Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnología de la Información

RoomieFinder: aplicación iOS para la Búsqueda de Compañeros de Piso en el Entorno Estudiantil

Guillermo Rodríguez Alonso

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnología de la Información

RoomieFinder: aplicación iOS para la Búsqueda de Compañeros de Piso en el Entorno Estudiantil

Autor: Guillermo Rodríguez Alonso

Tutor: Joaquín Adiego Rodríguez

No importa cuántas veces caigas,
lo importante es levantarte una vez más y seguir adelante con valentía.
Tony Stark, Iron Man 3

Agradecimientos

Han sido muchas personas las que han ayudado a que este proyecto haya salido adelante. En primer lugar agradecérselo a mi tutor ya que con sus constantes revisiones y correcciones tanto la aplicación como la memoria ha ido por el camino adecuado.

Me gustaría agradecer también a mis padres y hermanos, por haber estado siempre apoyándome y acompañándome desde el primer día, pero quiero dar un énfasis sobre mi hermano David, por ser mi referente desde el primer momento.

Por último, a todos mis amigos que han estado conmigo durante la carrera. A los que conocía antes de esta etapa y a los que he tenido la suerte de cruzarme con ellos durante estos años, porque gracias a ellos he podido llegar donde estoy ahora mismo.

Resumen

En este Trabajo de Fin de Grado se ha desarrollado una aplicación móvil llamada RoomieFinder, diseñada para facilitar la búsqueda de compañeros de piso. Hasta este momento, no existía una plataforma que permitiera a los usuarios interactuar con anuncios de búsqueda de compañeros de piso de manera efectiva y segura. El desarrollo de esta aplicación involucró la creación de una interfaz de usuario intuitiva, la implementación de funciones de búsqueda avanzadas y la integración de un sistema de mensajería para facilitar la comunicación entre los usuarios. Como resultado del proyecto, se ha obtenido una aplicación móvil fácil de usar y segura, que simplifica el proceso de búsqueda y la interacción entre personas que buscan y ofrecen compartir piso.

Palabras clave: iOS, Swift, SwiftUI, Aplicación móvil.

Abstract

In this Final Degree Project we have developed a mobile application called RoomieFinder, designed to facilitate the search for flatmates. Until now, there was no platform that allowed users to interact with roommate search ads in an effective and safe way. The development of this application involved the creation of an intuitive user interface, the implementation of advanced search functions and the integration of a messaging system to facilitate communication between users. As a result of the project, an easy-to-use and secure mobile application has been obtained, which simplifies the search process and the interaction between people looking for and offering to share a flat.

Key words: iOS, Swift, SwiftUI, mobile aplication.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

A lo largo de los años, el compartir pisos con múltiples compañeros ha ido ganando importancia, y más, en una sociedad cada vez más abierta y plural. La búsqueda de estos se ha convertido en una ardua tarea para todos aquellos que desean empezar una nueva experiencia, sobre todo, para lo más jóvenes en situación de cambio de ciudad por asuntos universitarios, así como aquellos que comienzan el desempeño de un nuevo puesto de trabajo, todos ellos con un punto en común, el abandono de su vivienda habitual para la entrada en una nueva.

A menudo se ha recurrido a métodos o técnicas convencionales como anuncios en distintos periódicos, folletos, redes sociales, foros de vivienda, páginas web o, incluso, a través de recomendaciones entre conocidos. En un escenario como este, caracterizado por la alta demanda de búsqueda de pisos y, con ellos, de compañeros para convivir, son muchas las soluciones que han surgido con el objetivo de optimizar y facilitar dicho cometido, aún así, aún se encuentran con numerosos límites que no permiten alcanzar su total efectividad.

Como respuesta a esta necesidad, el presente Trabajo de Fin de Grado propone la creación, diseño, desarrollo e implementación de una aplicación móvil para dispositivos iOS denominada “RoomieFinder”, cuyo objetivo sea lograr satisfacer las demandas de aquellos que presentan la necesidad objeto de estudio.

1.1 Motivación

En la época actual, marcada por las facilidades que presenta la globalización, la búsqueda de compañeros de piso para el público joven en diversos contextos ha experimentado una transformación significativa impulsada esencialmente por los avances tecnológicos. Aunque los métodos “tradicionales” aún están en uso, la digitalización ha introducido múltiples posibilidades y una serie de nuevas herramientas que tienen el potencial de agilizar y mejorar de manera significativa este proceso, como se puede comprobar a través de las redes sociales.

Las redes sociales surgen como estructuras formadas en Internet que permite una comunicación entre personas desde cualquier parte del mundo a partir de intereses o valores comunes, en dicha situación se encuadran los demandantes de compañeros de piso, y, es que, se hace habitual la búsqueda de convivientes a través de algo tan simple como una publicación en Instagram o un "tweet" en X con detalles sobre sus necesidades de alojamiento, preferencias de convivencia y datos de contacto. Pero ello también deja entrever una de las grandes dificultades a las que se enfrentan los medios dedicados a este objeto, la dispersión de la información.

Los datos más relevantes se distribuyen con mayor rapidez entre contactos y publicaciones, lo que complica la organización y seguimiento de oportunidades que surgen sobre alojamientos disponibles, a ello se le añade la carencia de estructura ordenada y personalizada lo que puede dar lugar a la formación de conexiones incompatibles y el surgimiento de convivencias insatisfactorias para los estudiantes.

A partir de esta situación, aparece una alta urgencia de desarrollar soluciones innovadoras que ofrezcan una experiencia de búsqueda de coinquilinos más completa, ajustada a los requisitos y, todo ello, centralizado en torno a una única aplicación. Este enfoque, fundamentado en la novedad desde un punto de vista tecnológico busca, no solo superar aquellas limitaciones existentes hoy en día, sino, también, poder ofertar una plataforma segura, eficaz y eficiente, que facilite la creación de conexiones entre potenciales usuarios en este proceso.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación para dispositivos móviles de iOS denominada "RoomieFinder", que facilite la búsqueda de compañeros de piso en el entorno estudiantil, proporcionando una plataforma eficiente y personalizada para conectar a estudiantes con necesidades, estilos de vida y gustos compatibles.

Los objetivos específicos del proyecto son los siguientes:

- Analizar los requisitos y necesidades de los estudiantes en relación con la búsqueda de compañeros de piso, con el fin de identificar los factores más importantes que influyen en su decisión y sus inquietudes.
- Crear una interfaz de usuario que sea fácil de usar e intuitiva, considerando las preferencias y comportamientos de lo estudiantes para mejorar la experiencia de navegación en la aplicación.
- Implementar un sistema de búsqueda y filtrado avanzado que permita a lo usuarios encontrar compañeros de piso básandose en diversos criterio, tales como ubicación, intereses, hábitos, estilo de vida y presupuesto.
- Incorporar un sistema de mensajería integrado que posibilite a los usuarios comunicarse entre sí de manera privada y segura.

1.3 Alcance

La aplicación esta específicamente orientada a estudiantes de grado, postgrado, participantes de intercambio y otros individuos que residan temporalmente en una cuidad o en una región concreta durante su etapa académica. Este grupo abarca tanto a estudiantes nacionales como internacionales que se trasladan a nueva ubicación para continuar sus estudios, buscando una solución de alojamiento adecuada, segura y compatible con sus necesidades y preferencias. Además, la aplicación también esta dirigida a jóvenes profesionales que están dando sus primeros pasos en el ámbito laboral. Estos individuos, recién graduados o en las primeras etapas de su carrera profesional, pueden encontrar en "RoomieFinder" una alternativa de alojamiento económica y socialmente beneficiosa. Al compartir piso con una persona con intereses y horarios compatibles, los jóvenes profesionales pueden disfrutar de un ambiente de convivencia positivo y enriquecedor, facilitando así su adaptación a la vida laboral y social en una nueva ciudad.

En resumen, "RoomieFinder" busca atender las necesidades de dos grupos principales: estudiantes universitarios en diversas etapas de su formación académica y jóvenes profesionales que están iniciando su carrera laboral. La aplicación se centra en proporcionar una plataforma centralizada y eficaz que mejore la experiencia de búsqueda de compañeros, ofreciendo opciones más personalizadas, completas y seguras.

1.4 Contenidos de la memoria

La memoria se organiza en capítulos, que contienen los diversos apartados en los que se lleva a cabo el trabajo:

- En el Capitulo 1, se presenta una introducción detallada sobre el problema abordado, exponiendo la necesidad que motiva el desarrollo propuesto. Además, se describen los objetivos del trabajo, su alcance, el estado actual de la situación y otros elementos pertinentes.
- En el Capitulo 2 se realiza un análisis minucioso de diversas aplicaciones móviles que ofrecen características similares a las del proyecto planteado. Se examinan sus características, prácticas destacadas y áreas de mejora con el propósito de obtener lecciones valiosas para el desarrollo de una aplicación innovadora y eficiente que satisfaga las demandas específicas de los usuarios en la búsqueda de compañeros de piso.
- En el Capítulo 3, se presenta de forma detallada la metodología empleada para la realización del proyecto. Se aborda no solo la organización de las tareas, sino también la distribución temporal de las distintas fases de trabajo. Este capítulo proporciona una guía estructurada sobre cómo se llevó a cabo el proceso de desarrollo, permitiendo entender cómo se gestionaron los recursos y se planificaron las actividades para alcanzar los objetivos propuestos en el proyecto.
- En el Capítulo 4 se abordan los elementos previos al diseño e implementación del sistema, como la identificación de actores, requisitos y especificaciones técnicas. Se proporciona una visión general y detallada de estos aspectos, sentando las bases para el desarrollo del proyecto.

- En el Capítulo 5 se presenta el diseño previsto para la implementación de la aplicación móvil. Se explica el uso del patrón MVVM y se justifica su elección como arquitectura software, además de detallar el modelo de capas a seguir en el desarrollo. También se explica el diseño de la base de datos, con su respectivo diagrama, los diagramas de clases y diagramas de secuencia. También se añade el diseño de la interfaz de la aplicación.
- En el Capitulo 6, se proporciona una descripción detallada de los aspectos técnicos necesarios para llevar a cabo el proyecto descrito en los capítulos anteriores. Abarca desde los requerimientos de hardware y software hasta las herramientas, tecnologías, servicios en la nube y organización del código fuente utilizados en el desarrollo. Esta información es fundamental para comprender la arquitectura del proyecto y cómo se materializó en una solución funcional.
- En el Capítulo 7 se presentan las pruebas realizadas durante la etapa de implementación con el fin de asegurar el funcionamiento adecuado del sistema.
- En el Capítulo 9.1, se muestran algunas posibles mejoras que se podrían realizar para el trabajo futuro.
- En última instancia, en el Capítulo 9.2 se presentan las conclusiones del trabajo y la evaluación personal del avance.

Capítulo 2

Estado del arte

Para comenzar con el desarrollo del proyecto propuesto, se realizará, previamente, un análisis detallado y fundamental de las diversas aplicaciones, de características similares, en el área de búsqueda de pisos/compañeros, disponibles en el mercado actual. Son numerosas las aplicaciones móviles existentes en la actualidad que abarcan funciones idénticas a las que se pretende implantar y desarrollar con “RoomieFinder”. Para ello, se completará una presentación y estudio de las más destacadas.

En primer lugar, se estudiarán aquellas aplicaciones móviles caracterizadas por sistemas de búsqueda de alojamiento, así como de compañeros de pisos con carácter genérico, como es el caso de la aplicación “Badi” y “Roomster”. Por otra parte, se abordarán aquellas aplicaciones caracterizadas por la implantación de sistemas de carácter especializado, tanto en la búsqueda de vivienda como en la búsqueda de convivientes, entre ellas se destaca “SpareRoom”. Todas ellas servirán de foco de atención en el proceso de creación de la aplicación del presente proyecto debido a lo representativo de las mismas.

A partir de la elaboración del análisis de estas, se pretende la identificación de las mejores prácticas, características más útiles e, incluso, detallar aquellos aspectos que se podrían calificar como mejorables y permitan, por tanto, desarrollar una metodología más novedosa y mejorada a través de una aplicación para iOS que sea acorde con el objetivo buscado en el presente Trabajo de Fin de Grado siendo este innovador, eficiente y focalizado en las necesidades específicas de los estudiantes universitarios y jóvenes profesionales en su proceso de búsqueda de compañeros de piso.

2.1 Badi

Badi se presenta como una aplicación móvil líder en el mercado de búsqueda de alojamiento compartido, diseñada para conectar a personas que buscan habitaciones o alojamientos completos con aquellos que ofrecen estos espacios disponibles. La plataforma se fundamenta en la oferta de funcionalidades claves como la creación de perfiles detallados, un sistema de coincidencias inteligente

basado en el uso de algoritmos avanzados, un sistema de mensajería integrado para facilitar la comunicación entre los distintos usuarios, y, un proceso de verificación de identidad para garantizar tanto la seguridad como la confianza en las conexiones establecidas. Es por ello por lo que utilizado un modelo denominado freemium, con servicios básicos gratuitos y opciones "premium" con funcionalidades adicionales. Su popularidad se ha consolidado especialmente entre estudiantes universitarios debido a su enfoque centrado en la experiencia del usuario y si eficaz sistema de coincidencias.

2.2 Roomster

Por su parte, Roomster se destaca en el área de búsqueda de alojamiento compartido por su enfoque en la personalización y adaptabilidad de la experiencia del usuario. Ofrece herramientas talese como la creación de perfiles individualizados y personalizados, un sistema de coincidencias que se basa tanto en algoritmos como en preferencias inalienables, y una plataforma de mensajería intuitiva para facilitar la interacción entre los usuarios. A diferencia de la anterior aplicación, Badi, tiene una fuerte presencia global y una amplia variedad de opciones de alojamiento, lo que lo hace atractivo para una amplia gama de usuarios, incluyendo el foco de estudiantes internacionales. Por otra parte, Badi y Roomster utiliza un modelo freemium.

2.3 SpareRoom

Finalmente, esta es una plataforma consolidada en el sector de búsqueda de alojamiento compartido, reconocida por su enfoque en la comunidad y la colaboración entre los usuarios. Da la oportunidad a los distintos usuarios a crear perfiles minuciosos, ofreciendo características únicas como la opción de buscar habitaciones disponibles en función de los intereses y hobbies compartidos. Además, su sistemas de mensajería facilita la comunicación entre las potenciales compañeros de piso. Esta aplicación se diferencia por su ambiente comunitario y su compromiso con la creación de conexiones genuinas entre los usuarios, más allá de la simple búsqueda de alojamiento.

2.4 Comparación de mercado

Tras analizar las opciones disponibles, se puede observar que existe una gran variedad de aplicaciones para encontrar alojamiento compartido. Sin embargo, la mayoría de estas aplicaciones se enfocan en un tipo de usuario diferente al que nuestro proyecto pretende alcanzar. Este enfoque distintivo nos ha llevado a considerar y adaptar diversas características que se han demostrado efectivas en otras aplicaciones.

A pesar de las diferencias en el enfoque, hemos tomado como ejemplo varias funcionalidades valiosas de estas aplicaciones existentes. Entre ellas se encuentran los filtros avanzados, que permiten a los usuarios refinar sus búsquedas de manera precisa, y el sistema de chat, que facilita la comunicación directa entre los usuarios, promoviendo una interacción más fluida.

Otra característica importante que hemos adoptado es la personalización del anuncio de búsqueda de compañero. Esta funcionalidad permite a los usuarios crear perfiles detallados y personalizados, aumentando la probabilidad de encontrar compañeros de piso compatibles.

Para ilustrar mejor estas comparaciones, hemos elaborado la tabla 2.1 que detalla las características principales de cada aplicación analizada. Esta tabla muestra claramente las similitudes y diferencias en relación con nuestro proyecto, destacando las áreas en las que podemos mejorar y aprender de las soluciones existentes.

Característica	**RoomieFinder**	**Badi**	**Roomster**	**SpareRoom**
Enfoque principal	Compatibilidad de estilos de vida entre estudiantes universitarios y recién graduados	Alquiler de habitaciones compartidas	Búsqueda de compañeros de piso	Alquiler de habitaciones compartidas
Filtros avanzados	Sí	Sí	Sí	Sí
Personalización	Alta	Media	Media	Media
Interfaz intuitiva	Sí	Sí	Sí	Sí
chat	Sí	Sí	Sí	Sí
Compatibilidad	iOS	iOS, Android	iOS, Android	iOS, Android

Tabla 2.1: Comparación de propuestas de mercado.

Capítulo 3

Planificación del proyecto

En el siguiente apartado se expondrá detalladamente la metodología adoptada para la ejecución del proyecto, abordando tanto la organización de las tareas como la distribución temporal de las diferentes fases de trabajo.

3.1 Metodología de trabajo

Para este proyecto se ha decidido utilizar un modelo de desarrollo incremental [1]. Este modelo descompone un proyecto en una sucesión de agregados denominados incrementos. Estos agregados conforman un fragmento de la funcionalidad total del proyecto. Este modelo descriptivo entrega un componente de trabajo con cada incremento.

Se ha decidido elegir este modelo de desarrollo por los siguientes beneficios:

- Entregar pronto el software que funciona: Con este enfoque, el software funcional se entrega en etapas, comenzando con un primer módulo. Esta entrega temprana puede proporcionar retroalimentación valiosa y ayudar a validar la dirección del proyecto desde las primeras etapas.

- Los módulos pueden completarse en diferentes momentos: Cada iteración produce un módulo funcional adicional del software. Esto significa que los diferentes componentes de la aplicación pueden completarse en momentos distintos, lo que permite una mayor flexibilidad en la planificación.

- Adaptable a los cambios de alcance: El desarrollo incremental permite que los módulos individuales sean añadidos, modificados o eliminados según sea necesario. Esto hace que el proceso de desarrollo sea altamente adaptable a los cambios en los requisitos o enfoque del proyecto, proporcionando una mayor capacidad de respuesta.

- Identificación y abordaje de riesgos por módulo: Al dividir el proyecto en módulos mas pequeños y manejables, se facilita la identificación y gestión de los riesgos. Los problemas que surgen en una iteración pueden ser abordados de una manera oportuna sin afectar al resto del proyecto. Además, la capacidad de entrega temprana permite identificar y corregir problemas potenciales a medida que surgen, reduciendo así el impacto de los riesgos en el proyecto en su conjunto.

Aun con todas estas ventajas, el desarrollo incremental puede enfrentar desafíos, ya que puede que la descomposición del producto en módulos requiere requisitos claros y completos, su adaptabilidad a cambios puede ser limitada sin trabajo iterativo, el producto no está completo hasta que todas las partes estén integradas, lo que puede complicar la evaluación, y la integración de las partes puede requerir esfuerzos adicionales y presentar desafíos técnicos.

3.1.1 Incrementos

Como hemos comentado antes, este proyecto se divide en incrementos, que son los siguientes:

- Análisis exhaustivo de requisitos.
- Diseño de la interfaz gráfica general de la aplicación.
- Desarrollo de la cabecera de la aplicación.
- Desarrollo de la vista de búsqueda y la vista de filtros.
- Desarrollo del componente de vista modal.
- Desarrollo de el registro del usuario y creación de anuncio.
- Desarrollo de la vista de perfil.
- Desarrollo de la vista de anuncios activos.
- Desarrollo de la vista de mensajes.
- Desarrollo de la vista de inicio de sesión.
- Conexión de la aplicación con la base de datos.
- Sistema de notificaciones en los mensajes del chat.
- Última iteración para revisión definitiva y documentación.

Cabe destacar que dentro de cada iteración se ha desarrollado tanto la parte gráfica, con sus pequeños componentes necesarios, como la lógica y su conexión con la base de datos y su recogida de datos.

