PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

INGENIERÍA EN SISTEMAS DE INFORMACIÓN



Trabajo de Titulación

Tema:

Desarrollo de un sistema web para la gestión de inventarios en pequeñas y medianas empresas (PYMES) con integración de modelos predictivos

AUTOR:

DANIEL CAMPOVERDE

TUTOR:

ING. PABLO DANIEL MARCILLO LARA

QUITO DM, JUNIO DE 2026

Dedicatoria

A mi mamá Sylvia Oña y a mi hermano Luis Campoverde, por su apoyo incondicional en cada etapa de este camino, por creer en mí incluso cuando yo dudaba, y por nunca dejarme caminar solo, por estar siempre presente, por su paciencia y su ánimo constante, y por ser un pilar fundamental en los momentos más difíciles.

A mi abuelito Alejandro Oña, por su cariño y sus enseñanzas, que siguen siendo guía en mi vida.

Este logro también es de ustedes.

INDICE DE CONTENIDO

CAPITULO 1. INTRODUCCIÓN 3
1.1.Tema.. 3
1. 2. Justificación 3
1.3. Planteamiento del problema 4
1.4. Objetivos 5
1.4.1 Objetivo General: 5
1.4.2 Objetivos Específicos: 5
1.5. Antecedentes, marco teórico y conceptual 6
1.5.1 Antecedentes o Marco Referencial 6
1.5.2 Marco Teórico 7
1.5.3 Marco Conceptual 8
1.6. Alcance 9
CAPITULO 2. FUNDAMENTACIÓN TEÓRICA 10
2.1. Sistemas de información para la gestión de inventarios 10
2.1.1. Arquitecturas de sistemas 10
2.1.2. Gestión de Inventarios en Pymes 11
2.1.3. Auditoría y Trazabilidad de operaciones 12
2.2. Visualización de datos y dashboards 12
2.2.1. Herramientas y tecnologías 13
2.2.2. Arquitectura de comunicación frontend y backend para dashboards 13
2.3. Análisis de datos e inteligencia artificial 13
2.3.1. Conceptos de Business Intelligence 14
2.3.2. Análisis predictivo en inventarios 14
2.3.3. Modelos de predicción de demanda 15
2.4. Antecedentes investigativos 15
2.4.1. Soluciones existentes en el mercado 15
CAPITULO 3. DISEÑO E IMPLEMENTACIÓN 17
3.1. Metodología de desarrollo 17
3.1.1. Enfoque metodológico 20
3.1.2. Herramientas y tecnologías 22
3.2. Arquitectura del sistema 22
3.2.1. Arquitectura general 23
3.2.2. Casos de uso del sistema 26
3.3. Diseño de base de datos 33

3.3.1. Entidades Principales 34
3.3.2 Automatizaciones mediante triggers 40
3.4. Desarrollo de módulos funcionales 40
3.4.1. Gestión de Productos 40
3.4.2. Movimientos de Inventario 41
3.4.3. Alertas Automáticas 41
3.5. Diagramas Técnicos del sistema 42
3.5.1 Flujo de Inicio de Sesión 42
3.5.2 Flujo de Gestión de Inventario 46
3.5.3 Flujo de Gestión de Productos 48
3.6. Implementación de Dashboards Interactivos 50
3.6.1. Definición de KPIs 50
3.6.2. Diseño de Interfaces 51
3.6.3. Visualización de datos 51
3.7. Implementación del Módulo de Análisis Predictivo 52
3.7.1. Selección de variables 52
3.7.2. Preparación del conjunto del entrenamiento 54
3.7.3. Ajuste de hiperparámetros 54
3.7.4. Comparación entre modelos 56
3.8. Implementación de Reportes Inteligentes 57
3.8.1. Análisis Automatizado 57
3.8.2. Detección de Anomalías 57
3.9. Pruebas y Validación 58
3.9.1. Pruebas Funcionales 58
3.9.2. Pruebas de Integración 59
CAPITULO 4. ANÁLISIS DE RESULTADOS 60
4.1. Evaluación del sistema 60
4.2. Pruebas de usabilidad 62
4.2.1. Diseño de pruebas 62
4.2.2. Resultados 63
4.3. Limitaciones del proyecto 64
CAPITULO 5. CONCLUSIONES Y TRABAJOS FUTUROS 65
5.1. Conclusiones 65
5.2. Trabajos Futuros 67
BIBLIOGRAFÍA 69

ANEXOS 71

ÍNDICE DE TABLAS

Tabla 1. Herramientas de desarrollo de dashboards 13
Tabla 2. Modelos de IA para predicción de demanda 15
Tabla 3. Tabla comparativa entre soluciones existentes para gestión de inventarios ... 16
Tabla 4. Tabla de Requerimientos Funcionales 17
Tabla 5. Tabla de Requerimientos No Funcionales 19
Tabla 6. Tabla de Iteraciones para desarrollo del proyecto 21
Tabla 7. Herramientas y Tecnologías escogidas para el desarrollo del proyecto 22
Tabla 8. Caso de uso 1 27
Tabla 9. Caso de uso 2 28
Tabla 10. Caso de uso 3 28
Tabla 11. Caso de uso 4 28
Tabla 12. Caso de uso 5 29
Tabla 13. Caso de uso 6 30
Tabla 14. Caso de uso 7 30
Tabla 15. Caso de uso 8 31
Tabla 16. Caso de uso 9 31
Tabla 17. Caso de uso 10 32
Tabla 18. Reglas de Integridad 35
Tabla 19. KPIs presentados en el sistema 51
Tabla 20. Columnas del dataset de entrenamiento 52
Tabla 21.Métricas de evaluación del modelo de IA 55
Tabla 22. Comparación modelos en predicción de ventas 56
Tabla 23. Comparación modelos en predicción de demanda 56
Tabla 24. Pruebas Funcionales del sistema 58
Tabla 25. Estado funcional de cada módulo 62

ÍNDICE DE FIGURAS

Figura 1. Arquitectura del sistema 23
Figura 2. Diagrama de casos de uso 27
Figura 3. Diagrama Entidad-Relación de la base de datos 36
Figura 4. Modelado Lógico de la base de datos 37
Figura 5. Modelado físico de la base de datos 38
Figura 6. Modelado conceptual de la base de datos 39
Figura 7. Flujo de Inicio de Sesión 43
Figura 8. Diagrama de secuencia-Login 44
Figura 9. Diagrama de Secuencia- Gestión de Inventario 47
Figura 10. Diagrama de Secuencia- Gestión de Productos 49
Figura 11. Prueba Registro de Producto 79
Figura 12. Prueba de ventas 80
Figura 13. Prueba ventas con stock insuficiente 81
Figura 14. Prueba de alerta de stock bajo 81
Figura 15. Prueba de predicción de IA 82
Figura 16. Prueba de Registro de logs 82
Figura 17. Resultados pruebas de Integración y Funcionales 83
Figura 18. Interfaz de Login 84
Figura 19. Interfaz de Registro 84
Figura 20. Dashboard Inteligente 85
Figura 21. Gestión de Productos 85
Figura 22. Módulo de Inventario 86
Figura 23. Módulo de Usuarios 86
Figura 24. Módulo de ventas 86
Figura 25. Inicialización del Backend 87
Figura 26. Instalar dependencias en backend 87

CAPITULO 1. INTRODUCCIÓN

En este capítulo, se presentan los componentes iniciales que enmarcan el desarrollo de la tesis actual. Primero se presenta la justificación del proyecto, en la que se explica la importancia y pertinencia de su realización; después, se expone el planteamiento del problema que impulsa la investigación; luego, se definen los objetivos generales y específicos que guían el desarrollo de la obra; por último, se determinan las limitaciones y el alcance del proyecto para especificar lo que será abordado y aquello que no estará cubierto.

1.1.Tema

Desarrollo de un sistema web para la gestión de inventarios en pequeñas y medianas empresas (PYMES) con integración de modelos predictivos.

1. 2. Justificación

Para la competitividad y la sostenibilidad de las pequeñas y medianas empresas (PYMES), es esencial una gestión efectiva del inventario, porque esta facilita que se mantenga un balance óptimo entre la falta y el exceso de productos. Una administración poco eficiente puede producir consecuencias negativas, como una reducción en la satisfacción del cliente, pérdidas financieras y una escasa rotación de inventarios.

El presente proyecto propone, en este marco, el diseño y la puesta en marcha de un sistema web para el manejo de inventarios que incorpore modelos de análisis predictivo-fundamentados en inteligencia artificial, dashboards interactivos e informes inteligentes. El propósito es perfeccionar la toma de decisiones a través del análisis, la visualización y la proyección de datos históricos sobre inventario y ventas.

El uso de métodos de inteligencia artificial hará posible prever la demanda, reconocer patrones de conducta, identificar irregularidades y pronosticar posibles situaciones futuras. Esto posibilitará una programación más eficaz de las compras y el control del nivel de existencias.

Así, el objetivo es optimizar la productividad de las empresas y la eficiencia en sus operaciones. Además, la propuesta fomenta que se utilicen tecnologías modernas para examinar datos y para visualizar la información de forma interactiva, lo que brinda a los

encargados del inventario datos exactos, actualizados y sencillos de entender. Esto mejora la capacidad de reacción frente a alteraciones del mercado y promueve que las decisiones estratégicas se tomen en base a pruebas.

1.3. Planteamiento del problema

En muchas pequeñas y medianas empresas (PYMES), la administración de inventarios se lleva a cabo manualmente o con sistemas restringidos que no posibilitan una supervisión eficaz de los productos. Esta circunstancia complica la detección temprana de productos críticos, el seguimiento apropiado de los niveles de existencias y la identificación de irregularidades en las transacciones de inventario.

Por lo tanto, las decisiones acerca de la rotación, el reabastecimiento y el almacenamiento de productos a menudo se fundamentan en información incompleta o en la intuición, lo que aumenta las posibilidades de cometer errores en la gestión del stock. Esto puede resultar en pérdidas financieras a causa de un inventario excesivo, rupturas del stock o una rotación insuficiente de productos, lo que impacta directamente la rentabilidad y la capacidad de competir de las compañías.

Adicionalmente, la ausencia de dashboards interactivos, reportes inteligentes y herramientas de análisis predictivo impide transformar los datos históricos de ventas e inventario en información estratégica. Esto limita la capacidad de anticipar la demanda, identificar patrones de comportamiento y prever posibles escenarios futuros, restringiendo la planificación eficiente de las adquisiciones y la toma de decisiones basada en datos.

En este contexto, surge la necesidad de desarrollar un sistema web de gestión de inventarios que integre dashboards interactivos, reportes inteligentes y modelos de análisis predictivo basado en inteligencia artificial. Este sistema permitirá proporcionar una visión clara, dinámica y en tiempo real del estado del inventario, facilitando la optimización de la eficiencia operativa y fortaleciendo la toma de decisiones estratégicas en las PYMES.

1.4. Objetivos

Los objetivos que guían la ejecución de esta investigación se exponen en esta sección. Primero se presenta el objetivo general, que establece la finalidad principal del trabajo; después, se describen los objetivos específicos, los cuales dividen dicho propósito en metas específicas y logrables durante el proceso de diseño, desarrollo e implementación del sistema web propuesto.

1.4.1 Objetivo General:

Crear y poner en marcha un sistema web de gestión de inventarios para pequeñas y medianas empresas (PYMES), que incorpore cuadros de mando interactivos, informes inteligentes y modelos de análisis predictivo-fundamentados en inteligencia artificial. El objetivo es optimizar la gestión de existencias, incrementar la eficacia operativa y robustecer el proceso de toma de decisiones empresariales.

1.4.2 Objetivos Específicos:

- Diseñar la estructura del sistema web y de la base de datos para posibilitar un registro, una conservación, una actualización y una consulta eficaces de los datos del inventario.
- Crear paneles de control interactivos que muestren indicadores fundamentales de gestión, como la rotación de productos, los niveles de inventario y las alertas automáticas por escasez de stock, permitiendo así la visualización en tiempo real.
- Crear informes inteligentes que puedan producir análisis automatizados y ofrecer sugerencias fundamentadas en los datos de ventas e inventario históricos.
- Desarrollar e implementar un modelo de análisis predictivo fundamentado en inteligencia artificial, capaz de prever la demanda futura, detectar tendencias conductuales y anticipar eventuales quiebras o sobrestock.

