



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLÁHUAC

“LA ESENCIA DE LA GRANDEZA RADICA EN LAS RAÍCES”

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

Ingeniería en sistemas computacionales

Informe técnico de
Residencias profesionales

“Diseño e implementación de un sistema de ventas online para microempresas, bajo plataformas móviles”

P R E S E N T A. (N)

JIMÉNEZ CERVANTES JOSÉ ANDRÉS

ASESOR INTERNO
ING. FABIAN DE JESÚS JOSÉ

REVISOR
ING. ABEL NUÑEZ AYALA



Tláhuac, CDMX., AGOSTO-2021

Agradecimientos

“Soy lo que soy, por la gracia de Dios” 1 Co 15, 10

Quiero agradecerle a Dios por todas las bendiciones que día a día derrama sobre mí y mis seres queridos, en especial por acompañarme en toda mi travesía académica.

De manera especial quiero agradecer a mis abuelos, ellos son quienes me han educado en la fe y en los valores humanos, gracias por estar siempre ahí para mí, ustedes son mi motor para superarme y ser mejor persona cada día. Con ustedes he pasado los mejores momentos de mi vida.

Gracias a mis tíos y tías, pues siempre me han apoyado y han sido un soporte para mí en los momentos difíciles, gracias por motivarme y ayudarme cuando lo necesito.

A mis amigos, en especial a Sadhi, Jonny, Ángel y Luis, con quienes compartí esta trayectoria académica y formamos el mejor equipo de todos. ¡Gracias por su apoyo!

También quiero agradecer a mis mentores y asesores de proyecto, pues con su guía y conocimiento me han ayudado a formarme como un profesional.

Resumen

Las aplicaciones móviles o apps son productos de software que pueden ejecutarse en dispositivos móviles. Estas aplicaciones han dejado de ser simples y esto es porque cada vez nacen proyectos mucho más robustos, pues surgen de la idea de “*Mobile first*” o primero móvil, esté es un concepto que se refiere a la comodidad que brinda el uso de aplicaciones en teléfonos inteligentes, pues existe una gran variedad de aplicativos que facilitan la vida de las personas, hay aplicaciones para la banca, de entretenimiento, educación y hasta para realizar comercio electrónico.

Sin embargo, para el desarrollo de una aplicación móvil se debe elegir adecuadamente las tecnologías y metodología a implementar para lograr el desarrollo del producto. Debido a que en la actualidad los usuarios exigen que las aplicaciones sean: rápidas, fluidas, seguras y mantenibles.

Es por ello que en el presente proyecto denominado “Diseño e implementación de un sistema de ventas online para microempresas, bajo plataformas móviles”, se decidió utilizar las siguientes tecnologías: el SDK de flutter, una base de datos relacional con el sistema gestor de base de datos MySQL, NodeJS para el desarrollo de la API REST, la API de Firebase para integrar notificaciones y se implementó la API de Mercado pago para permitir los pagos en línea.

La lógica de negocio de este producto de software permite la compra y venta de productos en línea por parte de las microempresas con la finalidad de que estas puedan tener un crecimiento económico, logrando llegar a más clientes y aumentando su volumen de ventas.

Abstract

Mobile applications or apps are software products that can run on mobile devices. These applications are no longer simple and this is because much more robust projects are being born every time, since they arise from the idea of "Mobile first", this is a concept that refers to the comfort offered by the use of applications in Smartphones, as there is a wide variety of applications that facilitate people's lives, there are applications for banking, entertainment, education and even for electronic commerce.

However, for the development of a mobile application, the technologies and methodology to be implemented must be properly chosen to achieve the development of the product. Because today users demand that applications be: fast, fluid, secure and maintainable.

That is why in this project called "Design and implementation of an online sales system for micro-businesses, under mobile platforms", it was decided to use the following technologies: the flutter SDK, a relational database with the base management system of MySQL data, NodeJS for the development of the REST API, the Firebase API for integrate notifications and the Mercado Pago API was implemented to allow online payments.

The business logic of this software product allows the purchase and sale of products online by micro-companies in order for them to have economic growth, reaching more customers and increasing their sales volume.

Contenido

Tabla de contenido

Agradecimientos.....	2
Resumen	3
Abstract	4
Capítulo 1 Introducción	11
Capítulo 2 Generalidades.....	12
2.1. Objetivos	12
2.1.1. Objetivo general.....	12
2.1.2. Objetivos específicos.....	13
2.2. Justificación.....	13
2.3. Caracterización de la empresa en la que participó	14
2.3.2. Descripción del departamento o área de trabajo	16
2.4. Problemas a resolver (priorizándolos)	16
2.5. Alcances y limitaciones	17
2.6 Estudio de factibilidad y viabilidad.....	18
2.7 Factibilidad operacional.....	19
2.8 Estudio de viabilidad.....	19
Capítulo 3 Fundamento teórico	20
3.1 Microempresas y comercio electrónico	20
3.2 Aplicación móvil.....	21
3.3 Sistemas operativos móviles.....	21
3.4 Android	22
3.5 Metodología Scrum	24
3.6 Lenguajes de programación y Frameworks	26
3.7 Flutter y Dart.....	26
3.8 Arquitectura de diseño	27
3.9 Arquitectura Bloc en Flutter	27
3.10 MySQL.....	28

3.10.1 Niveles de abstracción de una base de datos	29
3.11 Modelo Entidad-relación	30
3.11.1 Normalización	31
3.12 Sistema de gestión de base de datos MySQL.....	32
3.13 Servicio web.....	33
3.13.1 API REST.....	33
Capítulo 4 Desarrollo	35
4.1 Análisis.....	35
4.1.1 Entrevista con la empresa “Codeway Soluciones Integrales”	35
4.1.2 Requerimientos funcionales del sistema	37
4.2 Diseño.....	39
4.2.1 Diagrama entidad-relación de la base de datos	39
4.2.2 Diagrama relacional de la base de datos.....	40
4.2.3 Diagrama de estados	42
4.2.4 Diagramas de interacción	44
4.2.5 <i>Diagrama de componentes</i>	46
4.2.6 Diagrama de despliegue.....	47
4.2.7 Diagramas de casos de uso	48
4.2.8 Diseño de las interfaces de usuario	59
4.3 Implementación de la base de datos	72
4.4 Servicio web (API REST)	95
Capítulo 5 Pruebas y resultados	99
5.1 Implementación de los diseños previamente realizados.	99
5.2 Implementación de la base de datos	116
5.3 Implementación del servicio web	117
5.4 Pruebas a la aplicación móvil	118
Capítulo 6 Conclusiones, recomendaciones y experiencia profesional adquirida	140
6.2 Competencias desarrolladas y/o adquiridas.	141
6.2.1 Competencias instrumentales	141
6.2.2 Competencias interpersonales	141
6.2.3 Competencias sistémicas	141
Capítulo 7 Bibliografía	142

Lista de Figuras

Figura 2.1 Croquis de la ubicación de la empresa "Codeway Soluciones Integrales"	14
Figura 2.2 Organigrama general. Codeway Soluciones Integrales	15
Figura 3.1 Diagrama de Scrum	24
Figura 3.2 Logo de Flutter y Dart	26
Figura 3.3 Funcionamiento del protocolo HTTP	34
Figura 4.1 Diagrama entidad-relación.....	39
Figura 4.2 Diagrama relacional de la base de datos	41
Figura 4.3 Estado del usuario	42
Figura 4.4 Diagrama de estado del producto.	42
Figura 4.5 Estados del pago	43
Figura 4.6 Interacción entre el objeto usuario y producto	44
Figura 4.7 Interacción del flujo de compra	45
Figura 4.8 Diagrama de componentes de la tienda virtual	46
Figura 4.9 Diagrama de despliegue de la tienda virtual.....	47
Figura 4.10 Caso de uso "Iniciar sesión"	48
Figura 4.11 Caso de uso "Crear cuenta"	49
Figura 4.12 Caso de uso "Ver productos en venta"	50
Figura 4.13 Caso de uso "Aregar productos al carrito"	51
Figura 4.14 Caso de uso "Eliminar productos del carrito"	52
Figura 4.15 Caso de uso "Realizar compra".....	53
Figura 4.16 Caso de uso "Actualizar producto"	55
Figura 4.17 Caso de uso "Eliminar producto"	56
Figura 4.18 Caso de uso "Comentar productos"	57
Figura 4.19 Caso de uso "Cerrar sesión"	58
Figura 4.20 Diseño de la pantalla de Inicio.....	59
Figura 4.21 Diseño del Menú hamburguesa	60
Figura 4.22 Diseño de Pantalla de búsqueda	61
Figura 4.23 Diseño de Pantalla de detalle del producto	62
Figura 4.24 Diseño de Pantalla de carrito de compras	63
Figura 4.25 Diseño de Pantalla de Inicio de sesión.....	64
Figura 4.26 Diseño de Pantalla de registro de usuario	65
Figura 4.27 Diseño de Flujo de pantallas para realizar una compra.....	66
Figura 4.28 Diseño de Pantalla de "Mis compras"	67
Figura 4.29 Diseño de Pantalla de perfil	68
Figura 4.30 Diseño de Pantalla de productos en venta	69
Figura 4.31 Diseño de Pantalla de "Medios de pago"	70
Figura 4.32 Diseño de Menú y formulario de subir producto	71
Figura 4.33 Integración de rutas de la API REST	97
Figura 4.34 Conexión de la API REST con la base de datos	98
Figura 5.1 Implementación de la interfaz de inicio de sesión	99

Figura 5.2 Implementación de la interfaz "Inicio" cuando el usuario no está logeado.....	100
Figura 5.3 Implementación de la interfaz "Crear cuenta"	101
Figura 5.4 Interfaz de "Perfil".	102
Figura 5.5 Interfaz de "Inicio" cuando el usuario ya está logeado.	103
Figura 5.6 Interfaz "Carrito de compras"	104
Figura 5.7 Interfaz 1 del flujo de pago	105
Figura 5.8 Interfaz 2 del proceso de pago.....	106
Figura 5.9 Interfaz 3 del proceso de pago	107
Figura 5.10 Interfaz 4 del proceso de pago	108
Figura 5.11 Interfaz 5 del proceso de pago	109
Figura 5.12 Interfaz 6 del proceso de pago.....	110
Figura 5.13 Interfaz 7 del proceso de pago	111
Figura 5.14 Interfaz 8 del proceso de pago	112
Figura 5.15 Interfaz "Datos de mercado pago"	113
Figura 5.16 Interfaz "Subir producto"	114
Figura 5.17 Interfaz "Productos en venta"	115
Figura 5.18 Base de datos Storecode	116
Figura 5.19 Servicio web en ejecución	117
Figura 5.20 Aplicación móvil "Storecode" instalada en un dispositivo físico.	118
Figura 5.21 Pantalla "Perfil" de usuario no logeado	119
Figura 5.22 Pantalla "Crear cuenta"	120
Figura 5.23 Respuesta del servicio web al crear una cuenta	121
Figura 5.24 Gmail para la activación de la cuenta	122
Figura 5.25 Pantalla de "Login".....	123
Figura 5.26 Pantalla de "Perfil".....	124
Figura 5.27 Pantalla modal para solicitar permiso de acceder a la cámara.....	125
Figura 5.28 Galería del dispositivo	126
Figura 5.29 Pantalla "Subir producto".	127
Figura 5.30 Pantalla "Productos en venta".	128
Figura 5.31 Pantalla de "Inicio" cuando el usuario esta logeado.	129
Figura 5.32 Pantalla "Detalle del producto".....	130
Figura 5.33 Pantalla "Carrito de compras".....	131
Figura 5.34 Primera pantalla del flujo se pago	132
Figura 5.35 Tipo de tarjeta.	133
Figura 5.36 Datos de la tarjeta.....	134
Figura 5.37 Resumen de compra.	135
Figura 5.38 Pago acreditado.	136
Figura 5.39 Termina el flujo pago.	137
Figura 5.40 Barra de notificaciones	138
Figura 5.41 Actividad reflejada en la cuenta de mercado pago.	139

Lista de tablas

Tabla 2-1 Estudio de factibilidad.....	18
Tabla 3-1 Versiones de Android (Adeva, 2021).....	23
Tabla 4-1 Descripción textual del caso de uso "Iniciar sesión"	48
Tabla 4-2 Descripción textual del caso de uso "Crear cuenta".....	49
Tabla 4-3 Descripción textual del caso de uso "Ver productos en venta"	50
Tabla 4-4 Descripción textual del caso de uso "Agregar productos al carrito"	51
Tabla 4-5 Descripción textual del caso de uso "Eliminar productos del carrito"	52
Tabla 4-6 Descripción textual del caso de uso "Realizar compra".....	54
Tabla 4-7 Descripción textual del caso de uso "Actualizar producto"	55
Tabla 4-8 Descripción textual del caso de uso "Eliminar producto"	56
Tabla 4-9 Descripción textual del caso de uso "Comentar producto"	57
Tabla 4-10 Descripción textual del caso de uso "Cerrar sesión"	58
Tabla 4-11 Script de la base de datos.....	72
Tabla 4-12 Endpoints de la API REST	96
Tabla 5-1 Características del dispositivo de pruebas	118

Lista de acrónimos

Bps	bits por segundo.
SDK	software developer kit
IDE	Integrated development environment
API	Aplication programming interface
LMD	Lenguaje de manipulación de datos
LDL	Lenguaje de definición de datos
FN	Forma normal
Json	JavaScript Object Notation

Capítulo 1

Introducción

En la actualidad, la tecnología móvil está presente en todas partes y es utilizada principalmente para comunicarse. Esta tecnología se refiere a toda aquella infraestructura que permite que exista conectividad entre los diferentes dispositivos móviles, algunos ejemplos de esta tecnología son: antenas de telefonía celular, los teléfonos inteligentes, bluetooth, las aplicaciones móviles, etc. Cuando se habla de dispositivos móviles, no solo se hace referencia a un teléfono celular, sino a todos aquellos dispositivos que son pequeños, se pueden sostener con una sola mano, tienen conexión a internet y cuentan con una batería para poder utilizarlos sin necesidad de cables.

Aprovechando los beneficios que ofrece la tecnología móvil y los dispositivos móviles, especialmente el teléfono celular, se decidió enfocar el presente proyecto en desarrollar una tienda virtual, bajo plataformas móviles, que les permita a las microempresas realizar la compra y venta de sus productos por medio de internet, para que estás puedan llegar a más clientes y así aumentar el volumen de sus ventas.

Debido a que actualmente muchas microempresas no cuentan con un canal adecuado para vender sus productos, pues dependen únicamente de un local físico para comercializar, esto ocasiona que estas pequeñas empresas tengan clientes limitados dependiendo de su zona geográfica, lo cual a su vez genera pocas ventas y afecta la rentabilidad de las microempresas.

El presente proyecto fue desarrollado en la empresa “Codeway Soluciones Integrales” y las tecnologías implementadas para el desarrollo del proyecto son: el SDK de flutter con el lenguaje de programación Dart, el IDE de desarrollo Android Studio, NodeJS para la programación del lado del servidor, servicios y API'S de terceros como: Firebase, Hostinger y mercado pago. La metodología implementada para el desarrollo del aplicativo móvil fue Scrum, pues es una metodología ágil y se adaptó bien a las necesidades del proyecto.

Este informe técnico está organizado de la siguiente manera: en el capítulo 2 se describen las generalidades, el objetivo general y específicos, así como también se colocan datos importantes de la empresa, en el capítulo 3 se plasman temas teóricos que sustentan el proyecto, tales como: aplicación móvil, sistemas operativos, comercio electrónico y la metodología Scrum, en el capítulo 4 se encuentra el proceso que se siguió para lograr la implementación de la aplicación móvil y en el capítulo 5 se muestran las pruebas y resultados obtenidos.

Capítulo 2

Generalidades

Algunas microempresas de la Ciudad de México todavía no cuentan con las herramientas tecnológicas adecuadas para poder seguir comercializando sus productos, debido a las dificultades ocasionadas por la pandemia, tales como: mantener la sana distancia, no poder salir libremente de casa, etc. Estos negocios no cuentan con una plataforma que les permita comprar y vender sus productos de manera completamente en línea, por ende, muchas de estas microempresas tienden a desaparecer.

El problema al que se enfrentan las microempresas, es que, al no poder vender en línea, deben contar con un local físico, sin embargo, esto limita la cobertura de clientes al que pueden llegar, además al vender de manera tradicional, los clientes deben desplazarse de sus casas hasta la ubicación física del vendedor.

Para darle solución a esta situación se implementará una aplicación móvil que le permitirá a clientes y vendedores, realizar la compra y venta de sus productos. De esta manera las microempresas podrán captar más clientes y aumentar su volumen de ventas.

2.1. Objetivos

2.1.1. Objetivo general

Implementar una aplicación móvil multiplataforma para la venta, compra y oferta de productos en línea que permita a las microempresas aumentar sus ventas y captar más clientes.

2.1.2. Objetivos específicos

- Analizar la situación de las microempresas.
- Investigar las necesidades de las microempresas.
- Enunciar la lista de los requerimientos para realizar la aplicación móvil.
- Solicitar la requisición necesaria para realizar la aplicación móvil.
- Proponer un diseño de la aplicación móvil.
- Desarrollar la base de datos usando el sistema gestor de base de datos MySQL.
- Maquetar la interfaz gráfica de usuario.
- Ofrecer un método de conexión con la base de datos.
- Implementar la conexión de la aplicación con la base de datos.
- Programar la lógica de la aplicación móvil.
- Realizar pruebas y corrección de errores de la aplicación móvil.
- Liberar la aplicación móvil.

2.2. Justificación

Este proyecto será diseñado, desarrollado y orientado a todas las microempresas de la ciudad de México, pues actualmente estas tienen limitadas sus ventas desde una tienda física por su ubicación geográfica. El presente trabajo analiza la importancia de que las empresas pequeñas implementen las nuevas tecnologías en sus procesos principales para integrarse al comercio electrónico. La utilidad de una tienda virtual es que los micro negocios realicen sus actividades de compra y venta de una manera rápida y eficiente. La realización del presente proyecto busca implementar la versión móvil del proyecto web de la tienda virtual con el propósito de llegar a más clientes para aumentar las ventas y ganancias de las microempresas creando así, una sustentabilidad y rentabilidad en el mercado empresarial.

2.3. Caracterización de la empresa en la que participó

Codeway Soluciones Integrales es una empresa que brinda servicios de consultoría informática a personas y empresas. Entre los servicios que ofrecen se encuentran los siguientes: desarrollo de software, diagrama de redes y páginas web. A continuación, se presentan los datos generales de esta empresa.

2.3.1. Datos generales de la empresa

- **Giro:** Consultora Informática
- **Dirección:** Edo de México, Naucalpan de Juárez, col. La Guadalupana, calle Mario Ruiz de Chávez MZ 4 LT 13

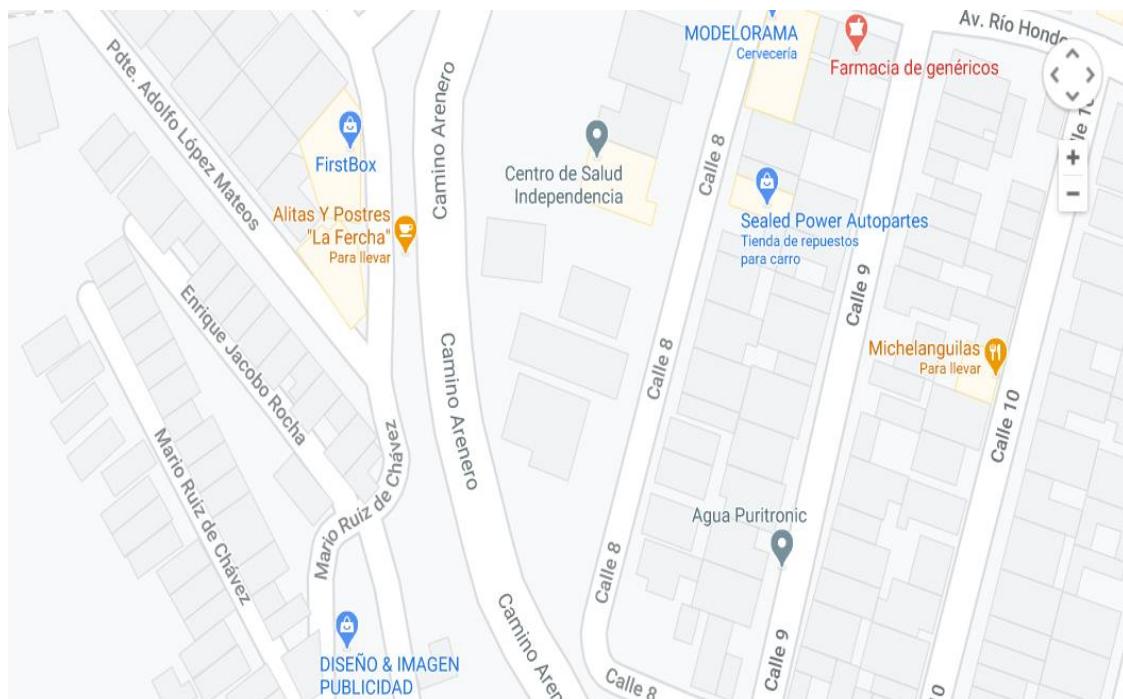


Figura 2.1 Croquis de la ubicación de la empresa "Codeway Soluciones Integrales"

Fuente: <https://www.google.com.mx/maps/@19.434207,-99.2508875,19z>

- **Teléfono:** 2283611136
- **Dirección de correo:** asistencia@codewaymx.com
- **RFC de la empresa:** CSI2008045D7
- **Nombre del departamento:** Tecnologías de la información

Misión

Combinar un profundo conocimiento de industria con el más alto expertise tecnológico en el mercado para crear soluciones personalizadas que te permitan superar tus desafíos de negocio y generar crecimiento disruptivo.

Visión

Intentamos llevar la transformación digital con una perspectiva orientada a negocios y empleando los mejores equipos técnicos disponibles, entrenados para solucionar los obstáculos de tu industria.

Valores

- Responsabilidad.
- Honestidad.
- Excelencia.

Política de calidad

- Aportar soluciones a las necesidades de la comunidad que rodea la empresa como parte de los resultados finales de la misma.
- Manejar precios siempre accesibles al consumidor.
- Contribuir con la formación de una generación de trabajadores nacionales de manera directa e indirecta.
- Fomentar el espíritu de trabajo desde los líderes hacia los trabajadores.

Estructura organizacional

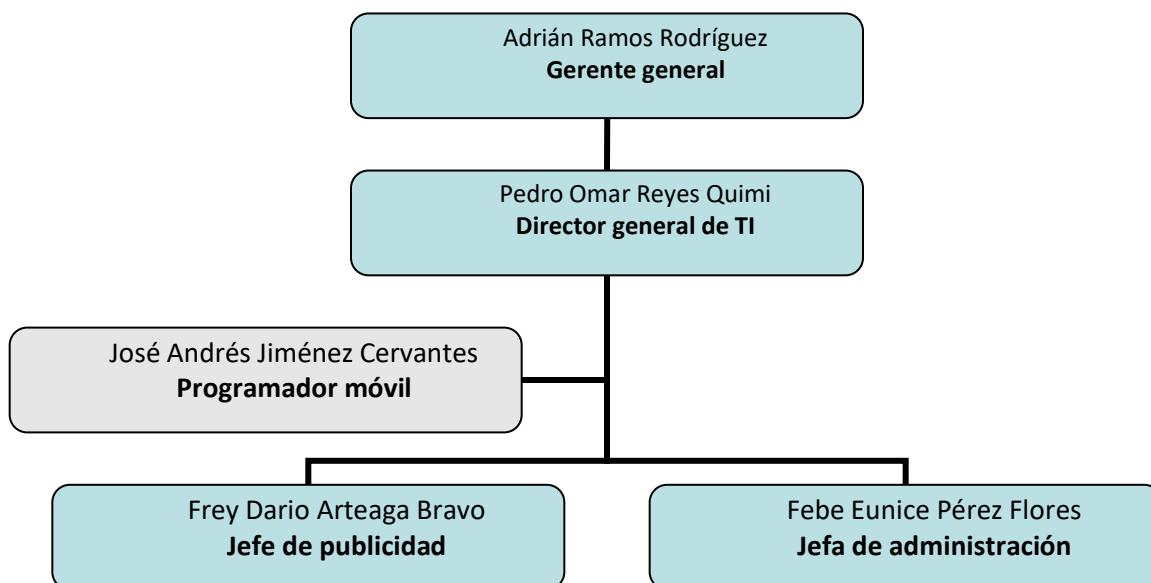


Figura 2.2 Organigrama general. Codeway Soluciones Integrales

2.3.2. Descripción del departamento o área de trabajo

El departamento en el que participé fue el área de Tecnologías de la información y mi jefe directo fue el ING. Pedro Quimi Reyes. Este departamento se encarga de crear soluciones de software para los clientes de la empresa.

2.4. Problemas a resolver (priorizándolos)

Un negocio que cuenta entre 1 a 10 trabajadores se considera que es una microempresa. Estas microempresas tienen la necesidad de incrementar sus ventas y así tener un crecimiento económico.

Para realizar el comercio de sus productos, algunas de estas microempresas lo siguen haciendo de manera tradicional, es decir desde una tienda física. Esto se debe a que no cuentan con sistemas de software que los apoye en este proceso. Actualmente, el no implementar las nuevas tecnologías supone una desventaja frente a los competidores y además con la actual epidemia causada por el COVID-19, muchas personas prefieren realizar sus compras vía internet.

Algunos de los problemas que presentan las microempresas al vender solo desde un canal físico son:

- Disminución en el volumen de ventas: La epidemia por el COVID-19, evita que las personas salgan libremente, es por eso que prefieren realizar sus compras vía internet.
- Pocos clientes: El depender de una ubicación física, provoca que los productos que ofrecen las microempresas no lleguen a clientes que se encuentran fuera de su zona geográfica.
- Riesgo de contagio de COVID-19: Al tener un contacto presencial con los clientes, los trabajadores de las microempresas ponen en riesgo su salud, así como la de sus clientes.
- Desaparición de microempresas: Al no generar más ventas, algunas empresas quiebran y desaparecen al no poder sobrevivir en el mercado.

Con el propósito de dar solución a esta problemática, se va a desarrollar una tienda virtual, dirigida a dispositivos móviles que permita la compra y venta de productos de manera remota.

2.5. Alcances y limitaciones

- La aplicación solo llegará a usuarios de la ciudad de México.
- En caso de que se requieran cambios se deberá ajustar en posteriores versiones de la aplicación móvil.
- Se necesita internet en los smartphones para hacer compras o ventas sobre el aplicativo.
- No todos los usuarios tienen un smartphone para instalar el aplicativo.
- Como es una aplicación móvil no se podrá instalar sobre equipos de escritorio o laptops.

2.6 Estudio de factibilidad y viabilidad

En la tabla 2-1 se muestra el estudio de factibilidad del producto de software.

Concepto	Descripción	Cantidad	Costo unitario	
EQUIPOS				
Computadora	Equipo de cómputo portátil	2	\$3,000.00	\$6,000.00
Impresora	Dispositivo periférico de impresión	1	\$500.00	\$500.00
Costo de equipos				6,500.00
LICENCIAS				
Paquetería de ofimática	Microsoft Office 365	2	\$125.00	\$250.00
SGBD	MySQL	1	\$ 0.00	\$ 0.00
Herramienta CASE	StarUML	1	\$350.00	\$350.00
IDE	Android Studio	1	\$ 0.00	\$ 0.00
Sistema Operativo	Windows 10 Home	1	\$300.00	\$300.00
Costo de licencias				900.00
GASTOS DE ADMINISTRACIÓN				
Tinta para impresora	Paquete de 4 tintas	1	\$500.00	\$500.00
Papelería	Paquete tamaño carta 500 hojas	1	\$100.00	\$100.00
Consumo de luz	Bimestral		\$950.00	\$950.00
Consumo de agua	Bimestral		\$200.00	\$200.00
Internet y telefonía	Mensual		\$400.00	\$400.00
Transporte	Diario por trabajador	6	\$100.00	\$600.00
Costo de administración				2,750.00
RECURSOS HUMANOS				
Líder de proyecto	Encargado de la dirección del proyecto	1	*Co ntrato	\$10,000.00
Analista	Encargado de análisis del proyecto	1	*Co ntrato	\$5,000.00
Entrevistador	Encargado de entrevistar al cliente	1	*Co ntrato	\$5,000.00
Diseñador	Encargado del diseño del sistema	1	*Co ntrato	\$8,000.00
Programador	Encargado de la programación del sistema	2	*Co ntrato	\$8,000.00
Costo de recursos humanos				44,000.00
Costo de EQUIPOS				\$6,500.00
Costo de SOFTWARE				\$ 900.00
Costo de ADMINISTRACIÓN				\$2,750.00
Costo de RECURSOS HUMANOS				\$44,000.00
COSTO TOTAL DEL SISTEMA				54,150.00

Tabla 2-1 Estudio de factibilidad

2.7 Factibilidad operacional

La situación actual de muchas microempresas ha expresado la necesidad de implementar este sistema y optimizar las actividades de compra y venta de sus productos, además de contar con interfaces intuitivas, por lo cual el sistema es funcional y operable.

2.8 Estudio de viabilidad

Basándonos en los resultados obtenidos en el estudio de factibilidad, se concluye que se cuenta con los recursos tecnológicos, económicos y operacionales necesarios para llevar a cabo el proyecto por lo que es totalmente viable continuar con el mismo.

Capítulo 3

Fundamento teórico

En este capítulo se verán conceptos teóricos referentes a las microempresas, comercio electrónico, aplicaciones móviles, así como también se abordarán conceptos técnicos que tienen que ver con todo el proceso de desarrollo de una aplicación móvil, tales como servicios web, base de datos, servidores, lenguajes de programación, entorno de desarrollo, sistemas operativos, etc.

3.1 Microempresas y comercio electrónico

Una microempresa es un negocio que cuenta de entre 1 a 10 trabajadores, algunos ejemplos de microempresas pueden ser: una tortillería familiar, una pollería, una tienda de aparatos electrónicos, un local de ropa, un local de zapatos, etc. Muchas de estas microempresas han enfrentado diferentes dificultades a causa del confinamiento. Algunas de ellas integraron adecuadamente las tecnologías de la información para sobrevivir, implementando el marketing digital y las ventas en línea. Sin embargo, según (Armenta, 2020) más del 80 % de las microempresas no tienen acceso a sistemas informáticos para enfrentar estas dificultades. De ahí surge la necesidad de integrar poco a poco el comercio electrónico en las empresas. El comercio electrónico se define de la siguiente manera.

“Uso de las tecnologías de la informática y telecomunicaciones, que soportan las transacciones de productos o servicios entre las empresas, entre estas y particulares o con el Estado”. (G, 2001)

De acuerdo con esta definición un comercio electrónico es aquel que utiliza las tecnologías de la información, tales como los sistemas de software, además también deben implementar las telecomunicaciones, es decir la tecnología que permite que las personas estén comunicadas entre sí, tales como: el internet, los teléfonos celulares, las laptops, computadoras de escritorio, etc. En la actualidad, el internet se encuentra en una gran variedad de dispositivos, que van desde los más grandes hasta los más pequeños, es por ello que surge el concepto de internet de las cosas. Aprovechando las ventajas que brinda internet, en este proyecto se decidió desarrollar una sistema de tienda virtual que funcione bajo plataformas móviles, en otras palabras desarrollar una aplicación móvil.

3.2 Aplicación móvil

Una aplicación móvil es un sistema informático que se ejecuta en dispositivos móviles, tales como teléfonos celulares, tabletas, smartwatch, etc. Hoy en día la aplicación de estas apps se encuentra en varios ámbitos, tales como: la educación, el entretenimiento, el comercio, el sector bancario, etc. Esto se debe a la asequibilidad que se tiene de los teléfonos celulares, pues es más fácil y barato adquirir un teléfono celular que una computadora de escritorio o laptops, pues gracias a las redes móviles es posible conectarnos a internet desde un celular.

Las aplicaciones móviles han surgido para brindar mayor movilidad y comodidad a las personas cuando se encuentran realizando alguna actividad, pues con estos dispositivos podemos hacer tareas complejas, tales como abrir un navegador y abrir páginas de internet, abrir documentos desde una aplicación de ofimática, hacer video llamadas, etc.

Dentro de los sistemas operativos más utilizados para dispositivos móviles existen 2: los cuales son iOS desarrollado por la compañía Apple y Android que es propiedad de Google.

3.3 Sistemas operativos móviles

Así como las computadoras requieren de un sistema operativo para poder interactuar con el hardware, los teléfonos celulares también tienen su propio software para que el usuario interactúe con él a través de una interfaz interactiva y fácil de usar. La diferencia entre un sistema operativo móvil y uno normal según (Vega, 2017) se describe a continuación.

“Los sistemas operativos móviles difieren a los sistemas operativos normales en seguridad, el acceso y manejo de hardware especial para telecomunicaciones, y la forma en la que se aceptan y distribuyen sus aplicaciones”.

Al desarrollar aplicaciones móviles, el programador debe prever y declarar que permisos de hardware que va a utilizar la aplicación, estos permisos pueden ser de: gps, cámara, etc y también debe tener en cuenta que las aplicaciones solo se distribuyen a través de los sistemas autorizados de estas plataformas. En Android, la forma de distribuir aplicaciones es mediante la Google Play Store, por otro lado, en iOS la forma de distribuir apps es por medio de la App Store.

En ambos sistemas operativos se debe declarar un manifiesto de permisos en el cual se solicitan los aditamentos necesarios para que la aplicación funcione correctamente, algunos ejemplos de permisos son: acceso al GPS, accounts, acelerómetros, cámara, micrófono, contactos, galería de fotos, archivos, etc.

La forma en la que estos sistemas manejan el sistema de archivos es diferente. En Android es posible acceder a una SD CARD y además cuenta con un sistema de archivos interno basado en *NIX. Por otro lado, en el mundo de iOS se utiliza un sistema de contenedores para sus apps, basado en una teoría llamada sandbox, que básicamente aisla las apps.

3.4 Android

Android es un sistema operativo para dispositivos móviles con pantalla táctil , su creador fue Andy Rubin, pero posteriormente fue comprado por Google en el 2005, este sistema operativo está disponible en una gran variedad de dispositivos según lo que dice (platzi.com, s.f.) “Android es el Sistema Operativo de código abierto más popular en todo el mundo y no se limita a teléfonos móviles, sino que además está incluido en *Smartwatch*, *IoT* con *Android Things* e inclusive en automóviles autónomos con *Android Auto* y con la posibilidad de crear aplicaciones en cada uno de ellos.”

Desde 2007 Android pertenece al Open Handset Alliance lo que lo hizo despegar y dominar en la mayoría de las marcas de dispositivos móviles. Este es un consorcio compuesto por las marcas de hardware en el mercado, en él se encuentra *Samsung*, *LG*, *Sony*, *Toshiba*, *Dell*, etc. (platzi.com, s.f.)

A lo largo de la evolución de Android se han lanzado diferentes versiones y esto es un punto a tomar en cuenta al desarrollar una aplicación para esta plataforma debido a que la aplicación solo estará disponible para la versión de Android que se elija y las versiones más recientes afectando la disponibilidad de la aplicación. En la tabla 3-1 se muestran las versiones de Android que han sido lanzadas.

Nombre de la versión	Descripción
Android Apple Pie	Versión 1.0 y fecha de lanzamiento 23 de septiembre de 2008.
Android Banana Bread	Versión 1.1 y fecha de lanzamiento 9 de febrero de 2009.
Android Cupcake	Versión 1.5 y fecha de lanzamiento 25 de abril de 2009
Android Donut	Versión 1.6 y fecha de lanzamiento 15 de septiembre de 2009.
Android Eclair	Versión 2.0-2.1 t fecha de lanzamiento 26 de octubre de 2009.
Android Froyo	Versión 2.2-2.3 y fecha de lanzamiento 20 de mayo de 2010.
Android Gingerbread	Versión 2.3-2.7 y fecha de lanzamiento 6 de diciembre.
Android Honeycomb	Versión 3.3-3.2.6 y fecha de lanzamiento 22 de febrero de 2011.
Android Ice Cream Sandwich	Versión 4.0-4.0.5 y fecha de lanzamiento 18 de octubre de 2011.
Android Jelly Bean	Versión 4.1-4.3.1 y fecha de lanzamiento 9 de julio de 2012.
Android Kitkat	Versión 4.4-4.4.4 y fecha de lanzamiento 31 de octubre de 2012.
Android Lollipop	Versión 5.0-5.1.1 y fecha de lanzamiento 12 de noviembre de 2014.
Android Marshmallow	Versión 6.0-6.0.1 y fecha de lanzamiento 5 de octubre de 2015.
Android Nougat	Versión 7.0-7.1.2 y fecha de lanzamiento 15 de junio de 2016.
Android Oreo	Versión 8.0-8.1 y fecha de lanzamiento 21 de agosto de 2017.
Android Pie	Versión 9.0 y fecha de lanzamiento 6 de agosto de 2018.
Android 10	Versión 10.0 y fecha de lanzamiento 3 de septiembre de 2019.
Android 11	Versión 11.0 lanzado el 8 de septiembre de 2020.

