



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

**INGENIERÍA
EN INFORMÁTICA**

PROYECTO FIN DE CARRERA

SocialFrame: Aplicación móvil amigable para la accesibilidad de personas dependientes o mayores a redes sociales

Juan Miguel Torres Triviño

Junio, 2012



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

Departamento de Tecnologías y Sistemas de Información

PROYECTO FIN DE CARRERA

SocialFrame: Aplicación móvil amigable para la accesibilidad
de personas dependientes o mayores a redes sociales

Autor: Juan Miguel Torres Triviño
Director: Dr. Ramón Hervás Lucas

Junio, 2012

Juan Miguel Torres Triviño

Ciudad Real – España

E-mail: juanmiguel.torres@alu.uclm.es

Web site: <https://github.com/ilidian/PFC-Socialframe>

© 2012 Juan Miguel Torres Triviño

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

TRIBUNAL:

Presidente:

Vocal 1:

Vocal 2:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE

VOCAL 1

VOCAL 2

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Fdo.:

Resumen

El objetivo primordial del Ambient Assisting Living (AAL)[1] es facilitar y mejorar la calidad de vida en personas mayores o dependientes de otras mediante el uso de tecnologías de información y comunicación. Dentro del ámbito del AAL surgen multitud de aplicaciones móviles que se centran en distintos aspectos de la vida como las relaciones sociales. Este tipo de relaciones han tomado un rumbo diferente desde que entrasen en escena las redes sociales en la Web, como por ejemplo Twitter, Google+ o Facebook entre otras.

Existe un gran problema alrededor de las redes sociales y es la dificultad que tienen las personas mayores o dependientes en acceder a ellas. Esto ocurre por diversos motivos: complejidad a la hora de navegar, mucha información accesible mostrada, gran variedad de acciones a realizar y rápida expansión de estas redes sociales sin una preparación previa, sobretodo en el caso de este conjunto de la sociedad. Por ello surge *SocialFrame* que intenta resolver estas dificultades y conseguir que estas personas puedan acceder a las redes sociales de una forma sencilla y cómoda.

Gracias a la integración de nuevas tecnologías en los dispositivos móviles como códigos QR, cámara y conexión a Internet es posible desarrollar aplicaciones que nos aportan información de redes sociales y permiten interactuar con ellas. *SocialFrame* utiliza estas tecnologías para que las redes sociales sean algo útil y divertido y no algo engorroso que implique un aprendizaje especial en aquellas personas que no han utilizado nunca una red social o en personas que las han utilizado y aún así prefieren un uso más intuitivo de éstas.

Índice general

Resumen	IX
Índice general	XI
Índice de cuadros	XV
Índice de figuras	XVII
Índice de listados	XXI
Listado de acrónimos	XXIII
Agradecimientos	XXV
1. Introducción	1
1.1. Contexto del proyecto	1
1.2. Descripción del problema	4
1.3. Solución propuesta	4
1.4. Terminología	7
1.5. Justificación del nombre del proyecto	8
1.6. Organización de la memoria	8
2. Objetivos	11
2.1. Objetivo general	11
2.2. Objetivos específicos	11
2.3. Herramientas de trabajo	12
2.3.1. Lenguaje de programación	12
2.3.2. Herramientas hardware	12
2.3.3. Herramientas software	14
2.3.4. Otras herramientas	18

3. Antecedentes	21
3.1. Trabajos relacionados	21
3.1.1. Gestión y difusión de datos por RFID	21
3.1.2. Interfaces de usuario acústicas para tecnologías AAL	22
3.1.3. Inteligencia ambiental	22
3.2. Casos de estudio	24
3.2.1. ElderCare	24
3.2.2. TweetDeck	25
3.2.3. SeniorChannel	27
3.2.4. Messenger Visual	27
3.2.5. Sistema para la visualización de información a través de realidad aumentada en dispositivos móviles	28
3.2.6. aQRdate	29
4. Método de trabajo	31
4.1. Metodología utilizada	31
4.2. Planificación	32
4.3. Análisis de requisitos	33
4.3.1. Iteración 1: Login del usuario	34
4.3.2. Iteración 2: Mostrar Información del usuario	34
4.3.3. Iteración 3: Mensajes privados	34
4.3.4. Iteración 4: Eventos y cumpleaños	34
4.3.5. Iteración 5: Lista de amigos	35
4.3.6. Iteración 6: Comentarios del muro	35
4.3.7. Iteración 7: Fotos	35
4.3.8. Diagrama de casos de uso	35
4.4. Fase de diseño	37
4.4.1. Diagramas de secuencia	37
4.4.2. Arquitectura de la aplicación	44
4.4.3. Patrones de diseño	45
4.4.4. Interfaz de usuario	45
4.4.5. Diagrama de clases	46
4.5. Fase de implementación	50
4.6. Fase de pruebas	55
4.6.1. Pruebas unitarias	55
4.6.2. Pruebas funcionales	56

4.6.3. Pruebas de sistema	58
5. Resultados	59
5.1. Seguimiento del proyecto	59
5.2. Coste del proyecto	60
5.3. Aplicación Android	60
5.3.1. Inicio de la aplicación	60
5.3.2. Login de la aplicación	62
5.3.3. Lector de códigos QR	63
5.3.4. Información del usuario	65
5.3.5. Mensajes privados	66
5.3.6. Eventos y cumpleaños	66
5.3.7. Amigos	68
5.3.8. Comentarios del muro	70
5.3.9. Fotos	71
5.3.10. Créditos	75
5.4. Evaluación	77
6. Conclusiones y propuestas	81
6.1. Conclusiones	81
6.2. Propuestas	82
6.2.1. Mejoras en la aplicación	82
6.2.2. Líneas de investigación futura	83
Bibliografía	85
A. Creación de una aplicación Android en Facebook	91
B. Añadir Google+ a la aplicación	95
C. Cuestionario de Evaluación	97

Índice de cuadros

4.1. Pruebas funcionales	57
5.1. Coste económico del proyecto	60
5.2. Horas totales del proyecto	60
5.3. Características de las personas	77

Índice de figuras

1.1.	Redes móviles con mayor número de usuarios	2
1.2.	Usuarios de Facebook por rangos de edad	3
1.3.	Código QR	4
1.4.	Visión general de la solución propuesta	5
1.5.	Logo de la aplicación SocialFrame	8
2.1.	Geeksphone One	13
2.2.	Samsung Galaxy Tab 7'	13
2.3.	Entorno de desarrollo Eclipse	14
2.4.	Android SDK	15
2.5.	Android Virtual Device Manager	16
2.6.	Emulador Android SDK	16
2.7.	Graph API Explorer	18
3.1.	Sistemas de Cuidado en el Hogar	23
3.2.	Eldercare móvil	25
3.3.	Tweets enviados desde Eldercare	26
3.4.	TweetDeck	27
3.5.	Sistema para la visualización de información a través de realidad aumentada	28
3.6.	aQRdate	29
4.1.	Diagrama de Casos de Uso	36
4.2.	Diagrama de secuencia del login	38
4.3.	Diagrama de secuencia para mostrar información del usuario	38
4.4.	Diagrama de secuencia para mostrar mensajes privados	39
4.5.	Diagrama de secuencia para mostrar eventos y cumpleaños	40
4.6.	Diagrama de secuencia para mostrar la lista de amigos	40
4.7.	Diagrama de secuencia para mostrar los comentarios del muro	41
4.8.	Diagrama de secuencia para escribir un comentario	42

4.9.	Diagrama de secuencia para mostrar las fotos de un amigo	43
4.10.	Diagrama de secuencia para escribir un comentario	44
4.11.	Modelo Vista Controlador	44
4.12.	Diagrama de Clases del paquete controller	48
4.13.	Diagrama de Clases del paquete model	49
5.1.	Diagrama de seguimiento de Gantt	59
5.2.	Pantalla inicial de la aplicación	61
5.3.	Pantalla inicial de la aplicación landscape	61
5.4.	Pantalla inicial sin conexión a internet	62
5.5.	Login de la aplicación	63
5.6.	Login de la aplicación	64
5.7.	Instrucciones de uso del lector	64
5.8.	Lector de códigos QR	65
5.9.	Aviso de cargando información	66
5.10.	Información personal del usuario	67
5.11.	Mensajes privados	67
5.12.	Eventos y cumpleaños	68
5.13.	Información detallada del evento	69
5.14.	Lista de amigos	69
5.15.	Opciones de acceso a información de amigo	70
5.16.	Comentarios del muro	71
5.17.	Subir un comentario (post)	72
5.18.	Ayuda para la intefaz de las fotos	72
5.19.	Fotos de un amigo	73
5.20.	Zoom sobre una foto	74
5.21.	Opciones para subir una foto	74
5.22.	Cámara del dispositivo móvil	75
5.23.	Galería de fotos	76
5.24.	Créditos de la aplicación	76
5.25.	Usabilidad percibida entre Facebook y SocialFrame	78
5.26.	Adecuación al dispositivo entre Facebook y SocialFrame	79
5.27.	Satisfacción final	80
A.1.	Facebook Developers	91
A.2.	Página de creación de aplicación en Facebook	92

A.3. Ingreso de la key hash	93
B.1. Google +	95
C.1. Cuestionario de evaluación	98
C.2. Cuestionario de evaluación	99

Índice de listados

1.1.	Respuesta de la Graph API de Facebook	7
4.1.	Layout de fila de amigos	50
4.2.	Comprobar si hay conexión a Internet	51
4.3.	Request a la API de Facebook para autentificar un usuario	51
4.4.	Tratamiento al valor devuelto por el lector de códigos QR	52
4.5.	Request a la Graph API de Facebook	52
4.6.	Request a través de una consulta FQL	53
4.7.	Actualización de información a través de un observer	54
4.8.	Request a la API para escribir un comentario en Facebook	54
4.9.	Clase padre del patrón adapter	55
4.10.	Test sobre petición de información de usuario	56
4.11.	Test sobre postear un comentario en el muro	56
B.1.	Clase específica para GooglePlus	96
B.2.	Ejemplo de mostrar usuario en Google +	96

Listado de acrónimos

3G	Third Generation
AAL	Ambient Assisting Living
ADB	Android Debug Bridge
ADT	Android Development Tools
API	Application Programming Interface
DDMS	Dalvik Debug Monitor
ECJ	Eclipse Compiler Java
FQL	Facebook Query Language
GPL	General Public License
HCS	HomeCare System
IBM	International Business Machines
IDE	Integrated Development Environment
JDT	Java Development Toolkit
JSON	JavaScript Object Notation
MVC	Model View Controller
OMG	Object Management Group
PFC	Proyecto Fin de Carrera
PUD	Proceso Unificado de Desarrollo
QR	Quick Response
RAM	Random access memory
RFID	Radio Frequency IDentification
SDK	Sofware Development Kit
SMS	Short Message Service
TIC	Tecnologías de la Información y Comunicación
TI	Tecnologías de la Información
UML	Unified Modeling Language

USB	Universal Serial Bus
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WiFi	Wireless Fidelity
XML	Extensible Markup Language

Agradecimientos

Quisiera agraceder a la *Escuela Superior de Informática* por haberme formado como ingeniero. También agraceder a *Ramón Hervás Lucas* por darme la posiblidad de realizar un proyecto fin de carrera que esté basado en un tema social y con una tecnología tan moderna como es *Android*. Por último, agraceder al grupo de investigación *MAmI* por los recursos prestados y su ayuda en todo momento.

Gracias.

Juan Miguel

*A mis padres por haberme dado la oportunidad de estudiar una carrera.
A Carmen por aguantarme y darme ánimos todos los días.*

Capítulo 1

Introducción

EN este capítulo se explican las razones o justificaciones de la realización de este proyecto, para ello se describe el problema encontrado, se propone una solución a dicho problema y, por último se especifica la organización de la memoria.

1.1. Contexto del proyecto

Las redes sociales son ya una realidad en nuestra vida cotidiana, la mayoría de usuarios de la Web utiliza con asiduidad todo tipo de redes sociales: Facebook, Twitter, Google+, Linkedin, etc. Tienen como propósito facilitar la comunicación entre millones de usuarios en el sitio web, además ofrecen relacionarse por medio de temas sociales como puede ser la búsqueda de trabajo o intereses comunes, en el caso de las redes específicas. En estos sitios web, un número inicial de usuarios envían mensajes a miembros de su propia red social invitándoles a unirse al sitio. Los nuevos participantes repiten el proceso, creciendo el número total de miembros y los enlaces de la red. Los sitios ofrecen características como actualización automática de la libreta de direcciones, perfiles visibles, la capacidad de crear nuevos enlaces mediante servicios de presentación y otras maneras de conexión social en línea. Las redes sociales también pueden crearse en torno a las relaciones comerciales.

Su auge se ha producido hace no más de cinco años, en parte gracias al avance de las conexiones a internet y al aumento en la cantidad de personas con acceso a una computadora. Dentro de ese aumento es importante destacar que el uso de dispositivos móviles ha hecho que las redes sociales sean más usadas de lo que ya eran. Los dispositivos móviles ya son prácticamente una computadora de bolsillo capaz de hacer llamadas telefónicas. Es por eso que no sorprende para nada que muchos sitios de Internet se enfoquen mas en redes móviles que en sitios diseñados para navegadores convencionales. Es por eso que seria interesante ver cuales son las redes sociales más accedidas desde el smartphone [2], como se puede ver en la figura 1.1.

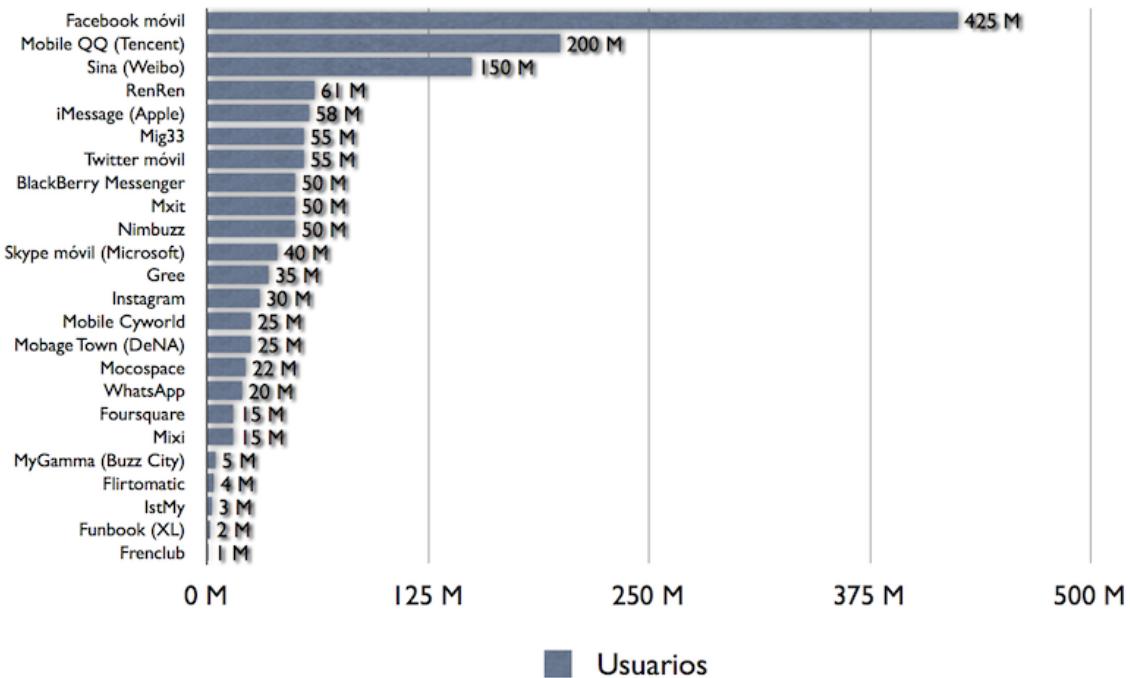


Figura 1.1: Redes móviles con mayor número de usuarios

Como se puede ver en la figura 1.1, la red social que más se utiliza en tecnología móvil es *Facebook*, ésta tiene un enorme número de usuario lo que ha implicado millones de relaciones sociales entre miembros de la red de todo el mundo. Ofrece una serie de servicios donde los más destacados son:

Lista de amigos: en ella, el usuario puede agregar a cualquier persona que conozca y esté registrada, siempre que acepte su invitación.

Muro: el muro es un espacio en cada perfil de usuario que permite que los amigos escriban mensajes para que el usuario los vea. Sólo es visible para usuarios registrados.

Fotos: según Facebook hay 5000 millones de fotos de usuario, es decir, unos 160 terabytes de almacenaje.

Facebook como cualquier red social tiene un inconveniente o desventaja importante y es el reducido número de usuarios de avanzada edad que no la utilizan. El rango de usuarios más proliferante en *Facebook* es de 18-34 años, para ver esta importante diferencia entre el rango de edad de los usuarios [3], se muestra en la figura 1.2. Se puede observar claramente como las personas de 45-64 años no interactúan con la red social *Facebook*, esto es una gran parte

de la sociedad. Estos datos reflejan un tipo de usuario en las redes sociales, por el contrario debería ser cualquier persona que pueda relacionarse con otra siempre cuando el idioma no sea un problema para la comunicación.

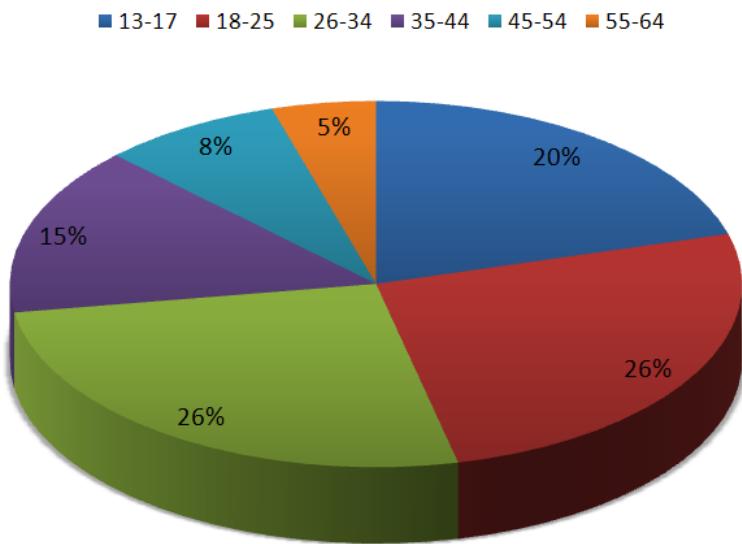


Figura 1.2: Usuarios de Facebook por rangos de edad

Como se ha podido observar existe una necesidad por hacer uso de las redes sociales, llegando a tener más importancia incluso que los e-mails o las llamadas telefónicas. La gran variedad de aplicaciones móviles que integran estas redes sociales ofrece un gran abanico de posibilidades a la hora de relacionarse en la Web. Sin embargo, estas redes sociales parecen enfocarse a un determinado grupo de la sociedad en vez de generalizarse para cualquier tipo de usuario o incluso usuarios que no han utilizado en ningún momento la Web.

Las redes sociales deben ser un medio importante de las personas mayores o dependientes para relacionarse, ya que por diversas razones no pueden hacerlo. Algunas de estas razones son el aislamiento en la vivienda (no pueden salir de casa o recorrer largas distancias) o tener algún tipo de enfermedad que no le permita relacionarse con facilidad con otras personas. Por ello las relaciones virtuales a través de las diferentes redes sociales de Internet pueden ser una solución para este tipo de casos.

1.2. Descripción del problema

La mayoría de aplicaciones existentes tienen un problema a la hora de conseguir que personas dependientes o mayores hagan uso de las redes sociales de una forma continua y progresiva. Esto se debe a que no se muestra la información que realmente necesita el usuario o que es poco intuitiva la interacción con la red social.

Es importante que la información mostrada sea generada de una forma automática y que no implique una búsqueda de ciertos campos en la red social. Este punto de vista permite que los usuarios accedan a cualquier tipo de información dentro de la red social sin tener que realizar complejas interacciones de navegación o de introducción de textos o imágenes.

Además los usuarios necesitan disponer de información específica sobre los temas que realmente les interesan, ya que la gran cantidad de información disponible en la red social puede provocar que los usuarios pierdan interés y por tanto que a la larga abandone la red social.

1.3. Solución propuesta

La solución propuesta al problema descrito en la sección anterior, es la de construir una aplicación que permite acceder a una red social de una forma rápida, sencilla e intuitiva para personas dependientes o mayores. Para ello es necesario la utilización de códigos QR[4](vease Figura 4.4), estos nos van a permitir una interacción a través de metáforas en el entorno de la persona dependiente o mayor, de tal forma que le sea cómodo acceder a información de esa red en particular.



Figura 1.3: Código QR

La idea principal de esta solución es acomodar objetos de la vida diaria del usuario a elementos en la red social (realizar metáforas), por ejemplo: cualquier pared de una habitación puede estar asociado a el muro de una red social. Además como se ha mencionado antes se muestra la información específica y necesaria para que el usuario interactúe de una forma lo más adecuada posible. Para entender mejor la idea principal se propone una visión general que se puede ver en la figura 1.4. Como se observa en la visión general mostrada una persona mayor o dependiente utiliza un dispositivo móvil para capturar una *metáfora*, en este caso el código QR representa el muro de comentarios de Facebook, a continuación se realiza una petición a la red social y esta devuelve la información requerida al dispositivo móvil, que es mostrada al usuario (persona dependiente o mayor).

