Foodlisting

Carlos Martínez Sánchez
Máster universitario en Desarrollo de aplicaciones para dispositivos móviles
Trabajo final de Máster

Pau Dominkovics Coll
Carles Garrigues Olivella

Junio de 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	Foodlisting
Nombre del autor:	Carlos Martínez Sánchez
Nombre del consultor:	Pau Dominkovics Coll
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo

La finalidad de este proyecto es gestionar de manera eficiente las comidas de un usuario. Mediante una organización semanal, se genera automáticamente su lista de la compra, lo que le permite crear esta lista a través de los platos que cocina o artículos que consume semanalmente. Aunque existen otras aplicaciones en el mercado que permiten almacenar tus platos, ninguna permite realizar ninguna gestión temporal.

Se ha creado una aplicación para dispositivos iOS desde cero, donde se han aplicado y ampliado los conocimientos obtenidos en el máster.

El desarrollo de este proyecto ha seguido una planificación en varias fases, aplicando estas mediante una metodología en cascada. Cabe destacar, por ejemplo, el diseño centrado en el usuario y la selección de una arquitectura MVVM.

En la fase de implementación, se ha utilizado el lenguaje Swift y también se ha hecho uso de algunos elementos propios de los sistemas iOS, como el almacenamiento local mediante CoreData. Además, se ha incluido la conexión con iCloud para proporcionar a los usuarios una sincronización de su contenido entre sus dispositivos.

Al finalizar este proyecto, puede considerarse que se han cumplido los objetivos, proporcionando una aplicación que permite crear artículos, platos, listas y asignar los elementos que el usuario quiera a sus días de la semana. Esto le permite organizar su alimentación de manera más eficiente.

Abstract

The purpose of this project is to efficiently manage a user's meals. Using a weekly organization, your shopping list is automatically generated, allowing you to create this list from the dishes you cook or items you consume on a weekly basis. Although there are other applications on the market that allow you to store your dishes, none allow any temporary management.

An application for iOS devices has been created from scratch, where the knowledge obtained in the master has been applied and expanded.

The development of this project has followed a planning in several phases, applying these through a cascade methodology. Noteworthy, for example, is the user-centered design and the selection of an MVVM architecture.

In the implementation phase, the Swift language has been used and some elements of iOS systems have also been used, such as local storage through CoreData. In addition, the connection to iCloud has been included to provide users with a synchronization of their content between their devices.

At the end of this project, it can be considered that the objectives have been met, providing an application that allows you to create articles, dishes, lists and assign the elements that the user wants to their days of the week. This allows you to organize your eating more efficiently.

Palabras clave

comida, platos, artículos, calendario, organización, listas, compra

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy es ya difícil encontrar en un supermercado a una persona con una lista de la compra escrita a mano. Los smartphone se han convertido una herramienta muy útil y los utilizamos para todo lo que podemos.

Como resulta evidente, nuestra lista de la compra depende en gran parte de los platos que cocinamos, por lo que generamos esta lista, de alguna forma, a partir de lo que pretendemos cocinar. Sin embargo, hacer listas para no tener que ir al supermercado una y otra vez y optimizar nuestro tiempo, con frecuencia no resulta tan eficiente. Es fácil olvidarse de algún ingrediente y tener que volver a la compra, con el tiempo que esto requiere. Como en muchas otras cuestiones, la mejor forma de hacer este proceso más eficiente es la planificación. Planificando los platos que vamos a cocinar durante un periodo de tiempo nos hace ahorrar tiempo en los días de la semana en los que más ocupados estamos.

La razón de esta aplicación es aumentar la productividad de los usuarios, permitiendo asignar platos a los días de la semana y generar a partir de estos una lista de la compra. Esta lista debe poder ser modificada por el usuario para añadir o eliminar productos según sus necesidades más allá de las comidas. Del mismo modo, a partir de unos ingredientes, deben mostrarse los platos que el usuario puede elaborar con estos, indicando los artículos que le faltan para completarlo.

Los platos serán creados por el usuario, ya que esto permite una total personalización en los ingredientes y elaboración de los mismos. Cada uno elabora sus comidas de manera diferente, por lo que esta es una característica importante, en contrapunto a seleccionar platos de una lista dada.

Para darle una mayor utilidad a esto, las listas podrán ser compartidas, de tal forma que varios usuarios puedan ver y modificar una lista.

Con características que podríamos considerar similares, podemos encontrar varias aplicaciones. Se va a tomar un subconjunto de ellas para su análisis. Además, se elaboran unas tablas de puntos fuertes y débiles en base a características que nuestra aplicación ofrece: uso sin registro, crear ingredientes, platos y listas, uso de forma gratuita e implementación de modo oscuro.

1. Bring! Lista de la compra es una aplicación que permite la creación de listas de la compra, platos y artículos. Cuenta con un diseño atractivo y es muy conocida en la App Store. Además de estas funciones, permite compartir recetas entre usuarios y ofrece sugerencias. Como punto negativo, es necesario registrarse para poder utilizar la aplicación.

Figura 1: Bring! Lista de la compra

Bring! lista de la compra

Puntos fuertes	Puntos débiles
Permite crear platos personalizados	Exige registro para poder utilizarse
Permite crear artículos personalizados	No diferencia entre modo claro y oscuro
Gratis	
Permite crear listas	

Tabla 1: Aplicación Bring! Lista de la compra [1]

2. Mi lista de la compra fácil permite crear listas y puede utilizarse de manera gratuita. Sin embargo, no permite crear artículos ni platos. Como punto positivo, puede utilizarse sin registro.

Figura 2: Mi lista de la compra fácil

Mi lista de la compra fácil

Puntos fuertes	Puntos débiles
Permite crear listas	No permite crear platos personalizados
No requiere registro	No permite crear artículos personalizados
Gratis	No tiene modo oscuro

Tabla 2: Aplicación Mi lista de la compra [2]

3. Noodle: recetas sanas fácil es una aplicación sobre recetas, pero contiene funcionalidades similares a la nuestra. Permite crear listas y platos, aunque para hacer esto requiere una suscripción. Además, sugiere platos sanos, con su información nutricional y el estado en el que deben estar los ingredientes para una buena preparación.

Figura 3: Noodle, recetas sanas fácil

Noodle: Recetas sanas fácil

Puntos fuertes	Puntos débiles
Permite crear listas	Requiere pago para crear los platos y listas
Permite crear platos	No permite crear artículos personalizados
No requiere registro	No tiene modo oscuro

Tabla 3: Aplicación Noodle: Recetas fáciles [3]

4. Kitchen Stories Recipes es una aplicación de recetas que guía a los usuarios paso a paso para elaborar un plato. Permite crear listas, aunque están fuertemente asociadas a los platos, de tal forma que no pueden añadirse o quitarse elementos de la lista. Como punto negativo, requiere registro para poder hacer uso de estas funciones.

Figura 4: Kitchen Stories Recipes

Kitchen Stories Recipes

Puntos fuertes	Puntos débiles
Permite crear listas, aunque se crean según los platos	El registro es obligatorio para crear platos
Permite crear los ingredientes en los platos	El registro es obligatorio para crear ingredientes
Gratis	
Modo oscuro	

Tabla 4: Aplicación Kitchen Stories Recipes [4]

Tras el análisis de estas aplicaciones, se observa que no existe en ninguna de ellas ningún método de planificación. Por lo general, estas aplicaciones sugieren platos y guían al usuario en su preparación. Además, algunas cuentan con limitaciones a la hora de crear ingredientes, listas o platos.

El objetivo de nuestra aplicación no es tanto la preparación del plato, sino la planificación de las comidas y la lista de la compra. En los días de diario, no es habitual experimentar con nuevos platos, por lo que una explicación de estos puede tener menos sentido. Aún así, se debe ofrecer al usuario la posibilidad de tomar notas sobre el plato por si este fuese menos habitual.

1.2 Objetivos del Trabajo

- Poner en práctica los conocimientos adquiridos durante el máster.

- Ofrecer al usuario una aplicación que le permita gestionar sus platos, ingredientes y listas de la compra de manera sencilla.

- Asegurar la mantenibilidad y escalabilidad, permitiendo así ampliar la funcionalidad de la aplicación de manera sencilla.

1.2.1 Objetivos funcionales

- Crear, modificar y eliminar platos, con ingredientes y comentarios por si el usuario quisiera tomar alguna nota sobre el plato.

- Asignar platos a comidas y cenas en un día de la semana.

- Generar una lista de la compra para una semana a partir de los platos asignados a esa semana, pudiendo añadir o eliminar elementos de esta lista.

- Almacenar platos, ingredientes y calendario del usuario de manera local.

- La aplicación permitirá su uso sin registro.

- Registro en la aplicación para poder compartir las listas con otros usuarios

1.2.2 Objetivos no funcionales

- La aplicación debe tener un diseño centrado en el usuario con elementos propios del sistema para el que se desarrolla.

- Crear test para verificar el funcionamiento de las funcionalidades de la aplicación.

- El desarrollo será *open source*, publicando el código en GitHub al finalizar el proyecto.

- Implementar el modo oscuro. [5]

- Ofrecer soporte a varios idiomas. [6]

1.2.3 Objetivos personales

- Ampliar los conocimientos y habilidades sobre el desarrollo de aplicaciones móviles en el sistema en el que se desarrolla.

- Ampliar las habilidades de diseño de aplicaciones móviles.

- Mejorar la capacidad de generar documentación formal.

1.3 Enfoque y método seguido

La aplicación se desarrollará desde cero, por lo tanto, se trata de un producto nuevo.

Entre las posibilidades a la hora de elegir un sistema con el que desarrollar la aplicación, se encuentran las aplicaciones nativas e híbridas. En este caso se opta por una aplicación nativa, ya que permite una mayor especificación a la hora de ofrecer una buena experiencia de usuario. Dentro de las opciones para aplicaciones nativas, pueden valorarse las opciones de Android o iOS. En este caso, la aplicación se desarrollará para sistemas iOS.

Aunque iOS es un sistema operativo móvil menos utilizado, dada su fuerte barrera económica de entrada, cuenta con algunas ventajas a tener en cuenta, siendo la fragmentación la más relevante a la hora de desarrollar.

Android cuenta con infinidad de dispositivos, lo que puede generar dificultades a la hora de adaptar la aplicación a todos ellos y puede suponer también una gran dificultad al hacer pruebas. La dificultad para probar la aplicación en un número relevante de dispositivos objetivo hace que ésta sea menos fiable. Además, la actualización a las nuevas versiones de los sistemas Android es mucho más lenta, lo que posterga enormemente la popularización de nuevas características que puedan ir implementándose. Apple soluciona esto teniendo un entorno más cerrado, en el que los usuarios actualizan rápido [7], de tal forma que tanto usuarios como desarrolladores pueden disfrutar de las últimas características del sistema.

Dentro de iOS es posible programar en dos distintos lenguajes: Objetive-C y Swift [8]. En este caso se ha elegido el lenguaje Swift, por ser más moderno y porque su utilización crece cada año en detrimento de Objetive-C.

Otro elemento a tener en cuenta es el framework sobre el que se desarrolla la aplicación. Hace 2 años, Apple anunciaba en la WWDC una nueva manera de desarrollar aplicaciones para iOS: SwiftUI [9]. Este framework permite crear aplicaciones de forma completamente diferente a cómo se venía haciendo hasta ahora mediante UIKit. Al ser una forma muy novedosa aún habrá que esperar un tiempo para que veamos su adopción masiva en el desarrollo de iOS. Por esto, UIKit seguirá siendo importante durante mucho tiempo y es importante aprender cómo funciona. Además, existen aún muchas funciones de UIKit que no están implementadas en SwiftUI. Esta aplicación se desarrollará en UIKit.

En cuanto a la arquitectura existen varias opciones: MVC, MVVM y VIPER, además de alguna variante de estas. Apple propone la utilización de MVC, pero este enfoque hace que los controladores puedan crecer demasiado y dificulta la escritura de tests. Por otro lado, VIPER es una arquitectura más compleja, que podría conllevar demasiada abstracción para un proyecto pequeño. Por todo esto, la arquitectura utilizada será MVVM.

Además, toda la aplicación se escribirá en Swift y esto incluye las vistas, lo que indica que no se utilizará el *Interface Builder* y la definición de la posición de las vistas irá en los *UIViewController*.

En el diseño del interfaz de usuario, se utilizarán en la medida de lo posible los estándares de Apple, definidos en *"Human Interface Guidelines"* [10], para poder de esta forma generar un diseño acorde a las costumbres de un usuario de iOS.

Por último, para realizar el registro en la aplicación y poder compartir las listas, se utilizará Firebase [11]. Esta plataforma de Google facilita, mediante librerías, la implementación de un back-end sencillo. Además, para mantener la privacidad de los usuarios, el registro se hará mediante *"Sign in with Apple"* [12], ya que de esta forma puede ocultarse el correo electrónico real de un usuario al registrarse. Este método es además de obligada implementación para aplicaciones con registro desde el 30 de junio de 2020. [13].

1.4 Planificación del Trabajo

Los recursos utilizados para el desarrollo de este trabajo son los siguientes:

- MacBook Pro 13” con un procesador Intel Core i5 de 4 núcleos (2,4GHz), 16GB de memoria RAM y gráficos Intel Iris Plus Graphics. El sistema operativo utilizado será el más actual disponible, MacOs Big Sur.

- iPhone XS con la última versión de iOS disponible en cada momento, actualmente iOS 14.

- Xcode: IDE de desarrollo y pruebas en distintos emuladores.

- Git: para el control de versiones.

- Figma: herramienta de diseño para la creación de las interfaces de la aplicación y también algunos elementos de la presentación, como las tablas.

- Draw.io: Para la creación de diagramas.

- Microsoft Office: para la creación de la documentación.

En cuanto a los recursos temporales, teniendo en cuenta que es necesario complementar este proyecto con un trabajo a jornada completa, la disponibilidad en número de horas es la siguiente:

- De lunes a viernes: 2 horas diarias
- Sábados, domingos y festivos: 5 horas diarias

Esto suma en total 309 horas, teniendo en cuenta los tres festivos que se producen durante el proyecto, que deberán ser distribuidas entre las tareas que lo componen.

Dada la división del trabajo en cuatro entregas parciales y que debe ser desarrollado por una sola persona, esto se llevará a cabo de manera secuencial. Se diferenciarán los hitos según las entregas parciales, con las siguientes fechas por cada tarea:

- PEC 1: del 17/02/2021 al 10/03/2021
- PEC 2: del 11/03/2021 al 31/03/2021
- PEC 3: del 01/04/2021 al 12/05/2021
- Entrega final: del 13/05/2021 al 02/06/2021

A partir de estos recursos es ha elaborado el siguiente diagrama de Gantt.

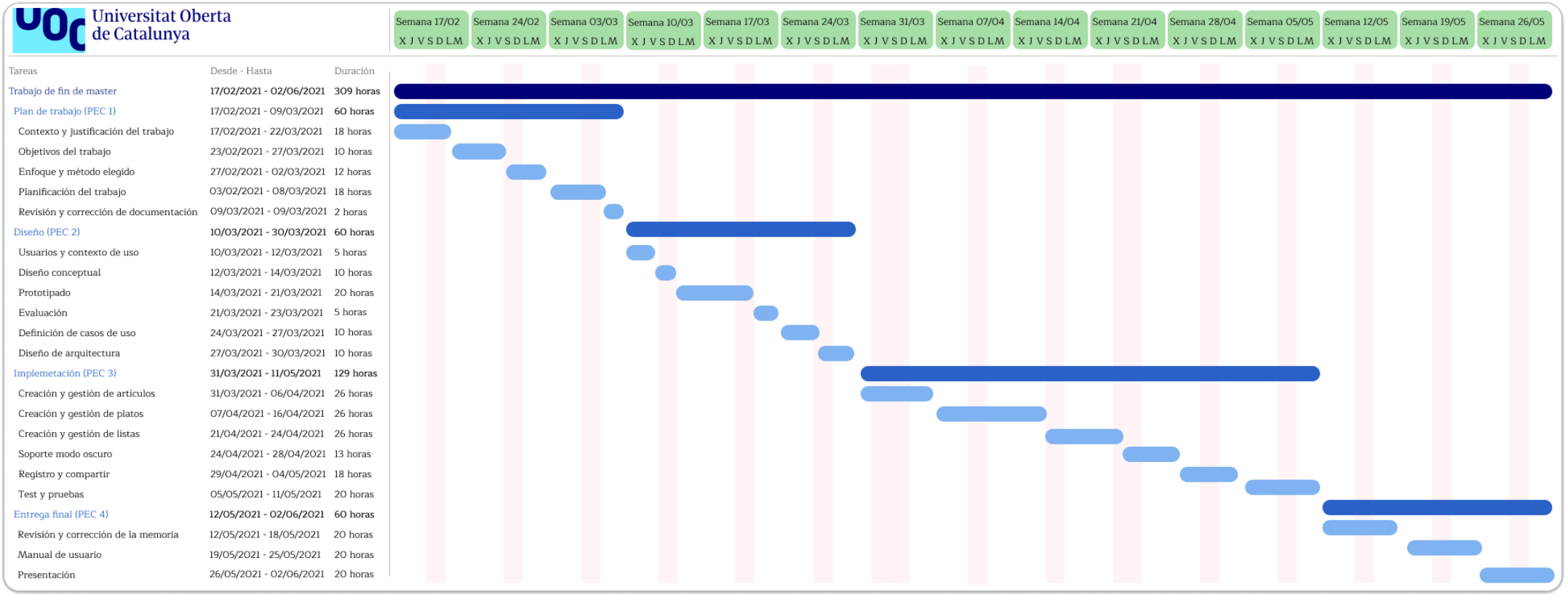

Figura 5: Diagrama de Gantt

En todo proyecto pueden surgir imprevistos o que se alargue alguna tarea mas allá del tiempo que se le asigna. Por esto, en la implementación se han colocado al final de la planificación las tareas de soporte al modo oscuro y registro y compartir. En caso de que no haya tiempo suficiente, estas tareas podrían no implementarse, ya que la funcionalidad base de la aplicación se podría conseguir implementándose el resto.

