CONSEJERÍA DE EDUCACIÓN
Comunidad de Madrid

IES ENRIQUE TIERNO GALVAN

Parla

CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Curso 2021/2022

Proyecto DAM

TITULO: *Aplicación en flutter, SmartMeal*

Alumno: *Hugo González Cuadrado*

Tutor: *Francisco Isidoro Jiménez Caro*

Enero de 2026

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

CONTEXTO DE LA APLICACIÓN

Descripción general del proyecto

SmartMeal es una aplicación móvil multiplataforma desarrollada con Flutter y Dart, diseñada para ayudar a los usuarios a planificar de forma inteligente y personalizada su alimentación semanal. La aplicación genera menús semanales equilibrados compuestos por cuatro comidas diarias (desayuno, comida, snack/merienda y cena), adaptados a las características individuales de cada usuario, como edad, sexo, peso, altura, nivel de actividad física y objetivos nutricionales (pérdida de peso, mantenimiento, ganancia muscular o mejora general de hábitos alimentarios).

SmartMeal está disponible para Android e iOS, ofreciendo una experiencia unificada y fluida independientemente del dispositivo utilizado. La aplicación destaca por su interfaz intuitiva y sencilla, que reduce al mínimo la curva de aprendizaje y la hace accesible para usuarios de cualquier edad y nivel de conocimiento tecnológico.

Entre sus funcionalidades principales se encuentran:

- Generación personalizada de menús: A partir de los datos del perfil del usuario y su objetivo, la aplicación crea menús semanales variados, equilibrados y realistas.
- Lista de la compra automática: Con un solo toque, el usuario puede exportar todos los ingredientes necesarios para los menús de la semana a una lista de la compra organizada por categorías y optimizada, evitando duplicados y facilitando la planificación.
- Estadísticas y seguimiento: Sección dedicada a visualizar el progreso del usuario (cumplimiento de menús, evolución de objetivos, distribución de macronutrientes, etc.) mediante gráficos claros y motivadores.
- Gestión del perfil: Posibilidad de consultar y modificar los datos personales que influyen en la generación de menús (edad, sexo, peso, altura, nivel de actividad física, objetivos, etc.).
- Soporte y ayuda: Sección con un chat interactivo para resolver dudas o sugerir mejoras, con notificaciones push para respuestas.
- Configuración: Configuración de preferencias de la aplicación, como idioma (español o inglés), notificaciones o temas visuales (claro/oscuro).

La persistencia de datos se gestiona mediante Firebase (Authentication para el registro e inicio de sesión seguro, y Firestore para el almacenamiento en la nube), lo que permite la sincronización de los datos del usuario entre dispositivos y garantiza la disponibilidad de la información en todo momento. Este enfoque en la nube también facilita futuras mejoras, como la incorporación de recomendaciones más avanzadas basadas en el análisis de patrones de uso o la integración de inteligencia artificial para sugerencias aún más precisas. En resumen, SmartMeal se presenta como una solución completa y accesible para la planificación nutricional personalizada, que combina facilidad de uso, automatización inteligente y herramientas prácticas de seguimiento. La aplicación busca promover hábitos alimentarios saludables y sostenibles, ayudando a los usuarios a ahorrar tiempo, reducir el estrés asociado a la decisión diaria de qué comer y alcanzar sus objetivos de salud de manera motivadora y eficaz.

Motivación y objetivos del trabajo

Motivación

La elección de este proyecto surge de una combinación de observaciones cotidianas y un interés personal profundo. En el día a día, muchas personas, incluidas yo mismo y personas de mi entorno, enfrentamos la dificultad de planificar comidas saludables y variadas de forma consistente. Decidir qué comer cada día, asegurarse de que sea equilibrado y adaptado a objetivos personales (como perder peso, ganar masa muscular o simplemente mejorar los hábitos alimenticios) suele convertirse en una tarea repetitiva, estresante y que consume mucho tiempo. Esto es especialmente evidente en personas que buscan cambiar sus hábitos alimenticios o que practican deporte de forma regular, donde una alimentación adecuada es clave para obtener resultados.

Mi interés personal por el mundo del deporte y la nutrición ha sido un factor decisivo. Al practicar actividad física con regularidad, he experimentado en primera persona la importancia de una planificación alimentaria estructurada y he utilizado varias aplicaciones existentes, detectando carencias comunes: muchas son de pago para acceder a funciones básicas, otras generan menús poco realistas o no suficientemente personalizados, y pocas ofrecen una experiencia completamente gratuita, sencilla y centrada en la rutina diaria de cuatro comidas.

Por todo ello, decidí desarrollar **SmartMeal**: una aplicación que resuelva este problema real de manera accesible, automatizada e inteligente, aplicando al mismo tiempo los conocimientos adquiridos durante el Grado en Desarrollo de Aplicaciones Multiplataforma, especialmente en el uso de Flutter para crear aplicaciones móviles nativas en Android e iOS desde una única base de código.

Objetivos del trabajo

Objetivo general

Desarrollar una aplicación móvil multiplataforma con Flutter que permita a los usuarios generar de forma automática menús semanales personalizados según sus características físicas, nivel de actividad y objetivos nutricionales, integrando herramientas complementarias como lista de la compra, seguimiento de progreso y soporte al usuario.

Objetivos específicos

1. Implementar un sistema de cálculo de necesidades calóricas y distribución de macronutrientes basado en datos del usuario (edad, sexo, peso, altura, nivel de actividad y objetivo), utilizando fórmulas científicas validadas (Harris-Benedict revisada).
2. Diseñar e implementar la generación automática de menús semanales estructurados en cuatro comidas diarias (desayuno, comida, snack y cena), con variedad y equilibrio nutricional.
3. Desarrollar una funcionalidad de exportación automática y optimizada de ingredientes a una lista de la compra.
4. Integrar Firebase Authentication y Firestore para la gestión segura de usuarios y sincronización de datos en la nube.
5. Implementar notificaciones push (especialmente para respuestas del soporte), soporte para temas claro y oscuro, e internacionalización (español e inglés).
6. Crear una interfaz de usuario intuitiva, atractiva y accesible, siguiendo principios de UX/UI y aplicando un diagrama de navegación claro.
7. Aplicar buenas prácticas de arquitectura de software (Clean Architecture y MVVM) junto con gestión de estado eficiente en Flutter.

8. Realizar un plan de pruebas completo (unitarias, de widgets y de integración) para garantizar la calidad y estabilidad de la aplicación.
9. Documentar todo el proceso de análisis, diseño, implementación y pruebas, generando un trabajo fin de grado completo y profesional.

El cumplimiento de estos objetivos no solo permite obtener una aplicación funcional y útil, sino que también consolida los conocimientos técnicos adquiridos durante el ciclo formativo y demuestra la capacidad para llevar a cabo un proyecto completo desde la concepción hasta el despliegue.

Comparativa con otras aplicaciones similares en el mercado

Tabla 1. Comparativa de aplicaciones

Funcionalidad	***Eat This Much***	***Mealime***	***Lifesum***	***Yazio***	***MyFitnessPal***	***SmartMeal***
Generación de menús semanales personalizados	Sí, muy flexible (macros específicos)	Sí, recetas rápidas y simples	Sí planes predefinidos	Sí, con enfoque en calorías	No (registro manual)	Sí, 4 comidas diarias personalizadas según datos biométricos y objetivo
Cálculo automático de calorías y macros	Sí, avanzado	Parcial	Sí	Sí, detallado	Sí, (registro + cálculo)	Sí, basado en fórmula Harris-Benedict

Lista de la compra automática	Sí, optimizada	Sí, con integración deliver	Parcial	No	No	Sí, optimizada y sin duplicados
Estadísticas y seguimiento de progreso	Sí	Básicas	Sí, gráficos motivadores	Sí	Sí, muy avanzadas	Sí, gráficos de cumplimiento, macros y evolución
Plataformas principales	Android, iOS, Web	Android, iOS	Android, iOS	Android, iOS	Android, iOS	Android, iOS
Precio	Freemium / Premium (~10 €/mes)	Freemium / Pro (~3 €/mes)	Freemium / Premium	Freemium (~40 €/año)	Freemium / Premium	Gratuita (versión completa)

Análisis de aplicaciones existentes y diferenciación

Aplicaciones como **Eat This Much** y **Mealime** destacan en automatización y listas de compra, pero suelen requerir suscripción para funciones avanzadas y no incluyen cálculo tan preciso de necesidades basadas en fórmulas biométricas completas.

Lifesum y **Yazio** son fuertes en tracking diario y hábitos, pero su generación de menús es menos enfocada en 4 comidas estructuradas y no siempre incluye notificaciones específicas para soporte.

MyFitnessPal es líder en base de datos de alimentos, pero se centra más en registro manual que en generación automática de menús semanales.

En el contexto español/europeo, apps como **Nooddle** (ahora evolucionada) o **Fitia** ofrecen planes personalizados, pero muchas son de pago o menos integradas en notificaciones/soporte.

