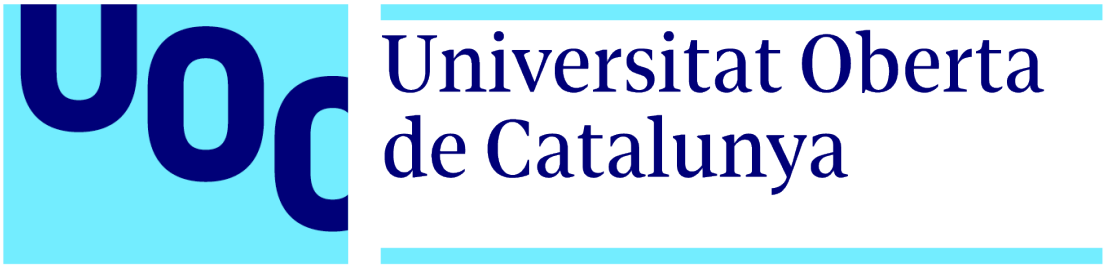

Desarrollo de la app *AvidReaders* para dispositivos iOS

Máster universitario en Desarrollo de aplicaciones para dispositivos móviles

Alumno: **Marcos García Rouco**
(marcosgr@uoc.edu)

Director: **Eduard Martín Lineros**
(emartinli@uoc.edu)

06/2020

FICHA DEL TRABAJO FINAL

Título del trabajo	Desarrollo de la app *AvidReaders* para dispositivos iOS
Nombre del autor	Marcos García Rouco
Nombre del consultor	Eduard Martín Lineros
Fecha de entrega (mm/aaaa)	06/2020
Titulación	*Máster universitario en Desarrollo de aplicaciones para dispositivos móviles*

Resumen del trabajo (máximo 250 palabras):

Para la realización de este trabajo de fin de máster se ha decidido desarrollar una aplicación móvil para dispositivos iOS que tendrá por nombre *AvidReaders*. Como su nombre indica, el público al que va dirigida esta app es aquella parte de la población a la que le entusiasma la lectura. De este modo, se pretende crear un software que permita a sus usuarios consultar información sobre libros, autores y géneros, así como hacer un seguimiento de sus hábitos de lectura y estimular esta mediante retos, recordatorios y estadísticas. Además, también se les permitirá definir una serie de etiquetas o *tags* para ver en la sección de noticias aquellas que estén relacionadas con el ámbito de la lectura y, más concretamente, con las etiquetas definidas por los usuarios. Todo esto se llevará a cabo para dispositivos iOS, tanto iPhones como iPads, desarrollando en el lenguaje de programación SwiftUI y no en su precursos Swift a fin de aprovechar este trabajo de fin de máster para aprender la nueva forma de desarrollar aplicaciones para iOS que se utilizará en los años venideros. Cabe destacar que el *target* o dispositivos objetivo de SwiftUI son aquellos con versión iOS 13.0 o superior, la cual salió a finales de 2019 y a día 27 de enero de 2020 el 77 % de usuarios de iPhones ya la tienen instalada [3], por lo que la app estaría dirigida a un gran porcentaje de los usuarios de iOS, un porcentaje que tendrá que ir creciendo paulatinamente.

Abstract (in English, 250 words or less):

For this master's degree's final project an iOS app named *AvidReaders* is going to created. As its name implies, the sought destinataries of this app will be the subset of the population that feels attracted by books and reading in general. Thus, the software to be developed will allow users to peruse information about books, authors and genres, as well as having their reading habits tracked and nudged in the right direction by means of challenges, reminders and statistics. Furthermore, they will be able to define tags so as to receive book news as a whole and, particularly, those related to their interests. All the previous will be developed for iOS devices, both iPhones and iPads, using SwiftUI as the programming language and not the prior version Swift in order to take advantage of this final thesis to learn the brand new way to develop new iOS apps. It must be noted that SwiftUI's target devices are those where iOS 13.0 or higher is installed, a version that was released in late 2019, and on January 27 2020 already 77 % of iPhone users had installed it [3]. On account of this, the app would be directed at a considerable percentage of iOS users, one which will have to grow in the months to come.

Palabras clave (entre 4 y 8):

lectura, retos, cultura

Índice

Índice de figuras

Índice de cuadros

1. Introducción

En esta primera sección de la memoria se describirán las necesidades que han hecho que surja el producto a realizar, así como los objetivos que se pretenden alcanzar al término del trabajo de fin de máster.

Así, en primer lugar se dará una descripción del contexto en que se desarrolla el proyecto, llevándonos a justificar el porqué de la aplicación. Acto seguido, se procederá a listar los objetivos que se pretende alcanzar con la realización del proyecto, los cuales serán descritos en mayor profundidad en forma de requisitos funcionales y no funcionales.

En un proyecto informático es de gran importancia saber a qué riesgos estamos expuestos, por ello la siguiente sección tratará la gestión de los riesgos, permitiendo identificarlos y especificar medidas para combatirlos.

Posteriormente se especificará la metodología a seguir para llevar a cabo el alcance de susodichos objetivos, es decir, se mencionará el ciclo de vida elegido. Tras esto, se procederá a presentar la planificación para el proyecto, la cual no estará completa sin un diagrama de Gantt que especifique las fases en las que se desglosará dicho proyecto, indicando subtareas e hitos marcados por ítemes a entregar.

Luego se procederá a realizar una gestión de las comunicaciones al identificar a los interesados en el proyecto (*stakeholders*) y los flujos de comunicación que habrá entre ellos.

Por último, se proporcionará una breve descripción de los ítemes finales, producto de la realización del proyecto.

1.1. Contexto y justificación del Trabajo

Hoy en día existe un sinfín de actividades de ocio para entretenernos: desde videojuegos o juegos de mesa a deportes al aire libre, pasando por ir al cine o quedar con nuestros amigos. No obstante, la lectura sigue siendo una de las actividades favoritas de muchas personas. Consecuentemente, existe la necesidad de crear aplicaciones para ayudar a estas personas a encontrar nuevos libros, comprobar si sus escritores favoritos han publicado algo nuevo o motivarlos a leer más, tal y como aplicaciones como Google Fit [1] han surgido para satisfacer la necesidad de cierta parte de la población de controlar su salud y actividad física mediante una aplicación móvil.

Para esta necesidad presentada ya existen aplicaciones que intentan satisfacerla, entre las cuales destaca Goodreads [2]. Esta es una aplicación extremadamente lograda, pero como todo, no es perfecta. En concreto, se considera que entre sus puntos más fuertes destacan la gran base de datos de libros y autores con la que cuenta o el hecho de que se permita escribir críticas o *reviews* sobre los libros. En lo que respecta a los puntos débiles de la aplicación, el no poder ver en ningún sitio noticias sobre nuestros libros y escritores favoritos (aparte de lo que estos últimos deciden publicar) constituye su

principal punto débil, así como la ausencia de una sección que nos permita ver nuestra progresión lectora, a menos en la versión móvil de la plataforma. También se echa en falta un apartado que le permita a los lectores realizar un seguimiento de su actividad de forma atractiva.

De este modo, lo que se intentará realizar a lo largo de este proyecto será la creación de una nueva app para lectores, ***AvidReaders***, la cual aparte de realizar las principales labores que lleva a cabo Goodreads (listar libros y autores y permitir consultarlos y filtrarlos), permitirá a los usuarios ver estadísticas de su actividad lectora, hacer un seguimiento eficaz de dicha actividad mediante la visualización de gráficos o asumir ciertos retos que les motiven a leer más. Además, también se incluirá en la aplicación una sección de noticias sobre el mundo de la lectura que podrán ser personalizadas por los usuarios al definir preferencias en forma de *tags* o etiquetas. De esta forma, lo que se pretende es echar mano de algunas de las funcionalidades interesantes de Goodreads, a las cuales se les añadirá algunas otras con un toque personal y destinadas a animar a los usuarios a leer más e interactuar con la aplicación.

La anterior idea se ejecutará para dispositivos iOS, tanto iPhones como iPads, desarrollando en el lenguaje de programación SwiftUI y no en su precursor Swift a fin de aprovechar este trabajo de fin de máster para aprender la nueva forma de desarrollar aplicaciones para iOS que se utilizará en los años venideros. Cabe destacar que el *target* o dispositivos objetivo de SwiftUI son aquellos con versión iOS 13.0 o superior, la cual salió a finales de 2019 y a día 27 de enero de 2020 el 77 % de usuarios de iPhones ya la tienen instalada [3], por lo que la app estaría dirigida a un gran porcentaje de los usuarios de iOS, un porcentaje que tendrá que ir creciendo paulatinamente.

1.2. Objetivos del Trabajo

En este apartado se listarán los principales objetivos del proyecto. Estos ítemes será los que habrá que realizar para dar el proyecto por concluido. En caso de que no pueda ser así, se deberá dar una explicación justificada, pues no se le estaría entregando al cliente todo lo que se había acordado a la hora de definir el alcance.

A continuación se muestran los objetivos del trabajo a alto nivel:

- Mostrar información de libros obtenida de APIs de terceros.
- Mostrar información de escritores obtenida de APIs de terceros.
- Animar a los usuarios a leer más mediante el empleo de recordatorios y retos.
- Mostrar noticias relacionadas con el mundo de la lectura.
- Permitir a los usuarios registrarse e iniciar sesión en la aplicación.
- Presentar gráficos y medidas estadísticas para hacer un seguimiento de la actividad lectora de los usuarios.
- Obtener una oportunidad para que el alumno aprenda SwiftUI.

- Crear una aplicación móvil atractiva visualmente y altamente usable.

Los anteriores objetivos se pueden desglosar a modo de requisitos funcionales (aquellos que se traducen directamente en una funcionalidad de la aplicación) y requisitos no funcionales (aquellos otros que representan restricciones de la aplicación pero que se no se ven reflejados en la aplicación en forma de acciones que puedan desempeñar los usuarios). En las siguientes secciones se listaran los requisitos de ambos tipos, permitiéndonos así hacer una definición más precisa de los objetivos, es decir, concretar con mayor exactitud el alcance del proyecto.

No obstante, antes de listar los requisitos se va a proporcionar una escala de importancia, lo cual nos permitirá determinar cuáles son los requisitos esenciales de la aplicación y cuáles son aquellos que, en el peor de los casos, pueden resultar prescindibles. Dicha escala es la especificada en el cuadro 1.

Importancia	Descripción
Vital	Es de extrema importancia que el requisito se cumpla para que el proyecto termine de forma satisfactoria.
Importante	Si el requisito se cumple supondría un aumento del valor del proyecto, pero en el caso de que no se pudiese cumplir, las pérdidas ocasionadas no serían fatales.
Estimulante	No es necesario llevar a cabo este requisito para que el proyecto finalice satisfactoriamente. No obstante, el hecho de que se realizase incrementaría la calidad del proyecto.

Cuadro 1: Escala de importancia de los requisitos

Como se podrá observar más adelante, muchos de los requisitos funcionales y no funcionales se marcarán como estimulantes o importantes debido a dos motivos:

1. No representan funcionalidades o restricciones que afecten directamente a la satisfacción de los objetivos definidos.
2. Debido a las restricciones temporales que se detallarán en la sección 1.5 y al desconocimiento bien del dominio o bien de partes de las tecnologías a emplear, no se tiene claro que vaya a ser posible realizarlos, a pesar de que se vayan a reflejar en la planificación.

1.2.1. Requisitos funcionales

La lista completa de requisitos funcionales se muestra a continuación:

1. Crear *splash screen* (Vital).

2. Crear pantalla de bienvenida o *Home* (Vital).
3. Listar/Filtrar libros (Vital).
4. Listar/Filtrar autores (Vital).
5. Ver detalle de libro (Vital).
6. Ver detalle de autor (Vital).
7. Visualizar listado de noticias sobre lectura (Vital).
8. Filtrar noticias de lectura por preferencias de usuario (Vital).
9. Ver detalle de noticia (Vital).
10. Permitir al usuario establecer objetivos personales de lectura en términos de páginas, libros, etc. (Vital).
11. Permitir al usuario introducir cuánto ha leído diariamente o semanalmente en términos de páginas, libros, etc., es decir, crear y modificar un objetivo personal de lectura (Vital).
12. Visualizar un objetivo personal de lectura (Vital).
13. Mostrar recordatorios sobre los objetivos marcados por el usuario (Vital).
14. Mostrar gráficos y estadísticas de la actividad lectora del usuario a partir de información introducida por el usuario (Vital).
15. Permitir al usuario registrarse con usuario y contraseña en la aplicación (Vital).
16. Permitir al usuario iniciar sesión con sus credenciales (Vital).
17. Permitir al usuario recuperar su contraseña (Vital).
18. Crear menú lateral para acceder a todas las funcionalidades de la app (Vital).
19. Listar retos presentados por la aplicación (Importante).
20. Permitir a un usuario autenticado unirse a retos presentados por la aplicación de manera pública o privada[1] (Importante).
21. Visualizar detalle de reto (Importante).
22. Filtrar los retos para ver solo los del usuario (Importante).
23. Permitir al usuario marcar y desmarcar reto como completado (Importante).
24. Permitir al usuario marcar y desmarcar noticias como favoritas y listarlas en una sección aparte (Importante).
25. Permitir al usuario consultar noticias favoritas en modo *offline* (Estimulante).

[1] La diferencia será que en el detalle del reto se listará el nombre del usuario o no, respectivamente.

26. Permitir al usuario iniciar sesión con su cuenta de Google (Estimulante).
27. Permitir al usuario acceder a iTunes para visualizar libros buscados en *AvidReaders* (Estimulante).
28. Permitir a los usuarios valorar libros y autores con un sistema de *rating* (Estimulante).
29. Mostrar gráficos y/o estadísticas de libros y autores basados en las valoraciones de los usuarios (Estimulante).

1.2.2. Requisitos no funcionales

En lo que se refiere a los requisitos no funcionales, su listado es el que se muestra a continuación:

1. Desarrollar la aplicación para sistemas iOS (Vital).
2. Desarrollar la aplicación usando SwiftUI (Vital).
3. Crear interfaces de usuario usables y atractivas (Vital).
4. Internacionalización: Permitir al usuario seleccionar entre castellano e inglés (Vital).
5. Emplear Firebase [4] para gestionar los usuarios de la aplicación y almacenar cierta información en una base de datos remota (Vital).
6. Adaptar pantallas para que se visualicen correctamente en iPad (Vital).
7. Hacer uso de *loaders* en pantallas de carga para mejorar la usabilidad (Importante).
8. Usar una base de datos Realm para almacenar información que se pueda consultar sin conexión a Internet (Estimulante).
9. Implementar medidas para mejorar la accesibilidad de la aplicación (Estimulante).
10. Adaptar la aplicación para gestionar el *Dark mode* (Estimulante).
11. Mostrar los detalles de autores, libros y retos en formato *master-detail* en iPad (Estimulante).

1.3. Enfoque y método seguido

Como se ha dicho anteriormente, lo que se pretende con este trabajo de fin de máster es realizar un nuevo producto para estimular la lectura en los usuarios del mismo. La verdad es que este producto podría verse como una ampliación de la plataforma Goodreads, puesto que se pretenden emplear varios servicios de su API, pero puesto que se van a implementar más cosas y que no todo lo relacionado con libros y autores provendrá de dicha API (también se pretende emplear otras APIs), no será realmente una ampliación.

Por otra parte, el hecho de comenzar de cero con la idea propuesta nos proporciona una mayor libertad, tanto a nivel de funcionalidades o estructura de la aplicación como a nivel visual, puesto que no será necesario atenerse a las guías de diseño de Goodreads, por ejemplo.

1.4. Gestión de riesgos

En cualquier proyecto con cierta envergadura, y concretamente en proyectos de ámbito informático, es de suma importancia realizar una gestión de riesgos. Esto conlleva analizar una serie de riesgos potenciales y definir medidas para afrontarlos o evitarlos, de modo que aunque dichos riesgos se den, estemos preparados para afrontarlos y no supongan el fracaso o abandono del proyecto.

1.4.1. Medidas cualitativas de riesgo

En un primer lugar será necesario definir cómo se van a medir los riesgos, de modo que se pueda determinar cuáles son más importantes y, por ende, cuales merecen un mayor grado de atención. Así, es necesario definir las que se consideran las dos mayores dimensiones de los riesgos:

- **Impacto**: Medida que hace referencia al daño producido o a las consecuencias de que un riesgo se materialice.
- **Probabilidad**: Indicador del grado de ocurrencia de un riesgo.

Con las dos anteriores medidas se puede obtener el llamado **grado de exposición**, una medida cualitativa que nos permite evaluar la gravedad de un riesgo para así poder categorizarlos y otorgarles prioridades. En los cuadros 2, 3 y 4 se identifican los baremos a utilizar para impacto, probabilidad y grado de exposición, respectivamente.

Nivel de Impacto	Definición
Bajo	Un riesgo se considera bajo cuando el restraso que vaya a introducir en el proyecto en caso de materializarse sea menor a 4 días de trabajo.
Medio	Un riesgo se considera medio cuando el restraso que vaya a introducir en el proyecto en caso de materializarse sea mayor a 4 días de trabajo, pero menor que 8.
Alto	Un riesgo se considera de impacto alto cuando el retraso que vaya a introducir en caso de que se materialice sea mayor a 8 días de trabajo del alumno. Esto podría llevar a tener que prescindir de alguna funcionalidad o de algún componente de la memoria, lo cual podría resultar fatal y suponer la no compleción del proyecto.

Cuadro 2: Clases de impacto

Nivel de Probabilidad	Definición
Baja	Un riesgo tendrá una baja probabilidad de ocurrencia cuando se materialice en menos del 25 % de las ocasiones.
Media	Un riesgo tendrá una baja probabilidad de ocurrencia cuando se materialice entre el 25 % y el 65 % de las ocasiones.
Alta	Un riesgo tendrá una baja probabilidad de ocurrencia cuando se materialice en más del 65 % de las ocasiones.

Cuadro 3: Clases de probabilidad

		Probabilidad		
		Baja	**Media**	**Alta**
Impacto	**Bajo**	Bajo	Bajo	Medio
	Medio	Bajo	Medio	Alto
	Alto	Medio	Alto	Alto

Cuadro 4: Clases de grado de exposición

1.4.2. Estrategias ante los riesgos

Para poder hacer frente a los riesgos se pueden adoptar distintas estrategias, las cuales se pueden dividir en dos grupos:

- **Acciones de mitigación**: Estrategias cuyo fin es evitar que el riesgo se materialice. Es decir, se trata de acciones que se llevan a cabo antes de que el riesgo se de con vista a que no llegue a ocurrir.
- **Acciones de contingencia**: Estrategias cuyo fin es reducir los efectos de un riesgo una vez este ya se ha materializado. Es decir, son acciones que se llevan a cabo cuando el riesgo ya se ha traducido en una amenaza real y es necesario hacer algo para impedir que eche el proyecto abajo.

1.4.3. Especificación de riesgos

Una vez se han definido las medidas para establecer la gravedad y prioridad de los riesgos y los tipos de acciones que se pueden realizar para hacerles frente, se procederá a listar los principales riesgos del proyecto, proporcionando sus valores para las medidas cualitativas identificadas y las acciones de mitigación y contingencia que se llevarán a cabo.