3.2 Fases de trabajo

Las fases de trabajo son componentes clave de cualquier plan de trabajo, ya que proporcionan una estructura organizativa para la planificación y ejecución de un proyecto. Estas fases permiten dividir el proyecto en etapas manejables y proporcionan una guía para el seguimiento del progreso a lo largo del tiempo.

Sin embargo, antes de entrar en detalle sobre las fases de trabajo, es importante tener en cuenta el horario disponible para dedicar al proyecto. En este caso, se especifica que los días laborales son de lunes a viernes, con un horario de 4:00 p.m a 10:00 p.m, debido a compromisos laborales previos. Además, los sábados se dispone de un horario de 10:00 a.m a 1:00 p.m y de 5:00 p.m a 7:00 p.m para trabajar en el proyecto.

Esta limitación de tiempo puede influir en la planificación de las actividades y en la distribución de tareas a lo largo de las semanas. Es importante tener en cuenta estos horarios al elaborar el plan de trabajo, para asegurarse de asignar adecuadamente el tiempo disponible y maximizar la eficiencia en la realización de las actividades del proyecto.

La imagen 3.1 representa la planificación inicial de las tareas que se llevarán a cabo a lo largo del proyecto en cada una de sus iteraciones, junto con su duración estimada y todo esto representado en un diagrama de Gantt.

Descripción de la tarea	Fecha inicio	Fecha fin
Análisis de requisitos	1-2-24	17-2-24
Diseño de la interfaz gráfica	18-2-24	1-3-24
Desarrollo de la cabecera de la app	2-3-24	3-3-24
Desarrollo de la vista de búsqueda	4-3-24	7-3-24
Desarrollo de la vista de filtros	8-3-24	11-3-24
Desarrollo del componente de vista modal	12-3-24	13-3-24
Desarrollo de el registro del usuario	14-3-24	17-3-24
Desarrollo de la creación de anuncios	18-3-24	20-3-24
Desarrollo vista de perfil	21-3-24	26-3-24
Desarrollo de la vista de anuncios activos	27-3-24	31-3-24
Desarrollo de la vista de los mensajes	1-4-24	10-4-24
Desarrollo de la vista de inicio de sesión	11-4-24	17-4-24
Conexión de la aplicación con la BBDD	18-4-24	3-5-24
implementación de los chats	4-5-24	23-5-24
Población de la base de datos	24-5-24	25-5-24
Pruebas de funcionalidad	26-5-24	31-5-24
Última iteración para revisión definitiva y documentación	1-6-24	17-6-24

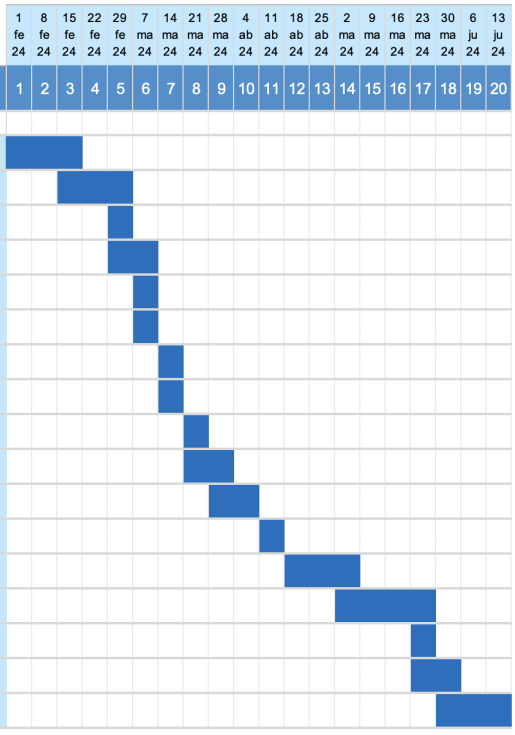


Figura 3.1: Fases del plan de trabajo representado en un diagrama de Gantt.

Estas estimaciones son flexibles y podrían ajustarse en función de la velocidad de desarrollo y cualquier contratiempo que pueda surgir durante el proceso de implementación.

3.3 Análisis de riesgos

El análisis de riesgos es un proceso fundamental en la gestión de proyectos, diseñado para identificar, analizar y abordar los posibles riesgos que podrían surgir durante su desarrollo. Este proceso se divide en varias fases, cada una de las cuales desempeña un papel crucial en la mitigación de los riesgos y la protección del éxito del proyecto:

1. Identificación: Esta fase implica localizar y reconocer los posibles riesgos antes de que se materialicen. Es fundamental realizar una evaluación exhaustiva de todos los aspectos del proyecto para identificar cualquier factor que pueda representar una amenaza en el futuro. Esto podría incluir riesgos técnicos, financieros, operativos, de recursos humanos, entre otros.
2. Análisis: Análisis: Una vez que se identifican los riesgos, es necesario evaluar su impacto potencial y su probabilidad de ocurrencia. Este análisis permite priorizar los riesgos según su gravedad y probabilidad, lo que ayuda a centrar los esfuerzos en abordar los riesgos más críticos para el éxito del proyecto.
3. Planificación: Con base en los resultados del análisis, se desarrolla un plan de acción detallado para mitigar los riesgos identificados. Este plan incluye medidas específicas para reducir la probabilidad de ocurrencia de los riesgos, así como para minimizar su impacto en caso de que se materialicen. Además, se establecen responsabilidades claras y se asignan recursos necesarios para implementar el plan de manera efectiva.
4. Monitorización: Una vez que el proyecto está en marcha, es crucial seguir de cerca los indicadores de riesgo para detectar cualquier signo de que los riesgos identificados podrían estar materializándose. Esto implica un monitoreo continuo del entorno del proyecto y una evaluación periódica del progreso realizado en la implementación del plan de mitigación de riesgos.

Los riesgos pueden clasificarse según su probabilidad de ocurrencia y las consecuencias que podrían tener en el desarrollo del proyecto, lo que permite valorar el impacto que podrían tener sobre el mismo, como se muestra en la imagen 3.2

Siguiendo esta clasificación, es factible examinar y filtrar los riesgos con un impacto significativo o superior, y para cada uno de estos, desarrollar un plan de contingencia.

Descripción	Demora en la programación inicial
Probabilidad	Casi seguro
Consecuencias	Moderadas
Impacto	Alto
Plan de contingencia	Ajustar el calendario en función de la nueva disponibilidad. Utilizar días no laborables para compensar cualquier posible retraso en el proyecto.

Tabla 3.1: Riesgo R1. Demora en la programación inicial.

		PROBABILIDAD				
		Raro	Poco probable	Posible	Muy probable	Casi seguro
CONSECUENCIAS	Despreciable	Bajo	Bajo	Bajo	Medio	Medio
	Menores	Bajo	Bajo	Medio	Medio	Medio
	Moderadas	Medio	Medio	Medio	Alto	Alto
	Mayores	Medio	Medio	Alto	Alto	Muy alto
	Catastróficas	Medio	Alto	Alto	Muy alto	Muy alto

Figura 3.2: Matriz de riesgos.

Descripción	Pérdida de información del proyecto
Probabilidad	Poco probable
Consecuencias	Catastróficas
Impacto	Alto
Plan de contingencia	Realizar copias de seguridad y utilizar herramientas como github.

Tabla 3.2: Riesgo R2. Pérdida de información del proyecto.

Descripción	Enfermedad del desarrollador
Probabilidad	Poco probable
Consecuencias	Moderadas
Impacto	Medio
Plan de contingencia	Se evaluará el lapso de tiempo perdido y se volverá a planificar con un aumento en la cantidad de horas disponibles.

Tabla 3.3: Riesgo R3. Enfermedad del desarrollador.

Descripción	Fallos en el hardware de un componente
Probabilidad	Raro
Consecuencias	Mayores
Impacto	Medio
Plan de contingencia	Sustituir o reparar dicho componente en el menor tiempo posible.

Tabla 3.4: Riesgo R4. Fallos en el hardware de un componente.

Descripción	Actualización o modificación de la API
Probabilidad	Poco probable
Consecuencias	Mayores
Impacto	Alto
Plan de contingencia	Adaptar el código a la nueva estructura que plantean

Tabla 3.5: Riesgo R5. Actualización o modificación de la API.

3.4 Costes del proyecto

En este proyecto, se han empleado tanto componentes de hardware como de software, además de tener en cuenta el coste asociado a los recursos humanos. Dado que el desarrollo ha sido realizado por un único desarrollador, se ha estimado el coste de los recursos humanos basándose en el salario bruto anual promedio de un desarrollador de software en España, que ronda los 18.000 € al año. Además, se ha presupuestado un trabajo a media jornada, incluyendo los fines de semana, debido a la limitación de disponibilidad del desarrollador durante las mañanas, tal como se menciona en la sección Fases de trabajo 3.2.

3.4.1 Hardware

Componente hardware	**Coste/Unidad (€)**	**Coste (€)**
MacBook Pro 14’ 2023 16GB	2.549,00	2.549,00
Ratón Kult-Radium	11,90	11,90
Teléfono móvil iPhone 13 128GB	729,00	729,00
Monitor Samsung Essential S24C330GAU 24”	110,00	110,00
Total		**3.399,90**

Tabla 3.6: Presupuesto para hardware.

3.4.2 Software

Componente software	**Coste/Unidad (€)**	**Coste (€)**
Paquete Office	149,00	149,00
Licencia Desarrollador Apple	92,15	92,15
Github	0,00	0,00
Firebase	0,00	0,00
SourceTree	0,00	0,00
Total		**241,15**

Tabla 3.7: Presupuesto para software.

3.4.3 Recursos humanos

Como se ha mencionado anteriormente, el cálculo del presupuesto para los recursos humanos se ha basado en un sueldo bruto anual de 18.000 €, equivalente a 1.500 € mensuales. Sin embargo, teniendo en cuenta que se ha planificado un trabajo de 35 horas semanales, el salario bruto real al mes sería de 1.260 €.

Empleados	**Coste/Unidad (€)**	**Coste (€)**
Guillermo Rodríguez Alonso	1.260,00	5.106,25
Total		**5.106,25**

Tabla 3.8: Presupuesto para recursos humanos.

3.4.4 Presupuesto total

Al sumar todos los presupuestos individuales, se ha calculado el presupuesto final necesario para el desarrollo del proyecto. El total resultante es el siguiente:

Componentes	**Coste (€)**
Hardware	3.399,90
Software	241,15
Recursos humanos	5.106,25
Total	**8.747,30**

Tabla 3.9: Presupuesto total del proyecto.

Capítulo 4

Análisis

En este capítulo se presentarán los diversos elementos que se han considerado antes de llevar a cabo el diseño e implementación del sistema. Esto incluye actores, requisitos y hasta los requisitos que el sistema debe cumplir, así como especificaciones y diagramas para su representación.

4.1 Actores del sistema

Los actores representan a cualquier agente interno o externo que actúa en el sistema. En este proyecto se han encontrado los siguientes:

- Usuario: Un usuario de una aplicación es un miembro del sistema que interactúa con la aplicación a través de una interfaz de usuario para llevar a cabo actividades o procesos específicos que satisfagan sus necesidades o logran sus objetivos.
- BD en la nube: Se trata de un servicio en la nube que almacena la información presentada al usuario en la aplicación. Se establece contacto con la aplicación mediante peticiones realizadas por el usuario. (Actor Externo)

4.2 Requisitos

Los requisitos son una declaración clara y específica que describe una funcionalidad, característica, o restricción que la aplicación debe incluir o cumplir. Los requisitos definen qué debe hacer la aplicación, cómo debe comportarse, y qué restricciones deben ser consideradas durante el desarrollo de la misma.

4.2.1 Requisitos funcionales

Los requisitos funcionales son especificaciones detalladas que detallan los objetivos y expectativas de una aplicación para satisfacer las necesidades y expectativas de sus usuarios y otras partes interesadas. Estos requisitos se enfocan en las funciones y servicios particulares que el sistema debe ofrecer.

El actor que va a realizar estos requisitos casi en su totalidad es el usuario, ya que lo demás son actores externos al sistema. Los requisitos funcionales identificados son los siguientes:

ID	Descripción
RF-01	El sistema permitirá el inicio de sesión para acceder a la pantalla principal.
RF-02	El sistema permitirá el registro.
RF-03	El sistema permitirá acceder a la aplicación.
RF-04	El sistema mostrará la lista de anuncios registrados en la BBDD.
RF-05	El sistema permitirá el filtrado de dichos anuncios.
RF-06	El sistema permitirá añadir como favorito a un anuncio.
RF-07	El sistema permitirá seleccionar un anuncio para ver la información del usuario de dicho anuncio.
RF-08	El sistema permitirá iniciar un chat con el usuario del anuncio desde la pantalla de información del anuncio.
RF-09	El sistema permitirá intercambiar mensajes entre dos usuarios.
RF-10	El sistema permitirá ver los chats existentes.
RF-11	El sistema permitirá modificar los datos de perfil.
RF-12	El sistema permitirá ver los anuncios activos del usuario.
RF-13	El sistema permitirá crear un nuevo anuncio.
RF-14	El sistema permitirá modificar un anuncio ya creado.
RF-15	El sistema permitirá eliminar un anuncio ya creado.
RF-16	El sistema permitirá cambiar la foto de perfil del usuario.
RF-17	El sistema permitirá ver el perfil del usuario.
RF-18	El sistema permitirá la creación de perfil después del registro.

Tabla 4.1: Requisitos funcionales del usuario.

4.2.2 Requisitos no funcionales

Los requisitos no funcionales en una aplicación móvil son aquellos que determinan cómo debe comportarse el sistema más allá de las funciones específicas que realiza. Se enfocan en aspectos como el rendimiento, la seguridad, la usabilidad, la compatibilidad y otros atributos de calidad que influyen en la experiencia del usuario y la eficiencia del sistema. Para desarrollar nuestros requisitos no funcionales, hemos utilizado el sistema FURPS. Este es un sistema para clasificar y analizar los requisitos de software. FURPS es un acrónimo que representa cinco categorías principales de requisitos:

- F-Funcionales (*Functional*): Describen lo que el sistema debe hacer y ya están definidos en la sección 4.2.1

- U-Usabilidad (*Usability*): Se refiere a la facilidad de utilización del sistema. Engloba aspectos como la interfaz de usuario, la experiencia de usuario (UX), la consistencia, la documentación y la habilidad para aprender a utilizar el sistema de manera rápida.
- R-Fiabilidad (*Reliability*): Enfocados en la capacidad del sistema para funcionar adecuadamente bajo determinadas condiciones. Disponibilidad, tasa de fallos, capacidad de recuperación ante fallos y precisión de los resultados.
- P-Rendimiento (*Performance*): La eficiencia del sistema en términos de tiempo de respuesta, tiempo de procesamiento, uso de recursos (memoria, CPU, ancho de banda) y la escalabilidad (capacidad del sistema para gestionar un incremento en la carga).
- S-Soportabilidad (*Supportability*): Asimismo, conocida como mantenibilidad, se refiere a la habilidad del sistema para ser alterado y extendido. Se trata de aspectos como la modularidad, la facilidad de prueba, la configurabilidad, la adaptabilidad y la capacidad de ser actualizado.

ID	**Descripción**
RNF-01	El sistema será ejecutado en cualquier dispositivo con sistema operativo iOS con versión 16 o superior.
RNF-02	El sistema funcionará en formato vertical.
RNF-03	La aplicación se adaptará a los diferentes tamaños de dispositivos disponibles en el mercado para su correcta visualización.
RNF-04	El sistema mostrará al usuario a través de una alerta si no carga alguna información o si necesita alguna información.
RNF-05	El sistema deberá estar disponible las 24 horas del día durante todos los dias del año.
RNF-06	El sistema deberá soportar una gran cantidad de tráfico de usuarios.
RNF-07	Los datos de los usuarios introducidos en el sistema se almacenarán de forma segura.
RNF-08	El sistema establecerá una comunicación con el servicio “Cloud Firestore” (de la compañía Firebase) en cada momento en que el usuario solicite información, con el fin de recuperarla y posteriormente mostrarla al usuario.
RNF-09	El sistema debe contar con un lapso de respuesta inferior a cinco segundos.
RNF-10	El sistema siempre estará en aspecto claro.

Tabla 4.2: Requisitos no funcionales.

4.3 Casos de uso

En esta sección se presentará el modelo de casos de uso elaborado en función de los actores y los requisitos funcionales mencionados en los apartados previos. El propósito consiste en proporcionar una comprensión detallada de las interacciones entre los usuarios y el sistema, así como detallar la eficacia del sistema. Los casos de uso son una herramienta esencial para el análisis y diseño de sistemas, ya que permiten identificar y organizar los requerimientos funcionales de manera estructurada. Cada caso de uso describe una secuencia de acciones que proporcionan un resultado tangible de valor para un individuo en particular.

En la imagen 4.1 se muestra un diagrama de casos de uso, el cual ilustra las diferentes funcionalidades del sistema y cómo interactúan los usuarios con él. Este diagrama sirve como una representación visual de los casos de uso identificados y cómo estos se relacionan entre sí.

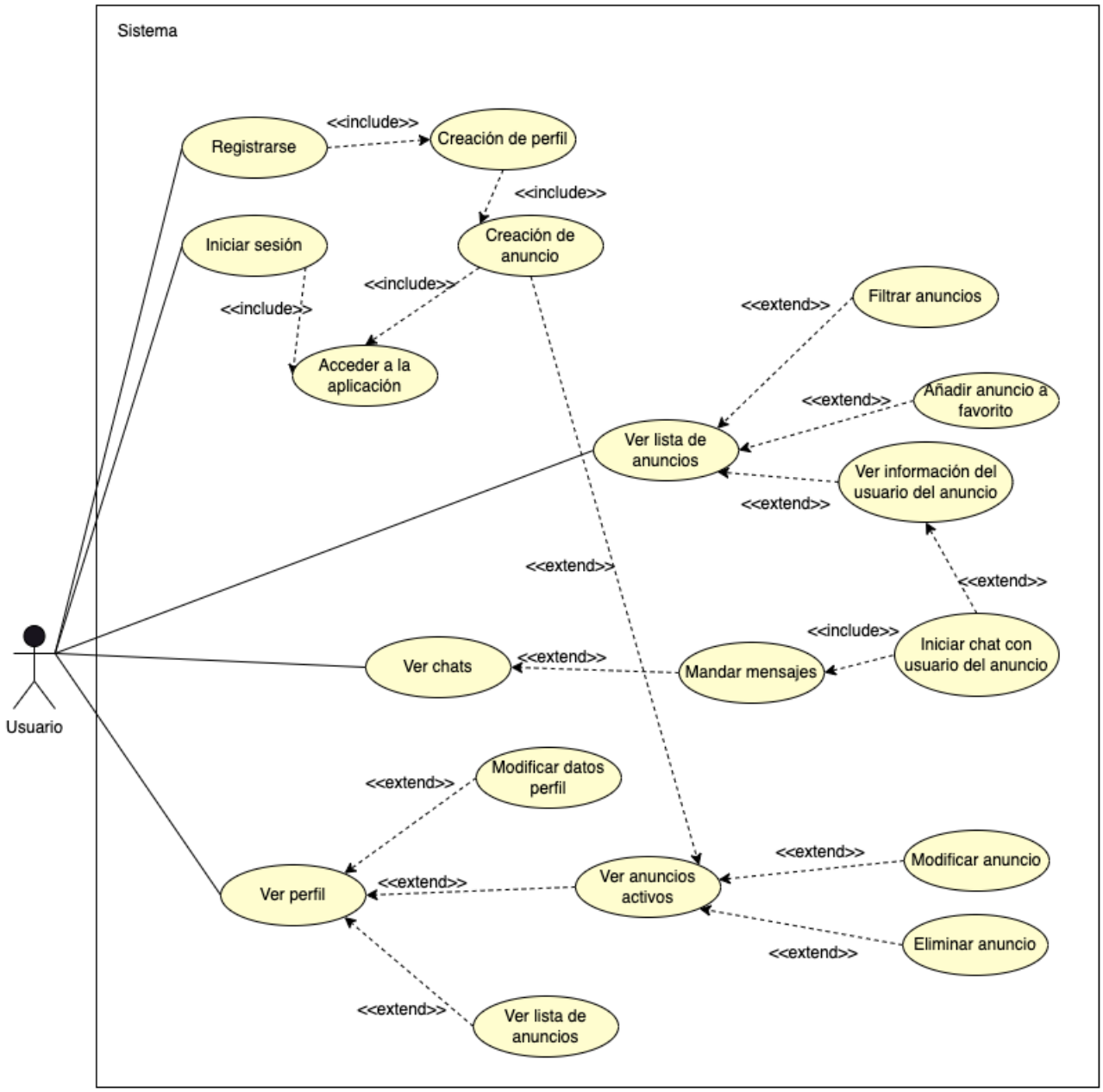


Figura 4.1: Diagrama de casos de uso.

