UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

18|19

Trabajo fin de grado

Estudio y prototipo de una aplicación de la compra inteligente

Miguel Álvarez Lesmes

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio y prototipo de una aplicación de la compra inteligente

Autor: Miguel Álvarez Lesmes
Tutor: Alejandro Bellogín Kouki
Ponente: Iván Cantador Guitérrez

junio 2019

Miguel Álvarez Lesmes
Estudio y prototipo de una aplicación de la compra inteligente

Miguel Álvarez Lesmes

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Agradecer a mi compañero Sergio Sanchez por ayudarme con mis dudas sobre Flutter y por prestarme su MacBook para hacer la compilación de la aplicación para iOS.

Agradecer la ayuda y el apoyo de mi tutor Alejandro Bellogín a lo largo de todo el proceso del trabajo de fin de grado.

Agradecer a mi familia por apoyarme, ayudarme y aconsejarme mientras me dedicaba a desarrollar y probar la aplicación.

Agradecer a mi compañero Carlos Gonzalez por ayudarme con cualquier duda que tuviese y estar siempre atento para poder ofrecer su ayuda.

También agredecer a todos mis compañeros y personas que han utilizado y dado su opinión sobre la aplicación.

Resumen

La sociedad de hoy en día avanza muy rápido y se apoya cada vez más en la tecnología. El mundo tal y como lo conocemos está sufriendo una transformación tecnológica, desde los ordenadores y los teléfonos móviles hasta los coches y casas inteligentes. Esto hace que nosotros evolucionemos junto a este movimiento, cambiando nuestras costumbres y hábitos adaptándolos a este nuevo entorno tecnológico. Donde cada vez más pedimos comida a domicilio, reservamos hoteles y viajes por internet y compramos desde el móvil con el simple movimiento de un pulgar.

Sin embargo, la mayoría de las personas seguimos acudiendo a los supermercados para hacer la compra de las cosas que más necesitamos, perdiendo gran parte de nuestro tiempo en hacer una tarea simple y repetitiva, buscando en un laberinto de productos para terminar en una cola que parece interminable. Las personas actualmente buscan soluciones simples y sencillas que les faciliten la vida, la compra online de supermercados está empezando a surgir para intentar cubrir esta necesidad, no obstante, aún está en pleno desarrollo y aún existen problemas a solucionar. Algunos de los supermercados online sólo ofrecen soluciones a través de páginas web, siendo complicadas y poco intuitivas para el usuario, sin ofrecer recomendaciones ni predicciones sobre nuestras compras. Todo esto, sin contar el número de supermercados diferentes donde poder comprar, lo que resulta en que la comparación de precios se convierte en una tarea casi imposible.

La idea principal de este Trabajo de Fin de Grado es plantear y proponer una solución a estos problemas, desarrollando ShopIntelli, una aplicación para dispositivos móviles, tanto Android como iOS, que genere recomendaciones y predicciones de listas de la compra para clientes de los distintos supermercados, unificando sus productos y simplificando la manera de hacer la compra gracias a una interfaz simple e intuitiva. La aplicación ShopIntelli se centra en simplificar la tarea de hacer la compra, agrupando los productos de diferentes supermercados y generando recomendaciones personalizadas para el usuario. Esto lo hace junto con una interfaz gráfica muy intuitiva y simple de usar. Además de ofrecer la posibilidad de crear listas grupales donde múltiples personas puedan participar.

Palabras clave

Supermercado, aplicación móvil, Android, iOS, Flutter, Firebase, Dart, ShopIntelli.

Abstract

Today's society moves very fast and more and more it supports itself on technology to do so. The world how we know it is suffering a technological transformation, from computers and smartphones to smart houses and cars. This makes us evolve beside this movement, changing our practices and habits adapting them to this new technological environment, where, more and more, we order food, book hotels, and flights on the internet and buy things with our phones on simple movement of a thumb.

However, most people keep buying groceries at grocery stores, losing big amount of our time, doing a simple and repetitive task, looking in a maze of groceries to end up in a queue that seems endless. Nowadays, people seek for simple and easy solutions that simplify their lives, online grocery shopping is starting to emerge attempting to meet these requirements, nonetheless, it is still in early development and some issues are still unresolved. Some online grocery stores only offer web-based solutions, being complicated and little intuitive to the user, not offering recommendations nor predictions on our purchases. Additionally, the number of grocery stores to buy gets larger and larger every year, resulting in a difficult (or almost impossible) price comparison.

The main idea of this Bachelor thesis is to propose a solution to these problems, developing ShopIntelli, a smartphone app, for Android and iOS, that will generate recommendations and predictions on grocery shopping lists for the clients of the different grocery stores, unifying their products and simplifying the overall way of doing grocery shopping with the help of a simple and intuitive graphic interface. The main focus of the ShopIntelli app is to simplify the task of grocery shopping, grouping the products of different grocery stores and generating personalized recommendations and predictions on shopping lists of future purchases. This is achieved along with an intuitive and simple graphical interface that makes the app easier to use. In addition to offer the possibility to create shopping lists along with other people.

For the development of the app, I have used the framework Flutter and the programming language Dart, that together allow creating beautiful apps and native-compiled for Android and iOS with only one source code. I have also used Firebase for the back-end server and database.

Keywords

Grocery store, grocery shopping, smartphone app, Android, iOS, Flutter, Firebase, Dart, ShoppIntelli.

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación del proyecto

Hoy en día, a pesar de ser una tarea simple y repetitiva, la gran mayoría de las personas seguimos haciendo la compra acudiendo a nuestros supermercados de confianza, con el gasto de tiempo y de energía que ello supone. Las listas de la compra acaban siendo iguales o muy similares, sin mencionar la cantidad de veces que se nos olvida añadir productos necesarios. Además, en entornos familiares o en grupos de personas, las listas pueden llegar a ser muy largas y debiendo cubrir las necesidades de todos los integrantes, conllevando así a la necesidad de organización y coordinación a la hora de comprar. En resumen, la tarea de hacer la compra puede llegar a ser desagradable, aburrida y en ocasiones complicada. Como consecuencia surge la necesidad de simplificar la tarea de hacer la compra, de ahí proviene la motivación de crear una aplicación intuitiva y simple de utilizar que permita al usuario comprar sin tener que acudir al supermercado, además de ofrecer la posibilidad de crear listas de la compra inteligentes que sean capaces de predecir, recomendar productos y ser compartidas con varias personas. Otra motivación importante de la realización de este trabajo de fin de grado es la posibilidad de aprender un nuevo kit de desarrollo para aplicaciones móviles desarrollado por Google llamado Flutter.

1.2. Objetivos

El principal objetivo de la aplicación es simplificar la tarea de hacer la compra, reduciendo el tiempo de esta tarea y ayudando al usuario a hacer compras más rápidas e inteligentes.

Los objetivos generales que debe seguir la aplicación son los siguientes:

- **Ser intuitiva y simple de utilizar**: Ha de ser intuitiva de utilizar, debe tener una interfaz gráfica amigable para que el usuario no se pierda. Las tareas se pueden completar fácil y rápidamente.
- **Listas de la compra compartidas**: Las listas de la compra pueden compartirse con múltiples personas y se sincronizan de forma instantánea. Las personas que estén en la misma lista de la compra pueden añadir o quitar productos de forma concurrente.

- **Búsqueda de productos fácil e intuitiva**: Existirán multitud de filtros para facilitar la búsqueda de productos.
- **Agrupación de productos de distintos supermercados**: Los productos disponibles pertenecerán a múltiples supermercados diferentes, resultando así en búsquedas mucho más amplias y completas.
- **Recomendación de productos**: Existirá la posibilidad de solicitar recomendaciones de productos.
- **Predicción de listas de la compra**La aplicación será capaz de predecir listas de la compra futuras, basándose en listas de la compra anteriores.

1.3. Estructura del trabajo

La memoria consta de los siguientes capítulos:

- **Estado del arte:** En este capítulo se realiza un análisis sobre las tecnologías disponibles para el desarrollo móvil, además de hacer un estudio sobre posibles aplicaciones móviles similares que se encuentran en el mercado, y un estudio de los algoritmos de recomendación y predicción de listas de la compra que existen en la actualidad.
- **Diseño:** Se realizará un análisis sobre los requisitos funcionales y no funcionales de la aplicación. Además de explicarse el diseño escogido para el desarrollo de la aplicación y de los componentes que la componen.
- **Desarrollo:** En este capítulo se explicará todo el proceso que se ha seguido para el desarrollo de la aplicación, además de las decisiones que se han ido tomando.
- **Pruebas:** Se explicará la metodología que se ha seguido para realizar las pruebas sobre la aplicación. Además, también se comentarán los datos obtenidos de los experimentos sobre las recomendaciones y predicciones de las listas de la compra.
- **Conclusiones y trabajo futuro:** Para terminar, sacaremos conclusiones sobre el trabajo realizado y se comentará el posible trabajo futuro que queda por hacer para esta aplicación.

2

Estado del arte

2.1. Tecnologías de desarrollo de aplicaciones móviles

Hoy en día, las tecnologías más comunes para desarrollo de aplicaciones móviles son XCode con Swift para desarrollo de aplicaciones móviles en iOS y Android Studio con Java o Kotlin para desarrollo de aplicaciones móviles en Android. Por lo que para poder desarrollar una aplicación para ambos sistemas operativos es necesario dos códigos fuente distintos para el mismo propósito.

Sin embargo, poco a poco están surgiendo otras alternativas que permiten desarrollar aplicaciones para ambos sistemas operativos sin necesidad de tener dos códigos fuente diferentes. Estas tecnologías alcanzan este objetivo de dos maneras diferentes: creando una aplicación web embebida en una aplicación móvil dando la sensación de que es una aplicación nativa o generando el código nativo para cada sistema operativo a partir de una sola fuente de código.

2.1.1. Cordova – Apache

Cordova de Apache [1], es una de las alternativas para desarrollar aplicaciones con un solo código fuente. Permite utilizar tecnologías web como HTML5, CSS3 y JavaScript para desarrollar aplicaciones multiplataforma . La aplicación se ejecuta dentro de una aplicación envoltura específica para cada plataforma, teniendo así acceso a las APIs, sensores, datos y hardware estándares del dispositivo.

De esta manera, se puede desarrollar una aplicación web por separado y en caso de necesitar tener aplicaciones móviles que cumplan la misma funcionalidad que la página web, gracias a Cordova se pueden generar aplicaciones móviles fácilmente, reduciendo considerablemente el tiempo y esfuerzo de desarrollar aplicaciones para las diferentes plataformas. En cambio, al no generar ningún código nativo, las aplicaciones pueden tardar más en procesarse y ser un poco más lentas, además de que no suelen tener soporte para efectos visuales específicos de cada sistema operativo [2].

2.1.2. React Native – Facebook

React Native [3] es una tecnología novedosa, que ha empezado a crecer en los últimos años, que además cuenta con el respaldo de Facebook que está apostando fuertemente por ella. Este framework permite desarrollar aplicaciones móviles gracias a una sola fuente de código. Funciona con Node y NPM, dando así acceso a la posibilidad de utilizar la gran cantidad de herramientas y librerías ya desarrolladas en JavaScript. React Native ofrece una experiencia cercana a la nativa, esto lo consigue gracias a que utiliza componentes visuales nativos y mantiene las animaciones nativas de la plataforma donde se ejecuta y se puede considerar un híbrido entre una Web App y una aplicación nativa. Además, también incluye una herramienta muy útil en el desarrollo de aplicaciones móviles, el "Hot Reload", que permite realizar cambios sin necesidad de recompilar cada vez la aplicación, aumentando considerablemente la velocidad de desarrollo.

No obstante, hay que tener en cuenta que es una tecnología muy novedosa y sigue en desarrollo, y puede llegar ocurrir que en ocasiones sea necesario añadir código nativo para poder conseguir una funcionalidad específica, también es posible encontrarse con problemas difíciles de resolver al estar en fases iniciales y poco estables [4]. Al estar en pleno crecimiento, será necesario ir realizando actualizaciones continuamente para asegurarse del mantenimiento de la aplicación.

2.1.3. Flutter – Google

Flutter [5] es un kit de desarrollo de software gratis de código abierto lanzado por Google que permite crear aplicaciones con aspecto nativo, tanto para Android como para iOS, con el mismo código fuente. Flutter lleva existiendo desde 2015, pero ha sido desde hace poco que ha empezado a captar más la atención de los desarrolladores móviles. Para el desarrollo de aplicaciones en Flutter se utiliza Dart, un lenguaje de programación simple orientado a objetos.

La idea principal de Flutter gira en torno a los widgets. La totalidad de la interfaz gráfica está hecha a partir de widgets o combinación de los mismos, donde cada uno define un elemento de la interfaz, como podrían ser botones, estilos de elementos, posicionamiento y demás. Una cosa interesante sobre Flutter es que ya aporta una gran cantidad de widgets preparados, con un aspecto nativo y bonito, agilizando así la creación de las aplicaciones considerablemente, sin dejar de ofrecer la posibilidad de crear widgets totalmente personalizados. Flutter también ofrece la herramienta de "Hot Reload", aumentando así la productividad del desarrollo. Gracias a estas características y su clara y concisa documentación, Flutter permite desarrollar aplicaciones fácil y rápidamente.

