



# Análisis, diseño e implementación de una aplicación para el reaprovechamiento de utensilios y muebles

**Carlos Solans**

Máster universitario de desarrollo de aplicaciones para dispositivos móviles

**Pau Dominkovics Coll**

06/06/2018



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivad a [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

**Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)**

**A) Creative Commons:**



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada a [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-Compartirlgual [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento-Compartirlgual [3.0 España de Creative Commons](#)



Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](#)

**B) GNU Free Documentation License (GNU FDL)**

Copyright © 2018 CARLOS SOLANS OLIVEROS

Permission is granted to copy, distribute and/or modify this document under the terms of the

GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

### **C) Copyright**

© (el autor/a)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.



## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	Análisis, diseño e implementación de una aplicación para el reaprovechamiento de utensilios y muebles
<b>Nombre del autor:</b>	Carlos Solans
<b>Nombre del consultor:</b>	Pau Dominkovics Coll
<b>Fecha de entrega (mm/aaaa):</b>	06/2018
<b>Titulación:</b>	<i>Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles</i>
<b>Resumen del Trabajo (máximo 250 palabras):</b>	
<p>Análisis, diseño e implementación de una aplicación para facilitar el reaprovechamiento de utensilios y muebles. Esta aplicación permite compartir la ubicación, foto y una leve descripción de cualquier objeto que se encuentre en la calle para que otro usuario pueda recuperarlo de la misma. De esta manera se potencia el reaprovechamiento de objetos y, complementariamente, se ayuda a los servicios de recogida del ayuntamiento a localizar los objetos depositados en la calle.</p>	

**Abstract (in English, 250 words or less):**

Analysis, design and implementation of an application to facilitate the reuse of utensils and furniture. This application allows you to share the location, photo and a slight description of any object that is on the street so that another user can retrieve it from it. In this way, the reuse of objects is enhanced and, in addition, the collection services of the town hall are helped to locate the objects deposited on the street.

**Palabras clave (entre 4 y 8):**

Android, reaprovechamiento, reciclaje, DIY





# Índice

<b>1. Introducción</b>	<b>12</b>
1.1 Contexto y justificación del trabajo	12
1.2 Objetivos del trabajo	13
1.3 Enfoque y método seguido	14
1.4 Planificación del trabajo	14
1.5 Breve sumario de productos obtenidos	16
<b>2. Diseño</b>	<b>17</b>
2.1 Diseño centrado en el usuario	17
2.1.1 Usuarios y contexto de uso	17
2.1.2 Fichas de personas	19
2.1.3 Diseño conceptual con escenarios	21
2.1.4 Prototipado horizontal	23
2.1.5 Evaluación	27
<b>3. Definición de los casos de uso (actores, precondiciones, flujo y postcondiciones)</b>	<b>28</b>
<b>4. Diseño de arquitectura de software</b>	<b>30</b>
4.1 UML base de datos	30
4.2 UML de las entidades y las clases	31
4.3 Diagrama explicativo de la arquitectura del sistema	33
<b>5. Implementación</b>	<b>34</b>
5.1 Estructura del proyecto de Android Studio	34
5.1.1 Código fuente y archivos de recursos asociados	35
5.1.1.1 NavigationDrawerActivity.java	35
5.1.1.2. AddItemFragment.java y fragment_add_item.xml	38
5.1.1.3 DetailItemFragment.java y fragment_detail_item.xml	39
5.1.1.4 ElementsListFragment.java y fragment_elements_list.xml	40
5.1.1.5 DetailUserFragment.java y fragment_detail_user.xml	41
5.1.1.6 ElementsMapFragment.java y fragment_elements_map.xml	42
5.1.1.7 FilterFragment.java y fragment_filter.xml	43
5.1.1.8 LoginFragment.java y fragment_login.xml	44
5.1.1.10 ItemContent.java	45
5.1.1.11 UserContent.java	45
5.1.1.12 CurrentLocationManager.java	46
5.1.1.13 FirebaseManager.java	46
5.1.2 AndroidManifest.xml	46
5.1.3 build.gradle	47
5.1.4 Servidor Firebase	47
<b>6. Líneas futuras de desarrollo</b>	<b>50</b>

<b>7. Conclusiones</b>	<b>51</b>
<b>8. Glosario</b>	<b>52</b>
<b>9. Bibliografía</b>	<b>53</b>
<b>10. Anexo 1: Manual de usuario</b>	<b>55</b>
10.1 Requerimientos	55
10.2 Descarga de la aplicación	55
10.3 Navegación de la Aplicación	55
10.3.1 Pantalla de inicio	55
10.3.2 Pantalla principal	55
10.3.3 Registro y configuración	55
10.3.4 Búsqueda	56
10.3.5 Subida de producto	56

## Lista de figuras

Ilustración 01 - Planificación del trabajo: Plan de trabajo	15
Ilustración 02 - Planificación del trabajo: Diseño	15
Ilustración 03 - Planificación del trabajo: Implementación	16
Ilustración 04 - Prototipado horizontal	23
Ilustración 05 - Menú lateral	24
Ilustración 06 - Botón “+”	24
Ilustración 07 - Boton resultados búsqueda	25
Ilustración 08 - Prototipo baja resolución	26
Ilustración 09 - UML base datos	30
Ilustración 10 - UML entidades I	32
Ilustración 11 - UML entidades II	32
Ilustración 12 - MVC	33
Ilustración 13 - activity_navigation_drawer.xml	35
Ilustración 14 - NavigationDrawer	37
Ilustración 15 - ActionBar I	37
Ilustración 16 - ActionBar II	37
Ilustración 17 - fragment_add_item.xml	39
Ilustración 18 - fragment_detail_item.xml	40
Ilustración 19 - fragment_elements_list.xml	41
Ilustración 20 - fragment_cardview_elements_list.xml	41
Ilustración 21 - fragment_detail_user.xml	42
Ilustración 22 - fragment_elements_map.xml	43
Ilustración 23 - fragment_filter.xml	44
Ilustración 24 - fragment_login.xml	45
Ilustración 25 - Firebase item	48
Ilustración 26 - Firebase Storage	49

# 1. Introducción

## 1.1 Contexto y justificación del trabajo

Este trabajo de final de máster se centra en el desarrollo de una aplicación para Android para la geolocalización de muebles desechados en la vía pública, tanto para particulares como para instituciones como ayuntamientos.

Dicha aplicación pretende cubrir varias necesidades relacionadas con la reutilización de objetos:

- Por un lado, la necesidad que tienen algunas personas de conseguir piezas de mobiliario u otros utensilios domésticos a un bajo precio.
- Por otro lado, reducir las consecuencias medioambientales de la cultura de consumo en la que vivimos, fomentando el reaprovechamiento y reciclaje de muebles u otros utensilios domésticos.
- Por otra parte, facilitar la recogida de residuos a los servicios de limpieza de los ayuntamientos.

El reaprovechamiento, tanto de muebles como de utensilios domésticos, se ha vuelto un tema relevante debido a:

- Por la mala situación económica de un sector amplio de la población, que se vé obligada en gran medida a optar por el reaprovechamiento y reciclaje de muebles, o su obtención por un bajo precio.
- Por los altos índices de contaminación del planeta, en parte debido a la cultura del “usar y tirar”.
- Por la cultura creciente del DIY y/o reciclaje impulsado por la gran cantidad de información para realizarlo disponible en la red. También se le tiene que añadir a este punto el plus que puede suponer tener un mueble o utensilio reciclado de manera “única” y distinto al de la gran mayoría (véanse globalización en muebles de IKEA).

Actualmente estos problemas se resuelven de la siguiente manera:

- Por un lado está el problema de acceder a muebles de bajo coste por parte de un sector de la población: Ahora mismo este problema se resuelve mediante plataformas de compra venta entre particulares de muebles o utensilios de segunda mano, como por ejemplo Wallapop y, en menor medida, mediante la búsqueda activa de mobiliario los días de recogida del mismo por parte del ayuntamiento o en los diversos rastros de nuestra geografía.
- Por otro lado está el problema de los altos índices de contaminación, ahora mismo problema sin resolver pendiente.
- Existe también el problema que se encuentran las personas que siguen la corriente del DIY y del reciclaje: estas tienen que recorrer a

aplicaciones de compra venta entre particulares, lo que normalmente les implica un desembolso mayor.

Por lo tanto, como resultado se quiere obtener:

- Una comunidad de personas que faciliten encontrar objetos desechados útiles para otros de una manera prácticamente desinteresada.
- Facilitar la recogida de objetos desechados en el espacio público a los servicios de recogida del ayuntamiento.

## 1.2 Objetivos del trabajo

El objetivo principal del proyecto es realizar una aplicación de Android para la reutilización de muebles y/o utensilios domésticos. Por lo tanto, podríamos desglosar los requisitos funcionales y no funcionales de la siguiente manera:

- Requisitos funcionales:
  - El futuro usuario tiene que poder registrarse en la aplicación mediante email y contraseña, cuenta de Facebook o de Google.
  - El usuario podrá configurar su perfil con su nombre o nick, avatar y zona geográfica.
  - El usuario podrá subir un objeto realizándole una foto, y añadiéndole un título, descripción, clasificación de tipo de objeto y ubicación.
  - El usuario podrá visualizar los objetos que aparecen sobre un mapa y al clickar encima accederá a una descripción más detallada.
  - El usuario podrá visualizar los objetos que aparecen ordenados en una lista según distancia, número de visitas o tiempo.
  - El usuario podrá abrir un chat con la persona que haya publicado un objeto con tal de conocer más detalles.
  - La aplicación descargará y guardará toda la información en un servidor externo.
- Requisitos no funcionales:
  - Disponibilidad: El servidor que nutre a la aplicación debe estar siempre disponible.
  - Rendimiento: La aplicación debe funcionar de manera rápida, optimizando tanto las imágenes de los objetos que se incluyen, como el resto de objetos.
  - Usabilidad: Debe ser fácil de usar, mediante una interfaz intuitiva.
  - Apariencia: La aplicación debe tener una apariencia moderna, con un logo y diseño atractivos.
  - Seguridad: La aplicación y el servidor deben integrar los protocolos de seguridad estándar.

- Estabilidad: La aplicación debe mostrarse estable en todo momento y en el caso de bloquearse debería reiniciarse inmediatamente.

## 1.3 Enfoque y método seguido

El método para realizar el trabajo viene marcado en gran parte por las entregas de las correspondientes PECs:

- PEC1: Plan de Trabajo
- PEC2: Diseño
- PEC3: Implementación
- PEC4: Entrega

Esta división propuesta enfatiza aspectos como el análisis y justificación de las decisiones tomadas, lo cual facilita la realización del proyecto de una manera estructurada y razonada.

La tecnología escogida para el desarrollo de la app es Android nativo. Por un lado, se ha decidido hacerla para Android por motivos puramente estadísticos: es la plataforma más utilizada en la actualidad. El hecho de realizarla nativa en Android es por un tema práctico y por una cierta visión de la escalabilidad de la aplicación: el primer objetivo es que la aplicación funcione en Android, plataforma principal hoy en día. A posteriori, si es que la aplicación funcionase, se podría realizar para iOS también, compartiendo, eso sí, el mismo back-end.

Como plataforma para el back-end se ha escogido Firebase. Me he decantado por esta plataforma por varios motivos:

- Compatibilidad con Android, iOS y web.
- Conocimiento de la misma: se ha utilizado en algunas de las prácticas del Máster.
- Escalabilidad: En su versión básica es gratuita pero si la app creciera en usuarios y, por lo tanto, en volumen de tráfico, tiene la capacidad de dar soporte a las nuevas necesidades.
- Informes de fallos.
- Estadísticas de uso.
- Mensajería instantánea in-app.
- Implementación progresiva: permite lanzar las funcionalidades nuevas a un subgrupo de usuarios para recibir su feedback.

## 1.4 Planificación del trabajo

El total de horas que se van a emplear al día es de aproximadamente 2 de lunes a viernes, intentando en los días festivos no tener que dedicarle ninguna hora. De esta manera, en caso de sufrir algún retraso siempre se puede recuperar durante alguno de los festivos. Esperemos que no haga falta...

