UNIVERSIDAD PÚBLICA DE EL ALTO

CARRERA INGENIERÍA DE SISTEMAS

PROYECTO DE GRADO

SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE

CASO: Micromarket Home Service

Para Optar al Título de Licenciatura en Ingeniería de Sistemas
MENCIÓN: GESTIÓN Y PRODUCCIÓN

Postulante: Jaime Calamani Mamani

Tutor Metodológico: M. Sc. Ing. Enrique Flores Baltazar

Tutor Revisor: Lic. Maria Magdalena Aguilar Guanto

Tutor Especialista: Ing. Hernan Quispe Conurana

EL ALTO – BOLIVIA

2022

DECLARACIÓN JURADA DE

AUTENTICIDAD Y RESPONSABILIDAD

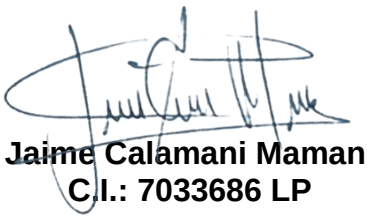
Yo Jaime Calamani Mamani estudiante con C.I. 7033686LP mediante la presente **declaro** de manera pública que la propuesta del **PROYECTO DE GRADO** titulada **“SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE” CASO: Micromarket Home Service** es original, siendo resultado de mi trabajo personal y no constituye una copia o replica de trabajos similares elaborados.

Autorizo la publicación del resumen de mi propuesta en internet y me comprometo a responder a todos los cuestionamientos que se desprenden de su lectura.

Asimismo, me hago responsable ante la universidad o terceros, de cualquiera irregularidad o daño que pudiera ocasionar, por el incumplimiento de lo declarado.

De identificarse falsificación, plagio, fraude, o que el **PROYECTO DE GRADO** haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, responsabilizándome por todas las cargas legales que se deriven de ello sometiéndome a las normas establecidas y vigentes de la Carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

El Alto, diciembre del 2022.



Jaime Calamani Mamani
C.I.: 7033686 LP

DEDICATORIA

Dedico el presente proyecto con mucho cariño:

A Dios por brindarme la fortaleza suficiente para afrontar los obstáculos en el camino hacia mis objetivos.

A mis padres Martina y Romulo por el apoyo incondicional y por el sustento desinteresado, por el constante sacrificio que pocas veces se los reconocí, por la comprensión a mis decisiones.

A mis hermanos(as) Amalia, Juana, Francisco, por ser mis mentores de la vida, por el apoyo moral, por brindarme el empuje suficiente en los momentos complicados.

A mis sobrinos(as) por ser el motor de inspiración para el cumplimiento de mis objetivos.

A mis tutores por guiarme e inculcarme sus conocimientos para el desarrollo y culminación del mismo.

AGRADECIMIENTOS

A Dios, por cuidarme en los momentos adecuados, por brindarme el apoyo inagotable en cada momento de mi vida.

A mis padres, por ser un ejemplo de sinceridad, solidaridad, respeto, honestidad, responsabilidad, los cuales los tendré presente en todo momento.

Agradecer a mis distinguidos tutores:

A mi tutor metodológico ***M. Sc. Ing. Enrique Flores Baltazar****, por su voluntad incondicional de brindar su tiempo, conocimiento, apoyo.*

A mi tutor especialista ***M. Sc. Ing. Hernan Quispe Conurana****, por aceptar ayudarme sin condiciones ni excusas, por brindarme sus conocimientos, por sacarse tiempo a pesar del cansancio.*

A mi tutor revisor ***Lic. Maria Magdalena Aguilar Guanto****, por ser una amiga, por tener esa disponibilidad inmediata de colaboración, por orientarme de forma acertada y detallada, por sus observaciones brindadas en el proceso de la realización del presente proyecto.*

A la ***Universidad Pública de El Alto****, a la carrera Ingeniería de Sistemas por acogerme en sus aulas en los años de estudio.*

Y a mis compañeros(as) por su amistad, su alegría, sus experiencias, durante los años de estudio.

¡Muchas gracias!

RESUMEN

El avance de la tecnología crece a pasos agigantados, lo que implica una revolución única en todas las áreas. Las instituciones para no quedarse aisladas de la era de la información optan por implementar un sistema de información para ahorrar en tiempo y eficiencia.

Por tal motivo el Micromarket Home Service, vio la opción de una aplicación móvil para el cliente, y sistema web para el lado del administrador, con lo cual se administra el manejo de la información de mejor forma, como en la toma de decisiones, además rompe la forma tradicional de operar en el proceso de ventas.

Por tal motivo el proyecto titulado “sistema de información web y aplicación móvil para la administración de inventario, ventas y compras online” caso: Micromarket Home Service, cumplirá con los objetivos deseados.

Para su desarrollo se aplicó la metodología UWE, que permite modelar el comportamiento del sistema en modo representación gráfica, como también la metodología Mobile-D para la App, el Backend se desarrolló con lenguaje TypeScript con Framework Nest.js con ORM TypeORM con gestor de base de datos PostgreSQL, el lado del Frondend web con Framework Angular, Angular Material, y la aplicación móvil con Android Studio con lenguaje java con base de datos SQLite con librería Room.

Además, se utilizó para el análisis de Métricas de calidad la ISO 25000 y para la seguridad la ISO 27000.

Finalmente, para el análisis de estimación de costos se aplicó el método COSMIC.

Palabras Clave: Sistema, UWE, COSMIC, ISO, API REST, SERVICIOS WEB.

ABSTRACT

The advancement of technology is growing by leaps and bounds, which implies a unique revolution in all areas. In order not to be isolated from the information age, institutions choose to implement an information system to save time and efficiency.

For this reason, the Micromarket Home Service, saw the option of a mobile application for the client, and a web system for the administrator's side, with which the handling of information is managed in a better way, as in decision making, in addition breaks the traditional way of operating in the sales process.

For this reason, the project entitled "web information system and mobile application for inventory management, sales and online purchases" case: Micromarket Home Service, will meet the desired objectives.

For its development, the UWE methodology was applied, which allows modeling the behavior of the system in graphical representation mode, as well as the Mobile-D methodology for the App, the Backend was developed with TypeScript language with Framework Nest.js with ORM TypeORM with manager of PostgreSQL database, the Frondend web side with Angular Framework, Angular Material, and the mobile application with Android Studio with java language with SQLite database with Room library.

In addition, ISO 25000 was used for the analysis of quality metrics and ISO 27000 for security.

Finally, for the cost estimation analysis, the COSMIC method was applied.

Keywords: System, UWE, COSMIC, ISO, API REST, WEB SERVICES.

LISTA DE ABREVIATURAS

UWE: Basado en el Proceso Unificado y UML.

COSMIC: Consorcio Internacional de Medición de Software en General.

ISO: Organismo Internacional de Estandarización.

SQUARE: Evaluación de Calidad de Productos Software.

IEC: Comisión Electrotécnica Internacional.

SGSI: Sistema de Gestión de Seguridad de la Información.

IDE: Entorno de Desarrollo Integrado.

API: Interfaz de programación de aplicaciones.

REST: Transferencia de representación de estado.

HTTP: Protocolo de transferencia de hipertexto.

POO: Programación orientada a objetos.

ORM: Mapeo Objeto Relacional.

JWT: Json Web Token.

HTML: Lenguaje de Marcas de Hipertexto

POJO: Un objeto Java Plano Antiguo.

URL: Localizador de Recursos Uniforme.

MVC: Modelo Vista Controlador.

APP: Aplicación.

GUI: Interfaz gráfica de usuario.

SDK: Kit de desarrollo de software.

VCS: Sistema control de versiones

ÍNDICE GENERAL

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

CAPÍTULO I

MARCO PRELIMINAR

1.1. INTRODUCCIÓN

En los últimos años el crecimiento acelerado de la tecnología ha revolucionado el mundo y por tal motivo esto conlleva a desarrollar sistemas informáticos cada vez más complejos, acorde a la necesidad de una sociedad en constante transformación. Hoy en día el uso de los teléfonos móviles inteligentes de gama alta en todos los estratos de la población es de manera continua, siendo esto necesario para distintos fines como investigación, consultas web, comunicación y otros. Es difícil no darse cuenta que al brindarte comodidad, control, supervisión, automatización y optimización son procesos que mejoran de una manera impresionante la calidad de vida, trabajo y estudio de la gente.

En la actualidad con el crecimiento de la tecnología es imprescindible que toda organización, institución, microempresa, pequeños negocios, cual fuera el campo donde ellos se dediquen. Sea de brindar servicios o ventas, están obligados no solamente estar presentes en sitios físicos como oficina, negocio, sino que también necesitan interacción online. En un entorno cada vez más competitivo, las herramientas tecnológicas se han constituido como elementos indispensables para optimizar, mejorar los procesos de acción y comunicación con los clientes potenciales.

El objetivo que persigue el presente proyecto de grado es brindar una solución tecnológica en la Micromarket Home Service, según la necesidad de la sociedad, que sea un facilitador e intermediario entre cliente y producto, que sea de acceso rápido y que esté disponible en la red, basado en un sistema de información web y aplicación móvil para la administración de inventario, ventas y compras online. Que permita un mejor control en cuanto a las fechas, el stock de productos, después de una jornada de ventas.

Para lo cual se seleccionó las metodologías de desarrollo de software: Mobile-D para la aplicación móvil y UWE para el sistema web. Y las herramientas respectivas: Android Studio, angular, Nest.js, TypeScript, Node.js, PostgreSQL, SQLite.

1.2. ANTECEDENTES

1.2.1. Antecedente institucional

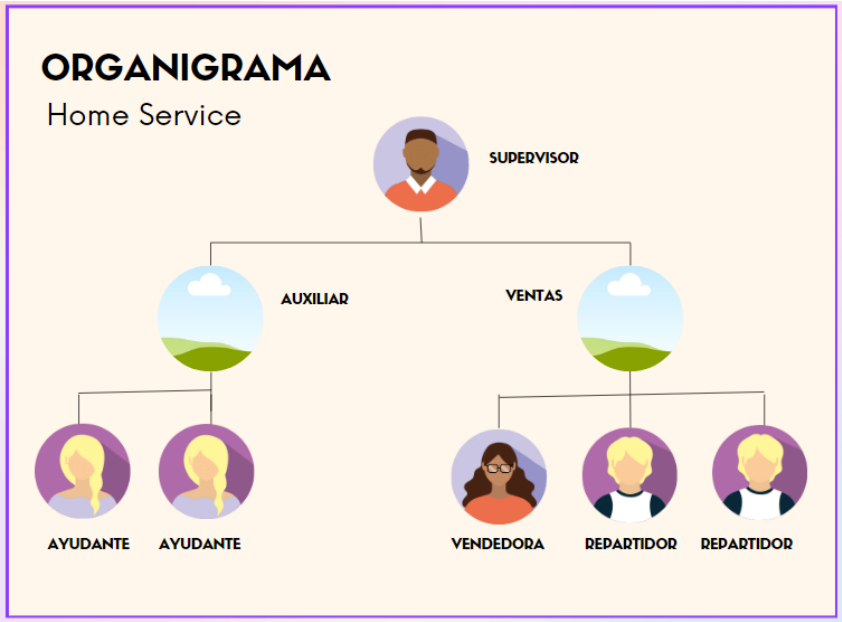
La Micromarket se encuentra ubicada en la Zona Zenkata en la Ciudad de El Alto, se creó hace 3 años, se dedica a la venta de productos de consumo masivo, es una iniciativa reciente, con gran proyección a futuro. Como la mayoría de los negocios o emprendimientos tiene un comienzo pequeño, no es diferente la situación de la Micromarket Home Service que todavía está dando sus primeros pasos como una alternativa más para las familias alteñas para que puedan proveerse de alimentos más demandados cotidianamente en cada hogar, actualmente cuenta con personal de confianza cercana, pero se apuesta por objetivos más grandes, por lo cual él supervisor está convencido que estamos en un buen camino.

Misión: Nuestra misión es brindar servicio de venta de productos de consumo masivo a hogares alteños.

Visión: Nuestra visión es establecerse como una de las grandes Micromarket en nuestro rubro.

Figura 1.1

Organigrama



Nota. Organigrama de la Micromarket Home Service.

1.2.2. Antecedentes internacionales

- ✓ Se cita el trabajo realizado por: **(Brito & Pinzon, 2016, p.17),** que titula **"*Diseño de una aplicación móvil para la oferta de servicios de información (tendencias, precios y ubicación) enfocado a las prendas de vestir, accesorios y calzado*".** Universidad Libre, Bogotá, Colombia.

  Suitty App proporciona información a los usuarios para que puedan filtrar sus compras según su necesidad bien sea de tendencia enseñando la prenda adecuada, o puede ser de precio conociendo con anticipación si esta se ajusta a sus presupuestos. También aplica para ubicación suministrando el dato de donde se puede conseguir el outfit, en tiempos modernos y vertiginosos el tiempo se convierte en un recurso valioso para el hombre. Para este fin se utilizó plataforma Android y como gestor de base de datos Sqlite.
- ✓ Se cita el trabajo realizado por: **(Romero, 2017, p.7),** que titula **"*Desarrollo de aplicación móvil de compraventa y subasta online de aparatos electrónicos*".** Universidad Oberta de Catalunya, España.

  Existen aplicaciones de venta directa con precio fijo y otras en modo subasta, en el caso de las primeras además facilitan la mensajería instantánea para la comunicación entre usuarios y búsquedas de artículos por localización geográfica. En el caso de la app se basa exclusivamente en los aparatos electrónicos, en donde se facilita información detallada de los dispositivos mediante enlaces a la web de fabricante, y así analizar y comparar entre distintas marcas y modelos. El usuario tiene la posibilidad de visualizar los artículos que han incluido otras personas y ponerse en contacto con ellas mediante un chat y si llegan a un acuerdo realizar el pago por la plataforma o utilizar la opción de localización para conocer la distancia entre los interesados, además puede dar de alta tantos productos como necesite. En cuanto a la finalidad de la propia herramienta es lograr una app de fácil acceso y usuarios móviles, que les ayude a la compra de nuevos aparatos electrónicos y a la venta de los que ya no utilicen o quieran desprenderse.

Para lo cual se desarrolló en plataforma Android integrada con gestor de base de datos Sqlite.

1.2.3. Antecedentes nacionales

- ✓ Se cita el trabajo realizado por: **(Hilaquita, 2014, p. 2),** que titula **"*Aplicación móvil para la publicación de la revista emistur sobre lugares turísticos de Bolivia, utilizando la tecnología realidad aumentada*"**. Universidad Mayor de San Andres, La Paz, Bolivia.

  El presente proyecto se apoya de las teorías y prácticas de la realidad aumentada, una tecnología que nos permite mezclar la realidad virtual con la realidad con la que vivimos, en este caso esta aplicada para la presentación de publicaciones de la revista Emistur, de lugares turísticos de Bolivia. Para lo cual es importante el apoyo de nuevas tecnologías como el uso de las herramientas de desarrollo de Metaio Mobile SDK, la cual es una librería de realidad aumentada de código libre. Para lograr este fin se utilizó plataforma Android, Vuforia, Unity 3D.
- ✓ Se cita el trabajo realizado por: **(Cadima, 2013, p. 1),** que titula **"*Desarrollo de una tienda virtual mediante el estudio comparativo de una tienda física de productos de computación*"**. Universidad Mayor de San Andrés, La Paz, Bolivia.

  La evolución de la tecnología ha guiado al comercio por nuevos caminos y horizontes, abriéndose paso dentro del internet, revolucionando la forma de comprar y vender, mediante este ámbito surgió el e-Commerce (comercio electrónico). Por ello este trabajo investigativo tiene como objetivo la implementación de una tienda virtual, con el propósito de mejorar la forma de mostrar los productos de una tienda física, aumentando de alguna manera las ventas de dicho negocio. La cual va a estar en mejores condiciones que una tienda física, el cual busca la comodidad de la empresa tanto como la del cliente, acortando distancias, disminuyendo costos y ofreciendo todo tipo de productos electrónicos, lo cual incrementa la confianza y el flujo de comercio en la red. Para el cumplimiento del objetivo se utilizó Php como lenguaje de programación, Mysql como base de datos.

2.2.3. Antecedentes locales

- ✓ Se cita el trabajo realizado por: **(Poma, 2019, pp. 1-2),** que titula **"*Aplicación y portal web para el diagnóstico preventivo y desarrollo cognitivo verbal de niños de 3 a 7 años con trastorno del espectro autista"***. Universidad Pública de El Alto, El Alto, Bolivia.

  Todos establecemos una gran dicotomía desde que nacemos, el mundo de las personas y el mundo de los objetos, para los niños con trastorno espectro autista esa línea divisoria no es tan clara, eso es lo que sucede en su mente y eso sucede con su cerebro conforme se bifurca el camino del aprendizaje si fuésemos capaces de reconocer a tiempo intervenir y tratar podemos mitigar absolutamente sus efectos.

  La institución educativa tenía la necesidad de una herramienta que permitiera el desarrollo verbal en niños de 3 a 7 años y una plataforma dedicada a padres y profesores para así tener un entorno educativo de apoyo homogéneo y constante en la casa y en el centro de educación. Con el uso de pictogramas y la tecnología móvil, el portal web por otro lado por su capacidad de alcance puede considerarse también como un medio para difundir los síntomas y conductas. También se utilizó herramientas como el Android Studio para la aplicación.
- ✓ Se cita el trabajo realizado por**: (Ochoa, 2018, pp. 1-2),** que titula **"*Desarrollo de portal web y aplicación móvil para la estructuración de programas radiales*"**. Universidad Pública de El Alto, El Alto, Bolivia.

  El objetivo principal es proporcionar al radio escucha para que ellos pueden ayudar a estructurar la programación de la radio mediante sus sugerencias con respecto a la programación ya existente en la radio. De esta manera la empresa de radio difusión tendrá una perspectiva mucho más clara de lo que el radioescucha espera de la radio. También se desarrolla el sistema de mensajería en tiempo real con la ayuda de la tecnología socket que nos proporciona node.js con su librería socket.io, la transmisión de la señal de audio Streaming configurada con el servidor icecast2.

1.3. PLANTEAMIENTO DEL PROBLEMA

El Micromarket Home Service se encuentra ubicado en la Ciudad de El Alto, actualmente tiene como finalidad brindar el servicio de venta de productos de consumo masivo.

Asimismo, se constató por medio de entrevista, observación, cuestionario con el propietario de la Micromarket, que no cuenta con la información organizada, coherente, oportuna, y el procesamiento se realiza de forma manual, el registro de ventas, compras, inventarios se efectúa con el uso de planillas electrónicas lo que repercute en tiempos innecesarios al momento de verificar las fechas de vencimiento, la disponibilidad de stock, salida de productos, como también a veces ocasiona perdida de documentos, complicación en la obtención de datos, deterioro de algunos productos, esto trae consigo pérdida de tiempo en la búsqueda de los productos, desencadenando una serie de confusiones. También la forma tradicional del proceso de ventas ofusca de cierta manera al negocio.

1.3.1. Problema principal

El manejo de la información y registros, de forma manual como el proceso tradicional, de ventas compras inventario en la Micromarket Home Service, repercute en pérdida de tiempo, de productos, de información importante de clientes, disminuyendo la efectividad de cierta manera al negocio.

1.3.2. Problemas secundarios

- ✓ El manejo manual de la información en la Micromarket tiene una repercusión en los datos, en algunas ocasiones se pierda o se mezcle en medio de otros documentos.
- ✓ La verificación de fechas de vencimiento, entrada y salida de productos de forma manual, es moroso, en algunos casos provoca la obtención de datos incorrectos.
- ✓ La manera tradicional o común de operar el proceso de ventas disminuye la efectividad de cierta manera al negocio.

- ✓ La demora en el procesamiento de datos repercute en las ventas ya que no hay como tener certeza de que productos tienen más preferencia.

1.3.3. Formulación del problema

Analizando los problemas nos planteamos la siguiente interrogante:

¿Cómo un sistema de información web y aplicación móvil para la administración de inventario, ventas y compras online ayudaría a un adecuado registro de información de clientes, proveedores, productos, de esta manera cambiar lo tradicional en la Micromarket Home Service?

1.4. OBJETIVOS:

1.4.1. Objetivo general

Diseñar un sistema de información web y aplicación móvil para la administración de inventario, ventas y compras online para Micromarket Home Service, que apoye en el procesamiento de datos de clientes, proveedores, productos y permita cambiar la forma tradicional de venta.

1.4.2. Objetivos específicos

- ✓ Realizar un módulo de registro de clientes, productos, proveedores, ventas, compras e inventario, para mejorar la organización de los documentos y evitar pérdida de información.
- ✓ Controlar las fechas de vencimiento, entrada y salida de productos para no tener confusiones ni datos erróneos.
- ✓ Desarrollar el módulo de pedidos, ventas, para mejorar el servicio como el control de los procesos.
- ✓ Generar reportes de ventas realizadas, reportes clientes, proveedores, productos, compras, para obtener credibilidad en la información.
- ✓ Implementar módulos en la app móvil para seleccionar categorías, productos, envío de pedido, agregar carrito, favoritos, historial pedidos, reclamos.

1.5. JUSTIFICACIÓN

1.5.1. Justificación técnica

El trabajo se justifica técnicamente porque utiliza métodos, técnicas y herramientas, acorde para el desarrollo óptimo de software, Mobile-D y UWE como metodologías, además lenguaje Java para el entorno de la aplicación, como Framework NestJS, TypeORM para el backend, y para el frontend Framework Angular, Angular Material con lenguaje de programación TypeScript para el lado administrador.

Por otra parte, la app contribuirá al cliente para que realice pedidos online de tal manera que sea accesible y cómodo, como también para el administrador que tendrá la facilidad de gestionar los registros de ventas, compras, inventario, reportes en la Micromarket.

1.5.2. Justificación económica

El proyecto se justifica económicamente porque la app y la web se desarrollará bajo la licencia open source, minimizando costos en cuanto a su adquisición.

También mejorará los tiempos, en la consulta de datos, generación de reportes, lo que conlleva reducción de costos, como la posibilidad de generar más ventas.

1.5.3. Justificación social

Se justifica socialmente ya que la aplicación móvil permitirá a los clientes acceder desde su comodidad al momento de realizar sus pedidos, y la variedad de productos que tendrá a disposición será considerable.

1.6. METODOLOGÍA

1.6.1. Metodología de desarrollo de software

1.6.1.1. Mobile-D

A la hora de elegir una metodología a seguir para realizar un desarrollo de una aplicación móvil, nos podemos encontrar varias, como por ejemplo SCRUM, Extreme Programming, etc. Pero me ha llamado la atención la metodología ágil Mobile-D, que se ajusta mucho a este tipo de desarrollos. (Cruz, 2014, p. 34).

Esta metodología está basada en diversas tecnologías como Rational Unified Process, Extreme Programming y Crystal Methodologies, y su finalidad es intentar obtener pequeños ciclos de desarrollo de forma rápida en dispositivos pequeños.

Un ciclo de proyecto con la metodología Mobile-D está compuesto por cinco fases:

- ✓ Fase de Exploración
- ✓ Fase de inicialización
- ✓ Fase de producción
- ✓ Fase de estabilización
- ✓ Fase de pruebas

1.6.1.2. UWE

La propuesta de ingeniería web basada en UML (UWE (Koch, 2000)) es una metodología detallada para el proceso de autoría de aplicaciones con una definición exhaustiva del proceso de diseño que debe ser utilizado. Este proceso, iterativo e incremental, incluye flujos de trabajo y puntos de control, y sus fases coinciden con las propuestas en el Proceso Unificado de Modelado. (Huanca, 2015, p. 32).

La metodología UWE está compuesto por las siguientes fases:

- ✓ Análisis de Requisitos
- ✓ Diseño Conceptual
- ✓ Diseño Navegacional
- ✓ Diseño de Presentación

1.6.1.3. Métricas de calidad

ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software. (ISO 25000, 2021).

La familia ISO/IEC 25000 es el resultado de la evolución de otras normas anteriores, especialmente de las normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto software, e ISO/IEC 14598, que abordaba el proceso de evaluación de productos software. Esta familia de normas ISO/IEC 25000 se encuentra compuesta por cinco divisiones.

Las características de calidad y sus mediciones asociadas pueden ser útiles no solamente para evaluar el producto software sino también para definir los requerimientos de calidad.La serie ISO/IEC 25000:2005 reemplaza a dos estándares relacionados: ISO/IEC 9126 (Software Product Quality) e ISO/IEC 14598 (Software Product Evaluation). (ISO 25000, 2021).

- ✓ ISO/IEC 2500n. División de gestión de calidad.
- ✓ ISO/IEC 2501n. División del modelo de calidad.
- ✓ ISO/IEC 2502n. División de mediciones de calidad.
- ✓ ISO/IEC 25030. División de requisitos de calidad.
- ✓ ISO/IEC 25040. División de evaluación de la calidad.

1.6.1.4. Estimación de costo

a). Método COSMIC

El tamaño de un software es la principal variable necesaria para determinar el esfuerzo de desarrollo que deberá invertirse para implementarlo. La medición y estimación de software utilizando COSMIC, es un método de segunda generación que determina el tamaño del software a partir del número de interacciones entre los componentes de los requerimientos funcionales.

Estandarizado bajo la ISO 19761, el método COSMIC puede aplicarse a diversos tipos de software, incluyendo aplicaciones de negocios, sistemas de información gerencial, software en tiempo real, infraestructura, e inclusive software científico y de ingeniería. (Vanquez, 2015, pp. 4-7).

Tiene la ventaja que no establece límites arbitrarios al tamaño funcional, por lo cual pueden medirse componentes de software independientemente de si son muy grandes o pequeños. (Vanquez, 2015, pp. 4-7).

1.6.1.5. Seguridad

Las normas que forman la serie ISO/IEC-27000 son un conjunto de estándares creados y gestionados por la Organización Internacional para la Estandarización (ISO) y la Comisión Electrónica Internacional (IEC). Ambas organizaciones internacionales están participadas por multitud de países, lo que garantiza su amplia difusión, implantación y reconocimiento en todo el mundo.

Las series 27000 están orientadas al establecimiento de buenas prácticas en relación con la implantación, mantenimiento y gestión del Sistema de Gestión de Seguridad de la Información (SGSI) o por su denominación en inglés Information Security Management System (ISMS). Estas guías tienen como objetivo establecer las mejores prácticas en relación con diferentes aspectos vinculados a la gestión de la seguridad de la información, con una fuerte orientación a la mejora continua y la mitigación de riesgos. (ISO27000.ES, 2018)

1.6.1.6 Pruebas de software

a). Prueba de caja blanca (estructural, caja transparente)

Las pruebas de caja blanca (también llamadas estructurales o de caja transparente) describen pruebas o métodos en los que se conocen los detalles y el funcionamiento interno del software que se está probando.

Dado que conoce las funciones, los métodos, las clases, cómo funcionan y cómo se unen, generalmente está mejor equipado para examinar la integridad lógica del código. Por ejemplo, es posible que sepa que hay una peculiaridad en la forma en que un determinado idioma maneja ciertas operaciones. Podría escribir pruebas específicas para eso, que de otro modo no sabría escribir en un escenario de caja negra.

Las pruebas unitarias y las pruebas de integración suelen ser una caja blanca. (Peño, 2015, p. 31)

b). Pruebas de caja negra (funcional, conductual, caja cerrada)

Por el contrario, las pruebas de caja negra (también llamadas funcionales, de comportamiento o de caja cerrada) describen cualquier prueba o método en el que se desconocen los detalles y el funcionamiento interno del software que se está probando.

Dado que no conoce ninguno de los detalles, realmente no puede crear casos de prueba que se dirijan a escenarios de nicho específicos o que hagan hincapié en la lógica específica del sistema. Por lo tanto, las pruebas de caja negra prueban principalmente el comportamiento de un sistema. Las pruebas de un extremo a otro suelen ser una caja negra. (Peño, 2015, p. 39).

c). Prueba de estrés (stress)

Consiste en probar los límites que un sistema puede soportar. En este tipo de pruebas se suele enviar más peticiones de las que el software podría atender normalmente para saber el comportamiento de la aplicación. (Chiu, 2018, p. 15).

1.6.1.7. Técnicas de investigación

i. Entrevista: Por medio de una interacción con el afectado o interesado.

ii. Observación: Por medio visual para captar todo lo importante.

iii. Cuestionario: Por medio de preguntas pueden ser, selección, abierta, cerrada.

1.7. HERRAMIENTAS

1.7.1. Backend

1.7.1.1. TypeScript

TypeScript es un lenguaje de programación de código abierto creado por Microsoft en el año 2012, que implementa muchos mecanismos de programación orientada a objetos. Se trata de un superconjunto de JavaScript, lo que significa que extiende la sintaxis de JavaScript.

Al compilarlo, se genera código JavaScript, ya que el navegador no puede interpretar TypeScript. Este proceso se llama transpilar, que significa generar código en un lenguaje específico a partir de otro. (Puciarelli, 2020, p. 9).

1.7.1.2. Framework nest.js

NestJS es un framework progresivo de Node.js para la creación de aplicaciones eficientes, confiables y escalables del lado del servidor, el cual está construido y es completamente compatible con TypeScript (no obstante aún nos permite la codificación en JS puro), combinando elementos de la programación orientada a objetos (POO en español; OOP, según sus siglas en inglés, Object-oriented programming), programación funcional (FP, según sus siglas en inglés, functional programming) y programación reactiva funcional (FRP, según sus siglas en inglés, functional reactive programming). (Nest.JS, 2022).

1.7.1.3. Typeorm

Typeorm es como cualquier otro ORM, permite interactuar con múltiples bases de datos y admite múltiples controladores de bases de datos para typeorm, admite MySQL, MariaDB, PostgreSQL, CockroachDB, SQLite, Microsoft SQL Server sql.js Oracle y MongoDB para bases de datos NoSQL, por lo que, según su caso, simplemente necesita instalar el controlador correspondiente de su base de datos y estará listo para comenzar. (TypeORM, 2022).

1.7.1.4. Postgresql

Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL,1 similar a la BSD o la MIT.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). (PostgreSQL, 2022).

1.7.1.5. Node.js

Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. (NodeJS, 2022).

1.7.2. Frontend

1.7.2.1. Aplicación móvil

a). Lenguaje Java

La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera bytecodes es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de applets que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto. (Fernandéz, 2005, p. 9).

b). Android studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ. (Developers, 2020).

c). Sqlite

SQLite es un sistema de gestión de base de datos relacional, contenida en una biblioteca muy pequeña, escrita en C. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. (SQLite, 2022).

d). Retrofit

Retrofit es un cliente HTTP de tipo seguro para Android y Java. Retrofit hace sencillo conectar a un servicio web REST traduciendo la API a interfaces Java. Esta poderosa librería hace sencillo consumir datos JSON o XML, que después es analizado en Objetos Java (Plain Old Java Objects, POJOs). Todas las peticiones GET, POST, PUT, PATCH, and DELETE pueden ser ejecutadas. (Retrofit, 2022).

1.7.2.2. Sistema Web

a). Html5

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente pero, incluso cuando algunas APIs (Interface de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. (Gauchat, 2012, p.1).

b). Css3

CSS es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc… (Gauchat, 2012, p.31).

c). Framework angular

Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (Angular, 2022).

d). Angular material

Como su nombre indica, es una librería de componentes web con un diseño Material design, una guía de estilo creada por Google para Android y para sus aplicaciones.

En los últimos tiempos se ha puesto muy de moda el estilo material design, y han salido un montón de librerías y frameworks que lo implementan, entre ellas Angular Material. (Angular Material, 2022).

1.8. LÍMITES Y ALCANCES

1.8.1. Límites

- ✓ La aplicación móvil para ventas solo estará disponible para dispositivos con tecnología Android.
- ✓ La aplicación móvil no permitirá pagos vía online ya sea tarjeta de crédito, débito, PayPal o cualquier otro medio.
- ✓ La aplicación móvil será compatible con sistemas operativos Android para versiones 4.4.2 y superiores.
- ✓ El sistema web solo estará orientado a ventas, proveedores, usuarios, del Micromarket Home Service.

1.8.2. Alcances

El presente proyecto Aplicación móvil y sistema web para ventas compras inventario mediante pedidos online de productos de consumo masivo, tendrá los siguientes módulos:

Aplicación Móvil:

- ✓ Módulo de pedidos de cliente
  - Módulo de agregar producto
  - Módulo de agregar producto deseado
  - Módulo de registro cliente
  - Módulo de productos en oferta

Sistema Web:

- ✓ Módulo de ventas
- ✓ Módulo usuario
  - Registro de usuario

- Actualización usuario
- Eliminación usuario

✓ Módulo de inventario y stock
- Módulo de registro de productos y compras
- Módulo de registro de proveedores
- Módulo de registro de categorías

✓ Módulo de pedidos de cliente
- Módulo de agregar producto
- Módulo de agregar producto deseado
- Módulo de registro cliente
- Módulo de productos en oferta

✓ Módulo reportes
- Reporte de ventas
- Reporte de productos
- Reporte de proveedores
- Reporte usuarios
- Reporte productos más vendidos

1.9. APORTES

El sistema web es de mucha utilidad para el administrador de la micromarket con lo que podrá realizar distintos procesos como agregar nuevos productos, editar, eliminar, agregar nuevas categorías, editar, eliminar, agregar nuevos proveedores, editar, eliminar, controlar las fechas de vencimiento, stock, gestionar los pedidos de productos, generar reportes de productos, ventas realizadas en una determinada fecha.

La aplicación móvil propuesto es útil para el cliente donde puede realizar una selección de categoría, ingresar y seleccionar los productos de su preferencia, agregar al carrito de compras, una vez que ya no tenga nada más que agregar enviar su pedido, como también si ocurre algún problema enviar un mensaje de su reclamo, y también revisar su historial de pedidos.

CAPÍTULO II

MARCO TEÓRICO

2.1. INTRODUCCIÓN

El capítulo en cuestión desglosa de manera más detallada la parte teórica que fundamentan las referencias empleadas en el documento en general. De forma implícita se divide en 3 secciones donde el primero consta de conceptos y definiciones relacionado al documento, la segunda parte trata sobre ingeniería de software, metodologías para desarrollo móvil Mobile-D y UWE para el lado admin, métricas según la ISO/IEC 25000, estándares y pruebas, por último, se tiene herramientas necesarias que serán utilizados en el desarrollo como PostGreSQL, Node.js, Nest.js, TypeORM, Angular, Angular Material, AndroidStudio, Retrofit.

2.2. SISTEMA

Un sistema es un objeto complejo cuyas partes o componentes se relacionan con al menos alguno de los demás componentes; ya sea conceptual o material. Todos los sistemas tienen composición, estructura y entorno, pero solo los sistemas materiales tienen mecanismos (o procesos), y solo algunos sistemas materiales tienen figura (forma). (Maldonado, 2021, p. 14).

Se entiende por un sistema a un conjunto ordenado de componentes relacionados entre sí, ya se trate de elementos materiales o conceptuales, dotado de una estructura, una composición y un entorno particulares. Se trata de un término que aplica a diversas áreas del saber, como la física, la biología y la informática o computación. (Jhonson, 2021, p. 1).

Un sistema es un conjunto de subsistemas o partes de componentes relacionados entre si para cumplir un fin común.

2.3. SISTEMA DE INFORMACIÓN

Un sistema de informaciones es un conjunto de componentes que interactúan entre sí con un fin común. En informática, los sistemas de información ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización. (Lindsay, 2000, p. 10).

La importancia de un sistema de información radica en la eficiencia en la correlación de una gran cantidad de datos ingresados a través de procesos diseñados para cada área con el objetivo de producir información válida para la posterior toma de decisiones. Un sistema de información se destaca por su diseño, facilidad de uso, flexibilidad, mantenimiento automático de los registros, apoyo en toma de decisiones críticas y mantener el anonimato en informaciones irrelevantes. (Sage, 1968, p. 63-69).

Un sistema de información es un conjunto de partes que se relacionan entre si para obtener un beneficio en común, importante para las empresas que quieren ahorrar costos y maximizar beneficios.

2.4. SISTEMA WEB

Es similar a un sitio web, pero con mucho más dinamismo y funcionalidades muy potentes que brindan respuestas a casos particulares. Las aplicaciones web son sistemas informáticos complejos, como los programas que antes teníamos en la computadora, pero para internet, es decir, que se codifican en lenguajes soportados por los navegadores web y se alojan en un servidor en Internet. Por ejemplo, un sistema para llevar la administración de una clínica, al que se accede mediante www o una red privada local, es una aplicación web. Las aplicaciones web siempre están en internet, pero pueden manejarse mediante intranets y extranets, depende la seguridad y privacidad requerida por el cliente. (Castillo Peña, , 2018, p. 24).

En la ingeniería de software se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. En otras palabras, es un programa que se codifica en un lenguaje interpretable por los navegadores web en la que se confía la ejecución al navegador. (Sotomayor, 2017, p. 20).

Un sistema web es mucho más complejo que un sitio web, donde tiene funcionalidades particulares para realizar procesos como de cálculo de datos, visualización de información con sierto grado de dinamismo.

2.5. APLICACIÓN

En informática, el software de aplicación es un tipo de software de computadora diseñado para realizar un grupo de funciones, tareas o actividades coordinadas para el beneficio del usuario. Ejemplos de una aplicación en ocasiones se usa el acortamiento inglés app, de application serían un procesador de textos, una hoja de cálculo, una aplicación de contabilidad, un navegador web, un reproductor multimedia, un simulador de vuelo aeronáutico, una consola de juegos o un editor de fotografías. (Ramíez Vique, 2019, p. 13).

Una app es un programa que se instala en un dispositivo móvil ya sea teléfono o tableta y que se puede integrar a las características del gadget, como su cámara o sistema de posicionamiento global (GPS). Además, se puede actualizar para añadirle nuevas características con el paso del tiempo. (BBC News Mundo, 2011, p. 1).

Una aplicación es un programa para computadora como para teléfono celular, donde tiene la facultad de procesar alguna acción como cálculo de datos.

2.6. APLICACIÓN MÓVIL

En el pasado se hablado de las aplicaciones móviles y, a pesar de que los móviles ya tenían una gran penetración en el mercado y de que su uso como herramienta de trabajo o elemento de la vida diaria era bastante común, las aplicaciones móviles no habían acabado de despegar. (Ramírez, 2012, p.40).

Actualmente, más del 70% de la población dispone de dispositivos móviles. El número de Smartphone no para de crecer (el 90% de los nuevos dispositivos son Smartphone, según los estudios de Gartner), Es, sin lugar a dudas, el sector que mayor innovación y expectación está generando y generará. (Ramírez, 2012, p. 40).

2.6.1. Tipos de aplicaciones móviles

2.6.1.1. Aplicaciones Web o Web App

Son aplicaciones muy usadas para brindar accesibilidad a la información desde cualquier dispositivo, sin importar el sistema operativo, ya que solo se necesita contar con un navegador para acceder.

Ventajas:

- ✓ Pueden ser utilizadas desde cualquier dispositivo.
- ✓ Costo de creación mínimo.
- ✓ No requieren de ninguna aprobación para su publicación.
- ✓ Mayor número de plataformas soportadas.

Desventajas:

- ✓ No utilizan los recursos del dispositivo de manera óptima.
- ✓ El rendimiento es menor que en las aplicaciones nativas.
- ✓ Capacidad de funcionamiento offline limitada.
- ✓ Existe la posibilidad de enviar Notificaciones push.

2.6.1.2. Aplicaciones Nativas

Las aplicaciones nativas son aquellas desarrolladas bajo un lenguaje y entorno de desarrollo específico, lo cual permite, que su funcionamiento sea muy fluido y estable para el sistema operativo que fue creada. (García Mendoza, 2015, p. 6).

Ventajas:

- ✓ Pueden ser publicadas en tiendas para su distribución.
- ✓ No necesitan siempre de conexión a internet para su funcionamiento.
- ✓ Mejor experiencia para el usuario.
- ✓ Costos económicos al utilizar creadores online.

Desventajas:

- ✓ Sólo pueden ser utilizadas con el sistema para el que han sido creadas.
- ✓ Costos y tiempos de desarrollo altos (sin el uso de creadores online).
- ✓ Necesitan aprobación de las tiendas para ser publicadas.

2.6.1.3. Aplicaciones Híbridas

Este tipo de aplicaciones se crean utilizando lenguajes de desarrollo web y un framework dedicado para la creación de aplicaciones híbridas. La facilidad que brinda este tipo de desarrollo es que no hay un entorno específico el cual hay que utilizar para su desarrollo y al igual que las aplicaciones HTML5. (García Mendoza, 2015, p. 6).

