman esta lista se ordenan en función de la prioridad para su resolución, y deben contener además, una estimación del tiempo necesario para llevar dicha tarea o elemento a cabo. Es necesario explicar en que consisten los items que se añaden a esta lista, que pese a que no es obligatorio que sean de un formato concreto, suelen consistir en Historias de Usuario o HU. Estas HU describen una necesidad, funcionalidad o característica, que

el sistema deberá contener para su desarrollo exitoso. Contienen únicamente aquello que debe hacerse, pero no el cómo, y deben ser lo suficientemente claras y cortas como para ser fácilmente entendibles. Un ejemplo de plantilla para estas HU se definió como *Como <role>, quiero <objetivo/deseo>para <beneficio>*. También se ha remarcado la posibilidad de suprimir la parte del *para* cuando no sea necesario o no aporte nada de utilidad.

- **Sprint Backlog (Lista de tareas pendientes del Sprint):** Esta lista contiene aquellas tareas que se han seleccionado del Product Backlog para su desarrollo en el Sprint actual. Estas historias se pueden desglosar en tareas a las que se les asigna un tiempo de realización. Generalmente se suele utilizar un mecanismo que se puede implementar mediante una pizarra o mediante herramientas más complejas como herramientas software, en el que se clasifican las tareas o historias según su situación actual dentro del Sprint como *to do* si aún no se ha empezado a trabajar en ella, *in progress* si se está trabajando y *done* si está completa.

4.1.2 Aplicación al proyecto

Siendo el marco explicado anteriormente el elegido para el desarrollo de este proyecto, es el momento de explicar como se ha planteado este proyecto de acuerdo al marco Scrum. En primer lugar es preciso identificar al equipo Scrum, el cual pese a no ser recomendable el hecho de que el Scrum Master y el Product Owner sean la misma persona, debido al carácter docente de este trabajo era la única solución. Así pues, el equipo queda de la siguiente manera:

- Product Owner: Jesús Serrano Guerrero
- Development Team: Victor Gualdras de la Cruz
- Scrum Master: Jesús Serrano Guerrero

En segundo lugar, se precisa la realización de una primera versión del Product Backlog 4.1 de manera que el grupo de desarrollo pueda comenzar a trabajar en el proyecto. En esta tabla, las historias de usuario se encuentran ordenadas según la prioridad que tienen para el proyecto. Esto quiere decir que se tendrán que desarrollar en el orden establecido, debido a que son las que mayor valor aportan al negocio, y son necesarias para las siguientes tareas.

4.2 Herramientas utilizadas

En esta sección se trataran las principales herramientas utilizadas para el desarrollo de este proyecto, tanto hardware como software, incluyendo las APIs más importantes.

4.2.1 Herramientas Hardware

A continuación se muestran los dispositivos hardware utilizados en el desarrollo de este TFG

- Ordenador portátil personal sobre el que se realizará el desarrollo del proyecto.
- Dos dispositivos Smartphone con Sistema Operativo Android sobre los que se realizarán las pruebas.

4.2.2 Herramientas Software

Las herramientas software más importantes empleadas en este proyecto se muestran a continuación.

Lenguajes de programación

Java

El lenguaje de programación Java[2] es un lenguaje de propósito general y alto nivel, cuyas principales características son las de ser Orientación a Objetos (OO). Además, el código generado por Java tiene la característica de que una vez compilado puede ejecutarse en cualquier plataforma que que soporte Java.

Java será el principal lenguaje en el que se desarrolle la aplicación Android, ya que es el lenguaje sobre el que funcionan las aplicaciones para esta plataforma. Si bien se usará algún otro lenguaje como XML para las interfaces, Java es el más importante.

Python

Python[3] es un lenguaje interpretado de alto nivel, que al igual que java es de propósito general. Ofrece la generación de código generalmente más breve que otro que cumpla el mismo propósito en otros lenguajes, y es comúnmente utilizado como lenguaje de prototipado rápido.

Será utilizado para el desarrollo del servidor de este proyecto debido a su facilidad y potencia de uso en el ámbito de las comunicaciones, y especialmente por su fácil integración con Google Cloud Platform (GCP).

LaTeX

LaTeX [4] es un procesador de textos para la preparación de documentos de gran calidad de tipografía especialmente diseñado para redactar documentos que pretenden ser publicados en algún medio, especialmente largos artículos técnicos o científicos. Permite que el escritor pueda centrarse en lo que escribe y no en el formato y el diseño que debe tener el

[2] https://www.oracle.com/es/java/index.html
[3] https://www.python.org/
[4] https://www.latex-project.org/

texto, encargándose LaTeX de esto. Este será el lenguaje utilizado para la redacción de este documento.

JSON

JSON que son las siglas para JavaScript Object Notation, es un lenguaje de formato para el intercambio de datos[5]. Está basado en un conjunto del lenguaje de programación JavaScript, y se compone principalmente de dos estructuras de alto nivel, una tupla de nombre valor, y una lista de atributos. Dentro de los atributos que se pueden encontrar en el valor de las tuplas encontramos los datos primitivos característicos de todo lenguaje de programación como cadenas, números, y valores booleanos, así como el elemento nulo. Este será el formato que se usará principalmente para la transmisión y comunicación de datos a través de la red.

Android Studio

Como se mencionó en la sección 1, se pretende desarrollar una aplicación para dispositivos Smartphones con SO Android, y por tanto es recomendable disponer de un IDE que nos ayude en esta tarea, y uno de estos entornos es Android Studio[6]. Pese a que existen otros entornos de desarrollo que nos pueden ayudar también para este propósito, como Eclipse utilizando el plugin ADT[7], se ha decidido utilizar Android Studio debido a ser el IDE oficial para el desarrollo de aplicaciones Android y por tanto, proporcionar la seguridad de que siempre estará actualizado y contendrá todo lo necesario para poder desarrollar estas aplicaciones, y además por la gran lista de funcionalidades y facilidades que aporta. Algunas de estas utilidades son su integración con Gradle[8] para la gestión de dependencias, la automatización de la compilación y el despliegue de las aplicaciones, un editor de código inteligente y auto completado, emulador integrado, herramientas que ayudan al depurado y al testing, y muchas otras opciones como la integración con sistemas de control de versiones.

Figura 4.3: Android Studio logo

[5]http://www.json.org/
[6]https://developer.android.com/studio/index.html
[7]https://marketplace.eclipse.org/content/android-development-tools-eclipse
[8]https://gradle.org/

Google Cloud Platform

Google Cloud Platform o GCP es una plataforma de computación en la nube [AFG+10] que ofrece diversos servicios de Infrastructure as a Service, Platform as a Service y Software as a Service. Estos servicios ofrecen características de escalabilidad, control automático y gestión de los recursos, herramientas de monitorización y sobre todo la simulación o virtualización de recursos hardware sin la necesidad de adquirirlos. Esta última característica es una de las principales para que este tipo de tecnología haya sido seleccionada. Esto permite que no sean necesarias tareas de mantenimiento del hardware, debido a que no habrá necesidad alguna de adquirir hardware específico para el servidor, que no haya que preocuparse por el consumo de energía, o de otras tareas de mantenimiento. Tampoco será necesario preocuparse por la infrautilización de recursos, o la necesidad repentina de tener que aumentar la potencia de procesamiento o almacenamiento según la demanda, ya que esta herramienta permite adaptarse con bastante facilidad a estos eventos. Estos servicios se conocen como IAAS.

Además de estos servicios de virtualización de recursos, Google Cloud Platform también proporciona como se ha mencionado anteriormente servicios de PAAS. Entre esos servicios se encuentran herramientas tanto para el desarrollo como el despliegue o el mantenimiento del software que se ejecutará en el servidor, además de la posibilidad de monitorizar tanto el tráfico como los datos que se encuentren almacenados en este. Es decir, pone gran cantidad de recursos al alcance del desarrollador y el equipo en general sin que este necesite tener que implementar e integrar por si mismo todos estos servicios. Ofrece también mecanismos para acceder a los servicios que se encuentren desplegados bajo los servidores.

Además de todo lo mencionado anteriormente, GCP ofrece la posibilidad de utilizar herramientas software ya desarrolladas de manera que no sea necesario invertir tiempo en desarrollar utilidades que ya existen y que se pueden utilizar mediante el alquiler de ese servicio. Estos servicios son conocidos como SAAS, y nos permite utilizar software de terceros para nuestras necesidades.

De la gran cantidad de recursos que ofrece GCP se explicarán los siguientes por ser los más relevantes en el desarrollo de este TFG. A pesar de ello, se han usado otras herramientas y características que proporciona esta herramienta como los registros o el panel de control de APIs.

Google Cloud Datastore

Google Cloud Datastore[9] es una base de datos NoSQL creada para escalar de manera automática, presentar un alto rendimiento y facilitar el desarrollo de aplicaciones. Los datos que se encuentran en el Datastore son accedidos mediante las denominadas *transacciones*, que cumplen las características ACID de atomicidad, consistencia, aislamiento y persistencia.

[9] https://cloud.google.com/datastore/docs/

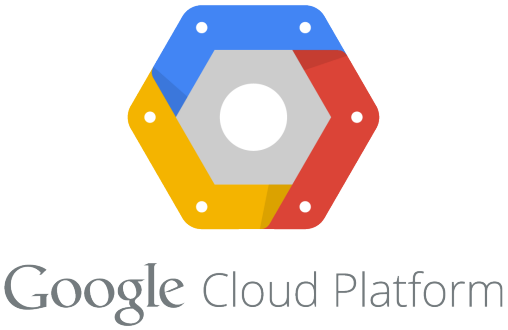

Figura 4.4: Google Cloud Platform logo

Ofrece consistencia fuerte cuando se trata de operaciones que se realizan mediante claves y ancestros, y consistencia eventual en el resto de los casos. Estas transacciones también ofrecen toda las operaciones CRUD de crear, leer, actualizar y eliminar. Google Datastore proporciona numerosos mecanismos para realizar estas transacciones, desde una librería JSON, clientes de código abierto o herramientas mantenidas por los usuarios como NDB o Objectify. Además de las características ACID, ofrece también:

- Disponibilidad: Usa replicación para minimizar el impacto de que uno de los puntos en los que se encuentre falle.
- Escalabilidad: Usa una arquitectura distribuida que maneja de manera automática el escalado.
- Rendimiento: A la vez que es escalable es capaz de mantener un alto rendimiento gracias a su combinación de índices. En una consulta, su rendimiento depende del resultado de la consulta y no del tamaño total del conjunto de datos.
- Almacenamiento y consultas flexibles: Es capaz de trasladar objetos y entidades propias de los lenguajes orientados a objetos y de script a entidades de manera muy natural. También provee de un lenguaje similar a SQL para realizar consultas.
- Encriptación: Se encriptan los datos previo a su almacenado y se desencriptan antes de leídas por los usuarios.
- Administración automática: Google se encarga de todo el mantenimiento.

Como se ha mencionado anteriormente, Cloud Datastore es una base de datos NoSQL y que no se corresponde con el sistema relacional de gestión de base de datos. Existen algunas diferencias ya en los conceptos que estos tratan como se puede apreciar en la tabla 4.2. A diferencia de las filas de una misma tabla en los sistemas relacionales, una entidad de Google Datastore puede tener diferentes campos respecto a otra entidad del mismo tipo, o pueden tener propiedades con el mismo nombre y tipos diferentes.

Identificador	Como	Quiero/Necesito	Estimación	Situación
HU	Usuario	Acceder a una aplicación de mensajería		To Do
HU	Usuario	Poder realizar la entrada en el chat mediante texto y mediante voz		To Do
HU	Usuario	Conocer cual de mis contactos utiliza la aplicación		To Do
HU	Usuario	Comunicarme con otros usuarios		To Do
HU	Usuario	Poder enviar imágenes a otros usuarios		To Do
HU	Usuario	Realizar búsquedas de imágenes filtrando mediante mi entrada		To Do
HU	Usuario	Que se me muestren aquellas imágenes que mas se adapten a mis gustos		To Do
HU	Usuario	Que la búsqueda no quede limitada a mi entrada, sino a su semántica		To Do

Cuadro 4.1: Primera versión del Product Backlog

Concepto	Google Datastore	RDBMS
Categoría del objeto	Kind/Tipo	Table/Tabla
Objeto único	Entity/Entidad	Row/Fila
Identificador único de objeto	Key/Clave	Primary Key/Clave Primaria
Dato individual sobre un objeto	Property/Propiedad	Field/Campo

Cuadro 4.2: Diferencia entre Google Datastore y los RDBMS

Otras diferencias notables con las bases de datos relacionales son su capacidad de escalado automática, capacidad de balanceo y distribución de datos para mejorar el rendimiento, las únicas consultas que permite son aquellas que permiten que su rendimiento solo dependa del tamaño del resultado proporcionado por la consulta, siendo independiente el número de entidades que existan en total. Esta última es una de las razones por las que algunas operaciones no existen en este sistema.

También es importante destacar que soporta multitud de tipos para sus propiedades, pudiendo incluso contener propiedades estructuradas compuestas por entidades de otros tipos.

Google Datastore se utilizará para almacenar los datos relativos a los usuarios que resulten de interés para la aplicación así como información referente a las imágenes para posteriormente poder ser utilizada en el sistema de recomendación.