Identificador	**RIS-1**
Nombre	Dificultades en la formación
Descripción	Existen dificultades a la hora de aprender cómo utilizar SwiftUI, ciertas librerías, APIs o funcionalidades necesarias para llevar a cabo ciertas facetas de la aplicación.
Impacto	Medio
Probabilidad	Alta
Exposición	Alta
Indicador	La realización de una funcionalidad de implementación está llevando un 50 % más de lo esperado debido a que el alumno no es capaz de comprender o adquirir los conocimientos necesarios para llevarla a cabo.
Acción de minimización	Para evitar que este riesgo se materialice, el alumno ha hecho un pequeño estudio de las tecnologías a utilizar de antemano para corroborar que su utilización es plausible. Además, se llevará a cabo una formación lo más exhaustiva posible de SwiftUI en los primeros estadios del proyecto, echando mano de una suscripción a la página web Ray Wenderlich [5], la cual se centra en el desarrollo móvil, centrándose en SwiftUI desde que salió en lo que se refiere a desarrollo para iPhones.
Acción de contingencia	En el caso de que el riesgo se materialice, se procederá momentáneamente a cambiar de funcionalidad y requisito para que el alumno no se atasque, en caso de que no sea una funcionalidad bloqueadora. En caso de que sí lo sea, se le pedirá ayuda al director o se planteará el hecho de reducir el alcance del proyecto al suprimir esta funcionalidad y todas aquellas que dependan directamente de ella, reemplazándolas posiblemente por otras en caso de que se considere necesario.

Cuadro 5: Riesgo RIS-1

Identificador	RIS-2
Nombre	Pérdida de información relacionada con el proyecto
Descripción	Debido a un fallo humano o técnico se podría perder información importante para el proyecto (código, memoria, etc.).
Impacto	Alto
Probabilidad	Baja
Exposición	Media
Indicador	En un momento dado, se ha perdido información en forma de código o documentación que era necesaria bien para realizar una entrega o bien para el posterior correcto desarrollo del proyecto.
Acción de minimización	Para evitar que este riesgo se materialice, toda la información importante se guardará en un repositorio de Gitlab que, al mismo tiempo, estará almacenado en el ordenador del alumno en la carpeta asociada a su cuenta de iCloud, la cual se sincroniza automáticamente. Se harán commits al repositorio cada hora y media de trabajo realizado.
Acción de contingencia	En el caso de que, aún teniendo en cuenta la acción de minimización, el riesgo se materialice, el proyecto se podría ver comprometido y habría que replantearlo.

Cuadro 6: Riesgo RIS-2

Identificador	RIS-3
Nombre	Imposibilidad temporal del alumno
Descripción	Debido a motivos personales o a restricciones relacionadas con la situación laboral del trabajador, el proyecto se ve retrasado respecto a la planificación presentada.
Impacto	Medio
Probabilidad	Media
Exposición	Media
Indicador	El proyecto se ve retrasado en 7 días o más de trabajo del alumno.
Acción de minimización	Para evitar que este riesgo se materialice, el alumno intentará ir siempre un paso por delante de la planificación. Además, no se pedirán todos los días de vacaciones en el trabajo para así poder pedirlos y dedicarlos completamente al proyecto en caso de que se necesite.
Acción de contingencia	En el caso de que el riesgo se materialice, se procederá a emplear días de vacaciones del trabajo para poder dedicárselos completamente al proyecto y volver a ajustarse en la medida de lo posible a la planificación original.

Cuadro 7: Riesgo RIS-3

1.5. Planificación del Trabajo

Para poder determinar qué tareas se van a realizar, en qué secuencia, cuáles se deben terminar antes de dar comienzo a otras o comprobar que se están cumpliendo los objetivos propuestos, es imperativo disponer de una planificación. Este elemento permitirá indicar claramente qué tareas será necesario realizar para alcanzar los objetivos del proyecto, así como la secuencia en la que se llevarán a cabo. Además, también se definirán los hitos del proyecto, es decir, aquellos puntos o fechas en las que habrá que proporcionar un ítem para entregar al cliente y que representan la finalización de una fase y, por lo general, el comienzo de la siguiente.

No obstante, antes de poder realizar la planificación es preciso determinar el ciclo de vida del proyecto. Esto significa que se debe decidir qué estrategia se va a seguir a la hora de repartir las tareas en el tiempo, valorar cuánto nos estamos ateniendo a la planificación o la importancia que le vamos a dar a las pruebas o a la aparición de nuevos cambios, entre otras cosas.

Tanto la elección del ciclo de vida a seguir, la disponibilidad temporal del alumno, los recursos que se necesitarán o el diagrama de Gantt serán elementos que se tratarán en las siguientes secciones.

1.5.1. Ciclo de vida

Tal y como se ha mencionado anteriormente, la elección del ciclo de vida es una decisión muy importante, puesto que determinará cómo se va a aproximar la realización de las diversas tareas y en qué momento se van a abordar. De este modo, es una decisión en la que se debe invertir bastante tiempo.

Tras haberle dedicado el tiempo necesario, se ha decidido que el ciclo de vida que mejor se adapta al proyecto que se va a realizar, tanto a nivel de alcance como temporal, es el ciclo de vida por **incrementos**. Este ciclo de vida se elige cuando los requisitos que se pretenden realizar (es decir, el alcance) están bien definidos desde un primer momento, pero sin dejar de lado el estudio del coste económico y temporal a lo largo de proyecto [6], a diferencia de lo que ocurre con el ciclo de vida en cascada. Se considera que esta aproximación podría encajar bien con este trabajo de fin de máster porque, al fin y al cabo, la entrega de las diversas PECs constituye una secuencia de incrementos a llevar a cabo, además de que será necesario visitar de nuevo la planificación temporal realizada en un primer momento, lo cual nos lleva al estudio y revisión continua del tiempo y la planificación.

Por otra parte, se considera que el producto a entregar no será solo el código fuente, sino que también será la memoria, los prototipos de diseño y el manual de usuario, elementos que se irán mejorando o añadiendo a lo largo del trabajo. Así, lo que realmente se estará haciendo será entregar nuevas partes que, al término del trabajo, acabarán constituyendo el producto final.

Otra ventaja de la adopción del ciclo de vida por incrementos para este trabajo es que en dicha aproximación se debería hacer un poco de cada una de las fases del proyecto en todos los incrementos (análisis, diseño, codificación y pruebas). Aunque parezca que esto no concuerde con la división de entregas correspondientes a las PECs, no es así. Esto es porque para poder completar la primera PEC ha sido preciso llevar a cabo una fase de análisis del estado del arte, el dominio de la necesidad que da lugar al proyecto y las tecnologías a emplear. Esta fase ha llevado al alumno a realizar un diseño preliminar para poder hacerse una idea de lo que se quería hacer y plasmar sus ideas sobre el papel. Posteriormente, ha sido necesaria una pequeña etapa de codificación y pruebas para comprobar que era factible el uso de ciertas librerías, fundamentales para poder cumplir con ciertos requisitos marcados como vitales. En consecuencia, se ha realizado un incremento para poder llevar a cabo la primera PEC, cosa que se repetirá para los siguientes.

Para concretar un poco más, el número de incrementos que se realizarán será 4, correspondiente a las 4 PECs que se deben entregar.

1.5.2. Disponibilidad temporal del alumno

La definición del alcance del proyecto y, por tanto, los tiempos que se manejarán en el mismo dependen en gran medida de los recursos humanos disponibles para dicho proyecto. En el caso presente,

puesto que se trata de un trabajo de fin de máster realizado individualmente, la cantidad de recursos humanos será 1, el alumno. Así, es de vital importancia conocer su disponibilidad temporal para poder llevar a cabo la planificación y obtener el diagrama de Gantt resultante.

En primer lugar, ha de decirse que el alumno se encuentra en el momento de realización del trabajo de fin de máster trabajando a jornada parcial, concretamente 35 horas a la semana. Esto hace que no pueda dedicarle la jornada completa al desarrollo del presente proyecto. Así, es de suma importancia establecer unas estimaciones del tiempo que se le va a dedicar al presente trabajo en el día a día (días laborales) y en los fines de semana y festivos. A continuación se muestran las horas que se le dedicará al trabajo en los dos tipos de días identificados:

- Días laborales: **1.5h**
- Días festivos/Fines de semana: **6h**

Una vez se saben las horas, ha de determinarse cuántos de estos dos tipos de días hay a lo largo del proyecto, de modo que se puedan determinar las horas que se van a dedicar en total. Cabe destacar que el alumno vive y trabaja en Santiago de Compostela (A Coruña), por lo que los festivos que se tendrán en cuenta serán los de dicha ciudad. La totalidad de festivos se refleja en el cuadro 8.

Festivo	PEC
25 de febrero	PEC1
19 de marzo	PEC2
9 de abril	PEC3
10 de abril	PEC3
1 de mayo	PEC3
21 de mayo	PEC4

Cuadro 8: Listado de festivos

Al saber los festivos, ahora es fácil determinar cuántos días laborales y festivos (contando también los fines de semana) habrá entre el 19 de febrero (inicio del trabajo de fin de máster) y el 5 de junio (fecha límite de entrega de la PEC3), lo cual nos permitirá calcular las horas totales del proyecto. Así, las horas se calculan a continuación:

$$73 \text{ días laborales} * 1.5\text{h horas} + 36 \text{ días festivos} * 6\text{h} = 325{,}5\text{h}$$

Como se puede ver, las horas de proyecto obtenidas a partir de los cálculos son **325.5 horas**. Puesto que el trabajo de fin de máster se corresponde con 12 ECTS y cada uno de estos se suele corresponder con 25 a 30 horas de trabajo, la multiplicación por estos valores resulta en el intervalo [300 horas, 360 horas], en el cual cae el valor calculado para el presente trabajo.

1.5.3. Diagrama de Gantt

Como se ha mencionado en el apartado 1.5.2, se dedicará un total de 325,5 horas repartidas en 109 días al trabajo de fin de máster. Para poder realizar el diagrama de Gantt, lo más fácil será determinar cuál será la disponibilidad media del alumno a lo largo del proyecto. Así, si estos 109 días los pasamos a semanas, obtenemos 15,5, que en horas (de una jornada laboral habitual de 40 horas) serían 622,85 horas. Puesto que se van a trabajar 325,5 horas en ese período, la disponibilidad media del alumno será de un 53 %.

De este modo, aunque la herramienta con la que se desarrolla el diagrama de Gantt, ProjectLibre [7], no gestione bien los fines de semana, las horas que se ponen de más por la semana se compensan con los fines de semana, en los que se refleja que no se trabaja. Así, al término de la semana los valores se contrarrestan y se obtiene una planificación ajustada a lo que realmente se pretende hacer.

El diagrama de Gantt resultante de tener en cuenta todas las tareas a realizar y asignar al alumno al 53 % a cada una de ellas es el que se puede ver en las figuras 1 a 4.

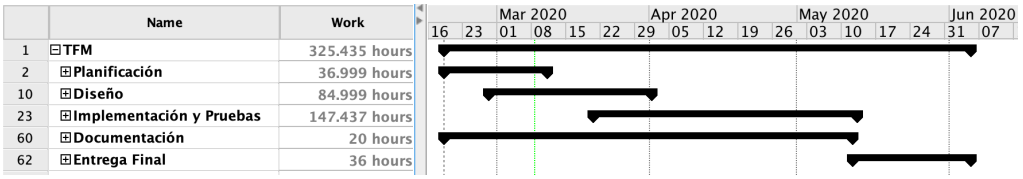

Figura 1: Diagrama de Gantt comprimido

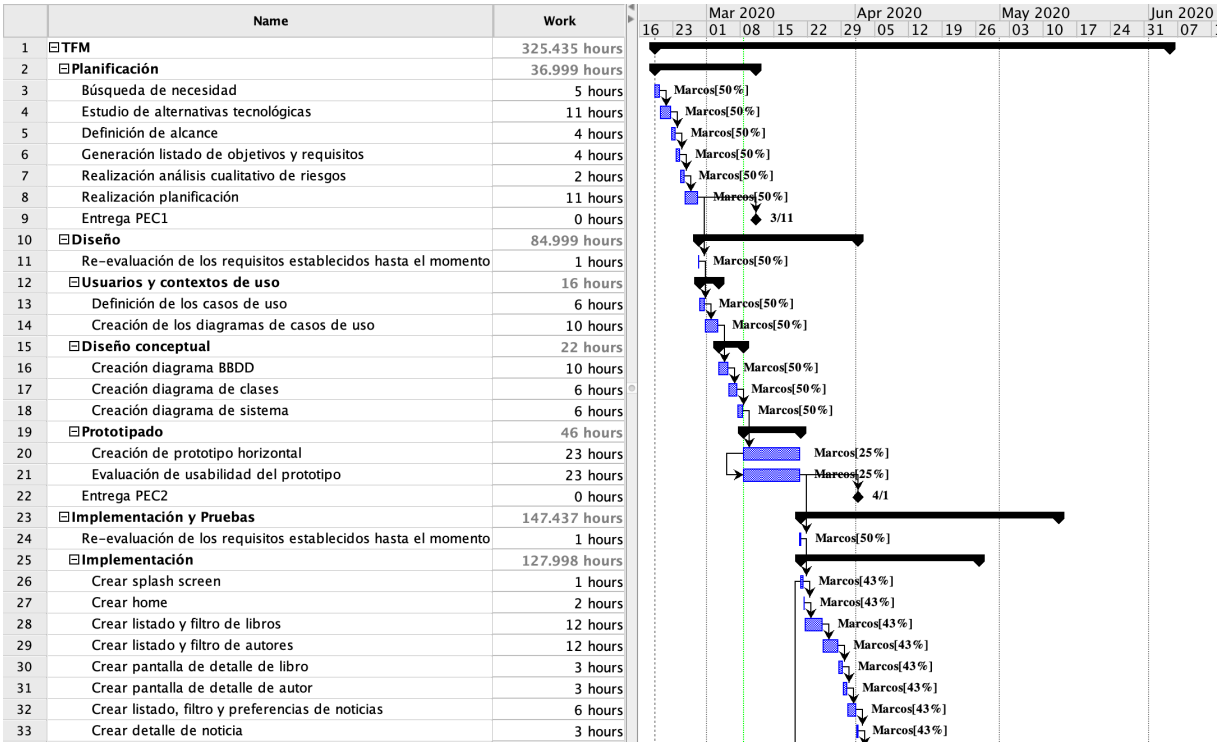

Figura 2: Diagrama de Gantt: Planificación y diseño

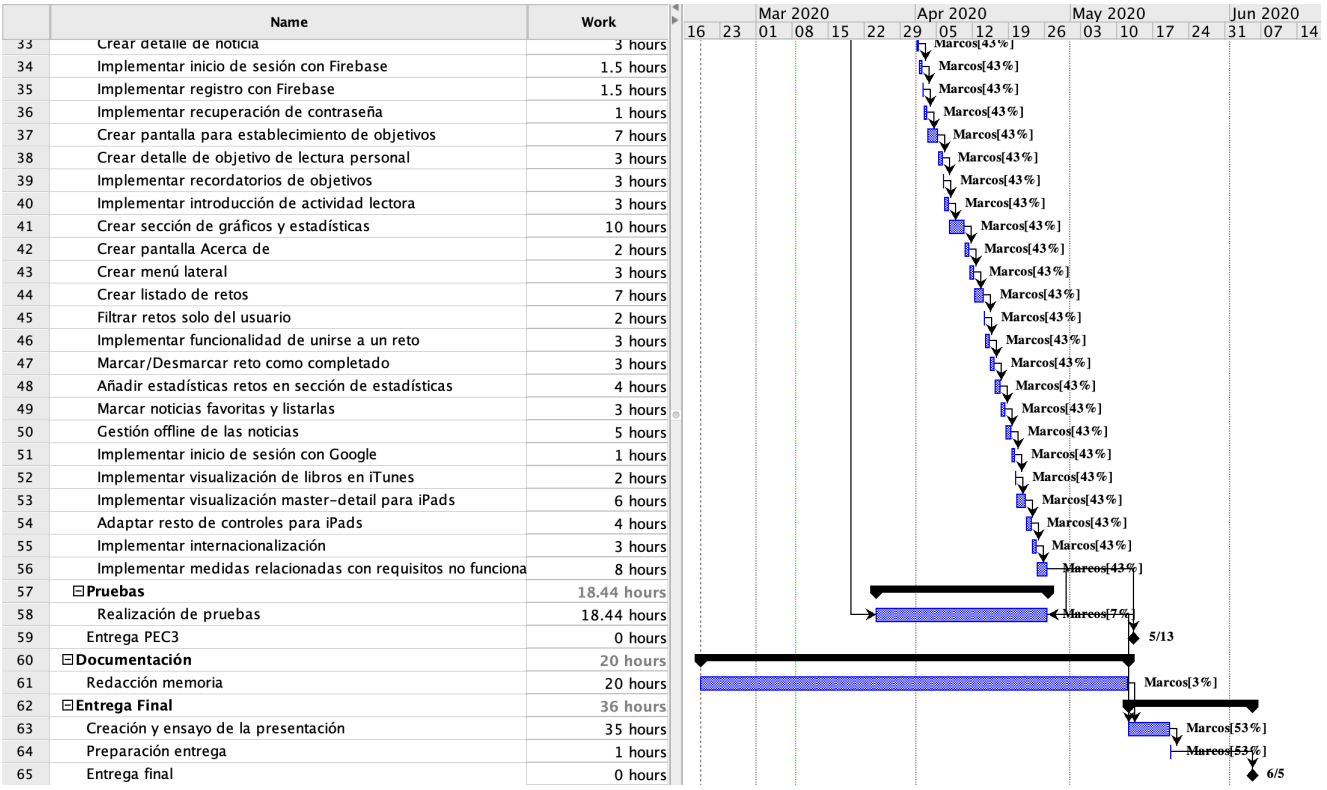

Figura 3: Diagrama de Gantt: Implementación, pruebas y entrega final

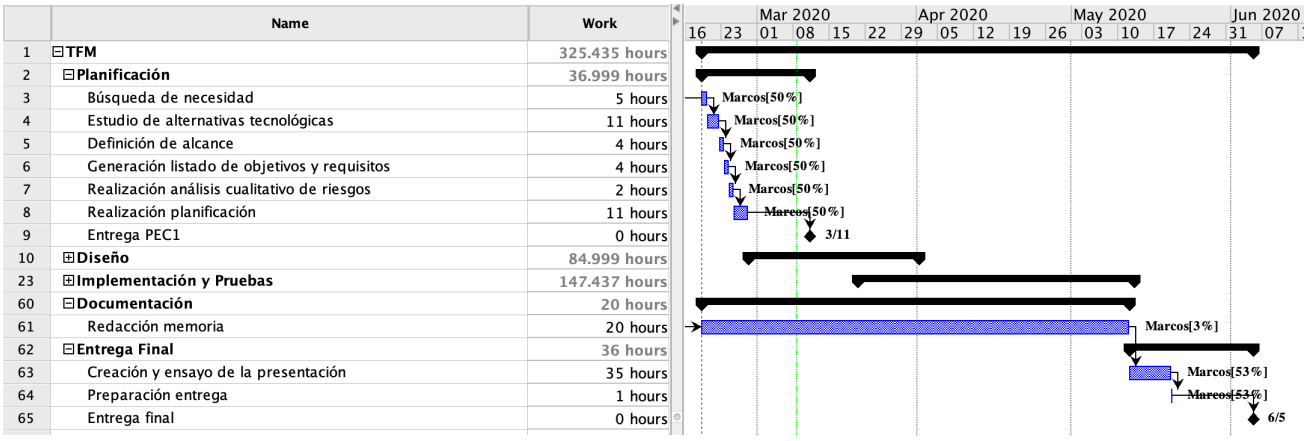

Figura 4: Diagrama de Gantt: Fase de documentación

Como se puede ver en las anteriores figuras, se han establecido 5 fases, 4 de ellas correspondientes a una de las PECs en las que se divide el trabajo de fin de máster y una dedicada a la documentación, puesto que la memoria se tendrá que ir mejorando y ampliando poco a poco a medida que avanza el proyecto. Tal y como se mencionó anteriormente, se ha añadido el recurso Marcos con una disponibilidad del 53 %, o bien de un nivel inferior cuando hay tareas que se deben realizar en paralelo. En caso contrario, el alumno estaría sometido a un esfuerzo mayor que su disponibilidad. Así, si nos fijamos en las tareas de implementación, por ejemplo, podemos ver como el recurso Marcos está a una disponibilidad del 43 %, ya que al mismo tiempo está realizando las pruebas (lo cual le supone un

esfuerzo del 7 %) y actualizando la documentación (esta tarea supone un 3 % del esfuerzo del recurso).