1.5. Antecedentes, marco teórico y conceptual

En esta sección se muestra la fundamentación teórica y referencial que apoya el progreso de este trabajo. Primero, se presentan los antecedentes o el marco referencial, que examina investigaciones anteriores y soluciones disponibles en el campo de la gestión de inventarios para determinar sus limitaciones y alcances. Después, se elabora el marco teórico, donde se explican las bases conceptuales y los enfoques teóricos que respaldan la creación y la ejecución del sistema sugerido.

1.5.1 Antecedentes o Marco Referencial

Dado que la administración de inventarios afecta directamente a la productividad y al rendimiento económico de las empresas, ha sido objeto de un estudio extenso en el campo de la gestión empresarial y la ingeniería de sistemas. Se ha observado en años recientes un cambio de métodos manuales o fundamentados en hojas de cálculo a sistemas automatizados que incluyen capacidades para el análisis y la visualización de datos.

Varios estudios han tratado el desarrollo de sistemas de inventario que tengan funciones elementales. Por ejemplo, crearon una solución con tableros de control básicos, lo que les permitió optimizar la exactitud en la anotación y supervisión de inventarios. No obstante, su propuesta no incluye el empleo de métodos de análisis predictivo o modelos fundamentados en inteligencia artificial para prever la demanda.

Oracle NetSuite y SAP Inventory Management son soluciones comerciales que existen en el sector de los negocios. Proporcionan módulos sólidos para la administración de inventarios, con características avanzadas de análisis y control. Sin embargo, la adopción de estas plataformas en las pequeñas y medianas empresas (PYMES) es restringida debido a los altos costos de licencia, implementación y mantenimiento.

Por otra parte, plataformas en la nube como TradeGecko y Zoho Inventory han posibilitado que las empresas más pequeñas digitalicen sus trámites de inventario. Aunque son accesibles, estos métodos tienen restricciones en lo que respecta a la personalización de paneles de control, la creación de informes inteligentes y avanzados y la integración de modelos predictivos que se basan en inteligencia artificial.

A partir de este análisis, se detecta una importante brecha en el desarrollo de sistemas de gestión de inventarios que incorporen con eficacia la visualización interactiva, las competencias analíticas predictivas y las funcionalidades operativas enfocadas en las PYMES. El presente trabajo tiene como objetivo ayudar a cerrar dicha

brecha en este contexto, a través de una solución tecnológica que sea accesible, intuitiva y tenga capacidades analíticas avanzadas. Esta solución permitirá mejorar la toma de decisiones sin tener que integrar sistemas ERP externos de manera complicada.

1.5.2 Marco Teórico

El sistema se basa en cuatro pilares teóricos fundamentales. Primero, se revisitan las teorías tradicionales de la gestión de inventarios, como el modelo EOQ (que posibilita estimar la cantidad ideal de pedido para reducir al mínimo los costos de almacenamiento y orden), la clasificación ABC y la teoría de restricciones.

En segundo lugar, el proyecto se basa en los principios del Business Intelligence (BI) y la visualización de datos. Se considera como punto de partida los lineamientos para el diseño de tableros de control que fueron sugeridos por (Tufte, 2001) y (Stephen Few, 2006), los cuales subrayan la eficacia, la claridad y la exactitud al mostrar visualmente indicadores esenciales. Además, para la identificación automática de patrones y anomalías en los datos históricos de inventario, se incorporan nociones de minería de datos y análisis exploratorio.

Las teorías del diseño y el desarrollo de sistemas de información son lo que conforman la tercera columna. Se implementa una arquitectura en capas que propicia la escalabilidad, mantenibilidad y modularidad del sistema. Se utiliza una metodología de desarrollo que es iterativa e incremental, lo que permite ajustes constantes basados en la retroalimentación.

Además, se utilizan principios de diseño centrado en el usuario con el fin de asegurar interfaces intuitivas.

Por último, el proyecto se basa en teorías sobre la toma de decisiones a nivel organizacional. Se basa en el modelo racional de toma de decisiones, que sostiene que las decisiones más adecuadas provienen del análisis sistemático de información pertinente. El sistema propuesto opera como un Sistema de Soporte a la Decisión, brindándole a los administradores información que ha sido analizada, visualizada y procesada, lo cual reemplaza la intuición por evidencia fundamentada en datos.

1.5.3 Marco Conceptual

En esta parte se definen los conceptos esenciales que respaldan el desarrollo del trabajo actual, con la intención de crear un lenguaje común y facilitar la comprensión de los términos técnicos utilizados en toda la tesis. El sistema de administración propuesto y otros componentes conceptuales vinculados a su diseño e implementación se explican a continuación.

1.5.3.1. Sistema para administrar inventarios

Aplicación creada para llevar un registro, una actualización, un control y una supervisión exhaustiva del ciclo de vida de los productos en el interior de una organización, desde la compra hasta su venta o consumo.

1.5.3.2. Dashboard Interactivos

Interfaces gráficas de usuario que muestran un grupo de métricas comerciales e indicadores clave de desempeño (KPIs) de forma integrada, dinámica y casi en tiempo real.

1.5.3.3. KPI

Métrica que se puede cuantificar y que tiene importancia estratégica, lo cual posibilita la medición del desempeño o el éxito de un proceso particular.

1.5.3.4. Análisis predictivo

Método de análisis de datos que emplea algoritmos de inteligencia artificial y modelos estadísticos con el objetivo de reconocer patrones en información histórica y pronosticar comportamientos o sucesos por venir, como la demanda futura de productos en inventarios.

1.6. Alcance

El proyecto incluye la creación y puesta en marcha de un sistema web de administración de inventarios dirigido a las pequeñas y medianas empresas (PYMES). Este sistema incluirá paneles interactivos, informes inteligentes y un módulo de análisis predictivo basado en inteligencia artificial, cuyo propósito es optimizar el proceso de tomar decisiones corporativas.

El sistema posibilitará la gestión, consulta, actualización y control de datos vinculados a productos, niveles de existencias y transacciones de entrada y salida. Además, producirá indicadores esenciales de gestión, como la rotación de productos, los niveles de stock y las alertas automáticas en situaciones de inventario crítico.

Además, los dashboards brindarán una visualización interactiva y dinámica de la información a través de gráficos e indicadores de rendimiento. Por su parte, los reportes inteligentes ofrecerán análisis automatizados que permitirán detectar irregularidades, tendencias y patrones en el comportamiento del inventario.

Un módulo de análisis predictivo, que posibilitará prever la demanda futura de productos, anticipar eventuales quiebras o excesos de inventario y respaldar el planeamiento efectivo de las compras, se incorporará como un componente esencial del sistema.

El proyecto incluye el diseño de la arquitectura del sistema, la modelación de la base de datos, el desarrollo de módulos funcionales y la aplicación de métodos para analizar datos y crear modelos predictivos. También implica evaluar cómo es el sistema en términos de usabilidad y funcionalidad a través de pruebas con usuarios.

El sistema no permite la integración con sistemas contables o ERP externos, lo que constituye una limitación; se concentra únicamente en el análisis y administración de inventario.

CAPITULO 2. FUNDAMENTACIÓN TEÓRICA

La fundamentación teórica que respalda el diseño y la ejecución del sistema web de gestión de inventarios sugerido se expone en este capítulo. Se tratan las arquitecturas de software más apropiadas para este tipo de soluciones, los mecanismos de auditoría y trazabilidad que son imprescindibles para asegurar la confianza y la integridad de la información administrada por el sistema, así como también los conceptos vinculados a los sistemas de información utilizados en la gestión de inventarios y los desafíos específicos a los que se enfrentan las PYMES en esta área.

2.1. Sistemas de información para la gestión de inventarios

Para el rendimiento operativo eficaz en las organizaciones contemporáneas, los sistemas de información que se utilizan en la administración de inventarios son un soporte crucial. Estos sistemas posibilitan el registro, procesamiento, almacenamiento y recuperación de datos vinculados con los niveles de existencias, las transacciones de productos, así como las alertas de reabastecimiento. aseguran que un sistema de información bien diseñado mejora la toma de decisiones al brindar datos que son oportunos, relevantes y precisos.

En el ámbito de las pequeñas y medianas empresas (PYMES), la implementación de sistemas de información para inventarios se ha transformado desde instrumentos simples, como hojas de cálculo, hasta aplicaciones específicas. No obstante, numerosas PYMES todavía no tienen funciones avanzadas como la visualización interactiva o el análisis predictivo, lo que restringe su habilidad para prever la demanda.

2.1.1. Arquitecturas de sistemas

Según (Bass et al., 2013) la arquitectura de software establece la estructura esencial de un sistema, sus elementos constitutivos, las interrelaciones entre estos y los principios que orientan su diseño y desarrollo. Existen dos enfoques que predominan en los sistemas web de gestión de inventarios:

Arquitectura de capas (n-tier): Distingue entre la lógica de negocio, capa de aplicación, el almacenamiento de datos capa de datos, y la presentación, capa cliente. Este modelo hace más sencillo el mantenimiento, la escalabilidad y la seguridad.

Para proyectos como el actual, es aconsejable utilizar una arquitectura de tres niveles:

Capa de presentación: Interfaz web que incluye paneles interactivos.

Capa de lógica empresarial: Backend que lleva a cabo las reglas de inventario, procesa peticiones y emplea modelos de inteligencia artificial.

Capa de datos: Sistema de gestión de bases de datos (ya sea relacional o NoSQL) con el propósito de guardar productos, transacciones y pronósticos.

Asimismo, se incluye un subsistema de análisis predictivo que tiene la capacidad de funcionar como un microservicio autónomo o en el marco de la capa lógica del negocio, según la carga computacional.

2.1.2. Gestión de Inventarios en Pymes

Las PYMES tienen retos específicos que las distinguen de las corporaciones grandes en el campo de la gestión de inventarios. La gestión ineficaz del inventario es una de las causas más significativas de pérdidas económicas en compañías pequeñas, ya que no tienen los recursos económicos, tecnológicos y humanos que se requieren para poner en marcha sistemas sólidos de control de existencias (Atnafu & Balda, 2018).

Se encuentran los siguientes problemas como los más habituales en las PYMES: Primero, la dependencia de procesos que son manuales o semi-automatizados, tales como hojas de cálculo o registros en papel, los cuales son susceptibles a fallas humanas y no posibilitan una actualización del estado del inventario en tiempo real. En segundo lugar, la falta de trazabilidad total a lo largo del ciclo vital de los productos, la cual imposibilita determinar con exactitud cuándo, cómo y por qué fluctúan las cantidades de existencias. En tercer lugar, la ausencia de visibilidad acerca de indicadores esenciales como la rotación de productos o las tendencias de demanda, lo que fuerza a los gerentes a decidirse con base en su intuición más que en datos específicos.

Ante estas restricciones, la puesta en marcha de sistemas de información web dirigidos a las PYMES es una opción factible y económica. Los sistemas web hechos a medida, a diferencia de los ERP comerciales como SAP u Oracle, cuyos costos y licencias para su implementación son inalcanzables para la mayor parte de las pequeñas empresas, posibilitan que las funcionalidades sean ajustadas a lo que requiere concretamente el negocio, con una curva de aprendizaje pequeña y sin necesitar tecnología costosa.

2.1.3. Auditoría y Trazabilidad de operaciones

La trazabilidad en los sistemas de información empresarial se refiere a la habilidad de registrar y recuperar el historial total de las operaciones llevadas a cabo con los datos del sistema. Esto incluye saber quién llevó a cabo cada acción, qué datos fueron alterados, cuándo sucedió el cambio y cuál era la condición previa de la información. El comportamiento del sistema puede ser auditado, se pueden identificar anomalías y asegurar la integridad de la información gracias a la trazabilidad, un elemento fundamental de la gobernanza de datos.

En la gestión de inventarios, si no hay mecanismos de auditoría, pueden surgir consecuencias serias. Sin un registro de cambios, no se pueden esclarecer las razones de las diferencias entre el inventario físico y el que aparece en el sistema, determinar si un cambio fue intencionado o por error, ni definir responsabilidades frente a manipulaciones indebidas o pérdidas de datos.

En términos técnicos, por lo general se lleva a cabo la auditoría de operaciones en sistemas de bases de datos utilizando tablas de auditoría o de log, las cuales registran automáticamente cada vez que se modifica, elimina o inserta un registro en las tablas esenciales del sistema. Un registro de auditoría eficaz debe incluir, al menos, los siguientes componentes: el tipo de operación realizada, la tabla o entidad implicada, el identificador del usuario que llevó a cabo la acción, la marca temporal precisa del suceso y los valores antes y después del cambio.