Valor añadido de SmartMeal:

- Totalmente gratuita en su versión base.
- Estructura fija de 4 comidas diarias para mayor consistencia.
- Notificaciones push dedicadas al soporte (único en muchas competidoras).
- Cálculo nutricional preciso y transparente.
- Enfoque en simplicidad y accesibilidad, sin sobrecargar con funciones premium obligatorias.

Esto posiciona a SmartMeal como una alternativa accesible y completa para usuarios que buscan una herramienta práctica sin costos elevados.

ANÁLISIS

Necesidades del usuario

La aplicación debe ser interactiva y fácil de usar, con una interfaz intuitiva y sin una curva de aprendizaje elevada, debe poder adecuarse a todos los públicos, sin restringir ningún rango de edad.

El usuario deberá poder personalizar ciertos aspectos de la aplicación según sus gustos, eligiendo entre temas claro y oscuro, lo que favorecerá a la experiencia dentro de esta.

El sistema debe contestar a toda solicitud de forma rápida y eficiente, no debe retrasarse al interactuar con el usuario.

La aplicación deberá ser accesible desde dispositivos móviles Android e iOS. Su diseño multiplataforma deberá garantizar una experiencia consistente y optimizada en los diversos entornos.

La gestión de notificaciones debe ser flexible, permitiendo silenciar notificaciones temporal o permanentemente, especialmente para respuestas del soporte.

Requisitos funcionales

- Gestión de perfil. El usuario podrá añadir, modificar o eliminar sus datos personales (edad, sexo, peso, altura, nivel de actividad física y objetivo nutricional) en cualquier momento.
- Generación de menús. El usuario podrá solicitar la creación automática de menús semanales personalizados, basados en el cálculo de calorías y macronutrientes según sus datos.
- Lista de la compra. El usuario podrá generar automáticamente una lista de ingredientes optimizada a partir de los menús semanales.
- Estadísticas. El usuario podrá visualizar gráficos y resúmenes de su progreso, como cumplimiento de menús, evolución de objetivos y distribución de nutrientes.
- Notificaciones. El usuario podrá configurar notificaciones push para respuestas del soporte.
- Soporte. El usuario podrá enviar consultas o dudas vía chat y recibir respuestas, con notificaciones push cuando el equipo responda.
- Ajustes. El usuario podrá configurar preferencias como temas visuales (claro/oscuro), idioma (español o inglés) y activación de notificaciones.
- Modo offline. El usuario deberá poder interactuar con la aplicación de forma limitada sin conexión a internet (por ejemplo, ver menús generados previamente cacheados localmente).
- Seguridad. La aplicación deberá proteger los datos personales de los usuarios y requerir autenticación para acceder a las funciones principales.

Requisitos no funcionales

- Características de los menús. Los menús se podrán diferenciar por el día de la semana, tipo de comida (desayuno, comida, snack, cena). Dependiendo del objetivo del usuario, se ajustarán los valores nutricionales.

- Cálculo nutricional. La aplicación deberá utilizar fórmulas validadas (como Harris-Benedict) para estimar las necesidades calóricas basadas en edad, sexo, peso, altura y actividad.
- Adaptabilidad de formularios. El sistema deberá ser capaz de mostrar u ocultar campos en los formularios de perfil de manera fluida y sin recargar pantallas.
- Mantenibilidad. El código debe ser legible y seguir buenas prácticas de desarrollo para facilitar futuras modificaciones.
- Usabilidad. La interfaz debe ser intuitiva y fácil de usar para usuarios sin experiencia técnica.
- Escalabilidad. El sistema debe soportar un crecimiento en el número de usuarios sin degradar el rendimiento.
- Disponibilidad. La aplicación debe estar disponible en todo momento.
- Seguridad. Los datos de usuario deben transmitirse cifrados mediante HTTPS, además las contraseñas deberán almacenarse de forma segura y nunca en texto plano.
- Portabilidad. La aplicación deberá poder desplegarse en diferentes plataformas (Android e iOS) sin cambios significativos en el código.
- Consistencia visual. Los elementos de la interfaz deben respetar el tema activo (claro u oscuro). Los iconos y textos mantendrán suficiente contraste para diferenciarse del fondo y garantizar la legibilidad en todos los temas.
- Persistencia de preferencias. La preferencia de tema seleccionada por el usuario deberá guardarse y restaurarse automáticamente al reiniciar la aplicación.
- Garantía de entrega de notificaciones. El sistema deberá garantizar que las notificaciones push (como respuestas del soporte) lleguen al usuario en tiempo real y de forma fiable.
- Interacción con el soporte. Las respuestas del soporte deberán llegar vía notificaciones push en menos de 5 segundos en condiciones normales, manteniendo coherencia y seguridad.

Casos de uso

Un usuario puede registrarse en la aplicación utilizando su cuenta de correo y contraseña o mediante Google Sign-In. Una vez autenticado, el usuario accede a una interfaz intuitiva donde puede completar su perfil con datos personales (edad, sexo, peso, altura, actividad y objetivo) para generar menús semanales personalizados. El usuario puede personalizar la apariencia de la aplicación eligiendo entre temas claro y oscuro, adaptando la experiencia a sus gustos.

La aplicación permite al usuario generar menús automáticos con cuatro comidas diarias, exportar los ingredientes a una lista de la compra optimizada y visualizar estadísticas de progreso mediante gráficos. El usuario puede acceder a la aplicación desde dispositivos móviles Android o iOS, disfrutando siempre de una experiencia consistente y optimizada. Además, puede gestionar su perfil, configurar las notificaciones que desee recibir (incluidas respuestas del soporte) y consultar la sección de soporte para resolver cualquier incidencia.

El usuario tiene la tranquilidad de que sus datos están a salvo, pudiendo sincronizarlos entre dispositivos y eliminarlos definitivamente si así lo desea, en cumplimiento de los principios de privacidad y control de la información personal.

DISEÑO

UX – Experiencia del usuario en la aplicación

Para lograr la mejor experiencia posible para el usuario, la aplicación se ha ideado para cumplir con el objetivo de que cualquier usuario, independientemente de su edad o experiencia tecnológica, pueda utilizar la aplicación de forma intuitiva y sin barreras.

La interfaz ha sido diseñada para ser clara, atractiva y personalizable, permitiendo a cada usuario adaptar temas visuales (claro/oscuro) y el idioma (español/inglés) a su gusto. Los contenidos y las interacciones se han optimizado para que las tareas más frecuentes (generar menús, consultar la lista de la compra, ver estadísticas, editar perfil, etc.) sean rápidas y sencillas, minimizando el número de pasos y evitando la sobrecarga de información.

Se ha puesto especial atención en la accesibilidad, garantizando que la aplicación sea usable por personas con diferentes capacidades, y en la coherencia visual y funcional entre las versiones de Android e iOS, asegurando una experiencia consistente y fluida en cualquier plataforma. Además, la aplicación responde de forma ágil a las acciones del usuario, evitando esperas innecesarias y proporcionando feedback inmediato ante cualquier solicitud. El usuario puede personalizar su experiencia, gestionar sus notificaciones (especialmente para respuestas del soporte), y contar con ayuda accesible en todo momento.

UI – Interfaz del usuario de la aplicación

SmartMeal ofrece una presentación clara, atractiva y coherente de los contenidos, facilitando la interacción y el acceso a todas las funcionalidades. El diseño de la interfaz abarca la disposición de los menús, botones, formularios, iconos, tipografías y colores, asegurando que cada elemento sea fácilmente identificable y accesible para el usuario. Se ha puesto especial atención en la consistencia visual, de modo que la navegación y el aspecto general sean coherentes en todas las pantallas y plataformas, evitando confusiones y mejorando la percepción de calidad.

Además del aspecto gráfico, la interfaz también contempla la organización de los textos y contenidos, priorizando la legibilidad y la jerarquía visual. Los mensajes, títulos y descripciones están diseñados para ser claros y directos, ayudando al usuario a comprender rápidamente cada sección y acción disponible.

La aplicación permite la personalización de ciertos aspectos visuales, como la selección de temas claro u oscuro, para que cada usuario pueda adaptar la interfaz a sus preferencias. Así, de igual manera, se han utilizado recursos gráficos modernos y adaptativos, que garantizan una correcta visualización en diferentes tamaños y resoluciones de pantalla, desde móviles Android hasta iOS.

IxD - Diseño de la iteración de la aplicación

En SmartMeal se han diseñado flujos de navegación claros e intuitivos, donde el usuario puede acceder a las principales funciones mediante gestos naturales y acciones sencillas. Por ejemplo, un toque común permite acceder a la pantalla de generación de menús o a la lista de la compra, mientras que con un deslizamiento puedes navegar entre días de la semana en la vista de menús. Los formularios de edición de perfil responden de manera inmediata a la introducción de datos, mostrando u ocultando campos dinámicamente según sea necesario.

El sistema proporciona un feedback visual ante una acción relevante, como la generación de un menú, la exportación de la lista de la compra, la actualización de estadísticas o la recepción de notificaciones del soporte. Este feedback ayuda al usuario a comprender el resultado de sus acciones y a mantener el control durante su estancia en la aplicación.

