



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**APLICACIÓN ANDROID PARA EL
CONTROL DE MICRONUTRIENTES Y
RECOMENDACIONES ALIMENTÍCIAS**

Autor: Manuel Rueda Algar

Tutor(a): Raúl Alonso Calvo

Madrid, mayo, 2023

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: APLICACIÓN ANDROID PARA EL CONTROL DE DIETA Y
RECOMENDACIONES ALIMENTÍCIAS

Mayo, 2023

Autor: Manuel Rueda Algar

Tutor: Raúl Alonso Calvo

Departamento de Lenguajes y Sistemas

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

El trabajo realizado consiste en una aplicación en lenguaje Java para dispositivos móviles Android. El principal objetivo de la aplicación es ayudar a personas mayores o con alguna deficiencia degenerativa a llevar un progreso y almacenamiento de los alimentos consumidos cada día, es decir, su dieta.

Para conseguir este objetivo, en la aplicación, hay distintos apartados que permiten al usuario no solo conocer información y micronutrientes de diversos alimentos, sino que, además, se encuentran formularios en los cuales, el usuario puede introducir los alimentos que ha ingerido ese día y se almacenan en su perfil.

Gracias a almacenar su dieta alimenticia, se puede conocer el consumo de micronutrientes del usuario y, por tanto, se puede hacer un cálculo total de los micronutrientes para conocer el grado de alimentación y valores nutricionales del usuario. A raíz de estos datos, se hacen recomendaciones en las cuales se aconseja al usuario en ingerir algún tipo de alimento que falta o es escaso en su dieta.

La aplicación está compuesta de varias secciones y funcionalidades. De primera mano, se encuentra una pantalla de inicio de sesión y registro para el usuario, donde se conectará con una base de datos Firebase de Google para almacenar los datos del usuario. A continuación, una vez el usuario ha iniciado sesión, entra en la página principal donde puede ver un pequeño menú para elegir donde quiere desplazarse en la aplicación.

Las principales funcionalidades de la aplicación son, una lista de comidas y alimentos conectada a una base de datos donde se puede buscar y filtrar por palabras y al seleccionar una de ellas, se podrá ver información nutricional. Y la siguiente, se trata de unos formularios donde el usuario puede buscar alimentos y añadirlos a estos mismos, que constituirán su dieta diaria.

Además de estas funciones, se encuentran tres apartados auxiliares que ayudan con el uso y ayuda de la aplicación:

El primero consiste en el perfil del usuario, donde se observan los propios datos del usuario y, además, se muestran las dietas almacenadas y las recomendaciones que se aportan según su nutrición.

Los siguientes dos apartados atribuyen al uso de la aplicación en sí, ya que uno de ellos consiste en una sección de ayuda donde se mostrarán los datos del creador de la aplicación.

En conjunto y para finalizar, se trata de una aplicación de gran utilidad para personas que necesitan llevar a cabo una dieta, especialmente personas mayores que necesitan ciertos nutrientes más que otros, y la facilidad de uso de la aplicación contribuye a su ayuda.

Abstract

The project consists of an application in Java language for Android mobile devices. The main objective of the application is to help elderly people or people with some degenerative deficiency to keep a progress and storage of their food consumed every day, that is, their diet.

To achieve this goal, in the application, there are different sections that allow the users not only to know information and micronutrients of various foods, but also fill forms in which the users can enter the food they have eaten that day and store it in their profile.

By storing their diet, the users' micronutrient intake can be known and, therefore, a total calculation of micronutrients can be made to calculate the degree of nutrition and nutritional values of them. As a result of this data, recommendations are made in which users are advised to increase the ingestion of some type of food that is lacking or scarce in their diet.

The application is composed of several sections and functionalities. First, there is a login and registration screen for the user, which will be connect to a Google Firebase database and store the user's data. Then, once the user is logged in, he/she enters the main page where they can see a small menu to choose where they want to scroll in the application.

The main functionalities of the application are, a list of meals and foods connected to a database where users can search and filter by words, additionally selecting one of them which will show them extended nutritional information. And the next one, it is about forms where the user can search for foods and add them to these same, which will constitute their daily diet.

In addition to these functions, there are three auxiliary sections that help with the use and assistance of the application:

The first consists of the user's profile, where the user's own data are observed and in addition, the stored diets and the recommendations provided according to their nutrition are shown.

The next two sections attribute to the use of the application itself, since one of them consists of a help section where the data of the creator of the application will be shown.

Overall and finally, this is a very useful application for people who need to carry out a diet, especially older people who need certain nutrients more than others, and the ease of use of the application contributes to their help.

Tabla de contenidos

1	Introducción.....	1
1.1	Motivación y necesidad del proyecto.....	1
1.2	Objetivos	2
1.3	Planificación.....	2
1.4	Alcance del proyecto.....	3
1.5	Estructura de la memoria.....	4
2	Estado del arte	6
3	Desarrollo	12
3.1	Estructura de la aplicación.....	12
3.1.1	Arquitectura de la aplicación.....	12
3.1.2	Flujo de navegación de la aplicación	12
3.1.3	Diseño de la interfaz de usuario y experiencia de usuario	12
3.1.4	Diseño de alto nivel.....	13
3.2	Clases y componentes principales	14
3.2.1	Descripción de las clases principales y su funcionalidad.....	14
3.2.2	Componentes de la interfaz de usuario y su interacción con las clases	14
3.3	Conexión con la base de datos.....	16
3.4	Almacenamiento de datos.....	17
3.4.1	Elección de sistema de gestión de datos	17
3.4.2	Gestión de los datos de usuario y perfiles individuales.....	17
3.5	Funcionalidades	18
3.5.1	Inicio de sesión y registro.....	18
3.5.2	Lista de comida.....	21
3.5.3	Formulario	25
3.5.3.1	Diseño e implementación del formulario para registrar alimentos consumidos en 72 horas.....	26
3.5.3.2	Diseño e implementación del formulario para registrar frecuencia de alimentos consumidos	31
3.5.4	Perfil de usuario.....	36
3.5.5	Información.....	38
3.6	Librerías utilizadas	38
4	Resultados y conclusiones	43
4.1	Análisis de los resultados	43
4.1.1	Resultados obtenidos en la implementación de la estructura de la aplicación	43
4.1.2	Resultados obtenidos en la conexión con la base de datos	44
4.1.3	Resultados obtenidos en el almacenamiento y gestión de datos..	44

4.1.4	Resultados obtenidos en la implementación de las funcionalidades	45
4.2	Evaluación del cumplimiento de objetivos	46
4.3	Limitaciones y posibles mejoras	46
4.4	Conclusiones personales	47
5	Análisis de Impacto	48
6	Agradecimientos	49
7	Bibliografía	50

1 Introducción

1.1 Motivación y necesidad del proyecto

El envejecimiento de la población es un fenómeno global que está aumentando rápidamente y representa un desafío importante para las sociedades modernas. Según estadísticas de la Organización Mundial de la Salud, en 2019 había 700 millones de personas mayores de 65 años en el mundo, lo que corresponde al 9% de la población total. Para 2050, se espera que este número se duplique a 1500 millones, o el 16% de la población mundial [1].

El aumento de la esperanza de vida y disminución de la tasa de natalidad han llevado a un cambio demográfico significativo en todo el mundo. Esta tendencia plantea la necesidad de encontrar soluciones innovadoras que permitan ayudar a personas mayores a abordar desafíos asociados a la edad como la nutrición adecuada y la salud cognitiva.

Además, cada vez más personas mayores viven solas en casa, lo que puede aumentar su riesgo de aislamiento social y problemas de salud mental. En muchos casos, estas personas tienen dificultades para preparar y cocinar una comida saludable, lo que conlleva a que tengan una mala ingesta de nutrientes y puede provocar desnutrición finalmente dando lugar a aumentar el riesgo de enfermedades crónicas.

Según la Organización Mundial de la Salud, las enfermedades neurodegenerativas como la enfermedad de Alzheimer y la demencia también están en aumento y se espera que aumenten significativamente para 2050 [2]. Estos trastornos pueden afectar significativamente la calidad de vida de las personas mayores y sus cuidadores.

En este contexto, desarrollar una aplicación móvil específicamente diseñada para recopilar hábitos de vida y alimentación de personas mayores y con degeneración cognitiva es una solución interesante. La aplicación, no solo incluirá información nutricional de macronutrientes presentes en diversas aplicaciones del mercado, si no que, además, aportará información sobre los micronutrientes y polifenoles de los alimentos.

Los polifenoles son unos compuestos bioactivos presentes en distintos alimentos, que han sido demostrados de tener efectos beneficiosos para la salud cerebral y ayudan a la reducción de enfermedades neurodegenerativas. Por tanto, la propuesta llenará ese vacío al ofrecer este tipo de bioactivos y otros micronutrientes, permitiendo beneficiar a estos grupos selectos de la población.

En resumen, la necesidad de recoger hábitos de vida saludable y de alimentación para personas mayores y personas con degeneración cognitiva aborda una necesidad creciente en la sociedad actual. Gracias a esta aplicación y propuesta, podemos abordar esa necesidad que sin duda puede ayudar a estas personas con más dificultades.

1.2 Objetivos

La propuesta se divide en varios objetivos que son necesarios cumplir para que pueda alcanzar un uso. Hay un objetivo que es el principal y del cual desembocan los demás objetivos. Estos son:

El principal objetivo y más importante es el realizar una implementación de una aplicación móvil para la recogida de hábitos alimenticios, que captura toda la esencia de la propuesta y es el paso general para las demás funcionalidades.

A raíz de este objetivo nacen otros tres objetivos específicos, siendo el primero de ellos crear una interconexión con un servicio web de los micronutrientes presentes en un alimento. Gracias a cumplir este objetivo, los datos recogidos en la base de datos interconectada permitirán al usuario no solo conocer los micronutrientes de los alimentos que busque, sino que podrá recoger alimentos y añadirlos a su dieta.

El siguiente objetivo se trata del desarrollo de los métodos y servicios necesarios para calcular los valores nutricionales de la dieta de un usuario. Una vez que se han recogido los datos del primer objetivo y el usuario ha hecho uso de ellos, el sistema calculará el valor nutricional del usuario y permitirá darle recomendaciones.

Y, por último, el despliegue del sistema para su uso generalizado y pruebas del sistema, incluyendo distintas funcionalidades como un inicio de sesión y registro, la posibilidad de que el usuario se mueva por un menú y apartados dentro de la aplicación y en general, que el usuario pueda usar la aplicación.

Al cumplir estos objetivos específicos, la propuesta del trabajo de fin de grado y el principal objetivo serán satisfechos, permitiendo al usuario recoger los hábitos de alimentación de su dieta y poder hacerle recomendaciones.

1.3 Planificación

La planificación de la propuesta del trabajo fin de grado está dividida en una lista de tareas que se han tenido que seguir para poder cumplir los objetivos de manera ordenada y secuencial. La lista de tareas es la siguiente:

En primer lugar, se trata del análisis del estado del arte, donde se analizó las aplicaciones actuales que cumplen un objetivo similar y diversos factores como la edad, natalidad y situación con la vejez en la actualidad.

A continuación, la siguiente tarea trata sobre el diseño de la arquitectura del sistema, donde se eligió la estructura de la aplicación, la base de la aplicación y demás modelos que se desarrollarán a lo largo de la memoria.

La tercera tarea trata del diseño del modelo de datos, donde se crearon las clases de la aplicación y para cada una de las funcionalidades de esta, además de inicializar los objetos que serán usados para la base de datos.

La siguiente tarea trata de la creación de la base de datos para la aplicación móvil, proyecto independiente llevado a cabo por el tutor, al

cual se me dispuso de una URL para conectar mi aplicación con esa base de datos para poder obtener los elementos necesarios.

Después y la tarea que más tiempo llevó, implementación de las funcionalidades y de la aplicación móvil, donde se crearon cada una de las partes funcionales de la aplicación permitiendo al usuario manejarse y atravesar distintos menús para que pueda realizar sus consultas. Esta tarea incluye, además, implementar las funciones principales de búsqueda en la base datos de un alimento en específico, de ver todos los alimentos y poder analizar extendidamente sus valores nutricionales y de permitir al usuario realizar su seguimiento de la dieta.

Y, por último, pero no menos importante, la realización de la memoria, análisis de conclusiones del proyecto y futuras líneas de desarrollo que se irán desarrollando a lo largo del documento.

1.4 Alcance del proyecto

El alcance de este trabajo consiste en desarrollar una aplicación móvil para Android en lenguaje Java que permita a los usuarios realizar un seguimiento de su dieta diaria y recibir recomendaciones personalizadas para mantener una alimentación saludable y equilibrada.

La aplicación contará con distintos servicios que aportarán a la usabilidad y comodidad del usuario. En primer lugar, poseerá un sistema de inicio de sesión y registro conectado a la base de datos de Google Firebase que permitirá al usuario crearse una cuenta para que sus datos sean almacenados y pueda tener acceso a ellos independientemente del dispositivo con el que acceda a la aplicación.

A continuación, se entrará a la pantalla principal donde tendrá recordatorios de realizar los formularios correspondientes y en el borde izquierdo se encontrará la funcionalidad principal de la aplicación, un menú de navegación de fácil acceso y preciso que ayudará a cualquier persona ya sea mayor o con alguna degeneración. En este menú de navegación se encontrarán los distintos apartados de la aplicación comenzando con su pantalla principal, seguido del formulario de la recogida de alimentos diarios, una lista con todos los alimentos del sistema y distintos apartados extra como perfil de usuario, información o cerrar sesión.

El apartado formulario permitirá al usuario registrar los alimentos que ha consumido a lo largo del día, parte esencial de la aplicación ya que, gracias a estos formularios, el sistema recogerá los micronutrientes de los alimentos ingeridos por el usuario. Gracias a esta información, se podrá después realizar recomendaciones al usuario para mantener una alimentación saludable y equilibrada, especialmente útil y de forma precisa para quien está destinada esta aplicación, personas mayores o con alguna deficiencia cognitiva.

Otro de los apartados se trata de una lista que está conectada a la base de datos, donde se mostrarán todos los alimentos del sistema acompañado de una barra de búsqueda para filtrar el alimento deseado a encontrar. Si el usuario pulsa un alimento, se accederá a una versión

extendida del alimento donde se encontrarán los diferentes micronutrientes de este.

En resumen, el proyecto se centrará en desarrollar una aplicación móvil para Android que proporcione información detallada sobre la alimentación saludable y equilibrada, permitiendo a los usuarios registrar su dieta diaria y recibir recomendaciones personalizadas para mejorar su salud. La aplicación contará con un servicio de inicio de sesión y registro, una base de datos de alimentos, un formulario fácil de usar, recomendaciones personalizadas y una sección de información general y contacto.

Gracias a estas funcionalidades y propuesta, podría esperar que se viera como una interesante idea para que otras empresas y personas decidan desarrollar una aplicación similar, ya que, como comentado en la introducción y ahora se comentará en el estado del arte, existen aplicaciones similares, pero no como esta específicamente.