Con motivo de mejorar la visibilidad del diagrama, se han suprimido las lineas desde los caso *Ver lista de anuncios*, *Ver chats* y *Ver perfil*, que extenderían desde el caso de uso *Acceder a la aplicación*. Esto significa que necesitas ser usuario registrado para acceder a estos casos de uso.

4.3.1 Especificaciones de los casos de uso

A continuación, se procederá a explicar y describir el flujo de cada caso de uso. En cada caso de uso identificado en el diagrama, se detallarán los pasos específicos que los usuarios deben seguir para interactuar con el sistema y alcanzar el objetivo deseado. Esta descripción constará con el flujo principal de eventos, así como cualquier flujo alternativo que pueda surgir debido a condiciones excepcionales o decisiones del usuario.

Cada caso de uso será presentado con la siguiente estructura:

- **Identificador**: Se indicará el título del caso de uso para identificarlo claramente.
- **Descripción**: Se proporcionará una descripción general del caso de uso, explicando su propósito y valor para los usuarios.
- **Actor**: Se identificará al actor que interactúe con el caso de uso.
- **Precondiciones**: Se listarán las condiciones que deben cumplirse antes de que el caso de uso pueda comenzar.
- **Post condiciones**: Se describirán los resultados esperados después de la ejecución exitosa del caso de uso.
- **Flujo principal**: Se detallará la secuencia de pasos que constituyen el flujo principal del caso de uso.
- **Flujos alternativos**: Se identificarán y describirán las secuencias de pasos alternativas que pueden ocurrir debido a diferentes condiciones o decisiones del usuario.

Identificador	CU-01 Registro
Descripción	Caso de uso que corresponde al RF-02
Actor	Usuario
Precondiciones	
Flujo principal	1. El usuario navega a la pagina de registro. 2. El sistema muestra el formulario de registro. 3. El usuario introduce los datos necesarios (nombre, apellido, fecha de nacimiento, correo y contraseña). 4. El usuario pulsa el botón de registrarse. 5. El sistema comprueba que los datos son correctos. 6. El sistema crea una nueva cuenta para el usuario. CU-02 Creación de perfil.
Flujos alternativos	5a. Si los datos no son los correctos, el sistema notifica al usuario. El caso de uso volvería al paso 3.
Post condiciones	El usuario está registrado en el sistema y puede iniciar sesión.

Tabla 4.3: CU-01 Registro.

Identificador	CU-02 Creación de perfil
Descripción	Caso de uso que corresponde al RF-18
Actor	Usuario
Precondiciones	El usuario debe haber completado el registro.
Flujo principal	1. El usuario accede a la sección de creación de perfil. 2. El sistema muestra el formulario de perfil. 3. El usuario introduce la información del perfil (foto, descripción, estudios, etc.). 4. El sistema comprueba que todo esta relleno correctamente. 5. El sistema guarda la información del perfil. CU-16 Crear un nuevo anuncio.
Flujos alternativos	4a. Si no esta bien rellenado, el sistema notifica al usuario. El caso de uso volvería al paso 2.
Post condiciones	El perfil del usuario está completo y guardado en el sistema.

Tabla 4.4: CU-02 Creación de perfil.

Identificador	CU-03 Iniciar sesión
Descripción	Caso de uso que corresponde al RF-01
Actor	Usuario
Precondiciones	El usuario esta registrado dentro de la aplicación.
Flujo principal	1. El usuario accede a la página de inicio de sesión. 2. El sistema muestra el formulario de inicio de sesión. 3. El usuario introduce su email y contraseña y pulsa el botón de iniciar sesión. 4. El sistema verifica las credenciales. CU-04 Acceder a la aplicación.
Flujos alternativos	4a. Si los datos no son los correctos, el sistema notifica al usuario. El caso de uso volvería al paso 2.
Post condiciones	El usuario está autenticado y tiene acceso a la aplicación.

Tabla 4.5: CU-03 Iniciar sesión.

Identificador	CU-04 Acceder a la aplicación.
Descripción	Caso de uso que corresponde al RF-03
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión o haberse registrado exitosamente.
Flujo principal	1. El usuario accede a la aplicación. 2. El sistema muestra la pantalla principal con las diferentes ventanas.
Flujos alternativos	
Post condiciones	El usuario tiene acceso a todas las funcionalidades principales de la aplicación.

Tabla 4.6: CU-04 Acceder a la aplicación.

Identificador	CU-05 Ver lista de anuncios.
Descripción	Caso de uso que corresponde al RF-04
Actor	Usuario
Precondiciones	El usuario debe haber accedido a la aplicación.
Flujo principal	1. El usuario accede a la vista de búsqueda. 2. El sistema recupera la información de la base de datos y muestra la lista de anuncios disponibles.
Flujos alternativos	
Post condiciones	El usuario ve la lista de anuncios disponibles en la aplicación.

Tabla 4.7: CU-05 Ver lista de anuncios.

Identificador	CU-06 Filtrar anuncios.
Descripción	Caso de uso que corresponde al RF-05
Actor	Usuario
Precondiciones	El usuario debe estar en la vista de búsqueda.
Flujo principal	1. El usuario selecciona la opción de filtrado. 2. El sistema muestra las opciones de filtrado disponibles. 3. El usuario selecciona los criterios de filtrado. 4. El sistema comprueba si los filtros están correctos. 5. El sistema actualiza la lista de anuncios según los criterios seleccionados
Flujos alternativos	4b. Los filtros están mal introducidos, el sistema notifica al usuario. El caso de uso volvería al paso 2.
Post condiciones	La lista de anuncios se muestra filtrada según los criterios seleccionados por el usuario.

Tabla 4.8: CU-06 Filtrar anuncios.

Identificador	CU-07 Añadir anuncio a favoritos.
Descripción	Caso de uso que corresponde al RF-06
Actor	Usuario
Precondiciones	El usuario debe estar viendo la lista de anuncios.
Flujo principal	1. El usuario selecciona un anuncio. 2. El sistema muestra la opción de añadir a favoritos. 3. El usuario selecciona la opción de añadir a favoritos. 4. El sistema añade el anuncio a la lista de favoritos del usuario.
Flujos alternativos	
Post condiciones	El anuncio se añade a la lista de favoritos del usuario.

Tabla 4.9: CU-07 Añadir anuncio a favoritos.

Identificador	CU-08 Ver información del usuario del anuncio.
Descripción	Caso de uso que corresponde al RF-07
Actor	Usuario
Precondiciones	El usuario debe estar viendo la lista de anuncios.
Flujo principal	1. El usuario selecciona un anuncio. 2. El sistema muestra la información del usuario que publicó el anuncio.
Flujos alternativos	
Post condiciones	El usuario ve la información del publicador del anuncio.

Tabla 4.10: CU-08 Ver información del usuario del anuncio.

Identificador	CU-09 Iniciar chat con el usuario del anuncio.
Descripción	Caso de uso que corresponde al RF-08
Actor	Usuario
Precondiciones	El usuario debe estar viendo la información del usuario del anuncio.
Flujo principal	1. El usuario selecciona la opción de iniciar chat. 2. El sistema crea un nuevo chat con el usuario del anuncio. 3. El usuario selecciona la opción de añadir a favoritos. 4. El sistema abre la ventana de chat para enviar mensajes. CU-10 Mandar mensajes.
Flujos alternativos	
Post condiciones	El usuario puede comenzar a enviar mensajes. Cuando se manda el primer mensaje, se crea y se guarda en la base de datos el chat. Hasta entonces no se habría creado.

Tabla 4.11: CU-09 Iniciar chat con el usuario del anuncio.

Identificador	CU-10 Mandar mensajes.
Descripción	Caso de uso que corresponde al RF-09
Actor	Usuario
Precondiciones	El usuario debe haber iniciado un chat o haber escrito anteriormente un mensaje.
Flujo principal	1. El usuario abre un chat existente o inicia un nuevo chat. 2. El usuario escribe y envía un mensaje. 3. El sistema entrega el mensaje al destinatario. 4. El sistema muestra los mensajes recibidos en el chat.
Flujos alternativos	
Post condiciones	Los mensajes se envían y reciben correctamente entre los usuarios.

Tabla 4.12: CU-10 Mandar mensajes.

Identificador	CU-11 Ver chats.
Descripción	Caso de uso que corresponde al RF-10
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión.
Flujo principal	1. El usuario selecciona la opción de ver chats. 2. El sistema muestra una lista con todos los chats existentes del usuario.
Flujos alternativos	
Post condiciones	El usuario ve la lista de todos los chats en los que está participando.

Tabla 4.13: CU-11 Ver chats.

Identificador	CU-12 Ver perfil.
Descripción	Caso de uso que corresponde al RF-17
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión.
Flujo principal	1. El usuario selecciona la opción de ver perfil. 2. El sistema muestra la foto de perfil y dos botones, uno para anuncios activos y otro para modificar la información del perfil.
Flujos alternativos	
Post condiciones	El usuario ve la vista de perfil.

Tabla 4.14: CU-12 Ver perfil.

Identificador	CU-13 Cambiar foto de perfil.
Descripción	Caso de uso que corresponde al RF-16
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión.
Flujo principal	1. El usuario selecciona la opción de ver perfil. 2. El sistema muestra la foto de perfil y dos botones, uno para anuncios activos y otro para modificar la información del perfil. 3. El usuario pincha en la foto de perfil para acceder a la galería. 4. El sistema muestra la galería para seleccionar una nueva foto de perfil. 5. El usuario selecciona una nueva foto. 6. el sistema guarda y muestra la nueva foto de perfil.
Flujos alternativos	
Post condiciones	La foto de perfil del usuario se actualiza en el sistema.

Tabla 4.15: CU-13 Cambiar foto de perfil.

Identificador	CU-14 Modificar datos de perfil.
Descripción	Caso de uso que corresponde al RF-11
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión.
Flujo principal	1. El usuario selecciona la opción de ver perfil. 2. El sistema muestra la foto de perfil y dos botones, uno para anuncios activos y otro para modificar la información del perfil. 3. El usuario pincha en el botón de modificar información del perfil. 4. El sistema muestra el formulario de perfil con los datos actuales. 5. El usuario modifica los datos deseados. 6. el sistema guarda los cambios. CU-12 Ver perfil
Flujos alternativos	
Post condiciones	Los datos del perfil del usuario se actualizan en el sistema.

Tabla 4.16: CU-14 Modificar datos de perfil.

Identificador	CU-15 Ver anuncios activos.
Descripción	Caso de uso que corresponde al RF-12
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión y estar en la vista de ver perfil.
Flujo principal	1. El usuario selecciona la opción de ver anuncios activos. 2. El sistema muestra una lista de todos los anuncios activos del usuario.
Flujos alternativos	
Post condiciones	El usuario ve todos los anuncios que ha publicado.

Tabla 4.17: CU-15 Ver anuncios activos.

Identificador	CU-16 Crear un nuevo anuncio.
Descripción	Caso de uso que corresponde al RF-13
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión y estar en la vista de ver anuncios activos.
Flujo principal	1. El usuario selecciona la opción de crear un nuevo anuncio. 2. El sistema muestra el formulario de creación de anuncio. 3. El usuario introduce los detalles del anuncio. 4. El sistema comprueba que los datos estén bien introducidos. 5. El sistema guarda el anuncio y lo publica
Flujos alternativos	4b. Los datos están mal introducidos, el sistema notifica al usuario. El caso de uso se reanuda en el punto 2.
Post condiciones	El nuevo anuncio se guarda y se publica en el sistema.

Tabla 4.18: CU-16 Crear un nuevo anuncio.

Identificador	CU-17 Modificar anuncio.
Descripción	Caso de uso que corresponde al RF-14
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión y tener anuncios creados.
Flujo principal	1. El usuario selecciona el anuncio que desea modificar. 2. El sistema muestra el formulario de edición con los detalles actuales del anuncio. 3. El usuario modifica los detalles del anuncio. 4. El sistema comprueba los datos introducidos. 5. El sistema guarda los cambios.
Flujos alternativos	4b. Los datos introducidos son incorrectos, el sistema notifica al usuario y el caso de uso se reanuda en el paso 2.
Post condiciones	Los cambios en el anuncio se guardan en el sistema.

Tabla 4.19: CU-17 Modificar anuncio.

Identificador	CU-18 Eliminar anuncio.
Descripción	Caso de uso que corresponde al RF-15
Actor	Usuario
Precondiciones	El usuario debe haber iniciado sesión y tener anuncios creados.
Flujo principal	1. El usuario selecciona el anuncio que desea eliminar. 2. El sistema solicita confirmación para eliminar el anuncio. 3. El usuario confirma la eliminación. 4. El sistema elimina el anuncio.
Flujos alternativos	3b. El usuario cancela la eliminación. Se acaba el caso de uso.
Post condiciones	El anuncio se elimina del sistema.

Tabla 4.20: CU-18 Eliminar anuncio.

Capítulo 5

Diseño

5.1 Arquitectura

La estructura general de una aplicación móvil establece la organización y la interacción entre los componentes de software. Este diseño es fundamental para asegurar que la aplicación sea eficiente, mantenible, escalable y fácil de implementar. En general, una buena arquitectura debe brindar:

- Separación de responsabilidades: Cada elemento de la aplicación debe poseer una función específica y detallada.
- Escalabilidad: La arquitectura debe permitir la incorporación de nuevas funcionalidades como el mínimo impacto en el sistema actual.
- Mantenibilidad: Los cambios y la corrección de errores deben ser sencillos de llevar a cabo.
- Reutilización: Los componentes deben ser reutilizados en distintas áreas de la aplicación

Con esto, vamos a pasar a explicar los dos tipos de arquitecturas, la lógica y la física.

5.1.1 Arquitectura lógica

La arquitectura lógica se enfoca en la estructura conceptual del sistema, describiendo cómo se organizan e interrelacionan los componentes software desde una perspectiva de diseño. Este tipo de arquitectura no se preocupa por la implementación física del sistema, solo por cómo se dividen las responsabilidades y cómo se comunican las diferentes capas. Esta arquitectura se selecciona y se diseña en función de los requisitos y las restricciones.

Para el diseño de la arquitectura lógica, se definirá el número de capas que compondrán la aplicación. En este caso, se ha optado por una arquitectura de 3 capas (presentación, lógica y persistencia):

- Presentación: La capa de presentación es la capa más alta del software. Se ubica la interfaz de usuario del programa. Su lógica se limita exclusivamente a la representación de los datos y la ejecución de cada operación CRUD (Crear, leer, actualizar o eliminar). En esta capa, la lógica debe ser la más simplificada.

- Lógica: Como su nombre lo indica, la mayoría de la lógica de la aplicación se ubicará en esta capa. La capa de lógica carece de un profundo conocimiento de ningún modelo de datos, por lo que no se puede llevar a cabo una consulta directa a la base de datos. La entrada de la capa es una estructura de información recibida de la capa de presentación, mientras que la salida de la capa es una estructura de datos recibida por la capa de persistencia. La capa de lógica posibilita a los desarrolladores diseñar aplicaciones que incluyen múltiples interfaces de usuario, lo cual reduce las posibilidades de duplicación innecesaria del código.

- La capa de persistencia es la capa más baja de la arquitectura de software, y se ocupa del almacenamiento y recuperación de datos. Su función principal es interactuar con las bases de datos u otros sistemas de almacenamiento, gestionando la forma en que los datos se guardan, actualizan y recuperan de manera eficiente y segura. En nuestro va a interactuar con nuestra base de datos en la nube (Firebase).

A mayores de esta división por capas, vamos a aplicar un patrón de software. Los patrones de software son soluciones verificadas y reutilizadas par problema habituales que surgen durante el desarrollo de software. La representación de las mejores prácticas adquiridas a lo largo del tiempo por expertos en el campo proporciona una herramienta idónea para abordar problemas de diseño, estructura y comportamiento.

En este caso, se va a utilizar el patrón de diseño MVVM [2]-*Model View ViewModel*. Este es un patrón de diseño arquitectónico que separa la lógica de negocio y la interfaz de usuario en aplicaciones de manera que cada componente tenga responsabilidades claras y se pueda mantener y probar de forma independiente. Este procedimiento es particularmente popular en el desarrollo de aplicaciones de escritorio y móviles.

- *Model*: Representa los datos y la lógica de negocio de la aplicación. Contiene las clases que representan los objetos de negocio y maneja la persistencia de datos.

- *View*: La vista es la representación visual de la información proporcionada por el ViewModel. Cumple una doble función: por un lado, envía acciones al ViewModel y, por otro, escucha los cambios en el modelo que recibe del ViewModel. Si alguna propiedad del ViewModel cambia y estamos escuchando desde la vista, detectaremos ese cambio y podremos actualizar la vista en consecuencia.

- *ViewModel*: El ViewModel es una clase que sirve como intermediario entre la interfaz de usuario (View) y el modelo de datos (Model). Su función principal es preparar y presentar los datos del modelo de una manera adecuada para la vista, y también manejar las interacciones del usuario con la vista y actualizar el modelo en consecuencia.

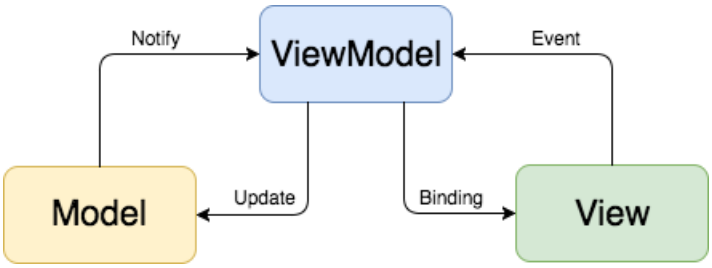


Figura 5.1: Patrón MVVM.

5.1.2 Arquitectura física

La arquitectura física se refiere al diseño y disposición física de los componentes hardware, tales como servidores, dispositivos de almacenamiento y redes, necesarios para respaldar y ejecutar una aplicación o sistema de software. Esta arquitectura determina la distribución y conexión física de los recursos informáticos, con el fin de asegurar el rendimiento, la disponibilidad y la escalabilidad del sistema.

En este proyecto vamos a utilizar el modelo cliente-servidor. Este modelo es un paradigma de arquitectura de software en el que los clientes solicitan servicios a los servidores mediante una red. Los servidores procesan las solicitudes y remiten las respuestas pertinentes. Se trata de una estructura que simplifica la distribución de tareas, la centralización de recursos y es ampliamente utilizada en sistemas distribuidos y de red. Llevándolo a nuestro proyecto, como parte de cliente estará el dispositivo usado por el usuario para acceder a la aplicación, mientras que en el servidor se encontrará el componente de Firebase, que hablaremos en el siguiente capitulo sobre él, para acceder a la base de datos.

Este tipo de modelo tiene las siguientes ventajas:

- Escalabilidad.
- Centralización de recursos.
- Seguridad.

5.2 Modelos de diseño

En esta sección se adjuntan diferente diagramas y representaciones que ayudarán a comprender mejor el diseño utilizado en nuestra herramienta.

5.2.1 Diagrama de Clases

Un diagrama de clases es una representación gráfica que presenta la estructura de un sistema, modelando sus clases, atributos, operaciones y las relaciones entre los objetos. Con el fin de mejorar la visualización y comprensión del diagrama, se ha decidido dividirlo en diversas categorías, según el patrón MVVM.

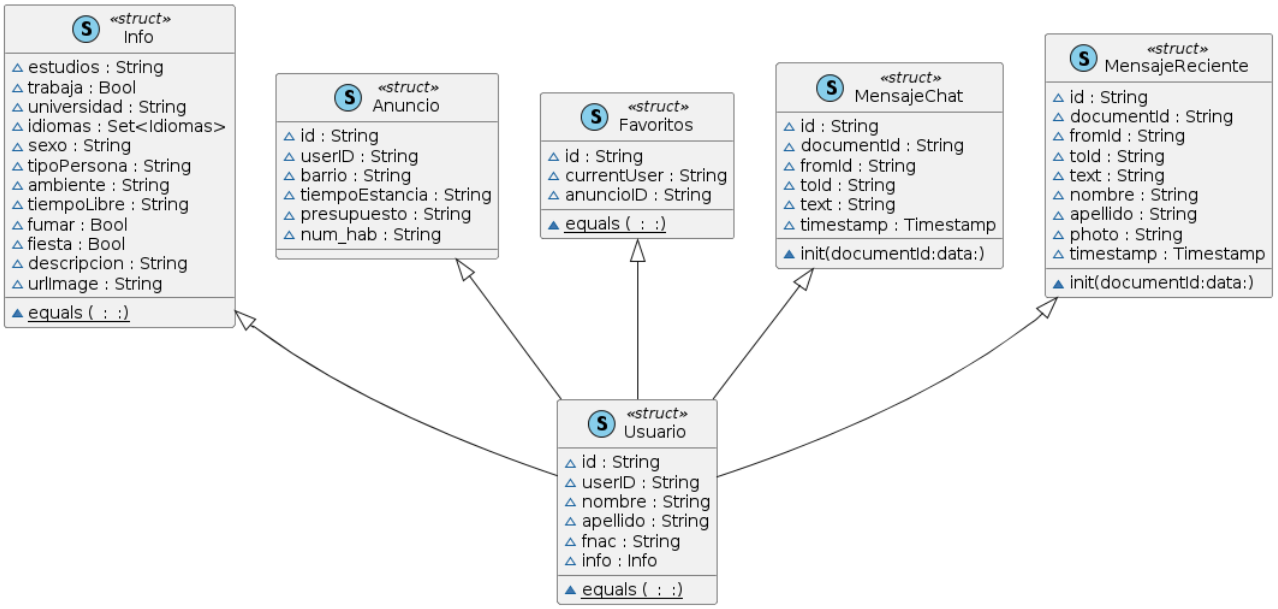


Figura 5.2: Diagrama de clases del modelo.

A continuación, se presentarán las distintas clases del diagrama:

- Usuario - Información de cada usuario registrado en el sistema.
- Info - Información adicional del usuario registrado en el sistema.
- Anuncio - Información de los anuncios registrados en el sistema.
- Favoritos - Representa los anuncios guardados como favoritos por los usuarios.
- MensajeChat - Representa cada mensaje del chat asociado a un usuario.
- MensajeReciente - Información de los ultimos mensajes registrados en el sistema.

El siguiente diagrama de clases se centra en las vistas (*View*) y los modelos de vista (*ViewModel*)

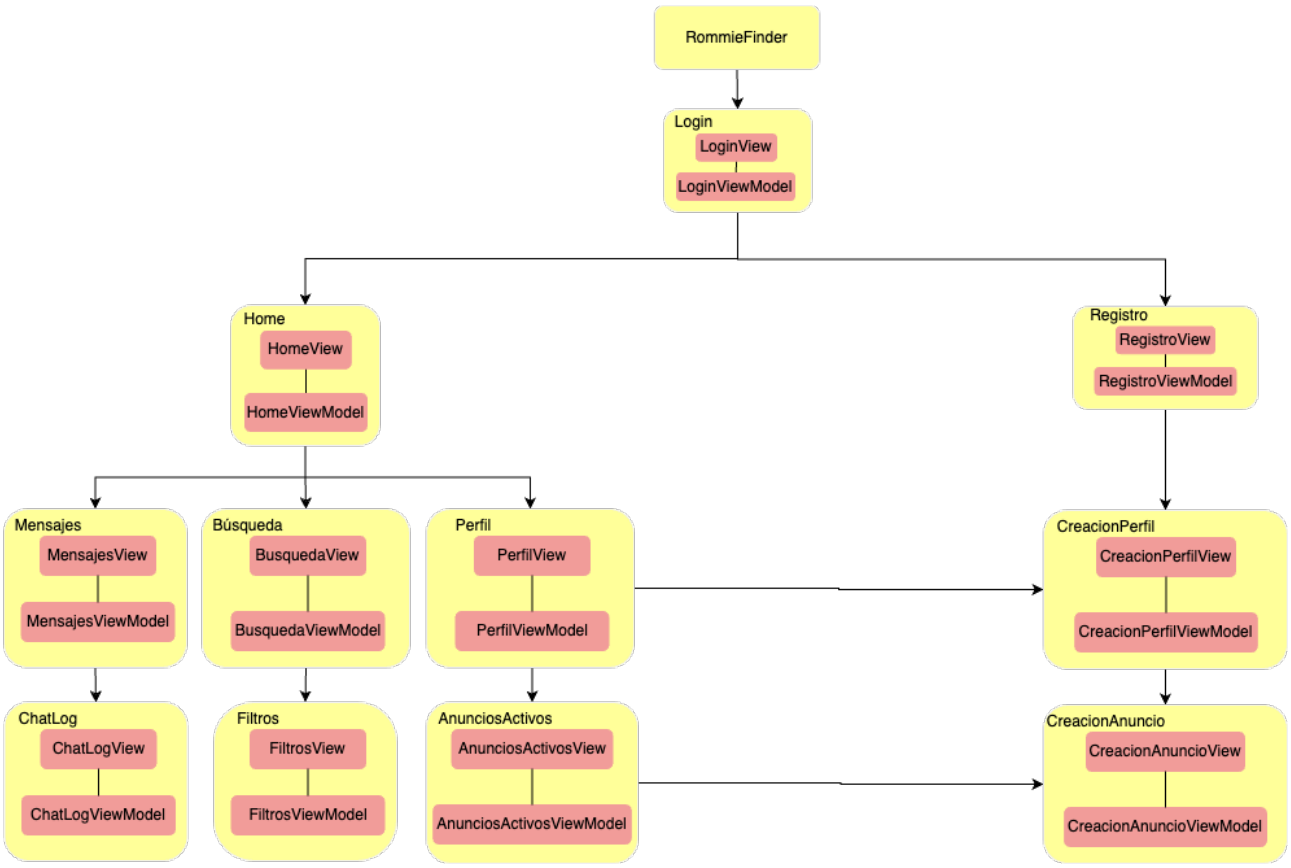


Figura 5.3: Diagrama de clases del View y del ViewModel.

Las clases de vista (*View*) representan las interfaces de usuario. Cada vista se ocupa de mostrar los datos y gestionar las interacciones del usuario. En el diagrama, cada vista está concebida para una función particular del sistema, tales como el inicio de sesión, la creación del perfil, la búsqueda de anuncios, entre otras. Como vemos el nombre se forma uniendo la palabra que define a la función, más la palabra *"View"*.

Por parte de las clases de modelo de vista (*ViewModel*), contienen la lógica y los datos necesarios para que las vistas funcionen correctamente. Actúan como intermediarios entre las vistas y los modelos de datos, gestionando la interacción del usuario y actualizando las vistas con datos desde el modelo. Cada vista tiene su *ViewModel* correspondiente que se encarga de proporcionar y procesar la información necesaria. Como hemos visto en los casos de las vistas, el nombre del *ViewModel* también se forma con la palabra que define la función, que siempre entre las vistas y los modelos de vista comparten está palabra, y luego se le añade la palabra *ViewModel*.

5.2.2 Diagramas de Secuencia

Un diagrama de secuencia es un método empleado en la ingeniería de software y en el diseño de sistemas para ilustrar la interacción entre los diferentes objetos en un período temporal específico. Se enfoca en el orden de las operaciones y en los mensajes que se envían entre los objetos para lograr una funcionalidad particular. En esencia, representa la secuencia de acontecimientos en un sistema, detallando la interacción dinámica entre los objetos para llevar a cabo un procedimiento específico.

En este documento no se presentarán los diagramas de secuencia de todos los casos de uso, sino que se seleccionarán los considerados más relevantes dentro del proyecto.

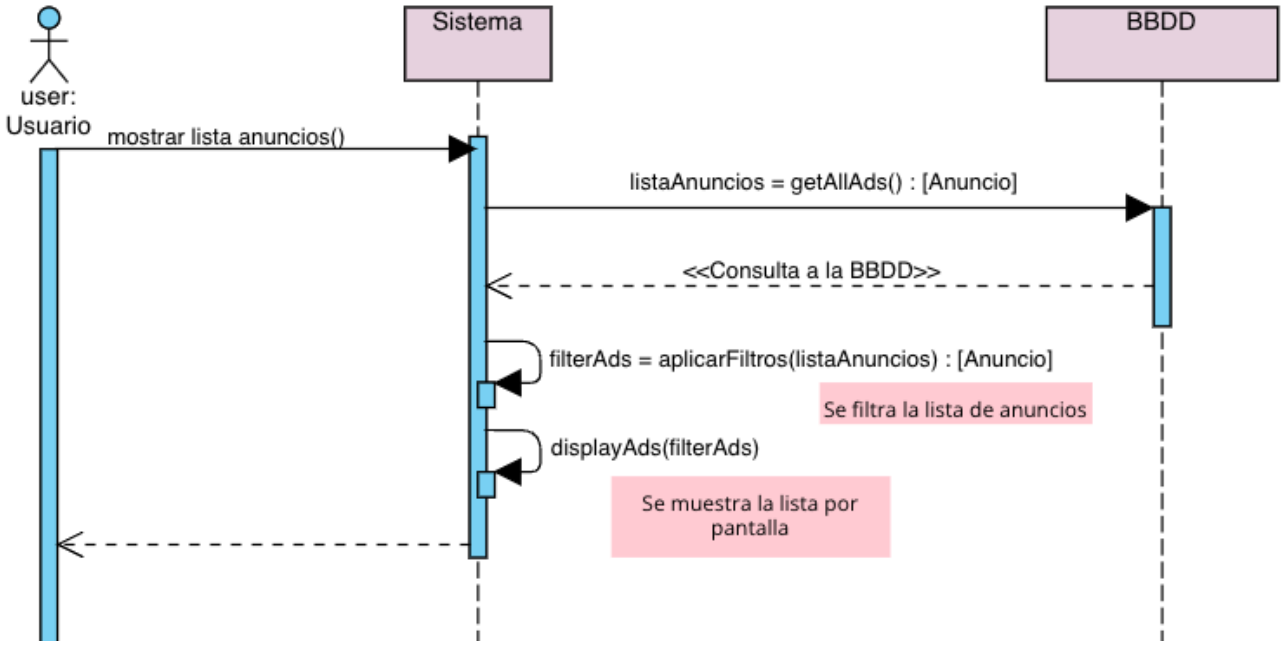


Figura 5.4: Diagrama de secuencia CU-05 Ver lista de anuncios.

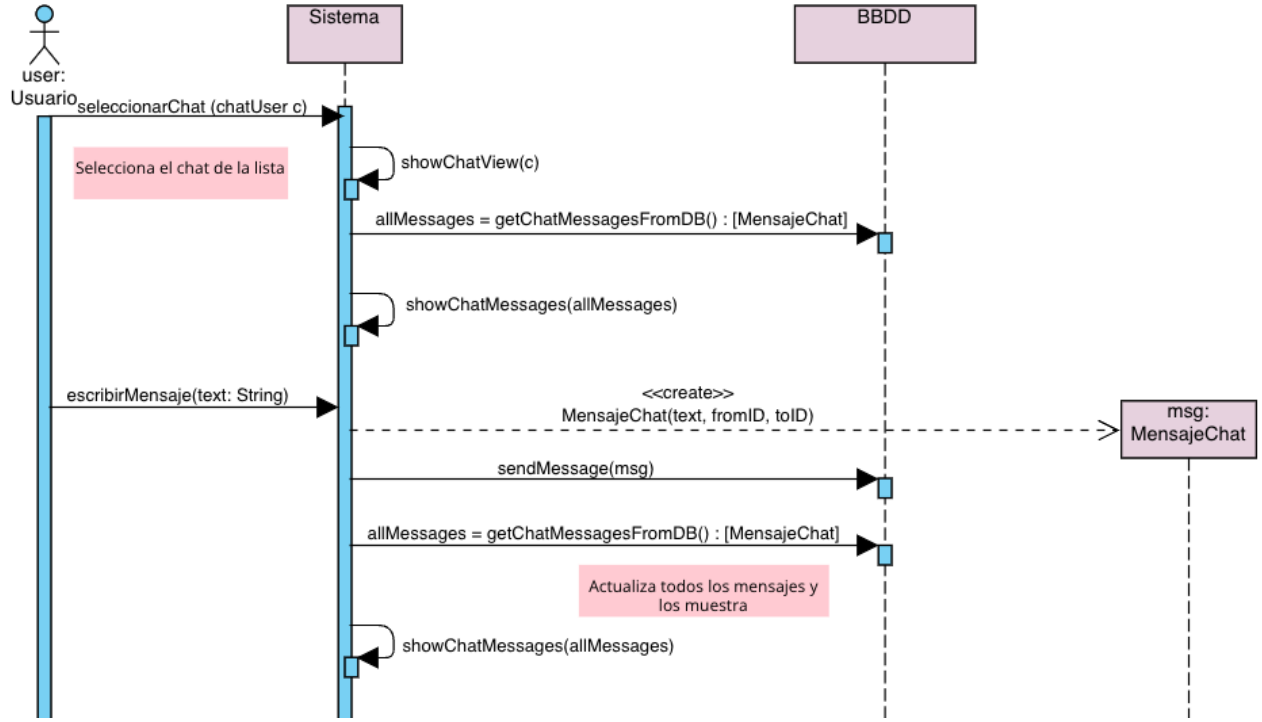


Figura 5.5: Diagrama de secuencia CU-10 Mandar mensajes.

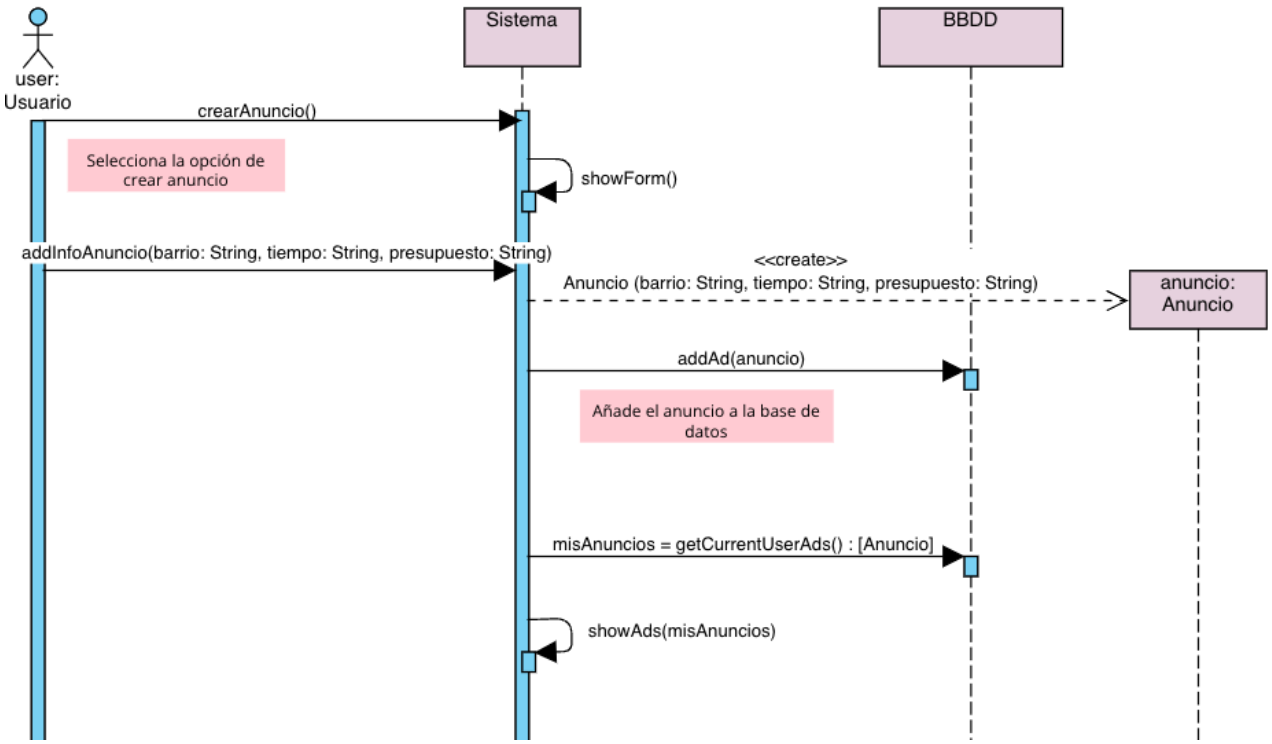


Figura 5.6: Diagrama de secuencia CU-16 Crear un nuevo anuncio.

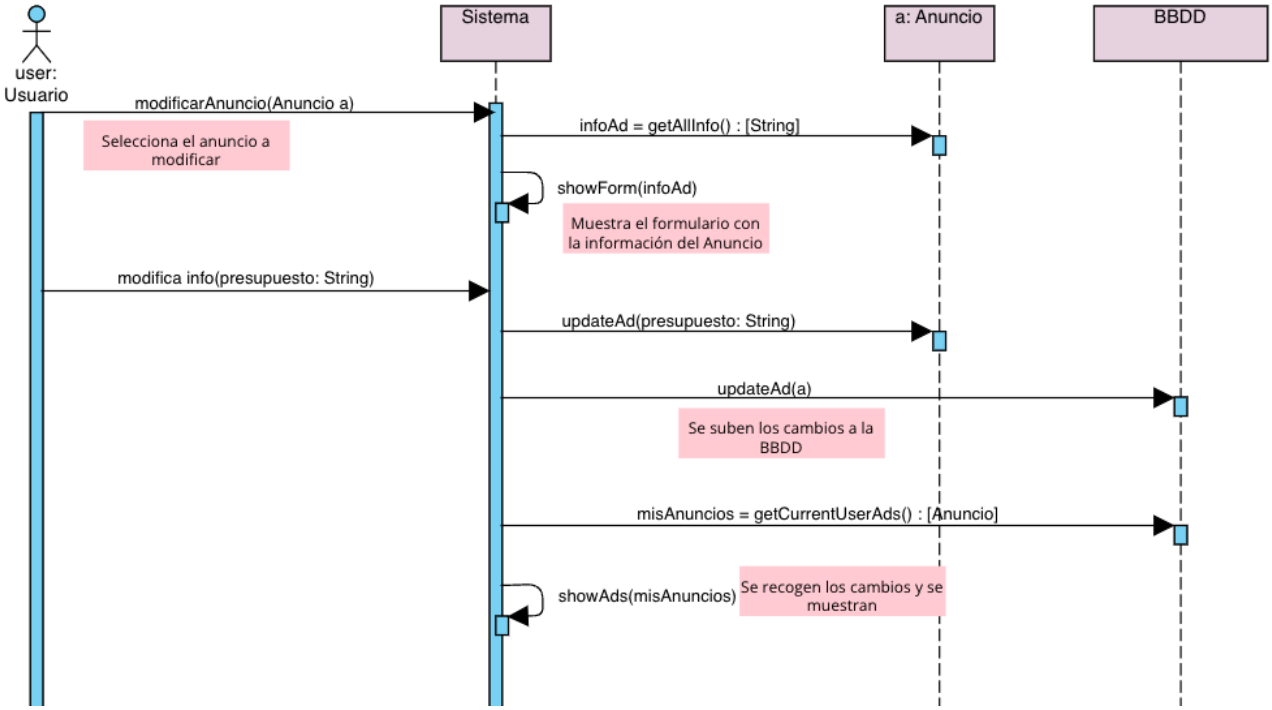


Figura 5.7: Diagrama de secuencia CU-17 Modificar anuncio.