2.2. Visualización de datos y dashboards

La representación gráfica de datos e información a través de componentes visuales como gráficos, mapas o tablas es lo que se conoce como visualización de datos. Los usuarios pueden entender de forma intuitiva tendencias, patrones y anomalías (Few, 2012). Esto es crucial en la gestión de inventarios, ya que los volúmenes de datos históricos de ventas y de movimientos de stock pueden ser abrumadores si se presentan en formato tabular.

2.2.1. Herramientas y tecnologías

Hay numerosas herramientas disponibles para crear paneles de control web interactivos (Tableau,s.f.):

Tabla 1.
Herramientas de desarrollo de dashboards

Herramienta	Tipo	Ventajas
Power BI	Comercial	Potente, integración con Microsoft
Tableau	Comercial	Visualizaciones avanzadas
Chart.js / D3.js	Bibliotecas JS	Personalización total, código abierto
Apache ECharts	Biblioteca JS	Interactividad y gran variedad de gráficos

Autor: Daniel Campoverde. Herramientas especializadas en visualización de datos

2.2.2. Arquitectura de comunicación frontend y backend para dashboards

Cómo el frontend recibe los datos del servidor para representarlos en los elementos visuales es un elemento técnico clave en la creación de dashboards web. Diferentes patrones de comunicación tienen un impacto directo en el rendimiento, la latencia y la experiencia del usuario.

La arquitectura REST es la más frecuente en las aplicaciones web actuales. En este modelo, los clientes hacen solicitudes HTTP a endpoints específicos del servidor, el cual responde con datos en formato JSON. Una API REST tiene que ser sin estado, lo que implica que cada solicitud incluye toda la información requerida para su procesamiento de manera autónoma, sin tener en cuenta el contexto de solicitudes anteriores (Fielding,2000). El sistema creado en el proyecto actual utiliza Axios para que el frontend React haga solicitudes a los endpoints del backend Express, tales como /productos o /ventas/dashboard, con la finalidad de conseguir los datos que se emplean en los elementos gráficos del dashboard.

2.3. Análisis de datos e inteligencia artificial

Según (Han et al., 2012), la inspección, limpieza, transformación y modelado de datos es el proceso que se conoce como análisis de datos. El propósito de este procedimiento es revelar información valiosa, respaldar al tomar decisiones y producir conocimiento. Al implementar técnicas de inteligencia artificial (IA), es posible automatizar actividades como la clasificación, la predicción y la identificación de patrones complejos.

2.3.1. Conceptos de Business Intelligence

Business Intelligence (BI) incluye una serie de métodos, herramientas y aplicaciones orientadas a la recopilación, integración, análisis y exposición de información derivada de datos operacionales, con el objetivo de respaldar la toma de decisiones fundamentada en hechos (IBM,2026). Los elementos tradicionales de BI son:

ETL (Extracción, transformación y carga): Proceso que consiste en obtener datos de diversas fuentes, depurarlos y almacenarlos en un depósito de datos (AWS, 2026).

Almacén de datos (Data warehouse): Repositorio optimizado para consultas analíticas centralizado.

OLAP (Procesamiento Analítico en Línea): Consultas que abarcan múltiples dimensiones, como las ventas diferenciadas por producto, mes y región (AWS,2026).

Informes y tableros de control: Visualización de los indicadores clave de rendimiento.

Un almacén de datos completo puede resultar excesivo en el marco de inventarios para pequeñas y medianas empresas. Es posible crear un Data Mart específico para las ventas y el inventario.

2.3.2. Análisis predictivo en inventarios

El análisis predictivo es una disciplina de la analítica de datos que emplea algoritmos de aprendizaje automático, machine learning, y modelos estadísticos para predecir sucesos futuros a partir de información histórica. El análisis predictivo posibilita en la administración de inventarios:

- Ajustar los niveles de existencias para prever la demanda futura de productos.
- Detectar productos que tienen la tendencia de romperse o sobrealmacenarse.

2.3.3. Modelos de predicción de demanda

Uno de los inconvenientes más analizados en la administración de inventarios es la predicción de la demanda. Los modelos se dividen en (Chen & Guestrin,2016):

Tabla 2.
Modelos de IA para predicción de demanda

Tipo	Modelo	Características
Series temporales	ARIMA, SARIMA	Basados en autocorrelación, requieren estacionariedad
Suavizamiento exponencial	Holt-Winters	Capturan tendencia y estacionalidad
Regresión lineal múltiple	OLS, Ridge, Lasso	Incorporan variables externas (precio, promociones)
Machine learning	Random Forest, XGBoost	Manejan relaciones no lineales

Autor Daniel Campoverde. Modelos de Machine Learning óptimos para predecir demanda

2.4. Antecedentes investigativos

Los antecedentes permiten detectar el estado del arte en relación a sistemas de gestión de inventarios que tienen capacidades visuales y predictivas, además de las carencias presentes que respaldan el proyecto actual.

2.4.1. Soluciones existentes en el mercado

Es conveniente examinar las soluciones disponibles en el mercado para la administración de inventarios antes de describir la propuesta tecnológica del presente trabajo. La Tabla 3 muestra un análisis comparativo de las principales opciones comerciales y SaaS, teniendo en cuenta su tipo de solución, sus funciones principales y las dificultades que enfrentan para ser adoptadas por las pequeñas y medianas empresas (PYMES). Este análisis ayuda a detectar las diferencias que existen y que hacen necesario desarrollar el sistema propuesto.

Tabla 3.
Tabla comparativa entre soluciones existentes para gestión de inventarios

Solución	Tipo	Funcionalidades clave	Limitaciones para PYMES
Oracle NetSuite	Comercial (ERP)	Gestión completa, análisis avanzado, integración contable	Alto costo, complejidad, curva de aprendizaje larga
SAP Inventory Management	Comercial	Control de stock, pronósticos básicos, lotes	Licencias costosas, requiere personal especializado
Zoho Inventory	SaaS (suscripción)	Múltiples almacenes, órdenes de compra, facturación	Sin análisis predictivo de IA, dashboards básicos
TradeGecko (QuickBooks Commerce)	SaaS	Integración con e-commerce, seguimiento en tiempo real	Precio elevado para PYMES, sin modelos predictivos avanzados

Autor: Daniel Campoverde. Comparación de herramientas usadas para gestionar inventarios

Análisis de la brecha:

- Para la mayoría de las pequeñas y medianas empresas (PYMES), no es posible acceder a soluciones comerciales robustas (SAP, Oracle) desde el punto de vista económico.

- Las soluciones SaaS asequibles (Zoho, TradeGecko) carecen de modelos de IA predictiva integrados y dashboards altamente personalizables.

Por ende, el proyecto actual ayuda a saldar esta brecha a través de un sistema web de código abierto (o con una implementación de bajo costo) que incorpora los tres pilares: gestión transaccional, dashboards interactivos, informes inteligentes y pronóstico de demanda basado en inteligencia artificial.

CAPITULO 3. DISEÑO E IMPLEMENTACIÓN

La metodología utilizada para desarrollar el sistema web de gestión de inventarios se detalla en esta sección. Primero, se expone el enfoque metodológico que se ha tomado, explicando por qué es pertinente para un proyecto que fusiona el desarrollo de software tradicional con la inclusión de modelos de inteligencia artificial. En las siguientes secciones, se describen las etapas y prácticas particulares que se aplicaron durante el proceso. La intención es mostrar cómo esa metodología guió la planificación, el diseño y la ejecución del sistema sugerido.

3.1. Metodología de desarrollo

Los métodos de desarrollo establecen las pautas, principios y procedimientos que guían la creación del sistema. Su elección es esencial porque afecta directamente la calidad del producto, los plazos de entrega y la habilidad para adaptarse a los cambios durante el proceso de tesis. Antes de explicar el método que se ha seguido, se muestra a continuación un análisis de los requisitos del sistema, que determina y categoriza las características y funcionalidades que debe tener. Este análisis se divide en dos categorías: los requerimientos funcionales, que detallan las acciones concretas que el sistema debe llevar a cabo; y los requerimientos no funcionales, que determinan las características y limitaciones bajo las cuales el sistema tiene que funcionar.

Tabla 4.
Tabla de Requerimientos Funcionales

Código	Descripción	Prioridad
RF-01	El sistema debe permitir el inicio de sesión de usuarios mediante credenciales, validando su autenticidad antes de otorgar acceso.	Alta
RF-02	El sistema debe gestionar roles de usuario con permisos diferenciados sobre los módulos disponibles.	Alta
RF-03	El sistema debe permitir el registro, edición, consulta y eliminación de productos del inventario, incluyendo nombre, categoría, subcategoría, precio y stock.	Alta

RF-04	El sistema debe permitir el registro de ventas, asociando los productos vendidos, cantidades y montos totales.	Alta
RF-05	El sistema debe actualizar automáticamente el stock disponible de cada producto al registrarse una venta o un ingreso de mercadería.	Alta
RF-06	El sistema debe mostrar un panel de control con indicadores clave como rotación de productos, niveles de stock y ventas totales.	Alta
RF-07	El sistema debe generar alertas automáticas cuando el stock de un producto sea inferior a un umbral mínimo establecido.	Alta
RF-08	El sistema debe generar reportes inteligentes basados en el análisis automatizado de los datos históricos de ventas e inventario.	Alta
RF-09	El sistema debe predecir la demanda futura de productos mediante un modelo de inteligencia artificial entrenado con datos históricos.	Alta
RF-10	El sistema debe mostrar gráficamente la comparación entre la demanda proyectada y el stock disponible por categoría/subcategoría.	Media
RF-11	El sistema debe registrar un historial de auditoría (log) de las operaciones realizadas sobre productos e inventario, indicando usuario, fecha, tipo de operación y valores antes/después del cambio.	Alta
RF-12	El sistema debe permitir exportar reportes de inventario y predicciones en formatos descargables (PDF/Excel).	Baja

Autor: Daniel Campoverde. Requisitos que debe cumplir el sistema en cuanto a funcionalidades

Tabla 5. *Tabla de Requerimientos No Funcionales*

Código	Categoría	Descripción	Prioridad
RNF-01	Rendimiento	El sistema debe responder a las solicitudes del usuario en un tiempo no mayor a 3 segundos bajo condiciones normales de uso.	Alta
RNF-02	Usabilidad	La interfaz debe ser intuitiva y permitir que un usuario sin experiencia técnica pueda operar el sistema tras una breve capacitación.	Alta
RNF-03	Seguridad	El sistema debe cifrar las contraseñas de los usuarios y proteger los endpoints mediante autenticación basada en tokens (JWT).	Alta
RNF-04	Disponibilidad	El sistema debe estar disponible al menos el 95% del tiempo mensual, considerando ventanas de mantenimiento programadas.	Media
RNF-05	Escalabilidad	La arquitectura del sistema debe permitir incrementar el volumen de datos (productos, ventas) sin degradar significativamente el rendimiento.	Media
RNF-06	Mantenibilidad	El código del sistema debe estar modularizado (capas de presentación, lógica de negocio y datos) para facilitar futuras modificaciones.	Alta
RNF-07	Confiabilidad	El sistema debe manejar adecuadamente los errores en la ejecución del modelo de IA, evitando caídas del sistema y mostrando mensajes claros al usuario.	Alta

RNF-8	Trazabilidad	Todas las operaciones críticas sobre el inventario deben quedar registradas de forma permanente para fines de auditoría.	Alta

Autor: Daniel Campoverde. Requisitos operativos que debe cumplir el sistema

3.1.1. Enfoque metodológico

Para el desarrollo de un sistema web de gestión de inventarios que utiliza análisis predictivo, la metodología escogida es ágil e iterativa; además, se han incorporado elementos del método Scrum. Las razones que fundamentan esta decisión son las siguientes:

Condiciones evolutivas: Aunque los propósitos generales están establecidos, es posible perfeccionar las condiciones específicas del sistema a medida que se desarrolla, basándose en la retroalimentación de los usuarios potenciales.

Entregas anticipadas: El método iterativo posibilita que se logren versiones operativas desde las fases iniciales. En el marco de esta tesis, esto implica contar con un prototipo elemental de administración previo a ejecutar módulos más complicados como el análisis predictivo.

Minimización de riesgos: Cuando se trabaja en iteraciones breves, se detectan a tiempo dificultades técnicas o de requisitos, lo cual previene desvíos significativos al concluir el proyecto.