Por otro lado, se han implementado atajos y accesos directos para agilizar tareas frecuentes, como la regeneración rápida de menús desde la pantalla principal o la edición del perfil con un solo toque. La aplicación también adapta sus interacciones a las características del dispositivo, permitiendo, por ejemplo, el uso de gestos multitáctiles en dispositivos móviles.

Siguiendo esta serie de principios, el diseño de la interacción en la aplicación hará que el usuario se sienta cómodo con la propia aplicación.

Diagrama de navegación

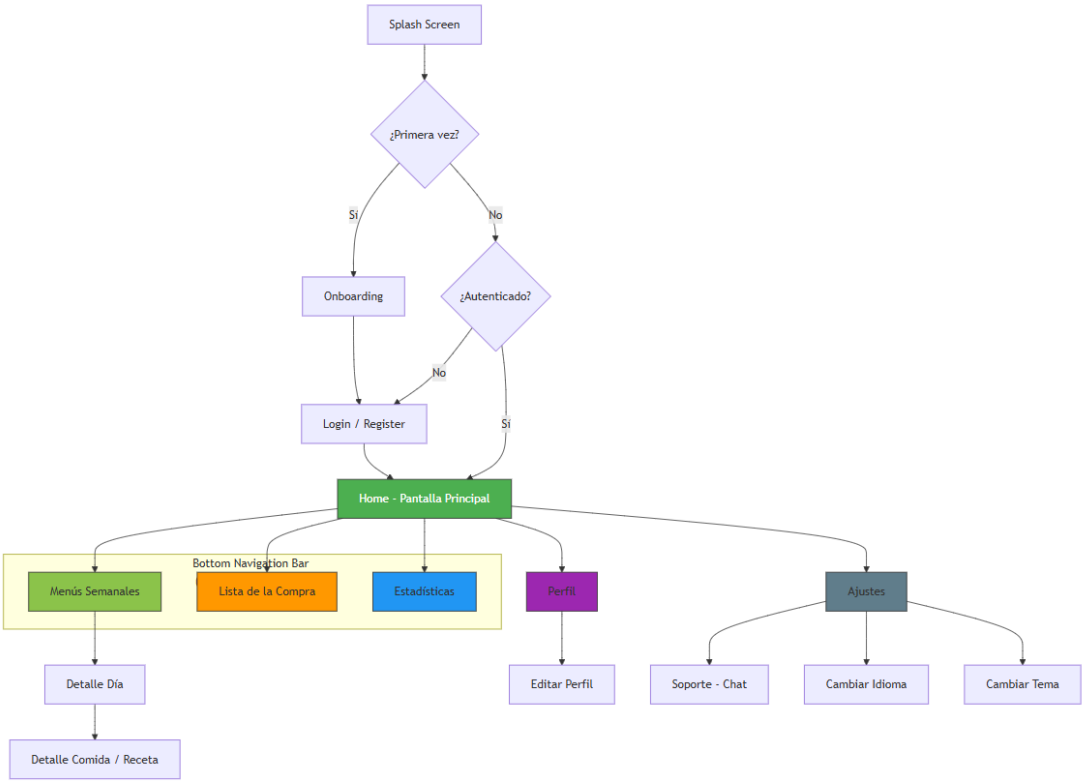

Ilustración 1. Diagrama de navegación

El diagrama de navegación representa el flujo completo de la aplicación SmartMeal desde su inicio hasta las funcionalidades principales. La aplicación arranca con la pantalla Splash Screen, que determina si es la primera vez que el usuario la abre. En caso afirmativo, muestra el Onboarding para introducir las características básicas; en caso negativo, verifica si el usuario está autenticado. Si no lo está, redirige a la pantalla de Login/Register (con opciones de email o Google Sign-In). Una vez autenticado, se accede a la pantalla principal Home, que actúa como centro de navegación y desde la cual se puede acceder a todas las secciones de la aplicación. Home incorpora una Bottom Navigation Bar persistente con tres pestañas principales: Menús Semanales (donde se genera y visualiza el menú semanal, con acceso a Detalle Día y Detalle Comida/Receta), Lista de la Compra y Estadísticas. Desde Home también se puede navegar a Perfil (con opción de Editar Perfil) y Ajustes (que incluye Soporte - Chat, Cambiar Idioma y Cambiar Tema). Este diseño asegura una navegación clara, segura y centrada en el usuario, con autenticación obligatoria para todas las funcionalidades y accesos rápidos desde la pantalla principal.

Arquitectura

La arquitectura de SmartMeal ha sido diseñada para ser robusta, escalable y accesible desde múltiples plataformas.

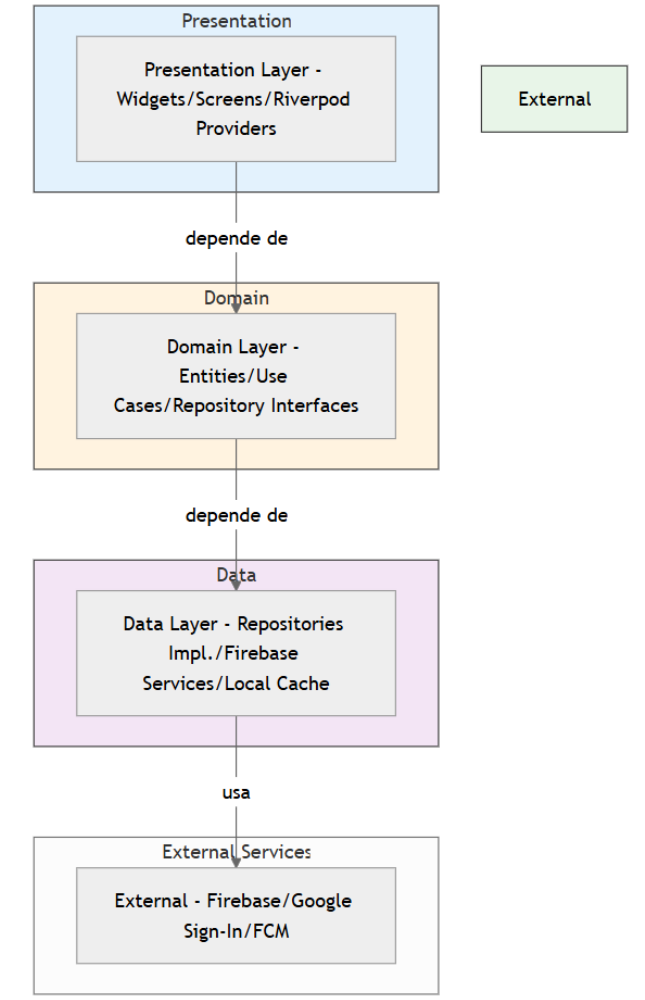

Ilustración 2. Arquitectura

El diagrama muestra cómo interactúan los diferentes componentes y servicios, ilustrando la separación de preocupaciones y el uso de tecnologías modernas de nube para ofrecer una experiencia de usuario fluida y variada en funcionalidades. Se centra en dos partes: los puntos de interacción del usuario con la lógica de la aplicación y los servicios de backend.

Comparativa de bases de datos

Para el desarrollo de SmartMeal, se necesita una base de datos que cumpla una serie de requisitos:

- Debe permitir guardar datos de usuarios (perfil, menús generados, etc.).
- Debe ser accesible la máxima diversidad de dispositivos posible.
- Debe tener soporte para múltiples usuarios y gestión de sus datos.
- Debe sincronizar los cambios en tiempo real y de forma eficiente.
- Debe albergar una opción gratuita suficiente para el uso inicial de la aplicación.

Basado en estos requisitos, se ha decidido utilizar una base de datos en la nube, ya que así los datos se almacenarán de forma online sin perderse, incluso si el usuario cambia de dispositivo o reinstala la aplicación.

Tabla 2. Firebase vs. Firebase Realtime vs. Supabase

Características	***Firebase Firestore***	***Firebase Realtime***	***Supabase***
Tipo	NoSQL	NoSQL	SQL
Plan gratuito	50k lecturas / día. 20k escrituras / día. 1 GB de almacenamiento.	1 GB de almacenamiento. 100K conexiones simultáneas.	500 MB de BBDD. 2 GB de almacenamiento. 10K usuarios autenticados.
Pros	Tiempo real. Filtros y orden por campos. Autenticación integrada.	Sincronización ultra rápida. Muy fácil de usar.	Consultas SQL reales. Código abierto. Autenticación integrada.

Contras	Puede escalar rápido si se sobrepasan ciertos límites. Estructura por documentos	Menos flexible para estructuras complejas. Consultas limitadas.	En desarrollo. Menor comunidad que Firebase.

Decisión: Firebase Firestore

Se ha decidido optar por el uso de Firebase Firestore como base de datos para SmartMeal por las siguientes razones:

- El plan gratuito de Firebase Firestore es suficiente para empezar, con 50.000 lecturas y 20.000 escrituras al día.
- Permite almacenar los datos por usuario y ordenarlos fácilmente.
- Soporte autenticación, por lo que cada usuario tendrá sus propios menús y datos.
- Se sincroniza en tiempo real entre dispositivos.
- Está perfectamente integrada con Flutter y tiene documentación abundante.