El código Dart compila generando código nativo directamente, significando así en menos comunicaciones entre la aplicación y la plataforma, esto contribuye a inicializaciones más rápidas y menos problemas de rendimiento, dando así una sensación más cercana a la nativa.

Sin embargo, al igual que React Native, es una tecnología muy novedosa que acaba de aparecer en el mercado, está en un estado muy inicial y sigue en proceso de desarrollo, haciendo que sea poco estable y fiable para aplicaciones a gran escala [6]. Aún así, gracias al apoyo de Google y las continuas actualizaciones que recibe, parece que está avanzando rápidamente hacia una situación de mayor estabilidad y fiabilidad.

2.2. Algoritmos de recomendación

Para tener una buena idea de por dónde empezar para la parte de recomendación de la aplicación se realizó un estudio sobre los algoritmos de recomendación que existen hoy en día y ver si se podrían aplicar a nuestro caso, ya que el problema de predecir los productos que se van a añadir a una lista de la compra (una tarea repetitiva y con una frecuencia variable, como se ha discutido anteriormente) no es un problema que se haya estudiado mucho en el área.

En primer lugar, se revisaron las soluciones más básicas y sencillas, que serían las de las recomendaciones aleatorias, recomendaciones basadas en el historial o últimos productos, así como las recomendaciones basadas en los productos más populares.

También se encontraron variaciones de aproximaciones con métodos de clasificación gracias al Aprendizaje Automático (Machine Learning). Se encontraron trabajos en la literatura con dos tipos de métodos que usaban aprendizaje automático. Los primeros usan árboles de decisión, más concretamente utilizando el algoritmo C4.5 que, gracias al concepto de entropía de la información, es capaz de generar árboles de decisión a partir de un grupo de datos de entrenamiento que está formado por grupos de ejemplo ya clasificados y donde cada ejemplo está compuesto por diferentes atributos o características del ejemplo. El algoritmo C4.5 divide el conjunto de datos en cada nodo eligiendo un atributo que sea más efectivo a la hora de dividir el conjunto, su criterio es normalizado para la ganancia de información.

Otros trabajos encontrados tratan con métodos lineales como Perceptrones, Winnow y Naïve Bayes, con éstos métodos se podrían aprender las clases obtenidas del árbol de decisión obtenido con el algoritmo C4.5.

Mezclando algunos de los métodos mencionados anteriormente se podrían obtener otros métodos híbridos que serían capaces de medir diferentes niveles de medidas generando así predicciones más completas. Un artículo que explica más en detalle el resultado de experimentos utilizando dichos métodos es Predicting Customer Shopping Lists from Point-of-Sale Purchase Data [7].

También existen otros métodos como sería el uso de algoritmos a priori que permitiría extraer de los datos productos comprados frecuentemente juntos. En éste artículo [8] se explica el enfoque de utilizar algoritmos a priori. En él utiliza R y el paquete “arules” para predecir cual será el siguiente producto

que un usuario compraría teniendo en cuenta los productos comprados anteriormente.

2.3. Aplicaciones similares ya existentes

Antes de empezar el desarrollo de la aplicación, se realizó un análisis sobre algunas aplicaciones similares, para así comparar funcionalidades y decidir si se podrían incluir o mejorar algunos aspectos presentes en estas aplicaciones.

2.3.1. Aplicación móvil Mercadona

Mercadona tiene a disposición una aplicación móvil muy simple que permite realizar la compra de forma remota a través de tu teléfono móvil. Tiene un buscador sencillo que te permite realizar búsquedas por nombre de producto o por categoría, los productos vienen con imagen y precio. También proporciona una lista de “habituales” que te indica los productos y cantidades que sueles comprar. No es necesario iniciar sesión para poder buscar productos.

En resumen, la aplicación resulta intuitiva y simple de utilizar, se encuentran fácilmente los productos y la sensación general es positiva, pero actualmente solo está disponible en Valencia.

2.3.2. Aplicación móvil Lidl

El supermercado Lidl también tiene una aplicación móvil que permite realizar la compra online a través de tu teléfono móvil. En su página de inicio permite realizar búsqueda de productos por nombre o categoría, la búsqueda no es muy intuitiva y no parece funcionar muy bien. Los productos tienen múltiples imágenes adjuntas y una breve descripción. Existe un apartado en el que aparecen folletos online en los que aparecen ofertas especiales. No es necesario iniciar sesión para poder buscar productos. Pero no permite crear listas de la compra o guardar listas anteriores. Al parecer existen algunos problemas con las actualizaciones.

2.3.3. Aplicación móvil Carrefour

La aplicación móvil del Carrefour ofrece la posibilidad de realizar compras online gracias al teléfono móvil. Esta aplicación es un poco más compleja que las anteriores, ofrece otras funcionalidades a parte de la compra online como: te permite buscar cuál es la tienda más cercana a ti y buscar gasolineras Carrefour en el mapa. También pone a disposición folletos además del uso de cupones gracias a su tarjeta Club. A diferencia de las anteriores, es la única que permite crear listas de la compra. La búsqueda es muy completa, permite realizar búsquedas de productos por: nombre, categoría, marca,

formato, características, sabor, ingredientes y muchos otros filtros, además también permite ordenar el resultado por: Bio, Top ventas, Precio por kilo y por precio (de menor a mayor). Incluso incluye la posibilidad de realizar un escáner de código de barras para encontrar productos.

En resumen, es una aplicación muy completa, pero que puede llegar a ser un poco confusa para algunos usuarios, además en algunas ocasiones las búsquedas son muy lentas y es necesario registrarse para realizar la gran mayoría de acciones en la aplicación.

2.3.4. Aplicación móvil Deliberry

Esta aplicación [9] no se trata de ningún supermercado, si no de una empresa que se encarga de reunir los productos de algunos supermercados y ofrecer el servicio de compra online y envío a domicilio. En su página principal te permite elegir de qué supermercado quieres hacer tu pedido, a continuación, puedes realizar una búsqueda por nombre o por categoría a la cual se le pueden aplicar filtros de: tipo de producto, marca, compañía y variedad. Ofrece la posibilidad de repetir compras o de crear listas de la compra favoritas para realizar pedidos rápidamente. Es necesario estar registrado para realizar el pedido o para crear listas de la compra. En general es una aplicación fácil e intuitiva, con un buscador bastante fiable y que te permite realizar pedidos a distintos supermercados desde una misma aplicación.

2.3.5. Aplicación móvil Bring!

Bring! es una aplicación móvil que te permite crear listas de la compra y compartirlas con tus amigos. A diferencia de las aplicaciones anteriores, Bring no tiene productos reales de supermercados, no obstante, utiliza tarjetitas que representan productos comunes en listas de la compra, además de dar la posibilidad de crear y modificar las tarjetitas para que se acerquen más a lo que realmente quieres representar. Ofrece la posibilidad de crear recomendaciones basadas en tus anteriores compras, esta funcionalidad está aún en desarrollo y no parece funcionar del todo correctamente. También existe un apartado en el que se muestran recetas de inspiración. En generar resulta una aplicación muy sencilla e intuitiva de utilizar que permite crear listas de la compra compartidas con tus amigos o familiares.

3

Diseño

3.1. Ciclo de vida

Primeramente, comentar que se ha seguido un modelo de cascada retroalimentada como ciclo de vida para el desarrollo de la aplicación, representado en la Figura 3.1.

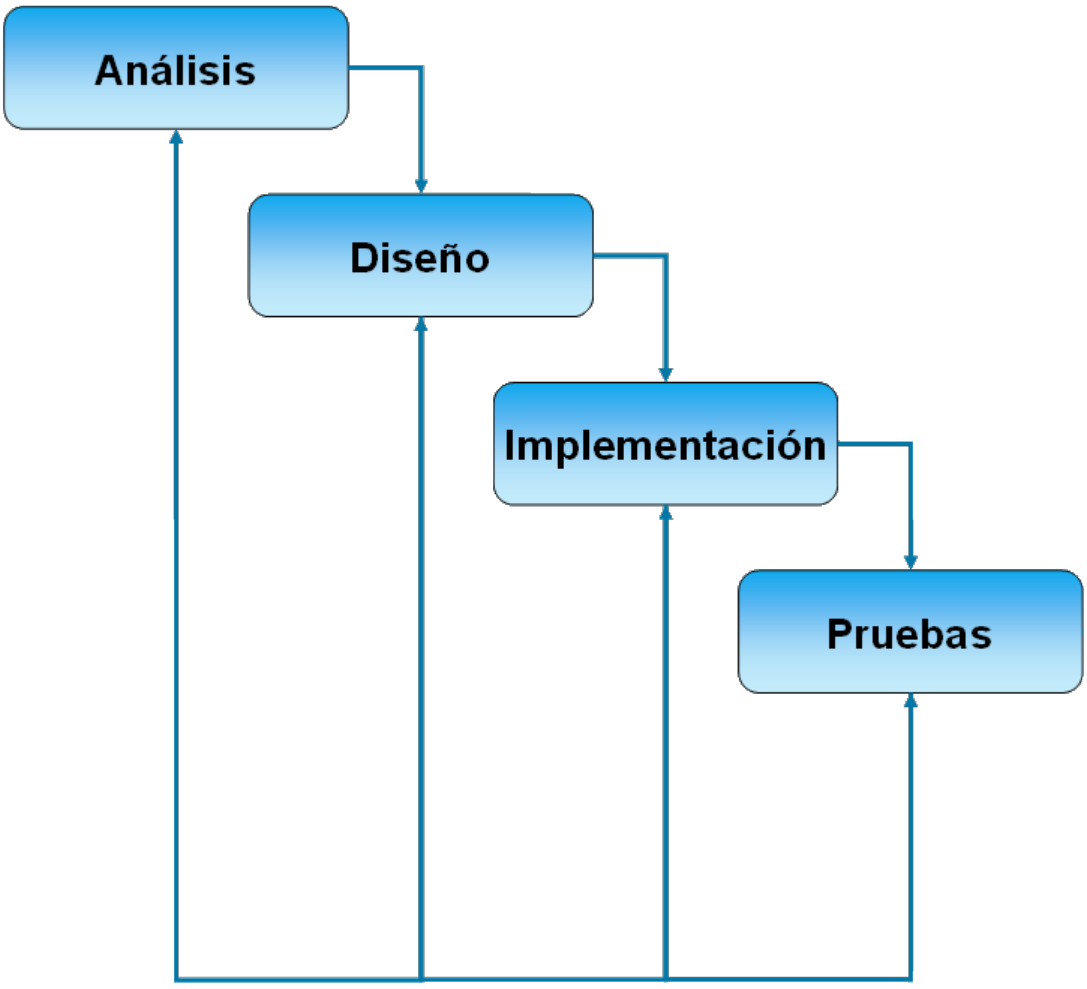


Figura 3.1: Modelo cascada retroalimentada.

- **Análisis:** En la primera fase se realizó un estudio sobre posibles aplicaciones similares. Se investigaron las tecnologías disponibles para el desarrollo de aplicaciones móviles. Por último, se definieron los requisitos funcionales de la aplicación.
- **Diseño:** Basándose en el análisis y requisitos definidos anteriormente, se crearon maquetas para definir las ideas de diseño de la interfaz gráfica y la estructura de la aplicación. Se tomaron decisiones sobre la estructura de las bases de datos generando así esquemas para su posterior uso. Además, se generó un diagrama de casos de uso y diagramas de secuencia.
- **Implementación:** Durante esta etapa se desarrolló el código fuente de la aplicación basándose en los esquemas, diagramas y maquetas de la fase anterior. Se puso en marcha el servidor y la base de datos remota. A lo largo de esta fase fue necesario volver a etapas anteriores para realizar algunos pequeños ajustes, debido a imprevistos y necesidades que fueron surgiendo.
- **Pruebas:** Se fueron realizando pruebas de caja blanca y negra a lo largo de la fase de implementación. Además de proporcionar la aplicación a diversos usuarios de prueba y recibir feedback para incorporar mejoras y resolución de errores.

3.2. Requisitos de la aplicación

En el siguiente apartado se definen los requisitos funcionales y no funcionales de la aplicación llevados a cabo en la fase de análisis.

3.2.1. Requisitos funcionales

RF-1.– Los productos serán reales y tendrán nombre, imagen, categoría y sub categoría.

RF-2.– Se podrán crear listas de la compra, con nombre y fecha programada de compra.

RF-3.– Se podrán configurar las listas de la compra de usuario. Cambiar el nombre, borrar la lista y añadir o borrar participantes.

RF-4.– Se podrán añadir un número de unidades deseadas de un producto a una lista de la compra.

RF-5.– Se podrán eliminar un número de unidades deseadas de un producto de una lista de la compra.

RF-6.– Cada usuario podrá tener varias listas de la compra y podrá elegir y cambiar entre cualquiera de ellas.

RF-7.– Se podrán crear enlaces de invitación a listas de la compra.

RF-8.– Los usuarios invitados a una lista de la compra tienen derecho a añadir y quitar productos de la misma.

