UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Agere: una aplicación móvil destinada a mejorar la salud a través de la alimentación

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Valentín Rodríguez, Ana

Tutor/a: Letelier Torres, Patricio Orlando

CURSO ACADÉMICO: 2022/2023

A mis padres y a mi hermano, por su amor y apoyo incondicional

Agradecimientos

A los profesores que en el transcurso de mi formación me han ayudado, guiado y enseñado los conocimientos que culminan este trabajo.

A mis padres y a mi hermano, por su constante respaldo, impulsándome a luchar por cada meta que me he propuesto.

A todos los voluntarios y participantes en este proyecto, por su compromiso y dedicación desinteresada.

Resumen

Cada vez es más común que las personas padezcan algún tipo de enfermedad, alergia o dolencia que afecta su calidad de vida. Desafortunadamente, en ocasiones no se le da la importancia necesaria a la relación que existe entre la alimentación y algunas de estas enfermedades, alergias o dolencias, lo que puede llevar a una aceptación de ciertos síntomas como algo normal, cuando en realidad podrían ser aliviados a través de una mejora en la alimentación diaria.

El objetivo de este TFG es desarrollar Agere, una aplicación que conciencie a los usuarios sobre la importancia de esta relación entre alimentación y salud, y les brinde una solución personalizada para mejorar su bienestar. Agere ofrecerá dietas a medida, en función de las necesidades de cada usuario, para que puedan llevar una alimentación saludable y equilibrada que les permita combatir sus síntomas y mejorar su calidad de vida.

La aplicación será accesible desde dispositivos Android (teléfonos móviles y tablets). Para ello Agere se desarrollará en Android Studio, usando Java junto con todas las ventajas que presenta la integración con la plataforma en la nube de Firebase, donde se hospedará la base de datos, el control de registro e inicios de sesión de usuarios y el almacenamiento de archivos entre otros. Para el desarrollo de esta aplicación se va a utilizar una metodología ágil y un enfoque Lean Startup por tratarse de un proyecto de emprendimiento. Así, se prestará atención especialmente a la validación de la viabilidad del proyecto y la adaptación necesaria según los cambios que vayan surgiendo. Para la gestión del trabajo se hará uso de la aplicación Trello. Este TFG se desarrolla en el marco de Start.inf, el espacio de emprendimiento de la ETSINF.

Palabras clave: Desarrollo de apps; emprendimiento.

Resum

Cada vegada és més comú que les persones pateixin algun tipus de malaltia, al·lèrgia o dolència que afecta la seua qualitat de vida. Desafortunadament, en ocasions no se li dóna la importància necessària a la relació que existeix entre l'alimentació i algunes d'aquestes malalties, al·lèrgies o dolències, la qual cosa pot portar a acceptar certs símptomes com a normals, quan en realitat podrien ser alleujats mitjançant una millora en l'alimentació diària.

L'objectiu d'aquest TFG és desenvolupar Agere, una aplicació que consciencie els usuaris sobre la importància d'aquesta relació entre alimentació i salut, i els proporcione una solució personalitzada per a millorar el seu benestar. Agere oferirà dietes a mida, en funció de les necessitats de cada usuari, perquè puguen dur una alimentació saludable i equilibrada que els permeta combatre els seus símptomes i millorar la seua qualitat de vida.

L'aplicació serà accessible des de dispositius Android (telèfons mòbils i tauletes). Per a això, Agere es desenvoluparà en Android Studio, fent servir Java juntament amb tots els avantatges que presenta la integració amb la plataforma en el núvol de Firebase, on s'allotjarà la base de dades, el control de registre i inici de sessió d'usuaris i l'emmagatzematge de fitxers, entre altres. Per al desenvolupament d'aquesta aplicació s'utilitzarà una metodologia àgil i un enfocament Lean Startup per tractar-se d'un projecte d'emprenedoria. Així, es prestarà especial atenció a la validació de la viabilitat del projecte i a l'adaptació necessària segons els canvis que vagen sorgint. Per a la gestió del treball s'utilitzarà l'aplicació Trello. Aquest TFG es desenvolupa en el marc de Start.inf, l'espai d'emprenedoria de l'ETSINF.

Paraules clau: Desenvolupament d'apps; emprenedoria.

Abstract

It is becoming increasingly common for people to suffer from some kind of illness, allergy, or condition that affects their quality of life. Unfortunately, the relationship between nutrition and these illnesses, allergies, or conditions is often not given the necessary importance, which can lead to accepting certain symptoms as normal when they could actually be alleviated through improvements in daily nutrition.

The objective of this thesis is to develop Agere, an application that raises awareness among users about the importance of the relationship between nutrition and health, and provides them with a personalized solution to improve their well-being. Agere will offer tailored diets based on each user's needs, allowing them to maintain a healthy and balanced diet that helps combat their symptoms and enhance their quality of life.

The application will be accessible on Android devices (mobile phones and tablets). For this purpose, Agere will be developed using Android Studio, utilizing Java along with the advantages of integrating with the Firebase cloud platform, where the database, user registration and login controls, and file storage will be hosted, among other features. An agile methodology and a Lean Startup approach will be employed in the development of this application, given its entrepreneurial nature. Special attention will be given to validating the project's feasibility and making necessary adaptations as changes arise. Trello will be used for project management. This thesis is being developed within the framework of Start.inf, the entrepreneurship space at ETSINF.

Key words: App development; entrepreneurship.

Contenido

Tabla de figuras

Capítulo 1

Introducción

Nos encontramos en una época donde las personas tienen a su alcance un gran número de fuentes de información de todo tipo (política, tecnología, salud, entretenimiento, etc.). Con respecto a épocas pasadas, es un gran avance para el ser humano disponer de toda esta información en cualquier momento.

A pesar de todo lo bueno que podría conllevar todo esto, el problema latente en nuestra sociedad a día de hoy es que cualquier persona puede publicar cualquier tipo de información, sin esta ser contrastada con otras fuentes ni verificada por especialistas [1].

Si hablamos de noticias relacionadas con las novedades de una actualización de un videojuego o un reportaje sobre los mejores restaurantes de tu localidad, es posible que un error no resulte de gran importancia, ya que no conlleva ningún riesgo para el usuario final que lee la noticia. Sin embargo, esto es diferente cuando hablamos de noticias que aconsejan sobre la salud directa del usuario, cuando no es un especialista el que publica dicho artículo y no está respaldado por ningún sanitario o nutricionista.

Son numerosos los casos de usuarios que buscan una solución a sus dolores de cabeza, cómo quitar oscurecimientos en la piel o mejorar el aspecto de sus uñas mediante búsquedas en internet, y son infinitas las entradas que nos aparecen en el buscador de internet dando cada uno soluciones diferentes, algunos con más rigor científico que otros [2]. Desafortunadamente, también son numerosos los casos de personas que han empeorado su salud por aplicar, precisamente, los consejos dados por estos artículos, los cuales confunden y, en ocasiones, engañan osadamente al usuario final.

El médico de familia, Raúl Piedra Castro, relata una experiencia en la que ha presenciado cómo las indicaciones de las redes sociales pueden empeorar un problema de salud preexistente. Según su testimonio, uno de sus pacientes diabéticos decidió reducir sus dosis de insulina después de haber seguido la recomendación de un " un preparado de herbolario que le habían recomendado". Afortunadamente, el paciente compartió esta información durante la consulta médica, lo que permitió al médico solucionar el problema a tiempo.

Vicente J. Álvarez, presidente del Colegio Oficial de Farmacéuticos de Orense, comparte una anécdota peculiar relacionada con una mujer que acudió a la farmacia junto a su hija, quien había sido picada por una avispa. En lugar de buscar un tratamiento adecuado, decidieron recurrir a internet para encontrar una solución para aliviar el dolor. Según relata Álvarez, encontraron una recomendación sorprendente: "Le mearon encima y le cubrieron la picadura con tierra y así vino la pobre niña" [3].

Hoy en día, es cada vez más común que las personas padezcan algún tipo de enfermedad, alergia o dolencia que afecta su calidad de vida. De hecho, muchas personas pueden estar conviviendo con estas condiciones sin siquiera saberlo.

Desafortunadamente, en ocasiones no se les da la importancia necesaria a la relación que existe entre la alimentación y algunas de estas enfermedades, alergias o dolencias, lo que puede llevar a una aceptación de ciertos síntomas como algo normal, cuando en realidad podrían ser aliviados a través de una mejora en la alimentación diaria.

Poco a poco se está viendo un incremento de la preocupación de las personas por su salud, tanto es así que en 2021 se realizó una encuesta por la Asociación Española de Comunicación Sanitaria, donde obtuvieron que, actualmente, más de la mitad de la población española ya tiene descargada alguna aplicación de salud y/o nutrición [4].

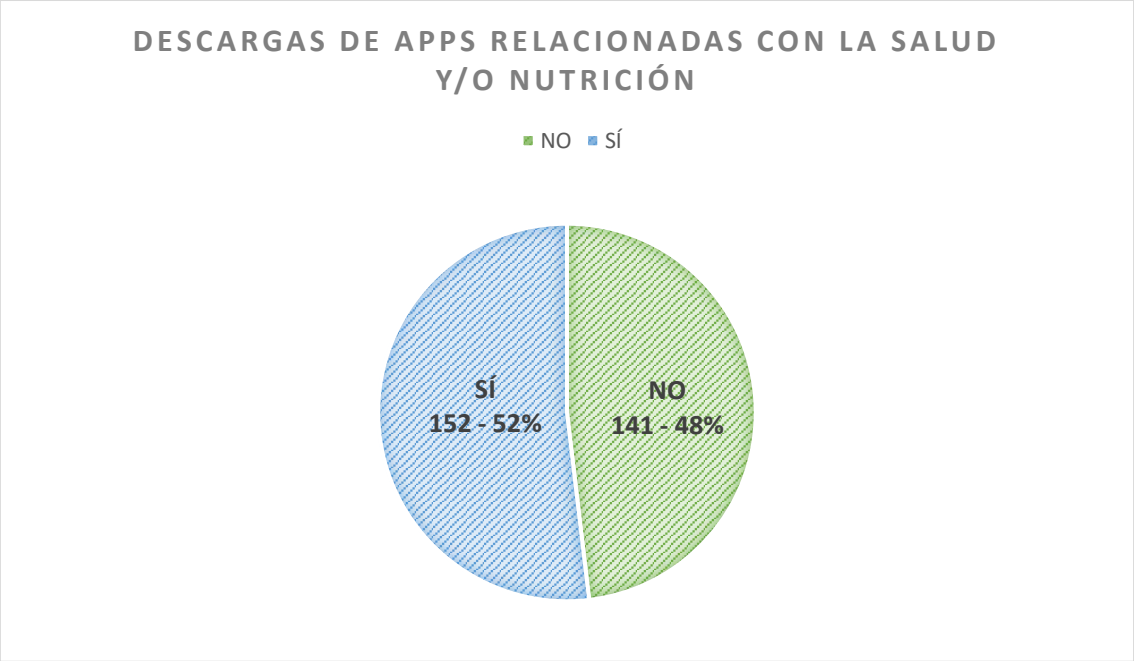


Figura 1.1 Gráfica descarga de apps relacionadas con la salud y/o nutrición

Como se observa en la Figura 1.1, el estudio realizado en 2021 concluyó que más de la mitad de la población se había descargado aplicaciones relacionadas con la salud.

Es aquí donde surge la aplicación móvil Agere, que busca concienciar a los usuarios sobre la importancia de la relación entre alimentación y salud, y brindarles una solución personalizada para mejorar su bienestar.

1.1. Objetivos

El objetivo de este proyecto es desarrollar una idea de negocio basada en una aplicación que ayude a mejorar la alimentación de los usuarios y concienciar sobre la directa relación existente entre la salud y la alimentación.

Para lograr este objetivo, hemos decidido establecer los siguientes subobjetivos:

- Que el usuario disponga de un acceso sencillo a toda la información necesaria básica sobre la salud y su alimentación.
- Ofrecer dietas personalizadas a las necesidades de los usuarios en función de las enfermedades que puedan padecer, asegurando siempre el bienestar del usuario.

- Ofrecer herramientas que permitan al usuario adaptar las dietas a sus necesidades.
- Ofrecer una gran variedad de posibilidades de recetas y platos a los que pueda optar el usuario.
- Configurar la app para una prueba (abierta o cerrada) en Google Play, es decir, dar un acceso anticipado a los usuarios para que puedan probar la aplicación y, en caso de que se necesite, reportar sugerencias o fallos que se encuentren durante el uso de esta.

1.2. Estructura de la memoria

Para una mayor comodidad, la memoria se ha dividido en 5 capítulos, los cuales constarán del siguiente contenido:

1. Capítulo 1

   Se abordará como y porqué surgió la idea de crear Agere y los objetivos a cumplir.
2. Capítulo 2

   Se realiza un estudio extenso sobre el proyecto, entre los que se destacan el estudio de clientes y de mercado, una previsión económica a 6 años así como nuestros puntos fuertes y débiles ante nuestros competidores.
3. Capítulo 3

   Se contarán en detalle todas las tecnologías utilizadas, así como pequeñas guías de como se hizo uso de las mismas y para que nos sirvieron en nuestro desarrollo.
4. Capítulo 4

   En este capítulo se detallará el desarrollo propio del proyecto, que metodología se decidió adaptar, así como detalles, desafíos y pruebas que se realizaron a lo largo de la realización de la aplicación.
5. Capítulo 5

   Como parte final se realizan unas conclusiones de todo el trabajo realizado y las tareas a proseguir en un futuro para la evolución del proyecto.

Capítulo 2

Evaluación de la idea de negocio

El origen de la aplicación se dio debido a la preocupación por la falta de actividad física y el consumo excesivo de "comida basura" en la sociedad actual. La idea se fue fortaleciendo a través de varias discusiones con personas cercanas hasta que se decidió buscar la opinión de especialistas. A partir de este momento, comenzó la segunda etapa de la aplicación.

Luego de consultar con expertos de la salud, se llegó a la conclusión que existían muchas aplicaciones que relacionasen el ejercicio, la alimentación y la salud final del usuario, pero ninguna se enfocaba adecuadamente en personas con enfermedades o condiciones específicas.

Curiosamente, los expertos coincidieron en que actualmente las fuentes de información (webs, videos, revistas, etc.) confunden y desinforman más a los pacientes en lugar de brindarles ayuda e información veraz.

Con este nuevo rumbo, se comenzó a explorar la posibilidad de hacer una aplicación donde se diesen consejos de salud a los usuarios en función de que enfermedad o condición padeciesen, además de aportarles orientación para saber que dieta es más recomendable seguir para cada usuario individualmente.

Con este cambio de dirección, es como se encaminó la ultima versión de Agere, donde buscamos crear una aplicación que brinde consejos de salud a los usuarios en función de su enfermedad o condición específica, proporcionándoles también orientación para saber cuál es la dieta recomendada para cada persona individualmente.

La evaluación de una idea de negocio consiste en analizar la viabilidad y potencial de éxito de una idea para iniciar un negocio. Para ello se debe realizar un estudio de mercado, donde se analizarán los clientes a los que se dirige la aplicación y las principales competencias. Este estudio se complementa con el análisis DAFO, la proyección de ingresos y gastos y un Lean Canvas.

2.1. Estudio de mercado

Un estudio de mercado es una herramienta importante ya que ayuda a comprender mejor el público objetivo, evaluar la demanda, analizar la competencia, identificar características clave y evaluar las posibilidades de monetización.

Vamos a comenzar llevando a cabo un breve estudio para determinar el público objetivo al que se dirigirá nuestra aplicación. A continuación, procederemos a recopilar información sobre las principales aplicaciones que podrían representar competencia en el mercado. Finalmente, utilizando todos estos datos recopilados, enumeraremos las características ofrecidas por estas aplicaciones y, teniendo en cuenta tanto a nuestros clientes potenciales como a la competencia existente, destacaremos aquellas que consideramos que Agere debería incorporar.

2.1.1. Clientes

La aplicación está enfocada a personas que padecen alguna enfermedad y se preocupan por mantener una buena alimentación. Se excluyen a los usuarios menores de 18 años ya que, en general, son los tutores legales lo que deberán establecer su dieta.

A partir de los 20-25 años, las personas comienzan a preocuparse por su salud y buscan formas de mantenerse saludables, especialmente si previamente padecen alguna enfermedad. En esta edad también es común el surgimiento de patologías como diabetes o alto colesterol, lo que aumenta la preocupación e iniciativa de seguir una alimentación adecuada.

Es comprensible que, a medida que envejecemos, la preocupación por una alimentación adecuada en relación a alguna condición médica aumente, ya que con el paso del tiempo es común que surjan nuevos problemas de salud.

Teniendo en cuenta todo esto, se buscará abarcar a una amplia gama de usuarios, desde los jóvenes entre 20 y 25 años hasta los adultos mayores de 80 años, por lo que el diseño de la aplicación debe encontrar un equilibrio entre la simplicidad y la intuitividad para aquellos menos experimentados en tecnología y, al mismo tiempo, proporcionar suficiente información para satisfacer las necesidades de todos los usuarios.

2.1.2. Competencias en el mercado

Un estudio de las competencias del mercado es una parte fundamental del estudio previo para el desarrollo de una aplicación móvil, ya que puede aportar información valiosa sobre los competidores existentes y las tendencias del mercado.

El mercado de aplicaciones dirigidas a la nutrición y salud de los usuarios es bastante amplio. El portal de estadísticas en línea alemán Statista, declaró a finales de 2020 que dicho año fue “el año de la tecnología fitness”, realizándose importantes ingresos en desarrollar y vender nuevas aplicaciones [5].

Para realizar este estudio se han escogido 5 aplicaciones destacadas que podrían competir directamente con nuestra aplicación:

Manzana Roja

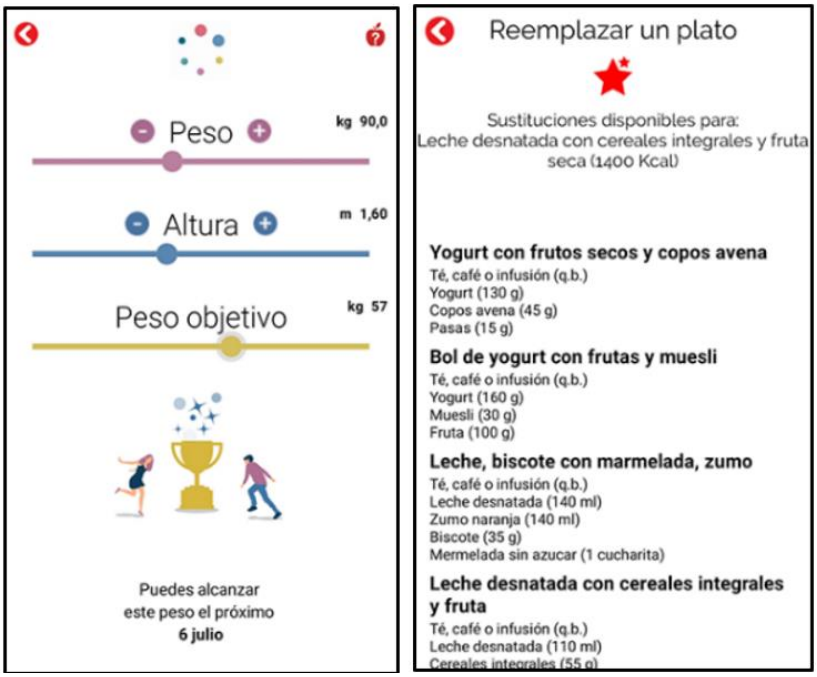


Figura 2.1 Ventanas de la aplicación Manzana Roja

Publicada el 12 de Diciembre de 2013, la aplicación Manzana Roja[1] cuenta con más de 1.000.000 de descargas en Google Play.

Se trata de una herramienta de nutrición que te ayuda a alcanzar tu objetivo de perder peso con una dieta personalizada. Diseñada por un equipo de expertos en nutrición y dietética, asegura ofrecer una alimentación equilibrada y adaptada a tus necesidades individuales. Además, cuenta con la supervisión de la Sociedad Italiana de Ciencia de los Alimentos (SISA).

Aunque cuenta con un respaldo científico relevante, la experiencia de usuario al utilizar la aplicación no es óptima. Su interfaz es muy simple, poco trabajada y sin apenas decoraciones, lo que la hace poco atractiva para los usuarios. Además, al utilizar la aplicación, la navegación no resulta del todo intuitiva y en ocasiones puede resultar complejo encontrar opciones específicas ofrecidas por la aplicación.

[1] Manzana Roja: www.manzanaroja.eu

Cookpad

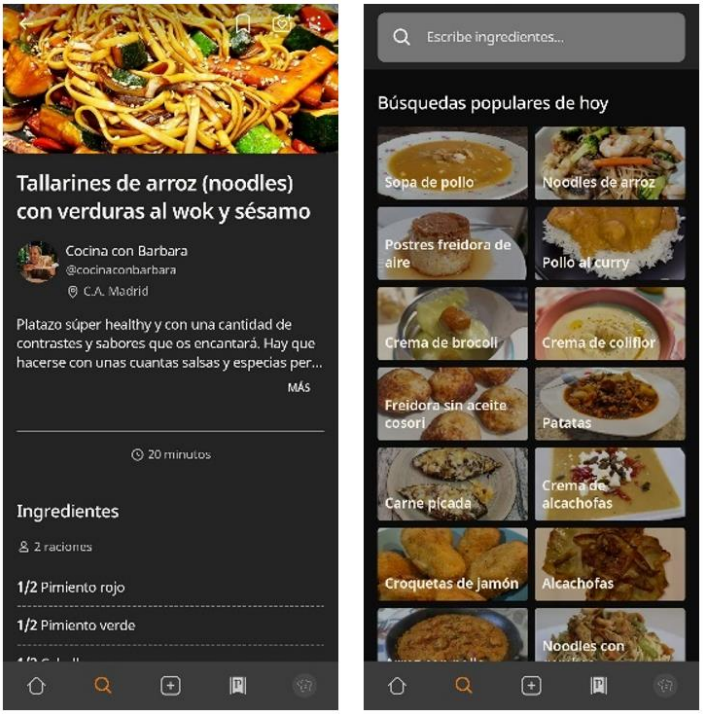


Figura 2.2 Ventanas de la aplicación Cookpad

Esta aplicación, a pesar de no estar enfocada a la salud, resulta muy completa en el ámbito de la nutrición, teniendo amplias funcionalidades enfocadas a las recetas, y consejos entre usuarios.

Publicada el 4 de septiembre de 2010, ya cuenta con más de 100.000.000 de descargas en Google Play. Cookpad[2] es una red global de personas con un mismo denominador común: la cocina casera.

Su carácter de comunidad le da su toque especial, te permite compartir las fotos de tus creaciones, compartir tus propias ideas, conectar con otros cocineros y tener a mano tu propia colección de recursos culinarios.

En esta aplicación se aprecia el trabajo aplicado en equilibrar la simplicidad de la interfaz con la presentación clara y concisa de la información relevante. Esto es importante porque una interfaz demasiado simple puede no ofrecer la información suficiente para el usuario, mientras que una interfaz sobrecargada puede resultar confusa y abrumadora.

[2] Cookpad: https://cookpad.com

Lifesum

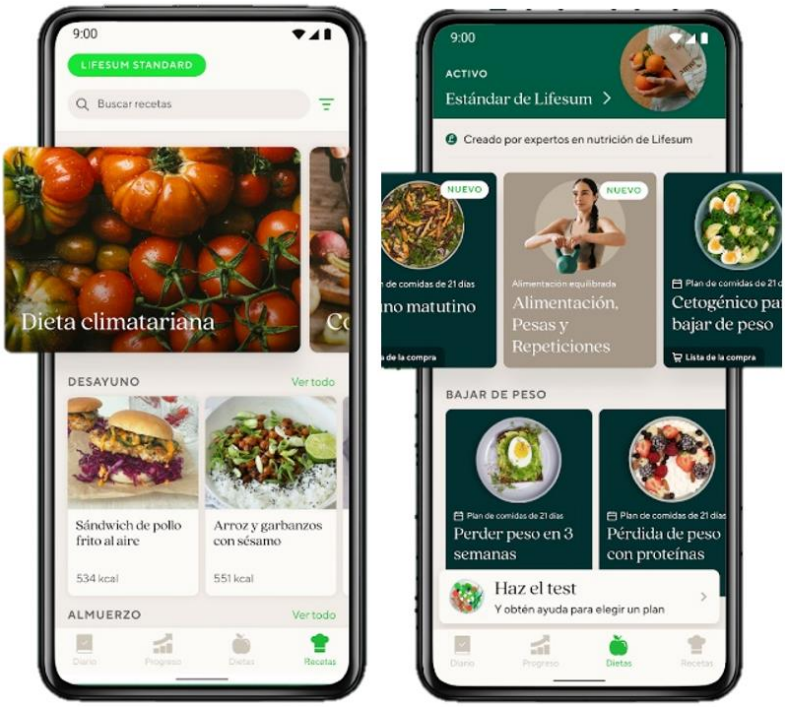


Figura 2.3 Ventanas de la aplicación Lifesum

Lifesum[3] es una de las aplicaciones de alimentación saludable más populares y completas disponibles en la actualidad. Te permite conocer las calorías y propiedades nutricionales de los alimentos, así como el peso que pierdes al hacer ejercicio. Además, ofrece consejos sobre hábitos alimentarios, como la importancia de masticar adecuadamente o beber la suficiente cantidad de agua todos los días.

Lanzada el 16 de junio de 2011, ya cuenta con más de 10.000.000 de descargas en Google Play.

Al tratar de probar el producto encontramos que las recetas, en su mayoría eran de pago, la versión gratuita únicamente contaba con un plan inicial, completamente generalizado a cualquier usuario y en ningún momento transmitió esa sensación de personalización en dietas ni ejercicios.

En el caso de esta aplicación, la limitación de funciones en la versión gratuita de la aplicación pueden ser una barrera para que los usuarios la utilicen de manera efectiva y encuentren valor en ella, ya que no existe variedad ni opciones de recetas para los usuarios.

[3] Lifesum: https://lifesum.com

Oorenji

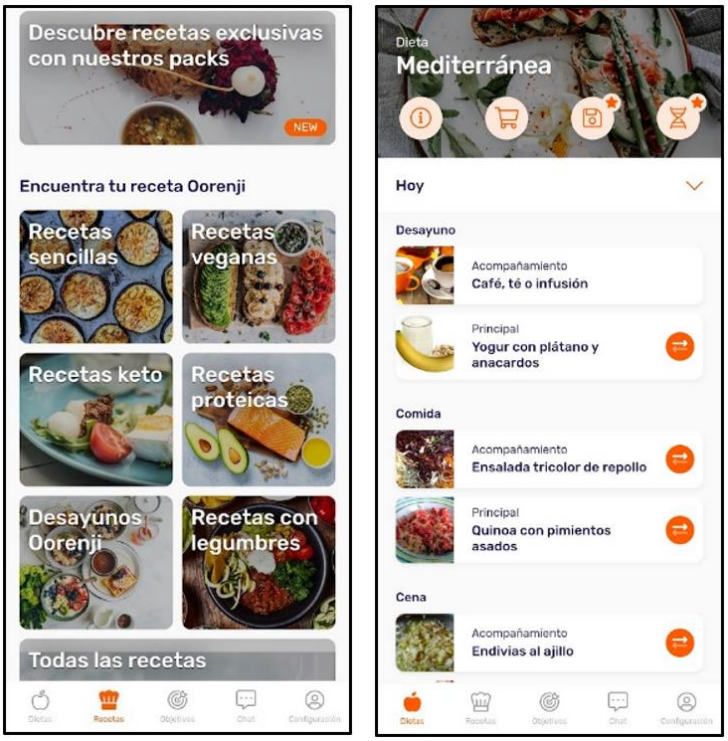


Figura 2.4 Ventanas de la aplicación Oorenji

Con más de 10.000 descargas en Google Play, encontramos la aplicación más joven de este listado, Oorenji[4], lanzada el 20 de diciembre de 2020.

Esta aplicación permite a los usuarios personalizar sus perfiles y brinda servicios y consejos especializados para mejorar en la alimentación diaria. Entre los servicios ofrecidos se incluyen dietas personalizadas, planes nutricionales adaptados a las necesidades individuales y recetas profesionales diarias.

La aplicación en cuestión es bastante completa y cuenta con una interfaz amigable, aunque hay margen de mejora, especialmente en el apartado de la elección de la dieta a seguir. Las opciones disponibles pueden resultar confusas para los usuarios menos experimentados, lo que puede dificultar su comprensión.

[4] Oorenji: https://oorenji.com

Nootric

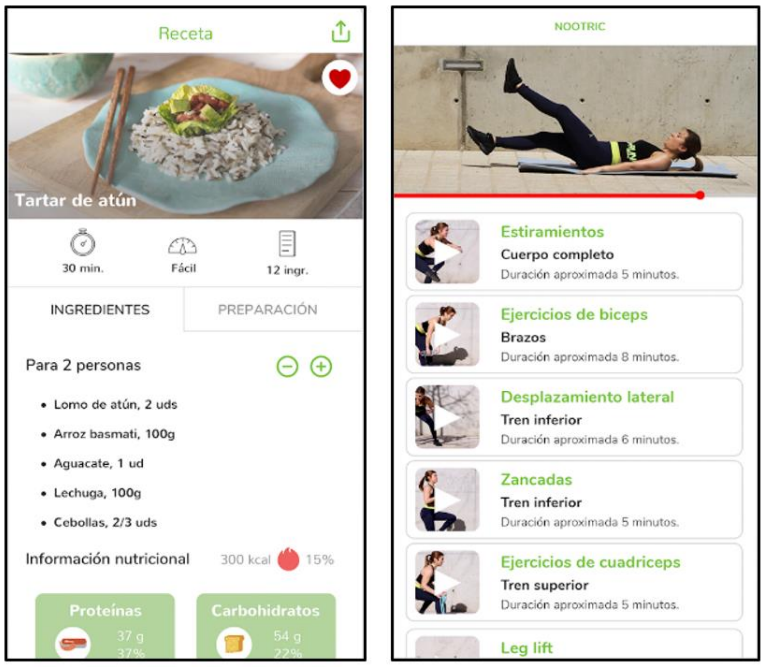


Figura 2.5 Ventanas de la aplicación Nootric

La aplicación Nootric[5] ofrece planes nutricionales con recetas gratuitas, consejos para bajar de peso, más de 100 recetas, vídeos de ejercicios adaptados a tu progreso, guías para una vida saludable y una lista de compra semanal asociada a cada plan nutricional, todo esto de manera gratuita.