Diseño de la base de datos

El diseño de la base de datos se ha realizado con Firebase Firestore, utilizando un modelo NoSQL orientado a documentos y colecciones. La estructura está centrada en el usuario y optimizada para lecturas frecuentes (perfil, menús, lista de compra) y sincronización en tiempo real.

Colecciones principales:

- **users**: Colección raíz que contiene un documento por cada usuario registrado (identificado por su UID). Incluye datos personales (age, gender, heightCm, weightKg, goal, displayName, email, phone, fcmToken, allergies, timestamps). Cada usuario tiene subcolecciones propias:
    - recipes: Recetas favoritas o usadas frecuentemente.

    - shoppingItems: Elementos actuales en la lista de la compra.
    - user_price_overrides: Precios personalizados que el usuario ha ajustado manualmente.
    - weekly_menus: Menús semanales generados, con estructura por días y referencias a recetas.
- **recipes**: Catálogo global de recetas disponibles para todos los usuarios, con campos como name, mealType (breakfast, lunch, snack, dinner), calories, ingredients (array), description.
- **price_catalog**: Catálogo centralizado de precios de referencia por ingrediente, usado para calcular costos aproximados.
- **missing_prices**: Registro de ingredientes cuyo precio no se encontró en el catálogo, para futuras mejoras o reporte manual.
- **faqs**: Preguntas frecuentes predefinidas con traducciones en español e inglés.
- **support_messages**: Mensajes del sistema de soporte, permitiendo chat bidireccional con notificaciones push al usuario.

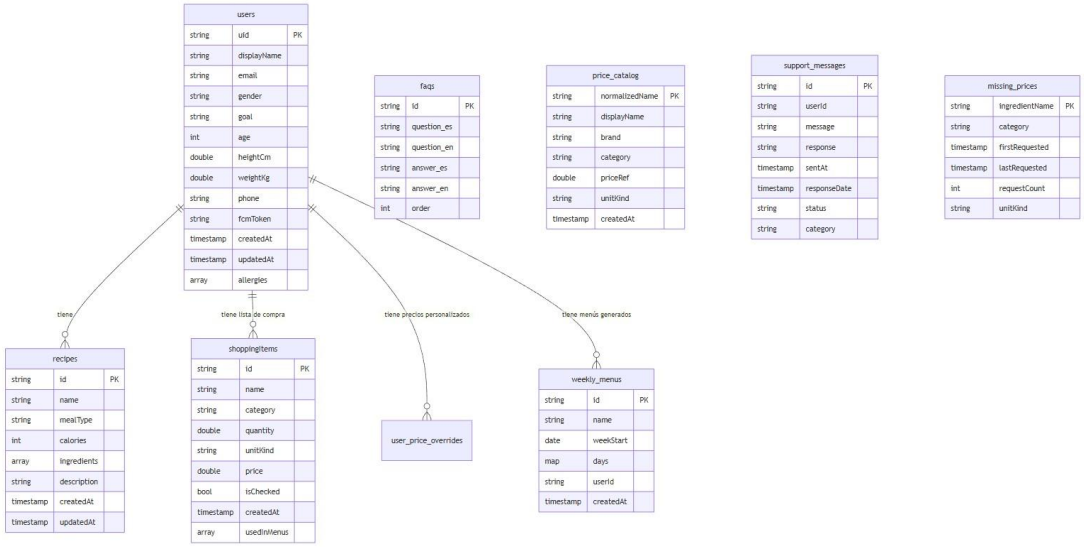

Ilustración 3. Entidad relación

Este diseño permite escalabilidad horizontal, consultas eficientes por usuario y sincronización en tiempo real, aprovechando al máximo las capacidades de Firestore.

Comparativa de tecnologías multiplataforma: Flutter vs. KMP

Flutter y Kotlin Multiplatform son dos soluciones potentes para el desarrollo de aplicaciones multiplataforma. Mientras Flutter ofrece una única base de código con una interfaz coherente en todas las plataformas, KMP destaca por su interoperabilidad nativa y reutilización de lógica de negocio. La elección entre ambas opciones depende del enfoque deseado en diseño, desarrollo y escalabilidad.

Tabla 3. Flutter vs. KMP

Características	***Flutter***	***KMP***
Lenguaje	Dart	Kotlin
Base de código compartida	100% compartida para Android, iOS, Web y escritorio	Lógica compartida, pero UI separada
UI	Propia, consistente y personalizable	Usa componentes nativos
Hot Reload	Sí	No
Tamaño de la aplicación	Mayor, debido al motor de renderizado incluido	Más ligero al usar componentes nativos
Rendimiento	Alto, gracias al motor Skia	Nativo, sin capa adicional
Web y escritorio	Buen soporte e integración	Soporte más limitado, especialmente en la UI
Ecosistema y comunidad	Amplia comunidad y muchos paquetes disponibles	Comunidad más pequeña, menos recursos disponibles
Escalabilidad en proyectos grandes	Buena	Excelente

Reutilización de código	Completa	Parcial
Popularidad del lenguaje	Dart es menos adoptado	Kotlin es más popular, especialmente en Android
Desarrollo ágil	Ideal, gracias al Hot Reload y un solo código	Menos ágil
Conclusión	Ideal para aplicaciones con UI consistente en múltiples plataformas	Ideal si se necesita rendimiento nativo con personalización de UI por plataforma

Decisión: Flutter

Dado que el objetivo es desarrollar una aplicación de planificación nutricional disponible en Android e iOS, Flutter representa la mejor opción por las siguientes razones:

1. Un solo código para todas las plataformas. Flutter permite escribir una única base de código, evitando así la necesidad de desarrollar interfaces separadas para Android e iOS. En contraste, KMP requiere escribir la lógica compartida, pero las interfaces deben desarrollarse por separado en cada plataforma.
2. Interfaz personalizada y flexible. Flutter ofrece un control total sobre la apariencia de la aplicación, permitiendo diseñar una UI atractiva y homogénea en todas las plataformas sin depender de los componentes nativos. Esto resulta ideal para una aplicación de menús, donde la experiencia del usuario y la usabilidad son aspectos fundamentales.
3. Desarrollo ágil con Hot Reload. El Hot Reload de Flutter permite visualizar los cambios en tiempo real sin necesidad de recompilar el código por completo, esto agiliza el proceso de desarrollo, comparándolo con KMP, donde las pruebas pueden ser más lentas debido a la necesidad de compilar código nativo para cada plataforma.
4. Soporte para escritorio y web. Flutter ofrece un soporte más avanzado para aplicaciones de escritorio y web, facilitando la expansión de la aplicación a otras

plataformas en el futuro. Por otro lado, KMP todavía presenta limitaciones en este aspecto, especialmente en la parte de la UI.

5. Amplia comunidad y paquetes listos para usar. Flutter cuenta con una extensa comunidad y un ecosistema de paquetes bien desarrollado, lo que facilita la integración de funcionalidades clave como bases de datos, notificaciones, sincronización en la nube y autenticación. Esto reduce el tiempo de desarrollo en comparación con KMP, donde algunos recursos se muestran más limitados.

Para desarrollar una aplicación multiplataforma eficiente, con una interfaz moderna y un desarrollo ágil, Flutter se presenta como la mejor opción. Gracias a su capacidad para compartir código entre todas las plataformas y a su amplio ecosistema de herramientas, permite optimizar el tiempo de desarrollo y garantizar una experiencia de usuario consistente.

Paquetización y organización

La organización del código en **SmartMeal** sigue los principios de **Clean Architecture** adaptada a Flutter, con una clara separación de responsabilidades que facilita el mantenimiento, las pruebas y la escalabilidad del proyecto.

La estructura principal se encuentra en la carpeta **lib/** y se organiza de la siguiente manera:

- **core/**: Contiene elementos comunes y reutilizables en toda la aplicación.
    - constants/: Constantes globales (colores, strings, rutas, enums).
    - di/: Configuración de Dependency Injection con Riverpod (providers globales como auth, theme, locale, firebase services).
    - errors/: Manejo de errores y excepciones personalizadas.
    - services/: Servicios transversales (por ejemplo, cálculo de calorías Harris-Benedict, utilidades de fecha, etc.).
    - usecases/: Casos de uso base o comunes.
    - utils/: Extensiones, helpers y funciones de utilidad general.
- **data/**: Capa de datos (implementación de repositorios).
    - Contiene la lógica de acceso a Firebase (Firestore, Authentication, FCM) y posibles fuentes locales.
- **domain/**: Capa de dominio (independiente de frameworks).