Cabe destacar que antes de un hito, representado por una entrega, siempre se empieza con la siguiente fase o incremento, pues se considera que, por ejemplo, no es necesario todo el tiempo reservado para la PEC1 para completarla, mientras que para la etapa de implementación se necesitará más tiempo que el conformado por la duración de la PEC3.

Por otra parte, existen tareas que se realizan en paralelo, como es la creación del prototipo de la app y su evaluación, las cuales son acciones que van de la mano y se retroalimentan entre ellas. Lo mismo ocurre con la implementación y las pruebas, pues será necesario hacer diversas pruebas a lo largo de todo el desarrollo para poder comprobar que las funcionalidades están bien y son estables, lo cual permite avanzar a las siguientes. Además, cabe destacar que en los tiempos indicados para las tareas de implementación ya están reflejadas las horas que llevará documentar cada una de ellas en código.

Además, en las fases intermedias se dedica tiempo para re-evaluar los objetivos y requisitos del proyecto, de modo que sea posible identificar problemas a tiempo y replantear el alcance del proyecto en caso de que sea necesario.

Por último, volviendo a la etapa de documentación, se puede ver como comienza al mismo tiempo que la primera tarea de la planificación (relación *Start-Start*) y debe finalizar para poder comenzar con la etapa final, lo cual puede verse mejor en la figura 4.

1.6. Gestión de las comunicaciones

La comunicación es un aspecto muy importante en un proyecto informático, puesto que sin una buena definición de las comunicaciones es muy fácil perder de vista dónde nos tendríamos que encontrar en el desarrollo o no generar determinados documentos en puntos cruciales del proyecto. Así, es muy importante definir quiénes serán los receptores de las comunicaciones y concretar cuáles serán estas. En los siguientes subapartados se determinarán los principales interesados o *stakeholders* del proyecto, así como las matrices de comunicación que definirán los intercambios entre las distintas partes que se verán implicadas en él.

1.6.1. Identificación de *stakeholders*

En el ámbito de la gestión de proyectos, un interesado o *stakeholder*, según el PMBOK[8], es un individuo, grupo u organización que puede afectar, verse afectado o percibirse a sí mismo como posible afectado por una decisión, actividad o resultado de un proyecto. De este modo, resulta de especial interés identificar quiénes son los interesados de un proyecto, puesto que, en función de la importancia que tengan dichos interesados en el ámbito del proyecto, pueden ejercer una gran influencia sobre el mismo.

Para este proyecto se considerará que, aparte del alumno, el único interesado será el coordinador, Eduard Martín Lineros. Se ha decidido no incluir a los otros miembros del equipo docente y asistente del máster por no tener claro en qué grado están interesados en el proyecto. En los cuadros 9 y 10 se muestra la información de los dos interesados identificados.

Faceta	Descripción
Nombre	Eduard Martín Lineros
Rol	Tutor/Coordinador/Director
Contacto	emartinli@uoc.edu
Requisitos primordiales	Ser la principal fuente de apoyo al alumno, tanto a nivel de resolución de dudas como de coordinación.
Expectativas principales	Finalización exitosa del proyecto, con un código y documentación asociada que cumpla con los requisitos impuestos.
Fases de mayor interés	Todo el proyecto
Influencia	Alta
Apoyo/Neutral/Opositor	Apoyo

Cuadro 9: Información sobre el interesado Eduard Martín Lineros

Faceta	Descripción
Nombre	Marcos García Rouco
Rol	Alumno/Programador
Contacto	marcosgr@uoc.edu
Requisitos primordiales	Llevar a cabo la implementación del software descrito en el presente documento y generar la documentación asociada.
Expectativas principales	Finalización exitosa del proyecto dentro de la fecha límite, con un código y documentación asociada que cumpla con los requisitos impuestos.
Fases de mayor interés	Todo el proyecto
Influencia	Alta
Apoyo/Neutral/Opositor	Apoyo

Cuadro 10: Información sobre el interesado Marcos García Rouco

1.6.2. Matrices de comunicación

En este apartado se procederá a indicar la dirección de las comunicaciones, así como el contenido de las mismas, entre los interesados del proyecto. Estas matrices pueden verse en los cuadros 11 y 12.

Faceta	Descripción
Flujo de la comunicación	Eduard Martín Lineros → Marcos García Rouco
Propósito de la comunicación	Resolver dudas, informar de errores o tareas, realizar aclaraciones, matizar o informar sobre nuevos requisitos y preguntar sobre el avance del proyecto.
Nivel de detalle	Alto
Importancia	Muy importante
Formato de la información	Correos electrónicos
Idioma(s)	Castellano

Cuadro 11: Matriz de comunicación Eduard Martín Lineros → Marcos García Rouco

Faceta	Descripción
Flujo de la comunicación	Marcos García Rouco → Eduard Martín Lineros
Propósito de la comunicación	Preguntar dudas, realizar aclaraciones, e informar sobre el avance del proyecto.
Nivel de detalle	Alto
Importancia	Muy importante
Formato de la información	Correos electrónicos
Idioma(s)	Castellano

Cuadro 12: Matriz de comunicación Marcos García Rouco → Eduard Martín Lineros

1.7. Breve sumario de productos obtenidos

El término de cada incremento se verá representado por un hito, el cual se traduce en una entrega. Es necesario definir qué se va a entregar en cada uno de estos momentos para que el cliente sepa qué va a poder evaluar en cada punto del proyecto. Dichos ítemes se listan a continuación:

- **Planificación**: Memoria con información sobre la planificación y alcance del proyecto, así como un pequeño informe de seguimiento.
- **Diseño**: Actualización de la memoria y quizás un archivo correspondiente a un prototipo interactivo, en caso que se realice con una herramienta de terceros y pueda generarse un prototipo interactivo, así como un pequeño informe de seguimiento.
- **Implementación**: Actualización de la memoria y código fuente, así como un pequeño informe de seguimiento.

- **Entrega Final**: Versión final de la memoria, manual de usuario (como anexo de la memoria) y presentación del trabajo realizado a lo largo del proyecto.

2. Diseño

La etapa de diseño se lleva a cabo una vez se han identificado el alcance del proyecto y la forma en la que este se llevará a su compleción, en términos de esfuerzo, tiempo y estrategia. Esto hace que en esta etapa sea posible identificar a los usuarios y definir cómo va a ser su interacción con la aplicación a crear, así como definir la estructura de la misma mediante el desarrollo de un conjunto de diagramas. Estos diagramas servirán dos funciones: determinar de forma clara y precisa qué se va a hacer exactamente y qué estructura tendrá, lo que a su vez nos llevará a poder detectar errores o carencias no identificados en la etapa de planificación.

2.1. Cambios o modificaciones producto de la reevaluación

Antes de proceder con la fase de diseño será necesario determinar si hay cambios que se quieran o se necesite realizar sobre lo especificado en la parte de planificación. En este caso, así es. Concretamente, se quiere modificar el requisito funcional en el que se especificaba que la aplicación contaría con un menú lateral para gestionar su navegación. No obstante, tras pensarlo en profundidad se ha decidido seguir la navegación empleada en las aplicaciones creadas por Apple. En ellas no existe menú lateral, sino que se echa mano de *tabs* o pestañas, así como de gestos laterales (*swipes*) para avanzar o retroceder en el árbol de navegación (véanse figuras 5 y 6).

Dicho esto, la planificación no tiene porque verse afectada, ya que el tiempo que se le iba a dedicar a diseñar e implementar el menú lateral se le dedicará al nuevo diseño y a la implementación de estos controles alternativos.

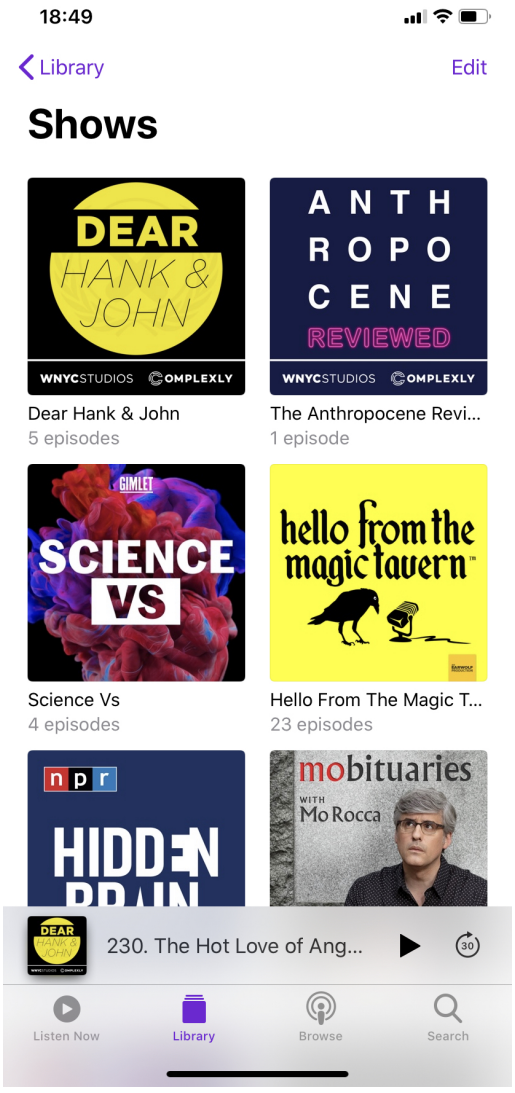

Figura 5: Navegación en Podcasts

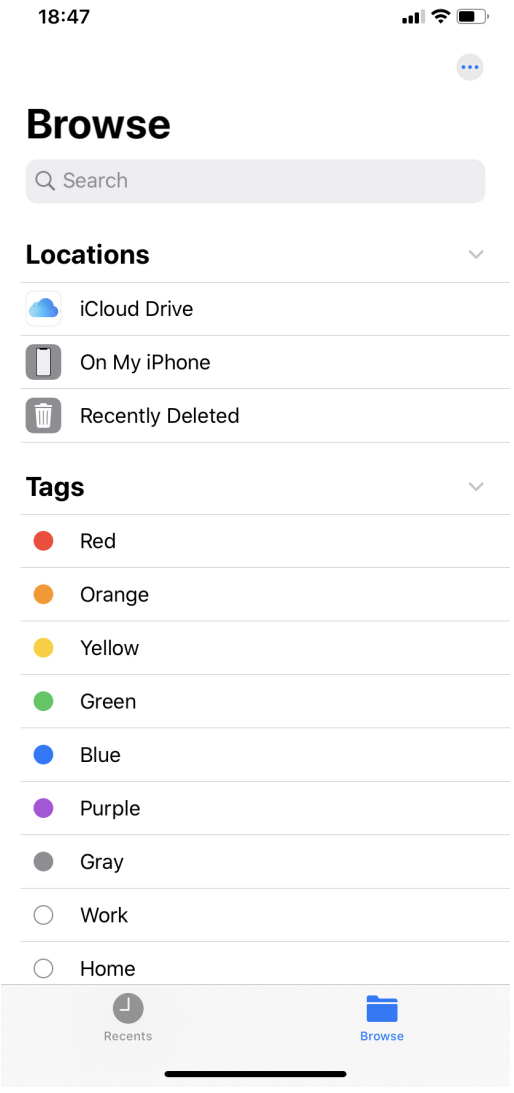

Figura 6: Navegación en Files

2.2. Casos de uso

Para poder diseñar una aplicación que se adapte lo mejor posible a sus usuarios es necesario determinar en primer lugar quiénes son estos y en qué contextos van a interaccionar con la aplicación, así como en qué modo y con qué fines. Una manera de dar respuesta a estas incertidumbres es la creación de un listado de casos de uso, los cuales se corresponden con casuísticas en las que los usuarios emplean la aplicación para llevar a cabo alguna acción.

Antes de pasar a listar los casos de uso, es preciso determinar quiénes serán los actores, es decir, las entidades o entes que llevarán a cabo acciones mediante la interacción con la aplicación. En este caso particular solo se contará con un tipo de actor: **usuario**. Esto se debe a que la aplicación o sistema solo se verá influenciado por las acciones de un usuario genérico, por ninguna otra entidad ajena a la aplicación.

Con fin de hacer más legible la información concerniente a los casos de uso, estos se han dividido en 5 diagramas, en los cuales se agrupan aquellos casos de uso más relacionados. A continuación de cada diagrama de casos de uso se muestran una serie de cuadros que los explican y especifican en mayor detalle.

2.2.1. Gestión de usuarios

Puesto que iniciar sesión en la aplicación hace que el abanico de funcionalidades se extienda considerablemente (sin iniciar sesión no se podrá valorar libros y autores o unirse a retos, entre otros), se empezará por esta sección de la aplicación. En la figura 7 se pueden ver los casos de uso relacionados con este ámbito. Los cuadros correspondientes a los casos de uso representados son del 13 al 15.

En este diagrama, al no representar todos los casos de uso de la aplicación, no se percibe, pero el caso de uso "Iniciar sesión" tendrá gran importancia por el hecho de que su compleción dependerán muchos otros casos de uso.

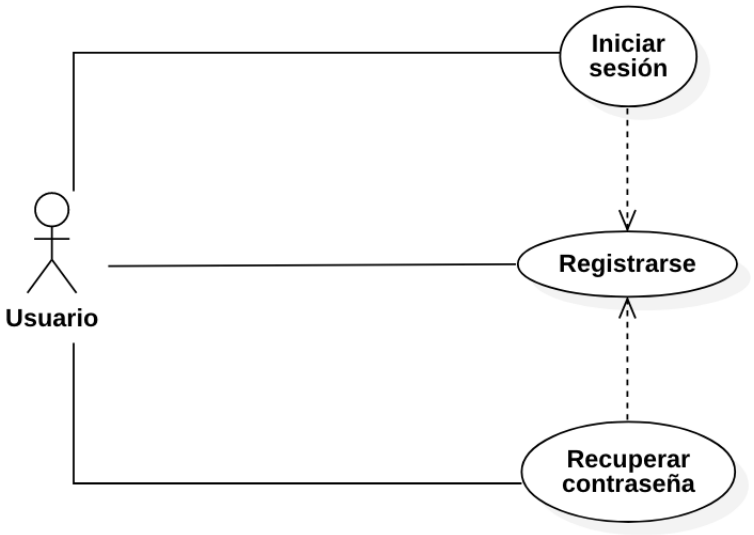

Figura 7: Diagrama de casos de uso: Gestión de usuarios

Como se puede ver, existen dependencias entre "Registrarse" y los otros dos casos de uso, pues ninguno de ellos sería posible sin la realización previa del primero.

Identificador	CU-1
Nombre	Registrarse
Descripción	Introducir email y contraseña (o credenciales de Google) para crear un nuevo usuario en la aplicación.
Actor(es)	Usuario
Precondiciones	—
Escenario principal	1. El usuario selecciona la opción Cuenta/Account del menú principal de la aplicación. 2. El usuario introduce sus credenciales en los controles presentados para tal fin. 3. El usuario presiona el botón para registrarse. 4. Se le muestra un mensaje al usuario indicándole que se ha registrado de manera satisfactoria. `(A1)`
Escenario(s) alternativo(s)	`A1-1` Las credenciales introducidas no son válidas y se le indicará por pantalla. `A1-2` Tras introducir unas credenciales válidas, el flujo será igual al del escenario principal.
Postcondiciones	El usuario visualiza un mensaje indicándole que se ha registrado correctamente.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que los usuarios introduzcan credenciales correctas puedan crear un nuevo usuario.

Cuadro 13: Caso de Uso CU-1

Identificador	CU-2
Nombre	Iniciar sesión
Descripción	Introducir email y contraseña para iniciar sesión en la aplicación.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-1 (cuadro 13) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario selecciona la opción Cuenta/Account del menú principal de la aplicación. 2. El usuario introduce sus credenciales en los controles presentados para tal fin. `(A1)` 3. El usuario presiona el botón para iniciar sesión. 4. Se le muestra un mensaje al usuario indicándole que ha iniciado sesión de manera satisfactoria. `(A2)`
Escenario(s) alternativo(s)	`A1-1` El usuario indica que quiere iniciar sesión con su cuenta de Google. `A1-2` El usuario introduce sus credenciales de Google e inicia sesión de este modo. — `A2-1` Las credenciales introducidas no son válidas y se le indicará por pantalla. `A2-2` Tras introducir unas credenciales válidas, el flujo será igual al del escenario principal.
Postcondiciones	El usuario visualiza un mensaje indicándole que ha iniciado sesión correctamente.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que los usuarios introduzcan credenciales correctas puedan iniciar sesión.

Cuadro 14: Caso de Uso CU-2

Identificador	CU-3
Nombre	Recuperar contraseña
Descripción	Introducir email para solicitar envío de nueva contraseña.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-1 (cuadro 13) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario selecciona la opción Cuenta/Account del menú principal de la aplicación. 2. El usuario introduce su email en el control presentado para tal fin. 3. El usuario presiona el botón para recuperar contraseña. 4. Se le muestra un mensaje al usuario indicándole que se le ha enviado a su correo para restablecer su contraseña.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza un mensaje indicándole que se le ha enviado un correo para restablecer su contraseña.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 90 % de los casos en los que los usuarios soliciten el cambio de contraseña les llegue un correo con este fin.

Cuadro 15: Caso de Uso CU-3

2.2.2. Gestión de libros y autores

En el diagrama de esta sección se indicarán las interacciones que se podrán producir entre usuarios y libros y autores. Como se ha mencionado anteriormente, el caso de uso de iniciar sesión (CU-1, reflejado en el cuadro 13) juega un papel importante en varios casos de uso de este diagrama (es una precondición para ellos). En los cuadros 16 a 22 se detallan los casos de uso que se representan en el diagrama de la figura 8.

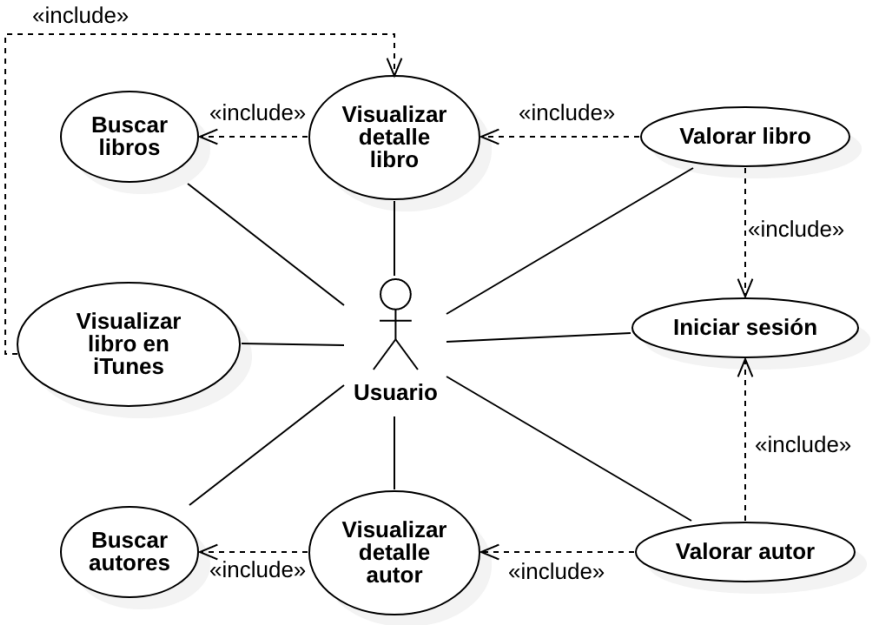

Figura 8: Diagrama de casos de uso: Gestión de libros y autores

Identificador	CU-4
Nombre	Buscar libros
Descripción	Buscar y listar libros filtrando por nombre.
Actor(es)	Usuario
Precondiciones	—
Escenario principal	1. El usuario selecciona la opción Libros del menú principal de la aplicación. 2. El usuario introduce una cadena de texto para buscar libros. 3. El usuario visualiza una serie de libros que cumplen con los parámetros introducidos. `(A1)`
Escenario(s) alternativo(s)	`A1-1` El usuario no visualiza ningún libro porque no existe ninguno que se corresponda con los parámetros introducidos.
Postcondiciones	El usuario visualiza uno o más libros que cumplen con los parámetros introducidos.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que los usuarios busquen libros con parámetros correctos, los encuentren.

