Trabajo de Fin de Grado

GRADO DE INGENIERIA INFORMÁTICA
Facultad de Matemáticas
Universitat de Barcelona

My Travel Manager, aplicación Android para la recomendación de vuelos aéreos

---

Alejandro Muñoz Rabal

Director: Santi Seguí
Realizado en: Departament de Matemàtica Aplicada i Anàlisi. UB

Abstract

This Project consists on an Android application that allows the user to search for a flight, set up an alert on a flight create alerts to be notified when the price of a flight is of interest, or be notified when the system believes it is a good time to buy. To develop it, we have created a database that allows to manage the application data and background tasks have been included so the user does not need to be checking the application in order to know when to buy.

Resumen

Este proyecto consiste en una aplicación Android que permite al usuario realizar búsquedas de vuelos, crear alertas para ser avisado cuando el precio de un vuelo sea de su interés, o ser avisado cuando el sistema considera que es un buen momento de compra. Para la realización del proyecto se ha generado una base de datos que permite gestionar los datos de la aplicación y se han creado tareas en background que permiten al usuario no tener que estar pendiente de utilizar la aplicación para saber cuándo comprar.

Contenido

1. Introducción

Actualmente vivimos en una sociedad en la cual una de las principales preocupaciones es la economía, desde el ámbito doméstico al ámbito estatal. La gran variedad de oferta que ofrece el mercado para un mismo producto ha sufrido un aumento considerable con la evolución de la tecnología que estamos viviendo, hecho que resulta en una búsqueda y contraste de resultados constante de las diferentes ofertas de un producto para poder obtenerlo al mejor precio.

Un ejemplo claro de este hecho lo podemos encontrar en las compras realizadas por internet. La gran cantidad de proveedores disponibles junto con la gran cantidad de productos que existen dificultan el hecho de poder encontrar el producto que deseamos al mejor precio disponible. Si a estos factores, añadimos el hecho de que los precios se encuentran en un cambio constante, nos encontramos que muchas veces, la tarea de búsqueda y contraste es larga y costosa, lo que resulta en decisiones con falta de información y por ello, muchas veces, en pérdidas económicas.

Ante este problema y dada la necesidad de viajar de muchas personas, surge la idea de crear una aplicación Android que se encargue de buscar y avisar al usuario de cuando comprar un billete de avión simplemente indicándole el día y el destino al que se desea viajar. De aquí surge el proyecto de final de carrera My Travel Manager, que, junto a otro proyecto de carrera encargado de las recomendaciones, crea un sistema que avisa al usuario cuando es un buen momento de compra o cuando el billete tiene un precio inferior al introducido por el usuario.

La finalidad de este proyecto es diseñar y desarrollar una aplicación Android que permita a los usuarios buscar los vuelos que sean de su interés para una fecha determinada y establecer una alerta que la aplicación utilizará para avisarle de manera automática en caso de que el vuelo cumpla con sus requisitos. Mediante el uso del recomendador, también se encargará de avisar al usuario si éste considera que es un buen momento para realizar la compra. De esta manera, el usuario podrá estar actualizado de los precios sin la necesidad de estar consultando la aplicación constantemente.

Para la obtención de los datos de los vuelos, hemos utilizado la API de Google llamada QPX Express y para la gestión de datos de los usuarios, Firebase.

El trabajo pues, quedará compuesto por dos partes: el desarrollo de una aplicación Android con la cual el usuario interactuará y el sistema de recomendación que utilizará la aplicación como soporte (del cual no se hablará en este proyecto, ya que pertenece a otro compañero).

En una primera fase, se hablará del estudio de mercado y el diseño de la aplicación y posteriormente, en la implementación de la misma.

2. Tendencias de mercado

Hoy en día hay una gran variedad de aplicaciones para Android, pero lo que realmente distingue unas de otras son algunas de las funcionalidades o diseños propios de cada una.

Es por eso que, cuando vamos a diseñar una nueva aplicación, conviene analizar las tendencias del mercado para poder extraer las características que debería de tener nuestra aplicación.

En nuestro caso, al realizar una aplicación entorno a vuelos aeronáuticos, se decidió buscar en Google Play diversas aplicaciones para analizar dichas características.

Las aplicaciones seleccionadas para el estudio fueron:

- Vueling
- Skyscanner
- easyJet

Tanto Vueling como easyJet son aplicaciones que pertenecen a sus compañías de vuelos respectivamente, mientras que Skyscanner es un buscador genérico de vuelos.

A continuación, se detalla el análisis obtenido de cada una de las aplicaciones.

2.1. Vueling

Vueling es la aplicación de la compañía con el mismo nombre, que permita realizar búsquedas de vuelos, hoteles y hasta de coches de alquiler.

La aplicación se puede utilizar tanto con una cuenta de Vueling como de manera anónima, en este segundo caso, en el momento de realizar algún pago, se nos solicitaran los datos personales y económicos para poder realizarlo.

Para realizar búsquedas, Vueling permite distinguir entre vuelos de ida y de ida y vuelta, mientras que mantiene como parámetro compartido en ambos métodos el nombre de pasajeros, entre los cuales, a primera vista, solo podemos distinguir los adultos (ya que el botón con la opción de pasajeros para niños o bebés no nos aporta esta información a primera vista).

Una vez realizada la búsqueda, Vueling distingue el tipo de cabina en 3 clases distintas, ordenadas por calidad, de menor a mayor:

- Basic
- Optima
- Excellence

Los resultados de la búsqueda aparecen por filas y en 3 columnas, cada una de ellas para indicar el precio de la categoría correspondiente.

Una vez seleccionado el día y la franja horaria en la cual queremos el vuelo (ambas opciones se pueden cambiar sin tener que volver al menú anterior), podemos proceder a realizar el pago.

2.2. Skyscanner

Skyscanner es una aplicación genérica para la búsqueda de vuelos (es decir, no pertenece a ninguna compañía en concreto) y nos permite realizar búsquedas y crear alertas para que recibamos una notificación cuando el precio mínimo de la ruta cambie.

Como ya veíamos en el caso de Vueling, la aplicación puede ser utilizada con una cuenta o sin ella, pero para recibir notificaciones en el cambio del precio mínimo es necesaria la creación de una cuenta.

Para realizar una búsqueda, Skyscanner nos solicita cual es el lugar de origen y una vez hemos introducido éste, nos ofrece diferentes países donde buscar el destino, cosa que hace un poco confusa la búsqueda. Cabe destacar también que, aunque permite introducir el tipo de cabina y el tipo de pasajeros, la opción podría estar más visible, ya que se encuentra en un desplegable que no tiene demasiado contraste y no te fijas a primera vista.

Una vez hemos encontrado el destino, podemos ver un estado de los precios mínimos actuales para los siguientes 6 meses y, en el momento de ver los datos para un vuelo concreto, nos permite crear la alerta para recibir una notificación en caso que el precio mínimo cambiara o ir a la dirección web del vuelo para la realización de su compra.

A parte de realizar búsquedas y crear alertas para la notificación del cambio de precio, la aplicación no dispone de opciones adicionales como sucedía con Vueling.

Como ya nos encontrábamos en Vueling, las opciones del menú son accesibles mediante un desplazamiento desde la parte izquierda de la pantalla o bien desde el botón con icono de 3 barras horizontales situado en la parte superior izquierda.

2.3. easyJet

easyJet es la aplicación de la compañía easyJet, que permite, como Vueling, realizar búsquedas de vuelos, hoteles y coches de alquiler.

Como ya nos habíamos encontrado en Vueling, se puede utilizar la aplicación con una cuenta o de manera anónima, teniendo en cuenta que, si la usamos de manera anónima, a la hora de realizar algún pago, se nos solicitaran los datos personales y económicos para poder tramitarlo.

A la hora de realizar búsquedas, podemos distinguir entre los vuelos de ida y de ida y vuelta, mientras que mantiene como parámetro compartido en ambos el número de pasajeros, entre los cuales tenemos la opción de adultos y al lado de esta la de niños y bebés (esta segunda opción abre una ventana para introducir el número de cada tipo). Añade como opción extra el tipo de pago en los resultados, a distinguir entre tarjeta de crédito o débito.

Una vez realizada la búsqueda, easyJet distingue entre dos tipos de reserva, en calidad creciente:

- Standard
- Flexi

Los resultados de búsqueda aparecen ordenados por días en columna y para cada columna, los diferentes vuelos disponibles.

Una vez seleccionado el día y la franja horaria en la cual queremos el vuelo, podemos proceder a realizar la reserva entre los tipos disponibles.

Otra opción disponible en el menú es la de *Mis vuelos*, la cual necesita de una sesión iniciada para poderse utilizar i nos permite hacer el check in de los vuelos y ver su estado.

Por último, disponemos de la búsqueda de hoteles o de coches de alquiler si quisiéramos alguno de estos servicios.

2.4. Conclusiones

Las conclusiones a las que podemos llegar, una vez analizadas las aplicaciones anteriores son:

La mayoría de aplicaciones actualmente disponen de un menú lateral, en lugar de utilizar *tabs* o una página de inicio con botones para acceder a las diferentes secciones, ya que eso facilita la navegación en la aplicación.

Otro aspecto a destacar es que cada una de las aplicaciones tiene su propia manera de presentar los datos, lo cual nos invita a pararnos a pensar y diseñar una manera sencilla y eficaz para que el usuario pueda entender rápidamente los datos que le presentemos.

Aparte de lo mencionado previamente, las aplicaciones comparten bastantes similitudes.

Ahora que ya hemos analizado las tendencias de mercado, ya estamos listos para pasar a la fase de diseño del prototipo.

3. Diseño e implementación

En este capítulo se explicará cual ha sido la versión de Android escogida para el desarrollo de la aplicación y por qué, la metodología de trabajo seguida para el desarrollo de la misma, la arquitectura de clases generada y la planificación del proyecto.

3.1. Android y las versiones de la plataforma

A la hora de diseñar una aplicación para Android, uno de los problemas que nos encontramos es elegir la versión para la cual desarrollamos la aplicación.

¿Por qué? Porqué al crear una nueva aplicación, se nos piden dos tipos de versiones:

- La versión mínima (Minimum SDK Target) para la cual desarrollamos. Esta será la versión mínima necesaria de Android que deberá tener el dispositivo donde queramos utilizar la aplicación para asegurar su completo funcionamiento. Toda versión superior a esta podrá ejecutar la aplicación. Esta versión es vital analizarla, ya que de ello dependerán directamente el número de usuarios potenciales.
- La versión objetivo (SDK Target), es decir, para que versión está pensada la aplicación. Esta versión no restringe el uso de la aplicación y, como norma general, se utiliza la última versión estable disponible.

En el momento de desarrollar este proyecto, el dispositivo con el que contábamos para hacer pruebas tenía instalada la versión 5.1 Lollipop (SDK 22), motivo por el cual la aplicación fue desarrollada con este Target SDK.