Capacidad de adaptación a la tecnología: Como el proyecto abarca inteligencia artificial para el modelo de predicción de demanda, se pueden probar variados algoritmos en diferentes iteraciones y escoger aquel que muestre un rendimiento óptimo con la información disponible.

Iteraciones planificadas:

Tabla 6. *Tabla de Iteraciones para desarrollo del proyecto*

Iteración	Nombre	Duración estimada	Entregable principal
Iteración 0	Configuración del entorno	1 semana	Entorno de desarrollo listo (React + Vite + Tailwind + Node.js + base de datos)
Iteración 1	Gestión básica de inventario	2 semanas	CRUD de productos, registro de movimientos, consulta de stock
Iteración 2	Dashboards interactivos y KPIs	1 semanas	Visualización de rotación, niveles de stock, alertas automáticas
Iteración 3	Reportes inteligentes	1 semana	Generación automatizada de informes en PDF/Excel
Iteración 4	Módulo de análisis predictivo	2 semanas	Integración del modelo de IA con el backend
Iteración 5	Pruebas y documentación	1 semanas	Informe de pruebas de usabilidad, ajustes finales, documentación técnica

Autor: Daniel Campoverde. Planificación para el desarrollo del sistema propuesto

3.1.2. Herramientas y tecnologías

Para escoger las herramientas tecnológicas, se han tomado en cuenta criterios como la facilidad de implementación, la escalabilidad, la capacidad para integrar modelos de inteligencia artificial, el soporte comunitario y que sean económicas, preferentemente open source. Se describe en detalle, por capas, la pila tecnológica utilizada en el proyecto.

Tabla 7. *Herramientas y Tecnologías escogidas para el desarrollo del proyecto*

Capa	Tecnología	Propósito
Frontend	React	Biblioteca para construir interfaces de usuario interactivas
Frontend (build)	Vite	Empaquetador rápido para desarrollo y producción
Frontend (estilos)	Tailwind CSS	Framework de CSS utilitario para diseño rápido
Frontend (rutas)	React Router DOM	Navegación entre páginas del sistema
Backend	Node.js + Express	Entorno de ejecución y framework web
Base de datos	PostgreSQL	Almacenamiento persistente de datos
Autenticación	JSON Web Token (JWT)	Seguridad y control de acceso
IA (predictiva)	Python + FastAPI	Microservicio para modelos predictivos
IA (librerías)	Pandas, NumPy, scikit-learn, Prophet	Procesamiento de datos y modelos de IA
Control de versiones	Git + GitHub	Gestión del código fuente

Autor: Daniel Campoverde. Herramientas de desarrollo escogidas para desarrollar el sistema

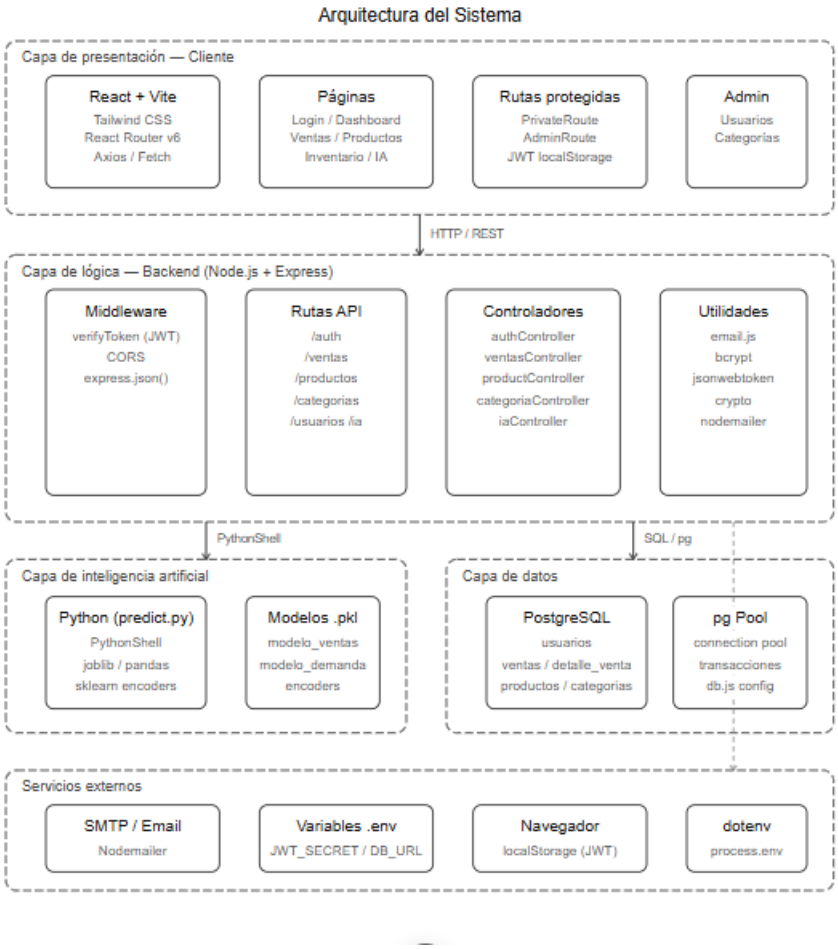
3.2. Arquitectura del sistema

La arquitectura establecida para el sistema web de gestión de inventarios se explica en esta sección. Esta arquitectura es la estructura fundamental que organiza sus módulos, capas y componentes. A continuación, se describen los componentes de esta arquitectura y los estándares técnicos que se han tomado en cuenta para asegurar la escalabilidad, la mantenibilidad y la eficacia del sistema planteado.

3.2.1. Arquitectura general

El sistema creado se basa en una arquitectura en capas (n-tier) que divide las responsabilidades del mismo en niveles claramente definidos, los cuales interactúan entre sí mediante interfaces preestablecidas. Esta división hace que la aplicación sea más fácil de mantener, escalar y asegurar. La arquitectura integral del sistema, dispuesta en cinco capas fundamentales: presentación, lógica de negocio, inteligencia artificial, datos y servicios externos, se expone en la Figura 1.

Figura 1. *Arquitectura del sistema*



Autor: Daniel Campoverde. Composición del sistema

La capa de presentación es el lugar donde se produce la interacción directa del usuario con el sistema. Se ha creado con Vite y React, siendo React la biblioteca fundamental para crear interfaces de usuario a través de componentes que se pueden reutilizar, mientras que Vite es un servidor de desarrollo y una herramienta de construcción que proporciona tiempos de compilación mucho más veloces en comparación con opciones como Create React App. Tailwind CSS, un marco de trabajo

de utilidades CSS que posibilita la creación de interfaces responsivas directamente en el marcado de los componentes, es el encargado de administrar el estilada visual, es decir, como el usuario va a percibir la interfaz. Axios y la API Fetch nativa del navegador se utilizan para comunicarse con el backend, mientras que React Router v6, que implementa el enrutamiento del lado del cliente sin recargar por completo la página, se encarga de la navegación entre vistas.

La aplicación cuenta con los siguientes módulos: el módulo de análisis predictivo, la gestión de ventas, la gestión de productos, el control del inventario, el panel principal y el módulo de autenticación. Un conjunto de rutas protegidas, que se implementan a través de los componentes AdminRoute y PrivateRoute, controla el acceso a estas vistas. Dichos componentes comprueban la existencia y validez del token JWT guardado en el localStorage del navegador antes de dar el acceso. Gracias al componente AdminRoute, las funciones de administración exclusivas, como la gestión de usuarios y categorías, están limitadas únicamente al rol de administrador.

La lógica de negocio se implementa con Node.js y el marco Express, y tiene como función gestionar las peticiones del cliente, poner en práctica las reglas comerciales del sistema y organizar la comunicación con las capas inferiores. Esta capa se organiza en cuatro elementos los cuales son fundamentales dentro del sistema.
El módulo de middleware, que tiene el papel de interceptar todas las solicitudes entrantes, es el primero. El middleware verifyToken comprueba el JWT que se encuentra en el encabezado de autorización de cada petición, asegurando que solamente los usuarios que han sido autenticados tengan acceso a los recursos protegidos. Además, para gestionar qué orígenes están autorizados a utilizar la API, se configura CORS y express.json() para manejar el cuerpo de las solicitudes HTTP.

El segundo elemento son las rutas API, que determinan los endpoints accesibles del sistema clasificados por dominio funcional: /auth para la autenticación, /ventas para administrar transacciones comerciales, /productos para el catálogo de productos, /categorías para categorizar productos, /usuarios para gestionar cuentas y /ia para las proyecciones del modelo de inteligencia artificial.

Los controladores constituyen el tercer elemento y contienen la lógica de negocio particular de cada dominio. El authController se encarga de la verificación del correo

electrónico, el registro, la recuperación de contraseña y el inicio de sesión. El ventasController gestiona el ciclo entero de una operación de venta, lo que comprende la verificación del inventario y el cálculo de los totales. El productController y el categoryController se encargan respectivamente de gestionar las operaciones CRUD de productos y categorías. El iaController funciona como intermediario entre el módulo de inteligencia artificial que se implementó en Python y el backend Node.js.

El cuarto elemento son las utilidades, que reúnen funciones transversales que pueden ser reutilizadas por varios controladores. Estas son: jsonwebtoken, para crear y validar tokens JWT; bcrypt, para encriptar contraseñas de manera segura; crypto, para generar tokens de verificación y restablecimiento de contraseña; y email.js junto con Nodemailer, para transmitir correos electrónicos.

En la arquitectura, la capa de inteligencia artificial funciona semiautónomamente y es invocada por el backend a través de la biblioteca PythonShell. Esta última posibilita que se ejecuten scripts Python desde un entorno Node.js, además de obtener sus resultados. El módulo principal es predict.py, que utiliza joblib para cargar en formato. pkl los modelos de machine learning serializados y efectuar las predicciones con los datos de entrada proporcionados por el controlador iaController. Los modelos que están disponibles son modelo_ventas.pkl y modelo_demanda.pkl. Estos han sido preparados con algoritmos de Random Forest, además de los codificadores requeridos para la conversión de variables categóricas a través de sklearn. Para estructurar y preprocesar los datos de entrada antes de que sean procesados por los modelos, se emplea la biblioteca pandas.

PostgreSQL, un sistema de gestión de bases de datos relacional y de código abierto, es el encargado de manejar la capa de datos. Este sistema guarda toda la información operativa del sistema, que está estructurada en las siguientes tablas: logs, categorías, productos, detalle_venta, ventas y usuarios. A través de pg Pool, el módulo oficial de Node.js para PostgreSQL, que se configura en db.js, se establece la conexión entre la base de datos y el backend. Reciclar conexiones ya existentes en vez de crear una nueva para cada solicitud, gracias a la utilización de un pool de conexiones, aumenta de manera significativa el desempeño del sistema cuando está bajo carga concurrente.

El conjunto de elementos que el sistema utiliza desde el ambiente de ejecución, pero que no son parte del código de la aplicación propiamente dicha, es agrupado en la

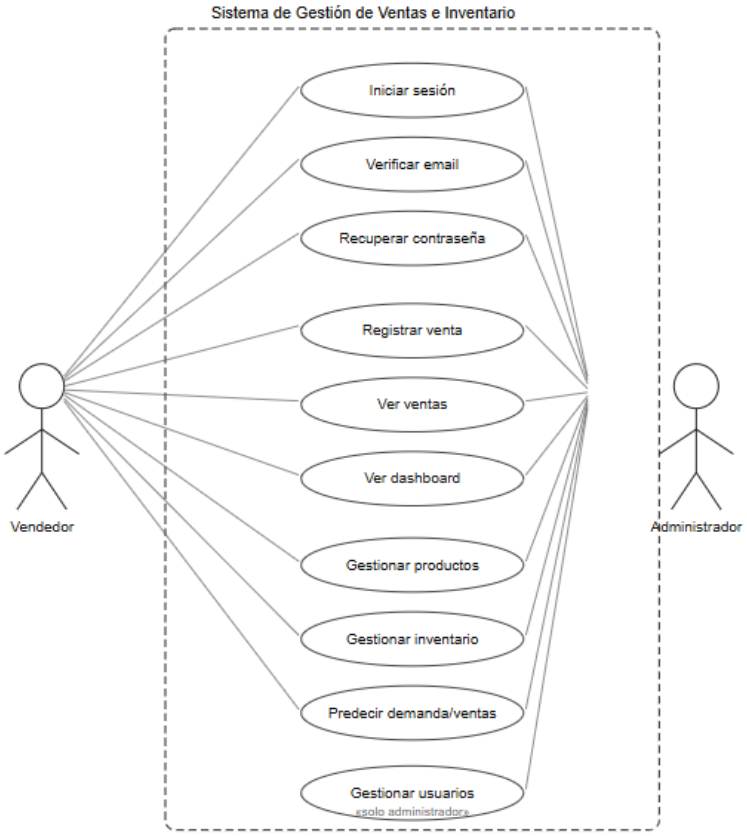
capa de servicios externos. El servicio SMTP de Nodemailer facilita la recuperación de contraseñas y la verificación de cuentas mediante el envío de mensajes electrónicos. Las variables del entorno, que se manejan a través de dotenv y se consultan por medio de process.env, concentran la configuración sensible del sistema para impedir que esas credenciales sean expuestas en el código fuente. El navegador, por otro lado, funciona como servicio de almacenamiento del lado del cliente, ya que conserva el token JWT en localStorage y hace posible que la sesión del usuario se mantenga entre recargas de página.

