Universitat Oberta de Catalunya

uoc.edu

Shoppath:
Lista de la compra en Android

Memoria de Proyecto Final de Máster

Máster universitario en Desarrollo de aplicaciones para dispositivos móviles

Trabajo final de Máster DADM

Autor: José Ángel Bravo Montañez

Consultor: Eduard Martín Lineros

Profesor: Carles Garrigues Olivella

Diciembre de 2020

Esta obra está sujeta a una licencia de
Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Dedicatoria

A mi mujer Laura, por apoyarme y animarme a continuar con mis estudios después de estar estos parados durante 17 años. Primero con un grado en multimedia y después con este Máster en desarrollo de aplicaciones para dispositivos móviles.

A mis hijos Xavi y Leo por tener que aguantar, desde que nacieron, a un padre estudiante, con todo lo que eso conlleva.

FICHA DEL TRABAJO FINAL

Título del trabajo:	ShopPath: Lista de la compra en Android
Nombre del autor:	José Ángel Bravo Montañez
Nombre del consultor/a:	Eduard Martín Lineros
Nombre del PRA:	Carles Garrigues Olivella
Fecha de entrega:	12/2020
Titulación:	Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Área del Trabajo Final:	Trabajo final de Máster DADM
Idioma del trabajo:	Castellano
Palabras clave:	Android, ruta, compra.

Resumen del Trabajo

La finalidad de este proyecto es la creación de una app para Android con la cual gestionar la lista de la compra. Existen muchas aplicaciones de este tipo en el mercado, pero es complicado encontrarlas que cumplan con todas las necesidades, sobretodo el que, a la hora de realizar la compra, los productos sean ordenados de manera automática por cómo se encuentran dentro del supermercado, evitando así al usuario vueltas innecesarias dentro de este.

Se ha programado una nueva aplicación, desde cero, con Kotlin como lenguaje de programación, con Firebase como backend, implementando el patrón de arquitectura MVP, y aprovechando la utilización de librerías de terceros. Durante todo el proceso ha primado el incluir la mayor parte de los conocimientos adquiridos en el curso del Máster del cual este es el trabajo final.

Todo el proceso ha sido planificado y documentado desde el principio, incluyendo las fases de diseño e implementación y pruebas obteniéndose como resultado la app finalizada, su manual de usuario, una presentación visual y esta memoria.
El resultado final ha sido satisfactorio, ya que se han cumplido prácticamente al 100% los objetivos, tanto principales como secundarios, propuestos al inicio del proyecto, obteniéndose una aplicación que responde a las expectativas, pero que tiene potencial para crecer y mejorar en el futuro inmediato.

En conclusión, en el proyecto se han puesto en práctica los conocimientos obtenidos durante el Máster, a la vez que se han cumplido los objetivos, obteniéndose una App funcional, práctica y con posibilidades de crecimiento.

Debido a su futura utilización con fines comerciales, el **código fuente** de la APP Shoppat, resultado de este TFM, **no será publicado** junto a esta memoria.

Abstract

This project purpose is to create an Android's App with which manage shopping lists. There are many applications of this kind in the market, but it's difficult to find some with all the needs, especially that, when making the purchase, the products are automatically ordered by how you find it when shopping, thus avoiding to the user unnecessary laps within the shop.

A new application has been programme, from scratch, with Kotlin as programming language, with Firebase as backend, implementing MVP architecture pattern, and taking advantage of third-party libraries. Throughout the process, it has prevailed to include most of the knowledge acquired in the course of the Master of which this is the final work.

The entire process has been planned and documented from the beginning, including the design, implementation and tests phases, resulting the finished app, its user manual, a visual presentation and this memory.

The final result has been satisfactory, since the objectives, both main and secondary, proposed at the beginning of the project have been practically 100% fulfilled, obtaining an application that meets expectations, but has the potential to grow and improve in the future.

In conclusion, the project has put into practice the knowledge obtained during the Master, while the objectives have been fullfilled, obtaining a functional and practical App with possibilities for growth.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

La utilización de las tecnologías hace tiempo que se ha instaurado en el día a día de las personas, acentuándose esta presencia gracias a los avances en las tecnologías móviles, las mejoras de las redes de comunicación (como puede ser la implantación del 5G), el uso generalizado de los smartphones y sus omnipresentes “APPs”.

Hoy día utilizamos la tecnología en todos los ámbitos de nuestra vida, nos aprovechamos de ella (o la sufrimos) en el trabajo, el ocio, y como no puede ser de otra manera, también en las tareas del día a día. Existen APPs para casi todo lo imaginable, cualquier necesidad que se nos presente es muy probable que ya esté cubierta con una, o varias, APPs de las que podemos disponer fácilmente tanto en la App store de Apple así como en la Google Play de Android, además de otras alternativas..

Por otro lado que haya tal cantidad de “APPs” en el mercado y que cubran una inmensa cantidad de requisitos, no implica necesariamente que todas nuestras necesidades queden satisfechas. Es muy posible que encontremos aplicaciones que podamos utilizar para una tarea, pero que sin embargo les falte una función o una característica que se adaptaría mejor a nuestros gustos, o haría que fuéramos más productivos. Sin embargo la utilizamos porque es la mejor opción que hemos encontrado, o bien es la más conocida, o la que utilizan nuestros amigos.

En lo personal, a la hora de ir a comprar, he probado distintas APPs de lista de la compra, y siempre se da el caso de que esa función que tanto me ayuda en cierta APP, no la encuentro en otra, que por el contrario me ofrece una característica que necesito y no ofrece aquella. Por otro lado, acostumbro a ir apuntando, durante la semana, los productos que veo que voy a necesitar o que se me han acabado. De esta manera, la lista de la compra queda con los productos en un orden que no sigue más lógica que la de cuando me he dado cuenta de que los tengo que comprar, de manera que este orden para nada ayuda a la hora de encontrar los productos en la tienda. Así pues, he de proceder a ordenar los productos de manera manual, intentando acordarme de como voy a circular por la tienda, que pasillos hay antes, cuales después, y cuando me voy a encontrar los productos que necesito, en uno u otro pasillo.

Es por esto que una función que seguro ayudaría mucho, y no encuentro en ninguna APP de las que he probado, es la de poder definir una ruta por las distintas secciones de la tienda en la que hago la compra, evitando así acabar yendo de un pasillo a otro en busca de los productos listados, por estar estos mal ordenados, o tener que ordenarlos manualmente con anterioridad, cosa que la mayoría de veces olvido.

De la realización de este proyecto debe resultar una APP que integre las funciones típicas existentes en la actualidad en APPs de este tipo, como crear artículos nuevos, añadir, eliminar o confirmar productos de la lista, etc. Así como incluir opciones que no se encuentran habitualmente, como es la gestión de la ruta de compra, basándose esta en la ubicación física de las diferentes categoría de la tienda.

En este trabajo se va a implementar gran parte de lo aprendido durante el Máster, principalmente las asignaturas de diseño de aplicaciones y las de desarrollo de aplicaciones para dispositivos Android. El trabajo realizado durante las diferentes PEC (prueba de evaluación continuada) de estas asignaturas servirá de base para el diseño y programación de la app resultante, así como parte de la información contenida en ellas es utilizado en esta memoria.

1.2 Objetivos del Trabajo

1.2.1 Objetivos principales

- Crear, utilizando el IDE de Android Studio, una nueva APP nativa Android para la gestión de la lista de la compra.
- Implementar BBDD on-line, así como persistencia de datos en local.
- Diseñar la app utilizando el patrón de arquitectura MVP.
- Implementar necesidades detectadas durante la fase de diseño.
- Se deben poder gestionar categorías, con su ubicación en tienda. De esta manera, al introducir elementos en la lista de la compra, estos se ordenen automáticamente respecto a la ubicación de su categoría.
- Realizar estudio para detectar los puntos fuertes y débiles de la competencia, con el fin de integrar en la APP las funciones que resulten en la conclusión.

1.2.2 Objetivos secundarios

- Aplicar elementos estudiados durante el Máster: splash screen, dashboard UI, progressbar, icono de la APP, navigation drawer, librerías de imágenes...
- Posibilidad de definir diferentes tiendas, con sus propias categorías.
- Implementar juegos de pruebas para el testeo de la APP.
- Implementar funciones de Firebase (storage, crash reporting, notificaciones)
- Implementar anuncios AdMob.
- Registro de usuario mediante la API de Google.
- Implementar seguridad mediante huella dactilar.
- Compartir listas entre diferentes usuarios.
- Internacionalizar la APP.
- Implementar obtención de datos mediante rest API disponibles (por ejemplo provincias)
- Interfaz dual smartphone/tablet
- Mostrar la lista de tiendas en un mapa.

1.3 Enfoque y método seguido

En este trabajo se programará una aplicación Android nativa desde cero, utilizando el lenguaje de programación Kotlin. Las funciones a implementar se definirán a partir de las conclusiones del estudio del mercado actual de APPs de "lista de la compra".

Cabría la posibilidad que la aplicación fuera web, o incluso híbrida, cada una con sus ventajas e inconvenientes, pero en esta ocasión la opción elegida es la aplicación nativa, no tanto por el acceso directo al hardware del dispositivo sin necesidad de intermediarios (como si necesitan las híbridas), como a las mejoras de rendimiento(1) y la oportunidad de profundizar en lo que ha sido la base de estudio del Máster.

En cuanto al lenguaje de programación, se ha elegido Kotlin debido a que ofrece mejoras, respecto a su predecesor Java, que facilitan la programación, mejorando la productividad. Además se pretende seguir con la línea del Máster, en la que se ha utilizado este lenguaje en las asignaturas de programación.

Será necesaria la de revisión de diferentes APPs, tanto publicadas en la Google play como en otros medios, con el fin de detectar las funcionalidades más comunes en las APPs de este tipo, así como sus puntos fuertes y débiles, con el fin de implementar de una forma u otra las fortalezas detectadas, así como posibles mejoras que se detecten durante el proceso.

El grueso de la APP se confeccionará desde cero, en parte porque no hay disposición de código público que tenga funcionalidad similar para poder ser modificado y reutilizado en la confección de esta APP. Sin embargo si se utilizarán bibliotecas de código, tanto de Google como de terceros, útiles para la implementación de las diferentes funcionalidades, así como para facilitar tareas, por ejemplo: acceso a bases de datos, tratamiento de las imágenes, etc.(2)(3) Por otro lado, el esqueleto de la app estará basado en un patrón de arquitectura MVP, ya que es el esquema más adecuado al tamaño de este proyecto. Utilizar una alternativa como MVMM añadiría una complejidad innecesaria a una app pequeña como es esta, y por contra, MVC, que si se adaptaría al tamaño, presenta problemas a la hora del testeo, ya que no es una arquitectura adecuada para ello en Android.(4)

1.3.1 Librerías de terceros(2)(3)

En general, es conveniente el uso de librerías, ya que ofrecen una serie de ventajas que sería muy difícil y costoso obtener con código programado por nosotros mismos, además, para que reinventar la rueda, cuando se pueden aprovechar todas las herramientas a nuestra disposición. Aún así, esto no significa que siempre sea la opción más conveniente, ya que hay que tener en cuenta una serie de consideraciones.

Librerías	
Algunas librerías interesantes para este proyecto son:	
Retrofit	Seguramente la librería más popular para realizar peticiones API REST - https://square.github.io/retrofit/
Glide	Para gestión de imágenes: transformar, redimensionar, soporta GIF... - https://github.com/bumptech/glide
Picasso	Simplifica la carga de imágenes, a veces con una sola línea de códigos - http//square.github.io/picasso/
Rxjava	Permite desarrollar componentes reactivos. - https://github.com/ReactiveX/Rx-
Otras conocidas librerías de gran utilidad son:	
Leackcanary	Para detectar fugas de memoria. - https://github.com/square/leakcanary
Dbflow	De los mejores ORM(mapeo objeto-relacional), simpley potente. - https://github.com/agrosner/DBFlow
Butterknife	Para reducir el boilerplate en el codigo, evita llamar a “findViewById”. - http://jakewharton.github.io/butterknife/
Logger	Ayuda a tener un registro de logs más práctico que el por defecto. - https://github.com/orhanobut/logger
Retrolambda	Permite utilizar expresiones lambda en java 7,6 y 5. - https://github.com/evant/gradle-retrolambda
Dagger2	La librería más conocida para inyección de dependencias. - https://github.com/Google/dagger
Jitpack.io	Gestión de proyectos Github - https://jitpack.io/
ZXing	Gestión de códigos de barras - https://github.com/zxing/zxing

Tabla 1: Librerías de terceros

1.3.1.1 Problemas asociados

- **Dependencia**: utilizar una librería de tercero es atar nuestro código a esa librería. A la vez, si es necesario cambiar librerías, nuestro código puede sufrir grandes cambios para adaptarla.
- **Posible falta de soporte**: Puede pasar que la librería sea abandonada por su autor, lo cual es un gran problema. Es necesario un continuo mantenimiento para que el comportamiento de nuestra app sea el óptimo.
- **Sobreuso**: utilizar demasiadas librerías puede causar problemas, como conflictos de dependencia, aumento del tamaño de la app, etc.
- **Problemas de seguridad**: puede haber ciertas vulnerabilidades no controladas por el autor, que sean utilizadas por hackers.

1.3.1.2 Criterios a la hora de escoger

Cada librería que queramos utilizar debe ser evaluada por sus propios méritos (o falta de ellos),teniendo en cuenta una serie de criterios, como puede ser la popularidad de la librería (en github, por ejemplo), si muchos desarrolladores confían y puntúan positivamente la librería, es un buen indicador de su calidad. Otro indicador de la popularidad puede ser el número de entradas en Stackoverflow, con lo que veremos como de grande es la comunidad implicada en esa librería, y como puede ser de fácil solucionar los problemas que aparezcan.

En la línea con la popularidad, también es importante la confiabilidad de su autor, comprobarla puede evitar que quedemos estancados en una librería abandonada. Echar un vistazo a si es activo en Github, si tiene varias librerias publicadas, si responde a los problemas y errores además de si acepta peticiones.

Otro criterio a tener en cuenta es como de bien escrita está la librería, si tiene documentación, un Readme comprensible, etc.

Un criterio a tener muy en cuenta es si la librería cumple con nuestros requerimientos específicos, evitando así tener que, eventualmente, modificarla para que acabe cumpliéndolos. Por el contrario, una librería que tiene demasiadas opciones que no necesitamos es posible que no nos encaje, siendo mejor desestimarla.

También es importante tener en cuenta la licencia y sus posibles limitaciones de uso. Además, como se ha dicho antes, es conveniente poder comprobar el código, con lo que debería ser open source con tal de poder explorarlo y estudiarlo.

Por último, y no menos importante, es que la librería sea recomendada, aparte de las archiconocidas, pueden aparecer nuevas de muy buena calidad y utilidad.

1.3.2 Patrones de arquitectura(4)(5)(6)(7)(8)

"La idea detrás de los patrones de arquitectura de aplicaciones [...] es que todos existen para ayudarnos a diseñar nuestra aplicación de tal manera que permita que esta sea mantenible a medida que esta crece. Dos conceptos, en particular, son útiles: separación de intereses y pruebas unitarias[...]"(4 Yung Cheng, pág 19)

La separación de responsabilidades proporciona la posibilidad de modificar, por ejemplo, la visual de una APP, sin tener que tocar el código restante, como el tratamiento de datos. Se trata de programar en módulos separados, que cada uno tenga una función concreta: mostrar la visual en pantalla, obtención de datos, lógica de negocio, etc. Esta forma de trabajar proporciona las ventajas de poder hacer crecer la aplicación, modificando tan solo la parte estrictamente necesaria, sin tener que tocar las demás, y que todo siga funcionando. Además, se facilita la posibilidad de testear el código, que por el contrario, de no utilizar una arquitectura adecuada, resultaría prácticamente imposible de conseguir, debido a la imposibilidad de los motores de testeo de acceder de manera correcta a las partes diferenciadas de la app, más si cabe con la estructura 'especial' de Android y sus activities.

Para la separación de responsabilidades se hace necesaria la programación por capas, un modelo de desarrollo de sofware para el desacoplamiento de las partes que componen un sistema de software o una arquitectura cliente-servidor. Estas capas son:

-Lógica de negocios o capa de negocio: recibe las peticiones del usuario y se envian las respuestas tras el proceso. Se establecen las reglas a cumplir, comunica la capa de presentación con la capa de datos.

-Capa de presentación o de usuario: es la que el usuario ve, presenta los datos al usuario, comunica y captura la información de E/S con el usuario. También es llamada interfaz de usuario.

-Capa de datos: encargada de acceder a los datos, recibe solicitudes de almacenamiento y recuperación de información desde la capa de negocio.

Aplicando esta programación por capas, resulta sencillo crear varias interfaces para un mismo sistema, sin tener que cambiar las capas de lógica o datos. La principal ventaja es que el desarrollo puede llevarse a cabo a varios niveles, de manera que cuando haya cambios, estos solo afectarán al nivel en cuestión, sin afectar al resto de módulos. También proporciona la posibilidad de trabajar en grupos abstraídos, cada uno de los cuales tan solo necesita saber la API existente entre niveles.

Existen diferentes tipos de arquitecturas, desde la más extendida MVC (modelo vista controlador), evoluciones de esta como MVP (modelo vista presentador) , MVI (modelo vista intent) y otros como MVVM (modelo vista modelo de vista), Viper, etc.

En la siguiente tabla podemos observar algunas de las ventajas y características de los patrones de arquitectura más comúnmente utilizados.

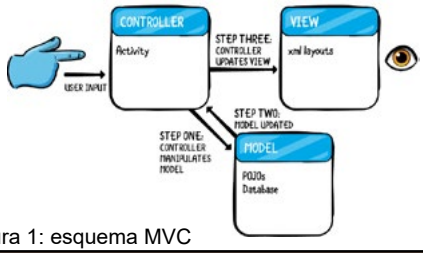
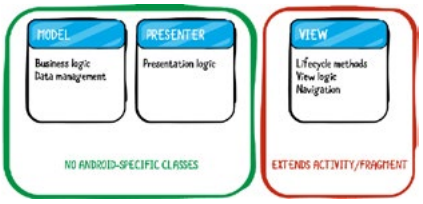
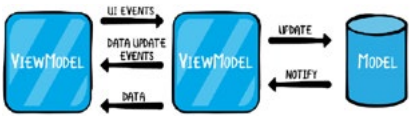
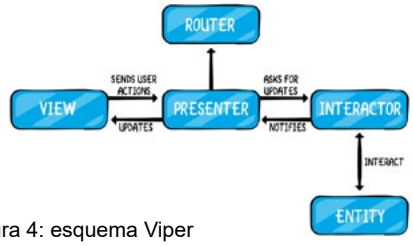
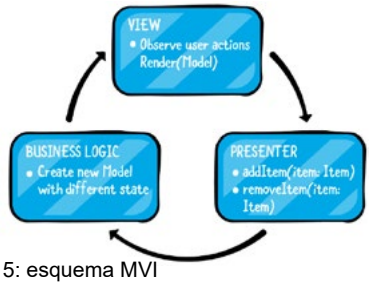
Patrones de arquitectura	
MVC Model-View-Controller	
Presenta problemas con Android, debido a que en este los activity engloban tanto vista como controlador, dificultando o imposibilitando el testeo.	 Figura 1: esquema MVC
MVP Model-View-Presenter	
Con este patrón de arquitectura se evitan los problemas de testabilidad (entre otros) del modelo MVC en Android, ya que separa el Controller fuera del Activity. Al dividir una Activity en interfaces separadas de las clases Model, View y Presenter, permite separación de preocupaciones así como Model y Presenter a los que se puede realizar test unitarios.	 Figura 2: esquema MVP
MVVM Model-Viev-ViewModel	
Al manipular los datos mediante ViewModels, MVVM permite fácilmente realizar test unitarios, ya que estos no tienen referencia a los View. Al confinar la manipulación de datos únicamente en el ViewModel, manteniendolo de cualquier código de View, la lógica de negocio resulta testable.	 Figura 3: esquema MVVM
Viper	
Máximo nivel de abstracción entre patrones. Útil cuando se espera que el código crezca en cortos periodos de tiempo, siendo fácil y rápido añadir características. Puede ser excesivo cuando se trabaja en proyectos con código base simple.	 Figura 4: esquema Viper
MVI Model-View-Intent	
Model. Representa un estado. Deben ser inmutables para asegurar un flujo de datos unidireccional entre él y las demás capas. Intent representa una intención de realizar una acción por parte del usuario. Por cada acción de usuario la View recibe un intent, siendo observada por el Presenter, y traducido en un nuevo estado en los Model.	 Figura 5: esquema MVI

Tabla 2: Comparativa patrones de arquitectura

Índice

1.3.2.1 Criterios a la hora de escoger

MVC queda descartada, ya que en Android no puede ser correctamente implementada, debido a que el Activity hace las funciones tanto de controlador, como de vista, con lo que no cabe la separación de responsabilidades.(8) Otras arquitecturas como MVI o Viper también quedan descartadas, ya que aunque se adaptan a las necesidades, nunca he trabajado con ellas y supondrían una dificultad añadida innecesaria.

MVVM y MVP son adecuadas, ya he trabajado con ellas anteriormente, y además se adaptan perfectamente a la estructura de Activities de Android, pudiendo mantener la correcta separación de responsabilidades. De entre estas dos, elijo MVP, ya que MVVM añade cierta complejidad, que no vale la pena asumir para este proyecto .

1.3.3 Gestión de la información(9)(10)(11)

La app necesita almacenar datos para poder gestionar las listas de la compra. Existen los datos de las propias listas, que han de ser combinados con los datos de los supermercados, los usuarios, los productos, las categorías, etc. Conformando así la información que muestra la app para tener utilidad y poder facilitar la compra.

Los datos deben estar almacenados en algún lugar, al que la app pueda acceder de manera segura, estable y rápida. Para esta función sería perfectamente útil utilizar bases de datos locales, como son SqLite o Realm, incluso podría hacerse con la propia persistencia de datos de Android. Pero la app necesita poder compartir información, ya que uno de los objetivos secundarios es la posibilidad de compartir las listas entre diferentes usuarios, por lo tanto se hace necesario utilizar bases de datos On-line, como pueden ser Firebase, MySql o similares.

1.3.3.1 Datos en conectado

Para el almacenaje y acceso de datos on line, existen diferentes opciones, una de ellas es Mysql, un sistema de gestión de base de datos relacional con una gran popularidad debido a que está basada en código abierto.(9)(10) Sin embargo, presenta el inconveniente de necesitar de infraestructura para el backend, requiere un servidor web, con webservices para la publicación de los datos. Por otro lado, existe la opción de Firebase, que es una plataforma de desarrollo de aplicaciones, dentro de la cual se ofrece el servicio, entre otros, de base de datos ya sea mediante Realtime Database o Cloud Firestore.(11) Para este caso en concreto es interesante la opción de Realtime Database, una base de datos NoSQL alojada en la nube que permite almacenar y sincronizar datos en tiempo real.(12)

1.3.3.2 Datos locales

Una vez se obtienen los datos de la base de datos remota, es posible que en algún momento se pierda la conectividad con esta (perdida de cobertura, caída de Internet...) Entonces es cuando se puede hacer necesaria la utilización del almacenaje de datos local, de manera que cuando se restablezca la conexión, se puedan sincronizar los datos locales con los de la base de datos remota. Existen diferentes opciones para realizar esta tarea en Android, entre ellas Sqlite, Room, Realm, entre otras.

1.3.3.3 Criterios a la hora de escoger (12)(13)

Una vez estudiadas las diferentes opciones, la que mejor se adapta a este proyecto es la de Firebase Realtime database (RD), ya que además de ofrecer sincronización de datos en tiempo real, también permite colaborar entre dispositivos, lo cual facilita las listas compartidas que se quieren implementar en este trabajo. Como añadido, se pueden prescindir de los servidores que si requieren, por ejemplo, las bases de datos MySql. Realtime Database está optimizada para el uso sin conexión, con lo cual es posible prescindir del uso de las bases de datos locales de las que hemos hablado en el punto anterior. También permite integrar seguridad de Firebase Authentication, así como las ventajas de otros servicios integrados que permiten mejorar la calidad de la app.

Productos Firebase (13)	
Productos de desarrollo	
Cloud Firestore	Almacena y sincroniza datos entre usuarios y dispositivos a escala global
Firebase ML	Características machine learning en la app.
Cloud Functions	Código back-end personalizado.
Authentication	Administración de usuarios simple y segura.
Hosting	Hosting web.
Cloud Storage	Almacena y comparte contenido.
Realtime Database	Base de datos sincronizada en tiempo real.
Productos de calidad	
Crashlytics	Obtiene estadísticas de errores de la app en los dispositivos de los usuarios.
Supervisión del rendimiento	Diagnostica problemas de rendimiento en los dispositivos de los usuarios.
Test Lab	Ejecuta pruebas automáticas y personalizadas en dispositivos virtuales.
App Distribuition	Permite enviar versiones preliminares a los testers.
Productos de crecimiento	
In-App Messaging	Aumenta la participación de los usuarios mediante mensajes.
Google Analytics	Analiza el comportamiento de los usuarios.
Predictions	Obtención de estadísticas, para predecir usuarios que desertan, conversiones...
Pruebas A/B	Realiza experimentos de marketing
Cloud Messaging	Envía mensajes y notificaciones
Remote Config	Personaliza la manera en que se muestra la app a cada usuario, cambiando diseño, funciones...
Dynamic Links	Permite usar vínculos directos para potenciar las conversiones, uso compartido de usuarios, campañas sociales...

Tabla 3: Servicios Firebase

Índice

1.4 Planificación del Trabajo

En este proyecto intervienen dos personas, yo mismo, José Ángel Bravo, como autor de esta memoria (en adelante autor) y de los productos resultantes de este trabajo: app con manual y presentación. Así como Eduard Martín Lineros, profesor colaborador de la asignatura de Trabajo final de Máster de la UOC, que realiza el seguimiento del trabajo y valoración de las diferentes entregas, así como de todo el proceso (en adelante director). Además podrán intervenir, en momentos puntuales, terceras personas que hagan las funciones de probador de la app, para la realización de comprobaciones y la detección de bugs.

Los recursos necesarios para el desarrollo del proyecto se detallan en la tabla siguiente:

Recursos	
Hardware	
Ordenador Pc: procesador i5, 8Gb de RAM, disco SSD y S.O. Windows 10 64bits	
Ordenador Pc: procesador i7: 16Gb de RAM, disco SSD y SO Windows 10 64bits	
Móvil Android Xiaomi Mi Note 10 Lite 5G: 6Gb RAM, pantalla 6,57" y Android 10	
Móvil Android Samsung A20e: 2Gb RAM, pantalla 5,8" y Android 10	
Software	
Android Studio 4.0.1	
Emulador de Android studio: Nexus 5 API 29 - Android 10	
Emulador de Android studio: Tablet 10.1 WXGA API 29 - Android 10	
Ganttproject : creación de diagramas de gantt para la planificación.	
Dropbox: almacenaje y copias de los archivos del proyecto.	
Indesign: confección de la documentación del proyecto	
Photoshop: creación y edición de material gráfico de la app y documentación.	
Camtasia Studio: Grabación de pantalla para la documentación.	
Adobe Premiere-After Effects: Documentación audiovisual.	
https://app.dbdesigner.net/: Diagrama de base de datos	
Axure RP 9: Wireframes	
Justinmind: Creación prototipo	
Screen to gif: grabación de pantalla del prototipo	
Personal	
José Ángel Bravo Montañez	
Funciones	Autor del trabajo, responsable del desarrollo y calidad final
Roles	Jefe de proyecto, analista, diseñador, desarrollador, gestor de pruebas y de documentación.
Eduard Martín Lineros	
Funciones	Seguimiento del trabajo, asesoramiento y valoración
Roles	Director de proyecto, cliente
Terceras personas	
Funciones	Realización de pruebas de la app

Tabla 4: Recursos del proyecto

El proyecto requiere de una serie de tareas a realizar por el personal implicado según su rol. Asimismo, estas tareas se dividen en diferentes fases, al final de las cuales el autor del trabajo deberá entregar los documentos acordados que serán evaluados por el director del proyecto. Por otra parte, el director del proyecto asesorará al autor del trabajo y resolverá dudas, realizando las aclaraciones necesarias que vayan apareciendo en el tiempo que dure el proyecto.

Las fases del proyecto son cuatro: plan de trabajo, diseño, implementación y entrega final. A la finalización de cada una de las fases, el autor realizará la entrega de una versión actualizada de la memoria: PEC1 tras el plan de trabajo, PEC2 tras la fase de diseño, PEC3 tras la implementación, y para finalizar, la versión final de la memoria, así como el resto de documentos y programas resultado del proyecto.

Fase del proyecto		
Título	**Entregables**	**Objetivo**
Plan de trabajo	PEC1 (versión 1 de la memoria)	Explicación del contexto, justificación y método del trabajo, así como la definición de objetivos.
Diseño	PEC2 (versión 2 de la memoria) Demostración del prototipo.	Estudio de usuarios, contexto y escenarios de uso. Definición de interacción y diseño de prototipos.
Implementación	PEC3 (versión 3 de la memoria) Ejecutable de la APP. Código fuente BBDD de datos por defecto (Json)	Realización del proyecto: programación y documentación.
Entrega final	PEC4 (versión final de la memoria) Demostración del prototipo Ejecutable de la APP. Código fuente BBDD de datos por defecto (Json) Manual de usuario Presentación visual	Finalizar la documentación y crear presentación y manual.

Tabla 5: Fases del proyecto

El proyecto tiene lugar entre los meses de septiembre y diciembre de 2020, teniendo una duración estimada de 350h, que se dividen en: 70h el plan de trabajo, 70h la fase de diseño, 140h la implementación y 70h para la entrega final. La dedicación viene dada por la disponibilidad de horas por parte del autor, ya que realiza este trabajo a tiempo parcial, dedicando 3h al día en jornadas en día laboral, y 4h al día en jornadas de festivos y fines de semana.

Las diferentes faenas tienen distinta Jerarquía según su importancia en cada momento del proyecto, por ejemplo, la documentación tiene la máxima jerarquía en la fase de plan de trabajo, dedicándose el 3/4 del tiempo disponible, el cual debe compartir con las tareas de investigación, que en este caso ocupan 1/4 del tiempo. Sin embargo en la segunda fase, la documentación pasa a ocupar tan solo 1/4 del tiempo, en detrimento del desarrollo del diseño, que obtiene 2/4 del tiempo, conservando la investigación 1/4 de este. De nuevo, en la fase de implementación, la documentación ocupa 1/4, mientras desarrollo 3/4 del tiempo. Sin embargo, en la última fase, todo el tiempo será dedicado a la documentación, excepto al reservado (como en cada una de las fases anteriores) a las correcciones indicadas por el director.