RF-9.– Se podrá realizar una búsqueda por nombre de producto en una lista de la compra.

RF-10.– Se podrá realizar una búsqueda por nombre de producto en la base de datos.

RF-11.– A la hora de buscar productos se podrá filtrar los productos por categorías, por precio máximo y por tienda.

RF-12.– El usuario podrá solicitar recomendaciones personalizadas de productos para sus listas de la compra.

RF-13.– Existirá la posibilidad de cerrar sesión.

RF-14.– El usuario tendrá la posibilidad de escanear códigos de barra para realizar una búsqueda sobre un producto.

3.2.2. Requisitos no funcionales

RNF-1.– Las listas de la compra estarán sincronizadas con el servidor (En caso de tener conexión a internet).

RNF-2.– Será necesario tener iniciada la sesión para realizar cualquier tarea.

RNF-3.– Existirá un formulario de registro al primer inicio de la aplicación.

RNF-4.– El usuario permanecerá en la sesión hasta que este mismo la cierre.

RNF-5.– La interfaz de usuario será sencilla e intuitiva.

RNF-6.– Será necesario que el dispositivo móvil tenga acceso a internet.

RNF-7.– La búsqueda de productos deberá ser intuitiva y rápida.

3.3. Diseño de la aplicación

3.3.1. Diseño gráfico de la aplicación

Para el diseño gráfico de la aplicación se tuvieron en cuenta las pautas de Material Design, “Un lenguaje visual que sintetiza los principios clásicos de un buen diseño con la innovación de la tecnología de y la ciencia”. [10] De esta manera tenemos un modelo a seguir a lo largo de la aplicación que le dará un aspecto más bonito y profesional.

En primer lugar, se eligió la paleta de colores que se utilizará para la interfaz gráfica de la aplicación. Los colores secundarios fueron elegidos tal que existiese el suficiente contraste para mantener buena accesibilidad de la aplicación. La paleta de colores elegida se muestra a continuación en la Figura 3.2:

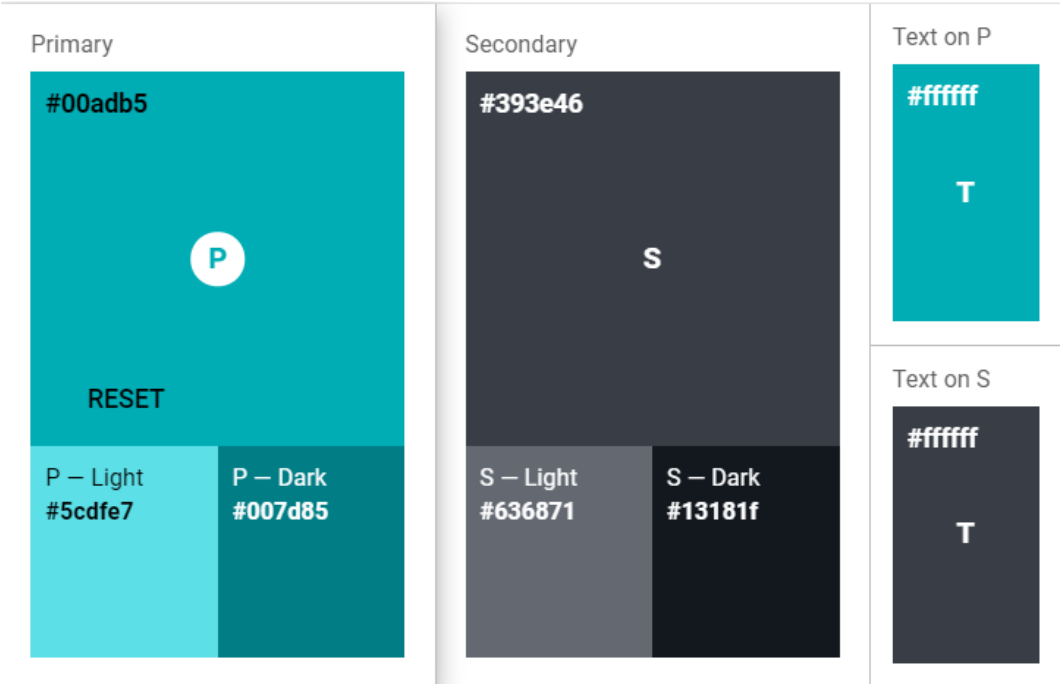


Figura 3.2: Paleta de colores elegida.

Durante el diseño de las maquetas se tuvo muy en consideración el objetivo de la sencillez y de la simplicidad de la aplicación, así como de la facilidad de realizar tareas en la misma, es por ello que se intentó minimizar el número de acciones necesarias para realizar una tarea, facilitando así el trabajo al usuario.

Destacar que las maquetas tuvieron un papel orientativo, para que a la hora de desarrollar se tuviese una idea general de cómo estructurar la aplicación, es por ello que se fueron añadiendo y realizando modificaciones sobre la marcha para poder cumplir con los requisitos y objetivos de la aplicación. Como se puede observar en la Figura 3.4, se fueron haciendo diferentes variantes de las vistas, para así ir probando variantes y eligiendo la que mejor encajase en la aplicación final. En la Figura 3.3 se muestra la maqueta de la página de inicio que se muestra al abrir la aplicación.

Figura 3.3: Maqueta de la página de inicio de la aplicación.

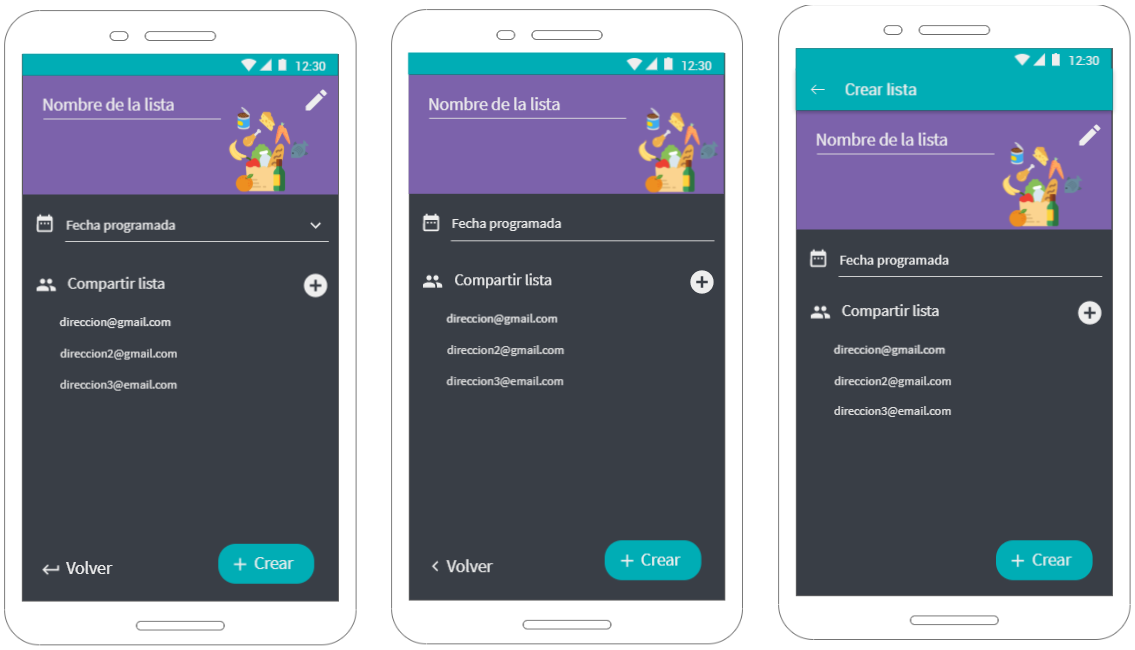


Figura 3.4: Múltiples maquetas para la página de creación de lista.

3.3.2. Diseño de la base de datos

Para trabajar con los datos de la aplicación se decidió utilizar una base de datos no relacional. La decisión de utilizar una base de datos no relacional fue tomada por las siguientes razones: la mayoría de aplicaciones móviles de hoy en día utilizan este tipo de bases de datos, esto es debido a que las aplicaciones móviles no tienden a solicitar grandes cantidades de datos, se realizan mayor cantidad de lecturas que de escrituras y por lo tanto no es tan necesario tener los datos tan bien estructurados, además de que en muchas ocasiones los datos no están totalmente completos, es por ello que el uso de una base de datos no relacional es mucho más adecuada para una aplicación móvil, ya que ofrece una mayor flexibilidad ante una base de datos relacional convencional.

A continuación, pasaremos a explicar la estructura de la base de datos Cloud Firestore [11] que hemos creado para nuestra aplicación. En la Figura 3.5 se muestra un diagrama orientativo de la estructura de la base de datos.

En el diagrama orientativo de la Figura 3.5 se pueden observar las diferentes colecciones representadas como rectángulos con bordes redondeados y los documentos como pequeños rectángulos con el borde de arriba a la derecha doblado. En los documentos se muestran los atributos que se espera que contengan, algunos atributos están seguidos de un asterisco, esto es debido a que representan atributos que pueden no existir en el documento. También se pueden apreciar flechas y líneas que unen diferentes documentos, esto representa relaciones entre los documentos, no son relaciones reales, pero sí orientativas. A continuación, pasaremos a entender lo que representan las diferentes colecciones de la base de datos, así como los atributos de los documentos. La base de datos está

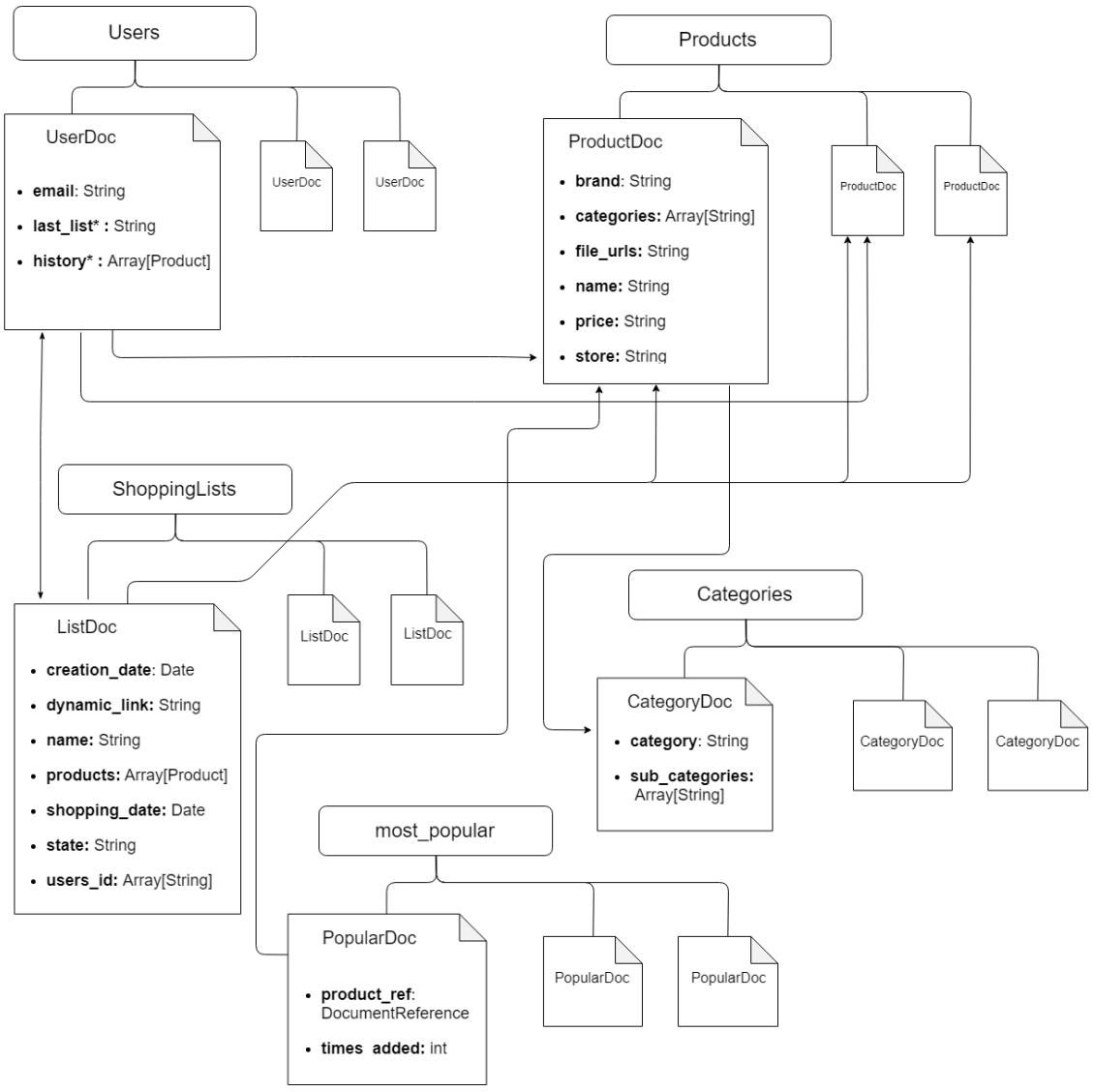


Figura 3.5: Diagrama orientativo de la estructura de la base de datos.

compuesta por 5 colecciones diferentes:

1.– **Users:** Esta colección agrupa los documentos que representan a los usuarios registrados en la aplicación. Los atributos que se espera que contengan los documentos de esta colección son los siguientes:

   Email: Es el email con el que se registró el usuario, no se espera que este atributo falte en el documento.

