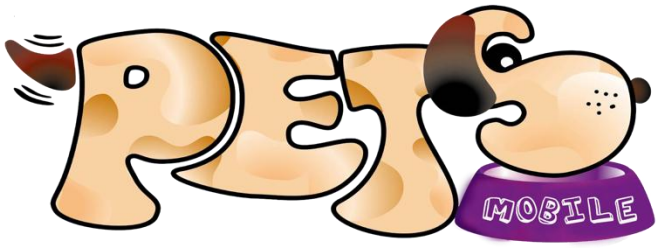
UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

<Grado en Ingeniería Informática en Ingeniería del Software>

Trabajo Fin de Grado

<PETSMobile: Red Social para dueños de mascotas>



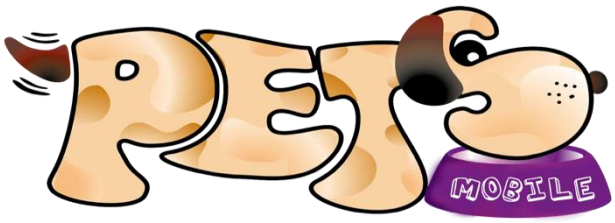
UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

< Grado en Ingeniería Informática en Ingeniería del Software>

Trabajo Fin de Grado

< PETSMobile: Red Social para dueños de mascotas>



Autor: <Juan Luis Collado Corraliza>

Tutor: <Juan Manuel Murillo Rodríguez>

Co-Tutor/es: <José Javier Berrocal Olmeda>

PRÓLOGO

El objetivo que tiene esta documentación es la elección, definición e implementación de la aplicación PETSMobile.

La elección de este proyecto para Trabajo Fin de Grado surge de una primera y una segunda reunión con el tutor de dicho Trabajo, Juan Manuel Murillo. En la primera reunión se me expone como idea para proyecto la utilización de un motor de eventos complejo en un dispositivo Android, con la finalidad de conseguir que el smartphone evalúe en función de cumplir varios eventos si debe mostrar o no las notificaciones Push enviadas por otro dispositivo. En una segunda reunión, se expone por mi parte la idea definitiva del proyecto explicado en esta documentación, “Una red social para dueños de mascotas”.

El objetivo fundamental de este proyecto es realizar una aplicación para dispositivos Android que llegue a ser utilizada por las personas en su vida diaria, con la finalidad de ayudarlas con las mascotas de las que sean dueños.

La realización de dicho proyecto me ha parecido una genial experiencia, ya que he conseguido realizar una aplicación funcional con los conocimientos adquiridos a lo largo de estos cuatro cursos en el Grado de Ingeniería Informática en Ingeniería del Software. Además, ha sido bastante útil para conocer la tecnología escogida para la realización de este proyecto. Por último, mencionar que pretendo seguir con ilusión en próximas actualizaciones y ampliaciones de esta aplicación, con el fin de ayudar a muchas personas, y sobre todo, ayudar a muchas mascotas.

Quisiera dar las gracias, desde aquí, a todos aquellos que me han apoyado en estos cuatro años de estudios, en especial, a mis padres y hermana, a mi pareja, amigos y a mi tutor de prácticas externas. Gracias a todos ellos por haberme apoyado y ayudado a su manera.

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Desde un tiempo hasta este momento, y muy probablemente siguiendo en el futuro, la tecnología está notando un gran avance. En la actualidad, la gran mayoría de las personas están conectadas a la tecnología y la informática.

Con la aparición al mercado de los teléfonos móviles inteligentes, los tan conocidos smartphones, y la llegada a la vida de las personas, hemos pasado de utilizar la tecnología, a estar continuamente conectados a ella. A día de hoy, un gran porcentaje de las personas es dueño de un dispositivo inteligente y está conectado a redes sociales en las que puede comunicarse con el resto de personas.

Posteriormente, comenzaron a utilizarse las notificaciones push segmentadas, es decir, notificaciones de una aplicación que llegan solamente a los dispositivos de los usuarios que cumplen una serie de requisitos, con la finalidad de no molestar a usuarios no interesados en dicha notificación y mantener la privacidad de los usuarios.

Además, con la desconsideración de un sector de las personas hacia los animales, los centros de acogida de mascotas han notado un incremento notable de animales que necesitan ser acogidos por parte de estas sedes actualmente. Muchas mascotas llegan también a estos centros por motivos de una pérdida no deseada de la misma por parte de su dueño, o una simple escapada del animal. Otro motivo que fomenta la llegada de esta idea al intento de un proyecto real es la imposibilidad de muchas personas a poder tener mascotas, por el hecho de no tener tiempo de pasear a estas, es decir, de no tener la vida adecuada que debe tener una mascota, o simplemente no poder realizar cualquier evento, como pueden ser viajes, cómodamente por no tener con quien dejar a sus mascotas para que estén bien atendidas en la ausencia de su dueño.

Por estos motivos mencionados anteriormente surge la idea final de crear este proyecto. El proyecto es una aplicación móvil para el sistema operativo Android, que consta de una red social en la que los usuarios podrán estar conectados entre sí pudiendo ayudarse entre ellos prestando atención y cuidado de las mascotas de los otros.

El desarrollo de este Trabajo Fin de Grado se compone de varias tareas o fases. En una primera tarea se acordó la idea principal del proyecto,

intentando estudiar las aplicaciones similares que ya pudieran existir, comprobando así que no había ninguna estrictamente relacionada con la temática del proyecto. La siguiente tarea fue la definición de los casos de uso de la aplicación, para definirla más concretamente así. A continuación, se llevó a cabo un diseño de la interfaz gráfica de usuario o mockup. Por último, la fase realizada ha sido la implementación del proyecto.

Tras la realización de todas estas fases, se ha creado la aplicación PETSMobile para dispositivos Android, una red social para dueños de mascotas, en la que el usuario podrá registrarse con sus datos personales y con sus mascotas, si es que las tuviera. Posteriormente todos los usuarios podrán poner en su aplicación que están dispuestos a ayudar a otros usuarios en un aspecto concreto. Así, un usuario puede poner que está dispuesto a pasear una mascota con unas características determinadas y un día determinado. Cualquier usuario, en posesión o no de mascotas, podrá ofrecer tantas ayudas como quiera (cuidado, paseo, encuentro y adopción) y de tantas mascotas como desee. Es decir, un usuario podrá estar dispuesto a cuidar mascotas con unas características determinadas en un periodo de tiempo, y también podrá comunicar que ha encontrado una mascota con unas características propias. Los usuarios que tengan mascotas, podrán solicitar ayuda de todos los tipos comentados anteriormente. Es decir, el usuario puede comunicar que necesita que adopten su mascota. Un usuario puede solicitar tantas ayudas de todas las tareas como quiera (cuidado, paseo, pérdida y adopción) y de tantas mascotas como desee. Cuando un usuario solicite ayuda le llegarán notificaciones a los usuarios que hayan puesto que están dispuestos a realizar una tarea del tipo solicitado y con mascotas con las características de las mascotas solicitadas para dicha ayuda. En caso de ser en periodos de fecha, como es el caso de cuidados o paseos, también tendrá que ser en esa disponibilidad diaria. Además, recibirán la notificación si están a una distancia determinada del usuario que solicita la ayuda. Por último, como toda buena red social, tendrá mensajería instantánea, para que los usuarios que reciban solicitudes de ayuda, puedan rechazarla o en caso de aceptarla, hablarlo mediante un chat personal entre ambos usuarios.

1. INTRODUCCIÓN

Hoy en día, vivimos conectados a la tecnología y transmitiendo información de unos a otros, por lo que decimos que estamos en la Era de la Tecnología y la Información. Debido a esto, surge la idea de realizar un proyecto para dispositivos móviles, ya que estos nos mantienen conectados permanentemente a la tecnología.

A lo largo de esta primera parte de la documentación, vamos a tratar algunos temas de la tecnología. Entre ellos, comenzaremos hablando de la evolución de la tecnología a lo largo de estos últimos años, explicando brevemente también el hecho de que para transmitir la información, necesitamos de internet, lo que comúnmente llamamos el Internet de las cosas. A continuación, comentaremos como ha ido avanzando la tecnología que se utiliza para esta aplicación, los dispositivos Android y la diferencia que existe entre los usuarios que utilizaban anteriormente estos dispositivos y los que realmente lo utilizan actualmente.

Posteriormente, veremos las motivaciones que me han hecho realizar este proyecto bajo la temática que tiene, además de para el sistema operativo Android, ya explicado en el punto anterior. El siguiente punto que explicaremos son los objetivos que tengo previstos para conseguir en la realización de este proyecto, así como los usuarios a los que va dirigida la aplicación realizada para el proyecto.

Para finalizar esta introducción, explicaremos brevemente la aplicación planteada para poder resolver todas estas cuestiones propuestas anteriormente.

1.1 EVOLUCIÓN DE LA TECNOLOGÍA

Desde hace unos años hasta esta parte, la tecnología ha ido evolucionando, y se estima que lo seguirá haciendo. Todo parece surgir con la desaparición de los teléfonos móviles de siempre, esos anticuados que solo servían para realizar llamadas, mandar mensajes, y algunos de ellos para hacer fotos o videos, y la aparición de los nuevos teléfonos inteligentes, los smartphones, esos que permiten realizar cualquier tipo de tarea, desde tener un GPS, hasta mantener conversaciones de mensajería instantánea, pasando por cámaras de video y fotografía con mayor calidad que una gran mayoría de las cámaras que solo sirven para realizar esa función. Pero si algo realmente nos ha hecho evolucionar con estos teléfonos inteligentes son las redes sociales, que hacen a las personas vivir conectadas permanentemente y transmitiendo información de unos a otros instantáneamente.
Evidentemente, para poder transmitir información, se necesita de una conexión a internet, solucionado en los smartphones mediante conexiones a redes Wifi y a redes de Datos en la tarifa del propio teléfono móvil. Por todos estas ventajas que nos proporcionan los teléfonos móviles inteligentes, las ventas han incrementado en gran número, hasta el punto de que hay más teléfonos móviles en el mundo que personas, ya que hay más de 7,9 mil millones de móviles para los aproximadamente 7,3 mil millones de personas que habitamos la Tierra.

ILUSTRACIÓN 1: INCREMENTO EN EL MUNDO EN VENTAS DE SMARTPHONES

Fuente: http://www.amic.media/media/files/file_352_1050.pdf

Junto a la aparición de los smartphones surgen las tablets, un dispositivo similar a los nuevos móviles, integrando además aplicaciones y tareas de un ordenador. Hasta ahora las tablets solo podían ejecutar una aplicación en un momento determinado, pero cabe mencionar que Apple ha avanzado y ha conseguido que la Tablet pueda ejecutar dos aplicaciones al mismo tiempo, viéndose ambas en la interfaz gráfica, como ya podían realizar los

ordenadores, y hasta ahora no estos dispositivos. La venta de tablets también se ha visto incrementada en los últimos años. Los sistemas operativos más utilizados para estas son Android e iOS. En la siguiente gráfica podemos ver el incremento mencionado y la comparativa entre los usuarios de cada sistema operativo.

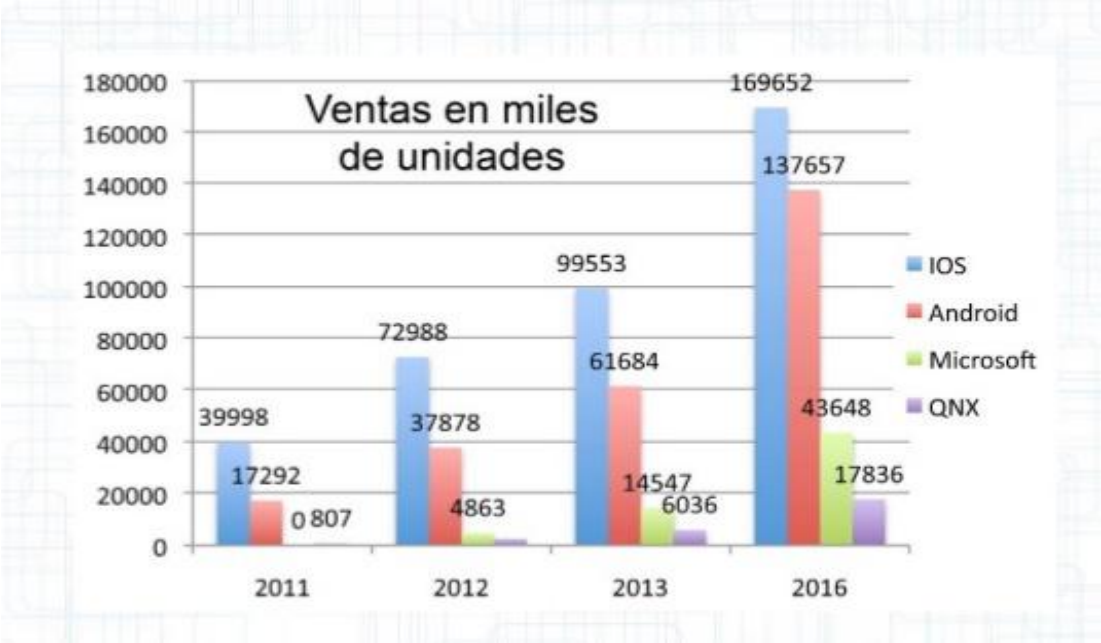


ILUSTRACIÓN 2: INCREMENTO EN EL MUNDO EN VENTA DE TABLETS

Fuente: http://es.slideshare.net/NachoRodriguez1/tablets-32917602

1.1.1 INTERNET DE LAS COSAS

Como ya hemos comentado anteriormente, para la transmisión de la información y la conexión entre las personas necesitamos de internet.

Esto también ha ido evolucionando de un tiempo hasta ahora, desde el punto de utilizarse únicamente en ordenadores bajo un navegador para visualizar páginas webs o mandar correos electrónicos, al punto de estar transmitiendo datos en cualquier aplicación y en un periodo de tiempo mínimo, incluso instantáneo. Actualmente también utilizamos el internet en la visualización de los programas de televisión, o incluso para ver distintas páginas y tener accesos a internet, con las conocidas SmartTV. Cabe también mencionar la utilización de internet en las videoconsolas, ya que hoy en día va incrementando también el juego entre dos personas que están a distancia, pero transmitiéndose datos para realizar una partida online.

El internet actualmente está mayormente utilizado en dispositivos como los smartphones y las tablets, en contraposición a la utilización que tenía antes únicamente en ordenadores y ordenadores portátiles. En la siguiente ilustración podemos ver la utilización de distintos dispositivos para acceder a internet y la evolución que está habiendo en cada uno de ellos.

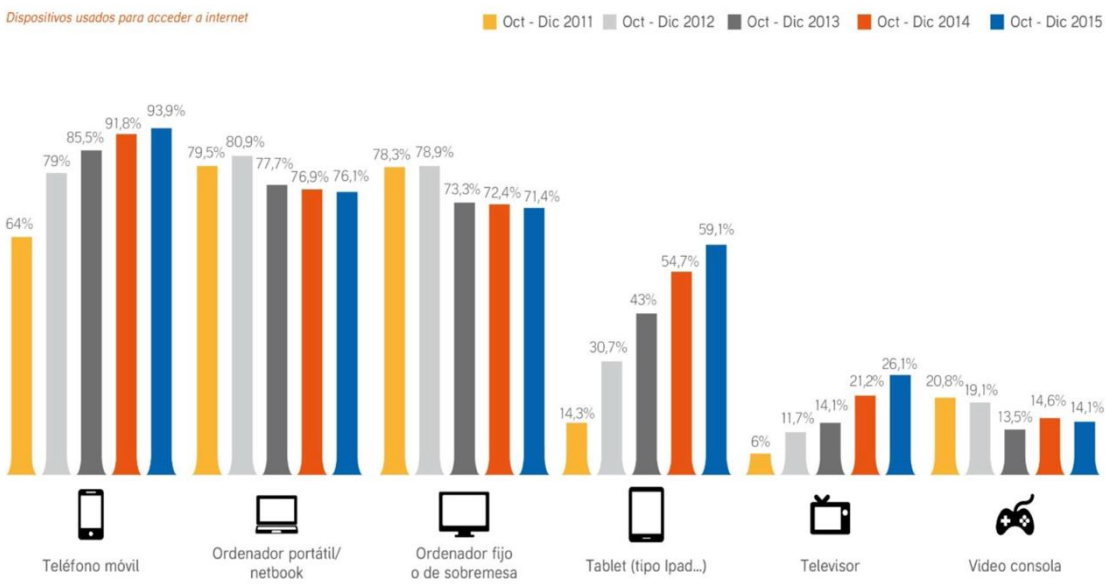


ILUSTRACIÓN 3: UTILIZACIÓN DE LOS DISPOSITIVOS PARA EL ACCESO A INTERNET

Fuente: http://www.amic.media/media/files/file_352_1050.pdf

Por último, mencionar que el acceso a internet en los dispositivos más utilizados, los smartphones, para estar permanentemente conectados se realiza mediante el consumo de datos móviles, contratados en la tarifa del consumo del dispositivo. Además, con finalidad de no gastar la tarifa de datos establecida, tenemos la conexión a las redes wifi.

En la utilización de la aplicación construida para este proyecto es de vital importancia la conexión a internet, ya que para poder conectar unos usuarios con otros, ya sea mediante notificaciones o mensajes, tienen que hacerlo mediante internet.

1.1.2 ANDROID

La aplicación realizada para este Trabajo Fin de Grado ha sido creada únicamente para el sistema operativo Android por diversos motivos. A continuación, explicaremos la claras ventajas que tiene dicho sistema con respecto a los otros que hay en el mercado, para así justificar la elección del mismo para este proyecto.

Android es un sistema operativo de código abierto (open-source) y está basado en Linux. Los smartphones que tiene Android son más fácil de utilizar que los que tienen otros sistemas operativos, además de la facilidad que da en aspecto de construcción a los desarrolladores al ser código abierto y poder utilizar muchas librerías previamente construidas por Google, o incluso por otros desarrolladores.

Este sistema operativo es utilizado por gran cantidad de dispositivos como smartphones, tablets, televisores, relojes inteligentes, incluso en automóviles.

Android inicialmente fue desarrollado por Android Inc. Respaldado por Google, pero más tarde, en el año 2005 paso a pertenecer a Google. Desde entonces, las ventas de dispositivos con este sistema han incrementado notablemente.

A día de hoy, los dispositivos móviles con Android son los más vendidos del mercado, superando al resto de sistemas operativos juntos. En el siguiente diagrama de sectores podemos ver como supera al resto de sistemas, con un 82% de todos los smartphones mundiales.

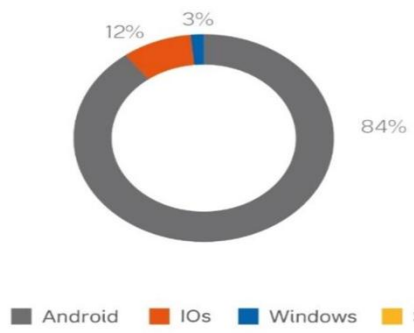


ILUSTRACIÓN 4: COMPARATIVA DE LOS SISTEMAS OPERATIVOS EN SMARTPHONES

Fuente: http://www.amic.media/media/files/file_352_1050.pdf

Al tener tantas ventas y tantos dispositivos en el mercado, es un sistema que se ve obligado a actualizarse muy continuamente, para poder adaptarse permanentemente a todos los dispositivos que lo utilizan y a las nuevas tecnologías que estos van añadiendo. Tanto es así, que existen gran cantidad de versiones de Android. En la siguiente ilustración podemos ver información sobre las versiones que han aparecido hasta el día de hoy.

Nombre código	Número de versión	Fecha de lanzamiento	Nivel de API
Apple Pie	1.0	23 de septiembre de 2008	1
Banana Bread	1.1	9 de febrero de 2009	2
Cupcake	1.5	27 de abril de 2009	3
Donut	1.6	15 de septiembre de 2009	4
Eclair	2.0–2.1	26 de octubre de 2009	5–7
Froyo	2.2–2.2.3	20 de mayo de 2010	8
Gingerbread	2.3–2.3.7	6 de diciembre de 2010	9–10
Honeycomb[1]	3.0–3.2.6	22 de febrero de 2011	11–13
Ice Cream Sandwich	4.0–4.0.4	18 de octubre de 2011	14–15
Jelly Bean	4.1–4.3.1	9 de julio de 2012	16–18
KitKat	4.4–4.4.4, 4.4W–4.4W.2	31 de octubre de 2013	19–20
Lollipop	5.0–5.1.1	12 de noviembre de 2014	21–22
Marshmallow	6.0–6.0.1	5 de octubre de 2015	23
Nougat	7.0 - 7.1	22 de agosto de 2016	24 - 25

ILUSTRACIÓN 5: VERSIONES DE ANDROID

Fuente: https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android

Por la facilidad a la hora de desarrollar aplicaciones en este sistema operativo es el sistema elegido por muchos desarrolladores.

Para ver el porque Android tiene mayores ventas y es elegido por más usuarios y más desarrolladores, exponemos brevemente algunas de las ventajas de Android:

- **Software libre**: Android es un sistema libre de código abierto, lo que permite que un desarrollador pueda utilizar el código de otros desarrolladores para optimizarlo, o para realizar modificaciones y conseguir un programa nuevo y distinto, pero aprovechando lo que le vale del programa anterior. Debido a esto, la mayoría de las

aplicaciones son de descarga gratuita, lo que permite al usuario de Android tener una mayor cantidad de programas o juegos y una mayor variedad de tipos de programas o juegos de una forma gratuita.

- **Comunidad de desarrolladores**: Este sistema operativo tiene la mayor comunidad de desarrolladores en el mundo. Incluso Google proporciona cursos para el aprendizaje del desarrollo para Android. Además, los desarrolladores tienen la posibilidad de aprender mediante diversos foros y páginas webs dedicadas única y exclusivamente a Android.

- **Costes**: Los dispositivos con este sistema operativo, en su gran mayoría tienen una relación calidad precio muy asequible y bastante más económica que el segundo sistema operativo más utilizado, iOS.

- **Sistema multitarea**: Android es capaz de gestionar más de una aplicación abierta al mismo tiempo, controlándolas todas a la vez, aunque dejando suspensas las que no esté utilizando en ese preciso momento el usuario, consiguiendo así una administración correcta de la memoria del dispositivo.

1.2 MOTIVACIONES Y OBJETIVOS

En este apartado de la documentación vamos a comentar los motivos que me han llevado a elegir este proyecto para la realización de mi Trabajo Fin de Grado, para explicar posteriormente los objetivos que pretendo conseguir con este proyecto. Para finalizar, voy a tratar brevemente los usuarios a los que va dirigida la aplicación realizada para el proyecto.

1.2.1 MOTIVACIONES

La principal motivación que me ha llevado a la realización de una red social para dueños de mascotas es, al ver la gran cantidad de animales que no tienen dueño y ver el incremento que hay actualmente en los centros de acogidas de animales, intentar conseguir ayudar a mi manera a todos los animales que sufren esta situación. Aunque la aplicación es para que las

personas se ayuden entre sí, el motivo real es la ayuda a las mascotas.

Además de la motivación principal, me conducen otra serie de motivaciones a intentar realizar este proyecto. A continuación, expongo brevemente los motivos que me han ayudado a elegir esta temática:

- En primer lugar, me motivó el pensar que, con el **amor que tengo a los animales** y la pena que me da ver la situación de dejadez en que se encuentran gran parte de ellos, podría ayudarles a mi manera, para que deje de existir la situación que hay, o al menos, pueda disminuir en gran porcentaje.

- En segundo lugar, intentar que **las personas se ayuden unas a otras**, ya que la mayoría de las personas estamos dispuestas a ayudar al resto, pero muchas veces necesitamos de motivaciones o simplemente no sabemos la forma correcta en que podemos hacerlo. Bajo mi punto de vista, las personas tendrán así una forma en que pueden ayudar al resto.

- En tercer lugar, al investigar en el mercado de aplicaciones, observé que **no había ninguna aplicación con la misma temática**, es decir, que sirviese para ayudar a las mascotas, por lo que me motivó el pensar que podría ser la primera del mercado.

Con respecto a la tecnología elegida, cabe destacar las siguientes motivaciones:

- En primer lugar, decidí que fuese una **aplicación móvil**, ya que actualmente prácticamente todas las personas del mundo tiene un teléfono móvil inteligente.

- En segundo lugar, tras decidir que fuese una aplicación móvil, había que decidir para que sistema operativo. El sistema operativo elegido fue **Android** ya que, como ya hemos comentado, es mucho más sencillo a la hora de desarrollar.

- En tercer lugar, yo había tenido una asignatura en la que habíamos aprendido una parte de Android y para complementar, me escogí las prácticas externas en una empresa que se dedicaba al desarrollo móvil y estuve dedicándome plenamente a este sistema operativo.

- En cuarto lugar, **podrá ser utilizada por una gran mayoría de los usuarios**, ya que Android, como ya hemos visto, es el sistema operativo más vendido del mercado en contraposición con el segundo sistema operativo, iOS.

1.2.1.1 MOTIVACION PRINCIPAL

Como ya he comentado anteriormente, el motivo principal que me llevó a hacer un proyecto con esta temática es la situación actual que sufren una gran cantidad de animales, al ser abandonados y llegar a centros de acogida, perreras, o incluso llegar a la muerte.

Podemos ver esto mediante datos reales, al saber que en España, en el año 2014 un total de 106.781 perros y 33.410 gatos fueron recogidos por las protectoras. Con estos datos podemos observar que un perro o un gato son abandonados cada 5 minutos en nuestro país.

Según estudios, la adopción es una de las últimas opciones cuando se decide tener un animal de compañía en el hogar, estando por encima la compra de mascotas o los regalos por parte de otras personas.

Los datos recogidos en 2015 permanecen estables, o siendo optimistas, disminuyeron en una muy pequeña medida. En ese año, se abandonaron más de 137.000 animales, de los que fueron un total de 104.501 perros y 33.330 gatos.

Nuestro país, se encuentra entre los países de Europa con mayor número de abandono animal. El perro es el animal más abandonado, por lo que es una de las mascotas en que está basada la aplicación.

ILUSTRACIÓN 6: PAISES CON MAYOR TASA DE ABANDONO DE PERROS EN EUROPA

Fuente: https://es.statista.com/grafico/6078/espana-quinto-pais-con-mas-perros-callejeros-de-europa/

Entre las principales causas del abandono de mascotas se encuentra la llegada de periodos vacacionales en los que muchas personas deciden viajar y abandonar sus mascotas por no poderlas llevar de viaje, por lo que con la aplicación podrían solicitar que algún otro usuario les cuidase a sus mascotas por un periodo de tiempo en el que se encuentran de vacaciones.

Otra de las principales causas es la falta de tiempo para pasear y cuidar a la mascota, por lo que también la aplicación da opción a solicitar ayuda para que otro usuario pueda pasear tus mascotas o cuidarlas.

Por último, una causa del abandono de mascotas es la pérdida involuntaria de la misma por parte del dueño, por lo que si solicitamos ayuda para que quien encuentre la mascota pueda devolvérnosla, evitando así que muchas mascotas lleguen a centros de acogida por una simple pérdida.

Al tener en la aplicación la opción de dar en adopción, un usuario podrá ofrecer su mascota para que si un usuario está dispuesto a adoptarla no tenga que ir ésta a un centro abandonada, pudiendo solucionar así gran parte del abandono animal.

1.2.2 OBJETIVOS

Como objetivo general a conseguir con este proyecto podemos destacar la creación de una aplicación en formato de red social, en la que los usuarios puedan ayudarse entre sí, prestándose servicios y ayudas a las mascotas de los otros usuarios.

Para solucionar todas las causas que me motivaron a elegir este Trabajo Fin de Grado, las cuales han sido explicadas en el apartado 1.2.1.Motivaciones, surgen los siguientes objetivos a intentar cumplir:

- Construir una red social, en la que los usuarios puedan solicitar ayuda de las siguientes formas:
    - **Cuidado**: Necesidad de que un usuario cuide a nuestra o nuestras mascotas, eligiéndose para ello las mascotas implicadas, dentro de las que tenga el usuario registradas, el día de inicio del cuidado y el día final del cuidado.
    - **Paseo**: Necesidad de que un usuario pasee a una o varias de nuestras mascotas. Para solicitar esta ayuda tenemos que elegir los animales que necesitamos que nos paseen y el día del paseo.