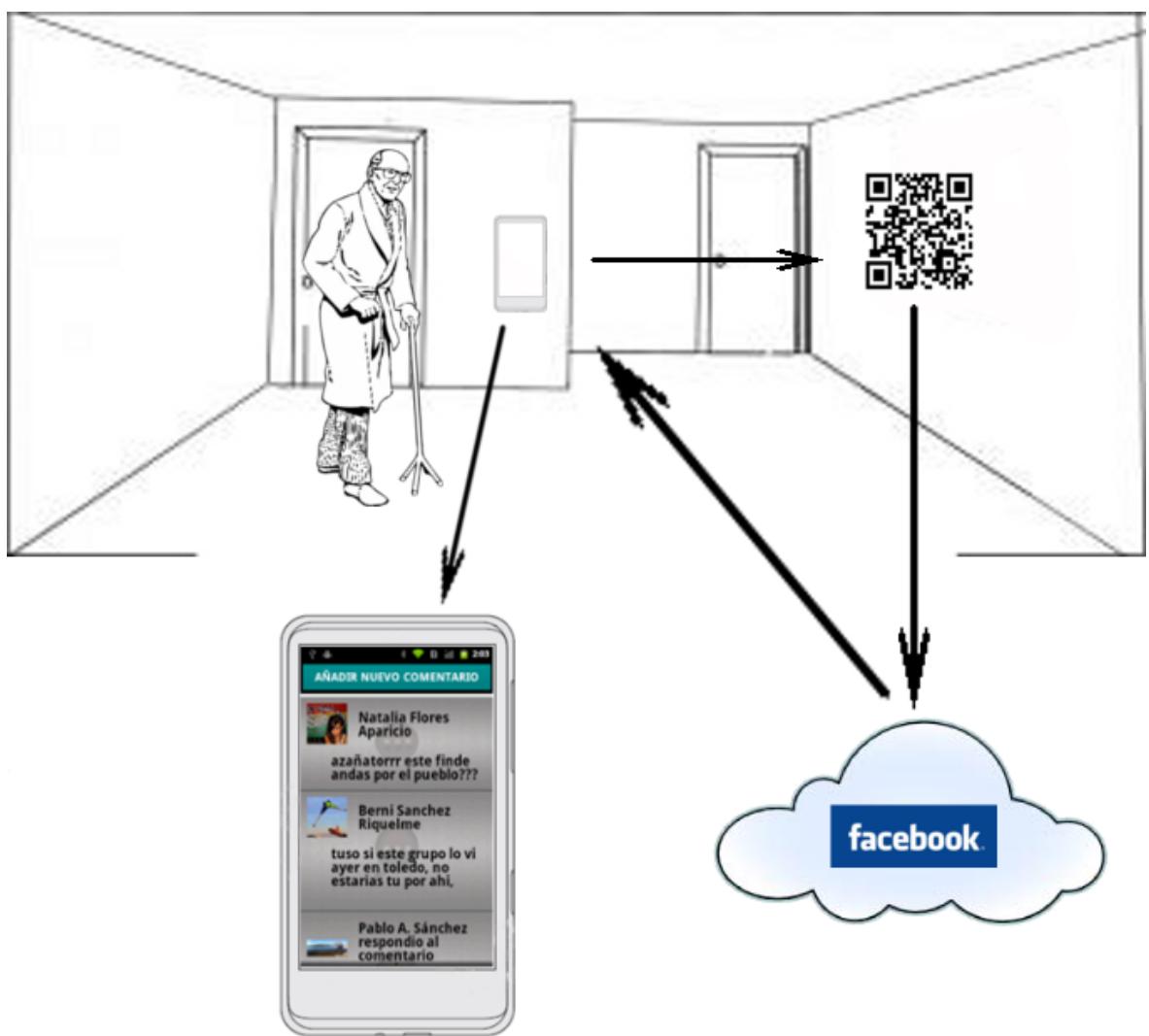


Figura 1.4: Visión general de la solución propuesta

La *metáfora* se puede entender como una correspondencia entre un objeto de la vida cotidiana del usuario con un elemento de la red social. También se ha incluido referencias entre elementos y colores, de tal forma que el usuario pueda realizar un ejercicio de memoria. Las *metáforas* utilizadas en la solución son las siguientes:

- *Datos personales*: código QR en un carnet, una tarjeta o una ficha personal. También puede ser en un objeto de color azul.
- *Mensajes privados*: código QR en una carta, una postal o cualquier mensaje. También puede ser en un objeto de color amarillo.
- *Eventos y cumpleaños*: código QR en un calendario, despertador o cualquier objeto con fecha. También puede ser en un objeto de color blanco.
- *Amigos*: código QR en una agenda, una libreta de direcciones o en un teléfono. También puede ser en un objeto de color verde.
- *Muro de comentarios*: código QR en un muro, una pared o un tablón de anuncios. También puede ser en un objeto de color gris.
- *Fotos*: código QR en un álbum, una foto o un marco de foto. También puede ser un objeto de color negro.

Las comunicaciones con la red social Facebook hacen uso de la *Graph API*, ésta publica diferentes recursos con los que se pueden realizar ciertas operaciones HTTP (GET, POST, DELETE). La forma de conocer la URI (Uniform Resource Identifier) de los recursos publicados es realizar una consulta sobre la raíz del servidor externo para conocer qué recursos publica.

El núcleo de Facebook es un gráfico social, donde la gente y las conexiones de importancia se muestran. La Graph API presenta una visión consistente del gráfico social de Facebook, de manera uniforme representa los objetos en el gráfico (por ejemplo, la gente, fotos, eventos y páginas) y las conexiones entre ellos (por ejemplo, las relaciones de amistad, de contenido compartido, y las etiquetas de fotos).

Cada objeto en el gráfico social tiene un identificador único. Puede acceder a las propiedades de un objeto mediante la solicitud de [https://graph.facebook.com/*ID*](https://graph.facebook.com>ID). Por ejemplo, la página oficial de la plataforma de Facebook tiene el id “19292868552”, por lo que se puede re-

cuperar el objeto JSON (JavaScript Object Notation) en <https://graph.facebook.com/19292868552> (véase el listado 1.1). Todas las respuestas de la Graph API son del tipo JSON.

```
{  
    "name": "Facebook Platform",  
    "website": "http://developers.facebook.com",  
    "username": "platform",  
    "founded": "May 2007",  
    "company_overview": "Facebook Platform enables anyone to build...",  
    "mission": "To make the web more open and social.",  
    "products": "Facebook Application Programming Interface (API)...",  
    "likes": 449921,  
    "id": 19292868552,  
    "category": "Technology"  
}
```

Listado 1.1: Respuesta de la Graph API de Facebook

1.4. Terminología

Algunos conceptos importantes para comprender mejor el proyecto son los siguientes:

Android [6] Es un sistema operativo basado en GNU. Fue diseñado para dispositivos móviles, primeramente para teléfonos inteligentes y posteriormente para tablets, pero actualmente existe una gran variedad de dispositivos de otros tipos que lo incluyen, como netbooks, ebooks o televisores. Fue desarrollado por una empresa llamada Android Inc. que fue comprada por Google en 2005 y actualmente es el principal producto de la Open Handset Alliance.

Código QR Quick response barcode (código de barras de respuesta rápida) es un sistema para almacenar información en una matriz de puntos o un código de barras bidimensional creado por la compañía japonesa Denso Wave, subsidiaria de Toyota, en 1994. Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector. Los códigos QR también pueden leerse desde PC, smartphone o tablet mediante dispositivos de captura de imagen, como puede ser un escáner o la cámara de fotos, programas que lean los datos QR y una conexión a Internet para las direcciones web.

Facebook Es una empresa creada por Mark Zuckerberg y fundada por Eduardo Saverin, Chris Hughes, Dustin Moskovitz y Mark Zuckerberg consistente en un sitio web de redes sociales. Originalmente era un sitio para estudiantes de la Universidad de Harvard, pero actualmente está abierto a cualquier persona que tenga una cuenta de correo

electrónico. Los usuarios pueden participar en una o más redes sociales, en relación con su situación académica, su lugar de trabajo o región geográfica.

1.5. Justificación del nombre del proyecto

Es necesario que el nombre sea fácil de recordar y se ha considerado que no haga referencia a ninguna red social en particular. Aparece como una combinación de palabras que hacen referencia a lo que la aplicación representa: *Social* referente a cualquier red social o relación social, *Frame* referente a la forma de representar la aplicación en marcos o cuadros que muestran la información en forma de secuencia. De esa forma surge el nombre de la aplicación *SocialFrame*, además el logo de creación propia es una combinación de las siglas del nombre, un fondo que representa un código QR y un pequeño ícono que es un búho verde que al igual que el animal hace referencia a ver en la oscuridad con facilidad. Esto quiere decir que la aplicación se asemeja a visionar algo complejo de una manera sencilla simplemente apuntando al objetivo adecuado. El logo de la aplicación se puede observar en la imagen 1.5.



Figura 1.5: Logo de la aplicación SocialFrame

1.6. Organización de la memoria

Este documento se organiza siguiendo las directrices marcadas por la normativa para el Proyecto Fin de Carrera (Proyecto Fin de Carrera (PFC)),[5]. La estructura es la siguiente:

Capítulo 1: Introducción Se trata de este capítulo de la documentación, donde se realiza una introducción de la idea del proyecto así como de los conceptos importantes del mismo.

Capítulo 2: Objetivos En esta parte de la documentación se explican los objetivos con los que se inició el desarrollo del PFC, detallando cada uno de una forma clara y sencilla. Se exponen tanto los objetivos generales de la aplicación como aquellos que son más específicos.

Capítulo 3: Antecedentes En esta parte se exponen artículos y casos de estudio, comerciales y/o de investigación, que se han estudiado para saber cual es el estado del arte respecto a la aplicación a desarrollar.

Capítulo 4: Método de trabajo Esta parte de la documentación se describe la metodología de desarrollo utilizada en la construcción de la aplicación. Inicialmente se explica la metodología y posteriormente se muestran las distintas partes del proyecto en función de ésta.

Capítulo 5: Resultados En este apartado del documento se encuentra el seguimiento del proyecto, el resultado final de la aplicación detallando cada una de las funciones de ésta y unas estadísticas evaluadoras sobre la satisfacción del usuario final.

Capítulo 6: Conclusiones y propuestas En este capítulo de la documentación se tratan las conclusiones obtenidas tras la realización del proyecto y las propuestas o líneas futuras para posibles mejoras de la aplicación.

Anexos: En los anexos se incluye un ejemplo de añadir la red social Google+, como registrar un aplicación en Android y un cuestionario de evaluación.

Capítulo 2

Objetivos

En este capítulo se explican los objetivos y las herramientas para el desarrollo del proyecto. Los objetivos están divididos en un objetivo general y varios específicos. Además se describen las herramientas hardware y software empleadas durante la elaboración de *SocialFrame*.

2.1. Objetivo general

El objetivo general del proyecto es el desarrollo de una aplicación móvil para acceder a redes sociales destinada a personas dependientes o mayores. La aplicación utiliza interfaces amigables y fáciles de entender.

2.2. Objetivos específicos

Se plantean una serie de objetivos específicos en los que se centra el trabajo a lo largo del proyecto:

Análisis de las dificultades Se estudian y analizan aquellas desventajas que presentan las personas mayores o dependientes en acceder y entender el uso de las redes sociales.

Metáforas para la red social Se proponen una serie de metáforas que consiga una relación directa entre un objeto de la vida cotidiana del usuario con uno o varios elementos de una red social. Además se realizan metáforas a referencias cromáticas del objeto o del elemento en la red social para que la persona ejercite la memoria y asocie dicho color a uno varios elementos en la red social. Estas metáforas se representan con los códigos QR que se generan.

Desarrollo de una aplicación móvil Una aplicación móvil posee una serie de características muy diferentes a las que podría tener si fuera una aplicación de escritorio o una aplicación web. Este será el medio por el que los usuarios accederán al sistema, per-

mitiéndoles realizar diversas operaciones. Se realiza a través de servicios web que permiten en cualquier instante acceder a esos datos, siempre que haya conexión a internet.

Desarrollo de una interfaz móvil Inicialmente se tienen en cuenta un análisis y diseño de interfaces amigables e intuitivas para mejorar la interacción de personas mayores a la aplicación. Se realiza una interfaz invisible y sencilla para la aplicación móvil. La interacción es llevada a cabo por el uso de códigos QR y metáforas que representan objetos cotidianos y su correspondencia con los posibles elementos de la red social. Además se diseñan técnicas de interacción para evitar complejos sistemas de navegación y simplificar la introducción de información y configuración de aspectos personales del usuario.

Integración de servicios del dispositivo móvil Se integran servicios del móvil que son útiles para la aplicación, entre ellos se encuentran: la *cámara* que permite hacer uso del lector de códigos QR y sacar fotos, el *lector de códigos QR* que permite obtener el contenido de dichos códigos, el *reconocedor de voz* que permite transcribir el audio en texto y la *conexión a internet*, ya sea por 3G o por WiFi que nos da acceso a la Web y a las redes sociales.

Acceso a información almacenada en redes sociales Esta extracción se realiza de forma automática accediendo a los datos existentes en dichas redes sociales llamando a las API (Application Programming Interface) que proporcionan. Se obtiene aquella información que sea realmente relevante para un uso sencillo de la red social.

2.3. Herramientas de trabajo

2.3.1. Lenguaje de programación

Java [7] es un lenguaje de programación que fue desarrollado por Sun Microsystems a principios de los años 90, empresa ahora propiedad de Oracle. La sintaxis de Java es similar a la que usan los lenguajes C y C++ pero tiene un modelo de objetos más simple y elimina diversas herramientas de bajo nivel que pueden provocar errores fácilmente. Java es un lenguaje orientado a objetos y multiplataforma. La máquina virtual de Java tiene licencia GPL (General Public License).

2.3.2. Herramientas hardware

El desarrollo del proyecto se ha llevado a cabo principalmente en un portátil HP Compaq nx6310 con sistema operativo Windows XP y GNU/Ubuntu, procesador Intel Celeron a 1.46

Ghz y 1 Gb de memoria RAM. Además se han utilizado varios dispositivos móviles para la ejecución de la aplicación. Estos dispositivos son los siguientes:

- *Geeksphone One* que posee conectividad WiFi y una versión android 2.3. Se puede ver en la figura 2.1.
- *Samsung Galaxy Tab 7'* que posee conectividad WiFi y una versión android 2.2. Se pueve ver el dispositivo en la figura 2.2.



Figura 2.1: Geeksphone One



Figura 2.2: Samsung Galaxy Tab 7'

2.3.3. Herramientas software

Se han usado los siguientes medios software:

Eclipse [17] Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar el proyecto. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE (Integrated Development Environment) de Java llamado Java + Development Toolkit (JDT) y el compilador (ECJ, Eclipse Compiler Java) que se entrega como parte de Eclipse. Permite técnicas avanzadas de refactorización y de análisis de código. Eclipse soporta el desarrollo de software en una gran cantidad de lenguajes de programación, por ejemplo Java, Python, C/C++, PHP o Ruby. Cuenta con un completo sistema de plugins que permite extender su funcionalidad de diversas formas. En la figura 2.3 se observa este entorno de desarrollo.

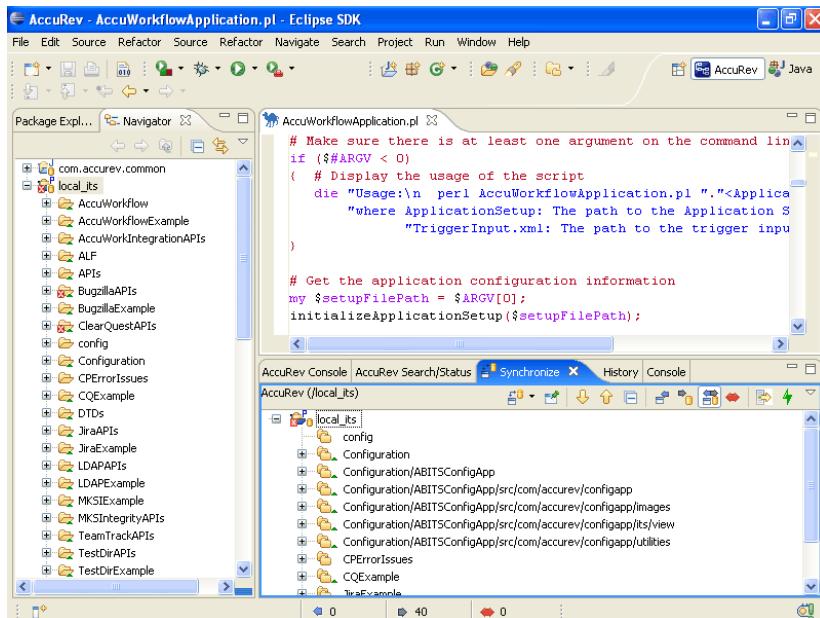


Figura 2.3: Entorno de desarrollo Eclipse

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Eclipse fue liberado originalmente bajo la Common Public License, pero después fue relicenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son de software libre, pero son incompatibles con la licencia pública general de GNU.

Android SDK El SDK (Software Development Kit [8]) de Android incluye diferentes herramientas necesarias para desarrollar software para dispositivos móviles Android. Tiene una arquitectura basada en add-ons o añadidos. Esto permite incluir actualizaciones de las API, toolkits de los diferentes fabricantes de dispositivos para la conexión por USB, plugins de compatibilidad hacia atrás o bibliotecas para el desarrollo de ciertas funciones. Las versiones de las API disponibles para instalar incluyen una amplia colección de ejemplos de código y una extensa documentación. Se puede ver en la figura 2.4 como es el Android SDK.

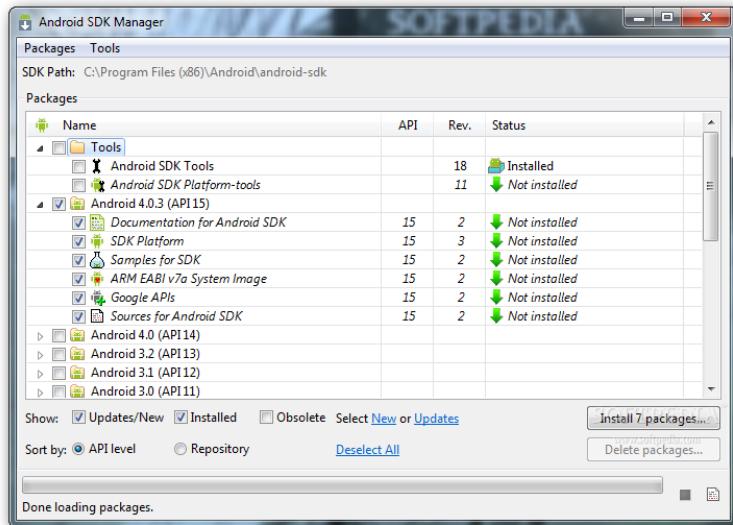


Figura 2.4: Android SDK

Las herramientas más importantes que contiene son las siguientes:

- Android SDK and Virtual Device Manager que permite administrar los diferentes dispositivos móviles reales y virtuales. Esta herramienta se puede observar en la figura 2.5.
- Un emulador que permite ejecutar las aplicaciones desarrolladas en un entorno similar al que se ejecutarán en el dispositivo móvil. El emulador incluye la máquina virtual Dalvik que es la encargada ejecutar las aplicaciones. Un ejemplo de emulador en android se puede ver en la figura 2.6.
- Herramientas de depuración y diagnóstico entre las que se incluyen las herramientas DDMS (Dalvik Debug Monitor) o ADB (Android Debug Bridge). El Dalvik Debug Monitor nos permite sacar las capturas de pantalla de la aplicación.

- Herramientas de empaquetado. Permiten empaquetar aplicaciones en paquetes APK. También realiza un firmado de la aplicación con una clave como paso previo a la distribución del software. Los archivos APK son los necesarios para poder instalar la aplicación en el dispositivo móvil.

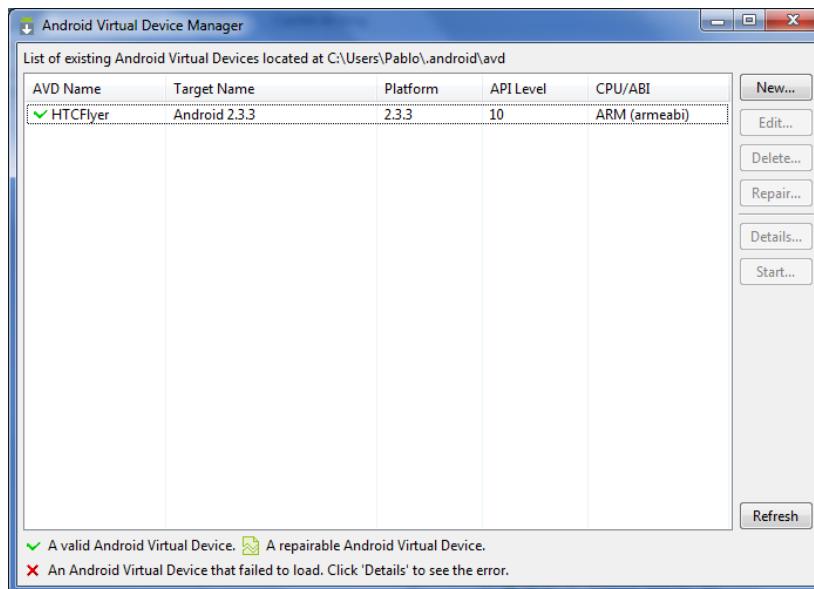


Figura 2.5: Android Virtual Device Manager

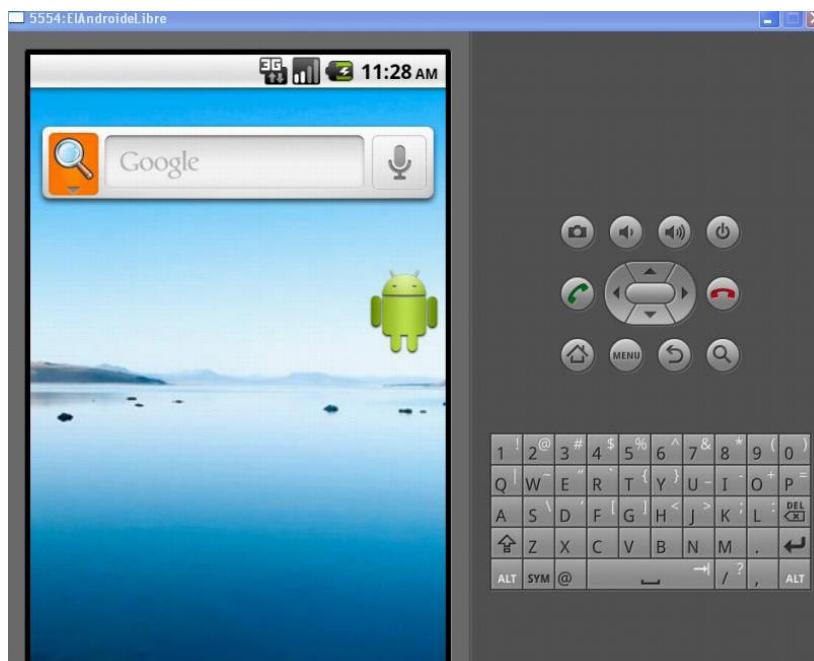


Figura 2.6: Emulador Android SDK

Android Development Tools (ADT) Es un plugin [9] para el entorno de desarrollo Eclipse que permite el desarrollo de aplicaciones Android. Las principales características que añade al entorno son las siguientes: editor gráfico de interfaces de usuario definidas por ficheros XML [18], integración del entorno con las diferentes herramientas que proporciona el SDK de Android y permite diferentes operaciones sobre proyectos Android.

Facebook Android SDK Esta biblioteca de Java de código abierto permite integrar Facebook en su aplicación para Android. Salvo que se indique lo contrario, el Facebook Android SDK [10] está licenciado bajo la Apache License, Version 2.0. Tiene las siguientes características:

- Una autentificación segura a través de OAuth 2.0 [12].
- Realizar peticiones a la Graph API [11].
- Permite el uso del lenguaje Facebook Query Language (FQL) [13], que es un lenguaje de consulta para las bases de datos de Facebook.
- Permite el uso de *dialogs* que muestran páginas oficiales de Facebook aplicadas a dispositivos móviles.

Graph API Explorer Herramienta que proporciona la plataforma Facebook que hace más fácil empezar con la Graph API y construir de forma más segura la aplicación, como se puede ver en la figura 2.7. Sus principales características son las siguientes:

- Realiza peticiones a la Graph API y ver los resultados en línea.
- Explora las conexiones para cada objeto y observa las descripciones de cada campo para entender lo que se devuelve en la respuesta.
- Fácil de obtención de un *acces token* con los permisos específicos necesarios para acceder a los datos que necesita su aplicación.
- Sencilla utilización entre HTTP GET, POST y DELETE para obtener, crear, actualizar o eliminar los objetos.
- Moverse entre los objetos de la Graph API, por ejemplo, haciendo clic en el

identificador del objeto de evento muestra los detalles de dicho objeto.