1.5 Breve sumario de productos obtenidos

- Memoria del proyecto
- Código fuente
- Presentación del proyecto
- Manual de usuario
- Demostración de la aplicación

1.6 Breve descripción de los otros capítulos de la memoria

- **Diseño gráfico**: contiene el diseño centrado en el usuario mediante la realización de una encuesta. A partir de esta se elaboran personas y escenarios, pasando después al diseño gráfico de las pantallas de la aplicación.
- **Diseño técnico de la aplicación**: contiene los casos de uso, diagrama de clases y arquitectura de la aplicación.
- **Implementación:** expone las herramientas utilizadas y los aspectos más relevantes ocurridos en la implementación de la aplicación.
- **Conclusiones:** presenta las conclusiones del trabajo.
- **Líneas de trabajo futuro:** contiene las posibles mejoras que podrían incluirse en la aplicación.
- **Glosario**: recoge términos utilizados en la memoria que puedan requerir aclaración.
- **Bibliografía**: recoge las fuentes consultadas y las referencias a ellas.
- **Anexos:** contiene el manual de usuario y las instrucciones de compilación de la aplicación.

2. Diseño gráfico

En esta sección se presentará el diseño centrado en el usuario de la aplicación. La evaluación de este diseño se hará de forma iterativa.

2.1 Usuarios y contexto de uso

2.1.1 Encuesta

El diseño centrado en el usuario requiere que los usuarios estén involucrados durante todo el proceso. El primer paso es conocer a los usuarios, así como sus necesidades y objetivos. Para conseguir esto se ha elaborado una encuesta. A continuación, se presentan las preguntas de esta encuesta con los resultados de cada pregunta y su justificación.

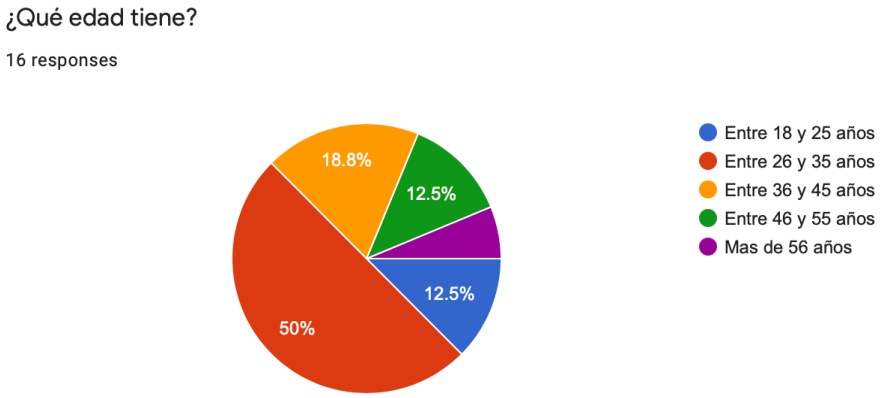

Figura 6: Pregunta 1 de la encuesta

La primera pregunta muestra el rango de edad en el que se encuentra el usuario, ya que actuarán de manera diferente según su edad, y sus hábitos de compra también serán diferentes. El resultado muestra que hay usuarios de todos los rangos de edades, aunque con una notable mayoría de los encuestados se encuentra entre 26 y 35 años.

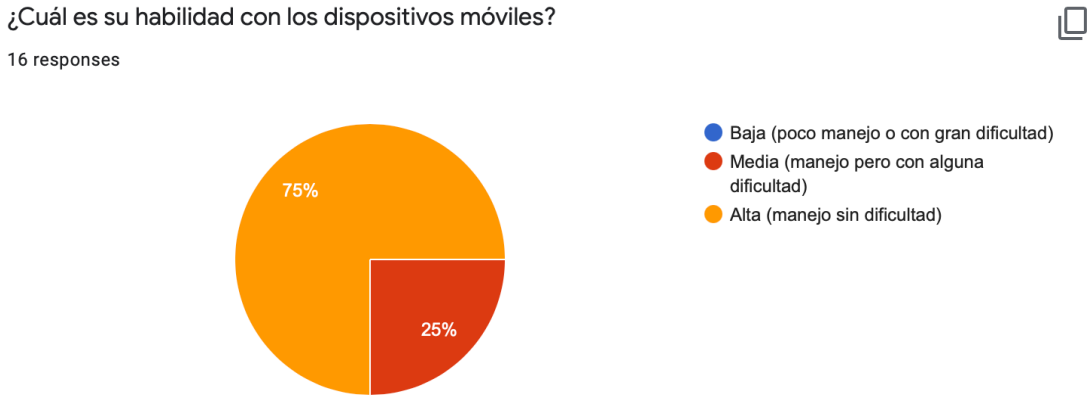

Figura 7: Pregunta 2 de la encuesta

La segunda pregunta intenta encontrar la habilidad de los usuarios con los dispositivos móviles en general. Cuanto más hábiles sean utilizándolos, más probable será que los utilicen para más tareas. La mayoría de los usuarios utilizan los dispositivos sin dificultad.

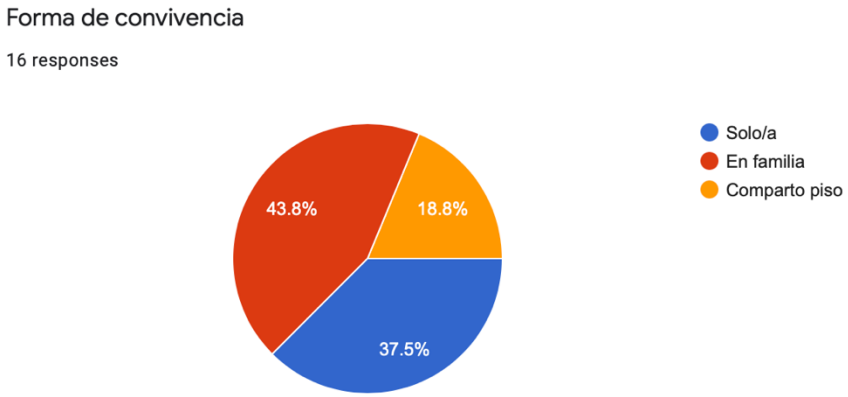

Figura 8: Pregunta 3 de la encuesta

La forma de convivencia afecta a cómo se hace la compra o con qué frecuencia. Quien vive solo debe encargase si o si de la compra, pero quien comparte piso o vive en familia puede hacer esta tarea por turnos. La forma de organizarse cambia. Por otro lado, las personas que no vivan solas podrán encontrar de valor la opción de compartir las listas.

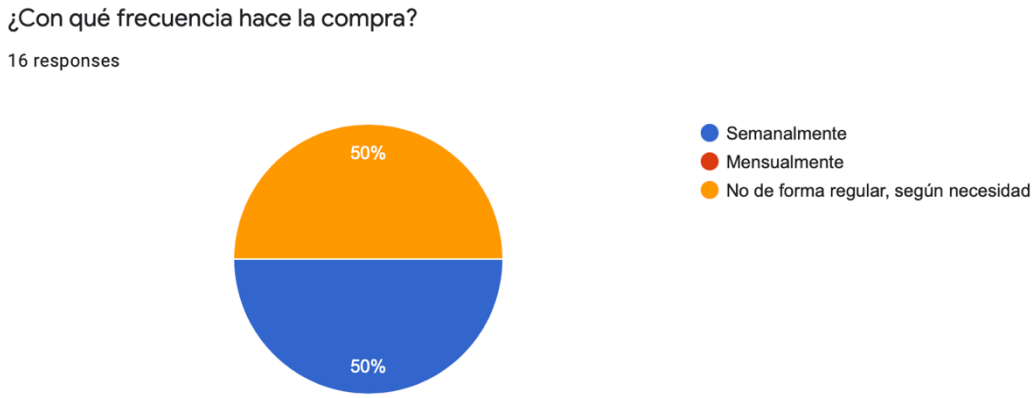

Figura 9: Pregunta 4 de la encuesta

La frecuencia con la que los usuarios hacen la compra muestra la regularidad o falta de ella a la hora de hacerlo, así como la planificación o no de esta. Los resultados muestran que la mitad de los usuarios hacen compra semanalmente, mientras que la otra mitad no parece llevar ninguna planificación.

Figura 10: Pregunta 5 de la encuesta

Esta pregunta también intenta descubrir la planificación que llevan a cabo los usuarios. Como se puede ver, la mayoría no es regular a la hora de hacer sus compras.

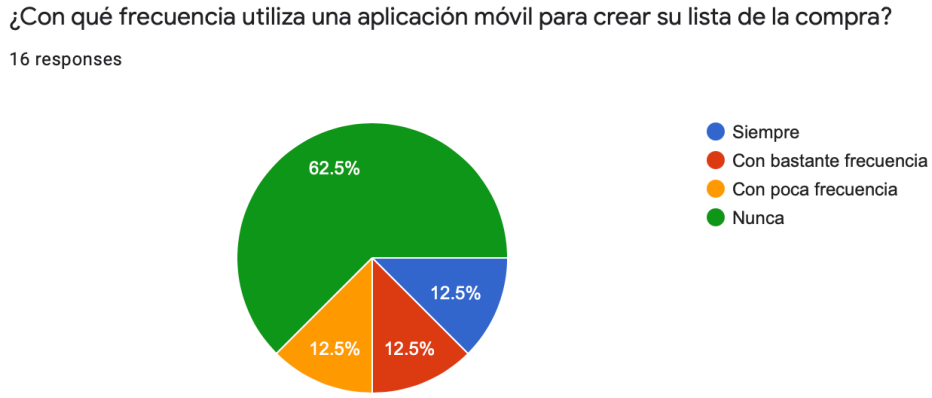

Figura 11: Pregunta 6 de la encuesta

En la pregunta número 6 se busca conocer si los usuarios ya utilizan alguna aplicación móvil para crear sus listas de la compra. El hecho de que la mayoría no utilicen una aplicación móvil puede resultar llamativo. Esto puede deberse a que no utilicen listas o que las utilicen con métodos más tradicionales.

Para intentar clarificar esto, se consultó a algunos encuestados cuál era su método utilizado. Las personas de mayor edad respondieron que utilizan listas en papel cuando se trata de muchos artículos, mientras que no llevan lista cuando son pocos. Por otro lado, las personas que utilizaban alguna aplicación hacían uso de aplicaciones simples para tomar notas como “Notas” o “Google Keep”.

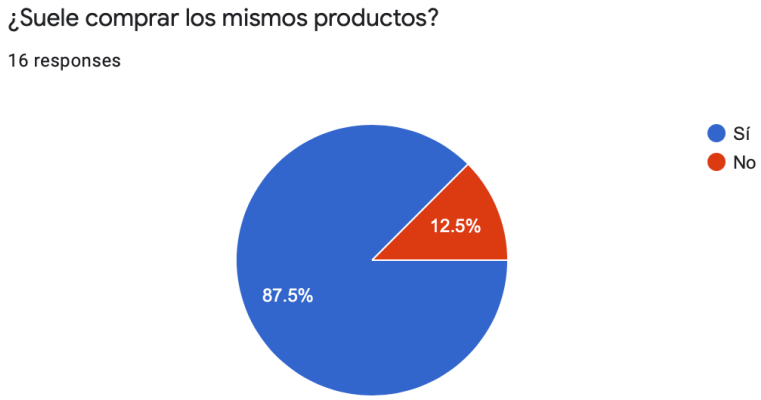

Figura 12: Pregunta 7 de la encuesta

En esta pregunta encontramos que la mayoría de los usuarios suele comprar siempre los mismos productos. Esto puede hacer útil mantener un histórico de listas o incluso poder duplicarlas, de tal forma que se tenga una lista básica con los artículos que siempre se compran y pueda modificarse para cada semana según necesidad.

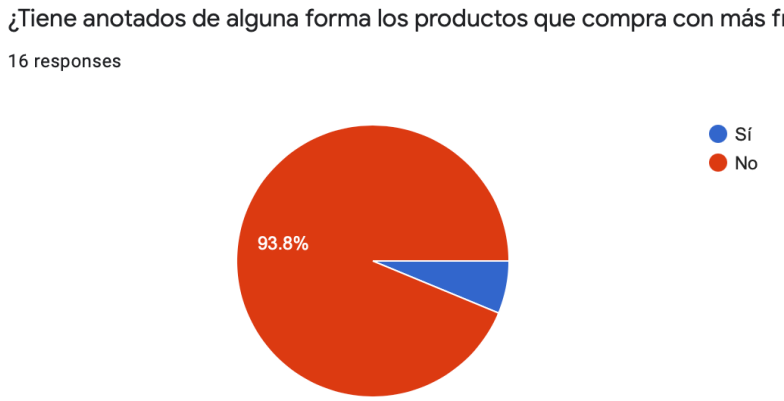

Figura 13: Pregunta 8 de la encuesta

Tener anotados los productos que se compran con frecuencia puede evitar que se olviden algunos de ellos. A la hora de hacer la lista frente a una nevera vacía, es fácil que algunos artículos se olviden. La mayoría de usuarios no tiene anotados sus productos más frecuentes.

Figura 14: Pregunta 9 de la encuesta

Enlazando con la pregunta anterior, se cuestiona si resultaría útil tener anotados los artículos más frecuentes. Solo un 13,3% de los usuarios no consideraría útil esto.

Figura 15: Pregunta 10 de la encuesta

Nuestra aplicación resultará más útil a quienes planifiquen sus comidas, ya que esto nos permitirá generar sus listas. Aunque muchos no parecen hacer ninguna planificación, existe una mayoría de ellos que lo hacen, al menos, de vez en cuando.

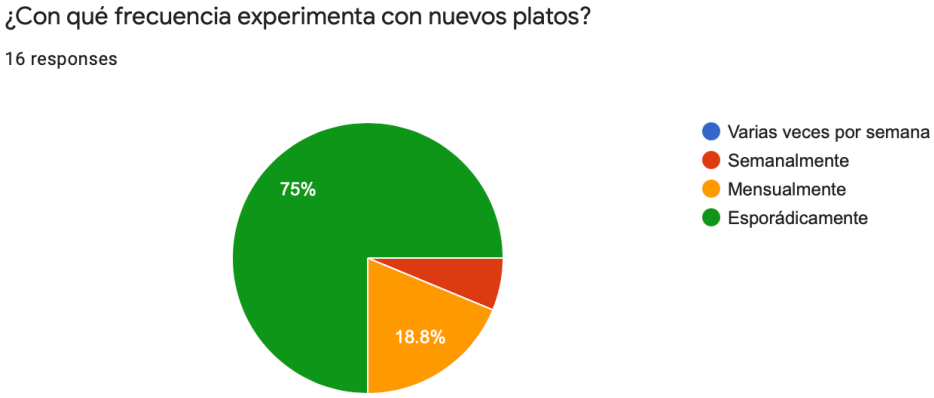

Figura 16: Pregunta 11 de la encuesta

En la pregunta número 11, se trata de encontrar la frecuencia con la que los usuarios experimentan con nuevos platos. La mayoría de ellos lo hacen esporádicamente. Este es un resultado que podría esperarse, ya que la mayoría de los días, no tenemos tiempo para dedicar a esto. Además, llevar una rutina facilita la planificación.

Figura 17: Pregunta 12 de la encuesta

Al igual que ocurría con los artículos, tener apuntados los platos que se cocinan con más frecuencia puede evitar que se olviden estos a la hora de hacer la lista de la compra. Como también ocurría con los artículos, la mayoría de usuarios no tiene anotados sus platos más frecuentes.

Figura 18: Pregunta 13 de la encuesta

En relación a la anterior pregunta, una vez más, una mayoría de los usuarios considerarían útil tener anotados sus platos más frecuentes junto con los ingredientes necesarios, algo que ofrecerá nuestra aplicación.

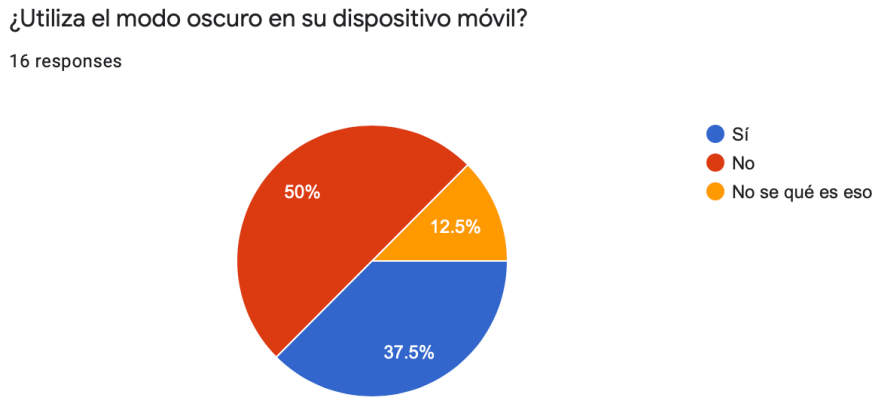

Figura 19: Pregunta 14 de la encuesta

La utilización o no del modo oscuro es más una fijación personal. Para alguien que lo utiliza puede resultar molesto cuando una aplicación no lo implementa. En las aplicaciones móviles este modo lleva ya en funcionamiento algunos años, por lo que toda aplicación nueva debería implementarlo. Como puede verse en el resultado, hay muchos usuarios que ya utilizan esta característica.