  - **Pérdida o encuentro**: Cuando un usuario pierda una o varias de sus mascotas, podrá solicitar ayuda, señalando que mascota ha sido la pérdida, intentando recuperarla de esta forma.
  - **Adopción**: Si un usuario ya no está dispuesto a seguir teniendo alguna o algunas de sus mascotas, por los motivos que tuviera, podrá ofrecerlas en adopción, en lugar de abandonarlas o llevarlas directamente a un centro de acogida.

- Cuando un usuario solicite ayuda, la aplicación solo notificará a las personas que estén inicialmente dispuestas a ayudar de dicha forma, evitando molestar a las personas que no estén interesadas en tal notificación. Mejoramos la privacidad del usuario así. Para solucionar esto, en la aplicación se considera que un usuario está dispuesto a ayudar de esta forma cuando crea un servicio del mismo tipo pero ofreciendo ayuda, en lugar de solicitarla, y además dicha ayuda es ofrecida para un animal con las características iguales a las de alguna de las mascotas para las que el usuario notificador solicita la ayuda. Además, si se trata de cuidado, el rango de los días de inicio y final de quien ofrece la ayuda debe ser mayor o igual que el rango de la solicitud de ayuda. En caso de ser paseo, debe coincidir el día del paseo. Por tanto, un usuario ofrecerá ayuda de algún tipo, y para animales con unas características establecidas, y solo recibirá notificaciones de quien solicite ayuda de ese tipo y para alguna mascota con esas características. Además solo llegará la notificación si, en caso de necesitar fechas, estas son aceptables para el rango o fecha seleccionada.

- La red social permitirá que los usuarios puedan contactar entre sí mediante mensajería instantánea, siempre y cuando alguno de los usuarios haya recibido una notificación del otro, o ya hayan mantenido alguna conversación anterior, lo que implica que haya habido alguna notificación de uno a otro. Esto controla que en la red social no podamos hablar a usuarios que no están interesados en ayudarnos.

- El dispositivo con esta aplicación ahorrará en batería, ya que no estará enviando datos continuamente al servidor para que éste controle si debe enviarnos la notificación, como lo realizan otras aplicaciones, si no que él se encargará de enviarlas a los móviles que cumplan los requisitos, pasando el control al propio dispositivo de gestionar si debe o no mostrar la notificación.

1.2.3 USUARIOS OBJETIVO

En este apartado, vamos a ver los usuarios a los que va dirigida la aplicación realizada para el proyecto. Los usuarios pueden tener cualquier edad, ya que cualquier persona puede necesitar de la ayuda de otras personas para tratar a sus mascotas, o por el contrario, puede ofrecerla. Procedo a describir brevemente los perfiles a los que va dirigidos:

- **Usuario sin mascotas**: Podrá únicamente ofrecer su ayuda a otros usuarios, ya que al carecer de mascotas registradas no podrá solicitar ayuda para ningún animal.

- **Usuarios con mascotas**: Podrá solicitar ayuda de todos los tipos explicados anteriormente y para cada una de sus mascotas. Además, también podrá ofrecerla en las ocasiones que lo desee.

1.2.3.1 ESCENARIOS

A continuación, pondremos unas situaciones imaginarias que podrán servirnos para ver el comportamiento de PETSMobile ante los distintos usuarios de la aplicación:

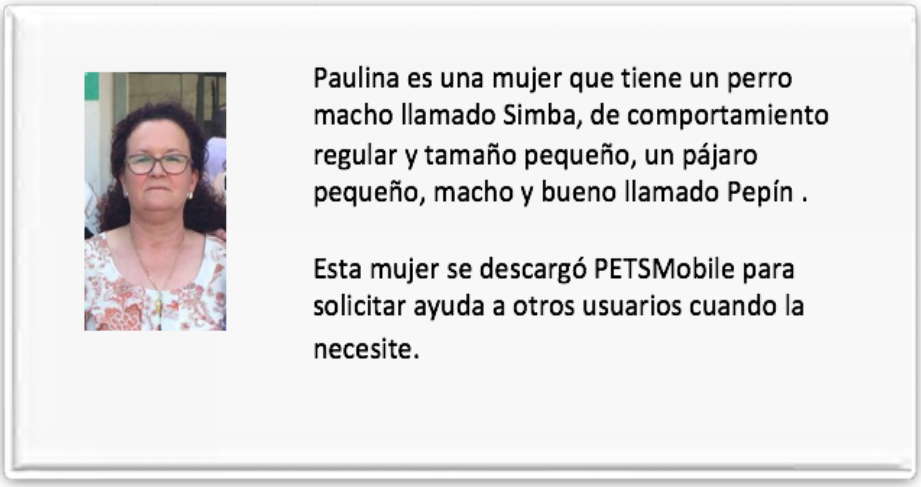
- **Usuario 1:**

Carmen es una chica que tiene una perrita llamada Zara, de comportamiento bueno y tamaño pequeño y un gato llamado Versacce de comportamiento bueno, tamaño pequeño y macho.

Esta chica se descargó PETSMobile para solicitar ayuda a otros usuarios cuando la necesite, ya que viaja mucho.

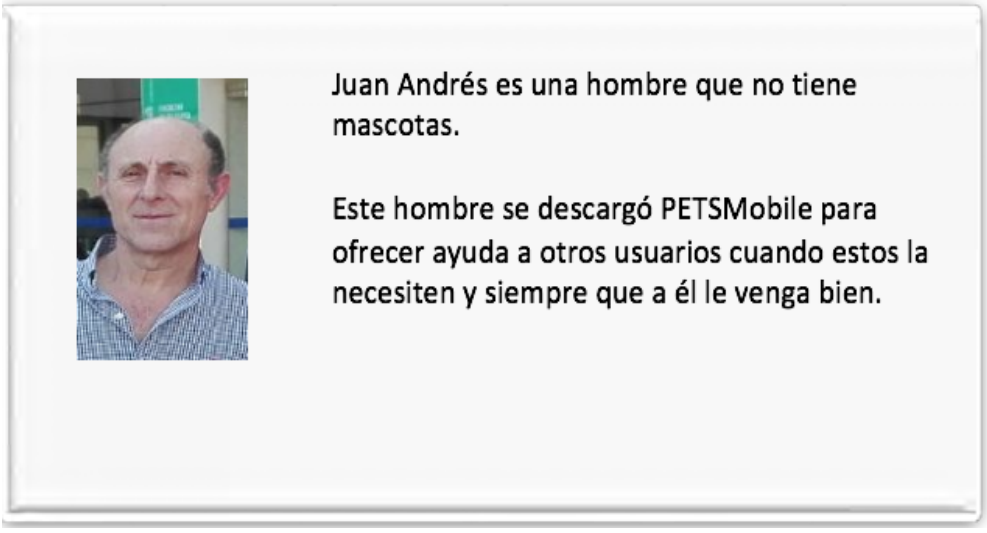
Escenario 1: Carmen necesita que alguien cuide su perrita Zara y a su gato Versacce desde el día 22 de Diciembre de 2016 al día 30 de diciembre de 2016, porque estará de vacaciones navideñas y no podrá llevarla en el viaje. Con PETSMobile podrá enviar una notificación a todas aquellas personas que estén dispuestas a cuidar un perro de estas características y durante esas fechas.

- **Usuario 2:**



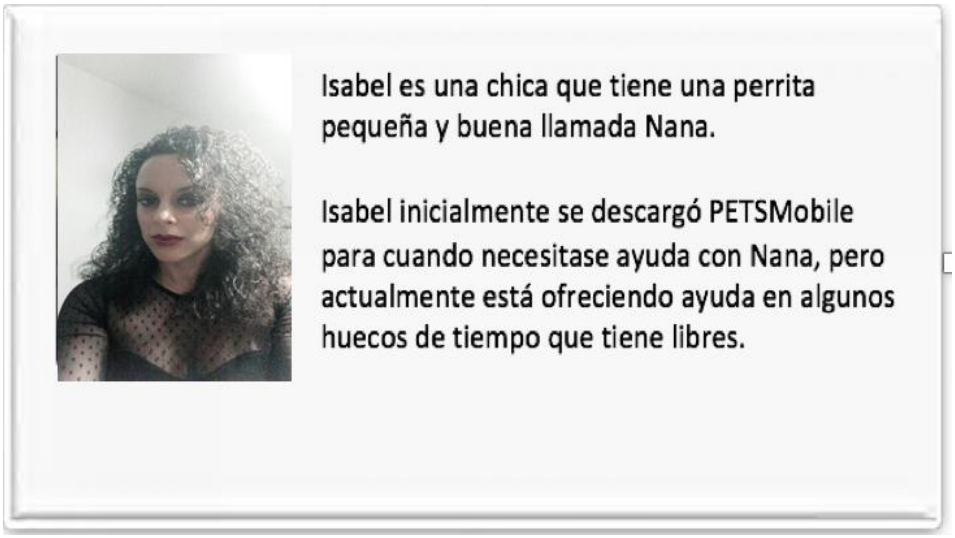
Escenario 2: Paulina estaba tan tranquila limpiando la jaula de su pájaro, cuando por una equivocación abrió la puerta y ha perdido al agapornis Pepín. Mediante PETSMobile, Paulina ha notificado que lo ha perdido, para que si alguien lo encuentra, no dude en contactar con ella, a la vez que lo busca por las zonas cercanas a su casa. Si no tuviese la aplicación, únicamente podría centrar sus esfuerzos en buscar al animal por su cuenta.

- **Usuario3:**



Escenario 3: Juan Andrés se encuentra a 1000 metros de distancia del lugar donde Paulina envió la notificación y además ha encontrado un pájaro de características iguales al de Paulina y lo había puesto en PETSMobile. Al recibir la notificación, decide aceptar y hablar a Paulina para cerciorarse si, efectivamente y con suerte, él ha encontrado el pájaro de Paulina. Hablando por el chat comprueban que si ha habido suerte y quedan en un punto cercano para devolvérselo.

- **Usuario 4:**



Escenario 4: Isabel se encuentra a 1300 metros de distancia del lugar donde Carmen envió la notificación y además había puesto que está dispuesta a cuidar perros de tamaño pequeño y comportamiento bueno, que además sean hembra (para que no haya percances con Nana si fuese un macho), y en rango de fechas mayor al que Carmen necesita, por lo que recibe la notificación. Decide aceptar y hablar a Carmen para decirle que está dispuesta a cuidar su perrita. Carmen le dirá que también necesita que cuiden de su gato, a lo que Isabel dirá que no está dispuesta a juntar gatos con Nana. Isabel cuidará de la perrita de Carmen, pero Carmen tendrá que esperar a que otro usuario le hable para cuidar su gato.

2. ESTADO DEL ARTE

En este segundo capítulo de la documentación se muestra la investigación y estudio de algunas aplicaciones ya existentes en el mercado, relativamente similares a la aplicación que se va a desarrollar para el Trabajo Fin de Grado, así como ver cuáles son sus ventajas y desventajas. Fijándonos en estos pros y contras, trataremos de obtener que cosas no debemos hacer similares, para no caer en dichos errores, tratando además de simular las cosas buenas, para tener los beneficios y ventajas de dichas aplicaciones.

Para dicha investigación he tenido en cuenta aplicaciones que tratan temas referidos a los animales, aunque actualmente no hay ninguna aplicación con la temática igual, además de algunas redes sociales, para ver la interactuación entre los usuarios en una red social, ya que son los aspectos en los que se basa PETSMobile.

2.1 REDES SOCIALES

La primera red social que he tenido en cuenta a la hora de llevar a cabo esta investigación fue Instagram. A continuación estudie las ventajas y desventajas de la tan conocidísima red social de mensajería instantánea, WhatsApp. Para finalizar, estudié los pros y contras de otra aplicación de mensajería instantánea, aunque menos utilizada, Telegram.

A continuación, vemos cada una de las aplicaciones estudiadas para esta investigación.

2.1.1 INSTAGRAM

Instagram es una red social basada en la subida de fotos y videos por parte de un usuario, las cuales podrán ser visualizadas y comentadas por los usuarios que tenga como amigos, además de por el resto de usuarios si la cuenta no está privada.

También posee una sección de chats del usuario con el resto de usuarios, en la que podrá hablar de forma privada.

A continuación, adjunto una serie de capturas de pantalla de la aplicación que estamos investigando, para ver exactamente las situaciones que se dan en dicha red social:

En esta imagen podemos ver la sección de chats de la red social en cuestión. Vemos como para cada usuario aparece su imagen de perfil, la última conexión y el último mensaje.

ILUSTRACIÓN 7: SECCIÓN DE CHATS EN INSTAGRAM

En esta ilustración podemos observar los datos del perfil del usuario. Aparece también su foto de perfil y el resto de datos que el usuario rellenó cuando se registró en dicha red social.

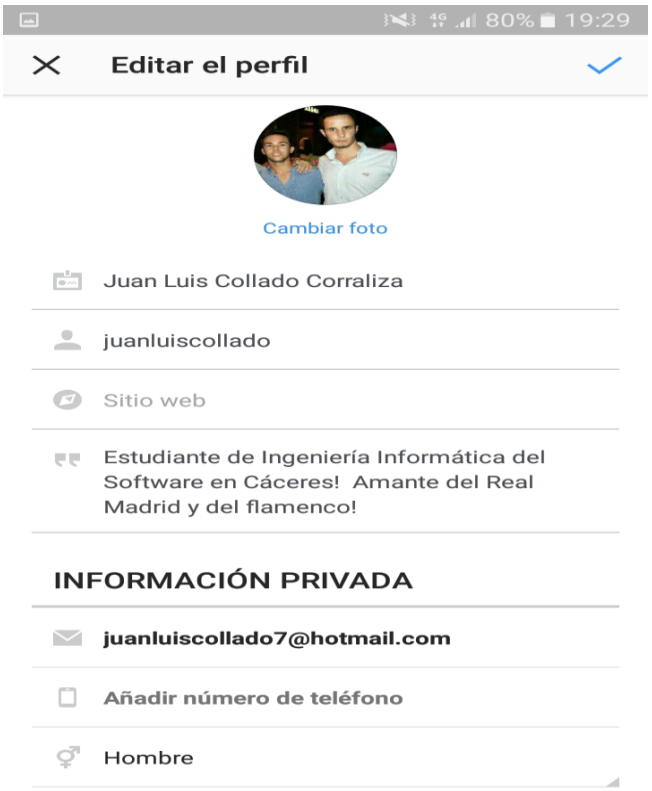


ILUSTRACIÓN 8: PERFIL DEL USUARIO EN INSTAGRAM

Por último, podemos ver en la siguiente imagen la pantalla de inicio de la aplicación investigada, en la que he basado el diseño de mi pantalla de inicio de la aplicación.

ILUSTRACIÓN 9: PANTALLA DE INICIO EN INSTAGRAM

Tras ver el diseño que tiene dicha red social, pasamos a ver las ventajas y desventajas de la misma.

Ventajas de Instagram:

- Acceso rápido a todas las pantallas principales mediante la barra de navegación de abajo.
- Facilidad para la modificación de los datos del usuario.
- Facilidad para mantener una mensajería privada con otro usuario.
- Posibilidad de vincular Instagram con otras redes sociales.

Desventajas de Instagram:

- Existencia de perfiles falsos.
- Demasiada dificultad para ver algunas de las pantallas secundarias de la aplicación.
- Cualquier usuario puede hablar a otro, salvo que éste lo tenga privado.

Conclusión:

Fijándonos en las ilustraciones y en las ventajas y desventajas de esta red social, decidimos tomar la barra de navegación debajo de la pantalla para tener acceso rápido a todas las pantallas principales de la aplicación.

Además, he querido hacer también sencilla la forma de modificación de los datos del perfil y la facilidad para mantener una conversación con otro usuario.

Una de las desventajas que he querido solucionar es que un usuario pueda hablar a otro, a pesar de no “ser amigos” o haber tenido un contacto previo, permitiendo únicamente a un usuario hablar a otro cuando éste reciba una notificación del otro, es decir, cuando uno necesite ayuda del tipo que el otro usuario está dispuesto a ayudar, o habiendo tenido una conversación anterior a esta, lo que implica la primera condición en una estancia anterior.

2.1.2 WHATSAPP Y TELEGRAM

WhatsApp es una aplicación que funciona como una red social de mensajería instantánea para smartphones, en la que se permite al usuario tener conversaciones con otras personas o mediante grupos.

De la misma forma, Telegram es otra red social que permite mantener mensajería instantánea entre usuarios, para mantener conversaciones entre usuarios o en grupos.

Ambas aplicaciones son utilizadas para lo mismo, pero WhatsApp ha conseguido ser la aplicación líder del mercado para este tipo de redes sociales.

A continuación podremos ver una serie de imágenes del funcionamiento de WhatsApp, en la que me he fijado con mayor detenimiento, al tener mayor número de usuarios, y por tanto, mayor usabilidad:

ILUSTRACIÓN 10: PANTALLA DE INICIO EN WHATSAPP

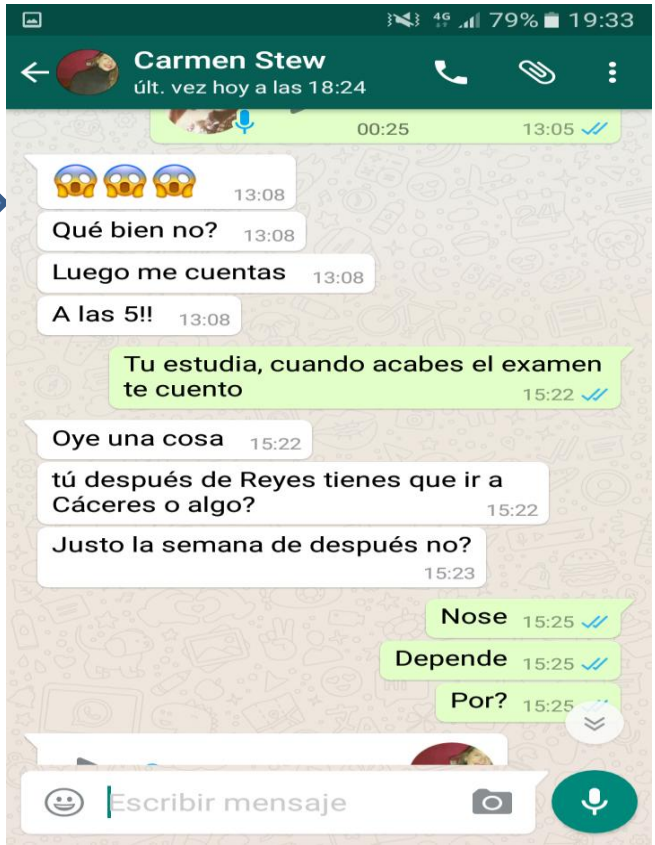


ILUSTRACIÓN 11: CONVERSACIÓN ENTRE 2 USUARIOS EN WHATSAPP

Tras ver las pantallas básicas de la aplicación en cuestión, las cuales he tratado de hacer similares, pasamos a ver una pantalla que también he intentado simular:

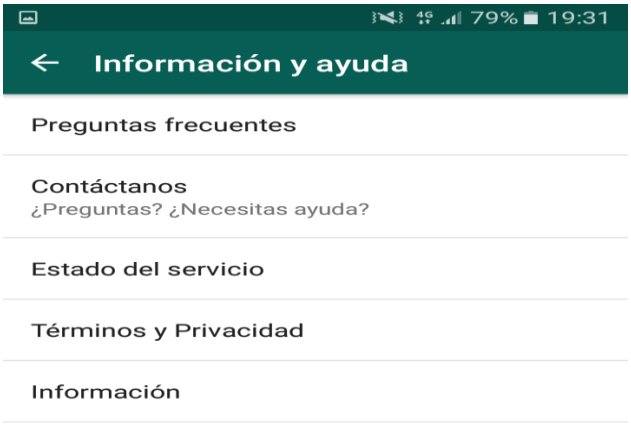


ILUSTRACIÓN 12: PANTALLA DE INFORMACIÓN Y AYUDA EN WHATSAPP

Pasamos a ver las ventajas y debilidades de WhatsApp, para intentar mantenerlo en nuestra aplicación, o evitarlo en los casos negativos.

Ventajas:

- Permite mantener una conversación textual o mediante grabaciones de audios entre 2 usuarios.
- Permite mantener una conversación entre un grupo de usuarios.
- Permite mandar música, imágenes, archivos adjuntos, etcétera.
- Da la posibilidad de contactar con los creadores para solicitar ayuda o sugerencias, así como da una sección de información de ayuda al usuario.
- Permite al usuario saber si el mensaje ha llegado correctamente al otro usuario.
- Permite personalizar la imagen y estado que verá el otro usuario.

Desventajas:

- No mantiene bien la privacidad, ya que permite a cualquier usuario que tenga el número de teléfono de otro usuario ver su foto de perfil, así como su estado, además de poder iniciar una conversación sin que uno de los usuarios lo desee.

- Es adictivo, por lo que muchas personas comienzan a perder tiempo útil en otras tareas de la vida cotidiana pegadas a la aplicación.

- Permite al usuario saber si el mensaje ha sido leído por el otro usuario, lo que puede crear conflictos.

Conclusión:

Viendo todas las ventajas e inconvenientes que tienen las redes sociales de mensajería instantánea, como son WhatsApp o Telegram, llegamos a la conclusión de los beneficios que queremos mantener y las debilidades que deseamos no acatar. Entre ello he decidido mantener el envío de mensajes instantáneos (por el momento solo textual) entre 2 usuarios, aunque en las siguientes actualizaciones me gustaría la posibilidad de conversaciones grupales.

También he mantenido la posibilidad de que el usuario pueda contactar mediante un correo electrónico con los creadores, en este caso conmigo.

Además he creado un apartado de preguntas frecuentes, con el que los usuarios podrán aprender a utilizar la aplicación o solucionar algunas de sus dudas con respecto a la misma.

De entre las desventajas he intentado evitar la pérdida de privacidad, permitiendo como he explicado en apartados anteriores, que un usuario pueda hablar únicamente con quien necesita nuestra ayuda, estando nosotros dispuestos a ayudarlo. Por tanto, nadie podrá ver tus datos sin haber tenido antes un contacto previo.

Por último, no se permite saber al usuario cuando el otro usuario ha leído el mensaje, ni tampoco cuando ha recibido correctamente el mensaje, aunque esto último se realizará para futuras actualizaciones.

2.2 APLICACIONES CON TEMÁTICA DE ANIMALES

En este aspecto, no he tenido mucha investigación, ya que actualmente no hay muchas aplicaciones que realicen la misma temática que la aplicación que se desarrolla para este proyecto.

En el transcurso del desarrollo de esta aplicación surgió una aplicación en el mercado de aplicaciones de Android, llamada Wizapet. Por tanto, para la temática decidí investigar sobre esta aplicación, aunque ya había realizado gran parte de mi trabajo.

2.2.1 WIZAPET

Wizapet es una aplicación que se utiliza para anunciar la pérdida de mascotas y anunciar el encuentro de animales, tratando así de poner de acuerdo a los contactos para encontrar a sus mascotas perdidas.

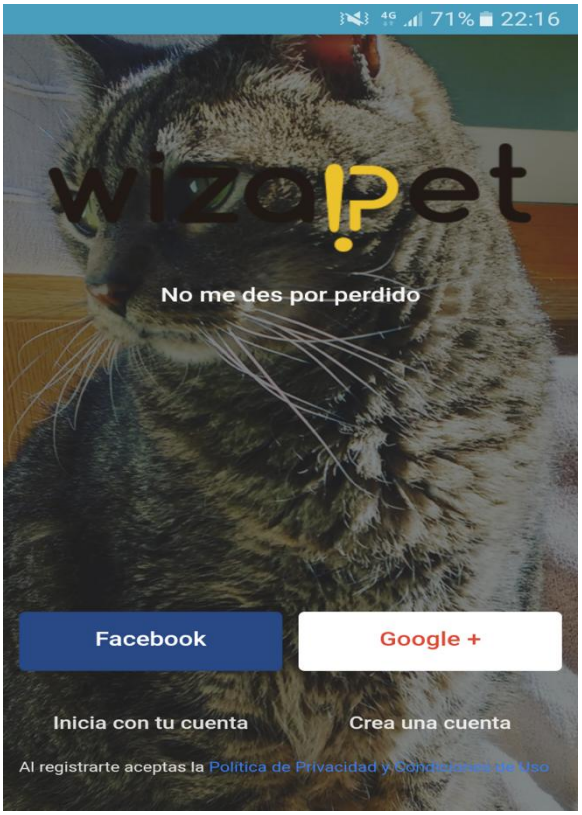


ILUSTRACIÓN 13: PANTALLA DE INICIO DE SESIÓN EN WIZAPET

Aquí podemos observar la pantalla de inicio de la aplicación en cuestión, en la que el usuario tiene que iniciar sesión o crear una cuenta.

A continuación veremos una serie de imágenes en las que podemos observar el funcionamiento de la aplicación investigada:

En esta captura podemos ver que en la pantalla de inicio de la aplicación tenemos un mapa en el que nos aparecen las pérdidas de mascotas que ha habido cercanas a nuestra ubicación.

Pinchando en cada publicación podemos observar la información de la misma.

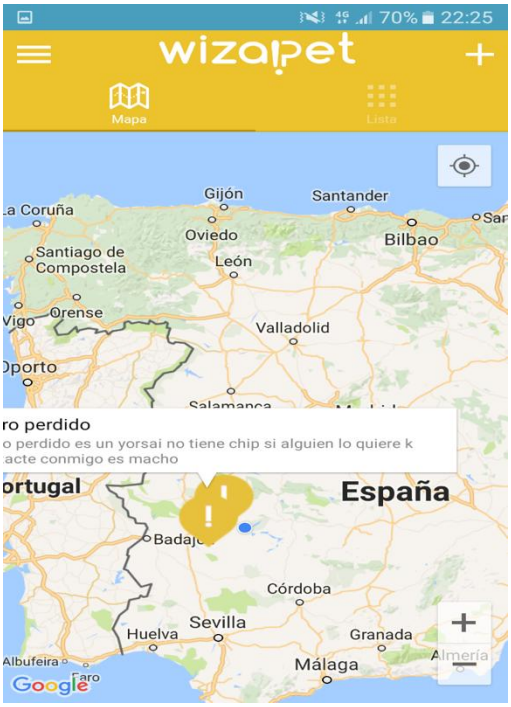


ILUSTRACIÓN 14: PANTALLA PRINCIPAL EN WIZAPET

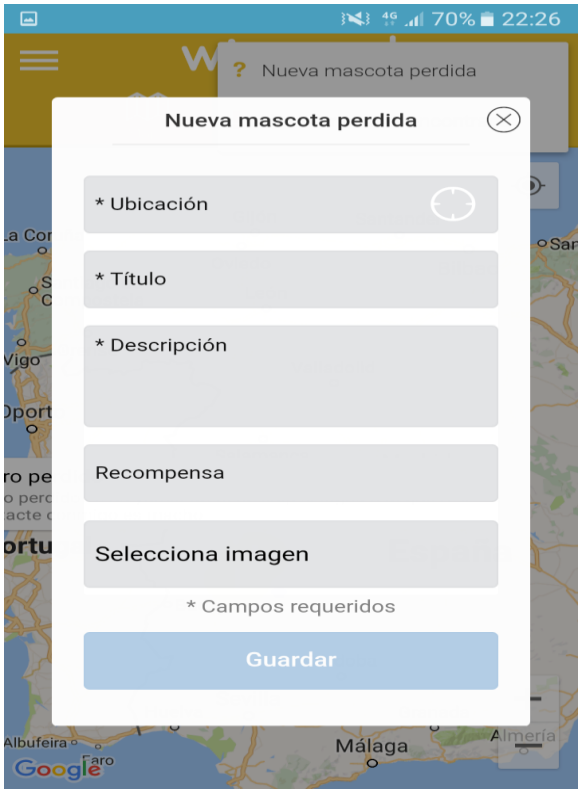


ILUSTRACIÓN 15: PANTALLA PUBLICACION PERDIDA DE MASCOTA EN WIZAPET

En esta imagen podemos ver la publicación que tenemos que rellenar cuando una mascota ha sido perdida, para poder crearlo y que el resto de usuarios lo vean.