   Last_list: Representa la última lista que el usuario eligió. Es el identificador de una lista a la que pertenece el usuario. Puede que no exista (o valga null) si el usuario no pertenece a ninguna lista o fue borrado de una.

   History: Representa el historial de productos que el usuario ha añadido a listas de la compra. Consiste en una cola de longitud 10, compuesta de las referencias a los productos que el usuario añadió recientemente. Es posible que no exista (o valga null) en caso de que el usuario aún no haya añadido ningún producto.

2.– **Products:** En esta colección se encuentran todos los documentos que representan los diferentes productos disponibles en la aplicación. Los atributos que se espera que contengan los documentos de esta colección son los siguientes:

   Brand: Este atributo indica la marca del producto.

   Categories: Lista con los nombres de las categorías y subcategorías a las que pertenece el producto.

   File_urls Es el enlace directo a la imagen del producto.

   Name: Atributo que indica el nombre del producto.

   Price: El precio en euros del producto.

   Store: La tienda a la cual pertenece el producto.

3.– **ShoppingLists:** Aquí se encuentran los documentos que representan las listas de la compra creadas por los usuarios. Los atributos que se espera que contengan los documentos de esta colección son los siguientes:

   Creation_date: La fecha en la que la lista de la compra fue creada.

   Dynamic_link: Representa el enlace a enviar para invitar a una persona a la lista.

   Name: Es el nombre de la lista de la compra.

   Products: Este atributo representa los productos añadidos a la lista de la compra. Para mayor eficiencia y menor número de consultas a la base de datos, se trata de una lista de ítems. Estos ítems son el conjunto de un producto y el número de unidades añadidas a la lista. Los productos de los ítems no son referencias a su documento, si no que están totalmente integrados en el ítem, reduciendo así el número de peticiones al servidor y aumentando la velocidad de la aplicación.

   Shopping_date: Representa la fecha programada de compra. Esta puede ser asignada por los participantes de la lista de la compra o por las recomendaciones.

   State: Es el estado de la lista de la compra. Puede ser INPROCESS (en proceso), BUYING (comprando), FINISHED (comprada), DISPACHED (archivada) o UNKNOWN (en caso de algún error).

   Users_id: Este atributo representa los usuarios que pertenecen a la lista. Se guarda una lista con los ids de cada usuario. Como se puede observar, los usuarios no tienen ninguna referencia a qué listas pertenecen, esta decisión de diseño es debida a que de esta manera son necesarias menos consultas a la base de datos.

4.– **Categories:**En esta colección se almacenan las diversas categorías y subcategorías de los productos presentes en la aplicación. Los atributos que se espera que contengan los documentos de esta colección son los siguientes:

Category: Atributo que representa el nombre de la categoría.

Sub_categories : Es una lista con los nombres de todas las subcategorías de esta categoría.

5.– **Most_Popular:** Esta colección recoge los productos más populares, es decir los productos que más veces se han añadido a una lista. Los atributos que se espera que contengan los documentos de esta colección son los siguientes:

Product_ref: Es la referencia al producto que representa el objeto.

Times_added: Representa el número de veces que dicho producto fue añadido a una lista de la compra.

3.3.3. Diagrama de casos de uso

A continuación, pasaremos a hablar de los casos de uso de la aplicación, de esta manera podemos ver un poco mejor qué acciones puede realizar el usuario y los pasos para poder completarlas. En la Figura 3.6 se muestra un diagrama de casos de uso que muestra las posibles acciones de la aplicación y cómo realizarlas. Ahora pasaremos a explicar más en detalle cada caso de uso:

1.– Registrarse: El usuario se podrá registrar.

2.– Iniciar sesión: El usuario podrá iniciar su sesión. Para ello tendrá que haberse registrado anteriormente. Además, para poder hacer uso de cualquier característica de la aplicación el usuario tendrá que haber iniciado sesión primero.

3.– Crear lista de la compra: Acción básica de la aplicación, el usuario podrá crear listas de la compra, tantas como quiera.

4.– Añadir productos a una lista de la compra: El usuario podrá añadir productos a una lista de la compra. Para ello tendrá que buscar el producto y haber creado una lista de la compra anteriormente.

5.– Eliminar unidades de un producto: El usuario podrá eliminar unidades de un producto previamente añadido a una lista. Para ello tendrá que haber primero añadido el producto a la lista.

6.– Ver los detalles de un producto: El usuario podrá ver más detalles de un producto en cualquier parte de la aplicación, tanto en su lista de la compra como a la hora de buscar un producto.

7.– Pedir una recomendación: El usuario podrá solicitar una recomendación sobre productos al servidor.

8.– Configurar una lista de la compra: El usuario podrá configurar su lista de la compra. Podrá eliminar usuarios de la lista de la compra, así como cambiar el nombre de la lista de la compra.

9.– Borrar una lista de la compra: El usuario podrá borrar una lista de la compra previamente creada.

10.– Invitar usuarios a una lista de la compra El usuario podrá invitar a otros usuarios a participar en su lista de la compra. Para ello el usuario tendrá que haber creado una lista de la compra previamente.

11.– Buscar un producto: El usuario podrá buscar productos por nombre, también podrá filtrar los resultados por categorías y subcategorías, por precio máximo del producto y por tienda del mismo.

3.3.4. Diagrama de secuencia

Con el fin de entender mejor el funcionamiento de algunas de las acciones en la aplicación se crearon dos diagramas de secuencia de las dos tareas más importantes y complejas de la aplicación.

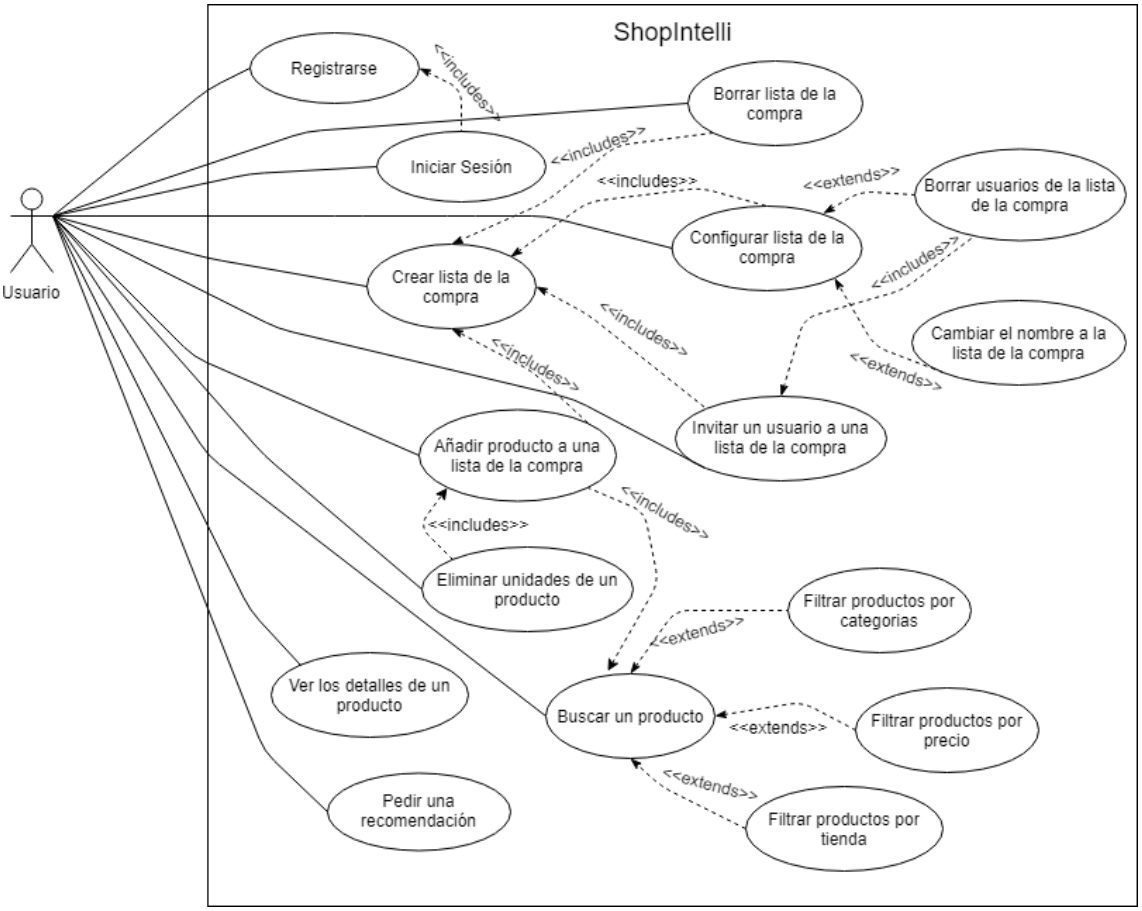


Figura 3.6: Diagrama de casos de uso.

En la Figura 3.7 se puede observar el diagrama de secuencia de realizar la acción de añadir un producto a una lista de la compra. Mencionar que el caso que representa el diagrama es el caso más sencillo, aquel en el que el usuario ya pertenece a una lista de la compra o tiene una lista de la compra seleccionada, en caso de que eso no se cumpla se le avisará de que tiene que crear una lista de la compra antes de poder añadir productos.

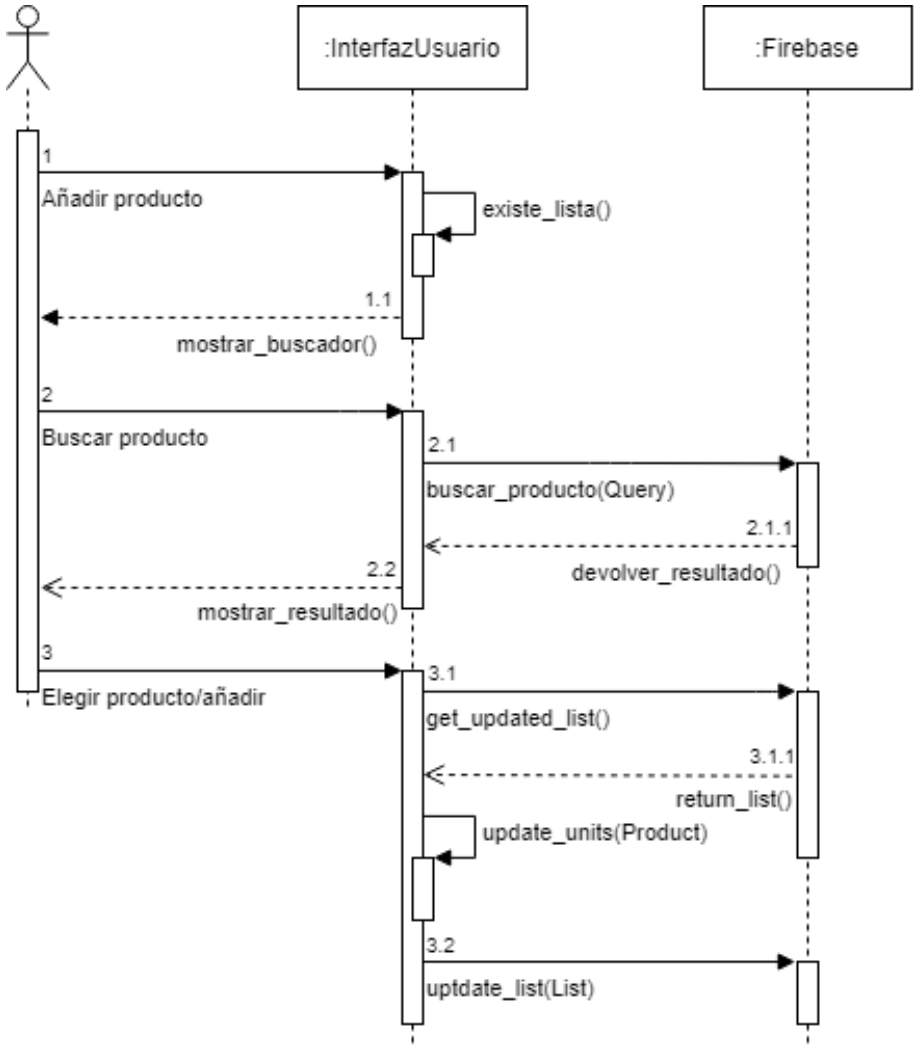


Figura 3.7: Diagrama de secuencia de añadir un producto a una lista de la compra.

En la Figura 3.8 se muestra el diagrama de secuencia de la acción de aceptar una invitación a una lista de la compra. Al igual que en el diagrama anterior se ha escogido uno de los casos más sencillos, ya que el usuario podría no tener descargada la aplicación, en ese caso el Dynamic link le mandaría o bien a la página web de la aplicación o a Google Play store para que se pueda descargar la aplicación, una vez descargada el proceso sería el mismo que el del diagrama. En el caso de que el usuario ya tenga la aplicación descargada, el Dynamic link la abrirá por él, si además ya está registrado y tiene una cuenta, será añadido automáticamente a la lista de la compra. Todo este proceso es posible gracias a la herramienta que ofrece Firebase, los Dynamic links [12] que guardan la información incluso si el usuario no tiene la aplicación descargada.