Tabla 3-1 Versiones de Android (Adeva, 2021).

Para desarrollar aplicaciones Android se recomienda utilizar el IDE oficial de Android que es Android Studio, pues está diseñado totalmente para desarrollar aplicaciones móviles muy rápido y tiene como motor constructor de aplicaciones a Gradle .

De acuerdo con la definición de (Muradas, 2020) “Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando Groovy o Kotlin”, es decir que Gradle es una herramienta de construcción de desarrollo y lo que hace es tomar las dependencias y librerías externas que se implementan en el proyecto, las integra y construye un archivo ejecutable con la extensión **apk**, el significado de apk es application package y es un archivo de empaquetado de aplicaciones Android.

Dentro del proceso de desarrollo de una aplicación móvil, además de elegir el software y herramientas necesarias para implementarlo, también es importante elegir una metodología de desarrollo de software que

se ajuste a las necesidades del proyecto, es por ello que en el presente proyecto se decidió implementar la metodología de desarrollo ágil Scrum, por lo que a continuación se describe esta metodología con cada uno de sus componentes y fases

3.5 Metodología Scrum

Según (Romero, 2019) “Scrum es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregan productos del máximo valor posible productiva y creativamente”.

Esto quiere decir que Scrum es una metodología implementada por un equipo de desarrollo de software. Implementado para crear software ligero o robusto y con la mayor calidad posible.

El diagrama de Scrum se muestra en la siguiente imagen.

SCRUM FRAMEWORK

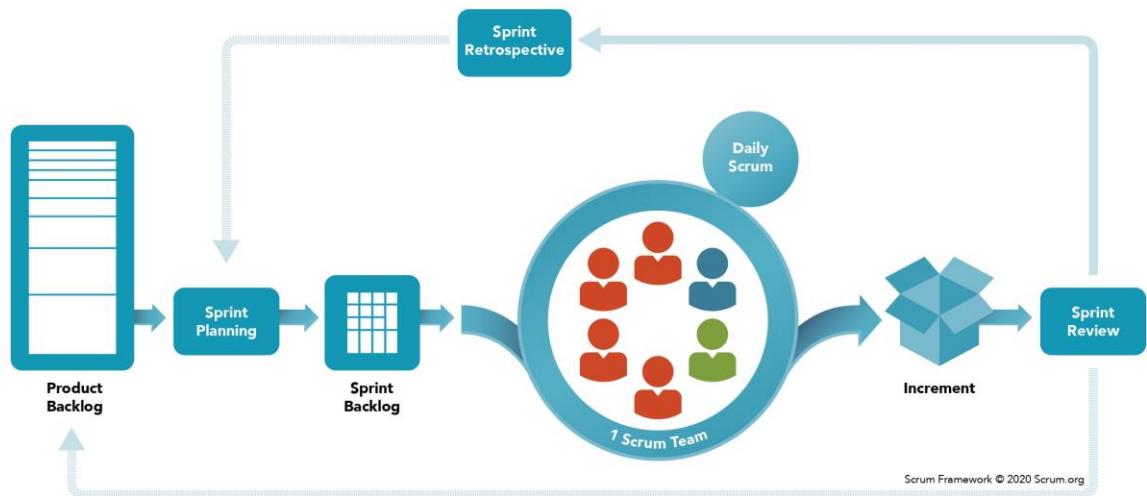


Figura 3.1 Diagrama de Scrum

Scrum es una metodología ágil de desarrollo de software y se compone de los siguientes elementos:

Product Backlog

El Product backlog se refiere a la lista del producto obtenido a través de la entrevista que se le realiza al cliente con el fin de obtener los requerimientos del software. A cada uno de los elementos de la lista se le asigna un numero de prioridad, para que al realizar el Sprint planning, estos sean los próximos elementos en desarrollarse.

Sprint Planning

El Sprint planning es una reunión entre los miembros de Scrum: El Product owner, scrum master y el equipo de desarrollo, en esta reunión deciden cuáles serán los elementos o requerimientos que se van a desarrollar en el sprint.

Sprint Backlog

El Sprint backlog son todos los elementos de la lista del producto que se está trabajando en el sprint actual.

Sprint Review

El Sprint review se refiere a la revisión del sprint, para hacer esto, se organiza una reunión entre el equipo de Scrum y evalúan los resultados del sprint actual, para mejorar en el próximo.

Sprint Retrospective

La retrospectiva del Sprint es una reunión en donde el equipo dialoga acerca de lo que hizo bien y de lo que no hizo bien, con el fin de mejorar en el desarrollo del próximo sprint.

Ventajas de Scrum

- Es una metodología de desarrollo ágil que permite responder al cambio.
- Trabaja con equipos pequeños de entre 3 a 9 personas.
- Los equipos son autoorganizados.
- Trabaja a través de sprints o iteraciones.
- Se hacen entregas funcionales del software.
- Se recibe retroalimentación continua para mejorar.
- Se obtiene un producto mínimo viable.
- Se puede aplicar a sistemas sencillos hasta los más robustos.

3.6 Lenguajes de programación y Frameworks

Para crear aplicaciones Android existen varios lenguajes de programación dependiendo del tipo de aplicación, existen aplicaciones nativas en las que se utiliza el lenguaje de programación java o kotlin para manejar la lógica del negocio y el lenguaje de etiquetado XML para la creación de las interfaces de usuario, por otro lado están las aplicaciones multiplataformas que son aplicaciones que pueden ser ejecutadas en iOS y Android, para crear estas aplicaciones se pueden utilizar frameworks como React Native, que utiliza JavaScript, HTML y CSS para la creación de interfaces, sin embargo, para manejar la funcionalidad de la aplicación se debe utilizar java o kotlin, otra opción para desarrollar estas aplicaciones es Xamarin que utiliza el lenguaje de programación C Sharp. Por último, se tiene el SDK de Flutter para desarrollar aplicaciones multiplataformas, este utiliza el lenguaje de programación Dart. Flutter es el SDK que se utilizará para desarrollar la aplicación móvil.

3.7 Flutter y Dart



Figura 3.2 Logo de Flutter y Dart

Fuente:

<https://www.google.com/url?sa=i&url=https%3A%2F%2Fboostmysites.com%2Fdifference-between-flutter-and-dart%2F&psig=AOvVaw0P2iYNuZ55mq9R95OjXxtv&ust=1625488901963000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCOikzp64yfECFQAAAAAdAAAAABAN>

(Salgado, 2018) dice que “Flutter es un SDK creado por Google, pensado para desarrollar aplicaciones nativas (cross platform) para iOS y Android”.

El principal motor de Flutter es la creación de interfaces móviles. La forma en la que Flutter se compila es la siguiente: Flutter compila directamente hacia el código final que interactúa con el procesador, por lo

que las aplicaciones resultantes tienen un mejor rendimiento. Flutter usa sus propios Widgets y motor de renderizado llamado Skia Canvas desarrollado en C++.

El lenguaje de programación que utiliza Flutter es Dart, este es un lenguaje orientado a objetos de acuerdo con lo que dice (*Salgado, 2018*) “Dart es un lenguaje de programación que usa el paradigma de Programación Orientada a Objetos (OOP)”.

3.8 Arquitectura de diseño

Además de elegir el framework con el que se trabajará, también es necesario elegir la arquitectura de diseño que se implementará en el desarrollo de la aplicación. La arquitectura de diseño se refiere a la forma en la que se va a estructurar el proyecto, (*Salgado, platzi.com, s.f.*) lo define así, “La arquitectura es el arte y técnica de diseñar, proyectar y construir. Una Arquitectura de diseño proporciona la estructura, funcionamiento e interacción entre las partes del software.”

Dentro de las arquitecturas más conocidas se encuentran MVC (Modelo-Vista-Controlador), MVP (Modelo-Vista-Presentador) y MVVM (Modelo-Vista-Vista Modelo), sin embargo, para el desarrollo de aplicaciones con Flutter se recomienda implementar la arquitectura BLoC (Business logic Components).

3.9 Arquitectura Bloc en Flutter

La arquitectura BLoC es un patrón de diseño que estructura el proyecto en capas separando la lógica del negocio de la interfaz de usuario. A continuación, se describen las diferentes capas de la arquitectura.

- View (UI Screen): Contendrá toda la interacción con las vistas; se puede organizar en screens y widgets.
- BLoC: La capa de negocio estará contenida en esta capa. Aquí se encuentra toda la lógica y funcionalidad de la aplicación.
- Repository: En esta capa se concentran las clases que se conectan con una fuente de datos; API, Endpoints, DataBase, etc.
- Data / Model: Son los modelos, los cuales ayudan a manejar los datos.

A continuación, se abordarán los temas de base de datos, servicios web, API'S y endpoints, para comprender mejor la capa de Repository de la arquitectura Bloc.

3.10 Sistema gestor de base de datos MySQL

Las bases de datos en aplicaciones móviles son imprescindibles, debido a la necesidad que se tiene de dar persistencia de los datos. Las bases de datos se clasifican en relacionales y no relacionales.

Las de bases de datos relacionales más conocidas son MySQL, SQL Server y PostgreSQL, por otro lado, las bases de datos no relacionales tienen su propia clasificación, según (Morales I. V., 2019) los tipos de base de datos no relacionales son:

- **Clave - valor:** Son ideales para almacenar y extraer datos con una clave única. Manejan los diccionarios de una manera excepcional. Ejemplos: **DynamoDB, Cassandra**.
- **Basadas en documentos:** Son una implementación de clave-valor que varía en la forma semiestructurada en que se trata la información. Ideal para almacenar datos JSON y XML. Ejemplos: **MongoDB, Firestore**.
- **Basadas en grafos:** Basadas en teoría de grafos, sirven para entidades que se encuentran interconectadas por múltiples relaciones. Ideales para almacenar relaciones complejas. Ejemplos: **neo4j, TITAN**.
- **En memoria:** Pueden ser de estructura variada, pero su ventaja radica en la velocidad, ya que al vivir en memoria la extracción de datos es casi inmediata. Ejemplos: **Memcached, Redis**.
- **Optimizadas para búsquedas:** Pueden ser de diversas estructuras, su ventaja radica en que se pueden hacer consultas y búsquedas complejas de manera sencilla. Ejemplos: **BigQuery, Elasticsearch**.

En el presente proyecto se utilizará una base de datos relacional SQL con el sistema gestor de base de datos MySQL.

3.10 MySQL

MySQL es un sistema gestor de base de datos relacional de código abierto y es considerado como una de las bases de datos más populares del mundo. Fue desarrollado por MySQL AB, pero fue adquirida por Sun MicroSystems en 2008 y esta a su vez fue comprada por Oracle Corporation en 2010. Es por eso que MySQL cuenta con una doble licencia. Por una parte, es de código abierto, pero por otra, cuenta por una versión comercial controlada por la compañía Oracle.

3.10.1 Niveles de abstracción de una base de datos

Los sistemas gestores de base de datos tienen 3 niveles de abstracción, los cuales son:

Nivel físico: en este nivel se describe la estructura interna de la base de datos mediante un esquema. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como también los métodos de acceso.

Nivel Lógico: en este nivel se define toda la estructura de la base de datos mediante un esquema conceptual. En este esquema se describen las entidades, atributos, relaciones, operaciones de los usuarios y restricciones.

Nivel de vistas: En este nivel, los usuarios solo pueden ver una simplificación en su interacción con el sistema, ya que ven un conjunto de vistas que esconden los detalles de la base de datos. (cs.us.es, n.d.)

3.10.2 Componentes del sistema de gestión de base de datos

Un sistema de gestión de base de datos tiene varios componentes y cada uno de ellos realiza una función específica. Con base en lo que dice(cs.us.es, n.d.) los componentes de un SGBD son los siguientes:

El procesador de consultas: es el componente principal de un SGBD. Transforma las consultas en un conjunto de instrucciones de bajo nivel que se dirigen al gestor de la base de datos.

El gestor de la base de datos: es la interface con los programas de aplicación y las consultas de los usuarios. El gestor de la base de datos acepta consultas y examina los esquemas externo y conceptual para determinar qué registros se requieren para satisfacer la petición. Entonces el gestor de la base de datos realiza una llamada al gestor de ficheros para ejecutar la petición.

El gestor de ficheros: maneja los ficheros en disco en donde se almacena la base de datos. Este gestor establece y mantiene la lista de estructuras e índices definidos en el esquema interno. Si se utilizan ficheros dispersos, llama a la función de dispersión para generar la dirección de los registros. Pero el gestor de ficheros no realiza directamente la entrada y salida de datos. Lo que hace es pasar la petición a los métodos de acceso del sistema operativo que se encargan de leer o escribir los datos en el buffer del sistema.

El preprocesador del LMD: convierte las sentencias del LMD embebidas en los programas de aplicación, en llamadas a funciones estándar escritas en el lenguaje anfitrión. El preprocesador del LMD debe trabajar con el procesador de consultas para generar el código apropiado. El compilador del LDD

convierte las sentencias del LDD en un conjunto de tablas que contienen metadatos. Estas tablas se almacenan en el diccionario de datos.

El gestor del diccionario: controla los accesos al diccionario de datos y se encarga de mantenerlo. La mayoría de los componentes del SGBD acceden al diccionario de datos.

Control de autorización: Este módulo comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita.

Procesador de comandos: Una vez que el sistema ha comprobado los permisos del usuario, se pasa el control al procesador de comandos.

Control de la integridad: Cuando una operación cambia los datos de la base de datos, este módulo debe comprobar que la operación a realizar satisface todas las restricciones de integridad necesarias.

Optimizador de consultas: Este módulo determina la estrategia óptima para la ejecución de las consultas.

Gestor de transacciones: Este módulo realiza el procesamiento de las transacciones.

Planificador (scheduler): Este módulo es el responsable de asegurar que las operaciones que se realizan concurrentemente sobre la base de datos tienen lugar sin conflictos.

Gestor de recuperación: Este módulo garantiza que la base de datos permanece en un estado consistente en caso de que se produzca algún fallo.

Gestor de buffers: Este módulo es el responsable de transferir los datos entre memoria principal y los dispositivos de almacenamiento secundario. A este módulo también se le denomina gestor de datos.

3.11 Modelo Entidad-relación

El modelo entidad-relacion es una representación visual de las entidades, atributos, relaciones que va a tener una base de datos, estos se representan a través de símbolos y flechas. Este diagrama nos permite modelar y entender cuáles son las entidades y cuáles son sus relaciones. Después de hacer el modelo entidad- relación, este se migra a un diagrama relacional en cuál se aplica el proceso de normalización.

3.11.1 Normalización

La normalización es el proceso por cuál se refina el modelo de la base de datos para que la información sea consistente y clara. Existen 4 formas normales, las cuales se explican a continuación:

Primera Forma Normal (1FN)

Esta FN nos ayuda a eliminar los valores repetidos y no atómicos dentro de una base de datos.

Formalmente, una tabla está en primera forma normal si:

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- No debe existir variación en el número de columnas.
- Los campos no clave deben identificarse por la clave (dependencia funcional).
- Debe existir una independencia del orden tanto de las filas como de las columnas; es decir, si los datos cambian de orden no deben cambiar sus significados.

Segunda Forma Normal (2FN)

Esta FN nos ayuda a diferenciar los datos en diversas entidades.

Formalmente, una tabla está en segunda forma normal si:

- Está en 1FN.
- Sí los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir, que no existen dependencias parciales.
- Todos los atributos que no son clave principal deben depender únicamente de la clave principal.

Tercera Forma Normal (3FN)

Esta FN nos ayuda a separar conceptualmente las entidades que no son dependientes.

Formalmente, una tabla está en tercera forma normal si:

- Se encuentra en 2FN
- No existe ninguna dependencia funcional transitiva en los atributos que no son clave.
- Esta FN se traduce en que aquellos datos que no pertenecen a la entidad deben tener una independencia de las demás y debe tener un campo clave propio.

Cuarta Forma Normal (4FN)

Esta FN nos trata de atomizar los datos multivaluados de manera que no tengamos datos repetidos entre filas.

Formalmente, una tabla está en cuarta forma normal si:

- Se encuentra en 3FN
- Los campos multivaluados se identifican por una clave única

Esta FN trata de eliminar registros duplicados en una entidad, es decir que cada registro tenga un contenido único y de necesitar repetir los datos en los resultados se realiza a través de claves foráneas.

3.12 Sistema de gestión de base de datos MySQL

MySQL es el motor más utilizado en el mercado, esto porque el *stack* de tecnología más utilizado es LAMP (Linux, Apache, MySQL y PHP) con esto se sirve la mayor parte de internet, aunque MariaDB, que es una base de datos de código abierto mantenido por la comunidad, es muy similar a MySQL.

3.12.1 Historia

SQL es un lenguaje que se basó en 2 principios fundamentales en la parte teórica: la teoría de conjuntos y el álgebra relacional de Edgar Codd. Edgar Codd fue una persona que se dedicó a evolucionar la teoría de conjuntos.

La compañía IBM en los años 70 creó un lenguaje llamado SEQUEL. Este es el nombre en el que se basa el moderno SQL.

Después apareció la compañía Relational Company (actualmente Oracle). Oracle creó su versión de software que llamó Oracle V2 en el año 1979 y este software se convirtió en uno de los motores de base de datos más famosos.

Posteriormente en el año 1989, SQL se convierte en un lenguaje estándar. SQL viene a resolver el problema que existía en esa época, las bases de datos en aquel momento estaban basadas en archivos y el consultar datos dentro de ella se volvía demasiado complicado. Y entonces SQL se convierte en el lenguaje que unifica a las bases de datos relacionales, es decir se convierte en una norma ANSI/ISO en el que se publican normas que todas las bases de datos deben implementar. De ahí surgieron varios motores de bases de datos y uno de ellos es MySQL.

“MySQL es una idea originaria de la empresa opensource Mysql AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael Monty Widenius. El objetivo que

persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo llevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

La procedencia del nombre de MySQL no es clara. Desde hace más de 10 años, las herramientas han mantenido el prefijo My. También, se cree que tiene relación con el nombre de la hija del cofundador Monty Widenius quien se llama My.

Por otro lado, el nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso “Name the Dolphin”. Este nombre fue enviado por Ambrose Twebaze, un desarrollador de software de código abierto africano, derivado del idioma SiSwate, el idioma local de Swazilandia y corresponde al nombre de una ciudad en Arusha, Tanzania, cerca de Uganda la ciudad origen de Ambrose.” (ecured.cu, s.f.)

Para conectar la aplicación móvil, es necesario hacerlo desde un servicio web, para no poner en riesgo los datos de los usuario, pues no se recomienda conectarse directamente a la base de datos desde el front-end.

3.13 Servicio web

Un servicio web es un programa informático que tiene la función de comunicar la base de datos con el front-end (aplicación móvil), este servicio está alojado en un servidor, lo cual le permite estar disponible en internet, de ahí el nombre de “servicio web”. En el presente proyecto se hace uso de una API REST, desarrollada con el lenguaje de programación JavaScript y el framework NodeJS.

3.13.1 API REST

“API son las siglas de *Application Programming Interface* o interfaz de programación de aplicaciones, se trata de un conjunto de reglas que definen como 2 aplicaciones interactuarán entre sí”. (Chojrin, 2019)

Es decir, una API REST, funciona como intermediario entre el cliente y el servidor haciendo que estos se comuniquen a través del protocolo HTTP, la API recibe las peticiones por parte del cliente y las envía al servidor, luego el servidor envía una respuesta al cliente. Los métodos más comunes de una API REST son GET, POST, PUT y DELETE.

HTTP

Son las siglas de *Hypertext Transfer Protocol* por sus en inglés que significa protocolo de transferencia de hipertexto. Un protocolo, es un conjunto de reglas que definen como se dará la comunicación entre 2 entidades y en el caso de HTTP, las entidades son computadoras, un hipertexto se trata de un texto que contiene referencias a otros textos distintos, ya sea enlaces o links.

El protocolo HTTP funciona de la siguiente manera:

La computadora cliente envía un mensaje siguiendo las reglas de http y este mensaje se transmite a través de internet y llega a otra computadora que es un servidor, el objetivo del servidor es interpretar ese mensaje, procesarlo, generar una respuesta y enviársela al cliente a través de internet, en la siguiente figura, se muestra un esquema de cómo funciona el protocolo HTTP.

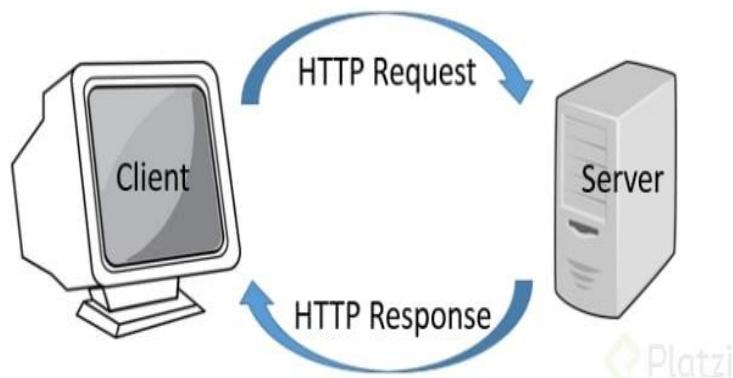


Figura 3.3 Funcionamiento del protocolo HTTP

Fuente:

<https://www.google.com/url?sa=i&url=https%3A%2F%2Fplatzi.com%2Fclases%2F1638-api-rest%2F21614-que-es-y-como-funciona-el-protocolo-http%2F&psig=AOvVaw2r3iZoWy5An0uhx8b-8HT8&ust=1626132278091000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCMCX4fmU3PECFQAAAAAdAAAAABAD>

Capítulo 4

Desarrollo

4.1 Análisis

4.1.1 Entrevista con la empresa “Codeway Soluciones Integrales”

1. ¿Qué tipo de sistema se requiere?

Se requiere una tienda virtual que se ejecute en dispositivos móviles, pues ya se tiene una versión web de este sistema.

2. ¿Quién va a ocupar la aplicación?

Los principales usuarios de la aplicación serán las microempresas y sus clientes.

3. ¿Qué datos del usuario requieren ser guardados?

Los datos mínimos que se desean almacenar del usuario son:
Nombre, apellido paterno, email y contraseña.

4. ¿Qué permisos va a tener el usuario?

Todos los usuarios tienen permisos para ser vendedor y a la vez cliente.

5. ¿Qué actividades va a realizar el usuario?

El usuario podrá hacer lo siguiente:

- Iniciar sesión en la aplicación.
- Subir sus productos.
- Vender productos.
- Comprar productos.
- Agregar productos a su carrito de compras.

6. Actualmente, ¿Cómo realiza esta actividad?

Actualmente muchas microempresas realizan la venta de sus productos de manera tradicional, desde un local físico, el cliente debe acudir al local y realizar la compra del producto.

7. ¿Qué datos del producto se desean almacenar?

Se desea almacenar el nombre del producto, una breve descripción del mismo, su precio y la cantidad disponible para su venta.

8. ¿Qué datos de la venta requieren ser almacenados?

El ticket de venta debe tener los siguientes datos:

- Clave de transacción.
- Fecha.
- Total.

9. ¿En qué contexto será utilizado el producto?

La aplicación móvil estará disponible para todas aquellas microempresas que necesitan integrarse al comercio electrónico.

10. ¿Qué medios de pago requiere implementar en este producto?

De momento se requiere implementar pagos por medio de mercado pago.

4.1.2 Requerimientos funcionales del sistema

Con base en la entrevista realizada a la empresa, se obtuvieron los siguientes requerimientos funcionales del sistema de ventas online para microempresas.

1. El usuario puede iniciar sesión en la aplicación. Los datos que debe ingresar son:
 - A. Email
 - B. Contraseña
2. El usuario puede ver la lista de productos en venta. Los productos deben tener los siguientes datos:
 - A. Nombre del producto
 - B. Descripción del producto
 - C. Precio Unitario
 - D. Imagen del producto
 - E. Cantidad disponible
 - F. Estado del producto
3. El usuario puede registrarse en la aplicación. Los datos mínimos para registrarse son:
 - A. Nombre
 - B. Apellido paterno
 - C. Email
 - D. Contraseña
4. El usuario puede agregar productos a su carrito de compras. Los datos que guardará el carrito son:
 - A. Cantidad de producto
 - B. Precio total
5. El usuario puede eliminar productos de su carrito de compras.
6. El usuario puede comprar productos. El ticket de la compra debe tener los siguientes datos:
 - A. Clave de transacción.
 - B. Fecha.
 - C. Total.
7. El usuario puede publicar sus productos para venderlos.
8. El usuario puede actualizar los datos de sus productos en venta.
9. El usuario puede modificar sus datos del perfil.
10. El usuario puede agregar diferentes direcciones. La dirección tiene los siguientes datos:
 - A. Código postal.
 - B. Estado.
 - C. Municipio.
 - D. Colonia.

- E. Calle principal.
- F. Número exterior.
- G. Calle 1.
- H. Calle 2.
- I. Referencia.

11. El usuario puede realizar sus pagos por medio de mercado pago

- 12. El usuario puede comentar acerca de los productos.
- 13. El usuario puede ver los comentarios de los demás usuarios.
- 14. El usuario puede cerrar sesión.

4.2 Diseño

4.2.1 Diagrama entidad-relación de la base de datos

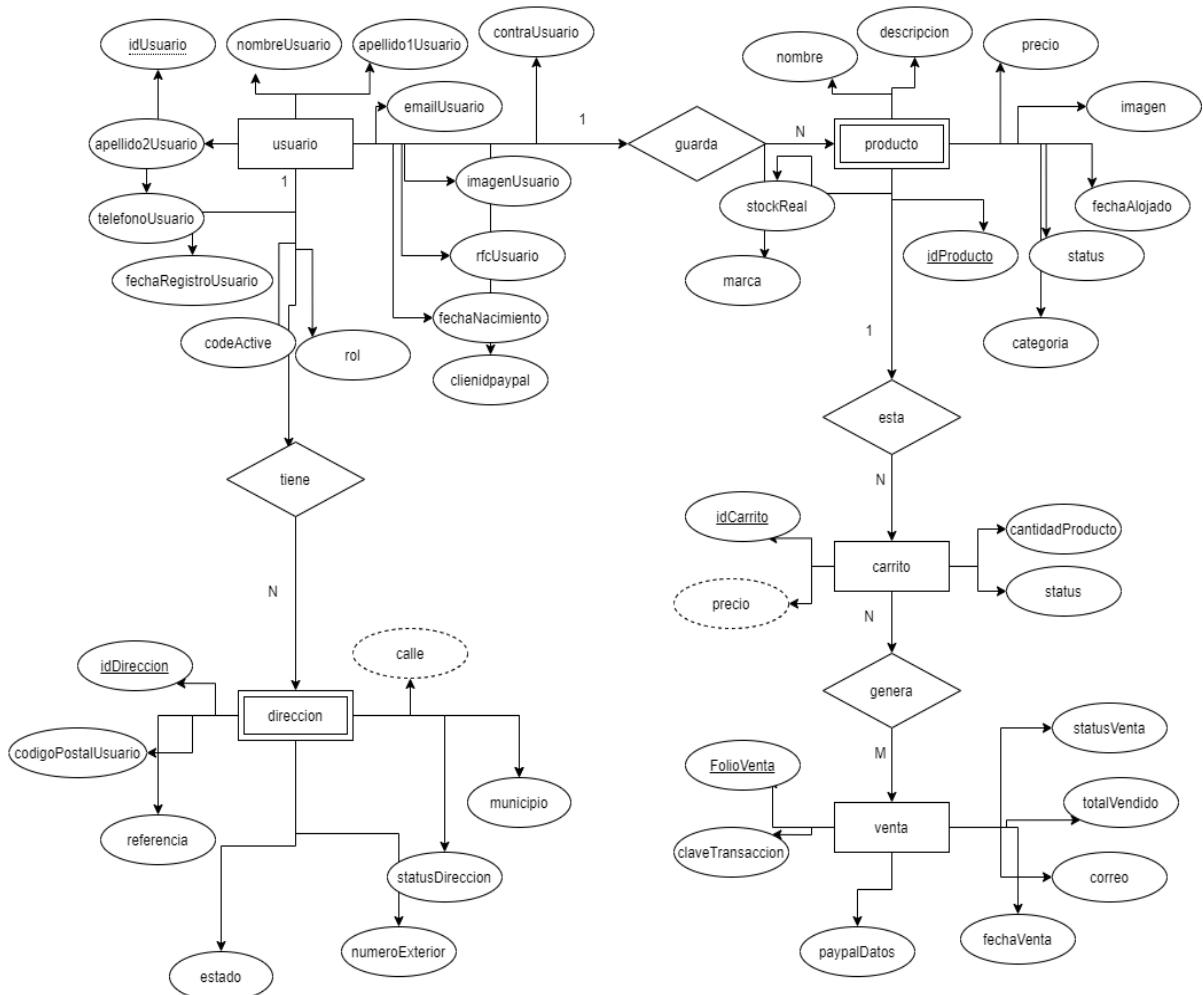


Figura 4.1 Diagrama entidad-relación

En la figura 4.1 se muestra el diseño de la base de datos por medio de un diagrama, entidad relación. En este diagrama se puede observar que las principales entidades son: usuario, producto, carrito y venta, además también se puede observar las relaciones entre cada uno de ellos.

4.2.2 Diagrama relacional de la base de datos

En la figura 4.2 se muestra el diseño de la base de datos de la tienda virtual, la base de datos tiene varios objetos, pero las entidades principales son: usuario y producto.

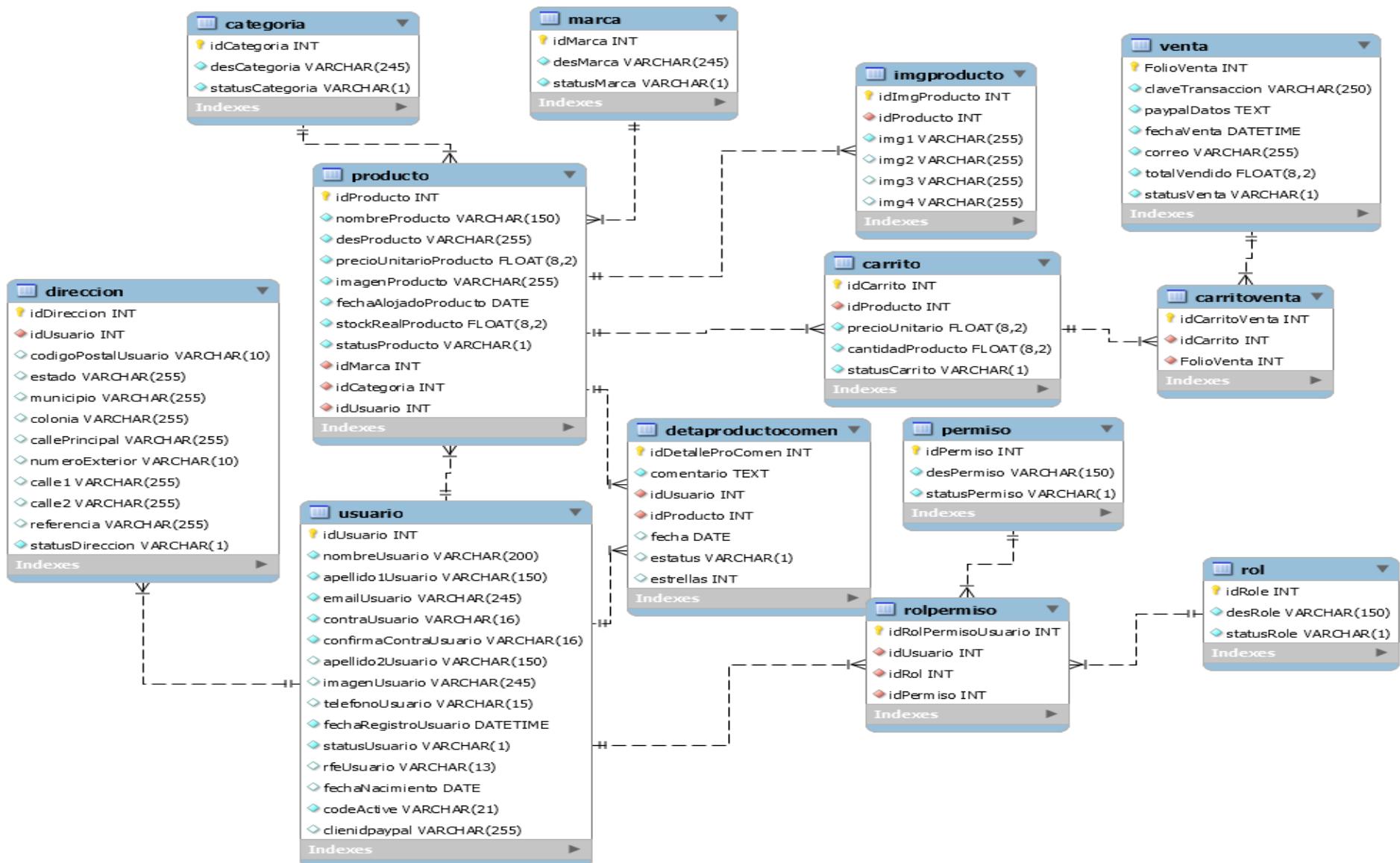


Figura 4.2 Diagrama relacional de la base de datos

4.2.3 Diagrama de estados

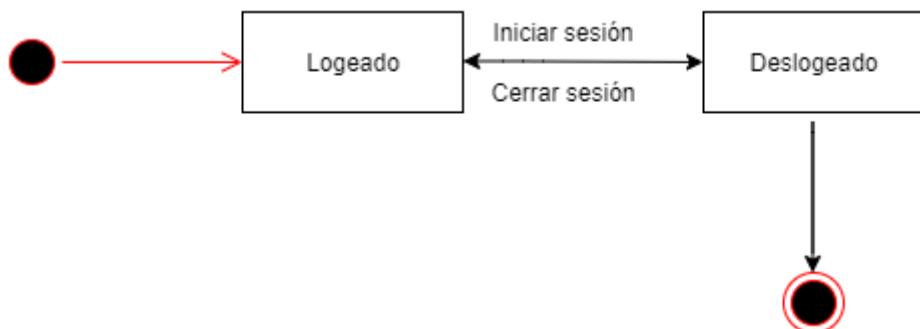


Figura 4.3 Estado del usuario

En la figura 4.3 se muestran los posibles estados de un usuario, los cuales son: Logeado y deslogado.

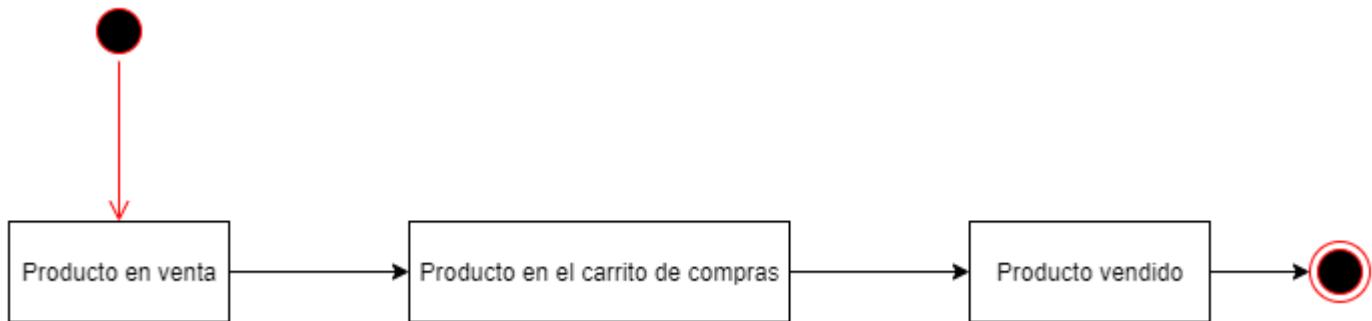


Figura 4.4 Diagrama de estado del producto.