Cuadro 16: Caso de Uso CU-4

Identificador	CU-5
Nombre	Visualizar detalle de libro
Descripción	Ver el detalle de un libro (título, descripción, portada, etc.).
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-4 (cuadro 16) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en uno de los resultados obtenidos al completar el CU-4. 2. Se le muestra al usuario el detalle del libro seleccionado.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza una pantalla en la que se muestra el detalle del libro seleccionado.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario presione sobre un libro, vea el detalle del mismo.

Cuadro 17: Caso de Uso CU-5

Identificador	CU-6
Nombre	Valorar libro
Descripción	Otorgarle una puntuación de 1 a 5 a un libro.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los casos de uso CU-1 y CU-5 (cuadros 13 y 17, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en los controles del detalle de libro creados con el fin de valorar un libro. 2. Se le indica al usuario que ha valorado el libro.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza un cambio en la interfaz de usuario que refleja su valoración del libro, la cual quedará almacenada.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente valorar un libro, le resulte posible.

Cuadro 18: Caso de Uso CU-6

Identificador	CU-7
Nombre	Visualizar libro en iTunes
Descripción	Ver en iTunes el mismo libro que se está viendo en *AvidReaders*.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el caso de uso CU-5 (cuadro 17) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en el botón para visualizar el libro en iTunes que se encuentra en el detalle de libro. 2. El usuario sale de la aplicación y es enviado a iTunes para visualizar el libro. `(A1)`
Escenario(s) alternativo(s)	`A1-1` No se encuentra el libro en iTunes y se muestra un aviso a tal efecto.
Postcondiciones	El usuario visualiza el libro en la aplicación iTunes.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 90 % de los casos en los que el usuario intente visualizar un libro en iTunes, le resulte posible.

Cuadro 19: Caso de Uso CU-7

Identificador	CU-8
Nombre	Buscar autores
Descripción	Buscar y listar autores filtrando por nombre.
Actor(es)	Usuario
Precondiciones	—
Escenario principal	1. El usuario selecciona la opción Autores del menú principal de la aplicación. 2. El usuario introduce una cadena de texto para buscar autores. 3. El usuario visualiza el autor cuyo nombre concuerde más con los parámetros introducidos. `(A1)`
Escenario(s) alternativo(s)	`A1-1` El usuario no visualiza ningún autor porque no existe ninguno que se corresponda con los parámetros introducidos.
Postcondiciones	El usuario visualiza el autor cuyo nombre concuerde más con los parámetros introducidos.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que los usuarios busquen autores con parámetros correctos, los encuentren.

Cuadro 20: Caso de Uso CU-8

Identificador	CU-9
Nombre	Visualizar detalle de autor
Descripción	Ver el detalle de un autor (nombre, información, foto, etc.).
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-8 (cuadro 20) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en uno de los resultados obtenidos al completar el CU-8. 2. Se le muestra al usuario el detalle del autor seleccionado.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza una pantalla en la que se muestra el detalle del autor seleccionado.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario presione sobre un autor, vea el detalle del mismo.

Cuadro 21: Caso de Uso CU-9

Identificador	CU-10
Nombre	Valorar autor
Descripción	Otorgarle una puntuación de 1 a 5 a un autor.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los casos de uso CU-1 y CU-9 (cuadros 13 y 21, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en los controles del detalle de autor creados con el fin de valorar un autor. 2. Se le indica al usuario que ha valorado el autor.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza un cambio en la interfaz de usuario que refleja su valoración del autor, la cual quedará almacenada.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente valorar un autor, le resulte posible.

Cuadro 22: Caso de Uso CU-10

2.2.3. Gestión de noticias

En el diagrama de la figura 9 se representan los casos de uso que tienen que ver con la gestión de noticias sobre libros y la lectura en general que se mostrarán en *AvidReaders*. Dichos casos de uso se detallan en los cuadros 23 a 28.

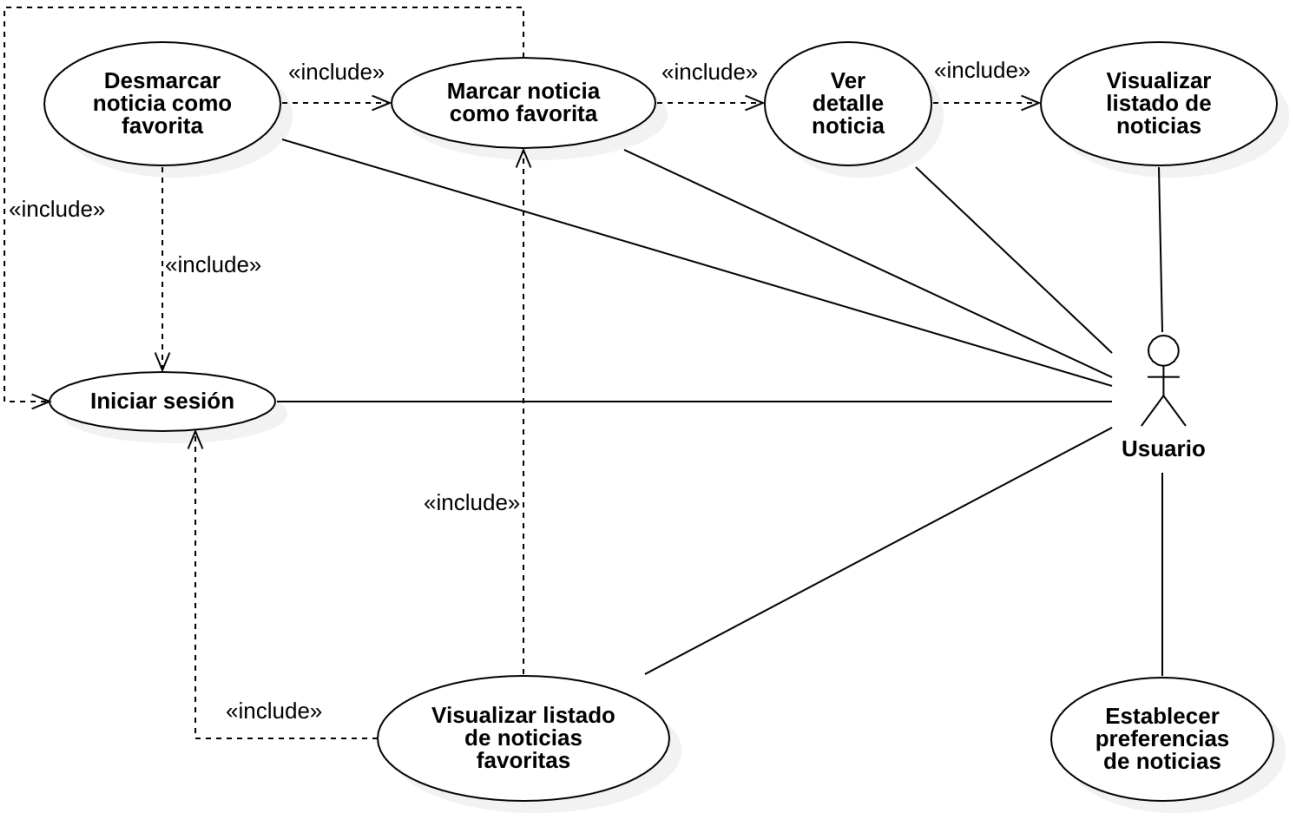

Figura 9: Diagrama de casos de uso: Gestión de noticias

Identificador	CU-11
Nombre	Visualizar listado de noticias
Descripción	Visualizar listado de noticias sobre lectura, las cuales pueden estar filtradas por preferencias de usuario establecidas previamente.
Actor(es)	Usuario
Precondiciones	—
Escenario principal	1. El usuario selecciona la opción Noticias del menú principal de la aplicación. 2. El usuario visualiza un listado de noticias.`(A1)`
Escenario(s) alternativo(s)	`A1-1` El usuario había establecido previamente una serie de criterios para los cuales no hay noticias actualmente.
Postcondiciones	El usuario visualiza un listado de noticias sobre lectura.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que los usuarios soliciten visualizar noticias de lectura, puedan hacerlo.

Cuadro 23: Caso de Uso CU-11

Identificador	CU-12
Nombre	Ver detalle de noticia
Descripción	Ver el detalle de una noticia (título, contenido, fuente, etc.).
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-11 (cuadro 23) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en uno de los resultados obtenidos al completar el CU-11. 2. Se le muestra al usuario el detalle de la noticia seleccionada.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario visualiza una pantalla en la que se muestra el detalle de la noticia seleccionada.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario presione sobre una noticia, vea el detalle de la mismo.

Cuadro 24: Caso de Uso CU-12

Identificador	CU-13
Nombre	Marcar noticia como favorita
Descripción	Marcar una noticia como favorita para que se guarde y se pueda visualizar en cualquier momento, incluso *offline*.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los casos de uso CU-1 y CU-12 (cuadros 13 y 24, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en el control para marcar una noticia como favorita. 2. El control presionado cambia de apariencia y se le indica al usuario que ha marcado la noticia como favorita.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario tendrá una noticia marcada como favorita, la cual podrá visualizar en la sección dedicada, incluso *offline*.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente marcar una noticia como favorita, le resulte posible.

Cuadro 25: Caso de Uso CU-13

Identificador	CU-14
Nombre	Desmarcar noticia como favorita
Descripción	Desmarcar una noticia como favorita.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el caso de uso CU-13 (cuadro 25) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario presiona en el control para desmarcar una noticia como favorita. 2. El control presionado cambia de apariencia y se le indica al usuario que se ha desmarcado la noticia como favorita.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario tendrá una noticia menos marcada como favorita, lo cual verá reflejado en la sección dedicada.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente desmarcar una noticia como favorita, le resulte posible.

Cuadro 26: Caso de Uso CU-14

Identificador	CU-15
Nombre	Visualizar listado de noticias favoritas
Descripción	Visualizar todas las noticias marcadas como favoritas.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los casos de uso CU-1 y CU-13 (cuadros 13 y 25, respectivamente) con anterioridad.
Escenario principal	1. El usuario accede a la sección Noticias del menú principal. 2. En el listado de noticias, el usuario presiona el control para visualizar las noticias favoritas. 3. El usuario es presentado con sus noticias favoritas. `(A1)`
Escenario(s) alternativo(s)	`A1-1` El usuario no había marcado noticias como favoritas con anterioridad, por lo que verá un listado vacío.
Postcondiciones	El usuario visualizará sus noticias favoritas, a pesar de que sean noticias antiguas.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar sus noticias favoritas, le resulte posible.

Cuadro 27: Caso de Uso CU-15

Identificador	CU-16
Nombre	Establecer preferencias de noticias.
Descripción	Determinar las etiquetas que se emplearán para buscar noticias.
Actor(es)	Usuario
Precondiciones	—
Escenario principal	1. El usuario accede a la sección Noticias del menú principal. 2. En la pantalla de listado de noticias el usuario deberá presionar el control para establecer las preferencias de noticias. 3. El usuario introducirá cadenas de texto (etiquetas). 4. El usuario debe presionar en el botón para guardar las preferencias para que estas se apliquen.
Escenario(s) alternativo(s)	—
Postcondiciones	A partir de este momento, en el presente dispositivo, el usuario solo visualizará noticias que tengan que ver con los criterios establecidos.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente establecer preferencias de lectura, le resulte posible.

Cuadro 28: Caso de Uso CU-16

2.2.4. Gestión de objetivos personales

En el diagrama de la figura 10 se representan los casos de uso que tienen que ver con la gestión de objetivos personales establecidos por los propios usuarios en la app. Dichos casos de uso se detallan en los cuadros 29 a 36.

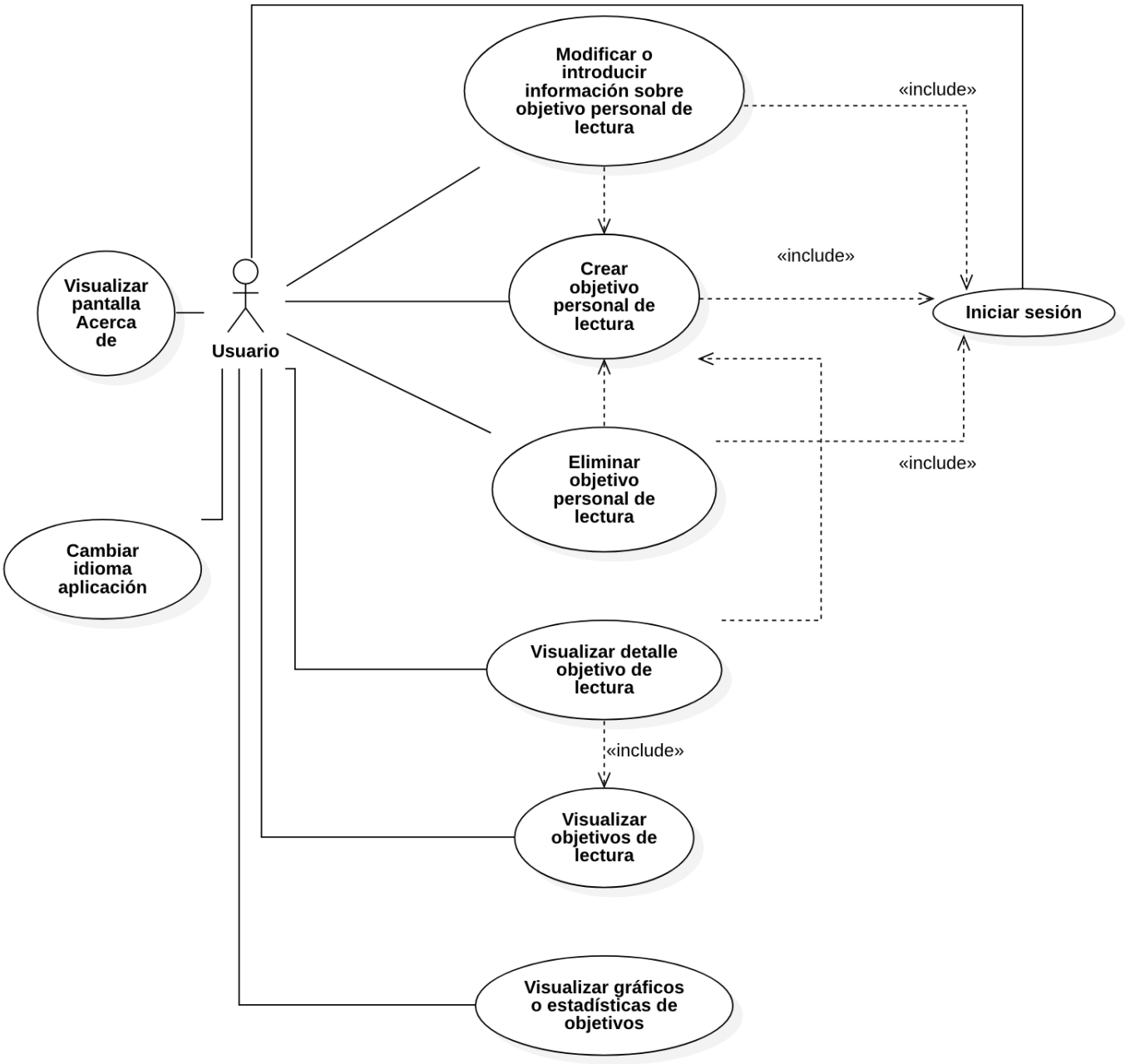

Figura 10: Diagrama de casos de uso: Gestión de objetivos personales

Identificador	CU-17
Nombre	Crear objetivo personal de lectura.
Descripción	Crear un objetivo personal de lectura en términos de libros o páginas y a cumplir diaria, semanal o mensualmente.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-11 (cuadro 23) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Objetivos personales del menú principal. 2. El usuario presiona en el botón para crear un nuevo objetivo. 3. El usuario introduce los parámetros necesarios para la creación del nuevo objetivo. 4. El objetivo es creado y el usuario es enviado de nuevo a la pantalla de objetivos personales.
Escenario(s) alternativo(s)	—
Postcondiciones	A partir de este momento el usuario tendrá un objetivo personal.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente crear un objetivo personal de lectura, le resulte posible.

Cuadro 29: Caso de Uso CU-17

Identificador	CU-18
Nombre	Eliminar objetivo personal de lectura.
Descripción	Eliminar un objetivo personal de lectura creado previamente.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-17 (cuadro 23 y 29, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Objetivos personales del menú principal. 2. El usuario hace un gesto horizontal o *swipe* sobre el objetivo personal a eliminar.
Escenario(s) alternativo(s)	—
Postcondiciones	A partir de este momento el usuario no verá el objetivo personal borrado y tampoco se reflejará en la sección de estadísticas de la aplicación.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente eliminar un objetivo personal de lectura, le resulte posible.

Cuadro 30: Caso de Uso CU-18

Identificador	CU-19
Nombre	Visualizar detalle objetivo personal de lectura.
Descripción	Ver las características y detalles de un objetivo personal de lectura.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-17 (cuadro 23 y 29, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Objetivos personales del menú principal. 2. El usuario hace un gesto horizontal o *swipe* sobre el objetivo personal a eliminar. 3. El usuario visualiza el listado actualizado de objetivos.
Escenario(s) alternativo(s)	—
Postcondiciones	A partir de este momento el usuario no verá el objetivo personal borrado y tampoco se reflejará en la sección de estadísticas de la aplicación.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente eliminar un objetivo personal de lectura, le resulte posible.

Cuadro 31: Caso de Uso CU-19

Identificador	CU-20
Nombre	Modificar objetivo personal de lectura.
Descripción	Introducir una actualización sobre el objetivo de lectura (más páginas leídas, compleción del objetivo, etc.).
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-17 (cuadro 23 y 29, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Objetivos personales del menú principal. 2. El usuario selecciona uno de los objetivos personales. 3. El usuario introduce las modificaciones sobre el objetivo de lectura y presiona el botón para actualizar. 4. El usuario es devuelto al listado de objetivos de lectura.
Escenario(s) alternativo(s)	—
Postcondiciones	El reto es actualizado de forma correspondiente.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente modificar un objetivo personal de lectura, le resulte posible.

Cuadro 32: Caso de Uso CU-20

Identificador	CU-21
Nombre	Visualizar listado de objetivos personales de lectura.
Descripción	Ver el listado de todos los objetivos personales de lectura del usuario.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-11 (cuadro 23) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Objetivos personales del menú principal. 2. El usuario hace un gesto horizontal o *swipe* sobre el objetivo personal a eliminar.
Escenario(s) alternativo(s)	—
Postcondiciones	A partir de este momento el usuario no verá el objetivo personal borrado y tampoco se reflejará en la sección de estadísticas de la aplicación.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar sus objetivos personales de lectura, le resulte posible.

Cuadro 33: Caso de Uso CU-21

Identificador	CU-22
Nombre	Visualizar gráficos o estadísticas.
Descripción	Ver gráficos y estadísticas relacionadas con la actividad lectora de los usuarios (valoraciones, retos y objetivos).
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-11 (cuadro 23) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Estadísticas del menú principal. 2. El usuario visualiza las estadísticas de su comportamiento en *AvidReaders*.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario es presentado con una serie de estadísticas que reflejan su actividad en la aplicación.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar sus estadísticas, le resulte posible.