1.5 Estructura de la memoria

La memoria está dividida en distintas secciones, cada una con sus propias secciones específicas o algunas simplemente siendo un apartado únicamente. Para ayudar y guiar un poco sobre el contenido de estas, se explican y resumen de la siguiente manera:

- Introducción: apartado inicial donde se muestran los motivos y necesidad del proyecto, objetivos a cumplir, planificación en relación con las tareas a seguir para cumplir esos objetivos. A continuación, el alcance del proyecto donde se traza una visión general del proyecto y se expresa la relevancia de este e innovación y por último este apartado ayudando a la comprensión de la estructura de la memoria.
- Estado del arte: apartado donde se analiza el estado actual de las aplicaciones dedicadas a la nutrición y recogida de hábitos alimenticios, donde se expresa la falta de aplicaciones similares a la propuesta del trabajo fin de grado.
- Desarrollo: sección donde se expone toda la realización de la aplicación y de las funcionalidades. Se muestra de igual forma el código y su explicación al igual que figuras relacionadas con la interfaz gráfica para facilitar el entendimiento en la lectura. Está dividida en distintas secciones cada una de ellas para las funcionalidades de esta, empezando con conceptos e ideas básicas de la estructura de la aplicación y terminando con las librerías utilizadas para su desarrollo.
- Resultado y conclusiones: apartado donde se pone en vista general la evaluación del proyecto, si se han cumplido los objetivos, los puntos fuertes y débiles de la aplicación, elementos que podrían optar a mejora y por último una conclusión sobre mi propia experiencia sobre el desarrollo de la aplicación.
- Alcance de proyecto: sección donde se desarrolla la idea principal de la aplicación y se concluye a quien puede beneficiar una vez se ha desarrollado y mirándola desde perspectiva de impacto social.

- Agradecimientos: sección personal donde dedico unas palabras a familiares y personas importantes que me han ayudado a continuar para culminar en graduarme de la carrera.
- Bibliografía: apartado final donde se muestra la bibliografía para los conocimientos que han sido necesarios y obtenidos en línea.

2 Estado del arte

En un mundo digitalizado como en el que nos encontramos y donde cada vez se le da más importancia a la salud de los habitantes, el seguimiento de la dieta a través de aplicaciones móviles ha ganado popularidad en los últimos años. Donde gracias a algoritmos, inteligencia artificial o simplemente conocimiento de nutricionistas, tener a nuestra disposición de la mano un seguimiento de los nutrientes que ingerimos es de agradecer.

La mayoría de estas aplicaciones se centran en personas jóvenes o de media edad que buscan simplemente tener una dieta saludable, perder peso o ganarlo en ciertos casos, buscando, por tanto, el control de macronutrientes. Algunas de las aplicaciones más famosas de este ámbito son:

- MyFitnessPal [3]: aplicación ampliamente utilizada para el seguimiento de la dieta, donde se permite al usuario realizar un seguimiento de las calorías y nutrientes consumidos.
- Lose it! [4]: aplicación que incita al seguimiento de una dieta ya que se pueden crear metas personalizadas y recibir recomendaciones nutricionales. Además, se pueden escanear códigos de barra de alimentos en supermercados para analizar sus nutrientes antes de comprarlos.

Sin embargo, estas aplicaciones aportan información para la mayoría del público que son personas con una edad menor de 65 años, para seguir sus calorías y nutrientes. La diferencia de cantidad de aplicaciones que hay que siguen estos objetivos respecto a aplicaciones que puedan ayudar a personas mayores o con degeneración cognitivas son abismales.

Por regla general, estas aplicaciones tienen muchas funcionalidades, anuncios, menús que son poco intuitivos, complejos y en general para alguien que no tiene mucha idea en la tecnología, un desafío grande o imposible para utilizar, especialmente para este grupo de personas que tienen dificultades añadidas.

La verdad es que existen algunas aplicaciones que podrían ayudar a estas personas como las siguientes:

- MealLogger [5]: aplicación que permite a los usuarios registrar sus comidas y compartir imágenes de los alimentos, ayudando a la accesibilidad y facilidad de uso a personas mayores.
- Lifesum [6]: aplicación que ofrece comidas y recetas específicamente diseñadas para seguir unos planes de comidas personalizados y adaptados a distintos objetivos.

Estas aplicaciones por ejemplo tienen un grado más amplio de ayuda a personas mayores por su facilidad de uso, pero por su usabilidad, pero no por centrarse exclusivamente en este grupo de personas. Por tanto, corre una necesidad de crear una aplicación como la que se encuentra en esta propuesta de trabajo de fin de grado.

Con una aplicación como la propuesta ayudaría en gran cantidad, ya que, primero con una interfaz intuitiva y accesible, permite ayudar a las personas mayores y con algún problema a utilizarla sin ningún problema y como no tiene tantos menús y funcionalidades como otras, se simplifica mucho su uso. Además, al analizar el perfil individual del usuario, se pueden dar unas recomendaciones nutricionales adaptadas a cada persona para una alimentación adecuada.

Y, por último, pero no menos importante, mostrar información detallada sobre micronutrientes y polifenoles, funcionalidad diferente con otras aplicaciones existentes en el mercado, que permiten brindar información relacionada con la salud cerebral y la prevención de enfermedades neurodegenerativas.

A lo largo del estado del arte hemos podido observar distintas aplicaciones que abordan funcionalidades parecidas a las desarrolladas en este proyecto. Sin embargo, es importante destacar que las tecnologías utilizadas para este proyecto son JSON, Restful API y Android Studio, que permiten mejorar la experiencia de usuario y ofrecer un conjunto de funcionalidades más completo.

La primera tecnología o formato, JSON, ha desempeñado un papel fundamental en el intercambio de datos en el mundo tecnológico. Su desarrollo hasta convertirse en uno de los formatos más populares y ampliamente utilizados en la actualidad se remonta a los primeros días de Internet.

JSON [7] fue creado por Douglas Crockford en la década de 2000 como alternativa ligera y fácil de utilizar respecto a otros formatos existentes en ese momento como XML. Durante ese periodo se estaba llevando una transición hacia aplicaciones web más interactivas y dinámicas, lo que requería un formato que pudiera ser fácilmente consumido y generado por los navegadores web.

La simple y legible estructura de JSON trata de un formato de intercambio de datos para los humanos. Está ordenado con una combinación de pares clave-valor y listas ordenadas, lo que facilita la comprensión y manipulación de datos por parte de los desarrolladores. Además, JSON es un formato basado en texto lo que lo hace legible para los humanos y fácilmente reconocible para las máquinas.

A medida que se empezó a utilizar el formato, con rapidez ganó popularidad y actualmente es ampliamente conocido ya que es compatible con una gran variedad de lenguajes de programación y frameworks, en este caso siendo utilizado para el lenguaje JAVA, lo que ayuda a que se emplee en distintas tecnologías.

Una de las grandes facultades de JSON [8] se trata de que acepta una gran cantidad de tipo de datos. En sus estructuras se pueden incluir, números, cadenas de texto, booleanos, objetos y listas. Gracias a ello, las aplicaciones como la de este proyecto pueden obtener estos objetos y parámetros para escribirlos o para recibirlos y actualizarlos por pantalla.

Sin embargo, el mayor aspecto positivo de JSON se trata de su facilidad a utilizarse en intercambios de datos entre cliente y servidor. La otra tecnología Restful API, permite conectarse con una URL, y, por ejemplo, como ocurre en esta aplicación, se obtiene un JSON que posteriormente es transformado dentro de los correspondientes métodos para obtener la información de los micronutrientes y de los alimentos. Esto da lugar a que JSON sea una de las partes más importantes de la aplicación ya que sin esos datos incluidos en la URL, no se podrían tener los datos de ningún alimento además de indicar sus micronutrientes.

La versatilidad de JSON se extiende más allá de su capacidad para almacenar datos. Además de que es compatible con muchos lenguajes, también es usado como herramienta de manera fluida y en otros servicios relacionados con el desarrollo de aplicaciones. Por ejemplo, hay librerías que usan este formato para simplificarlo y así acelerar el desarrollo de las aplicaciones. Además, JSON sirve como base de datos localmente en la propia aplicación, permitiendo la fácil recuperación y almacenamiento eficiente de estos datos.

Hoy en día, JSON se utiliza en una gran variedad de contextos, desde aplicaciones móviles, servicios web, APIs, aplicaciones móviles entre otros. Es el formato preferido para intercambiar datos entre cliente y servidor además convirtiéndose en el estándar para la configuración de datos en muchos sistemas y frameworks.

En resumen, JSON ha recorrido un largo camino desde que fue lanzado y la modernización de las tecnologías y los lenguajes de programación ha hecho que se haya popularizado en una cantidad inmensa por su facilidad y la liviandad de los datos. Hoy en día, JSON continúa siendo una herramienta invaluable en el desarrollo de aplicaciones y sistemas modernos. Su capacidad para facilitar el intercambio de datos de manera eficiente y compatibilidad con una amplia gama de tecnologías hace que se convierta en una opción segura y ampliamente utilizada para aplicaciones como la de la propuesta u otros servicios web.

La siguiente tecnología ya mencionada, se trata de Restful API, que ha revolucionado la forma en la que las aplicaciones móviles interactúan con los servicios web y servidores, convirtiéndose en un estándar para el intercambio de datos.

Antes de hablar sobre qué es Restful API [9] cabe destacar su historia, que se remonta a los principios de los años 2000 al igual que la de JSON. Roy Fielding propuso la arquitectura Rest como un conjunto de reglas y principios para el diseño de servicios web escalables y flexibles. Restful API se basa en utilizar modelos HTTP con sus respectivos métodos GET, POST, PUT, DELETE para realizar operaciones CRUD (Create, Read, Update, Delete).

La aplicación móvil utiliza este sistema, Restful API, aportando una solución ideal para la comunicación entre el servidor y la misma. Permite la transferencia de datos de manera eficiente a través de internet usando solicitudes HTTP, unas solicitudes que requieren poco ancho de banda

para funcionar, por tanto, útiles para la gran mayoría de usuarios móviles.

La utilización de Restful API en aplicaciones móviles como esta proporcionan una gran variedad de ventajas. En primer lugar, permite una integración sencilla entre servicios web existentes y APIs externas, dando lugar a una recogida de datos en tiempo real cuando se interactúan con otros sistemas. Para esta aplicación, se utilizan estos servicios para obtener el listado de comidas, y como siempre se realiza de manera dinámica y gracias a la rapidez de esta API, se pueden obtener los alimentos actualizados siempre que se utilice.

Otra de las grandes ventajas de Restful API se trata la posibilidad de tratar los datos en las estructuras que sean deseadas, como por ejemplo en JSON. Cuando se realiza la solicitud HTTP de GET en esta aplicación, se obtiene el conjunto de alimentos en su estructura JSON correspondiente, lo que permite trabajar con los datos de manera eficiente. Gracias a esta posibilidad, las distintas aplicaciones y servicios web puede adaptar cómo son la estructura de los datos que se trabajan en Restful API y trabajar como se crea conveniente.

Por eso, otro punto importante de esta tecnología se trata de su flexibilidad. La gran variedad de formas de utilizar Restful API ayuda a que el dinamismo de uso en las aplicaciones esté ajustado. Dependiendo de la URL a la que se llame, se pueden obtener distintos resultados, como para filtrar las listas de comidas y como es una herramienta que se puede llamar en cualquier momento a lo largo de la ejecución de la aplicación, ayuda a potenciar la usabilidad y eficiencia de esta.

Para concluir con Restful API, se puede determinar que ha cambiado la forma en la que las aplicaciones móviles se comunican con los servicios web y los servidores. La gran flexibilidad que tiene y utilización en las aplicaciones ha permitido una mejora en la experiencia del usuario y comunicación fluida entre la aplicación y los recursos necesarios para proporcionar información actualizada. En su totalidad, es una solución confiable y escalable para el intercambio de datos e interacción con servicios web, que, en esta aplicación, permiten una gran eficiencia.

Por último, tenemos la tecnología y entorno de desarrollo integrado (IDE) oficial para la creación de móviles en sistemas operativos Android, Android Studio. Esta compone la totalidad de la aplicación ya que dentro de ese entorno de desarrollo es donde se ha realizado cada una de las clases, interfaces, XML incluyendo las cadenas de caracteres, colores y demás componentes de la aplicación.

La historia de Android Studio [10] se remonta al entorno de desarrollo Eclipse, ampliamente conocido antes de su lanzamiento. Google admitió que se necesitaba una herramienta más especializada para el desarrollo de aplicaciones y funcionalidades Android y presentó en 2013 el entorno Android Studio en una conferencia.

Este entorno de desarrollo está basado en IntelliJ IDEA, otro entorno de desarrollo popular que junta varios lenguajes. Google optimizó este

entorno para darle un nuevo acabado y poder ser utilizado para las funcionalidades necesarias y específicas para el desarrollo de aplicaciones Android.

Android Studio posee una amplia gama de características y funcionalidades diseñadas específicamente para el desarrollo de aplicaciones móviles, las más destacadas son las siguientes y que han sido utilizadas en el desarrollo de esta aplicación [11]:

Primero se trata del editor de código inteligente. Android Studio ofrece un editor de código altamente eficiente y con autocompletado, sugerencias y resalta de sintaxis. Funciones perfectas para el desarrollo de aplicaciones Android ya que como ocurre en este proyecto hay una gran variedad de paquetes, clases, funciones, objetos y el poder tener una función como la mencionada ayuda mucho al desarrollo para organizarse.

La segunda trata de las herramientas de diseño de interfaz de usuario. Android Studio proporciona un editor de diseño virtual que permite a los desarrolladores crear interfaces de usuario atractivas y con muchas funcionalidades como listas, botones, tarjetas o botones. En general proporcionando al desarrollador de crear de manera intuitiva y adaptable interfaces para todos los tipos de dispositivos. Muchas aplicaciones en la actualidad, especialmente las que menos presupuesto tienen, cuando usan otros entornos de desarrollo no especializados, pueden crear interfaces un tanto difíciles de entender y utilizar, pero gracias a utilizar Android Studio se pueden conseguir resultados muy buenos con poco conocimiento y presupuesto.

La tercera se trata de la depuración y perfilado. Android Studio ofrece grandes herramientas de depuración gracias también a su procedencia de IntelliJ IDEA que ya posee grandes herramientas depurativas. De esta forma, además con el perfilado, que permite detectar posibles cuellos de botella y rendimiento de la aplicación, permiten cuidar hasta el más ínfimo detalle de la aplicación para que funcione correctamente en todos los dispositivos móviles.

Una de las funcionalidades más importantes se trata de que su integración con SDK de Android. Esta, permite entrelazar Android Studio con SDK a la perfección por lo tanto se pueden utilizar una inmensa gama de bibliotecas, APIs y herramientas para el desarrollo en Android.

Y a raíz de esta funcionalidad se encuentra la gestión de dependencias, que permiten al desarrollador obtener dependencias de bibliotecas externas para su integración con este sistema. En la aplicación de esta propuesta estas pueden ser como las librerías GSON usadas para la conexión con la base de datos, MPAndroidChart para la visualización de los micronutrientes en una gráfica, entre otras.