El producto puede tener 3 estados, los cuales son: producto en venta, producto en el carrito de compras y producto vendido, tal y como se muestra en la figura 4.4.

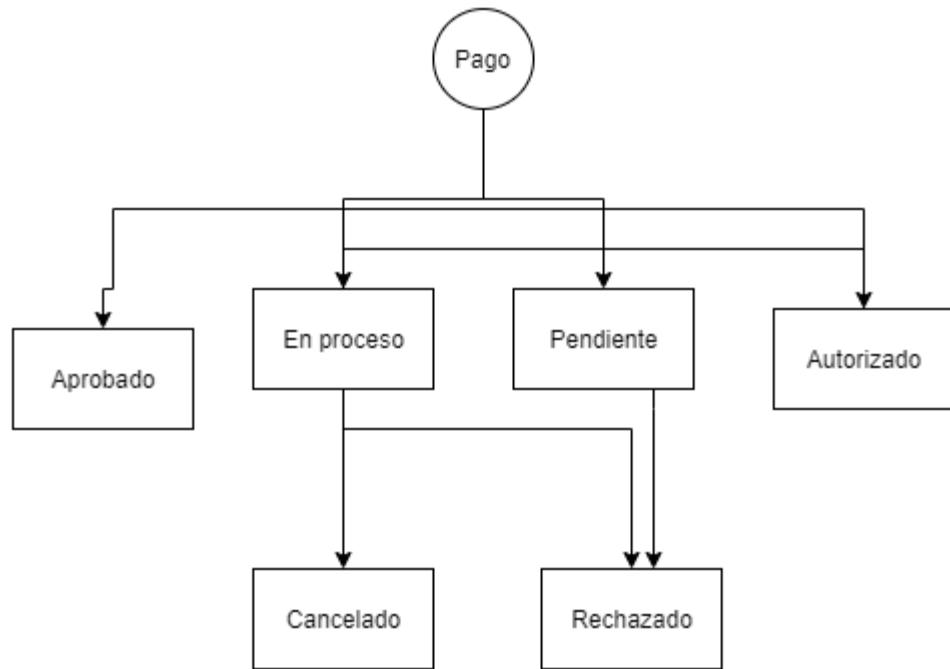


Figura 4.5 Estados del pago

La entidad pago, puede tener diferentes estados, los cuales se muestran en la figura 4.5. Los estados del pago son los siguientes:

- En proceso.
- Pendiente.
- Autorizado.
- Aprobado.
- Rechazado.
- Cancelado.

4.2.4 Diagramas de interacción

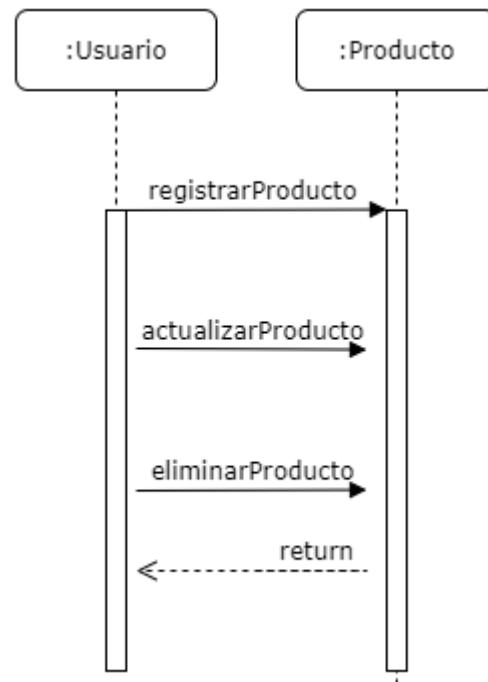


Figura 4.6 Interacción entre el objeto usuario y producto

En la figura 4.6 se muestra la interacción entre los objetos: usuario y producto. Un usuario puede registrar un producto, actualizarlo y eliminarlo.

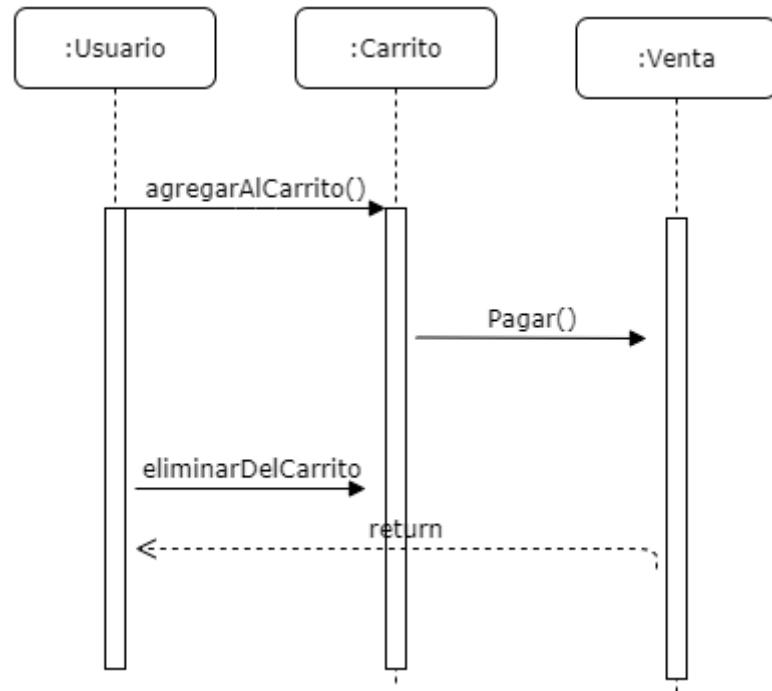


Figura 4.7 Interacción del flujo de compra

Para realizar el proceso de compra de un producto, el usuario debe interactuar con el objeto carrito, a través de la acción: agregar al carrito, una vez agregado al carrito, el usuario puede proceder al pago, interactuando con el objeto venta.

4.2.5 Diagrama de componentes

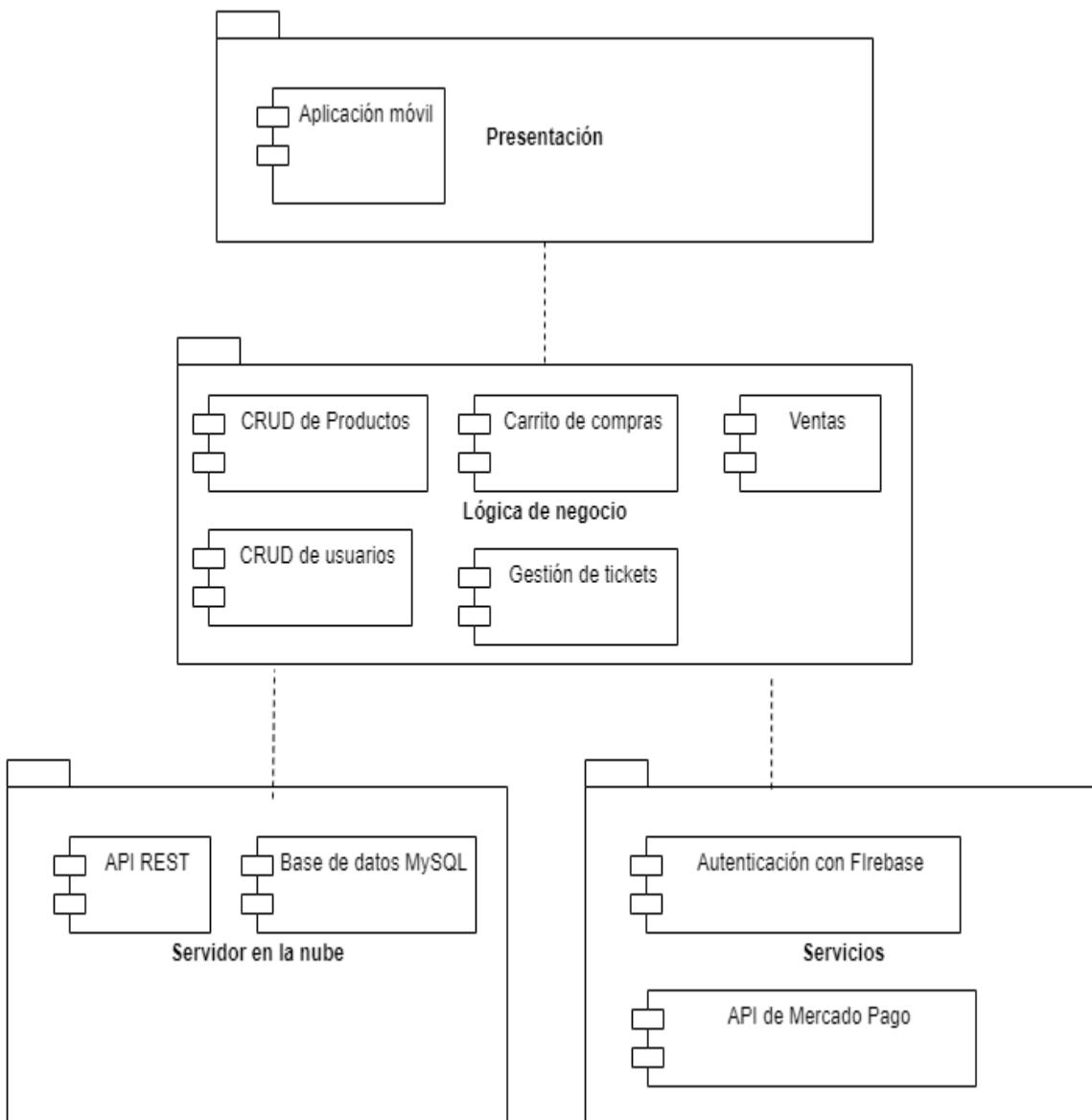


Figura 4.8 Diagrama de componentes de la tienda virtual

En la figura 4.8 se muestra el diagrama de componentes para describir los módulos del sistema. En la capa de presentación se encuentra la aplicación móvil, luego en el segundo nivel se encuentra la capa de la lógica de negocio, en él se encuentran los módulos que definen la funcionalidad de la aplicación móvil, dichos módulos son CRUD de producto, de usuarios, funciones para controlar la tabla de carrito de compras, gestión de tickets y ventas.

4.2.6 Diagrama de despliegue

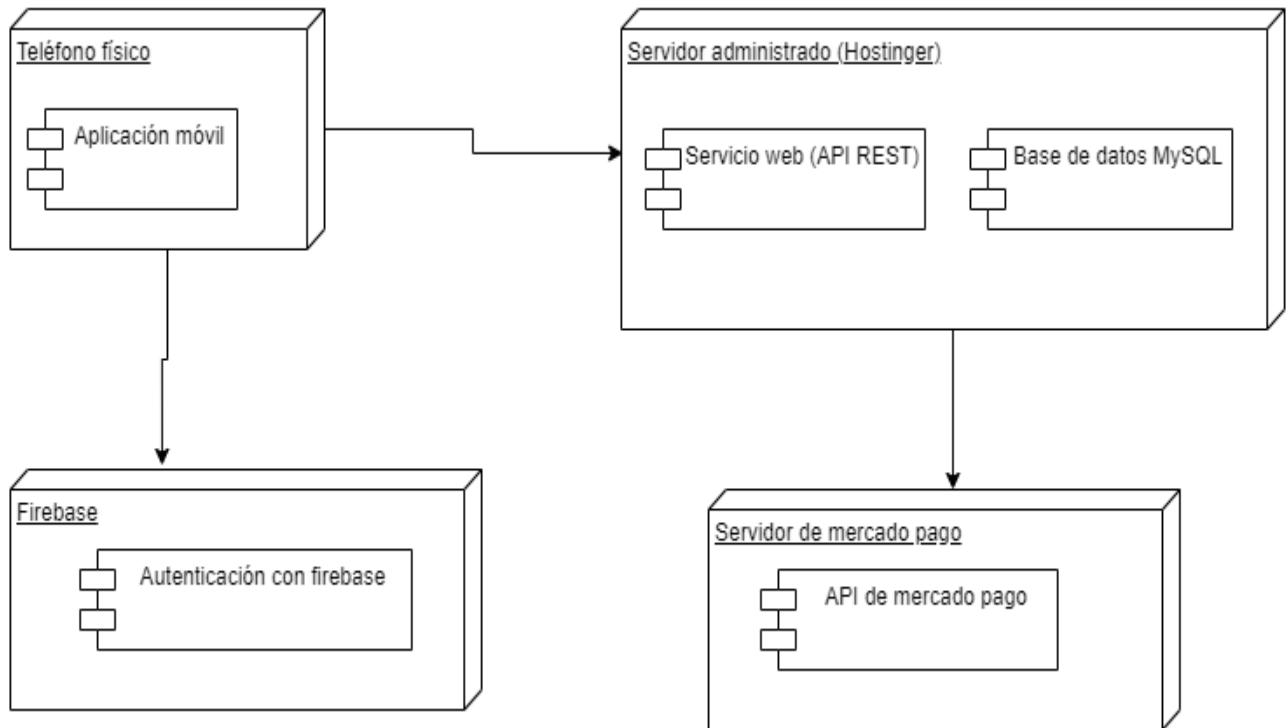


Figura 4.9 Diagrama de despliegue de la tienda virtual

La arquitectura de la tienda virtual se describe a través del diagrama de despliegue que se muestra en la figura 4.9, la aplicación móvil se instala en un Smartphone físico, la base de datos y la API REST, se encuentran en un servidor administrado por el proveedor Hostinger, a su vez la aplicación móvil se conecta a API'S de terceros las cuales son la API de firebase y Mercado pago.

4.2.7 Diagramas de casos de uso

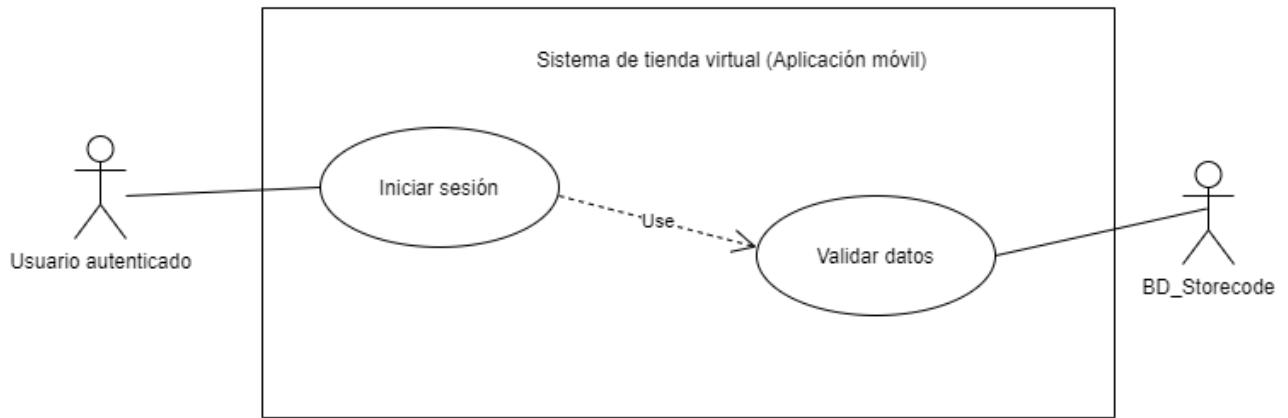


Figura 4.10 Caso de uso "Iniciar sesión"

En la siguiente tabla se encuentra la descripción textual del caso de uso “Iniciar sesión”

Nombre:	Iniciar sesión
Autor:	José Jiménez
Descripción: Le permite al usuario iniciar sesión en la aplicación móvil.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: El usuario debe estar registrado en la base de datos.	
Flujo Normal:	
1. El usuario autenticado pulsa sobre el botón “Iniciar sesión”. 2. La aplicación móvil muestra un formulario para que el usuario ingrese sus credenciales. 3. El usuario autenticado ingresa su email y contraseña para hacer login. 4. El sistema hace una consulta para verificar que los datos sean correctos. 5. Si el email y contraseña son correctos, el usuario inicia sesión en la aplicación.	
Flujo alternativo:	
4. A El sistema hace una consulta para verificar los datos, si los datos no son correctos, se le avisa al actor, permitiéndole que lo corrija	
Poscondiciones:	
El usuario ha sido autenticado se le muestra un nuevo inicio, donde ya puede realizar compras y ventas.	

Tabla 4-1 Descripción textual del caso de uso “Iniciar sesión”.

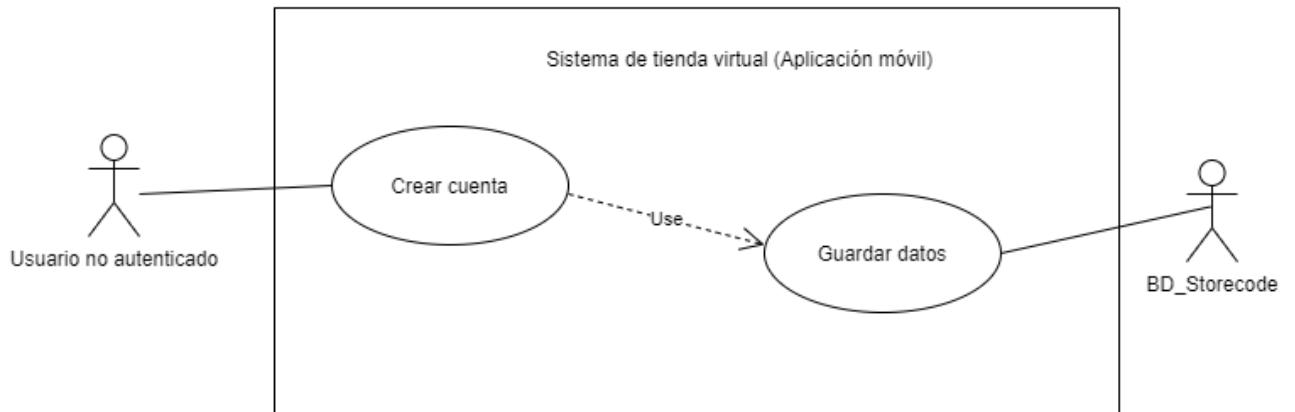


Figura 4.11 Caso de uso "Crear cuenta"

Nombre:	"Crear cuenta"
Autor:	José Jiménez
Descripción: Le permite al usuario no autenticado crear una cuenta	
Actores: Usuario no autenticado/ BD_Storecode.	
Precondiciones: Ninguna	
Flujo Normal:	
1. El usuario no autenticado pulsa sobre el botón "Crear cuenta". 2. El Sistema muestra un formulario para que el actor introduzca su nombre, apellido paterno, correo y contraseña. 3. El usuario da click en el botón "Crear cuenta". 4. El sistema verifica los datos y los almacena. 5. El sistema dirige al usuario hacia la pantalla "Iniciar sesión".	
Flujo alternativo:	
4. A El sistema verifica los datos, si los datos no son correctos, se le avisa al actor, permitiéndole que lo corrija	
Poscondiciones:	
La cuenta se crea con éxito y el usuario podrá iniciar sesión en la aplicación.	

Tabla 4-2 Descripción textual del caso de uso "Crear cuenta".

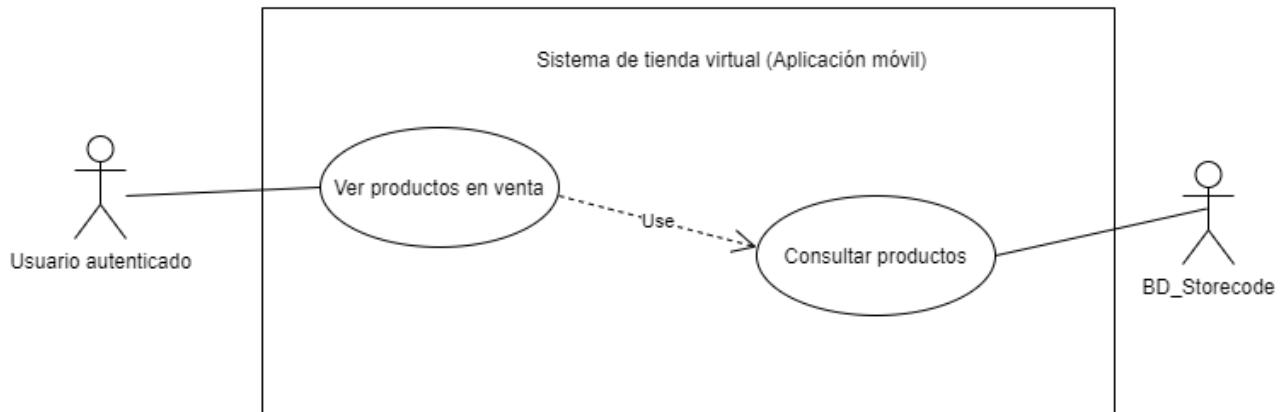


Figura 4.12 Caso de uso "Ver productos en venta"

Nombre:	"Ver productos en venta"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado ver todos los productos en venta que no han sido publicados por el mismo	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber iniciado sesión en la aplicación.	
Flujo Normal:	
1. El usuario autenticado navega hacia la pantalla de "Inicio", desde el botón de navegación. 2. El sistema hace una consulta de los productos a la base de datos. 3. En la pantalla de "Inicio", se muestran los productos en venta.	
Flujo alternativo:	
2. A. El sistema hace una consulta de los productos a la base de datos, si ocurre un error de conexión, se le avisará al actor para que verifique su conexión.	
Poscondiciones:	
El actor puede ver los productos que se encuentran en venta.	

Tabla 4-3 Descripción textual del caso de uso "Ver productos en venta"

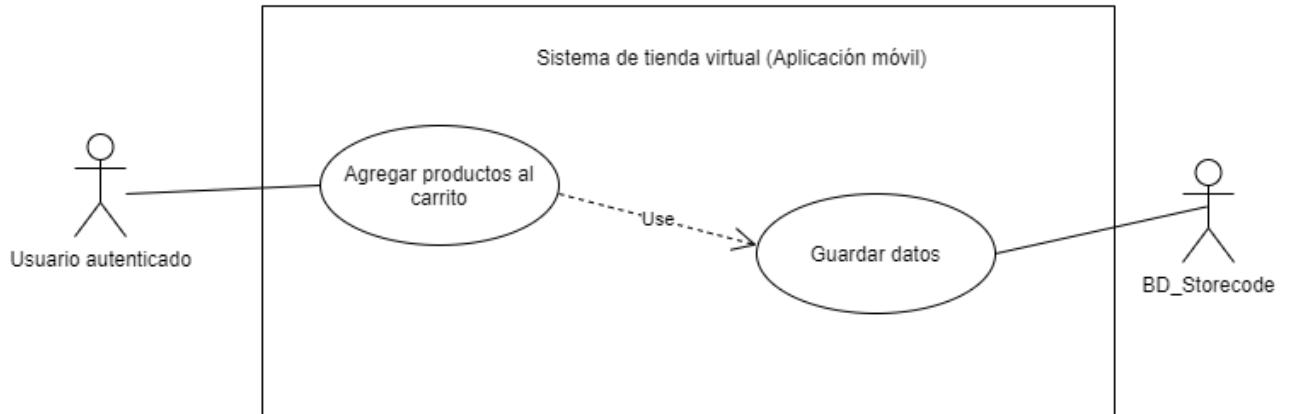


Figura 4.13 Caso de uso "Agregar productos al carrito"

Nombre:	"Agregar productos al carrito"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado agregar un producto a su carrito de compras.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber iniciado sesión en la aplicación.	
Flujo Normal:	
1. El usuario autenticado da click en un producto. 2. El sistema muestra la pantalla de "Detalle del producto". 3. El usuario da click en el botón "Agregar al carrito". 4. El sistema guarda los datos en la base de datos.	
Flujo alternativo:	
4. A. El sistema guarda los datos en la base de datos, si ocurre un error de conexión, se le avisará al actor para que verifique su conexión.	
Poscondiciones:	
El actor puede ver los productos que ha agregado a su carrito de compras.	

Tabla 4-4 Descripción textual del caso de uso "Agregar productos al carrito"

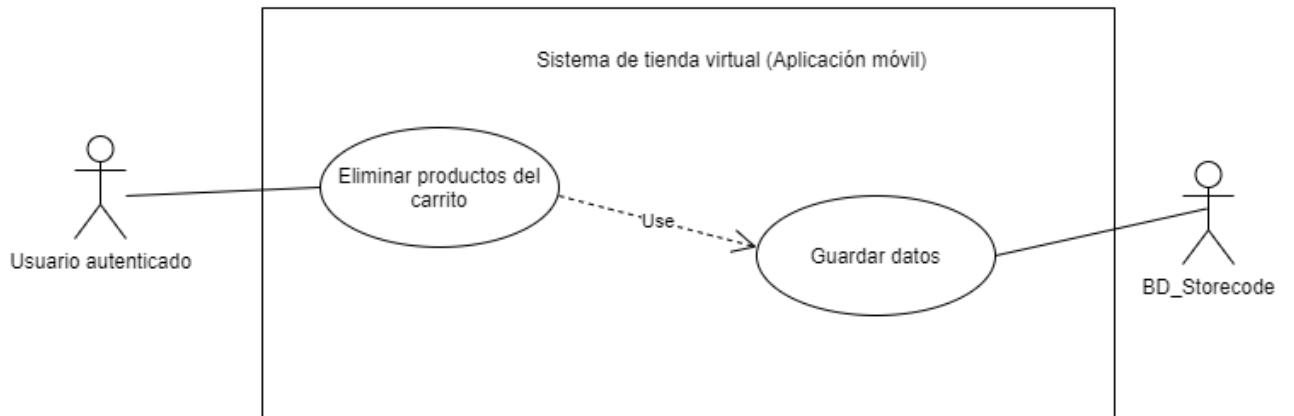


Figura 4.14 Caso de uso "Eliminar productos del carrito"

Nombre:	"Eliminar productos del carrito"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado eliminar productos de su carrito de compras.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber agregado al menos un producto a su carrito de compras.	
Flujo Normal:	
1. El usuario autenticado navega hacia la pantalla "Carrito" desde el botón de navegación. 2. El sistema realiza una consulta en la base de datos de los productos que el actor ha agregado a su carrito. 3. Se muestran los productos en la pantalla "Carrito".	
Flujo alternativo:	
4. A. El sistema guarda los datos en la base de datos, si ocurre un error de conexión, se le avisará al actor para que verifique su conexión.	
Poscondiciones:	
El actor puede ver los productos que ha agregado a su carrito de compras.	

Tabla 4-5 Descripción textual del caso de uso "Eliminar productos del carrito"

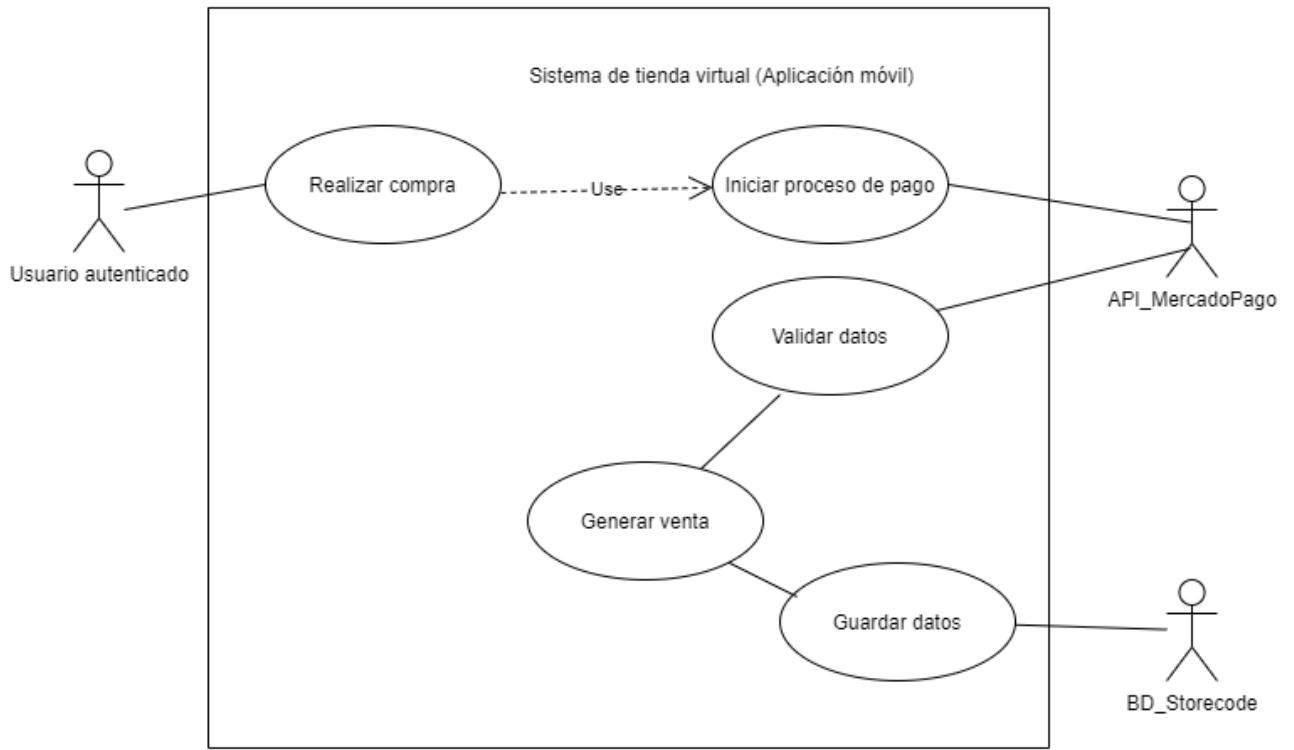


Figura 4.15 Caso de uso "Realizar compra"

Nombre:	"Realizar compra"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado realizar la compra de un producto.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber agregado al menos un producto a su carrito de compras.	
Flujo Normal:	
1. El usuario autenticado da click en el botón "Pagar" de la pantalla de "Carrito".	
2. El sistema muestra una interfaz donde puede elegir el método de pago de su preferencia (tarjeta de crédito y débito).	
3. El sistema muestra una interfaz para que el usuario pueda capturar los datos de su tarjeta.	
4. La API de mercado pago valida los datos de la tarjeta.	
5. La API de mercado pago le avisa al usuario que el pago fue exitoso.	
6. El sistema almacena los datos de la venta	
Flujo alternativo:	
1. 4. A. La API de mercado pago valida los datos de la tarjeta, si los datos no son correctos, se le avisa al usuario y se le permite intentar con otra tarjeta.	

Poscondiciones:

El actor completa exitosamente su compra.

Tabla 4-6 Descripción textual del caso de uso "Realizar compra".

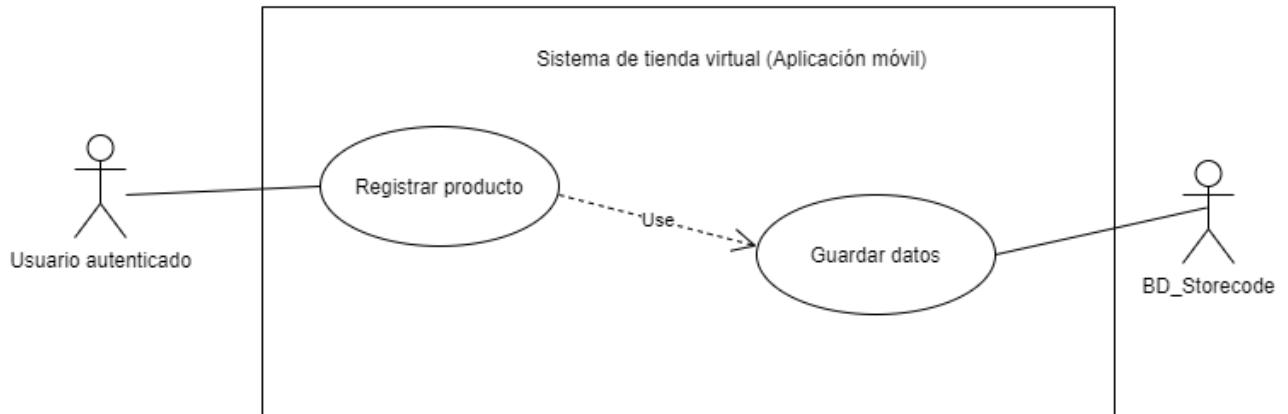


Figura 4.2.16 Caso de uso "Registrar producto"

Nombre:	"Registrar producto"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado registrar un producto para ponerlo en venta.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Estar en la pantalla de "Perfil".	
Flujo Normal:	
1. El usuario autenticado da click en el botón "Vender productos". 2. El sistema muestra una interfaz para que el actor ingrese los datos solicitados del producto. 3. El usuario da click en el botón "Guardar producto". 4. El sistema valida los datos y los almacena.	
Flujo alternativo:	
4. A. El sistema valida los datos, si los datos no son válidos, se le avisa al usuario y se le permite corregirlo.	
Poscondiciones:	
El producto queda almacenado en la base de datos.	

Tabla 4.2.8 Descripción textual del caso de uso "Registrar producto"

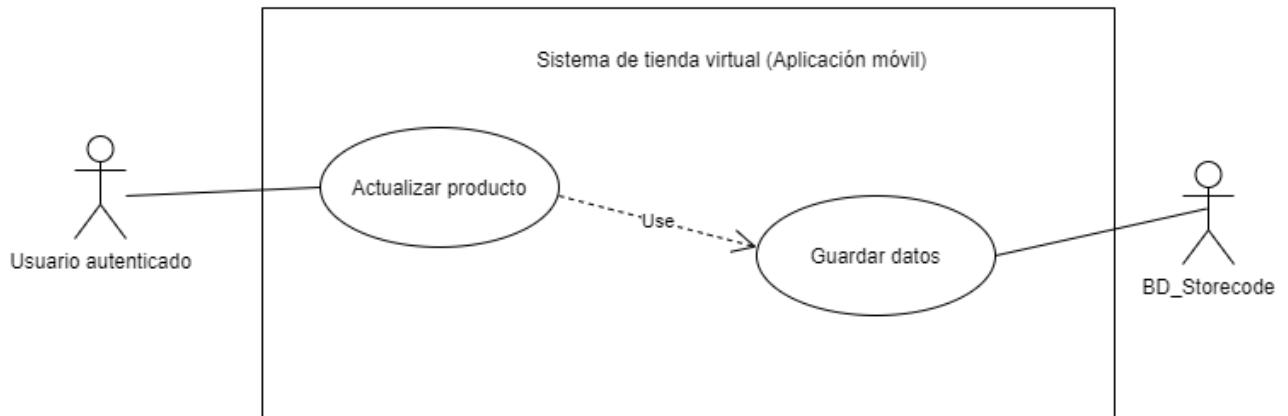


Figura 4.16 Caso de uso "Actualizar producto"

Nombre:	"Actualizar producto"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado actualizar los datos de un producto.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber registrado por lo menos un producto.	
Flujo Normal:	
1. El usuario autenticado da click en la opción “Productos en venta”. 2. Se muestra una interfaz que lista los productos que han sido registrados por el actor. 3. El usuario da click en el botón “Editar”. 4. El sistema muestra una interfaz con los datos actuales del producto, permitiéndole al actor editarlos. 5. El usuario da click en el botón “Guardar”. 6. El sistema valida y actualiza los datos del producto.	
Flujo alternativo:	
6. A. El sistema valida los datos, si los datos no son válidos, se le avisa al usuario y se le permite corregirlo.	
Poscondiciones:	
Los cambios realizados, quedan guardados.	