Figura 20: Pregunta 15 de la encuesta

Por último, se trata de conocer cuales son las características de las aplicaciones que más valoran los usuarios. Como podemos ver, la facilidad de uso es la más demandada. Esto reafirma la necesidad de un diseño centrado en el usuario. Por otro lado, podemos destacar la importancia cada que varios encuestados dan a la privacidad, característica cada vez más relevante para los usuarios.

La encuesta es accesible desde el siguiente enlace:
https://forms.gle/jvqyK5hGfQttH54S6

2.1.2 Personas

Basándonos en los resultados y conclusiones de la encuesta, elaboramos los perfiles de personas que podrían utilizar nuestra aplicación.

Figura 21: User-Persona Lucía

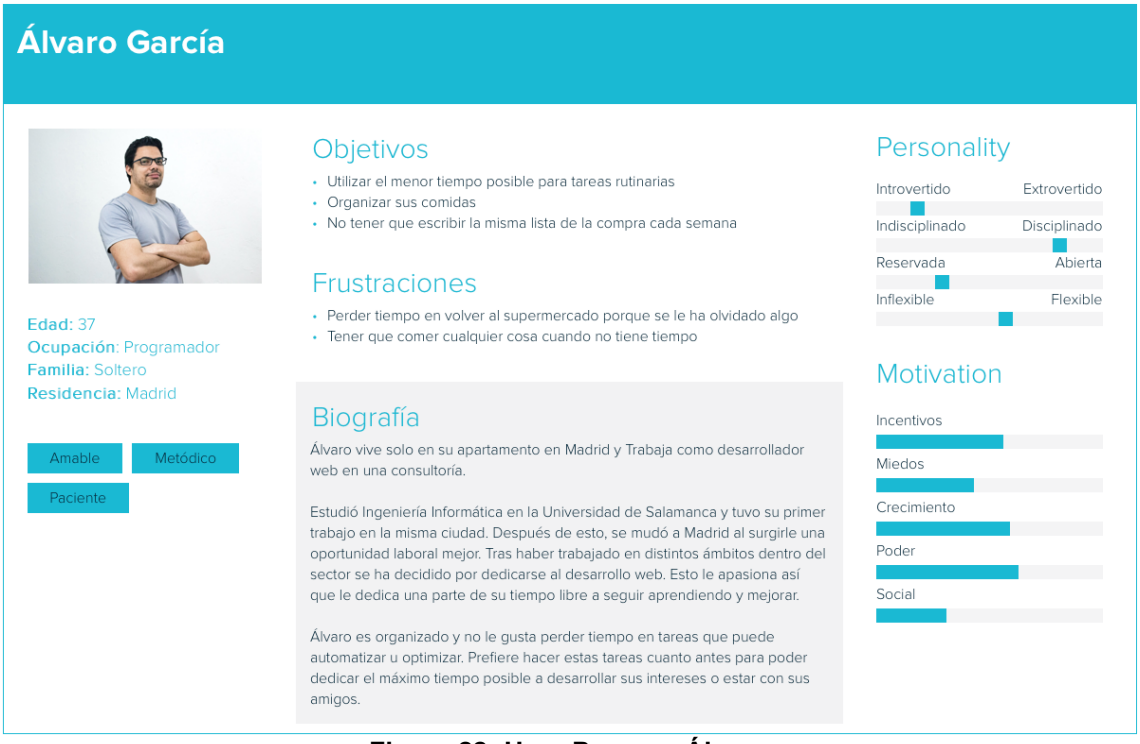

Figura 22: User-Persona Álvaro

2.2 Diseño conceptual

2.2.1 Escenarios

A continuación, se presentan los escenarios elaborados para cada persona.

Escenario de Lucía:
Lucía ha descargado una nueva aplicación para organizarse mejor con sus listas de la compra y sus platos. En ella incluye los platos que cocina con más frecuencia. Le gusta experimentar con nuevos platos con sus compañeros de piso los fines de semana, pero los días de diario elabora comidas más simples porque no tiene tiempo.
Unos minutos antes de ir a la compra, Lucía organiza las comidas de su semana y la aplicación genera su lista de la compra. Añade otros productos que necesita, como pasta de dientes.
Al día siguiente llega a casa con mucha prisa, pero esta vez sabe qué hacer y tiene todo lo que necesita.

Escenario de Álvaro:
Álvaro utiliza la aplicación “Notas” para hacer su lista de la compra. Recorre su cocina para hacer su lista y anota las cosas que se da cuenta que le faltan. Al volver, se da cuenta de que se le ha olvidado algo que necesita para la comida de hoy. Frustrado, vuelve al supermercado y lo compra.
La semana siguiente, antes de ir prueba una nueva aplicación para organizarse mejor, viendo qué es lo que realmente necesita. Anota los platos que quiere hacer y los distribuye durante la semana. Cuando se genera la lista, recorre su cocina y elimina de la lista lo que ya tiene. Esta vez solo tendrá que ir una vez y no se le olvidará nada.

Volviendo a los resultados de la encuesta, podría crearse otra persona y escenario para reflejar las personas que no ven necesario organizarse de la manera que sugiere la aplicación, que no consideran útil anotar sus platos o que simplemente prefieren seguir utilizando métodos más tradicionales. Para resumir, solamente se han creado las personas que representan usuarios potenciales. Esto se menciona para reflejar que se ha tenido en cuenta.

2.2.2 Funcionalidades principales

- La aplicación debe permitir asignar platos a días de la semana.
- La aplicación debe permitir crear y editar listas de la compra, tanto automáticas en función de los platos asignados a los días de una semana, como las que pueda crear el usuario.
- La aplicación debe permitir crear y editar platos para que puedan ser reutilizados fácilmente.
- La aplicación debe permitir crear y editar artículos para que puedan ser reutilizados fácilmente.
- Tratándose de una aplicación para aumentar la productividad, las opciones principales de la aplicación deben estar disponibles fácilmente.
- La interfaz debe ser simple, para permitir leer una lista o el contenido de un plato con rapidez.
- La aplicación debe poder usarse sin registro. Opcionalmente puede ofrecerse la opción de registrarse.
- La aplicación debe almacenar el contenido del usuario localmente, protegiendo así su privacidad y asegurando el total funcionamiento de la aplicación sin conexión. Opcionalmente, puede ofrecerse la posibilidad de almacenamiento online mediante el registro.

2.2.3 Árbol de navegación

En la siguiente figura se muestra el árbol de navegación de la aplicación creado en Figma. En este diagrama pueden verse los diferentes caminos que puede seguir un usuario al utilizar la aplicación.

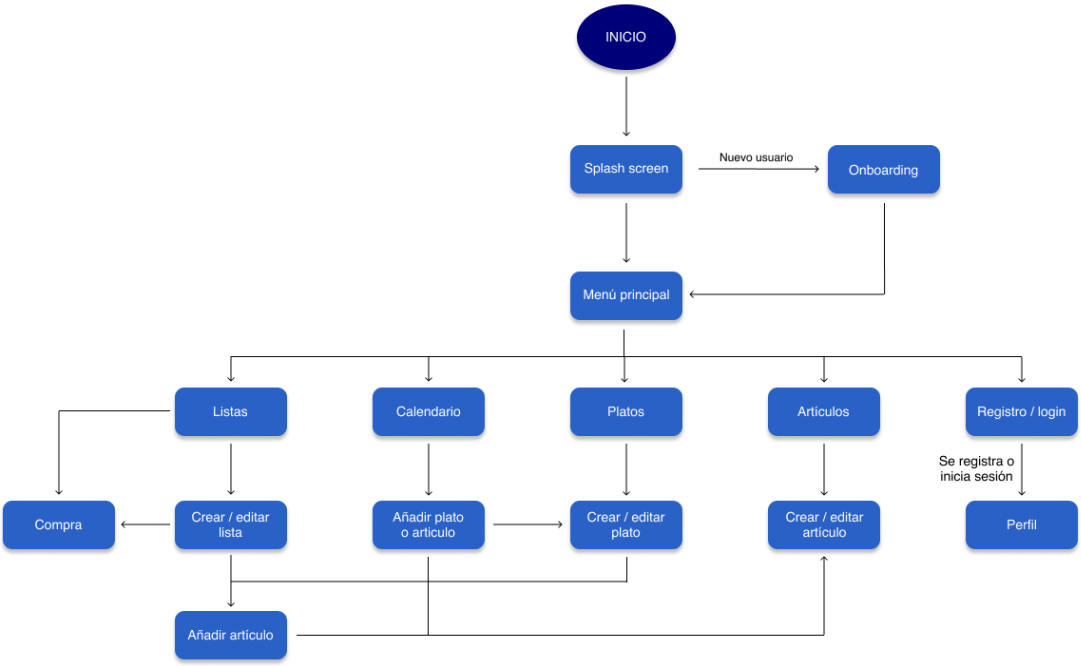

Figura 23: Árbol de navegación

Al entrar en la aplicación, el usuario verá un *splashscreen* [14] con el logo de la aplicación. En caso de ser un nuevo usuario, se le presentará un *onboarding* [15] en el que podrá ver algunas de las funciones que contiene la aplicación.

Después de esto, tendrá acceso a las principales características de la aplicación. Desde cada una de ellas, podrá llevar a cabo distintas acciones. Las de mayor importancia son las listas y el calendario, por lo que deberán posicionarse primero.

Cuando el usuario cree una lista o un calendario, debe tener la posibilidad de crear el resto de elementos necesarios en los mínimos clic posibles. Para esto, crear platos o artículos debe ser algo posible cuando estos deban asignarse al calendario o a una lista.

Como ejemplo de navegación, un usuario puede encontrarse en “Listas” y crear una nueva. Después de darle un nombre, añade los artículos desde “Añadir artículo”. En caso de necesitar alguno que no tenga creado anteriormente, podrá acceder a “Crear / editar artículo”. Después de esto volverá a la pantalla de añadir artículo y lo añadirá a la lista. Una vez creada, podrá pasar a un modo “Compra” desde el que ir tachando los artículos que ya haya comprado.

Del mismo modo ocurriría en el calendario. Si el usuario quiere añadir algún plato o producto que no tenga podrá pasar a las pantallas de crear plato o artículo y crearlos. La idea es tener acceso a las pantallas de creación de elementos desde cualquier sitio desde el que puedan añadirse.

2.3 Prototipado

2.3.1 Prototipo en papel

Teniendo en cuenta el diagrama de anterior, se define un prototipo en papel mediante dibujos, con el objetivo de reflejar la distribución de los elementos gráficos en las pantallas de la aplicación.

La primera vez que un usuario acceda a la aplicación, verá el *onboarding* [15]. Teniendo en cuenta que hay algunos usuarios que no están interesados en ver esto, se debe ofrecer la posibilidad de saltarlo.

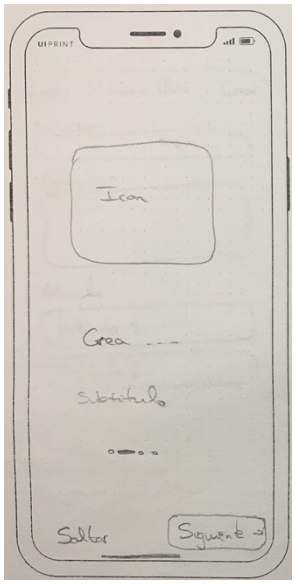

Figura 24: Prototipo en papel - onboarding

Después de esto, o si se trata de un usuario anterior, se accederá a la pantalla principal de la aplicación. Esta pantalla contendrá un *Tab Bar* [16] con las funciones principales de la aplicación: listas, calendario, platos, productos y perfil. Las *Tab Bar* permiten pasar rápidamente de una sección de la aplicación otra, de tal forma que podemos dar acceso a las funciones principales.

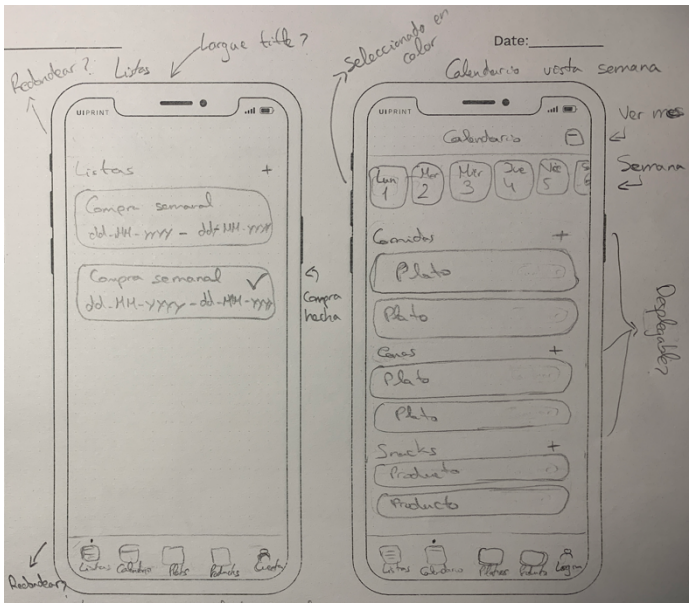

Figura 25: Prototipo en papel - Listas y calendario con vista semanal

En la pantalla de listas aparecerán tanto las que se generen automáticamente en base a los platos asociados a la semana, como las creadas manualmente por el usuario. En caso de ser de generación automática se incluirá la fecha. En cuanto al calendario, mostrará una vista con los días de la semana, pudiendo seleccionar cada uno de ellos y viendo así los platos y artículos asociados a este día. Además, en el *Navigation Bar* [17] se ofrece la posibilidad de acceder a una vista mensual.

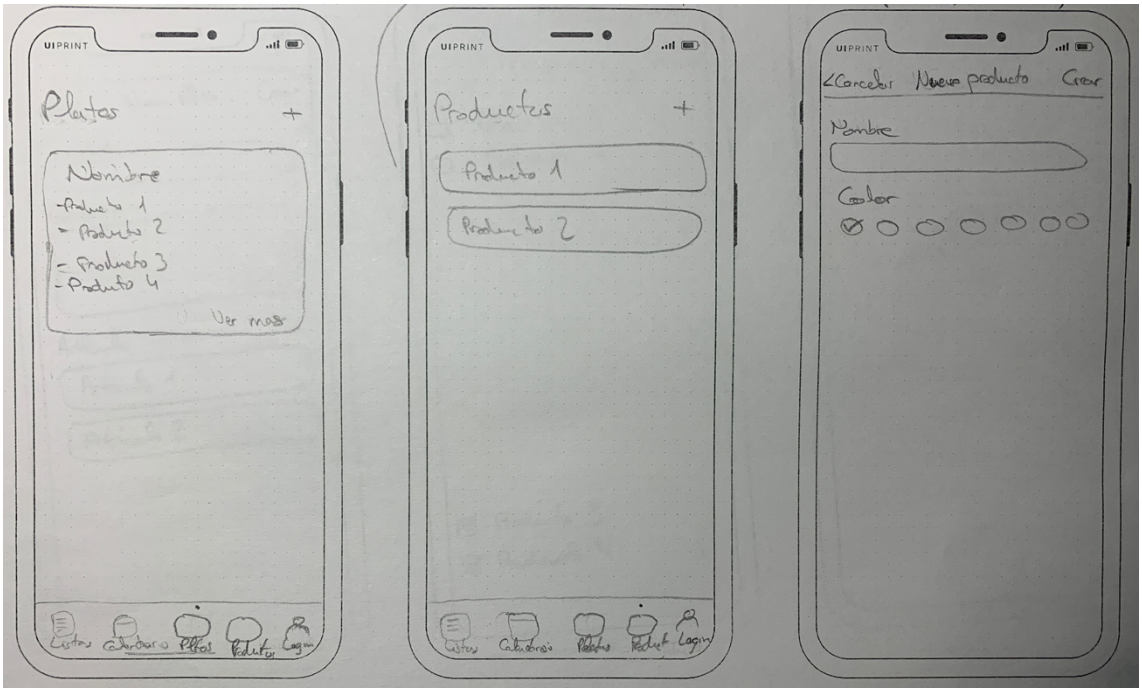

Figura 26: Prototipo en papel - Platos, productos y crear un producto

En la primera pantalla de la figura anterior puede verse la lista de platos. Esta vista incluirá el nombre y algunos de los productos que contiene, pudiendo desplegar todo el contenido del plato en un clic. De forma similar, la segunda pantalla muestra los productos. Para facilitar al usuario la organización de estos productos, es posible asignarles colores.
En la última de las pantallas puede verse la creación de un nuevo producto, asignando nombre y opcionalmente un color.