3.2.2. Casos de uso del sistema

El diagrama de casos de uso que se muestra en la Figura N.2 detalla las interacciones funcionales entre los actores del sistema y las operaciones que brinda, lo cual posibilita que el usuario final vea el alcance funcional del sistema. El Vendedor y el Administrador son los dos protagonistas principales del sistema de administración de inventario y ventas.

El actor Vendedor es el que representa al usuario operativo del sistema, cuyo deber es llevar a cabo las labores diarias vinculadas con la administración comercial y el monitoreo de existencias. En cuanto al actor Administrador, tiene un nivel de privilegio más alto y puede acceder a todas las funcionalidades del sistema, incluidas aquellas que están limitadas y no disponibles para el rol de vendedor.

Figura 2. *Diagrama de casos de uso*



Autor: Daniel Campoverde. Interacción del usuario con el sistema

Las especificaciones detalladas de los casos de uso que forman el diagrama de funcionalidades del sistema se muestran a continuación. Cada tabla explica de forma organizada el flujo principal, las opciones alternativas y los requisitos necesarios para asegurar una interacción adecuada entre los actores y la plataforma, lo que simplifica su implementación y validación.

Tabla 8. *Caso de uso 1*

Campo	Descripción
ID	CU-01
Nombre	Iniciar sesión
Actor	Vendedor / Administrador
Precondición	Usuario registrado y verificado.
Flujo básico	1. Accede a login. 2. Ingresa email y contraseña. 3. Valida credenciales. 4. Genera token. 5. Redirige según rol.
Flujo alternativo	A1. Credenciales incorrectas: mensaje de error y reintento. A2. Cuenta no verificada: redirige a Verificar email.
Postcondición	Sesión activa + token generado y almacenado.
CUs relacionados	Verificar email, Recuperar contraseña

Autor: Daniel Campoverde. Detalle del primer caso de uso del diagrama

Tabla 9. *Caso de uso 2*

Campo	Descripción
ID	CU-02
Nombre	Verificar email
Actor	Vendedor / Administrador
Precondición	Usuario registrado y con correo de verificación recibido.
Flujo básico	1. Recibe correo con enlace. 2. Accede al enlace. 3. Valida token. 4. Marca cuenta como verificada.
Flujo alternativo	A1. Token expirado: permite reenvío. A2. Token inválido: mensaje de error.
Postcondición	Cuenta habilitada para iniciar sesión.
CUs relacionados	Iniciar sesión

Autor: Daniel Campoverde. Detalle del segundo caso de uso del diagrama

Tabla 10.*Caso de uso 3*

Campo	Descripción
ID	CU-03
Nombre	Recuperar contraseña
Actor	Vendedor / Administrador
Precondición	Cuenta registrada con correo válido asociado.
Flujo básico	1. Selecciona "Olvidé mi contraseña". 2. Ingresa su correo. 3. Sistema envía enlace de restablecimiento. 4. Usuario accede y define nueva contraseña. 5. Sistema actualiza la contraseña.
Flujo alternativo	A1. Correo no registrado: mensaje de error. A2. Enlace expirado: solicita generar uno nuevo.
Postcondición	Contraseña actualizada correctamente.
CUs relacionados	Iniciar sesión

Autor: Daniel Campoverde. Detalle del tercer caso de uso del diagrama

Tabla 11. *Caso de uso 4*

Campo	Descripción
ID	CU-04
Nombre	Registrar venta
Actor	Vendedor
Precondición	Sesión iniciada; productos con stock disponible.

Flujo básico	1. Accede al módulo de ventas. 2. Selecciona productos y cantidades. 3. Sistema calcula subtotal, descuentos y total. 4. Vendedor confirma. 5. Sistema registra venta y descuenta stock.
Flujo alternativo	A1. Stock insuficiente: alerta y bloquea confirmación. A2. Cancelación antes de confirmar: no se registran cambios.
Postcondición	Venta almacenada; stock actualizado; auditoría generada.
CUs relacionados	Gestionar inventario, Predecir demanda/ventas

Autor: Daniel Campoverde. Detalle del cuarto caso de uso del diagrama

Tabla 12. *Caso de uso 5*

Campo	Descripción
ID	CU-05
Nombre	Ver ventas
Actor	Vendedor / Administrador
Precondición	Sesión iniciada.
Flujo básico	1. Accede al módulo de ventas. 2. Sistema consulta historial. 3. Sistema presenta listado con filtros por fecha, producto o estado.
Flujo alternativo	A1. Sin ventas registradas: mensaje de "no hay datos disponibles".
Postcondición	Ninguna
CUs relacionados	Registrar venta

Autor: Daniel Campoverde. Detalle del quinto caso de uso del diagrama

Tabla 13. *Caso de uso 6*

Campo	Descripción
ID	CU-06
Nombre	Ver dashboard
Actor	Vendedor / Administrador
Precondición	Sesión iniciada.
Flujo básico	1. Accede al panel principal. 2. Sistema consulta indicadores clave. 3. Sistema presenta información en gráficos y tarjetas.
Flujo alternativo	A1. Error en consulta: mensaje de error con opción de reintentar.
Postcondición	Ninguna (solo consulta).
CUs relacionados	Predecir demanda/ventas

Autor: Daniel Campoverde. Detalle del sexto caso de uso del diagrama

Tabla 14. *Caso de uso 7*

Campo	Descripción
ID	CU-07
Nombre	Gestionar productos
Actor	Administrador
Precondición	Sesión iniciada con permisos en módulo de productos.
Flujo básico	1. Accede al módulo. 2. Selecciona acción 3. Sistema valida datos. 4. Sistema actualiza catálogo.
Flujo alternativo	A1. Datos inválidos: mensaje de error. A2. Eliminar producto con ventas: desactivación lógica en lugar de borrado.
Postcondición	Catálogo actualizado; auditoría generada.
CUs relacionados	Gestionar inventario

Autor: Daniel Campoverde. Detalle del séptimo caso de uso del diagrama

Tabla 15. *Caso de uso 8*

Campo	Descripción
ID	CU-08
Nombre	Gestionar inventario
Actor	Administrador
Precondición	Sesión iniciada; productos previamente registrados.
Flujo básico	1. Accede al módulo. 2. Consulta o actualiza cantidades de stock. 3. Sistema valida valores. 4. Sistema actualiza inventario y genera alerta si stock < mínimo.
Flujo alternativo	A1. Cantidad negativa o inválida: rechaza operación. A2. Stock crítico: notificación automática.
Postcondición	Inventario actualizado; alertas generadas si aplica; auditoría generada.
CUs relacionados	Registrar venta, Predecir demanda/ventas

Autor: Daniel Campoverde. Detalle del octavo caso de uso del diagrama

Tabla 16. *Caso de uso 9*

Campo	Descripción
ID	CU-09
Nombre	Predecir demanda/ventas
Actor	Administrador
Precondición	Sesión iniciada; historial mínimo de ventas disponible.
Flujo básico	1. Accede al módulo de análisis predictivo. 2. Sistema obtiene historial y calcula variables. 3. Sistema ejecuta modelo de IA. 4. Sistema presenta proyecciones y recomendaciones de reabastecimiento.
Flujo alternativo	A1. Error en modelo: mensaje de error y conserva últimos datos disponibles. A2. Historial insuficiente: notifica que predicción puede no ser confiable.
Postcondición	Predicciones y recomendaciones generadas y mostradas.
CUs relacionados	Ver dashboard, Gestionar inventario

Autor: Daniel Campoverde. Detalle del noveno caso de uso del diagrama

Tabla 17. *Caso de uso 10*

Campo	Descripción
ID	CU-10
Nombre	Gestionar usuarios
Actor	Administrador
Precondición	Sesión iniciada con rol de administrador.
Flujo básico	1. Accede al módulo de gestión de usuarios. 2. Selecciona acción. 3. Sistema valida datos. 4. Sistema actualiza información del usuario.
Flujo alternativo	A1. Correo ya existe: mensaje de error. A2. Intenta eliminar su propia cuenta: sistema bloquea la acción.
Postcondición	Usuario creado, actualizado o eliminado; auditoría generada.
CUs relacionados	Iniciar sesión

Autor: Daniel Campoverde. Detalle del décimo caso de uso del diagrama

Los casos de uso detectados se clasifican en cuatro grupos funcionales, dependiendo de su naturaleza:

Control de acceso y autenticación: Este bloque incluye las situaciones de uso de Comprobar correo electrónico, Iniciar sesión y Restablecer contraseña. Al iniciar sesión, ambos actores pueden autenticarse en el sistema con credenciales de contraseña y correo electrónico, a cambio de lo cual obtienen un token JWT que les permite acceder a los recursos protegidos. El proceso de comprobar el correo electrónico se refiere a confirmar la cuenta a través de un vínculo que se envía al correo del usuario en el momento del registro, lo que asegura que la dirección de correo electrónico proporcionada es válida.

Administración comercial: Este bloque reúne los casos de uso Ver ventas y Registrar venta. Desde el punto de vista de las transacciones, registrar una venta es la operación fundamental del sistema. Esto permite al vendedor generar una nueva venta vinculada a uno o varios productos, verificar automáticamente el stock disponible y actualizar en tiempo real el inventario mediante una única transacción en la base de datos.

Examen y vigilancia: Esta sección incluye los casos de uso "Visualizar dashboard", "Gestionar inventario" y "Predecir la demanda/ventas". El dashboard proporciona a los dos actores una visión integrada de los indicadores clave del desempeño empresarial,

como el total de ventas, los artículos más vendidos, las alertas de stock crítico y las tendencias según el período. La administración del inventario facilita la consulta y la actualización de los niveles de existencias de los productos que constan en el sistema, además de ofrecer visibilidad sobre el stock actual comparado con los límites mínimo y máximo establecidos. Para prever la demanda/ventas, el módulo de inteligencia artificial del sistema se activa. Envía al modelo entrenado en Python los datos históricos de las ventas para proyectar las futuras y calcular las estimaciones de demanda por cada producto. Esto respalda la planificación del reabastecimiento y las compras.

Gestión del sistema: Este segmento incluye el caso de uso "Gestionar productos", que está disponible para los dos actores, y "Gestionar usuarios", que es solo para el administrador. El manejo de productos posibilita la creación, consulta, actualización y eliminación de productos en el catálogo, así como la asignación de precios, niveles de inventario, categorías y otros atributos del producto. El único caso de uso con acceso exclusivo para el rol de Administrador es gestionar usuarios, como se señala claramente en el diagrama con la anotación "solo administrador". Esta característica posibilita la creación de nuevas cuentas de usuario, la modificación de sus datos, la activación o desactivación de cuentas y la asignación de roles en el sistema, asegurando que se ejerza un control centralizado sobre quién tiene acceso a la plataforma y con qué nivel de privilegio.

La diferencia entre los dos actores indica el modelo de control de acceso por roles que se ha instaurado en el sistema, en el cual los permisos no se conceden individualmente a cada usuario, sino según la función que este ejerce dentro de la organización. Esta aproximación simplifica la gestión de permisos y disminuye el peligro de otorgar accesos no autorizados a características críticas del sistema.

