# UNIVERSIDAD DE CIENCIAS COMERCIALES UCC - SEDE MANAGUA

## COORDINACIÓN DE INGENIERIA INFORMÁTICA Y TELECOMUNICACIONES

**Título: Diseño de un Sistema de Inventario para una Pyme IBAGU STORE ubicado en Plaza Eclipse en Managua, en el segundo semestre del año 2025.**

**Autor: Br. Ronald Antonio Martínez Rosales.**

**Asesor Metodológica y Técnico: Ing. Jacqueline del Socorro Lacayo Cruz**
**Máster en Educación Superior.**

**Managua, noviembre 2025**

# UNIVERSIDAD DE CIENCIAS COMERCIALES

# UCC – SEDE MANAGUA

## COORDINACIÓN DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

**Informe de Trabajo Final para optar al título de Ingeniero en Informática y Telecomunicaciones**

**AVAL DEL TUTOR**

Grado Académico y nombre del tutor(es), tienen a bien:

**CERTIFICAR**

**Que**: El Informe Final con el título: **"Diseño de un Sistema de Inventario para una Pyme IBAGU STORE ubicado en Plaza Eclipse en Managua, en el segundo semestre del año 2025."**, elaborado por el **Br. Ronald Antonio Martínez Rosales,** ha sido dirigido por los suscritos.

Al haber cumplido con los requisitos académicos y metodológicos del informe final, damos de conformidad a la presentación de dicho informe final para proceder a su lectura y defensa, de acuerdo con la normativa vigente del Reglamento de Régimen Académico Estudiantil y Reglamento de Investigación, Innovación y Transferencia.

Para que conste donde proceda, se firma la presente en UCC Sede/Campus a 18 del mes de noviembre del año 2025.

_____

**Msc. Jacqueline del Socorro Lacayo Cruz**

**Tutor Técnico y Tutor Metodológico**

## DEDICATORIA

Queridos lectores y evaluadores,

Este trabajo monográfico es el fruto de dedicación, esfuerzo y pasión por el conocimiento. A cada uno de ustedes que ha contribuido a mi crecimiento académico, quiero expresar mi más sincero agradecimiento.

A mis profesores, cuyas enseñanzas han iluminado mi camino académico y han nutrido mi curiosidad. Agradezco su paciencia, guía y sabiduría compartida.

A mis amigos y compañeros de estudio, por los momentos compartidos, las risas, y el respaldo mutuo. Juntos hemos enfrentado desafíos y celebrando logros, creando recuerdos imborrables.

A mi pareja la cual me ha acompañado más de 4 años y ha sido mi fiel compañera en toda esta travesía y siempre apoyo mi crecimiento desde el día 1.

Pero principalmente a mi madre y a mi tía Marina sin las cuales esto no hubiese sido posible su amor y apoyo me ayudo desde el día uno y nunca dejaron de apoyarme gracias a ellas logre convertirme en el profesional que soy hoy en día.

## AGRADECIMIENTO

Me complace expresar mi más sincero agradecimiento a todos aquellos que han contribuido de manera significativa a la realización de este trabajo monográfico. Este proyecto no habría sido posible sin el apoyo y la colaboración de diversas personas a las que quiero reconocer.

En primer lugar, quiero agradecer a mis profesores y asesores, cuya orientación experta y dedicación han sido fundamentales para dar forma a este trabajo. Sus valiosas sugerencias y comentarios han enriquecido enormemente mi investigación.

A mi familia, les agradezco por su apoyo incondicional y por ser mi fuente constante de inspiración. Su aliento y comprensión han sido mi impulso durante todo este proceso.

A mis amigos y compañeros de estudio, gracias por compartir sus ideas, brindar perspectivas frescas y ser una red de apoyo en los momentos desafiantes. La colaboración y el intercambio de conocimientos han sido esenciales para el desarrollo de este trabajo.

A todas las personas que participaron en entrevistas, encuestas o proporcionaron información valiosa, su contribución fue fundamental para la calidad y relevancia de este proyecto.

Este trabajo lleva consigo la influencia positiva de cada persona que ha cruzado mi camino académico, y por ello, mi agradecimiento se extiende a todos aquellos que, de una u otra manera, han dejado su marca en este viaje.

Ronald Martínez.

## RESUMEN

El presente proyecto de investigación describe el análisis, y propuesta técnica de un sistema de inventario orientado a la Pyme Ibagu Store, con el objetivo de optimizar el control de productos, ventas y gastos mediante la automatización de procesos actualmente manuales. El estudio aborda la viabilidad operativa, técnica, económica y financiera, incorporando un análisis detallado de requerimientos funcionales y no funcionales, así como la definición de una arquitectura basada en el stack MERN, adaptado para utilizar PostgreSQL y PostgREST como capa de datos.

La metodología aplicada combina entrevistas, observación directa y talleres de codiseño con usuarios clave, complementada con la estimación de esfuerzo mediante el modelo COCOMO II y la evaluación financiera (VPN, TIR y periodo de recuperación). El diseño propuesto incluye autenticación JWT, control de acceso basado en roles (RBAC), políticas de seguridad a nivel de fila (RLS) y despliegue contenedorizado con Docker y CI/CD.

Los resultados esperados contemplan una reducción significativa de errores en el manejo de inventario, mejora en la trazabilidad de movimientos y generación de reportes estratégicos (rotación, bajo stock, análisis ABC), contribuyendo a la toma de decisiones y a la eficiencia operativa de la organización.



Palabras Claves: sistema de inventario, stack MERN, COCOMO.

## Abstract

This research project presents the analysis, design, and technical proposal of an inventory management system for the small business Ibagu Store, aimed at optimizing product control, sales, and expense tracking through process automation. The study evaluates the system's feasibility from operational, technical, economic, and financial perspectives, incorporating a detailed analysis of functional and non-functional requirements and defining an architecture based on the MERN stack adapted to use PostgreSQL with PostgREST as the data layer.

The methodology combines qualitative techniques such as interviews, direct observation, and co-design workshops with key users, complemented by effort estimation using the COCOMO II model and financial evaluation through indicators such as NPV, IRR, and payback period. The proposed design includes JWT-based authentication, role-based access control (RBAC), row-level security (RLS) in PostgreSQL, and containerized deployment with Docker and CI/CD pipelines.

Expected outcomes include a significant reduction in inventory management errors, improved traceability of stock movements, and the generation of strategic reports (e.g., stock alerts, ABC analysis, and rotation metrics), thereby enhancing decision-making and operational efficiency for the organization.



Keywords: inventory system, MERN stack, COCOMO.

# 1. Índice de Contenido


1. Introducción ..... 7
   1.1. Antecedentes y contexto del problema ..... 8
   1.2. Objetivos ..... 10
   1.3. Preguntas de investigación ..... 10
   1.4. Justificación ..... 11
   1.5. Limitaciones ..... 11
   1.6. Supuestos Básicos ..... 12
   1.7. Patrones emergentes de la investigación ..... 12
2. Marco Teórico ..... 13
   2.1. Estado del Arte ..... 13
   2.2. Teorías y conceptualizaciones asumidas ..... 13
3. Diseño Metodológico ..... 34
   3.1. Enfoque cualitativo asumido y su justificación ..... 34
   3.2. Muestra teórica y sujetos del estudio ..... 34
   3.3. Métodos y técnicas de recolección de datos utilizados ..... 35
5. Conclusiones ..... 41
6. Recomendaciones ..... 44
7. Referencias ..... 46
8. Glosario de términos técnicos ..... 47
9. Anexos o Apéndices ..... 50

## 1. Introducción

Un sistema de inventario automatizado es una herramienta tecnológica que la compone un hardware y un software que ayuda indagar para gestionar diversos niveles de stocks de una empresa de forma automática, para monitorear automáticamente la entrada y salida de los productos que la empresa se encarga de distribuir o vender, generando a la vez órdenes de compra, mantenimiento de bases de datos de inventario, de forma actualizada en tiempo real para evitar la falta o el exceso de stocks

La gestión efectiva de usar un inventario automatizado, Ibagu Store donde considera que puede construir su columna vertebral para el manejo de su stock, Ibagu Store es una pequeña empresa del sector retail dedicada a la venta de accesorios, considera una opción para optimizar sus procesos internos de control de su inventario de bodega, mediante el diseño de un sistema de inventario automatizado.

Esta investigación aborda esta viabilidad por tanto el diseño de dicho sistema, para resolver las posibles carencias en el flujo de trabajo, en la actual gestión de control de cantidades de productos, ventas y control de gastos, sustituyendo este control que es monitoreado de forma manual por un proceso digital más eficiente.

La funcionalidad que se propone incluye administración de catálogos de productos, control de existencias y movimientos (entradas, salidas y ajustes), gestión de proveedores y compras, registro de ventas y generación de reportes estratégicos (rotación, bajo stock, análisis ABC).

Este diseño de sistema se empleó un stack MERN adaptado con PostgreSQL y PostgREST como capa de datos, incorporando autenticación JWT, control de acceso basado en roles (RBAC) y despliegue contenedorizado con Docker.

La metodología utilizada combina técnicas cualitativas (entrevistas, observación y talleres de codiseño) con estimación de esfuerzo mediante el modelo COCOMO II y análisis financiero (VAN, TIR y periodo de recuperación).

Esta propuesta busca mejorar la adopción tecnológica en la empresa, reducir posibles errores en la gestión de inventario y aumentar la productividad en la generación de reportes, contribuyendo a la toma de decisiones estratégicas.

### 1.1. Antecedentes y contexto del problema

Ibagu Store es una pequeña empresa del sector retail ubicada en Plaza Eclipse, Managua, Nicaragua, se dedica a comercializar accesorios como relojes, gorras y artículos para dispositivos móviles. Conformada con el objetivo de ofrecer productos de calidad a precios competitivos en el sector de artículos para celulares, Ibagu Store ha experimentado un crecimiento en su base de clientes, incrementando la complejidad en la gestión de inventarios y control de ventas.

Actualmente, la administración de productos, movimientos de stock y registro de ventas se realiza en hojas de cálculo de Excel elaboradas y grabadas de forma manual, generando posibles limitaciones significativas, como falta de trazabilidad, duplicidad de datos y retrasos en la generación de reportes para un control efectivo para una toma de decisiones, generando posibles riesgos de un control poco fiable en el stock, sobre inventario y pérdida de oportunidades comerciales.

A esta problemática operativa se suma la incapacidad de escalabilidad del modelo actual. Al depender de procesos manuales no estandarizados, el conocimiento del negocio se centraliza en el personal que maneja las hojas de cálculo, creando una dependencia crítica que pone en riesgo la continuidad operativa ante la rotación de personal. Esta falta de institucionalización de la información impide a la gerencia realizar auditorías históricas fiables, lo que dificulta detectar mermas, robos hormiga o vencimiento de productos de baja rotación, impactando directamente en la rentabilidad neta de la empresa.

Además, la desconexión entre el inventario físico y el registro digital genera una brecha en la calidad del servicio al cliente. La incertidumbre sobre la disponibilidad real de un producto en el momento de la venta obliga al personal a realizar verificaciones físicas constantes, incrementando los tiempos de espera y afectando la percepción de eficiencia del negocio. En un entorno de retail cada vez más competitivo, esta latencia en la información no solo representa una ineficiencia administrativa, sino una desventaja competitiva que limita las posibilidades de Ibagu Store para proyectar su crecimiento hacia canales digitales o sucursales adicionales.

## 1.2. Objetivos

### Objetivo General:

Diseñar un sistema de inventario para control de gastos y ventas utilizando stack MERN con PostgreSQL / PostgREST en la tienda Ibagu Store, en el segundo semestre del año 2025.

### Objetivos Específicos:

1. Determinar los requerimientos funcionales y no funcionales necesarios para la implementación del sistema, adaptados a Ibagu Store.
2. Modelar el sistema utilizando UML, PostgreSQL, para representar los diagramas de desarrollo.
3. Calcular el presupuesto para el diseño de Sistema de Inventario.

## 1.3. Preguntas de investigación

Actualmente, Ibagu Store enfrenta diversos desafíos en la gestión de inventarios y control de ventas debido a la ausencia de un sistema automatizado. Entre los principales problemas identificados se encuentran:

1. ¿La gestión de inventario de Ibagu Store cumple con los requerimientos técnicos de un control de inventario?
2. ¿El sistema de inventario de Ibagu Store permite la trazabilidad de las entradas y salidas de inventarios?
3. ¿El uso de sistema de inventario en Ibagu Store es seguro para los datos de los productos?

## 1.4. Justificación

En los últimos años, Ibagu Store ha aumentado sus ventas y la variedad de productos, pero esto ha llevado a un mayor control manual del inventario y dependencia de hojas de cálculo, lo que consume tiempo y genera errores. En un mercado competitivo y digital, seguir así limita el crecimiento del negocio.

Como futuro Ingeniero en Informática y Telecomunicaciones, este proyecto es una oportunidad para aplicar conocimientos de software y bases de datos. Es esencial analizar los requerimientos del cliente para resolver problemas reales, como la falta de trazabilidad y la dificultad para generar reportes confiables. Un sistema de inventario es vital para controlar el stock y optimizar procesos, lo que resulta en decisiones más rápidas y menos pérdidas.

Digitalizar una PYME como Ibagu Store no solo mejora su eficiencia, sino que también fortalece la economía local y fomenta la transformación digital en Nicaragua. Este proyecto busca crear una herramienta que haga a Ibagu Store más competitiva y que contribuya a un modelo económico más moderno e inclusivo.

Este tipo de situaciones solicita modernizar sus procesos, Ibagu Store percibe la necesidad de transformar de forma digital mediante el diseño de un sistema de inventario que automatice operaciones críticas, que mejore la precisión de los datos y proporcione reportes fiables y estratégicos en tiempo real.

## 1.5. Limitaciones

Diseñar un Sistema de Inventarios, es identificar el proyecto que es la fase de análisis y diseño, sin incluir la implementación, pruebas ni puesta en marcha. Además, los beneficios estimados, como reducción de errores y mejora en la eficiencia, carecen de pruebas en un entorno real.

El trabajo se realiza en un periodo académico específico, lo que impide incluir funciones adicionales o hacer simulaciones. También podría no considerar la integración con

sistemas externos necesarios. Por último, la aceptación de la empresa es crucial, pero no se incluye en este alcance actual.

### 1.6. Supuestos Básicos

Para la implementación del sistema de inventario en Ibagu Store, se establecen varios supuestos.

Primero, se asume que habrá acceso a internet confiable.
Segundo, se espera la colaboración del personal clave en todas las etapas del proyecto.
Tercero, se prevé capacitación básica para los usuarios más adelante.
Cuarto, la empresa adoptará políticas claras para la gestión de datos, como la unicidad de códigos SKU y auditorías periódicas.
Finalmente, se considera que habrá infraestructura tecnológica adecuada, incluyendo computadoras y dispositivos con navegadores actualizados.

### 1.7. Patrones emergentes de la investigación

Durante el análisis y revisión de literatura de Ibagu Store, se han definido categorías y temas importantes para la investigación. Las categorías principales incluyen la gestión de inventarios, la transformación digital en Pymes, la seguridad y control de acceso, y el análisis financiero y viabilidad.

Los temas recurrentes abarcan la automatización de procesos manuales, la generación de reportes estratégicos y la escalabilidad en la nube. Los patrones emergentes destacan la integración de tecnologías abiertas, el enfoque en la experiencia de usuario y la seguridad como prioridad.

# 2. Marco Teórico

## 2.1. Estado del Arte

La gestión de inventarios en pequeñas y medianas empresas del sector retail es vital para su funcionamiento, optimización de recursos y satisfacción del cliente. Sin embargo, muchas de estas empresas usan procesos manuales o herramientas simples como hojas de cálculo, lo que puede aumentar errores y retrasar decisiones.

La transformación digital ha fomentado el uso de sistemas de gestión de inventarios modernos y basados en la web, que son escalables y accesibles. Estas soluciones incluyen un catálogo de productos, control de existencias, gestión de proveedores, registro de ventas, y reportes estratégicos.

En cuanto a la tecnología utilizada, se prefieren herramientas como React para interfaces y Node.js con Express para el backend, con PostgreSQL como base de datos robusta. También se recomienda el uso de PostgREST para facilitar la creación de APIs RESTful.

Las mejores prácticas de seguridad incluyen autenticación con JWT, control de acceso, y protección contra ataques. La adopción de estas tecnologías mejora la eficiencia y permite a las pymes competir mejor en un mercado cada vez más ágil.

## 2.2. Teorías y conceptualizaciones asumidas

El marco conceptual de este proyecto se basa en los siguientes conceptos clave:

a) Sistema de inventario: Se refiere a un sistema informático que permite la gestión y almacenamiento de productos para mejor control sobre estos y mantener un orden de los ya existentes y otros datos relevantes para la administración eficiente.
b) Eficiencia: La eficiencia se relaciona con la capacidad del sistema para realizar tareas de manera rápida y sin desperdicio de recursos. En este contexto, un sistema de inventario eficiente reducirá la carga de trabajo administrativo y mejorará la productividad.

c) Modernización: La modernización se refiere a la adopción de tecnologías y prácticas actualizadas que mejoran los procesos y la funcionalidad del sistema de inventario.
d) Necesidades Específicas: Hace referencia a los requisitos únicos y específicos de la tienda Ibagu Store en términos de gestión de inventario. Estas necesidades pueden estar relacionadas con la estructura operativa del negocio, políticas internas de control de productos, y requisitos legales o fiscales aplicables al comercio minorista en Nicaragua.

El proyecto se basa en conceptos teóricos de la ingeniería de software, la gestión de proyectos y la tecnología de la información. Algunos de los conceptos teóricos relevantes incluyen:

### 2.2.1 ¿Qué son los requerimientos funcionales?

Los requerimientos funcionales constituyen una categoría fundamental dentro del análisis y diseño de sistemas, ya que definen las funciones específicas que el sistema debe realizar para satisfacer las necesidades del usuario o del negocio. En el contexto de la ingeniería de software, estos requerimientos se expresan en términos de comportamientos observables, describiendo las entradas, procesos y salidas que el sistema debe gestionar.

Los requerimientos funcionales son esenciales porque permiten establecer la base operativa del sistema, asegurando que las funcionalidades respondan a los objetivos planteados en el proyecto. Su correcta identificación y documentación facilita la validación del sistema mediante pruebas funcionales, garantizando que cada función cumpla con lo esperado.

### 2.2.2 ¿Qué son los requerimientos no funcionales?

Los requerimientos no funcionales complementan a los funcionales, estableciendo las condiciones y restricciones bajo las cuales el sistema debe operar, así como los atributos de calidad que debe cumplir. Estos requerimientos no describen funciones concretas,

sino características que afectan la eficiencia, seguridad, usabilidad y rendimiento del sistema.

Metodológicamente, los requerimientos no funcionales son críticos porque determinan la calidad del producto final, influyendo en la arquitectura, el diseño y la experiencia del usuario. Su definición clara permite establecer métricas de desempeño y estándares que aseguren la confiabilidad del sistema en entornos reales.

### 2.2.3 ¿Qué es un sistema de información?

Un sistema de información es un conjunto organizado de componentes interconectados que trabajan juntos para recopilar, procesar, almacenar y distribuir información para apoyar la toma de decisiones y la gestión de una organización. Los sistemas de información pueden variar en tamaño y complejidad, desde sistemas simples utilizados para gestionar la información de una pequeña empresa hasta sistemas empresariales complejos utilizados por grandes corporaciones (M., 2016).

Un sistema de información consta de varios elementos clave:

#### 2.2.1.1 Entradas

Son los datos que se recopilan o ingresan en el sistema. Estos datos pueden provenir de fuentes externas o internas y son la materia prima del sistema.

#### 2.2.1.2 Procesamiento

Los datos se procesan y se transforman en información útil. Esto puede incluir tareas como clasificar, calcular, resumir, organizar y filtrar datos.

#### 2.2.1.3 Almacenamiento

La información procesada se almacena en una base de datos o en otros medios de almacenamiento. Esto garantiza que la información esté disponible cuando sea necesaria.

#### 2.2.1.4 Salida

La información procesada se presenta en forma de informes, gráficos, tablas u otros formatos que son comprensibles y útiles para los usuarios.

#### 2.2.1.5 Usuarios

Las personas que interactúan con el sistema de información, ya sea para ingresar datos, realizar consultas o tomar decisiones basadas en la información generada.

#### 2.2.1.6 Procesos y procedimientos

Los flujos de trabajo y las operaciones que gobiernan cómo se recopilan, procesan y utilizan los datos.

Un sistema de información puede ser automatizado, como un sistema de gestión de bases de datos o un sistema de planificación de recursos empresariales (ERP), o puede ser manual, como un sistema de archivo y seguimiento de documentos en una pequeña empresa.

Los sistemas de información desempeñan un papel fundamental en la gestión de una organización y son esenciales para la toma de decisiones informadas. Proporcionan a los directivos y a los empleados acceso a información actualizada y precisa que les permite planificar, controlar y coordinar las actividades de la organización de manera más eficaz (C., SIstemas de Informacion Gerencial, 2016).

### 2.2.2 ¿Qué es un sistema de información Web?

Una aplicación Web es un software o programa informático que se ejecuta en un navegador Web y que permite a los usuarios interactuar con servicios, funciones y datos a través de la World Wide Web. A diferencia del software tradicional, que se instala en un dispositivo específico, las aplicaciones Web son accesibles desde cualquier dispositivo con un navegador Web y una conexión a Internet.

Este software o programa informático que se ejecuta en un servidor Web y es accesible desde un navegador Web en dispositivos cliente, como computadoras, teléfonos móviles y tabletas. Estas aplicaciones se diseñan para ofrecer una variedad de funciones, como interacción con bases de datos, procesamiento de datos, gestión de contenido,

Un sistema ERP (Enterprise Resource Planning) es un software o conjunto de aplicaciones integradas que permite a una organización gestionar y automatizar una variedad de procesos y funciones empresariales en un único sistema unificado. Los

sistemas ERP están diseñados para ayudar a las empresas a mejorar la eficiencia, la visibilidad y la toma de decisiones al proporcionar una vista integral de sus operaciones

El concepto de "Web o Web informática" se refiere a la interconexión de computadoras a través de una red global, que permite el intercambio de información y servicios a través de Internet. La Web informática es un término amplio que engloba todas las actividades, servicios y recursos relacionados con la informática en línea. Esto incluye la transmisión de datos, el acceso a sitios web, el uso de aplicaciones en la nube, la comunicación en línea, la administración de servidores, la seguridad informática y muchas otras áreas de la tecnología de la información que se basan en la infraestructura de Internet.

La World Wide Web, comúnmente abreviada como "WWW" o simplemente "Web", es un sistema de información global y público que opera en Internet. Fue inventada por Tim Berners-Lee en 1989 y se ha convertido en una de las aplicaciones más populares de la red mundial.
Una aplicación Web también conocido como sistema de información en línea o sistema de información basado en Web, es un tipo de sistema de información que utiliza tecnologías Web para recopilar, procesar, almacenar y distribuir información en línea.

Estos sistemas permiten a los usuarios acceder a la información a través de navegadores Web desde cualquier lugar con conexión a Internet. Pueden variar en complejidad, desde simples sitios Web que ofrecen información estática hasta aplicaciones Web altamente interactivas y dinámicas que permiten a los usuarios realizar transacciones, colaborar y acceder a bases de datos en tiempo real (C., Management Information Systems, 2019).

Los sistemas de información Web suelen incluir elementos como bases de datos en línea, interfaces de usuario amigables, capacidad de búsqueda y navegación, así como la posibilidad de presentar información en diferentes formatos, como texto, gráficos, audio y video. (Turban, 2005).

Los beneficios que presentan los sistemas orientados a la Web son de gran ventaja y rentabilidad para la empresa o institución, entre los cuales se mencionan los siguientes:

- Uso de una única versión del sistema para los usuarios, una sola base de datos centralizada y poder contar con la información exacta y oportuna para la toma de decisiones.
- Solo se requiere tener acceso a internet, y se puede utilizar en cualquier momento y lugar.
- No necesariamente se necesitan equipos de alto costo para el uso de dicha aplicación, debido a que esta se encuentra alojada en un servidor Web sea físico o en la nube.
- Los sistemas de información Web se construyen a partir de dos conceptos fundamentales dentro de la programación, Frontend y Backend.

### 2.2.3 La Estandarización de las Aplicaciones Web

La estandarización de las aplicaciones Web es un proceso importante que implica la definición y adopción de estándares abiertos y comunes para el desarrollo y la implementación de aplicaciones en la Web.

Esto es esencial para garantizar la interoperabilidad, la accesibilidad, la seguridad y la sostenibilidad de las aplicaciones Web en un entorno en constante evolución. Algunos aspectos clave de la estandarización de las aplicaciones Web incluyen:

a) Estándares Web: Los estándares Web son especificaciones técnicas que definen cómo deben comportarse las tecnologías y protocolos Web, como
b) HTML, CSS, JavaScript, HTTP, etc. La estandarización de estas tecnologías asegura que las aplicaciones Web sean consistentes en diferentes navegadores y dispositivos.
c) Interoperabilidad: Los estándares Web facilitan la interoperabilidad entre diferentes sistemas, permitiendo que aplicaciones y servicios Web funcionen de manera coherente en diferentes entornos y plataformas.

d) Accesibilidad: Los estándares Web también se centran en la accesibilidad, garantizando que las aplicaciones sean utilizables por todas las personas, incluyendo aquellas con discapacidades, al seguir pautas como las Pautas de Accesibilidad para el Contenido Web (WCAG).
e) Seguridad: La estandarización de prácticas de seguridad Web es esencial para proteger las aplicaciones Web y los datos de los usuarios. Esto incluye la implementación de HTTPS, medidas de seguridad en el lado del servidor y en el lado del cliente, y la gestión segura de contraseñas.

La estandarización Web se lleva a cabo a través de organizaciones como el World Wide Web Consortium (World Wide Web Consortium, s.f.) y el Internet Engineering Task Force (The Internet Engineering Task Force, s.f.), que definen y mantienen los estándares Web.

Estas organizaciones trabajan en colaboración con la comunidad de desarrollo Web para garantizar que las tecnologías y estándares evolucionen de manera abierta y consensuada.

La estandarización de las aplicaciones Web es esencial para el crecimiento y el éxito continuo de la Web al proporcionar una base sólida para el desarrollo de aplicaciones y servicios en línea.

### 2.2.4 Lenguajes de programación utilizados en programación Web

La programación Web desempeña un papel fundamental en la creación y el mantenimiento de sitios Web y aplicaciones en línea. Los lenguajes de programación son las herramientas que permiten a los desarrolladores dar vida a estas experiencias digitales.

En este contexto, una amplia variedad de lenguajes de programación se utiliza para construir tanto el frontend (parte visible para los usuarios) como el backend (la lógica detrás de escena) de las aplicaciones Web.

A continuación, los lenguajes de programación más importantes para la programación de aplicaciones Web:

#### 2.2.4.1 HTML (HyperText Markup Language)

Es el lenguaje de marcado estándar utilizado para crear páginas Web y definir su estructura y contenido. Se trata de un conjunto de etiquetas o elementos que se utilizan para describir cómo se deben mostrar y organizar los elementos en una página Web. Estas etiquetas permiten definir títulos, párrafos, encabezados, enlaces, imágenes, formularios y otros elementos visuales y funcionales de una página Web (Duckett, 2011).

HTML utiliza una estructura jerárquica basada en etiquetas anidadas, lo que significa que los elementos HTML se organizan de manera que algunos elementos contienen a otros. La mayoría de las etiquetas HTML vienen en pares, una de apertura y otra de cierre, y el contenido se coloca entre estas etiquetas. Por ejemplo, a como se muestra en la Ilustración 1, un párrafo se define mediante las etiquetas  y :

Ilustración 1 Ejemplo de Sintaxis de Código HTML

```html
<p>Este es un ejemplo de párrafo en HTML.</p>
```

HTML utiliza una estructura jerárquica basada en etiquetas anidadas, lo que significa que los elementos HTML se organizan de manera que algunos elementos contienen a otros.

La mayoría de las etiquetas HTML vienen en pares, una de apertura y otra de cierre, y el contenido se coloca entre estas etiquetas. HTML también permite enlazar documentos, crear hipervínculos a otras páginas Web, definir listas, tablas, formularios y mucho más.

Además, HTML proporciona la estructura básica que luego se estiliza y formatea utilizando CSS (Cascading Style Sheets) para lograr el aspecto visual deseado en una página Web.

HTML5 es la última versión de HTML, se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance.

Diseñado para ser utilizable por todos los desarrolladores de Open Web, esta página hace referencia a numerosos recursos sobre las tecnologías de HTML5, clasificados en varios grupos según su función.

#### 2.2.4.2 CSS (Cascading Style Sheets)

Es un lenguaje de estilo utilizado en el desarrollo Web para definir la presentación y el formato visual de las páginas Web. CSS se utiliza para aplicar estilos, como colores, fuentes, márgenes, espaciado, alineación y diseño a los elementos HTML, permitiendo a los diseñadores y desarrolladores controlar la apariencia de una página Web de manera eficiente y coherente (Meyer, 2006).

CSS es esencial en la creación de sitios Web modernos, ya que separa la estructura y el contenido (definidos con HTML) de la presentación visual. Al utilizar CSS, los diseñadores pueden lograr una mayor consistencia en el aspecto de un sitio Web, mejorar la accesibilidad y facilitar la adaptación a diferentes dispositivos y tamaños de pantalla.

En CSS, se definen reglas de estilo que se aplican a elementos HTML mediante selectores. Por ejemplo, a como se muestra en la Ilustración 2, para cambiar el color del texto en un párrafo, se podría utilizar una regla CSS como esta:

Ilustración 2 Ejemplo de Sintaxis de Código CSS

```css
p {
  color: blue;
}
```

Esta regla CSS establece que todos los elementos <p> (párrafos) en la página tendrán el color del texto azul.

Se le denomina Hojas de Estilos en Cascada porque las características se aplican de arriba a abajo mediante reglas que poseen un esquema prioritario.

Esta especificación es un lenguaje de diseño gráfico que se escribe dentro del código HTML del sitio Web y, permite crear páginas de una manera más exacta y aplicarles estilos (colores, márgenes, formas, tipos de letras, etc.) por lo que se tiene mayor control de los resultados finales.

La tecnología CSS está diseñada para marcar la separación del contenido de las páginas Web y la forma de presentación de estas. Lo que genera múltiples beneficios, como:

- Presentar el documento final en diferentes estilos (pantalla, voz, impresión).
- Tener un sitio Web responsivo.
- Evitar hacer archivos demasiado pesados.
- Definir el estilo visual de todo un sitio Web. Así, si cambiamos una página, cambiarán todas automáticamente.
- Trabajar con estándares y separar (hasta cierto punto) la estructura de la presentación logrando un trabajo más definido.
- Provee más flexibilidad y control en las especificaciones del sitio Web.
- Simplifica la creación de la página.

El HTML y CSS tienen una relación muy fuerte entre ellos, ya que el HTML es un lenguaje de marcado (constituye la base de un sitio Web) y CSS define el estilo de la página (toda la parte estética).

A pesar de que las Hojas de Estilo en Cascada no son imprescindibles, son importantes para darle forma y apariencia a la página Web. Si solo se usa HTML la página se verá desnuda y no destacaría.

El HTML y CSS tienen una relación muy fuerte entre ellos, ya que el HTML es un lenguaje de marcado (constituye la base de un sitio Web) y CSS define el estilo de la página (toda la parte estética).

A pesar de que las Hojas de Estilo en Cascada no son imprescindibles, son importantes para darle forma y apariencia a la página Web. Si solo se usa HTML la página se verá desnuda y no destacaría.

#### 2.2.4.3 JavaScript

JavaScript es un lenguaje de programación interpretado que se ejecuta en el lado del cliente, lo que significa que se ejecuta en el navegador Web del usuario. Permite a los desarrolladores Web crear interacciones en tiempo real, validar formularios, manipular el contenido de una página Web y mucho más. JavaScript es esencial para el desarrollo de aplicaciones Web interactivas y dinámicas (Haverbeke, 2018).

JavaScript permite implementar funciones complejas en páginas Web, cada vez que una página Web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., esta es la tercera de las tecnologías Web estándar, dos de las cuales son HTML y CSS.

#### 2.2.4.4 Framework

En programación, un "Frameworks" (marco de trabajo en español) es un conjunto de herramientas, bibliotecas y estándares que se utilizan como base para desarrollar aplicaciones de software. Un Framework proporciona una estructura y una metodología para facilitar el desarrollo de aplicaciones al proporcionar componentes reutilizables y abstracciones de alto nivel para tareas comunes.

Los Frameworks se utilizan para acelerar el desarrollo de software, mejorar la eficiencia, mantener buenas prácticas de codificación y garantizar la consistencia en la aplicación. Permiten a los desarrolladores enfocarse en la lógica específica de su aplicación en lugar de preocuparse por la implementación de tareas repetitivas (Erich Gamma, 1994).

En pocas palabras es una estructura previa que se puede aprovechar para desarrollar un proyecto. Permite entregar un proyecto en menos tiempo y con un código más limpio.

Los Frameworks utilizados para Internet son sólo uno de los tantos que existen hoy en día. Algunos de ellos son:

- Para aplicaciones Web: Son aquellos Frameworks que se utilizan específicamente para la creación de proyectos online. Desde el diseño Web de una página hasta los servicios Web más específicos que puedas imaginarte.

  Dentro de estos Frameworks existen otros tipos, dependiendo del lenguaje de programación utilizado. Sin embargo, nada impide que se pueda utilizar un Framework originalmente pensado en un lenguaje de programación, en otro diferente.

- Para un buen programador en muchos casos es más fácil adaptar un lenguaje a otro, que modificar un proyecto con diferentes objetivos.

- Para aplicaciones en general: Permite complementar la estructura de una aplicación para un sistema operativo. Por ejemplo, Microsoft ha desarrollado el .NET Framework que ayuda a los programadores a reutilizar estructuras ya elaboradas. Este Framework ya viene instalado en su sistema operativo, el popular Windows.
- Para tecnología AJAX: La tecnología AJAX permite que el usuario haga solicitudes al servidor sin que sea necesario recargar una página después de cada nueva solicitud.

De esta forma, las informaciones van surgiendo a medida que son solicitadas sin la necesidad de que la página quede recargándose.

Existen Frameworks específicos para esta tecnología, permitiendo la reutilización de un código ya elaborado.

- De gestión de contenidos: A estos Frameworks también se les conoce como CMF, que significa, Content Manager Framework y facilita la programación de aplicaciones de un Sistema de Gestión de Contenidos, popularmente conocido como CMS, por ejemplo, WordPress.
  Existe una gran variedad de CMF de acuerdo con la plataforma para la que será creada la aplicación.

- De Multimedia: Esta interfaz facilita el trabajo de los programadores que trabajan con vídeo, audio e imagen y colabora con la creación de las aplicaciones multimedia en general, pudiendo servir para proyectos más complejos, como videoconferencias y conversores de medios.

#### 2.2.4.5 Node.js

Node.js es una plataforma de código abierto que se basa en el motor JavaScript V8 de Google Chrome y permite ejecutar código JavaScript en el lado del servidor. Se utiliza comúnmente para construir aplicaciones Web en tiempo real, servidores de aplicaciones y otras aplicaciones de red. Node.js es conocido por su enfoque de E/S sin bloqueo y su capacidad para manejar conexiones simultáneas de manera eficiente (Casciaro, 2020).

Node.js ha ganado popularidad en el desarrollo Web y de aplicaciones debido a su capacidad para manejar múltiples solicitudes en paralelo, lo que lo hace especialmente adecuado para aplicaciones en tiempo real y aplicaciones de alta concurrencia.

La referencia mencionada ofrece una guía detallada sobre cómo utilizar Node.js de manera efectiva en proyectos de desarrollo.

Node.js fue creado por los desarrolladores originales de JavaScript. Lo transformaron de algo que solo podía ejecutarse en el navegador en algo que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara. Gracias a Node.js se puede ir un paso más allá en la programación con JavaScript no solo creando sitios Web interactivos.

#### 2.2.4.6 React

React es una biblioteca de JavaScript que se utiliza para construir interfaces de usuario de una manera declarativa y componentizada. Permite a los desarrolladores crear componentes reutilizables que representan partes de la interfaz de usuario y luego componerlos para crear aplicaciones Web complejas. React utiliza un enfoque virtual del DOM (Document Object Model) para optimizar las actualizaciones y mejorar el rendimiento de las aplicaciones (Porcello, 2020).

React ha ganado popularidad en la comunidad de desarrollo Web debido a su enfoque en la construcción de componentes reutilizables y su capacidad para manejar actualizaciones de interfaz de usuario de manera eficiente. Es ampliamente utilizado en el desarrollo de aplicaciones Web, incluidas aplicaciones de una sola página (SPA) y aplicaciones móviles con React Native. La referencia mencionada es una fuente útil para aprender React y sus mejores prácticas.

#### 2.2.4.7 Backend

El término "backend" se utiliza en el desarrollo de software para referirse a la parte de una aplicación que se encarga de la lógica y la funcionalidad detrás de escena, en contraposición al "frontend", que es la parte de la aplicación con la que los usuarios interactúan directamente.

El backend se encarga de gestionar la lógica empresarial, la manipulación de datos, la seguridad, la autenticación y otros aspectos que no son visibles para el usuario, pero son esenciales para el funcionamiento de la aplicación.

#### 2.2.4.8 .NET Core

.NET Core es un marco de desarrollo de código abierto de Microsoft que permite a los desarrolladores crear aplicaciones y servicios multiplataforma utilizando C# y otros lenguajes de programación. Ofrece una amplia gama de bibliotecas y características que facilitan la creación de aplicaciones escalables y de alto rendimiento.

La última versión de .NET core, 2.0, viene con numerosas mejoras. .NET core 2.0 es la versión más rápida de todos los tiempos y puede ejecutar múltiples plataformas incluyendo varias distribuciones de Linux, macOS (sistema operativo) y Windows (Khan, 2018).

#### 2.2.4.9 Base de Datos

Una base de datos es un conjunto organizado y estructurado de datos que se almacena electrónicamente en un sistema de computadoras. Estos datos pueden incluir información de todo tipo, como números de teléfono, registros de clientes, inventarios, transacciones financieras, contenido multimedia, documentos y mucho más. La base de datos está diseñada para permitir el almacenamiento, la recuperación, la gestión y la manipulación eficiente de datos.

Las bases de datos se utilizan en una amplia variedad de aplicaciones y contextos, desde aplicaciones empresariales hasta sistemas de gestión de contenido, sistemas de información geográfica, redes sociales y mucho más. Algunos conceptos clave relacionados con las bases de datos incluyen:

1. Tablas: Las bases de datos suelen organizar los datos en tablas, donde cada fila representa un registro individual y cada columna representa un atributo o campo específico.
2. Sistemas de gestión de bases de datos (DBMS): Para administrar y acceder a los datos de manera eficiente, se utilizan sistemas de gestión de bases de datos, como MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, MongoDB y otros.
3. Consultas: Se pueden realizar consultas en una base de datos para recuperar datos específicos o realizar cálculos en los datos almacenados.
4. Integridad de datos: Las bases de datos suelen implementar restricciones y reglas para garantizar la integridad de los datos, como claves primarias y foráneas, restricciones de unicidad y validaciones.
5. Escalabilidad: Las bases de datos se pueden diseñar para escalar y crecer a medida que aumenta la cantidad de datos y la demanda de acceso.

6. Seguridad: Las bases de datos suelen incluir medidas de seguridad para proteger los datos confidenciales, como la autenticación y la autorización.

Las bases de datos son una parte fundamental en la mayoría de las aplicaciones informáticas y desempeñan un papel esencial en la gestión y el almacenamiento de información. Su diseño y administración adecuados son críticos para garantizar la eficiencia, la integridad y la seguridad de los datos (Navathe, 2000).

**Tabla 1. Base de Datos Utilizadas**

| Motor / Componente | Rol en la solución | Objetos de datos principales (según esquema) | Capacidades técnicas relevantes | Razones de elección |
|---|---|---|---|---|
| **PostgreSQL** | Base operacional transaccional que persiste catálogo, existencias, compras, ventas y logística. | Tablas: roles, usuarios, categorías, unidades_medida, productos, ubicaciones, inventario, proveedores, ordenes_compra, ordenes_compra_detalle, recepciones, recepciones_detalle, clientes, ventas, ventas_detalle, despachos, despachos_detalle, ajustes_inventario, ajustes_inventario_detalle, movimientos_inventario, transferencias, transferencias_detalle, auditoria.<br>Vistas: vista_stock_actual, vista_stock_bajo, vista_productos_proximos_vencer.<br>Triggers/funciones: actualización de fecha_actualizacion y bitácora automática de movimientos_inventario desde inventario.<br>Índices específicos para SKU, | ACID, FK/UNIQUE/CHECK, columnas generadas, JSONB en auditoria, vistas analíticas para KPIs de stock, triggers para auditoría operativa. Preparado para RLS y explotación vía PostgREST. | Integridad y consistencia en operaciones de inventario/ventas; SQL potente para reportes; escalabilidad y gobierno de datos; soporte nativo a RLS y excelente integración con PostgREST. |

| | | | | |
|---|---|---|---|---|
| | | código de barras, fechas, referencias y relaciones. | | |
| **PostgREST** | Capa REST automática sobre PostgreSQL; expone recursos a partir de tablas, vistas y funciones, con control fino de seguridad a nivel DB. | Exposición directa de recursos como /productos, /inventario, /movimientos_inventario, /ordenes_compra, /ventas, y vistas /vista_stock_actual, /vista_stock_bajo, /vista_productos_proximos_vencer. | CRUD inmediato, filtros y ordenamientos estándar, proyección de columnas, embedding, y aprovechamiento de vistas/funciones SQL; seguridad basada en roles PostgreSQL (combinable con JWT emitidos por el gateway). | Reduce tiempo de entrega del API, mantiene consistencia con el modelo relacional y habilita una seguridad centrada en datos (RLS) sin duplicar lógica. |
| **Redis (opcional)** | Caché para consultas frecuentes y soporte a rate limiting / sesiones en el API Gateway (Express). | No persiste dominio; actúa como capa de rendimiento para listados (e.g., catálogos, bajo stock) y reportes agregados; clave-valor con TTLs configurables. | Latencias de milisegundos, expiración e invalidación por clave, soporte a contadores atómicos, y patrones de cache-aside para endpoints de alta demanda. | Mejora desempeño y escalabilidad del front/back sin comprometer la fuente de verdad (PostgreSQL). Ideal para picos de lectura y dashboards. |

### 2.2.5 Metodología para el Desarrollo Web

Una metodología para el desarrollo Web es un conjunto de directrices, prácticas y procesos organizados que guían a los equipos de desarrollo en la creación de aplicaciones y sitios Web. Estas metodologías definen etapas, roles y actividades para asegurar la calidad, la eficiencia y el cumplimiento de los objetivos del proyecto.

A continuación, algunas metodologías comunes para el desarrollo Web:

- Metodología Ágil: Enfoque de desarrollo iterativo y colaborativo que se centra en la entrega rápida de funcionalidad. Scrum y Kanban son ejemplos de metodologías ágiles.
- Modelo en Cascada: Enfoque secuencial donde cada etapa se completa antes de pasar a la siguiente. A menudo se utiliza en proyectos con requisitos bien definidos.
- Metodología DevOps: Enfoque que integra el desarrollo y las operaciones para acelerar la entrega y mejorar la calidad del software.
- Metodología de Desarrollo en Espiral: Enfoque iterativo que incorpora la retroalimentación del cliente y la evaluación de riesgos en cada fase del proyecto.
- Metodología de Desarrollo Rápido de Aplicaciones (RAD): Enfoque que se centra en la entrega rápida de prototipos y versiones funcionales.
- Metodología Lean: Enfoque que busca eliminar el desperdicio y mejorar la eficiencia en el desarrollo Web.

### 2.2.6 UWE (UML-Based Web Engineering)

UWE (UML-Based Web Engineering), que se traduce como "Ingeniería Web Basada en UML", es una metodología de desarrollo Web que se basa en el uso de UML (Unified Modeling Language o Lenguaje de Modelado Unificado) como principal lenguaje de modelado. UWE-UML se utiliza para diseñar y desarrollar aplicaciones Web, permitiendo a los equipos de desarrollo planificar, visualizar y construir sistemas Web de manera estructurada y eficiente.

UWE-UML es una metodología de desarrollo Web que utiliza UML como un enfoque de modelado para definir la estructura, la funcionalidad y la interacción de una aplicación Web. Proporciona un conjunto de diagramas y artefactos que representan aspectos clave de una aplicación Web, como la navegación, la interacción del usuario, la estructura de la base de datos y la lógica del negocio.

UWE-UML es una metodología que ayuda a los desarrolladores Web a modelar y planificar proyectos de desarrollo Web de manera más efectiva. Al utilizar UML como lenguaje de modelado, los equipos pueden visualizar y comunicar de manera más eficiente los aspectos complejos de una aplicación Web, lo que puede mejorar la calidad y la eficiencia del desarrollo.

### 2.2.7 Cálculo de Costos en Ingeniería de Software

El cálculo de costos en ingeniería de software se refiere al proceso de estimar y determinar los costos asociados con el desarrollo de un proyecto de software. Esto implica calcular los gastos relacionados con recursos humanos, hardware, software, infraestructura, licencias, capacitación y otros aspectos necesarios para llevar a cabo el proyecto de manera exitosa. El objetivo principal del cálculo de costos es proporcionar una estimación precisa del presupuesto necesario para completar el proyecto de software de acuerdo con los requisitos especificados.

El cálculo de costos en ingeniería de software implica considerar diversos factores, como el tamaño y la complejidad del proyecto, la experiencia del equipo de desarrollo, el costo de los recursos humanos, el costo de las herramientas y tecnologías a utilizar, y otros gastos relacionados con el ciclo de vida del proyecto (McConnell, 2006).

### 2.2.8 COCOMO

COCOMO, que significa "Constructive Cost Model" o "Modelo de Costos Constructivos", es un modelo de estimación de costos ampliamente utilizado en la ingeniería de software para calcular los esfuerzos, los costos y los plazos necesarios para desarrollar proyectos de software. Fue desarrollado por Barry Boehm en la década de 1980 y se ha convertido

en un estándar de la industria para la estimación de costos de proyectos de software (Boehm, 1981).

COCOMO es un modelo de estimación de costos de software que utiliza una serie de ecuaciones y factores para predecir los recursos y el tiempo necesarios para completar un proyecto de desarrollo de software. El modelo se basa en diferentes niveles de detalle, como COCOMO I (básico), COCOMO II (intermedio) y COCOMO III (avanzado), que se adaptan a diferentes tipos de proyectos y niveles de complejidad.

**Tabla 2 Operacionalización de Variables**

| Objetivo | Variable | Tipo | Definición conceptual | Dimensión operacional | Técnicas e instrumentos |
|---|---|---|---|---|---|
| **Diseñar un sistema que optimice el control de inventario y ventas** | **Eficiencia operativa** | Dependiente | Grado en que el sistema reduce el tiempo y esfuerzo en procesos de inventario y ventas | Tiempo promedio para registrar movimientos; tiempo de cierre diario | Cronometría antes/después; observación directa |
| **Determinar requerimientos funcionales y no funcionales** | **Calidad de inventario** | Dependiente | Exactitud y consistencia entre el inventario físico y el registrado en el sistema | Porcentaje e diferencias en conteos físicos vs. sistema; porcentaje de quiebres de stock | Auditorías periódicas; revisión documental |

| | | | | | |
|---|---|---|---|---|---|
| **Proponer un diseño que incorpore automatización** | **Automatización** | Independiente | Nivel de sustitución de procesos manuales por procesos digitales | % de procesos automatizados; número de integraciones con otros sistemas | Análisis de flujos; entrevistas con usuarios |
| **Evaluar beneficios potenciales del sistema** | **Usabilidad** | Independiente | Facilidad de uso y satisfacción percibida por los usuarios | Puntuación SUS (System Usability Scale); nivel de errores en tareas | Encuestas SUS; pruebas de usabilidad |
| **Garantizar trazabilidad y seguridad** | **Trazabilidad** | Independiente | Capacidad del sistema para registrar y auditar todas las operaciones | % de movimientos con responsable y evidencia; existencia de bitácora | Revisión de logs; análisis de auditoría |

# 3. Diseño Metodológico

## 3.1. Enfoque cualitativo asumido y su justificación

El estudio adoptó un enfoque cualitativo–aplicado, orientado a comprender en profundidad los procesos actuales de gestión de inventario en Ibagu Store y proponer un diseño que respondiera a sus necesidades específicas. Este enfoque se justificó porque el objetivo principal no fue la medición estadística, sino la identificación y análisis detallado de los flujos de trabajo, puntos críticos y requerimientos funcionales y no funcionales que permitieran estructurar una solución tecnológica viable.

Para la recolección de información se emplearon entrevistas semiestructuradas con los actores clave (gerencia, encargado de almacén y personal de ventas), con el fin de obtener percepciones sobre las limitaciones del sistema actual y expectativas del nuevo diseño. Asimismo, se realizó observación directa de los procesos operativos (recepción de mercancías, registro de ventas, ajustes de inventario) para documentar tiempos, secuencias y riesgos asociados.

Complementariamente, se llevó a cabo un análisis documental de los formatos y registros utilizados (hojas de cálculo, comprobantes físicos), lo que permitió identificar inconsistencias y oportunidades de mejora. Esta combinación de técnicas cualitativas aseguró una comprensión integral del contexto, facilitando la elaboración de un diseño alineado con la realidad operativa y adaptable a la futura implementación bajo el stack MERN con PostgreSQL/PostgREST.

## 3.2. Muestra teórica y sujetos del estudio

La muestra teórica seleccionada para esta investigación está compuesta por los actores clave que intervienen en los procesos de gestión de inventario, ventas y control administrativo en Ibagu Store. La elección de estos sujetos responde a la necesidad de

obtener información cualitativa relevante que permita comprender las limitaciones del sistema actual y validar los requerimientos del diseño propuesto.

Composición de la muestra:

**Gerente General (1):** Responsable de la toma de decisiones estratégicas, control financiero y supervisión global de la operación. Su participación es esencial para definir los indicadores de gestión y las necesidades de reportes ejecutivos.

**Encargado de Almacén (1):** Actor principal en la recepción de mercancías, control de existencias y ejecución de ajustes de inventario. Su experiencia permite identificar los puntos críticos en la trazabilidad y control físico del stock.

**Personal de Ventas/Caja (2–3):** Encargados del registro de ventas y devoluciones, así como de la interacción directa con el cliente. Su aporte es fundamental para evaluar la usabilidad del sistema y la integración con el flujo de ventas.

La selección de esta muestra se justifica porque estos roles concentran el conocimiento operativo y estratégico necesario para garantizar que el diseño del sistema responda a las necesidades reales de la empresa.

### 3.3. Métodos y técnicas de recolección de datos utilizados

Para la identificación de requerimientos y el análisis del flujo actual de trabajo en Ibagu Store, se aplicaron técnicas cualitativas que permitieron obtener información profunda y contextual:

Entrevistas semiestructuradas: Se realizaron con los actores clave (gerencia, encargado de almacén y personal de ventas) para comprender las limitaciones del sistema actual, expectativas sobre la solución y necesidades específicas. Estas entrevistas incluyeron preguntas abiertas que permitieron explorar problemas como errores en el control de

stock, tiempos de registro y dificultades en la generación de reportes, lo que facilitó la priorización de funcionalidades críticas.

Observación directa en sitio: Se llevó a cabo un seguimiento detallado de los procesos críticos, como la recepción de mercancías, el registro de ventas y los ajustes de inventario. Esta técnica permitió documentar tiempos promedio, secuencia de actividades, puntos de congestión y riesgos operativos, información que fue clave para optimizar los flujos en el diseño del sistema.

Revisión documental: Se analizaron los formatos y registros utilizados actualmente (Kardex, hojas de cálculo, comprobantes físicos), identificando inconsistencias, duplicidades y carencias en la trazabilidad de la información. Estos hallazgos sirvieron como base para definir los requerimientos funcionales y no funcionales del sistema propuesto.

**Tabla 3 Principales Teorías y Aportes al Tema de Investigación.**

| Teoría / Marco | Aporte al diseño |
|---|---|
| **Gestión de Inventarios (ABC, punto de reorden)** | Priorización de productos y definición de niveles mínimos de stock. |
| **Sistemas de Información Gerencial (MIS)** | Generación de reportes estratégicos para la toma de decisiones. |
| **Ingeniería de Software Ágil (Scrum)** | Base para un desarrollo iterativo y modular en fases futuras. |
| **Seguridad de la Información (CIA, RBAC, JWT, RLS)** | Protección de datos, control de accesos y trazabilidad. |
| **Arquitectura RESTful y SOA** | Interoperabilidad, escalabilidad y facilidad de integración. |

Fuente:

## Discusión de Resultados o Hallazgos

El desarrollo del presente capítulo se fundamenta en los datos recolectados mediante el enfoque cualitativo–aplicado, utilizando las técnicas de entrevistas semiestructuradas, observación directa y análisis documental. Los sujetos clave, incluyendo la gerencia, el encargado de almacén y el personal de ventas, proporcionaron la información primaria.

El análisis de estos datos permite presentar los hallazgos en tres fases lógicas: el diagnóstico de la situación actual, la sistematización de estos hallazgos en requerimientos técnicos y, finalmente, la propuesta de diseño y viabilidad como respuesta a dichos requerimientos.

## Diagnóstico y Carencias del Proceso Manual

El trabajo de campo (observación directa y entrevistas) permitió validar la problemática central: la gestión actual de Ibagu Store se basa exclusivamente en procesos manuales, específicamente en hojas de cálculo. Este hallazgo es la respuesta directa a las primeras tres preguntas de investigación.

El control actual no cumple con requerimientos técnicos básicos. El uso de Excel carece de mecanismos de concurrencia, validación de datos robusta y normalización, generando duplicidad de datos.

El sistema manual impide la trazabilidad. Durante la observación del proceso, se evidenció que no existe un registro auditable de quién, cuándo y por qué se modifica una existencia. Esto genera riesgos de control poco fiable del stock.

La seguridad de los datos es vulnerable. Los archivos de hojas de cálculo carecen de controles de acceso granulares (RBAC) o protección a nivel de fila (RLS), exponiendo información sensible de costos y ventas a personal no autorizado.

Ineficiencia operativa. Las entrevistas con el personal revelaron que el registro manual consume tiempo significativo y los retrasos en la actualización impiden tomar decisiones en tiempo real, generando pérdida de oportunidades comerciales.

**Sistematización de Requerimientos**

Como resultado directo del diagnóstico, la investigación procedió a sistematizar estas carencias y las expectativas de los usuarios en un conjunto de requerimientos formales. Este proceso responde al primer objetivo específico: "Determinar los requerimientos funcionales y no funcionales...".

Los hallazgos cualitativos se tradujeron en las siguientes necesidades críticas, que su diseño debe solventar:

Necesidad de Trazabilidad: La falta de auditoría llevó a definir el requerimiento de un módulo de existencias con bitácora detallada por usuario.

Necesidad de Control de Gastos: Las entrevistas gerenciales destacaron la dificultad para vincular compras con costos, lo que fundamenta el requerimiento del módulo de proveedores y órdenes de compra.

Necesidad de Integridad de Datos: Los errores observados en el registro manual exigen la automatización de las salidas de inventario al momento de la venta, eliminando la doble digitación.

Necesidad de Información Estratégica: La gerencia reportó la incapacidad de generar reportes fiables, lo que se traduce en el requerimiento de un módulo de reportes avanzados (bajo stock, análisis ABC, rotación).

Necesidad de Seguridad: La vulnerabilidad de los archivos planos fundamenta los requerimientos no funcionales de autenticación (JWT) y control de acceso (RBAC).

(Nota del Catedrático: En este punto, usted debe insertar sus secciones ya existentes "Requerimientos funcionales" y "Requerimientos no funcionales", ya que ahora quedan justificadas como el resultado del análisis de los hallazgos de campo).

**Diseño Técnico y Viabilidad**

El Diseño Técnico El diseño propuesto no es arbitrario, sino una consecuencia directa de los hallazgos. La elección del stack MERN adaptado con PostgreSQL/PostgREST responde a la necesidad de una arquitectura moderna y escalable. Específicamente:

PostgreSQL y PostgREST se seleccionaron por su robustez transaccional (vital para un inventario) y la capacidad de aplicar seguridad a nivel de base de datos (RLS), resolviendo directamente el hallazgo de vulnerabilidad.

El API Gateway (Express) centraliza la seguridad (JWT, RBAC), respondiendo a la necesidad de accesos controlados.

React con Tailwind CSS se eligió para garantizar una alta usabilidad y diseño responsivo, facilitando la adopción por parte del personal de ventas y almacén.

**4.1. Diseño técnico propuesto (MERN + PostgreSQL/PostgREST):**

Como resultado del análisis de requerimientos funcionales y no funcionales, se definió una arquitectura basada en el stack MERN (MongoDB, Express, React, Node) adaptado para trabajar con PostgreSQL y PostgREST, con el objetivo de garantizar escalabilidad, seguridad y facilidad de mantenimiento. Esta decisión responde a la necesidad de integrar tecnologías modernas, ampliamente soportadas y con un ecosistema robusto para el desarrollo de aplicaciones empresariales.

### 4.1.1 Frontend

Se optó por React 18 con TypeScript, utilizando Vite como herramienta de construcción para optimizar el rendimiento y los tiempos de compilación. La interfaz incorpora React Router para la gestión de rutas, React Hook Form para validación eficiente de formularios y Tailwind CSS para un diseño responsivo y consistente. Esta combinación asegura una experiencia de usuario fluida, adaptable a diferentes dispositivos y con validaciones en tiempo real.

### 4.1.2 API Gateway

En la capa intermedia, se implementó un API Gateway con Node.js y Express, encargado de la orquestación de casos compuestos y la aplicación de políticas de seguridad. Se integraron mecanismos de autenticación mediante JWT, control de acceso basado en roles (RBAC) y rate limiting para mitigar ataques de denegación de servicio. Esta capa también centraliza la lógica de negocio y facilita la integración con servicios externos.

### 4.1.3 Gestión de Datos

Para la persistencia, se utilizó PostgreSQL como base de datos relacional, complementada con PostgREST para exponer recursos RESTful de manera automática a partir de vistas y funciones definidas en SQL. Se aplicaron políticas de Row-Level Security (RLS) para garantizar la protección granular de los datos según el rol y el usuario autenticado. Adicionalmente, se consideró el uso opcional de Redis como capa de caché para optimizar el rendimiento en consultas frecuentes y reducir la carga sobre la base de datos.

### 4.1.4 Despliegue y Escalabilidad

El diseño contempla la contenedorización mediante Docker, lo que permite entornos reproducibles y facilita la integración continua (CI/CD) para automatizar pruebas y despliegues. Se estableció la gestión de la base de datos en entornos administrados, junto con migraciones versionadas para mantener la integridad del esquema. Para

consultas analíticas y reportes pesados, se definió el uso de vistas materializadas, optimizando el tiempo de respuesta sin comprometer la consistencia de los datos.

Este diseño técnico no solo responde a los requerimientos identificados, sino que también incorpora principios de arquitectura moderna, asegurando que la solución sea **segura**, escalable y mantenible, con capacidad para evolucionar conforme crezcan las necesidades del negocio.

Se integra el estudio financiero provisto por el usuario con los siguientes resultados clave:

## 5. Conclusiones

El desarrollo del presente proyecto permitió alcanzar los objetivos planteados, logrando un diseño integral para la gestión de inventarios en Ibagu Store que responde a las necesidades identificadas durante el análisis. A partir de la investigación cualitativa y la validación técnica, se obtuvieron las siguientes conclusiones principales:

### Cobertura funcional completa y alineada con las necesidades del negocio

El diseño propuesto abarca los procesos críticos de la operación: catálogo de productos, control de existencias, registro de movimientos (entradas, salidas y ajustes), gestión de compras y proveedores, ventas, devoluciones, gastos y reportería avanzada. Esta cobertura garantiza la trazabilidad completa de las transacciones, reduciendo riesgos de pérdida de información y errores en el control de stock.

### Arquitectura técnica robusta y segura

La combinación de Express como API Gateway y PostgREST para la exposición de datos permitió mantener buenas prácticas de desarrollo, incorporando autenticación mediante JWT, control de acceso basado en roles (RBAC), documentación estandarizada con OpenAPI y despliegue automatizado mediante Docker y CI/CD. Además, la implementación de Row-Level Security (RLS) en PostgreSQL refuerza la seguridad a nivel de fila, asegurando la protección granular de la información.

**Mejoras proyectadas en eficiencia y toma de decisiones**

Se espera una reducción significativa en los tiempos de registro y en la ocurrencia de errores asociados a la gestión manual, lo que impactará positivamente en la productividad del personal y en la confiabilidad de los datos. Asimismo, la incorporación de reportes estratégicos (bajo stock, análisis ABC, rotación de productos, márgenes y gastos) proporcionará indicadores clave para la toma de decisiones, fortaleciendo la planificación de compras y la gestión financiera.

La implementación de este sistema representa un avance significativo en la transformación digital de Ibagu Store, contribuyendo a la competitividad en un mercado donde muchas PYMES aún dependen de procesos manuales. Este cambio no solo optimiza la operación interna, sino que también se alinea con las políticas nacionales que promueven la digitalización y el uso de Tecnologías de la Información y la Comunicación (TIC) como motor de desarrollo económico.

**Incremento de la competitividad y sostenibilidad**

La digitalización permite a la empresa adaptarse a la economía digital y a la Industria reduciendo la brecha tecnológica frente a competidores y mejorando su capacidad para responder a cambios del mercado. Esto es coherente con la Ley General de Telecomunicaciones Convergentes (Legislacion De Nicaragua, 2024), que reconoce la transformación digital como un factor transversal para el desarrollo económico y social del país, impulsando la productividad y la innovación.

**Acceso a programas de apoyo y financiamiento**

Las reformas recientes a la Ley 290 fortalecen el papel del MIFIC en la promoción de las MIPYMES, incluyendo políticas para mejorar el acceso a tecnología, financiamiento y capacitación empresarial. Esto significa que empresas que adopten soluciones digitales pueden beneficiarse de incentivos y programas de apoyo gubernamental (El diario Nica, 2025).

**Optimización del capital de trabajo y reducción de costos**

Al mejorar la precisión en la gestión de inventarios, la empresa reduce pérdidas por quiebres de stock, evita sobre inventarios y disminuye costos operativos asociados a errores manuales. Esto se traduce en una mayor liquidez y capacidad para reinvertir en crecimiento.

**Mejora en la experiencia del cliente y fidelización**

La disponibilidad de información en tiempo real sobre existencias y precios permite ofrecer un servicio más ágil y confiable, lo que incrementa la satisfacción del cliente y fortalece la reputación de la marca.

**Cumplimiento normativo y transparencia**

La digitalización facilita la trazabilidad de operaciones y la generación de reportes, contribuyendo al cumplimiento de normativas fiscales y comerciales, además de mejorar la transparencia en la gestión interna.

**Inclusión en la economía digital y acceso a nuevos mercados**

La adopción de herramientas digitales abre la puerta a estrategias de comercio electrónico y marketing digital, ampliando el alcance del negocio más allá del mercado local. Iniciativas como el Plan Nacional de Transformación Digital y programas como Conectadas impulsan la capacitación y el uso de plataformas digitales para PYMES en Nicaragua.

**Reducción de riesgos y resiliencia empresarial**

La automatización y el respaldo digital de la información reducen la vulnerabilidad ante pérdidas de datos, errores humanos y contingencias operativas, fortaleciendo la continuidad del negocio.

**Escalabilidad y sostenibilidad tecnológica**

El diseño basado en tecnologías modernas (React, Node.js, PostgreSQL) y principios de arquitectura modular asegura la posibilidad de incorporar nuevas funcionalidades, integraciones con sistemas externos y mejoras de rendimiento sin comprometer la estabilidad del sistema. Esto garantiza que la solución pueda evolucionar junto con el crecimiento del negocio.

En síntesis, el proyecto no solo responde a las necesidades actuales de Ibagu Store, sino que también establece una base tecnológica sólida para su transformación digital, contribuyendo a la eficiencia operativa, la reducción de riesgos y la generación de valor estratégico a largo plazo.

## 6. Recomendaciones

Con base en los hallazgos obtenidos y el diseño técnico propuesto, se plantean las siguientes recomendaciones para asegurar una implementación exitosa y sostenible del sistema:

1. Implementar un MVP (Producto Mínimo Viable) enfocado en procesos críticos. Se recomienda iniciar con un módulo funcional que cubra catálogo de productos, movimientos de inventario y reportes de bajo stock, dado que estos procesos representan el núcleo operativo y aportan el mayor impacto en la reducción de errores y quiebres de stock. Esta estrategia permite validar la solución en un entorno controlado antes de escalar a funcionalidades más complejas.

2. Definir políticas claras de gestión de datos. Es fundamental establecer lineamientos para garantizar la integridad y consistencia de la información, incluyendo:

- Asignación de SKU único para cada producto. Validaciones obligatorias en el registro de datos (campos requeridos, formatos). Auditoría de movimientos mediante bitácoras por usuario y motivo.
- Ejecución periódica de respaldos y pruebas de restauración para asegurar la continuidad operativa.

- Capacitación diferenciada por rol y pilotaje controlado. Se sugiere realizar capacitaciones breves y específicas para cada perfil (administrador, gerente, almacén, ventas), enfocadas en las funcionalidades que utilizarán. Posteriormente, implementar un plan piloto con métricas comparativas antes y después (tiempos de registro, exactitud del stock, reducción de errores), lo que permitirá medir el impacto real y ajustar la solución antes del despliegue completo.
- Optimización para reportería y cierres diarios. Para garantizar un desempeño adecuado en consultas analíticas, se recomienda planificar la creación de vistas materializadas e índices en PostgreSQL, especialmente para reportes de rotación, análisis ABC y cierres diarios. Esta práctica reducirá la carga sobre la base de datos y mejorará la experiencia del usuario en la generación de reportes.
- Monitoreo y mejora continua. Una vez implementado el sistema, se debe establecer un esquema de monitoreo que incluya métricas de uso, rendimiento y seguridad, así como un proceso de retroalimentación con los usuarios para identificar oportunidades de mejora. Esto permitirá mantener la solución alineada con las necesidades del negocio y garantizar su sostenibilidad a largo plazo.

## 7. Referencias


C., L. K. (2016). *SIstemas de Informacion Gerencial.* Managing the Digital Firm. Pearson.

C., L. K. (2019). *Management Information Systems.* Pearson Education.

Duckett, j. (2011). *HTML adn CSS: Design and Build Websites .* Jonh Wiley & Sons.

El diario Nica. (2025). Asamblea Nacional de Nicaragua. (2025). Reformas a la Ley N°. 290 para el fortalecimiento de las MIPYMES. *El diario Nica*.

Khan, O. M. (2018). *C# and .Net Core 2.0 High Performance.* Packt Publishing.

Legislacion De Nicaragua. (2024). *Legislacion de Nicaragua*. Retrieved from Legislacion de Nicaragua: http://legislacion.asamblea.gob.ni/normaweb.nsf/($All)/B36FFC254BBDD5DE06258692006DA0F3

M., S. R. (2016). *Fundamentos de Sistemas de Informacion.*

McConnell, S. (2006). *Software Estimation: Demystifying The Black Art.* Pearson.

Meyer, E. A. (2006). *CSS: The Definitive Guide.* O'Reilly Media.

Navathe, R. E. (2000). *Fundamentals of Database Systems.* Pearson Education.

Porcello, A. B. (2020). *Learning React.* O'Reilly Media.

## 8. Glosario de términos técnicos

- API REST: Estilo de arquitectura de software que define un conjunto de restricciones para la creación de servicios web. Permite la comunicación entre el cliente (Frontend) y el servidor (Backend) mediante peticiones HTTP estándar.

- Backend: Capa de acceso a datos y lógica del negocio de una aplicación web. En este proyecto, se refiere a la infraestructura que procesa las solicitudes, gestiona la seguridad y conecta con la base de datos PostgreSQL.

- CAPEX (Capital Expenditures): Gastos de Capital. Se refiere a la inversión inicial necesaria para adquirir, mejorar o mantener activos físicos o intangibles, como el desarrollo del software propuesto.

- CI/CD (Continuous Integration / Continuous Deployment): Práctica de ingeniería de software que permite integrar cambios en el código y desplegar nuevas versiones de la aplicación de manera automática y frecuente, reduciendo errores humanos.

- COCOMO II (Constructive Cost Model): Modelo algorítmico utilizado para estimar el esfuerzo, costo y tiempo de desarrollo de software, basándose en el tamaño del código y factores de complejidad del proyecto.

- Docker: Plataforma de software que permite crear, probar e implementar aplicaciones rápidamente mediante "contenedores". Estos contenedores empaquetan el software con todas sus dependencias para que funcione igual en cualquier entorno.

- Frontend: La parte de la aplicación web con la que interactúa el usuario final. En este proyecto, está desarrollada utilizando la librería React y Tailwind CSS.

- JWT (JSON Web Token): Estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto JSON. Se utiliza en este sistema para la autenticación de usuarios.

- MERN Stack (Adaptado): Acrónimo de las tecnologías MongoDB, Express, React y Node.js. En esta tesis, se adapta sustituyendo MongoDB por PostgreSQL para aprovechar la robustez de las bases de datos relacionales.

- MVP (Producto Mínimo Viable): Versión de un producto con las características suficientes para satisfacer a los clientes iniciales y proporcionar retroalimentación para el desarrollo futuro.

- OpEx (Operating Expenses): Gastos Operativos. Son los costos continuos necesarios para el funcionamiento del sistema, como el pago mensual de servicios en la nube (hosting) y mantenimiento.

- PostgREST: Servidor web independiente que convierte una base de datos PostgreSQL directamente en una API RESTful. Permite exponer los datos de manera segura y rápida sin escribir código repetitivo de backend.

- PostgreSQL: Sistema de gestión de bases de datos relacional de objetos (ORDBMS) de código abierto, conocido por su fiabilidad, robustez y rendimiento. Es el motor de base de datos principal de la propuesta.

- RBAC (Role-Based Access Control): Control de Acceso Basado en Roles. Mecanismo de seguridad que restringe el acceso al sistema a los usuarios autorizados según su función (ej. Administrador, Vendedor, Bodeguero).

- React: Biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario interactivas y componentes reutilizables en el Frontend.

- RLS (Row-Level Security): Seguridad a Nivel de Fila. Característica de PostgreSQL que permite restringir qué filas de una tabla de base de datos pueden ser vistas o modificadas por un usuario específico, aumentando la seguridad de los datos.

- SKU (Stock Keeping Unit): Código único de referencia utilizado para identificar y rastrear cada producto distinto en el inventario.

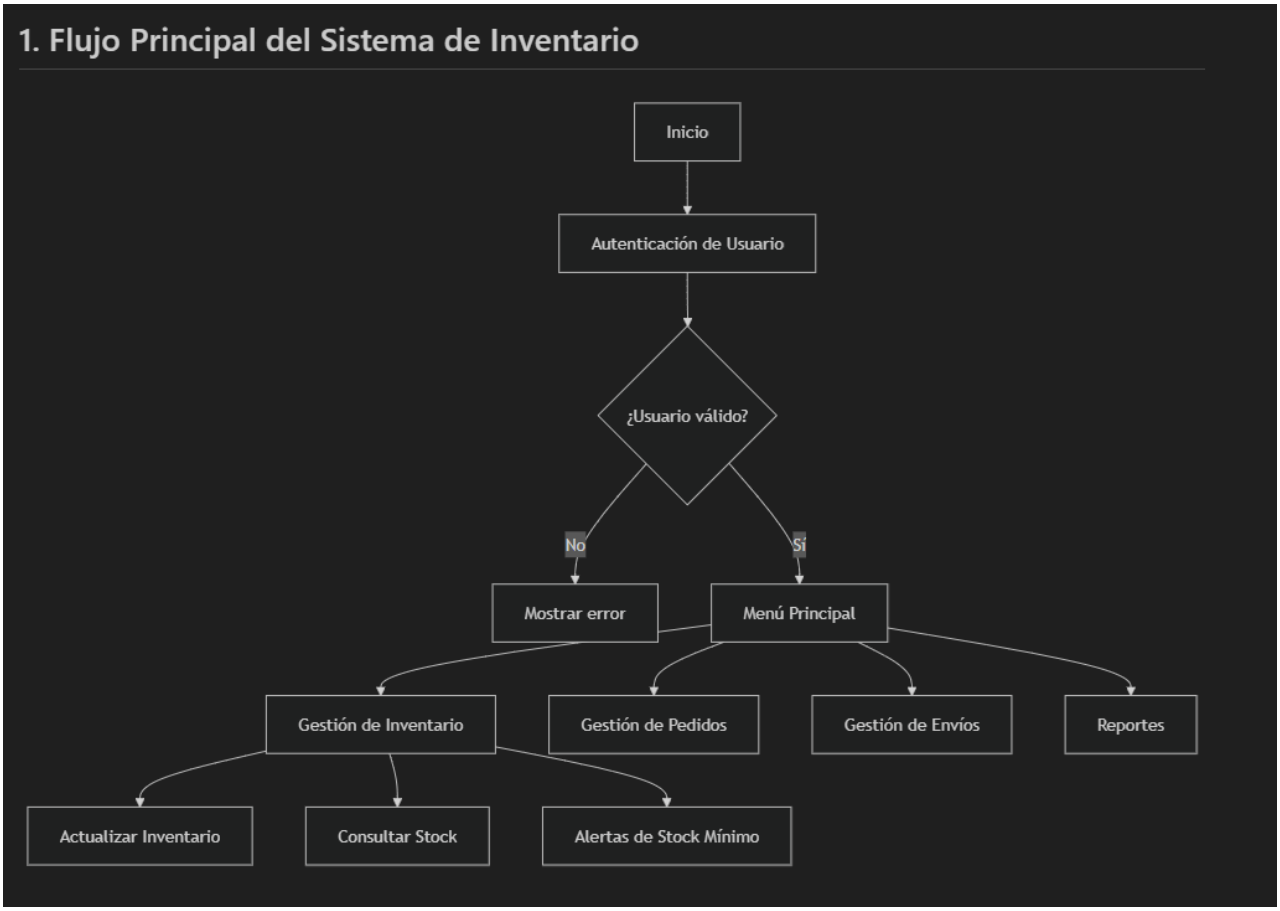
- TIR (Tasa Interna de Retorno): Indicador financiero que mide la rentabilidad de una inversión. Representa la tasa de interés con la cual el Valor Actual Neto (VAN) se hace cero.

- UML (Unified Modeling Language): Lenguaje estándar visual para especificar, construir y documentar los artefactos de los sistemas de software. Se utilizó para diseñar los diagramas de flujo y estructura del sistema.

- VAN (Valor Actual Neto): Indicador financiero que calcula el valor presente de los flujos de caja futuros originados por una inversión. Si el VAN es positivo, el proyecto crea valor y se considera viable.

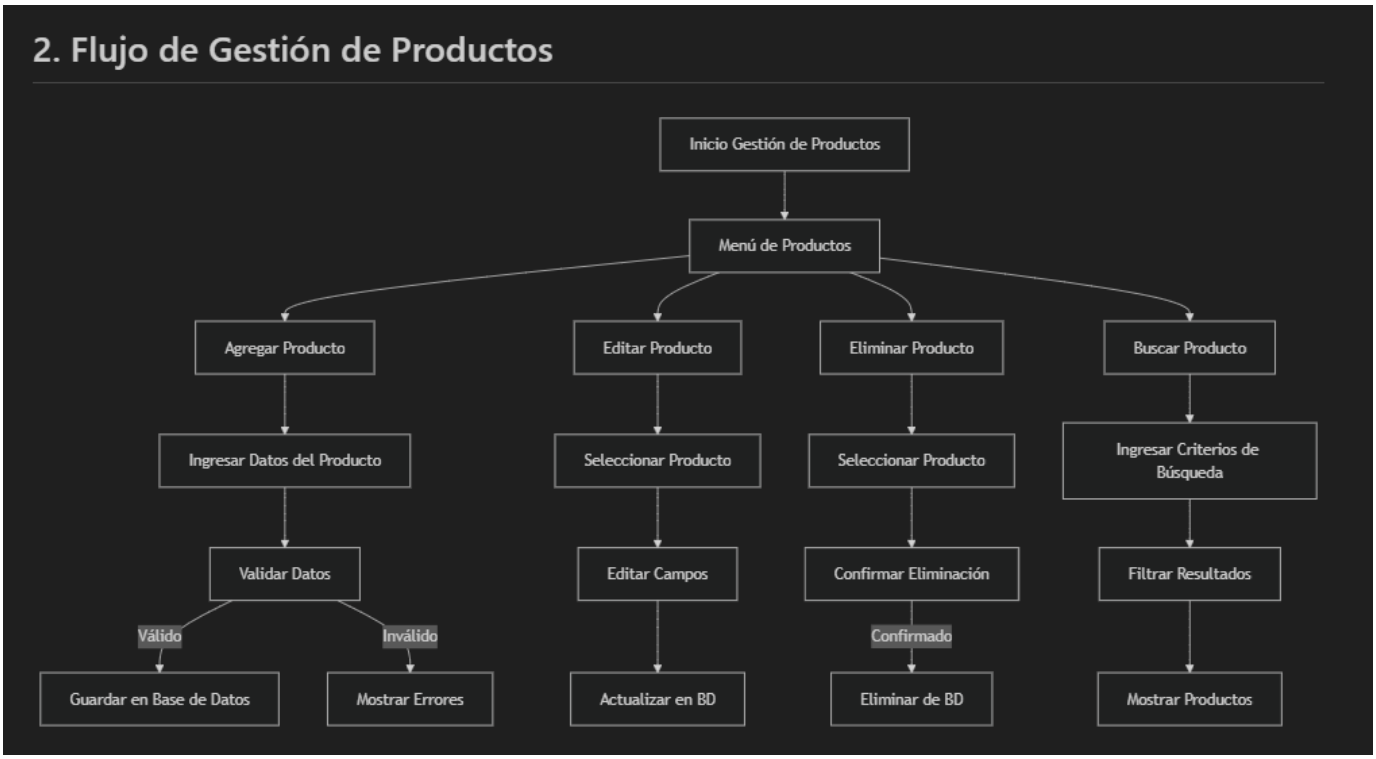
## 9. Anexos o Apéndices

**Figura 1. Flujo principal del Sistema de inventario.**



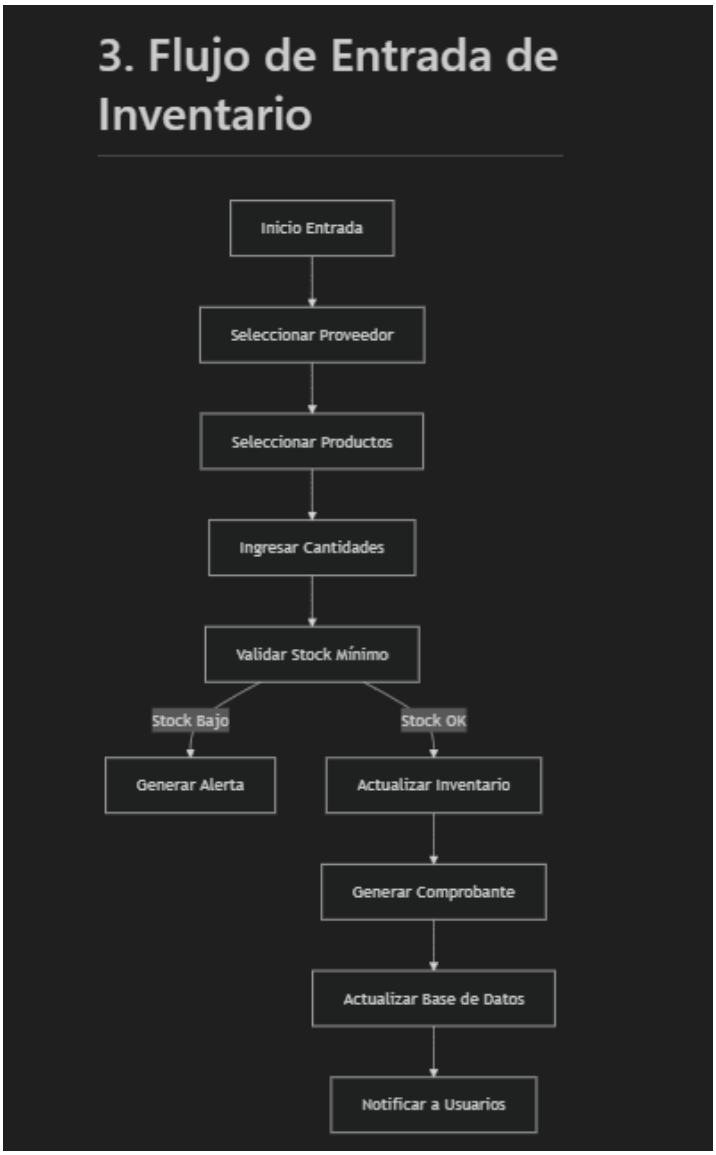

Fuente: elaboración Propia.

**Figura 2. Flujo de gestión de Productos.**



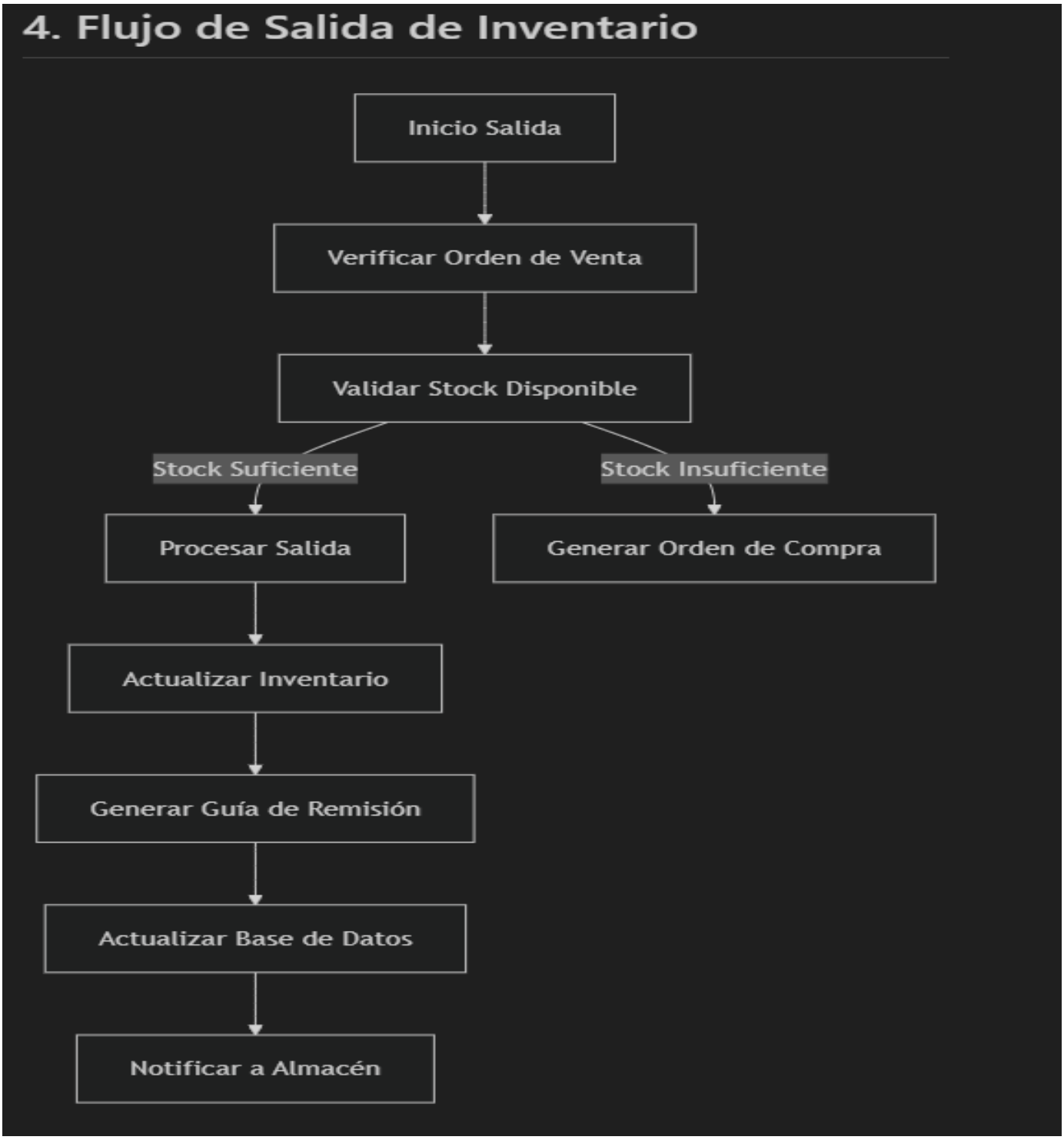

Fuente: elaboración Propia.

**Figura 3 Flujo de Entrada de inventario.**



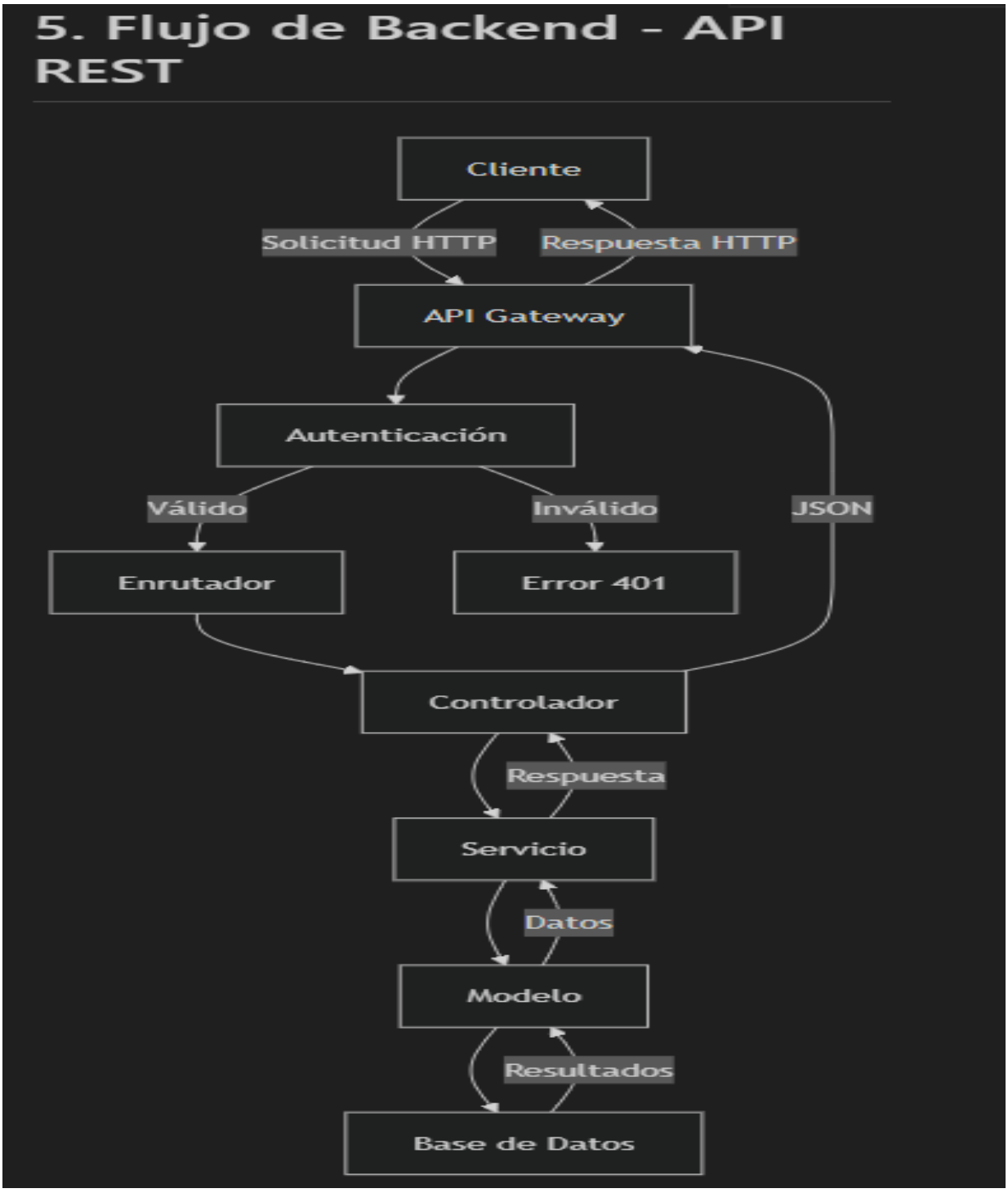

Fuente: elaboración Propia.

**Figura 4. Flujo de salida de Inventario**



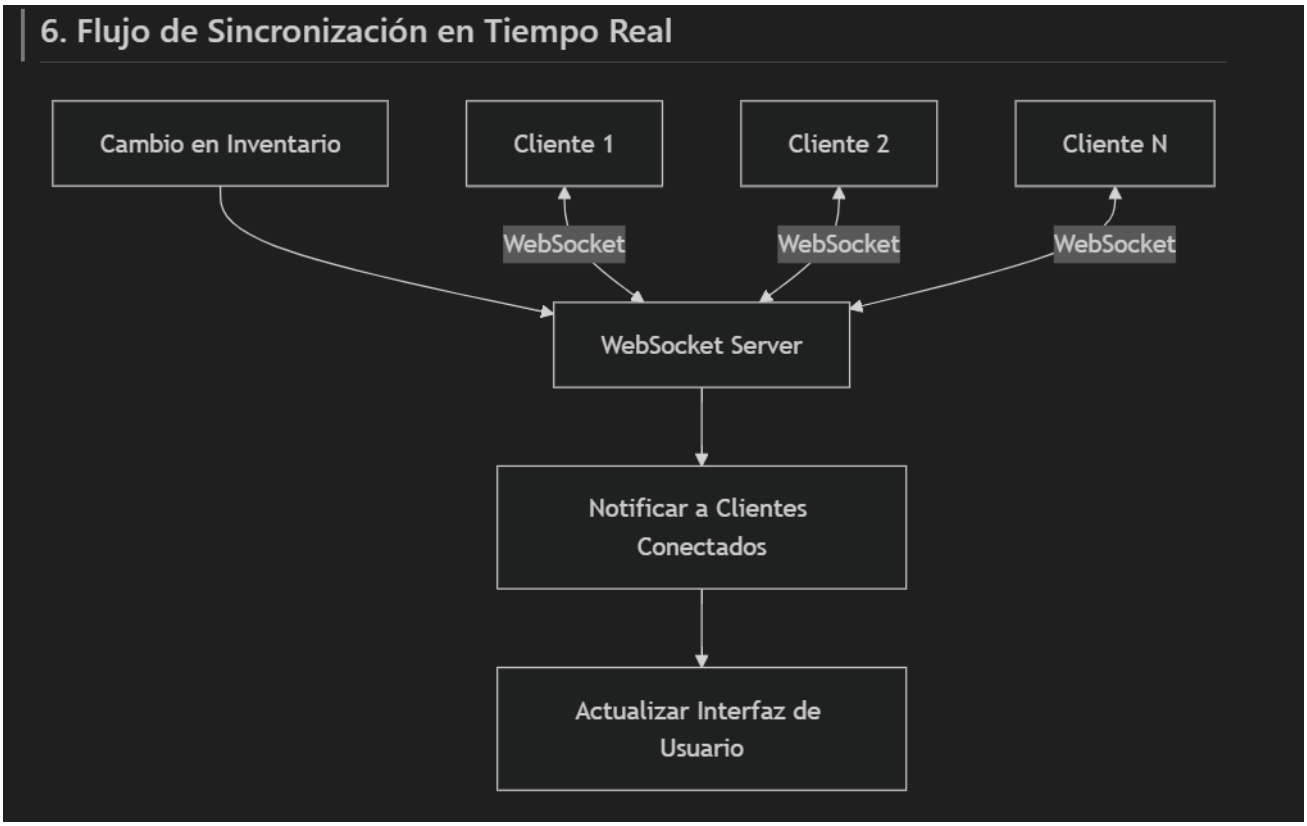

Fuente: elaboración Propia.

**Figura 5. Flujo de Backend – API REST**



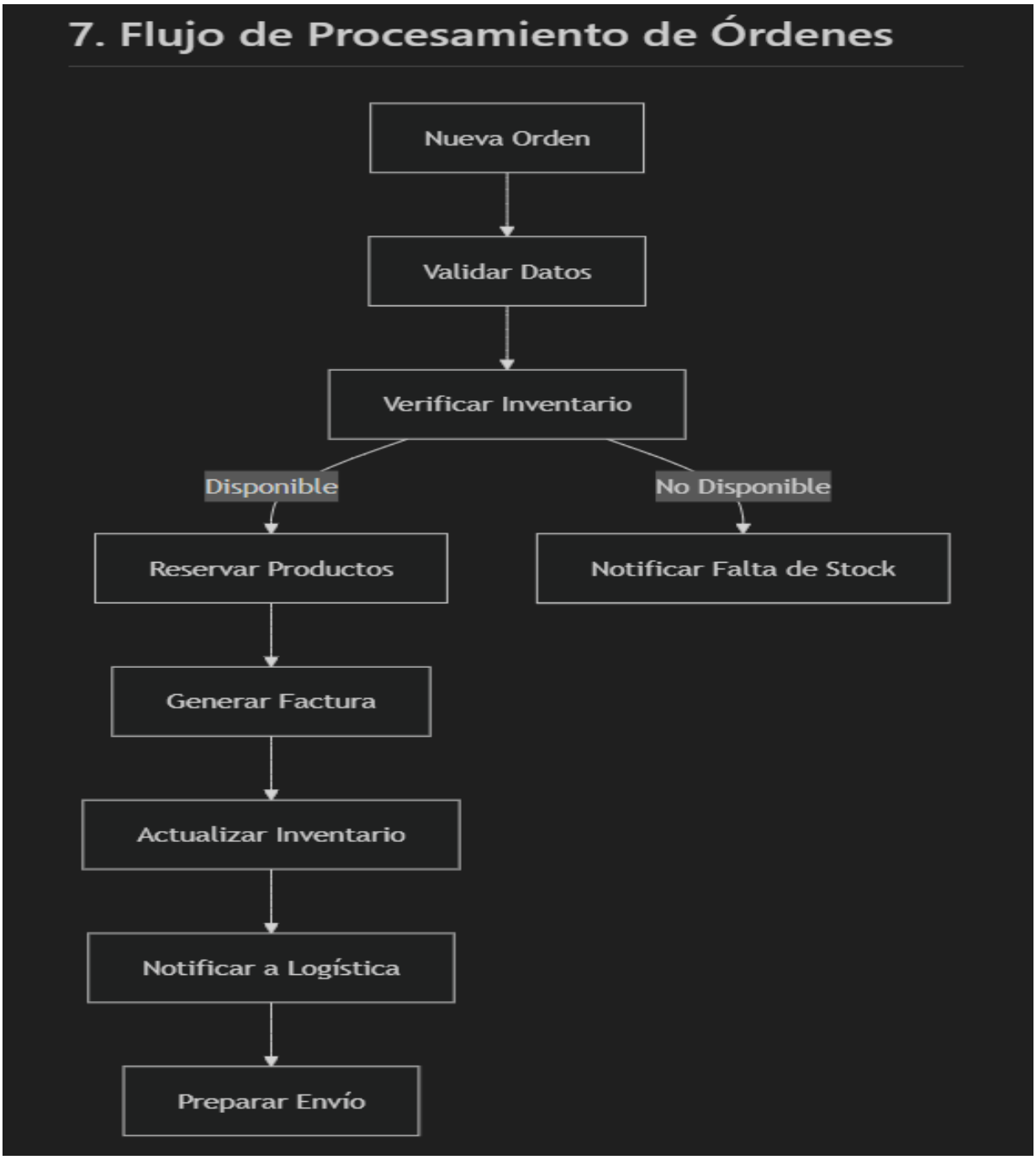

Fuente: elaboración Propia.

**Figura 6. Flujo de sincronización en tiempo real.**



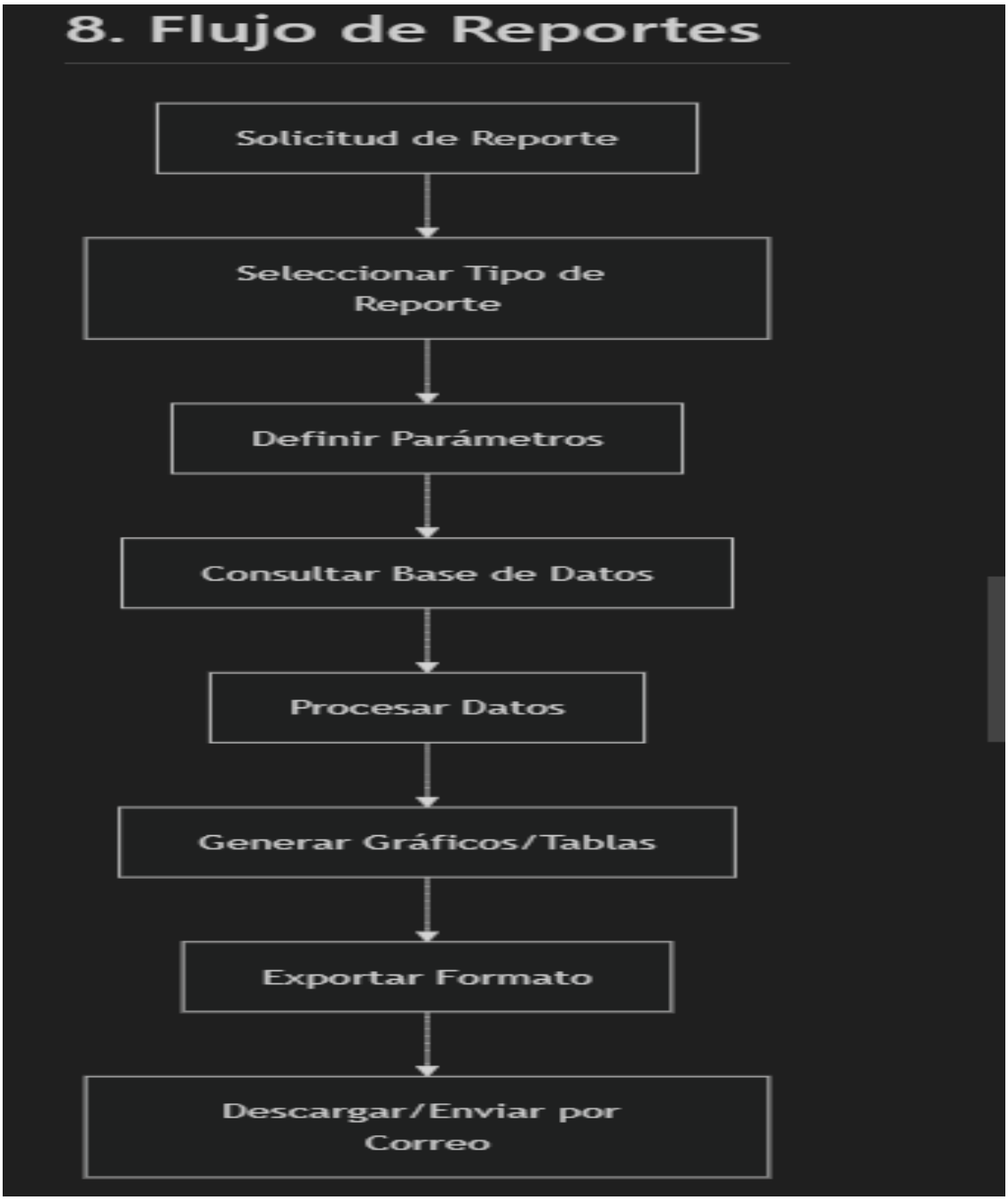

Fuente: elaboración Propia.

**Figura 7. Flujo de procesamiento de órdenes.**



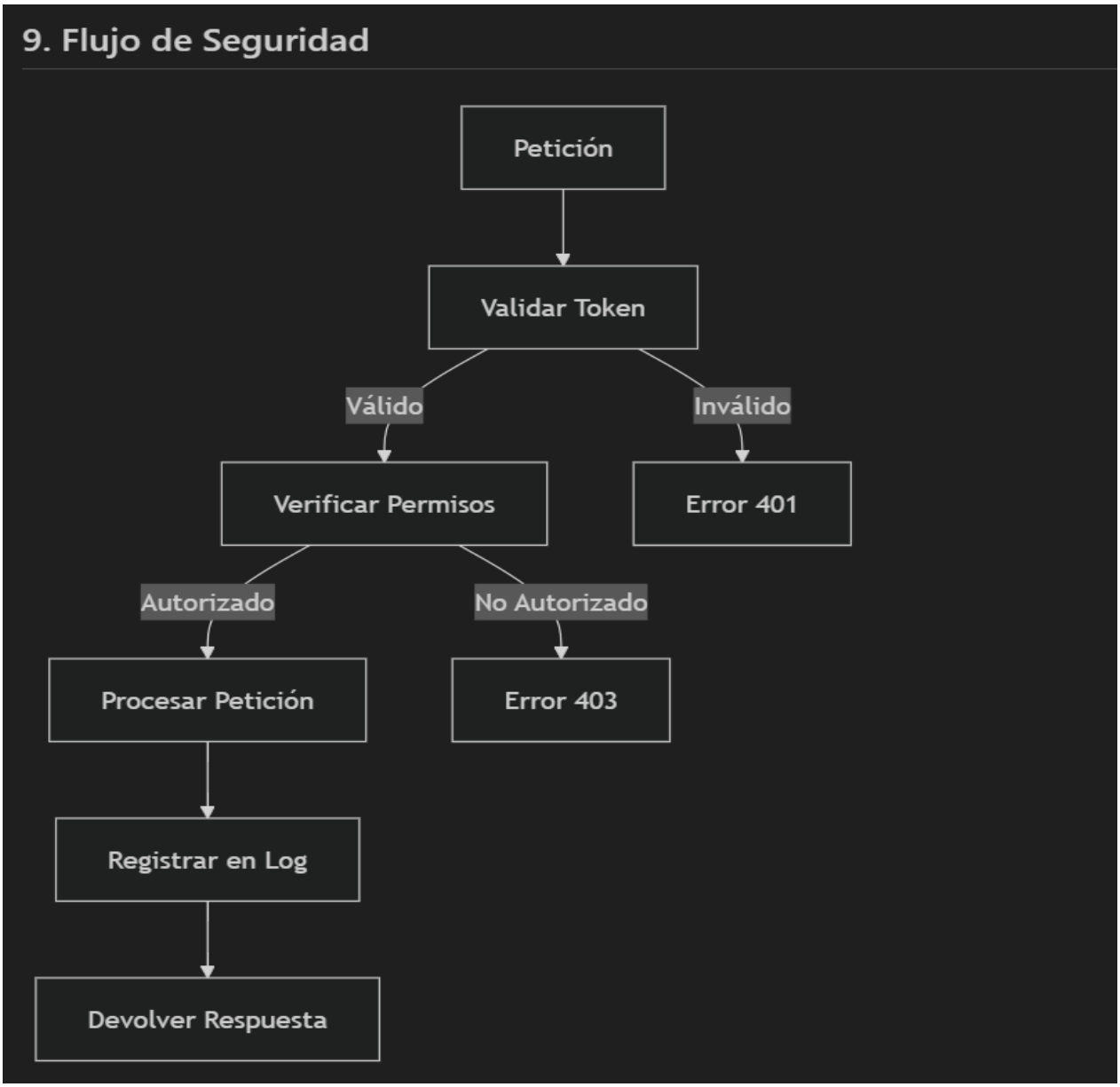

Fuente: elaboración Propia.

**Figura 8. Flujo de reportes.**



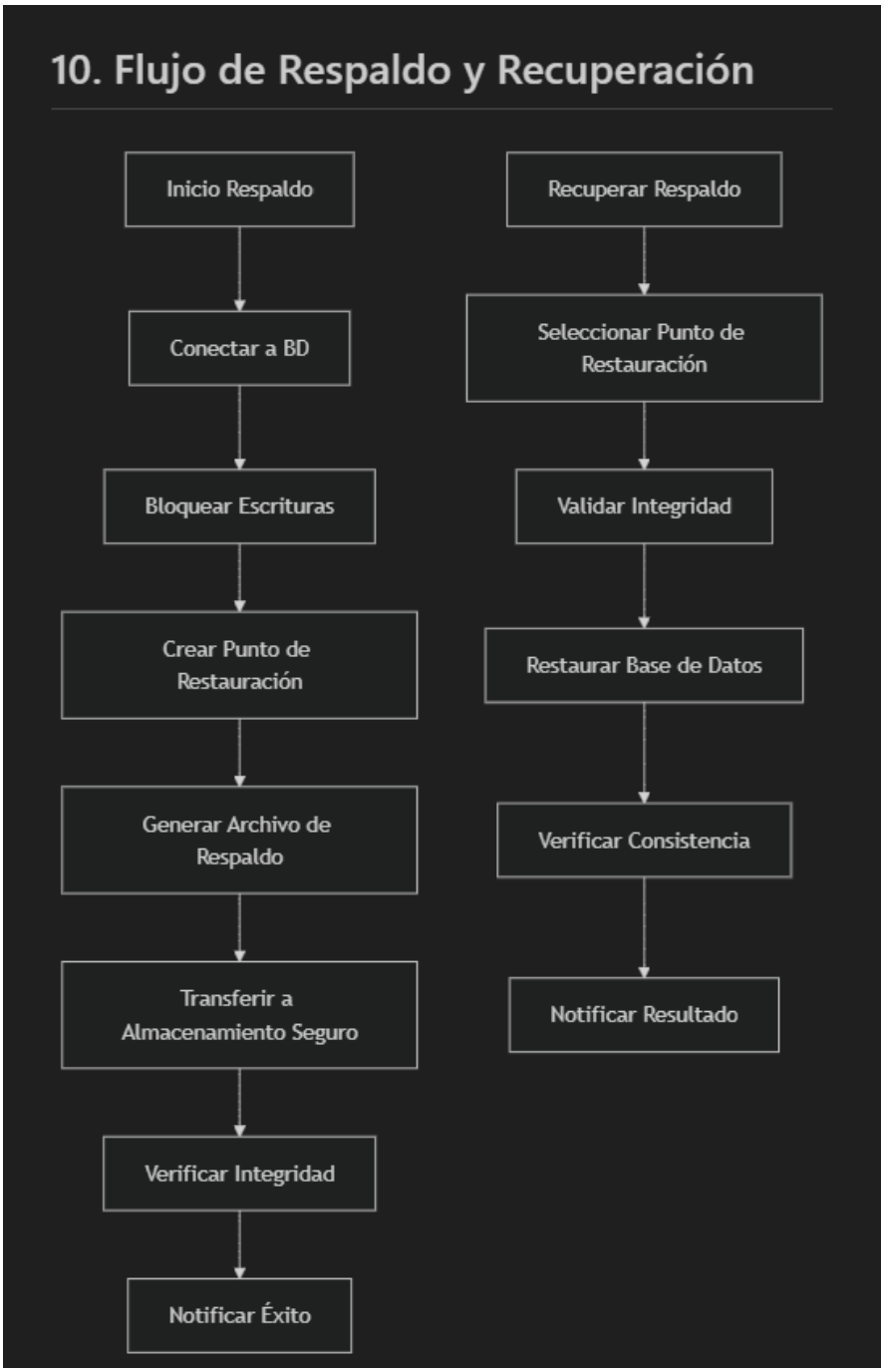

Fuente: elaboración Propia.

**Figura 9. Flujo de seguridad.**




Fuente: elaboración Propia.

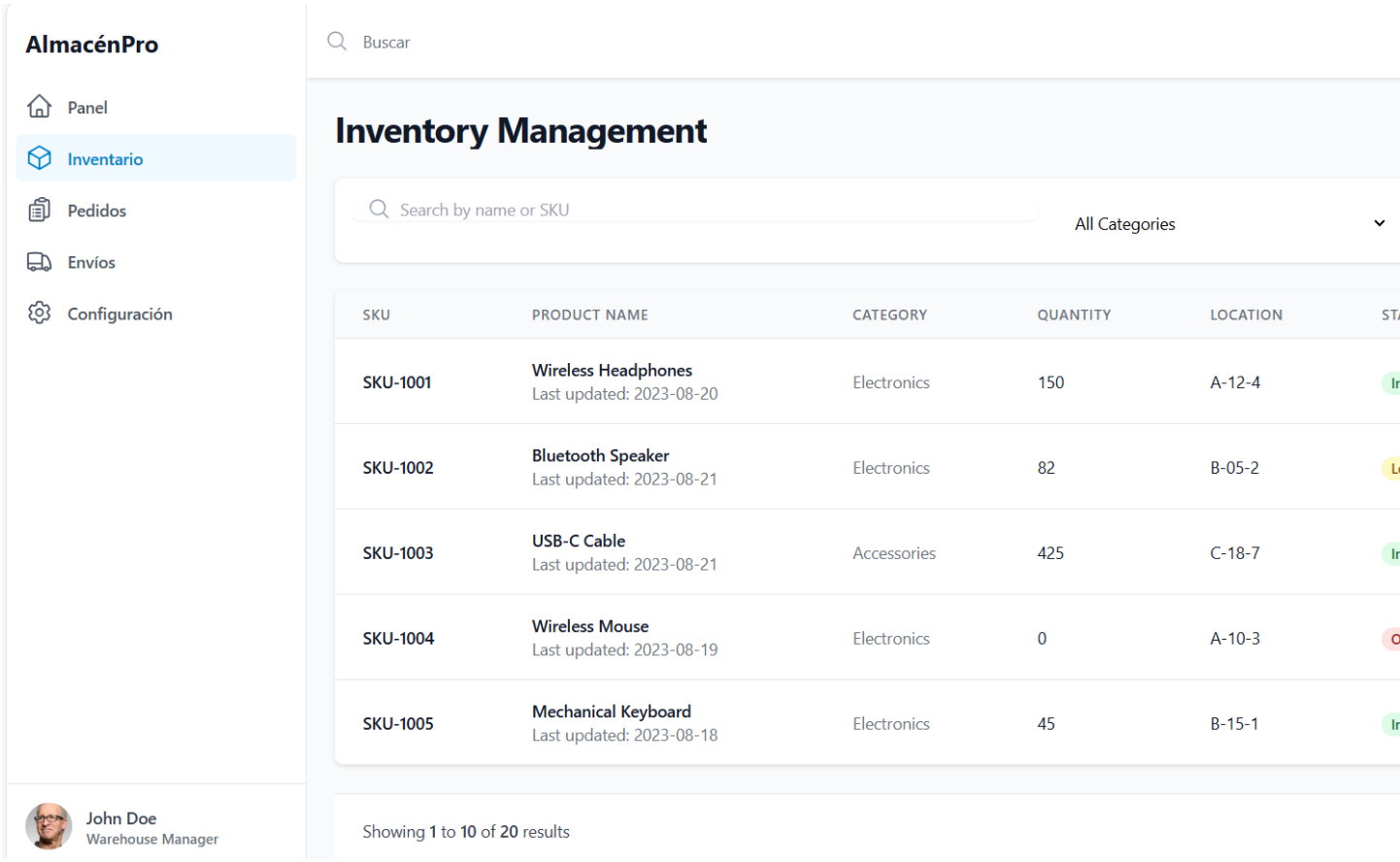
**Figura 9 Flujo de respaldo y recuperación.**




Fuente: elaboración Propia.

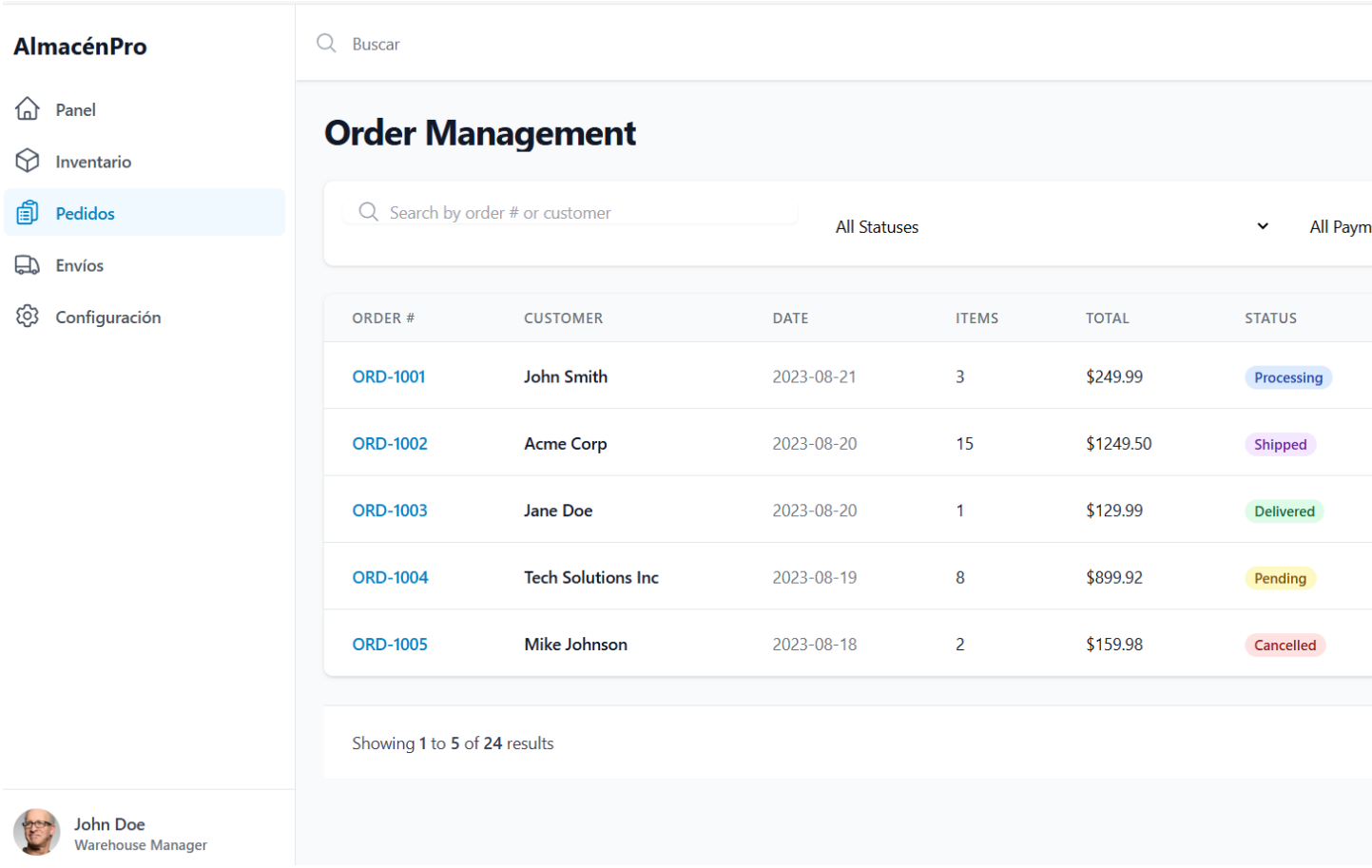
## Anexo 1 Capturas de pantalla del Frontend

Figura B-1. Interfaz de inventario



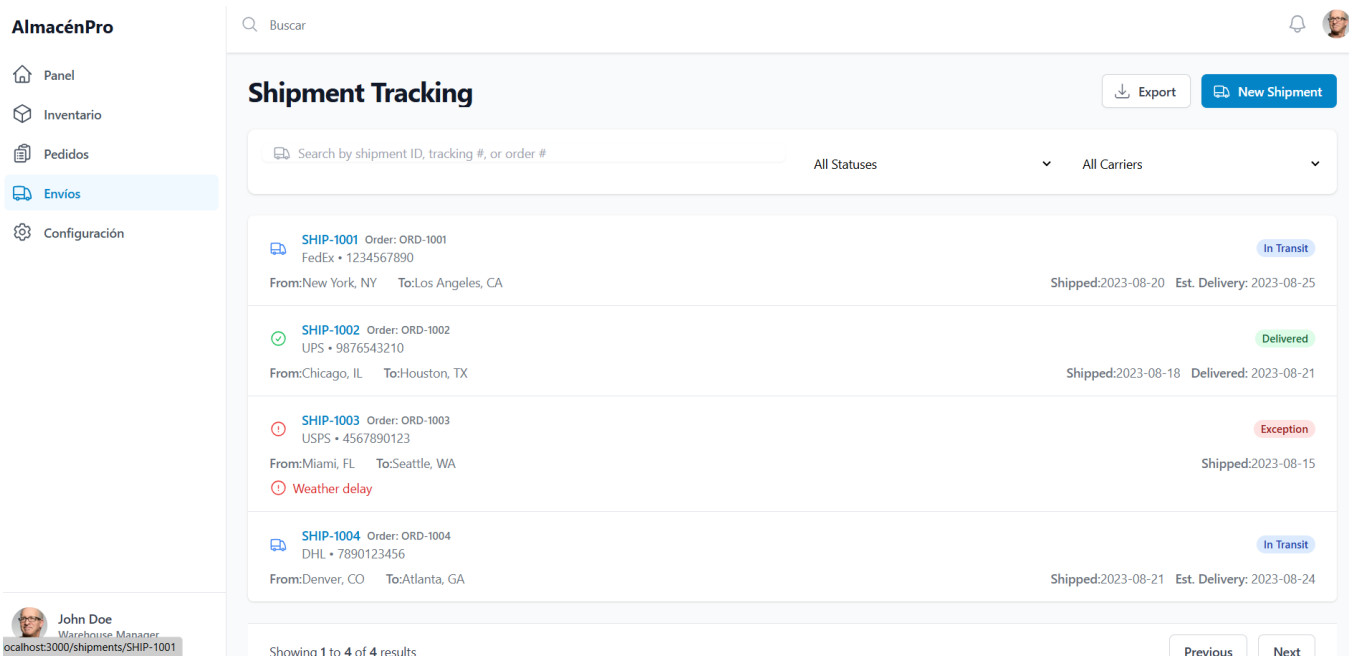

Fuente: elaboración Propia.

## Anexo 2 Interfaz de Pedidos



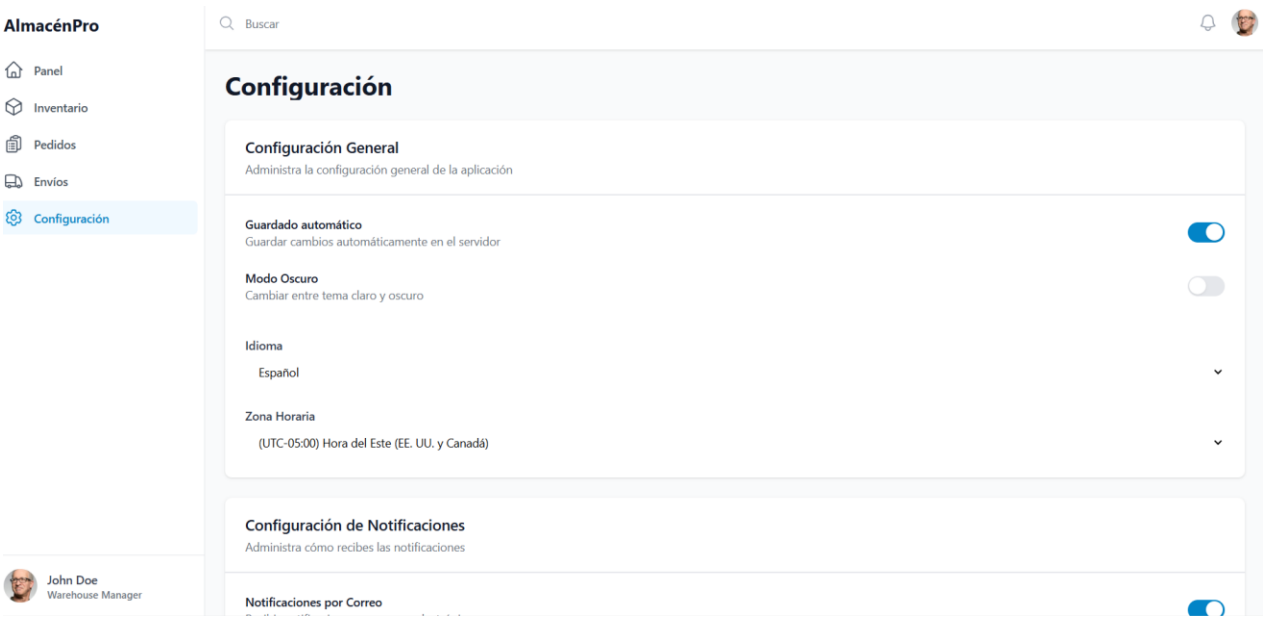

Fuente: elaboración Propia.

## Anexo 3  Interfaz de usuario 3



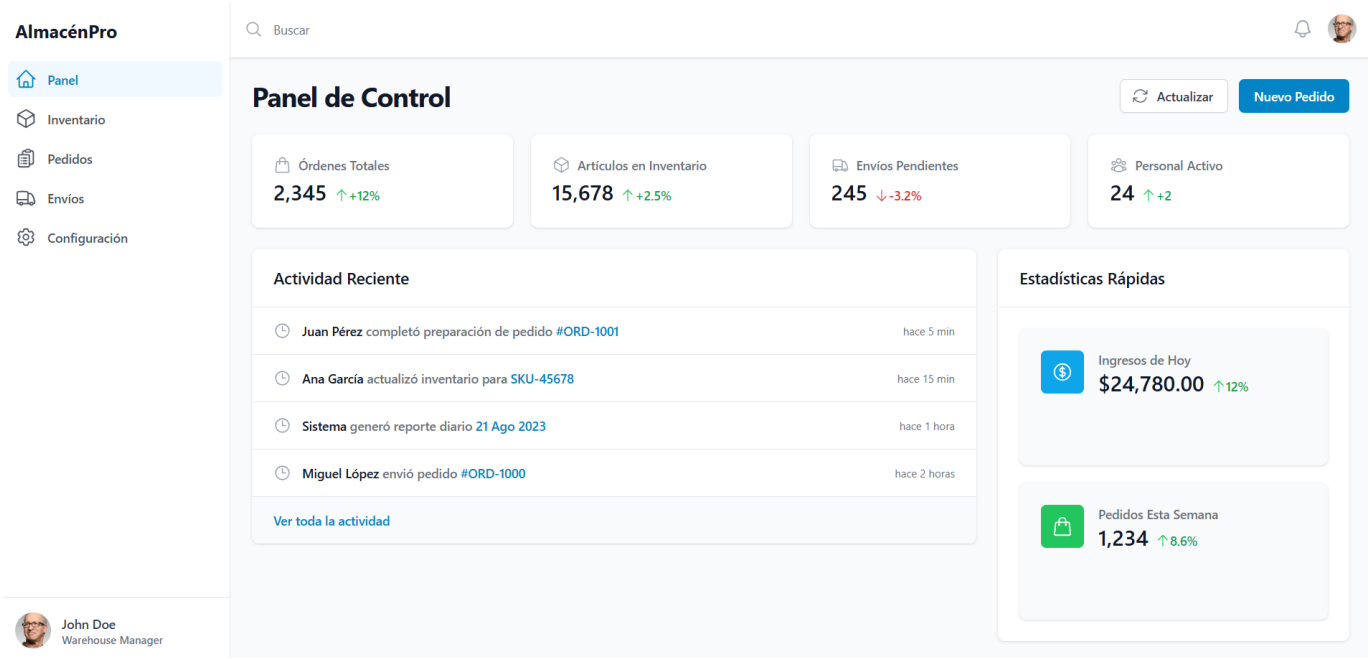

Fuente: elaboración Propia.

## Anexo 5. Interfaz de configuración




Fuente: elaboración Propia.

## Anexo 6 Interfaz de panel principal




Fuente: elaboración Propia.

## Anexo 7 Extracto del estudio financiero (COCOMO II y métricas)

| **Parámetro** | **Valor (córdobas)** | **Valor (USD)** | **Justificación** |
|---|---|---|---|
| Salario Base Promedio | C$ 10,000.00 | | Premisa de investigación (Nicaragua) |
| Tasa de Cambio | C$ 36.15 | | Tasa oficial 2025 |
| Salario Base (USD) | | $276.62 | (10,000 / 36.15) |
| Factor de Carga (Overhead) | | 2.2x | Costos indirectos y beneficios de ley |
| **Costo Persona-Mes (PM)** | | **$610.00** | ($276.62 \times 2.2, redondeado) |

| **Componente** | **Parámetro** | **Valor** | **Justificación** |
|---|---|---|---|
| Tamaño | Size (KLOC) | 12 | Alcance de (módulos críticos) |
| Escala | SF (Suma Factores) | 16.69 | Entorno Pyme (PMAT Bajo) |
| Esfuerzo | **II**EM (Producto Múltiple.) | 0.85 | Fiabilidad nominal (MVP), Analista Alto |
| Constantes | A, B, C | 2.94, 0.91, 3.67 | Estándar COCOMO II Post-Arquitectura |

| Métrica | Símbolo | Fórmula / Cálculo | Resultado |
|---|---|---|---|
| Exponente de Escala | E | $0.91 + (0.01 \times 16.69) $ | **1.0769** |
| Esfuerzo Total | PM | $2.94 \times (12) ^ {1.0769} \times 0.85$ | **39.11 PM** |
| Factor de Duración | F | $0.28 + 0.2 \times (1.0769 - 0.91) $ | **0.31338** |
| Duración del Proyecto | TDEV | $3.67 \times (39.11) ^ {0.31338} $ | **12.88 Meses** |
| Personal Promedio | Staff | $39.11 / 12.88$ | **~3 Personas** |
| **Costo Total (CAPEX)** | | **$39.11 \times \$610$** | **US$ 23,857.10** |

| Indicador | Símbolo | Valor | Justificación |
|---|---|---|---|
| Inversión Inicial (Año 0) | | ($23,857.10) | Costo Total COCOMO II (Tabla 3) |
| Gasto Operativo (OpEx) | OpEx | $36.00 / mes | Costo de hosting y mantenimiento |
| Beneficio Esperado | B | $300.00 / mes | Estimación de ahorros/eficiencia |
| Flujo de Caja Neto Anual | FC | $3,168.00 | ($300 - $36) \times 12 |

| Indicador | Símbolo | Valor | Justificación |
|---|---|---|---|
| Horizonte de Evaluación | n | 5 años | Estándar para proyectos de SW |
| Tasa de Descuento | k | 10.0% | TMAR (Tasa Mínima Aceptable) |

| Año (t) | Flujo de Caja (FC) | Factor de Descuento (10 por ciento) | Flujo Descontado (VPA) | Flujo Acumulado |
|---|---|---|---|---|
| **0** | **($23,857.10)** | **1.0000** | **($23,857.10)** | **($23,857.10)** |
| **1** | **$3,168.00** | **0.9091** | **$2,880.00** | **($20,977.10)** |
| **2** | **$3,168.00** | **0.8264** | **$2,618.18** | **($18,358.92)** |
| **3** | **$3,168.00** | **0.7513** | **$2,380.16** | **($15,978.76)** |
| **4** | **$3,168.00** | **0.6830** | **$2,163.79** | **($13,814.97)** |
| **5** | **$3,168.00** | **0.6209** | **$1,967.08** | **($11,847.89)** |
| **6** | **$3,168.00** | **0.5645** | **$1,788.26** | **($10,059.63)** |
| **7** | **$3,168.00** | **0.5132** | **$1,625.69** | **($8,433.94)** |
| **8** | **$3,168.00** | **0.4665** | **$1,477.89** | **($6,956.05)** |
| **9** | **$3,168.00** | **0.4241** | **$1,343.53** | **($5,612.52)** |
| **10** | **$3,168.00** | **0.3855** | **$1,221.75** | **($4,390.77)** |

| Año (t) | Flujo de Caja (FC) | Factor de Descuento (10 por ciento) | Flujo Descontado (VPA) | Flujo Acumulado |
|---|---|---|---|---|

| Año (t) | Flujo de Caja (FC) | Factor Descuento (10 por ciento) | Flujo Descontado (VPA) |
|---|---|---|---|
| **0** | ($23,857.10) | 1.0000 | ($23,857.10) |
| **11** | $3,168.00 | 0.9091 | $2,880.00 |
| **12** | $3,168.00 | 0.8264 | $2,618.18 |
| **13** | $3,168.00 | 0.7513 | $2,380.16 |
| **14** | $3,168.00 | 0.6830 | $2,163.79 |
| **15** | $3,168.00 | 0.6209 | $1,967.08 |
| **Total** | | **VAN (VPN) =** | **(US$ 11,947.89)** |

| Métrica | Cálculo | Resultado | Interpretación (Nueva Conclusión) |
|---|---|---|---|
| **Inversión Inicial** | Costo COCOMO II | **US$ 23,857.10** | **CAPEX** del proyecto de desarrollo. |
| **VAN (VPN)** | 15 años | **US$ 240.62** | **Positivo.** El proyecto es rentable a 15 años. |
| **TIR** | Tasa que hace VAN = 0 | **10.16%** | **Mayor que la TMAR (10 por ciento).** El proyecto es aceptable. |
| **Payback** | $23,857.10 / 3,168.00$ | **7.53 Años** | El tiempo de recuperación es el mismo (no se descuenta). |
| **Viabilidad** | - | - | **Proyecto Viable** (Horizonte de 15 años). |

# Anexo 4 Precios y características de proveedores nube 2025.

## 1. Upcloud

| Memory | CPU | MaxIOPS Storage | Transfer | Price Worldwide * | Price Helsinki * |
|---|---|---|---|---|---|
| 1 GB | 1 | 25 GB | 1 TB | **$5/mo** $0.007/h | **$7.5/mo** $0.011/h |
| 2 GB | 1 | 50 GB | 2 TB | **$10/mo** $0.015/h | **$15/mo** $0.022/h |
| 4 GB | 2 | 80 GB | 4 TB | **$20/mo** $0.030/h | **$30/mo** $0.045/h |
| 8 GB | 4 | 160 GB | 5 TB | **$40/mo** $0.060/h | **$60/mo** $0.089/h |
| 16 GB | 6 | 320 GB | 6 TB | **$80/mo** $0.119/h | **$120/mo** $0.179/h |
| 32 GB | 8 | 640 GB | 7 TB | **$160/mo** $0.238/h | **$240/mo** $0.357/h |
| 48 GB | 12 | 960 GB | 9 TB | **$240/mo** $0.357/h | **$360/mo** $0.536/h |
| 64 GB | 16 | 1280 GB | 10 TB | **$320/mo** $0.476/h | **$480/mo** $0.714/h |
| 96 GB | 20 | 1920 GB | 12 TB | **$480/mo** $0.714/h | **$720/mo** $1.071/h |

## 2. Amazon

| 3,50 USD USD/mes | 5 USD USD/mes | 10 USD USD/mes | 20 USD USD/mes | 40 USD USD/mes | 80 USD USD/mes | 160 USD USD/mes |
|---|---|---|---|---|---|---|
| Memoria **512 MB** Procesador de **1 núcleo** Disco SSD de **20 GB** **1 TB** de transferencia* | Memoria de **1 GB** Procesador de **1 núcleo** Disco SSD de **40 GB** **2 TB** de transferencia* | Memoria de **2 GB** Procesador de **1 núcleo** Disco SSD de **60 GB** **3 TB** de transferencia* | Memoria de **4 GB** Procesador de **2 núcleos** Disco SSD de **80 GB** **4 TB** de transferencia* | Memoria **8 GB** Procesador de **2 núcleos** Disco SSD de **160 GB** **5 TB** de transferencia* | Memoria de **16 GB** Procesador de **4 núcleos** Disco SSD de **320 GB** **6 TB** de transferencia* | Memoria de **32 GB** Procesador de **8 núcleos** Disco SSD de **640 GB** **7 TB** de transferencia* |

### 3. Google Cloud GCP.

## Servicio de SQL

| | Precio (USD) | 1 año de compromiso | 3 años de compromiso |
|---|---|---|---|
| vCPUs | 0,0413 USD por vCPU | 0,03098 USD | 0,01982 USD |
| Memoria | 0,0070 USD por GB | 0,00525 USD | 0,00336 USD |
| | | Precio | |
| Almacenamiento | | • Capacidad de almacenamiento SSD: 0,170 USD por GB al mes<br>• Copias de seguridad utilizadas: 0,080 USD por GB al mes | |

## Servicio de App Engine.

| Recurso | Unidad | Coste de la unidad |
|---|---|---|
| vCPU | Por hora de núcleo | 0,0526 USD |
| Memoria | Por GB por hora | 0,0071 USD |

**Tabla comparativa.**

| Proveedor | Servicio de Base de Datos (PostgreSQL) | Precio Base Mensual (USD) | Servicio de Aplicación (Web/API) | Precio Base Mensual (USD) | OpEx Total Estimado (USD/mes) |
|---|---|---|---|---|---|
| Amazon Web Services (AWS) | Amazon RDS (db.t4g.micro, 2vCPU, 1GB RAM) | $20.00 | AWS App Runner (Configuración Mínima) | $19.00 | $39.00 |
| Google Cloud Platform (GCP) | Cloud SQL (db-f1-micro, 1vCPU, 0.6GB RAM) | $28.00 | Cloud Run (1 CPU, 0.5GB RAM, 0% CPU al inactivo) | $15.00 | $43.00 |
| Microsoft Azure | Azure Database for PostgreSQL (Basic Tier, 1 vCore, 1GB RAM) | $24.00 | Azure App Service (B1 Basic Tier) | $13.50 | $37.50 |
| DigitalOcean | Managed Databases (Starter, 1vCPU, 1GB RAM) | $15.00 | App Platform (Basic Tier, 0.5GB RAM) | $5.00 | $20.00 |

**Tabla 2 Planes base de referencia para producción liviana. Precios estimados en USD/mes.**

| Proveedor/Plan | vCPU | RAM | SSD | Transferencia | Precio |
|---|---|---|---|---|---|
| **AWS Lightsail (Linux/IPv4) 1 GB** | 2* | 1 GB | 40 GB | 2 TB | $7 |
| **AWS Lightsail (Linux/IPv4) 2 GB** | 2 | 2 GB | 60 GB | 3 TB | $12 |
| **DigitalOcean Droplet (Basic) 1 GB** | 1 | 1 GB | 25 GB | 1 TB | $6 |
| **Azure VM B1s (Linux)** | 1 | 1 GB | 30 GB** | — | ~$7.59 |
| **GCP e2-micro** | 2 (shared) | 1 GB | disco aparte | — | $6.11–$9.78 |
| **Vultr Cloud Compute 1 GB** | 1 | 1 GB | 25 GB | 1 TB | $5 |
| **Linode/Akamai Nanode 1 GB** | 1 | 1 GB | 25 GB | 1 TB | $5 |