En la siguiente imagen [Figura 1] podemos observar el porcentaje de versiones actuales en el mercado. Como podemos apreciar, nuestro público ocupa aproximadamente un 42% de la cuota actual de mercado.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%

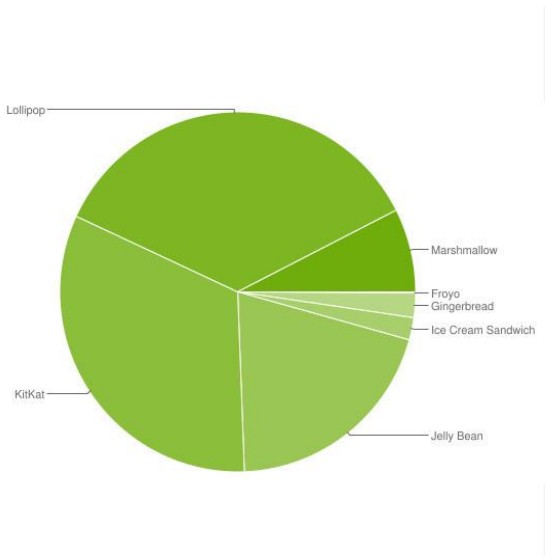

Figura 1. Versiones de Android en la actualidad, no se muestran las versiones con menos de un 0,1%. Fuente: Android Developers.

3.2. Metodología de trabajo

Para el desarrollo de nuestro proyecto, se ha utilizado una aproximación sencilla de la metodología de trabajo Scrum.

El Scrum consiste en realizar iteraciones (llamadas sprints) en las cuales se obtiene un producto pequeño pero funcional que va creciendo a cada iteración.

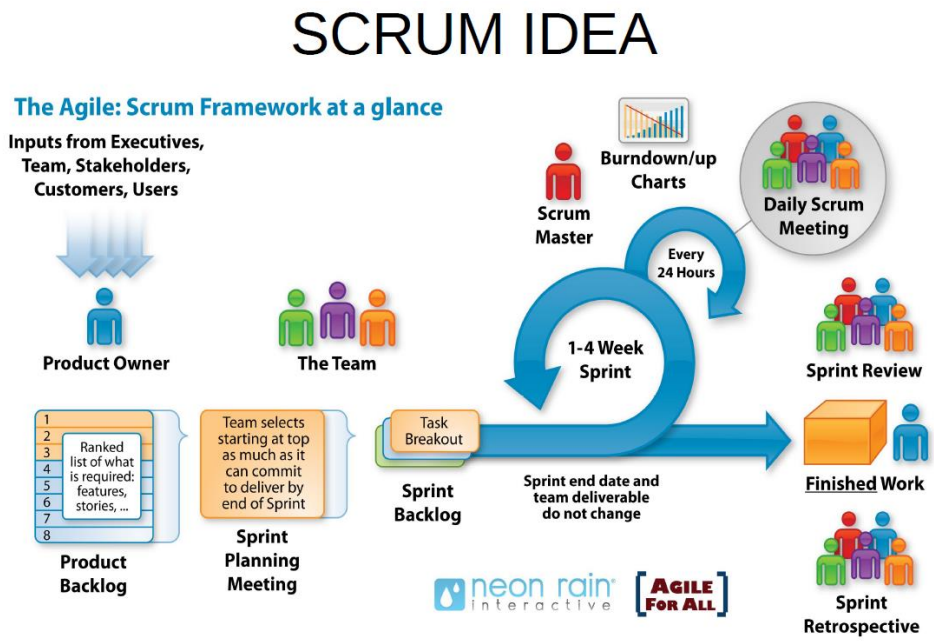

Figura 2. Gráfico de la metodología Scrum. Fuente: Transparencias de ESW, Universitat de Barcelona.

En nuestro caso, no obstante, la metodología de trabajo ha consistido en:

1. Realizar una pequeña reunión con el Director del proyecto y decidir que funcionalidades se implementarán de cara a la siguiente reunión.

2. Implementación de las funcionalidades acordadas con el Director del proyecto.

3. A la semana o dos semanas de la primera reunión, realizar una segunda reunión para observar los resultados de la implementación. Si se consideraban correctamente implementadas las funcionalidades que se acordaron, repetir el paso 1. En caso de no considerar correctamente, realizar el paso 2 con dichas funcionalidades.

Como podemos observar, guarda cierto parecido con la estructura cíclica de Scrum, pero no hemos llevado a cabo un modelo tan complejo, ya que, al ser un equipo pequeño, no nos hubiera resultado eficiente.

3.3 Diseño

Una vez analizadas las tendencias de mercado [sección: 2], es hora de realizar un prototipo de la aplicación.

Para ello se llevaron a cabo el diseño a papel de las pantallas principales de la aplicación:

- Búsqueda/Home: página principal de la aplicación una vez el usuario ha iniciado sesión, donde el usuario puede introducir los parámetros para realizar búsquedas.

- Resultados: página que aparece tras realizar una búsqueda, con los resultados de la misma.

- Seguimiento: página que muestra todos aquellos vuelos para los cuales el usuario ha creado una alerta.

- Log in: primera pantalla que verá el usuario al iniciar la aplicación, a través de esta, el usuario podrá iniciar sesión (vía mail + contraseña o Google) o acceder al formulario de registro (pantalla aparte).

En la siguiente imagen [Figura 3] podemos apreciar el prototipado de las mismas y su resultado final (en el mismo orden que se han nombrado).

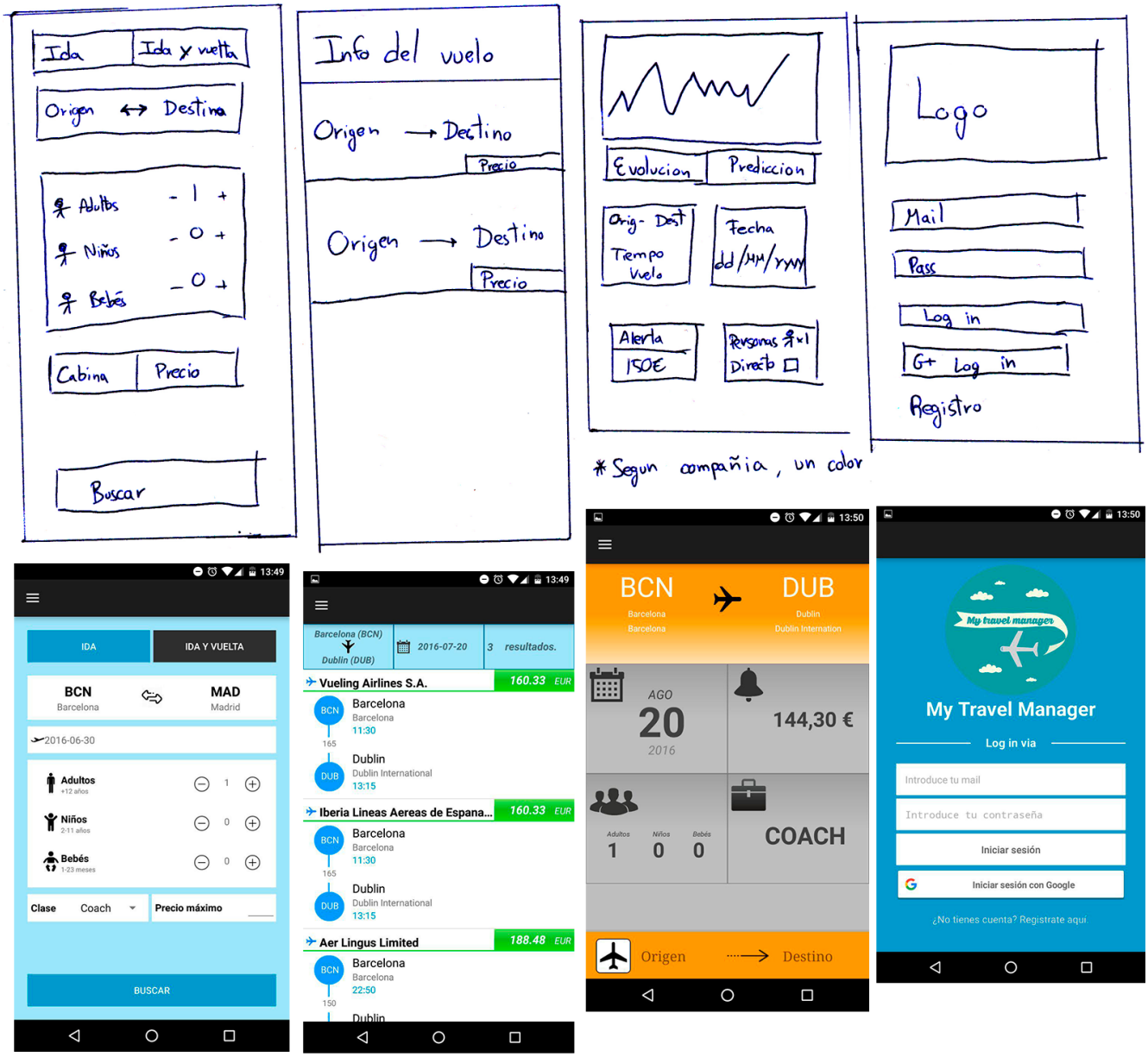

Figura 3. Prototipo de las pantallas principales de la aplicación y su resultado final en la aplicación.

Ahora ya tenemos una idea de cómo serán las pantallas principales de nuestra aplicación, pero falta definir como estarán conectadas entre sí, para ello vamos a ver el Flowchart de la aplicación [sección: 3.3.1].

3.3.1 Flowchart

Un diagrama de flujo (Flowchart) es una representación visual de la secuencia de pasos y decisiones necesarias para llevar a cabo un proceso. Cada paso en la secuencia se observa dentro de una forma de diagrama. Los pasos están unidos por líneas de conexión y flechas direccionales. Esto permite que cualquiera pueda ver el diagrama de flujo y lógicamente seguir el proceso de principio a fin.

Si bien la estructura del diseño es importante a la hora del desarrollo de una aplicación, hay que tener también muy en cuenta como conectamos las distintas pantallas de manera que sea cómoda la navegación entre ellas.

Para facilitar la navegación al usuario, se decidió implementar un sistema de menú conocido como Navigation Drawer. Un Navigation Drawer consiste en un menú desplegable lateral, normalmente situado a la izquierda de la aplicación al que se puede acceder deslizando el dedo de izquierda a derecha de la pantalla, o pulsando un icono con 3 barras horizontales.

Este tipo de menú facilita en gran medida la navegación por la aplicación, ya que nos da un acceso directo a la sección deseada desde cualquier lugar de la misma.

Un ejemplo de alguna aplicación muy utilizada que implementa este menú es YouTube, que lo combina con el uso de Tabs.