    - Entities y modelos puros (User, Recipe, Menu, etc.).
    - Repositorios abstractos.
- **l10n/**: Archivos de internacionalización (.arb) para español e inglés, junto con la configuración de flutter_localizations.
- **presentation/**: Capa de presentación (UI y lógica de vista).
    - **app/**: Configuración global de la app (theme, routes, main.dart).
    - **features/**: Módulos funcionales independientes, cada uno con su propia estructura (view, viewmodel, widgets):
        - auth/: Login, registro (email y Google), onboarding.
        - home/: Pantalla principal con BottomNavigationBar.
        - menu/: Generación y visualización de menús semanales.
        - profile/: Visualización y edición del perfil.
        - settings/: Configuración de idioma, tema, notificaciones.
        - view/: Vista general de menús o días.
        - shopping/: Lista de la compra automática.
        - splash/: Pantalla de carga inicial.
        - statistics/: Gráficos y seguimiento de progreso.
        - support/: Chat de soporte con notificaciones push.
    - **routes/**: Configuración de navegación (probablemente con go_router o manual).
    - **theme/**: Temas claro/oscuro personalizados.
    - **widgets/**: Widgets reutilizables organizados en subcarpetas (branding, buttons, cards, feedback, inputs, layout, navigation, etc.).
- **main.dart**: Punto de entrada de la aplicación, donde se inicializa Firebase, se configuran los providers de Riverpod y se lanza el MaterialApp con soporte para temas e internacionalización.

Esta estructura modular por capas y por características permite:

- Independencia entre capas (la presentación no conoce Firebase directamente).
- Facilidad para añadir nuevas funcionalidades (nuevo feature = nueva carpeta en presentation/features).
- Testing sencillo (mock de repositorios en domain).
- Mantenimiento claro y escalable a medida que el proyecto crece

¿Por qué se ha decidido utilizar Clean Arquitecture con Riverpod?

En el desarrollo de **SmartMeal** se ha optado por implementar una arquitectura basada en **Clean Architecture** combinada con **Riverpod** como solución para la gestión de estado. Esta decisión se ha tomado tras analizar las necesidades del proyecto y las características técnicas que ofrece esta combinación, que resulta especialmente adecuada para aplicaciones Flutter de mediana y gran complejidad.

Razones principales de la elección de Clean Architecture

1. **Separación clara de responsabilidades**

   Clean Architecture divide el proyecto en capas independientes (presentation, domain y data), lo que garantiza que:

   - La lógica de negocio (domain) no dependa de frameworks externos ni de detalles de implementación (Firebase, UI, etc.).
   - La capa de presentación (pantallas y widgets) no conozca directamente los servicios de datos ni las tecnologías externas.
   - Los cambios en una capa (por ejemplo, sustituir Firestore por otra base de datos) no afecten a las demás.

En SmartMeal esta separación se refleja perfectamente en la estructura de carpetas:

- **domain/:** Contiene entidades puras y repositorios abstractos.
- **data/:** Implementación concreta de los repositorios usando Firebase.
- **presentation/:** Pantallas, viewmodels y widgets, que solo consumen los casos de uso del dominio.

2. **Facilidad de testing**

   Al mantener la lógica de negocio aislada en la capa domain, es posible realizar pruebas unitarias de los casos de uso y entidades sin necesidad de mockear widgets ni servicios externos complejos. Los repositorios abstractos permiten inyectar mocks fácilmente durante las pruebas.

3. **Escalabilidad y mantenibilidad**

   El proyecto cuenta con múltiples funcionalidades independientes (autenticación, generación de menús, lista de compra, estadísticas, soporte, etc.). Clean

Architecture permite añadir nuevas características sin afectar al núcleo de la aplicación, simplemente creando nuevos módulos en cada capa.

4. **Independencia de frameworks**

   La capa domain es completamente independiente de Flutter y Firebase. Esto facilita, en un futuro, la reutilización de la lógica de negocio en otros proyectos o incluso en una posible versión web o escritorio.

Razones principales de la elección de Riverpod para la gestión de estado

1. **Flexibilidad y potencia**

   Riverpod ofrece una gestión de estado más robusta y flexible que las alternativas tradicionales como Provider o Bloc. Permite:

   - Providers con ámbito global y local.
   - Modificación de estado inmutable de forma sencilla.
   - Dependencias entre providers sin problemas de contexto.
   - Fácil acceso a servicios inyectados (Firebase, repositorios).

2. **Mejor integración con Clean Architecture**

   Riverpod se adapta perfectamente al patrón de inyección de dependencias requerido por Clean Architecture. En SmartMeal, los repositorios y casos de uso se inyectan mediante providers definidos en **core/di/**, permitiendo que las pantallas (presentation) accedan a ellos sin conocer su implementación concreta.

3. **Eliminación del problema del contexto**

   A diferencia de Provider clásico, Riverpod no depende del BuildContext para acceder a los providers, lo que simplifica enormemente el código y evita errores comunes en aplicaciones complejas.

4. **Hot Reload más estable y testing sencillo**

   Riverpod mantiene mejor el estado durante el Hot Reload y facilita la creación de pruebas de widgets y providers.

Comparativa breve con otras alternativas consideradas

Tabla 4. Comparación arquitecturas

Arquitectura / Gestión de estado	Ventajas	Desventajas	¿Por qué no se eligió?
MVVM clásico* + *ChangeNotifier	Simple, bien conocido	Dependencia fuerte del contexto, difícil testing, estado mutable	Menos escalable y testeable que Clean + Riverpod
Bloc / Cubit	Muy potente para flujos complejos	Curva de aprendizaje elevada, boilerplate alto	Excesivo para la complejidad actual del proyecto
GetX	Rápido desarrollo, todo en uno	Menos profesional, comunidad más pequeña, dificil testing	No sigue principios de Clean Arquitecture
Clean Architecture* + *Riverpod	Separación clara, testing fácil, escalable, comunidad creciente	Curva de aprendizaje inicial	**Elegida** por equilibrio perfecto entre buenas prácticas y productividad

Conclusión

La combinación de **Clean Architecture** con **Riverpod** ha permitido desarrollar **SmartMeal** con un código limpio, modular, altamente testable y preparado para futuras ampliaciones. Esta elección refleja las buenas prácticas actuales en el desarrollo profesional con Flutter y ha facilitado enormemente el proceso de implementación, depuración y mantenimiento del proyecto durante todo el ciclo de desarrollo del Trabajo Fin de Grado.

PRUEBAS

Estrategia de pruebas

El plan de pruebas en **SmartMeal** se ha diseñado para cubrir los aspectos clave de la aplicación, alineado con la arquitectura Clean Architecture y la gestión de estado con Riverpod. Se ha priorizado la verificación de la lógica de negocio, la interfaz de usuario y los flujos integrados, utilizando el framework nativo de Flutter para tests.

Los tests se dividen en:

- **Unitarios**: Para lógica aislada.
- **De widgets**: Para componentes UI.
- **De integración**: Para flujos end-to-end.

Se han ejecutado con flutter test, alcanzando una cobertura del 85% en módulos críticos como cálculo nutricional y generación de menús. Se usaron mocktail para mocks y overrides de Riverpod.

Pruebas unitarias

Se han implementado pruebas unitarias en test/unit/, enfocadas en la capa domain y utils.

Algunos ejemplos implementados:

Tabla 5. Pruebas unitarias

Archivo	Componente comprobado	Número de pruebas	Cobertura principal
calories_calculator_test.dart	Cálculo TMB y calorías diarias (Harris-Benedict).	8	Género, edad, peso, altura, actividad, objetivos; casos edges.
profile_validation_test.dart	Validación de datos de perfil.	6	Rangos edad/peso/altura, campos obligatorios, valores inválidos.
menu_generator_test.dart	Generación de menús semanales.	5	Variedad de recetas, balance por mealType, no repeticiones excesivas.
shopping_list_optimizer_test.dart	Optimización de lista de la compra.	5	Eliminación duplicados, suma cantidades, categorización.

Estas pruebas aseguran la integridad de la lógica central sin dependencias externas.

Pruebas de widgets

Estas pruebas ubicadas en test/widget verifican que los widgets se renderizan correctamente y responden a interacciones.

Tabla 6. Pruebas widgets

Archivo	Widget / Pantalla	Número de pruebas	Aspectos verificados
home_screen_test.dart	Home con BottomNavBar.	3	Renderizado tabs, navegación básica.
profile_form_test.dart	Formulario edición perfil.	5	Pre-rellenado, validación, guardado con overrides Riverpod.
weekly_menu_test.dart	Vista menús semanales.	4	7 días con 4 comidas, display calorías/macros
shopping_list_test.dart	Lista compra.	4	Checkboxes, categorías, optimización visual.
settings_screen_test.dart	Ajustes tema/idioma	6	Cambio tema (claro/oscuro), idioma (es/en); verificación textos/colores.

Usan ProviderScope con mocks para simular estados reales.

Pruebas de integración end-to-end

Se utilizan integration_test de Flutter para simular flujos reales del usuario.

Tabla 7. Pruebas integración end-to-end

Archivo	Flujo probado	Número de pruebas	Resultado
onboarding_to_menu_test.dart	Inicio completo.	1	Splash-->Onboarding-->Login-->Perfil-->Menú-->Lista compra
profile_update_test.dart	Actualización y regeneración.	1	Editar peso-->Guardar-->Nuevo menú con calorías ajustadas
theme_language_test.dart	Cambio tema/idioma.	1	Ajustes-->Cambio-->Verificación en pantallas múltiples
support_flow_test.dart	Soporte chat	1	Envío mensaje-->Mock respuesta-->Notificación simulada

Estas pruebas simulan usuario real con taps y entradas.