Google Cloud Blobstore

Blobstore[10] es un servicio proporcionado por Google Cloud Platform y que accedido mediante una API disponible en los lenguajes Python, Java y Go permite almacenar ficheros de un tamaño superior al permitido por el servicio de Google Datastore. Los ficheros se suben en forma de blobs, los cuales se crean indirectamente. En primer lugar es necesario solicitar mediante una llamada a la API la generación de una URL, posteriormente la aplicación cliente podrá proceder a la subida del archivo mediante un formulario web o algún otro tipo de petición HTTP POST. Tras enviar el formulario con el fichero, el servicio Blobstore almacena el archivo y genera una clave para poder recuperar posteriormente el archivo almacenado en el blob. Tras la creación del blob, este puede ser modificado o eliminado, y mantiene un registro con metadatos relativos a su creación, tipo y modificaciones.

Este servicio se usará en la aplicación para almacenar las imágenes que los usuarios envíen unos a otros y para almacenar las imágenes propias de las que se dispondrá antes de que la aplicación comience a ser utilizada por los usuarios.

Cloud Vision API

Google Cloud Vision[11] es una API que permite analizar imágenes. Funciona a partir de modelos de Machine Learning (ML) mediante una API REST. Ofrece diferentes posibilidades en el análisis de dichas imágenes, entre ellas están la clasificación de imágenes entre categorías, aportando también una estimación de la posibilidad de acierto, detecta objetos individuales y caras en las imágenes, pudiendo indicar por ejemplo el número de ciertos objetos o los sentimientos que reflejan dichas caras. Otra de sus funcionalidades es la de detectar texto que se encuentre de una manera u otra en las imágenes.

Para el desarrollo de este TFG interesa especialmente la funcionalidad de detectar la ca-

[10] https://cloud.google.com/appengine/docs/python/blobstore/
[11] https://cloud.google.com/vision/

tegoría con la que se corresponde la imagen para de esta forma poder extraer información de las imágenes seleccionadas por los usuarios y poder utilizar esta información extra en el sistema de recomendación.

Google Cloud Messaging

Google Cloud Messaging (GCM)[12] es un servicio proporcionado por Google que permite enviar datos desde un servidor a diferentes dispositivos y que además permite a estos dispositivos comunicarse entre ellos mediante mensajes. GCM proporciona la funcionalidad de gestionar todos los aspectos relacionados con el encolamiento y la entrega. Además a diferencia de los servicios anteriores, este es completamente gratuito sin importar la densidad del tráfico o la cantidad de usuarios. Además, permite la comunicación entre dispositivos de diferentes plataformas como Android, iOS y Chrome. Tiene la restricción de no permitir envíos superiores a los 4KB en los mensajes, por lo que es necesario recurrir a otros servicios como el Blobstore para enviar las imágenes.

Este servicio se utilizará en este proyecto para gestionar el envío y recepción de mensajes entre los diferentes usuarios. Esta herramienta es muy útil, ya que se encarga de gestionar todo lo relativo con la distribución de los mensajes, como gestionar el hecho de que el dispositivo para el que está destinado el mensaje no este disponible y no sea posible su envío en ese mismo momento. En este caso, este servicio se encargará de seguir intentando comunicarse con el dispositivo destinatario del mensaje de manera independiente al usuario. Además, ofrece también posibilidades de comunicación en grupos si en un futuro decidiese añadir dicha funcionalidad.

Google Custom Search Engine

Google Custom Search[13] permite crear un buscador personalizado para realizar búsquedas sobre una página web o un conjunto de estas. Además, permite grandes opciones de configuración entre las que destacan para la realización de este TFG la de poder filtrar de manera que solo se recuperen imágenes. Esta opción permitirá poder ampliar el abanico de posibles imágenes a utilizar a no solo las imágenes disponibles previo a comenzar a utilizar la aplicación, sino a imágenes de terceros que aumentarán el espectro de opciones disponibles y mejorarán la aplicación. Además proporcionan información y metadatos que resultan de utilidad a la hora de aplicar los sistemas de recomendación.

[12] https://cloud.google.com/vision/
[13] https://developers.google.com/custom-search/

MIT Java Wordnet Interface

MIT Java Wordnet Interface (JWI) [14] es una interfaz Java creada por el Massachusetts Institute of Technology (MIT) para interactuar con Wordnet [15]. Wordnet contiene una base de datos léxica de palabras en inglés, en las que se encuentran nombres, verbos, adjetivos y adverbios incluyendo las distintas definiciones y acepciones de estas palabras, junto con las relaciones léxicas y semánticas que las palabras tienen entre sí, como relaciones de hiperonimia o sinonimia [Fin14].

Se empleará esta API para el procesamiento de la entrada de los usuarios, de forma que esta siga un un procedimiento estándar y sea más sencillo y eficiente (tanto en términos de velocidad de procesamiento como de memoria) procesar la información.

GitHub

GitHub[16] es una herramienta web para la gestión de repositorios basada en Git. Permite almacenar y gestionar código, además de llevar un control de versiones que permiten poder consultar los cambios que han ocurrido durante el desarrollo del proyecto. También facilita la tarea del trabajo en equipo, permitiendo el desarrollo en paralelo y de manera distribuida por varios miembros de un equipo de un mismo trabajo. Proporciona una interfaz web que facilita el uso por parte del equipo de desarrollo del sistema de control de versiones Git, además de añadir algunas otras funcionalidades propias.

Es especialmente útil para este proyecto su funcionalidad de control de versiones, así como para cualquier otro proyecto software que se componga de una complejidad notable. Es especialmente útil en un proyecto que se desarrolla de manera incremental, ya que siempre es útil tener conocimiento de los cambios que se han ido realizando.

TexMaker

Texmaker[17] es una herramienta de edición de texto pensada para trabajar con LaTeX. Ofrece algunas características muy útiles para trabajar con el lenguaje LaTeX como herramientas de corrección ortográfica, autocompletado en el código, y otras herramientas como asistentes para la elaboración de tablas o automatizadores de la compilación.

[14] http://projects.csail.mit.edu/jwi/
[15] https://wordnet.princeton.edu/
[16] https://github.com/
[17] http://www.xm1math.net/texmaker/

Capítulo 5

Arquitectura

En este capítulo se presentará la estructura del proyecto, planteando en primer lugar la estructura global, y analizando posteriormente las fases de desarrollo del proyecto, así como la estructura y componentes de este, y el proceso que se ha seguido para su diseño y desarrollo.

5.1 Visión general

En la figura 5.1 se muestra una representación de estructura global del sistema. Por simplicidad y claridad, se muestra la interacción que habría entre dos usuarios, en la que un usuario *User1* se comunica con el otro usuario *User2*, pese a que esta comunicación es bidireccional y puede ir en cualquiera de los sentidos. Se puede comprobar en esta figura como los distintos módulos se comunican entre sí, y también las principales herramientas externas de las que hacen uso.

A continuación se analizarán los módulos del sistema:

Módulo aplicación

El principal propósito de la aplicación, es el de proporcionar una interfaz que presente las características necesarias para permitir al usuario comunicarse con otros, incluyendo el uso de imágenes. Esta interfaz será la encargada de interactuar con el usuario, presentándole todas las opciones de las que dispone el sistema. Gestionará tanto la configuración inicial del sistema como la posterior comunicación con otros usuarios. Se comunica con otros tres módulos. Utiliza el módulo de gestión de usuarios para gestionar el alta en el sistema del usuario y la sincronización con los contactos. El módulo de comunicaciones provee de los medios necesarios para enviar un mensaje a otro usuario. Finalmente el módulo de recomendación será empleado cuando el usuario quiera buscar una imagen, presentando una colección de posibles candidatas.

Módulo de usuarios

El módulo de usuarios es el encargado de gestionar toda la información relativa a los usuarios. Proporciona los mecanismos necesarios para poder dar de alta y sincronizar usuarios,

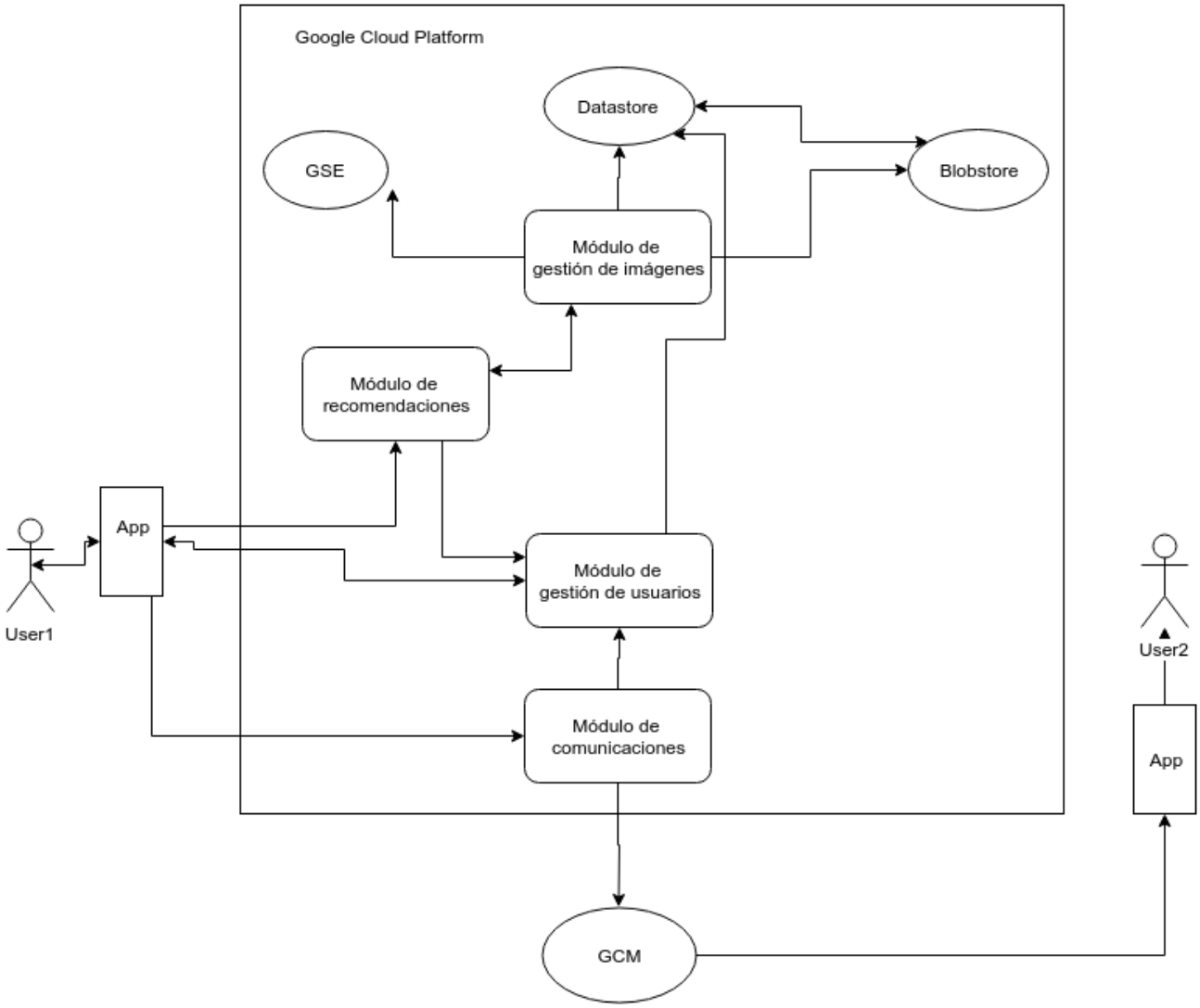

Figura 5.1: Arquitectura del sistema

así como para poder relacionar el número de teléfono de estos usuarios con el identificador necesario para poder hacer llegar el mensaje enviado por el usuario. Una vez obtenido el identificador envía el mensaje utilizando la API GCM. Otra labor importante de este módulo, es la de proporcionar la información relativa a los usuarios que sea necesaria para el modulo de recomendaciones, así como de actualizar esa información cuando existan cambios que así lo indiquen.

Módulo de comunicación

Como su propio nombre indica, este módulo es el encargado de gestionar las comunicaciones entre los usuarios. Cuando el usuario envía un mensaje, este es contendrá entre otras cosas el número de teléfono del destinatario así como toda la información relativa al mensaje. Este módulo se encargará de obtener el identificador necesario que debe proporcionar al sistema de GCM, haciendo uso para ello del módulo de gestión de usuarios.

Módulo de gestión de imágenes

El principal propósito de este módulo es el de proporcionar una interfaz al módulo de recomendaciones, de manera que este tenga acceso las imágenes que se encuentran actualmente en el sistema, así como de proporcionar de una nueva fuente de imágenes. Permite filtrar estas imágenes según los criterios que establezca el sistema de recomendación, además de almacenar y actualizar la información necesaria para que el sistema de recomendación pueda aplicar los algoritmos correspondientes. También se encarga de almacenar en el Blobstore las nuevas imágenes.

Módulo de recomendación

Este es el módulo encargado de recomendar imágenes al usuario a partir de una entrada proporcionada por este. Analizando el perfil del usuario que proporcione el modulo de usuarios, así como la información relativa a las imágenes, este módulo filtrará entre aquellas imágenes que se encuentren actualmente disponibles en el sistema y las imágenes que proporcionen los mecanismos externos. También será el encargado de indicar al módulo de gestión de imágenes de los cambios que debe realizar sobre la información que contiene de estas.