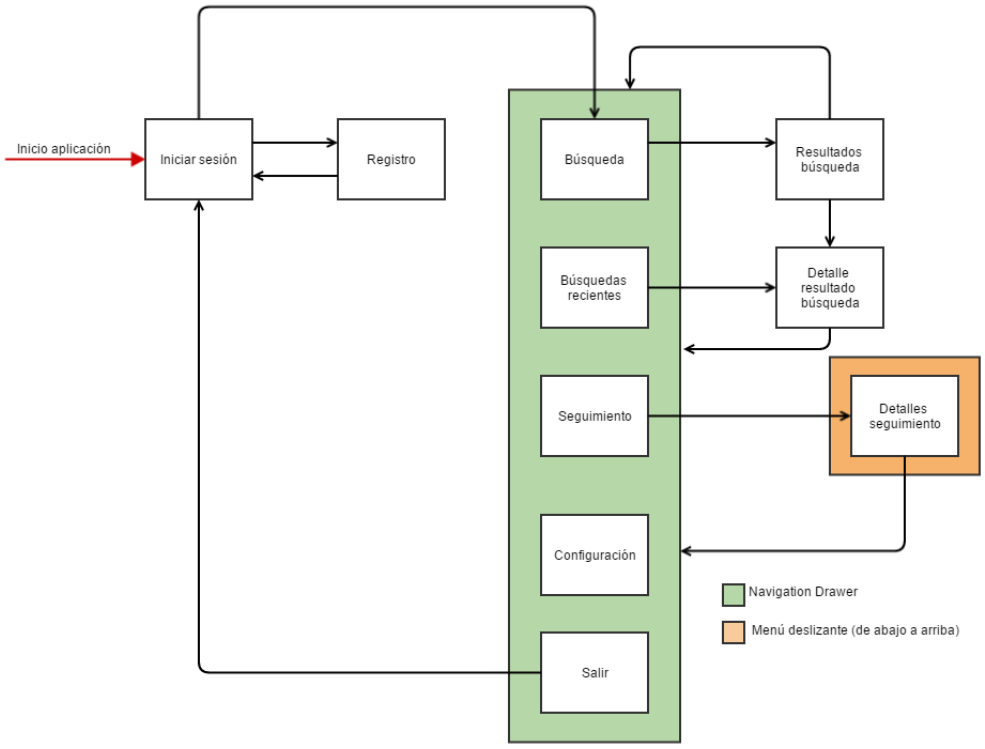

Figura 4. Flowchart de las pantallas de la aplicación.

En la imagen anterior [Figura 4] podemos observar las conexiones entre las diferentes pantallas de la aplicación, viendo que como máximo hay 4 acciones entre las pantallas más lejanas.
Al iniciar la aplicación, el usuario entrará en la pantalla de *Iniciar sesión*, mediante la cual podrá iniciar sesión directamente si ya dispone de cuenta creada por mail, o acceder mediante el acceso por Google.

Si el usuario no dispone de un perfil ya creado, o no puede/desea utilizar Google para autentificarse, desde un enlace disponible abajo podrá acceder a la pantalla de *Registro*.
En dicha pantalla, introducirá sus datos básicos que le crearan su perfil de usuario. Una vez creado el perfil, la aplicación le devolverá automáticamente a la pantalla de *Iniciar sesión*.

Cuando el usuario haya iniciado sesión, se encontrará con la pantalla de *Búsqueda*. Esta y todas las pantallas dentro del recuadro con color verde [Figura 4], son accesibles mediante el Navigation Drawer del que se ha hablado anteriormente.

Tanto realizando una búsqueda como yendo a la sección de *Búsquedas recientes* el usuario podrá acceder al detalle de un vuelo (*Detalle resultado búsqueda*). Desde dicha pantalla, el usuario podrá crear alertas para el vuelo y utilizará el menú deslizante para ir a la sección que desee posteriormente.

En *Seguimiento*, el usuario podrá ver los detalles de los vuelos para los cuales haya creado una alerta. Para cambiar los detalles del vuelo que se está visualizando se dispone de un menú deslizante que muestra la lista de vuelos que son del interés del usuario (deslizando desde debajo de la pantalla hacia arriba aparece la lista de vuelos, seleccionando uno de ellos el contenido de *Seguimiento* cambia y se oculta la lista). Para cambiar de sección, tal y como ocurría antes, el usuario dispone del menú deslizante lateral.

En *Configuración* el usuario podrá cambiar el idioma y seleccionar como desea las notificaciones.

Finalmente, mediante la opción *Salir* el usuario cerrará su sesión y será devuelto a la pantalla de *Iniciar sesión*.

3.4 Diagrama de Clases

Un diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, métodos y las relaciones entre los objetos.

En este proyecto, distinguiremos 3 tipos de clases:

- Clases de interfaz de usuario: son clases destinadas especialmente a crear algún componente visual para mostrar al usuario. En el diagrama se distinguen por el color naranja.
- Clases de funcionalidad o control: son clases destinadas a la realización de alguna funcionalidad específica de la aplicación. A diferencia de las clases de interfaz de usuario, estas están únicamente destinadas a realizar alguna función o actuar como soporte. En el diagrama se distinguen por el color verde.
- Clases de datos: son clases destinadas a la representación de objetos utilizados por la aplicación. Se utilizan principalmente para recuperar datos del servidor [sección 5.2] o como soporte hacia las clases de interfaz para poder representar correctamente listas u otros componentes. En el diagrama se distinguen por el color azul.

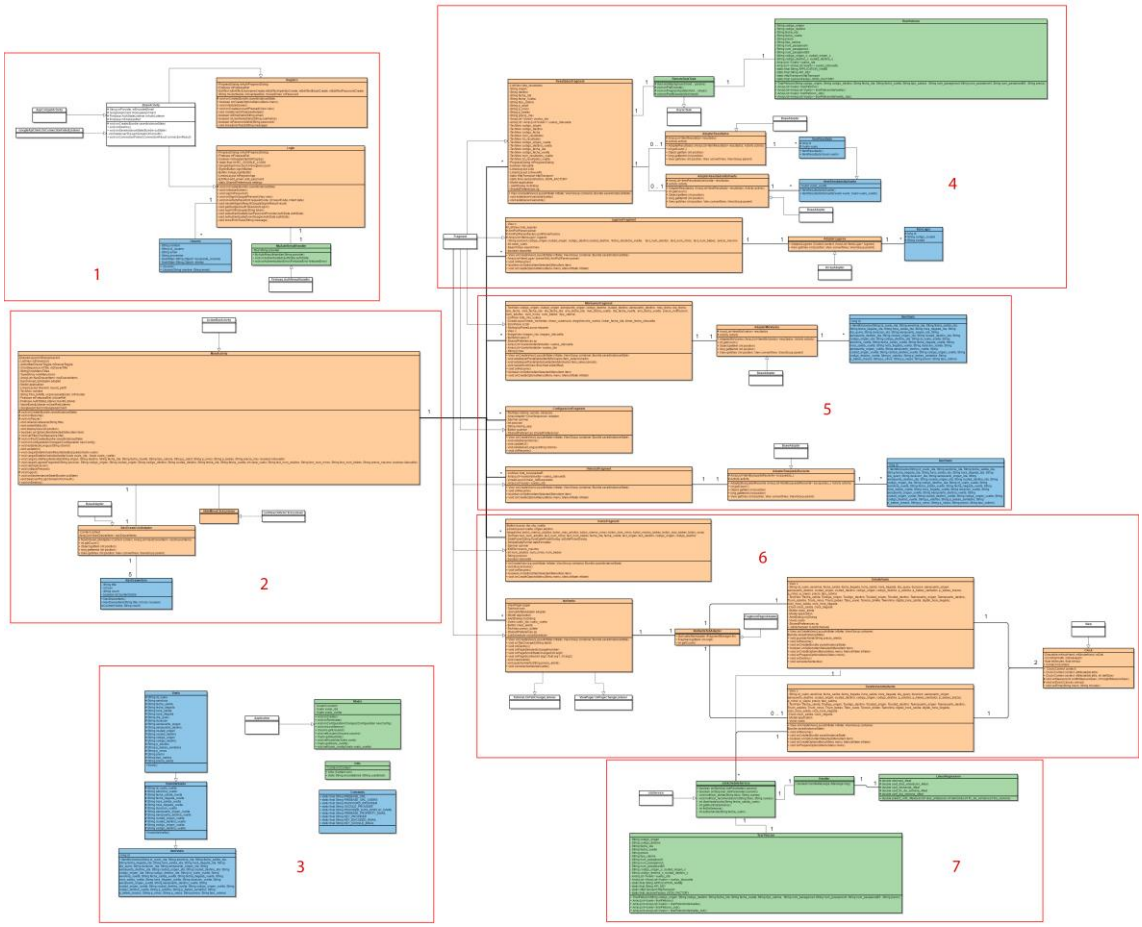

Figura 5. Diagrama de clases de la aplicación.

A continuación, se introducirán algunos conceptos básicos de componentes en Android.

Activities y Fragments: una Activity en una aplicación Android es una pantalla mediante la cual el usuario puede interactuar para realizar algo. Un Fragment es el equivalente a una Activity, pero sin la necesidad de ocupar toda la pantalla, es decir, puede ocupar solamente un fragmento de esta.

Adapters: un Adapter actúa como Puente entre una vista y los datos para dicha vista. El Adapter permite el acceso a los datos de la vista y también es responsable de crear una vista para cada objeto en el conjunto de datos.

View: una View ocupa un área rectangular en la pantalla y es responsable del dibujo y la gestión de eventos. View es la clase base para widgets, los cuales son utilizados para crear componentes interactivos para la IU.

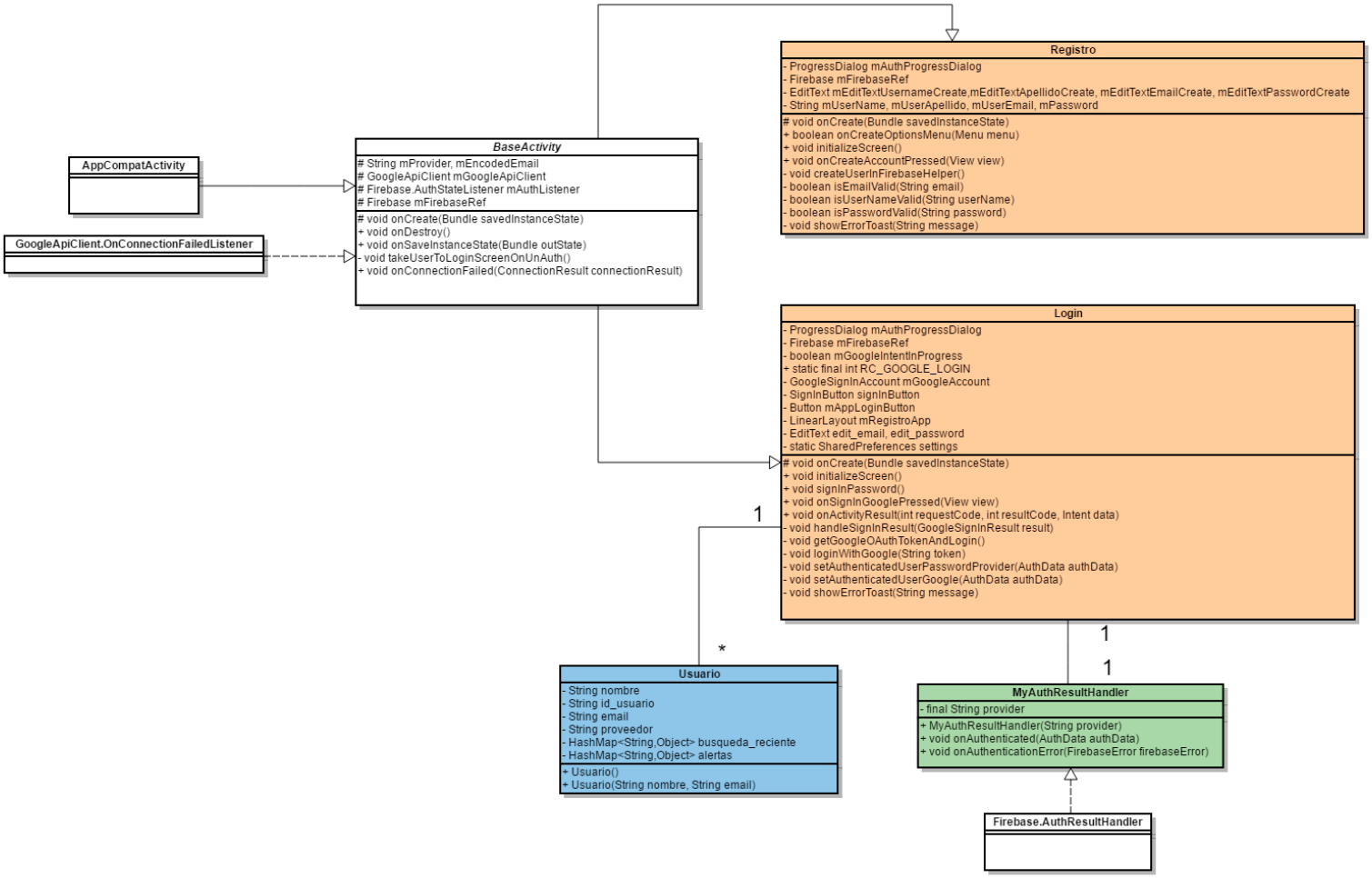

Figura 6. Diagrama de clases de la aplicación (1).