En esta última captura podemos observar la edición de los datos personales del usuario creado para la aplicación. Como podemos ver los datos aparecen vacíos, únicamente para que puedas poner los nuevos. Además en la fecha aparece una fecha preestablecida que el usuario no ha rellenado en ningún momento.

ILUSTRACIÓN 16: PANTALLA EDICIÓN DATOS PERSONALES EN WIZAPET

Tras ver todas estas capturas y al haber investigado acerca de la aplicación en cuestión, llegamos a las siguientes ventajas y desventajas.

Ventajas:

- El usuario puede encontrar su mascota perdida.
- El diseño de la aplicación es bastante creativo.
- Posee en la pantalla de inicio un mapa en el que podemos observar la información cercana a gran velocidad.
- El usuario puede iniciar sesión a partir de otras redes sociales como Facebook.
- El usuario puede crear una cuenta propia.
- Permite al usuario ver fotos de otros usuarios y mascotas.

Desventajas:

- Únicamente permite el encuentro de mascotas al usuario, por lo que limita la aplicación a una única función.

- La edición de datos personales no muestra los datos actuales, por lo que el usuario no sabrá si realmente está modificándolos en caso de no saber en dicho momento que dato tenía puesto.

- Al crear la cuenta propia, el usuario no puede escoger un nombre ya utilizado por otro usuario. Teniendo en cuenta que 2 personas pueden tener el mismo nombre, bajo mi punto de vista, debería tener como identificador único otro dato del usuario.

Conclusión:

Al haber visualizado tanto el funcionamiento de la aplicación, como las ventajas y debilidades de la misma, y teniendo en cuenta que la aplicación de este proyecto ya estaba bastante avanzada en el momento de la aparición de la aplicación investigada, llegamos a la conclusión de los pros que deseamos mantener, así como los inconvenientes que deseamos eliminar. Entre ellos están:

- Mantener la posibilidad de encuentro de mascotas cuando se hayan perdido.

- Tener otras funciones para el usuario, como son la posibilidad de paseos o cuidados de otros usuarios a nuestras mascotas, además de la posibilidad de adopción cuando el usuario no quiera mantener a la mascota.

- Tener una pantalla inicial de un mapa en el que veamos la información rápidamente.

- Un usuario puede ver sus datos personales cuando los está intentando editar.

Para futuras actualizaciones intentaré que un usuario pueda ver imágenes de las mascotas de los otros usuarios, además de que un usuario pueda iniciar sesión a partir de otras redes sociales como Facebook, lo que me parece bastante interesante.

3. ANÁLISIS Y DISEÑO

En este capítulo vamos a explicar las distintas tareas que se han llevado a cabo para realizar las fases de análisis y diseño de este proyecto.

En los distintos apartados podremos ver la definición de los requisitos funcionales y no funcionales del proyecto, los diagramas de casos de uso derivados de los requisitos, y por último, el diagrama de flujos de la interacción del usuario en la aplicación.

3.1 REQUISITOS

Inicialmente, para la correcta realización de este proyecto, así como del resto de proyectos software, se realiza la definición de los requisitos, que son las condiciones necesarias en el proyecto para el buen funcionamiento del mismo.

A partir de estos requisitos se conseguirán definir los casos de uso, definidos posteriormente en esta documentación, en los que podemos ver la funcionalidad de la aplicación.

3.1.1 REQUISITOS FUNCIONALES

En este apartado se describen los requisitos funcionales del proyecto desarrollado para este Trabajo Fin de Grado.

Los requisitos funcionales definen las funciones y/o servicios que el sistema debe proporcionar, así como las distintas formas en que el sistema debe actuar ante las diferentes situaciones concebidas en la aplicación.

A continuación podemos ver la definición de los requisitos funcionales establecidos para este proyecto:

- Proporcionar un acceso de registro para usuarios nuevos mediante la elección de uno de los correos electrónicos de Google registrados en el dispositivo, solicitando posteriormente la introducción de datos personales tales como: nombre, apellidos, email, posesión o no de mascotas y foto para el perfil.

- Comprobación de la validación de los campos rellenados para el registro del usuario.

- Se debe ofrecer la posibilidad de registro de las distintas mascotas que posee el usuario, introduciendo los siguientes datos: Nombre de la mascota, tipo de animal y comportamiento, tamaño y sexo del mismo.

- Comprobación de la validación de todos los datos introducidos para el registro de cada una de las mascotas.

- Ofrecer un acceso a la aplicación, para el usuario ya registrado, mediante la selección del correo electrónico de las cuentas de Google registradas en el dispositivo que le permita iniciar sesión.

- Ofrecer la posibilidad al usuario de visualización de sus datos personales, permitiéndole la edición de los mismos.

- Ofrecer la posibilidad al usuario de visualización de las mascotas registradas, permitiéndole la adicción de nuevas mascotas, así como el borrado de mascotas ya almacenadas.

- Proporcionar la posibilidad al usuario de eliminar la cuenta creada, eliminando con ello toda la información almacenada del mismo.

- Ofrecer al usuario la posibilidad del cierre de la sesión iniciada anteriormente, almacenando en el dispositivo la última sesión iniciada en la aplicación para futuras sesiones.

- Permitir al usuario la creación de un servicio de **solicitud de ayuda**, con la generación de notificación que conlleva, de las siguientes formas:

    - **Cuidado**: Seleccionar las mascotas, dentro de las registradas por el usuarios, que queremos que estén implicadas en dicho servicio, así como del día de inicio y de fin del cuidado. Debe seleccionar al menos una mascota.
    - **Paseo**: Selección de las mascotas, dentro de las registradas por el usuarios, que deseamos que estén implicadas en dicho servicio, así como del día del paseo. Debe seleccionar al menos una mascota.
    - **Perdida**: Seleccionar la mascota o mascotas, dentro de las registradas por el usuarios, que se han perdido y deseamos encontrar. Debe seleccionar al menos una mascota.
    - **Adopción**: Seleccionar, de entre las mascotas del usuario, la mascota o mascotas que queremos ofrecer en adopción porque ya no deseamos tener. Debe seleccionar al menos una mascota.

- Permitir al usuario la creación de un servicio para admitir que está **dispuesto a proporcionar ayuda** al resto de usuarios de las siguientes formas:

    - **Cuidado**: Seleccionar el tipo o tipos de animales que estamos dispuesto a cuidar, además del día de inicio y de fin en los que estamos dispuestos a realizar el cuidado de estos tipos de mascota. Debe seleccionar al menos un tipo de animal.
    - **Paseo**: Seleccionar el tipo o tipos de animales que estamos dispuesto a cuidar, además del día en que estamos dispuestos a realizar el paseo de este o estos tipos de mascota. Debe seleccionar al menos un tipo de animal.
    - **Perdida**: Seleccionar el tipo o tipos de animales que hemos encontrado y que por tanto otro usuario habrá perdido. Debe seleccionar al menos un tipo de animal.
    - **Adopción**: Seleccionar el tipo o tipos de animales que estamos dispuestos a adoptar. Debe seleccionar al menos un tipo de animal.

- Dar la posibilidad al usuario del borrado de cualquiera de los servicios creados anteriormente, así como de lo que el borrado de un servicio conlleve.

- Comprobar los servicios del tipo cuidado y paseo (son los que tienen fechas establecidas), para que si la fecha de estos está pasada, es decir, ya no necesitan ser realizados, se eliminen automáticamente, borrando lo que conlleve.

- Recepción de la aplicación de las notificaciones que tengamos que recibir, al haberlo aceptado mediante la creación de servicios.

- Recepción de notificaciones de tipo mensaje, enviado por otro usuario y almacenado en su chat.

- Dar la posibilidad al usuario para abrir un chat con otro usuario.

- Permitir al usuario enviar un mensaje a otro usuario, desde el chat.

- Dar la posibilidad al usuario de visualizar todas las notificaciones recibidas en ese móvil, permitiendo abrir un chat desde cada una de ellas con el usuario que lo envió.

- Dar la posibilidad al usuario de visualizar todos los usuarios con los que ha creado alguna conversación, así como la última fecha de conexión en esa conversación.

- Permitir al usuario borrar una conversación si no la desea tener almacenada.

- Visualización de un mapa, en el que se observe la localización del usuario y además veamos las notificaciones que hayamos recibido en un rango de distancia cercano a dicha ubicación, dando la posibilidad de abrir un chat con el usuario que nos la envió seleccionando la marca que deseemos de las existentes en el mapa.

- Visualización de preguntas frecuentes de ayuda, para obtener información de distintos puntos de la aplicación y facilitar el uso de la misma.

- Dar la posibilidad al usuario de enviar un correo electrónico a los desarrolladores de la aplicación, en este caso a mí, para solicitar ayuda no proporcionada en las preguntas frecuentes, sugerir actualizaciones o modificaciones de la aplicación. Estos correos serán procesados externamente por mí, para considerar las peticiones o sugerencias y ayudar en las dudas.

A continuación, podemos ver todos estos requisitos funcionales resumidos en la siguiente tabla resumen de requisitos:

TABLA 1: REQUISITOS FUNCIONALES

Ident	Descripción
RF1	Registro del usuario
RF2	Comprobación datos del usuario
RF3	Registro de las mascotas
RF4	Comprobación datos de las mascotas
RF5	Inicio de sesión del usuario
RF6	Visualización y edición de datos personales del usuario
RF7	Visualización de las mascotas del usuario, permitiendo el borrado y añadido de mascotas
RF8	Eliminar cuenta y perfil del usuario

RF9	Cerrado de sesión del usuario
RF10	Creación de un servicio de solicitud de ayuda, de uno de los tipos permitidos
RF11	Creación de un servicio de proporción de ayuda, de uno de los tipos permitidos
RF12	Borrado de servicios por parte del usuario
RF13	Borrado automático de servicios, mediante la comprobación de fechas
RF14	Recepción de notificación de otro usuario
RF15	Recepción de mensaje de otro usuario
RF16	Abrir chat con otro usuario
RF17	Enviar mensaje a otro usuario
RF18	Visualizar todas las notificaciones recibidas
RF19	Visualizar todas las conversaciones creadas
RF20	Borrado de una conversación
RF21	Visualización del mapa, con la ubicación actual y los servicios en los que estamos interesados de otras personas cercanas
RF22	Visualización de preguntas frecuentes para el correcto uso de la aplicación
RF23	Posibilidad del usuario de solicitar ayuda o sugerencias mediante correo electrónico hacia mí, para gestionar sus peticiones.

3.1.2 REQUISITOS NO FUNCIONALES

Estos requisitos no describen los servicios de la aplicación, sino que indican las restricciones y condiciones de los servicios y funciones ofrecidas por el sistema mediante los requisitos funcionales.

Estos requisitos suelen ser condiciones de fiabilidad, capacidad de almacenamiento, conexiones a redes, etcétera.

A continuación expongo la definición de los requisitos no funcionales establecidos para esta aplicación:

- La aplicación debe ser lo suficientemente **intuitiva** como para que cualquier usuario pueda utilizarla, teniendo FAQs para las pequeñas dudas que existan.

- La aplicación tiene que ser lo suficientemente **creativa y agradable** como para llamar inicialmente y en todo momento la atención del usuario.

- La **curva de aprendizaje** del usuario debe ser **mínima**.

- El sistema debe mostrar **mensajes de error cuando lo necesite**, permitiendo al usuario conocer el error que ha cometido o por qué no ha podido realizar una determinada tarea.

- El usuario debe tener constancia **en todo momento** de en qué lugar de la aplicación se encuentra y **saber que función está realizando**, mediante las barras de navegación y la de encima de la pantalla.

- El sistema debe estar correctamente organizado como para **minimizar los errores del usuario**.

- La aplicación debe gastar el **mínimo posible de batería**.

- Los datos manejados por la aplicación **no deben contener información sensible del usuario o sus mascotas**.

3.1.3. REQUISITOS DEL SISTEMA

Los requisitos que el sistema donde se ejecute la aplicación son:

- El dispositivo donde se ejecute la aplicación debe ser del sistema operativo **Android**.
- La versión de Android debe ser **igual o superior a Android 5.0 Lollipop**.
- El dispositivo tiene que disponer de una **conexión a internet**, ya sea mediante conexión a una red wifi o mediante una conexión de datos móviles.
- El dispositivo debe tener el **GPS activado** para la creación de servicios y recepción de notificaciones.
- El sistema debe tener un **tiempo de respuesta aceptable**, para mostrar al usuario la información en tiempo real y no entorpecer en su ejecución.

3.2 CASOS DE USO

En este apartado de la documentación vamos a ver el diagrama de casos de uso. Un caso de uso es una operación o tarea que se realiza tras una orden de algún agente externo, ya sea desde una petición de un actor o bien desde la invocación de otro caso de uso.

En una primera instancia vamos a ver la descripción del usuario que utilizará la aplicación, por tanto, veremos al llamado actor. Posteriormente veremos el diagrama de casos de uso y una explicación de éste.

3.2.1 DESCRIPCIÓN DE LOS ACTORES

Usuario: Será la persona que utilizará la aplicación móvil para tratar de relacionarse con otros usuarios consiguiendo ayudar o ser ayudado a través de la misma.

Usuario

ILUSTRACIÓN 17: ACTOR DE LA APLICACIÓN

3.2.2. DIAGRAMA DE CASOS DE USO

En este apartado vamos a ver el diagrama de casos de uso. A continuación, pasaremos a explicar cada uno de los casos de uso.

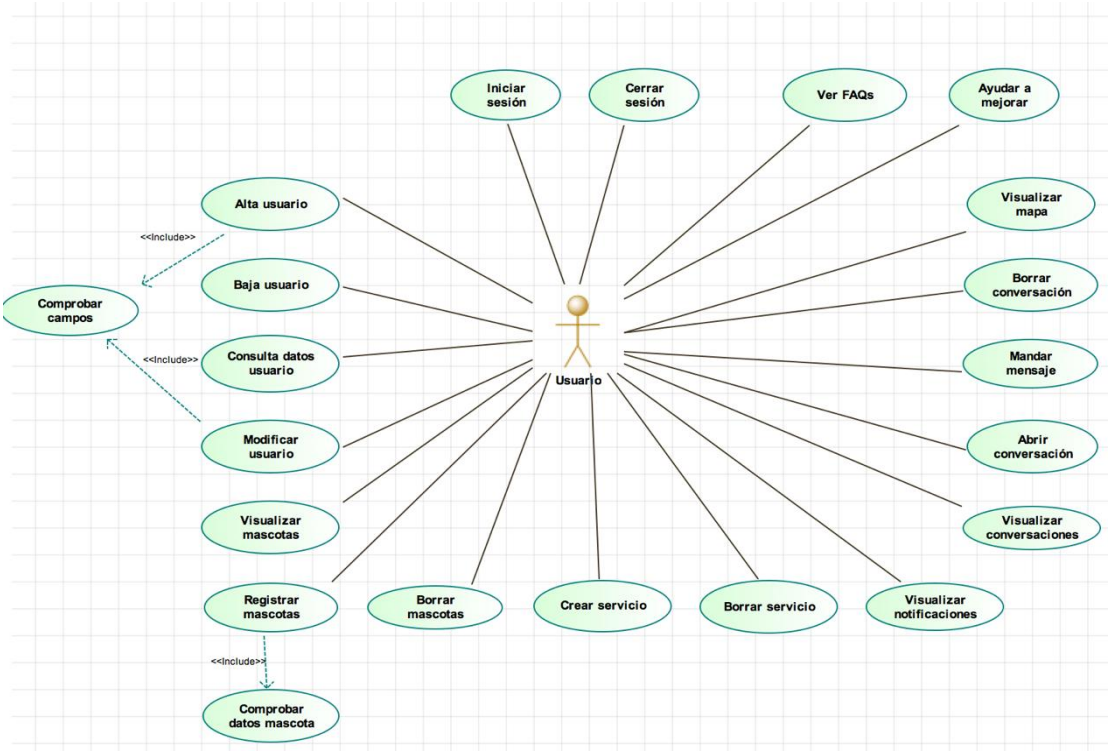


ILUSTRACIÓN 18: ACTOR DE LA APLICACIÓN

3.2.2.1 DESCRIPCIÓN DE LOS CASOS DE USO

En el presente apartado vamos a ver las fichas explicativas de cada uno de los casos de uso del diagrama visto anteriormente.

TABLA 2: CASO DE USO "ALTA USUARIO"

Nombre	Alta usuario
Descripción	Permite al usuario registrarse en la aplicación
Precondición	-
Postcondición	Una vez introducido los datos del usuario quedará guardado en la base de datos y registrado en Nimbees
Flujo principal	1. Rellenar los datos solicitados 2. Se realiza el caso de uso “Comprobar campos” 3. Se registra el usuario en Nimbees 4. Se almacenan los datos en las bases de datos 5. Queda registrado el usuario en la aplicación
Flujo alternativo	3.b No se registra en Nimbees y por tanto tampoco en la aplicación

TABLA 3: CASO DE USO "BAJA USUARIO"

Nombre	Baja usuario
Descripción	Permite al usuario borrar el perfil en la aplicación
Precondición	Debe estar registrado y logueado el usuario
Postcondición	El perfil del usuario queda eliminado
Flujo principal	1. Se eliminan los servicios de la base de datos 2. Se eliminan las mascotas de la base de datos 3. Se eliminan las conversaciones de la base de datos 4. Se eliminan el usuario de la base de datos 5. Se elimina el usuario de la cuenta

TABLA 4: CASO DE USO "CONSULTA USUARIO"

Nombre	Consulta usuario
Descripción	Permite al usuario ver sus datos personales
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se mostrarán los datos del usuario

TABLA 5: CASO DE USO "MODIFICAR USUARIO"

Nombre	Modificar usuario
Descripción	Permite al usuario modificar sus datos personales
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se actualizarán los datos del usuario
Flujo principal	1. Se ejecuta el caso de uso “Consultar usuario” 2. Se rellenan los datos personales 3. Se ejecuta el caso de uso “Comprobar campos” 4. Se modifica el usuario en la base de datos

Nombre	Visualizar mascotas
Descripción	Permite al usuario ver sus mascotas
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se visualizarán las mascotas del usuario

TABLA 6: CASO DE USO "VISUALIZAR MASCOTAS"

TABLA 7: CASO DE USO "REGISTRAR MASCOTAS"

Nombre	Registrar mascotas
Descripción	Permite al usuario añadir mascotas
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se almacenarán las mascotas del usuario
Flujo principal	1. Rellenar los datos solicitados 2. Se realiza el caso de uso “Comprobar datos mascotas” 3. Se almacenan las mascotas en las bases de datos

TABLA 8: CASO DE USO "BORRAR MASCOTAS"

Nombre	Borrar mascotas
Descripción	Permite al usuario borrar mascotas
Precondición	El usuario debe estar registrado y logueado y poseer mascotas
Postcondición	Se eliminarán las mascotas que el usuario desee
Flujo principal	1. Se eliminan de la base de datos las mascotas seleccionadas

TABLA 9: CASO DE USO "CREAR SERVICIO"

Nombre	Crear Servicio
Descripción	Permite al usuario crear un servicio de ayuda
Precondición	El usuario debe estar registrado, logueado y debe tener GPS activado
Postcondición	Se creará un servicio proporcionando ayuda o pidiéndola.
Flujo principal	1. Rellenar los datos solicitados (proporcionar ayuda o recibirla) 2. Se eligen las mascotas implicadas 3. Se almacenan los datos en las bases de datos 4. Se generan las notificaciones si el servicio es de pedir ayuda o se activan los eventos a los que estamos dispuestos a ayudar si el servicio es para proporcionar ayuda

TABLA 10: CASO DE USO "BORRAR SERVICIO"

Nombre	Borrar servicio
Descripción	Permite al usuario borrar un servicio de ayuda
Precondición	El usuario debe estar registrado, logueado y debe tener creado el servicio
Postcondición	Se borrará el servicio y se desactivarán los eventos correspondientes
Flujo principal	1. Se selecciona el servicio a borrar 2. Se eliminan los datos correspondientes en la BD 3. Si el servicio era de proporcionar ayuda, se desactivarán los eventos activados por tal servicio

TABLA 11: CASO DE USO "VISUALIZAR NOTIFICACIONES"

Nombre	Visualizar notificaciones
Descripción	Permite al usuario ver todas las notificaciones que ha recibido desde sus inicios en la aplicación
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se muestran todas las notificaciones recibidas por el usuario
Flujo principal	1. Se consulta a Nimbees las notificaciones de dicho usuario 2. Se muestran las notificaciones
Flujo alternativo	2.b No tiene notificaciones, por lo que se muestra un mensaje diciendo que no hay notificaciones

TABLA 12: CASO DE USO "VISUALIZAR CONVERSACIONES"

Nombre	Visualizar conversaciones
Descripción	Permite al usuario ver todas las conversaciones almacenadas
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se muestran todas las conversaciones que ha tenido el usuario y no han sido borradas
Flujo principal	1. Se consultan en la base de datos las conversaciones 2. Se muestran las conversaciones
Flujo alternativo	2.b No tiene conversaciones, por lo que se muestra un mensaje diciendo que no hay conversaciones

TABLA 13: CASO DE USO "ABRIR CONVERSACIÓN"

Nombre	Abrir conversación
Descripción	Permite al usuario abrir conversación con otro usuario
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se abre conversación entre los 2 usuarios
Flujo principal	1. Se consulta en la base de datos la conversación con ese usuario 2. Se muestra la conversación
Flujo alternativo	2.b No tiene conversación con ese usuario, por lo que se abre una nueva

TABLA 14: CASO DE USO "BORRAR CONVERSACIÓN"

Nombre	Borrar conversación
Descripción	Permite al usuario borrar una conversación
Precondición	El usuario debe estar registrado, logueado y debe existir la conversación
Postcondición	Se elimina la conversación seleccionada
Flujo principal	1. Se ejecuta el caso de uso “Visualizar conversaciones” 2. Se selecciona la conversación a borrar 3. Se eliminan los datos de la base de datos

TABLA 15: CASO DE USO "MANDAR MENSAJE"

Nombre	Mandar mensaje
Descripción	Permite al usuario mandar un mensaje a otro usuario
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se envía el mensaje al otro usuario
Flujo principal	1. Se envía el mensaje a Nimbees 2. Nimbees se lo mandará al usuario que lo debe recibir 3. Se almacenan los datos en la base de datos

TABLA 16: CASO DE USO "VISUALIZAR MAPA"

Nombre	Visualizar mapa
Descripción	Permite al usuario ver el mapa, con todas las notificaciones recibidas y cercanas en ese momento
Precondición	El usuario debe estar registrado y logueado y el GPS activado
Postcondición	Se muestran un mapa con las notificaciones recibidas y que hayan sido enviadas a una distancia igual o menor a 5000 metros
Flujo principal	1. Se consultan en Nimbees las notificaciones recibidas 2. Se pinta el mapa 3. Se pintan las notificaciones correspondientes

TABLA 17: CASO DE USO "VER FAQS"

Nombre	Ver FAQs
Descripción	Permite al usuario ver las preguntas frecuentes, para recibir información de ayuda en el uso de la aplicación
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se muestran las preguntas frecuentes, que servirán de ayuda al usuario
Flujo principal	1. Se muestran las preguntas frecuentes 2. Cuando selecciones una se muestra la respuesta a dicha pregunta

TABLA 18: CASO DE USO "AYUDA A MEJORAR"

Nombre	Ayuda a mejorar
Descripción	Permite al usuario enviar un correo al desarrollador de la aplicación, para hacer sugerencia o consultar dudas
Precondición	El usuario debe estar registrado y logueado y declarar tener mayoría de edad
Postcondición	Se enviar un correo electrónico al desarrollador de la aplicación
Flujo principal	1. Se rellenan los campos 2. Se pasa el token al dispositivo para que contacte con un gestor de correos 3. Seleccionamos el gestor 4. Se envía el mensaje

TABLA 19: CASO DE USO "INICIAR SESIÓN"

Nombre	Iniciar sesión
Descripción	Permite al usuario iniciar sesión en la aplicación mediante el login de Google, para poder realizar el resto de acciones de la aplicación
Precondición	El usuario debe estar registrado
Postcondición	El usuario iniciará sesión en la aplicación
Flujo principal	1. Se solicitará al usuario que elija de entre una de las cuentas de Google 2. Se iniciará sesión con ese correo 3. Se cargarán los datos asociados a esa cuenta
Flujo alternativo	2.b. Si estaba registrado con uno de los otros correos en lugar de con el seleccionado, se borrarán los datos del registro anterior y se pedirá el registro con este correo elegido

TABLA 20: CASO DE USO "CERRAR SESIÓN"

Nombre	Cerrar sesión
Descripción	Permite al usuario cerrar su sesión en la aplicación
Precondición	El usuario debe estar registrado y logueado
Postcondición	Se cerrará la sesión y se volverá a la pantalla de inicio de sesión
Flujo principal	1. En la pestaña de Ajustes, tenemos que pinchar en el botón de salir y se borrarán las sharedPreferences almacenadas con el usuario

TABLA 21: CASO DE USO "COMPROBAR CAMPOS"

Nombre	Comprobar campos
Descripción	Se comprueba si se han introducido todos los datos necesarios y obligatorios
Precondición	-
Postcondición	Si están todos los campos rellenos dará paso al caso de uso que le llamó
Flujo principal	1. Se recogen los datos introducidos en los parámetros solicitados 2. Se comprueba si están todos rellenos 3. Se continua ejecutando el caso de uso que le llamó
Flujo alternativo	2.b. Si no están rellenos se muestra un mensaje al usuario para que los rellene

TABLA 22: CASO DE USO "COMPROBAR DATOS MASCOTA"

Nombre	Comprobar datos mascota
Descripción	Se comprueba que los datos para el registro de una mascota están rellenos
Precondición	-
Postcondición	Si están todos los campos rellenos dará paso al caso de uso que le llamó
Flujo principal	1. Se recogen los datos introducidos 2. Se comprueba si están todos rellenos 3. Se continua ejecutando el caso de uso que le llamó
Flujo alternativo	2.b. Si no están rellenos se muestra un mensaje al usuario para que los rellene

3.3 ARQUITECTURA DE LA APLICACIÓN

Para la aplicación PETSMobile realizada para este Trabajo Fin de Grado, una de las tareas a realizar previas a la implementación, fue la elección de la arquitectura necesaria.

Tras grandes dudas sobre la elección de servidores, bases de datos y demás, se decidió utilizar la siguiente arquitectura.

Base de datos local para los siguientes datos:

- Datos personales
- Mascotas
- Mensajes
- Servicios de ayuda

Mediante el **servidor de Nimbees**, se distribuirán las notificaciones generadas por los dispositivos y se mandan a los dispositivos que realmente tienen que recibirlas.

3.4 DIAGRAMA DE FLUJOS

En este apartado vamos a ver el diagrama de flujos creado, que contendrá el flujo que sigue en la aplicación cada una de las acciones realizadas por el usuario en la aplicación.

Este diagrama de flujos fue de gran ayuda para comenzar la implementación de la aplicación.

A continuación, vamos a explicar todo el diagrama de flujos, para poder entender con mayor facilidad esta ilustración.