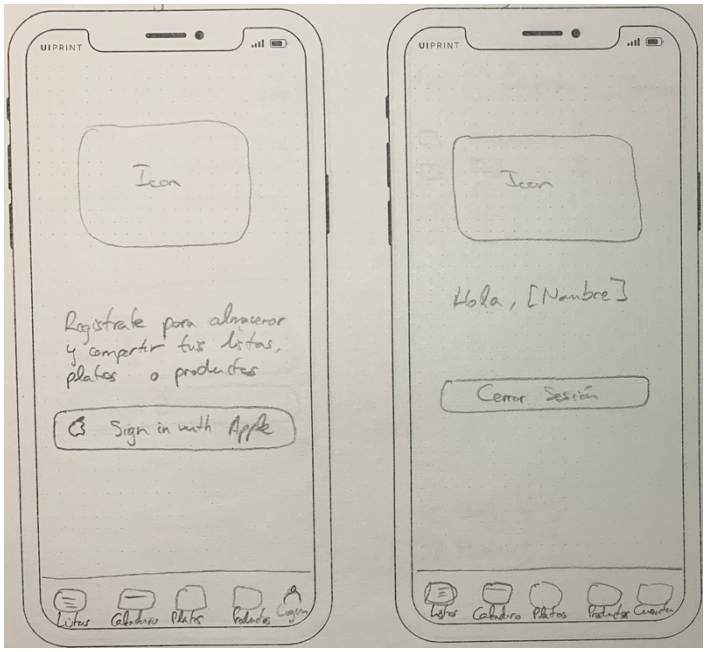

Figura 27: Prototipo en papel - login y perfil

Por último, entre las opciones principales encontramos la pantalla de perfil. En caso de que el usuario no esté registrado se mostrará la opción de hacerlo. Si lo está o ya ha iniciado sesión se mostrará un saludo y la opción de cerrarla. Crearse un perfil debe ser algo opcional, de tal forma que al crearse debe almacenarse el contenido de usuario de forma remota para que este no lo pierda si desinstala la aplicación. Además, esto permitiría compartir las listas con otros usuarios.

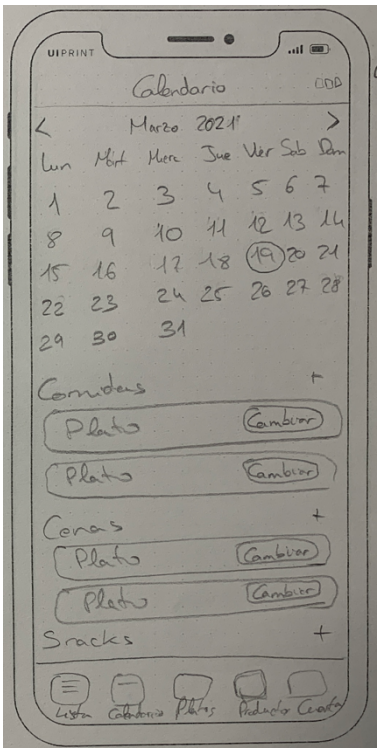

Figura 28: Prototipo en papel - Calendario con vista mensual

En la anterior figura puede verse una alternativa a la hora de ver el calendario. De la misma forma que la vista semanal, para cada día pueden verse los platos o productos asociados.

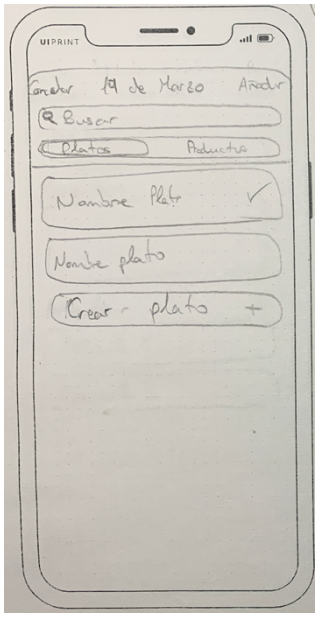

Figura 29: Prototipo en papel - Añadir desde calendario

Cuando un usuario hace clic en la opción de añadir plato o producto desde el calendario, se le mostrará una pantalla como la que aparece en la figura anterior. Deben poder filtrarse platos o productos y añadir varios a la vez. Además, si el elemento que se pretende añadir no existe, debe poder crearse desde aquí.

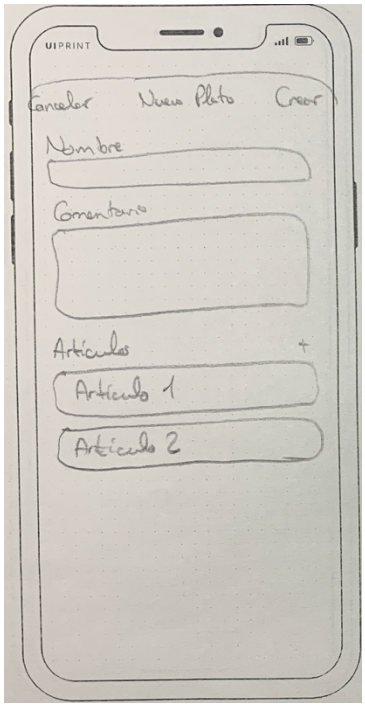

Figura 30: Prototipo en papel - Crear plato

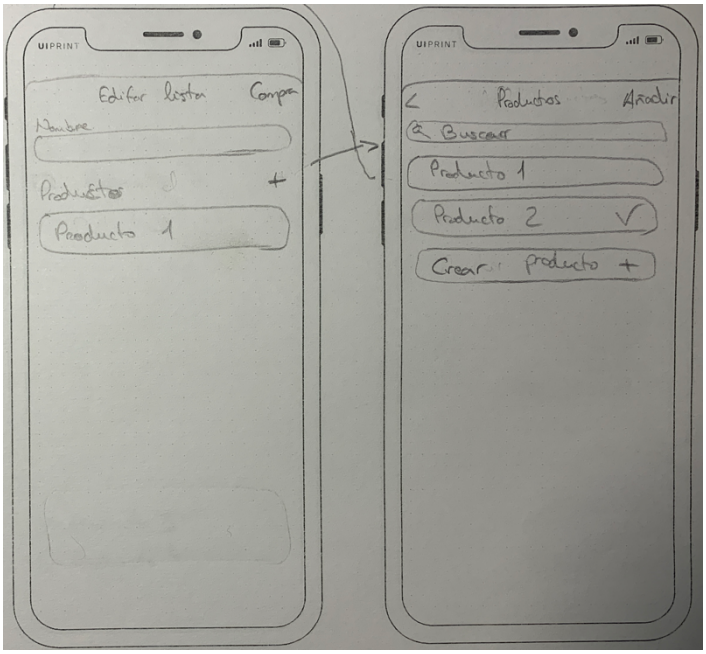

Figura 31: Prototipo en papel - Crear o editar lista y añadir producto

En la anterior figura puede verse como se crea o edita una lista. Deben poder añadirse o eliminarse elementos de la lista. Para añadirlos se accede a la segunda de las pantallas, que contiene un filtro para los productos y ofrece la posibilidad de crearlos si no existen.
Por último, la lista cuenta con la opción de entrar en modo comprar, que debe permitir marcar los artículos que ya se han comprado.

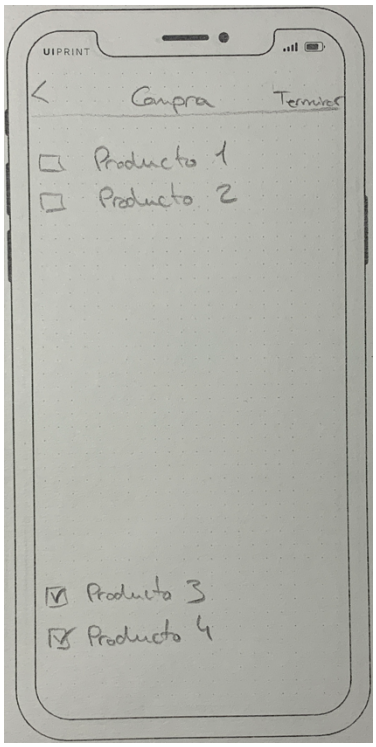

Figura 32: Prototipo en papel - modo compra

2.3.2 Prototipo en alta definición

El prototipo en alta definición intenta mostrar el estilo que se espera de la aplicación. Tras haber explicado las decisiones de diseño en la sección anterior, se muestran solamente algunas de las pantallas de la aplicación. Además, se presentan también los cambios como resultado de la evaluación con usuarios del prototipo en papel.

Figura 33: Onboarding

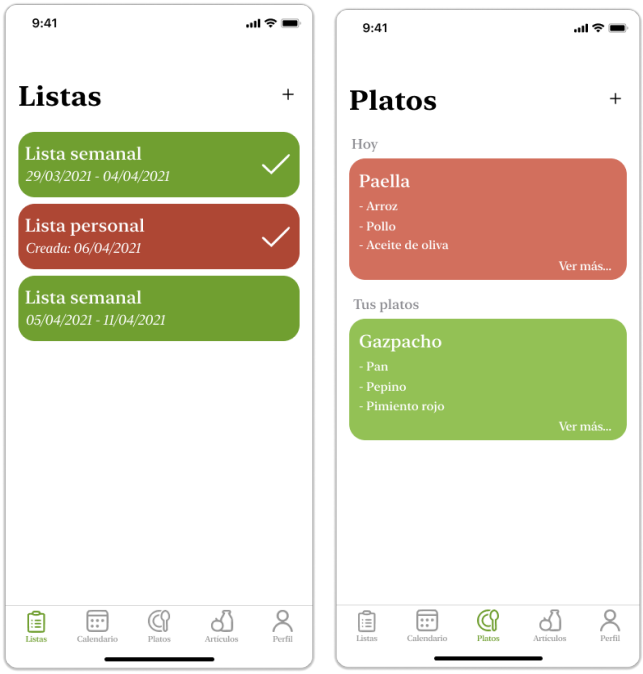

Figura 34: Listas y Platos

En la evaluación del prototipo uno de los usuarios consultados sugirió ver el plato asignado al día de hoy en la lista de platos. Se incluye esto antes de la lista por si el usuario quisiera consultar los ingredientes que necesitará hoy.

Figura 35: Vista semanal y mensual del calendario

Respecto a la pantalla del calendario, un usuario ha sugerido el cambio de la palabra “Snacks” por “Otros” ya que en esta categoría un usuario podría querer incluir cualquier alimento.

Figura 36: Crear plató

A la hora de crear un plato, uno de los usuarios que probó el prototipo sugirió que se incluyera también la opción de asignar un color, como ocurre con los ingredientes. Esto permite al usuario clasificar sus platos con más facilidad.

Por último, habiendo mostrado el prototipo en alta definición de la aplicación, uno de los usuarios sugirió poder ordenar los artículos o platos a la hora de añadirlos al calendario y a las listas. Este cambió se hará efectivo en la implementación.

3. Diseño técnico de la aplicación

3.1 Definición de casos de uso

Mediante los casos de uso se identifica la funcionalidad del sistema desde el punto de vista del usuario. A continuación, se muestran los casos de uso así como sus especificaciones.

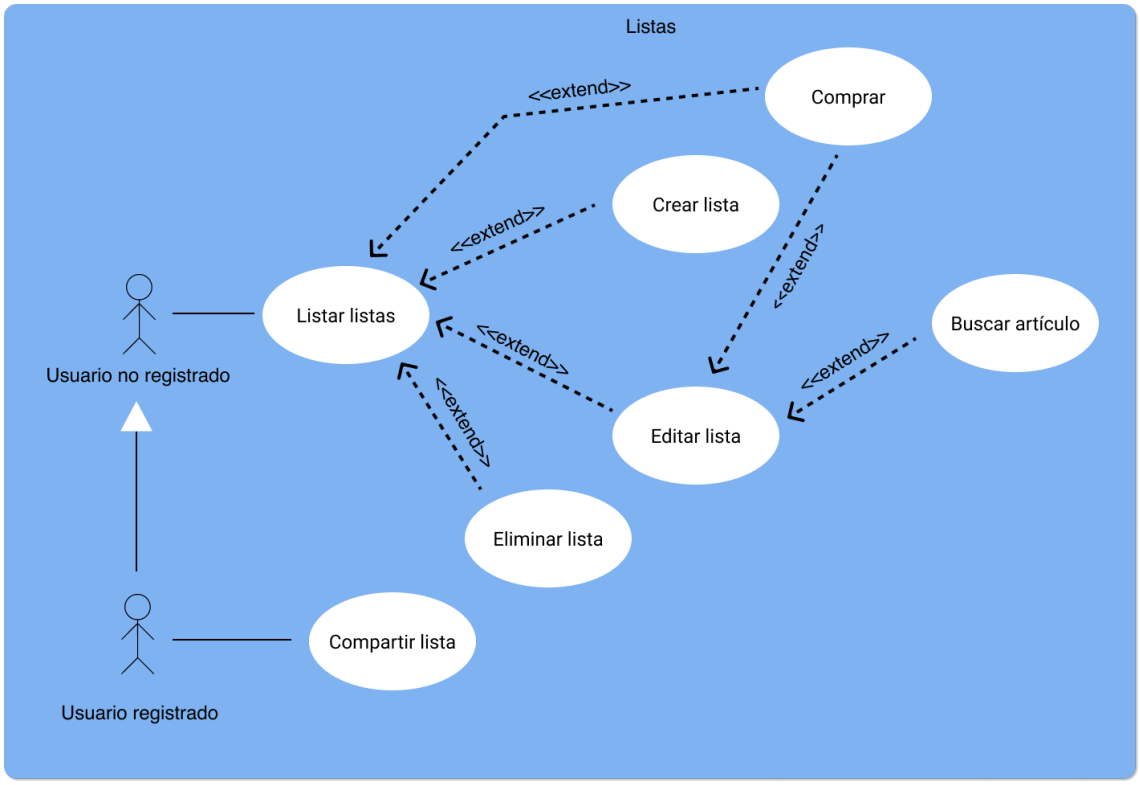

Figura 37: Casos de uso - Listas

CU_001 – Listar listas		
Nombre	Listar listas	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir mostrar las listas del usuario	
Precondición	El sistema se encuentra en un estado consistente para mostrar las listas del usuario	
Iniciado por	Usuario	
Flujo	1	El usuario inicia la aplicación o hace clic en el icono de listas del menú principal
Postcondición	El sistema muestra las listas de usuario	
Comentarios		

Tabla 5: CU_001 Listar listas

CU_002 – Crear lista		
Nombre	Crear lista	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir crear listas	
Precondición	El sistema se encuentra mostrando la pantalla listar listas	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en crear una nueva lista
Postcondición	El sistema crea una nueva lista y lleva al usuario a editar lista	
Comentarios		

Tabla 6: CU_002 Crear lista

CU_003 – Editar lista		
Nombre	Editar lista	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir editar listas	
Precondición	- El usuario se encuentra en el menú principal y ha seleccionado una lista - El usuario ha creado una nueva lista	
Iniciado por	Usuario	
Flujo	1	El usuario accede a la aplicación
	2	El usuario selecciona o crea una lista
	3	El usuario modifica la lista: nombre o artículos
Postcondición	El sistema guarda la lista editada	
Comentarios		

Tabla 7: CU_003 Editar lista

CU_004 – Eliminar lista		
Nombre	Eliminar lista	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir eliminar listas	
Precondición	El sistema se encuentra mostrando la pantalla listar listas	
Iniciado por	Usuario	
Flujo	1	El usuario hace un clic prolongado en la lista o desplaza el elemento de la lista hacia la izquierda
	2	El sistema muestra las opciones sobre las listas
	3	El usuario hace clic en eliminar lista
Postcondición	El sistema elimina la lista	
Comentarios	El sistema debe mostrar un mensaje de confirmación antes de eliminar una lista	

Tabla 8: CU_004 Eliminar lista

CU_005 - Comprar		
Nombre	Comprar	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe implementar el modo compra	
Precondición	El sistema se encuentra mostrando la pantalla listar listas	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en una lista
	2	El sistema muestra editar lista
	3	El usuario hace clic en comprar
	4	El sistema muestra el modo comprar
	5	El usuario termina su compra y marca la lista como comprada
Postcondición	El sistema marca una lista como comprada en el menú	
Comentarios		

Tabla 9: CU_005 Comprar

CU_006 – Buscar artículo		
Nombre	Buscar artículo	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir buscar artículos	
Precondición	El sistema se encuentra en el caso de uso CU_003 (Editar lista) o en CU_008 (Mostrar calendario) o en CU_020 (Editar plato)	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en añadir artículo
	2	El sistema muestra la lista de artículos del usuario
	3	El usuario busca un artículo
	4	El usuario selecciona un artículo
	5	El usuario hace clic en terminar para añadir el artículo
Postcondición	El sistema añade los artículos a la lista que corresponda	
Comentarios	El usuario debe poder seleccionar varios artículos	

Tabla 10: CU_006 Buscar artículo

CU_007 – Compartir lista		
Nombre	Crear lista	
Prioridad	Alta	
Actores	Usuarios autenticados	
Descripción	El sistema debe permitir compartir listas a los usuarios autenticados	
Precondición	El sistema se encuentra mostrando la pantalla listar listas	
Iniciado por	Usuario	
Flujo	1	El usuario hace un clic prolongado en la lista o desplaza el elemento de la lista hacia la izquierda
	2	El sistema muestra las opciones sobre las listas
	3	El usuario hace clic en compartir lista
	4	El sistema muestra las opciones de compartir
	5	El usuario elige una de las opciones de compartir
Postcondición	El sistema hace accesible la lista a quien se haya compartido	
Comentarios		