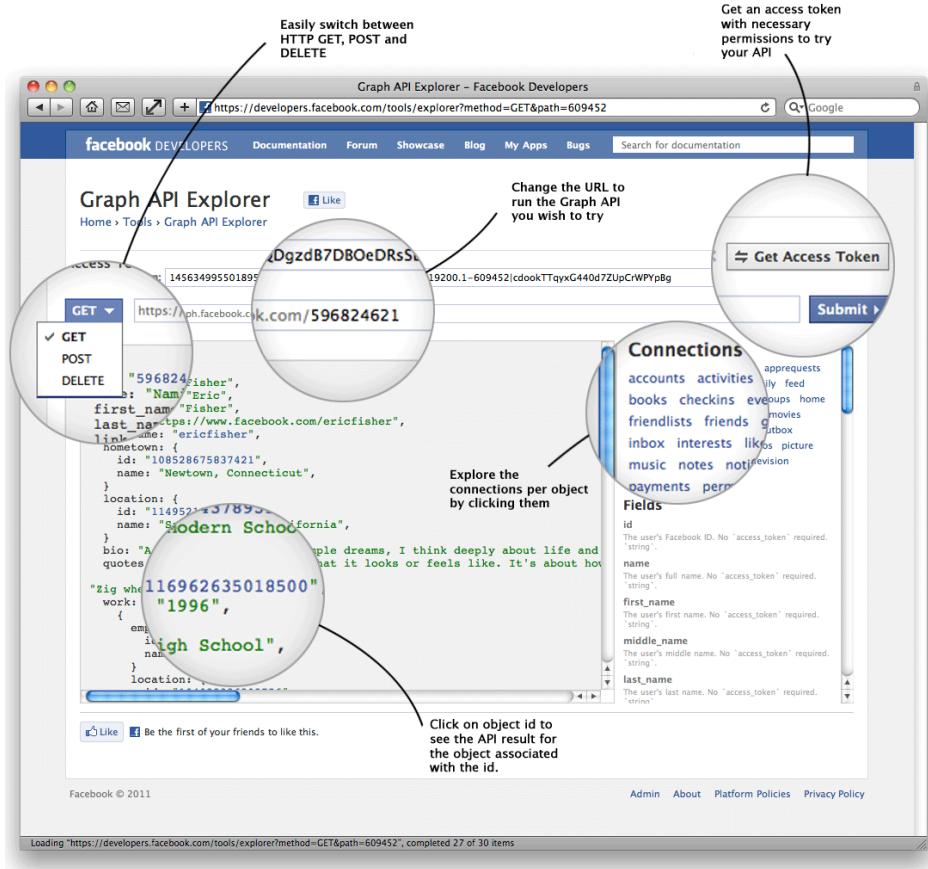


Figura 2.7: Graph API Explorer

2.3.4. Otras herramientas

- Para el desarrollo de las pruebas unitarias de la aplicación móvil se ha usado el framework JUnit [14].
- Se utiliza la librería de código abierto ZXing [15] que permite trabajar con códigos QR si previamente hay instalado en el dispositivo un lector de códigos.
- Se hace la búsqueda por voz de Google (reconocedor de voz).
- Se utiliza el sistema de control de versiones Git [19] para mantener el historial de cambios del proyecto. El proyecto se ha guardado en el repositorio libre GitHub [20].
- Para la realización del diseño del proyecto se ha utilizado el lenguaje de modelado Unified Modeling Language (UML) [16] y el software Visual Paradigm [22].

- Se utiliza el Microsoft Project [23] para el seguimiento de los plazos de tiempo del proyecto, además de generar los diagrams de Gantt.
- Para la edición de iconos e imágenes de la aplicación se usa la herramienta GIMP [21].
- Para la realización de esta memoria se ha usado el procesador de texto L^AT_EX junto con el editor de textos Emacs.

Capítulo 3

Antecedentes

EN este capítulo se realiza una revisión del estado del arte actual relacionado con la aplicación *SocialFrame*. Se exponen trabajos relacionados y casos de estudio, ya sean comerciales o no.

3.1. Trabajos relacionados

En la actualidad la AAL congrega numerosos eventos donde se presentan artículos o trabajos que intentan mejorar la vida cotidiana de las personas mayores o dependientes, entre los cuales, podemos relacionar con este proyecto: la gestión y difusión de datos a través de etiquetas RFID [24], interfaces de usuario acústicas para tecnologías AAL [25] e inteligencia ambiental que permite a las personas mayores utilizar interfaces de usuario modernas o futuras [26].

3.1.1. Gestión y difusión de datos por RFID

La tecnología RFID está demostrando que es un factor clave en el Internet de objetos, ya que es una solución de bajo coste para identificar objetos o incluso codificar una URL desde la que un objeto representa el servicio o los datos a los que se puede acceder fácilmente. El uso de etiquetas RFID para codificar un enlace o una red remota que permite acceder a los datos del repositorio o servicio, es el método más utilizado ya que hace más inmediato el acceso a los metadatos en vez de tocar o señalar el objeto. Por otra parte, no impone la existencia de un enlace de datos permanente back-end (puro enfoque de datos en la red), que puede no ser deseable o posible en ciertas situaciones. Aún así, no impide la adaptación de un enfoque híbrido que en ocasiones tiene acceso a un sistema de back-end con el fin de sincronizar o enriquecer los datos almacenados en la etiqueta con información adicional.

Cada vez más, las aplicaciones de RFID están tratando de incorporar los datos personalizados directamente sobre las etiquetas, lo que elimina la necesidad de utilizar el valor de la etiqueta simplemente como un identificador único para buscar información adicional en un sistema de base de datos. Por ejemplo, una empresa que desee almacenar los datos tales como fecha de caducidad, el fabricante original, un cheque, o cualquier otro dato relevan-

te directamente en la etiqueta RFID de manera que esta valiosa información esté siempre disponible. De hecho, muy pocos sistemas almacenan sólo el identificador o la URL en una etiqueta.

3.1.2. Interfaces de usuario acústicas para tecnologías AAL

Las ayudas tecnológicas son capaces de apoyar a las personas mayores y, por ello, permitir que estas personas permanezcan más tiempo en sus propios hogares de una manera independiente. En particular, las tecnologías de información y comunicación (TIC) son de gran importancia para ayudar a las personas mayores. Sin embargo, puesto que tales sistemas generalmente son complejos de usar, el diseño de interfaces simples e intuitiva es un aspecto importante. Se ha demostrado que las formas naturales y convenientes para interactuar con los sistemas es lo más adecuado. Una manera de realizar una intuitiva interacción hombre-máquina es utilizando el habla.

El habla es la forma más natural de intercambio de información entre los seres humanos y, por tanto, una persona física puede interactuar con los sistemas tecnológicos. Especialmente sirve para que las personas mayores tengan un control total de esos sistemas por comandos de voz de una manera fácil y sencilla. Sin embargo, el diseño de aplicaciones exitosas de TI inteligentes basado en la asistencia a personas mayores no requiere soluciones para cumplir objetivos puramente tecnológicos, sino también un examen continuo de las necesidades del usuario, aceptación de los usuarios, la integración en las estructuras de atención médica, así como la consideración de los aspectos económicos.

Esta contribución ha debatido el uso de interfaces de usuario acústicas que se centran en el reconocimiento automático del habla para tecnologías ambientales de la vida cotidiana. Un marco de interfaz de usuario acústica fue el control de una aplicación de calendario que formaba parte de una actividad personal y del hogar para ayudar a las personas mayores. El desarrollo del sistema se basa completamente en un enfoque de diseño centrado en el usuario. La entrada acústica y la salida de un sistema de ambiente se demuestra que es muy importante y una estrategia favorable para la interacción con ese sistema. La mayoría de los participantes prefirieron el habla natural como medio de entrada y de salida para una aplicación independiente de la edad, el sexo y la experiencia tecnológica.

3.1.3. Inteligencia ambiental

Los sistemas que ofrecen el servicio de una manera sensible y receptiva son integrados en nuestro entorno cotidiano y se hace referencia a ellos como inteligencia ambiental. La inteligencia ambiental ha sido reconocida como un enfoque prometedor para hacer frente a los problemas mencionados en el dominio de la vida asistida. Los sistemas de asistencia que se centran en el apoyo de las personas con necesidades especiales (ancianos, discapacitados) en sus propias casas se llaman Sistemas de Cuidado en el Hogar (HCS).

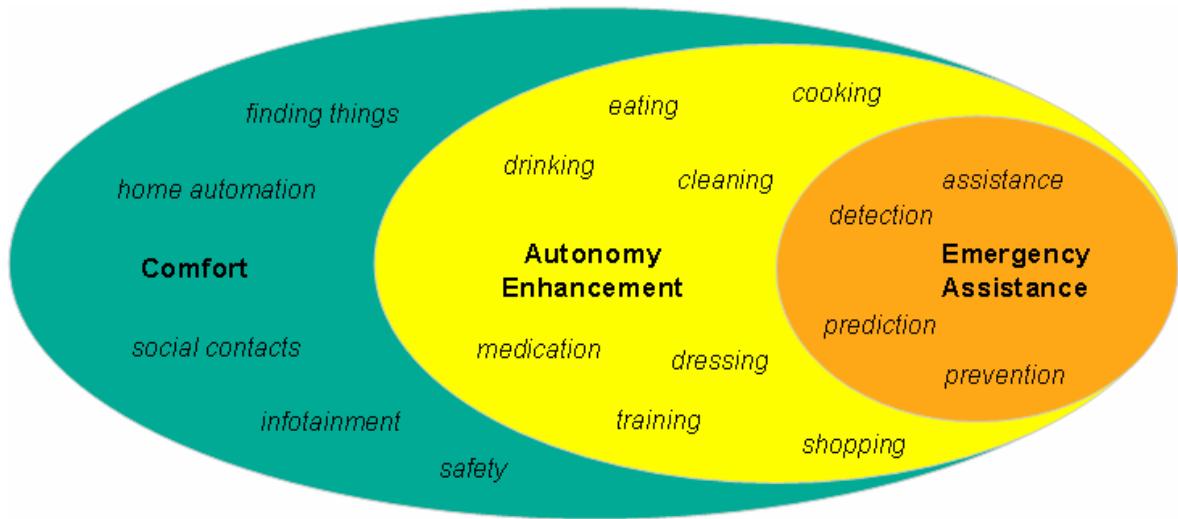


Figura 3.1: Sistemas de Cuidado en el Hogar

El objetivo de los HCS es permitir que las personas dependientes puedan vivir mejor en el ambiente del hogar conservando su independencia, aunque tengan discapacidades o enfermedades médicas. Como se ilustra en la figura 3.1 el dominio de los HCS se estructura en servicios de asistencia de emergencia, servicios de autonomía de mejora y servicios de confort. Estos sistemas tienen las siguientes características:

- Invisible, es decir, integrada en la ropa, relojes, gafas, etc.
- Móvil, es decir, llevado a su alrededor.
- Espontánea comunicación entre los nodos (ad hoc).
- Heterogénea y jerárquica, es decir, que comprenden diferentes tipos de nodos del sistema en cuanto a su potencia de cálculo y la funcionalidad prestada.
- Sensible al contexto, es decir, son conscientes de su entorno local y de forma espontánea intercambian información con los nodos a su alrededor.
- Anticipada, es decir, actuando en su propio nombre, sin peticiones explícitas extrínsecas.
- Comunicación natural con los usuarios a través de voz y gestos en lugar de teclado, ratón o introducción de textos.

- Interacción natural con los usuarios por medio de dispositivos que se utilizan para, por ejemplo, la ropa, relojes, TV, teléfono, electrodomésticos. Para este fin, los dispositivos serán equipados con algún tipo de inteligencia.
- Adaptativa, es decir, capaz de reaccionar a todas las situaciones anormales y excepcionales en una manera flexible.

Con las características mencionadas anteriormente, los sistemas HCS parecen tener potencial interesante para aplicaciones orientadas a la vida cotidiana. Debido a su transparencia, esta tecnología ofrece nuevas formas de detección sensible al contexto.

3.2. Casos de estudio

Debido al gran auge de las redes sociales en Internet, existe una gran cantidad de aplicaciones dedicadas a facilitar la interacción con ellas. Sin embargo, las aplicaciones dedicadas a personas dependientes o mayores relacionadas con las redes sociales escasean. Algunas de estas aplicaciones son ElderCare [24], TweetDeck [28], SeniorChannel [29], Messenger Visual [30], Sistema para la visualización de información a través de realidad aumentada en dispositivos móviles [27] y aQRdate [31].

3.2.1. ElderCare

ElderCare soporta diferentes mecanismos de notificación, posee una base de aplicaciones de Internet en tiempo real para mantener (por ejemplo, correo electrónico, SMS, RSS o Twitter) la evolución del usuario. En otras palabras, la plataforma ElderCare no solo mantiene los datos personalizados para mejorar las actividades de cuidado diario y reducir los errores en un centro de atención, sino que también exporta parte de esos datos para servicios externos de Internet en tiempo real para que los familiares y amigos pueden seguir la vida del usuario. En el lado del cliente, se utiliza una aplicación Java ME (observe la Figura 3.2), que permite a los usuarios móviles de una manera fácil añadir nueva información a través de la pulsera del residente, es una etiqueta RFID, y extraerlos con el propósito de revisarlos y, opcionalmente, transferirlos en el centro de atención.

ElderCare es muy sensible a la privacidad de muchos de los registros de cuidados en una residencia. Por esta razón, las entradas de registro de atención se filtran teniendo en cuenta las preocupaciones sobre la privacidad y luego se envían a los familiares y amigos de un residente determinado a través de diferentes mecanismos de notificación. Por ejemplo, un mensaje se hace para su página de Twitter de un residente o su muro de Facebook, un SMS se envía a familiares y amigos de un residente, o un correo electrónico con un informe diario



Figura 3.2: Eldercare móvil

de un residente se envía a sus familiares y amigos autorizados.

A modo de ejemplo, la figura 3.3 muestra los tweets enviados automáticamente por un residente. Observe que el tweet de creación de marcas de tiempo, ubicación y nombre de usuario se inserta implícitamente, ya que cada procedimiento de cuidado o atención de la anotación de eventos en una pulsera de residente es calificado por los atributos de contexto de cuándo, dónde y quién la realizó. Por otra parte, el proceso de sincronización se produce cada vez que el personal de atención inicia Bluetooth o sesión de 3G para subir los datos de los registros de los usuarios en el centro de atención.

3.2.2. TweetDeck

TweetDeck es una aplicación de escritorio desarrollada en Adobe AIR para Twitter, Facebook, LinkedIn, Google Buzz, Foursquare, y MySpace. Como otras aplicaciones para Twitter, interactúa con la API de Twitter para permitir a los usuarios ver y enviar tweets y consultar sus perfiles. Es la aplicación para Twitter más popular, con una cuota de mercado de un 19 % a junio de 2009, sólo por detrás de la web oficial de Twitter con un 45,70 % de mercado a la hora de *twittear*. Es compatible con diversos sistemas operativos incluyendo Microsoft Windows, Mac OS X y Linux. Una versión para iPhone fue lanzada el 19 de junio de 2009 y una para iPad fue lanzada en mayo de 2010. Una versión para Android se hizo pública después de un período abierto en fase beta.

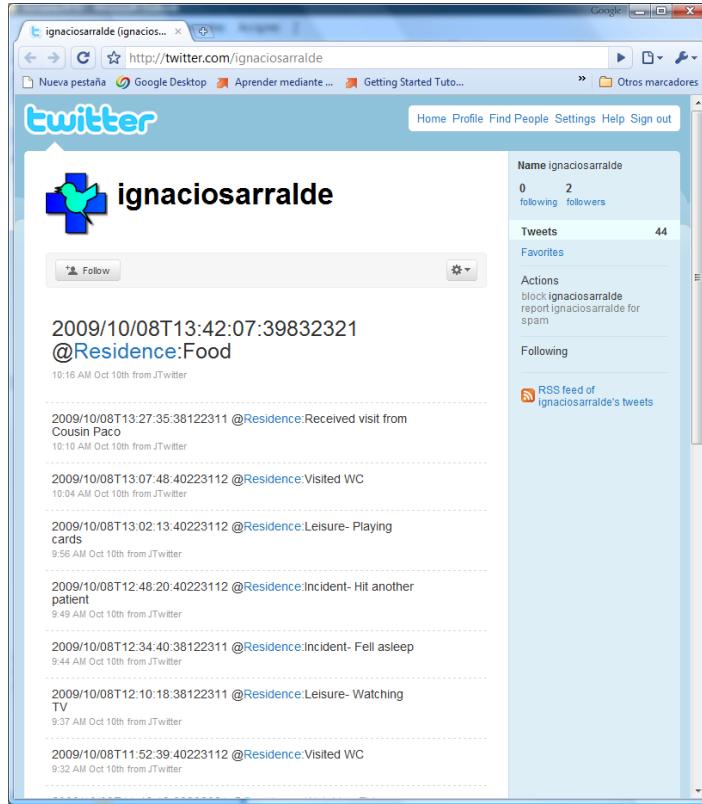


Figura 3.3: Tweets enviados desde Eldercare

Los usuarios pueden dividir la interfaz en columnas que muestran diferentes apartados, por ejemplo, tweets de amigos, como se puede ver en la figura 3.4. TweetDeck interactúa con Twitscoop, 12seconds y Stocktwits, todas ellos pudiendo aparecer en diferentes columnas. También permite dividir a diversos usuarios en grupos, una función útil para muchos usuarios. El programa permite acortar URLs desde su propia interfaz haciendo un clic sobre el enlace.

Originalmente, TweetDeck fue dirigido hacia Twitter. El 16 de marzo de 2009, una versión prelanzamiento fue liberada incluyendo las actualizaciones de los estados de Facebook. El 8 de abril de 2009, las actualizaciones de Facebook forman parte del programa estándar. A la llegada de la versión 0.30 se añade la integración con MySpace. Esto incluye la capacidad de que los usuarios compartan sus estados de ánimo, una característica única de MySpace. Además, la versión 0.32, lanzada el 30 de noviembre de 2009, añade la integración con LinkedIn, como nuevas prestaciones de Twitter. En mayo de 2010 TweetDeck añadió la integración con Google Buzz y Foursquare.



Figura 3.4: TweetDeck

3.2.3. SeniorChannel

El proyecto SeniorChannel es desarrollado en la segunda convocatoria del programa Ambient Assisted Living y con un presupuesto de casi 5 millones de euros que proporcionó el AAL Joint Programme. Es un proyecto de 3 años liderado por Indra y con empresas subcontratadas de 5 países europeos.

El objetivo principal es la calidad de vida en las personas mayores en nuestras comunidades para proporcionarles una serie de redes avanzadas que faciliten su interacción con las nuevas tecnologías. Básicamente es un canal de televisión interactivo en Internet que proporciona a las personas de edad avanzada un método de interacción con el que tener acceso a actividades en su comunidad, capacidad de participar en los debates de actualidad, servicios de entrenamiento y grupos de discusión independientemente de su ubicación geográfica.

La tecnología SeniorChannel facilita la interacción entre las personas de edad avanzada, así como para personas cercanas a su entorno, como por ejemplo, la familia y los supervisores de atención.

3.2.4. Messenger Visual

Messenger Visual es un servicio de mensajería instantánea que permite a las personas con discapacidad cognitiva comunicarse a través de Internet. El principal aspecto de este servicio es que se sustituyan los mensajes de texto por mensajes basados en pictogramas, signo que representa esquemáticamente un símbolo, objeto real o figura, además se proporciona la

funcionalidad básica de un servicio de mensajería instantánea. En este sentido, la interfaz de usuario contiene una serie de funciones básicas, tales como control de acceso o login, gestión de contactos y conversaciones de chat, todas ellas mostradas de una forma clara y fácil de entender.

3.2.5. Sistema para la visualización de información a través de realidad aumentada en dispositivos móviles

Este proyecto proporciona información al usuario sobre la disposición de determinados elementos dentro de un espacio físico en función de la situación exacta del individuo y la dirección en la que está mirando. Así pues, se intenta ayudar a las personas que visitan por primera vez un hospital, museo, parque o cualquier otro tipo de área guiándoles por las diferentes zonas que lo componen y mostrándoles la ubicación de determinados puntos de interés.

Cuando el usuario aproxime el dispositivo móvil a una etiqueta, el adaptador NFC obtiene los datos de la misma para a continuación transmitirlos a la aplicación. La información almacenada en la etiqueta NFC debe incluir, de forma estructurada, los datos necesarios para representar los elementos que hay en el entorno y la distancia respecto a la localización del usuario, es decir, su posición en relación a la perspectiva del dispositivo móvil.

El dispositivo móvil muestra por pantalla con realidad aumentada la información de los distintos lugares que rodean al individuo en una posición concreta. En la figura 3.5 se puede ver un ejemplo de como muestra la información esta aplicación.



Figura 3.5: Sistema para la visualización de información a través de realidad aumentada

3.2.6. aQRdate

Es un sistema que ayuda a personas con daño cerebral a realizar tareas cotidianas, como planchar o preparar una comida, recordándoles los pasos a seguir para completar la tarea. De esta forma tendrán una mayor autonomía ya que no hace falta que estén acompañados por un familiar o terapeuta que les indique los pasos, será su teléfono móvil el que los asista.

Este sistema genera, de forma automática interfaces de usuario ubicuas para controlar los elementos de un entorno inteligente desde un teléfono móvil. Para ello, se propuso etiquetar algunos elementos del entorno con códigos QR. Así el usuario tan sólo tiene que enfocar la cámara de su dispositivo móvil hacia la etiqueta y se genera la interfaz de usuario asociada a ese elemento de forma automática y adaptada al dispositivo.

El flujo de la aplicación es lineal. En la Figura 3.6 se muestra un esquema de éste. Una vez abierta la aplicación, automáticamente se muestra la pantalla de captura y decodificación de códigos QR (gracias a la aplicación Barcode Scanner y la librería ZXing).

El proceso es automático, el usuario no tiene que pulsar ningún botón, tan sólo apuntar con la cámara del móvil hacia el código. Tras la decodificación van apareciendo los pasos según los va pidiendo al servidor y el usuario según los vaya cumpliendo va avanzando hasta finalizar la tarea.

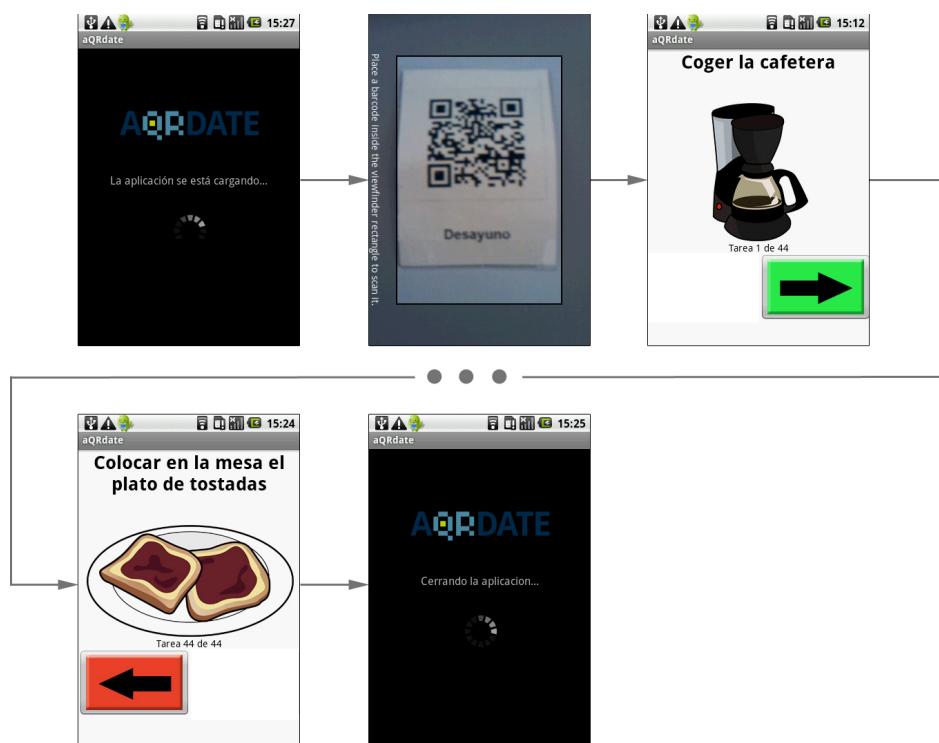


Figura 3.6: aQRdate

Capítulo 4

Método de trabajo

EN este capítulo se expone la planificación seguida durante el desarrollo del proyecto, la metodología utilizada y se explica cada fase de ésta con sus respectivos diagramas o fragmentos de código.