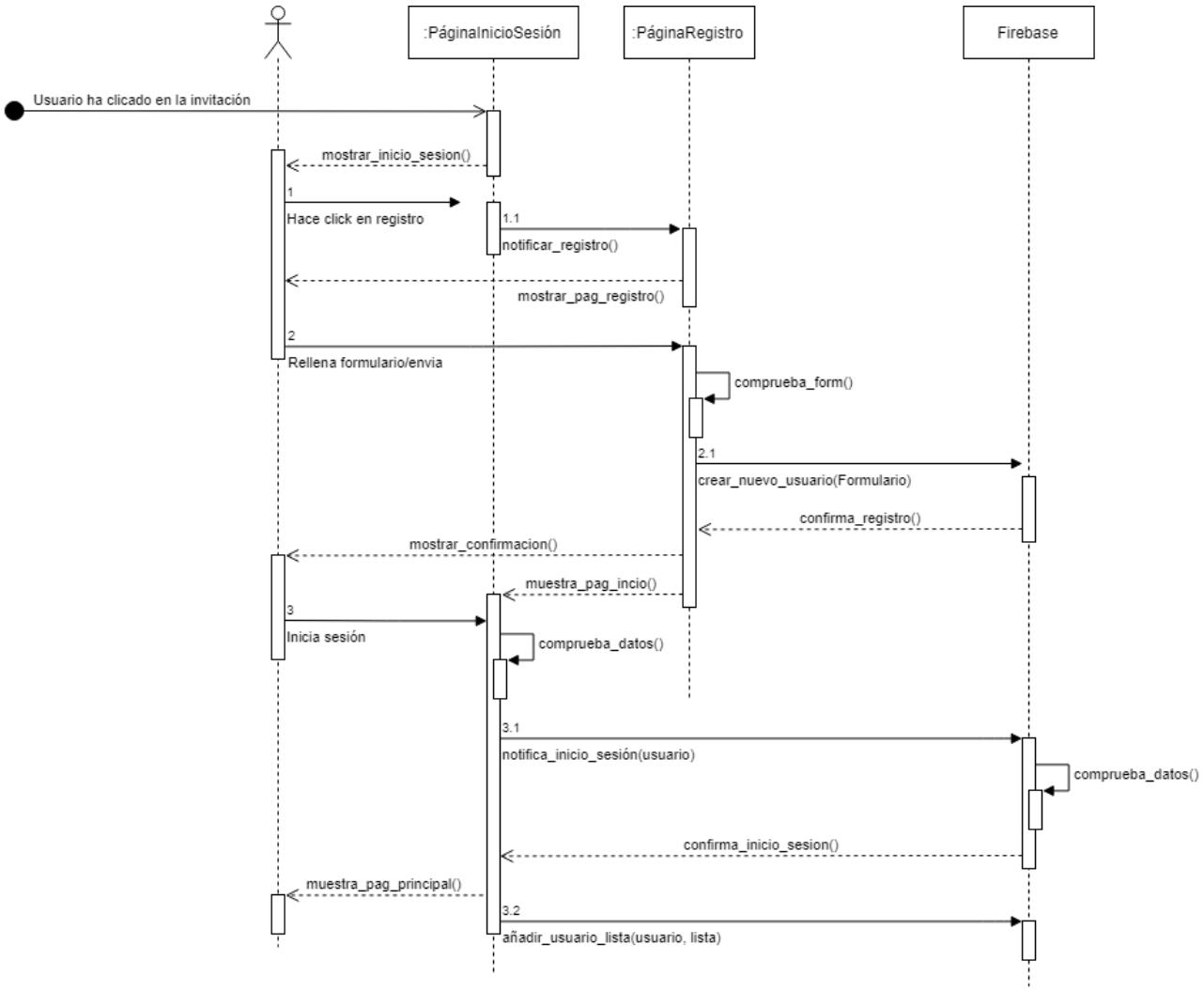


Figura 3.8: Diagrama de secuencia de aceptar invitación a lista de la compra.

4

Desarrollo

En este capítulo pasaremos a explicar más en detalle toda la parte del proceso de desarrollo que se ha llevado para la implementación de la aplicación.

4.1. Recopilación de datos

En primer lugar, se tuvieron que recopilar los datos de los productos que se iban a meter en la base de datos. Como uno de los requisitos de la aplicación es que “los productos tienen que ser reales y tienen que tener nombre, imagen, categoría y sub categoría” (RF-1), esto hizo que la tarea de recopilar la información fuera un poco más complicada. Finalmente se encontró una página web que utilizaba los productos de otros supermercados tal y como los necesitamos, el nombre de esta página es Deliberry [9].

Para poder recuperar los datos de la página web se tuvo que utilizar una técnica llamada “web scrapping”, que consiste en analizar páginas web con la intención de recopilar información de dicha página web, todo esto gracias a unos programas normalmente llamados arañas o spiders que se encargan de seguir unas pautas para la recopilación de datos, y su posterior almacenamiento, normalmente en archivos de tipo JSON.

Para la ejecución de esta técnica se decidió utilizar una librería Python llamada Scrapy [13], ya que consta de una documentación muy detallada y al ser en Python resulta muy rápido realizar programas-scripts que funcionen correctamente.

Gracias a Deliberry y Scrapy se recopiló la información de 14.407 productos reales de tres tiendas diferentes y en menos de dos minutos. La información recopilada de cada producto contenía los siguientes datos: tienda de origen, nombre del producto, precio, marca, enlace a la foto del producto y las categorías y subcategorías a las que pertenece el producto.

A continuación, se procedió a subir todos los datos de los productos al servidor de Firebase. Con la ayuda del lenguaje de programación Dart, se fue subiendo cada producto a la base de datos Firestore Cloud, con sus datos, además de crear una colección nueva que contendría los documentos que

representan las categorías y sus subcategorías.

También se buscaron bases de datos con productos relacionados con sus códigos de barras, se encontraron múltiples bases de datos bastante interesantes como por ejemplo las bases de datos EAN-SEARCH [14] y Barcodes Database [15]. Sin embargo el problema que se tuvo es que no se podían relacionar los productos de la base de datos obtenida con las arañas y los productos de las bases de datos de códigos de barras, por ello se decidió no implementar la parte de búsqueda de productos con códigos de barras.

4.2. Base de datos

Tras investigar sobre las bases de datos no relacionales se decidió escoger Firebase. Firebase es una plataforma de desarrollo móvil en la nube, ofrece muy buenas características como: Authentication, Cloud Firestore, Cloud Functions, Dynamic Links y muchas otras. Además, ofrece la posibilidad de ser utilizada de forma gratuita hasta unos límites que se planean que nuestra aplicación no va a superar, por lo que la hace una gran opción como servidor para nuestra aplicación.

Gracias a Cloud Firestore se puede tener una base de datos no relacional [16] en un servidor, que se actualiza en tiempo real con nuestra aplicación.

Con la funcionalidad que ofrece la parte de “Authentication” de Firebase, podemos tener toda la parte de autenticación de nuestra aplicación cubierta, ya que ofrece la posibilidad de administrar y crear usuarios con diferentes permisos, utilizar diferentes métodos de inicio y registro, además de facilitar toda la funcionalidad de autenticación dentro de nuestra aplicación ya que permite crear objetos que representan a los usuarios y que facilitan un montón la lógica del desarrollo.

Cloud Firestore es una base de datos no relacional que funciona parecido a MongoDB, hace uso de documentos, estos documentos pueden tener atributos de todo tipo (String, Integer, Array, Referencias a otros documentos, etc.), cada documento tiene un identificador único y forman parte de una colección. Las colecciones agrupan a documentos del mismo tipo. A diferencia de las bases de datos relacionales, Cloud Firestore trabaja con datos repetidos.

4.3. Arquitectura de la aplicación

Antes de empezar a explicar la arquitectura de la aplicación tenemos que explicar con qué plataforma se decidió desarrollar la aplicación. Tras investigar, se decidió utilizar el kit de desarrollo Flutter, por diferentes razones: al ser ShopIntelli la primera aplicación móvil desarrollada, se ha querido optar por una tecnología que sea más sencilla y fácil de aprender, Flutter es muy buena en ese sentido, ya que aporta gran cantidad de widgets y funcionalidades importantes directamente integradas sin perder la

posibilidad de crear implementaciones propias, también tiene una documentación muy buena y detallada que hace que sea mucho más sencillo aprender, además Flutter permite desarrollar aplicaciones para Android y iOS haciendo así la aplicación accesible a más grupos de usuarios.

Flutter es un tanto diferente al resto de kits de desarrollo de aplicaciones móviles más utilizados hoy en día, y es por ello que utilizar el famoso patrón modelo-vista-controlador hace que las cosas sean más complicadas, y al ser la primera vez que se utilizaba Flutter se decidió mantener las cosas sencillas y no utilizar este patrón de diseño. Este es uno de los mayores problemas que existen a día de hoy en Flutter, y algo a lo que la mayoría de los desarrolladores de Flutter se tienen que enfrentar, al ser Flutter una tecnología tan nueva en el mercado, no existen demasiadas opciones y variantes de cómo estructurar tu aplicación de manera sencilla y correcta.

Ahora pasaremos a ver la estructura interna de la aplicación. Se decidió separar los diferentes archivos de la siguiente manera:

- **Model:** En este paquete se encuentran las clases que representan la mayoría de los objetos de nuestro modelo de la base de datos. En ellas se implementaron métodos para transformar los datos recibidos de Firestore (en formato JSON) a objetos y viceversa, permitiendo así trabajar de manera más cómoda y sencilla con nuestros datos. Entre ellos se encuentran los siguientes:

  Categoria: Esta clase sirve para representar los objetos de tipo categoría, está compuesta por los atributos de nombre de categoría y una lista de subcategorías. También existen funciones para añadir y eliminar subcategorías de la lista. Además, se sobrescribió el método de comparador, para poder comparar las categorías por su nombre.

  Product: Clase para los objetos que representan productos, contiene todos los atributos necesarios para representar un producto, identificador del producto en la base de datos, nombre, marca, etc. También se tuvo que sobrescribir el método de comparación, ya que era necesario comparar los productos por su identificador único. Además, también existen métodos para subir los productos recopilados por la araña al servidor de Firebase.

  ListItem: Esta clase sirve para representar los ítems de una lista de la compra, junta un producto con el número de unidades añadidas del producto a la lista. Esta clase hereda de Producto, y le añade el atributo de unidades.

  ShoppingList: Representa las listas de la compra, tiene todos los atributos de las listas de la compra, nombre, fecha de compra, usuarios que pertenecen, etc. Se tuvieron que crear funciones para añadir y quitar unidades a un producto de la lista, estas funciones se encargan de buscar si el producto existe en la lista y si existe le suma o resta las unidades, si no se añade a la lista con dichas unidades o se elimina de la lista. Las listas están compuestas por ListItems.

  User: Clase para representar los usuarios de Firebase, en ella se guarda el email y el id del usuario.

- **Pages:** En este paquete se encuentran las vistas de la aplicación, los archivos están compuestos por los widgets y por la lógica de la aplicación.
- **Services:** En este paquete se encuentran los métodos que representan servicios de la aplicación, como podrían ser el servicio de autenticación, el servicio de actualizar las listas o de recuperar los productos del servidor dada una cierta query.
- **Widgets:** En este último paquete se pueden encontrar los widgets creados de forma personalizada o los widgets que se repiten con frecuencia en la aplicación para así simplificar el código.