Lanzada el 7 de junio de 2018, cuenta ya con más de 1.000.000 de descargas en Google Play.

La aplicación cuenta con una interfaz bien diseñada, una amplia variedad de opciones y guías útiles. En general, se percibe que es una aplicación muy bien trabajada y que puede servir como modelo a seguir para crear una aplicación de calidad.

[5] Nootric: https://www.nootric.com

2.1.3. Comparación de las características ofrecidas

Si nos enfocamos no solo en la interfaz, sino también en las características que cada aplicación ofrece individualmente, podemos identificar las funcionalidades mostradas en la Tabla 1.

Tabla 1 Tabla comparativa de productos competidores

	ManzanaRoja	cookpad	Lifesum	Oorenji	nootric	Agere
Cuestionario inicial	✓	✓	✓	✓	✓	✓
Consejos básicos de salud	✓	✗	✓	✗	✓	✓
Registro requerido	✓	✗	✓	✓	✓	✓
Variedad de recetas	✓	✓	✓	✓	✓	✓
Planes de nutrición preestablecidos	✓	✗	✗	✗	✓	✓
Menú inicial genérico	✓	✗	✗	✓	✓	✓
Tienen en cuenta enfermedades para generar menús	✓	✗	✗	✓	✓	✓
Cuestionario de preferencias	✓	✗	✗	✓	✓	✓
Objetivos diarios	✗	✓	✓	✓	✓	✓
Registro de peso	✓	✗	✓	✓	✓	✓
Sección de ejercicios	✗	✗	✗	✗	✓	✗
Contactar con nutricionistas	✗	✗	✗	✓	✓	✗
Consejos médicos de enfermedad/es que se padece	✗	✗	✗	✗	✗	✓
Libros de recetas	✗	✓	✗	✗	✗	✗
Permite crear recetas por los usuarios	✗	✓	✗	✗	✗	✗
Versión gratuita	✓	✓	✓	✓	✓	✓
Versión de pago	2,99€/mes	1,99€/mes	4,99€/mes	3,99€/mes - 29,99€/mes	39,99€/mes	3,99€/mes

Como podemos observar, hay características completamente necesarias para poder posicionarse en el mercado y competir en igualdad con el resto de aplicaciones, como puede ser “Variedad de recetas” o “Cuestionario inicial”, las cuales, si no las implementásemos, nos posicionaríamos con cierta desventaja respecto al resto de aplicaciones.

En algunos casos, se han dejado de lado ciertas características que no se consideraron necesarias, ya que prácticamente ninguna aplicación las ofrece. En caso de ser requeridas, podrían ser incluidas en futuras versiones de la aplicación. Un ejemplo de esto es la "Sección de ejercicios", la cual se ha decidido no implementar, ya que se aleja del objetivo principal de la aplicación, que es brindar consejos relacionados con la alimentación y tratar de mejorar los hábitos alimenticios.

Por último, podemos identificar características que nos destacan del resto, como por ejemplo "Consejos médicos específicos para las enfermedades que se padecen". Esta característica no se encuentra presente en ninguna aplicación actual del mercado, lo cual nos brinda la oportunidad de destacarnos y diferenciarnos de la competencia.

2.2. Análisis DAFO

Un análisis DAFO puede aportar información valiosa en el estudio previo para el desarrollo de una aplicación móvil. Los aspectos a considerar son:

- **Fortalezas**: identificar las fortalezas del equipo de desarrollo puede ayudar a aprovechar las habilidades y recursos disponibles para crear una aplicación móvil de alta calidad y que ofrezca una experiencia de usuario óptima.
- **Debilidades**: identificar las debilidades de la empresa o equipo de desarrollo puede ayudar a identificar áreas que necesitan mejorar, como la capacitación en habilidades específicas o la falta de experiencia en ciertos aspectos del desarrollo de aplicaciones móviles.
- **Oportunidades**: identificar oportunidades en el mercado móvil puede ayudar a crear una aplicación móvil que satisfaga las necesidades de los usuarios y que tenga un potencial de mercado atractivo.
- **Amenazas**: identificar las amenazas en el mercado móvil puede ayudar a prever posibles problemas que puedan surgir durante el desarrollo de la aplicación móvil, como la competencia fuerte o los cambios en la tecnología móvil.

Es decir, el análisis DAFO puede ayudar a comprender el entorno en el que se va a desarrollar la aplicación móvil y a identificar aspectos clave que puedan afectar su éxito en el mercado. Esto puede ayudar a desarrollar una estrategia efectiva para crear una aplicación móvil exitosa y con una ventaja competitiva.

Así pues, a continuación se presenta el análisis DAFO realizado para nuestro proyecto:

Figura 2.6 Análisis DAFO

Como podemos observar en la Figura 2.6, nuestras mayores debilidades tienen que ver con la falta de experiencia y conocimientos, pero con la ayuda y apoyo de los expertos que respaldarán la aplicación, se espera poder sortear dichas debilidades.

No debemos ignorar las posibles amenazas, especialmente considerando el entorno en el que planeamos lanzar nuestra aplicación, dado que se trata de un mercado bastante saturado. Sin embargo, también debemos aprovechar las oportunidades, como el creciente interés de la población en general por este tipo de aplicaciones. Si logramos captar la atención con características novedosas y únicas, podremos posicionarnos como una de las mejores aplicaciones en el mercado.

2.3. Modelo de negocio

El modelo de negocio define cómo la empresa crea valor al identificar las necesidades de los clientes y ofrecer soluciones adecuadas a cambio de un beneficio económico. También establece cómo se organiza la empresa, qué recursos utiliza, cómo se relaciona con los proveedores y socios, y cómo se entrega y distribuye su oferta.

A día de hoy existen múltiples modelos de negocio ampliamente extendidos en todo el mundo, entre los cuales podemos encontrar líder de mercado, suscripción, *freemium*… Tras un estudio de todas las posibilidades que se presentaban, finalmente la opción escogida fue el modelo *freemium*.

Este modelo se basa en ofrecer una versión gratuita de un producto, seguida de una oferta de mejoras o funcionalidades adicionales a cambio de una tarifa. Algunos ejemplos de este modelo incluyen proveedores de hosting que ofrecen una versión básica gratuita con la opción de actualizarse a planes de pago para obtener beneficios adicionales, aplicaciones de música que ofrecen una suscripción premium sin anuncios y características exclusivas, o tiendas en línea que ofrecen membresías con ventajas como envío prioritario o cancelación de gastos de envío.

Este enfoque permite a los usuarios experimentar y utilizar el producto de forma gratuita, lo que a menudo ayuda a atraer una base de usuarios amplia. Luego, se ofrece una propuesta de valor adicional a aquellos usuarios dispuestos a pagar por características mejoradas o beneficios adicionales.

Este modelo de negocio es beneficioso tanto para los usuarios como para la empresa. Los usuarios pueden disfrutar de una versión básica gratuita y, si lo desean, pueden optar por una experiencia mejorada mediante el pago de una tarifa, en nuestro caso, establecida a 3,99 € al mes. Por otro lado, la empresa puede generar ingresos adicionales al ofrecer mejoras premium, lo que ayuda a financiar la operación y el desarrollo continuo del producto.

Es importante destacar que la oferta de valor adicional debe ser lo suficientemente atractiva y relevante para que los usuarios estén dispuestos a pagar por ella. Además, es esencial establecer un equilibrio adecuado entre la versión gratuita y las mejoras de pago para garantizar la satisfacción de los usuarios y la rentabilidad del modelo de negocio.

2.4. Proyección de ingresos y gastos

La proyección de ingresos y gastos es una herramienta valiosa para evaluar la viabilidad financiera del proyecto y planificar estrategias para maximizar los ingresos y minimizar los gastos. Esto puede ayudar a garantizar que el proyecto sea rentable y tenga éxito en el mercado móvil [10].

Se ha realizado una estimación preliminar de ingresos y gastos durante los primeros 6 años de vida del producto. Para ello, se ha considerado únicamente a los usuarios que se estima adquirirán la licencia premium, la cual se ha fijado en 3,99€ al mes o 47,88€ al año. Con el fin de prever la cantidad de estos usuarios, se ha tenido en cuenta el número actual de usuarios de productos competidores, como Oorenji, que cuenta con más de 10.000 usuarios en 3 años desde su lanzamiento, y Nootric, que ha registrado más de 1.000.000 de descargas en casi 5 años.

Se asume que al cabo de 4 años, se debería alcanzar al menos el 1% del número de usuarios de Nootric, lo cual representaría alrededor de 10.000-15.000 usuarios. Además, se busca alcanzar o, si es posible, superar la cantidad de usuarios de Oorenji.

Los gastos más significativos a tener en cuenta durante el primer año se relacionan con los salarios, como los de los directores o los expertos médicos. Al contar el primer año con un equipo de unas seis personas, y teniendo en cuenta que se trata de un equipo multidisciplinar, se buscará el alquiler de una pequeña oficina donde poder reunir al equipo, lo cual permitirá una mayor colaboración y eficiencia en el trabajo en equipo. Los costos de los servicios de Google Firebase son variables y se pagarán en función del uso, pero gracias a su Calculadora de plan *Blaze* [6] se han podido realizar cálculos aproximados de los que pueden llegar a costar, suponiendo datos a la alza, de forma que da margen por si se le da un uso más exhaustivo del esperado por los usuarios.

Durante el segundo año, tras un cierre con 106.000€ en perdidas, se mantendrá el mismo equipo, realizando una mayor inversión en marketing. Con esto se buscará ampliar el alcance de la aplicación y permitir crecer nuestro número de usuarios en la aplicación.

En el tercer año, habiéndose visto incrementado el número de usuarios, los mantenimientos y exigencias de estos serán más críticos, por lo que se ampliará el equipo de expertos médicos, técnicos y programadores, siendo este último formado finalmente por un programador senior y dos programadores junior. Además se asignará una parte del presupuesto para mejorar el equipo de marketing. El objetivo es obtener mejores resultados en el año siguiente a través de estrategias de marketing más efectivas y optimizadas.

Al comienzo de este tercer año también cabe destacar que se alcanzará el punto de equilibro, es decir, al comienzo de este año se habrán cubierto los costos tanto fijos como variables y a partir de este momento se empezarán a contabilizar las ganancias de nuestra aplicación.

En el cuarto año, tras un cierre del tercer año en tan solo 1.800€ en perdidas y con un equipo de programadores bien asentado, se ampliará el equipo de técnicos de soporte. Además, se realizará una nueva y mayor inversión en estrategias de marketing enfocadas en la expansión y promoción del producto con el propósito de captar nuevos usuarios y consolidar la presencia de Agere en el mercado.

Al finalizar este cuarto año ya se habrá recuperado la inversión y comenzado a tener ganancias. Tras estos primeros 4 años, podemos indicar que la inversión inicial necesaria para llevar a cabo Agere es de 123.000€.

En la Figura 2.7 se ha representado de una forma más gráfica lo expuesto anteriormente. Como se puede observar, durante los tres primeros años no se obtienen suficientes beneficios para amortizar los gastos, por lo que se cierra con pérdidas hasta el cuarto año, donde las ganancias son suficientes como para recuperar todas las perdidas de los años anteriores y comenzar a tener beneficios.

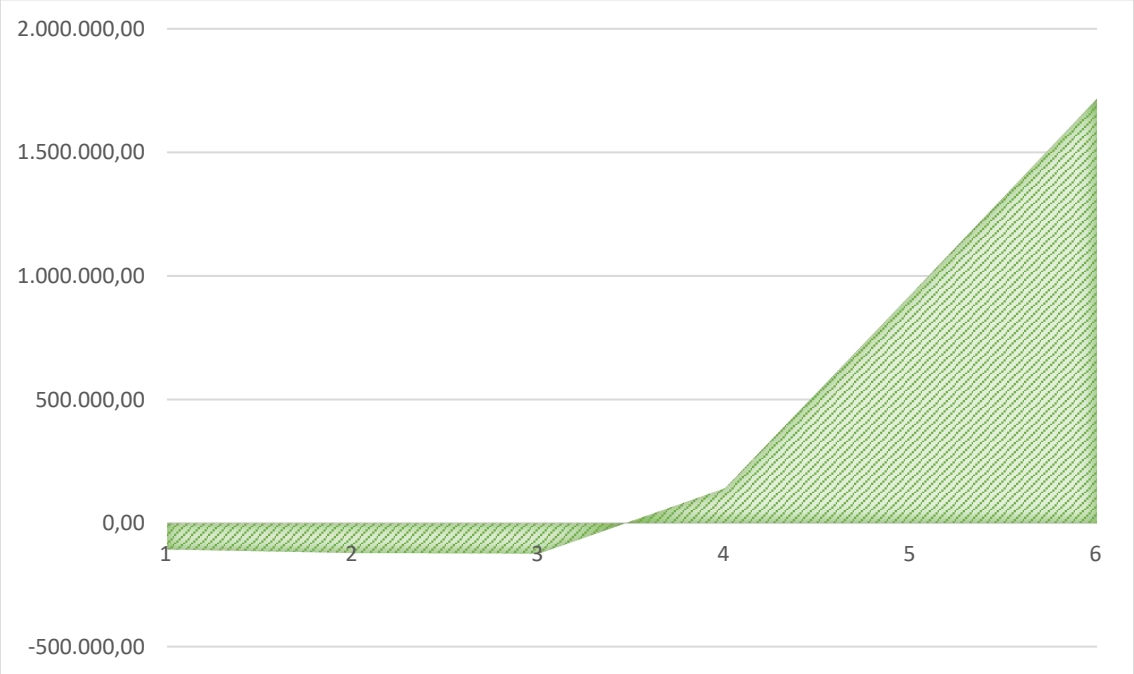


Figura 2.7 Resultados anuales

A continuación, en la Tabla 2 se detalla económicamente todos los gastos e ingresos expuestos previamente durante los primeros 6 años.

Tabla 2 Ingresos y gastos estimados en 4 años

	Años					
Tipos de licencias acumuladas	**1**	**2**	**3**	**4**	**5**	**6**
Versión premium	300	1.500	3.000	9.000	17.000	28.000
Versión premium renovada	210	1.050	2.100	6.300	10.000	16.600
Total de licencias nuevas vendidas y renovadas	510	2.550	5.100	15.300	27.000	44.000
Ingresos anuales						
(Ventas premium + renovaciones) * 3,99€ * 12	2.4418,8 €	122.094 €	244.188 €	588.924 €	1.292.760 €	2.106.720 €
Total ingresos	24.418,8 €	122.094 €	244.188 €	588.924 €	1.292.760 €	2.106.720 €
Gastos anuales						
Firebase (*Blaze*[6])	400 €	1.000 €	2.000 €	3.800 €	5.000 €	6.000 €
Marketing	8.000 €	12.000 €	20.000 €	40.000 €	50.000 €	70.000 €
Internet, electricidad, agua, teléfono, etc.	300 €	2.000 €	2.000€	4.000€	4.000 €	6.000 €
CTO - Director técnico	25.000 €	25.000 €	25.000 €	50.000€	50.000 €	50.000 €
CEO - Director ejecutivo	25.000 €	25.000 €	50.000 €	50.000 €	50.000 €	50.000 €
Técnicos de soporte	18.000 €	18.000 €	36.000 €	54.000 €	54.000 €	54.000 €
Programador	21.000 €	21.000 €	63.000 €	63.000 €	80.000 €	80.000 €
Expertos médicos	15.000 €	15.000 €	30.000 €	30.000 €	30.000 €	30.000 €
Alquiler oficina	18.000 €	18.000 €	18.000 €	30.000 €	45.000 €	45.000 €
Total gastos	130.700 €	137.000 €	246.000 €	324.800 €	368.000 €	391.000 €
Resultado anual	-106.281,2 €	-14.906 €	-1.812 €	264.124 €	924.760 €	1.715.720 €
Resultado anual acumulado	-106.281,2 €	-121.187,2 €	-122.999,2 €	141.124 €	1.065.884 €	3.431.440 €

2.5. Lean Canvas

Un Lean Canvas es una herramienta de planificación estratégica que se utiliza para definir el modelo de negocio de una empresa o proyecto. Puede aportar información valiosa en el estudio previo para el desarrollo de una aplicación móvil, permitiéndonos prever si el proyecto es rentable y tendrá éxito en el mercado .

El resultado de este proceso se muestra en la Figura 2.8. Cabe destacar el apartado “Ventaja competitiva” , ya que es en este apartado donde encontramos los aspectos que nos diferencian del resto de aplicaciones y nos pueden hacer destacar en el mercado tan competitivo que nos encontramos. El gran reto se presenta en que esta diferencia podría ser fácilmente replicable por nuestros competidores, por lo que tendremos que lograr posicionarnos en los primeros años destacando dicha característica para lograr ser una aplicación referente.

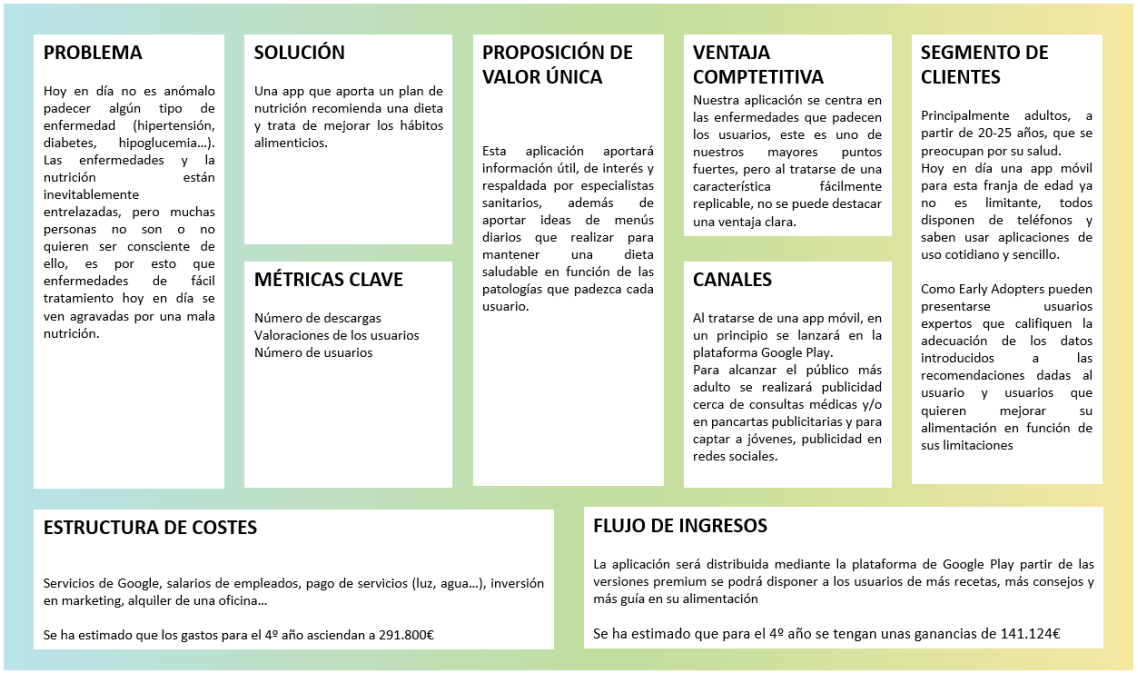


Figura 2.8 Lean Canvas

2.6. Conclusiones

El entorno de aplicaciones móviles enfocadas en la alimentación y la salud de los usuarios está altamente extendido y cada vez es más competitivo. A pesar de esto, es posible destacar y competir con éxito en este mercado, ya que cada aplicación se enfoca en un aspecto diferenciado. En este sentido, Agere, una aplicación móvil especializada en proporcionar una dieta adecuada a usuarios que padecen algún tipo de enfermedad, tiene la oportunidad de destacar en este mercado.

Al enfocarse en una audiencia específica, Agere puede ofrecer soluciones personalizadas y adaptadas a las necesidades de sus usuarios, lo que la diferencia de otras aplicaciones más generalizadas. Además, al enfocarse en la salud y el bienestar

[6] *Blaze*: Plan de pago por consumo utilizado actualmente por *Google Firebase*

de sus usuarios, Agere puede posicionarse como una aplicación de confianza, capaz de ofrecer recomendaciones y asesoramiento de alta calidad a sus usuarios.

Por tanto, a pesar de la alta competencia en el mercado de aplicaciones móviles de alimentación y salud, Agere puede destacar y competir con éxito al enfocarse en las necesidades específicas de sus usuarios y ofrecer soluciones personalizadas, adaptadas y de alta calidad.

Capítulo 3

Tecnologías utilizadas

3.1. Android Studio

Android Studio[7] es un IDE[8] completo y potente que brinda todas las herramientas necesarias para desarrollar aplicaciones de manera eficiente y productiva.

Se trata de la herramienta oficial proporcionada por Google para crear aplicaciones Android y está diseñada para ayudar a los desarrolladores en todas las etapas del proceso de desarrollo, desde el diseño de la interfaz de usuario hasta la depuración y la compilación de la aplicación.

Algunas de las principales características de Android Studio son su editor de código, el diseñador de interfaz y la depuración de pruebas así como la generación final de la APK de la aplicación.

3.1.1. Editor de código

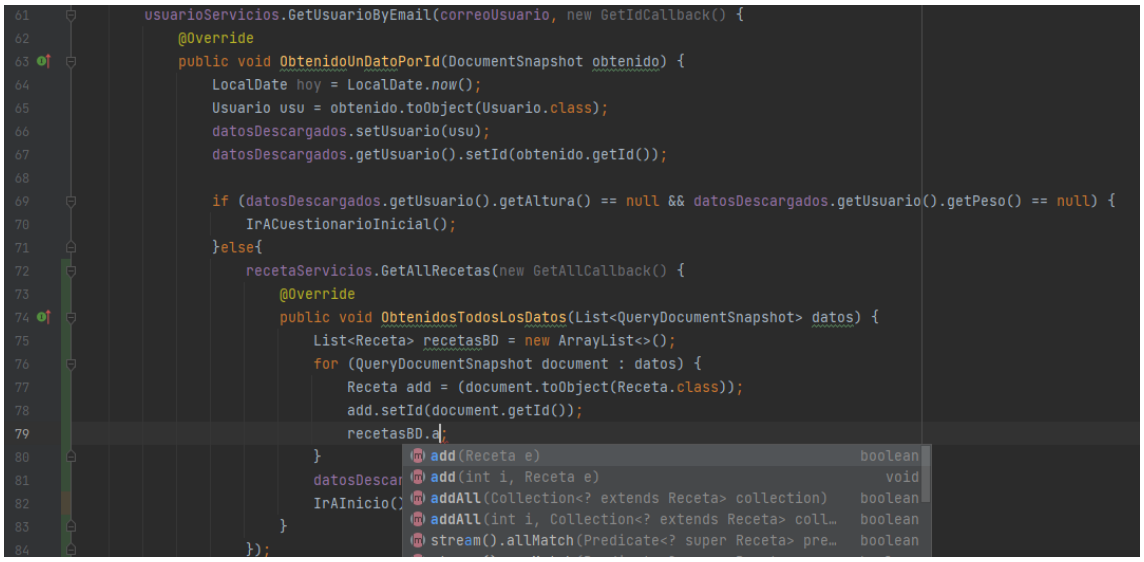


Figura 3.1 Editor de código Android Studio

Como se observa en la Figura 3.1, Android Studio proporciona un editor de código inteligente con resaltado de sintaxis, autocompletado y sugerencias para facilitar la escritura de código.

Esto permite agilizar en gran medida la programación, ya que no hace falta terminar la escritura de variables o métodos y, en caso de no estar seguros de que realiza un método no hace falta acudir a un buscador de internet o similar, ya que también incluye

[7] Android Studio: https://developer.android.com/studio/profile/android-profiler
[8] IDE: De sus siglas en inglés *Integrated Developement Enviroment*, se trata de un entorno de desarrollo integrado utilizado para escribir, depurar y ejecutar código de programación.

documentación referente a librerías integradas, como se puede observar en la Figura 3.2

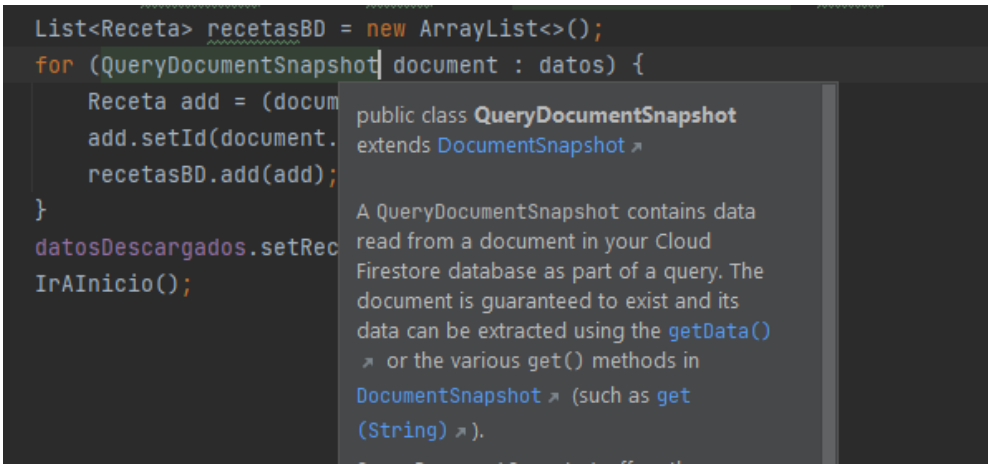


Figura 3.2 Documentación de la clase QueryDocumentSnapshot

3.1.2. Diseñador de interfaz de usuario

Figura 3.3 Diseñador de interfaz de usuario Android Studio

Android Studio permite diseñar la interfaz de usuario de la aplicación mediante una interfaz visual, de funcionamiento simple como “arrastrar y soltar elementos” y ajustando propiedades como se observa en la Figura 3.3, aunque también da la posibilidad de realizar diseños más complejos accediendo a su código asociado, programado en lenguaje xml.

3.1.3. Depuración y pruebas

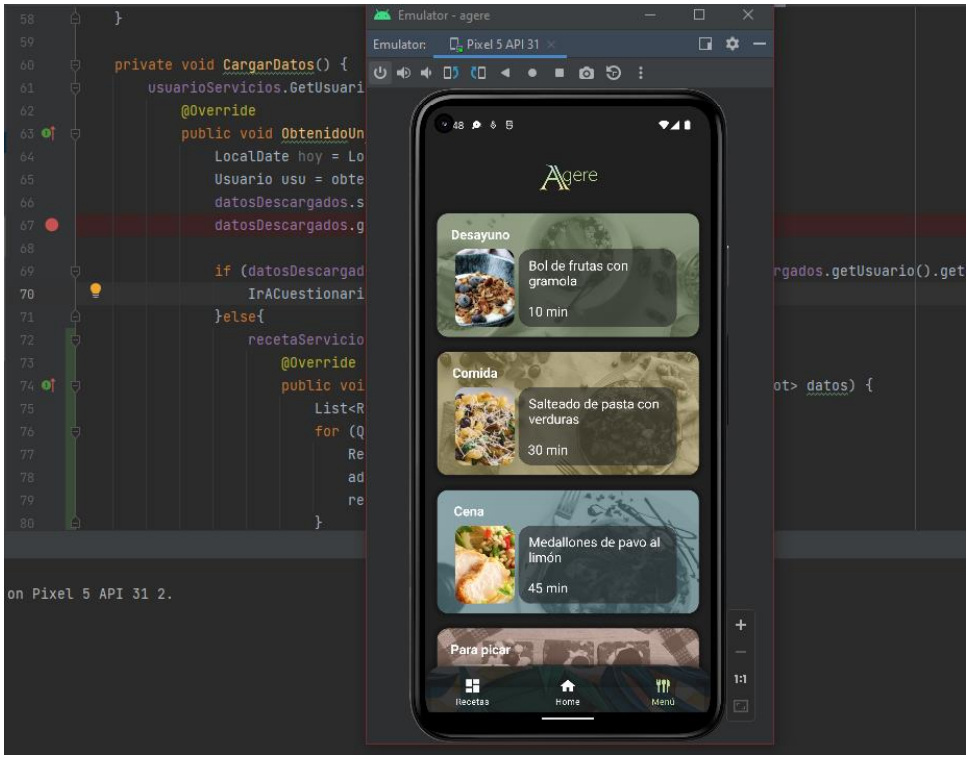


Figura 3.4 Puntos de interrupción y dispositivo virtual Android Studio

Como se observa en la figura 3.4, proporciona herramientas para depurar la aplicación, como puntos de interrupción, inspección de variables y seguimiento de la ejecución del código. También permite probar la aplicación en diferentes dispositivos virtuales o dispositivos físicos.

3.1.4. Generación de APK o Bundle

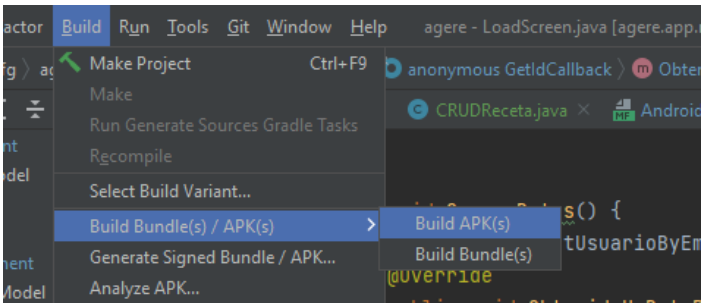


Figura 3.5 Generador APK Android Studio

Finalmente, como se observa en la figura 3.5, Android Studio permite compilar la aplicación y generar el archivo APK (*Android Package*) o *Bundle* (*Android App Bundle*) para su distribución e instalación en dispositivos Android.

Esto nos resultará muy útil cuando queramos publicar nuestra aplicación en tiendas como Google Play.

3.2. Firebase

Firebase[9] es una plataforma de desarrollo móvil y web que ofrece una amplia gama de servicios y herramientas para ayudar a los desarrolladores a crear, mejorar y hacer crecer aplicaciones. Fue adquirida por Google en 2014 y desde entonces se ha convertido en una opción popular para desarrolladores de aplicaciones tanto para Android como para iOS, así como para aplicaciones web.

Firebase proporciona una variedad de servicios que cubren áreas clave del desarrollo de aplicaciones, de los cuales se ha hecho uso del almacenamiento en la nube y la autentificación de usuarios.