Tabla 11: CU_007 Compartir lista

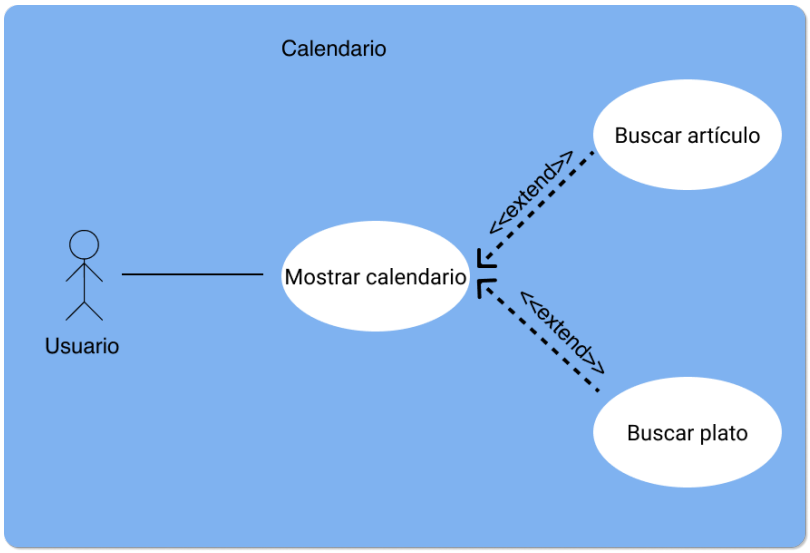

Figura 38: Casos de uso - Calendario

CU_008 – Mostrar calendario		
Nombre	Mostrar calendario	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir ver el calendario y sus platos e ingredientes asociados	
Precondición	El sistema se encuentra en un estado consistente para mostrar el calendario	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en la opción de calendario del menú principal
	2	El usuario elige entre vista semanal o vista mensual
Postcondición	El sistema muestra el calendario	
Comentarios		

Tabla 12: CU_008 Mostrar calendario

CU_009 – Buscar plato		
Nombre	Buscar plato	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe buscar platos	
Precondición	El sistema se encuentra mostrando la pantalla del calendario	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en buscar artículo
	2	El sistema muestra la lista de platos del usuario
	3	El usuario busca un plato
	4	El usuario selecciona un plato
	5	El usuario hace clic en terminar
Postcondición	El sistema añade el plato al día que corresponda	
Comentarios	El usuario debe poder seleccionar varios platos	

Tabla 13: CU_009 Buscar plato

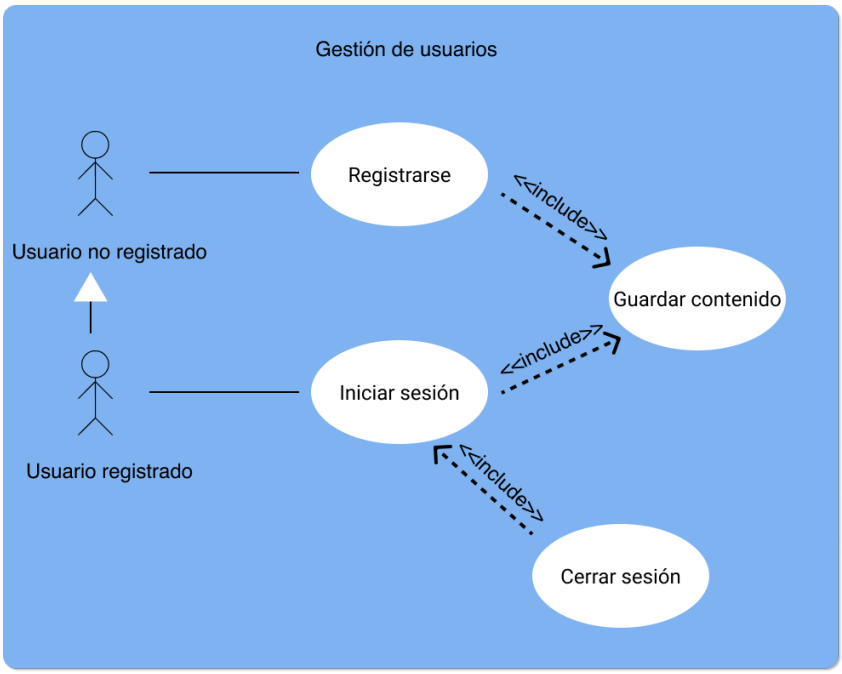

Figura 39: Casos de uso - Gestión de usuarios

CU_010 - Registrarse		
Nombre	Registrarse	
Prioridad	Baja	
Actores	Usuarios no registrados	
Descripción	El sistema debe permitir que un usuario se registre	
Precondición	El sistema se encuentra en un estado consistente en la pantalla de perfil	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en la opción “Sign in with Apple”
	2	El usuario elige una de las opciones disponibles para el registro
	3	El sistema ejecuta el caso de uso CU_012
Postcondición	El sistema crea un nuevo usuario en el sistema	
Comentarios		

Tabla 14: CU_010 Registrarse

CU_011 – Iniciar sesión		
Nombre	Iniciar sesión	
Prioridad	Baja	
Actores	Usuarios registrados	
Descripción	El sistema debe permitir iniciar sesión a un usuario registrado	
Precondición	El sistema se encuentra en un estado consistente en la pantalla de perfil	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en la opción “Continuar con Apple”
	2	El sistema ejecuta el caso de uso CU_012
Postcondición	El sistema autentica a un usuario	
Comentarios		

Tabla 15: CU_011 Iniciar sesión

CU_012 – Guardar contenido		
Nombre	Guardar contenido	
Prioridad	Baja	
Actores	Usuarios registrados	
Descripción	El sistema debe guardar el progreso a un usuario registrado	
Precondición	El sistema ha ejecutado el caso de uso CU_010 o CU_011	
Iniciado por	Sistema	
Flujo	1	El sistema guarda o actualiza el contenido en línea generado por el usuario
Postcondición	El sistema guarda en línea el contenido del usuario	
Comentarios	El contenido se debe actualizar automáticamente cuando el usuario lo genera si está autenticado	

Tabla 16: CU_012 Guardar contenido

CU_013 – Cerrar sesión		
Nombre	Cerrar sesión	
Prioridad	Baja	
Actores	Usuarios registrados	
Descripción	El sistema debe permitir cerrar sesión a un usuario autenticado	
Precondición	El sistema tiene autenticado al usuario	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en cerrar sesión
	2	El sistema cierra la sesión del usuario y vuelve a la pantalla de inicio de sesión
Postcondición	El sistema deja de guardar el contenido del usuario	
Comentarios		

Tabla 17: CU_013 Cerrar sesión

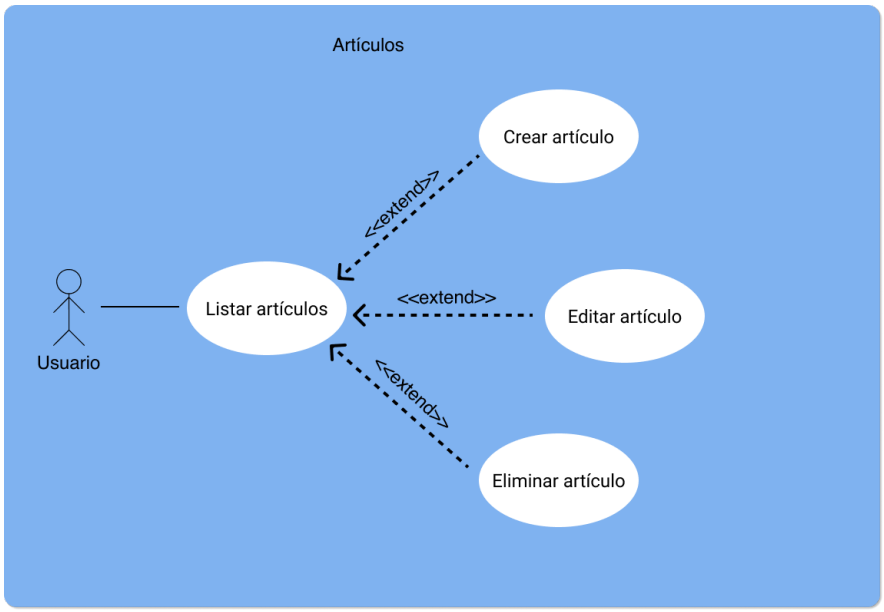

Figura 40: Casos de uso – Artículos

CU_014 – Listar artículos		
Nombre	Listar listas	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir mostrar los artículos del usuario	
Precondición	El sistema se encuentra en un estado consistente para mostrar los artículos del usuario	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en el icono de artículos del menú principal
Postcondición	El sistema muestra la lista de artículos del usuario	
Comentarios		

Tabla 18: CU_014 Listar artículos

CU_015 – Crear artículo		
Nombre	Crear artículo	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir crear artículos	
Precondición	El sistema se encuentra mostrando la pantalla listar artículo	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en crear un nuevo artículo
Postcondición	El sistema crea un nuevo artículo y lleva al usuario a editar artículo	
Comentarios		

Tabla 19: CU_015 Crear artículo

CU_016 – Editar artículo		
Nombre	Editar artículo	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir editar artículos	
Precondición	- El usuario se encuentra en el menú principal y ha seleccionado un artículo - El usuario ha creado un nuevo artículo	
Iniciado por	Usuario	
Flujo	1	El usuario selecciona o crea un nuevo artículo
	2	El usuario modifica el artículo
	3	El usuario hace clic en terminar
Postcondición	El sistema guarda el artículo editado	
Comentarios		

Tabla 20: CU_016 Editar artículo

CU_017 – Eliminar artículo		
Nombre	Eliminar artículo	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir eliminar artículos	
Precondición	El sistema se encuentra mostrando la pantalla listar artículos	
Iniciado por	Usuario	
Flujo	1	El usuario hace un clic prolongado en el artículo o desplaza el elemento de la lista hacia la izquierda
	2	El sistema muestra las opciones sobre los artículos
	3	El usuario hace clic en eliminar artículo
Postcondición	El sistema elimina el artículo	
Comentarios	El sistema debe mostrar un mensaje de confirmación antes de eliminar un artículo	

Tabla 21: CU_017 Eliminar artículo

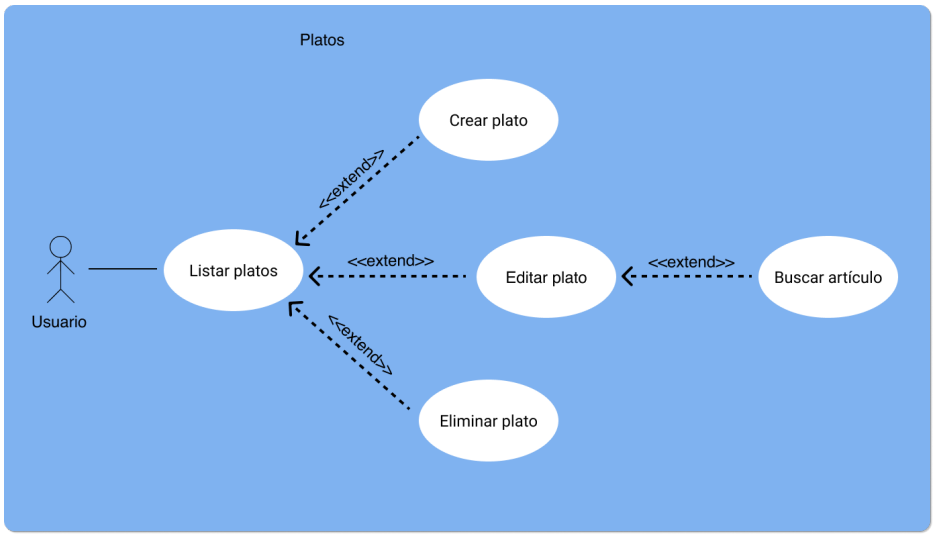

Figura 41: Casos de uso - Platos

CU_018 – Listar platos		
Nombre	Listar platos	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir mostrar los platos del usuario	
Precondición	El sistema se encuentra en un estado consistente para mostrar los platos del usuario	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en el icono de platos del menú principal
Postcondición	El sistema muestra los platos del usuario	
Comentarios		

Tabla 22: CU_018 Listar platos

CU_019 – Crear plato		
Nombre	Crear plato	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir crear platos	
Precondición	El sistema se encuentra mostrando la pantalla listar platos	
Iniciado por	Usuario	
Flujo	1	El usuario hace clic en crear un nuevo plato
Postcondición	El sistema crea un nuevo plato y lleva al usuario a editar plato	
Comentarios		

Tabla 23: CU_019 Crear plato

CU_020 – Editar plato		
Nombre	Editar plato	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir editar platos	
Precondición	- El usuario se encuentra en el menú principal y ha seleccionado un plato - El usuario ha creado un nuevo plato	
Iniciado por	Usuario	
Flujo	1	El usuario selecciona o crea un nuevo plato
	2	El usuario modifica el plato
	3	El usuario hace clic en terminar
Postcondición	El sistema guarda el plato editado	
Comentarios		

Tabla 24: CU_020 Editar plato

CU_021 – Eliminar plato		
Nombre	Eliminar plato	
Prioridad	Alta	
Actores	Todos los usuarios	
Descripción	El sistema debe permitir eliminar platos	
Precondición	El sistema se encuentra mostrando la pantalla listar platos	
Iniciado por	Usuario	
Flujo	1	El usuario hace un clic prolongado en el plato o desplaza el elemento de la lista hacia la izquierda
	2	El sistema muestra las opciones sobre los platos
	3	El usuario hace clic en eliminar plato
Postcondición	El sistema elimina el plato	
Comentarios	El sistema debe mostrar un mensaje de confirmación antes de eliminar un plato	

Tabla 25: CU_021 Eliminar plato

3.2 Arquitectura

Para comenzar a definir la arquitectura de la aplicación, se definen primero las clases que almacenarán los datos. Estas clases compondrán las tablas de la base de datos local, así como la remota, que será una copia sincronizada de la local. Para almacenar los datos de forma remota se utilizará la base de datos en tiempo real de Firebase, por lo que las clases serán serializadas para convertirse fácilmente a formato JSON.

A continuación, se muestra el diagrama de clases de la aplicación.

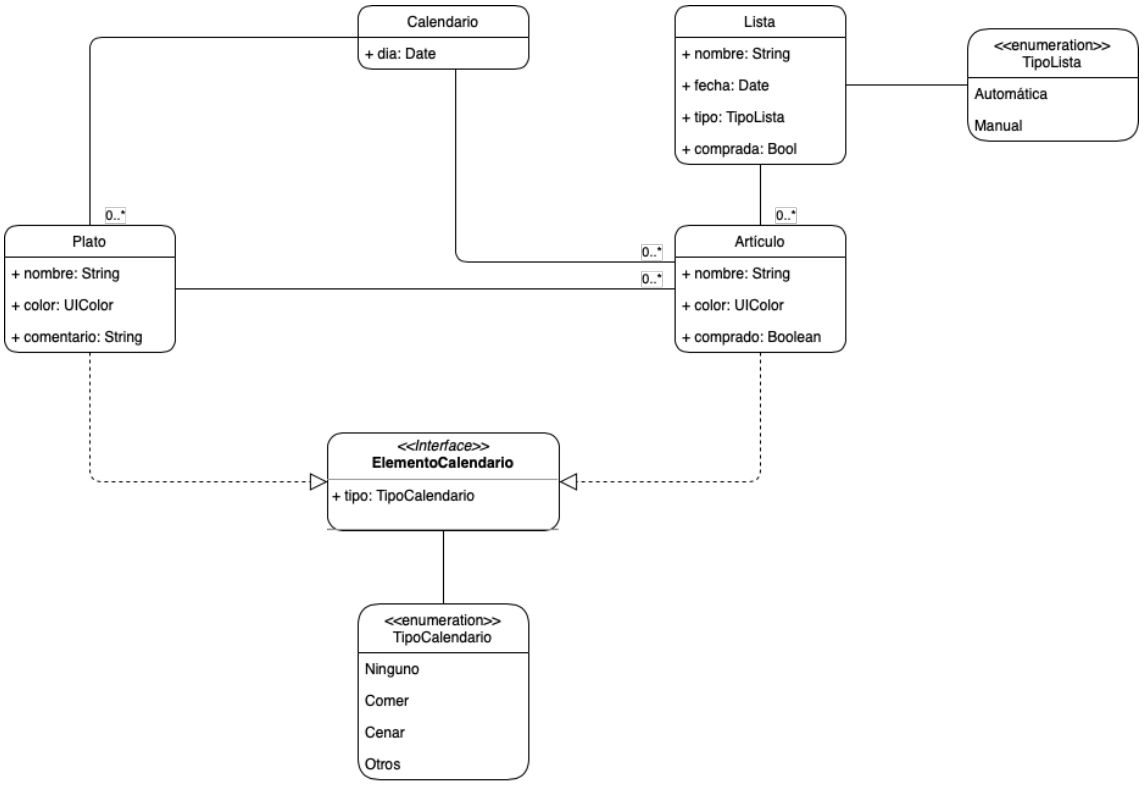

Figura 42: Diagrama de clases

La aplicación contiene cuatro clases principales: artículo, plato, lista y calendario.

La tabla *Calendario* solo almacenará los días que tengan algún elemento asociado, por lo tanto, contendrá el día y una lista de elementos que implemente la interfaz *ElementoCalendario*, en este caso, contendrá una lista de platos o artículos.