Cuadro 34: Caso de Uso CU-22

Identificador	CU-23
Nombre	Cambiar idioma.
Descripción	Elegir el idioma de los controles de la interfaz entre inglés y castellano.
Actor(es)	Usuario
Precondiciones	-
Escenario principal	1. El usuario accede a la sección Configuración del menú principal. 2. El usuario selecciona uno de los idiomas listados. 3. El usuario percibe un cambio en los textos de la aplicación.
Escenario(s) alternativo(s)	—
Postcondiciones	Los textos de la aplicación son modificados acorde al idioma seleccionado por el usuario.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente cambiar el idioma de la aplicación, le resulte posible.

Cuadro 35: Caso de Uso CU-23

Identificador	CU-24
Nombre	Visualizar pantalla Acerca de.
Descripción	Visualizar la pantalla en la que se muestra información sobre la aplicación, las APIs de terceros, etc..
Actor(es)	Usuario
Precondiciones	-
Escenario principal	1. El usuario accede a la sección Acerca de del menú principal. 2. El usuario visualiza la información presentada por la aplicación.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario es presentado con una pantalla con información diversa sobre la aplicación.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar la susodicha pantalla, le resulte posible.

Cuadro 36: Caso de Uso CU-24

2.2.5. Gestión de retos

En el diagrama de la figura 11 se representan los casos de uso que tienen que ver con la gestión de retos presentados por la propia aplicación *AvidReaders*. Dichos casos de uso se detallan en los cuadros 37 a 39.

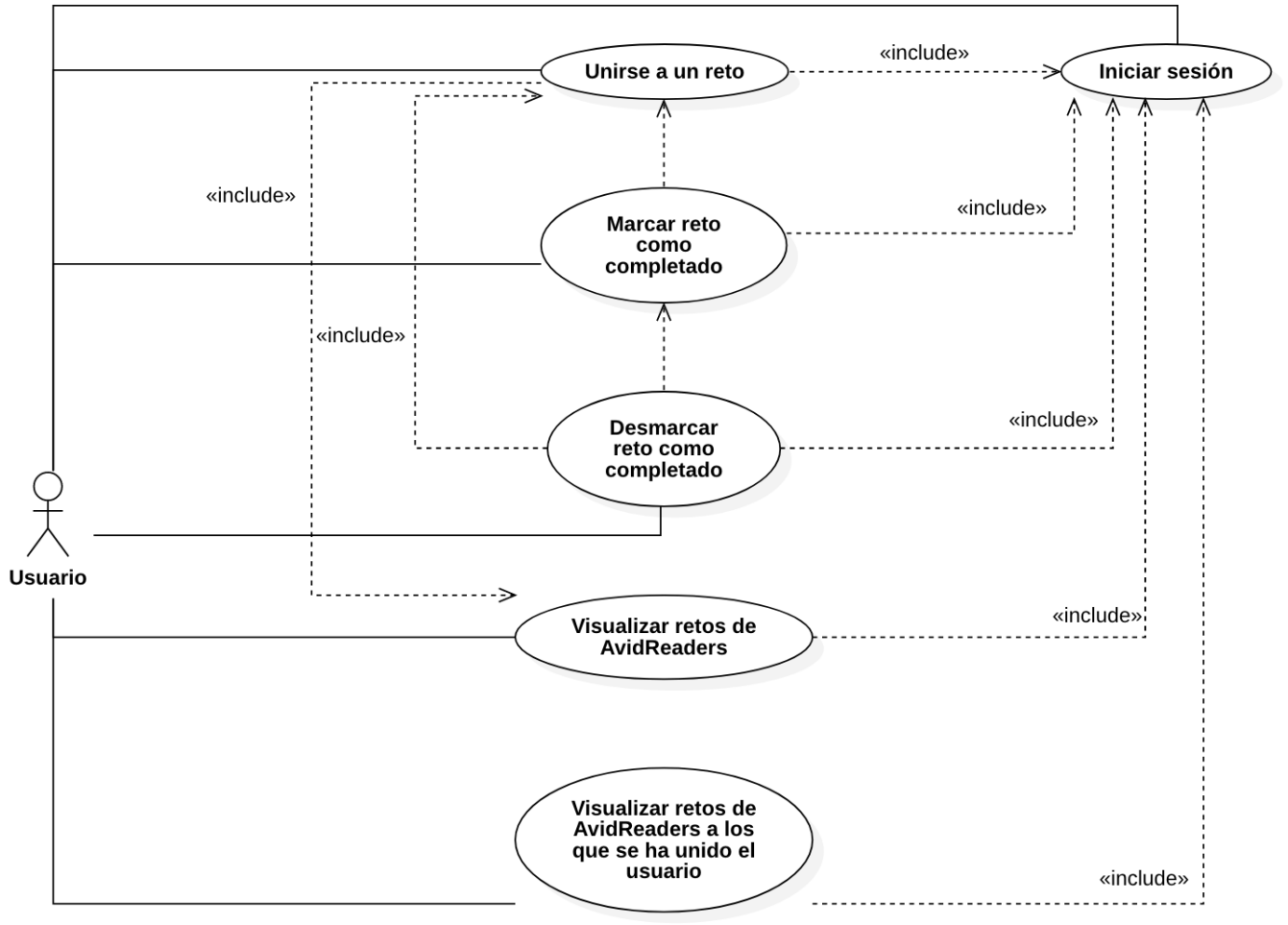

Figura 11: Diagrama de casos de uso: Gestión de retos

Identificador	CU-25
Nombre	Visualizar retos de *AvidReaders*.
Descripción	Ver retos de lectura creados por *AvidReaders*.
Actor(es)	Usuario
Precondiciones	Se debe haber completado el CU-11 (cuadro 23) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Retos del menú principal. 2. El usuario visualiza los retos de lectura presentados por *AvidReaders*.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario es presentado con una serie de retos a los que podrá unirse.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar los retos de *AvidReaders*, le resulte posible.

Cuadro 37: Caso de Uso CU-25

Identificador	CU-26
Nombre	Unirse a un reto de *AvidReaders*.
Descripción	Unirse a un reto creado por por *AvidReaders*.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-25 (cuadros 23 y 37, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Retos del menú principal. 2. El usuario selecciona uno de los retos de lectura presentados por *AvidReaders*. 3. El usuario presiona el botón designado para unirse al reto.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario estará unido a un reto de lectura y este pasará a figurar como parte de sus estadísticas.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente unirse a uno de los retos de *AvidReaders*, le resulte posible.

Cuadro 38: Caso de Uso CU-26

Identificador	CU-27
Nombre	Visualizar retos de *AvidReaders* a los que se ha unido el usuario.
Descripción	Ver aquellos retos de lectura creados por *AvidReaders* a los que el usuario se ha unido.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-26 (cuadros 23 y 38, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Retos del menú principal. 2. El usuario pulsa en el control para ver solo aquellos retos a los que se ha unido.
Escenario(s) alternativo(s)	—
Postcondiciones	El usuario es presentado con solamente aquellos retos a los que se ha unido.
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente visualizar sus retos de *AvidReaders*, le resulte posible.

Cuadro 39: Caso de Uso CU-27

Identificador	CU-28
Nombre	Marcar un reto como completado.
Descripción	Marcar uno de los retos a los que el usuario estaba unido como completado.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-26 (cuadros 23 y 38, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Retos del menú principal. 2. El usuario selecciona uno de los retos de lectura presentados por *AvidReaders*. `(A1)` 3. El usuario presiona el botón designado para marcarlo como completado.
Escenario(s) alternativo(s)	`A1-1` El usuario presiona en un reto al que no se ha unido, por lo que tendrá que unirse para poder marcarlo como completado.
Postcondiciones	El usuario verá ese reto como marcado (también se reflejará en la sección de estadísticas).
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente marcar uno de los retos de *AvidReaders* como completado, le resulte posible.

Cuadro 40: Caso de Uso CU-28

Identificador	CU-29
Nombre	Desmarcar un reto como completado.
Descripción	Desmarcar uno de los retos a los que el usuario estaba unido como completado.
Actor(es)	Usuario
Precondiciones	Se debe haber completado los CU-11 y CU-28 (cuadros 23 y 40, respectivamente) satisfactoriamente con anterioridad.
Escenario principal	1. El usuario accede a la sección Retos del menú principal. 2. El usuario selecciona uno de los retos de lectura presentados por *AvidReaders*. `(A1)` 3. El usuario presiona el botón designado para desmarcarlo como completado. `(A2)`
Escenario(s) alternativo(s)	`A1-1` El usuario presiona en un reto al que no se ha unido, por lo que tendrá que unirse para poder marcarlo como completado. — `A2-1` El usuario selecciona un reto que no estaba marcado como completado, por lo que no será posible desmarcarlo como tal.
Postcondiciones	El usuario verá ese reto como desmarcado (también se reflejará en la sección de estadísticas).
Criterio de validación	Se considerará que este caso de uso estará validado cuando en el 100 % de los casos en los que el usuario intente desmarcar uno de los retos de *AvidReaders* como completado, le resulte posible.

Cuadro 41: Caso de Uso CU-29

2.3. Diseño de la arquitectura

En esta sección se detallará mediante el uso de diagramas la arquitectura de la aplicación. En primer lugar se presentará el diagrama de base de datos de la aplicación, lo cual servirá para introducir el diagrama de clases, pues ambos diagramas están estrechamente relacionados. Por último, se mostrará el diagrama de arquitectura de la aplicación, la cual seguirá un patrón MVVM (*Model-View-ViewModel*).

2.3.1. Diagrama de base de datos

El diagrama de base de datos nos permite representar las tablas o entidades que se usarán para el correcto funcionamiento de la aplicación. Este diagrama puede verse en la figura 12.

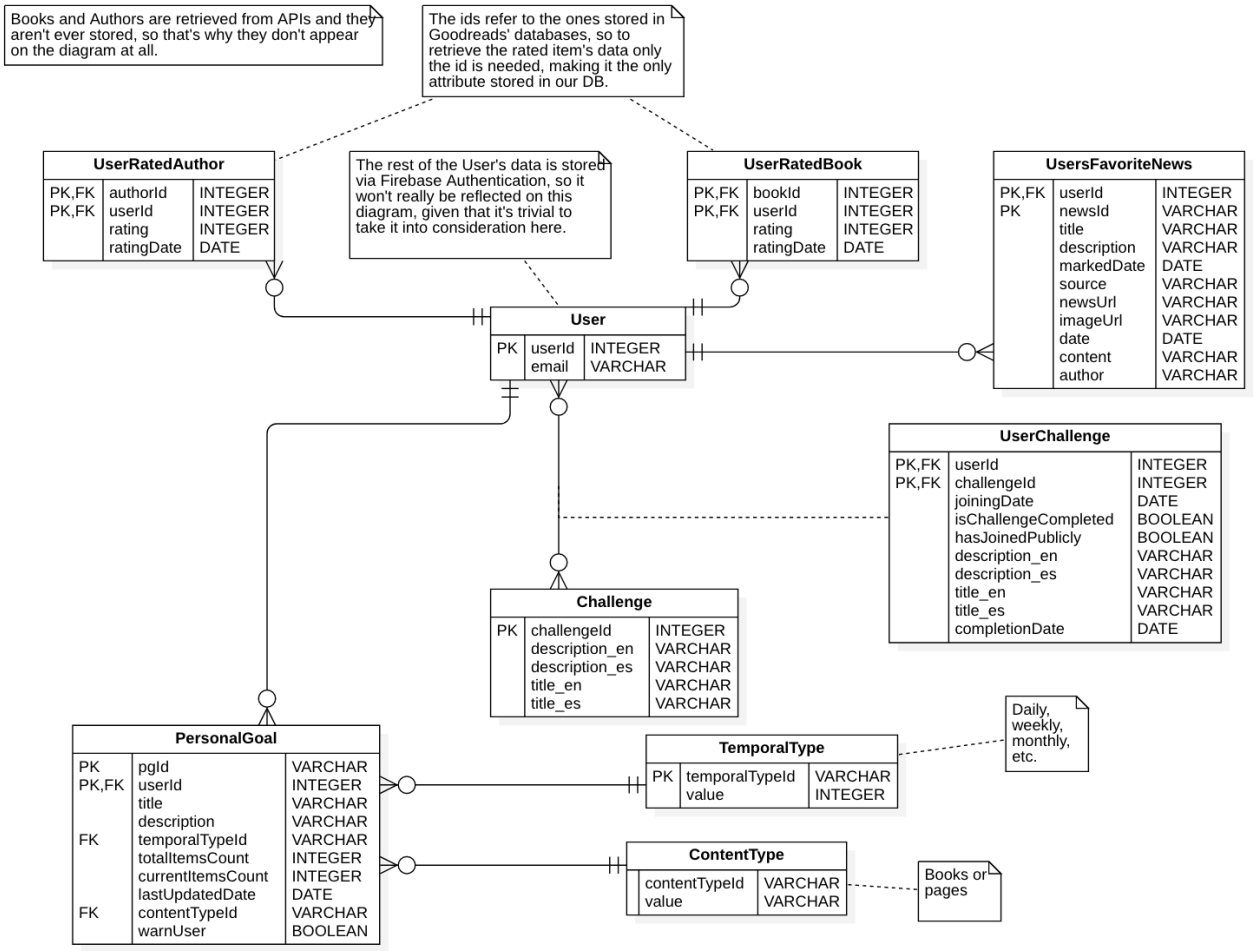

Figura 12: Diagrama de base de datos

En primer lugar se debe destacar el hecho de que aunque se ha creado este esquema de base de datos, la base de datos a utilizar no será relacional, sino que será no relacional. En concreto, se utilizará Firebase Realtime Database, la cual almacena los datos en formato JSON. No obstante, se considera que proporcionar un esquema de base de datos puede resultar de utilidad para tener claro como se van a relacionar los distintos elementos de la aplicación, ya que supone un buen soporte y punto de entrada al diagrama de clases. Un ejemplo preliminar de como se almacenarán los datos es el que puede verse a continuación:

```
{
  "users": {
    "1": {
      "email": "marcosgr@uoc.edu"
      }
    }
  ,
  "personalGoals": {
    "1": {
        "personalGoals_key1": {
        "pagesCount": 30,
```

```
      "observations": "I promise to read a lot!!",
      "lastUpdateDate": 1459361875666,
      "temporalTypeId" : "DAILY"
    },"personalGoals_key2": {
      "pagesCount": 210,
      "observations": "I'll try...",
      "lastUpdateDate": 1459361875666,
      "temporalTypeId" : "WEEKLY"
    }
  }
},
"temporalTypes": {
    "DAILY": {
      "value": 1
    },"WEEKLY": {
      "value": 7
    },"MONTHLY": {
      "value": 30
    }
},
"ratedAuthors": {
  "1": {
      "11588": {
      "rating": 3
    }
  }
},
"ratedBooks": {
  "1": {
      "11588": {
      "rating": 3
    }
  }
},
"favoriteNews": {
  "newsId1": {
    "title": "News about Stephen King's new book",
    "description": "bla bla",
    "markedDate": 1459361875666
  },"newsId2": {
    "title": "News about Mark Edwards's new book",
    "description": "bla bla",
    "markedDate": 1459361875666
  }
},
"challenges": {
  "challengeId1": {
    "description": "Description of this challenge."
  },"challengeId2": {
```

```
      "description": "Description of this challenge."
    },"challengeId3": {
      "description": "Description of this challenge."
    }
  },
  "userChallenges": {
    "1": {
      "challengeId1": {
        "joiningDate": 1459361875666,
        "isChallengeCompleted": false,
        "hasJoinedPublicly": true
      }
    }
  }

}
```

Como se puede ver, el diagrama se ha creado alrededor de la entidad *User*, pues de ella dependen muchas funcionalidades (recordemos que es necesario iniciar sesión en la aplicación para llevar a cabo muchas de las funcionalidades de la aplicación). Esta entidad se corresponde con el usuario que se autentica contra Firebase Authentication. Aparece representado en este diagrama, pero no sería necesario guardarlo como tal en una entidad, puesto que su información se obtiene directamente mediante la API de autenticación. No obstante, como el id de usuario se utilizará como índice para almacenar la información en Firebase Realtime Database, se refleja en este diagrama.

Por otra parte, se puede ver que se almacenarán los identificadores de Goodreads de los escritores y libros valorados por los usuarios (los cuales serán necesarios para obtener las entidades en un momento posterior), así como la propia valoración. En el caso de las noticias, estas se guardarán completamente bajo el id de cada usuario. Esto se debe a que la API de la cual se obtendrán, NewsAPI [9], solo muestra noticias de como mucho un día anterior a la búsqueda. De este modo, para guardar las noticias favoritas de los usuarios, que pueden estar en cualquier punto en el pasado, será necesario guardar toda la noticia, de ahí toda la información que se puede ver en la entidad *UsersFavoriteNews*.

En lo que respecta a los retos, estos se guardarán como una entidad independiente. Cuando un usuario se una a un reto entonces se echará mano de la entidad derivada *UserChallenge*, la cual servirá para guardar la fecha de unión al reto o para saber si el usuario quiere que se mantenga en privado que se ha unido al reto, entre otras cosas. Por su parte, los objetivos personales, como su propio nombre indica, son personales de un usuario, por eso son una entidad débil dependiente de la entidad *User*.

2.3.2. Diagrama de clases

Relacionado en gran medida con el diagrama de base de datos se encuentra el diagrama de clases. Este diagrama nos permite representar las clases y objetos que se emplearán en el proyecto, así como la relación entre los mismos. También se representan sus atributos, los cuales se utilizarán en la aplicación bien para mostrar información, bien para relacionar los objetos o bien estarán sujetos a modificaciones de los usuarios. Se entiende que para todos y cada uno de los atributos de los objetos

habrá un método *getter* y un método *setter*, pero se ha decidido no incluirlos en el diagrama para no saturarlo. Dicho diagrama puede verse en la figura 13.