3.2.1. Firebase Authentication

Firebase Auth proporciona un sistema de autenticación fácil de usar y seguro que permite a los desarrolladores gestionar la autenticación de usuarios con diferentes métodos de inicio de sesión, como correo electrónico y contraseña, Google, Facebook y más.

En nuestro caso, como se observa en la Figura 3.6, se hizo uso del correo y contraseña para realizar los inicios de sesión en nuestra aplicación.

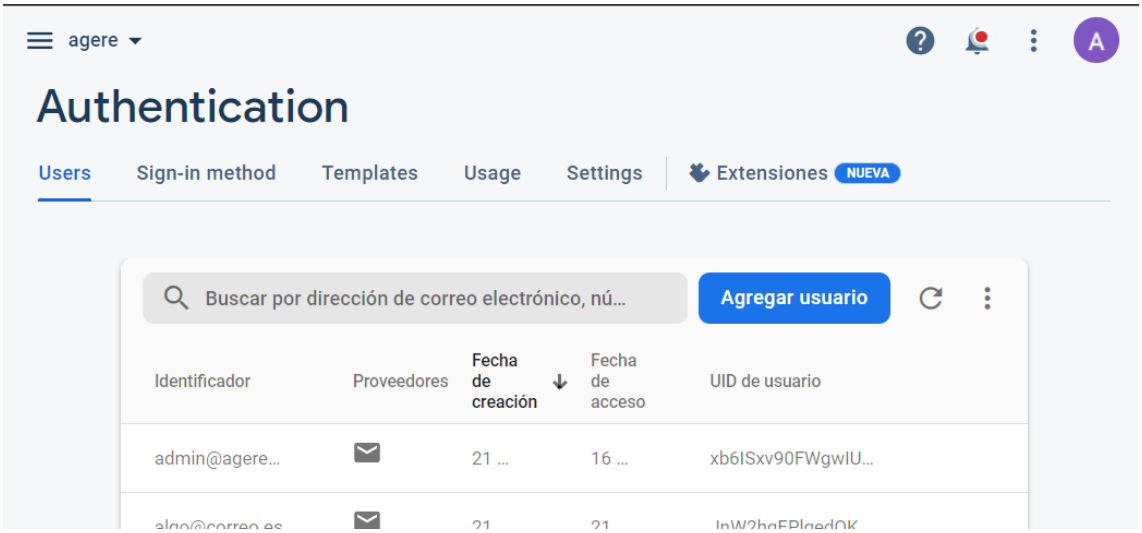


Figura 3.6 Firebase Auth

Este servicio de Firebase, a pesar de su aparente simplicidad de uso, es un servicio extremadamente potente que se convierte en el cimiento sobre el cual se construyen y dan soporte a los demás servicios.

[9] Firebase: https://firebase.google.com

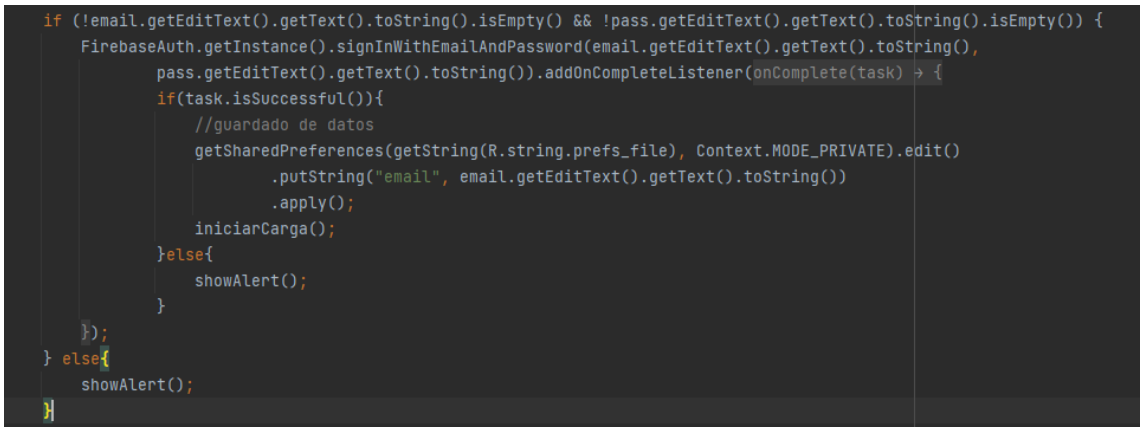


Figura 3.7 Ejemplo de uso de Firebase Auth en Agere

En la Figura 3.7, podemos ver lo simple que se realiza el inicio de sesión, mediante una única llamada a la clase FirebaseAuth junto a su método signInWithEmailAndPassword(), el cual comprueba las credenciales de inicio de sesión y gestiona la seguridad de la aplicación, impidiendo su inicio de sesión en caso de credenciales erróneas.

En caso de que las credenciales sean correctas, el inicio de sesión será efectivo en el dispositivo y, como se explicará en el siguiente apartado [8.2.2], se dará acceso a la base de datos al usuario.

3.2.2. Firestore Database

Firebase ofrece un sistema de almacenamiento en la nube escalable y seguro, que permite almacenar y recuperar datos en tiempo real.

A diferencia de una base de datos relacional tradicional, Firestore utiliza una estructura de documentos y colecciones en lugar de tablas. Los datos se organizan en documentos, que son objetos JSON, y los documentos se agrupan en colecciones, como se puede observar en la Figura 3.8. Cada documento en Firestore tiene un identificador único y puede contener múltiples campos, como cadenas de texto, números, booleanos, listas y objetos anidados.

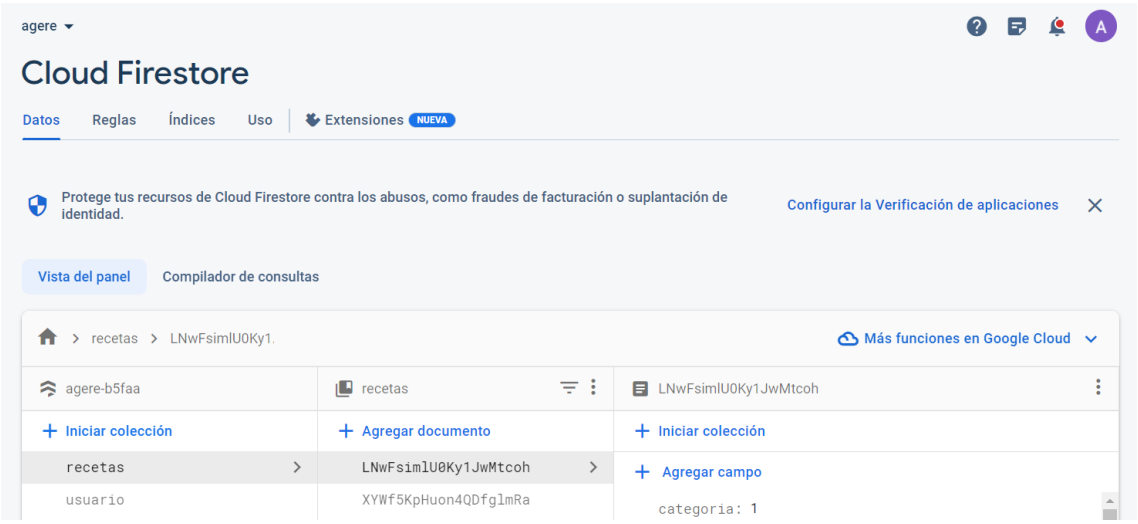


Figura 3.8 Cloud Firestore

Cabe destacar que los cambios realizados en los documentos se sincronizan automáticamente con todos los clientes conectados, lo que permite a las aplicaciones en tiempo real recibir actualizaciones instantáneas y mantener los datos consistentes en todos los dispositivos. Además, Firestore ofrece una latencia baja y un rendimiento rápido para lecturas y escrituras de datos.

Otra de las razones por las que se decidió almacenar los datos en esta base de datos de Firebase es por la seguridad y control de acceso que proporciona desde un inicio. Firestore ofrece reglas de seguridad personalizables que permiten a los desarrolladores definir quién tiene acceso y qué operaciones se pueden realizar en los datos. Esto garantiza que los datos estén protegidos y solo se acceda a ellos de acuerdo con las reglas establecidas.

Figura 3.9 Reglas de Cloud Firestore

Como Firestore se integra de manera nativa con otros servicios de Firebase, como Firebase Auth, en nuestro caso decidimos restringir las peticiones, dando únicamente acceso a los usuarios que se encuentren registrados en el momento de la llamada a base de datos, como se puede observar en la Figura 3.9.

3.3. GitHub

Se ha hecho uso de GitHub[10] como sistema de control de versiones distribuido para realizar el seguimiento y control de los cambios que se iban realizando a lo largo del proceso de desarrollo de la aplicación.

GitHub es una plataforma basada en la web que agrega funcionalidades adicionales al control de versiones de Git, como el alojamiento remoto de los repositorios o la integración con herramientas y servicios de desarrollo, como IDEs[11], en nuestro caso, Android Studio, facilitando ampliamente su uso.

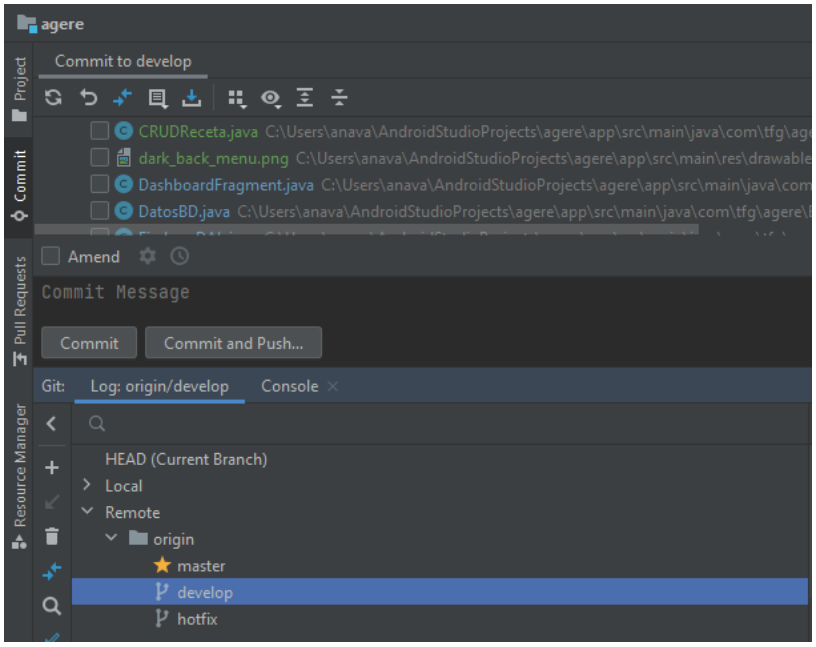


Figura 3.10 Commits a GitHub y gestión de ramas desde Android Studio

Tanto la realización de *commits*, resolución de conflictos al fusionar ramas o la gestión de las propias ramas se puede realizar de una forma sencilla mediante la interfaz de Android Studio, cuya integración con GitHub es excelente.

3.3.1. Git

Git es un sistema de control de versiones distribuido utilizado en programación y desarrollo de software. Es un sistema que permite a los desarrolladores realizar un seguimiento de los cambios realizados a lo largo del tiempo.

Git también ofrece diversas funcionalidades, como la posibilidad de crear y fusionar ramas, clonar y compartir repositorios, y el uso de etiquetas para marcar versiones específicas del proyecto.

[10] GitHub: https://github.com
[11] IDE: por sus siglas en ingles, *Integrated Development Environment*, es un entorno de desarrollo integrado que proporciona a los programadores todas las herramientas necesarias para desarrollar aplicaciones software en un solo lugar.

Ramas

En el contexto de Git, una rama (también conocida como "*branch*" en inglés) es una línea de desarrollo independiente que permite trabajar en diferentes versiones de un proyecto al mismo tiempo.

En Git, cada repositorio tiene al menos una rama predeterminada llamada "*master*" (o "*main*" en algunos casos), que representa la versión principal y estable del proyecto. Una rama se crea a partir de esta rama principal y puede contener cambios, adiciones o eliminaciones de archivos sin afectar directamente a la rama principal.

Las ramas permiten a los desarrolladores trabajar de manera aislada en nuevas características, solucionar problemas o experimentar sin afectar el código base. Se pueden crear ramas adicionales según sea necesario y, una vez completadas las modificaciones en una rama, se pueden fusionar con la rama principal para incorporar los cambios realizados.

En el caso específico de Agere, se hicieron uso de las siguientes ramas mostradas en la Figura 3.11

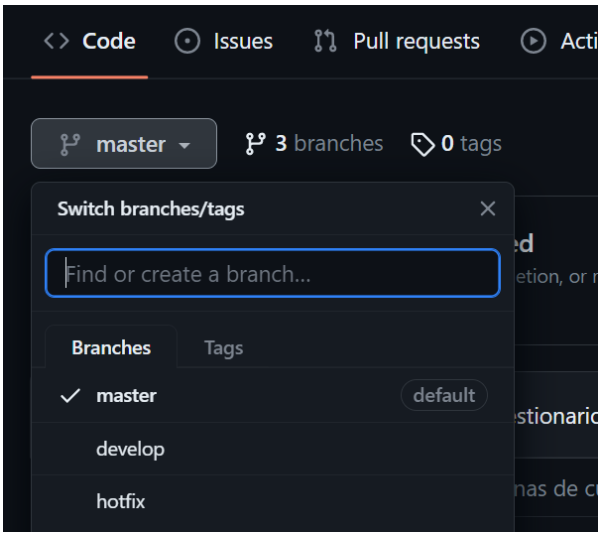


Figura 3.11 Ramas GitHub

Master

La rama *master* contiene el código final, funcional y sin fallos. En esta rama se almacena todo el código que previamente ya ha sido revisado y testeado, por lo que, si no surgen nuevos errores, esta rama contiene la última versión funcional del producto

Develop

En esta rama se lleva a cabo el desarrollo de la aplicación, se programa y se testea el código. Cuando un conjunto de UTs se dan por finalizadas y testeadas, todo el avance será mergeado[12] a la rama *main* para actualizar la última versión funcional de nuestro producto.

[12] Mergeado: acción de combinar una rama de Git con otra diferente del mismo proyecto.

Hotfix

En esta rama se trabaja eventualmente cuando surge algún *bug*[13] crítico. La finalidad de esta rama es aislar el código que contiene el fallo hasta ser solucionado, y en caso de ser necesario poder seguir trabajando en otras secciones de código en otra rama sin que el *bug* entorpezca el avance.

3.4. Figma

Figma[14] es una herramienta de diseño y prototipado. Proporciona un conjunto completo de herramientas para diseñar interfaces de usuario, incluyendo la creación y edición de elementos como botones, iconos, cuadros de texto, imágenes y más.

Además, también permite crear prototipos interactivos y navegables, como se puede observar en la Figura 3.12, lo cual puede aportar una gran ayuda al inicio de cualquier proyecto, como en nuestro caso, donde nos aportó soporte para visualizar si la navegación resultaba intuitiva, el tamaño de los objetos interactuables eran correctos y detectar fallos de diseño de forma relativamente temprana.

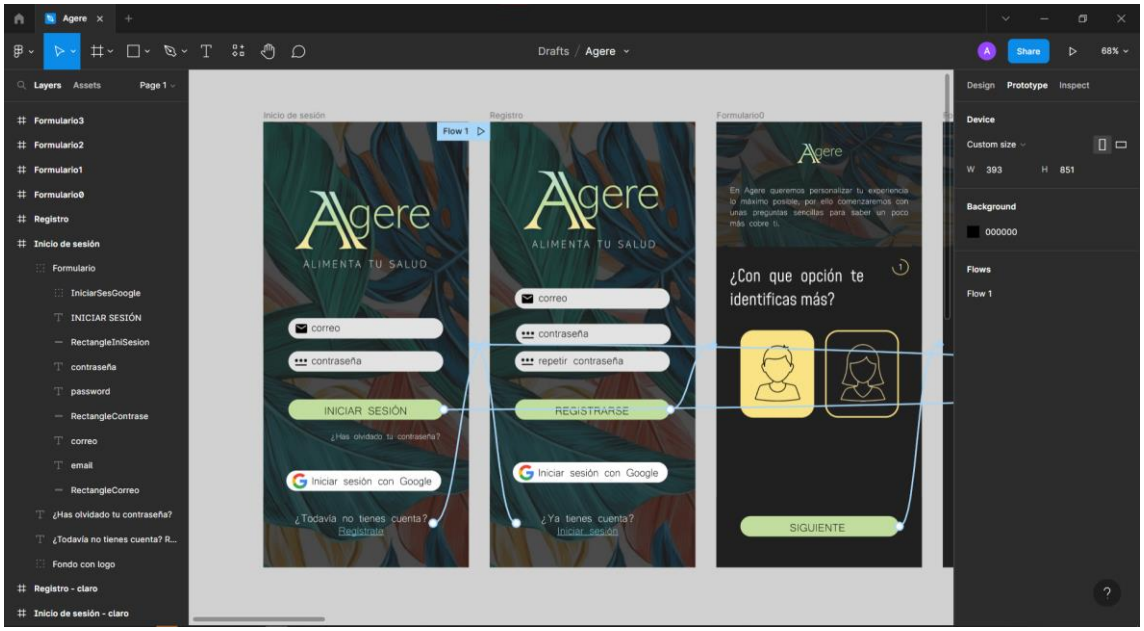


Figura 3.12 Creación de prototipos interactivos en Figma

Figma dispone también de una aplicación móvil donde, en el aspecto más funcional de la herramienta (crear y editar mockups), se encuentra bastante limitada, pero da acceso a los prototipos creados de cualquier proyecto. Esto presenta una gran ventaja, ya que no solo se va a poder simular el comportamiento en un PC, si no que se va a poder visualizar de una forma altamente temprana como será la aplicación y testear de forma prematura su usabilidad.

[13] Bug: comportamiento no deseado o incorrecto de un programa
[14] *Figma*: https://www.figma.com

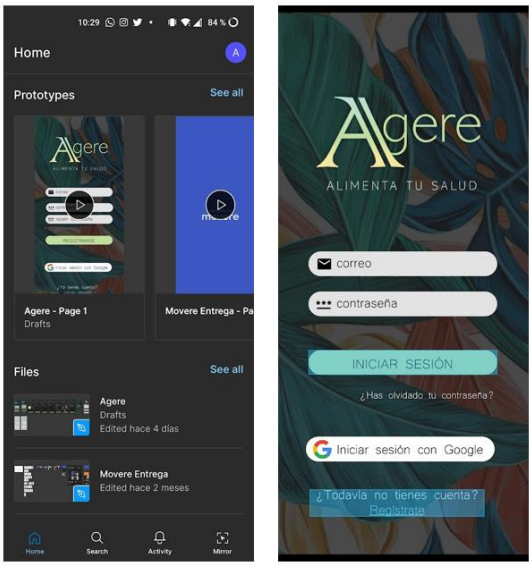


Figura 3.13 Aplicación móvil Figma y un prototipo interactivo

En la Figura 3.13 observamos a la izquierda la aplicación móvil con un listado tanto de los prototipos generados, como de los proyectos realizados. Mientras que en la derecha se muestra una captura de pantalla del prototipo de la aplicación Agere. Cuando se pulsa la pantalla, en algún espacio no interactivo, Figma te resalta con un recuadro azul los objetos interactivos los cuales, al pulsarlos, navegarán a una pantalla diferente, permitiendo al usuario visualizar una simulación de como se comportará la aplicación real y permitiendo realizar correcciones en procesos tan tempranos como este.

3.5. MidJourney

MidJourney[15] es una IA[16] que se enfoca en la generación de imágenes a partir de texto, utilizando técnicas de procesamiento de lenguaje natural y aprendizaje automático, es decir, es una *text-to-image* IA.

Esto significa que, dado un texto descriptivo, MidJourney es capaz de entender el significado detrás de las palabras y generar una imagen que representa lo que se describe en el texto.

Esta IA tiene la particularidad que interacciona con los usuarios mediante un canal de Discord, un servicio de mensajería instantánea y chat de voz VoIP[17]. Para solicitarle las peticiones únicamente se debe escribir “/imagine ” seguido del texto descriptivo que se desee generar.

En este caso particular, se ha hecho uso de esta IA para generar los avatares de la aplicación, aportando para ello textos descriptivos como “*redhead women with white background*”. Como resultado a cualquier petición, MidJourney siempre genera cuatro imágenes.

[15] MidJourney: https://www.midjourney.es
[16] IA: sigla de *inteligencia artificial*
[17] VoIP: Voz Sobre Protocolo de Internet, o en inglés *Voice Over Internet Protocol*, es una tecnología que permite realizar y recibir llamadas a través de la red

Figura 3.14 Respuesta generada por MidJourney mediante la descripción "redhead women with white background"

Tras la respuesta generada, la cual se puede observar en la Figura 3.14, con unas simples interacciones de *clic*, la IA te permite obtener "agrandada" una de las opciones (U1, U2, U3, U4), generar variaciones de una de las opciones (V1, V2, V3, V4) o volver a generar el conjunto al completo (↻) como se muestra en la Figura 3.15.

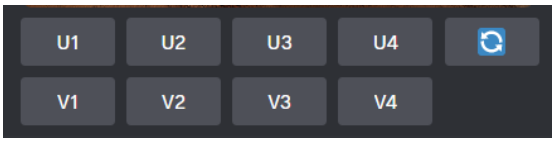


Figura 3.15 Tipos de interacción con MidJourney

Como se ha podido observar, MidJourney resulta ser una herramienta poderosa que puede ser de gran ayuda en una amplia gama de proyectos si se utiliza de manera efectiva. Además, una ventaja adicional de usar MidJourney es que todas las imágenes que genera no tienen *copyright*, lo que significa que no tendrás que preocuparte por posibles infracciones de derechos de autor al utilizarlas.

Si se aprovechan todas las herramientas y funciones que ofrece MidJourney de manera adecuada, esta herramienta puede ser un recurso valioso para cualquier proyecto que requiera diseño gráfico o imágenes.

3.6. Android Profiler

Android Profiler[18] se trata de una herramienta contenida en Android Studio que proporciona datos en tiempo real que te ayudan a comprender la forma en que tu app utiliza los recursos de la CPU, la memoria, la red y la batería. Esta herramienta resultará útil para comprobar y verificar que la aplicación no consume excesivos recursos de los dispositivos en los que esté instalado.

[18] Android Profiler: https://developer.android.com/studio/profile/android-profiler

Para poder ejecutar esta herramienta primero deberemos tener nuestra aplicación en ejecución. Luego, nos vamos a la pestaña View > Tool Windows > Profiler y con la aplicación ya en ejecución veremos lo mostrado en la Figura 3.16

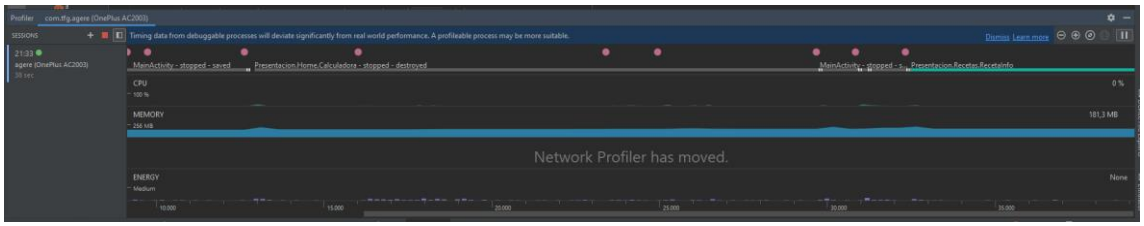


Figura 3.16 Consumo de la aplicación por Android Profiler

La Figura 3.16 muestra la interfaz que se esta mostrando y las acciones del usuario, los valores del porcentaje de CPU usada, la memoria del dispositivo en uso y el gasto energético de nuestra aplicación respectivamente.

Una aplicación móvil, por norma general, buscamos que no tenga un gran gasto energético, ya que los móviles disponen de una batería que, a pesar de que aumentan su capacidad año tras año con nuevas tecnologías, siguen siendo insuficientes si se les exige demasiado.

Utilizando la aplicación con normalidad no se observan grandes cambios en la demanda energética, como se puede observar en la Figura 3.17, pudiendo variar ligeramente según en que pantalla nos encontremos, pero nunca supera el indicador "*light*".

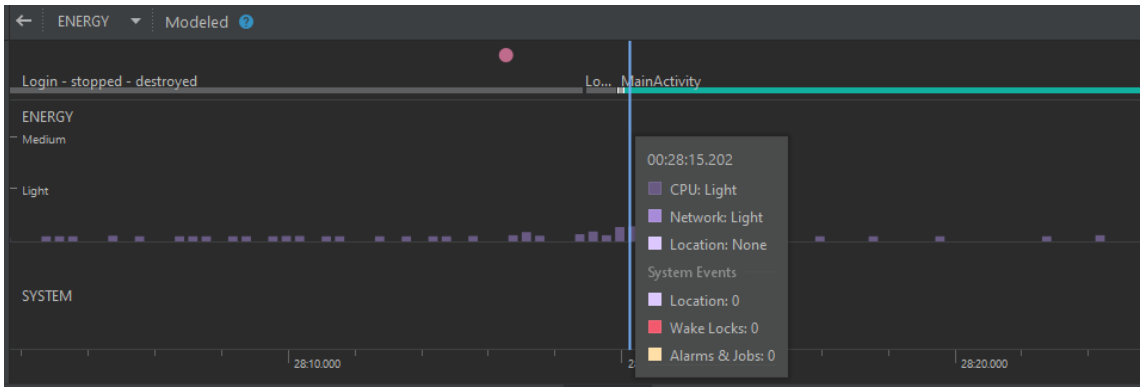


Figura 3.17 Demanda energética de Agere por Android Profiler

Respecto a la memoria, en rasgos generales, consume de media unos 189MB, como se puede observar mejor en la Figura 3.18. Esta cifra puede parecer algo alta para algunos dispositivos, pero teniendo en cuenta que la aplicación se está programando con la base de Android 8 o superior, podemos suponer que nuestros usuarios dispondrán de un dispositivo de 3GB de memoria como mínimo, por lo que el gasto de memoria de nuestra aplicación es más que aceptable.

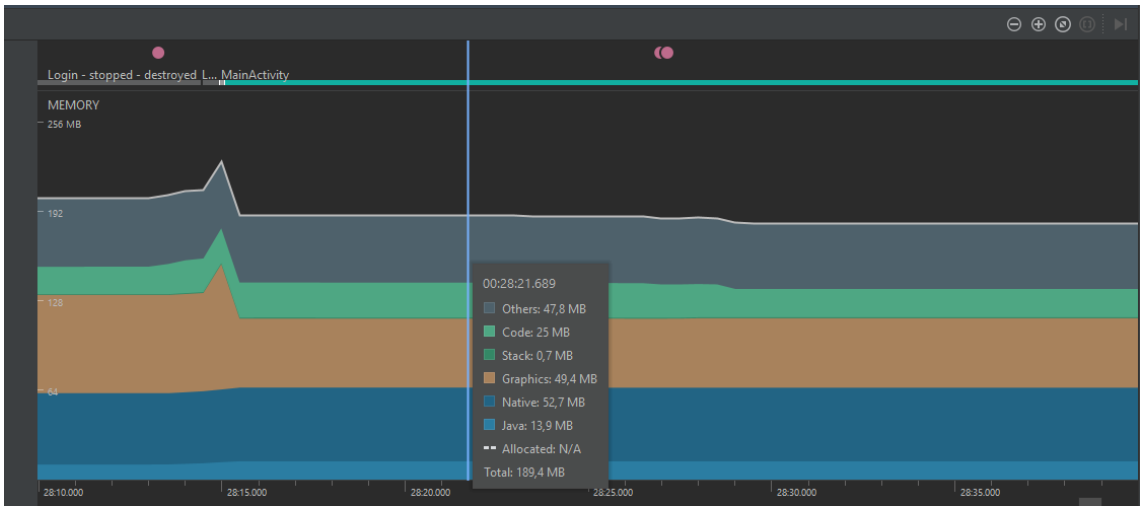


Figura 3.18 Consumo de memoria de Agere por Android Profiler

3.7. JUnit

JUnit[19] va a ser la herramienta utilizada para el testeo de pruebas automáticas, en concreto, para las pruebas automáticas que tengan relación con aspectos funcionales.

Su uso no requiere ninguna instalación específica si se está utilizando Java, pues se trata de una librería para la misma que suele venir incorporada en la mayoría de IDEs. Principalmente trataremos de probar distintos métodos que son usados en la aplicación para asuntos vitales, comprobando que su funcionamiento es correcto con los datos que corresponden.

En siguientes apartados se explicará y desarrollará el uso de esta tecnología en nuestra aplicación.

[19] JUnit: https://junit.org

Capítulo 4

Desarrollo de la solución

4.1. Metodología

Para llevar a cabo el proyecto, se ha optado por seguir la metodología *Lean Startup* [7].

La metodología Lean Startup se basa en la idea de crear y gestionar startups de manera más efectiva y con menor riesgo. Fue popularizada por Eric Ries en su libro "*The Lean Startup*" y se ha convertido en un enfoque ampliamente utilizado en el ámbito empresarial y emprendedor.

La metodología *Lean Startup* se centra en el aprendizaje validado, la iteración rápida y la experimentación continua. En lugar de seguir un enfoque tradicional de desarrollo de productos basado en planes detallados y largos ciclos de desarrollo, la metodología Lean Startup se basa en el ciclo de construir-medir-aprender.

El proceso se inicia con la creación de un Producto Mínimo Viable (MVP), que es la versión más simple del producto que permite recopilar retroalimentación y aprender de los clientes. Luego, se pone el producto en manos de los usuarios lo antes posible para obtener datos y métricas reales.

La metodología *Lean Startup* enfatiza la importancia de las hipótesis y la validación de estas mediante experimentos. Es por esto que deberemos identificar las hipótesis clave sobre el mercado, los clientes y el producto, y luego realizar pruebas para validar o refutar dichas hipótesis. Esto implica el uso de técnicas como encuestas, entrevistas, pruebas de usabilidad y análisis de datos para obtener información valiosa.

Además, la metodología *Lean Startup* promueve la flexibilidad y la capacidad de pivotar. Si los datos y la retroalimentación indican que el enfoque actual no está funcionando, deberemos poder ajustar su estrategia y realizar cambios en el producto, el mercado objetivo o el modelo de negocio [8].