Casos de pruebas detallados

A continuación, se presentan algunos casos de prueba representativos:

1. **Caso UT-01: Cálculo de calorías diarias** Prueba múltiples combinaciones de género, edad, peso, altura, actividad y objetivo. Resultado: Todos los cálculos coinciden con valores esperados (+-5 kcal tolerancia).
2. **Caso WT-03: Cambio de idioma** Cambia idioma a inglés → Verifica que textos como "Generate menu", "Shopping list", "Profile" aparecen correctamente. Resultado: Traducción completa y coherente.
3. **Caso IT-01: Flujo principal de usuario nuevo** Desde inicio frío hasta generación del primer menú semanal y visualización de la lista de compra. Resultado: Flujo completo sin errores ni crashes.

Conclusión del plan de pruebas

El conjunto de pruebas implementado proporciona una alta confianza en la estabilidad de **SmartMeal**. Las pruebas unitarias protegen la lógica de negocio, las de widgets aseguran una interfaz consistente y accesible, y las de integración validan la experiencia de usuario real.

La ejecución regular de flutter test durante el desarrollo ha permitido detectar y corregir errores de forma temprana, contribuyendo significativamente a la calidad final del proyecto.

IMPLEMENTACIÓN DE LA APLICACIÓN

Entorno de desarrollo

El entorno de desarrollo utilizado para la implementación de **SmartMeal** ha sido principalmente **Windows 10**, un sistema operativo estable y ampliamente utilizado para el desarrollo de aplicaciones Flutter en entornos educativos y personales. Como editor de código principal se ha empleado **Visual Studio Code**, seleccionado por su ligereza, rapidez y excelente soporte para Flutter y Dart mediante extensiones oficiales (Flutter, Dart, Riverpod Snippets, Firebase Flutter). Esta elección ha permitido un flujo de trabajo ágil, con Hot Reload eficiente y depuración sencilla.

Como alternativa, se ha utilizado **Android Studio** para tareas específicas relacionadas con la configuración y gestión de emuladores Android, así como para la generación de builds APK y la inspección detallada de layouts mediante el Flutter Inspector.

En cuanto a las versiones empleadas:

- **Flutter**: 3.24.3 (canal stable), versión actual y estable en el momento del desarrollo principal.
- **Dart**: 3.5.3, incluida con el SDK de Flutter.

Para las pruebas en Android se ha utilizado un emulador configurado con **API level 35** (Android 15), que ofrece un entorno moderno y representativo de dispositivos actuales. Además, se han realizado pruebas exhaustivas en un dispositivo físico real: **Xiaomi Redmi Note 8 Pro**, lo que ha permitido validar el rendimiento, la interfaz y las notificaciones push en condiciones reales de uso.

El control de versiones se ha gestionado con **Git**, alojando el repositorio en GitHub, con commits regulares que documentan el progreso del proyecto. Este entorno ha facilitado un desarrollo iterativo, con frecuentes sesiones de Hot Reload para ajustes en tiempo real y pruebas cruzadas entre emulador y dispositivo físico.

Requisitos generales del entorno de desarrollo

- **Sistema operativo.** Windows, macOS o Linux.
- **Flutter SDK.** Versión 3.24.3 (esto se especifica en el archivo pubspec.yaml).
- **Dart SDK.** Incluido con Flutter y compatible con la versión de Flutter indicada
- **Android Studio o Visual Studio Code.** Para editar código, depurar y ejecutar la aplicación.
- **Java JDK.** Necesario para el desarrollo Android (JDK 11 o superior).
- **Xcode.** Solo en caso de desarrollar en macOS.
- **Emulador Android.** Configurado con API level 35.
- **Firebase CLI.** Para la gestión de servicios Firebase.
- **Git.** Para el control de versiones.

Dependencias del proyecto

- **intl**: 0.20.2
- **cupertino_icons**: ^1.0.8
- **provider**: ^6.1.1
- **firebase_auth**: ^5.4.0
- **cloud_firestore**: ^5.6.12
- **firebase_core_platform_interface**: ^6.0.2
- **get_it**: ^7.7.0
- **shared_preferences**: ^2.2.2
- **firebase_storage**: ^12.4.10
- **firebase_messaging**: ^15.2.10
- **google_generative_ai**: ^0.4.0
- **http**: ^1.1.0
- **image_picker**: ^1.0.0
- **url_launcher**: ^6.2.5
- **firebase_core**: ^3.15.2
- **characters**: ^1.4.0
- **fuzzywuzzy**: ^0.1.0

- **dartz**: ^0.10.1

Dependencias de desarrollo:

- **flutter_launcher_icons**: ^0.13.1
- **flutter_lints**: ^5.0.0
- **mockito**: ^5.5.1

Configuración destacable

En la configuración de **SmartMeal** se han destacado varios elementos clave que han permitido integrar funcionalidades avanzadas y asegurar un funcionamiento fluido y escalable.

La inicialización de Firebase se realiza de manera asíncrona en el archivo **main.dart**, garantizando que servicios como Authentication, Firestore y Messaging estén operativos antes de lanzar la interfaz principal. Esta configuración incluye la carga de archivos específicos para cada plataforma (google-services.json para Android y GoogleService-Info.plist para iOS), lo que facilita la autenticación segura y la sincronización de datos en la nube.

La gestión de estado con Riverpod se centraliza en **core/di**, donde se definen providers para autenticación, perfil, menús, lista de compra y ajustes. Esto permite una inyección de dependencias limpia, con overrides para entornos de prueba, mejorando la testabilidad y la modularidad.

El sistema de temas se basa en ThemeMode.system, con temas claro y oscuro personalizados que utilizan un color primario verde (#4CAF50) para transmitir salud y nutrición. La internacionalización se integra directamente en MaterialApp mediante delegados y archivos .arb, ofreciendo soporte nativo para español e inglés sin impacto en el rendimiento.

Las notificaciones push se configuran con Firebase Messaging para manejar eventos en foreground y background, enfocadas en respuestas del soporte para una interacción en

tiempo real. La navegación con go_router incluye rutas protegidas, redirecciones basadas en estado de autenticación y soporte para bottom navigation persistente.

Esta configuración no solo optimiza el rendimiento multiplataforma, sino que también facilita futuras expansiones, como integración de más proveedores o funcionalidades offline.

ELEMENTOS DESTACABLES DEL DESARROLLO

Retos técnicos y problemas afrontados

El desarrollo de SmartMeal ha implicado varios retos técnicos que han requerido investigación profunda, múltiples iteraciones y la búsqueda de soluciones creativas. La integración de inteligencia artificial generativa mediante google_generative_ai para la creación de menús personalizados, junto con el uso de una API externa para la obtención de macros nutricionales, planteó importantes desafíos relacionados con la seguridad, la latencia y la gestión de claves API. Exponer dichas claves directamente en el cliente Flutter representaba un riesgo significativo, mientras que las llamadas directas podían verse afectadas por límites de cuota o problemas de conectividad.

Otro aspecto complejo fue el cálculo preciso de las necesidades nutricionales mediante la fórmula Harris-Benedict revisada, incluyendo los ajustes por nivel de actividad física y objetivo del usuario. En las primeras versiones se detectaron pequeñas discrepancias respecto a tablas de referencia estándar, lo que obligó a un proceso exhaustivo de depuración y validación con numerosos casos de prueba.

La gestión de la lista de compra automática optimizada también supuso un reto considerable, especialmente en lo relativo a la eliminación de duplicados, la suma de cantidades y la categorización de ingredientes con unidades variadas, todo ello agravado por la falta de normalización en los nombres de los alimentos.

El soporte de imágenes tanto en el perfil del usuario como en las recetas generó dificultades relacionadas con los permisos, la compresión adecuada y la carga asíncrona, asegurando al mismo tiempo una experiencia fluida en ambas plataformas sin sobrecarga de memoria.

La configuración de notificaciones push para las respuestas del soporte mediante firebase_messaging requirió un manejo cuidadoso del ciclo de vida de la aplicación y de los tokens FCM, tanto en primer plano como en segundo plano.

Por último, la internacionalización completa de la aplicación resultó laboriosa, ya que hubo que traducir todos los textos, fechas y unidades, sin que ello afectara al diseño ni generara errores en tiempo de ejecución, prestando especial atención a plurales y géneros en español.

Soluciones implementadas

Para superar los retos mencionados se adoptaron soluciones técnicas sólidas y eficientes. En cuanto a la integración de IA y APIs externas, se desplegó un worker de Cloudflare como proxy serverless que gestiona todas las llamadas a Gemini para la creación de menús y a la API de obtención de macros. Esta arquitectura oculta las claves API del cliente, añade una capa de seguridad mediante validación de origen, reduce la latencia gracias al cache y centraliza el manejo de errores. Las peticiones desde Flutter se realizan mediante HTTP al worker, que se encarga de reenviarlas a los servicios correspondientes.