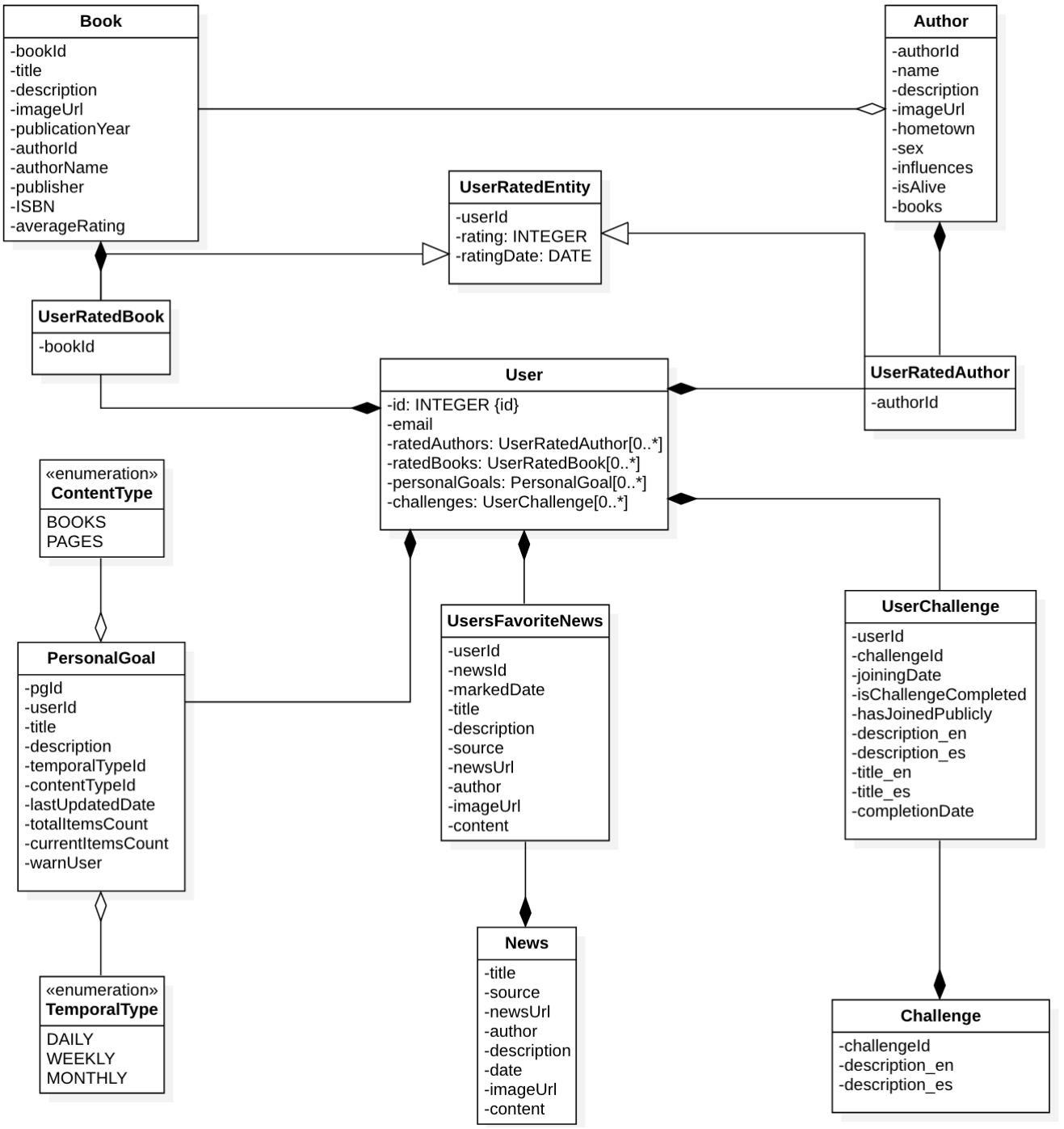

Figura 13: Diagrama de clases

Como se puede ver, este diagrama guarda una estrecha relación con el diagrama de base de datos. Lo único que difiere es que en este caso si se muestran clases para los libros y para los autores. Esto se debe a que al traer los datos mediante la API de Goodreads será necesario almacenarlos en entidades en memoria, para lo cual se echará mano de las clases *Book* y *Author*. Lo mismo ocurre con la API

de NewsAPI y la entidad *News*, pues las noticias también tendrán que ser guardadas temporalmente en memoria.

Por otra parte, se puede ver que se guardan una serie de atributos en cada clase que luego tendrán su importancia en el diseño, bien porque serán textos o bien porque serán controles que permitirán clasificar los distintos objetos o realizar unas u otras acciones sobre ellos.

2.3.3. Diagrama de arquitectura

Con el diagrama de arquitectura de un proyecto se pretende señalar cuáles serán sus componentes y cómo se relacionarán. A menudo, estos componentes se relacionarán siguiendo un patrón de diseño previamente establecido por la comunidad y utilizado de forma amplia, a pesar de que tenga diversas modificaciones y añadidos para adaptarse al proyecto particular. Este es el caso del trabajo de fin de máster actual, pues se ha optado por seguir un patrón MVVM (*Model-View-ViewModel*). Este se trata de un patrón de diseño muy similar al omnipresente Modelo-Vista-Controlador, pero en este caso se cambia el controlador, que, como su nombre indica, tenía que realizar el control entre el modelo, la vista y el usuario, por el llamado *ViewModel*. Este componente lo que permite es que la vista y el modelo estén relacionados de una manera más estrecha, de modo que un cambio en el modelo se refleje automáticamente en la vista sin necesidad de una entidad orquestadora. Del mismo modo, un cambio sobre la vista, si es sobre la representación de un elemento del modelo, hará que este se modifique inmediatamente.

En la figura 14 se muestra el diagrama MVVM para este proyecto. Como se puede ver, los modelos se corresponden con las clases del diagrama del diagrama de clases y con las entidades del diagrama de base de datos. Por su parte, en la parte etiquetada como ViewModel & Model se muestran los componentes que mostrarán los datos en pantalla. Estos serán componentes de SwiftUI, los cuales nos permiten integrar el modelo con la vista y echar mano de las posibilidades anteriormente mencionadas del patrón MVVM. Por otra parte, se ha decidido añadir en este diagrama los dos modelos de los que se leerán los datos: RealmDB para los datos guardados en el dispositivo (las noticias favoritas que se podrán visualizar *offline*) y Firebase Realtime Database para el resto de datos, incluidas también las noticias mencionadas (se guardarán las noticias en la nube por si el usuario utiliza la aplicación en otro dispositivo o la desinstala en algún momento).

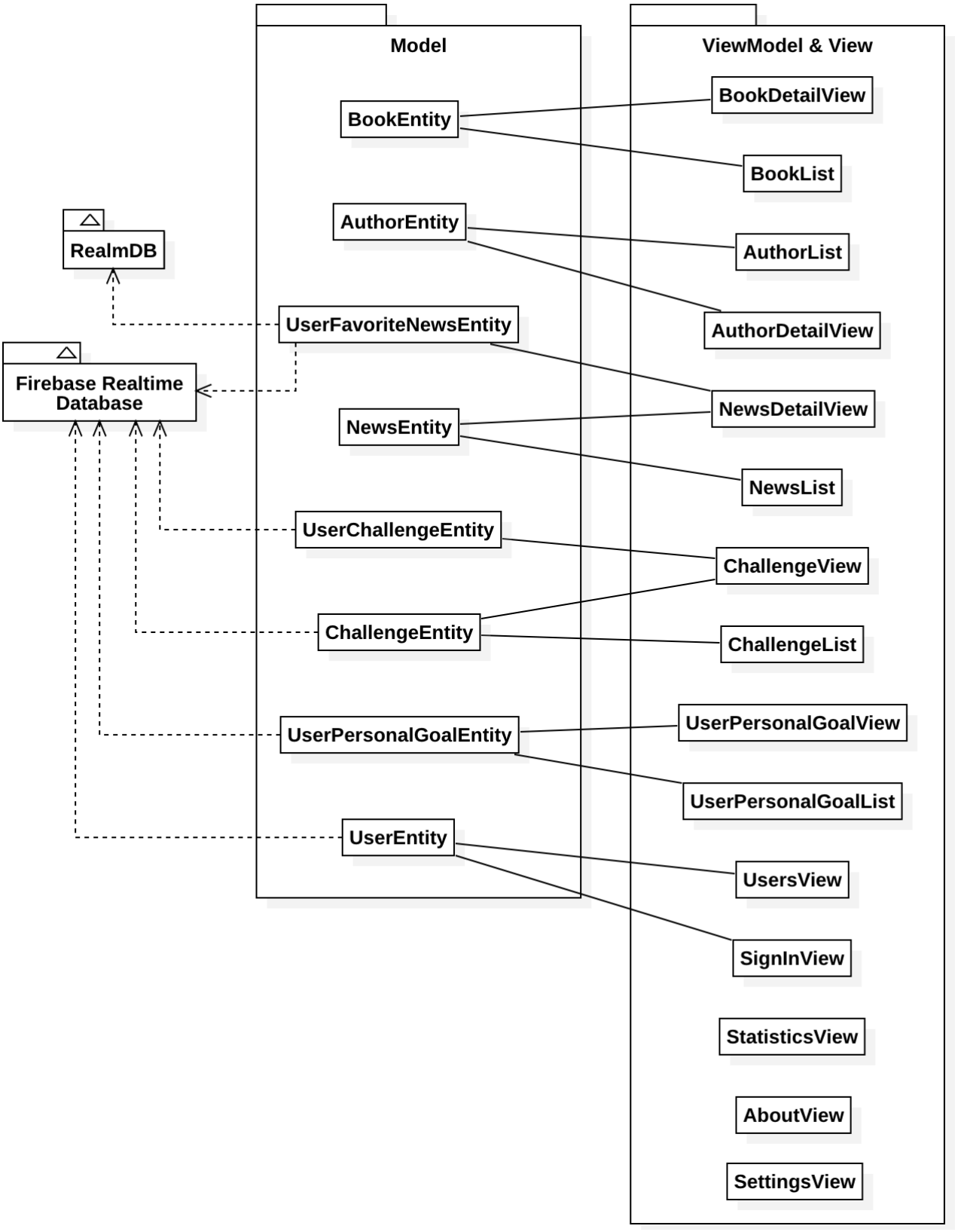

Figura 14: Diagrama de arquitectura

2.4. Prototipado

En cualquier proyecto de cierta envergadura se debe llevar a cabo una etapa de prototipado y no lanzarse a realizar la interfaz directamente en código, pues lo más seguro es que se detectarán carencias en los casos de uso, inconsistencias en los diagramas y problemas similares al empezar a colocar las distintas pantallas y elementos, ya que esto es lo que ha pasado al realizar esta parte durante el proyecto. De este modo, aunque en el documento las secciones del diseño se dispongan de forma secuencial para hacer la lectura del documento más coherente, se han llevado a cabo de forma casi cíclica, moviéndonos de una a otra al intentar suplir carencias y resolver errores no vistos previamente.

Como consecuencia, se han realizado dos prototipos: uno de baja fidelidad y otro de alta fidelidad. El primero se trata de una representación burda de las pantallas que tendrá la aplicación, sin prestar mucha atención al detalle, sino a obtener el conjunto de pantallas que harán falta y comprender un poco como será la navegación entre ellas. En el segundo de los prototipos, realizado con la herramienta Justinmind [10], ya se ha procedido a entrar en más detalle, dar toques de color y de posicionamiento de elementos y dejar zanjada la navegación entre pantallas. Cabe destacar que, al fin y al cabo, se trata de un prototipo, por lo que los datos no son reales ni se reproducen comportamientos reales (al presionar un *switch* no se cambian los datos, por ejemplo).

2.4.1. Prototipo de baja fidelidad

Como prototipo de baja fidelidad se ha escogido la realización de bocetos o *sketches* en papel. La ventaja de esta alternativa es que requiere poco esfuerzo, tanto a la hora de realizarlo como a la hora de introducir nuevos cambios. Este tipo de prototipos nos permite obtener una idea *grosso modo* de como será la interfaz sin necesidad de invertir mucho tiempo.

Lo que se ha hecho, aparte de realizar los bocetos como tal, ha sido anotar diversos controles para indicar a qué pantalla nos llevaría en caso de tocarlos o activarlos. En las figuras 15 a 17 se muestran todos los bocetos creados.

Como se puede ver en la figura 15, la app abriría y se mostraría un menú principal en el que estarían disponibles todas las opciones: Noticias, Libros y Autores, Retos, Objetivos personales, Estadísticas, Cuenta, Configuración y Acerca de. Pulsar en cada una de ellas nos llevaría a una pantalla distinta. Al lado de cada botón en el boceto hay una letra, la cual hace referencia luego al boceto que representa a dónde se viajaría al pulsarlo. Básicamente, como se puede ver en los bocetos, en los cuatro primeros casos pulsar el botón nos lleva a un listado de elementos que podemos pulsar y, a su vez, nos llevan a otras pantallas, en este caso de detalle. Cabe destacar que el caso de Libros y Autores es un poco diferente, pues aquí se echará mano de un *tab view*, es decir, habrá un control con dos pestañas para iterar entre libros y autores, lo cual le dará un toque diferente a la aplicación y permitirá al alumno experimentar con algo distinto.

Se puede ver como en la figura 16 se ha dejado un garabato de una pantalla en la que el alumno se había olvidado del título. Se ha dejado en la figura con toda intención para reflejar que los prototipos de baja fidelidad tienen esta finalidad: detectar errores y permitirnos corregirlos sin mucho esfuerzo temporal.