Por otro lado, los platos contendrán una lista de artículos, nombre, color y comentario. Los artículos además de las propiedades que define el usuario, contendrá una propiedad de tipo *Boolean* para el modo compra. Por último, las listas contendrán una lista de artículos, una fecha y un atributo para diferenciar si se ha generado automáticamente o de forma manual.

Como ya se indicó anteriormente en la sección de enfoque y método seguido, la arquitectura utilizada será MVVM. Se hará uso de clases *repository* que abstraerán los *ViewModel* de si la fuente de datos es local o remota. El esquema seguido será el que aparece en la siguiente imagen.

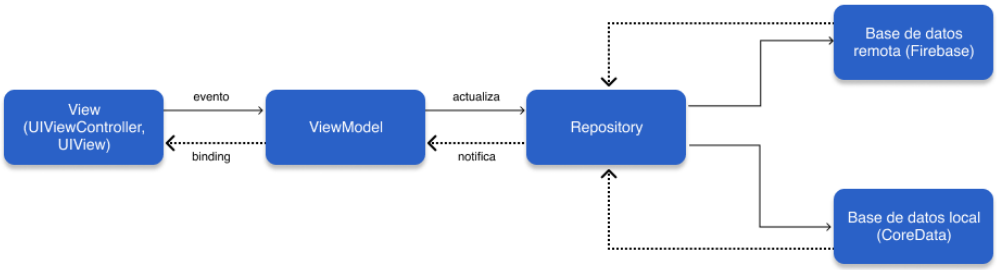

Figura 43: Arquitectura MVVM

4. Implementación

En esta sección se exponen los aspectos más relevantes de la implementación de la aplicación.

Como punto de partida, se ha creado un proyecto desde cero en Xcode[18]. Xcode es el IDE de desarrollo de Apple para aplicaciones tanto de iOS como para el resto de sistemas de esta empresa. Es la mejor opción disponible a la hora de desarrollar aplicaciones para iOS.

4.1 Estructura del proyecto

Generalmente, un proyecto en iOS está compuesto por archivos .swift, que contienen el código en Swift, y archivos .xib y .storyboard, donde se definen de forma gráfica las vistas que componen las pantallas de la aplicación. En estos archivos pueden definirse las restricciones de las vistas (Autolayout[19]), así como conexiones con sus correspondientes archivos en Swift. No obstante, en este proyecto no se han utilizado estos archivos, definiendo todas las vistas mediante código en Swift, a excepción de la pantalla de SplashScreen. Esta decisión se basa en una preferencia personal, por tener mayor control y flexibilidad a la hora de definir y colocar las vistas.

La estructura del proyecto está determinada por la arquitectura elegida anteriormente, MVVM. En la siguiente imagen se muestra la estructura de directorios y, a continuación, qué contienen cada uno de estos directorios.

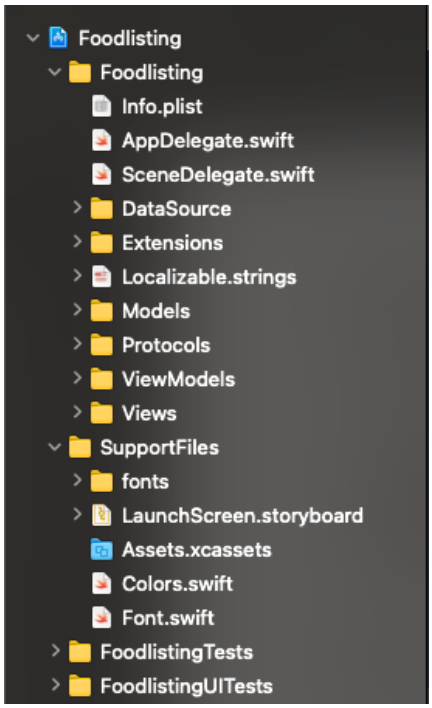

Figura 44: Estructura de directorios del proyecto

- El archivo Info.plist contiene información sobre la configuración de la aplicación. Por otro lado, los archivos AppDelegate y SceneDelegate, son el punto de partida de la aplicación. Contienen, por ejemplo, la definición del contenedor de CoreData, que se encargará del almacenamiento del contenido del usuario.

- El directorio DataSource es la primera capa arquitectónica. Aquí se encuentran las clases encargadas de acceder a los datos, tanto locales como remotos.

- Extensions contiene extensiones de clases ya definidas en iOS. Esto permite reutilizar estas funciones y no repetir código.

- Los archivos Localizable.strings contienen las traducciones de las distintas cadenas de la aplicación.

- El directorio Models contiene los modelos de nuestro proyecto, en este caso, se trata de las clases generadas por CoreData además de otras clases que contendrán datos a la hora de ser representados.

- Protocols contiene los protocolos definidos para facilitar la reutilización de código. En Swift los protocolos son muy similares a las interfaces de otros lenguajes como Kotlin.

- ViewModels contiene las clases que representan otra capa de la arquitectura. En este caso aquí se encuentran las clases con la lógica de negocio.

- Views contiene las vistas y las clases ViewController que definen las pantallas que verá el usuario.

- El directorio SupportFiles contiene algunos archivos que no tienen cabida en ninguno de los anteriores. Contiene la definición de colores de la aplicación, los Assets, que contienen los iconos, y la fuente New York[20]. Esta es la fuente que se utilizó en el diseño de la interfaz gráfica.

- Por último, se encuentran los directorios que contienen los test unitarios y test de UI.

En cuanto a la fuente, se han asignado a las vistas a lo largo de la aplicación utilizando tamaños que escalan de forma dinámica [21]. Esto se implementa con intención de mejorar la accesibilidad. Cuando un usuario cambie el tamaño de letra en su dispositivo, la aplicación adaptará el tamaño del texto automáticamente. Al utilizar una fuente externa, se ha creado una clase para poder proporcionar este comportamiento.

4.2 Adaptación a otros idiomas

Ya desde hace unos años las aplicaciones se distribuyen de forma internacional. Además, teniendo en cuenta la baja cuota de mercado de iOS tanto en España como en Hispanoamérica, proporcionar más idiomas que el español se convierte en una obligación. Por eso, la aplicación ha sido traducida tanto a inglés como a alemán, haciendo uso de las facilidades para la internacionalización que proporciona Apple.

4.3 Librerías y herramientas

4.3.1 Compositional Layout

En la conferencia mundial de desarrolladores que celebra cada año Apple, en 2019, se presentó el *compositional layout*. Mediante la clase UICollectionViewCompositionalLayout pueden definirse complejas representaciones de colecciones. Todas las listas y colecciones de vistas de la aplicación se han implementado mediante esta nueva forma de hacerlo.

A la hora de trabajar con esto, es necesario definir de manera jerárquica una serie de elementos que pueden verse en la siguiente imagen.

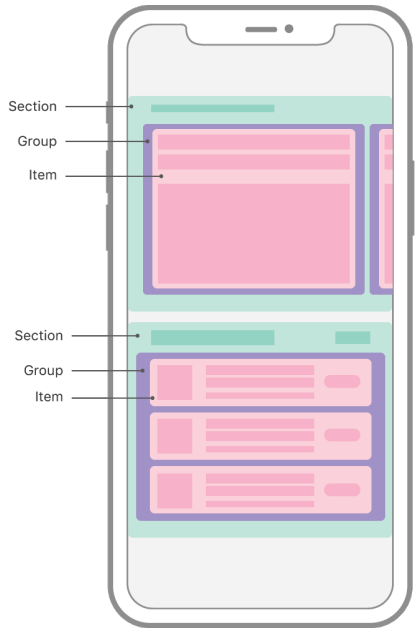

Figura 45: Elementos que contiene UICollectionViewCompositionalLayout

Deben definirse elementos, grupos de elementos y secciones. Como ejemplo de cómo se han creado las listas utilizadas tanto en artículos, listas de la compra y platos, puede verse el código de la siguiente imagen.

```
static func listLayout(groupEstimatedHeight: CGFloat) ->
UICollectionViewCompositionalLayout {
    let itemSize = NSCollectionLayoutSize(
      widthDimension: .fractionalWidth(1.0),
      heightDimension: .fractionalHeight(1.0))

    let layoutItem = NSCollectionLayoutItem(layoutSize: itemSize)
    layoutItem.contentInsets = NSDirectionalEdgeInsets(top: 4, leading: 8,
bottom: 4, trailing: 8)

    let layoutGroupSize = NSCollectionLayoutSize(
      widthDimension: .fractionalWidth(1.0),
      heightDimension: .estimated(groupEstimatedHeight))

    let layoutGroup = NSCollectionLayoutGroup.horizontal(layoutSize:
layoutGroupSize, subitems: [layoutItem])

    let section = NSCollectionLayoutSection(group: layoutGroup)
    return UICollectionViewCompositionalLayout(section: section)
  }
```

Figura 46: Ejemplo de UICollectionViewCompositionalLayout

4.3.2 CoreData

CoreData [23] es un framework de Apple que facilita la persistencia de datos de una aplicación. Su objetivo es simplificar la gestión de estos datos mediante las herramientas que proporciona. CoreData guarda la información en una base de datos SQLite.

En la siguiente imagen, puede verse el modelo de CoreData de la aplicación. Aunque algunas entidades aparecen sin propiedades en este editor, es simplemente un error en la representación.

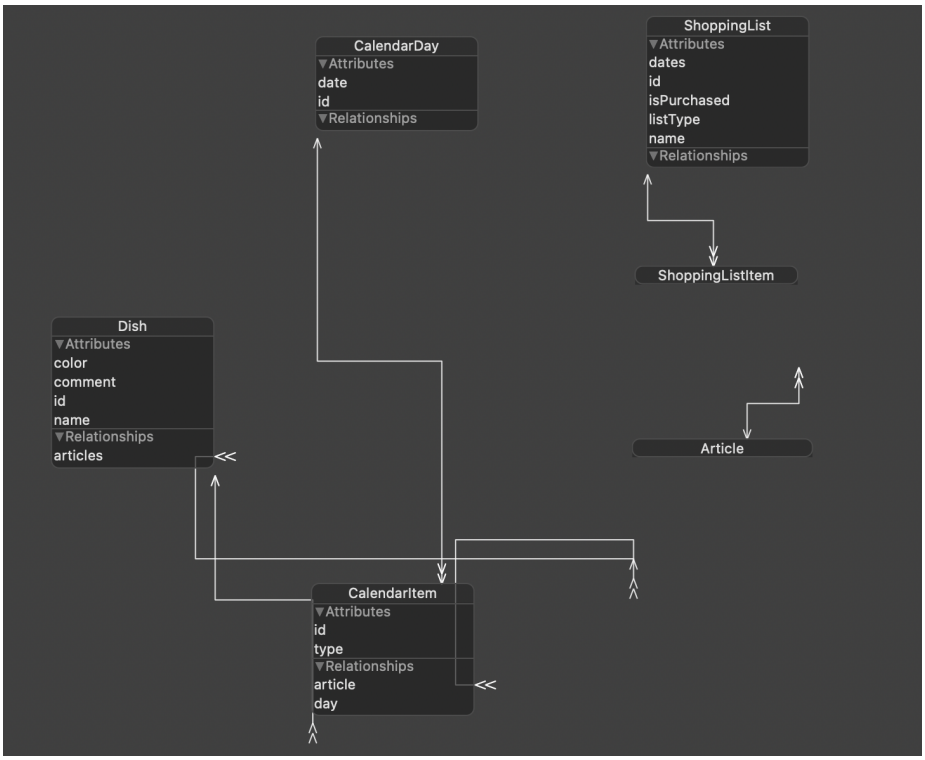

Figura 47: Modelo de CoreData

Para adaptar la forma de trabajar de CoreData al diagrama de clases definido anteriormente, ha sido necesario crear las entidades CalendarItem y ShoppingListItem, que contienen referencias a los días, artículos y listas según corresponda.

Además de esto, CoreData permite guardar la información fácilmente en iCloud, por lo que el contenido que un usuario guarde en la aplicación estará disponible también para el resto de dispositivos del usuario.

4.3.3 Git y GitHub

Git [24] es un sistema de control de versiones utilizado de manera masiva. Aunque este es un proyecto individual, los beneficios que aporta una herramienta como esta son importantes. Por esto, desde el principio del proyecto se ha utilizado. Además, para almacenar el repositorio de forma remota, se ha utilizado GitHub [25]. El código de esta aplicación se encuentra en un repositorio privado, pero será publicado tras la presentación de este proyecto.

4.3.4 Swiftlint

Swiftlint [26] es una herramienta que permite comprobar el estilo y las convenciones del código escrito para facilitar que distintos desarrolladores puedan entenderse con mayor facilidad. Aunque este proyecto es desarrollado por una sola persona, es una buena práctica utilizar estas herramientas para poder cumplir de una manera más estricta con un estilo de código determinado.

Existen muchos estilos diferentes. En este proyecto se ha utilizado el de raywenderlich [27]. El repositorio contiene un archivo llamado swiftlint.yml en el que se definen las reglas que la herramienta debe comprobar.

4.4 Otros aspectos relevantes del desarrollo

Durante la fase de desarrollo de la aplicación, han surgido algunos imprevistos que han llevado a tomar decisiones significativas. El tiempo dedicado al desarrollo se ha reducido, sobre todo en las primeras semanas, por un cambio de empleo. Las numerosas entrevistas que se han realizado han restado un tiempo que hace difícil cumplir con todas las previsiones de la planificación.

El cambio más significativo ha sido prescindir de Firebase. Esto podría haber llevado a prescindir también de todas las funcionalidades que iban a proporcionarse con Firebase, pero no ha sido así. La decisión de utilizar CoreData se basa en una preferencia personal de utilizar este proyecto para aprender sobre este framework. Durante el estudio del mismo, se ha descubierto que era simple integrar esto con iCloud, para poder ofrecer así la funcionalidad de la base de datos remota sincronizada.

Utilizar iCloud y CoreData evita una gran cantidad de tiempo de implementación, ya que no es necesario serializar a JSON los modelos de la aplicación ni crear una estructura diferente a la que se tiene en CoreData para adaptarla a la base de datos de Firebase. Aún más importante es evitar tener que implementar una sincronización entre la base de datos local y la remota. Esto evita no solo tiempo, sino también una gran cantidad de potenciales errores.

La funcionalidad que no ha sido posible implementar es la de compartir listas, para lo que habría que estudiar el framework con más tiempo y en más profundidad, o buscar alternativas.

Estas decisiones, han provocado también un cambio en el diseño, ya que la última de las opciones del TabBar carece de sentido no siendo necesario el registro en la aplicación.

4.5 Tests

Los test unitarios permiten comprobar el funcionamiento del código y asegurar que sigue funcionando tras los cambios. Siendo su implementación imprescindible en el desarrollo, se han incluido también en este proyecto.

Para implementar estos tests, se ha utilizado el framework XCTest [28] proporcionado por Apple en Xcode.

Se han escrito test para las extensiones de algunas clases, como la fecha, para asegurar el funcionamiento de CoreData y para los *ViewModel*.

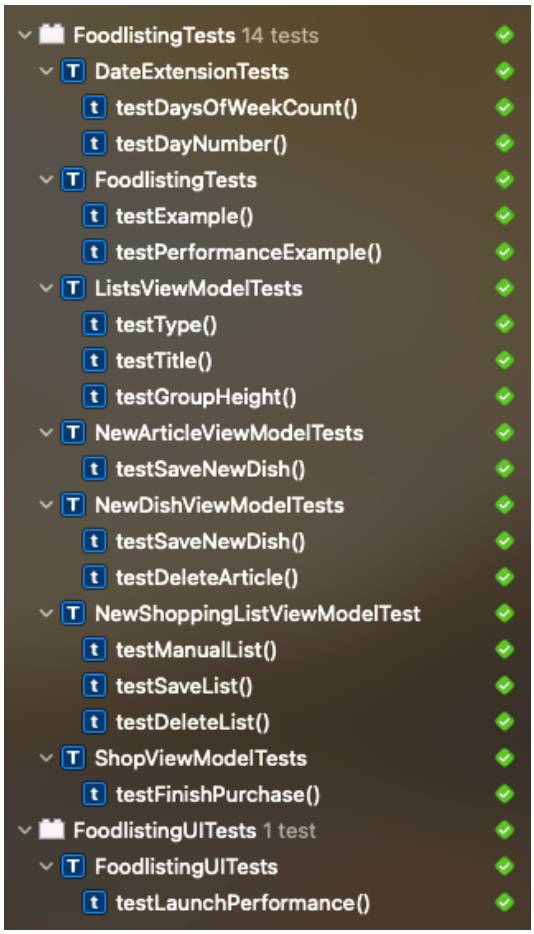

Figura 48: Tests

Por último, se ha probado la aplicación en emuladores de distintos tamaños, como en iPads, para comprobar que las vistas se adaptan correctamente en las distintas pantallas.