4.1. Metodología utilizada

La metodología que se ha seguido para el desarrollo de la aplicación *SocialFrame* es el PUD (Proceso Unificado de Desarrollo) [32] ya que se considera el más adecuado para el proyecto. El PUD es un marco de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML [16] constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Esta metodología está basada en una integración de forma unificada, cohesiva y adaptable del desarrollo de sistemas de software. Por tal razón es que este sistema no está del todo definido, sino que se adapta al contexto y necesidad de cada organización. Las características del PUD son las siguientes:

Iterativo e incremental Está compuesto por cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada fase está dividida en una serie de iteraciones. Las iteraciones finalizan con un incremento del producto desarrollado, añadiendo o mejorando las funcionalidades del sistema. En cada iteración se realizan una serie de actividades que incluyen un Análisis , Diseño, Implementación y Pruebas. A cada iteración, que engloba un caso de uso, se le dedica en principio el mismo tiempo respecto a cada una de las actividades.

Dirigido por casos de uso Los casos de uso capturan los requisitos funcionales del proyecto. Cada iteración incluye un conjunto de casos de uso que definen su contenido.

Centrado en la arquitectura No existe un único modelo que incluya todos los aspectos del sistema. Por ello la arquitectura software de un sistema está definida por múltiples

modelos y vistas.

El PUD combina las prácticas comúnmente conocidas como *buenas prácticas*, tales como ciclo de vida iterativo y desarrollo dirigido por casos de uso, en una descripción consistente y bien documentada. El proceso unificado se ha convertido en un proceso de desarrollo de software de gran éxito para la construcción de sistemas orientados a objetos.

Como se ha mencionado antes, esta metodología está completa gracias al lenguaje UML, es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema. Se trata de un estándar respaldado por el OMG (Object Management Group) [33]. Cuenta con una serie de diagramas que permiten modelar diferentes aspectos del sistema. Estos diagramas permiten especificar y describir métodos o procesos. Al fin y al cabo, el lenguaje de modelado se utiliza para documentar y construir sistemas software.

4.2. Planificación

Siguiendo el PUD, la planificación del proyecto está dividida en varias iteraciones, cada iteración tendrá un análisis de requisitos, un diseño, una implementación y unas pruebas. Además se tienen en cuenta unas iteraciones previas de preparación para conocer las distintas herramientas que se han utilizado en el proyecto y al final de éste se procede a realizar una serie de evaluaciones por parte del usuario final.

Las fases previas al desarrollo de la aplicación han sido las siguientes:

- Estudio del estado del arte y antecedentes: Octubre de 2011.
- Estudio de la plataforma Android: Noviembre 2011.
- Aprendizaje de utilización de códigos QR y API de Facebook: Diciembre de 2011.

Tras estas fases de preparación para poder abordar el desarrollo de la aplicación se procede al desarrollo de la misma, siguiendo las siguientes iteraciones:

- Iteración 1: Login del usuario.
- Iteración 2: Información del usuario.
- Iteración 3: Mensajes privados.

- Iteración 4: Eventos y cumpleaños.
- Iteración 5: Lista de amigos.
- Iteración 6: Visualizar y escribir comentarios en el muro.
- Iteración 7: Visualizar y subir fotos.

Se han elegido estas iteraciones que incluyen diversos elementos de una red social por dos grandes razones:

Sobrecarga del usuario Para no sobrecargar a la persona mayor o dependiente se han utilizado aquellos elementos más importantes de una red social. Aunque hay redes sociales que ofrecen más variedad de acciones, se ha creído poco conveniente no incluirlas para no desbordar al usuario.

Restricciones de la red social Otra razón de la elección de estos elementos de la red social es el problema de permisos que interponen dichas redes. Hay ciertos elementos de una red social que no pueden ser mostrados (amigos de un amigo) o acciones que no pueden ser llevadas a cabo (enviar mensaje privado).

Por último, tras el desarrollo de la aplicación se ha realizado una evaluación de la satisfacción de los usuarios finales (pruebas de usabilidad) que nos proporcionan unas estadísticas de la aceptación de la aplicación en ciertos grupos de la sociedad.

4.3. Análisis de requisitos

Ésta es la primera etapa del desarrollo de cualquier proyecto software, consiste en extraer los requisitos del producto. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios e intentar cambiarlos por otros que sean aceptados por los clientes. Para cada iteración del desarrollo del proyecto se definen los requisitos correspondientes, siendo lo más claro posible.

4.3.1. Iteración 1: Login del usuario

- Un usuario debe estar registrado en la red social.
- En caso de no estar autenticado en la aplicación, se procede a ello introduciendo un usuario y una contraseña.
- Si está autenticado no tiene que volver a hacerlo en próximas sesiones.

4.3.2. Iteración 2: Mostrar Información del usuario

- Capturar códigos QR con el lector para realizar las metáforas.
- Mostrar los datos personales de un usuario.
- Mostrar los datos personales de un amigo.
- Esos datos son el nombre, apellidos, fecha de nacimiento, localidad, país y sexo.

4.3.3. Iteración 3: Mensajes privados

- Mostrar los mensajes privados de un usuario.
- Dicho mensaje muestra la fecha, el contenido y el usuario que lo envío.
- Contestar a dicho mensaje a través de un comentario público.

4.3.4. Iteración 4: Eventos y cumpleaños

- Mostrar los eventos y cumpleaños de un usuario.
- Los eventos y cumpleaños mostrados son del día actual.
- Mostrar la información detallada del evento.

4.3.5. Iteración 5: Lista de amigos

- Mostrar la lista de amigos.
- Mostrar las peticiones de amistad.
- Aceptar la petición de amistad.

4.3.6. Iteración 6: Comentarios del muro

- Mostrar los comentarios del muro del usuario.
- Mostrar los comentarios del muro de un amigo.
- Escribir un comentario en el muro del usuario.
- Escribir un comentario en el muro de un amigo.

4.3.7. Iteración 7: Fotos

- Mostrar las fotos del usuario.
- Mostrar las fotos de un amigo.
- Hacer zoom sobre una foto.
- Subir una foto a través de la cámara.
- Subir una foto a través de la galería de imágenes.

4.3.8. Diagrama de casos de uso

Un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. En otras palabras, un caso de uso es una secuencia de interacciones

que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. Los diagramas de casos de uso se utilizan para ilustrar los requisitos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo.

En la figura 4.1, se puede observar el diagrama de casos de uso de la aplicación SocialFrame. En la parte izquierda del diagrama se encuentra el actor usuario que representa a la persona dependiente o mayor, en el lado derecho se encuentra la red social Facebook. La aplicación se representa en el centro como un sistema que contiene esos casos de uso, además se puede ver que para hacer cualquier acción sobre la red social, primero el usuario debe loguearse y después capturar un código QR. Algunos casos de uso tienen extensiones, es decir, en el mismo caso se pueden realizar otras acciones también importantes de la aplicación.

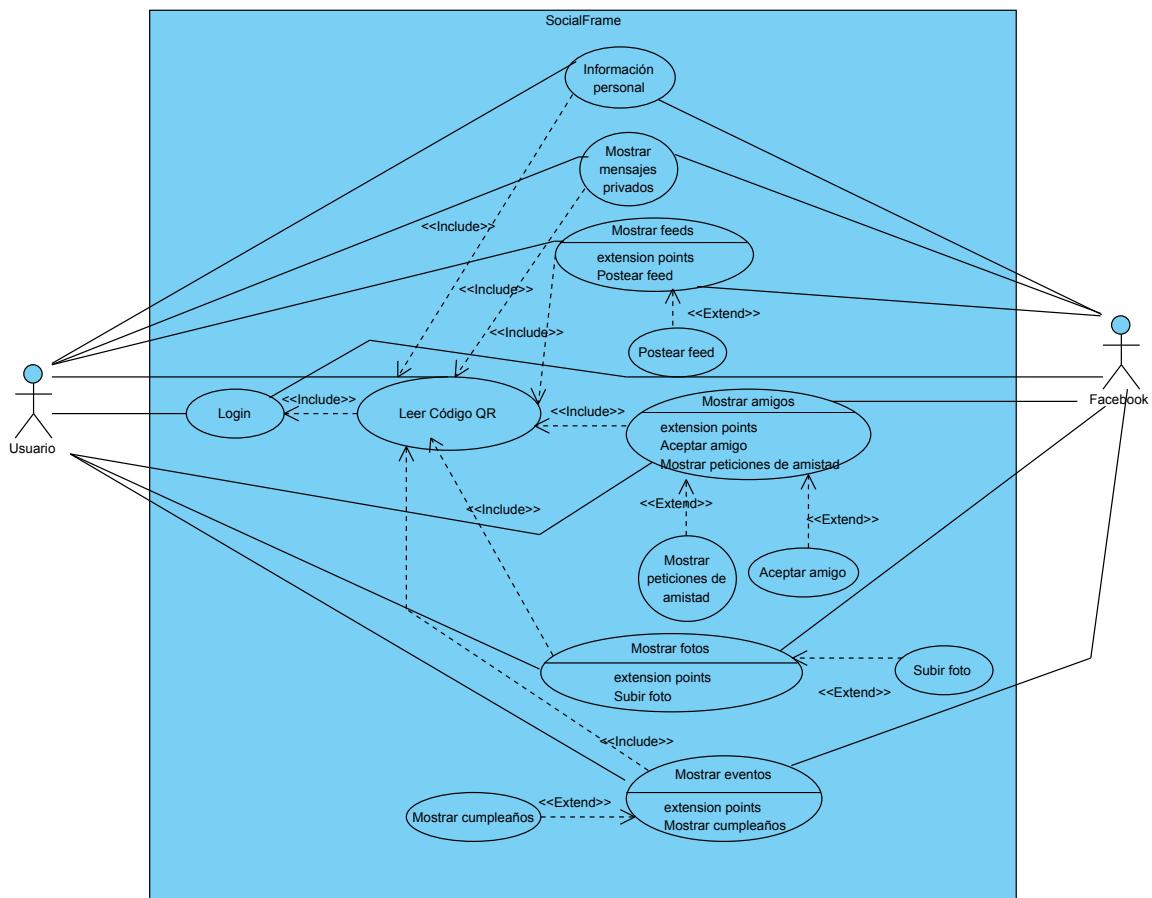


Figura 4.1: Diagrama de Casos de Uso

4.4. Fase de diseño

4.4.1. Diagramas de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Mientras que el diagrama de casos de uso permite el modelado de una vista del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos. Este diagrama consta de dos dimensiones: el eje vertical representa el tiempo, mientras que el horizontal representa los objetos.

Se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se tiene modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede caminar sobre esos pasos para descubrir qué objetos son necesarios. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

Se ha decidido mostrar los siguientes diagramas de secuencia: login del usuario, mostrar información del usuario, mostrar mensajes privados del usuario, mostrar eventos y cumpleaños, mostrar la lista de amigos, mostrar el muro de comentarios del usuario, escribir un comentario en el muro, mostrar las fotos de un amigo y subir una foto a través de la cámara.

Login del usuario

En la figura 4.2 se observa el diagrama de secuencia del login del usuario. Representa las acciones que debe realizar el usuario para poder *loguearse* en la red social. En este flujo de la aplicación, el usuario no introduce ningún dato inicial. Por defecto, SocialFrame lanza la llamada del login a la red social Facebook y si el usuario no está autenticado, la red social ejecuta un *dialog login* a la aplicación. A continuación, el usuario introduce su usuario y contraseña, si son incorrectos, los sigue pidiendo. En el caso de que los valores sean correctos, la aplicación SocialFrame muestra el lector de códigos QR.

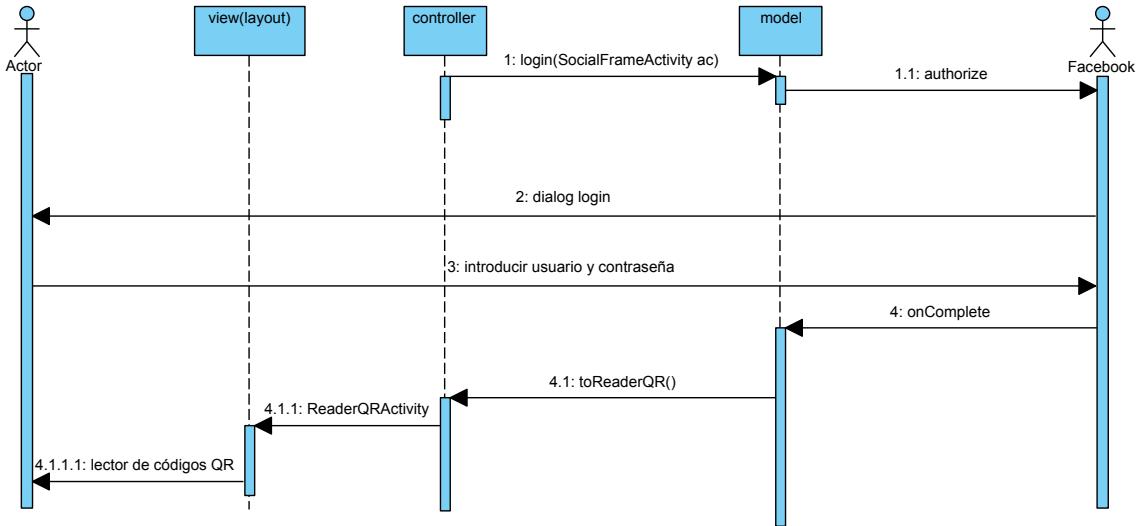


Figura 4.2: Diagrama de secuencia del login

Mostar información del usuario

Como se puede ver en la figura 4.3 se representa el diagrama de secuencia para poder mostrar la información del usuario. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de usuario. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *InfoActivity* y lanza el método *showUser* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updateUser*. Al método se le pasa un objeto *User* que actualiza la vista de la información del usuario y así el usuario puede consultar los datos.

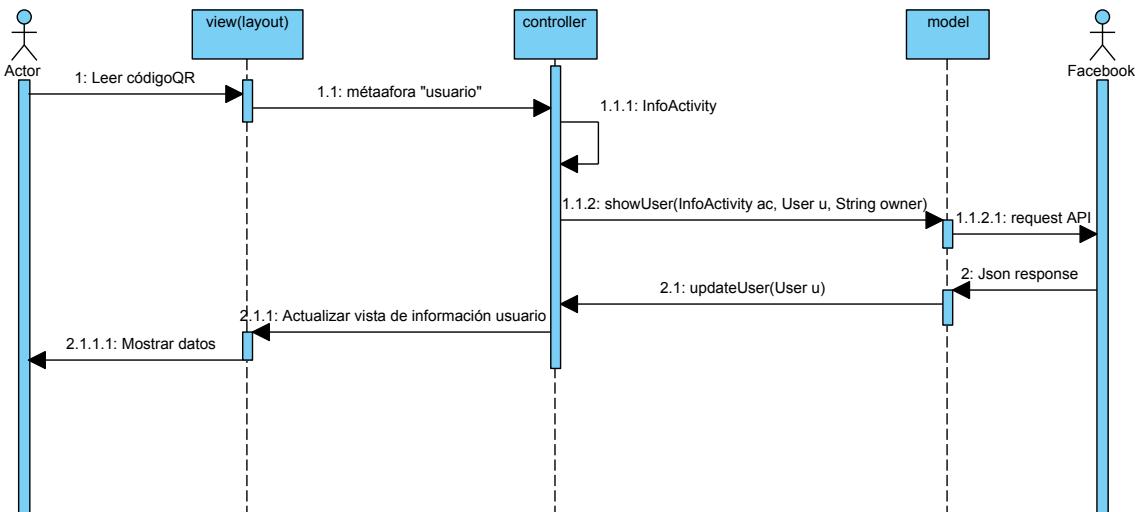


Figura 4.3: Diagrama de secuencia para mostrar información del usuario

Mostar mensajes privados del usuario

En la figura 4.4 se representa el diagrama de secuencia para poder mostrar los mensajes privados del usuario. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de mensaje. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *MessageActivity* y lanza el método *showMessages* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updateMessages*. Al método se le pasa un arraylist del tipo *Message* que actualiza la vista de los mensajes privados y así el usuario puede consultarlos.

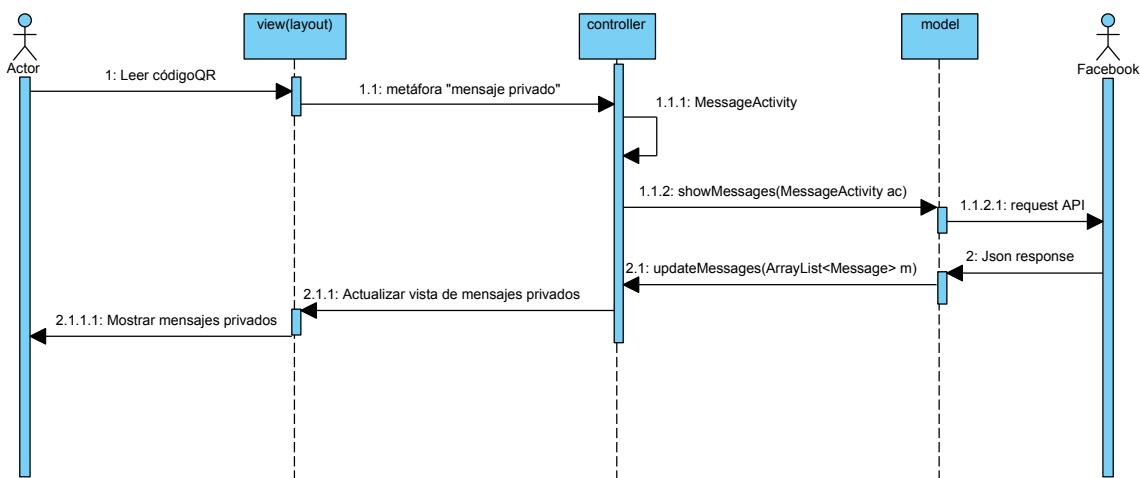


Figura 4.4: Diagrama de secuencia para mostrar mensajes privados

Mostar eventos y cumpleaños

Como se observa en la figura 4.5 se representa el diagrama de secuencia para poder mostrar los eventos del usuario y los cumpleaños de los amigos. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de evento. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *EventsActivity* y lanza el método *showEvents* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updateEvents*. Al método se le pasa un arraylist del tipo *Event* y un arraylist del tipo *String* (contiene el nombre de los amigos que cumplen años) que actualiza la vista de los eventos y así el usuario puede consultarlos.

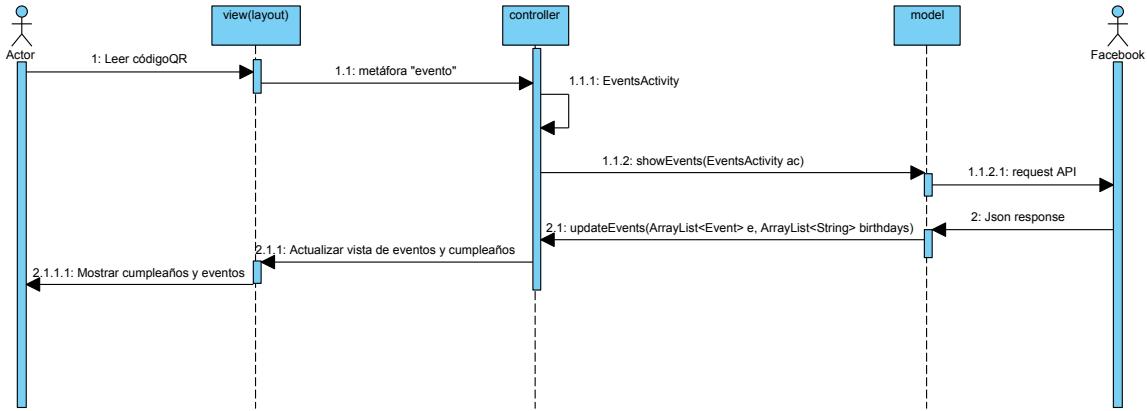


Figura 4.5: Diagrama de secuencia para mostrar eventos y cumpleaños

Mostrar la lista de amigos

En la figura 4.6 se representa el diagrama de secuencia para poder mostrar la lista de amigos del usuario. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de amigo. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *FriendsActivity* y lanza el método *showFriends* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updateFriends*. Al método se le pasa un arraylist del tipo *User* (contiene los amigos del usuario) que actualiza la vista con la lista de amigos donde el usuario puede consultarlos.

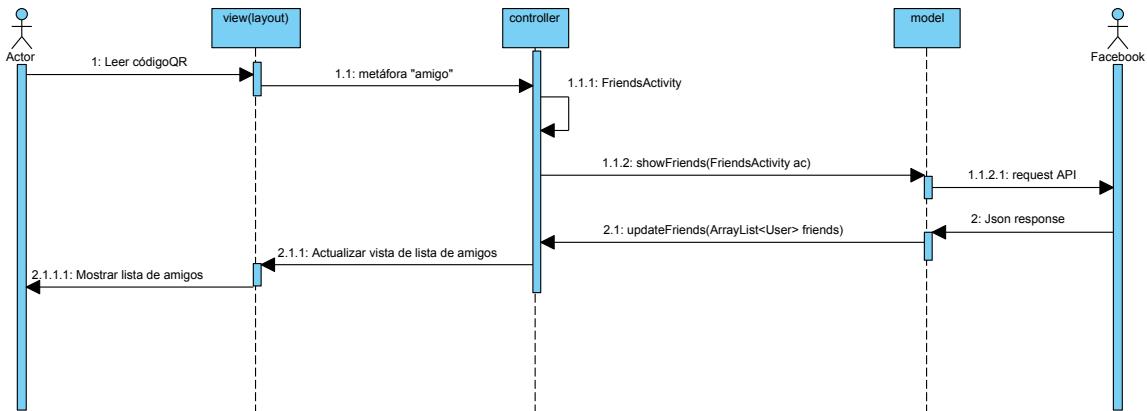


Figura 4.6: Diagrama de secuencia para mostrar la lista de amigos

Mostrar el muro de comentarios del usuario

Como se puede ver en la figura 4.7 se representa el diagrama de secuencia para poder mostrar los comentarios del muro del usuario. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de muro. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *FeedsActivity* y lanza el método *showFeeds* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updateFeeds*. Al método se le pasa un arraylist del tipo *Feed* (contiene los comentarios del muro del usuario) que actualiza la vista del muro donde el usuario puede consultarlos.

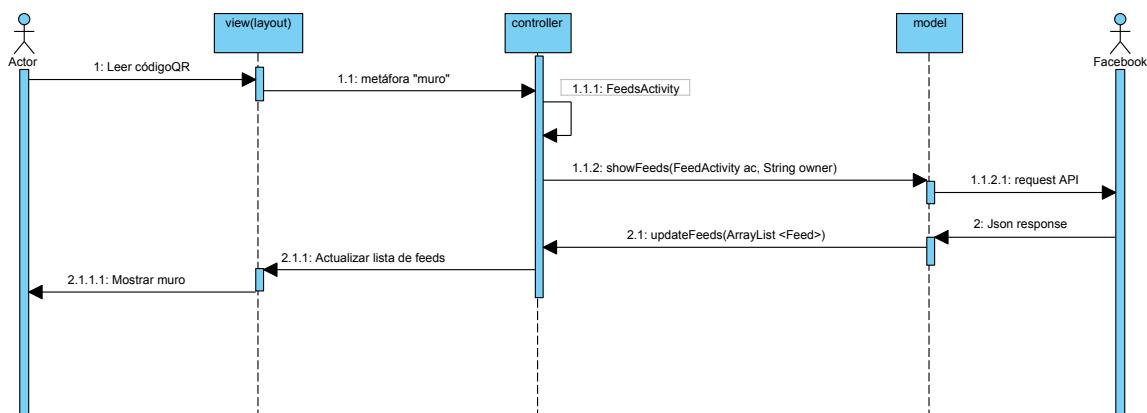


Figura 4.7: Diagrama de secuencia para mostrar los comentarios del muro

Escribir un comentario en el muro

En la figura 4.8 se observa el diagrama de secuencia correspondiente al caso de escribir un comentario en el muro. El flujo del diagrama es el siguiente: tras mostrar el muro del usuario, éste escribe el comentario a subir por voz, es decir, el usuario expresa su comentario por voz y la aplicación lo traduce a texto. La capa de vista envía a la capa de controlador el valor de ese comentario. Una vez hecho esto, la capa de controlador lanza el método *postFeed* que realiza una petición en la capa de modelo a la red social Facebook. Una vez completada esa petición, la capa de controlador lanza el método *refresh* que vuelve a realizar una petición a la red social Facebook. Como respuesta se recibe un objeto JSON, se decodifica y se procede a llamar a la capa de controlador con el método *updateFeeds*. Al método se le pasa un arraylist del tipo *Feed* (contiene los comentarios del muro del usuario) que actualiza la vista del muro donde el usuario puede consultarlos.