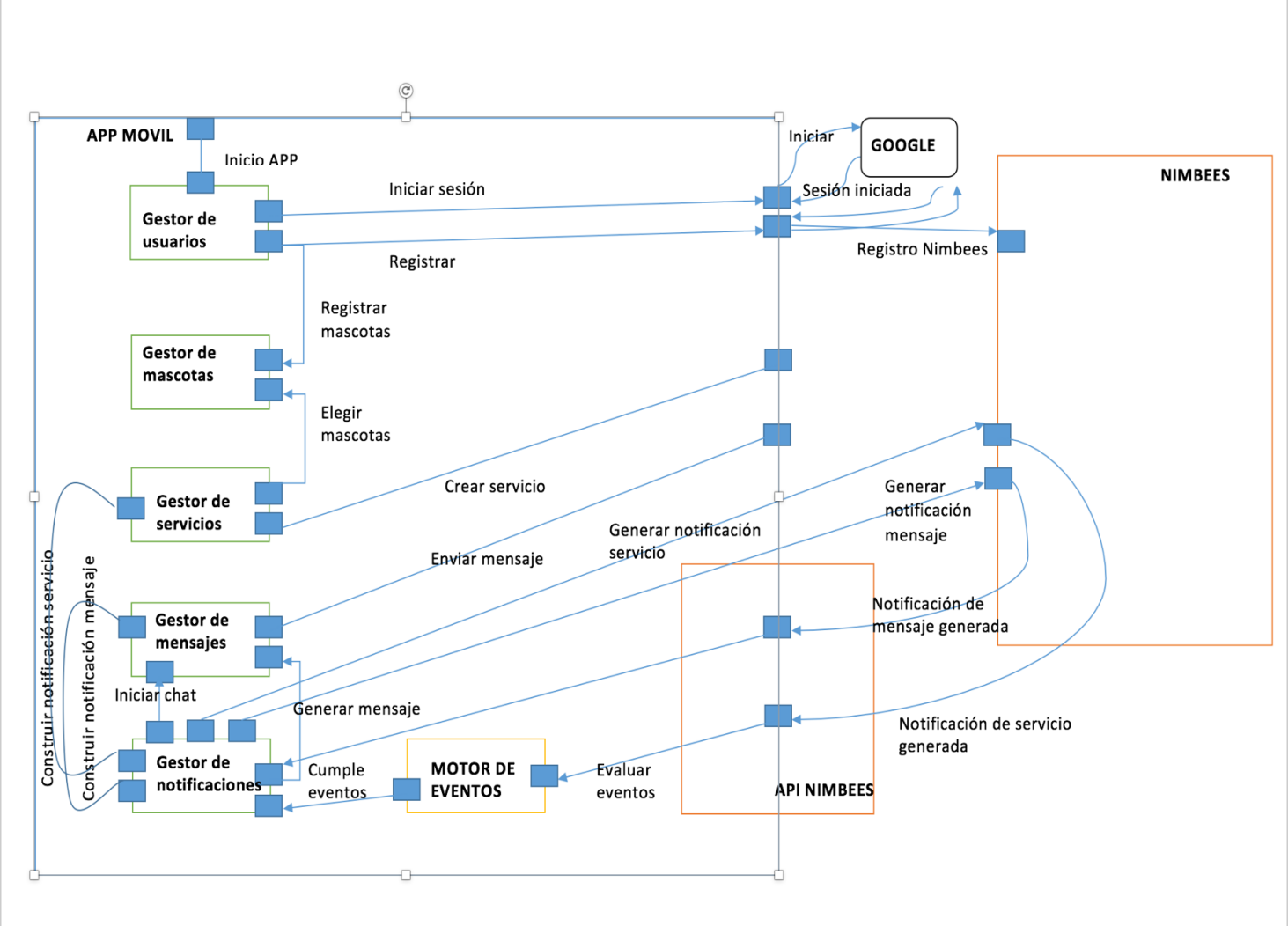


ILUSTRACIÓN 19: DIAGRAMA DE FLUJOS

3.4.1 EXPLICACIÓN DIAGRAMA DE FLUJOS

En la siguiente explicación, vamos a tratar de comentar con la mayor claridad cada uno de los textos que tiene el diagrama de flujos, en la ilustración anterior:

- **Inicio APP:** Representa una acción del usuario. Al realizar esta acción se desencadenarán el resto de funciones.

- **Iniciar sesión:** La APP Móvil pasa el control al dispositivo, que nos permitirá elegir una cuenta Google de las que tiene iniciada sesión en el mismo. Elegimos la cuenta de Google con la que queremos iniciar la sesión y por tanto nos devolverá el email, que será con el que iniciemos sesión en la APP.

- **Registrar:** La APP Móvil pasa el control al dispositivo, que nos permitirá elegir una cuenta Google de las que tiene iniciada sesión en el mismo. Elegimos la cuenta de Google con la que queremos registrarnos y por tanto nos devolverá el email, que será con el que registraremos nuestros datos en la APP. Rellenamos los datos adicionales con los que se registrará el usuario y que serán almacenados en la Base de datos local. Se rellenan además las mascotas, que se registrarán mediante el *Registrar Mascotas*, y serán almacenadas en la base de datos local. Por último se registra el usuario en Nimbees con la misma cuenta de Google.

- **Crear Servicio**: Representa la acción del usuario. Esta acción, desencadenará lo siguiente. Rellenamos los datos del servicio, es decir, si queremos ayudar o ser ayudados y que tipo de servicio queremos (cuidado, paseo, perdida, adopción), y seleccionamos las mascotas que queremos tener en este servicio mediante el *Elegir Mascotas.* Al crear un servicio, mediante el *Construir Notificación,* el Gestor de notificaciones creará una notificación del tipo Servicio que se le mandará al servidor de Nimbees para que la distribuya entre todos los dispositivos que tengan la aplicación.

- **Notificaciones de servicio generadas:** La notificación que ha recibido el servidor será enviada a todos los dispositivos registrados en la aplicación, y será recibida por la API de Nimbees.

- **Evaluar Eventos:** El motor de eventos evaluará los eventos de dicha notificación y comprobará si debe ser mostrada en dicho dispositivo. Si cumple los eventos, se mostrará la notificación push, en caso contrario, será desechada.

- **Enviar Mensaje:** Representa la acción del usuario. Esta acción, desencadenará inicialmente se escribe el mensaje que se quiere enviar. Dicho mensaje será almacenado en la base de datos local, en el chat concreto al que pertenezca. Al enviar un mensaje, mediante el *Construir Notificación de Mensaje* se generará una notificación del tipo Mensaje, que llevará dentro la información del mensaje. El gestor de Notificaciones enviará dicha notificación ya construida al servidor de Nimbees mediante el *Generar Notificación Mensaje*, para que este se lo mande al dispositivo que tiene que recibir dicho mensaje.

- **Notificación de Mensaje Generada:** La notificación que el Servidor de Nimbees ha recibido es enviada al dispositivo al que va dirigido dicho mensaje. Dicha notificación es recibida en la API de Nimbees.

- **Generar Mensaje:** El Gestor de Notificaciones tras observar que la notificación es del tipo Mensaje generará un mensaje que tendrá la información que viene en dicha notificación. El gestor de Mensajes será el que se encargue de mostrar dicho mensaje en el chat que le corresponda.

- **Iniciar chat:** A partir de una notificación, se generará un chat vacío con el usuario de la notificación. En caso de que exista ya un chat con dicho usuario, se abrirá el chat que ya existía.

3.5 PLANIFICACIÓN DEL PROYECTO

Al principio, el director de este Trabajo Fin de Grado y yo estuvimos decidiendo que idea se podía utilizar para la realización del mismo. Tras exponer en varias tutorías las distintas ideas, se acordó realizar la aplicación definitiva que ha sido realizada para el proyecto.

Para la realización del proyecto, se definieron las siguientes tareas a llevar a cabo:

- Realizar una investigación del mercado relacionado con la aplicación y existente actualmente.

- A continuación, la tarea a realizar fue un prototipo a papel de la aplicación, también conocido como mockup.

- Una vez el prototipo era creativo y bastante aceptable, se comenzaron a realizar la recolección de requisitos funcionales de la aplicación.

- Posteriormente, viendo las distintas necesidades y funciones de la aplicación, se eligió la arquitectura de la misma.

- A continuación, se realizó un diagrama de flujos de la aplicación, con el que pude ver como comenzar con la implementación del proyecto.

- La siguiente fase fue la de implementación. En ella estuve realizando el desarrollo de la aplicación para Android mediante el entorno de desarrollo Android Studio. Además, se estuvo llevando un control de versiones de la aplicación mediante bitbucket.

- Una vez realizada la implementación funcional de la aplicación, se realizaron las pruebas de la misma.

- Por último, se realizó toda la documentación del proyecto.

Durante todo el tiempo de realización de este trabajo se estuvieron teniendo reuniones entre el tutor, el cotutor y el autor del proyecto, para solucionar dudas y ver la evolución del trabajo.

La tarea más costosa del Trabajo Fin de Grado ha sido claramente la implementación de la aplicación.

3.6. DISEÑO DE LA INTERFAZ

En este apartado vamos a ver gráficamente y explicado el diseño inicial de la interfaz de la aplicación. Aunque el diseño de la interfaz no es un asunto estrictamente trivial, una aplicación con un diseño poco usable o poco atractivo para el usuario daría la sensación de dejadez y causaría poca fiabilidad y falta de confianza del usuario en la aplicación, por lo que lo tomamos como un apartado importante.

A continuación podremos ver en las ilustraciones el diseño realizado inicialmente. Estos diseños fueron realizados por el método más tradicional que existe, papel y bolígrafo, pero me ayudaron bastante a la hora de pasarlos a diseño real.

3.6.1 PANTALLA DE INICIO DE SESIÓN

Esta pantalla inicialmente iba a estar para solicitar las credenciales de un usuario y posteriormente iniciar sesión con ellas. Finalmente la aplicación realiza el inicio de sesión con las cuentas de Google, por lo que únicamente tenemos un botón para solicitar con que cuenta se desea iniciar la sesión o el registro personal.

ILUSTRACIÓN 20: PANTALLA INICIO SESIÓN EN MOCKUP

3.6.2 PANTALLA DE REGISTRO PERSONAL

En esta pantalla podemos ver el formulario de datos necesario para el registro personal de un usuario.

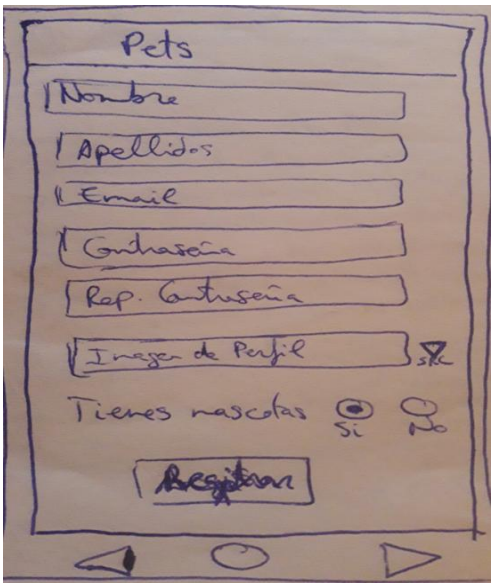


ILUSTRACIÓN 21: PANTALLA REGISTRO EN MOCKUP

Si el usuario selecciona que si tiene mascotas, se registrará sin ellas, pero si selecciona que si, le aparecerá la siguiente pantalla, donde podemos ver las mascotas que vamos añadiendo en una lista, un botón para añadir más mascotas (si lo pinchamos veremos la pantalla de la derecha), y un botón para registrarse con las mascotas que hay en la lista.

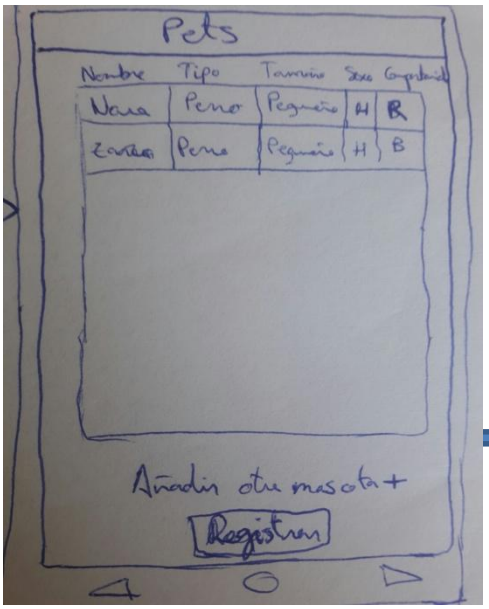


ILUSTRACIÓN 23: PANTALLA REGISTRO MASCOTAS EN MOCKUP

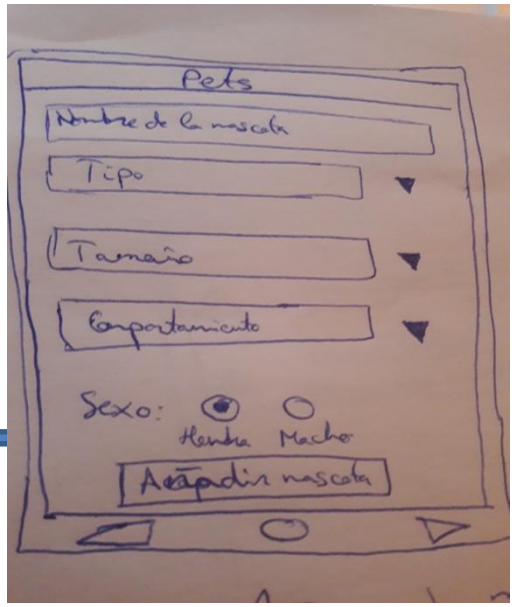


ILUSTRACIÓN 22: PANTALLA REGISTRO UNA MASCOTA EN MOCKUP

3.6.3 PANTALLA PRINCIPAL

En esta pantalla podemos ver un mapa geográfico, proporcionado por GoogleMaps. En dicho mapa, aparece nuestra ubicación y además aparecen unas marcas (con forma de huella de un animal, que es el icono de la aplicación), que representan las notificaciones que hemos recibido por parte del resto de usuarios. Solamente serán representadas en estas marcas las notificaciones que han sido enviadas desde un lugar que está a una distancia igual o menor a 5000 metros de nuestra ubicación actual.

Además, aparece un botón flotante, desde el que crearemos los servicios, es decir, las solicitudes de ayuda o donde podremos proporcionar la ayuda. Si lo pinchamos, aparecerá la pantalla de la derecha.

ILUSTRACIÓN 24: PANTALLA PRINCIPAL EN MOCKUP

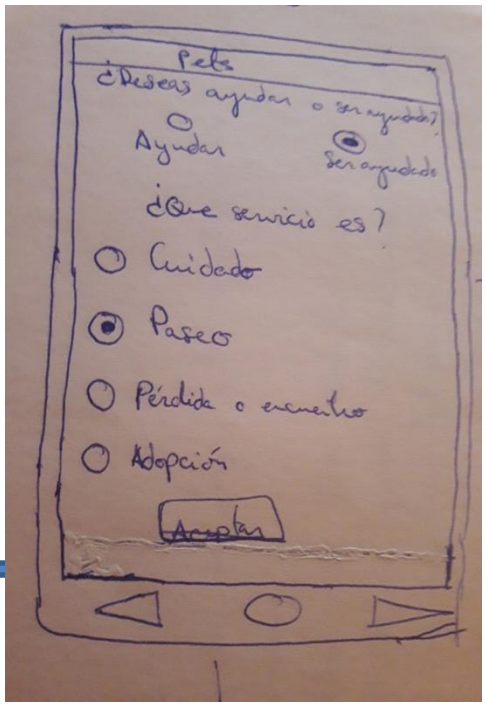


ILUSTRACIÓN 25: PANTALLA CREACIÓN DE SERVICIO EN MOCKUP

3.6.4 PANTALLA DE NOTIFICACIONES

En esta ilustración podemos ver la pantalla donde aparecen todas las notificaciones que ha recibido el usuario por parte de otros usuarios solicitando ayuda.

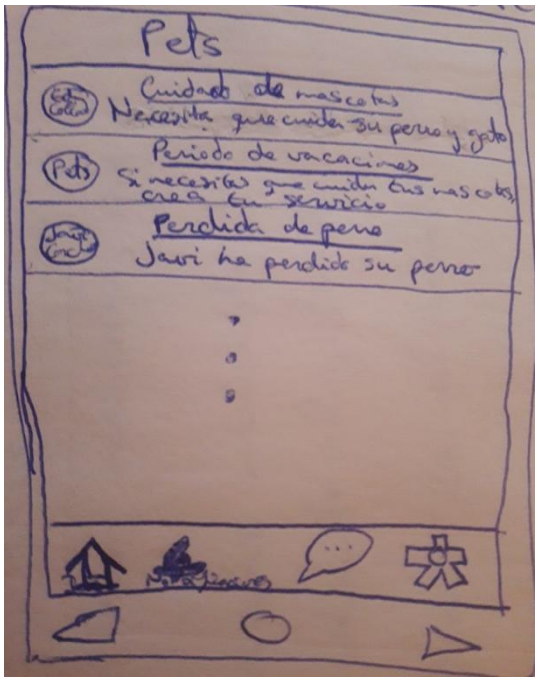


ILUSTRACIÓN 26: PANTALLA NOTIFICACIONES EN MOCKUP

A partir de cada una de estas notificaciones podemos, simplemente pinchándola, abrir una conversación con la persona que nos solicitó ayuda mediante dicha notificación.

3.6.5 PANTALLA DE MENSAJES

En la siguiente ilustración podemos observar la pantalla en la que aparecen las conversaciones que el usuario ha tenido con otros usuarios. Para cada conversación aparece ademas del nombre del otro usuario, la fecha en la que se mando el ultimo mensaje en dicha conversación.

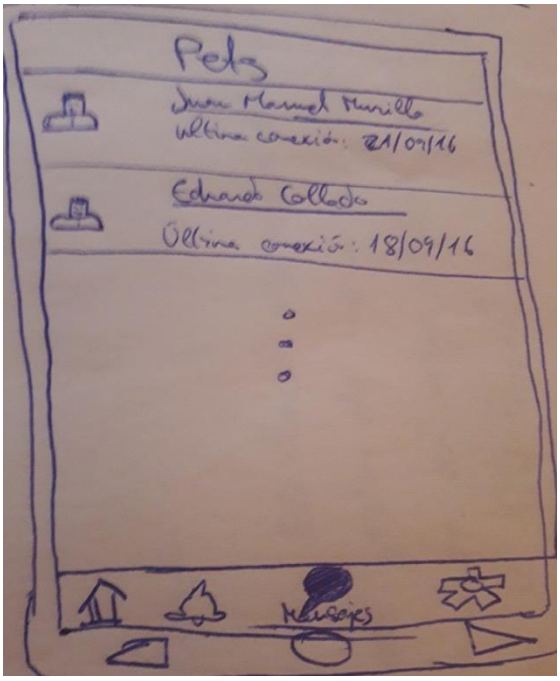


ILUSTRACIÓN 27: PANTALLA DE MENSAJES EN MOCKUP

Cuando seleccionemos alguna de esas conversaciones se abrirá, para que podamos ver los mensajes y podamos enviar mensajes nuevos.

Además, si hacemos un click largo en una de las conversaciones, nos dará la opción de borrar dicha conversación.

3.6.6 PANTALLA DE AJUSTES

En esta ilustración podemos ver la pantalla de ajustes generales de la aplicación.

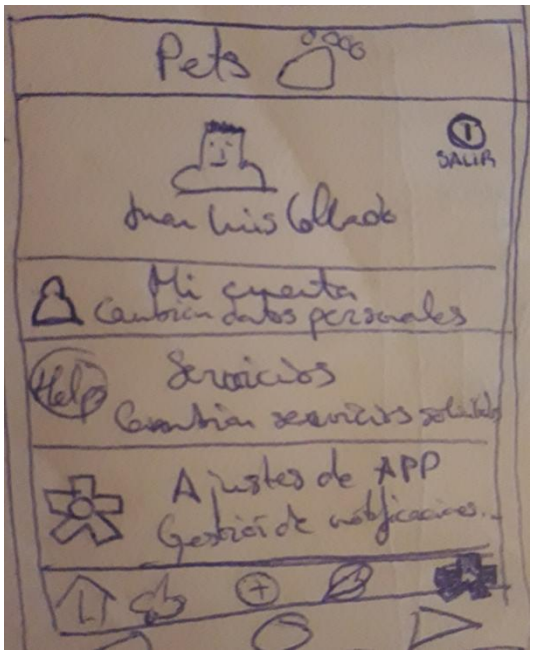


ILUSTRACIÓN 28: PANTALLA AJUSTES EN MOCKUP

En ella se puede ver la foto de perfil del usuario y su nombre. Si pinchamos en el botón de salir, la aplicación nos dará la oportunidad de cerrar sesión.

Si pinchamos en “Mi cuenta”, veremos los datos personales del usuario, así como sus mascotas, y podremos editar todos estos datos.

Al seleccionar “Servicios”, podremos ver todos los servicios de solicitud o proporción de ayuda que hayamos creado.

En la pestaña “Ajustes de APP” tendremos la oportunidad de cambiar ajustes de vibración y sonido de las notificaciones de la aplicación.

4. IMPLEMENTACIÓN

En este cuarto capítulo se encuentra la explicación de lo relativo a la implementación del proyecto.

En primer lugar, se explicará brevemente el sistema de control de versiones utilizado y la estructura elegida para el proyecto, así como el entorno de desarrollo.

Posteriormente, se comentarán las principales librerías utilizadas en el proyecto, además de la utilización en el mismo.

Para finalizar, explicaremos la implementación específica de algunos de los puntos más importantes en el desarrollo del proyecto.

4.1 CONTROL DE VERSIONES

El control de versiones es un sistema que almacena los cambios realizados sobre un archivo o conjunto de archivos de un proyecto, a lo largo del tiempo, facilitando así la recuperación de archivos modificados posteriormente mediante el modo de recuperación de versiones.

Para el control de versiones de este proyecto he utilizado GIT, ya que es más rápido, más sencillo de utilizar, y por último, almacena mucha más información en menos espacio, comparándolo con SVN.

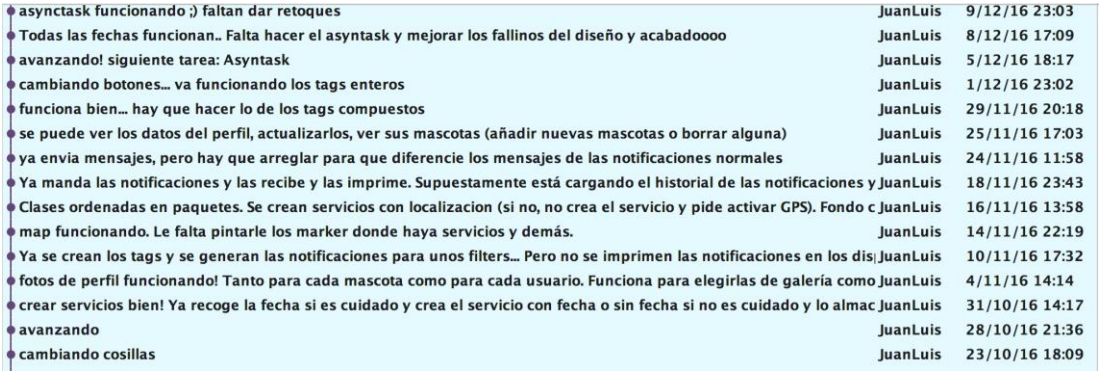


ILUSTRACIÓN 29: CONTROL DE VERSIONES EN ANDROID STUDIO

4.2 ENTORNO DE DESARROLLO

En este apartado vamos a exponer brevemente las características, tanto del hardware como de las herramientas software que hemos utilizado para el desarrollo del proyecto.

4.2.1 HARDWARE UTILIZADO

Para el desarrollo de la aplicación construida para este Trabajo Fin Grado se ha utilizado un ordenador portátil personal, con las siguientes características software:

- CPU: Intel Core i7 a 2,5 GHz
- RAM: 16 GB 1600 MHz DDR3
- GPU: Intel Iris Pro 1536 MB
- SO: OSX 10.11.6 El Capitan

Además del ordenador personal utilizado para el desarrollo, se ha hecho uso de 2 dispositivos móviles para la depuración de la aplicación con las siguientes características:

- Samsung J7:
  - CPU: Qualcomm Snapdragon 617 octa-core 1.6 GHz
  - GPU: Adreno 405
  - Versión Android: 6.0.1
  - Pantalla: 5,5''
- Motorolla:
  - CPU: Qualcomm Snapdragon 410 1.2 GHz quad-core
  - GPU: Adreno 306
  - Versión Android: 5.0.2
  - Pantalla: 4,5''

4.2.2 SOFTWARE UTILIZADO

Con fin de desarrollar la aplicación de este proyecto han sido utilizados unos programas y herramientas.

Las herramientas utilizadas has sido:

- **Entorno de desarrollo o IDE:** Android Studio ha sido la herramienta utilizada como entorno de desarrollo, ya que es la herramienta oficial propuesta por la empresa dueña de Android, Google, para el desarrollo de las aplicaciones Android. La versión de Android Studio utilizada ha sido la versión 2.2.2.

- **Kit de desarrollo Software:** El SDK utilizado es el que viene previamente integrado cuando instalamos la herramienta anterior, Android Studio.

- **Lenguaje utilizado para el desarrollo:** Para el desarrollo de aplicaciones Android el lenguaje de programación utilizado principalmente es Java, por lo que ha sido el que he utilizado.

- **Kit de desarrollo Java:** El entorno de desarrollo Android Studio necesita del JDK para un funcionamiento correcto. Por tanto, hemos necesitado instalarlo, obteniéndolo de la web de Oracle. La versión utilizada ha sido la 1.8.0.

ILUSTRACIÓN 30: VERSIÓN DE ANDROID STUDIO

4.2.3 ESTRUCTURA DE UN PROYECTO ANDROID

La estructura de todo proyecto Android realizado en Android Studio se organiza de la misma forma, por lo que nosotros hemos seguido dicha estructura para la aplicación realizada.

Si ponemos la vista de proyecto Android, podemos observar lo siguiente estructura:

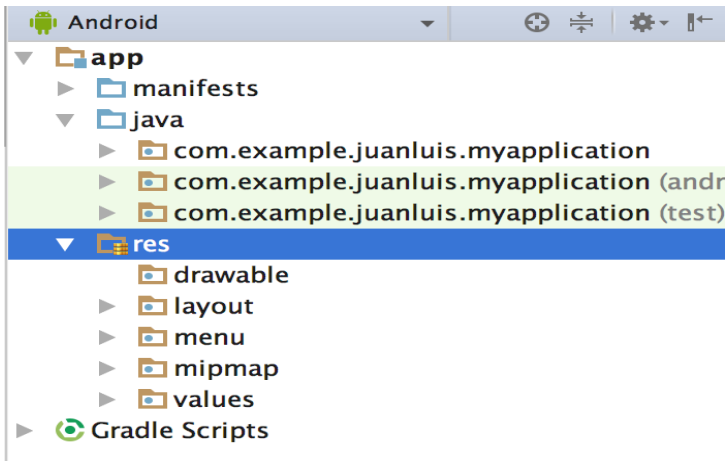


ILUSTRACIÓN 31: ESTRUCTURA DE UN PROYECTO ANDROID

El **paquete app** contiene todos los ficheros creados y necesarios para la aplicación desarrollada. Dentro de este paquete siempre encontraremos tres paquetes:

• **Manifests:** Dicho paquete contiene el fichero AndroidManifest.xml, que será el archivo donde son declarados los distintos componentes utilizados en la aplicación, es decir, la declaración de las activitys, los services, los broadcasts, etcétera.

• **Java:** Este paquete normalmente contiene 3 paquetes. El primero de ellos es donde se encuentra lo que hemos realizado mediante código fuente Java, es decir, las clases, broadcast, etcétera. Los otros 2 paquetes son generados para la realización de test y pruebas normalmente.

• **Res:** Este paquete contendrá todos los recursos necesarios por la aplicación. A continuación, explicamos cada una de las carpetas o paquetes que son creados dentro de res:

- Drawable: Este paquete contiene todas las imágenes y demás elementos gráficos necesarios por la aplicación. La carpeta drawable puede contener si es necesario algunos subdirectorios, cada uno de ellos creado para contener imágenes con distinta densidad de píxeles en la pantalla, para distintas pantallas:
  - drawable (recursos independientes de la densidad)
  - drawable-ldpi (densidad baja)
  - drawable-mdpi (densidad media)
  - drawable-hdpi (densidad alta)
  - drawable-xhdpi (densidad muy alta)
  - drawable-xxhdpi (la densidad más alta)

- Layout: Este paquete contendrá todos los archivos con extensión xml que actuarán posteriormente como las pantallas de la interfaz de usuario de la aplicación. Estos archivos serán posteriormente utilizados por las activitys o fragments de la aplicación, para darle formato a las pantallas que mostrarán las distintas activitys y fragments.

- Menu: Este paquete contiene ficheros con extensión xml utilizados para definir los menús de la aplicación, como por ejemplo los definidos en la barra de acciones, o ActionBar.

- Mimap: Este paquete contiene las imágenes que posteriormente serán utilizadas como iconos de la aplicación. Es decir, en esta carpeta se encuentra el icono que posteriormente se verá en la aplicación instalada en un Smartphone o tablet. Este paquete puede contener distintos paquetes en función de la densidad de pixeles de la pantalla, para contener iconos con distintos tamaños para las diferentes pantallas de dispositivos, nombrados igual que en la carpeta drawable.