Los recursos necesarios para realizar el proyecto son:

- PC
- Android Studio
- Smartphone
- Tiempo

Tareas a realizar:

- Plan de trabajo (Entrega 14/03):
  - Contexto y justificación del trabajo (4 horas)
  - Objetivos del trabajo (6 horas)
  - Enfoque y método seguido (2 horas)
  - Planificación del trabajo (6 horas)
  - Sumario productos obtenidos (2 horas)

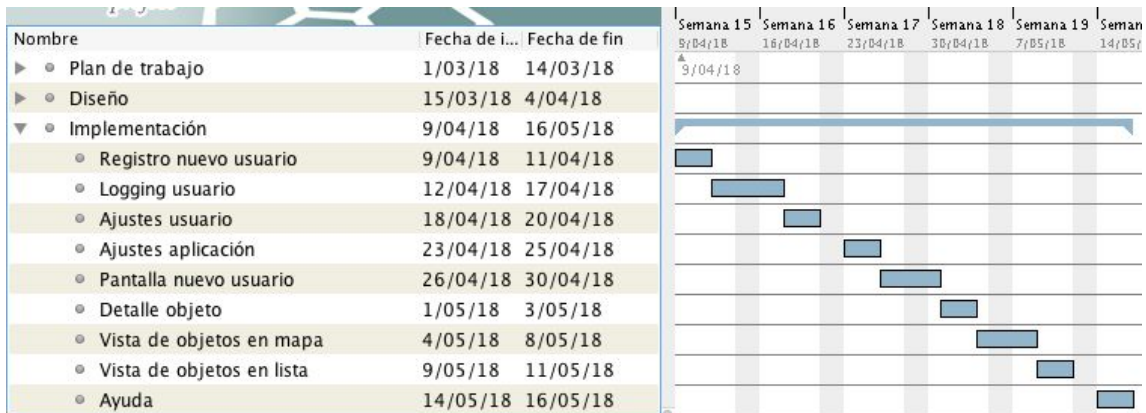


- Diseño (Entrega 04/04):
  - Diseño centrado en el usuario (Etapa iterativa) (14 horas)
    - Usuarios y contexto de uso
    - Diseño conceptual con escenarios
    - Prototipado horizontal
    - Evaluación
  - Definición de los casos de uso (actores, precondiciones, flujo y postcondiciones) (8 horas)
  - Diseño de arquitectura de software (8 horas)



- Implementación (Entrega 16/05):
  - Registro nuevo usuario (6 horas)
  - Logging usuario (6 horas)
  - Ajustes usuario (6 horas)

- Ajustes aplicación (6 horas)
- Pantalla nuevo usuario (6 horas)
- Detalle objeto (6 horas)
- Vista de objetos en mapa (6 horas)
- Vista de objetos en lista (6 horas)
- Ayuda (6 horas)



- Entrega Final (Entrega 06/06)
  - Elaboración final de la memoria (30 horas)
  - Elaboración de la presentación virtual (10 horas)

## 1.5 Breve resumen de productos obtenidos

Como resultado del trabajo obtendremos tres elementos:

- Proyecto de la aplicación de Android y la propia aplicación: Debido a que se ha elegido Android como plataforma para el desarrollo, la aplicación generada así como el código generado serán sobre Android Studio.
- Memoria describiendo los objetivos del proyecto y el proceso de desarrollo.
- Presentación que resume los puntos más importantes.



## **2. Diseño**

### **2.1 Diseño centrado en el usuario**

Para diseñar la aplicación se va a utilizar el proceso de diseño centrado en el usuario. Esta metodología sitúa a la persona en el núcleo del proceso del diseño, atendiendo a sus procesos cognitivos, sus necesidades y los posibles escenarios de uso. De esta manera se puede lograr un diseño útil para los usuarios potenciales, y, por lo tanto, se consigue una satisfacción en el usuario mejorando su experiencia con un esfuerzo relativamente pequeño. Normalmente es un proceso que consta de cuatro etapas, de las cuales las tres últimas suelen repetirse a lo largo del desarrollo de la aplicación, evitando perder de vista las necesidades reales de los usuarios.

Estas cuatro etapas suelen ser:

- Análisis de usuarios y el contexto de uso
- Diseño de producto
- Prototipado
- Evaluación del diseño con usuarios

#### **2.1.1 Usuarios y contexto de uso**

En esta primera fase se van a intentar identificar los diversos tipos de usuarios, sus necesidades y objetivos, así como los diversos contextos de uso donde la aplicación se puede utilizar.

Según las necesidades se pueden diferenciar tres grupos de usuarios potenciales:

- Personas que se encuentran, o que se quieren deshacer, de algún tipo de mueble o utensilio.
- Personas que buscan un mueble o utensilio, con la intención que sea, ya sea para quedárselo o restaurarlo para venderlo o regalarlo.
- El servicio de recogida de muebles o utensilios de la ciudad.

En referencia al tipo de público al que va dirigido se podría decir que hay tres grandes grupos:

- Por un lado, un gran grupo de edad variable y de intereses dispares, que serían las personas que se encuentran un mueble, las que se quieren deshacer de uno, o las que lo buscan. Este tipo de usuario puede tener un nivel de conocimiento informático desde básico hasta alto.

- Un segundo grupo de usuarios más especializado en la restauración y comercialización de muebles, con un rango de edad también amplio, que se pueden beneficiar de varias maneras de la aplicación:
  - Por un lado pueden obtener muebles para restaurar de una manera económica, restaurarlos, venderlos y así sacar un beneficio.
  - También pueden ofrecer sus servicios como especialistas, para la restauración de un mueble que un usuario haya adquirido a través de la aplicación.
  
- Un tercer grupo que sería el organismo de la administración pública que se encargaría de gestionar la recogida de los desechos. Este grupo estaría formado por especialistas del ayuntamiento en cuestión que, a través de la aplicación, se beneficiarían de varios aspectos:
  - Les ayudaría en la difusión de las pautas y horarios para el depósito de los muebles o utensilios en la vía pública a los habitantes de la población.
  - Un menor número de muebles a recoger y transportar al vertedero, lo que supondría un ahorro al ayuntamiento. Ayudaría a fomentar el reciclaje y la reutilización, por lo que a nivel ambiental también sería beneficioso.
  - Por otro lado, quedaría un registro de la ubicación y cantidad de muebles o utensilios a recoger por parte del ayuntamiento, pudiendo prever, en cierta manera, el número de efectivos por zona.

Por lo tanto, las funciones clave de la aplicación con tal de satisfacer a sus usuarios serán:

- Búsqueda de mueble o utensilio por clasificación de tipo y proximidad.
- Agregación de mueble o utensilio por clasificación de tipo y ubicación.

Luego habría un grupo de funciones secundarias que la aplicación debería cubrir:

- Login mediante cuenta de email o cuenta de Google.
- Personalización de perfil (nick, imagen...)
- Chat entre usuarios.

## 2.1.2 Fichas de personas

### Persona tipo 1

<b>Nombre</b>	Juan
<b>Biografía</b>	Juan es un trabajador social del barrio de Sants. Hace poco que le salió un trabajo por esta zona y decidió mudarse. Vive en pareja con Rogelia desde hace 5 años. Es una persona un tanto olvidadiza y caótica aunque muy cumplidora.
<b>Cita</b>	“Cambiar al piso está resultando una labor titánica. Después de haber dado las mensualidades correspondientes para poder entrar en él no nos queda apenas dinero para amueblarlo. Necesitamos conseguir muebles baratos para, aunque sea, salir del paso.”
<b>Objetivos</b>	Conseguir muebles para el piso al que se ha mudado recientemente.
<b>Comportamientos</b>	Le interesan las nuevas tecnologías y las utiliza diariamente e intensamente. Tiene PC y smartphone con conexión a internet.
<b>Necesidades</b>	Que la aplicación sea directa y fiable.

### Persona tipo 2

<b>Nombre</b>	Paco
<b>Biografía</b>	Paco es un trabajador del sector textil que vive en la zona de Sants desde hace ya 30 años junto a su mujer. Tienen un hijo y una hija, ambos estudiantes de la ESO. Es una persona un tanto seria y pensativa. Le gusta salir de pesca los fines de semana.

<b>Cita</b>	“Siempre que veo algún mueble por la calle tirado pienso en las utilidades que podría tener o si le podría ir bien a algún conocido”
<b>Objetivos</b>	Ayudar desinteresadamente a gente que busque muebles
<b>Comportamientos</b>	Siente curiosidad por las nuevas tecnologías pero realiza un uso tecnológico bastante limitado. Se atreve con el smartphone pero apenas para enviar algún que otro Whatsapp.
<b>Necesidades</b>	Que la aplicación sea muy fácil de utilizar.

### Persona tipo 3

<b>Nombre</b>	Maria
<b>Biografía</b>	Maria es una trabajadora del ayuntamiento del Masnou desde hace ya 9 años. Vive en pareja en la localidad y está encantada de vivir ahí. Es una persona muy sana y se considera una apasionada del deporte.
<b>Cita</b>	“La recogida de desechos supone a los servicios de recogida un gran esfuerzo físico. Cualquier ayuda es más que bienvenida”
<b>Objetivos</b>	Lograr recoger todos los desechos de la zona que tiene asignada dentro de su turno de trabajo
<b>Comportamientos</b>	Le interesan las nuevas tecnologías pero no realiza un uso avanzado de las mismas. Suele utilizar el Whatsapp y Facebook a casi todas horas. Le gusta jugar a Candy Crush.
<b>Necesidades</b>	Que la aplicación sea directa y fiable.

### **2.1.3 Diseño conceptual con escenarios**

En este apartado he inventado tres escenarios donde se podría utilizar la aplicación:

#### Escenario de uso 1: Búsqueda de una silla

Juan y Rogelia acaban de alquilar un piso sin amueblar en el barrio de Sants y necesitan conseguir muebles. Debido a los elevados gastos que conlleva entrar de alquiler a un piso hoy en día, les queda muy poco dinero para poder gastar en muebles y electrodomésticos. Este hecho, unido a la incertidumbre de si les va a gustar vivir ahí, hace que su prioridad ahora mismo no sea gastarse mucho dinero en muebles. Ambos deciden mirar en plataformas de segunda mano y aplicaciones de reutilización de muebles para ver si encuentran algo que les guste a un precio razonable. Se descargan la app y en unos instantes se registran con su cuenta de Google. Acceden al buscador de muebles y utensilios y en seguida ven varios muebles cerca de su ubicación. Deciden aplicar un filtro por tipo de mueble silla y en ese momento no hay nada que les interese pero igualmente deciden guardar la búsqueda para que les llegue una notificación cuando aparezca algo.

Al cabo de un rato les llega una notificación al móvil de una nueva silla en su zona. Juan y Rogelia se desplazan hasta la ubicación de la silla en cuestión y tienen suerte y la encuentran. Reportan en la aplicación que la han podido recoger y Paco, el creador de la alerta, gana puntos de la aplicación.

#### Escenario de uso 2: Subida desinteresada de ítem

Paco es un trabajador del textil que cada noche aprovecha para fumarse un pitillo mientras baja la basura. Se dá cuenta que hay una silla que no está mal pero que tampoco le hace falta ahora mismo por lo que decide sacar el móvil, abrir la aplicación, hacerle una foto y subirla a la aplicación clasificandola como silla. Paco está buscando una lámpara de pie, por lo que cree que vale la pena subir la foto a la aplicación para que otro usuario se beneficie de ella. Es una silla de una conocida marca de muebles que no se encuentra en mal estado pero que tiene un par de rascadas en el lateral. Debido a su bajo precio ya nueva, alguien decidió tirarla presuponiendo que no valía la pena subirla a alguna plataforma de segunda mano y perder el tiempo negociando por tan poco dinero.

### Escenario de uso 3: Recogida de muebles por el servicio de recogida municipal:

Son las 22:00h y Maria se dispone a empezar su turno de trabajo en la empresa que realiza la recogida de desechos en el ayuntamiento del Masnou. Antes de arrancar, Maria abre la app y dedica un par de minutos a mirar en la aplicación si alguien ha reportado algún mueble o utensilio en la zona de recogida que le toca cubrir hoy. En la aplicación aparecen un par de zonas con grandes montones de desechos, por lo que decide comentarle a su responsable que esa noche necesitaría apoyo en la zona y su responsable decide asignarle un otro compañero de apoyo para cubrir esa zona.

## 2.1.4 Prototipado horizontal

Para la realización del prototipo horizontal he tenido en cuenta los diversos flujos planteados en el apartado anterior. He optado por potenciar uno de los puntos que caracterizan a la aplicación: la proximidad. Para ello se parte de una primera pantalla donde se muestra un mapa con los objetos que hay cerca de nuestra ubicación: en un inicio no hará falta registrarse, ni configurar nada para poder acceder al contenido. De esta manera conseguimos mostrar algo interesante para el usuario desde el segundo cero y, además, acelerar la velocidad de uso de la aplicación.

Una vez dentro de la pantalla de “Resultados búsqueda”, se podrá acceder a cada una de las demás funciones, ya sea mediante el menú lateral de navegación o mediante los iconos situados en la parte superior e inferior de la pantalla.