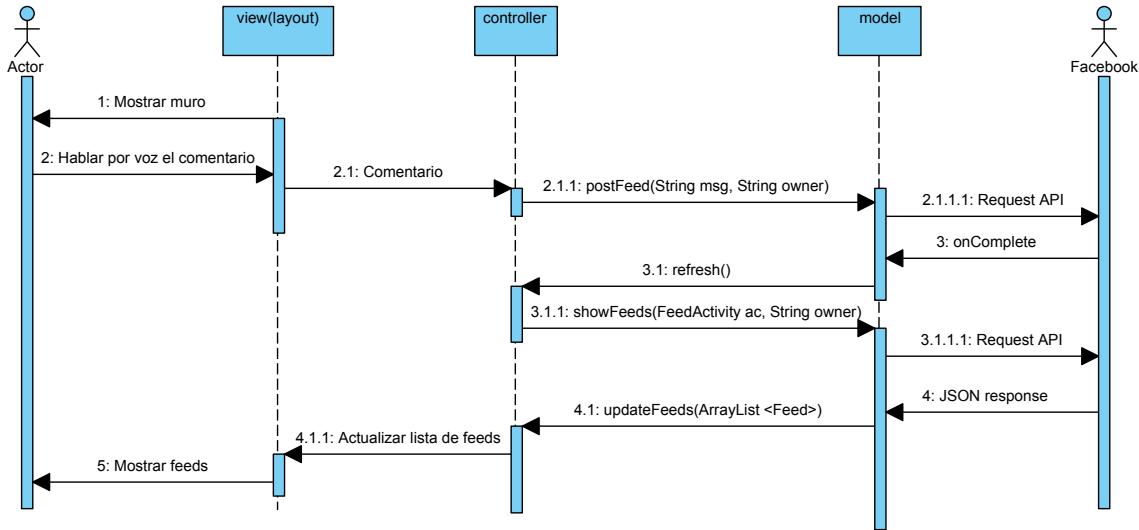


Figura 4.8: Diagrama de secuencia para escribir un comentario

Mostar las fotos de un amigo

Como se observa en la figura 4.9 se representa el diagrama de secuencia para poder mostrar las fotos de un amigo. El flujo del diagrama es el siguiente: el usuario capta un código QR con el lector cuya metáfora es la de amigo. La capa de vista envía a la capa de controlador el valor de ese código. Una vez hecho esto, la capa de controlador ejecuta la *FriendsActivity* y lanza el método *showFriends* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON, se decodifica y se procede a llamar a la capa de controlador con el método *updateFriends*. Al método se le pasa un arraylist del tipo *User* (contiene los amigos del usuario) que actualiza la vista con la lista de amigos donde el usuario puede consultarlos. A continuación, el usuario pulsa sobre un amigo de la lista, la capa de vista le envía ese amigo seleccionado a la capa de controlador que ejecuta la ventana de opciones de amigo. El usuario ve esas opciones ya que la capa de vista se las muestra y elige la opción fotos, que devuelve las fotos de una amigo. La capa de controlador ejecuta la *PhotoActivity* y lanza el método *showPhotos* que realiza una petición en la capa de modelo a la red social Facebook. Como respuesta se recibe un objeto JSON que se decodifica y se procede a llamar a la capa de controlador con el método *updatePhotos*. Al método se le pasa un arraylist del tipo *String* (contiene las url de las fotos) que actualiza la vista de las fotos donde el usuario puede consultarlas.

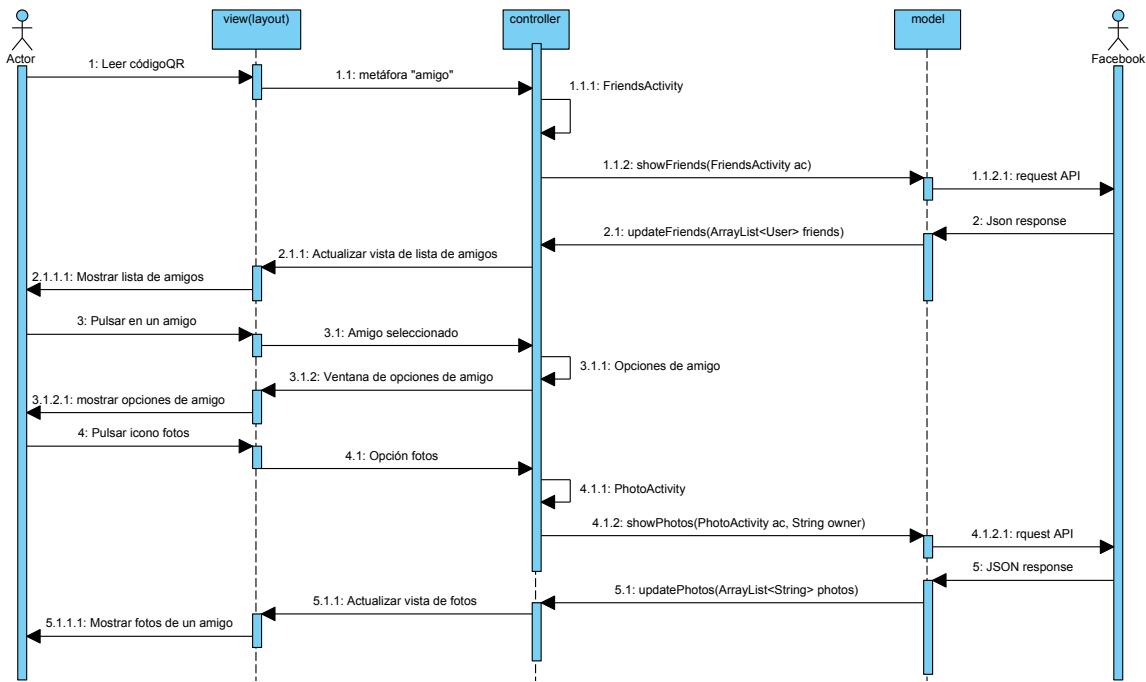


Figura 4.9: Diagrama de secuencia para mostrar las fotos de un amigo

Subir una foto a través de la cámara

En la figura 4.10 se observa el diagrama de secuencia correspondiente al caso de subir una foto a través de la cámara. El flujo del diagrama es el siguiente: tras mostrar las fotos del usuario, éste pulsa el botón de subir foto que provoca un evento en la capa de controlador. A continuación se lanza una ventana con las opciones para subir una foto, el usuario elige la opción cámara. La capa de controlador ahora procede a ejecutar la cámara del dispositivo móvil para que el usuario pueda sacar una foto. Una vez sacada, la capa de vista envía a la capa de controlador esa foto a través de un *Bundle*. Una vez hecho esto, la capa de controlador lanza el método *uploadPhoto* que realiza una petición en la capa de modelo a la red social Facebook. Una vez completada esa petición, la capa de controlador muestra las fotos de nuevo al usuario.

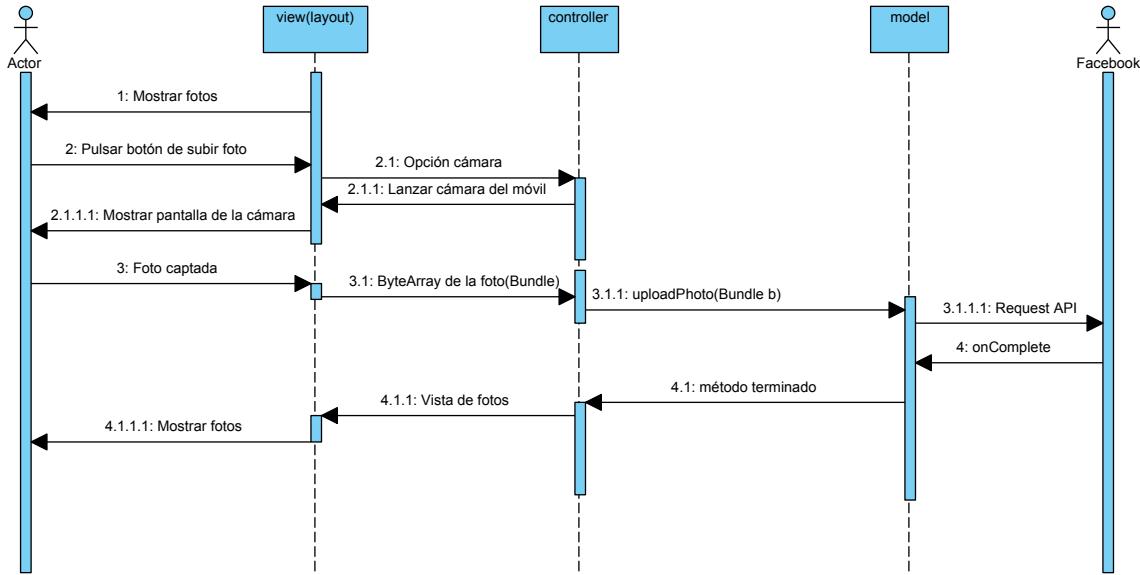


Figura 4.10: Diagrama de secuencia para escribir un comentario

4.4.2. Arquitectura de la aplicación

La aplicación está estructurada en diferentes capas (Modelo, Vista y Controlador), siguiendo el patrón MVC (Model View Controller) (véase figura 4.11). Se trata de un útil patrón de diseño arquitectónico que proporciona numerosas ventajas.

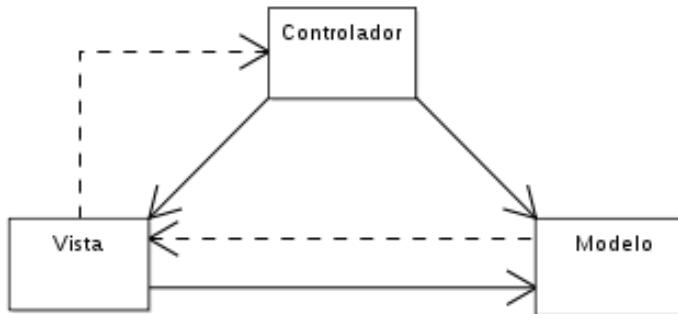


Figura 4.11: Modelo Vista Controlador

Capa de vista Contiene la interfaz gráfica móvil. Esta interfaz gráfica principalmente se compone de una serie de layouts, tanto en formato portrait como landscape, que muestran una pantalla en el dispositivo móvil y permiten la realización de operaciones llamando a la capa de controladores.

Capa de controladores Contiene la captura de información de la vista, así como la realización de eventos sobre el modelo y la vista. Se compone de una serie de Activities que recogen los datos de los layouts para llamar a la capa de modelo.

Capa de modelo Se encarga de realizar los servicios a la red social, así como de recibir los objetos devueltos para tratarlos y poder actualizar la vista.

La utilización de esta arquitectura implica directa o indirectamente el uso del patrón *Observer*.

4.4.3. Patrones de diseño

Los patrones de diseño [34] utilizados en el desarrollo de la aplicación SocialFrame son los siguientes:

Observer

Es un patrón de diseño que define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Se trata de un patrón de comportamiento, es decir, está relacionado con algoritmos de funcionamiento y asignación de responsabilidades a clases y objetos. La utilización de este patrón se basa en que la interfaz de usuario no se quede esperando a la respuesta de la petición al servidor (servicio web), de tal forma que cuando devuelva la respuesta, la interfaz actualizará dicho contenido en un hilo.

Adapter

Este patrón convierte la interfaz de una clase en otra distinta que es la que esperan los clientes y permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles. Se utiliza para poder adaptar cualquier red social a la aplicación, de esa forma un usuario podría tener varias redes sociales. De una forma más clara, una clase padre implementa de forma abstracta una serie de métodos, entonces se pueden generar clases hijas que deben implementar los métodos de la clase padre. Esta implementación puede ser vacía, es decir, si esa clase hija que es una red social específica no necesita un método de la clase padre, lo implementa vacío. Además el patrón Adapter permite que se incluya una red social específica (clase hija) pero con métodos distintos, es decir, que si los métodos son diferentes se acoplan los de clase hija a los de la clase padre y de esa forma el funcionamiento es igual que si tuviese los métodos genéricos.

4.4.4. Interfaz de usuario

La interfaz de usuario representa la capa de vista. Existen muchos tipos de ficheros que definen la interfaz del usuario. Todos ellos se hayan en la carpeta resources.

Drawable Iconos para diferentes resoluciones de pantalla del teléfono.

Layout XML que definen el layout portrait (posición vertical del móvil) de cada Activity.

Layout-land XML que definen el layout landscape (posición horizontal del móvil) de cada Activity.

La aplicación tiene una serie de pantallas definidas por ficheros XML y usando como base diversos componentes del sistema operativo Android. Entre los diferentes componentes utilizados destacan los siguientes:

LinearLayout: Es un contenedor que permite colocar los elementos de forma lineal.

RelativeLayout: Es un contenedor que permite colocar los elementos de forma relativa a otros.

ListView: Permite mostrar una lista de elementos.

Button y ImageButton: Botones para permitir al usuario la realización de eventos.

TextView: Muestran un texto.

ImageView: Muestran un ícono o imagen.

4.4.5. Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona. Estos ejemplos de clases de objetos reales son sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales, datos que corresponden al objeto. Debido a que no se puede mostrar el diagrama de clases entero, éste se divide en los paquetes que lo compone (vista, controlador y modelo). En el caso del paquete vista al ser layout (XML) no se muestra ya que los archivos XML no se pueden definir como clase y por tanto no se pueden representar en el diagrama de clases.

Controlador

Como se puede ver en la figura 4.12, el paquete controller está formado por los *activities* de la aplicación. Obtienen los datos de la vista y envian eventos al paquete de modelo o para actualizar el paquete de vista. Este diagrama contiene las siguientes *activities*: *EventsActivity* que crea la pantalla de eventos y cumpleaños, *HelpActivity* que genera pantallas de ayuda, *ReaderQRActivity* que ejecuta el lector de código QR, *InfoActivity* que crea la pantalla de información personal, *FriendsActivity* que crea una pantalla con la lista de amigos, *FeedActivity* que crea una pantalla con los comentarios del muro, *MessagesActivity* que crea una pantalla con los mensajes privados, *SocialFrameActivity* que crea la pantalla incial, *PhotoActivity* que crea un pantalla con las fotos y *ScaleImageView* que es un tipo de imagen para poder hacer zoom sobre una foto.

Modelo

Como se puede ver en la figura 4.13, el paquete model está compuesto por los distintos objetos de la aplicación y por los servicios para acceder a la red social. Este paquete incluye las siguientes clases: *Constants* que contiene las constantes necesarias para la aplicación, *Event* que crea objetos con información de eventos, *Message* que crea objetos con información de mensajes, *User* que crea objetos con información de usuarios, *Feed* que crea objetos con información de comentarios, *SocialNetwork* que es una clase padre para añadir redes sociales a la apliación, *FacebookNetSocial* que es una clase hija con los métodos para acceder a Facebook, *RequestListenerBase* que es una clase genérica para tratar las peticiones a Facebook, *Utility* que es una clase con métodos para ayudar a la aplicación en general y *UtilityFacebookNet* que es una clase con métodos de ayuda para la red social Facebook.

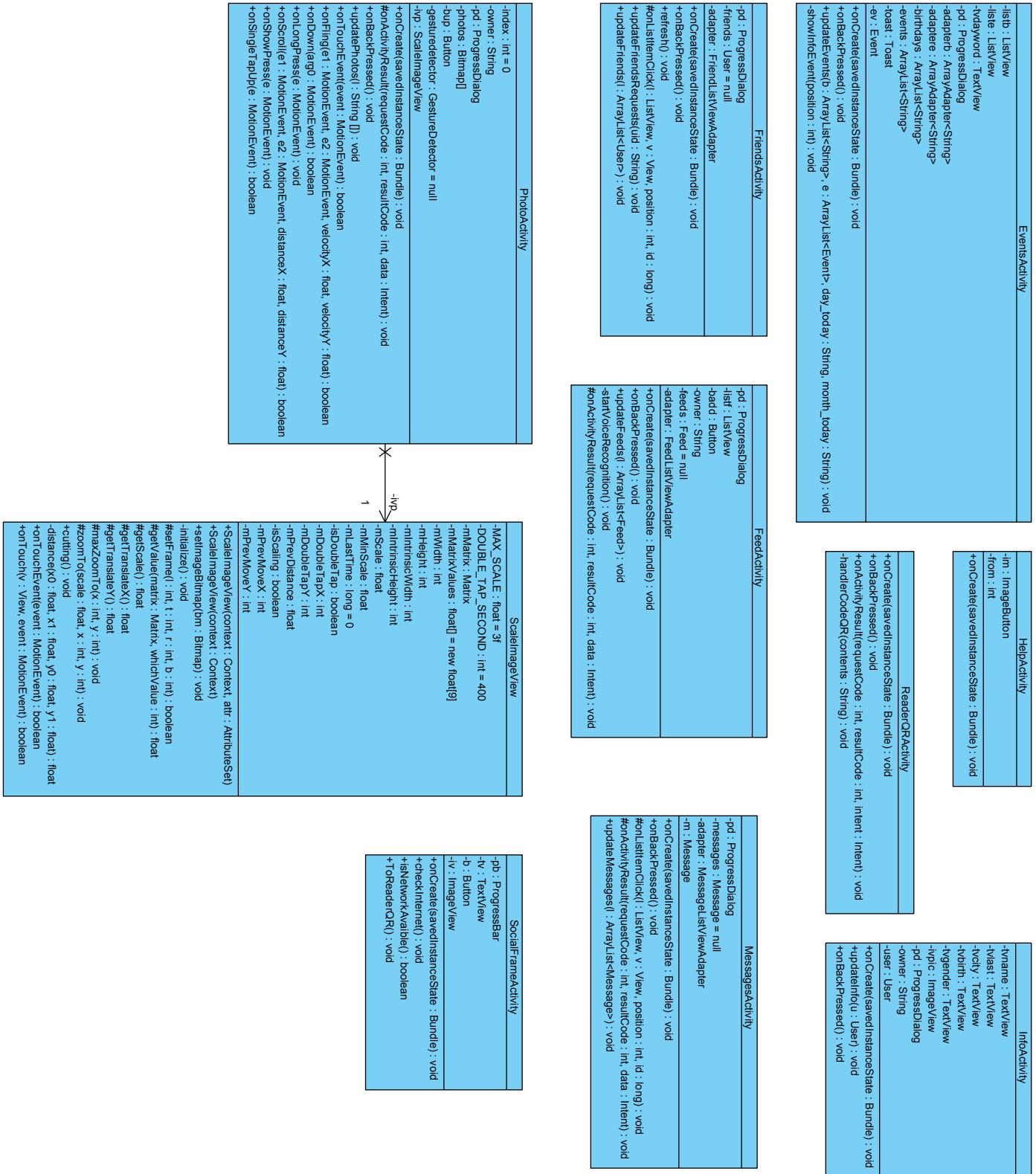


Figura 4.12: Diagrama de Clases del paquete controller

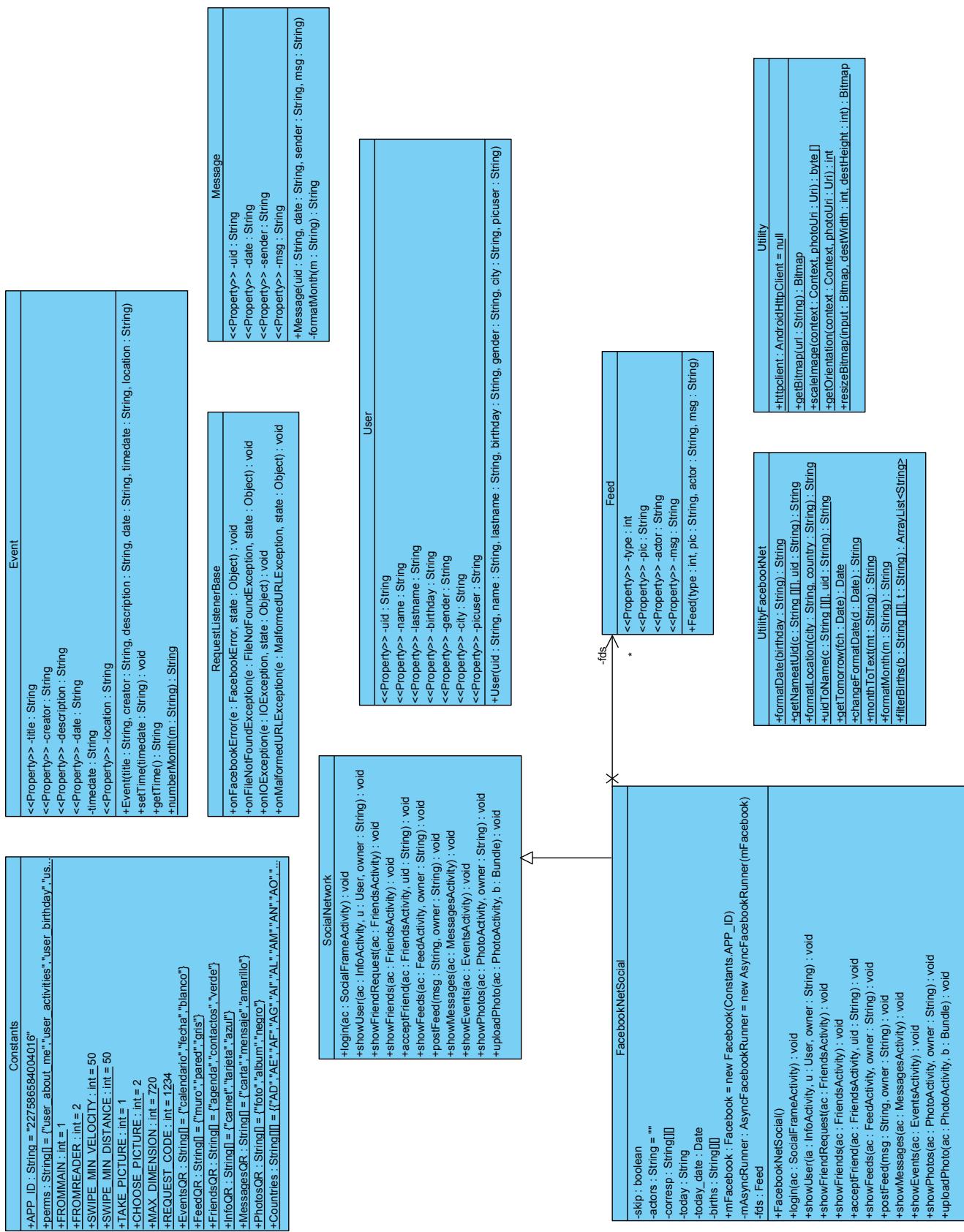


Figura 4.13: Diagrama de Clases del paquete model

4.5. Fase de implementación

En la fase de implementación de la aplicación SocialFrame se han incluido algunos fragmentos de código importantes y que describen cómo funcionan los servicios en la aplicación y otros aspectos considerables. Antes de iniciar la fase de implementación es necesario crear la aplicación y registrarla en Facebook, este proceso se observa en el Anexo A.