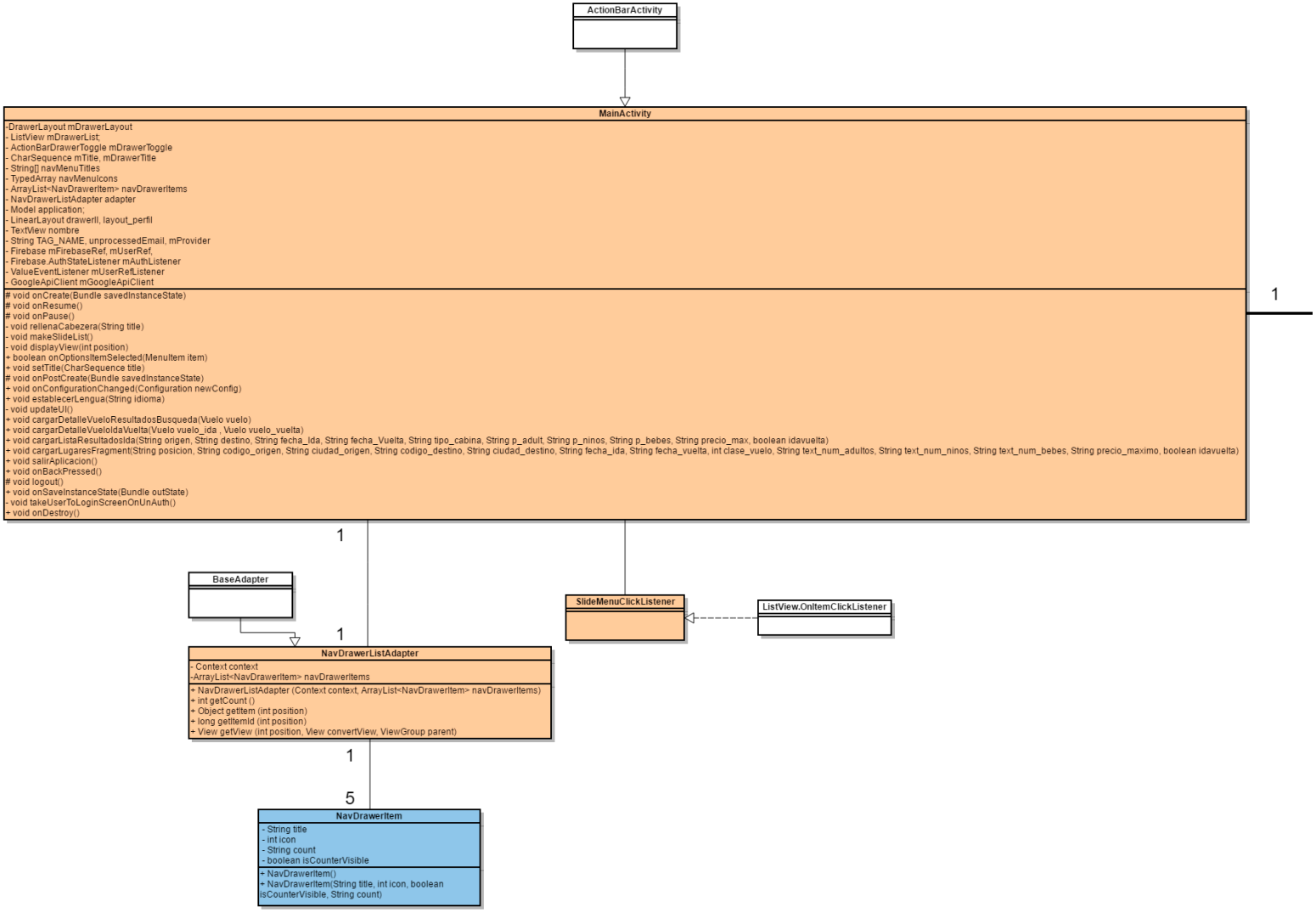

Figura 7. Diagrama de clases de la aplicación (2).

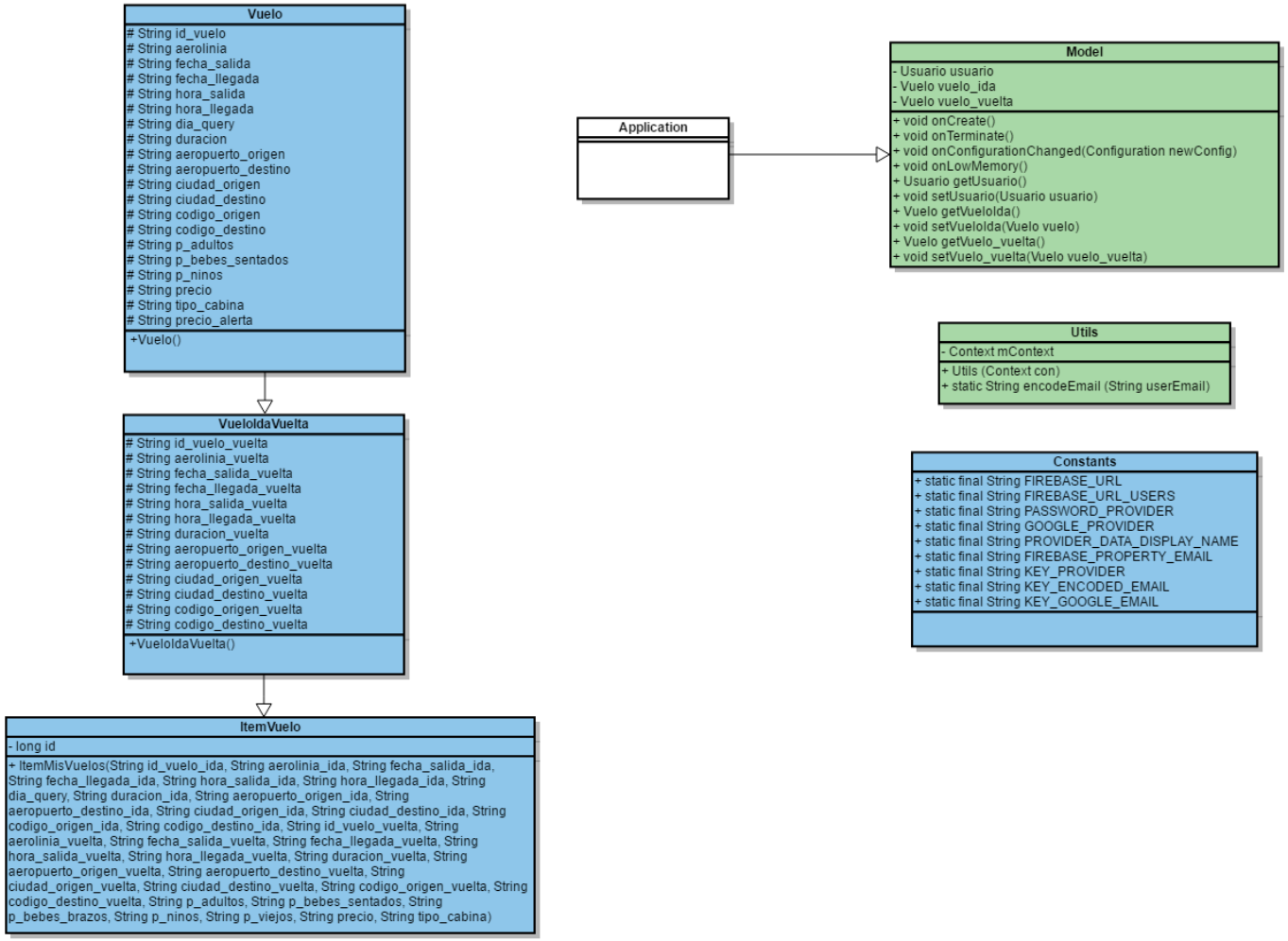

Figura 8. Diagrama de clases de la aplicación (3).

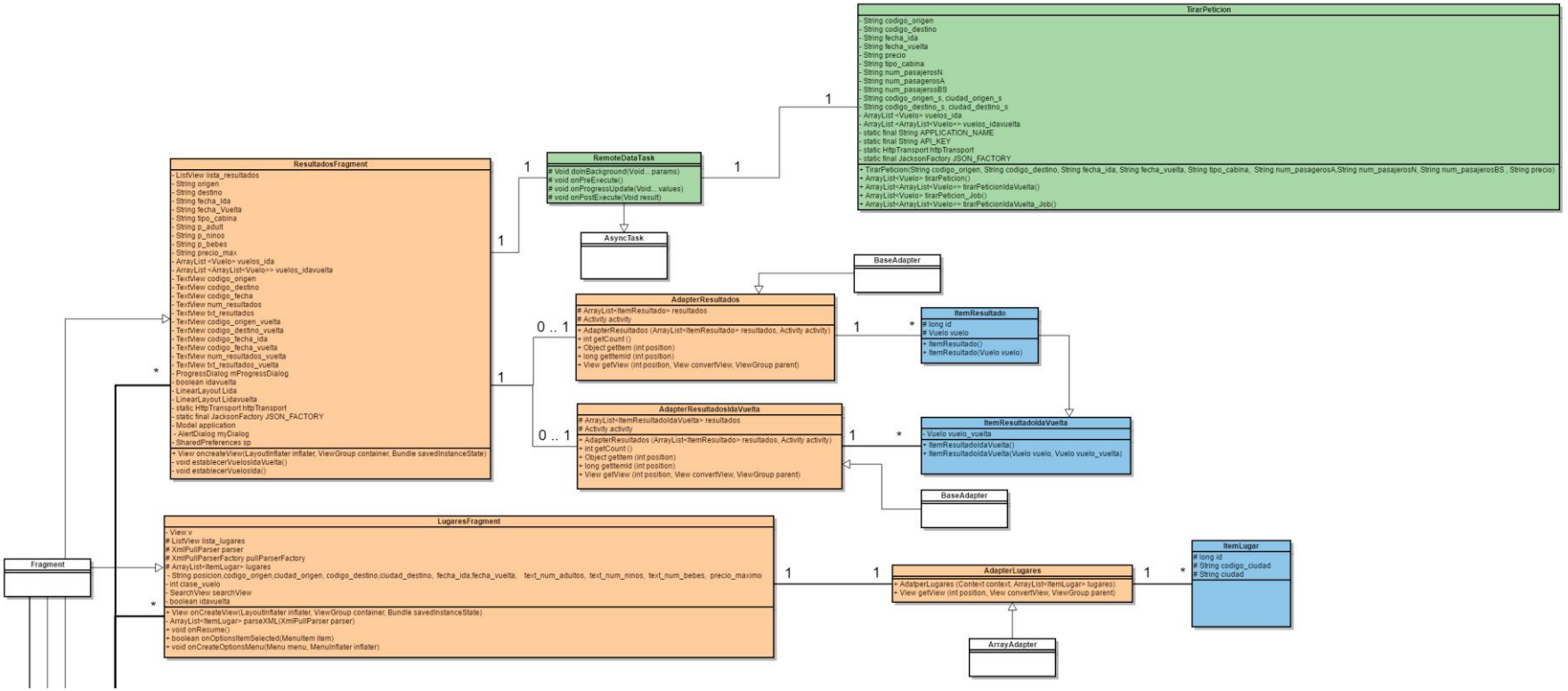

Figura 9. Diagrama de clases de la aplicación (4).