Ventajas:

- ✓ Son multiplataforma.
- ✓ Distribución en las tiendas de Apps.
- ✓ El coste de desarrollo es menor que el de una aplicación nativa.

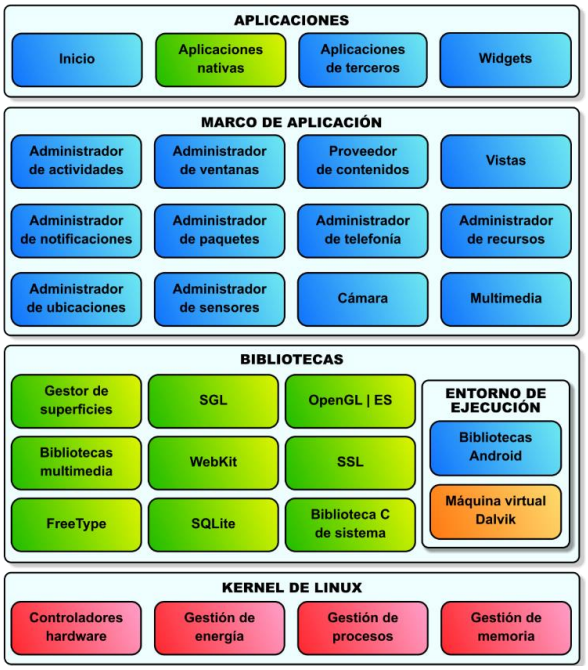
Desventajas:

- ✓ No cuenta con todas las funcionalidades nativas. (García Mendoza, 2015, p. 6).

2.6.2. Arquitectura Android

Figura 2.2

Arquitectura Android



Nota. Niveles de arquitectura Android. Obtenido: (Biaje, 2015, p. 20).

2.6.2.1. El núcleo linux

El núcleo de Android está formado por el sistema operativo Linux versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. (Biaje, 2015, p. 20).

2.6.2.2. Runtime de Android

Dadas las limitaciones de los dispositivos donde ha de correr Android (poca memoria y procesador limitado), no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones. A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%. (Biaje, 2015, p. 20).

2.6.2.3. Librerías nativas

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

2.6.2.4. Entorno de aplicación

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.).

Los servicios más importantes que incluye son:

- ✓ **Views:** extenso conjunto de vistas, (parte visual de los componentes).
- ✓ **Resource Manager:** proporciona acceso a recursos que no son en código.
- ✓ **Activity Manager:** maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- ✓ **Notification Manager:** permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.

- ✓ **Content Providers:** mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos). (Biaje, 2015, p. 20).

2.6.2.5. Aplicaciones

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema. (Biaje, 2015, p. 20).

2.7. PEDIDOS ONLINE

En materia de comercio electrónico un pedido online es la formalización de la intención de un consumidor (el comprador online) de comprar un producto o contratar un servicio, a un vendedor (la tienda online), en los términos y condiciones convenidos con éste. (Pérez Muzha , 2022, p. 43).

2.8. PRODUCTOS DE CONSUMO MASIVO

Son aquellos productos requeridos por todos los estratos de la sociedad, están para que sean consumidos en corto tiempo, son accesibles en todas partes lo que motiva a la competencia entre las empresas de este sector por la captación de clientela, intentando diferenciarse, ofreciendo alternativas, precios o agregados.

Entre las características principales de estos productos podríamos destacar las siguientes:

- ✓ **Consumo inmediato**: estos productos no suelen durar mucho tiempo en el hogar, están fabricados con la idea de que sean consumidos en un periodo corto de tiempo.
- ✓ **Compra cotidiana**: al ser productos de primera necesidad que se consumen rápidamente se adquieren por los consumidores de forma cotidiana.
- ✓ **Fáciles de encontrar**: puedes encontrar los productos de consumo masivo en distintos sitios sin mayor dificultad puesto que existen un sin fin de empresas dedicadas a este sector.

- ✓ **Precio reducido**: la demanda de estos productos varía en función de los precios. Como hemos comentado, todos los estratos de la sociedad los consumen de modo que tienen un precio muy asequible. (Alvarez Tanaka, 2009, p. 42).

2.9. VENTA ONLINE

Consiste en ofrecer productos, servicios, ideas u otros mediante un sitio web en internet, de tal forma, que los posibles compradores puedan conocer en qué consisten y cuáles son sus ventajas y beneficios a través de ese sitio web.

Y luego, otra empresa, organización o persona utiliza una computadora conectada a internet para comprar esos productos o servicios, se puede decir que las partes han intervenido en una transacción electrónica o venta online. (Thommson, 2019, p.23).

Los vendedores de productos y servicios ponen a disposición de sus clientes una página web (o aplicación informática) en la que se pueden observar imágenes de los productos, leer sus especificaciones y finalmente, adquirirlos. Este servicio le da al cliente rapidez en la compra, la posibilidad de hacerlo desde cualquier lugar del mundo y a cualquier hora del día. Algunas tiendas en línea incluyen dentro de la propia página del producto los manuales de usuario de manera que el cliente puede darse una idea de antemano de lo que está adquiriendo. (Sandhusen L., 2002, p. 637-638).

Los vendores ponen productos y servicios en una aplicación móvil o web para que los clientes puedan comprar facilmente desde cualquier punto.

2.10. COMPRA ONLINE

En materia de comercio electrónico, una compra online es la acción voluntaria de adquirir un bien o contratar un servicio a distancia, por medio de internet, a cambio de un precio. Para que haya una compra online, como en cualquier contrato, debe haber comprador online, vendedor (prestador de servicios de la sociedad de la información), cosa, precio y consentimiento. (Cordova Lopez, 2020, p. 7).

2.11. INVENTARIO

El inventario es una relación detallada, ordenada y valorada de los elementos que componen el patrimonio de una empresa o persona en un momento determinado. Antiguamente lo normal era que los inventarios se realizaran por medio físico (se escribían en un papel), pero ahora se suelen mantener en bases de datos de manera centralizada a toda una empresa, aunque haya empresas o tiendas pequeñas que lo sigan haciendo con papel. (Durán, 2012, p. 57).

El inventario es:

- ✓ Detallado porque se especifican las características de cada uno de los elementos que integran el patrimonio.
- ✓ Ordenado porque agrupa los elementos patrimoniales en sus cuentas correspondientes y las cuentas en sus masas patrimoniales.
- ✓ Valorado porque se expresa el valor de cada elemento patrimonial en unidades monetarias. La variación de números que encontramos en un inventario por ejemplo el reencuentro de datos de la empresa. (Durán, 2012, p. 57).

Los inventarios de una compañía están constituidos por sus materias primas, sus productos en proceso, los suministros que utiliza en sus operaciones y los productos terminados. Un inventario puede ser algo tan elemental como una botella de limpiador de vidrios empleada como parte del programa de mantenimiento de un edificio, o algo más complejo, como una combinación de materias primas y subensamblajes que forman parte de un proceso de manufactura. (Mangas Roca, 2018, p. 3).

Los inventarios de una organización son de primordial importancia, antes eran guardados en medios físicos pero en al actualidad con la ayuda y avance de la tecnología son almacenados en base de datos.

2.11.1. Método PEPS de inventarios

Método FIFO o PEPS. Este método se basa en que lo primero que entra es lo primero en salir. Su apreciación se adapta más a la realidad del mercado, ya que emplea una valoración basada en costos más recientes. (Durán, 2012, p. 57).

2.12. COMERCIO ELETRÓNICO

El comercio electrónico también conocido como e-commerce (electronic commerce en inglés), comercio por Internet o comercio en línea consiste en la compra y venta de productos o de servicios a través de internet, tales como redes sociales y otras páginas web. También puede ser definido como una actividad económica que permite el comercio de diferentes productos y servicios a partir de medios digitales, como páginas web, aplicaciones móviles y redes sociales. Por medio de la red virtual, los clientes pueden acceder a diversos catálogos de marcas, servicios y productos, en todo momento y en cualquier lugar. (Malca, 2021, p. 33).

Originalmente, el término se aplicaba a la realización de transacciones mediante medios electrónicos tales como el intercambio electrónico de datos; sin embargo, con el advenimiento del Internet y del World Wide Web, a mediados de la década de 1990 comenzó a referirse principalmente a la venta de bienes y servicios a través de Internet, usando como forma de pago medios electrónicos tales como las tarjetas de crédito y nuevas metodologías, como el pago móvil o las plataformas de pago. Vender y comprar ya es una tarea bastante sencilla propiciada, desde luego, por la tecnología, como los dispositivos móviles con acceso a la red. (Murillo, 2009, p. 151).

El comercio electrónico es la compra y venta de productos, servicios mediante internet donde hay un acuerdo mutuo para realizar un transaccion monetaria.

2.13. INGENIERÍA DE SOFTWARE

La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación. En esta definición se presentan dos fases clave: (Sommerville, 2011, p. 13).

1. Disciplina de ingeniería. Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde es adecuado. Sin embargo, los usan de manera selectiva y siempre tratan de encontrar soluciones a problemas, incluso cuando no hay teorías ni métodos aplicables.

Los ingenieros también reconocen que deben trabajar ante restricciones organizacionales y financieras, de modo que buscan soluciones dentro de tales limitaciones.

2.Todos los aspectos de la producción del software. La ingeniería de software no solo se interesa por los procesos técnicos del desarrollo de software, sino también incluye actividades como la administración del proyecto de software y el desarrollo de herramientas, así como métodos y teorías para apoyar la producción de software.

La ingeniería busca obtener resultados de la calidad requerida dentro de la fecha y del presupuesto. A menudo esto requiere contraer compromisos: los ingenieros no deben ser perfeccionistas. Sin embargo, las personas que diseñan programas para sí mismas podrían pasar tanto tiempo como deseen en el desarrollo del programa.

En general, los ingenieros de software adoptan en su trabajo un enfoque sistemático y organizado, pues usualmente esta es la forma más efectiva de producir software de alta calidad. No obstante, la ingeniería busca seleccionar el método más adecuado para un conjunto de circunstancias y, de esta manera, un acercamiento al desarrollo más creativo y menos formal seria efectivo en ciertas situaciones. El desarrollo menos formal es particularmente adecuado para la creación de sistemas basados en la web, que requieren una mezcla de habilidades de software y diseño gráfico. La ingeniería de software es importante por dos razones: (Sommerville, 2011, p. 13).

1. Cada vez con mayor frecuencia, los individuos y la sociedad se apoyan en los avanzados sistemas de software. Por ende, se requiere producir económica y rápidamente sistemas confiables.

2. A menudo resulta más barato a largo plazo usar métodos y técnicas de ingeniería de software para los sistemas de software, que solo diseñar los programas como si fuera un proyecto de programación personal. (Sommerville, 2011, p. 13).

Según los escritores destacados (Pradel & Raya, 2009, p. 13). El desarrollo es el proceso que lleva a la producción o creación del producto de software; la operación consiste en ejecutar el producto de software dentro de su entorno de ejecución para llevar a cabo su función.

Finalmente, el mantenimiento comprende la modificación posterior del producto de software para corregir los errores o adaptarlo a nuevas necesidades.

Por lo tanto, cuando hablamos de ingeniería de software no solamente estamos hablando de una manera de desarrollar software, sino que también debemos tener en cuenta la vida posterior del producto una vez creado. La ingeniería de software consiste en llevar a cabo todas estas actividades de manera que podamos medir, cuantificar y analizar los diferentes procesos relacionados con la vida del producto de software. (Pradel & Raya, 2009, p. 13).

2.13.1. Ingeniería de requerimientos

2.13.1.1. Requerimiento funcional

Los requerimientos funcionales para un sistema refieren lo que el sistema debe hacer. Tales requerimientos dependen del tipo de software que se esté desarrollando, de los usuarios esperados del software y del enfoque general que adopta la organización cuando se escriben los requerimientos funcionales se describen por lo general de forma abstracta que entiendan los usuarios del sistema. Sin embargo, requerimientos funcionales más específicos del sistema detallan las funciones del sistema, sus entradas y salidas, sus excepciones, etcétera. (Sommerville, 2011, p. 109).

Los requerimientos funcionales del sistema varían desde requerimientos generales que cubren lo que tiene que hacer el sistema, hasta requerimientos muy específicos que reflejan maneras locales de trabajar o los sistemas existentes de una organización (Sommerville, 2011, p. 109).

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requisitos de comportamiento para cada requisito funcional se muestran en los casos de uso.

Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

Como se define en la ingeniería de requisitos, los requisitos funcionales establecen los comportamientos del software. Típicamente, un analista de requisitos genera requisitos funcionales después de realizar los casos de uso. Sin embargo, esto puede tener excepciones, ya que el desarrollo de software es un proceso iterativo y algunos requisitos son previos al diseño de los casos de uso. Ambos elementos (casos de uso y requisitos) se complementan en un proceso bidireccional. (Ruiz, 2012, p. 15).

Un requisito funcional típico contiene un nombre, un número de serie único y un resumen. Esta información se utiliza para ayudar al lector a entender por qué el requisito es necesario, y para seguir al mismo durante el desarrollo del producto. El núcleo del requisito es la descripción del comportamiento requerido, que debe ser clara y concisa. (Ruiz, 2012, p. 10).

2.13.1.2. Requerimiento no funcional

Los requerimientos no funcionales, como indica su nombre, son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento.

De forma alternativa, pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos I/O, o la representación de los datos usados en las interfaces con otros sistemas (Sommerville, 2011, p. 109).

Los requerimientos no funcionales a menudo son más significativos que los requerimientos funcionales individuales. Es común que los usuarios del sistema encuentren formas para trabajar en torno a una función del sistema que realmente no cubre sus necesidades. No obstante, el fracaso para cubrir los requerimientos no funcionales haría que todo el sistema fuera inútil. (Sommerville, 2011, p. 109).

Por lo general es más difícil relacionar componentes con requerimientos no funcionales. La implementación de dichos requerimientos puede propagarse a lo largo del sistema. Para esto existen dos razones:

1. Los requerimientos no funcionales afectan más la arquitectura global de un sistema que los componentes individuales. Por ejemplo, para garantizar que se cumplan los requerimientos de rendimiento, quizá se deba organizar el sistema para minimizar las comunicaciones entre componentes.

2. Un requerimiento no funcional individual, como un requerimiento de seguridad, podría generar algunos requerimientos funcionales relacionados que definan nuevos servicios del sistema que se requieran. Además, también podría generar requerimientos que restrinjan los requerimientos ya existentes.

Los requerimientos no funcionales surgen a través de necesidades del usuario, debido a restricciones presupuestales, políticas de la organización, necesidad de interoperabilidad con otros software o sistemas de hardware, o factores externos como regulaciones de seguridad o legislación sobre privacidad. (Sommerville, 2011, p. 109).

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software. Un requisito que sabe bien y especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales.

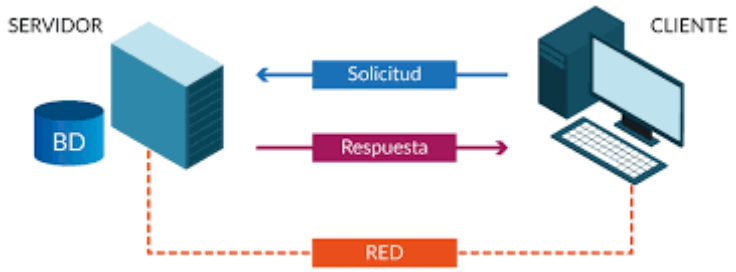
Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento, por eso suelen denominarse Atributos de calidad de un sistema. Queda entonces el requisito no funcional, que son las restricciones o condiciones que impone el cliente al programa que necesita, por ejemplo, el tiempo de entrega del programa, el lenguaje o la cantidad de usuarios. (Ruiz, 2012, p. 15).

2.13.2. Cliente servidor

La arquitectura cliente servidor tiene dos partes claramente diferenciadas, por un lado, la parte del servidor y por otro la parte de cliente o grupo de clientes donde lo habitual es que un servidor sea una máquina bastante potente con un hardware y software específico que actúa de depósito de datos y funcione como un sistema gestor de base de datos o aplicaciones. (Schiaffarino, 2019, p. 25).

Figura 2.2

Cliente Servidor



Nota. Muestra la interrelación cliente servidor. Obtenido: (Schiaffarino, 2019, p. 25).

En esta arquitectura el cliente suele ser estaciones de trabajo que solicitan varios servicios al servidor, mientras que un servidor es una máquina que actúa como depósito de datos y funciona como un sistema gestor de base de datos, este se encarga de dar la respuesta demandada por el cliente. (Schiaffarino, 2019, p. 25).

- ✓ Red: Una red es un conjunto de clientes, servidores y base de datos unidos de una manera física o no física en el que existen protocolos de transmisión de información establecidos.
- ✓ Cliente: El concepto de cliente hace referencia a un demandante de servicios, este cliente puede ser un ordenador como también una aplicación de informática, la cual requiere información proveniente de la red para funcionar.
- ✓ Servidor: Un servidor hace referencia a un proveedor de servicios, este servidor a su vez puede ser un ordenador o una aplicación informática la cual envía información a los demás agentes de la red.
- ✓ Protocolo: Un protocolo es un conjunto de normas o reglas y pasos establecidos de manera clara y concreta sobre el flujo de información en una red estructurada.
- ✓ Servicios: Un servicio es un conjunto de información que busca responder las necesidades de un cliente, donde esta información pueden ser mail, música, mensajes simples entre software, videos, etc.

- ✓ Base de datos: Son bancos de información ordenada, categorizada y clasificada que forman parte de la red, que son sitios de almacenaje para la utilización de los servidores y también directamente de los clientes. (Schiaffarino, 2019, p. 25).

2.13.3. Web services

Un web service es una vía de intercomunicación e interoperabilidad entre máquinas conectadas en Red. En el mundo de Internet se han popularizado enormemente, ya se trate de web services públicos o privados. Generalmente, la interacción se basa en el envío de solicitudes y respuestas entre un cliente y un servidor, que incluyen datos. Por tanto, podemos entender un servicio web como un tráfico de mensajes entre dos máquinas. (Morales Machuca, 2010, p. 1).

REST usa el propio protocolo HTTP para la comunicación entre máquinas. HTTP es ampliamente soportado por todos los sistemas y de hecho para la transferencia de datos en la web se usa HTTP. REST se caracteriza por no tener estado. Es decir, el servidor no es capaz de recordar el estado de la anterior solicitud REST que pudo, o no, hacer un cliente. Por ello, el cliente tiene que enviar en cada solicitud todo el estado de su sesión, lo que se suele hacer mediante un token que le «ayude a recordar» al servidor.

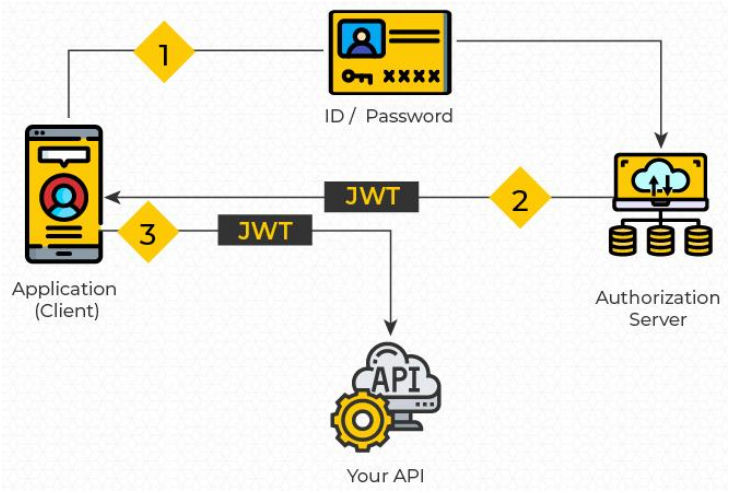
REST tiene a simplificar las cosas y en esa misma línea se suele usar un lenguaje diferente para representación de los datos, el lenguaje JSON. Hoy REST y JSON se han convertido en la opción más sencilla y por tanto más recomendable para implementar un servicio web. (Morales Machuca, 2010, p. 1).

2.13.4. Json web token

Json Web Token es un conjunto de medios de seguridad para peticiones http y así representar demandas para ser transferidos entre dos partes (cliente y servidor). Las partes de un JWT se codifican como un objeto JSON que está firmado digitalmente utilizando JSON Web Signature(JWS). (Gunawan & Rahmatulloh, 2019, p. 74).

Figura 2.3

Json Web Token



Nota. Agrega seguridad al login. Obtenido: (Gunawan & Rahmatulloh, 2019, p. 74).

2.13.5. Diseño

El diseño de software agrupa el conjunto de principios, conceptos y prácticas que llevan al desarrollo de un sistema o producto de alta calidad. (Pressman, 2010, p. 183).

Deben entenderse los conceptos de diseño antes de aplicar la mecánica de este, y la práctica del diseño en si lleva a la creación de distintas representaciones del software que sirve como guía para la actividad de construcción que siga. (Pressman, 2010, p. 183).

Según la plataforma de repositorio (Github, 2019) Es el proceso creativo de transformación del problema en una solución. Una vez que se analizan y especifican los requisitos, el diseño es la siguiente actividad técnica a realizar. Es independiente del modelo de procesos que se use. El diseño se centra en 4 áreas importantes:

- ✓ Datos
- ✓ Arquitecturas
- ✓ Interfaces
- ✓ Componentes

2.13.5.1. Tipos de Diseño

a) Diseño de Datos

Igual que otras actividades de la ingeniería de software, el diseño de datos (en ocasiones denominada arquitectura de datos) crea un modelo de datos o información que se representa en un nivel de abstracción elevado (el punto de vista del usuario de los datos). Este modelo de los datos se refina después en forma progresiva hacia representaciones más específicas de la implementación que puedan ser procesadas por el sistema basado en computadora.

En muchas aplicaciones de software, la arquitectura de los datos tendrá una influencia profunda en la arquitectura del software que debe procesarlo. (Pressman, 2010, p. 199).

La estructura de los datos siempre ha sido parte importante del diseño de software. En el nivel de componentes del programa, del diseño de las estructuras de datos y de los algoritmos requeridos para manipularlos, es esencial la creación de aplicaciones de alta calidad. En el nivel de la aplicación, la traducción de un modelo de datos (obtenido como parte de la ingeniería de los requerimientos) a una base de datos es crucial para lograr los objetivos de negocios de un sistema. En el nivel de negocios, el conjunto de información almacenada en bases de datos incompatibles y reorganizados en un “data warehouse” permite la minería de datos o descubrimiento de conocimiento que tiene un efecto en el éxito del negocio en sí. En cada caso, el diseño de los datos juega un papel importante. (Pressman, 2010, p. 199).

Según la plataforma (Github, 2019). Transforma el modelo del dominio obtenido del análisis, en estructuras de datos, objetos, relaciones, etc. Por ejemplo, un diagrama de entidad relación.

b) Diseño arquitectónico

El diseño de la arquitectura del software es el equivalente del plano de una casa. Este ilustra la distribución general de las habitaciones, su tamaño, forma y relaciones entre ellas, así como las puertas y ventanas que permiten el movimiento.

El plano da una visión general de la casa. Los elementos del diseño de la arquitectura dan la visión general del software. (Pressman, 2010, p. 199).

El modelo arquitectónico proviene de tres fuentes: 1) información sobre el dominio de la aplicación del software que se va a elaborar, 2) los elementos específicos del modelo de requerimientos, tales como diagramas de flujo de datos o clases de análisis, sus relaciones y colaboraciones para el problema en cuestión y 3) la disponibilidad de estilos arquitectónicos y sus patrones. (Pressman, 2010, p. 199).

Según la plataforma de repositorio (Github, 2019). Define la relación entre los componentes más importantes del software para lograr los requisitos del sistema.

La información puede derivarse de la especificación, del modelo de análisis y de la interacción de los subsistemas definidos.

c) Diseño de Interfaz

El diseño de la interfaz para el software es análogo al conjunto de trazos (y especificaciones) detalladas para las puertas, ventanas e instalaciones de una casa. Tales dibujos ilustran el tamaño y forma de puertas y ventanas, la manera en la que operan, la forma en la que llegan las instalaciones de servicios (agua, electricidad, gas teléfono, etc.) a la vivienda y se distribuyen entre las habitaciones indicadas en el plano. Indican donde está el timbre de la puerta, si se usara un intercomunicador para anunciar la presencia de un visitante y como se va a instalar el sistema de seguridad. (Pressman, 2010, p. 199).

En esencia, los planos (y especificaciones) detallados para las puertas, ventanas e instalaciones externas nos dicen como fluyen las cosas y la información hacia dentro y fuera de la casa y dentro de los cuartos que forman parte del plano. Los elementos de diseño de la interfaz de software permiten que la información fluya hacia adentro y afuera del sistema, y como están comunicados los componentes que son parte de la arquitectura.

Hay tres elementos importantes del diseño de la interfaz: 1) la interfaz de usuario (IU), 2) las interfaces externas que tienen que ver con otros sistemas, dispositivos, redes y otros productores o consumidores de información.

3) interfaces internas que involucran a los distintos componentes del diseño. Estos elementos del diseño de la interfaz permiten que el software se comunique externamente y permite la comunicación y colaboración internas entre los componentes que constituyen la arquitectura del software. (Pressman, 2010, p. 199).

Según la plataforma de repositorio (Github, 2019) Describe la forma de comunicación dentro del mismo sistema, con otros sistemas y con las personas. Una interface implica flujo de información (datos o control) y comportamiento.

d) Diseño a nivel de componentes

El diseño de componentes para el software describe por completo los detalles internos de cada componente. Para lograrlo, este diseño define estructuras de datos para todos los objetos de datos locales y detalles algorítmicos para todo el procesamiento que tiene lugar dentro de un componente, así como la interfaz que permite el acceso a todas las operaciones de los componentes(comportamientos). (Pressman, 2010, p. 201).

Según la plataforma de repositorio (Github, 2019) Transforma los elementos estructurales de la arquitectura en una descripción procedimental de los componentes del software. La información obtenida del diseño de datos, sirven como base.

2.13.5.2. Maquetación

La diagramación, también llamada a veces maquetación, es un oficio del diseño editorial que se encarga de organizar en un espacio contenidos escritos, visuales y, en algunos casos, audiovisuales (multimedia) en medios impresos y electrónicos, como libros, diarios y revistas. (Romero Parra, 2017, p. 27).

Estrictamente, el acto de maquetar tan solo se relaciona con la distribución de los elementos en un espacio determinado de la página.

Mientras que el diseño editorial incluye fases más amplias del proceso, desde el proyecto gráfico, hasta los procesos de producción denominados pre prensa (preparación para impresión), prensa (impresión) y pos prensa (acabados).

Sin embargo, usualmente todo el aspecto gráfico de la actividad editorial y periodística se conoce por el término maquetación. (Romero Parra, 2017, p. 27).

Según el autor del blog (Heredia, 2017, p. 4). En la manufactura y diseño, un mockup, mock-up, o maqueta es un modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Un mockup es un prototipo si proporciona al menos una parte de la funcionalidad de un sistema y permite pruebas del diseño. Los mockups son utilizados por los diseñadores principalmente para la adquisición de comentarios por parte de los usuarios. Los mock-ups abordan la idea capturada en la ingeniería popular.

Usted puede arreglarlo ahora en el dibujo con una goma de borrar o más tarde en la obra con un martillo.

2.13.6. Ingeniería Web

Es el proceso utilizado para crear, implantar y mantener aplicaciones y sistemas Web de alta calidad. Esta breve definición nos lleva a abordar un aspecto clave de cualquier proyecto como es determinar qué tipo de proceso es más adecuado en función de las características del mismo. (Huanca, 2015, p. 30).

El desarrollo de aplicaciones Web posee determinadas características que lo hacen diferente del desarrollo de aplicaciones o software tradicional y sistemas de información. La ingeniería de la Web es multidisciplinaria. (Huanca, 2015, p. 30).

Según los autores (Bolaños, Urrea, & Gomez, 2015). La ingeniería web es un área que abarca procesos, técnicas y modelos orientados a los entornos Web. Consiste en la aplicación de metodologías sistemáticas, disciplinadas y cuantificables al desarrollo eficiente, operación y evolución de aplicaciones web de alta calidad.

La ingeniería Web toma prestado muchos de los conceptos y principios básicos de la ingeniería del software, dando importancia a las mismas actividades técnicas y de gestión. Existen diferencias sutiles en la forma en que se llevan a cabo estas actividades, pero la filosofía primordial es idéntica dado que dicta un enfoque disciplinado para el desarrollo de un Sistema basado en computadora. (Bolaños, Urrea, & Gomez, 2015).

2.13.6.1. Metodología UWE

La propuesta de ingeniería web basada en UML (UWE (Koch, 2000)) es una metodología detallada para el proceso de autoría de aplicaciones con una definición exhaustiva del proceso de diseño que debe ser utilizado. Este proceso, iterativo e incremental, incluye flujos de trabajo y puntos de control, y sus fases coinciden con las propuestas en el Proceso Unificado de Modelado. (Huanca, 2015, p. 32).

UWE está especializada en la especificación de aplicaciones adaptativas, y por tanto hace especial hincapié en características de personalización.

Como es la definición de un modelo de usuario o una etapa de definición de características adaptativas de la navegación en función de las preferencias, conocimientos o tareas de usuario.

Otras características relevantes del proceso y método de autoría de UWE son el uso del paradigma orientado a objetos, su orientación al usuario, la definición de una meta modelo (modelo de referencia) que da soporte al método y el grado de formalismo que alcanza debido al soporte que proporciona para la definición de restricciones sobre los modelos. (Huanca, 2015, p. 32).

a) Fases de la Metodología UWE

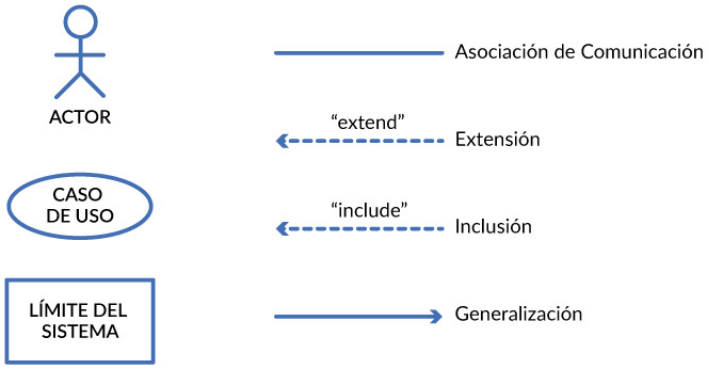
Las fases de la metodología UWE, son procesos o actividades que se utilizan y permiten identificar las necesidades de la aplicación o sistema web a desarrollar; estas actividades se describen y representan en cuatro fases que son:

i. Análisis de requisitos

Como en otras metodologías, la primera fase o actividad es la del análisis de requisitos funcionales, que permite visualizar los procesos y funciones que debe cumplir el sistema web, esta fase se ve reflejada en los casos de uso. (Huanca, 2015, p. 32).

Figura 2.4

Análisis de caso de uso



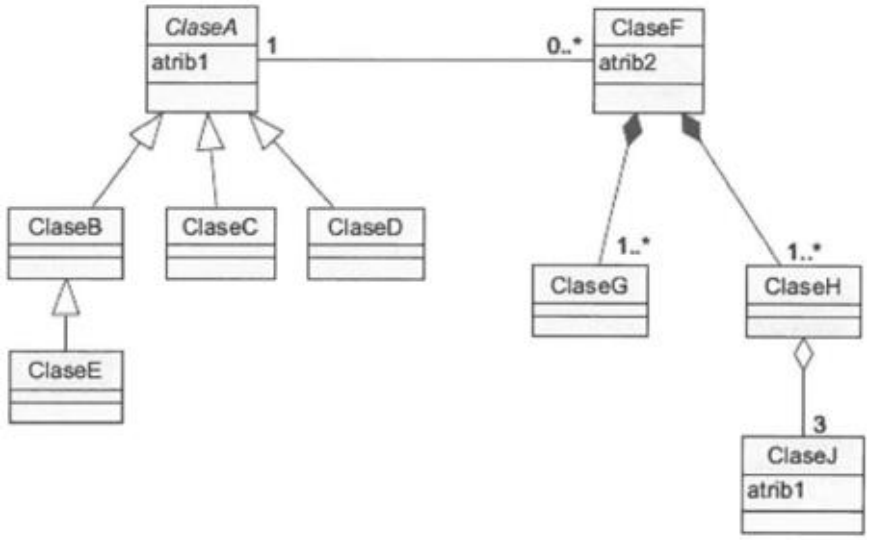
Nota. Caso de uso. Obtenido: (M. Gonzales, 2013, p. 32).

ii. Diseño conceptual

El diseño conceptual se basa en el análisis de requisitos del paso anterior. Esto incluye los objetos involucrados entre los usuarios y la aplicación. Este modelo propone construir un modelo de clases con estos objetos, ignorados los aspectos de navegación: Presentación e Interacción, que serán tratados posteriormente. Los principales elementos de modelado son; las clases, asociaciones y paquetes. (Huanca, 2015, p. 33).

Figura 2.5

Diseño conceptual



Nota. Caso de uso. Obtenido: (M. Gonzales, 2013, p. 32).

iii. Diseño navegacional

El diseño navegacional no es solo útil para la generación de la documentación de la estructura de la aplicación. Sino que también permite mejorar la estructura de navegabilidad. (Huanca, 2015, p. 34).

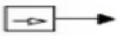
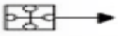
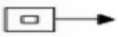
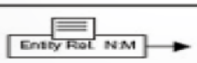
El modelo de la navegación comprende de:

- ✓ El modelo de espacio de navegación que especifica que objetos pueden ser visitados a través de la aplicación Web.
- ✓ El modelo de estructura de navegación que define como se alcanzan estos objetos a través de la web.
- ✓ En el proceso de construir el modelo espacial de navegación las decisiones del diseñador están basadas en el modelo conceptual y los requisitos de la aplicación definidos en el modo de caso de uso.

Cuando hablamos de un sistema web, es necesario conocer la relación y los enlaces entre las páginas web. Los elementos que se utilizan para el diseño de diagramas son:

Tabla 2.1

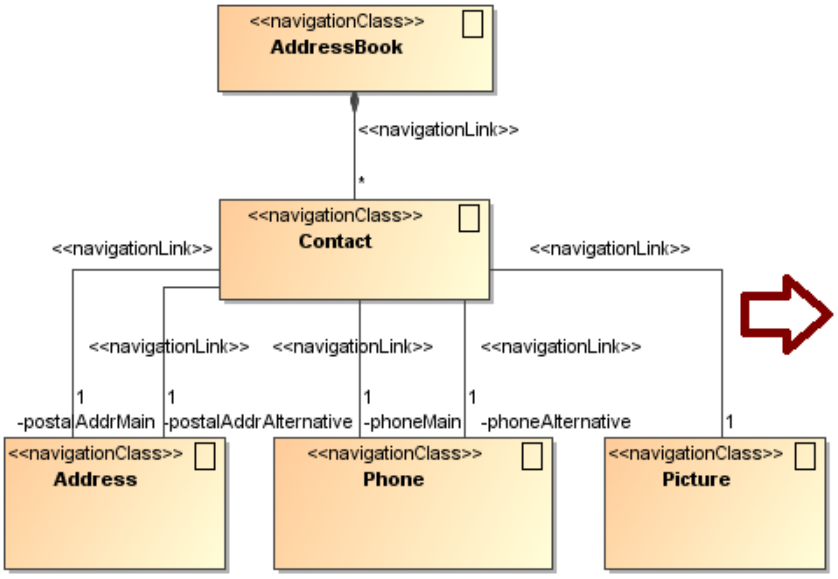
Elementos de diseño navegacional

ICONO	DESCRIPCION
	Hiperlance jerarquico inferido
	Hiperlance jerarquico inferido
	Indice
	Visita guiada
	Visita guiada indexada
	Acceso al calorio
	Acceso simultaneo
	Indice Multiple
	Visita guiada multiple
	Visita guiada indexada multiple
	Ejemplo de acceso a partir de una relacion N:M, en este caso utilizando un indice (podria ser cualquier otra primitiva de acceso simple)

Nota. Son elementos de diseño navegacional. Obtenido: (Ingeniería web basada en UML, Instituto de Informática).

Figura 2.6

Diseño navegacional UWE



Nota. Se muestra en la figura el diseño de navegación. Obtenido: (Ingeniería web basada en UML, Instituto de Informática).

iv. Diseño de presentación

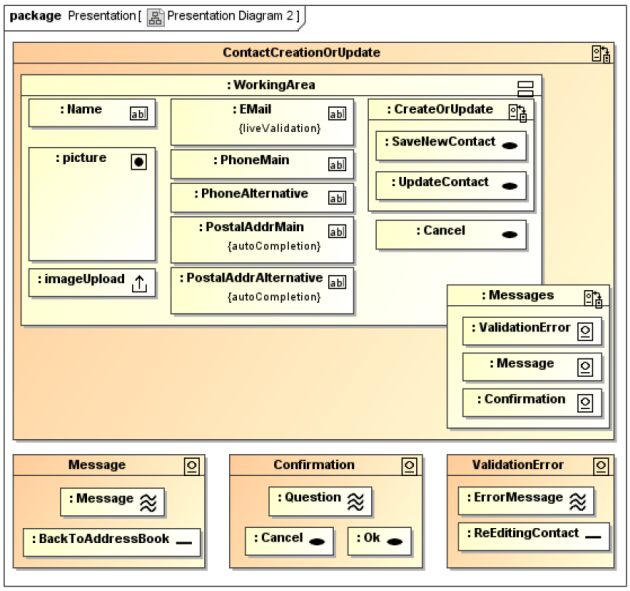
Este modelo permite una visión amplia de los procesos de la página web.

Que se representan en los diagramas de navegación; pueden interpretarse también con las interfaces del sistema web. Para el caso se tiene estereotipos o iconos que ayudan al diseño de los diagramas de presentación.

El diagrama de presentación de la metodología UWE, permite al usuario comprender y analizar, sobre el área de trabajo al que se someterá con la implantación del sistema. En la siguiente figura, se muestra la aplicación de los iconos que pertenecen a los diagramas de presentación. (Huanca, 2015, p. 32).

Figura 2.7

Diagrama de presentación



Nota. Se muestra el diagrama de presentación. Obtenido: (Ingeniería web basada en UML, Instituto de Informática).

v. Codificación del software

Durante esta etapa se realizan las tareas que se conocen como programación; que consiste, esencialmente, en llevar a código fuente, en el lenguaje de programación elegido, todo lo diseñado en la fase anterior. (Huanca, 2015, p. 32).

vi. Pruebas

Las pruebas se utilizan para asegurar el correcto funcionamiento de secciones de código.

vii. La Instalación o Fase de Implementación

Es el proceso por el cual los programas desarrollados son transferidos apropiadamente al computador destino, inicializados, y, eventualmente, configurados; todo ello con el propósito de ser ya utilizados por el usuario final.

Implementación y Lanzamiento: En la implementación de la Pagina Web es recomendable utilizar estándares (HTML, XHTML).

Para asegurar la futura compatibilidad y escalabilidad del sitio. Una vez implementada la página web y aprobada su funcionalidad se procede al lanzamiento del sitio.

viii. El Mantenimiento

Es el proceso de control, mejora y optimización del software ya desarrollado e instalado, que también incluye depuración de errores y defectos que puedan haberse filtrado de la fase de pruebas de control. (Huanca, 2015, p. 32).

Mantenimiento y Seguimiento: Una vez puesta la Pagina Web a Disposición de los usuarios hay que ir cambiando datos y mantener este sitio actualizado, ya que esta página no puede permanecer estática. Los problemas de uso no detectados durante el proceso de desarrollo pueden descubrirse a través de varios métodos, principalmente a través de los mensajes, opiniones, el comportamiento y uso del sitio. (Huanca, 2015, p. 32).

2.13.7. Ingeniería Móvil

El desarrollo de una aplicación o servicio conlleva una gran incertidumbre. Sin embargo, existen sistemas para paliar los riesgos asociados. (Yujra, 2016, p. 30).

En el caso de las aplicaciones móviles, con el tiempo, han ido apareciendo nuevas dificultades, como el acceso a la información del entorno o el control de las diferentes capacidades de los dispositivos. Al mismo tiempo, las oportunidades de negocio aparecen constantemente, lo que permite crear una gran variedad de aplicaciones en diversos contextos, desde juegos hasta aplicaciones para el hogar. (Yujra, 2016, p. 30).