Para ver como se ha creado la interfaz de usuario, se muestra en el fragmento de código 4.1 que representa el layout de una pantalla de la aplicación, está escrito en XML. Esta parte de código representa la fila genérica de la lista de amigos que contiene un *RelativeLayout* donde se colocan elementos en función de la posición de otros anteriormente colocados. Este layout tiene un *ImageView* que sirve para mostrar el icono de un amigo y un *TextView* que sirve para mostrar el nombre de un amigo.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:id="@+id/RelativeLayout1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bfriends"
        android:padding="6dip" >
    <ImageView
        android:id="@+id/pic"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="5dp"
        android:scaleType="fitCenter" />
    <TextView
        android:id="@+id/namefriend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="5dp"
        android:layout_toRightOf="@+id/pic"
        android:ellipsize="marquee"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#0000FF" android:textStyle="bold"
        android:typeface="sans" android:textSize="20dp"/>
</RelativeLayout>
```

Listado 4.1: Layout de fila de amigos

Uno de los aspectos principales de la aplicación es comprobar al principio si se tiene conexión a Internet o no, para ello se muestra el fragmento de código 4.2 este aspecto. Al método no se le pasa ningún parámetro, simplemente se comprueba en el gestor de conectividades si contiene dicho servicio, para ello ve si hay alguna red de conexión activa, si es así devuelve

true y si no *false*.

```
public boolean isNetworkAvailable(){
    ConnectivityManager cm = (ConnectivityManager) getSystemService(
        Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = cm.getActiveNetworkInfo();
    return activeNetworkInfo != null;
}
```

Listado 4.2: Comprobar si hay conexión a Internet

En el fragmento de código 4.3 se implementa un login para acceder a la red social Facebook, si el login es completado, la aplicación se dirige al lector de códigos QR. Como se puede observar, primero se comprueba si la sesión actual es válida, es decir, si el usuario se ha autenticado hace poco o tiene guardado su usuario y contraseña. En caso de que no sea válida, se realiza una petición llamada *authorize* que como su propio nombre indica autoriza a un usuario a acceder a la red social, para ello, debe introducir una serie de permisos, en este caso genéricos, iguales para todos los usuarios. Esta llamada muestra el *dialog* creado por Facebook. Una vez completado, se guarda la sesión y se lanza el lector de códigos QR.

```
if( !mFacebook.isSessionValid() ) {
    mFacebook.authorize(sfa, Constants perms, Facebook.
        FORCE_DIALOG_AUTH,new DialogListener() {
            @Override
            public void onFacebookError(FacebookError e) {
                e.printStackTrace();
            }
            @Override
            public void onError(DialogError e){
                e.printStackTrace();
            }
            @Override
            public void onComplete(Bundle values) {
                SessionStore.save(mFacebook, sfa);
                sfa.ToReaderQR();
            }
            @Override
            public void onCancel() {
            }
        });
} else sfa.ToReaderQR();
```

Listado 4.3: Request a la API de Facebook para autentificar un usuario

Cuando se lanza el lector de códigos QR, este captura un código y por tanto devuelve una cadena de texto en este caso representando una metáfora referente a los elementos de la red social Facebook. Esta cadena se devuelve como un *Activity result*, es decir, la *Activity* lanza un *Intent* que es un programa externo a la aplicación y cuando acaba devuelve ese *result*.

En el fragmento de código 4.4 se observa como ese valor devuelto es transformado a letras minúsculas para que el código no deba ser de una forma u otra, sino que simplemente tenga el contenido de la metáfora. A continuación, se lanza un método de comparación con las posibles metáforas para cada caso, llamado *handlerCodeQR*.

```
if (requestCode == 0) {
    if (resultCode == RESULT_OK) {
        String contents = intent.getStringExtra("SCAN_RESULT");
        handlerCodeQR(contents.toLowerCase());
    }
}
```

Listado 4.4: Tratamiento al valor devuelto por el lector de códigos QR

Una vez que el usuario se ha autenticado y ha capturado un código QR, la aplicación realiza la petición a la API de Facebook. Hay dos formas de hacerlo: a través de la Graph API y a través de una consulta FQL. En el fragmento de código 4.5 se implementa la primera forma y devuelve peticiones de amistad. Para ello primero se realiza una petición asíncrona a la parte de la Graph API referente a las peticiones de amistad. Devuelve un objeto JSON en el método *onComplete* que simplemente contiene el uid de la petición de amistad. En caso de algún problema con el objeto JSON o con la Graph API de Facebook, salta la excepción correspondiente.

```
mAsyncRunner.request("me/friendrequests", new RequestListenerBase()
{
    @Override
    public void onComplete(String response, Object state) {
        try {
            String uid= "";
            JSONArray json = Util.parseJson(response).
                getJSONArray("data");
            if(json.length()!= 0){
                uid = json.getJSONObject(0).getJSONObject("from").
                    getString("id");
            }
            fa.updateFriendsRequests(uid);
        } catch (JSONException e) {
            e.printStackTrace();
        } catch (FacebookError e) {
            e.printStackTrace();
        }
    }
});
```

Listado 4.5: Request a la Graph API de Facebook

En el fragmento de código 4.6 se implementa la segunda forma y devuelve la lista de amigos de un usuario. Para ello primero se realiza una petición asíncrona a una base de datos

de Facebook, se consulta la base de datos *user* y se almacena esta consulta en un objeto de tipo *Bundle*, una clase de Android que contiene tipos primitivo y objetos de otras clases. Devuelve un objeto JSON en el método *onComplete* que simplemente contiene una arraylist de tipo *User* con la lista de amigos de un usuario. En caso de algún problema con el objeto JSON o con la Graph API de Facebook, salta la excepción correspondiente.

```

Bundle params = new Bundle();
params.putString("method", "fql.query");
params.putString("query", "SELECT uid,name,pic FROM user WHERE uid IN
    (SELECT uid2 FROM friend WHERE uid1= me())");
mAsyncRunner.request(params, new RequestListenerBase() {
    @Override
    public void onComplete(String response, Object state) {
        try {
            response = "{\"data\":" + response + "}";
            JSONArray json = Util.parseJson(response).getJSONArray(
                "data");
            ArrayList<User> friendlist = new ArrayList<User>();
            User f ;
            for(int i=0; i < json.length(); i++){
                f = new User("", "", "", "", "", "", "");
                f.setUid(json.getJSONObject(i).getString("uid"));
                f.setName(json.getJSONObject(i).getString("name"));
                f.setPicuser(json.getJSONObject(i).getString("pic"))
                    ;
                friendlist.add(f);
            }
            fa.updateFriends(friendlist);
        } catch (JSONException e) {
            e.printStackTrace();
        } catch (FacebookError e) {
            e.printStackTrace();
        }
    }
});
```

Listado 4.6: Request a través de una consulta FQL

Cuando ya se tiene la información devuelta por la red social Facebook, en la *Activity* correspondiente se procede a actualizar los elementos de la interfaz de usuario o capa de vista. Como se puede observar en el fragmento de código 4.7 se realiza un patrón observer de forma directa, es decir, la *Activity* lanza un ventana de carga manteniendo la interfaz activa hasta que llegue la información requerida. Cuando esto ocurra, en la *Activity* se ejecuta un hilo donde se actualiza la información, en este caso la lista de amigos de un usuario. Cuando se haya actualizado, la interfaz que siempre ha estado activa, elimina la ventana de carga y muestra los resultados.

Además de poder conseguir información de la red social, también se puede mandar enviar, de tal forma que si un usuario quiere escribir un comentario en el muro tiene que realizar el

```

FriendsActivity.this.runOnUiThread(new Runnable() {
    @Override
    public void run() {
        if(friends != null && friends.size() > 0){
            for(int i=0;i<friends.size();i++) adapter.add(friends
                .get(i));
        }
        adapter.notifyDataSetChanged();
        pd.dismiss();
    }
});

```

Listado 4.7: Actualización de información a través de un observer

fragmento de código 4.8. Como se puede observar, se crea un objeto de tipo *Bundle* que añade un mensaje para escribir en el muro y se accede a la Graph API pero de la forma *POST* que permite añadir información a la Graph API. La petición que se realiza es síncrona ya que no requiere patrón *observer*. En caso de producirse alguna excepción, ésta es capturada y mostrada.

```

try{
    Bundle parameters = new Bundle();
    if (owner.equals("")) owner = "me";
    parameters.putString("message", msg);
    mFacebook.request(owner+"/feed",parameters,"POST");
} catch(Exception e){
    e.printStackTrace();
}

```

Listado 4.8: Request a la API para escribir un comentario en Facebook

Todos los métodos de la aplicación que muestran información de una red social o que añaden información a ella son genéricos para cualquier tipo de red social. Para ello se crea una clase llamada *SocialNetwork* que hace uso del patrón *Adapter*, esta clase tiene definidos una serie de métodos abstractos que se deben implementar para cualquier red social específica. En el caso de que una clase hija tenga métodos diferentes, ésta se puede acoplar a la clase padre que es *SocialNetwork*, la cual se muestra en el fragmento de código 4.9. Si se quisiese añadir una red social nueva, que no fuese Facebook, se realiza el proceso que se muestra con un ejemplo en el Anexo B.

```
public abstract class SocialNetwork {  
  
    abstract public void login(SocialFrameActivity ac);  
  
    abstract public void showUser(InfoActivity ac, final User u, String  
        owner);  
  
    abstract public void showFriendRequest(FriendsActivity ac);  
  
    abstract public void showFriends(FriendsActivity ac);  
  
    abstract public void acceptFriend(FriendsActivity ac, String uid);  
  
    abstract public void showFeeds(FeedActivity ac, String owner);  
  
    abstract public void postFeed(String msg, String owner);  
  
    abstract public void showMessages(MessagesActivity ac);  
  
    abstract public void showEvents(EventsActivity ac);  
  
    abstract public void showPhotos(PhotoActivity ac, String owner);  
  
    abstract public void uploadPhoto(PhotoActivity ac, Bundle b);  
}
```

Listado 4.9: Clase padre del patrón adapter

4.6. Fase de pruebas

Se han diseñado y ejecutado una serie de pruebas para detectar posibles fallos en el sistema. Las pruebas realizadas incluyen pruebas unitarias, funcionales y de sistema.

4.6.1. Pruebas unitarias

Se ha creado un proyecto de pruebas para la correcta comprobación del funcionamiento del sistema. Para el desarrollo de las pruebas se ha utilizado el framework de pruebas unitarias JUnit. Las pruebas realizadas comprueban la validez de algunas de las operaciones realizadas en la aplicación Android. En el fragmento de código 4.10 se muestra la prueba unitaria referente a una petición sobre información de usuario. Como se puede ver, primero se comprueba si el usuario está logueado, si es así realiza el *try-catch* donde envía una petición sobre información personal propia. En el caso correcto devuelve el nombre del usuario. Cualquier error de petición o que el usuario no esté autenticado se devuelve un *false*.

Otra prueba unitaria sería si se realiza bien un post, es decir, escribir un comentario en el muro, para ello el fragmento de código 4.11 muestra este tipo de prueba. Como se puede observar también se comprueba que el usuario esté autenticado y si es así procede a crear un objeto de tipo *Bundle* al cual le inserta un mensaje de tipo *String* y realiza la petición a la

```

if (!mFacebook.isSessionValid()) return false;
try {
    Log.d("Tests", "Testing request for 'me'");
    String response = mFacebook.request("me");
    JSONObject obj = Util.parseJson(response);
    if (obj.getString("name") == null ||
        obj.getString("name").equals("")) {
        return false;
    }
} catch (Throwable e) {
    e.printStackTrace();
    return false;
}

```

Listado 4.10: Test sobre petición de información de usuario

red social Facebook para poder escribir un comentario en su muro. Si la respuesta no es nula, ni vacía, ni falsa, quiere decir que se ha comentado correctamente y por tanto se ha escrito en el muro. Cualquier error de la petición o de la respuesta o de que el usuario no esté logueado devuelve el valor *false*