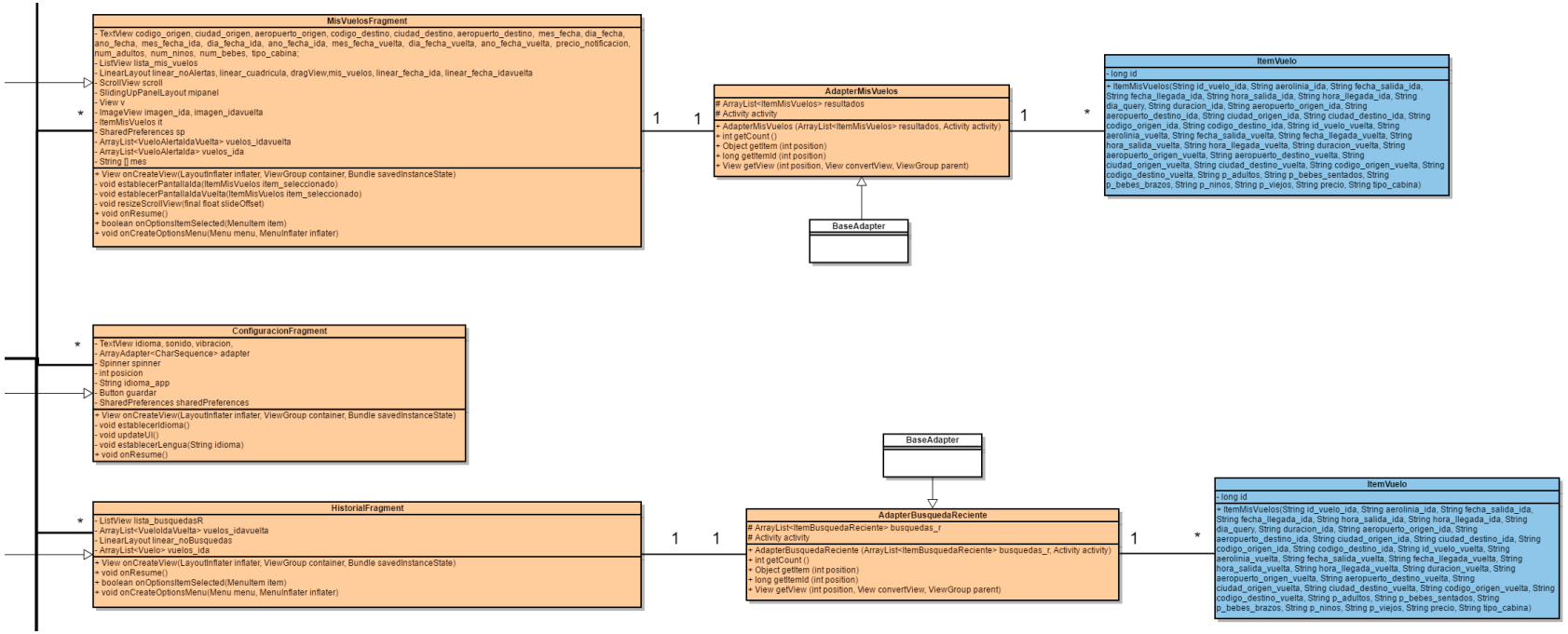

Figura 10. Diagrama de clases de la aplicación (5).

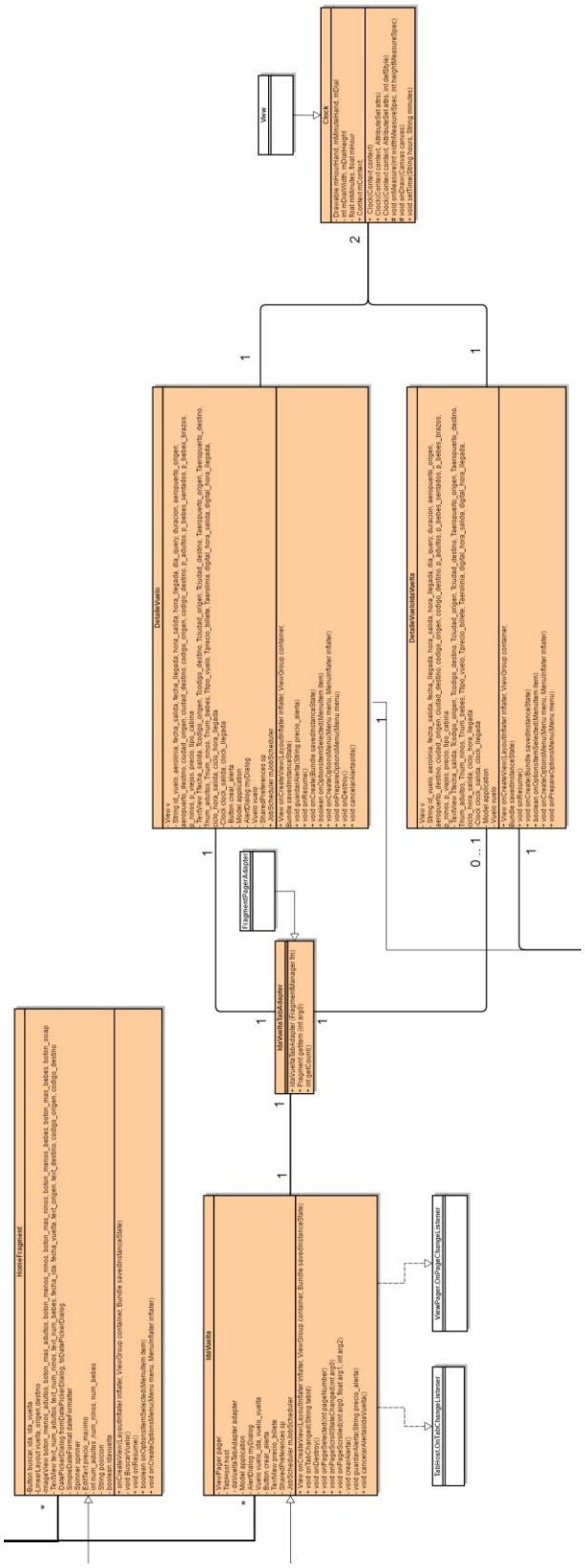

Figura 11. Diagrama de clases de la aplicación (6).

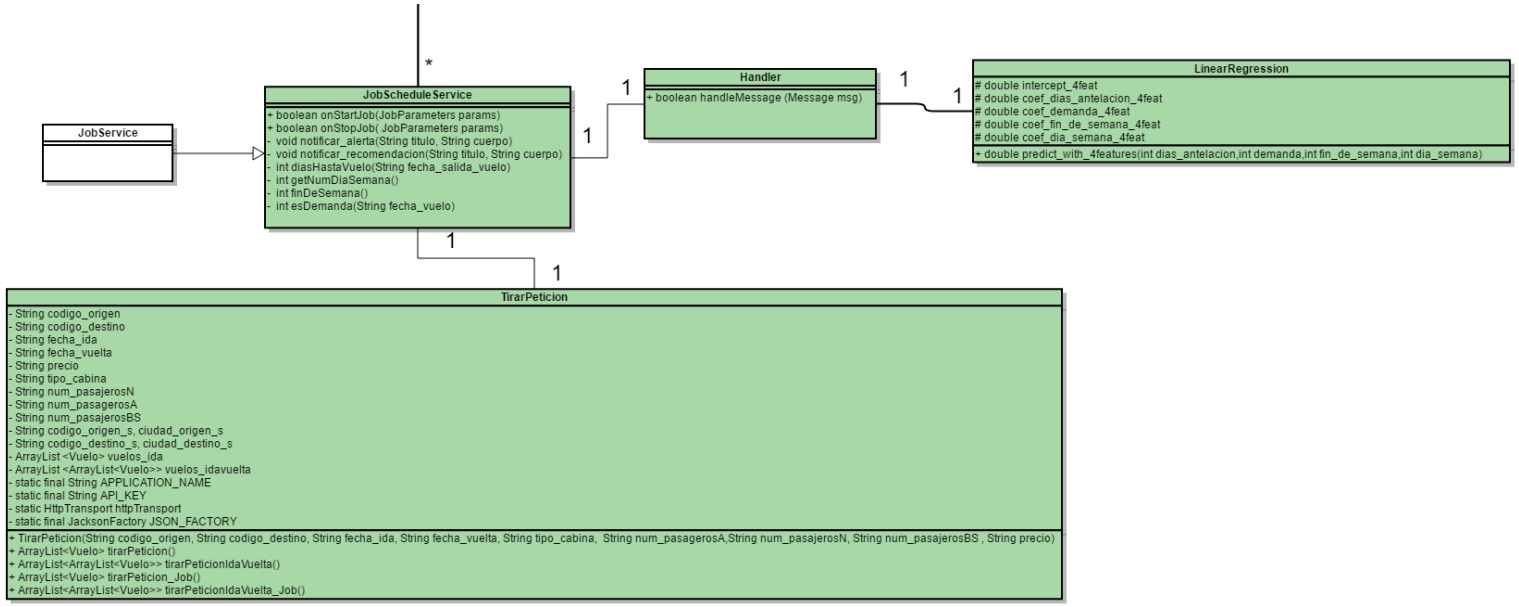

Figura 12. Diagrama de clases de la aplicación (7).

3.5 Planificación

Las diferentes tareas del proyecto se dividen de la siguiente manera:

- Diseño y prototipo
- Implementación de la interficie de usuario
- Formación y adquisición de conocimientos
- Análisis de datos necesarios
- Configuración de la gestión de datos
- Implementación de las funcionalidades
- Documentación
- Testing de la aplicación
- Arreglo de bugs

En el siguiente apartado se mostrará cómo han sido repartidas estas tareas a lo largo del tiempo, mediante un diagrama de Gantt.

3.5.1 Diagrama de Gantt

Un diagrama de Gantt es la representación gráfica del tiempo que dedicamos a cada una de las tareas en un proyecto concreto, siendo especialmente útil para mostrar la relación que existe entre el tiempo dedicado a una tarea y la carga de trabajo que supone.