Según el autor (Zarco, 2017, p. 9). La ingeniería de software móvil es una disciplina que incluye metodologías y técnicas para generar aplicaciones móviles de forma correcta, optimizada y que cumpla con los requerimientos de desarrollo pedidos por el cliente. Esta ingeniería cuenta con diversas etapas o pasos para concretar el proyecto, están incluidas el análisis de requerimientos, la especificación, arquitectura, la programación, las pruebas, la documentación y el mantenimiento.

2.13.7.1. Metodología Mobile-D

Figura 2.8

Metodología Mobile-D

Nota. Se muestra las fases de Mobile-D. Obtenido: (Cruz, 2014, p. 25).

Las metodologías ágiles poseen ciertas propiedades que las hacen totalmente aplicables al desarrollo de proyectos de software móvil. Por esta razón, se ha decidido hacer uso de la metodología Mobile-D, la cual fue creada apoyándose en muchas otras soluciones bien conocidas y consolidadas: Xtreme Program-ming (XP), Crystal methodologies y Rational Unified Process(RUP). (Cruz, 2014, p. 25).

Fases y etapas de la metodología a usar

a) Fase I: Levantamiento de información

En ésta primera fase, se espera adquirir la información necesaria para orientar adecuadamente el proyecto y para lograr esto, será necesario hacer una revisión de la bibliografía relacionada con la solución informática que se busca implementar. Para así proceder posteriormente a realizar las actividades sugeridas por la metodología Mobile-D. (Cruz, 2014, p. 25).

b) Fase II: Exploración

En ésta fase apoyándose en las recomendaciones de la metodología Mobile-D, se pretende llevar a cabo el proceso inicial de planificación, así como también, establecer los conceptos básicos del proyecto. (Cruz, 2014, p. 25).

c) Fase III: Inicialización

Ésta fase está pensada para posibilitar el éxito, por tal motivo, la meta será preparar el proyecto para evitar todos los posibles problemas que puedan surgir durante el desarrollo de la solución informática. Además, se prepararán todos los recursos físicos, tecnológicos y de comunicaciones para las actividades de producción.

d) Fase IV: Producción

En ésta fase se llevará a cabo toda la implementación de la solución informática usando un ciclo de desarrollo iterativo e incremental.

e) Fase V: Estabilización

La metodología Mobile-D contempla ésta etapa para hacer una integración completa del sistema en los casos en que el desarrollo del proyecto involucra grupos trabajando en diferentes módulos o sub-sistemas de un mismo proyecto desde puntos geográficos diferentes. (Cruz, 2014, p. 25).

f) Fase VI: Pruebas y preparación del sistema

Una vez terminado totalmente el desarrollo se pasará a la fase de pruebas, donde se iterará hasta llegar a una versión estable según lo establecido en los requerimientos definidos en las primeras fases. Generar retroalimentación con base en la iteración anterior. (Cruz, 2014, p. 25).

2.13.8. Métricas de calidad ISO/IEC 25000

ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software. (ISO 25000, 2021).

La familia ISO/IEC 25000 es el resultado de la evolución de otras normas anteriores, especialmente de las normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto software, e ISO/IEC 14598, que abordaba el proceso de evaluación de productos software. Esta familia de normas ISO/IEC 25000 se encuentra compuesta por cinco divisiones.

Figura 2.9

División para gestión de calidad

Nota. ISO/IEC 25000. Obtenido: (ISO 25000, 2021).

2.13.8.1 Medición de Calidad ISO/IEC 25010

En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. (ISO 25000, 2021).

Se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente figura:

Figura 2.10

Calidad del producto software

Nota. ISO/IEC 25010. Obtenido: (ISO 25000, 2021).

a) Adecuación Funcional

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Completitud funcional.** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- ✓ **Corrección funcional.** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- ✓ **Pertinencia funcional.** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados. (ISO 25000, 2021).

El punto función se puede calcular mediante la siguiente ecuación.

$$PF = Cuenta\ total * (0{,}65 + 0{,}01 * \sum F_i)$$

Donde:

Cuenta total: es la suma de todas las entradas obtenidas en N° de entradas, N° de salidas, N° peticiones, N° de archivos y N° de interfaces externas.

x: Nivel de confiabilidad del sistema es de (0.65).

y: Nivel de error igual a 0.01

F_i: Son los valores de ajuste de la complejidad, según respuestas de la tabla valores de ajuste de complejidad, bajo la siguiente ponderación.

$$Funcionalidad = \frac{\text{PF calculado}}{\text{PF maxima}} * 100\%$$

b) Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Comportamiento temporal.** Los tiempos de respuesta y procesamiento y los ratios de throughput de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (benchmark) establecido.
- ✓ **Utilización de recursos.** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- ✓ **Capacidad.** Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

c) Compatibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Coexistencia.** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento. (ISO 25000, 2021).
- ✓ **Interoperabilidad.** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

d) Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Capacidad para reconocer su adecuación.** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- ✓ **Capacidad de aprendizaje.** Capacidad del producto que permite al usuario aprender su aplicación.
- ✓ **Capacidad para ser usado.** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- ✓ **Protección contra errores de usuario.** Capacidad del sistema para proteger a los usuarios de hacer errores.
- ✓ **Estética de la interfaz de usuario.** Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario. (ISO 25000, 2021).
- ✓ **Accesibilidad.** Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

Para realizar el cálculo de usabilidad del sistema, aplicamos la siguiente ecuación:

$$FU = \frac{\left(\frac{\sum x_i}{n}\right) * 100}{\mathrm{N}}$$

Donde:

N: número de población.

n: número en la muestra.

e) Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Madurez.** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.

- ✓ **Disponibilidad.** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- ✓ **Tolerancia a fallos.** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- ✓ **Capacidad de recuperación.** Capacidad del producto software para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallo. (ISO 25000, 2021).

$$F(t) = f * e^{(\frac{-\beta}{10} * t)}$$

Donde:

f: Es la funcionalidad del sistema ya calculada 88,9%.

-β: Es la probabilidad de error que puede tener un sistema 0,03(3%).

t: El tiempo que dura una gestión en el sistema 12 meses.

f) Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Confidencialidad.** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- ✓ **Integridad.** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.
- ✓ **No repudio.** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente. (ISO 25000, 2021).
- ✓ **Responsabilidad.** Capacidad de rastrear de forma inequívoca las acciones de una entidad.

- ✓ **Autenticidad.** Capacidad de demostrar la identidad de un sujeto o un recurso. (ISO 25000, 2021).

g) Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Modularidad.** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- ✓ **Reusabilidad.** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- ✓ **Analizabilidad.** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- ✓ **Capacidad para ser modificado.** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño. (ISO 25000, 2021).
- ✓ **Capacidad para ser probado.** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Para determinar la estabilidad de un producto se tiene la siguiente ecuación.

$$IMS = \frac{[\mathrm{Mt} - (\mathrm{Fa} + \mathrm{Fc} + \mathrm{Fd})]}{\mathrm{Mt}}$$

h) Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- ✓ **Adaptabilidad.** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- ✓ **Capacidad para ser instalado.** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- ✓ **Capacidad para ser reemplazado.** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno. (ISO 25000, 2021).

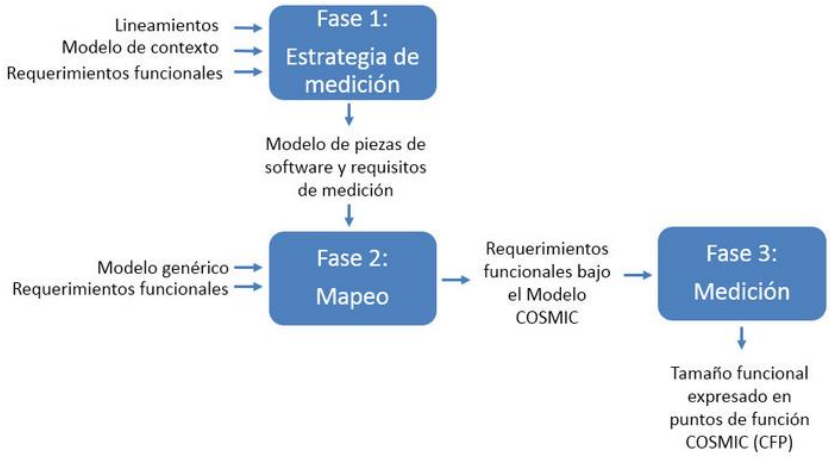
2.13.9. Análisis de costos

2.13.9.1. Método COSMIC

El método COSMIC define los principios, reglas y un proceso para medir un tamaño funcional estándar de una pieza de software. El 'tamaño funcional' es una medida de la cantidad de funcionalidad expresada en términos que un usuario puede comprender. Es completamente independiente de la tecnología utilizada para crearlo. El tamaño de COSMIC generalmente se determina a partir de los requisitos, pero también se puede contar de manera inversa a partir de otros artefactos de software, por ejemplo, diseños, sistemas instalados, etc. (COSMIC, 2021).

Figura 2.11

Proceso de medición COSMIC



Nota. Método de estimación de costos. Obtenido: (Vanquez, 2015).

a) Fase 1: Estrategia de medición

- ✓ Lo primero que se realiza en una medición y estimación de software con COSMIC, es determinar qué es lo que se va a medir.
- ✓ Una medición de software depende del punto de vista de lo que definimos como usuarios funcionales, por ejemplo, personas, dispositivos de hardware u otros sistemas que interactúan con el software. (Vanquez, 2015, pp. 4-7).
- ✓ En esta primera fase se define el propósito y alcance de la medición de software, que incluye cuales son los requerimientos funcionales de usuario que se van a medir, quienes son los usuarios funcionales y otros parámetros. Previo a esto, es necesario haber aplicado técnicas para el levantamiento de requerimientos de software.
- ✓ Es importante dejar documentados los parámetros de la medición de software, para asegurar que esta pueda ser interpretada adecuadamente por quienes harán uso de ella para realizar las estimaciones y presupuestos. (Vanquez, 2015, pp. 4-7).

b) Fase 2: Mapeo

- ✓ En una medición COSMIC, el mapeo se realiza para crear un modelo COSMIC de los requerimientos funcionales de usuario.
- ✓ El punto de partida para el mapeo son los artefactos disponibles, como por ejemplo un esquema o especificación de requerimientos detallada, modelos de diseño como por ejemplo los casos de uso, software que está instalado físicamente, entre otros.
- ✓ Para elaborar este modelo, se utilizan los principios del Modelo genérico de software COSMIC, aplicados a los requerimientos de software que se van a medir. (Vanquez, 2015, pp. 4-7).

El modelo de requerimientos de software COSMIC tiene 4 principios:
1.- La funcionalidad de software está comprendida de procesos funcionales. La tarea de cada proceso funcional es responder a un evento ocurrido fuera de la frontera del sistema (el mundo de los usuarios funcionales).

2.- Los procesos funcionales están compuestos de sub-procesos:

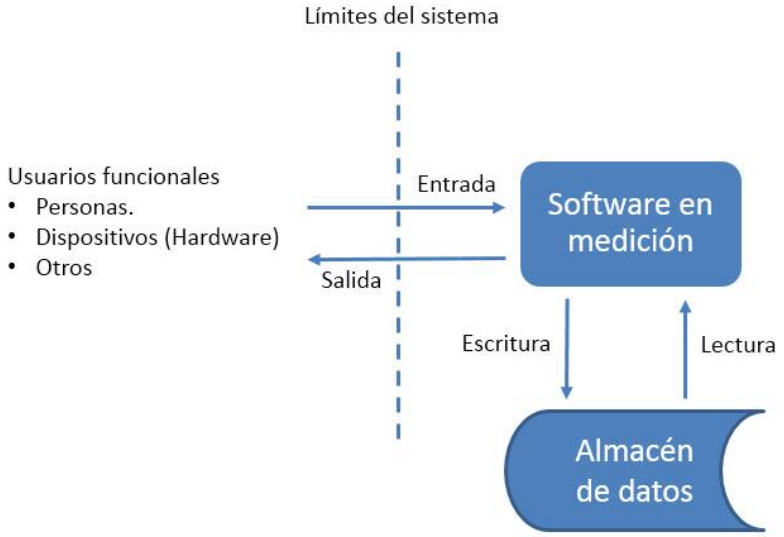
- ✓ Cada sub-proceso puede mover datos o manipular datos.
- ✓ Los sub-procesos de movimiento de datos que mueven datos de un usuario funcional a un proceso funcional se les llama “Entradas”.
- ✓ Los sub-procesos que mueven datos desde un proceso funcional hacia el exterior se les llama salidas.
- ✓ Los sub-procesos que mueven datos hacia un almacén de datos se les llama “Escrituras” mientras que a los que mueven datos desde dichos almacenes se les conoce como "lecturas". (Vanquez, 2015, pp. 4-7).

3.- Cada movimiento de datos (Entrada, salida, lectura o escritura) moviliza solamente un grupo de datos, cuyos atributos describen un solo objeto de interés.

4.- Se asume que la manipulación de datos forma parte de las entradas, salidas, lecturas o escrituras, por lo tanto estas no se miden por separado. La siguiente figura ilustra los tipos de sub-procesos definidos por el modelo COSMIC.

Figura 2.12

Los 4 tipos de movimientos de datos



Nota. Los 4 tipos de movimiento de datos para la estimación. Obtenido: (Vanquez, 2015).

Se entiende que un proceso funcional termina su ejecución cuando ha realizado todos los sub-procesos necesarios para responder a los datos que recibió del evento. (Vanquez, 2015, pp. 4-7).

c) Fase 3: Medición

- ✓ La unidad de medida del método COSMIC es el “punto de función COSMIC” (CFP). Cada movimiento de datos es medido como un (1) CFP.
- ✓ La medición de la nueva pieza de software se realiza identificando todos los movimientos de datos, es decir todas las entradas, salidas, lecturas y escrituras de cada proceso funcional. Luego sumándolas todas.
- ✓ Todo proceso funcional debe tener al menos dos movimientos de datos (al menos una entrada y una salida o una escritura). Solo de esta forma se garantiza que el proceso funcional modelado proporciona un servicio

completo. Por lo tanto, el tamaño funcional mínimo de un proceso es de 2 CFP. (Vanquez, 2015, pp. 4-7).

- ✓ No existe un límite superior al tamaño de un proceso funcional.
- ✓ Para realizar mediciones sobre mejoras a piezas de software existente, se identifican todos los movimientos de datos que se van a agregar, modificar o eliminar, sumándolos todos en cada uno de sus procesos funcionales. El tamaño mínimo de una modificación es de un CFP.

2.13.10. Seguridad de la información ISO/IEC 27000

Las normas que forman la serie ISO/IEC-27000 son un conjunto de estándares creados y gestionados por la Organización Internacional para la Estandarización (ISO) y la Comisión Electrónica Internacional (IEC). Ambas organizaciones internacionales están participadas por multitud de países, lo que garantiza su amplia difusión, implantación y reconocimiento en todo el mundo.

Las series 27000 están orientadas al establecimiento de buenas prácticas en relación con la implantación, mantenimiento y gestión del Sistema de Gestión de Seguridad de la Información (SGSI) o por su denominación en inglés Information Security Management System (ISMS). Estas guías tienen como objetivo establecer las mejores prácticas en relación con diferentes aspectos vinculados a la gestión de la seguridad de la información, con una fuerte orientación a la mejora continua y la mitigación de riesgos. (ISO27000.ES, 2018).

2.13.10.1. Seguridad de la información ISO/IEC 27001

ISO/IEC 27001 es un estándar para la seguridad de la información (Information technology - Security techniques - Information security management systems - Requirements) aprobado y publicado como estándar internacional en octubre de 2005 por International Organization for Standardization y por la comisión International Electrotechnical Commission.

Especifica los requisitos necesarios para establecer, implantar, mantener y mejorar un sistema de gestión de la seguridad de la información (SGSI).

Según el conocido como “Ciclo de Deming”: PDCA - acrónimo de Plan, Do, Check, Act (Planificar, Hacer, Verificar, Actuar). Es consistente con las mejores prácticas descritas en ISO/IEC 27002, anteriormente conocida como ISO/IEC 17799, con orígenes en la norma BS 7799-2:2002, desarrollada por la entidad de normalización británica, la British Standards Institution (BSI). (ISO27000.ES, 2018).

La ISO 27001 es una norma internacional de Seguridad de la Información que pretende asegurar la confidencialidad, integridad y disponibilidad.

Confidencialidad: Entendemos la cualidad de la información para no ser divulgada a personas o sistemas no autorizados, sólo las personas autorizadas pueden acceder a la información.

Integridad: Hace referencia a la cualidad de la información para ser correcta y no haber sido modificada, manteniendo sus datos exactamente tal cual fueron generados sin manipulación ni alteraciones por parte de terceros.

Disponibilidad: Entendemos aquella información a la que podemos acceder cuando la necesitamos a través de los canales adecuados siguiendo los procesos correctos, pueden ser accedida por las personas autorizadas cuando lo necesitan. (ISO27000.ES, 2018).

2.13.10.2. Seguridad de la información ISO/IEC 27002

ISO/IEC 27002 proporciona recomendaciones de las mejores prácticas en la gestión de la seguridad de la información a todos los interesados y responsables en iniciar, implantar o mantener sistemas de gestión de la seguridad de la información. La seguridad de la información se define en el estándar como "la preservación de la confidencialidad (asegurando que sólo quienes estén autorizados pueden acceder a la información), integridad (asegurando que la información y sus métodos de proceso son exactos y completos) y disponibilidad (asegurando que los usuarios autorizados tienen acceso a la información y a sus activos asociados cuando lo requieran)". (ISO27000.ES, 2018).

2.14. PRUEBAS

2.14.1. Prueba de Caja Negra

Por el contrario, las pruebas de caja negra (también llamadas funcionales, de comportamiento o de caja cerrada) describen cualquier prueba o método en el que se desconocen los detalles y el funcionamiento interno del software que se está probando.

Dado que no conoce ninguno de los detalles, realmente no puede crear casos de prueba que se dirijan a escenarios de nicho específicos o que hagan hincapié en la lógica específica del sistema. Lo único que sabe es que, para una solicitud o una entrada determinada, se espera un determinado comportamiento o salida. Por lo tanto, las pruebas de caja negra prueban principalmente el comportamiento de un sistema. Las pruebas de un extremo a otro suelen ser una caja negra. (Peño, 2015, p. 39).

2.14.2. Prueba de Caja Blanca

Las pruebas de caja blanca (también llamadas estructurales o de caja transparente) describen pruebas o métodos en los que se conocen los detalles y el funcionamiento interno del software que se está probando. Dado que conoce las funciones, los métodos, las clases, cómo funcionan y cómo se unen, generalmente está mejor equipado para examinar la integridad lógica del código. Por ejemplo, es posible que sepa que hay una peculiaridad en la forma en que un determinado idioma maneja ciertas operaciones. Podría escribir pruebas específicas para eso, que de otro modo no sabría escribir en un escenario de caja negra. Las pruebas unitarias y las pruebas de integración suelen ser una caja blanca. (Peño, 2015, p. 31).

2.14.3. Prueba de estrés

Consiste en probar los límites que un sistema puede soportar. En este tipo de pruebas se suele enviar más peticiones de las que el software podría atender normalmente para saber el comportamiento de la aplicación. (Chiu, 2018, p. 39).

La prueba de stress, a diferencia de las pruebas de carga, tienen por objetivo determinar el punto de ruptura del servicio y analizar sus causas que suelen estar derivadas de problemas que suceden ante condiciones muy elevadas de carga:

mala escalabilidad, agotamiento de capacidad, fugas de memoria, condiciones de carrera y otras. La prueba de stress se inicia con un número bajo de usuarios que se duplican ciclo a ciclo hasta que determina el punto de ruptura del servicio. Se pueden simular peticiones continuas para aumentar la carga. (Medina, 2014, p. 33).

2.15. TÉCNICAS DE INVESTIGACIÓN

2.15.1. Entrevista

Técnica orientada a obtener información de forma oral y personalizada sobre acontecimientos vividos y aspectos subjetivos de los informantes en relación a la situación que se está estudiando. (Chagona Ramos, 2008, pp. 19-22).

2.15.2. Observación

No es mera contemplación (sentarse a ver el mundo y tomar notas); implica adentrarnos a profundidad en situaciones sociales y mantener un papel activo, así como una reflexión permanente. Estar atento a los detalles, sucesos, eventos e interacciones. (Chagona Ramos, 2008, pp. 19-22).

2.15.3. Cuestionario

Técnica cuantitativa que consiste en una investigación realizada sobre una muestra de sujetos, representativa de un colectivo más amplio que se lleva a cabo en el contexto de la vida cotidiana, utilizando procedimientos estandarizados de interrogación con el fin de conseguir mediciones cuantitativas sobre una gran cantidad de características objetivas y subjetivas de la población. (Chagona Ramos, 2008, pp. 19-22).

2.16. HERRAMIENTAS

2.16.1. PostgreSQL

PostgreSQL, o simplemente Postgres para darle un nombre más pintoresco, es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON.

Como decíamos, se trata de un sistema de código abierto y además gratuito, y su desarrollo es llevado adelante por una gran comunidad de colaboradores de todo el mundo que día a día ponen su granito de arena para hacer de este sistema una de las opciones más sólidas a nivel de bases de datos.

Dos detalles a destacar de PostgreSQL es que posee data types (tipos de datos) avanzados y permite ejecutar optimizaciones de rendimiento avanzadas. Que son características que por lo general solo se ven en sistemas de bases de datos comerciales, como por ejemplo SQL Server de Microsoft u Oracle de la compañía homónima. (PostgreSQL, 2022).

Características

Siendo uno de los sistemas de bases de datos más avanzados y usados del mundo, es obvio que PostgreSQL debe tener algunas características bastante llamativas, así que vamos a echarle un vistazo a algunas de ellas.

Es de código abierto: una de las principales razones por la cual PostgreSQL se ha vuelto tan popular es que se trata de un sistema de código abierto. Esto ha permitido que una gran comunidad de desarrolladores crezca para respaldarlo y continuar mejorándolo. Gracias a todo el apoyo con el que cuenta ha logrado transformarse en uno de los mejores gestores de bases de datos a nivel mundial. (PostgreSQL, 2022).

Es gratuito: como cabe esperarse se trata de un sistema totalmente gratis, no tenemos que pagar nada por utilizarlo. Cualquier persona es libre de descargar PostgreSQL desde su sitio web oficial y darle uso sin ningún costo.

Es multiplataforma: una característica genial que de hecho es común en muchos grandes proyectos de código abierto es el hecho de que se trata de software multiplataforma, es decir, es un software que puede correr bajo distintos entornos y sistemas operativos, y es compatible con muchos de los servidores web más populares como Apache, Nginx y LiteSpeed por mencionar algunos. (PostgreSQL, 2022).

Es fácil de usar: la facilidad de uso de PostgreSQL es sin dudas otra de las principales características de este sistema. Su administración se vuelve muy sencilla por medio de paneles con PgAdmin.

Que básicamente viene a ser un phpMyAdmin orientado para PostgreSQL. La posibilidad de realizar diversos procedimientos en forma sencilla, hacen que PgAdmin sea ampliamente utilizado, aunque también permite realizar tareas más complejas, así que tanto novatos como usuarios expertos hacen uso de él.

Puede manejar un gran volumen de datos: una característica extremadamente importante de PostgreSQL es su gran capacidad para el manejo de grandes volúmenes de datos, algo en lo que otros sistemas como MySQL aún no hacen tan bien. Las bases de datos de gran tamaño pueden hacer pleno uso del MVCC de PostgreSQL, resultando en un gran rendimiento. MVCC es un método de control que nos permite realizar tareas de escritura y lectura simultáneamente.

Soporte total de ACID: otro punto muy importante que no se debe dejar de lado es el cumplimiento de ACID. ¿Qué es ACID? Estas siglas en inglés refieren a: atomicity, consistency, isolation y durability. Que si lo traducimos al español básicamente hablan de la atomicidad, consistencia, aislamiento y durabilidad de las transacciones que se realizan en una base de datos. ¿Y por qué es tan importante? Porque tener soporte completo de ACID da la seguridad de que, si se produce una falla durante una transacción, los datos no se perderán ni terminarán donde no deban. (PostgreSQL, 2022).

2.16.2. Node.js

Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Este entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript. (NodeJS, 2022).

Para qué sirve Node.js

Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente (con entrada nos referimos a solicitudes y con salida a respuestas). Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP.

La idea principal de Node.js es usar el modelo de entrada y salida sin bloqueo y controlado por eventos para seguir siendo liviano y eficiente frente a las aplicaciones en tiempo real de uso de datos que se ejecutan en los dispositivos. La finalidad de Node.js no tiene su objetivo en operaciones intensivas del procesador, de hecho, usarlo para programación de más peso eliminará casi todas sus ventajas. Donde Node.js realmente brilla es en la creación de aplicaciones de red rápidas, ya que es capaz de manejar una gran cantidad de conexiones simultáneas con un alto nivel de rendimiento, lo que equivale a una alta escalabilidad. (NodeJS, 2022).

2.16.3. TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un súper conjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Anders Hejlsberg, diseñador de C Sharp y creador de Delphi y Turbo Pascal, ha trabajado en el desarrollo de TypeScript. TypeScript puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js). (Puciarelli, 2020, p. 9).

TypeScript extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original. TypeScript soporta ficheros de definición que contengan información sobre los tipos de librerías JavaScript existentes. Similares a los ficheros de cabeceras de C/C++ que describen la estructura de ficheros de objetos existentes. Esto permite a otros programas usar los valores definidos en los ficheros como si fueran entidades TypeScript de tipado estático.

El compilador de TypeScript está escrito asimismo en TypeScript, compilado a JavaScript y con Licencia Apache 2. TypeScript se incluye como lenguaje de programación de primer nivel en Microsoft Visual Studio 2013 Update 2 y posteriores, junto a C# y otros lenguajes de Microsoft. Una extensión oficial permite a Visual Studio 2012 soportar también TypeScript. (Puciarelli, 2020, p. 9).

2.16.4. Framework Nest.js

Nest.js es un marco de Nodo progresivo con licencia de MIT de código abierto, escrito en TypeScript y que comparte muchos conceptos con Angular. Es para el lado del servidor y se puede utilizar para crear aplicaciones web eficientes, confiables y escalables para la empresa. Está construido por Kamil Mysliwiec. (Nest.JS, 2022).

Nest.js combina los mejores conceptos de Programación orientada a objetos, Programación funcional y Nest.js tiene una gran cantidad de características tales como:

Extensibilidad: Gracias a su arquitectura modular, Nest le permite usar las otras bibliotecas existentes en su proyecto.

Arquitectura: Nest tiene una arquitectura de proyecto que proporciona una capacidad de prueba sin esfuerzo, escalabilidad y capacidad de mantenimiento.

Versatilidad: Nest proporciona un ecosistema para la creación de todo tipo de aplicaciones de servidor.

Progresividad: Nest utiliza las últimas funciones de JavaScript e implementa soluciones y patrones de diseño maduros en el desarrollo de software. Ya que utiliza TypeScript y los conceptos básicos de Angular, los desarrolladores de Angular pueden aprenderlo rápidamente y podrán crear backends para sus aplicaciones Angular sin tener que recurrir a otros marcos del lado del servidor.

Nest.js hace uso de las bibliotecas existentes y maduras que los desarrolladores de Node.js han utilizado durante mucho tiempo, como Express.js y TypeORM. Express es un marco web minimalista, rápido y sin opciones para Node.js que proporciona muchas utilidades HTTP para crear API REST robustas de forma fácil y rápida.

Para TypeORM es el ORM más maduro (Object Relational Mapper) para el lenguaje TypeScript y el JavaScript moderno. Es compatible con los patrones Active Record y Data Mapper, que le permiten crear aplicaciones escalables, de alta calidad, acopladas de forma flexible y que se pueden mantener sobre los sistemas de bases de datos más populares existentes como MySQL, PostgreSQL y Oracle. (Nest.JS, 2022).

2.16.5. Typeorm

Typeorm es como cualquier otro ORM, permite interactuar con múltiples bases de datos y admite múltiples controladores de bases de datos para typeorm, admite MySQL, MariaDB, PostgreSQL, CockroachDB, SQLite, Microsoft SQL Server sql.js Oracle y MongoDB para bases de datos NoSQL, por lo que, según su caso, simplemente necesita instalar el controlador correspondiente de su base de datos y estará listo para comenzar. (TypeORM, 2022).

2.16.6. Html5

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente, pero, incluso cuando algunas APIs (Interface de Programación de Aplicaciones) y la especificación de CSS3 por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. (Gauchat, 2012, p.1).

2.16.7. Css3

CSS es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc… (Gauchat, 2012, p.31).

2.16.8. Framework Angular

Angular (comúnmente llamado Angular 2+ o Angular 2) es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. (Angular, 2022).

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript.

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el binding de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar). Angular es la evolución de AngularJS aunque incompatible con su predecesor. (Wikipedia, 2020).

2.16.9. Lenguaje Java

La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera bytecodes es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de applets que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto. (Fernandéz, 2005, p. 9).

2.16.10. Android studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes: (Developers, 2020).

- ✓ Un sistema de compilación basado en Gradle flexible
- ✓ Un emulador rápido con varias funciones
- ✓ Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
- ✓ Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- ✓ Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- ✓ Gran cantidad de herramientas y frameworks de prueba

- ✓ Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- ✓ Compatibilidad con C++ y NDK
- ✓ Soporte incorporado para Google_Cloud_Platform, lo que facilita la integración de Google Cloud Messaging y App Engine. (Developers, 2020).

2.16.11. Retrofit

Retrofit es un cliente REST para Android y Java, desarrollada por Square, muy simple y fácil de aprender. (Retrofit, 2022). Permite hacer peticiones GET, POST, PUT, PATCH, DELETE y HEAD, gestionar diferentes tipos de parámetros y parsear automáticamente la respuesta a un POJO.

Supongamos que yo tengo una app android que se va a conectar a un servidor, lo que normalmente se hace es hacer una petición GET a una API la cual es una URL, lo que hará sera hacer una petición al servidor que le devolverá un JSON, entonces esto es muy fácil, lo único que tienes que hacer es importar la librería de Retrofit a tu app de Android.

También se usa cualquier biblioteca HTTP para comunicarse con la solicitud y la respuesta el servidor y luego usar varios análisis sintácticos disponibles para analizar su respuesta de nuevo por ejemplo GSON que se encarga de analizar la respuesta del JSON. (Retrofit, 2022).

2.16.12. SQLite

Es un motor de base de datos SQL transaccional de código abierto, ligero, autónomo, de configuración simple y sin servidor. Que se caracteriza por almacenar información persistente de forma sencilla, SQLite gracias a sus características se diferencia de otros gestores de bases de datos, proporcionando grandes ventajas sobre ellos. Así mismo, por ser de dominio público es gratuito tanto para fines privados como para comerciales, se puede descargar de forma libre desde su sitio oficial.

Es importante mencionar que SQLite cuenta con varios enlaces a lenguajes de programación entre los que podemos destacar:

Java, C, C ++, JavaScript, C #, Python, VB Script, entre otros. Consideramos como puntos clave para la utilización de este motor de base de datos los siguientes:

Configuración sencilla: Una vez instalado este motor de base de datos no requiere configuración de rutas, tamaños, puertos, entre otros puntos que por lo general configuramos al inicio de una instalación de cualquier otro motor. (Muradas, 2018).

No demanda el soporte de un servidor: Implementa una serie de librerías que se encargan de la gestión y por ende no ejecuta procesos para administrar la información.

Es Software Libre: Por ser de código abierto, tanto los archivos de compilación como las instrucciones de escalabilidad, se encuentran disponibles para toda la comunidad de desarrolladores.

Genera un archivo para el esquema: SQLite almacena toda la base de datos en un archivo único multiplataforma. Siendo este punto una gran ventaja en cuanto a temas de seguridad y migración, puesto que los datos de las apps desarrolladas para Android no son accedidos por contextos externos, así mismo simplifica las copias de seguridad y los procesos de migración. (SQLite, 2022).

Almacena los datos de forma persistente: Permitiendo que, aunque se apague el dispositivo una vez se encienda los datos persistan y se encuentren correctos en la aplicación. (SQLite, 2022).

SQLite cumple con las características ACID (atomicidad, consistencia, aislamiento y durabilidad), forma parte integral de las aplicaciones basadas en el cliente, SQLite utiliza una sintaxis SQL dinámica y realiza múltiples tareas para hacer lecturas y escrituras al mismo tiempo, ambas (lectura y escritura) se efectúan directamente en los archivos de disco ordinarios. Para reducir la latencia se cuenta con una biblioteca SQLite, la cual es llamada dinámicamente a través de funciones simples y los programas de la aplicación utilizan esta funcionalidad, de igual forma implementa el estándar SQL-92 y usa un sistema inusual para sistemas de administración de bases de datos compatibles con SQL. (SQLite, 2022).

CAPÍTULO III

MARCO APLICATIVO

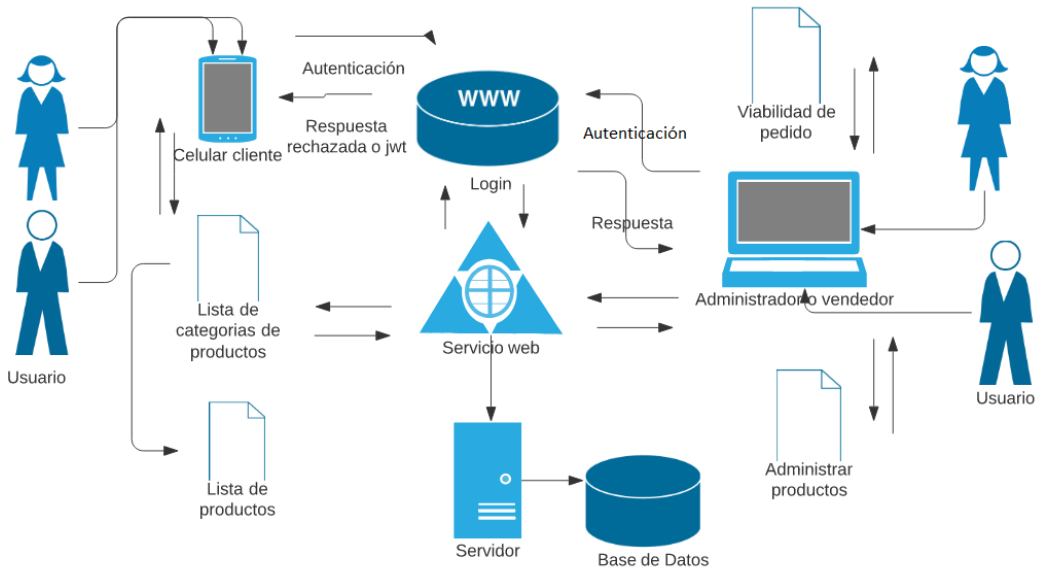
3.1. INTRODUCCIÓN

El desarrollo web para el administrador fue enmarcado con la metodología UWE, que es usada especialmente para desarrollo web, donde usamos como lenguaje TypeScript, Framework Angular, html, css, y para la aplicación móvil la metodología Mobile-D, siendo la misma para el desarrollo ágil del software, como también es utilizado para un grupo pequeño de desarrolladores, dando uso de lenguaje Java, Android studio, retrofit, para el backend se usó Framework NestJS, TypeORM.

3.2. ESQUEMA DEL SISTEMA

Figura 3.1

Esquema del sistema



Nota. Se muestra el funcionamiento completo del sistema con los actores.

3.3. DESARROLLO DEL COMPONENTE ADMIN EN UWE

3.3.1. Análisis de requisitos

La tarea de análisis de requisitos es de vital importancia para recabar la información para la construcción de un sistema, para que se pueda tener los datos precisos y necesarios para el buen desarrollo del software.

Tabla 3.1

Obtención de requisitos.

Tarea	Descripción
Entrevista	Se entrevistó a: Responsable o propietario de la micromarket donde es la persona que administra el negocio.
Observación	Se observó: Por lo cual se obtuvo un panorama más claro de la situación actual.
Cuestionario	Por medio de cuestionario se vio unas cuantas debilidades.

Nota. Métodos de obtención de datos para el desarrollo del sistema.

3.3.1.1. Definición de actores

Al identificar a los actores nos permitió conocer a las personas que interactúan en el proceso de pedidos.

Tabla 3.2

Definición de actores.

Actores	Descripción
Administrador	El administrador es el que tiene acceso y privilegio que los demás usuarios no tienen.
Cliente	Cliente es la persona que hace su pedido de los productos.
Vendedor	Vendedor es la persona encargada de coordinar y aprobar los pedidos.
Repartidor	La persona que se encarga de la entrega de los pedidos.

Nota. Se muestra los actores en el sistema.

3.3.1.2. Requerimientos funcionales

Los requerimientos funcionales son de vital importancia ya que con ellos se define el comportamiento del sistema.

Tabla 3.3

Requerimientos funcionales.

N°	Descripción	Evidente/oculto
R1	Plataforma administrador .	Evidente
R2	Altas bajas de productos.	Evidente

R3	Altas bajas de categorías.	Evidente
R4	Altas bajas de usuarios.	Evidente
R5	Dar viabilidad de pedidos de productos.	Evidente
R6	La web necesita autenticación de usuario.	Evidente
R7	Altas bajas de compras	Evidente
R8	Altas bajas ventas	Evidente
R9	Altas bajas proveedores	Evidente

Nota. Se puede observar los requisitos funcionales del sistema.

3.3.1.3. Requerimientos no funcionales

Los requerimientos no funcionales permiten ver los aspectos visibles para el usuario.

Tabla 3.4

Requerimientos no funcionales.

N°	Descripción	Evidente/ oculto
R1	Requiere un navegador web actualizado para su mejor rendimiento.	Evidente
R2	La web se visualiza en la mayoría de navegadores.	Evidente
R3	Los datos de ingreso son verificados y validados.	Oculto
R4	Interfaz de usuario de fácil manejo.	Evidente
R5	La web está en constante mejora.	Evidente

Nota. Se muestra los requisitos no funcionales del sistema.

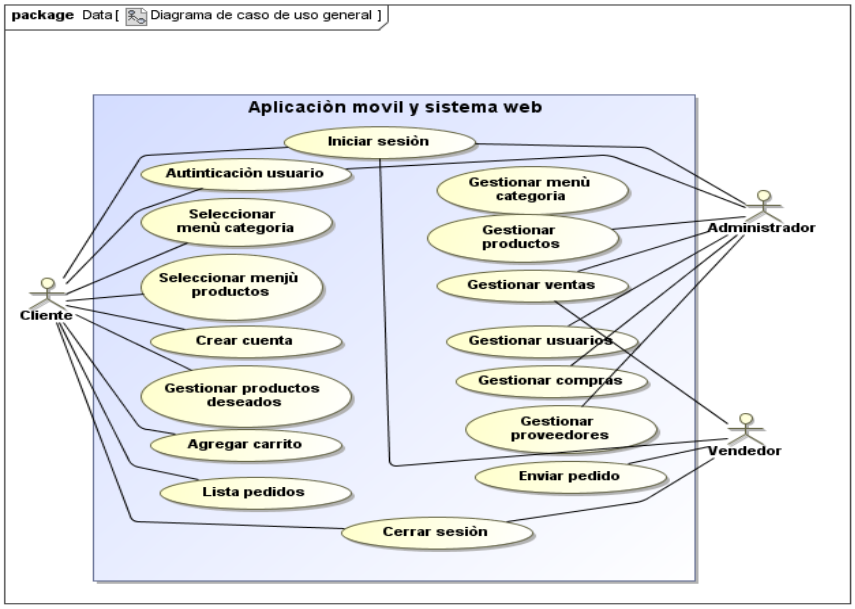
3.3.2. Modelo Casos de Uso

3.3.2.1. Diagrama de caso de uso general del sistema y aplicación móvil

En la figura N° 3.2 se muestra el caso de uso general del sistema y aplicación móvil para la administración de inventario, venta y compras online de productos de consumo masivo.

Figura 3.2

Caso de Uso General del Sistema

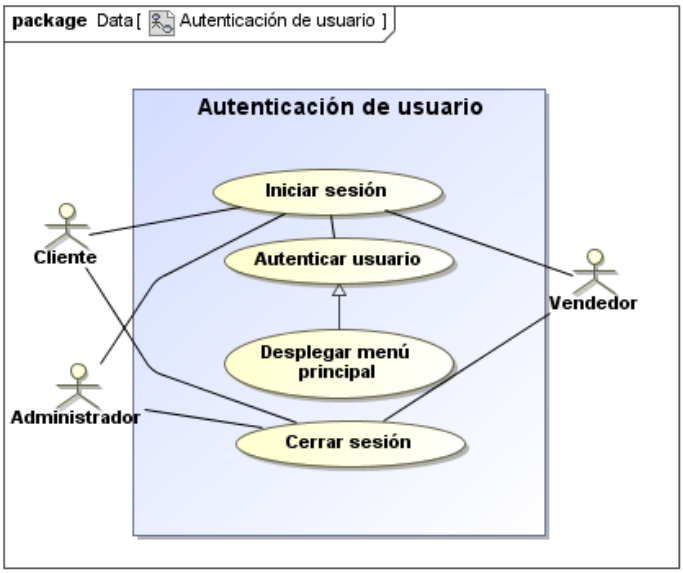


Nota. Se muestra caso de uso general del sistema y aplicación.