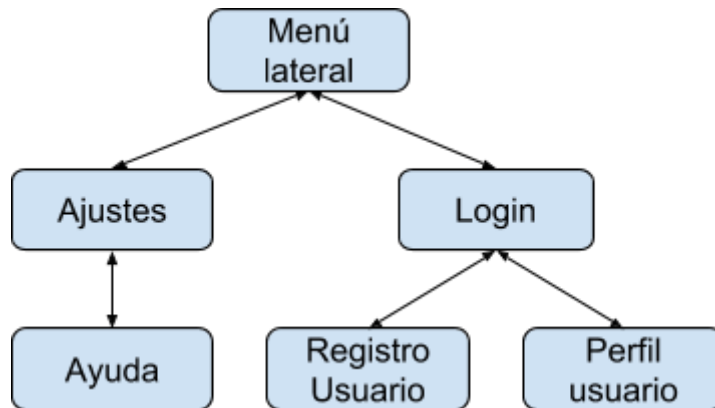
Estructura general de navegación de la aplicación:



## Elementos de navegación:

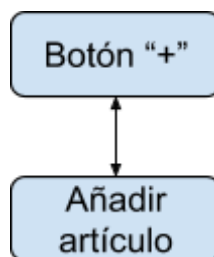
### Menú lateral:

La aplicación dispondrá de un menú lateral de navegación accesible desde todas las pantallas principales de la misma. Para desplegarlo habrá que pulsar en el icono de menú ubicado en la esquina superior izquierda. Desde este menú podremos acceder a la pantalla de “Ajustes” y a la pantalla de “Login”, desde donde podremos registrarnos con nuestro usuario o configurar nuestro perfil según el diagrama siguiente:



### Botón “+”:

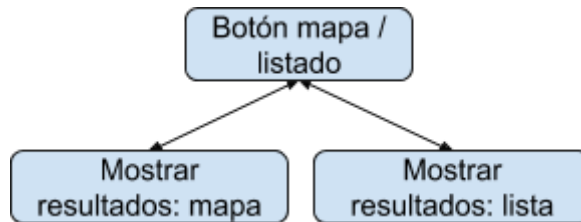
Se ha decidido incorporar un botón “+” flotante en la parte inferior derecha de la pantalla “Resultados búsqueda”. Este botón servirá para añadir un nuevo mueble o utensilio a la aplicación. Este tipo de botón, muy común en las aplicaciones actuales, resulta muy útil cuando se quiere hacer especialmente accesible una función concreta de la aplicación. En esta aplicación, donde los elementos son altamente volátiles y pueden durar solo unos pocos minutos activos, cualquier énfasis en la agregación de nuevos es poca y eso es lo que se busca ubicándolo ahí.





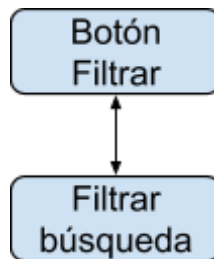
Botón “mostrar resultados en mapa” / “mostrar resultados en lista”:

Se ha decidido incorporar en la barra superior un botón para cambiar la manera en que se verá la pantalla de “Resultados búsqueda”, proporcionando dos opciones: vista de resultados en modo mapa o en modo lista. Mediante este botón podremos cambiar entre ambas de una manera ágil y rápida. Dicho botón cambia el icono que muestra en función de la pantalla que se está mostrando.

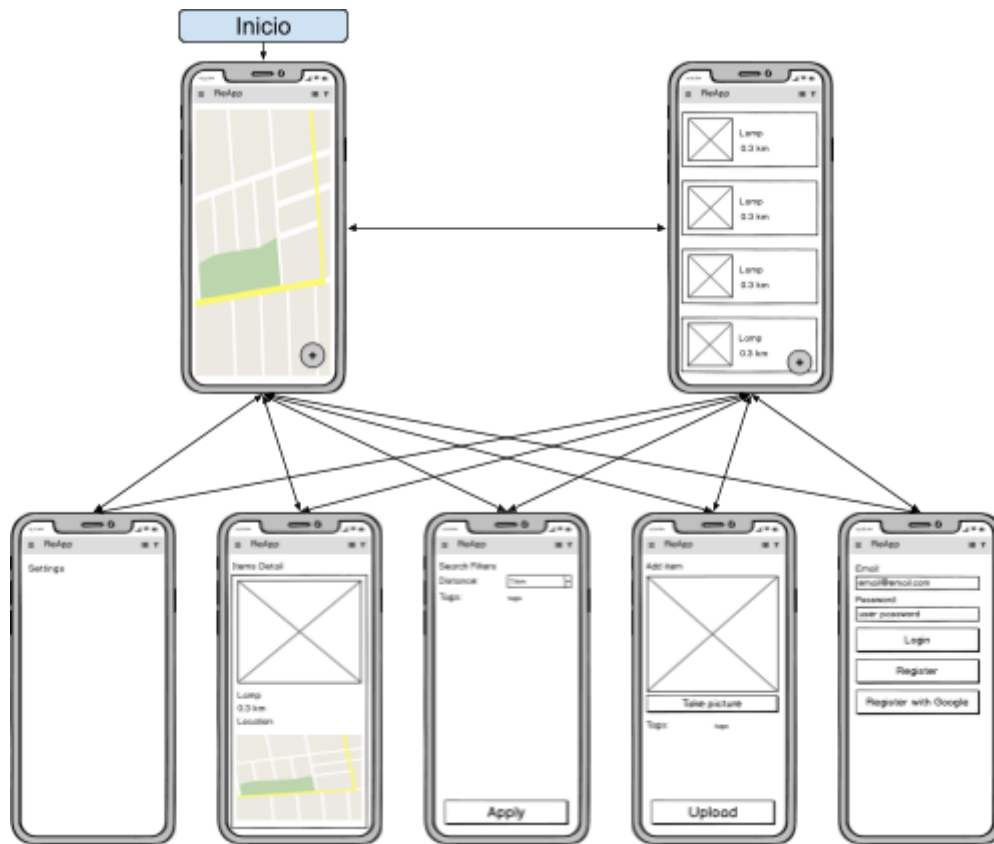


Botón “Filtrar búsqueda”:

En la zona superior derecha se ha decidido incorporar un botón para poder acceder a la pantalla de filtraje de ítems.



A continuación se muestra un prototipo de baja fidelidad de los conceptos anteriormente explicados. Esta etapa es iterativa, por lo que se espera ir desarrollando prototipos más definidos a medida que se realicen más iteraciones.



## 2.1.5 Evaluación

Para la realización de la evaluación se va a utilizar el método del paseo cognitivo. Se ha escogido este método debido a que tiene un bajo coste y permite la evaluación del diseño desde los primeros esbozos. Para ello, basándome en los escenarios expuestos anteriormente, voy a describir unas tareas representativas de la aplicación:

- Realizar una búsqueda por tipo de mueble silla a una distancia menor de 1 km.
- Crear un nuevo item realizándole una foto y añadiendo tipo de mueble o utensilio y una breve descripción.
- Registro usuario.

Estas tres tareas son, a grosso modo, el núcleo de la aplicación por lo que realizando un test sobre ellas tendremos cubierto gran parte del testeo de la app.

La evaluación se empezará a realizar una vez se disponga de una primera versión del software. Bien es cierto que según la planificación, este hecho no se dará hasta prácticamente la finalización de la implementación, pero en este caso, la planificación se debe tomar como algo orientativo, ya que se espera tener un prototipo mucho antes. Por ese motivo, se prefiere plantear las evaluaciones como algo a hacer periódicamente, empezando al tener una funcionalidad de las testeables desarrollada y repitiéndose al ir incorporando variaciones.

### 3. Definición de los casos de uso (actores, precondiciones, flujo y postcondiciones)

Caso de uso	Buscar mueble
Actores	Usuario
Precondiciones	Tener la aplicación instalada
Flujo	<p>1. Iniciar la aplicación.</p> <p>La aplicación por defecto muestra todos los ítems situados en las proximidades (realiza una búsqueda estándar). Desde este instante el usuario ya puede buscar un mueble.</p> <p>Si por otro lado prefiere afinar la búsqueda puede acceder al menú <i>Filter</i> a través del acceso ubicado en la parte superior derecha. Desde ahí puede especificar el tipo de objeto que busca y la distancia máxima. Pulsa sobre el botón <i>Apply</i> y la aplicación retorna a la pantalla anterior pero esta vez mostrando los resultados filtrados.</p>
Postcondiciones	Se visualizan en pantalla los muebles de la búsqueda

Caso de uso	Agregar mueble, etiquetarlo y añadirle foto y descripción
Actores	Usuario
Precondiciones	- Tener la aplicación instalada
Flujo	<p>1. Pulsar “+”.</p> <p>2. Pulsar sobre el símbolo “+” y hacer una foto.</p> <p>3. Pulsar sobre el campo de texto de tags y etiquetar el objeto en cuestión.</p>
Postcondiciones	Se ha añadido un nuevo ítem a la aplicación

Caso de uso	Registrarse con email y contraseña
Actores	Usuario
Precondiciones	- Tener la aplicación instalada
Flujo	<ol style="list-style-type: none"> <li>1. Pulsar en el icono menú para desplegar el menú lateral.</li> <li>2. Pulsar sobre la cabecera del menú lateral para que se abra la pantalla de login / registro.</li> <li>3. Escribir una dirección de email y password y pulsar sobre el botón de registro.</li> </ol>
Postcondiciones	Se ha registrado un nuevo usuario en el sistema

Caso de uso	Login con email y contraseña
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> <li>- Tener la aplicación instalada</li> <li>- Estar registrado en el sistema</li> </ul>
Flujo	<ol style="list-style-type: none"> <li>1. Pulsar en el icono menú para desplegar el menú lateral.</li> <li>2. Pulsar sobre la cabecera del menú lateral para que se abra la pantalla de login / registro.</li> <li>3. Escribir la dirección de email y password previamente utilizadas durante el registro y pulsar sobre el botón de login.</li> </ol>
Postcondiciones	Se ha logueado un usuario en el sistema

Caso de uso	Registrarse con una cuenta de Google
Actores	Usuario
Precondiciones	- Tener la aplicación instalada
Flujo	<ol style="list-style-type: none"> <li>1. Pulsar en el icono ubicado arriba a la izquierda menú para desplegar el menú lateral.</li> <li>2. Pulsar sobre la parte superior del menú lateral.</li> <li>3. Pulsar sobre el botón de registro con Google.</li> <li>4. Escoger la cuenta de Google que queremos utilizar.</li> </ol>
Postcondiciones	Se ha registrado un nuevo usuario en el sistema

## 4. Diseño de arquitectura de software

### 4.1 UML base de datos

La base de datos contendrá los datos de cada uno de los usuarios y de cada uno de los ítems, por lo que a nivel de estructura, deberá contener estas dos tablas de datos que se muestran a continuación.

Item	User
<ul style="list-style-type: none"><li>- Description</li><li>- Latitude</li><li>- Longitude</li><li>- UserId</li><li>- ImageURL</li><li>- Id</li><li>- DateTime</li></ul>	<ul style="list-style-type: none"><li>- Name</li><li>- Id</li><li>- Latitude</li><li>- Longitude</li><li>- ImageURL</li><li>- Email</li><li>- Phone</li><li>- Birthday</li></ul>

Como se comentó anteriormente, se ha elegido Firebase como plataforma para el back-end. Esta plataforma permite compatibilidad con Android, iOS y web y proporciona todas las herramientas necesarias para cubrir las necesidades de la aplicación.

## 4.2 UML de las entidades y las clases

Esta aplicación, al estar desarrollada de forma nativa en Android, se compondrá del tipo de clases del que se suele hacer uso en este entorno: actividades y fragmentos.

Generalmente, podríamos decir que, una actividad es un componente de la aplicación que contiene una pantalla con la que el usuario puede interactuar. Lo habitual en Android es estructurar la aplicación por actividades, siendo cada una de ellas una pantalla de la aplicación. Las actividades se componen de dos recursos, un archivo .java donde reside la lógica y un archivo xml con el aspecto gráfico.