En conclusión, podemos ver como Android Studio ha transformado el proceso de desarrollo de aplicaciones Android. Su popularidad y utilización, al igual que su mejora y especialización ha permitido que se convierta en un estándar para la industria, imprescindible si se quiere desarrollar aplicaciones Android de manera gratuita, completa y simple.

Gracias a este entorno de desarrollo, los desarrolladores pueden aprovechar el máximo las funciones y características de la que los sistemas Android son capaces para ofrecer al usuario unas grandes experiencias.

Tras haber comentado cada una de las tecnologías que se van a utilizar y haber mostrado un poco de su historia, se da por concluido el apartado Estado del Arte donde al final hemos podido ver cómo es necesario una aplicación que pueda ayudar a las personas mayores o que tengan alguna degeneración cognitiva. Gracias a usar esas tecnologías, se ha podido alcanzar en gran medida la propuesta seleccionada.

Sin embargo, como se comentará en los resultados, existe grados de mejora en la misma pero que si se desarrollase con más tiempo y con un equipo más grande se podrían solucionar sin problema.

3 Desarrollo

3.1 Estructura de la aplicación

La estructura de la aplicación está dividida en varias secciones, empezando con la arquitectura, el flujo de navegación y por último el diseño de la interfaz de usuario. En los siguientes apartados se desarrollará y explicará la decisión de diseño.

3.1.1 Arquitectura de la aplicación

La arquitectura de la aplicación proporciona una estructura necesaria para el desarrollo de la aplicación. La elegida para esta propuesta se trata del patrón Modelo-Vista-Controlador (MVC) especialmente conveniente para su uso en el desarrollo de aplicaciones Android Studio ya que la conexión con las distintas clases, vistas y controladores es intuitiva y de utilidad.

La propuesta tiene clases que corresponden a los modelos, que representa los datos y la lógica de la aplicación, como el fragmento de la lista de comida o el del formulario, que serán explicados a lo largo de este apartado de desarrollo. A continuación, se encuentran clases vista, que son las diferentes clases XML que muestran información por pantalla y por último las clases controlador, encargadas de coordinar las decisiones del usuario para mandar información entre las clases vista y modelo.

3.1.2 Flujo de navegación de la aplicación

El flujo de navegación comienza con una pantalla de inicio de sesión o registro, si el usuario no ha iniciado sesión anteriormente, en estas pantallas, el usuario puede introducir sus datos para registrarse en el sistema o iniciar sesión.

Una vez que el usuario entre acceda a la aplicación, se encontrará una pantalla principal que corresponde a uno de los fragmentos que se encuentran en la barra lateral correspondiente a un `NavigationView`. En esta barra lateral se encuentra, además del fragmento de la pantalla principal, los distintos fragmentos a los que el usuario puede acceder pulsando sobre los iconos correspondientes, como lista de comida, formulario entre otros.

Este `NavigationView` es el que se encarga de comunicar los distintos fragmentos entre sí. Siendo el principal, como ha sido comentado, la pantalla principal y que, en conjunto con el mencionado, forma una red de fragmentos a los que el usuario puede ir dependiendo del elemento pulsado en la barra lateral.

3.1.3 Diseño de la interfaz de usuario y experiencia de usuario

El diseño de la interfaz de usuario ha sido un aspecto crucial para garantizar que la aplicación pueda ser accesible y fácil de utilizar, especialmente para personas mayores o con degeneración cognitiva.

En primer lugar, se ha tenido un especial cuidado a los botones y textos de los fragmentos ya que se ha tendido a conseguir que los botones sean grandes y fáciles de detectar por sus iconos y los textos con claridad y

legibilidad. Además, se han utilizado colores contrastantes para resaltar elementos importantes y asegurar que sean fácilmente detectables.

Además, la organización y disposición de los elementos en cada fragmento asegura que la experiencia de usuario sea eficiente y fácil de seguir, ya que solo hay elementos que son de utilidad y no pueden llegar a la confusión a lo largo de toda la estructura de la aplicación. Esto ayuda a que los usuarios se familiaricen rápidamente con la aplicación y su navegación.

Por último, se han incorporado elementos interactivos en cada apartado. Elementos que, permiten al usuario interactuar con la aplicación y realizar acciones específicas, como los que se explicarán en los siguientes apartados del desarrollo como, completar formularios, buscar comidas mediante un filtro en una lista, conocer detalladamente los micronutrientes de un alimento, entre otros.

3.1.4 Diseño de alto nivel

A raíz de estas explicaciones se muestra aquí un diagrama de alto nivel de como es la arquitectura básica de la aplicación y que a lo largo de los siguientes puntos se desarrollará con más profundidad:

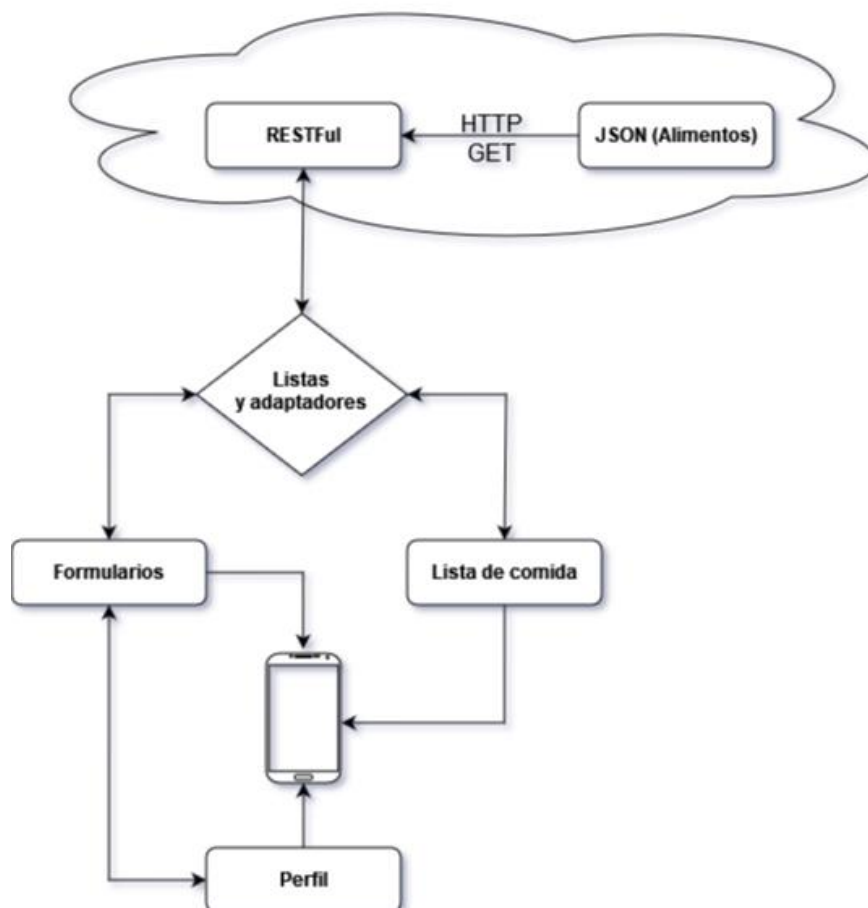


Ilustración 1. Diseño Alto Nivel

En forma de resumen, la API Restful obtiene mediante el método HTTP GET los Alimentos y los almacena en las listas y adaptadores. Después, según lo que se quiera realizar, se puede poblar la lista de comida que

aparece en la interfaz de la aplicación, o se pueden realizar los formularios donde se vuelve a llamar a Restful para obtener los alimentos deseados por un filtro y, por último, mostrarlo por pantalla o dar las recomendaciones en el Perfil.

3.2 Clases y componentes principales

En este apartado se desarrollará las clases y componentes principales que permiten a la aplicación funcionar y comunicarse entre sí con las operaciones del usuario.

3.2.1 Descripción de las clases principales y su funcionalidad

Gracias a que la aplicación y el desarrollo del código se controla mediante la conexión con unas bases de datos, la cantidad de clases objetos se simplifica a tres, las demás siendo controladas dinámicamente o almacenadas para su posterior uso.

Principalmente se encuentra la clase en torno la que gira toda la aplicación, la clase “Food”, simple y pequeña que contiene cada los parámetros que tiene cada comida en la base de datos que se explicará en los siguientes apartados. El primer elemento id de la comida, necesario para el desarrollo y gestión de datos ya que cada método busca la comida o la crea utilizando como clave este id, y a continuación, el nombre y la clase micronutrientes.

La siguiente clase de micronutrientes, es la otra clase principal “Micronutrients”, que contiene en su interior los parámetros de cada uno de los componentes de las comidas, como las proteínas, grasas, hidratos de carbono, etc., que están como objeto parámetro de la clase “Food” y que se completan gracias a la conexión con la base de datos.

Y la última clase, más básica, “Category” utilizada para mostrar la lista de categorías en el apartado explicado más adelante para la funcionalidad de la lista de frecuencias del usuario.

Después, además de las clases de las funcionalidades, se encuentran otras como “FoodListAdapter” o “CategoryListAdapter” necesarias para crear el Adapter que utilizarán las listas donde se mostrarán las comidas, “FoodListConnection” clase que permite la conexión con la base de datos donde se recogen las comidas, “PagingFoodList” clase que crea la funcionalidad de navegar entre páginas en las listas y “MainActivity” clase principal donde se crea la barra lateral NavigationView [12] y se realiza la autenticación del usuario.

Todas las mencionadas serán explicadas en los siguientes apartados donde se hará un desarrollo extenso de sus funciones.

3.2.2 Componentes de la interfaz de usuario y su interacción con las clases

Las interfaces de usuario están compuestas de varios elementos mezclados entre ellos utilizando el layout que se haya considerado el adecuado. Estos elementos son utilizados por la clase correspondiente haciendo llamada a la “View” que se infla con el correspondiente XML.

Aunque en los siguientes apartados se explicará en más profundidad los componentes que son utilizados en cada apartado de la aplicación, cabe destacar de X elementos que han sido utilizados a lo largo de todo el código:

Los elementos que más han sido utilizado se tratan de TextView, componente que permite mostrar al usuario un texto que puede ser modificado o mostrarse por defecto, EditText, franja donde el usuario puede escribir utilizado para el registro, inicio de sesión o filtro, ListView, esencial para poder determinar las listas de comidas y distintos formularios, permitiendo al usuario interactuar sobre cada elemento para dependiendo de la función correspondiente realizar ciertas funcionalidades y por último Button, utilizados para arrancar ciertas funcionalidades de la aplicación.

Cuando el usuario entra en la aplicación, se muestra una simple pantalla de bienvenida que corresponde a la pantalla principal. Sin embargo, la importancia viene en la barra de navegación lateral al desplegarla:

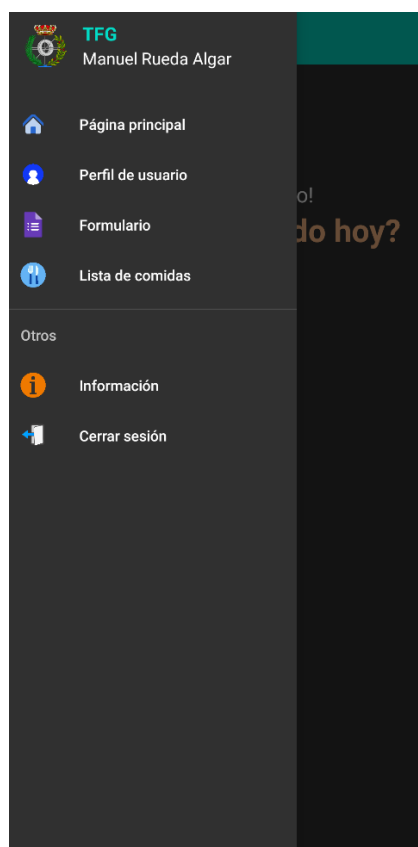


Ilustración 2. Menú lateral

En esta barra lateral, es donde el usuario puede interactuar para moverse entre secciones. Se han elegido unos iconos que ayudan con la aclaración de lo que es cada apartado, pero en conjunto crean una interfaz agradable y fácil de utilizar. Además, en cualquier momento dentro de cada uno de los fragmentos a los que puede ir el usuario, se puede volver a desplegar la barra lateral para ir a otros apartados de la aplicación.

3.3 Conexión con la base de datos

A continuación, adentrándonos en el código de la aplicación, la primera y más importante requerimiento, se trata de la conexión con una base de datos para poder obtener la lista de las comidas.

Antes de poder realizar la conexión con la base de datos es necesario crear un método que permita controlar y estructurar la lista de la comida, es decir, una clase Adapter. Aunque tiene más funcionalidades la clase “FoodListAdapter”, contiene las necesidades básicas para poder manipular una lista de clase “Food”.

Una vez construido este método que a lo largo de la aplicación será utilizado, continuamos con la base de datos mencionada que proviene de una URL. En esta, se encuentra un JSON que contiene un conjunto de alimentos, cada uno identificados con un id de comida para organizarlos, y a continuación, su nombre “nombre_es”, su “group_id” indicando a qué grupo de alimento pertenece y los demás, micronutrientes en una lista que contiene cada alimento. El formato de dicho JSON siendo el siguiente:

```
[{"food_id":"680","nombre_es":"Bacón, ahumado, a la parrilla","nombre_en":"Bacon, smoked, grilled","group_id":"3","micronutrientes":{"proteina_total":"23.40000","carbohidratos":"0.00000","fibra_total":"0.00000","azucares_totales":"0.00000","grasa_total":"22.10000","ag_saturados_total":"8.30000","ag_poliinsaturados_total":"2.90000","ag_monoinsaturados_total":"9.30000","ag_trans_total":null,"colesterol":"80.00000","sodio":"1760.00000","potasio":"330.00000","vitamina_a":"0.00000","vitamina_c":"0.00000","calcio":"7.00000","hierro_total":"0.70000"}}, ...]
```

Para poder realizar una conexión entre la aplicación y la URL donde se encuentra el JSON de cada uno de los alimentos y sus componentes, se ha utilizado Restful API incluido de una clase llamada “FoodListConnection”. La mencionada clase, extiende a una tarea asíncrona cuyo resultado se trata de una lista de objetos tipo “Food”. En el contenido de la clase se encuentra el método “doInBackground()”:

```

@Override
protected Food[] doInBackground(String... urls) {
    URL url;

    try {
        url = new URL(urls[0]);

        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(false);
        int code = conn.getResponseCode();
        if(code == HttpURLConnection.HTTP_OK){
            InputStream is = conn.getInputStream();
            StringBuilder result = new StringBuilder();
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));
            String line;
            while ((line = reader.readLine()) != null) {
                result.append(line);
            }
            reader.close();
            Gson gson = new Gson();
            return gson.fromJson(result.toString(), Food[].class);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

```

Ilustración 3. Código de método Restful

El método realiza una conexión `HttpURLConnection` y crea una petición de tipo “GET” para obtener como resultado la lista. A continuación, mediante `StringBuilder` y unos búferes de lectura se construye la lista de objetos tipo “Food”, gracias al uso de GSON [13], que en la parte final de la clase llama al método “updateData” con la lista resultante, que actualiza el Adapter correspondiente a la clase que llamó a esta clase conexión, por tanto, obteniendo la lista deseada.