Tabla 4-7 Descripción textual del caso de uso "Actualizar producto"

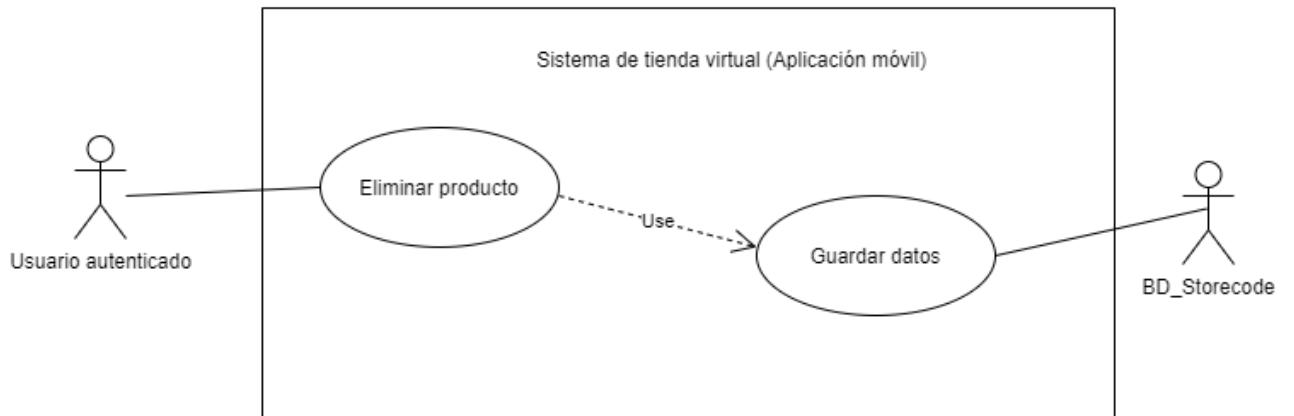


Figura 4.17 Caso de uso "Eliminar producto"

Nombre:	"Eliminar producto"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado eliminar un producto.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber registrado por lo menos un producto.	
Flujo Normal:	
1. El usuario autenticado da click en la opción “Productos en venta”. 2. Se muestra una interfaz que lista los productos que han sido registrados por el actor. 3. El usuario da click en el botón “Eliminar”. 4. El sistema elimina ese producto de la base de datos.	
Flujo alternativo:	
3.A Si el usuario decide no eliminar el producto, regresa a la pantalla de perfil.	
Poscondiciones:	
Los cambios realizados, quedan guardados.	

Tabla 4-8 Descripción textual del caso de uso "Eliminar producto"

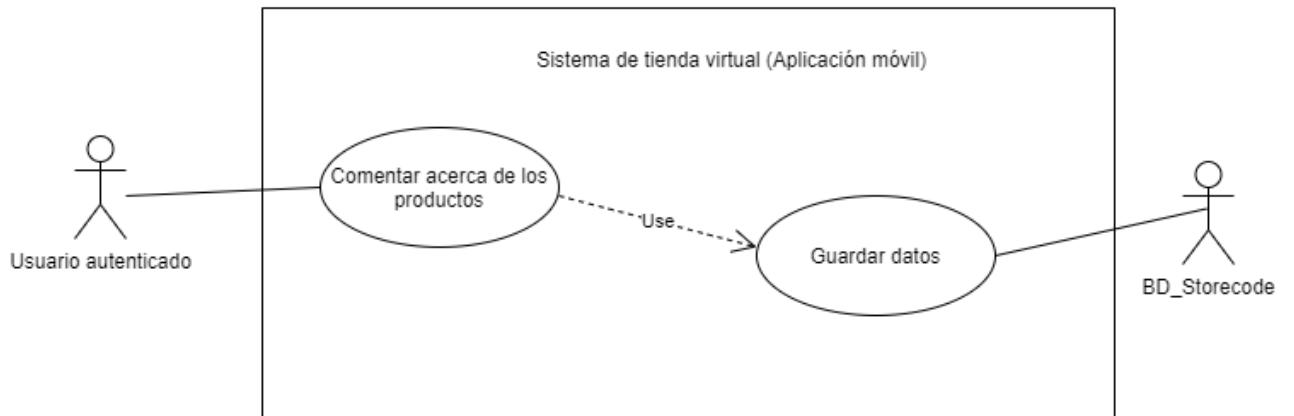


Figura 4.18 Caso de uso "Comentar productos"

Nombre:	"Comentar producto"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado ver los comentarios acerca de un producto.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber iniciado sesión en la aplicación.	
Flujo Normal:	
1. El usuario autenticado da click en un producto. 2. El sistema muestra la pantalla de "Detalle del producto". 3. El actor escribe un comentario y da click en el botón "Comentar". 4. El sistema almacena el comentario en la base de datos. 5. El comentario se muestra en la pantalla "Detalle del producto".	
Flujo alternativo: 4.A El sistema guarda los datos en la base de datos, si ocurre un error de conexión, se le avisará al actor para que verifique su conexión.	
Poscondiciones: El comentario queda almacenado.	

Tabla 4-9 Descripción textual del caso de uso "Comentar producto"

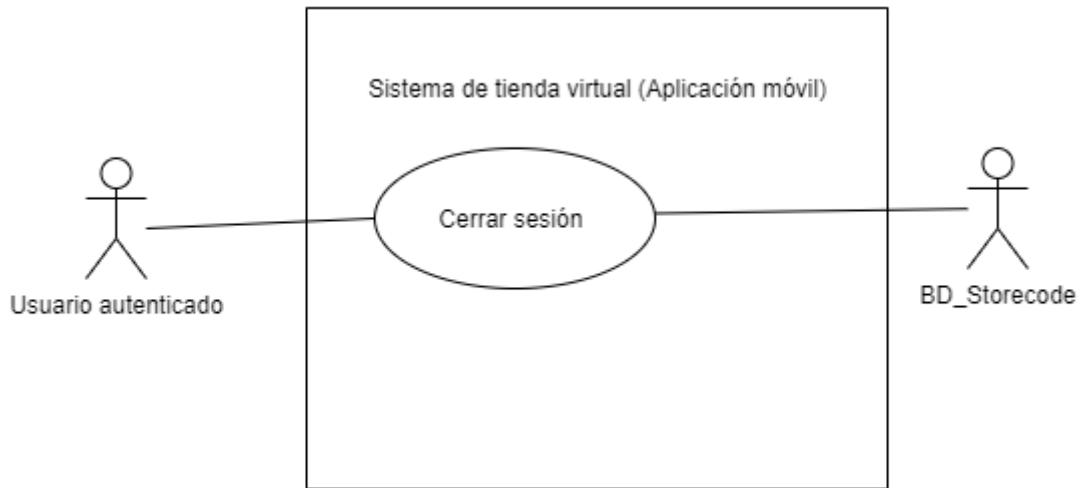


Figura 4.19 Caso de uso "Cerrar sesión"

Nombre:	"Cerrar sesión"
Autor:	José Jiménez
Descripción: Le permite al usuario autenticado cerrar sesión en la aplicación.	
Actores: Usuario autenticado/ BD_Storecode.	
Precondiciones: Haber iniciado sesión en la aplicación.	
Flujo Normal:	
1. El usuario autenticado da click en el botón "Cerrar sesión".	
2. El sistema cierra sesión y muestra el menú ordinario.	
Flujo alternativo:	
Poscondiciones:	
El usuario cierra sesión en la aplicación.	

Tabla 4-10 Descripción textual del caso de uso "Cerrar sesión"

4.2.8 Diseño de las interfaces de usuario

A continuación, se muestra el diseño de las diferentes interfaces de usuario, estos diseños fueron creados utilizando la herramienta web Balsamiq Mockups.

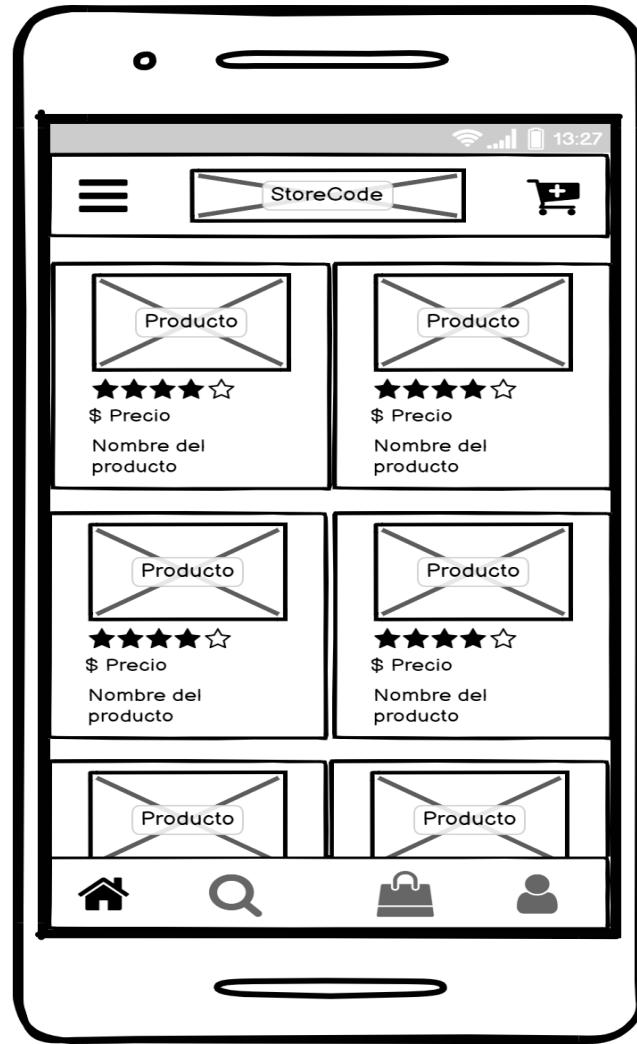


Figura 4.20 Diseño de la pantalla de Inicio

En la figura 4.20 se muestra el diseño de la pantalla de inicio, esta será la primera pantalla que verá el usuario cuando abra la aplicación.

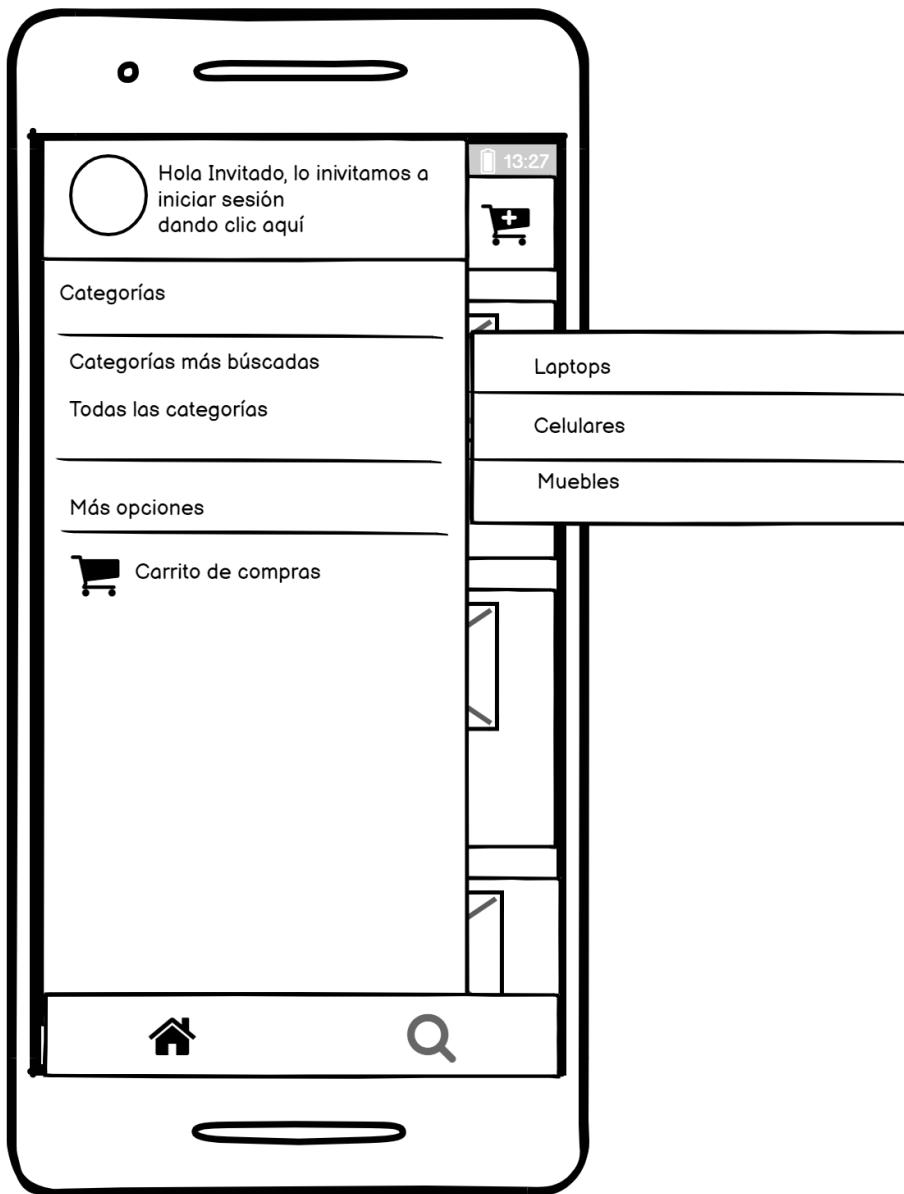


Figura 4.21 Diseño del Menú hamburguesa

En la figura 4.21 se muestra un menú que se despliega cuando el usuario da clic sobre el ícono del menú hamburguesa.

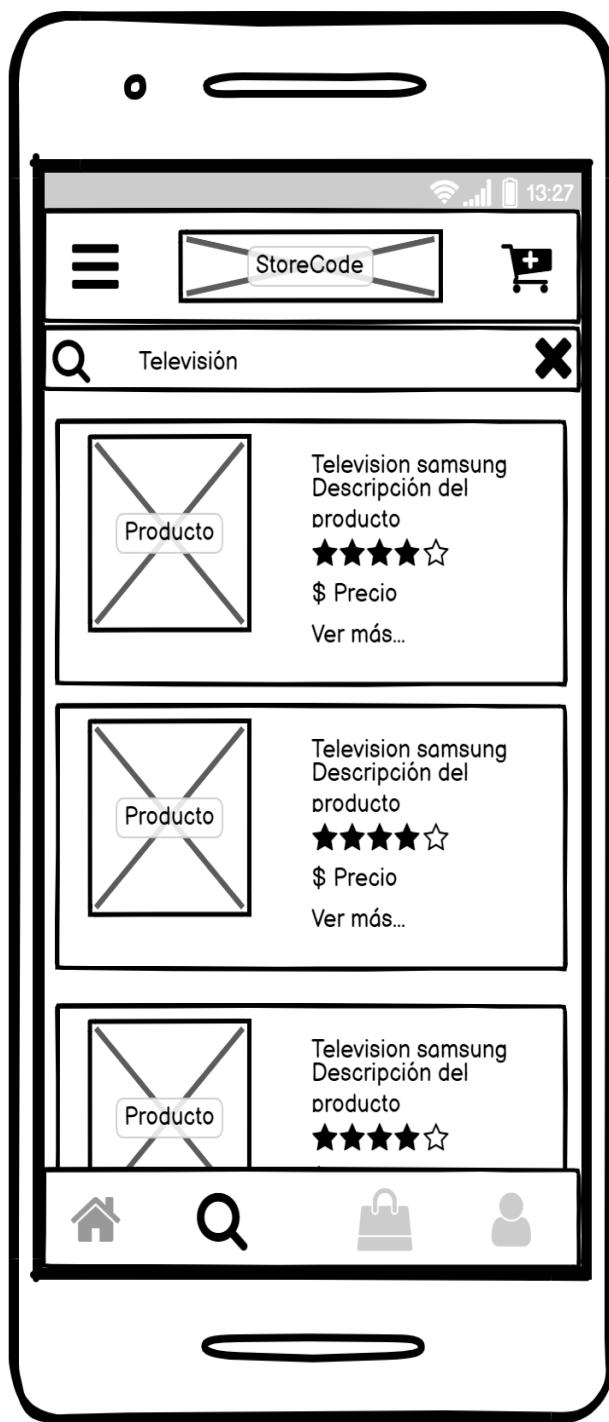


Figura 4.22 Diseño de Pantalla de búsqueda

En la figura 4.22 se muestra el diseño de la pantalla de búsqueda. Cuando el usuario requiera realizar una búsqueda, le aparecerá esta pantalla.

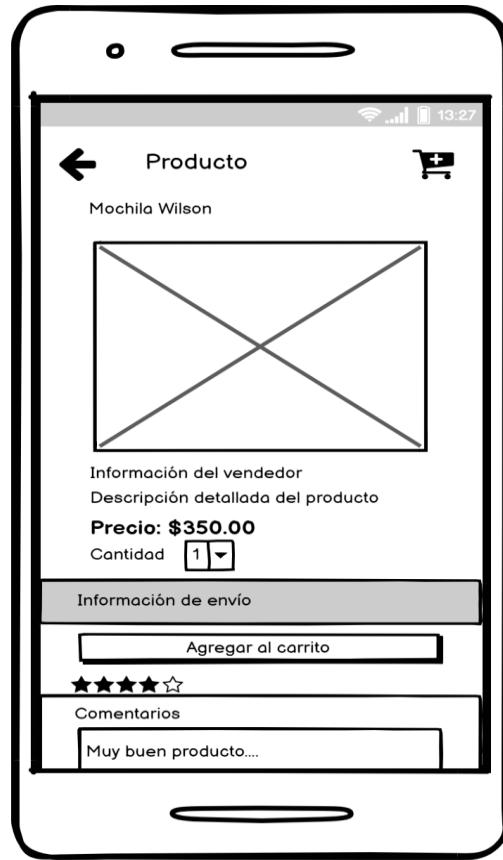


Figura 4.23 Diseño de Pantalla de detalle del producto

Cuando el usuario seleccione un producto, se le abrirá la pantalla de detalle de producto, el diseño de esta pantalla se muestra en la figura 4.23.

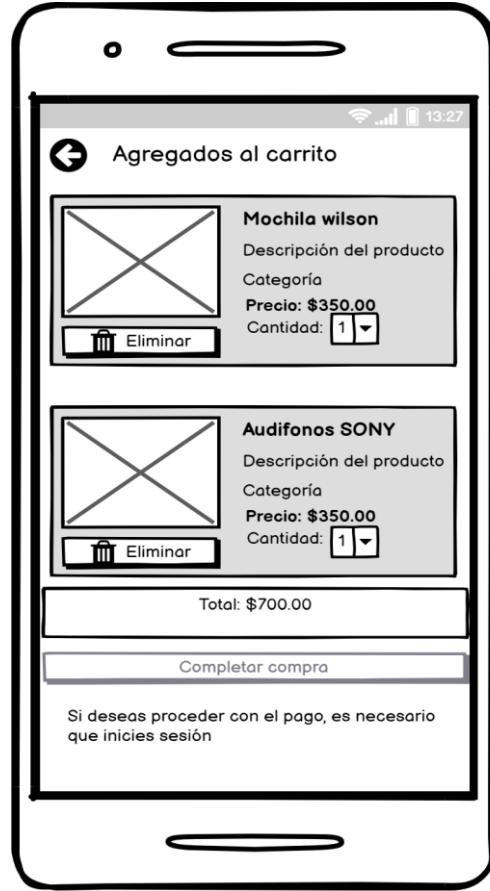


Figura 4.24 Diseño de Pantalla de carrito de compras

Si el usuario agrega un producto al carrito de compras, este le aparecerá en la pantalla “Carrito de compras” tal y como se muestra en la figura 4.24.



Figura 4.25 Diseño de Pantalla de Inicio de sesión

Si el usuario no ha iniciado sesión y quiere realizar una compra o vender un producto, deberá iniciar sesión, para lo cual le aparecerá una pantalla como la de la figura 4.25.

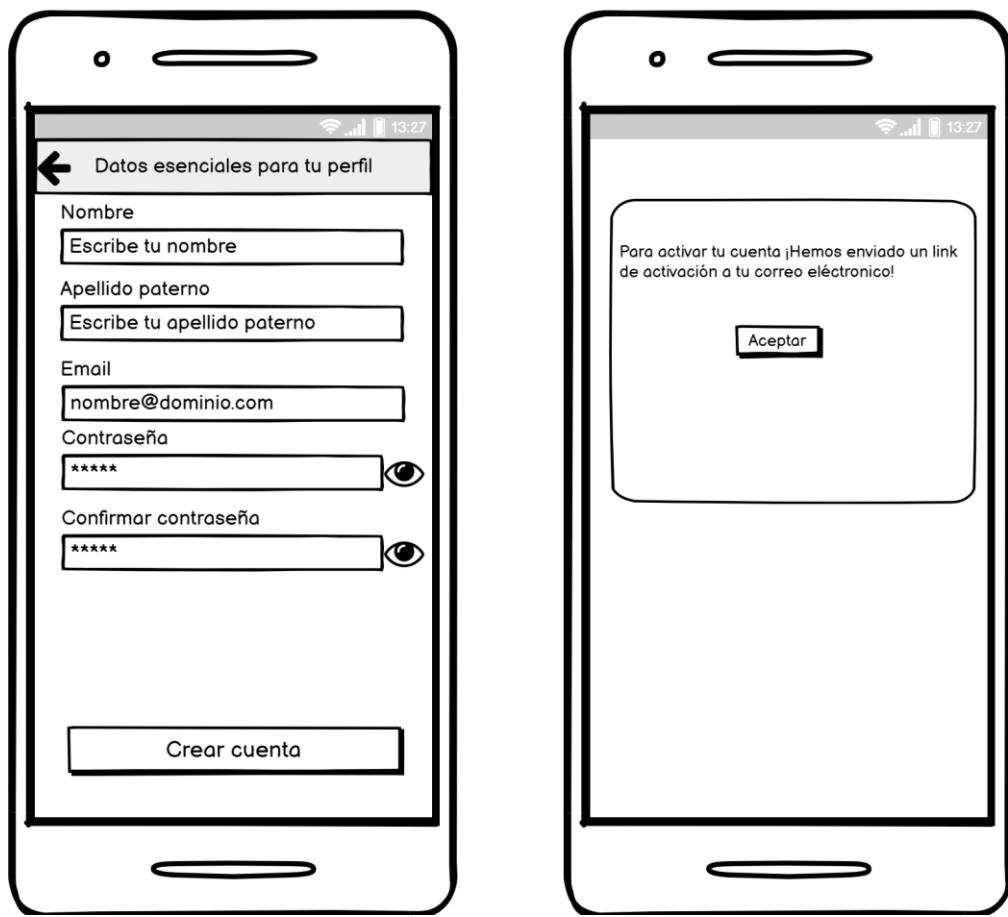


Figura 4.26 Diseño de Pantalla de registro de usuario

En la figura 4.26 se muestra la pantalla para que el usuario pueda registrarse, si este aún no tiene una cuenta.

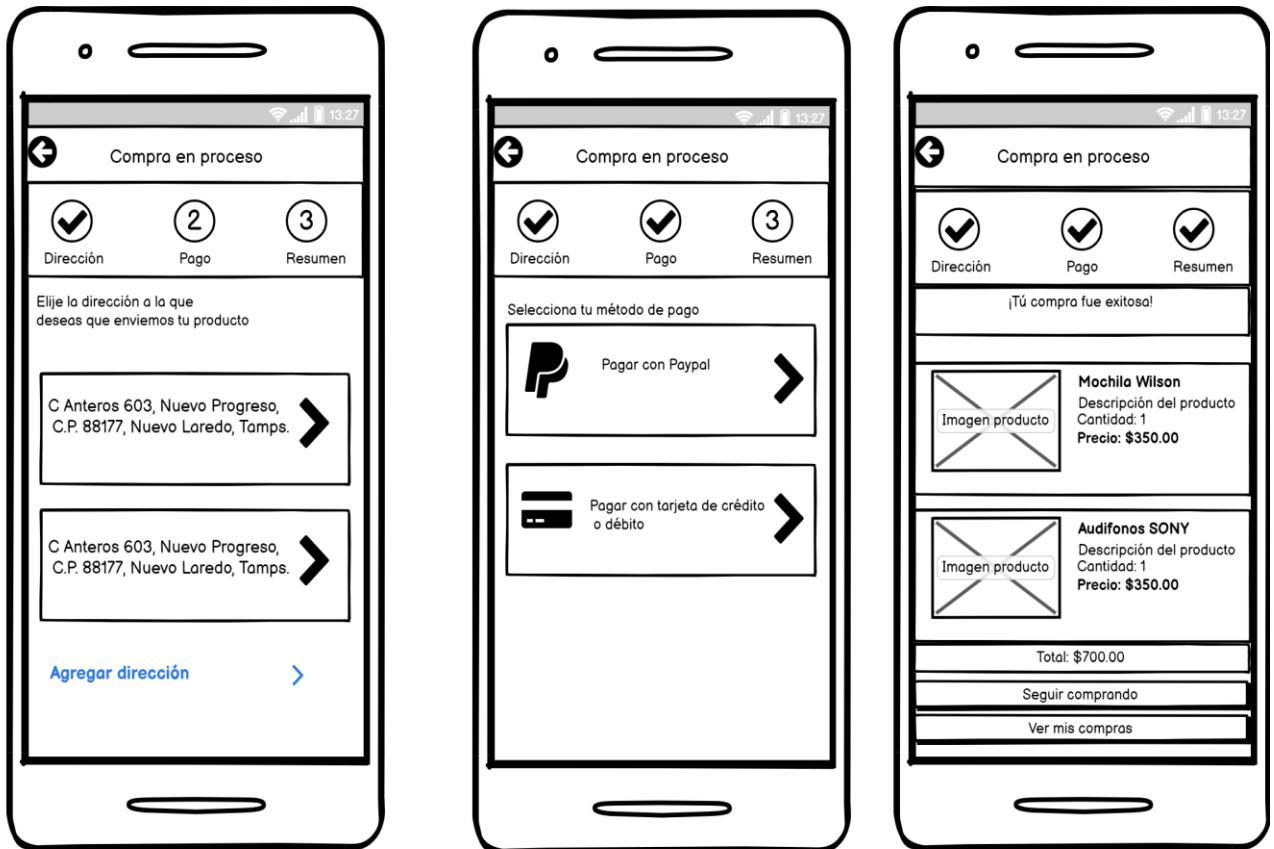


Figura 4.27 Diseño de Flujo de pantallas para realizar una compra

Cuando el usuario ya inicio sesión y está listo para realizar una compra deberá seguir los pasos que se muestran en la sucesión de pantallas de la figura 4.27. Primero deberá elegir o agregar una dirección, después deberá seleccionar un método de pago y por último le aparecerá un resumen de la compra que realizó.

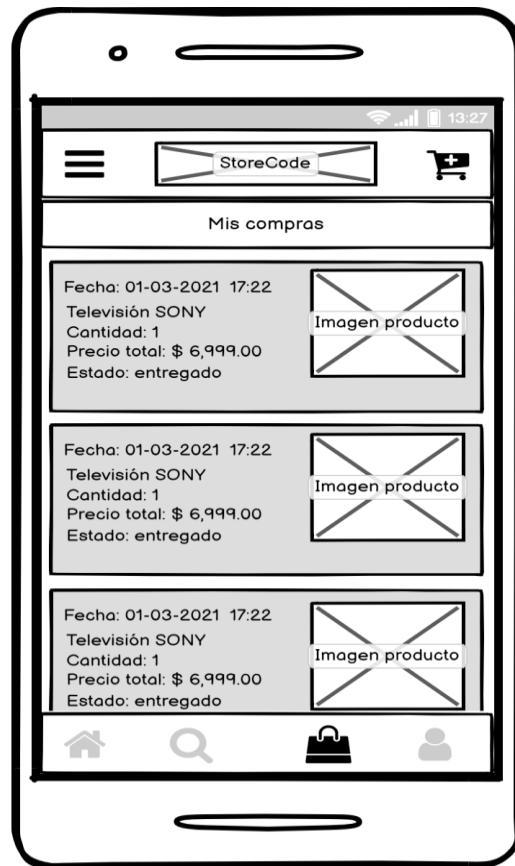


Figura 4.28 Diseño de Pantalla de "Mis compras"

Si el usuario navega al ícono de compras del botón de navegación, le aparecerá la pantalla de "Mis compras" tal y como se muestra en la figura 4.28.

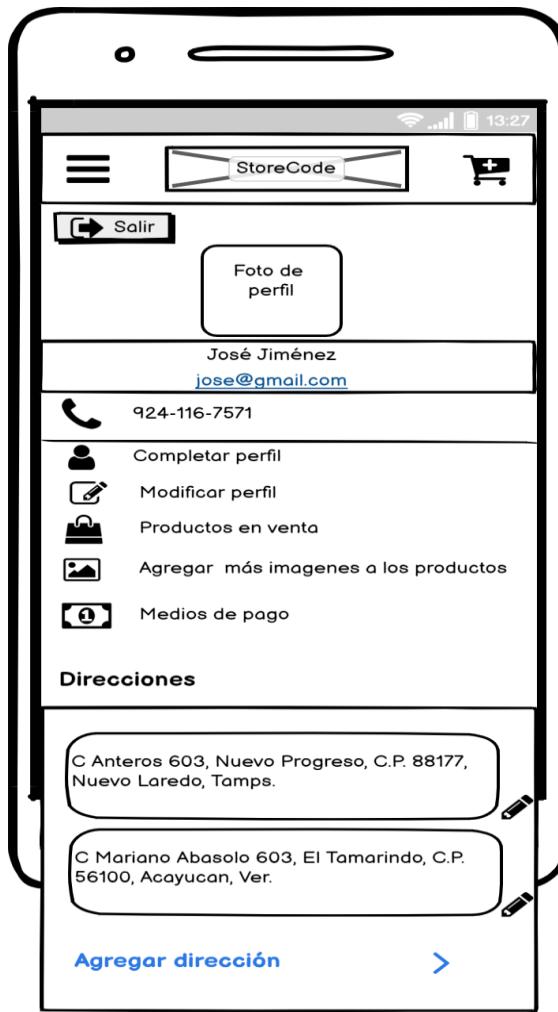


Figura 4.29 Diseño de Pantalla de perfil

Si el usuario navega hacia el ícono de perfil del botón de navegación, le aparecerá una pantalla como la que se muestra en la figura 4.29.

Si da clic en las opciones de su perfil, se le abrirá el formulario correspondiente para completar dicha acción, ya sea para completar su perfil, modificarlo, agregar más imágenes del producto y agregar dirección.

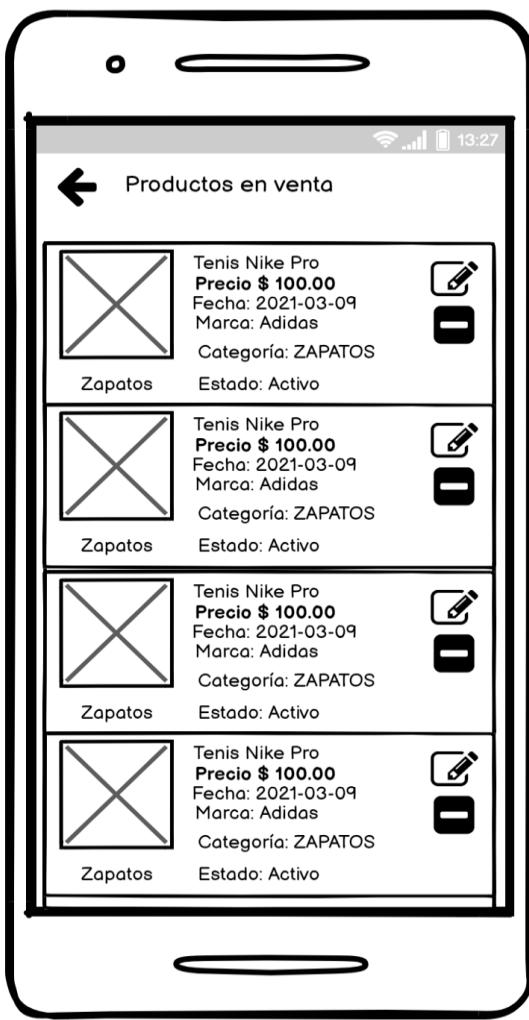


Figura 4.30 Diseño de Pantalla de productos en venta

Si el usuario da clic en la opción “Productos en venta” de la pantalla de perfil, le aparecerá una pantalla como la que muestra en la figura 4.30.

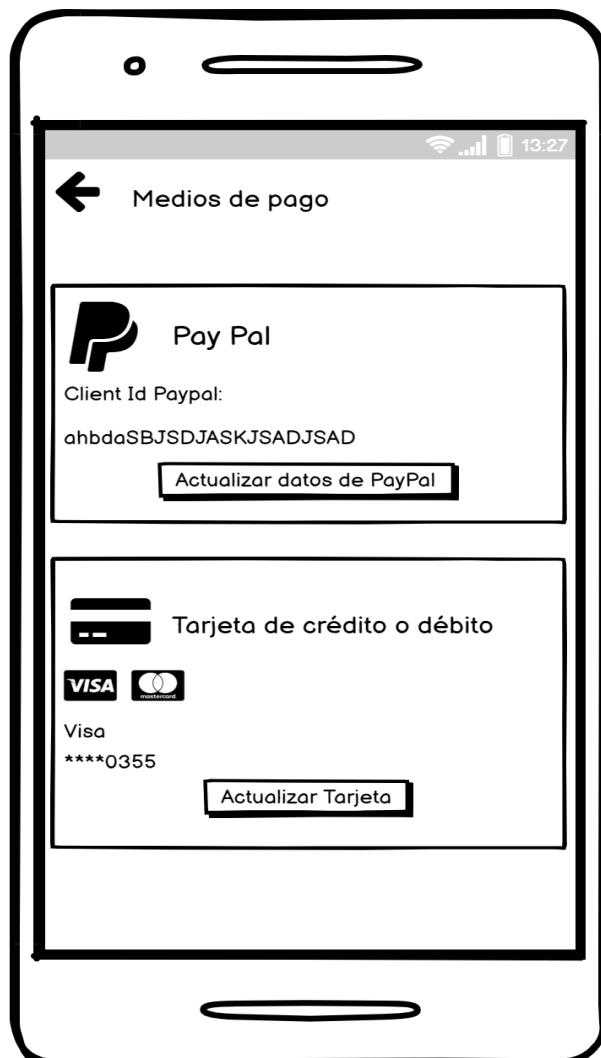


Figura 4.31 Diseño de Pantalla de "Medios de pago"

Si el usuario da clic en la opción “Medios de pago” de la pantalla de perfil le aparecerá una pantalla como la de la figura 4.31.

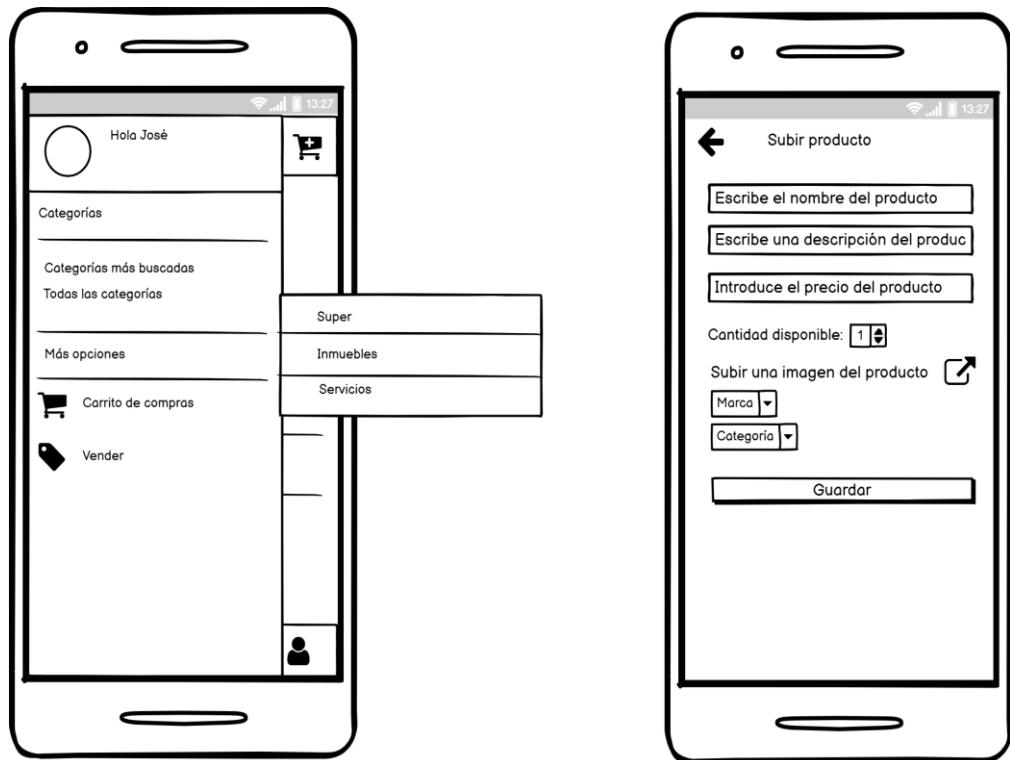


Figura 4.32 Diseño de Menú y formulario de subir producto

Cuando el usuario ya ha iniciado sesión le aparecerá la opción de “vender” en el menú y si da clic en esa opción le aparecerá el formulario para subir un producto, así como se muestra en la secuencia de imágenes de la figura 4.32.

4.3 Implementación de la base de datos

En la tabla 4-11 se muestra el script de la base de datos.