1.4.1 Diagrama de Gantt

Nombre	Fecha de inicio	Fecha de fin	Duración
Trabajo Final de Máster	16/09/20	30/12/20	350h
Plan de trabajo (PEC1)	16/09/20	7/10/20	70h
Propuesta de tema	16/09/20	16/09/20	1h
Definición de tema y resumen	17/09/20	17/09/20	0h
Investigación tecnologías	17/09/20	6/10/20	16h
Documentación	17/09/20	7/10/20	53h
Desarrollo memoria	17/09/20	5/10/20	48h
Repaso y correcciones	6/10/20	7/10/20	5h
Entrega PEC1	7/10/20	7/10/20	0h
Diseño (PEC2)	8/10/20	28/10/20	70h
Aplicar correcciones -Plan de trabajo-	8/10/20	8/10/20	3h
Investigación tecnologías	9/10/20	27/10/20	16h
Desarrollo	9/10/20	28/10/20	38h
Conceptualización	9/10/20	13/10/20	13h
Benchmarking	9/10/20	11/10/20	8h
Definición de funcionalidades	12/10/20	13/10/20	5h
Diseño de la arquitectura	14/10/20	28/10/20	25h
Diseño navegación	14/10/20	16/10/20	5h
Diseño de las BBDD	17/10/20	20/10/20	7h
Diseño de pantallas	21/10/20	28/10/20	13h
Esbozos en papel	21/10/20	22/10/20	3h
Prototipos baja fidelidad	23/10/20	28/10/20	10h
Documentación	14/10/20	28/10/20	13h
Desarrollo memoria	14/10/20	26/10/20	11h
Repaso y correcciones	27/10/20	28/10/20	2h
Entrega PEC2	28/10/20	28/10/20	0h
Implementación (PEC3)	29/10/20	9/12/20	140h
Aplicar correcciones -Diseño-	29/10/20	30/10/20	6h
Desarrollo	31/10/20	9/12/20	100
Programación	31/10/20	29/11/20	75h
Gestión de datos	31/10/20	14/11/20	30h
Implementación de pantallas	7/11/20	22/11/20	15h
Navegación y usabilidad	15/11/20	29/11/20	30h
Pruebas y mejoras	30/11/20	9/12/20	25h
Documentación	31/10/20	9/12/20	34h
Desarrollo memoria	31/10/20	5/12/20	30h
Repaso y correcciones	6/12/20	9/12/20	4h
Entrega PEC3	9/12/20	9/12/20	0h
Entrega Final (PEC4)	10/12/20	30/12/20	70h
Aplicar correcciones -Implementación-	10/12/20	18/12/20	30h
Documentación audiovisual	19/12/20	30/12/20	25h
Desarrollo presentación	19/12/20	27/12/20	22h
Repaso y correcciones	28/12/20	30/12/20	3h
Documentación	10/12/20	30/12/20	15h
Desarrollo manual usuario	10/12/20	20/12/20	9h
Desarrollo memoria	21/12/20	27/12/20	4h
Repaso y correcciones	28/12/20	30/12/20	2h
Entrega Final	30/12/20	30/12/20	0h

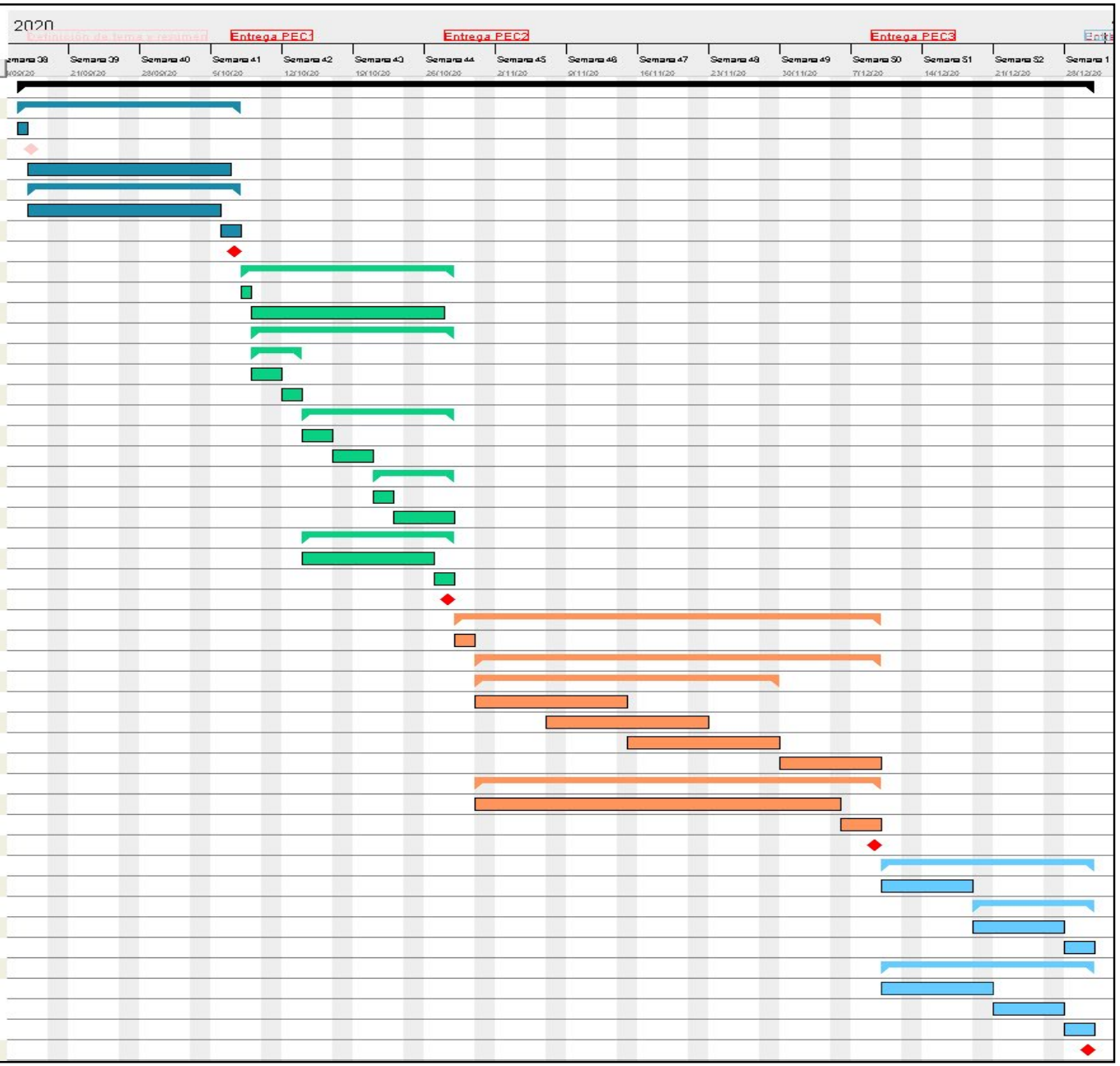


Figura 6: Diagrama de Gantt del proyecto

1.5 Breve sumario de productos obtenidos

- Memoria del proyecto.
- APK ejecutable de la APP.
- BBDD de datos por defecto (JSON)
- Demo del prototipo.
- Manual de usuario de la APP.
- Código fuente.
- Presentación visual.

1.6 Breve descripción de los otros capítulos de la memoria

Capitulo 2. Resto de capítulos:

- Capítulo 2.1-Diseño: Se realiza estudio de mercado de aplicaciones similares, con las conclusiones se definirán y diseñarán la visual y funciones de la app.

  Contendrá una **conceptualización**, donde se realiza el estudio de mercado en sí, estudio de usuarios, casos de uso, benchmarking, etc. A partir de esta, se define la **Arquitectura**: diseño de la estructura de datos y de pantallas. Para finalmente presentar la realización de un **Prototipo**.

- Capítulo 2.2-Implementación: Se realiza el proceso de programación y desarrollo de la app, con base a las conclusiones obtenidas en el punto anterior.

Capítulo 3. Conclusiones

- Lecciones aprendidas del trabajo, reflexión crítica del logro de objetivos planteados, análisis del seguimiento de la planificación y metodología, así como líneas de trabajo futuro que puedan quedar pendientes.

Capítulo 4. Glosario

- Definición de los términos y acrónimos más relevantes utilizados dentro de esta memoria.

Capítulo 5. Bibliografía

- Listado de referencias bibliográficas e imágenes utilizadas

Capítulo 6. Anexos

- Apartados demasiado extensos para incluirse dentro de la memoria.

2. Resto de capítulos

2.1 Diseño

La idea es diseñar una aplicación sencilla, nativa para el sistema operativo Android, con la que, de la manera más simple y optimizada posible, poder realizar la compra. Para conseguirlo será necesario que el usuario pueda entrar los datos de las categorías, y el orden que estos tienen en cada una de las tiendas. Asimismo, los productos, al ir asignados a las categorías, se ordenarán de manera automática a la hora de mostrar las diferentes listas de la compra.

Para conseguir el objetivo, se deberán diseñar (datos+visual) mantenimientos para: artículos, categorías, tiendas y las relaciones entre ellos. De esta manera un usuario podrá mantener varias tiendas, estas contener varios departamentos o categorías, y estos, a su vez, multitud de productos. Todos los datos deben poder "combinarse" entre sí, de manera que una categoría puede pertenecer a diferentes tiendas. Además para que un usuario pueda acceder a las listas (y por tanto productos, categorías y tiendas) de otro usuario, este deberá compartirlas, pudiendo entonces interactuar ambos (o más) usuarios con los mismos datos.

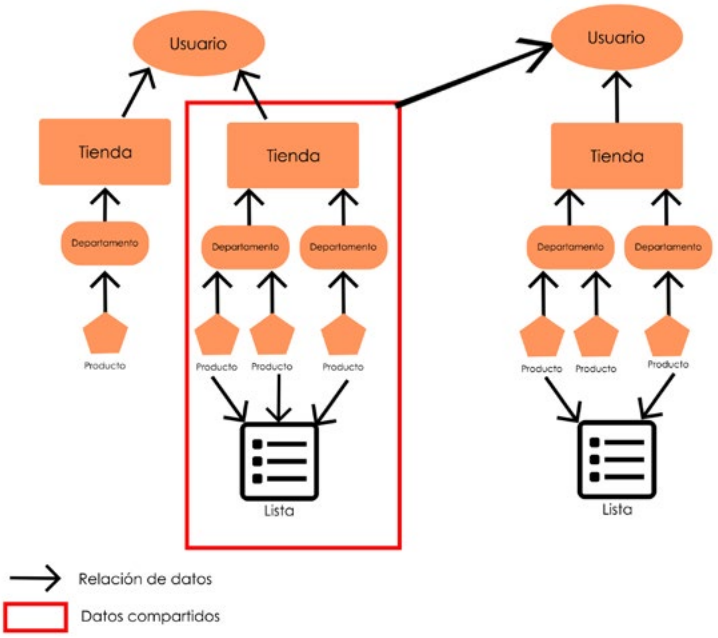


Figura 7: Flujo y pertenencia de datos

Todo debe ser presentado de manera minimalista, mostrando tan solo la información estrictamente necesaria, mediante fuentes con gran visibilidad (tamaño grande, colores contrastados...), y de una manera atractiva visualmente hablando. Además se deberá cuidar la usabilidad, facilitar al usuario el movimiento entre pantallas, la entrada de datos, etc. En especial, debe ser cuidado el diseño y usabilidad en la pantalla de compra, ya que es la que se usa "en campo" y puede resultarle complicado al usuario, quizás teniendo que utilizar el teléfono con una sola mano, entre otras dificultades.

2.1.1 Conceptualización

2.1.1.1 Benchmarking

Para la realización de este benchmarking he realizado una búsqueda en la Play Store de Android, con las palabras clave “lista compra” con lo que el resultado devuelve aplicaciones enfocadas a gestionar listas de la compra, de las cuales he realizado una selección con las que he considerado más cercanas, conceptualmente, a la propuesta de este trabajo de entre las primeras que aparecen en la lista, eligiendo bajo los criterios: puntuación de los usuarios, algunas que ya había utilizado en mi día a día y finalmente, alguna al azar.

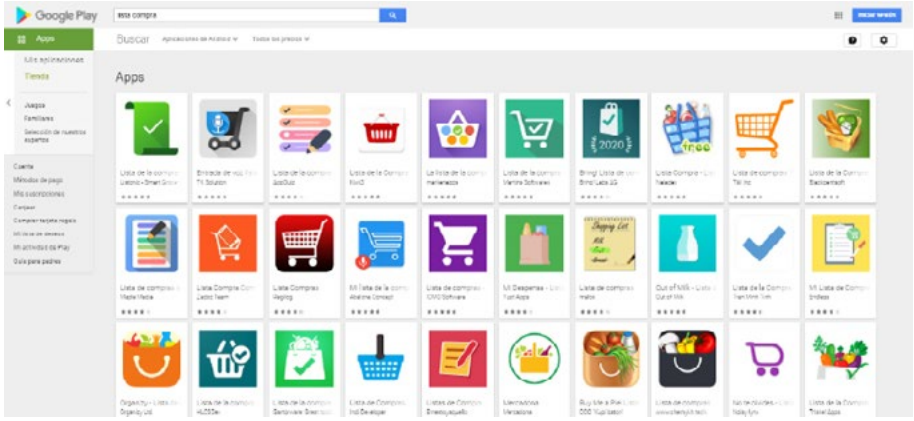


Figura 8: Búsqueda en Google play: lista compra

De la selección realizada a partir de la búsqueda anterior, he testeado las apps, comprobando su funcionamiento, su estructura y su usabilidad, con la intención de encontrar sus fortalezas y debilidades.

Como resultado de estos test, he anotado las cualidades que me han parecido interesantes, así como las que creo que podrían mejorarla. Como resultado se obtendrá la recopilación de funciones imprescindibles, elementos visuales e interacciones aplicables, así como, si es posible, la detección de mejoras e innovaciones, no existentes en la actualidad en las APPS estudiadas.

El objetivo del benchmarking es que todas las conclusiones resultantes del proceso puedan ser aplicadas en la app objeto de este trabajo. Tanto para la inclusión de las funciones imprescindibles, como, en la medida de lo posible, de las características interesantes. Por otro lado, las conclusiones también deben mostrar los puntos flacos de las apps, los cuales hay que evitar.

Se puede consultar el benchmarking en el siguiente enlace:

Enlace al anexo 1: Estudio de mercado

Conclusiones

Una funcionalidad común a prácticamente todas las apps estudiadas es la posibilidad de configurar diversas listas, pudiendo alternar entre ellas. Además en la mayoría de los casos, estas listas ofrecen la posibilidad de gestión compartida por varios usuarios.

Hay pocas app que ofrezcan la opción de gestión completa de categorías, de manera que estas tengan un orden configurable, y que los artículos se inserten en la lista automáticamente según este orden. Reafirmándose de esta manera la necesidad de incluir esta funcionalidad.

Las apps que utilizan una interfaz gráfica muy cargada, muchos iconos, tipografías complicadas, etc, al principio resultan muy atractivas a la vista, pero a la hora de su uso, resultan confusas y pueden acabar siendo cargantes. Es preferible primar la sencillez a la hora de diseñar la interfaz, priorizar el texto, sin renunciar a detalles gráficos, pero sin basarse en ellos.

Uno de los mayores problemas detectados es la selección de artículos con solo un toque, lo que implica que es bastante fácil tocar por error un producto, provocando que este pase a la lista de comprados sin que el usuario se percate. Es la manera más sencilla para el usuario de marcar productos, pero a la vez es también con la que más confusiones se producen. Por tanto, considero que la utilización de swipe es la más apropiada para realizar la tarea de marcar artículos como comprados.

Con todo esto, se puede concluir que hay algunas opciones que resultan muy interesantes, y deberán ser incluidas en la app. Estas son:

- Gestión de categorías con ordenación automática de artículos en la lista. Es básico, al insertar artículos en la lista, estos se deben ordenar según su orden específico.

- Lista con gestión compartida, que un usuario pueda compartir los datos de sus listas para que otros usuarios puedan participar de la gestión de estos.

- Uso de "swipe", tanto para moverse entre las opciones de la interfaz, como para la selección de usuarios, evitando así la "selección a un toque" y los errores que provoca.

- Interfaz gráfica sencilla. Una interfaz demasiado cargada puede ir en contra de la funcionalidad.

2.1.1.2 Usuarios y casos

Debido al contexto actual de pandemia mundial de COVID-19, resulta muy complicado el poder realizar las entrevistas a usuarios que suelen llevarse a cabo en este punto. Es por esto que los usuarios y situaciones a continuación serán ficticios, pero basados en la experiencia personal, trabajos realizados con anterioridad en el Máster, y en la investigación, realizada aprovechando el benchmarking del punto anterior, de los usuarios y comentarios de la tienda de apps Google Play, donde se puede llegar a detectar el público objetivo, y a entender ciertas necesidades e inquietudes de este.

"Que lo perfecto no sea enemigo de lo bueno." Voltaire.

Enlace al Anexo 2: fichas de usuario

2.1.1.3 Conclusiones

De la fase de conceptualización, de la detección de los puntos fuertes y débiles, así como de la observación de otros aspectos, se pueden extraer una serie de conclusiones de las cuales se obtiene la siguiente serie de funciones y requisitos interesantes para incluir en la app:

- La app debe abrirse en la última lista de la compra utilizada.
- Uso de "Navigation Drawer" para accesos rápidos a otras pantallas.
- Debe ofrecer la posibilidad de gestionar diversas listas.
- Debe tener una interfaz sencilla y fácil de utilizar
- Es preferible el uso de funciones swipe a las de un toque, ya que evitan errores al marcar elementos por accidente.
- Las listas deben poder ser compartidas, y gestionadas por varios usuarios, recibiendo estos notificaciones de las modificaciones.
- Debe haber una lista de productos y de categorías (o departamentos) definidos por defecto, y dar la posibilidad de modificar y añadir datos.
- El uso de imágenes es recomendable, pero de manera auxiliar. Es prioritaria la sencillez a la vistosidad.
- Es imprescindible que a las categorías o departamentos se les pueda asignar un orden dentro de la tienda.
- Gestión de acceso de usuarios con Google es recomendable.
- Poder utilizar códigos de barras para la búsqueda de artículos es recomendable.
- El control de los precios de los artículos en la lista es recomendable.

2.1.2 Arquitectura

En este capítulo se sientan las bases técnicas del diseño de la app. Se define la estructura de los datos, así como un primer contacto con el diseño de las diferentes pantallas y de como será la navegación entre ellas.

2.1.2.1 Árbol de navegación

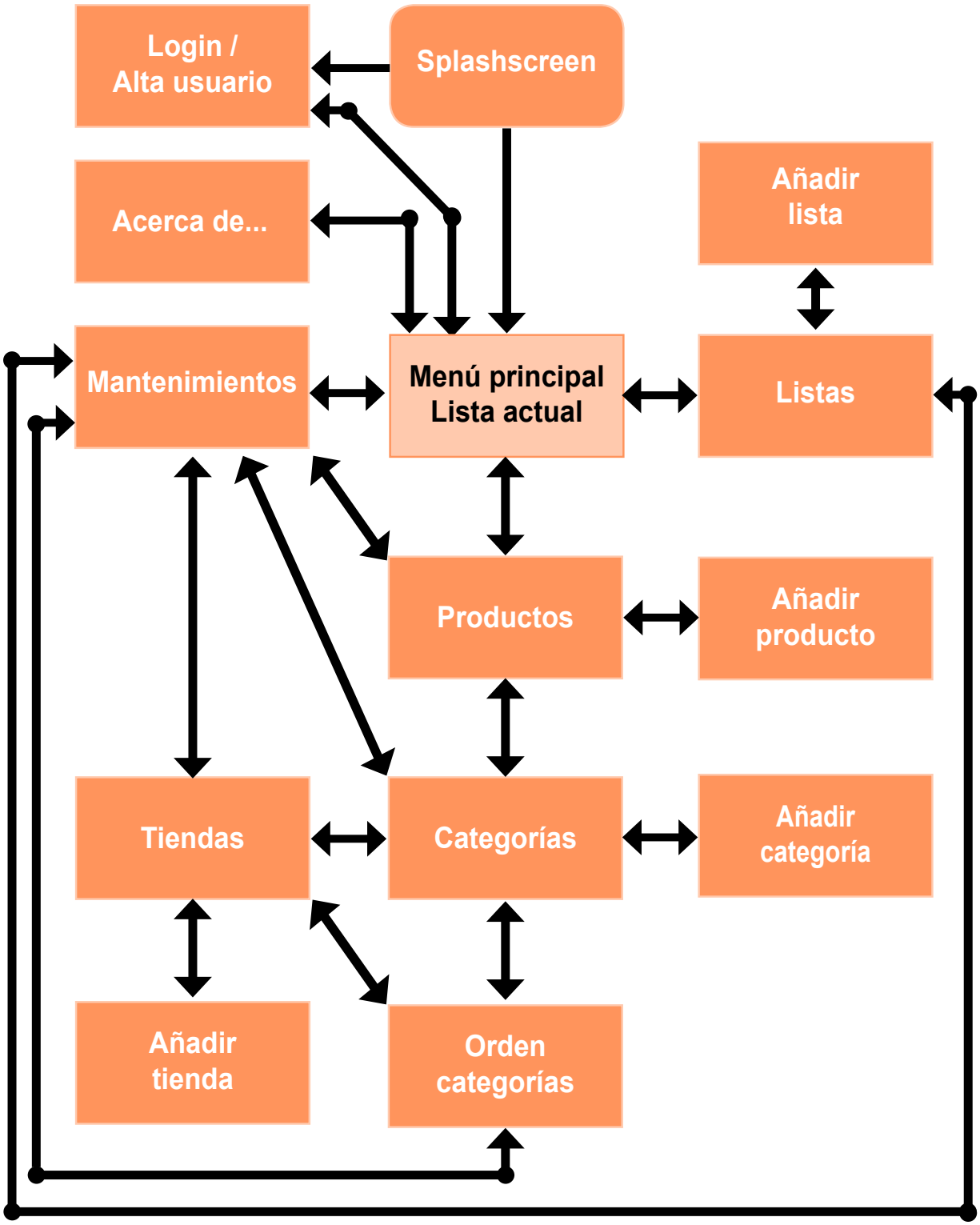


Figura 19: Árbol de Navegación

El punto de acceso a la app siempre es la pantalla de bienvenida o splashscreen. De ahí, se pasa a la pantalla de login/alta de usuario, en el caso de ser la primera vez que se utiliza la app, o directamente a la última lista utilizada, en caso de ya estar logado. La pantalla de menú principal ejerce de eje central, desde el que se puede acceder a las pantallas de login, cambio de lista, o mantenimientos. Esta pantalla contendrá en su interior el resto de las pantallas, mediante fragments.

En la pantalla de mantenimientos, se accede a las diferentes pantallas de gestión, como son productos, categorías y tiendas, así como a la de orden de categorías y las de las listas. Desde la pantalla de gestión de productos, se puede acceder a la de categorías, y de esta a la de tiendas.

Además, la pantalla de mantenimientos, así como las de listas y lista actual son accesibles en todo momento desde el menú lateral, que se abre mediante el botón de "hamburguesa" de la parte superior izquierda, o bien "arrastrando" del borde izquierdo hacia el centro de la pantalla.

2.1.2.2 Estructura de datos

La estructura de datos de la app es un árbol JSON de Realtime Database con 7 nodos que contienen los datos en forma de claves JSON.

En una estructura de árbol JSON Los nodos equivaldrían a las tablas de una BBDD SQL "tradicional", mientras que las claves equivaldrían a los campos de estas tablas. De esta manera, se puede representar como se muestra en la siguiente figura:

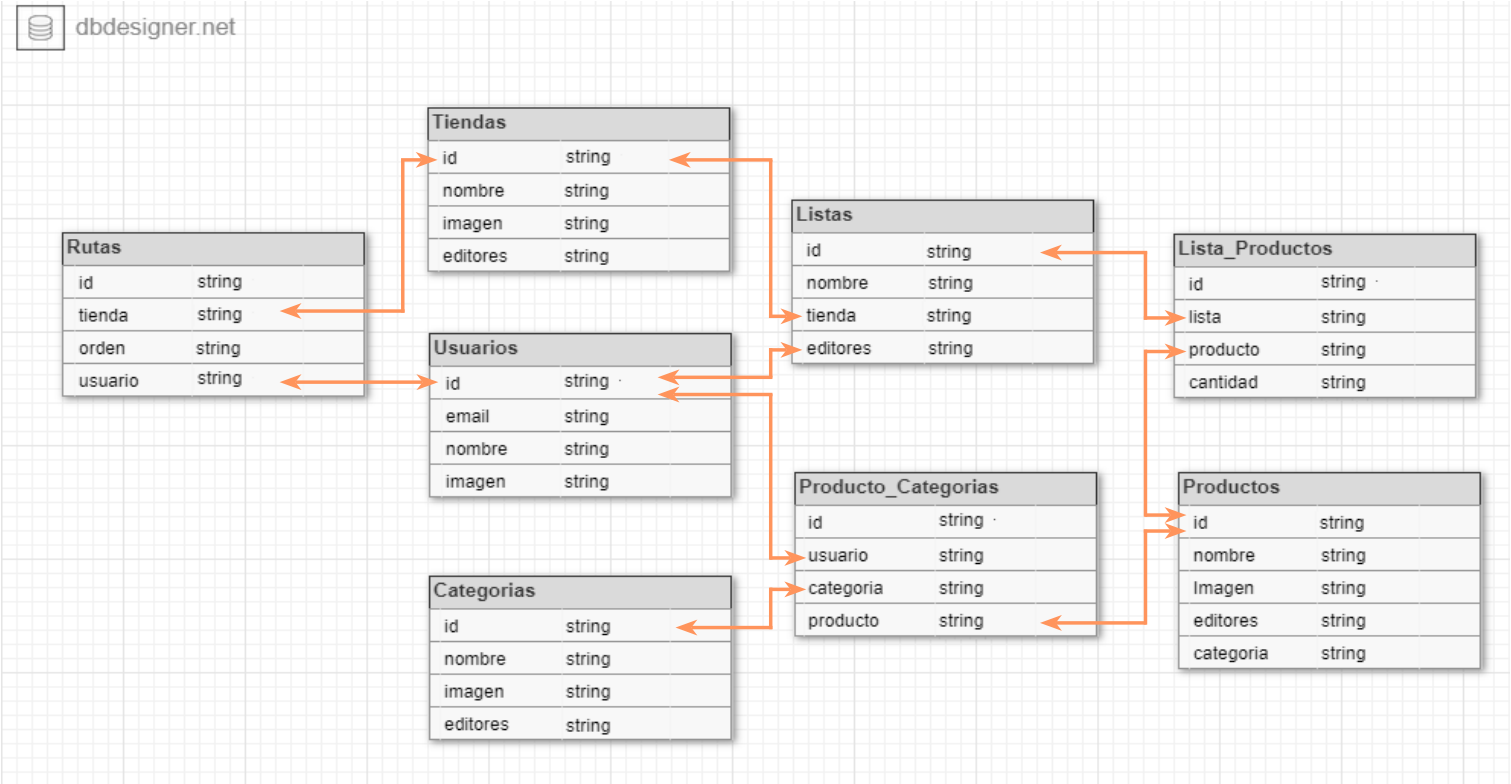


Figura 20: Estructura de datos

Existen 5 nodos: Usuarios, Tiendas, Categorías, Productos y Listas, donde se almacenan los datos básicos que contendrán las listas. Los dos nodos restantes: Lista_Productos, Productos_categorias y Orden_categorias, dotan de estructura a las listas, montando los datos de los nodos anteriores, conteniendo el orden en que deben mostrarse los productos, y quien puede modificar los datos.

Firebase entrega los datos de los árboles de datos declarados en Realtime Database en formato JSON, un formato texto sencillo utilizado para intercambio de datos.
A continuación se puede apreciar la manera en que se tratan y muestran los datos en la interfaz de Firebase, y como son entregados en formato Json:

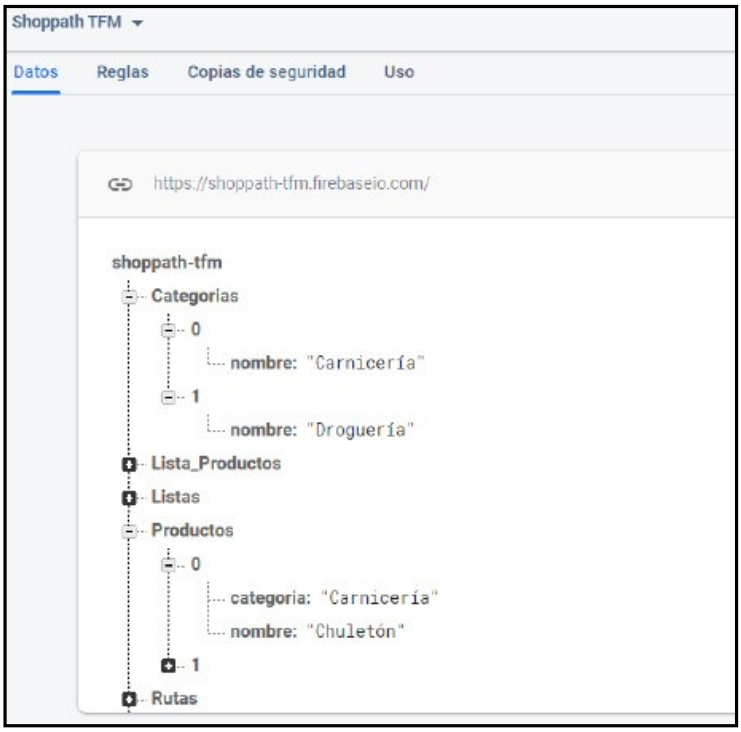


Figura 21: Vista en Realtime Database

Figura 22: Estructura de datos JSON

2.1.2.3 Pantallas

Esbozos en papel

Para comenzar con el diseño de las pantallas se realizan unos primeros esbozos, en papel y lápiz, de todas las pantallas que conforman la aplicación, tanto en su versión horizontal, como en vertical y para tablet.

La totalidad de las pantallas se pueden consultar en:

Enlace al anexo 3: Esbozos en papel

Wireframes

A continuación, como paso siguiente a los esbozos en papel, se diseñan wireframes de las pantallas, tanto en su versión horizontal, como en vertical y para tablet. Los wireframes son una primera evolución de los esbozos, con mayor calidad gráfica, a los que se añaden detalles.

La totalidad de los wireframes se pueden consultar en:

Enlace al anexo 4: Wireframes

2.1.3 Prototipo

Desde el enlace a continuación se accede al prototipo interactivo en alta fidelidad. Es el resultado de todo el proceso de diseño. Se trata de una evolución de los wireframes, con la que además se puede interactuar, sirviendo como demostración aproximada de como se verá y funcionará el producto final.

Para interactuar con el prototipo tan solo es necesario hacer click en las partes interactivas (estas se iluminan al hacer click en cualquier parte de la pantalla). Los campos de texto (email, pasword, etc) no es necesario rellenarlos, ya que tan solo es una demostración gráfica, no existe tratamiento de datos.

Enlace a demostración interactiva del prototipo

En la demostración existe un enlace al vídeo demostrativo, pero también se puede acceder desde aquí:

Enlace a vídeo demostrativo del prototipo

Como añadido, por si por alguna razón no funciona la versión en línea, se entrega, junto a esta memoria, como un entregable más del TFM, una copia del prototipo ejecutable en local.

2.2 Implementación

2.2.1 Desarrollo

En el diseño y desarrollo de la app se ha intentado en todo momento incluir la mayor parte de los contenidos estudiados durante el Máster, de una manera que estos contenidos fueran útiles y cumplieran con una funcionalidad necesaria de la app.

A continuación se detallan los contenidos estudiados que han sido aplicados, con la funcionalidad que cumplen dentro de la app

-Splashscreen

Se aplicó por primera vez en la segunda de las prácticas de la asignatura Tecnología y Desarrollo en Dispositivos Móviles (TDDM), siendo un fijo su uso como "bienvenida" en cada uno de los proyectos realizados durante el Máster. Suele aprovecharse para la descarga de los datos de la aplicación, pero no es el caso en este proyecto, en el que simplemente tiene finalidades estéticas.

-Login Activity y Login con Firebase

La Actividad de login, utilizada de una manera muy sencilla, en las dos prácticas de la asignatura TDDM, en este proyecto da un paso más, utilizando la conexión con cuenta de Google, mediante la funcionalidad que ofrece Firebase,(14) funciones del cual empezamos a ver en la primera PEC de Desarrollo de Aplicaciones para móviles Dispositivos Android (DADA), viéndose la autenticación en concreto en la segunda PEC de Desarrollo Avanzado de Aplicaciones para móviles Dispositivos Android (DAADA). (15)(16)(17)(18)

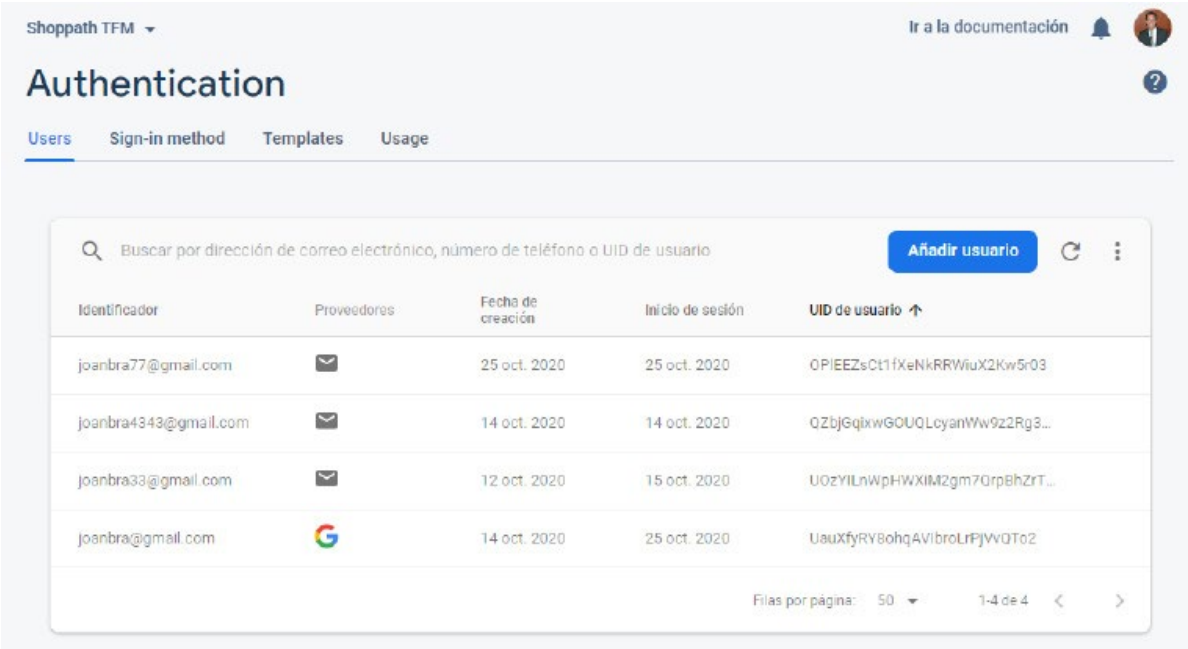


Figura 23: Detalle de la consola de Firebase: Autenticación.

La utilización de Firebase implica la implementación, mediante su consola, de unas

reglas[19] que permitan el acceso a los datos que ofrece, para este proyecto, ha habido que realizarlas tanto para el acceso a Realtime Database, como a Storage.