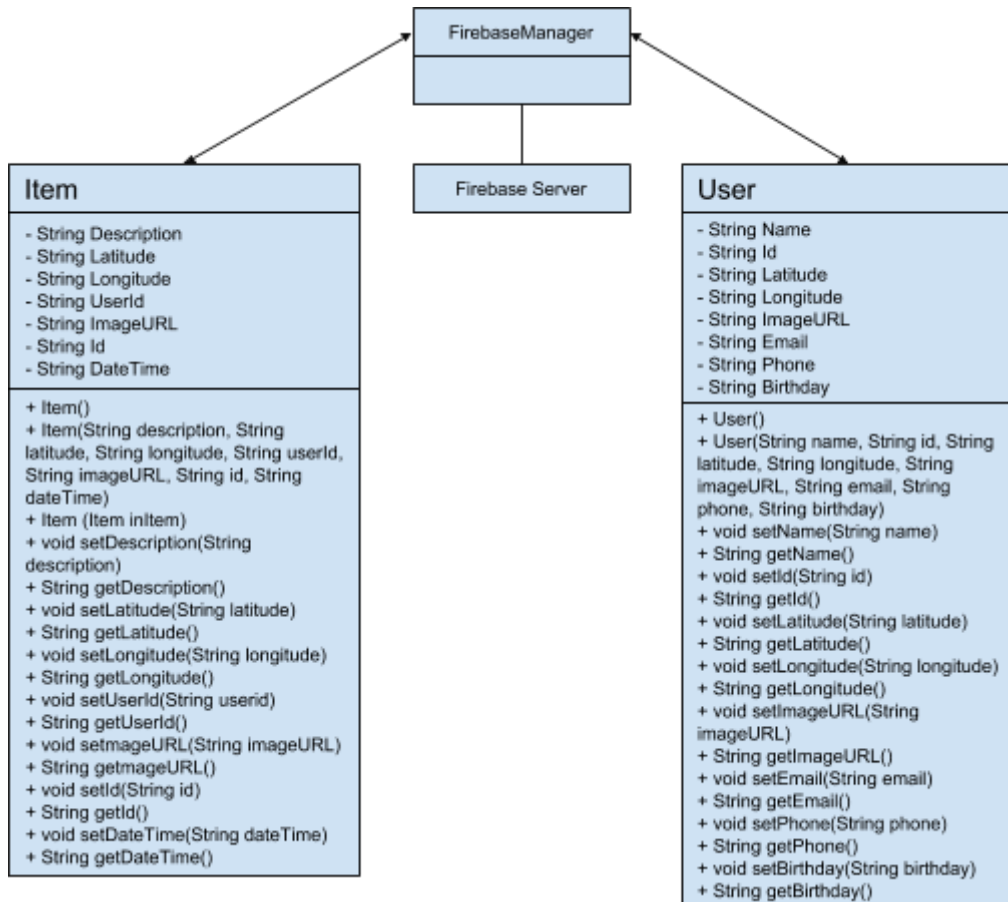
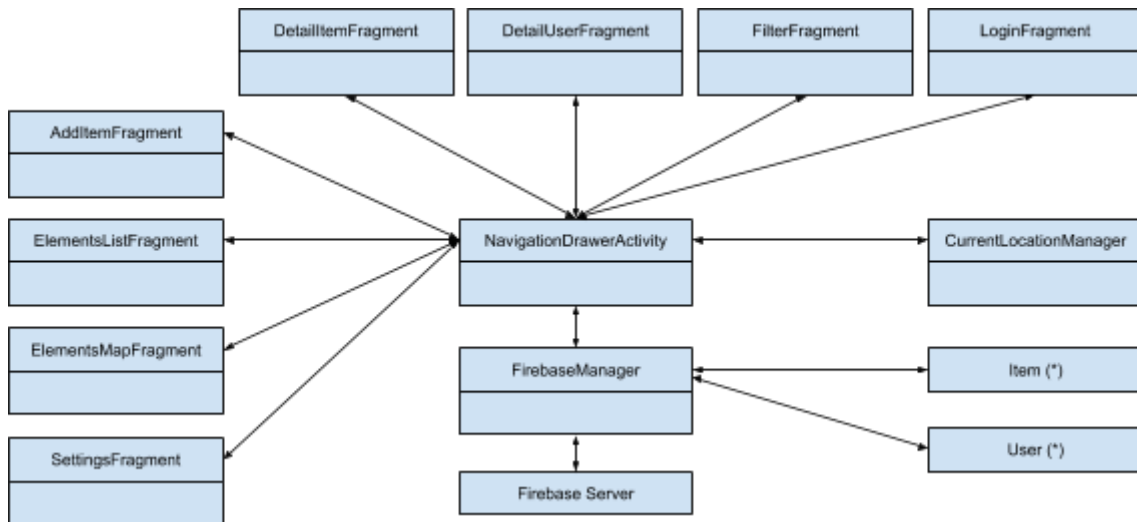
Por otro lado, un fragmento, en Android, representa un comportamiento o una parte de la interfaz de usuario en una actividad. Se pueden combinar múltiples fragmentos en una actividad o reutilizar un mismo fragmento en distintas actividades. También se compone de un archivo .java donde figura la lógica y un archivo xml con la apariencia gráfica.

En el caso de esta aplicación, optamos por compartir casi los mismos elementos gráficos entre pantallas, conservando el menú lateral y los botones de la barra superior o ActionBar. Por ese motivo, hemos decidido integrar estos elementos comunes en una actividad principal (**NavigationDrawerActivity**) y, sobre esta actividad principal, utilizar fragmentos en lugar de actividades para cada pantalla de la aplicación. De esta manera, a medida que navegamos por la aplicación, se irá reemplazando el fragmento que mostramos en la parte central de la aplicación.

Esta clase principal, la actividad **NavigationDrawerActivity**, supondrá el nexo de unión ya no solo entre los distintos fragmentos, también con las demás clases utilizadas, como pueden ser las clases encargadas de la gestión de los datos (**FirestoreManager**) o las clases encargadas de gestionar la geolocalización del dispositivo (**CurrentLocationManager**).

A su vez, la clase encargada de la gestión de los datos con Firebase (**FirestoreManager**), utilizará las dos clases base que conforman el modelo de datos (**Item** y **User**), y nutrirá a la aplicación de los datos del servidor.

El diagrama de clases de la aplicación será el siguiente:

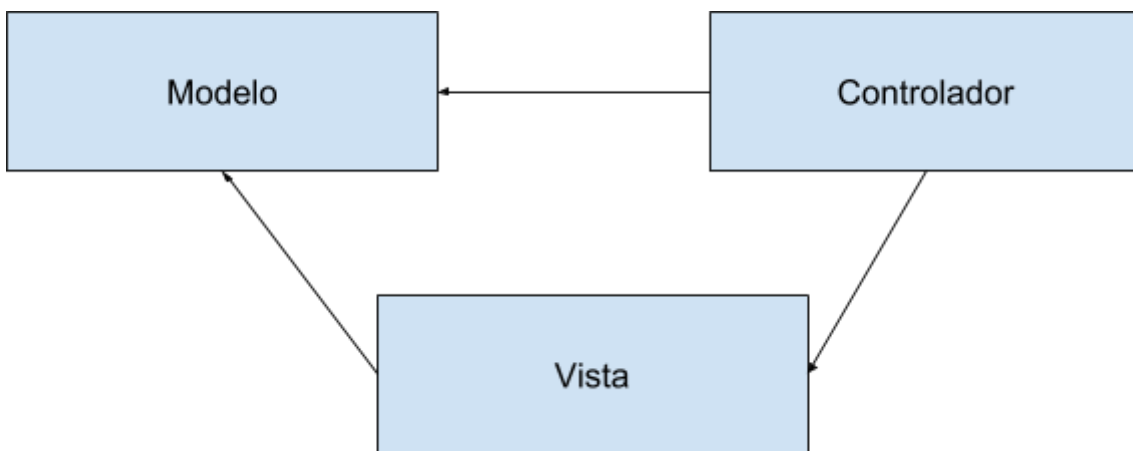




### 4.3 Diagrama explicativo de la arquitectura del sistema

Para el desarrollo de la aplicación se va a intentar utilizar el patrón de arquitectura MVC (Modelo Vista Controlador). El patrón consta de estos tres elementos:

- **Modelo:** La capa de datos, responsable de manejar la lógica y los datos. Estos datos serán los que se enviará a la vista para ser mostrados al usuario. En el caso de la aplicación que nos ocupa, el modelo lo representarán las clases que definen los dos objetos anteriormente comentados en el apartado 4.1, Item y User.
- **Vista:** Es el interfaz con el que va a interactuar el usuario. Es el encargado de mostrar los datos del modelo. En nuestra aplicación este componente vendrá representado por los distintos archivos xml de layout de que disponga el proyecto.
- **Controlador:** Es el intermediario entre el modelo y la vista. Responde a las acciones que el usuario va realizando y actualiza los datos del modelo según se necesite. Este elemento vendrá representado en nuestra aplicación por la parte lógica de las actividades y de los fragmentos, los archivos .java de las actividades.



## 5. Implementación

El proyecto se ha desarrollado nativamente para Android, mediante el lenguaje de programación Java, utilizando el IDE (entorno de desarrollo integrado) Android Studio.

Android Studio es el IDE oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA, un IDE para el desarrollo de programas informáticos. Se ha escogido este IDE para el desarrollo de la aplicación debido a que es el oficial y actualmente representa el estándar. Ahora mismo se encuentra en una fase estable y recibe actualizaciones constantemente.

Para compilar se utiliza Gradle, integrado dentro de Android Studio. Para ejecutar el código durante gran parte del desarrollo se ha utilizado un dispositivo estándar virtual con las siguientes características:

- Nexus 5X con:
  - API 24
  - Sistema operativo Android versión 7.0 Nougat
  - Espacio interno de 2048 MB.

La imagen del sistema operativo Android que se ha elegido incorpora Google Play Store, lo que posibilita la instalación de Google Play Services, lo que a su vez posibilita la utilización de los mapas de Google Maps dentro de la aplicación.

### 5.1 Estructura del proyecto de Android Studio

Un proyecto de Android se divide en varias carpetas o elementos:

- **Código fuente (ubicado en la carpeta /app/src/main/java):** Es la parte donde se guarda todo el código fuente de la aplicación.
- **Recursos (ubicado en la carpeta /app/src/main/res/):** Aquí se ubican los ficheros de recursos que necesita el proyecto: imágenes, layouts de pantallas, elementos de menús, etc
- **Fichero AndroidManifest.xml (ubicado en /app/src/main/):** Contiene la definición en XML de muchos de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, icono, ...), sus componentes (pantallas, servicios, ...), o los permisos necesarios para su ejecución.
- **Fichero build.gradle (ubicado en /app/):** Contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android

que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

- **Librerías externas (ubicadas en la carpeta /app/libs/)**
- **Elementos compilados (ubicados en la carpeta /app/build/):** Cada vez que ejecutamos la aplicación es en esta carpeta donde se guardan los archivos generados.

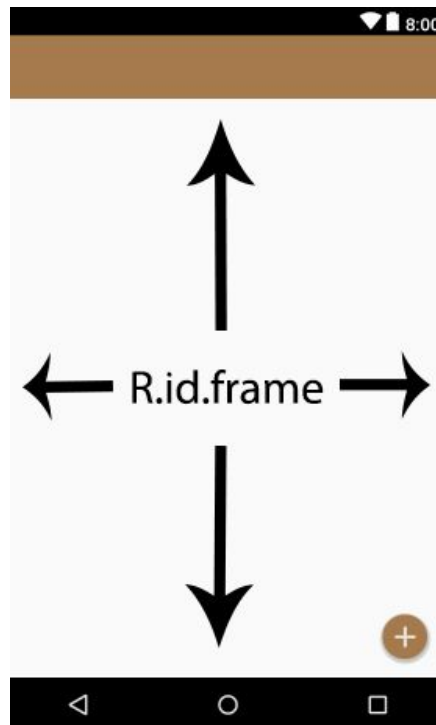
### 5.1.1 Código fuente y archivos de recursos asociados

En este apartado se van a describir los archivos de código más importantes de la aplicación y cómo funcionan entre ellos.

Para el desarrollo se ha intentado seguir una estructura MVC, aunque no siempre ha sido posible. A continuación se describe de qué manera se ha estructurado archivo a archivo.

#### 5.1.1.1 NavigationDrawerActivity.java

La aplicación está estructurada entorno a esta actividad, desarrollándose muchas de las funcionalidades dentro de ella. Cada una de las pantallas que conforman la aplicación están formadas por fragmentos, la mayoría lanzados directamente desde esta actividad, utilizando la región delimitada por el elemento **R.id.frame** presente en el archivo de recursos **content\_navigation\_drawer.xml**.



Así, cada vez que se quiere presentar un nuevo fragmento, se realiza una llamada a un objeto del tipo **FragmentManager** y se realiza un reemplazo de **R.id.frame** por el fragmento que se quiera inicializar. Al realizar el reemplazo, se llama al método **addToBackStack**, lo que se traduce en que si el usuario más adelante presiona el botón de atrás del dispositivo, el fragmento que se ha inicializado se liberará, para pasar a mostrarse el fragmento que se encontraba operativo anteriormente.