5.2 Sistema de recomendación

El sistema de recomendación de imágenes es el componente más importante de este proyecto, y el que aporta a la aplicación de mensajería desarrollada la diferenciación respecto al resto que existen actualmente en el mercado. También es el componente que mayor complejidad computacional posee del proyecto, y por tanto se ha considerado necesario explicarlo con mayor detenimiento.

5.2.1 Procesamiento de la entrada

Atendiendo al carácter de la entrada es necesario aplicar un preprocesamiento de esta para poder tratar con ella. Esto es debido a que los usuarios pueden introducir la petición o consulta en un formato muy variado, ya que se trata de lenguaje natural.

Limpieza de la entrada

En primer lugar es necesario realizar una limpieza de la misma. En esta limpieza, entre otras cosas, se eliminarán aquellas palabras o elementos que no aporten nada, como las preposiciones, pronombres, artículos, etc., dejando únicamente los nombres, verbos, adjetivos y adverbios, que para nuestro caso son los elementos que aportan utilidad, mientras que los otros aportan ruido.

Por otra parte, también se hace necesario tratar aquellas palabras que si han pasado el proceso de selección (nombres, verbos, adjetivos y adverbios), y preprocesarlas de manera

que se eliminen las diferencias entre los plurales y los singulares (pasando todo a singular) o pasando los verbos a su forma en infinitivo.

Puede darse la situación en que al *limpiar* aquellas palabras de elementos como prefijos o sufijos aparezcan varias opciones alternativas sobre las que esa palabra se puede reducir. En este caso se seguirá el criterio que establece Wordnet y se escogerá la primera opción por considerarse la más probable.

La salida de este proceso será la consulta que se utilice cuando se recurra a proveedores externos para buscar imágenes.

Expansión de la entrada

En segundo lugar encontramos el proceso de expansión de esta entrada. Debido a que diversos usuarios pueden hacer referencia al mismo elemento de diferentes maneras, por ejemplo, algunos pueden utilizar la palabra *colegio* mientras que otros pueden utilizar *escuela* refiriéndose a lo mismo, y la misma imagen puede ser perfecta para ambos usuarios.

Por este motivo, se hace necesario procesar dicha entrada, buscando para todas aquellas palabras que han pasado el filtro inicial sinónimos y palabras relacionadas. Encontramos aquí un problema notable debido al hecho de que la mayoría de las palabras suelen tener varias acepciones, tanto en inglés como en español, y además una misma palabra puede ser a la vez un nombre, un verbo, un adjetivo y/o un adverbio. Esto implica que una misma palabra puede tener diferentes sinónimos o palabras relacionadas en función de aquella de las acepciones que tomemos.

Generalmente para nuestro caso no afectará aquella ambiguación referente a la diferencia de verbo y nombre, ya que en inglés, aquellos nombres que se derivan de un verbo representan ese verbo, como por ejemplo *run* en inglés, puede significar tanto correr como carrera, pero la imagen necesaria para representar el concepto representará normalmente tanto al nombre como al verbo. Respecto a esto, y debido a la naturaleza propia de las imágenes se escogerá siempre en primer lugar la acepción referente a los nombres, ya que será aquella con la que más se trabaje. En caso de no existir una definición referente a esa palabra como nombre, lo cual implicaría que esa palabra nunca puede ser un nombre, se recurriría en segundo lugar a las acepciones de verbo, en caso de que tampoco existan acepciones de esa palabra que sean verbos se recurriría a adjetivos y finalmente si esto tampoco funciona a adverbios.

Respecto a la existencia de varias acepciones de la misma forma gramatical, es dónde surgen mayores problemas. Volviendo al caso de antes, *run* en inglés, puede significar una simple carrera realizada a pie, o, en el contexto del fútbol americano, una jugada en la que un jugador intenta pasar el balón a través del equipo contrario, además de otras muchas acepcio-

nes como sustantivo. En numerosas ocasiones, esta diferencia de definición hace referencia a palabras que han adquirido el mismo nombre por ciertas relaciones semánticas o históricas, y aunque distintas es su definición exacta, están muy relacionadas, por lo que la selección de una acepción no es tan crucial. Sin embargo, en muchos otros casos, las diferentes acepciones no se limitan a este tipo de relaciones, y una misma palabra puede referirse a dos conceptos muy distintos. Por ejemplo, la palabra *mean* como adjetivo puede significar tanto ser mezquino, como la media estadística.

En el caso de este proyecto y atendiendo a sus limitaciones, se ha decidido recurrir a la opción de escoger la primera acepción que aparezca en cada caso. Como se verá más adelante, aquí es donde nos encontramos con una de las posibilidades más interesantes de mejora del sistema, tanto para reconocer la forma gramatical de la que se trata, como para reconocer a partir del contexto su significado semántico, pudiendo aplicar para ello procedimientos más sofisticados.

Respecto a la aplicación de estos sinónimos, se utilizarán únicamente con aquellos recursos que se encuentren en el sistema. Esto quiere decir que no se utilizarán estas palabras extras en las búsquedas de proveedores externos. El motivo de este mecanismo de actuación, es debido a que sería tanto computacionalmente como en términos monetarios muy costoso aplicar estos mecanismos a la búsqueda por ejemplo en GSE, ya que para cada palabra sería necesario realizar una nueva búsqueda, con todo lo que ello conlleva. Además, en este caso, Google ya realiza un procesamiento propio en las búsquedas. Si bien, cuando un usuario seleccione una imagen que proceda de un medio ajeno, y esta se incluya en el sistema, en este caso si se incluirán estas nuevas palabras como palabras clave de la imagen. A continuación, se mostrarán los diferentes perfiles relevantes para el sistema de recomendación, así como las técnicas aplicadas, y se podrá observar en más detalle como se tratan todas estas nuevas palabras a las que se llamará palabras clave.

5.2.2 Técnicas empleadas

En esta sección se presentarán las técnicas empleadas que se vieron en la sección 3.1.3, así como los mecanismos de hibridación utilizados para combinar estas técnicas.

En primer lugar es necesario indicar que el contenido de las valoraciones en este caso será principalmente binario, considerándose la selección de una imagen como valoración positiva y la no selección como negativa. Sin embargo, en este proyecto se usará únicamente la información referente a las valoraciones positivas, es decir, se tendrá en cuenta para los diferentes algoritmos únicamente la información correspondiente a las imágenes utilizadas por los usuarios.

Respecto a las técnicas de recomendación empleadas son las siguientes:

Basada en contenido

Por un lado tenemos la técnica de recomendación basada en contenido. Como se ha mencionado anteriormente, esta tipo de técnica busca relaciones entre elementos que ya han sido positivamente valorados por el usuario y los nuevos elementos que están en el proceso de selección para la recomendación, y escoge aquellos que según cierto criterio más se asemejen a los ya valorados.

En este caso, esta técnica valorará la información sobre las imágenes que ya han sido seleccionadas por el usuario, comprobando si existe relación entre estas, y creando un perfil para este usuario. Así, el perfil de una imagen usada (*ImagenUsada* por un usuario estará compuesto del identificador de la imagen, junto con un contador del número de veces que ha utilizado esa imagen.

Por otra parte la información relativa a la *Imagen*, la cual podemos identificar como el perfil de la imagen, será la que se muestra a continuación:

- identificador_blob: Es el identificador que permite recuperar la imagen que esta almacenada en el Blobstore.
- etiquetas: Las etiquetas que proporciona Google Cloud Vision sobre la imagen.
- palabras_clave: Las palabras utilizadas en las búsquedas por los usuarios que hayan seleccionado esa imagen. Esto incluye también aquellas palabras relacionadas que se han extraído utilizando la herramienta Wordnet.
- enlace: El enlace completo donde se encontraba el recurso originalmente.
- enlace_sitio: El enlace del sitio web de cual se tomó la imagen.

Se usará la información referente a los sitios web para conocer las preferencias y gustos de los usuarios por ciertos sitios web en concreto. Esto se usará en las búsquedas de nuevas imágenes, especificando búsquedas en aquellos sitios web que contienen o contenían imágenes previamente seleccionadas por este. Estos perfiles o entidades ayudarán a comprender la estructura del sistema, así como a entender el funcionamiento del algoritmo que se describirá más adelante.

Colaborativa

También se va a aplicar la técnica colaborativa. Esta técnica establece relaciones entre los usuarios basándose en los elementos que han valorado previamente. Posteriormente, cuando un usuario solicita una recomendación o el sistema le ofrece una, se analiza la relación existente entre ese usuario y otros que hayan valorado ítems en los que el usuario pueda estar interesado, y le recomienda uno o varios de estos ítems.

Respecto a la adaptación de esta técnica en este proyecto, la relación entre los usuarios

se medirá atendiendo a las imágenes que estos han utilizado (en este caso es lo mismo que valorar positivamente).

Se valorará una única vez el hecho de que dos usuarios utilicen una misma imagen, es decir, si ambos usuarios valoran positivamente la misma imagen varias veces, a efectos de la relación existente entre estos usuarios para esta técnica, será igual que si ambos han valorando positivamente la imagen una única vez. Esto se hace de esta manera para que una sola imagen introduzca tanto ruido que ambos usuarios parezcan similares al sistema pese a que solo coincidan en esto. Así pues, el grado de relación o de semejanza entre dos usuarios se medirá en función del número de imágenes en común que estos hayan utilizado.

Para cada relación de usuarios (*UsuariosRelacionados*) se creará una entidad que contenga un identificador de cada usuario junto con el grado de relación existente entre ellos. El perfil de un *Usuario* está determinado por lo siguiente:

- numero_telefono: Número de teléfono del usuario.
- identificador_comunicaciones: Identificador de registro, necesario para el sistema de comunicaciones
- palabras_clave: Utilizadas por los usuarios en sus búsquedas. Esto incluye también aquellas palabras relacionadas que se han extraído utilizando la herramienta Wordnet.
- imagenes_utilizadas: identificador de la imagen junto con un contador).

Además de estas relaciones, existen otras que son auxiliares a estas y que utilizarán en el pseudocódigo. Están las *PalabrasClave*, que contienen una palabra clave junto con el número de veces que ha sido empleado. Y por último la *Etiqueta*, que contiene una etiqueta proporcionada por Google Cloud Vision junto con una probabilidad que este aporta de que el término se corresponda con la imagen.

En el listado 5.1 se muestra el pseudocódigo referente a los procesos que se ejecutan cada vez que una imagen es seleccionada, y que modifican la información referente a esa imagen y al usuario, así como a las relaciones entre estos. Este pseudocódigo muestra una versión simplificada, aislando de los detalles de la implementación y la tecnología usada, y mostrando únicamente el proceso de manera más abstracta.

Cada vez que se seleccione una imagen, se comprueba si esta ya se encuentra en el sistema. En caso de no encontrarse aún, crea un blob para almacenarla, y la sube al sistema. Además añade la información que será estática en el sistema, como son el enlace, enlace del sitio y las etiquetas de Google Cloud Vision, que aportarán información similar a la proporcionada por las palabras clave que hayan utilizado los usuarios. Después, y al igual que haría si la imagen existiese en el repositorio propio, actualiza la información referente a las palabras clave que han sido empleadas en este caso. Esta actualización se hace tanto para la imagen como para el propio usuario, ya que cada perfil lleva un registro de las palabras utilizas (por

uno en el caso del usuario y empleadas en el otro en el caso de la imagen), junto con el número de veces que se han empleado. En el pseudocódigo mostrado, solo existe un método de actualización que será válido para los dos tipos de entidades. De esta manera, si no se han utilizado nunca se añaden con valor uno, y si ya han sido utilizadas se incrementa el contador.

Además de actualizar las etiquetas es necesario realizar otras modificaciones respecto de las relaciones existentes entre los usuarios, y entre los usuarios y las imágenes que han utilizado. Cada usuario tiene un registro de las imágenes que este ha seleccionado junto con el número de veces que lo ha hecho, y por tanto se actualiza cada vez que se añade una imagen en caso de que ya la haya utilizado alguna vez, o por el contrario se crea una nueva referencia inicializando el contador.