3.3. Diseño de base de datos

El diseño de la base de datos es el cimiento estructural del sistema como se detalla en el ANEXO A, lo que asegura que la información administrada sea coherente, íntegra y accesible. Se utilizó PostgreSQL como gestor de base de datos relacional para construir el sistema de gestión de inventarios con análisis predictivo basado en inteligencia artificial, dado que este cuenta con una sólida estructura, permite la implementación de lógica empresarial a través de triggers y procedimientos almacenados.

El modelo de datos fue creado con la finalidad de eliminar redundancias y garantizar la dependencia funcional entre los atributos, de acuerdo con los principios de la tercera forma normal (3FN). Cada entidad del modelo se explica a continuación:

3.3.1. Entidades Principales

En la tabla de usuarios se guardan los datos de los participantes del sistema, distinguiendo entre dos roles: el de administrador y el de vendedor. Incorpora campos de seguridad como el token que verifica el correo electrónico, la contraseña encriptada y los campos para restablecer la contraseña utilizando un token temporal con fecha de caducidad. Además, incluye información de contacto como teléfono, cédula y dirección.

La tabla categorías posibilita que los productos se agrupen de manera lógica, lo que simplifica la navegación y el filtrado en el sistema. Cada categoría tiene un nombre único y puede incluir una descripción si se desea.

El inventario tiene como núcleo la tabla de productos. Incluye características como el nombre, la descripción, el precio, el stock actual, el stock mínimo y máximo, el SKU (código único del producto), la localización física en la bodega y su estado (si está activo o inactivo). El campo sub_categoria fue añadido con el fin de integrar el módulo de inteligencia artificial. Este campo facilita la clasificación de cada producto dentro de las subcategorías que el modelo predictivo ha identificado: Tables (Mesas), chairs (Sillas), furnishings (Muebles) y bookcases (Libreros).

Cada operación comercial es anotada en la tabla de ventas, que incluye al cliente, el medio de pago, la condición de la venta (si está cancelada, pendiente o finalizada) y a quien la realizó. Para nutrir el modelo de inteligencia artificial con datos contextuales, se añadieron los campos shipping_days (días de envío), region (región geográfica), segment (segmento del cliente) y ship_mode (modalidad de envío). Estas variables representan las entradas del modelo predictivo.

La tabla detalle_venta contiene información sobre los productos que se venden, incluyendo el número de unidades, el precio por unidad, el subtotal y la rebaja aplicada. Este último campo es igualmente uno de los elementos de entrada del modelo predictivo.

La tabla logs utiliza un sistema de auditoría sofisticado que se basa en el tipo de dato UUID para los identificadores únicos y JSONB para guardar la situación previa y posterior de cada registro que se modifica. Esto posibilita que todas las operaciones de

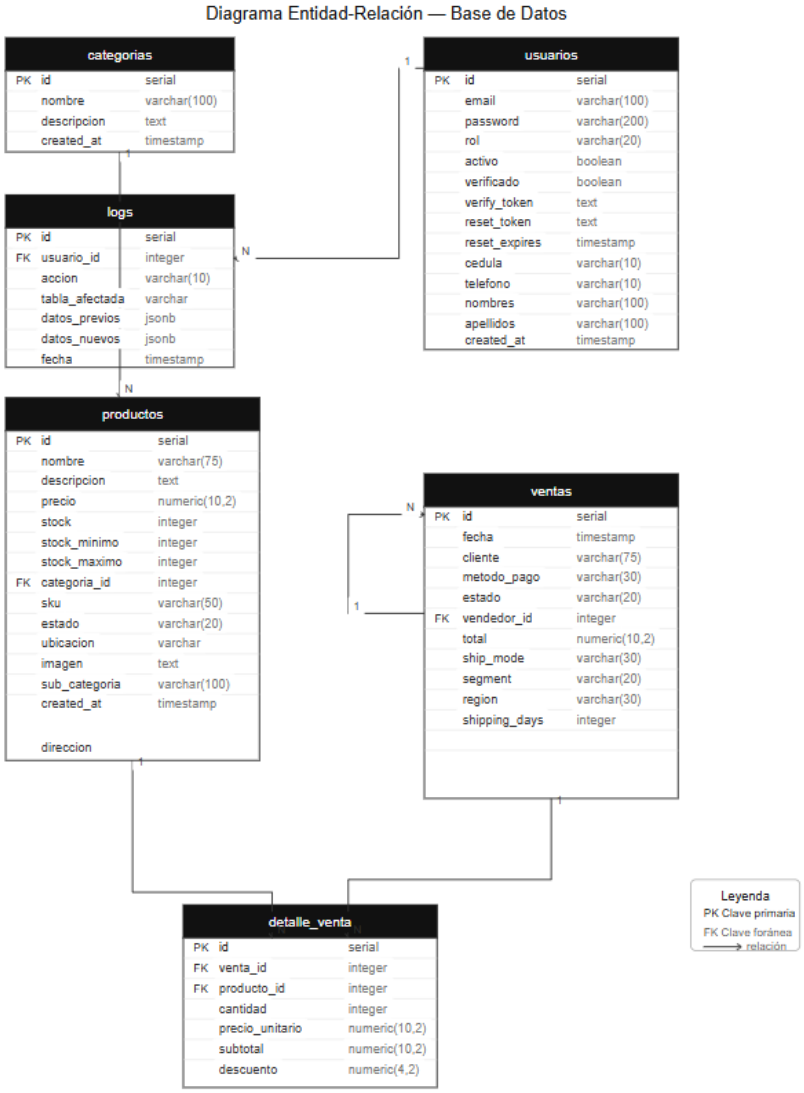
inserción, actualización y eliminación realizadas con respecto a las entidades críticas del sistema sean completamente rastreadas.

Tabla 18.*Reglas de Integridad*

#	Tipo	Tabla	Campo(s)	Regla
1	Entidad	usuarios	id	PK única y no nula
2	Entidad	categorias	id	PK única y no nula
3	Entidad	productos	id	PK única y no nula
4	Entidad	ventas	id	PK única y no nula
5	Entidad	detalle_venta	id	PK única y no nula
6	Entidad	logs	id	PK única y no nula
7	Referencial	productos	categoria_id	FK → categorias.id
8	Referencial	ventas	vendedor_id	FK → usuarios.id
9	Referencial	detalle_venta	venta_id	FK → ventas.id
10	Referencial	detalle_venta	producto_id	FK → productos.id
11	Referencial	logs	usuario_id	FK → usuarios.id
12	Dominio	usuarios	email	Formato válido
13	Dominio	usuarios	activo	TRUE / FALSE
14	Dominio	usuarios	verificado	TRUE / FALSE
15	Dominio	usuarios	rol	Roles definidos por sistema
16	Dominio	usuarios	cedula	Única
17	Dominio	productos	precio	> 0
18	Dominio	productos	stock	≥ 0
19	Dominio	productos	stock_minimo	≥ 0
20	Dominio	ventas	total	≥ 0
21	Dominio	detalle_venta	cantidad	> 0
22	Dominio	detalle_venta	precio_unitario	> 0
23	Dominio	detalle_venta	descuento	≥ 0
24	Negocio	ventas	total	

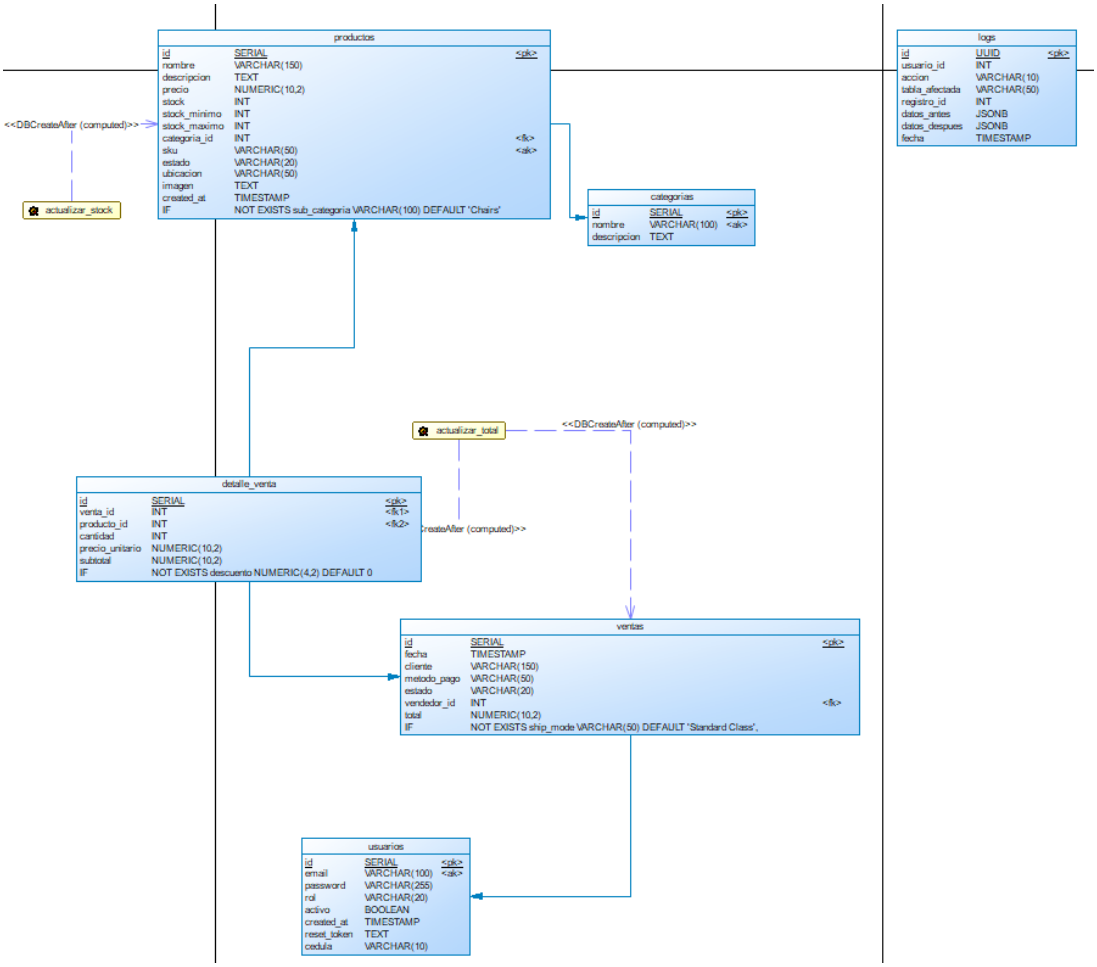
Autor: Daniel Campoverde. Reglas para priorizar integridad de datos

Figura 3. *Diagrama Entidad-Relación de la base de datos*



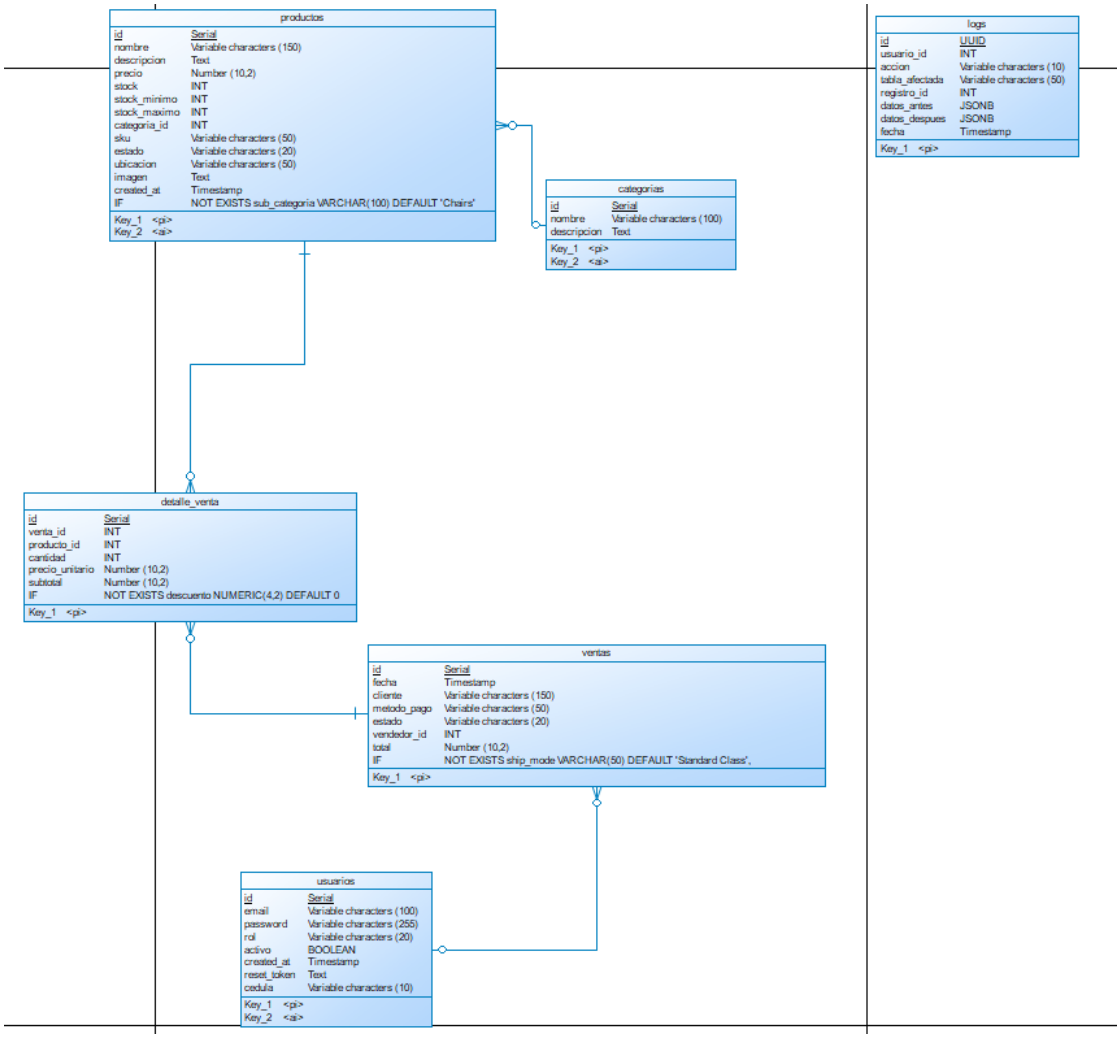
Autor: Daniel Campoverde. Tablas de la base de datos

Figura 4. *Modelado Lógico de la base de datos*



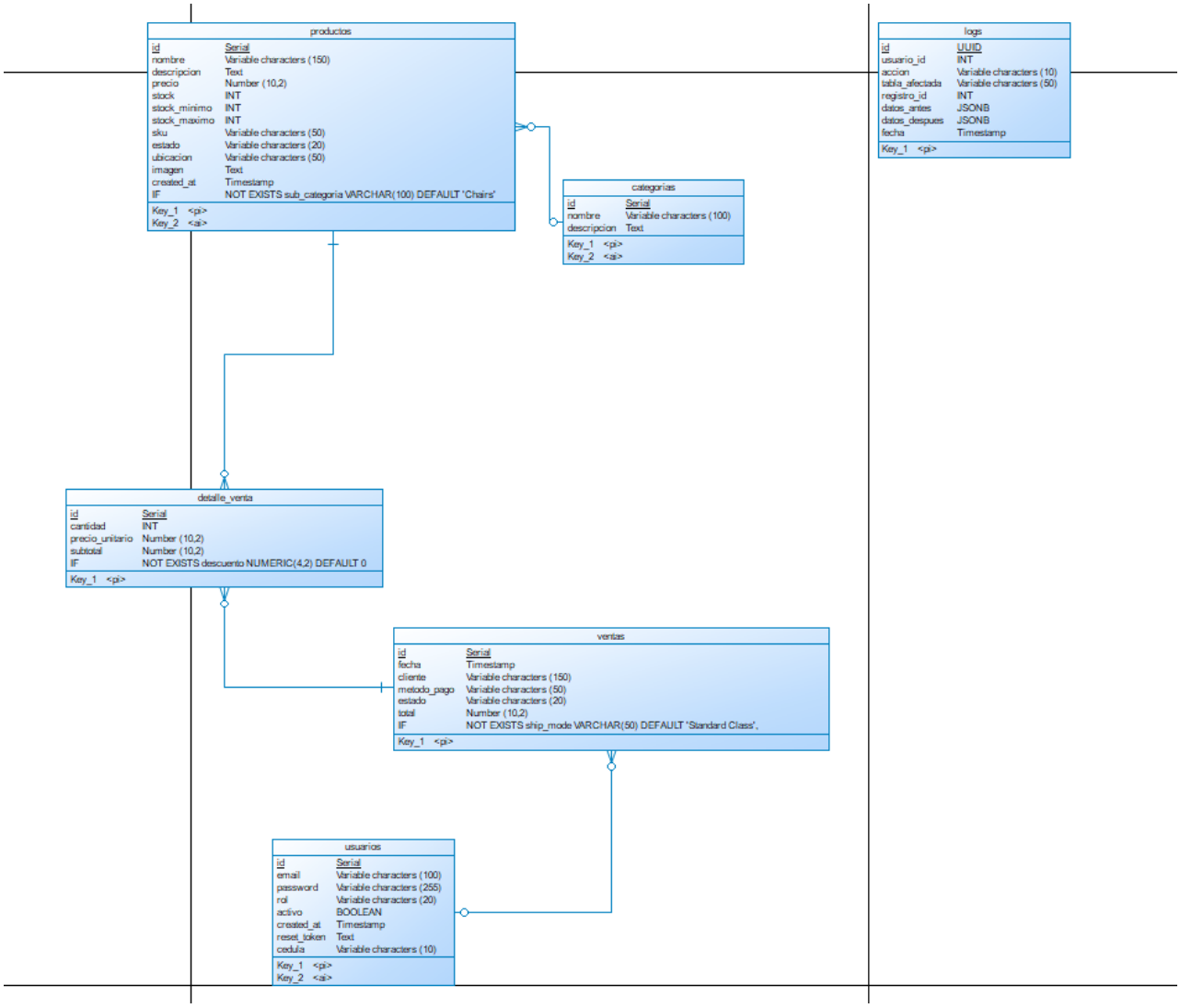
Autor: Daniel Campoverde. Modelado de la base de datos

Figura 5. *Modelado físico de la base de datos*



Autor: Daniel Campoverde. Modelado de la base de datos

Figura 6. *Modelado conceptual de la base de datos*



Autor: Daniel Campoverde. Modelado de la base de datos

3.3.2 Automatizaciones mediante triggers

Se pusieron en marcha tres triggers con el propósito de automatizar la lógica de negocio directamente en la base de datos y así disminuir la carga en la capa de aplicación:

• **trg_stock:** Tras cada inserción en detalle_venta, se lleva a cabo y deduce automáticamente del stock el número de unidades vendidas del producto correspondiente.

• **trg_total:** Se lleva a cabo tras cada inserción en detalle_venta y vuelve a calcular el total de la venta, sumando los subtotales de todos los artículos.

• **trg_logs_productos, trg_logs_ventas y trg_logs_detalle:** Mediante la función get_current_user_id(), guardan en la tabla logs el ID del usuario activo, registrando cualquier operación de INSERT, UPDATE o DELETE sobre las tablas primarias.

3.4. Desarrollo de módulos funcionales

El sistema fue creado empleando una estructura de tres niveles: la presentación (frontend, hecha en React.js), la lógica de negocio (backend, realizada con Node.js y Express) y la persistencia de datos (PostgreSQL). Esta división de responsabilidades hace más fácil incorporar nuevas funciones, realizar mantenimiento y escalar. Se describen los módulos funcionales que se han puesto en funcionamiento a continuación.

3.4.1. Gestión de Productos

Los usuarios tienen la posibilidad de gestionar el catálogo total de artículos del inventario mediante el módulo de gestión de productos. A través de la interfaz web, el usuario tiene la posibilidad de generar, modificar, observar y borrar productos mediante un formulario modal que realiza validaciones en tiempo real.

Los siguientes atributos están incluidos en cada artículo registrado: nombre, descripción, SKU, precio, stock actual, stock mínimo y máximo, localización en bodega, categoría y subcategoría de inteligencia artificial. Este último apartado, que en la interfaz se presenta con el rótulo "subcategoría IA", posibilita categorizar el producto en las categorías que el modelo predictivo identifica. Las alternativas disponibles son: Tables (Mesas), chairs (Sillas), furnishings (Muebles) y bookcases (Libreros).

El módulo muestra un panel de estadísticas en tiempo real que incluye: el total de artículos registrados, el monto total del inventario (que se obtiene sumando el precio por unidad de cada artículo), la cifra de productos cuyo stock está por debajo del mínimo y el

volumen total de ventas relacionadas. La tabla de productos contiene filtros por categoría y nombre, lo que facilita una navegación rápida en catálogos extensos.

El controlador productController.js, en el backend, se encarga de las operaciones CRUD (crear, leer, actualizar y eliminar) a través de consultas SQL parametrizadas que evitan inyecciones SQL. La consulta de lectura ejecuta un JOIN con la tabla de categorías para devolver el nombre de cada categoría junto a su producto correspondiente.

3.4.2. Movimientos de Inventario

Las transacciones de inventario se administran automáticamente mediante el procedimiento de registro de ventas. El trigger trg_stock actualiza de manera instantánea y atómica el stock cuando el cliente registra una venta con sus productos asociados, lo que asegura la integridad de los datos aún en situaciones de acceso simultáneo.

El módulo de ventas posibilita la anotación de transacciones que involucren varios productos, la aplicación de descuentos por producto, el cálculo automático de subtotales y el total. El sistema comprueba si el stock disponible es suficiente para satisfacer la cantidad solicitada antes de validar la transacción. Si no lo es, se notifica al usuario y se rechaza la operación.

Cada transacción contiene información contextual logística que es importante para el análisis predictivo: modalidad de envío, segmento del cliente, zona geográfica y estimación de los días de envío. La fecha de la transacción, los productos comercializados y estos datos juntos constituyen el conjunto de variables de entrada que nutre al modelo de inteligencia artificial.

3.4.3. Alertas Automáticas

El sistema pone en marcha un procedimiento de alertas automáticas que se fundamenta en la comparación entre el stock presente de cada producto y su stock mínimo establecido. Mientras se carga el módulo de productos y el panel de control principal, este control se lleva a cabo en tiempo real.

En la vista de productos, cada producto tiene un indicador visual de color que transmite intuitivamente el estado del stock: verde si es suficiente, amarillo si es menor al mínimo fijado y rojo si está agotado. Además, el contador de productos en estado crítico se puede ver en el panel de estadísticas.

La tarjeta Productos Críticos del dashboard principal muestra en tiempo real la cantidad de productos cuyo inventario actual es menor a lo que el modelo de inteligencia artificial estima que se demandará, fusionando así el análisis predictivo con el control tradicional del inventario. Si este valor supera cero, el indicador aparece en rojo, lo que supone una advertencia inmediata para el administrador del sistema.

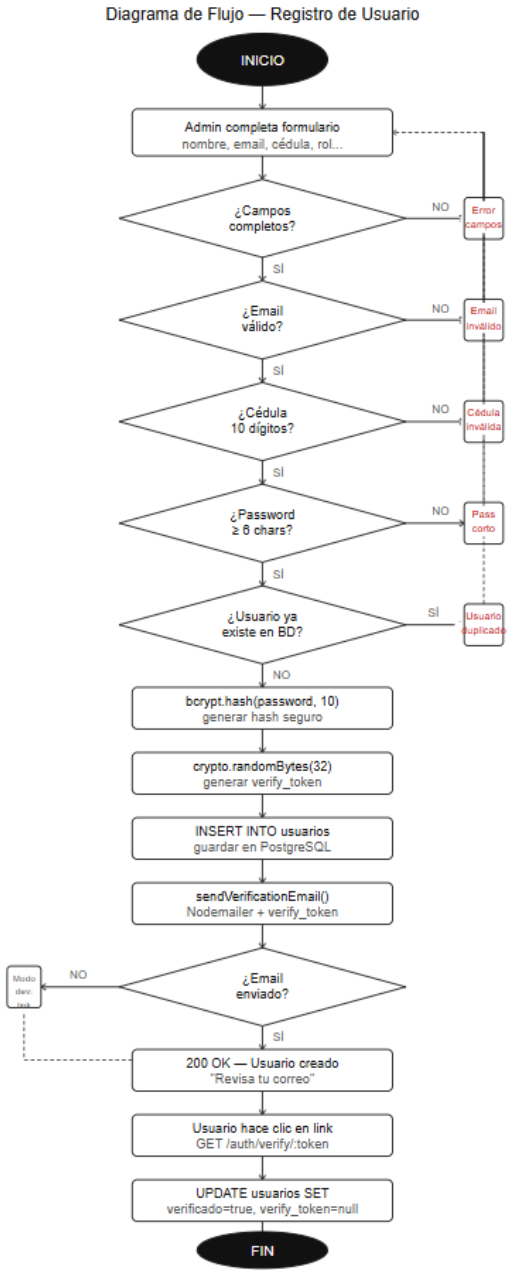
3.5. Diagramas Técnicos del sistema

Con la finalidad de ilustrar gráficamente el comportamiento, la estructura y las interacciones del sistema sugerido, se muestran en esta sección los diagramas técnicos que se han creado durante la etapa de diseño. Estos diagramas, que utilizan la notación UML, posibilitan una visualización nítida de los diversos componentes del sistema, las conexiones entre sus partes y los flujos de interacción entre los usuarios y la aplicación. De esta manera, se facilita el entendimiento de su estructura y funcionamiento interno.