4.4. Estructura de la aplicación

En primer lugar, para poder entender mejor la estructura de la aplicación se ha creado un diagrama que explica la navegación por la aplicación, que se muestra en la Figura 4.1

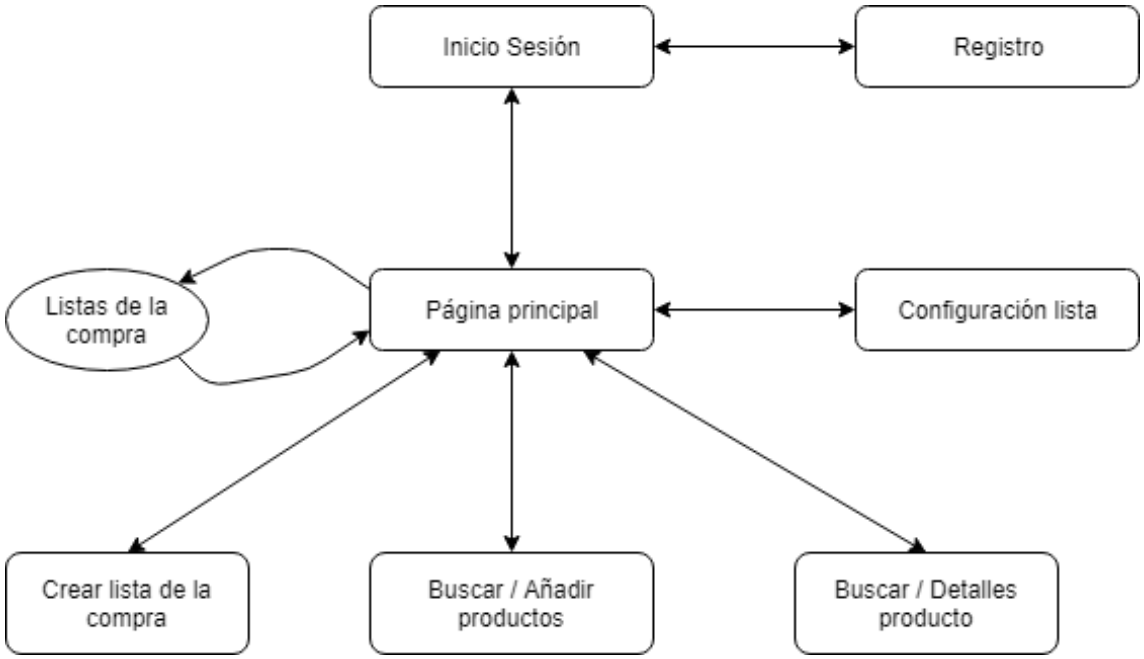
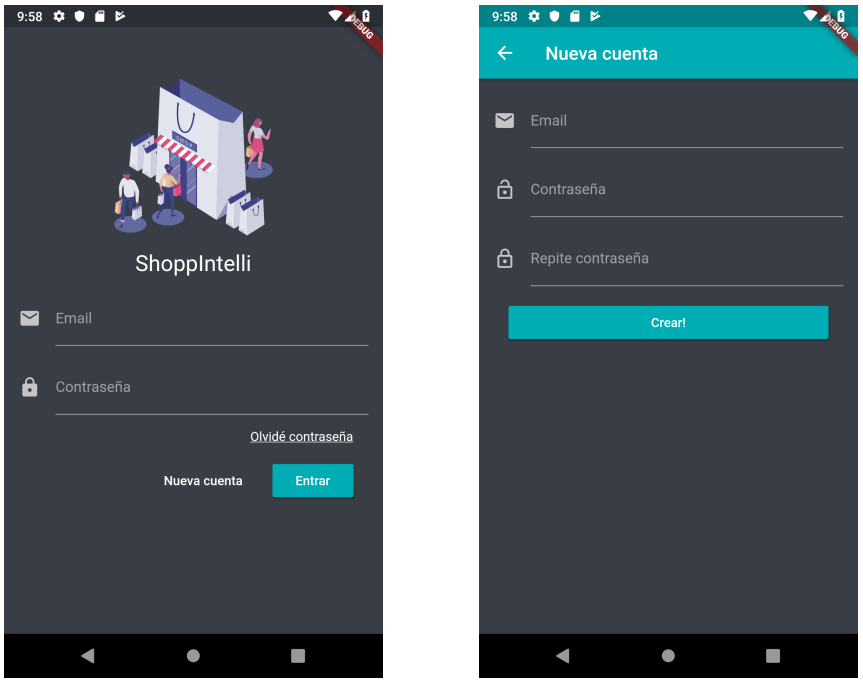


Figura 4.1: Diagrama de Navegación.

Como se puede observar la estructura de la aplicación es muy sencilla, esto es así para seguir el objetivo de realizar las tareas de manera rápida y sencilla, consiguiendo así una sensación de simplicidad y sencillez a lo largo de la aplicación. Ahora pasaremos a explicar un poco más en detalle cada una de las pantallas de la aplicación.

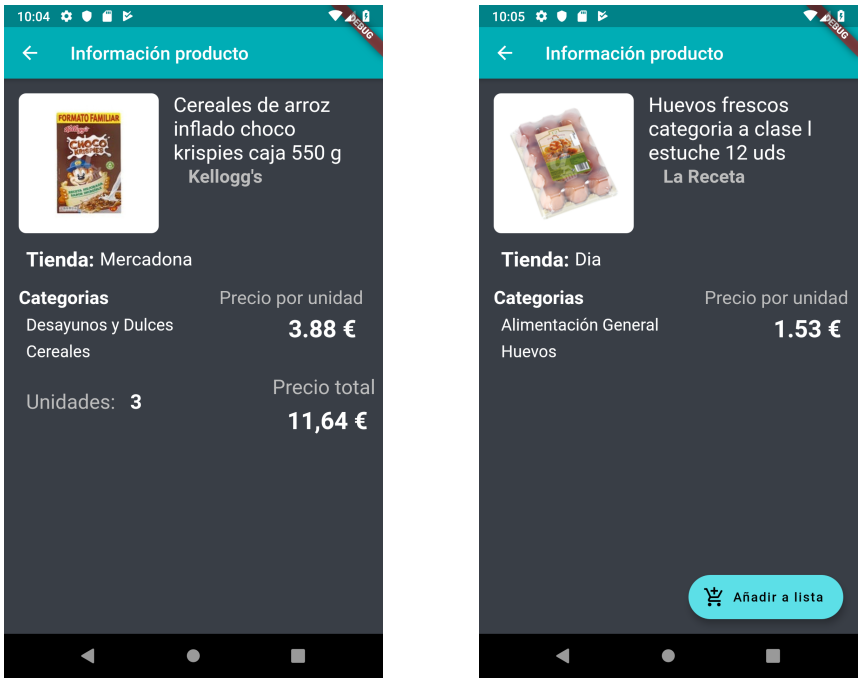
En primer lugar, al abrir la aplicación se mostrará una pantalla de inicio de sesión, donde el usuario podrá introducir sus datos de usuario o en caso de no estar registrado podrá ir a la pantalla de registro de usuario. Una vez registrado se enviará un correo de confirmación de email y podrá iniciar sesión. La pantalla de inicio se muestra en la Figura 4.2(a) y la pantalla de registro en la Figura 4.2(b).

En cualquier parte de la aplicación, al pulsar encima de un producto, se mostrarán los detalles del mismo, donde se podrá ver una imagen más grande y el título completo del producto, también se mostrarán las categorías a las que pertenece así como la marca y precio, en caso de haber pulsado un producto que pertenece a la lista de la compra del usuario, se le mostrará con más detalle el número de unidades añadidas a la lista así como el precio total de esas unidades, esta pantalla se muestra en la Figura 4.3(a). En caso de haber pulsado en los detalles de un producto desde la pantalla de añadir/buscar producto se mostrará un botón que permitirá al usuario añadir unidades de dicho producto, esta pantalla se muestra en la Figura 4.3(b).



(a) Inicio de sesión. (b) registro.

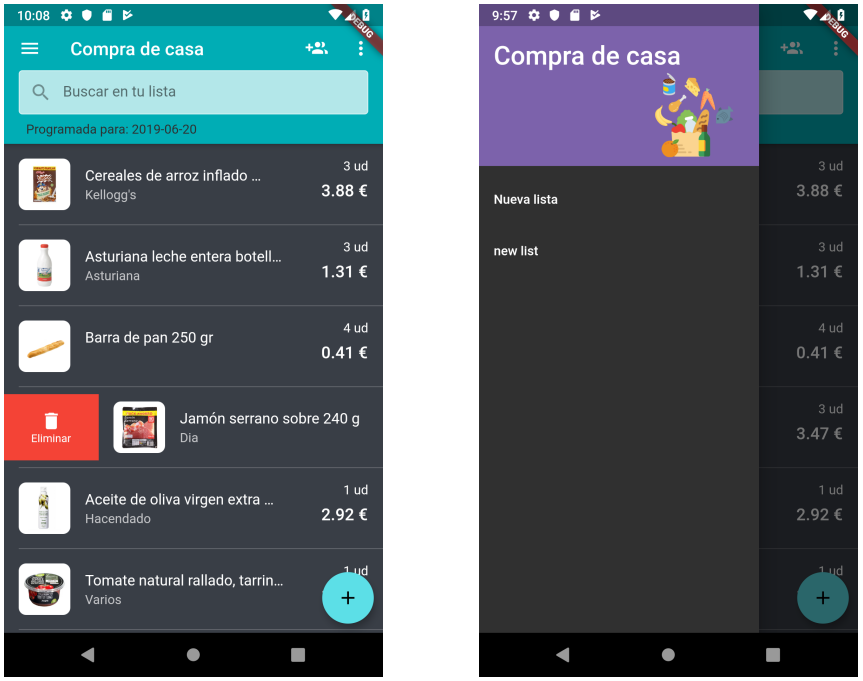
Figura 4.2: Capturas de la aplicación el inicio de sesión y el registro de usuario



(a) Detalles de productos en una lista. (b) Detalles de un producto al añadir

Figura 4.3: Capturas de la aplicación en la página de detalles de producto

Una vez iniciada sesión, la aplicación recordará sus datos y siempre que abra la aplicación se le mostrará directamente la página principal sin necesidad de iniciar sesión cada vez, también existirá la posibilidad de cerrar sesión desde la página principal. Desde la página principal de la aplicación se podrán realizar la gran mayoría de las acciones de la aplicación. En ella se mostrará la última lista de la compra que seleccionó junto con todos los productos que se añadieron a la misma, existirá una barra de búsqueda que le permitirá buscar dentro de los productos de la propia lista. Arriba a la derecha existirá un botón para entrar a la configuración de la lista, además del botón de invitación de usuarios y arriba a la izquierda un botón que desplegará todas las listas de la compra a las que pertenece, al hacer clic en él se podrá cambiar de lista de la compra. Si el usuario lo desea podrá ver los detalles de un producto haciendo clic en cualquier producto de su lista de la compra. Por último, abajo a la derecha existirá un botón que permitirá o bien añadir/buscar nuevos productos para su lista de la compra o bien crear una nueva lista de la compra. Para eliminar un producto de la lista se deslizará el producto hacia la derecha. En la Figura 4.4(a) se muestra una captura de la página principal con productos añadidos. En la Figura 4.4(b) se muestra una captura de cómo se aprecia el desplegable con una lista de las listas de la compra del usuario.



(a) Página principal. (b) Desplegable de listas de la compra.

Figura 4.4: Capturas de la aplicación en la página principal y en el desplegable de listas de la compra.

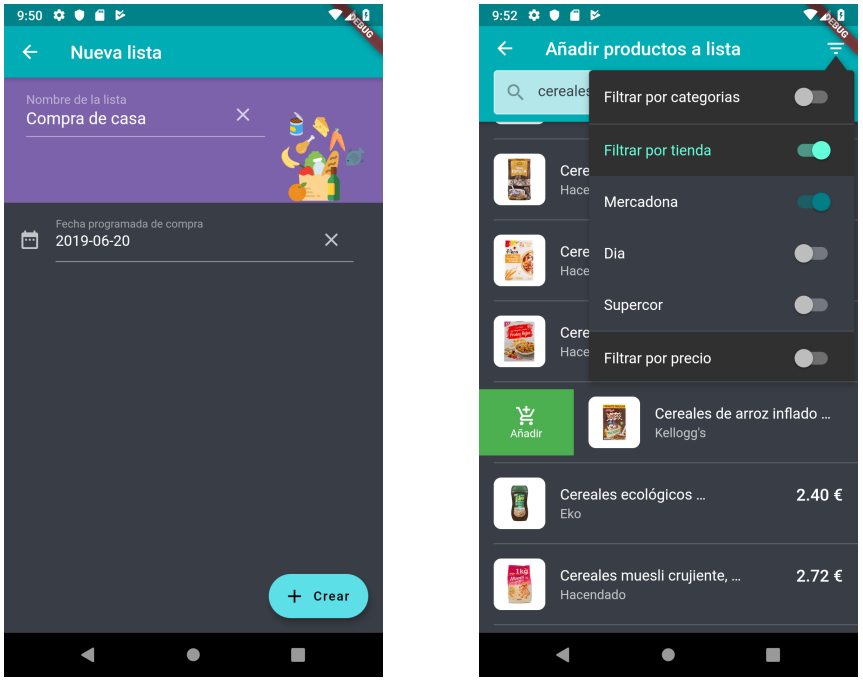
En la pantalla de configuración de la lista se podrá cambiar el nombre de la lista, así como ver y eliminar participantes de la misma. También se podrá eliminar la lista de la compra. Esta pantalla se puede ver en la Figura 4.5.

Figura 4.5: Captura de la pantalla de la lista de la compra.

En la pantalla de añadir/buscar productos se podrá realizar una búsqueda por nombre de producto, además en la parte de arriba a la derecha existirá un menú desplegable que permitirá al usuario filtrar los productos con diferentes filtros: filtro por categorías y subcategorías, filtrar por tiendas, y filtrar por precio máximo, todos estos filtros se podrán aplicar a la vez resultando así en búsquedas más precisas y concisas, este desplegable se puede ver en la Figura 4.6(b). Al final de la barra de búsqueda existirá un botón con forma de bombilla que generará recomendaciones personalizadas al usuario de posibles productos para añadir a la lista. Para añadir unidades de un producto, el usuario podrá o bien ir a su página de detalles y añadirlas desde ahí haciendo clic en el botón de abajo a la derecha como se muestra en la Figura 4.3(b) o bien deslizando el producto hacia la derecha y pulsando en añadir como se muestra en la Figura 4.6(a).