3.3.2.2. Diagrama de caso de uso autentificación de usuario

Figura 3.3

Caso de Uso autentificación de usuarios



Nota. Se muestra caso de uso de autentificación al sistema.

Tabla 3.5

Caso de Uso autentificación de usuarios.

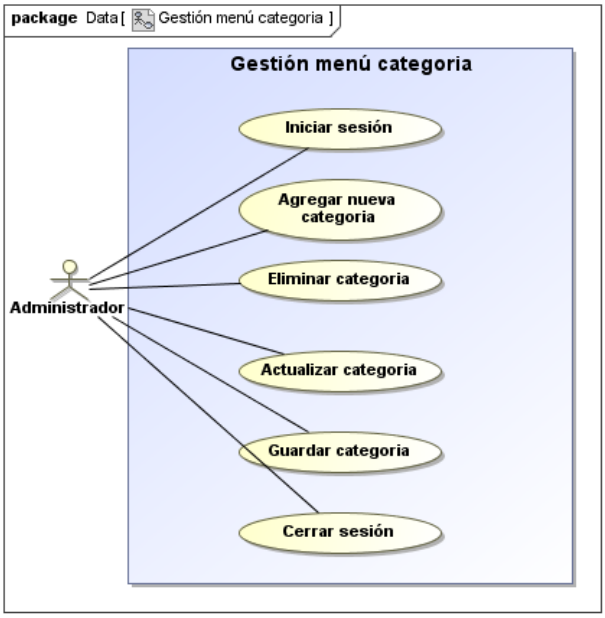
Caso de uso	**Autentificación de Usuarios**	
Actor	Administrador, Cliente, Vendedor	
Descripción	El caso de uso comienza cuando el administrador, cliente, vendedor, quiere ingresar al sistema con su cuenta de usuario.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El administrador, cliente, vendedor, quiere iniciar sesión en la aplicación.	3.-El sistema lanza un formulario de login para que ahí el usuario ingrese los datos requeridos.
	2.-El usuario entra al menú para login.	5.-El sistema valida los datos introducidos, sino son válidos vuelve a requerirlos.
	4.-El usuario introduce los datos	6.-El sistema verifica los datos ingresados con lo registrado en la base de datos.
		7.-El sistema devuelve la interfaz.
Precondición	El usuario debe tener una cuenta registrada.	
Post condición	Los datos registrados deben ser almacenados en la base de datos.	

Nota. Se muestra el proceso que un actor interactúa con el sistema.

3.3.2.3. Diagrama de caso de uso gestión menú categoría

Figura 3.4

Caso de Uso gestión menú categoría



Nota. Se muestra caso de uso de gestión menú categoría.

Tabla 3.6

Caso de Uso gestión menú categoría.

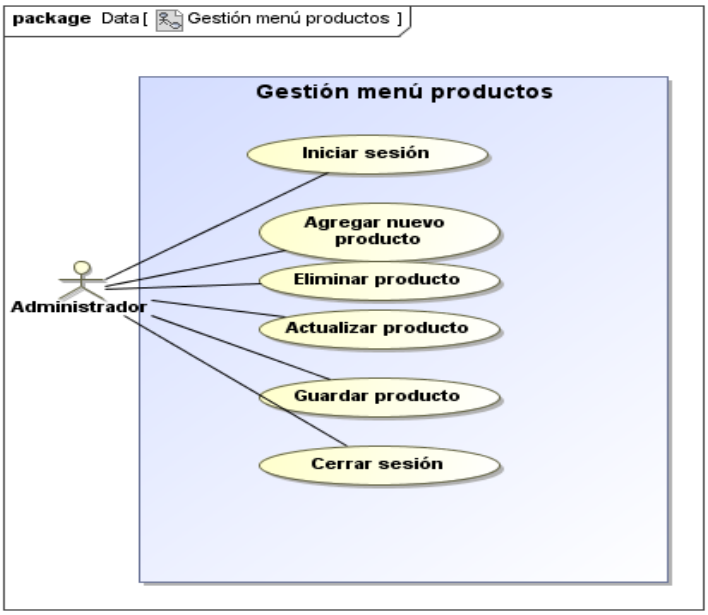
Caso de uso	**Gestión Menú Categoría**	
Actor	Administrador	
Descripción	El caso de uso comienza cuando el administrador necesita realizar cambios.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El administrador quiere realizar altas bajas en la aplicación.	2.-El sistema devuelve formulario para realizar altas bajas al sistema.
	3.-El administrador realizar los cambios.	4.-El sistema valida los datos.
		5.-El sistema guarda los datos.
Precondición	Debe haber iniciado sesión como administrador.	
Post condición	Los cambios no deben afectar en el funcionamiento del sistema.	

Nota. Se muestra la interacción del administrador con el sistema.

3.3.2.4. Diagrama de caso de uso gestión menú productos

Figura 3.5

Caso de Uso gestión menú productos



Nota. Se muestra caso de uso de gestión menú productos.

Tabla 3.7

Caso de Uso gestión menú productos.

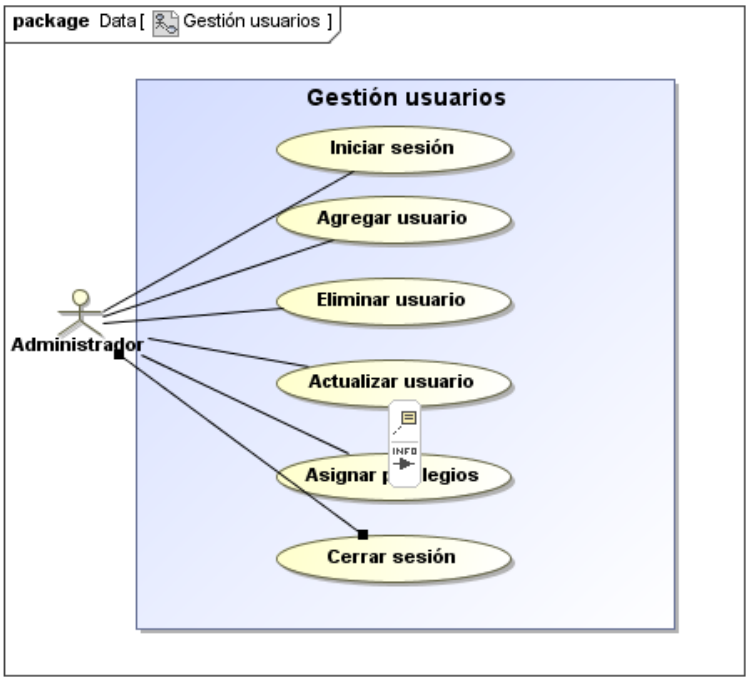
Caso de uso	**Gestión Menú Productos**	
Actor	Administrador	
Descripción	El caso de uso comienza cuando el administrador necesita realizar cambios.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El administrador quiere realizar altas bajas en la aplicación.	2.-El sistema devuelve formulario para realizar altas bajas al sistema.
	3.-El administrador realizar los cambios.	4.-El sistema valida los datos.
		5.-El sistema guarda los datos.
Precondición	Debe haber iniciado sesión como administrador.	
Post condición	Los cambios no deben afectar en el funcionamiento del sistema.	

Nota. Se muestra la interacción del administrador con el sistema.

3.3.2.5. Diagrama de caso de uso gestión usuarios

Figura 3.6

Caso de Uso gestión usuarios



Nota. Se muestra caso de uso de gestión de usuarios.

Tabla 3.8

Caso de Uso gestión usuarios.

Caso de uso	**Gestión Usuarios**	
Actor	Administrador	
Descripción	El caso de uso comienza cuando el administrador quiere realizar cambios en los usuarios.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El administrador quiere realizar altas bajas en los usuarios registrados.	2.-El sistema devuelve formulario para realizar altas bajas al sistema.
	3.-El administrador realizar los cambios.	4.-El sistema valida los datos.
	6.-El administrador quiere asignar privilegios a usuarios.	5.-El sistema guarda los datos.

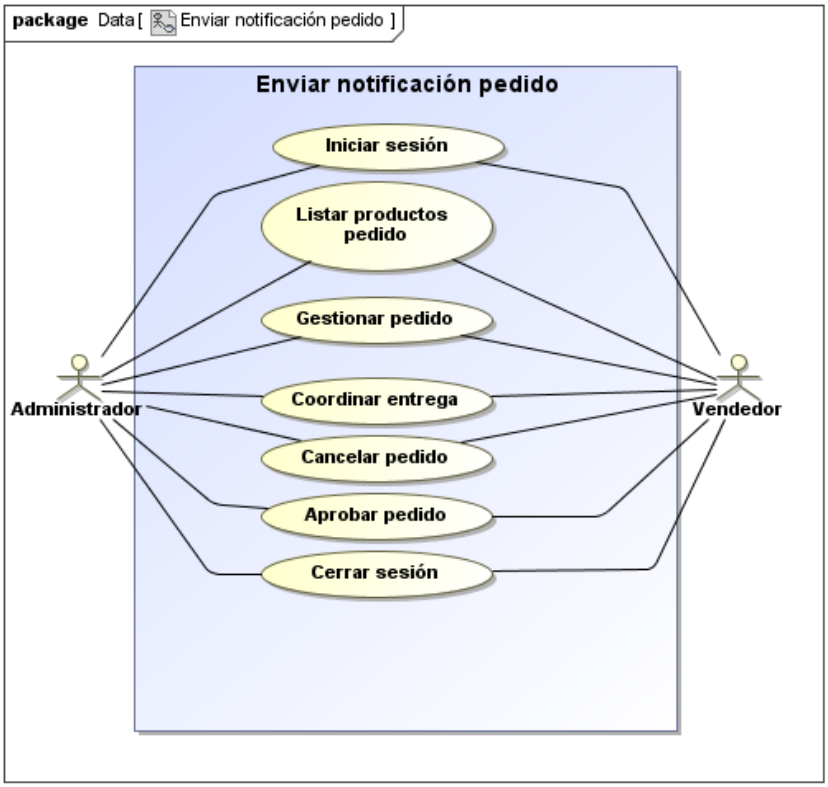
	8.-El administrador asigna privilegios a usuarios.	7.-El sistema devuelve formulario de asignación de privilegios. 9.-El sistema guarda la asignación.
Precondición	Debe haber iniciado sesión como administrador.	
Post condición	Los cambios no deben afectar en el funcionamiento del sistema.	

Nota. Se muestra la interración del administrador con gestión de usuarios.

3.3.2.6. Diagrama de caso de uso enviar notificación pedido

Figura 3.7

Caso de Uso notificación de pedido



Nota. Se muestra caso de uso del proceso de pedido.

Tabla 3.9

Caso de Uso enviar notificación pedido.

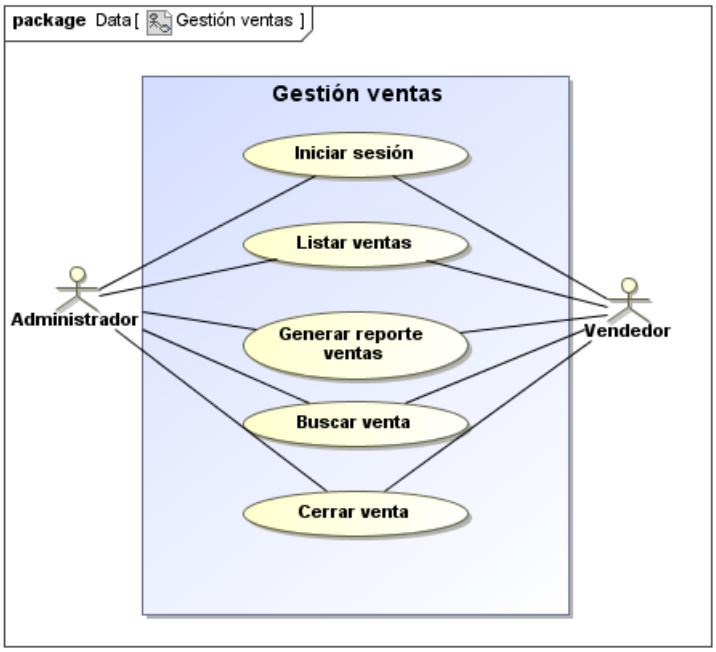
Caso de uso	**Enviar Notificación Pedido**	
Actor	Cliente, Administrador, Vendedor	
Descripción	El caso de uso comienza cuando el cliente envía su pedido.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente realiza pedido.	2.-El sistema guarda la lista de pedidos.
	4.-El vendedor coordina la entrega y aprueba pedido.	3.-El sistema envía en tiempo real la lista de pedidos.
	5.-El vendedor, administrador quieren ver la lista de pedidos.	6.-El sistema devuelve la lista de pedidos.
	7.-El administrador quiere gestionar los pedidos.	8.-El sistema devuelve ventana.
	9.-El administrador gestiona pedidos.	10.-El sistema guarda los cambios.
Precondición	El cliente debe enviar el pedido.	
Post condición	El pedido debe llegar al vendedor en tiempo real.	

Nota. Se muestra la interacción de los actores con el sistema.

3.3.2.7. Diagrama de caso de uso ventas

Figura 3.8

Caso de Uso ventas



Nota. Se muestra caso de uso del proceso de ventas.

Tabla 3.10

Caso de Uso ventas.

Caso de uso	**Ventas**	
Actor	Cliente, Administrador, Vendedor	
Descripción	El caso de uso comienza cuando el vendedor o admin aprueba pedido.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El vendedor o admin aprueba venta.	2.-El sistema guarda aprobación.
	3.-El vendedor o admin selecciona listar productos vendidos.	4.-El sistema desplaza ventas realizados según parámetros.
	5.-El vendedor o admin realiza generar reportes.	6.-El sistema genera reportes de ventas según parámetros.
		8.-El sistema devuelve ventana.

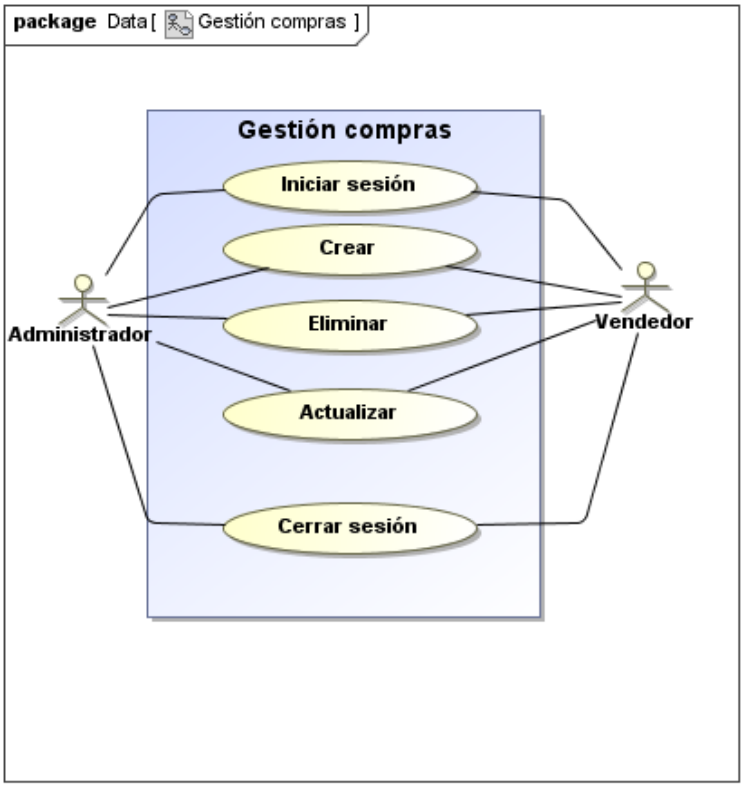
	7.-El administrador o admin realiza búsqueda según parámetro una venta realizada.	9.-El sistema desplaza la venta realizada.
Precondición	El cliente debe enviar el pedido.	
Post condición	El pedido debe llegar al vendedor en tiempo real.	

Nota. Se muestra la interacción en el proceso de ventas.

3.3.2.8. Diagrama de caso de uso compras

Figura 3.9

Caso de Uso compras



Nota. Se muestra caso de uso del proceso de compra.

Tabla 3.11

Caso de Uso compras.

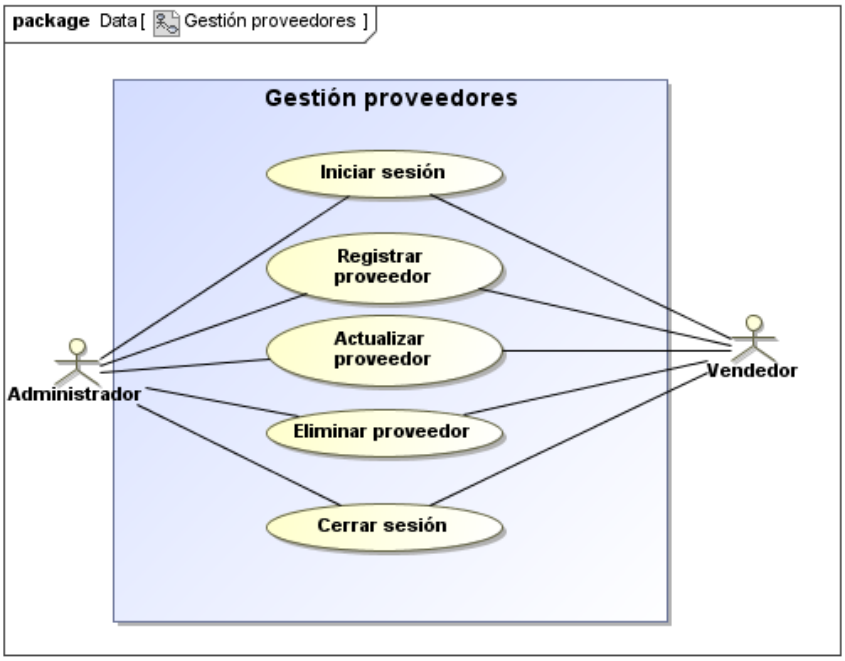
Caso de uso	**Compras**	
Actor	Administrador, Vendedor	
Descripción	El caso de uso comienza cuando admin o vendedor quieren realizar altas bajas.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El vendedor o admin quiere agregar una compra	2.-El sistema guarda la compra.
	3.-El vendedor o admin realiza una actualización de compras.	4.-El sistema visualiza el registro para actualizar.
	5.-El vendedor o admin realiza un listado de compras.	6.-El sistema devuelve la lista de compras.
	7.-El vendedor y admin realiza eliminación de una compra.	8.-El sistema devuelve ventana. 9.-El sistema guarda los cambios.
Precondición	El cliente debe enviar el pedido.	
Post condición	El pedido debe llegar al vendedor en tiempo real.	

Nota. Se muestra la interacción del administrador y vendedor.

3.3.2.9. Diagrama de caso de uso proveedor

Figura 3.10

Caso de Uso proveedor



Nota. Se muestra caso de uso de gestión de proveedores.

Tabla 3.12

Caso de Uso proveedor.

Caso de uso	**Proveedores**	
Actor	Administrador, vendedor	
Descripción	El caso de uso comienza cuando admin o vendedor realizan altas bajas.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El vendedor o admin requiere agregar a un proveedor.	2.-El sistema guarda datos del proveedor.
	3.-El vendedor o admin realiza actualización de un proveedor.	4.-El sistema modifica registro.
		6.-El sistema devuelve la lista de proveedor.

	5.-El vendedor o admin realiza lista de proveedor.	8.-El sistema devuelve ventana.
	7.-El vendedor o admin realiza eliminación de un proveedor.	9.-El sistema guarda los cambios.
Precondición	El cliente debe enviar el pedido.	
Post condición	El pedido debe llegar al vendedor en tiempo real.	

Nota. Se muestra la interacción para la gestión de proveedores.

3.3.3 Modelo Conceptual

El propósito del modelo de contenido es proporcionar una especificación visual de la información relevante para el fácil manejo del sistema web del lado de administrador.

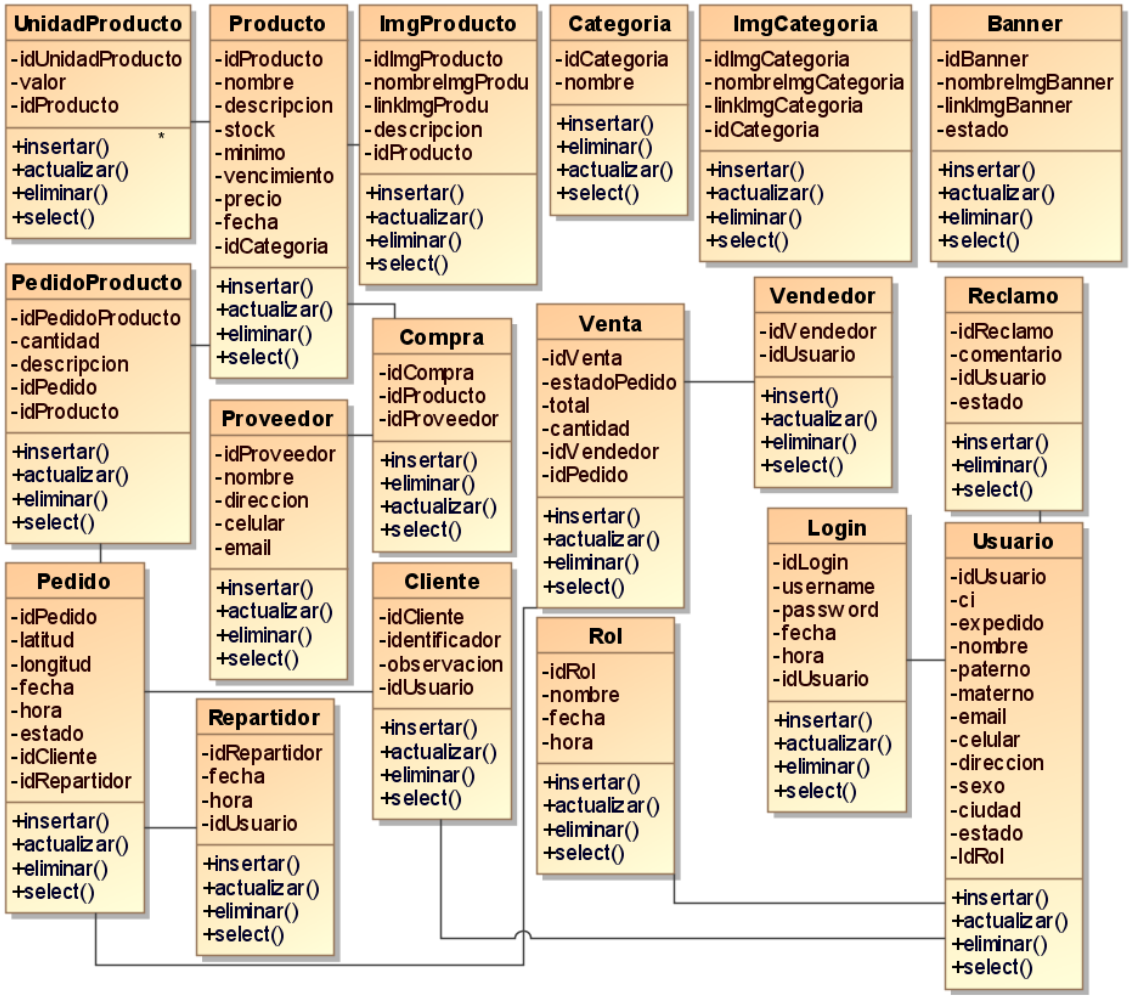
El diseño conceptual del sistema web para el lado administrador se puede observar en la siguiente figura.

3.3.3.1. Diagrama de clases

Según lo requerimientos recabados con los involucrados directamente con la institución se realizó un diagrama de clases, lo cual está representado por los objetos principales del sistema y la interrelación entre ellos.

Figura 3.11

Diagrama de clases



Nota. Se muestra el diagrama de clases.

3.3.3.2. Diagrama relacional de la base de datos

Para el diseño de la base de datos se tomó los requerimientos que se consensuaron con los actores directamente involucrados, donde se muestra los datos necesarios para ser almacenados.

Figura 3.12

Diagrama relacional

Nota. Se muestra el diagrama relacional de las tablas de la base de datos del sistema.

3.3.3.3. Mapeo objeto relacional

Figura 3.13

Mapeo objeto relacional

```
@Entity('producto')
export class Producto extends  BaseEntity{

    @PrimaryGeneratedColumn('increment')
    idproducto:number;

    @Column({type:'varchar', length:100, unique:true, nullable:false})
    nombre:string;

    @Column({type:'varchar', length:100})
    descripcion:string;
```

Nota. Se muestra el mapeo objeto relacional por medio de TypeORM.

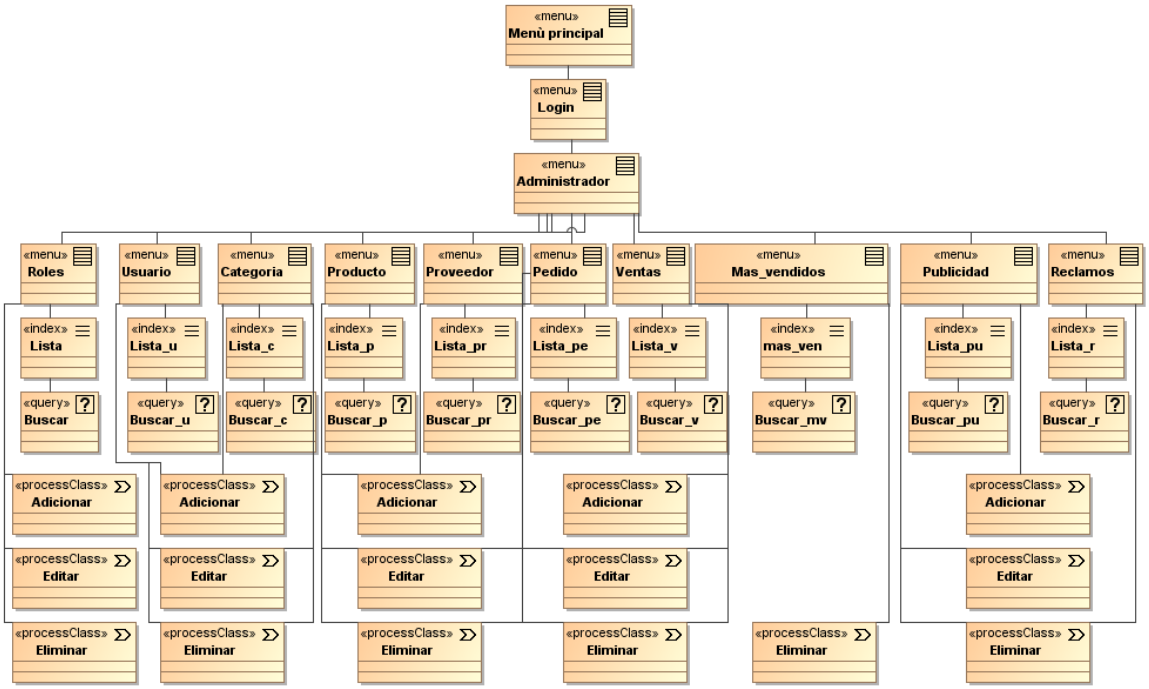
3.3.4. Modelo de Navegación

En los modelos de navegación se muestra todo lo que un usuario puede desplazarse en el sistema, teniendo acceso a cada uno de ellos.

3.3.4.1 Modelo de Navegación administrador

Figura 3.14

Modelo navegación menú administrador

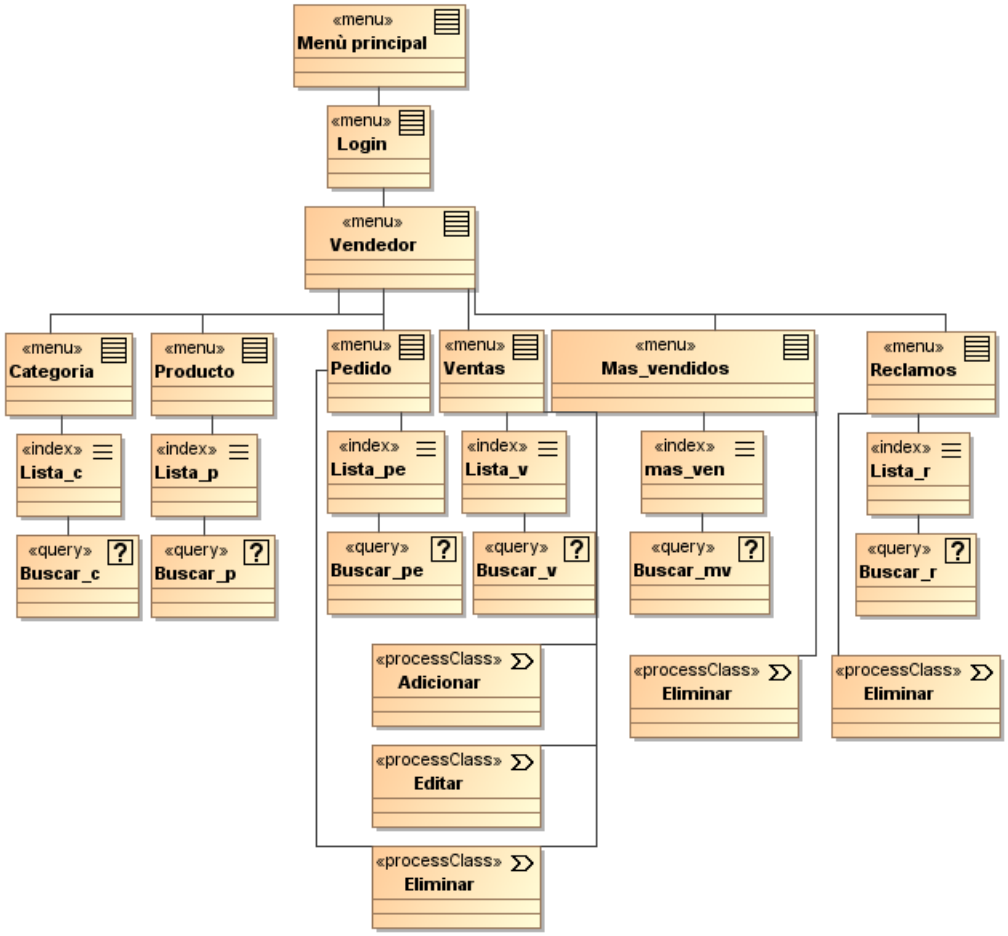


Nota. Se muestra el modelo de navegación menú administrador donde puede realizar acciones de altas bajas, de las principales funcionalidades que tiene acceso el mismo.

3.3.4.2 Modelo de Navegación vendedor

Figura 3.15

Modelo navegación menú vendedor



Nota. Se muestra el modelo navegación menú vendedor mediante el cual tiene acceso a una limitada funcionalidad de todo el sistema por razones de seguridad.

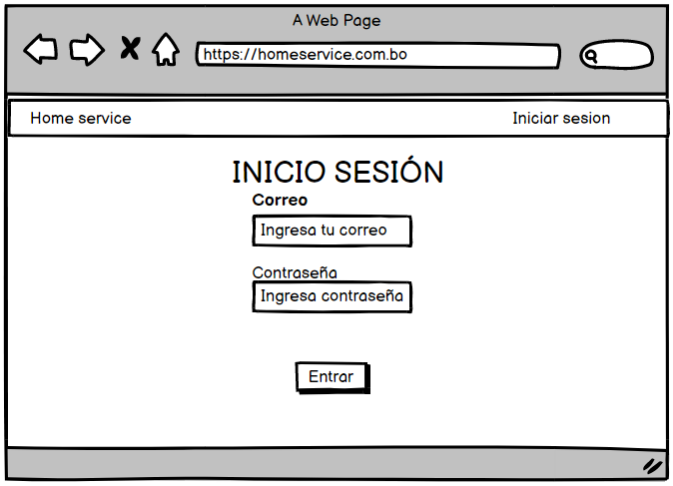
3.3.5. Modelo de Presentación

En el modelo de presentación se muestra las principales ventanas que tiene el sistema, donde cada uno de los usuarios puede interactuar con el mismo.

3.3.5.1 Modelo de Presentación inicio sesión

Figura 3.16

Inicio de sesión

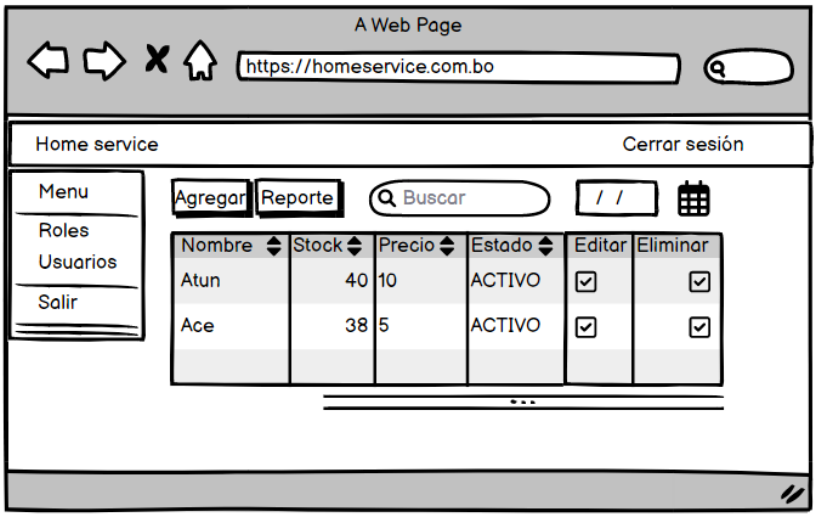


Nota. Se muestra la ventana de inicio de sesión para el administrador, vendedor.

3.3.5.2 Modelo de Presentación producto

Figura 3.17

Menú productos

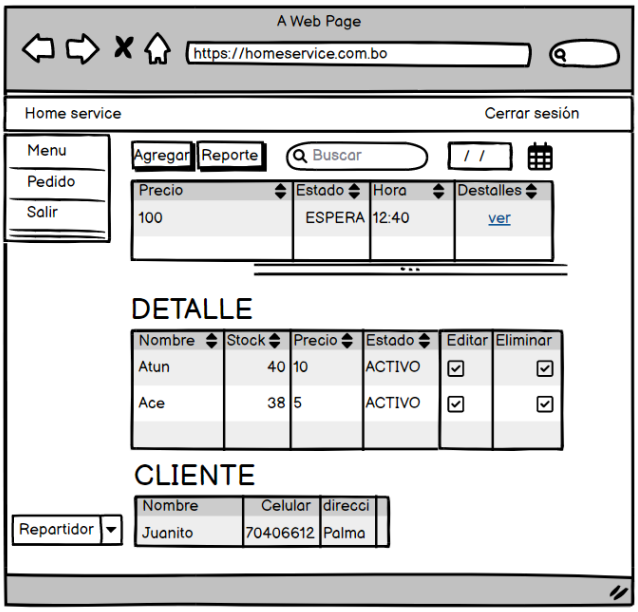


Nota. Se muestra la ventana productos en stock para ser vendidos.

3.3.5.3 Modelo de Presentación pedido

Figura 3.18

Menú pedidos

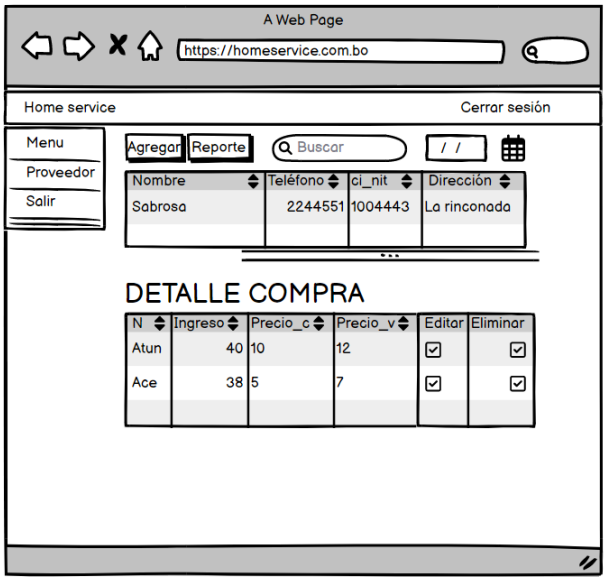


Nota. Se muestra la ventana de pedidos y sus detalles entrantes.

3.3.5.4 Modelo de Presentación proveedor

Figura 3.19

Menú proveedores



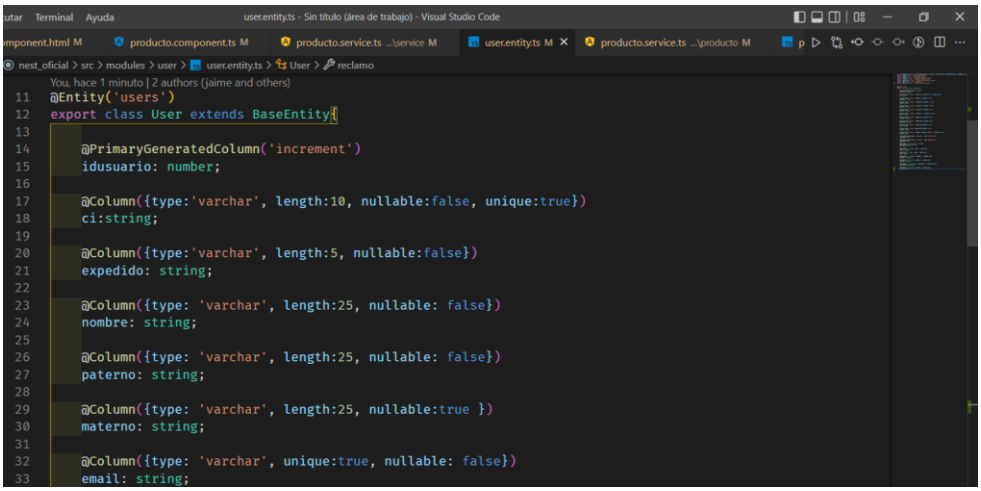
Nota. Se muestra la ventana de proveedor y detalles de compra.

3.3.6. Modelo Implementación

a). Implementación de base de datos

Figura 3.20

Entity usuario

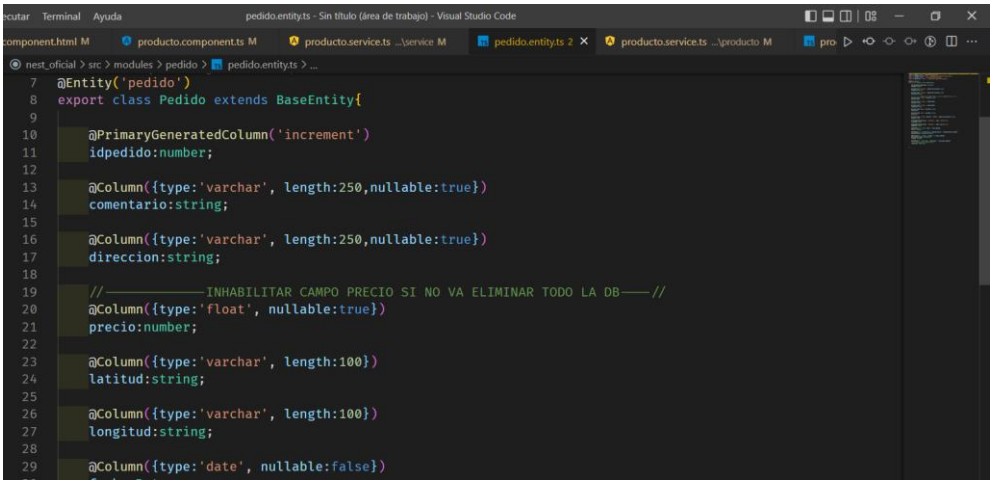


```
@Entity('users')
export class User extends BaseEntity{

    @PrimaryGeneratedColumn('increment')
    idusuario: number;

    @Column({type:'varchar', length:10, nullable:false, unique:true})
    ci:string;

    @Column({type:'varchar', length:5, nullable:false})
    expedido: string;

    @Column({type: 'varchar', length:25, nullable: false})
    nombre: string;

    @Column({type: 'varchar', length:25, nullable: false})
    paterno: string;

    @Column({type: 'varchar', length:25, nullable:true })
    materno: string;

    @Column({type: 'varchar', unique:true, nullable: false})
    email: string;
```

Nota. Se muestra la implementación de la tabla usuario mediante la herramienta TypeORM.