3.4 Almacenamiento de datos

El siguiente apartado trata sobre la forma en la que se almacenan los datos de cada usuario, esto incluyendo su formulario de 72 horas, su frecuencia de consumo de alimento y sus datos de inicio de sesión.

3.4.1 Elección de sistema de gestión de datos

Tras comparar las opciones y considerar en usar un sistema de almacenamiento de datos o base de datos local, se consideró el uso del servicio de Google Firebase [14]. Este servicio permite conectar el proyecto de Android Studio con una base de datos gratis y en línea, que permite intercambiar datos de forma dinámica, permitiendo que la poca cantidad de datos que se necesita almacenar se accedan de forma eficiente y sin riesgo de fallo.

3.4.2 Gestión de los datos de usuario y perfiles individuales

Gracias a utilizar Google Firebase Firestore [15], no es necesario almacenar clases adicionales como por ejemplo de Usuario, ya que todo se gestiona de manera online con esta herramienta.

La base principal de la base de datos se trata de un conjunto de usuarios, donde cada usuario al registrarse, como se explicará en el siguiente apartado, mediante un sistema de autenticación de usuario en las clases principales `MainActivity`, `LoginActivity` y `RegisterActivity`, según

genera un UUID aleatorio correspondiente. Y dentro de este UUID, se almacenan los distintos parámetros como su contraseña, correo electrónico y nombre de usuario:

tfmanuel	users	t2hkqlIASwbBEw37Y5hkemd5D4e2
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
users >	t2hkqlIASwbBEw37Y5hkemd5D4	+ Agregar campo
		Contraseña: "123456"
		CorreoElectronico: "pepe@gmail.com"
		NombreUsuario: "Pepe"

Ilustración 4. Estructura base de datos Firebase

De esta manera, cuando el usuario tenga la sesión iniciada en la aplicación, se puede manejar las demás listas que se almacenen dentro de la propia base de datos para ese usuario en concreto, que se añaden y modifican en otras clases, como por ejemplo la lista de las comidas que el usuario ha añadido a su formulario de consumo o frecuencia.

3.5 Funcionalidades

A continuación, se va a desarrollar cada una de las funcionalidades que se esperan de esta aplicación para que cumpla los requisitos de la propuesta que faltan por resolver. Se explicará de igual manera las clases utilizadas al igual que los layout XML y demás usos de la base de datos.

3.5.1 Inicio de sesión y registro

La primera funcionalidad y la que primero se encuentra un nuevo usuario se trata del inicio sesión y registro. Está compuesta por dos clases, “LoginActivity” y “RegisterActivity” como su nombre indica correspondiendo a la actividad de inicio de sesión y registro y cada una con su correspondiente interfaz de usuario. Gracias a Google Firebase, el usuario puede iniciar sesión o registrarse almacenando sus datos en esa base de datos online mediante un sistema de autenticación llamado FirebaseAuth [16].

El sistema de autenticación está inicializado en “MainActivity” como se mencionó anteriormente siendo la actividad que inicia todo y desemboca en las demás. En esta, en el método onStart(), correspondiente al primero que se ejecuta al iniciar la aplicación, se comprueba si hay algún usuario actualmente con la sesión iniciada en ese instante, donde si no es el caso, se inicia “LoginActivity”. Si existe un usuario, directamente inicia la interfaz principal donde se encuentran todos los fragmentos, es decir, dentro de la aplicación. Adicionalmente, en la barra lateral se encuentra un botón de cierre de sesión que termina la sesión con el autenticador.

Dentro de la clase “LoginActivity” se obtiene una instancia de la autenticación Firebase y se añade la funcionalidad a cada uno de los elementos la interfaz. Se ha tomado especial cuidado en que la interfaz gráfica de inicio de sesión y registro sean claros y precios, para ayudar a la usabilidad de la aplicación:

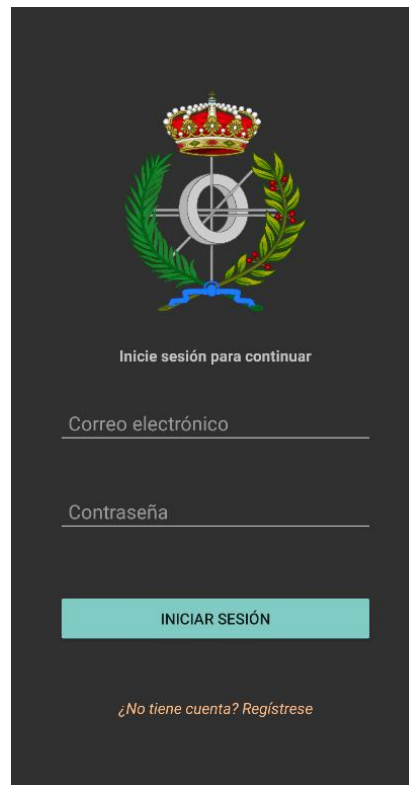
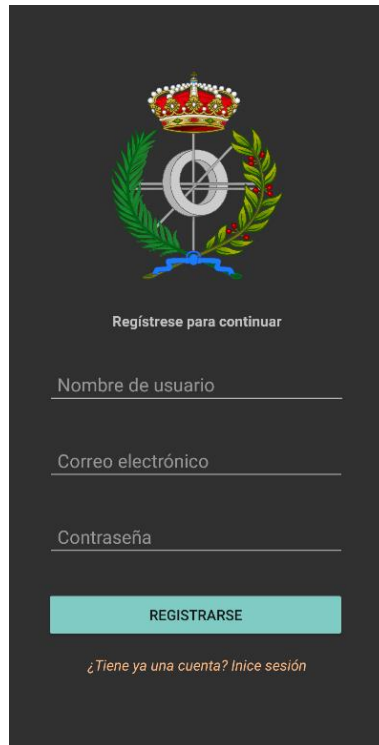


Ilustración 5. Interfaz inicio de sesión

Se encuentran dos EditText, correspondientes al correo electrónico y contraseña, un botón de inicio de sesión y un botón que permite entrelazar “RegisterActivity” permitiendo al usuario registrarse.

En la clase “LoginActivity” se realiza el método login(), que obtiene los valores de dichos EditText, comprueban si están vacíos y si no lo están, al pulsar sobre Iniciar Sesión se procede al inicio de sesión, llamando a un método de Firebase llamado signIn(email, password) que mediante el sistema de autenticación comprueba si el usuario es existente para iniciar sesión o si no, indica un mensaje de error.

Y por último se encuentra la clase “RegisterActivity” donde se realiza el registro. El funcionamiento es similar a la de la clase de inicio de sesión, donde hay parámetros en la interfaz gráfica incluyendo un nombre de usuario que más adelante se mostrará en el perfil de usuario. En esta clase es el primer momento en el usa FirebaseFirestore que como se mencionó es la base de datos de la aplicación.



Regístrese para continuar

Nombre de usuario

Correo electrónico

Contraseña

REGISTRARSE

¿Tiene ya una cuenta? Inicie sesión

Ilustración 6. Interfaz registro

Al igual que en su clase hermana, se toman los valores de cada uno de los EditText y al momento de pulsar el botón de registrarse, primero se comprueban si alguno de esos EditText está vacío. En el caso de que todo esté correcto, se llama a la versión de registro del método de autenticación para el registro en el sistema `createUser(email,password)` y se procede con el registro del usuario en la base de datos.

Una vez que se ha registrado el usuario con la autenticación, se obtiene el Uid generado para el usuario y una referencia a la base de datos “users” donde se encuentra ese usuario. Después, se crea un Map, donde se almacena su nombre de usuario, correo electrónico y contraseña y se actualiza la base de datos.

```

 mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()){
            userID = mAuth.getCurrentUser().getUid();
            DocumentReference documentReference = db.collection( collectionPath: "users").document(userID);

            Map<String, Object> user = new HashMap<>();
            user.put( k: "NombreUsuario", username);
            user.put( k: "CorreoElectronico", email);
            user.put( k: "Contraseña", password);

            documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    Log.d( tag: "TAG", msg: "onSuccess: Datos registrados de " + userID);
                }
            });
            Toast.makeText( context: RegisterActivity.this, text: "Usuario registrado", Toast.LENGTH_SHORT).show();
            startActivity(new Intent( packageContext: RegisterActivity.this, LoginActivity.class));
        } else {
            Toast.makeText( context: RegisterActivity.this, text: "Usuario no registrado", Toast.LENGTH_SHORT).show();
        }
    }
});

```

Ilustración 7. Método autenticación usuario

Tras el registro del usuario correcto, se devuelve al usuario a la “LoginActivity” donde ya podrá iniciar sesión cuando introduzca sus correspondientes valores.

3.5.2 Lista de comida

A continuación, tenemos una de las funcionalidades principales de la aplicación, un fragmento en el cual se muestra una lista con las comidas que se han obtenido mediante la conexión con la base de datos Restful API mencionada anteriormente.

Este fragmento llamado “ListFragment”, está compuesto por varias partes y realiza distintas llamadas las clases fundamentales mencionadas con anterioridad al igual que muchas nuevas.

La funcionalidad del fragmento está vinculada con la interfaz gráfica correspondiente al fragmento:

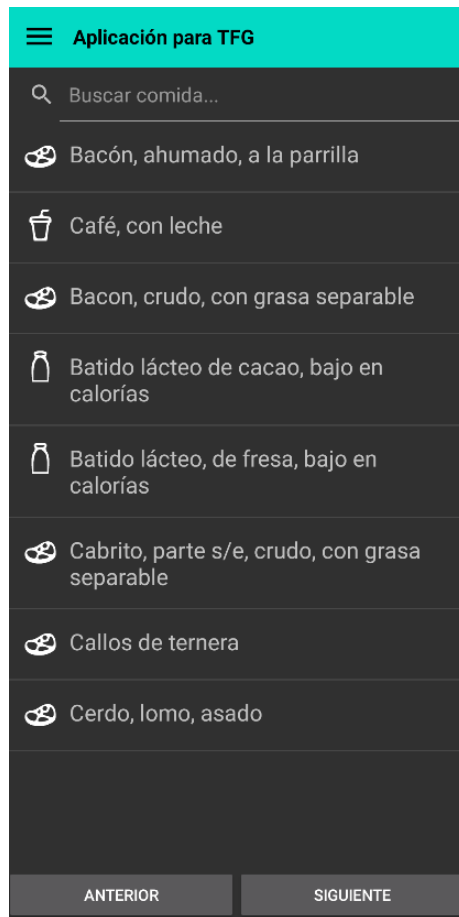


Ilustración 8. Lista de comidas

Visualmente, la interfaz está formada por varios elementos, una barra de búsqueda correspondiente a un `SearchView`, una lista de comida `ListView` y unos botones de anterior y siguiente para pasar a los siguiente 8 elementos de la lista.

Aunque, para que funcione toda esta funcionalidad primero es necesario que se realicen unos procedimientos. Al crearse el fragmento “`ListFragment`”, se inicializan los distintos parámetros a usar como la lista, el `SearchView`, los botones, etc. A continuación, y la parte importante, se crea un nuevo adaptador para la lista, y se realiza la conexión con la base de datos donde se popula la lista.

Para asegurar de que la cantidad de comida mostrada no sobrepase en distintos dispositivos la pantalla en sí, una vez que se ejecuta la conexión con la base de datos y se obtienen las comidas, se ha elegido que se muestren los 8 primeros elementos únicamente, elemento que se pasa como parámetro de creación de objeto para el adaptador. Esto se consigue gracias al método `getView()` del adaptador, que popula la lista con interfaces que contienen un único `TextView` modificado para que cada uno sea el de un objeto `Food`.

```

@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    ViewHolder holder;

    if (convertView == null) {
        convertView = inflater.inflate(R.layout.food_inlist, parent, attachToRoot: false);
        holder = new ViewHolder();
        holder.foodNameTextView = convertView.findViewById(R.id.foodName_inlist);
        holder.iconFood = convertView.findViewById(R.id.foodImage_inlist);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    Food food = getItem(position);
    holder.foodNameTextView.setText(food.getFoodName());
    Long groupId = food.getGroup_id();
    if(groupId==1){
        holder.iconFood.setImageResource(R.drawable.icons8_milk_bottle_32);
    } else if(groupId==2){
        holder.iconFood.setImageResource(R.drawable.icons8_eggs_32);
    } else if(groupId==3){
        holder.iconFood.setImageResource(R.drawable.icons8_meat_32);
    } else if(groupId==4){
        holder.iconFood.setImageResource(R.drawable.icons8_fish_food_32);
    } else if(groupId==5){
        holder.iconFood.setImageResource(R.drawable.icons8_olive_oil_32);
    } else if(groupId==6){
        holder.iconFood.setImageResource(R.drawable.icons8_corn_32);
    } else if(groupId==7){
        holder.iconFood.setImageResource(R.drawable.icons8_seeds_32);
    } else if(groupId==8){
        holder.iconFood.setImageResource(R.drawable.icons8_eggplant_32);
    } else if(groupId==9){
        holder.iconFood.setImageResource(R.drawable.icons8_fruit_32);
    } else if(groupId==10){
        holder.iconFood.setImageResource(R.drawable.icons8_sugar_cube_32);
    } else if(groupId==11){
        holder.iconFood.setImageResource(R.drawable.icons8_soda_32);
    } else if(groupId==12){
        holder.iconFood.setImageResource(R.drawable.icons8_cherry_cheesecake_32);
    }

    return convertView;
}

private static class ViewHolder {
    TextView foodNameTextView;
    ImageView iconFood;
}

```

Ilustración 9. Método para cada elemento de lista

La vista se infla para que utilice el layout “food_inList” que como se ha mencionado solo tiene un TextView que mediante la función getItem() busca en la lista de comidas obtenidas por la conexión con la base de datos cada una de ellas y cambia ese texto por el nombre de la comida, así, creando una lista de 8 elementos.

Adicionalmente y para ayudar a la visualización y uso de la aplicación, las comidas también poseen un icono que puede cambiar entre 12, dependiendo del grupo al que pertenezca la comida, como se puede observar en la figura anterior al código.

Una vez que se ha poblado la lista con cada uno de los elementos, se espera a que el usuario pinche en alguno de los elementos de la interfaz gráfica. Primero, se tiene la barra de búsqueda, que llama a un método de filtrado. Independientemente de que la URL tenga un método de

filtrado a sí mismo, fue implementado con antelación un método manual, y como su funcionalidad es perfecta, sería reutilizar código por tanto se optó por seguir utilizando este método de filtrado manual. El filtrado llama mediante el adaptador de la lista al método “getFilter()” para obtener el filtro y “filter(query)”, query siendo la comida a filtrar una vez que el usuario presione el intro en su teclado.