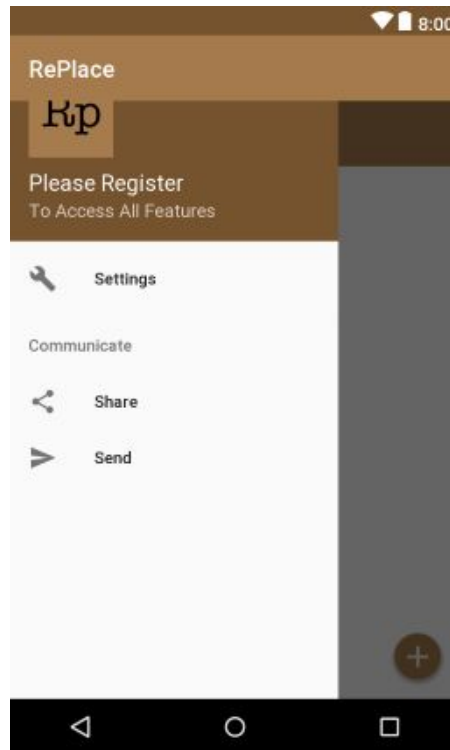
De esta manera, desde esta actividad principal, se van presentando sobre la parte central de la pantalla los fragmentos que se vayan necesitando, manteniendo cierta independencia entre pantallas pero a la vez, centralizando las funciones y variables principales de la aplicación en esta actividad.

Esta estructura empleada, aunque no representa un modelo MVC puro, si que tiene una estructura que se podría asemejar, representandose de la siguiente manera:

- Las vistas están representadas como los archivos layout xml donde se configura el aspecto y distribución de cada uno de los elementos que el usuario verá en pantalla.
- Los controladores vienen representados por las actividades (en este caso la actividad **NavigationDrawerActivity**) y los distintos fragmentos que se van inicializando conforme se van necesitando.
- Los modelos quedarán representados por las clases **ItemContent** y **UserContent**, las cuales utilizan a su vez las clases **Item** y **User**, donde se ha configurado la estructura que tendrán los datos que maneja la aplicación y donde se han configurado las operaciones que se realizan directamente con ellos.

Además, desde la actividad que nos ocupa, gestionamos varios elementos de la UI para disparar los distintos fragmentos:

- El propio **NavigationDrawer** (o panel de navegación lateral) que utilizamos con distintos fines:
  - Mostrar el nombre y el email del usuario en el caso de que esté logueado.
  - Acceder a la pantalla de login o registro si es que aun no estamos logeados (fragmento **LoginFragment**).
  - Acceder a la pantalla de perfil de usuario en el caso de que estemos logeados (fragmento **DetailUserFragment**)
  - Acceder a la pantalla de ajustes de la aplicación (fragmento **SettingsFragment**).
  - Permitir a los usuarios compartir la aplicación.



En la imagen de encima de estas líneas podemos ver el aspecto del archivo de layout **activity\_navigation\_drawer.xml**, archivo que contiene el elemento **NavigationView**, donde irá montado el menú desplegable.

- El **FloatingActionButton**, o botón flotante, que se utilizará para añadir nuevos ítems (fragmento **AddItemFragment**).



- La **ActionBar**, o barra de acción, donde se ubican los botones para cambiar la manera en que se muestran los ítems, en mapa o en lista (fragmentos **ElementsListFragment** y **ElementsMapFragment**), o el botón que permite lanzar el filtrado de los mismos (fragmento **FilterFragment**).



Desde esta actividad también se inicializan varias funcionalidades principales de la aplicación:

- La comunicación con Firebase, tanto el login, con email o mediante cuenta de Google, como el registro y la recuperación de la información del servidor conforme haya cambios. Se ha incorporado una clase intermedia llamada **FirestoreManager** que ayuda a centralizar la gestión de Firebase pero que, por la tipología en la que, por ejemplo, funciona el login con Google, existiendo ciertas dependencias con las actividades, no ha sido posible realizar una separación total de las funcionalidades.
- El posicionamiento mediante el GPS del dispositivo y la actualización de la ubicación en tiempo real. Para ello, se ha creado una clase que se encarga de esta gestión, llamada **CurrentLocationManager**. Esta clase, también debido a las particularidades de la arquitectura nativa de Android, no ha podido abstraerse completamente de la actividad que la ejecuta, por lo que no ha sido posible realizar una separación total de las funcionalidades.

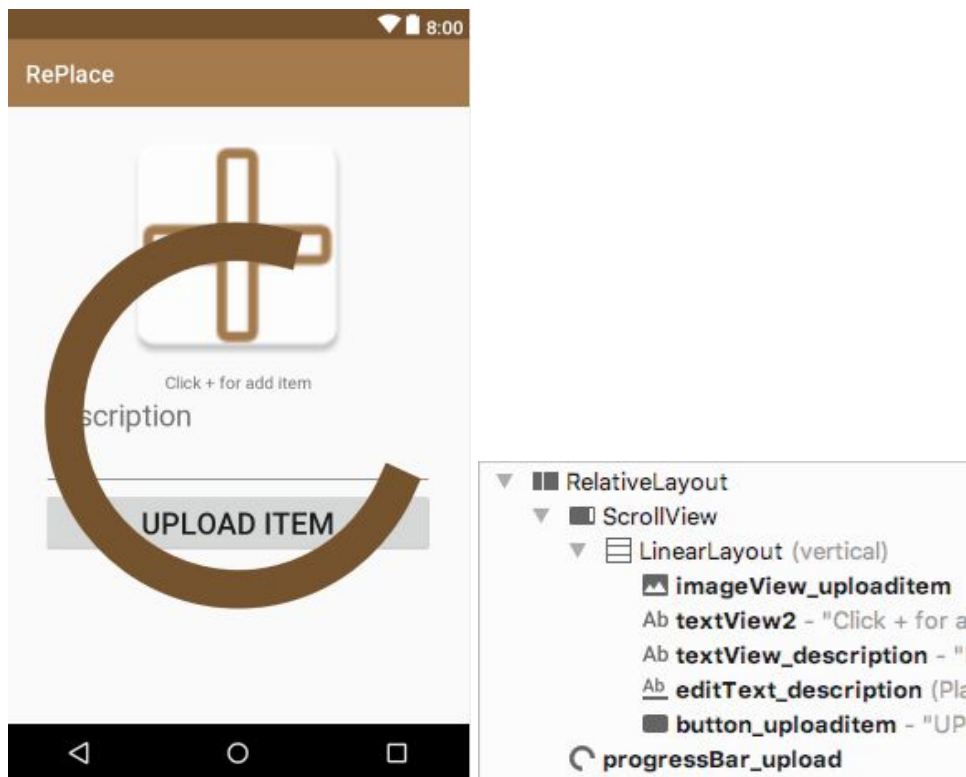
#### 5.1.1.2. AddItemFragment.java y fragment\_add\_item.xml

Desde este fragmento se realiza la recolección de los datos que el usuario quiera aportar para subir un nuevo ítem al servidor de la aplicación. Permite realizar subidas de artículos con foto, con descripción o con ambos.

Esta aplicación se basa en la geolocalización y la inmediatez. Es por estos dos motivos que no permite el uso de imágenes realizadas previamente. Por un lado los ítems una vez están en la calle tienen una elevada volatilidad. Cuando hay un objeto en la calle hay altas probabilidades que en poco tiempo ya no se encuentre disponible o que sufra algún tipo de deterioro. Por ese motivo tanto la foto como la subida tienen que ser casi inmediatas. Por otro lado, al hacer la fotografía a través de la aplicación en tiempo real, la aplicación puede obtener la información de geolocalización y subirla conjuntamente con el ítem. De esta manera los demás usuarios disponen de la ubicación exacta en el momento de realizarla, lo que facilita su localización.

Este fragmento utiliza el archivo de recursos **fragment\_add\_item.xml**. Este archivo está formado por un **RelativeLayout** que, a su vez, contiene un **ScrollView**. Este **ScrollView** permite que al escribir la descripción o etiquetas del ítem, el usuario pueda visualizar lo que está escribiendo. Si no se utilizara esta **ScrollView**, el campo que el usuario está editando quedaría escondido detrás del teclado de Android.

Dentro del **ScrollView** tenemos los elementos de la pantalla: una **progressBar** que abarca toda la pantalla (visible mientras se realiza la carga de la misma) y un **LinearLayout**, dentro del cual se ubican organizados verticalmente todos los componentes de la UI necesarios para realizar la subida del ítem.

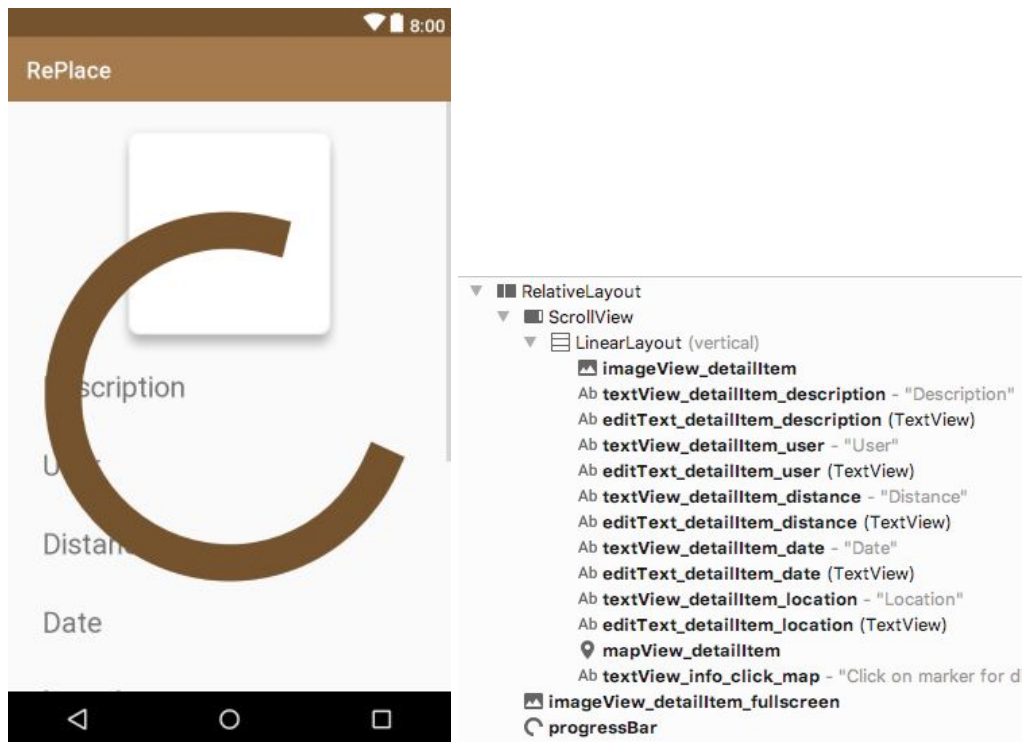


### 5.1.1.3 DetailItemFragment.java y fragment\_detail\_item.xml

Desde este fragmento se muestra el detalle del ítem seleccionado previamente desde la lista de elementos o mediante el mapa de ítems. Es similar al fragmento anterior pero con ligeras diferencias:

- Únicamente se muestran los datos, sin posibilidad de editar ningún campo.
- Se muestra la foto del ítem, con la posibilidad de verla ampliada haciendo click encima. Para ello internamente se recupera la imagen a través de **FirestoreStorage**. Mientras se realiza la descarga se muestra un ProgressBar animado.
- Se muestra el nombre del usuario que ha subido el ítem en el caso de estar logueado en el momento en que la subió.
- Se muestra la distancia que hay hasta el ítem desde nuestra ubicación actual.
- Se muestra el nombre de la calle y población donde se encuentra el mismo. Para ello se utiliza la clase **Geocoder**, la cual dispone de un método llamado **getFromLocation** el cual si le introducimos una latitud y longitud nos devuelve las posibles direcciones que hay en la misma.
- Se muestra la ubicación del elemento en un mapa (con la posibilidad de conseguir las indicaciones para llegar). Para realizar esta acción hemos utilizado un elemento **googleMaps** sobre un **MapView**.
- Si el usuario que ha subido el artículo es el mismo que el que se encuentra logueado en ese momento, aparecerá un icono en la parte

superior derecha en forma de cubo de basura que permitirá eliminar el ítem.