Figura 3.21

Entity pedido



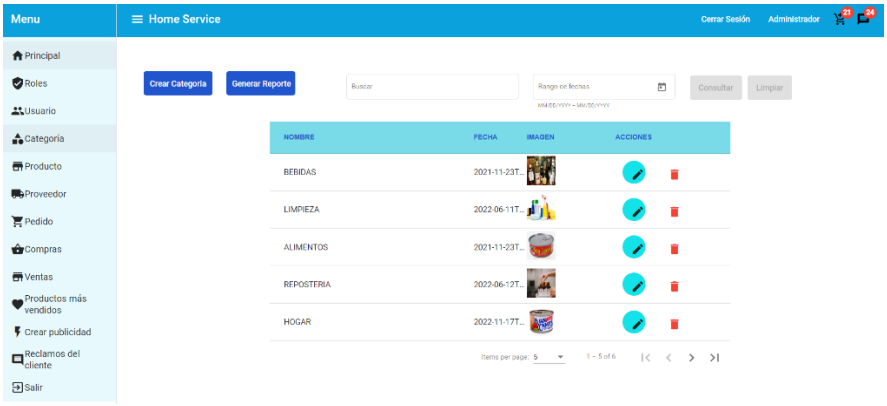
```
@Entity('pedido')
export class Pedido extends BaseEntity{

    @PrimaryGeneratedColumn('increment')
    idpedido:number;

    @Column({type:'varchar', length:250,nullable:true})
    comentario:string;

    @Column({type:'varchar', length:250,nullable:true})
    direccion:string;

    //————————INHABILITAR CAMPO PRECIO SI NO VA ELIMINAR TODO LA DB——//
    @Column({type:'float', nullable:true})
    precio:number;

    @Column({type:'varchar', length:100})
    latitud:string;

    @Column({type:'varchar', length:100})
    longitud:string;

    @Column({type:'date', nullable:false})
```

Nota. Se muestra la implementación de pedido mediante ORM.

b). Implementación interfaz de usuario

Figura 3.22

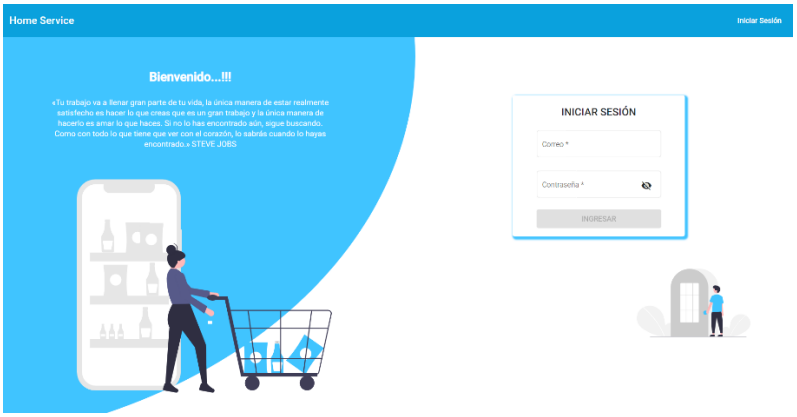
Pantalla principal



Nota. Se muestra la ventana principal del menú administrador.

Figura 3.23

Login



Nota. Se muestra la ventana para acceder al sistema administrador.

Figura 3.24

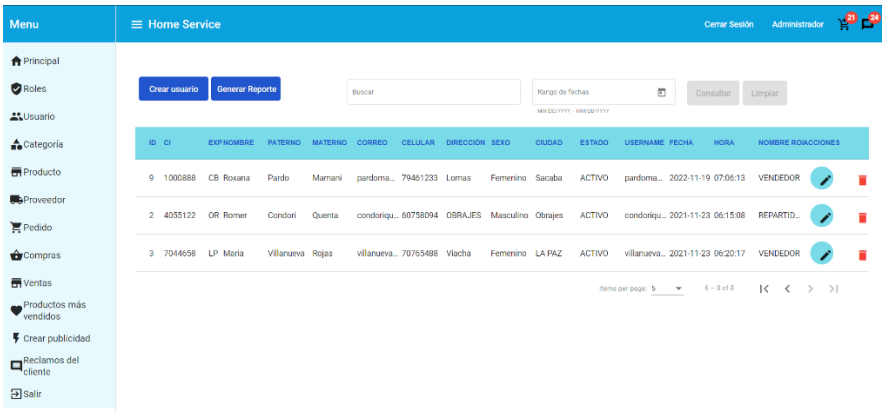
Código fuente login

```
<div>
    <mat-card>
    <mat-card-content>
        <div class="header">
            <h2>INICIAR SESIÓN</h2>
        </div>
        <form   [formGroup]="loginForm" (ngSubmit)="onLogin()" name="loginForm">

                <mat-form-field class="full-width-input" appearance="outline">
                    <mat-label>Correo</mat-label>
                    <input type="username" formControlName="username" matInput required/>
                    <mat-error *ngIf="isValidField('username')">
                    {{getErrorMessage("username")}}
                    </mat-error>
                </mat-form-field>


                <!--<span class="alink">Olvidó su contraseña?</span>-->
                <mat-form-field class="full-width-input" appearance="outline" separator>
                    <mat-label>Contraseña</mat-label>
```

Nota. Se muestra la implementación del código fuente login.

Figura 3.25

Lista de usuarios



Nota. Se muestra la ventana del menú lista de usuarios.

Figura 3.26

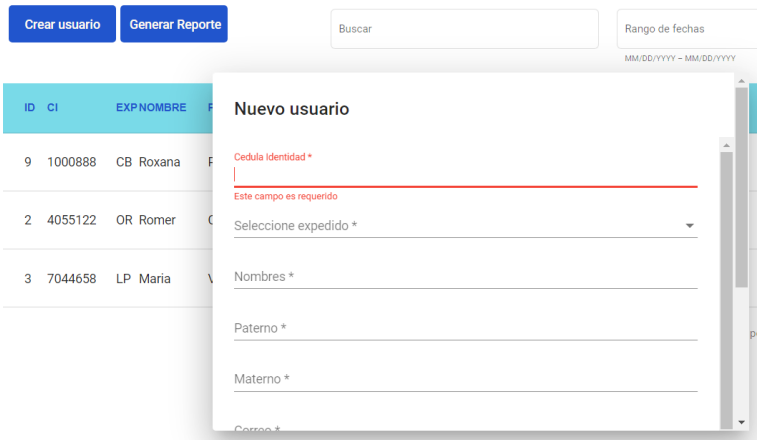
Código fuente lista usuarios

```
<div class="table-button-row">

  <button mat-flat-button class="button-rosado" (click)="onOpenModal()">Crear usuario</butto
  <mat-form-field appearance="outline" class="search-form-field" floatLabel="never">
    <input matInput   placeholder="Buscar" autocomplete="off" (keyup)="applyFilter($event.ta
    <!--<button mat-button matSuffix mat-icon-button aria-label="clear" *ngIf="searchKey" (c
          <mat-icon>close</mat-icon>
    </button>-->
  </mat-form-field>
</div>

<table mat-table [dataSource]="dataSource" matSort >

  <!-- Position Column -->
  <ng-container matColumnDef="idusuario">
    <th mat-header-cell *matHeaderCellDef mat-sort-header> ID </th>
    <td mat-cell *matCellDef="let element"> {{element.idusuario}} </td>
```

Nota. Se muestra el código fuente de la ventana usuarios.

Figura 3.27

Nuevo usuario



Nota. Se muestra la ventana para agregar un nuevo usuario al sistema.

Figura 3.28

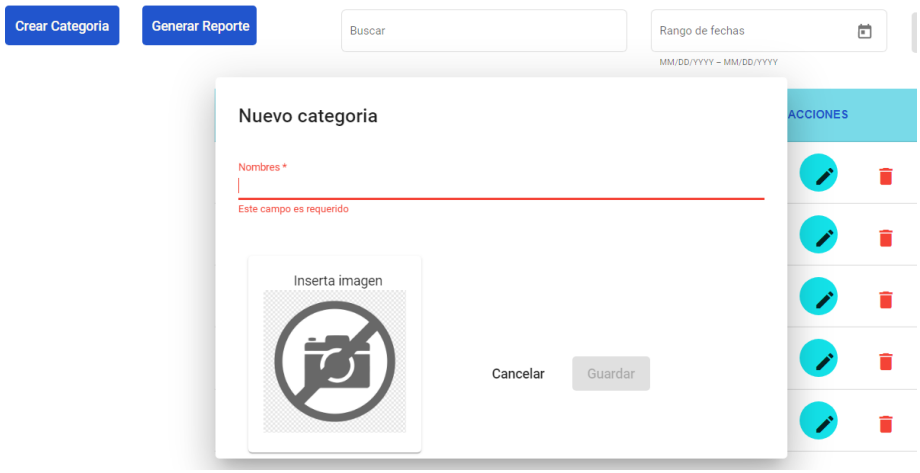
Código fuente nuevo usuario

```
                <mat-option value="BE">BE</mat-option>
                <mat-option value="PD">PD</mat-option>
                <mat-option value="OTRO">OTRO</mat-option>

            </mat-select>
            <mat-error *ngIf="checkField('expedido')">
               {{getErrorMessage('expedido')}}
            </mat-error>

        </mat-form-field>

        <mat-form-field class="full-width-input">
            <input formControlName="nombre" matInput placeholder="Nombres" required/>
            <mat-error *ngIf="checkField('nombre')">
                {{getErrorMessage('nombre')}}
            </mat-error>
        </mat-form-field>
```

Nota. Se muestra la implementación del código fuente de la ventana nuevo usuario.

Figura 3.29

Nueva categoría



Nota. Se muestra la ventana para agregar nueva categoría.

Figura 3.30

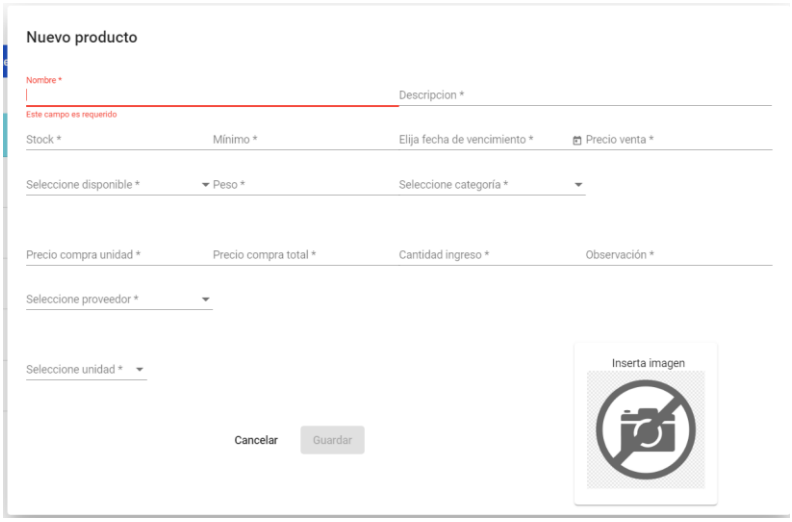
Código fuente nueva categoría

```
<h2 mat-dialog-title>{{data?.title}}</h2>
<mat-dialog-content>
    <div class="modal-form">
        <form [formGroup]="categoriaForm" >

            <mat-form-field class="full-width-input">
                <input formControlName="nombre" matInput placeholder="Nombres" id="nombrecate" oninput="this.valu
                <mat-error *ngIf="checkField('nombre')">
                    {{getErrorMessage('nombre')}}
                </mat-error>
            </mat-form-field>

        </form>
    </div>
    <div class="modal-form">
        <form [formGroup]="imgCategoriaForm">
            <div class="container-input-file">
```

Nota. Se muestra el código de la ventana agregar categoría.

Figura 3.31

Nuevo producto



Nota. Se muestra el formulario para agregar nuevo producto.

Figura 3.32

Código fuente nuevo producto

```
<h2 mat-dialog-title>{{data?.title}}</h2>
<mat-dialog-content>
    <div class="modal-form">
        <form [formGroup]="productoForm" >

            <mat-form-field class="full-width-input-medio">
                <input formControlName="nombre" matInput placeholder="Nombre"   required/>
                <mat-error *ngIf="checkFieldProducto('nombre')">
                    {{getErrorMessageProducto('nombre')}}
                </mat-error>
            </mat-form-field>

            <mat-form-field class="full-width-input-medio">
                <input formControlName="descripcion" matInput placeholder="Descripcion"   required/>
```

Nota. Se muestra el código fuente de la ventana agregar producto.

Figura 3.33

Detalle pedido

Nota. Se muestra la ventana para ver el detalle de pedidos entrantes.

Figura 3.34

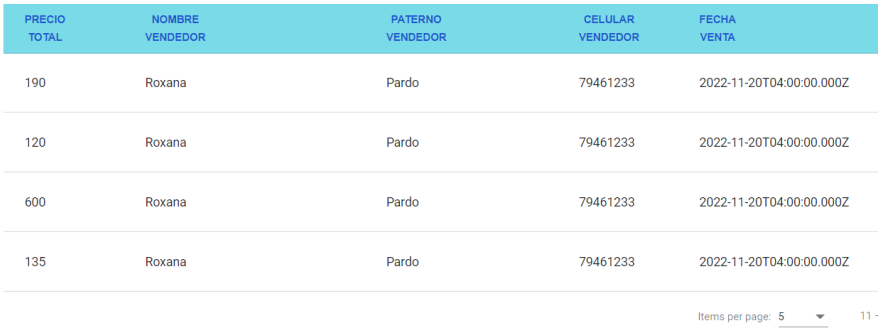
Compras

PRECIO UNIDAD	PRECIO TOTAL	CANTIDAD INGRESO	OBSERVACIÓN	FECHA	HORA	ACCIONES
8	160	20	Ninguna	2021-11-23	06:41:35	Detalles
5	100	20	Ninguno	2022-06-11	09:44:13	Detalles
28	100	40	ninguna	2022-11-16	11:05:08	Detalles
18	720	40	ninguna	2022-11-16	11:26:37	Detalles
34	680	20	Ninguna	2021-11-23	12:47:47	Detalles

Nota. Se muestra la ventana del detalle de las compras entrantes.

Figura 3.35

Ventas



PRECIO TOTAL	NOMBRE VENDEDOR	PATERNO VENDEDOR	CELULAR VENDEDOR	FECHA VENTA
190	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z
120	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z
600	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z
135	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z

Items per page: 5 11 -

Nota. Se muestra la ventana detallada de la venta realizada y por quien.

Figura 3.36

Productos más vendidos

CÓDIGO PRODUCTO	NOMBRE PRODUCTO	SUMA PRODUCTO MÁS VENDIDOS	ACCIONES
2	Sprit	14	Detalles
10	ARANJUES	12	Detalles
1	Nescafé	10	Detalles
9	Ace bolivar	5	Detalles
3	Crema de leche	4	Detalles

Nota. Se muestra la lista según orden de productos más vendidos.

Figura 3.37

Código fuente verificación fechas de vencimiento del producto

```
async getProductosVencimiento(): Promise<any> {
/* const producto: Producto[] = await this.repository.find({
 return producto;
      });*/
    const  fechahoy = new Date();
    const suma= moment(new Date(fechahoy)).add(10, 'days').format('YYYY-MM-DD');
    const produvence= await getManager()
                          .createQueryBuilder(Producto, "producto")
                          .addSelect('producto.vencimiento', 'vencimiento')

                          //.where(fechahoy.getDay-Producto.vencimiento = 30)
                          .where('producto.vencimiento <=:suma',{suma: suma})
                          .getCount()

    return produvence;          You, hace 1 segundo • Uncommitted changes
```

Nota. Se muestra el código para verificar los productos por vencer en stock.

Figura 3.38

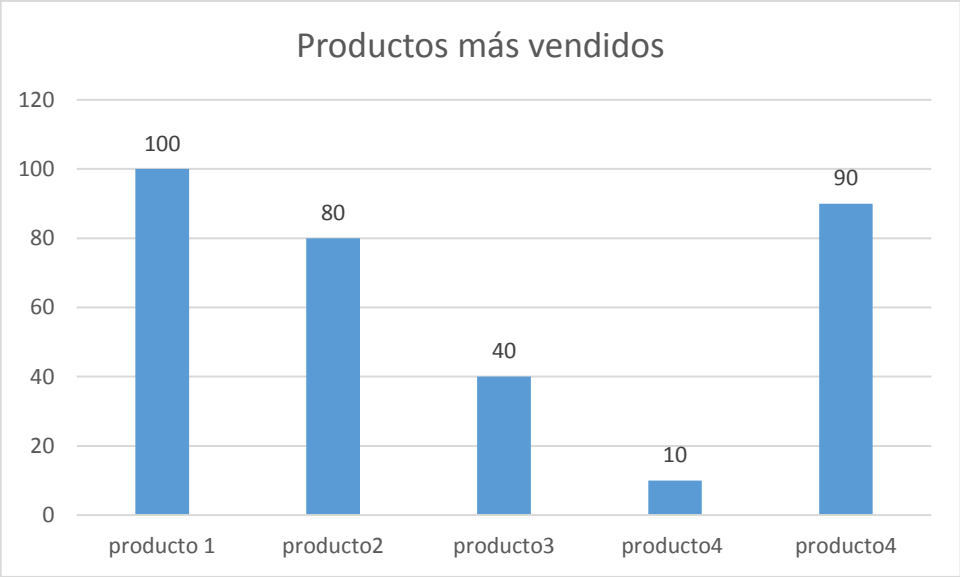
Alerta de la cantidad de productos por vencer

Nota. Se muestra la alerta con la cantidad de productos a vencer.

Reportes de productos más vendidos.

Figura 3.39

Reporte productos más vendidos



Nota. Se muestra un reporte de los productos que fueron más vendidos hasta el momento.

Reporte de productos vendidos al cliente

Figura 3.40

Reporte venta

MICROMARKET HOME SERVICE
DIRECCIÓN: Zona Zenkata

FECHA: 24-11-2022

HORA: 14:37:59

DATOS CLIENTE :

- CI: 7045127
  NOMBRE: Guillermo Apaza Flores
  CELULAR: 70454588
  DIRECCIÓN: Villa

CONSUMO

CANTIDAD PRODUCTO	PRECIO UNIDAD BS	PRECIO TOTAL BS	NOMBRE PRODUCTO
5	27	135	Ace bolivar

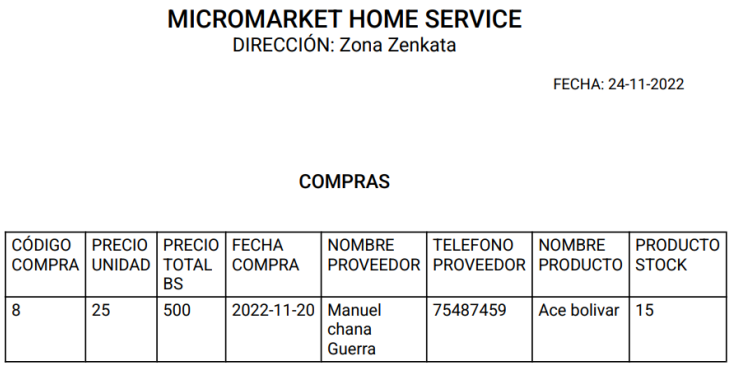
COSTO TOTAL BS.: 135

Nota. Se muestra el reporte generado de la venta realizada con cada uno de sus detalles, como el nombre, precio total.

Reporte de compras realizadas al proveedor

Figura 3.41

Reporte compras



MICROMARKET HOME SERVICE
DIRECCIÓN: Zona Zenkata

FECHA: 24-11-2022

COMPRAS

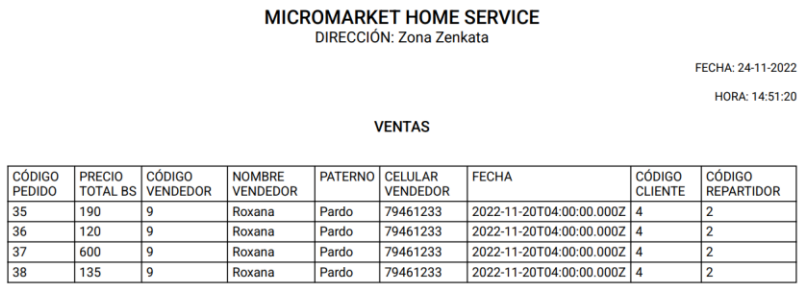
CÓDIGO COMPRA	PRECIO UNIDAD	PRECIO TOTAL BS	FECHA COMPRA	NOMBRE PROVEEDOR	TELEFONO PROVEEDOR	NOMBRE PRODUCTO	PRODUCTO STOCK
8	25	500	2022-11-20	Manuel chana Guerra	75487459	Ace bolivar	15

Nota. Se muestra el reporte de compra realizada para abastecer el stock de la micromarket.

Reporte de datos del que realizó una venta

Figura 3.42

Reporte venta



MICROMARKET HOME SERVICE
DIRECCIÓN: Zona Zenkata

FECHA: 24-11-2022

HORA: 14:51:20

VENTAS

CÓDIGO PEDIDO	PRECIO TOTAL BS	CÓDIGO VENDEDOR	NOMBRE VENDEDOR	PATERNO	CELULAR VENDEDOR	FECHA	CÓDIGO CLIENTE	CÓDIGO REPARTIDOR
35	190	9	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z	4	2
36	120	9	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z	4	2
37	600	9	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z	4	2
38	135	9	Roxana	Pardo	79461233	2022-11-20T04:00:00.000Z	4	2

Nota. Se muestra los detalles principales de la venta realizada por un vendedor.

3.4. DESARROLLO DE LA APLICACIÓN MOVIL EN MOBILE-D

3.4.1. Fase I – Exploración

Lo primordial de esta fase es la planificación y el establecimiento de la aplicación móvil, de esta manera se define el alcance general, involucrados, posteriormente se define los requerimientos de la aplicación.

3.4.1.1. Visión general de la aplicación

Se diseñó y desarrolló una aplicación móvil que pueda facilitar a los clientes en la compra de productos que son de consumo masivo, de forma rápida, desde la comodidad de su casa, y de la misma manera que los dueños puedan obtener beneficios en la expansión de la marca, como también mejores ganancias.

3.4.1.2. Establecimiento de Stakeholders

El establecimiento de Stakeholders (Partes interesadas en la aplicación), tiene como objetivo primordial y necesario identificar actores en varias tareas que se presenten.

- ✓ **Establecimiento de Usuario:** Se identificó al cliente como el principal actor, quien tiene como objetivo realizar de una forma intuitiva las compras online desde su hogar. El mismo tiene acceso a la aplicación como una herramienta tecnológica para disminuir tiempo, clima, trancaderas, incomodidad.
- ✓ **Establecimiento de Administrador:** Se identificó como un autor de importancia, quien es el encargado de administrar las altas y bajas de los productos, como también otorga permisos al vendedor sobre a qué funcionalidades puede tener acceso.
- ✓ **Establecimiento de Vendedor:** Es el encargado de recibir o ver la notificación del pedido, como también confirmar el envío del mismo.
- ✓ **Establecimiento de Grupo de Desarrollo:** El diseño y desarrollo de la aplicación móvil, fue a cargo por el autor del proyecto de grado.

3.4.1.3. Definición del alcance del proyecto

Una vez que el usuario o cliente envía un pedido, el web service recibe la información de todos los productos seleccionados, de esta manera insertar en la base de datos, una vez realizado la operación envía una notificación de un nuevo pedido al encargado de ventas, donde el mismo se encarga de coordinar la venta.

a) Análisis de la situación actual

La micromarket se lo encontró con un sistema de ventas manual, en un entorno donde se ubicaba de forma físico, las ganancias estaban de cierta manera obstaculizadas por el desconocimiento de la marca, ahora con la aplicación móvil se tiene más posibilidad de ventas a mayor escala y realizar pedidos de forma cómoda desde su hogar, también permite la expansión de la marca y mayor ganancia económico.

b) Análisis de requerimientos

i. Requerimientos funcionales

Los requerimientos funcionales son de vital importancia ya que con ellos se define el comportamiento del sistema.

Tabla 3.13

Requerimientos funcionales.

N°	Descripción	Evidente / Oculto
R1	La aplicación tiene menú categoría.	Evidente
R2	La aplicación tiene menú productos.	Evidente
R3	El usuario puede ver la lista de pedidos.	Evidente
R4	Permite al usuario ver la lista de productos deseados.	Evidente
R5	La aplicación captura la ubicación del cliente para la entrega.	Oculto
R6	La aplicación necesita autenticación de usuario para enviar pedido.	Oculto

Nota. Se muestra los requerimientos funcionales.

ii. Requerimientos no funcionales

Los requerimientos no funcionales permiten ver los aspectos visibles para el usuario.

Tabla 3.14

Requerimientos no funcionales.

N°	Descripción	Evidente / Oculto
R1	Requiere un dispositivo móvil con sistema operativo Android de la versión 4.4.2 hasta la versión actual.	Evidente
R2	La aplicación se instala directamente en el dispositivo móvil.	Evidente
R3	Los datos de ingreso son verificados y validados.	Oculto
R4	Interfaz de usuario amigable de fácil manejo.	Evidente
R5	La aplicación está en constante mejora.	Evidente

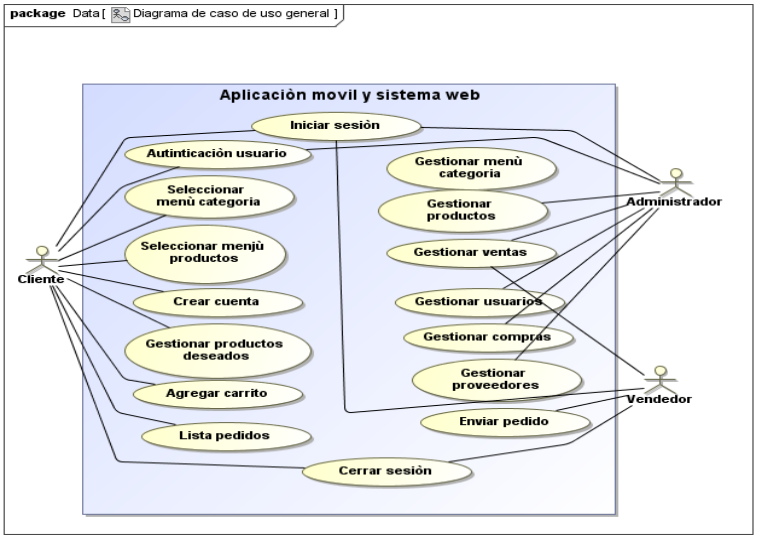
Nota. Se muestra los requerimientos no funcionales.

3.4.1.4. Diagramas de Casos de Uso

a) Diagrama de caso de uso general de la aplicación

Figura 3.43

Caso de Uso General de la aplicación

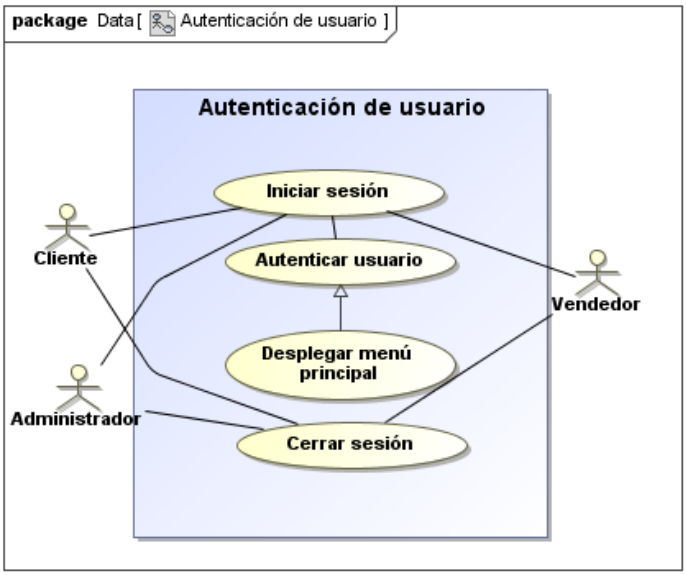


Nota. Se muestra el caso de uso general de la aplicación móvil para la venta online de consumo masivo.

b) Diagrama de caso de uso autentificación de usuario

Figura 3.44

Caso de Uso autentificación de usuarios



Nota. Se muestra el caso de uso para iniciar sesión de parte del cliente.

Tabla 3.15

Caso de Uso autentificación de usuarios.

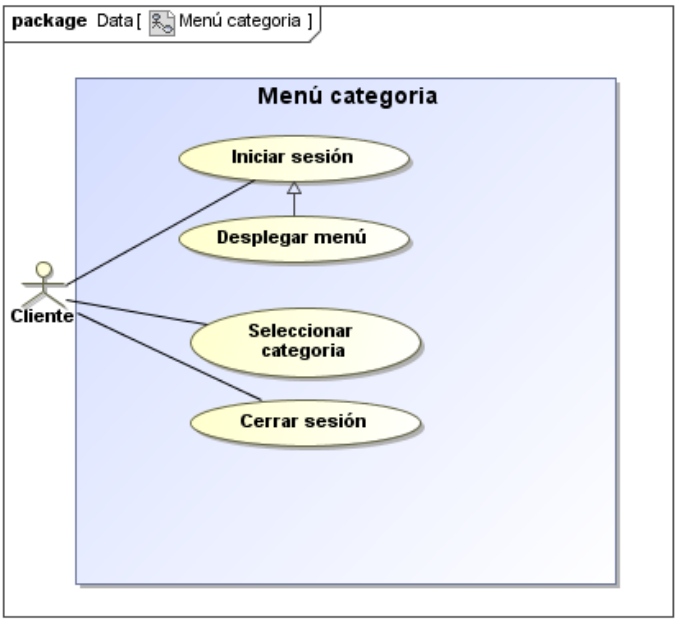
Caso de uso	**Autentificación de Usuarios**	
Actor	Administrador, Cliente, Vendedor	
Descripción	El caso de uso comienza cuando el administrador, cliente, vendedor, quiere ingresar al sistema con su cuenta de usuario.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El administrador, cliente, vendedor, quiere iniciar sesión en la aplicación. 2.-El usuario entra al menú para login. 4.-El usuario introduce los datos	3.-El sistema lanza un formulario de login para que ahí el usuario ingrese los datos requeridos. 5.-El sistema valida los datos introducidos, sino son válidos vuelve a requerirlos. 6.-El sistema verifica los datos ingresados con lo registrado en la base de datos. 7.-El sistema devuelve la interfaz.
Precondición	El usuario debe tener una cuenta registrada.	
Post condición	Los datos registrados deben ser almacenados en la base de datos.	

Nota. Se muestra la ventana de autentificación a la aplicación.

c) Diagrama de caso de uso menú categoría

Figura 3.45

Caso de Uso menú categoría



Nota. Se muestra el caso de uso menú categoría.

Tabla 3.16

Caso de Uso menú categoría.

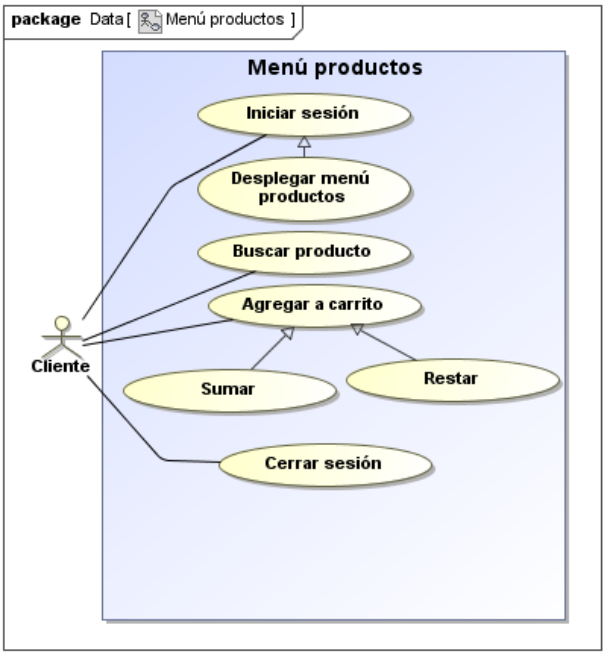
Caso de uso	**Menú Categoría**	
Actor	Cliente	
Descripción	El caso de uso comienza cuando el cliente quiere ver las categorías existentes y acceder a ellas.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente quiere ingresar al menú principal de categorías.	2.-El sistema de devuelve las categorías disponibles.
	3.-El cliente selecciona una categoría de producto.	4.-El sistema devuelve productos de la categoría seleccionada.
Precondición	El cliente debe ingresar al sistema con su cuenta de usuario.	
Post condición	El cliente debe poder elegir la categoría e ingresar a ella.	

Nota. Se muestra el menú categoría para el cliente.

d) Diagrama de caso de uso menú productos

Figura 3.46

Caso de Uso menú productos



Nota. Se muestra el caso de uso menú productos.

Tabla 3.17

Caso de Uso menú productos.

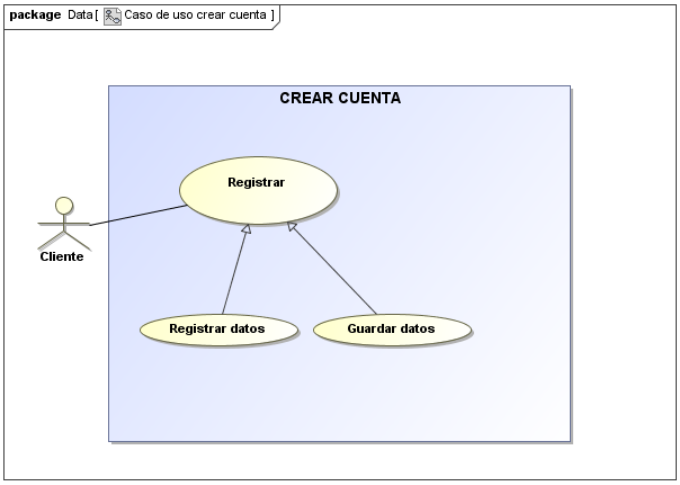
Caso de uso	**Menú Productos**	
Actor	Cliente	
Descripción		
	Evento Actor	Evento Sistema
Flujo Principal	1.-El cliente ingresa al menú productos.	3.-El sistema devuelve el producto buscado.
	2.-El cliente busca un producto.	5.-El sistema devuelve un ventana modal para aumentar o disminuir la cantidad del producto.
	4.- El cliente selecciona agregar.	
Precondición	El cliente debe haber ingresado a una categoría.	
Post condición	Los productos seleccionados deben agregarse al carrito.	

Nota. Se muestra el menú productos para agregar al carrito.

e) Diagrama de caso de uso crear cuenta

Figura 3.47

Caso de Uso crear cuenta



Nota. Se muestra el caso de uso crear cuenta.

Tabla 3.18

Caso de Uso crear cuenta.

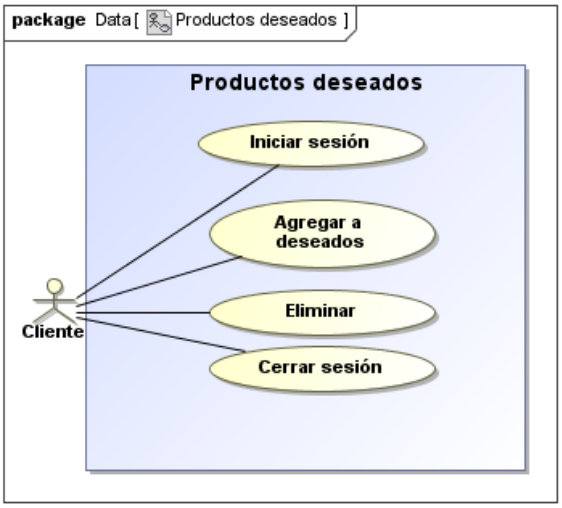
Caso de uso	**Crear Cuenta**	
Actor	Cliente	
Descripción	El caso de uso comienza cuando el cliente debe registrar sus datos.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente quiere ingresar con su cuenta o quiere enviar su pedido.	2.-El sistema devuelve el formulario de registro de datos.
		4.-El sistema valida los datos.
	3.-El cliente ingresa los datos.	5.-El sistema guarda los datos.
Precondición	El cliente debe introducir datos correctos para crear su cuenta.	
Post condición	Los datos introducidos deben guardarse en la base de datos.	

Nota. Se muestra caso de uso crear cuenta por parte del cliente.

f) Diagrama de caso de uso productos deseados

Figura 3.48

Caso de Uso productos deseados



Nota. Se muestra el caso de uso de productos deseados por el cliente.

Tabla 3.19

Caso de Uso productos deseados.

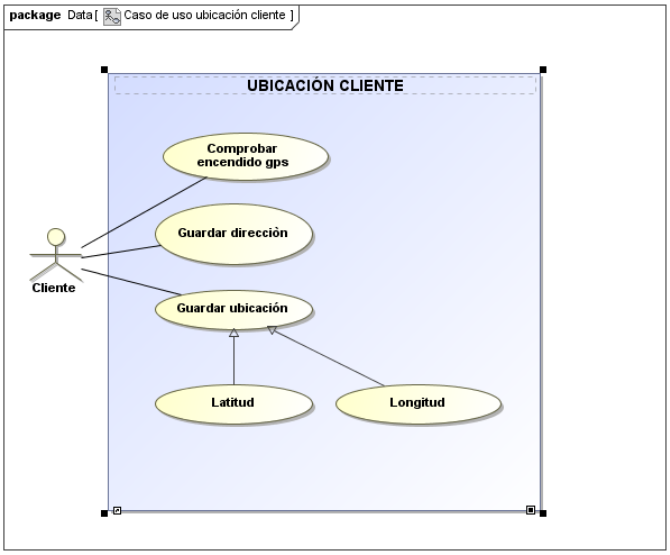
Caso de uso	**Productos Deseados**	
Actor	Cliente, Administrador	
Descripción	El caso de uso comienza cuando el cliente selecciona los productos deseados.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente al buscar productos selecciona algunos como productos deseados.	2.-El sistema guarda los productos seleccionados.
	3.-El cliente, administrador quieren ver la lista de productos deseados.	4.-El sistema devuelve la lista de productos deseados.
Precondición	El cliente debe ingresar a una categoría de productos.	
Post condición	Los productos seleccionados deben mostrarse en la pestaña productos deseados.	

Nota. Se muestra el caso de uso para seleccionar los productos deseados.

g) Diagrama de caso de uso ubicación cliente

Figura 3.49

Caso de Uso ubicación cliente



Nota. Se muestra el caso de uso de la ubicación o dirección del cliente.

Tabla 3.20

Caso de Uso ubicación cliente.

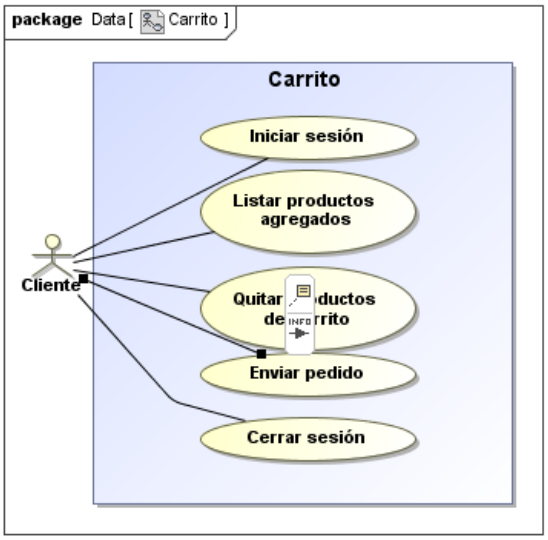
Caso de uso	**Ubicación Cliente**	
Actor	Cliente	
Descripción	El caso de uso comienza cuando el cliente debe brindar su ubicación exacta para que le envíen su pedido.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente quiere ingresar su dirección.	2.-El sistema devuelve un formulario.
	3.-El cliente introduce los datos y activa GPS.	4.-El sistema valida los datos.
		5.-El sistema guarda los datos.
Precondición	El cliente debe activar su GPS de su móvil.	
Post condición	La ubicación debe guardarse en la base de datos.	