Respecto a la relación entre usuarios, como se ha mencionado anteriormente, por cada imagen que dos usuarios tienen en común, su relación gana fuerza.

```
def imagen_seleccionada(imagen_id, usuario, palabras_clave):
    imagen = imagen_id.get()
    if(not(imagen)):
        crea_blobImagen()
        sube_imagen()
        actualiza_informacionImagen()

    imagen.actualiza_palabras_clave_imagen(palabras_clave)
    usuario.actualiza_palabras_clave_usuario(palabras_clave)
    usuario.actualiza_imagenes_utilizadas(imagen_id)


def actualiza_palabras_clave(entidad, palabras_clave):
    for palabra in palabras_clave:
        if(palabra in entidad.palabras_clave):
            entidad.palabras_clave.get(palabra).contador = entidad.palabras_clave.get(
                palabra).contador + 1
        else:
            entidad.palabras_clave.append(PalabraClave(palabra_clave = palabra, contador =
                1))

def actualizar_imagenes_utilizadas(usuario, imagen_id):
    if(ImagenesUtilizadas.query(usuario = usuario, imagen = imagen_id) == None):
        ImagenesUtilizadas(usuario = usuario, imagen = imagen_id, contador = 1)
        actualizar_usuarios_relacionados(user, imagen_id)
    else:
        ImagenesUtilizadas.query(usuario = usuario, imagen = imagen_id)[0].contador =
            ImagenesUtilizadas.query(usuario = usuario, imagen = imagen_id)[0].contador + 1


def updateRelatedUsers(usuario1, imgagen_id)
    imagenes_usadas = ImagenesUsadas.query(ImagenesUsadas.img == imagen_id)
    for imagen_usada in imagenes_usadas:
        usuario2 = imagen_usada.user.get()
        if(usuario1 != usuario2):
            usuarios_relacionados = UsuariosRelacionados.query(UsuariosRelacionados.
                usuario1.IN([usuario1, usuario2]), UsuariosRelacionados.usuario2.IN([
                usuario1, usuario2]))
            if(usuarios_relacionados != None):
                rUser.relation = rUser.relation
```

Listado 5.1: Actualización de la información tras la selección de una imagen

Actualización de la información tras la selección de una imagen

Hibridación

Por último, se hace necesario aplicar alguno de los mecanismos de hibridación vistos en la sección 3.1.5 para combinar las técnicas que se han decidido utilizar. En este caso se ha decidido emplear un sistema conmutado. En este tipo de sistemas, se combinan en la salida recomendaciones de todas las técnicas que se emplean, pudiendo aportar cada una de las técnicas un mayor o un menor número de elementos en función del mecanismo que se

decida implantar.

En este caso, se ha decidido utilizar este mecanismo debido a que las recomendaciones proporcionadas por todas las técnicas son útiles en todo momento, ya sea en mayor o menor medida. Siempre será útil disponer de nuevas imágenes de manera que el sistema no sufra de sobreespecialización y se quede estancado con las imágenes que tenga, y será necesario buscar estas para lo que la técnica basada en contenido será de gran utilidad. Posteriormente, cuando la cantidad de imágenes de las que se disponga en el repositorio propio empiece a aumentar también irá siendo cada vez más útil aquellas imágenes proporcionadas por la técnica colaborativa. De esta manera, permite adaptar en cada momento que técnica tiene mayor relevancia, y dar mayor peso a esta, sin dejar de aprovechar las ventajas que aportan las demás.

Cuando el sistema comienza a funcionar únicamente se dispone de las primeras imágenes que se encuentran en el sistema que son los pictogramas, y de aquellas proporcionadas por los proveedores externos. De esta manera, al comienzo de su puesta en marcha, el sistema se limitará a ser un sistema de filtrado de imágenes, sin aportar ningún rasgo de personalización a los usuarios.

Conforme el usuario empiece a utilizar en mayor medida el sistema, este será capaz de aprender de los gustos de los usuarios y comenzarán a cobrar mayor importancia, lo cual resultará en mayor número de imágenes aportadas por este tipo de técnica en la salida final que se le proporcionará al usuario. Al igual que en este caso, cuando el número de usuarios se expanda, y aumente la relación entre estos, será la técnica colaborativa la que pase a aportar mayor peso en las recomendaciones.

5.2.3 Algoritmo

Respecto al algoritmo de recomendación, en el listado 5.2 se muestra una versión más simple del código en la que se aislan de los detalles propios de la tecnología utilizada.

En primer lugar, aparecen los valores de unas constantes que nos permitirán configurar el número máximo de imágenes que se recomendarán al usuario en total, así como el máximo número de imágenes que proporcionará cada uno de las técnicas de recomendación. Más adelante, se verá como existen mecanismos para controlar que si una técnica no es lo suficientemente confiable con los datos que tiene, aportará menos imágenes de este máximo. Los valores de estas constantes se pueden adaptar en caso de considerarse necesario dar más importancia a alguna de estas técnicas.

En el caso de la técnica colaborativa por ejemplo, es necesario establecer un valor mínimo para la relación entre los usuarios, ya que si no, puede darse el caso de que ambos hayan valorado tan solo unas pocas imágenes en común por casualidad pero sus gustos difieran enormemente. Lo mismo ocurre con la técnica basada en contenido, sí el usuario ha seleccionado imágenes de numerosos portales y no tiene una clara predilección por ninguno, esta

técnica puede no aportar valor útil.

Usando la aplicación de la técnica colaborativa se procederá en primer lugar a obtener aquellas imágenes que contienen las palabras clave que el usuario ha introducido. Posteriormente se obtendrán todos aquellos usuarios que hayan seleccionado en algún momento alguna de las imágenes que se encuentran el la selección anterior, es decir, que hayan seleccionado imágenes que contenga alguna de las palabras clave que se encuentren en la búsqueda. A continuación, se obtendrá el conjunto de los usuarios que estando relacionados (mediante el procedimiento explicado anteriormente), también se encuentren en el conjunto de usuarios que hayan valorado imágenes que contengan las palabras clave de la consulta. Este conjunto que contiene los usuarios relacionados, estará ordenado por la fortaleza entre la unión de ambos.

Así, para cada usuario de este conjunto, y hasta que no se alcance el número de imágenes mínimo, o la relación de usuarios deje de ser lo suficientemente relevante como para considerar que ambos usuarios no están lo suficientemente relacionados, se seleccionará de cada usuario aquella imagen que conteniendo alguna de las palabras clave, haya sido empleada mayor número de veces por el usuario en cuestión. Además el sistema se asegurará de que no se introducen imágenes repetidas en el conjunto de salida.

```
1  NUMBER_OF_IMAGES = 10
2  LIMIT_OF_RUSERS = NUMBER_OF_IMAGES * 0.5
3  IMAGES_CONTENT = NUMBER_OF_IMAGES * 0.3

5  RELACION_MINIMA = 20

7  def obtener_imagenes_recomendadas(usuario_id, palabras_clave):
8      sitios_web_contenido = obtener_imagenes_contenido(usuario_id)
9      imagenes_colaborativo = obtener_imagenes_colaborativo(usuario_id, palabras_clave)

11     return json({'sitios': sitios_web_contendio, 'imagenes': imagenes})

13 def obtener_imagenes_colaborativo(usuario_id, palabras_clave):
14     imagenes_colaborativo = []
15     imagenes = Imagenes.query(Imagenes.palabras_clave.palabra_clave.IN(palabras_clave) or
           Imagenes.etiquetas.etiqueta.IN(palabras_clave))
16     usuarios = Usuarios.query(Usuario.images_used.img.IN(images))

18     usuarios_relacionados = UsuariosRelacionados.query(UsuariosRelacionados.usuario1 =
           usuario or UsuarioRelacionado.usuario2 = usuario)).filtrar(UsuariosRelacionados.
           usuario1.IN(usuarios) or UsuariosRelacionados.usuario2.IN(usuarios)).ordenar(
           UsuariosRelacionados.relacion)


21     for ur in usuarios_relacionados:
22         if(ur.relacion > RELACION_MINIMA):
23             buscando_imagen = True
24             usuario = ur.get_usuario_relacionado(usuario_id).get()
25             imagenes_utilizadas = ImagenesUilizadas.query(ImagenesUilizadas.usuario =
```

```
                usuario, ImagenesUilizadas.imagen.IN(imagenes).ordenar(ImagenesUtilizadas.
                contador)
            while(buscando_imagen):
                img_candidata = imagenes_utilizadas.pop(0)
                if(not (img_candidata in imagenes_colaborativo)):
                    imagenes_colaborativo.append(img_candidata)
                    buscando_imagen = False
            if(len(imagenes_colaborativo) >= LIMIT_OF_RUSERS):
                break
        else:
            break
    return imagenes_colaborativo


def obtener_imagenes_contenido(usuario_id):
    sitios_preferidos = []
    sitios = {}
    imagenes_utilizadas = ImagenesUtilizadas.query(user = usuario_id)
    for imagen_utilizada in imagenes_utilizadas:
        imagen = imagen_utilizada.imagen.get()
        sitios[imagen.enlace_sitio] = sitios[imagen.enlace_sitio] + 1

    mediana = calcularMediana(sitios)
    sitios_ordenados = sorted(sitios, key=sitios.__getitem__, reverse = True)

    for x in range(IMAGES_CONTENT):
        if sitios[sitios_ordenados[x]] > 2 * mediana:
            sitios_preferidos.append(sitios_ordenados[x])

    return sitios_preferidos
```

Listado 5.2: Algoritmo de recomendación

El algoritmo que aplica la técnica basada en contenido, en primer lugar obtendrá las imágenes utilizadas por el usuario. Para cada una de estas imágenes comprobará el sitio web en el que se encontraban, elaborando una lista con todos los sitios así como del número de veces que el usuario ha seleccionado una imagen de ese sitio. Tras ordenar esta lista, seleccionará los sitios que han sido más recurridos por el usuario y dentro de los límites que se hayan delimitado para esta técnica, y comprobará si es lo suficientemente relevante, para lo cual comprobará si el número de imágenes que se han extraído de ese portal es superior al doble de la mediana de las veces que ha visitado cada portal.

La salida, como se puede comprobar, será un archivo json que será enviado al usuario, para que este pueda tanto realizar la búsqueda de las imágenes en esos portales, como la búsqueda de las imágenes genérica para completar el total de imágenes necesarias para recomendar. Esta búsqueda será realizada y gestionada directamente por el dispositivo del usuario, de manera que se libere de carga al servidor, y disminuya a la vez la carga en la red, lo cual permitirá ahorrar tanto recursos como por consiguiente dinero en caso de que por motivos de la demanda de usuarios y uso sea necesario pasar a hacer uso de algunas versiones de pago de las tecnologías usadas. El orden en el que se le presentarán las imágenes al usuario será tal

cual se descarguen en el dispositivo y muestren, es decir, no existirá un orden específico.

5.3 Fases de desarrollo

A lo largo de esta sección, se mostrarán las diferentes iteraciones o sprints que se han llevado a cabo para desarrollar este proyecto. Estos sprints se derivan de la planificación inicial establecida en el Capítulo 4, en concreto se puede observar dicha planificación en el cuadro 4.1. Para cada sprint se mostrarán las Historias de Usuario que se han seleccionado (Sprint Backlog), mostrando estas en detalle, y desglosándolas en tareas. Se analizarán estas tareas mostrando cuando sea preciso el código desarrollado para implementar esta tarea, así como los diagramas que se considere oportuno para aclarar las diferentes tareas.

5.3.1 Sprint 1

En el sprint planning meeting se ha decidido que para este sprint se realizarán la primera y segunda Historia de Usuario (HU). En los cuadros 5.1 y 5.2 se muestra el Sprint Backlog, con las Historias de Usuario seleccionadas para esta interacción, indicando entre otras cosas los mecanismos de validación, así como las tareas en las que se desglosa, y el tiempo estimado para su realización.

Historia de usuario	
Número: 1	**Rol:** Usuario
Esfuerzo:	**Iteración:** 1
Nombre: Aplicación Android	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Como usuario quiero poder acceder a una aplicación Android que presente las características y funcionalidades básicas de una aplicación de mensajería	
Validación: Los usuarios pueden instalar la aplicación en su dispositivo y navegar por la aplicación, realizando acciones como escribir a contactos que serán falsos y se insertarán de manera manual.	
Tareas: Tarea: Interfaz de instalación. Tarea: Interfaz genérica de la aplicación. Tarea: Diseñar e implementar base de datos local. Tarea: Simular usuarios para poder probar la interfaz.	

Cuadro 5.1: Historia de Usuario 1

Tarea: Interfaz de instalación

Será necesario desarrollar una interfaz que se ejecutará tras la instalación de la aplicación en el dispositivo. Consistirá en una ventana en la que aparecerá un cuadro de texto donde escribir el número de teléfono, junto con un desplegable donde aparecen los prefijos internacionales. La aplicación detectará automáticamente la nacionalidad de la tarjeta SIM, fijando

directamente el prefijo. Aún así, el usuario podrá seleccionar otro si así lo considera.

Esta pantalla inicial permitirá en un futuro realizar acciones antes de comenzar a ejecutar la aplicación, como registrar al usuario en algún servicio si así fuese necesario.

Tarea: Interfaz genérica de la aplicación

Esta será la interfaz que el usuario utilizará tras la configuración inicial. Consistirá de una lista donde este podrá ver aquellos usuarios de la misma que el tiene como contactos. En caso de que el usuario tenga asignada una imagen para ese contacto en el teléfono esta será la utilizada por la aplicación. En caso de no tener ninguna imagen asignada se utilizará una por defecto. El nombre del contacto también sera aquel que aparezca en la información del contacto almacenada por el usuario.

Todas aquellas cadenas de texto que aparezcan a lo largo de la interfaz estarán configuradas para que se adapten al idioma por defecto del teléfono. Aunque como ya se comentó al principio, la aplicación está orientada al inglés, se realiza de esta forma en caso de que en un futuro se decida ampliar el alcance de la misma.

Tarea: Diseñar e implementar base de datos local

Será necesario la utilización de una base de datos local que nos permita almacenar la información relativa a los contactos que el usuario tiene. Se utilizará la clase SQLiteDatabase proporcionada por Android para almacenar los datos. Para el acceso a la base de datos se usará la clase ContentProvider, la cual proveerá de una interfaz para acceder a estos datos, y realizar las operaciones pertinentes. El código referente a la Base de Datos (BBDD) se encuentra en la clase *DavaProvider.java*, en el paquete de persistencia del código referente a la aplicación. A continuación se muestran los campos que componen la tabla referente a los perfiles contactos, la cual se llama profile.