Tabla 4-11 Script de la base de datos

```
-- phpMyAdmin SQL Dump
-- version 4.9.5
-- https://www.phpmyadmin.net/
-- Servidor: 127.0.0.1:3306
-- Tiempo de generación: 27-02-2021 a las 03:35:14
-- Versión del servidor: 10.4.15-MariaDB
-- Versión de PHP: 7.2.34
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
-- Base de datos: `u743223069_storecode`
--
DELIMITER $$
-- Procedimientos
--
CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psActDatFalUsu`(`PidUAct` INT) BEGIN
    SELECT
        usuario.idUsuario,
        usuario.nombreUsuario,
        usuario.apellido1Usuario,
        usuario.apellido2Usuario,
        usuario.contraUsuario,
        usuario.confirmaContraUsuario,
        usuario.direccionUsuario,
        usuario.codigoPostalUsuario,
        usuario.telefonoUsuario,
        usuario.rfeUsuario,
        usuario.fechaNacimiento
    FROM
        usuario
    WHERE usuario.idUsuario = PidUAct;
END$$
CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psActualizarProdcto`(`PnombreActP` VARCHAR(150), `PdescripActP` VARCHAR(255), `PprecioUActP` FLOAT(8,2), `PimagenActP` VARCHAR(255), `PstockActP` FLOAT(8,2), `PmarcaActP` INT, `PcategoriaActP` INT, `PidProductoActP` INT) BEGIN
    UPDATE producto SET producto.nombreProducto = PnombreActP,
        producto.desProducto = PdescripActP,
        producto.precioUnitarioProducto = PprecioUActP,
        producto.imagenProducto = PimagenActP,
        producto.stockRealProducto = PstockActP,
        producto.idMarca = PmarcaActP,
        producto.idCategoria = PcategoriaActP
    WHERE producto.idProducto = PidProductoActP AND producto.statusProducto = '1';
END$$
```

```

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psActulmgs` (`PidUsuarioActProImg` INT)
BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.imagenProducto,
        imgproducto.img1,
        imgproducto.img2,
        imgproducto.img3,
        imgproducto.img4
    FROM producto
    INNER JOIN imgproducto ON producto.idProducto = imgproducto.idProducto
    WHERE producto.idUsuario = PidUsuarioActProImg AND statusProducto= '1' AND
    producto.stockRealProducto >0;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psBusIdProduCU` (`PidProUmC` INT)
BEGIN
    SELECT producto.idUsuario FROM producto
    WHERE producto.idProducto = PidProUmC;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psCateAct` () BEGIN
    SELECT categoria.idCategoria,categoria.desCategoria FROM categoria WHERE categoria.statusCategoria
    = '1';
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psConsultaUsuario` (`PdesContra` VARCHAR(16), `PdesEmail` VARCHAR(245)) BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "A OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT usuario.emailUsuario, usuario.contraUsuario FROM usuario WHERE
    usuario.contraUsuario = PdesContra AND usuario.emailUsuario = PdesEmail) THEN
        SELECT "Datos Correcto" AS "Mensaje";
    ELSE
        SELECT "Datos incorrectos" AS "Aviso";
    END IF;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psDateActual` () BEGIN
SELECT CURDATE();
END$$

```

```

CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psDatFalUsu` (`Pape2U` VARCHAR(200),
`PdirecU` VARCHAR(150), `PcpU` VARCHAR(5), `Ptelefono` VARCHAR(55), `PRFC` VARCHAR(13), `PfeNa` DATE,
`PidU` INT) BEGIN
    UPDATE usuario SET
        usuario.apellido2Usuario = Pape2U,
        usuario.direccionUsuario = PdirecU,
        usuario.codigoPostalUsuario = PcpU,
        usuario.telefonoUsuario = Ptelefono,
        usuario.rfeUsuario = PRFC,
        usuario.fechaNacimiento = PfeNa
    WHERE usuario.idUsuario = PidU;
END$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psEliProduct` (`Pidrod` INT) BEGIN
    UPDATE producto SET stockRealProducto = '0', producto.statusProducto ='0' WHERE
producto.idProducto=Pidrod;
END$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psEmailConfirm` (`Pcodi` VARCHAR(20),
`Pemail` VARCHAR(245)) BEGIN
    UPDATE usuario SET usuario.statusUsuario = '1' WHERE usuario.codeActive = Pcodi AND
usuario.emailUsuario = Pemail;
END$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psFeActD` () BEGIN
select CURDATE();
END$$

CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psFiltroCorreo` (`Pemail` VARCHAR(245))
BEGIN
    SELECT * FROM usuario WHERE usuario.emailUsuario = Pemail;
END$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psFiltroEmail` (IN `Pemail` VARCHAR(245))
NO SQL
    SELECT * FROM usuario
    WHERE usuario.emailUsuario = Pemail$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psImagenUsuarioPerfil` (`PdiUser` INT)
BEGIN
    SELECT usuario.imagenUsuario FROM usuario WHERE usuario.idUsuario = PdiUser AND
usuario.statusUsuario='1';
END$$
CREATE DEFINER='u743223069_storc'@`127.0.0.1` PROCEDURE `psInsCarrito` (`PFolioVenta` INT,
`PidProducto` INT, `PprecioUnitario` FLOAT, `PcantidadProducto` FLOAT) BEGIN
    INSERT INTO carrito
    VALUES (
        null,
        PFolioVenta,
        PidProducto,
        PprecioUnitario,
        PcantidadProducto,
        '1');
    UPDATE producto SET stockRealProducto = stockRealProducto - PcantidadProducto WHERE
producto.idProducto=PidProducto;
END$$

```

```

CREATE DEFINER='u743223069_stored'@'127.0.0.1` PROCEDURE `psInsClienIdU` (`PidUsuaCpay` INT,
`PclienId` VARCHAR(255)) BEGIN
    UPDATE usuario SET usuario.clienidpaypal = PclienId WHERE usuario.idUsuario = PidUsuaCpay;
END$$

CREATE DEFINER='u743223069_stored'@'127.0.0.1` PROCEDURE `psInserImgs` (`PidPro` INT, `Pimg1` VARCHAR(255), `Pimg2` VARCHAR(255), `Pimg3` VARCHAR(255), `Pimg4` VARCHAR(255)) BEGIN
    INSERT imgproducto VALUES(DEFAULT,PidPro,Pimg1,Pimg2,Pimg3,Pimg4);
END$$

CREATE DEFINER='u743223069_stored'@'127.0.0.1` PROCEDURE `psInsertCategoria` (`PdesCategoria` VARCHAR(150)) BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "A OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT categoria.desCategoria FROM categoria WHERE categoria.desCategoria = PdesCategoria)
    THEN
        SELECT CONCAT('Categoría Ya Existente: ', PdesCategoria) AS "Mensaje";
        ROLLBACK;
    ELSE

        INSERT categoria VALUES(
            DEFAULT,
            UPPER(PdesCategoria),
            '1');

        COMMIT;
        SELECT CONCAT('Categoría insertada con éxito!!!: ', PdesCategoria) AS "Aviso";

    END IF;
END$$

CREATE DEFINER='u743223069_stored'@'127.0.0.1` PROCEDURE `psInsertComenUsuario` (`PComentario` TEXT, `PidUsuario` INT, `PidProducto` INT) BEGIN
    INSERT detaproductocomen (comentario, idUsuario, idProducto, fecha)
        VALUES (PComentario, PidUsuario, PidProducto, NOW());
END$$

```

```

CREATE DEFINER='`u743223069_storec`@`127.0.0.1`' PROCEDURE `psInsertFormaPago`(`PdesFormaPago` VARCHAR(150)) BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "HA OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT formapago.desFormaPago FROM formapago WHERE formapago.desFormaPago = PdesFormaPago) THEN
        SELECT CONCAT('Forma de Pago ya existente : ', PdesFormaPago) AS "Mensaje";
        ROLLBACK;
    ELSE

        INSERT formapago VALUES(
            DEFAULT,
            UPPER(PdesFormaPago),
            '1');

        COMMIT;
        SELECT CONCAT('Forma de Pago insertada con éxito!!!: ', PdesFormaPago) AS "Aviso";
    END IF;
END$$

CREATE DEFINER='`u743223069_storec`@`127.0.0.1`' PROCEDURE `psInsertMarca`(`PdesMarca` VARCHAR(150)) BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "HA OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT marca.desMarca FROM marca WHERE marca.desMarca = PdesMarca) THEN
        SELECT CONCAT('Marca ya existente : ', PdesMarca) AS "Mensaje";
        ROLLBACK;
    ELSE
        INSERT marca VALUES(
            DEFAULT,
            UPPER(PdesMarca),
            '1');

        COMMIT;
        SELECT CONCAT('Marca insertada con éxito!!!: ', PdesMarca) AS "Aviso";
    END IF;
END$$

```

```

CREATE DEFINER='u743223069_storec'@`127.0.0.1` PROCEDURE `psInsertPermisos`(`PdesPermiso` VARCHAR(150)) BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "A OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT permiso.desPermiso FROM permiso WHERE permiso.desPermiso = PdesPermiso) THEN
        SELECT CONCAT('Permiso Ya Existente: ',PdesPermiso) AS "Mensaje";
        ROLLBACK;
    ELSE

        INSERT permiso VALUES(
            DEFAULT,
            UPPER(PdesPermiso),
            '1');

        COMMIT;
        SELECT CONCAT('Permiso insertado con exito!!!',PdesPermiso) AS "Aviso";
    END IF;
END$$

CREATE DEFINER='u743223069_storec'@`127.0.0.1` PROCEDURE `psInsertProducto`(`PnombrePro` VARCHAR(200), `PdesPro` VARCHAR(200), `PpreuniPro` DOUBLE, `PcanPro` DOUBLE, `PimagePro` VARCHAR(240), `PidMarcaPro` INT, `PidCategoriaPro` INT, `PidUsuarioPro` INT) BEGIN
    INSERT producto (
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.stockRealProducto,
        producto.imagenProducto,
        producto.idMarca,
        producto.idCategoria,
        producto.idUsuario,
        producto.fechaAlojadoProducto,
        producto.statusProducto)
    VALUES(
        DEFAULT,
        PnombrePro,
        PdesPro,
        PpreuniPro,
        PcanPro,
        PimagePro,
        PidMarcaPro,
        PidCategoriaPro,
        PidUsuarioPro,
        NOW(),
        '1'
    );
END$$

```

```

CREATE DEFINER='u743223069_stored'@'127.0.0.1' PROCEDURE `psInsertRoles` (`PdesRol` VARCHAR(150))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "A OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;
    IF EXISTS(SELECT rol.desRole FROM rol WHERE rol.desRole = PdesRol) THEN
        SELECT CONCAT('Rol Ya Existente: ', PdesRol) AS "Mensaje";
        ROLLBACK;
    ELSE
        INSERT rol VALUES(
            DEFAULT,
            UPPER(PdesRol),
            '1');

        COMMIT;
        SELECT CONCAT('Rol insertado con exito!!! ', PdesRol) AS "Aviso";
    END IF;
END$$
CREATE DEFINER='u743223069_stored'@'127.0.0.1' PROCEDURE `psInsertUsuario` (`PnombreUsu` VARCHAR(200), `Pape1Usu` VARCHAR(150), `Pape2Usu` VARCHAR(150), `PemailUsu` VARCHAR(245), `PfotoUsu` VARCHAR(245), `PdireccionUsu` VARCHAR(255), `PcodigoPostalUsu` VARCHAR(5), `PtelefonoUsu` VARCHAR(15), `PcontraUsu` VARCHAR(16), `PConfircontraUsu` VARCHAR(16), `PrfcUsu` VARCHAR(13)) BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT "A OCURRIDO UN ERROR" AS "AVISO";
        ROLLBACK;
    END;
    SET AUTOCOMMIT=0;

    IF EXISTS(SELECT usuario.emailUsuario FROM usuario WHERE usuario.emailUsuario = PemailUsu) THEN
        SELECT CONCAT('Usuario ya registrado con ese Email: ', PemailUsu) AS "Mensaje";
        ROLLBACK;
    ELSE
        INSERT usuario VALUES(
            DEFAULT,
            PnombreUsu,
            Pape1Usu,
            Pape2Usu,
            PemailUsu,
            PfotoUsu,
            PdireccionUsu,
            PcodigoPostalUsu,
            PtelefonoUsu,
            PcontraUsu,
            PConfircontraUsu,
            NOW(),
            '1',
            PrfcUsu);

        COMMIT;
        SELECT CONCAT('Se a realizado con éxito tu registro!!!: ', PemailUsu) AS "Aviso";
    END IF;

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psInsertUsuarioV1`(`PnombreUsu`  

VARCHAR(200), `Pape1Usu` VARCHAR(150), `PemailUsu` VARCHAR(245), `PcontraUsu` VARCHAR(16),  

`PConfircontraUsu` VARCHAR(16), `PcodeEmail` VARCHAR(21)) BEGIN  

    DECLARE EXIT HANDLER FOR SQLEXCEPTION  

    BEGIN  

        SELECT "A OCURRIDO UN ERROR" AS "AVISO";  

        ROLLBACK;  

    END;  

    SET AUTOCOMMIT=0;  

    IF EXISTS(SELECT usuario.emailUsuario FROM usuario WHERE usuario.emailUsuario = PemailUsu) THEN  

        SELECT CONCAT('Usuario ya registrado con ese Email: ', PemailUsu) AS "Mensaje";  

        ROLLBACK;  

    ELSE  

        INSERT usuario (idUsuario,  

                        nombreUsuario,  

                        apellido1Usuario,  

                        emailUsuario,  

                        contraUsuario,  

                        confirmaContraUsuario,  

                        fechaRegistroUsuario,  

                        statusUsuario,  

                        codeActive)VALUES(  

                        DEFAULT,  

                        PnombreUsu,  

                        Pape1Usu,  

                        PemailUsu,  

                        PcontraUsu,  

                        PConfircontraUsu,  

                        NOW(),  

                        '0',  

                        PcodeEmail  

                    );  

        COMMIT;  

        SELECT CONCAT('Se a realizado con éxito tu registro!!!: ', PemailUsu) AS "Aviso";  

    END IF;  

END$$  

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psInsVenta`(`PclaveTran` VARCHAR(250),  

`PpayDatos` TEXT, `Pemail` VARCHAR(250), `PtotalV` FLOAT) BEGIN  

    INSERT INTO venta (  

        claveTransaccion,  

        paypalDatos,  

        fechaVenta,  

        correo,  

        totalVendido,  

        statusVenta)  

    VALUES (  

        PclaveTran,  

        PpayDatos,  

        NOW(),  

        Pemail,  

        PtotalV,  

        '1');  

END$$

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psLoginUser` (IN `Pcontra` VARCHAR(16),
IN `PemailUsu` VARCHAR(245)) BEGIN
    SELECT CONCAT(usuario.nombreUsuario," ",usuario.apellido1Usuario) AS 'Nombre',
           usuario.emailUsuario AS 'Email',
           usuario.contraUsuario AS 'Contra'
    FROM usuario WHERE usuario.contraUsuario = Pcontra AND usuario.emailUsuario = PemailUsu;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMarcaAct` () BEGIN
    SELECT marca.idMarca,marca.desMarca FROM marca WHERE marca.statusMarca = '1';
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosActProducto` (`PidPro` INT, `PidUsu` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.imagenProducto,
        producto.stockRealProducto,
        marca.idMarca,
        marca.desMarca,
        categoria.idCategoria,
        categoria.desCategoria,
        producto.idUsuario
    FROM
        producto
    INNER JOIN marca ON marca.idMarca = producto.idMarca
    INNER JOIN categoria ON categoria.idCategoria = producto.idCategoria
    WHERE producto.idProducto = PidPro AND producto.idUsuario = PidUsu ;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosActProImg` (`PidUsuarioActProImg` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.imagenProducto,
        imgproducto.img1,
        imgproducto.img2,
        imgproducto.img3,
        imgproducto.img4,
        imgproducto.idImgProducto,
        producto.idUsuario,
        producto.statusProducto,
        producto.stockRealProducto
    FROM producto
    INNER JOIN imgproducto ON producto.idProducto = imgproducto.idProducto
    WHERE producto.idUsuario = PidUsuarioActProImg AND statusProducto= '1' AND
producto.stockRealProducto >0;
END$$

```

```

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosActProImg4`(`PidUsuarioActProImg` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.imagenProducto,
        imgproducto.img1,
        imgproducto.img2,
        imgproducto.img3,
        imgproducto.img4
    FROM producto
    INNER JOIN imgproducto ON producto.idProducto = imgproducto.idProducto
    WHERE producto.idUsuario = PidUsuarioActProImg AND statusProducto= '1' AND
    producto.stockRealProducto >0;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosActuImgsc`(`PidProImg` INT) BEGIN
    SELECT
        imgproducto.idProducto,
        imgproducto.img1,
        imgproducto.img2,
        imgproducto.img3,
        imgproducto.img4,
        producto.nombreProducto,
        producto.imagenProducto
    FROM
        producto
    INNER JOIN imgproducto ON imgproducto.idProducto = producto.idProducto
    WHERE producto.idProducto = PidProImg;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosCliidU`(`PidMosUsuaConpay` INT)
BEGIN
    SELECT usuario.clienidpaypal FROM usuario WHERE usuario.idUsuario= PidMosUsuaConpay;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosComenProdu`(`PidProducto` INT)
BEGIN
    SELECT
        CONCAT(usuario.nombreUsuario,'',
               usuario.apellido1Usuario) AS Nombre,
        detaproductocomen.comentario,
        usuario.imagenUsuario,
        detaproductocomen.fecha
    FROM
        detaproductocomen
    INNER JOIN usuario ON usuario.idUsuario = detaproductocomen.idUsuario WHERE
    detaproductocomen.idProducto = PidProducto;
END$$

```

```

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosComentarioImagen` ('PidProducto` INT) BEGIN
    SELECT
        usuario.imagenUsuario
    FROM
        detaproductocomen
    INNER JOIN usuario ON usuario.idUsuario = detaproductocomen.idUsuario WHERE
detaproductocomen.idProducto = PidProducto;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosComentarios` ('PidProducto` INT)
BEGIN
    SELECT
        usuario.emailUsuario AS usuario
    FROM
        detaproductocomen
    INNER JOIN usuario ON usuario.idUsuario = detaproductocomen.idUsuario WHERE
detaproductocomen.idProducto = PidProducto;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosIDpCPA` (`idProMPaL` INT) BEGIN
    SELECT
        usuario.clienidpaypal
    FROM
        usuario
    INNER JOIN producto ON usuario.idUsuario = producto.idUsuario
    WHERE producto.idProducto = idProMPaL;
END$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosNomCliente` ('PidVenta` INT)
SELECT venta.correo AS 'Cliente' FROM Venta WHERE venta.FolioVenta = PidVenta$$

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosProdaUsuario` ('PidUsuario` INT)
BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto AS 'Nombrw',
        producto.desProducto AS 'Descripcion',
        producto.precioUnitarioProducto AS '$ Unitario',
        producto.imagenProducto AS 'Imagen',
        producto.fechaAlojadoProducto AS 'Fecha',
        IF(producto.stockRealProducto = '0', 'Agotado', producto.stockRealProducto) AS 'Stock',
        IF(producto.statusProducto = '1', 'Activo', 'Inactivo') AS 'Estatus',
        marca.desMarca AS 'Marca',
        categoria.desCategoria AS 'Categoria'
    FROM producto
    INNER JOIN categoria ON categoria.idCategoria = producto.idCategoria
    INNER JOIN marca ON marca.idMarca = producto.idMarca
    WHERE producto.idUsuario = PidUsuario AND producto.statusProducto = 1 AND

```

```

CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosProdImgAgregar`(`PidUsuarioAI` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.imagenProducto,
        producto.idUsuario
    FROM
        producto
    INNER JOIN imgproducto ON imgproducto.idProducto = producto.idProducto
    WHERE producto.idUsuario = PidUsuarioAI AND producto.statusProducto = '1' AND
        imgproducto.idImgProducto = " AND producto.stockRealProducto >0;
    END$$
CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosProducto` () BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.imagenProducto,
        producto.stockRealProducto,
        producto.idUsuario
    FROM
        producto
    WHERE producto.statusProducto = '1' AND producto.stockRealProducto >= 1;
    END$$
CREATE DEFINER='u743223069_storec'@'127.0.0.1` PROCEDURE `psMosProductoComple` (`PidPro` INT)
BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.imagenProducto,
        producto.fechaAlojadoProducto,
        producto.stockRealProducto,
        producto.statusProducto,
        marca.desMarca,
        categoria.desCategoria,
        CONCAT(usuario.nombreUsuario,'',
               usuario.apellido1Usuario) AS Vendedor,
        usuario.emailUsuario as 'Email Vendedor'
    FROM
        producto
    INNER JOIN categoria ON categoria.idCategoria = producto.idCategoria
    INNER JOIN marca ON marca.idMarca = producto.idMarca
    INNER JOIN usuario ON usuario.idUsuario = producto.idUsuario
    WHERE producto.idProducto=PidPro;
    END$$

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosProductoCompleImg` (`PidProimg` INT) BEGIN
    SELECT * FROM imgproducto WHERE idProducto = PidProimg;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosProductoConUsu` (`PidUsuMosNo` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.imagenProducto,
        producto.stockRealProducto,
        producto.idUsuario
    FROM
        producto
    WHERE producto.statusProducto = '1' AND producto.stockRealProducto >= 1 AND producto.idUsuario
<> PidUsuMosNo;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosProductoConUsuFCPSU` (`PidUM` INT, `PidMP` INT) BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.precioUnitarioProducto,
        producto.imagenProducto,
        producto.stockRealProducto,
        producto.idUsuario
    FROM
        producto
    WHERE producto.statusProducto = '1'
    AND producto.stockRealProducto >= 1
    AND producto.idUsuario = PidUM
    AND producto.idProducto <> PidMP;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosProIns` (`PidPro` INT, `PidUsu` INT)
BEGIN
    SELECT
        producto.idProducto,
        producto.nombreProducto,
        producto.desProducto,
        producto.idUsuario
    FROM
        producto
    WHERE producto.idProducto = PidPro AND producto.idUsuario = PidUsu;
END$$

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosTicket` (`PidVenta` INT) SELECT
venta.correo AS 'Cliente',
producto.nombreProducto AS 'Producto',
carrito.precioUnitario AS 'Precio Unitario',
carrito.cantidadProducto AS 'Cantidad de Productos',
venta.totalVendido AS 'Total'
FROM carrito
INNER JOIN venta ON venta.FolioVenta = carrito.FolioVenta
INNER JOIN producto ON producto.idProducto = carrito.idProducto
WHERE venta.FolioVenta = PidVenta$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosTicketFechav` (`PidVF` INT) BEGIN
SELECT fechaVenta FROM venta WHERE FolioVenta = PidVF;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosTicketProductos` (`PidVenta` INT)
SELECT
producto.nombreProducto AS 'Producto',
carrito.precioUnitario AS 'Precio Unitario',
carrito.cantidadProducto AS 'Cantidad de Productos'
FROM carrito
INNER JOIN producto ON producto.idProducto = carrito.idProducto
WHERE carrito.FolioVenta = PidVenta$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosTicketUsuario` (`PidUsuNom` INT)
BEGIN
    SELECT CONCAT(nombreUsuario, " ",
apellido1Usuario) AS nombre
FROM usuario WHERE usuario.idUsuario = PidUsuNom;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psMosTotalVenta` (`PidVenta` INT)
SELECT venta.totalVendido AS 'Total' FROM Venta WHERE venta.FolioVenta = PidVenta$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psSelectClientes` () BEGIN
SELECT
usuario.idUsuario AS "No.",
CONCAT(usuario.nombreUsuario, " ",
usuario.apellido1Usuario) AS 'Nombre',
usuario.emailUsuario AS 'Correo',
usuario.fechaRegistroUsuario AS 'Fecha de Registro',
IF(usuario.statusUsuario = '1', 'Activo','Inactivo') AS 'Estado del Cliente'
FROM usuario;
END$$

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psSelectConCiplU` ('PidUsuaConpay' INT)
BEGIN
    SELECT COUNT(usuario.clienidpaypal)AS contador FROM usuario WHERE usuario.idUsuario=
    PidUsuaConpay;
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psUpdateDaFa` ('PidUsuario' INT) BEGIN
SELECT usuario.idUsuario,
    usuario.apellido2Usuario,
    usuario.direccionUsuario,
    usuario.codigoPostalUsuario,
    usuario.telefonoUsuario,
    usuario.rfeUsuario,
    usuario.fechaNacimiento
FROM usuario WHERE usuario.idUsuario = PidUsuario AND usuario.statusUsuario ='1';
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psUpdateImagenP` ('PidUser' INT,
`PimagenPro` VARCHAR(245)) BEGIN
UPDATE usuario SET usuario.imagenusuario = PimagenPro
WHERE usuario.idUsuario= PidUser AND usuario.statusUsuario = '1';
END$$

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psUpdateUsuarioT` ('PnombreUAct'
VARCHAR(200), `Papellido1UAct` VARCHAR(150), `Papellido2UAct` VARCHAR(150), `PcontraUAct` VARCHAR(16),
`PconfirmcontraUAct` VARCHAR(16), `PtelUAct` VARCHAR(15), `PrfcUAct` VARCHAR(13), `PdireccionUAct` VARCHAR(245),
`PcpUAct` VARCHAR(10), `PfechaUAct` DATE, `PidUAct` INT) BEGIN
    UPDATE usuario SET
        usuario.nombreUsuario = PnombreUAct,
        usuario.apellido1Usuario = Papellido1UAct,
        usuario.apellido2Usuario = Papellido2UAct,
        usuario.contraUsuario = PcontraUAct,
        usuario.confirmaContraUsuario = PconfirmcontraUAct,
        usuario.telefonoUsuario = PtelUAct,
        usuario.rfeUsuario = PrfcUAct,
        usuario.direccionUsuario = PdireccionUAct,
        usuario.codigoPostalUsuario = PcpUAct,
        usuario.fechaNacimiento = PfechaUAct
    WHERE usuario.idUsuario = PidUAct;
END$$

```

```

CREATE DEFINER=`u743223069_storec`@`127.0.0.1` PROCEDURE `psValidacionActivoE` ('Pcorreo`  

VARCHAR(245), `Pstatus` VARCHAR(1)) BEGIN  

SELECT * FROM usuario WHERE usuario.emailUsuario = Pcorreo AND usuario.statusUsuario = Pstatus;  

END$$

DELIMITER ;

-----

--  

-- Estructura de tabla para la tabla `carrito`  

--  

CREATE TABLE `carrito` (  

`idCarrito` int(11) NOT NULL,  

`FolioVenta` int(11) NOT NULL,  

`idProducto` int(11) NOT NULL,  

`precioUnitario` float(8,2) NOT NULL,  

`cantidadProducto` float(8,2) NOT NULL,  

`statusCarrito` varchar(1) NOT NULL  

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-----

--  

-- Estructura de tabla para la tabla `categoria`  

--  

CREATE TABLE `categoria` (  

`idCategoria` int(11) NOT NULL,  

`desCategoria` varchar(245) NOT NULL,  

`statusCategoria` varchar(1) NOT NULL  

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  

--  

-- Volcado de datos para la tabla `categoria`  

--  

INSERT INTO `categoria` (`idCategoria`, `desCategoria`, `statusCategoria`) VALUES  

(1, 'MUEBLES', '1'),  

(2, 'ELECTRÓNICOS', '1'),  

(3, 'ALIMENTOS Y BEBIDAS', '1'),  

(4, 'ROPA', '1'),  

(5, 'ZAPATOS', '1'),  

(6, 'JUGUETES Y JUEGOS', '1'),  

(7, 'LIBROS', '1'),  

(8, 'DEPORTES', '1');

```

```
--  
-- Estructura de tabla para la tabla `imgproducto`  
--  
  
CREATE TABLE `imgproducto` (  
  `idImgProducto` int(11) NOT NULL,  
  `idProducto` int(11) NOT NULL,  
  `img1` varchar(255) NOT NULL,  
  `img2` varchar(255) DEFAULT NULL,  
  `img3` varchar(255) DEFAULT NULL,  
  `img4` varchar(255) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
-----  
--  
-- Estructura de tabla para la tabla `marca`  
--  
  
CREATE TABLE `marca` (  
  `idMarca` int(11) NOT NULL,  
  `desMarca` varchar(245) NOT NULL,  
  `statusMarca` varchar(1) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
  
--  
-- Volcado de datos para la tabla `marca`  
--  
  
INSERT INTO `marca`(`idMarca`, `desMarca`, `statusMarca`) VALUES  
(6, 'COCA - COLA', '1'),  
(7, 'SALANDENS', '1'),  
(8, 'WRANGLER', '1'),  
(9, 'FLEXI', '1'),  
(10, 'ADIDAS', '1');  
-----  
--  
-- Estructura de tabla para la tabla `permiso`  
CREATE TABLE `permiso` (  
  `idPermiso` int(11) NOT NULL,  
  `desPermiso` varchar(150) NOT NULL,  
  `statusPermiso` varchar(1) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
--  
-- Volcado de datos para la tabla `permiso`  
--  
INSERT INTO `permiso`(`idPermiso`, `desPermiso`, `statusPermiso`) VALUES  
(1, 'TOTAL', '1');
```

```

-- 
-- Estructura de tabla para la tabla `producto` 
-- 

CREATE TABLE `producto` (
  `idProducto` int(11) NOT NULL,
  `nombreProducto` varchar(150) NOT NULL,
  `desProducto` varchar(255) NOT NULL DEFAULT '',
  `precioUnitarioProducto` float(8,2) NOT NULL DEFAULT 0.00,
  `imagenProducto` varchar(255) NOT NULL DEFAULT '',
  `fechaAlojadoProducto` date NOT NULL,
  `stockRealProducto` float(8,2) NOT NULL,
  `statusProducto` varchar(1) NOT NULL,
  `idMarca` int(11) NOT NULL,
  `idCategoria` int(11) NOT NULL,
  `idUsuario` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-----


-- 
-- Estructura de tabla para la tabla `rol` 
-- 

CREATE TABLE `rol` (
  `idRole` int(11) NOT NULL,
  `desRole` varchar(150) NOT NULL,
  `statusRole` varchar(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 
-- Volcado de datos para la tabla `rol` 
-- 

-- 

INSERT INTO `rol` (`idRole`, `desRole`, `statusRole`) VALUES
(2, 'ADMIN', '1');

-----


-- 
-- Estructura de tabla para la tabla `rolpermiso` 
-- 

CREATE TABLE `rolpermiso` (
  `idRolPermisoUsuario` int(11) NOT NULL,
  `idUsuario` int(11) NOT NULL,
  `idRol` int(11) NOT NULL,
  `idPermiso` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

```

-- 
-- Estructura de tabla para la tabla `usuario` 
-- 

CREATE TABLE `usuario` (
  `idUsuario` int(11) NOT NULL,
  `nombreUsuario` varchar(200) NOT NULL,
  `apellido1Usuario` varchar(150) NOT NULL,
  `emailUsuario` varchar(245) NOT NULL,
  `contraUsuario` varchar(16) NOT NULL,
  `confirmContraUsuario` varchar(16) NOT NULL,
  `apellido2Usuario` varchar(150) DEFAULT NULL,
  `imagenUsuario` varchar(245) DEFAULT NULL,
  `direccionUsuario` varchar(255) DEFAULT NULL,
  `codigoPostalUsuario` varchar(10) DEFAULT NULL,
  `telefonoUsuario` varchar(15) DEFAULT NULL,
  `fechaRegistroUsuario` datetime NOT NULL,
  `statusUsuario` varchar(1) NOT NULL,
  `rfeUsuario` varchar(13) DEFAULT NULL,
  `fechaNacimiento` date DEFAULT NULL,
  `codeActive` varchar(21) NOT NULL,
  `clientidpaypal` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

```

----- 

-- 
-- Estructura de tabla para la tabla `venta` 
-- 

CREATE TABLE `venta` (
  `FolioVenta` int(11) NOT NULL,
  `claveTransaccion` varchar(250) NOT NULL,
  `paypalDatos` text NOT NULL,
  `fechaVenta` datetime NOT NULL,
  `correo` varchar(255) NOT NULL,
  `totalVendido` float(8,2) NOT NULL,
  `statusVenta` varchar(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
-- 
-- Índices para tablas volcadas
-- 
```

```
-- 
-- Indices de la tabla `carrito`
-- 
```

```

ALTER TABLE `carrito`
  ADD PRIMARY KEY (`idCarrito`),
  ADD KEY `cp` (`idProducto`),
  ADD KEY `cfv` (`FolioVenta`);
```

```

-- Indices de la tabla `categoria`
ALTER TABLE `categoria`
ADD PRIMARY KEY (`idCategoria`),
ADD UNIQUE KEY `desCategoria`(`desCategoria`);

-- Indices de la tabla `detaproductocomen`
ALTER TABLE `detaproductocomen`
ADD PRIMARY KEY (`idDetalleProComen`),
ADD KEY `idUsuario`(`idUsuario`),
ADD KEY `idProducto`(`idProducto`);

-- Indices de la tabla `imgproducto`
ALTER TABLE `imgproducto`
ADD PRIMARY KEY (`idImgProducto`),
ADD KEY `idProducto`(`idProducto`);

-- Indices de la tabla `marca`
ALTER TABLE `marca`
ADD PRIMARY KEY (`idMarca`),
ADD UNIQUE KEY `desMarca`(`desMarca`);

-- Indices de la tabla `permiso`
ALTER TABLE `permiso`
ADD PRIMARY KEY (`idPermiso`),
ADD UNIQUE KEY `desPermiso`(`desPermiso`);

-- Indices de la tabla `producto`
ALTER TABLE `producto`
ADD PRIMARY KEY (`idProducto`),
ADD KEY `pm`(`idMarca`),
ADD KEY `pc`(`idCategoria`),
ADD KEY `pu`(`idUsuario`);

-- Indices de la tabla `rol`
ALTER TABLE `rol`
ADD PRIMARY KEY (`idRole`),
ADD UNIQUE KEY `desRole`(`desRole`);

```

```

-- Indices de la tabla `rolpermiso`
--
ALTER TABLE `rolpermiso`
ADD PRIMARY KEY (`idRolPermisoUsuario`),
ADD KEY `rpu` (`idUsuario`),
ADD KEY `rpp` (`idPermiso`),
ADD KEY `rpr` (`idRol`);

-- Indices de la tabla `usuario`
--
ALTER TABLE `usuario`
ADD PRIMARY KEY (`idUsuario`);

-- Indices de la tabla `venta`
--
ALTER TABLE `venta`
ADD PRIMARY KEY (`FolioVenta`);

-- AUTO_INCREMENT de las tablas volcadas
--

-- AUTO_INCREMENT de la tabla `carrito`
--
ALTER TABLE `carrito`
MODIFY `idCarrito` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT de la tabla `categoria`
--
ALTER TABLE `categoria`
MODIFY `idCategoria` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;

-- AUTO_INCREMENT de la tabla `detaproductocomen`
--
ALTER TABLE `detaproductocomen`
MODIFY `idDetalleProComen` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT de la tabla `imgproducto`
--
ALTER TABLE `imgproducto`
MODIFY `idImgProducto` int(11) NOT NULL AUTO_INCREMENT;

```

```

-- 
-- AUTO_INCREMENT de la tabla `marca` 
-- 

ALTER TABLE `marca` 
    MODIFY `idMarca` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11; 

-- 
-- AUTO_INCREMENT de la tabla `permiso` 
-- 

ALTER TABLE `permiso` 
    MODIFY `idPermiso` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2; 

-- 
-- AUTO_INCREMENT de la tabla `producto` 
-- 

ALTER TABLE `producto` 
    MODIFY `idProducto` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=34; 

-- 
-- AUTO_INCREMENT de la tabla `rol` 
-- 

ALTER TABLE `rol` 
    MODIFY `idRole` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3; 

-- 
-- AUTO_INCREMENT de la tabla `rolpermiso` 
-- 

ALTER TABLE `rolpermiso` 
    MODIFY `idRolPermisoUsuario` int(11) NOT NULL AUTO_INCREMENT; 

-- 
-- AUTO_INCREMENT de la tabla `usuario` 
-- 

ALTER TABLE `usuario` 
    MODIFY `idUsuario` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=39; 

-- 
-- AUTO_INCREMENT de la tabla `venta` 
-- 

ALTER TABLE `venta` 
    MODIFY `FolioVenta` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=77; 

-- 
-- Restricciones para tablas volcadas 
-- 

-- 
-- Filtros para la tabla `carrito` 
-- 

ALTER TABLE `carrito` 
    ADD CONSTRAINT `cfv` FOREIGN KEY (`FolioVenta`) REFERENCES `venta`(`FolioVenta`), 
    ADD CONSTRAINT `cp` FOREIGN KEY (`idProducto`) REFERENCES `producto`(`idProducto`); 