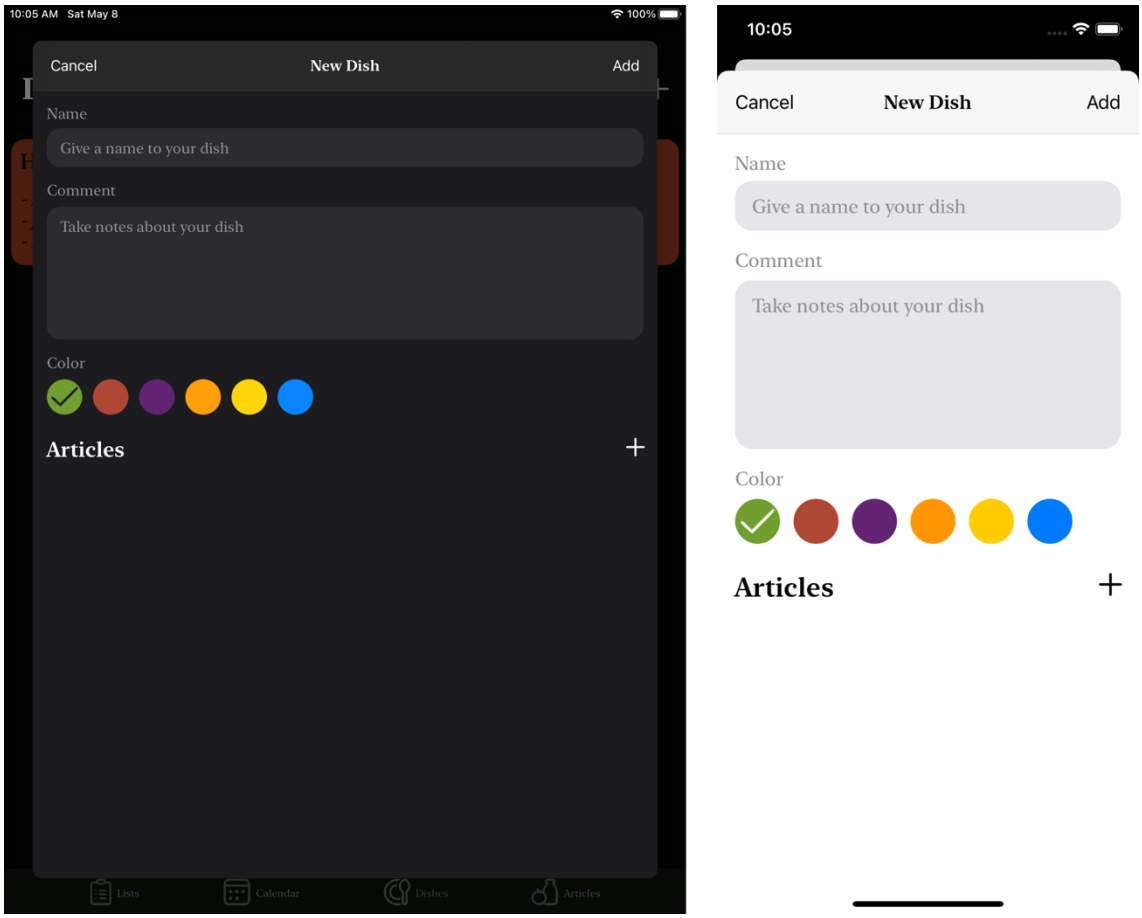

Figura 49: Pantalla de creación de un plato en un iPad pro y en un iPhone 12 mini

4.6 Aspecto final de la aplicación

A continuación, se adjuntan algunas capturas en las que se puede ver el aspecto final de la aplicación.

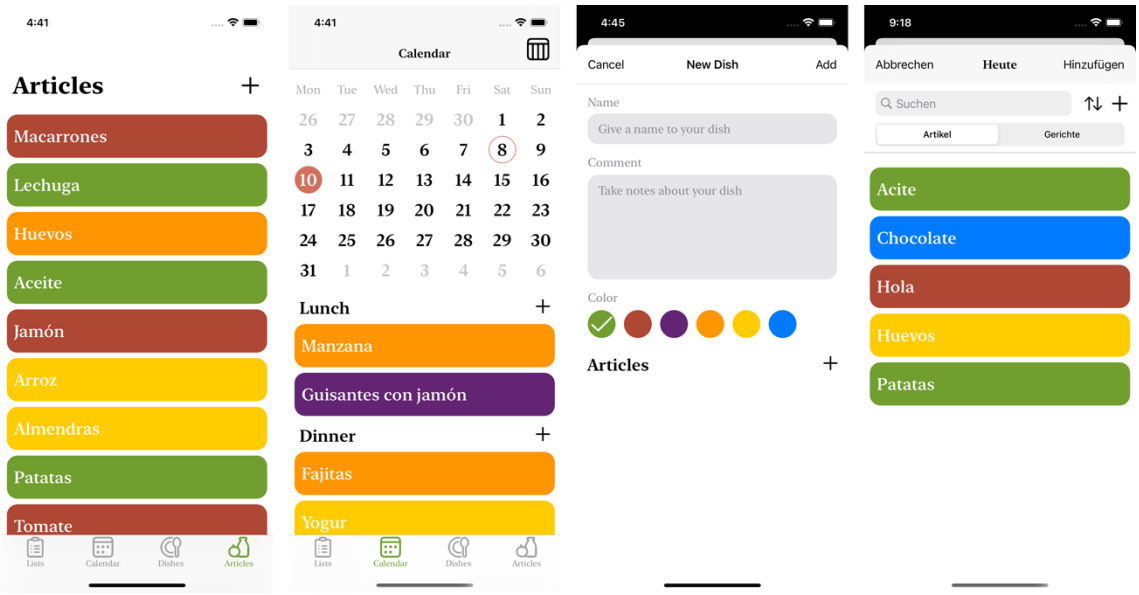

Figura 50: Aspecto final de la aplicación: light mode

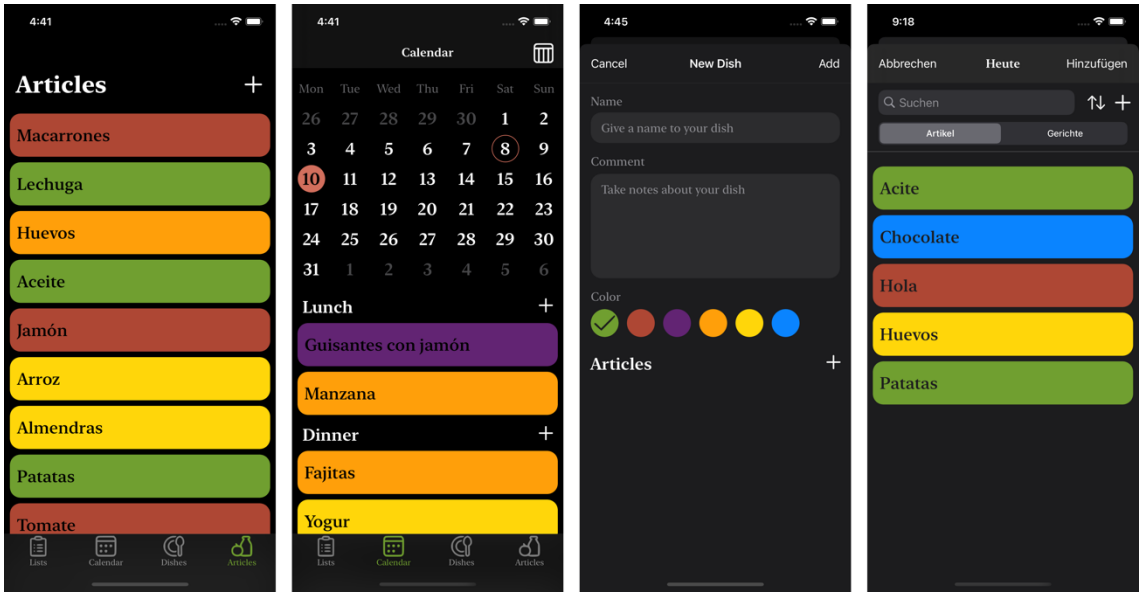

Figura 51: Aspecto final de la aplicación: dark mode

5. Conclusiones

Al concluir este proyecto, puede decirse que se ha completado satisfactoriamente. Se han podido aplicar conocimientos de varias asignaturas del máster, así como ampliarlos con herramientas necesarias para el desarrollo de este trabajo.

Se ha cumplido el objetivo del proyecto de crear una aplicación que permite al usuario gestionar sus ingredientes, platos y organizarse semanalmente. Así mismo, se ha implementado con éxito la arquitectura definida en la planificación, lo que permite añadir nuevas funcionalidades sin gran dificultad.

Como en todo proyecto, han surgido dificultades a lo largo de su desarrollo. Desconocer algunas herramientas ha provocado que se tuvieran que hacer modificaciones sobre la marcha, además de las circunstancias externas al proyecto, que también han afectado y causado modificaciones. La falta de tiempo durante la fase de implementación no ha permitido cumplir con todos los objetivos funcionales, ya que la función de compartir listas no se ha implementado.

Este trabajo ha servido para darse cuenta de algunos aspectos importantes. La importancia de una buena planificación es clave, ya que ahorra una gran cantidad de tiempo en las fases posteriores. En el desarrollo de este proyecto, se ha cumplido con gran parte de la planificación, pero también hay algunas fases que podrían haberse mejorado de manera significativa. Como ejemplo, haber conocido CoreData para poder hacer un diagrama de clases acorde a cómo funciona esta herramienta, habría ahorrado tiempo en la implementación, al no tener que plantearse cómo pasar de un modelo al otro.

Uno de los objetivos personales mas complejos que se plantearon desde el principio fue el de poder gestionarse para completar el proyecto. Tener que compaginar esto con un trabajo a jornada completa, estudios de idiomas y la vida personal es todo un reto de gestión de tiempo y autodisciplina. Además, también ha servido para darse cuenta de la gran cantidad de tiempo que requiere realizar un proyecto completo en solitario, lo que sirve para plantearse los proyectos personales en el futuro de forma más realista.

6. Líneas de trabajo futuro

En esta sección se exponen algunas mejoras que podrían implementarse en la aplicación para hacerla más útil o eficiente.

- Incluir un campo en los artículos del plato que permita indicar la cantidad necesaria de ese artículo para elaborar el plato. Después de esto, tenerlo en cuenta al generar las listas para indicar también cantidades.

- Poder cambiar el día de la semana en el que esta empieza. En otros países la semana no empieza en lunes como hace por defecto la aplicación.

- Poder ocultar listas que ya se han comprado.

- Mejorar el uso de la herencia para la reutilización de código. Aunque se ha utilizado en algunos puntos, existe código que resulta repetitivo y podría evitarse.

- Revisión de las transacciones de base de datos para evitar escribir cuando no sea necesario. Es posible que se estén haciendo tanto lecturas como escrituras en exceso, utilizando así la aplicación más recursos de los necesarios.

- Realizar las operaciones de lectura y escritura en segundo plano. Cuando hay pocos elementos almacenados puede no notarse demasiado, pero aún así, no es buena práctica hacer estas operaciones en el hilo principal.

- Función de compartir listas, que no se ha implementado por falta de tiempo.

7. Glosario

Open Source: término que hace referencia a los programas informáticos que permiten que se consulte su código fuente públicamente.

Modo oscuro: función que permite oscurecer los colores del sistema y de las aplicaciones. Puede reducir el consumo del dispositivo.

MVC: siglas de Model View Controller. Es un patrón de diseño de arquitectura de software que separa la lógica de negocio de la fuente de datos y su representación haciendo uso de tres elementos: modelo, vista y controlador.

MVVM: siglas de Model View ViewModel. Es un patrón de diseño de arquitectura de software. Trata de desacoplar también la lógica de negocio de los datos y la vista haciendo uso de tres elementos: modelo, vista y vista del modelo.

VIPER: patrón de diseño de arquitectura que desacopla los elementos de una aplicación y se divide en cinco elementos: view, interaction, presenter, entity, router.

Interface Builder: es una aplicación, parte de Xcode, que permite crear interfaces para aplicaciones de manera gráfica.

8. Bibliografía

[1]. *Bring! Lista de la compra [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://apps.apple.com/es/app/bring-lista-de-compras/id580669177#?platform=iphone

[2]. *Mi lista de la compra fácil [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://apps.apple.com/es/app/mi-lista-de-la-compra-fácil/id1462399042

[3]. *Noodle: Recetas sanas fácil [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://apps.apple.com/es/app/nooddle-recetas-sanas-fáciles/id1329457709

[4]. *Kitchen Stories Recipes [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://apps.apple.com/us/app/kitchen-stories-recipes/id771068291

[5]. *Dark mode [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/dark-mode

[6]. *Localizar aplicación en varios idiomas [En línea]* [Fecha de consulta: 28/02/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/dark-mode

[7]. *Adopción de la última versión de iOS [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://developer.apple.com/support/app-store/

[8]. *Swift.org [En línea]* [Fecha de consulta: 07/03/2021] Disponible en: https://swift.org

[9]. *Presentación de SwiftUI en WWDC 2019 [En línea].* [Fecha de consulta: 28/02/2021]. Disponible en: https://developer.apple.com/videos/play/wwdc2019/204/

[10]. *Human Interface Guidelines [En línea].* [Fecha de consulta: 07/03/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/overview/theme

[11]. *Firebase [En línea].* [Fecha de consulta: 07/03/2021]. Disponible en: https://firebase.google.com

[12]. *Sign in with Apple [En línea].* [Fecha de consulta: 07/03/2021]. Disponible en: https://developer.apple.com/sign-in-with-apple/

[13]. Iniciar sesión con Apple, obligatorio desde el 30 de junio de 2020. [En línea]. [Fecha de consulta: 07/03/2021]. Disponible en: https://www.applesfera.com/servicios-apple/iniciar-sesion-apple-sera-obligatorio-para-apps-a-partir-30-junio-recordamos-sus-ventajas

[14]. Splashscreen o Launch Screen. [En línea]. [Fecha de consulta 21/03/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/launch-screen/

[15]. *Onboarding. [En línea].* [Fecha de consulta 21/03/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/onboarding/

[16]. *Tab Bar [En línea].* [Fecha de consulta 21/03/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/bars/tab-bars/

[17]. *Navigation Bar [En línea].* [Fecha de consulta 21/03/2021]. Disponible en: https://developer.apple.com/design/human-interface-guidelines/ios/bars/navigation-bars/

[18]. *Xcode [En línea].* [En línea]. [Fecha de consulta 07/05/2021]. Disponible en: https://developer.apple.com/xcode/

[19]. *AutoLayout [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html

[20]. *Fuente New York de Apple [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://developer.apple.com/fonts/

[21]. *Scaling fonts automatically [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://developer.apple.com/documentation/uikit/uifont/scaling_fonts_automatically

[22]. *Compositional layout series [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://nemecek.be/blog/series/compositional-layout

[23]. *CoreData [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://developer.apple.com/documentation/coredata

[24]. *Git [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://git-scm.com

[25]. *Github [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://github.com

[26]. *Swiftlint [En línea].* [Fecha de consulta 07/05/2021]. Disponible en: https://github.com/realm/SwiftLint

[27]. *Raywenderlich Swift Style Guide [En línea].* Disponible en: https://github.com/raywenderlich/swift-style-guide

[28]. *XCTest [En línea].* Disponible en: https://developer.apple.com/documentation/xctest

9. Anexos

9.1 Manual de usuario

En esta sección se explica el funcionamiento de la aplicación con la ayuda de capturas de pantalla.

Cuando un usuario abre la aplicación por primera vez, se encuentra sin ningún elemento.

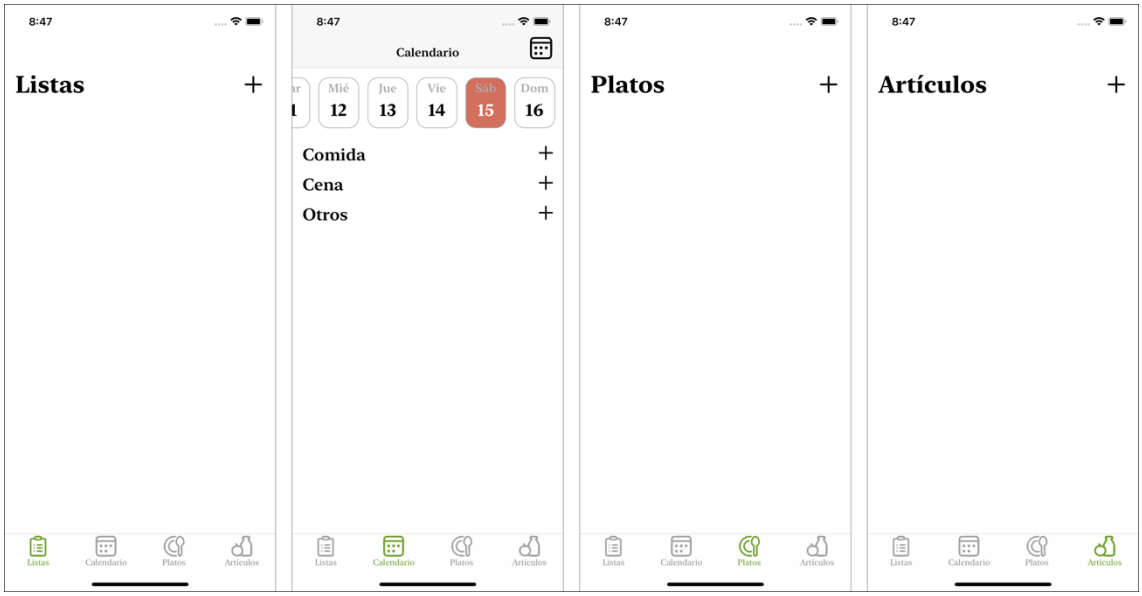

Figura 52: Vista de la aplicación en la instalación

Después de esto, puede añadir artículos, haciendo clic en el icono + de la parte superior derecha.

