Universidad Nacional de Ingeniería

Área de Conocimiento de Tecnología de la Información y Comunicación

# Desarrollo de sistema E-Commerce para tienda MombaShop en el departamento de managua

## Trabajo Monográfico para optar al título de Ingeniero de Sistemas


**Elaborado por:**

Br. Fidel Francisco Hernandez Ruiz
Carnet: 2017-0834i

Br. Gerald Antonio Solano Macias
Carnet: 2017-0669i

Br. José Fernando Torres González
Carnet: 2017-0628i

**Tutor:**

MSc. Ing. Mario Jose Selva Mendoza



31 de octubre de 2025
Managua, Nicaragua

Secretaria Académica
DACTIC

## SECRETARIA DE ÁREA ACADÉMICA

## F-8:CARTA DE FINALIZADO PLAN DE ASIGNATURA

El Suscrito Secretario del **ÁREA DE CONOCIMIENTO DE TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN** hace constar que:

**TORRES GONZÁLEZ JOSÉ FERNANDO**

Carné: **2017-0628I** Turno: **Diurno** Plan de Asignatura: **2015** de conformidad con el Reglamento Académico vigente en la Universidad, ha aprobado todas las asignaturas correspondientes a la carrera de **INGENIERÍA DE SISTEMAS**, en el año 2022 y solo tiene pendiente la realización de una de las formas de culminación de estudio.

Se extiende la presente **CARTA DE FINALIZADO PLAN DE ASIGNATURA**, a solicitud del interesado en la ciudad de Managua, a los quince días del mes de octubre del año dos mil veinte y cinco.

**Atentamente,**

SECRETARÍA ACADÉMICA DE DACTIC

MSc. Luisa Massiel Mercado Gutiérrez
**SECRETARIO DE ÁREA ACADÉMICA**

Móvil: (505) 83803517

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

hazzely.orozco@dactic.uni.edu.ni
www.uni.edu.ni

Secretaria Académica
DACTIC

**SECRETARIA DE ÁREA ACADÉMICA**

## F-8:CARTA DE FINALIZADO PLAN DE ASIGNATURA

El Suscrito Secretario del **ÁREA DE CONOCIMIENTO DE TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN** hace constar que:

**SOLANO MACIAS GERALD ANTONIO**

Carné: **2017-0669I** Turno: **Diurno** Plan de Asignatura: **2015** de conformidad con el Reglamento Académico vigente en la Universidad, ha aprobado todas las asignaturas correspondientes a la carrera de **INGENIERÍA DE SISTEMAS**, en el año 2022 y solo tiene pendiente la realización de una de las formas de culminación de estudio.

Se extiende la presente **CARTA DE FINALIZADO PLAN DE ASIGNATURA**, a solicitud del interesado en la ciudad de Managua, a los quince días del mes de octubre del año dos mil veinte y cinco.

**Atentamente,**

SECRETARIA ACADÉMICA DE DACTIC

Luisa Mercado

MSc. Luisa Massiel Mercado Gutiérrez
**SECRETARIO DE ÁREA ACADÉMICA**

Móvil: (505) 83803517

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

hazzely.orozco@dactic.uni.edu.ni
www.uni.edu.ni

Secretaria Académica
DACTIC

**SECRETARIA DE ÁREA ACADÉMICA**

## F-8:CARTA DE FINALIZADO PLAN DE ASIGNATURA

El Suscrito Secretario del **ÁREA DE CONOCIMIENTO DE TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN** hace constar que:

**HERNANDEZ RUIZ FIDEL FRANCISCO**

Carné: **2017-0834I** Turno: **Diurno** Plan de Asignatura: **2015** de conformidad con el Reglamento Académico vigente en la Universidad, ha aprobado todas las asignaturas correspondientes a la carrera de **INGENIERÍA DE SISTEMAS**, en el año 2022 y solo tiene pendiente la realización de una de las formas de culminación de estudio.

Se extiende la presente **CARTA DE FINALIZADO PLAN DE ASIGNATURA**, a solicitud del interesado en la ciudad de Managua, a los quince días del mes de octubre del año dos mil veinte y cinco.

**Atentamente,**

SECRETARÍA ACADÉMICA DE DACTIC

Luisa Mercado

MSc. Luisa Massiel Mercado Gutiérrez
**SECRETARIO DE ÁREA ACADÉMICA**

Móvil: (505) 83803517

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

hazzely.orozco@dactic.uni.edu.ni
www.uni.edu.ni

Miércoles, 29 de Octubre del 2025

MSc. Claudia Lucía Benavides Rugama

Decana – Facultad de Ciencias y Sistemas

Su despacho:

Estimada Maestra Benavides, reciba mis más altas muestras de consideración y estima.

A través de la presente doy fe de haber acompañado la elaboración del estudio monográfico titulado: **"Desarrollo de sistema e-commerce para tienda MombaShop en el departamento de Managua"**, elaborado por:

- Br. Fidel Francisco Hernández Ruiz. 2017-0834i
- Br. Gerald Antonio Solano Macias. 2017-0669i
- Br. José Fernando Torres González. 2017-0628i

Para que de acuerdo a la normativa de culminación de estudios y el reglamento académico se proceda a entrega de la monografía y se solicite su autorización para la programación de la defensa.

Sin más a que referirme, y deseándole éxito en sus funciones académicas me despido.

Cordialmente,

MSc. Ing. Mario José Selva Mendoza

Docente Titular UNI

Área de Conocimiento de
Tecnología de la Información
y Comunicación

***Managua, 13 de mayo 2024***

***Br. Fidel Francisco Hernández Ruiz***
***Br. Gerald Antonio Solano Macias***
***Br. José Fernando Torres González***
*Egresados Programa académico Ingeniería de Sistemas*
*Sus manos.-*

*Estimados Egresados:*

*Reciban cordiales saludos y éxito en sus actividades.*

*Por medio de la presente, les comunico la aprobación del Protocolo de trabajo monográfico titulado:* ***"Desarrollo de sistema E-Commerce para tienda Mombashop en el departamento de Managua",*** *el cual cumple con los requisitos establecidos en el capítulo II de la normativa de trabajos monográficos de la UNI como forma de culminación de estudios, por lo que queda oficialmente aprobado por esta Dirección.*

*El docente responsable de acompañarle en el proceso de desarrollo de su tema monográfico es el Msc. Mario José Selva Mendoza.*

*A partir de la fecha de aprobación del protocolo monográfico, tienen un máximo de doce meses para presentar los documentos correspondientes para la coordinación del proceso de pre defensa.*

*Atentamente,*

UNIVERSIDAD NACIONAL DE INGENIERÍA · REPÚBLICA DE NICARAGUA · AMERICA CENTRAL · S. O. F.

***Msc. Claudia Lucía Benavidez Rugama***
***Directora Área de Conocimiento de***
***Tecnología de la Información y Comunicación***

*Cc. Msc. Mario José Selva Mendoza – Tutor*
*Archivo DACTIC.*

Móvil: (505) 8588 8333

Recinto Universitario Simón Bolivar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

www.uni.edu.ni

## DEDICATORIA

Dedicamos este trabajo a Dios, fuente de nuestra fortaleza y guía constante a lo largo de este camino académico. A Él le debemos la paciencia, la sabiduría y la determinación para enfrentar cada desafío y alcanzar esta meta compartida.

Asimismo, lo dedicamos a nuestras madres y padres, quienes con su amor incondicional, apoyo constante y ejemplo de vida han sido la base de nuestro crecimiento personal y académico. A nuestras abuelas, por sus consejos y oraciones silenciosas que nos han sostenido; a nuestros hermanos, por su compañía y motivación diaria; y a toda nuestra familia extendida, cuyo cariño y respaldo han sido fundamentales en cada paso de esta trayectoria.

Finalmente, dedicamos este logro a nuestro tutor, MSc. Ing. Mario José Selva Mendoza, cuya guía, paciencia y conocimientos nos han acompañado en cada etapa del proceso, fortaleciendo nuestro aprendizaje y contribuyendo al éxito de este trabajo.

A todos ustedes, con profundo respeto y gratitud, dedicamos este esfuerzo, conscientes de que su apoyo ha sido esencial para llegar hasta aquí.

**RESUMEN**

La presente investigación tiene como objetivo principal el diseño e implementación de un sistema E-commerce para tienda MombaShop, con el propósito de fortalecer su presencia digital y mejorar la experiencia de compra de los usuarios. Para ello, se establecieron etapas estratégicas que comprendieron el análisis del modelo de negocio, la evaluación de la viabilidad del proyecto, el diseño técnico del sistema y su implementación mediante tecnologías modernas.

El proceso incluyó un estudio detallado de los procesos comerciales clave, así como la identificación de requerimientos funcionales y no funcionales relacionados con la gestión de productos, procesos de compra, necesidades del usuario y administración interna. Asimismo, se efectuó una evaluación de viabilidad técnica, económica, operativa y legal, con el fin de asegurar la sostenibilidad del sistema y su conformidad con las normativas vigentes.

El sistema fue diseñado utilizando la metodología UWE (UML-based Web Engineering), y desarrollado con tecnologías basadas en JavaScript y MySQL como gestor de base de datos. Los resultados obtenidos muestran que el desarrollo e implementación de la plataforma de comercio electrónico en la tienda Mombashop no solo optimiza los procesos comerciales y la gestión de inventario, sino que también mejora significativamente la experiencia de compra y el servicio ofrecido a los clientes.

Este proyecto aporta una solución digital replicable y adaptable a otros negocios del sector minorista (retail), resaltando la importancia de las tecnologías modernas y la digitalización en el avance y la competitividad de las microempresas.

INDICE

## Índice de tablas y figuras.
## Figuras.

## Tablas.

# I. INTRODUCCION

Con el presente trabajo se pretendió que la tienda MombaShop cuente con un sistema de información web que responda a las necesidades actuales en materia de gestión del comercio electrónico, permitiendo fortalecer su presencia digital y mejorar la experiencia de compra de sus clientes. Este sistema busca optimizar los procesos internos de la tienda, tales como la gestión de productos, el control de pedidos, la atención al cliente y la administración operativa en general.

Entre las necesidades principales que motivaron el desarrollo de este sistema se encuentran:

1. La falta de una plataforma digital que permita a los clientes realizar compras en línea de forma continua y segura.

2. La necesidad de modernizar y automatizar procesos comerciales clave, reduciendo la carga operativa manual.

3. La carencia de herramientas que permitan analizar el comportamiento de los usuarios y facilitar la toma de decisiones estratégicas.

Para abordar estas necesidades, se aplicó la metodología de desarrollo UWE (UML-based Web Engineering) como base para el diseño del sistema, y se utilizaron tecnologías actuales como Node.js, React, Sequelize y MySQL en el proceso de implementación.

Asimismo, se siguió un enfoque sistemático que incluyó el análisis del modelo de negocio de la tienda, la identificación de requerimientos funcionales y no funcionales, así como una evaluación de viabilidad técnica, operativa, económica y legal. Este enfoque permitió definir una solución alineada con los objetivos de MombaShop y ajustada a las buenas prácticas en el desarrollo de aplicaciones web modernas.

## II. ANTECEDENTES

En Nicaragua, a partir del año 2019, se ha evidenciado un incremento sostenido en la adopción de soluciones de comercio electrónico por parte de pequeñas y medianas empresas (PYMES). Este fenómeno se vio impulsado por las restricciones derivadas de la pandemia por COVID-19, las cuales motivaron a los negocios a incorporar herramientas digitales con el propósito de ampliar su alcance en el mercado, mejorar la eficiencia operativa y mantener la continuidad de sus actividades económicas.

El surgimiento y consolidación de plataformas internacionales como **Shopify (2006)** y **WooCommerce (2011)**, así como la expansión de los servicios de entrega a domicilio en el país desde el año 2018, han contribuido significativamente a la digitalización del sector comercial. Estas herramientas ofrecen funcionalidades esenciales, entre ellas la gestión de inventario, control de pedidos y seguimiento de clientes, que han permitido a diversas empresas fortalecer su competitividad y optimizar la relación con los consumidores.

En este contexto, la tienda **MombaShop**, dedicada a la comercialización de artículos de moda y accesorios, representa un caso característico de las micro y pequeñas empresas que buscan adaptarse a las nuevas tendencias digitales. Desde sus inicios, la gestión operativa de la empresa se ha desarrollado de manera manual, utilizando principalmente hojas de cálculo para el registro y control de inventarios, ventas y pedidos.

A pesar del reconocimiento del potencial del comercio electrónico, se identificaron limitaciones técnicas y operativas que dificultaron la implementación de plataformas como WooCommerce o Shopify. Entre dichas limitaciones destacan la falta de personal especializado, la ausencia de infraestructura tecnológica adecuada y las dificultades de integración con los servicios logísticos locales. Estas condiciones han restringido el control de los procesos comerciales, afectando la eficiencia administrativa y la capacidad de respuesta ante las demandas del mercado.

La situación descrita evidencia la necesidad de contar con un sistema de comercio electrónico diseñado conforme a las particularidades de MombaShop, que permita presencia digital, automatizar la gestión de productos, pedidos y clientes.

## III. PLANTEAMIENTO DE LA SITUACION PROBLEMÁTICA

El comercio electrónico se ha consolidado como una herramienta esencial para el fortalecimiento y competitividad de las pequeñas y medianas empresas, al facilitar la expansión de su alcance comercial, la optimización de procesos internos y la mejora en la atención al cliente. No obstante, en Nicaragua, la adopción de estas soluciones tecnológicas continúa siendo limitada debido a factores como la falta de infraestructura digital, el escaso conocimiento técnico y la dificultad para adaptar las operaciones empresariales a plataformas estandarizadas de gestión comercial.

En este contexto, la tienda **MombaShop**, dedicada a la comercialización de productos en el mercado local, presenta un modelo de gestión basado en procedimientos manuales para el control de inventarios, registro de pedidos y atención a los clientes. Estas actividades se ejecutan mediante herramientas ofimáticas básicas, principalmente hojas de cálculo, las cuales resultan ineficientes ante el incremento de la demanda y el crecimiento de la información. Esta situación genera retrasos en los procesos, errores en el manejo de datos y dificultades para obtener información precisa que respalde la toma de decisiones administrativas.

Asimismo, se ha identificado que las plataformas comerciales disponibles en el mercado, como **WooCommerce** y **Shopify**, no se ajustan completamente a las necesidades y particularidades operativas de MombaShop, principalmente por limitaciones de personalización, costos asociados y falta de compatibilidad con los procesos internos de la empresa. La ausencia de una herramienta tecnológica adaptada a su contexto ha restringido la capacidad de la organización para modernizar sus operaciones y consolidar su presencia en el entorno digital.

Por lo tanto, se evidencia la necesidad de **desarrollar una solución tecnológica personalizada** que permita a MombaShop automatizar la gestión de inventarios, pedidos y atención al cliente, optimizar sus recursos y fortalecer su competitividad mediante la implementación de un sistema de comercio electrónico propio y funcional.

## IV. JUSTIFICACION

La incorporación de sistemas de información basados en la web constituye una estrategia fundamental para mejorar la eficiencia operativa y la competitividad de las pequeñas y medianas empresas, en un contexto económico cada vez más orientado hacia el comercio electrónico y la digitalización de los procesos empresariales. En este sentido, la tienda MombaShop enfrenta limitaciones asociadas a la gestión manual de sus operaciones, lo que ha generado retrasos en la atención al cliente, errores en el control de inventarios y dificultades para mantener un registro confiable de los pedidos realizados.

El desarrollo de un sistema de información web personalizado permitirá automatizar y centralizar la gestión de los procesos comerciales, mejorando la organización interna y optimizando el uso de los recursos disponibles. La implementación de esta solución tecnológica aportará mayor control sobre la información, reducción de errores operativos, disponibilidad permanente de los servicios y una mejor experiencia de compra para los clientes, lo que contribuirá directamente al fortalecimiento de la competitividad de la empresa en el entorno digital.

Desde una perspectiva económica y social, el proyecto fomentará la adopción de herramientas tecnológicas en pequeñas empresas locales, promoviendo la modernización de los modelos de negocio tradicionales y generando oportunidades de crecimiento en el comercio electrónico nicaragüense.

En el ámbito tecnológico, la propuesta integra metodologías de ingeniería de software orientadas a la web, como UWE (UML-based Web Engineering), y tecnologías modernas basadas en JavaScript, Node.js, React y bases de datos relacionales, lo que garantiza un desarrollo estructurado, escalable y alineado con las tendencias actuales de la industria.

## V. OBJETIVOS

**Objetivo general:**

- Desarrollar sistema E-commerce para la gestión de comercio electrónico en la tienda MombaShop.

**Objetivos específicos:**

- Analizar la viabilidad técnica, operativa, económica y legal del desarrollo del sistema de información web, con el fin de definir los requerimientos funcionales y no funcionales a partir del modelo de negocio de la tienda MombaShop.

- Diseñar el sistema de información utilizando la metodología UWE.

- Codificar el sistema de comercio electrónico utilizando tecnologías basadas en JavaScript y MySQL como motor de base de datos.

# VI. MARCO TEORICO

En esta sección se desarrolla la fundamentación teórica que sustenta la investigación. Se exponen los conceptos centrales, se revisan los enfoques teóricos pertinentes y se analizan los antecedentes relevantes que definen el contexto del objeto de estudio. Esta revisión proporciona la base conceptual sobre la cual se construye el presente trabajo.

## 6.1. Sistema de Información

Los sistemas de información constituyen la base tecnológica sobre la cual las organizaciones gestionan, procesan y transforman datos en conocimiento útil para la toma de decisiones. De acuerdo con Laudon y Laudon (1996), un sistema de información está conformado por componentes interrelacionados que permiten capturar, almacenar, procesar y distribuir información para apoyar las actividades de control, análisis y gestión dentro de una organización.

En el contexto de la tienda MombaShop, la implementación de un sistema de información orientado al comercio electrónico busca sustituir los procesos manuales existentes, permitiendo una gestión más eficiente de inventarios, pedidos y atención al cliente.

## 6.2. Comercio Electrónico (E-Commerce)

El comercio electrónico ha transformado los modelos de negocio tradicionales al trasladar transacciones y procesos comerciales hacia entornos digitales. Según Somalo (2017), el e-commerce implica la digitalización de operaciones comerciales a través de redes de telecomunicación, abarcando actividades como compra y venta de productos, atención al cliente y control logístico.

En el caso de las PYMES nicaragüenses, el comercio electrónico representa una oportunidad para ampliar su mercado y optimizar recursos, aunque enfrenta desafíos relacionados con infraestructura tecnológica y adaptación operativa.

En este marco, el sistema propuesto para MombaShop busca aprovechar los beneficios del e-commerce mediante el diseño de una plataforma ajustada a sus necesidades específicas.

### 6.3. **Ingeniería de Requerimientos en proyectos de software**

La identificación y documentación de requerimientos constituye una fase esencial en la ingeniería de software, ya que define las funciones y características que debe cumplir el sistema para satisfacer las necesidades del usuario (Wiegers, 1999).

Los requerimientos funcionales describen las acciones que el sistema debe ejecutar, tales como la gestión de productos, control de pedidos y administración de usuarios, mientras que los requerimientos no funcionales establecen criterios de desempeño, seguridad y usabilidad (Sommerville, 2011; Pressman, 2010).

En el desarrollo del sistema para MombaShop, ambos tipos de requerimientos son determinantes para garantizar una solución escalable, confiable y adaptable a las condiciones del mercado.

### 6.4. **Estudios de Viabilidad Técnica**

La evaluación de la viabilidad permite determinar la factibilidad de implementación del sistema desde diferentes perspectivas: técnica, económica, operativa y legal (Gómez, 2001). La viabilidad técnica analiza la disponibilidad de recursos tecnológicos y humanos necesarios para el desarrollo; la viabilidad económica, por su parte, examina si los beneficios superan los costos asociados; mientras que la viabilidad operativa y legal valoran la aceptación del sistema y su cumplimiento normativo.

Para la estimación económica se considera el modelo COCOMO II, propuesto por Boehm y citado por Pressman (2010), el cual permite cuantificar el esfuerzo, tiempo y costo del desarrollo a partir del tamaño del software y factores de complejidad. Este modelo ofrece una base teórica sólida para planificar los recursos del sistema de comercio electrónico propuesto.

## 6.5. **COCOMO II**

En el desarrollo de sistemas de software, uno de los modelos más utilizados para estimar el esfuerzo, tiempo y costos asociados es el modelo **COCOMO II** (Constructive Cost Model II).

Este modelo, propuesto por Barry Boehm y ampliamente adoptado en la ingeniería de software, permite estimar de forma cuantitativa los recursos necesarios en función del tamaño del sistema, expresado en líneas de código fuente o puntos de función.

Según Pressman (2010), COCOMO II es una evolución del modelo original que incluye factores de escala y multiplicadores de esfuerzo ajustados a los entornos modernos de desarrollo, incluyendo metodologías iterativas y tecnologías orientadas a objetos.

COCOMO II se organiza en tres niveles de evaluación: **composición de aplicaciones**, **post-arquitectura** y **próximas a la entrega**. Para sistemas de complejidad media o alta, como un eCommerce con funcionalidades administrativas y de gestión de ventas, el modelo post-arquitectura resulta el más apropiado, ya que considera que se cuenta con un diseño preliminar del sistema, una estructura de componentes y una base definida de requisitos. A través de este modelo, es posible calcular el esfuerzo estimado (en persona-mes) y, posteriormente, determinar los costos a partir de los salarios promedio del equipo de desarrollo.

Al aplicar este modelo, se incorporan factores como la confiabilidad requerida del software, la complejidad del producto, la experiencia del equipo de desarrollo y las herramientas utilizadas. Wiegers (1999) enfatiza que el uso de modelos de estimación como COCOMO II no solo mejora la precisión en la planificación, sino que también facilita la toma de decisiones gerenciales, al ofrecer una visión objetiva de los recursos necesarios y de la factibilidad económica del sistema.

### 6.6. **UML basado en Ingeniería Web (UWE)**

La metodología **UWE (UML-based Web Engineering)** constituye un marco estructurado para el diseño de aplicaciones web mediante modelos que representan la navegación, el contenido, los procesos y la presentación de la información (Koch, Kraus & Hennicker, 2008). Su enfoque orientado a modelos permite separar las preocupaciones de diseño, garantizando coherencia entre las vistas estructurales y funcionales del sistema.

Entre sus principales componentes se incluyen los diagramas de casos de uso, que describen las interacciones entre actores y funciones del sistema; los diagramas de navegación, que muestran la relación entre las interfaces de usuario; los diagramas de contenido, que estructuran la información presentada; y los diagramas de actividades, que modelan los flujos de procesos internos.

La adopción de esta metodología en el proyecto de MombaShop posibilita un desarrollo ordenado y documentado, asegurando la trazabilidad entre los requerimientos definidos y los componentes implementados.

### 6.7. **Diagrama de Casos de Uso**

Los casos de uso describen secuencias de acciones que generan resultados observables para un actor del sistema. En UWE, se emplean estereotipos como «browsing» y «processing» para indicar si un caso de uso modifica datos persistentes (Koch et al., 2008).

### 6.8. **Diagrama de Actividades**

El diagrama de actividades es una herramienta de modelado UML que permite representar flujos de trabajo y procesos de negocio mediante símbolos y reglas formales. Es especialmente útil para describir comportamientos dinámicos dentro del sistema (Booch, Rumbaugh & Jacobson, 2005).

### 6.9. **Diagrama de Navegación**

Este diagrama representa la interacción entre las diferentes vistas de una aplicación web, permitiendo visualizar cómo navega el usuario entre interfaces o componentes, así como los eventos que desencadenan dichas transiciones (Koch et al., 2008).

### 6.10. **Diagrama de Contenido**

El diagrama de contenido modela la organización de la información y los datos que conforman las vistas del sistema web. Es útil para estructurar los elementos informativos que estarán disponibles para los usuarios.

### 6.11. **Tecnologías del Sistema**

### 6.12. **JavaScript**

JavaScript es un lenguaje de programación interpretado, orientado a objetos y ampliamente utilizado en el desarrollo de aplicaciones web. Su integración nativa con HTML y CSS permite la creación de interfaces interactivas y dinámicas, así como la gestión de eventos del lado del cliente (Flanagan, 2020). En el sistema desarrollado para MombaShop, JavaScript constituye la base para la implementación de funcionalidades dinámicas en el frontend y la interacción con componentes del backend.

### 6.13. **Frameworks**

Un framework es un conjunto de herramientas y convenciones que proporciona una estructura reutilizable para el desarrollo de software, facilitando la organización del código y la implementación de buenas prácticas. Su utilización permite a los desarrolladores concentrarse en la lógica de negocio y reducir el esfuerzo asociado a tareas repetitivas o de bajo nivel (Fowler, 2002). En este proyecto, se emplean frameworks modernos de JavaScript para garantizar la escalabilidad, mantenibilidad y consistencia del sistema.

### 6.14. **Node.js**

Node.js es un entorno de ejecución basado en JavaScript que posibilita la construcción de aplicaciones del lado del servidor. Su arquitectura orientada a eventos y modelo de ejecución no bloqueante favorecen el desarrollo de aplicaciones escalables y de alto rendimiento (Tilkov & Vinoski, 2010). En el sistema propuesto, Node.js se utiliza para manejar la lógica del backend, la gestión de rutas, la comunicación con la base de datos y la integración con APIs externas, asegurando eficiencia y capacidad de respuesta.

### 6.15. **React**

React es una biblioteca de JavaScript para la construcción de interfaces de usuario mediante componentes reutilizables y reactivos. Permite desarrollar aplicaciones web dinámicas con una actualización eficiente del DOM virtual, optimizando el rendimiento y la experiencia del usuario (Banks & Porcello, 2017). En MombaShop, React se emplea para estructurar el frontend, garantizando interactividad, modularidad y mantenimiento simplificado de la interfaz.

### 6.16. **Sequelize**

Sequelize es un ORM (Object Relational Mapping) para Node.js que permite mapear modelos de datos a bases de datos relacionales de manera estructurada y segura. Facilita la gestión de consultas SQL mediante sintaxis en JavaScript, reduciendo errores en la manipulación directa de la base de datos y mejorando la mantenibilidad del código (Wilson, 2016). En el proyecto, Sequelize se emplea para gestionar los modelos de productos, pedidos y usuarios, asegurando consistencia y escalabilidad.

### 6.17. **MySQL**

MySQL es un sistema de gestión de bases de datos relacional de código abierto, reconocido por su fiabilidad, rapidez y facilidad de integración en aplicaciones web (DuBois, 2008). Constituye el repositorio principal de información del sistema, almacenando de manera estructurada los datos de inventario, pedidos y clientes, y garantizando la integridad y seguridad de la información.

### 6.18. **APIs REST**

Las APIs REST (Representational State Transfer) constituyen un estándar para el desarrollo de servicios web que permiten la comunicación entre aplicaciones a través de protocolos HTTP. Este enfoque facilita la integración de componentes distribuidos, el acceso a datos de manera estandarizada y la construcción de arquitecturas escalables (Richardson & Ruby, 2007). En MombaShop, las APIs REST permiten que el frontend y el backend se comuniquen de manera eficiente, garantizando la sincronización de datos y la interoperabilidad entre módulos del sistema.

# VII. CAPITULO I: ANALISIS DE LA SITUACION ACTUAL DE LA TIENDA

## 7.1. Generalidades del Negocio

MombaShop se destaca como una tienda de ropa especializada en ofrecer una amplia gama de artículos para mujeres, hombres, niños y niñas de todas las edades. Operando bajo el modelo de negocio "business-to-consumer", la empresa se centra en la industria de la moda y se compromete a facilitar la experiencia de compra en línea.

## 7.2. Cultura Organizacional

### 7.2.1. Misión

Ofrecer a nuestros clientes productos de calidad, a precios competitivos que cumplan con sus necesidades y exigencia, abarcando su gusto de acuerdo con su estilo de ver y vivir la vida

### 7.2.2. Visión

Ser una empresa líder y reconocida proporcionado cada día más un servicio de excelencia a nuestros clientes y que al mismo tiempo nos permitan competir en el mercado nacional con los mejores precios.

### 7.2.3. **Valores**

Los valores que presentan para sostener una estrategia de crecimiento constante y diferenciación son:

**Tabla 1** *Valores de la tienda MombaShop*

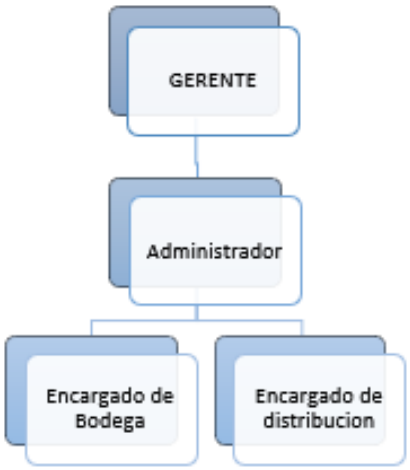
| # | Valores |
|---|---|
| 1 | Trabajo en Equipo |
| 2 | Colaboracion |
| 3 | Servicio |
| 4 | Innovación y mejora continua |

Fuente: Tienda MombaShop

## 7.3. **Estructura Organizacional**

En Mombashop se trabaja de lunes a viernes con horario de 8 am hasta 5 pm y los días sábados de 8 am hasta 12 pm. Este negocio cuenta con 4 personas que son los que llevan a cabo las operaciones de dicho local. Donde cada uno de ellos ejerce diferentes funciones en las operaciones.

**Figura 1** *Estructura Organizacional de MomabaShop*




### 7.3.1. **Fichas Ocupacionales**

**Tabla 2** *Ficha Ocupacional Gerente*

| **Categoría** | **Detalle** |
|---|---|
| **Cargo** | Gerente |
| **Descripción del cargo** | Responsable de la planificación, organización y supervisión general del ecommerce. Coordina las estrategias comerciales, gestiona recursos y toma decisiones orientadas al cumplimiento de los objetivos del negocio. |
| **Funciones** | - Supervisar la operación general del ecommerce.<br>- Establecer objetivos estratégicos y comerciales.<br>- Coordinar con las áreas de administración, logística y distribución.<br>- Evaluar el desempeño del personal y la rentabilidad del proyecto.<br>- Tomar decisiones administrativas y financieras. |
| **Requisitos** | Título universitario en Ingeniería en Sistemas, Administración de Empresas o carreras afines. Experiencia en dirección de proyectos |

| | |
|---|---|
| | digitales y comercio electrónico. Habilidades de liderazgo y toma de decisiones. |

Fuente: Elaboración Propia

**Tabla 3** *Ficha Ocupacional del Administrador*

| **Categoría** | **Detalle** |
|---|---|
| Cargo | Administrador |
| Descripción del cargo | Encargado de la gestión operativa del ecommerce, control de inventarios, atención al cliente y administración de los recursos internos. Supervisa las actividades diarias y asegura el correcto funcionamiento de la plataforma. |
| Funciones | - Gestionar el sistema de inventario y actualizar el catálogo de productos.<br>- Atender consultas de clientes y resolver incidencias.<br>- Coordinar con bodeguero y repartidor para el cumplimiento de entregas.<br>- Supervisar las órdenes de compra y facturación.<br>- Mantener actualizada la base de datos del ecommerce. |
| Requisitos | Estudios técnicos o universitarios en Administración, Sistemas o áreas afines. Experiencia en plataformas ecommerce y gestión de inventarios. Habilidades comunicativas y de organización. |

Fuente: Elaboración Propia

**Tabla 4** *Ficha Ocupacional del responsable de Bodega*

| Categoría | Detalle |
|---|---|
| **Cargo** | Responsible de Bodega |
| **Descripción del cargo** | Responsable de la recepción, almacenamiento y despacho de los productos comercializados en la tienda en línea. Controla el inventario físico y asegura el buen estado de los productos. |
| **Funciones** | - Recibir, verificar y almacenar la mercancía.<br>- Actualizar el inventario físico conforme a las ventas realizadas.<br>- Preparar pedidos para su envío conforme a las órdenes generadas.<br>- Realizar reportes periódicos del estado del almacén. |
| **Requisitos** | Educación secundaria completa. Conocimientos básicos de logística e inventario. Capacidad para el trabajo físico y organizado. Experiencia previa deseable en almacenamiento. |

Fuente: Elaboración Propia

**Tabla 5** *Ficha Ocupacional del Repartidor*

| Categoría | Detalle |
|---|---|
| **Cargo** | Repartidor |
| **Descripción del cargo** | Encargado de la distribución y entrega de pedidos a los clientes, garantizando puntualidad, integridad del producto y buena atención durante el proceso de entrega. |
| **Funciones** | - Verificar las rutas de entrega y los pedidos asignados.<br>- Entregar productos a domicilio conforme a los tiempos establecidos.<br>- Recolectar firmas o evidencias de entrega según protocolo. |

| | |
|---|---|
| | - Reportar inconvenientes o devoluciones al área administrativa.<br>- Mantener en buen estado el vehículo asignado. |
| **Requisitos** | Licencia de conducir vigente. Experiencia en entregas locales. Buen trato con clientes y responsabilidad en el manejo de mercancía. |

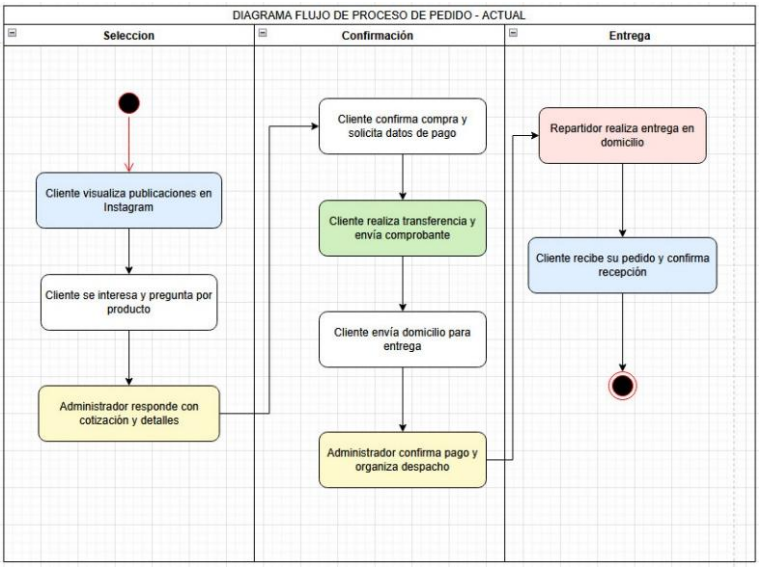
Fuente: Elaboración Propia

### 7.3.2. **Procesos Operativos**

A continuación, se describen los principales procesos operativos asociados al funcionamiento del sistema eCommerce propuesto. Estos procesos han sido identificados a partir del análisis funcional del negocio y la estructura de sus operaciones comerciales. La identificación y ordenamiento de estos procesos se realizó considerando la secuencia lógica del ciclo de compra y venta, lo cual ha sido fundamental para la definición de los requisitos del sistema web.

Con el propósito de mejorar la comprensión del flujo operativo, se presenta el diagrama que representa el proceso actual dentro del entorno del comercio electrónico. Este diagrama permite visualizar de forma estructurada las actividades clave involucradas en la gestión del sistema, tales como el registro de clientes, navegación por el catálogo de productos, conformación del carrito de compras, procesamiento de pedidos, emisión de facturas y recepción de pagos.

**Figura 2** *Diagrama de Flujo de Proceso actual – Gestión de pedidos*




**Fuente**: Elaboración Propia

El diagrama muestra el flujo de trabajo actual para procesar pedidos en la tienda MombaShop a través de Instagram, Meta y WhatsApp.

1. Inicia con la visualización de publicaciones por parte del cliente, quien se interesa en un producto y consulta al administrador.
2. Este responde con la cotización y detalles.
3. Luego, el cliente confirma la compra, realiza la transferencia, envía el comprobante y proporciona la dirección de entrega.
4. El administrador verifica el pago y organiza el despacho.
5. Finalmente, el repartidor entrega el pedido al cliente, quien confirma su recepción.

Nota: El diagrama está dividido en tres etapas: Selección, Confirmación y Entrega.

## 7.4. **Análisis de Viabilidad**

El estudio de viabilidad para la creación del sistema E-commerce para la tienda MombaShop será esencial para determinar la factibilidad y rentabilidad del proyecto. Este análisis permitirá evaluar la viabilidad técnica, económica, financiera y operativa del sistema propuesto, lo que ayudará a la empresa a tomar decisiones informadas sobre su implementación.

En última instancia, el estudio de viabilidad será clave para garantizar que la inversión en el sistema propuesto sea justificada y que contribuya positivamente al crecimiento y la eficiencia de MombaShop Nicaragua.

### 7.4.1. **Estudio Viabilidad Técnica**

En la presente sección se detalla el Estudio de Viabilidad Técnica del proyecto, cuyo objetivo es definir y analizar los componentes tecnológicos necesarios para la implementación de la plataforma de comercio electrónico. Este estudio valida que la solución propuesta es factible, escalable y robusta.

Para ello, este estudio parte de un análisis del equipamiento informático actual con el que cuenta Mombashop. A partir de esta base, se describe la arquitectura del sistema propuesta y se detallan las tecnologías y el stack de desarrollo (software) seleccionados. Posteriormente, se definen y justifican los servicios y requisitos de hardware recomendados (tanto para el servidor como para el cliente) que garantizarán un funcionamiento óptimo. Finalmente, se analizan los aspectos clave de seguridad y despliegue del proyecto.

#### 7.4.1.1. **Análisis del equipo de computo e infraestructura tecnológica actual**

Para establecer un punto de partida viable para la implementación del proyecto, se realizó un inventario de los recursos tecnológicos y el equipamiento informático existente en la tienda Mombashop. Este análisis permite identificar la infraestructura actual que servirá de base para las tareas administrativas de la nueva plataforma de comercio electrónico, así como determinar las carencias que deberán ser cubiertas mediante la adquisición o contratación de nuevos servicios y hardware.

**Tabla 6** *Equipo de Cómputo Actual*

| **Tipo de Equipo** | **Cantidad** | **Especificaciones (CPU/RAM/Almacenamiento)** | **Rol Asignado** |
|---|---|---|---|
| Computadora de Escritorio | 1 | Intel Core i3 (4ta Gen) / 4GB RAM / 500GB HDD | Punto de venta y administración |
| Laptop | 1 | Intel Core i5 (6ta Gen) / 8GB RAM / 256GB SSD | Gerencia / Uso móvil |
| Impresora | 1 | (Modelo) Térmica para facturas | Facturación |
| Router | 1 | (Modelo) Básico | Conexión de red |

Fuente: Elaboración Propia

**Tabla 7** *Inventario Software Actual*

| **Tipo de Software** | **Software Específico / Versión** | **Sistema Operativo** | **Propósito Actual** |
|---|---|---|---|
| Sistema Operativo | Windows 10 Home | N/A | Base en equipos H-01 y H-02 |
| Ofimática | Microsoft Office 2016 | N/A | Documentación general |
| Gestión Actual | **Hojas de cálculo (Excel)** | N/A | **Control de inventario y ventas** |
| Comunicación | WhatsApp Business | N/A | Atención al cliente y pedidos |

Fuente: Elaboración Propia

**Tabla 8** *Inventario de Conectividad y Red*

| **Servicio** | **Proveedor** | **Velocidad Contratada** | **Observaciones** |
|---|---|---|---|
| Internet (Banda Ancha) | Claro | 200 Mbs | Conexión principal de la tienda. |

Fuente: Elaboración Propia

El análisis del equipamiento actual revela que Mombashop opera con equipos de cómputo suficientes para las tareas administrativas de la nueva plataforma (gestión de pedidos, actualización de productos, atención al cliente). Sin embargo, se identifica una carencia total de infraestructura de servidor (hardware dedicado, IP pública fija) y una dependencia de métodos manuales (Hojas de cálculo) para la gestión del inventario.

Asimismo, la velocidad de conexión a internet puede ser una limitante para la gestión fluida de una plataforma web con alto volumen de imágenes. Este diagnóstico justifica la necesidad de contratar un servicio de hosting web externo y la mejora del plan de conectividad, tal como se detallará en las siguientes secciones

#### 7.4.1.2. **Definición de requisitos mínimos de software para que el sistema trabaje optimo.**

La infraestructura del servidor constituye el núcleo operativo de la plataforma de comercio electrónico. Para garantizar el rendimiento, la seguridad y la escalabilidad del sistema (definidos en los requisitos no funcionales), se establecen las siguientes especificaciones técnicas mínimas, tanto para el hardware del servidor como para el software que compone el stack tecnológico (backend y base de datos)

**Tabla 9** *Requerimientos de Hardware y Software*

| Componente | Especificación Mínima | Notas |
|---|---|---|
| **Hardware** | | |
| **Procesador** | Intel Core i5 (4 núcleos @ 2.5 GHz) o equivalente | Recomendado para manejar múltiples peticiones API. |
| **Memoria RAM** | 4 GB | 16 GB recomendados si el tráfico o el catálogo crecen. |
| **Almacenamiento** | 100 GB SSD | El uso de SSD es crítico para la velocidad de la base de datos. |
| **Software (Stack)** | | |
| **Sistema Operativo** | Ubuntu 20.04 LTS o superior | Versión estable y con soporte a largo plazo. |
| **Entorno de Ejecución** | Node.js v18 o superior | Para el backend y la API REST. |

| | | |
|---|---|---|
| **Base de Datos** | MySQL v8.0 o superior | Sistema de gestión de BBDD relacional. |
| **Servidor Web** | Nginx o Apache | Usado como proxy inverso para Node.js y servir archivos estáticos. |
| **Seguridad** | Certificado SSL/TLS | Mandatorio para habilitar HTTPS. |
| **Control de Versiones** | Git | Para el despliegue y mantenimiento del código. |

Fuente: Elaboración Propia

**Tabla 10** *Requisitos del Cliente para ejecutar el sistema Ecommerce*

| Componente | Especificación Mínima | Notas |
|---|---|---|
| **Hardware** | | |
| Dispositivos | PC, Laptop, Tablet, Smartphone | |
| Resolución (PC) | 1920x1080 (Recomendado) | |
| Resolución (Móvil) | 360x640 (Viewport Mínimo) | |
| **Software** | | |
| Navegadores | Google Chrome (v95+), Firefox (v90+), Safari (v14+), Edge (v91+) | |
| Conexión | 5 Mbps | Para una navegación fluida con carga de imágenes. |

Fuente: Elaboración Propia

El análisis del equipamiento informático actual evidencia que Mombashop opera con equipos de cómputo suficientes para la gestión administrativa, pero carece por completo de una infraestructura de servidor (hardware dedicado, almacenamiento redundante, conectividad de alta disponibilidad) capaz de soportar una plataforma de comercio electrónico.

La adquisición, configuración y mantenimiento de dicha infraestructura física (*on-premise*) no solo implicaría una elevada inversión de capital inicial, sino que también demandaría la gestión técnica especializada (administración de sistemas, seguridad, copias de seguridad) que, como se observó en la estructura organizacional, no forma parte de la planilla actual de la empresa.

Considerando estas barreras financieras y operativas, se concluye que la implementación de un servidor físico local no es una solución viable para Mombashop. Por lo tanto, este estudio propone una arquitectura de sistema desplegada íntegramente en servicios de Nube Pública, seleccionando Amazon Web Services (AWS). Este modelo (IaaS/PaaS) elimina la inversión inicial en hardware y delega la gestión de la infraestructura física al proveedor, alineándose con la capacidad operativa y financiera del negocio.

#### 7.4.1.3. **Alojamiento del Sistema (Cloud Hosting)**

Como se estableció en la justificación anterior, el modelo de implementación seleccionado es el de Computación en la Nube (Cloud Computing), específicamente bajo la modalidad de Infraestructura como Servicio (IaaS) y Plataforma como Servicio (PaaS). Este enfoque permite a Mombashop alquilar la capacidad computacional necesaria, pagando solo por el consumo y delegando la compleja administración del hardware físico.

El proveedor de servicios en la nube seleccionado para este proyecto es Amazon Web Services (AWS), dada su madurez en el mercado, su robusto portafolio de servicios y su escalabilidad.

A continuación, se detallan los servicios específicos de AWS recomendados para desplegar la arquitectura de tres capas (presentación, aplicación y datos) del sistema de comercio electrónico.

**Tabla 11** *Descripción de Servicios utilizados para la infraestructura*

| **Servicio AWS** | **Función principal** | **Costo (USD/mes)** |
|---|---|---|
| Amazon EC2 (t2.medium) | Backend y Base de Datos | 33.00 |
| Amazon S3 (Simple Storage Service) | Almacenamiento de archivos | 1.25 |
| Amazon CloudFront (CDN) | Distribución de contenido | 4.50 |
| Amazon SES | Correos | 0.05 |
| AWS Certificate Manager (ACM) | Certificado SSL/TLS | 0.00 |

| | | |
|---|---|---|
| Amazon Route 53 | Administración de dominios y DNS | 0.50 |
| Amazon CloudWatch | Monitoreo y métricas | 1.00 |
| RDS | Servicio para Gestionar la Base de Datos | 26.00 |
| Total estimado mensual | — | **40.30 USD** |

Fuente: Calculadora de AWS

La infraestructura del sistema de comercio electrónico Mombashop se diseñó bajo una arquitectura de servicios desacoplados en Amazon Web Services (AWS). El objetivo es garantizar alta disponibilidad, escalabilidad y, fundamentalmente, viabilidad operativa al delegar la gestión de la infraestructura crítica, dado que Mombashop no cuenta con personal técnico dedicado.

El entorno de despliegue integra los servicios EC2 (para la aplicación), RDS (para la base de datos), S3, CloudFront, SES, Route 53, ACM y CloudWatch, conformando una solución robusta y alineada con las mejores prácticas de la nube.

Para la codificación del sistema, se eligieron los lenguajes React.js para el sitio web (frontend) y Nest.js para la API (backend), ambos implementados con TypeScript. Este enfoque permite definir de forma tipada la información que se expone al cliente, reduciendo errores en tiempo de ejecución.

En el entorno de despliegue, el *frontend* es transpilado a JavaScript y alojado en Amazon S3 como un sitio web estático. Desde allí, es distribuido globalmente a los usuarios mediante Amazon CloudFront (CDN), lo que optimiza los tiempos de carga. El tráfico es gestionado por Amazon Route 53, configurado con un certificado SSL/TLS (para HTTPS) emitido por AWS Certificate Manager (ACM).

El *backend* (API Nest.js) se ejecuta en una instancia Amazon EC2. La base de datos MySQL se despliega en Amazon RDS (Relational Database Service).

Esta separación entre la aplicación y la base de datos es un pilar fundamental de la viabilidad del proyecto: AWS gestiona automáticamente las copias de seguridad, los parches de seguridad y la alta disponibilidad de la base de datos, eliminando la necesidad de un administrador de bases de datos en la planilla de Mombashop y protegiendo la integridad de los datos (pedidos y clientes). Las imágenes de productos también se almacenan en S3 y se consumen desde el frontend a través de CloudFront.

Para la mensajería transaccional del sistema, se implementó Amazon SES (Simple Email Service), encargado del envío automatizado de notificaciones (confirmación de pedidos, alertas) por correo electrónico. Finalmente, se habilita el uso de Amazon CloudWatch para el monitoreo continuo del rendimiento de la infraestructura (como el uso de CPU de EC2 y RDS) y la detección de anomalías.

Esta arquitectura balanceada garantiza una solución segura, escalable y operativamente sostenible. El costo mensual estimado, basado en esta configuración, es de aproximadamente 47.30 USD, lo cual representa una solución económicamente viable que prioriza la resiliencia de los datos sobre el costo mínimo de infraestructura.

#### 7.4.1.4. **Análisis de la infraestructura actual vs la infraestructura propuesta**

El análisis de la infraestructura actual confirmó la ausencia de un centro de datos o servidores dedicados en Mombashop. Basado en este diagnóstico y en los requisitos no funcionales de escalabilidad, seguridad y disponibilidad se propone una Arquitectura de Tres Capas (Three-Tier Architecture) desplegada íntegramente en un entorno de nube pública.

Para la implementación de esta arquitectura, se ha seleccionado la plataforma de Amazon Web Services (AWS) como proveedor de servicios de infraestructura (IaaS) y plataforma (PaaS). Esta decisión estratégica elimina la necesidad de una inversión inicial en capital para hardware y adopta un modelo de costos operativos basado en el consumo, lo cual es financieramente viable para una microempresa.

La arquitectura se descompone en las siguientes capas:

**1. Capa de Presentación (Cliente / Frontend)**

Es la interfaz con la que interactúa el usuario final (el cliente de Mombashop) y los usuarios administrativos.

**Descripción:** Consiste en el navegador web (Google Chrome, Safari, etc.) ejecutándose en el dispositivo del usuario (PC, laptop, smartphone).

**Función:** Es responsable de renderizar la interfaz de usuario (UI), capturar las entradas del usuario y consumir los datos expuestos por la Capa de Aplicación a través de una API.

**Tecnología:** React JS

**2. Capa de Aplicación (Lógica de Negocio / Backend)**

Este es el núcleo del sistema, donde reside la lógica de negocio y se procesan todas las peticiones.

**Descripción:** Es el servidor *backend* que responde a las solicitudes de la Capa de Presentación.

**Función:** Gestiona el registro de usuarios, el procesamiento del carrito de compras, la validación de pedidos y la lógica del panel de administración (gestión de inventario, reportes).

**Tecnología y Despliegue (AWS):**

**Software: Node.js** (versión 18 o superior), tal como se definió en los requisitos de software.

**Servicio AWS Propuesto: Amazon EC2 (Elastic Compute Cloud)**

**3. Capa de Datos (Persistencia / Base de Datos)**

Es donde se almacena y gestiona toda la información crítica del negocio.

**Descripción:** El sistema gestor de base de datos que almacena de forma persistente los productos, clientes, órdenes y usuarios.

**Función:** Asegurar la integridad, consistencia y disponibilidad de los datos.

**Tecnología y Despliegue (AWS):**

**Software: MySQL** (versión 8.0 o superior).

**Servicio AWS Propuesto: Amazon RDS (Relational Database Service)**. Se selecciona RDS en lugar de instalar MySQL en un servidor EC2 manual, ya que RDS es un servicio gestionado que automatiza tareas complejas como la aplicación de parches, las copias de seguridad (backups), la replicación y la recuperación ante desastres, garantizando así la alta disponibilidad.

### 7.4.2. **Estudio Viabilidad Económica**

El estudio económico representa un componente esencial en la evaluación de la viabilidad de proyectos de desarrollo de software, al permitir la estimación de si los recursos financieros requeridos guardan proporción con los beneficios proyectados.

“Según Boehm, COCOMO II es una técnica ampliamente utilizada para estimar el esfuerzo, tiempo y costo requeridos en proyectos de software. Esta metodología permite calcular de manera estructurada los recursos necesarios a partir de variables como el tamaño del software, los factores de escala y los multiplicadores de esfuerzo, ofreciendo así una proyección fundamentada del costo total del proyecto.” (Boehm, 2000)

En el caso del sistema en desarrollo, dicho análisis tiene como propósito estimar los costos asociados a las etapas de desarrollo, operación y mantenimiento, así como evaluar su rentabilidad mediante indicadores financieros representativos. Para este fin, se aplican metodologías reconocidas como el modelo **COCOMO II** y el análisis por Puntos de Función, que facilitan la proyección del esfuerzo requerido y su conversión en términos económicos.

Este enfoque proporciona una base técnica y objetiva para respaldar la toma de decisiones estratégicas y asegurar la sostenibilidad del proyecto a mediano y largo plazo. (Boehm, 2000) propone que COCOMO II se compone por tres modelos.

Estos se denominan:

**Composición de Aplicación**: se emplea en desarrollos de software durante la etapa de prototipado.

**Diseño Temprano**: Este modelo es utilizado en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software, por lo cual en esta etapa se dispone de poca información.

**Post – Arquitectura**: se aplica en la etapa de desarrollo propiamente dicho, después que se define la arquitectura del sistema, y en la etapa de mantenimiento.

Para la estimación del esfuerzo y costo en el desarrollo del sistema, se adopta el modelo **Post-Arquitectura** de **COCOMO II**, el cual resulta apropiado en etapas donde la arquitectura del software ya ha sido definida y se cuenta con mayor claridad sobre los requerimientos técnicos y funcionales. Este modelo permite una estimación más precisa al considerar diversos factores de escala y multiplicadores de esfuerzo asociados a atributos del producto, del personal, de la plataforma y del proceso. Su aplicación se justifica por el nivel de madurez alcanzado en el diseño del proyecto eCommerce, permitiendo una planificación más realista de los recursos necesarios para su implementación.

#### 7.4.2.1. **Identificación Puntos de Función**

El proceso de estimación mediante Puntos de Función inicia con la identificación de los componentes funcionales del sistema, clasificados según el estándar propuesto por el **International Function Point Users Group (IFPUG)**. Esta fase permite descomponer el sistema en tipos de funciones tales como entradas externas, salidas externas, consultas externas, archivos lógicos internos y archivos de interfaz externa. Cada una de estas categorías representa una unidad funcional que contribuye al tamaño del software y, por tanto, influye directamente en la estimación del esfuerzo requerido para su desarrollo.

**Tabla 12** *Puntos de Función*

| ***Tipo de función*** | ***Abreviatura*** | ***Descripción breve*** |
|---|---|---|
| *Entradas externas* | EE | Datos ingresados por el usuario que afectan el sistema |
| *Salidas externas* | SE | Información derivada o calculada enviada al exterior |
| *Consultas externas* | CE | Solicitudes que no alteran datos pero recuperan información |
| *Archivos lógicos internos* | ALI | Tablas o estructuras controladas por el sistema |
| *Archivos de interfaz externos* | AIE | Tablas o estructuras mantenidas por otros sistemas |

Nota: Elaboración Propia

7.4.2.2. **Factores de Escala**

**Tabla 13** *Factores de escala*

| *Factor* | *Nivel* | *Valor* |
|---|---|---|
| *Precedencia (PREC)* | Nominal | 3.72 |
| *Flexibilidad (FLEX)* | Alta | 2.03 |
| *Resolución de riesgos (RESL)* | Alta | 2.83 |
| *Cohesión del equipo* | Nominal | 3.29 |

| | | |
|---|---|---|
| *(TEAM)* | | |
| *Madurez del proceso* (PMAT) | Nominal | 4.68 |

Suma de factores de escala (∑SF):

∑SF = 3.72 + 2.03 + 2.83 + 3.29 + 4.68 = 16.55

B = 0.91+0.01×16 = 1.07

Exponente de esfuerzo (E):

E = B + 0.01 × ∑SF = 0.91 + 0.01 × 16.55 = 1.0755

7.4.2.3. **Factores de costos**

Se presenta la tabla que resumen los factores determinantes que influyen en el esfuerzo requerido para el desarrollo del proyecto de software. La tabla describe distintos aspectos relevantes, como la fiabilidad exigida del sistema y la complejidad inherente al producto, junto con sus correspondientes descripciones y valores estimados.

**Tabla 14** *Factores de Costos*

| Factor | Descripción | Valor |
|---|---|---|
| RELY | Confiabilidad requerida del software | 1 |
| DATA | Tamaño de la base de datos | 0.9 |
| CPLX | Complejidad del producto | 1 |
| TIME | Restricciones de tiempo de ejecución | 1 |
| STOR | Restricciones de almacenamiento | 1 |

| | | |
|---|---|---|
| ACAP | Capacidad del analista | 0.85 |
| PCAP | Capacidad del programador | 0.9 |
| AEXP | Experiencia del analista | 1 |
| VEXP | Experiencia con la plataforma | 1 |
| LEXP | Experiencia en el lenguaje de programación | 1 |
| TOOL | Herramientas de software utilizadas | 1 |
| SCED | Restricción del cronograma | 1 |
| **EAF** | Esfuerzo Ajustado Final (producto de todos los EMi) | **0.903** |

Fuente: Elaboración Propia

A continuación, se presenta el cálculo del esfuerzo requerido para el desarrollo del sistema, utilizando la fórmula propuesta por el modelo COCOMO II en su etapa **Post-Arquitectura**. Este cálculo se basa en un conjunto de multiplicadores de esfuerzo (**EMi**) que representan diversos factores ambientales del proyecto, como la capacidad del personal, la experiencia, el uso de herramientas y la estabilidad de los requerimientos. Al multiplicar estos valores entre sí, se obtiene un multiplicador total que ajusta el esfuerzo estimado. Dicho valor se incorpora posteriormente en la fórmula general del modelo, dando como resultado el número de persona-meses necesarios para completar el proyecto.

**EAF**=1.00×0.90×1.00×1.00×1.00×0.85×0.90×1.00×1.00×1.00×1.00×1.00=0.902 5

**PM**=2.94×14.37×0.9025 = 38.03

Este resultado indica que el esfuerzo total requerido para el desarrollo del sistema es de aproximadamente **38.03 persona-meses**.

#### 7.4.2.4. **Asignación de Complejidad y Ponderación**

A continuación, se presenta una tabla que detalla los parámetros clave utilizados para evaluar el proyecto de desarrollo de software. A cada función se le asigna una ponderación basada en su complejidad, obteniendo un total estimado de puntos de función brutos.

**Tabla 15** *Asignación Complejidad y Ponderación*

| **Tipo PF** | **Simple** | **Medio** | **Complejo** | **Cantidad Estimada** | **Ponderación (prom.)** | **Subtotal** |
|---|---|---|---|---|---|---|
| *EE* | 3 | 4 | 6 | 11 | 4 | 44 |
| *SE* | 4 | 5 | 7 | 6 | 5 | 30 |
| *CE* | 3 | 4 | 6 | 5 | 4 | 20 |
| *ALI* | 7 | 10 | 15 | 8 | 10 | 80 |
| *AIE* | 5 | 7 | 10 | 2 | 7 | 14 |
| ***Total PF bruto*** | 188 | | | | | |

#### 7.4.2.5. **Cálculo de PFA (Puntos de función ajustado)**

El valor de los puntos de función ajustados se obtiene aplicando un factor de corrección basado en la complejidad técnica del sistema. Considerando una suma total de 35 puntos entre los 14 factores de complejidad definidos, el factor de ajuste calculado es igual a 1.00. Por tanto, el valor final de los puntos de función ajustados es:

**PFajustado** = PFbruto × (0.65 + 0.01 × ΣFi)

**Donde** ∑Fi es la suma de los 14 factores de ajuste (respuesta a características generales del sistema).

**PFajustado** = 188 × (0.65 + 0.01 × 35) = 188 × 1.00 = 188 PF

#### 7.4.2.6. Conversión de Puntos de Función a Líneas de Código (LOC)

Para estimar el número total de líneas de código del sistema, se utilizó un valor de conversión promedio de 53 LOC por Punto de Función, basado en estándares reconocidos para tecnologías web como JavaScript. Aplicando este valor al total de 188 Puntos de Función ajustados, se obtiene un tamaño estimado del software de 9,964 líneas de código (LOC), valor que se utiliza en la aplicación del modelo COCOMO II.

**LOC** = 188 × 53 = 9,964 LOC

#### 7.4.2.7. Estimación de Esfuerzo

Para estimar el esfuerzo requerido en persona-mes (PM), se aplica la fórmula principal del modelo COCOMO II Post-Arquitectura, la cual considera el tamaño del software (expresado en KLOC), un exponente ajustado por factores de escala y un multiplicador de esfuerzo derivado de atributos del proyecto. Esta estimación permite proyectar con mayor precisión el esfuerzo total necesario para el desarrollo del sistema.

**PM** = A×(Size)B×EAF

- PM: Esfuerzo estimado en persona-mes
- A: Constante de calibración (2.94)
- Size: Tamaño del software en KLOC (9.964)
- B: Exponente ajustado por escala (1.07)

EAF: Esfuerzo total ajustado, producto de los multiplicadores EMi = 0.9025

Sustituyendo:

**PM** = 2.94 × (9.964)1.07 × 0.9025 = **31.79 Persona/Mes**

Este valor representa el esfuerzo estimado requerido para completar el desarrollo del sistema bajo las condiciones y características evaluadas del equipo y del entorno del proyecto.

### 7.4.2.8. Calculo Tiempo Estimado

COCOMO II estima el tiempo total en meses calendario (TDEV) con la fórmula:

**TDEV = C×(PM)**

- C = 3.67 (constante empírica del modelo)
- PM = 31.79
- F = 0.28 + 0.2 × (B - 1.01), con B = 1.07

**Calculando F**

F= 0.28 + 0.2 × (1.07−1.01) = 0.28+0.012 = 0.292

**Calculando TDEV Tiempo Estimado:**

$$\boldsymbol{TDEV = 3.67\ x\ (31.79)^{0.292} = 3.67\ x\ 3.06 = 11.24 Meses}$$

**Cálculo de Costos Totales**

**Costo Total = PM×CPM**

**Donde:**

- PM = Persona-Mes estimado (31.79).
- CPM = Costo por Persona al Mes ( $500 USD).
- Costo Total = Costo total del desarrollo en dólares.

**Costo Total** = 31.79 PM x 500 USD = 15,895 USD

El análisis del esfuerzo y costos para el desarrollo del sistema se ha llevado a cabo utilizando el modelo COCOMO II en su modalidad Post-Arquitectura, complementado con estimaciones a partir de Puntos de Función ajustados. A través de este enfoque, se identificó un total de 188 Puntos de Función ajustados, que fueron convertidos a 9,964 Líneas de Código Fuente (LOC) utilizando un factor de conversión estándar. Posteriormente, se aplicaron los factores de escala y multiplicadores de esfuerzo correspondientes a las características del producto, la plataforma, el personal y el proyecto, lo cual condujo a una estimación de 31.79 persona-mes (PM) como esfuerzo requerido para el desarrollo. Considerando un equipo de tres desarrolladores trabajando a tiempo completo y un costo mensual por persona de 500 USD, se estimó un costo total de desarrollo de 15,895 USD.

La duración del proyecto, según la fórmula estándar del modelo, se proyecta en 11.24 meses.

Este estudio permite cuantificar de forma sistemática el esfuerzo y recursos necesarios, facilitando una planificación fundamentada del proyecto. El modelo COCOMO II, desarrollado por Barry Boehm y colaboradores, constituye una herramienta robusta para la estimación de costos de software, al considerar múltiples dimensiones que afectan el esfuerzo, como la complejidad técnica, el entorno operativo y la experiencia del equipo.

La implementación del sistema desarrollado representa una inversión estratégica que permitirá a la organización reducir significativamente los costos operativos asociados a la gestión manual de procesos. Actualmente, muchas de las actividades clave, como la administración de inventarios, procesamiento de pedidos, control de pagos y generación de reportes, se realizan de manera semiautomatizada o dependientes de múltiples herramientas desconectadas. Esta fragmentación no solo incrementa los tiempos de operación, sino que también eleva el riesgo de errores humanos, reprocesos y pérdida de información relevante.

Con la automatización e integración proporcionada por el nuevo sistema, se espera una disminución directa en el tiempo dedicado por el personal a tareas repetitivas, lo que se traduce en una mayor eficiencia laboral. Asimismo, se anticipa una mejor toma de decisiones gracias al acceso oportuno a indicadores clave y reportes en tiempo real. Estos beneficios se reflejan en ahorros tangibles, como la reducción en horas hombre, disminución de errores operativos y menor dependencia de soluciones de terceros o licencias innecesarias.

Si bien el desarrollo del sistema implica un costo total aproximado de **15,895 USD**, este puede ser recuperado en un corto plazo mediante los **ahorros anuales proyectados en costos de personal, insumos y optimización de recursos**. De forma conservadora, si se estima que la organización reduce en al menos **2,500 USD mensuales en costos operativos**, lo que refuerza la viabilidad económica del proyecto. Además, se debe considerar el valor agregado intangible que implica contar con una solución tecnológica propia, escalable y alineada a las necesidades reales del negocio.

### 7.4.3. **Estudio Viabilidad Legal**

El estudio de viabilidad legal tiene como objetivo determinar la inexistencia de restricciones legales que puedan impedir la instalación, operación y mantenimiento del sistema propuesto, así como la ausencia de normativas internas de la empresa que contravengan su implementación (Chain, 2011).

En Nicaragua, no existen leyes que regulen específicamente el uso de herramientas informáticas para la gestión y control de operaciones comerciales, salvo en el caso de sistemas vinculados directamente con la contabilidad empresarial. Dado que el sistema propuesto para MombaShop se centra en la **gestión de inventarios, control de pedidos y generación de reportes complementarios**, y no afecta directamente la contabilidad de la empresa, no se identifican restricciones legales que limiten su implementación.

No obstante, el desarrollo del sistema se encuentra sujeto a un marco legal general que incluye la protección de derechos de autor, la regulación del software y la protección de datos personales, dentro del cual se destacan los siguientes instrumentos:

1. **Ley No. 312, Ley de Derechos de Autor y Derechos Conexos (1999)** El artículo 39 establece que el propietario legítimo de un programa de ordenador puede realizar copias o adaptaciones necesarias para su uso, sin autorización del autor, siempre que se utilice el software para los fines para los cuales fue diseñado. Esto implica que todas las herramientas de desarrollo, librerías, plugins o plantillas empleadas deben contar con la licencia correspondiente. En el proyecto, se utilizarán herramientas **Open Source**, de uso gratuito y legalmente habilitado, así como herramientas de proveedores comerciales como Microsoft y Oracle, para las cuales se garantizará la adquisición de licencias necesarias. Cabe destacar que **MySQL Community Server**, que se empleará para la gestión de la base de datos, se utilizará en su versión gratuita y legal.

2. **Ley No. 787, Ley de Protección de Datos Personales (2012)** Esta ley tiene como objetivo garantizar el derecho a la privacidad y la autodeterminación informativa de las personas, regulando el tratamiento de datos personales en ficheros públicos y privados. En cumplimiento de esta normativa, el sistema solo recopilará y almacenará información estrictamente necesaria de los usuarios, previa autorización expresa. Durante la fase de desarrollo, se emplearán datos de prueba y, posteriormente, la información utilizada por la empresa se limitará a fines operativos y académicos, evitando cualquier vulneración de derechos.

En conclusión, **el sistema propuesto para MombaShop es legalmente viable**, ya que no infringe derechos de autor, se emplean herramientas con licencias válidas y la recolección de datos cumple con la legislación vigente en materia de protección de datos personales. De este modo, la implementación de tecnologías como **Node.js** para el backend y **MySQL** para la base de datos se realiza dentro del marco legal aplicable, garantizando seguridad jurídica en el desarrollo y operación del sistema.

# VIII. CAPITULO II: ANÁLISIS Y DISEÑO DEL SISTEMA

En el presente capítulo se expone un análisis detallado del diseño del sistema, abarcando la definición de sus componentes estructurales, arquitectura, módulos funcionales e interfaces de interacción. Asimismo, se examina el flujo de datos que circula entre dichos elementos, lo cual permite una comprensión integral de la organización y funcionamiento del sistema propuesto.

## 8.1. **Ingeniería de Requerimientos**

La presente sección aborda el proceso de ingeniería de requerimientos, etapa fundamental en el desarrollo de sistemas de software, cuya finalidad es identificar, analizar, documentar y validar las necesidades del sistema desde una perspectiva funcional y no funcional. Esta fase permite establecer una base sólida para el diseño, implementación y posterior evaluación del sistema, asegurando que las funcionalidades desarrolladas respondan adecuadamente a las expectativas y necesidades de los usuarios y demás partes interesadas. La correcta definición de los requerimientos contribuye a reducir errores en etapas posteriores y a garantizar la calidad del producto final.

### 8.1.1. **Requerimientos Funcionales**

Los requisitos funcionales describen las acciones específicas que el sistema debe ser capaz de ejecutar, en respuesta a las necesidades de los usuarios y los objetivos del negocio. Estos requisitos establecen el comportamiento del sistema ante diversas situaciones, definiendo las funciones esenciales que deben implementarse, tales como el inicio de sesión, la gestión de usuarios, el procesamiento de pedidos o la generación de reportes. Su correcta especificación permite delimitar el alcance del sistema y orientar el desarrollo de funcionalidades de manera precisa y controlada.

***Tabla 16*** *Requerimientos Funcionales*

| Codigo | Funcionalidad | Descripcion |
|---|---|---|
| RF-01 | Gestion de pedido | Permite registrar, procesar y dar seguimiento a los pedidos realizados por los usuarios en la plataforma de comercio electrónico, desde la selección de productos hasta la entrega final. |
| RF-02 | Gestion carrito de compra | Permite la incorporación, modificación y eliminación de productos seleccionados por el usuario antes de formalizar un pedido, permitiendo visualizar precios, cantidades y totales de forma dinámica. |
| RF-03 | Seccion de contactenos | Permite a los usuarios enviar consultas, comentarios o solicitudes a la tienda mediante un formulario, facilitando la comunicación directa con el administrador del sistema. |
| RF-04 | Gestion de usuario | Permite el registro, autenticación y administración de cuentas de los usuarios, controlando los niveles de acceso y garantizando la seguridad de la información personal dentro del sistema. |
| RF-05 | Gestion inhabilitar usuario | Permite desactivar temporal o permanentemente el acceso de un usuario al sistema, restringiendo el inicio de sesión y el uso de funcionalidades |

| | | |
|---|---|---|
| | | según criterios administrativos o de seguridad. |
| RF-06 | Gestionar cliente | Permite registrar, actualizar y consultar la información personal y de contacto de los clientes, facilitando la administración eficiente de sus perfiles y el seguimiento de sus actividades dentro del sistema. |
| RF-07 | Gestionar departamento | Permite crear, modificar y eliminar departamentos dentro del sistema, facilitando la organización jerárquica de productos, usuarios o áreas de gestión según la estructura de la tienda. |
| RF-08 | Gestionar Municipio | Permite gestionar el registro, actualización y eliminación de municipios dentro del sistema, facilitando su vinculación con departamentos y la organización territorial de datos relacionados. |
| RF-09 | Gestionar factura | Permite la generación automática de facturas por cada transacción realizada en el sistema, incluyendo los detalles del cliente, productos adquiridos, precios, impuestos aplicables y métodos de pago utilizados. |
| RF-10 | Gestionar tipo de cambio | El sistema debe permitir configurar, actualizar y aplicar automáticamente el tipo de cambio entre monedas para |

| | | |
|---|---|---|
| | | mostrar precios ajustados al usuario según su región o preferencia. |
| RF-11 | Gestionar tarifa de envio | El sistema debe permitir definir, actualizar y aplicar automáticamente tarifas de envío según ubicación del cliente, peso del pedido y método de entrega seleccionado. |
| RF-12 | Registro de empleado | El sistema debe permitir a los administradores registrar nuevos empleados, asignarles roles y gestionar su información personal y credenciales de acceso. |
| RF-14 | Agregar productos a inventario | El sistema debe permitir a los usuarios autorizados agregar nuevos productos al inventario, especificando sus atributos comerciales, logísticos y de stock inicial. |
| RF-15 | Funcion para cargar existencia al inventario | El sistema debe permitir a los usuarios autorizados cargar existencias al inventario |
| RF-16 | Gestionar Categoria de Productos | El sistema debe permitir al administrador crear, editar, eliminar y organizar categorías para clasificar los productos del ecommerce de manera jerárquica. |
| RF-17 | Gestionar catalogo de presentacion de producto | El sistema debe permitir crear y administrar las diferentes presentaciones de un producto base, como combinaciones de talla, color u otra |

| | | |
|---|---|---|
| | | variante, cada una con su propio inventario y precio si aplica. |
| RF-18 | Reporte de Ventas | El sistema debe permitir generar un reporte de ventas mediante un rango de fechas |
| RF-19 | Notificar Correo Electronico | El sistema debe enviar notificaciones mediante correo electronicos al administrador cuando se genere un nuevo pedido |
| RF-20 | Notificar al Correo electronico del cliente | El sistema debe generar un correo electronico al cliente tras cada actualizacion de estado en el pedido |
| RF-21 | Acceso al Repartidor | El sistema debera brindar un acceso al repartidor para marcar el pedido como entregado |

Fuente: Elaboración Propia

### 8.1.2. **Requerimientos No Funcionales**

Los Requisitos No Funcionales (RNF) son un conjunto de especificaciones que definen los criterios de calidad y las restricciones operativas del sistema. A diferencia de los requisitos funcionales, que describen qué debe hacer el sistema, los requisitos no funcionales describen cómo debe hacerlo. Estos requisitos establecen los atributos de calidad, tales como el rendimiento, la seguridad, la usabilidad, la compatibilidad y la disponibilidad, sirviendo como el marco bajo el cual se evalúa la efectividad y la robustez de la plataforma.

**Tabla 17** Requerimientos No Funcionales

| Codigo | Funcionalidad | Descripcion |
|---|---|---|
| RNF-01 | Compatibilidad | El sistema debe ser compatible con navegadores populares como Google Chrome, Mozilla Firefox, Safari en sus versiones más recientes. |
| RNF-02 | Disponibilidad | El sistema debe tener una disponibilidad de 24 horas los 7 días de la semana |
| RNF-03 | Seguridad | Se implementará la seguridad de roles a nivel de usuarios para segmentar el acceso de usuarios a datos no autorizados.<br><br>La base de datos deberá estar respaldada al menos dos veces a la semana.<br><br>Toda la comunicación entre el cliente y el servidor debe estar protegida mediante protocolos de encriptación SSL/TLS. |
| RNF-04 | Ergonomía | La aplicación web deberá ser responsiva. La aplicación debe ser intuitiva y fácil de usar. El sistema debe de dar mensajes fáciles de interpretar por el usuario. |
| RNF-05 | GOOGLE ANALYTICS | El sistema debe tener una integracion con google analytics para analizar el trafico y preferencias de los usuarios |

Fuente: Elaboración Propia

El sistema será desarrollado como una plataforma web utilizando tecnologías modernas y escalables, lo que permitirá a los usuarios acceder a sus funcionalidades desde cualquier dispositivo con conexión a internet. Esta arquitectura técnica garantiza facilidad de acceso, usabilidad y conveniencia para los distintos roles del sistema.

En el entorno web, los clientes podrán consultar el catálogo de productos desde la comodidad de su hogar o lugar de trabajo, visualizando información actualizada de los artículos disponibles. Además, tendrán la posibilidad de interactuar con el sistema mediante comentarios y consultas en tiempo real, lo que mejora la experiencia del cliente y fomenta la comunicación directa.

A nivel local, el sistema permitirá realizar la gestión integral del inventario, facilitando la actualización, monitoreo y control de los productos en existencia. Los usuarios administrativos podrán registrar nuevos clientes, gestionar ventas y acceder a reportes detallados que ofrecerán información clave para la toma de decisiones. Entre los datos procesados, el sistema proporcionará:

- Un análisis de los productos con mayor rotación en el inventario, lo que ayudará a identificar tendencias de consumo y planificar el reabastecimiento.

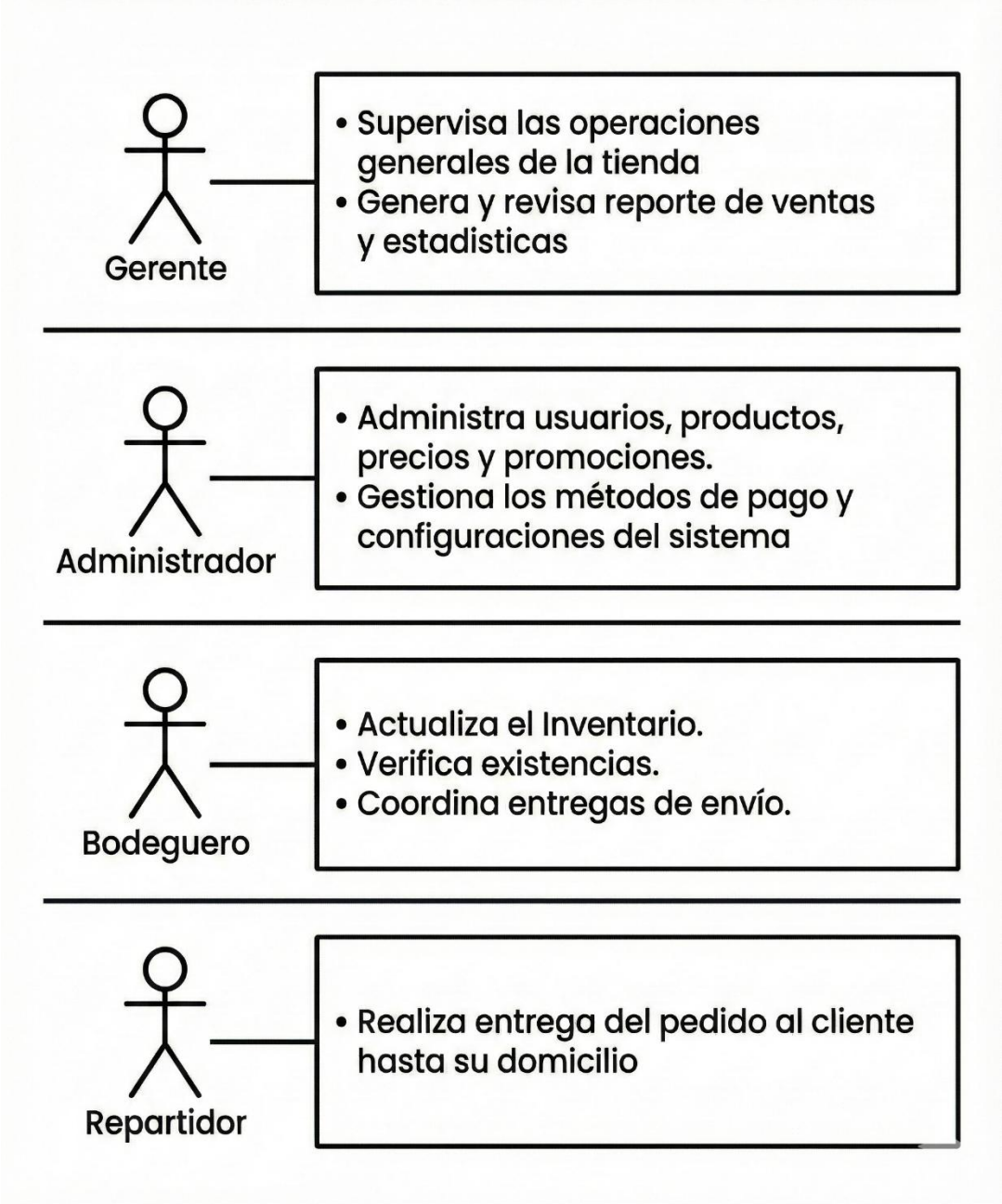
- Información consolidada sobre los clientes, incluyendo su historial de compras, preferencias y datos relevantes, optimizando así las estrategias de fidelización.

El sistema está diseñado para garantizar alta disponibilidad y seguridad, empleando tecnologías de código abierto robustas y confiables. Esto asegura un entorno eficiente y accesible para todos los usuarios, cumpliendo con los objetivos operativos y técnicos de la organización.

## 8.2. **Diagrama de Actores del Sistema**

En la siguiente figura se muestran los actores del sistema web con una breve descripción de las funciones que realizan.

***Figura 3.*** *Diagrama de Actores del sistema*

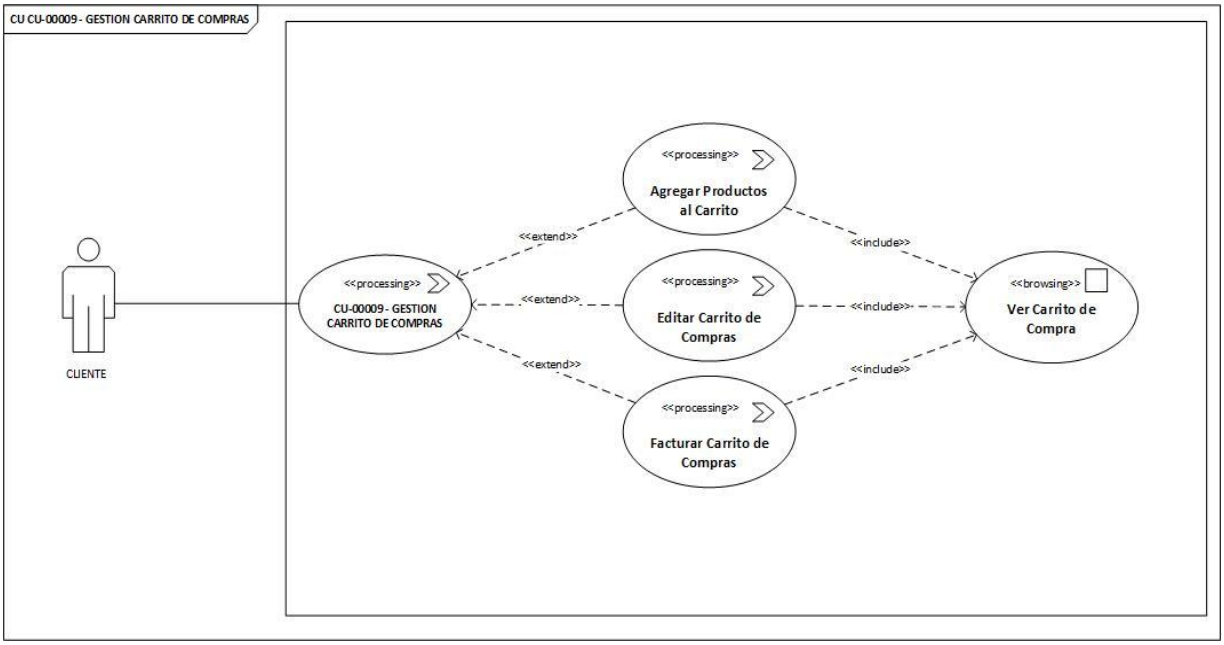


## 8.3. **Diagrama de Flujo de Actividades y Casos de Uso**

La presente sección expone los diagramas de flujo de actividades y casos de uso correspondientes al sistema en desarrollo. Estos recursos permiten representar de manera estructurada las operaciones que se ejecutan dentro del sistema, así como la interacción entre los diferentes actores y sus respectivos roles. Los diagramas de actividades describen la secuencia lógica de acciones y decisiones involucradas en los procesos, mientras que los casos de uso identifican las funcionalidades clave del sistema desde la perspectiva del usuario.

En esta sección se ilustran los procesos principales del sistema, tales como la creación del carrito de compras y la gestión de pedidos. Las funciones complementarias, así como sus respectivos diagramas, se incluyen en los Anexos. Esta representación gráfica facilita la comprensión de los comportamientos esperados, contribuye a la validación de los requerimientos y orienta las etapas posteriores del diseño y desarrollo.

***Figura 4.*** *Diagrama Caso de Uso "Gestión de Carrito de compras"*



Este diagrama de caso de uso describe las acciones permitidas al cliente para gestionar el carro de compras.

- Agregar Productos al carrito: El cliente selecciona un producto que desee comprar y lo adiciona al carro de compras

- Editar carrito de compras: El cliente puede eliminar productos previamente adicionados al carro de compras

- Facturar carrito de compras: El cliente puede generar un pedido a partir del carro de compras previamente creado.

**Actividad de Caso de Uso: Agregar Productos al carrito**

La actividad “Agregar Productos al carrito” permite al cliente adicionar un producto seleccionado del catalogo al carro de compras. Los pasos principales incluyen:

1. El usuario selecciona una categoría de preferencia

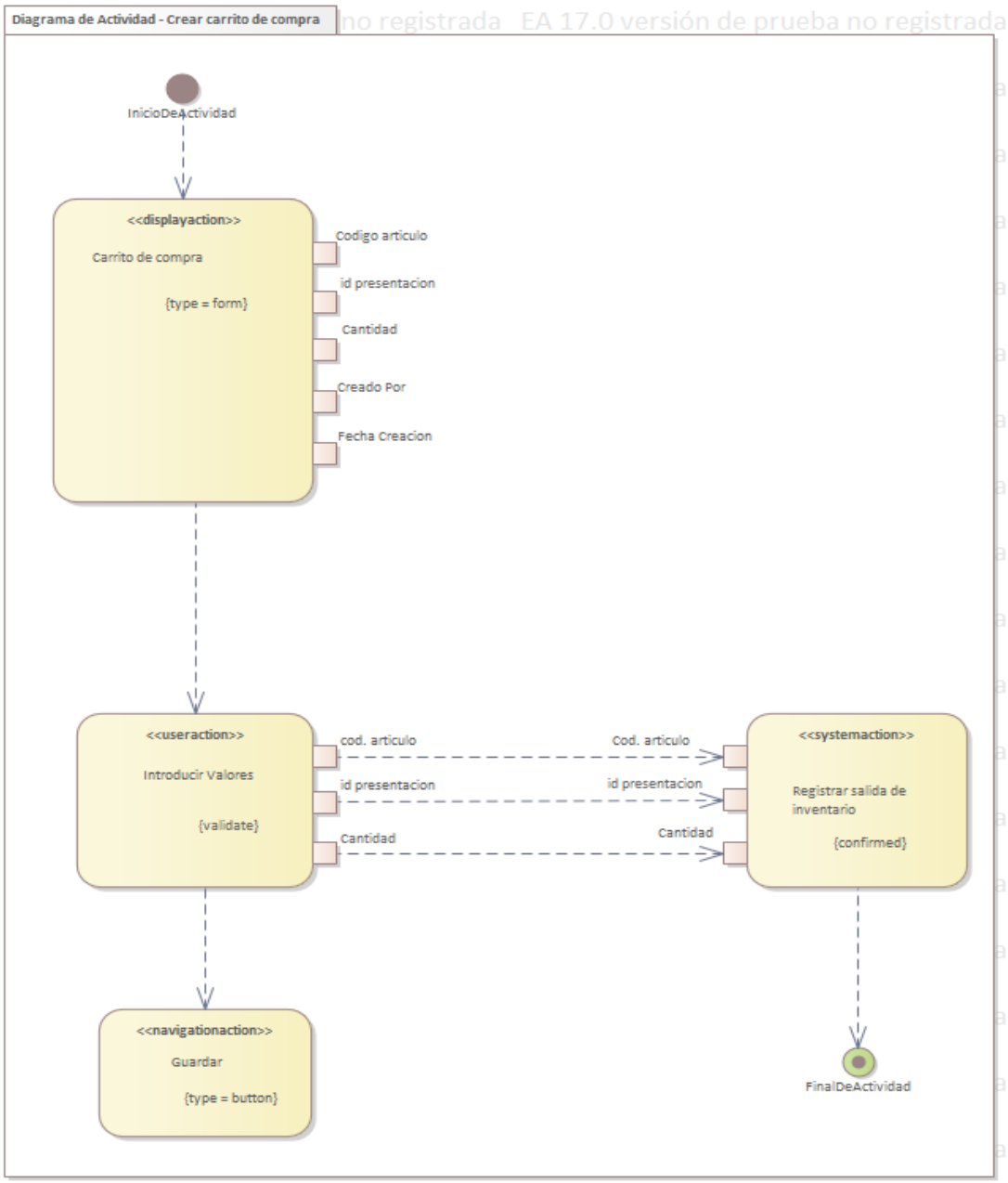
2. Se despliegan todos los artículos asociados a esa categoría

3. El usuario selecciona el producto de interés y en la ventana de detalle se encarga de seleccionar la presentación/talla y cantidad a requerida a comprar

4. El usuario presiona el botón agregar al carro de compra

Este proceso finaliza al confirmar el registro del producto en el carro de compras.

**Figura 5.** Diagrama de Actividad de Caso de Uso: Crear carro de compra



**Actividad de Caso de Uso: Editar Carrito de compras.**

La actividad “Editar carrito de compras” permite al cliente actualizar el contenido del carro de compras, Los pasos son los siguientes:

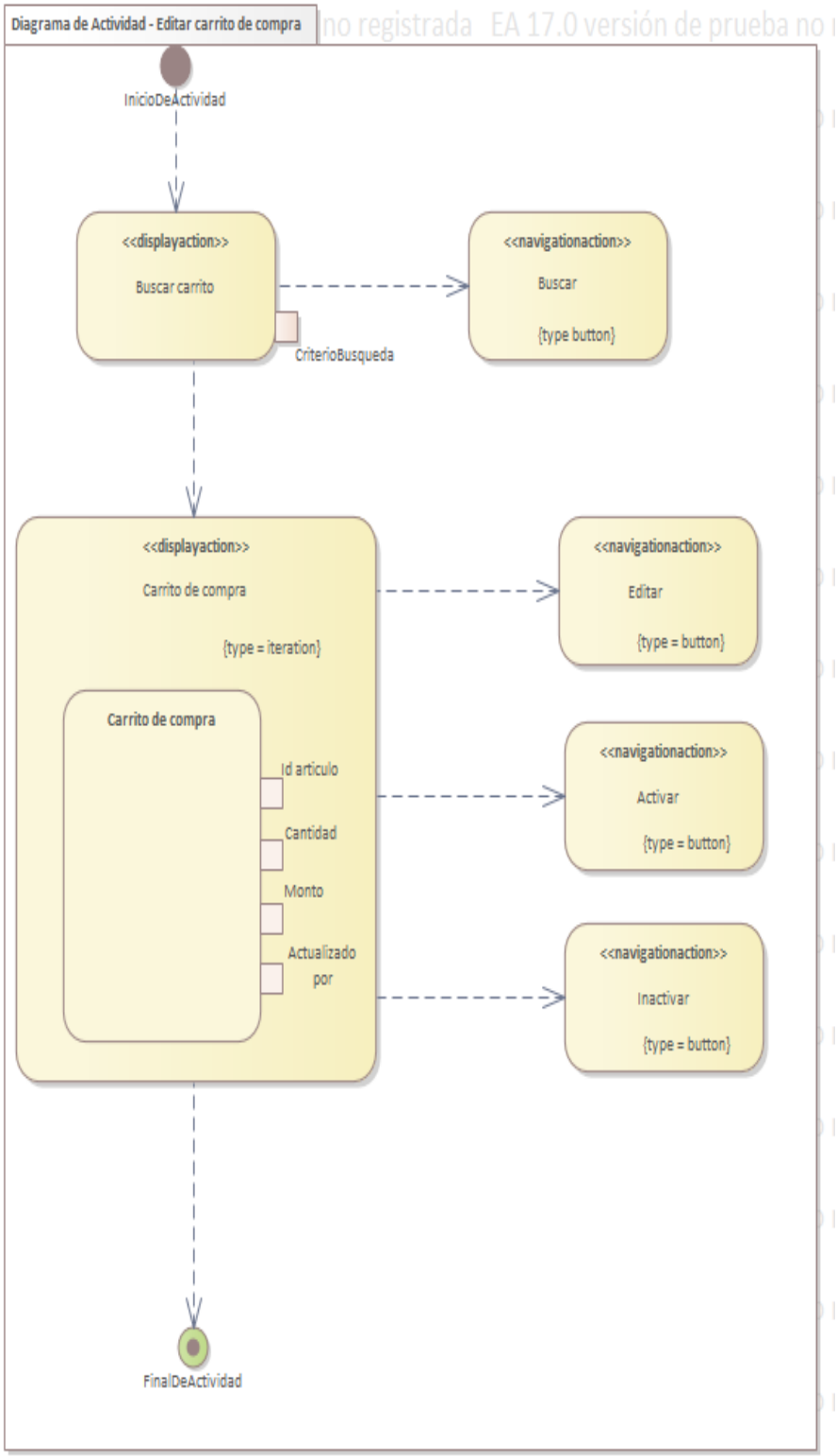
1. Buscar el carro de compras: El usuario se dirige hacia el menú que contiene la opción de ver carrito de compras

2. Seleccionar: El usuario selecciona el carro de compras y le da click sobre la opción ver detalle

3. Remover: El usuario selecciona el producto que desea remover dándole click sobre el botón eliminar.

Este proceso concluye con la confirmación de que el producto ha sido removido del carro de compras de forma satisfactoria

***Figura 6.*** *Diagrama de actividad de Caso de Uso: Editar Carrito de compras*



**Actividad de Caso de Uso: Pagar Carrito de compras**

La actividad de “pagar el carrito de compras” consiste en la capacidad que tiene el usuario en generar un pedido a partir de un carro de compras, Los pasos a seguir son los siguientes:

1. Buscar el carro de compras: El usuario se dirige hacia el menú que contiene la opción de ver carrito de compras

2. Seleccionar: El usuario selecciona el carro de compras y le da click sobre la opción ver detalle

3. Pagar: El usuario selecciona el método de pago con el cual desea marcar como pagado su carro de compras y generar el pedido

Este proceso concluye con la confirmación de que el pedido ha sido generado de forma satisfactoria.

***Figura 7.*** *Diagrama de actividad de Caso de Uso: Creación de Pedido*

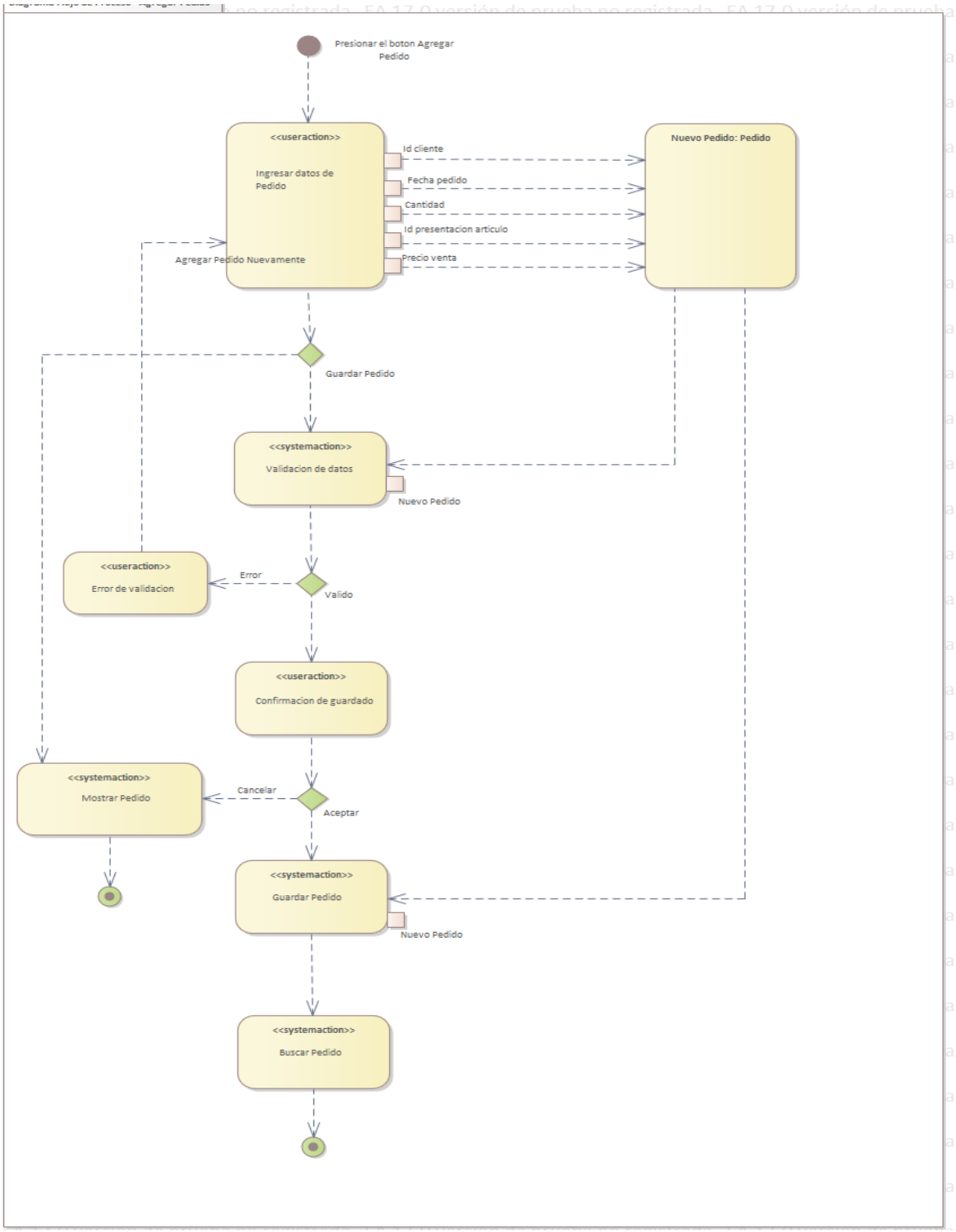
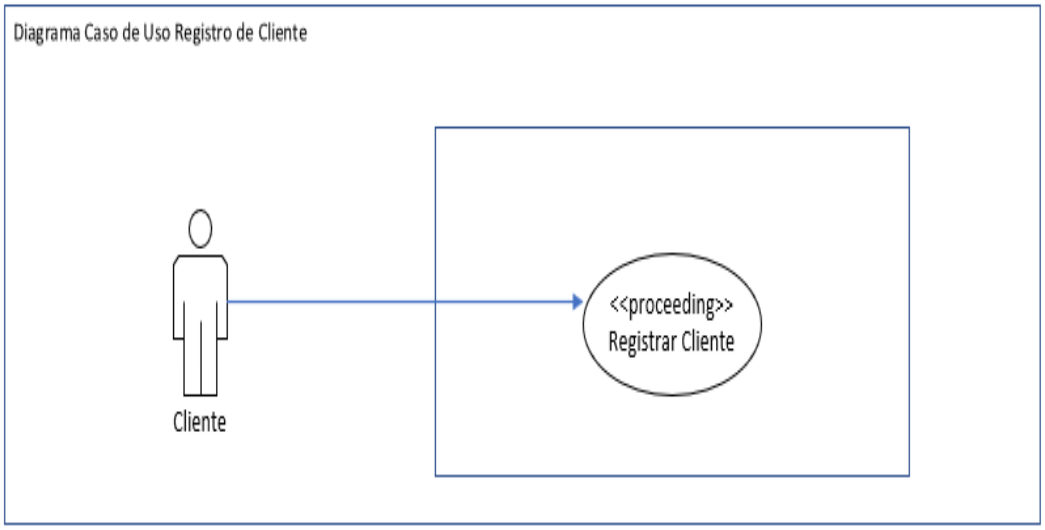


***Figura 8.*** *Diagrama Caso de Uso: Registro de Cliente*




Este diagrama de caso de uso se encarga de representar el proceso que tiene el cliente para registrarse en la plataforma.

**Actividad de Caso de Uso: Registro Cliente**

La actividad de registro de cliente consiste en la capacidad para Auto Registrarse en la plataforma para poder generar un pedido, Los pasos son los siguientes:

1. Selección de opción del menú para registrar
2. Llenado de formulario para registrar al cliente

***Figura 9.*** *Diagrama de actividad de caso de uso: Registro de cliente*

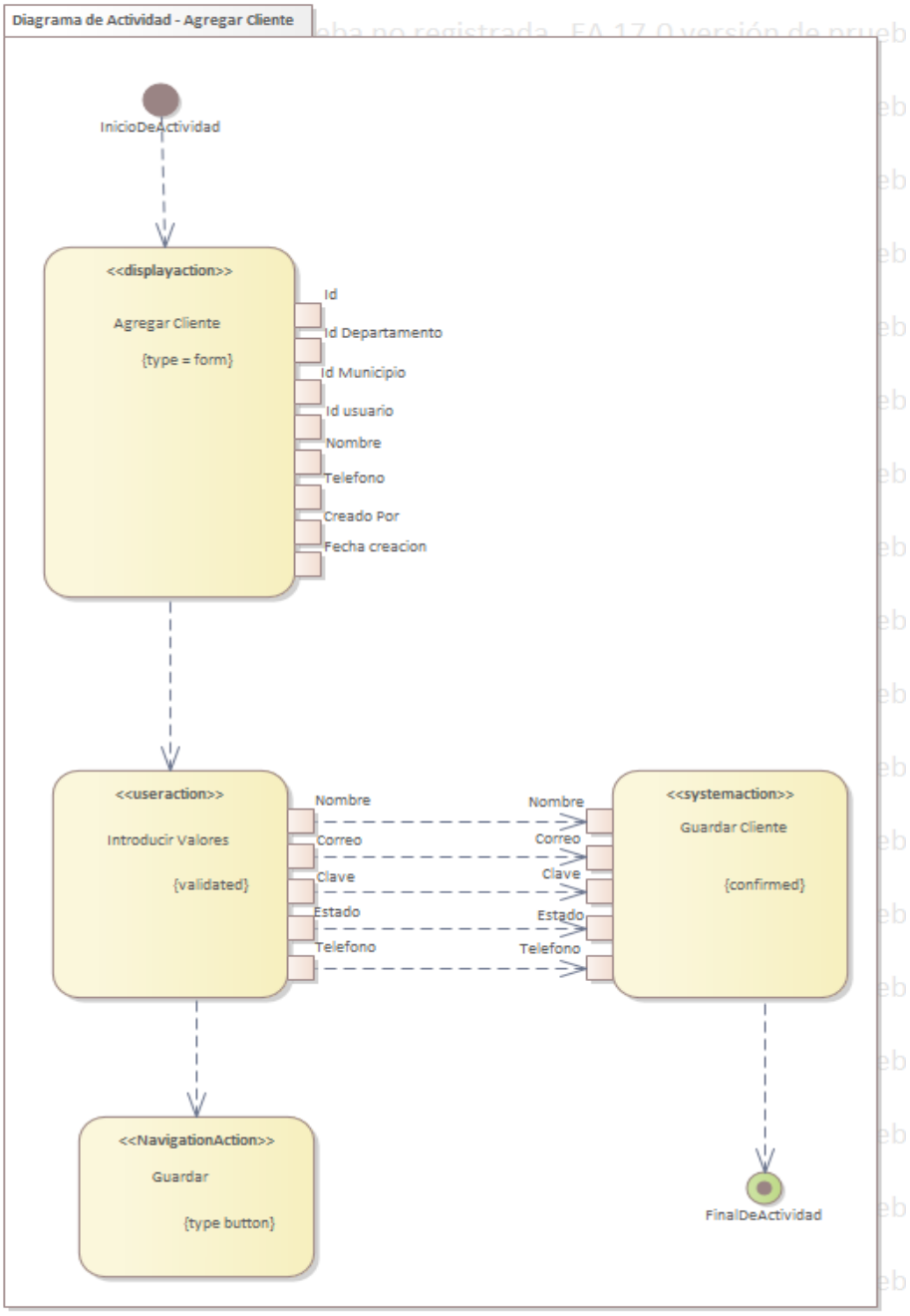
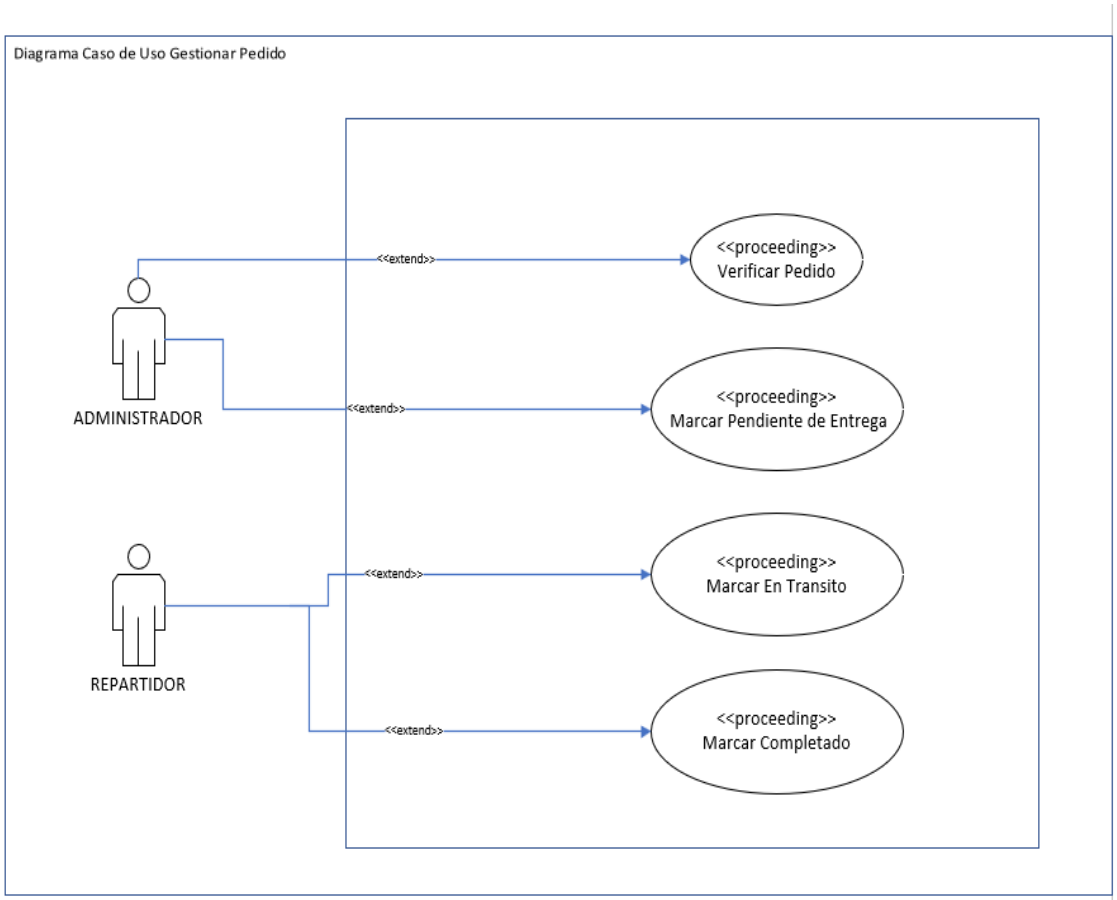


***Figura 10.*** *Diagrama Caso de Uso: Gestión de Pedido*




Este diagrama de caso de uso representa el proceso de la gestión de un pedido dentro de las cuales se encuentran las siguientes funciones:

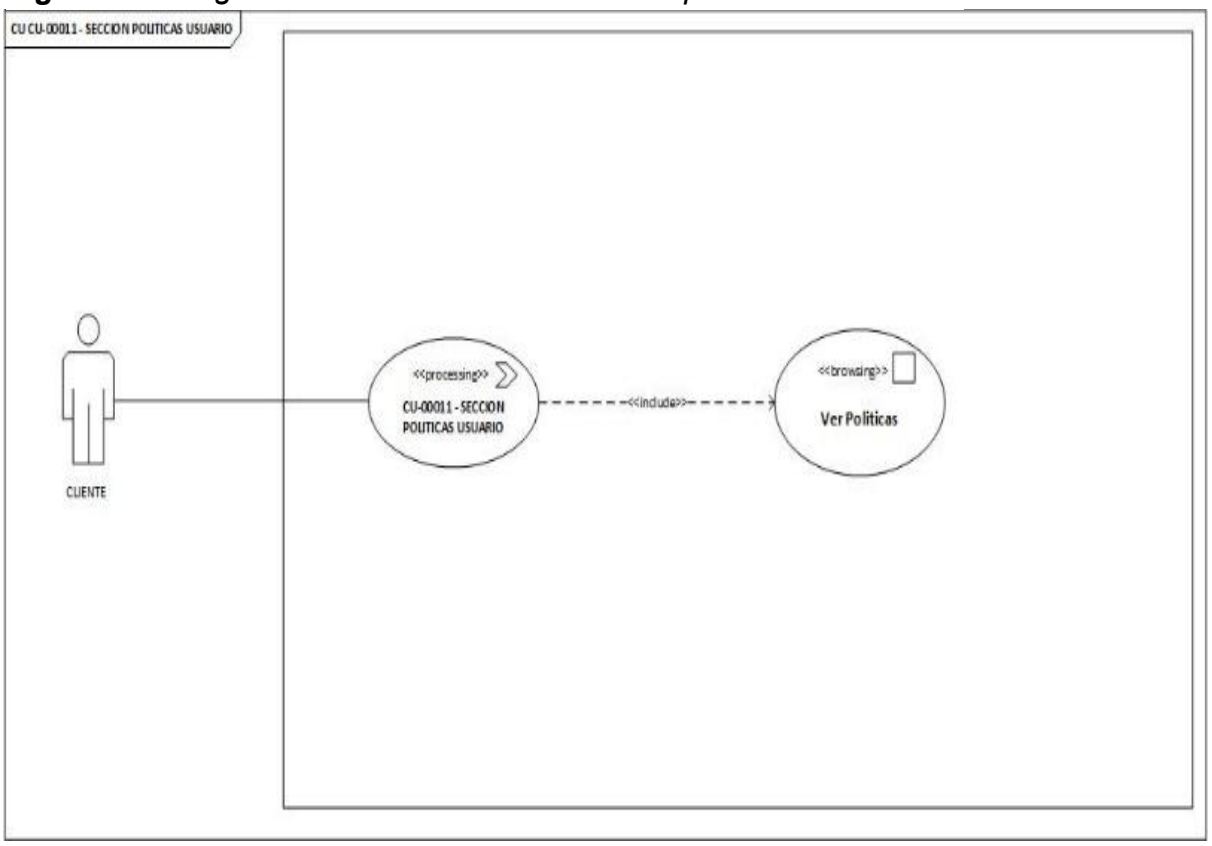
1. Verificar Pedido: Proceso en el cual el administrador verifica el pedido, confirma que el pago se ha recibido de forma satisfactoria.

2. Marcar Pendiente Entrega: Se asigna el repartidor encargado en entregar el pedido y el pedido queda en un estado de pendiente de entrega

3. Marcar en Transito: El usuario tiene la capacidad de marcar el pedido que se encuentra en camino a ser entregado

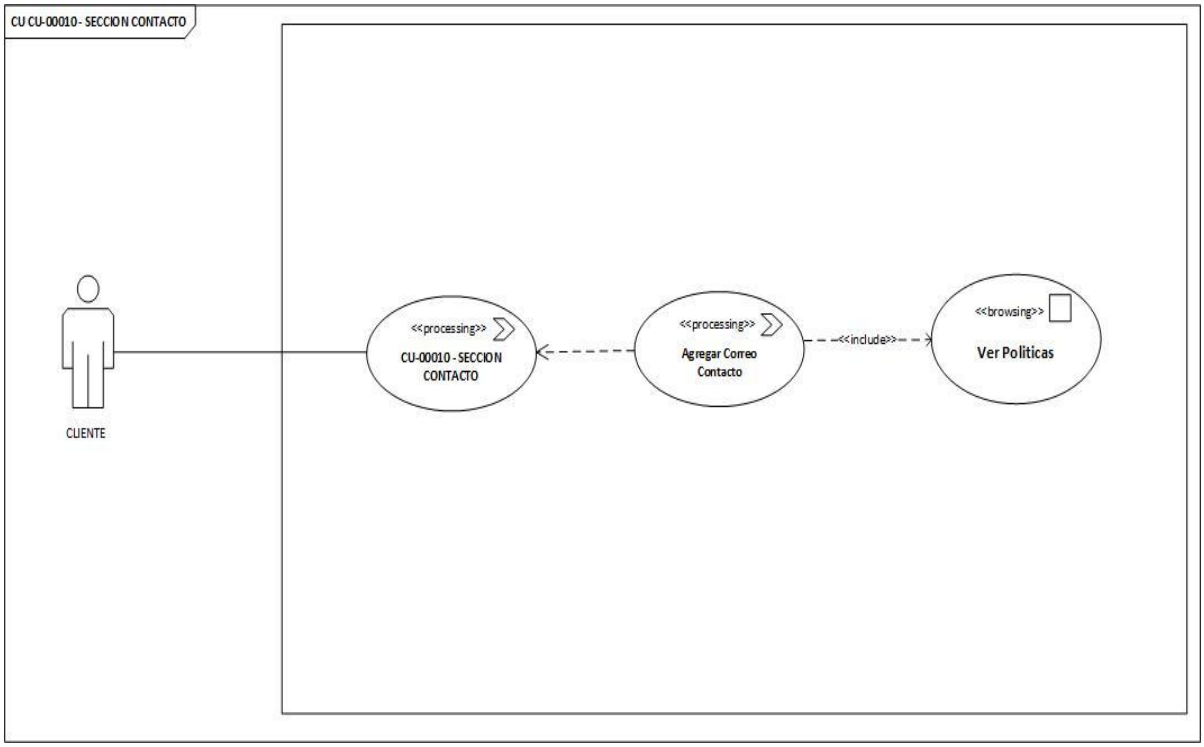
4. Marcar Completado: Función que permite al usuario marcar el pedido como completado

***Figura 11**. Diagrama Caso de Uso: Sección de políticas*




Este diagrama de caso de uso representa la capacidad que tiene el cliente para acceder a la sección de políticas de la tienda.

***Figura 12**. Diagrama Caso de Uso: Acceso a Contáctenos*

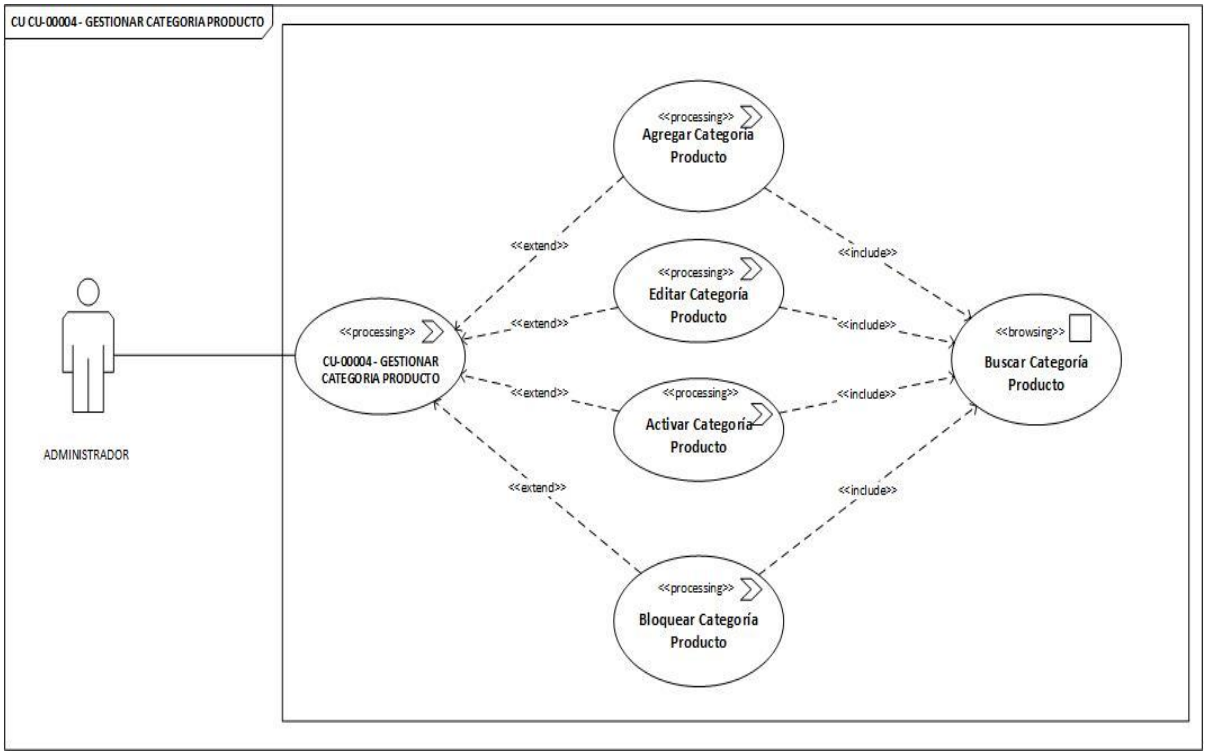



**Fuente**: Elaboración Propia

Este diagrama de caso de uso representa el proceso del usuario para poder acceder a la sección de contáctenos y poder enviar un correo de contacto a los administradores, dentro de las cuales se pueden ver las siguientes acciones:

1. Redactar mensaje de contacto

**Figura 13.** *Diagrama Caso de Uso: Gestión de Categoría Producto*




Este diagrama de caso de uso representa las acciones que se pueden realizar con las categorías del producto entre las cuales se encuentran las siguientes:

1. Agregar Categoría Producto: El Usuario tiene la capacidad de crear la categoría de un producto.

2. Editar Categoría Producto: El usuario tiene la capacidad de actualizar la categoría de un producto.

3. Activar Categoría Producto: El usuario tiene la capacidad de activar o inactivar la categoría de un producto.

4. Buscar Categoría Producto: El usuario tiene la capacidad de buscar la categoría de un producto.

**Actividad Caso de Uso: Agregar Categoría Producto**

La actividad registro de categoría producto consiste en la capacidad que tiene el administrador de adicionar clasificaciones de productos para posterior ser presentados en el menú del cliente, los pasos para llevar a cabo el registro son los siguientes:

1. El administrador accede a la opción para administrar categorías de producto
2. Llena el formulario de nueva categoría producto.

***Figura 14.*** *Diagrama de Actividad de Caso de uso: Crear Categoría Producto*

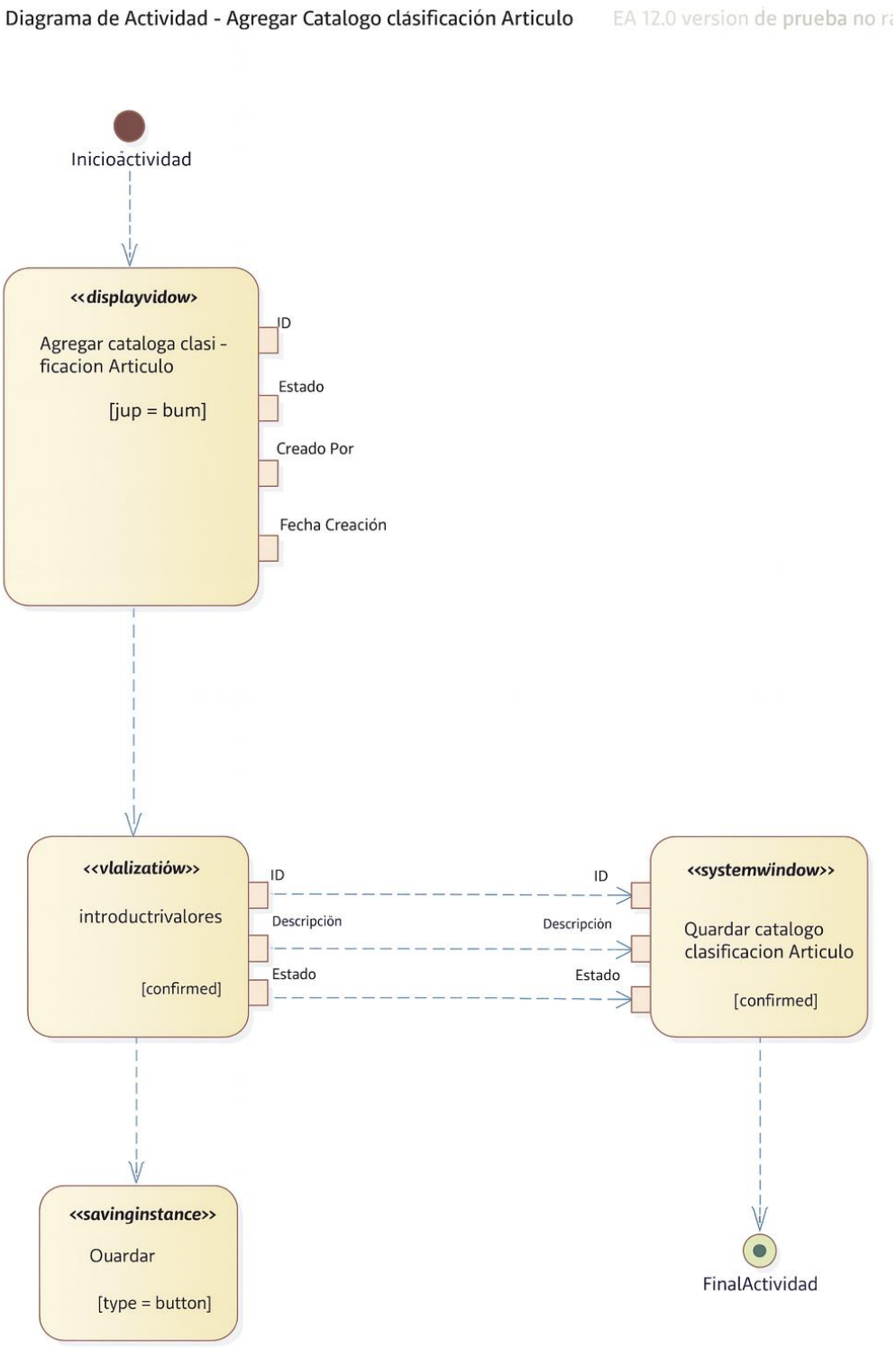
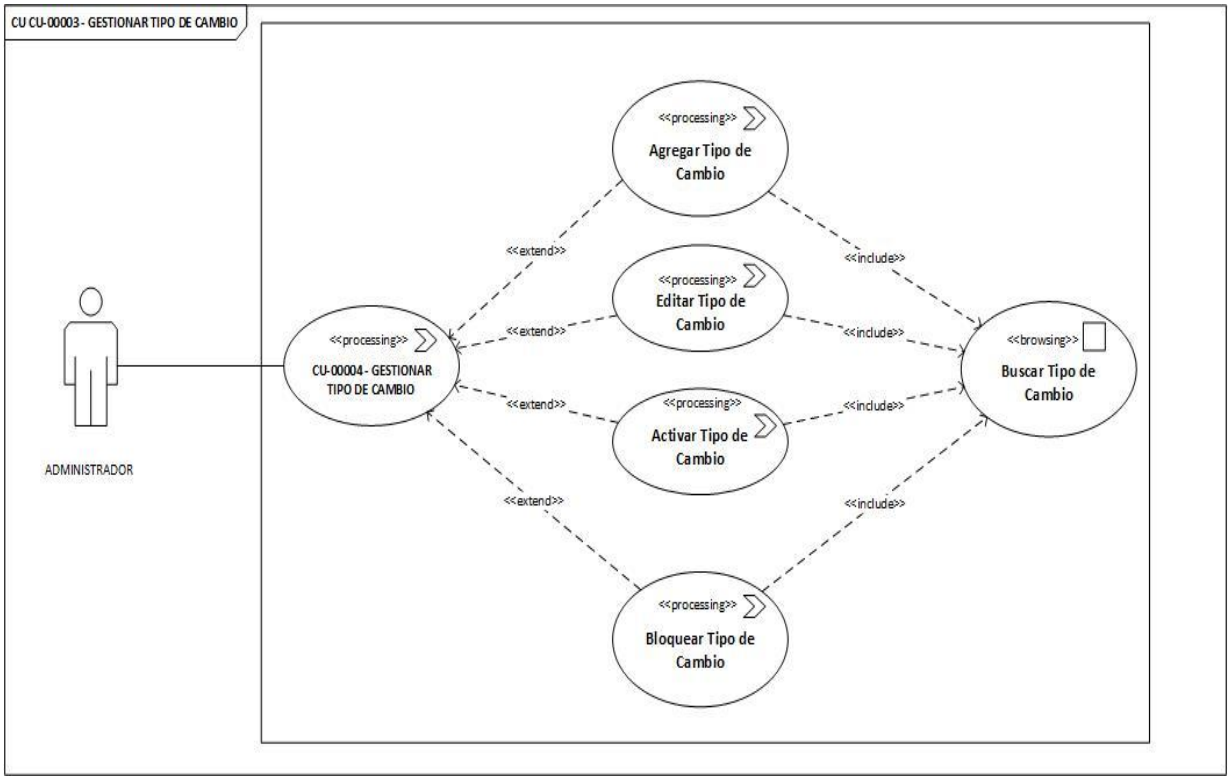


***Figura 15**. Diagrama de Caso de Uso: Gestión de Tipo de Cambio*




Este diagrama de caso de uso expresa las funcionalidades que posee el usuario para gestionar los tipos de cambios dentro de las cuales se encuentran:

1. Agregar Tipo de Cambio: El usuario puede crear tipos de cambios para manejar los cobros en dólares como también en moneda local.
2. Editar Tipo de Cambio: El usuario puede actualizar el tipo de cambio del día.
3. Activar Tipo de Cambio: El usuario puede activar el tipo de cambio del día.
4. Bloquear Tipo de Cambio: El usuario puede bloquear el tipo de cambio del día.

**Actividad de Caso de Uso: Agregar Tipo de Cambio**

1. La actividad “Agregar Tipo de Cambio” permite al usuario ingresar y almacenar nuevos tipos de cambios, los pasos a son:

2. El administrador inicia el proceso de nuevo tipo de cambio.

3. El administrador rellena la información del formulario de un nuevo tipo de cambio

***Figura 16.*** *Diagrama de actividad de caso de uso: Crear Tipo de Cambio*

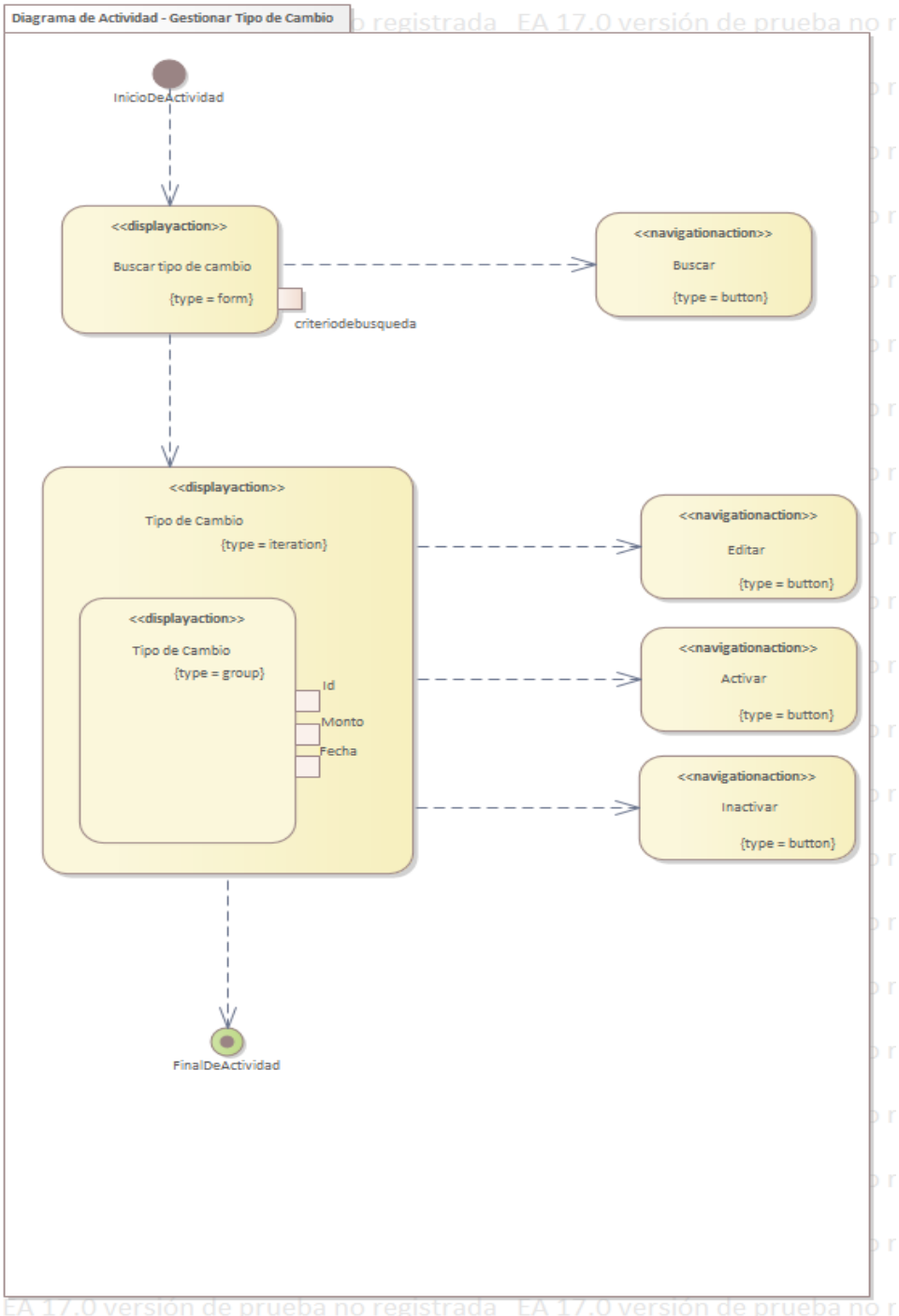
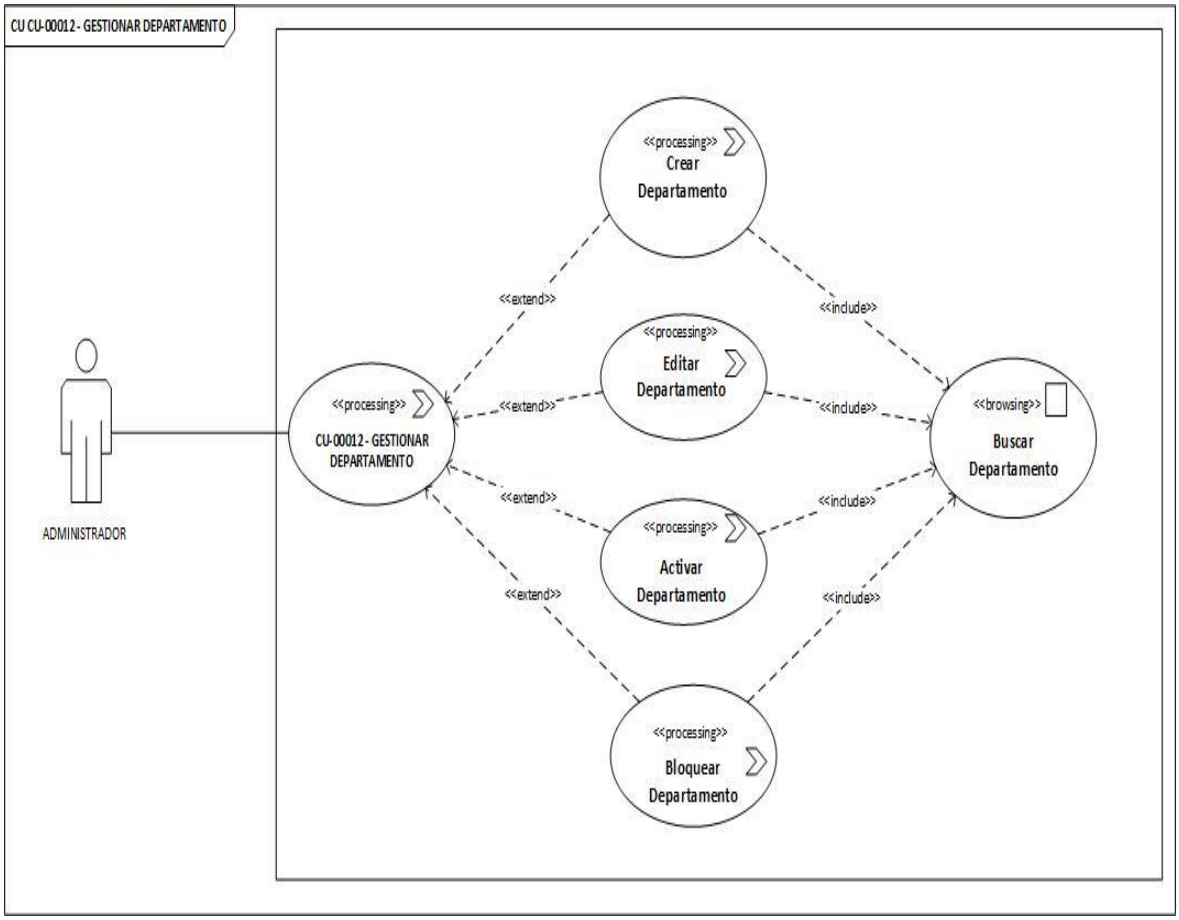


***Figura 17**. Diagrama de Caso de Uso: Gestión de Departamento*




Este diagrama de caso de uso expresa las funcionalidades para gestionar departamentos de entrega dentro de la plataforma, dentro de los cuales se encuentran las siguientes funcionalidades:

1. Crear Departamento: El administrador crea los departamentos para habilitarlos en el proceso de envió y registros del cliente.

2. Editar Departamento: El administrador edita la información general de los departamentos.

3. Activar Departamento: El administrador tiene la capacidad de activar un departamento.

4. Bloquear Departamento: El administrador tiene la capacidad de bloquear un

departamento.

**Actividad de Caso de Uso: Crear Departamento**

La actividad de “crear departamento” permite al administrador adicionar departamentos disponibles de entrega en el sistema.

1. Ingreso a la opción de gestión de departamentos

2. Ingreso de información: el administrador se encarga de rellenar la información del departamento

3. Una vez completado los campos, el usuario guarda el nuevo departamento en el sistema

***Figura 18.*** *Diagrama de Actividad de Caso de Uso: Gestión de departamentos*

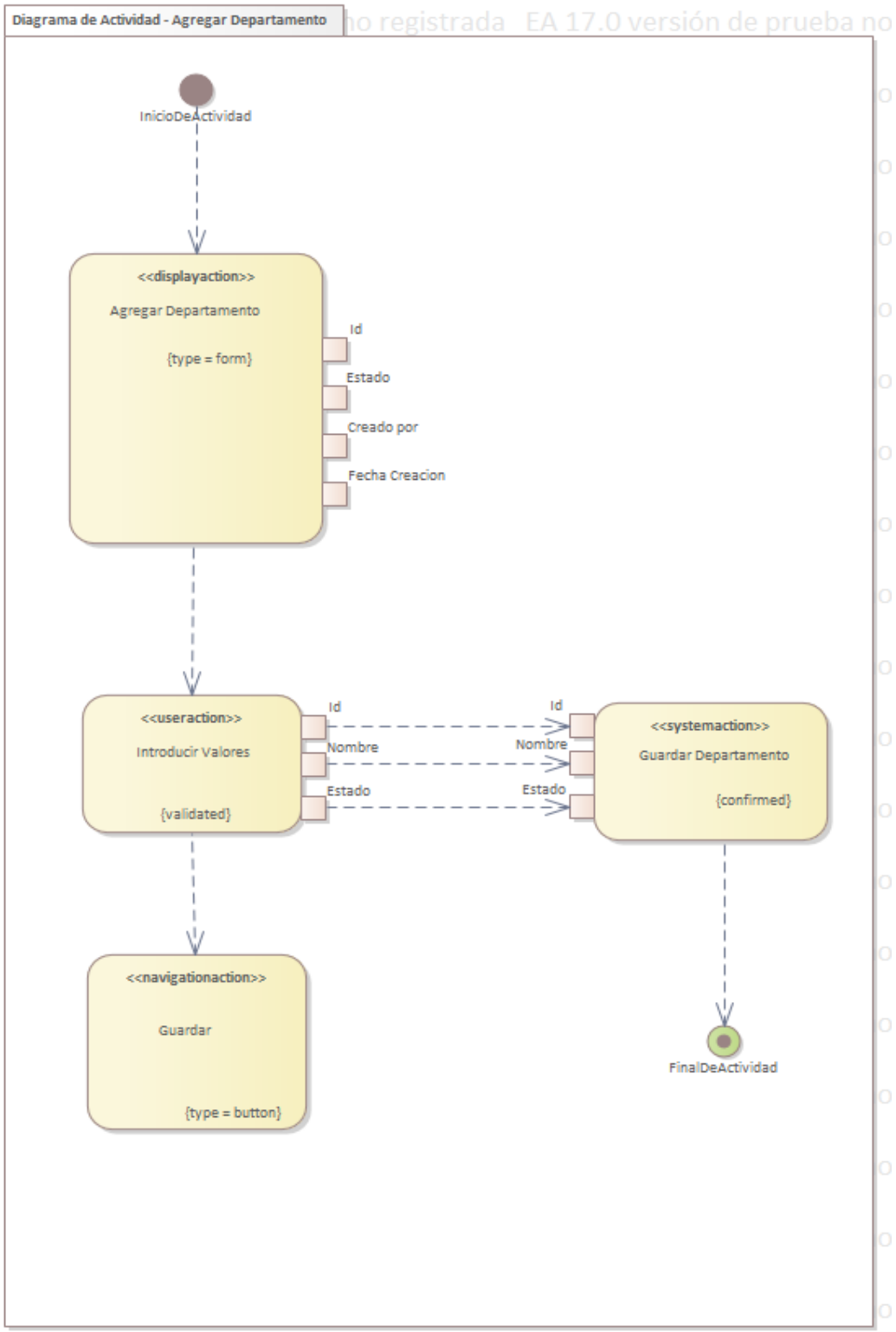
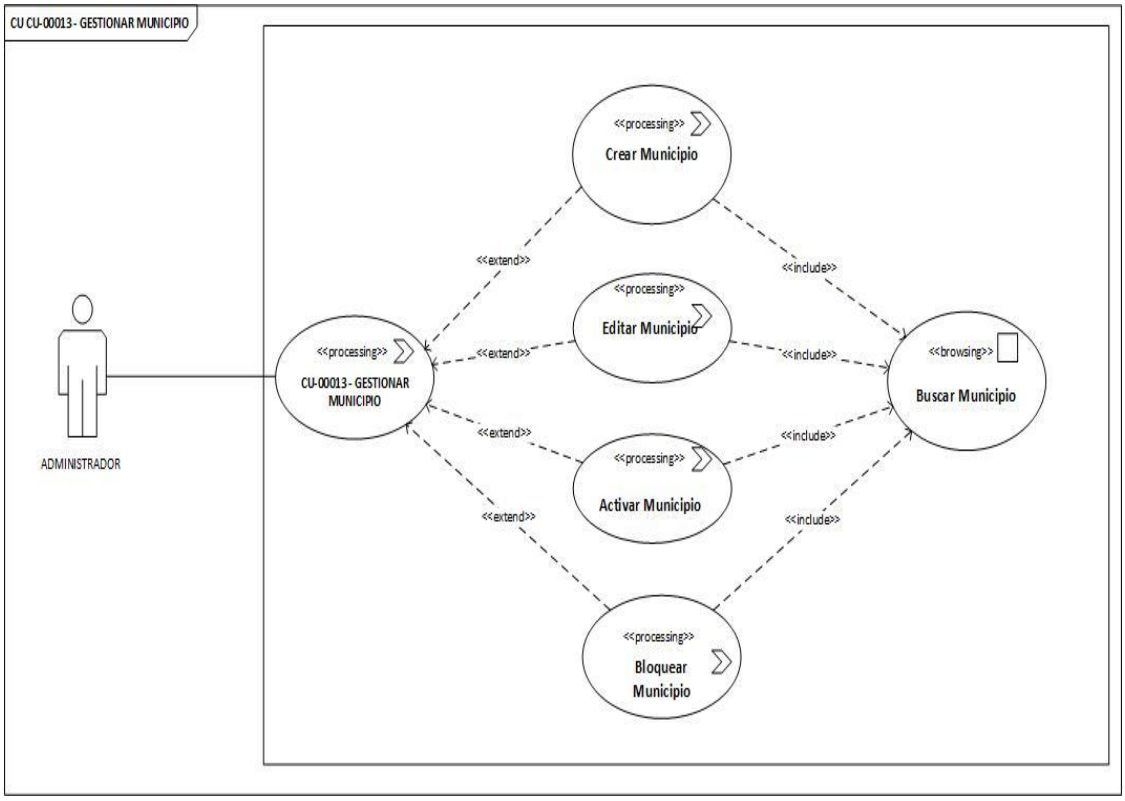


***Figura 19.*** *Diagrama de Caso de Uso: Gestión de municipios*




Este diagrama de caso de uso expresa las funcionalidades para gestionar municipios de entrega dentro de la plataforma, dentro de los cuales se encuentran las siguientes funcionalidades:

1. Crear Municipios: El administrador crea los municipios asociados a un departamento para habilitarlos en el proceso de envió y registros del cliente.

2. Editar Municipio: El administrador edita la información general de los municipios.

3. Activar Municipio: El administrador tiene la capacidad de activar un municipio.

4. Bloquear Municipio: El administrador tiene la capacidad de bloquear un municipio.

**Actividad de Caso de Uso: Crear Municipio**

La actividad de “crear municipios” permite al administrador adicionar municipios disponibles de entrega en el sistema.

1. Ingreso a la opción de gestión de municipios

2. Ingreso de información: el administrador se encarga de rellenar la información del municipio

3. Una vez completado los campos, el usuario guarda el nuevo municipio en el sistema

***Figura 20.*** *Diagrama de Actividad de Caso de Uso: Crear Municipio*

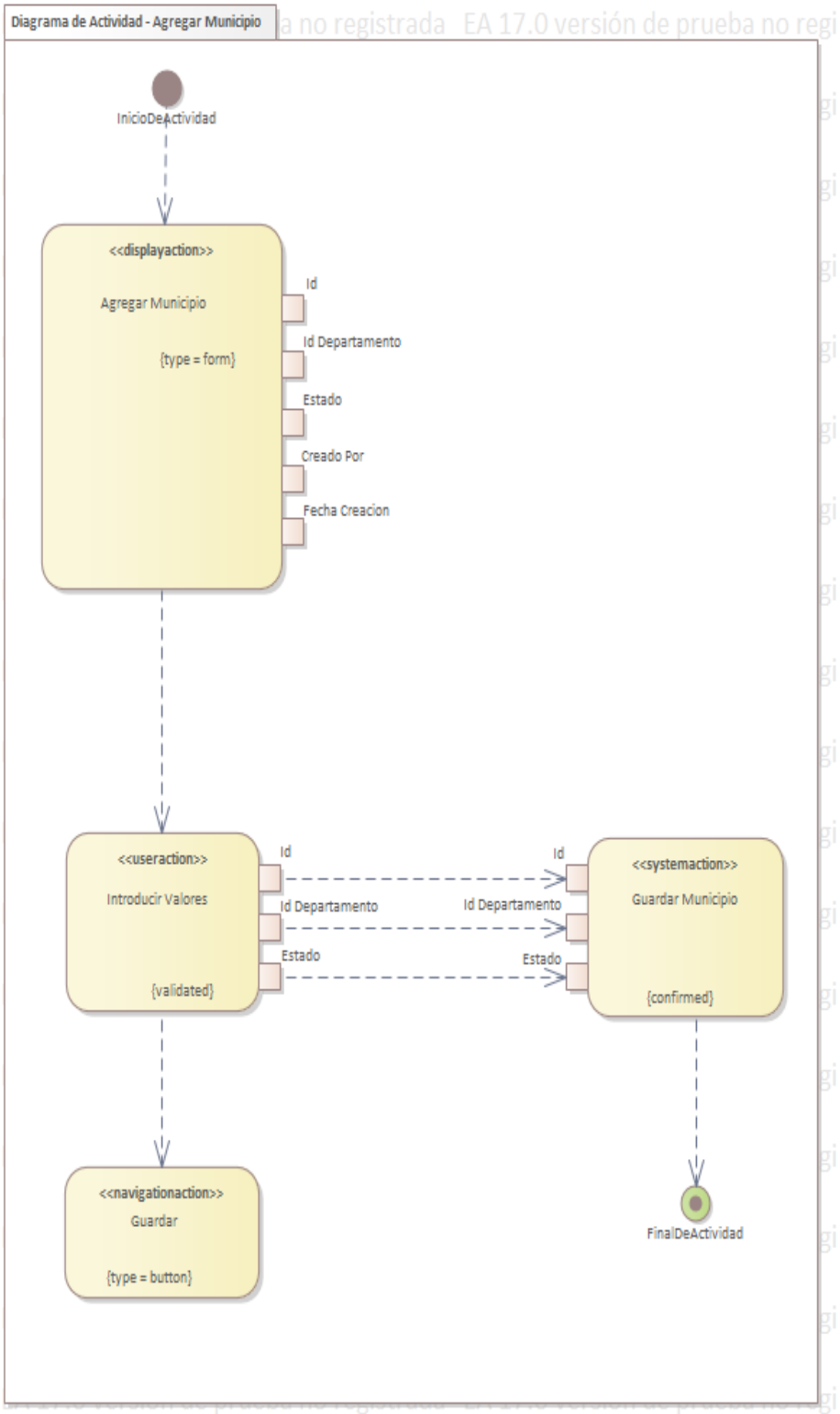
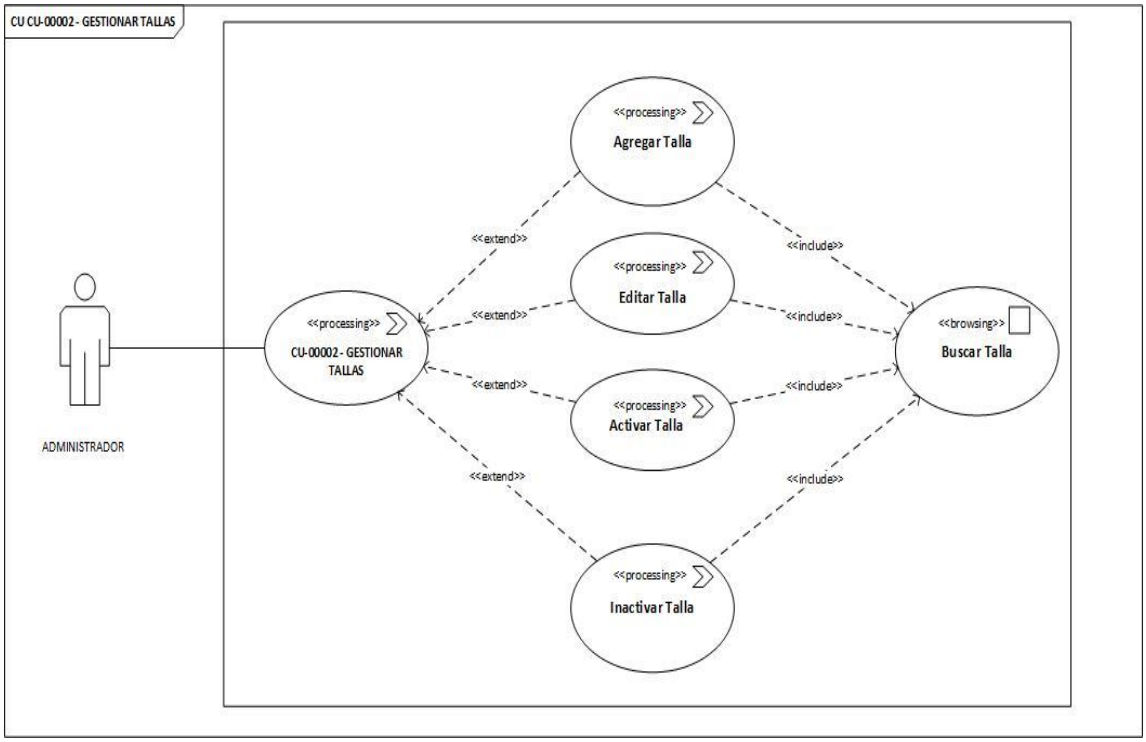


***Figura 21.*** *Diagrama de Caso de Uso: Gestión de Tallas*




Este diagrama de caso de uso expresa las funcionalidades para gestionar las tallas y presentaciones de los productos de entrega dentro de la plataforma, dentro de los cuales se encuentran las siguientes funcionalidades:

1. Crear Tallas: El administrador crea las tallas/Tipos de presentación para luego ser asociados a un producto y habilitarlos en el proceso de agregar productos al carro de compras.

2. Editar Talla: El administrador edita la información general de las tallas.

3. Activar Talla: El administrador tiene la capacidad de activar una talla.

4. Inactivar Talla: El administrador tiene la capacidad de bloquear una talla.

**Actividad de Caso de Uso: Crear Tallas**

La actividad de “crear tallas” permite al administrador adicionar tallas disponibles para los artículos.

1. Ingreso a la opción de gestión de tallas

2. Ingreso de información: el administrador se encarga de rellenar la información de las tallas

3. Una vez completado los campos, el usuario guarda la nueva talla en el sistema

***Figura 22**. Diagrama de Actividad de Caso de Uso: Agregar Nueva Clasificación*

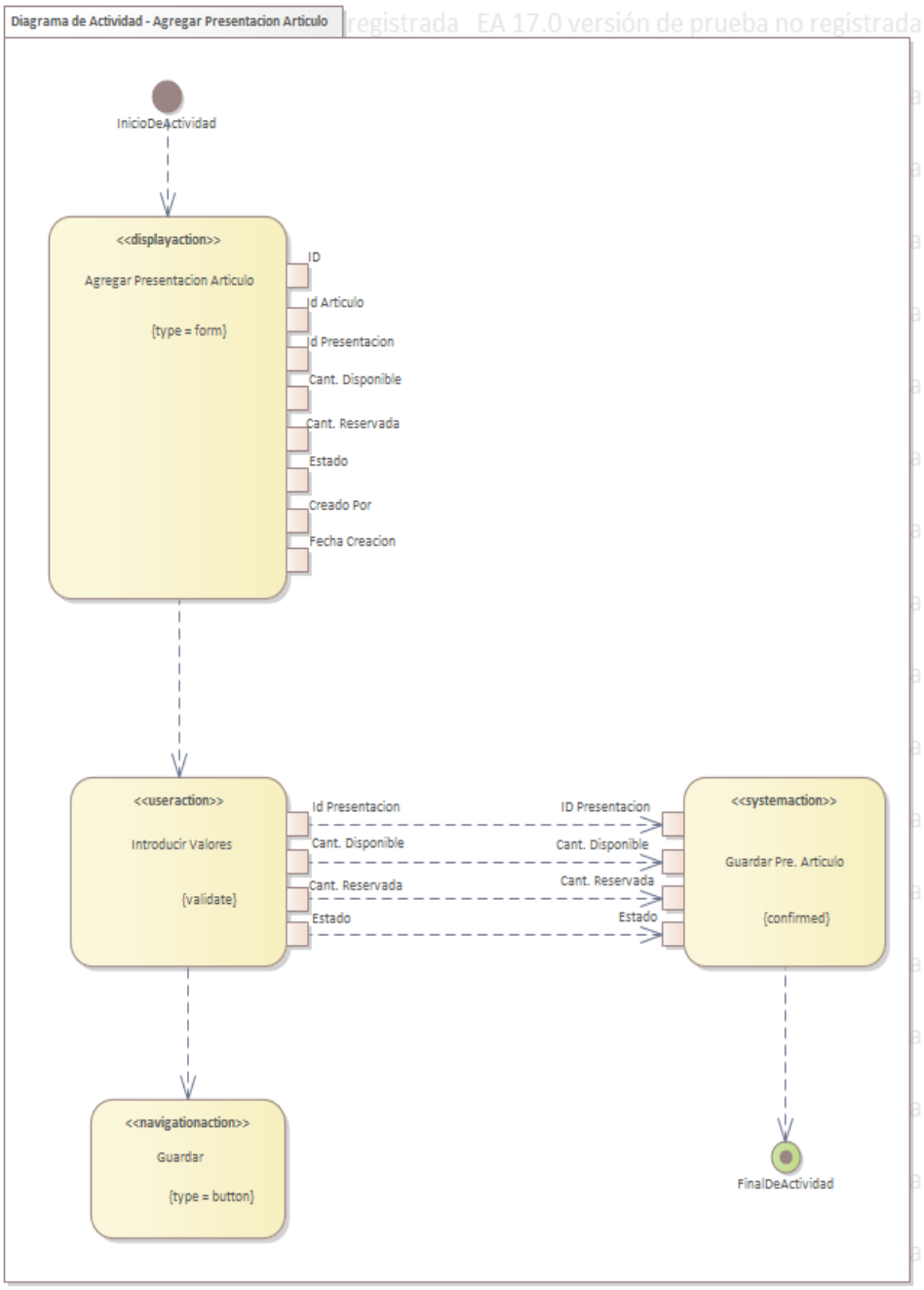
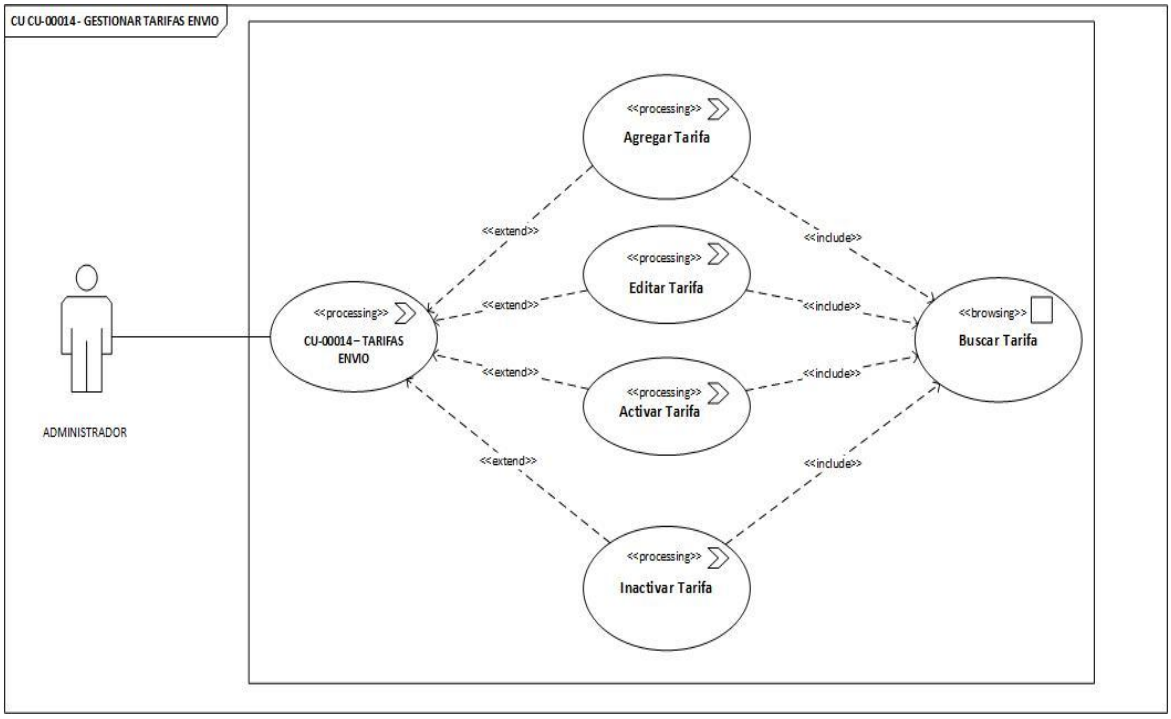


***Figura 23**. Diagrama de Caso de Uso: Gestión de Tarifas de Envió*




Este diagrama de caso de uso describe los pasos seguidos por el administrador para gestionar las tarifas de envió, que posteriormente serán utilizadas en el proceso de generación de pedido mismas que dependen del departamento y municipio que el cliente indica como dirección de entrega, las funciones realizadas son las siguientes:

1. Agregar Tarifa: El administrador adiciona tarifas en el sistema las cuales son usadas para calcular el costo del envío.

2. Editar Tarifa: Si es necesario, el administrador puede modificar las tarifas.

3. Activar Tarifa: El administrador puede activar o inactivar una tarifa.

**Actividad de Caso de Uso: Agregar Tarifa de Envió**

La actividad de “crear tarifas” permite al administrador adicionar tarifas disponibles que ayuda a calcular el costo del envío.

1. Ingreso a la opción de gestión de tarifas

2. Ingreso de información: el administrador se encarga de rellenar la información de las tarifas

3. Una vez completado los campos, el usuario guarda la nueva tarifa en el sistema

***Figura 24.*** *Diagrama de Actividad de caso de Uso: Agregar Nueva tarifa de envío*

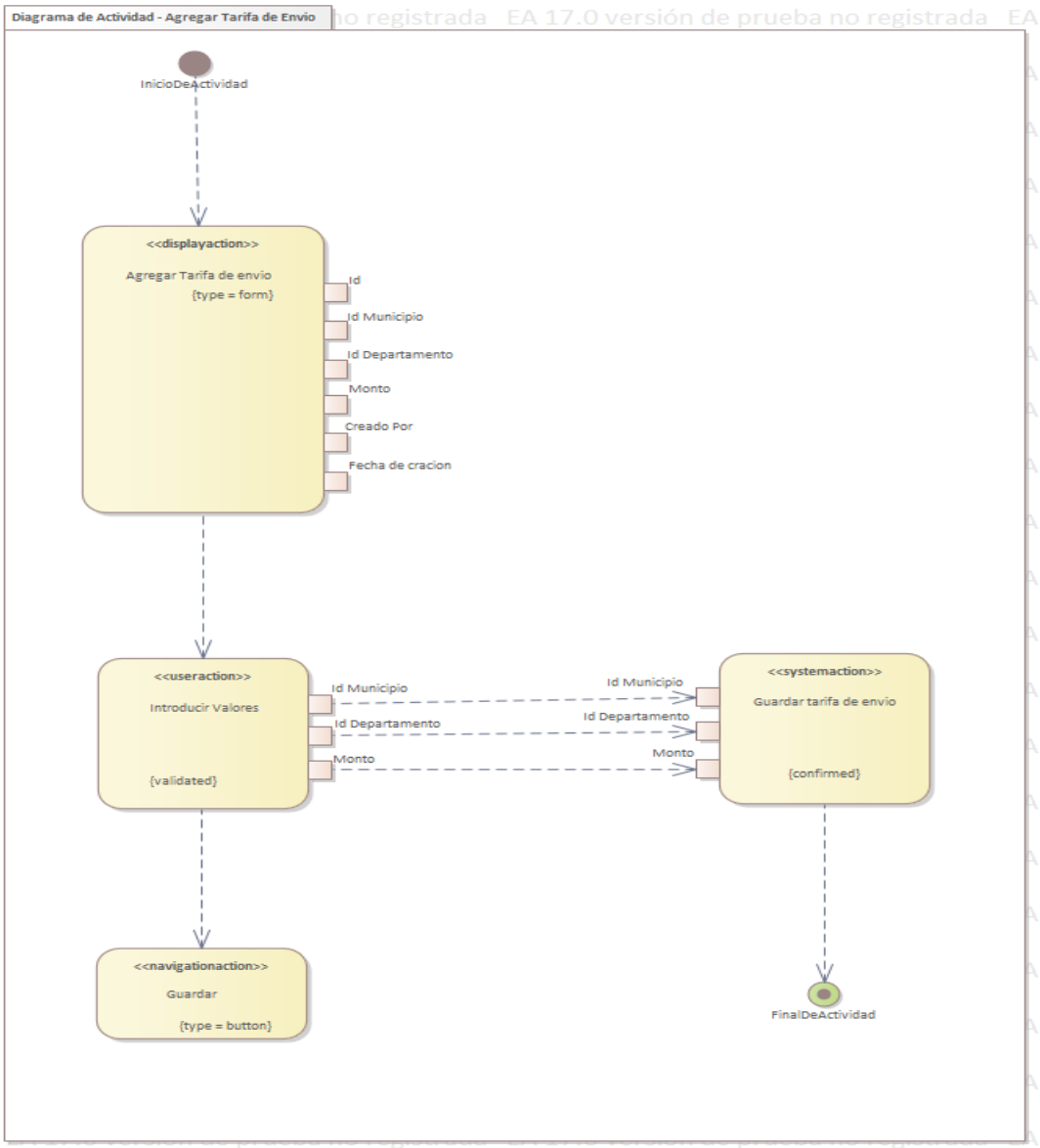
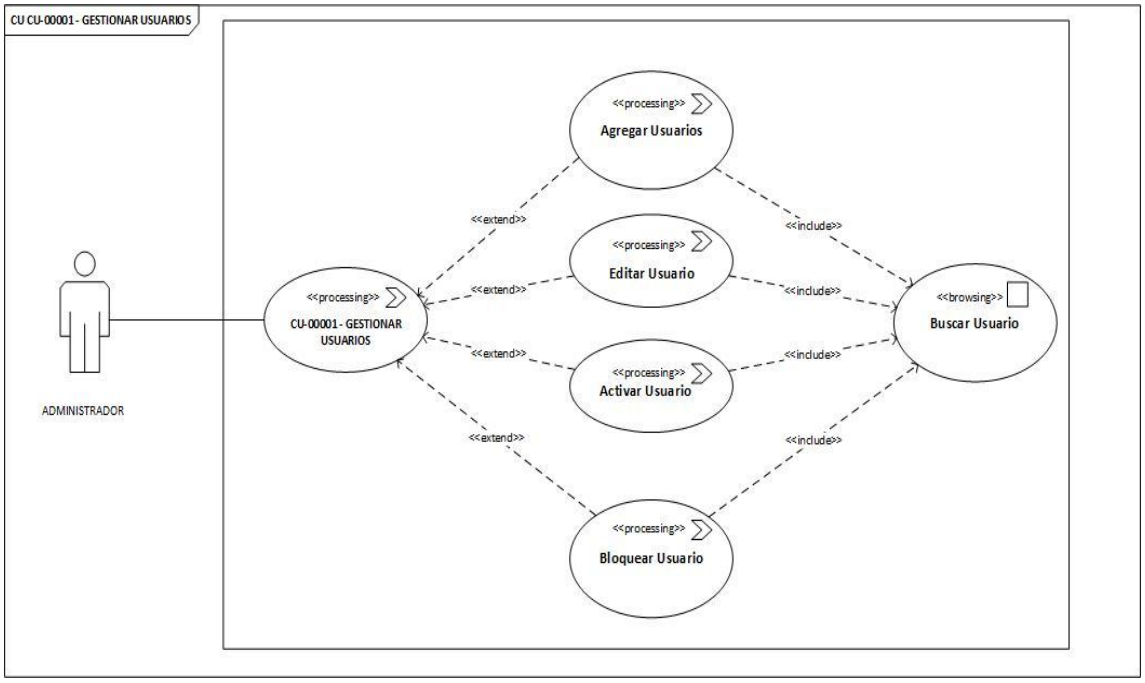


***Figura 25.*** *Diagrama de Caso de Uso: Gestión de Usuarios*




Este diagrama de caso de uso describe los pasos seguidos por el administrador para la gestión de usuarios de la plataforma.

1. Agregar Usuarios: El administrador puede adicionar usuarios en la plataforma que pueden ser administradores, clientes o repartidores.

2. Editar Usuario: Si fuese necesario el administrador puede actualizar los usuarios registrados en la plataforma.

3. Activar Usuario: El administrador puede activar un usuario.

4. Bloquear Usuario: El administrador puede bloquear un usuario del sistema.

**Actividad de Caso de Uso: Crear Usuario**

La actividad “Crear Usuario” le permite al administrador adicionar nuevos usuarios en la plataforma

1. Ingreso a la opción de gestión de usuarios

2. Ingreso de información: el administrador se encarga de rellenar la información de los usuarios

3. Una vez completado los campos, el usuario guarda la nueva tarifa en el sistema

***Figura 26.*** *Diagrama de Actividad de Caso de Uso: Crear Usuario*

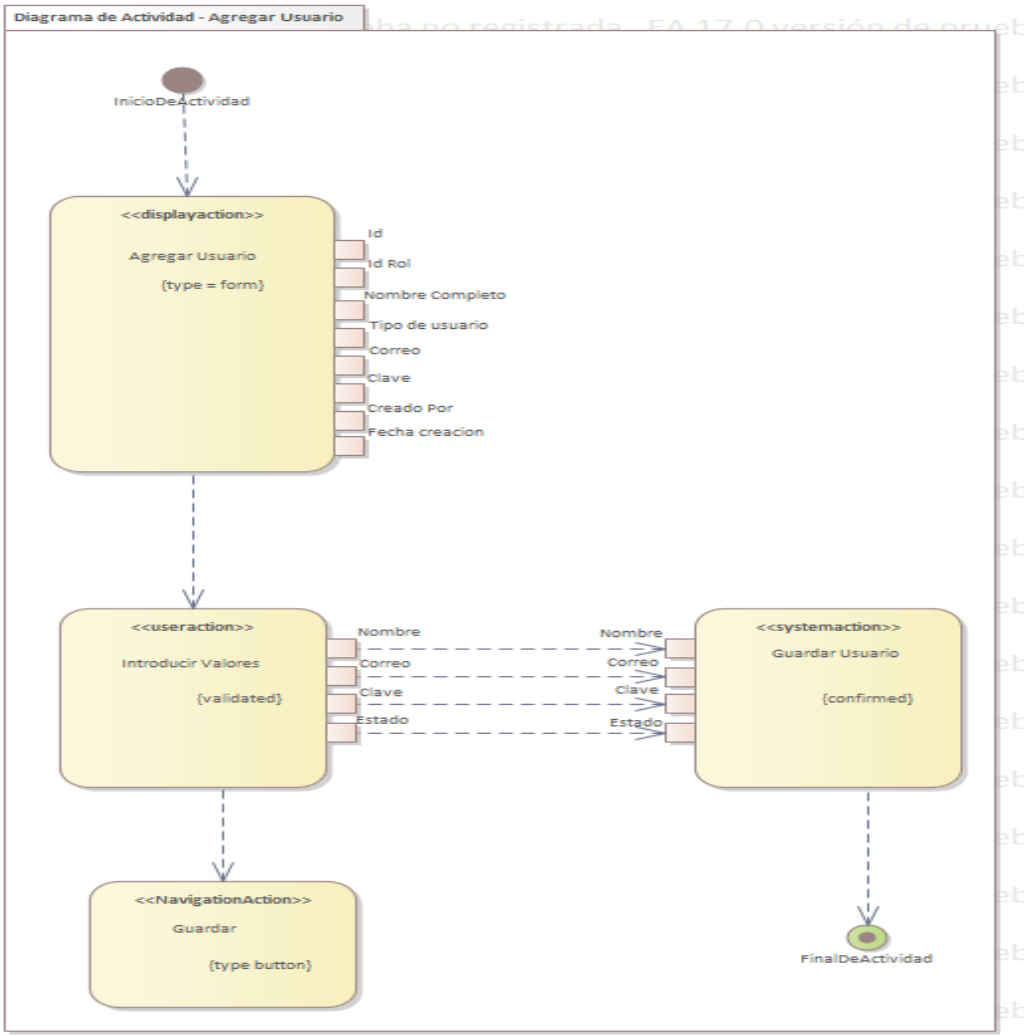
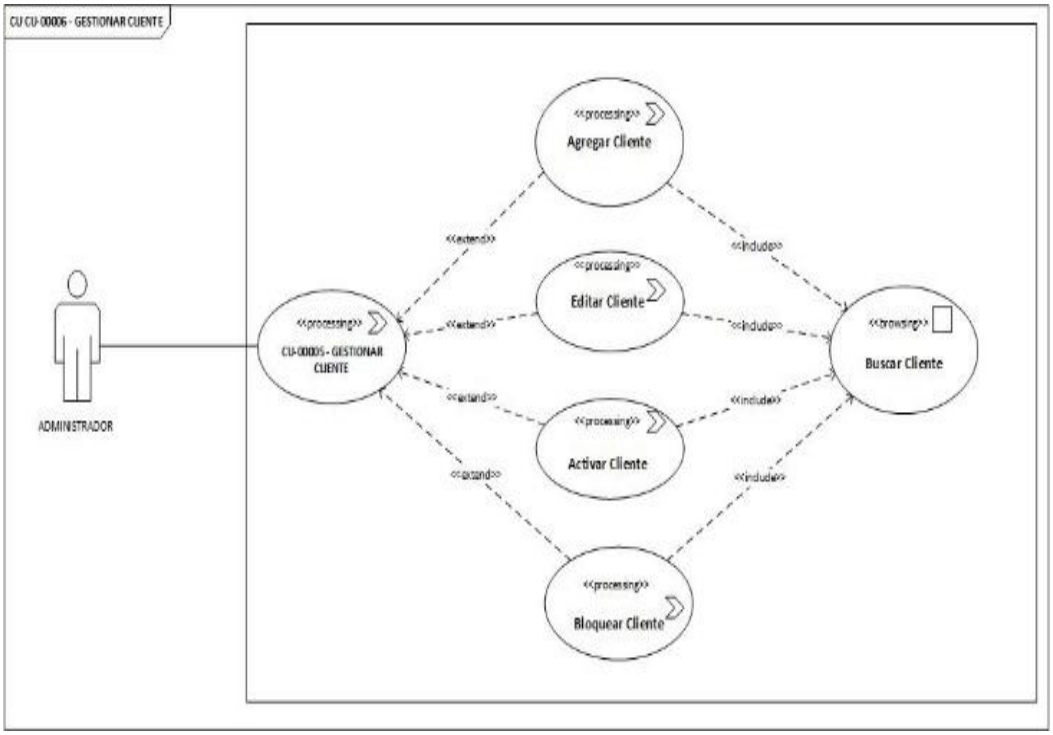


***Figura 27.*** *Diagrama de Caso de Uso: Gestión de cliente*




**Fuente**: Elaboración Propia

Este diagrama de caso de uso describe los pasos seguidos por el administrador para la gestión de clientes de la plataforma.

1. Agregar Cliente: El administrador puede adicionar clientes en la plataforma.

2. Editar Cliente: Si fuese necesario el administrador puede actualizar los clientes registrados en la plataforma.

3. Activar Cliente: El administrador puede activar un usuario.

**Actividad de Caso de Uso: Agregar Cliente**

La actividad “Agregar Cliente” le permite al administrador adicionar nuevos usuarios en la plataforma

1. Ingreso a la opción de gestión de clientes

2. Ingreso de información: el administrador se encarga de rellenar la información de los clientes adicionados manualmente

3. Una vez completado los campos, el usuario guarda el nuevo cliente en el sistema

***Figura 28.*** *Diagrama de Actividad de Caso de Uso: Agregar Cliente*

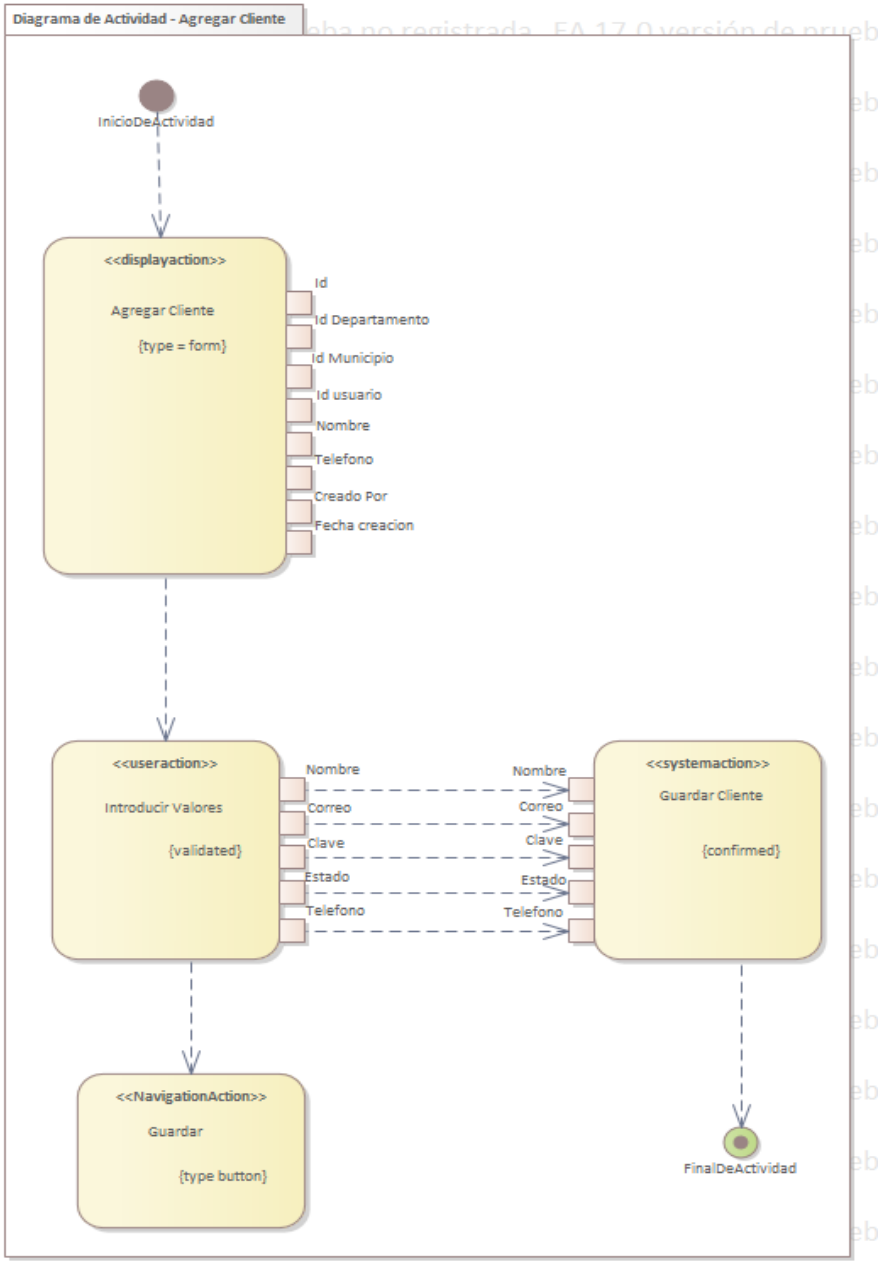
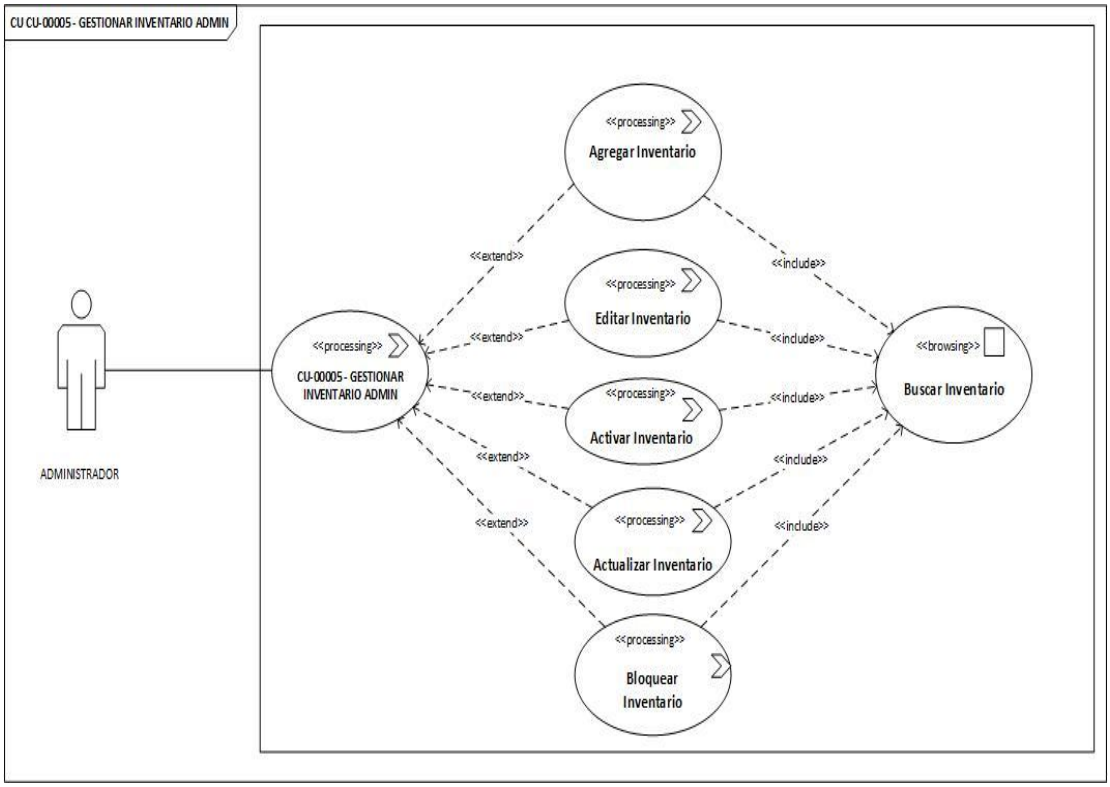


***Figura 29.*** *Diagrama Caso de Uso: Gestión de Inventario*

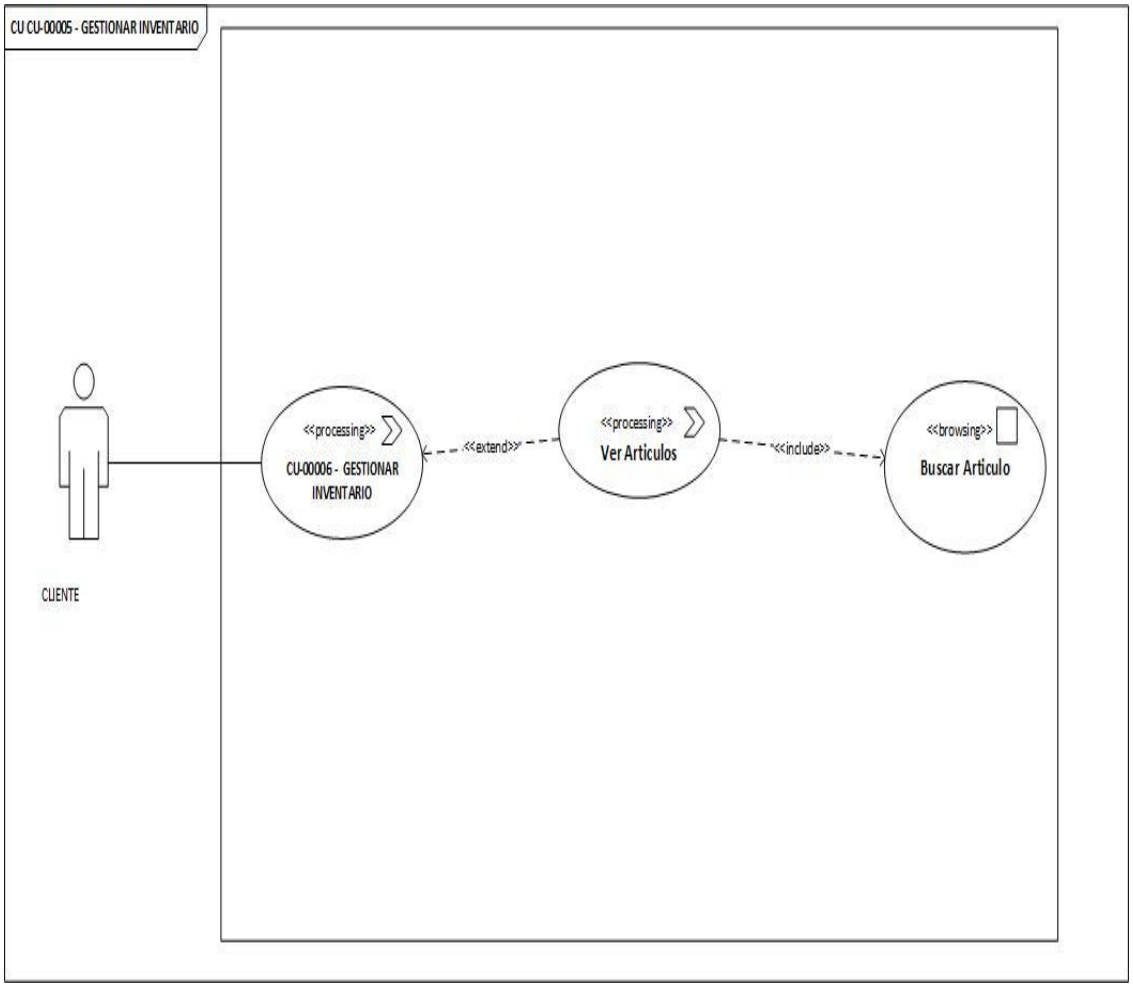



**Fuente**: Elaboración Propia

Este diagrama de caso de uso describe los pasos seguidos por el administrador para la gestión de inventario

1. Agregar Inventario
2. Editar Inventario
3. Activar Inventario
4. Inactivar Inventario

***Figura 30.*** *Diagrama de Caso de Uso: Gestión Inventario Cliente*




**Fuente**: Elaboración Propia

Este diagrama de caso de uso describe los pasos seguidos por el cliente para poder visualizar el catálogo de artículos por categoría

1. Ver Artículos: El cliente puede visualizar los artículos asociados a una categoría

## 8.4. **Diagrama de Contenido**

El diagrama de contenido tiene como finalidad representar la estructura lógica de los datos gestionados por el sistema, haciendo énfasis en las entidades que lo componen, sus atributos esenciales y las relaciones que se establecen entre ellas. Esta

representación permite comprender de manera abstracta cómo se organiza la información y cómo fluye entre los distintos módulos funcionales del sistema.

En el presente sistema, el modelo de contenido fue elaborado a partir de las entidades definidas en la capa de persistencia del backend. Estas entidades incluyen, entre otras, usuarios, artículos, pedidos, facturas, clientes y presentaciones, las cuales interactúan mediante relaciones uno a muchos y muchos a muchos. La definición clara de estas estructuras y sus asociaciones resulta fundamental para garantizar la integridad de los datos y el correcto funcionamiento de los procesos del sistema.

**1. Entidades Principales**

**Usuario**

Atributos:

- id
- email
- name
- role
- is_active

Relaciones:

- 1 -- N ShippingCart: Un usuario puede poseer varios carritos de compra.
- 1 -- N Order: Un usuario puede registrar o gestionar múltiples pedidos.

**Cliente**

Atributos:

- id
- full_name
- dni
- email
- phone
- address
- is_active

Relaciones:

- N -- 1 User: Un cliente es creado por un usuario.
- 1 -- N Order: Un cliente puede realizar múltiples pedidos.
- N -- 1 Municipality: Un cliente reside en un municipio asociado a un departamento.
- N -- 1 Department: Asociación indirecta mediante el municipio.

**Departamento**

Atributos:

- id
- description
- is_active

Relaciones:

- 1 -- N Municipality: Un departamento contiene múltiples municipios.
- 1 -- N Order: Los pedidos registran un departamento de entrega.
- 1 -- N FinancialEntityDetail: La información financiera puede incluir datos por departamento.

**Municipio**

Atributos:

- id
- description
- is_active
- id_department

Relaciones:

- N -- 1 Department: Un municipio pertenece a un departamento.
- 1 -- N ShippingRate: Cada municipio puede tener tarifas de envío asociadas.
- 1 -- N Order: Los pedidos especifican un municipio de entrega.

**Tarifa de Envío (ShippingRate)**

Atributos:

- id
- rate_in_cents

Relaciones:

- N -- 1 Municipality: La tarifa se asocia a un municipio específico para el cálculo del envío.

**2. Productos, Variantes e Inventario**

**Categoría de Producto (ProductCategory)**

Atributos:

- id
- description
- is_active

Relaciones:

- 1 -- N Product: Cada categoría agrupa múltiples productos.

**Producto (Product)**

Atributos:

- id
- sku
- name
- description
- price
- is_active

Relaciones:

- N -- 1 ProductCategory: Un producto pertenece a una categoría.
- 1 -- N ProductVariant: Un producto puede tener múltiples variantes.
- 1 -- N StockMovement: Los movimientos de inventario se asocian al producto.
- 1 -- N ShippingCartDetail: Un producto puede aparecer en múltiples carritos.
- 1 -- N OrderDetail: Un producto puede ser parte de múltiples pedidos.

**Variante del Producto (ProductVariant)**

Atributos:

- id
- name
- stock

Relaciones:

- N -- 1 Product: Una variante pertenece a un producto.
- 1 -- N ShippingCartDetail: Puede estar incluida en múltiples carritos.
- 1 -- N OrderDetail: Puede formar parte de varios pedidos.
- 1 -- N StockMovement: Registra movimientos específicos de inventario.

**Movimiento de Inventario (StockMovement)**

Atributos:

- id
- quantity
- movement_type
- balance_after_movement
- notes

Relaciones:

- N -- 1 Product
- N -- 1 ProductVariant

Representa entradas y salidas del inventario, reflejando el stock disponible.

**3. Carrito de Compra y Detalles**

**Carro de Compra (ShippingCart)**

Atributos:

- id
- is_active
- invoiced
- total_item
- sub_total

Relaciones:

- N -- 1 User
- 1 -- N ShippingCartDetail

**Detalle del Carrito (ShippingCartDetail)**

Atributos:

- id
- quantity
- unit_price
- sub_total

Relaciones:

- N -- 1 ShippingCart
- N -- 1 Product
- N -- 1 ProductVariant
- N -- 1 OptionValue (opciones agregadas)

**OptionValue**

Atributos:

- id
- name

Relaciones:

- 1 -- N ShippingCartDetail: Una opción puede asociarse a múltiples detalles de carrito.
- 1 -- N OrderDetail: También se refleja en los pedidos.

**4. Pedidos**

**Pedido (Order)**

Atributos:

- id
- tracking_code
- customer_name
- customer_address
- contact_phone
- total_item
- sub_total
- shipping_fee
- tax_amount
- total

Relaciones:

- N -- 1 Customer

- N -- 1 User
- 1 -- N OrderDetail
- 1 -- N OrderPayment
- 1 -- N OrderStateLog
- N -- 1 OrderState: Representa su estado actual
- N -- 1 Department: Departamento de entrega
- N -- 1 Municipality: Municipio de entrega

**Detalle del Pedido (OrderDetail)**

Atributos:

- id
- quantity
- unit_price
- sub_total

Relaciones:

- N -- 1 Order
- N -- 1 Product
- N -- 1 ProductVariant
- N -- 1 OptionValue

**Estado del Pedido (OrderState)**

Atributos:

- id
- code
- name
- color

Relaciones:

- 1 -- N Order: Estado actual
- 1 -- N OrderStateLog: Historial de estados

**Historial de Estado (OrderStateLog)**

Atributos:

- id
- observation
- is_active

Relaciones:

- N -- 1 Order
- N -- 1 OrderState

**5. Pagos y Entidades Financieras**

**Método de Pago (PaymentMethod)**

Atributos:

- id
- code
- description
- is_active

Relaciones:

- 1 -- N OrderPayment

**Pago del Pedido (OrderPayment)**

Atributos:

- id
- account_number
- reference_number
- currency
- amount

Relaciones:

- N -- 1 PaymentMethod
- N -- 1 Order
- N -- 1 FinancialEntity

**Entidad Financiera (FinancialEntity)**

Atributos:

- id
- description
- is_active

Relaciones:

- 1 -- N FinancialEntityDetail
- 1 -- N OrderPayment

**Detalle de Entidad Financiera (FinancialEntityDetail)**

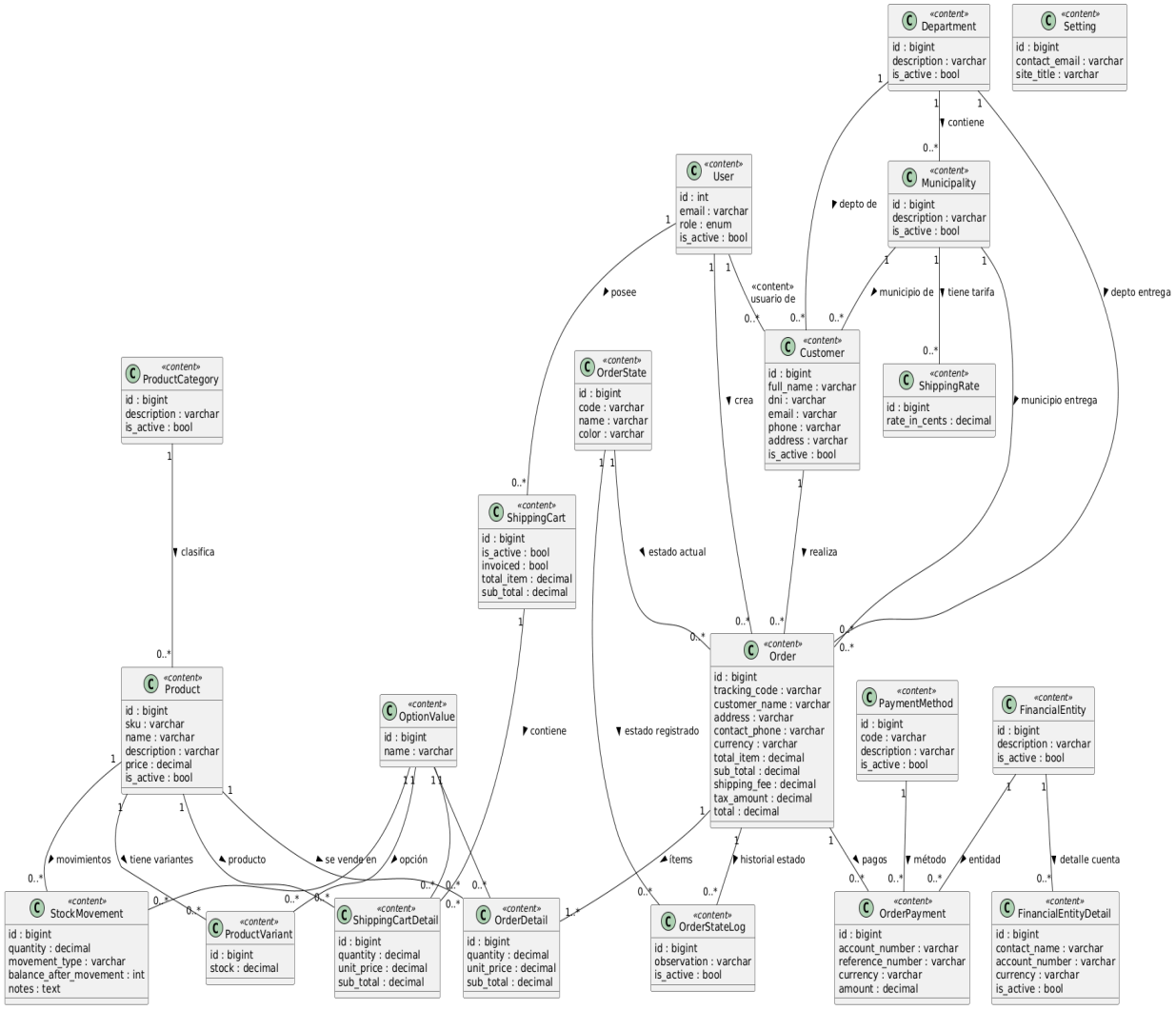
Atributos:

- id
- contact_name
- account_number
- currency

Relaciones:

- N -- 1 FinancialEntity

***Figura 31.*** *Diagrama de Contenido*

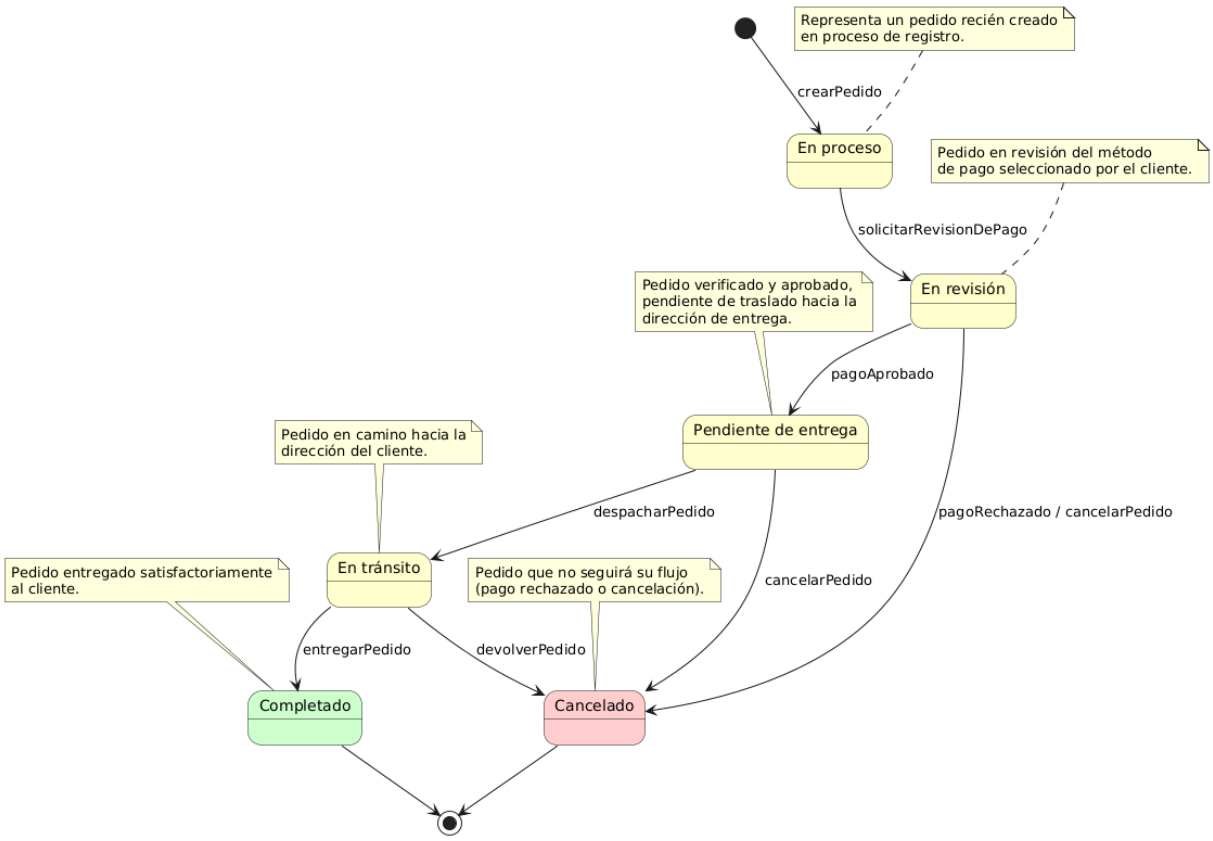



## 8.5. **Diagrama de cambio de estado**

Los diagramas de cambio de estado permiten representar gráficamente el comportamiento dinámico de las entidades del sistema ante determinados eventos. A través de estos diagramas, se visualizan los distintos estados por los que puede transitar una entidad y las condiciones que provocan dichos cambios. Esta representación resulta especialmente útil para modelar procesos que involucran ciclos de vida complejos, como la gestión de pedidos o el seguimiento del estado de una

factura, facilitando así la comprensión y validación del comportamiento esperado del sistema.

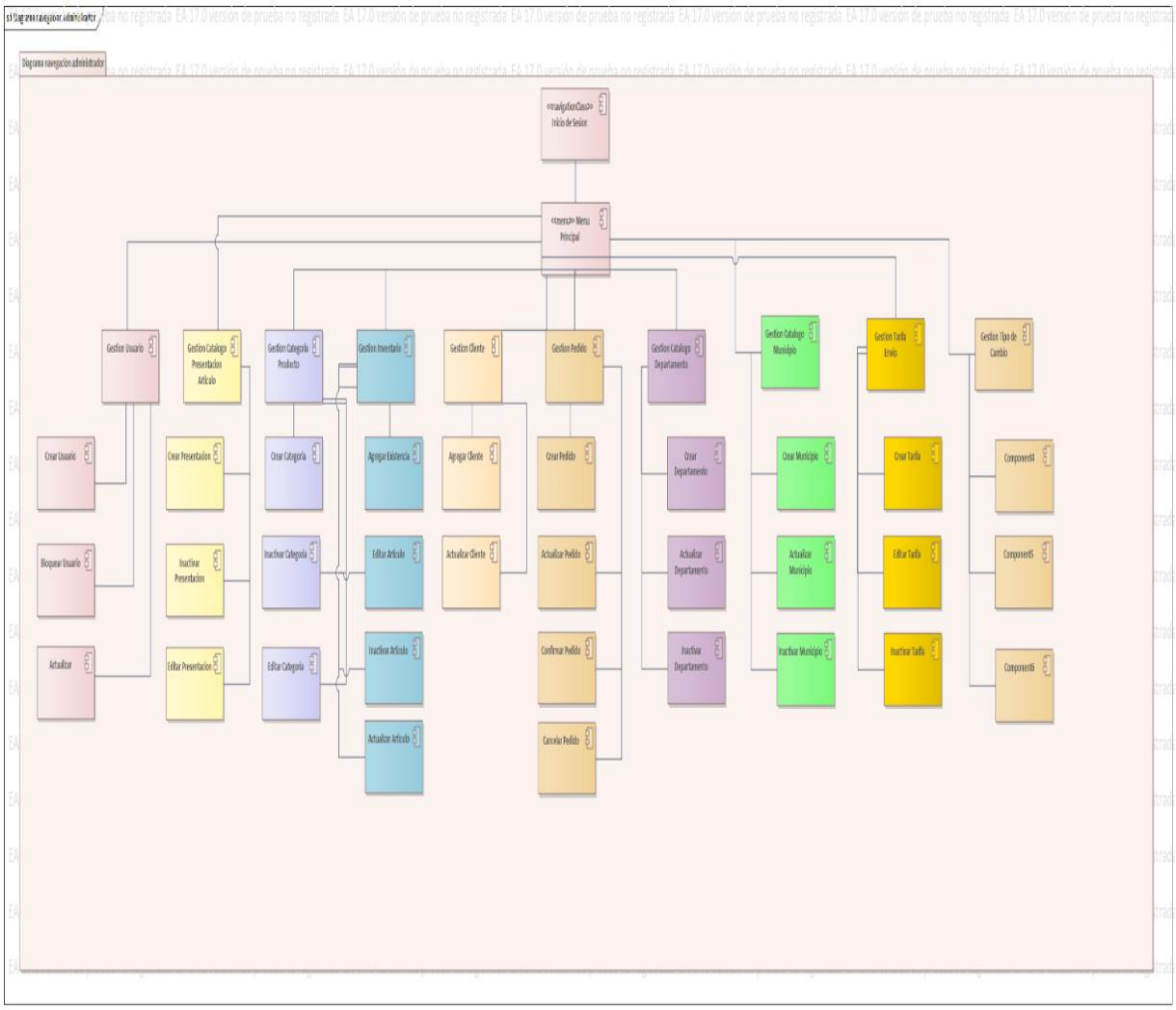
***Figura 32.*** *Diagrama cambio de estado en "gestión de pedido"*




## 8.6. **Diagrama de Navegación**

El Diagrama de Navegación representa las posibles rutas de acceso entre las diferentes vistas o páginas del sistema, permitiendo visualizar cómo los usuarios pueden desplazarse a través de la aplicación web. En el contexto de la metodología UWE, este diagrama es fundamental para definir la estructura de enlaces y el comportamiento de navegación entre las distintas interfaces. Su elaboración facilita la organización lógica del contenido y contribuye a garantizar una experiencia de usuario coherente e intuitiva, alineada con los casos de uso definidos en etapas previas del diseño.

***Figura 33.*** *Diagrama de Navegación Administración*



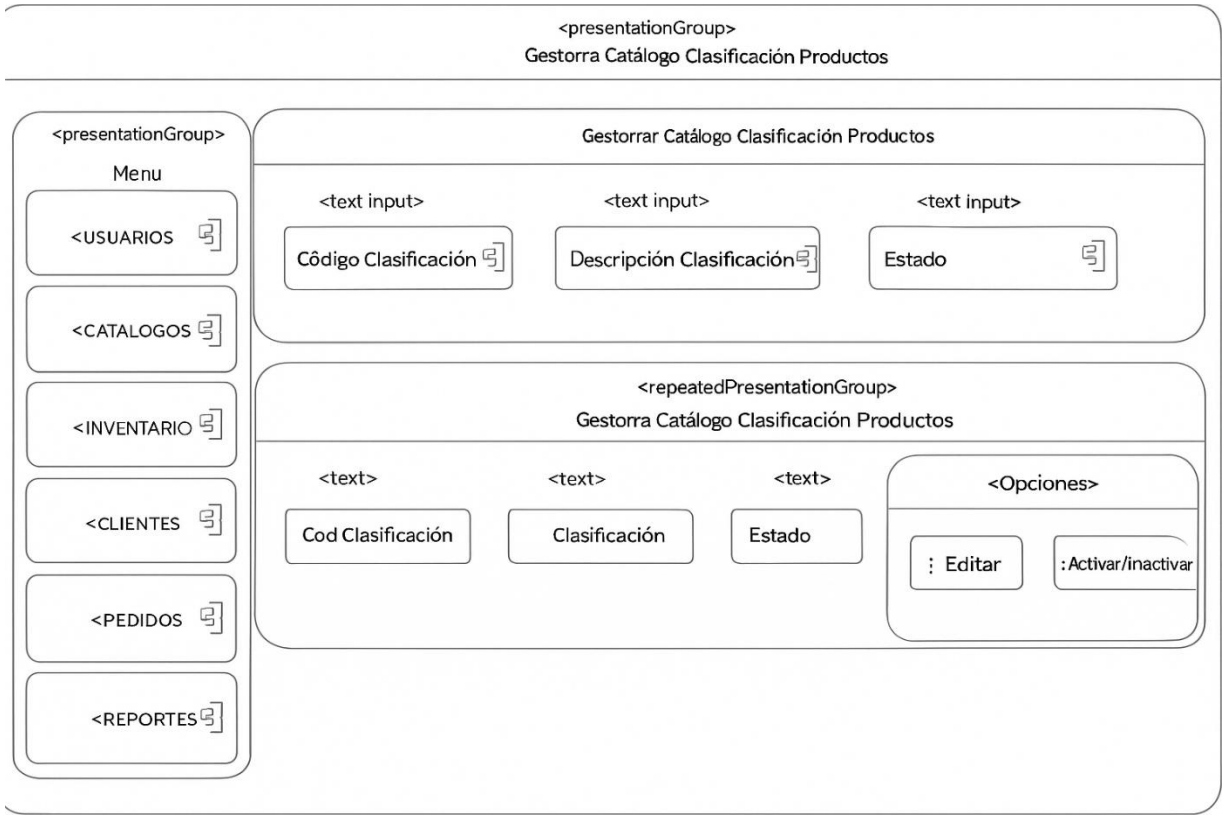
## 8.7. **Diagrama de Presentación**

El diagrama de presentación, conforme a la metodología UWE (Unified Web Engineering), tiene como propósito modelar la organización y disposición de los elementos visuales en la interfaz de usuario del sistema web. Este tipo de diagrama permite representar las vistas disponibles, los componentes que las conforman y la forma en que se estructuran para facilitar la interacción del usuario con las funcionalidades del sistema. Su elaboración contribuye a garantizar la coherencia entre el diseño de la interfaz y la lógica de navegación, favoreciendo una experiencia de uso clara, intuitiva y centrada en las necesidades del usuario final.

**Diagrama de Presentación Gestión de Clasificación Producto**

El diagrama de presentación de gestión de usuario describe como un administrador interactúa con la pantalla de gestión de inventario. En esta interfaz, el administrador visualiza el listado de artículos registrados en la plataforma como también botones para actualizar o activar e inactivar un registro, permite una opcion para ver el detalle de las tallas asociadas al artículo.

***Figura 34.*** *Diagrama de Presentación Gestión de Clasificación Producto*

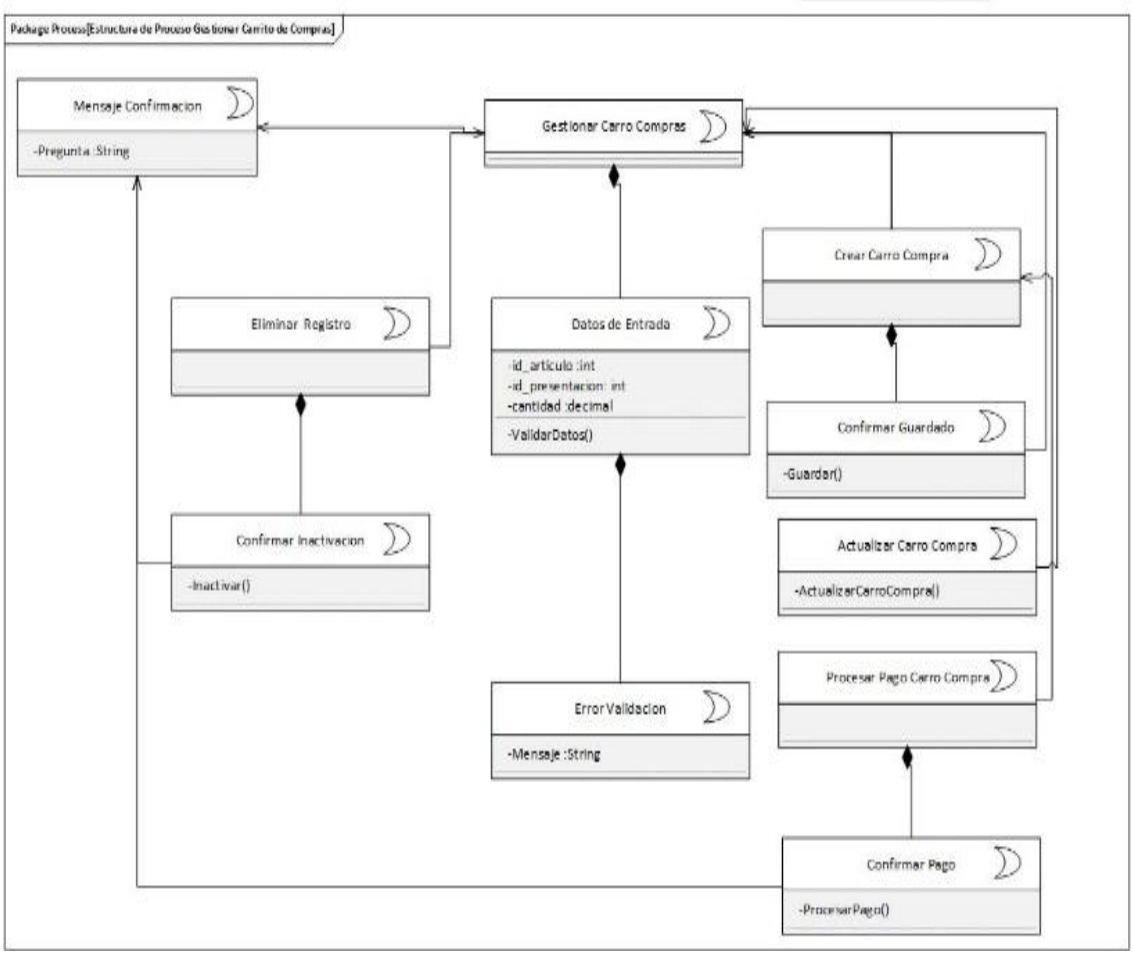


## 8.8. **Diagrama Estructura de Proceso**

El Diagrama de Estructura de Proceso, el cual forma parte de la metodología UWE (UML-based Web Engineering) y tiene como objetivo modelar el flujo de actividades que componen los procesos interactivos del sistema. Este diagrama permite representar, de forma estructurada, las acciones que pueden llevar a cabo los usuarios en la plataforma, así como sus transiciones y condiciones asociadas. Su elaboración contribuye a identificar claramente los pasos involucrados en los casos de uso más relevantes, facilitando así la comprensión y validación del comportamiento funcional del sistema propuesto.

### Diagrama Estructura de Proceso Gestión Carro de Compras

***Figura 35.*** *Diagrama Estructura Proceso Gestión Carrito de Compras*



1. Mensaje Confirmación

- **Descripción**: Muestra una pregunta al usuario para confirmar una acción (como eliminar un artículo o procesar una compra).
- **Dato**: pregunta: String

2. Gestionar Carro Compras

- **Descripción**: Componente principal que controla todo el flujo del carrito de compras, incluyendo creación, actualización, eliminación y procesamiento del pago.

3. Eliminar Registro

- **Descripción**: Proceso encargado de eliminar un artículo del carrito de compras.

4. Confirmar Instrucción

- **Descripción**: Confirma la acción del usuario antes de proceder con la eliminación o alguna otra instrucción crítica.
- **Método**: Instruccion()

5. Datos de Entrada

- **Descripción**: Define los datos requeridos para gestionar el carrito, como:
  - id_articulo: int
  - cantidad: decimal
  - validarDatos(): función para validar que los datos estén correctos.

6. Error Validación

- **Descripción**: Se activa cuando los datos de entrada no pasan la validación. Muestra un mensaje de error.

- **Dato**: mensaje: String

7. Crear Carro Compra

- **Descripción**: Proceso para iniciar la creación de un nuevo carrito de compras.

8. Confirmar Guardado

- **Descripción**: Confirma que el carrito fue creado correctamente y los datos fueron guardados.

- **Método**: Guardado()

9. Actualizar Carro Compra

- **Descripción**: Permite modificar el contenido del carrito ya existente.

- **Método**: ActualizarCarroCompra()

10. Procesar Pago Carro Compra

- **Descripción**: Realiza las acciones necesarias para procesar el pago de lo que contiene el carrito.

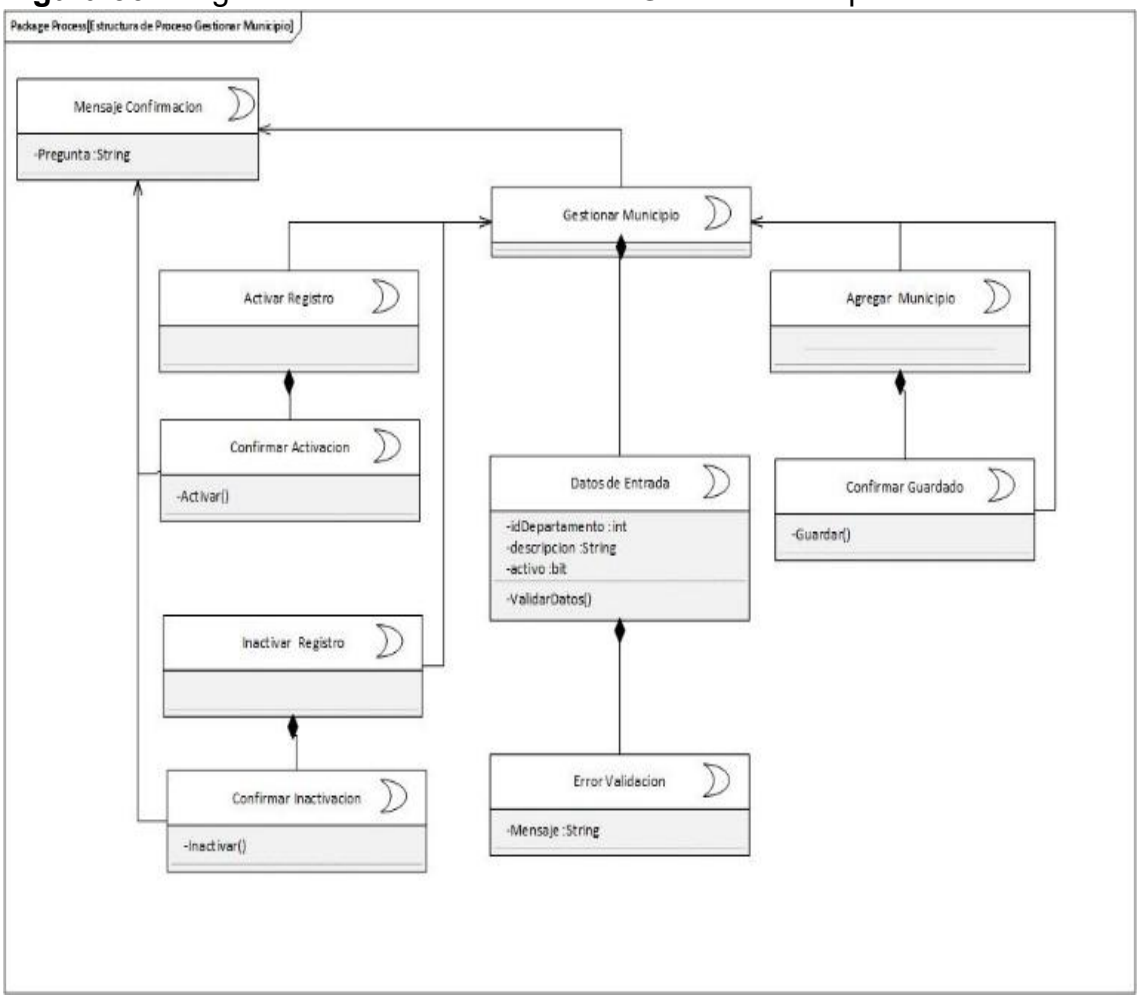
11. Confirmar Pago

- **Descripción**: Confirma que el pago se ha realizado exitosamente.

- **Método**: ProcesarPago()

**Diagrama Estructura de Proceso Gestión de Municipios**

**Figura 36.** Diagrama Estructura de Proceso Gestión Municipios




Mensaje Confirmación

- **Descripción**: Solicita al usuario una confirmación previa para ejecutar una acción crítica, como activar, inactivar o guardar un municipio.

- **Dato**: pregunta: String

2. Gestionar Municipio

- **Descripción**: Proceso principal que engloba todas las operaciones relacionadas con los municipios: agregar, activar, inactivar y validar datos.

3. Activar Registro

- **Descripción**: Permite reactivar un municipio que anteriormente fue inactivado.

4. Confirmar Activación

- **Descripción**: Solicita confirmación antes de proceder con la activación del municipio.
- **Método**: Activar()

5. Inactivar Registro

- **Descripción**: Permite desactivar un municipio activo, probablemente por razones administrativas o de mantenimiento.

6. Confirmar Inactivación

- **Descripción**: Solicita confirmación del usuario antes de inactivar el municipio.
- **Método**: Inactivar()

7. Agregar Municipio

- **Descripción**: Permite ingresar un nuevo municipio al sistema.

8. Confirmar Guardado

- **Descripción**: Confirma que el nuevo municipio fue guardado correctamente en la base de datos.
- **Método**: Guardado()

9. Datos de Entrada

- **Descripción**: Contiene los datos que deben ingresarse o validarse para una operación sobre municipios:
  - idDepartamento: int → Identificador del departamento al que pertenece el municipio.
  - descripcion: String → Nombre o descripción del municipio.

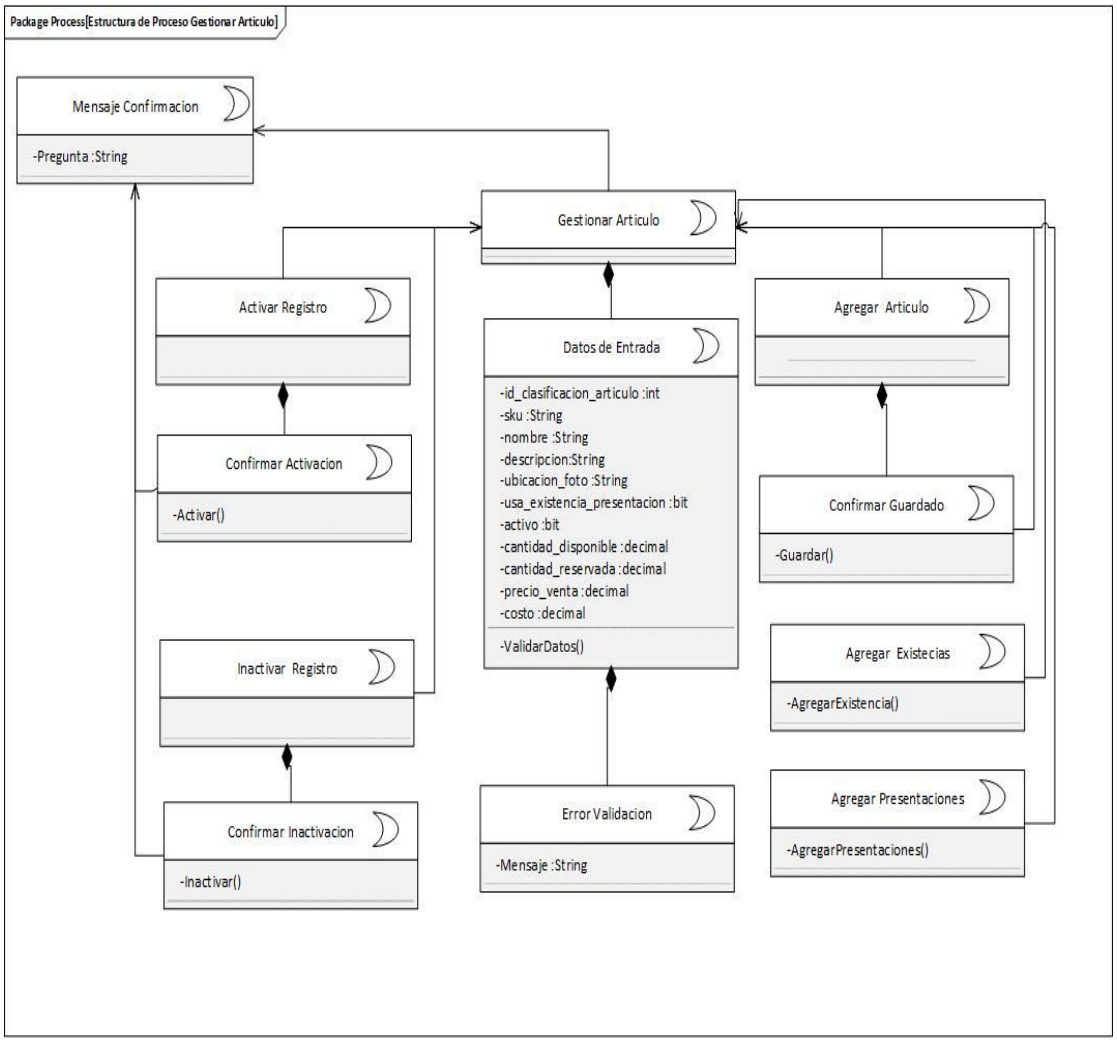
- estado: Bool → Indica si está activo o inactivo.
- validarDatos() → Función para validar la integridad y completitud de los datos.

10. Error Validación

- **Descripción**: Muestra un mensaje de error si los datos ingresados no cumplen con los criterios de validación.
- **Dato**: mensaje: String

**Diagrama Estructura Proceso Gestión Articulo**

***Figura 37.*** *Diagrama Estructura de Proceso Gestión Articulo*




1. Mensaje Confirmación

- **Descripción**: Se utiliza para mostrar una pregunta al usuario que requiere confirmación antes de ejecutar acciones críticas como activación, inactivación o eliminación.

- **Atributo**: pregunta: String

2. Gestionar Artículo

- **Descripción**: Proceso principal que engloba la gestión completa de un artículo, incluyendo creación, edición, activación, inactivación y validación de datos.

3. Activar Registro

- **Descripción**: Permite volver a habilitar un artículo que anteriormente fue inactivado.

4. Confirmar Activación

- **Descripción**: Solicita confirmación del usuario antes de activar un artículo.
- **Método**: Activar()

5. Inactivar Registro

- **Descripción**: Permite deshabilitar un artículo, dejándolo inactivo sin eliminarlo.

6. Confirmar Inactivación

- **Descripción**: Solicita confirmación antes de proceder con la inactivación del artículo.
- **Método**: Inactivar()

7. Datos de Entrada

- **Descripción**: Conjunto de datos requeridos para agregar o modificar un artículo en el sistema:
  - id_clasificacion_articulo: int
  - sku: String
  - nombre: String
  - descripcion: String
  - ubicacion_foto: String
  - usa_existencia_presentacion: bit

- activo: bit
- cantidad_disponible: decimal
- cantidad_reservada: decimal
- precio_venta: decimal
- costo: decimal
- validarDatos(): método para validar la integridad de la información.

8. Error Validación

- **Descripción**: Se activa si los datos ingresados no son válidos, mostrando un mensaje de error.
- **Atributo**: mensaje: String

9. Agregar Artículo

- **Descripción**: Permite crear un nuevo artículo en el sistema con la información proporcionada en "Datos de Entrada".

10. Confirmar Guardado

- **Descripción**: Informa que el artículo fue agregado exitosamente.
- **Método**: Guardado()

11. Agregar Existencias

- **Descripción**: Permite asignar o aumentar el stock de un artículo.
- **Método**: AgregarExistencia()

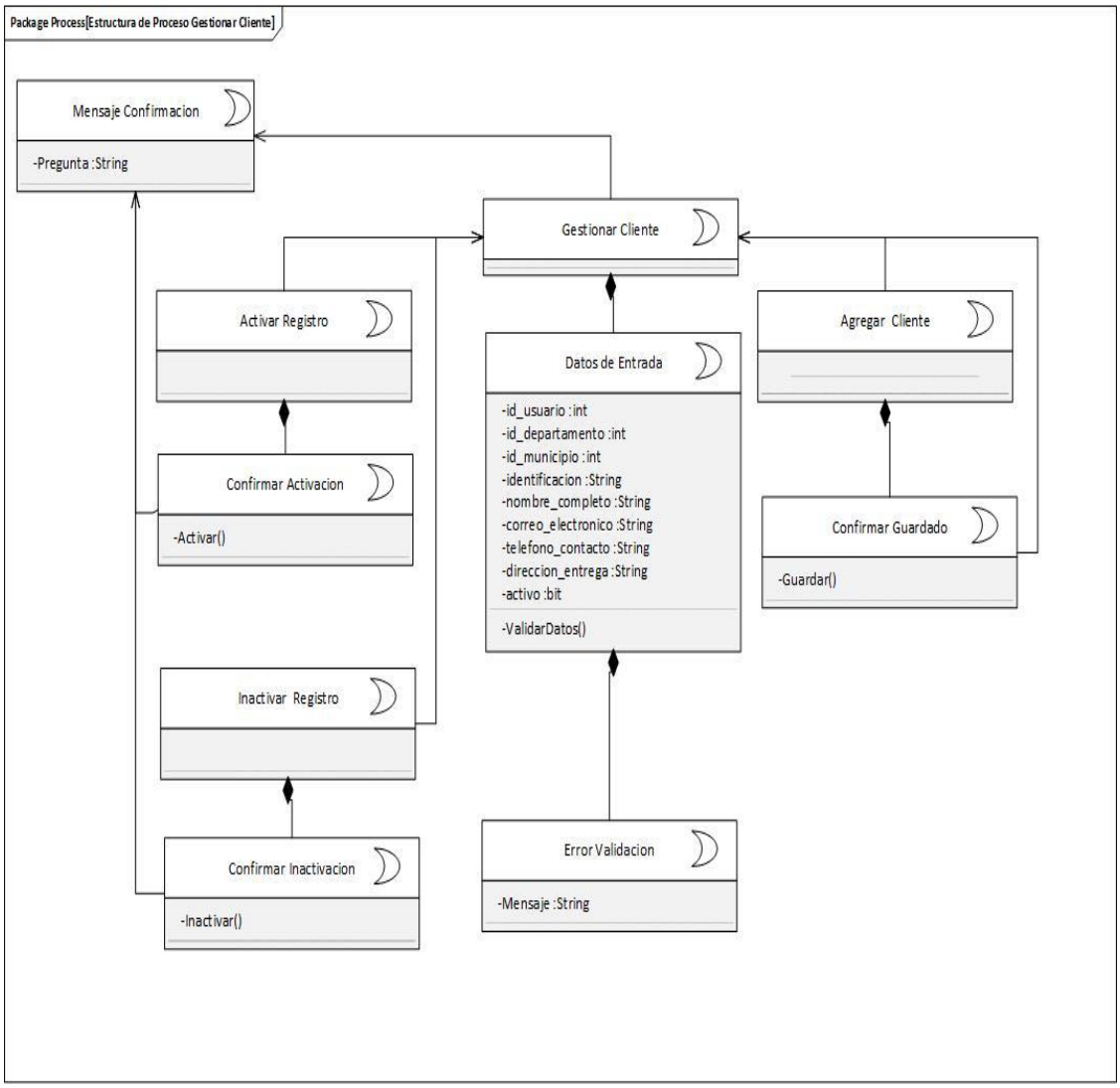
12. Agregar Presentaciones

- **Descripción**: Permite asignar diferentes presentaciones o formatos del mismo artículo (por ejemplo, unidad, caja, paquete).

- **Método**: AgregarPresentaciones()

**Diagrama Estructura de Proceso Gestión Cliente**

***Figura 38.*** *Diagrama Estructura de Proceso Gestión Cliente*




**1.Mensaje Confirmación**

- Contiene la propiedad: Pregunta : String

- Representa el mensaje que se le muestra al usuario para confirmar una acción, como activar o inactivar registros.

2. **Activar Registro**

- Método: Activar()

- Procesa la activación del registro de un cliente. Es decir, permite habilitarlo para su uso dentro del sistema.

3. **Confirmar Activación**

- Método: Inactivar() *(posible error, debería ser algo como Confirmar() o Activar() para mantener consistencia)*

- Realiza la validación final para asegurar que el cliente fue activado correctamente.

4. **Inactivar Registro**

- Método: Inactivar()

- Desactiva o deshabilita el registro de un cliente. Es útil cuando ya no se desea que el cliente esté activo en el sistema.

5. **Confirmar Inactivación**

- Método: Inactivar() *(al igual que en el anterior, el método puede estar redundante o mal etiquetado)*

- Verifica que la inactivación del cliente fue completada con éxito.

6. **Gestionar Cliente**

- Datos de Entrada:

  - id_usuario : int

- id_departamento : int
- nombre_cliente : String
- apellido_cliente : String
- telefono_cliente : String
- email_cliente : String
- direccion_empresa : String
- nombre_empresa : String

- Método: ValidarDatos()

- Clase central del sistema que gestiona la creación y modificación de clientes. Valida que los datos sean correctos antes de almacenarlos.

7. **Error Validación**

- Propiedad: Mensaje : String

- Si hay errores en los datos del cliente (como campos vacíos o mal formateados), este proceso se encarga de informar al usuario con un mensaje.

8. **Agregar Cliente**

- Método: Guardar()

- Permite almacenar los datos del nuevo cliente en la base de datos tras pasar por el proceso de validación.

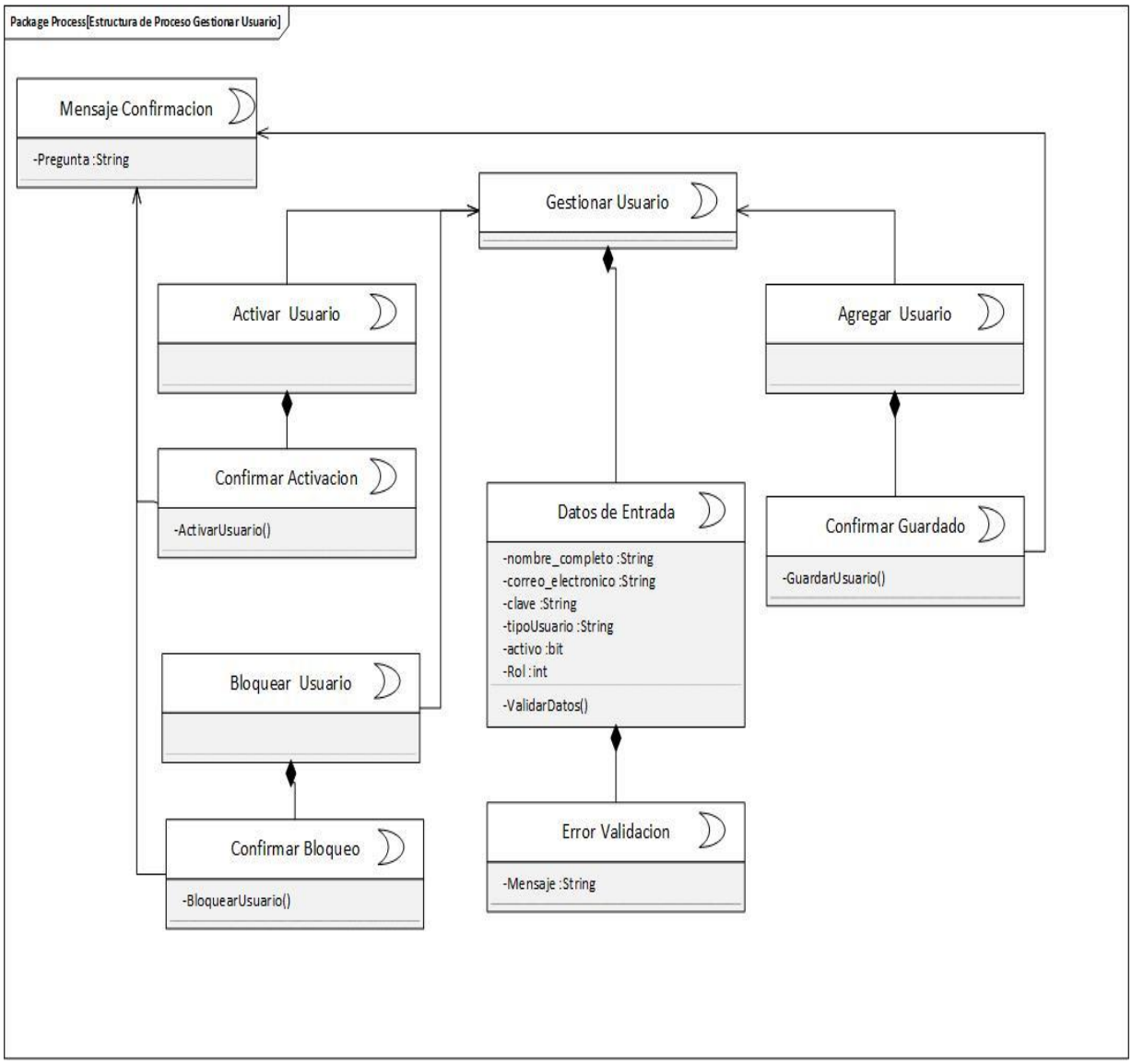
9. **Confirmar Guardado**

- Método: Guardar()

- Confirma que el cliente fue guardado exitosamente y que el proceso de inserción finalizó sin problemas.

## Diagrama Estructura de Proceso Gestión de Usuario

***Figura 39.*** *Diagrama Estructura de Proceso Gestión de Usuario*




## Mensaje Confirmación

- **Propiedad:** Pregunta : String

- Es una interfaz que solicita al usuario confirmar acciones sensibles, como activar o bloquear cuentas.

2. **Activar Usuario**

- **Método:** ActivarUsuario()

- Habilita a un usuario que estaba inactivo para que pueda utilizar el sistema. Se actualiza su estado a “activo”.

3. **Confirmar Activación**

- **Método:** ActivarUsuario()

- Confirma que el usuario fue activado correctamente. Podría representar una retroalimentación visual o lógica del sistema.

4. **Bloquear Usuario**

- **Método:** BloquearUsuario()

- Impide el acceso del usuario al sistema. Se usa cuando el usuario representa un riesgo o está bajo revisión.

5. **Confirmar Bloqueo**

- **Método:** BloquearUsuario()

- Verifica que el bloqueo se aplicó exitosamente. Ayuda a confirmar que los cambios se reflejaron como esperados.

6. **Gestionar Usuario**

- Es el proceso central que coordina todas las operaciones de usuario. Aquí se validan datos, se agregan registros y se ejecutan activaciones o bloqueos.

7. **Datos de Entrada**

- **Atributos del usuario:**
  - nombre_completo : String
  - usuario : String
  - clave : String
  - correo : String
  - telefono : String
- **Método:** ValidarDatos()
- Se encargan de asegurar que la información ingresada por el usuario sea válida y completa.

8. **Error Validación**

- **Propiedad:** Mensaje : String
- Se dispara cuando algún dato es incorrecto o falta. Muestra un mensaje descriptivo para corregir la entrada.

9. **Agregar Usuario**

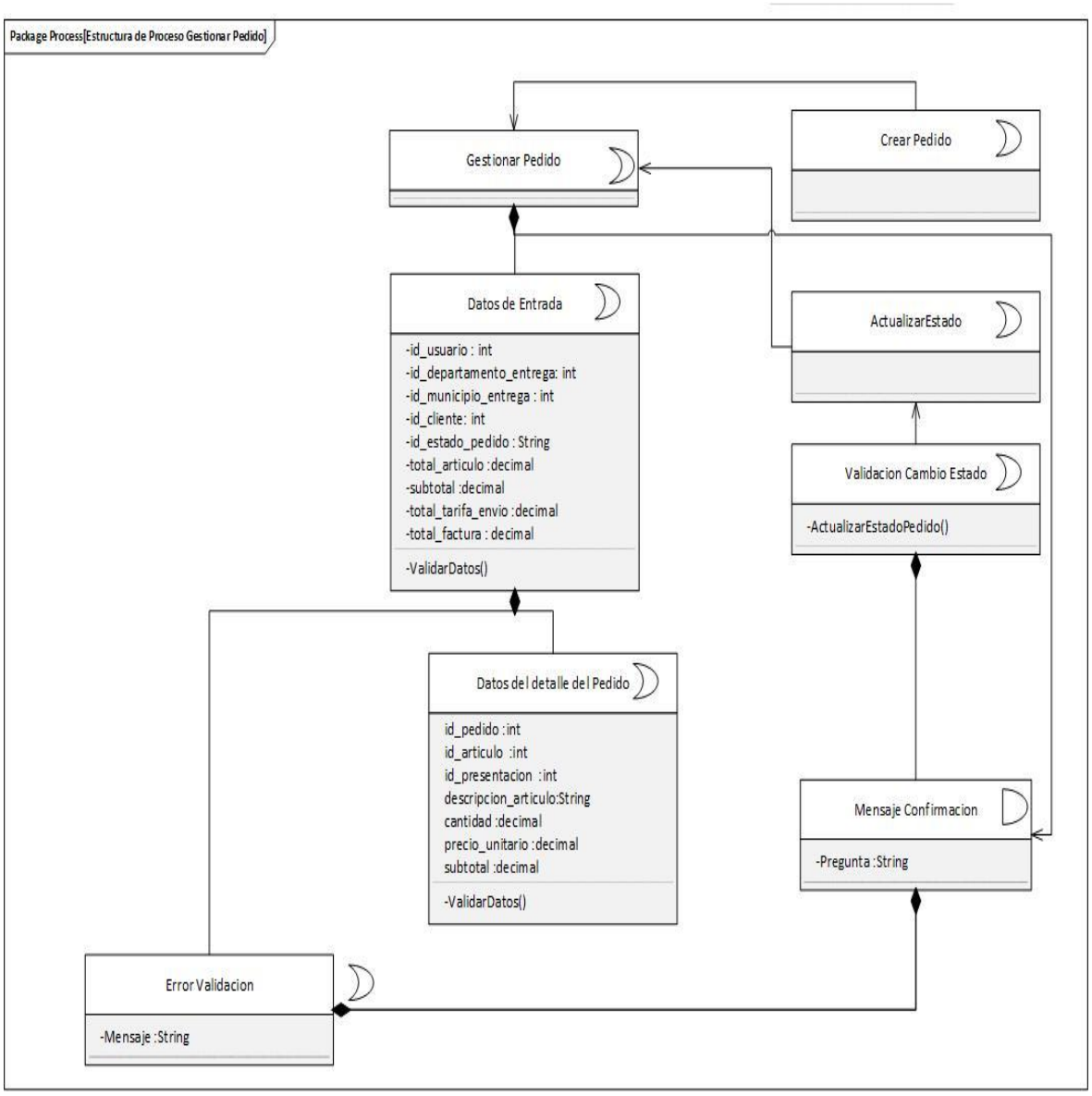
- **Método:** GuardarUsuario()
- Inserta un nuevo registro de usuario en el sistema tras validar los datos correctamente.

10. **Confirmar Guardado**

- **Método:** GuardarUsuario()
- Verifica que el nuevo usuario fue registrado sin errores. Confirma la persistencia del dato en la base de datos.

## Diagrama Estructura de Proceso Gestionar Pedido

***Figura 40.*** *Diagrama Estructura de Proceso Gestión Pedido*




## Gestionar Pedido

- Es el proceso central que coordina la lógica de negocio para crear, validar y actualizar pedidos.

- Actúa como controlador general y puede invocar otros procesos como creación o actualización.

2. **Crear Pedido**

- Se encarga de iniciar la generación de un nuevo pedido en el sistema.
- Invoca la validación de los datos antes de almacenarlos.

3. **Datos de Entrada**

- Contiene la estructura de atributos que debe tener un pedido:
  - id_usuario : int
  - id_pedido : int
  - fecha_pedido : date
  - cantidad_total : int
  - precio_total : decimal
  - estado_pedido : string
  - fecha_entrega : date
- Método: ValidarDatos()
- Realiza validaciones para garantizar que los datos del pedido cumplen con los criterios establecidos.

4. **Actualizar Estado**

- Permite modificar el estado de un pedido (por ejemplo, de “pendiente” a “procesado”).

- Se suele ejecutar como parte de la lógica del proceso o de una acción administrativa.

5. **Validación Cambio Estado**

- Verifica que el nuevo estado del pedido sea válido.
- Previene transiciones incorrectas o que no estén permitidas por las reglas del negocio.

6. **Datos del Detalle del Pedido**

- Incluye la información específica de cada ítem del pedido:
  - id_pedido : int
  - id_producto : int
  - cantidad : int
  - precio_unitario : decimal
  - precio_total : decimal
- Método: ValidarDetalle()
- Revisa que los valores por producto sean correctos y coherentes con el pedido general.

7. **Mensaje Confirmación**

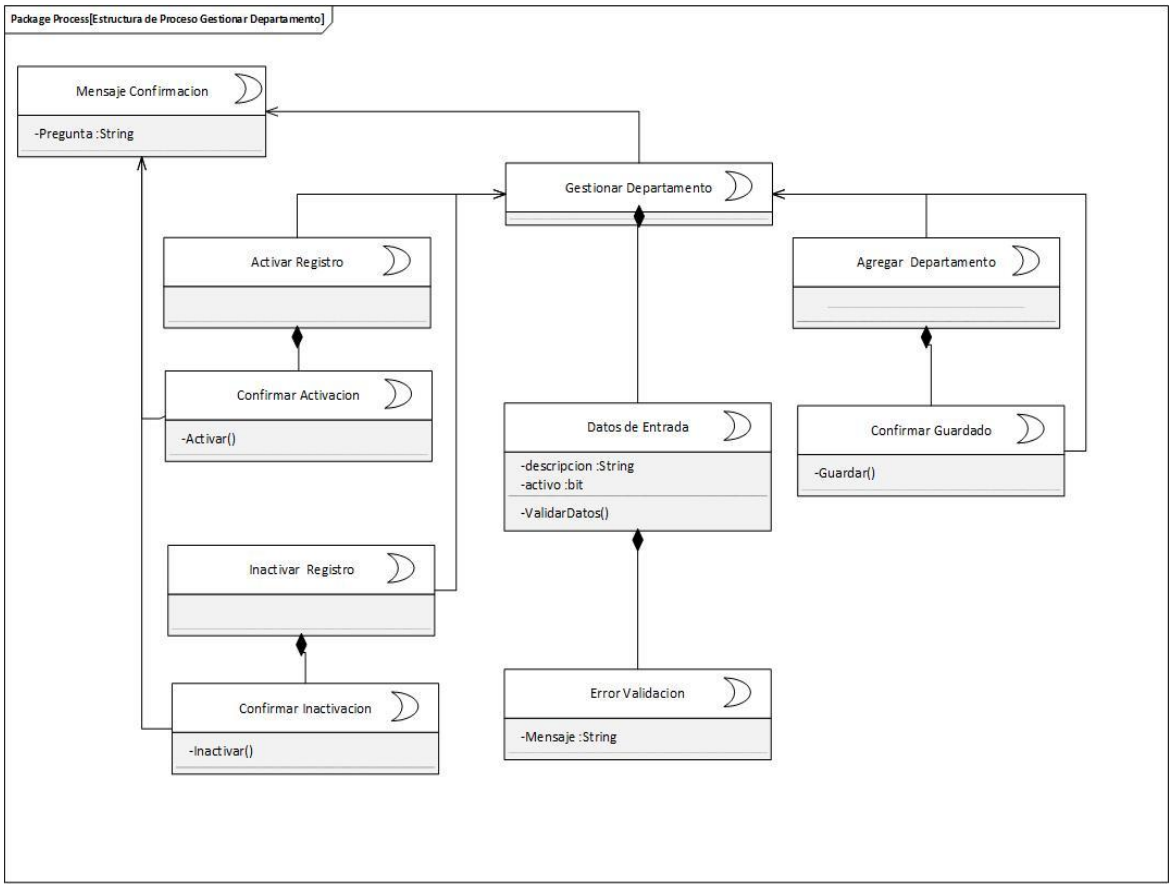
- Propiedad: Pregunta : String
- Presenta al usuario una pregunta antes de ejecutar una acción significativa (como actualizar el estado del pedido).

8. **Error Validación**

- Propiedad: Mensaje : String
- Se activa cuando algún dato del pedido o sus detalles no pasa la validación.
- Muestra una alerta para corregir errores antes de continuar.

**Diagrama Estructura de Proceso Gestión Departamento**

***Figura 41**. Diagrama Estructura de Proceso Gestión Departamento*




**Mensaje Confirmación**

- **Propiedad:** Pregunta : String
- Muestra un mensaje de confirmación al usuario antes de ejecutar acciones importantes (como activar o inactivar un departamento).

2. **Activar Registro**

- **Método:** Confirmar Activación()
- **Propiedad:** Activado()
- Cambia el estado del departamento a “activo”. Se utiliza cuando un departamento previamente inactivo debe volver a operar.

3. **Inactivar Registro**

- **Método:** Confirmar Inactivación()
- **Propiedad:** Inactivado()
- Deshabilita el departamento, impidiendo que se utilice en operaciones futuras hasta que se reactive.

4. **Gestionar Departamento**

- **Métodos:**
  - AgregarDepartamento()
  - ConfirmarGuardado()
- Es el proceso principal que controla el flujo completo de creación y actualización de departamentos. Interactúa con las validaciones y el almacenamiento.

5. **Datos de Entrada**

- **Atributos:**
  - descripcion : String
  - id : String
- **Método:** ValidarDatos()

- Asegura que los datos ingresados para crear o actualizar un departamento sean válidos (por ejemplo, que la descripción no esté vacía y el ID sea único).

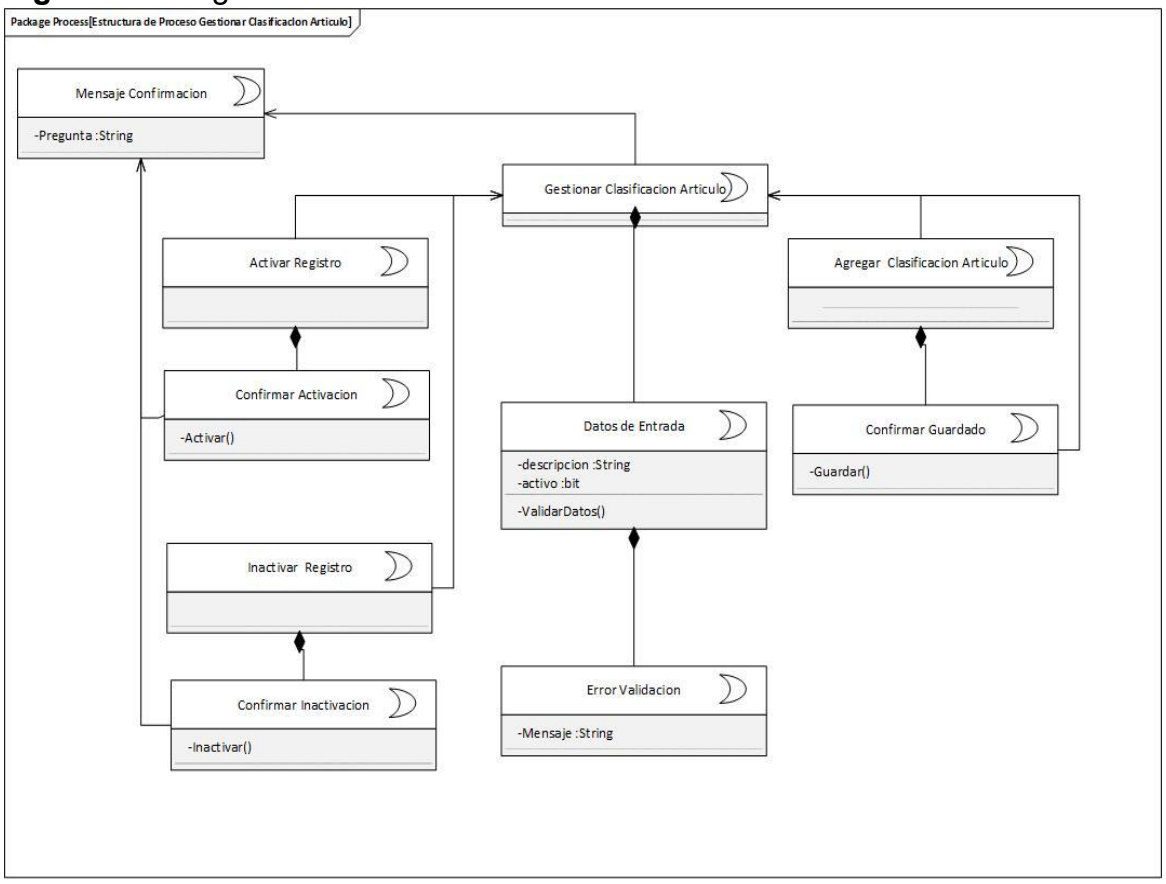
6. **Error Validación**

- **Propiedad:** Mensaje : String

- Se activa si hay errores en los datos de entrada, mostrando un mensaje informativo al usuario para corregirlos.

**Diagrama Estructura de Proceso Gestión Clasificación Articulo**

**Figura 42.** Diagrama Estructura de Proceso Gestión Clasificación Articulo




**Gestión de Clasificación de Artículos**

- Es el proceso principal que administra la lógica de negocio relacionada con la clasificación de artículos.

- Coordina tareas como activación, inactivación, validación y almacenamiento.

**2. Agregar Clasificación Artículos**

- **Método:** Guardar()

- Se encarga de registrar una nueva clasificación en el sistema tras validar la información.

**3. Confirmar Guardado**

- **Método:** Guardado()

- Verifica que la clasificación fue guardada exitosamente en la base de datos.

**4. Activar Registro**

- **Método:** Activar()

- Habilita una clasificación previamente inactiva para que sea visible y utilizable en el sistema.

**5. Confirmar Activación**

- **Método:** Activar()

- Confirma que la clasificación fue activada correctamente y que el estado cambió con éxito.

**6. Inactivar Registro**

- **Método:** Inactivar()

- Deshabilita la clasificación de artículos para que no se use ni se muestre en el sistema activo.

**7. Confirmar Inactivación**

- **Método:** Inactivar()
- Verifica que el estado del registro fue cambiado a “inactivo” correctamente.

**8. Datos de Entrada (Propiedades del Registro)**

Contiene los campos que forman una clasificación:

- descripcion : String
- activo : Bit
- **Método:** ValidarDatos()
- Revisa que los datos cumplan las reglas de formato y presencia antes de procesarlos.

**9. Mensaje Confirmación**

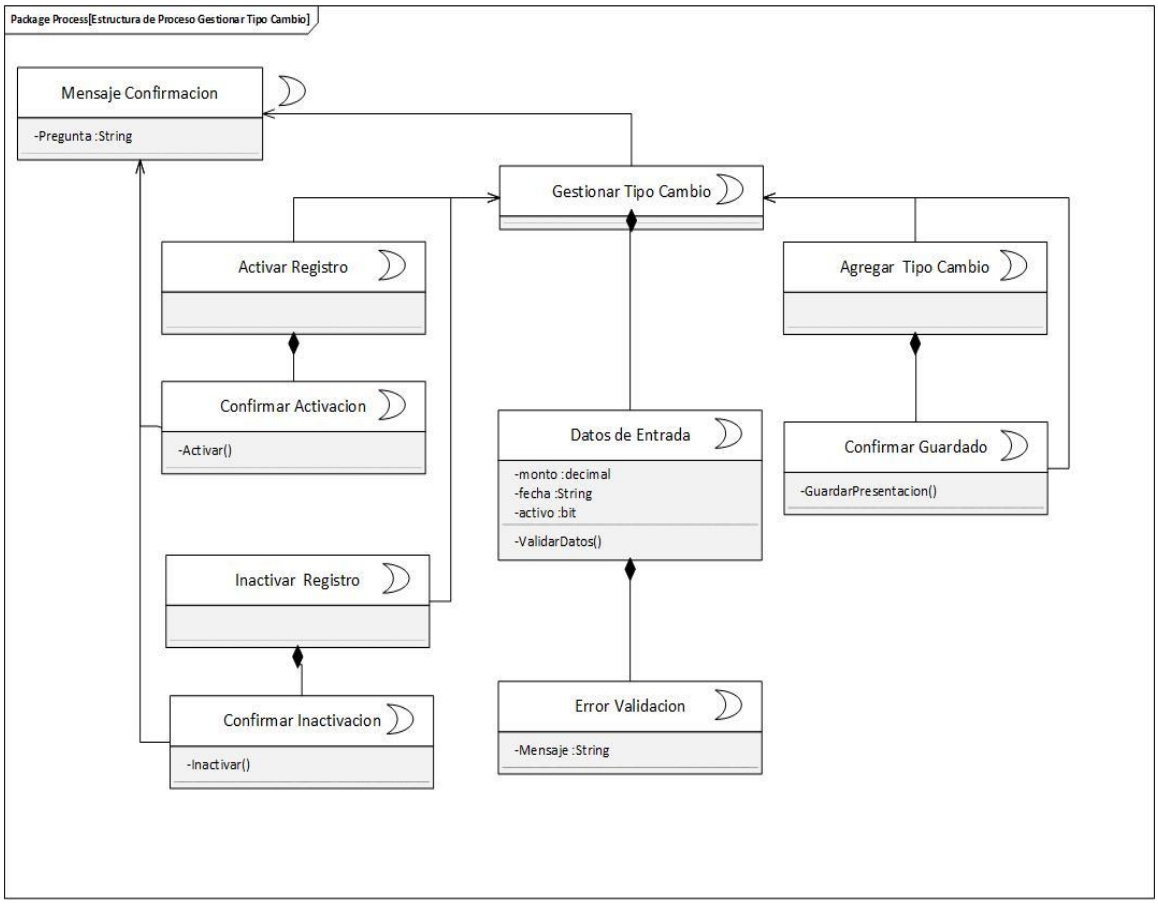
- **Propiedad:** Pregunta : String
- Solicita la aprobación del usuario antes de realizar una acción importante (como activar/inactivar un registro).

**10. Error Validación**

- **Propiedad:** Mensaje : String
- Muestra los errores detectados si los datos de entrada no son válidos.
- Detiene el flujo y solicita corrección antes de continuar.

**Diagrama de Estructura de Proceso Gestión Tipo de Cambio**

***Figura 43.*** *Diagrama Estructura de Proceso Gestión Tipo de Cambio*




**Fuente:** Elaboración Propia

**Gestionar Tipo de Cambio**

- Es el **proceso principal** que organiza y controla todas las operaciones relacionadas con los tipos de cambio.
- Recibe datos, valida entradas y orquesta el flujo hacia activación, inactivación o almacenamiento.

**2. Agregar Tipo de Cambio**

- Guarda un nuevo tipo de cambio en el sistema.
- Se ejecuta después de validar que los datos ingresados son correctos.

**3. Confirmar Guardado**

- **Método:** GuardadoCorrecto()
- Verifica que el registro fue almacenado exitosamente en la base de datos.

**4. Activar Registro**

- **Método:** Activar()
- Habilita un tipo de cambio previamente inactivo para que pueda usarse en cálculos o reportes.

**5. Confirmar Activación**

- **Método:** Activado()
- Confirma que el estado cambió a "activo" de forma correcta y consistente.

**6. Inactivar Registro**

- **Método:** Inactivar()
- Desactiva un tipo de cambio, impidiendo que se use sin eliminarlo del sistema.

**7. Confirmar Inactivación**

- **Método:** Inactivado()
- Verifica que el tipo de cambio fue marcado como "inactivo" correctamente.

**8. Datos de Entrada**

- Estructura base del registro:
  - entrada : decimal
  - fecha_entrada : date

- hora_entrada : time

- **Método:** Validación()

- Evalúa que los datos estén completos y cumplan con el formato correcto.

**9. Mensaje Confirmación**

- **Propiedad:** Pregunta : String

- Solicita al usuario confirmar una acción antes de aplicarla (por ejemplo, activar o inactivar).

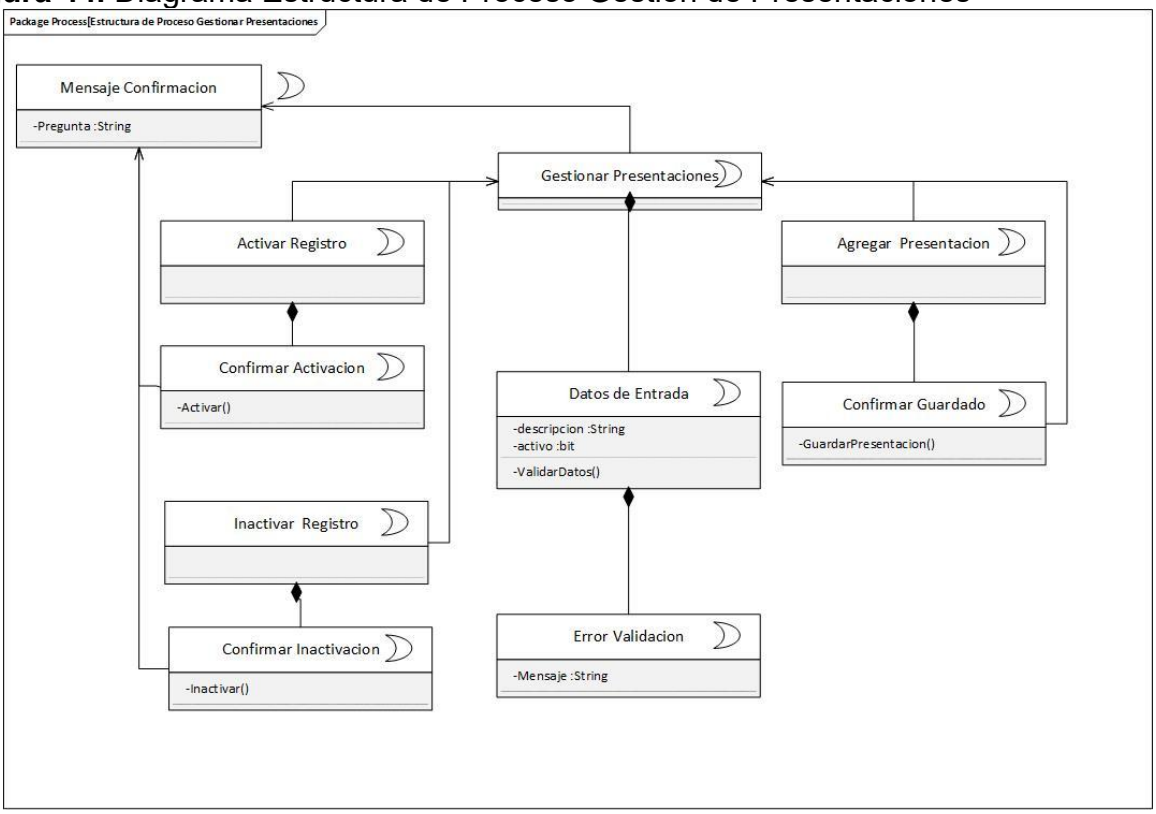
**10. Error Validación**

- **Propiedad:** Mensaje : String

- Se muestra cuando hay errores en los datos. Evita guardar información incorrecta y guía al usuario a corregir.

## Diagrama de Estructura de Proceso Gestión Presentaciones Articulo

**Figura 44.** Diagrama Estructura de Proceso Gestión de Presentaciones




Gestionar Presentaciones

- Es el proceso central que coordina toda la lógica del módulo.
- Se conecta con los procesos de activación, inactivación, creación y validación.
- Orquesta las acciones que se realizan sobre las presentaciones, como alta, baja o modificación.

### 2. Agregar Presentación

- **Método:** guardarPresentacion()
- Registra una nueva presentación en la base de datos.

- Se ejecuta después de validar los datos para asegurar que la entrada sea válida.

**3. Confirmar Guardado**

- **Método:** guardarPresentacion()
- Confirma que la operación de guardado fue exitosa.
- Podría implicar una respuesta visual al usuario o un log interno del sistema.

**4. Activar Registro**

- **Método:** activar()
- Permite que una presentación que estaba inactiva se vuelva funcional y visible en el sistema.

**5. Confirmar Activación**

- **Método:** activar()
- Verifica que el estado del registro fue actualizado correctamente a “activo”.

**6. Inactivar Registro**

- **Método:** inactivar()
- Desactiva una presentación sin eliminarla del sistema, impidiendo su uso o visualización.

**7. Confirmar Inactivación**

- **Método:** inactivar()
- Comprueba que el estado fue cambiado correctamente a “inactivo”.

**8. Datos de Entrada**

- Propiedad esencial:
  - descripcion : String
- **Método:** validarDatos()
- Verifica que el texto de descripción esté presente y tenga el formato adecuado antes de ejecutar el registro.

**9. Error Validación**

- Propiedad:
  - mensaje : String
- Se activa cuando hay errores en los datos ingresados.
- Muestra un mensaje explicativo al usuario para corregir la entrada.

**10. Mensaje Confirmación**

- Propiedad:
  - pregunta : String
- Aparece antes de realizar una acción crítica como activar o inactivar.
- Ayuda a prevenir errores por acciones no intencionada

## IX. Capitulo III: Implementación del Sistema

En este capítulo, se presentan los aspectos técnicos relacionados con la implementación del sistema, abarcando tanto el hardware como el software utilizado en su desarrollo. Se detallan las herramientas, lenguajes de programación, frameworks y entornos de desarrollo empleados, así como la estructura del código y los principios de diseño que guiaron su construcción. También se describen los procesos clave en la codificación, incluyendo la integración de módulos, la gestión de bases de datos, la

implementación de interfaces y la optimización del rendimiento. Posteriormente se analizan los desafíos enfrentados durante la fase de desarrollo y las soluciones adoptadas para garantizar un sistema eficiente y funcional.

## 9.1. **Aspectos Generales de Implementación**

A continuación, se detallan los componentes del sistema que requieren una infraestructura específica para su correcta implementación en un entorno de hospedaje:

**Aplicación web administrativa**: Esta interfaz frontend está desarrollada utilizando Next.js, por lo que se requiere un servidor que cuente con Node.js en una versión igual o superior a la 18.12.1, garantizando compatibilidad con las características modernas del framework.

**API del sistema**: El backend ha sido construido utilizando Node.js con Express, lo que implica la necesidad de un entorno que permita la ejecución de servicios basados en JavaScript del lado del servidor. Se recomienda contar con Node.js en su versión LTS más reciente y soporte para manejadores de procesos como PM2 o servicios equivalentes.

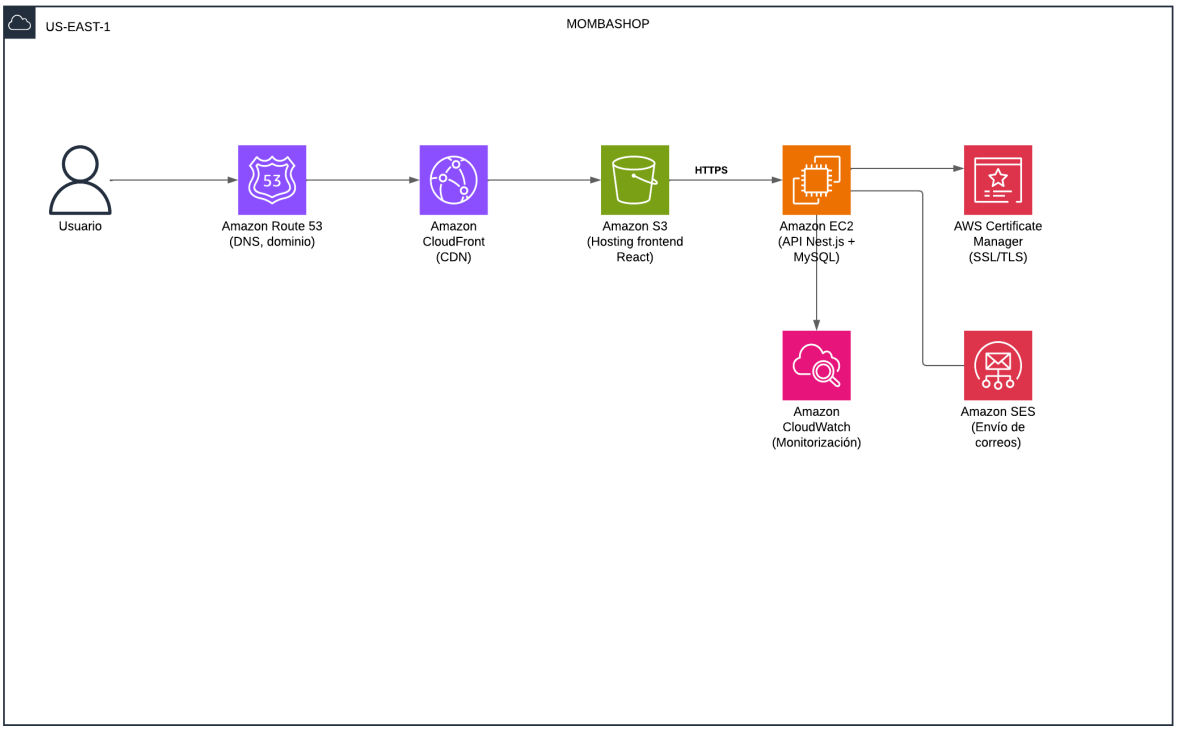
**Base de datos**: El sistema de almacenamiento de datos se fundamenta en MySQL. Se requiere un servidor de base de datos que cuente con MySQL en una versión igual o superior a la 5.7, lo cual asegura compatibilidad con las operaciones relacionales definidas en el modelo de datos.

## 9.2. **Diagrama de Arquitectura**

La arquitectura del sistema Mombashop está diseñada sobre la nube de AWS en la región US-East-1, empleando una estructura orientada a servicios y componentes desacoplados. Los usuarios acceden al sistema mediante el dominio administrado por Amazon Route 53, el cual dirige las solicitudes a través de Amazon CloudFront como CDN para optimizar la entrega del frontend. La interfaz de usuario está alojada en Amazon S3, mientras que la API de negocio, implementada con Nest.js y conectada a

MySQL, se ejecuta en instancias de Amazon EC2. La comunicación entre el frontend y la API se realiza mediante HTTPS, asegurada con certificados gestionados por AWS Certificate Manager. El sistema integra servicios adicionales como Amazon SES para envío de correos electrónicos y Amazon CloudWatch para monitoreo de la infraestructura y las aplicaciones, garantizando operación confiable, escalabilidad y seguimiento continuo del desempeño.

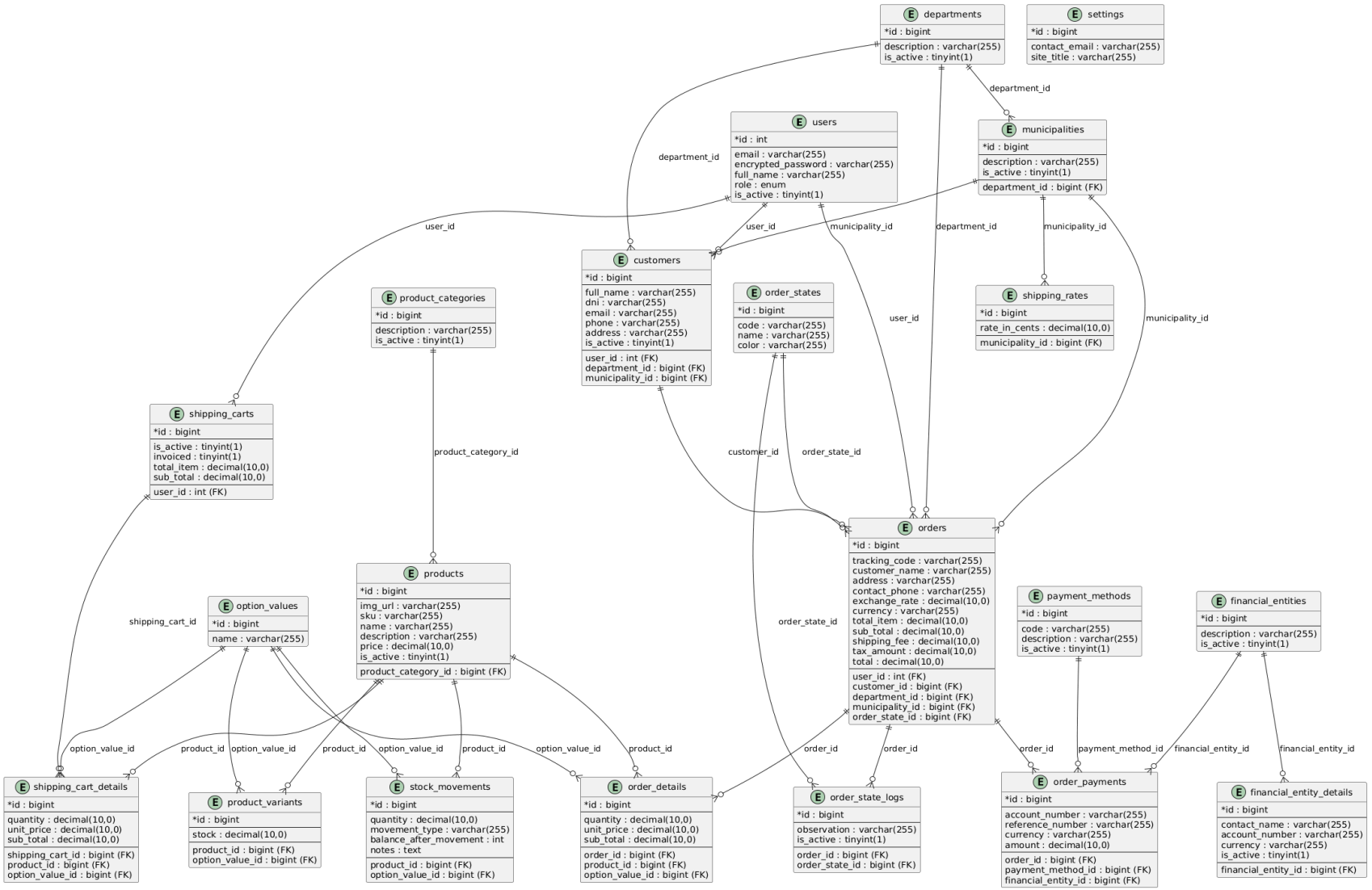
***Figura 45**. Diagrama de Arquitectura*




## 9.3. Modelo de Datos

El modelo de datos que se presenta a continuación describe la estructura lógica de la información que será gestionada por el sistema. Este modelo define las entidades principales, sus atributos y las relaciones existentes entre ellas, permitiendo una organización coherente y eficiente de los datos. Su elaboración responde a la necesidad de garantizar la integridad, consistencia y escalabilidad de la base de datos, sirviendo como base para el diseño físico y la implementación del almacenamiento de la información en el sistema.

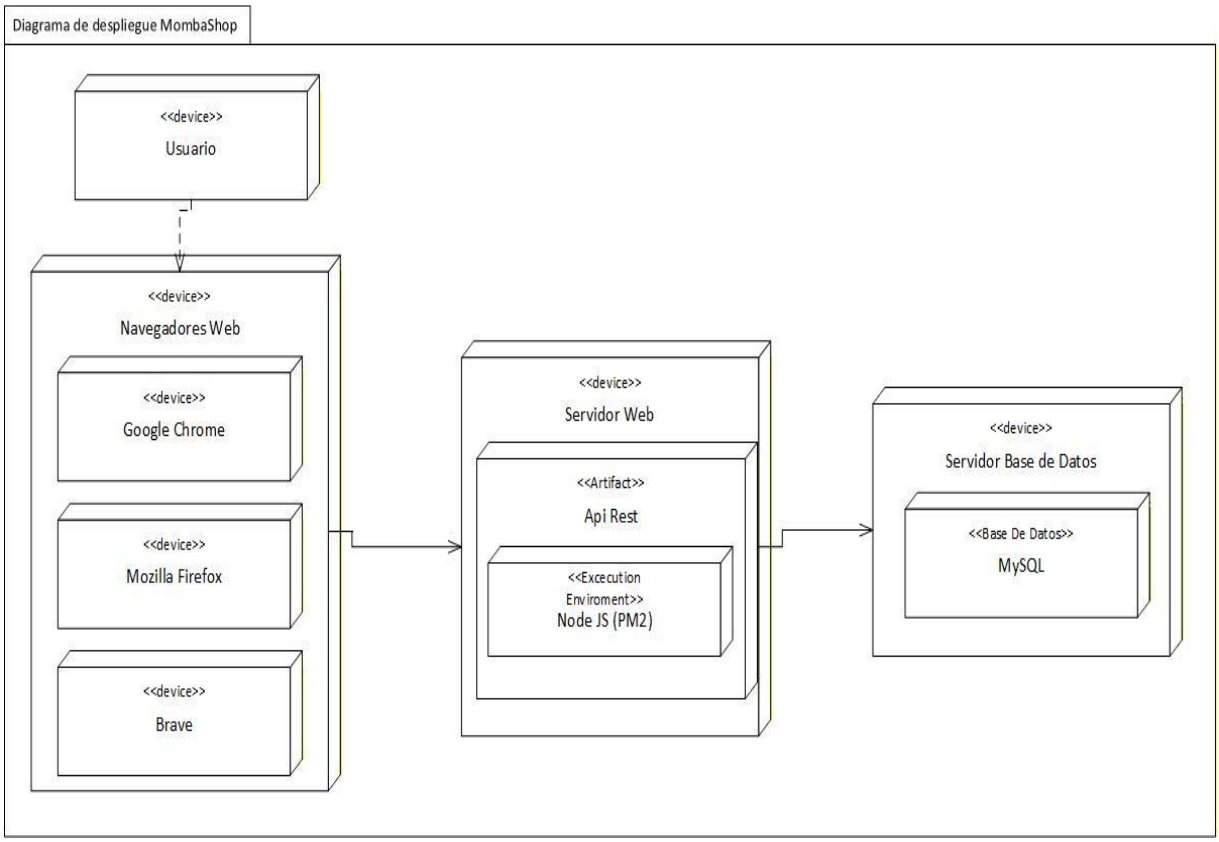
***Figura 46**. Diagrama Entidad Relación*



## 9.4. Diagrama de Despliegue

A continuación, se presenta el diagrama de despliegue del sistema, el cual permite visualizar la arquitectura física sobre la que se ejecutará la aplicación. Su propósito es proporcionar una representación clara de cómo se distribuyen los elementos del sistema en el entorno de ejecución real.

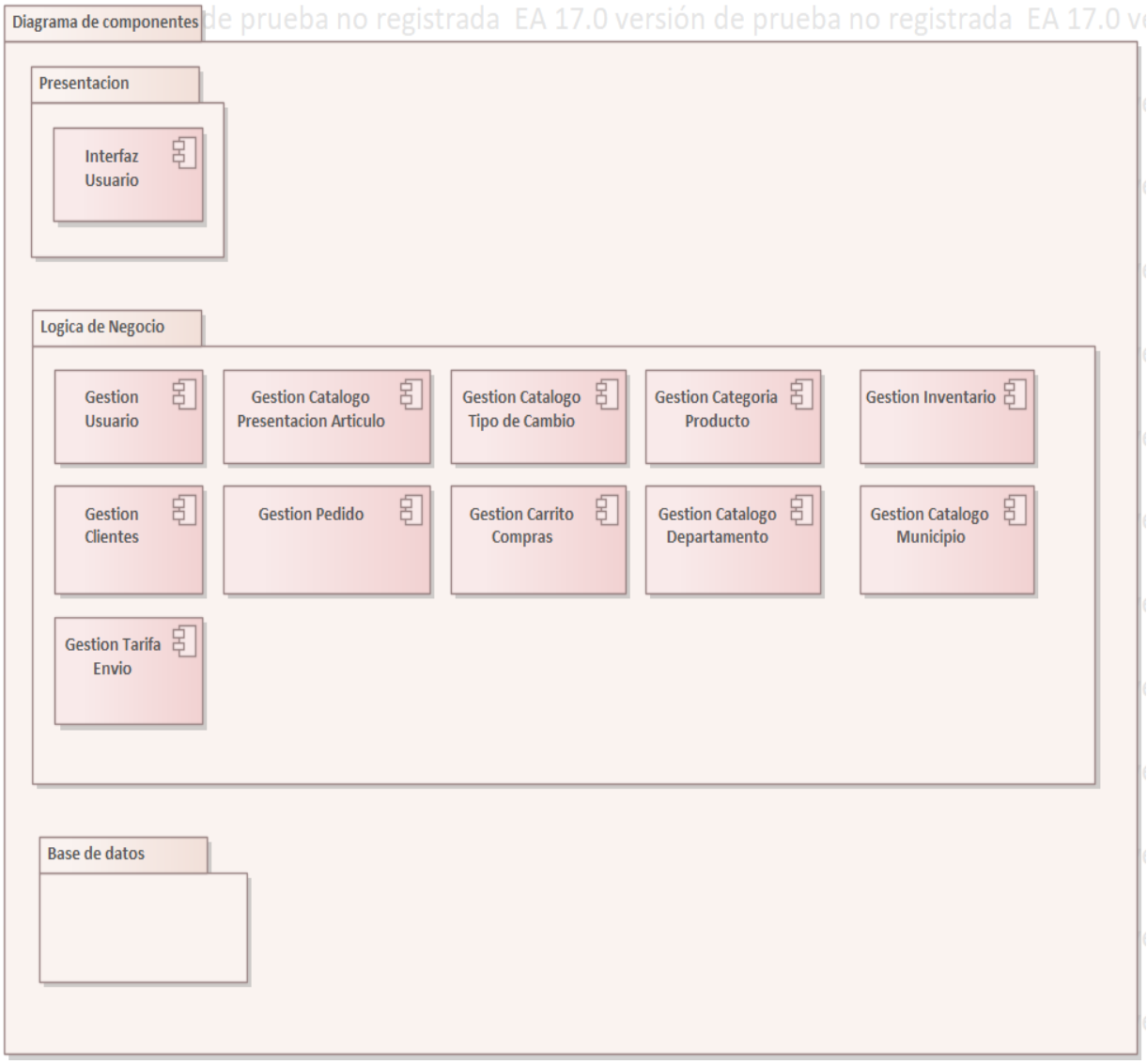
***Figura 47*** *Diagrama de Despliegue*



## 9.5. Diagrama de Paquetes

El diagrama de paquetes permite representar de forma estructurada los componentes principales del sistema, agrupando clases, módulos o funcionalidades relacionadas en conjuntos lógicos. Esta representación facilita la comprensión de la arquitectura general del software, destacando las dependencias entre los distintos paquetes y promoviendo una organización modular que favorece el mantenimiento, la reutilización y la escalabilidad del sistema. A través de este diagrama, es posible visualizar cómo se distribuyen las responsabilidades funcionales dentro del proyecto, así como identificar posibles acoplamientos entre módulos.

***Figura 54*** *Diagrama de Paquetes*



## 9.6. **Selección del Hardware para la construcción del sistema Ecommerce.**

**Tabla 18** Hardware Utilizado para la construcción del Ecommerce

| **PC #1** | **PC #2** | **PC #3** |
|---|---|---|
| Intel Core i5 12va G | Intel Core i5 10ma G | Intel Core i7 |
| 16 GB de Ram | 16 GB de Ram | 12 GB de Ram |
| 1T SSD | 500 SSD | 1T SSD |

Fuente: Elaboración Propia

## 9.7. **Selección del Software seleccionado para la construcción del sistema Ecommerce.**

A continuación, se detalla el software y las tecnologías de desarrollo web que se utilizaron durante la codificación del sistema:

- Visual Studio Code
- Gestor de Base de Datos Relacional MySQL
- Navegador Google Chrome
- Postman
- Node JS
- Javascript
- NestJS

- JWT

## 9.8. Seguridad Back End

En esta sección, se detallan las prácticas de seguridad implementadas en el backend para proteger tanto la aplicación como los datos de los usuarios. A continuación, se describen algunas de las principales medidas adoptadas:

### 9.8.1. Implementacion JSON Web Token (JWT)

En el contexto de la seguridad, uno de los componentes clave implementados en el proyecto es la utilización de JSON Web Tokens (JWT) para la autenticación y autorización de los usuarios. JWT es un estándar abierto que define un formato compacto y seguro para transmitir información entre las partes como un objeto JSON, que se puede verificar y confiar en su autenticidad mediante una firma digital.

La configuración de seguridad mediante JWT permite implementar un sistema de autenticación basado en tokens que proporciona una mayor flexibilidad y escalabilidad en comparación con otros métodos tradicionales como las sesiones de usuario

### 9.8.2. Rate Limiting

Se implementó la librería rateLimiter para evitar ataques de denegación de servicio (DoS) y proteger la aplicación contra el abuso de las rutas. Esta librería limita el número de solicitudes que un cliente puede hacer en un período determinado. En nuestro caso, se configuró un límite de 60 solicitudes por cada 15 minutos para cada usuario, lo que ayuda a prevenir un uso excesivo y no autorizado de los recursos de la API.

### 9.8.3. Protección XSS (Cross–Site Scripting)

Para proteger la aplicación contra ataques de XSS (Cross-site Scripting), se implementó la librería xss. Esta ayuda a sanitizar los datos que se reciben, eliminando cualquier intento de inyección de scripts maliciosos.

### 9.8.4. **Prevención clickjacking, CSFR mediante Helmet**

Helmet es una colección de middleware que ayuda a proteger la aplicación Express configurando varios encabezados HTTP de seguridad. Entre otras cosas, Helmet previene ataques como clickjacking, XSS y falsificación de solicitudes entre sitios (CSRF), mejorando así la seguridad de la aplicación.

### 9.8.5. **Cors**

CORS es un middleware que gestiona las políticas de acceso entre orígenes. En nuestro proyecto, se configuró para permitir solicitudes solo desde dominios de confianza, evitando que sitios no autorizados puedan interactuar con la API. Esto ayuda a prevenir ataques de Cross-Site Request Forgery (CSRF) y garantiza que solo los clientes autorizados puedan acceder a los recursos del backend.

# X. CONCLUSIONES

El presente estudio tuvo como propósito principal el desarrollo de un sistema E-commerce para la tienda MombaShop, con el objetivo de optimizar su gestión comercial mediante una solución tecnológica que facilite la venta en línea, mejore la experiencia del usuario y garantice el almacenamiento seguro de la información.

En cumplimiento del primer objetivo específico, se llevó a cabo un análisis integral de la viabilidad técnica, operativa, económica y legal del proyecto. Dicho análisis permitió identificar que el desarrollo del sistema es factible en todos estos ámbitos. Desde el punto de vista técnico, la elección de tecnologías basadas en JavaScript para el desarrollo del frontend y backend, junto con MySQL como motor de base de datos, ofreció una arquitectura robusta, escalable y de fácil mantenimiento. Operativamente, el sistema propone mejoras sustanciales en la gestión de productos, pedidos y clientes, permitiendo una automatización eficiente de los procesos comerciales. Económicamente, el análisis de costos evidenció que la inversión en el desarrollo del sistema resulta viable y rentable, considerando los beneficios proyectados. Legalmente, se contemplaron los marcos normativos aplicables al comercio electrónico y la protección de datos, garantizando el cumplimiento de las disposiciones vigentes.

En cuanto al segundo objetivo, el diseño del sistema fue desarrollado utilizando la metodología UWE (UML-based Web Engineering), lo que permitió modelar de manera estructurada y coherente los aspectos conceptuales, navegacionales y de interacción del sistema. Esta metodología favoreció la alineación entre los requerimientos del negocio y la solución propuesta, asegurando una experiencia de usuario intuitiva y orientada a los objetivos comerciales de la tienda.

Finalmente, respecto al tercer objetivo, el sistema fue codificado implementando tecnologías modernas basadas en JavaScript, como Node.js para el backend y React para la interfaz del usuario, garantizando una arquitectura modular, mantenible y con capacidad de integración futura. La base de datos MySQL permitió organizar de forma eficiente la información relevante, brindando soporte sólido a la operación del sistema.

En síntesis, el desarrollo del sistema E-commerce para MombaShop representa una solución tecnológica viable, alineada con las necesidades del negocio y orientada a mejorar significativamente su presencia digital y operatividad comercial. La implementación de este sistema permitirá a la tienda fortalecer su competitividad en el mercado, optimizar procesos internos y ofrecer una experiencia de compra más satisfactoria para sus clientes.

## XI. RECOMENDACIONES

**Capacitación al personal**: Se recomienda brindar formación básica a los colaboradores encargados del uso del sistema, tanto en la gestión de productos como en la atención al cliente, con el fin de asegurar un uso adecuado y eficiente de la plataforma.

**Monitoreo y mantenimiento periódico**: Es importante implementar rutinas de mantenimiento preventivo y correctivo del sistema para garantizar su buen funcionamiento y prevenir posibles fallos que afecten la operatividad.

**Actualización tecnológica continua**: Se sugiere mantener actualizadas las tecnologías utilizadas (frameworks, dependencias y motor de base de datos) para asegurar la seguridad, compatibilidad y rendimiento del sistema a largo plazo.

**Mejora continua basada en retroalimentación**: Se recomienda establecer canales para recibir retroalimentación de los usuarios y clientes, lo que permitirá identificar oportunidades de mejora y adaptar la plataforma a las necesidades cambiantes del negocio.

**Implementación de estrategias de marketing digital**: Para maximizar los beneficios del sistema E-commerce, se aconseja desarrollar campañas de posicionamiento en motores de búsqueda (SEO), publicidad en redes sociales y promociones en línea que atraigan tráfico hacia la tienda virtual.

**Integración con pasarelas de pago seguras**: Se recomienda integrar servicios de pago confiables y con cumplimiento normativo para facilitar las transacciones y garantizar la seguridad de la información financiera de los usuarios.

**Respaldo frecuente de la base de datos**: Es fundamental establecer políticas de respaldo automático y frecuente de la base de datos para prevenir pérdidas de información ante posibles fallos técnicos o ciberataques.

**Cumplimiento de normativas legales vigentes**: Se sugiere mantener una revisión periódica del marco legal relacionado con comercio electrónico y protección de datos personales, asegurando que el sistema cumpla con las normativas actuales y futuras.

**Escalabilidad del sistema**: Considerar la futura expansión del negocio y asegurar que el sistema esté preparado para escalar en términos de volumen de usuarios, productos y operaciones.

**Análisis de indicadores de desempeño**: Se recomienda incorporar métricas de rendimiento del sistema y de comportamiento del usuario, con el fin de tomar decisiones informadas y estratégicas para el crecimiento de la tienda.

# XII. BIBLIOGRAFIA


- Amazon Web Services. (s. f.). *AWS Pricing Calculator*. Recuperado de https://calculator.aws/
- Asamblea Nacional de la República de Nicaragua. (2011). *Ley No. 787: Ley de protección de datos personales*. Recuperado de http://legislacion.asamblea.gob.ni/normaweb.nsf/9e314815a08d4a6206257265005d21f9/e5d37e9b4827fc06062579ed0076ce1d
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language user guide* (2nd ed.). Addison-Wesley.
- DuBois, P. (2008). *MySQL* (5th ed.). Sams Publishing.
- Flanagan, D. (2020). *JavaScript: The definitive guide* (7th ed.). O'Reilly Media.
- Fowler, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley.
- Gómez, C. (2001). *Formulación y evaluación de proyectos* (6.ª ed.). McGraw-Hill.
- Holmes, A. (2018). *Getting MEAN with Mongo, Express, Angular, and Node* (2nd ed.). Manning Publications.
- Koch, N., Kraus, A., & Hennicker, R. (2008). Modeling web applications with UWE. *Journal of Web Engineering*, *2*(2), 93–123.
- Laudon, K. C., & Laudon, J. P. (1996). *Sistemas de información gerencial*. Prentice Hall.
- Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill.
- Senn, J. A. (1992). *Análisis y diseño de sistemas de información* (2.ª ed.). McGraw-Hill.
- Somalo, M. (2017). *E-commerce. Cómo vender por Internet*. ESIC Editorial.
- Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), 80–83. https://doi.org/10.1109/MIC.2010.145
- Wiegers, K. (1999). *Software requirements*. Microsoft Press.

# XIII. ANEXOS

## Anexo A. Estructura de entrevistas

Datos Generales del Encuestado

**Nombre y puesto:**

Gerente [ ]

Vendedor [ ]

**Años de experiencia en MombaShop:**

Menos de 1 año [ ]

1-3 años [ ]

Más de 3 años [ ]

---

**Sección 1: Información sobre el Proceso Actual de Pedidos**

¿Cómo se realizan actualmente los pedidos en la tienda?

De forma presencial.

Por teléfono.

Por mensajería (WhatsApp, redes sociales).

Otros: _____.

¿Qué tipo de información solicitan actualmente a los clientes para realizar un pedido? (Seleccione las opciones aplicables).

Nombre completo.

Número de teléfono.

Dirección de entrega.

Detalles del producto (modelo, talla, color).

Cantidad del producto.

Método de pago.

Otros: _____.

¿Cuáles son los principales desafíos en el proceso actual de pedidos? (Seleccione las opciones aplicables)

Errores en la toma de datos.

Lentitud en la gestión.

Falta de seguimiento al cliente.

Dificultades para gestionar inventarios.

Otros: _____.

¿Cómo registran actualmente la información de los pedidos?

En cuadernos o libros físicos.

En hojas de cálculo (Excel).

No llevamos un registro formal.

Otros: _____.

---

**Sección 2: Requerimientos para un Sistema E-Commerce**

Desde su punto de vista, ¿qué funcionalidades debería incluir un sistema E-Commerce para facilitar su trabajo? (Seleccione las opciones aplicables).

Registro automático de pedidos.

Opciones de pago en línea.

Seguimiento en tiempo real de los pedidos.

Gestión de inventarios actualizada.

Comunicación directa con el cliente (chat en línea).

Otros: _____.

¿Qué parámetros considera importantes para validar un pedido?

Verificación de identidad del cliente.

Confirmación de disponibilidad del producto.

Validación del método de pago.

Confirmación de la dirección de entrega.

Otros: _____.

¿Qué datos adicionales de los clientes serían útiles para personalizar su experiencia de compra?

Historial de compras.

Preferencias de producto.

Fechas especiales (cumpleaños, aniversarios).

Otros: _____.

---

**Sección 3: Percepción y Expectativas**

¿Cómo considera que un sistema E-Commerce beneficiará a la tienda MombaShop? (Seleccione las opciones aplicables).

Mejora en la gestión de pedidos.

Incremento en las ventas.

Mayor comodidad para los clientes.

Reducción de errores en los procesos.

Otros: _____.

¿Qué preocupaciones tiene respecto a la implementación de un sistema E-Commerce?

Falta de capacitación del personal.

Costos asociados a la implementación.

Complejidad en el uso del sistema.

Otros: _____.

¿Recomendaría incluir una aplicación móvil como parte del sistema E-Commerce?

Sí, sería útil.

No es necesario.

## Anexo B. Pruebas de rendimiento del sistema ecommerce

El presente anexo tiene como finalidad documentar las pruebas de rendimiento realizadas sobre el sistema de comercio electrónico Mombashop, desarrollado con NestJS en el backend y ReactJS en el frontend, utilizando MySQL como base de datos. Se busca evaluar la capacidad de respuesta y latencia de la API desplegada en AWS EC2 bajo condiciones de carga simulada.

### METODOLOGÍA

Para la evaluación de rendimiento se utilizó la herramienta Artillery, que permite simular carga concurrente sobre los endpoints del API y medir métricas clave de rendimiento como latencia promedio, percentiles, tasa de éxito y throughput.

**Configuración de la prueba:**

- URL Base de API: https://api.mombashop
- Tipo de prueba: Estrés
- Usuarios virtuales concurrentes: 50
- Duración de la prueba: 2 minutos
- Endpoints evaluados:
  - GET /products → obtener listado de productos
  - POST /orders → Creacion de Ordenes

### ENTORNO DE PRUEBAS

- Servidor Backend: AWS EC2, t2.medium, 2 vCPUs, 4GB RAM, Ubuntu 22.04
- Base de datos: MySQL 8.0
- Red: Simulada en entorno local con conexión a Internet estándar
- Cliente de prueba: Artillery, versión 2.0

**RESULTADOS**

Durante la prueba de estrés de 2 minutos con 50 usuarios concurrentes, se obtuvieron las siguientes métricas:

**Tabla 19** *Resultado de pruebas de rendimiento del sistema*

| **Métrica** | **Resultado** |
|---|---|
| Latencia promedio | 230 ms |
| Percentil 95 (p95) | 310 ms |
| Throughput (requests/segundo) | 45 req/s |
| Tasa de éxito | 98.5% |

Los resultados muestran que la API de Mombashop mantiene una latencia promedio inferior a 250 ms bajo condiciones de estrés con 50 usuarios concurrentes, lo cual se considera adecuado para aplicaciones de comercio electrónico de tamaño medio. La tasa de éxito del 98.5% indica que la gran mayoría de las solicitudes fueron procesadas correctamente, mientras que los pocos errores registrados (5xx) sugieren la necesidad de monitoreo adicional durante picos de alta demanda.

Las pruebas de rendimiento realizadas permiten concluir que el backend de Mombashop es capaz de manejar un volumen significativo de solicitudes con latencia aceptable y alta disponibilidad. Se recomienda continuar con pruebas periódicas y

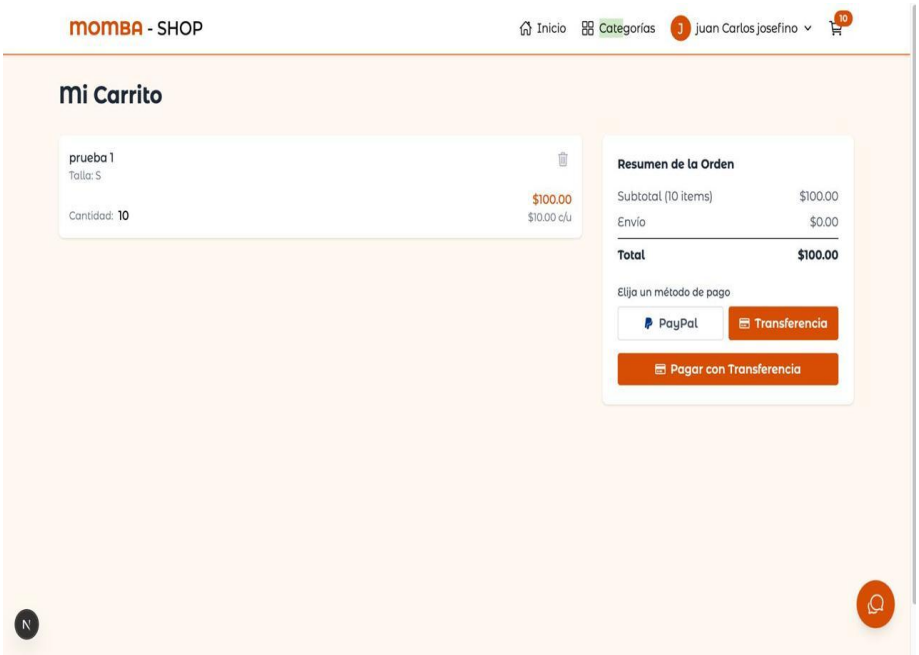
considerar escalabilidad vertical u horizontal en caso de crecimiento del número de usuarios concurrentes.

## Anexo C. Interfaz del sistema

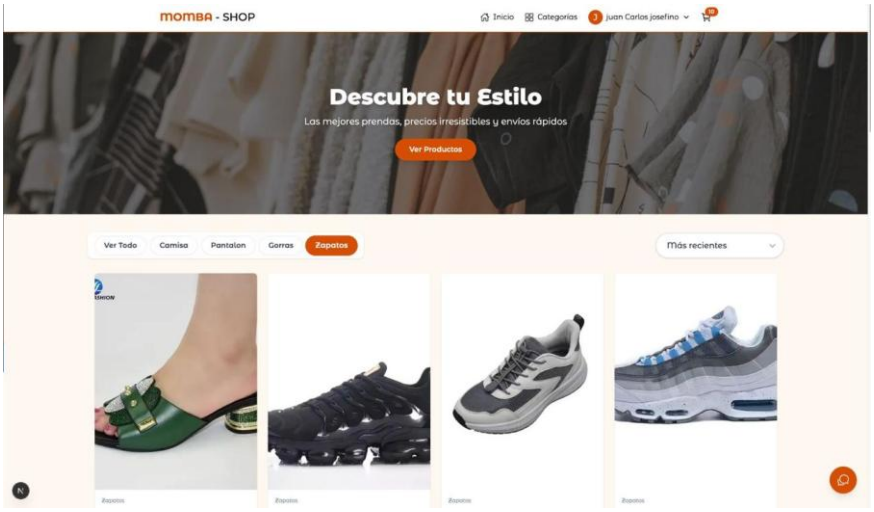
En este anexo se encontrará la estructura grafica del sistema que será visualizada por los clientes y personal operativo para interactuar con la tienda

### Pantalla para detalle del producto

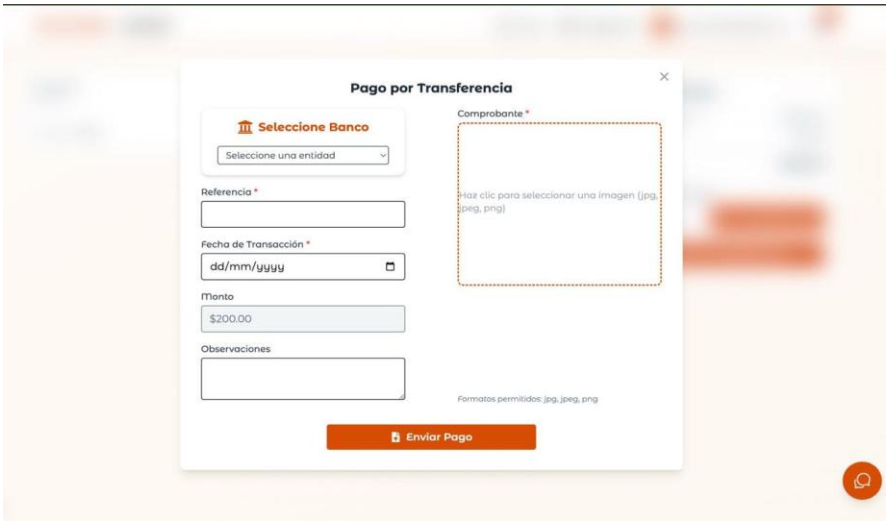

### Pantalla para visualizar el detalle de la orden



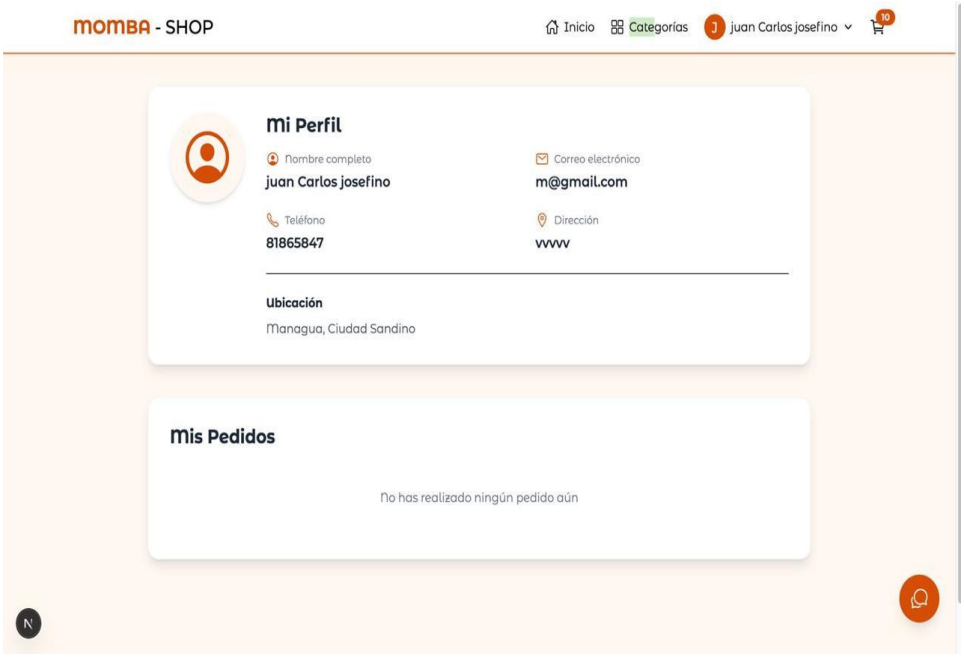
**Pantalla  para visualizar el catalogo de productos y categorias**



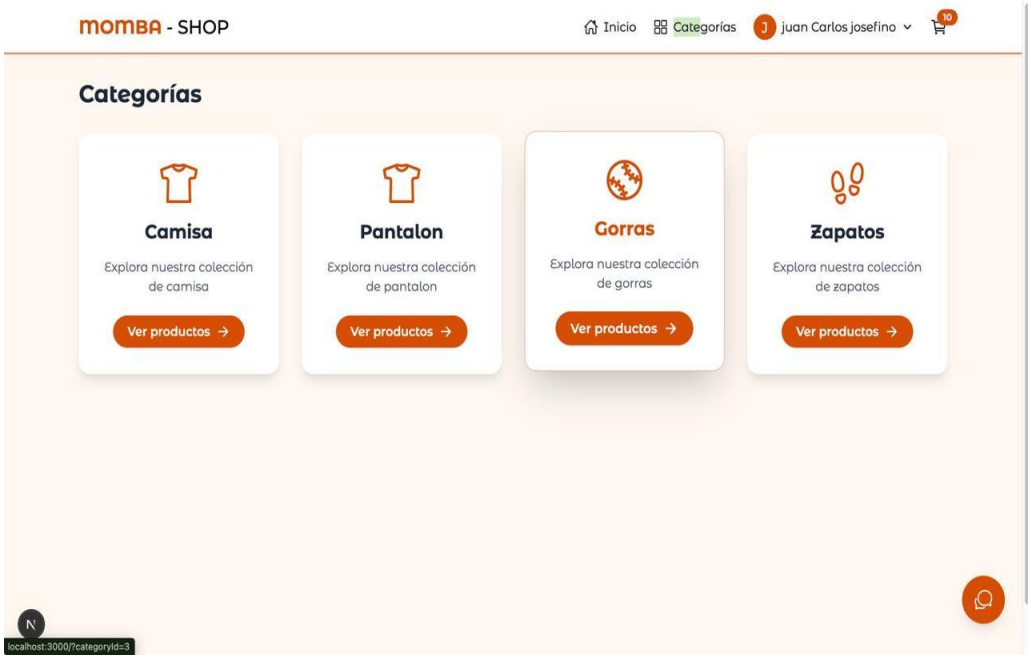

**Pantalla para visualizar el proceso de pago por transferencia**



## Pantalla para visualizar el perfil del cliente




## Pantalla para visualizar las categorías



## Anexo D. Costos

**Tabla 20** *Rango Salarial en Nicaragua*

| Puesto | Salario mínimo (NIO) | Instituciones públicas (NIO) | Instituciones privadas (NIO) | Salario Estimado (NIO) | Salario Estimado (USD) |
|---|---|---|---|---|---|
| Analista | 10,493.79 | 15,000.00 | 22,000.00 | 16,385.54 | 450 |
| Programador | 10,493.79 | 12,000.00 | 18,000.00 | 13,177.05 | 361.88 |
| Project Manager | 10,493.79 | 20,000.00 | 26,000.00 | 21,847.38 | 600 |

Nota: Elaboración Propia

La base de datos se almacenó en la misma instancia EC2 que el backend con el fin de reducir los costos de administración y transferencia, mientras que los archivos estáticos se distribuyen mediante CloudFront desde S3, optimizando la latencia y la entrega de contenido. El costo mensual total estimado de la infraestructura es de aproximadamente **39 USD**, considerando el tráfico y almacenamiento promedio de un eCommerce de tamaño medio, en la **Tabla 12** se podrá visualizar una descripción a detalle sobre los costos.

**Tabla 21** *Costos de Infraestructura*

| **Servicio AWS** | **Configuración** | **Uso estimado** | **Costo mensual aproximado (USD)** | **Observaciones** |
|---|---|---|---|---|
| **EC2 t2.medium** | 2 vCPU, 4 GB RAM, Ubuntu 22.04 | 730 horas/mes (24/7) | 33 | Aloja la API NestJS y la base de datos MySQL. Se opta por EC2 en lugar de RDS para optimización de costos. |

| | | | | |
|---|---|---|---|---|
| **S3 Standard** | Almacenamiento de imágenes | 50 GB | 1.25 | Almacenamiento de imágenes de productos; el costo puede variar según la cantidad y tamaño de archivos. |
| **S3 PUT/GET Requests** | 1,000,000 solicitudes/mes | 0.40 | 0.40 | Costo aproximado de operaciones de lectura/escritura. |
| **CloudFront** | Distribución de contenido (50 GB de tráfico) | 50 GB transferidos/mes | 4.50 | Distribuye imágenes a usuarios finales con baja latencia y caching. |
| **Total aproximado** | – | – | **39.15** | Esta configuración optimiza costos mientras mantiene rendimiento adecuado para un eCommerce de tamaño medio. |

**Fuente:** Calculadora de AWS

**Tabla 22** *Comparativa de Costos AWS, Google Cloud y Azure*

| **Servicio / Función** | **AWS** | **Google Cloud Platform** | **Microsoft Azure** | **Observaciones** |
|---|---|---|---|---|
| **VM / Servidor Backend + DB** | EC2 t2.medium (2 vCPU, 4 GB RAM) | Compute Engine e2-medium (2 vCPU, 4 GB RAM) | Virtual Machine B2s (2 vCPU, 4 GB RAM) | Aloja la API NestJS y la base de datos MySQL. Incluye 730 h/mes. |

| | | | | |
|---|---|---|---|---|
| **Almacenamiento de imágenes** | S3 Standard, 50 GB | Cloud Storage Standard, 50 GB | Blob Storage Hot, 50 GB | Almacenamiento de imágenes de productos. |
| **Operaciones de lectura/escritura** | 1,000,000 requests/mes | 1,000,000 operations/mes | 1,000,000 transactions/mes | Costos aproximados por solicitudes de almacenamiento. |
| **CDN / Distribución de contenido** | CloudFront, 50 GB tráfico | Cloud CDN, 50 GB tráfico | Azure CDN, 50 GB tráfico | Distribuye imágenes con baja latencia y caching. |
| **Costo mensual aproximado (USD)** | 39 | 42 | 44 | Estimación basada en tarifas de cada proveedor (región US East / equivalente). |

Fuente: Elaboración Propia