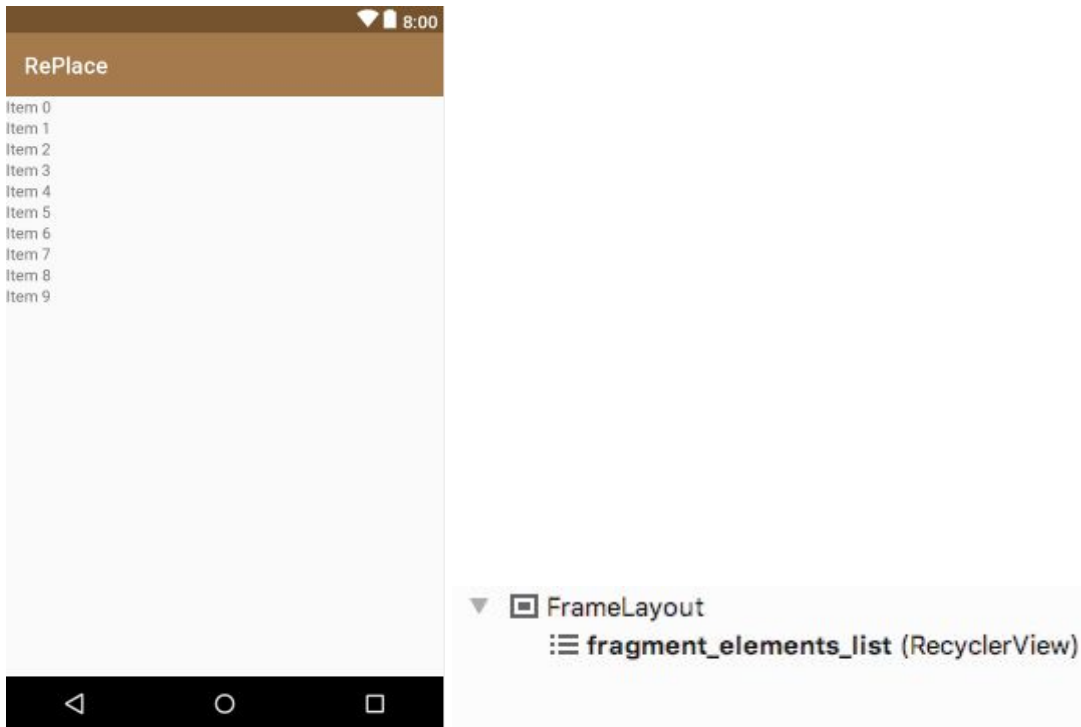
Como en el fragmento anterior, se ha incorporado un elemento del tipo **ScrollView** en casi la raíz de la jerarquía de elementos de la escena xml para permitir desplazarse por la totalidad de la vista.

#### 5.1.1.4 ElementsListFragment.java y fragment\_elements\_list.xml

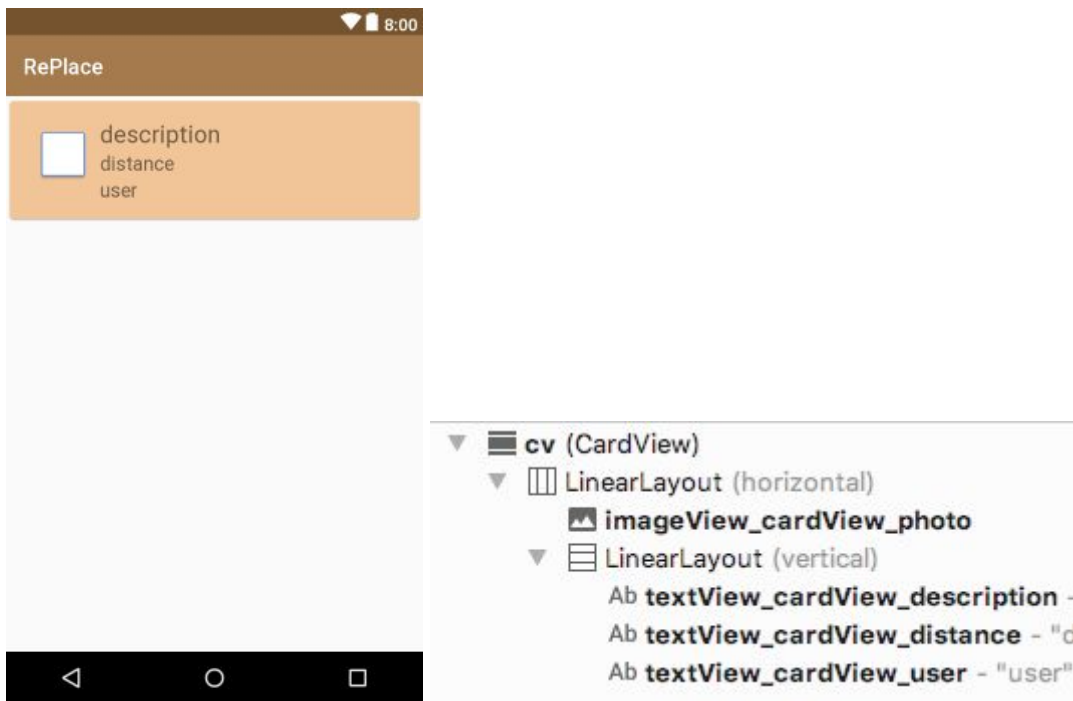
En este fragmento se realiza la visualización de los ítems a modo de lista que cumplan con los criterios de filtraje fijados en el fragmento de **FilterFragment**. Debido a que la aplicación se basa en la proximidad a los objetos, sobretodo por la alta volatilidad de los mismos al encontrarse en la calle en muchos casos, el orden de la lista se ha hecho por distancia hasta ellos, de más cercano a más lejano.

Este fragmento utiliza el archivo de recursos **fragment\_elements\_list.xml**. Este archivo está formado por un **FrameLayout** y dentro de este, a su vez, un **RecyclerView**. A este **RecyclerView** se le asigna un adapter que, a su vez, fija mediante el archivo de recursos **fragment\_cardview\_elements\_list.xml** el aspecto de cada uno de los ítems de la lista. En el archivo **fragment\_cardview\_elements\_list.xml** cada ítem está formado por un **LinearLayout** horizontal que, dentro suyo, incluye un elemento del tipo **ImageView** y otro **LinearLayout**, en este caso con orientación vertical, el cual contiene la leve descripción del ítem, la distancia hacia él y el usuario que lo ha subido.





fragment\_element\_list.xml

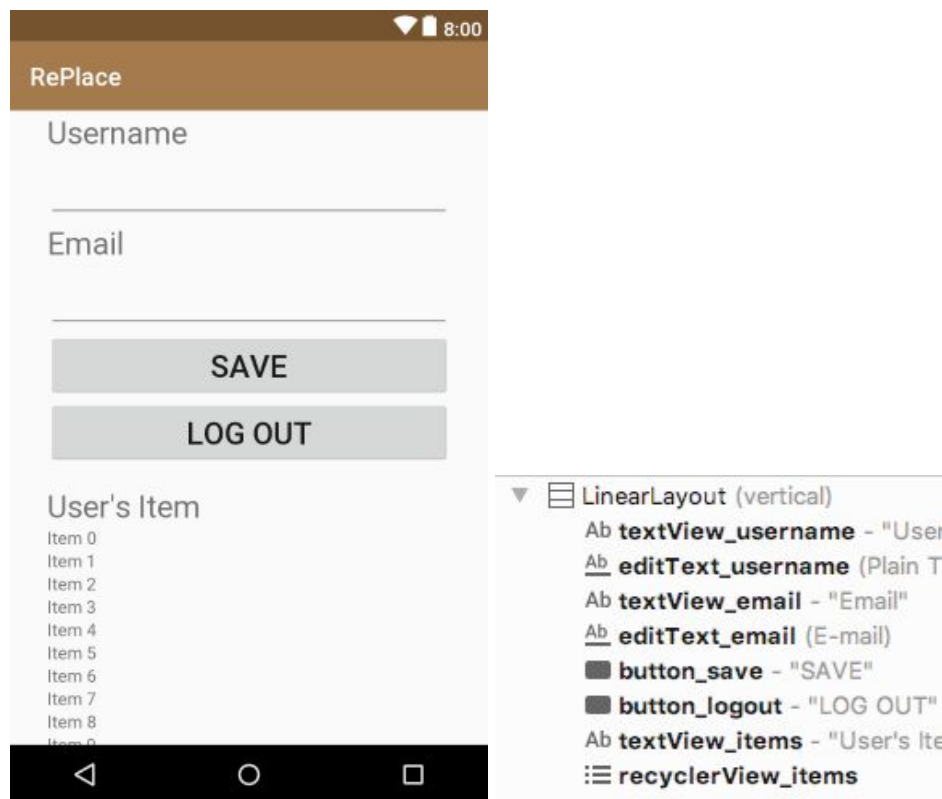


fragment\_cardview\_elements\_list.xml

### 5.1.1.5 DetailUserFragment.java y fragment\_detail\_user.xml

Este fragmento muestra el detalle del perfil de usuario: muestra el nombre, email del usuario, dos botones para poder salvar los cambios y desloguearse y, si el usuario ha subido algún artículo, una lista en la parte inferior de pantalla.

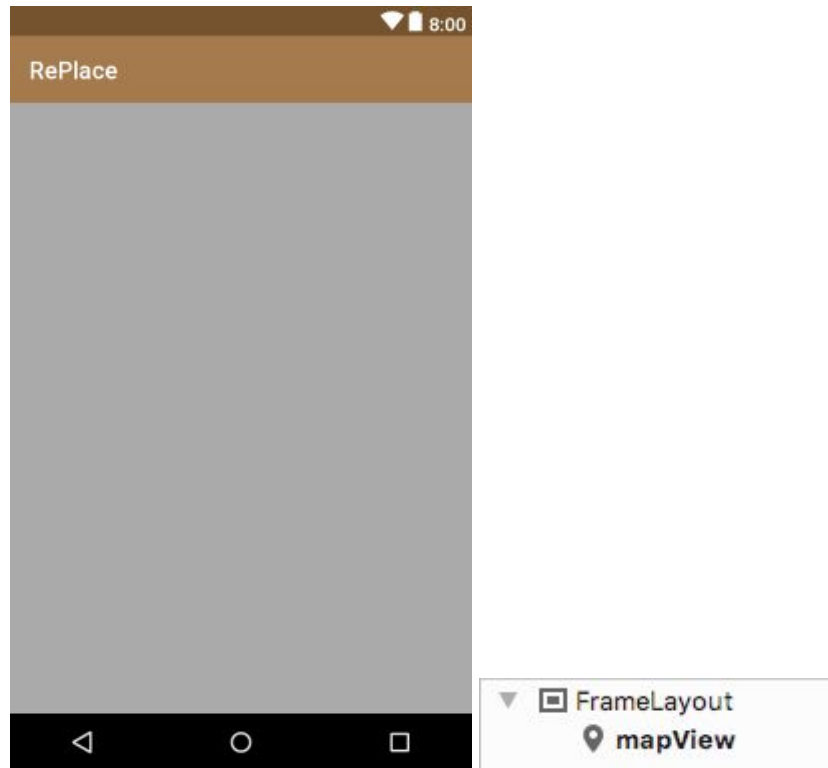
Este fragmento utiliza el archivo de recursos **fragment\_detail\_user.xml**. Este archivo está formado por un **LinearLayout** con orientación vertical dentro del cual se encuentran los distintos elementos de la pantalla y, finalmente, un **RecyclerView** que servirá de container para la lista de elementos del usuario. Este **RecyclerView** utiliza los mismos elementos y el mismo funcionamiento que el del apartado anterior.



### 5.1.1.6 ElementsMapFragment.java y fragment\_elements\_map.xml

En este fragmento se realiza la visualización de los elementos en una **MapView** mediante Google Maps. Es el fragmento que se lanza por defecto al arrancar la aplicación.

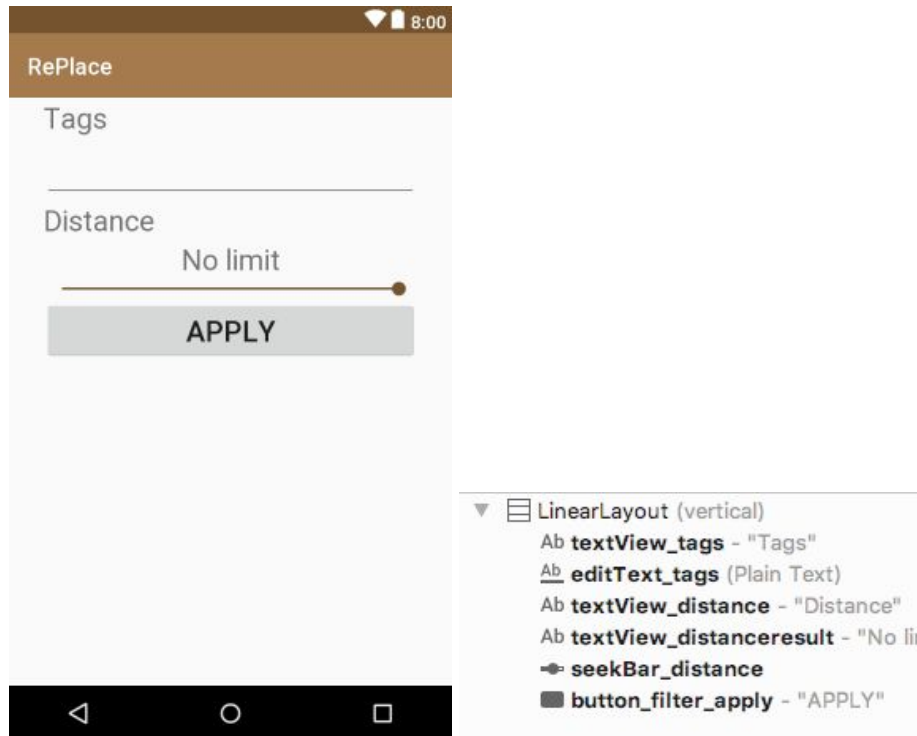
Este fragmento utiliza el archivo de recursos **fragment\_elements\_map.xml** donde simplemente se utiliza un **FrameLayout** y, dentro de este, un **MapView** sobre donde se cargarán los mapas y marcadores de cada elemento.