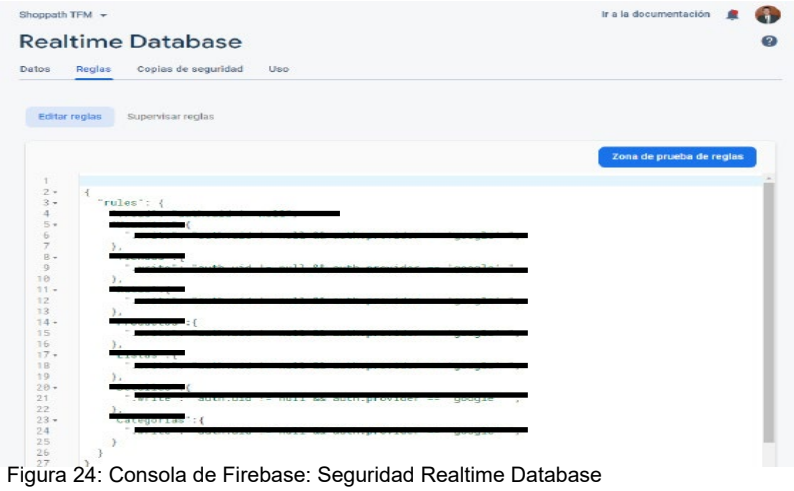


Figura 24: Consola de Firebase: Seguridad Realtime Database

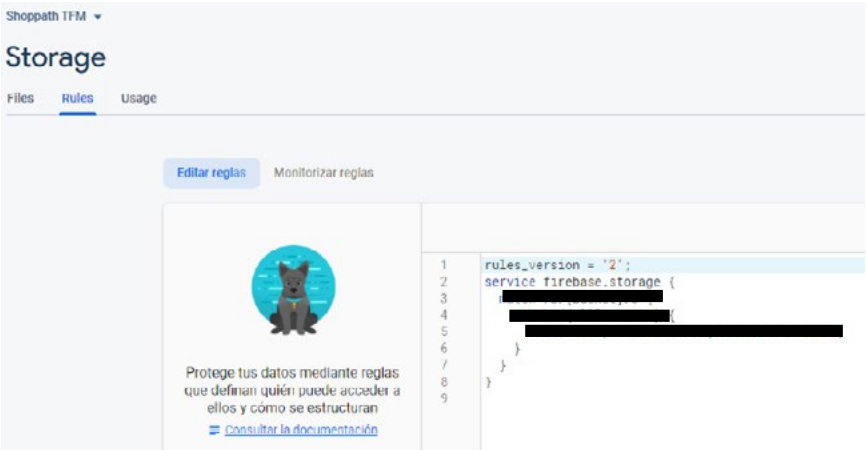


Figura 25: Consola de Firebase: Seguridad Storage

-Intents y Fragments[20]

Lo más básico de Android, o como cambiar las vistas de pantalla. Utilización por primera vez en la primera de las prácticas de TDDM, y utilizado, como no puede ser de otra manera, a lo largo de todo el Máster.

-Recyclerview[21]

Otro de los básicos de Android, se estudió en la primera práctica de TDDM, para utilizarse constantemente durante todo el Máster, y este trabajo final no es una excepción, solo que se le ha añadido funcionalidad no estudiada anteriormente. Esta es la posibilidad de reordenar filas, o de eliminarlas mediante arrastrarlas con el dedo, el llamado Swipe, gracias a la inclusión del elemento touchhelper. [22]

Dentro del Recyclerview, y en la primera PEC de DAADA, se aplicó el uso de Cardviews, para mejorar la visual de la lista.

El hecho de poder eliminar filas, llevó a la aparición de un problema, por lo visto no solucionado por Google, con el Recyclerview, los errores "Inconsistency detected"[23][24]

Otros errores que aparecieron durante el desarrollo, y que hacían que el programa fallara al eliminar elementos del recyclerviev, y esto era debido a que se "rompían" los índices de los datos que se manejan para mostrarse en el Recyclerview. Estos errores me obligaron a realizar una refactorización del código con el que se obtenían los datos, pasando de descargar de Firebase directamente a un DataSnapshot, a tener que incluir un paso intermedio que agregara los datos, de uno en uno, a una lista, y asignar esta al Recyclerview.

-Shared preferences (25)(26)

Trabajado en la segunda práctica de TDDM, es una de las formas de consistencia de datos que posee Android. En este proyecto han sido utilizadas para guardar la última lista con la que se ha trabajado, de manera que al volver a arrancar el programa, obtiene de nuevo estos datos, pudiendo arrancarse en esta misma lista de la compra.

- Navigation Drawer (27)(28)

Se trata de un elemento importante de la interfaz de usuario de esta app, ya que conforma el menú principal y soporta la navegación por las diferentes pantallas. En el Máster, se utilizó por primera vez en la segunda práctica de TDDM.

-Internacionalización (29)

Mediante el duplicado de los archivos strings.XML, creando uno para cada lenguaje deseado, se puede asignar a los elementos de texto el recurso concreto y es el sistema el que se encarga de utilizar uno u otro archivo de strings, para mostrar el idioma correspondiente. En este caso se han utilizado 3, castellano, catalán e Ingles.

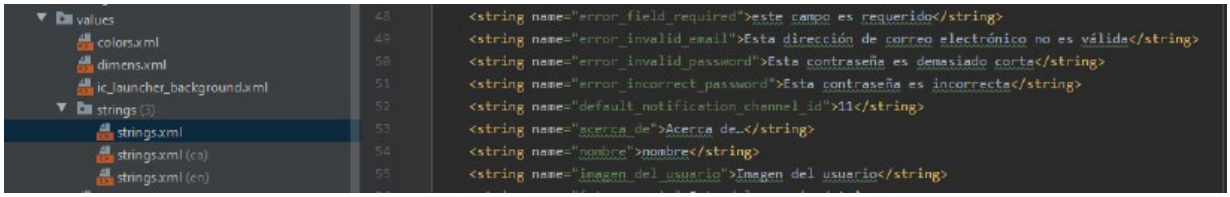


Figura 26: Vista de los archivos de strings de los diferentes idiomas.

-Icono del proyecto

Uno de los aspectos más simples estudiados durante el Máster, pero que sin el, la app queda huérfana de personalidad. Se vio por primera vez en la primera PEC de DADA.

-Toast y SnackBar

Mostrar mensajes por pantalla, en sus diferentes formas. Visto por primera vez en la segunda PEC de DADA.

-Dashboard UI

En la tercera PEC de DADA se nos introdujo al patrón de diseño Dashboard, o panel de control, en este proyecto lo he implementado en la pantalla de mantenimientos.

Índice

-Picasso

Se trata de una librería para el tratamiento de las imágenes mostradas dentro de las activities. Estudiado en primer lugar durante la tercera PEC de DADA.

Como añadido, he aplicado una transformación de imagen para que tenga forma redonda(30)(31) de manera que he podido utilizar la imagen de usuario para substituir al botón flotante del menú principal.

-Firebase Realtime Database

El uso de la base de datos del BAAS Firebase lo implementamos en la PEC3 de DADA por primera vez, siendo una constante posteriormente, y este proyecto no es una excepción.

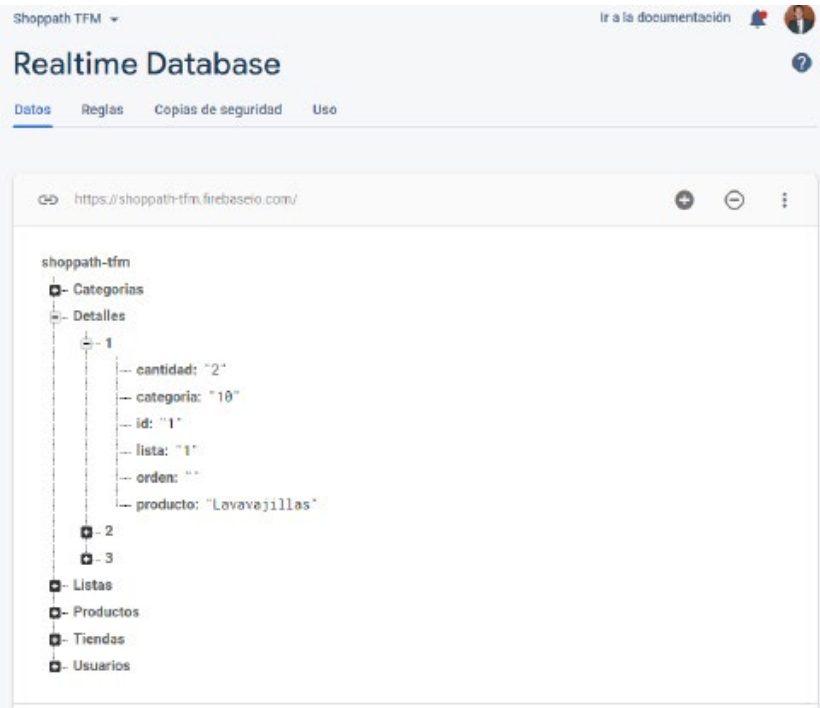


Figura 27: Vista de la consola de Firebase - Realtime Database

-Funcionalidad API de la cámara

En la tercera PEC de DADA se estudió el uso de la API para el uso de la cámara, por contra, en este proyecto he optado por utilizar una librería externa (32), ya que ofrece una funcionalidad completa para la selección de imágenes, tanto de la cámara como de la galería.

Además, como añadido, se ha iniciado una colaboración con el autor, para la traducción de los textos de la librería al idioma castellano.

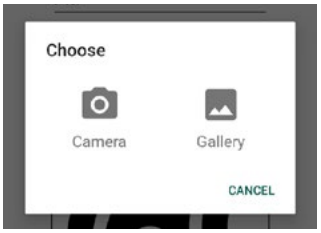


Figura 28: Selección de imagen librería Dhaval2404/ImagePicker

-Firebase Crashlytics y Analytics (33)(34)

Mediante la implementación de estos servicios de Firebase es posible controlar remotamente los errores que ocurren en los clientes con la app instalada, así como estadísticas de uso y la participación de usuarios. Estudiado durante la PEC 4 de DADA.

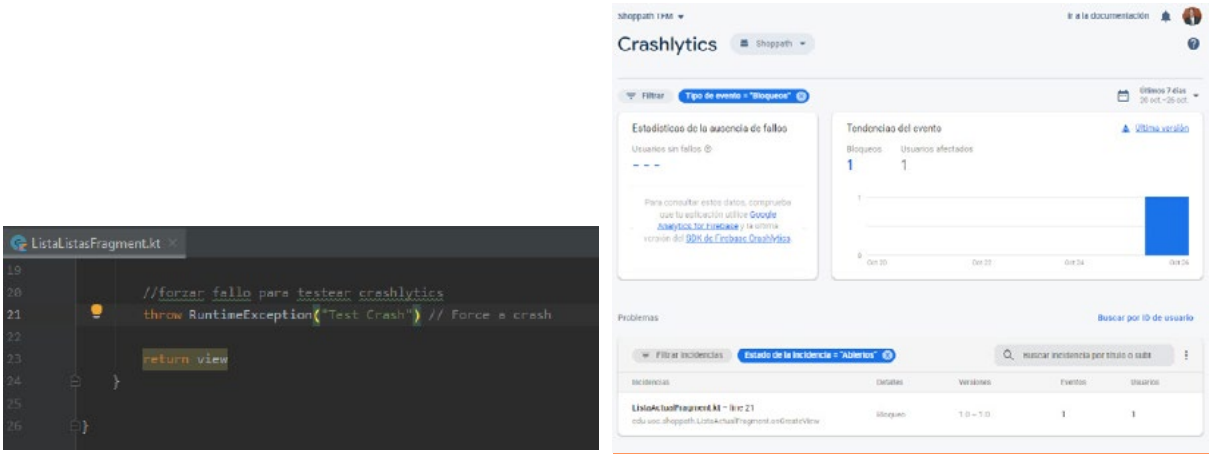


Figura 29: Error forzado por código, y su reflejo en consola Crashlytics

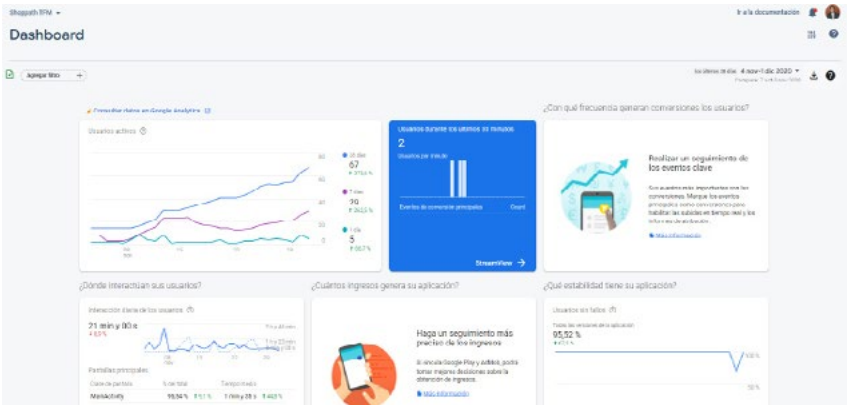


Figura 30: Consola Analytics

-Diferentes diseños de pantalla según dispositivo

En la primera de las PEC de DAADA se introdujo la diferenciación de diseños para los diferentes dispositivos (smartphone-tablet) y posiciones de pantalla (horizontal-vertical). En este proyecto se han implementado diseños diferenciados tanto para pantalla horizontal-vertical, para smartphone, así como un único diseño para tablets (horizontal)

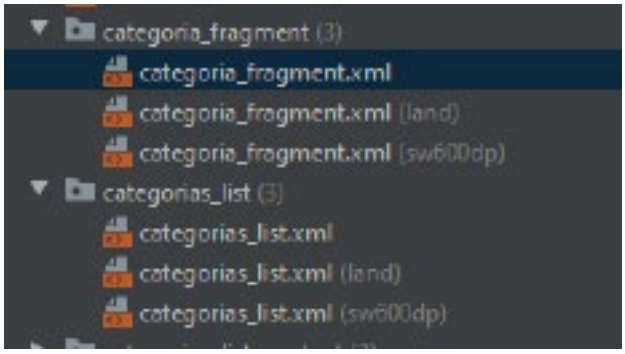


Figura 31: Diferentes XML para diferentes diseños de pantalla

- Firebase cloud messaging

En la tercera PEC de DAADA vimos el uso de notificaciones de Firebase, y su personalización, también se ha aplicado en este proyecto.

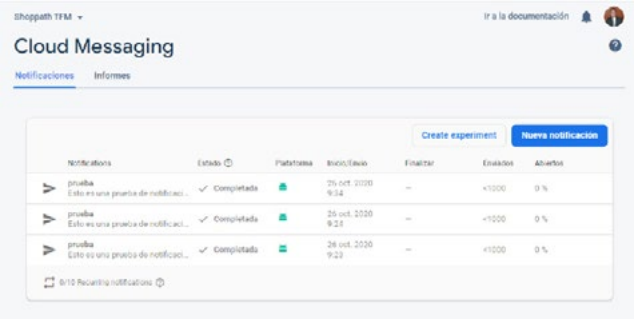


Figura 32: Consola Cloud Messaging Firebase

Figura 33: Ejemplo notificación

-Google adMob

Plataforma de Google para anuncios in-app. Se estudió en la PEC3 de DAADA.

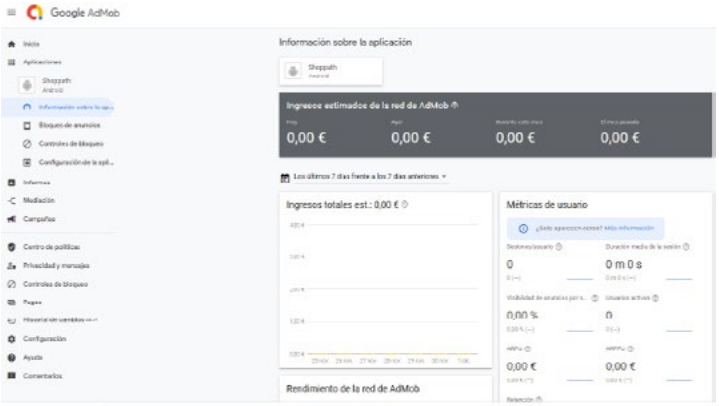


Figura 34: Consola AdMob

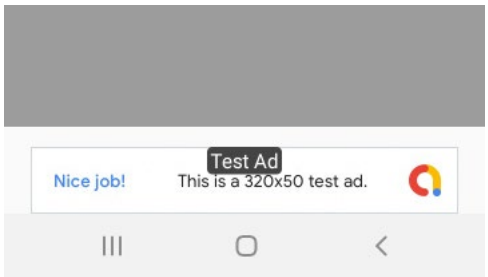


Figura 35: Vista en la app de banner AdMob

-MVP

Estudiado en la última de las PECs del Máster (DAADA) es el patrón de arquitectura implementado en (casi la totalidad) el proyecto.

Figura 36: Diferentes archivos para diferentes responsabilidades

-Inyección de dependencias
Estudiado junto a los patrones de arquitectura de la última PEC e implementado en la confección de esta APP

-Análisis de código estático
Comprobaciones finales para la calidad del código, vistas también en la última PEC de DADA y del Máster.

-Testing (Espresso/Mockito/JUnit)
Visto tanto en la cuarta PEC de DADA, como en la última de DAADA. Se ha intentado implementar en este proyecto, pero no ha resultado un proceso exitoso.

-ReactiveX
Mediante esta librería se permite crear programas asíncronos basados en eventos, siguiendo el patrón observador. Utilizando objetos observable para manejar y controlar el flujo de eventos. Se introdujo en la cuarta PEC de DAADA.

Y hasta aquí el contenido aplicado en este proyecto que ha sido directamente estudiado (y requerido mediante PEC y prácticas) durante el transcurso de las asignaturas del Máster. Obviamente ha habido más contenido estudiado, pero por diferentes motivos no ha podido ser aplicado en esta app, principalmente por ser funciones que no han tenido cabida o necesidad (mapas, Rest-API, BBDD locales, Product flavour o compartir contenido con otras apps).

Por otro lado, existen algunas funciones que han sido añadidas, y han tenido que ser investigadas durante el transcurso de este trabajo, algunos ejemplos son:

-Botón personalizado(35)
En la pantalla de gestión de Tienda, se me hizo necesario crear un botón personalizado, con imagen y texto, para acceder a la gestión de orden de categorías.

Mediante un nuevo archivo XML de layout, que insertado dentro del fragment hacía las funciones de botón

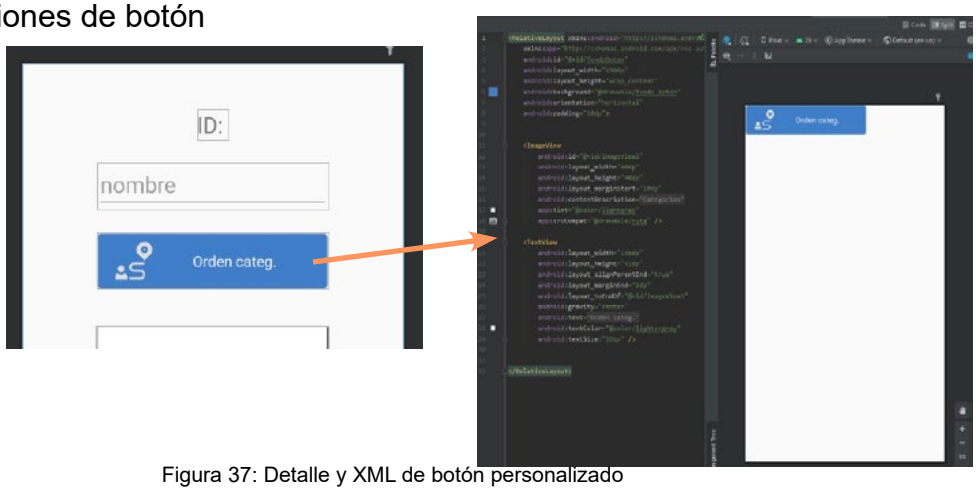


Figura 37: Detalle y XML de botón personalizado

-Options menu (36)

Aplicado para añadir funcionalidad tanto para la ordenación de los productos (nombre o categoría), así como en la gestión de tiendas, como alternativa al botón, para acceder a la gestión de orden de categorías.

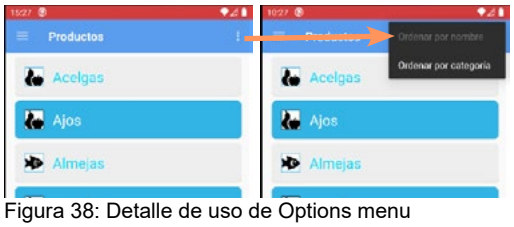


Figura 38: Detalle de uso de Options menu

-FAB (Floating Action Button) (37)(38)

Elemento de Material Design utilizado para la realización de una acción principal clara, en este caso, cuando hay que añadir nuevos elementos, o cuando hay que grabarlos. En esta app, además se les ha aplicado una animación, que al pulsarlos, giren y vuelvan a su posición inicial.

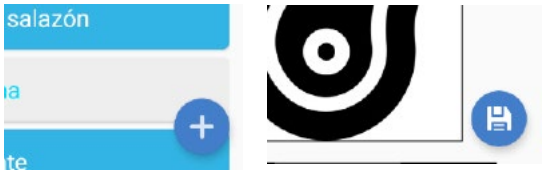


Figura 39: Detalle de uso de Floating Action Button

-Control del botón atrás (39)(40)

Se ha controlado el uso del botón atrás propio de Android, para controlar el flujo entre las diferentes pantallas, de manera que las pantallas de edición, vuelven a su pantalla principal con la lista, y estas vuelven siempre a la pantalla de la lista actual. Además se ha implementado que en la pantalla principal, sea necesario pulsar doblemente el botón atrás para salir de la app.

-Texto como HTML (41)

Por último, en la pantalla "acerca de..." Donde se encuentran los créditos, se ha implementado que el texto aparezca en formato HTML, tal y como se especifica en la licencia de ciertas imágenes, que han de contener un enlace a la página del autor.

Figura 40: Detalle de texto como HTML

2.2.2 Pruebas (42)(43)

El proceso de pruebas propiamente dicho se ha realizado, según se definió en la programación, durante los últimos días de la fase de desarrollo, mediante la función App Distribuition de Firebase, y la colaboración, mediante invitación de conocidos que han hecho las funciones de tester de manera desinteresada.

Gracias a los comentarios de los probadores, los informes de crashlytics se han solucionado diversos bugs i mejorado aspectos de usabilidad y aspecto.
Aún así, antes de este proceso de pruebas "formal", en fases más tempranas de la fase de desarrollo, ya se habían realizado algunas pruebas con conocidos, a los que se enviaba la APK, pudiendo estos hacer pequeñas pruebas, sobretodo durante la programación de la compartir de listas.

Además, y de manera adicional a las pruebas con usuarios, se han programado diversos tests mediante expresso/mockito y Junit para la comprobación del funcionamiento de diversos aspectos de la app, pero por diferentes motivos no han sido provechosos debido a diferentes fallos, desestimándose la corrección de estos debido a lo ajustado del calendario. Queda pendiente la implementación de estas pruebas para posteriores actualizaciones ajenas al ámbito de este proyecto final de Máster.

Gracias a estas pruebas se han encontrado y solucionado algunos bugs, como fallos al logar, errores al mostrar las listas, no arrancar con la lista actual al principio... Además, se han modificado aspectos visuales, como tamaños de letra, o posición de los elementos de pantalla, todo ello siguiendo las indicaciones de los usuarios.

2.2.3 Problemas encontrados

Durante el proceso de programación de la App han aparecido diferentes problemas de muy diferente índole, pudiéndose arreglar la gran mayoría de ellos si mayor complicación, aunque si que hay algunos que han ocupado más tiempo del deseable hasta su resolución (o desestimado). Por fortuna, en la planificación ya se contemplaba la aparición de problemas, con lo que esto no ha supuesto un desvío en cuanto a esta.

Algunos de estos problemas han sido:

-Problemas con Firebase y ReactiveX, ya que la manera en que se recogen los datos estudiadas durante los trabajos realizados en el Máster, no se ajustaban exactamente a lo que se ha necesitado en este proyecto. Desafortunadamente este problema se detectó en las fases finales, durante las pruebas, lo que puso en peligro la planificación, pero finalmente pudo ser solucionado a tiempo, mediante la refactorización del código del RemoteDataSource, aprovechando para pasar de una función para cada uno de los diferentes mantenimientos y listas (12 en total) a tan solo dos, una utilizada para consultas que solo deben devolver el resultado una vez, y la otra que devuelve datos cada vez que estos cambian.

-Problema de conocimientos de Firebase, y el problema anterior viene directamente dado por este. Durante el Máster vimos muy de pasada la obtención de datos de Firebase Realtime Database, pero nada de su organización y administración. Este trabajo ha sido un aprendizaje continuo en este aspecto, observándose una evolución, y habiendo sido necesario cambiar alguna estructura de la BBDD para adecuarla a las necesidades.

-Cargar lista en Recyclerview, El uso de este elemento con Firebase ha sido un continuo de pequeños inconvenientes que he tenido que ir solucionando. Me he encontrado con que cuando se descargan datos y la clave no es un numero, no funcionaba con el *Adapter* programado, con lo cual lo he tenido que solucionar evitando que no haya claves no numéricas. Además, me he encontrado con un problema cuando se eliminan filas "intermedias", ya que al devolverse los resultados al Recyclerview, este no era capaz de deserializar los datos, al no cuadrar con un Array (por los índices), esto ha sido solucionado cambiando que las funciones de Firebase, en vez de devolver un Dataset, recorren este, y crean una lista, la cual deja de tener el inconveniente citado.

-Pequeños cambios en la estructura de datos, Como se ha comentado anteriormente, durante todo el proceso ha sido necesario modificar la estructura de Realtime Database, para adecuarla a una forma de trabajar diferente a la pensada en un principio, como por ejemplo la desaparición de la tabla productos_categorias, para integrarse en la tabla productos.

-Concurrencia, Es un problema potencial que se ha detectado pero no solucionado, es posible que en algún momento dos usuarios creen un mismo ítem, con un mismo Id. Las probabilidades son mínimas, pero existen.

-Problemas con la firma de Google. Un problema que me ha traído de cabeza durante bastante tiempo, ya que parecía aleatorio, pero no lo era. Me encontré con que a veces, no me funcionaba, sin razón aparente, el login con Google. Finalmente, después de muchas vueltas, descubrí que el problema era que al utilizar dos ordenadores diferentes para la programación, cuando compilaba con uno de ellos, la firma no coincidía con la definida en el código, con lo que no permitía que el login se realice de manera correcta. También tuve problemas con la firma a la hora de publicar el release.

Bugs y problemas detectados <u>sin solucionar</u> a la entrega del proyecto

- Posibles problemas con listas compartidas (concurrencia, seguridad, privacidad).
- Posibles problemas de seguridad Firebase, puede requerir rediseño de datos.
- Crashes aleatorios al eliminar filas en recyclerview (bug conocido de Android) (24)

3. Conclusiones

La realización de este proyecto ha resultado muy interesante, he podido profundizar en todo el proceso creativo de la creación de una APP para Android. Desde el nacimiento de la idea, hasta su publicación en la Google Play Store, poniendo énfasis en el desarrollo y la documentación de la APP.

Este proyecto ha sido un buen aglutinador de todo lo aprendido durante el Máster, ya que se han puesto en práctica la gran mayoría de los conocimientos obtenidos en el mismo. Desde el **diseño** en sus primeras etapas, estudiado en *Diseño de Productos Interactivos Multidispositivo*, pasando por el **desarrollo** visto en las asignaturas: *Tecnología y desarrollo en dispositivos móviles* y en las dos de *Desarrollo de Aplicaciones Móviles para Dispositivos Android* (básico y avanzado). Hasta finalizar en la **publicación y promoción** en la tienda de Android, como se vio en *Modelos de Negocio y Márqueting basados en Dispositivos Móviles*.

Durante el proceso me he ido encontrado con dificultades, como en cualquier proyecto de cierta envergadura, pero todas han podido ser superadas, a mi entender, de manera satisfactoria. Quizás las mayores dificultades han venido derivadas de la necesidad de profundizar en los conocimientos del funcionamiento de Realtime Database de Firebase, ya que durante el Máster se ha visto ampliamente el acceso a datos, pero nada de diseño y administración. Lo cierto es que he echado en falta estos conocimientos, he tenido que aprender durante el proceso, lo que me ha supuesto no pocas rectificaciones y cambios en el programa desde la idea inicial.

En cuanto a los objetivos planteados al principio del proyecto, el nivel de cumplimiento ha sido muy satisfactorio, cumpliéndose al 99% los objetivos principales, tan solo quedando excluida la implementación de persistencia de datos en local, por decisión propia, ya que durante la fase de estudio e investigación se observó que no era una funcionalidad necesaria, ya que Firebase se ocupa de manera nativa de la gestión de los datos en momentos sin conexión. En cuanto a los objetivos secundarios, también se han visto cumplidos en su mayoría, excepto el uso de huella dactilar, que con la implementación de login con Google dejaba de tener sentido, y la utilización de mapas, y descarga de datos mediante rest API, excluidos en este caso por no encontrar una funcionalidad realmente útil para ellos dentro de la App. El caso de implementar pruebas de testeo ha sido descartado antes de finalizar, por falta de tiempo.

En conclusión, este ha sido un proyecto en el que se han podido poner en práctica las competencias obtenidas durante el Máster, y en el que se han cumplido con los objetivos planteados al inicio del mismo, concluyéndose de manera altamente satisfactoria, resultando un producto, la app Shoppath, con posibilidades de crecer y evolucionar, pudiendo llegar a convertirse en un producto comercial.

3.1 Propuestas de mejora

Durante el proceso de desarrollo de la aplicación, sobretodo con las pruebas con usuarios, han ido apareciendo funcionalidades y posibles mejora que por motivos de tiempo no se han podido incluir dentro del alcance de este proyecto, pero que resultan interesantes para el posible futuro comercial de la app.

Algunas de estas posibles mejoras son:

- Mejorar la estructura de los datos en Firebase con el fin mejorar la seguridad y la escalabilidad de los datos.

- Mejorar el proceso de compartir listas: pedir confirmación al destinatario para que acepte la lista antes de que le aparezca en su app.

- Mejorar el proceso de compartir listas: poder elegir desde lista de contactos del usuario.

- Mejorar el proceso de eliminación de productos de la lista: confirmación previa o poder retroceder la eliminación.

- Mejorar el proceso de eliminación de productos de la lista: agregar lista de productos eliminados en la pantalla de lista de la compra.

- Mejoras visuales en tablet (10" y 7").

- Gestión de concurrencia, pueden producirse problemas si dos usuarios acceden a los mismos datos en el mismo momento.

4. Glosario

- **Activity**: Pantallas o vistas que forman la APP.
- **AdMob**: Sistema de monetización para Apps de Google.
- **Android**: Sistema operativo de Google. Se utiliza principalmente para dispositivos móviles, pero también se integra con wearables, Tv, etc.
- **Android Studio**: Entorno de desarrollo para Android propiedad de Google, potenciado por JetBrains
- **API**: Application Programming Interface. Conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores simplificar el trabajo de programación, permitiendo integrar en nuevos programas código preexistente.
- **APK**: Archivo instalable de la plataforma Android.
- **APP**: Application. Aplicación, sobretodo en el entorno de tecnologías móviles.
- **App híbrida**: App que se crea con HTML5 y una única interfaz. Posteriormente se compila dentro de un contenedor nativo para lanzarse a través de la plataforma
- **App nativa**: App desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante.
- **BAAS**: Backend AS A Service. Modelo para proporcionar a desarrolladores una serie de servicios en la nube.
- **Backend:** Capa de acceso a datos, parte de la aplicación de los servicios del servidor.
- **BBDD**: Base de datos. Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso
- **Benchmarking**: Evaluación de productos de la competencia con el objetivo de detectar puntos fuertes y características.
- **Bug**: bicho. Fallo en el código de programación
- **CardView**: Implementación visual en forma de tarjeta de información.
- **Clave**: Parte de un nodo JSON.
- **Cloud Firestore**: BBDD de Firebase.
- **Crash -** Error. Cierre abrupto del programa
- **Crash reporting**: Informe de errores.
- **DADA**: Asignatura de Desarrollo de aplicaciones móviles para dispositivos Android.
- **DAADA**: Asignatura de Desarrollo avanzado de aplicaciones móviles para dispositivos Android.
- **Dashboard**: Diseño de interfaz gráfica que ubica en una sola pantalla las secciones más importantes.
- **DataSnapshot**: conjunto de datos obtenidos de Realtime Database.
- **Demo**: Demostración.
- **Diagrama de Gantt**: Gráfica que muestra las tareas de un proyecto en el tiempo. Se indican tareas e hitos a completar.
- **Dropbox**: Almacenamiento en la nube.