- **_id:** será un entero generado automáticamente de manera incremental, y que será la clave primaria.
- **Name:** Un campo de tipo texto que contendrá el nombre de cada usuario y que será el que se muestre.
- **phoneNumber:** Campo de tipo texto y único. Contendrá el teléfono del usuario.
- **photo:** Campo de tipo texto. Contendrá la ruta relativa al lugar donde se encuentra almacenado el archivo deseado.

Además, la clase ContentProvider proveerá de una funcionalidad muy importante, y es la de que permitirá que cada vez que se modifique la base de datos, automáticamente se actualice la interfaz que presenta los datos relativos a los contactos.

Para este refresco de la interfaz también será necesario hacer uso de clase CursorLoader, que será el encargado de realizar las consultas en segundo plano (evitando bloquear la in-

terfaz) sobre el ContentProvider, y que estará integrado con la interfaz. Este CursorLoader establecerá aquella información que será necesario consultar y retornar.Esta clase a su vez devolverá un cursor, que a su vez proporcionará a las vistas con la información necesaria para actualizarse. En el listado 5.3 se puede observar como funciona este proceso. Debido a la gran cantidad de código que contenía la clase referente a este listado, se ha eliminado aquello que no se ha considerado relevante, pero como se ha mencionado en la introducción puede consultarse en el CD.

```
1  public class ContactTab extends ListFragment implements LoaderManager.LoaderCallbacks<
       Cursor> {
2      private SimpleCursorAdapter adapter;
4      @Override
5      public void onCreate(Bundle savedInstanceState) {
6          super.onCreate(savedInstanceState);
8          adapter = new SimpleCursorAdapter(getActivity(),
9                  R.layout.contact_item,
10                 null,
11                 new String[]{DataProvider.COL_NAME, DataProvider.COL_PHOTO},
12                 new int[]{R.id.contact_name, R.id.contact_profile_photo},
13                 0);
14         adapter.setViewBinder(new SimpleCursorAdapter.ViewBinder() {
16             @Override
17             public boolean setViewValue(View view, Cursor cursor, int columnIndex) {
18                 switch(view.getId()) {
19                     case R.id.contact_profile_photo:
20                         Bitmap bitmap = null;
21                         try {
22                             bitmap = MediaStore.Images.Media.getBitmap(getActivity().
                                   getContentResolver(), Uri.parse(cursor.getString(
                                   columnIndex)));
23                         } catch (IOException e) {
24                             e.printStackTrace();
25                         }
26                         CircleImageView photo = (CircleImageView) view;
27                         photo.setImageBitmap(bitmap);
28                         return true;
29                 }
30                 return false;
31             }
32         });
33         setListAdapter(adapter);
34         getLoaderManager().initLoader(0, null, this);
35     }
37     @Override
38     public Loader<Cursor> onCreateLoader(int id, Bundle args) {
39         return new CursorLoader(getActivity(),
40                 DataProvider.CONTENT_URI_PROFILE,
41                 new String[]{DataProvider.COL_ID, DataProvider.COL_NAME, DataProvider.
                       COL_PHOTO},
42                 null,
```

```
43                  null,
44                  DataProvider.COL_ID + " DESC");
45      }

47      @Override
48      public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
49          adapter.swapCursor(data);
50      }

52      @Override
53      public void onLoaderReset(Loader<Cursor> loader) {
54          adapter.swapCursor(null);
55      }
56  }
```

Listado 5.3: Refresco de la interfaz tras modificación BBDD

Tarea: Simular usuarios para poder probar la interfaz

Se añadirá la información de los usuarios simulada en la base de datos para comprobar que todo funciona correctamente. Servirá tanto para comprobar que los usuarios se muestran correctamente como para comprobar que las conversaciones también se comportan conforme a lo establecido.

Tarea: Realizar la entrada mediante texto y mediante voz

En la conversación entre usuarios, estos podrán seleccionar tanto entrada mediante texto como por voz, y podrán intercambiar los métodos durante las mismas en el momento que deseen.

Tarea: Almacenar mensajes y visualizarlos

Será necesario crear una nueva tabla en la base de datos para los mensajes junto con la tabla de los usuarios. Se seguirá el mismo mecanismo que en el caso de los contactos, actualizando la interfaz de mensajes de manera muy similar a la empleada con los contactos por lo que no se volverá a mostrar el código. La información sobre los campos de esta tabla es la siguiente:

- **_id:** será un entero generado automáticamente de manera incremental, y que será la clave primaria.
- **msg:** Campo de tipo texto con el contenido del mensaje. En caso de tratarse de un archivo multimedia contendrá la ruta en la que se encuentra dicho archivo.
- **type:** Tipo texto que indica el tipo del mensaje, texto o imagen en este caso.
- **phoneNumberFrom:** Texto que contiene el número de teléfono que ha enviado el mensaje en caso de ser un mensaje recibido o el propio en caso de ser enviado.

- **phoneNumberTo:** Es un campo en formato de texto con el número de teléfono del destinatario del mensaje en caso de ser un mensaje enviado o nulo en caso de ser recibido.

- **at:** Campo en formato datatime que contiene el momento en el que se recibió o envió el mensaje. Se establece de manera automática al almacenar un mensaje en la BBDD.

Sprint Review

Durante el Sprint Review se ha evaluado junto con el Product Owner el primer prototipo de la aplicación, quedando este satisfecho con el mismo.

Sprint Retrospective

Una de las cosas que se decide durante esta primera revisión, es empezar a realizar las pruebas de la aplicación en un dispositivo Android real, y terminar con las pruebas en el emulador debido a la lentitud que supone este.Por lo demás se comprueba que todo va correctamente.

5.3.2 Sprint 2

Durante el Sprint Planning Meeting del segundo Sprint, se comprueba que por el momento se va bastante bien respecto a la planificación inicial, por lo que no es necesario realizar ningún cambio. Se ha decidido incluir en el Sprint Backlog las HU 3 y 4. Además, durante esta reunión, se presenta la posibilidad de utilizar el servicio Google Cloud Messaging para la parte del envío de mensajes. La idea inicial, era desarrollar el sistema de mensajes completo, haciendo que los usuarios se comunicasen directamente mediante el servidor, sin ningún otro intermediario. Sin embargo, la idea de utilizar este servicio que proporciona Google de manera absolutamente gratuita, y que aporta mayor funcionalidad de la que estaba prevista incluir en un principio se muestra como una muy buena oportunidad. Entre otras cosas, ofrece un sistema muy confiable, y además permite comunicar con otros dispositivos, entre los que se encuentra iOS, permitiendo en un futuro que se pueda integrar esta plataforma con mayor facilidad.

En los cuadros 5.3 y 5.4 se muestra la información relativa a las las HU.

Tarea: Desarrollo de una primera versión del servidor

Se precisa el desarrollo de un servidor que se encargue de gestionar a los usuarios. Se diseñará un servidor basado en la arquitectura REST, utilizando como formatos para el envío de datos JSON.

Historia de usuario	
Número: 2	**Rol:** Usuario
Esfuerzo:	**Iteración**: 1
Nombre: Entrada por texto y por voz	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Como usuario quiero poder realizar la entrada metiante texto y voz en una conversación, así como que mis mensajes se guarden al enviar.	
Validación: El usuario podrá seleccionar un contacto con el que simular una conversación. Podrá seleccionar la vía escrita o mediante voz, y al pulsar en enviar verá como el mensaje aparece en la pantalla. Si sale y vuelve a entrar el mensaje seguirá ahí.	
Tareas: Tarea: Realizar la entrada mediante texto y mediante voz. Tarea: Almacenar mensajes y visualizarlos como si se hubiesen enviado.	

Cuadro 5.2: Historia de Usuario 2

Historia de usuario	
Número:3	**Rol: Usuario**
Esfuerzo:	**Iteración: 2**
Nombre: Conocer que contactos utilizan la aplicación	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Es necesario registrarse en el sistema y sincronizar los usuarios que disponen de la aplicación y que están dentro de los contactos del usuario.	
Validación: El usuario sincroniza los contactos de su dispositivo con el sistema y obtiene todos aquellos que poseen la aplicación, los cuales se muestran dentro de la lista de contactos.	
Tareas: Tarea 1: Desarrollo de una primera versión del servidor que utilizará arquitectura REST. Tarea: Integración del servidor con Google Cloud Platform. Tarea: Estandarizar el formato de los números de teléfono. Tarea: Sincronizar los contactos del usuario que sean usuarios de la aplicación.	

Cuadro 5.3: Historia de Usuario 3

Tarea: Integración con Google Cloud Platform

El servidor diseñado anteriormente se desplegará utilizando los servicios proporcionados por Google Cloud Platform, en concreto App Engine, en lugar de usar un servidor tradicional. Para ello, es necesario adaptar el servidor a esta tecnología. Entre otras cosas se deben incluir algunos archivos de configuración indicando las dependencias de librerías externas que utilizará la aplicación, y que es necesario desplegar junto con el servidor en el caso de que App Engine no las tenga integradas. Se ha utilizado como base para desplegar este servidor una plantilla proporcionada en la asignatura *Aplicaciones Distribuidas en Internet*, y utiliza Flask framework para la comunicación web.

Se utilizará la librería Google Datastore NDB Client Library para el almacenamiento de los distintos elementos como es la información sobre los usuarios y las imágenes. Los perfiles diseñados para las diferentes entidades se muestran en el listado de código 5.4 donde se pueden observar los parámetros para cada entidad.

```
1  from google.appengine.ext import ndb

3  class KeyWord(ndb.Model):
4      keyWord = ndb.StringProperty()
5      count = ndb.IntegerProperty()

7  class Tag(ndb.Model):
8      tag = ndb.StringProperty()
9      probability = ndb.FloatProperty()

11 class Image(ndb.Model):
12     blobKey = ndb.StringProperty()
13     tags = ndb.StructuredProperty(Tag, repeated=True)
14     keyWords = ndb.StructuredProperty(KeyWord, repeated=True)
15     link = ndb.StringProperty()
16     siteLink = ndb.StringProperty()
17     flickrTags = ndb.StringProperty(repeated=True)

19 class User(ndb.Model):
20     phoneNumber = ndb.StringProperty()
21     regID = ndb.StringProperty()
22     keyWords = ndb.StructuredProperty(KeyWord, repeated=True)

24 class ImageUsed(ndb.Model):
25     image = ndb.KeyProperty(kind=Image)
26     user = ndb.KeyProperty(kind=User)
27     count = ndb.IntegerProperty()

29 class RelatedUsers(ndb.Model):
30     user1 = ndb.KeyProperty(kind=User)
31     user2 = ndb.KeyProperty(kind=User)
32     relation = ndb.IntegerProperty()

34     def get_related_user(self, user):
35         if self.user1 == user:
36             return self.user2
37         else:
```

```
38            return self.user1
```