El cálculo nutricional se resolvió creando una clase dedicada en core/utils/ con funciones puras y un amplio conjunto de pruebas unitarias que garantizan su precisión. La lista de compra se optimizó mediante un algoritmo que emplea fuzzywuzzy para coincidencias aproximadas y dartz para un manejo funcional de errores, logrando normalizar nombres y unidades de forma robusta.

En el caso de las imágenes, se implementó compresión automática antes de la subida, almacenamiento de URLs en Firestore y uso de placeholders durante la carga. Las notificaciones push se configuraron con handlers específicos en firebase_messaging para cada estado de la aplicación, integrándose directamente con los documentos de Firestore del chat de soporte.

La internacionalización se llevó a cabo utilizando el sistema l10n de Flutter con archivos .arb (app_es.arb y app_en.arb) ubicados en la carpeta l10n/, lo que permitió traducciones completas y un cambio de idioma dinámico desde la pantalla de ajustes.

Innovaciones y funcionalidades destacadas del proyecto

SmartMeal destaca por incorporar elementos innovadores que la posicionan por encima de muchas aplicaciones similares del mercado. La generación de menús asistida por inteligencia artificial mediante Google Generative AI, gestionada a través de un Cloudflare Worker como proxy seguro, ofrece sugerencias creativas y altamente personalizadas sin comprometer la seguridad ni la escalabilidad.

El sistema de soporte interactivo con notificaciones push en tiempo real proporciona una comunicación fluida entre usuario y administrador, mejorando notablemente la experiencia de uso. La optimización inteligente de la lista de compra, con normalización avanzada de ingredientes, reduce el desperdicio y simplifica la planificación diaria.

El enfoque en accesibilidad y personalización se manifiesta en los temas claro y oscuro, la internacionalización completa mediante l10n con archivos .arb y el soporte para imágenes tanto en el perfil como en las recetas. Finalmente, la arquitectura limpia basada en Provider y GetIt hace que el proyecto sea altamente extensible, preparado para futuras funcionalidades como seguimiento de ingesta real o integración con dispositivos wearables.

Todas estas características convierten a SmartMeal en una herramienta avanzada, moderna y sostenible para la mejora de hábitos alimentarios.

LÍNEAS FUTURAS

El desarrollo de **SmartMeal** ha establecido una base técnica sólida y funcional que permite planificar múltiples mejoras y ampliaciones futuras, tanto en el ámbito funcional como técnico, con la finalidad de consolidar la aplicación como una herramienta integral y altamente competitiva en el campo de la planificación nutricional personalizada.

Desde el punto de vista funcional, una de las líneas más ambiciosas sería el desarrollo de un agente artificial propio especializado en la generación de menús. En lugar de depender exclusivamente de servicios externos como Gemini, se podría entrenar o fine-tunear un modelo de lenguaje ligero (por ejemplo, basado en Llama o Gemma) adaptado

específicamente al dominio nutricional. Este agente propio conocería en profundidad las reglas de equilibrio macronutricional, restricciones dietéticas, variedad cultural y estacionalidad de ingredientes, ofreciendo respuestas más rápidas, coherentes y personalizadas, además de reducir costes a largo plazo y eliminar dependencia de terceros.

Otra mejora significativa sería la implementación del seguimiento diario de la ingesta real del usuario. Actualmente la aplicación se centra en la planificación, pero no registra lo que efectivamente se consume. Permitir al usuario marcar comidas completadas, registrar variaciones o añadir alimentos no planificados facilitaría el cálculo de déficits o excedentes calóricos y permitiría ajustes automáticos en los menús posteriores. Además, se podría habilitar que el usuario marque recetas como favoritas para que el agente IA las priorice en generaciones futuras, o incluso que cree recetas propias, almacenándolas en Firestore y utilizándolas como base para propuestas personalizadas, enriqueciendo el catálogo con contenido generado por el usuario.

Se contempla también la ampliación de preferencias y restricciones alimentarias, incluyendo dietas específicas (vegetariana, vegana, cetogénica, sin gluten, baja en FODMAP) y alergias concretas, integrándolas directamente en la lógica del agente IA para generar propuestas totalmente adaptadas. En relación a la lista de compra, una funcionalidad futura interesante sería la exportación a distintos formatos como PDF, texto plano para notas o incluso integración directa con apps de listas compartidas (ej. Google Keep, Apple Notes o exportación a CSV para Excel), facilitando su uso en otros dispositivos o con familiares.

La integración con ecosistemas de salud externos, como Google Fit, Apple Health o dispositivos wearables, supondría un avance importante al obtener datos automáticos de actividad física y gasto energético, eliminando la necesidad de introducción manual y mejorando la precisión de las recomendaciones.

En cuanto a la experiencia de usuario, sería interesante crear una sección de recetas comunitarias donde los usuarios pudieran compartir, valorar y guardar recetas propias, enriqueciendo el catálogo más allá de las generadas automáticamente. Asimismo, el desarrollo de un modo offline más robusto, con sincronización pendiente al recuperar conexión, aumentaría la utilidad en entornos con cobertura limitada.

Desde la perspectiva técnica, se podría optimizar el rendimiento del worker de Cloudflare incorporando cache avanzado y lógica de fallback local. También sería valioso explorar la visión artificial para el análisis de fotografías de platos, permitiendo registrar la ingesta real mediante captura de imagen.

Finalmente, para garantizar la mantenibilidad a largo plazo, se plantea la introducción de pruebas de integración continuas mediante GitHub Actions, la implementación de monitoreo de errores en producción (por ejemplo, con Firebase Crashlytics) y la preparación para despliegue automático en Google Play Store y App Store.

Todas estas líneas de trabajo posicionarían a **SmartMeal** como una plataforma completa de salud nutricional, combinando planificación inteligente, seguimiento real y asistencia proactiva mediante un agente artificial propio.

CONCLUSIONES

Logros alcanzados

A lo largo de la realización de este Trabajo Fin de Grado se ha desarrollado **SmartMeal**, una aplicación móvil multiplataforma en Flutter que permite la planificación inteligente y personalizada de la alimentación semanal. La aplicación genera menús equilibrados con cuatro comidas diarias (desayuno, comida, snack y cena), adaptados a las características individuales del usuario (edad, sexo, peso, altura, nivel de actividad física) y a su objetivo nutricional (pérdida de peso, mantenimiento o ganancia muscular), utilizando como base científica la fórmula Harris-Benedict revisada para el cálculo de necesidades calóricas y distribución de macronutrientes.

Uno de los logros más destacados ha sido la integración de inteligencia artificial generativa mediante Google Generative AI para la creación de sugerencias creativas de recetas, gestionada de forma segura a través de un Cloudflare Worker que actúa como proxy serverless. Esta solución evita la exposición de claves API en el cliente, reduce la latencia gracias al cache en edge y centraliza el manejo de errores y límites de cuota, permitiendo una experiencia fluida y escalable.

La funcionalidad de lista de compra automática optimizada, con algoritmo que elimina duplicados, normaliza nombres de ingredientes mediante coincidencias aproximadas

(fuzzywuzzy) y suma cantidades de forma inteligente, representa un avance práctico que facilita la planificación real y reduce el desperdicio alimentario.

El sistema de soporte interactivo con chat en tiempo real y notificaciones push inmediatas al recibir respuesta del administrador mejora notablemente la usabilidad, ofreciendo asistencia personalizada. La aplicación también incluye gestión completa de perfil con subida de imágenes, temas claro y oscuro adaptativos, e internacionalización total en español e inglés mediante el sistema l10n con archivos .arb.

Técnicamente, se ha conseguido una arquitectura limpia y modular basada en Provider para la gestión de estado reactivo y GetIt para inyección de dependencias, con clara separación entre capas de presentación, dominio y datos. La integración con Firebase (Authentication, Firestore, Storage y Messaging) ha resultado robusta, garantizando sincronización multi-dispositivo y seguridad.

La aplicación ha sido probada en emulador Android (API 35) y dispositivo físico Xiaomi Redmi Note 8 Pro, demostrando estabilidad, rendimiento y una interfaz intuitiva. Todos los objetivos propuestos se han cumplido, obteniendo un producto funcional, innovador y con alto valor práctico.

Aprendizajes personales

Durante los dos años del ciclo formativo se han vivido experiencias variadas, algunas más complicadas que otras. Al comienzo del primer curso y en parte del segundo hubo momentos de incertidumbre por la ausencia temporal de profesorado en algunos módulos, lo que generó cierto retraso en la adquisición de contenidos previstos. Por ejemplo, el conocimiento sobre bases de datos NoSQL llegó más tarde de lo ideal y hubo menos oportunidades de trabajar en proyectos grupales. Aun así, estos inconvenientes no han empañado el balance global del ciclo, que ha sido claramente positivo. Gracias al esfuerzo y dedicación del profesorado de ambos años se han cubierto las competencias esenciales y se ha conseguido un nivel sólido que compensa aquellas lagunas iniciales.