```

```

-- 
-- Filtros para la tabla `detaproductocomen` 
-- 
ALTER TABLE `detaproductocomen` 
    ADD CONSTRAINT `detaproductocomen_ibfk_1` FOREIGN KEY (`idUsuario`) REFERENCES `usuario`(`idUsuario`), 
    ADD CONSTRAINT `detaproductocomen_ibfk_2` FOREIGN KEY (`idProducto`) REFERENCES `producto`(`idProducto`);

-- 
-- Filtros para la tabla `imgproducto` 
-- 
ALTER TABLE `imgproducto` 
    ADD CONSTRAINT `idProducto` FOREIGN KEY (`idProducto`) REFERENCES `producto`(`idProducto`);

-- 
-- Filtros para la tabla `producto` 
-- 
ALTER TABLE `producto` 
    ADD CONSTRAINT `pc` FOREIGN KEY (`idCategoria`) REFERENCES `categoria`(`idCategoria`), 
    ADD CONSTRAINT `pm` FOREIGN KEY (`idMarca`) REFERENCES `marca`(`idMarca`), 
    ADD CONSTRAINT `pu` FOREIGN KEY (`idUsuario`) REFERENCES `usuario`(`idUsuario`);

-- 
-- Filtros para la tabla `rolpermiso` 
-- 
ALTER TABLE `rolpermiso` 
    ADD CONSTRAINT `rpp` FOREIGN KEY (`idPermiso`) REFERENCES `permiso`(`idPermiso`), 
    ADD CONSTRAINT `rpr` FOREIGN KEY (`idRol`) REFERENCES `rol`(`idRole`), 
    ADD CONSTRAINT `rpu` FOREIGN KEY (`idUsuario`) REFERENCES `usuario`(`idUsuario`); 
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */; 
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */; 
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

4.4 Servicio web (API REST)

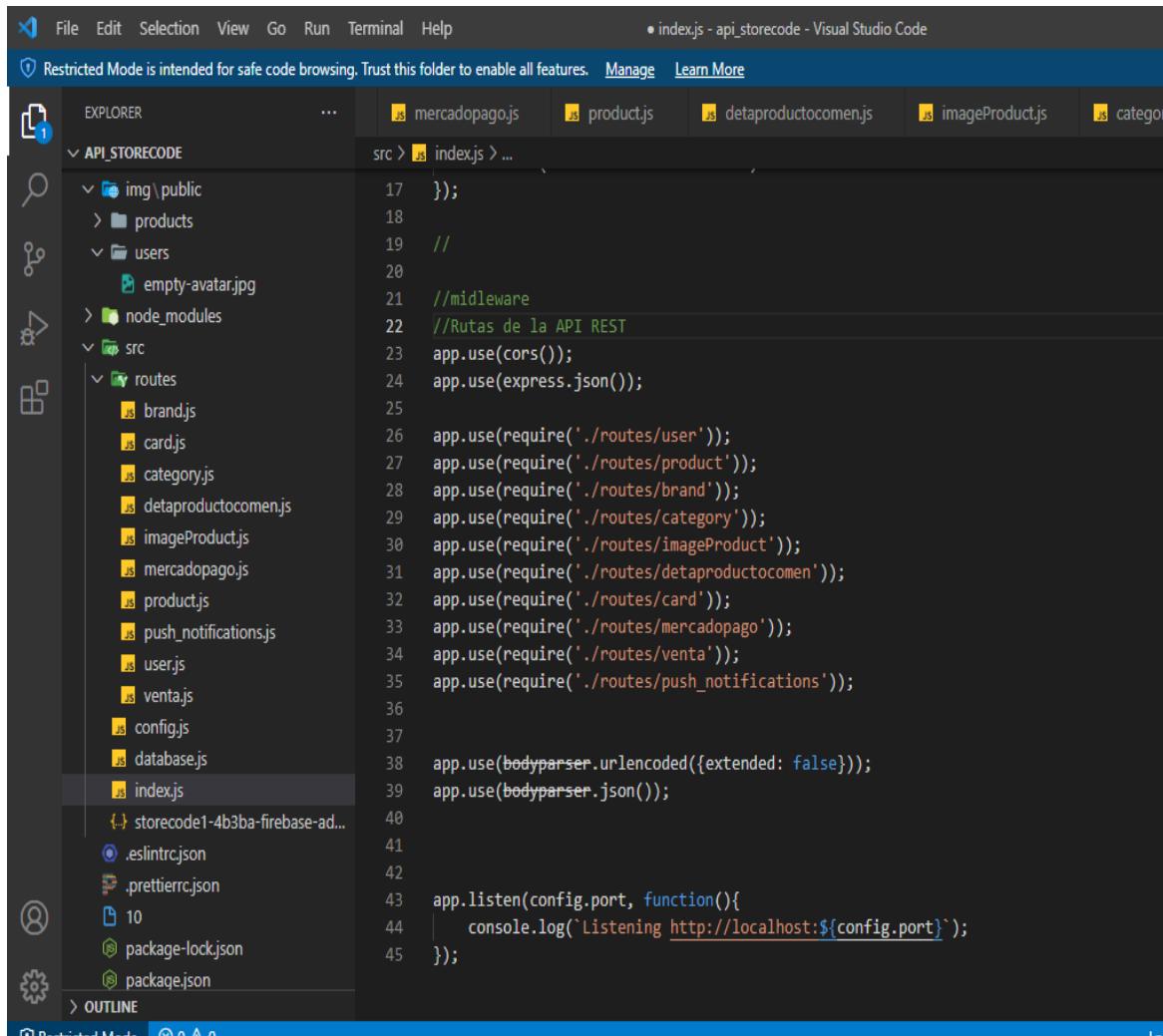
En la tabla 4-12 se describe la API Web con los endpoints más esenciales del Sistema de ventas online para microempresas, bajo plataformas móviles.

URI	Acción	Entrada	Salida	Script que lo consume	Descripción
storecode/login	POST	Objeto ReqLoginDto	Objeto RespUserData	Login.dart	Solicita el inicio de sesión de un usuario y si es autorizado devuelve la información del usuario.
storecode/users/{id}	GET	Id del usuario	Objeto RespUserData	Profile.dart	Obtiene la información del usuario.
storecode/users	POST	Objeto User	Mensaje: "Usuario registrado"	RegisterUser.dart	Inserta un nuevo usuario en la base de datos.
storecode/users/{id}	PUT	Objeto User	--	--	Actualiza los datos del usuario.
storecode/delete/users/{id}	PUT	Id del usuario	--	--	Realiza un eliminado lógico del usuario, al actualizar su status a "0".
storecode/products	GET	--	Lista de objeto producto	Home.dart	Obtiene la lista de los productos que están disponibles para su venta.
storecode/upload	POST	Objeto producto	--	RegisterProduct.dart	Registra un producto en la base de datos.
storecode/categories	GET	--	Lista de objeto categoría	ProductDetail.dart	Obtiene una lista de las categorías existentes.
storecode/brands	GET	--	Lista de objeto marca	Product_.dart	Obtiene una lista de las

				detail.dart	marcas existentes.
storecode/products/iduser/{id}	GET	Id del usuario	Lista de objeto producto.	Product s_onsale. dart	
storecode/products/{id}	PUT	Id del producto y objeto producto	--	Update — product s.dart	Actualiza los datos de un producto.
storecode/delete/products/{id}	PUT	Id del producto.	--	Product s_onsale. dart	Realiza un eliminado lógico del producto, al actualizar su status a “0”
storecode/create_preference	POST	Objeto ReqItemProduct	IdPreference	Carrito. dart	Obtiene el id preference para realizar el pago.
storecode/venta	POST	Objeto venta	--	Carrito. dart	Guarda el registro de una venta.

Tabla 4-12 Endpoints de la API REST

A nivel de código, las rutas se integran con la instrucción `app.use`, así como se muestra en la figura 4.34.



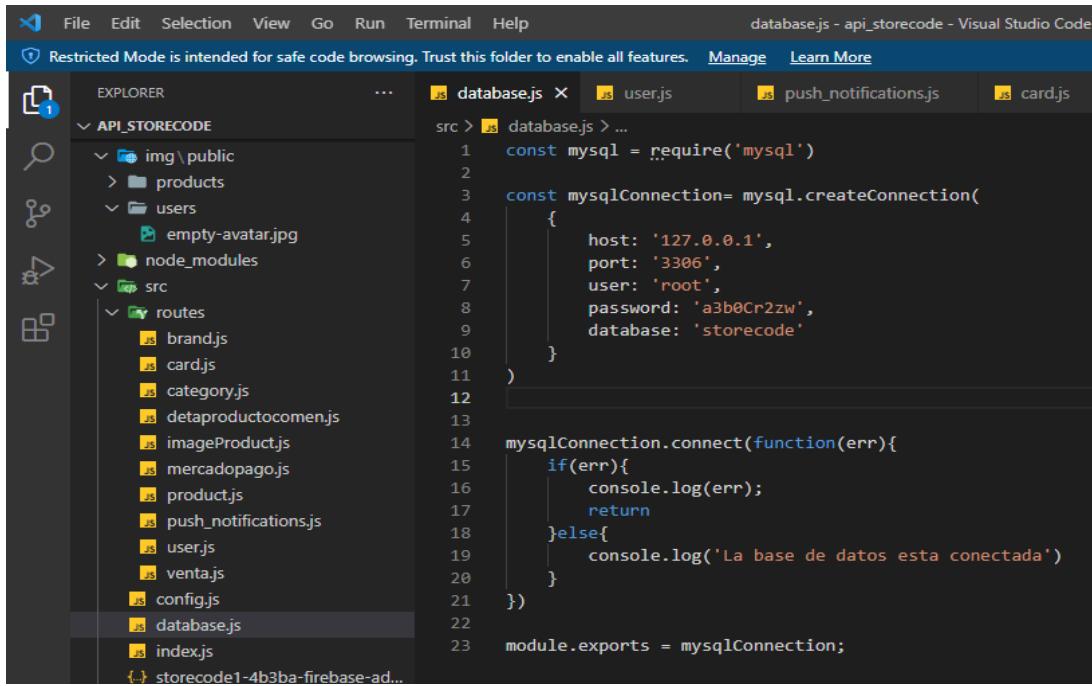
The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** index.js - api_storecode - Visual Studio Code.
- Message Bar:** Restricted Mode is intended for safe code browsing. Trust this folder to enable all features. Manage Learn More.
- Explorer View:** Shows the project structure under 'API_STORECODE'. The 'src' folder contains 'routes' which includes files like brand.js, card.js, category.js, dataproductocomen.js, imageProduct.js, mercadopago.js, product.js, push_notifications.js, user.js, and venta.js. Other files in 'src' include config.js, database.js, and index.js. Outside 'src' are 'img/public', 'products', and 'users' folders, along with 'node_modules' and configuration files (.eslintrc.json, .prettierrc.json, package-lock.json, package.json).
- Code Editor:** The 'index.js' file is open, showing the following code:

```
17 });
18
19 // middleware
20 //Rutas de la API REST
21 app.use(cors());
22 app.use(express.json());
23
24 app.use(require('./routes/user'));
25 app.use(require('./routes/product'));
26 app.use(require('./routes/brand'));
27 app.use(require('./routes/category'));
28 app.use(require('./routes/imageProduct'));
29 app.use(require('./routes/dataproductocomen'));
30 app.use(require('./routes/card'));
31 app.use(require('./routes/mercadopago'));
32 app.use(require('./routes/venta'));
33 app.use(require('./routes/push_notifications'));
34
35 app.use(bodyParser.urlencoded({extended: false}));
36
37 app.use(bodyParser.json());
38
39 app.listen(config.port, function(){
40   console.log(`Listening ${config.port}`);
41
42 });
43
44 });
45 }
```

Figura 4.33 Integración de rutas de la API REST

Para conectar el servicio web con la base de datos, se integró la dependencia **mysql** en NodeJS, a través de un objeto json se enviaron las credencias de la base de datos y con la instrucción **mysqlConnection.connect** se hace la conexión con la base de datos, tal y como se muestra en la figura 4.35.



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** database.js - api_storecode - Visual Studio Code. A warning message "Restricted Mode is intended for safe code browsing. Trust this folder to enable all features." is displayed.
- Explorer View:** Shows the project structure under "API_STORECODE". It includes folders for "img\public", "products", "users" (containing "empty-avatar.jpg"), "node_modules", and "src" (containing "routes" with files like brand.js, card.js, category.js, dataproductocomen.js, imageProduct.js, mercadopago.js, product.js, push_notifications.js, user.js, venta.js), "config.js", "database.js", and "index.js".
- Code Editor:** The "database.js" file is open. The code connects to a MySQL database using the "mysql" module. It defines a connection object with host, port, user, password, and database details, then attempts to connect and logs a success message if successful.

```
const mysql = require('mysql');
const mysqlConnection= mysql.createConnection(
{
  host: '127.0.0.1',
  port: '3306',
  user: 'root',
  password: 'a3b0Cr2zw',
  database: 'storecode'
})
mysqlConnection.connect(function(err){
  if(err){
    console.log(err);
    return;
  }else{
    console.log('La base de datos esta conectada')
  }
})
module.exports = mysqlConnection;
```

Figura 4.34 Conexión de la API REST con la base de datos

Capítulo 5

Pruebas y resultados

5.1 Implementación de los diseños previamente realizados.

En la figura 5.1 se muestra el resultado de la implementación de la interfaz “Inicio de sesión”.

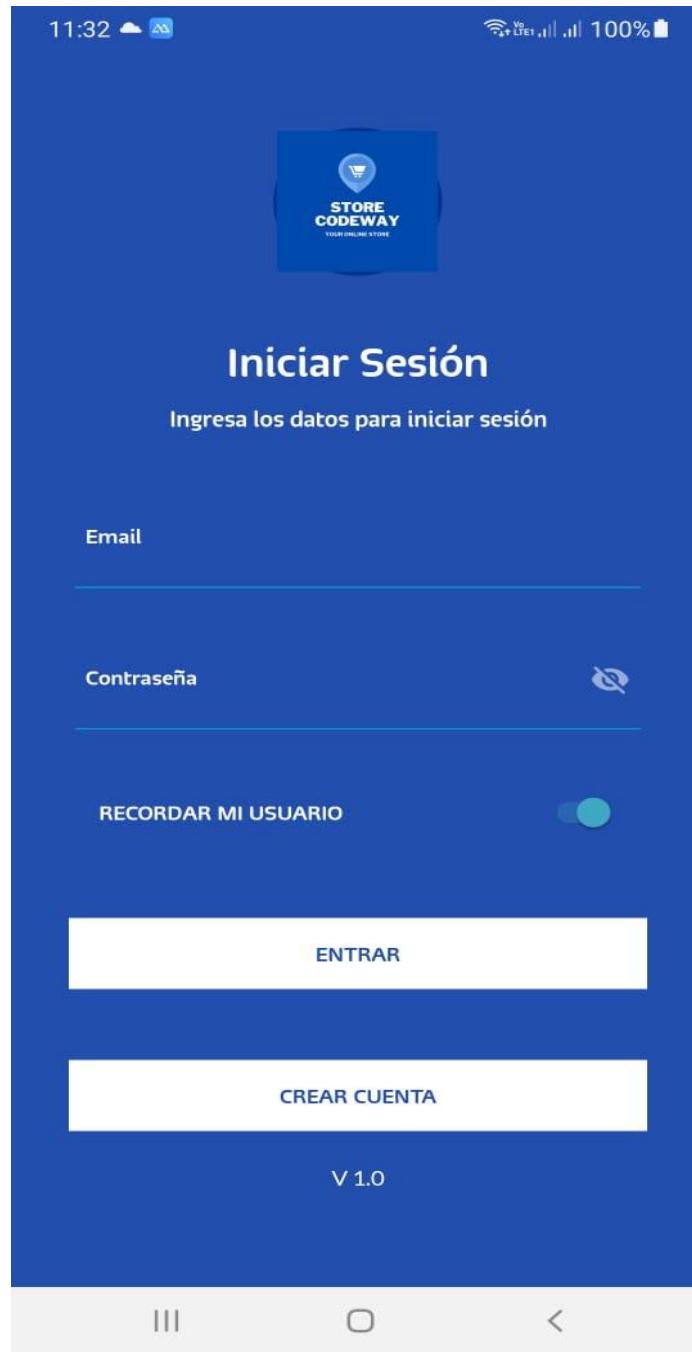


Figura 5.1 Implementación de la interfaz de inicio de sesión

La implementación de la interfaz de “Inicio” cuando el usuario no está logeado se muestra en la figura 5.2.



Figura 5.2 Implementación de la interfaz "Inicio" cuando el usuario no está logeado

El resultado de la implementación de la interfaz “Crear cuenta”, se muestra en la interfaz 5.3.

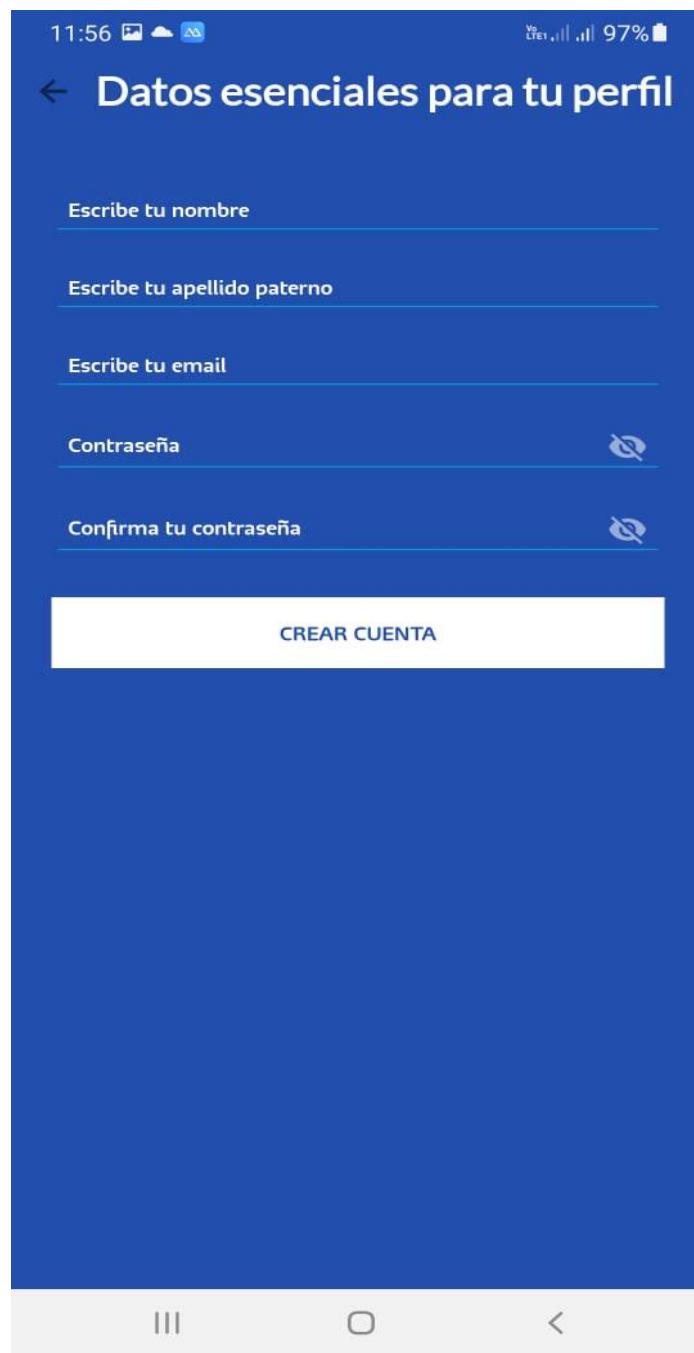


Figura 5.3 Implementación de la interfaz "Crear cuenta"

5. Pruebas y resultados

En la figura 5.4 se muestra la interfaz de “perfil”.



Figura 5.4 Interfaz de "Perfil".

En la figura 5.5 se muestra la interfaz de “Inicio” cuando el usuario ya está logeado, es decir que ya ha iniciado sesión dentro del aplicativo móvil.



Figura 5.5 Interfaz de "Inicio" cuando el usuario ya está logeado.

En la figura 5.6 se muestra la Interfaz “Carrito de compras”, dónde el usuario puede ver todos los productos que ha agregado a su carrito.



Figura 5.6 Interfaz "Carrito de compras"

En la figura 5.7 se muestra la primera interfaz de todo el flujo de pago, en el cual se elige una forma de pago, dentro del aplicativo móvil solo se configuro el pago con tarjetas.

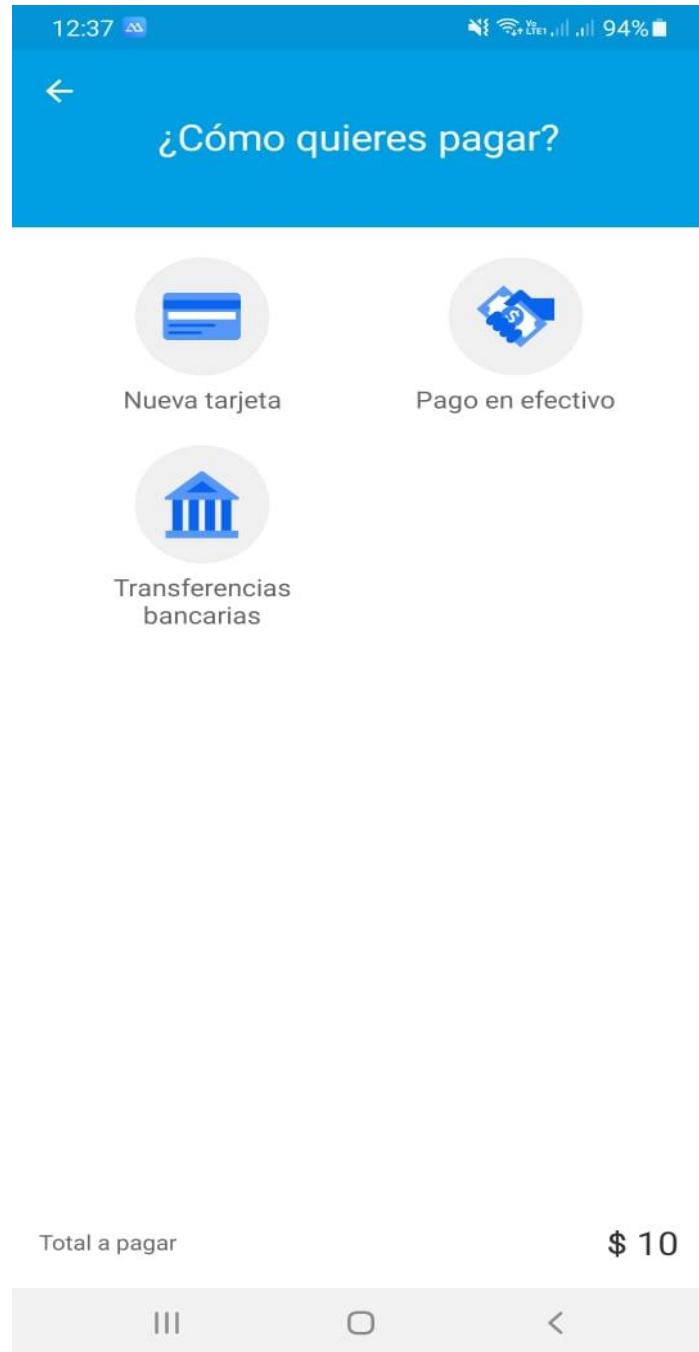


Figura 5.7 Interfaz 1 del flujo de pago

En la figura 5.8 se muestra la segunda interfaz del proceso de pago, donde se debe elegir el tipo de tarjeta con la cual se desea pagar.

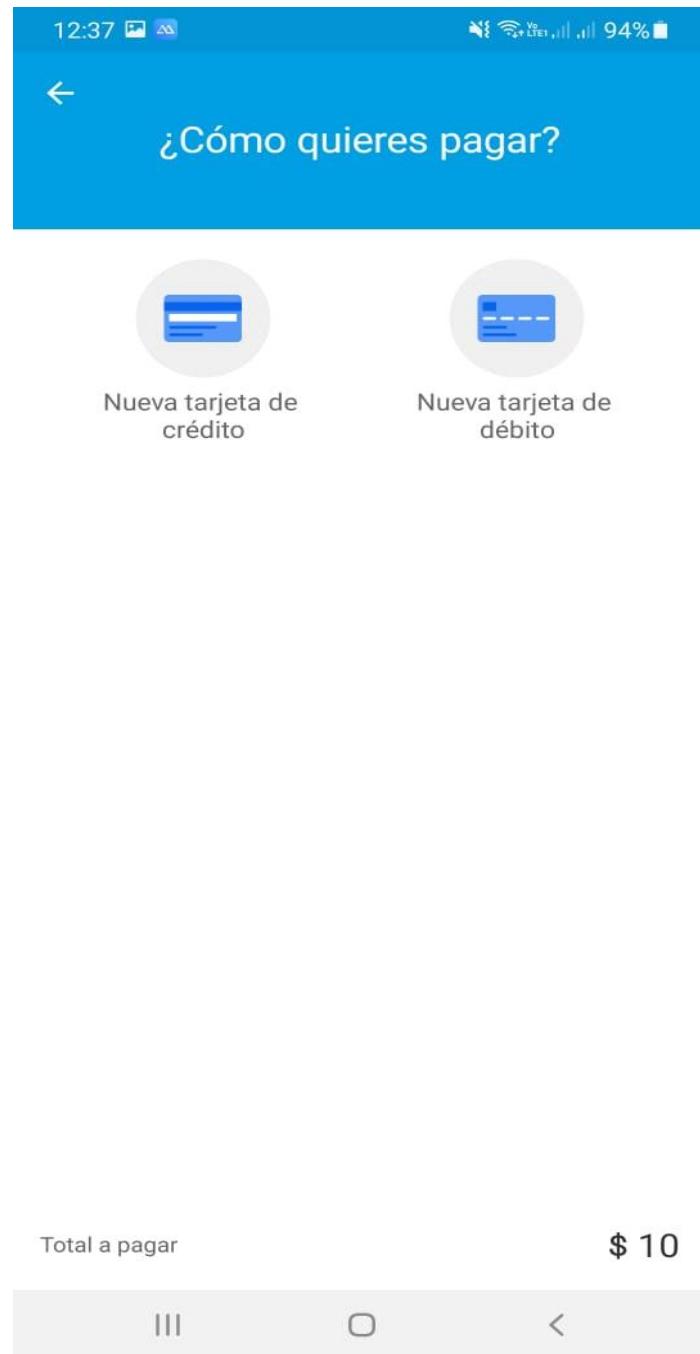


Figura 5.8 Interfaz 2 del proceso de pago

En la figura 5.9 se muestra la tercera interfaz del proceso de pago, donde el usuario ingresa el número de su tarjeta.

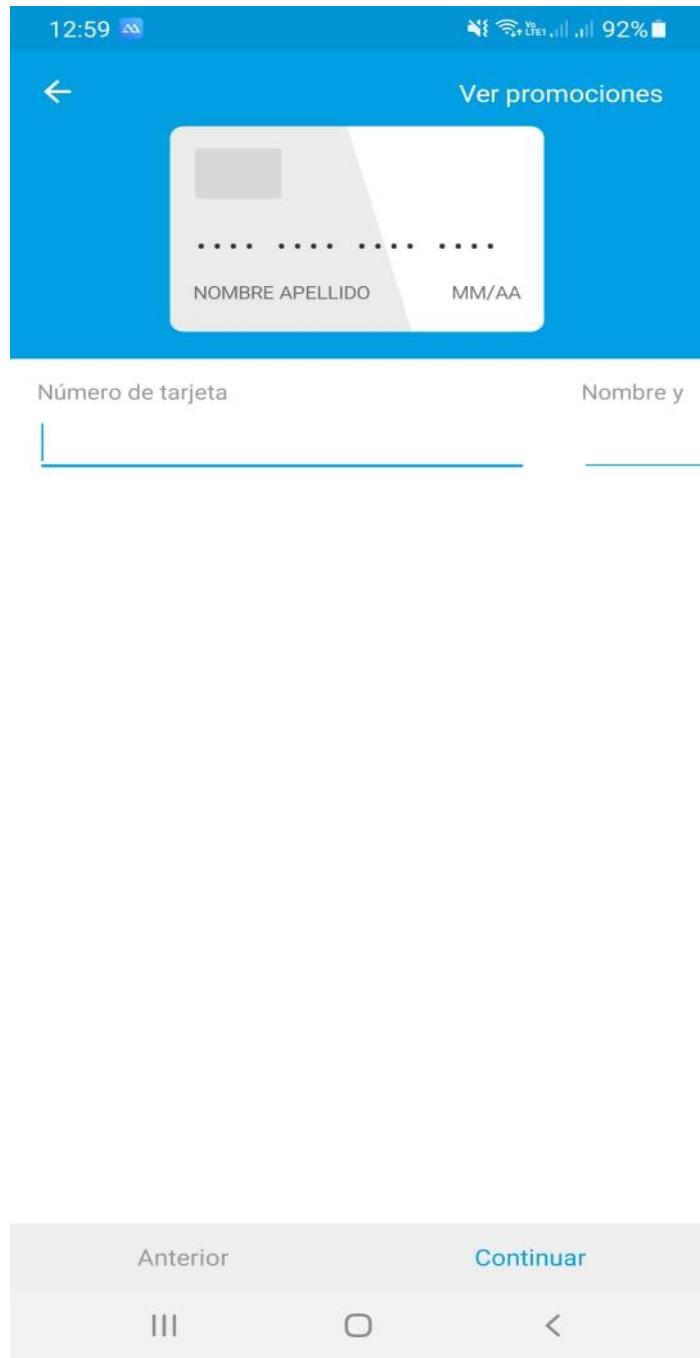


Figura 5.9 Interfaz 3 del proceso de pago

En la figura 5.10 se muestra la cuarta interfaz del proceso de pago, donde el usuario debe ingresar su nombre y apellidos asociados a la tarjeta para continuar.

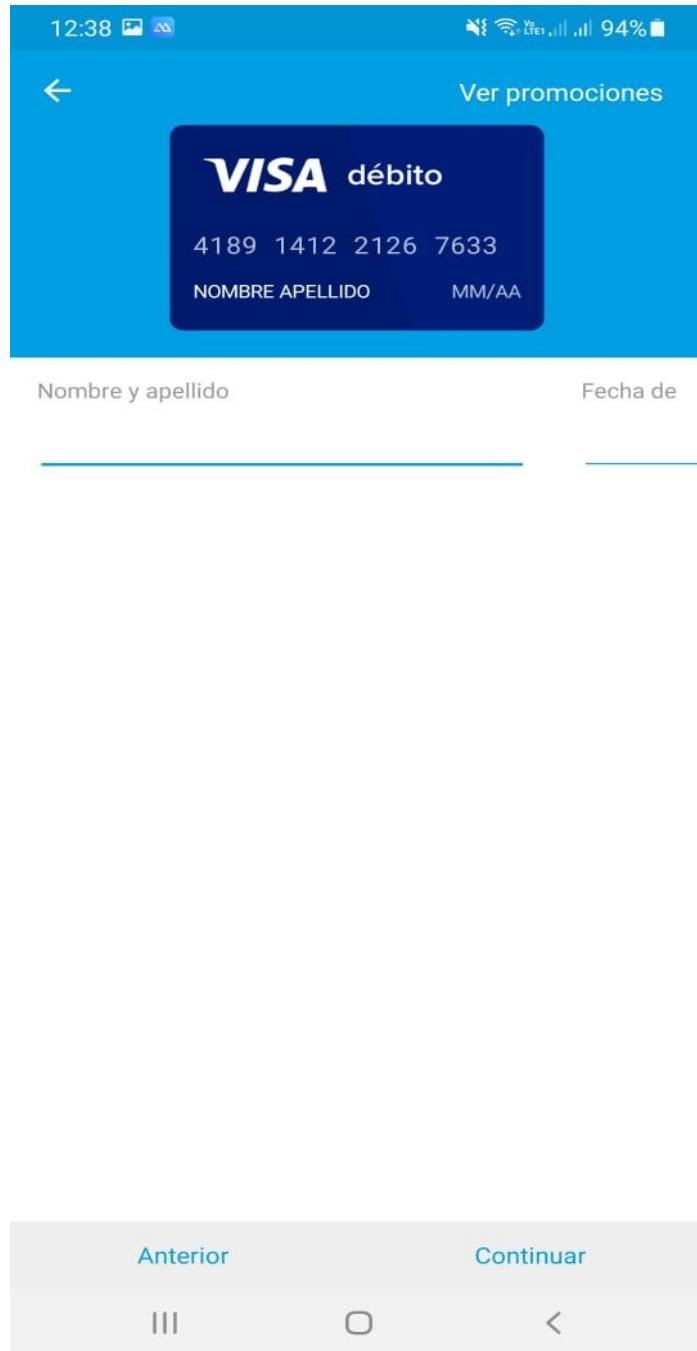


Figura 5.10 Interfaz 4 del proceso de pago

En la figura 5.11 se muestra la quinta interfaz del proceso de pago, donde el usuario ingresa la fecha de expiración de su tarjeta.



Figura 5.11 Interfaz 5 del proceso de pago

En la figura 5.12 se muestra la sexta interfaz del flujo de pago, donde el usuario debe ingresar el código de seguridad de su tarjeta.

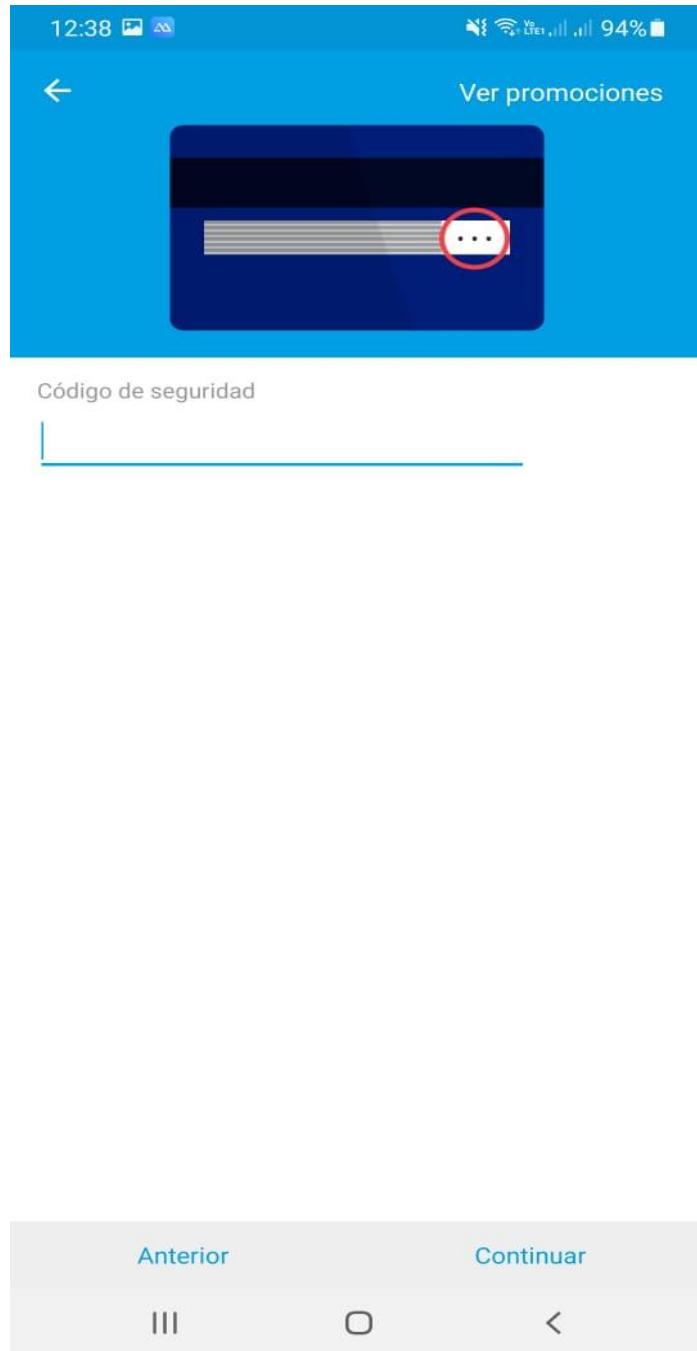


Figura 5.12 Interfaz 6 del proceso de pago.

En la figura 5.13 se muestra la interfaz número 7 del proceso de pago, donde se muestra un resumen de lo que el cliente va a pagar.

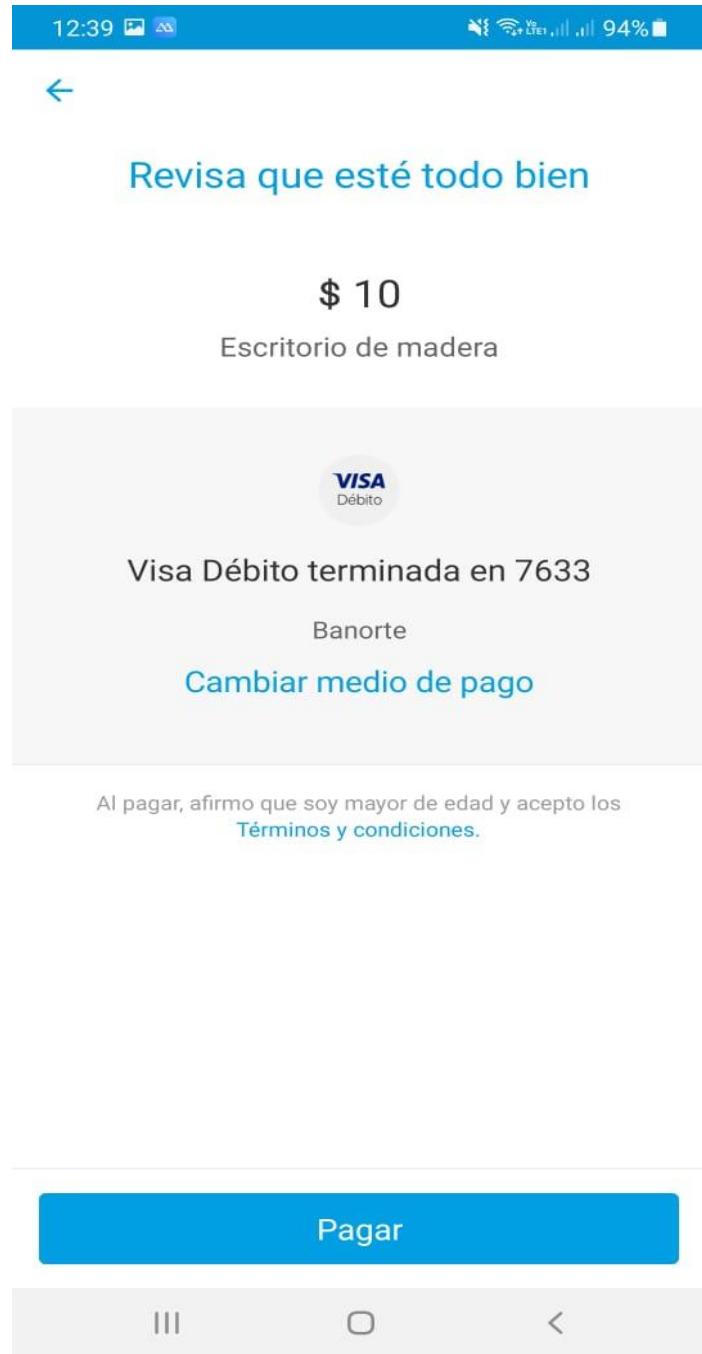


Figura 5.13 Interfaz 7 del proceso de pago

En la figura 5.14 se muestra la interfaz 8 y final del flujo de pago, donde el usuario puede visualizar el número de la operación y la cantidad que pago por el producto.

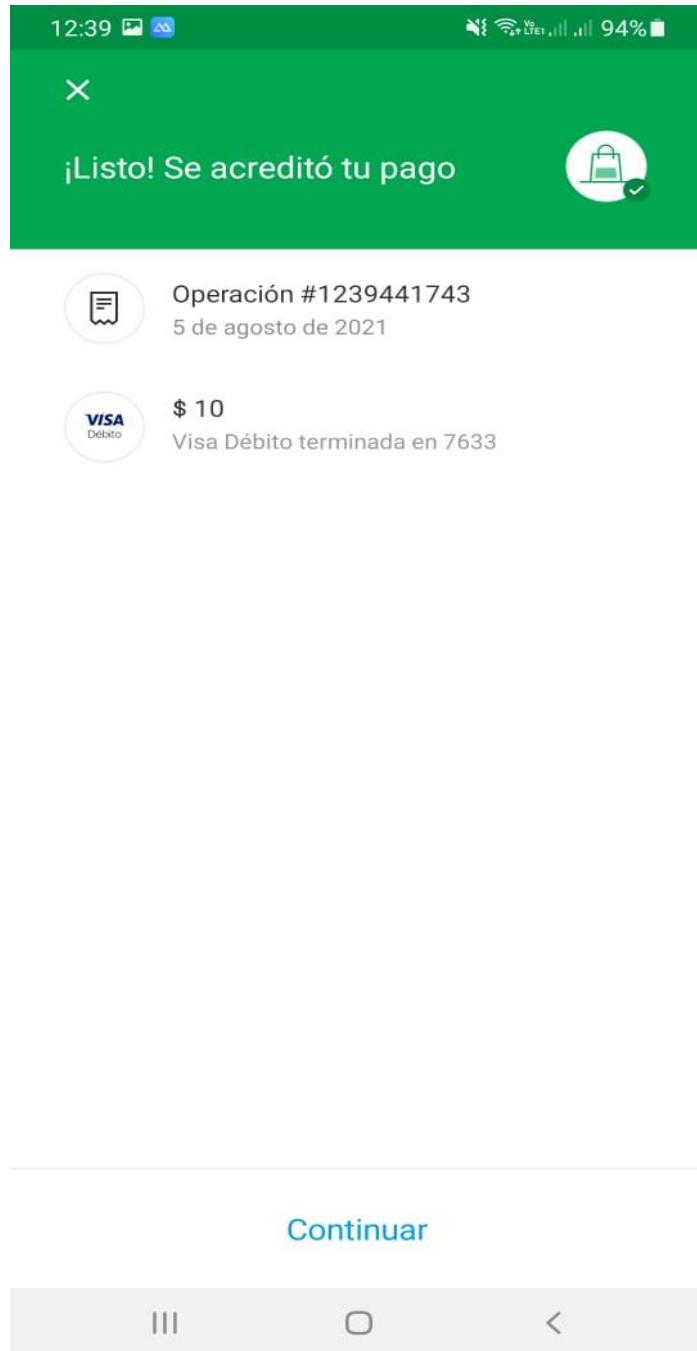


Figura 5.14 Interfaz 8 del proceso de pago

En la figura 5.15 se muestra la interfaz “Datos de mercado pago”, donde el usuario vendedor debe ingresar su public key y Access token de producción que le brinda la plataforma de Mercado pago.

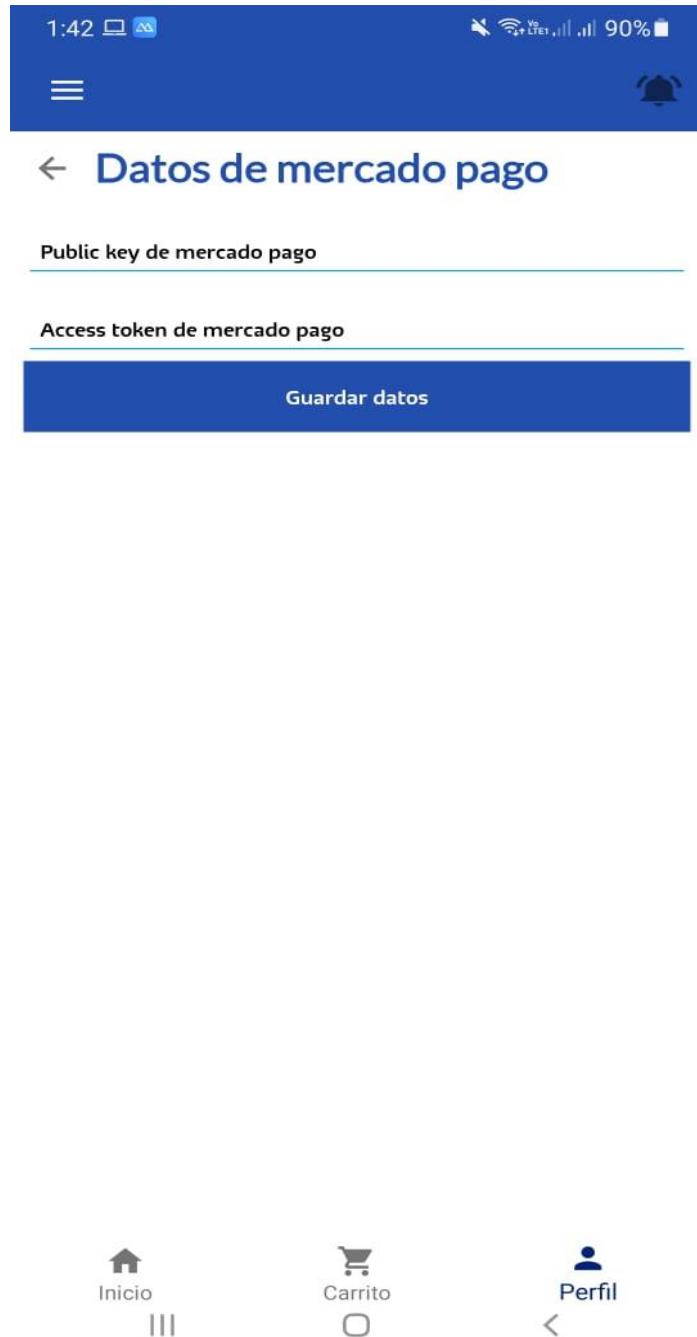


Figura 5.15 Interfaz "Datos de mercado pago"

En la figura 5.16 se muestra la interfaz “Subir un producto”, donde el usuario puede registrar un nuevo producto en la plataforma.



Figura 5.16 Interfaz “Subir producto”

En la figura 5.17 se muestra la interfaz “Productos en venta”, donde el usuario puede visualizar los productos que ha registrado para venderlos.



Figura 5.17 Interfaz "Productos en venta"

5.2 Implementación de la base de datos

El resultado obtenido tras ejecutar el script de la base de datos, es el siguiente, se creó el Schema **storecode**, con las siguientes tablas: carrito, carritoventa, categoría, detaproductocomen, dirección, imgproducto, marca, metodopago, permiso, producto, productocarrito, rol, rolpermiso, usuario y venta, así como se muestra en la figura 5.18.

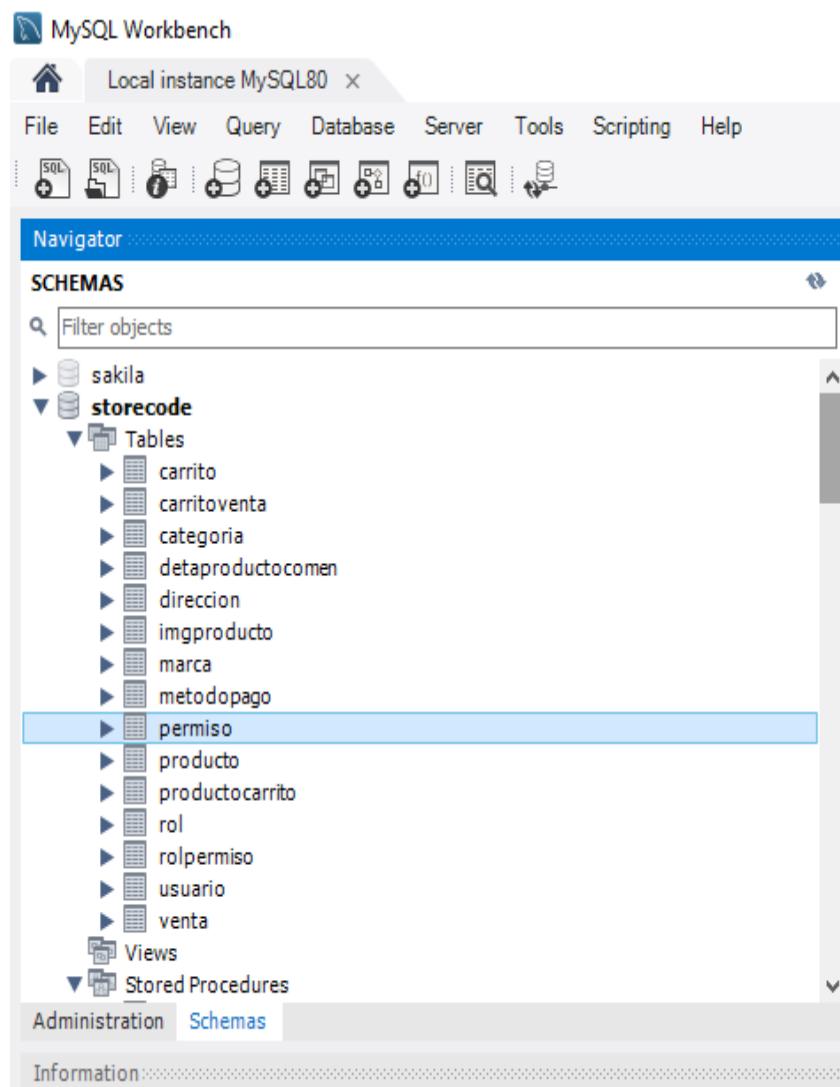
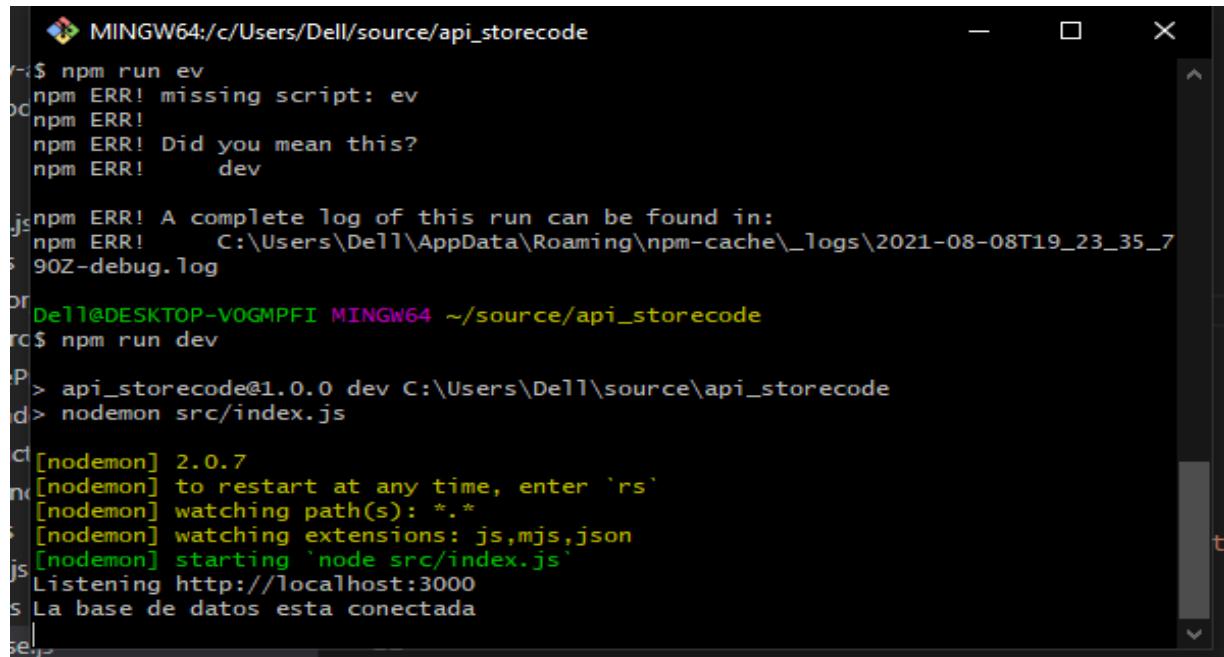


Figura 5.18 Base de datos Storecode

5.3 Implementación del servicio web

En la figura 5.19 se muestra el servicio web ejecutándose correctamente, pues la conexión con la base de datos fue exitosa.