- Values: Este paquete contiene ficheros con extensión xml, utilizados para la definición de los string, de los colores, etcétera.

- Xml: Aunque no aparece en la ilustración anterior, este paquete contiene archivos con extensión xml, generalmente utilizados para definición de estilos, definición de la pantalla de preferencias, etcétera.

En el **paquete Gradle Scripts** están contenidos los archivos necesarios e imprescindibles para la compilación del proyecto. Se creará un fichero Gradle por cada módulo de la aplicación y uno general para el proyecto. En estos ficheros se compilan las librerías, se definen los requisitos de los dispositivos que podrán ejecutar correctamente la aplicación, etcétera.

4.2.4 ESTRUCTURA ESPECÍFICA DE PETSMOBILE

Como ya hemos comentado anteriormente la estructura de un proyecto Android, nos centramos en este apartado en lo que nos diferencia de la estructura explicada anteriormente. Esto es la distinta distribución de los archivos Java creados con código fuente, los cuales se encuentran en el paquete Java, mencionado también en el apartado anterior.

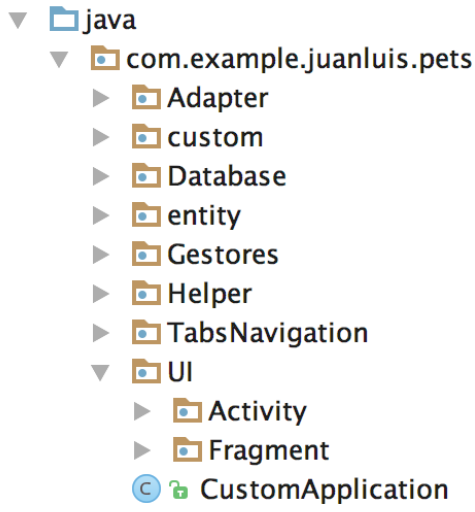


ILUSTRACIÓN 32: ESTRUCTURA DEL PROYECTO PETSMOBILE

A continuación, explicaremos cada uno de los directorios creados para contener todos los archivos con código fuente:

- **Adapter**: En este directorio hemos guardado las clases que extienden de utilizadas como adaptadores de Adapter, dándose el caso de que todas ellas son de BaseAdapter. Estos adaptadores posteriormente serán utilizados en las vistas de listview, para darles el formato que deseemos a cada elemento de la lista que rellena el listview.

- **Custom**: En dicho paquete se almacenan las clases creadas para tratar las notificaciones de la aplicación. Una de las clases que almacena, CustomNotificationManager, mediante un servicio que tiene creado, será la encargada de intervenir cuando la aplicación reciba una notificación. La otra clase que contiene, es la utilizada para construir las notificaciones.

- **Database**: En esta carpeta únicamente hay una clase almacenada, que es la que define la base de datos mediante la librería DBFlow, que explicamos en el apartado 4.3 Librerías utilizadas.

- **Entity**: Este paquete contiene todas las clases que definen una entidad de las utilizadas en la aplicación, ya sea por la base de datos o por la propia aplicación.

- **Gestores**: En este directorio están contenidos, como su propio nombre indica, todos los gestores de la aplicación, es decir, las clases que hemos creado para gestionar algunas tareas de la aplicación, teniendo en su mayoría de métodos las consultas a la base de datos.

- **Helper**: Esta carpeta contiene una clase definida para tener dentro métodos o rutinas que son llamadas en algunos sitios, para poderlas tener definidas comúnmente.

- **TabsNavigation**: Este directorio contiene algunas clases utilizadas para la definición de la barra de navegación que aparece al final de la pantalla en la aplicación, además de notificarle los fragments que va a cargar en la vista según se vaya navegando por dicha barra de navegación.

- **UI**: Este directorio es el que contiene todas las clases que componen la interfaz de usuario. A su vez, es contenedor de otros 2 directorios:
    - Activity: Este subdirectorio es contenedor de todas las activitys creadas en la aplicación para modificar la interfaz de usuario, o simplemente para realizar acciones que carguen datos para posteriormente mostrarlos en dicha activity, en otra, o en un fragment.
    - Fragments: Este subdirectorio contiene los fragments creados para modificar la interfaz de usuario y realizar las acciones necesarias para mostrar datos en las distintas pantallas de la aplicación.

- **CustomApplication**: Esto no es un directorio. Es la clase que se crea inicialmente al ejecutar la aplicación. Esta clase la aprovechamos para inicializar la base de datos e iniciar la conexión con el servidor de Nimbees.

4.3. LIBRERÍAS UTILIZADAS

En este apartado vamos a explicar las librerías que han sido utilizadas en la implementación de la aplicación, describiendo su uso, la forma de instalación y el motivo de la utilización.

4.3.1 GOOGLE MAPS API

Esta librería es la herramienta utilizada para la creación de mapas, además de la herramienta oficial de Google para estas tareas. Con esta herramienta, podremos insertar mapas en una aplicación de una forma bastante sencilla.

ILUSTRACIÓN 33: ICONO DE GOOGLE MAPS APIS

Fuente: https://twitter.com/googlemapsapi/status/638761004604616704

4.3.1.1 UTILIZACIÓN

En la aplicación PETSMobile, esta librería ha sido utilizada para mostrar un mapa en la pantalla de “Inicio” de la aplicación. En este mapa se pintará la ubicación actual del usuario. Además, se pintarán en el mapa unas marcas con el icono de la aplicación.

ILUSTRACIÓN 34: ICONO DE PETSMOBILE

Estas marcas en el mapa, simbolizarán que alguien desde ese lugar solicitó ayuda de una forma determinada, estando el usuario que está viendo dicha marca dispuesto a ayudar de dicha manera, además de a menos de 5000 metros del usuario que solicitó la ayuda.

Además, si haces click sobre la marca, nos mostrará la información y podremos volver a clickar encima de la información para abrir una conversación con la persona que solicitó esa ayuda.

4.3.1.2 INSTALACIÓN

Para poder instalar esta librería y posteriormente utilizarla en esta aplicación, hemos tenido que seguir los siguientes pasos (pueden variar algunos puntos para otras aplicaciones):

1. En el fichero Gradle tenemos que compilar las siguientes líneas:

```
compile 'com.google.android.gms:play-services-maps:9.8.0'
compile 'com.google.android.gms:play-services-location:9.8.0'
```

2. En el fichero AndroidManifest.xml debemos poner lo siguiente:

```
<meta-data
   android:name="com.google.android.maps.v2.API_KEY"
   android:value="****************************************" />
```

Donde los asteriscos simbolizan una clave API_KEY generada en Google.

Debemos crear un fragment que implemente un `OnMapReadyCallback` y posteriormente debemos sobrescribir el siguiente método:

```
public void onMapReady(GoogleMap googleMap);
```

Donde googleMap será el mapa que se pintará.

4.3.2 GLIDE

Glide es una librería utilizada en Android. Esta librería ha sido utilizada facilitar la gestión de las imágenes en la aplicación. Con esta librería podremos poner imágenes descargadas de servidores, que estén dentro de los recursos, que estén en la galería del dispositivo, o de cualquier otro sitio, pasándole únicamente la dirección de la imagen.

ILUSTRACIÓN 35: ICONO DE GLIDE

Fuente: https://expocodetech.com/glide-en-android/

4.3.2.1 UTILIZACIÓN

En la aplicación desarrollada, la librería en cuestión ha sido utilizada para poner una imagen en una vista ImageView mediante la dirección de la imagen. En concreto, la he utilizado para poner la imagen de perfil del usuario, almacenando en la base de datos la ruta que dicha imagen tiene en el dispositivo, y posteriormente cargándola en el ImageView simplemente pasándole la ruta almacenada.

```
imagenPerfil = (ImageView) findViewById(R.id.imagePerfil);
Glide.with(this).load(user.getRutaImagen()).into(imagenPerfil);
```

En estas líneas de código, podemos ver que mediante la librería ponemos la imagen de perfil, pasándole la actividad donde se realiza (this), la ruta de la imagen a cargar (que la cogemos de la ruta almacenada en el usuario user) y la vista donde ponerla (imagenPerfil), que es un ImageView como podemos ver en la primera línea de las dos.

4.3.2.2 INSTALACIÓN

Para la instalación y utilización de esta librería debemos seguir los siguientes pasos:

1. Compilar en el fichero de Gradle las siguientes líneas:

```
compile 'com.github.bumptech.glide:glide:3.7.0'
compile 'com.android.support:support-v4:25.0.0'
```

4.3.3 GSON

Gson es una librería para Java. Esta librería es utilizada para transformar objetos Java en un formato JSON que almacene todos los datos del objeto Java. También puede ser usado para convertir un JSON en su equivalente objeto Java.

Esta librería permite trabajar con objetos Java arbitrarios incluyendo los objetos preexistentes de los que no se tiene el código fuente, además de los objetos que hayamos creado mediante código fuente.

ILUSTRACIÓN 36: ICONO DE GSON
Fuente: http://blog.growthd.in.th/gson-library-android/

4.3.3.1 UTILIZACIÓN

En la aplicación construida para este proyecto, la librería GSON ha sido utilizada para transformar un objeto construido para mandar la notificación, CustomMessage, en un JSON, que será lo que realmente se mande en la notificación.

```
String mymessage = new Gson().toJson(cmessage, CustomMessage.class);
```

En la línea anterior de código, podemos ver como mediante la librería GSON transformamos un objeto Java, CustomMessage, en un String, mymessage, que será el que enviaremos posteriormente en la notificación.

Además, también lo hemos utilizado al contrario, es decir, para pasar de dicho JSON a un objeto Java, como podemos ver en la siguiente línea de código:

```
CustomMessage cm=new Gson().fromJson(content,CustomMessage.class);
```

Podemos ver que pasamos a cm, un objeto de tipo CustomMessage, un Json, que es el parámetro content, mediante la librería en cuestión. Esto lo realizamos al recibir una notificación, que sería content, para transformarlo en un objeto, que podremos procesar según los valores de sus atributos, obtenidos del JSON.

4.3.3.2 INSTALACIÓN

Para la instalación de esta librería no necesitamos compilar nada, simplemente, donde hagamos uso de la misma, debemos importar la siguiente línea de código:

```
import com.google.gson.Gson;
```

4.3.4 ORM PARA BASES DE DATOS

Para el almacenamiento de los datos de la aplicación era necesaria una base de datos. Una forma sencilla de realizar las consultas, adicciones y borrados en una base de datos es mediante un ORM ya definido. Un ORM es un mapeo de objeto-relacional, o lo que es lo mismo, es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.

Aunque el lenguaje SQL se utiliza en la gran mayoría de las bases de datos, existen múltiples variaciones de lenguajes en función de los distintos SGBD usados, por lo que teniendo un ORM que transforme las consultas del lenguaje que sea a un lenguaje común, facilitaríamos el tener que conocer todas las variaciones.

Las ventajas de utilizar un ORM frente a realizar las consultas normales a la base de datos son las siguientes:

- Facilidad y velocidad de uso
- Abstracción de la base de datos usada.
- Seguridad de la capa de acceso a datos contra ataques.

Existen varios ORM ya definidos. Para el desarrollo de esta aplicación, he investigado sobre 2 de ellos, que yo ya había utilizado en la empresa donde realicé las prácticas externas curriculares, ActiveAndroid y DBFlow.

Las diferencias y las ventajas y desventajas de uno frente a otro han sido lo que me han condicionado a utilizar uno en lugar del otro. Comparándolos, podemos observar que la utilización de DBFlow es, aunque en pequeña proporción, más complicada. Pero, como ventaja de DBFlow, vemos que es el ORM más potente de los que existen actualmente para Android. Además, probando con ambos, comprobamos que con ActiveAndroid obteníamos errores que no se obtenían con DBFlow. Por último, y como ventaja clara, y definitiva para elegir uno sobre otro, comprobamos que DBFlow es un ORM que está siendo permanentemente actualizado, para dar soporte a todas las versiones de Android que van apareciendo, por el contrario a ActiveAndroid, que lleva 2 años sin actualizar (desde que se escribió esta documentación) y que no da soporte a las versiones posteriores a Android 7.0.

Por tanto, la elegida para finalmente ser utilizada en la aplicación como ORM y para realizar las acciones con la base de datos, es DBFlow.

A continuación, explicaremos más claramente el ORM elegido.

4.3.4.1 DBFLOW

DBFlow es, como ya hemos dicho anteriormente, el mapeo de objeto-relacional que hemos utilizado para mapear los datos de la aplicación a una base de datos.

Uno de los problemas con los ORM existentes es que se basan en la reflexión de Java para definir modelos de base de datos, esquemas de tablas y relaciones de columnas. DBFlow es uno de los pocos ORMs que se basa estrictamente en el procesamiento de anotaciones para generar código Java basado en el framework SQLiteOpenHelper que evita este problema. Esto se traduce en un aumento del rendimiento en tiempo de ejecución.

ILUSTRACIÓN 37: ICONO DE DBFLOW

Fuente: https://javiercruzweb.com/2016/05/29/dbflow-fundamentos-de-implementacion/

4.3.4.1.1 UTILIZACIÓN

En PETSMobile hemos utilizado esta librería para realizar las consultas con la base de datos de las distintas entidades. Veremos un ejemplo de la utilización de esta librería para una consulta, una adicción y un borrado sobre la base de datos, para la entidad Pet.

En la siguiente consulta, estamos recuperando de la tabla Pet todas las tuplas en las que el atributo emailDuenio de dicha tabla tenga el valor del parámetro de entrada:

```
public static List<Pet> recuperarMisMascotas(String identifier) {
   return new Select()
         .from(Pet.class)
         .where(Pet_Table.emailDuenio.eq(identifier))
         .queryList();
}
```

Nota: Para hacer la sentencia Where, tenemos que hacerlo sobre el nombre de la tabla, guión bajo, la palabra Table, es decir, en este caso Pet_Table. Para poder realizar esto, debemos hacer el import de dicha tabla, es decir, debemos poner, en este caso:

```
import com.example.juanluis.pets.entity.Pet_Table;
```

En la siguiente línea de código podemos observar el método con el que añadimos una mascota a la base de datos:

```
public static void addmascota(Pet pet) {
    pet.save();
}
```

Vemos que le pasamos la mascota ya inicializada con todos sus atributos, añadiéndola a la base de datos con la sentencia save().

Por último, vemos el proceso utilizado para el borrado de una mascota de la base de datos en la siguiente línea de código:

```
public static void deletemascota(Pet pet) {
    pet.delete();
}
```

Observamos que le hemos pasado por parámetro la mascota que queremos borrar, haciendo luego la sentencia delete() sobre dicha mascota. Otra forma de realizar el borrado de una tupla, habiendo sido utilizada también en este proyecto, es pasando por parámetro únicamente el identificador de la entidad, haciendo posteriormente una consulta de la tupla que tenga ese identificador en la tabla, y realizando el delete() sobre la tupla devuelta por la consulta anterior.

Podríamos ver muchas diferencias entre todas las sentencias distintas que hemos realizado para esta aplicación con la librería que estamos explicando, pero por no hacer muy extenso este subapartado, limitamos a las 3 consultas más básicas, pudiendo verse el resto de consultas en el código fuente de la aplicación del proyecto.

4.3.4.1.2 INSTALACIÓN

Para la instalación y posterior utilización de DBFlow, tenemos que realizar los siguientes pasos:

1. Poner en el fichero Gradle, de la aplicación general, lo siguiente:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'

    }
}
```

2. Poner en el mismo fichero Gradle lo siguiente:

```
allprojects {
    repositories {
        jcenter()
        maven { url "https://jitpack.io" }
    }
}
```

3. Poner en el fichero Gradle, del módulo donde vayamos a utilizar DBFlow, lo siguiente:

```
        apply plugin: 'com.neenbedankt.android-apt'
        def dbflow_version = "4.0.0-beta1"

dependencies {
    apt "com.github.Raizlabs.DBFlow:dbflow-processor:${dbflow_version}"
    compile "com.github.Raizlabs.DBFlow:dbflow-core:${dbflow_version}"
    compile "com.github.Raizlabs.DBFlow:dbflow:${dbflow_version}"
}
```

En dbflow_version, hemos decidido poner esa, porque es la última versión creada de DBFlow hasta el día en que fue escrita esta documentación.

4. Iniciar la base de datos en la clase que inicia la aplicación, que en mi caso es la clase CustomApplication, de la siguiente forma:

```
public class CustomApplication extends Application {
    public void onCreate() {
        super.onCreate();
        FlowManager.init(new FlowConfig.Builder(this).build());
    }
}
```

5. Debemos crear una clase que contenga el siguiente código:

```
import com.raizlabs.android.dbflow.annotation.Database;

@Database(name = MyDatabase.NAME, version = MyDatabase.VERSION,
foreignKeysSupported = true)
public class MyDatabase {

    public static final String NAME = "MyDatabase";

    public static final int VERSION = 1;
    public static final String MODULE = "Data";
}
```

6. Con cada entidad para la que queramos crear una tabla en la base de datos debemos crear una clase, que tenga los atributos que deseemos, y posteriormente debemos poner las siguientes anotaciones:
   - **@Table(database=MyDatabase.class)**: Para definir que esa clase será una entidad que tenga una tabla en la base de datos MyDatabase, que fue el nombre que pusimos a la clase en el punto 5 de ésta instalación.
   - **@Column**: Para definir que ese atributo es una columna en la tabla.
   - **@PrimaryKey**: Esto define que ese atributo será la clave principal de la tabla. Solamente puede haber un atributo que tenga dicha etiqueta.
     - Además, podemos ponerle: @PrimaryKey(autoincrement=true), si queremos que este identificador o clave principal sea un atributo que se vaya autogenerando incrementalmente cuando vayamos insertando tuplas a la tabla en cuestión.

A continuación, podemos ver un ejemplo para la tabla Pet:

```
@Table(database = MyDatabase.class)
public class Pet extends BaseModel {


    @Column
    @PrimaryKey(autoincrement = true)
    private int id;

    @Column
    private String Name;
    @Column
    private String Tipo;

}
```

Nota: Para la creación de tablas intermedias, de N a M, aunque existe una etiqueta en DBFlow que lo gestiona, he decidido crear una tabla con los identificadores de las dos tablas en cuestión y un identificador incremental para la tupla, gestionándolo yo, ahorrando así los posibles fallos que me pueda dar dicha etiqueta.

4.3.5 MÓDULO BOTTOMNAVIGATION

En este subapartado voy a explicar brevemente el módulo bottomnavigation que está añadido en la aplicación. Este módulo fue implementado en la empresa donde realicé las prácticas externas. Con él, conseguimos gestionar fácilmente las distintas vistas, según los diferentes fragments que tengamos creados, mediante una barra de navegación.

Por tanto, esta librería nos resuelve el cambio de la vista en la pantalla, en función al apartado que seleccionemos de la aplicación.

4.3.5.1 UTILIZACIÓN

Este módulo ha sido utilizado, como ya hemos dicho en la breve explicación anterior, para modificar la vista de la pantalla cuando seleccionemos las distintas pestañas de la aplicación en la barra de navegación.

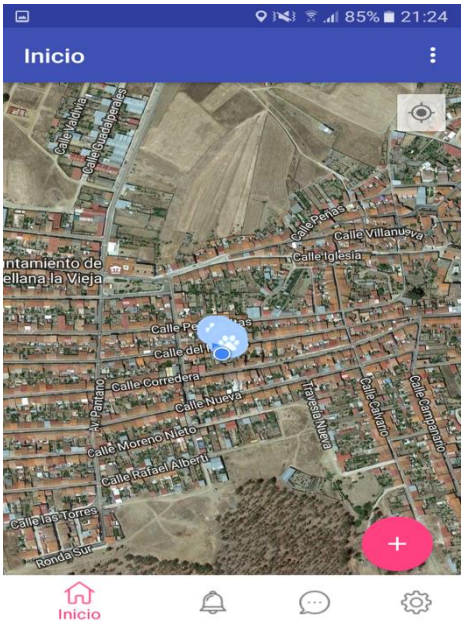


ILUSTRACIÓN 39: PANTALLA DE INICIO EN PETSMOBILE

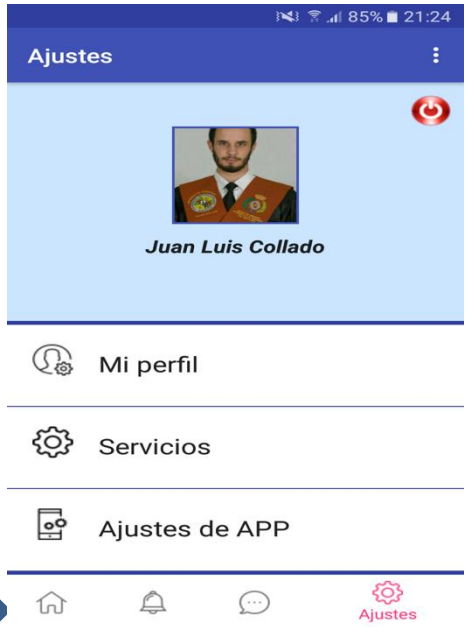


ILUSTRACIÓN 38: PANTALLA DE AJUSTES EN PETSMOBILE

En las ilustraciones anteriores, vemos como con esta librería, al clickar en Ajustes en la barra de navegación (barra creada por la librería también), se cambia la vista, poniendo la vista de ajustes y quitando la vista de Inicio.

Nota: Mencionar que esta librería, carga el fragment que pinchamos y los consecutivos, para facilitar la velocidad de respuesta si cambiamos a uno de los de al lado del seleccionado actualmente.

4.3.5.2 INSTALACIÓN

Para instalar esta librería, únicamente tenemos que estar en posesión de este módulo, construido por un tercero, y posteriormente meterlo en el proyecto como un módulo. Además, tenemos que añadir en el Gradle del módulo donde utilicemos esta librería lo siguiente:

```
compile project(path: ':bottomnavigation')
```

Teniendo en cuenta que el módulo se llama bottomnavigation.

4.3.6 NIMBEES

Nimbees es el servidor, proporcionado por el tutor de este Trabajo Fin de Grado, para la gestión de las notificaciones push de la aplicación.

ILUSTRACIÓN 40: ICONO DE NIMBEES
Fuente: https://api.nimbees.com/login

Mediante este servidor, el dispositivo construye una notificación cuando solicita ayuda, le pide a éste que la distribuya, y éste se la envía a los usuarios que cumplan los requisitos que lleva dicha notificación. Además, cuando un usuario mande un mensaje de chat a otro usuario, también se construirá una notificación de tipo mensaje, que será enviada a Nimbees para que se lo envíe al usuario receptor.

Para la utilización de Nimbees, tenemos que crear una cuenta en la siguiente web: https://api.nimbees.com/register

Posteriormente, se nos creará una cuenta, que nos permitirá crear una aplicación, para utilizarla con este proyecto. Para poder acceder a nuestra cuenta, lo hacemos mediante la siguiente web:

https://api.nimbees.com/login , en la que veremos la siguiente pantalla y deberemos poner las credenciales de la cuenta creada anteriormente, para acceder a la consola:

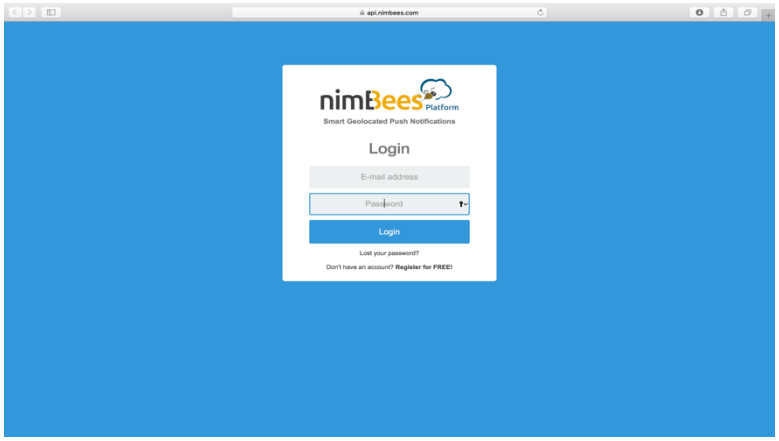


ILUSTRACIÓN 41: ACCESO A CONSOLA DE NIMBEES

Con la consola podremos crear una aplicación, en la que gestionar las notificaciones y los usuarios de la aplicación, entre otras cosas. A continuación podemos ver la consola de la aplicación PETSMobile:

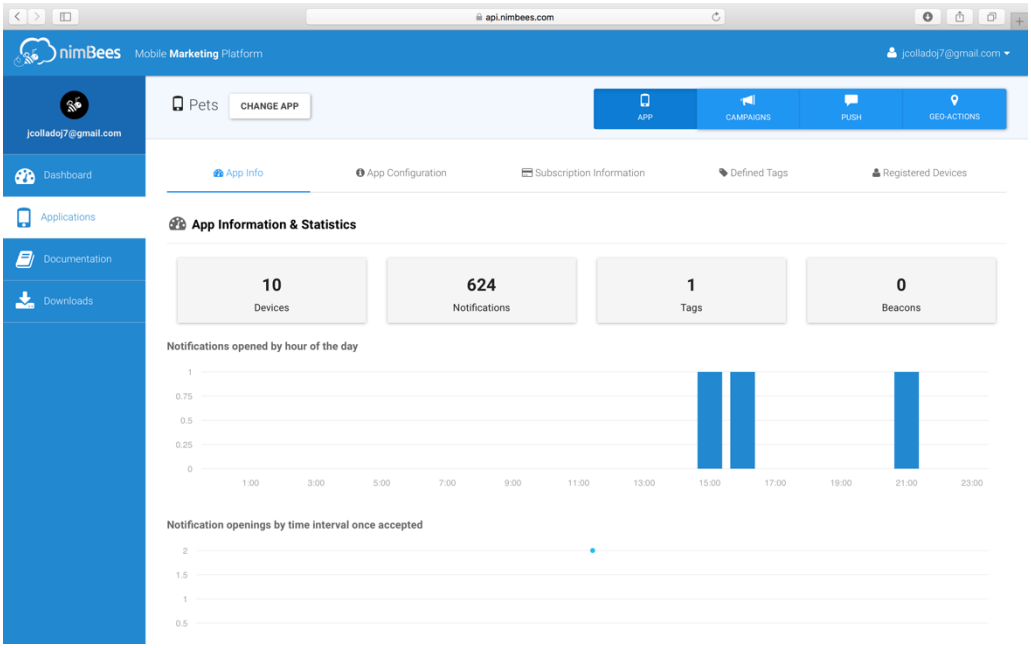


ILUSTRACIÓN 42: CONSOLA DE NIMBEES

Vemos que aparece información de la aplicación. Además, tenemos una pestaña para cambiar la configuración de la aplicación. También tenemos información sobre la subscripción a Nimbees, en la que actualmente podemos ver que estamos con una cuenta gratuita. Consecutivo, tenemos la pestaña de TAGs, que explicaremos en el subapartado de utilización de esta librería. Por último, podemos ver que hay una pestaña para ver los dispositivos que hay registrados en la aplicación.

Además, tenemos la posibilidad de mandar notificaciones push, lanzar una campaña, etcétera, para que sean enviadas a los usuarios que deseemos.

4.3.6.1 UTILIZACIÓN

Como ya hemos mencionado anteriormente, este servidor ha sido utilizado para la distribución de las notificaciones de un usuario solicitando ayuda al resto de usuarios. Además, se ha utilizado para enviar los mensajes de mensajería instantánea de un usuario a otro, mediante notificaciones a un usuario en concreto.

Para la gestión de las notificaciones, hemos hecho uso de los TAGs, mencionados anteriormente. Estos TAGs sirven para actualizarle el valor a un TAG en concreto a un usuario determinado, admitiendo así que ese dispositivo tiene ese activado ese valor para ese TAG. Para verlo con mayor sencillez pondremos un ejemplo, el utilizado realmente en la aplicación.

Inicialmente creamos un TAG llamado “opción”, y lo creamos de la forma que explicaremos posteriormente.

4.3.6.1.1 EJEMPLO DE UTILIZACIÓN

Un usuario pone en la aplicación que está dispuesto a Adoptar perros de tamaño mediano, comportamiento bueno y sexo macho.

La aplicación actualizará el TAG “opción” para el dispositivo que lo puso, poniéndole como valor: adopcionPerroBuenoMachoMediano.

Si posteriormente pone que está dispuesto a pasear un gato hembra, de tamaño grande y comportamiento bueno, el TAG “opción” tendrá, además del valor añadido anteriormente, el siguiente valor para ese dispositivo: paseoGatoBuenoHembraGrande.

Cuando otro usuario mande una notificación solicitando ayuda porque necesita que alguien adopte su perro macho, de comportamiento bueno y tamaño mediano, se generará una notificación con un TAGFilter (diciendo que el TAG “opción” tiene que tener el valor adopcionPerroBuenoMachoMediano). Esta notificación será enviada a

Nimbees, y Nimbees se encargará de enviársela a los usuarios que en el TAG “opción” tengan el valor del TAGFilter puesto en la notificación.

Por tanto, la notificación si llegará al primer usuario, no llegando a los usuarios que no tengan ese valor para el TAG “opción”.

4.3.6.1.2 CREACIÓN DE UN TAG

Para la creación de un TAG, en la consola de Nimbees de la aplicación, desde la pestaña de Defined Tags, debemos darle a “Add new tag”, poniendo el nombre que queramos, el tipo de valor que tendrá el TAG (en nuestro caso, fue text, ya que es un String), y seleccionando Simple o Multiple, teniendo en cuenta que simple significa que únicamente podrá tener un valor para ese tag cada usuario, y múltiple que podrá tener más de un valor por usuario y TAG.

4.3.6.1.3 ACTUALIZACIÓN DEL VALOR DE UN TAG

Para actualizar el valor de un TAG para un dispositivo, lo hacemos haciendo uso del siguiente servicio que nos ofrece Nimbees:

```
NimbeesClient.getTagManager().addServerTagValue("opcion", valueOpcion, new
NimbeesCallback<Boolean>() {
    @Override
    public void onSuccess(Boolean aBoolean) {
    }

    @Override
    public void onFailure(NimbeesException e) {
    }
});
```

Teniendo en cuenta que el parámetro “opcion” debe ser el nombre del TAG en el servidor, y que valueOpcion debe ser el valor que se le pondrá a ese dispositivo para el TAG “opcion”. Evidentemente valueOpcion debe ser del tipo que se creó el TAG, en este caso es de tipo String, como ya hemos mencionado.

4.3.6.1.4 BORRADO DEL VALOR DE UN TAG

Para borrar el valor de un TAG a un dispositivo, por ejemplo, porque el usuario ya no está dispuesto a cuidar perros macho, de comportamiento bueno y tamaño mediano, debemos hacerlo mediante el siguiente servicio, proporcionado por Nimbees:

```
NimbeesClient.getTagManager().deleteServerTagValue("opcion", valueOpcion, new
NimbeesCallback<Boolean>() {
```

```
    @Override
    public void onSuccess(Boolean aBoolean) {

    }
    @Override
    public void onFailure(NimbeesException e) {
    }
});
```

Teniendo en cuenta, como ya hemos dicho para la adicción de un valor a un TAG, que el parámetro “opcion” debe ser el nombre del TAG en el servidor, y que valueOpcion debe ser el valor que se le pondrá a ese dispositivo para el TAG “opcion”. Evidentemente valueOpcion debe ser del tipo que se creó el TAG, en este caso es de tipo String, como ya hemos mencionado.

Tanto para la actualización de un valor, como para el borrado de un valor de un TAG, la petición realizada al servidor devolverá la respuesta por el onSuccess() si pudo realizarla, o por el onFailure() si no pudo resolver la petición.

4.3.6.1.5 ENVÍO DE UNA NOTIFICACIÓN

Para enviar una notificación, el usuario debe crear un servicio solicitando ayuda de una forma concreta, y para una o varias de sus mascotas. A continuación, se construirá una notificación, que llevara un conjunto de filtros:

- LocationFilter: Para mandar la notificación únicamente a los usuarios que se encuentren en una distancia menor a un radio establecido, del sitio donde se encuentra quien solicita la ayuda. Lo hacemos de la siguiente forma:

  ```
  lf = new LocationFilter(servicio.getLatitude(), servicio.getLongitude(), (double)
  30000);
  ```

Donde le estamos pasando por parámetros la latitud y longitud de donde el usuario solicitó la ayuda, y el radio máximo de distancia, que en este caso son 30000 metros.

- TAGFilter: Para que la notificación se mande únicamente a los usuarios que tengan en el TAG en cuestión el valor que vaya en este filtro. Para la adopción de una mascota, lo hacemos de la siguiente forma:

  ```
  tfopcionServicio = new TagFilter("opcion", TagFilter.TagFilterOperator.EQ,
  "adopcion"+pet.getTipo()+pet.getComportamiento()+pet.getSexo()+pet.get
  Tamanio());
  ```

Donde le estamos pasando por parámetros el nombre del TAG, diciendo que el valor de los dispositivos que reciban la notificación debe ser exactamente igual al pasado a continuación. Como vemos, el valor lo construimos con el tipo de ayuda, en este caso adopción, y concatenándole la mascota, con sus características. Evidentemente siempre concatenamos esto en el mismo orden para construir el valor del TAG.

Para añadir estos 2 filtros, y alguno más si lo necesitásemos, debemos hacerlo definiendo una lista de objetos, en el que añadiremos todos los filtros que tendrá la notificación, de la siguiente forma:

```
List<Object> filters = new ArrayList();
filters.add(tfopcionServicio);
filters.add(tfopcionServicio);
```

Para enviar la notificación que deseemos, con todos estos filtros lo hacemos de la siguiente forma:

Construimos primero un CustomMessage, con todos los atributos que queramos rellenos, con los valores del servicio creado por el usuario. En el atributo type le ponemos el valor “custom”.

Posteriormente transformamos ese objeto a JSON.

```
String mymessage = new Gson().toJson(cmessage, CustomMessage.class);
```

A continuación, mediante el servicio que nos proporciona Nimbees, mandamos la notificación, de la siguiente forma:

```
NimbeesClient.getNotificationManager().sendNotification(mymessage,
MessageContent.NotificationType.CUSTOM, filters, new NimbeesCallback<Integer>() {
    @Override
    public void onSuccess(Integer integer) {
    }

    @Override
    public void onFailure(NimbeesException e) {
    }
});
```

Donde el parámetro mymessage es el JSON construido para enviar, el tipo CUSTOM es para decir que es de tipo notificación normal, en vez de mensaje y los filtros son los que hemos creado anteriormente.

Si la notificación es bien enviada a Nimbees, nos devolverá la respuesta por el onSuccess() del callback, mientras que si hay algun fallo, volverá por el onFailure().

4.3.6.1.6 ENVIAR MENSAJE CHAT

Para enviar una notificación de tipo mensaje lo realizamos de la siguiente forma:

```
List<String> emailusers=new LinkedList<>();
emailusers.add(emailUserReceptor);
UserNameListFilter lf=new UserNameListFilter(emailusers);
List<Object> filters = new ArrayList();
filters.add(lf);
```

Creo una lista de emails receptores, a la que añadiré únicamente la persona a la que quiero enviar el mensaje. Creo un filtro de tipo UserNameListFilter, en el que pasaré esa lista de usuarios (que tendrá únicamente el usuario que de verdad queremos que la reciba) y añadimos el filtro a una lista de objetos, como hacíamos para la notificación normal.

En el CustomMessage construido para el atributo type ponemos el valor “mensaje”, para diferenciarlo de las notificaciones normales.

Para enviar el mensaje lo hacemos con el servicio sendNotification que nos proporciona Nimbees, el mismo que hemos utilizado antes, pero pasándole esta lista de filtros.

4.3.6.2 INSTALACIÓN

Para instalar Nimbees en la aplicación y que posteriormente funciones correctamente tenemos que seguir los pasos que veremos en el tutorial ya creado por los creadores del Servidor Nimbees. El tutorial está compuesto de varias partes, que podremos ver en las siguientes web:

- http://doc.nimbees.com/index.php/Android_library_usage/Integration_guide/Register_for_Google_Cloud_Messaging
- http://doc.nimbees.com/index.php/Android_library_usage/Integration_guide/Create_the_project_on_nimBees_Platform
- http://doc.nimbees.com/index.php/Android_library_usage/Integration_guide/Configure_the_mobile_library_(Android_Studio)

He preferido no explicar esta instalación ya que nadie podrá explicarlo mejor que los creadores del propio servicio.

4.4 PUNTOS ESPECÍFICOS DE LA IMPLEMENTACIÓN DE LOS CASOS DE USO

En este apartado vamos a describir algunos de los puntos más específicos, los métodos realizados para solventar casos de uso, que tienen mayor dificultad o son de mayor interés para ser tratados en esta documentación, teniendo en cuenta que ya han sido algunos de ellos resueltos en algunos de los anteriores subapartados, como por ejemplo el envío de notificaciones y de mensajes.

4.4.1 REGISTRO O INICIO DE SESIÓN

Inicialmente, en la clase MainActivity, que es la primera que se inicia en la aplicación, se comprueba si hay algún usuario en el dispositivo con una sesión iniciada. Esto lo realizamos mediante SharedPreferences, almacenando ahí el email del usuario que tenga la sesión iniciada, o dejándolo como vacío si no hay ninguna sesión. Lo hacemos mediante SharedPreferences porque se mantienen durante todo el transcurso de la aplicación en el dispositivo, aun cerrando la aplicación, borrándose únicamente cuando sea pedido mediante el proyecto, o cuando sea borrada la aplicación del Smartphone.

```
sharedPreferences=getSharedPreferences("emailUserLogueado",MODE_PRIVATE);
emailUser=sharedPreferences.getString("emailUserLogueado","");
```

En las líneas anteriores hemos podido ver como consultamos del sharedPreferences una preferencia almacenada, a la que hemos llamado “emailUserLogueado”. Posteriormente si esta consulta nos devuelve un email, sabremos que hay una sesión iniciada, por lo que pasaremos a cargar todos sus datos, como explicaremos posteriormente, mientras que si no obtenemos ningún email, pasaremos de MainActivity a la clase PreSplashActivity, que es la clase donde solicitamos el inicio de sesión o registro en caso de no estar registrado.

En la clase PreSplashActivity comprobaremos si anteriormente ha habido algún usuario que ha iniciado sesión en ese Smartphone (ya que solo puede haber un usuario registrado con cada dispositivo). Lo hacemos mediante sharedPreferences, de la siguiente forma:

```
sharedPreferences=getSharedPreferences("emailUserLogueado",MODE_PRIVATE);
emailprimerregistro=sharedPreferences.getString("emailprimerregistro","");
```

Mediante estas dos líneas anteriores, obtenemos el email del último usuario que estuvo con una sesión iniciada en el dispositivo. Si obtenemos algún email, sabremos que ya hay un usuario almacenado en la base de datos, y que por tanto inicio sesión alguna vez en este móvil. Mostraremos un cuadro de dialogo al usuario, para que sepa que si elige un email distinto al que se registró anteriormente, se borrarán todos los datos almacenados anteriormente, teniendo que registrarse de nuevo con el email elegido en esta selección. Será decisión del usuario, elegir el email que ya estaba registrado e iniciar sesión, cargando los datos ya almacenados, o elegir un email distinto, eliminándose los datos almacenados y registrándose de nuevo. Para la solicitud del email de registro o inicio de sesión, lo hacemos mediante el OAuth de Google, es decir, mostrándole al usuario las cuentas que tiene con Google en el dispositivo y dándole la opción de iniciar la sesión o el registro con una de esas cuentas. Esto lo hacemos de la siguiente forma:

1. Llamamos al siguiente método:

```
private void showGoogleAccountPicker() {
   Intent googlePicker = AccountPicker.newChooseAccountIntent(null, null,
         new String[]{GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE}, true, null, null,
null, null);
   startActivityForResult(googlePicker, PICK_ACCOUNT_REQUEST);
}
```

2. En el onActivityResult() de la clase, hacemos lo siguiente:

```
if (requestCode == PICK_ACCOUNT_REQUEST && resultCode ==
MainActivity.RESULT_OK) {
   emailUser=data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);
}
```

Sabiendo que el email obtenido es el email de la cuenta de Google seleccionada por el usuario.

A partir de ese email seleccionado comenzaremos el registro o inicio de sesión. Consultamos en la base de datos, mediante DBFlow, si el usuario con ese correo electrónico está almacenado. Si lo estuviese, iniciaremos la sesión, cargando sus datos y saltando a la clase MainActivity, mientras que si no está en la base de datos, iniciaremos el proceso de registro del usuario saltando a la clase Registrase.

Para el registro, cogeremos los datos rellenos del usuario, cuando este acepte que desea registrarse, comprobando si están todos rellenos, y comprobamos si el usuario puso que tiene o no mascotas. En caso de no tener mascotas, almacenaremos al usuario en la base de datos, registraremos al usuario en Nimbees y guardaremos su email en SharedPreferences, por ese orden. En caso de tener mascotas, pasaremos a la clase RegistrarMascotas, para darle la oportunidad al usuario de

registrar también a sus mascotas en su perfil, haciendo el mismo proceso que anteriormente, y además, almacenando las mascotas en la base de datos.

Para el registro en Nimbees lo hacemos mediante el servicio register que nos ofrece Nimbees, de la siguiente forma:

```
NimbeesClient.getUserManager().register(username, new
NimbeesRegistrationCallback() {

    @Override
    public void onSuccess() {

        //borra todos los valores de tag del usuario, por si los tuviese de otra vez que
estuvo registrado
        gestorNotificaciones.deleteAllValueTagsUser();
        registerUserinApp(username);
        SharedPreferences.Editor editor = mainActivity.getApplicationContext().
                getSharedPreferences("emailUserLogueado",
Context.MODE_PRIVATE).edit();
        editor.putString("emailUserLogueado", username)
                .commit();
        mainActivity.getSharedPreferences("emailUserLogueado",
Context.MODE_PRIVATE).edit();
        editor.putString("emailprimerregistro",username)
                .commit();

        Intent intent=new Intent(mainActivity,MainActivity.class);
        intent.putExtra("emailUserActual",username);
        mainActivity.startActivity(intent);
        mainActivity.finish();
    }

    @Override
    public void onFailure(NimbeesException e) {

        List<Pet> list=GestorMascotas.recuperarMisMascotas(username);
        for(int i=0;i<list.size();i++){
            list.get(i).delete();
        }
        deleteUserById(username);
        registererror(username);
    }
});
```

En el código anterior, observamos que hacemos la petición de registro en Nimbees, con el email que le hayamos pasado (username), teniendo en cuenta que si el callback nos devuelve un error en el registro, debemos borrar al usuario de la base de datos, mostrándole un mensaje al usuario de error en el registro, para que lo intente posteriormente y compruebe su conexión a internet. Si el callback devuelve por el onSuccess(), sabremos que se ha registrado correctamente y proporcionaremos el paso del usuario

a las funciones de la aplicación, pasándole al MainActivity, y almacenaremos el email en SharedPreferences, como hemos comentado anteriormente.

Para ver cómo hacemos el almacenamiento en sharedPreferences, podemos verlo en el código anterior también.

En caso de no tener que registrarse, porque el usuario con ese email ya tiene perfil, le daremos paso a MainActivity, cargando sus datos de la siguiente forma.

4.4.1.1 SPLASHSCREEN

La aplicación cuando se abra con un inicio de sesión o una sesión ya iniciada, en MainActivity ocultará en la vista del usuario todos los campos y funcionalidades que el usuario podrá ver posteriormente, mostrando una vista con un SplashScreen. Esta pantalla da la sensación al usuario de que se están cargando los datos del usuario que quiere entrar en la aplicación, dándole además información de lo que falta para iniciar la aplicación, mediante un progressBar. Para ocultar las funcionalidades y mostrar la vista de carga, lo hacemos así:

1. Creamos un Asynctask, en el que con el onPreExecute, quitamos las funcionalidades de la pantalla al usuario y mostramos el SplashScreen, de la siguiente forma:

```
@Override
protected void onPreExecute() {
    super.onPreExecute();
    splash.setVisibility(View.VISIBLE);
    getSupportActionBar().hide();
    progressBar.setMax(100);
    ocultarFabButton();
    ocultarBottomNavigation();
    ocultarViewPager();
    listanotificaciones=new LinkedList<>();
}
```

2. En el doInBackground, para hacerlo en segundo plano, cargamos los datos del usuario, de la siguiente forma:

```
@Override
protected Boolean doInBackground(Void ... params) {
    publishProgress(10);
    cargarMisMascotas();
    try {
        Thread.sleep(500);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    publishProgress(30);
    cargarNombresChats();
    try {
```

```
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        publishProgress(60);
        cargarMisServicios();
        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        publishProgress(80);
        obtenerNotificaciones();
        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        publishProgress(100);
        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return true;
    }
```

Teniendo en cuenta que los try/catch con el Thread.sleep(ms) es para darle tiempo al usuario a ver esta pantalla cuando tiene pocos datos que cargar y la aplicación lo hace sin dar tiempo al usuario de verla. Además los publishProgress(progreso) lo que hacen es actualizar el progressBar poniendo el valor pasado por parámetro en la barra. Mientras tanto, vamos cargando los datos del usuario, como podemos ver, sus mascotas, chats, servicios, etcétera.

3. Por último, en el onPostExecute(), que se ejecutará cuando finalice de cargar los datos, quitaremos el SplashScreen y pondremos la vista normal con todas las funcionalidades, de la siguiente forma:

```
@Override
protected void onPostExecute(Boolean result) {
    super.onPostExecute(result);
    if(result){
        mostrarBottomNavigation();
        mostrarFabButton();
        mostrarViewPager();
        getSupportActionBar().show();
        splash.setVisibility(View.GONE);
    }
}
```

Al finalizar esto, el usuario tendrá todos sus datos cargados, y además podremos ver las funcionalidades de la aplicación. En la siguiente ilustración podemos ver el SplashScreen:

ILUSTRACIÓN 43: SPLASHSCREEN DE PETSMOBILE

4.4.2 CREACIÓN DE SERVICIO DE AYUDA

Cuando un usuario necesita ayuda o está dispuesto a ayudar a otro usuario, deberá crear un servicio de ayuda. Para esto, inicialmente, el usuario debe tener el GPS activado, con la finalidad de capturar la ubicación del dispositivo con el que se quiere solicitar o proporcionar la ayuda. La ubicación la capturamos con el servicio getLocation() proporcionado por Nimbees, de la siguiente forma:

```
NimbeesLocationManager.NimbeesLocationListener mNimbeesLocationListener = new
NimbeesLocationManager.NimbeesLocationListener() {
    @Override
    public void onGetCurrentLocation(Location location) {
        mLocation = location;

    }

    @Override
    public void onError(NimbeesException e) {
    }
};
NimbeesClient.getLocationManager().obtainCurrentLocation(mNimbeesLocationListener);
```

Cumpliendo este requisito, podemos dividir la creación de servicios de ayuda en 2 servicios distintos, los que el usuario ofrece su ayuda y los que solicita ayuda de otro usuario. Explicaremos ambos por separado:

4.4.2.1 SERVICIO DE SOLICITUD DE AYUDA

Cuando un usuario necesita ayuda, deberá seleccionar el tipo de ayuda que desea, además de las mascotas, dentro de las que tiene registradas, quiere que estén implicadas en esa ayuda. Para ello, tras elegir que desea “Ser ayudado” y elegir el tipo de ayuda, le aparecerá una lista con todas sus mascotas, para que pueda seleccionar una o varias de ellas. Al tener que elegir las mascotas, lo haremos mediante la clase “ElegirMascotas”, donde saldrá la lista mencionada anteriormente.

Comprobaremos el tipo de ayuda, dividiéndolo en 4 tipos distintos, ya que algunos de ellos necesitan fechas y otros no:

- **Cuidado:** Necesita fecha de inicio y fecha de fin del cuidado.
- **Paseo:** Necesita fecha del día del paseo.
- **Perdida:** No necesita fecha, ya que simplemente es notificar que se ha perdido una o varias mascotas.
- **Adopción:** No necesita fecha, ya que es notificar que se quiere dar una o varias mascotas en adopción.

Debido a esto, tenemos 3 constructores de Servicio. Uno sin fechas, uno con una fecha y otro con dos fechas.

Teniendo en cuenta lo anterior, si el servicio que desea crear es de los que no requieren fecha, lo crearemos, almacenando en la base de datos el servicio, así como en la tabla intermedia MascotasServicio, en la que almacenaremos una tupla para cada una de las mascotas implicadas en este servicio, con el identificador de la mascota, el del servicio y el identificador único de tupla. Si por el contrario necesita fecha, nos mandará a una activity que muestra un calendario al usuario, dándole la posibilidad de elegir las fechas. Al iniciar esa otra activity, controlaremos si se necesita una fecha o dos fecha (ya sea paseo o cuidado), dándole la posibilidad de elegir dos fechas si fuese un servicio de cuidado. Cuando esa activity finalice, devolverá el día, mes y año seleccionado en el calendario, con lo que ya podremos crear el servicio y generar las notificaciones. Las notificaciones se generan y envían como explicamos en el apartado 4.3.6.1.5 Envío de una notificación.

4.4.2.1 SERVICIO DE PROPORCIÓN DE AYUDA

Cuando un usuario está dispuesto a ofrecer su ayuda a otros usuarios, deberá seleccionar “Ayudar” y el tipo de ayuda que desea ofrecer. Además, el usuario deberá elegir qué tipo de animal o animales está dispuesto a ayudar, es decir, al no solicitar ayuda y ofrecerla, no podrá elegir dentro de sus mascotas, si no que deberá crear los tipos de animal, con sus características propias que está dispuesto a ayudar. Para elegir estos animales, lo haremos mediante la clase “TipoMascotas”, donde podrá crear todas las mascotas que está dispuesto a ayudar, almacenando éstas mascotas en una tabla de la base de datos llamada “PetFicticio”, ya que necesitamos tenerlas almacenadas y no son mascotas reales, para guardar en la tabla de “Pet”.

Realizaremos las mismas comprobaciones para la elección de las fechas, igual que hemos explicado en el subapartado anterior.

A continuación, tras elegir el tipo de ayuda y crear las mascotas ficticias e implicadas en el servicio, comprobamos si el servicio que se desea crear es de los que no requieren fecha, creándolo entonces, almacenando en la base de datos el servicio, así como en la tabla intermedia MascotasServicio, en la que almacenaremos una tupla para cada una de las mascotas ficticias implicadas en este servicio, con el identificador de la mascota ficticia, el del servicio y el identificador único de tupla. Si por el contrario necesita fecha, nos mandará a la activity “Calendar” para elegir la fecha o fechas si necesitase dos fechas. Cuando esta activity finalice, devolverá el día, mes y año seleccionado en el calendario, con lo que ya podremos crear el servicio y anunciar que ese dispositivo está dispuesto a ayudar de esa forma y para esas mascotas. Para anunciar que estamos dispuestos a ayudar, lo hacemos actualizando el valor del TAG “opción” para ese dispositivo, con los valores que obtenemos de este servicio, como lo explicamos anteriormente, en el apartado 4.3.6.1.3 Actualización del valor de un TAG.

4.4.3 CONVERSACIONES

Para el almacenamiento de los mensajes de las distintas conversaciones en la base de datos, lo he solucionado almacenando cada uno de los mensajes, con un atributo que sea el email de la persona con la que tuve ese mensaje, además de un atributo booleano para saber si el mensaje es enviado o recibido. Además, en la base de datos, tenemos una tabla en la que almacenamos únicamente el email de cada uno de los usuarios y la fecha en la que se mandó el último mensaje con ese usuario. Así, cuando quiero mostrar las distintas conversaciones que tengo, en la pestaña de “Mensajes”, no tengo que cargar todos los mensajes, si no que cargo

únicamente todos los nombres de los usuarios con los que he tenido una conversación (y no la he borrado, ya que se pueden borrar), minimizando así el coste de la consulta. Cuando posteriormente elija una conversación para abrirla, realizaré la consulta de todos los mensajes que haya almacenados con este usuario, mediante un asynctask (para que no hubiese problemas en la carga, si tuviésemos muchos mensajes con este usuario). La consulta para obtener solamente los mensajes con este usuario es:

```
public static List<Chat> obtenerConversacionConUser(String emailotrouser){
   return new Select()
         .from(Chat.class)
         .where(Chat_Table.userName.eq(emailotrouser))
         .queryList();
}
```

4.4.4 AJUSTES DE APP

Los ajustes de aplicación, para PETSMobile, se corresponden con la posibilidad para el usuario de elegir si desea que cuando reciba una notificación de otro usuario, el dispositivo vibre, suene, ambas o ninguna, dando la misma posibilidad para la recepción de mensajes de otro usuario.

Para la realización de los ajustes de la aplicación, lo hemos hecho mediante una clase que extiende de PreferenceFragment, cargando las opciones que el usuario ponga para su aplicación en las SharedPreferences. La clase mencionada anteriormente es la siguiente:

```
public class AjustesApp extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getSupportActionBar().setTitle("Ajustes de APP");
      getSupportActionBar().show();
      getFragmentManager().beginTransaction().replace(android.R.id.content, new
MyPreferenceFragment()).commit();
   }

   public static class MyPreferenceFragment extends PreferenceFragment
   {
      @Override
      public void onCreate(final Bundle savedInstanceState){
         super.onCreate(savedInstanceState);
         addPreferencesFromResource(R.xml.ajustesapp);
      }
   }
}
```

Observamos cómo esta clase, pone en el fragment que estamos viendo un fragment que creamos posteriormente y que extiende de PreferenceFragment, que es un tipo de fragment creado por Android, para tratar las preferencias del usuario, almacenándolo en las SharedPreferences, que como hemos dicho, son permanentes para la vida de la aplicación en el dispositivo. En ese fragment ponemos las preferencias desde un archivo con extensión xml creado en los recursos de la aplicación. El archivo xml comentado anteriormente divide las preferencias en 2 categorías de preferencia (para mensajes y para notificaciones), como vemos a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">

<PreferenceCategory
   android:title="Notificaciones de Pets"
   android:key="ajustesapp">
   <SwitchPreference
      android:key="opcion1"
      android:title="Sonido"
      android:defaultValue="true"
      android:summary="¿Desea que su dispositivo tenga sonido cuando reciba
notificaciones de otro usuario solicitando ayuda?">
   </SwitchPreference>

   <SwitchPreference
      android:key="opcion2"
      android:title="Vibración"
      android:defaultValue="true"
      android:summary="¿Desea que su dispositivo tenga vibración cuando reciba
notificaciones de otro usuario solicitando ayuda?">

   </SwitchPreference>
</PreferenceCategory>

   <PreferenceCategory
      android:title="Notificaciones de Mensajes"
      android:key="ajustesusuario">
      <SwitchPreference
         android:key="opcion3"
         android:title="Sonido"
         android:defaultValue="true"
         android:summary="¿Desea que su dispositivo tenga sonido cuando reciba
mensajes de otro usuario?">

      </SwitchPreference>

      <SwitchPreference
         android:key="opcion4"
         android:title="Vibración"
         android:defaultValue="true"
         android:summary="¿Desea que su dispositivo tenga vibración cuando reciba
mensajes de otro usuario?">
      </SwitchPreference>
   </PreferenceCategory>
</PreferenceScreen>
```

4.4.5 RECEPCIÓN DE NOTIFICACIONES

Para la recepción de las notificaciones, lo hacemos mediante la clase CustomNotificationManager que extiende del manager de notificaciones de Nimbees. Esta clase ofrece un servicio que se encarga de recibir las notificaciones, es decir, toda notificación que llegué a la aplicación PETSMobile, pasará por dicho servicio o método y por tanto podrá ser tratada ahí como lo deseemos. En las siguientes líneas de código podemos apreciarlo:

```
public void handleCustomMessage(long idNotification, String content, Map<String, String>
additionalContent) {

   CustomMessage cm=new Gson().fromJson(content,CustomMessage.class);
   if(cm.getType()!=null){
      type=cm.getType();
      if(type.contentEquals("custom")){
            title=cm.getTitle();
            texto=cm.getMessage();
            showNotification(title,texto,type);
      }else{
         if(type.contentEquals("mensaje")){
            title=cm.getTitle();
            texto=cm.getMessage();
            showNotification(title,texto,type);
         }
      }
   }
```

Aunque se ha disminuido el código real en las líneas copiadas anteriormente, basta para mostrar el funcionamiento del método. Vemos que recibe por parámetros un String content. Ese parámetro es el JSON enviado por el otro usuario en la notificación, por lo que lo tratamos con la librería GSON y lo pasamos a un objeto de tipo CustomMessage. Posteriormente tratamos el atributo type de dicho CustomMessage y vemos si es de tipo “custom” o de tipo “mensaje”, para tratarlo de formas distintas (aunque en este fragmento de código parezca que es de la misma manera, en el código real es de forma distinta).