Nota. Se muestra el caso de uso para guardar la dirección del cliente.

h) Diagrama de caso de uso carrito

Figura 3.50

Caso de Uso carrito



Nota. Se muestra el caso de uso carrito de productos agregados.

Tabla 3.21

Caso de Uso carrito.

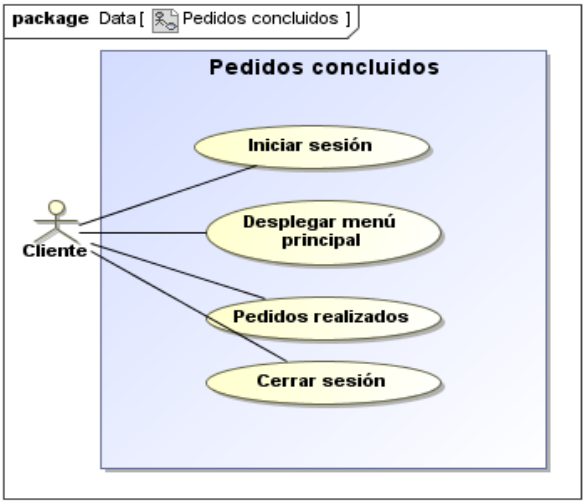
Caso de uso	**Carrito**	
Actor	Cliente	
Descripción	El caso de uso comienza cuando el cliente quiere verificar la lista de productos agregados al carrito.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente ingresa al carrito.	2.-El sistema devuelve lista de pedidos, suma del precio total.
	3.-El cliente quiere aumentar o disminuir la cantidad de pedido de un producto.	4.-El sistema aumenta o disminuí la cantidad de producto.
	5.-El cliente quiere enviar el pedido.	6.-El sistema guarda la lista de pedidos en la base de datos.
Precondición	El cliente debe tener productos agregados.	
Post condición	La lista enviada debe guardarse en la base de datos.	

Nota. Se muestra el caso de uso para agregar al carrito un producto.

i) Diagrama de caso de uso lista de pedidos concluidos

Figura 3.51

Caso de Uso pedidos concluidos o realizados



Nota. Se muestra el caso de uso de pedidos realizados.

Tabla 3.22

Caso de Uso pedidos concluidos.

Caso de uso	**Pedidos Concluidos**	
Actor	Cliente	
Descripción	El caso de uso comienza cuando el cliente quiere ver su historial de pedidos.	
Flujo Principal	Evento Actor	Evento Sistema
	1.-El cliente quiere ver su historial de pedidos. 2.-El cliente ingresa al menú principal y selecciona pedidos.	3.-El sistema devuelve historial de pedidos concluidos.
Precondición	El cliente debe tener sesión iniciada.	
Post condición	Se debe mostrar la lista de pedidos satisfactoriamente completados.	

Nota. Se muestra el caso de uso de pedidos realizados o concluidos.

3.4.2. Fase II – Inicialización

En esta fase se identifica y se determina los recursos físicos, técnicos y las herramientas necesarias para el desarrollo de la aplicación, logrando así la configuración general del proyecto.

3.4.2.1. Herramientas para el desarrollo de la aplicación

Android Studio, como entorno de desarrollo integrado oficial para la plataforma Android.

SqlLite, como gestor de base de datos del sistema interno.

Retrofit, para transmisión de datos por servicios rest.

3.4.2.2. Cronograma del proyecto

Según la metodología se estableció un cronograma con fechas de entrega para las distintas fases. Siendo muy importante para así contralar los tiempos de entrega en las fechas establecidas. De esta manera se muestra el cronograma de fechas en la tabla siguiente:

Tabla 3.23

Cronograma del proyecto

Actividades		Exploración	Inicialización	Producción	Estabilización	Pruebas
Julio	1					
	2					
	3					
	4					
Agosto	1					
	2					
	3					
	4					
Septiembre	1					
	2					
	3					
	4					
Octubre	1					
	2					
	3					
	4					
Noviembre	1					
	2					
	3					
	4					

Nota. Se muestra el cronograma para realizar la aplicación.

3.4.2.3. Requerimiento del producto

De acuerdo a la estructura propuesta se tiene las tareas necesarias para desarrollar la aplicación, que se detalla en la parte de producción. Los requerimientos planteados para el desarrollo de la aplicación son descritos en la siguiente tabla:

Tabla 3.24

Planificación para el desarrollo de la aplicación.

Interacción	Tareas
Aplicación movil	Diseño de la base de datos.
	Implementar de la base de datos local para la app.
	Construir clase interfaz para los endpoint.
	Crear model para distintos funciones.
	Desarrollar maquetado de interfaz para seleccionar productos
	Visualizar historial de pedidos.
	Visualizar productos favoritos.
	Visualizar productos en oferta.
	Desarrollar agregar producto al carrito.
	Desarrollar módulo envío del pedido.

Nota. Se muestra la planificación para el desarrollo de la aplicación.

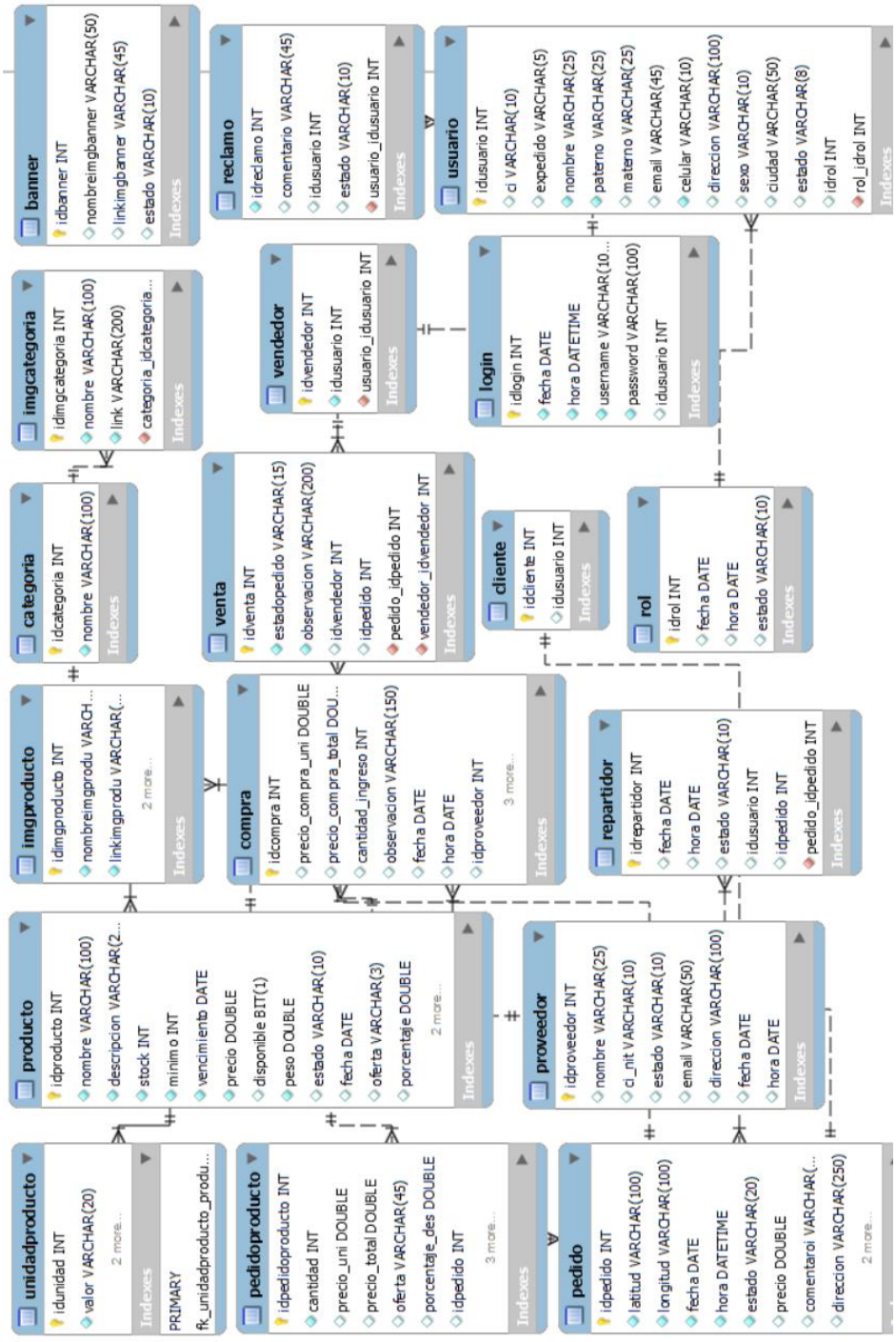
3.4.3. Fase III - Producción

En esta fase se describe las tareas planteadas anteriormente para el desarrollo de la aplicación, de la misma manera también implementar las funcionalidades de la aplicación para que tenga el desempeño requerido y esperado por parte de los actores involucrados y como también para el buen desempeño, como la facilidad de aprendizaje en la manipulación de cada uno de las funciones.

3.4.3.1. Diagrama relacional de la base de datos

Figura 3.52

Diagrama relacional de la base de datos



Nota. Se muestra las tablas del sistema.

3.4.3.2 Implementación de módulos del sistema

a) Maquetación

i. Módulo de inicio de sesión

Figura 3.53

Vista inicio de sesión de la aplicación

Nota. Se muestra la ventana de inicio de sesión.

Figura 3.54

Código fuente inicio de sesión

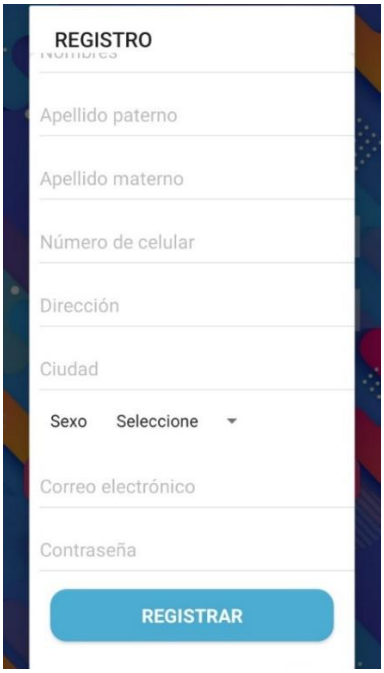
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="120dp"
        android:layout_height="150dp"
        android:layout_marginLeft="130dp"
        android:layout_alignParentTop="true"

        android:layout_marginTop="26dp"
        app:srcCompat="@drawable/homeservice" />
```

Nota. Se muestra el código fuente de inicio de sesión.

ii. Módulo de registro del usuario

Figura 3.55

Formulario de registro del cliente



Nota. Se muestra el formulario de registro para el cliente.

Figura 3.56

Código fuente de formulario registro del cliente

```
<com.rengwuxian.materialedittext.MaterialEditText
    android:id="@+id/edtCedulaUser"
    android:hint="Cédula de Identidad"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:inputType="number"
    android:layout_marginLeft="10dp"
    app:met_floatingLabel="highlight"/>
```

Nota. Se muestra código fuente de formulario de registro para el cliente.

iii. Módulo de listado de categorías

Figura 3.57

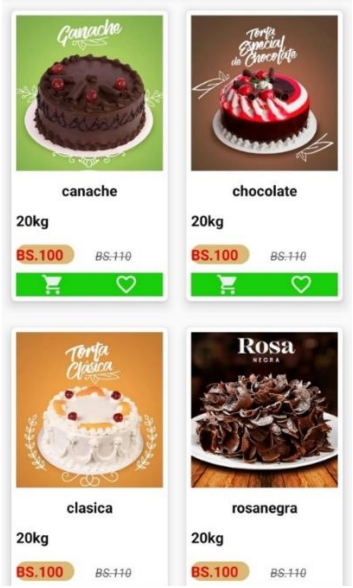
Vista de listado de categorías

Nota. Se muestra la vista de todas las categorías disponibles.

iv. Módulo de listado de productos

Figura 3.58

Vista de listado de productos



Nota. Se puede observar el módulo de listado de todos los productos disponibles dentro de una categoría.

Figura 3.59

Código fuente de listado de productos

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:cardview="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/cardview_productos_id"
    android:clickable="true"
    android:focusable="true"
    android:foreground="?android:attr/selectableItemBackground"
    android:layout_width="match_parent"
    android:layout_height="330dp"
    android:layout_margin="5dp"
    cardview:contentPadding="5dp"
    cardview:cardElevation="8dp"
```

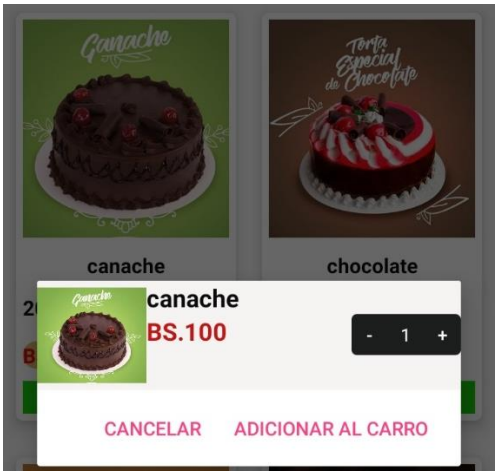
Nota. Se puede observar el código fuente del listado de productos.

v. Módulo de agregar al carrito

Como se puede observar en la figura 3.60, lo cual corresponde al módulo de agregar al carro los productos que desea adquirir el cliente.

Figura 3.60

Vista agregar al carrito



Nota. Se muestra la ventana para agregar un producto al carrito.

Figura 3.61

Código fuente de agregar al carrito

```
    android:textStyle="bold"
    android:gravity="center"
    />

<com.cepheuen.elegantnumberbutton.view.ElegantNumberButton
    android:id="@+id/txt_count"
    android:layout_width="80dp"
    android:layout_height="30dp"
    app:initialNumber="1"
    app:finalNumber="100"

    android:layout_alignParentRight="true"
    app:backGroundColor="#1c1d1d"/>
```

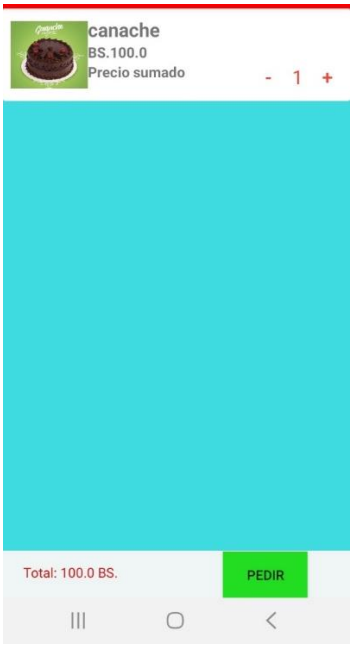
Nota. Se muestra el código fuente para agregar un producto al carrito.

vi. Módulo de productos agregados al carrito

A continuación, en la figura 3.62, se puede observar los productos agregados y la cantidad de cada uno de ellos, como también la suma total del costo de los productos.

Figura 3.62

Vista de productos agregados al carrito



Nota. Se muestra los productos agregados al carrito.

Figura 3.63

Código fuente de productos agregados al carrito

```
<ImageView
    android:id="@+id/delete_icon"
    android:src="@drawable/ic_delete_black_24dp"
    android:layout_marginRight="10dp"
    android:layout_alignParentRight="true"
    android:layout_centerVertical="true"
    android:layout_width="30dp"
    android:layout_height="30dp" />

<TextView
    android:layout_toLeftOf="@+id/delete_icon"
    android:text="DELETE"
```

Nota. Se muestra el código fuente de productos agregados al carrito.

3.4.4. Fase IV - Estabilización

En esta fase es donde se verifican los últimos detalles de integración de todos los partes involucrados para su correcto funcionamiento.

Para la verificación del funcionamiento de la aplicación móvil, se tuvo que unir las partes de cada módulo que tiene una función específica, para que de esta manera se puede ver el correcto funcionamiento.

En la figura 3.64, se puede observar las partes involucradas y como también los módulos que intervienen en la aplicación móvil.

Figura 3.64

Vista de herramienta de desarrollo Android studio

Nota. Se muestra la interfaz de desarrollo Android studio.

3.5. IMPLEMENTACION Y DESPLIEGUE

3.5.1. Requerimientos de hardware

- ✓ Servidor alguilado en Digital Ocean, para almacenar el sistema backend y frontend.
- ✓ Una computadora o laptop para coordinar los pedidos que van llegando.

3.5.2. Requerimientos de software

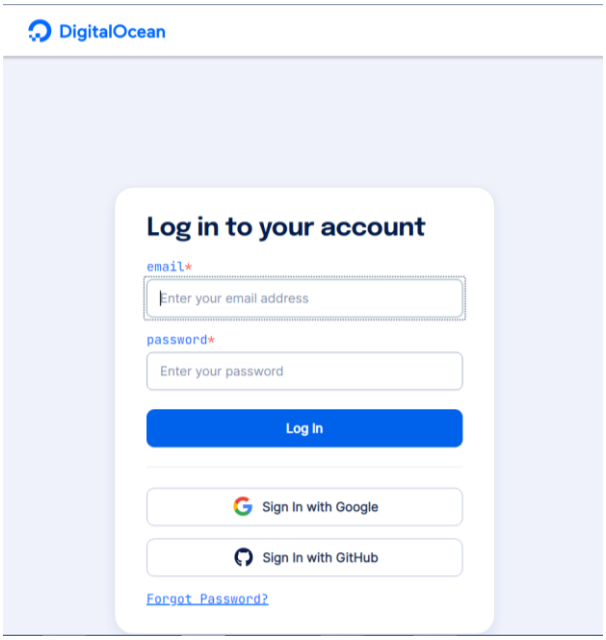
- ✓ Nodejs
- ✓ Framework NestJS
- ✓ Nodemon
- ✓ PostgreSQL 13
- ✓ Nginx

3.5.3. Despliegue

1. Abrir una cuenta en digital ocean

Figura 3.65

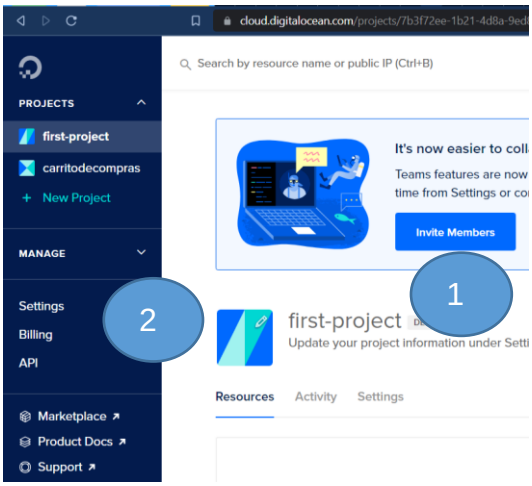
Ventana para crear cuenta de digitalOcean



Nota. Se muestra la interfaz para crear cuenta en digitalOcean.

Figura 3.66

Ventana para crear proyecto

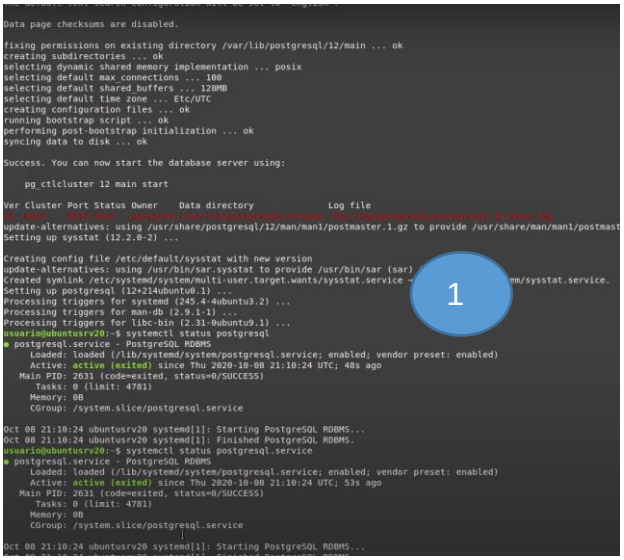


Nota. 1. Entrar a la página de digital ocean, registrarse con tu correo, poner una cuenta de banco. 2. Una vez dentro crear nuevo proyecto, crear un droplet para instalar lo necesario para tu backend y frontend.

2. Instalar postgreSQL

Figura 3.67

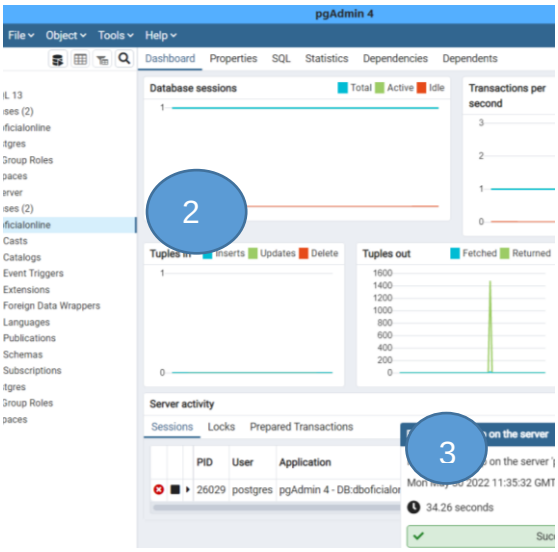
Instalación de postgresql 13



Nota. 1. Seguir la guía de instalación en la página oficial de postgresql para no tener errores, muchas veces si cometes un error da problemas.

Figura 3.68

Backup de postgresql



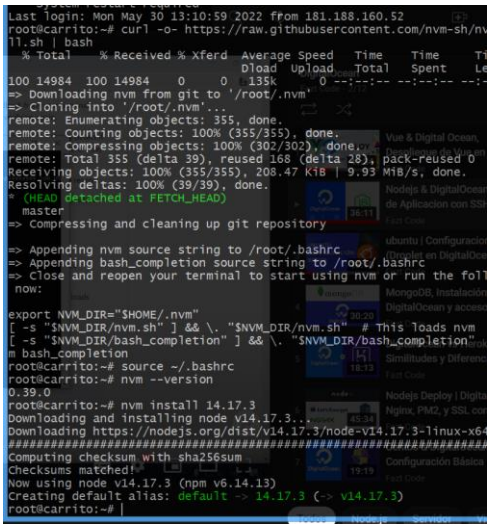
Nota. 2. Para conectarse de tu máquina local de esta manera subir la base de datos al sevidor crear un server nuevo y poner la ip de digital ocean, como también la contraseña de la base de datos.

3. Pulsar sobre tu base de datos con click derecho donde muestra la opción de backup, buscar un lugar, poner el nombre el backup, ir al nuevo servidor crear nueva base de datos y restaurar el backup.

3. Instalar Node.js en el servidor

Figura 3.69

Instalación de Nodejs

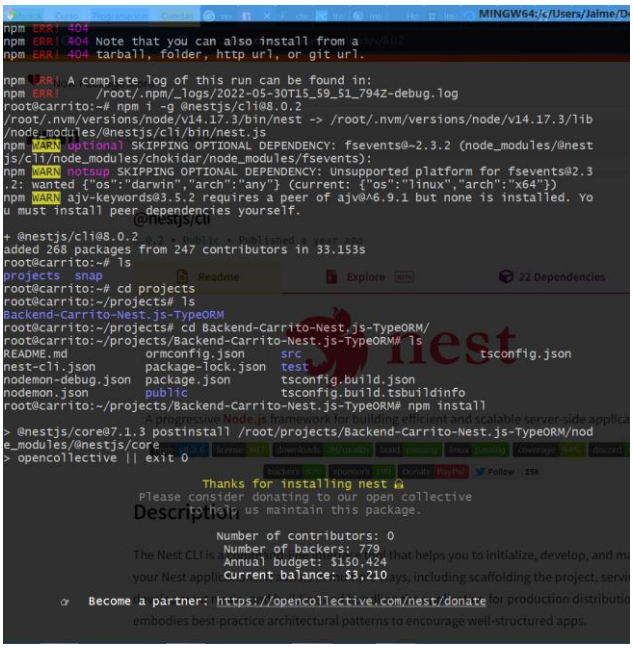


Nota. Instalación de nodejs para el funcionamiento de nestjs.

4. Desplegar proyecto Nest.js

Figura 3.70

Desplegando backend con Nestjs



Nota. Desplegando backend con framework NestJS

5. Instalar Nginx para desplegar frontend

Figura 3.71

Instalación Nginx

```
$ sudo apt update
$ sudo apt install nginx
```

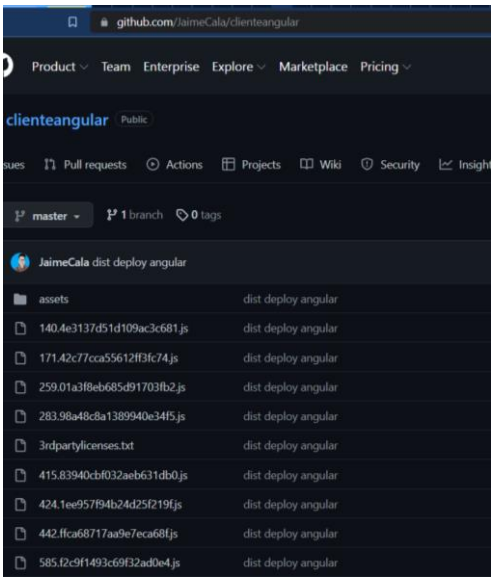
Copy

Nota. Instalación de nginx para desplegar frontend en angular

6. Subir a Github para desplegar web

Figura 3.72

Subir a github frontend en angular

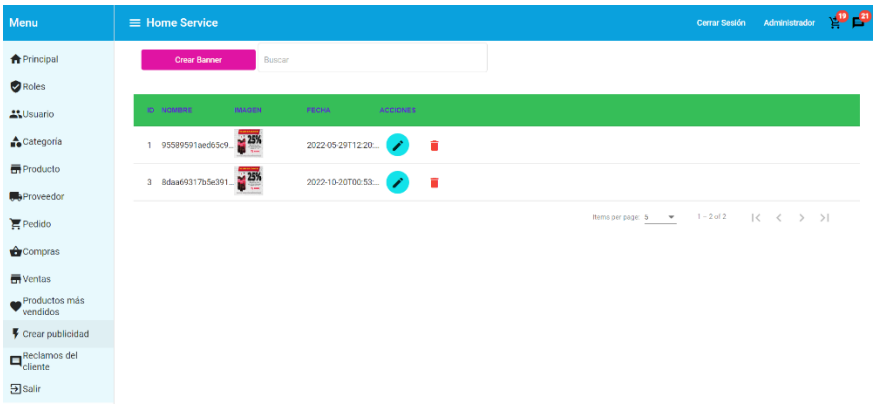


Nota. Subir a github para desplegar frontend en angular

7. Pantalla principal

Figura 3.72

Resultado despliegue de sistema web



Nota. Funcionamiento de sistema web administrador.

CAPÍTULO IV

PRUEBAS

Y

RESULTADOS

4.1 INTRODUCCIÓN

En este capítulo se aplicará las normas iso para las métricas de calidad, seguridad, y método COSMIC para la estimación de costos de desarrollo de software.

4.2 MÉTRICAS DE CALIDAD NORMA ISO/IEC 25010

4.2.1 Adecuación funcional

Es la capacidad del software de cumplir y proveer las funciones para satisfacer las necesidades explicitas e implícitas cuando es utilizado en condiciones específicas. Para poder determinar la adecuación funcional del sistema se debe determinar cinco características de dominio de información.

1. Número de entradas de usuario. - Se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación. Las entradas deben ser restringidas de las peticiones que se contabilizan por separado.

Tabla 4.1

Número de entradas de usuario.

	Entradas del usuario
1	Ingreso al sistema
2	Registro de usuarios
3	Registro de productos
4	Registro de proveedores
5	Registro de categoría
6	Registro de pedido
7	Registro de compras
8	Registro de ventas
9	Registro de vendedor
10	Registro de rol
11	Registro de repartidor
Total	11

Nota. Se muestra el número de entradas de usuario.

2. Número de salidas de usuario. - La salida se refiere a informes, pantallas, mensajes de error.

Tabla 4.2

Número de salidas de usuario.

	Salidas del usuario
1	Listado de usuarios
2	Listado de categorías
3	Listado de productos
4	Listado de pedidos
5	Listado de ventas
6	Listado de compras
7	Listado de proveedor
8	Listado de rol
9	Listado de vendedores
10	Listado de repartidores
Total	10

Nota. Se muestra el número de salidas de usuario.

3. Número de peticiones de usuario. - Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva.

Tabla 4.3

Número de peticiones de usuario.

	Peticiones del usuario
1	Listado de usuarios
2	Listado de categorías
3	Listado de productos
4	Listado de pedidos
5	Listado de ventas
6	Listado de compras
7	Listado de proveedor
8	Listado de rol
9	Listado de vendedores
10	Listado de repartidores
11	Autenticación de usuario
Total	11

Nota. Se muestra el número de salidas de usuario.

4. Número de archivos. - Se cuenta cada archivo maestro lógico. En otras palabras, las tablas existentes en la base de datos.

Tabla 4.4

Número de archivos.

	Número de archivos	
1	Todos los archivos	16
Total		16

Nota. Se muestra el número archivos.

5. Número de interfaces externas. - Se cuenta todas las interfaces legibles por el ordenador que son utilizados para transmitir la información.

Tabla 4.5

Número de interfaces externos.

	Número de interfaces externos	
1	Internet	1
Total		1

Nota. Se muestra el número de interfaces externos.

Para calcular los puntos función no ajustado se tiene:

Tabla 4.6

Cálculo de punto función no ajustado.

Parámetros de medición	**Factor de ponderación**				
	CUENTA	SIMPLE	MEDIO	COMPLEJO	TOTAL
N° de entradas de usuario.	11	3	**4**	6	44
N° de salidas de usuario.	10	4	**5**	7	50
N° de peticiones de usuario.	11	3	**4**	6	44
N° de archivos de operación.	16	7	**10**	15	160
N° de interfaces externas.	1	5	**7**	10	7
				Total	305

Nota. Se muestra el cálculo de punto función no ajustado.

El punto función se puede calcular mediante la siguiente ecuación.

$$PF = Cuenta\ total * (0{,}65 + 0{,}01 * \sum F_i)$$

Donde:

Cuenta total: es la suma de todas las entradas obtenidas en N° de entradas, N° de salidas, N° peticiones, N° de archivos y N° de interfaces externas.

x: Nivel de confiabilidad del sistema es de (0.65).

y: Nivel de error igual a 0.01

F_i: Son los valores de ajuste de la complejidad, según respuestas de la tabla valores de ajuste de complejidad, bajo la siguiente ponderación.

Tabla 4.7

Valores de puntos función.

Valor	Factor
0	Sin influencia.
1	Mínima.
2	Moderada.
3	Promedio.
4	Significativa.
5	Fuerte.

Nota. Se muestra los valores de punto función.

Tabla 4.8

Valores de ajuste de complejidad.

	Factor de ajuste de valor	Valor
1	¿Requiere el sistema copias de seguridad y de recuperación fiables?	4
2	¿Se requiere comunicación de datos?	4
3	¿Existen funciones de procesamiento distribuido?	3
4	¿Es crítico el rendimiento?	4
5	¿Se ejecutaría el sistema en un entorno operativo existente y fuertemente utilizado?	4
6	¿Requiere el sistema entrada de datos interactiva?	4

7	¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	4
8	¿Se actualizan los archivos maestros de forma interactiva?	4
9	¿Son complejas las entradas, las salidas, los archivos o las peticiones?	4
10	¿Es complejo el procesamiento interno?	4
11	¿Se ha diseñado el código para ser reutilizable?	4
12	¿Están incluidas en el diseño la conversión y la instalación?	4
13	¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?	4
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?	4
	Factor de complejidad total	**55**

Nota. Se muestra los valores de ajuste de complejidad.

Empleando la fórmula para hallar el PF y PF(Máximo):

$$PF = Cuenta\ total * (0{,}65 + 0{,}01 * \sum F_i)$$
$$PF = 335 * (0{,}65 + 0{,}01 * 55)$$
$$PF = 366$$
$$PF(Máximo) = 335 * (0{,}65 + 0{,}01 * 70)$$
$$PF(Máximo) = 411{,}8$$

El indicador de funcionalidad se lo encuentra calculando la siguiente relación.

$$Funcionalidad = \frac{\text{PF calculado}}{\text{PF maxima}} * 100\%$$

$$Funcionalidad = \frac{366}{411{,}8}$$

$$Funcionalidad = 0{,}888 * 100\% = 88{,}8\%$$

Por tanto, la funcionalidad del sistema es un 88,89% lo que quiere decir que el sistema tiene un 88,9% de funcionar sin fallo, y un 11,1% de sufrir un fallo.

4.2.2. Fiabilidad

La fiabilidad es la capacidad del software para asegurar un nivel de funcionamiento adecuado cuando es utilizando en condiciones específicas.

En este caso a la fiabilidad se amplía sostener un nivel especificado de funcionamiento y no una función requerida.

Para poder medir la fiabilidad del sistema, se toma la siguiente fórmula que calcula la confiabilidad del sistema.

$$F(t) = f * e^{(\frac{-\beta}{10} * t)}$$

Donde:

f: Es la funcionalidad del sistema ya calculada 88,9%.

-β: Es la probabilidad de error que puede tener un sistema 0,03(3%).

t: El tiempo que dura una gestión en el sistema 12 meses.

A continuación, se realiza el cálculo de la probabilidad de que el sistema sufra fallas:

$$F(t) = 88{,}8 * e^{(\frac{-3}{10} * 5)}$$

$$F(t) = \ 88{,}8 * 0{,}2231$$

$$F(t) = 19\%$$

La probabilidad de que el sistema sufra fallo es de 19%, lo que quiere decir que la probabilidad de que el sistema esté libre de fallo es:

$$P\ (T \geq t) = 1 - F(t)$$

$$P\ (T \geq t) = (1 - 0{,}01963) * 100\%$$

$$P\ (T \geq t) = 80{,}36\%$$

Por tanto, se deduce que el sistema tiene un grado de confiabilidad de: $P\ (5) = 80{,}36\%$

4.2.3. Usabilidad

La usabilidad es la capacidad del software de ser entendido, aprendido, y usado en forma fácil y atractiva. La usabilidad está determinada por los usuarios finales y los usuarios indirectos del software, dirigidos a todos los ambientes, a la preparación del uso y el resultado obtenido

Se utilizó la siguiente escala de valores para determinar la usabilidad.

Tabla 4.9

Escala de ponderación para responder a la encuesta.

Escala	Valor
Pésimo	1
Malo	2
Regular	3
Optimo	4
Muy Bueno	5

Nota. Se muestra escala de ponderación para responder la encuesta.

Tabla 4.10

Cuestionario de evaluación de uso.

N°	Pregunta	Valor
1	¿Son complicadas las respuestas de la aplicación?	4
2	¿Le resulta fácil de recordar las órdenes y aprender las operaciones básicas?	3
3	¿Considera usted que es una herramienta útil?	4
4	¿Los resultados que le proporciona la aplicación le facilitan su trabajo?	4
5	¿Cómo le parece el tiempo de ejecución de tareas?	4
6	¿Utiliza la aplicación con facilidad?	4
	Total	**23**

Nota. Se muestra el cuestionario de evaluación de uso.

Para realizar el cálculo de usabilidad del sistema, aplicamos la siguiente ecuación:

$$FU = \frac{\left(\frac{\sum x_i}{n}\right) * 100}{\mathrm{N}}$$

Donde:

N: número de población.

n: número en la muestra.

$$FU = \frac{\left(\frac{23}{6}\right) * 100}{5}$$

$$FU = 76{,}6\%$$

Por lo tanto, la usabilidad del sistema es un 76,6%, que manifiesta la facilidad de su uso por parte del usuario.

4.2.4. Mantenibilidad

La capacidad de mantenimiento es la cualidad que tiene el software para ser modificado. Incluyendo correcciones o mejoras del software, a cambios en el entorno, y especificaciones de requerimientos funcionales.

Para determinar la estabilidad de un producto se tiene la siguiente ecuación.

$$IMS = \frac{[\mathrm{Mt} - (\mathrm{Fa} + \mathrm{Fc} + \mathrm{Fd})]}{\mathrm{Mt}}$$

Tabla 4.11

Facilidad de mantenimiento.

Tipo	Descripción	Valor
Mt	Número de módulos de la versión actual	6
Fc	Número de módulos en la versión actual que se han cambiado.	0
Fa	Número de módulos en la versión actual que se han añadido.	1
Fd	Número de módulos de la aversión anterior que se han borrado en la versión actual.	0

Nota. Se muestra que tan fácil es mantener el sistema.

$$IMS = \frac{[6 - (1 + 0 + 0)]}{6}$$

$$IMS = 0{,}83 * 100\%$$

$$IMS = 83\%$$

Por tanto, la mantenibilidad del sistema es de un 83%, lo que quiere decir que el margen de conflictividad de cambios futuros en el software no es alto.

4.2.5. Portabilidad

La capacidad que tiene el software para ser trasladado de un entorno a otro.

Tabla 4.12

Facilidad de portabilidad.

Factor de portabilidad	Valor%
Puede ser transferido de un entorno a otro	75
Se puede adaptar a otros ambientes con facilidad	80
Es fácil instalar	85
Es capaz de ser reemplazado por una nueva versión	90
Total	**82%**

Nota. Se muestra la facilidad de portabilidad.

La interpretación a este resultado significa que nuestra aplicación móvil y el parte web administrador tiene la capacidad de ejecutarse en otro entorno: **Portabilidad = 96%**

4.2.6. Calidad global

De acuerdo a los resultados obtenidos anteriormente se puede establecer la calidad total del sistema en base a los parámetros medidos. La calidad está directamente relacionada con el grado de satisfacción con el usuario que ingresa al sistema.

Tabla 4.13

Evaluación de la calidad global.

Característica	%
Funcionalidad	88,8%
Confiabilidad	80,36%
Usabilidad	76%
Mantenimiento	83%
Portabilidad	82%
Calidad global	**82%**

Nota. Se muestra la evaluación de la calidad global.

El sistema según los parámetros evaluados tiene una calidad global de un **82%,** lo que significa un adecuado funcionamiento del sistema.

4.3. SEGURIDAD

4.3.1. Norma ISO/IEC 27000

El sistema web y la aplicación móvil debe contar con medidas de seguridad como la encriptación, como también debe tener variables de sesión. La encriptación ayuda a cifrar contraseñas, el usuario debe tener un usuario y contraseña válida para tener vía libre para el acceso, como también esto debe crear una variable de sesión, y cuando finaliza la tarea se destruye las variables de sesión creadas por el sistema.

4.3.1. Aplicación web y aplicación móvil

El proyecto tiene implementado varios recaudos en cuanto a seguridad:

- ✓ Control de accesos
- ✓ Codidficación de datos
- ✓ Protección de peticiones GET, POST, PUT, DELETE.

4.3.1.1. Control de acceso

Todos los accesos de la aplicación móvil y web están controlados por un token en este caso Json Web Token, lo cual es devuelto al usuario cuando sus datos de acceso son correctos, de esta manera el token tiene un tiempo de caducidad lo que pone más seguridad a todo acceso por parte del cliente lo que impide otras vulnerabilidades de acceso.

Figura 4.1

Token de inicio de sesión


```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZGxvZ2luIjo5LCJ1c2VybmFtZSI6Impj
uaS5tQGdtYWlsLmNvbSIsInJvbGUiOiJBRE1JTiIsImlhdCI6MTYzNDY1NzM1MSwiZXhwIjoxN
OTUxfQ.QQffFv4Hb_1V2wPZBtHZ4_RGaknOkcrtqVN_qzVq2Xc",
  "roles": "ADMIN"
}
```


Nota. Se muestra el token de seguridad para iniciar sesión.

Tabla 4.14

Nivel de acceso y seguridad Administrador.

Usuarios	Operaciones de administrador				
	Altas	Bajas	Modificaciones	Consultas	Reportes
Usuarios					
Proveedores					
Pedidos					
Productos					
Categorías	x	x	x	x	x
Ventas					
Compras					

Nota. Se muestra el nivel de acceso para el administrador.

Tabla 4.15

Nivel de acceso y seguridad Vendedor.