Si consideramos que para el desarrollo de un proyecto debemos efectuar una serie de tareas simultáneas o consecutivas, que podríamos denominar unidades mínimas de trabajo, el diagrama de Gantt nos permite mostrar este desarrollo en el tiempo de las tareas de forma gráfica. Es pues, un gráfico en el que se muestra la duración de un conjunto de actividades.

Cada tarea es representada por una línea en cada una de las filas del diagrama, mientras que las columnas representan el tiempo en distintas escalas (días, semanas, o meses del programa) dependiendo de la duración del proyecto. En cada una de las tareas la fecha de inicio y la fecha de finalización del proyecto corresponden al inicio y final (respectivamente) de la barra correspondiente a dicha actividad.

Se han generado dos diagramas, uno para la planificación inicial del proyecto [Figura 12] y otro para la planificación real [Figura 13].

Podemos observar que, aunque la planificación inicial seguía un esquema lineal, en la realidad ha habido que hacer adaptaciones, hecho que ha alterado notablemente la planificación inicial. Por ejemplo, se planificó el aprendizaje inicial para luego realizar una implementación total, pero se pudo observar que no era un método eficaz y por ello se repartió el aprendizaje en sesiones en función de la implementación a realizar.

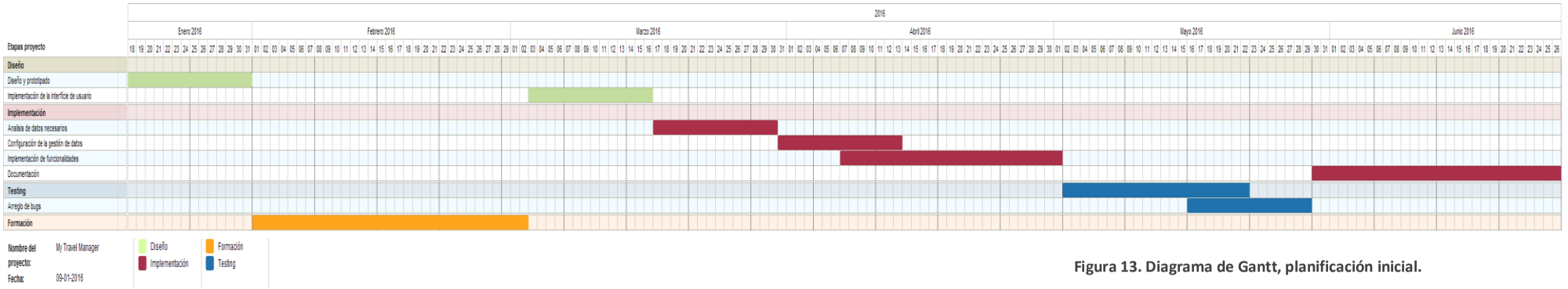

Figura 13. Diagrama de Gantt, planificación inicial.

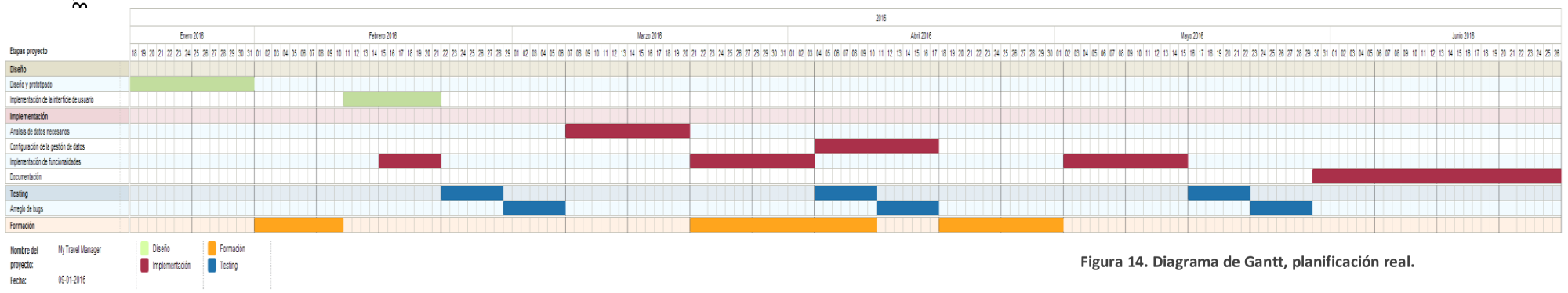

Figura 14. Diagrama de Gantt, planificación real.

Si tenemos en cuenta que el Trabajo de Fin de Grado equivale a 18 ECTS y que cada crédito equivale a 25 horas de trabajo obtenemos que todas las tareas deben realizarse en 450 horas.

Las horas dedicadas a cada tarea han sido las siguientes:

	Diseño y prototipo	Implementación de la IU	Análisis de requisitos	Configuración de la gestión de datos	Implementación	Documentación	Testing	Arreglo de bugs	Formación
Horas	30	50	30	30	100	50	10	50	100

Tabla 1. Horas de dedicación para cada tarea.

4. Aplicación

En esta sección se hablará de la implementación de las funcionalidades de la aplicación, así como de cómo han sido implementadas y el razonamiento seguido para determinar el método de implementación. Además, se analizarán cuáles son los datos necesarios para la aplicación y se explicaran los diferentes procesos de obtención y tratamiento de los mismos.

4.1 Casos de uso

Un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Para poder identificar los diferentes casos de uso de nuestra aplicación, recordemos cuáles son sus funcionalidades.

En la introducción, podemos obtener cuales serán: *crear una aplicación Android que se encargue de buscar y avisar al usuario de cuando comprar un billete de avión simplemente indicándole el día y el destino al que se desea viajar*.

Lo cual nos indican los siguientes casos de uso:

- Búsqueda de un vuelo
- Creación de una alerta para un vuelo
- Ver las búsquedas realizadas (Historial de búsquedas)
- Ver los vuelos que disponen de una alerta creada (Seguimiento)

Y como es evidente, cada usuario tendrá sus propios intereses, lo que nos conduce a que habrá que llevar un control por usuarios, dando lugar a los siguientes casos de uso:

- Registro en la aplicación por email
- Registro en la aplicación por Google
- Iniciar sesión en la aplicación
- Cerrar sesión en la aplicación

Y dado que la aplicación se encuentra disponible en 3 idiomas distintos, nos surge un último caso de uso:

- Cambiar el idioma de la aplicación

A continuación, detallaremos cada uno de los casos, por orden de sucesión de los mismos teniendo en cuenta que el usuario inicia la aplicación por primera vez y no dispone de cuenta.

CDU 1: Registro en la aplicación por email

Precondiciones: el usuario tiene instalada la aplicación y conexión a internet.
Resultado: el usuario obtiene una cuenta para poder utilizar la aplicación.

1. El usuario inicia por primera vez la aplicación y se encuentra con la pantalla de inicio de sesión, la cual incluye un enlace a un formulario de registro.

2. El usuario hace clic en dicho enlace.

3. El usuario es llevado a la pantalla con el formulario de registro, llena los campos solicitados y pulsa el botón de finalizar registro. Si los datos introducidos no son válidos el sistema le muestra en que campos hay un error. En caso contrario se crea la cuenta.

4. Una vez creada la cuenta el usuario es devuelto a la pantalla de inicio de sesión.

CDU 2: Registro en la aplicación por Google

Precondiciones: el usuario tiene instalada la aplicación y conexión a internet, además de una cuenta de Google.
Resultado: el usuario obtiene una cuenta para poder utilizar la aplicación.

1. El usuario inicia por primera vez la aplicación y se encuentra con la pantalla de inicio de sesión, la cual incluye un botón para iniciar sesión con Google.

2. El usuario hace clic en el botón y selecciona una de las cuentas de Google de su terminal.

3. El usuario obtiene acceso a la aplicación y es redirigido a la pantalla de búsqueda.

4. Éste método de registro identifica al usuario a la vez que le crea una cuenta.

CDU 3: Iniciar sesión en la aplicación

Precondiciones: el usuario dispone de cuenta en la aplicación y conexión a internet. El usuario se encuentra en la pantalla de inicio de sesión.
Resultado: el usuario se autentifica y es redirigido a la pantalla de búsqueda.

1. El usuario introduce su email y contraseña con los cuales creó la cuenta.

2. Seguidamente el usuario hace clic sobre el botón de Iniciar sesión.

3. Si el usuario y contraseña introducidos son correctos, el usuario inicia sesión y es redirigido a la pantalla de búsqueda.

CDU 4: Cerrar sesión en la aplicación
Precondiciones: el usuario tiene una sesión activa en la aplicación.
Resultados: el usuario cierra su sesión.

1. El usuario abre el menú desplegable deslizando el dedo de izquierda a derecha de la pantalla o haciendo clic en el icono de 3 barras horizontales situado en la parte superior izquierda de la misma.

2. Aparece el menú desplegable con las diferentes opciones de la aplicación.

3. El usuario selecciona la opción Salir del menú.

4. El usuario es redirigido a la pantalla de inicio de sesión y debe autentificarse para volver a acceder.

CDU 5: Búsqueda de un vuelo

Precondiciones: el usuario dispone de conexión a internet y está identificado en la aplicación. Además, se encuentra en la pantalla de búsqueda.

Resultado: el usuario obtiene una serie de vuelos que cumplen con los parámetros introducidos (si los hay).

1. El usuario selecciona si desea un vuelo de ida o de ida y vuelta mediante uno de los botones disponibles en la parte superior de la pantalla.

2. El usuario hace clic en el origen y la pantalla cambia a una con la lista de posibles ciudades de origen/destino. Mediante la barra de búsqueda, introduce la ciudad de origen y la selecciona. Al hacerlo es devuelto a la pantalla de búsqueda, donde la interfaz refleja el cambio en el origen seleccionado.

3. Repite el proceso anterior para seleccionar el destino.

4. El usuario hace clic en la fecha que aparece bajo el origen/destino y selecciona la fecha de salida del vuelo en el calendario emergente (en caso de ser vuelo de ida sólo verá una fecha). En caso de haber seleccionado ida y vuelta, en la fecha situada a la derecha seleccionará la fecha de salida del vuelo de vuelta, con otro calendario emergente.

5. A continuación, el usuario introducirá el número de pasajeros para el vuelo, haciendo clic en los botones con símbolos '-' y '+' situados a la derecha del tipo de pasajero (Adultos, niños o bebés).

6. Una vez introducidos los pasajeros, el usuario elegirá la clase en la que desea buscar el vuelo haciendo clic en el desplegable situado a la derecha de *Clase*.

7. Finalmente, el usuario introducirá el precio máximo que está dispuesto a pagar por el vuelo y hará clic en buscar. La pantalla cambiará a una lista con los vuelos que cumplan las condiciones introducidas por el usuario (si existen).

CDU 6: Ver las búsquedas realizadas (Historial de búsquedas)

Precondiciones: el usuario dispone de conexión a internet y está identificado en la aplicación. Además, el usuario ha realizado alguna búsqueda y ha visualizado alguno de los resultados.
Resultado: el usuario visualiza los vuelos para los que ha realizado una búsqueda y ha visualizado alguno de sus resultados.