En nuestro caso, adaptaremos esta metodología tratando los MVP como resultado de un *sprint* de aproximadamente 1 mes de duración.

Durante el transcurso de cada sprint, nos enfocaremos en el desarrollo de la aplicación, así como en la ejecución de pruebas y las reuniones necesarias con los expertos. Una vez finalizado el sprint, destinaremos entre una o dos semanas (dependiendo de la disponibilidad de los voluntarios) para llevar a cabo experimentos con usuarios.

Para planificar y realizar un seguimiento de todo este proceso nos hemos apoyado en la herramienta de Trello[20].

Trello es una herramienta en línea que facilita la gestión de proyectos y la organización de tareas. Permite crear tableros virtuales donde se pueden agregar listas y tarjetas

[20] Trello: https://trello.com

para representar tareas, proyectos o cualquier otro tipo de actividad que necesite ser administrada.

En nuestro caso específico, las tarjetas representaban tareas a realizar. Como se puede ver en la Figura 4.1, el título nos indica la tarea a realizar y en la descripción se muestran una serie de puntos a tener en consideración al desarrollar dicha tarea. No se han redactado como una serie de pruebas de aceptación como tal, pero al finalizar la tarea si que se comprueban que el funcionamiento de este sea el correcto y, finalmente, en algunas tarjetas, si las tareas a realizar se podían desglosar en subtareas, estas se especificaban en una *checklist*.

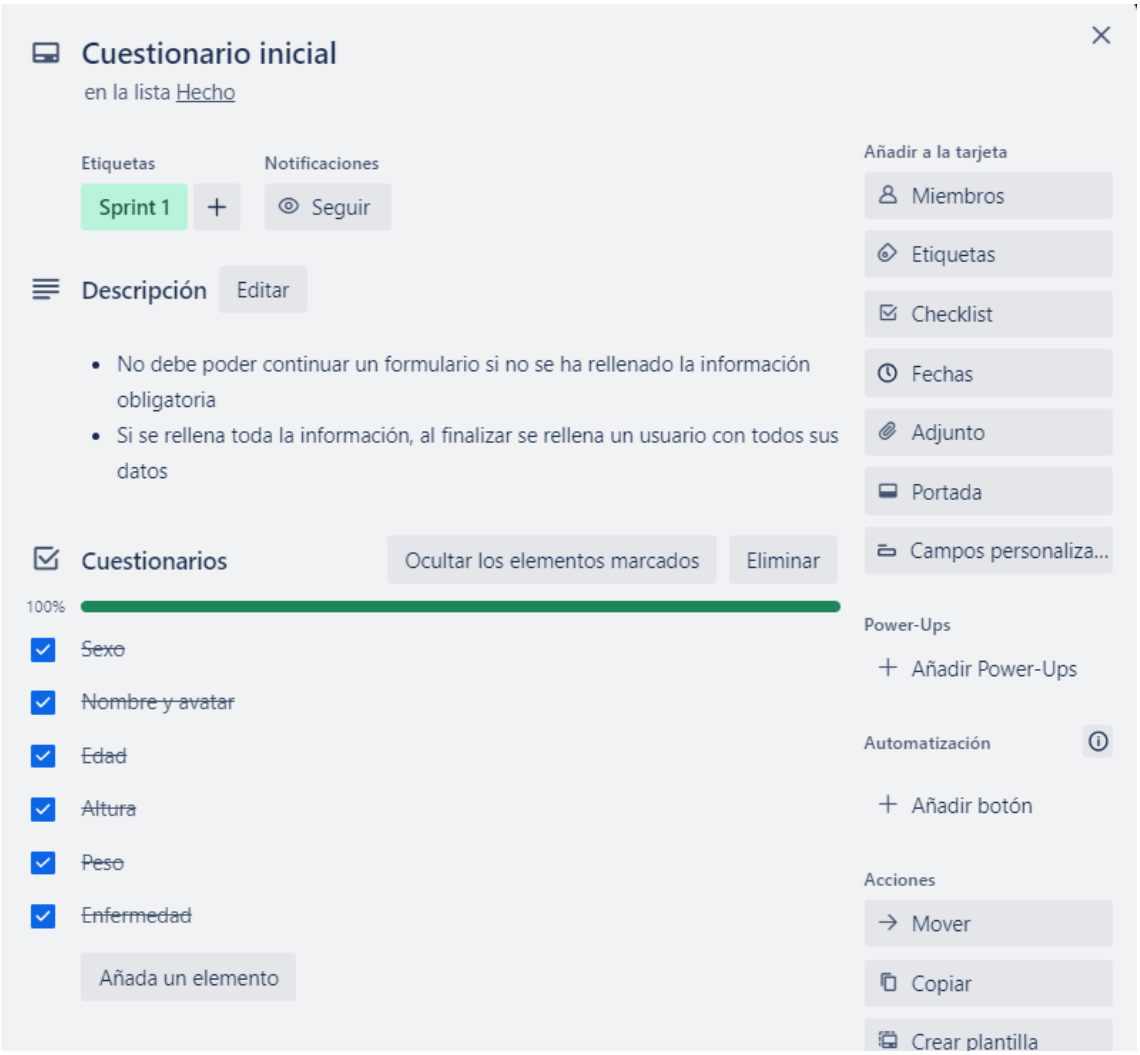


Figura 4.1 Tarjeta de una tarea en Trello

4.2. Especificación de requisitos

El backlog en la metodología ágil es una lista ordenada de elementos de trabajo pendientes que reflejan los requisitos y funcionalidades a implementar en un proyecto. En el backlog, las tareas se priorizan en función de su importancia y valor, y actúa como una guía fundamental para la planificación y desarrollo del producto de manera iterativa e incremental.

4.2.1. Requisitos de la aplicación

En el siguiente apartado desarrollaremos los aspectos imprescindibles los cuales deberían constar nuestra aplicación, Agere.

Casos de uso

Mediante el siguiente diagrama de casos de uso mostrado en la Figura 4.2, nos permitirá visualizar, además de los casos de uso identificados en nuestra aplicación, los diferentes roles del sistema y como estos interactúan con el sistema [10].

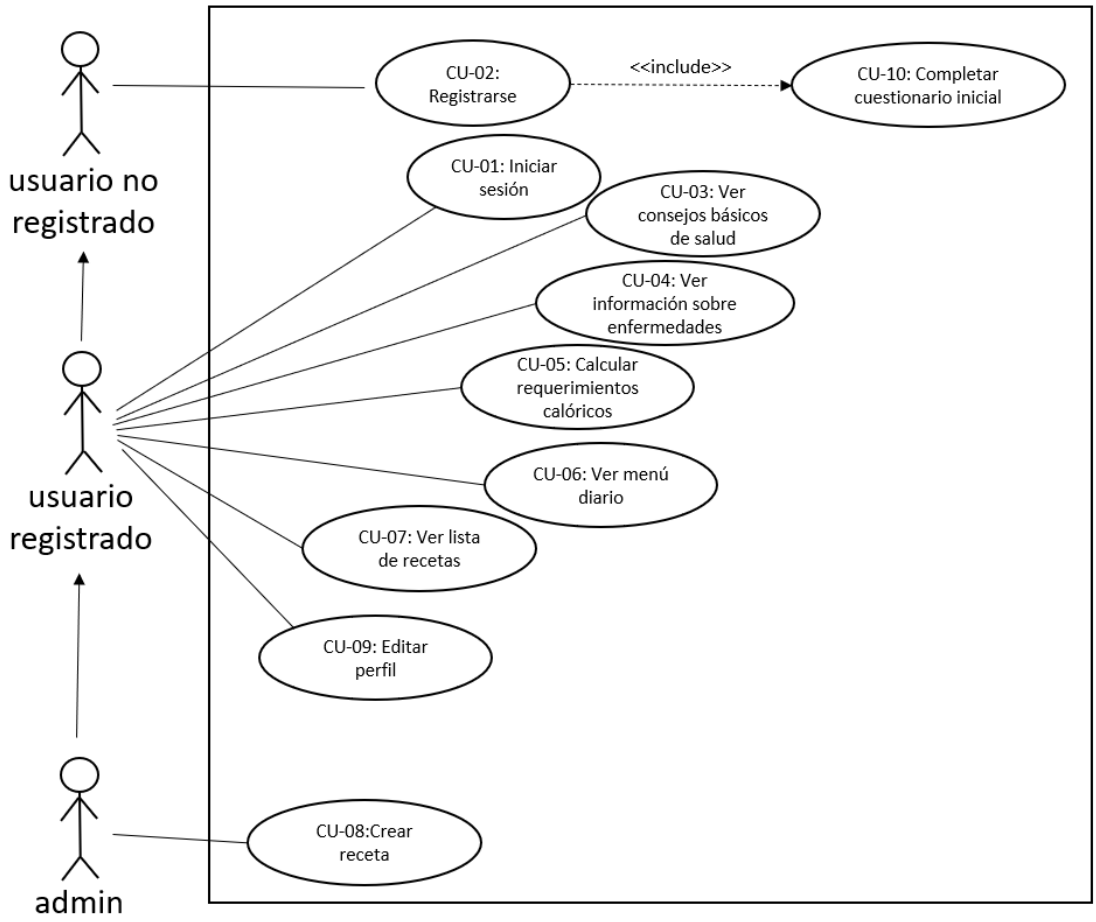


Figura 4.2 Diagrama de casos de uso

CU-01: Iniciar sesión	
Objetivo	Permitir al usuario iniciar sesión
Resumen	Para iniciar sesión los usuarios deberán indicar su email y contraseña.
Precondición	El usuario debe haberse registrado
Postcondición	Si las credenciales son correctas → navegar a la página de inicio de la aplicación Si las credenciales son incorrectas → bloquear el acceso a la aplicación y mostrar un mensaje de error indicando al usuario que sus credenciales son incorrectas

CU-02: Registro de nuevo usuario	
Objetivo	Permitir al usuario registrarse en la aplicación
Resumen	Para registrarse los usuarios deberán indicar un email al que enlazar su cuenta de la aplicación y una contraseña para iniciar sesión en un futuro en cualquier dispositivo.
Precondición	-
Postcondición	Si el correo con el que quiere registrarse ya existe → no permitir el registro y mostrar un mensaje de error indicando el motivo. Si falta algún dato por rellenar → no permitir el registro y mostrar un mensaje de error indicando el motivo Si todos los datos son correctos → registrar al usuario en base de datos y permitirle iniciar el cuestionario inicial para completar su perfil.

CU-03: Ver consejos básicos de salud	
Objetivo	Ofrecer información de salud básica
Resumen	La información básica de salud permitirá a los más inexpertos comenzar a tener una base de como cuidar su salud y a los que ya disponen de esta información permitir refrescarla para no olvidarla y tenerla presente siempre en su día a día.
Precondición	Disponer de una sesión iniciada
Postcondición	Mostrar un listado con toda la información recopilada

CU-04: Ver información sobre enfermedades	
Objetivo	Ofrecer información relativa a las enfermedades del usuario
Resumen	La información de las enfermedades que padece, ofrecerá al usuario conocimientos de las causas por las que padece la enfermedad, condiciones que pueden agravarla y como paliarla de la mejor manera posible.
Precondición	Disponer de una sesión iniciada y haber seleccionado que se padece de, al menos, una enfermedad.
Postcondición	Mostrar un listado con toda la información recopilada sobre la/las enfermedad/es que padece el usuario

CU-05: Calcular los requerimientos calóricos	
Objetivo	Permitir al usuario calcular cual debe ser su ingesta calórica diaria
Resumen	Mediante los datos del usuario y la cantidad/intensidad de deporte que practique, con una calculadora de requerimientos calóricos Harrys Benedict, se le mostrará al usuario cuantas kcal deberá consumir al día en función de sus necesidades.
Precondición	Disponer de una sesión iniciada
Postcondición	Mostrar las kcal que debe consumir el usuario

CU-06: Ver menú diario	
Objetivo	Mostrar al usuario el menú diario
Resumen	Mediante los datos del usuario, se le generará al usuario un menú en función de sus restricciones alimentarias. Además se le deberá permitir visualizar la elaboración de las mismas recetas
Precondición	Disponer de una sesión iniciada
Postcondición	Mostrar el menú del día

CU-07: Ver listado de recetas	
Objetivo	Mostrar al usuario un listado con las recetas disponibles en la aplicación
Resumen	Mostrar un listado con las recetas disponibles en la aplicación, permitiendo además, visualizar en detalle estas.
Precondición	Disponer de una sesión iniciada
Postcondición	Mostrar el listado de recetas

CU-08: Crear receta	
Objetivo	Permitir al usuario la creación de nuevas recetas
Resumen	Mediante un sencillo e intuitivo formulario, permitir al usuario añadir nuevas recetas a la base de datos de Agere
Precondición	Disponer de una sesión iniciada
Postcondición	Si falta algún dato necesario por rellenar → bloquear el avance al usuario y mostrarle un mensaje de error solicitando que rellene todos los campos Si rellena el formulario bien → crear la nueva receta en base de datos

CU-09: Editar perfil	
Objetivo	Permitir al usuario editar sus datos
Resumen	El usuario tendrá acceso a la visualización de todos sus datos y podrá editarlos (a excepción del correo asociado a su cuenta) en cualquier momento que desee
Precondición	Disponer de una sesión iniciada
Postcondición	
Requisitos asociados	RF-03, RNF-01, RNF-02

CU-10: Cuestionario inicial	
Objetivo	Recopilar la información inicial necesaria del usuario
Resumen	Un conjunto de preguntas se le realizarán al usuario con el fin de recopilar la información inicial necesaria del usuario para poder comenzar a hacer uso de la aplicación
Precondición	Haberse registrado previamente
Postcondición	Si falta introducir algún dato → bloquear el avance del usuario y mostrar un error indicando que faltan datos por introducir Si se han introducido todos los datos → actualizar todos los datos del usuario

Mockups

Los mockups son una herramienta muy útil en el proceso de diseño de la interfaz de una aplicación móvil, ya que permiten crear representaciones visuales de la interfaz antes de construir la versión final. Esto puede ayudar a obtener una idea clara de cómo será la aplicación, a identificar posibles problemas de usabilidad y a realizar ajustes antes de invertir tiempo y recursos en la construcción del producto final.

En el caso de Agere, la creación de los mockups fue evolucionando a lo largo de varias etapas, hasta alcanzar el producto final. El tipo de mockup que se utiliza dependerá de la etapa del proceso de diseño en la que se encuentre el proyecto. Los mockups de baja fidelidad son adecuados para la etapa inicial del proceso de diseño, mientras que los mockups de alta fidelidad y los prototipos interactivos son adecuados para la etapa de diseño detallado y las pruebas de usabilidad.

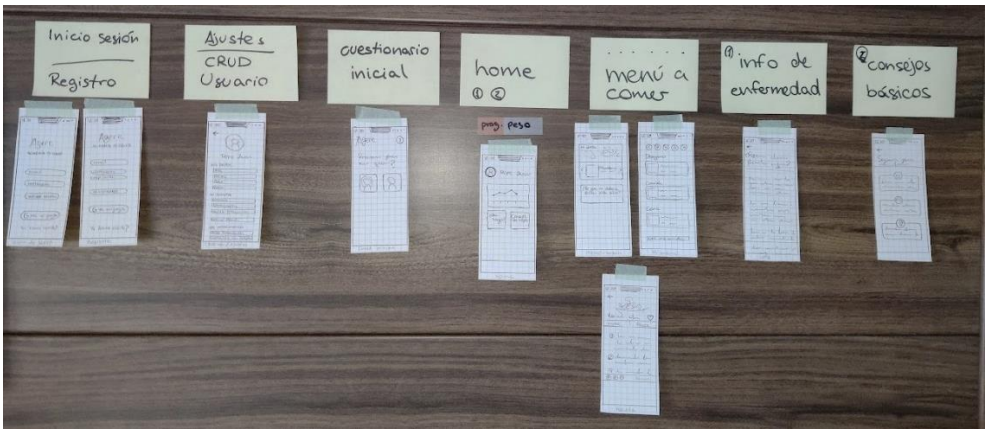


Figura 4.3 Mockups de baja fidelidad

En la Figura 4.3 se observan unos mockups esquemáticos, no contienen detalles específicos de diseño, como colores o tipografía. Estos fueron útiles para comenzar a asentar lo que sería el diseño futuro de Agere y poder hacerse una idea temprana de cuantas ventanas dispondría la aplicación y como estas se distribuirían.

Previamente a comenzar la implementación de la aplicación, se realizaron unos mockups más detallados:

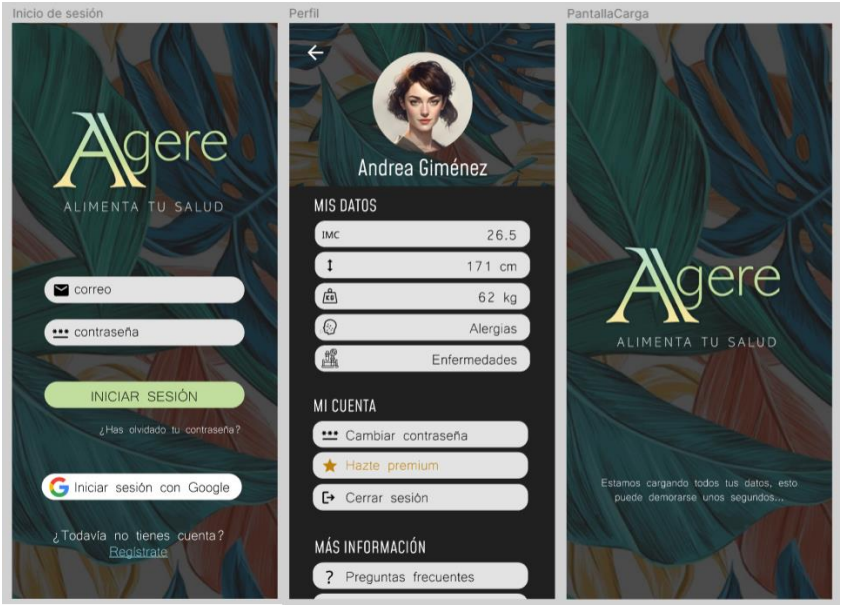


Figura 4.4 Mockups de alta fidelidad

Estos mockups permiten tener una visualización más fidedigna de lo que se deberá implementar en etapas posteriores del desarrollo de la aplicación y una idea más clara de como será el producto final.

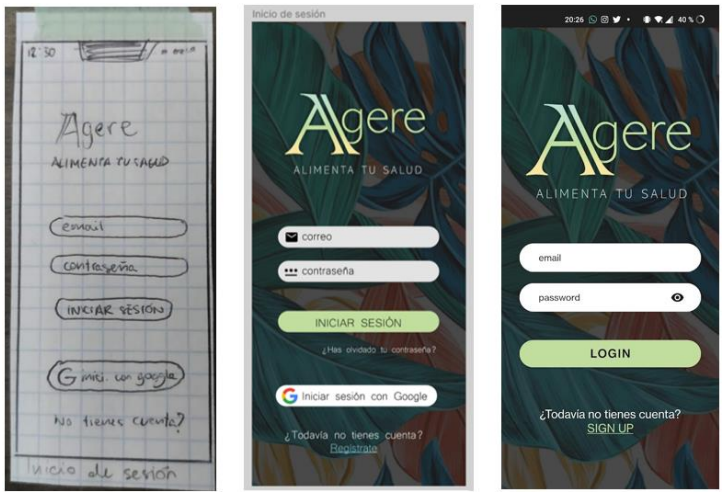


Figura 4.5 Mockup de baja fidelidad - Mockup de alta fidelidad - Captura de pantalla

Se puede observar como los mockups de baja fidelidad dan la base a los mockups de alta fidelidad, los cuales, son bastante cercanos a lo que finalmente se logró en la aplicación.

En este caso, para la realización de los mockups de alta fidelidad se hizo uso de la herramienta de generación de prototipos Figma, la cual no solo permite generar mockups, si no que también permite simular la navegación entre las diferentes ventanas diseñadas para poder visualizar, de manera extremadamente temprana, como sería la aplicación en un futuro.

Requisitos no funcionales

- **RNF-01** – Cargar los datos de la aplicación en menos de 1 segundo.
- **RNF-02** – Los datos modificados deben ser cambiados en base de datos en menos de 2 segundos.
- **RNF-03** – La aplicación debe asegurar que los datos estén protegidos del acceso no autorizado.
- **RNF-04** – La aplicación debe ser capaz de operar adecuadamente con hasta 100 usuarios con sesiones concurrentes.
- **RNF-05** – La aplicación debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
- **RNF-06** – La aplicación debe tener una disponibilidad del 90% de las veces en que un usuario intente accederlo.
- **RNF-07** – La aplicación debe ser funcional hasta, al menos, la versión de Android 12.0

Profesionales implicados

La salud es un aspecto fundamental en la vida de las personas y, en el caso de una aplicación móvil que busca impactar en la salud del usuario, es crucial contar con información precisa y verificada. Por esta razón, se ha buscado la ayuda de grandes profesionales en el ámbito de la salud, quienes han brindado su experiencia y conocimientos para asegurar que la aplicación ofrezca información confiable y respaldada por profesionales.

A lo largo de sucesivos meses, se han mantenido reuniones y conversaciones con estos expertos para recopilar la información más precisa y actualizada sobre la salud, la nutrición y otros aspectos relacionados que resulten relevantes para la aplicación. La colaboración de estos profesionales ha sido fundamental para garantizar que la aplicación cuente con información rigurosa y de calidad.

Además, la colaboración de estos expertos ha permitido que la aplicación pueda ofrecer recomendaciones personalizadas y adaptadas a las necesidades específicas de cada usuario. La información brindada ha sido rigurosamente analizada y verificada, para asegurar que cumple con los estándares más exigentes en el ámbito de la salud.

Reuniones realizadas

Esperanza Garcés – Médico – 22/12/22 : Abordar la idea principal de la aplicación y explicación del proyecto. Qué tipo de personas van a querer hacer uso de la aplicación.

Esperanza Garcés – Médico – 04/02/23 : Primer borrador de posibles enfermedades. Como se relacionan las enfermedades con los alimentos a ingerir.

Esperanza Garcés – Médico – 11/02/23 : Documento final de enfermedades con links a webs de interés relacionadas y alimentos a evitar.

Javier Valentín – Enfermero – 12/03/23 : Alergias y enfermedades. Riesgos que conllevan para la salud del usuario.

María Domingo – Enfermera/Estudiante de medicina – 13/03/23 : Que alimentos de deberían evitar en una dieta saludable.

Javier Valentín – Enfermero – 16/03/23 : Que consejos básicos de salud debería conocer cualquier tipo de usuario. Guía inicial para usuarios inexpertos.

Javier Valentín – Enfermero – 25/04/23 : Reunión para obtener unas recomendaciones y dietas básicas a seguir.

María Domingo – Enfermera/Estudiante de medicina – 26/05/23 : Reunión de control para comprobar la veracidad de los datos mostrados en la aplicación.

Documentos recopilados

Tras la realización de sucesivas reuniones, se han recopilado documentos que han servido de apoyo para completar la información mostrada en Agere.

Enfermedades con importantes restricciones alimenticias e información sobre las mismas

Este documento contiene la información a partir de la cual se han podido generar gran parte de la estructura de la aplicación. Con ella se ha podido saber que alimentos se debían evitar según que enfermedad padezca el paciente, además de una breve introducción sobre qué es la enfermedad en sí.

Hipertensión Arterial

La presión arterial es una medición de la fuerza ejercida contra las paredes de las arterias a medida que el corazón bombea sangre a su cuerpo. Hipertensión es el término que se utiliza para describir la presión arterial alta.

A evitar: sal, alcohol, tabaquismo. Dieta DASH.

Dislipemia

La dislipidemia (o dislipemia) es una concentración elevada de lípidos (colesterol, triglicéridos o ambos) o una concentración baja de colesterol rico en lipoproteínas (HDL). Tiene relación con el estilo de vida, con la genética, con las enfermedades (como concentraciones bajas de hormona tiroidea o enfermedad renal), con los medicamentos o con una combinación de estos factores. Puede producir ateroesclerosis, que da lugar a angina de pecho, infarto de miocardio, accidente cerebrovascular y arteriopatía periférica.

A evitar: Grasas trans y saturadas.

Diabetes Mellitus

La diabetes mellitus es un trastorno en el que el organismo no produce suficiente cantidad de insulina o no responde normalmente a la misma, lo que provoca que las concentraciones de azúcar (glucosa) en sangre sean anormalmente elevadas.

A evitar: Azúcares añadidos, hidratos de absorción rápida, alcohol, grasas trans, etc.

Hipertiroidismo

El hipertiroidismo ocurre cuando la glándula tiroides produce demasiada cantidad de hormona tiroidea. Esta afección también se conoce como tiroides hiperactiva. El hipertiroidismo acelera el metabolismo del cuerpo, lo que puede provocar muchos síntomas: pérdida de peso, temblor de manos, latidos cardíacos acelerados o irregulares, etc.

A evitar: Yodo, presente en sal yodada, algas (especialmente kelp y alga roja), lácteos, soja y yema de huevo especialmente.

Insuficiencia renal crónica

Se llama enfermedad renal crónica (ERC) a un daño prolongado que sufren los riñones y que puede empeorar con el transcurso del tiempo. Este daño puede ocasionar que los desechos se acumulen en su cuerpo y causen otros problemas . Si el daño es severo, los riñones pueden dejar de funcionar. A esta situación se la llama falla renal y significa que la persona necesita recibir diálisis o un trasplante renal.

A evitar: acumulación de sodio y potasio.

Hiperuricemia

El ácido úrico es el producto final de la degradación de ciertas proteínas, bien de nuestro cuerpo, bien de proteínas ingeridas con la dieta. Este ácido se elimina principalmente por el riñón formando parte de la orina.

La hiperuricemia es la elevación del ácido úrico en la sangre, en general por encima de 7 mg/dl. Es una alteración muy frecuente que se observa en más de 5 de cada 100 personas adultas de la población.

En general el aumento del ácido úrico no produce síntomas y es un hallazgo casual al realizar un análisis de sangre por cualquier otro motivo. En algunas personas el aumento del ácido úrico puede:

- Favorecer la formación de piedras en el riñón **(nefrolitiasis)** y, por tanto, la aparición de cólicos nefríticos.
- Favorecer un deterioro progresivo de la función renal **(nefropatía por uratos)**
- Producir ataques de inflamación aguda de las articulaciones **(gota)** siendo el más característico la afectación del dedo gordo del pie **(podagra).**

A evitar: carnes rojas (cerdo, ternera, cabrito), mariscos, pescados (trucha, atún, palometa, vieiras, anchoa, arenque, sardinas y atún en aceite), tocino, vísceras, pavo, cordero…

Consejos básicos de salud

A partir del siguiente documento se pudieron generar lo que serían en un futuro unos consejos básicos de salud que cualquier usuario debería seguir y adoptar, de forma genérica, para comenzar a llevar unos mínimos cuidados en su salud.

Ajustar la ingesta calórica a los requerimientos individuales de cada uno. Entre 1800-2000 en mujeres y 2000-2400 en hombres promedio.

Limitar el azúcar libre (azúcar blanco) (menos de un 10% del total en la dieta diaria, recomendable hasta un 5%)

Reducir el consumo de sal a menos de 5 gramos diarios

Relación de macronutrientes de menos del 30% de grasa (entre un 50-70% hidratos de carbono y unos 10-15% de aporte calórico de proteínas (unos 0,8g de peso))

Priorizar el consumo de las grasas no saturadas por encima de las saturadas (saturadas menos del 10% del total de cal) evitar bajo toda costa las grasas trans.

Hidratos de carbono: priorizar las alternativas integrales,

Tomar al menos 5 porciones de frutas y hortalizas al día, exceptuando tubérculos feculentos como las patatas.

Las recomendaciones diarias de fibra son 20-30g/día.

- Las recomendaciones de fibra soluble al día son de 5-10g. Obteniendo mayores beneficios de un consumo de 10-20g diarios.

- Fibra insoluble: Podemos encontrarla en el salvado de trigo, los cereales integrales y vegetales. Tiene la propiedad de captar agua y aumentar el volumen del contenido intestinal. Produce una estimulación mecánica del tránsito intestinal, favoreciendo la evacuación. Este efecto es mayor si se consume acompañada de agua.

- Fibra soluble: Está presente en las frutas, verduras, legumbres, avena, cebada, etc. Es fermentada por las bacterias del colon, favoreciendo el crecimiento y mantenimiento de la flora intestinal. La fibra soluble atrapa agua y forma una solución viscosa en el intestino, haciendo la digestión más lenta y recubriendo la pared intestinal de una capa gruesa. Esta acción disminuye la absorción intestinal de algunos nutrientes como la glucosa (beneficioso para los diabéticos) y el colesterol, contribuyendo así a la prevención de enfermedades cardiovasculares.

Proteínas: puede ser necesario subir estos requerimientos en momentos demandantes de la vida (menopausia, vejez…).

Evitar el consumo de alcohol y tabaco e incluso suprimirlos.

Beber la cantidad de agua suficiente al día, ajustado a los requerimientos de cada persona (35ml por kg del usuario). Evitar como bebida principal todo aquello que no sea agua.

HIGIENE DEL SUEÑO

Se recomienda dormir entre 7-8 horas diarias, en horario nocturno.

ACTIVIDAD FÍSICA

Evitar el sedentarismo

150 min de ejercicio aeróbico moderado o 75min de ejercicio vigoroso en sesiones de 10min como mínimo. Viéndose aumentados los beneficios de actividad a los 300min de ejercicio en intensidad moderada o 150min en intensidad alta.

2 o más veces de ejercicio anaeróbico de fortalecimiento muscular

4.3. Arquitectura del código

La estructura del código se dividió en los 3 grupos mostrados en la Figura 4.6

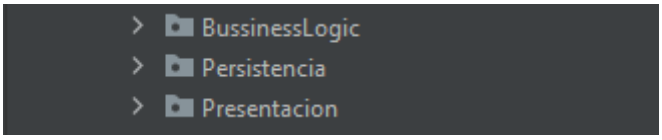


Figura 4.6 Estructura del código

BussinessLogic

En esta carpeta se encuentra:

- Servicios: para realizar interacciones directas con la base de datos.
- Constantes: se utilizarán a lo largo del código para mejorar la legibilidad y asegurar una buena mantenibilidad del código y evitar errores causados por una mala asignación de valores incorrectos.
- Entidades: donde se establecen las estructuras de datos que se usarán a lo largo de todo el código.

