UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Aplicación Android: Gestionando Bohodot

Trabajo final de carrera

ETSETB

Universidad Politécnica de Catalunya

Maria del Mar Torras de Fortuny

INGENIERIA DE TECNOLOGIAS Y SERVICIOS DE TELECOMUNICACIÓN: SISTEMAS ELECTRÓNICOS

Marcel Fernández Muñoz

Barcelona, Octubre 2019

Abstract

This project is to design, develop and implement an Android application with the aim of controlling Bohodot's stock. Firebase has been used for the application backend, both for real-time data synchronization, user authentication, file storage, and function deployment.

Resum

Aquest treball consisteix en dissenyar, desenvolupar i implementar una aplicació Android amb l'objectiu de controlar el stock de l'empresa Bohodot. Pel backend de l'aplicació s'ha utilitzat Firebase, per la sincronització de dades en temps real, l'autenticació dels usuaris, l'emmagatzematge d'arxius i la implementació de funcions.

Resumen

Este trabajo consiste en diseñar, desarrollar e implementar una aplicación Android con el objetivo de controlar el stock de Bohodot. Para el backend de la aplicación se ha utilizado Firebase, tanto para la sincronización de datos en tiempo real, como para la autenticación de usuarios, el almacenamiento de archivos y la implementación de funciones.

A mi familia.

Historial de revisiones y aprobaciones

Revisión	Data	Propuesta
0	24/09/2019	Creación del documento
1	12/10/2019	Revisión del documento

DOCUMENT DISTRIBUTION LIST

Nombre	Correo electrónico
Maria del Mar Torras de Fortuny	martorras3@gmail.com
Marcel Fernández Muñoz	marcelf@entel.upc.edu

Escrito por:		Revisado y escrito por:	
Data	24/09/2019	Data	12/10/2019
Nombre	Mar Torras	Nombre	Marcel Fernández
Posición	Autor del proyecto	Posición	Supervisor del proyecto

Indice

Abstract 1
Resum 2
Resumen 3
Historial de revisiones y aprobaciones 5
Indice 6
Lista de gráficas, tablas e imagenes 7
1. Introducción 8
1.1. Definición y contexto del problema 9
1.2. Objetivos 9
1.3. ¿Por qué Android? 10
1.4. Requisitos y especificaciones 11
1.5. Limitaciones y dificultades encontradas 11
2. Solución propuesta 12
3. Estado del arte 14
3.1. Android 14
3.2. Firebase 16
4. Metodología 17
4.1. Fase inicial 17
4.2. Desarrollo del software 18
4.3. Fase final 18
5. Diagaram de Gantt 20
6. Resultados 21
6.1. Android Studio 21
6.2. Firebsase 29
7. Costes 33
8. Conclusiones y propuestas de mejora 35
Bibliografía 37
Anexo 38

Lista de gráficas, tablas e imágenes

Lista de gráficas

1. Número de apps instaladas y descargadas en 2017 8
2. Porcentaje de instalacines 10
3. Número de applicaciones disponibles 10

Lista de tablas

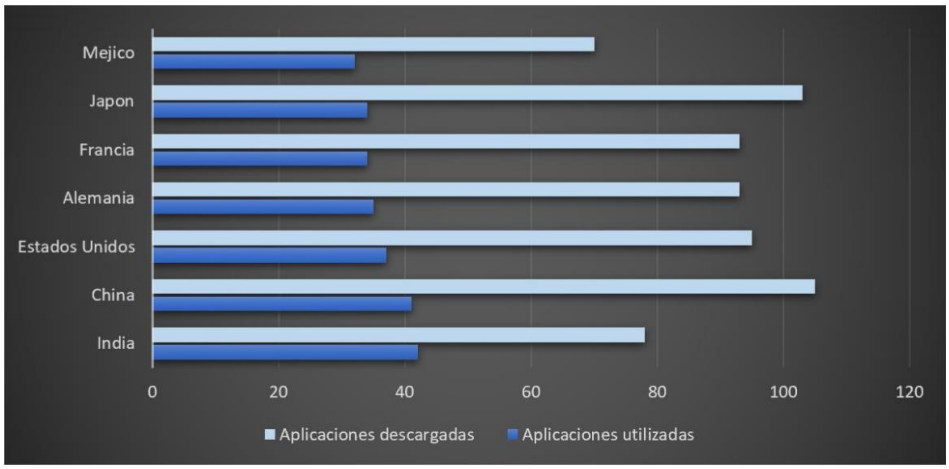
1. Versiones Android 15
2. Costes recursos humanos 33
3. Costes recursos materiales 33

Listas de imágenes

1. Estructura Android 14
2. Firebase 16
3. Metodología seguida 17
4. Iniciar sesión 21
5. Registrar usuario 22
6. Busar 23
7. Estampado seleccionado 23
8. Modelo seleccionado 23
9. Bikini concreto 23
10. Tiquet 24
11. Producto seleccionado 24
12. Pedidos a tiendas 25
13. Pedido concreto 25
14. Ventana emergente datos tienda 25
15. Ventana emergente tallas 25
16. Notificaciones 26
17. Resrevas 27
18. Reserva concreta 27
19. Usuarios 28
20. Usuario concreto 28

1. Introducción

El mercado de las aplicaciones móviles está creciendo exponencialmente a nivel mundial y según estudios realizados se prevé un incremento del 380% entre los datos de 2016 y lo que se alcanzará en 2021. El tiempo invertido en las aplicaciones también es un factor que va creciendo significativamente. Para hacernos una idea cuantitativa de las aplicaciones que nos llegamos a descargar podemos ver la gráfica 1.



GRÁFICA 1 – Número de apps descargadas y utilizadas en 2017

Una media de 85 aplicaciones en cada teléfono, si te paras a pensar es sorprendente el número. Hoy en día hay aplicaciones de todo tipo que te permiten hacer una gran cantidad de cosas desde tu mano sin apenas esfuerzo.

Es evidente que es un sector donde si te sumerges en él tienes muchísimas cosas que aprender. Por este motivo mi trabajo de final de grado consiste en hacer una aplicación. Para entrar y empezar a introducirme desde cero en este gran mundo.

Seguramente cuando piensas en aplicaciones pensarás en *Instagram*, *Whatsapp*, *Candy Crush*, *Google maps*, *just eat* o en cualquier aplicación con una finalidad de entretenimiento, de ocio, de acceso a servicios… pero no debemos olvidar las que tienen una finalidad profesional.

Cada vez más, las empresas tienen la necesidad de disponer de una tecnología que les ayude y facilite el trabajo ya sea para llegar a los clientes o para los empleados.

La tecnología es cara, es por eso, que las grandes empresas acaban adoptando su propio departamento de informática ya que económica y profesionalmente hablando es más rentable a largo plazo.

Este trabajo consiste en diseñar y crear una aplicación Android para la empresa Bohodot. La app está destinada a los trabajadores, es decir, a nivel interno de la empresa.

Bohodot es una marca de trajes de baño para el sector femenino de la población. Es una empresa familiar lo que implica que no dispone de los recursos a los que tienen acceso con mayor facilidad grandes empresas. Se encuentra en un momento de crecimiento y disponer de una buena tecnología puede facilitar mucho el trabajo.

Bohodot diseña los biquinis y los lleva al taller donde los confeccionan. Una vez los bikinis están a punto se llevan al almacén de Bohodot donde están listos para venderse. La marca vende de maneras diferentes: a través de la página web, en la tienda propia Bohodot Barcelona, a tiendas alrededor del mundo y en algunos markets.

La tienda Bohodot Barcelona ha sido inaugurada en febrero de este año, muestra del crecimiento de la marca que ya no se conforma con solo vender online, markets y distintas tiendas sino que quiere su propia tienda.

1.1. Definición y contexto del problema

Para entender el problema tenemos que adentrarnos un poco en la historia de Bohodot. Es una marca que se ha creado desde cero y originariamente la escala de ventas era de decenas. La empresa, con años de por medio, se ha ido haciendo un hueco en la moda femenina y ahora la escala de ventas es infinitamente mayor.

Ha habido un cambio de cantidades a producir y vender muy significativo y, por lo tanto, el control que podías tener vendiendo decenas de biquinis al mes no es el mismo que vendiendo cientos. Hay que adaptarse a los cambios antes de que se descontrole.

Para adentrarse más y hacerse una idea visual de la marca consultar la web a través del *link 1* que se encuentra en la bibliografía.

El principal problema para Bohodot es que llegan bikinis de un taller y se van bikinis por distintos sitios, es decir, de manera online, se van a la tienda propia, se venden a las tiendas o se lo llevan a los markets. Y cada vez las cantidades son más grandes. Realmente es todo un reto controlar y gestionar el stock sin utilizar ningún tipo de tecnología u otro recurso.

Ahora mismo no tienen un control total del stock y eso puede llegar a suponer un problema y, sobre todo, supone más trabajo.

Este proyecto ha salido a través de estar trabajando en Bohodot y parte totalmente de cero, no ha habido trabajos previos para solucionar este problema. Ya que, como he comentado, no se han encontrado con esta necesidad previamente.

1.2. Objetivos

El objetivo de este trabajo final de carrera es diseñar e implementar una aplicación Android hecha a medida según las necesidades de la empresa Bohodot.

La aplicación debe encargarse de:

- Mantener el stock controlado en todo momento para facilitar el día a día en Bohodot.

En ocasiones clientas de la tienda o incluso de los markets piden un producto concreto el cual no está en la tienda. Ahora mismo, cuando eso pasa no puedes decirle a la clienta si queda en el almacén ya que no hay manera de saberlo más allá de llamar y que lo confirmen. Es un método pesado y que implica que haya alguien en el almacén quitándole tiempo de su trabajo.

En estos momentos, no se sabe cuántos bikinis quedan de cada estampado a no ser que vayas y lo cuentes o recuerdes ese dato. En pleno verano, cuando hay el boom de trabajo, un empleado no debe perder tiempo en esto y, con tanto trabajo, la memoria suele flojear.

- Facilitar a nuevos empleados la integración en la empresa lo más rápido posible.

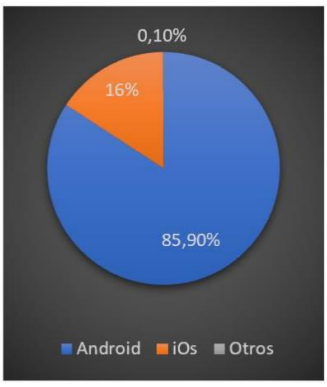
Fruto del crecimiento de la empresa, en verano sobre todo, se necesita a más gente. Son muchos los estampados y modelos que diseña Bohodot y, como es normal, de buenas a primeras no memorizas ni la mitad. Pero en pleno verano no tienes tiempo para ir recordando a tus empleados qué modelo es este y qué estampado es el otro, necesitas agilidad y rapidez.

Entrar en una empresa nueva y tener una aplicación donde ves todos los modelos con sus estampados y viceversa puede ser de gran ayuda. De esta manera las preguntas de los nuevos empleados son mínimas y el resto de empleados pueden dedicarse cien por cien a su trabajo sin desviarse con las nuevas llegadas.

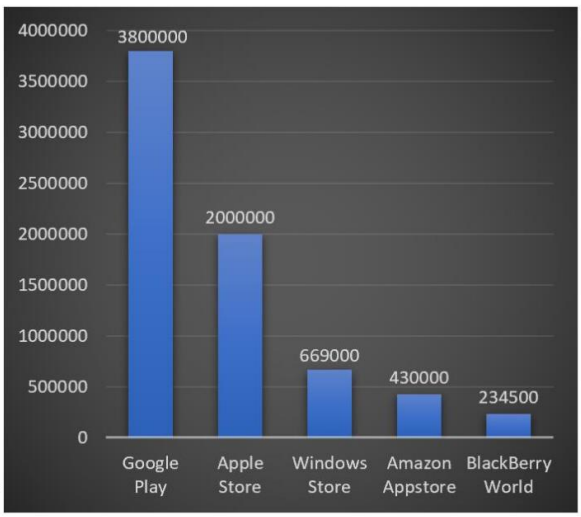
1.3. ¿Qué es Android?

Según estadísticas realizadas entre 2016 y 2017, Apple y el sistema operativo iOS están perdiendo cuota de mercado en todos los países importantes. Apple solo domina los mercados anglosajones, Japón y algunos países nórdicos europeos, el resto es territorio Android.

Android gana indudablemente en número de instalaciones, así lo muestra la *gráfica 2* mostrada en la siguiente página. En concurrencia a esta estadística, las apps existentes en Google Play respecto las de iOS son casi el doble, podemos ver la *gráfica 3*. El problema de Android a diferencia de iOS es que esta fragmentado. Esto provoca que las nuevas versiones lleguen a los consumidores más tarde.



GRÁFICA 2 – Porcentaje de instalaciones



GRÁFICA 3 – Número de apps disponibles

Muchas aplicaciones de Android no existen en Apple, esto se refleja en las descargas. Google Play gana a Apple Store en cifras de uso un 145% más.

1.4. Requisitos y especificaciones

Los requisitos y especificaciones que mi aplicación debe cumplir son los siguientes:

- Interfaz con el usuario que sea intuitiva.
- Aplicación única para Bohodot.
- Gran capacidad de almacenaje y con opción a ir aumentando.
- Personas ajenas a la empresa no tendrán acceso.

1.5. Limitaciones y dificultades encontradas

A lo largo del proyecto no he tenido ninguna limitación grave que me haya impedido conseguir el objetivo principal pero sí he tenido pequeños retos y dificultades que me han hecho invertir más tiempo del que le quería dedica a esa tarea.

Retos y dificultades:

- Descarga de Android Studio versión 3.4.2.

Me descargué el programa para poder empezar a conocer el entorno. La descarga fue bien pero no me generaba la clase R lo que significa que no se creaban las nuevas variables para relacionar los distintos *views* con las funciones java. A modo resumen, sin clase R no puedes hacer aplicación.

Solución exitosa: desinstalarme la versión descargada y probar con otra versión anterior.

- Versiones de Firebase

Firebase es una herramienta que está muy activa y, como es normal, van sacando nuevas versiones. Lo que implica que funciones y métodos que servían con una versión con la nueva no funcionan. En más de una ocasión me ha pasado y me ha tenido muy entretenida buscando operaciones sustitutas.

- Demasiadas ideas

En la fase de desarrollo del proyecto cada vez aparecían más ideas para implementar nuevas funciones y hacer una aplicación más completa. Eso me hacía empezar a buscar cómo poderlo implementar dejando de lado el hilo principal de la aplicación. Sin darme cuenta que muchas veces no se trata de trabajar más sino de trabajar mejor.

Solución: apuntar todas las ideas pero sin darle más vuelta hasta que las bases están asentadas.

2. Solución propuesta

La solución propuesta a este problema es crear una aplicación Android conectada a una base de datos de la plataforma Firebase.

Esta propuesta pretende solucionar los problemas comentados anteriormente basado en situaciones que han ocurrido en el día a día. Me veo en la obligación de obviar el tema de venta online ya que eso supondría conectar mi aplicación a la web y entrar en temas web totalmente desconocidos para mí. El tema de ventas a través de markets tampoco va a ser implementado debido a que la propia empresa Bohodot me ha comunicado su interés por eliminar estos eventos de su agenda en cuanto antes.

Es una aplicación a nivel de gestión interna de la empresa a la cual accederán todos los empleados de Bohodot pero no de igual manera. Habrá dos perfiles de usuarios el "super administrador" y el "administrador". El primero tendrá todos los permisos en todas las pantallas, en cambio, el segundo perfil no tendrá tantos privilegios.

En Firebase se guardará la cantidad de stock que hay de cada modelo en la tienda y en el almacén.

Primero de todo, el empleado tendrá que registrarse con su correo electrónico de la empresa para crear su propia cuenta. Una vez creada el usuario iniciará sesión con esos datos, de esta manera estará siempre identificado.

Tiene 8 apartados: Buscar, Tienda, Pedidos a tiendas, Notificaciones, Reservar, Usuarios, Reponer y Cerrar Sesión.

Buscar. En este apartado se podrá buscar todos los biquinis Bohodot y consultar cuántos quedan de cada modelo y en dónde. De esta manera, conseguimos dos cosas: por un lado si no queda en la tienda y una clienta lo pide poderlo reservar y que se pase en otro momento por la tienda y por el otro en el caso de que queden pocos en los dos sitios, si es necesario, enviar al taller a por más.

Tienda. Con el objetivo de controlar las ventas de la tienda y reponer lo que se vende semanalmente para mantener la tienda con el stock suficiente para atender la demanda.

Pedidos a tiendas. Cada vez son más las tiendas que quieren vender el producto Bohodot. Para poder gestionar todo el procedimiento de hacer los pedidos, prepararlos y enviarlos cuando toquen.

Notificaciones. A través de las notificaciones los usuarios van a estar enterados de los distintos eventos que pasan y, en consecuencia, actuar si es necesario.

Reservar. Puede pasar que una clienta llegue a la tienda a por un producto que ha visto en la web y antes de comprarlo ha preferido venir a la tienda a probárselo y se encuentra con que en la tienda no hay. En el caso que en el almacén quede, se puede reservar. Todos los productos reservados se llevarán del almacén a la tienda donde los clientes pasaran a buscarlo.

Usuarios. Todos los empleados estarán listados con su nombre y email para que en cualquier momento un usuario pueda contactar con otro. El "super administrador", además, tendrá acceso a los datos de todos los usuarios y podrá hacer "super administrador" a uno que actualmente sea "administrador" y viceversa.

Reponer. Cada vez que lleguen biquinis del taller listos para vender hay que reponerlos en la base de datos para que quede todo bien actualizado.

Para desarrollar esta propuesta trabajamos en un entorno de Android Studio y con la plataforma de Google Firebase.

Principalmente utilizaremos las funciones de Firebase para autenticar a los usuarios y para almacenar todos los datos.

3. Estado del arte

En el desarrollo de esta aplicación se combina Android con Firebase, por lo tanto, vamos a ver el estado del arte de estos dos temas.

3.1. Android

Android es un sistema operativo basado en el kernel de Linux. Está diseñado para dispositivos con pantalla táctil. Inicialmente desarrollado por Android Inc, y respaldado económicamente por Google, que más tarde, en el año 2005 adquirió la empresa.

Uno de los aspectos fundamentales del sistema operativo de Android fue su orientación a la multiplataforma, algo realmente novedoso, debido a que hace unos años, un sistema operativo se asociaba a un único dispositivo. Rápidamente esta característica hizo que Android alcanzara sus objetivos, convirtiéndose en el sistema operativo más utilizado.

- Estructura Android

El contexto de Android aplicado a un teléfono móvil es el que se puede apreciar en la *imagen 1*. A parte de su sistema operativo cada dispositivo móvil dispone de su Hardware, Firmware y Ensamblador.

IMAGEN 1 – Estructura Android

Esta imagen la podemos encontrar en el *link 4* de la *Bibliografía.*

Analizamos el sistema operativo de abajo a arriba. Primero de todo tenemos el núcleo de Android. Basado en el kernel del sistema operativo Linux. En esta capa se encuentran todos los drivers y proporciona servicios de seguridad, manejo de memoria, multiproceso y soporte para controladores de dispositivo.

A continuación tenemos la capa de abstracción del hardware donde se conectan con los drivers y el entorno de ejecución (Runtime), basado en la máquina virtual de Java. Dadas las limitaciones de los dispositivos Google decidió crear la máquina virtual Dalvik, la cual respondió mucho mejor a dichas limitaciones. Entre sus características destacan, la optimización de recursos debido a la ejecución de ficheros Dalvik ejecutables (.dex) y la delegación al kernel de Linux procesos como el threading y el manejo de memoria a bajo nivel.

Si seguimos subiendo, nos encontramos con las librerías nativas, escritas en C/C++ y compiladas en código nativo del procesador. Muchas librerías utilizan código abierto. Entre ellas se destacan: System C Library, WebKit, SQLite y SSL.

Finalmente los servicios del Sistema Operativo Android donde se manejan las notificaciones, el ciclo de vida de las actividades, permite acceder a los datos entre apps.

El entorno de aplicación (Framework Android) ofrece una plataforma de desarrollo libre para aplicaciones, donde su principal riqueza reside en la reutilización de componentes. Componentes desarrollados por Google, o por usuarios.

Justo por encima se encuentra la capa de aplicaciones formada por el conjunto de aplicaciones del dispositivo, tanto las nativas como las instaladas por el usuario.

- Versiones Android

El éxito de las tecnologías está en adaptarse a los cambios. Para ello se necesita un cambio constante. Android va evolucionando para mantenerse vivo en este mercado tan dinámico. Las actualizaciones incorporan funcionalidades nuevas o mejoran las ya existentes.

Todas las versiones de Android reciben del inglés el nombre de diferentes postres, siguiendo orden alfabético.

API	Android	Nombre	% disp.
1	1.0	Apple Pie	100
2	1.1	Banana Bread	100
3	1.5	Cupcake	100
4	1.6	Donut	100
5 - 7	2.0 – 2.1	Éclair	100
8	2.2 – 2.2.3	Froyo	100
9 - 10	2.3 – 2.3.7	Gingerbread	100
11 - 13	3.0 – 3.2.6	Honeycomb	100
14 - 15	4.0 – 4.0.5	IceCreamSandwich	100
16 - 18	4.1 – 4.3.1	JellyBean	95.9

19 - 20	4.4 – 4.4.4	Kitkat	95.3
21 - 22	5.0 – 5.1.1	Lollipop	80.2
23	6.0 – 6.0.1	Marshmallow	62.6
24 - 25	7.0 – 7.1.2	Nougat	37.5
26 - 27	8.0 – 8.1	Oreo	1.1
28	9.0	Pie	<1
29	10.0	Q	<1

TABLA 1 – Versiones Android

3.2. Firebase

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Tiene una gran cantidad de funcionalidades (como podemos ver en la *imagen 2*, disponible en el *link 7* de la *Bibliografía*) para que el desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades.

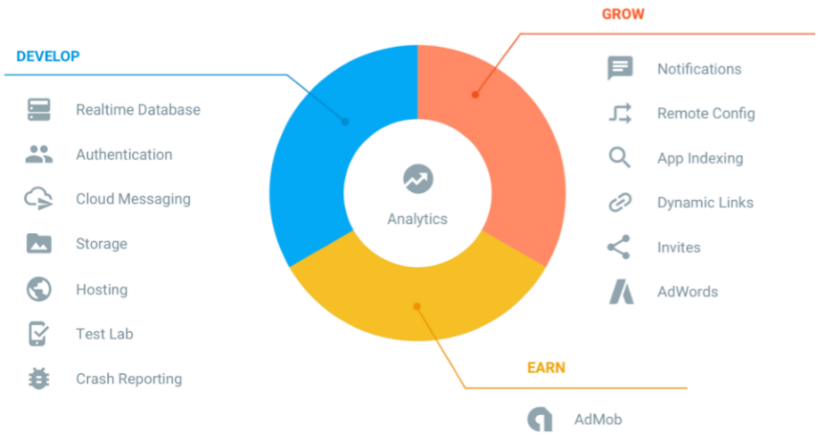


IMAGEN 2 - Firebase

Las funcionalidades a destacar que Firebase nos proporciona son: una base de datos para almacenar información, una manera de identificarte para acceder a tu app, enviar mensajes entro los distintos usuarios, guardar ficheros entre ellos imágenes, obtener estadísticas de la aplicación, generar pruebas para ver qué pasaría si lo implemento y todo esto en tiempo real y se sincroniza con los distintos dispositivos.

4. Metodología

Para realizar el proyecto con éxito se ha hecho de una manera ordenada y organizada. Se han seguido unos pasos pasando el proyecto por una fase inicial, luego se ha desarrollo el software y finalmente la fase final del proyecto. Todas estas fases son imprescindibles. Queda bien plasmado en la *imagen 3* y a continuación lo explico detalladamente.

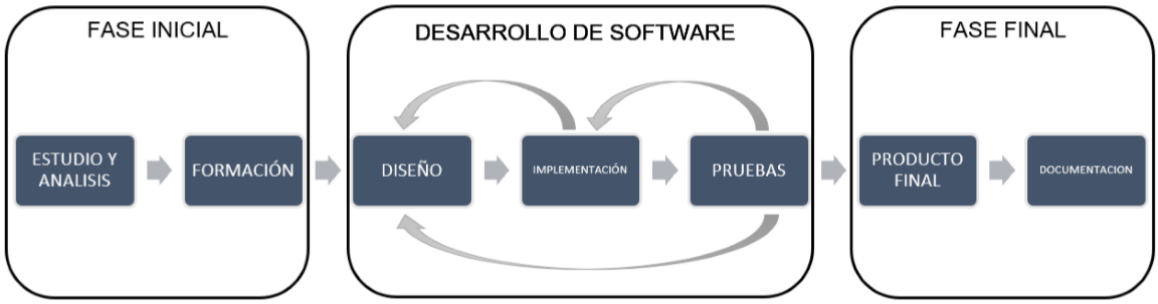


IMAGEN 3 – Metodología seguida

4.1. Fase inicial

El desarrollo de un proyecto software no se puede empezar sin hacer un estudio previo de lo que quiero implementar y desarrollar. De la misma manera, no podemos hacer algo si no sabemos hacerlo, es decir, unos días son dedicados a la formación.

- Estudio y análisis

En esta primera fase, se estudia a fondo la necesidad de nuestro cliente para poder buscar la solución más optima y adecuada para Bohodot.

Se definen los requisitos, definidos en el apartado *1.4 Requisitos y especificaciones*, que el cliente exige acorde con el servicio que le puedo llegar a prestar.

Una vez definido, se analiza y se buscan las posibles implementaciones. A partir de aquí decidir cómo voy a hacerlo y qué herramientas voy a tener que utilizar para ello.

Es importante esta fase ya que va a ser la base del diseño de nuestro producto, por lo tanto debe dejarse todo muy claro, sin ninguna ambigüedad que luego pueda ser motivo de futuras quejas.

- Formación

Una vez decidido cómo voy realizar el proyecto y con qué herramientas voy a trabajar, necesito tenerlas al abasto y saberlas utilizar.

En mi caso, utilizo dos herramientas totalmente nuevas para mí. Por un lado, Android Studio para escribir todo el código que va a ir a la parte de la aplicación Android y, por otro lado, la plataforma de Firebase.

Primero de todo, he tenido que instalarme Android Studio y luego documentarme.

Por suerte para mí, hoy en día, en internet hay muchísima información acerca de estas herramientas y desde ahí he hecho mi fase de formación.

En esta fase he incluido pequeños experimentos para llevar a cabo los conocimiento adquiridos en la formación.

4.2. Desarrollo del software

Con los parámetros definidos y especificados, es el momento de diseñar, implementar y desarrollar la aplicación.

Los pasos no han sido diseñar, implementar y probar de manera unidireccional, es decir, después de algunas pruebas he tenido que modificar o parte del diseño o parte de la implementación.

- Diseño

Teniendo claro el objetivo del proyecto y sin olvidar los requisitos, diseño la aplicación para cumplir con todo esto.

Evidentemente el primer diseño hecho no ha sido el definitivo. Ha habido varios diseños hasta llegar al final.

- Implementación

El diseño se tiene que implementar, es decir, tenemos que hacer que nuestro diseño funcione. Esta fase ha consistido en programar en Android Studio e implementar los servicios de Firebase.

- Pruebas

Las pruebas son las que nos hacen darnos cuenta de si funciona como esperamos o no. Por lo tanto, cuantas más pruebas más datos tenemos para confirmar el comportamiento que vemos y poder afirmar, al final, el buen funcionamiento de la aplicación.

Para realizar las pruebas, he necesitado un móvil Android donde cargar el código y evaluar el resultado.

4.3. Fase final

En esta última fase del proyecto es el momento de retomar el contacto con el cliente e informarles que el producto está listo.

- Producto final

Hechas todas las pruebas de manera exitosa que confirman el buen funcionamiento de la aplicación podemos decir que tenemos el producto final.

En este caso, presentar a Bohodot el resultado final de estos meses de trabajo para que realicen las pruebas pertinentes y ver que todo funciona correctamente.

- Documentación

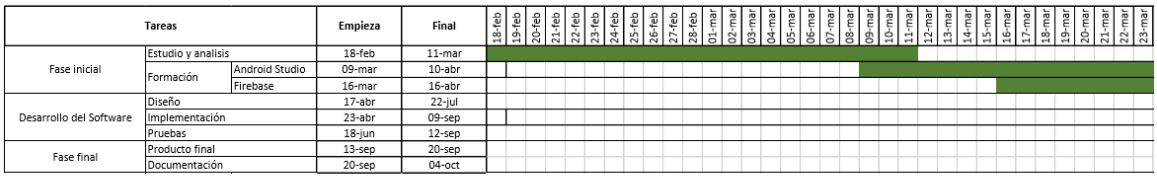
Para acabar con todo proyecto es necesario documentar la aplicación para explicar para qué sirve, cómo funciona y cómo se ha llevado a cabo.

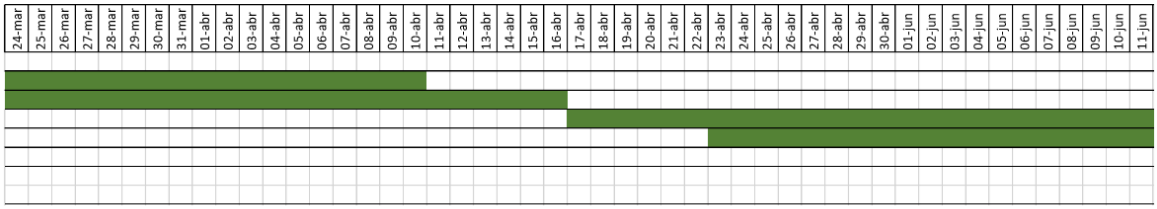
5. Diagrama de Gantt

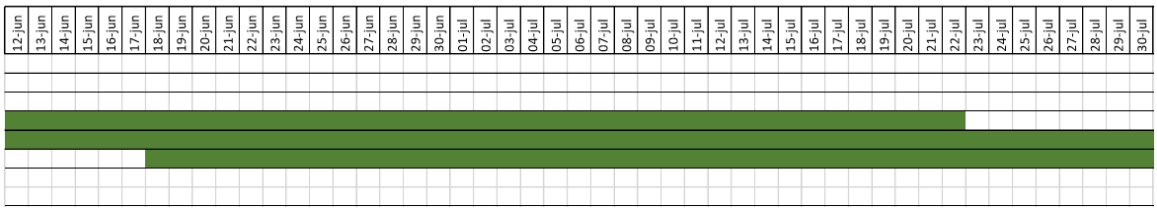
El proyecto se ha realizado entre el 18 de febrero de 2019 y el 7 de octubre de 2019. A continuación se muestra la planificación que se ha seguido.

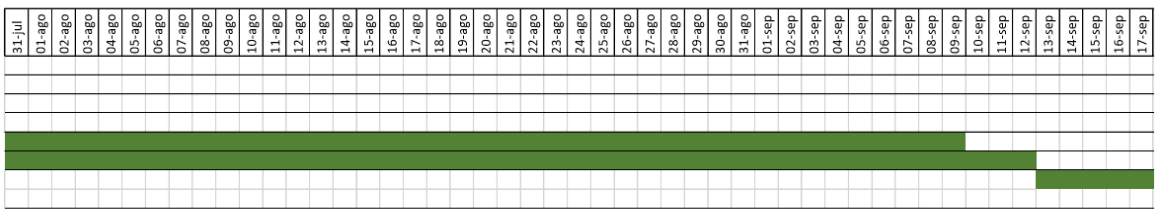
La formación que se hace en la fase inicial es intensa y con el objetivo de aprender todo aquello que es importante en este entorno. Es cierto que a lo largo del desarrollo del software no me ha bastado con lo visto en la fase inicial y también he tenido que ir investigado pero no de la misma manera.

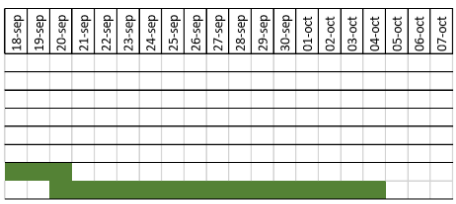
Tareas			Empieza	Final
Fase inicial	Estudio y analisis		18-feb	11-mar
	Formación	Android Studio	09-mar	10-abr
		Firebase	16-mar	16-abr
Desarrollo del Software	Diseño		17-abr	22-jul
	Implementación		23-abr	09-sep
	Pruebas		18-jun	12-sep
Fase final	Producto final		13-sep	20-sep
	Documentación		20-sep	04-oct











6. Resultados

La aplicación final tiene una arquitectura cliente-servidor donde la aplicación es el cliente y Firebase el servidor.

Para exponer el resultado primero explicamos Android Studio y finalmente Firebase.

Para hacerse una idea y poder visualizar la aplicación final, en el anexo adjunto hay un mapa de todas las pantallas de la aplicación.

6.1. Android Studio

Android Studio es el programa utilizado para crear el diseño y la funcionalidad de cada pantalla de la aplicación.

El diseño de las distintas pantallas o fragmentos de pantalla se crea con los ficheros escritos en *XML*. En cambio, la funcionalidad se implementa en *JAVA* de forma independiente a los ficheros del diseño.

Para asociar el diseño a su funcionalidad se crean identificadores únicos para cada elemento que se guardan y acceden a través de la clase R que genera y actualiza automáticamente Android Studio.

El tema y estilo aplicado en esta aplicación está basando en la web. Intentando acercarse lo máximo posible.

En esta aplicación se han creado dos perfiles de usuario diferentes: administrador y super administrador. El primero, a diferencia del segundo, no tendrá acceso a todas las funcionalidades de la aplicación.

Analizamos el producto final pantalla por pantalla:

- Iniciar sesión

Al instalarse la aplicación y entrar por primera vez, aparece la pantalla para iniciar sesión mostrada en la *imagen 4*. Siempre que el usuario cierre sesión y quiera volver a entrar, empezará con esta pantalla.

IMAGEN 4 – Iniciar sesión

En esta *Activity* se pueden dar dos casos: el usuario ya tiene cuenta o el usuario nunca ha entrado y no tiene cuenta.

En el primer caso, el usuario rellena los campos con su correo y contraseña. Aprieta el botón de entrar, lo que provoca que desaparezca el botón y aparezca un *progress bar* para indicar que *Firebase Authentication* está confirmando los datos. Si los datos son correctos, la aplicación va a la pantalla de *Buscar*. Si los datos no son correctos, aparece un mensaje diciendo: *usuario o contraseña incorrecto.*

En el segundo caso, el usuario no podrá acceder a la aplicación sin tener una cuenta, por lo tanto, tendrá que ir a *Registrar nuevo usuario*.

- Registrar nuevo usuario

En esta pantalla el usuario crea su propia cuenta. La cuenta se crea a partir de un correo electrónico existente. En Bohodot todo empleado tiene su correo *@bohodot.es* a través del cual deben crear su cuenta. Cualquier otro correo no es válido para acceder a la aplicación.

IMAGEN 5 – Registrar usuario

Para crear la cuenta se rellenan los siguientes datos: nombre, correo electrónico, contraseña, confirmar contraseña y poner una foto.

Al apretar por primera vez el icono para añadir la foto, aparece una ventana emergente pidiendo permiso para acceder a la galería de fotos. Una vez aceptada la petición, se tiene que seleccionar una foto. Si la petición no se acepta no habrá foto y, por lo tanto, no se podrá crear la cuenta.

Con todos los datos rellenados, *Firebase Authentication* confirma si esa dirección de correo existe. Si es inventada o no es Bohodot no se podrá crear la cuenta. Si es correcto se guardarán los datos del nuevo usuario en la colección users de *Firebase Firestore* y el usuario aparecerá en la pantalla de *Buscar*. Por defecto, el nuevo usuario será administrador.

- Buscar

Nos encontramos en la pantalla principal de la aplicación.

Esta pantalla tiene dos objetivos. El primero es poder buscar un biquini concreto a través de su modelo o estampado para ver cuántos quedan de cada talla en la tienda y en el almacén. El segundo es poder consultar qué modelos existen para cada estampado y qué estampados existen para cada modelo top o bottom. De manera que si una clienta busca un modelo concreto, se le pueda decir con total exactitud en qué estampados está.

En esta pantalla, como podemos ver en la *imagen 6*, tenemos un *SearchView* para poder buscar directamente un estampado, top o bottom concreto. También hay un *TabHost* con tres secciones: estampados, tops y bottoms. En cada sección aparecen todos los estampados, tops o bottoms existentes.

Si mantengo presionado un estampado me aparecerá una lista con todos los modelos que existen para ese estampado. De igual manera, si mantengo presionado un top o bottom me aparecerá un lista con los estampados disponibles para ese modelo.

Si presiono un estampado iré a la pantalla de la *imagen 7* donde aparecen todos los modelos, tanto tops como bottoms, del estampado escogido.

Si presiono un top o bottom iré a la pantalla de la *imagen 8* donde está el modelo seleccionado con los distintos estampados disponibles.

Desde la pantalla de la *imagen 7 y 8* accedo a la última sub pantalla de Buscar donde encuentro los datos del biquini concreto seleccionado mostrado en la *imagen 9*.

IMAGEN 6 – Buscar

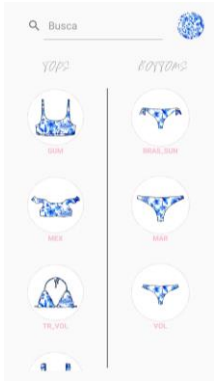


IMAGEN 7 – Estampado seleccionado

IMAGEN 8 – Modelo seleccionado

IMAGEN 9 – Biquini concreto

Todos estos datos están almacenados en *Firebase Firestore*. Los estampados se encuentran en la colección estampados. En cambio, los modelos no se guardan en ninguna carpeta especial. Es decir, para saber qué modelos existen el sistema accede a cada estampado guardando los modelos en un arraylist y mostrándolos.

Las imágenes de cada estampado y modelo se encuentran en *Firebase Storage*. Los estampados se encuentran en la carpeta estampados, los tops en modelos/tops y los bottoms en modelos/bottoms.

Desde la pantalla principal de Buscar, deslizo el dedo de izquierda a derecha y nos encontramos con el menú a través del cual puedo acceder a: Buscar, Tienda, Pedidos a tiendas, Notificaciones, Reservar, Usuarios, Reponer y Cerrar sesión. En todas estas pantallas se puede saltar de una a otra a través de este menú. El menú tiene una cabecera donde aparecen los datos del usuario conectado: la foto, el nombre, correo y tipo de usuario.

- Tienda

Este apartado pretende registrar y notificar las ventas de la tienda Bohodot para tener un control en todo momento.

Cada venta puede tener uno o más productos. Lo que implica tener un tíquet donde ir colocando los productos que se van a vender hasta finalizar la venta. Para acceder al tíquet tenemos que deslizar el dedo de derecha a izquierda. El tíquet implementado se muestra en la *imagen 10.*

Esta pantalla consiste en una lista de todos los estampados existentes de la misma forma que se hace en la sección estampados de la pantalla Buscar. Al seleccionar uno de los estampados aparecemos en otra pantalla como la *imagen 7* mostrando los distintos modelos agrupados por tops y bottoms. Al presionar uno de estos modelos aparecerán tres botones, *imagen 11,* que representan las tallas S, M y L con la intención de que escojas una de ellas.

En el momento de seleccionar un producto pueden pasar tres cosas. La primera es que el producto que se pide existe en la tienda, en ese caso se añade automáticamente al tíquet. La segunda es que el producto no exista en la tienda pero sí en el almacén, en

ese caso aparece un mensaje diciendo que se reserve el producto. El último caso es que el producto este agotado, en ese caso aparece un mensaje diciendo que está agotado. Para saber en qué caso nos encontramos el sistema accede al documento concreto del producto que se encuentra en la colección de bikinis de *Firebase Firestore.*

Para finalizar la venta se tiene que apretar el botón *pagar* disponible en el tíquet. En ese momento se registra en la colección tienda/{mes}/{semana} de *Firebase Firestore* la nueva venta y se limpia el tíquet pudiendo así seguir con una nueva venta.

Para no perder la venta que se está haciendo en el momento, cada vez que se añade un producto al tíquet se registra en un documento temporal en *Firebase Firestore* que se eliminará al pagar y finalizar la venta.

IMAGEN 10 – Tíquet

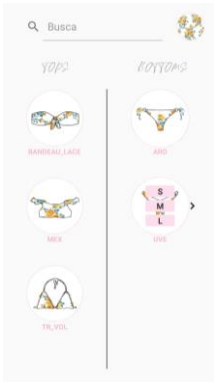


IMAGEN 11 – Producto seleccionado

- Pedidos a tiendas

El objetivo de esta pantalla es controlar y gestionar los pedidos de tiendas externas.

Inicialmente nos encontramos con la pantalla mostrada en la *imagen 12.* Hay una lista de todas las tiendas que tienen algún pedido en Bohodot. En esta misma pantalla, abajo a la derecha, hay un botón para añadir una nueva tienda. Por lo tanto desde esta pantalla podemos crear un pedido de una nueva tienda o consultar una tienda con un pedido existente.

Si seleccionamos una tienda iremos a todos sus pedidos, tanto pendientes como entregados. Podemos clicar uno de estos pedidos para verlo al detalle. Podemos verlo en la *imagen 13*.

Si accedemos al pedido concreto de una tienda obtenemos toda la información del pedido. En la cabecera tenemos el nombre de la tienda, la fecha de entrega y el teléfono de la tienda. A la derecha de la cabecera está el estado del pedido que puede ser: pedido, preparando, listo, enviar y enviado. Luego tenemos una lista con todos los productos pedidos que aparecen en verde o rojo en función de si tenemos stock suficiente para servirlo o no. Finalmente tenemos el precio total del pedido en euros y dos números justo debajo. Uno de color rojo que indica lo que no está pagado y uno de color verde que indica lo que está pagado.

Explicamos cada estado del pedido.

Pedido. Pedido hecho a la espera de que se tenga que preparar. Solo el super administrador puede cambiar su estado y ordenar que lo preparen.

Preparando. El pedido se tiene que preparar. Una vez preparado cualquier usuario puede cambiar su estado.

Listo. Pedido listo para enviar y a la esperar de que el super administrador dé la orden de enviar.

Enviar. El pedido se tiene que enviar. Una vez hecho, cualquier usuario puede decir que ha sido enviado. En este caso, el pedido pasa de estar en la lista de pendientes a la lista de entregados.

Cada vez que se cambia el estado, se notifica a los usuarios correspondientes. Se explica más al detalle en notificaciones.

Si queremos añadir una tienda nueva, tendremos que indicar su nombre, la fecha de entrega del pedido y el teléfono a través de una ventana emergente que aparece, véase en la *imagen 14*. Seguidamente hacer el pedido de una manera muy similar a la tienda a diferencia de que cuando selecciono un modelo aparece una ventana emergente donde tengo que poner cuantas tallas S, M y L quiero, mostrado en la *imagen 15*.

Todos estos datos se almacenan en *Firebase Firestore.* Cada tienda tiene su propia colección donde se guardan los pedidos pendientes en un documento y en otro los pedidos entregados.

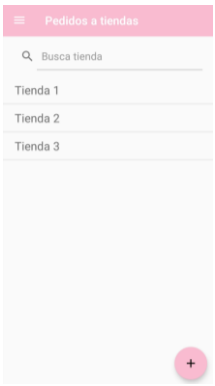


IMAGEN 12 – Pedidos a tiendas

IMAGEN 13 – Pedido concreto

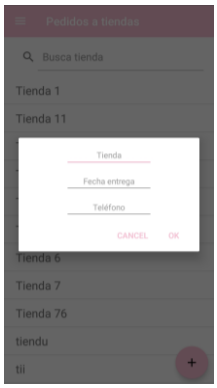


IMAGEN 14 – Ventana emergente datos tienda

IMAGEN 15 – Ventana emergente tallas

- Notificaciones

Como ya he comentado, en ciertas ocasiones, se notifica al usuario del evento ocurrido con el objetivo de estar informado en todo momento de los nuevos movimientos. En toda la aplicación hay siete notificaciones distintas de las cuales el super administrador recibe todas pero el administrador solo tres de ellas.

Para gestionar las notificaciones se guardan en *Firebase Firestore* en la colección de notificaciones donde se encuentran dos documentos: admin y super admin. Dentro de un documento irán las notificaciones que van destinadas a los administradores y en el otro a los super administradores. La notificación no se elimina de la base de datos hasta que todos los usuarios (admin o super admin) han leído esa notificación. Para saberlo, cada vez que un usuario lee esa notificación añade un campo con su identificador de usuario.

Hasta que no aparecen todos los identificadores en la notificación no desaparece de la base de datos.

Las notificaciones que aparecen listadas en esta pantalla son aquellas que el usuario todavía no ha leído.

Vemos notificación por notificación mostradas en la *imagen 16*.

Notificación al registrarse un nuevo usuario. Se avisa la super administrador de la nueva cuenta creada para estar siempre informado de los nuevos miembros. Al apretar vamos a los datos del nuevo usuario.

Notificación de un nuevo pedido. Se avisa al super administrador del nuevo pedido para que sea previsor y gestione bien el stock que queda o se pida hacer más repeticiones al taller si es necesario. Al apretar vamos a los datos del nuevo pedido.

Notificación de nueva reserva. Se avisa al super administrador que se tiene que llevar ese bikini en concreto a la tienda ya que lo reclaman. Al apretar vamos a los datos de la nueva reserva.

Notificación preparar pedido. Se envía a todos los usuarios para informar que se tiene que preparar el pedido con la intención de que a quien le toque realizar la tarea se encargue de ello. Al apretar vamos a los datos del pedido a preparar.

Notificación pedido listo. Cuando el pedido se ha preparado y está listo para enviar, se envía notificación al super administrador para estar informado del estado en todo momento. Al apretar vamos a los datos del pedido listo.

Notificación enviar pedido. Cuando el super administrador cree conveniente, da la orden de enviar el pedido. Esta notificación es recibida por todos los usuarios. Al apretar vamos a los datos del pedido a enviar.

Notificación bikinis repuestos. En cuanto llegan bikinis y se registran en la base de datos, todos los usuarios son notificados. Al apretar vamos a los datos del bikini que se ha repuesto.

IMAGEN 16 – Notificaciones

- Reservar

En este apartado se pretende reservar desde la tienda de Bohodot un producto que se ha agotado en la tienda y queda en el almacén. Todo producto reservado irá del almacén a la tienda para ser recogido por el cliente.

En esta pantalla aparece una lista con todas las reservas mostrado en la *imagen 17*. Puedo entrar dentro de la reserva para ver los detalles y para cambiar su estado si es el caso como se muestra en la *imagen 18.* También puedo crear una nueva reserva apretando el botón de abajo a la derecha y siguiendo un mecanismo igual al de la tienda.

En los detalles de la reserva hay una cabecera con el nombre de la reserva, teléfono, fecha del día de la reserva y el estado. Seguidamente los productos reservados y finalmente el precio total indicando si se ha pagado o no.

El estado de la reserva puede ser: preparando o listo. Desde que se hace la reserva hasta que llega el producto a la tienda, la reserva se está preparando. Cuando llega a la tienda el o los bikinis, la reserva está lista para ser entregada. Una vez entregada se elimina de la lista.

Los datos de las reservas se guarda en la colección reservas de *Firebase Firestore*. Se almacenan todas las reservas hasta que son entregadas, en ese momento se eliminan.

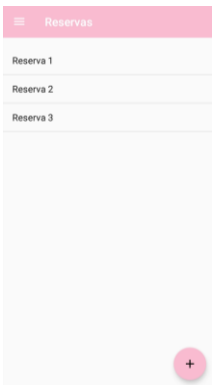


IMAGEN 17 – Reservas

IMAGEN 18 – Reserva concreta

- Usuarios

En este apartado el objetivo es que todos los usuarios puedan contactar entre ellos siempre que se necesite.

Esta pantalla, como se muestra en la *imagen 19*, tiene una lista de todos los usuarios con su nombre, correo electrónico y foto. Para facilitar la búsqueda hay un *SearchView* que permite buscar un usuario concreto por su nombre.

Todos los usuarios pueden acceder a esta lista pero solo los que tienen perfil de “super administrador” pueden apretar cualquiera de los usuarios de la lista y ver todos sus datos en la pantalla de la *imagen 20*.

En esta nueva pantalla se muestran los datos del usuario donde el super administrador puede modificar el nombre del usuario y el tipo de usuario. Para realizar estas modificaciones se aprieta en el dato correspondiente y aparecerá una ventana emergente donde se pondrán los nuevos datos y quedarán registrados en la base de datos de *Firestore*. Desde esta pantalla también es posible eliminar el usuario, en este caso, se eliminará el usuario de la base de datos y la función correspondiente de *Firebase Cloud Functions* eliminará el usuario de *Firebase Authentication.*

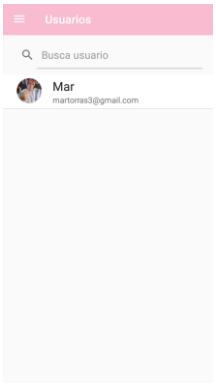


IMAGEN 19 – Usuarios

IMAGEN 20 – Usuario concreto

- Reponer

Cuando llegan bikinis del taller, es el momento de actualizar nuestra base de datos introduciendo los nuevos bikinis.

Cada bikini tiene guardada toda su información en *Firestore* y las imágenes del modelo y estampado almacenado en *Storage*. En caso de tener un bikini nuevo se tendrá que crear el documento y guardar las imágenes.

Nos podemos encontrar con varios casos aquí, los exponemos a continuación.

Bikini existente. Han llegado bikinis de repetición. En este caso, actualizo su documento en *Firestore*.

Modelo existente, estampado existente pero no ese modelo con ese estampado concretamente. En este caso la imagen del estampado y del modelo ya las tenemos en el *Storage* pero en *Firestore* tenemos que crear un nuevo documento con los datos para esta nueva combinación.

Modelo existente y estampado no existente. Tenemos que subir a *Storage* el nuevo estampado y crear un nuevo documento en *Firestore* con estos atributos.

Estampado existente y modelo no existente. Igual que en el caso anterior pero lo que subimos a *Storage* es el modelo nuevo.

- Cerrar sesión

Una vez dentro de la aplicación el sistema mantiene la sesión abierta del último usuario que ha accedido desde ese dispositivo.

En el menú deslizable tenemos la opción de cerrar nuestra sesión de manera que la próxima vez que intente acceder tendré que poner el correo y la contraseña.

- Extras

En diversas pantallas se han aplicado ciertas cosas que comento a continuación.

Tipo de letra. Para crear una aplicación más visual, en los títulos de las pantallas se han utilizado dos fuentes distintas. Para ello me he descargado sus archivos con

extensión *.ttf* correspondiente de internet y lo guardado en el directorio Assets de Android Studio. Para aplicarlo al texto he creado un *Typeface*.

Animación botones. Se ha creado un fichero XML para que al apretar un botón se cree una sensación de apretado.

Imagen circular. Para que las imágenes aparezcan con forma circular he implementado la librería *'de.hdodenhof:circleimageview:2.1.0'.*

Deslizar. Para crear los distintos tíquets y el menú principal, he implementado la librería *'com.android.support:design:28.0.0'.*

6.2. Firebase

Para utilizar los diferentes servicios de Firebase en el proyecto de Android Studio primero de todo se tiene que conectar el proyecto a Firebase. Una vez conectado, se implementa en el fichero *build.gradle* de la app los distintos servicios que se van a utilizar con su versión.

Las funciones de Firebase utilizadas en el proyecto son:

- Authentication

Firebase Authentication nos permite identificar a los usuarios y guardar sus datos en la nube de forma segura.

Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook y Twitter, y mucho más.

En esta aplicación he considerado la mejor opción identificarse mediante un correo y una contraseña. En Bohodot cada empleado tiene un correo de trabajo, la idea es entrar con ese correo. Es más, que solo se pueda entrar con @bohodot.es .

Para hacer referencia a este servicio de Firebase en Android Studio primero de todo se tiene que instanciar de la siguiente manera:

```
FirebaseAuth  fa = FirebaseAuth.getInstance();
```

Para crear una nueva cuenta usamos el siguiente método:

```
fa.createUserWithEmailAndPassword(correo, contraseña)
```

Para comprobar que esa cuenta ya existe:

```
fa.signInWithEmailAndPassword(correo, contraseña)
```

- Firestore

Es la base de datos de la aplicación. Se organiza en colecciones, documentos y campos del documento. En nuestra base de datos hay depositadas seis colecciones: bikinis, notificaciones, pedidos, reservas, tienda y users.

Bikinis. Guardamos la cantidad de stock que hay en cada estampados con sus respectivos modelos. Es decir, aquí está lo que queremos controlar y lo que vamos a ir consultando constantemente.

Notificaciones. Creado para la gestión de la notificaciones, para saber cuál he leído y cuál me falta por leer. Almacena todas las notificaciones que no han sido leídas por todos los usuarios.

Pedidos. Es donde se guardan todos los pedidos a tiendas según si está pendiente por entregar o si ya se ha enviado. En este apartado también nos permite que el tíquet implementado en Android tenga memoria y no se pierdan los productos pedios antes de acabar el pedido.

Reservas. Todas las reservas se guardan hasta ser entregadas al cliente en la tienda.

Tienda. Aquí almacenamos las ventas por semanas para así poder hacer en un futuro algún estudio. Saber qué se ha vendido implica saber qué puede faltar en la tienda, a pesar de esto, también lo podemos consultar en bikinis.

Users. Para guardar todos los datos de cada usuario y el "super administrador" poder acceder a ellos. De esta manera pueden pasar de un roll a otro roll.

Para hacer referencia a este servicio de Firebase en Android Studio primero de todo se tiene que instanciar de la siguiente manera:

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
```

En la aplicación escribimos y leemos constantemente en Firestore. Para escribir usamos el siguiente código:

```
db.collection("users").document("nombre").set(user);
```

"users" es la colección donde está el documento que quiero escribir y "*nombre"* es el documento donde quiero escribir. Es decir, users/nombre es el path donde voy a escribir. *user* es un Map<String, Object> donde están los datos que quiero guardar en la base de datos.

Para leer, utilizo el siguiente código:

```
db.collection("users").document("nombre").get();
```

Con este método obtengo el documento que está en el path users/nombre.

- Storage

En este servicio de Firebase podemos guardar varios tipos de archivos, entre ellos extensiones .png y .jpg. Se organiza según carpetas.

Para nuestra aplicación solo necesitamos almacenar imágenes. Hemos creado tres carpetas: modelos, estampados y usuarios.

Modelos. Guardamos las imágenes de todos los modelos, tanto tops como bottoms, existentes y es donde se guardaran los futuros modelos.

Estampados. De igual manera que en los modelos, necesitamos guardar los estampados existentes y es donde se guardaran los nuevos.

Usuarios. Es habitual tener una imagen asociada a tu perfil. En una empresa grande es normal porque no conoces a todos los empleados y en algún momento puntual te puede ser de ayuda. Bohodot es más bien una empresa pequeña y no haría falta. Aun así nunca está de más.

Para hacer referencia a este servicio de Firebase en Android Studio primero de todo se tiene que instanciar de la siguiente manera:

```
StorageReference mStorage = FirebaseStorage.getInstance();
```

Para subir una imagen uso el método:

```
mStorage.getReference().child("users").child("nombre").putFile(pickedImgUri);
```

users/nombre es el path donde guardo la nueva imagen. pickedImgUri es la Uri de la imagen seleccionada en la galería de fotos.

Para descargar una imagen lo hago a través de su url con el siguiente método:

```
mStorage.getReference().child("nombre_imagen").getDownloadUrl();
```

Con este método obtengo la url de la imagen desesada y a partir de un AsyncTask se descara la imagen en un segundo plano mientras la aplicación sigue funcionando.

- Cloud Functions

En Firebase hay varias funciones que no puedes llevar a cabo sin el SDK admin, para ello van muy bien el Cloud Functions.

Estas funciones se activan en respuesta a eventos y son externas al cliente y a su código. Se ejecuta todo el servidor de Firebase. Si una función falla el resto puede seguir funcionando.

Para desarrollar las funciones del Cloud Function he utilizado el editor de código Visual Studio Code y el terminal.

En este proyecto se ha implementado una función con el objetivo de eliminar un usuario concreto. En la plataforma Firebase puedes consultar el panel de control para ver qué funciones están vigentes y también puedes ver los registros de cada vez que se ha implementado una función, qué función y si ha salido con éxito.

Para trabajar con Cloud Funtion primero hay que instalas en tu entorno para poder trabajar con ellas. Desde el terminal lo instalamos en nuestro directorio de trabajo con el siguiente comando:

```
npm install -g firebase-tools
```

Luego, tenemos que entrar en nuestra cuenta de Firebase y en el proyecto que vamos a implementar estas funciones. Para hacerlo seguimos con este comando:

```
firebase login
```

Para instalar las dependencias necesarias en nuestro proyecto, lo hacemos de la siguiente manera:

```
firebase init functions
```

Una vez preparado el entorno donde desarrollaremos nuestra función, es el momento de implementarla. La función se llama elimnarUsuario y utiliza el método admin.auth().deleteUser(uid) para eliminar un usuario de Firebase Authentication. Esta función se activa cuando en Firestore se ha borrado un usuario. El código de la función lo podemos ver a continuación.

```
exports.eliminarUsuario = functions.firestore.document('users/{id}')
    .onDelete(dataSnapshot => {
        return admin.auth().deleteUser(dataSnapshot.data().uid)
            .then(function() {
                return console.log('Successfully deleted user');
            })
            .catch(function(error) {
                console.log('Error deleting user:', error);
            });
   });
```

Finalmente, cargamos la función en Firebase con el siguiente comando:

```
firebase deploy --only functions
```

La función ya está funcionando en nuestro proyecto. Para ver qué funciones tenemos en el proyecto lo podemos mirar en el panel de control que nos proporciona Firebase.

7. Costes

El coste estimado teniendo en cuenta los recursos humanos, los recursos materiales y herramientas software utilizadas.

- Recursos humanos

Si tenemos un salario bruto de 20.000 euros anuales, es decir, 18 euros la hora trabajada y las horas invertidas han sido 660 tenemos un total de 11.880 euros. Lo podemos ver en la tabla 2.

Salario bruto (€/año)	Euros/hora	Horas invertidas	Total (€)
20.000	18	660	18*660=11.880

TABLA 2 – Coste recursos humanos

- Recursos materiales

El material utilizado ha sido el ordenador con sus accesorios y un dispositivo móvil donde realizar las pruebas.

Hemos de tener en cuenta que estos materiales son amortizados. Si le ponemos una amortización de 5 años, obtenemos los datos siguientes. Si en un año se trabajan 1780 horas y, en este proyecto se ha invertido 660, tenemos una amortización inferior a la anual.

Material	Coste (€)	Amortización anual (€)	Amortización 660h (€)
Ordenador y accesorios	2.000	2.000*0,2 = 400	148
Dispositivo Android	500	500*0,2 = 100	185

TABLA 3 – Coste recursos materiales

- Herramientas software

Todas las herramientas utilizadas han sido gratuitas, incluida Firebase. En el caso de que esto se llegue a implementar, seguramente necesitaríamos acudir a la de pago.

- Costes totales

Para tener la estimación de costes de este proyecto tenemos que sumar los costes calculados anteriormente.

Coste total = 11.880 + (148+185) + 0 = 12.213 €

8. Conclusiones y propuestas de mejora

Hacer una aplicación Android se dice rápido y más ahora que está creciendo tanto este sector. Es cierto que puedes conseguir hacer una aplicación sencilla en relativamente poco tiempo pero cuando te sumerges en tu propio proyecto y te das cuenta de lo amplio que es este tema y la gran cantidad de cosas que puedes llegar a implementar en tu app, en ese momento, es cuando empiezas a ver que no todo es tan sencillo. Los limites los pones tú o, en este caso, también interviene y te condiciona según sus intereses y necesidades la empresa cliente.

He aprendido que sí, que toda idea es buena y no está de más apuntarla pero que no debes pecar de optimista y querer hacerlo todo desde un principio. Paso a paso. Primero te tienes que centrar en hacer la base de tu aplicación para cumplir con los objetivos y, una vez hecho, empezar con los detalles, las posibles mejoras y la parte extra a añadir.

Realizar este proyecto me ha ayudado a entender lo importante que es escuchar lo que pide el cliente para a la hora de desarrollar la aplicación todas las decisiones que se tomen estén lo mejor encaminadas posibles a cumplir sus necesidades y evitar posibles futuras quejas del cliente.

El objetivo principal que era desarrollar una aplicación para mantener el stock controlado en todo momento se ha cumplido.

Personalmente ha sido un trabajo muy satisfactorio ya que mis conocimientos acerca de Android Studio y Firebase eran totalmente nullos y considero que he conseguido crear una aplicación que, implementando alguno de las propuestas de mejora, puede llegar a introducirse en Bohodot.

Me quedo con la siguiente lección: “En muchos casos no se trata de trabajar más sino de trabajar mejor”.

8.1. Propuestas de mejora

El resultado de este proyecto no ha sido una aplicación Android perfecta. Una vez hecha la primera versión, se prepara la siguiente con todas las mejoras que se han pensado hasta el momento.

Las mejoras que haría en esta primera aplicación en orden de importancia son las siguientes:

- Crear almacenamiento interno

En este almacenamiento interno guardar todas las imágenes referentes a los productos de Bohodot ya que al tener que descargarlas de internet cada vez que accede a la pantalla, es un proceso que hace que la aplicación sea más lenta.

Estas imágenes de los productos no cambian muy constantemente y en el momento de cambiar notificar al usuario para que actualice el fichero donde se almacenan las fotos y tener siempre los datos actualizados evitando errores.

- Cloud Functions de Firebase

Hacer más uso de les Cloud Functions de Firebase. Todas las funciones que estén implementadas en la parte del servidor, mejor. Todo lo que esté en el servidor no se tendrá que encargar de hacerlo la aplicación. Optimizamos la app.

- Añadir más perfiles

No creo que sea algo super importante pero sí útil. Actualmente hay dos perfiles que son muy generales. Sería interesante crear perfiles más concretos de manera que quede más claro el rol de cada empleado y cuáles son sus tareas.

Por ejemplo, un posible perfil sería la persona encargada de estar en la tienda. De esta manera las notificaciones, en este caso, de la tienda llegarían a la dependienta y al resto de administradores que están trabajando en la oficina no ya que para ellos es información relevante.

- Conectar con la web

Este apartado es importante pero lo pongo el último porque ya he comentado que desde un inicio no lo iba a implementar ya que los conocimientos para hacerlo se me escapan ahora mismo. Pero para tener la base de datos de Firebase actualizada con los datos reales, tienen que estar las ventas online. Por lo tanto, queda pendiente.

Bibliografía

La información de la parte teórica del trabajo la he encontrado en diversos enlaces adjuntos a continuación:

1. www.bohodot.es

2. https://www.appannie.com/en/insights/market-data/app-annie-2017-retrospective/

3. https://computerhoy.com/reportajes/industria/android-vs-iphone-guerrasmartphones-cifras-271447

4. https://jarroba.com/wp-content/uploads/2015/05/Pila-de-arquitectura-Android-www.Jarroba.com_.png

5. https://firebase.google.com

6. https://es.wikipedia.org/wiki/Firebase

7. https://i2.wp.com/www.artit-k.com/wp-content/uploads/2016/07/Google-Firebase-Feature.png

Anexo

Mapa de las distintas pantallas de la aplicación. Empezando por la primera pantalla que aparece al abrir la app por primera vez, continuando por el menú y los distintos puntos de acceso y finalmente cada ítem del menú con sus pantallas.

Las flechas de los mapas indican que pasa de la pantalla origen a la pantalla destino. Las que son de color negro indican que ha habido este cambio debido al hacer click en algún *view* de la pantalla origen. En cambio las verdes es debido al apretar el botón de atrás.

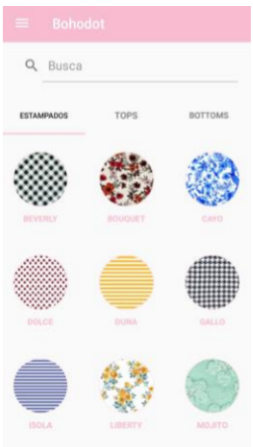
El icono de la aplicación es el logo de Bohodot:

Primera pantalla

Iniciar sesión

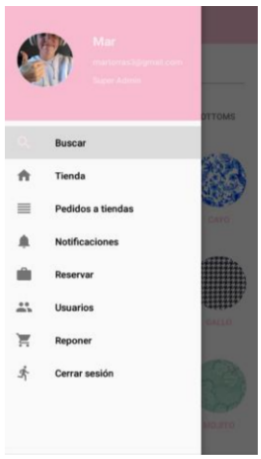
Registrar nuevo usuario

Pedir acceso a la galería de imágenes



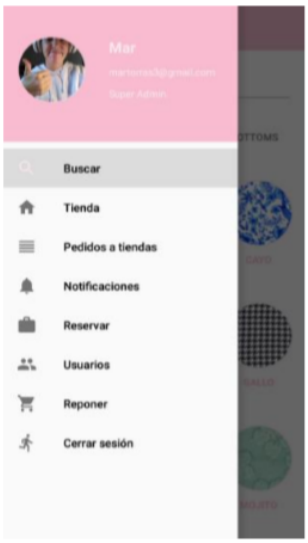
Buscar

(Pantalla principal)



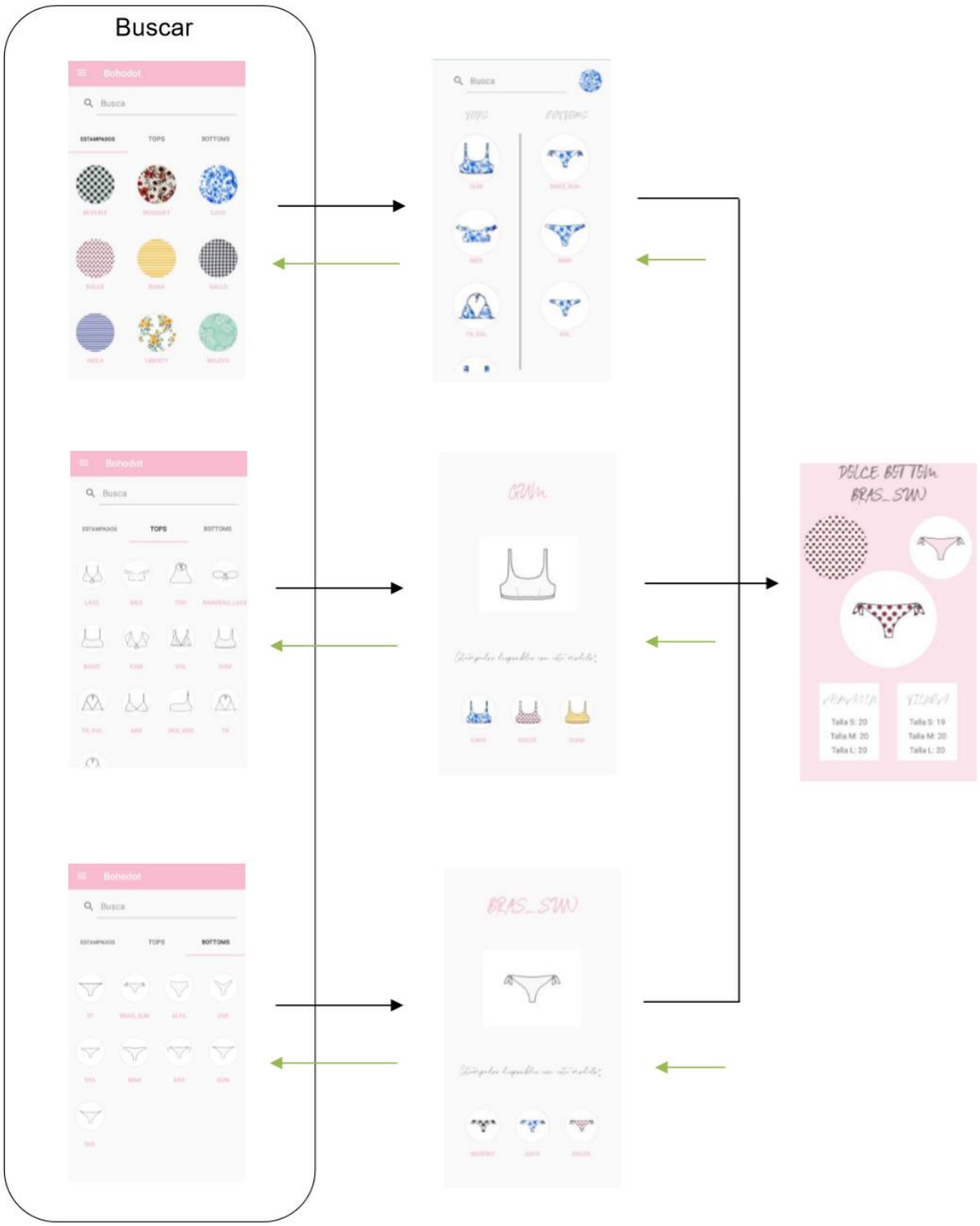
Deslizando de izquierda a derecha

Menú deslizable

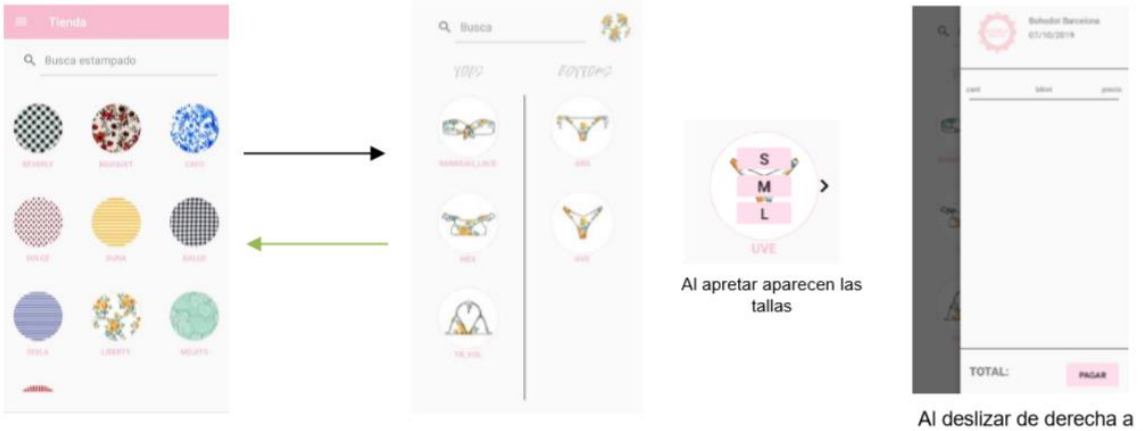


Menú deslizable

Buscar


Buscar
Bohodot
Busca
ESTAMPADOS
TOPS
BOTTOMS
Bohodot
Busca
ESTAMPADOS
TOPS
BOTTOMS
Bohodot
Busca
ESTAMPADOS
TOPS
BOTTOMS
Busca
TOPS
BOTTOMS
GUM
BRAS_SUM
Talla S: 20
Talla M: 20
Talla L: 20
Talla S: 19
Talla M: 20
Talla L: 20
BRAS_SUM

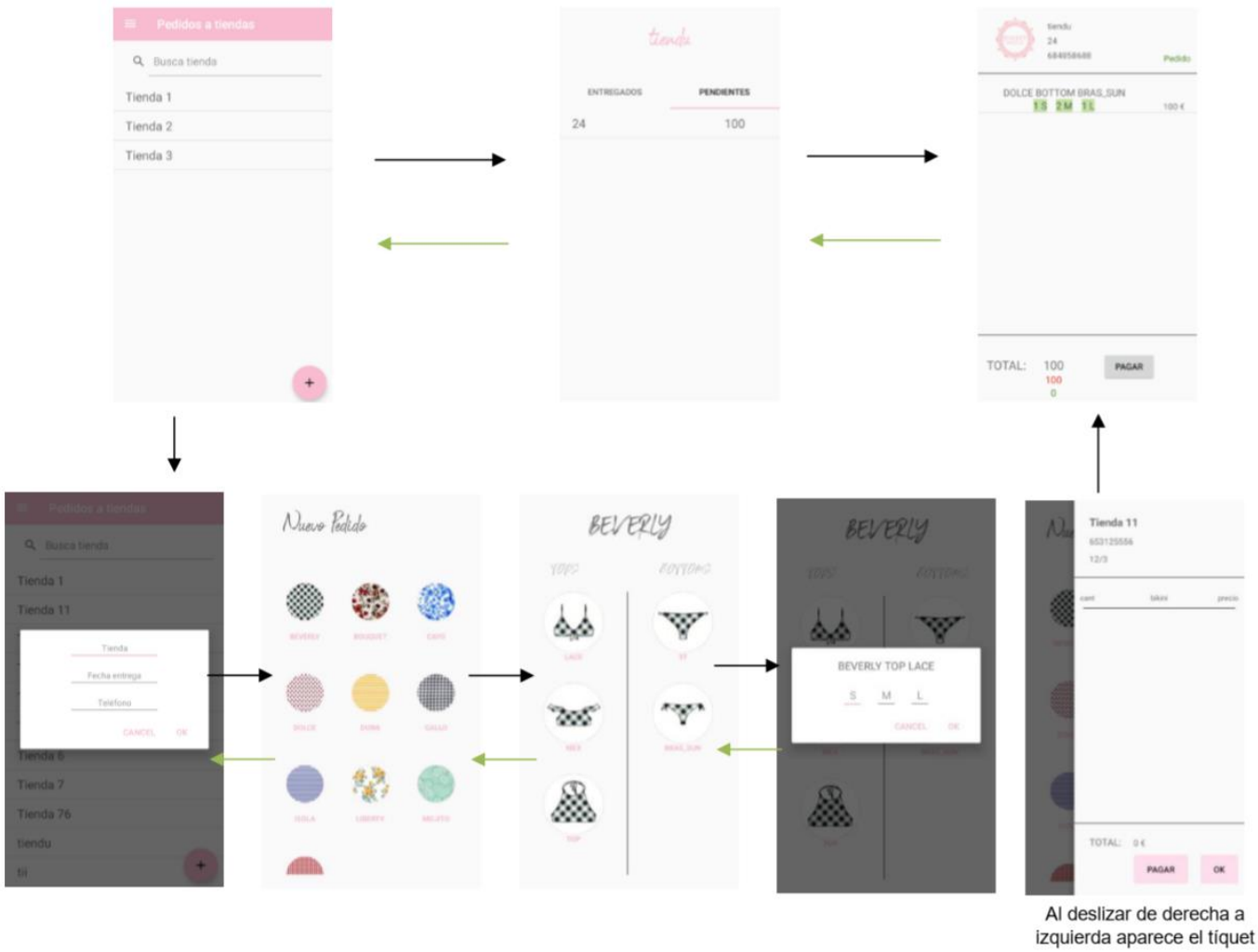
Tienda



Al apretar aparecen las tallas

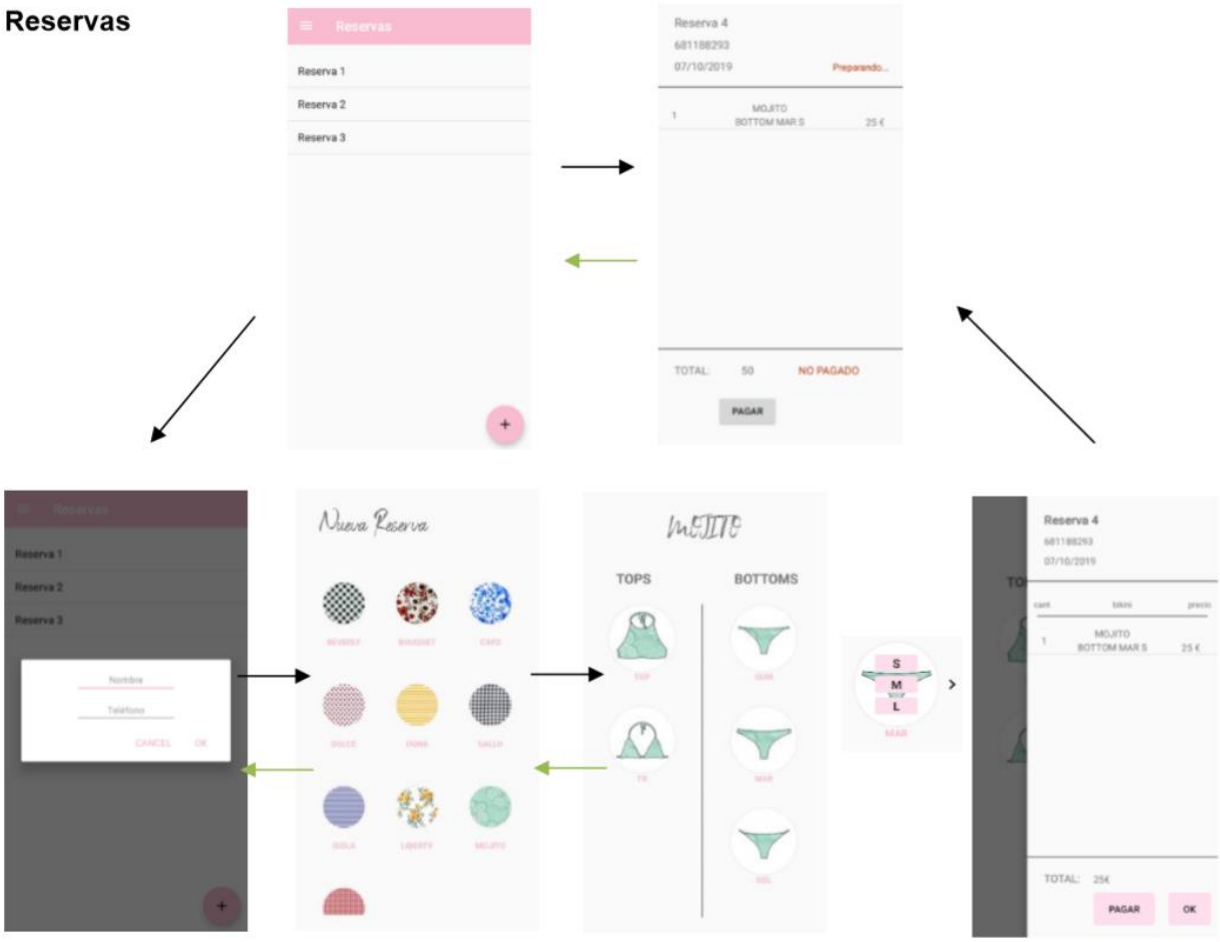
Al deslizar de derecha a izquierda aparece el tíquet

Pedidos a tiendas



Al deslizar de derecha a izquierda aparece el tíquet

Reservas



Usuarios