1. El usuario abre el menú desplegable deslizando el dedo de izquierda a derecha de la pantalla o haciendo clic en el icono de 3 barras horizontales situado en la parte superior izquierda de la misma.

2. Aparece el menú desplegable con las diferentes opciones de la aplicación.

3. El usuario selecciona la opción Búsquedas recientes.

4. La pantalla cambia a una lista con los vuelos para los cuales el usuario ha visualizado los detalles cuando ha realizado dichas búsquedas.

CDU 7: Ver los vuelos que disponen de una alerta creada (Seguimiento)

Precondiciones: el usuario dispone de conexión a internet y está identificado en la aplicación. Además, se encuentra en la pantalla de búsqueda.
Resultado: el usuario obtiene una serie de vuelos que cumplen con los parámetros introducidos (si los hay).

1. El usuario abre el menú desplegable deslizando el dedo de izquierda a derecha de la pantalla o haciendo clic en el icono de 3 barras horizontales situado en la parte superior izquierda de la misma.

2. Aparece el menú desplegable con las diferentes opciones de la aplicación.

3. El usuario selecciona la opción Seguimiento.

4. La pantalla cambia a los detalles del último vuelo para el cuál se ha creado la alerta. Si el usuario desea ver los detalles de otro vuelo, debe abrir la lista de vuelos para los que ha creado una alerta deslizando el dedo desde la parte inferior de la pantalla hacia arriba.

5. Aparecerá la lista de vuelos para los cuales el usuario tiene una alerta creada. Cuando el usuario haga clic en uno de ellos, el contenido de la pantalla cambiará a los detalles de la alerta creada.

CDU 8: Creación de una alerta para un vuelo

Precondiciones: el usuario ha realizado la búsqueda de un vuelo y dispone de conexión a internet. El usuario se encuentra en la pantalla de resultados de búsqueda.

Resultados: se crea una alerta para el vuelo con las opciones introducidas por el usuario.

1. El usuario hace clic en el resultado de búsqueda para el cual quiere crear una alerta.

2. La pantalla cambiará y mostrará en detalle la información del vuelo.

3. El usuario hace clic en el botón Crear alerta.

4. Aparecerá una ventana emergente donde el usuario seleccionará el precio para el cual desea ser avisado (el precio aparecerá como un porcentaje del precio actual).

5. El usuario seleccionará dicho porcentaje y hará clic en el botón Aceptar.

6. El sistema notificará al usuario que la alerta ha sido creada y cuando el precio sea igual o menor al seleccionado, o el sistema de recomendación crea que es un momento oportuno de compra, éste enviará una notificación al terminal del usuario.

CDU 9: Cambiar el idioma de la aplicación

Precondiciones: el usuario ha iniciado sesión en la aplicación.

Resultados: el idioma de la aplicación cambia al nuevo idioma seleccionado por el usuario.

1. El usuario abre el menú desplegable deslizando el dedo de izquierda a derecha de la pantalla o haciendo clic en el icono de 3 barras horizontales situado en la parte superior izquierda de la misma.

2. Aparece el menú desplegable con las diferentes opciones de la aplicación.

3. El usuario selecciona Configuración y la pantalla le muestra las opciones de configuración.

4. El usuario selecciona uno de los idiomas disponibles en el desplegable Idiomas (Castellano, catalán o inglés) y hace clic en el botón Guardar.

5. La aplicación muestra ahora los textos en el nuevo idioma seleccionado.

4.2. Gestión de datos

En la sección anterior hemos analizado los diferentes casos de uso de la aplicación y hemos concluido que necesitamos tener un sistema de registro y de inicio de sesión para los usuarios. Además, nuestra aplicación necesita guardar la información sobre los vuelos que ha buscado el usuario y aquellos para los cuales tiene una alerta creada.

Para nuestro sistema de Log in, vamos a proceder con dos métodos:

- Log in por email: éste método consiste en introducir un usuario y contraseña. El sistema comprobará que el usuario y la contraseña introducidos estén registrados en la base de datos y, si lo están y son correctos, autentificará al usuario y le dará acceso a la aplicación. Éste método está pensado para los usuarios que no quieren vincular una cuenta ajena (Google, Facebook, etc…) para realizar un inicio de sesión.
- Log in utilizando Google: éste método consiste en autentificarse mediante una cuenta de Google. En el momento de la primera autentificación con la cuenta, se crea en la base de datos un usuario con el email de la cuenta utilizada, de manera que, si posteriormente otro usuario intentara crear una cuenta con ese email, le sería denegado.

Por lo tanto, del usuario necesitamos guardar:

- Nombre del usuario: para personalizar la aplicación, es conveniente que sepamos el nombre del usuario, para dirigirnos a él de una forma más amigable.
- Email del usuario: tanto si se registra por email como si lo hace por Google, necesitamos saber con qué email lo ha hecho, para poder comprobarlo posteriormente cuando inicie sesión.
- Contraseña: en el caso de haber creado una cuenta por email, necesitamos saber cuál es la contraseña que ha utilizado el usuario, para poder corroborar que es el propietario de la cuenta cuando inicie sesión.
- Vuelos en seguimiento: cada usuario dispondrá de una lista con los vuelos para los cuales haya creado alertas.
- Historial de vuelos: cada usuario dispondrá de una lista con los vuelos a los que haya accedido a ver sus detalles previamente, por si más adelante fueran de su interés.

Respecto a los vuelos, hemos de guardar:

- La compañía del vuelo.
- Los aeropuertos, tanto de origen como de destino del vuelo.
- Las ciudades de origen y destino del vuelo.
- Las fechas de salida y llegada del vuelo.
- El número y tipo de pasajeros.
- El precio de la búsqueda y el precio de alerta que configura el usuario.
- El tipo de cabina.
- La hora de salida y de llegada del vuelo, así como la duración del vuelo.
- La fecha en que se ha realizado la búsqueda del vuelo.

4.2.1. Base de Datos

Ahora que ya sabemos los datos que necesita nuestra aplicación, es momento de seleccionar y configurar una BDD para la aplicación, de manera que podamos guardar y consultar los datos analizados en la sección anterior.

Para este proyecto, se ha decidido implementar la BDD con Firebase. Firebase es una colección de servicios, que incluye bases de datos y que se puede utilizar como backend simple para aplicaciones.

Los motivos para elegir Firebase han sido:

- Los datos son accesibles a tiempo real, siempre que se disponga de internet. En caso de perder la conexión, el usuario puede trabajar sin ella y al recuperarla los cambios realizados se realizarán de manera automática.
- No hay que configurar un servidor para poder utilizarlo, es fácil de configurar y dispone de un SDK para Android, iOS y JavaScript. Ver posibles ampliaciones para más información [sección 6.1].
- Solo debemos escribir un poco de código en el lado del cliente. Firebase se encarga de la autentificación y acceso a los datos de forma automática, lo que nos ahorra tener que utilizar Threads y Services.

4.2.2. Configuración de la Base de Datos

Para poder utilizar Firebase en nuestra aplicación, primero debemos crear una cuenta en la página de Firebase (www.firebase.com) y posteriormente seguir los siguientes pasos:

1. **Instalar Firebase en la aplicación:** sólo hay que añadir la librería de Firebase en el archivo Gradle del proyecto.
2. **Añadir permisos de Internet al Manifest:** ya que para poder utilizar Firebase necesitamos de conexión a internet, hay que añadir este permiso al Manifest de la aplicación.
3. **Configurar Firebase en el método onCreate:** Firebase necesita saber el contexto de la aplicación Android para poder utilizarse. Para asegurarnos que se hace antes de utilizarlo, inicializamos Firebase en una clase que hereda de Application.
4. **Añadir una clase con las constantes (opcional):** cada BDD de Firebase tiene una URL única, por lo que resulta útil crear una clase de constantes para los diferentes accesos.

4.2.3. Estructura y funcionamiento de la BDD

En Firebase los datos son guardados en un árbol en lenguaje JSON (JavaScript Object Notation). Como ya se ha comentado en el apartado anterior, una BDD en Firebase consta de una URL raíz. Esta URL raíz es, a su vez, el nodo padre del árbol JSON.

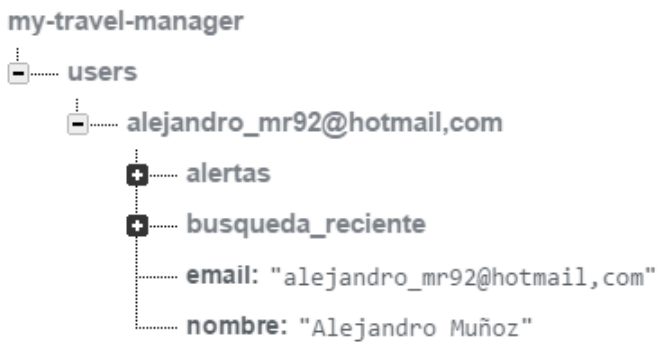

Figura 14. Estructura de la BDD en Firebase.

Nuestra BDD consta del nodo raíz, que tiene como hijo el nodo *users*, que a su vez tiene como hijos los usuarios de la aplicación. Cada usuario, tiene como atributos la lista de vuelos con alerta configurada, la lista de vuelos buscados recientemente, el email del usuario y su nombre. Los atributos de los vuelos son los vistos en la sección 4.2.

Cada nodo de nuestra BDD tiene una URL única que permite el acceso a los datos que contiene dicho nodo. Asimismo, un nodo consta de una pareja <key, value>, donde la key es siempre un String, y los valores pueden ser de los siguientes tipos:

- Booleanos.
- Longs.
- Doubles.
- Lists.
- Maps.
- Objetos (siempre y cuando contengan como atributos combinaciones de los tipos anteriores).

A la vez, si queremos guardar Objetos en Firebase, hay que cumplir 3 reglas:

1. Todas las instancias de la variable deben coincidir con los nombres clave de sus hijos y ser de un tipo válido (de la lista anterior).
2. La clase ha de tener un constructor vacío, incluso aunque no se utilice.
3. La clase ha de contener getters públicos para todas sus variables.

Para guardar datos en Firebase, basta con crear un objeto del tipo Firebase y pasarle como parámetro la URL de referencia, en nuestro caso https://my-travel-manager.firebaseio.com/.

A partir del objeto, podemos crear nuevos nodos, modificar los existentes o eliminarlos (más información en el anexo).

La parte interesante de Firebase viene en la lectura de datos. No utiliza el modelo Request-Response, con lo cual no ha de estar constantemente preguntando al servidor si hay nuevos datos. Esto nos lleva a un ahorro de batería para el terminal del usuario y a su vez, un ahorro en el ancho de banda del mismo.

Para saber cuándo se han añadido, modificado o eliminado datos en nuestra BDD, Firebase utiliza Listeners. Agregamos un Listener al nodo del cual queremos saber cuándo ha ocurrido algún evento en él o sus hijos y Firebase es el encargado de avisarnos cuando esto ocurra. Es en el momento del aviso, donde nosotros debemos ejecutar las acciones pertinentes (por ejemplo, actualizar la IU).

Cuando el Listener nos avisa, nos retorna un DataSnapshot, que consiste en una copia sólo de lectura del estado de la BDD en el momento del cambio. Esta copia incluye el nodo al cual hemos adjuntado el Listener y todos sus hijos.