En el showNotification() crearemos un intent con los datos a mostrar al usuario como extras. Posteriormente ese intent se meterá en un pendingIntent, que será el que le mostrará la notificación al usuario.

Al intent que hemos metido en el PendingIntent le ponemos el siguiente flag:

```
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
```

Esto lo hacemos, para que cuando el usuario pinche en la notificación, esta tarea se superponga a la que estaba haciendo y borre la anterior. Además, para que no se abran 2 veces la misma actividad, en el apartado de la actividad en el manifest, ponemos:
android:finishOnTaskLaunch="true".

5. MANUAL DE USUARIO

En este quinto capítulo de la documentación se ofrece una guía o manual de usuario con la finalidad de que el usuario pueda aprender a navegar correctamente por la aplicación y conozca todas las funcionalidades de las que ésta dispone.

Primeramente, se va a realizar una breve descripción de la aplicación. A continuación, se indican los requisitos necesarios para utilizar correctamente la aplicación. Después veremos la instalación de la aplicación.

Posteriormente, se realizará una navegación completa por toda la aplicación, con 2 dispositivos móviles (para ver cómo funciona la red social entre dos personas), con imágenes sobre el funcionamiento, además de la descripción de cada funcionalidad. De esta manera facilitaremos al usuario la comprensión de la utilización de la aplicación.

5.1 DESCRIPCIÓN DE LA APLICACIÓN

PETSMobile es una red social para dueños de mascotas. Con ella, unos usuarios podrán pedir ayuda para sus mascotas, mientras que otros podrán ofrecerla. Cuando un usuario decida contactar con otro que necesita una ayuda que él está dispuesto a proporcionar, podrá hablarlo por mensajería instantánea para ayudarlo en la medida de los posible.

5.2 REQUISITOS NECESARIOS

Para la correcta utilización de la aplicación realizada para este proyecto, el usuario debe disponer de lo siguiente:

- Un teléfono inteligente con Android como sistema operativo.
- La versión de Android debe ser igual o superior a 5.0 Lollipop.
- Tener acceso a internet.
- El dispositivo debe poseer GPS.

5.3 GUÍA DE INSTALACIÓN

Para la instalación de la aplicación, se deberá tener un archivo con extensión apk para poder ser instalado en el dispositivo, ya que actualmente la aplicación no se encuentra en el Google Play Store, lo que nos facilitaría la instalación a simplemente descargarla como hacemos con el resto. Para descargar el archivo apk, podemos hacerlo a partir del siguiente enlace: https://drive.google.com/open?id=0B0VYoqsuW4Z8LWdCUGhwZ29URVE

Puede darse el caso de que el dispositivo nos anuncie que la aplicación viene de una fuente desconocida, por lo que tendremos que activar desde los ajustes del móvil la instalación de aplicaciones con orígenes desconocidos. Tras activar esto, se realizará la instalación correctamente.

Tras descargar la apk desde nuestro dispositivo, si la versión Android del dispositivo es menor a la versión 6.0, veremos lo siguiente:

ILUSTRACIÓN 44: SOLICITUD DE PERMISOS PARA PETSMOBILE EN ANDROID MENOR A 6.0

5.4 MANUAL DE LA APLICACIÓN

Aunque no es completamente necesario, puesto que la aplicación es fácil de manejar y además contiene las preguntas frecuentes, para que el usuario pueda ser ayudado con ellas, en este apartado vamos a realizar un manual de la utilización de todas las funcionalidades, paso por paso, de la aplicación.

Inicialmente, según abrimos la aplicación por primera vez, lo primero que veremos será la pantalla de inicio de sesión. En esta pantalla, además del logotipo de la aplicación, tenemos un único botón, con el que podremos iniciar la sesión mediante las cuentas de Google que tiene el dispositivo. Al clickarlo vemos la pantalla de la derecha.

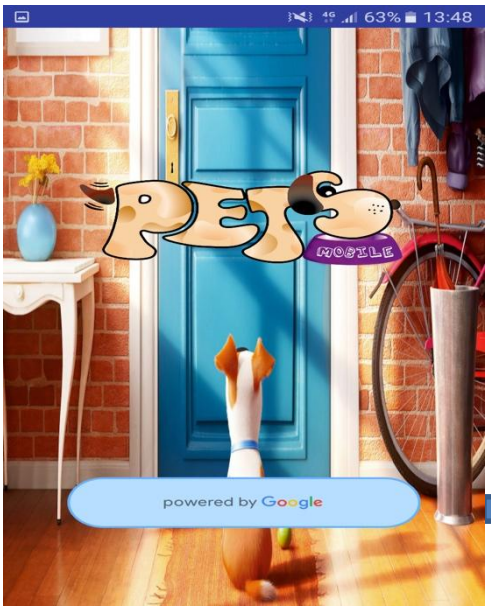


ILUSTRACIÓN 45: PANTALLA INICIO DE SESIÓN EN PETSMOBILE

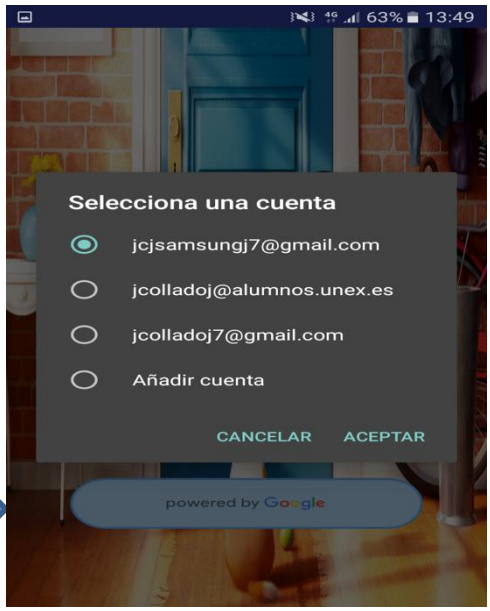


ILUSTRACIÓN 46: CUENTAS DE GOOGLE PARA INICIO DE SESIÓN

En esa pantalla elegimos el email con el que iniciaremos sesión, si es que ya tuviese una cuenta en este dispositivo con dicho email, o con el que nos registraremos en caso de no tener cuenta creada hasta el momento.

Posteriormente, al ser la primera vez que entramos en la aplicación, deberemos registrarnos, mediante la siguiente pantalla:

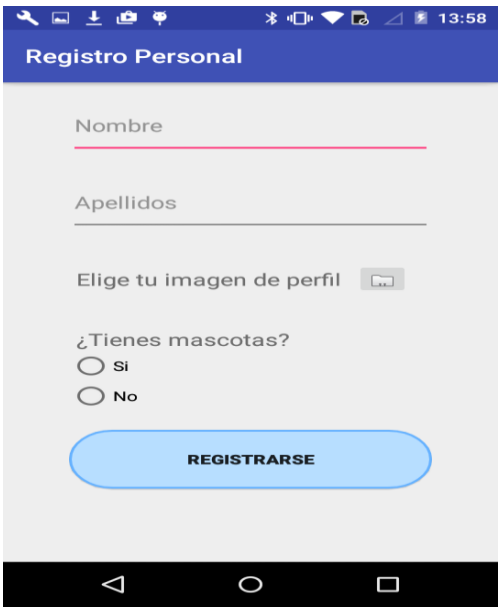


ILUSTRACIÓN 47: FORMULARIO REGISTRO PERSONAL EN PETSMOBILE

En la pantalla anterior, rellenaremos los datos personales del usuario. Si el usuario elige no tener mascotas, se registrará e iniciará la aplicación, pero si elige que sí, tendrá que comenzar a registrar sus mascotas mediante las siguientes pantallas:

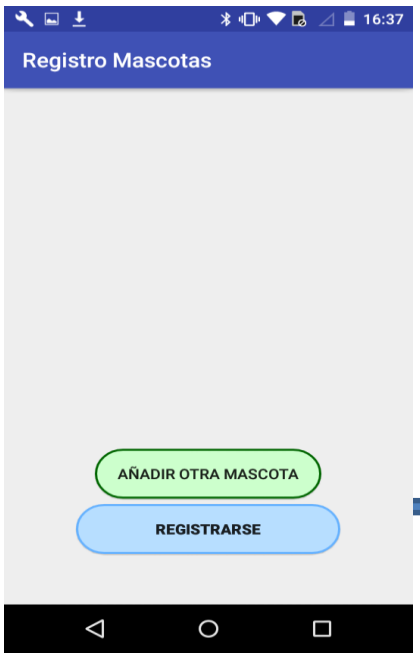


ILUSTRACIÓN 48: PASO 1 DEL REGISTRO MASCOTAS

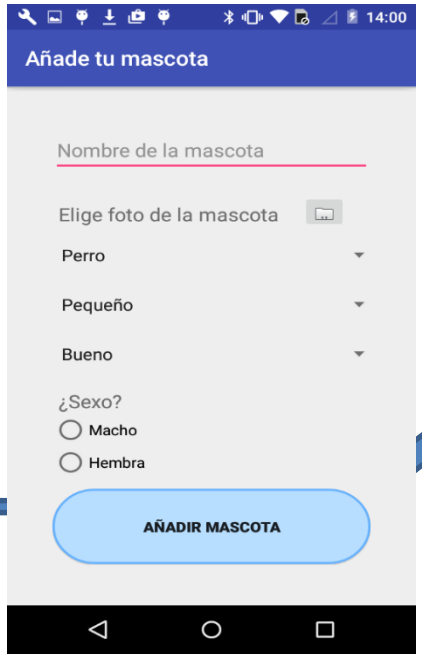


ILUSTRACIÓN 50: PASO 2 DEL REGISTRO DE MASCOTAS

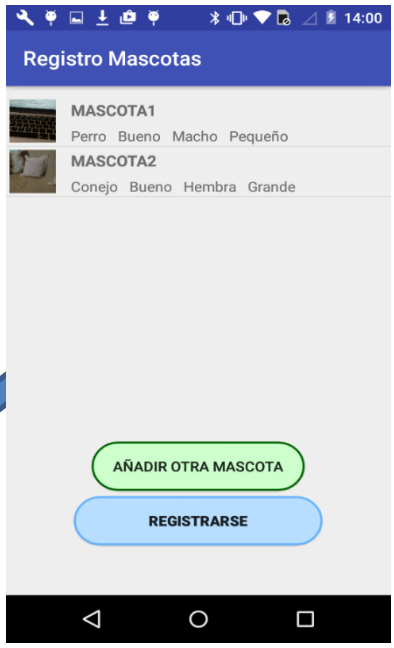


ILUSTRACIÓN 49: PASO 3 DEL REGISTRO DE MASCOTAS

Si hacemos un click largo sobre la mascota de la lista, nos da la oportunidad de borrarla de la lista.

Cuando hayamos registrado todas las mascotas que queramos, le damos al botón “Registrarse”, lo que hará que el usuario se registre correctamente, con los datos introducidos anteriormente y con estas mascotas.

Tras registrarse, se abrirá la pantalla del splashscreen, que ya habíamos visto y comentado anteriormente, por lo que no vuelvo a poner la imagen de la pantalla. Una vez esa pantalla haya cargado los datos del usuario, se abrirá la aplicación con todas sus funcionalidades. La primera pantalla que saldrá, es la del mapa, en el que podremos ver las notificaciones que hayamos recibido anteriormente, y que además estén cercanas. Si pinchamos una de las marcas del mapa, podremos observar un resumen de la notificación, y si pinchamos encima de ese resumen, nos mostrará la notificación completa y nos dará la posibilidad de abrir un chat con este usuario. Podemos verlo en las siguientes ilustraciones:

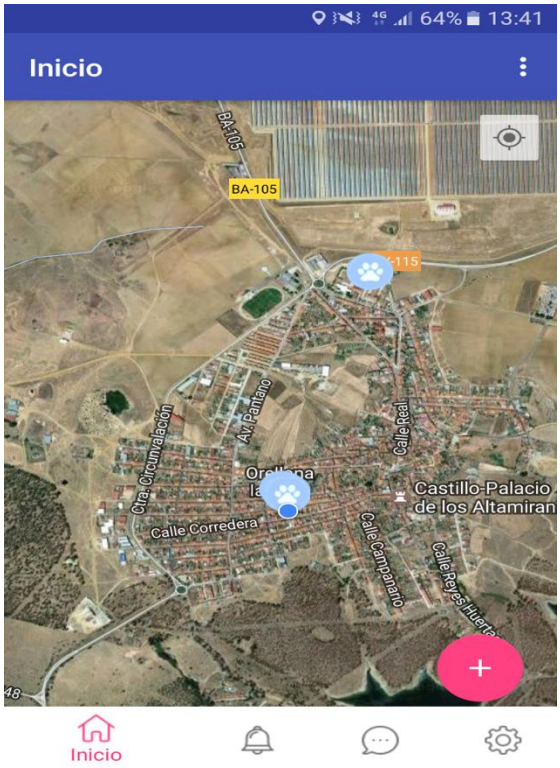


ILUSTRACIÓN 52: PANTALLA PRINCIPAL DE PETSMOBILE

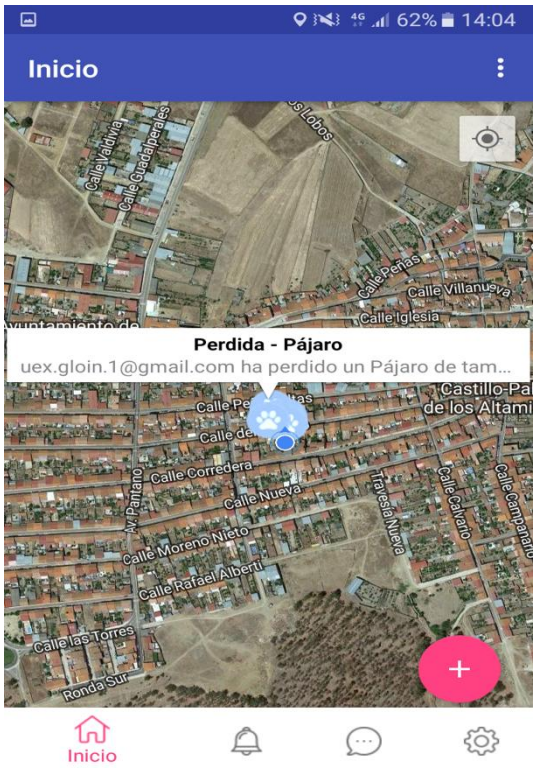


ILUSTRACIÓN 51: CLICK EN MARCA DEL MAPA EN PANTALLA PRINCIPAL

Si pinchamos en el botón rosado con el símbolo “+”, podremos crear un servicio de solicitud de ayuda, o para ofrecer ayuda. Lo vemos en la siguiente ilustración:

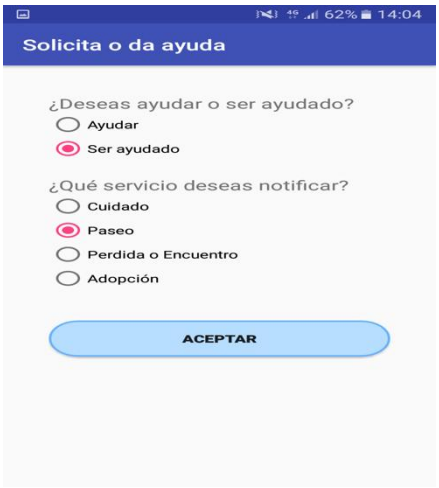


ILUSTRACIÓN 53: PANTALLA CREACIÓN DE SERVICIO EN PETSMOBILE

En la pantalla anterior, seleccionaremos si deseamos ayudar, o ser ayudados, y en que estamos dispuestos a ayudar o ser ayudados. Tras darle a “Aceptar”, tendremos 2 opciones:

- Si es ayudar: Debemos elegir los tipos de mascotas que estamos dispuestos a ayudar.

ILUSTRACIÓN 56: PASO 1 PARA AYUDAR

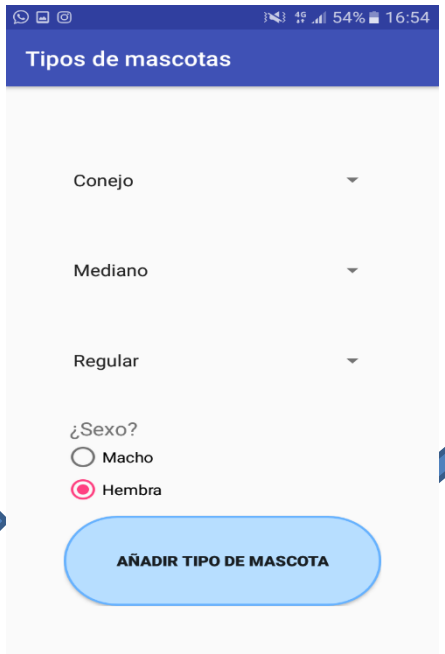


ILUSTRACIÓN 55: PASO 2 PARA AYUDAR

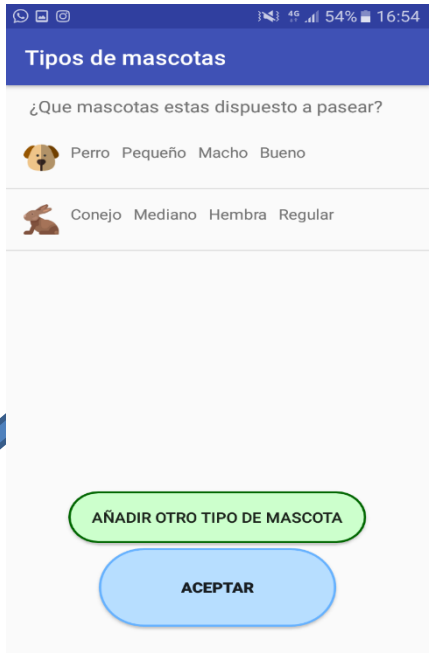


ILUSTRACIÓN 54: PASO 3 PARA AYUDAR

Si hacemos click largo sobre una de las mascotas de la lista, podremos borrarla.

- Si es ser ayudado: Debemos elegir de entre las mascotas que tenemos registradas, las que realmente queremos que estén implicadas en la ayuda.

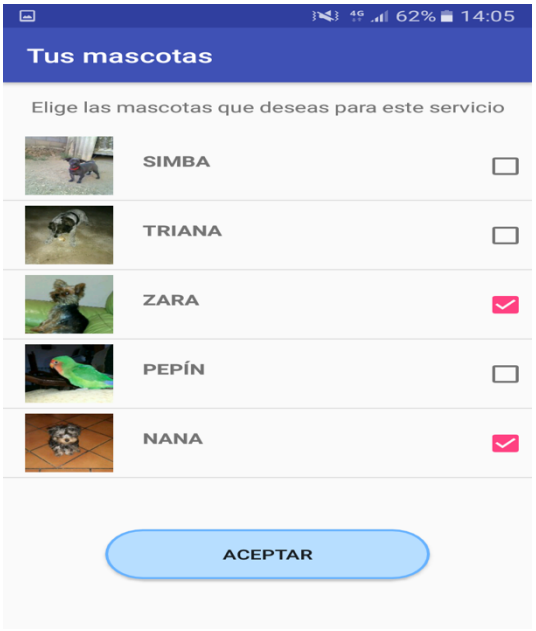


ILUSTRACIÓN 57: PASO 1 PARA SER AYUDADO

Independientemente de si es ayudar o ser ayudado, si el servicio es para cuidado o paseo, tendrá que elegir fechas, eligiendo la de inicio y fin para el servicio de cuidado y eligiendo únicamente una fecha para el paseo. Podemos ver en la siguiente ilustración la elección de la fecha del paseo:

ILUSTRACIÓN 58: ELECCIÓN DE FECHA EN EL CALENDARIO

Tras elegir las fechas del servicio, si hay que hacerlo, o tras elegir las mascotas si no hay que elegir fechas, se nos mostrará un mensaje como que el servicio ha sido creado.

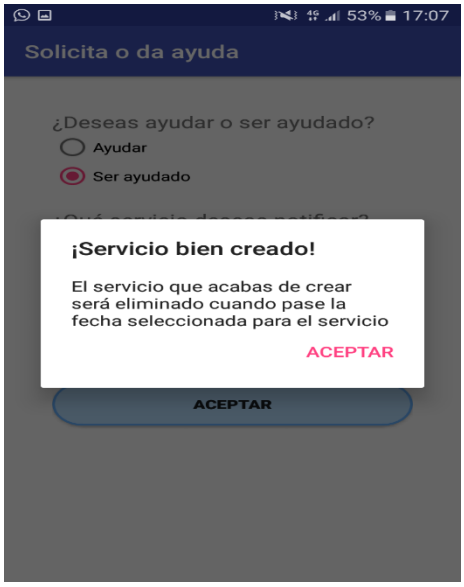


ILUSTRACIÓN 59: MENSAJE DE CONFIRMACIÓN DE CREACIÓN DEL SERVICIO

Como podemos ver en ese mensaje, si el servicio era de los que tienen fecha, cuando pase la fecha de ese servicio, la propia aplicación se encargará de borrarlo y desactivarlo automáticamente. Teniendo fechas o no, podemos borrar ese servicio desde la pestaña de “Servicios” en “Ajustes”.

Si el servicio creado era para ayudar, se actualizará este dispositivo en Nimbees, para recibir notificaciones de ese tipo. Si es para ser ayudado, se mandará la notificación a los usuarios que tengan que recibirlo. En la siguiente captura vemos la recepción de una notificación:

ILUSTRACIÓN 60: RECEPCIÓN DE UNA NOTIFICACIÓN

Tras recibir esta notificación, si la abrimos, veremos lo siguiente:

ILUSTRACIÓN 61: NOTIFICACIÓN MOSTRADA AL USUARIO

Si rechazamos esta notificacion, el usuario que solicita la ayuda no sabrá nada, y nosotros podremos tener la oportunidad de no ayudar, porque no lo vemos posible para nosotros. Si aceptamos, se abrirá una conversación con este usuario. Vemos en las siguientes capturas como un dispositivo habla a otro, éste le contesta, y el primero recibe el mensaje contestado por el segundo:

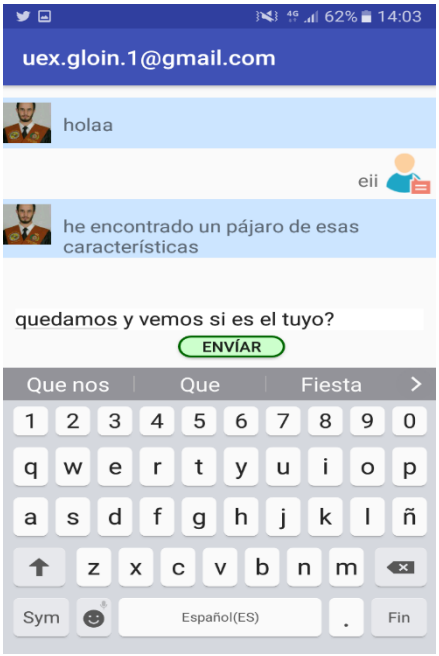


ILUSTRACIÓN 63: ENVÍO DE MENSAJE

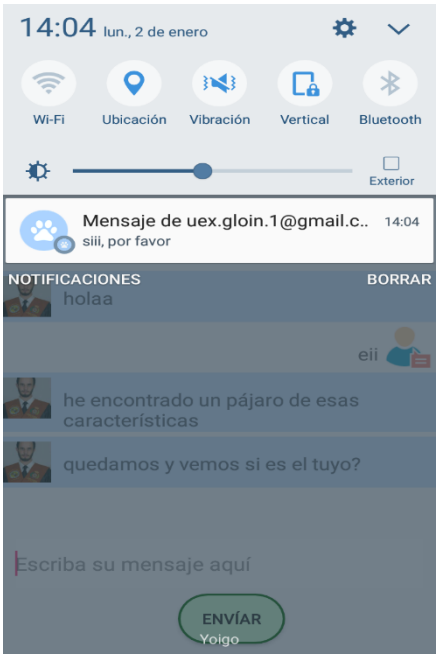


ILUSTRACIÓN 62: RECEPCIÓN DE MENSAJE

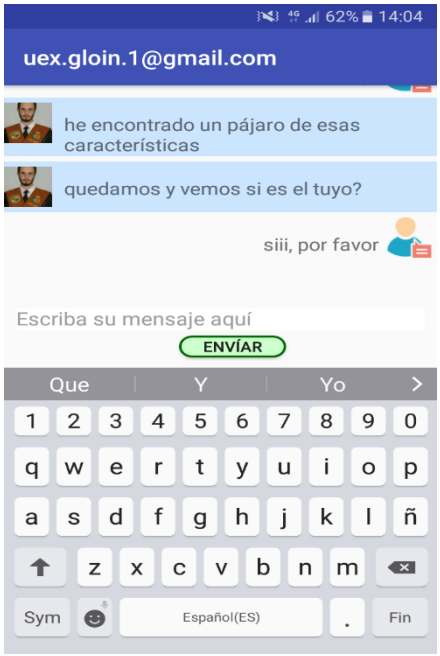


ILUSTRACIÓN 64: MOSTRADO DEL MENSAJE RECIBIDO

También, podemos ver la pantalla de conversaciones, pinchando desde la pantalla inicial en la pestaña de “Mensajes” y veremos lo siguiente:

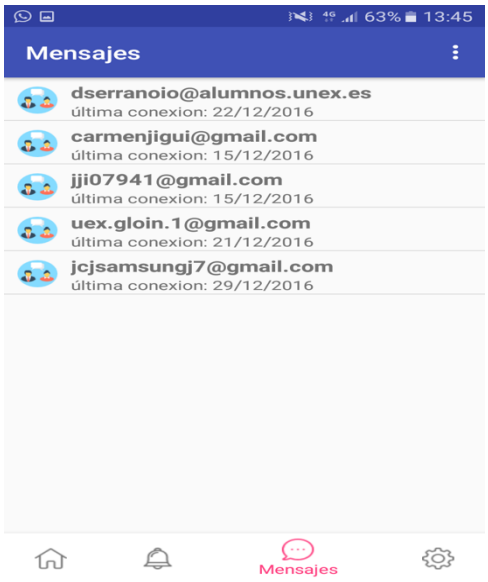


ILUSTRACIÓN 65: PANTALLA DE CONVERSACIONES EN PETSMOBILE

Si pinchamos en cualquier conversación, ésta se abrirá y mostrará todos los mensajes, así como dará la posibilidad de mandar mensajes nuevos. Además, si hacemos click largo sobre la conversación, nos dará la posibilidad de borrarla.

También podemos ver la lista de notificaciones recibidas por el dispositivo, desde la pestaña de “Notificaciones”. Lo vemos en la siguiente ilustración:

ILUSTRACIÓN 66: PANTALLA DE NOTIFICACIONES EN PETSMOBILE

Si pinchamos en una notificación, nos dará la oportunidad de abrir un chat con el usuario que nos la envió.

También podemos ver cierta información, así como modificarla, sobre la cuenta personal del usuario y los ajustes que tiene establecidos en la aplicación, desde la pestaña de “Ajustes”

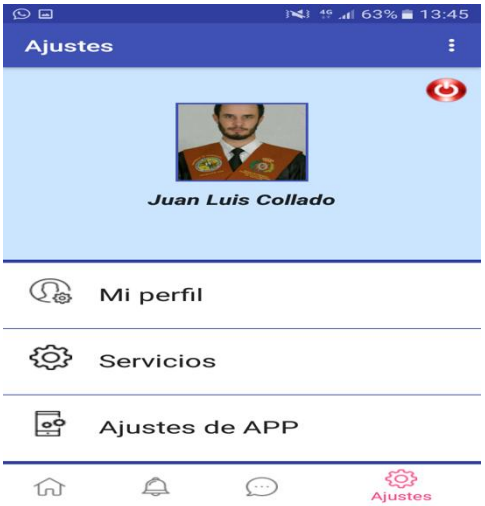


ILUSTRACIÓN 67: PANTALLA DE AJUSTES EN PETSMOBILE

En esta pantalla, tenemos 3 apartados:

- **Mi perfil:** Tenemos información personal del usuario, así como de las mascotas que tenemos registradas. En la siguiente ilustración vemos que podemos modificar el nombre y apellidos, así como la imagen de perfil. Para que se quede modificado, tendríamos que pulsar el botón verde “Modificar Perfil”, y ya quedaría modificado todo lo que hayamos cambiado del perfil. También podemos borrar el perfil, desde el botón rojo “Borrar tu perfil”, lo que hará que se borre el usuario permanentemente.