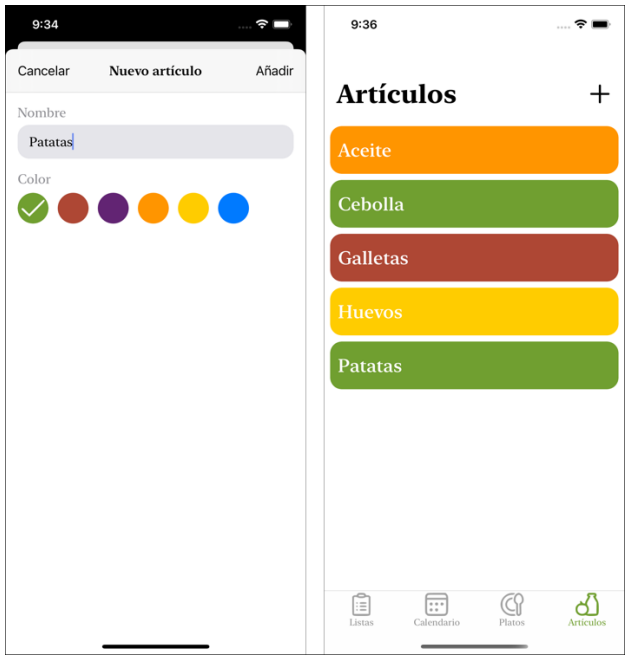

Figura 53: Artículos

Cada artículo debe llevar un nombre y se le puede asignar un color. Cuando se hace clic en la opción de añadir, este artículo se crea. En la imagen que se puede ver a la derecha, se listan varios artículos creados.

En cuanto a los platos, pueden crearse también haciendo clic en el icono + de la parte superior derecha. Al crear un plato, se le debe asignar un nombre y un color. Después de esto, se puede incluir un comentario de manera opcional y añadir los artículos que contenga este plato. La pantalla en la que se realiza la creación del plato puede verse a la izquierda de las siguientes capturas de pantalla.

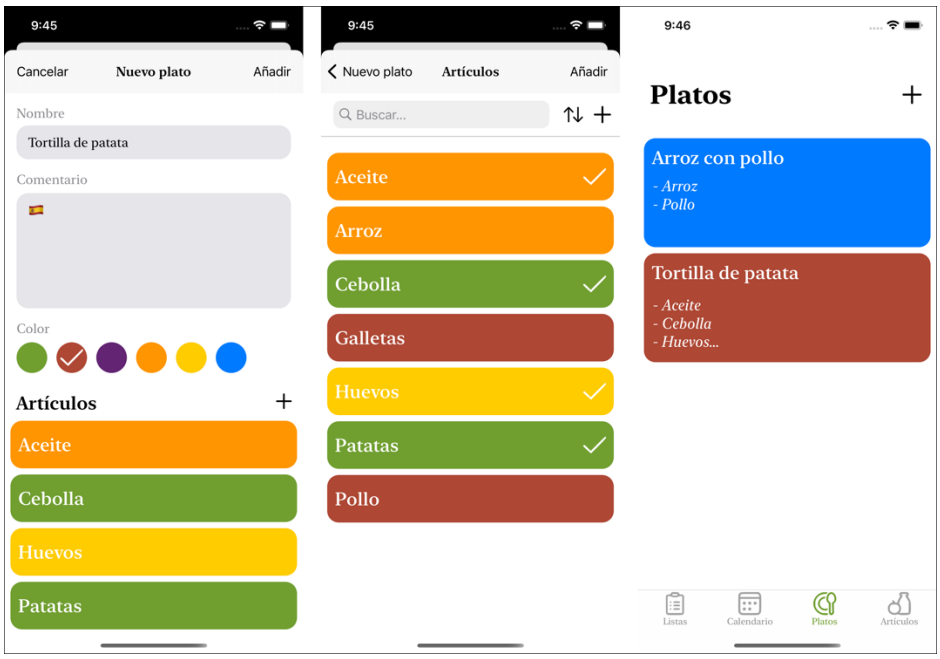

Figura 54: Platos

En la imagen del centro, podemos ver la lista de artículos a seleccionar. Es posible buscar por nombre, así como ordenar los artículos visibles o crear nuevos en caso de que sea necesario.

Por último, vemos listados los platos ya creados. En caso de que tengamos asignado alguno al día de hoy, nos aparecerá el primero.

El calendario es lo que nos permite hacer una organización temporal en la aplicación. En las siguientes imágenes, se muestra la vista mensual del calendario.

Es posible seleccionar los diferentes días y a partir de aquí, asignar a cada día los platos o artículos que queramos en cada sección. Al hacer clic en añadir (+), nos encontraremos con la pantalla que aparece en el centro en las siguientes imágenes. Podremos buscar, ordenar y añadir tanto artículos como platos. El resultado puede verse en la imagen que aparece a la derecha.

En este punto, podremos seleccionar cualquier dia y ver los platos o artículos asignados. Además, asignar elementos a los días hará que se nos genere una lista de la compra automáticamente.

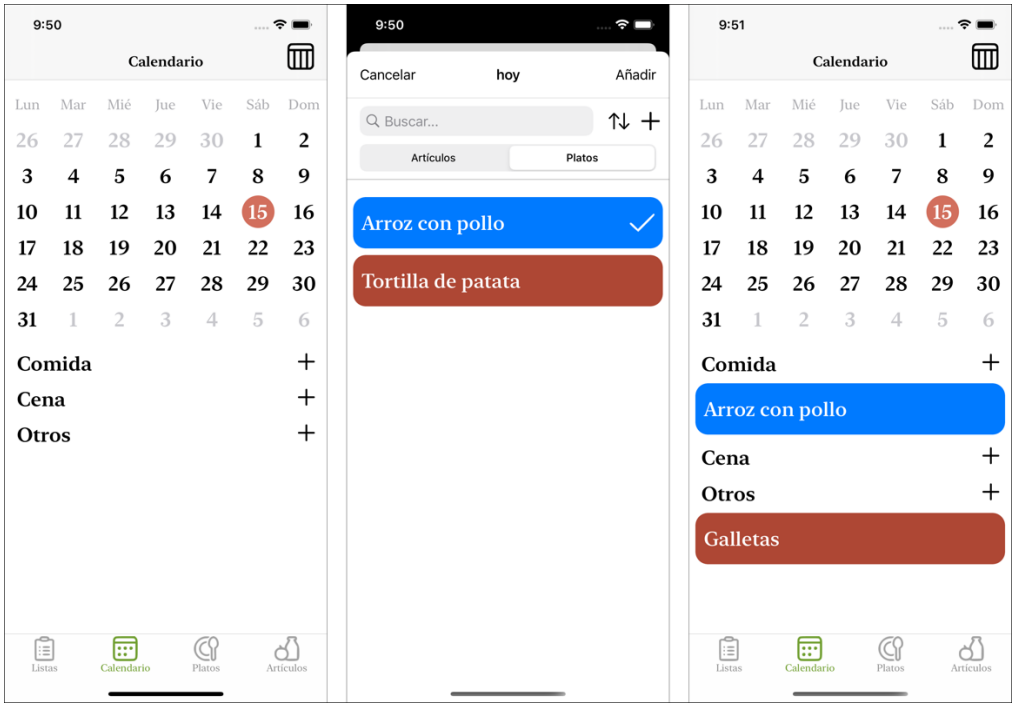

Figura 55: Calendario

Por último, tenemos las listas. En la siguiente imagen podemos ver la lista generada automáticamente, así como el contenido de la lista. Es posible asignar nuevos artículos o eliminarlos en caso de que ya los tengamos y no

necesitemos comprarlos. Además, es posible acceder al modo compra haciendo clic en el carro de la compra.

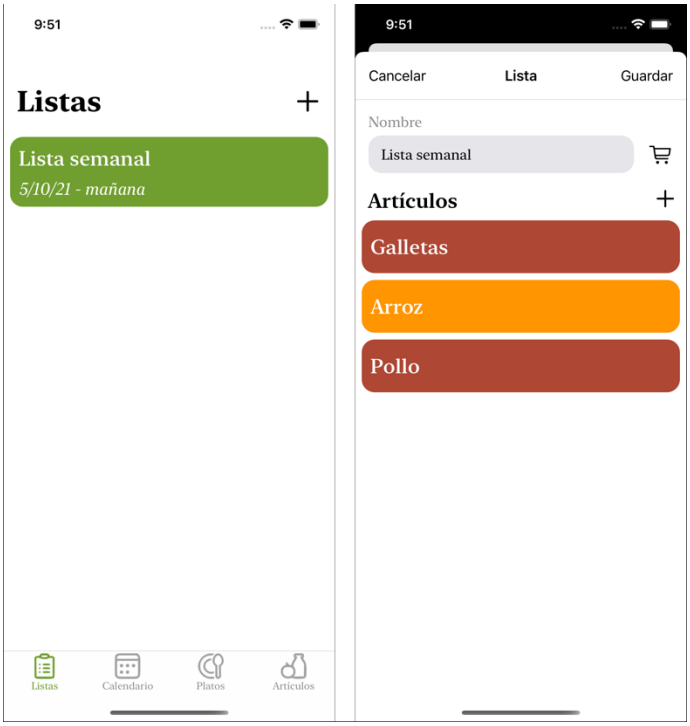

Figura 56: Listas

Es posible acceder al modo comprar también desde el menú principal, desplegando el menú de opciones con una pulsación larga. Esto también servirá para eliminar una lista. De la misma manera, es posible eliminar artículos, platos o elementos del calendario. Eso sí, se debe tener en cuenta que si un artículo se elimina, este se eliminará también de los platos o listas que lo contengan.

En la captura que se puede ver en la derecha en las siguientes imágenes, se muestra el modo compra. Cuando se compre un artículo puede marcarse y este se desplazará a la parte inferior de la lista. Cuando la compra esté terminada, se debe hacer clic en terminar.

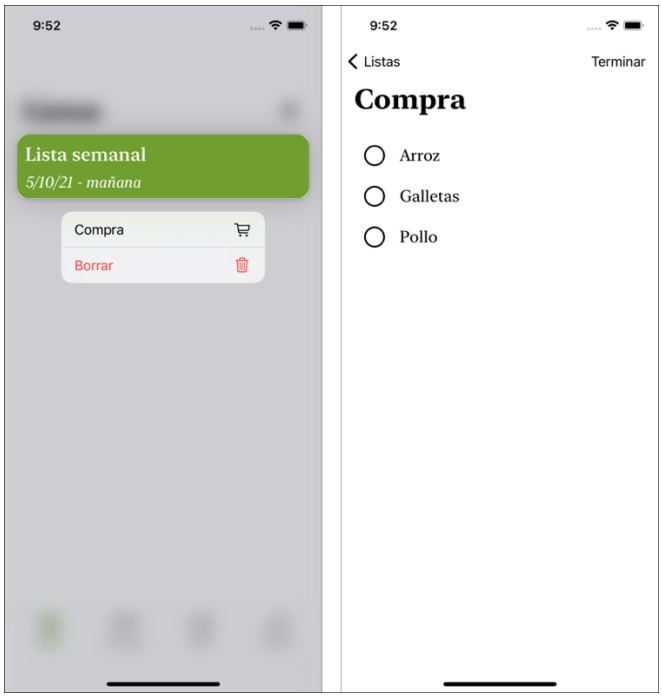

Figura 57: Modo compra

Cuando hayamos terminado una compra, veremos la lista marcada con un *check*. Es posible también crear lista de manera manual y asignarle los artículos que queramos.

Figura 58: Lista completada

9.2 Instrucciones de compilación

Instalar una nueva aplicación de iOS no es tan sencillo como en Android, donde puedes copiar el archivo empaquetado de la aplicación (apk) e instalarlo directamente. En este documento se recogen las instrucciones para compilar y ejecutar la aplicación entregada mediante Xcode.

Para facilitar la compilación, se ha desactivado la función de iCloud. Compilar la aplicación con esto activo requiere estar dado de alta en el Apple Developer Program, por lo que no puede hacerse con una cuenta gratuita. En cualquier caso, se van a incluir aquí las instrucciones para poder restaurar este comportamiento.

Para compilar la aplicación, es necesario abrir el proyecto con Xcode. Al hacer clic en el botón *play* podemos obtener un error por la falta de certificado. En este caso, será necesario cambiar el *Bundle Identifier*, para que no esté duplicado, y volver a intentarlo. Con esto debería ser suficiente.

En caso de querer probar todas las funciones de la aplicación, habrá que activar el almacenamiento en iCloud. Esto requiere una cuenta de pago y algunas acciones que se describen a continuación.

Lo primero, será necesario añadir la capacidad de iCloud en el proyecto. Para esto, en la misma pantalla que contiene el *Bundle Indentifier* (Targets -> Foodlisting -> Signing & Capabilities) añadiremos iCloud. Esto se consigue a partir del botón '+ Capability' que se encuentra en la parte superior izquierda. Después de esto, habrá que marcar la casilla iCloud en la sección de servicios y asignarle un contenedor. Este contenedor puede existir o crearse uno nuevo. En este punto, veremos que también se ha añadido la capacidad de notificaciones push y modos en segundo plano. Esto es algo que necesita iCloud.

El resultado debe ser algo parecido a lo que se ve en la siguiente imagen.

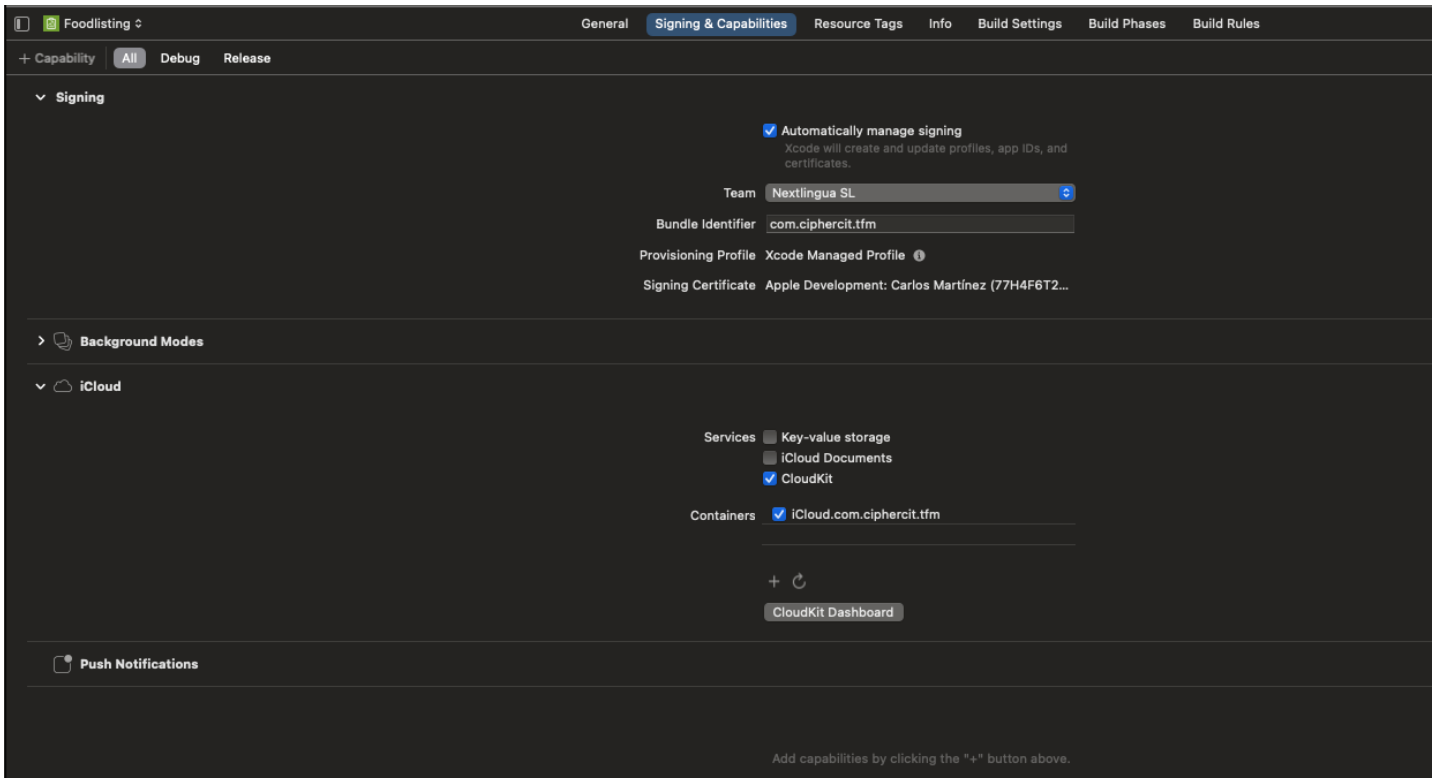

Figura 59: Xcode con iCloud configurado

Por último, tendremos que ir al archivo *AppDelegate.swift*. Aquí encontraremos el contenedor persistente de CoreData. Para activar el contendor de iCloud será necesario descomentar la línea 42 y comentar la línea 43. Después de esto, al compilar y ejecutar la aplicación ya se sincronizará entre dispositivos.

Cabe destacar, que esta función tiene un retardo considerable si se prueba en emuladores, por lo que se recomienda probar en dispositivos físicos.