Dentro de la clase adaptador, se tienen dos listas, una con las comidas y otra, copia inicialmente de la anterior, de la lista de las comidas filtradas. Para realizar el filtro, dentro de la propia clase se crea una clase llamada FoodFilter que extiende a Filter. En ella, según la secuencia de caracteres que reciba, lee la lista de comidas originales y añade las que contengan en su nombre la query a filtrar. Una vez terminada, devuelve la lista filtrada con las comidas que contienen la query, se limpia la lista por si hubiera una interacción anterior y se popula la lista de comidas filtradas actualizando la lista en donde se muestran las comidas terminando con un notifyDataSetChanged() para actualizarla.

```
private class FoodFilter extends Filter {

    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        FilterResults filterResults = new FilterResults();

        if (constraint == null || constraint.length() == 0) {
            filterResults.count = foods.size();
            filterResults.values = foods;
        } else {
            List<Food> filteredList = new ArrayList<>();
            String filterString = constraint.toString().toLowerCase();
            for (Food food : foods) {
                if (food.getFoodName().toLowerCase().contains(filterString)) {
                    filteredList.add(food);
                }
            }
            filterResults.count = filteredList.size();
            filterResults.values = filteredList;
        }
        return filterResults;
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        filteredFoods.clear();
        filteredFoods.addAll((List<Food>) results.values);
        notifyDataSetChanged();
    }
}
```

Ilustración 10. Filtro de lista de comidas

A continuación, si el usuario pulsa los botones de la interfaz gráfica anterior o siguiente, se llaman a los métodos del adaptador para ir a la siguiente página o anterior. Estos métodos, actualizan un valor privado dentro de la propia lista que suma uno o resta uno indicando las siguientes ocho comidas o anteriores. Esto ocurre gracias a que el método getItem() comentado anteriormente, calcula la posición de las comidas a agregar a la lista final mediante un cálculo, todo ello terminando de nuevo en un notifyDataSetChanged() y provocando que las siguientes o anteriores 8 comidas aparezcan en la lista que ve el usuario.

Por último y no menos importante, se trata de la implementación de que el usuario pueda pinchar sobre cualquiera de esas comidas y se entre en una vista extendida donde se pueden observar todos los micronutrientes de la comida seleccionada:



Ilustración 11. Interfaz comida detallada

Esta funcionalidad se activa cuando el usuario pincha sobre un elemento de la lista de la interfaz gráfica. Esto genera un `onItemClick()` con la posición del elemento pinchado que busca mediante `getItem(position)` la comida deseada guardándola en una variable `Food`.

Después, como esa comida está compuesta cada uno de los micronutrientes y parámetros gracias a la conexión con la base de datos cuando se pobló, simplemente se guarda cada uno de los valores de tipo `Doble` de los micronutrientes de la comida y su nombre, para después iniciar la transición al fragmento “`FoodDetailedFragment`” encargado de incluir esa información en la interfaz gráfica mostrada.

Este fragmento se infla con la interfaz “`fragment_food_detailed`” y simplemente sustituye cada uno de los `TextView` que la componen con su parámetro micronutriente que le corresponde, finalmente dando lugar a que se conozcan esos parámetros por pantalla.

3.5.3 Formulario

La siguiente funcionalidad de la aplicación trata sobre unos formularios. El primero de ellos siendo para realizar un control de alimento del usuario para los alimentos que ingiere en un periodo de 72 horas y el segundo registrando la frecuencia de alimentos que el usuario consume durante el tiempo que desee registrarlo.

Cuando el usuario pincha el icono “Formulario” en el menú de navegación lateral, accede al fragmento “FormFragment” donde se encuentra con varios menús en la parte superior correspondiendo con los dos tipos de formularios.

3.5.3.1 Diseño e implementación del formulario para registrar alimentos consumidos en 72 horas

Para el diseño del formulario correspondiente a las 72 horas se ha optado por dividir en tres secciones correspondientes a desayuno, almuerzo y cena, en las que el usuario dependiendo del día que quiera rellenar puede moverse dinámicamente entre días y hacer cambios como eliminar el alimento.

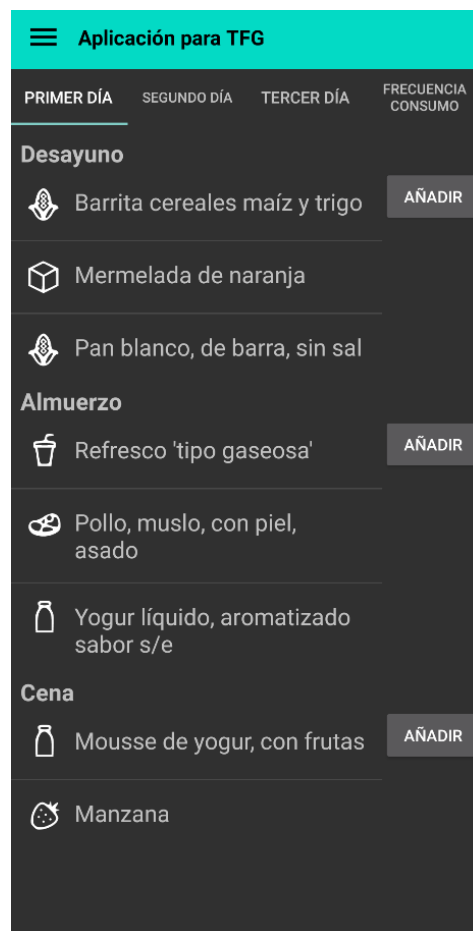


Ilustración 12. Interfaz formulario 72 horas

La interfaz gráfica está compuesta por las secciones mencionadas como TextViews, y debajo de cada una de ellas se encuentra una ListView y un botón. La lista de cada uno de los momentos del día y qué día, donde se muestra los alimentos, está conectada con la base de datos Firebase, ya que desde un principio y cada vez que el usuario accede a este fragmento y se recogen los alimentos de la lista correspondiente, añade o elimina un alimento, esta se actualiza al igual que manda una actualización a la base de datos.

Para manejar cómo interactúa el usuario con cada uno de los días, se ha optado por crear tres clases similares, pero con sus propias listas y conexiones con la base de datos. Tomando como ejemplo la primera,

correspondiente al primer día del formulario, al crear el fragmento, es decir, cuando el usuario se mueve a esta sección de la aplicación, se inicializan los parámetros de las listas y distintos botones y se realiza el método “fetchSelectedItemsFromFirestores()”.

La base de datos Firestore, además de encontrarse con los datos que se explicaron con anterioridad como su contraseña, correo electrónico, etc., cada usuario tiene sus propias listas para cada uno de los momentos del día para los tres días además de su frecuencia.

```
Contrasena: "123456"
CorreoElectronico: "pepe@gmail.com"
NombreUsuario: "Pepe"
▶ frequencyList: {1: {920: {foodId: 920, gr...}}
▼ selectedItemsDinner1Day
  ▼ 920
    amountInMg: 5
    foodId: 920
    foodName: "Mousse de yogur, con frutas"
    groupId: 1
  ▼ micronutrientes
    ag_monoinsaturados_total: null
    ag_poliinsaturados_total: null
    ag_saturados_total: null
    ag_trans_total: null
    azucares_totales: null
    calcio: null
    carbohidratos: 17.7
    colesterol: null
    fibra_total: null
    grasa_total: 6.7
    hierro_total: null
    potasio: null
    proteina_total: 3.3
    sodio: null
```

Ilustración 13. Estructura de formulario diario en Firebase

Cada lista corresponde a un HashMap que contiene las comidas con su id como key. Dentro de cada alimento se encuentra los datos pertenecientes a este, además de incluir el listado de sus micronutrientes, su grupo y la cantidad en mg que se explicará más adelante.

El método mencionado permite popular las listas de forma dinámica para que el usuario siempre tenga actualizado lo que se muestra por pantalla. Para ello, se obtiene un Snapshot de cada una de las listas del día correspondiente de la base de datos, “selectedItemsMorning1Day”, “selectedItemsLunch1Day” y “selectedItemsDinner1Day”, dependiendo de qué fragmento se llame al método será día 1, 2 o 3, correspondiente a las 72 horas. Una vez que se encuentra ese Snapshot, se crean objetos tipo “Food” y “Micronutrients” en un bucle creando cada uno de los alimentos que se encuentran en esta base de datos del usuario, rellenando así las listas que se muestran por pantalla. Al terminar de rellenar las listas con los alimentos, se espera a que el usuario interactúe con la interfaz gráfica.

Cuando el usuario pulsa uno de los tres botones presentes en cada día para añadir comida, se llama al método `showPopupListView(time)`. Gracias a este método, se inicia una ventana popup donde el usuario puede interactuar para buscar en la lista de alimentos de la base de datos Restful API:

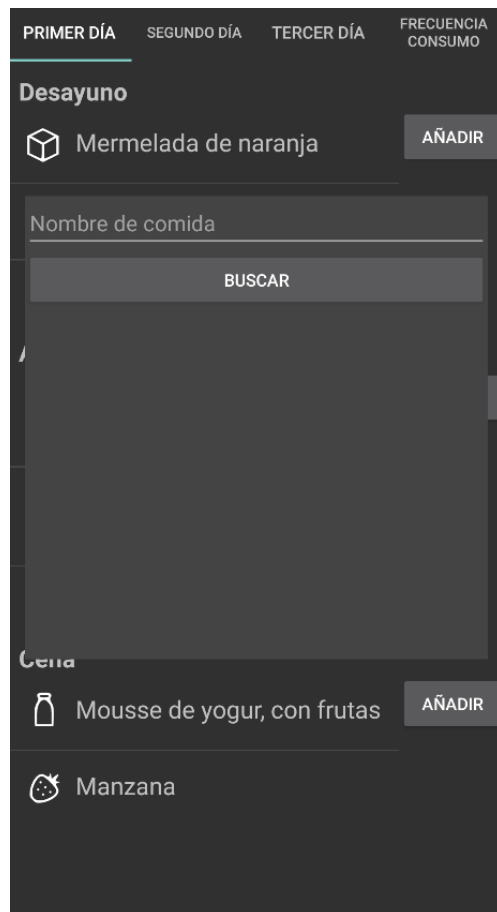


Ilustración 14. Interfaz para añadir alimento

Para crear esta ventana popUp mediante la biblioteca `PopupWindow`, se infla la vista con el layout correspondiente y se obtienen tres componentes, la barra de búsqueda, el botón buscar y una lista debajo del botón. A diferencia del fragmento lista de comida explicado anteriormente donde el filtro de búsqueda se realiza mediante una función del adaptador de la lista, en este caso, cuando el usuario introduce un query en la barra de búsqueda y pulsa el botón buscar, se llama a la conexión Restful API incluyendo en la URL ese query de la manera “.../query”.

Gracias a que se crea un nuevo adaptador para la lista que se encuentra debajo de el botón de búsqueda, se pobla la lista cuando se realiza la conexión y la búsqueda query, actualizándola y mostrándole al usuario los alimentos filtrados de 8 alimentos para no sobrecargar mucho la lista.



Ilustración 15. Resultado búsqueda de comida

Al igual que cada una de las comidas, estas utilizan el mismo adaptador hasta ahora usado, por tanto, tienen su icono correspondiente al grupo de alimentos al que pertenecen. A continuación, cuando el usuario pulsa uno de esos alimentos, aparece otro pequeño popup para indicar al usuario que introduzca los gramos que ha consumido de ese alimento.

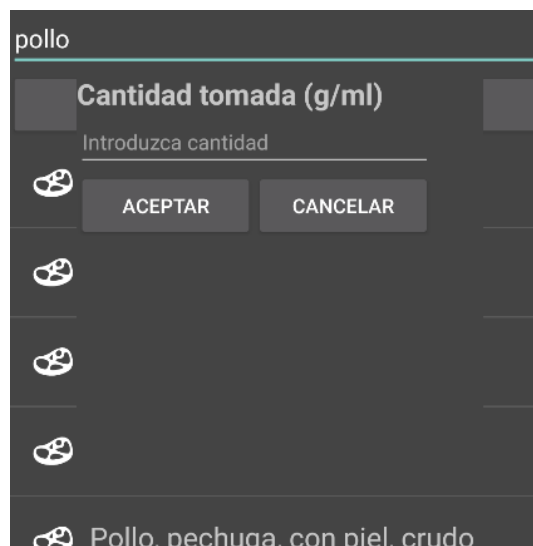


Ilustración 16. Interfaz para introducir cantidad consumida

Aquí, de igual forma, se infla el layout con el correspondiente y se le da al usuario la opción de aceptar, cancelar y rellenar ese TextEdit. Cuando el usuario indica la cantidad y le da a aceptar, porque si le diese a cancelar se cerrarían ambos popups, se llama al método `addSelectedItemToLayout(food, time, quantity)` donde `food` es la comida seleccionada, `time` el tiempo de cual boton pulsó y `quantity` la cantidad que ha introducido en la barra de texto, después de eso cerrando ambos popUps.

Antes de que se cierren, en el método mencionado, se crea la comida que será añadida a la lista de comidas correspondiente a ese momento del día utilizando como comprobante el parámetro `time`. Una vez añadido a la lista y antes de actualizar lo que ve el usuario, se llama al método

`addSelectedItemToFirestore()` con los mismos parámetros que el anterior y como su nombre indica, el método crea el Map del alimento que será introducido en alguna de las tres listas de ese día en la base de datos Firestore. El Map contiene los parámetros de la comida, como nombre, id de grupo, id de comida y micronutrientes, pero, además, contiene uno nuevo que corresponde a la cantidad en Mg que el usuario introdujo en el layout mencionado.

Así, se realiza un update de la lista correspondiente, añadiendo este map nuevo, con el id de comida como key, y se realiza antes de terminar una función que conecta de forma lógica el formulario 72 horas con el formulario de frecuencia, ya que se trata de la función `increaseFoodFrequency(food)`, función que como su nombre indica, aumenta la frecuencia de esa comida en la lista de frecuencias.

Este método, tiene como objetivo dos opciones, que la comida exista ya en la lista de la base de datos Firebase “frequency” o que no exista. Si no existe, se crea el mismo Map de comida, pero, además, se añade el valor frequency a uno, ya que este valor representa la cantidad de veces que se ha consumido ese alimento, y como obviamente no existe, significa que es la primera vez que lo toma.

En el caso de que el alimento ya exista en el map “frequency” de la base de datos, lo que se realiza es obtener el Map del alimento y modificar el valor frequency aumentando su contador en uno por el usuario habiendo consumido ese alimento una vez más. La estructura de la frequency se explicará más adelante ya que no es una simple lista donde se añaden los alimentos porque están divididos por el id de grupo.

Finalmente, al terminar todos estos pasos, se actualiza el adaptador de la lista del momento del día correspondiente y aparece esa comida en la ListView.