5.3 Diseño de la base de datos

En este apartado nos enfocaremos en la información almacenada en la base de datos Firebase Cloud Firestore, que más adelante explicaremos en profundidad, dado que es la base de datos principal donde se guarda la mayor parte de los datos del programa.

En el presente servicio, almacenamos en colecciones toda la información referente a perfiles, anuncios, favoritos, chats y mensajes recientes, conforme a las tablas a continuación. Dentro de cada colección, se crean documentos, que estos a su vez tienen ya la información de cada instancia. Las claves primarias siempre serán los identificadores de los documentos, creados automáticamente. En algún caso especial y mas complejo, como lo son ambas colecciones relacionadas con los chats, se explicará como ha sido la organización antes de su respectiva tabla. Por lo tanto, en el caso de los Perfiles, la clave primaria sería el conjunto de el identificador del documento más el campo *"userID"*. Para cada una de las tablas, se proporcionará el nombre del campo, una descripción detallada, el tipo de dato, si pueden ser nulos, si son únicos en el sistema y un ejemplo de cada uno. También se presentarán las claves primarias (PK) y foráneas (FK) presentes en cada tabla.

Perfiles:

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
userID PK	UID de usuario	String	Sí	No	"41lwFff4nxZPu69VtEOzJdO5FRJ3"
nombre	Nombre del usuario	String	No	No	"Rodrigo"
apellido	Apellido del usuario	String	No	No	"Barragán"
fnac	Fecha de nacimiento del usuario	String	No	No	"2000-12-05"
estudios	Estudios que tiene o esta cursando	String	No	No	"Nutrición Humana y dietética"
trabaja	El usuario trabaja o estudia	Bool	No	No	false
universidad	Universidad donde hizo o está realizando sus estudios	String	No	No	"Universidad de Valladolid"
idiomas	Idiomas que habla el usuario	List(String)	No	No	["ingles", "español"]
sexo	Género del usuario	String	No	No	"hombre"
tipoPersona	Como se considera la persona	String	No	No	"tranquilo"
ambiente	Ambiente que le gustaría en la casa	String	No	No	"tranquilo"
tiempoLibre	Qué le gusta al usuario hacer en su tiempo libre	String	No	No	"Me gusta mucho hacer deporte"
fumar	El usuario fuma o no	Bool	No	No	true
fiesta	Al usuario le gusta la fiesta o no	Bool	No	No	false
descripcion	Descripción de como es el usuario	String	No	No	"Tranquilo pero sociable, risueño, simpático, paciente pero un poco temperamental"
urlImage	url donde se alberga su imagen de perfil	String	Si	No	"images/41lwFff4nxZPu69VtEOzJdO5FRJ3.jpg"

Tabla 5.1: Organización de la tabla de perfiles de la base de datos.

Anuncios:

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
userID FK	UID de usuario	String	Sí	No	"41lwFff4nxZPu69VtEOzJ-dO5FRJ3"
barrio	Zona o barrio donde tiene o quiere vivir	String	No	No	"Centro"
tiempoEstancia	Tiempo que quiere estar de estancia	String	No	No	"4 meses"
presupuesto	Precio o presupuesto máximo del anuncio	String	No	No	"300"
num_hab	En el caso de que tenga piso, el número que disponga de habitaciones	String	No	Sí	"2"

Tabla 5.2: Organización de la tabla de anuncios de la base de datos.

Favoritos:

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
currentUser FK	UID de usuario	String	Sí	No	"41lwFff4nxZPu69VtEOzJ-dO5FRJ3"
anuncioID PK	UID del anuncio al que se da favorito	String	Sí	No	"lgRlA9sFMwMCc-jAhNR3n"

Tabla 5.3: Organización de la tabla de los anuncios favoritos en la base de datos.

Chats: En este caso, la organización es diferente. El identificador del documento en este caso es el mismo identificador único que tiene el usuario que envía el mensaje y dentro de ese documento se crea una colección, que tiene como identificador el mismo que el del usuario que recibe el mensaje. A su vez dentro de esta colección, se crea un documento, ya con identificadores generados automáticamente, donde se guarda la información de cada mensaje. Todo esto sirve para crear las celdas de cada conversación que tiene el usuario abierta, con el nombre del usuario al que se le envía mensajes, su apellido y su foto de perfil

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
fromID PK	UID de usuario que envia el mensaje	String	Sí	No	"41lwFff4nxZPu69VtEOzJ-dO5FRJ3"
toID PK	UID del usuario que recibe el mensaje	String	Sí	No	"Gn5XYW3r27daqjHwGywyD-gNWXNE3"
text	Mensaje del chat	String	No	No	"me gusta tu anuncio"
timestamp	Momento en el tiempo en el que se envia el mensaje	Timestamp	No	No	31 de mayo de 2024, 9:58:20p.m. UTC+2

Tabla 5.4: Organización de la tabla de los chats en la base de datos.

recent_messages: Al igual que los chats, los mensajes recientes tienen una organización diferente. En este caso también se guarda un documento, con el identificador único del usuario que manda el mensaje, luego una colección llamada messages y dentro de esa colección, otro documento, que a diferencia de los chats, esta vez tiene el identificador del usuario al que va dirigido el mensaje. La diferencia más grande, es que en la colección entre medias, solamente se guarda un documento, ya que solamente se guarda el último mensaje de cada chat. Esto sirve para recoger todas las conversaciones del usuario y mostrarlas en la vista de mensajes, por eso a parte del mensaje, que es lo único que se modifica junto al timestamp, los demás datos no vuelven a cambiar desde el momento de su creación.

Campo	Descripción	Tipo	Único	Nulo	Ejemplo
fromID PK	UID de usuario que envia el mensaje	String	Sí	No	“41lwFff4nxZPu69VtEOzJ-dO5FRJ3”
toID PK	UID del usuario que recibe el mensaje	String	Sí	No	“Gn5XYW3r27daqjHwGywyD-gNWXNE3”
text	Mensaje del chat	String	No	No	“me gusta tu anuncio”
timestamp	Momento en el tiempo en el que se envia el mensaje	Timestamp	No	No	31 de mayo de 2024, 9:58:20p.m. UTC+2
nombre	Nombre del usuario al que se le envia el mensaje	String	No	No	“Marta”
apellido	Apellido del usuario al que se le envia el mensaje	String	No	No	“Rivas”
profileImage	URL donde está albergada la foto de perfil del usuario al que se le manda los mensajes	String	Sí	No	“image-s/Gn5XYW3r27daqjHwGywyD-gNWXNE3.jpg”

Tabla 5.5: Organización de la tabla de los mensajes recientes en la base de datos.

En la siguiente imagen, se muestra un diagrama E/R para que se vea gráficamente como queda la base de datos.

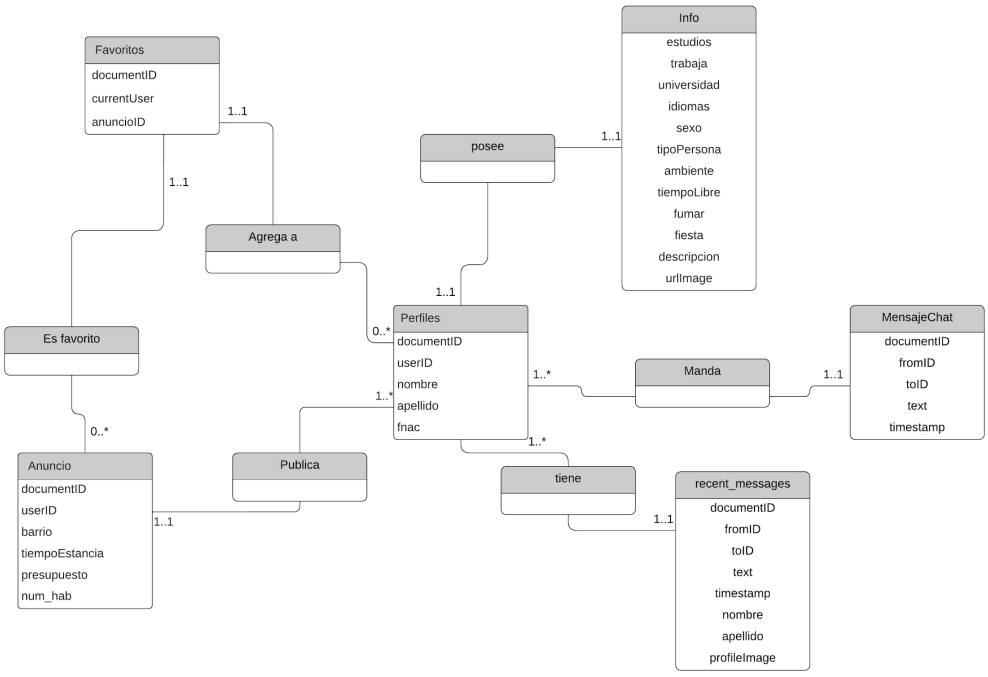


Figura 5.8: Diagrama Entidad-Relación de la base de datos.

5.4 Diseño de interfaces

En esta sección se describen las interfaces de la aplicación, detallando la información presentada y las interacciones posibles. Los diseños originales eran para una aplicación web, según el Trabajo de Fin de Grado «Diseño de una solución que facilite la búsqueda de compañeros de piso en el entorno estudiantil» [3], por lo que tuve que adaptarlos a un diseño móvil. A continuación, se explican brevemente algunos de los componentes nativos más interesantes utilizados en el desarrollo de las interfaces.

- LazyVGrid - Vista que organiza sus elementos hijos en una cuadrícula vertical con un número fijo de columnas y utiliza carga perezosa para mejorar el rendimiento al cargar solo los elementos visibles. En la aplicación, se ha utilizado en la vista de búsqueda para cargar únicamente los anuncios visibles.
- Sheet - Es un modificador en SwiftUI que se utiliza para presentar una vista modal desde el momento en que se activa. Esta vista modal generalmente se muestra encima de la vista principal y se utiliza para presentar contenido adicional o acciones específicas. En mi caso se ha utilizado para mostrar la información del usuario en el momento que se pulsa encima de un anuncio en la pantalla de búsqueda.

5.4.1 Inicio de sesión/registro

Interfaz del splash

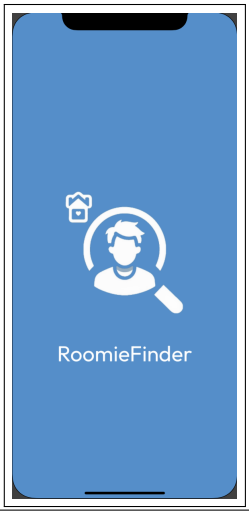
Detalle	Pantalla que aparece en el momento de abrir la aplicación y en la se mantiene hasta que carguen todos los servicios. Se muestra el logo de la aplicación junto con el color principal de la misma.
Eventos	
Captura	

Tabla 5.6: Interfaz del splash.

Interfaz del login

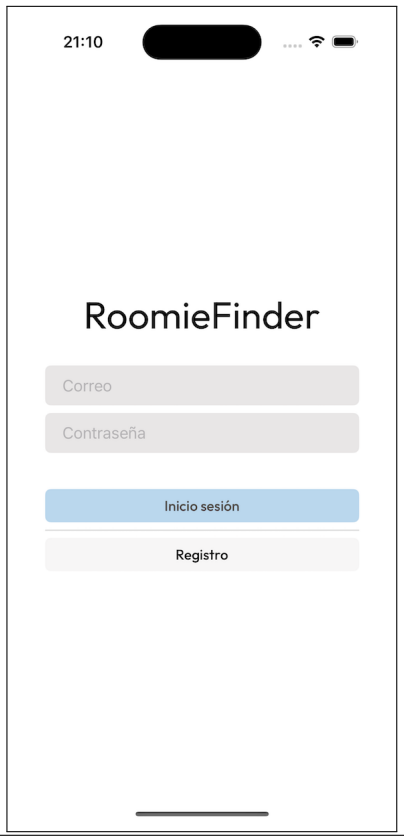
Detalle	Pantalla que se muestra después del splash, que contiene dos campos de texto y dos botones, uno para iniciar sesión que hasta que ambos campos no estén rellenos, no se activa, y el otro para acceder a la pantalla de registro. También se muestra arriba el nombre de la aplicación.
Eventos	▪ Se inicia sesión rellenando los campos y pulsando el botón, y se navegará a la pantalla principal de la aplicación. ▪ Si hay algún error al introducir los datos y no se consigue iniciar sesión, se muestra una alerta informando al usuario. ▪ Si el usuario pulsa el botón de registro, se mostrará la pantalla de registro.
Captura	

Tabla 5.7: Interfaz del login.

Interfaz del registro

Detalle	Pantalla en la que el usuario introduce los datos más importantes para su registro. Tenemos cuatro entradas de texto (nombre, apellido, email y contraseña) y también tenemos a mayores un selector de fechas (DatePicker). En la cabecera tenemos la opción de volver a la pagina de login, ya que tenemos el botón de iniciar sesión.
Eventos	▪ Se registra rellenando todos los campos y dando al botón y el sistema comprueba que los datos son válidos, si lo son, navega a la pantalla de creación de perfil. ▪ Si hay algún error al introducir los datos y no se consigue registrarse, se muestra una alerta informando al usuario. ▪ Si el usuario pulsa el botón de iniciar sesión, volverá a la pantalla de login.
Captura	

Tabla 5.8: Interfaz del registro.

Interfaz de la creación de perfil

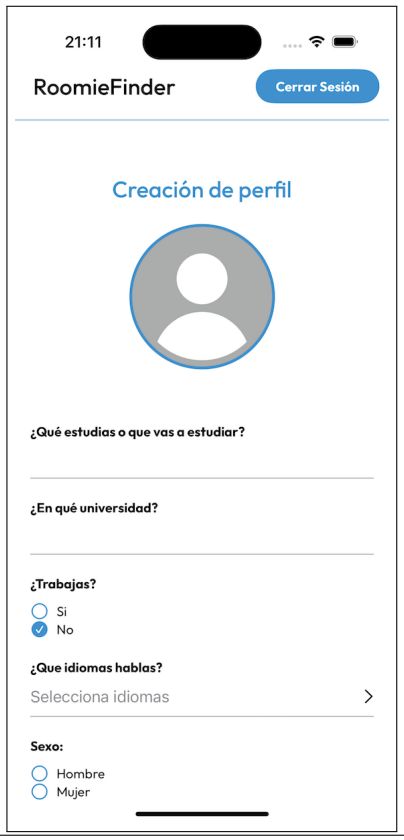
Detalle	Pantalla en la que el usuario introduce su información y su foto de perfil. Se trata de un formulario donde se le preguntan cosas sobre como es como persona, el ambiente que le gustaría en la casa, etc. Todo esto para así luego poder filtrar sobre ello.
Eventos	▪ Se rellenan todos los campos y se pulsa el botón de crear perfil. El sistema valida que los datos sean correctos y navega a la pantalla de creación de anuncio. ▪ Si hay algún error al introducir los datos o falta algo por rellenar, se muestra una alerta informando al usuario. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.9: Interfaz de la creación de perfil.

Interfaz de la creación de anuncio

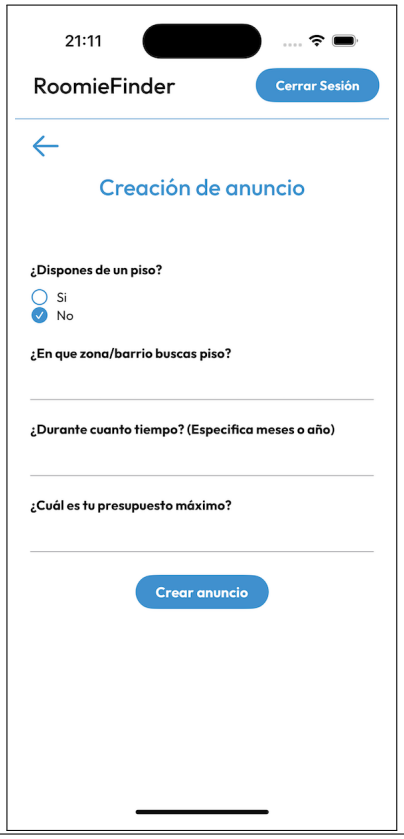
Detalle	Pantalla en la que el usuario introduce la información del anuncio que quiere hacer. Se compone de tres entradas de texto si no dispone de piso. Por el contrario serian los tres que aparecen cuando no disponen (barrio/zona, tiempo de estancia y presupuesto máximo) y a mayores aparece el cuarto (número de habitaciones).
Eventos	▪ Se rellenan todos los campos y se pulsa el botón de crear anuncio. El sistema valida que los datos sean correctos y navega a la pantalla de inicio, que siempre será la pantalla de búsqueda. ▪ Si hay algún error al introducir los datos o falta algo por rellenar, se muestra una alerta informando al usuario. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación. ▪ Si el usuario pulsa el botón de volver, volverá a la pantalla anterior.
Captura	

Tabla 5.10: Interfaz de la creación de anuncio.

5.4.2 Pantallas principales

Interfaz de la búsqueda

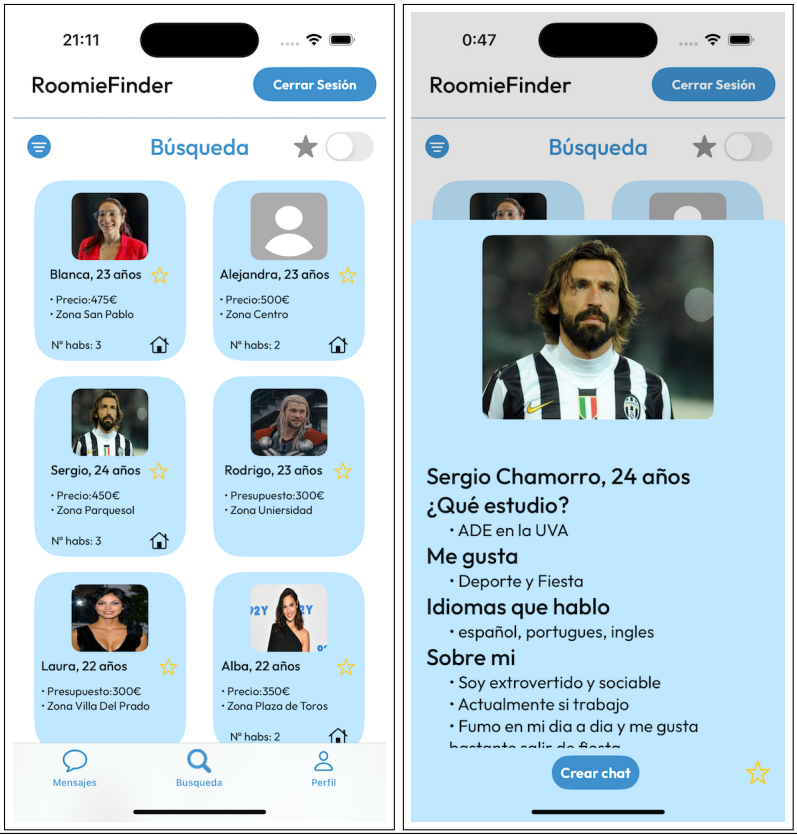
Detalle	Pantalla que muestra todos los anuncios creados por los usuarios de la aplicación, con la opción de filtrar por favoritos y con el botón para navegar a filtros. Arriba está la misma cabecera que en la creación de anuncios y perfiles. Abajo se encuentra una *TabBar* con las pantallas principales.
Eventos	▪ Se pulsa en un anuncio y se despliega la información del usuario. ▪ Al pulsar el switch con la estrella, se filtran los anuncios favoritos del usuario. ▪ Si se pulsa en el botón de filtros, se navega a dicha pantalla. ▪ Si el usuario pulsa una de las opciones de la *TabBar*, navegará a dicha pantalla ▪ Si pulsa en la estrella de algún anuncio, marca o desmarca ese anuncio como favorito. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.11: Interfaz de la búsqueda.