#### 5.1.1.7 FilterFragment.java y fragment\_filter.xml

En este fragmento se inicializa la vista con las opciones de filtrado disponibles: filtrado por tag o filtrado por distancia.

Como archivo de recursos se utiliza el archivo **fragment\_filter.xml**, el cual, se compone primeramente de un **LinearLayout** con orientación vertical y, dentro de este, un caja de texto del tipo **EditText**, para especificar que es lo que queremos buscar, y una barra deslizador **SeekBar** para establecer la distancia máxima sobre la que queremos buscar. Disponemos de un **Button** para confirmar y aplicar los nuevos criterios de búsqueda.



### 5.1.1.8 LoginFragment.java y fragment\_login.xml

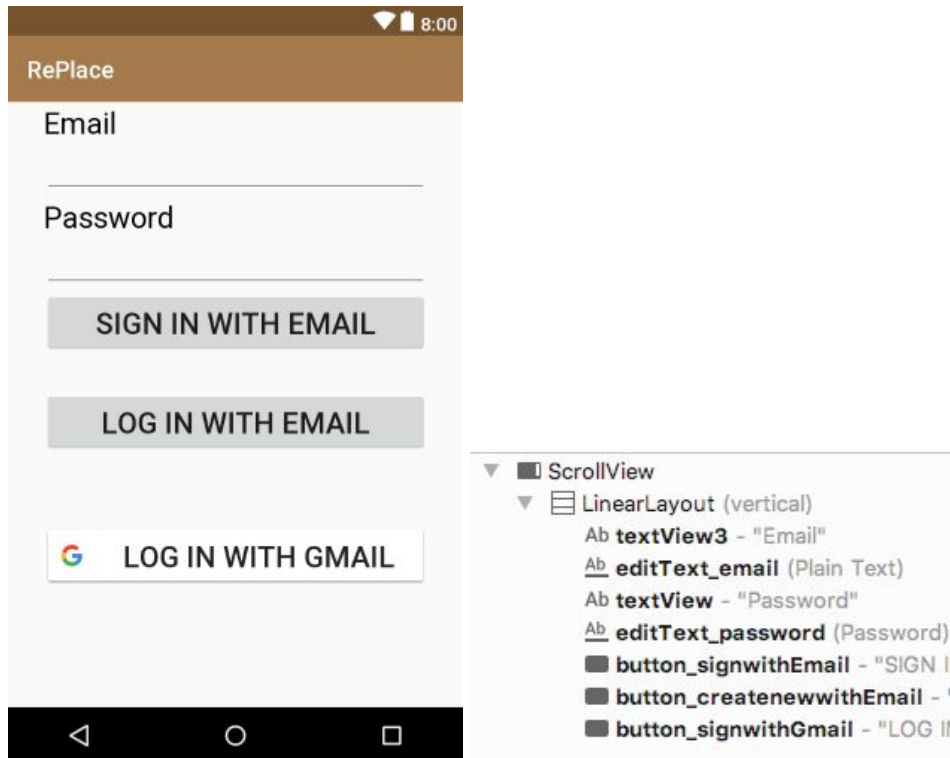
En este fragmento se realiza el registro de usuarios o el login de usuarios mediante email o cuenta de Google a través de la clase **FirebaseManager**.

Su funcionamiento se basa en tres eventos `setOnClickListener`, uno para cada botón de los que intervienen en las tres acciones:

- **Login de usuario con email y contraseña:** Para el login de usuario por email necesitaremos que el email esté registrado en Firebase previamente y que el password coincida con el establecido. Para realizar el login llamamos al método **signInWithEmailAndPassword** a través de un objeto **FirebaseAuth**. Una vez Firebase compruebe la validez del email y contraseña escritos, disparará el evento **OnCompleteListener** donde si todo es correcto se procederá a confirmar al usuario el login. Si algo ha salido mal se informará al usuario que ha habido un error.
- **Registro de usuario con email y contraseña:** Para el registro de nuevos usuarios mediante email y password llamaremos al método **createUserWithEmailAndPassword** a través de un objeto **FirebaseAuth**. Como en el caso anterior, una vez Firebase realice la operación, se disparará el evento **OnCompleteListener** donde si todo es correcto se procederá a confirmarle al usuario. Si el registro no se puede llevar a cabo se le notificará al usuario.
- **Login de usuario mediante cuenta de Google:** Para realizar este login inicializamos un objeto **GoogleSignInOptions** y lo añadimos a **GoogleApiClient**. A partir de este objeto podemos conseguir un Intent el cual lanzamos mediante **startActivityForResult**. El resultado lo

recuperamos en **onActivityResult** y si es satisfactorio realizamos el login con ese usuario.

Este fragmento utiliza el archivo de recursos **fragment\_login.xml**, archivo compuesto en primer lugar por un **ScrollView** y, a su vez, un **LinearLayout** de orientación vertical. Dentro de este **LinearLayout** encontramos dos campos de texto editables **EditText** y tres botones, uno para logearse con email y password, otro para registrarse con email y password y un tercero para logearse con Google.



#### 5.1.1.10 ItemContent.java

Esta clase es la que contiene el modelo de datos de los ítems. Por un lado tenemos la clase **ItemContent**, la cual se encarga de gestionar y permite realizar las operaciones más comunes sobre una lista de Items. Por otro lado tenemos la clase **Item** del ítem en sí, que se compone de tres constructores y los setters y getters correspondientes.

#### 5.1.1.11 UserContent.java

Esta clase es la que contiene el modelo de datos de los usuarios. Como en el caso anterior, tenemos la clase **UserContent**, la cual nos permite realizar las operaciones más habituales sobre un **ArrayList** de objetos **User** y, por otro lado, tenemos la clase **User**, la cual se compone de un par de constructores y sus setters y getters correspondientes.

### 5.1.1.12 CurrentLocationManager.java

Esta clase es la que se encarga de gestionar la geolocalización del smartphone. En primer lugar se ocupa de preguntarle al usuario si le permite obtener acceso a su ubicación mediante el GPS. A partir de que el usuario dé su consentimiento, la aplicación inicializa y configura un objeto **FusedLocationProviderClient** que le permite obtener de manera cíclica la última ubicación conocida del dispositivo.

### 5.1.1.13 FirebaseManager.java

Esta clase es la que se encarga de gran parte de la comunicación con **Firestore**. Desde ella se realizan varias operaciones con el servidor, entre ellas, por ejemplo, el registro de nuevos usuarios, añadir estos usuarios a la base de datos, actualizar los datos de un usuario en el servidor, las operaciones de login, etc.

Cuando la base de datos sufre un cambio como, por ejemplo, la supresión de un objeto, un evento **onDataChange** se dispara dentro de esta clase. En ese momento se actualiza la información de que dispone la aplicación y se actualizan los ítems que se muestran en el mapa o lista.

## 5.1.2 AndroidManifest.xml

El archivo AndroidManifest.xml proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la app.

En dicho fichero se han tenido que realizar pequeñas modificaciones para el correcto funcionamiento de la app:

- Se ha tenido que añadir la clave para que los servicios de localización de la API de Google Maps funcionaran.

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="@string/google_maps_key" />
```

- Se han tenido que añadir unas cuantas líneas para permitir el acceso a determinadas funciones como el acceso a Internet, a información referente al estado de la red y a la localización.

```
<uses-permission  
    android:name="android.permission.INTERNET" />
```

```
<uses-permission
android:name="android.permission.ACCESS_NETWORK_ST
ATE" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOC
ATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATIO
N" />
```

### 5.1.3 build.gradle

El archivo build.gradle contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

En el caso de esta aplicación, se ha definido como la versión mínima del SDK para ejecutar la aplicación la 14.

Aparte se han tenido que añadir las dependencias que han hecho falta para Firebase y para Google Play Services:

- Firebase:

```
implementation 'com.google.firebase:firebase-database:15.0.0'
implementation 'com.google.firebase:firebase-auth:15.0.0'
implementation 'com.google.firebase:firebase-storage:15.0.2'
```

- Google Play Services:

```
implementation 'com.google.android.gms:play-services-maps:15.0.0'
implementation
'com.google.android.gms:play-services-location:15.0.0'
implementation 'com.google.android.gms:play-services-auth:15.0.1'
```

### 5.1.4 Servidor Firebase

Firebase es una plataforma de back-end que dispone de una serie de herramientas para el desarrollo de aplicaciones: el almacenamiento y sincronización de datos en la nube, medición del comportamiento del usuario y soluciones para monetizar productos.

En el caso de esta aplicación se han utilizado varias características, tal y como se muestra en el apartado anterior de build.gradle:










- **Autenticación:** La aplicación permite al usuario autenticarse mediante email y password o mediante la utilización de una cuenta de Google.
- **Base de datos en tiempo real:** La aplicación accede a esta base de datos en tiempo real y es notificada cada vez que se realiza un cambio en ella. Ello facilita enormemente la actualización de los datos que se muestran en ella.



Ejemplo ítem en la base de datos online de Firebase

- **Almacenamiento:** Al realizar una foto con la aplicación, se comprime en un archivo del tipo jpg. y se almacena mediante **FirestoreStorage** en una ubicación distinta a donde reside la base de datos. Cuando se necesita la imagen solo hay que acudir a la url guardada en el campo **imageURL** del objeto y descargarla del servidor.



<input type="checkbox"/>	Nombre	Tamaño	Tipo	Última modificación
<input type="checkbox"/>	 a.jpg	19,9...	image/jpeg	4 jun. 2018
<input type="checkbox"/>	 bv.jpg	37,9...	image/jpeg	3 jun. 2018
<input type="checkbox"/>	 fb.jpg	18,7...	image/jpeg	29 may. 2...
<input type="checkbox"/>	 fwfsxfttkw.jpg	46,5...	image/jpeg	1 jun. 2018
<input type="checkbox"/>	 hcyhixsnie.jpg	41,5...	image/jpeg	29 may. 2...
<input type="checkbox"/>	 ivcl.jpg	31,8...	image/jpeg	2 jun. 2018
<input type="checkbox"/>	 kectqnwotuubhawot.jpg	33,1...	image/jpeg	2 jun. 2018
<input type="checkbox"/>	 legypvwt.jpg	21,4...	image/jpeg	31 may. 2...
<input type="checkbox"/>	 nrpvjxehydbdriyx.jpg	27,8...	image/jpeg	29 may. 2...

Ejemplo de lista de imágenes guardadas con Firebase Storage

Cabe decir que debido a la idea de querer hacer lo máximo accesible el contenido de la aplicación a todos los usuarios, las reglas de acceso a la base de datos y al almacenamiento de **FirebaseStorage** se han obviado, por lo que ahora mismo todo el mundo puede escribir y leer el contenido sin restricciones.

Para ello se han definido estas reglas de la siguiente manera:

- Base de datos:

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

- Almacenamiento:

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if true
    }
  }
}
```

## 6. Líneas futuras de desarrollo

A pesar de haber conseguido una aplicación funcional, por falta de tiempo no se han podido desarrollar todas las funcionalidades y mejoras que se hubieran querido. La realidad es que, planificaciones más o menos acertadas aparte, un proyecto así y su documentación requieren de una gran dedicación, sobretodo a la hora de realizarlo en el tiempo estándar estipulado. Por ese motivo, se han dejado pendiente una serie de desarrollos futuros que muy probablemente se lleven a cabo a posteriori de la entrega:

- **Aspecto gráfico:** Mejorar el aspecto gráfico de varias pantallas: Cambiar la ubicación y aspecto de varios botones, mejorar el diseño general de la aplicación intentando que sea un poco más atractiva para el usuario final.
- **Sistema de notificaciones:** Implementar el sistema de notificaciones y su configuración en la pantalla de settings o ajustes. La idea sería que este sistema generara una notificación al usuario cuando un nuevo item ingresase al sistema y cumpliera los filtros impuestos por él.
- **Chat entre usuarios:** Añadir la opción de chat en la vista detallada de los ítems para poder entablar conversación entre el usuario que sube el ítem y el usuario interesado en el mismo.
- **Orden elementos:** Permitir elegir el tipo de orden que se va a seguir en la lista de objetos: por distancia, antigüedad o número de visitas.
- **Iteraciones con los usuarios de prueba:** Un problema que no se planteó bien en la fase de la planificación fueron las iteraciones con los usuarios de prueba. Las fases de pruebas se han cimentado sobre varios usuarios que han ido recibiendo versiones de la aplicación conforme se iba avanzando en el desarrollo pero no de una manera periódica y metódica. En esta fase se deberían haber registrado todas las aportaciones de cada uno de los usuarios.