Listado 5.4: Perfiles de las entidades del Datastore

Tarea: Estandarizar el formato de los números de teléfono

Debido a la problemática de que los usuarios guardan los diferentes teléfonos móviles con diferentes formatos, es necesario estandarizar estos. Muchos usuarios no añaden el prefijo de nacionalidad cuando crean un contacto, ya que por defecto se aplica el prefijo nacional, sin embargo, puede haber usuarios que tengan también contactos de otros países y por tanto precisen indicar la nacionalidad, o que simplemente al guardar un contacto que le ha llamado este se guarde utilizando el prefijo por defecto sin que el usuario lo seleccione. Es por esto, que es necesario aplicar un estándar para estos contactos, ya que a efectos prácticos, si un usuario español tiene guardado un contacto como 612345678, y en el sistema el otro usuario esta dado de alta como +34612345678, estos contactos deberían ser los mismos. En el listado 5.5 se puede ver como se realiza el preprocesado así como el envío de estos números preprocesados al servidor, asegurándose de que no se envían aquellos contactos que ya existen en la aplicación ni contactos repetidos para liberar carga del servidor. Al igual que en el caso anterior y en los futuros, se ha eliminado código poco relevante destinado al proceso del JSON de respuesta

```
2     String TAG = "refreshContactTab";
3     Boolean error = true;
4     Context mContext;

6     @Override
7     protected String doInBackground(Context... params) {
8         mContext = params[0];
9         String data = "";

11        HttpURLConnection httpURLConnection = null;

13        JSONObject jsonParams = processPhoneNumbers();

15        try {
16            httpURLConnection = ServerComunication.post(Constants.USERS_URL, jsonParams,
                  Constants.MAX_ATTEMPTS);
17            int code = httpURLConnection.getResponseCode();
18            if (code == HttpURLConnection.HTTP_OK) {
19                InputStream in = new BufferedInputStream(httpURLConnection.getInputStream()
                      );
20                data = NetworkUtils.readStream(in);
21                error = false;
22            }

24        } catch (IOException e) {
25            e.printStackTrace();
26        } finally {
27            if (null != httpURLConnection)
```

```
28                  httpURLConnection.disconnect();
29          }
30          return data;
31      }

33      @Override
34      protected void onPostExecute(String response) {
35          if (!error) {
36              updateContacts(response);
37          }
38          ContactTab.OnRefreshContactsFinish();
39      }

41      private JSONObject processPhoneNumbers(){

43          String contactId, completeNumber;

45          ArrayList<String> contactsAlreadyChecked = new ArrayList<>();

47          JSONObject jsonParams = new JSONObject();
48          JSONArray contactsJson = new JSONArray();

50          PhoneNumberUtil phoneUtil = PhoneNumberUtil.getInstance();
51          Phonenumber.PhoneNumber numberProto = null;
52          Cursor phones = null;

54          ContentResolver cr = mContext.getContentResolver();
55          Cursor users = cr.query(DataProvider.CONTENT_URI_PROFILE, new String[]{DataProvider
                .COL_PHONE_NUMBER}, null, null, null);
56          Cursor contacts = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null,
                 null);

58          //Add contacts to contactsAlreadyChecked that are already users
59          if(users!= null){
60              while (users.moveToNext()){
61                  contactsAlreadyChecked.add(users.getString(users.getColumnIndex(
                        DataProvider.COL_PHONE_NUMBER)));
62              }
63          }
64          if (contacts != null && contacts.getCount() > 0) {
65              while (contacts.moveToNext()) {
66                  contactId = contacts.getString(contacts.getColumnIndex(ContactsContract.
                        Contacts._ID));
67                  phones = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
68                          ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " +
                                contactId, null, null);

70                  while (phones != null && phones.moveToNext()) {
71                      String unformattedNumber = phones.getString(phones.getColumnIndex(
                            ContactsContract.CommonDataKinds.Phone.NUMBER));

73                      TelephonyManager manager = (TelephonyManager) mContext.getSystemService
                            (Context.TELEPHONY_SERVICE);
74                      String countryID = manager.getSimCountryIso().toUpperCase().trim();
75                      try {
76                          numberProto = phoneUtil.parse(unformattedNumber, countryID);
```

```
77                      } catch (NumberParseException e) {
78                          e.printStackTrace();
79                      }

81                      if (phoneUtil.isValidNumber(numberProto)) {
82                          completeNumber = Long.toString(numberProto.getCountryCode()) + Long
                                .toString(numberProto.getNationalNumber());

84                          if (!contactsAlreadyChecked.contains(completeNumber)) {
85                              contactsAlreadyChecked.add(completeNumber);
86                              JSONObject c = new JSONObject();
87                              try {
88                                  c.put(ServerSharedConstants.ID, contactId);
89                                  c.put(ServerSharedConstants.PHONE_NUMBER, completeNumber);
90                                  contactsJson.put(c);
91                              } catch (JSONException e) {
92                                  Log.d(TAG, e.toString());
93                              }
94                          }
95                      }
96                  }
97              }
98          }

100         if (phones != null && !phones.isClosed()) {
101             phones.close();
102         }

104         if (contacts != null && !contacts.isClosed()) {
105             contacts.close();
106         }
107         try {
108             jsonParams.put(ServerSharedConstants.CONTACTS, contactsJson);
109         } catch (JSONException e) {
110             Log.d(TAG, e.toString());
111         }

113         return jsonParams;
114     }

116     private void updateContacts(String response) {
117         ArrayList<ContactItem> newContacts = JSONProcess(response);

119         for (ContactItem contact : newContacts) {
120             contact.saveContact(mContext);
121         }
122     }
123 }
```

Listado 5.5: Actualización de contactosBBDD

Tarea: Sincronizar contactos

Es el servidor el que mantiene actualizada la información de los contactos, y por tanto, el usuario tiene que comunicarse con este para obtener dichos contactos. De esta manera, el

servidor recibirá todos los números de teléfono preprocesados previamente por la aplicación móvil, comprobará si existen usuarios registrados con ese número en el sistema, y le devolverá al usuario aquellos que efectivamente se encuentren dados de alta en el sistema. En el listado 5.6 se muestra la operación realizada en el servidor.

```
def checkRegisteredUsers():
    users = request.json[USERS]
    matches = []

    for u in users:
        key = ndb.Key(User, u[USER_ID])
        user = key.get()
        if user:
            matches.append(u)

    return make_response(jsonify({USERS: matches}), http.OK)
```

Listado 5.6: Sincronización de contactos

Tarea: Diseño de un sistema de mensajes

Aunque ya se ha realizado un trabajo previo al diseñar la BBDD en la aplicación para almacenar los mensajes es necesario establecer un estándar a seguir en estos mensajes. Debido a que se tratará de mensajes tanto de tipo textual como multimedia, es preciso poder realizar esta distinción. Por esta razón, los mensajes estarán compuestos por dos campos, por un lado el contenido del mensaje y por otro el tipo. El tipo indicará como su propio nombre indica el formato que tendrá el mensaje, y el contenido consistirá en el texto enviado si se trata de un mensaje de texto, o si por el contrario se trata de un archivo multimedia un identificador con el que poder descargar este archivo.

En el listado de código 5.7 se muestra la configuración de la clase java que representa a los mensajes obviando los getter y setter.

```
public class MessageItem {

    private String from, to, type, msg, localResource;

    public static final String TEXT_TYPE = "text";
    public static final String IMG_TYPE = "img";

    public MessageItem(String from, String to, String type, String msg, String
        localResource) {
        this.from = from;
        this.to = to;
        this.type = type;
        this.msg = msg;
        this.localResource = localResource;
    }

    public MessageItem(String from, String to, String type, String msg) {
        this.from = from;
        this.to = to;
```

```
19          this.type = type;
20          this.msg = msg;
21      }

23      public MessageItem(String from, String type, String msg) {
24          this.from = from;
25          this.type = type;
26          this.msg = msg;
27          this.to = null;
28      }

30      public JSONObject messageToJSON(){
31          JSONObject jsonParam = new JSONObject();
32          try {
33              jsonParam.put(ServerSharedConstants.FROM, from);
34              jsonParam.put(ServerSharedConstants.TYPE, type);
35              jsonParam.put(ServerSharedConstants.MSG, msg);
36          } catch (JSONException e) {
37              e.printStackTrace();
38          }
39          return jsonParam;
40      }

42      public void saveMessageSent(Context mContext){
43          ContentValues values = new ContentValues(2);
44          values.put(DataProvider.COL_TO, to);
45          if(localResource == null){
46              values.put(DataProvider.COL_MSG, msg);
47          } else{
48              values.put(DataProvider.COL_MSG, localResource);
49          }
50          values.put(DataProvider.COL_TYPE, type);
51          mContext.getContentResolver().insert(DataProvider.CONTENT_URI_MESSAGES, values);
52      }
53      public void saveMessageReceived(Context mContext){
54          ContentValues values = new ContentValues(2);
55          if(localResource == null){
56              values.put(DataProvider.COL_MSG, msg);
57          } else{
58              values.put(DataProvider.COL_MSG, localResource);
59          }       values.put(DataProvider.COL_TYPE, type);
60          values.put(DataProvider.COL_FROM, from);
61          mContext.getContentResolver().insert(DataProvider.CONTENT_URI_MESSAGES, values);
62      }
63  }
```

Listado 5.7: Clase «MessageItem.java»

Tarea: Integración de GCM con la aplicación

Se ha utilizado la Application Programming Interface (API) de GCM tanto en el cliente como en el servidor para poder interactuar con este servicio. En primer lugar es necesario obtener un archivo de configuración y añadirlo a la aplicación. Además de añadir las dependencias necesarias para poder utilizar la API, también es necesario configurar el proyecto

para utilizar los servicios del SDK de Google Play Service, y asegurarse de que el dispositivo donde se instala tiene disponibles estos servicios. Es necesario también configurar el Manifest de la aplicación e incluir una serie de permisos, así como de servicios que se ejecutarán en segundo plano y que gestionarán entre otras cosas los mensajes que llegan así como los tokens o identificadores necesarios para poder enviar los mensajes, aunque la aplicación no este ejecutándose.

Cada vez que la aplicación se instale en el dispositivo, será necesario registrar este dispositivo con los servidores de conexión de GCM. Para ello se solicitará un token o testigo que será necesario para enviar mensajes a ese usuario en cuestión, por lo que será necesario enviarlo al servidor donde se almacenará.

En el listado de código 5.8 se muestra el código que se ejecutaría cada vez que se recibe un mensaje.

```
1  public class MyGcmListenerService extends GcmListenerService {
2
3      private static final String TAG = "MyGcmListenerService";
4      private static final int MY_NOTIFICATION_ID = 1;
5      private static final String IMAGE = "Image received";
6
7      @Override
8      public void onMessageReceived(String senderID, Bundle data) {
9          String from = data.getString(ServerSharedConstants.FROM);
10         String type = data.getString(ServerSharedConstants.TYPE);
11         String msg = data.getString(ServerSharedConstants.MSG);
12         Log.d(TAG, "SenderID: " + senderID);
13         Log.d(TAG, "Message: " + msg);
14
15         MessageItem messageItem = new MessageItem(from, type, msg);
16
17     if(type.equals(MessageItem.IMG_TYPE)){
18             new ReceivedImageDownloadTask(this, messageItem).execute();
19             msg = IMAGE;
20         }
21         else {
22             messageItem.saveMessageReceived(this);
23         }
24         sendNotification(msg);
25     }
26     private void sendNotification(String msg) {
27         Intent intent = new Intent(this, MainActivity.class);
28         intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
29         PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
30                 PendingIntent.FLAG_ONE_SHOT);
31
32         Uri defaultSoundUri= RingtoneManager.getDefaultUri(RingtoneManager.
               TYPE_NOTIFICATION);
33         NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(
               this)
34                 .setSmallIcon(android.R.drawable.stat_notify_chat)
35                 .setContentTitle("Message")
36                 .setContentText(msg)
```

```
                .setAutoCancel(true)
                .setSound(defaultSoundUri)
                .setContentIntent(pendingIntent);

        NotificationManager notificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        notificationManager.notify(MY_NOTIFICATION_ID, notificationBuilder.build());
    }
}
```

Listado 5.8: Recepción de un mensaje

Tarea: Integración de GCM con el servidor

También es necesario integrar la API de GCM para python en el servidor. Para ello es necesario añadir esta la librería en una carpeta en la que se encuentran todas las librerías que se utilizan en el servidor. Para que no sea necesario añadir esta librería a mano cada vez que se realicen cambios en estas, es preciso indicarlo en el archivo `requirements.txt`. El servidor será el encargado de enviar hacia el destinatario adecuado el mensaje recibido por el usuario que lo ha enviado. Este usuario ha indicado el número de teléfono al que se lo quiere enviar, y es tarea del servidor relacionar el número con el token que el usuario ha registrado previamente en el servidor, y enviárselo.

Se muestra el código relativo al envío del mensaje al destinatario del mismo por parte del servidor en el listado 5.9.

```
@app.route('/users/<path:id_user>/send', methods = [POST])
def manager_user_send(id_user):
    if request.method == POST:
        return sendMsg(id_user)

def sendMsg(id_user):
    u = request.json
    key = ndb.Key(User, id_user)
    user = key.get()

    params = json.dumps({"data": {FROM: u[FROM], TYPE: u[TYPE], MESSAGE: u[MESSAGE]}, "to":
         user.regID})

    conn = http.HTTPSConnection(GCM_SEND_URL)
    conn.request("POST", "", params, GCM_HEADERS)

    response = conn.getresponse()

    status = response.status
    data = response.read()

    conn.close

    return make_response(jsonify({'response':data}), status)
```

Listado 5.9: Envío del mensaje por parte del servidor

Sprint Review

Durante esta reunión se han mostrado los avances, y el Product Owner ha podido comprobar por sí mismo como la aplicación sincroniza los contactos, además de observar como funciona el envío de mensajes entre varios dispositivos.

Sprint Retrospective

Se ha evaluado el trabajo realizado, y se ha acordado que es necesario un mayor aprendizaje sobre la plataforma Google Cloud Platform para aprovechar en el mayor grado posible las herramientas y ventajas que ofrece. Pese a que se ha conseguido desplegar el servidor es necesario aprender sobre el manejo de otras herramientas como los registros, o las consultas sobre el Datastore. También se ha discutido el hecho de que se ha producido un retraso debido a la integración de GCM la cual ha requerido mas tiempo del que se había previsto en un principio.

5.3.3 Sprint 3

Para este Sprint, en la reunión previa se ha acordado que se realizará únicamente la HU 5. Debido al retraso provocado por el anterior Sprint se desea presentar cuanto antes un avance notorio y significativo para entregar al cliente, y se considera que el hecho de enviar y recibir mensajes será un avance que el Product Owner valorará muy positivamente. Se muestran en el cuadro 5.5 la HU seleccionada que conforma el Sprint Backlog de esta iteración.

Tarea: Integración del Blobstore con el servidor

Al igual que el Datastore, el Blobstore es un servicio proporcionado por Google App Engine, por lo que no es necesaria la descarga de ninguna API adicional. Antes de nada, los blobs son contenedores que permiten almacenar archivos que exceden el tamaño permitido por el Datastore. Estos se usarán para almacenar aquellas imágenes que se decida incluir en el repositorio de imágenes propio que se creará. Para subir un archivo a un blob, en primer lugar es necesario realizar una petición para obtener una dirección a la que subir el archivo deseado. Posteriormente será necesario mediante un formulario web enviar el archivo multimedia deseado junto con la información que se considere oportuna a la dirección generada previamente. Será necesario implementar un manejador que será el encargado de a partir de los datos proporcionados por el usuario, almacenará el recurso en un blob, el cual generará una clave que será la que se utilizará para recuperar el archivo del blob. En este caso, se creará una entidad imagen en el servidor estableciendo como clave única de esta entidad la clave del blob generada. Posteriormente se actualizará la información correspondiente a la imagen utilizando esta clave.