Interfaz de los filtros

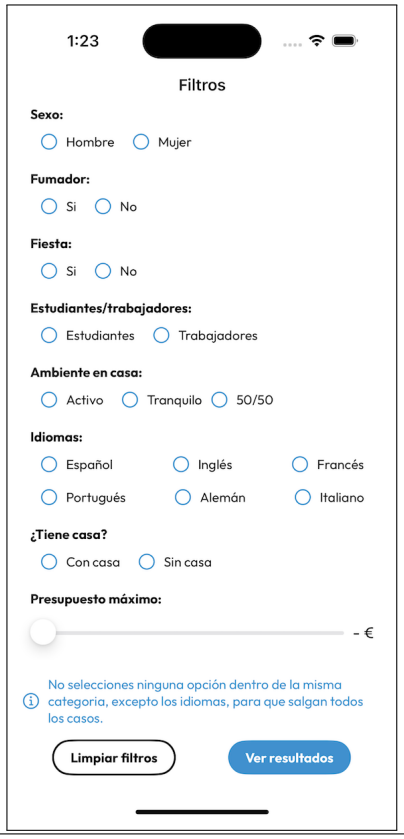
Detalle	Pantalla donde se gestionan todos los filtros que se pueden realizar sobre los anuncios. Son diferentes *checkBox* y una barra para elegir el presupuesto. A mayores hay un texto informativo donde se especifica que en las categorías, excepto en los idiomas, si se quiere que salgan todas las opciones, no se selecciona ninguna. Luego tenemos dos botones, uno para limpiar los filtros y otro para aplicar los filtros seleccionados y navegar de vuelta a la página de búsqueda.
Eventos	▪ Se seleccionan los filtros que se quieren y se pulsa el botón de ver resultados, se navega a la pantalla de búsqueda con los filtros aplicados. ▪ Si se pulsa en el botón de limpiar filtros, se quitan todos. ▪ Si se selecciona todo en una misma categoría, saltará una alerta informando al usuario.
Captura	

Tabla 5.12: Interfaz de los filtros.

Interfaz de los mensajes

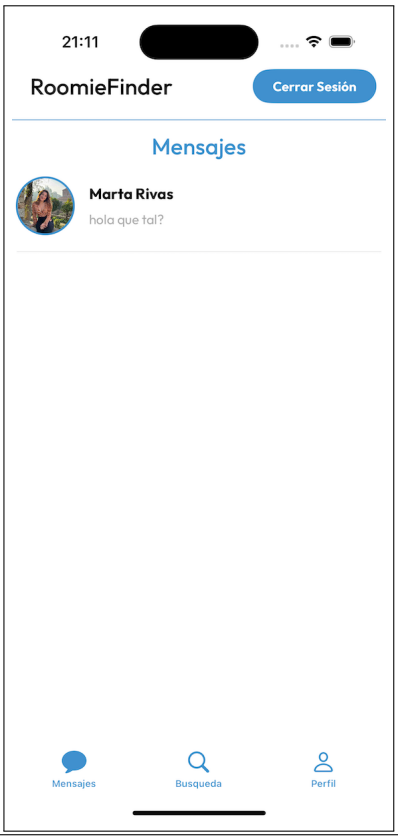
Detalle	Pantalla donde se muestran todas las conversaciones abiertas del usuario. Están ordenadas de más recientes a más antiguas, por lo que en el caso de que le llegue un nuevo mensaje, esta conversación se colocará la primera.
Eventos	▪ Se seleccionan los filtros que se quieren y se pulsa el botón de ver resultados, se navega a la pantalla de búsqueda con los filtros aplicados. ▪ Si se pulsa en el botón de limpiar filtros, se quitan todos. ▪ Si se selecciona todo en una misma categoría, saltará una alerta informando al usuario.
Captura	

Tabla 5.13: Interfaz de los mensajes.

Interfaz de un chat

Detalle	Pantalla donde se muestra la conversación entre dos usuarios. Marcados con azul y a la derecha los mensajes enviados por el usuario que esta utilizando la aplicación y marcados de blanco los que esta recibiendo. Abajo se dispone de un editor de texto y un botón para enviar. Este botón no se activa hasta que se rellene algo en el editor de texto.
Eventos	▪ Se escribe un mensaje y se manda, aparece en la pantalla. ▪ Recibe un mensaje y aparece en la pantalla. ▪ Se pulsa el botón de arriba a la izquierda de volver atrás para salir a la pantalla de mensajes.
Captura	

Tabla 5.14: Interfaz de un chat.

Interfaz del perfil

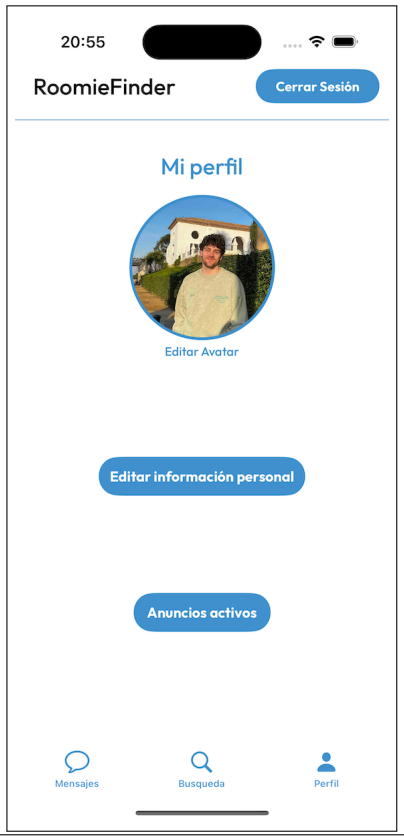
Detalle	Pantalla de perfil, donde se gestiona tanto la foto de perfil, como la información personal y los anuncios activos del usuario.
Eventos	▪ Si el usuario pulsa en la foto de perfil o en el texto ”Editar avatar”, se despliega la galería para elegir la nueva foto de perfil. ▪ Si pulsa el botón de editar información personal, se navega a la pantalla de modificar perfil. ▪ Si pulsa en el botón de anuncios activos, navega a la pantalla de anuncios activos. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.15: Interfaz del perfil.

Interfaz de modificación de información personal

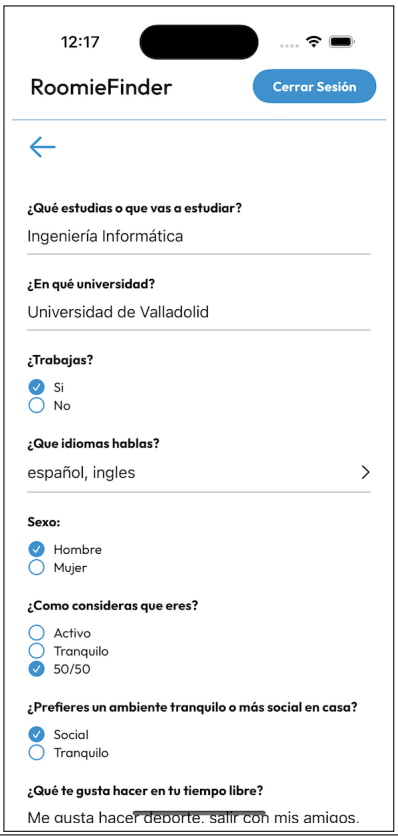
Detalle	Pantalla donde se modifica la información del perfil. Se compone de la misma pantalla que la de creación de perfil, lo único que rellena los campos con la información que ya está guardada y quita la opción de cambiar la foto de perfil, ya que está en la pantalla de perfil.
Eventos	▪ El usuario modifica datos personales, el sistema valida estos datos y actualiza el perfil. ▪ Si el usuario deja algún campo sin completar, saltará una alerta y se notificará al usuario. ▪ Si pulsa la flecha, volverá a la pantalla de perfil, descartando los cambios. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.16: Interfaz de modificar perfil.

Interfaz de anuncios activos

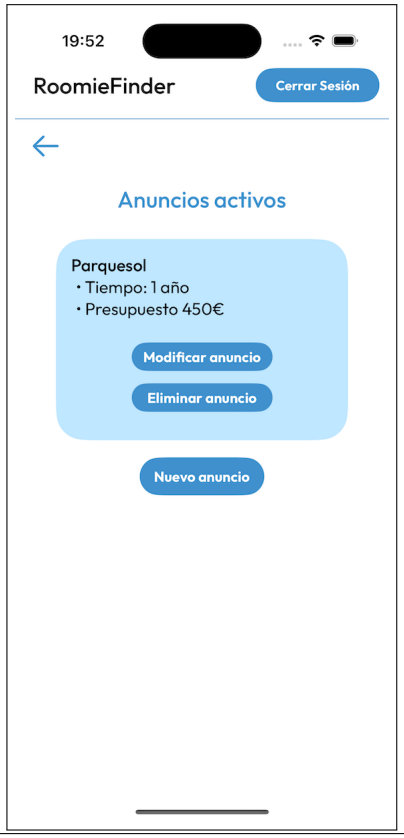
Detalle	Pantalla donde se muestran todos los anuncios creados por el usuario. Cada anuncio tendrá un botón para modificar el anuncio y otro para eliminarlo. También se encuentra el botón para crear un nuevo anuncio.
Eventos	▪ El usuario pulsa el botón de crear un nuevo anuncio y navega a la pantalla de creación de anuncio. ▪ El usuario pulsa el botón de modificar anuncio y navega a la pantalla de modificar anuncio. ▪ El usuario pulsa el botón de eliminar anuncio, se le pide una confirmación al usuario y si acepta, se elimina de la base de datos y se actualiza la pantalla. ▪ Si pulsa la flecha, volverá a la pantalla de perfil, descartando los cambios. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.17: Interfaz de anuncios activos.

Interfaz de modificar anuncios

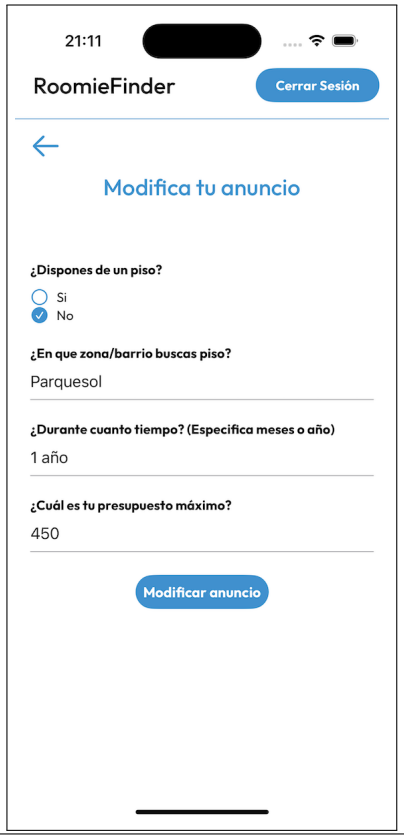
Detalle	Pantalla donde se modifica la información del anuncio. Se compone de la misma pantalla que la de creación de anuncio, lo único que rellena los campos con la información que ya está guardada.
Eventos	▪ El usuario pulsa el botón de guardar, el sistema valida y se actualizan los datos. ▪ Si el usuario deja algún campo sin completar, saltará una alerta y se notificará al usuario. ▪ Si pulsa la flecha, volverá a la pantalla de anuncios activos, descartando los cambios. ▪ Si el usuario pulsa el botón de cerrar, volverá a la pantalla de login y se cierra su sesión dentro de la aplicación.
Captura	

Tabla 5.18: Interfaz de modificar anuncios.

5.4.3 Alertas

Interfaz de una alerta de notificación

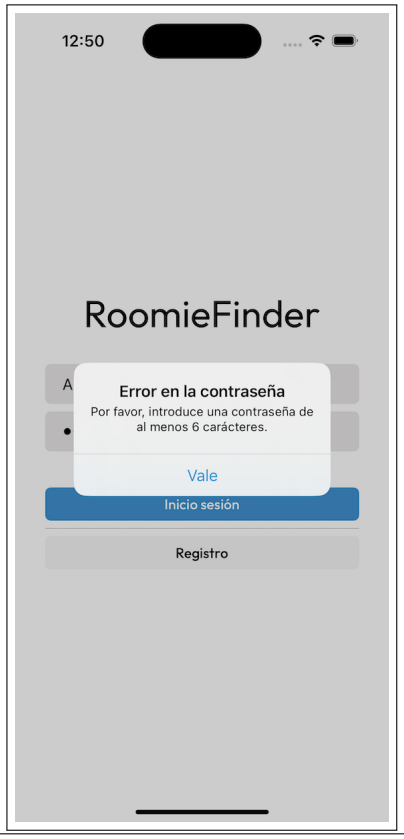
Detalle	Ejemplo de una alerta de notificación utilizada en la aplicación. Dependiendo del caso en el que se encuentre el titulo y el texto cambia. Sirve para notificar de algún error al usuario.
Eventos	▪ El usuario pulsa el botón y se cierra la alerta.
Captura	

Tabla 5.19: Interfaz de una alerta de notificación.

Interfaz de una alerta de confirmación

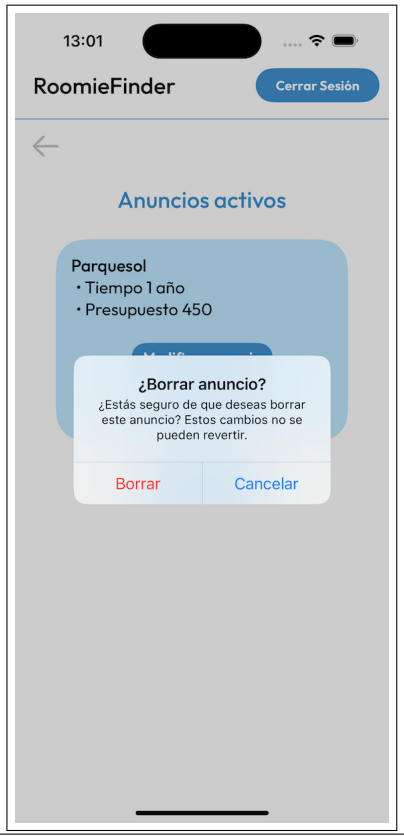
Detalle	Ejemplo de una alerta de confirmación utilizada en la aplicación. Se utiliza solamente en un punto y es en el momento de borrar un anuncio, que la aplicación te pide una confirmación ya que son cambios irreversibles.
Eventos	▪ El usuario pulsa el botón de borrar y confirma la acción. ▪ El usuario pulsa el botón de cerrar y deniega la acción.
Captura	

Tabla 5.20: Interfaz de una alerta de confirmación.

Capítulo 6

Implementación

6.1 Requerimientos de Hardware y Software

En este capitulo se presentarán los requisitos necesarios para poder utilizar la aplicación de forma eficiente.

6.1.1 Hardware

Después de realizar unas pruebas con la herramienta interna de ejecución de Xcode podemos aproximar los requisitos minimos a lo siguiente:

- Espacio en el dispositivo: 300 MB.
- RAM: 300 MB.
- Conexión a internet, ya sea con Wi-Fi o con servicio de datos móviles.

6.1.2 Software

- Sistema Operativo iOS con la versión 16.0 o superior.

Después de ver los datos recogidos, podemos asegurar que cualquier dispositivo compatible con el sistema operativo iOS 16.0 podrá utilizar esta aplicación, asegurando que la mayoría de los usuarios de iPhone y iPad tengan acceso a todas las funciones y características. Esto incluye dispositivos más recientes y algunos modelos más antiguos que aún soportan esta versión de iOS, garantizando una amplia accesibilidad para todos los usuarios.

6.2 Herramientas utilizadas

6.2.1 Herramientas de desarrollo

El desarrollo de la aplicación se ha realizado utilizando Xcode [4]. Xcode es el entorno de desarrollo integrado oficial para la plataforma iOS y macOS, creado por Apple. Consta de diversas herramientas de desarrollo como:

- Editor de código avanzado: Xcode soporta lenguajes de programación como Swift y Objective-C.
- Soporte para construcción basada en Swift Package Manager: Permite una gestión aficiente de dependencias y librerías.
- Herramientas de diagnóstico y análisis: Instrumentos para detectar problemas de rendimiento, uso de memoria, entre otros.
- Simulador de iOS: Permite ejecutar y probar aplicaciones en diferentes dispositivos y versiones de iOS.
- Soporte integrado para servicios de Apple: Facilitando la integración con servicios como iCloud, Apple Pay y Game Center.
- Interfaz de diseño visual: Storyboards y XIBs para diseñar interfaces de usuario de manera visual e intuitiva.

6.2.2 Herramientas utilizadas de soporte

A mayores de lo ya mencionado, se han utilizado las siguientes para realizar el proyecto completo:

- Overleaf: Es una aplicación en linea que simplifica la escritura, edición y colaboración en documentos LaTeX. Se encuentra de gran ayuda para elaborar artículos científicos, tesis, presentaciones y otros documentos técnicos. Con Overleaf, los usuarios pueden colaborar a tiempo real con coautores y obtener una vista previa del documento compilado, lo cual simplifica el proceso de escritura y edición. La plataforma también brinda conexión con herramientas como GitHub y brinda la administración de bibliografías, imágenes y otros recursos de manera eficaz.
- Draw.io: Es una aplicación de diagramación en línea que permite a los usuarios crear una amplia variedad de diagramas, incluidos diagramas de flujo, diagramas de red, organigramas, mapas mentales, diagramas UML y más. Se ha utilizado para todos los diagramas que se han visto durante la memoria.
- GitHub: Es una plataforma de desarrollo colaborativo que utiliza Git para gestionar y controlar versiones de proyectos de software. Permite a los desarrolladores colaborar, revisar código y seguir el progreso de sus proyectos de manera eficiente. En este proyecto se ha utilizado para guardar las versiones del desarrollo [5].

- Figma [6]: Es una herramienta de diseño de interfaz de usuario basada en la web que permite a los equipos colaborar en tiempo real. Es utilizada para crear wireframes, prototipos, y diseños de alta fidelidad, ofreciendo una integración fluida con otros sistemas y servicios. Con Figma, los diseñadores pueden trabajar juntos de manera eficiente, compartir comentarios instantáneamente y gestionar versiones, todo desde una plataforma centralizada.

6.3 Tecnologías utilizadas

- Swift [7]: Es un lenguaje de programación desarrollado por Apple para crear aplicaciones iOS, macOS, watchOS y tvOS. Es moderno, seguro y eficiente, diseñado para ser fácil de usar y leer. Swift combina rendimiento y seguridad, permitiendo a los desarrolladores escribir código robusto y rápido.

- SwiftUI [8]: Es un framework de interfaz de usuario desarrollado por Apple para construir aplicaciones en iOS, macOS, watchOS y tvOS. Permite a los desarrolladores crear interfaces de usuario de manera declarativa, utilizando un código conciso y expresivo. SwiftUI facilita la creación de interfaces consistentes y adaptables en todas las plataformas de Apple.