Persistencia

El contenido de esta carpeta se trata en el siguiente apartado [5.3], donde se explica detalladamente el contenido de la estructura que comunica con la base de datos.

Presentación

Aquí se establece, ordenado por carpetas, todas las ventanas a las que tendrá acceso y podrá interactuar el usuario final.

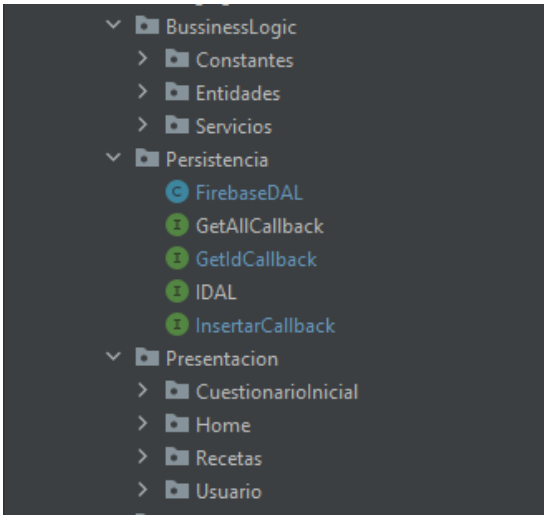


Figura 4.7 Arquitectura del código en detalle

4.4. Programación

Durante el desarrollo de la aplicación nos fuimos encontrando con ciertos retos para generar un código de la forma más óptima posible, como son la aplicación de patrones o las refactorizaciones de código. A su vez nos encontramos con ciertas dificultades para implementar algunas funcionalidades, en concreto, referentes a la base de datos y la conexión a esta desde la aplicación.

4.4.1. Patrones de programación. *Singleton*

El patrón de diseño *Singleton* es una técnica creacional que garantiza la existencia de una única instancia de una clase y proporciona un punto de acceso global a dicha instancia. Esto significa que, dentro de un programa, solo habrá una única instancia de la clase *Singleton* en cualquier momento dado [10].

La implementación de este patrón involucra la creación de un constructor privado en la clase deseada, lo que evita que se instancie directamente desde otros lugares del código. En su lugar, la clase *Singleton* proporciona un método estático que permite acceder a la instancia única. Si aún no se ha creado una instancia, este método se encargará de crearla y devolverla. Si la instancia ya existe, simplemente se devuelve sin crear una nueva.

El patrón *Singleton* es útil en situaciones en las que se necesita una única instancia de una clase para coordinar acciones en todo el programa. En nuestra aplicación este caso se dio en una clase que denominamos DatosBD, la cual su finalidad era gestionar los datos que se administran desde la aplicación.

```
8   public class DatosBD {
9
10      public Usuario usuario;
11      public List<Receta> recetas;
12
13      private static DatosBD instance;
14
15      private DatosBD() {
16      }
17
18      public static DatosBD getInstance() {
19          if (instance == null) {
20              instance = new DatosBD();
21          }
22          return instance;
23      }
24
```

Figura 4.8 Clase DatosBD

Como podemos observar en la Figura 4.8, esta clase mantendría durante el flujo de la aplicación un único usuario y una única lista de recetas para el usuario, gracias a establecer el constructor de la clase como privado y dando como única inicialización a la misma el método *getInstance()*, el cual, en caso de ser la primera vez que se inicializa,

se le da un nuevo valor y en caso contrario, se devolverá siempre la misma instancia de nuestra clase.

4.4.2. Refactorizaciones.

Duplicidad en código

Al analizar nuestro código, es evidente que nos encontramos con múltiples variables temporales, y en muchos casos, su presencia se repite numerosas veces. Como se observa en la Figura 4.9.

```
private void calcularDatos(){
    // HOMBRES: (10 x peso en kg) + (6,25 × altura en cm) - (5 × edad en años) + 5
    // MUJERES: (10 x peso en kg) + (6,25 × altura en cm) - (5 × edad en años) - 161
    String peso = this.peso.getEditText().getText().toString();
    String altura = this.altura.getEditText().getText().toString();
    String edad = this.edad.getEditText().getText().toString();
    if(!peso.equals("") && !altura.equals("") && !edad.equals("")){
        double res = paraMantener();

        mantenerse.setText(String.format("%.2f", res*valActividad) + " kcl");
        bajarLento.setText(String.format("%.2f", (res-300)*valActividad )+ " kcl");
        bajarRapido.setText(String.format("%.2f", (res-600)*valActividad) + " kcl");
        subirRapido.setText(String.format("%.2f", (res+600)*valActividad) + " kcl");
        subirLento.setText(String.format("%.2f", (res+300)*valActividad) + " kcl");
    }

}
```

```
private double paraMantener(){
    String peso = this.peso.getEditText().getText().toString();
    String altura = this.altura.getEditText().getText().toString();
    String edad = this.edad.getEditText().getText().toString();
    String sexo = this.sexo.getEditText().getText().toString();
    double form = (10 * Double.parseDouble(peso)) + (6.25 * Double.parseDouble(altura));
    double res1;
    if(sexo.equals("Hombre")){
        res1 = (5 * Double.parseDouble(edad)) + 5;
    }else{
        res1 = (5 * Double.parseDouble(edad)) - 161;
    }

    return form - res1;
}
```

Figura 4.9 Métodos que realizan el calculo de kcal a consumir

En el caso específico de las variables temporales repetidas, podemos considerar reemplazarlas por llamadas a métodos que realicen esas tareas temporales. De esta manera, podemos encapsular la lógica relacionada en un solo lugar y reutilizarla en diferentes partes del código, evitando la necesidad de repetir el mismo código una y otra vez.

Así pues, en la Figura 4.10, observamos los métodos que realizan dichas tareas temporales y como los métodos *calcularDatos()* y *paraManterner()* resultan mucho más sencillos de comprender.

```
private void calcularDatos(){
    // HOMBRES: (10 x peso en kg) + (6,25 x altura en cm) - (5 x edad en años) + 5
    // MUJERES: (10 x peso en kg) + (6,25 x altura en cm) - (5 x edad en años) - 161
    if(!getPeso().equals("") && !getAltura().equals("") && !getEdad().equals("")){
        double res = paraMantener();

        mantenerse.setText(String.format("%.2f", res*valActividad) + " kcl");
        bajarLento.setText(String.format("%.2f", (res-300)*valActividad )+ " kcl");
        bajarRapido.setText(String.format("%.2f", (res-600)*valActividad) + " kcl");
        subirRapido.setText(String.format("%.2f", (res+600)*valActividad) + " kcl");
        subirLento.setText(String.format("%.2f", (res+300)*valActividad) + " kcl");
    }

}

private double paraMantener(){
    double form = (10 * Double.parseDouble(getPeso())) + (6.25 * Double.parseDouble(getAltura()));
    double res1;
    if(getSexo().equals("Hombre")){
        res1 = (5 * Double.parseDouble(getEdad())) + 5;
    }else{
        res1 = (5 * Double.parseDouble(getEdad())) - 161;
    }

    return form - res1;
}


private String getPeso() { return this.peso.getEditText().getText().toString(); }

private String getAltura() { return this.altura.getEditText().getText().toString(); }

private String getEdad(){ return this.edad.getEditText().getText().toString(); }

private String getSexo(){ return this.sexo.getEditText().getText().toString(); }
```

Figura 4.10 Código del cálculo de kcal a consumir refactorizado

Métodos extensos

En la creación de una aplicación móvil haciendo uso del lenguaje Java se crea una estructura peculiar en la que por un lado, encontramos contenidas las interfaces, con su código xml y por otro, las clases asociadas a sus interfaces, donde se le dota de funcionalidad a cada ventana individualmente. Si la información mostrada en pantalla es estática y no varía, la clase *.java* prácticamente carece de sentido ya que no aporta ninguna funcionalidad extra, pero en caso de ventanas que interactúan plenamente con el usuario, podemos llegar a observar códigos extensos.

Esto es debido a que para poder enlazar cada elemento de la interfaz (botones, textos, imágenes, etc.) al código, para así poder editar su funcionalidad, cambiar su texto en momentos determinados o acciones similares, deben declararse como variables en nuestra clase *.java*. Además, si nuestra funcionalidad va a depender de *listeners*, es decir, cuando se detecten ciertos cambios, esta inicialización se hace más compleja y extensa.

Las inicializaciones de variables en una aplicación móvil, a diferencia de una aplicación Java normal que se realizan en el constructor, deben realizarse en el método sobrescrito *onCreate().*

Uno de los problemas que pudimos observar es que a lo largo de toda la aplicación se abusó en exceso de este método, dando lugar a métodos *onCreate()* como el que se presenta en la Figura 4.11, donde dicho método abarcaba una extensión de más de 120 líneas de código.

Figura 4.11 Método onCreate() sin refactorizar

Al ser conscientes de esto, se decidió realizar una refactorización para hacer más legible este método, encapsulando secciones de código en métodos privados de la propia clase. Gracias a esto, el método *onCreate()*, pudo reducirse considerablemente como se puede observar en la Figura 4.12. Cabe destacar que no todas las inicializaciones ni *listeners* se abstrajeron en este caso específico, debido a que esas variables no se volvían a invocar en otras partes del código, por lo que, por optimización y no mantener unas variables globales que no se usan, se mantuvieron en el método *onCreate()*.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getSupportActionBar().hide();
    setContentView(R.layout.activity_calculadora);

    datosBD = DatosBD.getInstance();
    Usuario usu = datosBD.getUsuario();

    ImageView atras = findViewById(R.id.imageAtras);
    TextInputEditText sexoEdit = findViewById(R.id.sexoEdit);
    TextInputEditText actividadEdit = findViewById(R.id.actividadEdit);

    inicializarVariables();
    setConfiguracionInicial(usu);
    calcularDatos();

    inicializarListeners();

    atras.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            irAtras();
        }
    });

    actividadEdit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mostrarActividades();
        }
    });

    sexoEdit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mostrarSexo();
        }
    });
}
```

Figura 4.12 Método onCreate() refactorizado

Como resultado, los métodos creados resultaron abstraídos como se muestra en la Figura 4.13

```java
private void inicializarVariables(){
    actividad = findViewById(R.id.actividadText);
    peso = findViewById(R.id.pesoText);
    altura = findViewById(R.id.alturaText);
    edad = findViewById(R.id.edadText);
    sexo = findViewById(R.id.sexoText);

    bajarLento = findViewById(R.id.bajarLento);
    bajarRapido = findViewById(R.id.bajarRapido);
    mantenerse = findViewById(R.id.mantenerse);
    subirLento = findViewById(R.id.subirLento);
    subirRapido = findViewById(R.id.subirRapido);
}

private void setConfiguracionInicial(Usuario usu){
    peso.getEditText().setText(usu.getPeso() + "");
    altura.getEditText().setText(usu.getAltura()+"");
    edad.getEditText().setText(usu.getValorEdad()+"");
    sexo.getEditText().setText(usu.getValorSexo());
}
```

Figura 4.13 Métodos auxiliares resultado de la refactorización (1)

A pesar de los esfuerzos por minimizar código, el método para inicializar los *listeners* continúa siendo extenso debido a su propia estructura, la cual obliga a declarar ciertos métodos a pesar de que, para nuestra funcionalidad, no sean necesarios, como puede verse en la Figura 4.14.

Sin embargo, el objetivo principal de esta refactorización se logró satisfactoriamente, permitiendo ocultar de alguna forma toda la inicialización de estos métodos, que solo disminuían la legibilidad del código.

```
private void inicializarListeners(){
    actividad.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mostrarActividades();
        }
    });

    sexo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mostrarSexo();
        }
    });

    peso.getEditText().addTextChangedListener(new TextWatcher() {
        @Override
        public void afterTextChanged(Editable s) {}
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            calcularDatos();
        }
    });

    altura.getEditText().addTextChangedListener(new TextWatcher() {
        @Override
        public void afterTextChanged(Editable s) {}
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            calcularDatos();
        }
    });

    edad.getEditText().addTextChangedListener(new TextWatcher() {
        @Override
        public void afterTextChanged(Editable s) {}
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            calcularDatos();
        }
    });
}
```

Figura 4.14 Métodos auxiliares resultado de la refactorización (2)

4.4.3. Desafíos en *back-end*

El *back-end* es la parte de la aplicación que maneja la lógica y la gestión de datos detrás de la interfaz de usuario, es decir, la parte invisible de la aplicación que permite que el usuario realice acciones y obtenga resultados.

Base de datos

En este proyecto se ha elegido como base de datos *Cloud Firestore*, la base de datos de *Firebase* NoSQL. Además, también es una base de datos en tiempo real, lo que significa que las actualizaciones de datos se propagan automáticamente a través de todos los dispositivos y usuarios que están conectados a la base de datos.

A pesar de tratarse de una base de datos NoSQL y no exigir en ningún momento ninguna estructura específica en cada objeto que subimos o actualizamos, por ejemplo, en Firebase se puede subir un usuario que contenga como elementos el nombre, un email y un rol, y a continuación subir un usuario que únicamente contenga un nombre y esto no resulta en ningún error

En nuestro caso hemos construido unas estructuras y relaciones que, mediante las entidades mencionadas anteriormente al inicio punto 5.2 Estructura del código, podremos asegurarnos consistencia en *Firestore*.

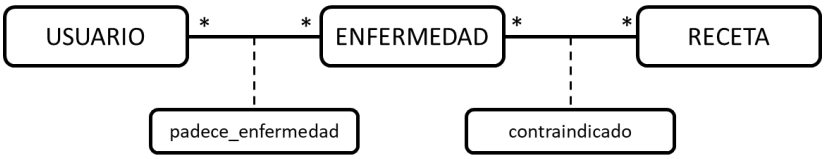


Figura 4.15 Estructura de datos

A pesar de que pueda verse extremadamente simple la relación de los datos, nuestra aplicación no necesita más complejidad dada la simplicidad inicial de nuestro proyecto a nivel de lógica. Nuestra base de datos, por el momento, únicamente requiere almacenar información de los usuarios individualmente, las enfermedades que pueden padecer y las recetas que están contraindicadas en función de las enfermedades que padezca.

- En caso de que el usuario no padezca ninguna enfermedad, podrá recomendársele cualquier receta.
- En caso de que el usuario padezca alguna enfermedad, si esta tiene contraindicaciones alimenticias, no se le recomendarán las recetas contraindicadas.

Comunicación

En el lenguaje escogido para el desarrollo de Agere nos encontramos con un inconveniente, y es que, en Java, el hilo principal no podía ser bloqueado a la espera de la obtención de los resultados de una consulta a la base de datos o la confirmación de una correcta subida de los mismos. La solución fue comunicarse con la base de datos mediante *callbacks*.

Los *callbacks*, o también conocido como funciones de retorno, son unas funciones que se pasan como argumento a otra función y que se ejecuta después de que se complete

la función principal. El objetivo principal de los *callbacks* es permitir mantener un orden y control sobre la ejecución de los códigos asíncronos.

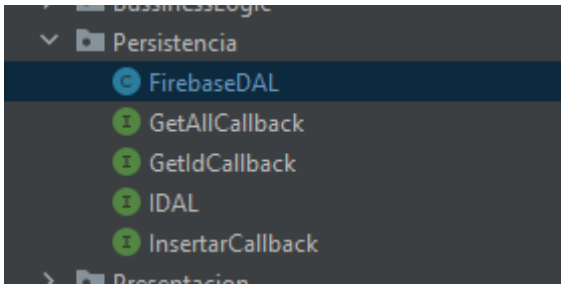


Figura 4.16 Estructura de la persistencia en Agere

Como se puede observar en la Figura 4.16, la persistencia consta de cuatro interfaces[21] que permitirán dar una correcta estructura a la clase FirebaseDAL.

GetAllCallback

```
7  public interface GetAllCallback {
8      //En este método obtenemos los resultados de los getAll();
9      void ObtenidosTodosLosDatos(List<QueryDocumentSnapshot> datos);
10 }
```

Figura 4.17 Interfaz GetAllCallback

La interfaz *GetAllCallback*, mostrada en la Figura 4.17, consta únicamente de un método, *ObtenidosTodosLosDatos*, el cual recibe como argumento una lista de los datos obtenidos en una consulta a la base de datos.

Esta interfaz es usada cuando se hacen llamadas a base de datos y la respuesta espera ser un conjunto de documentos, pudiendo ser este un conjunto vacío, con un único elemento o muchos elementos.

GetIdCallback

```
5  public interface GetIdCallback {
6      //Documento obtenido con el id
7      void ObtenidoUnDatoPorId(DocumentSnapshot obtenido);
8  }
```

Figura 4.18 Interfaz GetIdCallback

La interfaz *GetIdCallback*, mostrada en la Figura 4.18, como en el caso anterior, implementa únicamente un método, ObtenidoUnDatoPorId, en este caso recibe como argumento un único dato, en lugar de un conjunto de los mismos.

Esta interfaz es usada cuando se hacen llamadas a base de datos y se espera como respuesta un único dato, por ejemplo, cuando se realizan búsquedas en función del id del documento, en estas llamadas, Firestore nos devolverá un único documento, siempre y cuando este exista.

[21] Interfaces: en Java, interfaz es una colección de métodos abstractos y propiedades constantes.

InsertarCallback

```
public interface InsertarCallback {
    //Si usamos una interfaz podemos acceder a los objetos una vez el listener haya finalizado
    void InsertadoCorrectamente(DocumentReference documentReference);
}
```

Figura 4.19 Interfaz InsertarCallback

La interfaz *InsertarCallback*, mostrada en la Figura 4.19, se usa al guardar cualquier tipo de documento en base de datos, indistintamente si es un nuevo documento o una sobreescritura. El método *InsertadoCorrectamente* nos permite tener acceso al documento que se ha guardado en base de datos.

Esta interfaz es útil cuando se guardan nuevos documentos, ya que estos se identifican por la ausencia de un id y, en caso de requerirlo, podrá obtenerse en el "*documentReference*" que se obtenga como respuesta a la subida del nuevo archivo.

IDAL

```
public interface IDAL {

    String Insert(Object entidad, InsertarCallback respuesta);

    void Delete(Object entidad);
    public void getAll(Object tipo, GetAllCallback respuesta);


    Object getById(String idGet, Object tipo, GetIdCallback respuesta);

    Object getByEmail(String email, Object tipo, GetIdCallback respuesta);
}
```

Figura 4.20 Interfaz IDAL

La interfaz IDAL, mostrada en la Figura 4.20, indicará los métodos que deberá contener finalmente la clase FirebaseDAL. Como se puede observar, todos los métodos, a excepción de *Delete* e *Insert*, piden como argumento un objeto nombrado "tipo". Esto se hace debido a que Firestore, en sus comunicaciones, pide el nombre de la colección a la cual queremos acceder. Como FirebaseDAL será una clase accedida por cualquier servicio y será la única encargada de comunicarse directamente con la base de datos, una forma sencilla de indicar a que colección se va a acceder es pasando como argumento una instancia del tipo de objeto con el que estamos trabajando. De esta forma, como se verá a continuación, podremos gestionar las colecciones de Firestore.

FirebaseDAL

```
public class FirebaseDAL implements IDAL {

    private String path = "";
    private String id = "";
    private final FirebaseFirestore databaseReference;

    public FirebaseDAL() {
        databaseReference = FirebaseFirestore.getInstance();
    }
```

Figura 4.21 Atributos y constructor de FirebaseDAL

Finalmente, la clase FirebaseDAL, mostrada en la Figura 4.21, como podemos observar, implementa la interfaz IDAL, que le forzará a usar todos los métodos de las interfaces que se presentaron anteriormente.

En referencia a los atributos,

- ***path***: contendrá, en caso de que sea necesario, la colección con la que querremos comunicarnos en base de datos.
- ***id***: contendrá, en caso de que sea necesario, el id del elemento con el que estamos trabajando.
- ***databaseReference***: contendrá la referencia a nuestra base de datos.

Como se mencionó anteriormente, es en esta clase en la que se gestionará a que colección se hará referencia, y esto es mediante el método *ComprobarTipoObjeto*, que lo podemos observar en la Figura 4.22.

```
117    private void ComprobarTipoObjeto(Object entidad) {
118        if (entidad instanceof Usuario) {
119            path = "usuarios";
120            id = ((Usuario) entidad).getId();
121        }else if(entidad instanceof Receta){
122            path = "recetas";
123            id = ((Receta) entidad).getId();
124        }else if(entidad instanceof Enfermedad){
125            path = "enfermedades";
126            id = ((Enfermedad) entidad).getId();
127        }else{
128            System.out.println("ERROR DE TIPO");
129        }
130    }
```

Figura 4.22 Método ComprobarTipoObjeto de FirebaseDAL

Este método, como se podrá observar en siguientes apartados, será invocado al comienzo del resto de métodos que traten de comunicarse con la base de datos, como puede ser *getById*() o *getAll*().

Almacenamiento

Si continuamos con la estructura de FirebaseDAL, encontramos el método que nos permite subir un dato a Cloud Firestore.

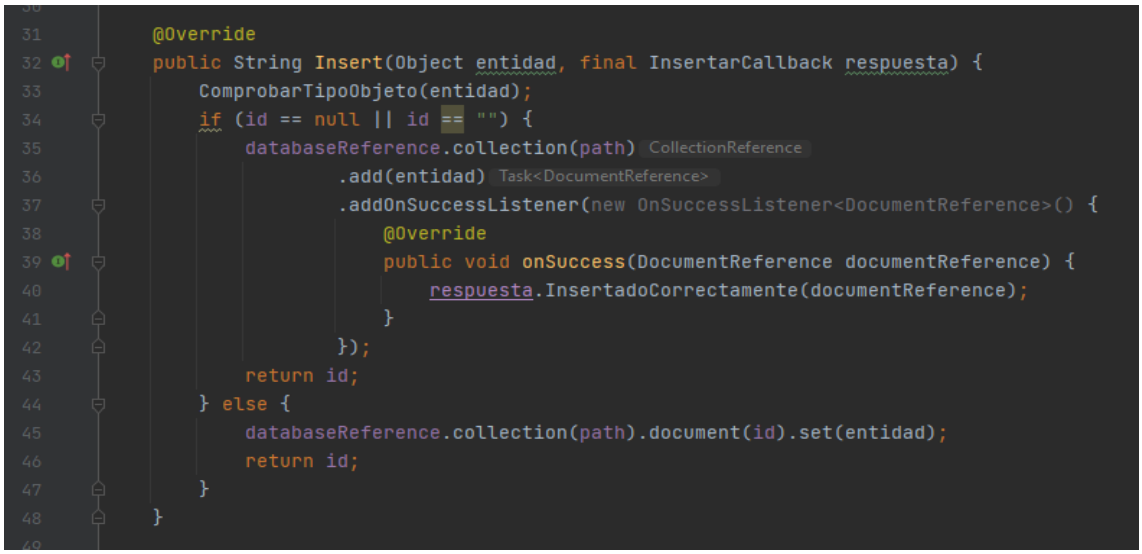


Figura 4.23 Método Insert de FirebaseDAL

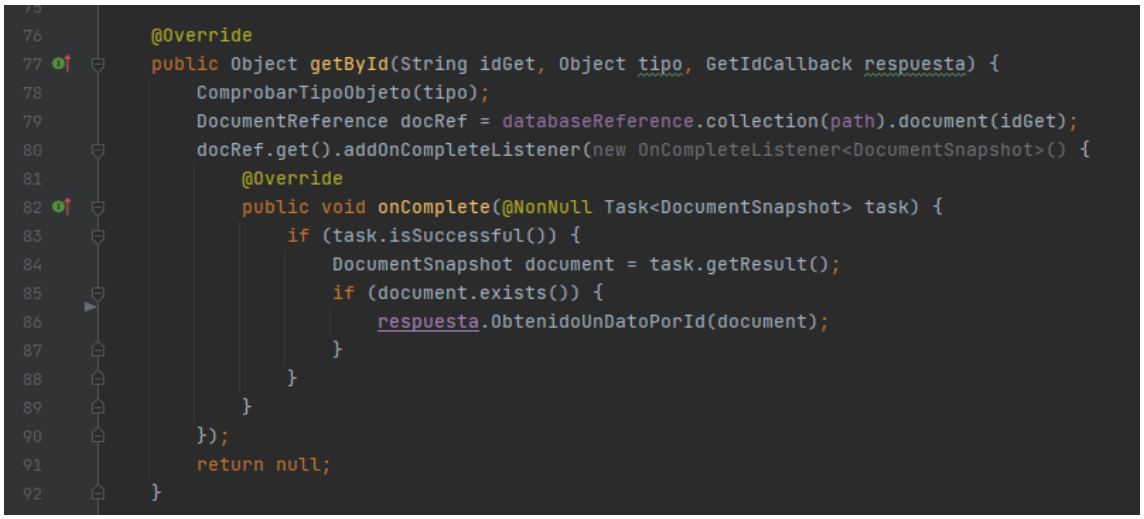
Como se puede observar en la Figura 4.23, en primer lugar invocamos al método *ComprobarTipoObjeto*, en el que se le dará un valor a la variable *path* (que hace referencia a la colección a la que subiremos el dato) y a id (en caso de que el objeto a insertar en base de datos ya tenga o no).

Tras la invocación a este método se hace una distinción en función de si “entidad” ya tiene o no un id, para diferenciar los casos de sobreescritura o inserción de nuevos objetos.

Como podemos observar, únicamente en los casos de ser un nuevo dato a insertar se invocará el método *callback*, ya que, si es una sobreescritura, se entiende que disponemos de todos los datos del objeto que estamos insertando, mientras que si es nuevo, podríamos estar interesados en obtener su id generado.

Descarga

Para la descarga de datos, a diferencia del almacenamiento, encontramos más variedad de casos posibles y, por tanto, un mayor número de métodos destinados a realizar diferentes llamadas.



```
@Override
public Object getById(String idGet, Object tipo, GetIdCallback respuesta) {
    ComprobarTipoObjeto(tipo);
    DocumentReference docRef = databaseReference.collection(path).document(idGet);
    docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()) {
                DocumentSnapshot document = task.getResult();
                if (document.exists()) {
                    respuesta.ObtenidoUnDatoPorId(document);
                }
            }
        }
    });
    return null;
}
```

Figura 4.24 Método getById de FirebaseDAL

En este método, mostrado en la Figura 4.24, encontramos que, en primer lugar, se invoca el método *ComprobarTipoObjeto*, esto nos proporcionará el *path* (colección) a la que queremos acceder y, al recibir la respuesta, retornamos el documento obtenido mediante el *callback* cuyo método será el creado en la interfaz *GetIdCallback*, mostrada ya anteriormente.

De similar manera se estructura el método getAll, mostrado en la Figura 4.25.

```
@Override
public void getAll(Object tipo, GetAllCallback respuesta) {
    ComprobarTipoObjeto(tipo);
    databaseReference.collection(path) CollectionReference
            .get() Task<QuerySnapshot>
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        List<QueryDocumentSnapshot> Objects = new ArrayList();
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Objects.add(document);
                        }
                        respuesta.ObtenidosTodosLosDatos(Objects);
                    }
                }
            });
}
```

Figura 4.25 Método getAll de FirebaseDAL

La única diferencia es que, en este método, usaremos la interfaz *GetAllCallback* para poder gestionar un número mayor de documentos recibidos en la respuesta a la llamada.

4.5. Pruebas

Durante las etapas iniciales del desarrollo de una aplicación, es crucial enfocarse en la verificación y validación constante del producto.

Para verificar el producto, se llevan a cabo pruebas exhaustivas para identificar y corregir posibles errores, bugs o fallas en la funcionalidad. Esto implica realizar pruebas unitarias y pruebas de aceptación para asegurarse de que el producto cumpla con los estándares de calidad y funcione correctamente.

Las pruebas de aceptación se realizaron de manera manual y exploratoria durante el propio desarrollo de cada tarea individual donde, como se explicó anteriormente, en las descripciones de las tareas de Trello se establecían ciertas pautas que debían seguir las tareas.

En este apartado nos centraremos en las pruebas unitarias, las cuales se han realizado mediante JUnit.

Su uso no requiere ninguna instalación específica si se está utilizando Java, pues se trata de una librería para la misma que suele venir incorporada en la mayoría de IDEs, como es nuestro caso, Android Studio.

A pesar de encontrarse ya por defecto, debemos mirar si JUnit se encuentra habilitado en Android Studio.

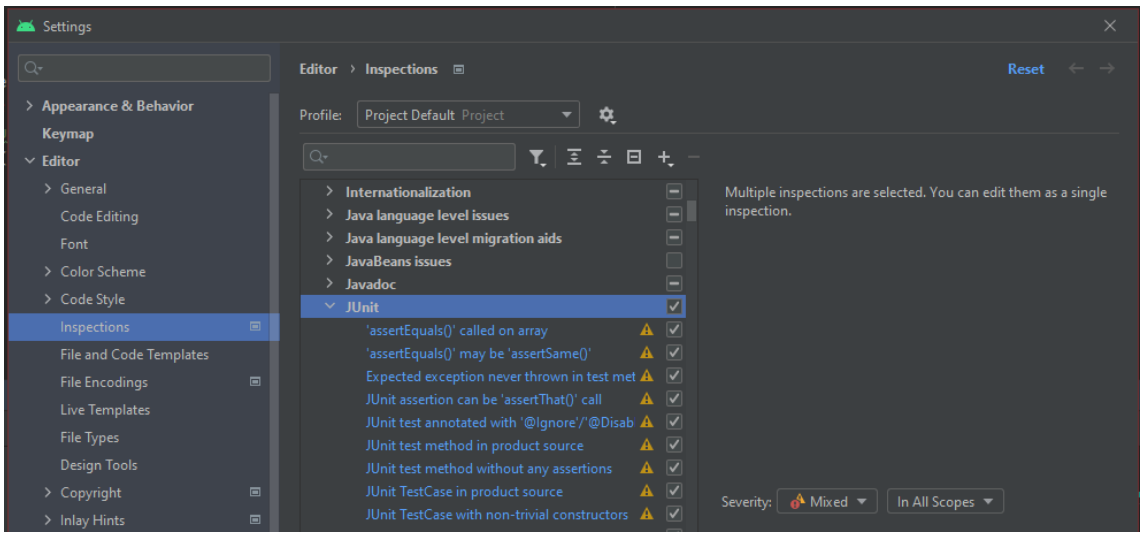


Figura 4.26 Activar JUnit en Android Studio

En nuestro caso venía desactivado, pero seleccionando JUnit y salvando los cambios, como se muestra en la Figura 4.26, podríamos comenzar la realización de pruebas unitarias en nuestro proyecto.

Los test automatizados mediante JUnit se realizaron en las partes de la aplicación que consideramos más críticas y, a nuestro parecer, debíamos asegurar que siempre funcionasen correctamente.

Entre los cuales encontramos:

- Inicio de sesión
- Registro de usuario

- Edición de datos del usuario
- Cálculos correctos de la calculadora Harrys Benedict

Y como se puede observar en la Figura 4.27, todos los test pasan satisfactoriamente.

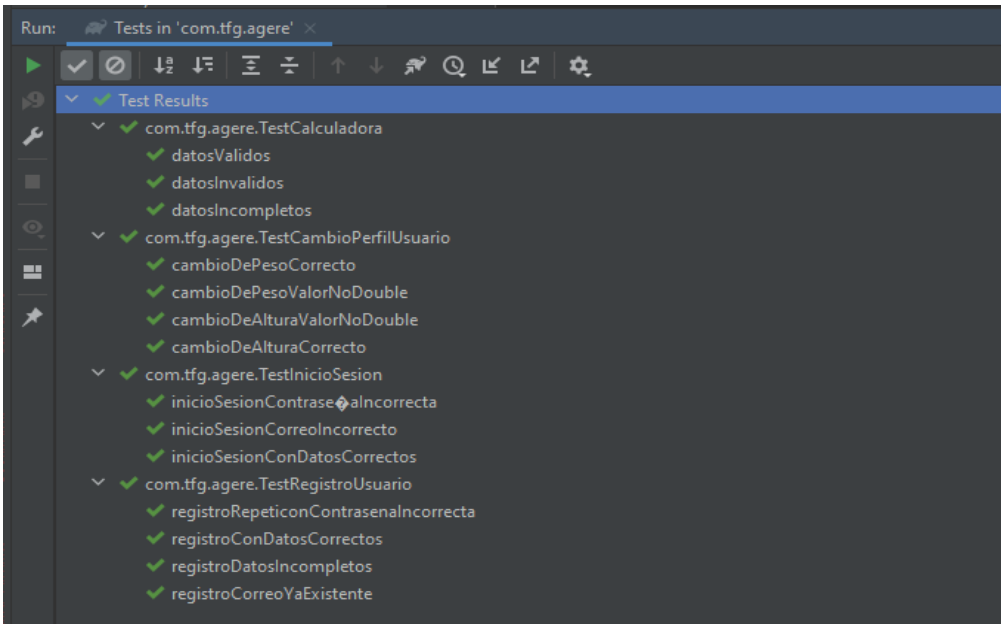


Figura 4.27 Resultado de los test realizados en JUnit

Como ejemplo, se va a explicar paso a paso como realizamos uno de los test, en concreto, comprobar el correcto funcionamiento de la edición de datos en el perfil del usuario, y para simplificar la explicación, nos centraremos en la funcionalidad para cambiar el peso y altura del usuario.

En primer lugar, debe identificarse la sección de código crítica que va a ser testeada.

```java
public static void actualizaAltura(String altura, DatosBD datosBD, UsuarioServicios usuarioServicios){
    if(!altura.equals("") && altura.charAt(altura.length() - 1) != '.'){
        try{
            double val = Double.parseDouble(altura);
            datosBD.getUsuario().setAltura(val);
            usuarioServicios.AddUsuario(datosBD.getUsuario(), new InsertarCallback() {
                @Override
                public void InsertadoCorrectamente(DocumentReference documentReference) {

                }
            });
        }catch (NumberFormatException e){
            System.out.println("LA ALTURA NO CORRESPONDE A UN DOUBLE");
        }
    }
}

public static  void actualizaPeso(String peso, DatosBD datosBD, UsuarioServicios usuarioServicios){
    if(!peso.equals("") && peso.charAt(peso.length() - 1) != '.'){
        try{
            double val = Double.parseDouble(peso);
            datosBD.getUsuario().setPeso(val);
            usuarioServicios.AddUsuario(datosBD.getUsuario(), new InsertarCallback() {
                @Override
                public void InsertadoCorrectamente(DocumentReference documentReference) {

                }
            });

        } catch (NumberFormatException e){
            System.out.println("EL PESO NO CORRESPONDE A UN DOUBLE");
        }
    }
}
```

Figura 4.28 Métodos a testear con JUnit

En la Figura 4.28 se muestran los métodos encargados de realizar el cambio de altura y peso del usuario respectivamente.

Como podemos observar, ambos métodos realizan exactamente el mismo funcionamiento, a diferencia de que uno realiza cambios en los atributos correspondientes del usuario.

Veamos como funciona el método detenidamente:

- En primer lugar se realiza una comprobación para ver si el valor introducido no es un vacío o no termina en punto (esto java lo detecta como un error tipográfico y no lo sabe castear a *double*.
- Si el valor introducido es posible castearlo a *double*, se guarda en una variable llamada “val” para, a continuación, modificar los datos del usuario actual.
- Tras realizar todos los cambios pertinentes en pantalla, se procede a salvar el usuario en *Firestore*.

Durante todo este proceso, en la línea donde se castea el valor introducido pueden surgir errores, debido a datos inválidos, es por esto que el código se encuentra encapsulado por un try-catch encargado de capturar este error y no proceder con el cambio erróneo de datos.

Ahora, con el código ya analizado, procederemos a presentar la clase test donde se realizarán las pruebas.

Al realizar el primer intento de los test surgieron los errores mostrados en la Figura 4.29.

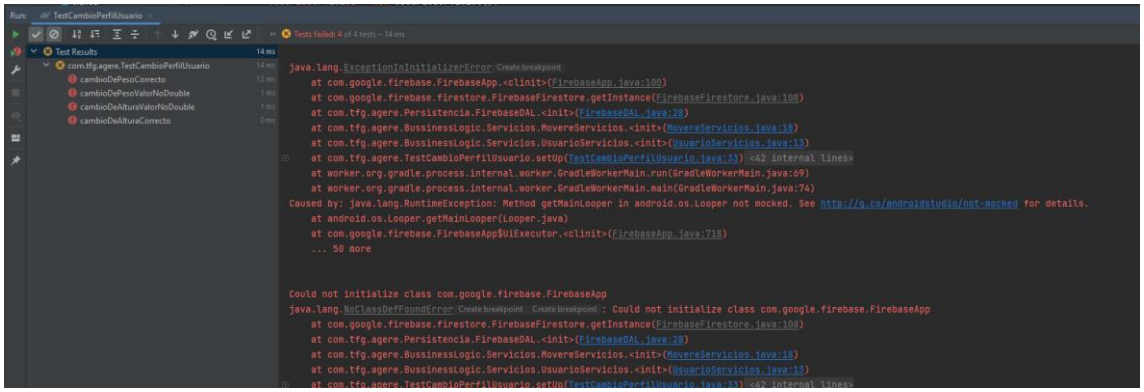


Figura 4.29 Error en la ejecución de test

Esto fue debido a que, para lanzar los test de JUnit, la aplicación no es ejecutada en ningún momento, únicamente se prueban los métodos y líneas de código indicados, por lo que en ningún momento iniciaba sesión ningún usuario. Además, realizar una instancia de nuestro *Firestore Database* no resolvería nuestro problema, puesto que la base de datos, por las reglas que especificamos en un inicio, únicamente se puede acceder si uno tiene una sesión iniciada en ese instante y, modificar estas reglas para poder acceder a la base de datos fuera de un uso normal de la aplicación supondría una importante brecha de seguridad. Por esta razón decidimos realizar ciertas modificaciones en los métodos a testear, sin modificar el funcionamiento de los mismos.

Así pues, tras las modificaciones pertinentes, el código resultó como se muestra en la Figura 4.30.

```
public static boolean actualizaAltura(String altura, DatosBD datosBD){
    if(!altura.equals("") && altura.charAt(altura.length() - 1) != '.'){
        try{
            double val = Double.parseDouble(altura);
            datosBD.getUsuario().setAltura(val);
            return true;
        }catch (NumberFormatException e){
            System.out.println("LA ALTURA NO CORRESPONDE A UN DOUBLE");
            return false;
        }
    }else{
        return false;
    }
}

public static  boolean actualizaPeso(String peso, DatosBD datosBD){
    if(!peso.equals("") && peso.charAt(peso.length() - 1) != '.'){
        try{
            double val = Double.parseDouble(peso);
            datosBD.getUsuario().setPeso(val);
            return true;
        } catch (NumberFormatException e){
            System.out.println("EL PESO NO CORRESPONDE A UN DOUBLE");
            return false;
        }
    }else{
        return false;
    }
}
```

Figura 4.30 Métodos a testear modificados

De tal forma que ahora los métodos no realizan ninguna llamada a base de datos y, en cambio, devuelve un booleano que nos indicará si se ha podido realizar o no

correctamente el cambio de datos. En caso de ser afirmativo, se realizará la actualización en base de datos y, en caso negativo, no se realizará ningún tipo de actualización.

En nuestro caso de la clase de test quedará como se observa en la Figura 4.31.

Figura 4.31 Método Before JUnit

En primer lugar inicializamos los datos que son necesarios para realizar las pruebas. En nuestro caso, inicializaremos un usuario e inicializaremos una instancia de DatosBD, donde guardaremos el usuario recién creado.

Con la etiqueta *@Before* colocada justo antes del método *setUp()*, indicaremos a JUnit que se método deberá realizarse antes que cualquier test.

```java
@Test
public void cambioDeAlturaCorrecto() {
    double nuevaAltura = usuario.getAltura()+5;
    Ajustes.actualizaAltura( altura: ""+nuevaAltura, datosBD);
    double alturaTrasActualizacion = datosBD.getUsuario().getAltura();

    assertThat(nuevaAltura, is(alturaTrasActualizacion));
}

@Test
public void cambioDeAlturaValorNoDouble() {
    String nuevaAltura = "170.";
    double alturaPreviaActualizacion = datosBD.getUsuario().getAltura();
    Ajustes.actualizaAltura(nuevaAltura, datosBD);
    double alturaTrasActualizacion = datosBD.getUsuario().getAltura();

    assertThat(alturaPreviaActualizacion, is(alturaTrasActualizacion));
}

@Test
public void cambioDePesoCorrecto() {
    double nuevoPeso = usuario.getPeso()+5;
    Ajustes.actualizaPeso( peso: ""+nuevoPeso, datosBD);
    double pesoTrasActualizacion = datosBD.getUsuario().getPeso();

    assertThat(nuevoPeso, is(pesoTrasActualizacion));
}

@Test
public void cambioDePesoValorNoDouble() {
    String nuevoPeso = "80.";
    double pesoPrevioActualizacion = datosBD.getUsuario().getPeso();
    Ajustes.actualizaPeso(nuevoPeso, datosBD);
    double pesoTrasActualizacion = datosBD.getUsuario().getPeso();