Historia de usuario	
Número: 4	**Rol:** Usuario
Esfuerzo:	**Iteración:** 2
Nombre: Comunicación con otros usuarios	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Como usuario quiero poder enviar y recibir mensajes de otros usuarios.	
Validación: Dos usuarios desde distintos dispositivos se comunican satisfactoriamente.	
Tareas: Tarea: Diseño de un sistema de mensajes para el envío de texto y multimedia. Tarea: Integración de GCM con la aplicación. Tarea: Integración de GCM con el servidor.	

Cuadro 5.4: Historia de Usuario 4

Historia de usuario	
Número: 5	**Rol:** Usuario
Esfuerzo:	**Iteración:** 3
Nombre: Envío de imágenes	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Como usuario quiero poder enviar imágenes a mis contactos, además de poder recibir y visualizar las que me llegan.	
Validación: Se pueden enviar y recibir imágenes entre diversos dispositivos.	
Tareas: Tarea: Integración de Google Blobstore en el servidor. Tarea: Almacenar imágenes en blobs. Tarea: Enviar imágenes a otro usuario. Tarea: Descargar y almacenar las imágenes en local.	

Cuadro 5.5: Historia de Usuario 5

En el listado 5.10 se muestra como el servidor manejaría las diferentes peticiones en primer lugar para obtener una dirección sobre la que subir el archivo, y en segundo lugar como este servidor gestiona el envío de un formulario por parte la aplicación cliente.

```
@app.route("/upload_form", methods=[GET])
def manager_upload_form():
    if request.method == GET:
        return upload()

def upload():
    uploadUri = blobstore.create_upload_url('/upload_photo')
    return make_response(uploadUri, http.OK)

@app.route("/upload_photo", methods=[POST])
def manager_upload_photo():
    if request.method == POST:
        return upload_photo()

def upload_photo():
    f = request.files['file']
    header = f.headers['Content-Type']
    parsed_header = parse_options_header(header)
    blob_key = parsed_header[1]['blob-key']

    image = Image(blobKey=blob_key, id=blob_key)

    image.put()

    return make_response(blob_key, http.CREATED)
```

Listado 5.10: Gestión por parte del servidor de la creación de blobs

Tarea: Almacenamiento de imágenes en Blobs

Como se ha mencionado anteriormente, es necesario que la aplicación cliente solicite en primer lugar la generación de una dirección URL sobre la que posteriormente este mismo usuario subirá un formulario web que contenga la imagen que desea subir. En el listado de código 5.11 se muestra este proceso. Es necesario indicar que debido a que aún no se ha añadido la funcionalidad de recurrir a imágenes que se encuentren en la red, se realizarán las pruebas con imágenes que los usuarios tengan en sus dispositivos.

```
public class UploadImageTask extends AsyncTask<Void, Void, String> {
    Context mContext;
    MessageItem messageItem;
    SuggestedImage suggestedImage;

    public UploadImageTask(Context context, SuggestedImage suggestedImage, MessageItem
        messageItem){
        this.suggestedImage = suggestedImage;
        this.mContext = context;
        this.messageItem = messageItem;
    }
```

```
12      @Override
13      protected String doInBackground(Void... params) {
14          String imgURL = suggestedImage.getBlobUrl();
15          if(imgURL == null){
16              String blobURL = getBlobURL();

18              imgURL = uploadImage(blobURL);
19              suggestedImage.setBlobUrl(imgURL);
20              putImageInformation();
21          }

23          return imgURL;
24      }

26      private String getBlobURL(){
27          String blobUrl = "";
28          HttpURLConnection httpUrlConnection = null;

30          try {
31              httpUrlConnection = (HttpURLConnection) new URL(Constants.UPLOAD_FORM_URL)
32                      .openConnection();

34              switch (httpUrlConnection.getResponseCode()){
35                  case HttpURLConnection.HTTP_OK:
36                      InputStream in = new BufferedInputStream(
37                              httpUrlConnection.getInputStream());

39                      blobUrl = NetworkUtils.readStream(in);
40                      break;
41                  case HttpURLConnection.HTTP_NOT_FOUND:
42                      InputStream err = new BufferedInputStream(httpUrlConnection.
                            getErrorStream());

44                      blobUrl = NetworkUtils.readStream(err);
45                      break;
46              }
47          } catch (MalformedURLException exception) {
48              Log.e(TAG, "MalformedURLException");
49          } catch (IOException exception) {
50              Log.e(TAG, "IOException");
51          } finally {
52              if (null != httpUrlConnection)
53                  httpUrlConnection.disconnect();
54          }
55          return blobUrl;
56      }

58      private String uploadImage(String imgUploadURL){
59          String imgURL = "";
60          OkHttpClient client = new OkHttpClient();
61          RequestBody requestBody = new MultipartBody.Builder()
62                  .setType(MultipartBody.FORM)
63                  .addFormDataPart("file", "image.png", RequestBody.create(MediaType.parse("
                        image/png"), new File(messageItem.getLocalResource())))
64                  .addFormDataPart("title", "My photo")
65                  .build();
```

```
67          Request request = new Request.Builder()
68                  .url(imgUploadURL)
69                  .post(requestBody)
70                  .build();
71          try {
72              Response response = client.newCall(request).execute();
73              imgURL = response.body().string();
74              Log.d("response", "uploadImage:"+imgURL);
75          } catch (IOException e) {
76              e.printStackTrace();
77          }
78          return imgURL;
79      }

81      @Override
82      protected void onPostExecute(String imgURL) {
83          messageItem.setMsg(imgURL);
84          new SendMessageTask(mContext, messageItem).execute();
85      }
86  }
```

Listado 5.11: Envío de la imagen al servidor para su almacenamiento

Tarea: Enviar imágenes a otros usuarios

Al igual que se ha hecho anteriormente con los mensajes es necesario ser capaces de enviar las imágenes. Como ya se ha indicado, lo que se enviará será la dirección que se necesita para descargarla. En concreto se enviará la clave del blob que identifica a esta imagen, y que permitirá enviarla al servidor el cual se encargará de devolvernos la imagen. En el listado 5.12 se muestra como el servidor retorna la imagen a partir de la clave del blob.

```
1  @app.route("/images/<path:id_img>", methods=[GET, PUT])
2  def manager_image(id_img):
3      if request.method == GET:
4          return download_image(id_img)

6  def download_image(id_img):
7      blob_info = blobstore.get(id_img)
8      response = make_response(blob_info.open().read())
9      response.headers['Content-Type'] = blob_info.content_type
10     return response
```

Listado 5.12: Envío de la imagen desde el servidor al cliente

Tarea: Descargar y almacenar imágenes en local

Cada vez que llegue una imagen a través de un mensaje, se detectará a partir de la información que contenga el tipo de mensaje, y se procederá a descargar y almacenar su contenido. Esto se realiza mediante la clase `ReceivedImageDownloadTask.java`. En el listado 5.13 se muestra el proceso a seguir.

```java
public class ReceivedImageDownloadTask extends AsyncTask<Void, Void, String> {
    private static final String TAG = "ImageDownload";
    Context mContext;
    MessageItem messageItem;

    public ReceivedImageDownloadTask(Context context, MessageItem messageItem){
        this.mContext = context;
        this.messageItem = messageItem;
    }

    @Override
    protected String doInBackground(Void... params) {
        Bitmap bitmap = null;
        HttpURLConnection httpUrlConnection = null;
        String imgPath;

        try {
            httpUrlConnection = (HttpURLConnection) new URL(Constants.IMAGES_URL + "/" +
                messageItem.getMsg())
                    .openConnection();

            switch (httpUrlConnection.getResponseCode()){
                case HttpURLConnection.HTTP_OK:
                    InputStream in = new BufferedInputStream(
                            httpUrlConnection.getInputStream());

                    bitmap = BitmapFactory.decodeStream(in);
                    break;
            }
        } catch (MalformedURLException exception) {
            Log.e(TAG, "MalformedURLException");
        } catch (IOException exception) {
            Log.e(TAG, "IOException");
        } finally {
            if (null != httpUrlConnection)
                httpUrlConnection.disconnect();
        }
        imgPath = MediaStore.Images.Media.insertImage(mContext.getContentResolver(), bitmap
            , "img" , null);
        return imgPath;
    }

    @Override
    protected void onPostExecute(String imgPath) {
        Uri uri = Uri.parse(imgPath);
        String path = Utils.getRealPathFromURI(mContext,  uri);
        messageItem.setLocalResource(path);
        messageItem.saveMessageReceived(mContext);
    }
}
```

Listado 5.13: Descarga y almacenamiento de la imagen en el cliente

Sprint Review

Se ha mostrado el avance al Product Owner el cual ha comprobado que el proyecto va tomando forma, y promete.

Sprint Retrospective

El tiempo perdido durante el anterior Sprint se ha conseguido recuperar en parte a lo largo de esta iteración. Además, la estructura del sistema está más consolidada.

5.3.4 Sprint 4

Debido al haber realizado el anterior Sprint en menor tiempo del previsto, se decide acometer para esta iteración con las HU 6 y 8. Esto se debe a que no se pueden acometer al mismo tiempo las historias 5 y 6 debido a que ambas son demasiado complejas y se tardaría demasiado en proporcionar un producto con avances. Estas HU se muestran en los cuadros 5.6

Historia de usuario	
Número: 6	**Rol:** Usuario
Esfuerzo:	**Iteración:** 4
Nombre: Búsqueda de imágenes a partir de una entrada	
Desarrollador responsable: Victor Gualdras de la Cruz	
Descripción: Como usuario quiero poder buscar imágenes a partir de la consulta que introduzca.	
Validación: En una conversación, uno de los usuarios puede a partir de una consulta seleccionar una de las imágenes qu se muestran y enviársela al otro usuario.	
Tareas: Tarea: Mecanismo para la búsqueda de imágenes en diferentes portales. Tarea: Extraer información de las imágenes seleccionadas y almacenar esta información junto con las imágenes. Tarea: Integrar la búsqueda y descarga de estas imágenes con su envío con el actual sistema.	

Cuadro 5.6: Historia de Usuario 6

Tarea: Búsqueda de imágenes en diferentes portales

Debido a que se necesitarán imágenes que representen las diferentes necesidades de los usuarios, y ya que no se está en posesión de un conjunto de estas imágenes, será necesario recurrir a otras fuentes. Para ello, se configurará el motor de búsqueda Google Custom Search Engine, el cuál permite realizar búsquedas en los dominios web que se le indiquen. Se utilizarán dominios que provean de imágenes con libre licencia de uso, ya que se utilizarán estas imágenes y se almacenarán en un servidor propio. Para utilizar este servicio es necesario la creación de un motor de búsqueda, lo cual resulta bastante sencillo, y posteriormente obtener

una serie de credenciales que se utilizarán para tener acceso a estos servicios de búsqueda. Se configurará el motor de búsqueda con los sitios en los que se desea realizar estas, y será posible cambiar los sitios de manera dinámica sin que las aplicaciones de usuario o el servidor tengan que sufrir cambio ninguno. Además, en las búsquedas se podrán especificar de manera personalizada y en función de ciertos parámetros los sitios web en los que se realiza. Se utilizará la API CustomSearch disponible para java debido a que esto nos ahora el tener que estar tratando los datos que se envían y reciben en esta búsqueda, si no que con la esta API los tendremos directamente en formato java listo para utilizar, siendo más sencillo la recuperación y codificación de estos.

En el listado **??** se puede observar como mediante una clave, y el identificador cx que indica el motor de búsqueda a seleccionar (debido a que se pueden utilizar varios configurados con perfiles diferentes), se realiza la búsqueda mediante una entrada en modo texto. Además se le pueden indicar los sitios exactos donde debe buscar, en caso de que la consulta así lo requiera.

```
1  private List<Result> searchGSE(@Nullable String site){
2          Customsearch customsearch = new Customsearch(new NetHttpTransport(), new
               JacksonFactory(), null);
3          com.google.api.services.customsearch.Customsearch.Cse.List list;
4          Search results;
5          List<Result> searchResults = null;

7          try {
8              list = customsearch.cse().list(search);
9              list.setKey(key);
10             list.setCx(cx);
11             list.setSearchType("image");
12             if(site != null){
13                 list.setSiteSearch(search);
14             }

16             results = list.execute();
17             searchResults = results.getItems();
18         } catch (IOException e) {
19             e.printStackTrace();
20         }

22         return searchResults;
23     }
```

Listado 5.14: Búsqueda de imágenes utilizando Custom Search

Además, se añadirán al servidor los pictogramas proporcionados por ARASAAC.

Tarea: Extraer información de las imágenes seleccionadas

Para poder utilizar estas imágenes en el futuro dentro del sistema de recomendación, será necesario extraer información de las mismas. Para este propósito se utilizará Google Cloud Vision. Cloud Vision es una API integrada en los servicios proporcionados por Google Cloud

Platform la cual permite extraer entre otras cosas información relativa a imágenes tales como etiquetas junto con la probabilidad estimada por este sistema de que se adapten a la realidad. Proporcionan otras funcionalidades, pero en este caso la que resulta de utilidad es esta.