```

if (!mFacebook.isSessionValid()) return false;
try {
    Log.d("Tests", "Testing graph API wall post");
    Bundle parameters = new Bundle();
    parameters.putString("message", "hello world");
    parameters.putString("description", "test test test");
    response = mFacebook.request("me/feed", parameters,
"POST");
    Log.d("Tests", "got response: " + response);
    if (response == null || response.equals("") ||
        response.equals("false")) {
        return false;
    }
} catch (Throwable e) {
    e.printStackTrace();
    return false;
}

```

Listado 4.11: Test sobre postear un comentario en el muro

4.6.2. Pruebas funcionales

Las pruebas funcionales han sido tenidas en cuenta a lo largo del desarrollo del proyecto, incluyen la comprobación de los requisitos funcionales. En el cuadro 4.1 se pueden observar las pruebas funcionales que han resultado satisfactorias.

Prueba funcional	Resultado
<i>Login del usuario correcto</i>	Prueba correcta
<i>Login del usuario incorrecto</i>	Prueba correcta
<i>Capturar código QR</i>	Prueba correcta
<i>Mostrar información del usuario</i>	Prueba correcta
<i>Mostrar mensajes privados</i>	Prueba correcta
<i>Mostrar cumpleaños de amigos</i>	Prueba correcta
<i>Mostrar eventos propios</i>	Prueba correcta
<i>Mostrar lista de amigos</i>	Prueba correcta
<i>Mostrar peticiones de amistad</i>	Prueba correcta
<i>Aceptar una petición de amistad</i>	Prueba correcta
<i>Mostrar información personal del amigo</i>	Prueba correcta
<i>Mostrar fotos del amigo</i>	Prueba correcta
<i>Mostrar muro del amigo</i>	Prueba correcta
<i>Escribir un comentario en el muro de un amigo</i>	Prueba correcta
<i>Mostrar muro propio</i>	Prueba correcta
<i>Escribir un comentario en mi muro</i>	Prueba correcta
<i>Mostrar mis fotos</i>	Prueba correcta
<i>Hacer zoom sobre una foto</i>	Prueba correcta
<i>Visualizar siguiente y anterior foto</i>	Prueba correcta
<i>Subir foto a través de la cámara</i>	Prueba correcta
<i>Subir foto a través de la galería</i>	Prueba correcta

Cuadro 4.1: Pruebas funcionales

4.6.3. Pruebas de sistema

Se han llevado a cabo las siguientes pruebas de sistema: pruebas de usabilidad (véase sección 5.4) y pruebas de compatibilidad (diferentes dispositivos móviles, diferentes versiones de Android). Las pruebas de compatibilidad ejecutan la aplicación en el Samsung Galaxy Tab 7' con una versión Android 2.2 y en el Geeksphone One con una versión Android 2.3.

Capítulo 5

Resultados

EN este capítulo se explican los resultados obtenidos durante el proyecto, las horas dedicadas a él a través de un seguimiento con un diagrama de Gantt, la aplicación final y la evaluación de la satisfacción del usuario final.

5.1. Seguimiento del proyecto

Para la planificación y posterior seguimiento del proyecto se ha utilizado un diagrama de Gantt desarrollado mediante una herramienta de gestión de proyectos como es Microsoft Project. El diagrama de Gantt es, seguramente, el tipo de calendario de proyecto más utilizado, quizás porque muchas personas lo encuentran más comprensible que otros. Es un diagrama de barras en forma de tabla que establece la relación entre las tareas (filas) y las duraciones de estas (columnas). El seguimiento del proyecto se puede ver en la figura 5.1, se detallan las iteraciones del desarrollo del proyecto, son secuenciales, es decir, hasta que no termina la iteración 1 no empieza la 2 y así sucesivamente. Se indica la fecha de comienzo y la de finalización.

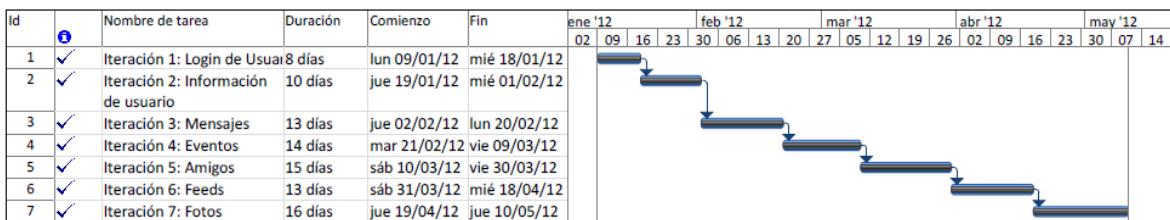


Figura 5.1: Diagrama de seguimiento de Gantt

5.2. Coste del proyecto

El coste económico del proyecto se puede ver en la tabla 5.1. Los precios por hora mostrados han sido obtenidos del sitio oficial <http://www.tusalario.es/main>.

Descripción	Coste	Unidades	Coste total
<i>Analista</i>	35 euros/h	144 h	5040 euros
<i>Diseñador</i>	50 euros/h	100 h	5000 euros
<i>Programador</i>	20 euros/h	255 h	5100 euros
<i>Tester</i>	15 euros/h	140 h	2100 euros
<i>Subtotal</i>			17240 euros
<i>18 % IVA</i>			3103 euros
<i>Total</i>			20343 euros

Cuadro 5.1: Coste económico del proyecto

El coste en horas del proyecto se puede ver en la tabla 5.2.

Fase	Horas
<i>Análisis</i>	144 horas
<i>Diseño</i>	100 horas
<i>Implementación</i>	255 horas
<i>Pruebas</i>	140 horas
<i>Total</i>	639 horas

Cuadro 5.2: Horas totales del proyecto

5.3. Aplicación Android

En esta sección se presenta la aplicación *SocialFrame* finalizada, se va detallando las posibles acciones que puede realizar el usuario así como los resultados mostrados por cada acción. La aplicación contiene un fichero *AndroidManifest.xml* que incluye la configuración principal de la aplicación dentro del sistema operativo Android. Algunos de los datos que podemos encontrar aquí son los permisos de la aplicación, la relación de actividades, la actividad de inicio, el logo y el nombre o la versión mínima de API Android en la que puede ejecutarse.

5.3.1. Inicio de la aplicación

Cuando el usuario ha instalado la aplicación en el dispositivo móvil, se inicia y aparece la pantalla principal o inicio, que se puede ver en la figura 5.2. En esta pantalla aparece el logo de la aplicación y se procede a comprobar que el dispositivo móvil tiene conexión a Internet, ya sea bien con WiFi o con 3G.

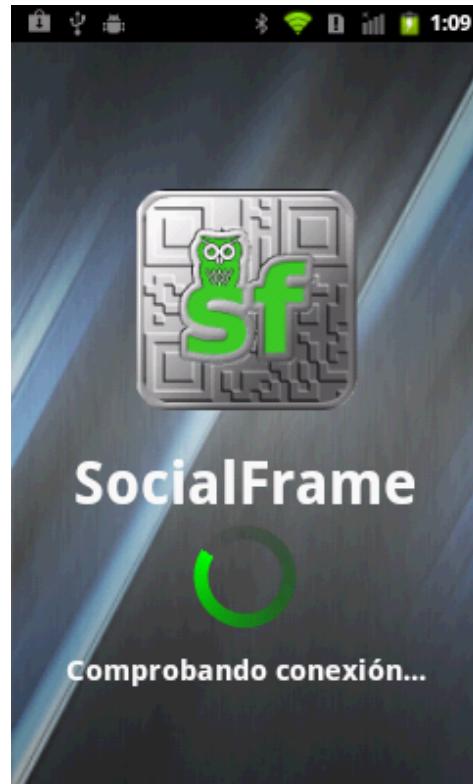


Figura 5.2: Pantalla inicial de la aplicación

La aplicación al tener definidos los layouts tanto en *portrait* como en *landscape*, es decir, permite ver la interfaz en horizontal y en vertical. Como se puede ver en la figura 5.3, la aplicación está en formato *landscape*, esto ha ocurrido cuando el dispositivo móvil ha sido girado 90 grados. Se observa que al haber conexión a internet, SocialFrame muestra que dispone de ella y por tanto procede a realizar el login.



Figura 5.3: Pantalla inicial de la aplicación landscape

En el caso de no disponer de conexión a Internet, la aplicación no muestra el login, condición indispensable para poder acceder a la red social Facebook. En su lugar aparece un símbolo que indica que puede volver a comprobar si tiene conexión a internet. De esta forma el usuario puede intentar conectarse a una red, mientras la ejecución sigue abierta y después volver a pulsar ese símbolo para poder acceder al login.



Figura 5.4: Pantalla inicial sin conexión a internet

5.3.2. Login de la aplicación

Una vez que la aplicación ha comprobado que hay conexión se procede a mostrar el login de la red social Facebook, en el caso de que el usuario no esté autentificado todavía. Para realizar el login, el usuario debe tener una cuenta en la red social, es decir, debe haberse registrado previamente. En la figura 5.5 se observa el login que lanza Facebook en la aplicación SocialFrame.



Figura 5.5: Login de la aplicación

Si se han introducido los datos correctamente (usuario y contraseña) la propia aplicación pide si se quieren guardar los datos. En el caso de que desee en próximas sesiones no tener que introducir los datos, debe pulsar guardar en el navegador (véase figura 5.6).

5.3.3. Lector de códigos QR

Cuando el usuario ya está autentificado en la red social, la aplicación muestra una pantalla de ayuda para aprender a utilizar un lector de códigos QR, las instrucciones mostradas pueden saltarse pulsando la imagen que aparece abajo (véase 5.7). El lector de códigos QR va a capturar no sólo las metáforas referentes a los objetos de la vida cotidiana, sino también a colores para que así la persona dependiente o mayor pueda realizar un ejercicio de memoria, estos colores son: azul (información personal), verde (amigos), amarillo (mensajes privados), gris (muro), blanco (eventos y cumpleaños) y negro (fotos).



Figura 5.6: Login de la aplicación



Figura 5.7: Instrucciones de uso del lector

Una vez leídas las instrucciones de uso y pulsada la imagen que aparece en la pantalla de ayuda, el lector de códigos QR se muestra. Para poder utilizarlo, el usuario simplemente debe apuntar hacia un código QR y automáticamente lee su contenido. Dependiendo del contenido muestra una información u otra de la red social. En la figura 5.8 se observa como se captura un código QR. Para salir del lector de códigos QR, el usuario debe pulsar la tecla de *volver atrás* o *salida* del dispositivo móvil con lo que muestra la pantalla de créditos.



Figura 5.8: Lector de códigos QR

5.3.4. Información del usuario

Si el código QR leído hace referencia a una metáfora sobre la información de un usuario, la aplicación muestra una pantalla en que cargará dicha información. Como Facebook realiza peticiones asíncronas, la interfaz espera el resultado poniendo en pantalla un aviso con el mensaje *Cargando..*, como se puede ver en la figura 5.9.

Cuando la interfaz ha recibido la información de la red social, ésta procede a actualizar en pantalla y por tanto a mostrar la información requerida. Para el caso de los datos personales del usuario se muestra su ícono o avatar actual, su nombre, sus apellidos, su fecha de nacimiento, su localidad, su país y su sexo (véase figura 5.10). Si el usuario pulsa la tecla de *volver atrás* o *salida* del dispositivo móvil se vuelve al lector de códigos QR.

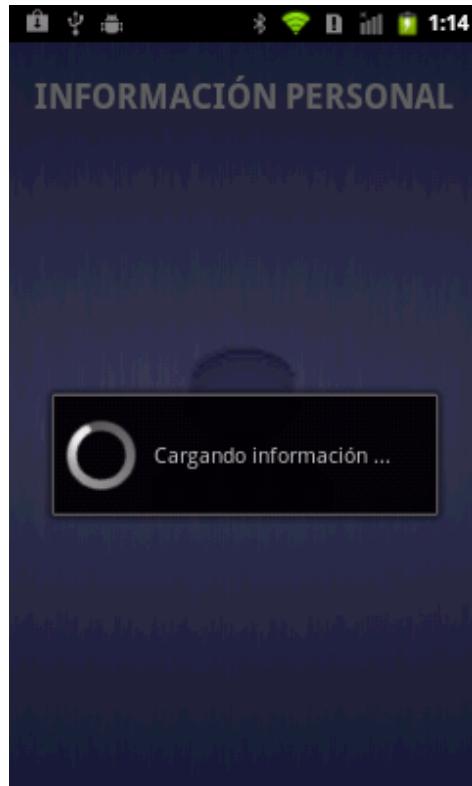


Figura 5.9: Aviso de cargando información

5.3.5. Mensajes privados

Otra de las acciones que puede realizar el usuario es mostrar los mensajes privados. Como se puede ver en la figura 5.11, la aplicación muestra la fecha, el contenido y el usuario que envío el mensaje privado. Aparece en forma de lista y por tanto arrastrando el dedo de abajo hacia arriba, la pantalla va mostrando los mensajes gradualmente. Para responder un mensaje privado simplemente hay que pulsar en él, de esta forma aparece el reconocedor de voz que veremos más adelante como funciona. Si el usuario pulsa la tecla de *volver atrás* o *salida* del dispositivo móvil se vuelve al lector de códigos QR.

5.3.6. Eventos y cumpleaños

Si la metáfora introducida se refería a calendario o cita, la aplicación SocialFrame muestra la pantalla de eventos y cumpleaños (véase figura 5.12). Esta pantalla muestra los eventos y cumpleaños que tienen lugar en el día actual, es decir, en la fecha de hoy. Aparecen dos listas, una con los amigos que cumplen años y otra con los eventos. Si el usuario pulsa la tecla de *volver atrás* o *salida* del dispositivo móvil se vuelve al lector de códigos QR.



Figura 5.10: Información personal del usuario

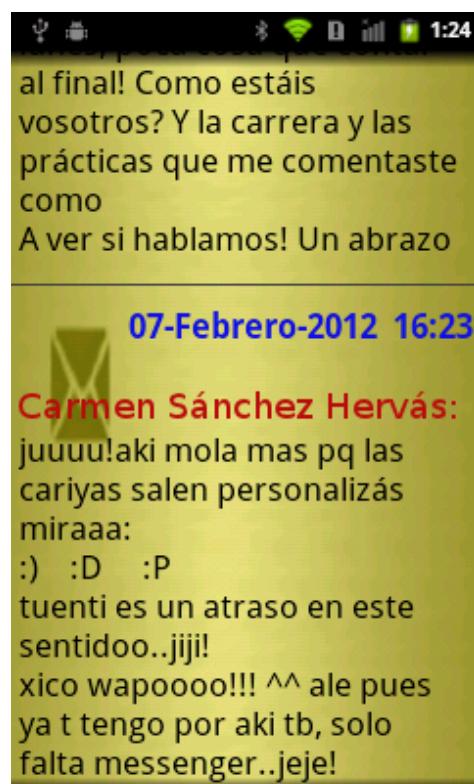


Figura 5.11: Mensajes privados



Figura 5.12: Eventos y cumpleaños

La mayoría de eventos tienen información detallada con el usuario que lo creó, la hora en la que va a ocurrir en el evento, el lugar y una pequeña descripción sobre él. Para poder consultar esta información el usuario simplemente tiene que pulsar el evento, a continuación aparece un ventana de información, como se puede ver en la figura 5.13. La ventana de información desaparece después de unos segundos.

5.3.7. Amigos

Si el usuario ha decidido acceder a los amigos que tiene en la red social, la aplicación muestra una lista con su nombre y su ícono, como se observa en la figura 5.14. Esta pantalla tiene un *scroll* vertical que va avanzando según arrasta el dedo de abajo hacia arriba, para bajar la lista y al contrario para subirla. En el caso de que tenga peticiones de amistad de nuevos amigos, la aplicación muestra un *dialog* con la posibilidad de aceptar o rechazar dicho amigo. Si el usuario pulsa la tecla de *volver atrás* o *salida* del dispositivo móvil se vuelve al lector de códigos QR.

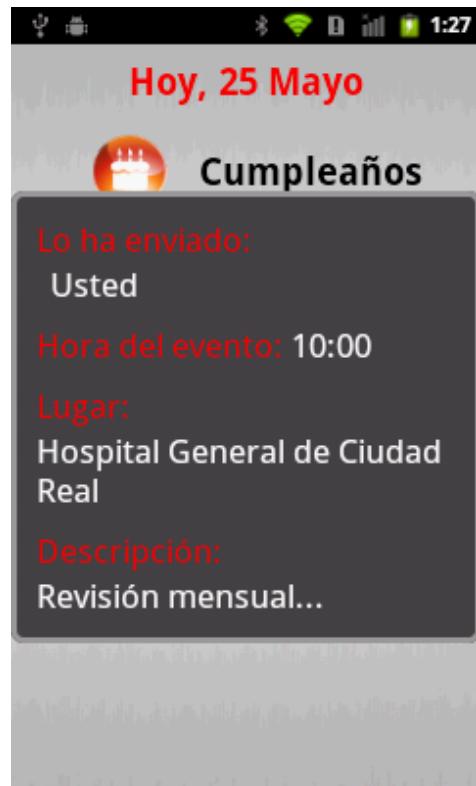


Figura 5.13: Información detallada del evento

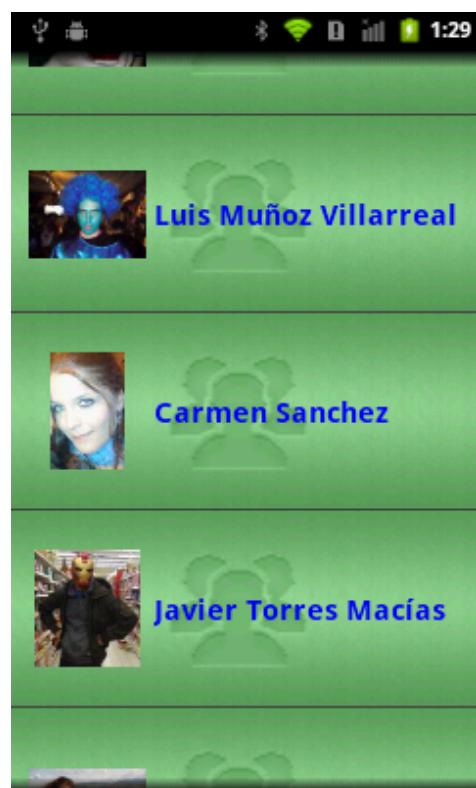


Figura 5.14: Lista de amigos

La aplicación SocialFrame puede mostrar información de los amigos, las posibilidades son: información personal, comentarios del muro y fotos. Para poder acceder a estas opciones, el usuario debe pulsar en cualquier amigo y le aparecen estas opciones, como se puede ver la figura 5.15.



Figura 5.15: Opciones de acceso a información de amigo

5.3.8. Comentarios del muro

Otra información a la que puede acceder el usuario es el muro, donde se muestran los comentarios realizados por tu amigos, tus cambios de estado y las respuestas a esos comentarios. Como se observa en la figura 5.16 la aplicación muestra una lista de *feeds* o comentarios, así como las posibles respuestas. Estos *feeds* están formados por el ícono y el nombre del usuario que lo envío y el contenido del comentario. En caso de que sea una respuesta, se muestra el nombre del usuario seguido de “*respondio al comentario anterior...*”. Si el usuario pulsa la tecla de *volver atrás* o *salida* del dispositivo móvil se vuelve al lector de códigos QR.



Figura 5.16: Comentarios del muro

Si el usuario quiere subir un comentario a su muro o al muro de un amigo, simplemente tiene que pulsar el botón *Añadir nuevo comentario* y automáticamente le sale el reconocedor de voz. Éste simplemente capta las palabras que pronuncie el usuario y las traduce a texto que se envía al muro (véase figura 5.17). Es recomendable seguir las breves instrucciones que indica el reconocedor de voz, sugiere que se hable de forma clara, despacio y con un tono alto, de esa forma se asegura de que lo ha reconocido bien y por tanto escribe lo que realmente ha dicho.

5.3.9. Fotos

El último tipo de información que tiene para acceder el usuario en la red social, son las fotos. Lo primero que muestra la aplicación es un pantalla de ayuda con unas breves instrucciones para entender mejor el funcionamiento de la interfaz de las fotos (véase figura 5.18). Para pasar de la pantalla, sólo hay que pulsar la imagen que aparece en ella.

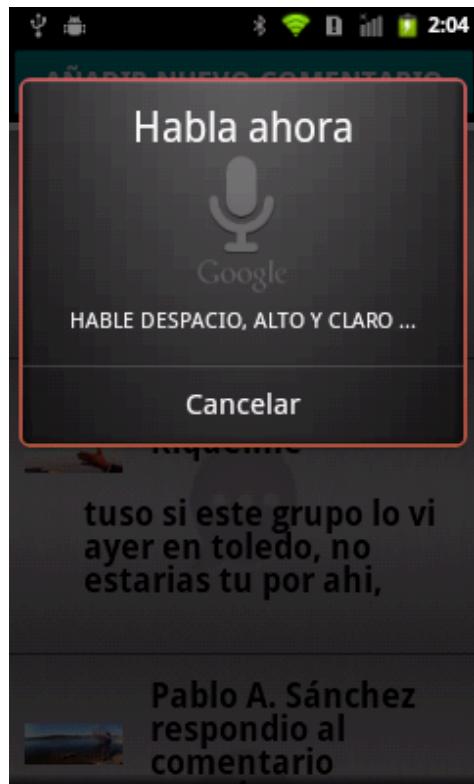


Figura 5.17: Subir un comentario (post)



Figura 5.18: Ayuda para la interfaz de las fotos

Una vez que el usuario se ha leído la ayuda para esta pantalla, éste puede ver sus fotos o las de un amigo. En la figura 5.19 se muestra las fotos de un amigo, por eso en esta pantalla no aparece el botón *Subir foto*, en el caso de que sean nuestras fotos si aparece dicho botón. Para pasar las fotos disponibles, el usuario debe arrastrar el dedo en los marcos de madera de la pantalla, si el dedo va hacia la derecha, se visualiza la foto anterior, si el dedo va hacia la izquierda, se visualiza la foto siguiente. Para salir de las fotos, el usuario debe pulsar la tecla de *volver atrás* o *salida* del dispositivo móvil con lo que vuelve al lector de códigos QR.



Figura 5.19: Fotos de un amigo

Esta pantalla permite la opción de poder hacer zoom en la foto, para ello el usuario debe pulsar dos veces en la zona que quiere aumentar y pulsar otras dos veces para volver al estado original. En la figura 5.20 se observa un zoom sobre la foto visualizada.

En el caso de que el usuario quiera subir una foto a su cuenta de red social, la aplicación permite dos opciones de carga de fotos. Cuando el usuario pulse el botón *Subir foto*, la pantalla muestra una ventana con dos opciones: cámara y galería (véase figura 5.21).

La opción cámara permite subir una foto a la red social sacandóla en ese mismo instante, la aplicación inicia la cámara del móvil y cuando el usuario haya realizado la foto, se procede a subirla, como se puede ver en la figura 5.22.

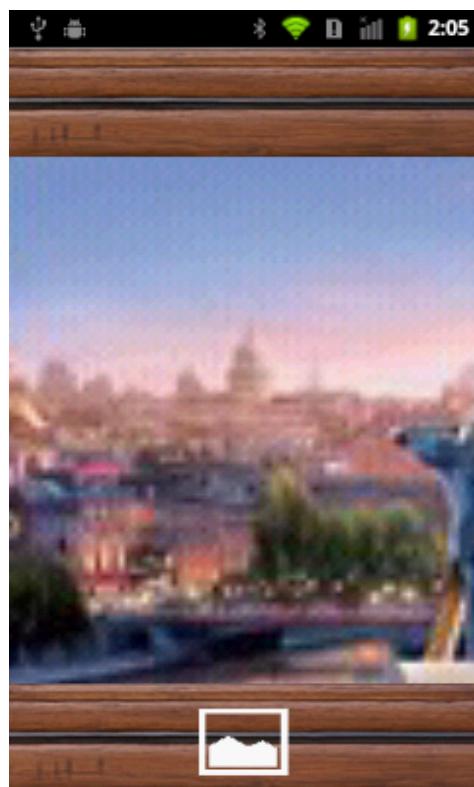


Figura 5.20: Zoom sobre una foto



Figura 5.21: Opciones para subir una foto



Figura 5.22: Cámara del dispositivo móvil

La segunda opción es la galería permite subir una foto a partir de las guardadas en el dispositivo móvil, para ello la aplicación inicia la galería del dispositivo y nos permite seleccionar una cualquiera. Una vez hecho esto se procede a subirla a la red social (véase figura 5.23).

5.3.10. Créditos

La última pantalla que tiene la aplicación y aparece siempre al finalizar la aplicación es la de créditos. En ella se muestra información de donde se ha creado la aplicación SocialFrame que se observa en la figura 5.24. Para salir de la aplicación, el usuario debe pulsar la tecla de *volver atrás* o *salida* del dispositivo móvil.



Figura 5.23: Galería de fotos



Figura 5.24: Créditos de la aplicación

5.4. Evaluación

Para la realización de la evaluación de una aplicación móvil se debe tener en cuenta el número de usuarios que evalúan como sus características más importantes [35]. Se ha comparado la aplicación SocialFrame con la aplicación móvil para Facebook, para ello se ha realizado un cuestionario de evaluación (véase Anexo C) en personas con diferentes características. El número de cuestionarios realizados ha sido siete y las características de las personas son las que se pueden ver en la tabla 5.3.

	Edad	Sexo	Estudios	Conocimientos tecnológicos	Uso de redes sociales
<i>Persona 1</i>	22	hombre	medios	medios	esporádico
<i>Persona 2</i>	25	mujer	superiores	bajos	frecuente
<i>Persona 3</i>	65	hombre	superiores	bajos	ninguno
<i>Persona 4</i>	59	mujer	bajos	bajos	ninguno
<i>Persona 5</i>	26	mujer	superiores	superiores	frecuente
<i>Persona 6</i>	26	hombre	superiores	superiores	frecuente
<i>Persona 7</i>	26	hombre	superiores	superiores	frecuente

Cuadro 5.3: Características de las personas

La edad de las personas es variada aunque se hubiera preferido hacerlo sobre personas dependientes o mayores de 60 años. Hay cuatro hombres y tres mujeres. Se construyen dos rangos de edad para evaluar: 20-50 y *más de 50* años. Cada una de estas personas ha realizado el cuestionario que está formado de la siguiente forma:

- Tareas específicas a realizar: login, subir una foto, etc.
- Valoración sobre la realización de las tareas en Facebook.
- Valoración sobre la realización de las tareas en SocialFrame.
- Comparativa de resultados.

Cada valoración realizada sobre Facebook o SocialFrame está formada por una serie de ítems que deben valorarse del 1 al 5 (1-Nada satisfactorio, 5-Totalmente satisfactorio). Ya que en los cuestionarios algunos de los ítems entre SocialFrame y Facebook, tienen el mismo valor, se van a mostrar aquellos que difieren entre sí y sean más importantes o relevantes. En la figura 5.25 se pueden ver los ítems que difieren respecto a la usabilidad percibida.

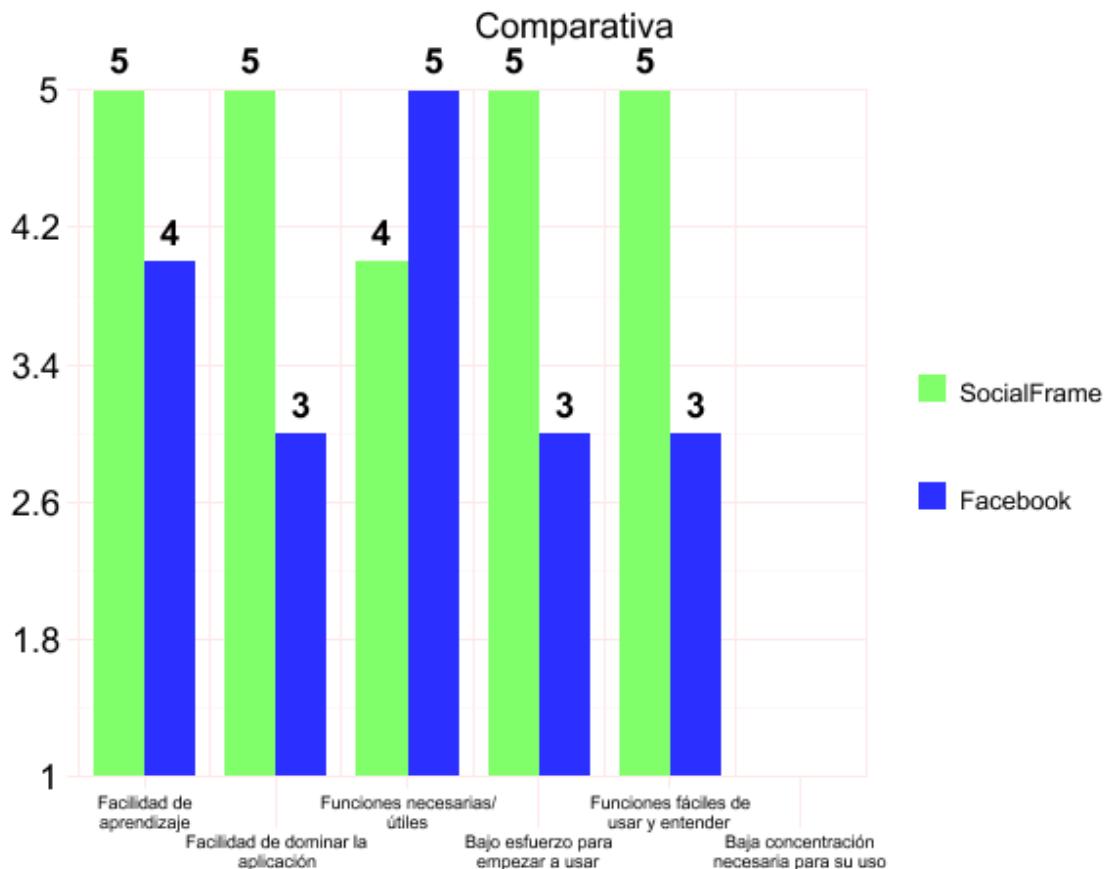


Figura 5.25: Usabilidad percibida entre Facebook y SocialFrame

Los valores que se muestran en la gráfica 5.25 son los más elegidos por los usuarios, es decir, puede haber personas jóvenes que crean que las funciones necesarias o útiles de Facebook sean mejores que las de SocialFrame, sin embargo, las personas mayores creen que las funciones necesarias o útiles son mejores en SocialFrame que en Facebook. Ya que el número de personas jóvenes es superior al número de personas mayores, estos ítems reflejan diferentes resultados dependiendo del grupo. En cambio en los ítems en que SocialFrame es mejor valorado que Facebook hay que destacar la gran aceptación a la hora de la facilidad de uso y aprendizaje de la aplicación evaluada.

Otra valoración importante entre SocialFrame y Facebook es la adecuación de estas herramientas al dispositivo móvil. Para ello se observa en la figura 5.26 una comparativa entre Facebook y SocialFrame de los ítems que difieren en el aspecto de la adecuación del dispositivo. Como se puede ver la valoración es mejor con la aplicación SocialFrame que con Facebook ya que el uso del dispositivo es más sencillo porque no se realizan enlaces entre las distintas secciones de la red social. Además la información mostrada es clara y sencilla y lo suficientemente visible para no tener que hacer zoom. En cambio Facebook muestra demasiada información por tanto es necesario hacer zoom para visualizarla.

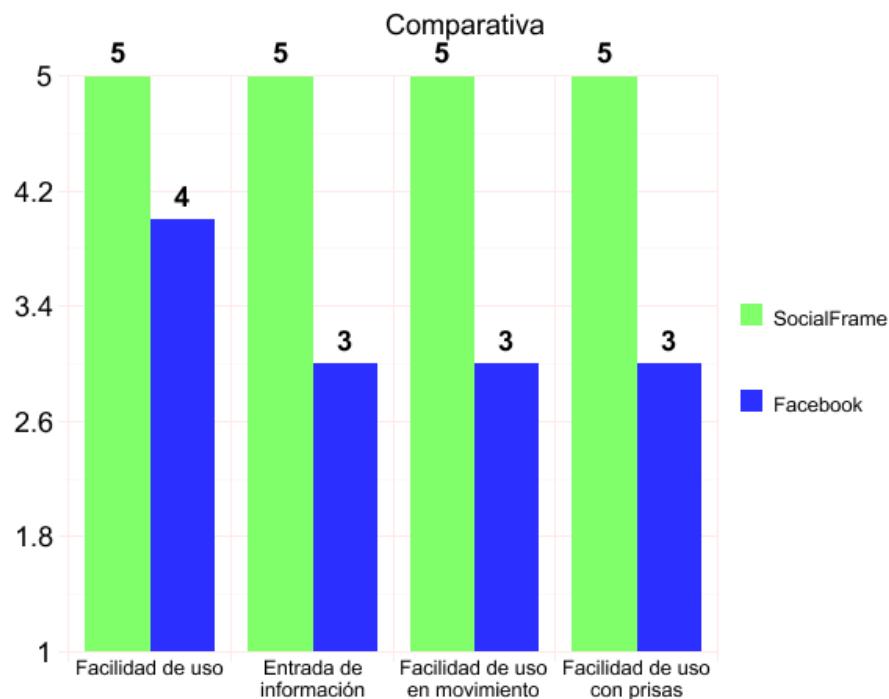


Figura 5.26: Adecuación al dispositivo entre Facebook y SocialFrame

Por último se realiza una comparativa entre estas valoraciones mostradas, por lo que en la figura 5.27 se puede ver el resultado de la valoración final de la aplicación SocialFrame respecto a la aplicación Facebook. Esta comparativa se hace para los dos grupos de edades que se han construido, ya que el sexo, el tipo de estudios y los conocimientos tecnológicos no se consideran relevantes porque la aplicación es fácil de usar e intuitiva y contiene mensajes de ayuda que permiten un aprendizaje rápido. La comparativa hace referencia a la satisfacción general de los usuarios sobre la aplicación SocialFrame, esto incluye tanto la realización de tareas en menos tiempo como la sencillez de hacer estas tareas.

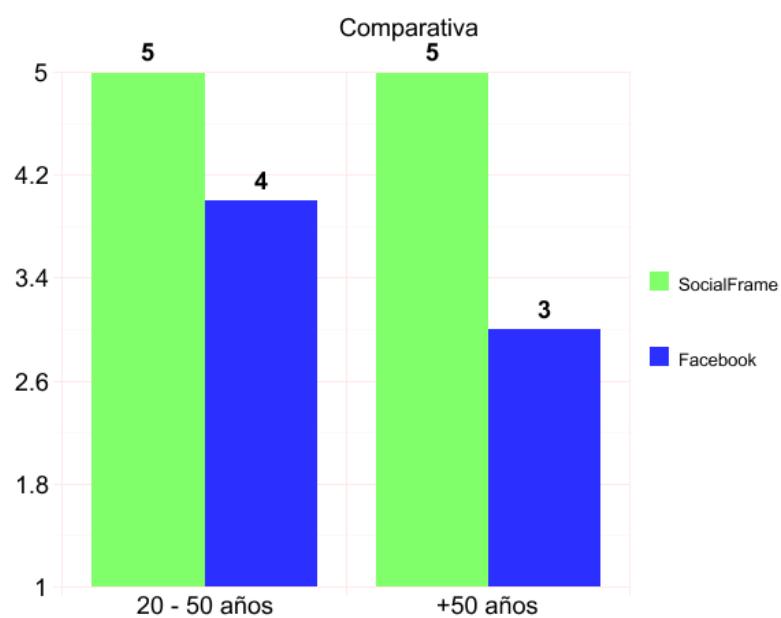


Figura 5.27: Satisfacción final

Capítulo 6

Conclusiones y propuestas

EN este capítulo se exponen una serie de conclusiones obtenidas durante el desarrollo del proyecto y se proponen propuestas para mejorar la aplicación *SocialFrame*.

6.1. Conclusiones

Como conclusión principal se plantea que la aplicación SocialFrame cumple los objetivos definidos al principio del proyecto. Estos objetivos cumplidos son los siguientes:

- Se han estudiado y analizado desventajas que presentan las personas mayores o dependientes en acceder y entender el uso de las redes sociales, de forma que se ha simplificado este uso de las redes sociales.
- Se han propuesto una serie de metáforas que consigan una relación directa entre un objeto de la vida cotidiana del usuario con uno o varios elementos de una red social. Además se han realizado metáforas a referencias cromáticas del elemento en la red social para que la persona ejercite la memoria y asocie dicho color a uno varios elementos en la red social. Estas metáforas se representan con los códigos QR que se generan.
- Se ha desarrollado una interfaz móvil invisible y sencilla para la aplicación. La interacción es llevada a cabo por el uso de códigos QR y metáforas que representan objetos cotidianos y su correspondencia con los posibles elementos de la red social. Además se han desarrollado técnicas de interacción para evitar complejos sistemas de navegación y simplificar la introducción de información y configuración de aspectos personales del usuario.
- Se han integrado servicios del dispositivo móvil que son útiles para la aplicación, entre ellos se encuentran: la cámara que permite hacer uso del lector de códigos QR y sacar fotos, el lector de códigos QR que permite obtener el contenido de dichos códigos, el

reconocedor de voz que permite transcribir el audio en texto y la conexión a internet, ya sea por 3G o por WiFi que nos da acceso a la Web y a las redes sociales.

- Se ha accedido a información almacenada en redes sociales de forma automática llamando a las API que proporcionan. Se ha obtenido aquella información que sea realmente relevante para un uso sencillo de la red social. La información obtenida de la red social es aquella que tenga permisos dados por dicha red social.
- Se ha conseguido una aplicación que permite adaptar otras redes sociales que no sean Facebook. La posibilidad de añadir redes sociales a la aplicación es gracias al uso de un sencillo patrón llamado *Adapter*.
- Se ha desarrollado una aplicación móvil que posee una serie de características muy diferentes a las que podría tener si fuera una aplicación de escritorio o una aplicación web. Este será el medio por el que los usuarios acceden al sistema, permitiéndoles realizar diversas operaciones. Se realiza a través de servicios web que permiten en cualquier instante acceder a esos datos, siempre que haya conexión a internet.

La única conclusión negativa que se puede sacar del desarrollo del proyecto, es no haber incluido la fase de registro en la red social. El principal problema fue que la API no permite tal acción, por lo que se hace complicado el desarrollo de esta tarea.

6.2. Propuestas

La aplicación SocialFrame ha cumplido los objetivos iniciales propuestos y es bastante completa, sin embargo, ésta puede mejorarse actualizando aspectos de la aplicación o aplicando otras líneas de investigación.

6.2.1. Mejoras en la aplicación

La aplicación SocialFrame puede mejorarse de la siguiente forma:

- Automatización de las pruebas.
- Mejora del rendimiento de la aplicación móvil.
- Permitir la visualización correcta en las diferentes resoluciones de pantalla.

- Inclusión de un lector de códigos QR propio.
- Añadir redes sociales accesibles para el usuario, es decir, a parte de Facebook poder acceder a Twitter o Google+ entre otras.

6.2.2. Líneas de investigación futura

Aplicar líneas de investigación futura a la aplicación SocialFrame implica mejorar o cambiar algunos conceptos iniciales del desarrollo. Estas líneas de investigación futura son las siguientes:

- En vez de utilizar códigos QR para acceder a la información de la red social, se podría implementar una interfaz acústica, es decir, el usuario a través de su voz indicaría que información de la red social quiere ver o modificar. Aplicar esta línea implica que el usuario en principio sea invidente.
- Se podría integrar realidad aumentada para mostrar la información de la red social en la vida real. Para ello el usuario tendría que enfocar con el dispositivo móvil hacia el objeto para mostrar sobre él la información requerida. Un ejemplo podría ser si el usuario al enfocar en el calendario ve los eventos disponibles.
- La utilización de NFC para poder recibir o añadir información de redes sociales es otra línea interesante de utilizar. El usuario con un dispositivo móvil que tenga tecnología NFC debe poder acceder a la información de una red social acercando dicho dispositivo a una etiqueta NFC que contendrá la información necesaria para realizar esta consulta a la red social.

Estas líneas de investigación futura pueden cambiar la forma de acceder a la red social por parte de personas mayores o dependientes. Además se pueden desarrollar interfaces de usuario para otros sistemas operativos móviles como iPhone o Windows Mobile. También se podría integrar un widget en el dispositivo móvil que permitiese informar de las actualizaciones en la red social elegida.

Bibliografía

- [1] Ambient Assisted Living Joint Programme. <http://www.aal-europe.eu>
- [2] Principales redes sociales móviles del mundo. <http://www.poderpda.com/editorial/las-principales-redes-sociales-móviles-del-mundo/>
- [3] Sitio de audiencias Web. <http://www.quantcast.com/facebook.com>
- [4] Sitio oficial códigos QR. <http://www.qrplanet.com>
- [5] Normativa de Proyecto Fin de Carrera. <http://webpub.esi.uclm.es/archivos/89/NormativaPFC2007>
- [6] Sitio oficial de Android. <http://www.android.com>
- [7] Lenguaje de programación Java. <http://www.java.com>
- [8] Sitio oficial de Android SDK. <http://developer.android.com/sdk/index.html>
- [9] Sitio oficial de ADT plugin. <http://developer.android.com/sdk/eclipse-adt.html>
- [10] Facebook Android SDK. <https://github.com/facebook/facebook-android-sdk>
- [11] Graph API Explorer. <https://developers.facebook.com/tools/explorer/>
- [12] Sitio oficial de OAuth 2.0. <http://oauth.net/2/>
- [13] Sitio oficial de Facebook Query Language <http://developers.facebook.com/docs/reference/fql/>
- [14] Sitio oficial de JUnit. <http://www.junit.org>
- [15] Libreía ZXing. <http://code.google.com/p/zxing/>
- [16] Sitio oficial de UML. <http://www.uml.org>
- [17] Sitio oficial de Eclipse. <http://www.eclipse.org>

- [18] Sitio oficial de XML. <http://www.w3.org/XML/>
- [19] Sitio oficial de GIT. <http://git-scm.com/>
- [20] Sitio oficial de Github. <https://www.github.com>
- [21] Sitio oficial de GIMP. <http://www.gimp.org.es/>
- [22] Sitio oficial de Visual Paradigm. <http://www.visual-paradigm.com/>
- [23] Sitio oficial de Microsoft Project. <http://www.microsoft.com/project/>
- [24] Diego López-de-Ipiña, Ignacio Díaz-de-Sarralde, Javier García-Zubia. *An Ambient Assisted Living Platform Integrating RFID Data-on-Tag Care Annotations and Twitter*. Journal of Universal Computer Science, vol. 16, no. 12 (2010).
- [25] Stefan Goetze, Niko Moritz, Jens-E. Appell, Markus Meis, Christian Bartsch, Jörg Bitzer. *Acoustic User Interfaces for Ambient Assisted Living Technologies*. Fraunhofer Institute for Digital Media Technology.
- [26] Thomas Kleinberger, Martin Becker, Eric Ras, Andreas Holzinger, Paul Müller. *Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces*. Fraunhofer Institute for Experimental Software Engineering.
- [27] Alberto García Lillo. *Sistema para la visualización de información a través de realidad aumentada en dispositivos móviles*.
- [28] Sitio oficial de TweetDeck. [http://www\(tweetdeck.com](http://www(tweetdeck.com)
- [29] Ana Hernández, Francisco Ibañez, Neftis Atallah. *SENIORCHANNEL: An Interactive Digital Television Channel for Promoting Entertainment and Social Interaction amongst Elderly People*. Third International Workshop, IWAAL 2011.
- [30] Pere Tuset, Juan Miguel López, Pere Barberán, Léonard Janer, Cristina Cervelló-Pastor. *Designing Messenger Visual, an Instant Messaging Service for Individuals with Cognitive Disability*. Third International Workshop, IWAAL 2011.
- [31] Javier Gómez Escribano, Germán Montoro Manrique. *aQRdate: Sistema para el recordatorio de actividades de la vida diaria a personas con daño cerebral adquirido* Septiembre 2011.
- [32] Ivar Jacobson, Grady Booch, y James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley Professional, 1999.
- [33] Sitio oficial del Object Management Group. <http://www.omg.org/>

- [34] Erich Gamma, Richard Helm, Ralph Johnson, y John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
- [35] Ramón Hervás, José Bravo, Jesús Fontecha. *Awareness marks: adaptive services through user interactions with augmented objects* January 2011.
- [36] Facebook Developers. <http://developers.facebook.com/>
- [37] API Google +. <https://developers.google.com/+api/>

ANEXOS

Anexo A

Creación de una aplicación Android en Facebook

N este anexo se plantea cómo crear una aplicación Android que haga uso de la red social Facebook. Antes de comenzar propiamente con la configuración de Android se registra la aplicación de Facebook. Para ello, Facebook Developers (véase figura A.1) [36] es una aplicación más de Facebook, que ofrece un API Key con el que se realizan las peticiones al servidor. Por ello, se accede a Facebook Developers y se le otorgan los permisos básicos, como toda aplicación.

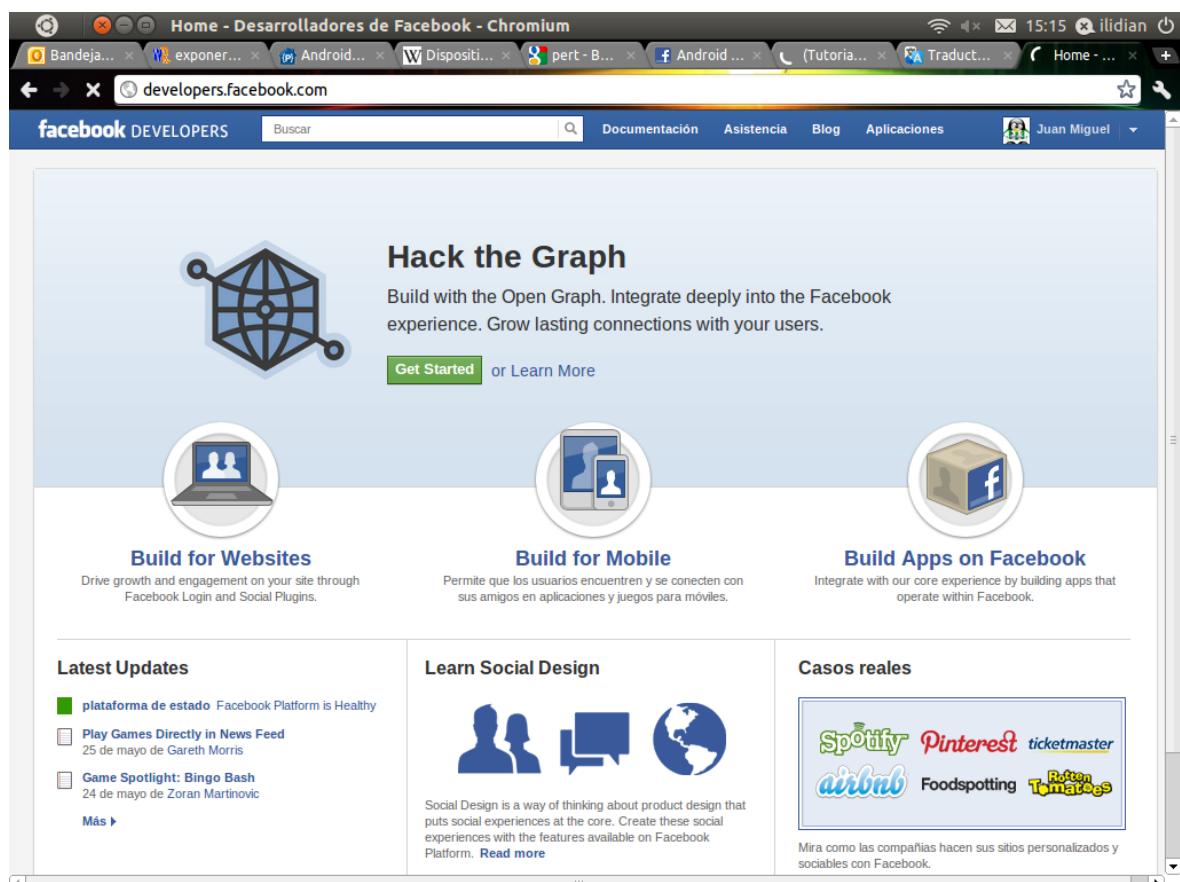


Figura A.1: Facebook Developers

Se selecciona la pestaña *Aplicaciones* y a continuación se pulsa la opción *Create New app* y acto seguido va a pedir la validación de nuestra cuenta usando el móvil o tarjeta de crédito. A continuación se ingresa el nombre de la aplicación, el captcha y listo, aplicación creada.

Apps ▶ AndroidSSO ▶ Basic

The screenshot shows the Facebook App Creation Wizard. At the top, there's a placeholder icon for the app and the name "AndroidSSO". Below it, fields for "App ID: YOUR_APP_ID" and "App Secret: YOUR_APP_SECRET" are shown, along with a "(reset)" link and a "(edit icon)" link. The main form is titled "Basic Info" and contains the following fields:

- App Display Name:** [?]
- App Namespace:** [?]
- Contact Email:** [?]
- App Domain:** [?]
- Category:** [?]

Below this is a section titled "Cloud Services" with a note: "Hosting URL: [?] You have not generated a URL through one of our partners ([Get one](#))".

The next section is "Select how your app integrates with Facebook", which lists several options:

- Website**: I want to allow people to log in to my website using Facebook.
- App on Facebook**: I want to build an app on Facebook.com.
- Mobile Web**: I have a mobile web app.
- Native iOS App**: I have a native iOS app.
- Native Android App**: I have a native Android app. This option is highlighted with a green border.
- Page Tab**: I want to build a custom tab for Facebook Pages.

At the bottom of the form is a blue "Save Changes" button.

Figura A.2: Página de creación de aplicación en Facebook

En la figura A.2 se puede ver la página donde se encuentra el “ID de la aplicación”, que es indispensable para poder acceder a la información de la red social Facebook. Acto seguido se procede a editar las características de la aplicación, se accede a la pestaña móvil y allí se selecciona aplicación nativa Android. Además hay que ingresar la clave hash, que es una firma de la aplicación. Se pueden modificar los demás campos (url, descripción, etc) y listo. El hash que se ingresa en la aplicación en Facebook se genera de la siguiente forma:

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/myKey.keystore | openssl sha1 -binary | openssl base64
```