## 7. Conclusiones

Al término de este trabajo final de máster puedo decir que ha sido una experiencia muy provechosa y gratamente reconfortante. En este proyecto he podido aplicar muchos conocimientos de los que he ido aprendiendo a lo largo del máster, por lo que acabo con la sensación de poder afrontar nuevos proyectos con una solvencia relativamente alta.

En cuanto a la planificación considero que las metas que se fijaron eran un tanto optimistas y poco reales. Debido a la inexperiencia de no haber afrontado nunca la totalidad del desarrollo, planificación y documentación de un proyecto de este tipo, no se tuvieron en cuenta aspectos muy importantes que sin duda ayudaron a engrosar el número de horas dedicadas a él, como, por ejemplo, las necesarias refactorizaciones de código, el tiempo de elaboración de una memoria suficientemente detallada o la elaboración de un vídeo explicativo por alguien con pocas dotes comunicativas...

No os lo voy a negar, acabo esprintando, pero acabo, que es lo importante. Muchas gracias a todas y todos y nos vemos en futuros retos!

## 8. Glosario

**Android:** Sistema operativo basado en Linux.

**Android Studio:** Entorno de desarrollo integrado (IDE) para el desarrollo de aplicaciones para el sistema operativo Android.

**APK:** Extensión de las aplicaciones de Android.

**DCU (Diseño centrado en el usuario):** Metodología de diseño que tiene por objeto resolver necesidades concretas de los usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible.

**Google Play:** Tienda de aplicaciones de Google.

**GPS (Sistema de posicionamiento global):** Sistema que permite determinar en toda la tierra la posición de un objeto.

**IDE (Entorno de desarrollo integrado):** Entorno de programación.

**Java:** Lenguaje de programación originalmente desarrollado por Sun Microsystems, de propósito general, concurrente y orientado a objetos.

**SDK (Kit de desarrollo de software):** Conjunto de herramientas de desarrollo que permite al programador crear aplicaciones.

**Smartphone:** Dispositivo móvil de última generación, que dispone de pantalla táctil, que permite la conexión a internet y ejecutar aplicaciones.

## 9. Bibliografía

La bibliografía utilizada se ha basado en su totalidad en consultas a webs de internet.

A continuación enumero las más relevantes:

<a href="http://blog.teamtreehouse.com/add-navigation-drawer-android">http://blog.teamtreehouse.com/add-navigation-drawer-android</a>
<a href="https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/">https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/</a>
<a href="https://androidstudiofags.com/conceptos/que-es-un-activity-en-android">https://androidstudiofags.com/conceptos/que-es-un-activity-en-android</a>
<a href="https://androidstudiofags.com/tutoriales/modelo-vista-controlador-en-android-mvc">https://androidstudiofags.com/tutoriales/modelo-vista-controlador-en-android-mvc</a>
<a href="https://bbvaopen4u.com/es/actualidad/firebase-como-google-quiere-mejorar-las-aplicaciones-traves-de-los-datos">https://bbvaopen4u.com/es/actualidad/firebase-como-google-quiere-mejorar-las-aplicaciones-traves-de-los-datos</a>
<a href="https://developer.android.com/guide/components/activities?hl=es-419">https://developer.android.com/guide/components/activities?hl=es-419</a>
<a href="https://developer.android.com/guide/components/fragments?hl=es-419">https://developer.android.com/guide/components/fragments?hl=es-419</a>
<a href="https://developer.android.com/guide/practices/ui_guidelines/icon_design_menu">https://developer.android.com/guide/practices/ui_guidelines/icon_design_menu</a>
<a href="https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419">https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419</a>
<a href="https://developer.android.com/guide/topics/ui/dialogs?hl=ES">https://developer.android.com/guide/topics/ui/dialogs?hl=ES</a>
<a href="https://developer.android.com/guide/topics/ui/layout/recyclerview">https://developer.android.com/guide/topics/ui/layout/recyclerview</a>
<a href="https://developer.android.com/reference/android/app/Activity.html">https://developer.android.com/reference/android/app/Activity.html</a>
<a href="https://developer.android.com/reference/android/app/AlertDialog">https://developer.android.com/reference/android/app/AlertDialog</a>
<a href="https://developer.android.com/reference/android/app/FragmentManager.html">https://developer.android.com/reference/android/app/FragmentManager.html</a>
<a href="https://developer.android.com/reference/android/support/design/widget/NavigationView.OnNavigationItemSelectedListener.html">https://developer.android.com/reference/android/support/design/widget/NavigationView.OnNavigationItemSelectedListener.html</a>
<a href="https://developer.android.com/reference/android/support/v4/app/ActionBarDrawerToggle">https://developer.android.com/reference/android/support/v4/app/ActionBarDrawerToggle</a>
<a href="https://developer.android.com/reference/android/support/v4/app/FragmentTransaction">https://developer.android.com/reference/android/support/v4/app/FragmentTransaction</a>
<a href="https://developer.android.com/reference/android/support/v7/widget/RecyclerView">https://developer.android.com/reference/android/support/v7/widget/RecyclerView</a>
<a href="https://developer.android.com/studio/intro/?hl=es-419">https://developer.android.com/studio/intro/?hl=es-419</a>
<a href="https://developer.android.com/studio/write/firebase">https://developer.android.com/studio/write/firebase</a>
<a href="https://developer.android.com/training/appbar/actions">https://developer.android.com/training/appbar/actions</a>
<a href="https://developer.android.com/training/basics/fragments/creating.html">https://developer.android.com/training/basics/fragments/creating.html</a>
<a href="https://developer.android.com/training/implementing-navigation/nav-drawer">https://developer.android.com/training/implementing-navigation/nav-drawer</a>
<a href="https://developer.android.com/training/location/receive-location-updates">https://developer.android.com/training/location/receive-location-updates</a>
<a href="https://developer.android.com/training/location/retrieve-current">https://developer.android.com/training/location/retrieve-current</a>
<a href="https://developer.android.com/training/permissions/requesting.html?hl=es-419">https://developer.android.com/training/permissions/requesting.html?hl=es-419</a>
<a href="https://developers.google.com/android/guides/setup">https://developers.google.com/android/guides/setup</a>

<https://developers.google.com/android/reference/com/google/android/gms/common/api/GoogleApiClient.Builder>

<https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi>

<https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient>

<https://developers.google.com/android/reference/com/google/android/gms/location/LocationResult>

<https://developers.google.com/identity/sign-in/android/sign-in>

[https://es.wikipedia.org/wiki/Java\\_\(lenguaje\\_de\\_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))

[https://es.wikipedia.org/wiki/IntelliJ\\_IDEA](https://es.wikipedia.org/wiki/IntelliJ_IDEA)

<https://firebase.google.com/docs/android/setup>

<https://firebase.google.com/docs/database/android/start/?authuser=0>

<https://google-developer-training.gitbooks.io/android-developer-advanced-course-practicals/unit-4-add-geo-features-to-your-apps/lesson-9-mapping/9-1-p-add-a-google-map-to-your-app/9-1-p-add-a-google-map-to-your-app.html>

<https://inducesmile.com/android-tutorials-for-nigerian-developer/android-load-image-from-url/>

<https://jarroba.com/programar-fragments-fragmentos-en-android/>

<https://medium.com/@fahedhermoza/notas-del-libro-programaci%C3%B3n-con-android-the-big-nerd-ranch-cap-2-46d9071d9e7e>

<https://platzi.com/blog/arquitectura-android-app/>

<https://stackoverflow.com/questions/26174527/android-mapview-in-fragment>

<https://www.desarrolloweb.com/articulos/android-que-es-una-activity-o-actividad.html>

<https://www.dev2qa.com/android-multiple-fragments-in-one-activity-example/>

<https://www.jetbrains.com/idea/>

<https://www.paradigmadigital.com/dev/versus-apps-hibridas-vs-apps-nativas/>

# 10. Anexo 1: Manual de usuario

## 10.1 Requerimientos

- Smartphone con sistema operativo Android versión 4.0 o superior.
- Smartphone con GPS
- Smartphone con conexión a internet

## 10.2 Descarga de la aplicación

Pendiente de subida

## 10.3 Navegación de la Aplicación

### 10.3.1 Pantalla de inicio

Una vez ubicado el icono de la aplicación en nuestra pantalla del teléfono o bien desde el menú de aplicaciones, presionar sobre la aplicación para que se abra de forma automática. El primer paso será un aviso referente a activar el GPS para poder proseguir utilizando la aplicación. Al darle al *ok* nos aparecerá el cuadro propio de sistema de Android preguntándonos si deseamos permitir el acceso de la aplicación al GPS. La aplicación necesita este acceso para funcionar convenientemente.

### 10.3.2 Pantalla principal

Con el GPS ya activo, nos encontraremos ante el menú principal de la aplicación, en que podemos localizar un menú desplegable, un filtro (con un menú desplegable incorporado), el mapa de la zona donde nos encontramos y un icono circular donde podremos acudir para subir objetos a la aplicación.

### 10.3.3 Registro y configuración

Con ello podemos acceder en la esquina superior izquierda a un panel de navegación lateral el cual muestra el texto *Please Register – To Acces All Features, Settings* y en nuevo apartado, *Communicate* dentro del cual podemos ver los textos *Share* y *Send*.

Si accedemos a *Please Register – To Acces All Features*, nos surgirá una nueva pantalla en la que podremos cumplimentar el apartado *Email* y *Password* para acceder de forma manual clicando en el comando *Sign In With Email* o bien *Log In With Email* si nos hemos registrado de forma previa. En esta misma pantalla podemos ver en la parte inferior el texto *Log In With Gmail* mediante el cual al clicar, podemos acceder a partir de nuestra cuenta personal de *Gmail*.

Una vez logueados en la aplicación si volvemos al menú comentado, podemos ver el texto *Username* con nuestro nombre y apellidos así como el *Email* con

nuestro correo personal y los comandos *Save* para guardar cambios en estos comandos previos o bien *Log out* para abandonar la sesión. Si ya hemos podido subir productos desde nuestra cuenta, podremos ver el texto *User's Item* donde podremos ver fotos de nuestros productos con la descripción y distancia hasta los mismos.

Volviendo al panel de navegación lateral, una vez identificados con nuestra cuenta podemos acceder al comando *Settings* donde en esta primera versión esta funcionalidad está por desarrollar.

Dentro del apartado *Communicate* podemos ver los apartados *Share* y *Send* funcionalidades que se encuentran de la misma manera pendientes de desarrollar.

### **10.3.4 Búsqueda**

Desde la pantalla inicial nos encontramos el apartado *Filter* el cual nos abre una pantalla donde podemos ver los textos *Tags* y *Distance*.

En la etiqueta *Tags* tenemos la opción de escribir palabras de filtro en tanto a los objetos que encontramos subidos a la aplicación, podemos escribir tumbona para consultar si hay tumbonas disponibles en la aplicación. Por otro lado podemos acudir a *Distance* para aplicar un nuevo filtro esta vez basado en la distancia entre el usuario y el objeto final. La forma de graduar esta distancia se realiza mediante una barra deslizable la cual nos da la opción de hacer filtro desde 1 Km a infinito, pudiendo consultar una distancia de 1 a 10km de forma gradual y posteriormente pasar a sin límite desde los 10 Km comentados.

Una vez aplicado el filtro de distancia podemos confirmar los filtros clicando en el comando *Apply*, el cual nos abrirá la pantalla de inicio donde podemos ver el mapa de la zona y los objetos disponibles mediante etiquetas localizadas en los puntos donde se encuentran los objetos.

Para consultar los objetos de forma más rápida, podemos picar en el icono localizado a la izquierda del comando *Filter*. Este icono nos mostrará los objetos disponibles según el filtro realizado ordenados por distancia hasta ellos y mostrando una foto en miniatura, nombre, distancia y el usuario que la ha subido. Si picamos en uno de los productos que nos interesan se nos abrirá una nueva pantalla con la foto del mismo, el apartado *Description* donde podemos ver el nombre, *User* en referencia a la persona que ha subido el producto, *Distance* con la distancia exacta hasta el producto, *Date* con la fecha de día, mes, año y hora concreta en la que el objeto fue introducido en la app y *Location* donde podremos ver tanto el número de la calle como el código postal, ciudad y país donde podemos encontrar el objeto.

### **10.3.5 Subida de producto**

Volviendo a las pantallas que muestran el resultado de las búsquedas, en la esquina inferior derecha encontramos un icono redondo con el símbolo "+". Este botón permite acceder a la pantalla para añadir un nuevo producto. Desde esta pantalla podremos añadir una foto realizada con el dispositivo y etiquetar



el producto. La ubicación del mismo vendrá marcada por la ubicación del terminal en ese momento.