4.2.4. Configuración y funcionamiento de la autentificación

Para poder autentificar usuarios con Firebase, hemos de realizar unas configuraciones previas para habilitar la funcionalidad.

Configurar la identificación con email y contraseña

Hemos de acceder al Dashboard de nuestra BDD de Firebase e ir a la sección *Login & Auth*, donde debemos habilitar la autentificación con email y password.

Cuando un usuario se registre, nos aparecerá en la parte inferior de dicha página.

Configurar la identificación con Google

Es un poco más compleja de configurar que la de email.

Primero, hay que habilitar el acceso utilizando Google de la misma forma que habíamos habilitado el acceso por email, en este caso haciéndolo para Google.

Posteriormente hay que acceder a Google Developers para obtener un archivo de configuración de Google Sign-In y añadir el mismo al proyecto.

Finalmente hemos e actualizar el Gradle y el Manifest de la aplicación.

Cuando un usuario se identifica en Firebase, se genera un token que es válido para la longitud de tiempo establecida en la configuración de la aplicación de Firebase (en nuestro caso, 30 días).

Para identificar al usuario, basta con realizar las llamadas a los métodos que Firebase nos proporciona y él se encarga de obtener el token y de autentificar al usuario.

Una vez identificado, Firebase nos retorna un objeto AuthData, que es un objeto que contiene datos básicos sobre el usuario.

4.3. Obtención de la información de los vuelos

Para obtener la información relevante a los vuelos se ha utilizado una API llamada QPX Express API. Esta API, desarrollada por Google, se utiliza para la búsqueda de vuelos aéreos y ofrece capacidades básicas de compra y facilidad en el acceso a la búsqueda.

La API forma parte del conjunto de API's de Google, disponibles para los desarrolladores mediante Google Developers Console, y está diseñada para obtener respuestas rápidas y con una alta fiabilidad. QPX Express realiza una búsqueda entre todas las combinaciones de vuelos y tarifas con los parámetros introducidos para dar respuesta a tiempo real.

La API se ejecuta mediante una petición POST en HTTP y tanto los parámetros de petición como la respuesta obtenida tienen formato JSON. Permite la parametrización del número de pasajeros y el tipo, el origen y destino del vuelo y sus correspondientes fechas entre otros parámetros adicionales.

Al obtener la respuesta, nos encargamos de realizar un parseo para obtener los datos y mostrárselos al usuario, así mismo como de actualizar la BDD si es necesario.

Una de las restricciones que aporta al proyecto el uso de esta API es la limitación a 50 peticiones diarias por API Key, lo cual restringe en gran medida el lanzamiento al mercado de la aplicación. Es por este motivo que a cada búsqueda hemos restringido el número de resultados posibles a 5 y que, cuando comprobamos el precio en el servicio de notificaciones, lo hacemos mediante 1 única búsqueda (los resultados son devueltos en orden creciente de precio, por lo que con una búsqueda garantizamos el precio más económico).

4.4. Notificaciones

Una de las funcionalidades más importantes de la aplicación es la de procesar de manera automática, cada cierto tiempo establecido, la comparación del precio de alerta introducido por el usuario con el precio actual del vuelo. De este modo, en caso de que el precio actual sea igual o inferior al que el usuario configuró, éste recibe una notificación en su terminal que le avisa que para ese vuelo hay un precio de su interés.

A la vez, el sistema introduce los datos obtenidos por la consulta automática y los pasa a una función que nos retorna si es un buen momento para realizar la compra, aunque el precio del billete no cumpla con el establecido por el usuario. Si éste determina que efectivamente es un buen momento para obtener el billete, también notifica al usuario.

El proceso anterior ocurre para cada vuelo que tenga una alerta configurada y las comprobaciones se realizan una vez al día. El proceso se inicia en el mismo instante que el usuario crea la alerta, realizando una comparación al momento. Esto es debido a que, a pesar que el precio no será inferior, es posible que sí sea un buen momento para comprar y, por lo tanto, haya que avisar de tal situación al usuario.

Para hacer el proceso se ha utilizado una API introducida en Android Lollipop: JobScheduler.
Esta API permite la ejecución de una tarea de manera iterada bajo unas circunstancias que nosotros definamos. En el caso de nuestra aplicación, la tarea se ejecutará si hay conexión a internet, cada 24 horas y aunque el dispositivo se reinicie o apague.

A diferencia de otros métodos, el utilizar esta API no nos asegura que el tiempo exacto de ejecución sean 24 horas exactas del momento en que se crea la alerta. A cambio de este pequeño desajuste de tiempo, éste método conserva mucho más la batería del terminal.

5. Testing

Para el testeo de las diversas funcionalidades de la aplicación se han utilizado los siguientes terminales:

- OnePlus One con CyanogenOS 12.1.1 basado en Android Lollipop
- Nexus 5X con Android 6.0.1 Marshmallow

Para realizar las pruebas, se han considerado los casos de uso descritos en la sección 4.1.

Se han realizado los siguientes test:

Test	Resultado
CDU 1: Crear usuario vía email	Ok
CDU 2: Crear usuario vía Google	Ok
CDU 5: Realizar la búsqueda de varios vuelos	Ok
CDU 8: Crear alertas para diferentes vuelos	Ok
Comprobar que el sistema envía notificaciones en caso de cumplir los requisitos del usuario	Ok
CDU 9: Cambiar el idioma de la aplicación	Ok
CDU 6: Comprobar que la lista de historial coincide con los vuelos vistos	Ok
CDU 7: Comprobar que en la lista de seguimiento aparecen los vuelos que tienen una alerta creada	Ok
CDU 4: Cerrar sesión y abrirla con otro usuario	Ok
Comprobar que en la base de datos se escribe la información debida y siguiendo la estructura detallada	Ok

Tabla 2. Test de funcionalidad realizados.

Para poder comprobar la fiabilidad de las notificaciones, se han forzado vía código las distintas situaciones que pueden ocurrir y se ha minimizado el intervalo de alerta.

Una vez comprobado que en estos casos la aplicación respondía como se esperaba, se ha procedido a testearlo de manera real, con alertas configuradas cada 24 horas.

Un ejemplo de resultado obtenido es el siguiente:

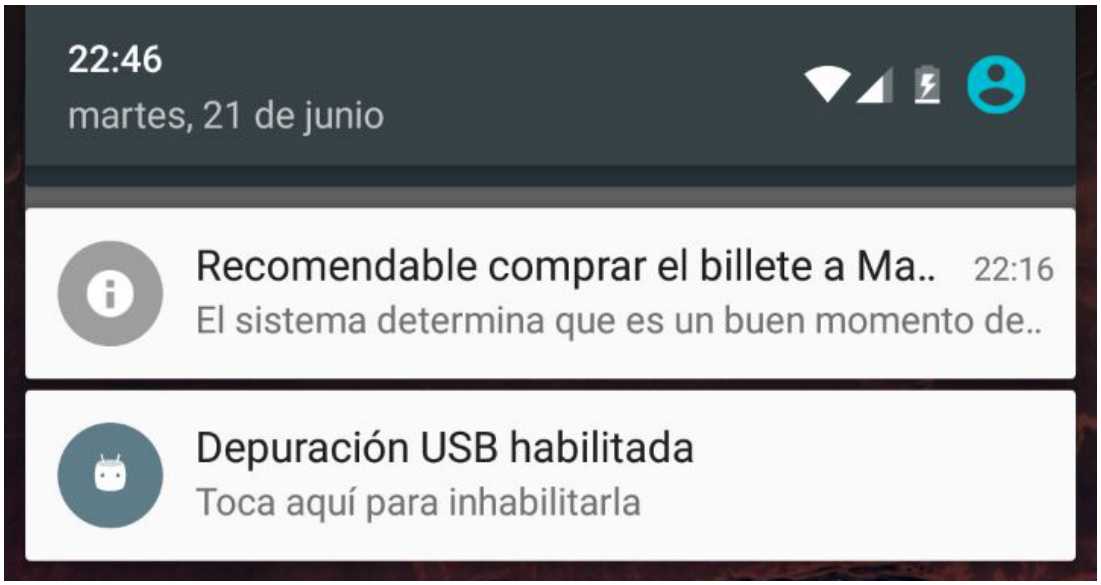

Figura 15. Ejemplo de notificación en Nexus 5X.

En la figura anterior podemos apreciar, como el sistema nos avisa que es recomendable comprar el billete a Madrid.

6. Herramientas utilizadas

Para el desarrollo del proyecto se han utilizado las siguientes herramientas:

Android Studio – versión 2.0
Android Studio es el IDE oficial para el desarrollo de aplicaciones Android basado en IntelliJ IDEA. El código fuente del proyecto ha sido desarrollado íntegramente en Android Studio.

Github
Github es una plataforma online que se utiliza para desarrollar software de manera colaborativa o para crear puntos de control entre las diversas fases del mismo. Gracias a Github se han podido ir creando puntos de guardado entre las diferentes implementaciones del proyecto y nos ha permitido tener el código disponible desde cualquier lugar.

Google Calendar
Google Calendar es el calendario online creado por Google. Se ha utilizado especialmente para la gestión de reuniones con el tutor del proyecto.

Microsoft OneNote
OneNote es un programa de la suite Office de Microsoft que permite la toma de notas en formato digital y el poder compartirlas con otros usuarios de manera rápida y eficaz.
Para el proyecto ha sido muy útil en los casos en los que había que obtener información de alguna de las páginas de desarrollo, ya que así se ha podido resumir su contenido para su uso posterior.

Gliffy
Gliffy es una aplicación web que permite la creación de distintos tipos de diagramas. Para el proyecto ha sido utilizado en la creación del Diagrama de Clases y el Flowchart.

Tomsplanner
Tomsplanner es una aplicación web que permite la creación de Diagramas de Gantt. Para el proyecto se ha utilizado para la creación de la planificación.

7. Conclusiones

Al inicio del proyecto, lo veía como una oportunidad de aprendizaje y pensaba que sería imposible de realizar. En las primeras semanas, realizar cualquier tarea se volvía todo un reto, ya que apenas tenía conocimientos sobre los diferentes componentes de Android o como utilizarlos.

Poco a poco, he ido aprendiendo algunos de los muchos componentes que tiene Android y como utilizarlos para conseguir el resultado que deseo. He aprendido a depurar y debugar el código y a crear una base de datos online con la que la aplicación interactúa. También he implementado un sistema de registro y autentificación de usuarios, aprendiendo a utilizar diferentes SDK y API's en el proceso.

Han sido muchas horas de autoaprendizaje, muchas horas de errores. Con cada error he aprendido algo nuevo, he aprendido a analizar detalladamente cual es el resultado que yo espero y cuál es la secuencia de código que me lleva a ello. Equivocarme ha sido lo mejor del proyecto, ya que gracias a ello he podido aprender.

El mayor logro de este proyecto es la satisfacción obtenida al haber creado por mí mismo una aplicación que tiene un uso real.

Si se hubiera dispuesto de más tiempo o un equipo de personas, se podrían haber implementado otras funcionalidades e incluso hacer la aplicación colaborativa entre usuarios.

8. Fuentes de información

Para realizar el proyecto se han consultado las siguientes fuentes de información:

- Android Developers: https://developer.android.com/index.html?hl=es
- Firebase: https://www.firebase.com/
- Google Developers: https://www.firebase.com/
- Stack Overflow: http://stackoverflow.com/