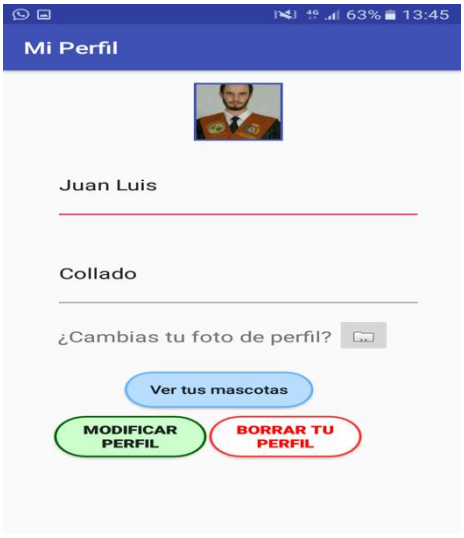


ILUSTRACIÓN 68: PANTALLA DE DATOS PERSONALES

El botón azul “Ver tus mascotas” nos permite lo que dice el propio botón. Además nos permite borrar alguna de las mascotas o añadir alguna nueva. Lo vemos en la siguiente pantalla:

ILUSTRACIÓN 69: PANTALLA VISUALIZACIÓN DE MIS MASCOTAS

Si pinchamos en el botón “Añadir Mascota”, nos dará la oportunidad de añadir una mascota nueva a la lista que se nos muestra. Si hacemos click largo sobre alguna de las mascotas, nos permitirá borrar esta mascota de la lista. Para que los cambios realizados en esta lista se guarden y se hagan las modificaciones en las mascotas del usuario, debemos darle click al botón “Aceptar”, que sirve para confirmar los cambios.

- **Servicios:** En esta pantalla podemos ver los servicios de solicitud o proporción de ayuda que tenemos. La vemos a continuación:

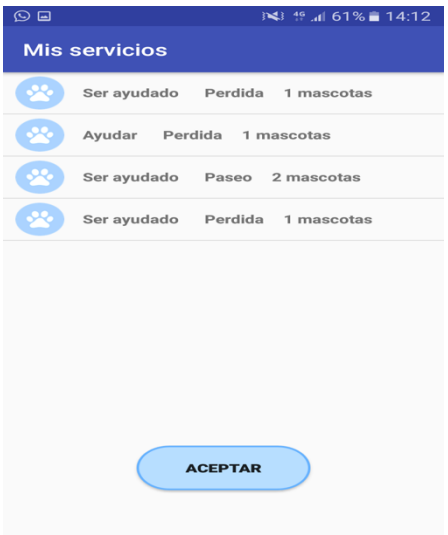


ILUSTRACIÓN 70: PANTALLA DE VISUALIZACIÓN DE MIS SERVICIOS

El botón de aceptar es únicamente para volver hacia atrás, confirmando que ya has visto lo que querías ver en esta pantalla. Para borrar un servicio creado, hacemos click largo sobre el que queramos, y nos dará la oportunidad de borrarlo.

Vemos que la información que nos muestra de cada servicio es: el tipo de servicio, además de si es ayudar o ser ayudado y el número de mascotas implicadas.

Para obtener toda la información del servicio, hacemos click sobre el servicio que queramos, y veremos toda su información. A continuación, se muestra la pantalla de un servicio en el que ofrecemos ayuda, y de uno donde la solicitamos, es decir, de un servicio de cada tipo:

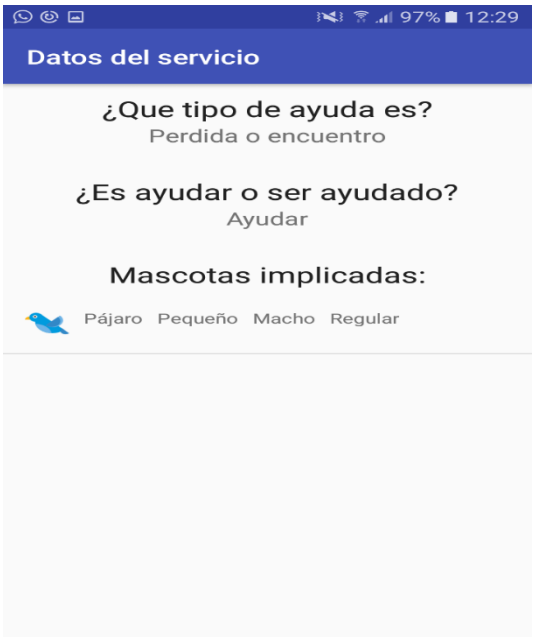


ILUSTRACIÓN 71: SERVICIO OFRECIENDO AYUDA

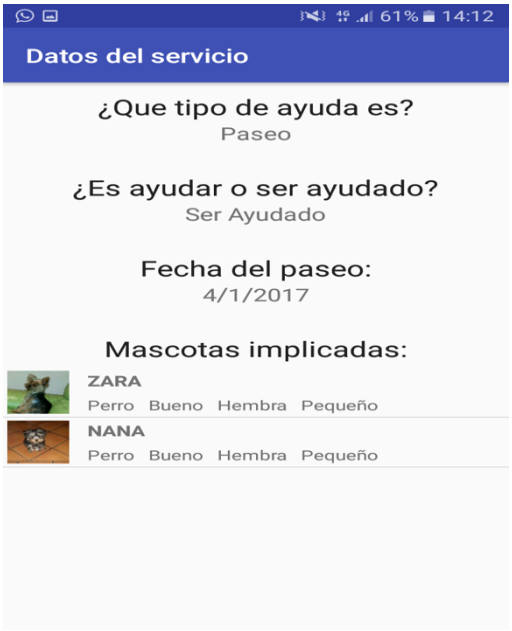


ILUSTRACIÓN 72: SERVICIO PIDIENDO AYUDA

- **Ajustes de APP:** En esta pantalla podemos modificar las ajustes de la aplicación, es decir, podemos poner si deseamos que el dispositivo vibre, suene, ambas o ninguna para cuando recibamos una notificación de otro usuario, e igualmente para mensajes de chat.

ILUSTRACIÓN 73: PANTALLA AJUSTES DE APP

Además de esos 3 apartados, en la pantalla principal de ajustes, también tenemos la oportunidad de cerrar sesión, mediante el botón rojo que está a la derecha de la imagen de perfil. Si lo pinchamos, nos mostrará un mensaje de confirmación de cierre de sesión, que si confirmamos, nos mandará a la pantalla de inicio de sesión.

Cuando estemos en la pantalla principal de la aplicación, en cualquiera de sus pestañas, tenemos en la barra de acciones de arriba (en el ActionBar) **un botón con 3 puntitos**, que nos dará 2 oportunidades:

- **Obtener ayuda:** Nos mostrará una serie de preguntas frecuentes, que podremos pinchar para ir mostrando u ocultando sus respuestas. Podemos verlo en las siguientes ilustraciones:

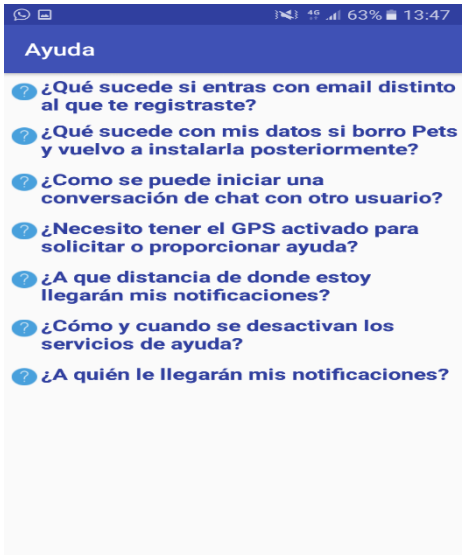


ILUSTRACIÓN 75: PREGUNTAS DE AYUDA

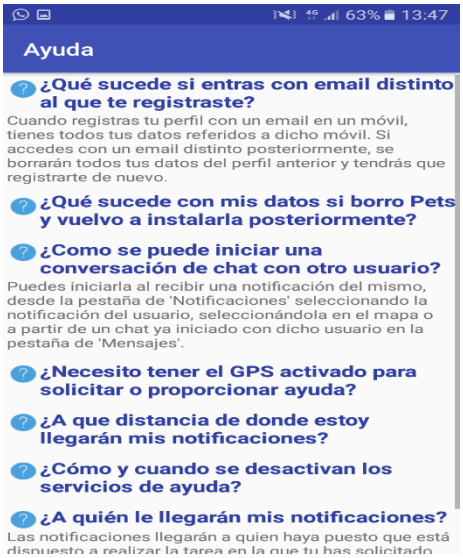


ILUSTRACIÓN 74: RESPUESTA DE AYUDA

- **Ayudar a mejorar al creador de la aplicación:** Con esta opción, se le permite al usuario rellenar unos campos, que le permitirán, mediante un gestor de correos del móvil, enviar un email al creador de la PETSMobile, en este caso, a mi.

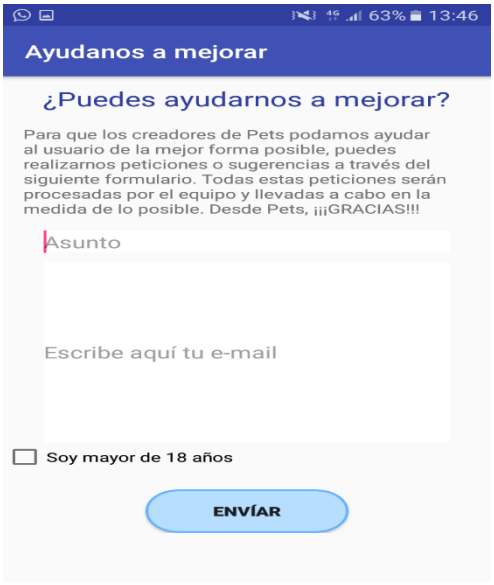


ILUSTRACIÓN 76: PANTALLA PARA AYUDARNOS A MEJORAR

Nota: El cuadro para aceptar la mayoría de edad únicamente se solicita para que en el caso de que un usuario mande algún email extraño o con faltas de respeto, poder argumentar que aceptó ser mayor de edad y escribió esas palabras voluntariamente, es decir, como medida de prevención.

5.5 POSIBLES MENSAJES DE INFORMACIÓN Y/O DE ERROR

Como ya hemos comentado anteriormente en esta documentación y en los requisitos no funcionales, PETSMobile está preparada para tratar con excepciones los posibles errores de la aplicación, y la posible información a notificar al usuario para que sepa en todo momento el por que sí o por que no pudo realizar la tarea deseada, evitando el típico mensaje molesto de Android de “La aplicación no puede responder y debe cerrarse”.

A continuación, mostraremos una serie de mensajes que el usuario podrá ver en algunos momentos de la ejecución de la aplicación. Algunos de los mensajes que el usuario podrá ver en la aplicación ya han sido mostrado en otras ilustraciones anteriormente, por lo que no se volverán a repetir en este apartado.

Si el usuario intenta registrarse en la red social sin una conexión a internet, o hay algún fallo de conexión con Nimbees, se mostrará el siguiente mensaje de error, para que el usuario se intente registrar posteriormente con una conexión a internet:

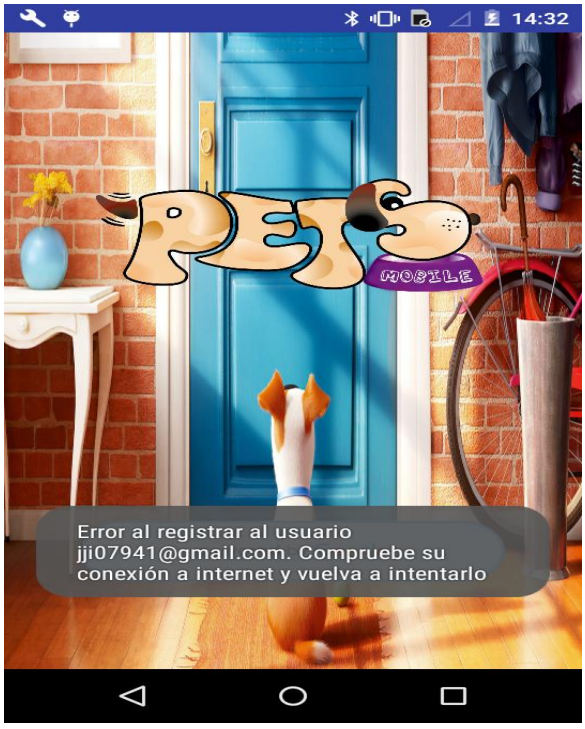


ILUSTRACIÓN 77: MENSAJE DE ERROR EN REGISTRO DE USUARIO

Si el usuario, en cualquier formulario, ya sea en el proceso de registro personal, o en el proceso de registro de mascotas, o en la creación de un servicio (a la hora de crear las mascotas que estamos dispuestos a ayudar), no rellena alguno de los campos necesarios y obligatorios, recibirá un mensaje de error como el que vemos en las siguientes ilustraciones:

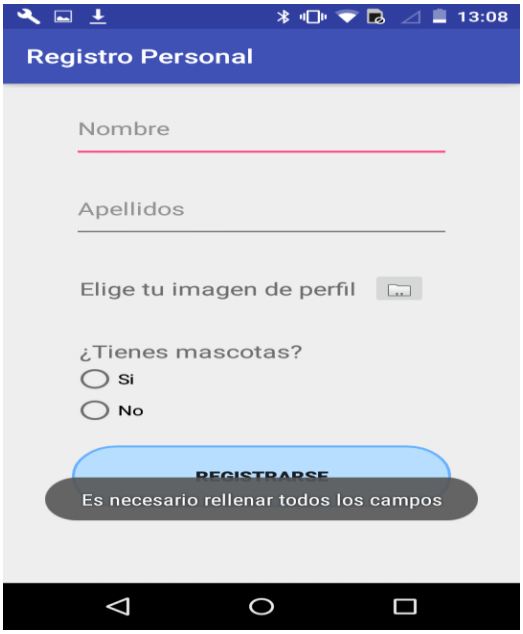


ILUSTRACIÓN 78: MENSAJE 1 DE ERROR EN FORMULARIO

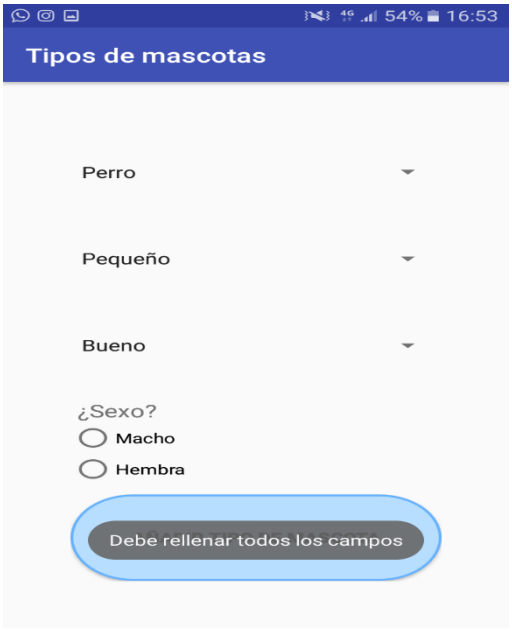


ILUSTRACIÓN 79: MENSAJE 2 DE ERROR EN FORMULARIO

También, cuando deseemos cerrar sesión o eliminar el perfil, no solicitará la confirmación del usuario:

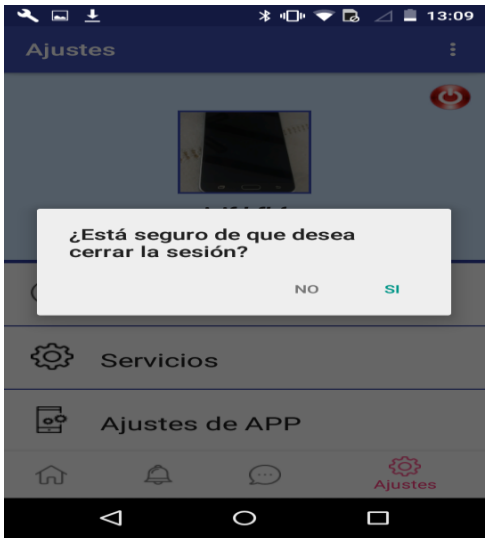


ILUSTRACIÓN 80: MENSAJE DE CONFIRMACIÓN DE CIERRE DE SESIÓN

Para la creación de servicios ya hemos comentado que es necesario tener el GPS activado, por lo que se mostrarán mensajes informando de ello, cuando sea necesario:

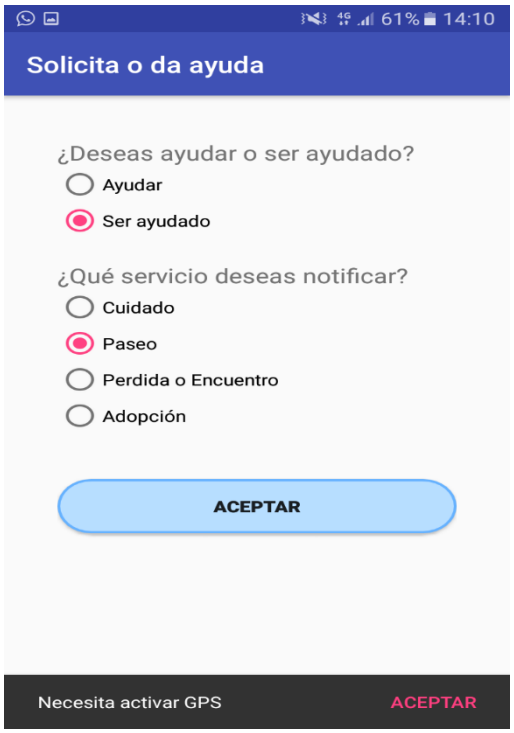


ILUSTRACIÓN 81: MENSAJE DE SOLICITUD DE GPS

Además, cuando se crea un servicio, ya sea para ofrecer ayuda o para solicitarla, se debe elegir mínimo una mascota, por lo que si no elegimos ninguna, se mostrará un mensaje informando al usuario:

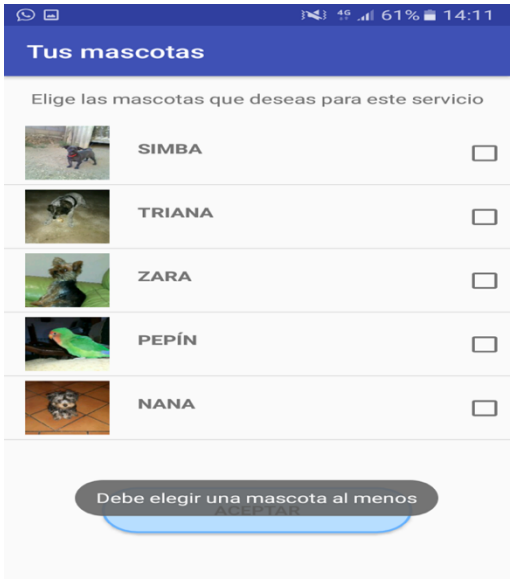


ILUSTRACIÓN 82: MENSAJE DE ERROR ELECCIÓN DE MASCOTAS

Cuando el usuario se ha registrado en PETSMobile con un email, almacenará la información de perfil para dicho email. Como ya hemos comentado, la aplicación, por motivos del servidor de Nimbees, funciona para dispositivos, por lo que no podremos tener 2 usuarios en un dispositivo. Por tanto, si un usuario ya se ha registrado en el dispositivo, se le informa cuando va a iniciar sesión en otra ocasión que si elige una cuenta distinta a la elegida anteriormente, perderá los datos almacenados anteriormente y deberá registrarse de nuevo para el nuevo email elegido. Lo vemos a continuación:

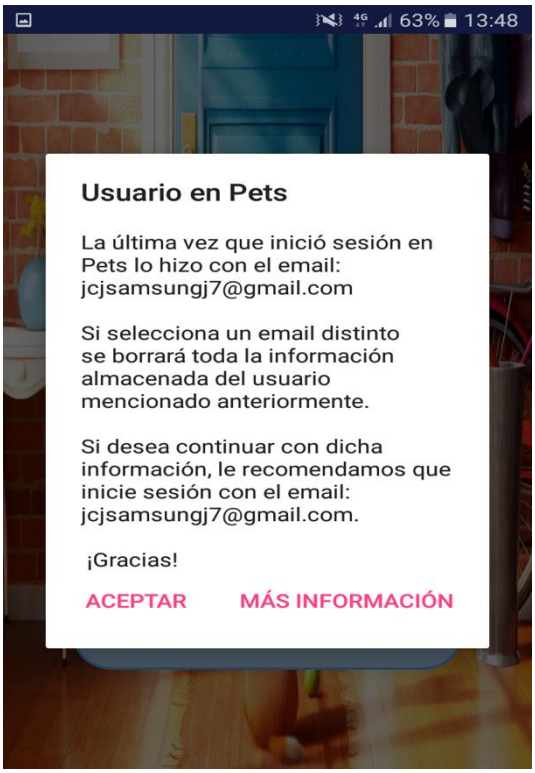


ILUSTRACIÓN 83: MENSAJE DE INFORMACIÓN EN EL INICIO DE SESIÓN

Además, la aplicación mostrará algunos otros mensajes, que no han sido recogidos en estas ilustraciones. Algunos de los ejemplos de estos mensajes son los que muestran cuando el usuario elige una fecha inválida, es decir, una fecha anterior al día de hoy, o una fecha de fin anterior a la fecha de inicio.

Mencionar también que, para los dispositivos con versiones de Android iguales o superiores a 6.0 se mostrarán los mensajes necesarios para la petición de los permisos necesarios en los momentos y apartados en que realmente son necesarios.

6. CONCLUSIONES

En este sexto capítulo vamos a exponer las conclusiones personales obtenidas, una vez se ha finalizado la realización del proyecto desarrollado para este Trabajo Fin de Grado.

Para la realización de este proyecto, se ha necesitado investigar en otras redes sociales ya existentes, así como en el servidor de Nimbees, proporcionado por el tutor y el cotutor de este Trabajo Fin de Grado. Tras realizar estas investigaciones, y acordar claramente la idea del proyecto final, se ha conseguido realizar con éxito todos y cada uno de los objetivos marcados inicialmente para la aplicación.

6.1 CONSECUCIÓN DE OBJETIVOS

Inicialmente se marcaron una serie de objetivos, que habría que tratar de conseguir con la realización de la aplicación y por tanto del proyecto. Una vez vemos realizado todo el trabajo, podemos observar que han sido cumplidos objetivos como:

- Conseguir una red social en la que los usuarios puedan ayudarse y ayudar a las mascotas.
- Conseguir que solamente lleguen las notificaciones a los usuarios que estén cercanos y que además estén realmente interesados en ayudarnos, ya que lo han puesto en su aplicación, evitando molestar a los no interesados.
- Conseguir que los usuarios puedan mantener una conversación instantánea.
- Conseguir que la aplicación funcione correctamente en un dispositivo Android.
- Conseguir que la interfaz de usuario sea, en gran parte, atractiva y agradable.

Además de los objetivos mencionados anteriormente, podemos confirmar que se han conseguido más objetivos. El único objetivo que quedaría

pendiente por conseguir es, la publicación de la aplicación, para que realmente los usuarios puedan ayudarse, y con gran importancia, puedan ayudar a las mascotas.

6.2 POSIBLES AMPLIACIONES

Para posibles ampliaciones de la aplicación, puedo contemplar las siguientes:

- Permitir que un usuario pueda ver las imágenes de perfil y de las mascotas de un usuario con el que ha tenido contacto.

- Inicio de sesión mediante una red social conocida, como podría ser Facebook.

- Muro de publicaciones de las mascotas perdidas, para que todos los usuarios puedan ver ahí, si lo desean, las mascotas perdidas, aun no habiendo recibido la notificación.

- Muro de publicaciones de las mascotas en adopción, para un usuario pueda ver si lo desea, las mascotas que el resto de usuarios ofrecen en adopción, aun no habiendo recibido las notificaciones.

- Posibilidad de creación de grupos de chat, para que varios usuarios puedan hablar mediante un grupo. Facilitaríamos, por ejemplo, a los centros de acogida que tengan la aplicación, la posibilidad de hablar entre todos los usuarios del centro que estén en el grupo la forma de comunicarse entre todos.

- Posibilidad de mandar mediante el chat la ubicación en el mapa del usuario, facilitando así las posibles quedadas entre usuarios que vayan a ayudarse.

Además, como ya se ha mencionado, la subida de la aplicación a Google Play Store, así como, la creación de la aplicación para el otro sistema operativo líder, iOS

6.3 CONCLUSIONES PERSONALES

Por lo general me siento bastante contento y satisfecho con el proyecto realizado para este Trabajo Fin de Grado, tanto de la aplicación realizada para los dispositivos Android, como de esta documentación realizada para ayudar al conocimiento de la aplicación.

Gracias a la realización de este proyecto he adquirido competencias, tanto técnicas como personales. En las competencias técnicas, mencionar el mayor aprendizaje del desarrollo para el sistema operativo Android, así como el conocimiento de librerías y la API del Servidor de Nimbees.

Con respecto a las competencias personales, me quedo con la consecución de un proyecto desde cero, tratando de conseguir un Software de calidad y con bastante robustez.

Además, he utilizado con este trabajo bastante de los conocimientos adquiridos en el Grado de Ingeniería del Software, como son la especificación de requisitos, realización de casos de uso, implementación en alguno de los lenguajes estudiados, etcétera. Todo esto se ha conseguido con bastante esfuerzo y aportando los conocimientos obtenidos en los estudios de estos 4 años.

7. BIBLIOFRAFÍA

1. Documentación de Nimbees. Consultado en múltiples ocasiones, en el año 2016.

http://doc.nimbees.com/index.php/Getting_Started

2. Diferencias en el desarrollo de iOS vs Android. Consultado en Septiembre de 2016

https://www.yeeply.com/blog/crear-apps-moviles-diferencias-android-e-ios/

3. Android. Consultado en Septiembre de 2016.

https://es.wikipedia.org/wiki/Android

4. Librería GSON, para Android. Consultado en Octubre de 2016.

https://sites.google.com/site/gson/gson-user-guide

5. ¿Qué es un ORM? Consultado en Octubre de 2016

http://www.tuprogramacion.com/glosario/que-es-un-orm/

6. ORM DBFlow para Android. Consultado en Octubre de 2016

https://guides.codepath.com/android/DBFlow-Guide

7. Comparativa de ORMs: ActiveAndroid vs DBFlow. Consultado en Octubre de 2016

https://android.libhunt.com/project/dbflow/vs/activeandroid

8. Implementación de mapas en Android. Consultado en Noviembre de 2016.

https://developers.google.com/maps/documentation/android-api/?hl=es-419

9. Librería Glide, para Android. Consultado en Noviembre de 2016.

http://expocodetech.com/glide-en-android/

10. Creación de un proyecto en Google. Consultado en Noviembre de 2016.

http://doc.nimbees.com/index.php/Android_library_usage/Integration_guide/Register_for_Google_Cloud_Messaging

11. Firebase en Android. Consultado en Noviembre de 2016.

https://firebase.google.com/docs/cloud-messaging/

12. Firebase, plataforma de desarrollo. Consultado en Noviembre de 2016.

http://www.elandroidelibre.com/2016/05/firebase-plataforma-desarrollo-android-ios-web.html

13. Abandono de mascotas en 2015 en España. Consultado en Diciembre de 2016.

http://www.eldiario.es/sociedad/Espana-abandonan-animales-compania-ano_0_533696869.html

14. Infografía de abandono y adopción de mascotas en 2015. Consultado en Diciembre de 2016.

http://www.fundacion-affinity.org/observatorio/infografia-estudio-de-abandono-y-adopcion-2015