6.4 Servicios en Firebase

Como se mencionó en el apartado de Arquitectura física, se implementó un arquitectura cliente-servidor, utilizando Firebase [9] para la administración de este último.

Firebase es una plataforma de desarrollo de aplicaciones de Google que brinda diversas opciones y servicios para crear, mejorar y escalar aplicaciones. Se encuentran disponibles características como bases de datos en tiempo real, autenticación, hosting, entre muchas otras. Algunas de las ventajas de usar Firebase son las siguiente:

- Integración rápida y sencilla: Firebase ofrece SDKs fáciles de usar para diferentes plataformas, lo que permite una integración rápida en aplicaciones móviles y web.

- Bases de datos en tiempo real: La base de datos de Firebase permite la sincronización instantánea de datos entre clientes y servidores, lo que me ha venido genial para toda la parte de los chats.

- Autenticación segura: Proporciona un sistema de autenticación robusto que soporta varios métodos, como email y contraseña, Google, Facebook, Twitter y más.

A continuación, se van a explicar las diferentes herramientas utilizadas, así com una breve explicación de la integración de Firebase al proyecto.

6.4.1 Creación del proyecto en Firebase

Antes de poder integrar Firebase en el proyecto, es necesario crear un proyecto en Firebase. A continuación, se detallan los pasos para hacerlo:

1. Accedemos a Firebase Console y escogemos la opción de Agregar proyecto.

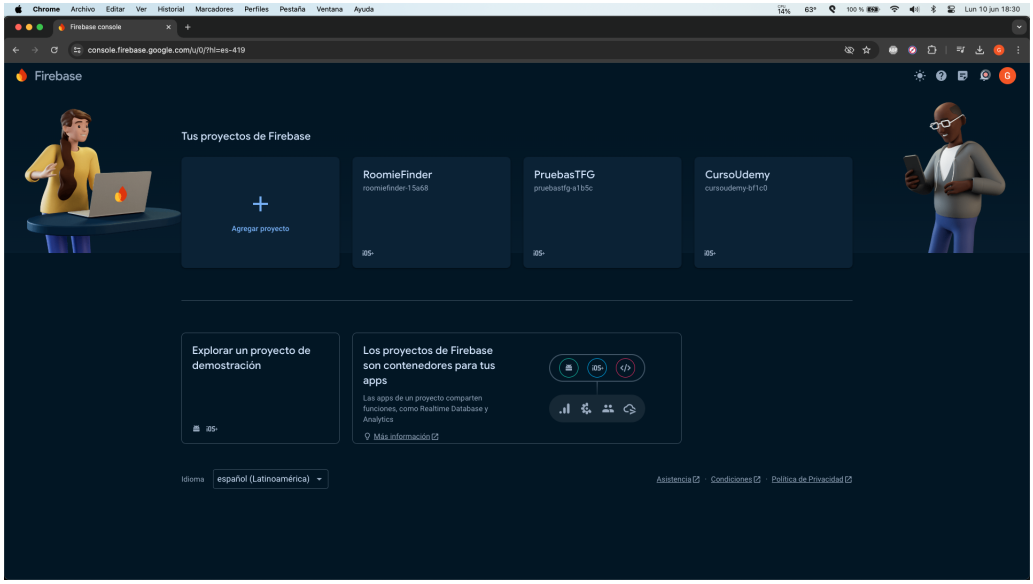


Figura 6.1: Paso 1 para la creación del proyecto de Firebase.

2. Ingresamos el nombre que queremos para el proyecto.

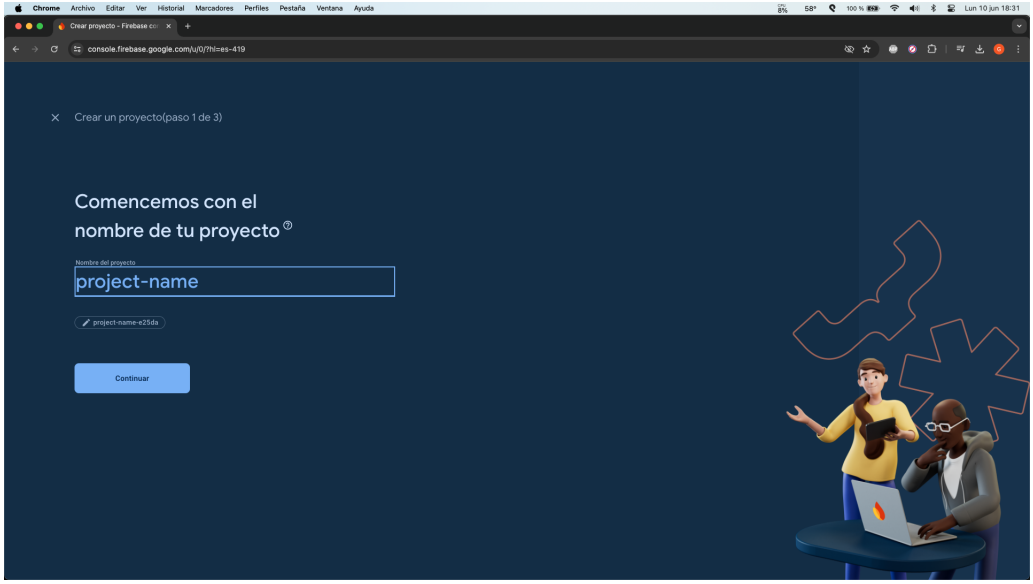


Figura 6.2: Paso 2 para la creación del proyecto de Firebase.

3. Este punto es opcional, que es el añadir Google Analytics para el proyecto.

   Google Analytics es una herramienta gratuita de Google que permite rastrear y analizar el tráfico y comportamiento de usuarios en sitios web y aplicaciones. Proporciona datos detallados para optimizar el rendimiento y la experiencia del usuario.

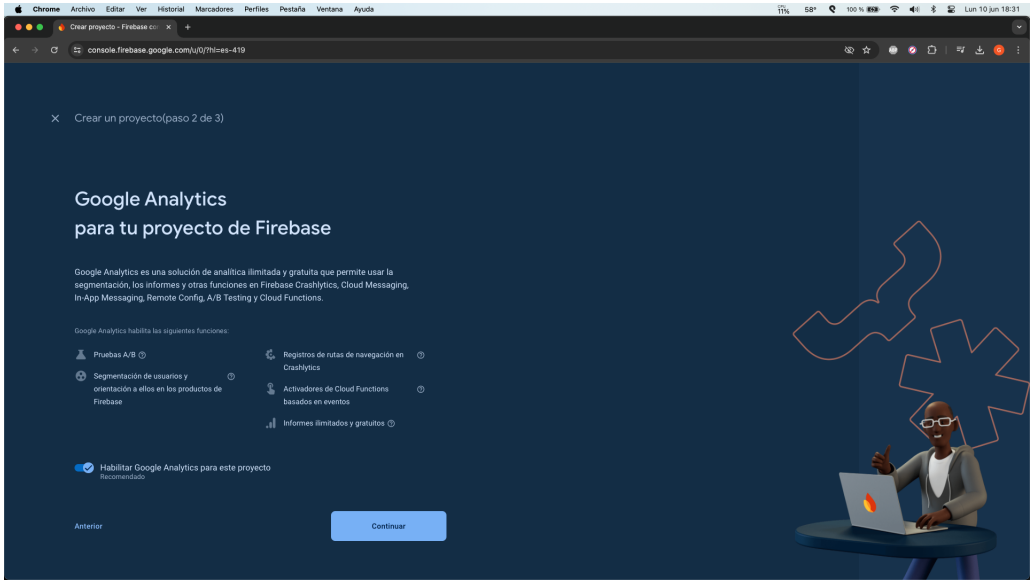


Figura 6.3: Paso 3 para la creación del proyecto de Firebase.

4. Pulsamos en Crear proyecto. Firebase hará la configuración para aprovisionarte de los recursos de forma automática. Cuando esto termine, le llevará a la página donde se ve la descripción general del proyecto que se acaba de crear

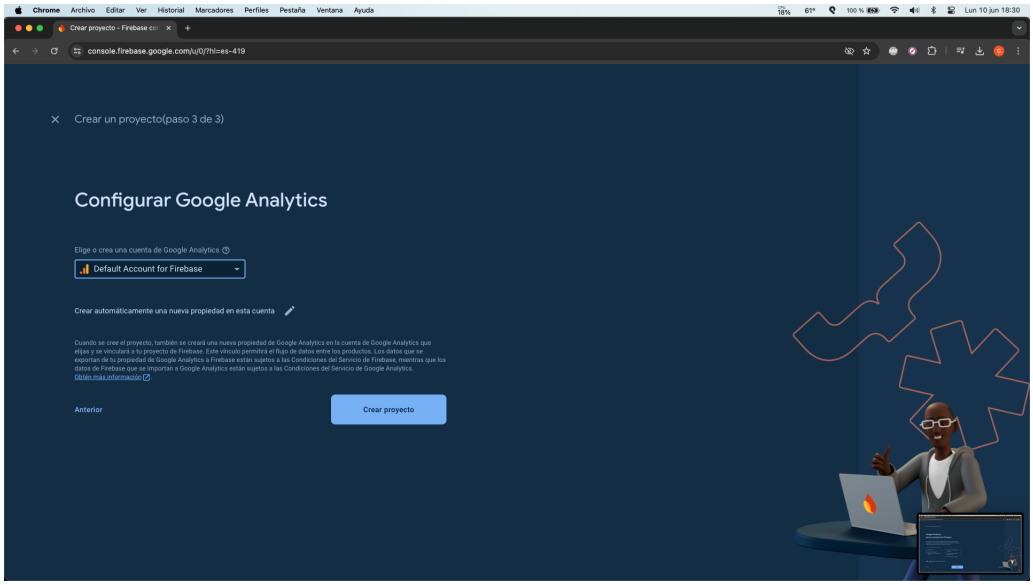


Figura 6.4: Paso 4 para la creación del proyecto de Firebase.

Después de haber creado dicho proyecto, vamos a proceder a registrar la aplicación en el este.

1. Nos dirigimos a Firebase Console y entramos en nuestro proyecto que hemos creado.
2. En la página de descripción general del proyecto, pinchamos en el botón en el que pone agregar aplicación y elegimos la tecnología que queramos. En nuestro caso será iOS.

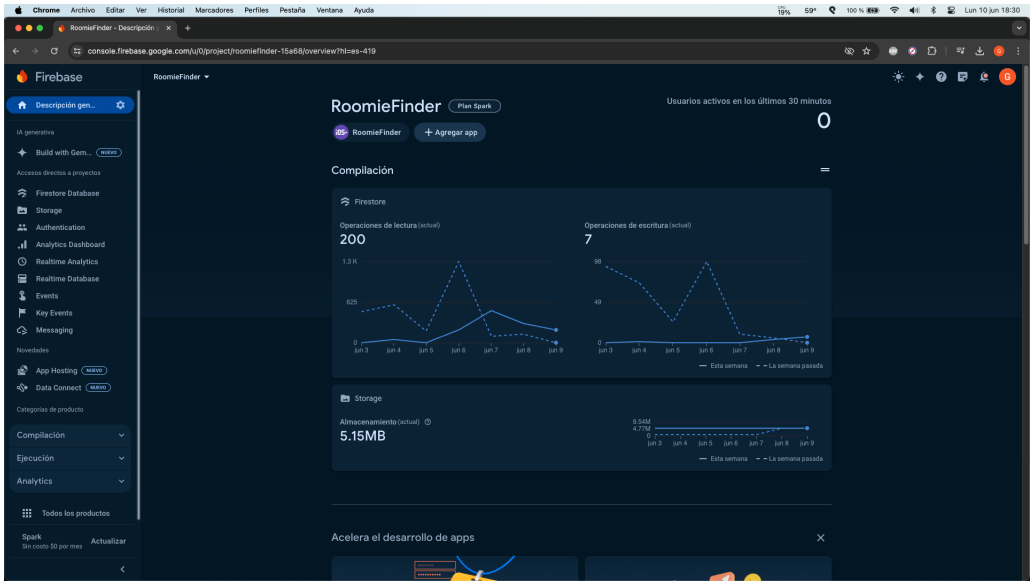


Figura 6.5: Paso 2 para el registro de la aplicación en en el proyecto Firebase.

3. Hay que ingresar el id del paquete de Apple, esto se encuentra en el archivo *.xcodeproj* de tu proyecto en Xcode. A mayores rellenas los siguientes campos y hacemos clic en Registrar aplicación.

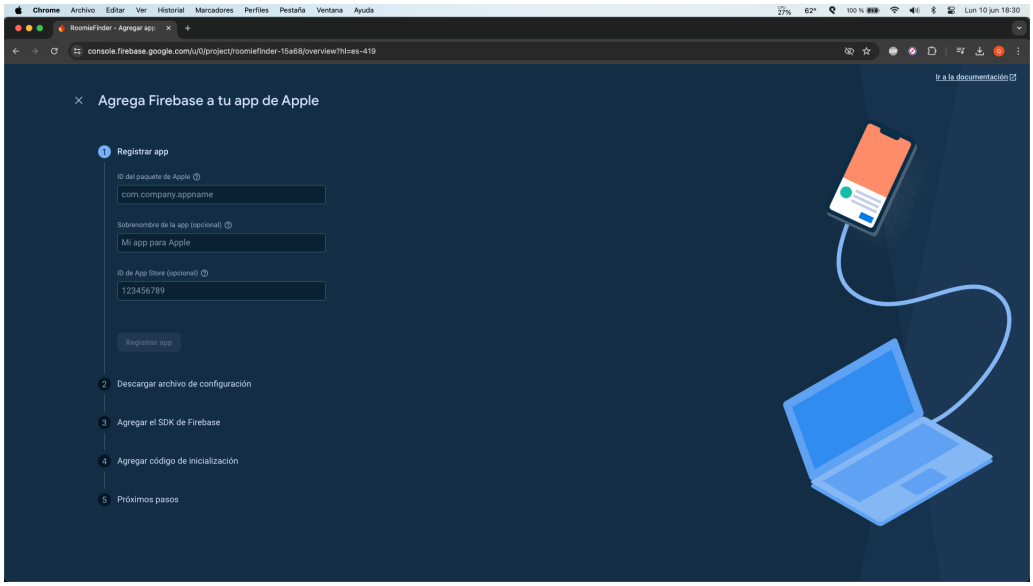


Figura 6.6: Paso 3 para el registro de la aplicación en en el proyecto Firebase.

Una vez registrada, hay que agregar Firebase a nuestro proyecto en Xcode y añadir los SDKs.

1. Agrega el archivo de configuración de Firebase para iOS a la aplicación. Para realizar esto, descarga *GoogleService-Info.plist* para obtener el archivo de configuración de Firebase para iOS y transfiere el archivo al directorio raíz del proyecto de tu aplicación.
2. Utilizando *Swift Package Manager*, obtenemos el paquete de Firebase, donde estarán todas las herramientas de Firebase. Podemos seleccionar todas o solamente las que necesitemos.
3. Realizar la configuración en el código para que en cuanto la aplicación se inicie, también se inicen los servicios de Firebase implementados.

```
7
8  import SwiftUI
9  import Firebase
10
11 @main
12 struct RoomieFinderApp: App {
13
14     @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate
15
16     var body: some Scene {
17         WindowGroup {
18             LoginView(LoginViewModel())
19                 .environmentObject(GlobalViewModel.shared)
20         }
21     }
22 }
23
24 class AppDelegate: NSObject, UIApplicationDelegate {
25     func application(_ application: UIApplication,
26                      didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
27         FirebaseApp.configure()
28         return true
29     }
```

Figura 6.7: Paso 3 para el registro de la aplicación en en el proyecto.

6.4.2 Firebase Auth

Firebase Auth es un servicio de autenticación que permite a los desarrolladores autenticar usuarios de manera sencilla y segura utilizando solo código del lado del cliente. Soporta múltiples proveedores de inicio de sesión como Google, Facebook, Twitter, GitHub, Yahoo y Microsoft, además de métodos tradicionales como correo electrónico y contraseña. También ofrece un sistema de administración de usuarios que facilita la gestión y almacenamiento seguro de las credenciales de los usuarios en Firebase.

Utilizamos este servicio para gestionar todo el sistema de autenticación, ya que mejora la incorporación, el acceso y la seguridad de los usuarios en comparación con los métodos de autenticación tradicionales. Firebase proporciona una manera sencilla, eficaz y segura de gestionar usuarios.

6.4.3 Firebase Storage

Firebase Storage es un servicio de almacenamiento en la nube que proporciona una solución escalable y segura para almacenar y compartir archivos estáticos, como imágenes, videos, archivos de

audio y documentos. Permite a los desarrolladores almacenar estos archivos de manera fácil y eficiente, acceder a ellos desde cualquier lugar y compartirlos con otros usuarios de la aplicación. Firebase Storage integra funciones avanzadas de seguridad y permite controlar el acceso a los archivos mediante reglas de seguridad personalizadas. Además, ofrece un rendimiento rápido y confiable gracias a su infraestructura global de servidores.

En el proyecto, utilizamos este servicio para guardar todas las fotos de perfil de cada usuario de la aplicación.

6.4.4 Firebase Cloud Firestore

Firebase Cloud Firestore es un servicio de base de datos NoSQL en tiempo real y totalmente administrado, ofrecido por Google Firebase. Permite a los desarrolladores almacenar, sincronizar y consultar datos para aplicaciones web, móviles y de servidor en tiempo real. Firestore está diseñado para escalar automáticamente y admitir grandes volúmenes de datos, ofreciendo una sincronización en tiempo real y una integración fácil con otras herramientas de Firebase. Además, proporciona una estructura de datos flexible con colecciones y documentos que permite organizar y consultar datos de manera eficiente.

Durante el proyecto se ha utilizado para guardar toda la información relevante que vimos en el apartado Diseño de la base de datos.

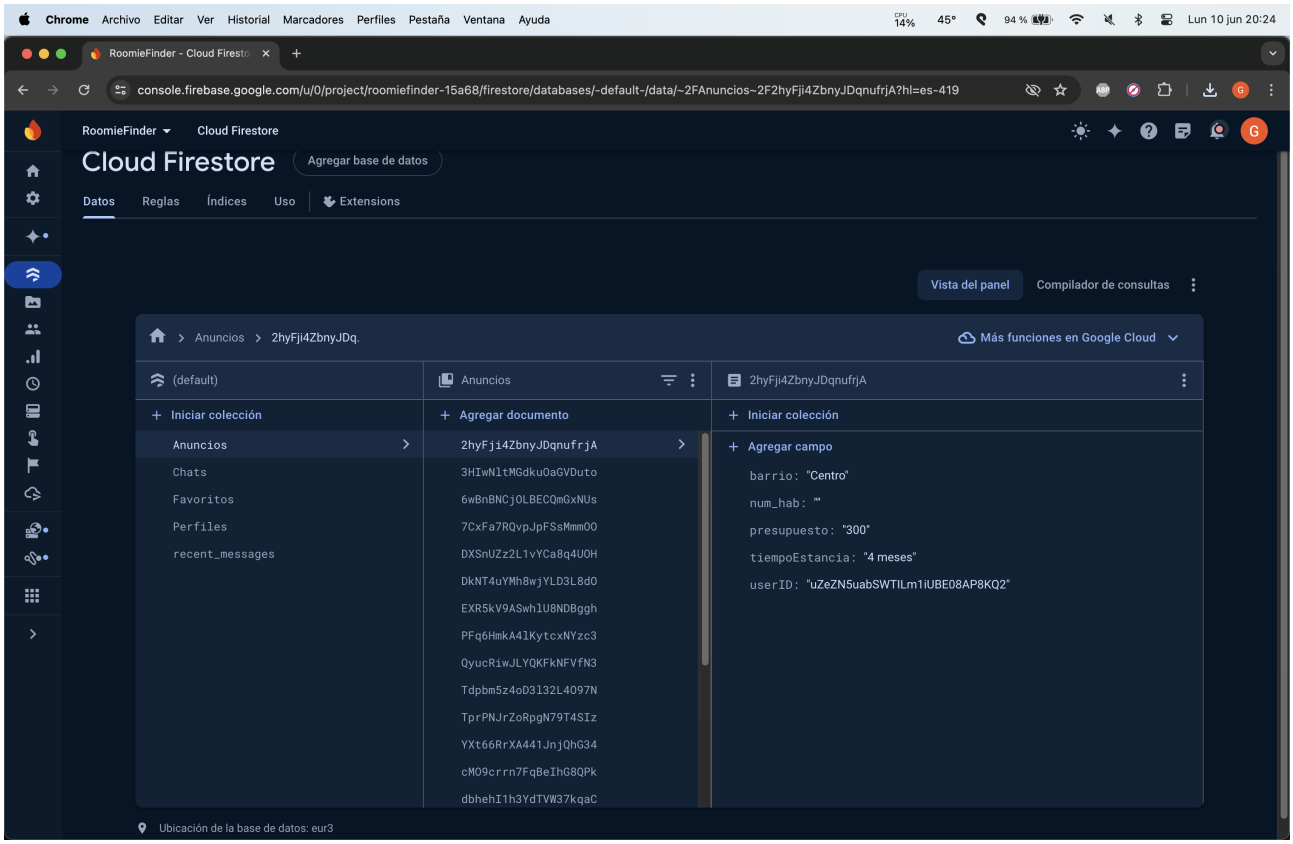


Figura 6.8: Pantalla de Cloud Firestore Database.