Una vez creada la key hash se procede a ingresarla en la aplicación creada en Facebook (véase figura A.3), con esto se ha creado una aplicación Android en Facebook, ahora queda implementarla.

The screenshot shows the 'Basic Info' section of the Facebook App Settings. It includes fields for App Display Name (Android App), App Namespace, Contact Email (aryehs@gmail.com), App Domain, Category (Other), and Hosting URL. Below this, the 'Select how your app integrates with Facebook' section is shown, specifically the 'Native Android App' configuration. It has options for Configured for Android SSO (Enabled), Android Native Deep Linking (Disabled), and an input field for Android Key Hash containing 'PQA+0Q9I5LxdzKc7mmTKrHwFLwY='.

Android App

App ID: XXXXXXXXXXXXXXXXXX
App Secret: XXXXXXXXXXXXXXXXXXXXXXXXX (reset)
(edit icon)

Basic Info

App Display Name: [?] Android App

App Namespace: [?]

Contact Email: [?] aryehs@gmail.com

App Domain: [?] Enter your site domains and press enter

Category: [?] Other Choose a sub-category

Hosting URL: [?] You have not generated a URL through one of our partners (Get one)

Select how your app integrates with Facebook

Mobile Web

Mobile Web URL: [?] http://www.facebookmobileweb.com/hackbook/

Native Android App

Configured for Android SSO: Enabled Disabled

Android Native Deep Linking: Enabled Disabled

Android Key Hash: [?] PQA+0Q9I5LxdzKc7mmTKrHwFLwY=

Android Package Name:

Android Class Name:

Android Market URL:

Website Log in to my website using Facebook.

App on Facebook Use my app inside Facebook.com.

Native iOS App Publish from my iOS app to Facebook.

Page Tab Build a custom tab for Facebook Pages.

Save Changes

Figura A.3: Ingreso de la key hash

Anexo B

Añadir Google+ a la aplicación

En este anexo se expone como añadir una red social a la aplicación SocialFrame. Para realizar este proceso es necesario utilizar el patrón Adapter, que nos permite crear clases hijas (redes sociales específicas) a partir de una clase padre (red social genérica) que se puede ver en el fragmento de código 4.9.



Figura B.1: Google +

Se va a crear un ejemplo con la red social *Google +*, para ello primero se crea una clase hija llamada *GooglePlusNetSocial* que se puede observar en el fragmento de código B.1. Esta clase debe implementar los métodos de la clase padre, en el caso de que no los utilice, se implementan como vacíos.

La API de Google + [37] permite especificar qué campos deben ser devuelto en la respuesta HTTP. Esto reduce significativamente el tamaño de la respuesta y por lo tanto, reduce el uso de la red, el tiempo de respuesta de análisis, y el uso de memoria. Funciona tanto con JSON y XML. En el fragmento de código B.2 que se incluye en el método *showUser* de la clase *GooglePlusNetSocial*, que devuelve los datos de un usuario.

```

public class GooglePlusNetSocial extends SocialNetwork{

    public void login(SocialFrameActivity ac);

    public void showUser(InfoActivity ac, final User u, String owner);

    public void showFriendRequest(FriendsActivity ac);

    public void showFriends(FriendsActivity ac);

    public void acceptFriend(FriendsActivity ac, String uid);

    public void showFeeds(FeedActivity ac, String owner);

    public void postFeed(String msg, String owner);

    public void showMessages(MessagesActivity ac);

    public void showEvents(EventsActivity ac);

    public void showPhotos(PhotoActivity ac, String owner);

    public void uploadPhoto(PhotoActivity ac, Bundle b);
}

```

Listado B.1: Clase específica para GooglePlus

```

HttpTransport httpTransport = new NetHttpTransport();
JsonFactory jsonFactory = new JacksonFactory();

GoogleAccessProtectedResource requestInitializer = new
    GoogleAccessProtectedResource(accessToken, httpTransport,
        jsonFactory, clientId, clientSecret, refreshToken);

Plus plus = Plus.builder(httpTransport, jsonFactory).
    setHttpRequestInitializer(requestInitializer).build();

Person profile = plus.people().get("me").execute();
u.setName(profile.getDisplayName());
u.setPicuser(profile.getImage().getUrl());

```

Listado B.2: Ejemplo de mostrar usuario en Google +

Anexo C

Cuestionario de Evaluación

EN este anexo se puede observar el cuestionario para la evaluación final de la aplicación. Para poder realizar las pruebas de usabilidad, que nos proponen una serie de estadísticas para saber la satisfacción del usuario final con la aplicación, se crea un cuestionario de evaluación con una serie de datos personales y una lista de preguntas que se valoran del 1 al 5. Este cuestionario se puede observar en las figuras C.1 y C.2.

Cuestionario de la experiencia de usuario

Datos de población

Edad _____ Sexo: Mujer Hombre
 Estudios: Básicos Medios Superiores
 Conocimientos tecnológicos: Bajos Medios Altos
 Uso de redes sociales: Nunca Esporádico Frecuente

Tareas específicas

- T1: Login en la aplicación
 - T2: Añadir un comentario
 - T3: Revisar fotos de un amigo
 - T4: Subir una foto
 - T5: Revisar muro propio (Mientras se está andando)
 - T6: Saber de donde es un amigo
 - T7: Consultar mensaje privado
 - T8: Revisar si tiene eventos hoy
 - T9: Revisar si algún amigo cumple hoy los años (En el menor tiempo posible)
 - T10: Hacer zoom en una foto
-

Escala (Satisfacción): 1- Nada satisfactorio 5- Totalmente satisfactorio

Pretest (Facebook en dispositivo móvil)

Dimensión	Item	1	2	3	4	5
Usabilidad percibida	Facilidad de aprendizaje					
	Facilidad de dominar la aplicación					
	Confiabilidad					
	Adecuado para usar en movimiento					
	Ausencia de errores					
	Suficientemente ágil					
	Funcionalidades necesarias/útiles					
	Facilidad de navegación					
	Bajo esfuerzo para empezar a usar					
	Funciones fáciles de usar y entender					
	Baja concentración necesaria para su uso					
	Recomendable a otros					
Adecuación de las capacidades del dispositivo	Facilidad de uso con el dispositivo					
	Tamaño de la pantalla					
	Adecuación del dispositivo en movimiento					
	Entrada de información					
	Aspectos ambientales (poca luminosidad, reflejos, ruido)					
	Seguridad personal al usar en movimiento					
	Facilidad de uso en movimiento					
	Facilidad de uso con prisas					

Figura C.1: Cuestionario de evaluación

PosTest (Sistema evaluado)

Dimensión	Item	1	2	3	4	5
Usabilidad percibida	Facilidad de aprendizaje					
	Facilidad de dominar la aplicación					
	Confiabilidad					
	Adecuado para usar en movimiento					
	Ausencia de errores					
	Suficientemente ágil					
	Funcionalidades necesarias/útiles					
	Facilidad de navegación					
	Bajo esfuerzo para empezar a usar					
	Funciones fáciles de usar y entender					
	Baja concentración necesaria para su uso					
	Recomendable a otros					
Adecuación de las capacidades del dispositivo	Facilidad de uso con el dispositivo					
	Tamaño de la pantalla					
	Adecuación del dispositivo en movimiento					
	Entrada de información					
	Aspectos ambientales (poca luminosidad, reflejos, ruido)					
	Seguridad personal al usar en movimiento					
	Facilidad de uso en movimiento					
	Facilidad de uso con prisas					
Productividad percibida en las tareas (A)	Satisfacción en la eficacia en realizar tareas					
Comparativo con FaceBook	Uso mejora el desarrollo de las tareas					
	Uso mejora la motivación en las tareas					
	Uso mejora la satisfacción personal en las tareas					
	Uso mejor la fluidez en la realización de tareas					
	Las tareas se realizan más eficientemente					
	Permite realizar una tarea en menos tiempo					
	Permite realizar una tarea más fácilmente					
	Satisfacción con el resultado de las tareas					
Productividad percibida en las tareas (B)	Menos etapas en las tareas					
(Comparativo con Facebook)	Menos tiempo en pasar de una tarea a otra					
	Mejor acceso a la información					
	Ayuda a la toma de decisiones					
	Adecuación en tareas distribuidas o multiusuario					
	Adecuación a la coordinación y planificación					
	Ayuda a encontrar información adecuada					
	Ayuda a compartir información					

Figura C.2: Cuestionario de evaluación

Este documento fue editado y tipografiado con L^AT_EX
empleando la clase **arco-pfc** que se puede encontrar en:
https://bitbucket.org/arco_group/arco-pfc

[Respetá esta atribución al autor]