Por último, queda la funcionalidad de que ocurre cuando el usuario pulsa uno de esos alimentos de la lista. Al pulsarlos, se obtendría la comida en específico y se procedería a su eliminación. Para poder eliminarlos simplemente, se eliminan de la List correspondiente a esa lista y se actualiza el adaptador donde dinámicamente desaparece ese alimento en vista del usuario notificándolo con lo ocurrido.

Sin embargo, además se tiene que realizar la eliminación de este alimento de la lista de la base de datos Firebase correspondiente, para ello se llama también al método `deleteSelectedItemToFirestore(food, time)`, donde se realiza la contra parte al método mencionado anteriormente que lo añadía, porque se obtiene el Snapshot de la lista dependiendo del día gracias a time donde se tienen a todos los alimentos, se busca a la comida seleccionada y se procede a eliminarla. Además, también es necesario reducir la frecuencia de alimento de la otra lista “frequency”, por lo tanto se llama a `decreaseFoodFrequency(food)` donde se procede a realizar los mismos pasos que para incrementar la frecuencia de comida solo que en este caso, se dan otros dos opciones, si al restar la frecuencia actual con uno, ya que se elimina la comida, da cero, significa que ese alimento ya no debería estar en la lista de frecuencia porque fue la única vez que el

usuario lo consumió, dando lugar a que se elimine por completo ese alimento de la lista. En el otro caso donde la nueva frecuencia tras la resta sea mayor que cero, simplemente se actualiza a la frecuencia del alimento seleccionado reduciendo su valor en uno.

Con esto daría concluido esta primera función de los formularios de 72 horas, pasando ahora a la frecuencia de alimentos que es un tanto similar a las funciones del explicado, pero con variaciones.

3.5.3.2 Diseño e implementación del formulario para registrar frecuencia de alimentos consumidos

La última funcionalidad del fragmento formulario se trata de registrar la frecuencia de consumo de cada usuario para cada grupo de alimentos. Como se ha comentado en el punto anterior, los alimentos de la lista “frequencyList” de la base de datos tienen almacenados además de sus propios parámetros como id de comida, id de grupo, nombre o micronutrientes, en esta lista tienen un parámetro frequency indicando la cantidad de veces que el usuario ha consumido ese alimento.

Para tratar esta funcionalidad se tiene el fragmento “QuantityFormFragment” que como se mencionó anteriormente, se encuentra dentro de “FormFragment” de modo que el usuario mediante las pestañas superiores puede acceder a este mismo.

Ilustración 17. Interfaz formulario frecuencias

La parte de la interfaz gráfica sigue una estructura parecida al formulario de 72 horas, con una lista en pantalla en la que se puede pulsar los elementos y un botón de añadir, sin embargo, para poder realizar una lista fija con cada una de las categorías presentes, fue necesario crear la clase Category y CategoryListAdapter, para adaptarla a la funcionalidad presente.

La clase Category es una simple clase básica, con un id de categoría y un nombre, esta clase es usada para determinar el adaptador de categoría. En el adaptador se tienen las funciones básicas para poder obtener el elemento pulsado en la lista, y la diferencia se encuentra en el getView responsable de como se muestra la lista por pantalla.

Como se crea el adaptador usando una lista de categorías, en el getView se inicializan los parámetros correspondientes al layout category_inlist que tiene una imagen y un textview. Estos parámetros se sustituyen con el objeto Category que ha sido mandado, el TextView del nombre se sobre escribe con el nombre de la categoría y la foto, dependiendo de cuál sea el id del grupo se le asigna uno de los iconos que se muestran por pantalla.

Así, una vez dentro del fragmento de la frecuencia de comida, primero se crea esa lista de categorías con todas ellas, cada una con su id y nombre correspondiente creando así el adaptador que posteriormente pobla la lista mostrada por pantalla. A continuación, el fragmento se queda esperando a que el usuario o pulse uno de los grupos de la lista o el botón de añadir.

Si el usuario pulsa el botón añadir se realiza la misma operación que se realizaba cuando el usuario pulsaba el mismo botón en los apartados del formulario 72 horas. Se crea un popUp donde el usuario con un recuadro de búsqueda y un botón de buscar, si el usuario pone algún texto y lo pulsa el botón aparecen las comidas filtradas mediante la búsqueda en el Restful API y la diferencia viene en cómo se añade ese alimento a qué base de datos y el siguiente popUp. A diferencia del popup que salía en el formulario de 72 horas para añadir los gramos consumidos del alimento, aquí aparece el mismo popUp pero para introducir la frecuencia deseada.



Ilustración 18. Interfaz añadir frecuencia a alimento

Cuando el usuario introduce la frecuencia deseada, si pulsa el botón cancelar se cierran ambos popUp y si pulsa el botón aceptar se realiza el método `addSelectedItemToLayout(food, frequency)`, parecido al usado en el otro formulario, pero con algunos cambios.

En este método, de igual manera se añade a la lista que dinámicamente será reemplazada cada vez que se accede a la categoría correspondiente para ahorrar memoria. Además, se llama a `addSelectedItemtoFirestore(food, frequency)` donde se realiza la misma operación que en su respectiva contraparte en el formulario de 72 horas cambiando que se añade la frecuencia en lugar de los gramos consumidos a la base de datos Firestore en la lista `frequencyList` pero con su grupo dando como resultado la siguiente estructura:

```

▼ frequencyList
  ▼ 1
    ▼ 920
      foodId: 920
      foodName: "Mousse de yogur, con frutas"
      frequency: 1
      groupId: 1
      ▼ micronutrientes
        ag_monoinsaturados_total: null
        ag_poliinsaturados_total: null
        ag_saturados_total: null

```

Ilustración 19. Estructura de lista frecuencia en Firestore

Por otro lugar, si el usuario pulsa una de las categorías presentes en la lista, se llama al método `showFoodsCategoryPopUp(category)`. En el método, se infla la vista con el layout `foodcategory_detailed` en un popUp que contiene un texto superior indicando la categoría pulsada y una lista abajo con los elementos que se encuentran en ella:

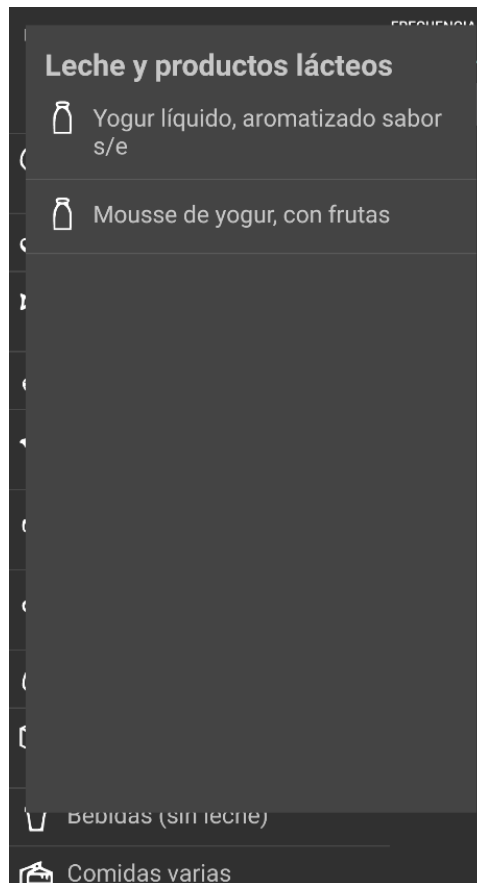


Ilustración 20. Interfaz alimentos por categoría

Para poder mostrar los alimentos correspondientes a la lista de alimentos de ese grupo se reinicia la lista de alimentos por si ya estuviera poblada con otros y se asigna el adaptador de alimentos a esa lista, a continuación, llamando al método `fetchCategoryItemsFromFirestore(category)`.

El método, obtiene un Snapshot de la `frequencyList` de la base de datos y busca por la categoría deseado gracias al parámetro de entrada correspondiente a la categoría que el usuario pulsó. Si esa categoría no existe en la lista porque todavía el usuario no la ha inicializado al no haber añadido nunca una comida perteneciente a ese grupo, se crea la categoría con el `id_categoria` como key. A continuación, se obtiene el listado de alimentos pertenecientes a esa categoría que se guardan en un `HashMap` y se realiza las mismas operaciones que con el método en el fragmento de formulario 72 horas, se obtiene cada uno de los micronutrientes y parámetros de cada alimento y se añade a la lista mencionada anteriormente actualizándola.

En consecuencia, se obtiene una lista con los alimentos pertenecientes a ese grupo de comida en específico y se espera a que el usuario pulse alguno de ellos para poder continuar con la interacción.

Cuando el usuario pulsa un elemento de esa lista aparece otro `popUp` poblando la vista con `food_frequency_detailed`. En este `popUp` se encuentra el nombre de la comida, la frecuencia y dos botones para cambiar la frecuencia y para eliminar la comida. Cuando se accede a este `popUp`, lo primero que se realiza es obtener un Snapshot de la

frequencyList y se busca por el alimento contenido en ella para obtener los datos y actualizarlos en los TextView correspondientes.

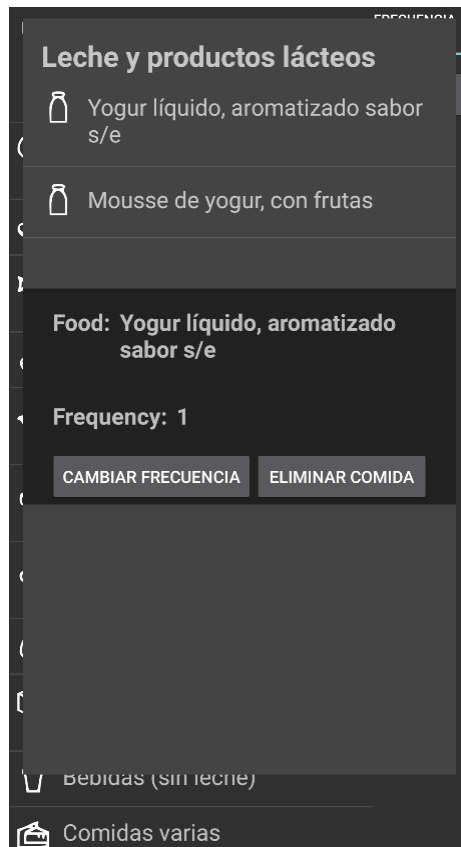


Ilustración 21. Interfaz modificar alimento

Si el usuario pulsa el botón cambiar frecuencia se inicia otro popUp inflando el layout food_inlist_popUp_frequency, el mismo que se utilizó para cuando el usuario quiere añadir una nueva comida a la lista y se le requiere añadir la frecuencia.

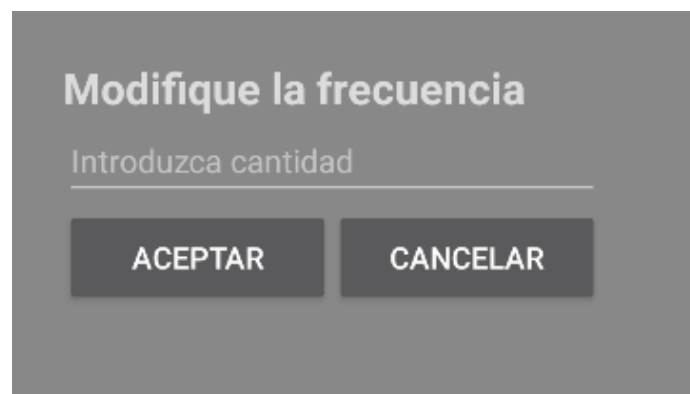


Ilustración 22. Interfaz modificar frecuencia de alimento

El funcionamiento es similar puesto a que cuando se añade una cantidad a la barra de EditText y se pulsa el botón Aceptar, se busca el alimento seleccionado en la frequencyList de la base de datos Firebase y se modifica el parámetro frequency sustituyendo el valor por el que se haya escrito. Para que se actualice dinámicamente, se realiza de nuevo el método addSelectedItemToLayout(food, quantity) para que al pulsar de

nuevo sobre el alimento este esté actualizado. En el caso de que pulse cancelar se cerrarán ambos popUps volviendo a la lista de alimentos de la categoría.

Por último, el usuario puede pulsar el botón eliminar comida que llamaría al método “deleteSelectedItemFromFirestore(food)”, donde como se ha estado viendo a lo largo de la memoria, en este caso busca la comida dentro de la “frequencyList” en la categoría que le corresponda usando su id de categoría y una vez terminado llamando de nuevo al método fetch para actualizar la lista.

Y aquí concluiría el funcionamiento de la aplicación más importante, el de los formularios, permitiendo en resumen actualizar dinámicamente una base de datos donde se almacenan los alimentos consumidos en un periodo de 72 horas y al igual una lista de frecuencias donde se almacenan la cantidad de veces que se ha consumido un alimento.

3.5.4 Perfil de usuario

La última funcionalidad de la aplicación se trata del fragmento de perfil de usuario. En este, se infla la interfaz con el layout fragment_profile que tiene un texto donde se muestra el correo electrónico del usuario, un botón para que el usuario lo pulse y le muestre un quesito con los micronutrientes totales y unos textos de recomendaciones.

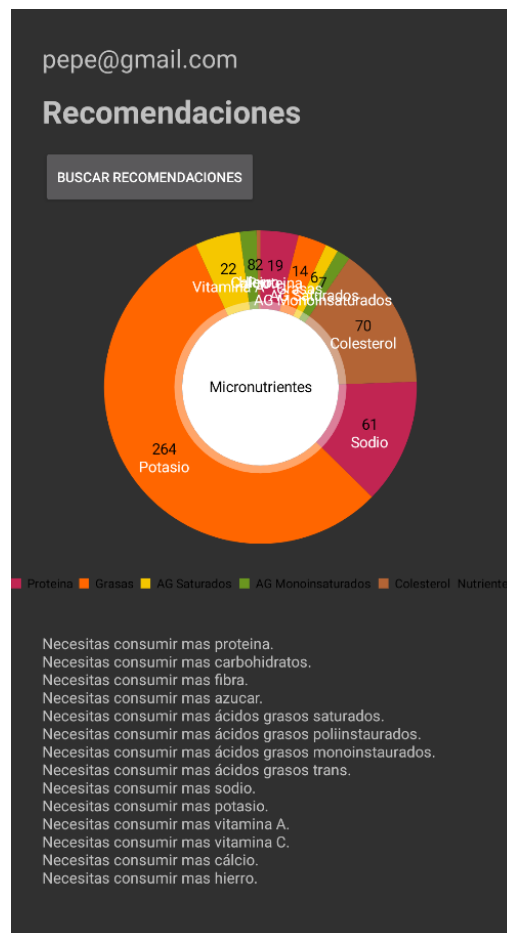


Ilustración 23. Interfaz perfil de usuario

Además de los demás parámetros, en la clase se tiene unos números Double privados que tienen como valor predeterminado 0.0 y corresponden con la suma total de cada uno de los micronutrientes que ha consumido el usuario, siendo modificados más adelante.

Para obtener el correo electrónico y mostrarlo por pantalla se obtiene la instancia de la autenticación del usuario, consiguiendo su correo electrónico y entonces la aplicación se queda esperando a que el usuario pulse el botón Buscar Recomendaciones.