Se muestra el proceso seguido para realizar la extracción de la información a una imagen mediante esta API en el listado **??**.

```
public class ImageLabelDetectionTask extends AsyncTask<Void, Void, SuggestedImage> {
    SuggestedImage suggestedImage;
    ImageInteractionListener mListener;

    public ImageLabelDetectionTask(Activity activity, SuggestedImage suggestedImage) {
        this.suggestedImage = suggestedImage;
        this.mListener = (ImageInteractionListener) activity;
    }

    @Override
    protected SuggestedImage doInBackground(Void... params) {
        try {
            HttpTransport httpTransport = AndroidHttp.newCompatibleTransport();
            JsonFactory jsonFactory = GsonFactory.getDefaultInstance();

            Vision.Builder builder = new Vision.Builder(httpTransport, jsonFactory, null);
            builder.setVisionRequestInitializer(new
                    VisionRequestInitializer(CLOUD_VISION_API_KEY));
            Vision vision = builder.build();

            BatchAnnotateImagesRequest batchAnnotateImagesRequest =
                    new BatchAnnotateImagesRequest();

            ArrayList<AnnotateImageRequest> requests = new ArrayList<>();
            Bitmap bitmap = suggestedImage.getBitmap();
            AnnotateImageRequest annotateImageRequest = new AnnotateImageRequest();

            Image base64EncodedImage = new Image();

            ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.JPEG, 90, byteArrayOutputStream);
            byte[] imageBytes = byteArrayOutputStream.toByteArray();

            base64EncodedImage.encodeContent(imageBytes);
            annotateImageRequest.setImage(base64EncodedImage);

            annotateImageRequest.setFeatures(new ArrayList<Feature>() {{
                Feature labelDetection = new Feature();
                labelDetection.setType("LABEL_DETECTION");
                labelDetection.setMaxResults(10);
                add(labelDetection);
            }});
            requests.add(annotateImageRequest);

            batchAnnotateImagesRequest.setRequests(requests);

            Vision.Images.Annotate annotateRequest =
                    vision.images().annotate(batchAnnotateImagesRequest);
            annotateRequest.setDisableGZipContent(true);
```

```
50              Log.d(TAG, "created Cloud Vision request object, sending request");

52              BatchAnnotateImagesResponse response = annotateRequest.execute();
53              suggestedImage.setTags(convertResponseToMap(response));

55          } catch (GoogleJsonResponseException e) {
56              Log.d(TAG, "failed to make API request because " + e.getContent());
57          } catch (IOException e) {
58              Log.d(TAG, "failed to make API request because of other IOException " +
59                      e.getMessage());
60          }
61          return suggestedImage;
62      }

64      @Override
65      protected void onPostExecute(SuggestedImage suggestedImage) {
66          mListener.onImageLabeled(suggestedImage);
67      }
68  }
```

Listado 5.15: Búsqueda de imágenes utilizando Custom Search

Tarea: Integrar la búsqueda y el envío de imágenes

Sprint Review

Sprint Retrospective

5.3.5 Sprint

Tarea:

Sprint Review

Sprint Retrospective

Capítulo 6

Resultados

Capítulo 7

Conclusiones y trabajos futuros

Para terminar, es necesario indicar que se ha cumplido con el objetivo de desarrollar una aplicación de mensajería capaz de recomendar imágenes a los usuarios a partir de sus gustos. Pese a núcleo principal que se pretendía conseguir era el sistema de recomendación, siendo la aplicación una mera herramienta para poder interactuar con este, se ha establecido una estructura sobre la que se podrá trabajar en un futuro. Existe un sistema de comunicaciones muy confiable, así como una muy buena base respecto a la aplicación de mensajería sobre la que en un futuro se podría trabajar para ampliar. Además, se ha desarrollado el sistema para almacenar una gran cantidad de información útil que podrá ser usada en el futuro para combinar con otras técnicas y poder mejorar aún más el sistema.

Como se puede comprobar en la mayoría de los portales webs en los que existen una gran cantidad de elementos, los sistemas de recomendación, pese a ser relativamente nuevos, tienen mucho futuro por delante. En este proyecto se ha conseguido desarrollar e integrar uno de estos sistemas en una aplicación real, en un tipo de aplicación en el que no es común, pero en el que tiene mucho potencial. Por tanto, este tipo de sistemas es uno de los que mayor futuro presentan en el ámbito de la computación

Alguno de los objetivos iniciales que se propusieron cuando se comenzó el trabajo no se han podido llevar a cabo debido a la magnitud de este trabajo. Destaca entre ellos la prueba una gran cantidad de usuarios reales para poder determinar el verdadero potencial y funcionamiento de este sistema. Otro objetivo que se planteó, pero que resultaba menos interesante que el anterior es la opción de incluir gifs además de imágenes. Como se verá más adelante, estas son algunas de las propuestas para posibles ampliaciones futuras del sistema.

En el plano personal, este proyecto me ha permitido aprender y mejorar en el empleo de un gran número de herramientas y tecnologías, así como de adquirir nuevos marcos teóricos muy útiles para la rama en la que quiero (y de hecho ya he empezado) especializarme. Entre los aptitudes tecnológicas, destacan la programación para la plataforma Android, y el aprendizaje sobre el funcionamiento de herramientas para el trabajo en la nube, como Google Cloud Platform y el resto de tecnologías asociadas a esta. También me ha parecido muy útil e interesante la base de datos Wordnet, que permite un gran abanico de posibilidades a la

hora de tratar con lenguaje natural. Además, este trabajo me ha permitido mejorar en aquellas destrezas que en menor medida ya poseía como en el diseño y desarrollo de software, o la gestión y planificación.

En el contexto teórico, he aprendido mucho sobre los sistemas de recomendación, sistemas que creo que tienen mucha utilidad en un gran número de campos, y que al no tener mucho tiempo de vida, suponen un campo muy interesante sobre el que poder investigar y especializarse. También me ha mostrado como numerosos algoritmos o técnicas pueden combinarse entre sí para crear herramientas y sistemas más potentes.

7.1 Trabajos futuros

Debido al carácter de uno de los componentes de este sistema que es la aplicación de mensajería, existen en este sentido una cantidad de ampliaciones y mejoras prácticamente infinita, como mejoras en la interfaz, adición de nuevas funcionalidades como posibilidades de creación de grupos, configuración de perfil, listas de difusión... Sin embargo, el propósito de este proyecto nunca ha sido ser la competencia de Whatsapp o Telegram, si no el desarrollo de un sistema de recomendación integrado en una aplicación de estas características. Es por eso que se hará especial énfasis en las mejoras y ampliaciones que se pueden añadir para mejorar este proceso de recomendación, además de mejorar la experiencia de comunicación de los usuarios.

- Se plantea la posibilidad de trabajar con otro tipo de archivos multimedia como pueden ser gifs, pequeños videos o incluso efectos de sonido, los cuales aumentarían las posibilidades de la comunicación mediante medios no escritos. Por supuesto, se utilizarían mecanismos propios de los sistemas de recomendación para proporcionar estos recursos a los usuarios.
- Una de las disciplinas de la computación que mayor aplicación tienen combinadas con los sistemas de recomendación es el aprendizaje automático. Esto se debe a que utilizando técnicas de aprendizaje automático es posible inferir relaciones existentes entre los usuarios que posteriormente pueden resultar de gran utilidad en los algoritmos colaborativos. También podría ser utilizado en la clasificación de los perfiles de cada usuario estableciendo categorías en los que estos están interesados, permitiendo aplicar filtros a las búsquedas. Es esta, de manera personal la vía de expansión más atractiva junto con el procesamiento de la entrada que se mostrará a continuación.
- El aumento de sitios de los que se extraer recursos. Esto hace referencia tanto a incluir nuevos sitios web como ha crear algún tipo de portal en los que la gente puede subir nuevas imágenes o recursos multimedia y etiquetarlos así como permitir que otros usuarios puedan valorar estos. Además sería útil obtener nuevas formas de extraer información de estas las imágenes, ya sea de manera automática como permitiendo que los usuarios puedan etiquetarlas.

- La mejora del procesamiento de la entrada es una de las posibles vías de mejora que mayor utilidad aportarían al sistema. Se podría desarrollar un procesador de lenguajes, que fuese capaz de analizar una frase determinando tanto la verdadera forma gramatical de una palabra como su significado léxico, de manera que la extracción de sinónimos y palabras relacionadas fuese siempre correcta. Sin embargo esta no es una tarea sencilla por la naturaleza del lenguaje natural.

Referencias

[AFG+10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[All90] Robert B Allen. User models: theory, method, and practice. *International Journal of man-machine Studies*, 32(5):511–543, 1990.

[BEL+07] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, y Domonkos Tikk. KDD Cup and Workshop 2007. *SIGKDD Explor. Newsl.*, 9(2):51–52, Diciembre 2007. url: http://doi.acm.org/10.1145/1345448.1345459.

[BS97] Marko Balabanović y Yoav Shoham. Fab: Content-based, Collaborative Recommendation. *Commun. ACM*, 40(3):66–72, Marzo 1997. url: http://doi.acm.org/10.1145/245108.245124.

[Bur02] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Noviembre 2002. url: http://dx.doi.org/10.1023/A:1021240730564.

[Bur07] Robin Burke. Hybrid web recommender systems. En *The adaptive web*, páginas 377–408. Springer, 2007.

[CS00] Paul Cotter y Barry Smyth. PTV: Intelligent Personalised TV Guides. En *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, páginas 957–964. AAAI Press, 2000. url: http://dl.acm.org/citation.cfm?id=647288.760209.

[Fin14] Mark Alan Finlayson. Java libraries for accessing the princeton wordnet: Comparison and evaluation. En *Proceedings of the 7th Global Wordnet Conference, Tartu, Estonia*, 2014.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, y Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM*, 35(12):61–70, Diciembre 1992. url: http://doi.acm.org/10.1145/138859.138867.

[HCZ04] Zan Huang, Hsinchun Chen, y Daniel Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, Enero 2004. url: http://doi.acm.org/10.1145/963770.963775.

[Her00] Jonathan Lee Herlocker. *Understanding and improving automated collaborative filtering systems*. PhD thesis, Citeseer, 2000.

[HK70] E. M. Housman y E. D. Kaskela. State of the Art in Selective Dissemination of Information. *IEEE Transactions on Engineering Writing and Speech*, 13(2):78–83, Sept 1970.

[HSS01] Uri Hanani, Bracha Shapira, y Peretz Shoval. Information Filtering: Overview of Issues, Research and Systems. *User Modeling and User-Adapted Interaction*, 11(3):203–259, Agosto 2001. url: http://dx.doi.org/10.1023/A:1011196000674.

[IIS04] Sylvia Ilieva, Penko Ivanov, y Eliza Stefanova. Analyses of an agile methodology implementation. En *Euromicro Conference, 2004. Proceedings. 30th*, páginas 326–333. IEEE, 2004.

[Lan95] Ken Lang. Newsweeder: Learning to filter netnews. En *Proceedings of the 12th international conference on machine learning*, páginas 331–339, 1995.

[Lee01] Wee Sun Lee. Collaborative learning for recommender systems. En *ICML*, volume 1, páginas 314–321, 2001.

[LMY+12] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, y Tao Zhou. Recommender systems. *Physics Reports*, 519(1):1 – 49, 2012. Recommender Systems. url: http://www.sciencedirect.com/science/article/pii/S0370157312000828.

[MGT86] T. W. Malone, K. R. Grant, y F. A. Turbak. The Information Lens: An Intelligent System for Information Sharing in Organizations. *SIGCHI Bull.*, 17(4):1–8, Abril 1986. url: http://doi.acm.org/10.1145/22339.22340.

[MR00] Raymond J. Mooney y Loriene Roy. Content-based Book Recommending Using Learning for Text Categorization. En *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, páginas 195–204, New York, NY, USA, 2000. ACM. url: http://doi.acm.org/10.1145/336597.336662.

[Paz99] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, Diciembre 1999. url: http://dx.doi.org/10.1023/A:1006544522159.

[PB97] Michael Pazzani y Daniel Billsus. Learning and Revising User Profiles: The Identification ofInteresting Web Sites. *Mach. Learn.*, 27(3):313–331, Junio 1997. url: http://dx.doi.org/10.1023/A:1007369909943.

[RAC+02] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, y John Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. En *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, páginas 127–134, New York, NY, USA, 2002. ACM. url: http://doi.acm.org/10.1145/502716.502737.

[RIS+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, y John Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. En *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, páginas 175–186, New York, NY, USA, 1994. ACM. url: http://doi.acm.org/10.1145/192844.192905.

[Rob04] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*, 60(5):503–520, 2004.

[RV97] Paul Resnick y Hal R. Varian. Recommender Systems. *Commun. ACM*, 40(3):56–58, Marzo 1997. url: http://doi.acm.org/10.1145/245108.245121.

[Sch04] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[SKB+98] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, y John Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. En *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, páginas 345–354, New York, NY, USA, 1998. ACM. url: http://doi.acm.org/10.1145/289444.289509.

[SWY75] G. Salton, A. Wong, y C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, Noviembre 1975. url: http://doi.acm.org/10.1145/361219.361220.

[TC00] Thomas Tran y Robin Cohen. Hybrid recommender systems for electronic commerce. En *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*, 2000.

[YST+04] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, y Hans-Peter Kriegel. Probabilistic Memory-Based Collaborative Filtering. *IEEE Trans. on Knowl. and Data Eng.*, 16(1):56–69, Enero 2004. url: http://dx.doi.org/10.1109/TKDE.2004.1264822.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20160824) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`