    assertThat(pesoPrevioActualizacion, is(pesoTrasActualizacion));
}
```

Figura 4.32 Métodos test JUnit

Así pues, en la Figura 4.32 encontramos los test realizados, donde:

- CambioAlturaCorrecto: en este método pasaremos un dato de altura correcto, en el cual, tras la llamada del método actualizarAltura(), lo que esperaremos es que el usuario que se encuentra en datosBD contenga el mismo peso que el valor del peso nuevo introducido. Esto se indica mediante el método ***assertThat(****objectX,* ***is(****objectY****))***, donde, en lenguaje natural, estamos diciendo que “esperamos que objetoX sea igual a objetoY”.

- CambioAlturaValorNoDouble: en este método, el proceso a realizar es el mismo pero, en este caso, enviaremos al método actualizarAltura() un valor erróneo, en concreto, un valor no parseable[22] a double. La comprobación a realizar para saber si el método realiza bien su tarea será esperar a que el peso que se tenía antes del intento del cambio sea el mismo que el obtenido tras este. Esto se realiza mediante el método ***assertThat(****objectX,* ***is(****objectY****))***.

[22] Parseable: cualidad de un valor cuando este se puede interpretar y convertir en un formato específico.

CambioPesoCorrecto y CambioPesoValorNoDouble siguen la misma estructura que los dos métodos explicados respectivamente.

Así pues, podemos disponernos a ejecutar los test viendo finalmente que todos se realizan satisfactoriamente, como se puede observar en la Figura 4.33.

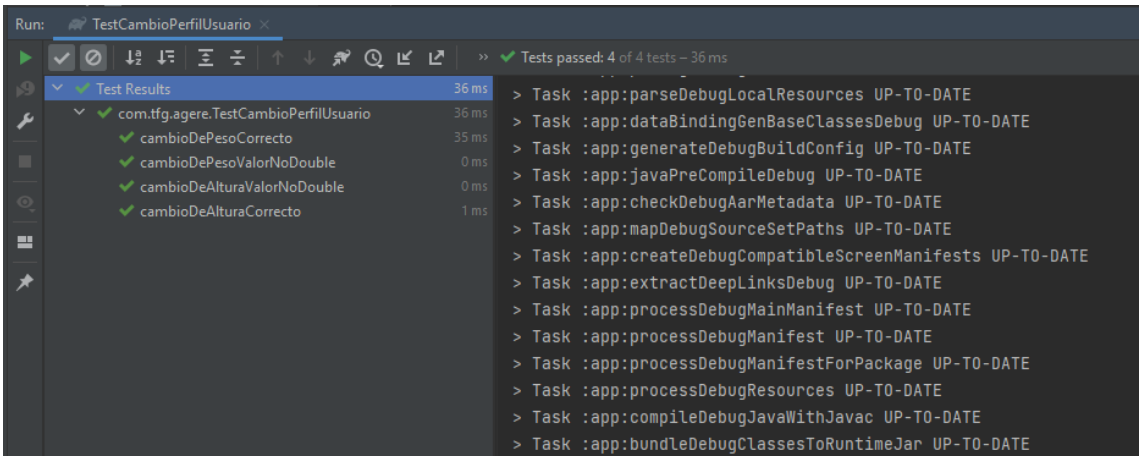


Figura 4.33 Resultado de ejecución test JUnit en Android Studio

4.6. Guía de uso

Si somos nuevos usuarios el primer paso en la aplicación será registrarse.

Figura 4.34 Pantalla registro de Agere

Como observamos en la Figura 4.34, registro únicamente necesita un correo y una contraseña. Si el correo introducido no se encuentra registrado en nuestra app, este procederá a registrarse como nuevo usuario y se le redirigirá a un formulario de inicio como el mostrado en la Figura 4.35.

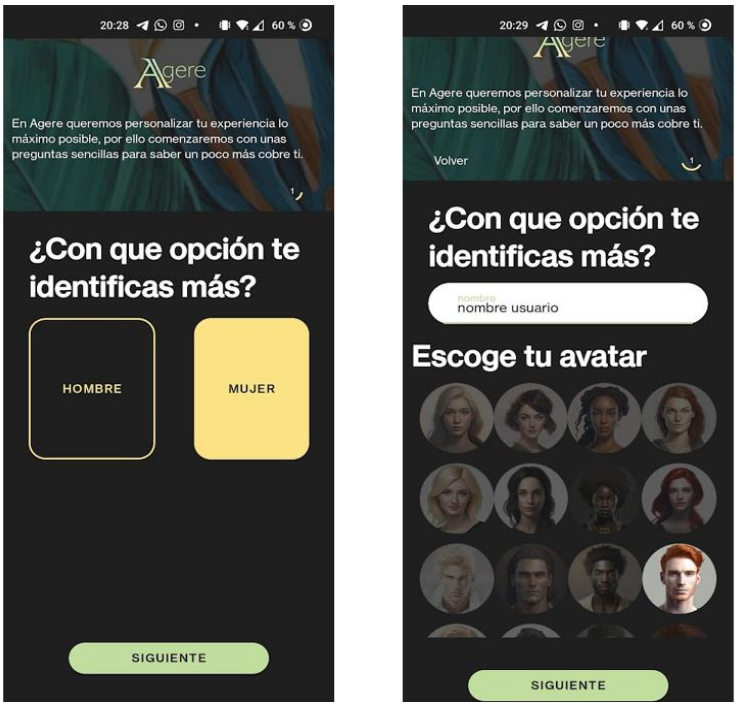


Figura 4.35 Dos primeras pantallas del formulario inicial de Agere

Entre otros, se le pedirá al usuario que introduzca algunos datos básicos para el correcto cálculo de algunos parámetros necesarios dentro de la aplicación, como es el sexo del usuario, fecha de nacimiento y enfermedades que padece, pero también se le pedirán otros datos, para mejorar la experiencia del usuario, como es el nombre y avatar del usuario.

Tras haber completado correctamente el formulario, el usuario tendrá pleno acceso a la aplicación, donde se le redirigirá por primera vez al *Home* de la aplicación, mostrado en la Figura 4.36.

Figura 4.36 Pantalla home de Agere

Desde el *Home* de la aplicación tendremos acceso a:

- Ajustes de usuario → seleccionando la imagen de perfil
- Recomendaciones básicas de salud → seleccionando la tarjeta “Empecemos por lo sencillo”
- Calcular las kcal recomendadas a ingerir cada día → seleccionando la tarjeta “Calcula tu dieta”
- Obtener información básica sobre enfermedades que padezca el usuario → seleccionando “¿Qué cuidados debo llevar?

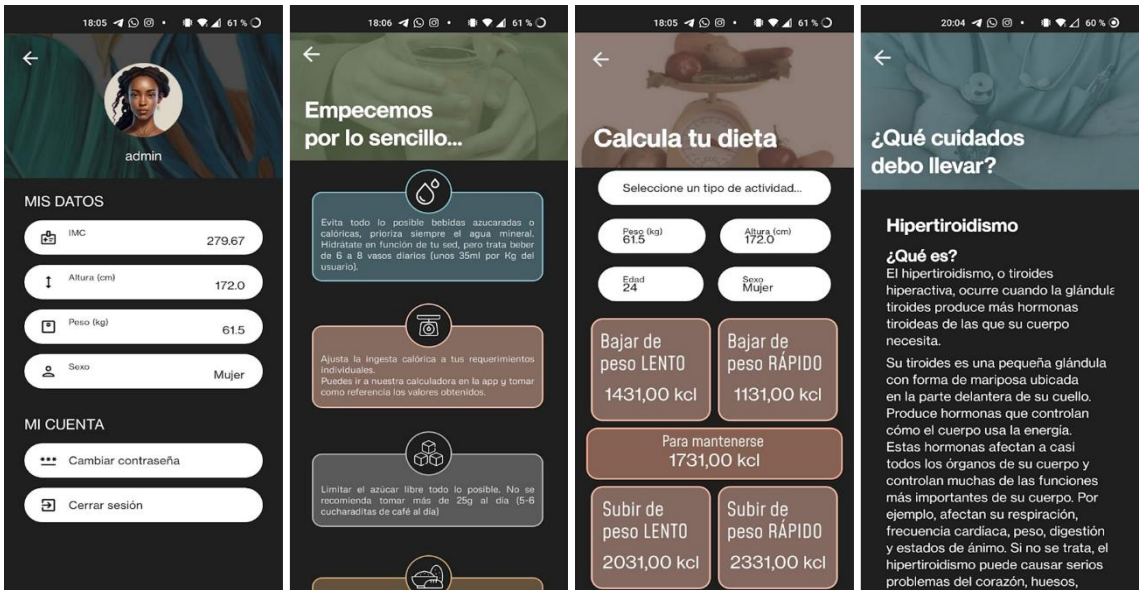


Figura 4.37 Pantallas: Ajustes de usuario - Consejos básicos de salud - Calculadora Harrys Benedict - Información enfermedades

En ajustes de usuario se podrán modificar los datos del usuario así como cerrar sesión.

En consejos básicos de salud se muestran una serie de consejos dados por nuestros especialistas que, según estos, toda persona debería tratar de aplicar en su día a día.

La calculadora de Harrys Benedict no solo te permitirá ver las kcal a consumir según tus propios parámetros, también permitirá calcular las kcal para cualquier usuario en cualquier momento del día.

La sección de información sobre enfermedades proporciona detalles básicos sobre lo que es una enfermedad en particular, las posibles causas por las que se padece y las recomendaciones generales para evitar o manejar la enfermedad seleccionada por el usuario.

Si nos desplazamos al menú *Menú*, podremos tener una vista rápida del menú recomendado por la aplicación en función de la enfermedad que padezca el usuario, mientras que en el menú Recetas, encontraremos un listado con las recetas disponibles para nosotros. Estas ventanas las podemos observar en la Figura 4.38.

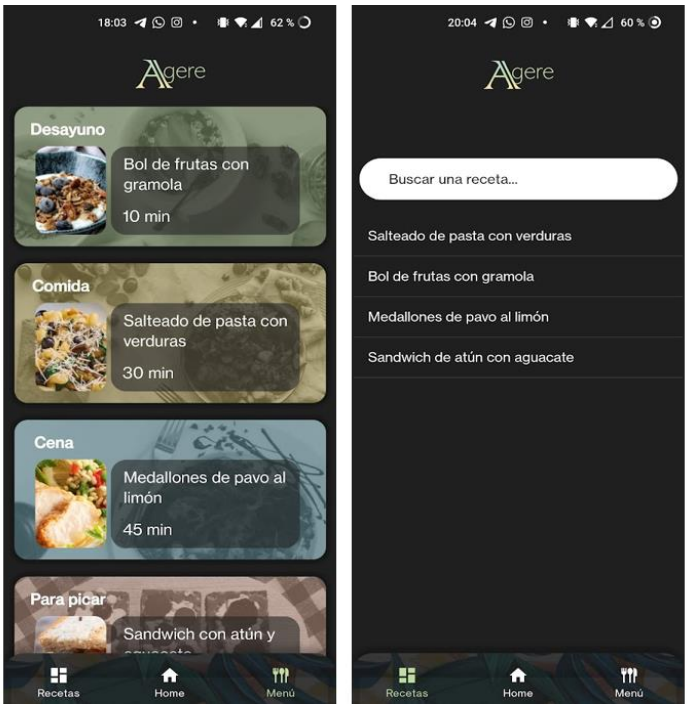


Figura 4.38 Pantallas Menú y Recetas de Agere

En ambas pantallas, si seleccionamos alguna de las recetas, podremos ver el detalle de la receta, es decir, el tiempo, ingredientes y pasos para llevar a cabo la receta, como se muestra en la Figura 4.39.

Figura 4.39 Pantalla detalle de receta

4.7. Experimentos

Un MVP (*Minimum Viable Product* o Mínimo Producto Viable) es una versión lo suficientemente básica de un producto o servicio que se desarrolla y lanza al mercado con el objetivo de obtener retroalimentación de los clientes de manera temprana y validar la viabilidad del concepto antes de invertir tiempo y recursos en su desarrollo completo.

4.7.1. Primer experimento

Después de completar la primera versión básica de Agere, el siguiente paso consistió en preparar el primer experimento con usuarios reales con el fin de obtener retroalimentación inicial de los mismos. Para ello, se seleccionó cuidadosamente un grupo reducido pero representativo de personas que abarcara todos los perfiles de *early adopters*[23]

La elección de este grupo se basó en la importancia de contar con usuarios que estuvieran dispuestos a probar nuevas tecnologías y que tuvieran la capacidad de brindar comentarios y sugerencias valiosas.

Se buscó incluir diferentes perfiles que representaran a diversos segmentos del público objetivo, con el objetivo de obtener una perspectiva más amplia y completa. Los perfiles seleccionados fueron los siguientes:

Usuario 1	Hombre. Especialista sanitario.
Edad	25
Estado de salud	Bueno
Conocimientos	Altos conocimientos en salud y nutrición

Usuario 2	Hombre. Estudiante informática.
Edad	22
Estado de salud	Padece asma
Conocimientos	Conocimientos básicos de nutrición

Usuario 3	Mujer. Trabajo semi-sedentario.
Edad	56
Estado de salud	Padece hipertensión y tiroides
Conocimientos	Escasos conocimientos de nutrición ni salud

Usuario 4	Hombre. Trabajo activo.
Edad	59
Estado de salud	Bueno
Conocimientos	Conocimientos básicos de salud

Usuario 5	Hombre. Jubilado.
Edad	65
Estado de salud	Bueno
Conocimientos	Conocimientos básicos de nutrición

[23] Early adopters: clientes que, además de usar el producto o la tecnología del proveedor, también brindan comentarios sinceros y considerables para ayudar al proveedor a perfeccionar sus futuros lanzamientos.

Estos usuarios fueron invitados a utilizar Agere y a compartir sus experiencias, destacando los aspectos positivos, identificando posibles problemas o áreas de mejora, y sugiriendo nuevas funcionalidades o características que consideraran relevantes.

Para esto, se entregaron una serie de preguntas a los usuarios en donde pudiesen expresar sus ideas, además de realizar una pequeña encuesta para saber si el funcionamiento inicial de la aplicación era correcto y si estaba siendo desarrollada de forma correcta.

Encuesta

La siguiente encuesta se entregó tras el uso de la aplicación por parte de los usuarios. En la primera parte se les realizó una serie de preguntas en referencia a la facilidad de uso de Agere, donde 1-totalmente en desacuerdo y 5-totalmente de acuerdo. Tras esto, se les concedió un espacio donde escribir libremente y poder dar su opinión, sugerencias y posibles mejoras o fallos detectados.

Responda a las preguntas siguientes según su conformidad, teniendo en cuenta que 1 es “totalmente en desacuerdo” y 5 es “totalmente de acuerdo”

	1	2	3	4	5
¿La aplicación es intuitiva?					
¿Te ha parecido atractiva la interfaz de la aplicación?					
¿Te ha parecido útil la información mostrada en la aplicación?					
¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?					
¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?					

Tras conocer el objetivo de la aplicación y haber tenido un primer acceso a ella, ¿Qué te gustaría ver en futuras versiones y consideras imprescindible? ¿Consideras que algún aspecto de la aplicación debería mejorar?

Respuestas

¿La aplicación es intuitiva?

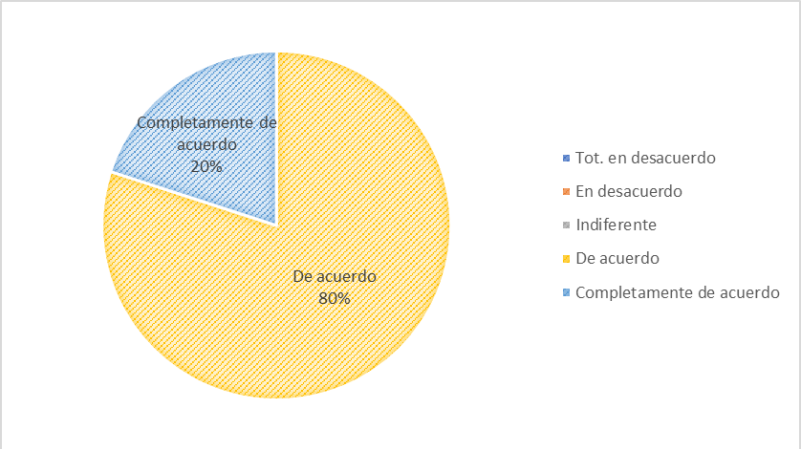


Figura 4.40 ¿La aplicación es intuitiva?

Como podemos observar en la Figura 4.40, en general todos concordaban en que la aplicación era bastante intuitiva, a pesar de que un 80% consideraban que podían haber algunos aspectos a mejorar, como algunos apartados de navegación. Esto se debió en parte porque parte de los entrevistados eran adultos que no estaban experimentados con el dispositivo de prueba que se les dio. Resultó que este dispositivo móvil era controlado por gestos en la pantalla y no por una botonera digital incorporada en la pantalla.

Es por esto que sufrieron un periodo de adaptación hasta que lograron adaptarse a la navegación en dicho dispositivo.

¿Te ha parecido atractiva la interfaz de la aplicación?

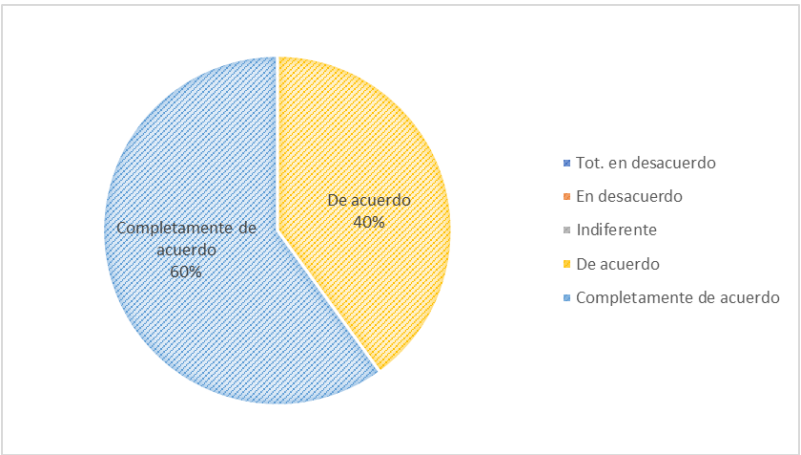


Figura 4.41 ¿Te ha parecido atractiva la interfaz de la aplicación?

A pesar de que el diseño puede ser uno de los aspectos de una aplicación más subjetivos debido al amplio abanico de gustos que pueden tener las personas, se puede observar en la Figura 4.41 una gran aceptación de la interfaz diseñada. El minimalismo y una selección primaria de colores hace que, a más o a menos, la interfaz en rasgos generales sea atractiva.

¿Te ha parecido útil la información mostrada en la aplicación?

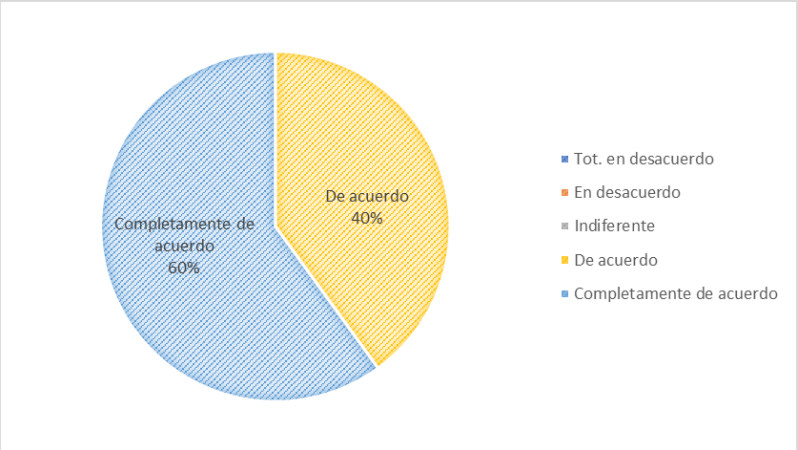


Figura 4.42 ¿Te ha parecido útil la información mostrada en la aplicación?

En este caso, se deben interpretar correctamente los resultados, como se puede observar en la Figura 4.42, ha agradado altamente la información mostrada, pero en algunos casos no les ha parecido extremadamente llamativa. ¿Esto a que se puede deber? Pues, analizando las respuestas individualmente, se pudo observar que los sujetos con mayores conocimientos de la salud y nutricionales les daban una menor importancia que los sujetos que en un principio no poseían dichos conocimientos.

¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?

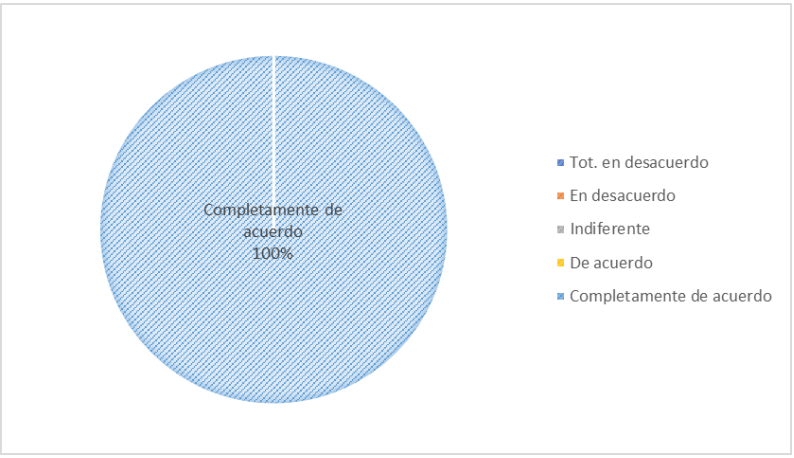


Figura 4.43 ¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?

En esta pregunta, mostrada en la Figura 4.43 la respuesta resultó ser unánime, si no resultaba ser por una ventana de la aplicación, era otra, pero todas mostraban información útil y relevante a los usuarios, que, si no tenían conocimientos previos, les resultaba de gran curiosidad, y para los más experimentados, encontraban algún dato que creían olvidado.

¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?

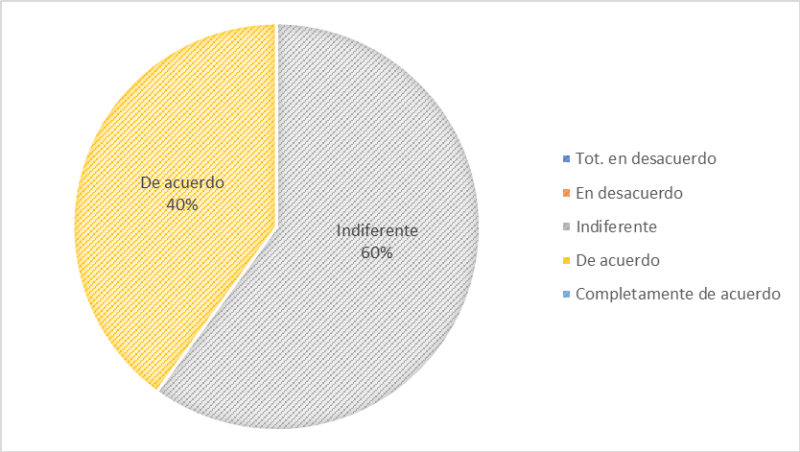


Figura 4.44 ¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?

Esta se trata de la pregunta peor valorada como se puede observar en la Figura 4.44, pero se debe tener en cuenta que la aplicación se encuentra en su primera etapa, por lo que quedan muchas funcionalidades por realizar y queda mucho margen de mejora. Teniendo esto en cuenta, los resultados obtenidos no son tan malos, ya que, a pesar de que sea una aplicación en pleno desarrollo, ha agradado a los usuarios y tienen confianza en todo lo que puede aportar.

Conclusiones primer MVP

De los resultados obtenidos se pueden sacar varios puntos a tener en cuanta:

- A los usuarios les gusta la aplicación, le ven potencial al producto final que se podrá alcanzar cuando Agere se encuentre finalizado y listo para lanzarse al mercado.
- Algunos aspectos son más difíciles de considerar, como el caso del coste de la aplicación premium, donde es difícil saber su opinión sin un producto más desarrollado que el actual.

Es por esto que para el siguiente MVP se considerará dar cierta prioridad a las sugerencias de los *early adpters*, para así poder entregarles un producto más completo y dirigido a las necesidades de estos.

4.7.2. Segundo experimento

Tras la implementación, ampliación y testeo de las nuevas funcionalidades en Agere se dio comienzo al segundo experimento con usuarios. Las pautas a seguir fueron las mismas que en el anterior, con dos ligeras modificaciones:

- Se permitió instalar la aplicación en los dispositivos de los usuarios si estos querían, en vez de usar un único dispositivo de pruebas.
- Se aumentó el número de usuarios a probar la aplicación. Se volvieron a escoger los 5 mismos usuarios que testearon la aplicación en el primer experimento, para

que así estos pudiesen puntuar la evolución de la aplicación y se escogieron 10 personas más a las que mostrar la aplicación por primera vez.

Los 10 nuevos usuarios tienen los siguientes perfiles:

Usuario 6	Mujer. Trabajo sedentario.
Edad	56
Estado de salud	Padece artritis
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 7	Hombre. Especialista informático
Edad	56
Estado de salud	Padece arritmias
Conocimientos	Conocimientos básicos de nutrición. Escasos conocimientos de salud

Usuario 8	Mujer. Jubilada.
Edad	60
Estado de salud	Padece fibromialgia
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 9	Hombre. Trabajo sedentario.
Edad	38
Estado de salud	Bueno
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 10	Mujer. Especialista sanitaria.
Edad	26
Estado de salud	Bueno
Conocimientos	Altos conocimientos en salud y nutrición

Usuario 11	Mujer. Trabajo semi-sedentario.
Edad	37
Estado de salud	Padece ovarios poliquísticos.
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 12	Mujer. Jubilada.
Edad	66
Estado de salud	Bueno
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 13	Mujer. Especialista sanitaria.
Edad	29
Estado de salud	Padece ovarios poliquísticos.
Conocimientos	Altos conocimientos de salud. Conocimientos básicos de nutrición.

Usuario 14	Hombre. Trabajo sedentario.
Edad	30
Estado de salud	Bueno
Conocimientos	Escasos conocimientos de nutrición y salud

Usuario 15	Mujer. Trabajo sedentario.
Edad	60
Estado de salud	Bueno
Conocimientos	Escasos conocimientos de nutrición y salud

Encuesta

La encuesta que se realizó fue la misma para los nuevos usuarios, pero para los 5 usuarios que pudieron participar en el primer experimento, se les añadió una pregunta más, obteniendo pues el siguiente documento:

Responda a las preguntas siguientes según su conformidad, teniendo en cuenta que 1 es “totalmente en desacuerdo” y 5 es “totalmente de acuerdo”

	1	2	3	4	5
¿La aplicación es intuitiva?					
¿Te ha parecido atractiva la interfaz de la aplicación?					
¿Te ha parecido útil la información mostrada en la aplicación?					
¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?					
¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?					
Si ya habías usado Agere anteriormente, ¿Consideras que ha evolucionado en la dirección que esperabas?					

Tras conocer el objetivo de la aplicación y haber tenido un acceso temprano a ella, ¿Qué te gustaría ver en futuras versiones y consideras imprescindible? ¿Consideras que algún aspecto de la aplicación debería mejorar?

Respuestas

¿La aplicación es intuitiva?

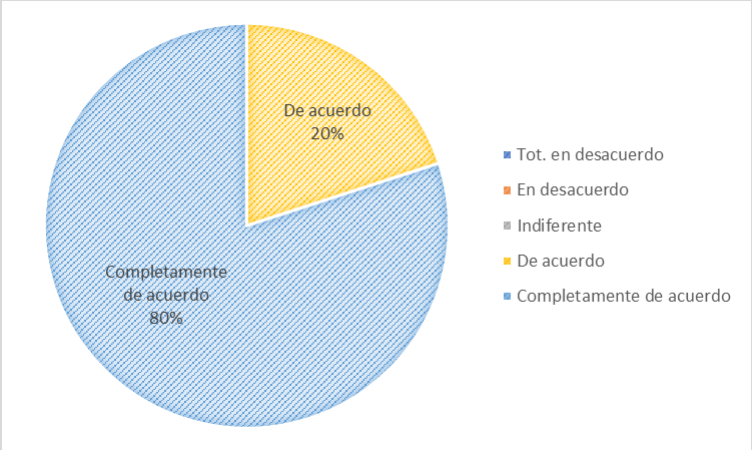


Figura 4.45 ¿La aplicación es intuitiva?

Respecto al primer MVP, la intuitividad de la aplicación se puede observar en la Figura 4.45 que ha mejorado considerablemente. Esto ha podido darse gracias a dos factores:

- Un porcentaje de los usuarios ya habían sido expuestos al sistema y al dispositivo, por lo que volver a usarlo, a pesar de las nuevas funcionalidades no les ha supuesto ningún esfuerzo.
- Al permitir a los usuarios probar Agere con sus propios dispositivos móviles, estos ya estaban familiarizados con la navegación, por lo que no supuso esfuerzo o tiempo extra para adaptarse, como sucedió en el primer experimento.

¿Te ha parecido atractiva la interfaz de la aplicación?

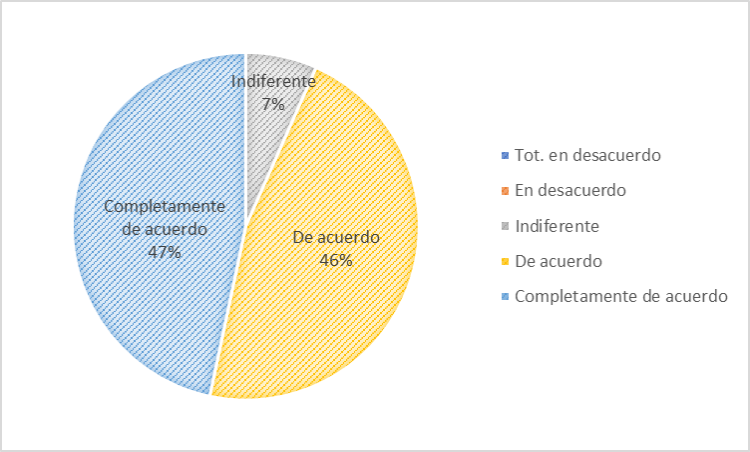


Figura 4.46 ¿Te ha parecido atractiva la interfaz de la aplicación?

Al haber un mayor numero de usuarios, las opiniones comienzan a ser más dispares. Al igual que en el primer experimento, la interfaz resulta ser algo más subjetivo. A pesar de que a gran parte de los usuarios concuerdan en que la aplicación tiene una interfaz agradable, comienzan a haber algunas discrepancias entre algunos usuarios, como podemos bien observar en la Figura 4.46.

¿Te ha parecido útil la información mostrada en la aplicación?

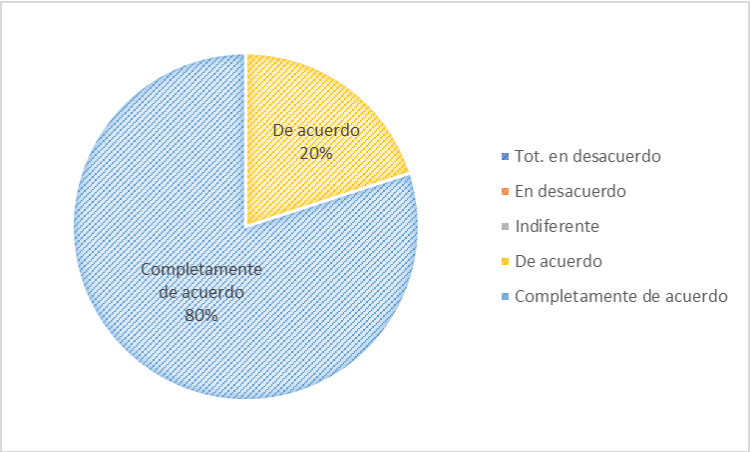


Figura 4.47 ¿Te ha parecido útil la información mostrada en la aplicación?

En general, como se observa en la Figura 4.47, a todos los usuarios, al igual que en el primer experimento, les gusta la información que ofrece la aplicación. Resulta de interés general y permite asentar ciertos conocimientos básicos que toda persona debería tener para comenzar a cuidar su salud.

¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?

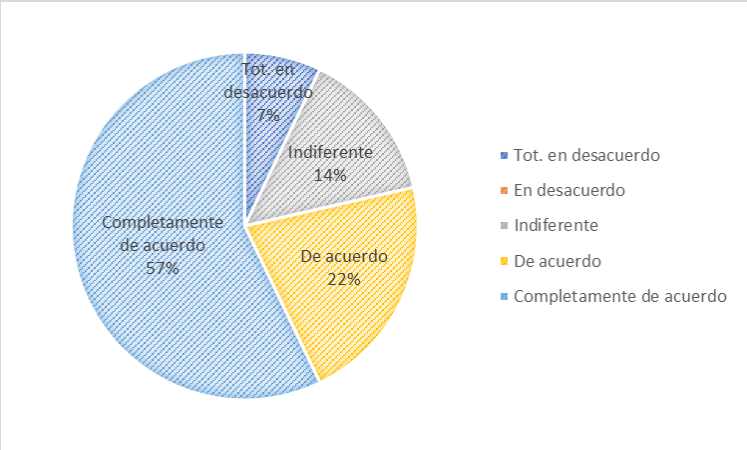


Figura 4.48 ¿Has obtenido nuevos conocimientos de nutrición que previamente no conocías?

En esta pregunta, mostrada en la Figura 4.48, al observar las puntuaciones tan dispares se tuvo que realizar un estudio por separado de cada usuario. Las conclusiones obtenidas fueron las siguientes:

- Las puntuaciones más bajas fueron dadas por usuarios que ya tenían altos conocimientos en nutrición y/o salud.
- Los siguientes usuarios que aportaron puntuaciones bajas fueron usuarios que previamente ya habían probado la aplicación.
- La gran mayoría de los nuevos usuarios fueron los que aportaron las puntuaciones más altas.

¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?

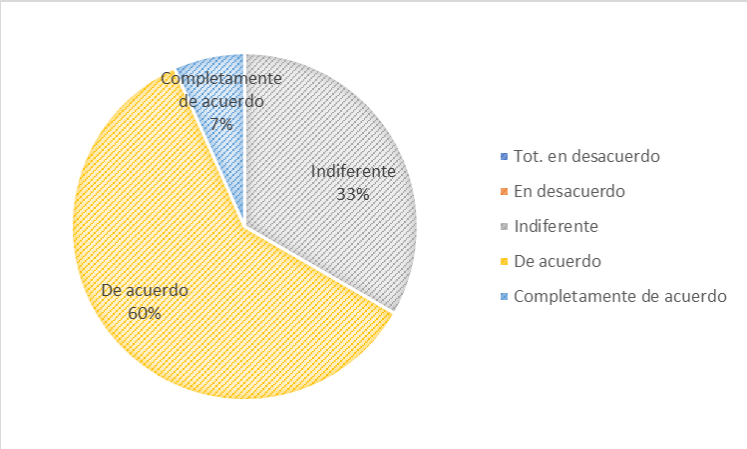


Figura 4.49 ¿Considerarías adecuados 5€ al mes para un plan premium de la aplicación?

A pesar de que las puntuaciones siguen siendo para esta pregunta algo bajas, como podemos observar en la Figura 4.49, si que han mejorado considerablemente desde el primer experimento. Esto es algo a tener en cuenta, ya que la aplicación sigue sin disponer de todas las funcionalidades a las que aspira, pero aún así, los usuarios confían cada vez más en esta.

Si ya habías usado Agere anteriormente, ¿Consideras que ha evolucionado en la dirección que esperabas?

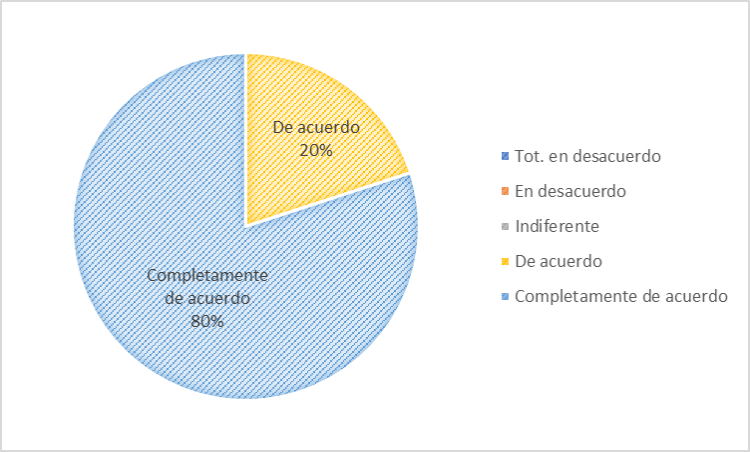


Figura 4.50 Si ya habías usado Agere anteriormente, ¿Consideras que ha evolucionado en la dirección que esperabas?

Gracias a la implantación de funcionalidades nuevas que fueron aportadas por los propios usuarios en el primer experimento, la aplicación a obtenido buenas calificaciones respecto a su mejora desde el primer MVP como podemos observar en la Figura 4.50.

Conclusiones

Los resultados de este segundo experimento nos muestran grandes resultados, donde la aplicación, poco a poco, recibe más aceptación. Uno de los puntos clave ha sido tratar de desarrollar funcionalidades fruto de las propias peticiones de los usuarios.

Algo a mejorar y a tener en cuenta será disponer de un sistema de actualizaciones para variar periódicamente la información que se muestra en la aplicación, sin este cambio, la aplicación podría dejar de usarse paulatinamente ya que, poco a poco los usuarios tendrían menos interés por entrar a la aplicación por no disponer de información nueva y actualizada.

4.8. Cronología del proyecto

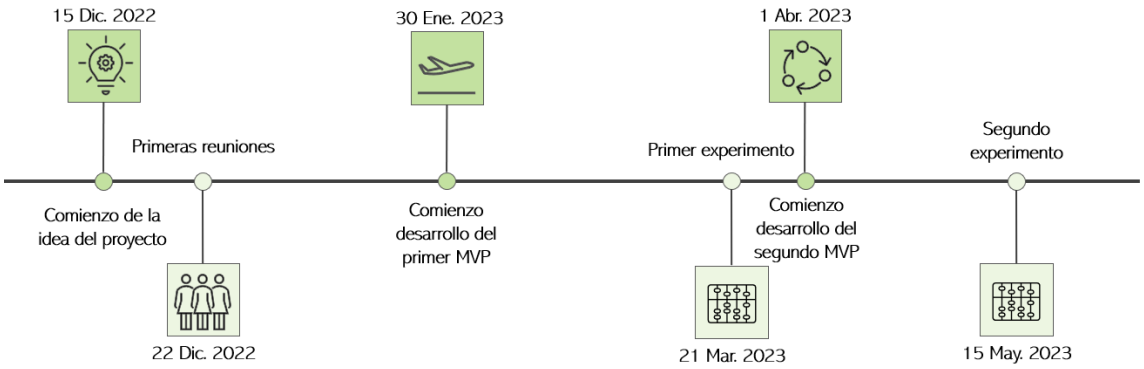


Figura 4.51 Línea temporal del avance del proyecto

Durante el mes de diciembre de 2022 y enero de 2023 se estuvieron realizando charlas con diferentes usuarios, por lo general, especialistas sanitaros, para lograr acordar finalmente cual sería el mejor enfoque para la aplicación.

Tras, finalmente, tener una idea base clara, a partir de febrero de 2023 se dio comienzo al proyecto.

Las tareas a realizar se organizaron en la aplicación web Trello. En esta aplicación se crearon en un inicio 4 listas en las que ir estableciendo las tareas a realizar.

- **Lista de tareas**: donde se almacenaban todas las tareas, por orden de preferencia, pendientes de realizar. Con las etiquetas incluidas en las tarjetas de Trello, se indica a que sprint pertenecía cada tarjeta.
- **En proceso**: las tareas que se encuentran realizando en ese mismo instante. Por lo general, en esta lista no deben acumularse demasiadas tareas simultáneamente.
- **En test**: al finalizar una tarea, esta pasa a test, donde deberá pasar unas pruebas de aceptación establecidas en dichas tarjetas para poder asegurar el correcto funcionamiento de la aplicación.
- **Hecho**: en esta lista se encuentran las tareas realizadas y testeadas.

Las tareas asignadas a cada *Sprint* fueron organizados en función del conjunto de tareas siguiente:

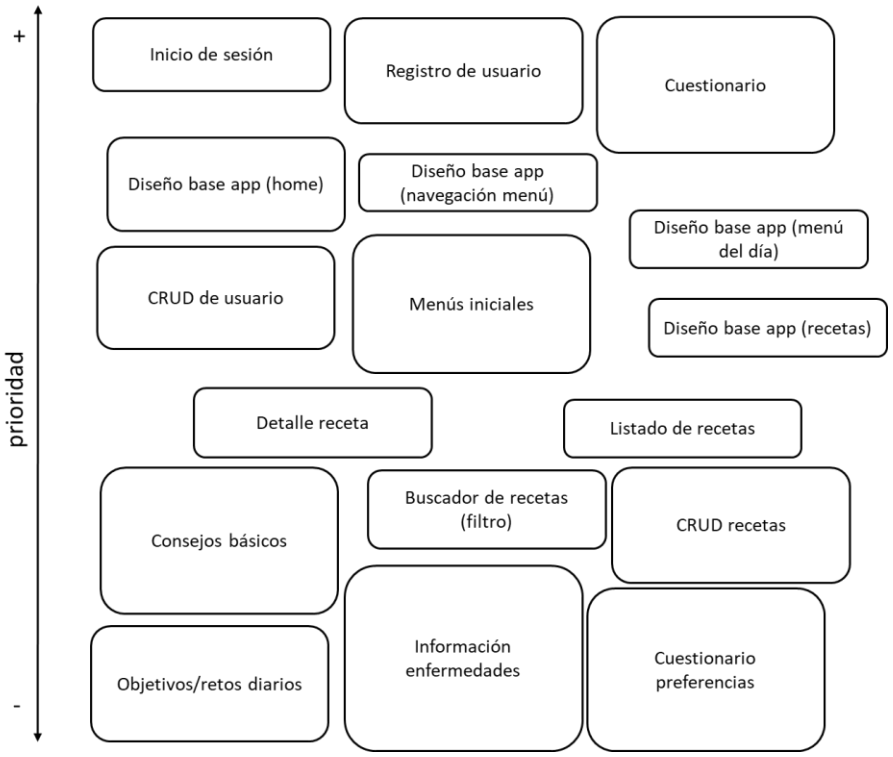


Figura 4.52 Mapa de características inicial

En la Figura 4.52 se presentan todas las tareas que deben llevarse a cabo, organizadas por prioridad. Esta clasificación nos ayudará a determinar el orden en el que las tareas se agregarán a los *Sprints*. Además, el grosor de cada tarea nos proporciona una estimación aproximada del esfuerzo requerido para completarla.

Para realizar dichas estimaciones, no solo se tiene en cuenta la dificultad que puede llevar únicamente programarlo, si no, todo su trabajo previo y posterior de preparación, diseño, implementación y pruebas que debe seguir cada tarea. Además, algunas tareas específicas, como pueden ser Consejos básicos o Menús iniciales, llevan un trabajo extra incorporado, que es la recopilación de información de los expertos y su validación final de los resultados mostrados en la aplicación.

4.8.1. Febrero 2023

En febrero, con una idea de la aplicación ya asentada, se realizó la base del proyecto de la aplicación, así como la realización del estudio previo necesario para comprender la viabilidad y obstáculos que podríamos encontrarnos a la hora de lanzar nuestra aplicación.

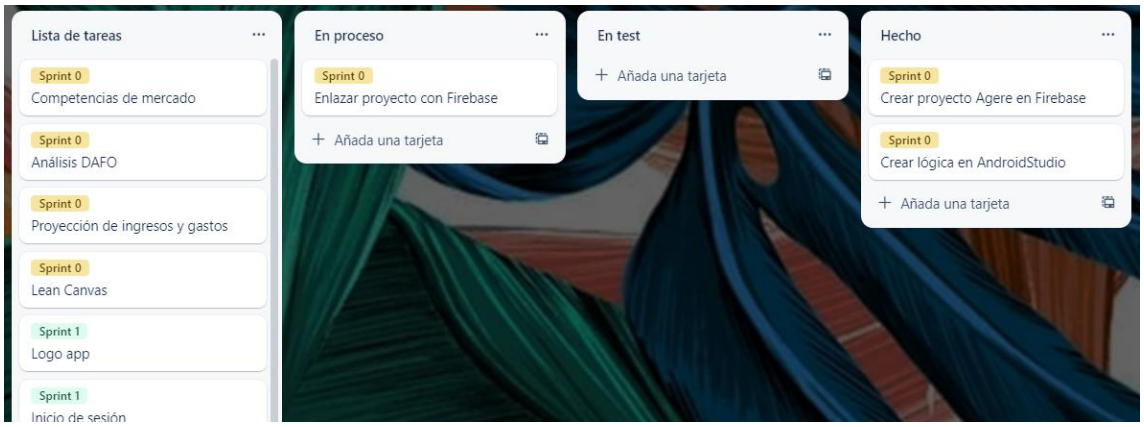


Figura 4.53 Estado de Trello en Febrero 2023

4.8.2. Marzo 2023