```
MINGW64:/c/Users/Dell/source/api_storecode
$ npm run ev
npm ERR! missing script: ev
npm ERR!
npm ERR! Did you mean this?
npm ERR!     dev

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\DELL\AppData\Roaming\npm-cache\_logs\2021-08-08T19_23_35_7
90Z-debug.log
Dell@DESKTOP-V0GMPFI MINGW64 ~/source/api_storecode
$ npm run dev
> api_storecode@1.0.0 dev C:\Users\DELL\source\api_storecode
d> nodemon src/index.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node src/index.js'
[nodemon] Listening http://localhost:3000
La base de datos esta conectada
se,
```

Figura 5.19 Servicio web en ejecución

5.4 Pruebas a la aplicación móvil

A continuación, se describen las pruebas realizadas a la aplicación móvil, dichas pruebas fueron realizadas en colaboración con la empresa “Codeway Soluciones Integrales”.

Para la realización de pruebas, la aplicación móvil se instaló en un dispositivo físico, las características de este dispositivo se describen en la tabla 5-1.

Sistema operativo	Android 11
Modelo	Samsung Galaxy A32
Memoria RAM	4GB

Tabla 5-1 Características del dispositivo de pruebas

Las pruebas se realizaron a los siguientes casos de usos:

- Crear cuenta.
- Iniciar sesión.
- Subir un producto.
- Realizar una compra.

A. Crear cuenta

Pasos:

1. Dar click en la aplicación “Storecode” y esta se abrirá automáticamente.

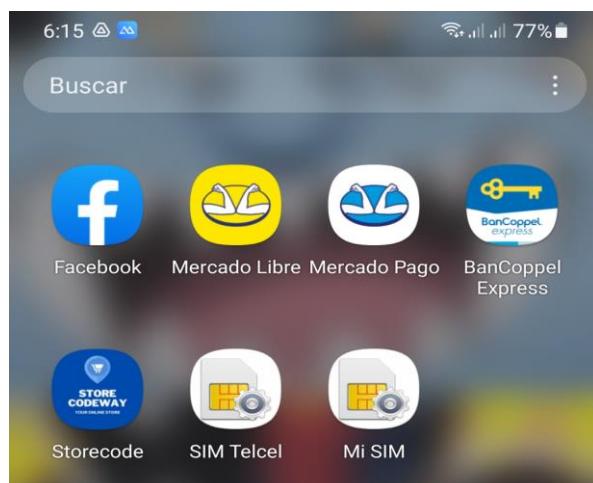


Figura 5.20 Aplicación móvil "Storecode" instalada en un dispositivo físico.

- Al abrir la aplicación, se navega hacia la pantalla “Perfil”, el sistema detectará que no se ha iniciado sesión y mostrara una pantalla como la que se muestra en la figura 5.21. Luego se da click en el botón “Iniciar sesión”.

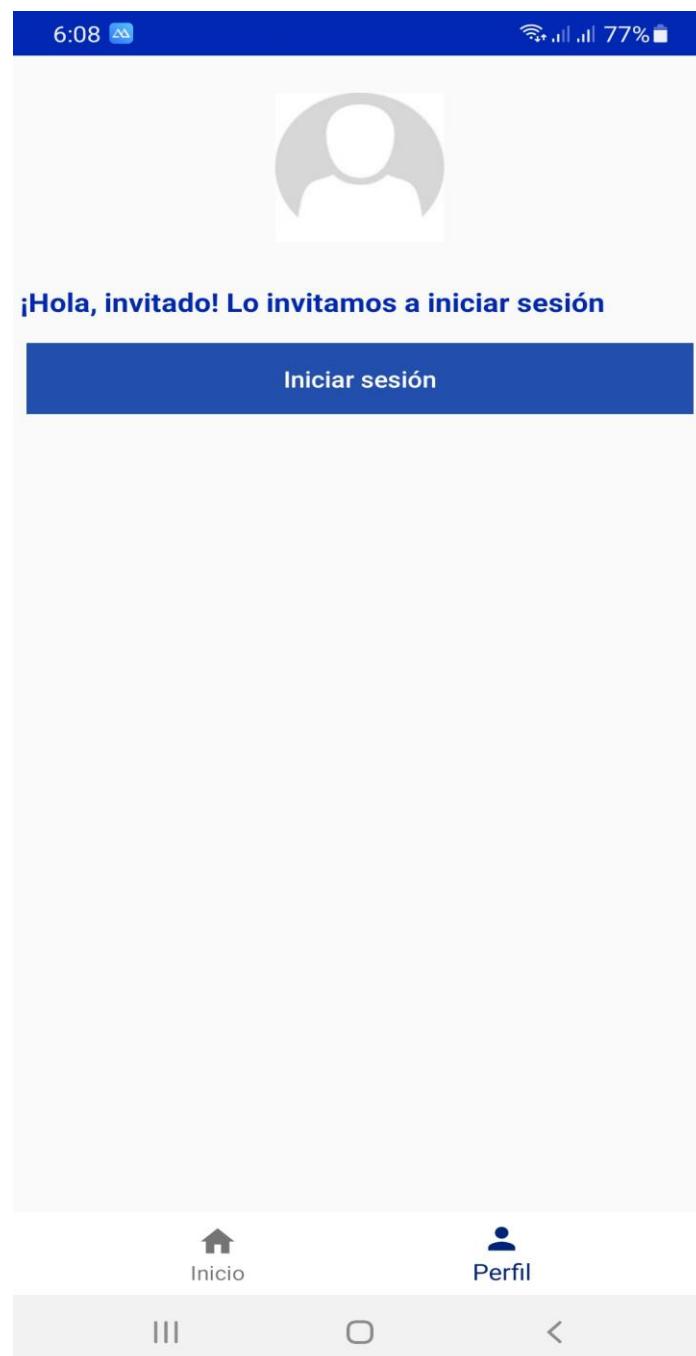


Figura 5.21 Pantalla "Perfil" de usuario no logeado

3. Se da click en el botón “Crear cuenta”, de la pantalla “Iniciar sesión”, y a continuación se muestra la pantalla “Crear cuenta”, el cual contiene un formulario solicitando los datos esenciales para crear una cuenta, se llenan los datos que nos solicita el formulario y posteriormente se da click en el botón “Crear cuenta”.

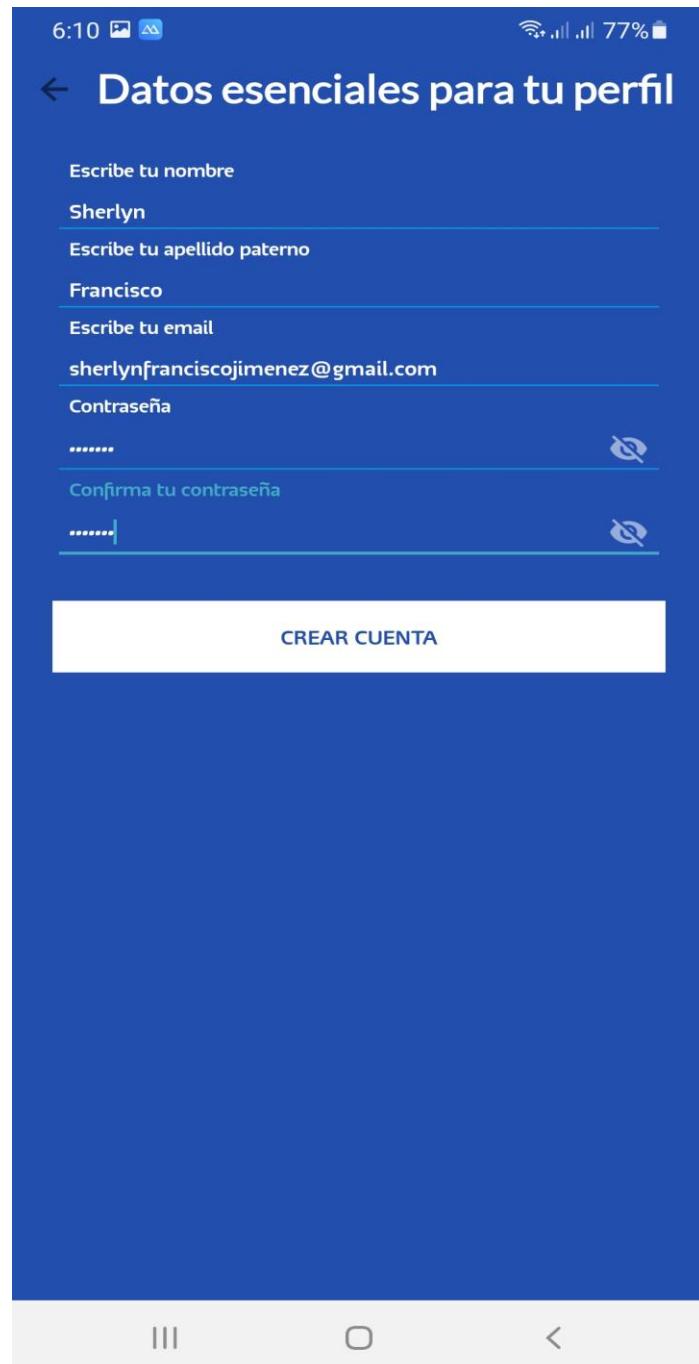


Figura 5.22 Pantalla "Crear cuenta"

4. Se muestra en pantalla un mensaje de aviso diciendo que la cuenta se creó con éxito, así como se ve en la figura 5.23.

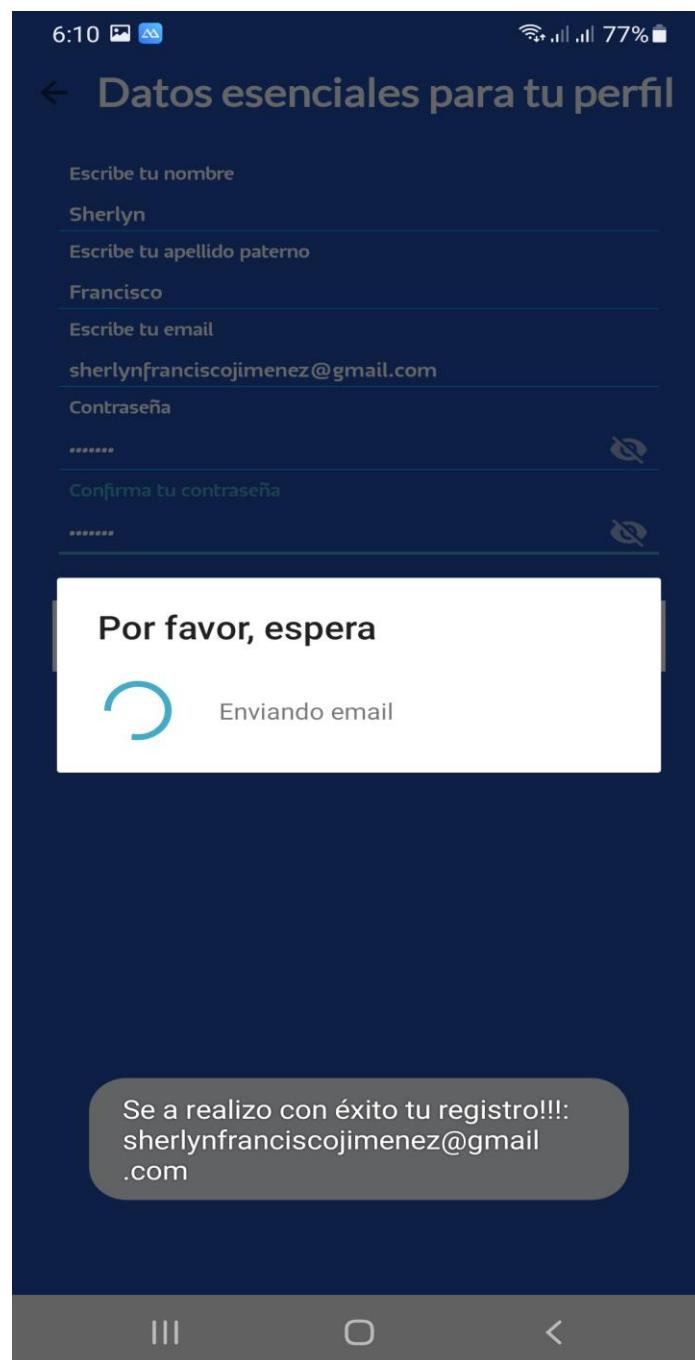


Figura 5.23 Respuesta del servicio web al crear una cuenta

5. Se envía un enlace para la activación de la cuenta al correo que registró el usuario. Después se da click en el link y se activa la cuenta del usuario.

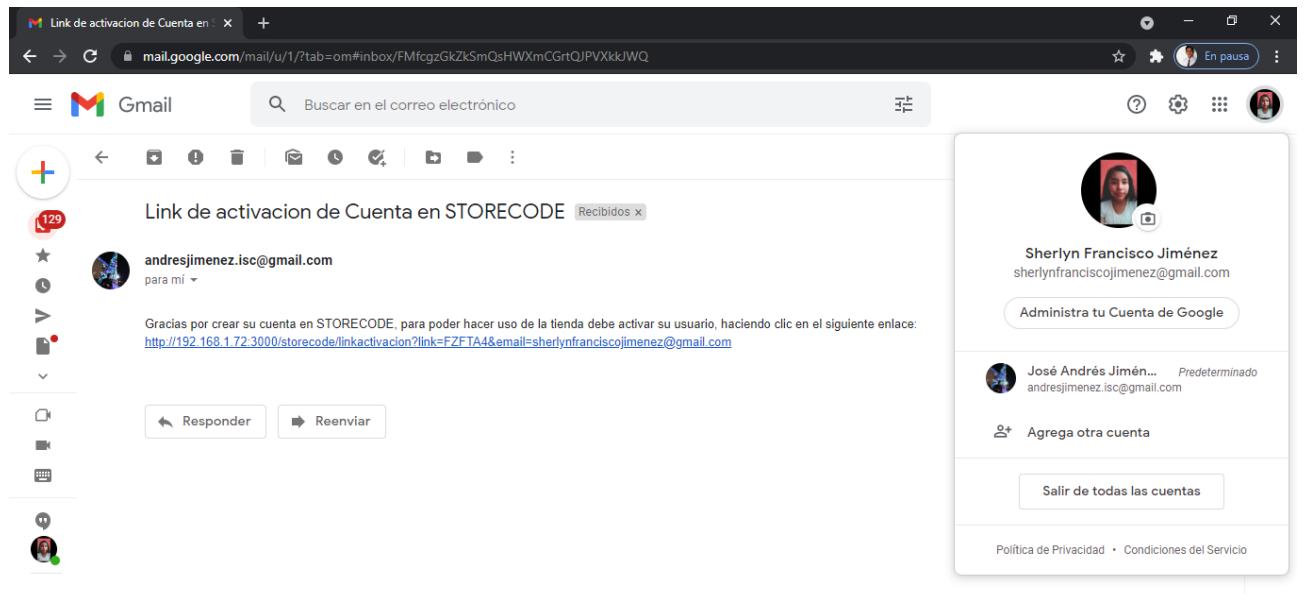


Figura 5.24 Gmail para la activación de la cuenta

B. Iniciar sesión

Pasos:

1. Ingresar el correo electrónico y contraseña, después dar clic en el botón “Iniciar sesión”. Así como se muestra en la figura 5.25.

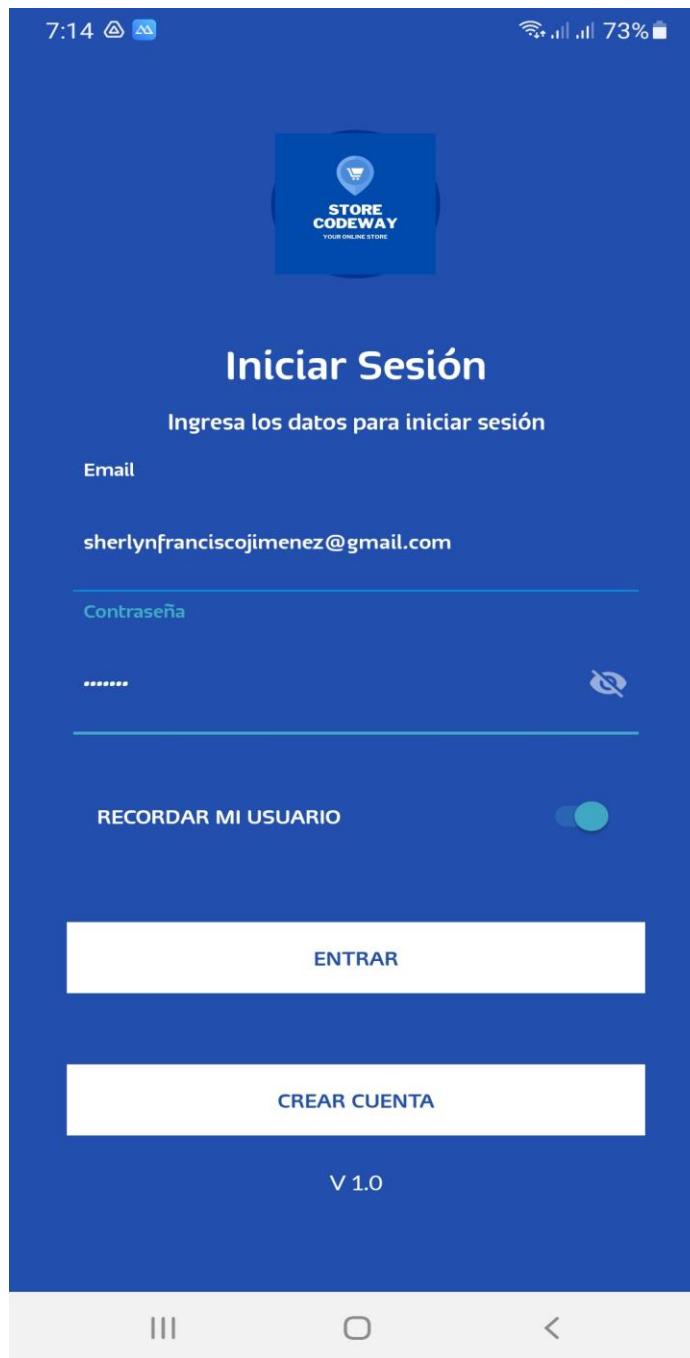


Figura 5.25 Pantalla de "Login".

2. Al iniciar sesión, los datos del usuario logeado se muestran en la pantalla de perfil, tal y como se ve en la figura 5.26.



Figura 5.26 Pantalla de "Perfil".

C. Publicar un producto

Pasos:

1. Dar click en el botón “Vender productos” de la pantalla de perfil y se desplegará la pantalla “Subir producto”, en la cual se deben otorgar permisos a la aplicación para acceder a la cámara.

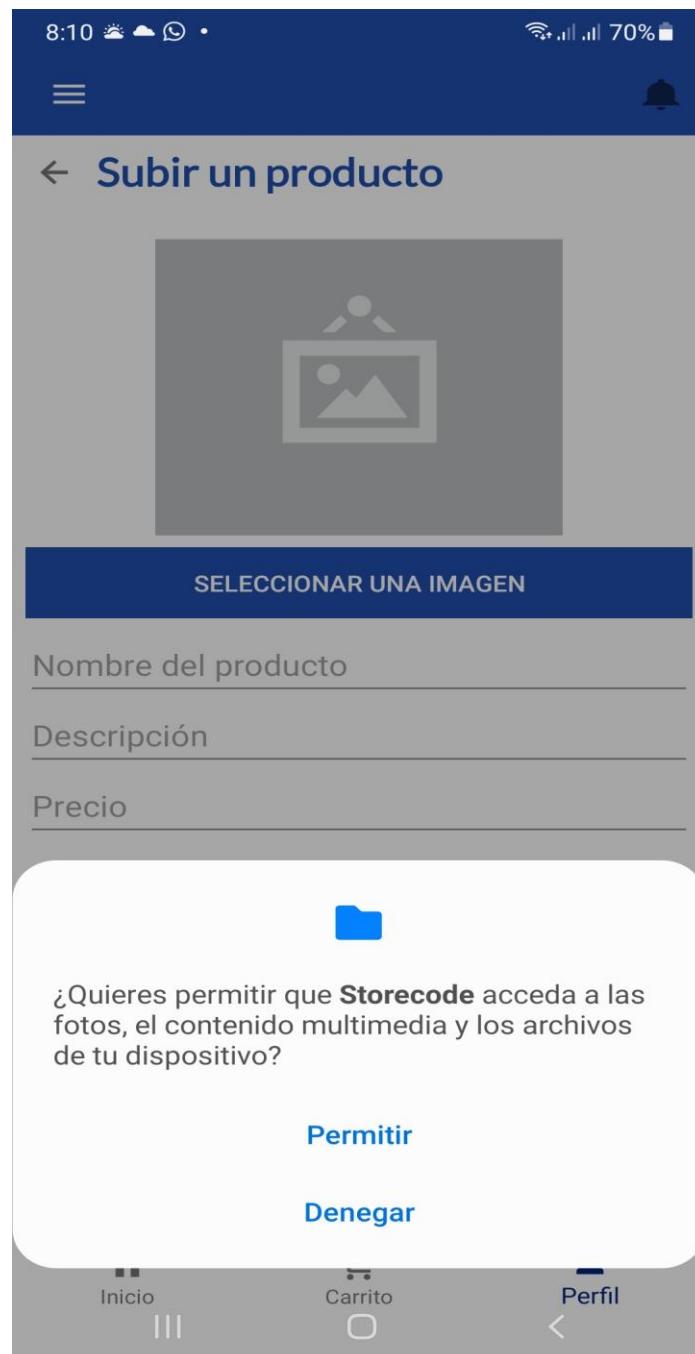


Figura 5.27 Pantalla modal para solicitar permiso de acceder a la cámara.

2. Dar click en el botón “Seleccionar imagen”, enseguida se abrirá la galería del dispositivo y desde ahí se selecciona la imagen deseada. Tal y como se muestra en la figura 5.28.

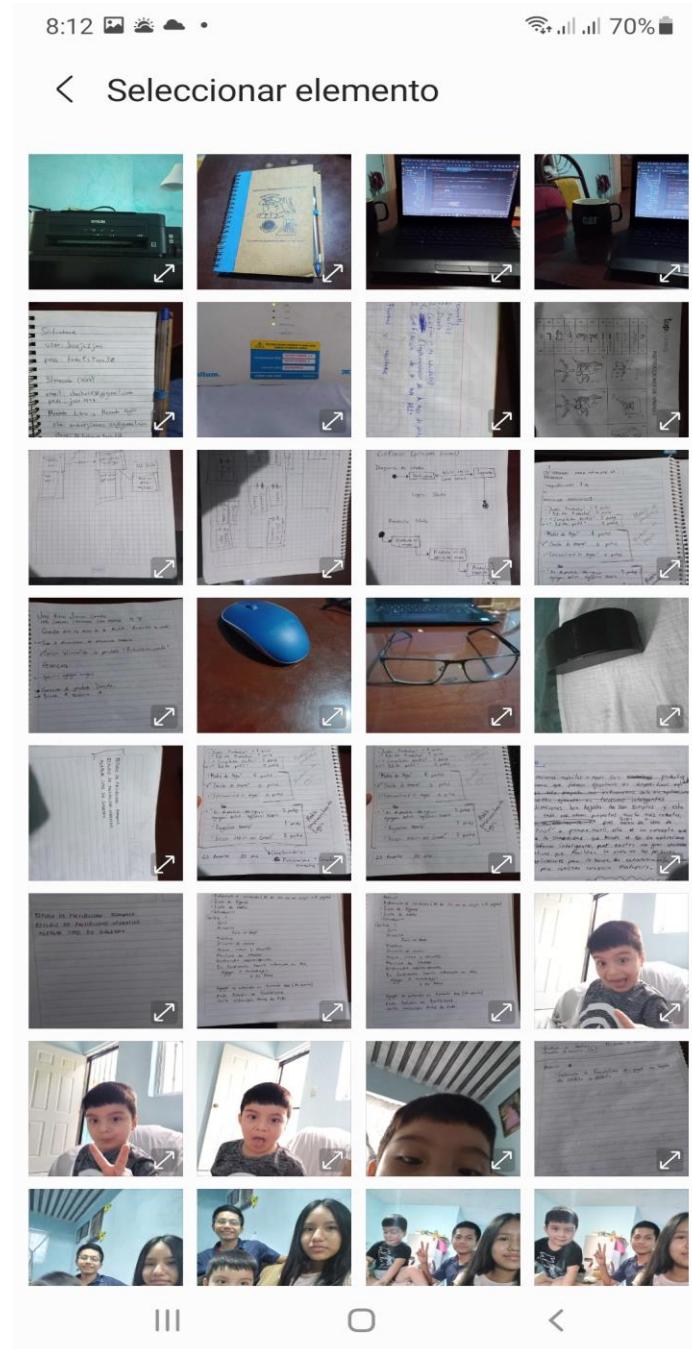


Figura 5.28 Galería del dispositivo

- Después de seleccionar la imagen, se llenan los demás datos solicitados por el formulario y finalmente se da click en el botón “Guardar producto”.



Figura 5.29 Pantalla "Subir producto".



4. El usuario puede ver sus productos publicados, dando click en el botón “Productos en venta” de la pantalla de “perfil”.

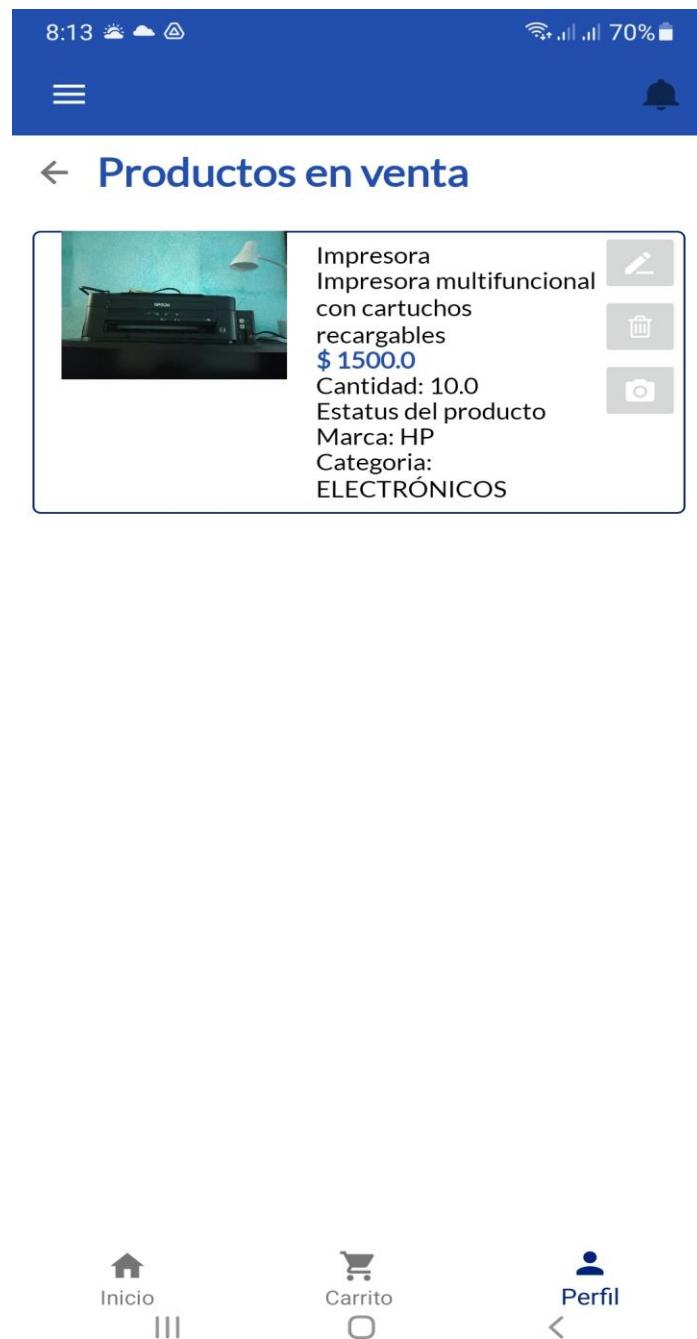


Figura 5.30 Pantalla "Productos en venta".

D. Comprar

Pasos:

1. Seleccionar el producto que se desea comprar, dando click sobre él desde la página de "Inicio".



Figura 5.31 Pantalla de "Inicio" cuando el usuario esta logeado.

2. Se muestra la pantalla de “Detalle del producto”, luego damos clic en el botón “Aregar al carrito”, Tal y como se muestra en la figura 5.32.



Figura 5.32 Pantalla "Detalle del producto".

3. Nos dirigimos a la pantalla “Carrito de compras” y ahí aparecerán todos los productos que hemos agregado. En esto caso se procede a comprar el Escritorio, dándole clic en botón “Pagar” que le corresponde a este producto.



Figura 5.33 Pantalla "Carrito de compras".

4. Se despliega una pantalla como la de la figura 5.34 que nos brinda las opciones de pago, en este caso se selecciona la opción “Nueva tarjeta”.

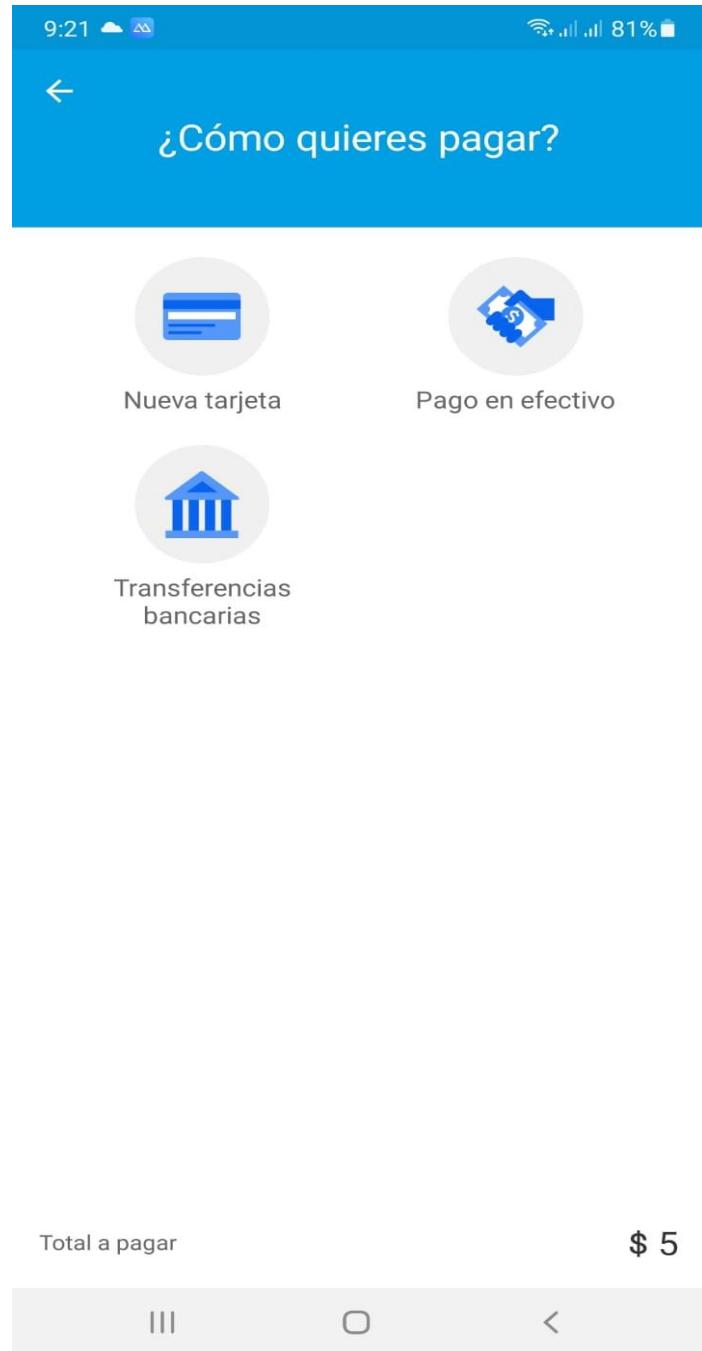


Figura 5.34 Primera pantalla del flujo de pago

5. Luego se selecciona la opción “Nueva tarjeta de débito”.

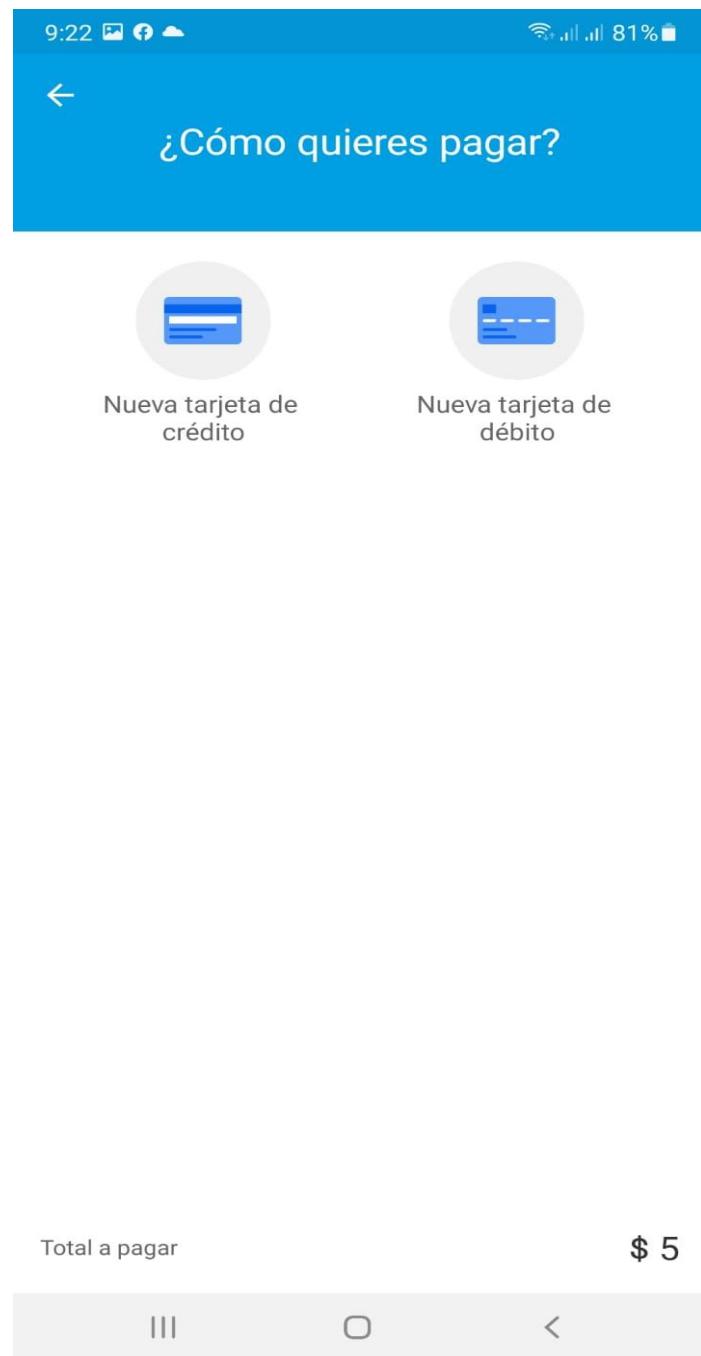


Figura 5.35 Tipo de tarjeta.

6. A continuación, se escribe el número de la tarjeta de débito, el nombre del titular, la fecha de vencimiento y el código de seguridad.



Nombre y apellido

Fecha de

SE ANDRES JIMENEZ CERVANTES



Figura 5.36 Datos de la tarjeta.

7. Se muestra un resumen del producto que se va a pagar y si los datos son correctos se da clic en el botón “Pagar”.

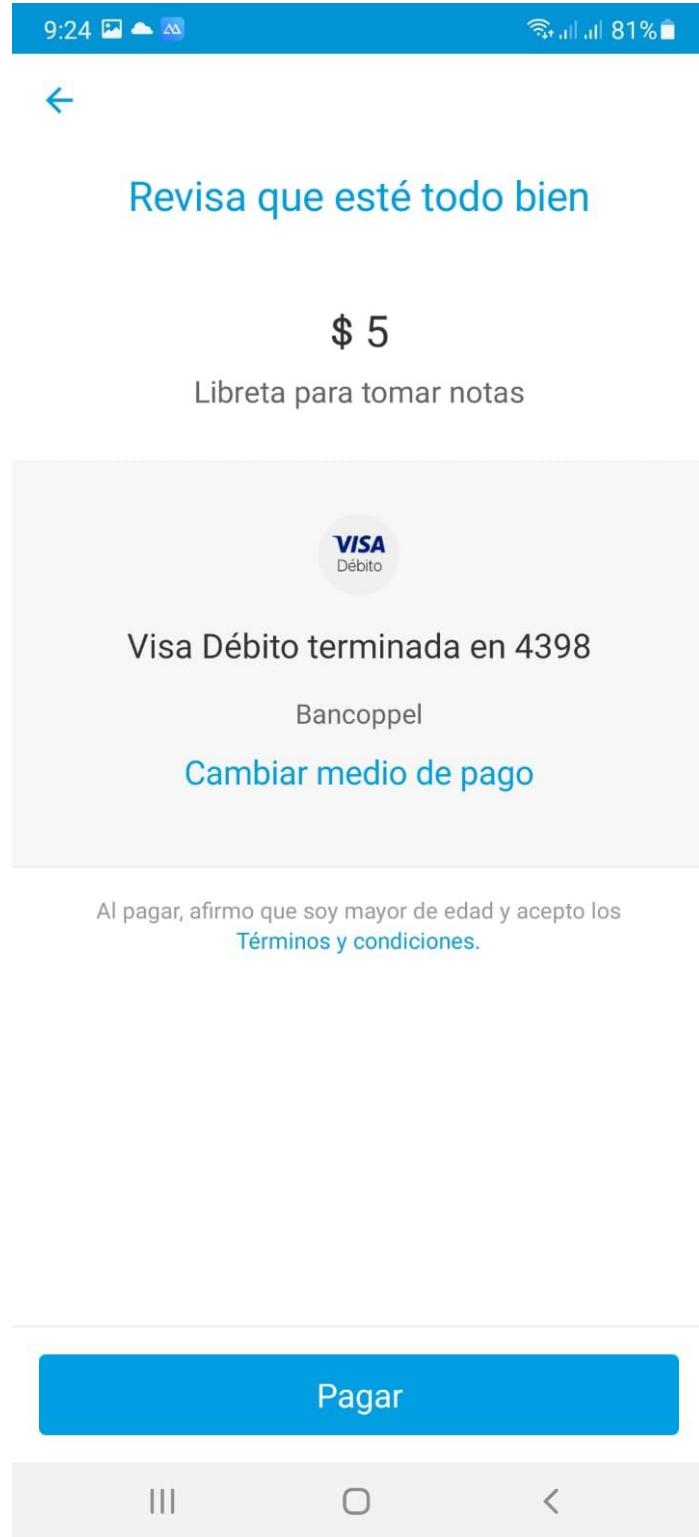


Figura 5.37 Resumen de compra.

8. Si el pago es acreditado, damos click en el botón “Continuar”, para proceder con la compra, de lo contrario si el pago no es acreditado, se deberá intentar pagar con una tarjeta diferente.

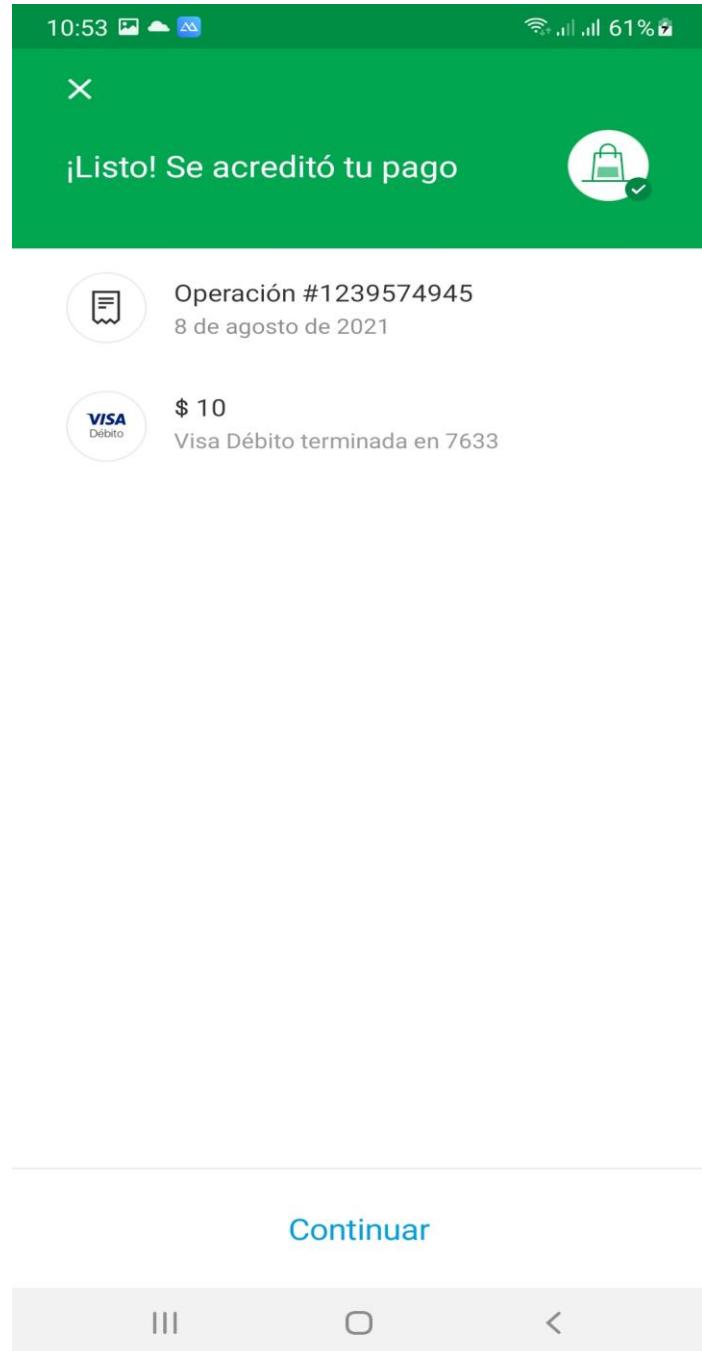


Figura 5.38 Pago acreditado.

9. Se concluye el flujo de pago y se muestra el mensaje “Tu pago fue aprobado”, lo cual quiere decir que ya se realizó la compra.