Al pulsarlo, se realizan dos operaciones, la de calcular el total de micronutrientes consumidos por usuario y mostrar por pantalla el gráfico con la información nutricional del usuario. Aunque, de primera mano se realiza la función `calculateTotalNutrients()`, que da como resultado una Task, necesario su uso ya que los fragmentos actúan de forma asíncrona con las llamadas, por tanto, no modificarían el valor de las variables Double correctamente.

Dentro de este método, se crea una lista de Task y una variable para comprobar que no ha ocurrido ningún error en cualquiera de las tareas. La lista es poblada con, de forma ordenada, la llamada al método `calculateNutrients(databaselist)`. Este método, obtiene la Snapshot de la lista de la base de datos Firebase que se haya pasado por parámetro de entrada, y procede a realizar la suma del valor actual de los nutrientes totales Double con cada uno de los micronutrientes que va encontrando en cada uno de los alimentos de esa lista.

Por ejemplo, si el usuario tiene en su lista “selectedItemsLunch1Day”, correspondiente a la lista para el primer día en el almuerzo, un pollo marinado, un vaso de zumo de naranja y un yogurt natural, se hace el sumatorio de cada uno de los micronutrientes de los alimentos mencionados y se almacena el resultado en las variables dobles, así permitiendo su uso para las tablas restantes.

Como hay alimentos que no aportan ciertos micronutrientes, estos aparecen en la lista de micronutrientes como “null” y por ese caso, se comprueba antes de realizar ningún sumatorio si es null o no el valor del micronutriente para así poder asignarle el valor 0.0 o el suyo.

Una vez que se han comprobado todos los alimentos de cada una de las listas del usuario y por obtenido el valor total de cada uno de los micronutrientes, se realiza la parte del gráfico.

Para el gráfico, se ha usado una librería que se expondrá en el punto de librerías usadas proveniente de GitHub. Se llama PieChart [17] y permite añadir de una forma dinámica los elementos y que automáticamente asigne el rango de cada elemento del gráfico adecuado al tamaño que le corresponde.

El uso es simple, ya que se añaden entradas a una lista de PieEntry donde cada una de las PieEntry están compuesta de dos valores, un numero float y el label que se verá en la gráfica. Así, a esa lista se añaden

cada uno de los valores totales de los micronutrientes previamente calculados, transformados en float, y el micronutriente que le corresponde.

Finalmente, mediante un StringBuilder, con unos comprobantes se añaden las necesidades del usuario. Actualmente tienen unos valores predeterminados, pero, si se desarrollase la aplicación más, solo haría falta cambiarlos según el consejo de una persona dedicada a la dieta y con conocimiento alimenticio. Para realizar la función y que se pueda mostrar en la aplicación, se comprueban estos números que corresponden con el límite inferior recomendado en un caso hipotético de cada uno de los micronutrientes con el valor total, y en el caso de que sea inferior se añade un String al String builder que finalmente es el que es mostrado por pantalla.

3.5.5 Información

Fragmento que simplemente muestra el nombre de la aplicación, mi nombre y el de la universidad. Se pretendía hacer un desarrollo más exhaustivo, pero al final se optó por mostrar una información general.

En un futuro como con los demás apartados si se desarrollase la aplicación más, se podrían incluir los correos y teléfonos de los desarrolladores de la aplicación, así como un apartado de preguntas frecuentes, etc.

3.6 Librerías utilizadas

A lo largo de la aplicación se han utiliza diversas librerías, ayudando por igual a layouts o a las clases a permitir la función de cada uno de los componentes. Muchas de ellas son librerías básicas de Java que vienen implementadas simplemente con tener JDK, sin embargo, la gran mayoría son librerías propias de Android.

Estas librerías, permiten conectar layouts con las clases, crear cualquier componente de las interfaces, permitir la conexión con las bases de datos, permitir la autenticación del usuario, entre otras.

La principal fuente de las librerías se trata de la construcción de la aplicación, donde se implementan y se descargan estas mismas para poder ser utilizadas. A continuación, se muestra un listado de las dependencias utilizadas a lo largo de toda la aplicación:

- “androidx.core:core-ktx:1.7.0”: implementación que se refiere a la biblioteca Core-KTX de AndroidX para Kotlin. AndroidX es un conjunto de librerías de soporte que proporcionan funciones y componentes adicionales para facilitar el desarrollo de la aplicación. Las funcionalidades tienden a ayudar a la interacción con las API Android, como las vistas, recursos, fragmentos y preferencias compartidas.
- “androidx.appcompat:appcompat:1.6.1”: implementación de la biblioteca de soporte AndroidX que permite que las aplicaciones se ejecuten en versiones antiguas de Android sin perder compatibilidad con las características más recientes. Proporciona

widgets y estilos compatibles con versiones antiguas y nuevas de Android, lo que facilita el diseño y el desarrollo de aplicaciones en distintos dispositivos.

- “com.google.android.material:material:1.7.0”: implementación de componentes de Material Design de Google. Proporciona una amplia gama de elementos de interfaz de usuario como botones, barras de herramientas, tarjetas y más, permitiendo la creación de aplicaciones con apariencia moderna e intuitiva.
- “com.google.firebase:firebase-auth:21.2.0”: librería Firebase de Google que ofrece servicios de autenticación para aplicaciones Android. Permite realizar funciones de autenticación con proveedores como Google, Facebook, correo electrónico, registro de usuarios, contraseñas, etc.
- “com.google.firebase:firebase-firestore:24.4.5”: librería Firebase que proporciona una base de datos en tiempo real y en la nube para aplicaciones Android. Permite almacenar, recuperar, modificar e sincronizar datos estructurados en distintos dispositivos.
- “androidx.navigation:navigation-fragment-ktx:2.5.3”: componente de navegación de AndroidX, que facilita implementación entre la navegación de fragmentos en una aplicación Android. Gestiona las rutas de navegación, incluida la transición entre fragmentos y la manipulación del historial de navegación.
- “androidx.navigation:navigation-ui-ktx:2.5.3”: componente que se enfoca en la integración de la navegación con la interfaz de usuario. Proporciona funciones para trabajar con elementos de la interfaz relacionados con la navegación, configuración de menús de navegación, sincronización de barra de acciones y animaciones.
- “androidx.navigation:navigation-compose:2.5.3”: componente de navegación diseñada para el uso de Jetpack Compose, permitiendo la navegación entre pantallas de aplicaciones Compose.
- “androidx.navigation:navigation-dynamic-features-fragment:2.5.3”: componente que permite dividir la carga y navegación de características dinámicas en una aplicación.
- “androidx.navigation:navigation-testing:2.5.3”: componente que facilita pruebas unitarias y integración relacionadas con la navegación en una aplicación.
- “androidx.paging:paging-runtime:3.1.0”: componente que permite cargar y mostrar datos de manera incremental mientras el usuario se desplaza por una lista o una vista.
- “com.google.code.gson:gson:2.6.2”: proporciona funciones para serializar y de serializar objetos Java con formato JSON. Permite la conversión de objetos Java en representaciones JSON y viceversa, útil para los modelos de conexión.

- “com.github.PhilJay:MPAndroidChart:v3.1.0”: librería de gráficos para aplicaciones Android. Proporciona una amplia gama de gráficos interactivos, en esta aplicación se utiliza el gráfico del quesito, de forma atractiva y útil.

Estas implementaciones permiten a la mayoría de las clases desarrollar las distintas librerías que se utilizan. Las librerías están divididas en distintas secciones, desde la interfaz gráfica, contexto, gráficos, etc.:

Contexto y vistas

- Context: proporciona acceso a información global sobre la aplicación y el entorno en tiempo de ejecución.
- LayoutInflater: permite inflar vistas a partir de archivos XML en Android.
- View: representa una vista en la interfaz de usuario de Android.
- ViewGroup: contenedor que agrupa y organiza vistas hijas en la interfaz de usuario Android.
- ImageView: componente de interfaz de usuario utilizado para mostrar imágenes en la aplicación Android.
- TextView: componente de interfaz de usuario utilizado para mostrar texto en la aplicación Android.
- Button: componente de interfaz de usuario utilizado para realizar acciones o enviar eventos en la aplicación Android.
- EditText: componente de interfaz de usuario que permite a los usuarios ingresar y editar textos en la aplicación Android.
- Toast: muestra mensajes breves y temporales en forma de notificaciones emergentes en la aplicación.

Adaptadores y filtros

- BaseAdapter: clase base abstracta que proporciona una estructura para la creación de adaptadores personalizados en Android para llenar vistas como ListView en este caso con datos.
- Filter: clase utilizada para implementar la funcionalidad de filtrado en las listas o conjuntos de datos de la aplicación.
- Filterable: interfaz que indica que una clase puede ser filtrada mediante el uso de un objeto Filter en la aplicación.

Colecciones y datos

- ArrayList: implementa una lista dinámica para almacenar elementos.
- List: interfaz que define la estructura de una lista permitiendo el almacenamiento y manipulación de elementos.
- Arrays: proporciona métodos y utilidades para trabajar con arreglos estáticos.
- HashMap: implementa la interfaz Map y proporciona una estructura de datos para almacenar pares clave-valor.
- Locale: representa la configuración regional o idioma específico.
- Map: interfaz que define colección de pares clave-valor.

Entrada y salida de datos

- BufferedReader: permite leer datos de una fuente de entrada como un archivo o flujo de caracteres, utilizado para leer los datos de la conexión Restful API.
- InputStream: proporciona un flujo de entrada para leer bytes de una fuente como archivo o una conexión de red.
- InputStreamReader: actúa como puente entre flujo de entrada de bytes y un flujo de entrada de caracteres.

Conexión y redes

- WeakReference: permite mantener una referencia débil a un objeto, donde puede ser recolectado y evitar la retención de memoria innecesaria.
- URLConnection: proporciona una interfaz para realizar solicitudes HTTP a un servidor, estableciendo conexiones, enviando solicitudes y recibiendo respuestas de los protocolos correspondientes.
- URL: representa una dirección URL para manipular los datos procedentes de los distintos protocolos.

Componentes de navegación

- NavController: proporciona la navegación entre fragmentos y actividades en una aplicación.
- NavHostFragment: actúa como un contenedor para alojar y mostrar fragmentos de navegación en una interfaz de usuario.
- NavigationUI: proporciona métodos y utilidades para simplificar la integración de navegación de interfaz de usuario. Incluye elementos como barra de acción, menú de opciones entre otros.
- NavigationView: proporciona una vista de navegación lateral que se utiliza comúnmente en patrones de diseño de aplicaciones con un menú de navegación desplegable.

Autenticación y Firebase

- FirebaseAuth: proporciona la funcionalidad para autenticar usuarios en una aplicación móvil con tareas de autenticación como registro de usuarios, inicio de sesión o cierre de sesión.
- FirebaseUser: representa al usuario autenticado en una aplicación proporcionando información como su ID único, nombre, correo, contraseña o correo electrónico.
- AuthResult: representa el resultado de una operación de autenticación exitosa.
- DocumentReference: representa una referencia a un documento específico en una colección, utilizado para realizar operaciones de lectura o escritura en la base de datos de Firestore.
- FirebaseFirestore: representa la conexión con la base de datos en línea Firestore.
- DocumentSnapshot: representa la referencia a un documento para obtener el listado de su contenido y poder actualizarlo o modificarlo.

Registro y depuración

- Log: permite imprimir mensajes de registro y depurar problemas durante el desarrollo de la aplicación.

Persistencia y serialización

- GSON: permite convertir objetos Java en formato JSON y viceversa para almacenamiento y comunicación de datos.

Componentes de interfaz de usuario

- AppCompatActivity: proporciona una base para crear actividades compatibles con versiones antiguas de Android.
- GravityCompat: proporciona métodos compatibles con versiones antiguas para el manejo de gravedad en componentes de interfaz de usuario.
- DrawerLayout: crea un panel deslizable en la interfaz de usuario para mostrar opciones de navegación.
- TabLayout: crea pestañas en la interfaz de usuario para la navegación entre diferentes secciones de contenido.

Intenciones y fragmentos

- Intent: permite el intercambio de datos entre actividades o el inicio de actividades y servicios.
- Bundle: permite almacenar y pasar datos en forma de pares clave-valor entre componentes de la aplicación.
- FragmentManager: gestiona la interacción entre fragmentos y permite realizar transacciones entre ellos.
- FragmentPagerAdapter: permite mostrar fragmentos en una vista de páginas para facilitar la navegación.

Componentes de paginación

- PagingSource: proporciona datos paginados de una fuente, como una base de datos o una API.
- PagingState: contiene información sobre el estado de paginación, como la posición actual y los elementos cargados.

Gráficos

- PieChart: representa datos en forma de gráfico circular.
- PieData: contiene los conjuntos de datos y la configuración para el gráfico circular.
- PieDataSet: define los datos y la apariencia de un conjunto de datos en el gráfico circular.
- PieEntry: represente una entrada en el gráfico circular con su valor y etiqueta asociados.
- ColorTemplate: proporciona una selección de paletas de colores predefinidas para utilizar en gráficos.

4 Resultados y conclusiones

Terminando con el desarrollo de la aplicación y el funcionamiento de cada una de las implementaciones, se van a exponer a continuación los resultados y conclusiones que se han obtenido a lo largo de los meses de desarrollo del trabajo fin de grado de la propuesta.

4.1 Análisis de los resultados

Comenzando con el primero apartado, se va a analizar los resultados obtenidos para cada uno de los elementos de la aplicación que tienen algún funcionamiento y han sido desarrollados siguiendo las pautas y objetivos planificados.

4.1.1 Resultados obtenidos en la implementación de la estructura de la aplicación

La estructura es un apartado que ha sido relativamente fácil y sin muchos problemas de realizar a lo largo del proyecto, ya que como se ha utilizado Android Studio para el desarrollo, esta herramienta posee todas las necesidades requeridas y las maneja de manera automática.

Para comenzar, la arquitectura modelo-vista-controlador está totalmente manejada por Android Studio, donde lo único necesario es crear las clases que sean requeridas e implementarlas en donde estén necesitadas. Esto facilita el desarrollo, ya que, en muchos otros proyectos realizados previamente, estos modelos fueron necesarios estructurarlos de manera manual y supusieron un reto más significativo.

Al igual ocurre con la navegación del usuario con las interacciones. Al usar fragmentos y otros métodos de navegación, la usabilidad y dinamismo de la aplicación gracias a herramientas propias de Android Studio permiten que esta sea agradable de usar y no constituya a que el usuario se pierda o pulse algún lugar donde no debería, ya que todo está conectado entre sí.

Y, por último, la interfaz de usuario que engloba los dos puntos anteriores. Por consecuencia, este apartado también se gestiona de una manera adecuada y suficiente para la usabilidad del usuario.

En torno a la estructura de la aplicación, el desarrollo ha ido sin problemas y con alto grado de satisfacción, sin embargo, cabe destacar que, aunque la interfaz gráfica ha tenido un grado más de complejidad. Al tener que realizar manualmente cada una de las interfaces de la aplicación, cada una teniendo sus propios elementos y estructura, ha sido un reto por conseguir debido a la gran cantidad presentes en este proyecto.