A finales de Febrero, tras algunas entrevistas y debates en los que se mejoraron algunas ideas del estudio previo y tras finalizar la base del proyecto, es decir, la creación del mismo y su conexión con la base de datos, se dio finalmente el visto bueno a la aplicación y pudo comenzar su desarrollo.

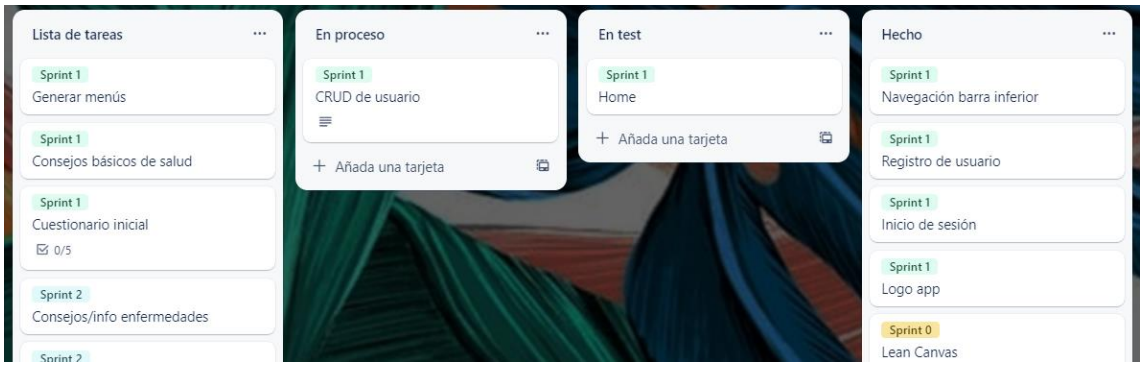


Figura 4.54 Estado de Trello en Marzo 2023

Durante este mes se desarrolló exclusivamente el primer producto viable de nuestra aplicación para así poder realizar el primer experimento y recibir la retroalimentación necesaria de los primeros usuarios.

Este primer producto viable debía satisfacer las siguientes características de la aplicación:

- Inicio de sesión
- Registro de usuario
- Cuestionario inicial (datos iniciales necesarios del usuario)
- Visualizar un menú inicial
- Visualizar consejos básicos de salud

- Editar perfil

Con estas funcionalidades iniciales, el primer MVP podría darse por desarrollado y con este comenzar los primeros experimentos con usuarios.

4.8.3. Abril 2023

En Abril, tras la realización del primer experimento, se tuvieron que llevar a cabo ciertos cambios en la organización de las tareas, tratando de dar cierta prioridad, como ya se explicó en el punto 4.7.1.3 Conclusiones primer MVP, a las mejoras dadas por los usuarios, sin dejar de lado las tareas ya programadas con anterioridad.

Es por esto que en el tablero de Trello se realizaron ciertas modificaciones:

- **Mejoras**: una nueva tabla en la que se recopilarían todas las ideas obtenidas como resultado de las pruebas con los usuarios finales.
- **Bugs**: las pruebas de aceptación establecidas no siempre logran cubrir todos los casos posibles, y los usuarios finales fueron capaces de producir algunos de estos fallos, los cuales eran registrados en una nueva tarjeta con una etiqueta llamada “Bugs”, cuando existían tarjetas de este estilo, se les trataba con la mayor prioridad posible para asegurar siempre el mejor funcionamiento en el producto final.

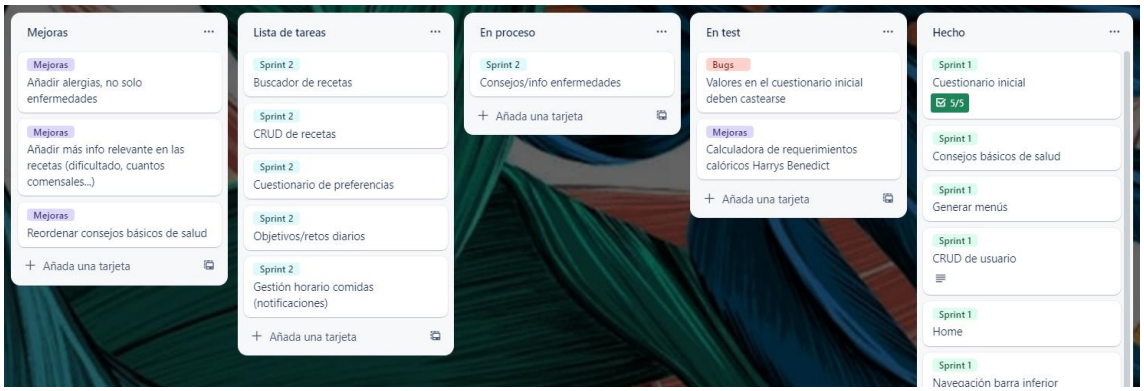


Figura 4.55 Estado de Trello en Abril 2023

Durante el desarrollo de este segundo MVP, se tuvieron que priorizar algunas mejoras, dejando para un siguiente desarrollo tareas del sprint 2 menos prioritarias. Finalmente, las tareas que se decidieron desarrollar para este segundo sprint fueron:

- (Mejora) Calculadora Harrys Benedict
- (Mejora) Añadir información relevante en las recetas
- (Mejora) Reordenar consejos de salud
- Consejos/información de enfermedades
- Listado de recetas
- Buscador de recetas
- CRUD de recetas

4.8.4. Mayo 2023

Finalmente alcanzamos Mayo, donde vemos que ciertas tareas del Sprint 2, al haber dado cierta prioridad a algunas mejoras, se tuvieron que dejar en un segundo plano, esperando a ser desarrolladas para un siguiente MVP.

Como se puede observar, también surgieron nuevas mejoras debido al segundo experimento que se realizó, a las cuales también se les dará cierta prioridad en el desarrollo del siguiente MVP.

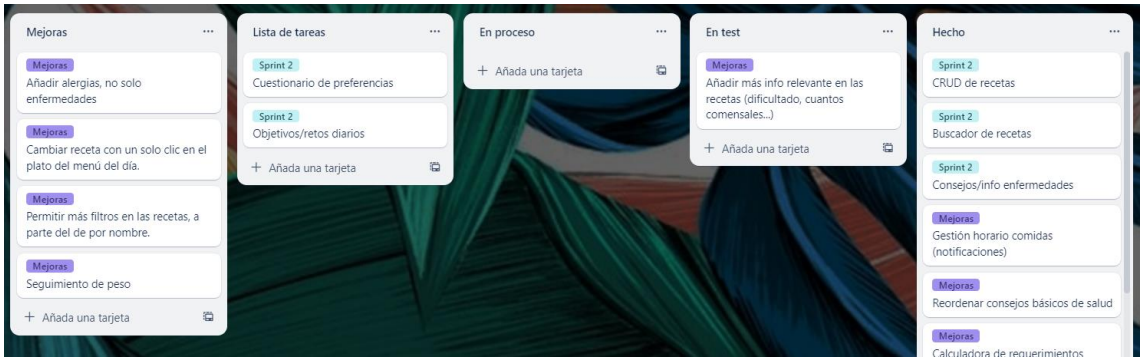


Figura 4.56 Estado de Trello en Mayo 2023

4.8.5. Observaciones

La participación de los usuarios en las pruebas ha sido fundamental para comprender sus necesidades y expectativas, lo que ha llevado a una evolución significativa de la aplicación. Gracias a sus comentarios y sugerencias, se han podido incorporar características valiosas que mejoran la experiencia del usuario y agregan valor al producto.

Sin embargo, la incorporación de estas nuevas funcionalidades ha generado la necesidad de reorganizar y replantear la prioridad de las tareas. En algunos casos específicos, ha sido necesario descartar temporalmente ciertas tareas para realizarlas en un futuro. Esta decisión se ha tomado con el objetivo de asegurar que las nuevas funcionalidades se implementen de manera efectiva y cumplan con altos estándares de calidad.

Aunque posponer ciertas tareas puede resultar una medida difícil, se ha considerado como una estrategia necesaria para mantener el enfoque en las funcionalidades más importantes y brindar una experiencia óptima a los usuarios.

Capítulo 5

Conclusiones y trabajo a futuro

5.1. Despliegue

La parte final en el desarrollo de una aplicación es el despliegue de la misma. El primer paso, tras habernos asegurado de un correcto funcionamiento de nuestra app, tras aplicar exhaustivos test en la misma, podemos comenzar a plantearnos donde nos interesaría lanzarla al mercado.

Al tratarse de una aplicación desarrollada para dispositivos Android, tiendas destinadas a la publicación de aplicaciones iOS quedan descartadas, aún así nos quedan un gran numero de opciones posibles para elegir:

- Samsung Galxy Store
- Amazon Appstore
- HUAWEI AppGallery
- Xiaomi Mi GetApps
- OPPO App Market
- VIVO App Store
- ...

Las opciones como vemos son muchas, pero entre todas las opciones disponibles, finalmente nos decantamos por Google Play Store. Esto es porque entre todas las opciones disponibles en el mercado, Google Play es la más extendida y usada con diferencia, además de todas las facilidades y ayudas que aportan para desarrolladores.

Lo primero que deberemos tener en cuenta es que necesitaremos acceso a *Google Play Console*[24], la consola de desarrollo oficial de Android para desarrolladores.

El proceso de registrarnos en esta consola, tiene un coste fijo de 25$, por lo que, a partir de este momento, el proceso explicado será meramente informativo para ver como se realizaría el proceso.

[24] Google Play Console: developer.android.com/distribute/console

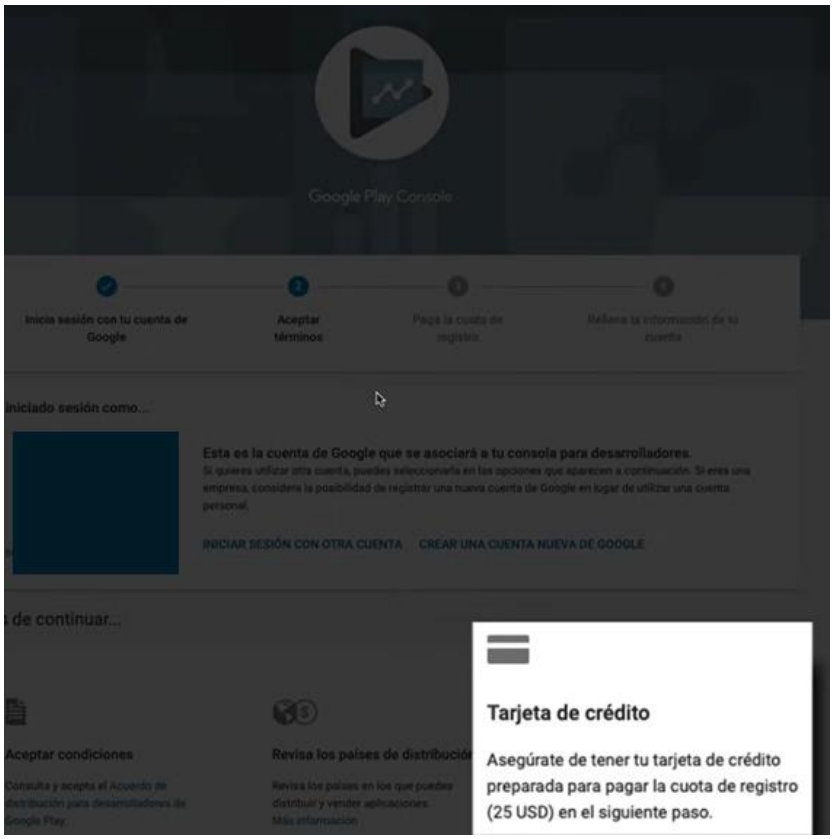


Figura 5.1 Proceso de registro en Google Play Console

Una vez finalizado el proceso de registro, ya podremos acceder a *Google Play Console*, y será aquí donde podremos seleccionar la aplicación que queramos subir a tienda, seleccionando el botón situado en la esquina superior derecha.

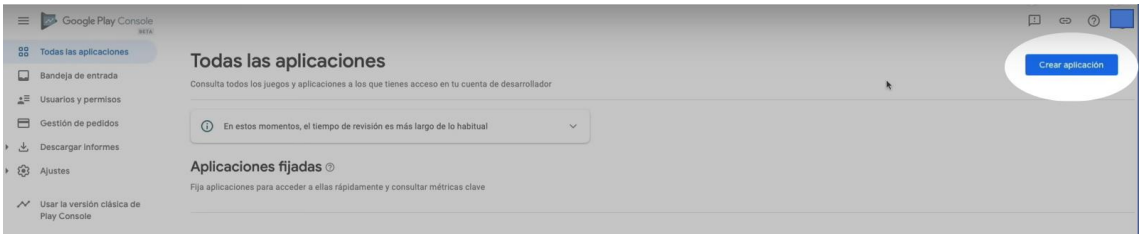


Figura 5.2 Crear una versión en Google Play Console

Esto nos llevará a una serie de ventanas en las cuales se deberá rellenar toda la información propia de la aplicación que será visible desde la propia *Google Play Store*.

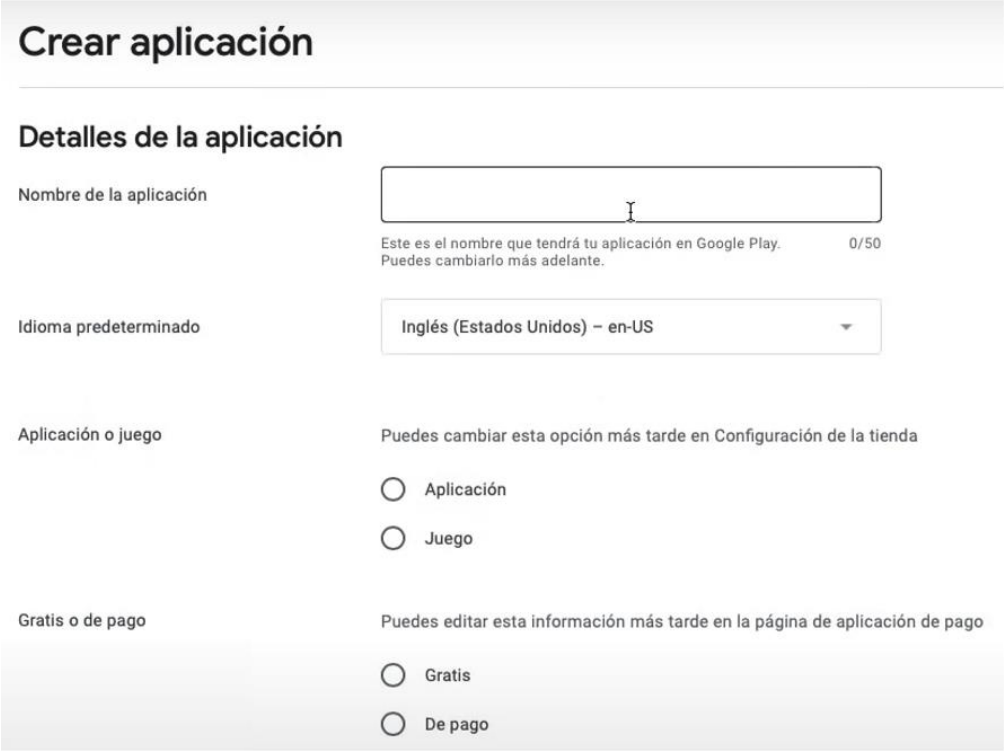


Figura 5.3 Datos requeridos de una nueva aplicación en Google Play Console

Algunos de los campos a rellenar son:

- Nombre de la aplicación
- Idioma predeterminado
- Indicar si es una aplicación o un videojuego
- Indicar si se trata de una aplicación de pago o es gratis
- Añadir más idiomas disponibles en nuestra aplicación
- Una descripción breve
- Una descripción completa
- Icono de la aplicación
- Portada
- Capturas de pantalla de teléfonos móviles
- Capturas de pantalla de tablets (de 7 y 10 pulgadas)
- Video demo subido en YouTube

Una vez completados todos los datos podremos subir finalmente nuestra aplicación.

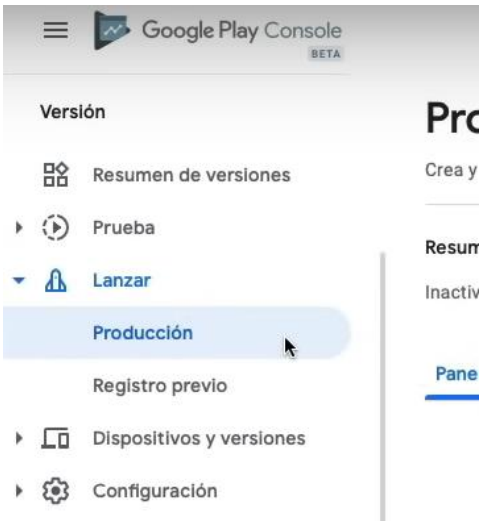


Figura 5.4 Menú Google Play Console

En el panel de *Google Play Console* se dispondrá de diversos apartados, de los que cabe destacar tres de ellos:

- Resumen de versiones: en este apartado se podrán visualizar el historial de todas las versiones de la aplicación que lancemos.
- Prueba: en este apartado se nos permitirá lanzar nuestra aplicación a un conjunto limitado de usuarios para realizar pruebas.
- Lanzar: aquí se subirán y gestionarán las versiones finales que tendrán disponible los usuarios desde Google Play Store.

Ahora nos centraremos en como lanzar una aplicación al mercado, por lo que navegaremos a Lanzar → Producción, y una vez dentro, seleccionaremos el botón que nos indica “Crear versión”

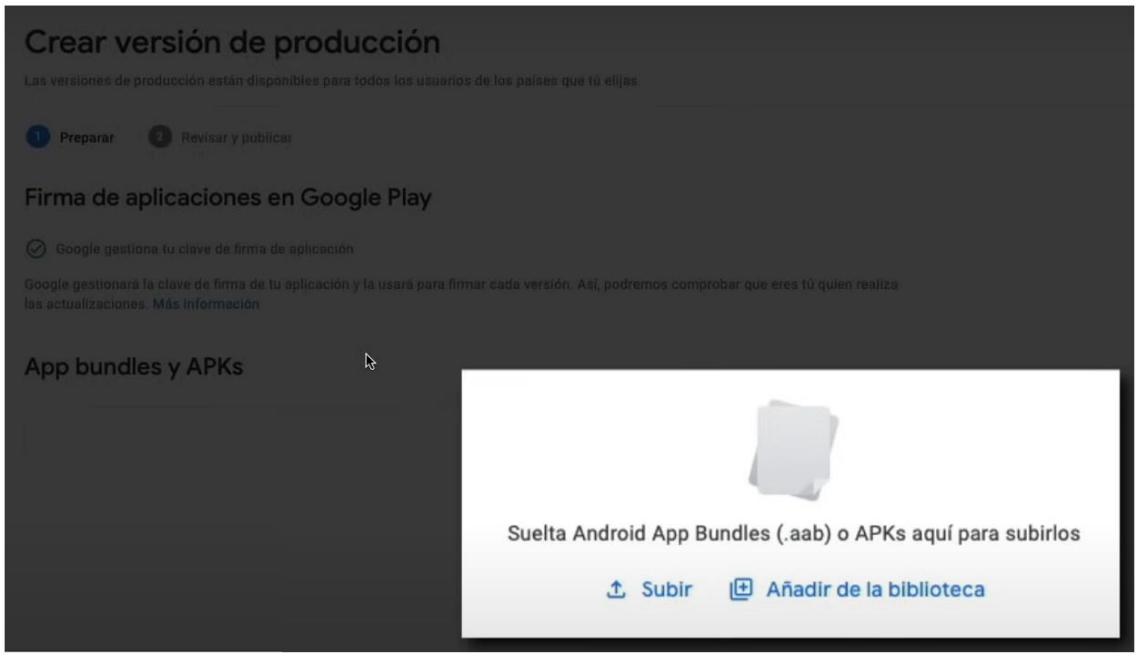


Figura 5.5 Agregar archivo de la aplicación en Google Play Console

Es aquí donde deberemos añadir nuestra aplicación. Para obtener el archivo a subir tendremos que ir a nuestro proyecto en *Android Studio* → *Build* → *Build Bundle(s) / APK(s)*.

Esta sección nos permitirá crear tanto una APK como una *Android App Bundle*. En nuestro caso escogeríamos la opción de *Bundle*, ya que esta aporta mayor seguridad y optimización a nuestra aplicación.

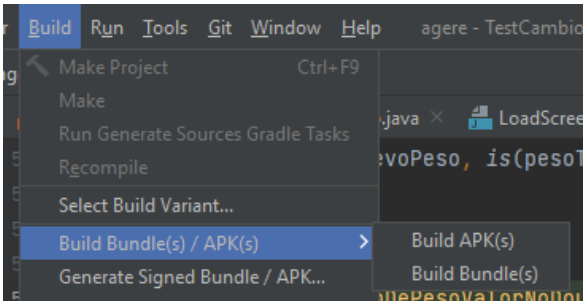


Figura 5.6 Generar archivo de la aplicación desde Android Studio

Una vez generado el archivo y subido a *Google Play Console*, podremos ver la aplicación subida.

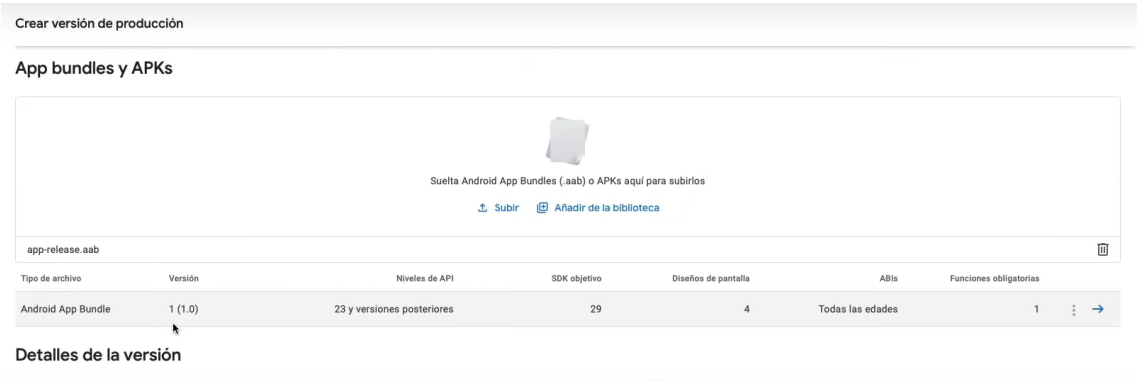


Figura 5.7 Versión actual de nuestra aplicación

Cuando guardamos la versión y le damos a revisar, habrá que comprobar si hay algún error y se requiere rellenar algún dato más. Google este proceso lo tiene bastante guiado por lo que no se deberían encontrar dificultades en completar los datos faltantes.

Una vez completados todos los datos, si es que fuese necesario, volveríamos a seleccionar Revisar versión. Una vez aceptados todos los datos, nos aparecerá un botón para lanzar finalmente la versión.

Figura 5.8 Lanzamiento de la aplicación en Google Play Console

Con esto nuestra aplicación podrá estar disponible para todos los usuarios desde la Google Play Store y podrá ser actualizada con nuevas versiones en cualquier momento.

5.2. Conclusiones

Tras el desarrollo de este proyecto podemos afirmar que el objetivo inicial que se estableció al comienzo se puede establecer como cumplido, a excepción de un subobjetivo relacionado con el despliegue de la aplicación en la plataforma de Google Play.

En particular, consideramos que no era conveniente realizar un pago para comenzar a utilizar dichos servicios, especialmente porque nuestra aplicación se encontraba en etapas tempranas de desarrollo. Además, estimamos que aún falta al menos un MVP adicional para lograr una aplicación que se sienta completamente funcional. Aunque la versión actual cumple con su propósito, todavía carece de ciertas funcionalidades que, en nuestra opinión, son necesarias para su lanzamiento al mercado.

En este trabajo se ha detallado el proceso seguido para desarrollar una idea de negocio y evaluar su viabilidad en el mercado actual. Mediante este análisis, se ha logrado definir la estrategia de negocio más adecuada para competir en un entorno altamente competitivo.

Actualmente, se dispone de un producto software que aún debe continuar desarrollándose, listo para comenzar un tercer MVP y proseguir con la evolución del mismo.

Tal y como se encuentra actualmente Agere, un usuario ya dispone de información útil sobre su salud y consejos básicos que toda persona debería conocer, además de disponer de un menú generado especialmente para el, además de una biblioteca de recetas donde buscar alternativas. También cuenta con algunas funcionalidades extra, como es el caso de la Calculadora de Requerimientos calóricos Harrys Benedict, la cual aconseja las kcal a consumir por el usuario en función de algunos parámetros.

A la hora de escoger este proyecto, éramos conscientes de el gran reto que suponía adentrarse en una aplicación que estuviese tan estrechamente relacionada con aspectos médicos y nutricionales, ya que los conocimientos en este ámbito son limitados. No obstante, gracias a la ayuda desinteresada de ciertos especialistas en el ámbito médico y tras muchas reuniones en las que perfilar que información era más adecuada mostrar y cual era mejor obviar, se han logrado obtener resultados satisfactorios.

Cabe destacar que este proyecto tampoco habría sido posible sin la valiosa colaboración de los voluntarios que se ofrecieron para la realización de los experimentos. Su actitud fue en todo momento muy positiva y activa, y permitió obtener nuevas funcionalidades en la aplicación que sin su ayuda no podrían haber sido llevadas a cabo.

Una de las mayores dificultades encontradas fue lograr unificar los conocimientos médicos de los especialistas y las peticiones de los usuarios tras los experimentos realizados a lo largo de este proyecto. Finalmente esto fue posible gracias a la participación activa de ambos grupos y la arquitectura usada en nuestra aplicación, que permitía realizar cambios en tecnologías o lógicas de negocio rápidamente y con la mínima repercusión en el resto de la aplicación.

En cuanto al desarrollo de la propia aplicación, conforme se avanzaba en el proyecto, fueron surgiendo ciertos obstáculos fruto de algunas decisiones tomadas, como es el uso de Java como lenguaje para el desarrollo de esta aplicación. En un principio fue escogido por ser el lenguaje nativo de Android, además de presentar una integración

óptima con los servicios de Google. Sin embargo, Java está comenzando a mostrar ciertas limitaciones y se le considera un lenguaje que está quedando rezagado para ciertas situaciones.

Mediante una estructura de interfaces y *callbacks* se logró solventar el obstáculo que presentaba la asincronía en Java (Android) y, a partir de ese momento, la aplicación funcionó sin mayores problemas.

A pesar de los desafíos encontrados, la realización de pruebas y la validación de ciertos datos resultó ser bastante sencilla gracias al uso de Java. Las tecnologías que empleamos ya estaban integradas por defecto en este lenguaje, lo que evitó la necesidad de realizar instalaciones adicionales.

Es importante resaltar el acierto de utilizar los servicios de Firebase en nuestro proyecto, ya que, al tratarse de un servicio de Google, podíamos contar con una alta disponibilidad, cercana al 100%, además de su velocidad para obtener respuestas a las peticiones de datos (de menos de 1 segundo) y la seguridad que ofrece nos permite aportar seguridad y privacidad a los datos de nuestros usuarios.

Este proyecto ha supuesto un autentico reto para el equipo de desarrollo que, por el momento, únicamente ha estado formado por una única desarrolladora *full stack*, que ha aprendido, luchado y ganado con cada una de las tecnologías, técnicas y metodologías que se han empleado a lo largo del trabajo, además de haber podido enfrentarse a la dirección de un equipo formado con personas fuera del ámbito informático.

Esta experiencia ha sido enriquecedora, ya que ha brindado la oportunidad de adquirir nuevos conocimientos y habilidades que no se poseían al inicio del proyecto. Sin embargo, nada de esto habría sido llevado a cabo sin los conocimientos adquiridos previamente a lo largo del grado. Asignaturas como Proceso de Software o Proyecto de Ingeniería de Software permitieron adaptar una correcta dinámica y desarrollo de Agere, mientras que otras asignaturas como, Ingeniería del software, Diseño de Software, Calidad de Software, Bases de datos y sistemas de información, Gestión de proyectos, Análisis y especificación de requisitos y Mantenimiento y evolución de software entre otras muchas asignaturas, han aportado los conocimientos, prácticos y teóricos, necesarios para el desarrollo de todo el proyecto.

Es gracias a esto que se ha podido observar la importancia de la formación académica y cómo los conocimientos adquiridos en diferentes asignaturas pueden converger y complementarse en un proyecto concreto.

5.3. Trabajo a futuro

Actualmente, solo se permite a un administrador añadir recetas a la plataforma. Como trabajo a futuro, una funcionalidad importante a añadir será permitir a cualquier tipo de usuario añadir recetas a la plataforma, de tal forma que el crecimiento de variedad en recetas sea amplio y satisfaga la demanda de los usuarios.

Obviamente, para continuar aportando una aplicación que sea segura para los usuarios, todas las recetas añadidas no deberían estar disponibles automáticamente para todos los usuarios, ya que, como se mencionó al inicio de esta memoria, hoy en día cualquier persona puede publicar cualquier información, pero en nuestra aplicación queremos aportar veracidad a nuestros datos y permitir a los usuarios confiar en la información dada. Es por esto que, cuando un usuario quisiera añadir una nueva receta, esta deberá ser revisada y verificada por un experto antes de ser mostrada al resto de usuarios.

Por otro lado, también se comenzará a aportar acceso a información exclusiva mediante la adquisición de la suscripción premium a la plataforma. Mediante plataformas de pago como Stripe[25], esta tarea resultaría relativamente sencilla de implementar, ya que su documentación y opciones para desarrolladores son bastante amplias.

En cuanto a la parte de alimentación de nuestra aplicación se dé por completada y satisfactoria por nuestros usuarios, a demás de continuar un mantenimiento de la misma, también se buscará ampliar la aplicación hacia el sector deportivo, ya que somos conscientes que una buena alimentación, si no se combina con ejercicio físico, los resultados pueden ser algo insatisfactorios para los usuarios. Con esto también conseguiríamos seguir destacándonos en el mercado tan competitivo que nos encontramos y posicionarnos como una aplicación referente en salud y bienestar.

A parte de continuar el desarrollo que se estaba siguiendo actualmente, también se buscará abarcar a un mayor número de usuarios comenzando el desarrollo de la aplicación para dispositivos iOS. Gracias a Flutter[26], esto sería altamente rentable, ya que con el desarrollo de la propia aplicación para este sistema operativo, se permitiría lanzar también a Android y plataformas web. Es decir, con un único desarrollo, podríamos lanzar una aplicación a prácticamente cualquier dispositivo.

Esto nos permitiría abaratar considerablemente los costos, además que su integración con la aplicación actual no resultaría excesivamente compleja, gracias a la estructura que tiene Flutter y su fuerte apoyo de Google por promover el uso del mismo.

[25] Stripe: https://stripe.com
[26] Flutter: https://flutter.dev

5.4. Referencias

[1] Infracomunicación: El exceso de información genera desinformación. Consultado en febrero de 2023. https://www.elperiodico.com/es/entre-todos/participacion/infracomunicacion-el-exceso-de-informacion-genera-desinformacion-194515

[2] ¿Cuántas personas verifican la información que consultan en internet?. Consultado en febrero de 2023. https://es.statista.com/grafico/27146/porcentaje-de-personas-que-verificaron-informacion-encontrada-en-sitios-web-de-noticias-o-redes-sociales-en-los-ultimos-3-meses/#:~:text=Seg%C3%BAn%20datos%20de%20Eurostat%2C%20solo,o%20redes%20sociales%20en%202021

[3] Autodiagnóstico y terapias por internet: ¡Peligroso hábito!. Consultado en Junio de 2023. https://efesalud.com/autodiagnostico-internet-peligroso-nuevo-habito/

[4] Asociación Española de Comunicación Sanitaria. *Estudio piloto sobre el uso de las aplicaciones móviles como información nutricional para el consumidor de alimentos*. Revista española de comunicación en salud, 2021.

[5] 2020, el año de la tecnología fitness. Consultado en Junio de 2023. https://es.statista.com/grafico/23058/ingresos-por-compra-de-wearables-y-descargas-de-apps-de-fitness/

[6] Planes de precios. Consultado en Mayo de 2023. https://firebase.google.com/pricing?hl=es&authuser=1

[7] Lean Startup: monta tu negocio invirtiendo de manera segura gracias a este método. Consultado en Febrero de 2023. https://es.godaddy.com/blog/lean-startup/

[8] Eric Ries.*El método Lean Startup*. Crown Business Publishing, S.A., 2011.

[9] Documentación de Firebase Realtime Database. Consultado en febrero de 2023. https://firebase.google.com/docs/database

[10] Material y transparencias proporcionadas por los cursos realizados en la Escuela Técnica Superior de Ingeniería Informática de la Universidad Politécnica de Valencia:

- Análisis y Especificación de Requisitos
- Proyecto de Ingeniería de Software
- Mantenimiento y Evolución de Software

Anexo I

Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**	**X**			
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**	**X**			
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**			**X**	
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**			**X**	
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**	**X**			
ODS 17. **Alianzas para lograr objetivos.**			**X**	

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Los Objetivos de Desarrollo Sostenible representan una poderosa visión para construir un futuro más equitativo, sostenible y próspero para todos. En este contexto, una aplicación que brinda consejos sobre nutrición puede desempeñar un papel significativo en la promoción y el logro de varios de estos objetivos.

De los anteriores objetivos de desarrollo sostenibles mencionados, el proyecto lo encontramos estrechamente relacionado con:

- **Hambre cero**: La aplicación puede contribuir a este objetivo al brindar consejos sobre una alimentación saludable y equilibrada, promoviendo la seguridad alimentaria y la nutrición adecuada. Esto ayuda a combatir el hambre y la desnutrición, fomentando prácticas de alimentación sostenibles.

- **Salud y bienestar**: este proyecto promueve el mantener una vida saludable, mejorar unos hábitos nutricionales que es posible que no se tuviesen previamente e informa al usuario con datos veraces sobre los que respaldarse para cuidar su salud.

- **Educación de calidad**: no es de extrañar que a día de hoy no se tenga una buena educación en lo que respecta a la buena alimentación, y mediante los consejos de salud básicos aportados por la aplicación y la información sobre las enfermedades, aseguramos una mejor educación en salud y nutrición en los usuarios que disponen de Agere.

- **Paz, Justica e Instituciones sólidas**: este ODS también puede estar vinculado en cierta medida, ya que fomenta la difusión del conocimiento, lo cual, de manera indirecta, contribuye a la creación de sociedades más informadas y, en última instancia, más justas y pacíficas.

Es importante destacar que, si bien estos son los ODS más relevantes, la aplicación también se encuentra en menor medida relacionada con otros objetivos como podría ser **Producción y Consumo Responsables** o **Trabajo decente y crecimiento económico**.

Finalmente, podemos concluir que, al empoderar a las personas con conocimientos y herramientas para tomar decisiones informadas sobre su nutrición, podemos fomentar cambios positivos en la salud, la sostenibilidad y la equidad. La aplicación se convierte así en un catalizador para el cambio, permitiendo que cada individuo desempeñe un papel activo en la construcción de un mundo mejor y más sostenible para todos.