- **Emulador**: Aplicación que emula, dentro de un ordenador y de manera virtual algún dispositivo móvil.
- **Esbozo**: Diseño simplificado, normalmente hecho a mano con lápiz y papel.
- **Firebase**: Conjunto de servicios Backend en la nube (base de datos, notificaciones, identificación...). Propiedad de Google.
- **Fragment**: Porción del interfaz de usuario que se integra en un activity.
- **Github**: Sistema de gestión de proyectos y control de versiones de código.
- **Google Play**: Tienda de APPS de Google.
- **Hardware**: Parte física, tangible, de un sistema informático.
- **IDE**: Integrated Development Environment. Entorno de desarrollo integrado.
- **Interfaz de usuario**: Medio con el que un usuario se comunica con una máquina, equipo, computadora o dispositivo.
- **JSON**: JavaScript Object Notation. Formato de texto para el intercambio de datos.
- **Kotlin**: Lenguaje de programación, desarrollado principalmente por JetBrains, utilizado (junto al lenguaje Java) de forma predeterminada por Android Studio.
- **Librería**: Conjunto de código que se pone a disposición de los programadores para implementarlo en sus APPs.
- **Licencia de software**: Permisos para la utilización de un software.
- **Material Design**: Concepto, filosofía y pautas enfocadas al diseño en Android.
- **MVC**: Modelo Vista Controlador. Patrón de diseño que separa datos, lógica e interfaces.
- **MVP**: Modelo Vista Presentador. Patrón de diseño que separa datos, lógica e interfaces.
- **MVVM**: Modelo-Vista Vista-Modelo. Patrón de diseño que separa datos, lógica e interfaces.
- **MVI**: Modelo Vista Intent. Patrón de diseño que separa datos, lógica e interfaces.
- **Mysql**: BBDD de código abierto-
- **Navigation drawer**: Menú lateral deslizante.
- **Nodo**: Parte de un árbol JSON.
- **NoSQL**: Not Only SQL. No solo SQL. BBDD no relacional de gran rendimiento.
- **Nube, la**: Almacenamiento o funcionalidades en Internet.
- **On-line**: En línea, conectado.
- **PEC**: Prueba de evaluación continuada.
- **Progress bar**: Barra de progreso, en una APP indica que está realizando un proceso.
- **Prototipo**: Modelo con aspectos funcionales, estructurales y gráficos.
- **Readme**: Léeme. Archivo de texto con aclaraciones que es conveniente leer.
- **Realm**: BBDD para almacenamiento de datos en local.
- **Realtime Database**: BBDD de Firebase.
- **Recyclerview**: Lista que muestra los datos que se van reciclando cuando ya no son visibles por el scroll de la vista.
- **Rxjava**: es una implementación para java de ReactiveX, que es una API que facilita el manejo de flujos de datos y eventos de manera asíncrona.

Índice

- **Smartphone**: Teléfono inteligente. Teléfono celular con pantalla táctil, que permite conectarse a Internet, ejecutar apps y recursos a modo de pequeño ordenador.
- **Splash screen**: Pantalla de inicio o bienvenida de una app.
- **SQL**: Standard Query Languaje. Lenguaje de consulta estándar. Se utiliza para interrogar BBDD's.
- **SQLite**: BBDD para almacenamiento de datos en local.
- **Storage**: Almacenamiento.
- **Swipe**: Deslizar. Arrastrar un elemento de pantalla para realizar una acción.
- **Tablet**: Tableta. Dispositivo móvil, similar a un smartphone pero de mayores dimensiones y mas enfocado a la ejecución de apps que a la comunicación telefónica.
- **TDDM:** Asignatura Tecnología y Desarrollo en Dispositivos Móviles.
- **Tester:** Provador. Persona que utiliza la app en búsca de fallos y mejoras.
- **TFM**: Trabajo final de Máster.
- **UI**: User interface. Interfaz de usuario.
- **Viper**: un tipo de patrón de diseño.
- **Wireframe**: Representación del marco esquelético del diseño visual de una app o página web.

5. Bibliografía

-Clarisó Viladrosa, Robert. Introducción al trabajo final. PID_00197259. FUOC. [fecha de consulta: septiembre de 2020]

-Sáenz Higueras, Nita. Vidal Oltra, Rut. Redacción de textos científico-técnicos. P08/89018/00445. FUOC. [fecha de consulta: septiembre de 2020]

-Flamarcich Zampalo, Jordi. Arquitectura y wireframes . PID_00245396. FUOC [fecha de consulta: octubre de 2020]

(1) Flamarich Zampalo, Jordi. Conceptualización. PID_00245395. FUOC. [fecha de consulta: septiembre de 2020]

(2) Aris, Papadopoulos. SHOULD DEVELOPERS USE THIRD-PARTY LIBRARIES? Scalable Pah. Marzo de 2018 [en línea] [fecha de consulta: septiembre de 2020] disponible en: < https://www.scalablepath.com/blog/third-party-libraries/ >

(3) Antunes, Arthur. 10 LIBRERÍAS QUE DEBES CONOCER SI ERES DESARROLLADOR Android. Bemobile. Octubre de 2016. [en línea] [fecha de consulta: septiembre de 2020] disponible en: < http://bemobile.es/blog/2016/10/10-librerias-que-debes-conocer-si-eres-desarrollador-Android/ >

(4) Cheng, Yung, Olivares Domínguez, Aldo. Advanced Android App Architecture. Razeware LLC. 2019. [fecha de consulta: septiembre de 2020]

(5) Patrones de arquitectura y diseño de sofware. Desarrollopaginasweb.com.mx. Agosto de 2018 [en línea] [fecha de consulta: septiembre de 2020] disponible en: < https://www.desarrollodepaginasweb.com.mx/patrones-de-arquitectura-de-software/ >

(6) MVP Android. Develapps.com. [en línea] [fecha de consulta: septiembre de 2020] disponible en: < http://www.develapps.com/es/noticias/modelo-vista-presentador-mvp-en-Android >

(7) Miheev, Konstantin. Introduction to Model View Presenter on Android. [Konstantin Mikheev's programming blog] Marzo de 2015 [en línea] [fecha de consulta: septiembre de 2020] disponible en: < http://konmik.com/post/introduction_to_model_view_presenter_on_Android/ >

(8) Hermoza, Fahed. ¿por qué no funciona MVC en Android?. Medium.com. Enero de 2020 [en línea] [fecha de consulta: septiembre de 2020] disponible en: < https://medium.com/@fahedhermoza/por-qu%C3%A9-no-funciona-mvc-en-Android-d0b747a823c0#:~:text=El%20patr%C3%B3n%20MVC%20en%20un,componentes%20no%20est%C3%A1n%20realmente%20separadas >

(9) Mysql documentation. [en línea] [fecha de consulta: Octubre de 2020] disponible en: < https://dev.mysql.com/doc/ >

(10) Robledano, Ángel. Qué es Mysql: características y ventajas. OpenWebinars. septiembre de 2019 [en línea] [fecha de consulta: Octubre de 2020] disponible en: < https://openwebinars.net/blog/que-es-mysql/ >

(11) Firebase permite que los equipos de apps para dispositivos móviles y web alcancen el éxito. Firebase [en línea] [fecha de consulta: Octubre de 2020] disponible en: < https://Firebase.Google.com/?hl=es >

(12) Almacena y sincroniza datos en tiempo real. Firebase. [en línea] [fecha de consulta: Octubre de 2020] disponible en: < https://Firebase.Google.com/products/realtime-database? >

(13) Firebase permite que los equipos de apps para dispositivos móviles y web alcancen el éxito [en línea] [fecha de consulta: Octubre de 2020] < https://Firebase.Google.com/products?hl=es >

(14) Start Integrating Google Sign-In into Your Android App.[en línea] [fecha de consulta: Noviembre de 2020] disponible en: < https://developers.Google.com/identity/sign-in/Android/start-integrating >

(15) Autentica mediante el Acceso con Google en Android [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/auth/Android/Google-signin >

(16)September, Myric. Authenticate Using Google Sign-In (Kotlin + Firebase). Myric September. octubre 2018 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://medium.com/@myric.september/authenticate-using-Google-sign-in-Kotlin-Firebase-4490f71d9e44 >

(17)Puri, Shobhit. How to change the text and theme of Google's Sign-In button on Android?. Dev. octubre de 2017 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://dev.to/shobhit/how-to-change-text-and-theme-of-Google-sign-in-button-Android-6bf >

(18) Autentica con Firebase mediante cuentas con contraseña en Android . Firebase [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/auth/Android/password-auth >

(19) Get started with Firebase Security Rules. Firebase [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/database/security/get-started?utm_source=studio#sample-rules >

(20) Capítulo 22 – Fragments en Kotlin. Curso Kotlin Para Android [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://cursoKotlin.com/capitulo-22-fragments-en-Kotlin/ >

Índice

(21) Cómo crear una lista con RecyclerView. Developer[en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://developer.Android.com/guide/topics/ui/layout/recyclerview?hl=es-419 >

(22)Moore, Kevin. Android Recyclerview Tutorial with Kotlin. Raywenderlich.com. marzo de 2019 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://medium.com/@ipaulpro/drag-and-swipe-with-Recyclerview-b9456d2b1aaf >

(23) Recycler View: Inconsistency detected. Invalid view holder adapter positionViewHolder. StackOverflow [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://stackoverflow.com/questions/35653439/recycler-view-inconsistency-detected-invalid-view-holder-adapter-positionviewh >

(24) Recycler View: Inconsistency detected. Invalid view holder adapter. StackOverflow. Febrero de 2016 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://xbuba.com/questions/35653439 >

(25) Cómo guardar datos de pares clave-valor. Developers [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://developer.Android.com/training/data-storage/shared-preferences >

(26)Android shared preferences using Kotlin. JournalDev [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://www.journaldev.com/234/Android-sharedpreferences-Kotlin >

(27) Diseño Android: Menu lateral con Navigation Drawer y AndroidX. Danielme.com. noviembre de 2020 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://danielme.com/2018/12/19/diseno-Android-menu-lateral-con-navigation-drawer >

(28)SmallAcademy. Navigation Drawer With Fragments (2020) | Part 1/4 | Android Studio Navigation Drawer Tutorial enero 2020[Youtube] [fecha de consulta: noviembre de 2020] disponible en: < https://www.youtube.com/watch?v=U-SenYOBJw9Y >

(29) Cómo localizar tu app. developers [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://developer.Android.com/guide/topics/resources/localization >

(30)Julianshen. CircleTransform.java [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://gist.github.com/julianshen/5829333 >

(31)Kitek. CircleTransform.kt [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://gist.github.com/kitek/0d82f40685727c37d1f91ab103087729 >

(32)Android libray[en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://developer.Android.com/studio/projects/Android-library?hl=es >

(33)Google Analytics. Firebase [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/analytics?hl=es >

(34)Registra eventos. Firebase [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/analytics/events?platform=Android >

(35) <ImageButton> con dos líneas de texto. StackOverflow. febrero de 2019[en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://es.stackoverflow.com/questions/237043/imagebutton-con-dos-l%C3%ADneas-de-texto >

(36)Pervaiz, Atif. Options Menu Fragment - Kotlin. AndroidTutorials. febrero 2019 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://devofAndroid.blogspot.com/2019/02/options-menu-fragment-Kotlin.html >

(37)Floating Action Buttons. Codepath [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://guides.codepath.com/Android/floating-action-buttons >

(38) Creando un FloatingActionButton en Android. DesarrolloLibre[en línea] [fecha de consulta: noviembre de 2020] disponible en:<https://www.desarrollolibre.net/blog/Android/creando-un-floatingactionbutton-en-Android-lib-de-soporte#.X5_3_IhKjcs>

(39) Kotlin ¿Cómo implementar onBackPressed () en Fragments?. CodeLab de Webserveis Enero de 2020 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://dev4phones.wordpress.com/2020/01/16/Kotlin-como-implementar-onbackpressed-en-fragments/ >

(40)Al hacer clic dos veces en el botón atrás para salir de una actividad. Dokry Desarrollo.[en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://www.dokry.com/9269 >

(41)Narayan, Saket. Displaying HTML tags on TextView the right way. What the Html. Skate. Junio de 2017 [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://saket.me/HTML-tags-textview/ >

(42) Firebase App distribuition. Firebase [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://Firebase.Google.com/docs/app-distribution >

(43) Cómo compilar pruebas de unidades locales. Developers [en línea] [fecha de consulta: noviembre de 2020] disponible en: < https://developer.Android.com/training/testing/unit-testing/local-unit-tests >

Índice

Imágenes

https://pixabay.com/es/photos/hombre-carreteras-forestales-1150058/
https://pixabay.com/photos/professor-man-male-people-person-836151/
Pixabay License

https://www.publicdomainpictures.net/en/view-image.php?image=4795&jazyk=ES
CC0 Public Domain

https://www.flaticon.es/icono-gratis/casa-silueta-negra-sin-puerta_20176?term=home&page=1&position=3
https://www.flaticon.es/icono-gratis/deshacer_725004?term=back&page=1&position=5
Flaticon License

6. Anexos

Anexo 1
Benchmarking

Lista de la compra - Listonic

https://play.Google.com/store/apps/details?id=com.l

https://listonic.com/es/

Fortalezas
- Multi-lista.
- Listas compartidas con notificaciones.
- Añadir artículos por voz.
- Lista de productos preexistente.
- Productos agrupados por categorías.
- Fotos de productos.
- Datos en la nube/listas compartidas.
- multilenguaje.

Debilidades
- Selección de artículos un toque.

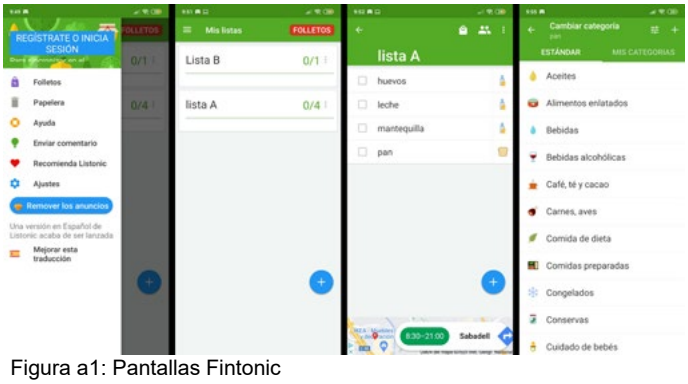


Figura a1: Pantallas Fintonic

Entrada de voz lista compras

https://play.Google.com/store/apps/details?id=com.tksolution.einkaufszettelmitspracheingabe

Fortalezas
- Multi-lista.
- Añadir artículos por voz y por código de barras.
- Selección múltiple de elementos.
- Swipe para eliminar/archivar elementos.
- Gestión de cantidades.
- Productos agrupados por categorías.

Debilidades
- No se pueden ordenar grupos de productos (categorías).
- Categorías sin orden predefinido.
- Sin gestión compartida de listas.
- Selección de artículos un toque.

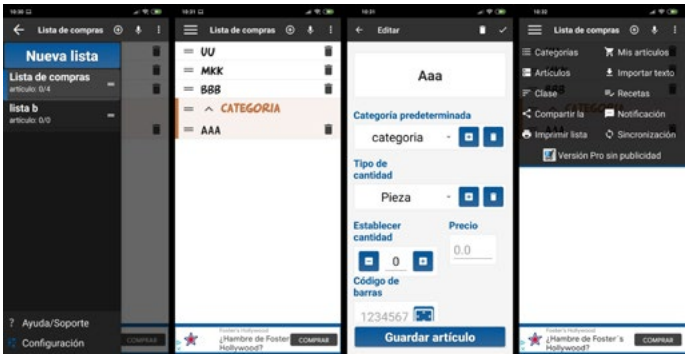


Figura a2: Pantallas entrada de voz lista de la compra

Lista de la compra fácil

https://play.Google.com/store/apps/details?id=com.edujoy.shopping.list

Fortalezas
- Multi-lista.
- Swipe para ordenar elementos.
- Gestión de cantidades.
- Productos agrupados por categorías predefinidas.
- Interfaz muy amigable.
- Multilenguaje.

Debilidades
- Categorías sin orden predefinido.
- Sin gestión compartida de listas.
- Selección de artículos un toque.

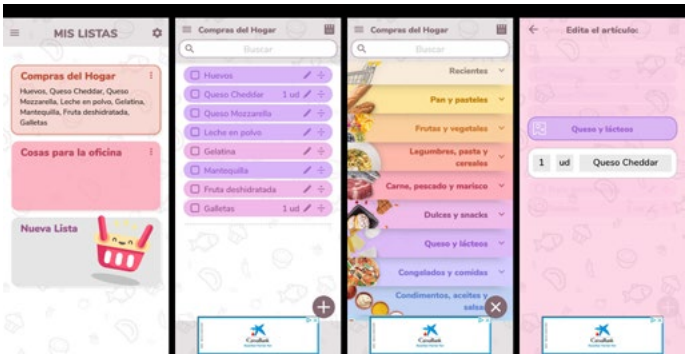


Figura a3: Pantallas lista de la compra fácil

Buy Me a Pie! Lista

https://play.Google.com/store/apps/details?id=com.buymeapie.bmap

Fortalezas
- Multi-lista.
- Gestión de cantidades.
- Multilenguaje.
- Listas compartidas con notificaciones.
- Simple e intuitivo.

Debilidades
- Categorías sin utilidad más que visual.
- Selección de artículos un toque.
- Número de listas reducido.
- No ordena automáticamente.

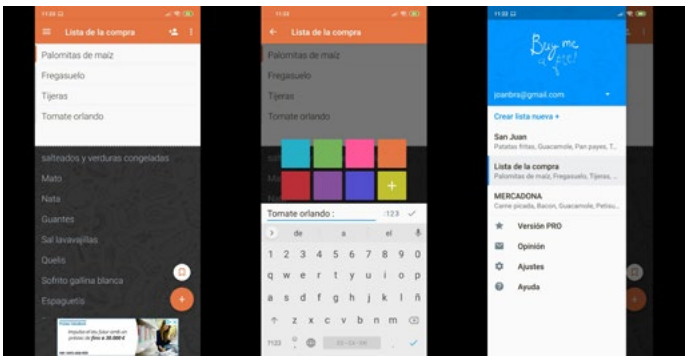


Figura a4: Pantallas Buy me a pie

Google Shopping list

https://shoppinglist.Google.com/

Fortalezas
- Multi-lista.
- Acceso a productos rápidos.
- Listas compartidas con notificaciones.
- Autocompletado al escribir productos.
- swipe completo (aceptar, eliminar, ordenar...).
- Imágenes de los productos.

Debilidades
- Webapp.
- No existen categorías.
- No ordena automáticamente.

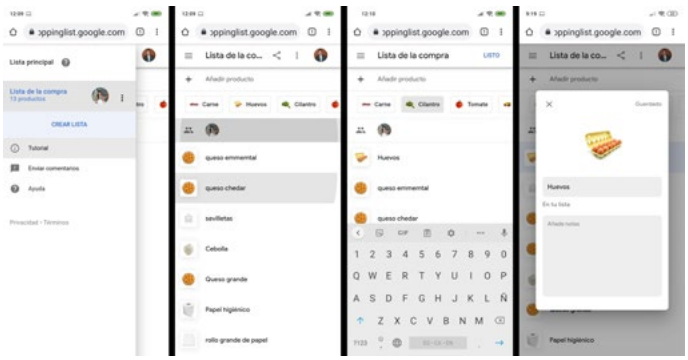


Figura a5: Pantallas Google Shopping List

Bring! Lista de compras

https://play.Google.com/store/apps/details?id=ch.publisheria.bring

Fortalezas
- Compatible asistente Google.
- Imagen de productos.
- Login con Google.
- Listas compartidas con notificaciones.
- Multi-lista.
- Productos agrupados por categorías.
- Categorías predefinidas.

Debilidades
- Categorías sin orden predefinido.
- No ordena automáticamente.
- No uso de swipe.
- Selección de artículos un toque.
- Sin posibilidad de ordenar grupos de productos (categorías).

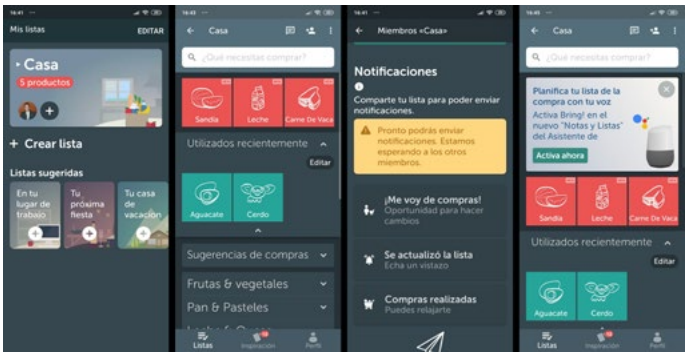


Figura a6: Pantallas Bring!

Out of Milk - Lista de la Compra

https://play.Google.com/store/apps/details?id=com.capigami.outofmilk

Fortalezas
- Añadir artículos por voz y por código de barras.
- Login con Google.
- Listas compartidas con notificaciones.

Debilidades
- Sin categorías.
- Sin ordenación automática.
- Selección de artículos un toque.

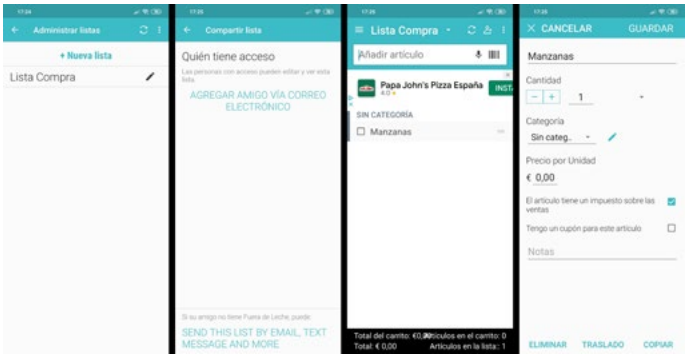


Figura a7: Pantallas Out of Milk

Lista de la Compra - TuLista

https://play.Google.com/store/apps/details?id=es.blackpent.tulista

Fortalezas
- Uso de "Swipe".
- Lista predefinida de artículos.
- Orden automático por categorías al entrar productos.
- Posibilidad de ordenar categorías.
- Aspecto configurable.

Debilidades
- No ofrece gestión compartida de listas.
- Listas compartidas solo como lista de texto.
- Selección de artículos un toque.

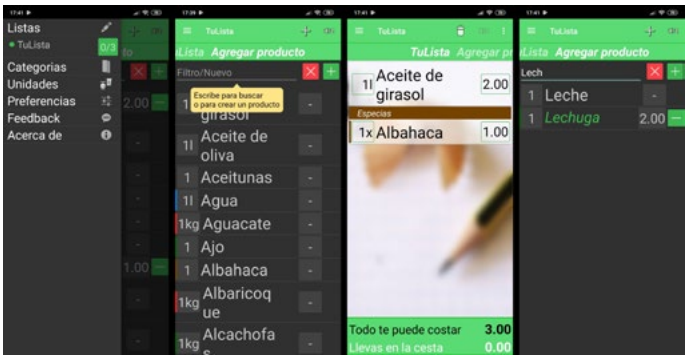


Figura a8: Pantallas TuLista

Lista Compra - ListOn Free

https://play.Google.com/store/apps/details?id=shopping.list.free.lista.compra.gratis.liston

Fortalezas
- interface atractiva.
- Multi lista.
- Lista predefinida de artículos.
- Orden automático por categorías al entrar productos.
- Posibilidad de ordenar categorías.

Debilidades
- Interfaz gráfica demasiado cargada.
- Selección de artículos un toque.

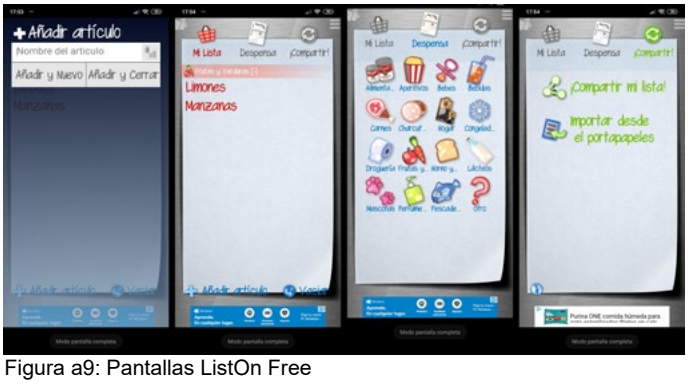


Figura a9: Pantallas ListOn Free

Google Keep: notas y listas

https://play.Google.com/store/apps/details?id=com.Google.Android.keep

Fortalezas
- Gran sencillez.
- Muy intuitiva.

Debilidades
- No es específica para lista de la compra.
- Pocas opciones.

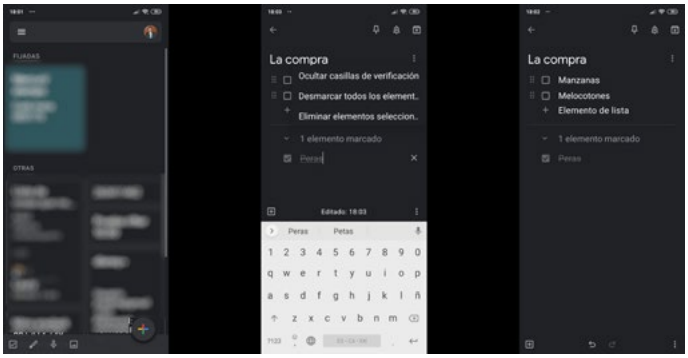


Figura a10: Pantallas Google Keep

Referencia
Índice

Anexo 2
Fichas de usuario

Usuario focal

Alberto Pascual Pastor

Persona:

Alberto es un chico de clase media de 34 años que vive con su novia en Roses, donde trabaja como jefe de ventas. Es licenciado en empresariales por la universidad de Girona, y tiene unos ingresos netos anuales de unos 32.000 €. En casa tiene fibra óptica instalada, pero no utiliza el ordenador o la tablet más que para temas puntuales, le gusta el uso de la tecnología, y el móvil le acompaña allá donde va.

Realiza la compra semanal, habitualmente siempre en el mismo establecimiento y utiliza una app de lista de la compra, pero no está demasiado contento con ella, siente que faltan opciones.

Le gusta salir a pasear con su novia y su perro, ir a conciertos y ver todo tipo de deportes por la TV.

Escenario:

Alberto está en casa, haciendo la comida, y deja la levadura prácticamente acabada, le pide a su novia que apunte, en la app que utilizan, para que no se les olvide cuando vayan a comprar. Unos días después, mientras está leyendo, le apetece un trozo de pastel, pero no tienen, así que decide apuntarlo en la app. Su chica, que ha recibido una notificación de la app, borra el pastel de la lista, y le regaña diciéndole que nada de comer porquerías.
Llega el sábado, y Alberto sale a hacer la compra semanal, mientras su novia María trabaja. Mientras va aceptando los productos que ya ha comprado, María se da cuenta, gracias a las notificaciones que le van apareciendo, que les ha faltado añadir vinagre balsámico, lo añade a la lista y esta avisa a Alberto, que está acabando la compra, y ahora tiene que volver a recorrer el súper para encontrar el vinagre.

Índice

Usuario Secundario

María Fernández Pérez

Persona:

María tiene 23 años, está acabando la carrera de periodismo y vive en Barcelona, en un piso universitario, que comparte con dos amigas.

Junto a sus compañeras de piso comparten los gastos comunes, como son agua, luz, Internet... La compra la gestionan de una manera un tanto especial, ya que cada una compra sus propios caprichos y necesidades, pero artículos de primera necesidad, como leche, servilletas, papel de wc, los van comprando una de ellas compartiendo el gasto, controlan los faltantes con una pizarra blanca que tienen pegada con imán en la puerta de la nevera.

A María no le gusta mucho la tecnología, tiene un móvil, pero no gasta mucho en él, lo usa para llamadas, Whatsapp, y las redes sociales. Dice que al cabo del día miramos demasiado una pantalla, y que lo detesta.

Escenario:

María ha salido a comprar al super de la esquina, ha avisado a sus compañeras de piso, y se ha apuntado en un papel los artículos que hay en la pizarra blanca de la nevera. Cuando lleva la mitad de los productos comprados, recibe una llamada de una de sus amigas, que si le podía comprar chocolate. Le contesta que sin problema y sigue con la compra. A los pocos minutos, mientras inspecciona los ingredientes de un plato combinado, recibe un mensaje por WhatsApp de la otra compañera, que por favor le compre galletas, que se le han acabado. Después de la interrupción, continúa con la compra, aún le faltan un par de artículos de la lista. La repasa y ve que el último de los artículos apuntados es lejía, ¡vaya! Pasó por droguería hace un rato, va a tener que volver sobre sus pasos para buscarla.

Finalmente, se encuentra en la fila, cuando recibe una nueva llamada, son sus compañeras, habían olvidado apuntar el papel higiénico en la lista... María hastiada por tanta interrupción, tiene que abandonar la cola de la caja para ir a buscar el papel de WC.

Volviendo a casa, María se pregunta si no habrá una manera más eficiente de gestionar la compra entre las tres amigas, aunque esto implique utilizar la tecnología que tan poco le gusta.

Índice

Usuario Excluido

Antonino Martínez Díaz

Persona:

Antonino es un jubilado de Rubí, ha trabajado toda su vida como comercial, vendiendo seguros de vida. Trabajó en una época en la que aún no se utilizaban ordenadores, como mucho alguna máquina de escribir para algún informe, con la calculadora, papel y su pluma estilográfica se bastaba, y nunca quiso saber nada de esos aparatos nuevos llamados ordenadores.

No utiliza el móvil, ya que aunque sus nietos le regalaron uno, no lo entiende demasiado, además de que tiene pérdida de visión y no aprecia bien la pantalla, está guardado en un cajón desde la segunda semana.

Le gusta dar largos paseos por su pueblo, ir a ver jugar a fútbol a sus nietos y ver los programas de la tarde en la TV, después de la siesta, que no la perdona. Es aficionado a la pesca y a la lectura.

Escenario:

Antonino baja a comprar al colmado de la esquina cuando tiene alguna necesidad, cuando se le acaba la leche, cuando le apetecen unos huevos con jamón y no le quedan. Si alguna vez olvida algo, vuelve a bajar y lo compra. No utiliza lista de la compra porque no la necesita, y si alguna vez tiene que apuntar, utiliza su vieja libreta y un lápiz.