Usuarios	Operaciones de administrador				
	Altas	Bajas	Modificaciones	Consultas	Reportes
Usuarios	x	x	x	x	x
Pedidos		x		x	x
Ventas		x	x	x	x
Compras					x

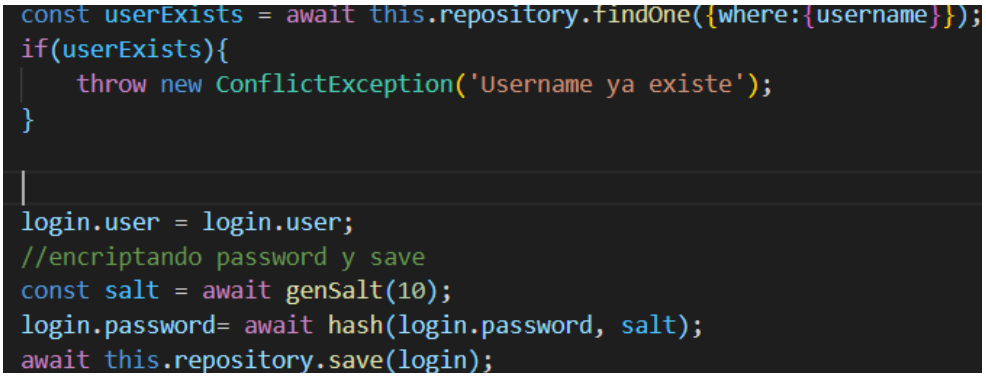
Nota. Se muestra el nivel de acceso del vendedor.

4.3.1.2. Codificación de datos contraseña

Cuando un usuario pone como entrada sus datos personales y lo que viene a ser su contraseña, lo cual le permite el acceso y inicio de sesión en su cuenta personal, esto es encriptado por los algoritmos de cifrado hash, de este modo el dato es guardado en la base de datos de forma es más complicado su descifrado por parte de gente desconocida.

Figura 4.2

Codificación de contraseña



```
const userExists = await this.repository.findOne({where:{username}});
if(userExists){
    throw new ConflictException('Username ya existe');
}

login.user = login.user;
//encriptando password y save
const salt = await genSalt(10);
login.password= await hash(login.password, salt);
await this.repository.save(login);
```

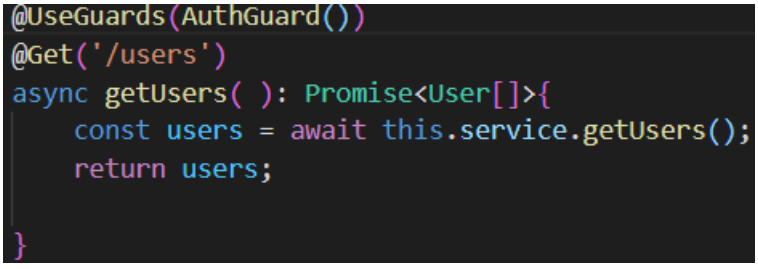
Nota. Se muestra el código para codificar la contraseña.

4.3.1.2. Protección de peticiones con guards

En cuanto al acceso al api rest que brinda el backend, para que no sea accedido desde otro lado, utilizamos los beneficios del framework netsjs, y para restringir las entradas fraudulentas utilizamos Guards, lo que hace el mismo es que permite solamente las peticiones de los que tengan un token valido. De esta manera se tiene un servicio más seguro.

Figura 4.3

Decorador guard



```
@UseGuards(AuthGuard())
@Get('/users')
async getUsers( ): Promise<User[]>{
    const users = await this.service.getUsers();
    return users;

}
```

Nota. Se muestra el decorador guard para proteger los apis.

4.3.1.3. Política de creación de copias de respaldo

Las políticas de creación de copias de respaldo sirven para mitigar cualquier daño que pueda sufrir el sistema. Para lo cual se debe realizar periódicamente el respaldo.

Tabla 4.14

Política de creación de copias de respaldo.

Copia de respaldo	
Tipo de copia	Completa, datos
Fecha	(Año, mes, día)
Datos	Se debe velar la confidencialidad de los mismos, en un lugar que tenga la menor posibilidad de sufrir un daño.
Técnico	Encargado del manejo del sistema.

Nota. Se muestra el detalle de la creación de copias de seguridad.

4.4. ESTIMACIÓN DE COSTOS

La tarea principal es medir los costos y recursos necesarios para el desarrollo completo del proyecto, como también se necesita estimar el tiempo requerido. Para este fin se utilizó el Método COSMIC.

4.4.1. Costo hardware

Para calcular el costo hardware se enumera en una lista, en nuestro caso son los servicios de alojamiento del sistema web y la subida de la app a play store, para lo cual se detalla en la siguiente tabla.

Tabla 4.15

Costo hardware.

N°	Descripción	Mes	Tiempo	Total
1	Costo de alojamiento web	70/mes	6 meses	420
2	Subida al Play Store	Ilimitado	Ilimitado	250
		Total costo en bolivianos		**670**

Nota. Por tanto, el costo del prototipo para poner en la nube es de 670 Bs.

4.4.2. Método de estimación COSMIC

COSMIC es un método de análisis de puntos de función de segunda generación, en el cual se determina el tamaño funcional del software a partir del número de interacciones entre los procesos funcionales.

Los pasos para realizar esta medición son los siguientes:

Tabla 4.16

Procesos de medición de COSMIC.

Fase	Procesos	Descripción
1	Estrategia de medición	Definición de partes del software a medido.
2	Mapeo	Requerimientos funcionales del usuario.
3	Medición	Tamaño funcional PFC.

Nota. Se detalla el proceso de medición COSMIC.

Fase 1: Estrategia de medición

Se define el propósito y alcance de la medición de software, que incluye los requerimientos funcionales de usuario que se medirán.

Tabla 4.17

Requerimientos funcionales sistema admin web.

N°	Descripción
R1	Inicio de sesión y después muestra la plataforma administrador .
R2	Altas bajas de productos.
R3	Altas bajas de categorías.
R4	Altas bajas de usuarios.
R5	Dar viabilidad de pedidos de productos.
R6	La web necesita autenticación de usuario.
R7	Altas bajas de compras
R8	Altas bajas ventas
R9	Altas bajas proveedores

Nota. Se muestra los requerimientos funcionales de sistema web.

Tabla 4.18

Requerimientos funcionales aplicación movil.

N°	Descripción
R1	Inicio de sesión y después muestra el menú categoría.
R2	La aplicación tiene menú productos.
R3	El usuario puede ver la lista de pedidos.
R4	Permite al usuario ver la lista de productos deseados.
R5	La aplicación captura la ubicación del cliente para la entrega.
R6	La aplicación necesita autenticación de usuario para enviar pedido.

Nota. Se muestra los requerimientos funcionales de la app.

Usuarios: Los usuarios funcionales del sistema son, administrador, cliente, usuario, vendedor, repartidor.

Fase 2 y 3: Mapeo y medición

Tabla 4.19

Matriz de movimiento de datos.

Proceso	Movimiento de datos	(E)Entrada	(X)Salida	(R)Lectura	(W)Escritur	Total
Login	Disparo de datos	1				
	Datos de usuario y contraseña	1			1	
	Vista usuario		1			
	Mensaje de error		1			
	Suma					5
Consultar productos deseados	Disparo de datos	1				
	Vista de productos		1			
	Suma					2
Perfil de cliente	Disparo de datos	1				
	Datos cliente			1		
	Detalle cliente		1			
	Suma					3
Editar perfil cliente	Disparo de datos	1				
	Datos usuario			1		
	Guardar				1	
	Suma					3
Editar usuario contraseña	Disparo de datos	1				
	Datos de usuario contraseña				1	
	Mensaje de éxito		1			
	Mensaje error		1			
	Suma					4

Pedido cliente	Disparo de datos	1			
	Vista categoría		1		
	Selección categoría	1			
	Vista productos		1		
	Agregar al carrito	1		1	
	Vista carrito		1	1	
	Agregar dirección	1		1	
	Enviar			1	
	Mensaje de error		1		
	Mensaje exitoso		1		
		Sumar			13
Eliminar productos deseados	Disparo de datos	1			
	Vista productos deseados		1		
	Eliminar			1	
	Mensaje error		1		
	Mensaje salida		1		
		Suma			5
Eliminar productos del carrito	Disparo de datos	1			
	Vista productos en carrito		1		
	Eliminar agregados y guardar			1	
		Suma			3
Registro cliente	Disparo de datos	1			
	Vista formulario		1		
	Llenado datos	1			
	Guardar			1	
	Mensaje de error		1		
	Mensaje exito		1		
		Suma			6
Historial pedido cliente	Disparo de datos	1			
	Vista productos pedidos		1		
		Suma			2
Cargar categorias	Disparo de datos	1			
	Agregar categoría			1	
	Mensaje error		1		
	Mensaje exito		1		
		Suma			4
Cargar productos	Disparo de datos	1			
	Agregar producto			1	
	Mensaje error		1		
	Mensaje exito		1		
		Suma			4
Registro usuario	Disparo de datos	1			
	Agregar usuario			1	
	Mensaje error		1		
	Mensaje exito		1		
		Suma			4
Registro proveedor	Disparo de datos	1			
	Agregar proveedor			1	
	Mensaje error		1		
	Mensaje exito		1		
		Suma			4
Registro compras	Disparo de datos	1			
	Agregar compras			1	
	Mensaje error		1		
	Mensaje exito		1		

		Suma 1			4
Aprobar pedido	Disparo de datos Marcar aprobado Asignar repartidor		1 1		
		Suma 1			3
Eliminar categoria	Disparo de datos Eliminar categoría Mensaje error Mensaje exito	1 1 1	1		
		Suma 1			5
Eliminar producto	Disparo de datos Mensaje error Mensaje exito	1 1			
		Suma 1			3
Eliminar usuario	Disparo de datos Mensaje de error Mensaje exito	1 1			
		Suma 1			3
Eliminar proveedor	Disparo de datos Mensaje error Mensaje éxito	1 1			
		Suma 1			3
Eliminar compra	Disparo de datos Mensaje error Mensaje exito	1 1			
		Suma 1			3
Eliminar pedido	Disparo de datos Mensaje error Mensaje exito	1 1			
		Suma 1			3
Buscar producto	Disparo de datos Vista productos	1			
		Suma 1			2
Mostrar ventas	Disparo de datos Vista ventas	1			
		Suma 1			2
Generar reportes	Disparo de datos Datos		1		
		Suma			2
TOTAL		**CFP**			**95**

Nota. Se muestra la matriz de movimiento de datos.

De esta manera se tiene los puntos de función que en total son:

95 CFP puntos de función

Costo mes del equipo de trabajo de desarrollo de software

Para determinar el costo mes se debe considerar, cantidad de desarrolladores, analista de prueba, diseñador, líder de proyecto, etc. Como también se debe considerar otros gastos del personal como beneficios de fin de año, seguros.

Costo por grupo de personas = 3000*3 = 9000 bs mes.

La unidad de medida basándonos en el historial de desarrollo del equipo en un determinado tiempo es de 48 puntos de función COSMIC mes.

Determinar el costo por unidad de medida

Para determinar el costo de desarrollo por cada punto de función se utiliza la siguiente formula:

$$Costo\ por\ punto\ de\ función = \frac{\text{Costo mes del equipo de trabajo}}{\text{Puntos de función del mes}}$$

$$Costo\ por\ punto\ de\ función = \frac{9000}{48}$$

$$Costo\ por\ punto\ de\ función = 187{,}5\ \text{Bs.}$$

Se determinó el costo por punto función del sistema de 187,5 Bs.

Estimación de costo de software

Teniendo el tamaño del software y el costo por unidad de medida, se calcula el costo del proyecto de software aplicando la siguiente fórmula:

$$Costo\ de\ software = \text{Tamaño del software} * \text{Costo por punto función}$$

$$Costo\ de\ software = 95\ \text{CFP} * 187{,}5\ \text{Bs}$$

$$Costo\ por\ punto\ de\ función = 17.812{,}5\ \text{Bs.}$$

Según los cálculos se determinó una estimación de 17.812,5 Bs. De costo del sistema desarrollado.

El costo total: 670 + 17.812,5 = 18.482,5 Bs.

Tiempo de duración del desarrollo de software

Una vez se determinó los puntos función también podemos utilizar para calcular el tiempo de duración del desarrollo del sistema, para lo cual se aplica la siguiente fórmula:

$$Duración\ del\ desarrollo\ de\ software = \frac{\text{Puntos de función COSMIC}}{\text{Puntos de función COSMIC mes}}$$

$$Duración\ del\ desarrollo\ de\ software = \frac{95}{48}$$

$$Duración\ del\ desarrollo\ de\ software = 2 \text{ meses}$$

Según los cálculos se determinó el tiempo de desarrollo del software que tendrá un aproximado de 2 meses de trabajo.

Tabla 4.20

Resumen esfuerzo, trabajo y costo.

N°	Descripción
Esfuerzo	Personas 3 para el desarrollo del software.
Trabajo	La duración es de 2 meses.
Costo	El costo del desarrollo es de Bs. **17.812,5** .

Nota. Se muestra el resumen del esfuerzo, trabajo y costo.

4.4.3. Valor actual neto (VAN)

El VAN es un método de valoración de inversiones en la que partimos de la rentabilidad mínima que queremos obtener. Con esta rentabilidad mínima calcularemos el valor actualizado de los flujos de caja. Interpretación del VAN:

VAN > 0: Se recomienda pasara la siguiente etapa del proyecto.

VAN = 0: Es indiferente realizar la inversión.

VAN < 0: Se recomienda desechar o postergar.

Se calcula según la siguiente fórmula:

$$VAN = -A + \sum_{i=1}^{n} \frac{Q_i}{(1+k)^i}$$

Donde:

Q_1, Q_2, …, Q_n: Flujos de caja.

k: Tasa de descuento seleccionado.

A: Inversión inicial.

n: Vital útil del proyecto.

i: Periodo

Se calcula el Van con los siguientes datos:

A=17.812,5 Bs, k=0.12, n =5 y los flujos de caja que se muestran en la tabla siguiente:

Tabla 4.21

Flujo de caja por años.

Año	Gastos	Beneficios	Flujo de caja(BS)
2023	3000	6200	3200
2024	3500	7500	4000
2025	3800	9000	5200
2026	4000	10000	6000
2027	4500	12100	7600
TOTAL	**18800**	**44800**	**26000**

Nota. Se muestra el flujo de caja por años.

$$VAN = -17.812{,}5 + \frac{3200}{(1+0.12)^1} + \frac{4000}{(1+0.12)^2} + \frac{5200}{(1+0.12)^3} + \frac{6000}{(1+0.12)^4} + \frac{7600}{(1+0.12)^5}$$

$VAN = 60.23\ Bs.$

El resultado es mayor a 0 lo que significa que el proyecto es aceptable.

4.4.4. Tasa interna de retorno (TIR)

Es una tasa porcentual que indica la rentabilidad promedio anual que genera el capital que permanece invertido en el proyecto. También se define como la tasa de descuento que hace que el VAN=0, su valor no depende del tiempo y representa el máximo costo que el inversionista podría pagar por el capital prestado.

TIR > 0: Se recomienda pasara la siguiente etapa del proyecto.

TIR = 0: Es indiferente realizar la inversión.

TIR < 0: Se recomienda desechar o postergar.

Se calcula según la siguiente fórmula:

$$0 = -A + \sum_{i=1}^{n} \frac{Q_i}{(1 + TIR)^i}$$

Donde:

Q_1, Q_2, …, Q_n: Flujos de caja.

A: Inversión inicial.

n: Vital útil del proyecto.

i: Periodo

Se calcula TIR con los siguientes datos:

A=17.812,5 Bs, n =5 y los flujos de caja que se muestran en la tabla 5.6.

Se tiene un valor de TIR de 12%. El proyecto es viable.

Tabla 4.22

Tiempo de recuperación de la inversión.

Número de años	Descripción
4	La recuperación de la inversión en el costo del software de 17.812,5 Bs.

Nota. Se muestra el tiempo de recuperación de la inversión.

Tabla 4.23

Resumen de costos.

Nombre	Descripción
Desarrollo del sistema	17.812,5 Bs.
Servidor	70 bs mes, 6 meses 420.
PlayStore	250 bs.
Total	**18482.5 bs** primera inversión.

Nota. Se muestra el resumen de costos para poner en marcha el sistema.

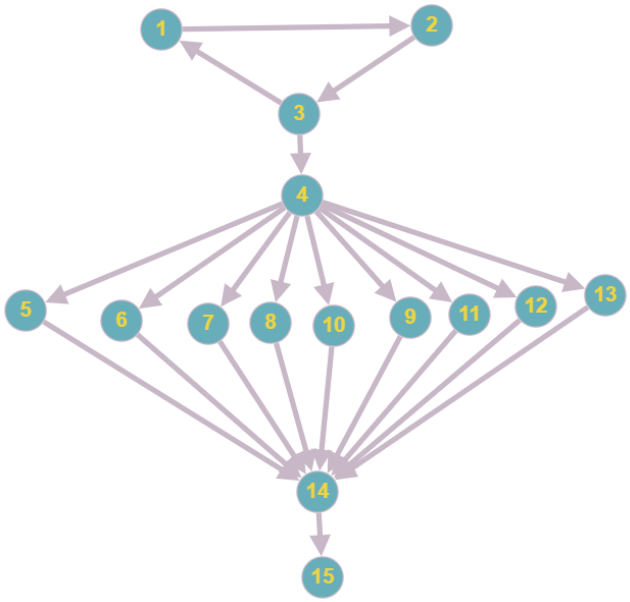
4.5. PRUEBAS DE SOFTWARE

Se podrá observar y verificar si la web y aplicación móvil, están en condiciones o está funcionando acorde a lo que se esperaba en primera instancia, para lo cual se aplicará la complejidad ciclomática para determinar los mejores caminos a seguir.

4.5.1. Prueba de caja blanca

Figura 4.4

Flujograma del sistema web



Nota. Se muestra el flujograma para determinar la vía más adecuado.

Donde:

1. Inicio de sistema
2. Usuario y contraseña
3. Validar usuario y contraseña
4. Menú principal
5. Gestión usuarios
6. Gestión categoría
7. Gestión producto
8. Gestión proveedor
9. Gestión compras
10. Gestión pedido
11. Gestión ventas
12. Gestión informes
13. Gestión reportes
14. Fin ciclo sistema
15. Fin sistema

La complejidad se obtiene en base al grafo de flujo se define como:

$$V(G) = E - N + 2$$

Donde:

E = número de aristas.

N = número de nodos.

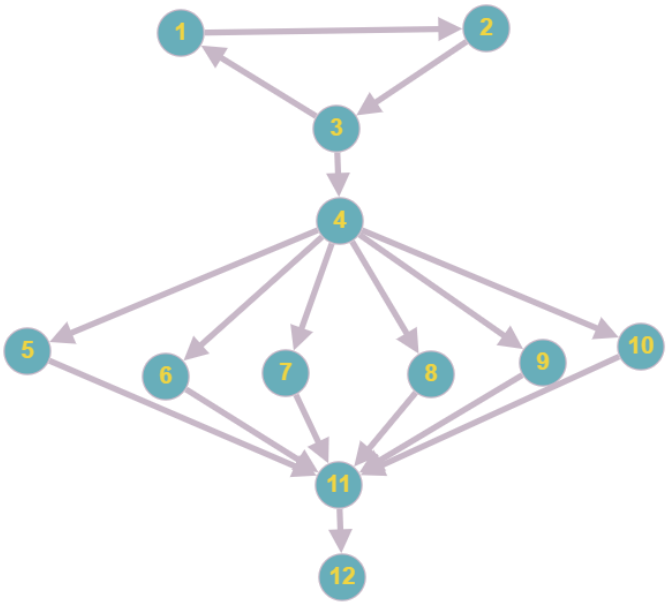
$$V(G) = 23 - 15 + 2$$

$$V(G) = 10$$

Por tanto, la complejidad ciclomática es: V(G)= 10, esto significa que existe 10 caminos independientes.

Figura 4.5

Flujograma aplicación móvil



Nota. Se muestra el flujograma para determinar la vía más adecuado de la app.

Donde:

1. Inicio de sistema
2. Usuario y contraseña
3. Validar usuario y contraseña
4. Menú principal
5. Productos deseados
6. Carrito
7. Productos
8. Categoría
9. Configuración
10. Historial pedidos
11. Fin ciclo sistema
12. Fin sistema

La complejidad se obtiene en base al grafo de flujo se define como:

$$V(G) = E - N + 2$$

Donde:

E = número de aristas.

N = número de nodos.

$$V(G) = 17 - 12 + 2$$

$$V(G) = 7$$

Por tanto, la complejidad ciclomática es: V(G)= 7, esto significa que existe 7 caminos independientes.

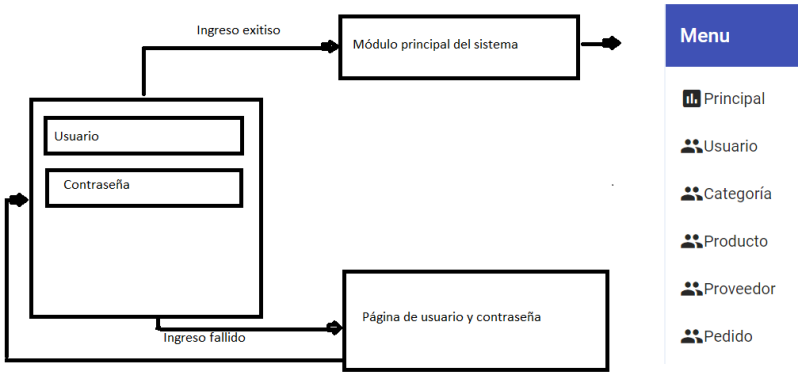
4.5.2. Prueba de caja negra

La prueba de caja negra consiste en probar cada una de las funciones del sistema. Con la prueba se busca que las funciones sean operativas, buscar la cantidad de errores posibles que pueda tener.

A continuación, se muestra la prueba de autenticación del cliente y usuario tanto en la aplicación móvil como en la web.

Figura 4.6

Prueba autentificación administrador



Nota. Se muestra la prueba de autentificación para el administrador sistema web.

Tabla 4.24

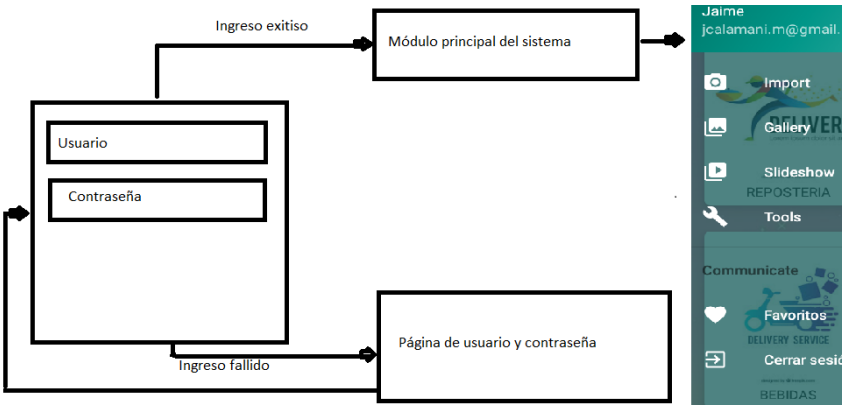
Prueba de caja negra autenticación sistema web.

	Entrada
Aprobación	Registro de usuarios. Registro productos. Registro categorías. Registro proveedores Registro compras. Registro ventas.
Seguridad	Una vez autenticado el usuario.
Software	Funciones Menú administrador. Menú vendedor.
Resultado	Salida Reporte de productos mínimos. Reporte de ventas. Reporte de proveedores.
Seguridad	Resultado que verán los usuarios que acceden al sistema.

Nota. Se muestra la prueba de caja negra autentificación en sistema web.

Figura 4.7

Prueba autentificación cliente app



Nota. Se muestra la prueba de autentificación para el cliente en la aplicación.

Tabla 4.25

Prueba de caja negra autenticación app móvil.

	Entrada
Aprobación	Registro de cliente. Registro carrito. Registro pedido.
Seguridad	Una vez autenticado el usuario.
	Funciones
Software	Menú cliente
	Salida
Resultado	Historial de pedidos.
Seguridad	Resultado que verán los usuarios que acceden al sistema.

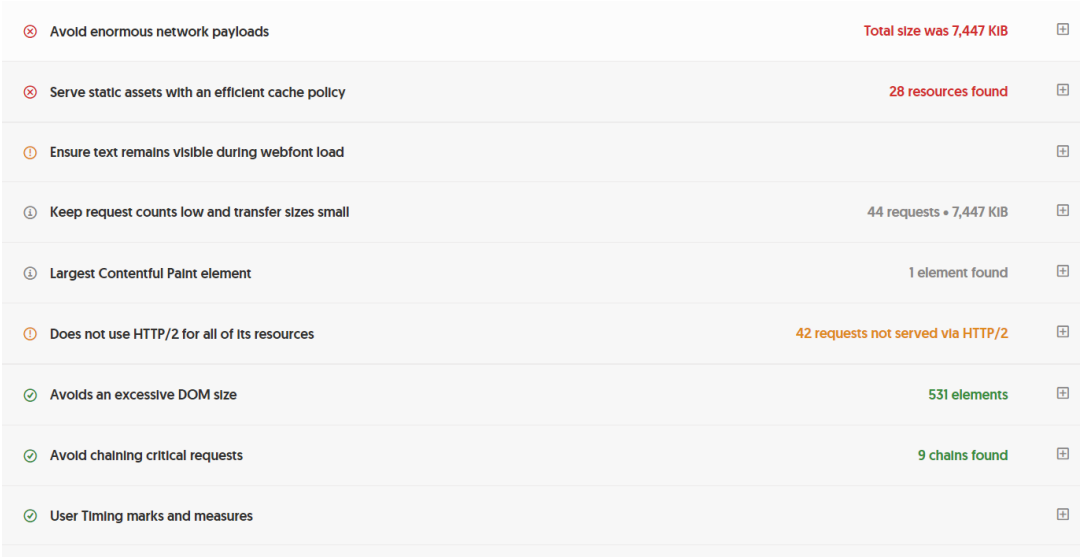
Nota. Se muestra la prueba de caja negra autentificación cliente.

4.5.3. Prueba de estrés

El objetivo de estas pruebas es asegurar que el sistema no colapse a una gran concurrencia de usuarios.

Figura 4.8

Resumen prueba de estrés del sistema.



Nota. Se muestra los datos de la prueba de estrés, donde lo más importante es la cantidad de peticiones que soporta nuestras apis.

4.5.4. Resultados

En la siguiente tabla 4.26 se describe el antes y el después.

Tabla 4.26

Escala de ponderación.

Escala
Pésimo
Malo
Regular
Optimo
Muy
Bueno

Nota. Se muestra la escala de ponderación para evaluar adecuadamente.

Tabla 4.27

Antes y después del proyecto.

ANTES	DESPUES	ESCALA
Registros manual de productos, proveedores, categorías, compras.	El registro de productos, proveedores, categorías, compras, mediante el sistema es más rápido y confortable. .	**Optimo**
Registro manual de ventas	El registro y ver los detalles es más accesible y rápido.	**Optimo**
Búsqueda manual de productos más vendidos.	Con el sistema te visualiza en orden de productos más vendidos.	**Optimo**
Control manual sobre stock de productos después de cada venta.	El sistema disminuye el producto con cada venta.	**Optimo**

Nota. Se muestra las mejoras de lo que era antes y ahora.

CAPÍTULO V

CONCLUSIONES

Y

RECOMENDACIONES

5.1. INTRODUCCIÓN

En el presente capítulo se detallan las conclusiones y recomendaciones que se observó en el proyecto, ya que en el camino fue apareciendo detalles que no se pudieron implementar para que el sistema sea más completo.

5.2. CONCLUSIONES

a) El alcance de los objetivos

A continuación, se realiza la valoración de los objetivos del proyecto mencionado en el capítulo I.

- ✓ Realizar un módulo de registro de clientes, productos, proveedores, ventas, compras e inventario, para mejorar la organización de los documentos y evitar pérdida de información. Se logró realizar los módulos al 100%.
- ✓ Controlar las fechas de vencimiento, entrada y salida de productos para no tener confusiones ni datos erróneos. Se logró controlar las fechas de vencimiento, productos al 100%.
- ✓ Desarrollar el módulo de pedidos, ventas, para mejorar el servicio como el control de los procesos. Se logró desarrollar el módulo pedidos, ventas al 100%.
- ✓ Generar reportes de ventas realizadas, reportes clientes, proveedores, productos, compras, para obtener credibilidad en la información. Se logró generar reportes al 100%
- ✓ Implementar módulos en la app móvil para seleccionar categorías, productos, envío de pedido, agregar carrito, favoritos, historial pedidos, reclamos. Se logró implementar los módulos en la app al 100%.

Después de realizar modificaciones tanto en el backend como en frontend de la aplicación y sistema web, para cumplir con los requerimientos recabados de parte del administrador de la micromarket, se concluyó dando cumplimiento en los módulos planteados.

También mencionar la importancia de aplicar dos metodologías tanto para la app como la parte web, para un desarrollo adecuado y en menor tiempo posible.

En conclusión, en el capítulo iv, sección 4.27 resultados, se alcanzó con el apoyo que brinda el sistema de información web y aplicación móvil para la administración de inventario, ventas y compras online para la Micromarket Home Service al 100%.

5.3. RECOMENDACIONES

A partir del presente trabajo realizado se propone las siguientes recomendaciones, con el fin de buscar el mejoramiento del sistema.

- ✓ Se recomienda antes de actualizar las librerías tanto del backend como del frontend, realizar una evaluación de los cambios que puede surgir.
- ✓ Se recomienda incorporar un módulo de pagos online para la aplicación móvil.
- ✓ Se recomienda tener cuidado con los claves de acceso para el administrador.
- ✓ Se recomienda realizar respaldo de la información almacenada por lo menos una vez al mes.
- ✓ Revisión periódica de la aplicación y sistema web por un entendido en desarrollo de software para tener eficiencia en el funcionamiento.
- ✓ Se recomienda adicionar modulos de seguimiento pedido.
- ✓ Se recomienda mejorar los modulos de pedidos con socket.
- ✓ Se recomienda contratar el servicio de backup en el servidor digital ocean.

REFERENCIAS BIBLIOGRAFÍCAS

ISO 25000. (2021). *NORMAS ISO/IEC 25000*. Obtenido de NORMAS ISO/IEC 25000: https://iso25000.com/index.php/normas-iso-25000?start=4

Abmed. (01 de 06 de 2019). *Nest.js*. Obtenido de Stips: https://stips.wordpress.com/2019/05/01/introduccion-a-nest-js-para-desarrolladores-angulares-sitepoint/

Alvarez Tanaka, R. A. (2009). *Análisis y propuesta de implementación de pronósticos y gestión de inventarios en una distribuidora de productos de consumo masivo.* Universidad Católica del Perú.

Alvarez, C. (03 de 01 de 2016). *Sequelize*. Obtenido de https://www.genbeta.com/desarrollo/sequelize-un-orm-para-node-js

Angular. (2022). *Angular*. Obtenido de Angular: https://angular.io/docs

Angular Material. (2022). *Angular Material*. Obtenido de Material: https://material.angular.io/guide/theming

Appyoutself. (24 de 06 de 2017). *Tipos de aplicaciones móviles*. Obtenido de https://appyourself.net/es/blog/tipos-de-aplicaciones-moviles/

Atlassian. (2021). *Diferencias entre integración continua, entrega continua y despliegue continuo*. Obtenido de entrega continua: https://www.atlassian.com/es/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Atlassian. (2021). *Diferencias entre integración continua, entrega continua y despliegue continuo*. Obtenido de Despliegue continuo: https://www.atlassian.com/es/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

BBC News Mundo. (09 de 04 de 2011). *Que son las Apps*. Obtenido de https://www.bbc.com/mundo/noticias/2011/04/110408_1336_tecnologia_apps_negocios_celulares_telefonos_inteligentes_dc

Biaje. (22 de 04 de 2015). *Arquitectura android*. Obtenido de http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android

Bolaños, L., Urrea, C., & Gomez, R. (20 de 02 de 2015). *Ingeniería web*. Obtenido de https://laingenieriaweb.wordpress.com/la-ingenieria-web/

Borbon, N. (12 de 03 de 2013). *Métricas de calidad iso 9126*. Obtenido de Evalucación de software: http://actividadreconocimiento-301569-8.blogspot.com/2013/03/norma-de-evaluacion-isoiec-9126.html

Borges, S. (19 de 11 de 2019). *PostgreSQL*. Obtenido de Infranetworking: https://blog.infranetworking.com/servidor-postgresql/

Brito, M., & Pinzon, A. (2016). Diseño de una aplicación móvil para la oferta de servicios de información (tendencias, precios y ubicación) enfocado a las prendas de vestir, accesorios y calzado. *Proyecto de grado*, 17.

Cachaga, E. (2015). Escala de ponderación. *Sistema web de gestión académica y repositorio virtual*, 101.

Cadima, E. (2013). Desarrollo de una tienda virtual mediante el estudio comparativo de una tienda física de productos de computación. *Tesis de grado*, 1.

Calle, J. (2014). Cuestionario de evaluación de uso. *Sistema de registro y seguimiento de tramites para personalidades jurídicas*, 81.

Castillo Peña, , G. E. (20 de 02 de 2018). *Implementación de un sistema web de gestión documentaria en la municipalidad distrital de Pararin-provincia Recuay-departamento de Ancash; 2017.* Obtenido de https://www.llamacreativa.com.ar/clientes/index.php?rp=/knowledgebase/3/iQue-es-una-aplicacion-o-sistema-web.html

Chagona Ramos, E. (2008). *Métodos y técnicas de investigación.* Obtenido de https://www. gestiopolis. com/metodos-y-tecnicas-de-investigacion.

Chiu, C. C. (2018). *Las pruebas en el desarrollo de software [Tesis de grado, Universidad Nacional de México].* Repositorio Institucional. Obtenido de Prueba

de estrés: https://d1wqtxts1xzle7.cloudfront.net/55180608/Las_pruebas_en_el_desarrollo_de_software-with-cover-page-v2.pdf?Expires=1663386735&Signature=Uqn1~tkNEOYpKtiQVyutuhMGWFQPPD6VTTaAqsI~3Y-MKMg4NjWB-JmBZM2IN7gCy5kEjkzkj5d5VtNPsrmswF8-RwsLF6bs~F9Pc1IFUlSPA2A0oCy

Cordova Lopez, J. L. (09 de 11 de 2020). *Influencia de los Aplicativos Móviles en la Compra en Línea de Alimentos Durante la Emergencia Sanitaria COVID-19.* Universidad César Vallejo, Lima. Obtenido de Compra online: https://www.consumoteca.com/comercio/compra-online/

COSMIC. (2021). *Método COSMIC*. Obtenido de Método COSMIC: https://cosmic-sizing.org/cosmic-sizing/intro/

Cruz, R. (12 de 01 de 2014). *Metodología mobile-D*. Obtenido de Doc2car: http://pegasus.javeriana.edu.co/~PA133-05-PMovVidaAutomotor/Metodologia.html

Developers. (18 de 02 de 2020). *Android Studio*. Obtenido de https://developer.android.com/studio/intro/?hl=ES

Durán, Y. (25 de 09 de 2012). *Administración del inventario: elemento clave para la optimización de las utilidades en las empresas. Visión gerencial.* Obtenido de https://es.wikipedia.org/wiki/Inventario

ESERP. (2021). *Métodos de control de inventarios*. Obtenido de Tipos de control de inventarios: https://es.eserp.com/articulos/metodo-control-inventarios/?_adin=02021864894

Fernandéz, O. B. (2005). *Introducción al lenguaje de programación Java.*

García Mendoza, G. M. (2015). *USOS Y TIPOS DE APLICACIONES MÓVILES.* Tecnológico Nac. México Inst. tecnológico Salina Cruz.

Gauchat, J. D. (2012). *El gran libro de Html5 Css3 y JavaScript.* Barcelona: MARCOMBO S.A.

Gestiopolis. (19 de 05 de 2020). *Que es inventario*. Obtenido de https://www.gestiopolis.com/que-es-inventario-tipos-utilidad-contabilizacion-y-valuacion/

Github. (02 de 08 de 2019). *Diseño*. Obtenido de Tipos de Diseño: https://gist.github.com/brunocascio/5e89fafa7fd86bdd1a715d2f6f0432d1

Github. (02 de 08 de 2019). *Diseño a nivel de componentes*. Obtenido de Tipos de Diseño: https://gist.github.com/brunocascio/5e89fafa7fd86bdd1a715d2f6f0432d1

Github. (02 de 08 de 2019). *Diseño arquitectónico*. Obtenido de Tipos de Diseño: https://gist.github.com/brunocascio/5e89fafa7fd86bdd1a715d2f6f0432d1

Github. (02 de 08 de 2019). *Diseño de datos*. Obtenido de Tipos de Diseño: https://gist.github.com/brunocascio/5e89fafa7fd86bdd1a715d2f6f0432d1

Github. (02 de 08 de 2019). *Diseño de interfaz*. Obtenido de Tipos de Diseño: https://gist.github.com/brunocascio/5e89fafa7fd86bdd1a715d2f6f0432d1

Gunawan, R., & Rahmatulloh, A. (04 de 07 de 2019). *JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service.*

Heredia, D. (30 de 01 de 2017). *Maquetación*. Obtenido de http://www.ticarte.com/contenido/que-son-los-mockup-y-como-se-hacen

HIlaquita, G. (2014). Aplicación móvil para la publicación de la revista emistur sobre lugares turísticos de Bolivia, utilizando la tecnología realidad aumentada. *Proyecto de grado*, 2.

Huanca, S. (2015). Ingeniería web. *Sistema web de control de pagos, citas e historiales clínicos*, 30.

Huanca, S. (2015). Metodología Uwe. *Sistema web de control de pagos, citas e historiales clínicos*, 32.

ISO 25000. (2 de Junio de 2021). *ISO /IEC 25000*. Obtenido de ISO /IEC 25000: https://iso25000.com/index.php/normas-iso-25000

ISO 25000. (2021). *NORMAS ISO/IEC 25000*. Obtenido de ISO/IEC 25010: https://iso25000.com/index.php/normas-iso-25000/iso-25010?start=6

ISO Tools. (2021). *Software y riesgos de seguridad ISO/IEC 27001*. Obtenido de Gestión de riesgos: https://www.isotools.org/normas/riesgos-y-seguridad/iso-27001/

ISO27000.ES. (2018). *ISO 27000*. Obtenido de ISO 27000.ES: https://www.iso27000.es/iso27000.html

Jhonson, L. (2021). Que es sistema. Obtenido de https://utexa.edu

Lindsay, J. (2000). Information System.

Lucas, J. (04 de 09 de 2019). *Node.js*. Obtenido de Openwebinars: https://openwebinars.net/blog/que-es-nodejs/

Malca, O. (17 de 08 de 2021). *Comercio electrónico.* Universidad del Pacífico, Perú. Obtenido de Comercio electrónico.

Maldonado, C. E. (2021). Que es sistema complejo. *Revista Colombiana de filosofía de la ciencia*, 14. Obtenido de Sistema: https://unbosque.edu.co

Mamani, F. (2016). Usabilidad. *Sistema de conteo y control de peso automatizado en el ganado ovino*, 49-50.

Mangas Roca, I. (2018). *Diseño e implementación de un sistema automático de control de inventario en un almacén basado en minidrones.* (Doctoral dissertation, Universitat Politècnica de València).

Master. (07 de 03 de 2017). *Implementar*. Obtenido de www.definicionabc.com/general/implementar.php

Medina, J. (2014). *Pruebas de rendimiento TIC.* Madrid.

Morales Machuca, C. A. (03 de 08 de 2010). *Estado del arte: Servicios Web.* Universidad Nacional de Colombia, Tesis de Maestría.

Murillo, R. S. (2009). *Beneficios del comercio electrónico.*

Nest.JS. (2022). *Que es Nest.js*. Obtenido de NestJS: https://docs.nestjs.com/

NodeJS. (2022). *Que es Node.js*. Obtenido de Node.js: https://nodejs.org/en/docs/

Ochoa, R. (2018). Desarrollo de portal web y aplicación móvil para la estructuración de programas radiales. *Proyecto de grado*, 1-2.

Peño, J. M. (15 de 04 de 2015). *Pruebas de Software. Fundamentos y Técnicas [Proyecto, Universidad Politécnica de Madrid].* Repositorio Institucional. Obtenido de Caja Negra, Caja Blanca: https://gc.scalahed.com/recursos/files/r161r/w24802w/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf

Pérez Muzha , J. Y. (11 de 06 de 2022). *Implementación De Solución Web Para La Gestión De Pedidos Online Y Procesos Operativos Del "Almacén Vásquez".* UNIVERSIDAD AGRARIA DEL ECUADOR. Obtenido de Pedido online: https://www.consumoteca.com/comercio/pedido-online/

Poma, J. (2019). Aplicación y portal web para el diagnóstico preventivo y desarrollo cognitivo verbal de niños de 3 a 7 años con trastorno del espectro autista. *Proyecto de grado*, 1-2.