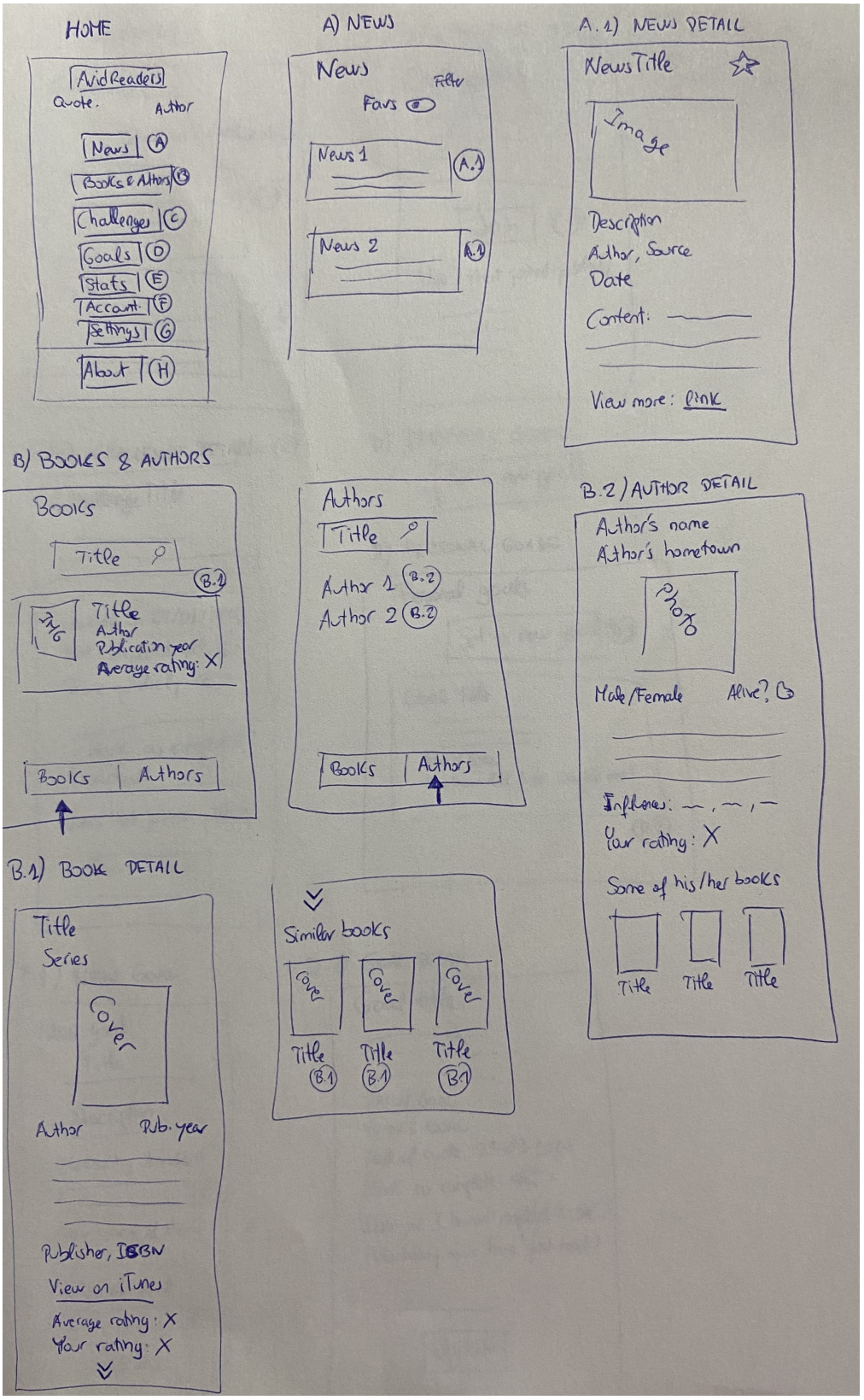

Figura 15: Bocetos, I

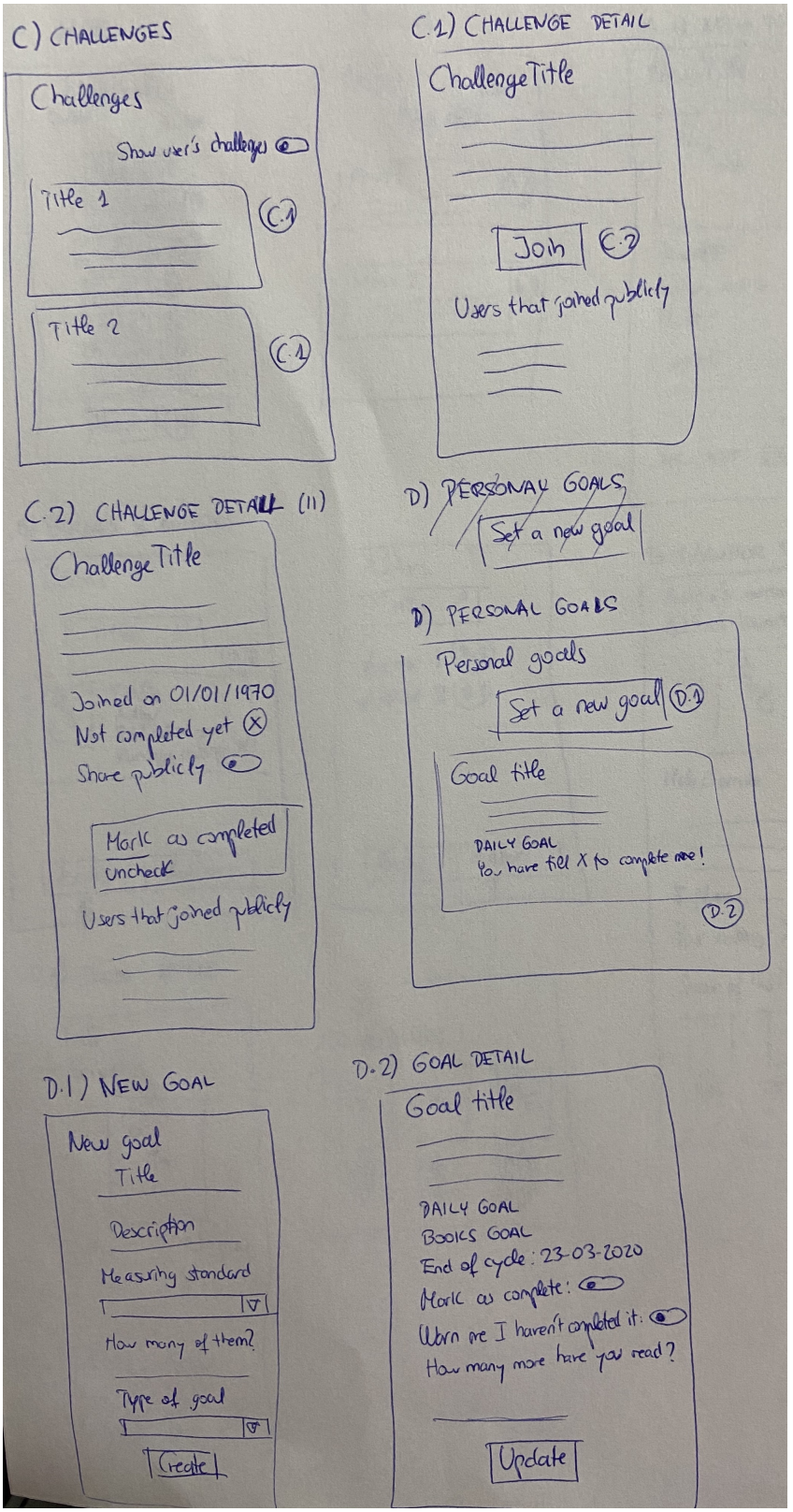

Figura 16: Bocetos, II

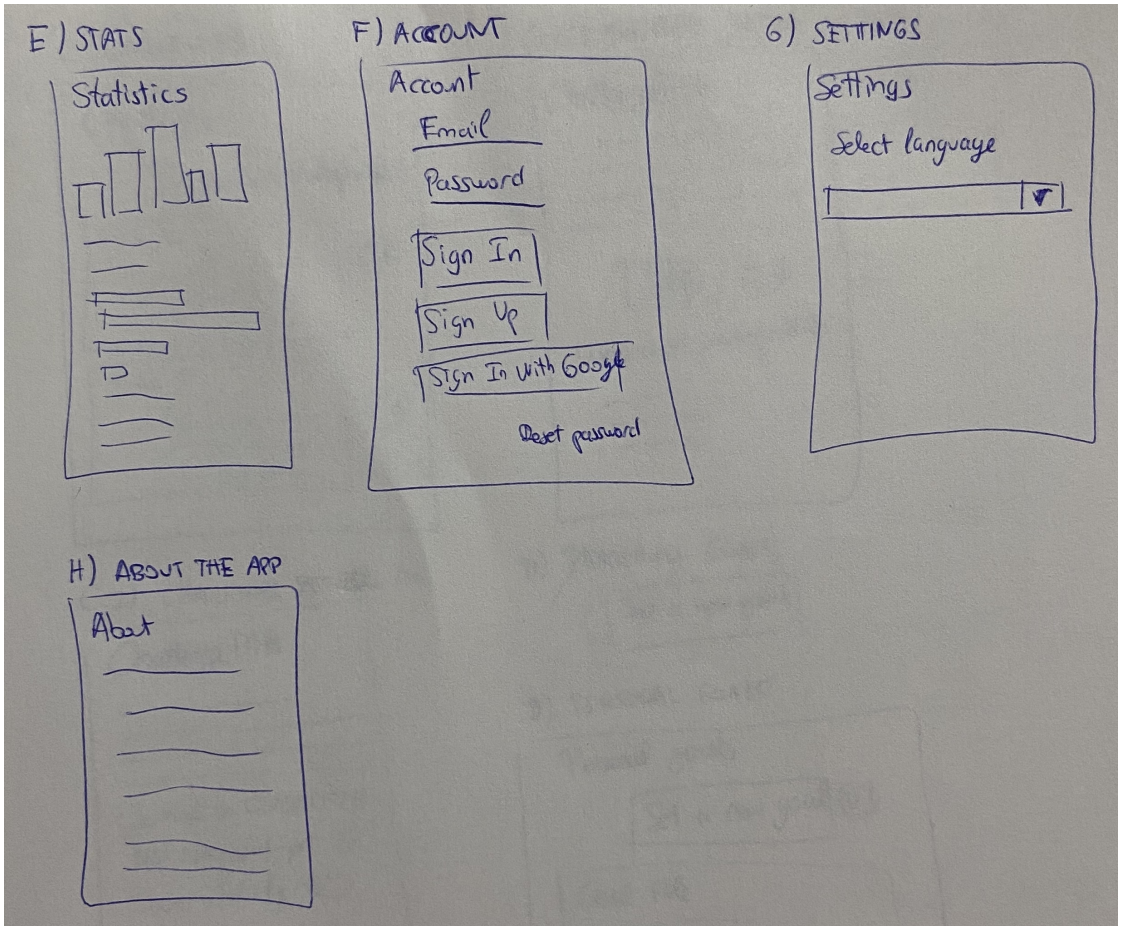

Figura 17: Bocetos, III

2.4.2. Prototipo de alta fidelidad

En un prototipo de alta fidelidad ya se busca entrar en un mayor grado de detalle, siendo más fiel a cómo será la interfaz de usuario en el producto real (de ahí la denominación de este tipo de prototipos). Además, idealmente se usan herramientas que nos permiten determinar de forma clara cuál será la navegación entre las distintas pantallas. Para este fin se ha usado la herramienta Justinmind, la cual nos permite exportar el proyecto en formato HTML, lo que resulta muy conveniente, ya que no hace falta tener ningún software para ejecutar el prototipo, basta con un navegador web.

Puesto que se va a entregar el prototipo aparte, no se van a incluir las capturas de las pantallas en el presente documento. No obstante, si que se incluye una pantalla no accesible mediante el prototipo pero que está contenida en los archivos de importación. Esta pantalla representa un listado de noticias representadas en un iPad y, por tanto, tal y como se ha indicado en la planificación, empleando la vista en tipo maestro-detalle. Esta pantalla es la que se muestra en la figura 18.

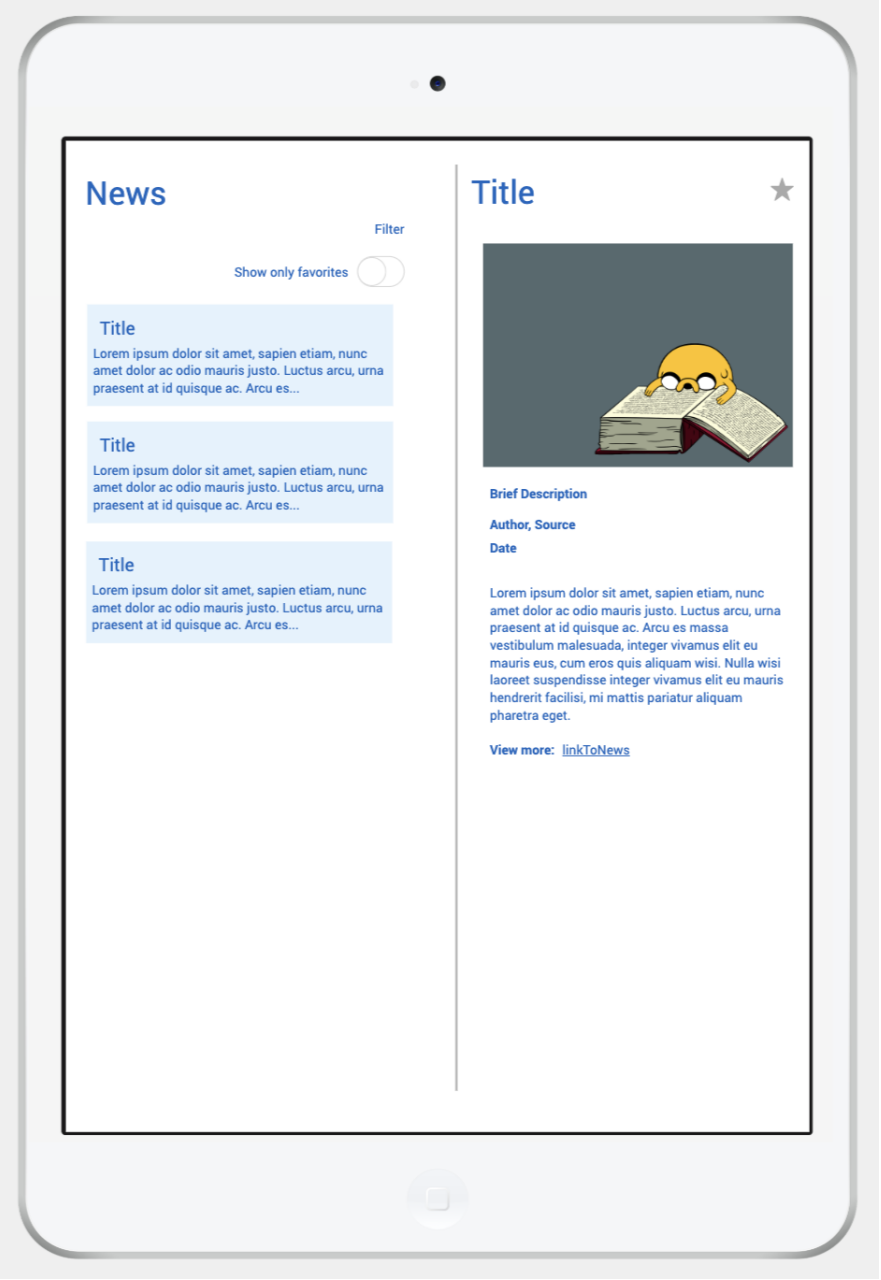

Figura 18: Ejemplo de visualización maestro-detalle

Por otra parte, cabe destacar que para volver atrás se puede usar un gesto de deslizamiento horizontal o *swipe* hacia la derecha, así como pulsar el botón *Back* que se encuentra en la esquina superior izquierda.

Para poder ejecutar el prototipo basta con abrir el archivo index.html que se encuentra dentro de la ruta **Prototype/AvidReadersPrototype**.

2.4.3. Evaluación del prototipo

Una cosa que no se puede perder nunca de vista al realizar una aplicación es el usuario. Por ello se deben desarrollar productos teniendo siempre presente la usabilidad de los mismos. Puesto que el prototipo es la principal fuente de la aplicación final en lo que a usabilidad se refiere, es aquí donde debemos empezar a plantearnos temas de usabilidad y cómo afrontarlos. Así, se deben escoger qué técnicas se utilizarán para evaluar esta dimensión del prototipo.

La evaluación del prototipo se ha llevado a cabo apoyándonos en los diez principios heurísticos de Jakob Nielsen [11], una referencia en este campo. Estos principios definen unas directivas a seguir si se quiere mantener un buen nivel de usabilidad en la aplicación. El hecho de que se cumplan estos principios en la medida de lo posible nos da una garantía de que el nivel de usabilidad es aceptable, si no bueno.

A continuación se describen *grosso modo* los 10 principios de usabilidad de Nielsen:

1. **Visibilidad del estado del sistema**: El usuario siempre debe estar informado de lo que está pasando en el sistema mediante mensajes apropiados recibidos en un tiempo aceptable.
2. **Emplear el lenguaje del usuario**: El sistema deberá emplear expresiones comunes y en lenguaje natural, no usar códigos extraños que no pertenezcan al dominio de los usuarios.
3. **Libertad y control para los usuarios**: El usuario, puesto que es humano y está sujeto a cometer errores, podrá en cualquier momento volver atrás y cancelar acciones realizadas.
4. **Consistencia y estándares**: Los diversos términos empleados en el sistema deben ser consistentes, es decir, mantener el mismo significado a lo largo de la aplicación.
5. **Prevención de errores**: Siempre es mejor prevenir errores que explicar luego muy detalladamente en qué consistían. Para paliar estos errores es una muy buena idea pedirle a los usuarios confirmaciones cuando vayan a realizar alguna acción potencialmente peligrosa o que pueda generar errores.
6. **Reconocimiento antes que memorización**: Se debe minimizar la memorización que tenga que llevar a cabo el usuario para poder usar la aplicación. De este modo, es mejor introducir elementos fácilmente reconocibles por el usuario, mensajes explicativos e información varia antes que optar por que los usuarios recuerden cosas. Así, la información de ayuda al usuario debe estar siempre accesible.
7. **Flexibilidad y eficiencia**: Los atajos permiten que el usuario experto trabaje mucho más rápido sin ponerle trabas a los usuarios menos experimentados, los cuales también se pueden aprovechar de ellos.
8. **Diseño minimalista y estético**: Se debe mostrar solo aquella información relevante y necesaria, ya que todo lo que se añada a mayores compite con los datos realmente importantes a nivel de visibilidad.
9. **Ayuda para que los usuarios reconozcan, diagnostiquen y se recuperen de los errores**: Los mensajes de error deberían expresar de forma sencilla y comprensible por el usuario el problema que se acaba de dar, sugiriendo una solución cuando sea posible.

10. **Ayuda y documentación**: Sería preferible que el sistema se pudiese usar sin necesidad de documentación, pero podría ser necesaria. Dicha documentación debería ser fácil de buscar y encontrar en la aplicación, además de ser concisa.

Lo que se ha hecho en la práctica ha sido revisar cada uno de los puntos descritos anteriormente para detectar posibles errores a lo largo de la creación del prototipo. De este modo, por ejemplo, no se ha detectado que no se había incluido un botón de atrás hasta la última iteración de prototipado. Se había introducido el deslizamiento horizontal para retroceder en la navegación, pero pensando en términos de usabilidad se llegó a la conclusión de que solamente este medida podría resultar insuficiente para cumplir el principio número 3 de Nielsen (ser capaz de deshacer un error). Este problema se hace aún más evidente si el usuario de la aplicación no ha tenido mucha interacción con dispositivos iOS y está acostumbrado, por ejemplo, a dispositivos Android, en los cuales esta forma de volver atrás no es tan común. Así, el hecho de revisitar los principios a lo largo del desarrollo del prototipo nos ayuda a detectar errores de usabilidad y, a su vez, modificar el prototipo y seguir mejorando.

Otro punto que se ha tenido mucho en cuenta es el 8, pues la interfaz se ha creado con un diseño minimalista para evitar sobrecargarla y dificultar al usuario su interacción con ella.

3. Implementación y Pruebas

En esta sección se detallarán las decisiones tomadas en lo concerniente a la etapa de implementación del producto descrito en las secciones anteriores. Así, se justificarán diversos cambios introducidos con respecto a lo establecido en las etapas anteriores y, sin entrar en los detalles de código, se dará una visión general del funcionamiento de la aplicación móvil. Por otra parte, también se hará mención a las pruebas a realizar para comprobar que se han alcanzado los requisitos establecidos en la memoria y que todos los casos de uso se pueden llevar a cabo.

3.1. Cambios o modificaciones producto de la reevaluación

Antes de proceder a explicar las decisiones tomadas en el proceso de implementación, se deben indicar aquellos cambios que se han llevado y que no concuerdan con lo descrito anteriormente en este documento. Estos cambios se derivan de errores no identificados en un primer momento, decisiones alternativas que resultan ser mejores que las presentadas originalmente y de carencias que no se habían identificado y las cuales ha sido necesario suplir para garantizar un buen funcionamiento de la aplicación. Dichos cambios se describen a continuación:

- Se ha introducido un nuevo atributo en la entidad correspondiente a la información de un libro, *BookEntity*, para registrar el número de páginas que este tiene. Este atributo, el cual se ha definido como *pagesCount*, no se había definido en la parte de diseño y es imprescindible para poder actualizar los retos personales de un usuario.
- De un modo similar, no es posible gestionar correctamente los retos del usuario sin tener en cuenta su fecha de creación, un parámetro que no se había reflejado en un primer momento

en la entidad *GoalEntity*. Este parámetro es muy importante, ya que determina el inicio del ciclo de un objetivo personal. Así, si se establece, por ejemplo, un objetivo personal de leer 200 páginas semanalmente, es imprescindible saber cuándo se ha creado el objetivo para poder determinar cuando se termina una semana y comienza la siguiente.

- Relacionado con lo anterior, en el prototipo, tanto de baja como de alta fidelidad, no se había mostrado el campo que indica cuántos ítems ha leído el usuario. Este campo se ha añadido.
- Respecto a los retos presentados por la aplicación, el usuario se puede unir a uno de forma pública o privada. En el prototipo se había reflejado un control de tipo toggle para cambiar este parámetro, pero en SwiftUI un toggle no lanza una acción al pulsarlo. De este modo, se ha decidido añadir un botón para actualizar este valor en la base de datos de Firebase que se muestra en cuanto el valor cambia.
- En lo que respecta a los colores de la interfaz, se había comentado que se iban a seguir los principios heurísticos de Nielsen, entre los cuales hay uno que defiende la consistencia en toda la aplicación. No obstante, esto no era así en el prototipo de alta fidelidad para ciertos listados, ya que los colores estaban invertidos (letras azules oscuras sobre un fondo azul claro y viceversa). Lo que se ha hecho es unificar todos los listados, tanto para light mode como para dark mode. Así, para light mode las letras serán claras sobre una celda oscura y del revés para dark mode.
- En cuanto a las noticias, se había comentado con anterioridad que las favoritas se guardarían en el dispositivo para poder visualizarlas en modo offline o sin conexión a la red. Se había comentado que se haría con una base de datos Realm, pero posteriormente se vio que la propia base de datos de Firebase ya soporta esta funcionalidad. Dado que se consideró que utilizar una segunda base de datos pudiendo utilizar un único sistema era volver a inventar la rueda, las noticias favoritas se pueden visualizar sin conexión utilizando Firebase, sin necesidad de echar mano de Realm.
- En un primer momento, en el prototipo se mostraba una única sección para libros y autores, dividida en 2 pestañas o tabs. No obstante, una vista de este tipo, denominada *TabView*, ha de ser la raíz de la navegación de la aplicación, puesto que en caso contrario se pueden producir errores y crashes inesperados. Puesto que la navegación raíz de la aplicación está en la pantalla de inicio, con todas las opciones en forma de botones, no se pudo emplear el *TabView* sin tener errores. La solución por la que se optó fue separar este apartado en dos secciones: Libros y Autores. De este modo la opción del menú da lugar a dos y en vez de una vista compartida, cada una de ellas se corresponde con una pantalla diferente.

3.2. Implementación

En este apartado se dará una explicación de cómo ejecutar la aplicación para poder evaluarla, se describirá brevemente cómo se ha organizado el código fuente y se indicarán ciertos problemas y/o bugs que se han detectado y, que en un proyecto real, se podrían abordar en versiones subsecuentes.

3.2.1. Ejecución

Para ejecutar el código será necesario hacerlo mediante un simulador de Xcode, empleando un dispositivo con sistema operativo Mac OS Catalina. Concretamente, la versión más reciente en la que se

ha ejecutado el código es la 10.15.4. Debido a que el alumno no dispone de cuenta de desarrollador de Apple, no es posible ejecutar el código en dispositivos móviles físicos, a diferencia de lo que ocurre con aplicaciones Android, las cuales pueden ser ejecutadas en dispositivos físicos sin necesidad de tener cuenta de desarrollador.

Tras abrir Xcode, se ejecuta `Product>Run` o se emplea la combinación de teclas `Cmd+R` para ejecutar la aplicación en el simulador. Para cambiar el simulador en el que ejecutar la aplicación, en la barra superior puede elegirse el simulador que se quiera (véase figura 19). El resultado de ejecutar la aplicación puede verse en la figura 20. A partir de este momento el usuario podrá interaccionar con la aplicación como desee.

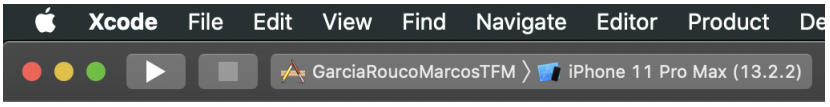

Figura 19: Seleccionador simulador

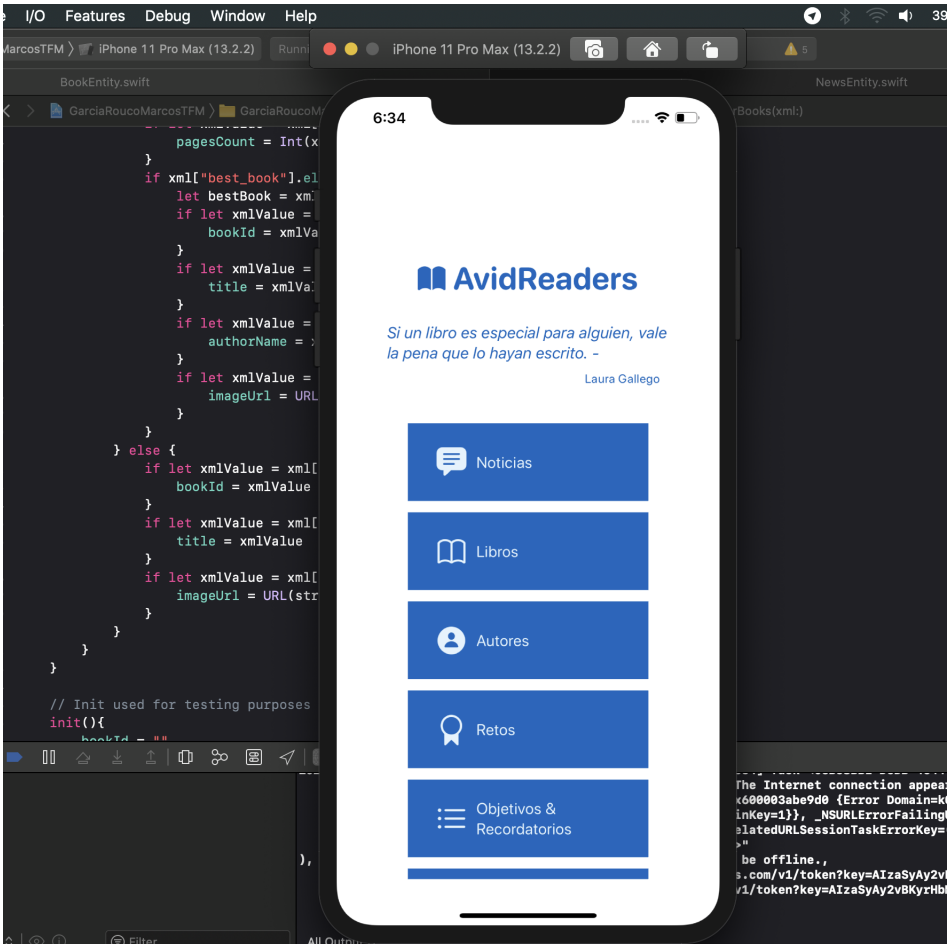

Figura 20: Ejecución de la aplicación

Cabe destacar que si se quiere probar que la aplicación está adaptada para el dark mode, se ha de ir a la aplicación nativa Settings y bajo el menú Developer, activar el modo oscuro.