Por lo general, las interfaces cumplen con su función y son agradables de usar y a la vista, aunque, si se dedicase más tiempo y desarrollo en el caso de que acabara siendo una aplicación con más meses de desarrollo, podría alcanzar un mejor grado de satisfacción para el usuario añadiendo decoraciones y otros elementos que la puedan hacer más funcional.

4.1.2 Resultados obtenidos en la conexión con la base de datos

La conexión con la base de datos ha sido un éxito rotundo gracias a que utilizando las librerías que se han mencionado en el apartado anterior, se logra una conexión simple y efectiva con la base de datos donde se encuentran las comidas.

Cuando se tiene la conexión simplemente se utiliza otra de las librerías para transformar ese JSON en los objetos determinados en el proyecto y finalmente están listos para su uso en la aplicación.

En este apartado no tengo ningún punto que pueda mejorarse ya que la conexión es simple, rápida y segura permitiendo a cualquier usuario de manera eficiente conectarse a esta base de datos, descargarse los alimentos que incluso pueden actualizarse para añadir nuevos o modificar alguno y lo tiene listo para su uso.

4.1.3 Resultados obtenidos en el almacenamiento y gestión de datos

Gracias a haber utilizado Google Firebase para el manejo de la gestión de datos, podemos concluir que el sistema es seguro, confiable y dinámico. Cada usuario tiene su propio UUID con sus parámetros y listas permitiendo que no exista la posibilidad de fallo por duplicación de UUIDs, o algún otro dato, y, además, la estructura de las listas que se almacenan es simples ya que son HashMaps y List por tanto hay grado de confianza en su usabilidad y confiabilidad.

Además de las listas mencionadas de la base de datos de Google hay otras listas que no requieren estar conectadas por internet ya que se manejan de manera local, como por ejemplo las listas de filtrados o las propias listas que se descargan tras la conexión con la base de datos comentada en el punto anterior. Estas listas, son de igual forma simples, eficientes ya que son implementaciones de librerías propias de Java que son manejadas de manera automática y se ha tenido especial cuidado en que no puedan darse fallos en su uso, dándole la seguridad al usuario que todo lo que interactúa y se muestra está actualizado y es confiable.

Por consecuente, cuando el usuario entra de nuevo de la aplicación o solicita rellenar estas listas para que las vea en la interfaz, la operación consiste simplemente, como se ha explicado en el desarrollo, de tomar los elementos buscados en las listas de Google Firebase, recogerlas y actualizar las locales sin posibilidad de fallo aparentemente.

Cabe destacar que, en un principio con la incertidumbre de no conocer el servicio Google Firebase y tener que ir conociéndolo a medida que el proyecto se ha ido realizándolo, ha supuesto cierto reto el conocer cómo se estructuraban internamente las listas y los datos de los usuarios. Por ello, se han tenido que utilizar herramientas y librerías como LOGs entre otras para depurar las listas, ver por qué no se agregaban a la gestión de datos Firebase, detectar problemas con las clases entre otras, pero que a lo largo del desarrollo del proyecto se han ido solucionando.

4.1.4 Resultados obtenidos en la implementación de las funcionalidades

Y para terminar con los resultados queda por hablar sobre la implementación de las funcionalidades. En este como se ha comentado a lo largo del proyecto, tenía tres funcionalidades principales, los formularios, siendo los de 72 horas y frecuencia de consumo, la lista de los alimentos, mostrando detalladamente los micronutrientes y por último las recomendaciones en el perfil de usuario.

Empezando con los formularios, se puede decir que se ha cumplido con creces lo que se requería de ellos, ya que el usuario puede, añadir alimentos consumidos en ciertas horas del día durante tres días, permitiendo al usuario eliminarlos en caso de equivocación y añadir los gramos determinados. Además, el formulario de los 3 días estando conectado con la frecuencia del usuario, donde se añade dinámicamente a esta lista los que consume el usuario durante los días, dándole además la posibilidad de modificar a mano cualquiera de las frecuencias de los alimentos, eliminarlos y añadir otros que no estén en las otras listas.

La siguiente funcionalidad que muestra una lista de todos los alimentos al usuario, funciona a la perfección ya que gracias a la conexión con la base de datos anteriormente mencionada le permite al usuario ver la lista de comidas, filtrarlas y ver de manera extendida los componentes del alimento seleccionado.

Y por último la funcionalidad del perfil de usuario, que, gracias a la librería mencionada en el punto anterior para representar valores en un gráfico, permite al usuario ver una idea general de los micronutrientes que ha consumido en ese periodo de 72 horas y le muestra unas recomendaciones en torno a esos mismos valores.

En general, se observa un resultado positivo ya que se han podido realizar todas las funcionalidades, permitiendo al usuario poder utilizar la aplicación de una manera retroalimentativa. El reto de crear cada una de las funcionalidades y lo que conlleva en cuanto crear clases, utilizar componentes y conectarlos ha sido un gran reto, el que más tiempo ha llevado. Sin embargo, se ha podido resolver los problemas y funciona todo a la perfección.

Cabe destacar, que la funcionalidad del perfil de usuario es la más pobre comparada con los formularios y la lista de comida, ya que para las recomendaciones se necesitan unos valores nutricionales mínimos y máximos, que pueden variar entre persona y ahora únicamente muestra unas recomendaciones generales. Si se desarrollase la aplicación más y se alcanzara ayuda de profesionales esa sección de recomendaciones quedaría más completa y útil.

4.2 Evaluación del cumplimiento de objetivos

En conclusión y tras observar los resultados de cada implementación, se va a realizar una recapitulación de los objetivos que se indicaron al principio del trabajo fin de grado, y que han sido cumplidos cada uno.

El principal objetivo que trataba sobre realizar la implementación de una aplicación móvil para la recogida de hábitos alimenticios estaba dividido en tres subobjetivos. Los subobjetivos comenzaban en realizar una interconexión con un servicio web de los micronutrientes presentes en un alimento, realizado con éxito ya que gracias a esa conexión se obtienen los alimentos que se utilizan en las distintas clases y funcionalidades.

El siguiente subobjetivo trataba sobre desarrollar los métodos y servicios necesarios para que se puedan calcular los valores nutricionales de la dieta de un usuario. Gracias a la implementación de los formularios de 72 horas y la conexión con ambas bases de datos, se cumple el objetivo ya que el usuario como se ha ido explicando a lo largo de la memoria, añade los alimentos consumidos en el momento en específico del día y se almacena en la base de datos. A continuación, en el perfil de usuario se calculan los valores nutricionales totales que el usuario ha incluido en sus formularios y gracias a ellos se pueden realizar las recomendaciones.

Y, por último, el subobjetivo desplegar el sistema tras haber realizado pruebas. Cumplido con creces debido a que se le permite al usuario usar la aplicación, moverse entre fragmentos, entre menús de la barra lateral, utilizar las funciones, realizar inicio de sesión y registro.

Como se han cumplido los tres subobjetivos, podemos entonces dar por conclusión que las propuestas y objetivo principal del trabajo fin de grado ha sido satisfecho, con cierto grado de mejora como se han explicado en los resultados, pero en general, correspondiente a lo requerido.

4.3 Limitaciones y posibles mejoras

A continuación del último párrafo comentado, cabe destacar unas posibles mejoras del sistema que podrían aumentar su usabilidad y utilidad para un futuro desarrollo.

La limitación principal del proyecto se trata de la longitud y anchura de las interfaces o pantalla de las funcionalidades. Al haber sido desarrollada la aplicación en un mismo simulador de teléfono Android con un tamaño específico, si la aplicación se llegase a probar en otros dispositivos más pequeños puede ocurrir que algunos apartados no se adapten correctamente a esa pantalla. Por ello, un cambio posible sería realizar unas adaptaciones que permitan que funcionen en toda la gama de dispositivos móviles asignando valores específicos a las ventanas e interfaces dependiendo del tamaño del dispositivo o modelo.

Por lo tanto, se podría comenzar hablando sobre la interfaz gráfica para las mejoras, que, aunque se ha conseguido realizar una interfaz agradable y fácil de usar para el usuario, tiene la apariencia de una aplicación por defecto. Si se dedicase un tiempo exhaustivo al desarrollo

de una apariencia más moderna se podría mejorar lo que el usuario ve al utilizarla. Sin embargo, para las funcionalidades, se considera que está adecuada a las necesidades y no necesitaría cambios.

Y, por último, una mejora que considero la más necesaria y que ha sido comentada en el apartado anterior, se trata de mejorar la recomendación de la nutrición. Ahora, las recomendaciones son generales, con unos valores predeterminados para todos los usuarios. Si se quisiese lanzar esta aplicación a todo el mundo y que mejorase su utilidad, se podría implementar un sistema en el que se recogiesen las necesidades del usuario en específico para así poder darle una recomendación de micronutrientes más adecuadas.

En resumen, podrían mejorarse apartados de la interfaz gráfica y de la funcionalidad del perfil de usuario, sin embargo, en general se cumplen con los requisitos de los objetivos y de lo requeridos.

4.4 Conclusiones personales

Y, para terminar, pasando a hablar en primera persona, me cabe destacar que este desarrollo de aplicación y trabajo me ha permitido alcanzar unos grandes conocimientos de uso en Android. Originalmente, este trabajo de fin de grado lo quería hacer porque me interesaba Android debido a la asignatura “Programming for Mobile Devices” de la cual mi tutor es profesor. Gracias a todas las dificultades que he tenido a lo largo del proyecto, como no saber cómo utilizar una librería, como no saber cómo continuar en desarrollar las funcionalidades, cómo poder solucionar unos errores de las interfaces e investigar sobre estos problemas, he podido alcanzar un desarrollo de mi conocimiento general sobre el desarrollo de aplicaciones Android en un gran grado.

Gracias a este proyecto sé que Android es un sector del mercado en el que estoy interesado, y que podría centrarme en un futuro, especialmente conociendo la obvia popularidad de las aplicaciones de estos dispositivos. Por lo tanto, estoy agradecido con haber tomado esta propuesta de trabajo y haberla desarrollado.

5 Análisis de Impacto

Gracias al desarrollo de esta aplicación móvil, donde recogemos los hábitos de vida y alimentación de personas mayores o con alguna degeneración cognitiva, se puede observar que hay un gran impacto en la salud y el bienestar de los usuarios.

En primer lugar, se puede esperar un impacto positivo en la salud de los usuarios. La aplicación da la oportunidad a las personas mayores y con alguna degeneración cognitiva de acceder a una herramienta y aplicación fácil de usar para registrar su dieta y analizarla. Al registrar el tipo y cantidad de alimentos consumidos en cada comida, permite al usuario conocer la ingesta de micronutrientes, llevándolos a una mayor conciencia sobre su alimentación.

Además, la aplicación da una independencia a los usuarios gracias a la autonomía. Al proporcionarles la capacidad de planificar y preparar comidas saludables, permite a las personas mayores y usuarios a mantenerse activas en su hogar y tomar decisiones informadas sobre su alimentación. Permitiendo, por tanto, tener un impacto en la autoestima y confianza de ellos mismos, ya que pueden sentirse capaces de cuidar su salud de manera autónoma.

Por último, e independientemente del impacto en la salud, la aplicación tiene un efecto social y emocional importante. Al facilitar la participación de los cuidadores de las personas mayores o familiares en el seguimiento de la dieta y hábitos de vida, se fortalecen esos lazos entre ellos. Y por consecuencia, reduce el aislamiento social al dar una oportunidad que fomenta la comunicación y el apoyo.

Por lo tanto y, en conclusión, podemos ver que la aplicación no solo tiene un impacto significativo en la salud y autonomía, sino que también en el bienestar social y emocional de las personas mayores y otros usuarios que la utilicen. Los resultados mostrados en el punto anterior respaldan estos puntos ya que la utilidad y eficacia de la aplicación es un punto fuerte de la misma. Sin embargo, y como es de esperar y se ha comentado, si el desarrollo de la aplicación en un futuro siguiese, se podría aún más mejorar su impacto en este grupo demográfico vulnerable.

6 Agradecimientos

Para terminar con el trabajo de fin de grado, quería dedicar una pequeña sección más personal para mis cercanos, ya que acaba mi etapa de universidad.

Desde pequeño, gracias a mi padre, me han interesado los ordenadores, los videojuegos y ha sido el principal responsable de que acabase estudiando esta interesante carrera. Adicionalmente, gracias al apoyo incondicional de mis familiares, en especial mi madre, que, aunque hemos vivido situaciones familiares bastante desgraciadas, me ha permitido conocer a una mujer que no para de trabajar y querer sacar todo adelante, siendo la referente en mi vida, ayudándome en todo lo que sea necesario y sacrificándose para que pueda estudiar y aprender todo lo que necesite en mi futuro.

Estoy completamente seguro, de que, si no fuera por mi madre, no hubiera sido capaz de sacar adelante esta carrera, que está a punto de acabar. De igual forma, mis familiares por parte de padre y madre, mis tíos, primos, primas y abuelos y también mis amigos, todos me han dado fuerza para, aunque no todas las asignaturas sean de mi agrado y no tenga siempre las ganas de estudiar, el ponerme y tratar de sacar todo a la primera sin tener que darles ninguna preocupación.

Mis formas de actuar son calladas, introvertidas y trato de ser humilde en no fardar de mis logros, pero, aunque no lo mencione mucho por palabras, el amor incondicional que tengo para ellos es algo tan profundo y fuerte que no va a cambiar en mi vida. Gracias a ellos, en especial a mi madre, soy la persona que soy y aunque puedo mejorar muchos aspectos de mí, la parte que verdaderamente importa, es decir, la interna, estoy orgulloso porque se cómo soy y mis cercanos saben cómo soy.

Así, que solamente puedo acabar esta pequeña sección de agradecimiento y dando por concluido el trabajo de fin de grado con unas pequeñas palabras pero que todo lo engloban, de la parte más profunda de mi corazón, gracias por todo, os quiero.

7 Bibliografia

- [1] [World population prospects 2019 - Data booklet, United Nations, Department of Economic and social affairs](#)
- [2] [Ageing and health, World Health Organization](#)
- [3] [MyFitnessPal, MyFitnessPal, Inc.](#)
- [4] [Lose it! FitNow, Inc.](#)
- [5] [MealLogger, Wellness Foundry](#)
- [6] [Lifesum, Lifesum AB](#)
- [7] [JSON](#)
- [8] [JSON Usage](#)
- [9] [Restful](#)
- [10] [Android Studio History](#)
- [11] [Android Studio Documentation](#)
- [12] [Android Developers Guide](#)
- [13] [GSON Documentation](#)
- [14] [Google Firebase](#)
- [15] [Firebase Firestore](#)
- [16] [Firebase Authentication](#)
- [17] [MPAndroidChart Github](#)

Este documento esta firmado por



Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Tue May 30 19:22:54 CEST 2023
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)