6.5 Organización del proyecto

Ahora se va a explicar la estructura interna dentro del proyecto (en el entorno de programación), es decir, como se ha organizado los archivos en sus carpetas, para facilitar la localización de cada archivo.

La carpeta principal es RoomieFinder/ y dentro de ella se encuentran:

- ../Architecture - Dentro de este directorio se encuentran dos diferentes carpetas más, Extensions y Utils.
  - La carpeta Extensions sirve para todos los archivos de código que sean extensiones de código creados por mi. Un ejemplo es el archivo *DoubleExtension.swift*. Esta extensión se hace sobre todas las variables de tipo Double, a la que puedo ponerle la función que yo he creado llamada *toInt()*, que como su nombre indica, transforma una variable Double a Int.
  - La carpeta Utils contiene tanto las constantes que se usan durante el proyecto, ya sean cadenas de texto o un color. También contienen todas las funciones y gestión de llamado a la base de datos.
- ../Components - Dentro de esta carpeta se encuentran todos los componentes que he tenido que crear customizables, ya que algunos de los nativos no completaban los requisitos que yo necesitaba y necesitaba hacerlos personalizados. Cada componente dentro tiene su propia carpeta. Algún ejemplo es el *MessageBubble*, que es el código que crea cada burbuja con el mensaje en los chats.
- ../Data - Carpeta donde se guardan todos los modelos de datos que tenemos en la aplicación. También esta el archivo *GlobalViewModel*, que es un archivo donde se guarda toda la información en local durante toda la aplicación.
- ../Resources - Carpeta donde se guardan todos los recursos visuales, ya sean las fuentes, los colores o las imágenes dentro de la aplicación.
- ../UseCases - Carpeta más grande, ya que es donde se guardan cada pantalla diferente, con su vista y su *ViewModel*. Está subdivido por carpetas refiriendo a cada caso de uso, como su nombre indica (Ej: Login, Mensajes, Registro, etc.)

Capítulo 7

Pruebas

7.1 Prueba de usuarios

Durante el proceso del proyecto, se han ido realizando pruebas de funcionalidad, aunque sobre todo se han hecho en las últimas etapas. Se ha mostrado la aplicación a familiares y amigos para ver el feedback que daban y lo intuitivo que les resultaba utilizarlo. Sobre estas pruebas se obtuvieron unos datos para mejorar en algunos aspectos y es lo que se va a explicar a continuación:

- Mejoras en los filtros: En un principio, si seleccionabas todos los idiomas, la aplicación no te dejaba buscar ya que reconocía que todos los idiomas estaban seleccionados. Esto se arregló, ya que puede caber la casualidad de que algún usuario busque un compañero que hable todos y cada uno de los idiomas.

- Foto de perfil: En la creación de perfil, en un principio si no seleccionabas ninguna de la galería, en la base de datos, la url de este usuario quedaba en blanco y luego no se podía actualizar nunca. Esto se arreglo comprobando que si el usuario no ha metido ninguna foto de perfil de su galería, se quede una foto de perfil por defecto.

- Tamaños de las fotos de perfil: Cuando se muestra la información del usuario al pinchar sobre su anuncio, muchos usuarios veían demasiado dimensionada la foto de perfil y perdía el aspecto original. Esto se solvento manteniendo la relación de la foto y haciéndola más pequeña.

- Favorito a un usuario: Se vio que cuando se creo en un principio la búsqueda, se daba favorito a un usuario, no a un anuncio. Por lo que cuando yo daba favorito, todos los anuncios de ese usuario, se marcaban como favorito. Se corrigió dando favorito al anuncio y no al usuario.

- Favorito en información de perfil: En un principio, el botón para añadir a favoritos un anuncio, estaba solamente en la pantalla de búsqueda. El problema es que muchos de los usuarios querían mirar primero la información del propietario del anuncio antes de dar favorito. Esto se corrigió

y se añadió también otro botón dentro de la vista modal de la información de favoritos, para que tengan ambas opciones para añadir a favoritos, ya sea fuera de la vista modal o dentro de ella.

Al ser una aplicación iOS, no se ha podido distribuir por lo que las pruebas han sido más reducidas de lo normal en una aplicación móvil.

7.2 Pruebas de caja blanca

Las pruebas de caja blanca constituyen un enfoque de evaluación de software, en el cual el evaluador posee conocimiento del código fuente interno y la estructura del programa. Este tipo de pruebas se enfoca en verificar el flujo lógico y las estructuras internas del software, incluyendo la cobertura del código, las ramas y los caminos lógicos. Los objetivos fundamentales son detectar errores en el código, optimizar el rendimiento y asegurar que todas las rutas posibles de ejecución se comporten de la manera esperada. Las pruebas de caja blanca requieren de habilidades técnicas y programáticas, ya que implican una revisión detallada del código y la lógica subyacente.

A lo largo de la implementación del código, se han realizado estas pruebas de caja blanca, comprobando que los caminos independientes de cada método se lleven a cabo por lo menos una vez, que se ejecuten los bucles en sus límites de manera correcta y las estructuras internas de datos para asegurar su validez.

7.3 Pruebas de caja negra

Las pruebas de caja negra son un método de prueba de software en el que el evaluador verifica la eficacia del sistema sin tener en cuenta su estructura interna o código. Se fundamentan en las especificaciones y requisitos, proporcionando entradas y examinando las salidas para confirmar que el software cumple con las expectativas y funciona correctamente. Algunas de las pruebas que se han hecho son las siguientes:

Descripción de la prueba	En la pantalla de inicio de sesión, se introducen las credenciales y se pulsa el botón de iniciar sesión.
Resultado esperado	El usuario es autenticado e ingresa a la pantalla principal.
Resultado obtenido	Tras pulsar el botón, el sistema comprueba las credenciales y le redirecciona a la pantalla principal.
Resultado de la prueba	Satisfactorio.

Tabla 7.1: Prueba de inicio de sesión.

Descripción de la prueba	En la pantalla de anuncios activos, se pulsa sobre el botón de borrar anuncio de uno de ellos.
Resultado esperado	El anuncio se borra de la base de datos y se actualiza la pantalla.
Resultado obtenido	Tras pulsar el botón, vemos que en la base de datos se ha eliminado y la pantalla se ha actualizado.
Resultado de la prueba	Satisfactorio.

Tabla 7.2: Prueba de borrado de anuncio.

Descripción de la prueba	En la pantalla de registro, se deja algún campo sin rellenar.
Resultado esperado	El sistema muestra una alerta avisando al usuario del error.
Resultado obtenido	Tras pulsar en el botón de continuar, el sistema comprueba los campos, ve que hay uno sin rellenar y muestra una alerta.
Resultado de la prueba	Satisfactorio.

Tabla 7.3: Prueba de alerta registro.

Descripción de la prueba	En la pantalla de búsqueda, se tira hacia abajo de la pantalla para refrescar los anuncios y a demás borrar los filtros.
Resultado esperado	El sistema hace una petición a la base de datos y elimina todos los filtros que estaban impuestos, exceptuando el filtro de favorito.
Resultado obtenido	Tras hacer el gesto para refrescar, el sistema hace una correcta petición a la base de datos y elimina los filtros que había, sin quitar el de favorito.
Resultado de la prueba	Satisfactorio.

Tabla 7.4: Prueba de refrescar la pantalla.

Descripción de la prueba	En un chat con otro usuario, en el momento que el otro usuario manda un mensaje, a nosotros nos llega instantáneamente.
Resultado esperado	El sistema recibe el mensaje del usuario, lo recoge de la base de datos y lo muestra por pantalla.
Resultado obtenido	Tras enviar este mensaje, se actualiza correctamente la pantalla y se muestra dicho mensaje.
Resultado de la prueba	Satisfactorio.

Tabla 7.5: Prueba de envío de mensaje.

Descripción de la prueba	Cuando un usuario quiere modificar su foto de perfil, en el momento que elige y acepta una nueva de la galería, se actualiza en la base de datos y se actualiza en la vista de perfil.
Resultado esperado	La base de datos recibe la nueva imagen, la actualiza eliminando la anterior y el sistema muestra la nueva foto de perfil.
Resultado obtenido	Tras cambiar la imagen de perfil, vemos como en la base de datos, se actualiza dicha foto y también vemos como se actualiza en la vista de perfil.
Resultado de la prueba	Satisfactorio.

Tabla 7.6: Prueba de actualización de perfil.

Descripción de la prueba	Cuando el usuario quiere crear un nuevo anuncio, rellena los datos y le da a guardar y se crea en la base de datos.
Resultado esperado	El sistema comprueba que los datos son correctos, crea el anuncio en la base de datos y lo muestra por pantalla.
Resultado obtenido	Tras rellenar y pulsar el botón de crear anuncio, el sistema comprueba y crea un nuevo anuncio en la base de datos, mostrando así el nuevo anuncio en la aplicación.
Resultado de la prueba	Satisfactorio.

Tabla 7.7: Prueba de creación de anuncio.

Se han llevado a cabo una serie de pruebas adicionales con el fin de garantizar la excelencia y el rendimiento general del sistema. Estas comprobaciones abarcan diversos aspectos de la aplicación, con el objetivo de asegurar una experiencia de usuario óptima y una funcionalidad robusta en diferentes escenarios. Son las siguientes:

- Para la comprobación de los aspectos visuales, lo que se ha hecho es escalar los componentes y las pantallas para ver como se comportan.
- Se ha verificado la eficacia del filtrado de anuncios con el fin de asegurar que los filtros utilizados se ajustaron adecuadamente en diversos contextos. Se comprobó que los anuncios mostrados cumplían con los criterios seleccionados por el usuario y se verificó su funcionamiento en diferentes dispositivos. Asimismo, se llevó a cabo una evaluación minuciosa de la precisión y rapidez del filtrado, con el fin de garantizar la gestión adecuada de los casos sin resultados y los errores que puedan surgir durante el proceso.
- Se ha verificado la adaptabilidad de las interfaces para confirmar que se ajusten adecuadamente en diversos dispositivos y tamaños de pantalla. Se constató que los componentes de la interfaz han sido redimensionados y reorganizados con el fin de proporcionar una experiencia consistente y funcional en diversos dispositivos móviles. Se evaluó, además, la legibilidad del contenido, la navegación fluida y la disposición de los elementos interactivos para garantizar una usabilidad óptima en todos los casos.

Capítulo 8

Instalación

En este apartado se va a explicar como se realiza la instalación de la aplicación, tanto si se hubiera subido a la App Store, como en el caso actual sin el certificado de desarrollador de Apple.

8.1 App Store

La App Store de Apple es el principal medio para distribuir aplicaciones en dispositivos iOS, brindando una forma más eficaz y segura de descargar e instalar aplicaciones en estos dispositivos.

No obstante, como se ha mencionado previamente, el registro como desarrollador en Apple y la publicación de aplicaciones en la App Store implican costos elevados. Para subir una aplicación a la App Store, se necesita una cuenta de desarrollador de Apple y un certificado válido. Además, cada aplicación requiere un ID de Bundle y un Provisioning Profile. La aplicación debe estar desarrollada utilizando herramientas compatibles con iOS y cumplir con las directrices de diseño de Apple. Se deben proporcionar iconos, capturas de pantalla y metadatos precisos. Antes de enviarla, la aplicación debe ser probada y certificada para asegurar su calidad y compatibilidad. Una vez cumplidos estos requisitos, se puede iniciar el proceso de envío a través del portal de desarrolladores de Apple. Debido a esto, nuestra aplicación no está disponible en la tienda en estos momentos.

No obstante, deseamos explorar esta opción en el futuro para distribuir nuestra aplicación a través de la App Store, aunque requiere tener en cuenta estos posibles costos.

1. Abrir la App Store y buscar la aplicación por su nombre “RoomieFinder”

2. Instalar la aplicación.

8.2 Instalación a través de Xcode

Para instalar una aplicación a través de Xcode, es necesario que el dispositivo iOS esté habilitado para el desarrollo. Esto se logra mediante el proceso de activación del modo de desarrollador en el dispositivo. Una vez que el dispositivo se encuentra en funcionamiento como desarrollador, Xcode puede ponerse en contacto con él y posibilitar la instalación de aplicaciones de desarrollo directamente desde el entorno de desarrollo. Para la instalación se requiere conectar un dispositivo iOS al ordenador, abrir el proyecto de la aplicación en Xcode y seleccionar el dispositivo como lugar de implementación. A continuación, se compila y se instala la aplicación en el dispositivo conectado. Durante esta etapa, Xcode se encarga de elaborar el código fuente, generar el archivo de la aplicación y transferirlo al dispositivo. Una vez que haya completado la instalación, la aplicación se encuentra disponible en el dispositivo para su uso. Este método se utiliza con frecuencia por desarrolladores para explorar y depurar aplicaciones en dispositivos reales antes de introducirlas al mercado.

Capítulo 9

Conclusiones y mejoras a futuro

9.1 Mejoras a futuro

En esta sección se van a mostrar algunas posibles mejoras que se podrían realizar si el desarrollo de la aplicación llegara más lejos que este proyecto.

- Notificaciones push: Al llegar mas lejos con el proyecto, invertiríamos en una cuenta de desarrollador de Apple para poder subir la aplicación. Con esto también nos abre la puerta de poder realizar notificaciones push desde la nube. Esto lo podríamos hacer para los mensajes de los chats, si alguien ha dado favorito a tu anuncio, etc.
- Desarrollo de un nuevo usuario: La idea es añadir un nuevo tipo de usuario en la aplicación, los propietarios. Este usuario tendría su propia interfaz específica y se encargaría de subir anuncios de sus propiedades en la aplicación. Los usuarios que buscan un compañero de piso y no disponen de una vivienda podrían buscar estos anuncios dentro de la misma aplicación. Además, podrían ver diferentes opciones y comenzar un chat con los propietarios para obtener más información o concretar una visita.
- Mejora de interfaz: Actualmente, la interfaz cumple con lo necesario y es perfectamente funcional, algunas ideas podrían mejorarla:
    - Modo oscuro: Actualmente, la aplicación solamente cuenta con el modo claro, por lo que sería una buena idea añadir este modo oscuro, que esto sería sobre todo en la paleta de colores, tener en cuenta este modo.
    - Animaciones: A algunas transiciones entre ventanas o a algunas acciones se le podrían añadir estas animaciones para hacer la aplicación más llamativa.
    - Sonido: Un poco ligado con las notificaciones, se podrían añadir algún sonido para cuando te llega un mensaje, cuando le das favorito, cuando se han guardado los datos, etc.

- Desarrollo de WEB/Android: En el futuro, se pueden generar versiones tanto para una página web (para ordenadores) como para dispositivos con sistema operativo Android. Esto requeriría un proyecto completamente distinto y requeriría una suma considerable de trabajo. Para llevarlo a cabo, sería necesario establecer nuevos clientes para estas versiones, aunque se podría mantener el servidor actual de Firebase en todas las plataformas.
- Publicidad en la aplicación: Implementar un sistema de publicidad para generar ingresos. Esto podría involucrar la integración de una API específica para mostrar anuncios dentro de la aplicación. Habría que investigar las opciones disponibles, evaluar su efectividad y considerar cualquier costo asociado. Esta mejora permitiría monetizar la aplicación sin cobrar a los usuarios directamente.

9.2 Conclusiones

Desde mi opinión técnica, este proyecto me resultó sumamente complejo desde mi perspectiva técnica. Mi formación académica se enfocó exclusivamente en la creación de aplicaciones para dispositivos móviles o iOS. En consecuencia, tuve que reiniciar mi aprendizaje para usar el marco de Apple y comprender el lenguaje de programación necesario. Esta experiencia me ha enseñado el valor del compromiso y la persistencia para adquirir nuevas aptitudes e conocimientos.

El uso de Firebase también supuso otra barrera en cuanto a los servicios, ya que no tenía experiencia previa con esta plataforma. Pero después de mucho ensayo y error, pude llegar a dominarla y utilizar sus funciones para crear el programa. Al poner en práctica Firebase, pude incorporar funciones esenciales como el almacenamiento en la nube, el análisis en tiempo real y la autenticación de usuarios, lo que mejoró enormemente la experiencia del usuario y la productividad del desarrollo.

Me complace decir de que los objetivos del proyecto se han cumplido a pesar de las dificultades. La aplicación producida satisface tanto los requisitos funcionales como los de rendimiento, establecidos al inicio del proyecto. Este logro no sólo representa el éxito técnico del proyecto, sino también el desarrollo personal y profesional que he experimentado durante este proceso. La capacidad de adquirir nuevas habilidades y utilizarlas bien ha sido una lección útil e inspiradora para trabajar en el futuro.

A nivel más personal, este proyecto me ha enseñado mucho sobre la gestión del tiempo. Suelo ser una persona desorganizada que deja las cosas para última hora. Pero debido a las responsabilidades y la envergadura de este trabajo, tuve que alterar mi rutina. Me enseñó a programar con antelación, establecer objetivos intermedios y cumplir un calendario rígido. Pude terminar el proyecto con éxito gracias a esta nueva estrategia, que también aumentó mi productividad en otras esferas de mi vida. Ahora reconozco el valor de la gestión eficaz del tiempo y la organización.

Bibliografía

1. Mancuzo, G. Qué es el modelo incremental. *ComparaSoftware Blog*. https://blog.comparasoftware.com/que-es-el-modelo-incremental/ (2021).
2. Mathur, A. MVVM in iOS Swift. *Medium*. https://medium.com/@abhilash.mathur1891/mvvm-in-ios-swift-aa1448a66fb4 (2020).
3. Jankowski, M. Á. *Diseño de una solución que facilite la búsqueda de compañeros de piso en el entorno estudiantil* Proyecto fin de grado (Facultad de Ingeniería, Universidad de Deusto, 2023).
4. *Xcode* https://developer.apple.com/documentation/xcode (2024).
5. *Repositorio Github* https://github.com/Guiller300100/RoomieFinder (2024).
6. *Figma* https://help.figma.com/hc/es-es (2024).
7. *Swift* https://www.swift.org/ (2024).
8. *SwiftUI* https://developer.apple.com/xcode/swiftui/ (2024).
9. *Firebase* https://firebase.google.com/docs?hl=es-419 (2024).