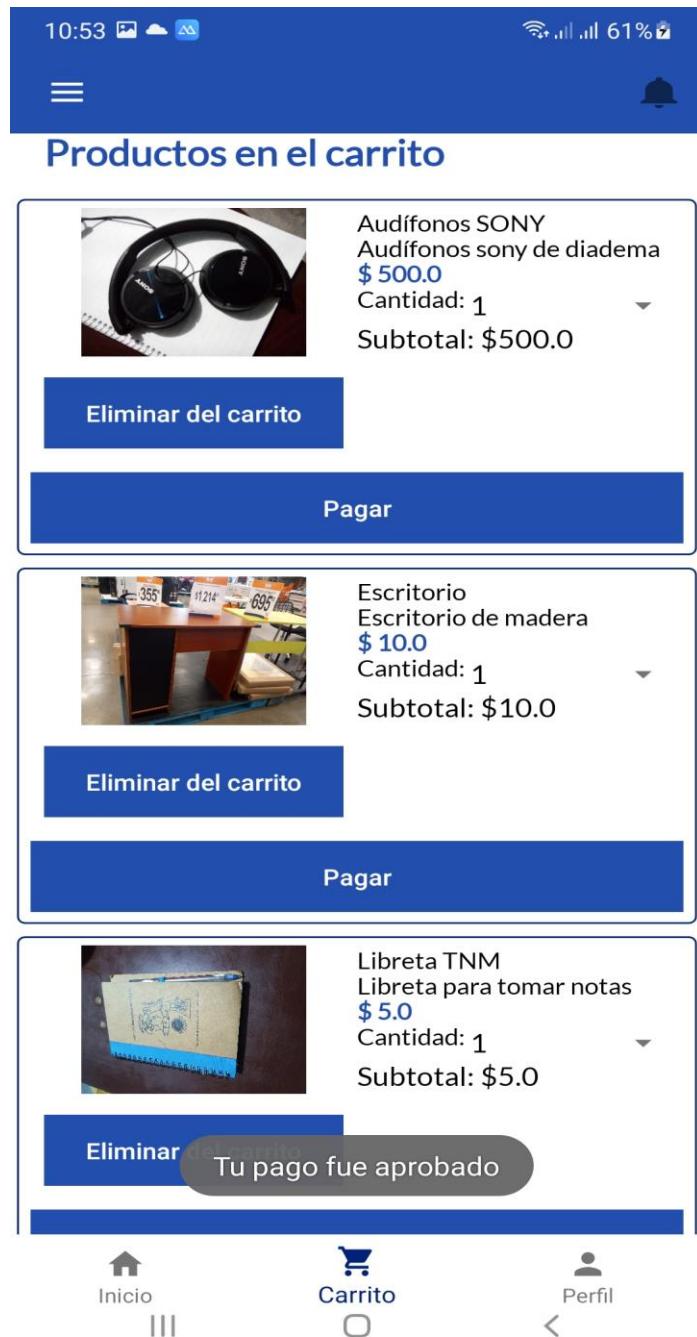


Figura 5.39 Termina el flujo pago.

10. Después de realizar la compra, llega una notificación con los detalles de la compra realizada.

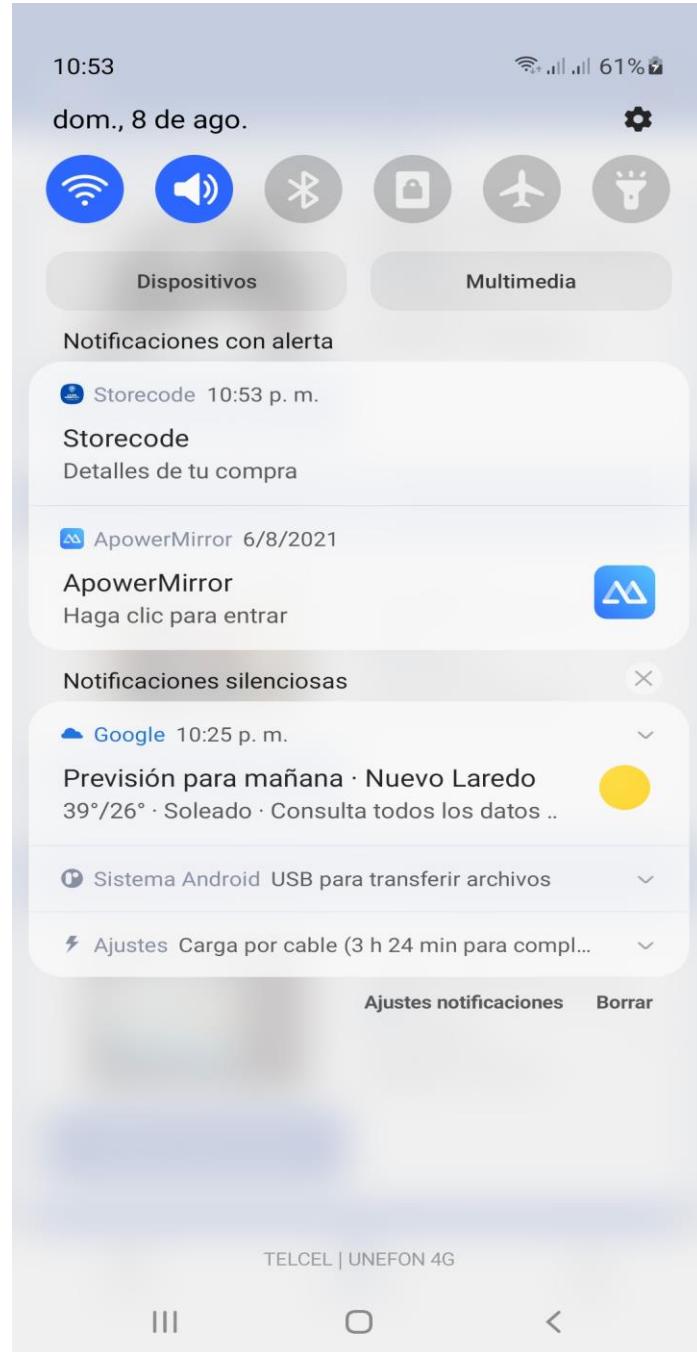


Figura 5.40 Barra de notificaciones

11. Por parte del vendedor, ya puede ver reflejado esta venta, desde su cuenta de mercado pago y ya cuenta con el dinero de la venta. Tal y como se muestra en la figura 5.41.

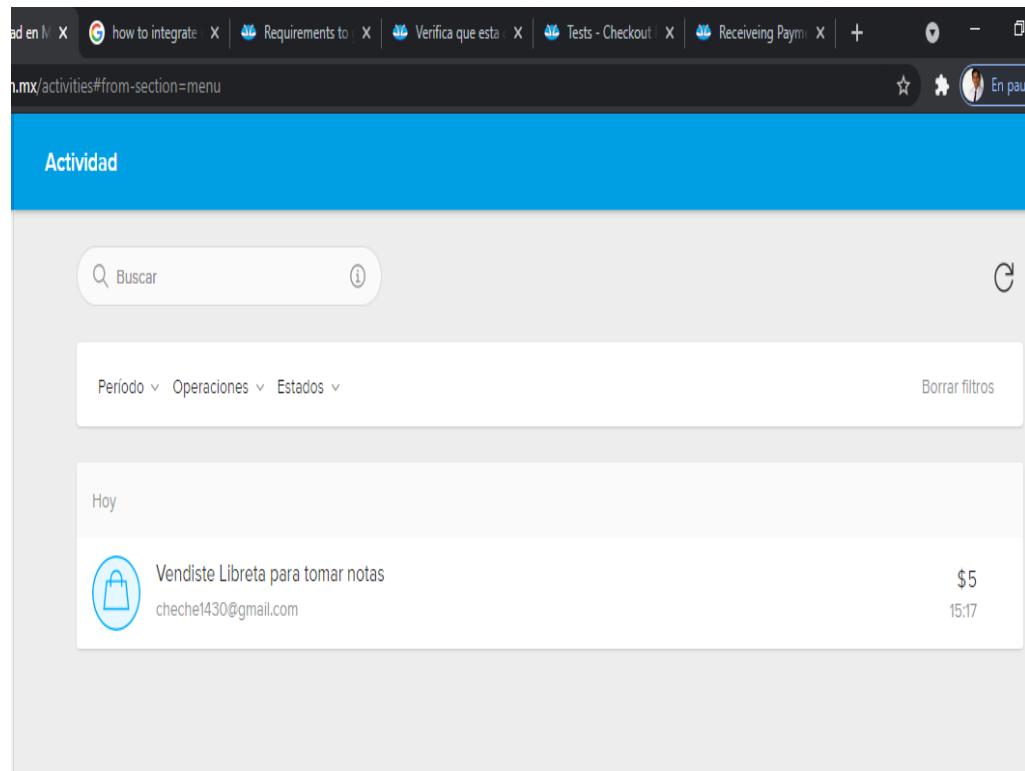


Figura 5.41 Actividad reflejada en la cuenta de mercado pago.

Así se concluyen las pruebas realizadas a la aplicación móvil desde un dispositivo físico, como se observa, la aplicación móvil se integró correctamente con los módulos de “Inicio de sesión”, módulo de productos y el módulo de usuarios. Así como también se integró correctamente con la API de mercado pago para permitir los pagos en línea.

Capítulo 6

Conclusiones, recomendaciones y experiencia profesional adquirida

Las aplicaciones móviles se encuentran al alcance de todas las personas, pues es muy sencillo instalarlas en sus smartphones, es por ello que se crean aplicaciones en diferentes ámbitos para darle mayor comodidad y agilidad a las personas al momento de realizar alguna tarea específica.

El objetivo de la empresa “Codeway Soluciones Integrales” al crear este producto de software es que las microempresas cuenten con una nueva herramienta digital, que les permita integrarse al comercio electrónico y así aumentar el volumen de sus ventas y ganancias, es por eso, que la aplicación móvil “Storecode” será de gran ayuda para las microempresas, ya que podrán realizar la venta de sus productos vía internet y los clientes no tendrán que acudir a una tienda física, tan solo con abrir la aplicación desde su teléfono celular, realizar el proceso de compra, pagar con tarjeta de débito o crédito y listo.

Mis recomendaciones para las personas que le darán mantenimiento a este producto de software es que sigan implementando la arquitectura de software en este proyecto, para que el código fuente se mantenga limpio y fácil de leer, además que al mantener una buena arquitectura, la aplicación será: fluida, rápida y mantenible.

En este periodo de residencia profesional logré adquirir experiencia laboral muy valiosa para mi crecimiento profesional. Aprendí a resolver problemas reales con la implementación de un producto de software, a trabajar en equipo, a pesar de trabajar en la modalidad home office, también mejore mis habilidades de comunicación y sobre todo mejoré mis habilidades como programador móvil. En mi estancia en la empresa “Codeway Soluciones Integrales” aprendí que el éxito se alcanza trabajando en equipo, teniendo compromiso y manteniendo una cultura de calidad.

6.2 Competencias desarrolladas y/o adquiridas.

Con el desarrollo de este Proyecto y con la experiencia que viví en mi residencia profesional, pude aplicar todos los conocimientos adquiridos a lo largo de mi Carrera y desarrollar nuevas competencias, pues pude resolver un problema real con un producto de software y así ayudar a los usuarios finales. Estas competencias las describo a continuación.

6.2.1 Competencias instrumentales

- Habilidades cognitivas, capacidad de análisis y comprensión de ideas.
- Habilidades de coordinación, capacidad de organización, toma de decisiones.
- Destrezas lingüísticas, capacidades de comunicación: expresión oral y escrita.
- Destrezas informáticas, manipulación y uso del equipo informático.

6.2.2 Competencias interpersonales

- Habilidades para socializar con el entorno.
- Habilidades para el trabajo en equipo, buena comunicación e interacción.
- Habilidades para trabajar en el ambiente laboral.

6.2.3 Competencias sistémicas

- Habilidades para la aplicación del conocimiento en entornos reales.
- Habilidades de investigación y documentación.
- Capacidad de adaptación a nuevos entornos.
- Capacidad de rápida reacción a problemas reales.
- Capacidad de la búsqueda de las mejores soluciones.

Capítulo 7

Bibliografía

Adeva, R. (3 de marzo de 2021). *adslzone.net*. Obtenido de <https://www.adslzone.net/reportajes/software/que-es-android/>

Armenta, M. H. (30 de Marzo de 2020). *forbes.com*. Recuperado el 11 de Marzo de 2021, de <https://www.forbes.com.mx/baja-conectividad-a-internet-otro-reto-que-enfrentan-las-microempresas-ante-coronavirus/>

Chojrin, M. (2019). *platzi.com*. Obtenido de platzi.com: <https://platzi.com/clases/1638-api-rest/21613-que-es-una-api-y-para-que-sirve/>

cs.us.es. (s.f.). cs.us.es. Recuperado el 30 de 11 de 2020, de cs.us.es: <http://www.cs.us.es/cursos/bd-2002/Theoria/Tema2.pdf>

ecured.cu. (s.f.). *ecured.cu*. Recuperado el 1 de 12 de 2020, de ecured.cu: <https://www.ecured.cu/Mysql>

G, Ó. M. (2001). *Comercio electrónico*. Lima: Universidad del Pacífico.

Morales, I. V. (2019). *platzi.com*. Obtenido de platzi.com: <https://platzi.com/clases/1566-bd/19813-que-son-y-cuales-son-los-tipos-de-base-datos-no-re/>

Morales, I. V. (s.f.). *platzi.com*. Recuperado el 30 de 11 de 2020, de platzi.com: <https://platzi.com/clases/1566-bd/19813-que-son-y-cuales-son-los-tipos-de-base-datos-no-re/>

Muradas, Y. (25 de Febrero de 2020). *openwebinars.net*. Obtenido de <https://openwebinars.net/blog/que-es-gradle/>
platzi.com. (s.f.). *platzi.com*. Obtenido de <https://platzi.com/desarrollo-android/>

Romero, G. (2019). *platzi.com*. Obtenido de platzi.com: <https://platzi.com/clases/1750-scrum/24280-introduccion-a-scrum/>

Salgado, A. (2018). *platzi.com*. Obtenido de platzi.com: <https://platzi.com/clases/1386-flutter/16255-que-es-flutter/>

Salgado, A. (s.f.). *platzi.com*. Recuperado el 04 de 07 de 2021, de platzi.com: <https://platzi.com/clases/1619-arquitectura-android/20847-patron-de-diseno-vs-arquitectura-de-diseno/>

Vega, F. (2017). *platzi.com*. Recuperado el 24 de 05 de 2021, de <https://platzi.com/clases/1098-ingeneria/6569-fundamentos-de-sistemas-operativos-moviles/>