3.2.2. Organización del código fuente

En esta sección se proporcionará una breve descripción de cómo se ha estructurado el código. La estructuración del código fuente es algo muy importante en los proyectos de desarrollo, sobre todo a la hora de volver a tocar un proyecto antiguo o cuando alguien nuevo pasa a formar parte del equipo de desarrollo. Si el código está bien estructurado y hay consistencia en dicha estructuración (todas las clases del mismo tipo se almacenan en el mismo paquete o los archivos se organizan por módulos de desarrollo, por ejemplo), será mucho más fácil interaccionar con el código y encontrar lo que se busca.

La estructura base del proyecto puede verse en la figura 21. Como se puede observar, los distintos elementos del código se han separado por capas, dedicando una carpeta a la vista y ViewModel, otra carpeta al modelo y luego carpetas para constantes, utilidades de red y extensiones a clases para añadir nuevas funcionalidades. Por otra parte, en *Assets* se han definido los colores empleados en la aplicación y *LaunchScreen.storyboard* se corresponde con la launch screen o primera pantalla con la que se encuentran los usuarios al abrir la aplicación.

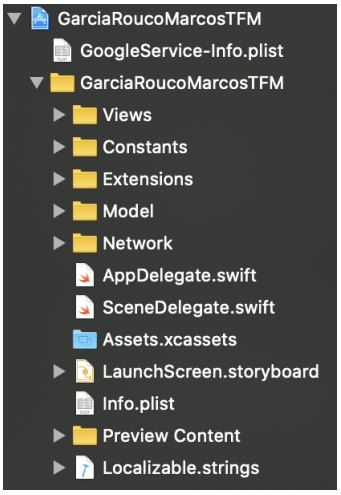

Figura 21: Estructura base del código

Por su parte, en la figura 22 se puede ver cómo se estructura a su vez la carpeta View. En ella los elementos se separan por ámbitos (hay una carpeta para la vista de Noticias y las vistas incluidas, por ejemplo), ademas de haber carpetas para utilidades o vistas empleadas en varios ámbitos (por ejemplo, la vista con 5 estrellas para dar una valoración se utiliza tanto para libros como para autores).

Lo mismo ocurre para las carpetas Model y Network (véanse figuras 23 y 24, respectivamente), donde las clases se separan por su naturaleza o funcionalidad. No obstante, los nombres de las carpetas son descriptivos, lo que facilita la acción de buscar ciertas clases o directorios donde añadir nuevos ficheros.

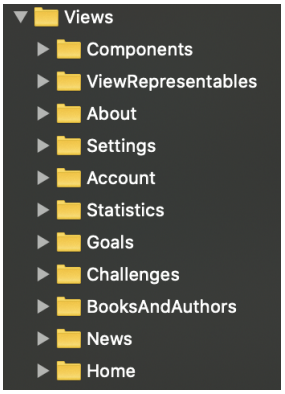

Figura 22: Estructura de la carpeta Views

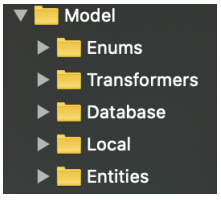

Figura 23: Estructura de la carpeta Model

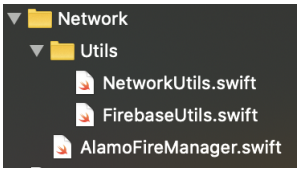

Figura 24: Estructura de la carpeta Network

En lo que se refiere al código en si, las clases en las que se hacen llamadas o peticiones HTTP constan de funciones y en ocasiones de atributos, por lo que no hay nada que destacar en especial. No obstante, las clases correspondientes al ViewModel si tienen una estructura más definida. Estas clases se estructuran siempre de la siguiente manera:

1. Primero se muestran las variables de la clase y el body, el cual se corresponde con lo que se presentará en la pantalla del dispositivo físico o simulador.

2. Luego, en caso de que sea necesario crear y llamar a algún método, se crea una extensión a la clase en cuestión y ahí se añaden dichos métodos.

3. Por último, se añade una preview, la cual nos permite visualizar la pantalla actual sin necesidad de construir o ejecutar la aplicación, siempre y cuando el sistema operativo instalado sea Mac OS Catalina.

En la figura 25 se muestra la estructura descrita, mientras que en las figuras 26, 27 y 28 se muestran partes de estas secciones para enseñar un poco de cómo están conformadas.

```
//
//  Created by Marcos García Rouco on 19/03/2020.
//  Copyright © 2020 Marcos García Rouco. All rights reserved.
//

import SwiftUI
import Alamofire
import AlamofireObjectMapper
import ObjectMapper

/**
 View to request and display news
 */
struct NewsView: View {•••}

extension NewsView {•••}

struct NewsView_Previews: PreviewProvider {
    static var previews: some View {
        NavigationView {
            NewsView()
        }
    }
}
```

Figura 25: Estructura básica de una clase correspondiente al ViewModel

```
struct NewsView: View {
    // Array that holds the news retrieved from the NewsAPI API
    @State private var newsArray:Array<NewsEntity> = Array()
    // Variable that determines whether the activity indicator is shown or not
    @State private var isShowing:Bool = true
    // Variable that determines whether the user's favorite news are shown
    @State private var showFavorites:Bool = false
    // Variable used to manage the pagination of news
    @State private var page: Int = 1
    // Alert variables
    @State private var showAlert:Bool = false
    @State private var alertTitle:String = ""
    @State private var alertMessage:String = ""

    // Environment object to obtain the app's current language, necessary to determine what'll be search language
    @EnvironmentObject var appLanguage:AppLanguage
    @EnvironmentObject var globalProperties:GlobalProperties

    var body: some View {
        LoadingView(isShowing: $isShowing) {
            GeometryReader { geometry in
                List {
                    // The ZStack in conjunction with the EmptyView are used to remove the chevron painted on the
                        part of a list
                    ZStack {
                        HStack{
                            Spacer()
```

Figura 26: ViewModel: Parte del body

```swift
extension NewsView {
    // Method to obtain the current date so as to add it as part of the request's
        parameters
    func addOnlyTodayNewsParam() -> String? {
        // We try to retrieve the value from the UserPreferences, which in turn makes a
            call to UserDefaults
        guard let onlyTodayNews = UserPreferences.retrieveValue(key:
            Constants.Params.kTodayNews) as? Bool, onlyTodayNews else { return nil}
        // If the value is retrieved, then it's returned after being formatted
        var dateFormat:DateFormatter {
            let formatter = DateFormatter()
            formatter.dateFormat = "yyyy-MM-dd"
            return formatter
        }
        return dateFormat.string(from: Date())
    }
```

Figura 27: ViewModel: Parte de la extensión

```swift
struct NewsView_Previews: PreviewProvider {
    static var previews: some View {
        NavigationView {
            NewsView()
        }
    }
}
```

Figura 28: ViewModel: Parte de la preview

3.2.3. Problemas o consideraciones

SwiftUI es un lenguaje de programación bastante reciente, ya que apareció a mediados de 2019. De este modo, aún cuenta con bastantes errores y bugs que Apple aún no ha subsanado. A estos problemas se suman los derivados de las modificaciones introducidos por iOS 13 el pasado septiembre, que han dado lugar a cambios considerables en la presentación de las vistas, por ejemplo. Además, Mac OS Catalina también es reciente y se ha informado de varios bugs. Todo ello en conjunto hace que la aplicación presente varios bugs al ejecutarla en el simulador, mientras que al ejecutarla en un dispositivo físico no se producirían (o al menos eso es lo que se deduce al leer y obtener información sobre estos bugs). Cabe destacar que las funcionalidades descritas en este documento se han probado en dispositivos con versiones iOS de **13 a 13.2.2** instaladas. No se puede garantizar que no haya errores para versiones posteriores (arreglar estos hipotéticos bugs sería lo que se haría en fixes, subversiones o versiones sucesivas a esta en un proyecto real).

A continuación se listan estos problemas para que el profesorado, a la hora de evaluar la aplicación los tenga en cuenta y esté en sobre aviso de que el alumno es consciente de ellos:

- Cuando se selecciona un ítem en una lista, se va al detalle, se vuelve de nuevo a la lista y se selecciona una vez más el mismo elemento, en muchas ocasiones no pasa nada y es necesario seleccionar un ítem distinto. No obstante, este es un error que toda la comunidad se ha encontrado y que está bien identificado y documentado, por ejemplo en StackOverflow [12] o

RayWenderlich [13] (véase figura 29 para ver la parte de la última página web en la que se habla del error).

- La rotación del simulador de iPhone (no así de iPad) en la pantalla principal o *HomeView* hace que la pantalla aparezca en blanco, no así en otras pantallas. De hecho, en las otras pantallas al rotar el iPhone incluso se puede ver la utilización del patrón master-detail.
- La funcionalidad de ver un libro en iTunes se podía llevar a cabo correctamente al tener instalado Mac OS Catalina 10.14. No obstante, al actualizar a 10.15.4 esta funcionalidad dejó de funcionar, puesto que Safari indica que no se puede abrir la url (véase figura 30). Ese no es el caso si se abre exactamente la misma url en un dispositivo físico o en el propio ordenador. Se espera que este bug se subsane lo antes posible, pero la funcionalidad podría seguir sin funcionar en el momento de la entrega del presente documento.

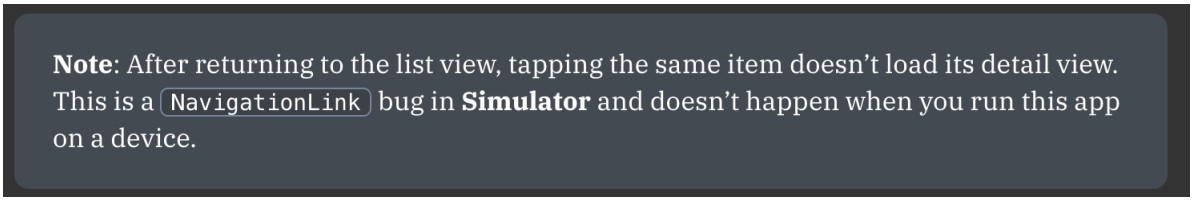

Figura 29: Bug de selección del mismo ítem de una lista dos veces consecutivas

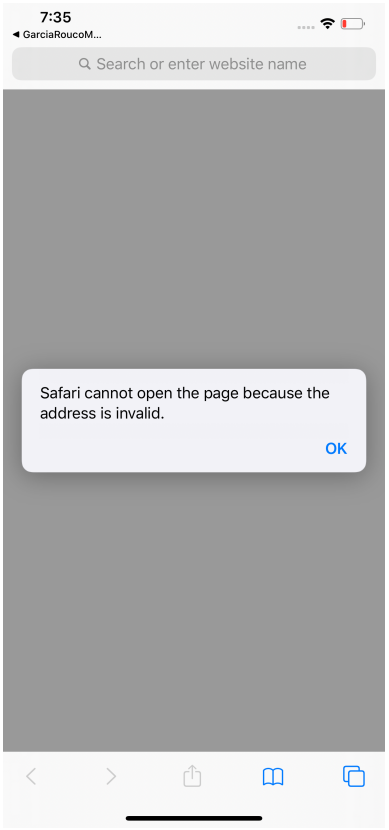

Figura 30: Bug de apertura de url de un libro en iTunes

3.3. Pruebas

La realización de los casos de uso en la etapa de diseño nos proporciona con las pruebas que la aplicación debe pasar para considerar el proyecto un éxito. Esto es porque en los casos de uso se definieron criterios de validación, es decir, condiciones que se tenían que dar para cada caso de uso para considerar que se habían completado.

De este modo, a excepción del caso de uso de la visualización de un determinado libro en iTunes, el cual funcionaba antes de la actualización del sistema operativo del equipo de desarrollo, todos los otros casos de uso han pasado o cumplido con los criterios de validación. Esto constituye que la aplicación ha pasado los tests unitarios, puesto que que cada caso de uso se corresponde con una funcionalidad llevada a cabo en una pantalla. No obstante, esto no es del todo cierto, porque cada uno de estos casos de uso requiere en muchos casos la ejecución de un método de otra clase o la realización de una llamada HTTP. Así, pasar cumplir estos criterios de validación significa que todos estos elementos funcionan, demostrando que la aplicación cumple con los requisitos establecidos y que se han probado de manera correcta.

Por otra parte, se ha probado a llevar a cabo todos los casos de uso uno detrás de otro y en diferentes secuencias sin cerrar la aplicación o salir de ella para demostrar que no se generan estados de error o condiciones similares.

3.4. Utilización de la aplicación

Para facilitar el uso de la aplicación se ha proporcionado un manual de usuario a fin de que el usuario sepa cuáles son las posibilidades de la aplicación y como llevarlas a cabo. Dicho manual de usuario se corresponde con un fichero pdf llamado **GarciaRoucoMarcos_ManualUsuario.pdf**.

En lo que respecta a poder ejecutar la aplicación, primero ha de ejecutarse la instrucción `pod install` en el directorio en el que se encuentra el fichero Podfile para que se instalen los pods o librerías necesarios para el correcto funcionamiento del proyecto. En el caso de que CocoaPods [14] aún no se haya instalado en el proyecto, se ha de ejecutar primero `sudo gem install cocoapods`.

4. Conclusiones

Una vez finalizado el presente trabajo de fin máster, es hora de realizar una auto-evaluación de todas las tareas llevadas a cabo, así como situar el TFM en el contexto del máster realizado en la UOC.

En primer lugar, se considera que el hecho de tener que dar con una idea y desarrollar una solución para ella ha sido un proceso muy constructivo e instructivo, puesto que el alumno ha tenido que llevar a cabo todas las fases que se desempeñarían en un proyecto real. Así, se ha tenido que dar forma a una planificación y se ha podido ver de primera mano que algo que parece trivial (estimar cuanto tiempo va a llevar desarrollar una simple pantalla, por ejemplo) puede llevar mucho más tiempo del esperado (la pantalla era trivial en un lenguaje de programación distinto pero el alumno no ha tenido en cuenta un cambio de paradigma o de elementos de interfaz de usuario, por ejemplo). Posteriormente se ha tenido que crear un catálogo de requisitos que, en un primer momento, parecía completo e inamovible. No obstante, a lo largo del proyecto se ha visto como estos requisitos se han tenido que ir cambiando, otros han perdido su justificación y también ha sido necesario añadir algunos nuevos. Esto nos lleva a entender la aparición de ciclos de vida no monolíticos, pues se ha experimentado de primera mano que un proyecto está sujeto a cambios, incluso si no hay una parte cliente como tal (como es el caso del presente trabajo, en donde el propio alumno se ha podido percatar de carencias y fallos).

Aparte de la experiencia obtenida al realizar el trabajo de fin de máster para comprender todo el trabajo y complejidad que supone un proyecto real, también se considera que esta ha sido una muy buena oportunidad para practicar y poner en uso capacidades y habilidades adquiridas a lo largo de todo el máster. Así, en el presente trabajo se ha tenido que echar mano de una tecnología nueva como es SwiftUI. No obstante, el máster ya había preparado al alumno para reaccionar contra nuevas tecnologías y aprenderlas poco a poco echando mano de la documentación oficial y otros sitios web de renombre. Estas otras tecnologías con las que el usuario no había tenido contacto previo a la realización del máster incluyen a Swift, Kotlin e Ionic.

En términos más generales, se considera que el máster de desarrollo de aplicaciones móviles, cuya guinda está representada por el presente trabajo de fin de máster, ha sido una experiencia extremadamente positiva en la vida y carrera profesional el alumno. Tanto es así que el alumno se encuentra trabajando de desarrollador de aplicaciones móviles para dispositivos iOS gracias a estar cursando este máster. De este modo, el alumno desea aprovechar las conclusiones del trabajo de fin de máster para agradecer a todo el profesorado y comunidad de la UOC por su apoyo, ayuda e instrucción, las cuales han sido fundamentales para obtener dicho puesto de trabajo. Así, quizás la conclusión más importante obtenida de la realización de este máster es afianzar la idea de que es posible formarse de modo totalmente online de una manera efectiva y con calidad.

Bibliografía

[1] *Google Fit*. [Consultado el 25 de febrero de 2020 a las 09:23]. Disponible en https://www.google.com/fit/.

[2] *Goodreads | Meet your next favorite book*. [Consultado el 25 de febrero de 2020 a las 09:55]. Disponible en https://www.goodreads.com/.

[3] *App Store - Support - Apple Developer*. [Consultado el 25 de febrero de 2020 a las 11:05]. Disponible en https://developer.apple.com/support/app-store/.

[4] *Firebase*. [Consultado el 25 de febrero de 2020 a las 16:56]. Disponible en https://firebase.google.com/.

[5] *raywenderlich.com | High quality programming tutorials: iOS, Android, Swift, Kotlin, Flutter, Server Side Swift, Unity, and more!*. [Consultado el 25 de febrero de 2020 a las 17:21]. Disponible en https://www.raywenderlich.com/.

[6] *Incremental Life Cycle - Project Management Knowledge*. [Consultado el 26 de febrero de 2020 a las 18:34]. Disponible en https://project-management-knowledge.com/definitions/i/incremental-life-cycle/.

[7] *Home | Projectlibre*. [Consultado el 29 de febrero de 2020 a las 14:10]. Disponible en https://www.projectlibre.com/.

[8] Project Management Institute, *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)*, 5ª edición.

[9] *News API - A JSON API for live news and blog articles*. [Consultado el 15 de marzo de 2020 a las 20:05]. Disponible en https://newsapi.org/.

[10] *Free prototyping tool for web & mobile apps - Justinmind*. [Consultado el 16 de marzo de 2020 a las 19:52]. Disponible en https://www.justinmind.com/.

[11] *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [Consultado el 18 de marzo de 2020 a las 19:44]. Disponible en https://www.nngroup.com/articles/ten-usability-heuristics/.

[12] *Can't select same row twice in SwiftUI - Stack Overflow*. [Consultado el 18 de abril de 2020 a las 19:15]. Disponible en https://stackoverflow.com/questions/59061298/cant-select-same-row-twice-in-swiftui.

[13] *iOS Accessibility in SwiftUI Tutorial Part 1: Getting Started | raywenderlich.com*. [Consultado el 18 de abril de 2020 a las 19:28]. Disponible en https://www.raywenderlich.com/7180554-ios-accessibility-in-swiftui-tutorial-part-1-getting-started.

[14] *CocoaPods Guides - Getting Started*. [Consultado el 3 de mayo de 2020 a las 18:46]. Disponible en https://guides.cocoapods.org/using/getting-started.html.