PostgreSQL. (2022). *Que es PostgreSQL*. Obtenido de PostgreSQL: https://www.postgresql.org/docs/

Pradel, J., & Raya, J. (2009). *Ingeniería de software.* España: Universitat Oberta de Catalunya.

Pressman. (2002). *Métricas de calidad.* México: McGrawHill.

Pressman. (2010). *Diseño.* México: Mc Graw Hill.

Pressman. (2010). *Diseño a nivel de componentes.* México: Mc Graw Hill.

Pressman. (2010). *Diseño arquitectónico.* México: Mc Graw Hill.

Pressman. (2010). *Diseño de datos.* México: Mc Graw Hill.

Pressman. (2010). *Diseño de interfaz.* México: Mc Graw Hill.

Pressman. (2010). *Estimación de costos con COCOMO II.* México: Mc Graw Hill.

Pressman. (2010). *Prueba alfa.* México: Mc Graw Hill.

Pressman. (2010). *Prueba beta.* México: Mc Graw Hill.

Programación y Más. (2021). *Diferentes tipos de test en desarrollo de software*. Obtenido de Test unitarios: https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software

Programación y Más. (2021). *Diferentes tipos de test para desarrollo de software*. Obtenido de Test de integración: https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software

Puciarelli, L. (2020). *Angular.* (C. Peña, Ed.) Miguel Liderkremer. Obtenido de https://books.google.es/

Ramíez Vique, R. (09 de 10 de 2019). *Métodos para el Desarrollo de Aplicaciones Móviles.* Fundavio per a la Universitat Oberta de Catalunya. Obtenido de Aplicación: https://es.wikipedia.org/wiki/Software_de_aplicaci%C3%B3n

Ramírez, R. (2012). Aplicación móvil. *Métodos para el desarrollo de aplicaciones móviles*, 5.

Retrofit. (2022). *Retrofit2*. Obtenido de https://square.github.io/retrofit/

Roca. (11 de 08 de 2017). *Que son los productos de consumo masivo*. Obtenido de https://www.iebschool.com/blog/productos-de-consumo-masivo-comercio-ventas/

Romero Parra, J. D. (18 de 02 de 2017). *Análisis de los tipos de maquetación web adaptativa y propuesta de diseño para el portal web “CIMOGSYS". BS thesis.* Escuela Superior Politécnica de Chimborazo. Obtenido de https://es.wikipedia.org/wiki/Maquetaci%C3%B3n_(edici%C3%B3n)

Romero, M. (2017). Desarrollo de aplicación móvil de compraventa y subasta online de aparatos electrónicos. *Proyecto de grado*, 7.

Ruiz, F. (2012). Ingeniería de Software. 15.

Sage, S. M. (1968). Information System.

Sandhusen L., R. (2002). *Mercadotenia.* Compañia Editorial Continental.

Schiaffarino, A. (12 de 03 de 2019). *Cliente servidor*. Obtenido de https://blog.infranetworking.com/modelo-cliente-servidor/

Sommerville. (2011). *Ingeniería de seguridad.* México: Pearson Educación.

Sommerville. (2011). *Ingeniería de software.* México: Pearson Educación.

Sommerville. (2011). *Requerimiento funcional.* México: Pearson Educación.

Sommerville. (2011). *Requerimiento no funcional.* México: Pearson Educación.

Sotomayor, R. (2017). *Implementación de un sistema web para mejorar el proceso de gestión académica en las escuelas de la PNP.*

SQLite. (2022). *SqLite*. Obtenido de Documentación: https://www.sqlite.org/docs.html

Thommson. (24 de 04 de 2019). *Que es una venta online*. Obtenido de https://www.promonegocios.net/venta/venta-online.html

TypeORM. (2022). *TypeORM*. Obtenido de TypeORM: https://typeorm.io/

Ugalde, F. (27 de 08 de 2019). *Slim framework*. Obtenido de https://www.franciscougalde.com/2019/08/27/introduccion-a-slim-framework/

Vanquez, C. E. (21 de 02 de 2015). Estimaciones de Software con COSMIC. *FATTO Consultoría y Sistemas*, 4-7. Obtenido de Método COSMIC: http://www.pmoinformatica.com/2018/02/medicion-estimacion-metodo-cosmic.html

Yujra, A. (2016). Ingeniería móvil. *Diccionario móvil especializado de teminología informatíca*, 30.

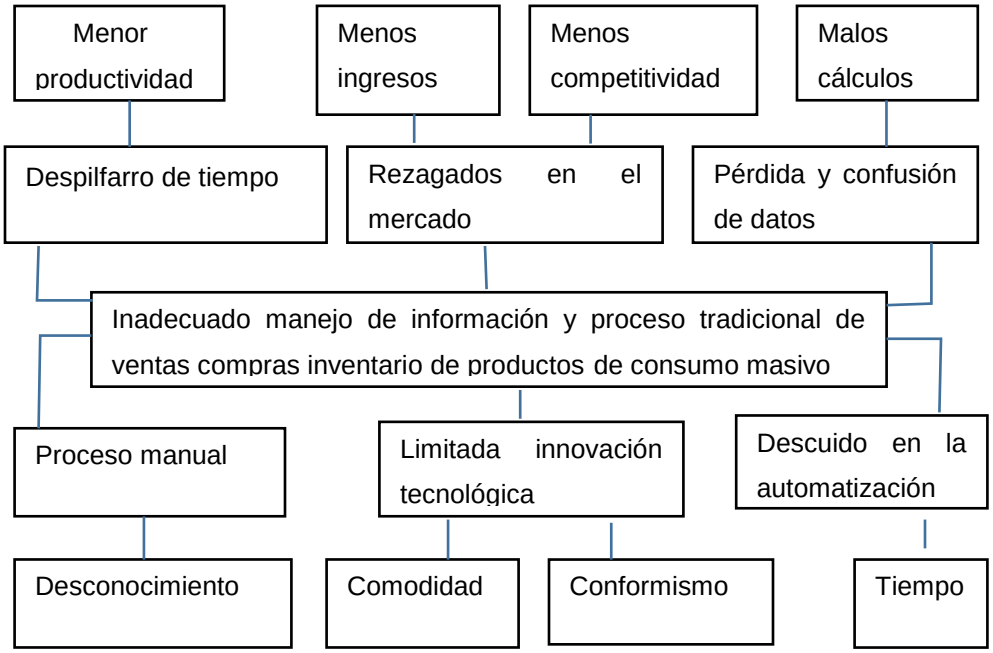
Zapata, C., & Olaya, Y. (2007). *Implementar.* Colombia: Litonueve.

Zarco, J. (2017). Ingeniería móvil. *Realidad aumentada aplicada al turismo de las iglesias de la ciudad de La Paz*, 9.

ANEXOS

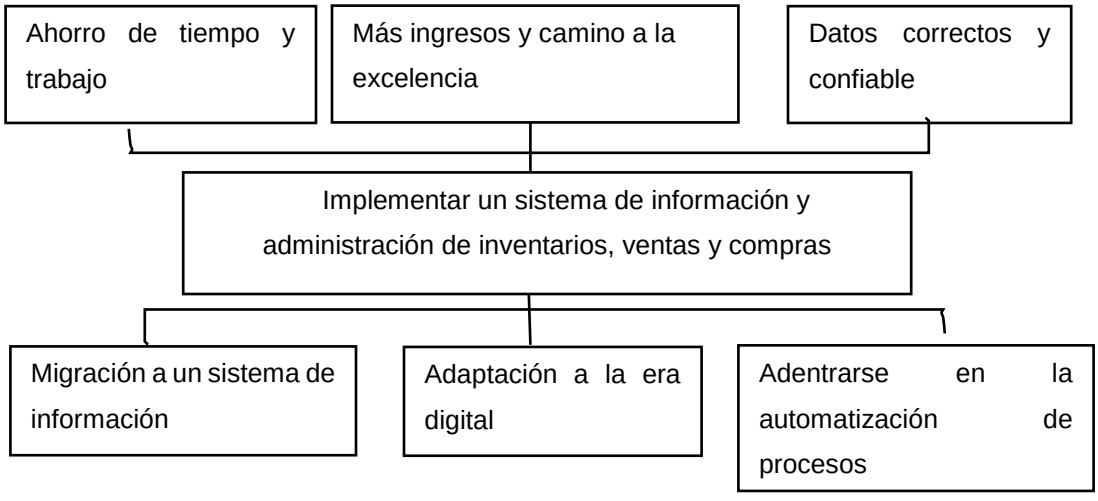
ANEXO A:

Árbol de problemas



ANEXO B:

Árbol de objetivos



ANEXO C:
Manual administrador

Manual administrador

SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE

Autor: Jaime Calamani Mamani

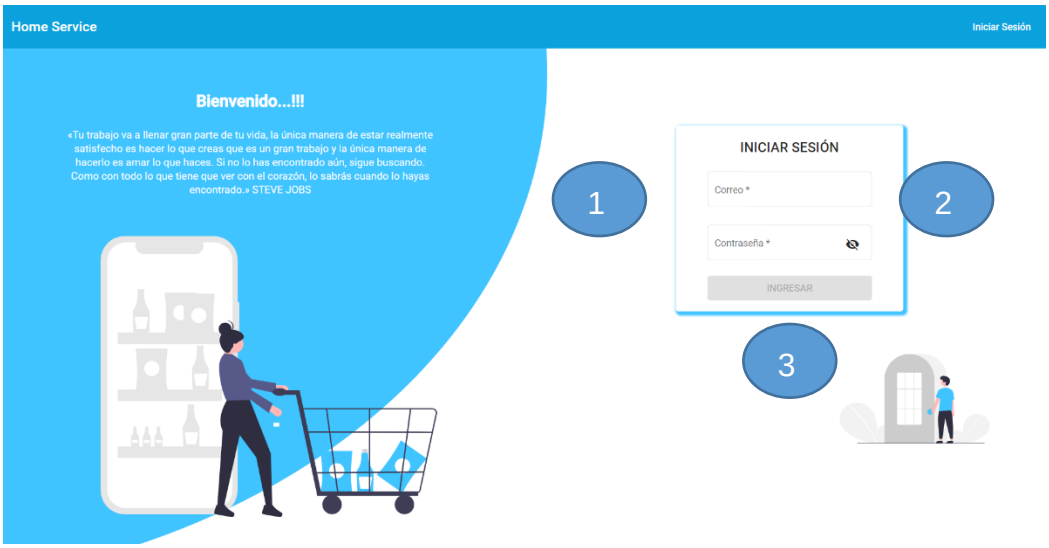
La Paz - Bolivia

2022

Manual administrador

1. Login

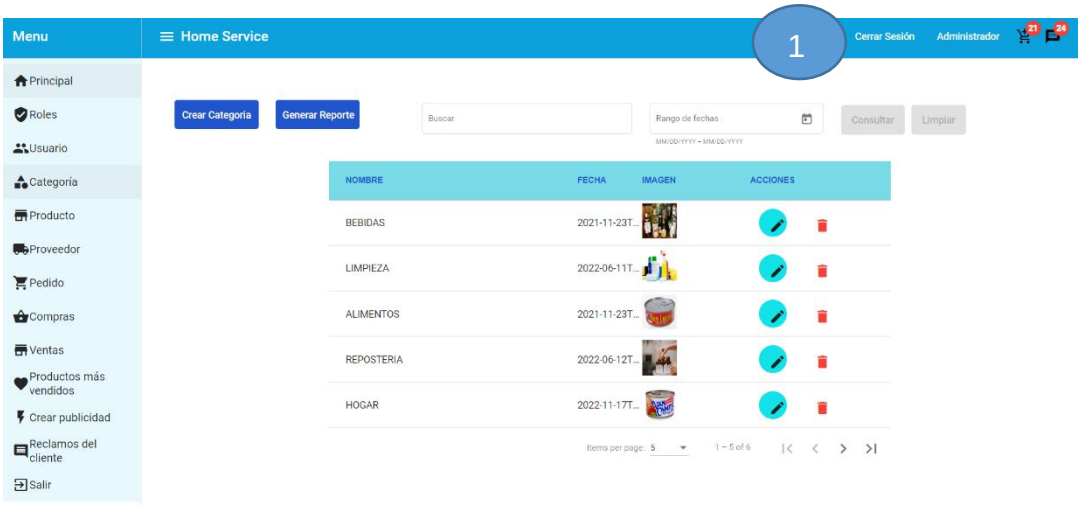
El login es la interfaz de acceso al sistema mediante correo y contraseña.



1. Correo: Debe introducir el correo con el que se registró.

2. Contraseña: Debe introducir la contraseña con que se dio de alta.

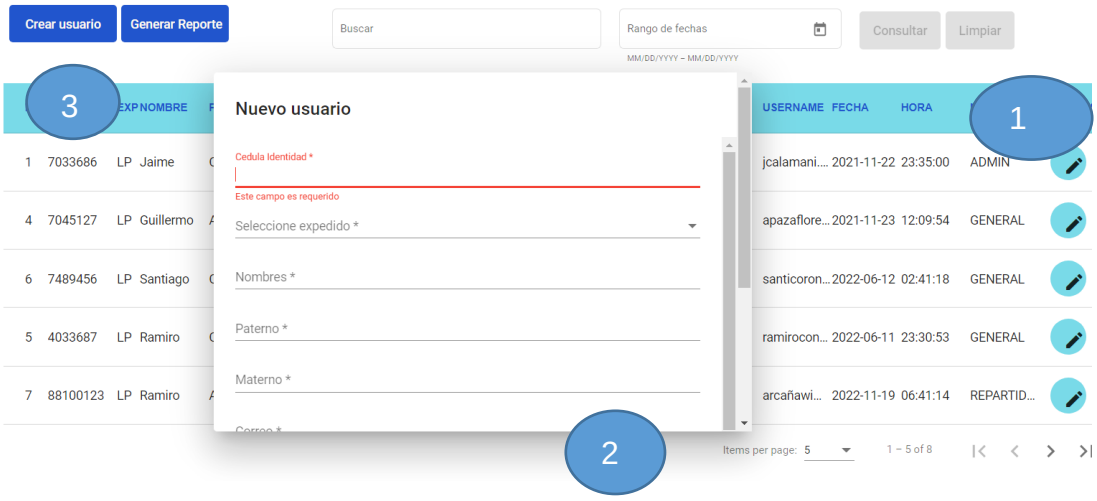
3: Ingresar: Dando click le llevará al menú principal del sistema



1. Cerrar sesión: Hacer click para salir del sistema.

2. Módulos

2.1. Módulo usuario.

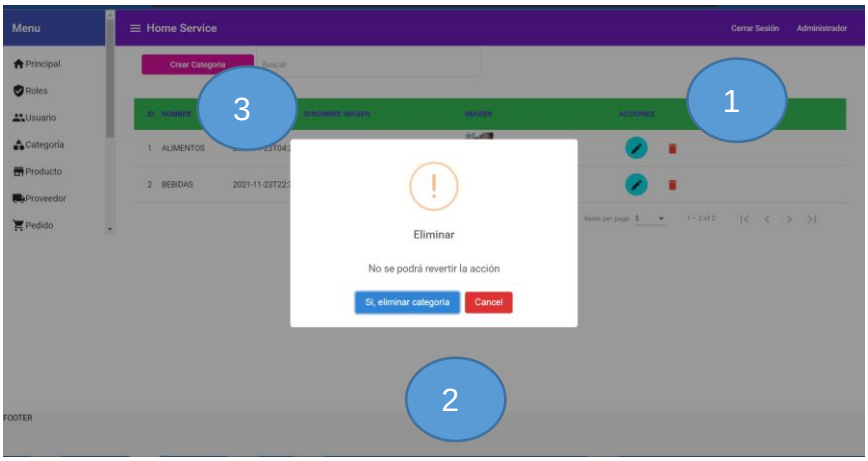


1. Para realizar la edición de un usuario se debe hacer click sobre el lápiz y para eliminar sobre la figura de eliminar color rojo, después que pone en aceptar o guardar todos los cambios serán grabados en la base de datos del sistema.

2. En la interfaz de edición después de cambiar los campos debe hacer click en guardar.

3. Si quiere crear un nuevo usuario debe hacer click en botón nuevo usuario.

2.2. Módulo categoría



1. Si quiere editar debe pulsar sobre el ícono de color celeste con la figura de lápiz, una vez se abrirá una ventana con los datos de la categoría que quiere cambiar, después debe pulsar en guardar para que todo sea grabado en la base de datos.

2. Despues de pulsar sobre el ícono de color rojo de elimiar se abrirá la ventana de aceptación para dar de baja a la categoría, si está convencido realmente pulsar en el botón si.

3. Si quiere agregar una nueva categoría al sistema pulsar sobre el botón nueva categoría, cuando se abra la ventana llene los datos necesarios para guardarlo.

2.3. Módulo producto

1. Para poner como oferta a un producto debe hacer click sobre la figura de publicidad, una vez esto se abrirá una ventana modal.

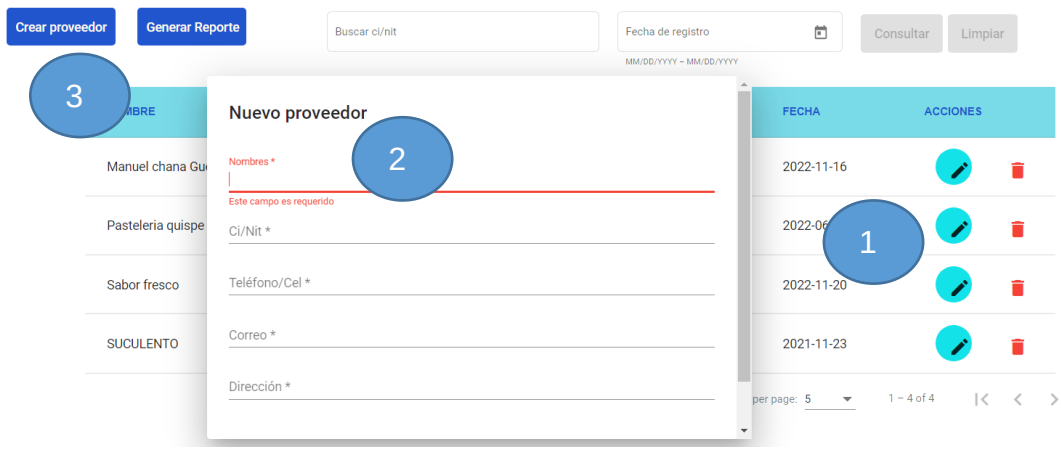
2. Debe poner monto en bs., de descuento a continuación desplegar las opciones y seleccionar en si para la oferta o en no para quitar oferta como también poner en cero el descuento, después pulsar en el botón guardar.

3. Para editar y eliminar un producto debe pulsar sobre el ícono de lápiz y sobre ícono de eliminar, una vez realizado esto le saltará una ventana modal donde debe cambiar en caso de que quiera editar o en botón si en caso de que quiera eliminar el producto.

4. Para generar el reporte de los productos debe pulsar sobre el ícono de pdf, de esta manera le llevará a otra pestaña con los datos de todos los productos.

5. Si quiere crear un nuevo producto pulsar sobre el botón nuevo producto y llenar los campos vacíos con los datos del producto.

2.4. Módulo proveedor



1. Para editar o eliminar un proveedor, hacer click sobre los íconos respectivamente.

2. Para generar reporte de todos los proveedores, click en botón pdf.

3. Para crear un nuevo proveedor llenar los campos vacíos y poner en guardar.

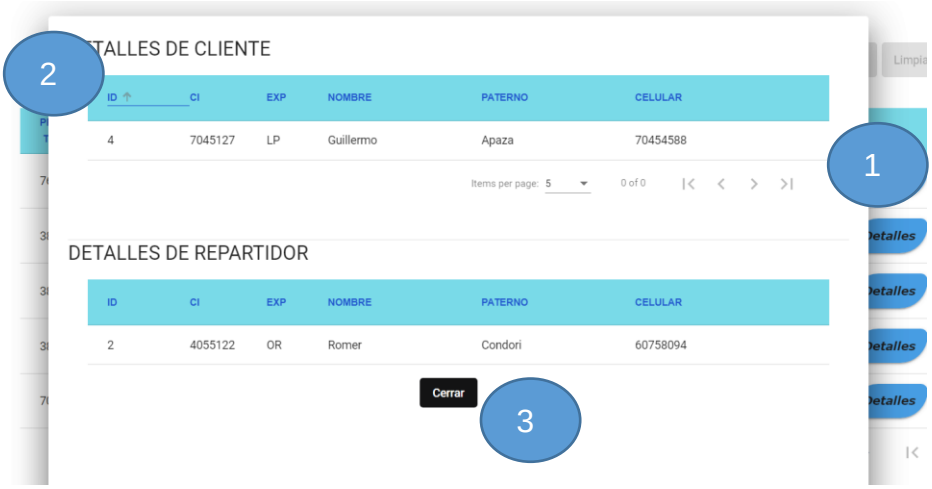
2.5. Módulo pedido

1. Para ver los detalles de cada uno de los pedidos pulsar sobre el botón detalles.

2. Para generar el reporte de los pedidos pulsar sobre el botón pdf.

3. Antes de dar de alta el pedido llenar las opciones con el vendedor que hace la venta y el repartidor designado.

2.6. Módulo de ventas



1. Para ver los detalles de la venta realizada pulsar sobre el botón detalles.

2. Para generar el reporte de la venta pulsar sobre el botón crear pdf

3. Para verificar los datos del cliente, y el repartidor que fue designado a su entrega.

ANEXO D:

Manual de cliente

Manual de cliente

SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE

Autor: Jaime Calamani Mamani

La Paz - Bolivia

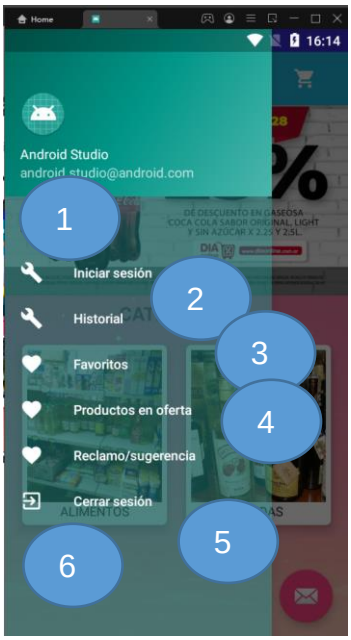
2022

Manual cliente

1. Módulo categoría de productos

1. Cuando entra a la aplicación despliega la interfaz principal, donde observa las categorías de productos existentes, como también si hay algún comunicado en el banner.

2. Menú



1. Para registrarte e iniciar sesión al sistema pulsar sobre la opción iniciar sesión.

2. Para ver el historial de tus pedidos pulsar sobre la opción historial.

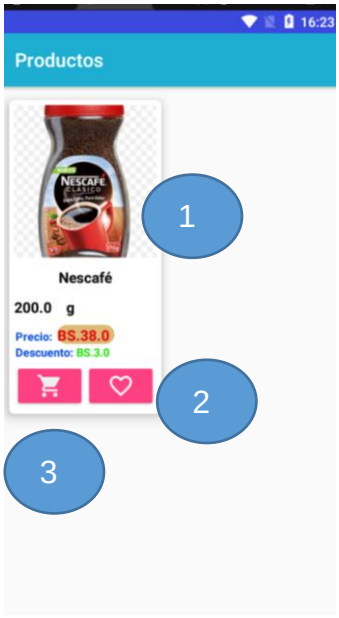
3. Para ver tus productos favoritos.

4. Para revisar que productos están en oferta.

5. Para reclamar o dar sugerencia pulsar sobre la opción mencionado.

6. Para salir de la aplicación pulsar en cerrar sesión.

3. Módulo productos

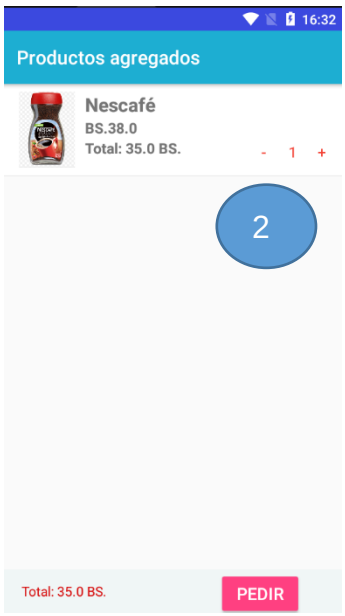


1. Después de pulsar sobre una categoría le muestra los productos disponibles para esa categoría, donde el producto tiene su nombre, su unidad de medida y peso, como el precio, descuento si está disponible.

2. Si quiere agregar el producto a la lista de productos favoritos pulsar sobre el botón con corazón.

3. Para agregar el producto al carrito pulsar sobre botón de carrito de compras.

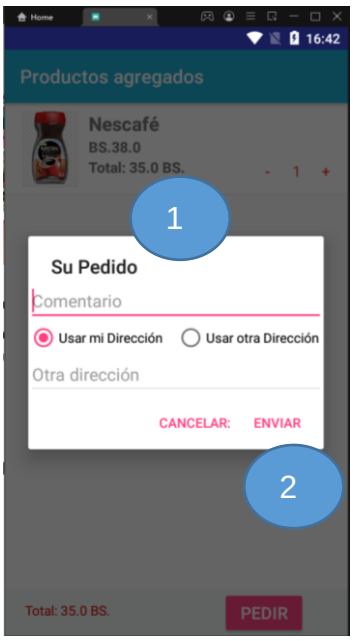
4. Módulo carrito



1. Para ver los productos al carrito pulsar en la interfaz principal sobre ícono de carrito.

2. Muestra todos los productos agregados, donde ahí puede incrementar la cantidad o quitar algún producto que no desee en última instancia.

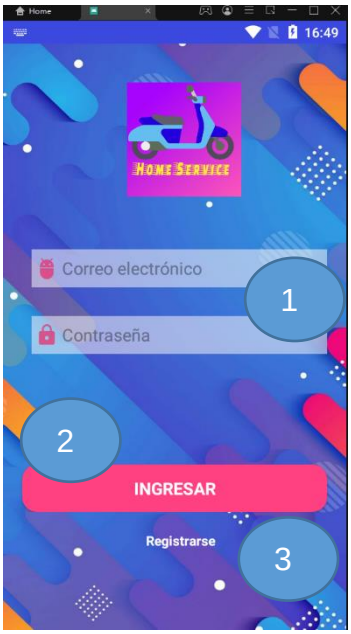
5. Módulo pedido

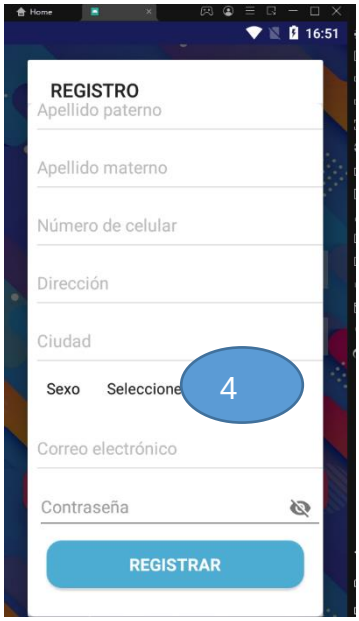


1. Una vez que pulse el botón pedir le lanza una ventana donde debe poner un comentario si gusta o puede dejarlo vacío, también debe seleccionar si quiere usar la dirección con que se registró o seleccionar usar otra dirección para introducir otra dirección.

2. Después pulsar sobre la opción enviar para finalizar su compra.

6. Login y registro





1. Para iniciar sesión deberá introducir el correo y la contraseña con lo que se registró.

2. Pulsar sobre botón ingresar si ya tiene una cuenta en la aplicación.

3. Si no tiene una cuenta en la aplicación pulsar en registrarse.

4. Llenar los campos vacíos y finalizar pulsando en el botón registrarse.

ANEXO E:
Manual Técnico

Manual Técnico

SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE

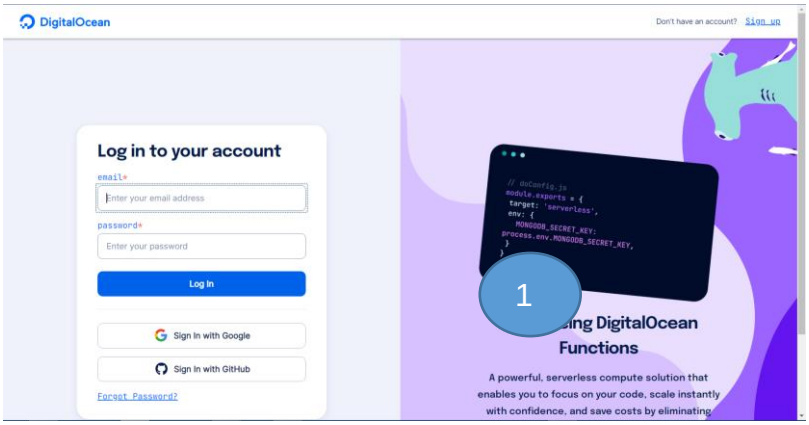
Autor: Jaime Calamani Mamani

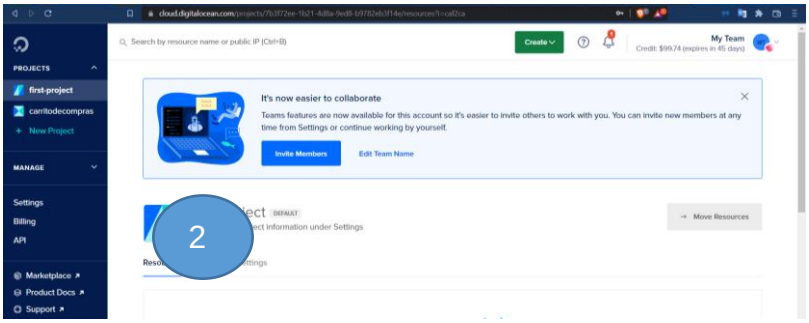
La Paz - Bolivia

2022

Manual técnico

1. Abrir una cuenta en digital ocean

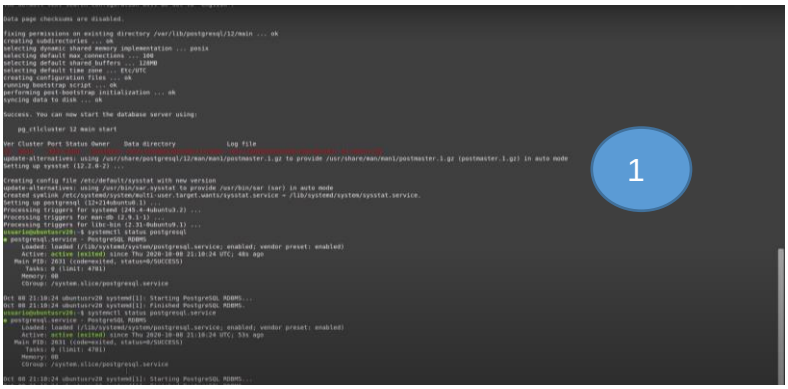




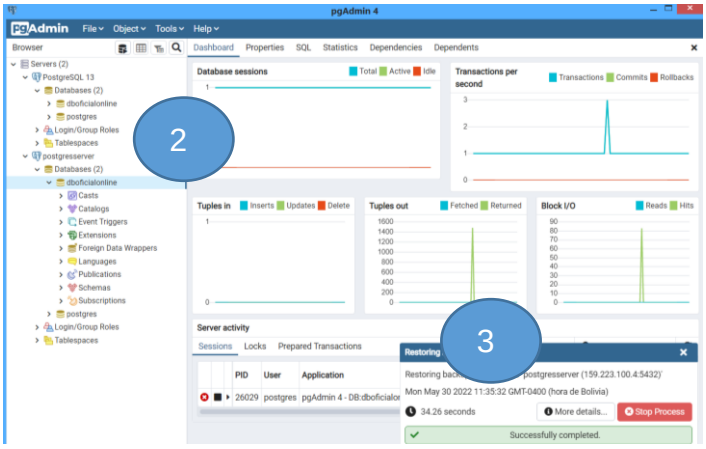
1. Entrar a la página de digital ocean, registrarse con tu correo, poner una cuenta de banco.

2. Una vez dentro crear nuevo proyecto, crear un droplet para instalar lo necesario para tu backend y frontend.

2. Instalar postgreSQL



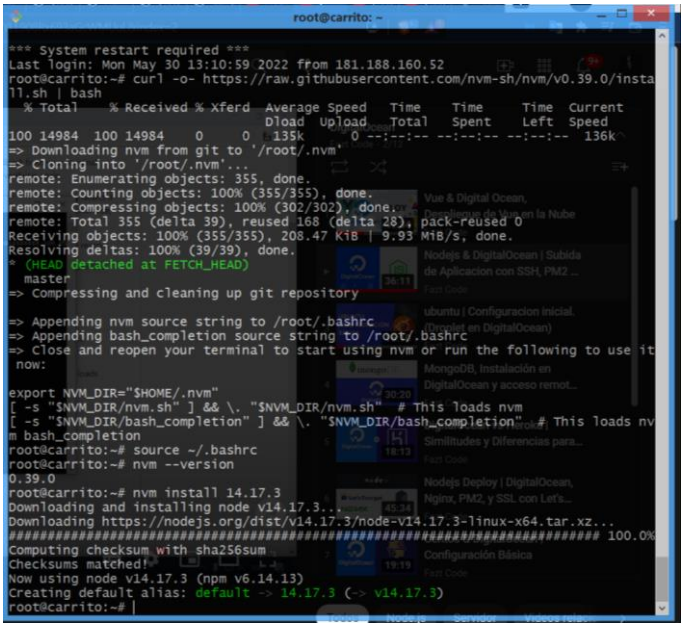
1. Seguir la guía de instalación en la página oficial de postgresql para no tener errores, muchas veces si cometes un error da problemas.



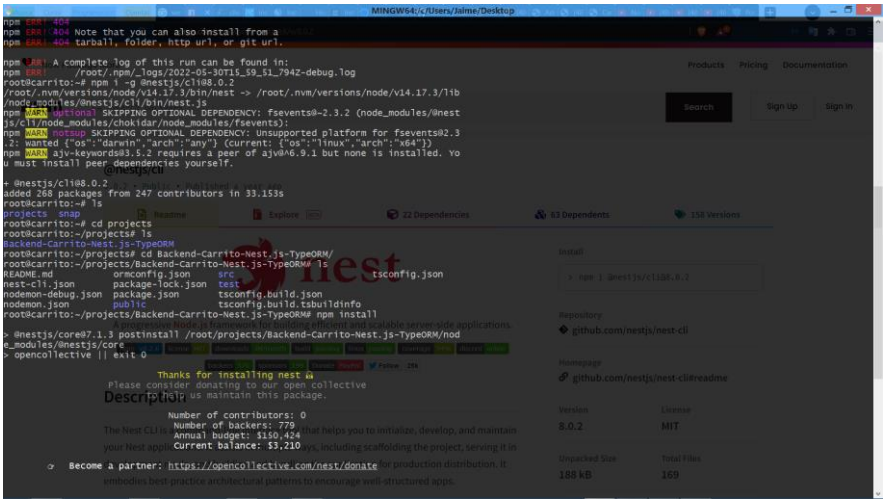
2. Para conectarse de tu máquina local de esta manera subir la base de datos al sevidor crear un server nuevo y poner la ip de digital ocean, como también la contraseña de la base de datos.

3. Pulsar sobre tu base de datos con click derecho donde muestra la opción de backup, buscar un lugar, poner el nombre el backup, ir al nuevo servidor crear nueva base de datos y restaurar el backup.

3. Instalar Node.js en el servidor



4. Desplegar proyecto Nest.js

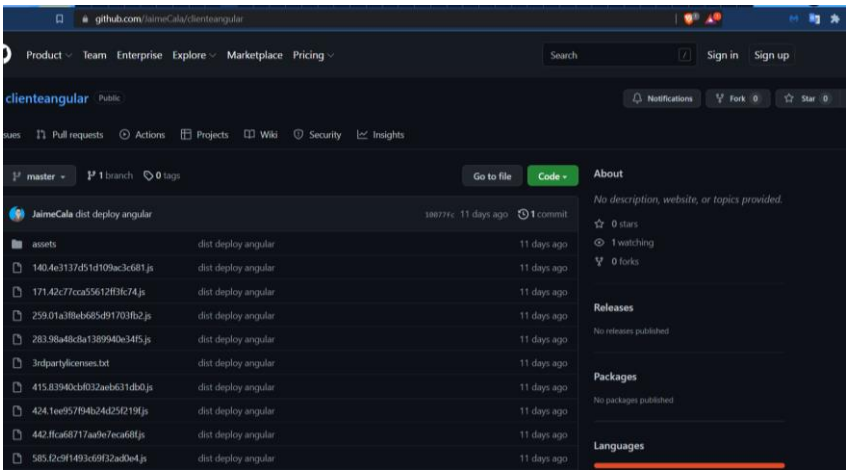


5. Instalar Nginx para desplegar frontend

```
$ sudo apt update
$ sudo apt install nginx
```

Copy

6. Subir a Github para desplegar web



MICROMARKET

HOME SERVICE

El Alto, noviembre de 2022

Señor:
Ing. Enrique Flores Baltazar
TUTOR METODOLÓGICO TALLER DE GRADO II
Presente. –

REF.: AVAL DE IMPLEMENTACIÓN

Distinguido ingeniero:

Mediante la presente tengo a bien comunicarle mi conformidad del proyecto de grado realizado, titulado "**SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE" CASO: Micromarket Home Service,** por parte del Universitario Jaime Calamani Mamani, con cédula de identidad 7033686 LP. y registro universitario 10036096, de haber realizado e implementado satisfactoriamente con éxito la aplicación móvil y sistema web mencionado.

Sin otro particular, me despido de usted.
Atentamente.

MICROMARKET
HOME SERVICE
B-MARZA
CEL: 73245730

Braulio Marza Mamani
Supervisor Micromarket Home Service

AVAL DE CONFORMIDAD

El Alto, 21 de noviembre del 2022

Señor
M. Sc. Ing. David Carlos Mamani Quispe
DIRECTOR DE LA CARRERA DE INGENIERIA DE SISTEMAS

Presente. –

REF. AVAL DE CONFORMIDAD

Distinguido Ingeniero:

Mediante la presente tengo a bien comunicarle mi conformidad del proyecto de grado "**SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE" CASO: Micromarket Home Service,** que propone el postulante Universitario **Jaime Calamani Mamani** con cédula de identidad **7033686 LP.,** para su defensa pública y evaluación correspondiente a la materia de Taller de Grado II, de acuerdo al reglamento vigente de la carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, reciba saludos cordiales.

Atentamente,

M. Sc. Ing. Enrique Flores Baltazar
TUTOR METODOLÓGICO

AVAL DE CONFORMIDAD

El Alto, 21 de noviembre del 2022

Señor
M. Sc. Ing. Enrique Flores Baltazar
TUTOR METODOLÓGICO TALLER DE GRADO II

Presente.

REF.: AVAL DE CONFORMIDAD

Distinguido Ingeniero:

Mediante la presente tengo a bien comunicarle mi conformidad del proyecto de grado "**SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE" CASO: Micromarket Home Service**, que propone el postulante Universitario **Jaime Calamani Mamani** con cédula de identidad **7033686 LP.,** para su defensa pública y evaluación correspondiente a la materia de Taller de Grado II, de acuerdo al reglamento vigente de la carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, reciba saludos cordiales.

Atentamente.

Lic. Maria Magdalena Aguilar Guanto
TUTOR REVISOR

AVAL DE CONFORMIDAD

El Alto, 21 de noviembre del 2022

Señor
M. Sc. Ing. Enrique Flores Baltazar
TUTOR METODOLÓGICO TALLER DE GRADO II

Presente.

REF.: AVAL DE CONFORMIDAD

Distinguido Ingeniero:

Mediante la presente tengo a bien comunicarle mi conformidad del proyecto de grado "**SISTEMA DE INFORMACIÓN WEB Y APLICACIÓN MÓVIL PARA LA ADMINISTRACIÓN DE INVENTARIO, VENTAS Y COMPRAS ONLINE" CASO: Micromarket Home Service**, que propone el postulante Universitario **Jaime Calamani Mamani** con cédula de identidad **7033686 LP.**, para su defensa pública y evaluación correspondiente a la materia de Taller de Grado II, de acuerdo al reglamento vigente de la carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, reciba saludos cordiales.

Atentamente.

Ing. Hernan Quispe Conurana
TUTOR ESPECIALISTA