Hay que mencionar que, para las recomendaciones personalizadas del usuario, solamente se han implementado los métodos más sencillos (basados en popularidad, historial y aleatorios) que se comentan en la sección 2.2, en vez de incluir los algoritmos más complejos basados en árboles de decisión, a priori y aprendizaje automático. Esto es debido a la falta de grandes cantidades interacciones de usuarios reales, las cuales son muy necesarias para poder hacer uso de estos algoritmos más complejos; por ello, dejamos para el futuro (cuando existan más datos reales en el sistema) la implementación de estos algoritmos más complejos.

Por último, comentar que la tarea de conseguir una búsqueda simple, intuitiva y a la vez efectiva ha resultado una tarea más difícil de lo esperado, debido a la manera en que Firebase estructura la



(a) Crear lista de la compra.

(b) Buscar/Añadir productos.

Figura 4.6: Capturas de la aplicación en las páginas de crear lista nueva y Buscar y añadir productos

información en la base de datos, y se ha intentado resolver de la mejor manera posible.

Se revisó la posibilidad de proporcionar búsquedas basadas en índices gracias a servicios de terceros, como por ejemplo Algolia [17] y Elastic Search [18], pero por falta de tiempo y complejidad de la tarea, se decidió tomar un enfoque diferente y más sencillo. De esta forma, las búsquedas y filtros se realizan localmente, sin la necesidad de realizar peticiones al servidor, aumentando la eficiencia de las búsquedas considerablemente. La idea general es que al inicio de la aplicación (en la pantalla de carga) se precargan todos los productos disponibles en memoria, para que a la hora de buscar se puedan utilizar métodos de filtrado más sencillos. La búsqueda por nombre se realiza comprobando que cada término de la búsqueda se encuentra en el nombre del producto.

5

Integración, pruebas y resultados

En este capítulo pasaremos a explicar las pruebas que se han ido realizando a lo largo del proceso de desarrollo y los resultados obtenidos de los diferentes tipos de pruebas, además de los resultados obtenidos de un cuestionario de usabilidad realizado por diferentes tipos de usuarios que probaron la aplicación. Dividiremos las pruebas en distintos tipos: Pruebas unitarias, pruebas de componentes, pruebas de integración y sistema, pruebas de rendimiento y compatibilidad, pruebas de interfaz gráfica y por último pruebas y cuestionario de usabilidad.

5.1. Pruebas unitarias

Las pruebas unitarias se fueron realizando a largo del proceso de desarrollo. Cada vez que se terminaba de desarrollar una cierta acción de la aplicación, se procedía a realizar una prueba de caja blanca gracias al depurador que proporciona Android Studio, para comprobar que todo el código desarrollado funcionase como se esperaba.

La lógica del código responsable de las acciones más importantes de la aplicación fue sometida a estas pruebas, como podrían ser: la creación y eliminación de listas de la compra, la acción de añadir o eliminar unidades de un producto de una lista de la compra, la búsqueda por nombre de productos, el correcto funcionamiento de los filtros y su combinación con la búsqueda y demás funciones importantes.

El hecho de realizar esta tarea junto con el proceso de desarrollo de código de la aplicación permitió asegurar que todas las unidades de código de la aplicación funcionasen según lo esperado, facilitando así la tarea, y proporcionando estabilidad a las siguientes unidades de código que se fuesen añadiendo.

5.2. Pruebas de componentes

Los diferentes componentes de la aplicación se fueron sometiendo a pruebas de caja negra para comprobar su correcto funcionamiento además de asegurar que los componentes que tuviesen algún tipo de relación funcionaran de forma conjunta y sin errores. Para ello se fueron probando diferentes casos de error para comprobar si la aplicación avisase de dicho error, y casos válidos para ver si la aplicación respondiese según lo esperado.

Al igual que las pruebas unitarias, las pruebas de componentes se fueron realizando sobre la marcha, asegurando una mayor estabilidad y seguridad sobre los siguientes componentes que se fuesen desarrollando.

A continuación pasaremos a explicar algunos ejemplos de pruebas sobre componentes donde pueden aparecer los errores más comunes, mostrando el posible flujo de trabajo del resto de pruebas realizadas a otros componentes.

Empezaremos mostrando algunos de los primeros errores que podrían ocurrir dentro de nuestra aplicación. Como por ejemplo a la hora de registrarse, que el usuario no introduzca toda la información solicitada o bien introduzca datos erróneos o no válidos. Como se muestra en la Figura 5.1, la aplicación muestra un mensaje de error entendible y visible al usuario.

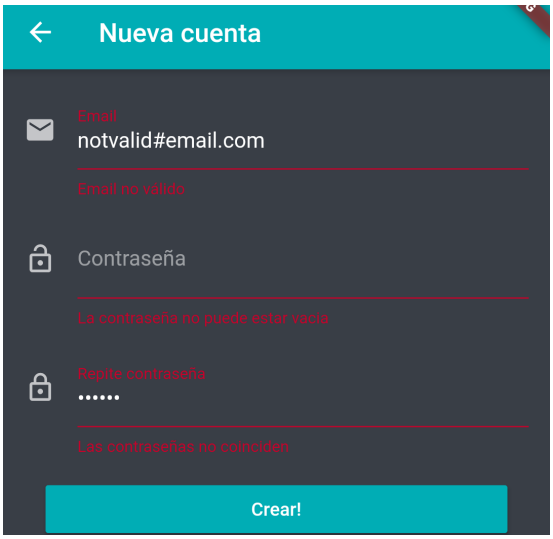
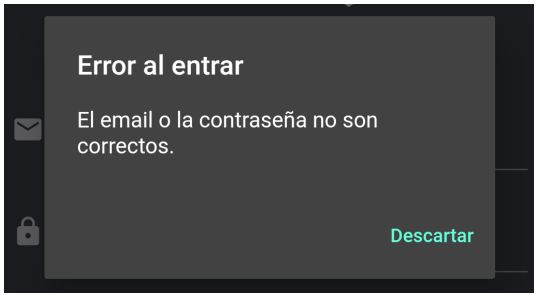


Figura 5.1: Error al registrarse.

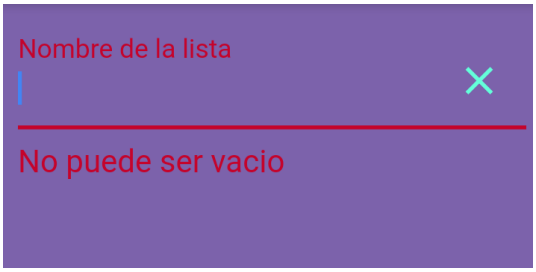
En la Figura 5.2(a), se puede observar el error que se muestra si un usuario intenta iniciar sesión con datos incorrectos. O por ejemplo intentar crear una lista de la compra sin nombre, como se muestra en la Figura 5.2(b), incluso, para que los nombres de las listas no sean muy largos y quepan en la pantalla se ha limitado el máximo de caracteres a 17.

Un error común que podría ocurrir a la hora de utilizar la aplicación es que el usuario intentase añadir productos sin tener una lista previamente creada; como se puede comprobar en la Figura 5.3,

se le mostrará un mensaje indicando que primero será necesario crear una lista de la compra.



(a) Error al iniciar sesión.



(b) Error al crear lista de la compra.

Figura 5.2: Capturas de los errores al iniciar sesión y al crear lista.

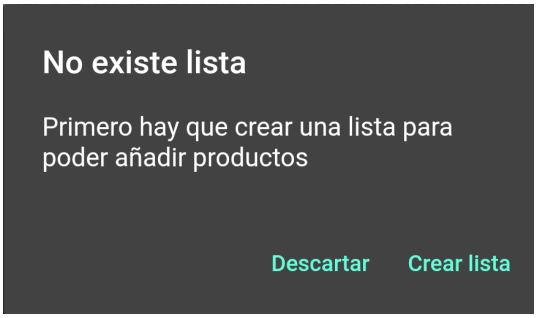


Figura 5.3: Error al añadir un producto.

Mencionar nuevamente que no se han podido realizar pruebas exhaustivas del componente de recomendaciones personalizadas, por una falta de gran cantidad de interacciones de usuarios reales. Aún así se han probado los métodos más sencillos que se tenían implementados e integrados y se ha comprobado su correcto funcionamiento.

Para las pruebas del componente de búsqueda se fueron comprobando los resultados obtenidos de búsquedas variadas, en las que se fueron utilizando tanto filtros por separado como búsquedas con combinaciones de filtros y búsquedas por nombre, confirmando que los resultados obtenidos se asemejasen a lo esperado. Como hemos hablado en el capítulo anterior, la funcionalidad de la búsqueda es un poco menos intuitiva y eficiente de lo que se pretendía, aunque por un lado ha hecho que la búsqueda sea más rápida, por otro lado, ha resultado en un mayor uso de memoria. Además, es apreciable que el mayor inconveniente de esta búsqueda es a la hora de realizar búsquedas por nombre, ya que puede llegar a mostrar resultados no deseados para ciertas situaciones.

5.3. Pruebas de integración y sistema

Una vez pasadas las pruebas unitarias y de componentes, pasamos a realizar pruebas sobre el funcionamiento de los componentes y las relaciones que existen entre ellos, es decir, el funcionamiento

completo del sistema. Estas pruebas se realizaron al final de la fase de desarrollo, ya que requería que la aplicación estuviese terminada.

Algunos ejemplos de pruebas a las que se sometió la aplicación fueron las siguientes:

- Se probó a crear listas de la compra y a añadir y eliminar productos de las mismas, comprobando así que los productos apareciesen en la lista y que la base de datos se actualizase correctamente.
- Se probó a cambiarle el nombre a una lista de la compra y que se reflejase en la aplicación y en el servidor.
- Se probó añadir y eliminar participantes de una lista de la compra, comprobando si cada uno de los usuarios tenían acceso a la lista de la compra desde las diferentes cuentas.
- Además, se probó a intentar añadir productos iguales y diferentes en instantes de tiempo muy cercanos para observar qué efecto tendría en la lista de la compra compartida por los diferentes usuarios.

Todas estas y otras pruebas realizadas obtuvieron los resultados esperados además de un correcto funcionamiento.

5.4. Pruebas de rendimiento y compatibilidad

Para asegurar una aplicación de calidad y disponibilidad máxima, se realizaron pruebas de rendimiento y compatibilidad en diferentes dispositivos, diferentes plataformas, incluso en máquinas virtuales y dispositivos reales, para obtener la mayor cantidad y diversidad de resultados, pudiendo así realizar comparaciones.

Para las pruebas de rendimiento se utilizaron las herramientas disponibles en Android Studio, que permiten ver cuáles son los métodos y clases que más recursos demandan, además de poder ver en todo momento el estado de la memoria y los frames por segundo de la interfaz gráfica.

Una vez obtenidos los resultados de las pruebas de rendimiento, se ha podido observar que la funcionalidad que más recursos demanda es la parte de búsqueda y filtros. También se ha comprobado que en las máquinas virtuales de Nexus 5X y iPhone X la aplicación funciona muy fluidamente y sin mayores problemas. Sin embargo, al realizar pruebas sobre dispositivos reales, se ha observado que los dispositivos Android responden mejor que los dispositivos iOS. Más concretamente en los dispositivos Android la búsqueda con múltiples hilos tarda en torno a los 10 segundos para búsquedas por nombre, mientras que en iOS puede llegar a tardar casi 1 minuto, es por ello, que, tras realizar diferentes pruebas, se ha decidido volver a versiones anteriores de la aplicación sin multi-threading para los dispositivos iOS ya que muestran mejores resultados.

Para asegurar que la aplicación es compatible con la gran variedad de dispositivos disponibles hoy en día, se han realizado pruebas en las que se instala y ejecuta en dispositivos para las dos plataformas Android y iOS, e incluso para diferentes versiones de sistema operativo.

Se ha comprobado que no existen problemas mayores problemas de compatibilidad con dispo-

sitivos Android. Sin embargo, para los dispositivos iOS se han encontrado más problemas de compatibilidad debido a las restricciones que presentan los dispositivos iOS respecto a la instalación de aplicaciones externas a la App Store. Como consecuencia de estas restricciones no se ha podido incluir la funcionalidad de los Dynamic Links (invitaciones a listas de la compra) para los dispositivos iOS, debido a que se requiere de una cuenta de desarrollador iOS de la cual no disponemos. Por lo demás, la aplicación funciona correctamente, y usuarios de diferentes plataformas que pertenecen a la misma lista de la compra puede utilizar la aplicación sin ningún problema.

5.5. Pruebas de interfaz gráfica

Al ser una aplicación móvil es muy importante que la interfaz gráfica funcione correctamente y se adapte a todos los tipos de dispositivos que existen en el mercado. Por ello se realizaron pruebas que comprobasen la adaptación de la aplicación en los posibles entornos, plataformas y tamaños de dispositivo diferentes, asegurando una experiencia de buena calidad en todos los dispositivos diferentes.