Referencia
Índice

Anexo 3
Esbozos en papel

- **Pantallas en vertical**

Splashscreen

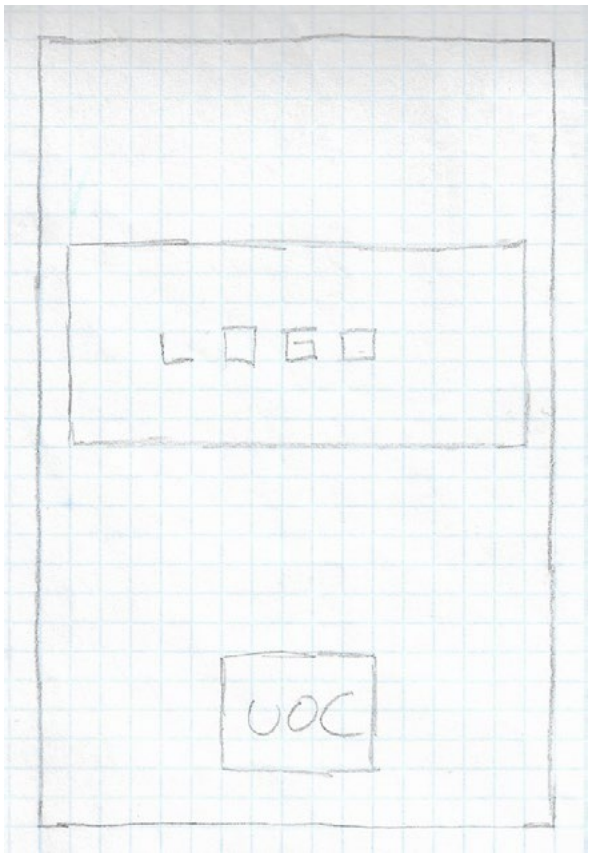


Figura a11: Esbozo pantalla Splashscreen vertical

Login / Alta usuario

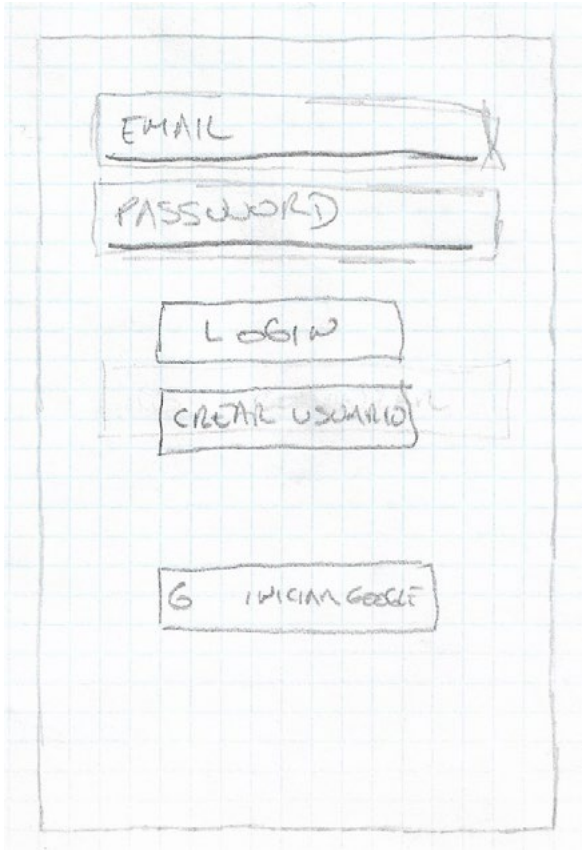


Figura a12: Esbozo pantalla Login vertical

Mantenimientos

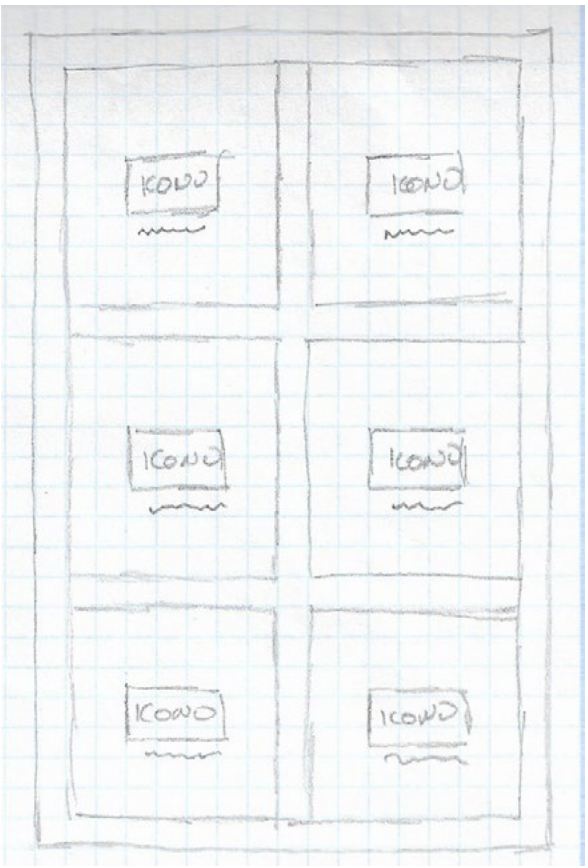


Figura a13: Esbozo pantalla Mantenimientos vertical

Lista actual

Figura a14: Esbozo pantalla Lista actual vertical

Tiendas

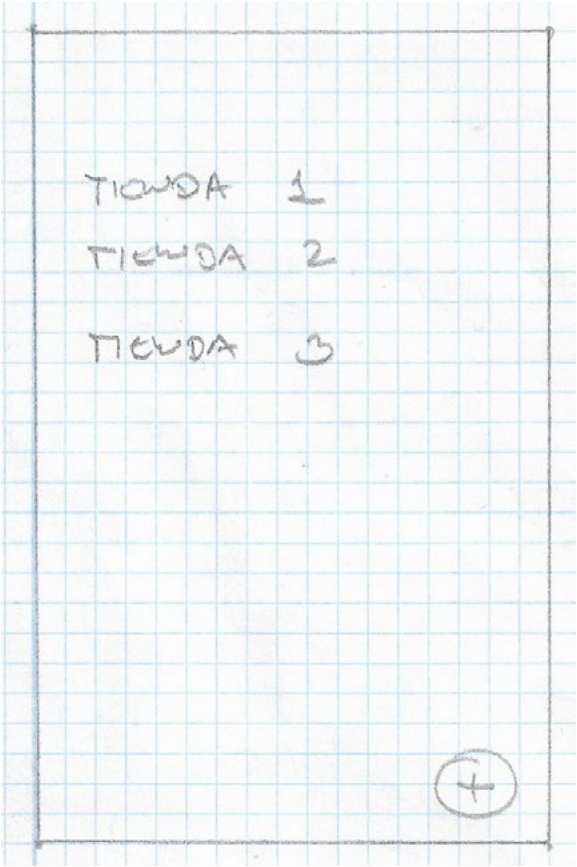


Figura a15: Esbozo pantalla Tiendas vertical

Añadir tienda

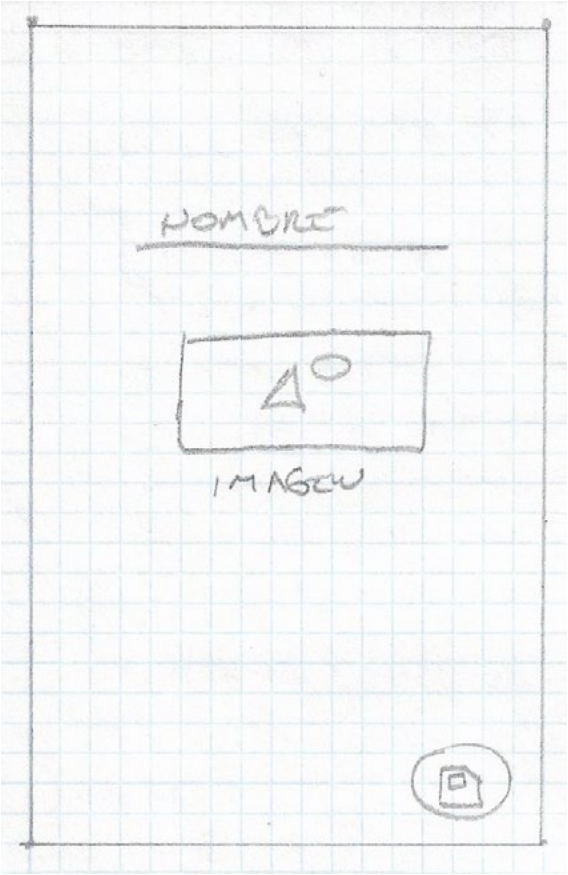


Figura a16: Esbozo pantalla Añadir Tienda vertical

Productos

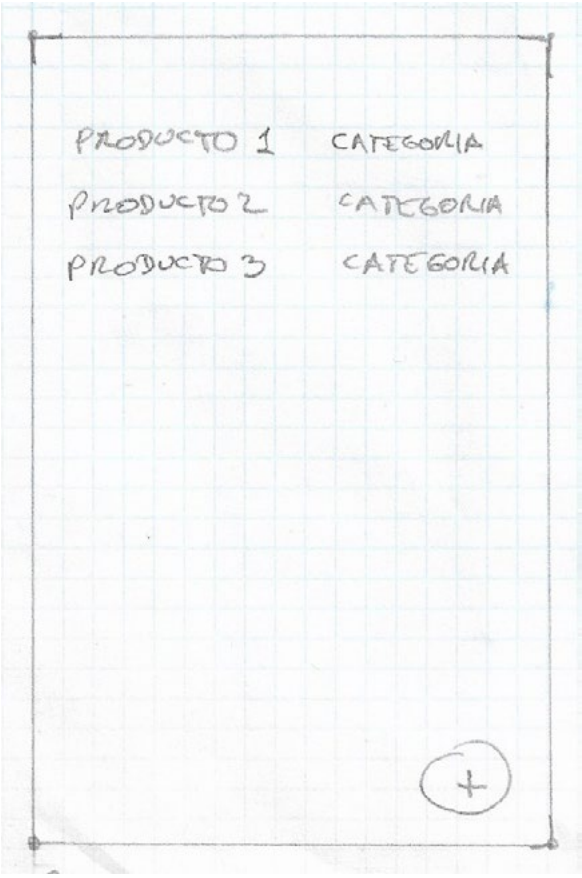


Figura a17: Esbozo pantalla Productos vertical

Añadir productos

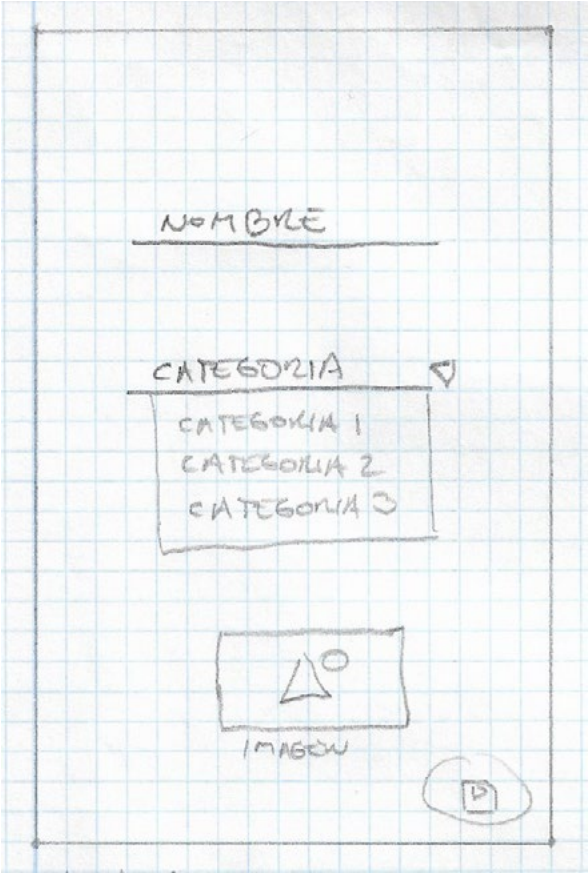


Figura a18: Esbozo pantalla Añadir Productos vertical

Categorías

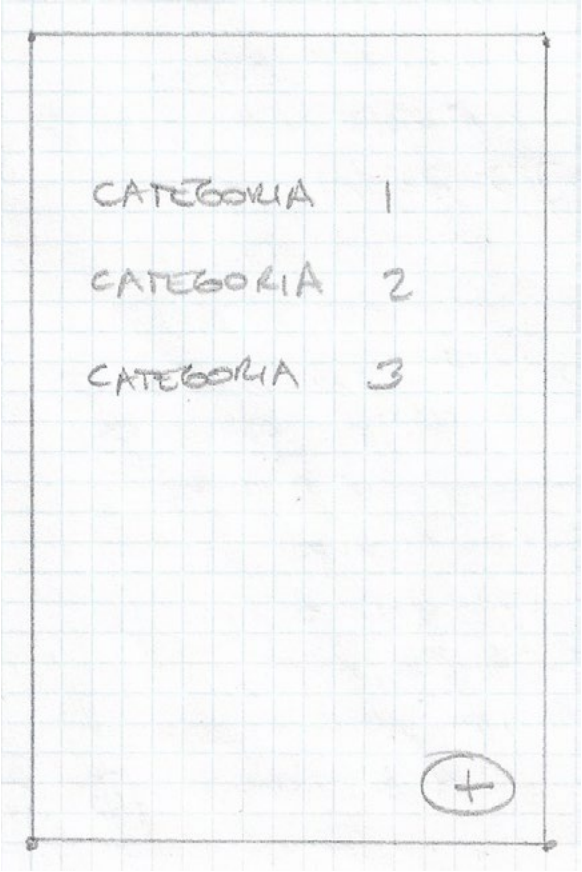


Figura a19: Esbozo pantalla Categorías vertical

Añadir categorías

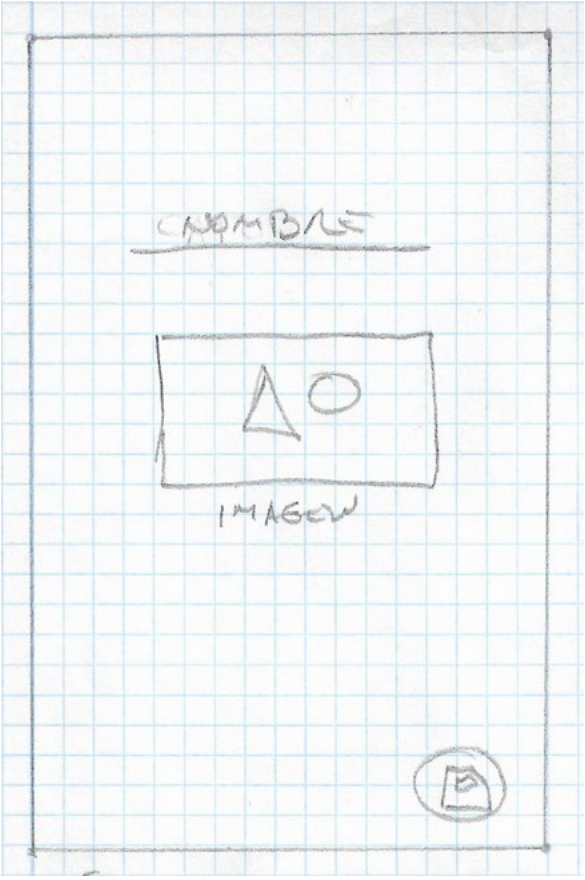


Figura a20: Esbozo pantalla Añadir Categorías vertical

Listas

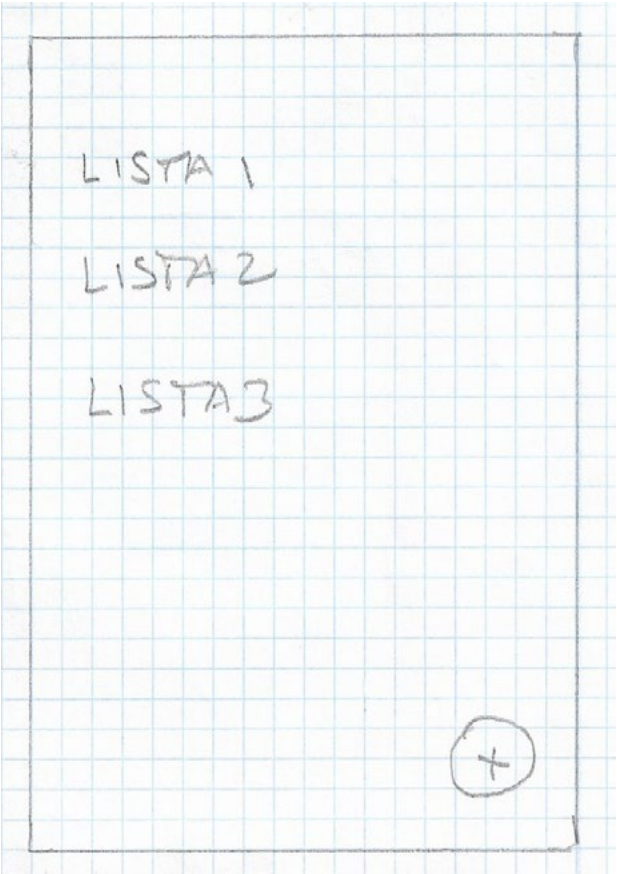


Figura a21: Esbozo pantalla Listas vertical

Añadir lista

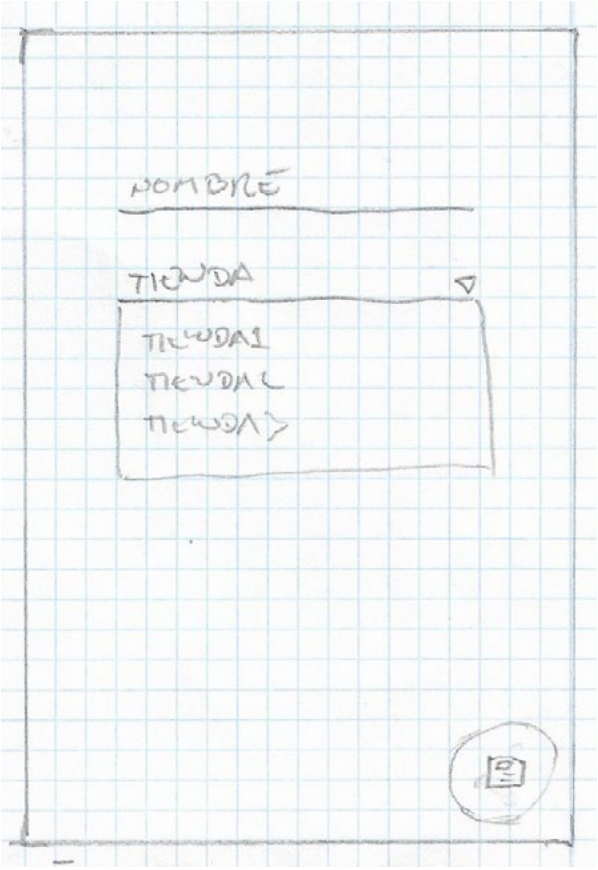


Figura a22: Esbozo pantalla Añadir Listas vertical

Orden categorías

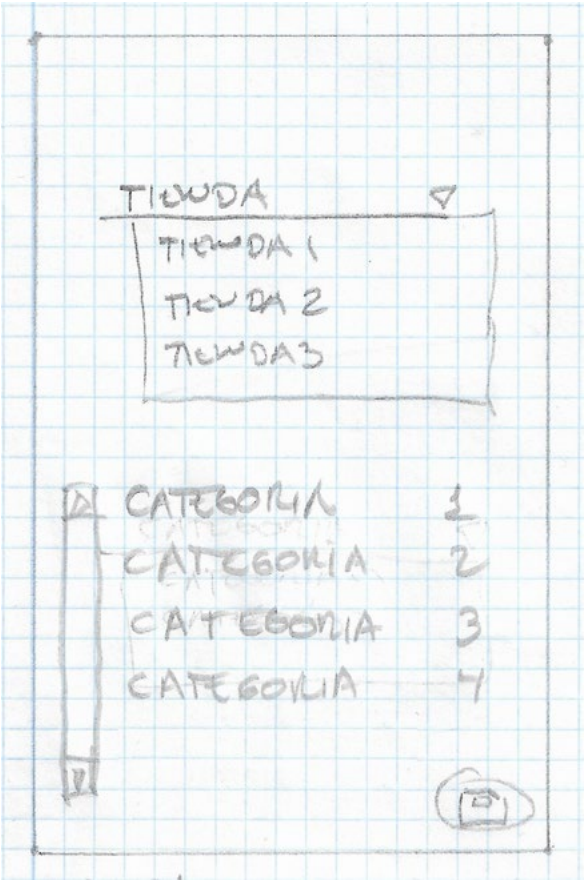


Figura a23: Esbozo pantalla Orden Categorías vertical

Usuario

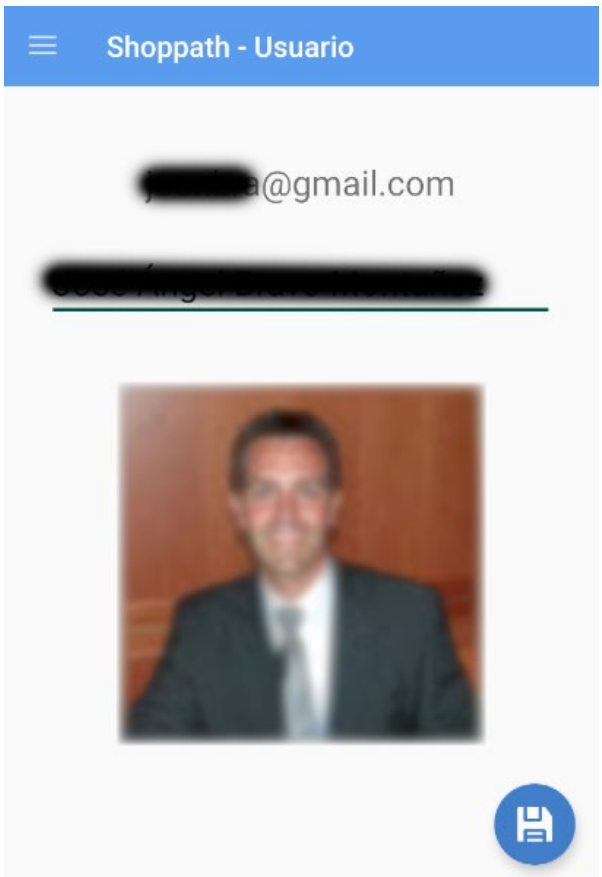


Figura A24: Esbozo pantalla Usuario vertical

Acerca de...

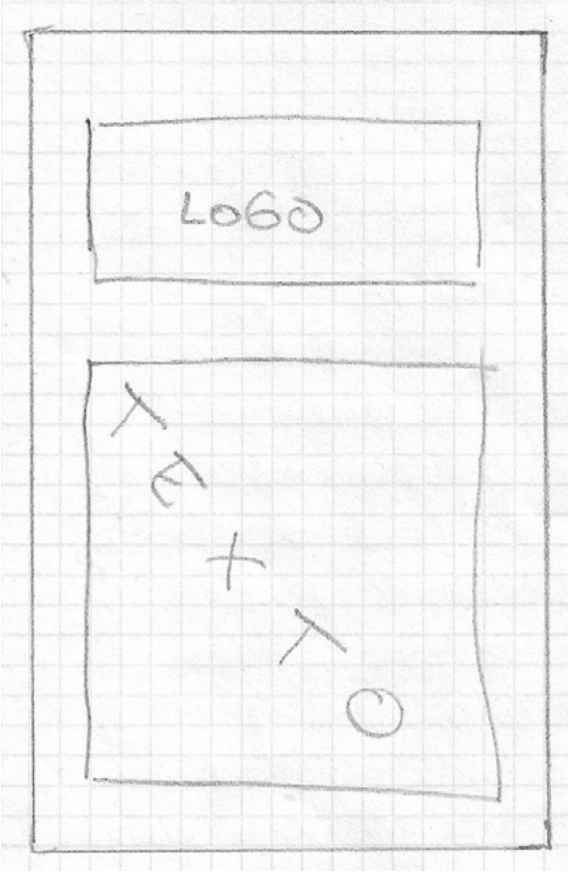


Figura a25: Esbozo pantalla Acerca de vertical

Menú principal

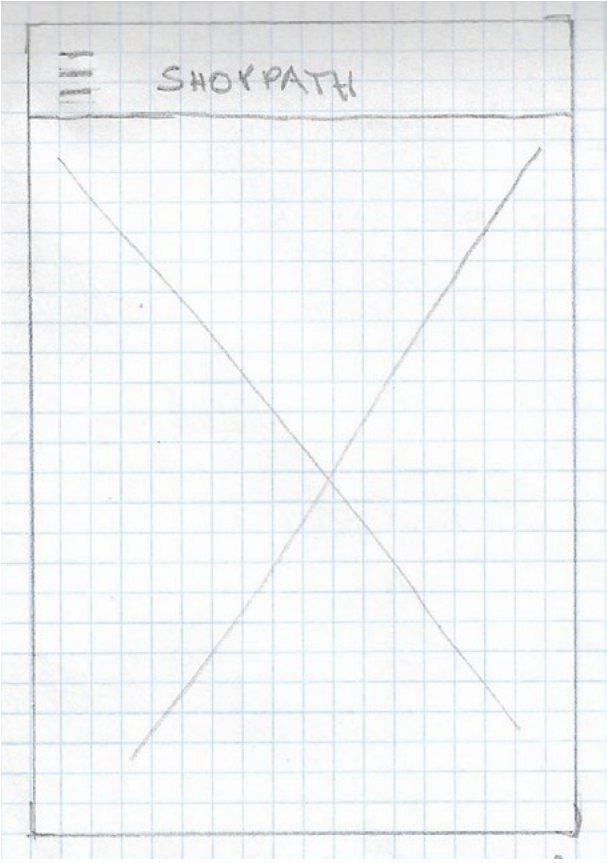


Figura a26: Esbozo pantalla Menú Principal vertical

Menú principal (desplegado)

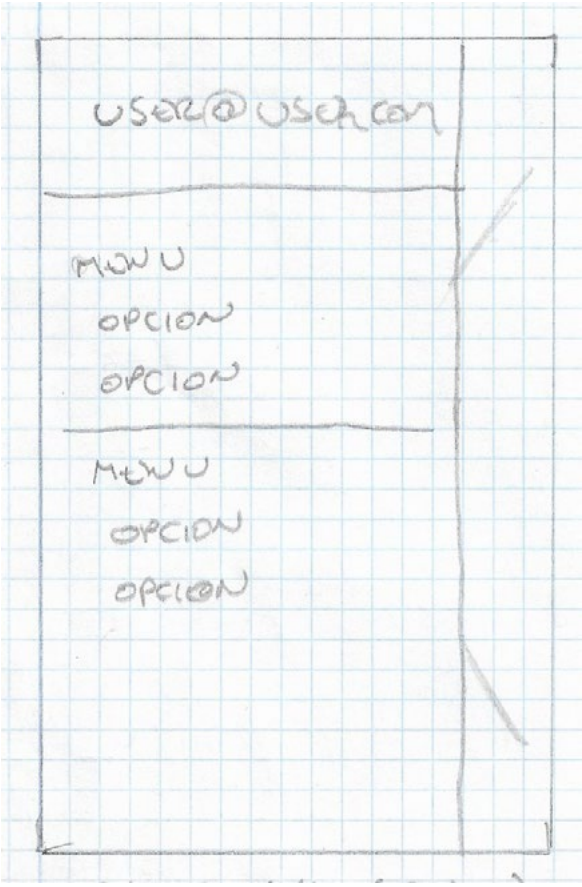


Figura a27: Esbozo pantalla Menú Principal desplegado vertical

- **Pantallas en horizontal/tablet**

Splashscreen

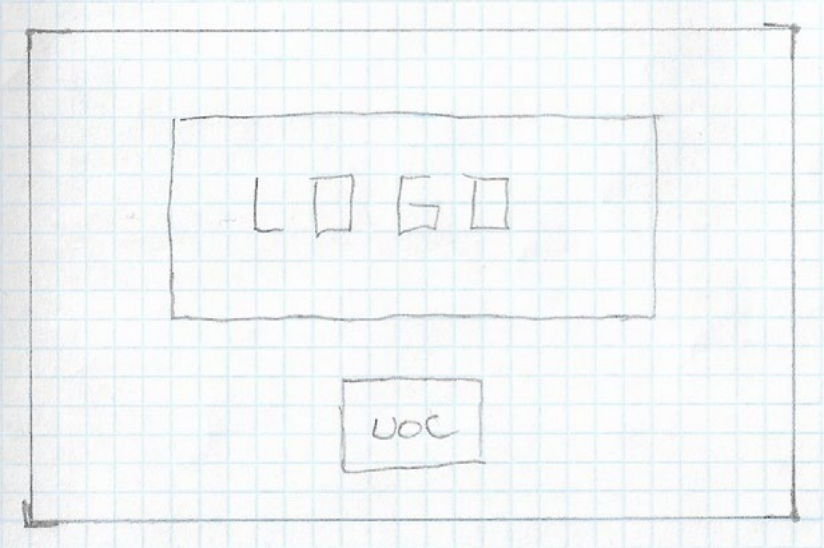


Figura a28: Esbozo pantalla Splashscreen horizontal

Login/alta usuario

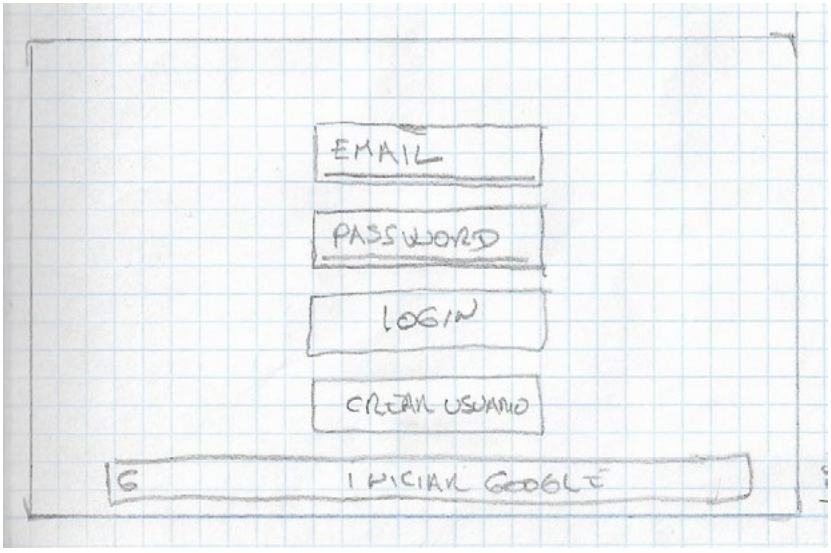


Figura a29: Esbozo pantalla Login horizontal

Mantenimientos

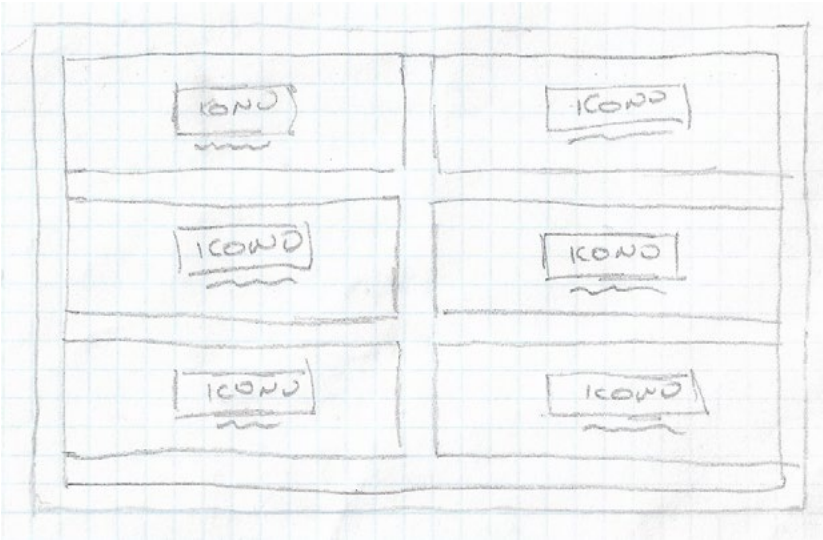


Figura A20: Esbozo pantalla Mantenimientos horizontal

Lista actual

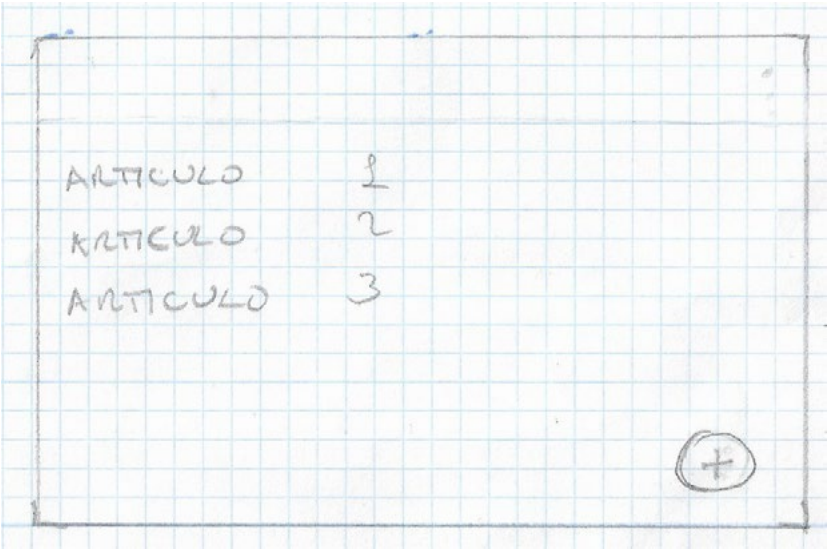


Figura a31: Esbozo pantalla Lista Actual horizontal

Tiendas

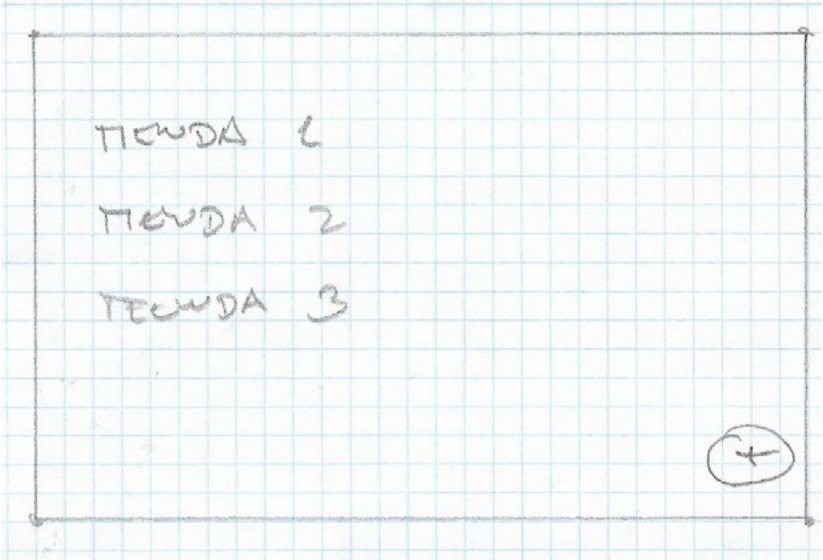


Figura a32: Esbozo pantalla Tiendas horizontal

Añadir tienda

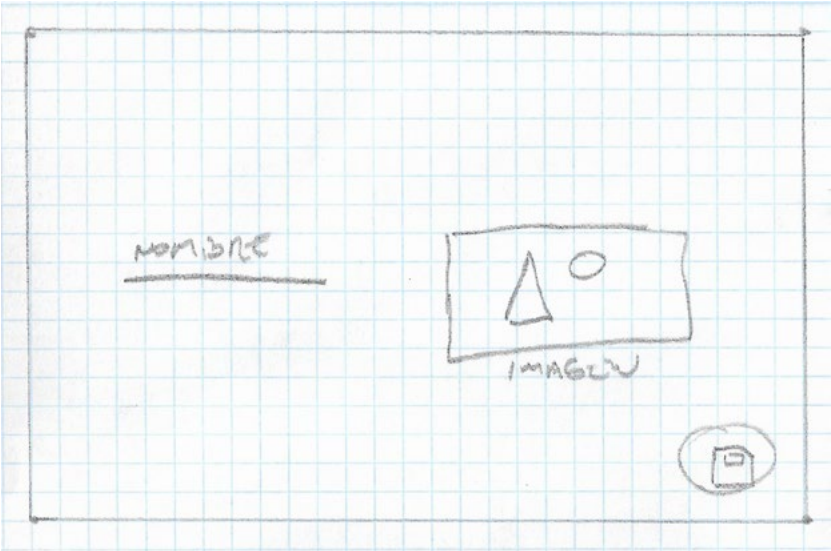


Figura a33: Esbozo pantalla Añadir Tienda horizontal

Productos

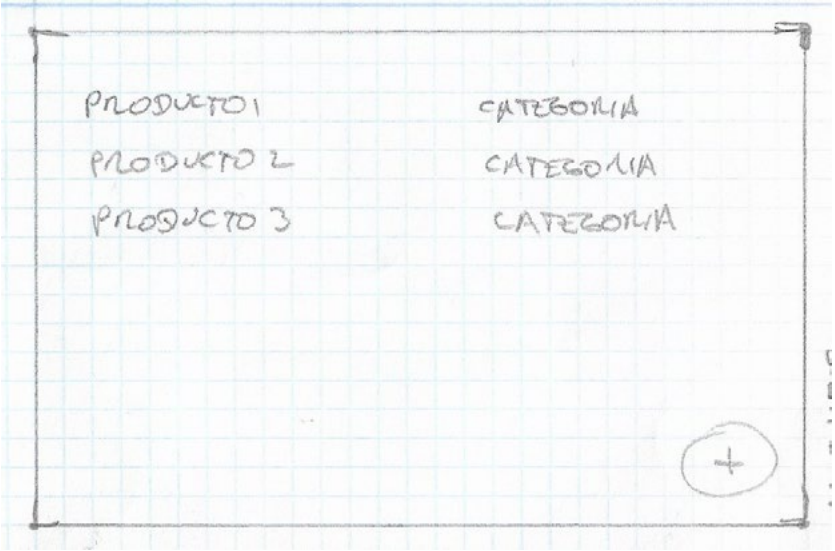


Figura a34: Esbozo pantalla Productos horizontal

Añadir productos

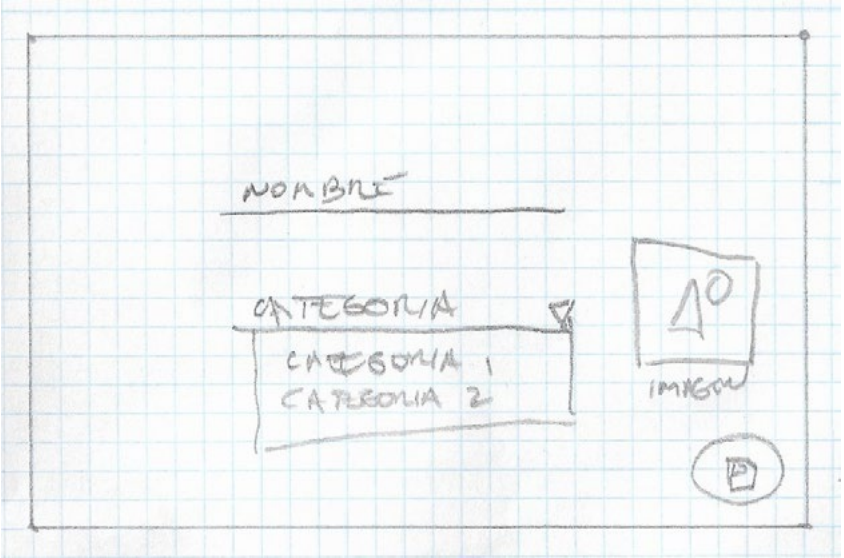


Figura a35: Esbozo pantalla Añadir Productos horizontal

Categorías

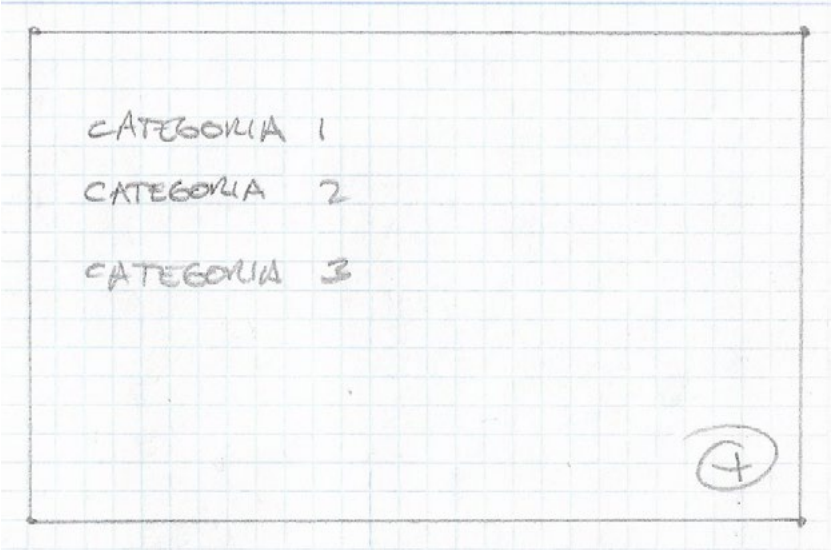


Figura a36: Esbozo pantalla Categorías horizontal

Añadir categorías

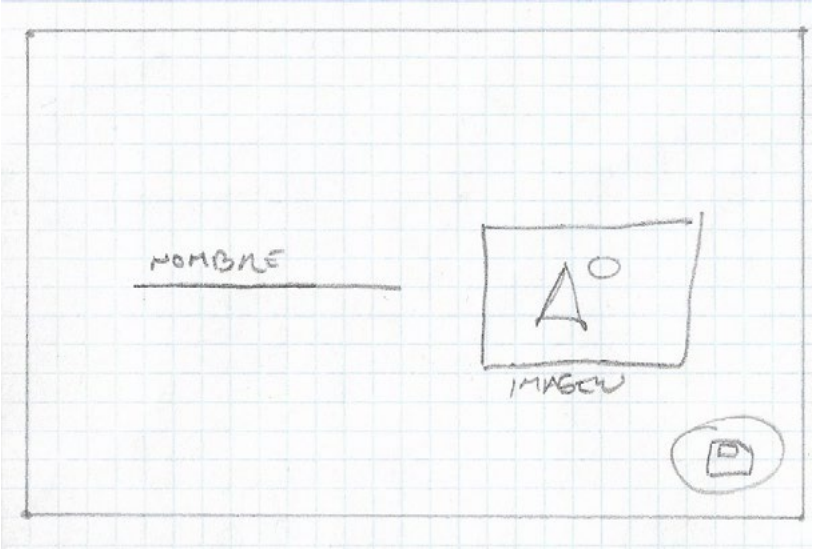


Figura a37: Esbozo pantalla Añadir Categorías horizontal

Listas

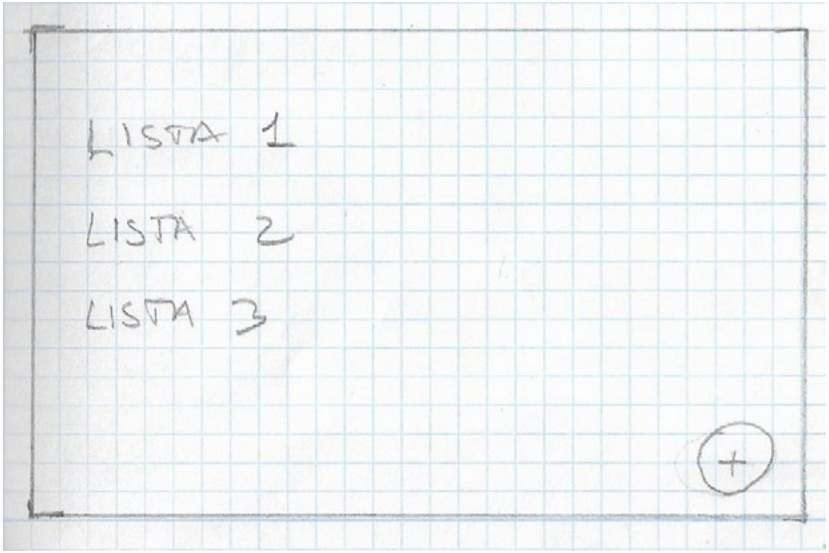


Figura a38: Esbozo pantalla Listas horizontal

Añadir listas

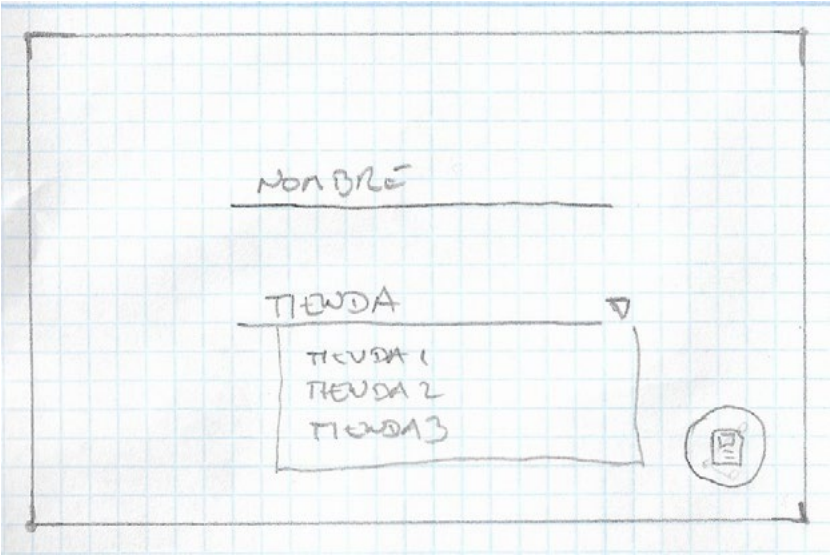


Figura a39: Esbozo pantalla Añadir Listas horizontal

Orden categorías

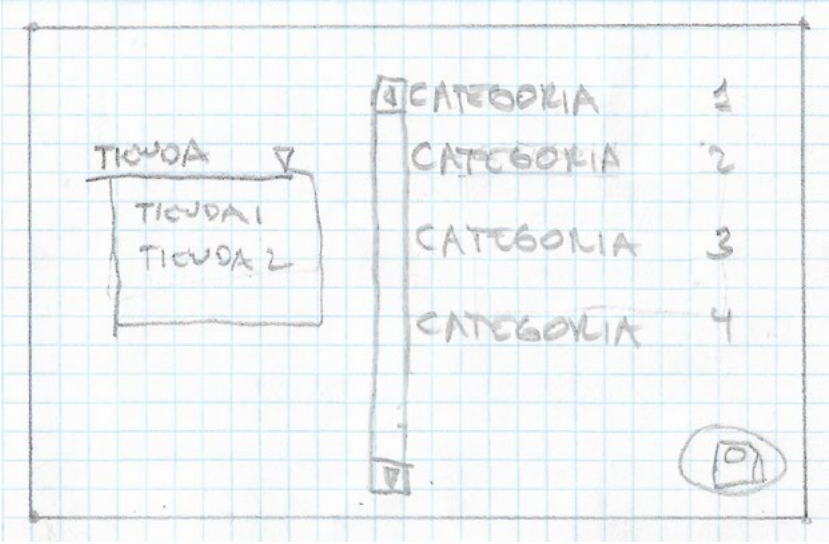


Figura a40: Esbozo pantalla Orden Categorías horizontal

Usuario

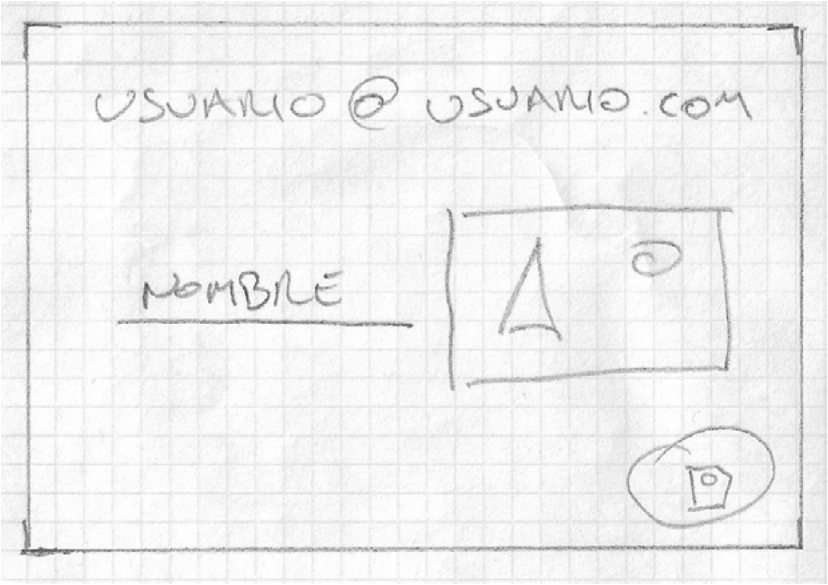


Figura a41: Esbozo pantalla Usuario horizontal

Acerca de

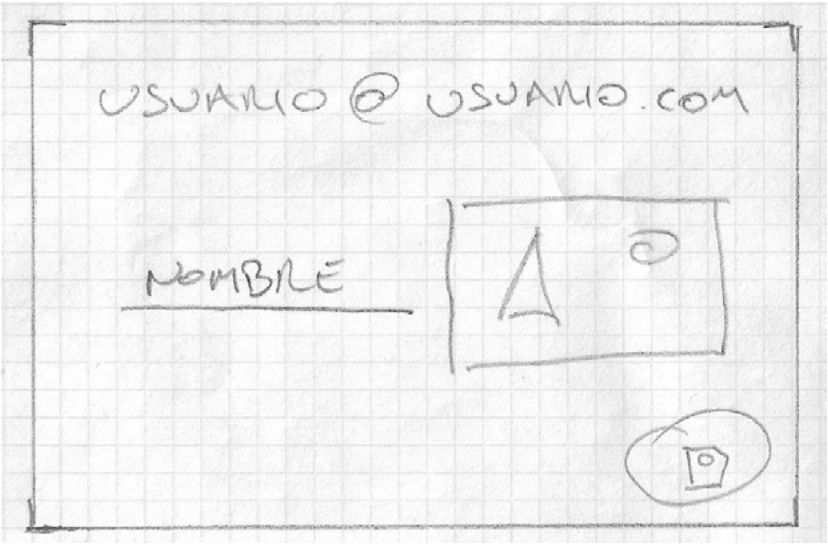


Figura a42: Esbozo pantalla Acerca de horizontal

Menú principal

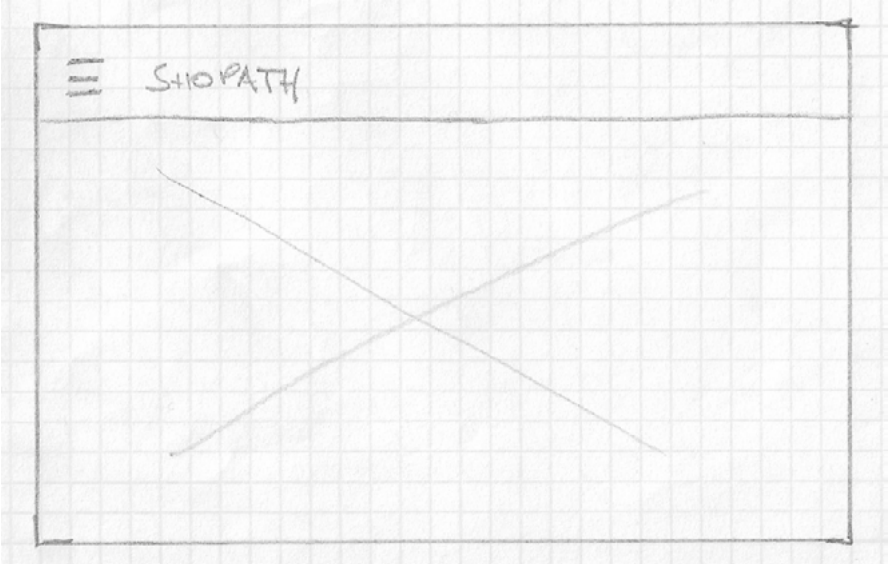


Figura a43: Esbozo pantalla Menú Principal horizontal

Menú principal (desplegado)

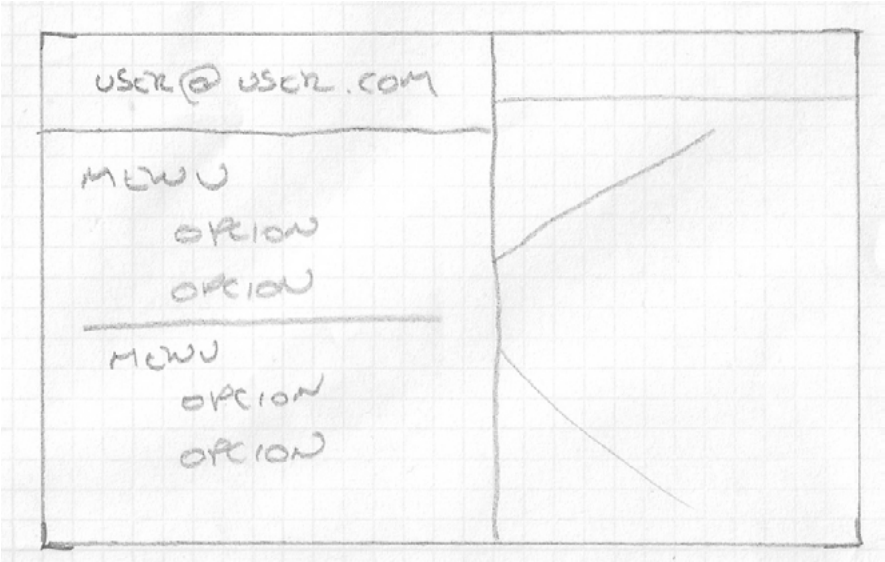


Figura a44: Esbozo Menú Principal (desplegado) horizontal

Anexo 4
Wireframes

- **Pantallas en vertical**

Splashscreen

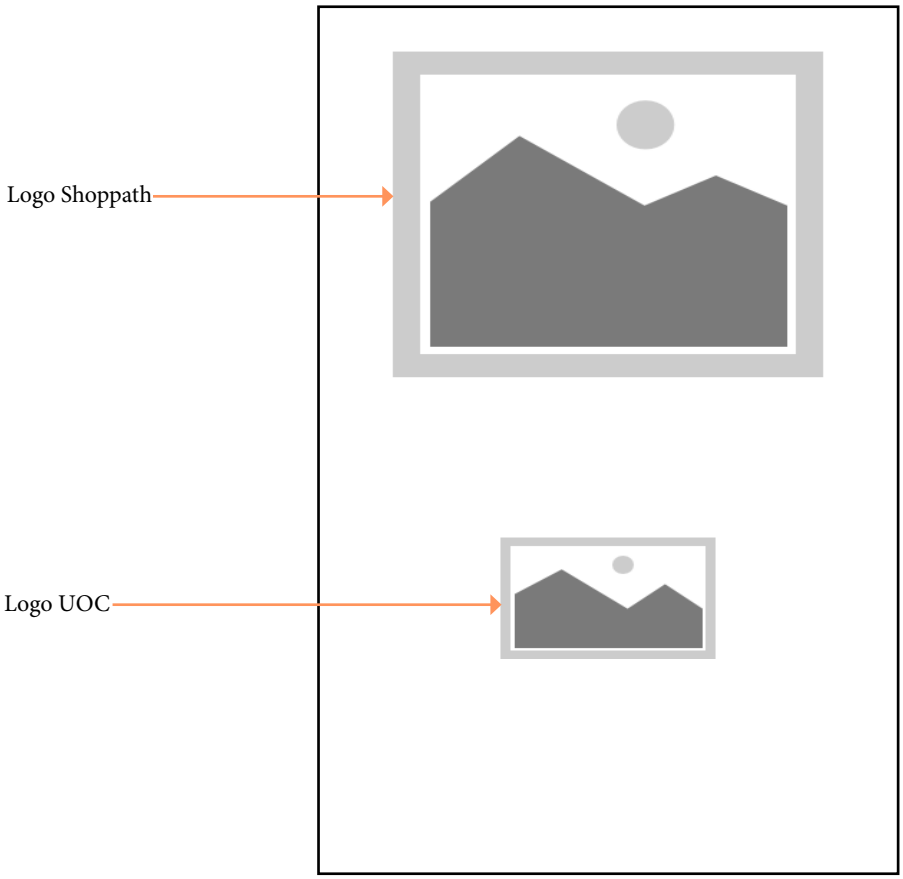


Figura a45: Wireframe pantalla Splashscreen vertical

Login / Alta usuario

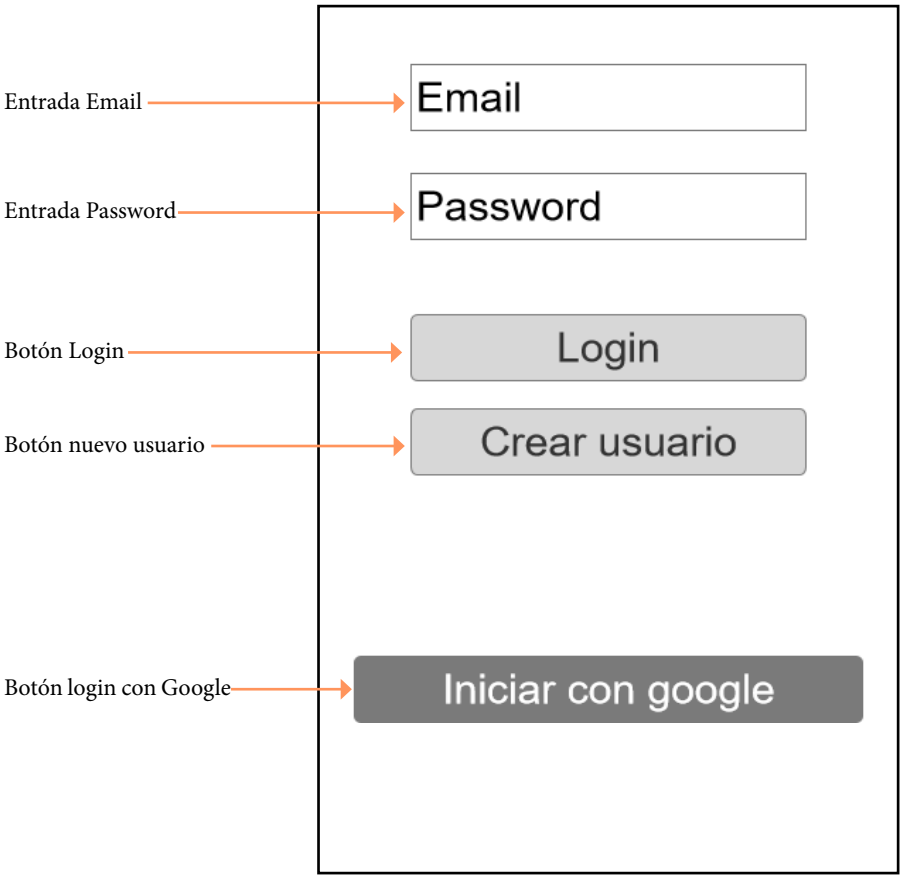


Figura a46: Wireframe pantalla Login/alta usuario vertical

Mantenimientos

Figura a47: Wireframe pantalla Mantenimientos vertical

Lista actual

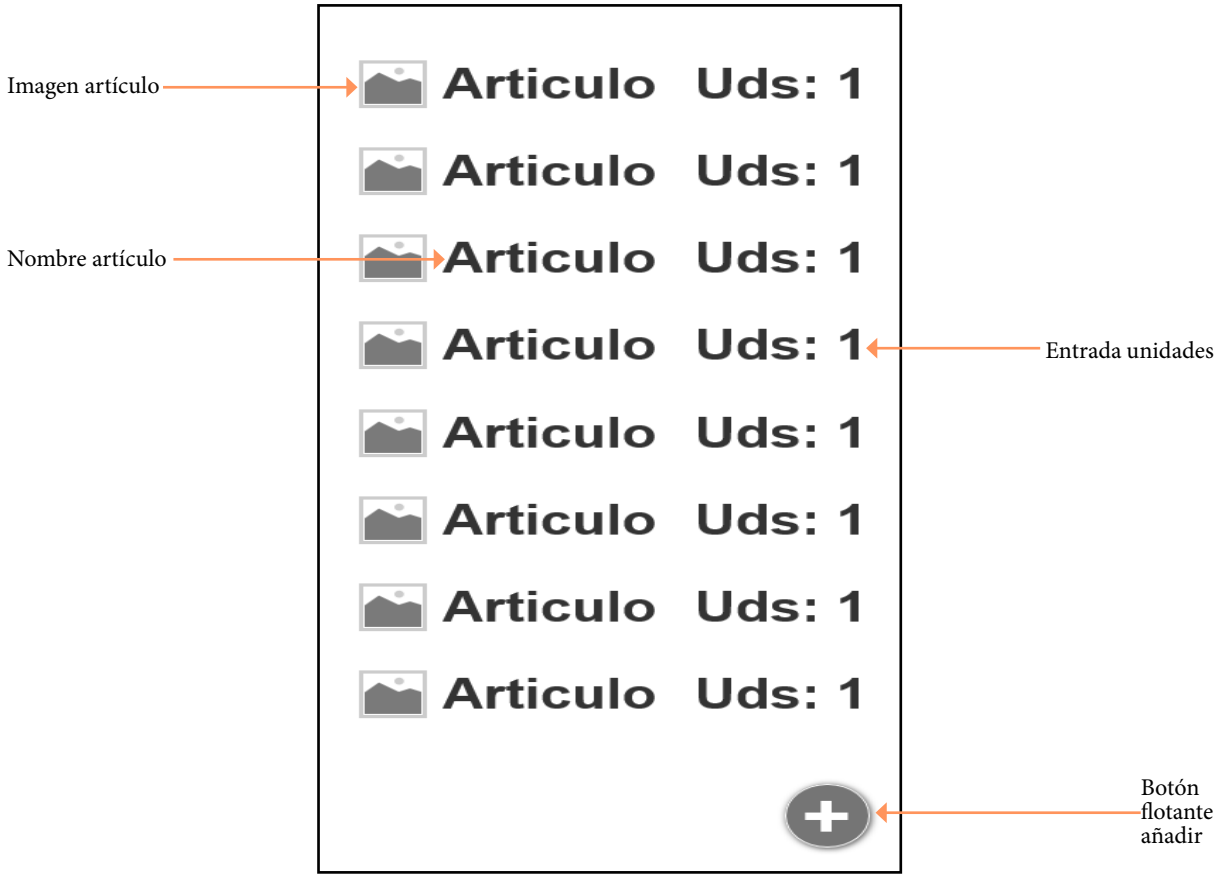


Figura a48: Wireframe pantalla Lista Actual vertical

Tiendas

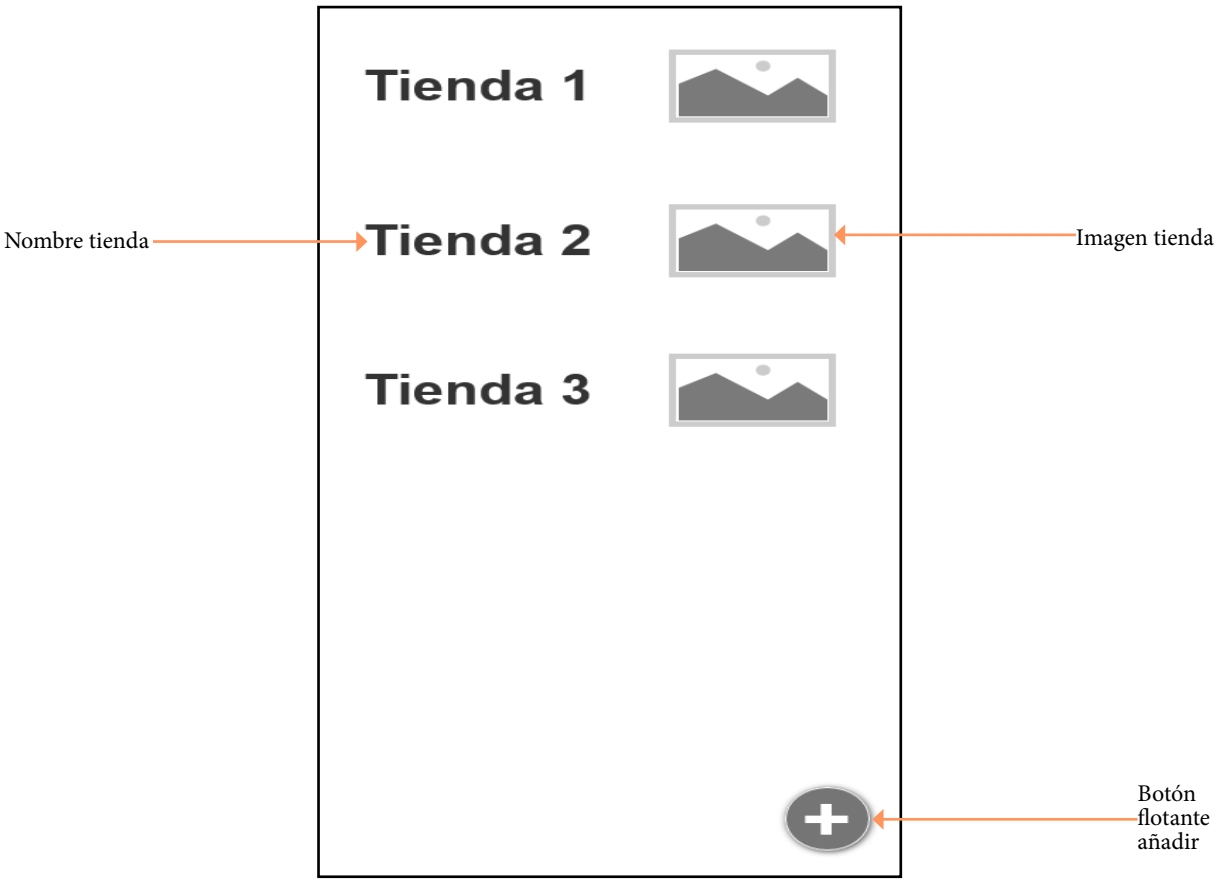


Figura a49: Wireframe pantalla Tiendas vertical

Añadir tienda

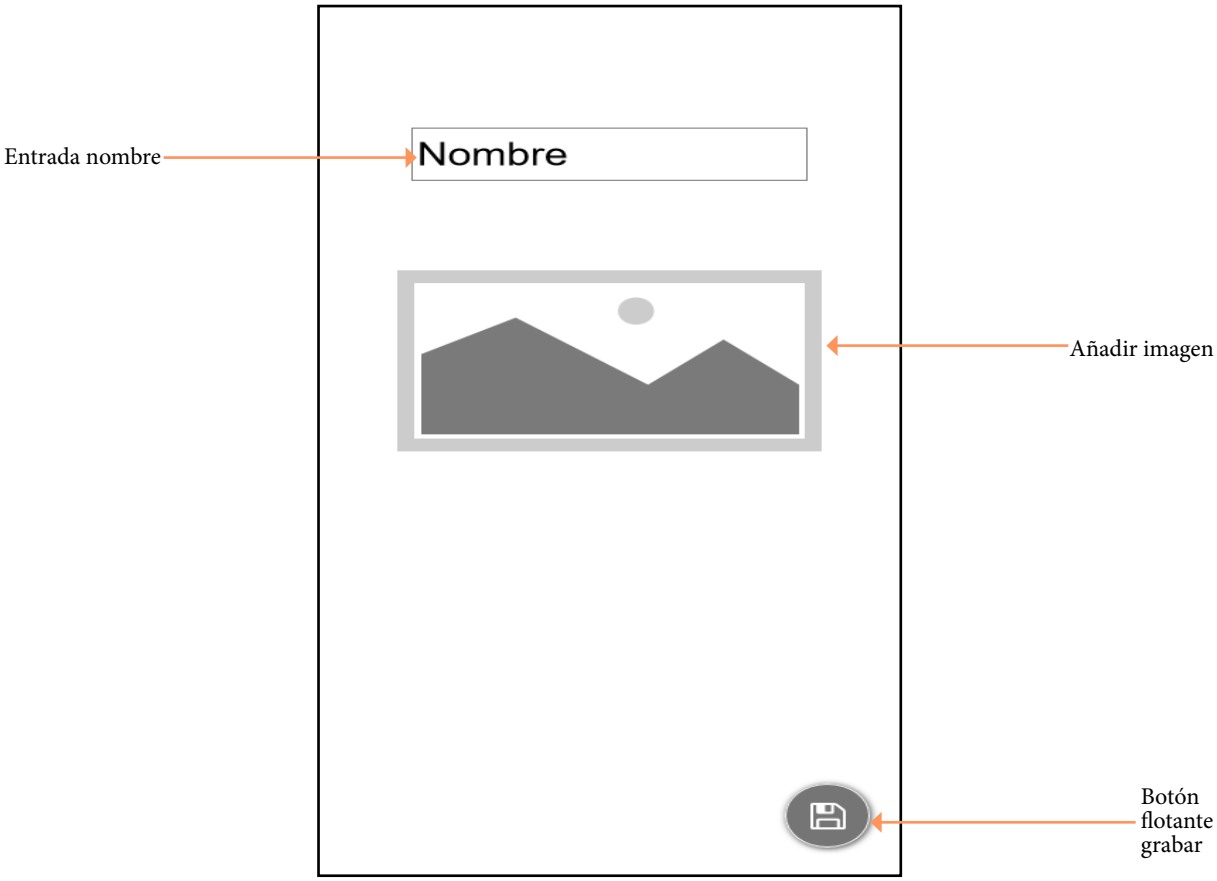


Figura a50: Wireframe pantalla Añadir Tienda vertical

Productos

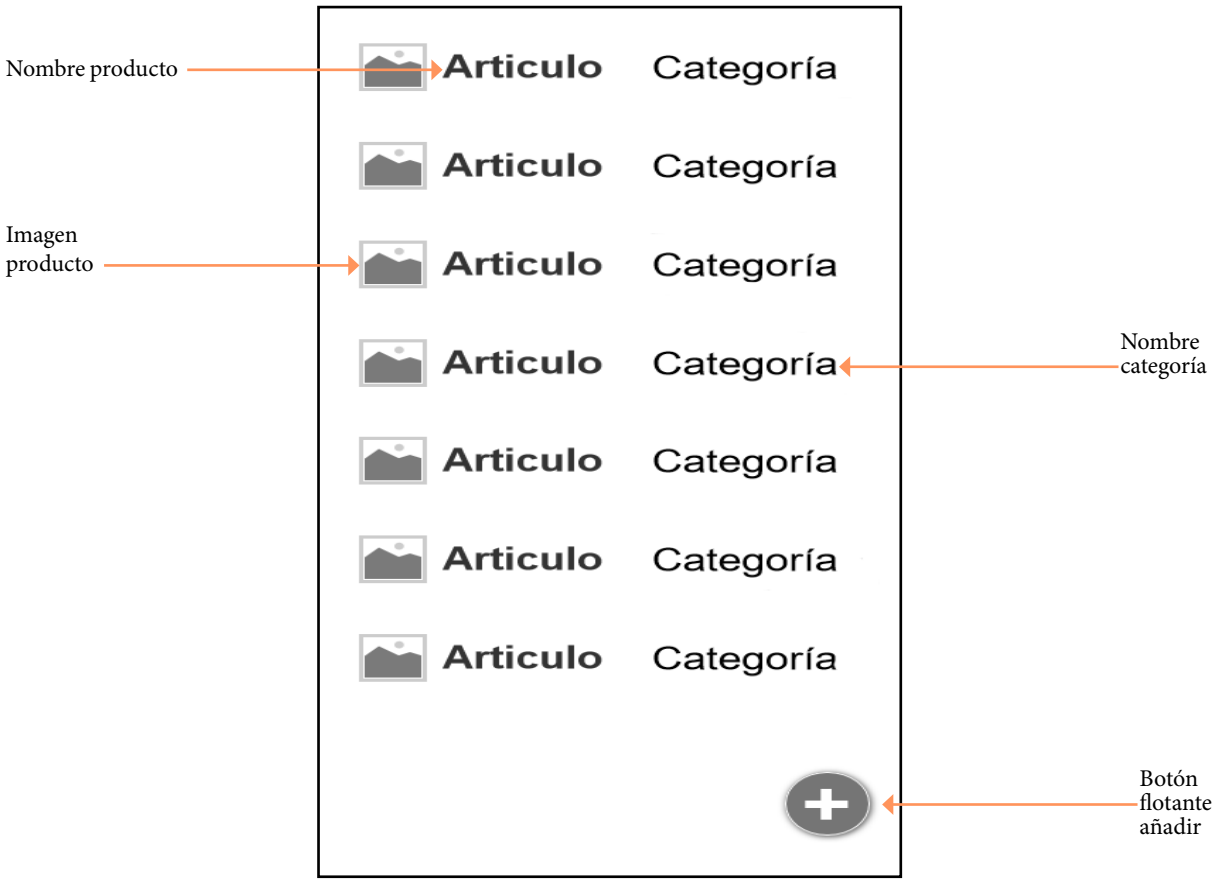


Figura a51: Wireframe pantalla Productos vertical

Añadir productos

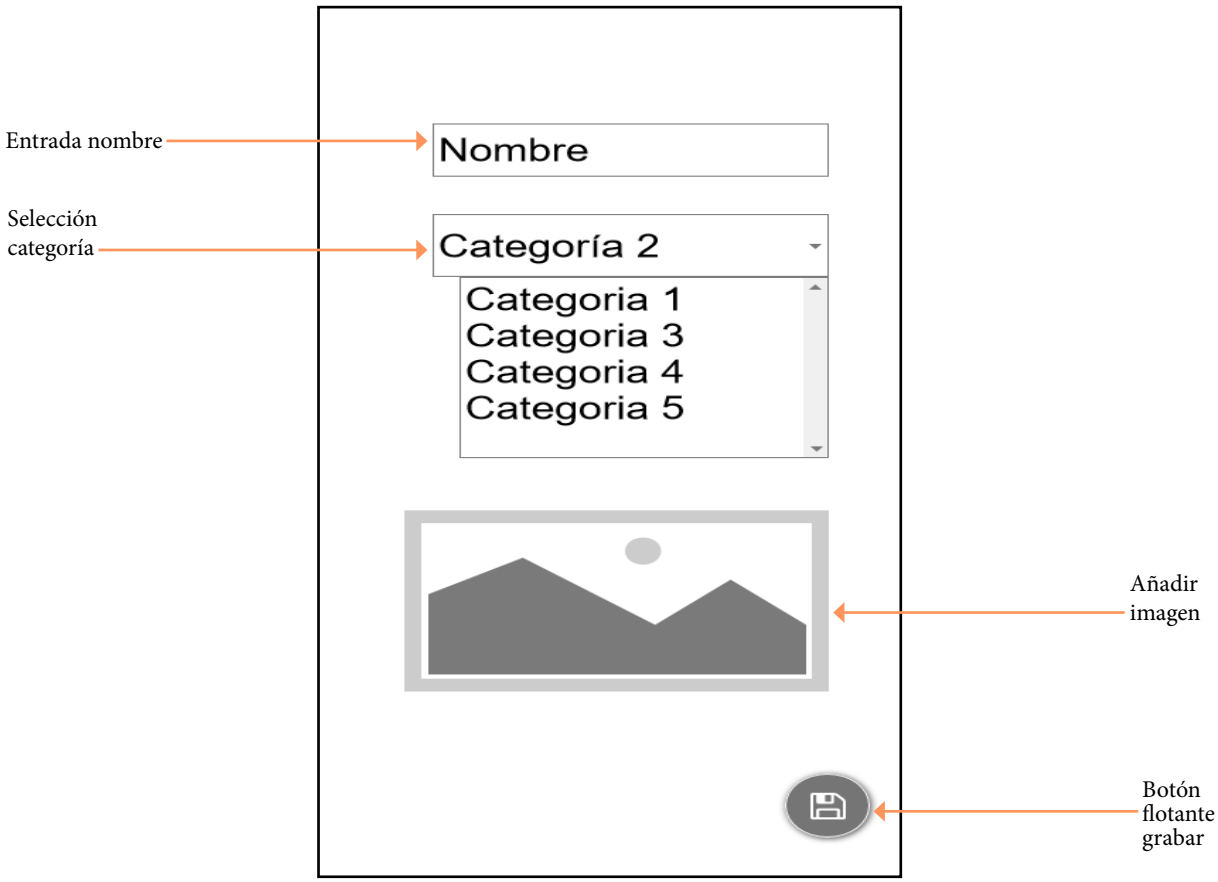


Figura a52: Wireframe pantalla Añadir Productos vertical

Categorías

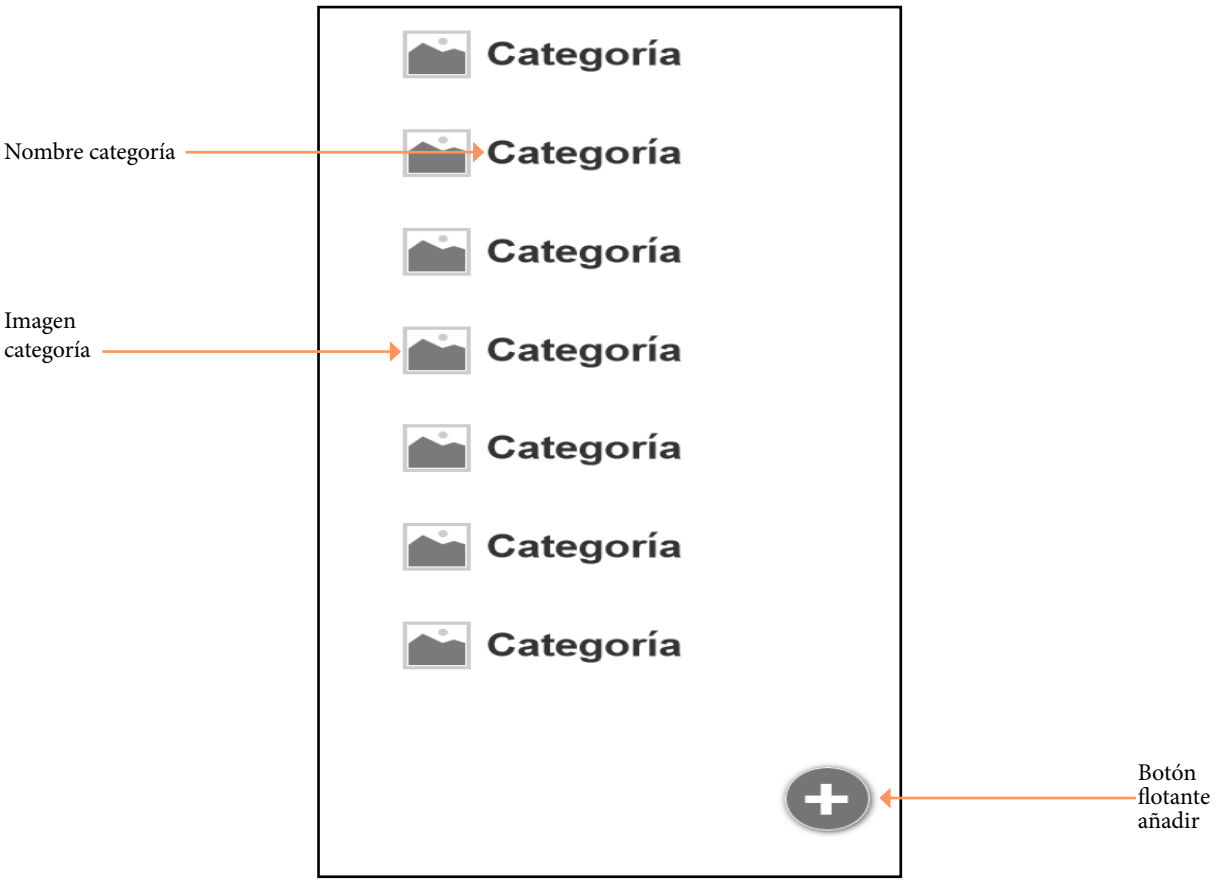


Figura a53: Wireframe pantalla Categorías vertical

Añadir categorías

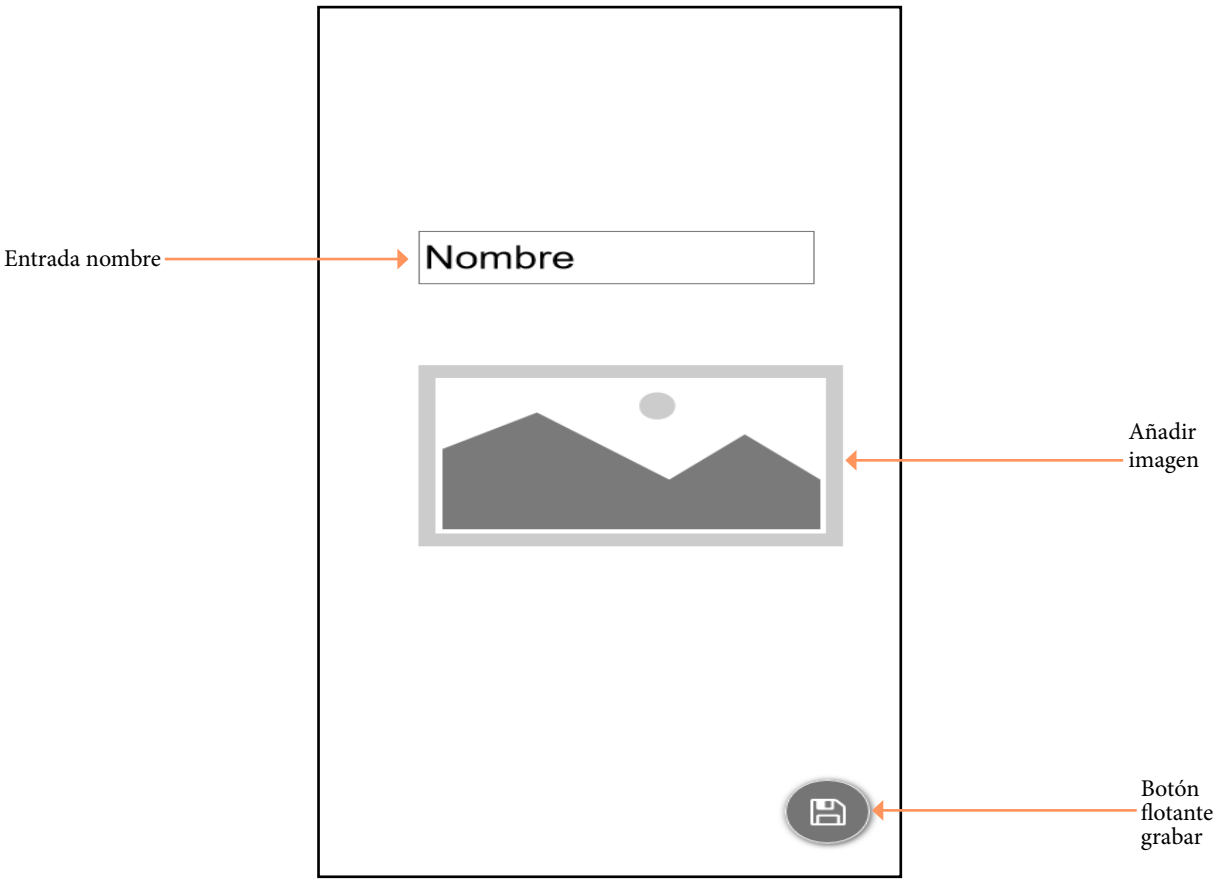


Figura a54: Wireframe pantalla Añadir Categorías vertical

Listas

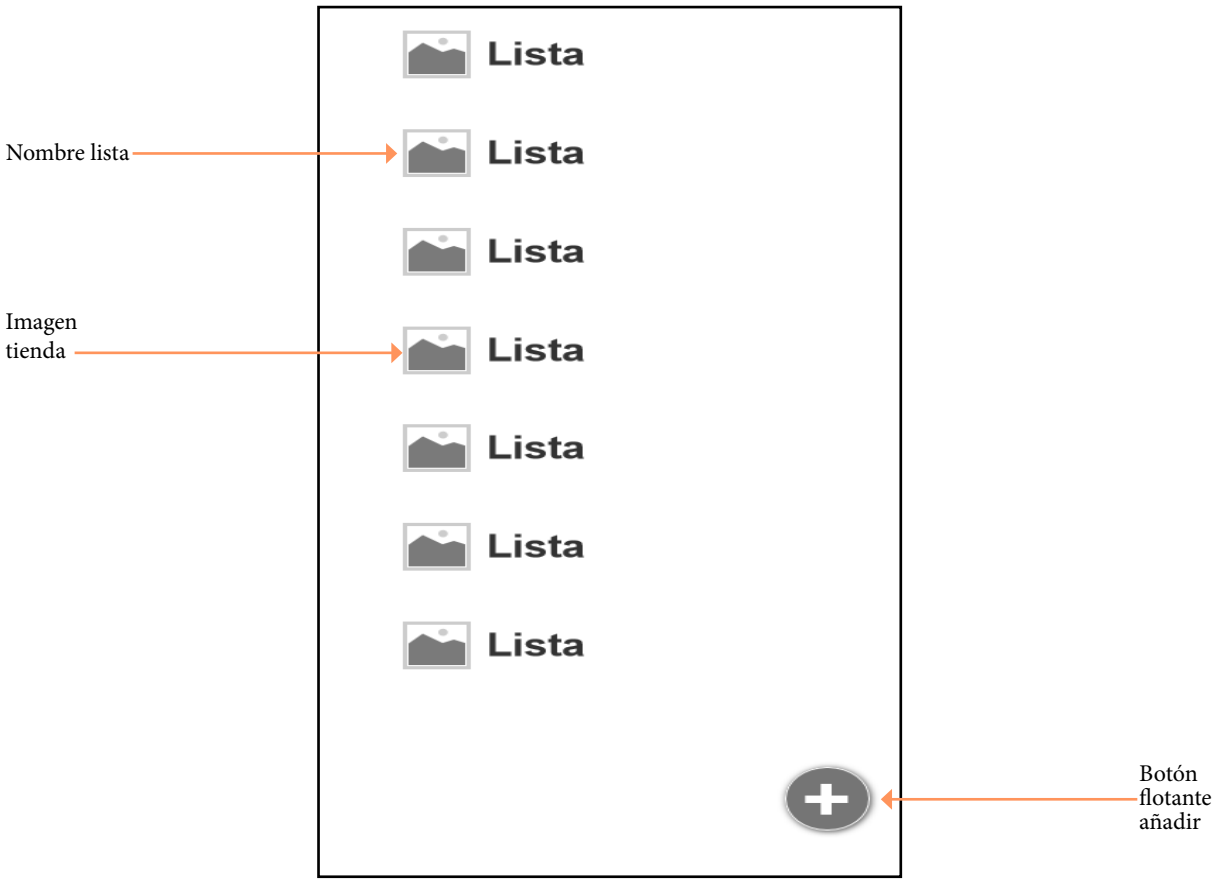


Figura a55: Wireframe pantalla Listas vertical

Añadir lista

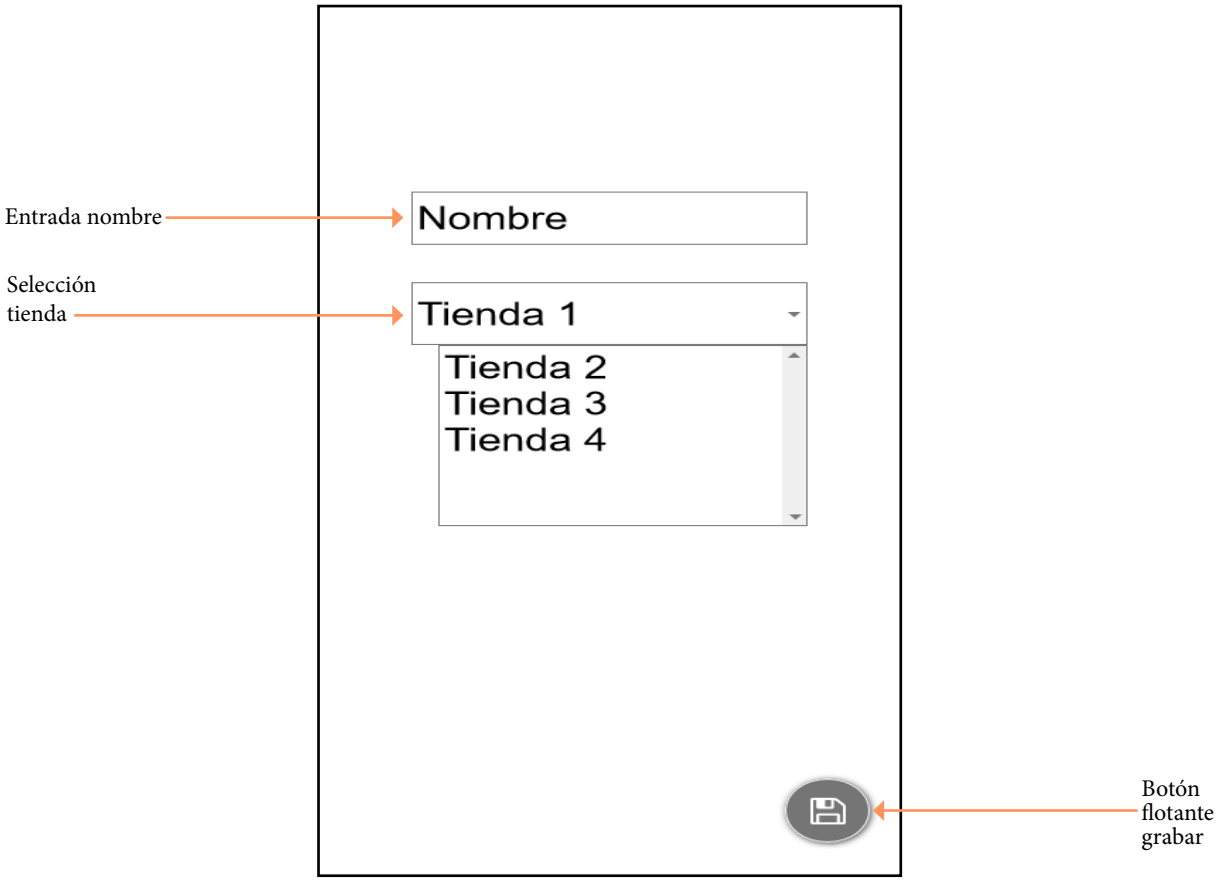


Figura a56: Wireframe pantalla Añadir Lista vertical

Orden categorías

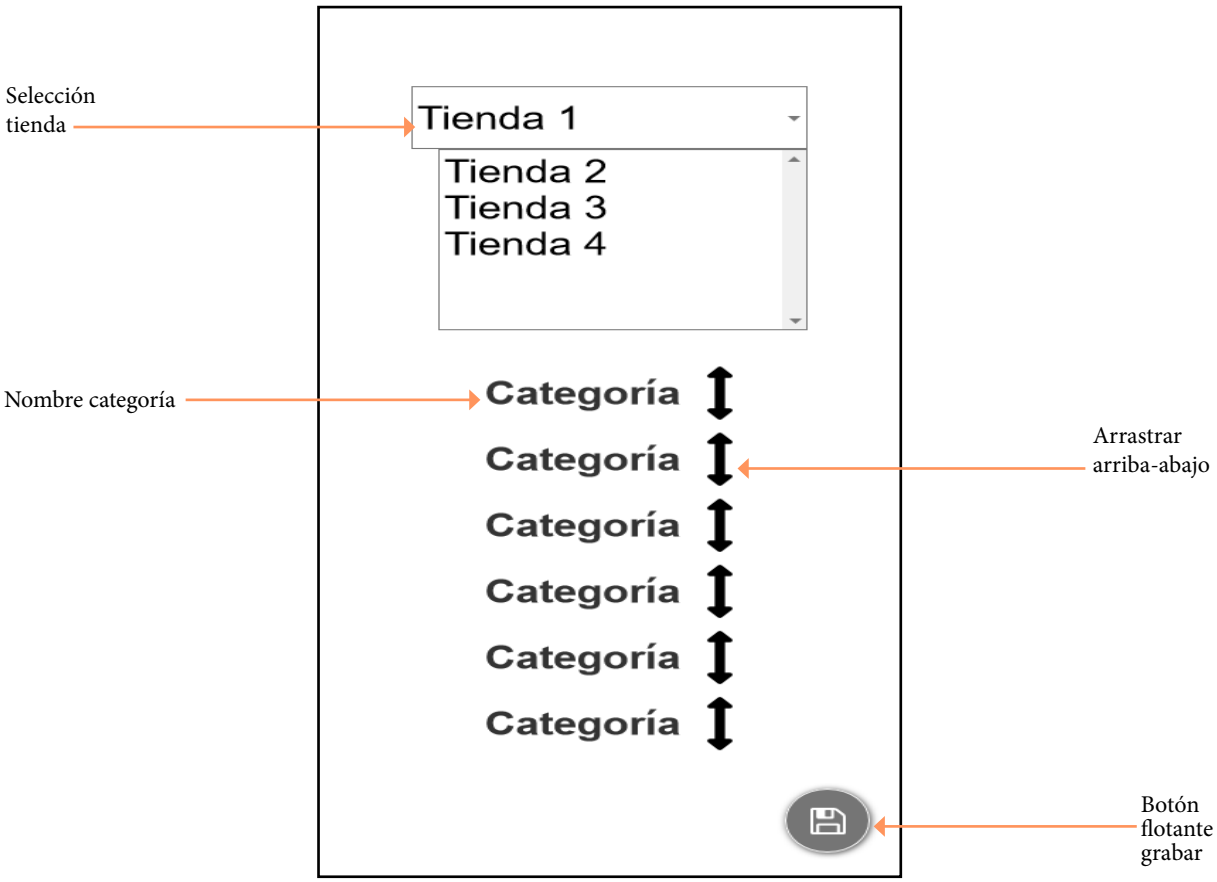


Figura a57: Wireframe pantalla Orden Categorías vertical

Usuario

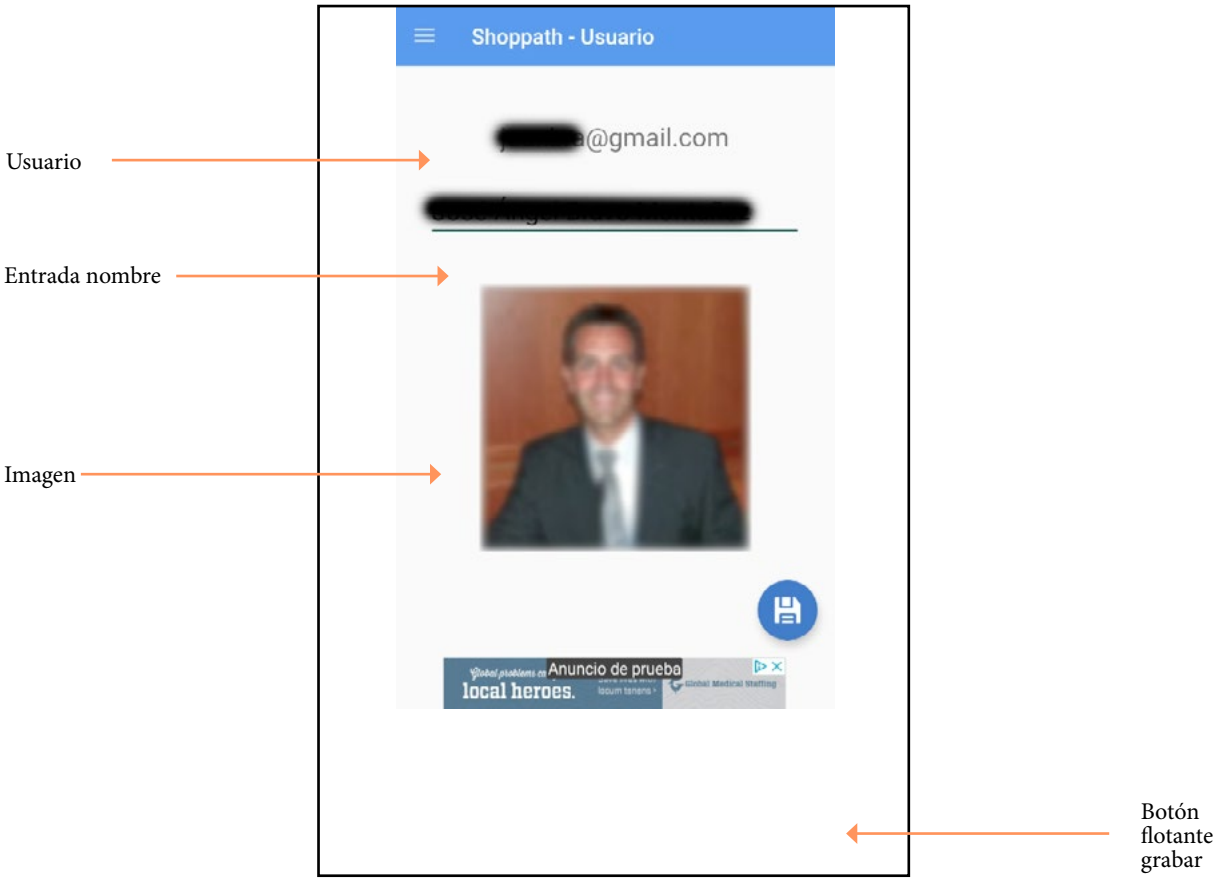


Figura a58: Wireframe pantalla usuario vertical

Acerca de

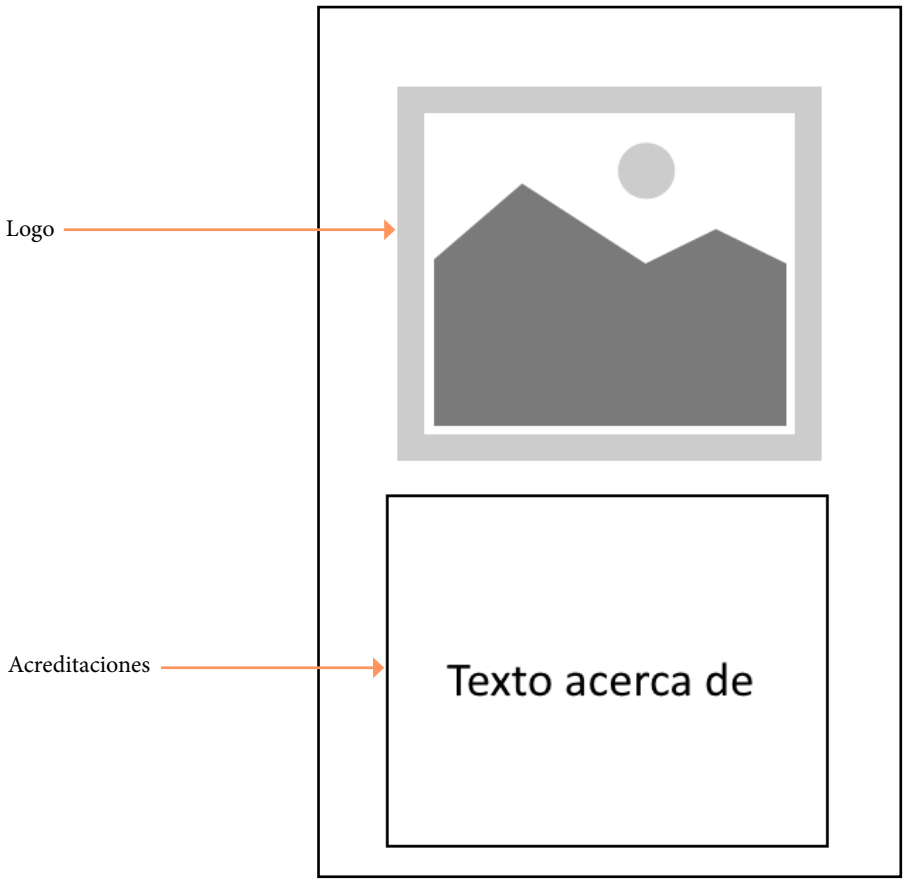


Figura a59: Wireframe pantalla Orden Acerca de vertical

Menú principal

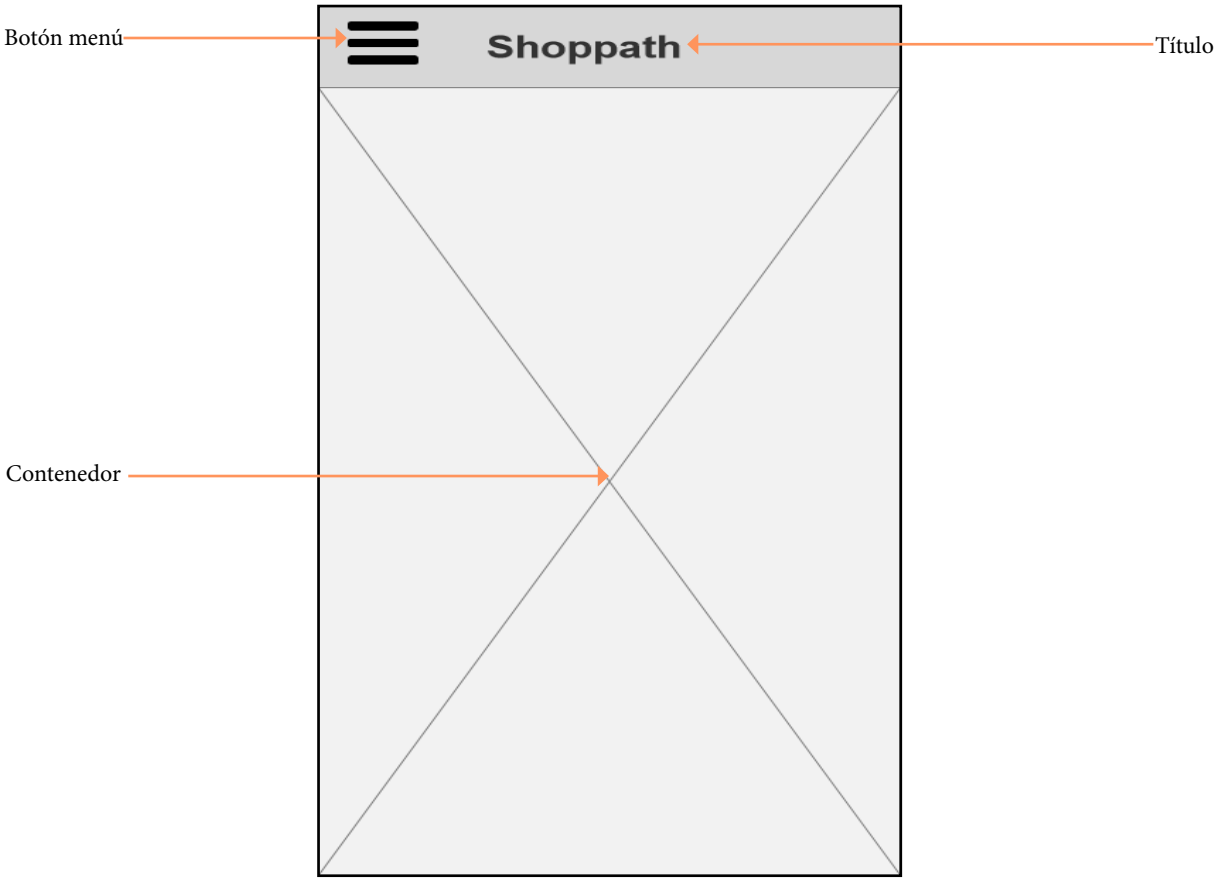


Figura a60: Wireframe pantalla Menú Principal vertical

Menú principal (desplegado)

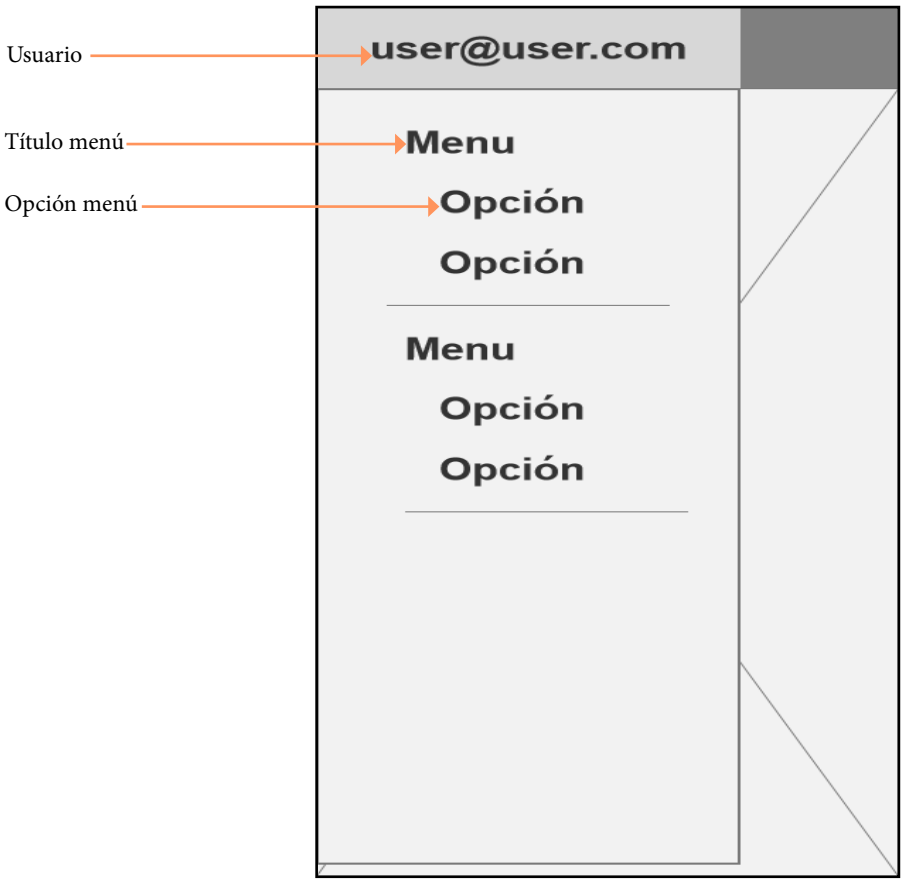


Figura a61: Wireframe pantalla Menú Principal (desplegado) vertical

- **Pantallas en horizontal/tablet**

Splashscreen

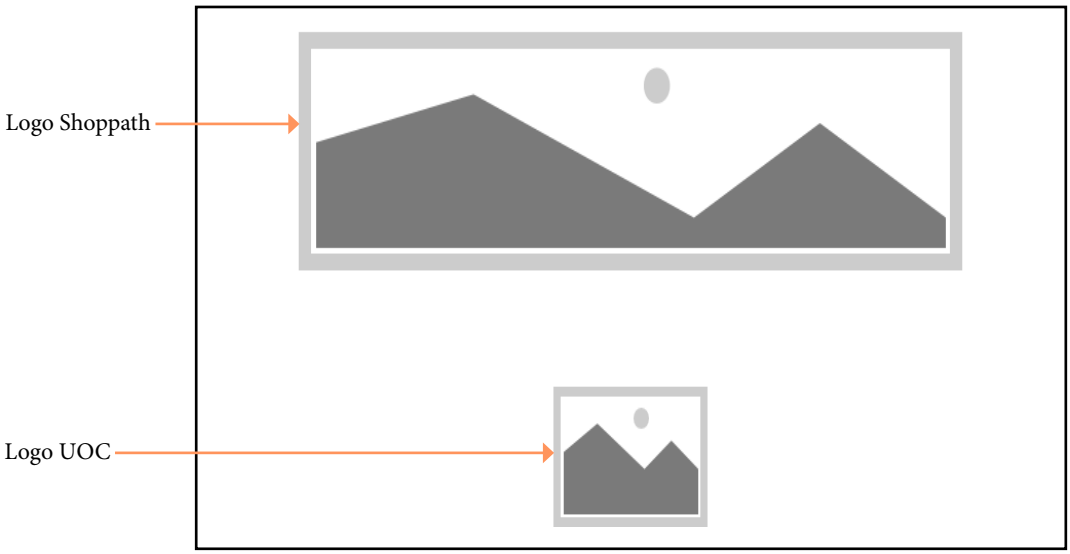


Figura a62: Wireframe pantalla Splashscreen horizontal

Login/alta usuario

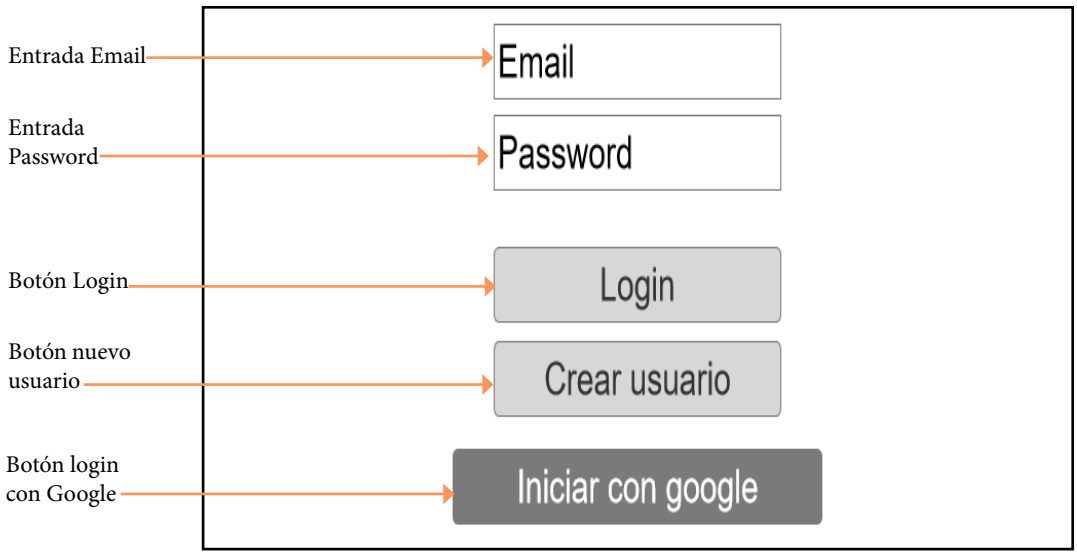


Figura a63: Wireframe pantalla Login horizontal

Mantenimientos

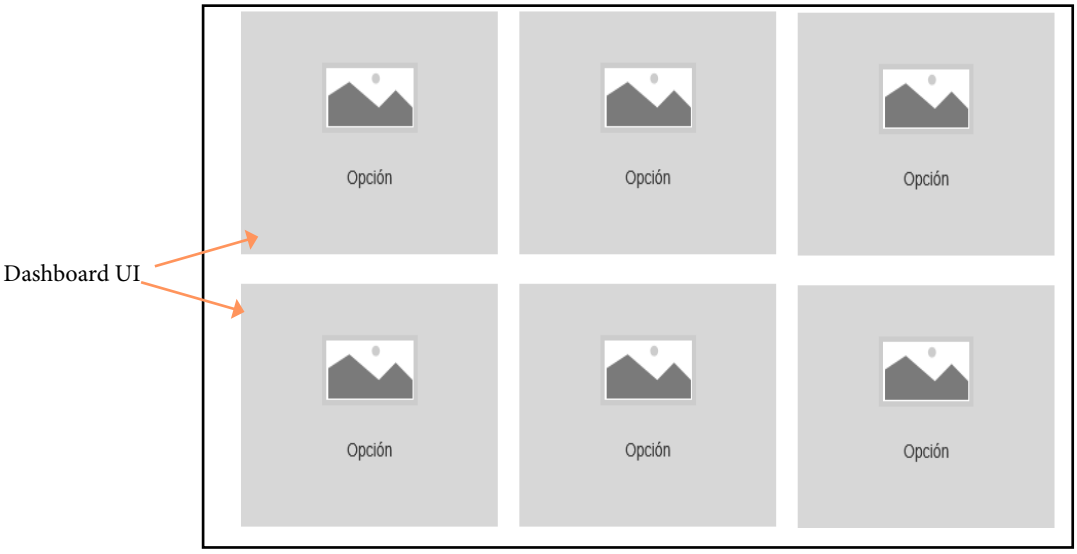


Figura a64: Wireframe pantalla Mantenimientos horizontal

Lista actual

Figura a65: Wireframe pantalla Lista Actual horizontal

Índice

Tiendas

Figura a66: Wireframe pantalla Tiendas horizontal

Añadir tienda

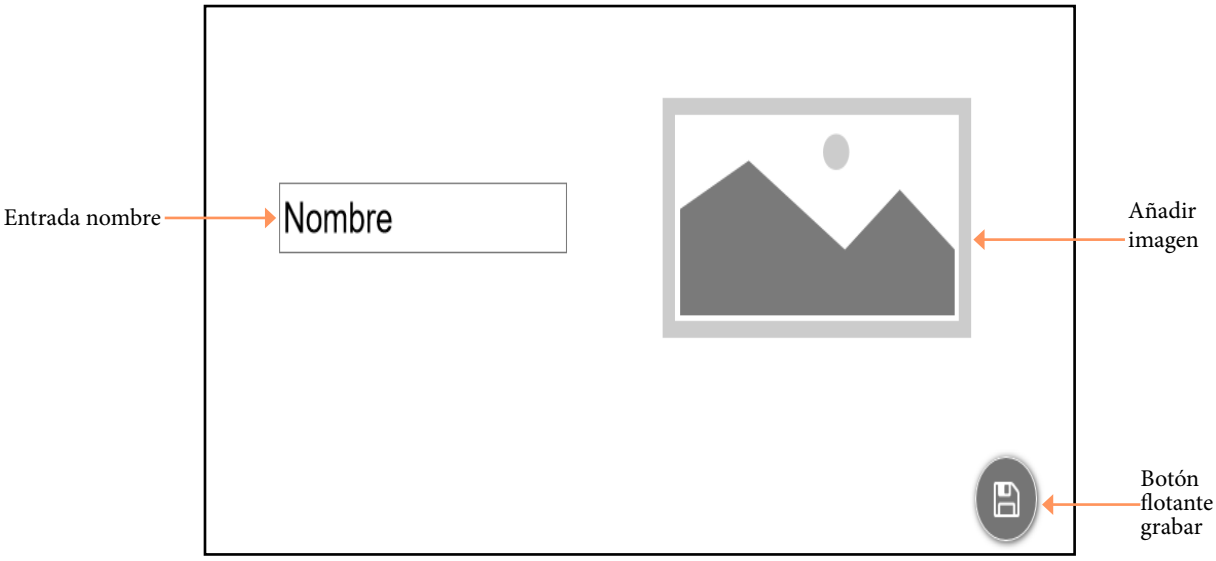


Figura a67: Wireframe pantalla Añadir Tienda horizontal

Productos

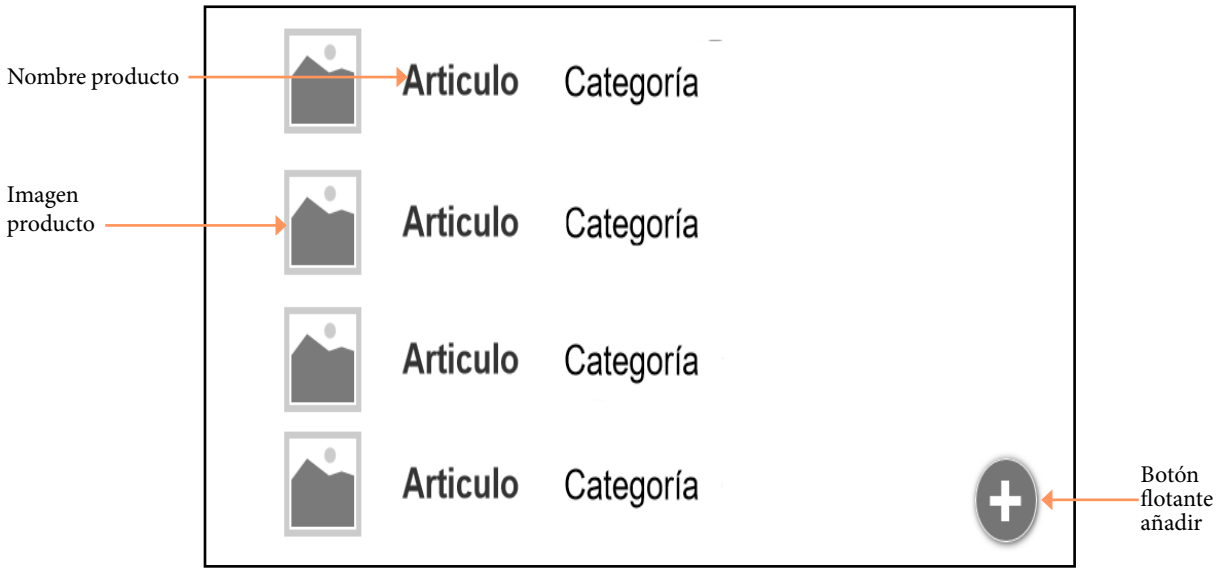


Figura a68: Wireframe pantalla Productos horizontal

Añadir productos

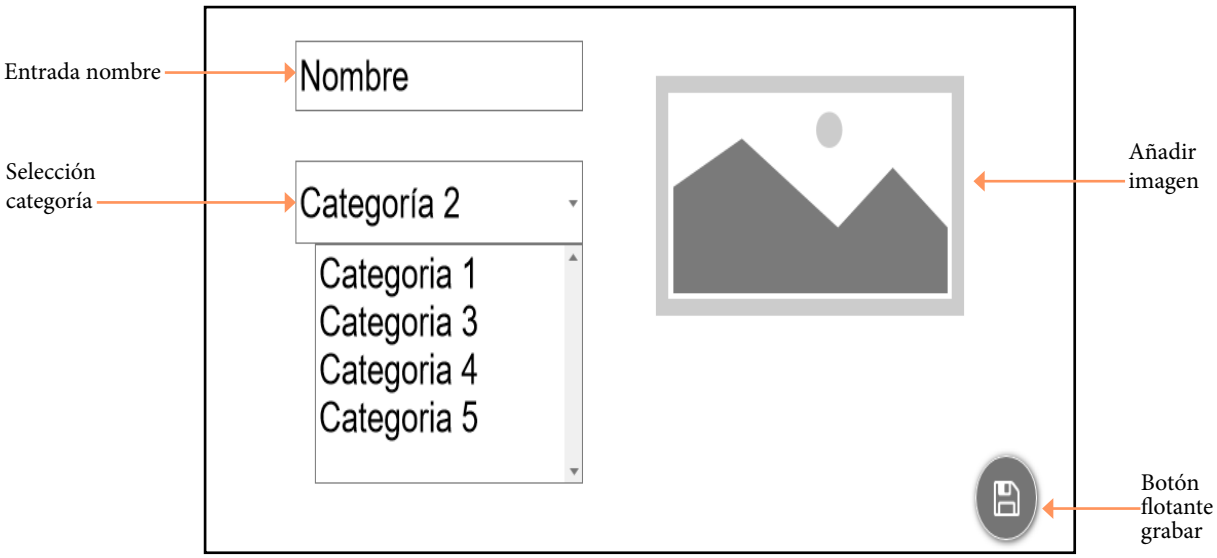


Figura a69: Wireframe pantalla Añadir Productos horizontal

Categorías

Figura a70: Wireframe pantalla Categorías horizontal

Añadir categorías

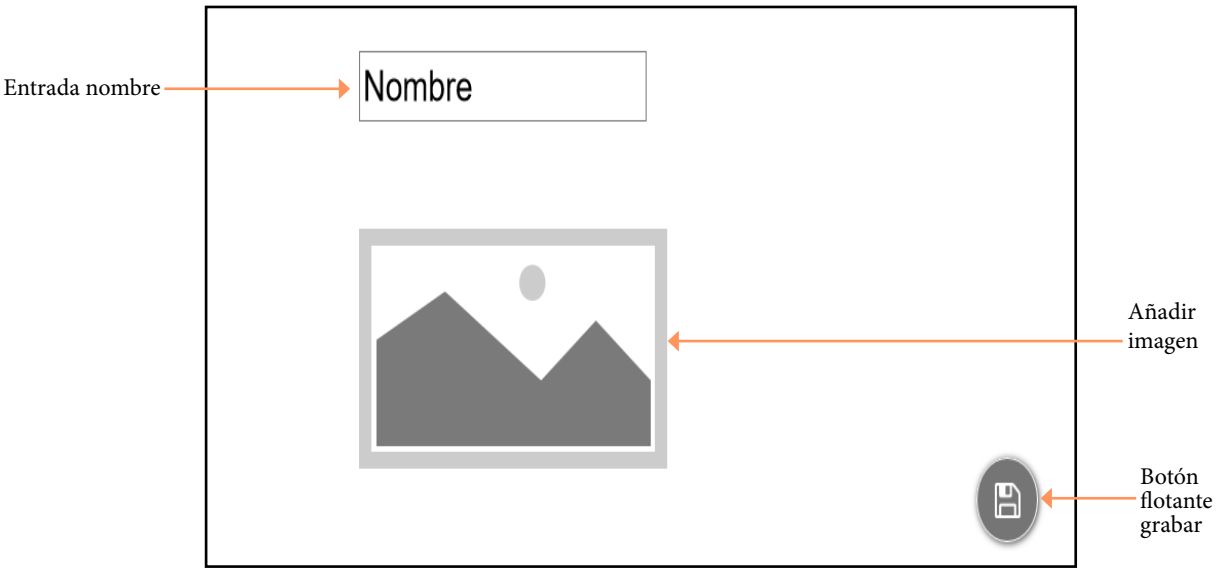


Figura a71: Wireframe pantalla Añadir Categorías horizontal

Índice

Listas

Figura a72: Wireframe pantalla Listas horizontal

Añadir listas

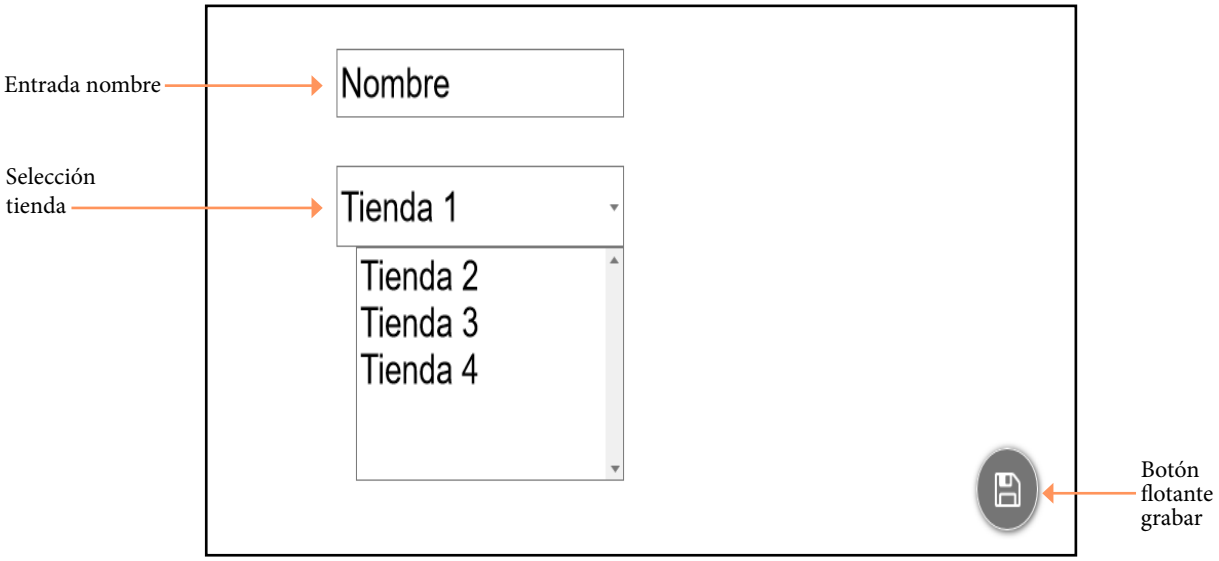


Figura a73: Wireframe pantalla Añadir Listas horizontal

Orden categorías

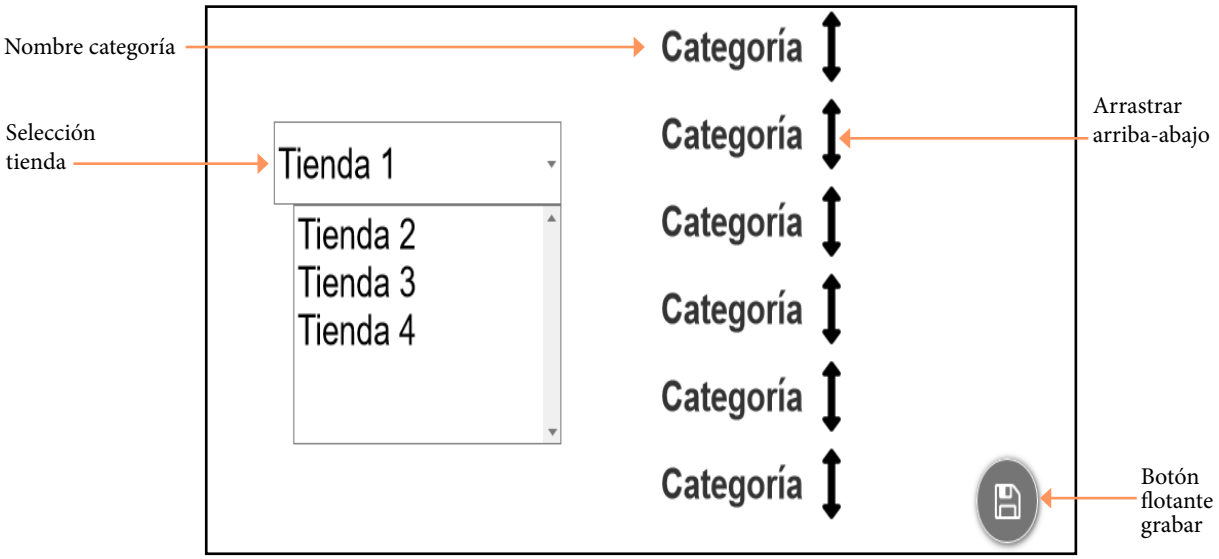


Figura a74: Wireframe pantalla Orden Categorías horizontal

Índice

Usuario

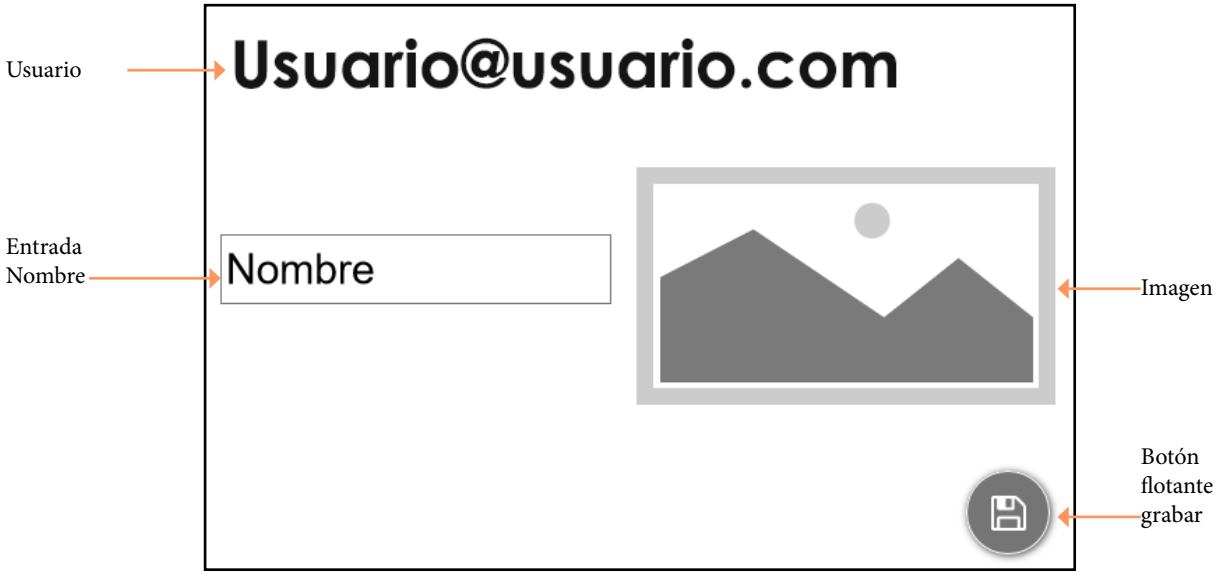


Figura a75: Wireframe pantalla usuario horizontal

Acerca de...

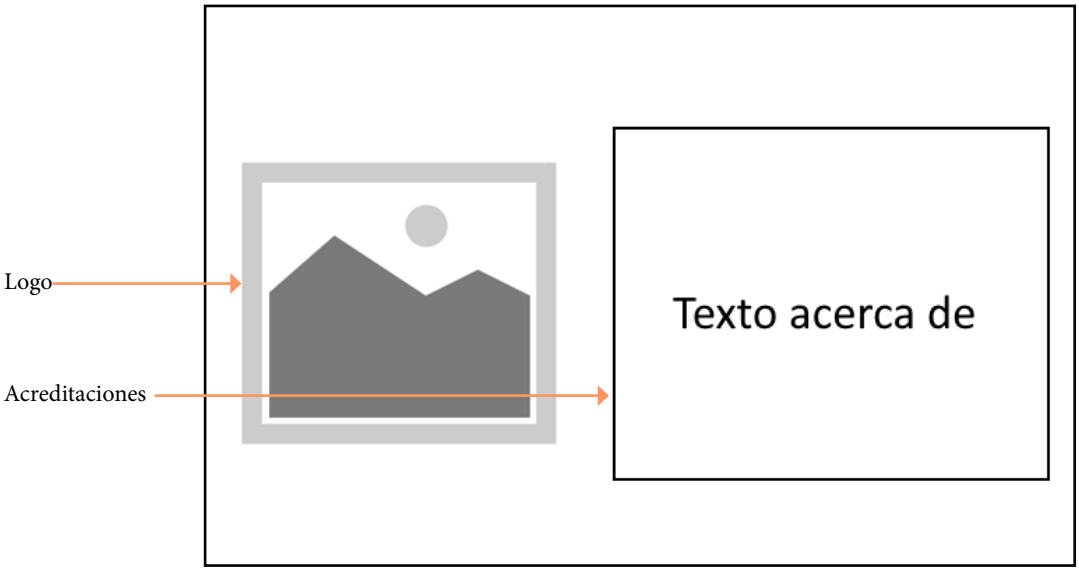


Figura a76: Wireframe Acerca de horizontal

Menú principal

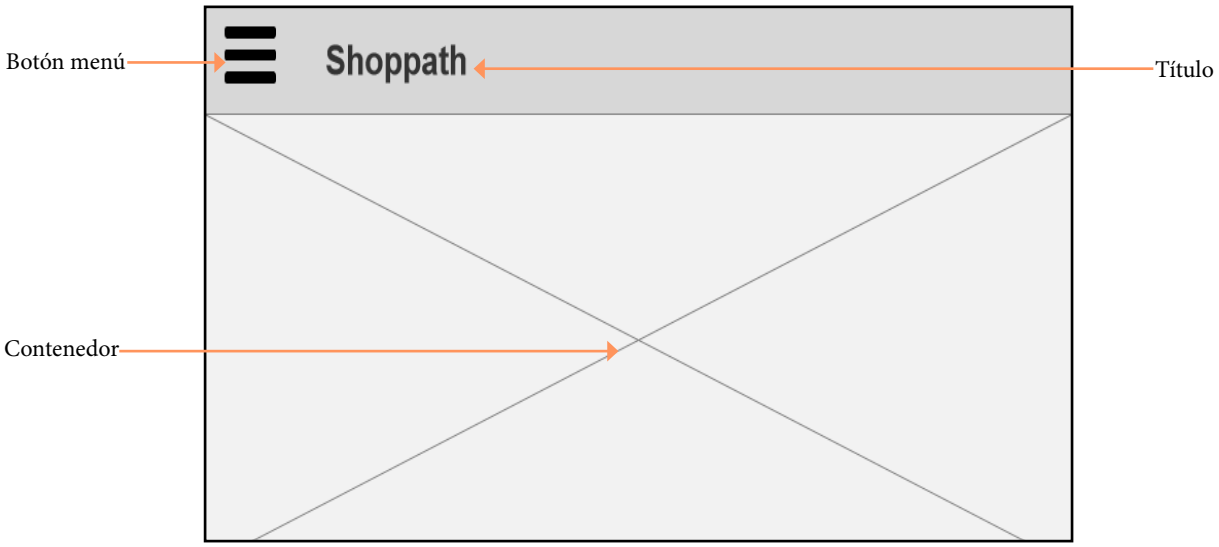


Figura a77: Wireframe pantalla Menú Principal horizontal

Menú principal (desplegado)

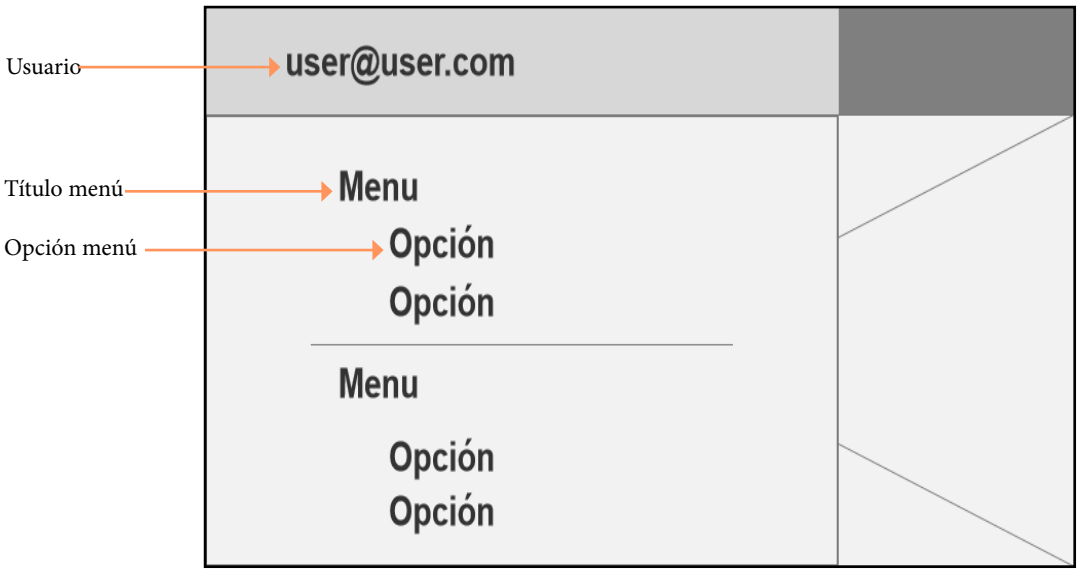


Figura a78: Wireframe pantalla Menú Principal (desplegado) horizontal

Anexo 5. Vita

El autor, José Ángel Bravo Montañez, padre de dos niños de 1 y 5 años, es programador y técnico de sistemas en Mina Serveis D'aigua, en Terrassa (Barcelona).

Su carrera se ha desarrollado principalmente en el sector informático, realizando tareas de programación, gestión de sistemas y soporte a usuario en las empresas:

Continente/Carrefour [2000-2001] Operador informática/soporte a usuario.
Logic Control [2001-2002] Técnico de Software.
Mina Serveis D'aigua [2002-Actualidad] Programador/sistemas/soporte a usuario.

El primer contacto con la informática fue cursando 6º de EGB el 1988, con 11 años, haciendo los primeros cursillos de Basic con un Sinclair ZX Spectrum. Ya entonces decidió que seguiría su educación por este camino, haciendo formación profesional, un ciclo formativo de grado superior y, años más tarde, un grado en multimedia, seguido del Máster al que pertenece este trabajo.