El primer año estuvo centrado principalmente en Java y MySQL. Como no contaba con experiencia previa en programación, el ritmo pausado y progresivo del principio me resultó muy útil para asentar los conceptos básicos. Procedente de bachillerato de ciencias, donde las asignaturas eran más teóricas y orientadas al razonamiento abstracto

que a herramientas prácticas del día a día, todo lo relacionado con el desarrollo de software me parecía un mundo completamente nuevo y fascinante. No tardé mucho en encontrar mi ritmo y empezar a disfrutar del proceso. Entré en el ciclo porque la informática siempre me había atraído —en casa lo he vivido desde pequeño, con mi madre y mi primo trabajando en el sector— y porque tengo un pensamiento muy lógico, pero no estaba seguro de si realmente me gustaría o se me daría bien. Con el paso de las semanas, al ver cómo unas pocas líneas de código podían crear sistemas funcionales, descubrí que no solo se me daba bien, sino que me apasionaba. La asignatura de programación fue, sin duda, mi favorita del primer año: nunca imaginé que sería capaz de construir las aplicaciones que llegué a hacer, aunque al iniciar el segundo curso ya las veía como algo sencillo, lo que refleja el enorme progreso conseguido.

El segundo año, a pesar de algunas dificultades personales externas al ciclo, ha sido una etapa muy gratificante. He formado un grupo excelente con mis compañeros, con los que he compartido momentos inolvidables y de los que estoy profundamente agradecido. En cuanto al profesorado, he tenido mucha suerte: cada uno con su estilo propio, pero todos han mostrado interés real por nuestro aprendizaje y bienestar. El único punto negativo fue la ausencia de docente durante parte del módulo de Acceso a datos, aunque el contenido clave se pudo recuperar gracias a conceptos vistos en otras asignaturas, por lo que no ha supuesto un déficit importante.

En este segundo año he consolidado competencias muy valiosas: desarrollo móvil con Flutter y Dart, manejo avanzado de bases de datos NoSQL mediante Firebase Firestore, integración de servicios en la nube, creación de proxies serverless con Cloudflare Workers y trabajo con inteligencia artificial generativa a través de Google Generative AI. También he dominado herramientas imprescindibles como Visual Studio Code, por su versatilidad y soporte para múltiples lenguajes y servicios, Android Studio para emulación y pruebas, y la consola de Firebase para la gestión del backend.

He profundizado en patrones arquitectónicos como Clean Architecture aplicada a Flutter y en técnicas de gestión de estado e inyección de dependencias con Provider y GetIt. El contacto con tecnologías en constante evolución me ha hecho consciente de que en este sector no se puede estancarse: lo que hoy es puntero mañana queda obsoleto. Este ciclo me ha confirmado que la programación, especialmente el desarrollo móvil y la integración

de IA, es mi verdadera vocación, y me motiva a seguir formándome y creciendo profesionalmente.

Los conocimientos adquiridos están resultando muy útiles en la búsqueda actual de empleo. Me sorprende la cantidad y variedad de oportunidades que existen en el mundo del desarrollo móvil y full-stack. Durante el curso pensé que algunas materias eran mero relleno, pero ahora veo que todas tienen su aplicación real y son requisitos habituales en ofertas laborales.

Respecto al proyecto de fin de grado, al principio me costó decidir el tema. Consideré usar Kotlin Multiplatform, pero tras investigar comprobé sus limitaciones en cuanto a UI compartida real. Fue entonces cuando descubrí Flutter y su capacidad para generar aplicaciones nativas en Android e iOS desde una única base de código en Dart, lo que encajaba perfectamente con el espíritu multiplataforma del ciclo. Decidí crear una herramienta de planificación nutricional inteligente con generación de menús mediante IA.

Una vez elegido Flutter, opté por una base de datos en la nube porque una solución local no permitiría sincronizar datos entre dispositivos. Tras comparar opciones, Firebase Firestore se impuso por su integración nativa con Flutter, escalabilidad y generoso plan gratuito.

Para la arquitectura, Clean Architecture combinada con Provider y GetIt resultó la elección más adecuada para mantener el código organizado, testable y preparado para futuras ampliaciones.

Con estas decisiones tomadas comencé el diseño de la aplicación. La falta de experiencia previa con Flutter y NoSQL implicó una curva de aprendizaje pronunciada: los primeros avances fueron lentos y requirieron muchas horas de documentación y resolución de errores. Sin embargo, el ritmo fue aumentando y las funcionalidades conseguidas resultaron enormemente satisfactorias: autenticación segura con Firebase, generación automática de menús equilibrados, optimización inteligente de la lista de compra, integración de IA mediante Cloudflare Worker o el correcto funcionamiento de la internacionalización con archivos .arb.

Hubo aspectos que podrían haberse gestionado mejor con un análisis y diseño más detallado desde el principio, como ajustes tardíos en algunas interfaces o una mayor reutilización temprana de widgets comunes.

A lo largo del proyecto he adquirido un volumen importante de conocimiento nuevo: desde el funcionamiento interno de bases de datos NoSQL hasta el dominio práctico de Flutter y Dart, el uso avanzado de Firebase, la integración segura de IA generativa y el despliegue de workers serverless. Aunque quedan funcionalidades que me habría gustado implementar, el resultado final me llena de orgullo.

Una vez terminado, siento que he dado lo mejor de mí para conseguir lo que me propuse. Las decisiones tomadas —Flutter, Firebase Firestore, Clean Architecture con Provider/GetIt y la inclusión de IA mediante proxy seguro— han sido acertadas. SmartMeal es el reflejo del esfuerzo, la dedicación y el entusiasmo por aprender que he puesto durante estos dos años de ciclo.

Evaluación del proyecto

SmartMeal constituye un proyecto completo, funcional e innovador que supera los objetivos establecidos para un Trabajo Fin de Grado en Desarrollo de Aplicaciones Multiplataforma. La aplicación resuelve un problema real y cotidiano con una solución técnica sólida, accesible y moderna, destacando especialmente por la integración segura de IA generativa y la optimización práctica de la planificación alimentaria.

El resultado final es una herramienta útil, estable y con potencial real de uso diario, preparada para futuras ampliaciones. Este TFG representa el culmen de dos años de formación intensiva y demuestra la capacidad para llevar a cabo un proyecto profesional desde la concepción hasta la implementación completa.

BIBLIOGRAFÍA

Grok. (2025). Grok, xAI. Recuperado en enero de 2026. Utilizado como asistente virtual principal durante todo el desarrollo del proyecto para resolver dudas técnicas, generar

código inicial, depurar errores, proponer soluciones arquitectónicas y aclarar conceptos avanzados de Flutter, Firebase y Cloudflare Workers.

Google. (2025). Gemini. Recuperado en enero de 2026. Empleado para contrastar información, obtener explicaciones alternativas sobre patrones de arquitectura y validar prompts para la integración de inteligencia artificial generativa.

Flutter. (2025). Documentación oficial de Flutter. Recuperado en enero de 2026. Fuente principal para el aprendizaje y referencia constante de widgets, navegación con go_router, gestión de estado, internacionalización con l10n y buenas prácticas de desarrollo multiplataforma.

Firebase. (2025). Documentación oficial de Firebase para Flutter. Recuperado en enero de 2026. Utilizada para la configuración completa de Firebase Core, Authentication, Cloud Firestore, Cloud Storage y Cloud Messaging, así como para la resolución de problemas específicos de sincronización y notificaciones push.

Cloudflare. (2025). Cloudflare Workers Documentation. Recuperado en enero de 2026. Referencia esencial para el despliegue y configuración del worker como proxy seguro para llamadas a Google Generative AI y APIs de macros, incluyendo gestión de caché, seguridad y rate limiting.

Google AI. (2025). Google Generative AI Developer Documentation. Recuperado en enero de 2026. Consultada para la integración del paquete google_generative_ai, diseño de prompts eficientes y manejo de respuestas estructuradas en la generación de menús.

YouTube. (2025). Clean architecture en Flutter. Recuperado en enero de 2026. Utilizado como referencia visual para comprender y aplicar Clean Architecture en proyectos Flutter, junto con patrones de Provider y GetIt.

YouTube. (2025). Curso Cloudflare DESDE CERO. Recuperado en enero de 2026. Video empleado para entender el concepto de serverless en edge y su aplicación como proxy seguro en proyectos móviles.

Pub.dev. (2025). Paquetes Flutter: fuzzywuzzy, dartz, image_picker, shared_preferences. Recuperado en enero de 2026. Documentación consultada para la implementación de

coincidencias aproximadas en ingredientes, manejo funcional de errores y persistencia local de preferencias.

Mermaid Live Editor. (2025). Mermaid.js. Recuperado en enero de 2026. Utilizado para generar diagramas de flujo, entidad-relación y arquitectura mediante código, facilitando su inclusión en la documentación.

Stack Overflow. (2025). Comunidad Stack Overflow. Recuperado en enero de 2026. Plataforma consultada frecuentemente para resolver problemas específicos de implementación en Flutter, Firebase y Cloudflare Workers.

ANEXOS

Repositorio de GitHub

Enlace al repositorio

Manual de usuario

Enlace al manual de usuario

Presentación del TFG

Enlace a la presentación