Para la realización de las pruebas se utilizaron dispositivos tanto reales como virtuales, de diferentes tamaños y resoluciones, así como para ambas plataformas Android y iOS. Comprobando en cada caso si la interfaz gráfica se mostraba como se preveía y si la experiencia seguía siendo de calidad.

Algunos de los dispositivos utilizados para la realización de las pruebas fueron: Nexus 5X, Samsung Galaxy S8 y S6, Samsung Galaxy Grand Prime, Pixel XL, iPhone 6 y 6s, iPhone X y máquinas virtuales de 2.7” QVGA (240x320), Galaxy Nexus (720x1280).

5.6. Pruebas y cuestionario de usabilidad

La realización de estas pruebas fue posible gracias a la participación de algunos usuarios objetivo de la aplicación. Se intentó que el conjunto de usuarios fuese variado, para poder obtener valoraciones de la aplicación desde puntos de vista lo más heterogéneos posible.

Los usuarios que participaron en la realización de las pruebas fueron los que se muestran en la Tabla 5.1. Como se puede observar, son de diferentes rangos de edad además de tener móviles de ambas plataformas.

En la Tabla 5.2 se muestran los resultados de la realización de las acciones, se puede observar que para las tareas sencillas los usuarios tienden a tardar poco, alrededor de los 20 o 30 segundos. Pero, sin embargo, para las acciones que hacen uso de la funcionalidad de búsqueda, los tiempos aumentan considerablemente hasta entrar en los minutos, esto es debido, en parte, a que la tarea de buscar productos es más compleja que el resto, pero por otro lado también puede ser debido a que es menos intuitiva o eficiente que el resto de las acciones. También se puede observar que, para los

	Edad	Móvil
Usuario 1	19	iPhone 6
Usuario 2	15	iPhone 6
Usuario 3	54	Samsung Galaxy S6
Usuario 4	53	Samsung Galaxy S8

Tabla 5.1: Tabla que refleja la edad y telefono móvil de los usuarios

usuarios con móviles iOS, la tarea de hacer una búsqueda es mucho más lenta, esto puede ser por que la funcionalidad de búsqueda es más lenta en los móviles iOS antiguos. Los tiempos de borrar/elimnar unidades de un producto no se asemejan a la facilidad de la tarea, esto puede ser porque no es intuitivo la forma de hacerlo.

	Tiempo en realizar acción			
	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Registrarse en la aplicación	35s	1min	1min 30s	47s
Iniciar sesión en la aplicación	17s	20s	1min 17s	38s
Crear una lista de la compra	18s	35s	49s	30s
Añadir un producto a la lista de la compra	1min 20s	2min 10s	1min 25s	54s
Pedirles buscar un producto usando filtros	1min 20s	2min 15	44s	56s
Cambiar el nombre de la lista de la compra	18s	25s	20s	21s
Cambiar de lista de la compra	5s	20s	22s	19s
Invitar a un amigo a la lista de la compra	NP	NP	50s	37s
Eliminar a un usuario de la lista de la compra	10s	18s	32s	25s
Eliminar unidades/productos de la lista de la compra	1min	55s	40s	35s

Tabla 5.2: Tiempos que tardaron los usuarios en realizar las tareas propuestas

A continuación, para obtener valores cuantitativos de su experiencia, se les pidió responder un cuestionario sencillo, en el cual los usuarios fueron valorando las actividades y preguntas propuestas, con valores desde 1 a 5, siendo 1 muy en desacuerdo o muy negativo y 5 muy de acuerdo o muy positivo. Al final de la prueba también se les pidió realizar una valoración global de la aplicación, así como dar un pequeño feedback sobre la experiencia y los posibles puntos a mejorar de la aplicación.

En la Tabla 5.3 se muestran los resultados obtenidos del cuestionario tras haber realizado las tareas. Se puede comprobar que, en general, los resultados son bastante satisfactorios. También se puede observar que para los usuarios iOS la aplicación se nota más lenta.

	Valoración			
	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Considero que utilizaría esta aplicación	4	2	5	3
Encontré el sistema innecesariamente complicado	1	3	2	1
Creo que necesitaría soporte de un técnico para poder usar la aplicación	1	1	1	1
Encontré que las funciones de la aplicación estaban bien integradas	4	5	5	5
Considero que había demasiada inconsistencia en la aplicación	1	1	2	1
Considero que la aplicación es fácil de utilizar	5	3	5	4
Creo que se aprende rápido a utilizar la aplicación	5	4	5	5
Necesito aprender muchas cosas antes de utilizar el sistema	1	1	1	1
En general opino que la aplicación es buena	4	4	5	4
La aplicación le avisa cuando hay un error	NP	NP	NP	5
La aplicación es rápida	3	1	5	5
La aplicación es intuitiva	5	3	4	4
Los filtros son intuitivos de utilizar y de aplicar	4	4	3	3
Considero que los filtros son útiles de utilizar	3	3	5	4

Tabla 5.3: Resultados del cuestionario de usabilidad

6

Conclusiones y trabajo futuro

Para finalizar este trabajo de fin de grado pasaremos a exponer las conclusiones que se han obtenido de su realización, además de analizar y explicar el posible trabajo futuro que surge de la realización de este proyecto.

6.1. Conclusiones

Tras la realización de este trabajo podemos llegar a la conclusión de que se han podido cumplir los objetivos principales que se tenían en mente al inicio del mismo. Se ha conseguido desarrollar una aplicación funcional, bonita y desde cero, todo ello sin tener ningún conocimiento sobre desarrollo de aplicaciones móviles ni experiencia con el SDK de Google (Flutter), resultando en una experiencia muy enriquecedora y de gran adquisición de conocimientos sobre el mundo de desarrollo móvil.

La idea principal de la aplicación estuvo muy clara desde el principio, pero también resultaba evidente que era muy ambiciosa, es por ello que se analizaron los objetivos inicialmente considerados y finalmente se decidió tomar una trayectoria más viable, en la cual existiese la posibilidad de incluir nuevos objetivos o de descartar objetivos anteriormente seleccionados, permitiendo así tener un proyecto más flexible y que se fuese adaptando a las posibilidades.

Por la falta de conocimientos en el ámbito de las aplicaciones móviles y sobre el SDK de Flutter, en primera estancia, no se tenía una trayectoria demasiado clara, además de que no se sabía exactamente cómo se iban a alcanzar los objetivos más importantes de la aplicación, pero tras la fase de análisis se empezó a vislumbrar una trayectoria más clara, donde los objetivos más importantes ya estaban fijados, pero también dejando la posibilidad de ser sustituidos por otros nuevos. A lo largo de la fase de desarrollo se fueron adquiriendo algunos conocimientos y cierta experiencia con Flutter y los objetivos empezaban a verse más claros y lentamente se iban convirtiendo en funcionalidad de la aplicación.

Aunque Flutter facilite la tarea de crear aplicaciones para Android y iOS, se ha podido comprobar la dificultad que tiene desarrollar aplicaciones para ambas plataformas y los problemas que pueden surgir de trabajar con los dos sistemas operativos.

Sin duda ha resultado un gran reto personal el conseguir desarrollar una aplicación móvil desde cero, además de ser un proyecto que siempre he querido abordar.

En general estoy muy contento con el producto final que he conseguido, teniendo en cuenta el punto de partida, en el que se tenían conocimientos nulos sobre el desarrollo de aplicaciones, y que la idea principal era crear un primer prototipo para la aplicación. Además, quiero resaltar la especial atención y dedicación en los detalles de la aplicación e interfaz gráfica, que tienen como objetivo asegurar que el uso de la aplicación resulte en una experiencia agradable e intuitiva.

En el siguiente enlace, se puede acceder a una página donde se describe la aplicación y desde la cual se puede descargar: `http://bit.ly/ShopIntelli`. En estos momentos sólo está disponible para la versión Android debido a que aún no se dispone de una cuenta de desarrollador para iOS.

6.2. Trabajo futuro

Como se comenta en el apartado anterior, existen algunos aspectos de la aplicación que se podrían mejorar, además de muchas ideas que no se han podido añadir al producto final debido a la falta de tiempo o de recursos. A continuación, pasamos a detallar algunas ideas para el trabajo futuro:

- Mejorar la funcionalidad de la búsqueda, por ejemplo a través del uso de índices que permitan realizar búsquedas por nombre con métodos de similitud parecidos a los utilizados en búsqueda web, resultando así en una búsqueda más efectiva e intuitiva para el usuario.
- Permitir al usuario buscar productos por código de barras, facilitando la búsqueda de productos; para esto, convendría encontrar una base de datos de productos con sus códigos de barras asociados, algo que no hemos sido capaces de encontrar durante el desarrollo de este proyecto.
- Dar la posibilidad a los usuarios de poder crear productos que no existan en la base de datos.
- Crear recomendaciones y predicciones personalizadas más inteligentes y completas, haciendo uso de los algoritmos estudiados y mencionados en este trabajo.
- Hacer uso del patrón de diseño BloC (Business Logic Component), que facilita el desarrollo y mejora la escalabilidad y la corrección de errores dentro de la aplicación.
- Añadir la posibilidad de elegir una imagen/icono o colores para las diferentes listas de la compra, facilitando así la clasificación de las mismas.
- Permitir la posibilidad de ordenar las listas de la compra según distintos criterios.
- Ofrecer más opciones de configuración tanto de las listas de la compra como de usuario.
- Crear animaciones o tutoriales para que facilite el aprendizaje del funcionamiento de la aplicación.

Bibliografía

[1] "Resumen – cordova." https://cordova.apache.org/docs/es/latest/guide/overview/, Accedido en Junio 2019.

[2] L. Peris, "Ventajas y desventajas de apache – cordova." https://luisperis.com/apache-cordova/, Accedido en Junio 2019.

[3] C. District, "¿qué es react native?." https://clouddistrict.com/blog-dev/que-es-react-native/, Accedido en Junio 2019.

[4] AssertSoft, "Desventajas de react native." https://www.assertsoft.com/entrada/3-desventajas-apps-con-react-native/, Accedido en Junio 2019.

[5] "Flutter." https://flutter.dev/, Accedido en Junio 2019.

[6] CodeMagic, "Beneficios y limitaciones de flutter." https://blog.codemagic.io/what-is-flutter-benefits-and-limitations/, Accedido en Junio 2019.

[7] C. M. Cumby, A. E. Fano, R. Ghani, and M. Krema, "Predicting customer shopping lists from point-of-sale purchase data," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004* (W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, eds.), pp. 402–409, ACM, 2004.

[8] A. V. C. TEAM, "Mining frequent items bought together using apriori algorithm (with code in r)," *Analitics Vidhya*, Agosto 2017. https://www.analyticsvidhya.com/blog/2017/08/mining-frequent-items-using-apriori-algorithm.

[9] "Deliberry." https://www.deliberry.com/, Accedido en Junio 2019.

[10] "Material design." https://material.io/, Accedido en Junio 2019.

[11] "Firebase." https://firebase.google.com/, Accedido en Junio 2019.

[12] D. Velasquez, "Flutter : Firebase dynamic link," *Medium*, Agosto 2018. https://medium.com/@diegoveloper/flutter-firebase-dynamic-link-6f1b79278ce0.

[13] "Libreria para utilizar arañas con python." https://scrapy.org/, Accedido en Junio 2019.

[14] "Ean search base de datos con productos y códigos de barras." https://www.ean-search.org/, Accedido en Junio 2019.

[15] "Barcode database, base de datos con productos y códigos de barras.." https://barcodesdatabase.org/, Accedido en Junio 2019.

[16] "Get to know cloud firestore." https://www.youtube.com/playlist?list=PLl-K7zZEsYLluG5MCVEzXAQ7ACZBCuZgZ, Accedido en Junio 2019.

[17] Algolia, "Servicio que permite crear indices y realizar búsquedas más eficientemente.." https://www.algolia.com/, Accedido en Junio 2019.

[18] E. Search, "Otro servicio que permite crear indices y realizar búsquedas más eficientemente.." https://www.elastic.co/es/products/elasticsearch, Accedido en Junio 2019.

Definiciones

BloC Es un patrón de diseño que permite estructurar, de forma muy eficiente, la arquitectura de las aplicaciones diseñadas en Flutter.

Cloud Firestore Es la base de datos no relacional de Firebase.

Dart Lenguaje de programación utilizado por Flutter.

Firebase Es una plataforma de desarrollo móvil en la nube creada por Google, que proporciona características como Dynamic Links, base de datos no relacional, gestión de usuarios, etc.

Flutter Flutter es un SDK creado por Google que permite crear aplicaciones móviles, web y de escritorio bonitas y profesionales con un solo código fuente.

Naïve Bayes Se trata de un clasificador probabilístico fundamentado en el teorema de Bayes.

Node Es un código abierto de JavaScript que está diseñado para generar aplicaciones web de forma altamente optimizada.

NPM Node package manager, es un gestor de paquetes, que facilita la tarea de trabajar con Node.

Perceptrón En el campo de las redes neuronales un perceptrón es una unidad básica de inferencia, a partir de la cual se puede desarrollar algoritmos capaces de generar criterios de clasificación.

Winnow Es una técnica de aprendizaje automático que permite realizar clasificaciones lineales a partir de ejemplos etiquetados.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID