PROTOTIPO SOFTWARE PARA EL COMERCIO ELECTRÓNICO B2C EN MICROEMPRESA DE VENTA DE ROPA DE LA CIUDAD DE BUCARAMANGA

DIEGO FERNANDO MEDINA BLANCO
LUIS ERNESTO PÁEZ ORTIZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2020

PROTOTIPO SOFTWARE PARA EL COMERCIO ELECTRÓNICO B2C EN MICROEMPRESA DE VENTA DE ROPA DE LA CIUDAD DE BUCARAMANGA

DIEGO FERNANDO MEDINA BLANCO
LUIS ERNESTO PÁEZ ORTIZ

Una tesis presentada en cumplimiento de los requisitos para el grado de:
Ingeniero de Sistemas e Informática

Director:
FERNANDO ANTONIO ROJAS MORALES
Magíster en Ciencias Computacionales

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2020

AGRADECIMIENTOS

Los autores expresan su agradecimiento:

A mi madre, Belén Blanco Grast, por su cariño y apoyo día a día.

A mi padre, Gustavo Eduardo Medina Ramos, por su guía, supervisión, dirección y aporte desde su experiencia como docente, director y calificador de proyectos, garantizando que este proyecto se llevara a cabo, y tuviera una alta calidad.

A mis hermanos, Juan Camilo Medina Blanco, por su disposición y apoyo desde un rol de tester y analista, y David Eduardo Medina Blanco, por su atención e incondicional apoyo.

A nuestro director de proyecto, Fernando Antonio Rojas Morales, el cual tomó su rol a pecho y con su paciencia, conocimiento y vocación nos guió desde el comienzo, estuvo para ayudarnos cada momento que lo necesitamos con una prontitud impecable.

A mis compañeros, amigos y colegas de Ingeniería de Sistemas, por brindarme recursos, consejos, ayuda, y parte de su tiempo para motivarme y enriquecer este proyecto, principalmente Oscar, Douglas, Henry, Edgar, Nicolás, William, Andrés, entre otros.

A mis amigos, por su apoyo emocional para motivarme.

Por ultimo y no menos importante, a la escuela de Ingeniería de Sistemas e Informática(EISI), por brindarme las herramientas y los medios para formarme como profesional.

Diego Fernando Medina Blanco

He culminado esta maravillosa etapa de mi vida.

Siendo la vida un tesoro para vivir agradecido y siendo este espacio una oportunidad para inmortalizar agradecimientos, quiero dar una principal mención a Dios y a María, quienes en medio de su infinita misericordia me han bendecido a pesar de ser un hombre que adueña innumerables defectos. A Ellos debo mi pasado, mi presente y mi futuro.

En segundo lugar, quiero agradecer a mi madre, Cecilia Ortiz, quien es el ser humano más maravilloso, misericordioso y amable que he conocido. Gracias por formar a este hombre en valores, virtudes y responsabilidades junto a mi padre, Luis Enrique Páez Capacho, hombre echado para adelante, con un corazón humilde y deseoso de amar a los suyos. Gracias papá.

En tercer lugar, quiero dedicar este espacio a Tite, mi hermano, hombre e ingeniero lleno de resiliencia y valentía para asumir nuevos retos, quien nunca duda y siempre confía en mí.

Por último, agradezco a las personas que han hecho parte de este proceso en específico: a mi profesora de matemáticas del colegio; a mi entrenador de Atletismo; a mis profesores de ingeniería; a mis primos y primas quienes siempre han estado para mí; gracias a mis colegas y compañeros de carrera; a mis amigos Camilo, Jose Daniel, Jorge Andrés y Andrea, seres humanos intachables y hermanos de vida; a mi pareja María Paula, quien es una persona que derrocha amor en cada palabra y me hace sentir bendecido; a mis amigos por su acompañamiento emocional; a mis guías espirituales Dina, Alexander y Luswin, y a todos aquellos que en este preciso instante, quizás, no logro recordar.

Gracias, estoy infinitamente agradecido porque me siento alguien valioso, en parte, por la presencia de cada uno de ustedes en mi vida para lograr este gran objetivo.

Luis Ernesto Páez Ortiz

Índice general

Pág

Índice de figuras

Pág

RESUMEN

TÍTULO: PROTOTIPO SOFTWARE PARA EL COMERCIO ELECTRÓNICO B2C EN MICROEMPRESA DE VENTA DE ROPA DE LA CIUDAD DE BUCARAMANGA [*]

AUTORES: DIEGO FERNANDO MEDINA BLANCO, LUIS ERNESTO PÁEZ ORTÍZ [**]

PALABRAS CLAVE: ECOMMERCE, FLUTTER, DESARROLLO, SOFTWARE, PROTOTIPADO, EVOLUTIVO, SCRUM.

DESCRIPCIÓN: El comercio electrónico es un tema de discusión y aplicación de alto crecimiento y demanda en el mercado nacional. Las compras realizadas a través de los canales virtuales de las compañías permiten salvar tiempo y recursos físicos a la hora de realizar una venta. MOI Colombia es una compañía santandereana que tiene como objetivo comercializar ropa de moda para mujeres. La compañía comercializa sus productos a través de su tienda física y sus canales virtuales tales como página web y redes sociales. Con el fin de expandir su capacidad de venta electrónica, se hizo la propuesta del desarrollo de una solución software de tipo móvil para el comercio electrónico de la compañia, disponible para los sistemas operativos iOS y Android que permite administrar los pedidos y productos de la compañía, así como mostrarlos a los clientes, dándoles la posibilidad de realizar pedidos bajo distintos escenarios a través de una interfaz amigable, generando la factura correspondiente a dicho pedido, además de otras funcionalidades que faciliten el proceso de comercio electrónico. Con el desarrollo de esta solución se busca mejorar el proceso de venta virtual, facilitando la compra de prendas a la empresa MOI Colombia. Adicionalmente, con la realización de este proyecto se pretende explorar el framework Flutter, kit de herramientas para desarrollo de aplicaciones móviles de la empresa Google, como herramienta de desarrollo.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Fernando Antonio Rojas Morales, magíster en Ciencias Computacionales

ABSTRACT

TITLE: SOFTWARE PROTOTYPE FOR B2C E-COMMERCE IN CLOTHING MICROENTERPRISE IN THE CITY OF BUCARAMANGA [*]

AUTHORS: DIEGO FERNANDO MEDINA BLANCO, LUIS ERNESTO PÁEZ ORTÍZ [**]

KEYWORDS: ECOMMERCE, FLUTTER, DEVELOPMENT, SOFTWARE, PROTOTYPING, EVOLUTIONARY, SCRUM

DESCRIPTION: E-commerce is a topic of discussion and application of high growth and demand in the national market. The purchases made through the virtual channels of the companies allow to save time and physical resources when making a sale. MOI Colombia is a company from Santander that aims to market fashionable clothing for women. The company markets its products through its physical store and its virtual channels such as website and social networks. In order to expand its electronic sales capacity, a proposal was made for the development of a mobile-type software solution for the company's e-commerce, available for the iOS and Android operating systems, which allows the company to manage its orders and products, as well as to show them to customers, giving them the possibility to place orders under different scenarios through a friendly interface, generating the corresponding invoice for that order, in addition to other functionalities that facilitate the e-commerce process. With the development of this solution, we seek to improve the virtual sales process, facilitating the purchase of garments from MOI Colombia. Additionally, with the realization of this project, we intend to explore the Flutter framework, a toolkit for the development of mobile applications from the company Google, as a development tool.

---

[*] Degree work

[**] Faculty of Physical-Mechanical Engineering. School of Systems and Computer Engineering. Advisor: Fernando Antonio Rojas Morales, master's in Computer Science

INTRODUCCIÓN

La compañía de venta de ropa femenina MOI Colombia, la cuál para este documento será nombrada como MOI, es una empresa santandereana consolidada en la venta al por menor de ropa femenina a la moda.

Con base en la necesidad de expandir sus canales de venta y permitir a sus clientes conocer sus productos de manera efectiva, se ha desarrollado MOI y MOI Admin, dos aplicaciones móviles encargadas de gestionar el proceso de venta y presentación de artículos disponibles en la tienda. La aplicación para el uso de los clientes permite a los clientes de MOI conocer los productos, precios, colores y tallas, así como realizar pedidos en un proceso de venta amigable. Por otro lado, la aplicación para el uso de la administración permite realizar una completa gestión de los productos y pedidos de la empresa.

Adicionalmente, este proyecto utiliza el framework Flutter de Google y los servicios de Firebase, buscando explorar estos elementos del mundo de la programación e ingeniería contemporáneo y las posibilidades que brindan como herramientas de desarrollo.

Este documento tiene como objetivo mostrar una justificación detallada del problema, el marco teórico, la metodología junto con al arquitectura de software utilizada, y las practicas específicas implementadas en el ámbito conceptual y técnico para el desarrollo y construcción de este proyecto.

1. PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

En el contexto del comercio nacional, las soluciones por comercio electrónico han crecido a buen ritmo. Para el 2016, el country manager de Mercado Libre revelaba que cada año las ventas en materia de comercio electrónico aumentaban un 50 %[1]. Además, para finales de 2017, la Federación Nacional de Comerciantes (FENALCO) en Bucaramanga, Colombia, expresaba que la implementación de soluciones tecnológicas en materia de comercio electrónico aumentaría hasta un 30 % el ingreso en ventas de, no solo microempresas tales como un supermercado, sino también de las pymes bumanguesas en general[2]. Las soluciones de comercio electrónico implicarían un apoyo importante para la economía microempresarial colombiana, teniendo en cuenta que, entre otras ventajas, se encuentra la facilidad en el procedimiento de compra y venta, búsqueda de productos y la falta de limitación geográfica teórica[3]. Además, estadísticamente se encuentra una relación entre la facilidad en el procedimiento de compra (facilidad de uso de la solución) y la decisión de un cliente para usar o no usar la solución[4]. Es principalmente esta relación la que sugiere dos consideraciones: 1. Que un prototipo software para el comercio electrónico ha de ser un camino útil para comerciar entre una microempresa de venta de ropa sin

---

[1] REVISTA DINERO. *¿Comprar por internet? Los colombianos aún no se deciden.* 2016. URL: `https://www.dinero.com/pais/articulo/como-va-el-comercio-electronico-en-colombia/232305` (visitado 17-03-2019).

[2] VANGUARDIA LIBERAL. *Comerciantes de Santander dan paso hacia el E-commerce.* 2017. URL: `http://www.vanguardia.com/economia/local/410132-comerciantes-desantander-dan-paso-hacia-el-e-commerce` (visitado 17-03-2019).

[3] NIRANJANAMURTHY, M, et al. "Analysis of e-commerce and m-commerce: advantages, limitations and security issues". En: *International Journal of Advanced Research in Computer and Communication Engineering* 2.6 (2013), págs. 2360-2370.

[4] MESÍAS TAVERA, Juan F; GIRALDO SÁNCHEZ, Juan C y DÍAZ BALLESTEROS, Bernardo. "Aceptación del e-commerce en Colombia: un estudio para la ciudad de Medellín". En: *Revista Facultad de Ciencias Económicas: Investigación y Reflexión* 19.2 (2011), págs. 9-23.

solución móvil disponible y sus clientes (B2C), teniendo en cuenta que la relación entre mercar y facilidad para hacerlo es un factor diferenciador[5] y 2. Que el uso de una tecnología emergente para el desarrollo móvil como Flutter SDK permitiría la cobertura de una mayor población de clientes (aplicación nativa en dispositivo inteligente), disponiendo de su virtud como única tecnología para el desarrollo multiplataforma nativo con compilación directa en el dispositivo[6]. En consecuencia, con este proyecto se busca desarrollar un prototipo software que implemente soluciones de comercio electrónico de tipo empresa a cliente (B2C) en una microempresa de venta de ropa, exceptuando el pago electrónico, basados en la importancia de facilitar al cliente de esta industria su interacción con la compañía mediante el uso de Flutter como kit de desarrollo de software. Flutter es una tecnología emergente, por lo que con la realización de este proyecto se pretende explorar esta herramienta del estado del arte, para así verificar su viabilidad, utilidad y explorar tanto sus beneficios como sus desventajas y limitaciones.

1.1. ALCANCE

MOI es una tienda física que realiza ventas en su domicilio físico y canales virtuales como redes sociales y página web. Sus clientes se caracterizan por ser del sexo femenino. Este proyecto plantea una solución para fortalecer el canal de ventas virtuales a través de una solución software compuesta de una aplicación móvil funcional para los sistemas operativos iOS y Android que permita a los clientes de MOI conocer el catálogo de productos, realizar pedidos y guardar productos deseados para futuras compras, y una aplicación móvil funcional para la administración del negocio, con la estructuración y cargue del catálogo de productos, gestión de pedidos, descuentos y . Tales acciones fueron posibles a través de la implementación de una aplicación para la gestión de productos, inventarios y pedidos en la tienda. Para esta solución software se

---

[5] AGUDELO MONTOYA, César Alveiro y SAAVEDRA BOYERO, Martín Ramiro. "El CRM como herramienta para el servicio al cliente en la organización". En: *Visión de futuro* 17.1 (2013), pág. 5.

[6] DAGNE, Lukas. "Flutter for cross-platform App and SDK development". En: (2019).

cuenta con un inventario virtual, el cuál es una parte del inventario real de la empresa, definida de acuerdo al criterio del administrador que haga uso de la solución. Esta solución no cuenta con contabilidad ni integración con un sistema externo de contabilidad, así como tampoco cuenta con conexión a un sistema externo de facturación. Este proyecto no abarca la implementación de pagos electrónicos (uso de tarjetas de crédito, debito y de canales de pago virtuales) debido a la falta de oferta de Kit de Desarrollo de Software (SDK) y poco soporte por parte de empresas con presencia en Colombia de transacción segura en relación con Flutter SDK, así como la implementación de la facturación electrónica establecida por la Dirección de Impuestos y Aduanas Nacional (DIAN), ente que rige la tributación nacional.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Desarrollar un prototipo software para el comercio electrónico tipo negocio a cliente (B2C) para microempresa de venta de ropa utilizando Flutter como kit de desarrollo de software.

2.2. OBJETIVOS ESPECÍFICOS

1. Definir los requerimientos de software a partir de la necesidad del cliente mediante el uso de la técnica de desarrollo ágil SCRUM
2. Proponer un diagrama de arquitectura de software.
3. Diseñar un prototipo software en base a los requerimientos definidos y que responda a las siguientes funcionalidades básicas:
   a) Presentación del catálogo de productos.
   b) Edición del catálogo de productos.
   c) Capacidad para vincular varios productos a una canastilla de compra, controlando el total de la compra en cantidades y precios.
   d) Facturación
4. Validar consecuentemente cada prototipo hasta encontrar un equilibrio entre los objetivos del proyecto y la necesidad del cliente.

3. MARCO TEÓRICO

3.1. CÓDIGO DE COLOR HEXADECIMAL

Código estándar para la representación en el espacio RGB del color en 3 pares hexadecimales, el primero representando la intensidad del rojo, el segundo la intensidad del verde, y el tercero la intensidad del azul. Se lo suele anteceder con el símbolo '# ' o '0X' para indicar que es un código de color hexadecimal, ampliamente utilizado en el diseño, desarrollo web, HTML.[7]

3.2. CSV

Un CSV es un archivo de valores separados por comas, que permite guardar los datos en un formato tabular. Los CSV se parecen a una hoja de cálculo de jardín pero con una extensión .csv. Estos archivos sirven para diferentes propósitos comerciales. Por ejemplo, ayudan a las empresas a exportar un gran volumen de datos a una base de datos más concentrada. En el mundo del comercio en línea, uno de sus principales objetivos es llegar a un gran número de clientes. Ya que los archivos CSV son fáciles de organizar, los dueños de negocios de comercio electrónico pueden manipular estos archivos de muchas maneras diferentes. Los archivos CSV se utilizan principalmente para importar y exportar información importante, como datos de clientes o pedidos, hacia y desde su base de datos. [8]

---

[7] HTML COLOR CODES INFO. *Teoría sobre los códigos de colores HTML*. 2020. URL: `https://html-color-codes.info/codigos-de-colores-hexadecimales/` (visitado 15-06-2020).

[8] BIGCOMMERCE. *What is a .CSV file and what does it mean for my ecommerce business?* 2020. URL: `https://www.bigcommerce.com/ecommerce-answers/what-csv-file-and-what-does-it-mean-my-ecommerce-business/` (visitado 26-06-2020).

3.3. TIEMPO UNIX

El tiempo Unix es el número de segundos que han transcurrido desde el comienzo de la Época Unix. Este tiempo de la Época Unix es un contador cada vez mayor que sube cada segundo y muestra el número de segundos transcurridos desde las 00:00:00 UTC del 1 de enero de 1970. Estos números tan grandes son bastante inútiles en su forma cruda para las tareas de confirmar la hora y fecha actuales, pero son perfectos para medir entre dos puntos en el tiempo, es más fácil y rápido encontrar la diferencia entre dos sellos de tiempo Unix que calculan lo mismo entre dos sellos de tiempo legibles para los humanos. [9]

3.4. E-COMMERCE

Se dice que el término o denominación e-commerce reúne todas las posibles formas de transacciones de un negocio a través de datos electrónicos. El diario The Economist dice que "está más estrechamente identificado con el comercio transmitido a través del internet, y es el internet quien ha puesto al e-commerce en los primeros lugares de la agenda corporativa en los primeros años del siglo XXI"[10].

3.4.1. E-commerce B2C El comercio electrónico de negocio a cliente (Business to customer -B2C), es un tipo de comercio electrónico [3] en internet donde se realizan transacciones entre una compañía y un cliente.

---

[9] UNIX TUTORIAL. *Unix Epoch*. 2020. URL: `https://www.unixtutorial.org/unix-epoch/` (visitado 26-06-2020).

[10] THE ECONOMIST. *E-commerce*. 2009. URL: `https://www.economist.com/news/2009/10/08/e-commerce` (visitado 20-03-2019).

3.5. MICROEMPRESA

Diferentes gobiernos del mundo, como los países pertenecientes a la Unión Europea, suelen agrupar las microempresas en un conjunto de organizaciones catalogadas como pyme, que en nuestro país el Departamento Nacional de Planeación (DNP) las reconoce como mipymes[11].
Pyme es una abreviatura a la categoría empresarial que ocupan las Pequeñas y Medianas Empresas. La Unión Europea define a la categoría pyme como aquella que "está constituida por las empresas que ocupan a menos de 250 personas y cuyo volumen de negocios anual no excede de 50 millones de euros o cuyo balance general anual no excede de 43 millones de euros"[12].
En Colombia, el concepto no es distante de la definición de la organización europea, donde Mipyme abrevia las Micro, Medianas y Pequeñas Empresas y la ley 905 de 2004 en su artículo 2, las define en tres distintas formas: mediana empresa entre 51 y 200 empleados o activos entre 5001 y 30000 salarios mínimos legales vigentes; pequeña empresa entre 11 y 50 trabajadores o activos entre 501 y 5000 salarios mínimos legales vigentes; microempresa no más de 10 trabajadores o activos no superiores a 500 salarios mínimos legales vigentes excluida la vivienda[13].

3.6. FLUTTER SDK

Google LLC define Flutter como el kit de herramientas de interfaz de usuario de Google para realizar aplicaciones hermosas, que compilan nativamente, para móvil, web y escritorio desde un único código fuente.[14]

---

[11] NIETO, Victor, et al. "La clasificación por tamaño empresarial en Colombia: Historia y limitaciones para una propuesta". En: *Archivos de economía* 434 (2015).

[12] UNIÓN EUROPEA. "Recomendación de la Comisión, del 6 de mayo de 2003, sobre la definición de microempresas, pequeñas y medianas empresas". En: *Diario Oficial de la Unión Europea L* 124 (2003), pág. 20.

[13] CÁMARA DE COMERCIO DE BOGOTÁ, et al. "Ley 905 de 2004". En: (2004).

[14] GOOGLE LLC. *Flutter SDK*. 2019. URL: `https://flutter-es.io/` (visitado 20-06-2020).

3.7. FIREBASE

Google LLC define Firebase como una pataforma completa de desarrollo de aplicaciones, que permite compilar apps rápido, sin necesidad de administrar la infrastructura, con el respaldo de Google y la confianza de apps reconocidas, una plataforma con productos que funcionan mejor en conjunto.[15]

3.7.1. Firebase Realtime Database Firebase Realtime Database es una base de datos NoSQL alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando se compilan apps multiplataforma con los SDK de iOS, Android y JavaScript, todos los clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.[16]

3.7.2. Firebase Authentication Firebase Authentication es un servicio de autenticación que proporciona servicios de backend, SDK y bibliotecas de IU ya elaboradas para autenticar a los usuarios en las apps, con soporte y robustez para su integración con Flutter.Se integra estrechamente en otros servicios de Firebase y aprovecha los estándares de la industria como OAuth 2.0 y OpenID Connect, por lo que se puede integrar fácilmente con backends personalizados.[17]

3.7.3. Firebase Cloud Storage Firebase Cloud Storage es un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google, tanto que los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y

---

[15] GOOGLE LLC. *Firebase*. 2019. URL: `https://firebase.google.com/` (visitado 14-06-2020).

[16] GOOGLE LLC. *Firebase Realtime Database*. 2020. URL: `https://firebase.google.com/docs/database` (visitado 14-06-2020).

[17] GOOGLE LLC. *Firebase Authentication*. 2020. URL: `https://firebase.google.com/docs/auth` (visitado 25-06-2020).

descarga de archivos para las apps que utilizan Firebase, sin importar la calidad de la red. Cloud Storage almacena los archivos en un depósito de Google Cloud Storage y los hace accesibles a través de Firebase y Google Cloud. [18]

3.8. PROTOTIPADO SOFTWARE EVOLUTIVO

El prototipado software se define como el proceso de desarrollar una réplica funcional de un producto software previamente diseñado. Esta práctica es realmente popular a la hora de desarrollar software comercial, puesto que permite al comprador conocer su producto poco a poco sin la obligación de tener completamente claro el producto que está buscando.
El prototipado de software evolutivo es un método de prototipado de software que permite el constante diálogo entre el cliente y el desarrollador para refinar los resultados del prototipo hasta que este es aceptado[19].

3.9. ARQUITECTURA REACTIVE

La arquitectura reactiva es la combinación de la programación reactiva con arquitecturas de software[21], de tal forma que dicha arquitectura que permita operar al lenguaje de programación imperativa en secuencias de datos independientemente de si los datos son síncronos o asíncronos, basado en la propagación de cambios.

---

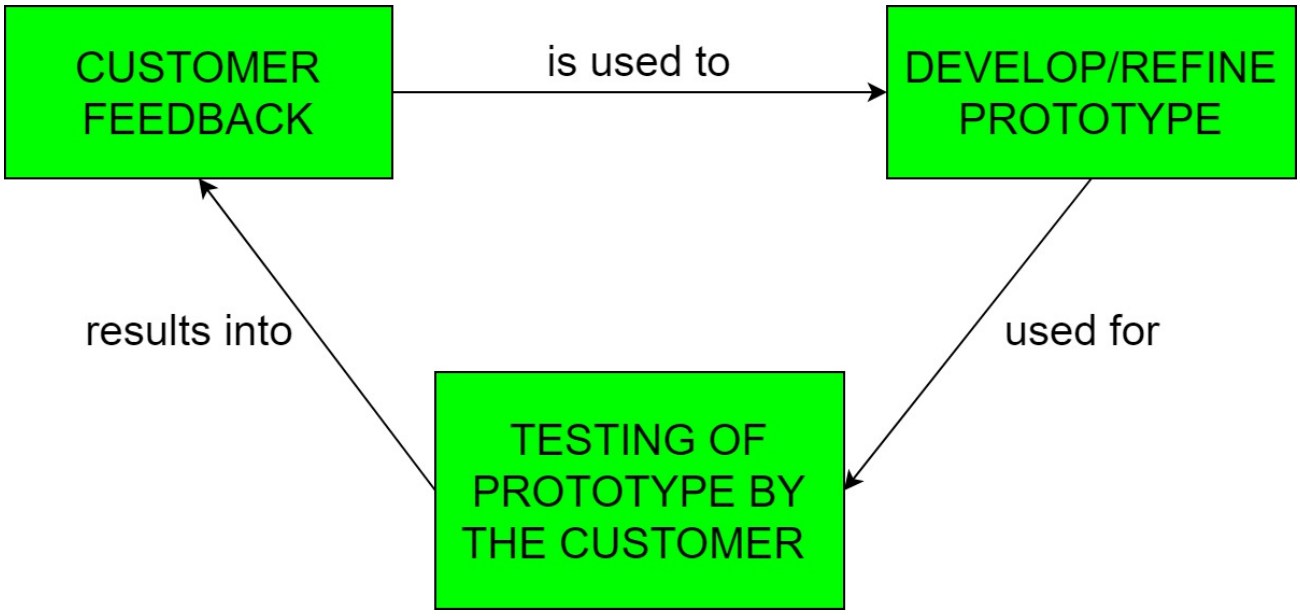
[18] THOMPSON, Arthur. *firebase_storage*. Ver. 0.16.1. 25 de mayo de 2020.

[19] GEEKS FOR GEEKS. *Software Engineering | Prototyping Model*. 2019. URL: `https://www.geeksforgeeks.org/software-engineering-prototyping-model/` (visitado 21-03-2019).

[21] LEONARDIS, Dan. *Reactive Architecture*. 2020. URL: `https://android.jlelse.eu/reactive-architecture-7baa4ec651c4?gi=182b1cd478e7` (visitado 28-06-2020).

Figura 1. Software Engineering | Prototyping Model

20



3.10. PROGRAMACIÓN FUNCIONAL

La Programación Funcional es un paradigma de programación declarativo que está basado en funciones matemáticas. Utiliza expresiones condicionales y recursividad en vez de la ejecución de sentencias imperativas, evitando los conceptos de estados compartidos y valores mutables[22]. Es el proceso de construir software al componer funciones puras, con la posibilidad de construir funciones de orden superior.

3.11. PRINCIPIO DE ENCAPSULAMIENTO

En la programación orientada a objetos, la encapsulación es un atributo del diseño de objetos. Significa que todos los datos del objeto están contenidos y ocultos en el objeto y el acceso a

---

[22] CHTIOUI, Mahdi. *ReactiveX: Reactive Programming Principles*. 2020. URL: `https://medium.com/@mahdichtioui/reactivex-reactive-programming-principles-dbb1bafa8384` (visitado 28-06-2020).

ellos está restringido a los miembros de esa clase[23].

3.12. PATRONES DE DISEÑO

3.12.1. Patrón BLoC El patrón de diseño BLoC (Business Logic Components) permite manejar el estado de los widgets y hacer accesible los datos desde un lugar central en el proyecto. Permite que la lógica del negocio esté centraliza en componentes separados a la interfaz de usuario, de tal forma que estos sean reutilizables y accesibles por toda la aplicación[24].

3.12.2. Patrón Provider El patrón de diseño Provider para flutter permite a un Widget proveer datos a un modelo mientras que otro Widget está escuchando a los cambios de ese mismo modelo[25]. Procura proveer un punto de acceso general en el árbol de Widgets, exponiendo los objetos que se necesitan usar[26].

3.12.3. Patrón Singleton El patrón de diseño Singleton proporciona una solución al problema de la multiplicidad de objetos en ejecución que deberían ser únicos. De esta forma, este patrón permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Así, su intención es garantizar que una clase solo tenga una instancia, y todo en el sistema que necesita a esta clase interactúa con este único objeto. El acceso de esta instancia debe ser accesible para todos los clientes.

---

[23] BOLTON, David. *Definition of Encapsulation in Computer Programming*. 2020. URL: `https://www.thoughtco.com/definition-of-encapsulation-958068` (visitado 28-06-2020).

[24] MORTON, J y ODELL, JJ. *Object oriented analysis and design*. Englewood Cliffs (New Jersey): Prentice-Hall, 1992.

[25] LOCAL DROID. *Widget Communication in Flutter using Provider*. 2020. URL: `https://www.local-droid.com/post/provider-example/` (visitado 28-06-2020).

[26] STOLL, Scott. *Flutter Tutorial: Provider Overview for Humans*. 2020. URL: `https://blog.codemagic.io/flutter-tutorial-provider/` (visitado 28-06-2020).

3.12.4. Patrón Observer El patrón de diseño Observer apunta a definir una relación uno-a-muchos tal que cuando el estado de un objeto cambia, otros son notificados y actualizados automáticamente[27]. Un objeto llamado sujeto mantiene una lista de sus dependientes, denominados observadores, y los notifica automáticamente de cualquier cambio de estado. Es usado principalmente para implementar sistemas de manejo de eventos, en software orientado a eventos.

3.12.5. Patrón Iterator El patrón de diseño Iterator da una interfaz limpia para el acceso elemento por elemento a una colección, independiente de su forma, siendo tal que el iterador accede a los elementos de un objeto de forma secuencial sin exponer su representación subyacente[28].De esta forma el patrón Iterator desacopla los algoritmos de los contenedores.

3.12.6. Patrón Factory Method El patrón Factory Method es un patrón de creación que utiliza métodos de fábrica para crear objetos sin especificar la clase exacta de objeto que se crea. En el patrón Factory Method la lógica de creación de objetos se oculta y el objeto recién creado se devuelve al cliente ya sea especificado en una interfaz e implementado por clases hijas, o implementado en una clase base y opcionalmente sobrescrito por clases derivadas (utilizando una interfaz común)[29]. El patrón Factory Method provee abstracción entre la implementación y clases cliente por medio de la herencia.

---

[27] POYIAS, Andreas. *Design Patterns — A quick guide to Observer pattern.* 2020. URL: `https://medium.com/datadriveninvestor/design-patterns-a-quick-guide-to-observer-pattern-d0622145d6c2` (visitado 28-06-2020).

[28] GIBBONS, Jeremy y DOS SANTOS OLIVEIRA, Bruno César. "The essence of the iterator pattern". En: *Journal of functional programming* 19.3and4 (2009).

[29] DEY, Anik. *Design Patterns — A quick guide to Observer pattern.* 2020. URL: `https://medium.com/@info.anikdey003/factory-method-design-pattern-277dd4bd3a11` (visitado 28-06-2020).

4. ESTADO DEL ARTE

4.1. SEVEN SEVEN

La App de Seven Seven tiene como objetivo que la persona esté donde esté tenga a un solo clic todas las tendencias, y se pueda enterar de lo último que se está llevando en el mercado y ser pionera/o en tener las prendas más espectaculares[30].Dentro de la App se encuentra:

- Acceso directo a todas las categorías de ropa para mujer y hombre, con los productos actualizados. En ellas encontramos: apartado de nueva colección.
- Editoriales, para conocer las mejores tendencias acompañadas de las fotografías de los modelos.
- Sección de rebajas y ofertas.
- Slider con todas las novedades que se puede ver en la pantalla principal de la App.
- Disposición del equipo de atención al cliente.-Resolución de dudas sobre pedidos.
- Elección entre distintos medios de pago.-Comprar por Whatsapp.-Recibir envío GRATIS a partir de un mínimo de compra.
- Notificaciones.
- Promociones y ofertas exclusivas.

Para el día 7 de Junio de 2020 en Google Play cuenta con más de 10.000 descargas, un total de 63 reseñas con 2,5 estrellas de calificación en promedio, en general abundando las reseñas negativas, con muchas de 1 estrella y quejas al respecto de la aplicación y el servicio.

---

[30] PASH S.A.S. *Seven Seven*. Ver. 1.2.0. 12 de nov. de 2018.

4.2. FOREVER 21

La App de Forever 21 tiene como objetivo permitir comprar, encontrar inspiración para un nuevo atuendo, guardar los favoritos y conectarse [31]. Características de la App:

- Tienda: Explorar, navegar, deslizar verticalmente hacia más imágenes, deslizar horizontalmente hacia otro producto y comprar desde cualquier lugar.
- Explorar y comprar: ¿Se necesita otro tamaño? ¿Se requiere más información? Escanear el código de barras de un artículo para obtener detalles, leer comentarios y compartir los hallazgos.
- F21Home: Visualizar la última tienda Shop by Outfit, y los videos de tendencia para una inspiración de estilo instantánea.
- Social: Conectarse con la comunidad de Forever 21 a través de Facebook e Instagram.
- Bandeja de entrada: Actualizaciones sobre exclusivas, ventas y ofertas especiales.
- Encuentra una tienda: Localizador de tiendas para buscar tiendas cercanas, obtener direcciones y averiguar qué marcas están disponibles en cada tienda.

Para el día 13 de Junio de 2020 en Google Play cuenta con más de 5.000.000 de descargas, un total de 33.754 reviews con 4,0 estrellas de calificación media, y en general priman las reviews positivas, con muchas de 5 estrellas.

4.3. PATPRIMO

La App de Patprimo tiene como objetivo ofrecer un amplio portafolio de productos y una calidad inigualable, con las últimas tendencias de moda para hombre y mujer[32]. Dentro de la App se

---

[31] FOREVER 21. *Forever 21.* Ver. 3.4.5.221. 18 de mayo de 2020.

[32] PASH S.A.S. *Patprimo.* Ver. 1.2.0. 9 de mayo de 2020.

encuentra:

- Acceso directo a todas las categorías.
- Última moda en Hombre y Mujer.
- Variedad en tallas, que se adaptan a las necesidades de cada persona.
- Ropa deportiva.
- Línea Infantil.
- Descuentos exclusivos online.
- Todo el catálogo de productos siempre actualizado
- Equipo de atención al cliente a un solo click.
- Registro de todas las órdenes.
- Mapa de todas las tiendas.
- Toda la información de la marca, políticas de tratamiento de Datos.
- Apartado catálogo, en el cual se encuentran las últimas editoriales, con imágenes y videos de las próximas campañas.
- Novedades.
- Notificaciones.

Para el día 13 de Junio de 2020 en Google Play cuenta con más de 10.000 de descargas, un total de 91 reviews con 3,0 estrellas de calificación media, y en general se encuentran reviews a los extremos, muchas de 5 estrellas y muchas de 1 estrella.

4.4. FALABELLA

La App de Falabella tiene como objetivo permitir encontrar toda la información que sobre tiendas Falabella, ofertas, promociones, productos y mucho más.[33] Características de la App:

- Escanear los códigos de barra de productos de tiendas y acceder a información como precio, color, talla, entre otras características.
- Acceder a los catálogos digitales y enterarte de las últimas tendencias y novedades.
- Mapa del módulo físico en tiendas para retirar las compras online.
- Activando el GPS, permite encontrar la tienda Falabella más cercana.
- Conocer las promociones y ofertas vigentes en tiendas Falabella.
- Ofertas CMR.
- Comprar productos de Falabella.com.
- Compartir en las redes sociales, productos, promociones y ofertas.

Para el día 13 de Junio de 2020 en Google Play cuenta con más de 1'000.000 de descargas, un total de 43.540 reviews con 4,3 estrellas de calificación media, y en general priman las reviews positivas, con muchas de 5 estrellas.

[33] FALABELLA RETAIL. *Falabella*. Ver. 1.11.10. 9 de jun. de 2020.

5. METODOLOGÍA

El desarrollo y consolidación del proyecto estuvo basado en el uso de la metodología de desarrollo ágil Scrum. Esta metodología establece el uso de "Sprints"para validar el progreso en cada fase del proyecto a través de historias de usuario.
Las siguientes son las fases para el diseño y desarrollo del prototipo software para el comercio electrónico B2C:

5.1. PLANTEAMIENTO DE LOS REQUERIMIENTOS Y DEL DISEÑO

Durante esta fase se formalizaron las primeras reuniones con el administrador de la compañía, donde se explicaron las limitaciones y objetivos del proyecto y el uso de la tecnología.

5.1.1. Justificación del objetivo del proyecto. Se expone al administrador de la empresa y relacionados el alcance del proyecto con base en los objetivos descritos en el plan de desarrollo. Se da comienzo a la documentación de requerimientos basados en la indicaciones del administrador y la realimentación dada en las primeras reuniones. El administrador de la aplicación tiene como primer objetivo definir unos requisitos mínimos para el software.

5.1.2. Generación de requerimientos Con base en los requisitos descritos por el administrador, se documenta el proyecto de forma no técnica. Se procede a la generación de requerimientos técnicos así como la definición de roles específicos para el proyecto y la aplicación.

5.1.3. Validación de requerimientos Durante esta fase se presentó al administrador una propuesta clara de los requerimientos técnicos basados en las anteriores reuniones.

5.2. PRIMERA FASE DE PROTOTIPADO: PROTOTIPO UNO

En esta primera fase de prototipado, se diseñó una versión mínima de la estructura de datos con sus respectivas entidades para la persistencia de productos de la microempresa, así como también en la aplicación del cliente se construyó la interfaz principal de vista del catálogo de productos y en la app Moi Admin las vistas principales para el CRUD de las entidades necesarias para la creación del catálogo de productos y el cargue de sus existencias.

5.2.1. Configuración del ambiente de desarrollo y base de datos Durante esta fase fue posible instalar el SDK y framework de Flutter, así como la creación del proyecto en el servicio de Firebase para poder conectar nuestros datos de la aplicación. Se da inicio a la programación.

5.2.2. Catálogo de productos Esta fase permitió la creación de productos de prueba para validar la conexión y correcto uso de los datos desde la aplicación móvil de los clientes. Se validó la conexión y persistencia de datos entre la base de datos y la aplicación móvil. Se desarrolla la vista inicial del cliente, detalles del producto y un menú lateral para el usuario.

5.2.3. CRUD para entidades relacionadas al catálogo de productos Esta fase consistió del desarrollo de todas las vistas en la app Moi Admin para garantizar el funcionamiento de las operaciones CRUD para las entidades relacionadas con los productos: líneas, grupos, colecciones, marcas, productos, tallas, colores, imágenes de producto y color y existencias.

5.3. SEGUNDA FASE DE PROTOTIPADO: PROTOTIPO DOS

A lo largo de la segunda fase de prototipado, se desarrolló la funcionalidad de "canastilla de compras" para la aplicación del cliente, la cual permitió elegir productos y contener dicha selección de productos para realizar una compra, controlando las cantidades, precios y el total de la compra. en la app Moi Admin se desarrollaron las vistas funcinales de CRUD de ciudades y puntos de venta, y la vista para visualización y edición de la numeración. También se creó el

diagrama UML de entidad-relación de la base de datos.

5.3.1. Creación de diagrama UML entidad-relación de la base de datos Este proceso permitió la estructuración de la base de datos a través de un diagrama entidad-relación con el fin de representar los objetos de la realidad de forma adecuada manteniendo una arquitectura general coherente con las reglas del negocio. Esta estructura se utilizó para organizar tareas en el desarrollo de las vistas de forma eficaz.

5.3.2. Creación de la vista de canastilla de compras Este proceso permitió la creación de una vista para mostrar los productos que el usuario ha seleccionado para posteriormente realizar una compra. Se realiza la conexión con la tabla de la base de la datos para realizar la tarea.

5.3.3. Conexión entre detalles del producto y canastilla de compras A continuación se procede a realizar el envío de productos desde la vista de detalles de productos hacia la canastilla, utilizando así los métodos para el consumo y creación de productos en la tabla de canastilla de compras.

5.3.4. Validación de cantidades disponibles en la vista de detalles del producto Es necesario validar que existan productos suficientes para ser agregados a la canastilla de compras. Esta eventualidad puede suceder si el usuario ha guardado un producto en la canastilla de compras con antelación y decide tiempo después agregar el mismo producto pero no hay suficientes existencias.

5.3.5. Validación de cantidades disponibles en la vista de canastilla de compras Fue necesario realizar una nueva validación en la vista de canastilla de compras momentos antes de proceder a la pasarela de pago. Se prevé la eventualidad de guardar productos con antelación en la canastilla de compras, acabar las existencias e intentar comprar productos que ya no

existen en la tienda.

5.4. TERCERA FASE DE PROTOTIPADO: PROTOTIPO TRES

Durante esta tercera fase de prototipado, se desarrolló la funcionalidad de generación de pedidos para la app cliente, donde cada pedido es generado a partir del contenido de la “canastilla de compras". En la aplicación Moi Admin se desarrollaron las vistas para CRUD de usuarios y descuentos.

5.4.1. Estudio de eventualidades En este proceso fue necesario considerar el evento en que un cliente desea recoger su pedido en la tienda así como recibirlo en una dirección a domicilio.

5.4.2. Desarrollo de vistas generación y visualización de pedidos app cliente Creación de la vista para el envío a domicilio, creación de pedido para recoger en la tienda y una vista para visualizar las órdenes realizadas por el usuario a lo largo del tiempo.

5.4.3. Desarrollo de vistas para el CRUD de usuarios y descuentos Moi Admin Se desarrollaron las vistas siguiendo la estructura utilizada en las anteriores vistas de CRUD desarrolladas para la entidad Usuario de la base de datos, además de la vista que implementa la funcionalidad de aplicar descuento por línea, grupo, colección o marca.

5.4.4. Creación de pedidos Por último, se procede a crear el pedido en la base de datos a partir de los datos ingresados por el cliente, permitiendo a futuro al administrador manejar el pedido.

5.5. CUARTA FASE DE PROTOTIPADO: PROTOTIPO CUATRO

El cuarto prototipo contiene la gestión y facturación de los pedidos desde la app Moi Admin, de tal forma que cuenta con las vistas para ver los pedidos, sus detalles y entregarlos para los

usuarios de rol repartidor, y ver los pedidos, sus detalles y cancelarlos para los usuarios de rol administrador.

5.5.1. Gestión de pedidos por el administrador Las acciones definidas para el administrador son ver los pedidos, ver los detalles del pedido (items), en caso de que sea un pedido entregado o incompleto reimprimir la factura, y cancelarlos.

5.5.2. Gestión de pedidos por el repartidor Las acciones definidas para el repartidor son ver los pedidos, ver los detalles del pedido (items), seleccionar la cantidad a entregar de cada item y entregar el pedido, que tomará estado incompleto en el caso que de la cantidad entregada sea menor a la cantidad pedida.

5.5.3. Facturación La factura se genera en formato pdf en tamaño de papel 8 milímetros con una disposición de los elementos basada en las facturas de otras soluciones software, y se guarda en el almacenamiento interno del dispositivo, para posteriormente mostrarla en un renderizador nativo de pdf en una vista de la app Moi Admin. No sigue los lineamientos de la DIAN, por lo cuál no es es una facturación electrónica legal.

5.6. QUINTA FASE DE PROTOTIPADO: PROTOTIPO

Durante esta quinta fase de prototipado se implementaron funcionalidades extra en la solución software como componentes de estadísticas y seguridad para el proyecto, que consiste respectivamente en la generación y consulta de históricos de Pedidos y Clientes, y la definición y generación de logs de seguridad sobre operaciones sensibles en el sistema.

5.7. SEXTA FASE DE PROTOTIPADO: PROTOTIPO SEIS

Durante esta sexta fase de prototipado se implementó en la app Moi Admin la funcionalidad de importación de archivos en formato csv y exportación en archivos en formato csv de las

entidades de la base de datos pertinentes, definiendo sus respectivos formatos y cabeceras, para así permitir al usuario administrador realizar el cargue inicial del sistema, el cargue de existencias para movimiento de inventarios, entre otras operaciones que implican gran cantidad de transacciones en la base de datos.

5.8. SÉPTIMA FASE DE PROTOTIPADO: PROTOTIPO SIETE

Durante esta séptima fase de prototipado se implementó la funcionalidad de registro, inicio de sesión y manejo de sesiones, construyendo las vistas con los formularios respectivos y haciendo uso de la API de Firebase Authentication, junto a los registros de cada usuario y cliente en la base de datos.

5.9. OCTAVA FASE DE PROTOTIPADO: PROTOTIPO OCHO

Durante esta octava fase de prototipado se desarrolló el último prototipo del proyecto, en el cuál se añadieron los menús de nagevación y se configuró y adecuó la navegación para permitir una experiencia de usuario fluida. Este prototipo fue además el resultado de la corrección de errores, adición de funcionalidades útiles, así como ajustes finales para el despliegue.

5.9.1. Menú Se construye un menú de tipo drawer para cada rol en cada aplicación (un menú para cliente, un menú para repartidor, un menú para administrador), conteniendo este las opciones de navegación a cada vista a la que dicho usuario o cliente tiene acceso, procurando incorporar ergonomía cognitiva y teniendo una estructuración que facilite el aprendizaje del uso de la herramienta y la navegabilidad en las tareas cotidianas que se realicen con ella.

5.9.2. Módulo para lista de deseos Se ve la necesidad de permitir al cliente tener una lista de deseos diferente a la canastilla de compras, permitiéndole guardar productos persisténtemente.

5.9.3. Extras Se adiciona una vista a la aplicación de cliente de preguntas, quejas, reclamos y felicitaciones.

5.9.4. Verificación de funcionamiento íntegro Se verifica la correcta comunicación entre la aplicación móvil del cliente y del administrador teniendo en cuenta diferentes eventualidades en el momento de crear y gestionar pedidos, productos e información.

6. DESARROLLO DEL PROYECTO

6.1. DESCRIPCIÓN GENERAL

De acuerdo con los requerimientos obtenidos, se planteó una arquitectura para la solución de software que consta de dos aplicaciones móviles desarrolladas con el framework de Google para desarrollo móvil Flutter y una base de datos de Firebase de tipo Realtime Database.

Primeramente se tenía planteada una aplicación para el cliente, que permitiese ver el catálogo de productos, añadir a la canastilla de compras y generar su pedido. La decisión de realizar dos aplicaciones fue motivada por la necesidad de permitir al administrador establecer las condiciones iniciales del sistema, el cargue del catálogo y la información requerida por parte del negocio para poner en marcha la tienda virtual, con base en lo cual se planteo además el desarrollo de una aplicación móvil para el administrador.

La comunicación entre la base de datos de Firebase Realtime Database y ambas aplicaciones se realiza por medio de la API REST de Realtime Database, pero la existencia de un plugin de flutter para hacer uso de la API REST de Realtime Database[34] llevó al planteamiento de la siguiente pregunta de investigación: ¿Es mejor utilizar directamente la API REST de Realtime Database, o utilizar el plugin de flutter que hace uso de dicha API?. Con base en esta pregunta se tomó una decisión orientada al componente investigativo del proyecto, que es la siguiente: Se utilizó la API REST de firebase Realtime Database directamente en la aplicación del administrador, y se utilizó el plugin firebase_database para la aplicación del cliente.

De esta forma, en la aplicación para el cliente se utilizaron requests http definidas en la API REST, y en la aplicación para el administrador se utilizaron los métodos definidos en el plugin para interactuar con la base de datos.

Se utilizó la plataforma de Google Cloud Functions para la verificación de permisos por tipo

---

[34] THOMPSON, Arthur. *firebase_ database*. Ver. 3.1.6. 25 de mayo de 2020.

de usuario e interacción con Firebase Authentication, para la obtención del email del usuario a través del uid generado y eliminación de usuarios.

6.1.1. Roles Con base en los requerimientos de la solución planteada se identificaron ciertos perfiles, y con base en estos se definieron los siguientes roles de usuario dentro del sistema:

1. Cliente: El cliente es el consumidor del público general que realiza la adquisición de prendas a la empresa Moi Colombia, que hará uso de la aplicación de Cliente en la cuál podrá principalmente ver el catálogo disponible, agregar productos a la canastilla de compras y realizar pedidos.

2. Administrador: El administrador es la persona asignada por parde de Moi Colombia, que se encargará del cargue, configuración y control de la solución, administrando la estructuración del catálogo, los productos, las existencias, los pedidos, los descuentos, y toda la información que permite el correcto funcionamiento del sistema.

3. Repartidor: El repartidor es la persona asignada por Moi Colombia para realizar la entrega de los pedidos.

6.1.2. Arquitectura e implementación de los patrones de diseño Cuando el usuario interactúa con la UI (interfaz de usuario) y dicha interacción involucra una petición a la base de datos, esta petición se realiza utilizando uno de los métodos definidos en el BLoC designado para esa tarea con la funcionalidad relacionada a dicha estructura de datos, implementando el patrón de diseño BLoC, de tal forma que su funcionalidad está siendo llamada desde dicha vista, dicho método conteniendo una ejecución el método correspondiente en el Provider e introduciendo estos datos en el Stream establecido, que siendo alimentado a través de Behaviour Subject notifica a todos los Widgets observadores del cambio de su estado, permitiendo a estos consumir estos datos, y reconstruir su estado, implementando el patrón de diseño Observer, y mientras que estos datos son observados, se itera sobre ellos independientemente de la estructura interna de dicha colección de datos, implementando el patrón de diseño Iterator, además de realizar

operaciones como la transformación y combinación, siguiendo el paradigma de Programación Funcional. De esta forma se utilizó una Arquitectura Reactive para integrar la programación reactiva y la programación asíncrona en la arquitectura Software definida para la solución.
Los datos son traídos de la base de datos por medio de los Providers en los cuáles se encuentran centralizadas las peticiones de tipo http haciendo uso de la REST API de Firebase Realtime Database, complementando la implementación del patrón BLoC. Estos datos se encuentran en tiempo de ejecución en modelos de objetos definidos como clases, cada cuál con un método factory que implementa el patrón Factory Method, especializado en la construcción de dicho objeto a partir de un mapa llave-valor en formato json, además de un método para serializar dicho objeto como un mapa json. Cuando los datos son recibidos en formato json de la base de datos, se utiliza el método para construir el objeto en base a este mapa, y en el sentido contrario cuando se realiza envío de información a la base de datos se serializa el objeto como un mapa json antes de enviar sus datos. En cada uno de los BLoCs se encuentra instanciado su Provider correspondiente, de forma privada, y este objeto es inaccesible directamente, mientras que sus métodos son ejecutados en los métodos propios del BLoC, respetando el principio de encapsulamiento.
Los BLoCs se encuentran instanciados únicamente en un Widget denominado InheritedProvider, que extiende al Widget InheritedWidget, y contiene un constructor de tipo factory cuyo funcionamiento está basado en el patrón de diseño Factory Method, retornando una nueva instancia del mismo únicamente si no se ha inicializado antes, o en caso contrario retornando la instancia ya existente, derivando así en la implementación del patrón de diseño Singleton. Este InheritedProvider envuelve a un Widget MultiProvider que contiene un mixin de ChangeNotifier, un tipo de observable, encargado del tema elegido para la aplicación (tema oscuro o claro), y este envuelve al Widget MaterialApp, de tal forma que el InheritedProvider está al mayor nivel posible como ancestro de todo el arbol de Widgets que constituye la aplicación, exponiendo los BLoCs a dicho arbol de Widgets, siendo el Widget que articula la implementación del patrón de diseño Provider.

La arquitectura implementando el patrón BLoC se puede apreciar en la Figura 2.
La arquitectura del árbol de Widgets junto con su InheritedProvider se puede apreciar en la Figura 3.

6.1.3. Preferencias y datos de usuario Para ambas aplicaciones se utilizó el plugin shared_preferences[35] que envuelve NSUserDefaults (en iOS) y SharedPreferences (en Android), para guardar asíncronamente de forma persistente en disco preferencias simples del usuario y datos del usuario actual logueado, tales como en la aplicación de administrador el tema actual elegido, el uid, el tipo de usuario, el email, los nombres, apellidos y la url de su imagen de perfil.

6.1.4. Sesión y gestión de usuarios Para el registro, inicio de sesión y gestión de usuarios se eligió Firebase Authentication [17]. Se utilizó el plugin firebase_auth[36] para usar la API de Firebase Authentication. El registro de los usuarios y clientes consta procedimentalmente de dos momentos:

1. Registo en Firebase Authenticacion: Con el correo electrónico y contraseña se utiliza la API de Firebase Authentication para registrar al cliente/usuario, y el servicio le asigna un uid (user id) específico y único a dicho usuario.

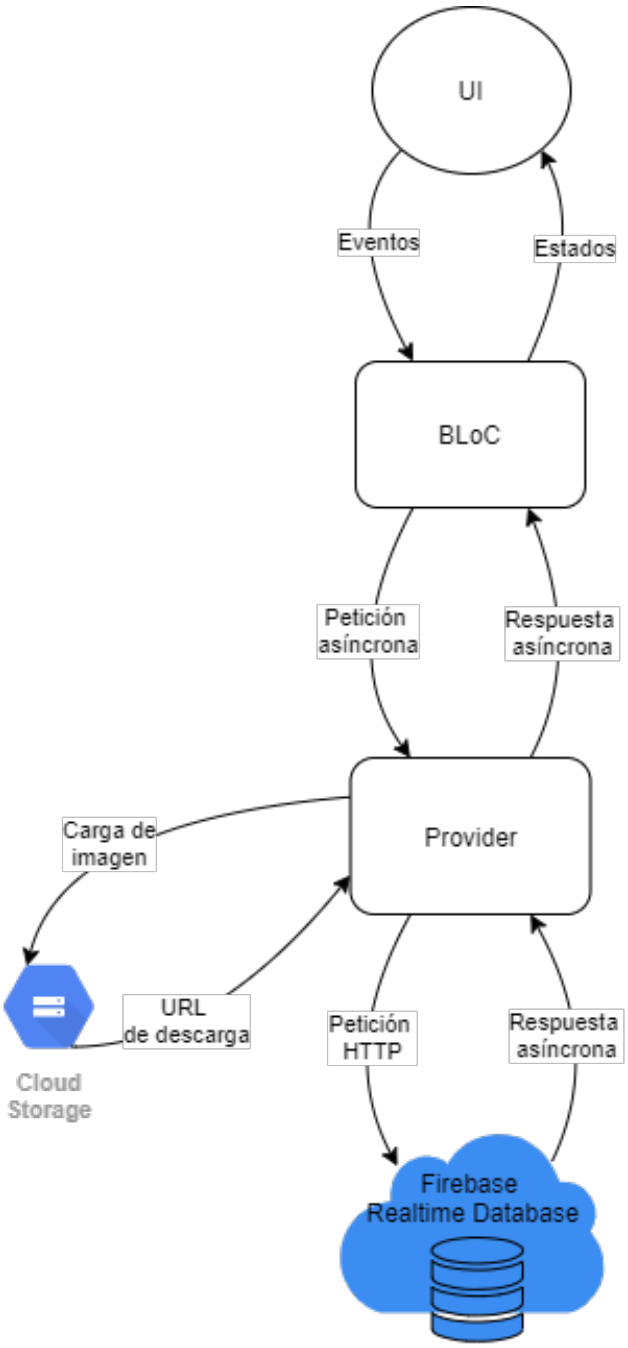
2. Registro de los datos del cliente/usuario en la base de datos: Se toma ese uid generado por Firebase Authentication, y se crea un registro en el nodo de Clientes o el de Usuarios en la base de datos, dependiendo de si se está registrando desde la aplicación de clientes o la aplicación de administración, y se guarda con el uid como la clave del registro, siendo el valor la lista de todos los atributos extra a guardar relativos al cliente/usuario que está ingresando al sistema, tales como los nombres, apellidos, documento de identidad, entre otros.

---

[35] FLUTTER TEAM. *shared_preferences*. Ver. 0.5.7+3. 21 de mayo de 2020.

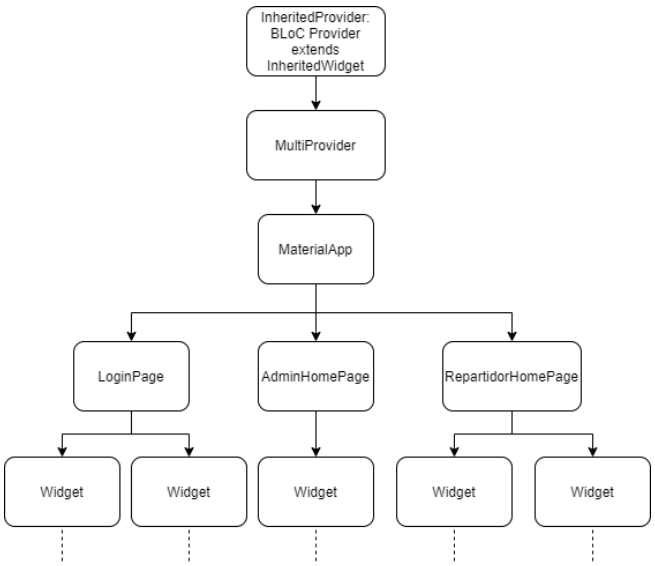
[36] THOMPSON, Arthur. *firebase_auth*. Ver. 0.16.1. 25 de mayo de 2020.

Figura 2. Diagrama de Arquitectura - Implementación del patrón BLoC



De esta forma, se tienen relacionadas las cuentas de usuario de Firebase authentication con las cuáles se autentica el usuario y se generan los tokens para las consultas a la base de datos,

Figura 3. Diagrama de Arquitectura - Estructura Árbol de Widgets



con la información extra guardada en la base de datos por medio del uid. También, el email se guarda en la base de datos para agilizar las consultas, pero la clave de la cuenta de usuario no es guardada en la base de datos, está únicamente tercerizada en este servicio, de tal manera que la api es utilizada para realizar los procedimientos de login, cambio de clave y recordar clave al correo electrónico.

6.1.5. Peticiones HTTP Para la aplicación de administrador se realizaron las peticiones http de tipo post, put, patch, delete y get utilizando el plugin http de dart[37], autenticadas con el token obtenido de Firebase Authentication previamente mencionado.

6.1.6. Selección y captura de imágenes Para la selección de imágenes de la galería o su captura desde la cámara se utilizó el plugin image_picker[38], con el cuál permite tener estos archivos en una variable de tipo File temporal, que luego se guarda en el almacenamiento

---

[37] DART TEAM. *http*. Ver. 0.12.1. 27 de abr. de 2020.

[38] FLUTTER TEAM. *image_picker*. Ver. 0.6.7+2. 15 de ene. de 2020.

persistente cuando se termina la transacción de introducir información por medio de formularios con el seleccionador de imágenes.

6.1.7. Almacenamiento de imágenes El almacenamiento de las imágenes se estableció en Firebase Cloud Storage[39], utilizando el plugin firebase_storage[18] para cargar y descargar los archivos generados por los usuarios del almacenamiento persistente en la nube, en carpetas cuyos nombres se encuentran definidas por los objetos de la realidad que representan, tales como Marcas, Productos, ProductosColores, PuntosVenta y Usuarios. El nombre de los archivos viene del código interno que se asigna a estos objetos en la base de datos, junto con la extensión de archivo correspondiente, tales como jpg y png.

6.1.8. Fechas y horas La cantidad de estructuras y maneras específicas de almacenar e interactuar con datos de tiempo ligadas al lenguaje de programación o herramientas utilizadas sugería la necesidad de buscar una forma estándar de almacenar y guardar todo lo relacionado con fechas y hora. Para esto, se utilizó el sistema de Época Unix[9], debido a que el tiempo Unix permite guardar una fecha y hora como un número entero, siendo tal que se puede guardar en cualquier base de datos que soporte números enteros, y luego al realizar la consulta a la base de datos, construir y poner en una variable temporal el objeto que representa ya sea fecha u hora de acuerdo al lenguaje de programación o herramienta que se esté utilizando en la aplicación software. En este caso, al utilizar firebase Realtime Database se guardaron los datos de fecha y hora como un atributo entero en su respectiva entidad, y al realizar las consultas a la base de datos se utiliza el objeto de tipo DateTime[40] para guardar estas fechas en tiempo de ejecución utilizando el constructor DateTime.fromMilisecondsSinceEpoch() y pasándole como argumento

---

[39] GOOGLE LLC. *Firebase Cloud Storage*. 2020. URL: `https://firebase.google.com/docs/storage` (visitado 25-06-2020).

[40] GOOGLE LLC. *Flutter DateTime class*. 2020. URL: `https://api.flutter.dev/flutter/dart-core/DateTime-class.html` (visitado 26-06-2020).

el número de milisegundos guardados en la base de datos. asimismo, se pasa el objeto DateTime a milisegundos desde la época para guardar en la base de datos.

6.1.9. Responsive Design Con el objetivo de aplicar los criterios de diseño adaptable al dispositivo, se utilizó al establecer en código el tamaño de la mayoría de los widgets visibles por el usuario que no tuvieran responsive design en su naturaleza, el widget MediaQuery[41] para obtener las dimensiones de la pantalla desde la cuál se está utilizando la aplicación, y de acuerdo a ese tamaño variable establecer las dimensiones como porcentajes de la misma, haciendo el tamaño de los widgets proporcional al tamaño de la pantalla. Además, para algunos widgets se utilizó el widget Expanded[42] para envolverlos, lo cuál hace que se expandan tomando el tamaño máximo posible en su contenedor sin obstaculizar o cubrir otros widgets, realizando de esta forma una disposición más adecuada de los elementos en pantalla. Debido a la naturaleza del texto y la dificultad lograr una disposición adecuada en pantalla y de acuerdo a los límites definidos, se tomó la decisión de utilizar el plugin auto_size_text[43] que permite que el texto se redimensione para ajustarse a los límites definidos.

6.1.10. Importación y exportación masiva de datos Debido a que hay operaciones que contienen un gran volumen de datos, tales como el cargue inicial, llegada de una nueva colección, nuevos productos, ingreso o egreso de existencias, entre otras, se tomó la decisión de implementar un mecanismo de importación masiva de datos. Asimismo, debido a la necesidad de realizar backups y tener copias de los datos que se puedan trasladar a otras localizaciones físicas, se decidió implementar un mecanismo de exportación masiva de datos. Para la inte-

---

[41] FLUTTER TEAM. *Flutter Widget MediaQuery*. 2020. URL: `https://api.flutter.dev/flutter/widgets/MediaQuery-class.html` (visitado 25-06-2020).

[42] FLUTTER TEAM. *Flutter Widget Expanded*. 2020. URL: `https://api.flutter.dev/flutter/widgets/Expanded-class.html` (visitado 25-06-2020).

[43] LEIER, Simon. *auto_size_text*. Ver. 0.6.7+2. 13 de ago. de 2019.

gración de estas funcionalidades se estableció el uso del formato estándar csv[8], ampliamente utilizado en el mundo empresarial y el desarrollo de software para empresas de comercio, y especialmente empresas de comercio electrónico. Se definieron plantillas para importación y para exportación, con los nombres de las columnas, denominados headers o cabeceras, en los cuáles el usuario puede introducir los datos desde su editor de texto de preferencia, y luego desde una vista para importación puede importarlos a la base de datos tan sólo eligiendo el archivo del almacenamiento de su dispositivo interno de su dispositivo, o elegir qué tabla desea exportar para que sea guardada en el almacenamiento interno de su dispositivo, en una ruta definida específicamente para este propósito. La importación de datos comprende las marcas, colecciones, tallas, colores, líneas, grupos, productos y existencias, mientras que la exportación de los datos comprende además de las anteriormente mencionadas, los pedidos. La exportación de pedidos tiene una particularidad, y es que permite al usuario definir un intervalo, seleccionando una fecha inicial y una fecha final, además de si quiere filtrar los pedidos exportados de acuerdo al estado y/o al tipo de entrega. En la parte técnica se utilizó el plugin csv[44], que permite la conversión de un String de texto en formato csv a estructuras de tipo List de dart (utilizado en el proceso de importación), y al contrario, la conversión de listas de dart a un String en formato csv (utilizado en el proceso de exportación). Para cada una de las entidades que se importan y exportan, se definió la lógica de importación y exportación, incluyendo las validaciones de formato de los datos ingresados, existencia de las precondiciones para el cargue de datos (tales como para productos existencia de las línea a la cuál pertenece el producto), y las inserciones y actualizaciones necesarias para la integridad de la base de datos.

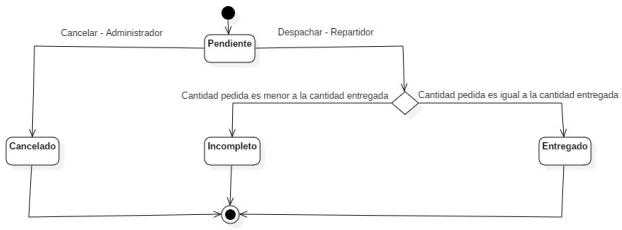
6.1.11. Gestión y estado del pedido Cuando un cliente realiza un pedido exitosamente, este pedido se inicializa en estado pendiente y tiene la información o cabecera del pedido, la información de entrega o despacho, y sus items definidos cada cual con el producto, talla, color

---

[44] LOITSCH, Christian. *csv*. Ver. 4.0.3. 5 de abr. de 2019.

y cantidad pedida. Las acciones que cambian el estado se dan por medio de los botones desde la UI: para cambiar al estado cancelado un usuario con rol Administrador debe deslizar la tarjeta de pedido en la vista de "Pedidos", y para cambiar el estado a entregado o incompleto un usuario con rol Repartidor debe tocar el botón de despachar, y a nivel de código se verifica que la cantidad entregada sea igual a la pedida, si es así el estado cambia a entregado, en caso contrario cambia su estado a incompleto. Para apreciar la dinámica del estado del pedido ver la Figura 4.

Figura 4. Diagrama UML de estados - Estados del pedido



6.1.12. Facturación En el momento en el que un pedido es procesado, lo cuál significa que el repartidor selecciona las cantidades a entregar y acciona el botón de entregar pedido, el último paso es realizar en un archivo con formato pdf la factura, y guardarla en el almacenamiento interno el dispositivo, en una ruta definida específicamente para este propósito. Para la construcción de esta factura se utilizó el plugin pdf[45], tiene un sistema de Widgets muy similar al de Flutter, para la creación del pdf a alto nivel, y una librería de creación de pdf a bajo nivel que se ocupa de la generación de los bits pdf. De esta forma, se utilizó una estructura y tipo de programación muy similar a la utilizada en Flutter con estos Widgets específicos para organizar una factura que, aunque no sigue los lineamientos específicos y los requerimientos requeridos por la DIAN para ser considerada factura electrónica legal, es un buen soporte para entregar al cliente y contar con la documentación del procesamiento de los pedidos, para cualquier reclamo

[45] NFET.NET. *pdf*. Ver. 1.6.0. 28 de mar. de 2020.

o situación que pueda surgir, tanto de parte del cliente como de parte de la empresa. Tiene un formato de 80 milimetros, cuyo uso está ampliamente extendido en el ámbito empresarial para la impresión en POS, y el tamaño de los contenedores, la tabla de items y el texto está definida de acuerdo a un porcentaje del tamaño del formato. Los valores monetarios tienen un formato de tipo pesos colombianos, haciendo uso de la librería intl[46]. Esta factura cuenta con la siguiente información:

- Nombre de la empresa.
- NIT de la empresa.
- Número de factura de venta.
- Fecha y hora de facturación.
- Documento de identidad o NIT del cliente.
- Nombre y apellido del cliente.
- Información de cada artículo del pedido, de la siguiente forma:
    1. Nombre del artículo.
    2. Precio unitario.
    3. Cantidad entregada.
    4. valor total del artículo
- Subtotal de la compra.
- Valor del IVA.
- Total a pagar de la compra.

---

[46] DART TEAM. *intl*. Ver. 1.6.0. 6 de ene. de 2020.

- Observaciones, texto opcional para mostrar en toda factura a definir por la empresa.
- Resolución, texto de la resolución autorizando la facturación para la empresa por la DIAN.

También, en caso de ser una reimpresión de la factura, cuenta con una marca de agua que la identifica como una copia de la original.

6.1.13. Visualización de la factura Para la visualización de la factura se utilizó el plugin native_pdf_view[47], que permite renderizar pdf y mostrar un archivo pdf en Web, MacOs 10.11+, Android 5.0+, iOS. De esta forma, a continuación de la generación de la factura y de que se haya guardado en el almacenamiento interno del dispositivo, se crea la vista del pdf con la ruta del archivo recién creado, que lo muestra dentro de la aplicación en su propio visualizador nativo, mientras que el dispositivo desde el cuál se esté visualizando el pdf tenga soporte para el mismo. Asimismo, desde la vista de pedidos cuando el administrador reimprima la factura, también permitirá la visualización del archivo creado.

6.1.14. Formularios Para la entrada de datos se construyeron formularios que tuvieran tamaños y organización ergonómicos para el usuario, además del flujo del formulario como tal, se implementaron las acciones de cada campo, las cuáles permiten al usuario moverse fácilmente entre campos, con las acciones de siguiente y terminado, además de scrolls o desplazamientos de pantalla para situar el campo en edición en la parte superior de la pantalla, evitando que el teclado emergente lo obstruya.

6.1.15. Filtrado y búsqueda En las vistas para visualización e interacción con las entidades tales como Líneas, Grupos, Productos, entre otros, se establecieron criterios e implementaron cajas de búsqueda para que el usuario pueda filtrar los resultados por atributos de las entidades, así como borrar rápidamente los filtros y ocultar dicha caja de búsqueda para encontrar

---

[47] SHKURKO, Serge. *native_pdf _view*. Ver. 3.6.2. 31 de mayo de 2020.

eficazmente los datos deseados.

6.1.16. Flujo y redirección para creación y filtrado de productos Para la creación de productos se implementó la redirección con solamente un toque de acuerdo al flujo normal de uso de la aplicación. Dado que las prendas se crearán dentro de un grupo, y dicho grupo estará inscrito en una línea, el flujo convencional de la creación de una línea, grupo y/o prenda se facilita de la siguiente forma:

1. El usuario ingresa a la vista de Líneas, en la cuál puede ver las líneas creadas, editar alguna o crear una nueva línea. Mantener presionada esta línea redirecciona a la vista de Grupos filtrando por dicha línea.

2. El usuario se encuentra en la vista de Grupos, viendo los grupos de la línea escogida, con la posibilidad de editarlos y crear un nuevo grupo, que quedará inmediatamente dentro de la línea elegida. El usuario puede entonces con dar toque simple a uno de estos grupos, y es redireccionado a la vista de Productos filtrando por dicho grupo.

3. El usuario se encuentra en la vista de Productos, viendo los productos del grupo escogido, con la posibilidad de editarlos y crear un nuevo producto, que quedará inmediatamente dentro del grupo elegido.

Además, en todas las abstracciones de categorías de productos, las cuáles son Líneas, Grupos, Colecciones y Marcas, al dar un toque simple a cualquiera de las cartas en su vista respectiva, redirecciona a la vista de Productos filtrando por ella.

6.1.17. Acciones del teclado personalizadas Debido a que en iOS se presentaba el problema de que no existía un botón que realizara una acción de acuerdo al campo de entrada que hubiese desencadenado su apertura, se optó por utilizar el plugin keyboard_actions[48] que

[48] VELASQUEZ LOPEZ, Diego. *keyboard_ actions*. Ver. 3.2.1+1. 20 de mayo de 2020.

permite añadir características personalizadas a los teclados de Android y iOS de forma simple. De esta forma, se implementaron los botones para seguir al siguiente campo y para finalizar y cerrar el teclado para los campos dentro de los formularios, para mejorar la experiencia de usuario y facilitar la entrada de datos.

6.1.18. Históricos Los análisis de desempeño del comportamiento de los pedidos y clientes se volvieron un tema prioritario en la evaluación de la transferencia tecnológica y la revisión de la efectividad del modelo de negocios, por lo cuál se implementó un mecanismo de históricos para guardar los registros de Pedidos por estado y Clientes en cada lapso determinado. En vez de calcular los históricos cada vez que se quiera realizar una consulta de las estadísticas del sistema, se guarda un registro que se actualiza con cada transacción que involucre un cambio de estado del pedido o un registro del cliente. Cada histórico tiene una base general que es el año y el mes al que corresponde, siendo tal que si el mes y el año son cero es el histórico general desde el momento en que empezó a usarse la solución, si el mes es cero con un año determinado es el histórico del año entero, y si el año y el mes están determinados, es el registro de ese mes en ese año específico, y puede ser de uno de los siguientes dos tipos:

1. Clientes: Guarda para ese intervalo el número de clientes existentes, cantidad de clientes que se han registrado y cantidad que han realizado pedidos.

2. Pedidos: Guarda para ese intervalo, para cada uno de los estados del pedido, la cantidad de pedidos, la cantidad de prendas asociadas a dichos pedidos, y la suma del valor de esos pedidos.

6.1.19. Logs Se definieron 8 operaciones sensibles:

1. Creación de productos.

2. Eliminación de productos.

3. Modificación de precio de lista.

4. Modificación de porcentaje de descuento.

5. Cambio de estado de pedidos.

6. Creación de usuarios.

7. Modificación de la información de usuarios.

8. Cambio de clave de usuarios.

9. Eliminación de usuarios.

Para cada una de estas operaciones se creó un modelo de log, con el usuario que ejecutó la operación y la información relevante del cambio realizado en dicha transacción. De esta forma en cada operación se crea el log correspondiente y se guarda en la base de datos.

6.2. BASE DE DATOS

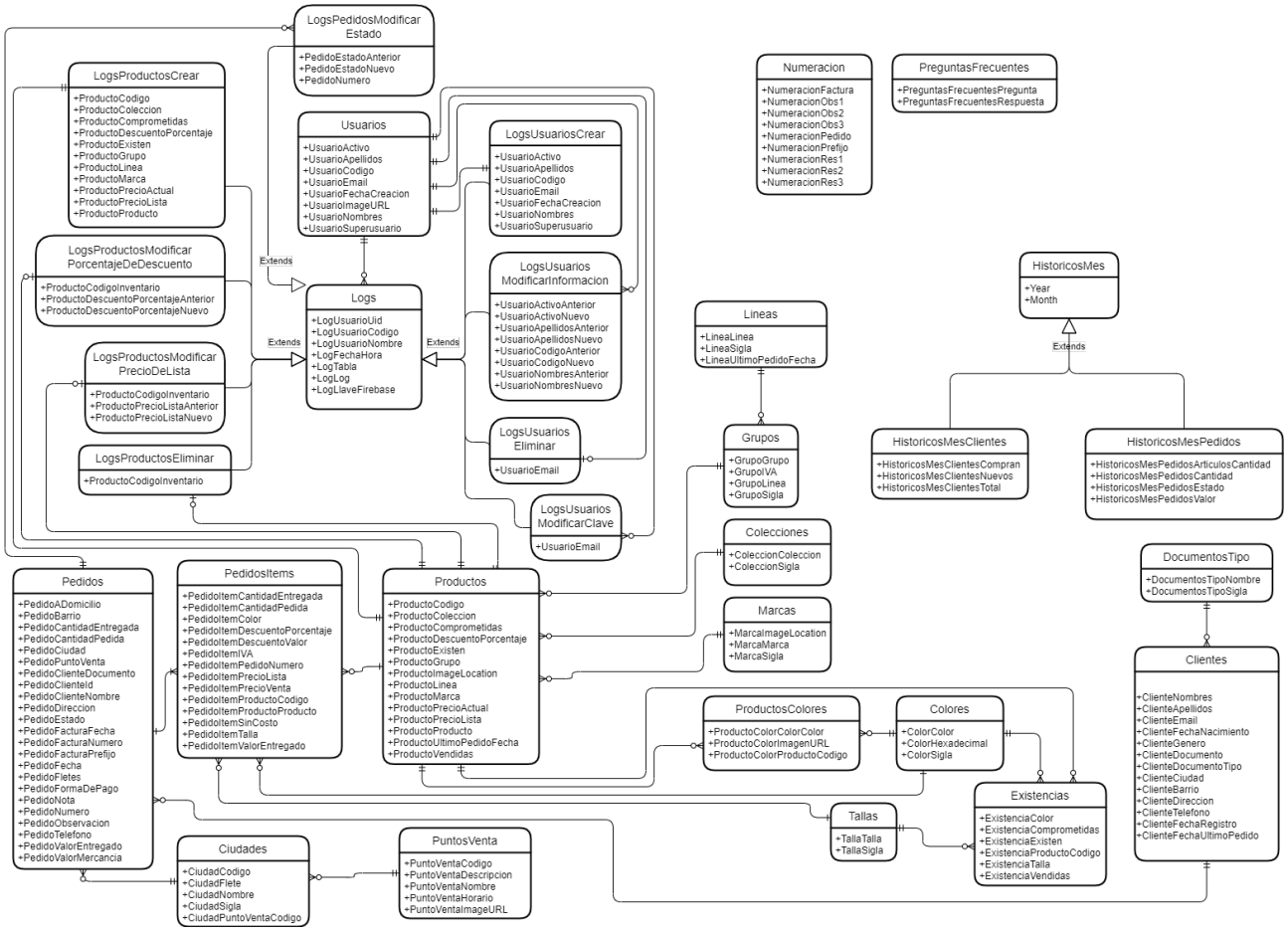
Al utilizar la base de datos de Firebase de tipo Realtime Database no existen tablas ni registros como tal, sino objetos JSON, y la base de datos puede conceptualizarse como un árbol JSON alojado en la nube. Al agregar datos al árbol JSON estos se convierten en un nodo de la estructura JSON existente con una clave asociada[49]. La estructura de la base de datos se puede apreciar en la Figura 5.
Siendo así, se definieron los siguientes nodos raíz, que serían conceptualmente equiparables a las tablas de una base de datos relacional tradicional, con sus correspondientes registros, cada cuál con atributos específicos:

1. Ciudades: Contiene registros de las ciudades en las cuáles está definido el valor del flete, y sirve para añadir los costos de envío cuando el pedido se entregue a domicilio. Cada nodo tiene los siguientes atributos:

---

[49] GOOGLE LLC. *Estructura tu base de datos Realtime Database*. 2020. URL: `https://firebase.google.com/docs/database/web/structure-data` (visitado 15-06-2020).

Figura 5. Diagrama Entidad-Relación Base de Datos



- CiudadFlete: Valor del flete para la ciudad.
- CiudadNombre: Nombre de la ciudad.
- CiudadSigla: Identificador corto de la ciudad.

2. Clientes: Esta colección contiene los datos requeridos para los usuarios de la aplicación móvil del cliente. Trabaja de la manejo del servicio de Firebase Authentication para permitir la autenticación de los usuarios de la app y llevar un control de los datos de usuario y movimientos. Cada nodo contiene los siguientes atributos:

- ClienteNombres: Nombres del cliente. Variable utilizada para los envíos.

- ClienteApellidos: Apellidos del cliente. Variable utilizada para los envíos.
- ClienteEmail: Correo electrónico del cliente. Variable utilizada para la gestión de autenticación del cliente y mostrar al usuario sus datos dentro de la sesión.
- ClienteFechaNacimiento: Fecha de nacimiento del cliente. Información no utilizada actualmente. Se recomienda su uso hacia el futuro para campañas personalizadas y manejo de datos.
- ClienteGenero: Género del cliente. Información no utilizada actualmente. Se recomienda su uso hacia el futuro para campañas personalizadas y manejo de datos.
- ClienteDocumento: Número de documento de identidad del cliente. Dato utilizado para los envíos y generación de factura no oficial.
- ClienteDocumentoTipo: Tipo de documento de identidad.
- ClienteCiudad: Ciudad donde reside el cliente. Dato utilizado para los envíos a domicilio.
- ClienteBarrio: Barrio donde reside el cliente. Dato utilizado para los envíos a domicilio.
- ClienteDireccion: Dirección donde reside el cliente. Dato utilizado para los envíos a domicilio.
- ClienteTelefono: Teléfono del cliente. Dato utilizado para los envíos a domicilio. Con este dato la empresa tiene la oportunidad de contactar al cliente.
- ClienteFechaRegistro: Fecha en la que el cliente se registró en la plataforma. Dato utilizado para futuras estadísticas.
- ClienteFechaUltimoPedido: Fecha en la que el cliente realizó su ultimo pedido. Debe ser vacío al crear la cuenta.

3. Colecciones: Usado en la clasificación de productos. Parte de la estructuración específica del negocio particular de la moda, en el cuál se maneja el concepto de las colecciones, tales

como por ejemplo colección 2019, 2020, colección de verano, otoño. Cada nodo tiene los siguientes atributos:

- ColeccionColeccion: Código interno de la colección.
- ColeccionSigla: Nombre de la colección para desplegar en la interfaz de usuario.

4. Colores: Usado en la clasificación de productos e interfaz de usuario, cada instancia o unidad del producto, en este caso prenda, hace referncia a una existencia del producto, y tiene un color asociado, siendo tal que aunque sea el mismo producto puede venir en diferentes colores. Cada nodo tiene los siguientes atributos:

   - ColorColor: Código interno del color.
   - ColorHexadecimal: Código hexadecimal utilizado para identificar los valores RGB de cada color asociado a los productos, usado para la presentación en la interfaz de usuario.
   - ColorSigla: Nombre del color para desplegar en la interfaz de usuario.

5. DocumentosTipo: Lista de los posibles tipos de documentos a ingresar al realizar el registro de un cliente, o la identificación con documento de una persona o entidad interactuando con el sistema. Cada nodo tiene los siguietes atributos:

   - DocumentosTipoNombre: Nombre del tipo de documento a desplegar en la interfaz de usuario.
   - DocumentosTipoSigla: Identificador corto del tipo de documento.

6. Existencias: Contiene el recuento de existencias de los productos que vende la empresa, siendo tal que cada nodo corresponde a un trío de valor producto-talla-color, y sus cantidades existentes y comprometidas. Al restar las comprometidas a las existentes, se obtiene la cantidad de prendas disponibles para la venta. Cada nodo tiene los siguientes atributos:

   - ExistenciaColor: Código del color que corresponde a dicho registro de existencias.

- ExistenciaComprometidas: Cantidad de prendas correspondientes al trío producto-talla-color que se encuentran comprometidas o apartadas para su despacho al cliente.
- ExistenciaExisten: Cantidad de prendas correspondientes al trío producto-talla-color poseídas por la empresa para la venta.
- ExistenciaProductoCodigo: Código de inventario del producto que corresponde a dicho registro de existencias.
- ExistenciaTalla: Código de la talla que corresponde a dicho registro de existencias.

7. Grupos: Clasificación o división más específica de los productos, reúne y organiza bajo criterios cada línea. Cada nodo tiene los siguientes atributos:

   - GrupoGrupo: Código interno del grupo.
   - GrupoIVA: porcentaje del iva para el grupo, con este se realiza un control del requerimiento legal del IVA, y se pueden tener grupos exentos de IVA, o en el caso de que haya diferentes valores de IVA por el tipo de productos que estés en dicho grupo.
   - GrupoLinea: Código de la línea a la cuál pertenece el grupo.
   - GrupoSigla: Nombre del grupo para desplegar en la interfaz de usuario.

8. HistoricosMes: Contiene los históricos de clientes y de pedidos, por mes(cuando el año y el mes están definidos), por año (cuando el mes es cero y el año está definido) y el total (cuando el año y mes son cero). Contiene los dos siguientes subnodos:

   a) HistoricosMesClientes: Contiene los siguientes atributos:

      - HistoricosMesClientesCompran: De la totalidad de clientes, cantidad que ha realizado por lo menos un pedido en este lapso.
      - HistoricosMesClientesMonth: Mes correspondiente al lapso, puede tomar valores desde cero hasta doce.
      - HistoricosMesClientesNuevos: Cantidad de clientes que se han registrado en este lapso.

- HistoricosMesClientesTotal: Total acumulado de clientes que se han registrado hasta este lapso.
- HistoricosMesClientesYear: Año correspondiente al lapso.

b) HistoricosMesPedidos: Históricos de pedidos con respecto a un lapso y al estado de pedido. Contiene los siguientes atributos:

- HistoricosMesPedidosArticulosCantidad: Cantidad de artículos asociados a pedidos con dicho estado realizados en el lapso.
- HistoricosMesPedidosCantidad: Cantidad de pedidos con dicho estado realizados en el lapso.
- HistoricosMesPedidosEstado: Estado de pedidos al que corresponde este registro.
- HistoricosMesPedidosMonth: Mes correspondiente al lapso, puede tomar valores desde cero hasta doce.
- HistoricosMesPedidosValor: Suma del valor de los pedidos con dicho estado realizados en el lapso.
- HistoricosMesPedidosYear: Año correspondiente al lapso.

9. Lineas: Clasificación más general de los productos, reúne y organiza todo bajo un criterio. Cada nodo tiene los siguientes atributos:

   - LineaLinea: Código interno de la línea.
   - LineaSigla: Nombre de la línea para desplegar en la interfaz de usuario.

10. Logs: Contiene los registros de seguridad definidos para operaciones importantes o sensibles. Cada log cuenta con una base genérica independiente del tipo de log, constituida por los siguientes atributos:

    - LogUsuarioUid: Identificador de Firebase Authentication del usuario que realizó la operación.

- LogUsuarioCodigo: Código interno del usuario que realizó la operación.
- LogUsuarioNombre: Nombre del usuario que realizó la operación.
- LogFechaHora: Marca de tiempo cuando la operación fue realizada.
- LogTabla: A qué tabla de la base de datos afectó la operación.
- LogLog: Tipo de log o descripción del log.
- LogLlaveFirebase: Identificador auto-generado de Firebase del registro de la base de datos el cuál específicamente es afectado por la operación.

Además de esta base genérica, dependiendo de la tabla que afecta y el tipo de log guarda información específica de la operación:

- Tabla-Productos:
  a) Log-Crear: Guarda toda la información del nuevo producto creado. Tiene los siguientes atributos específicos:
    - ProductoCodigo.
    - ProductoColeccion.
    - ProductoComprometidas.
    - ProductoDescuentoPorcentaje.
    - ProductoExisten.
    - ProductoGrupo.
    - ProductoLinea.
    - ProductoMarca.
    - ProductoPrecioActual.
    - ProductoPrecioLista.
    - ProductoProducto.

  b) Log-Modificar Porcentaje de Descuento: Tiene los siguientes atributos específicos:

- ProductoCodigoInventario.
- ProductoDescuentoPorcentajeAnterior
- ProductoDescuentoPorcentajeNuevo

c) Log-Modificar Precio de Lista: Tiene los siguientes atributos específicos:

- ProductoCodigoInventario.
- ProductoPrecioListaAnterior.
- ProductoPrecioListaNuevo.

d) Log-Eliminar: Tiene el siguiente atributo específico:

- ProductoCodigoInventario.

- Tabla-Pedidos:

a) Log-Modificar Estado: Tiene los siguientes atributos específicos:

- PedidoEstadoAnterior
- PedidoEstadoNuevo
- PedidoNumero

- Tabla-Usuarios:

a) Log-Crear: Guarda toda la información del usuario creado, exceptuando la clave. Tiene los siguientes atributos específicos:

- UsuarioActivo.
- UsuarioApellidos.
- UsuarioCodigo.
- UsuarioEmail.
- UsuarioFechaCreacion
- UsuarioNombres
- UsuarioSuperusuario

b) Log-Modificar Clave: Tiene el siguiente atributo específico:

- UsuarioEmail.

c) Log-Modificar Información: Tiene los siguientes atributos específicos:

- UsuarioActivoAnterior.
- UsuarioActivoNuevo.
- UsuarioApellidosAnterior.
- UsuarioApellidosNuevo.
- UsuarioCodigoAnterior.
- UsuarioCodigoNuevo.
- UsuarioNombresAnterior.
- UsuarioNombresNuevo.

d) Log-Eliminar: Tiene el siguiente atributo específico:

- UsuarioEmail.

11. Marcas: Usado en la clasificación de productos. Cada nodo tiene los siguietes atributos:

    - MarcaImageLocation: Ubicación en el almacenamiento de la imagen asociada a la marca, para desplegarla en la interfaz de usuario.
    - MarcaMarca: Código interno de la marca.
    - MarcaSigla: Nombre de la marca para desplegar en la interfaz de usuario.

12. Numeracion: Información relevante para la numeración de las facturas generadas por el sistema. Cada nodo tiene los siguientes atributos:

    - NumeracionFactura: Número auto-incremental de la factura.
    - NumeracionObs1: Primer segmento de las observaciones a imprimir en factura.
    - NumeracionObs2: Segundo segmento de las observaciones a imprimir en factura.
    - NumeracionObs3: Tercer segmento de las observaciones a imprimir en factura.

- NumeracionPedido: Número auto-incremental del pedido.
- NumeracionPrefijo: Prefijo antepuesto al número de factura.
- NumeracionRes1: Primer segmento de la resolución de la DIAN a imprimir en factura.
- NumeracionRes2: Segundo segmento de la resolución de la DIAN a imprimir en factura.
- NumeracionRes3: Tercer segmento la resolución de la DIAN a imprimir en factura.

13. Pedidos: Cada nodo tiene los siguientes atributos:

    - PedidoADomicilio: Bandera de tipo booleano que indica si el pedido será entregado a domicilio o es para ser recogido en punto de venta.
    - PedidoBarrio: Barrio al que se entrega el pedido, en caso de ser a domicilio.
    - PedidoCantidadEntregada: Cantidad total de artículos que fueron entregados al cliente.
    - PedidoCantidadPedida: Cantidad total de artículos que pidió el cliente.
    - PedidoCiudad: Ciudad en la cuál se entrega el pedido, en caso de ser a domicilio.
    - PedidoClienteDocumento: Documento de identidad del cliente.
    - PedidoClienteId: Identificador interno del cliente.
    - PedidoClienteNombre: Nombre del cliente.
    - PedidoDireccion: Dirección de entrega del pedido, en caso de ser a domicilio.
    - PedidoEstado: Estado actual del pedido.
    - PedidoFacturaFecha: Fecha en la cual se generó la factura.
    - PedidoFacturaNumero: Número de factura asociado al pedido.
    - PedidoFacturaPrefijo: Prefijo de la factura asociada al pedido.
    - PedidoFecha: Fecha de realización del pedido por parte del cliente.

- PedidoFletes: Valor del servicio de entrega a domicilio.
- PedidoFormaDePago: Forma de pago elegida por el cliente.
- PedidoNota: Comentario o anotación escrita por el cliente de cualquier información extra con relación al pedido.
- PedidoNumero: Número único de identificación de pedido, autoincremental.
- PedidoObservacion: Comentario u observación escrita por el usuario de cualquier información extra con relación al pedido.
- PedidoTelefono: Teléfono de contacto del cliente.
- PedidoValorEntregado: Valor total de la mercancía entregada al cliente.
- PedidoValorMercancia: Valor total de la mercancía pedida por el cliente.

14. PedidosItems: Cada nodo tiene los siguientes atributos:

    - PedidoItemCantidadEntregada: Cantidad de artículos de dicho producto-talla-color que fueron entregados al cliente.
    - PedidoItemCantidadPedida: Cantidad de artículos de dicho producto-talla-color que fueron pedidos por el cliente.
    - PedidoItemColor: Color del producto solicitado por el cliente.
    - PedidoItemDescuentoPorcentaje: Porcentaje de descuento del producto al momento de la compra.
    - PedidoItemDescuentoValor: Valor del descuento aplicado al producto al momento de la compra.
    - PedidoItemIVA: IVA del producto, derivado del grupo.
    - PedidoItemPedidoNumero: Número del pedido al que este ítem corresponde.
    - PedidoItemPrecioLista: Precio de lista del producto al momento de la compra.

- PedidoItemPrecioVenta: Precio de venta del producto, después de aplicar el descuento, al momento de la compra.
- PedidoItemProductoCodigo: Código de inventario del producto.
- PedidoItemProductoProducto: Nombre del producto.
- PedidoItemSinCosto: Bandera de tipo booleano que indica si el ítem del pedido viene sin costo o gratuito.
- PedidoItemTalla: Talla del producto solicitado por el cliente.
- PedidoItemValorEntregado:

15. PreguntasFrecuentes: Preguntas frecuentes para mostrar al usuario, primera instancia de soporte antes de comunicarse directamente con personal de la tienda. Cada nodo tiene los siguientes atributos:

    - PreguntasFrecuentesPregunta.
    - PreguntasFrecuentesRespuesta.

16. Productos: Productos que se encuentran en el catálogo de la empresa. El código de inventario del producto corresponde a la unión del código de línea, el código de grupo y el código interno del producto, siendo tal un código de trece caracteres numéricos. Cada nodo tiene los siguientes atributos:

    - ProductoCodigo: Código de seis caracteres interno del producto. Tercera parte del código de inventario del producto.
    - ProductoColeccion: Código de la colección a la que pertenece el producto.
    - ProductoComprometidas: Cantidad de unidades comprometidas del producto, independiente de la talla y color.
    - ProductoDescuentoPorcentaje: Porcentaje de descuento actual del producto.

- ProductoExisten: Cantidad de unidades existentes del producto, independiente de la talla y color, poseídas por la empresa para la venta.
- ProductoGrupo: Código del grupo al que pertenece el producto. Segunda parte del código de inventario del producto.
- ProductoImageLocation:
- ProductoLinea: Código de tres caracteres de la línea a la que pertenece el producto. Primera parte del código de inventario del producto.
- ProductoMarca: Código de la marca a la que pertenece el producto.
- ProductoPrecioActual: Precio de venta actual del producto después de aplicar el descuento.
- ProductoPrecioLista: Precio base del producto antes de aplicar el descuento.
- ProductoProducto: Nombre del producto.

17. ProductosColores: Contiene las imágenes para cada producto de acuerdo a su color. Cada nodo tiene los siguientes atributos:
    - ProductoColorColorColor: Código del color.
    - ProductoColorImagenURL: Ubicación en el almacenamiento de la imagen asociada a dicho par producto-color, para desplegarla en la interfaz de usuario.
    - ProductoColorProductoCodigo: Código de inventario del producto.
18. PuntosVenta: Puntos de venta autorizados por la empresa. Cada nodo tiene los siguientes atributos:
    - PuntoVentaCodigo: Código interno del punto de venta.
    - PuntoVentaDescripcion: Descripción textual, que puede contener información al respecto de la ubicación, términos y condiciones, a discreción del administrador.

- PuntoVentaHorario: Texto en el cuál se describe el horario de atención del punto de venta.
- PuntoVentaImageURL: Ubicación en el almacenamiento de la imagen asociada al punto de venta, para desplegarla en la interfaz de usuario.

19. Tallas: Usado en la clasificación de productos e interfaz de usuario, cada instancia o unidad del producto, en este caso prenda, hace referencia a una existencia del producto, y tiene una talla asociada, siendo tal que aunque sea el mismo producto puede venir en diferentes tallas. Cada nodo tiene los siguientes atributos:
    - TallaTalla: Código interno de la talla.
    - TallaSigla: Nombre de la talla para desplegar en la interfaz de usuario.
20. Usuarios: Registrados en la aplicación del administrador, que pueden ser de tipo superusuario o repartidor. Cada nodo tiene los siguientes atributos:
    - UsuarioActivo: Bandera de tipo booleano para identificar que el usuario se encuentra activo en el sistema, y puede ingresar e interactuar con la aplicación, o se le deniega el acceso.
    - UsuarioApellidos.
    - UsuarioCodigo: Código interno del usuario.
    - UsuarioEmail: Correo electrónico del usuario, parte de las credenciales para el login.
    - UsuarioFechaCreacion: Marca de tiempo de registro del usuario en la plataforma.
    - UsuarioImageURL: Imagen de perfil de usuario.
    - UsuarioNombres.
    - UsuarioSuperusuario: Bandera de tipo booleano para identificar el tipo de usuario, si es superusuario y puede acceder a las funcionalidades completas del sistema, o si es repartidor y sólo tiene acceso a las funcionalidades asociadas al despacho.

6.3. IMPLEMENTACIÓN DE LA METODOLOGÍA

Con base en la metodología de desarrollo Ágil Scrum el proyecto fue desarrollado como se describe a continuación.

6.3.1. Equipo de trabajo:

- **Representante de la empresa**: Juan Camilo Moreno.
- **Desarrollador**: Diego Fernando Medina Blanco y Luis Ernesto Páez Ortiz.
- **Tester**: Juan Camilo Moreno.
- **Cliente**: Clientes MOI Colombia.

6.3.2. Lista de requerimientos A continuación se presentan la lista de requerimientos de software,

No.	Solicitante	Característica	Funcionalidad	Rol objetivo
1	Administrador MOI	Ver productos	Visualizar los productos de la empresa MOI.	Cliente
2	Administrador MOI	Ver fotos	Visualizar las fotos de los productos en detalle.	Cliente
3	Administrador MOI	Añadir tabla de tallas	El administrador especifica que puede ser una simple foto por cada producto para permitir al cliente de forma visual ver las tallas de los productos.	Cliente
4	Administrador MOI	Carrito de compras	Sección que permita visualizar los productos que desea el cliente antes de proceder al pago	Cliente

5	Administrador MOI	Pasarela de pagos	Posibilidad para recoger el producto en la tienda o a través de un envío. Es deseable tener pagos a través de PayU	Cliente
6	Administrador MOI	Estado de pedido	El cliente y el administrador deben conocer el estado de su pedido en tiempo real.	Cliente
7	Administrador MOI	Gestionar perfil	El cliente tiene la posibilidad de gestionar datos de su perfil	cell6
8	Administrador MOI	Chatear	Requerimiento deseable pero no obligatorio. Posibilidad para chatear con el cliente. Puede ser un link directo a whatsapp.	Cliente
9	Administrador MOI	Registro con Gmail o Facebook.	El cliente debe poder registrarse con su email así como también a través de su cuenta de google o cuenta de Facebook.	Cliente
10	Administrador MOI	Ver pedidos y entregarlos	Repartidor debe tener un modulo de pedidos y permitir entregarlos.	Administrador
11	Administrador MOI	Historial de pedidos.	Administrador puede conocer el estado de los pedidos realizados por los clientes así como cancelarlos.	Administrador
12	Administrador MOI	Ver información del cliente	Gestionar a los clientes de MOI y conocer su información	Administrador
13	Administrador MOI	CRUD de productos	Capacidad para la creación, lectura, edición y eliminación de productos existentes así como su inventario.	Administrador

14	Administrador MOI	Cambio de foto	El administrador tiene la posibilidad de cargar una nueva foto para los productos de la tienda	Administrador
15	Administrador MOI	Enviar notificaciones	El objetivo es poder enviar notificaciones a los clientes al gestionar pedidos.	Administrador
16	Administrador MOI	Bloquear clientes	El administrador tiene la posibilidad de inhabilitar una cuenta de usuario.	Administrador

6.3.3. Metodología Scrum para validación de progreso Se acuerda con la administración de MOI realizar 4 Sprints tal como lo establece la metodología de desarrollo Ágil, que se definieron como sigue:

No. Sprint	**Sprint**
1	Verificación de requerimientos y diseño de la Aplicación
2	Catálogo de productos
3	Canastilla de compras
4	Generación de pedidos a partir de la canastilla de compras
5	Gestión de pedidos y facturación de pedidos pendientes
6	Históricos, logs y preguntas frecuentes
7	Importación y exportación masiva
8	Manejo de sesiones y autenticación
9	Menú, corrección de errores y ajustes finales

6.3.4. Desarrollo de Sprints

Primer Sprint: Verificación de requerimientos y diseño de la Aplicación. Para la realización de este este primer Sprint se realizó un análisis de requerimientos profundo, resolución de dudas con la administración de la compañía con base en el análisis y primeros pasos para la creación de una óptima estructura de datos.

El primer Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Muestra de casos de éxito por parte de Juan Camilo Moreno y visualización de Mockup hecho por él	Juan Camilo Moreno
2	Validación de requerimientos: Se explica a Juan Camilo las dificultades que existen para la activación de pagos en línea debido a la falta de paquetes en el framework para realizar una transacción de forma segura	Luis Páez
3	Definición de una estructura mínima funcional de la base de datos.	Diego Medina - Luis Páez
4	Definición de roles de usuario	Diego Medina - Luis Páez
5	Formalización de las vistas principales de la aplicación del cliente	Diego Medina - Luis Páez
6	Definición de las vistas principales asociadas al catálogo de productos app Moi Admin	Diego Medina - Luis Páez
7	Diseño de diagrama UML Entidad relación.	Diego Medina - Luis Páez

Al finalizar el primer Sprint pudimos obtener una consolidación de requerimientos válidos con base en las expectativas del cliente y las capacidades del sistema, así como también se definió la estructura de datos base mínima del sistema y los roles de usuario presentes en la solución software.

Segundo Sprint: Catálogo de productos Primeros pasos de aplicación móvil de cliente mostrando catálogo de productos en la interfaz inicial. Durante este Sprint se hizo uso de la estructura de datos de productos y su implementación para ser mostrados. Esta etapa permitió identificar la necesidad de crear una nueva Entidad en la estructura de datos llamada Existencia para poder realizar un manejo adecuado de los productos y su inventario. Primeros pasos de aplicación móvil de administración, definiendo las vistas para el soporte y cargue inicial del catálogo de productos.

El segundo Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Configuración de la base de datos utilizando el SDK de Firebase para Flutter junto con las normas de seguridad	Luis Ernesto Páez Ortiz
2	Configuración de ambiente de trabajo	Diego Medina - Luis Páez
3	Agregar datos de prueba a la base de datos	Luis Ernesto Páez
4	Construcción de vistas para la muestra de productos app Moi	Luis Páez
5	Construcción de vistas para CRUD de Líneas app Moi Admin	Diego Medina
6	Construcción de vistas para CRUD de Grupos app Moi Admin	Diego Medina
7	Construcción de vistas para CRUD de Colecciones app Moi Admin	Diego Medina
8	Construcción de vistas para CRUD de Marcas app Moi Admin	Diego Medina
9	Construcción de vistas para CRUD de Productos app Moi Admin	Diego Medina

10	Construcción de vistas para CRUD de Tallas app Moi Admin	Diego Medina
11	Construcción de vistas para CRUD de Colores app Moi Admin	Diego Medina
12	Construcción de vistas para CRUD de Imágenes para Producto por Color app Moi Admin	Diego Medina
13	Construcción de vistas para CRUD de Existencias app Moi Admin	Diego Medina
14	Validación con el administrador de la tienda	Luis Ernesto Páez - Juan Camilo Moreno

Al finalizar el segundo Sprint fue posible obtener una vista funcional para mostrar productos recientes, ver los detalles del producto así como los colores, tallas y unidades disponibles en la aplicación Moi Cliente. Se pueden apreciar las vistas correspondientes de este Sprint en las figuras 6, 7, 8. Como resultado de este sprint en la aplicación Moi Admin se obtuvieron las vistas funcionales para realizar el CRUD hacia todas las entidades necesarias para la creación de los productos y sus correspondientes existencias. Para cada entidad se siguió el mismo patrón de diseño, se desarrolló una página para visualización como la que se puede apreciar en la figura 9 con una barra de búsqueda y filtros como se puede apreciar en la figura 10 y otra para creación y edición, que se puede apreciar en la figura 11. La vista construida para la visualización de existencias se puede apreciar en la figura 12 y la desarrollada para la modificación de existencias (añadir o sustraer) se puede observar en la figura 13.

Tercer Sprint: Canastilla de Compras Durante este Sprint se creó una vista para la canastilla de compras así como la validación de diferentes escenarios a los que un cliente se puede someter a la hora de utilizar una canastilla de compras, además de complementar la definición inicial de la base de datos y preparar las funcionalidades extra necesarias para la generación del pedido desde la app Moi Admin en el siguiente sprint.
El tercer Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Creación de la vista de canastilla de compras	Luis Ernesto Páez
2	Envío de productos a la canastilla de compras del cliente en la base de datos desde la vista de detalles.	Luis Ernesto Páez
3	Validación de existencias antes de agregar el producto a la canastilla	Luis Ernesto Páez
4	Validación de existencias en la vista de la canastilla de compras antes de proceder al pago..	Luis Ernesto Páez
5	Construcción de vistas para CRUD de Ciudades app Moi Admin	Diego Medina
6	Construcción de vistas para CRUD de Puntos de Venta app Moi Admin	Diego Medina
7	Construcción de vista para visualización y edición de Numeración app Moi Admin	Diego Medina
8	Ajustes de la interfaz y validación con el administrador de la tienda	Luis Ernesto Páez - Juan Camilo Moreno

Al finalizar el tercer Sprint el cliente cuenta la capacidad de agregar los productos que desea a su canastilla de compras, verificando que los productos existan. En la app Moi Admin se obtienen las vistas para realizar el CRUD de ciudades y puntos de venta que serán asociados a los, con

una estructura muy similar a las de CRUD obtenidas en el sprint 2. También se obtiene la vista para visualización y edición de Numeración que se puede apreciar en la figura 15

Cuarto Sprint: Generación de pedidos a partir de la canastilla de compras Durante este Sprint se creó una vista para la ejecutar la generación de pedidos a partir de los datos recibidos en una canastilla de compras en la aplicación del cliente. Se tuvieron en cuenta diferentes escenarios tales como recoger el producto en un punto de venta o recibirlo a domicilio. Por otro lado, se realiza la transición desde la vista de la canastilla de compras a la vista de generación de pedido. En la app Moi Admin se implementan los descuentos para productos y las vistas para manejar el CRUD de los usuarios.
El cuarto Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Creación de la vista para la captación de datos de envío definidos en la estructura de datos	Luis Ernesto Páez
2	Generación de pedidos para envío a domicilio	Luis Ernesto Páez
3	Generación de pedidos para recoger en punto de venta	Luis Ernesto Páez
4	Persistencia de datos del pedido	Luis Ernesto Páez
5	Comprometer prendas y realizar registros necesarios para mantener la integridad de los datos	Luis Ernesto Páez
6	Construcción de vista para aplicar descuentos a productos	Diego Medina
6	Construcción de vistas para CRUD de usuarios	Diego Medina
7	Validación de interfaz.	Juan Camilo Moreno

Al finalizar el cuarto Sprint el cliente cuenta con la posibilidad de realizar un pedido con todos los items que contiene su canastilla de compras, como se puede evidenciar en las figuras 16 y 17

En la app Moi Admin se obtienen la vista funcional para que el usuario pueda aplicar descuentos a los productos por Línea, Grupo, Colección o Marca, como se puede apreciar en la figura 18. Y las vistas para el manejo del CRUD de usuarios con una estructura similar a la evidenciada en las vistas de CRUD de los anteriores dos sprints.

Quinto Sprint: Gestión de pedidos y facturación de pedidos pendientes Durante este Sprint se procedió a implementar las vistas y la lógica para ver y entregar pedidos con facturación incluida por parte del usuario con rol de repartidor, y para ver y cancelar pedidos por parte del usuario con rol de administrador con la posibilidad de reimprimir factura.
El quinto Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Creación de la vista de pedidos pendientes para el usuario de rol repartidor	Diego Medina
2	Creación de la vista de detalles del pedido y entrega para el usuario de rol repartidor	Diego Medina
3	Creación de la vista de notas y observaciones del pedido para el usuario de rol repartidor	Diego Medina
4	Creación de la vista de pedidos con la funcionalidad de cancelación para el usuario de rol de administrador	Diego Medina
5	Creación de la vista de detalles del pedido y reimpresión de factura para el usuario de rol de administrador	Diego Medina
6	Creación de la vista de notas y observaciones del pedido para el usuario de rol de administrador	Diego Medina
7	Vista para la gestión de usuarios.	Diego Medina
8	Validación de progreso con el administrador.	Diego Medina - Luis Páez - Juan Camilo Moreno

Al finalizar el quinto Sprint el administrador puede visualizar y gestionar los pedidos filtrando y cancelando en caso de ser necesario, y reimprimir facturas, como se puede apreciar en las figuras 19, 20 y 21. También se obtienen las vistas para que el repartidor pueda visualizar y gestionar los pedidos pendientes, filtrando y entregando eligiendo la cantidad entregada, pasando el pedido a un estado ya sea completo o incompleto, y generando la factura correspondiente, como se puede apreciar en las figuras 22, 23 y 24.

Sexto Sprint: Históricos, logs y preguntas frecuentes Durante este Sprint se procedió a implementar la parte de análisis y seguridad del sistema, con los históricos y los logs de operaciones definidas como sensibles en el sistema.
El sexto Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Implementación de la lógica para guardar históricos de pedidos e históricos de clientes	Diego Medina - Luis Páez
2	Creación de la vista de consultar estadísticas para visualizar históricos de pedidos y clientes	Diego Medina
3	Definición de las operaciones sensibles que requieren el registro de logs de seguridad	Diego Medina
4	Definición de la estructura de los logs de seguridad	Diego Medina
5	Implementación de la lógica para realizar el registro de logs de seguridad en todas las operaciones definidas como sensibles	Diego Medina
6	Creación de vistas para el CRUD de preguntas frecuentes en la app Moi Admin	Diego Medina
7	Creación de vista para visualización de preguntas frecuentes en la aplicación del cliente	Luis Páez

8	Validación de progreso con el administrador.	Diego Medina - Luis Páez - Juan Camilo Moreno

Al finalizar el sexto Sprint el administrador puede visualizar los históricos en el apartado de consultar estadísticas, por mes, año y desde el inicio del uso de la solución, como se puede apreciar en la figura 25. Asimismo, se obtuvo el diseño de una estructura de log para las operaciones definidas como sensibles y luego de crear estos logs se registran en la base de datos. También se obtuvieron las vistas de CRUD de preguntas frecuentes en la app Moi Admin con una estructura muy similar a utilizada en sprints anteriores, y se obtuvo la vista para visualización de preguntas frecuentes en la aplicación de cliente.

Séptimo Sprint: Importación y exportación masiva Durante este Sprint se procedió a implementar la funcionalidad de importación para el cargue masivo de datos para las entidades pertinentes en la base de datos, además de la exportación del contenido de estas "tablas"de entidades.

El séptimo Sprint se resume a continuación en las siguientes historias de usuario:

No.	**Nombre**	**Encargado**
1	Definición de entidades a importar, cabeceras y formatos de importación	Diego Medina
2	Definición de entidades a exportar, cabeceras y formatos de exportación	Diego Medina
3	Implementación de la lógica de importación para cada entidad determinada	Diego Medina
4	Creación de vista para importación	Diego Medina
5	Implementación de la lógica de exportación para cada entidad determinada	Diego Medina

6	Creación de vista para exportación	Diego Medina
7	Validación de progreso con el administrador.	Diego Medina - Luis Páez - Juan Camilo Moreno

Al finalizar el séptimo Sprint el administrador puede realizar por medio de archivos con formato csv el cargue masivo de Marcas, Colecciones, Tallas, Colores, Líneas, Grupos, Productos y Existencias con las cabeceras y estructura definidas como se puede apreciar en las vistas ilustradas en las figuras 26 y 27, así como también puede realizar la exportación de tablas de las entidades mencionadas anteriormente además de Pedidos, con la particularidad de que para esta última permite el filtrado de dichos pedidos a exportar por estado, por tipo de entrega y/o a un lapso de tiempo, como se puede apreciar en la figura 28. Un ejemplo de archivo exportado de pedidos se puede observar en la figura 29, así como un ejemplo de archivo para importación de existencias en la figura 30.

Octavo Sprint: Manejo de sesiones y autenticación .

En este sprint se construyeron las vistas y funcionalidades ligadas al manejo de sesiones y autenticación dentro de la aplicación.

No.	**Nombre**	**Encargado**
1	Desarrollo de la lógica para creación de usuarios y clientes en Firebase Authentication ligados a los registros de usuarios y clientes en la base de datos	Diego Medina - Luis Páez
2	Desarrollo de la lógica para inicio de sesión, generación y guardado de información de sesión activa y token de autenticación de Firebase Authentication	Diego Medina - Luis Páez
3	Refactor para envío del token de autenticación en las consultas http en la app MoiAdmin	Diego Medina

4	Construir la vista de inicio de sesión y reinicio de clave para la app Moi Admin	Diego Medina
5	Construir la vista de registro, inicio de sesión y restauración de clave en la aplicación del cliente	Luis Páez
6	Implementar la funcionalidad de envío de email para restaurar clave	Diego Medina
7	Validación de progreso con el administrador	Diego Medina - Luis Páez - Juan Camilo Moreno

Al finalizar el octavo Sprint se obtienen las vistas funcionales para el inicio y manejo de sesiones de usuario, junto con su relación a las llamadas a la base de datos. Para la app Moi Admin se puede apreciar la vista del inicio de sesión en la figura 31, la vista de restauración de clave en la figura 32.

Noveno Sprint: Menú, corrección de errores y ajustes finales .

No.	**Nombre**	**Encargado**
1	Adición de módulo para lista de deseos en la aplicación del cliente	Luis Ernesto Páez
2	Consolidación y versión definitiva del menú de navegación de tipo Drawer para la app Moi Admin	Diego Medina
3	Correción y ajustes de navegación para la app Moi Admin	Diego Medina
4	Adición de módulo para preguntas, quejas, reclamos y felicitaciones en la aplicación del cliente.	Diego Medina
5	Adición de módulo para ubicación de las tiendas en la aplicación del cliente.	Luis Ernesto Páez

6	Verificación del funcionamiento de las dos aplicaciones en conjunto.	Diego Medina - Luis Páez - Juan Camilo Moreno

Al finalizar el noveno Sprint se validó el correcto funcionamiento de la solución integrada a través de la creación de pedidos, verificando diferentes escenarios tales como entrega completa de pedido, cancelación, entrega parcial, así como el uso de la canastilla de compras en tiempos asíncronos para la correcta gestión del inventario en el sistema. Se válida un prototipo de software útil.

Para la app Moi Admin para el rol de administrador se puede apreciar el menú drawer en la figura 33, el menú seleccionando la opción de Personalización del sistema en la figura 34 y el menú seleccionando las opciones clasificación y codificación en la figura 35.

Figura 6. Vista inicial de la aplicación móvil del cliente

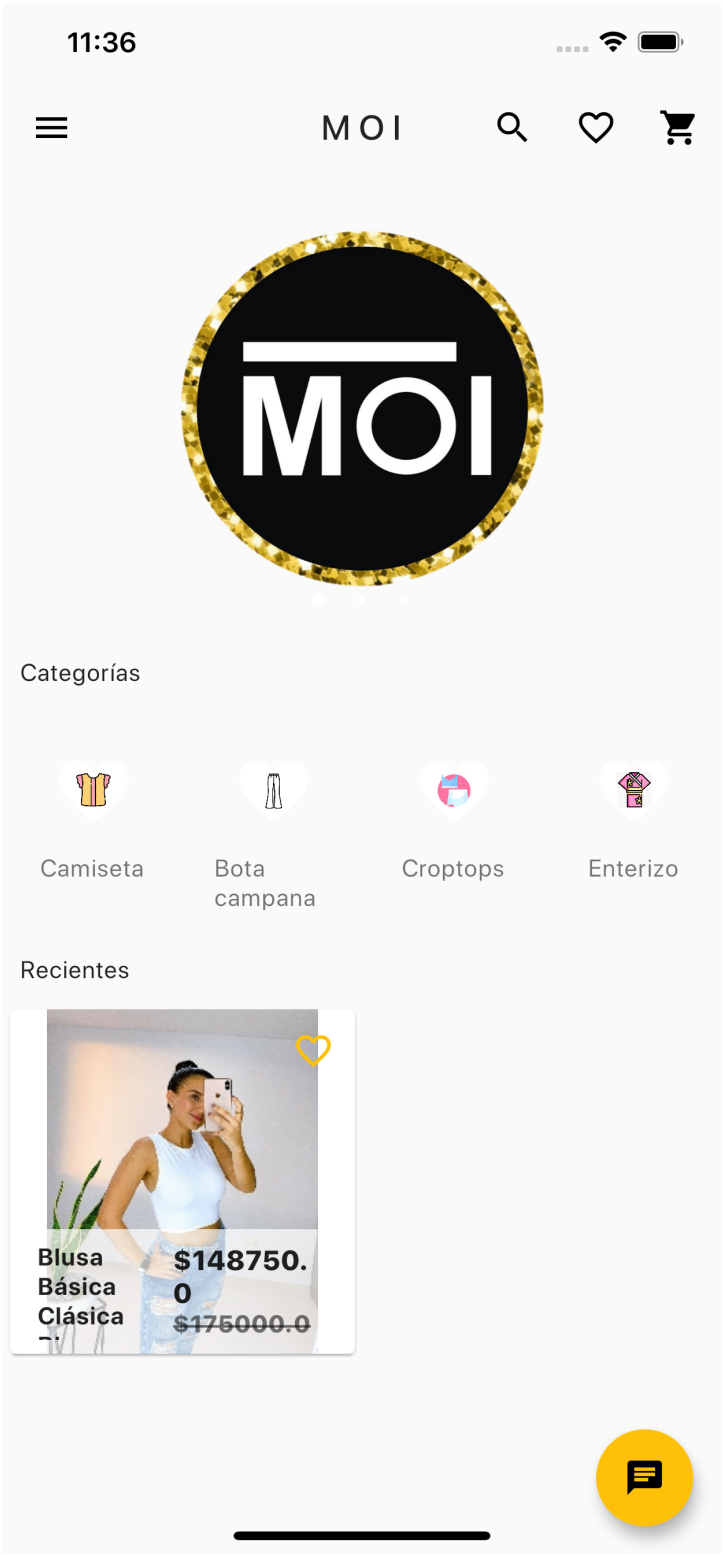


Figura 7. Menú lateral principal aplicación móvil del cliente

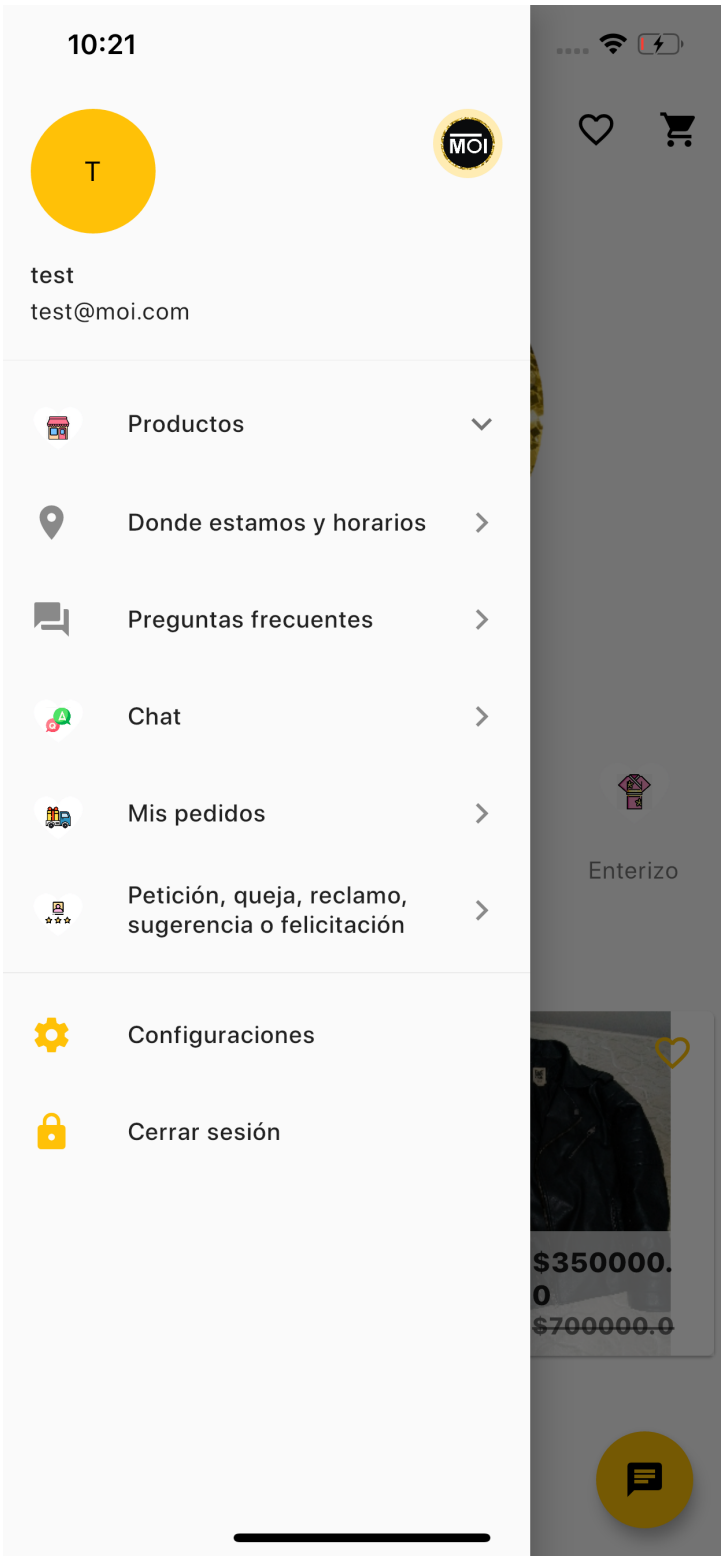


Figura 8. Vista para conocer los detalles de un producto con base en los requerimientos de la empresa

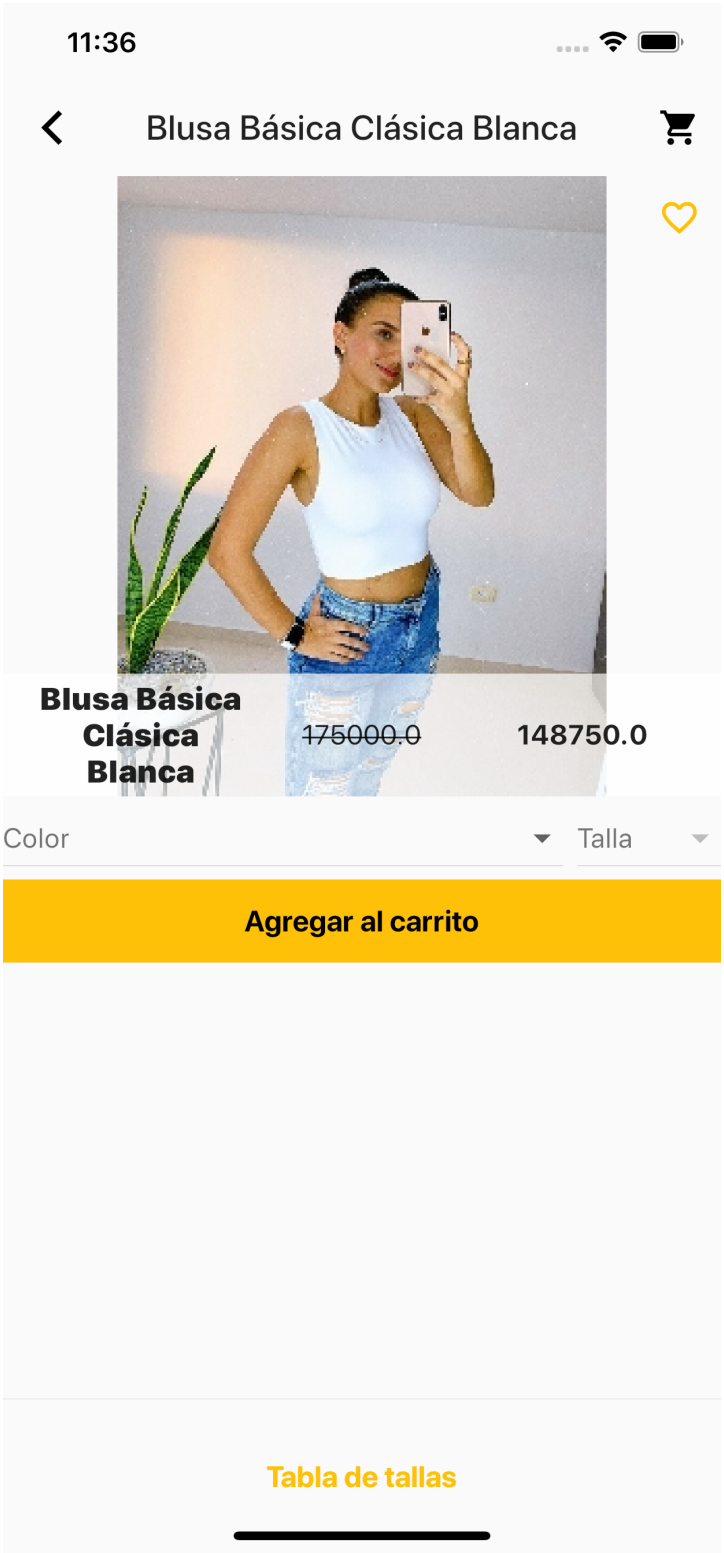


Figura 9. Vista para visualizar Productos

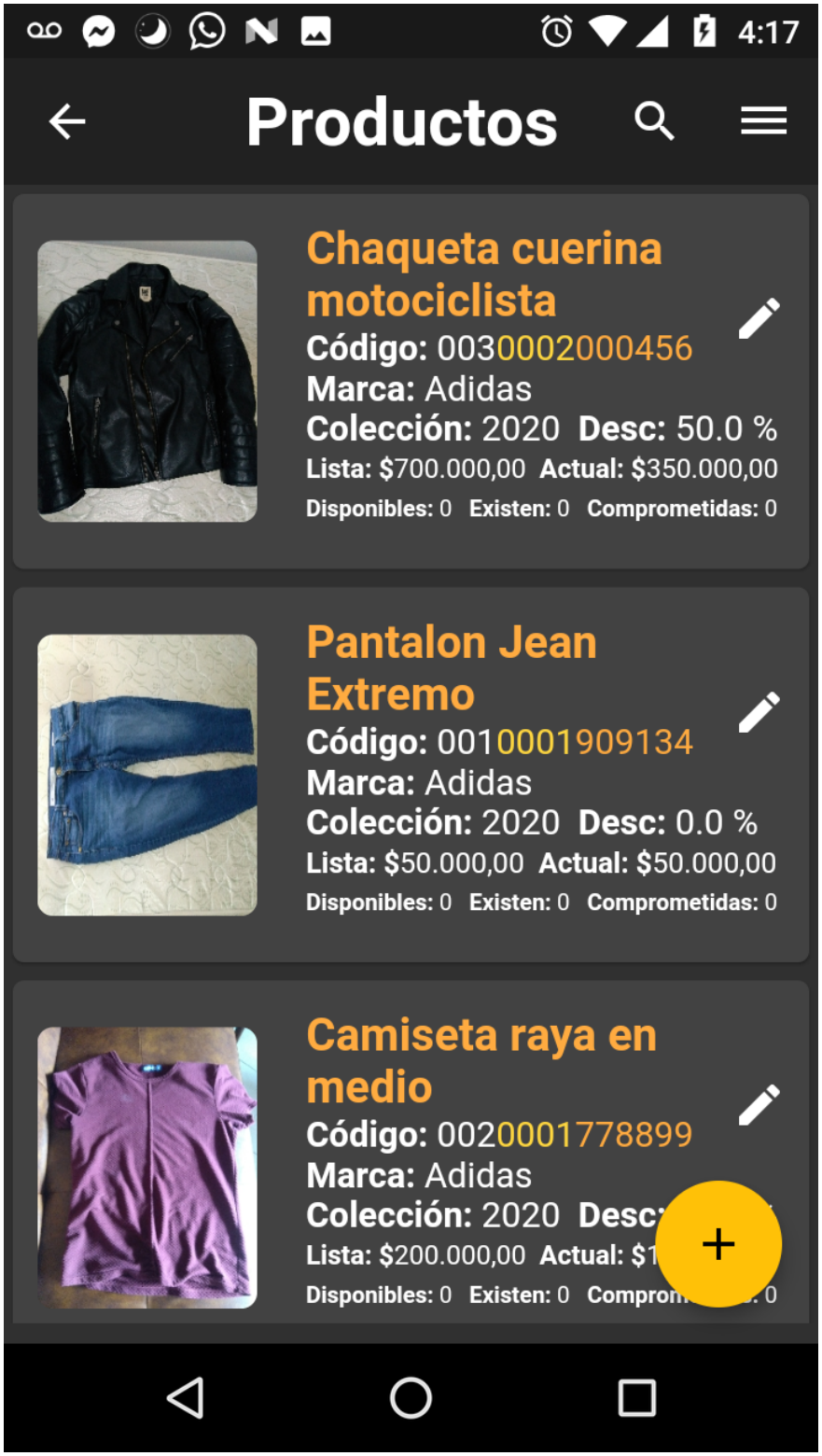


Figura 10. Vista para buscar y filtrar productos

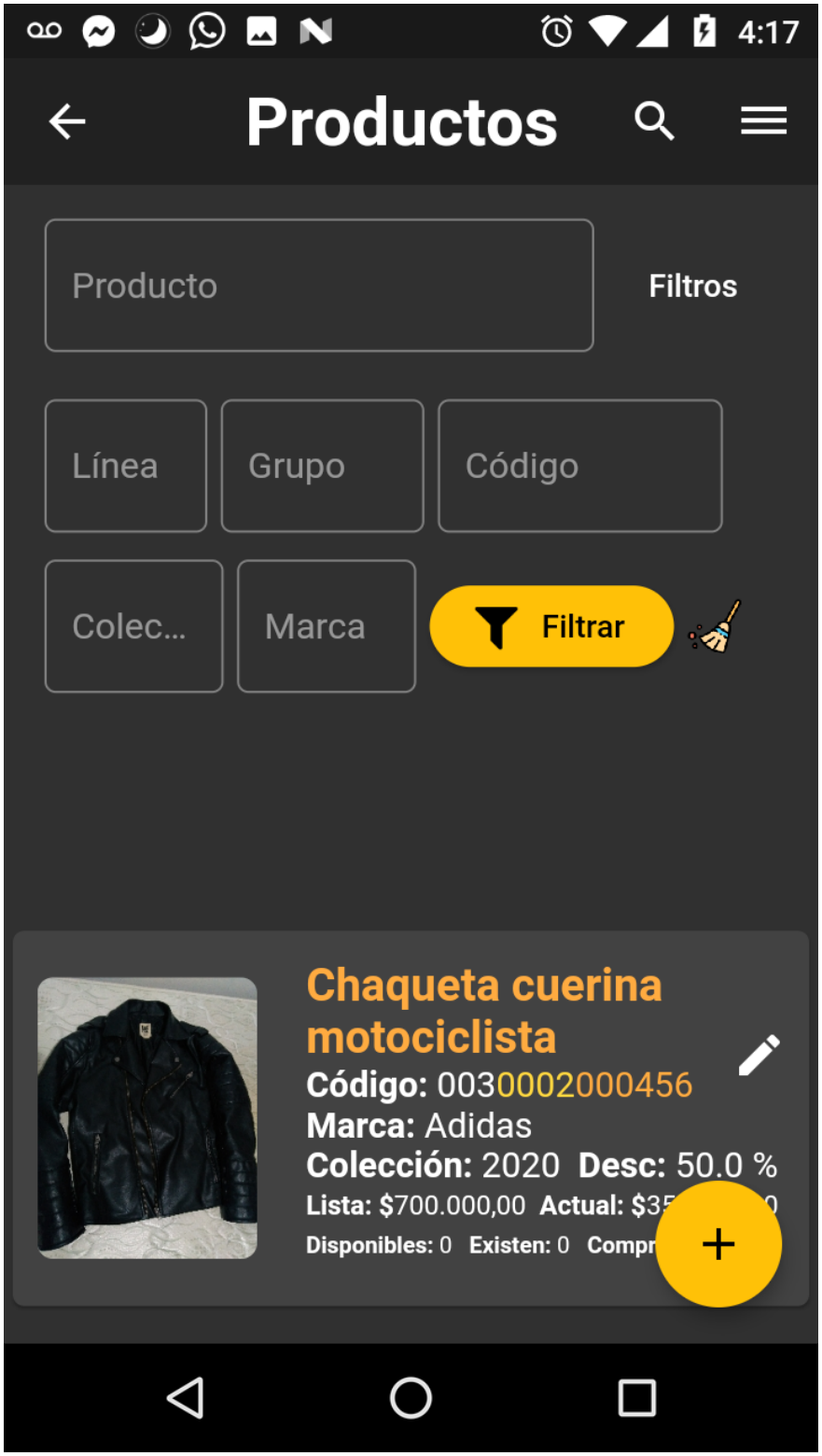


Figura 11. Vista para crear o editar Productos

Figura 12. Vista para visualizar existencias

Figura 13. Vista para modificar Existencias

Figura 14. Vista de la canastilla de compras

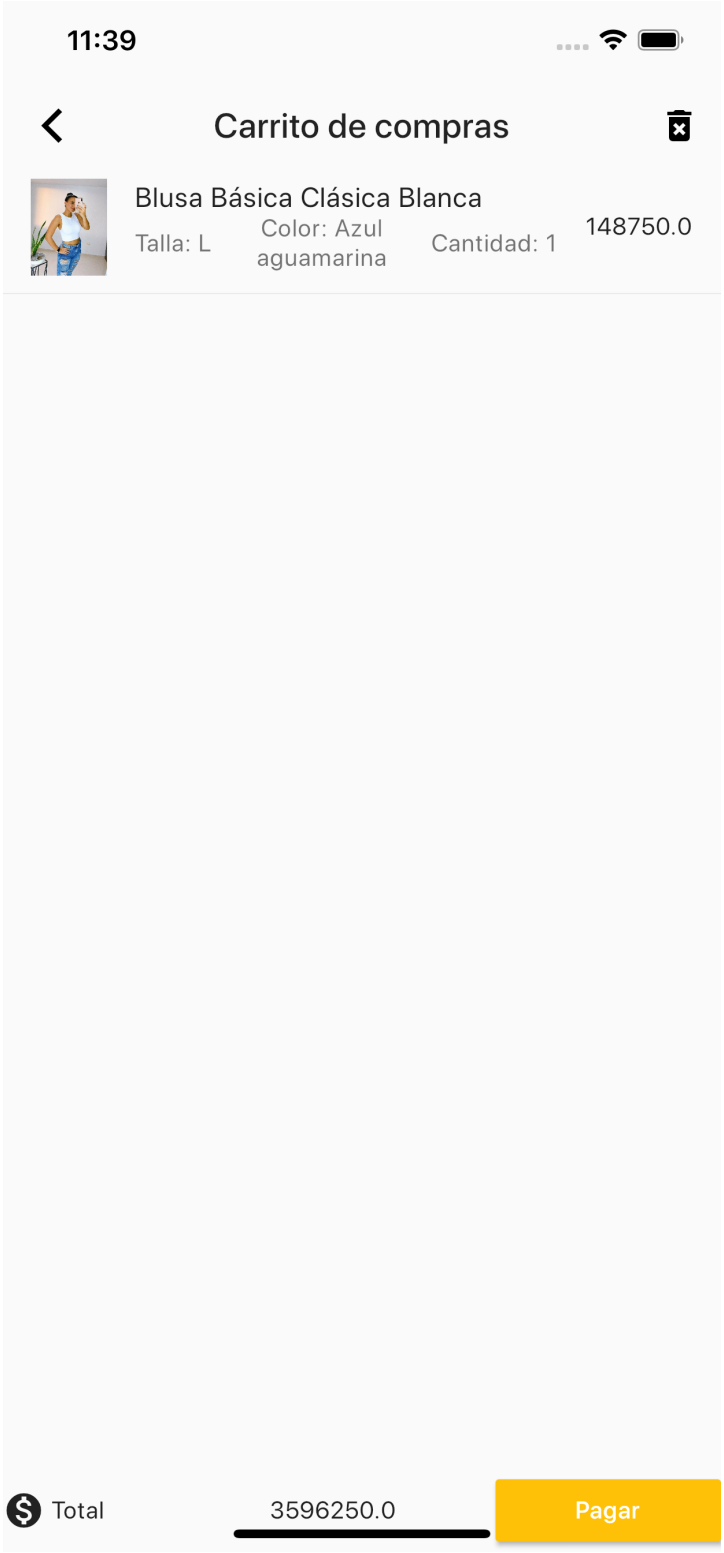


Figura 15. Vista de la numeración

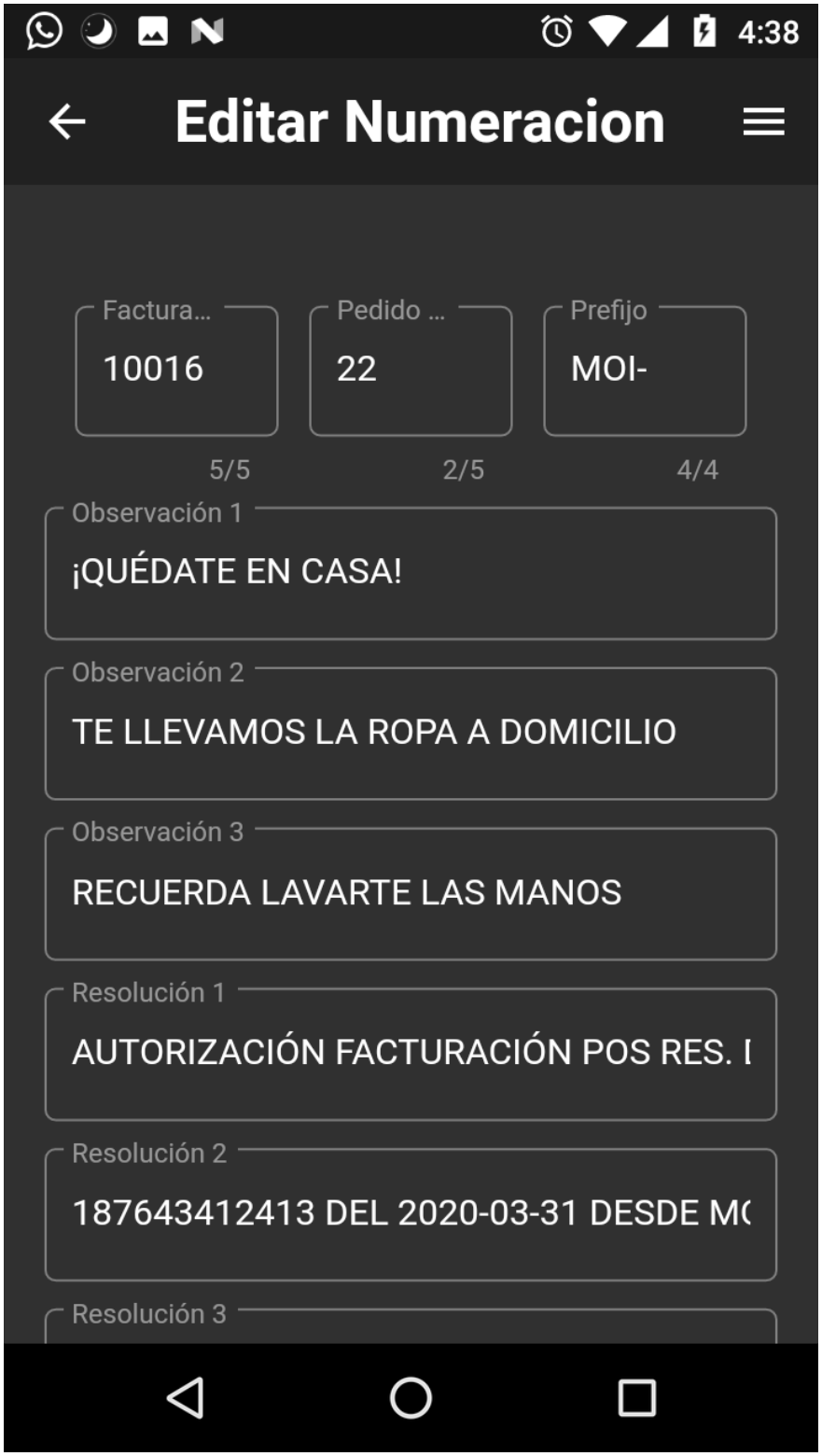


Figura 16. Vista para la creación de pedidos a domicilio en la aplicación del cliente

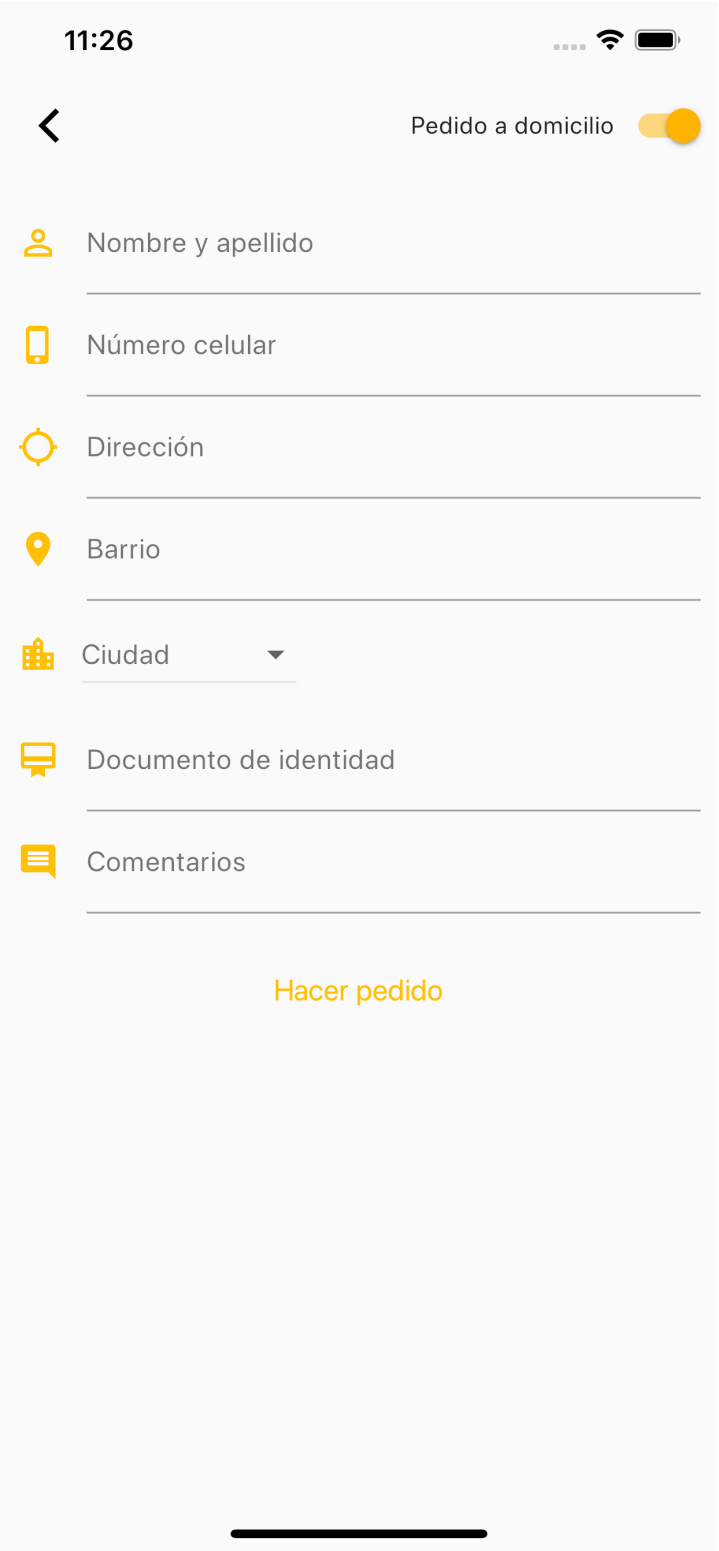


Figura 17. Vista para la creación de pedidos que serán recogidos en la tienda en la aplicación del cliente

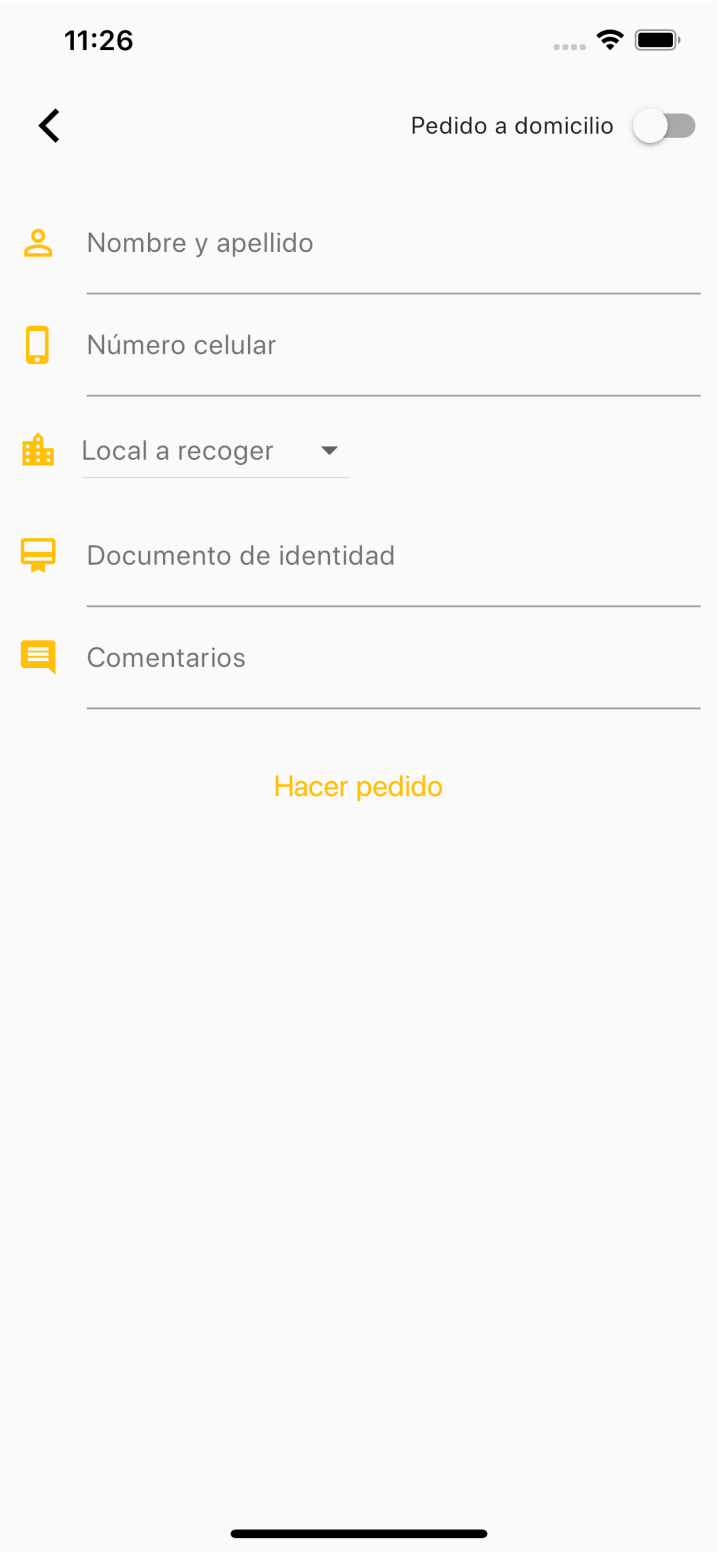


Figura 18. Vista de descuentos por tabla

Figura 19. Vista de Pedidos para administrador

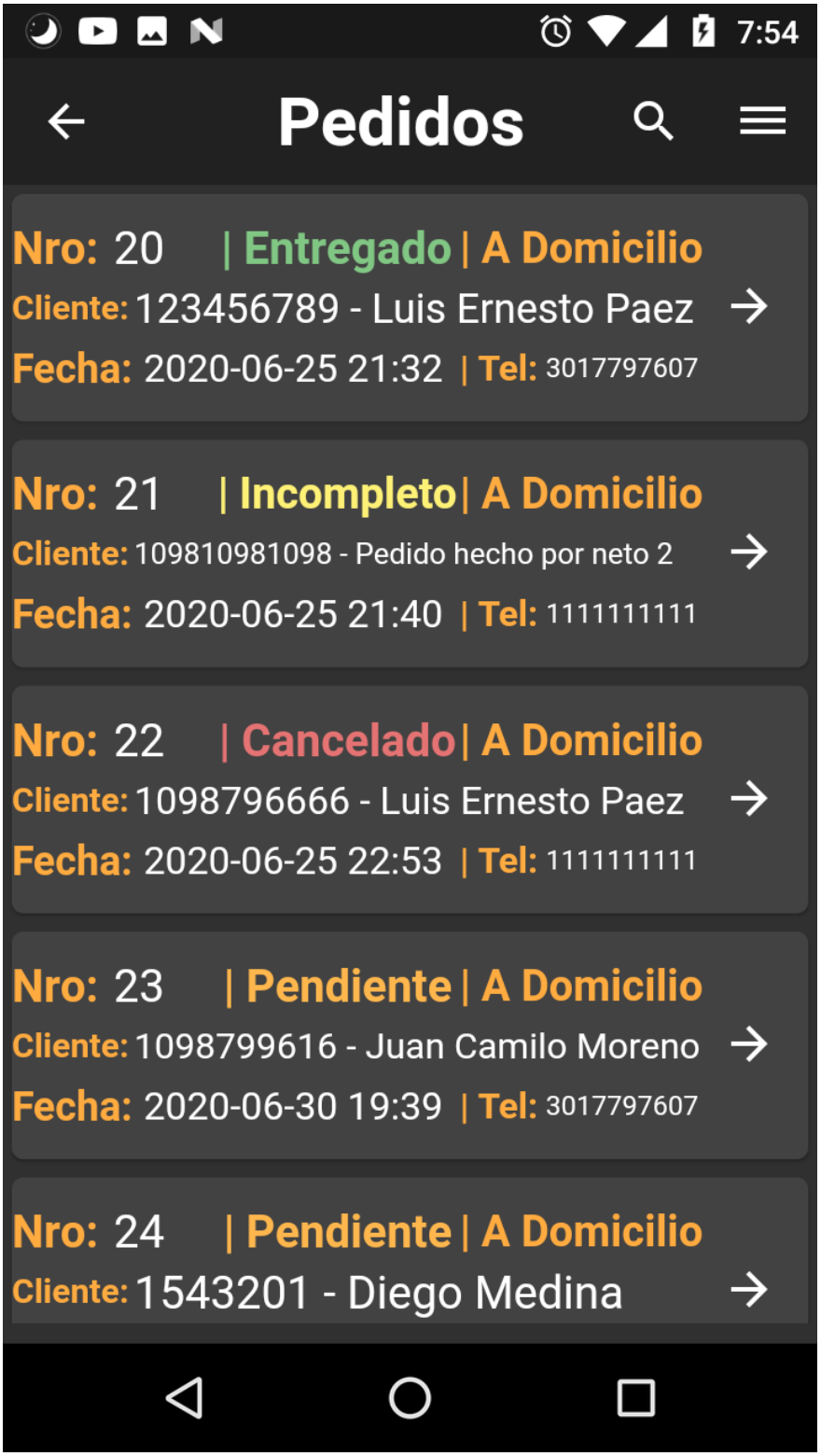


Figura 20. Vista de filtros para Pedidos para administrador

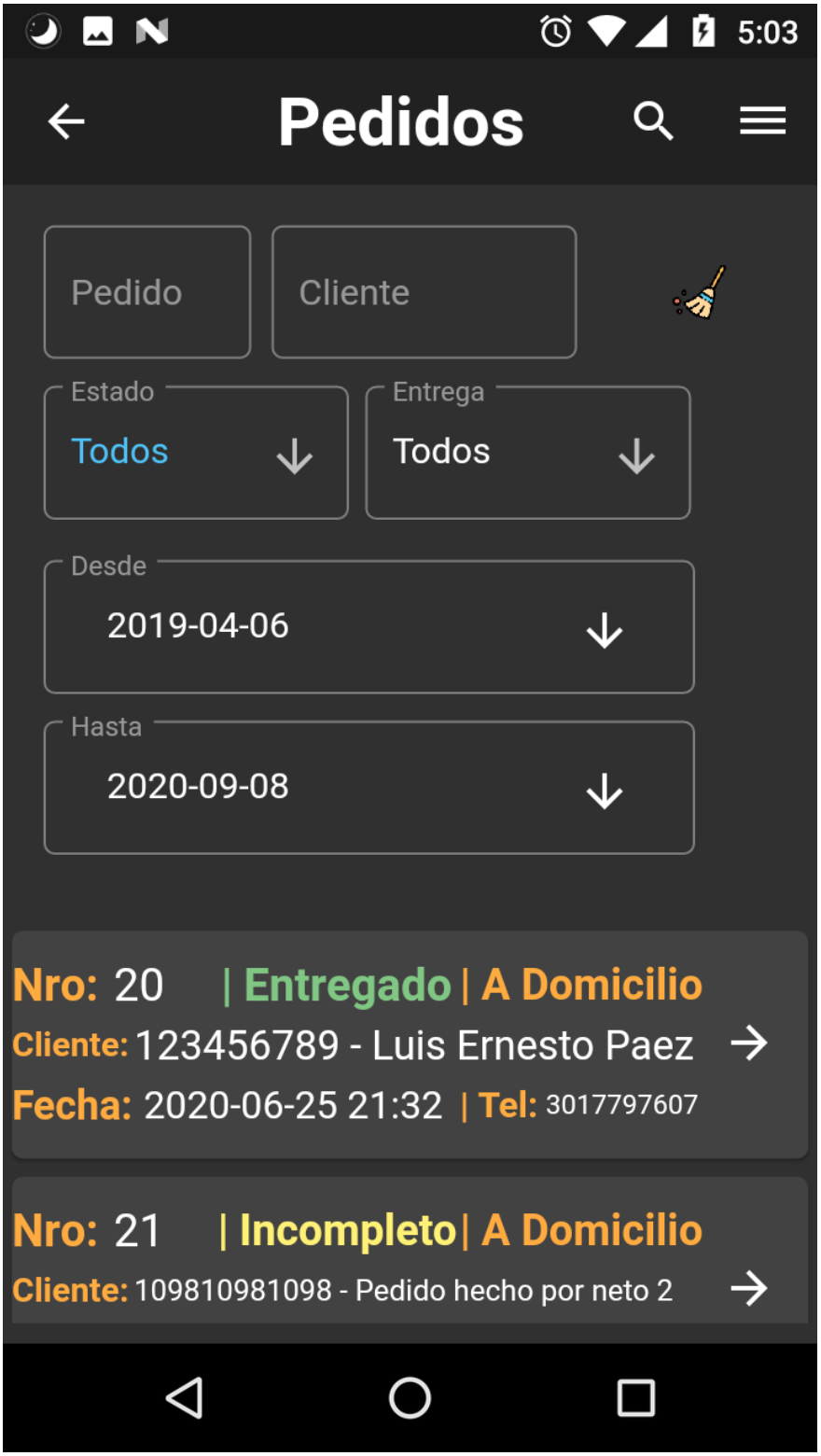


Figura 21. Vista de detalles del pedido para administrador

Figura 22. Vista de pedidos pendientes para repartidor

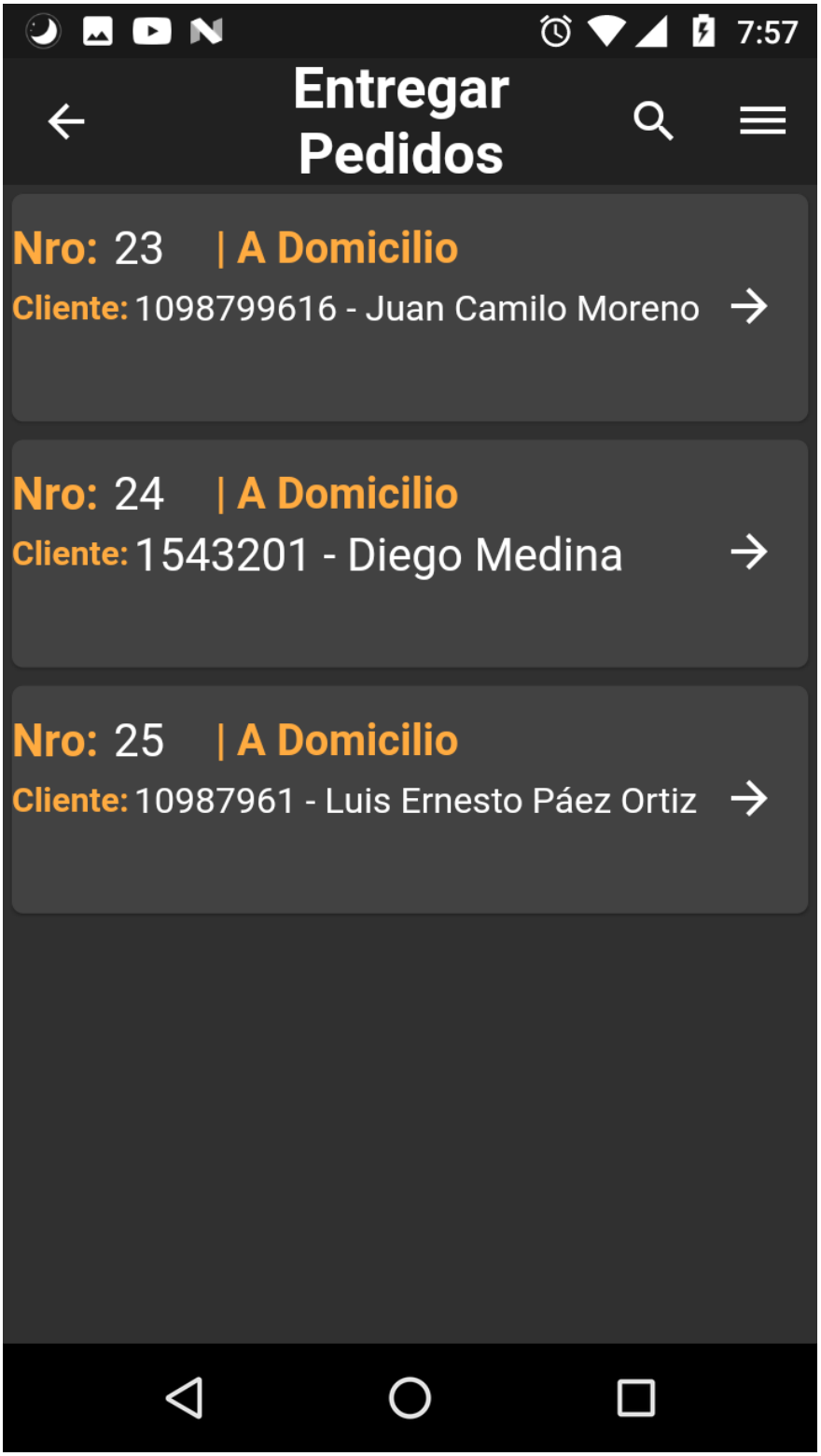


Figura 23. Vista de detalles del pedido para repartidor

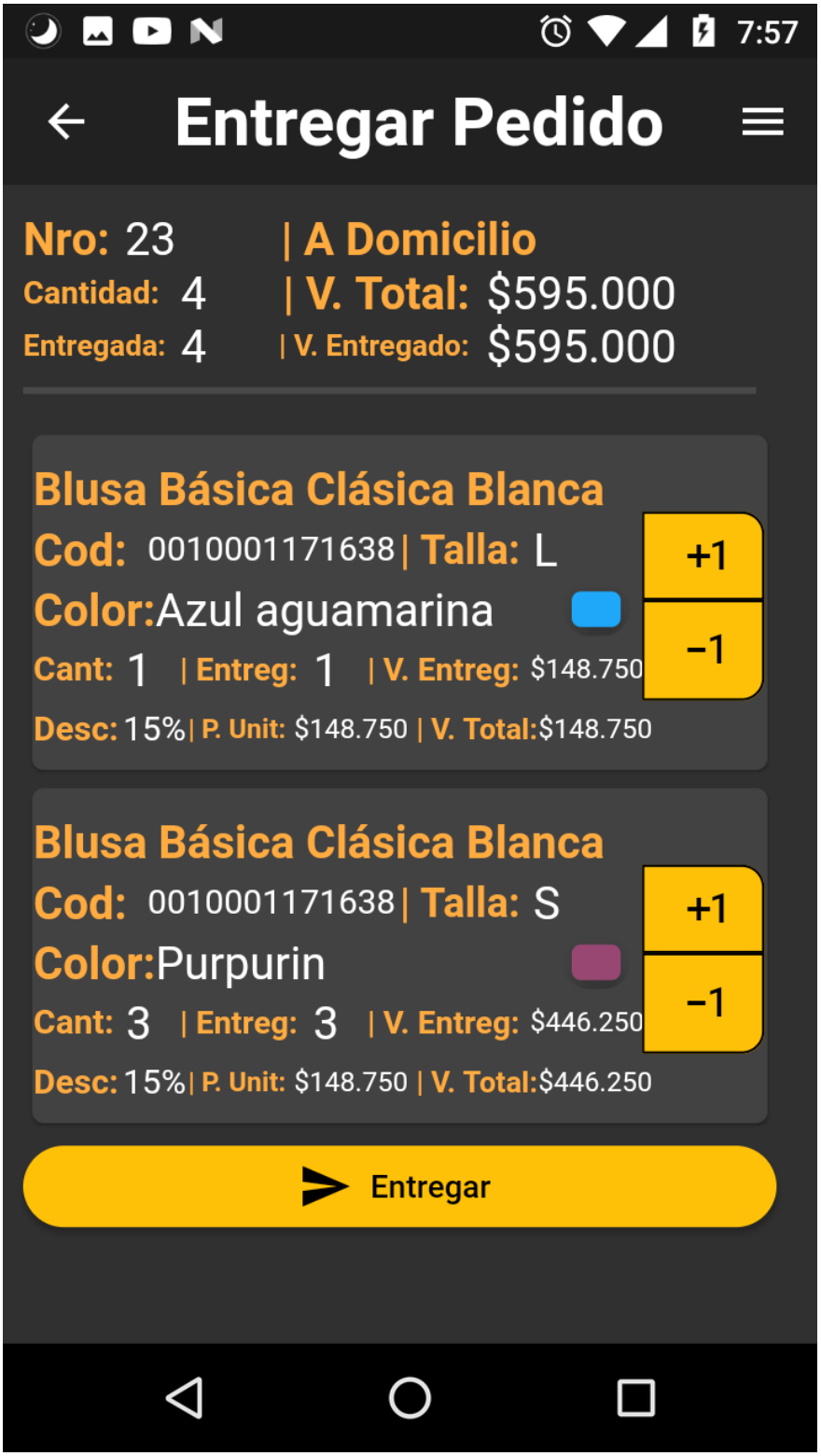


Figura 24. Vista de visualización factura para repartidor

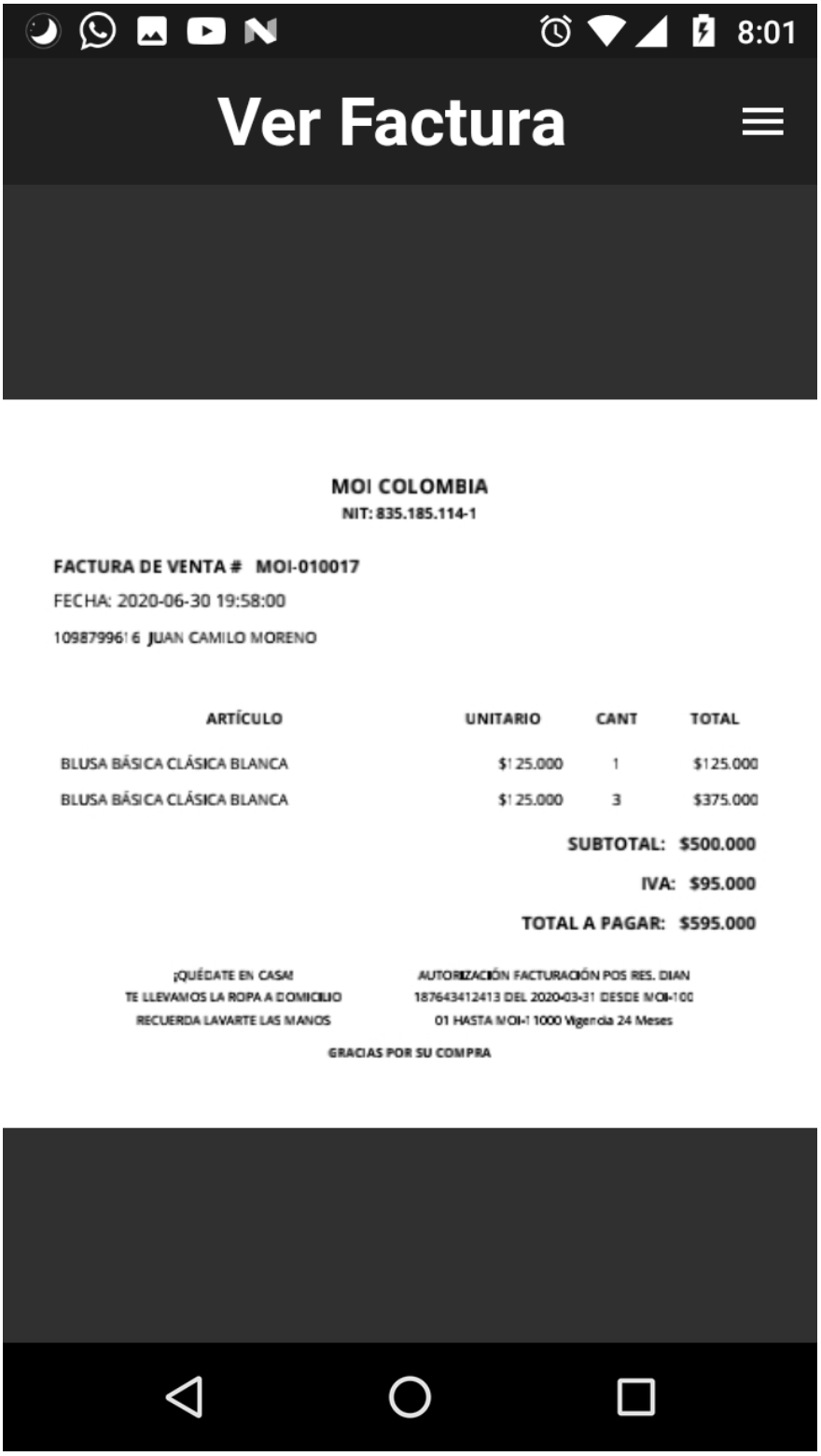


Figura 25. Vista de consulta de estadísticas, históricos de pedidos

Figura 26. Vista de importación

Figura 27. Vista de importación tablas

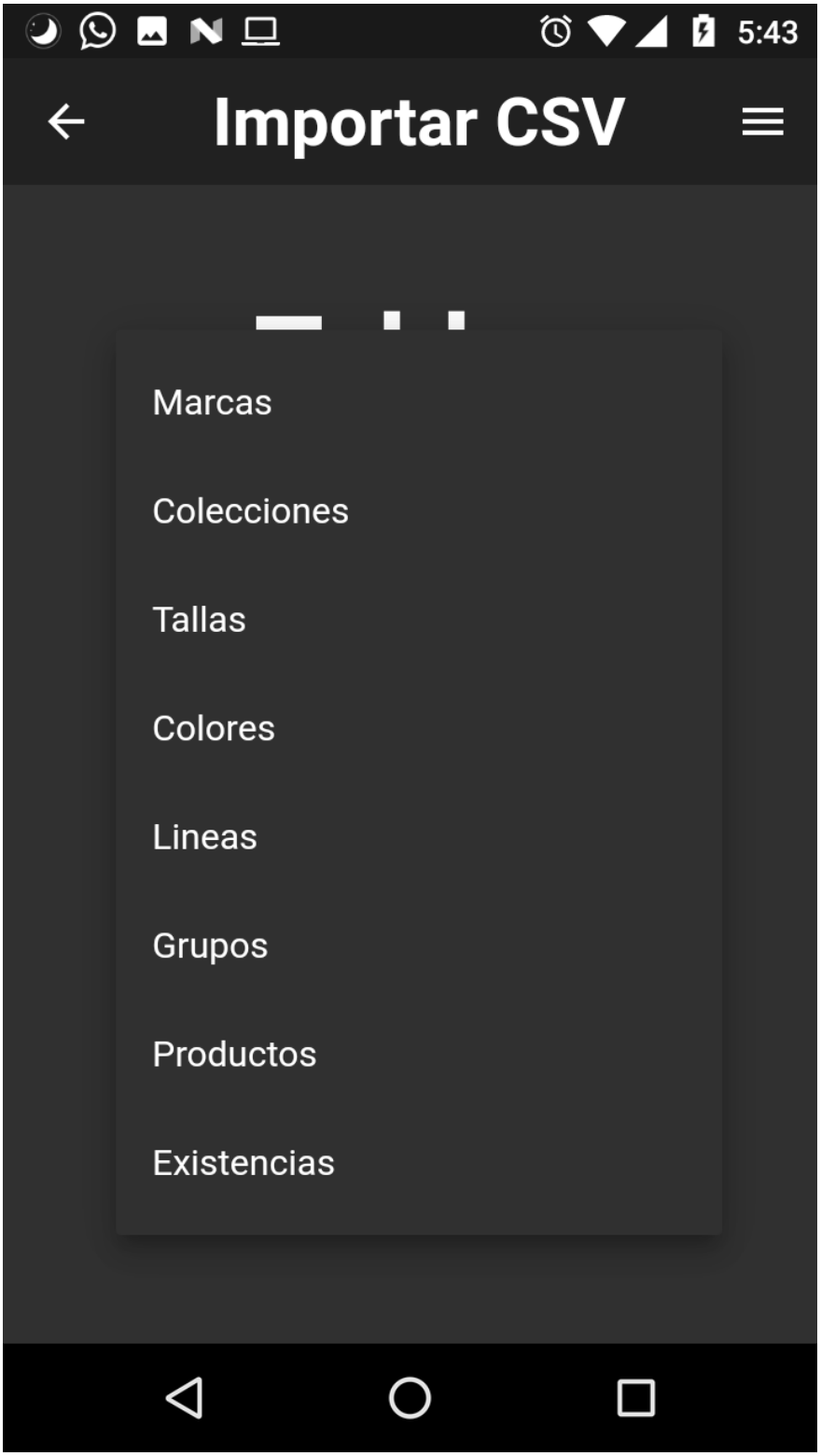
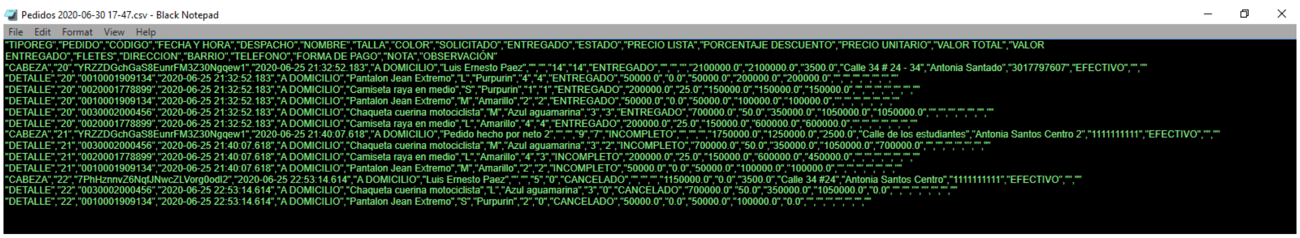


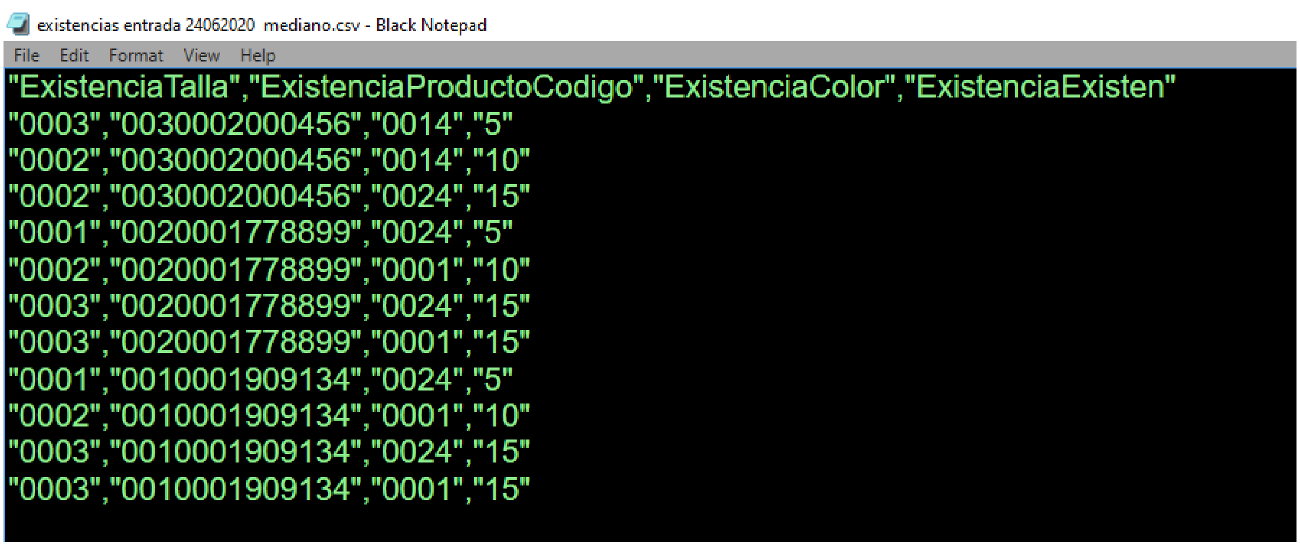
Figura 28. Vista de exportación de pedidos

Figura 29. Archivo ejemplo de exportación de pedidos




```
"TIPOREG","PEDIDO","CÓDIGO","FECHA Y HORA","DESPACHO","NOMBRE","TALLA","COLOR","SOLICITADO","ENTREGADO","ESTADO","PRECIO LISTA","PORCENTAJE DESCUENTO","PRECIO UNITARIO","VALOR TOTAL","VALOR
ENTREGADO","FLETES","DIRECCION","BARRIO","TELEFONO","FORMA DE PAGO","NOTA","OBSERVACIÓN"
"CABEZA","20","YRZZDGchGaS8EunrFM3Z30Ngqew1","2020-06-25 21:32:52.183","A DOMICILIO","Luis Ernesto Paez","","","14","14","ENTREGADO","","","","2100000.0","2100000.0","3500.0","Calle 34 # 24 - 34","Antonia Santado","3017797607","EFECTIVO","",""
"DETALLE","20","0010001909134","2020-06-25 21:32:52.183","A DOMICILIO","Pantalon Jean Extremo","L","Purpurin","4","4","ENTREGADO","50000.0","0.0","50000.0","200000.0","200000.0","","","","","","",""
"DETALLE","20","0020001778899","2020-06-25 21:32:52.183","A DOMICILIO","Camiseta raya en medio","S","Purpurin","1","1","ENTREGADO","200000.0","25.0","150000.0","150000.0","150000.0","","","","","","",""
"DETALLE","20","0010001909134","2020-06-25 21:32:52.183","A DOMICILIO","Pantalon Jean Extremo","M","Amarillo","2","2","ENTREGADO","50000.0","0.0","50000.0","100000.0","100000.0","","","","","","",""
"DETALLE","20","0030002000456","2020-06-25 21:32:52.183","A DOMICILIO","Chaqueta cuerina motociclista","M","Azul aguamarina","3","3","ENTREGADO","700000.0","50.0","350000.0","1050000.0","1050000.0","","","","","","",""
"DETALLE","20","0020001778899","2020-06-25 21:32:52.183","A DOMICILIO","Camiseta raya en medio","L","Amarillo","4","4","ENTREGADO","200000.0","25.0","150000.0","600000.0","600000.0","","","","","","",""
"CABEZA","21","YRZZDGchGaS8EunrFM3Z30Ngqew1","2020-06-25 21:40:07.618","A DOMICILIO","Pedido hecho por neto 2","","","9","7","INCOMPLETO","","","","1750000.0","1250000.0","2500.0","Calle de los estudiantes","Antonia Santos Centro 2","1111111111","EFECTIVO","",""
"DETALLE","21","0030002000456","2020-06-25 21:40:07.618","A DOMICILIO","Chaqueta cuerina motociclista","M","Azul aguamarina","3","2","INCOMPLETO","700000.0","50.0","350000.0","1050000.0","700000.0","","","","","","",""
"DETALLE","21","0020001778899","2020-06-25 21:40:07.618","A DOMICILIO","Camiseta raya en medio","L","Amarillo","4","3","INCOMPLETO","200000.0","25.0","150000.0","600000.0","450000.0","","","","","","",""
"DETALLE","21","0010001909134","2020-06-25 21:40:07.618","A DOMICILIO","Pantalon Jean Extremo","M","Amarillo","2","2","INCOMPLETO","50000.0","0.0","50000.0","100000.0","100000.0","","","","","","",""
"CABEZA","22","7PhHznnvZ6NqfJNwcZLVorg0odI2","2020-06-25 22:53:14.614","A DOMICILIO","Luis Ernesto Paez","","","5","0","CANCELADO","","","","1150000.0","0.0","3500.0","Calle 34 #24","Antonia Santos Centro","1111111111","EFECTIVO","",""
"DETALLE","22","0030002000456","2020-06-25 22:53:14.614","A DOMICILIO","Chaqueta cuerina motociclista","L","Azul aguamarina","3","0","CANCELADO","700000.0","50.0","350000.0","1050000.0","0.0","","","","","","",""
"DETALLE","22","0010001909134","2020-06-25 22:53:14.614","A DOMICILIO","Pantalon Jean Extremo","S","Purpurin","2","0","CANCELADO","50000.0","0.0","50000.0","100000.0","0.0","","","","","","",""
```

Figura 30. Archivo ejemplo de importación de existencias




```
"ExistenciaTalla","ExistenciaProductoCodigo","ExistenciaColor","ExistenciaExisten"
"0003","0030002000456","0014","5"
"0002","0030002000456","0014","10"
"0002","0030002000456","0024","15"
"0001","0020001778899","0024","5"
"0002","0020001778899","0001","10"
"0003","0020001778899","0024","15"
"0003","0020001778899","0001","15"
"0001","0010001909134","0024","5"
"0002","0010001909134","0001","10"
"0003","0010001909134","0024","15"
"0003","0010001909134","0001","15"
```

Figura 31. Vista de inicio de sesión Moi Admin

Figura 32. Vista de restablecer contraseña Moi Admin

Figura 33. Menú drawer de la app Moi Admin rol administrador

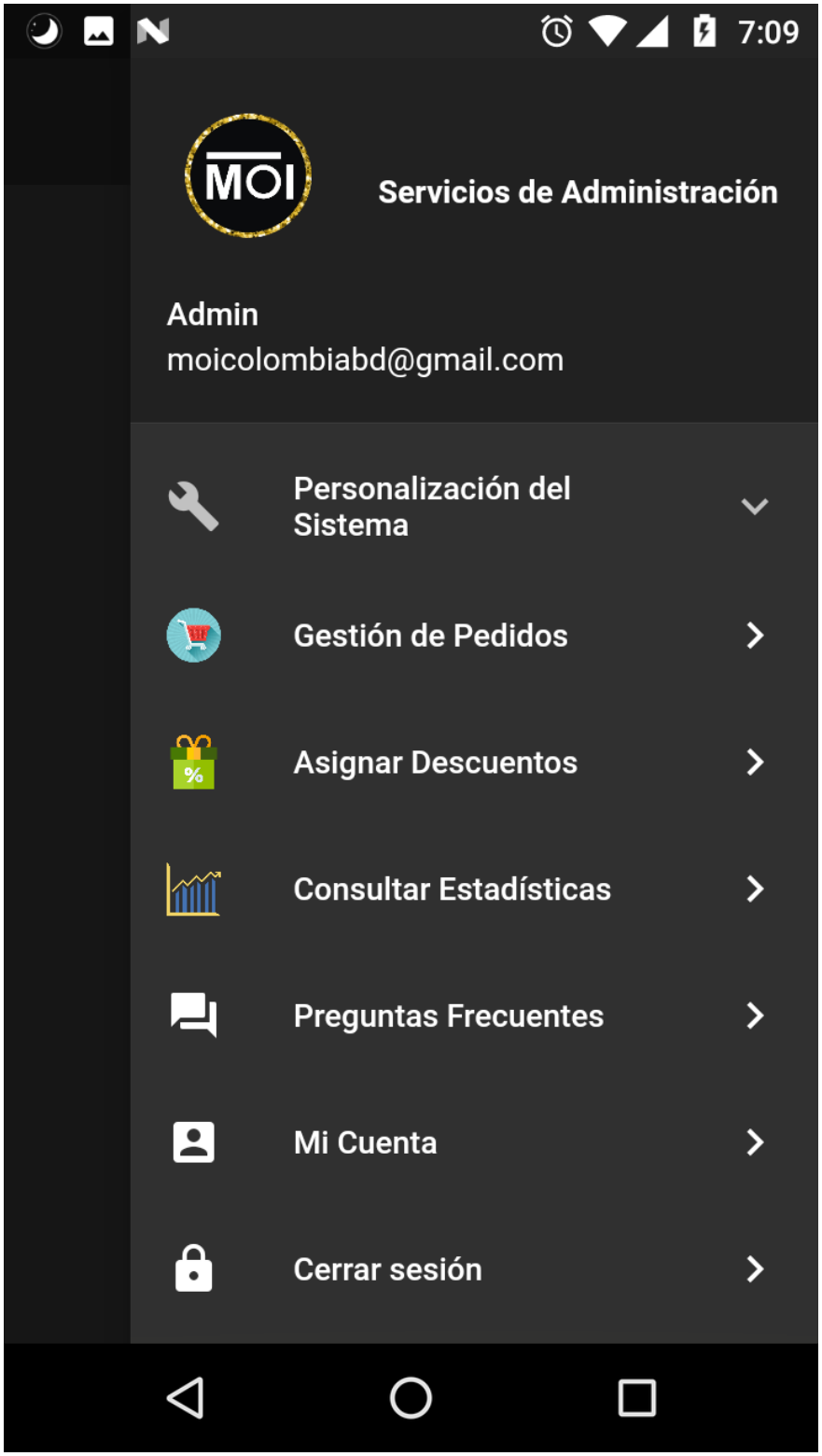


Figura 34. Menú drawer seleccionando personalización del sistema de la app Moi Admin rol administrador

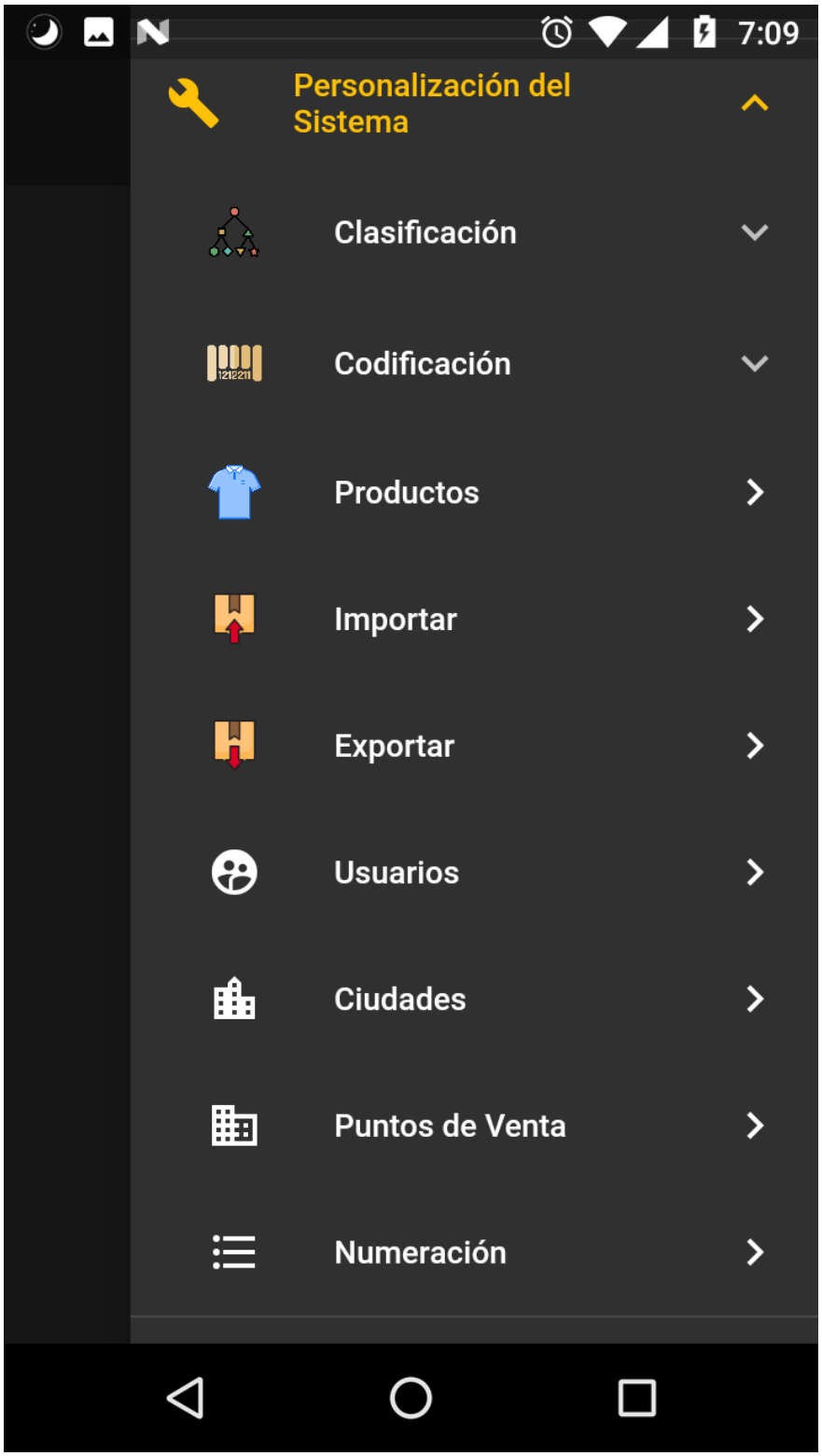
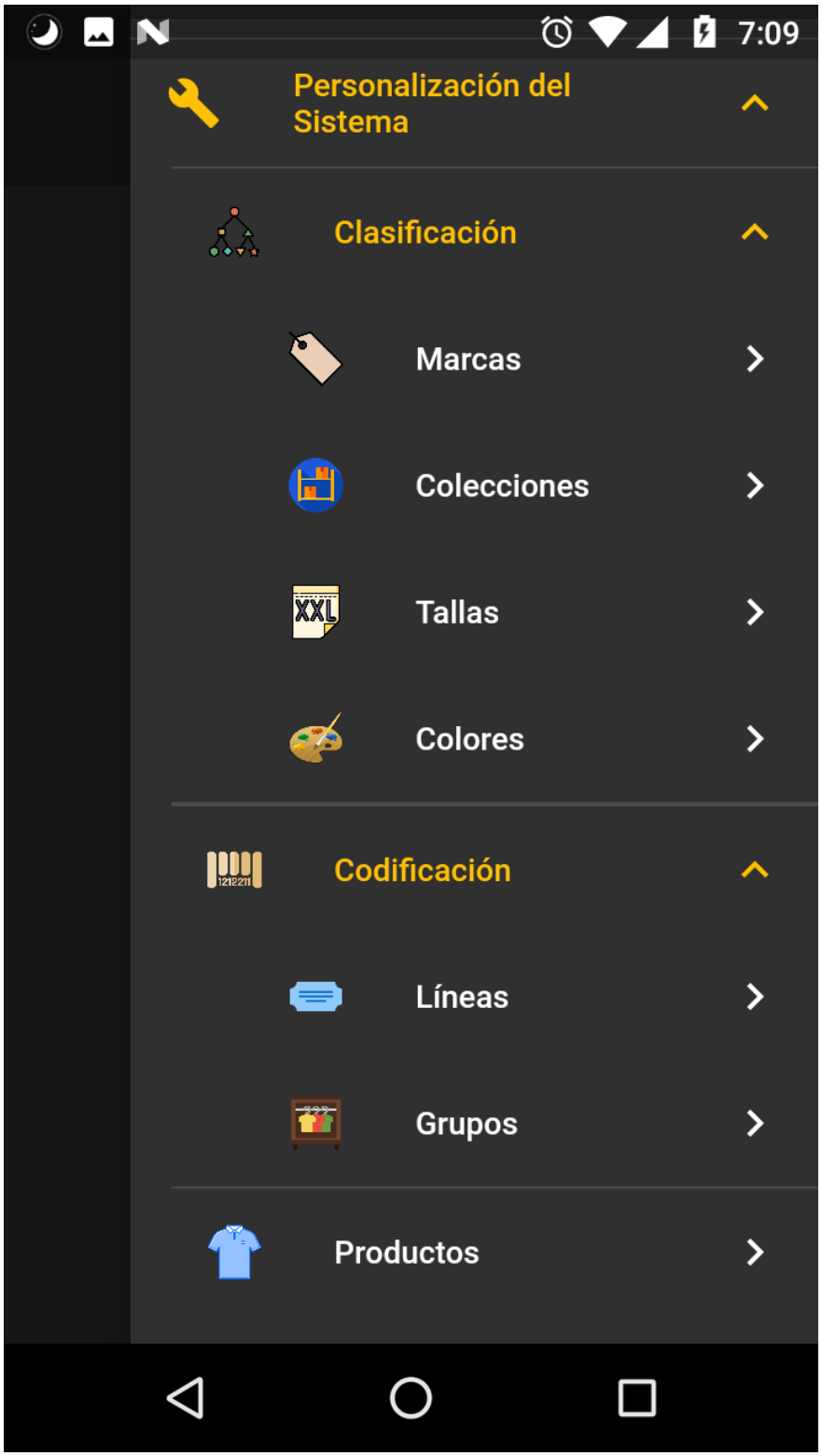


Figura 35. Menú drawer seleccionando codificación y clasificación de la app Moi Admin rol administrador



7. CONCLUSIONES Y PERSPECTIVAS

- Es poco viable realizar un planteamiento definitivo de la solución desde el comienzo, porque aunque el cliente interesado en el desarrollo de la solución tiene una idea de lo que necesita y desea, las reglas del negocio llevan a exigir cosas que llegan a chocar o complementar dicha idea inicial, y la transferencia tecnológica junto con la naturaleza de las herramientas y paradigma de desarrollo generan cambios en esta estructura que viene inmersa en la visión del cliente, por lo que el prototipado evolutivo y desarrollo incremental, con su realimentación, feedback y flexibilidad en términos adaptativos, permiten un proceso más adecuado de construcción de la solución de ingeniería.

- El paradigma de programación en Flutter con los Widgets, al ver los Widgets como un tipo de Objeto, tiene muchas similitudes con el paradigma de POO (Programación Orientada a Objetos) que se puede encontrar en Java, por lo cuál muchas consideraciones al utilizar este paradigma son útiles y aplicables en el desarrollo con este framework.

- La programación asíncrona y la reactiva son inherentemente dirigidas por eventos, por lo cual su integración con el paradigma de programación de Flutter y la programación síncrona en algunos casos llega a ser contraintuitiva y los errores al programar pueden ser difíciles de notar. Se recomienda llevar una muy buena trazabilidad del flujo de la aplicación y prestar especial atención y cuidado al usar métodos o peticiones asíncronas en Flutter.

- El manejo de estados suele ser una de las principales ventajas de utilizar Flutter como framework, al poder redibujar únicamente las partes de cada vista que hayan tenido cambios, en vez de redibujar por completo la vista, pero debido a que hay gran cantidad de aproximaciones a esta tarea puede ser complejo definir un manejo adecuado al contexto. Al plantear la solución software particular, es muy importante buscar la forma más adecuada

de manejar el estado de cada parte de la aplicación para procurar el buen rendimiento y la escalabilidad, aprovechando las ventajas que ofrece el framework.

- Flutter está diseñado para tener una fácil y adecuada integración con los productos de Google, tales como Google Cloud Functions, Google Cloud Storage y Firebase. Por este motivo, cuando un requerimiento surge para la solución que se desarrolla con Flutter, es muy recomendable explorar la posibilidad de adquirir e integrar un servicio de Google que cuente con la funcionalidad para satisfacer dicho requerimiento.

- Firebase Realtime Database no permite realizar consultas que filtren por más de un valor o atributo. Para realizar consultas que tengan que filtrar por varios valores, no hay una forma de hacerlo directamente en la consulta, y el manejo más adecuado en términos de rendimiento al cual se llegó en este proyecto es a filtrar sobre el valor que implique mayor carga en términos computacionales en la consulta, y luego filtrar por el resto de valores en la aplicación. Esto supone una carga extra del lado cliente del programa, por lo cual disminuye el rendimiento enormemente. Se recomienda aplicar este enfoque en la medida de la posible al utilizar Realtime Database, o utilizar desde un principio Cloud Firestore que no cuenta con esta limitación, permitiendo realizar consultas compuestas, dejando la computación de los filtros al servidor.

- Se valida los beneficios del uso de la metodología de desarrollo Ágil Scrum para la validación de requerimientos e iteraciones evolutivas por parte del cliente. El método SCRUM, en particular, facilitó la eficiencia y efectividad del desarrollo de la interfaz como de la validación de requerimientos.

8. RECOMENDACIONES Y TRABAJO FUTURO

- Debido a la incapacidad de la base de datos Firebase Realtime Database de realizar consultas con más de un filtro, se recomienda para futuros desarrollos realizar una migración evolutiva a Cloud Firestore de Firebase, con el fin de realizar una migración prudente, manteniendo la persistencia de los datos que garantiza el uso del SDK de Firebase. Sin embargo, para futuros desarrollos es posible utilizar otro tipo de base de datos según convenga, si así se considera pertinente.

- Realizar una investigación rigurosa acerca de las posibilidades de pagos dentro de la aplicación. A la fecha no se cuenta con SDK estables para la implementación de pagos en línea. Se recomienda revisar mercadopago sdk[50] en la tienda de paquetes de Flutter, puesto que es un paquete funcional pero no desarrollado por MercadoPagos para la realización de pagos dentro de la aplicación.

- Implementar una versión beta para verificar el funcionamiento del código fuente en su versión web. Para ello, se recomienda revisar cada uno de los paquetes instalados en la ruta principal del proyecto en el archivo pubspec.yaml, verificando la disponibilidad de estos para una versión web. De no ser posible, investigar paquetes alternos o la viabilidad para una implementación que sustente la necesidad perdida para el uso web.

- Adecuar la facturación desarrollada para que cumpla con los lineamientos de la DIAN y pueda ser legalizada como facturación electrónica, o bien explorar la posibilidad de contratación con un proveedor autorizado de facturación electrónica.

- Inclusión de bonos y promociones relámpago, con la construcción de una metodología y módulo de fidelización adecuado para la promoción del negocio, con promociones focali-

---

[50] HOSTELIXISRAEL. *mercadopago_ sdk*. Ver. 1.1.3. 29 de jun. de 2020.

zadas por clientes que hayan hecho cierta cantidad de pedidos o hayan hecho pedidos por cierto valor.

- Exploración de la viabilidad técnica e implementación de un chat dentro de la aplicación para la comunicación en cualquier momento de la transacción por parte del cliente.
- Eliminación o desactivación de clientes automáticamente después de cierto lapso inactivo.
- Envío de promociones y campañas de mailingship de forma masiva.
- Distribución y manejo de disponibilidad ligado a puntos de venta.

BIBLIOGRAFÍA

AGUDELO MONTOYA, César Alveiro y SAAVEDRA BOYERO, Martín Ramiro. "El CRM como herramienta para el servicio al cliente en la organización". En: *Visión de futuro* 17.1 (2013), pág. 5 (vid. pág. 15).

BIGCOMMERCE. *What is a .CSV file and what does it mean for my ecommerce business?* 2020. URL: `https://www.bigcommerce.com/ecommerce-answers/what-csv-file-and-what-does-it-mean-my-ecommerce-business/` (visitado 26-06-2020) (vid. págs. 18, 45).

BOLTON, David. *Definition of Encapsulation in Computer Programming*. 2020. URL: `https://www.thoughtco.com/definition-of-encapsulation-958068` (visitado 28-06-2020) (vid. pág. 24).

CÁMARA DE COMERCIO DE BOGOTÁ, et al. "Ley 905 de 2004". En: (2004) (vid. pág. 20).

CHTIOUI, Mahdi. *ReactiveX: Reactive Programming Principles*. 2020. URL: `https://medium.com/@mahdichtioui/reactivex-reactive-programming-principles-dbb1bafa8384` (visitado 28-06-2020) (vid. pág. 23).

DAGNE, Lukas. "Flutter for cross-platform App and SDK development". En: (2019) (vid. pág. 15).

DART TEAM. *http*. Ver. 0.12.1. 27 de abr. de 2020 (vid. pág. 42).

— *intl*. Ver. 1.6.0. 6 de ene. de 2020 (vid. pág. 47).

DEY, Anik. *Design Patterns — A quick guide to Observer pattern.* 2020. URL: https://medium.com/@info.anikdey003/factory-method-design-pattern-277dd4bd3a11 (visitado 28-06-2020) (vid. pág. 25).

FALABELLA RETAIL. *Falabella.* Ver. 1.11.10. 9 de jun. de 2020 (vid. pág. 29).

FLUTTER TEAM. *Flutter Widget Expanded.* 2020. URL: https://api.flutter.dev/flutter/widgets/Expanded-class.html (visitado 25-06-2020) (vid. pág. 44).

— *Flutter Widget MediaQuery.* 2020. URL: https://api.flutter.dev/flutter/widgets/MediaQuery-class.html (visitado 25-06-2020) (vid. pág. 44).

— *image_picker.* Ver. 0.6.7+2. 15 de ene. de 2020 (vid. pág. 42).

— *shared_preferences.* Ver. 0.5.7+3. 21 de mayo de 2020 (vid. pág. 40).

FOREVER 21. *Forever 21.* Ver. 3.4.5.221. 18 de mayo de 2020 (vid. pág. 27).

GEEKS FOR GEEKS. *Software Engineering | Prototyping Model.* 2019. URL: https://www.geeksforgeeks.org/software-engineering-prototyping-model/ (visitado 21-03-2019) (vid. págs. 22, 23).

GIBBONS, Jeremy y DOS SANTOS OLIVEIRA, Bruno César. "The essence of the iterator pattern". En: *Journal of functional programming* 19.3and4 (2009) (vid. pág. 25).

GOOGLE LLC. *Estructura tu base de datos Realtime Database.* 2020. URL: https://firebase.google.com/docs/database/web/structure-data (visitado 15-06-2020) (vid. pág. 51).

— *Firebase.* 2019. URL: https://firebase.google.com/ (visitado 14-06-2020) (vid. pág. 21).

GOOGLE LLC. *Firebase Authentication*. 2020. URL: https://firebase.google.com/docs/auth (visitado 25-06-2020) (vid. págs. 21, 40).

— *Firebase Cloud Storage*. 2020. URL: https://firebase.google.com/docs/storage (visitado 25-06-2020) (vid. pág. 43).

— *Firebase Realtime Database*. 2020. URL: https://firebase.google.com/docs/database (visitado 14-06-2020) (vid. pág. 21).

— *Flutter DateTime class*. 2020. URL: https://api.flutter.dev/flutter/dart-core/DateTime-class.html (visitado 26-06-2020) (vid. pág. 43).

— *Flutter SDK*. 2019. URL: https://flutter-es.io/ (visitado 20-06-2020) (vid. pág. 20).

HOSTELIXISRAEL. *mercadopago_sdk*. Ver. 1.1.3. 29 de jun. de 2020 (vid. pág. 110).

HTML COLOR CODES INFO. *Teoría sobre los códigos de colores HTML*. 2020. URL: https://html-color-codes.info/codigos-de-colores-hexadecimales/ (visitado 15-06-2020) (vid. pág. 18).

LEIER, Simon. *auto_size_text*. Ver. 0.6.7+2. 13 de ago. de 2019 (vid. pág. 44).

LEONARDIS, Dan. *Reactive Architecture*. 2020. URL: https://android.jlelse.eu/reactive-architecture-7baa4ec651c4?gi=182b1cd478e7 (visitado 28-06-2020) (vid. pág. 22).

LOCAL DROID. *Widget Communication in Flutter using Provider*. 2020. URL: https://www.local-droid.com/post/provider-example/ (visitado 28-06-2020) (vid. pág. 24).

LOITSCH, Christian. *csv*. Ver. 4.0.3. 5 de abr. de 2019 (vid. pág. 45).

MESÍAS TAVERA, Juan F; GIRALDO SÁNCHEZ, Juan C y DÍAZ BALLESTEROS, Bernardo. "Aceptación del e-commerce en Colombia: un estudio para la ciudad de Medellín". En: *Revista Facultad de Ciencias Económicas: Investigación y Reflexión* 19.2 (2011), págs. 9-23 (vid. pág. 14).

MORTON, J y ODELL, JJ. *Object oriented analysis and design*. Englewood Cliffs (New Jersey): Prentice-Hall, 1992 (vid. pág. 24).

NFET.NET. *pdf*. Ver. 1.6.0. 28 de mar. de 2020 (vid. pág. 46).

NIETO, Victor, et al. "La clasificación por tamaño empresarial en Colombia: Historia y limitaciones para una propuesta". En: *Archivos de economía* 434 (2015) (vid. pág. 20).

NIRANJANAMURTHY, M, et al. "Analysis of e-commerce and m-commerce: advantages, limitations and security issues". En: *International Journal of Advanced Research in Computer and Communication Engineering* 2.6 (2013), págs. 2360-2370 (vid. págs. 14, 19).

PASH S.A.S. *Patprimo*. Ver. 1.2.0. 9 de mayo de 2020 (vid. pág. 27).

— *Seven Seven*. Ver. 1.2.0. 12 de nov. de 2018 (vid. pág. 26).

POYIAS, Andreas. *Design Patterns — A quick guide to Observer pattern*. 2020. URL: `https://medium.com/datadriveninvestor/design-patterns-a-quick-guide-to-observer-pattern-d0622145d6c2` (visitado 28-06-2020) (vid. pág. 25).

REVISTA DINERO. *¿Comprar por internet? Los colombianos aún no se deciden*. 2016. URL: `https://www.dinero.com/pais/articulo/como-va-el-comercio-electronico-en-colombia/232305` (visitado 17-03-2019) (vid. pág. 14).

SHKURKO, Serge. *native_pdf _view*. Ver. 3.6.2. 31 de mayo de 2020 (vid. pág. 48).

STOLL, Scott. *Flutter Tutorial: Provider Overview for Humans*. 2020. URL: https://blog.codemagic.io/flutter-tutorial-provider/ (visitado 28-06-2020) (vid. pág. 24).

THE ECONOMIST. *E-commerce*. 2009. URL: https://www.economist.com/news/2009/10/08/e-commerce (visitado 20-03-2019) (vid. pág. 19).

THOMPSON, Arthur. *firebase_ auth*. Ver. 0.16.1. 25 de mayo de 2020 (vid. pág. 40).

— *firebase_ database*. Ver. 3.1.6. 25 de mayo de 2020 (vid. pág. 37).

— *firebase_ storage*. Ver. 0.16.1. 25 de mayo de 2020 (vid. págs. 22, 43).

UNIÓN EUROPEA. "Recomendación de la Comisión, del 6 de mayo de 2003, sobre la definición de microempresas, pequeñas y medianas empresas". En: *Diario Oficial de la Unión Europea L* 124 (2003), pág. 20 (vid. pág. 20).

UNIX TUTORIAL. *Unix Epoch*. 2020. URL: https://www.unixtutorial.org/unix-epoch/ (visitado 26-06-2020) (vid. págs. 19, 43).

VANGUARDIA LIBERAL. *Comerciantes de Santander dan paso hacia el E-commerce*. 2017. URL: http://www.vanguardia.com/economia/local/410132-comerciantes-desantander-dan-paso-hacia-el-e-commerce (visitado 17-03-2019) (vid. pág. 14).

VELASQUEZ LOPEZ, Diego. *keyboard_ actions*. Ver. 3.2.1+1. 20 de mayo de 2020 (vid. pág. 49).