3.5.1 Flujo de Inicio de Sesión

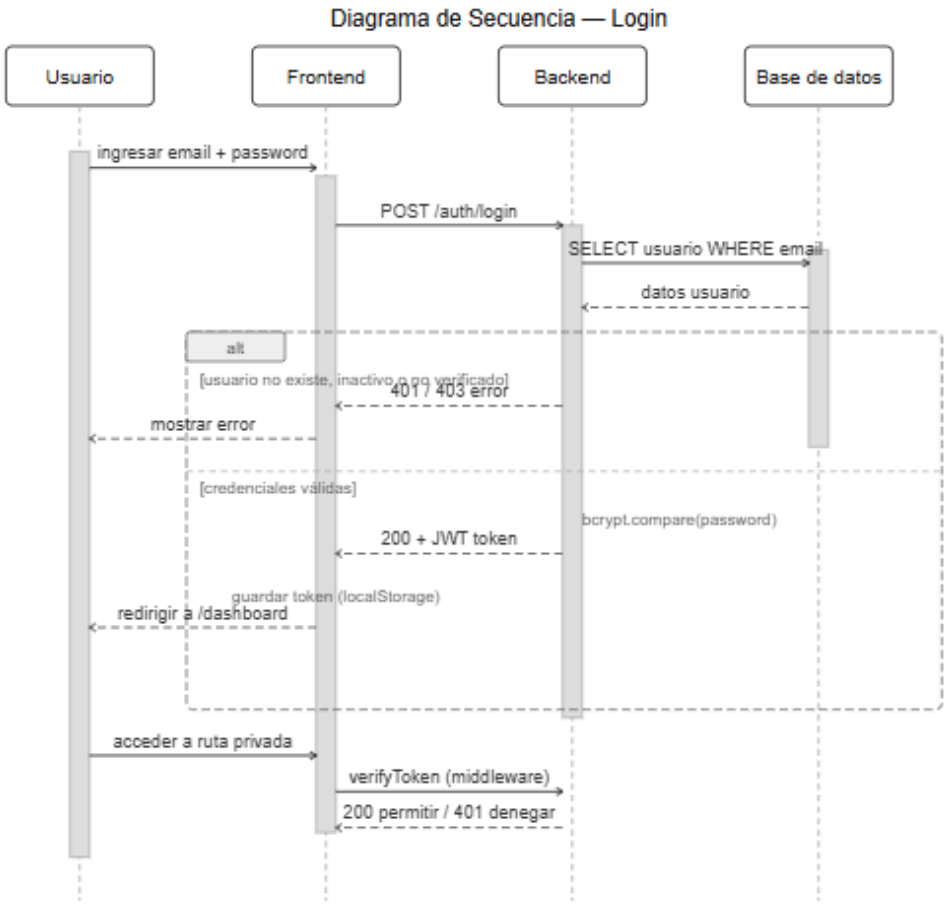
El diagrama que se presenta a continuación describe el procedimiento de inicio de sesión del sistema. En él, se muestra la secuencia de interacciones entre el usuario y la aplicación, desde que el usuario ingresa sus credenciales hasta que es validado y tiene acceso al sistema; además, incluye cómo manejar posibles errores durante la autenticación.

Figura 7. *Flujo de Inicio de Sesión*



Autor: Daniel Campoverde. Proceso de inicio de sesión

Figura 8. *Diagrama de secuencia-Login*



Autor: Daniel Campoverde. Funcionamiento interno del proceso de login

Desde que se llena el formulario para crear una cuenta hasta que se valida correctamente la dirección de correo electrónico, el diagrama de flujo mostrado en la Figura N.4 ilustra todo el procedimiento de registro de un nuevo usuario en el sistema. Este flujo se llevó a cabo en el módulo authController.js del backend, y abarca tanto el camino exitoso como las diversas etapas de validación que tienen la capacidad de detener el procedimiento si los datos son incorrectos o inconsistentes.

El procedimiento comienza cuando el cliente del sistema llena el formulario de registro, que incluye la información del nuevo usuario: nombres, apellidos, dirección de correo electrónico, número de cédula, teléfono, dirección, contraseña y rol otorgado. Cuando se envía el formulario, el backend realiza una serie de validaciones encadenadas; cada verificación debe ser superada exitosamente antes de pasar a la siguiente. El sistema devuelve un mensaje de error detallado al cliente y el procedimiento se interrumpe si alguna de estas validaciones no es satisfactoria. Esto posibilita que el administrador rectifique los datos introducidos y lo intente nuevamente.

La validación inicial comprueba que no haya campos vacíos en el formulario y que todos los campos requeridos estén presentes. Si falta algún campo necesario, se produce inmediatamente una respuesta de error sin procesar los demás datos. La segunda validación verifica que el formato del correo electrónico sea válido, lo cual requiere como mínimo la existencia del símbolo "@" en él. La validación número tres utiliza una expresión regular que requiere que la cédula y el teléfono estén formados exactamente por diez cifras numéricas, desechando cualquier valor que incluya letras, caracteres especiales o una longitud diferente. La cuarta validación comprueba que la contraseña sea de al menos seis caracteres, lo que asegura un nivel básico de seguridad en las credenciales de acceso.

Una vez que se han completado todas las validaciones de formato, el sistema verifica si ya hay un usuario registrado con la misma dirección de correo electrónico o cédula consultando la base de datos PostgreSQL. Como el correo electrónico y la cédula son identificadores únicos por naturaleza, esta comprobación es esencial para asegurar que los registros en el sistema sean únicos. El sistema, si detecta un duplicado, emite un error que señala que el usuario ya está registrado y no procede a crear el registro.

Si se superan todas las validaciones y no se encuentran entradas duplicadas, el procedimiento pasa a la etapa de creación del registro. Para empezar, la contraseña se procesa con la función bcrypt.hash() en diez rondas de coste, creando así un hash criptográfico seguro que es el único dato guardado en la base de datos. En todo momento, no se registra la contraseña original en ninguna parte del sistema. En segundo lugar, se crea un token de verificación único a través de crypto.randomBytes(32), que genera una cadena hexadecimal de 64 caracteres con alta entropía, lo que hace prácticamente imposible su reproducción o predicción por fuerza bruta.

El sistema ejecuta la instrucción INSERT INTO usuarios en PostgreSQL con el hash de contraseña y el token de verificación creados, guardando todos los datos del nuevo usuario junto con el token de verificación. En esta etapa, el usuario queda registrado en la base de datos, pero el campo verificado tiene un valor falso. Esto hace que no pueda iniciar sesión hasta que termine la verificación de su correo electrónico.

Después, el sistema llama a la función sendVerificationEmail(). Esta función crea y manda un correo electrónico a la dirección registrada; el mensaje contiene un enlace de verificación que incluye el token generado. El diagrama incluye un camino alternativo

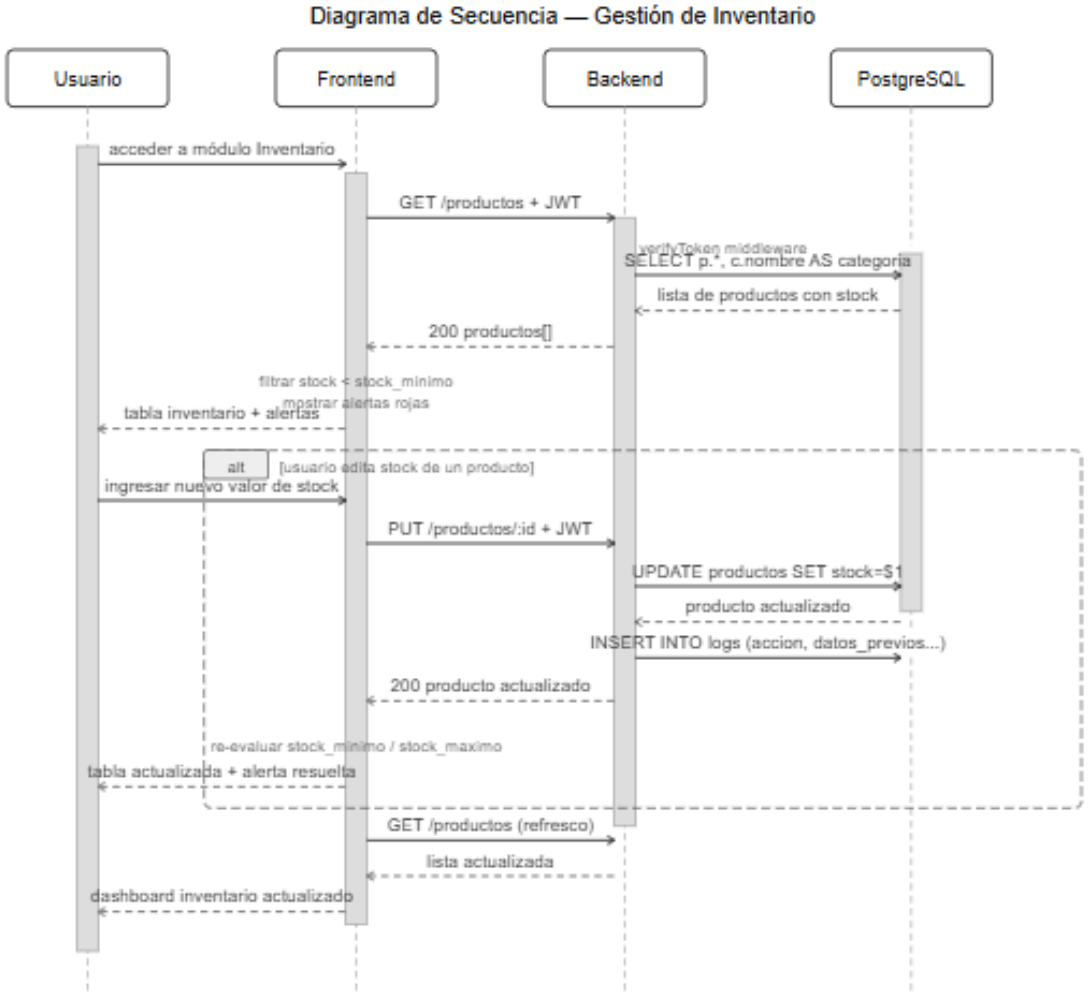
para el entorno de desarrollo si el servidor SMTP no está disponible o configurado. En este caso, el sistema crea localmente el enlace de verificación y lo devuelve en la respuesta de la API, lo que posibilita que el desarrollador finalice el proceso de verificación sin necesitar un servidor real de correo. La variable emailResult.dev en la respuesta del controlador es la que regula este comportamiento.

El sistema devuelve una respuesta HTTP 200 al cliente con el mensaje de confirmación después de que se ha enviado el correo. Esto indica que la cuenta fue creada correctamente y que el usuario debe abrir su correo para activarla. Para ello, necesita hacer clic en el enlace de verificación que recibió por correo electrónico, lo cual genera una solicitud GET al endpoint /auth/verify/:token. El backend identifica al usuario que corresponde con el token recibido y lleva a cabo una instrucción UPDATE que coloca el campo verificado en verdadero y suprime de la base de datos el token de verificación, lo cual deja la cuenta totalmente activa y disponible para ser empleada. El usuario únicamente tiene la posibilidad de autenticarse con éxito en el sistema a partir de este instante.

3.5.2 Flujo de Gestión de Inventario

A continuación, se presenta el diagrama que ilustra el flujo de gestión del inventario. En él se muestra la sucesión de acciones e interacciones que tienen lugar al realizar un registro, una actualización o una consulta de los productos en el sistema; también se evidencian las validaciones y respuestas producidas en cada uno de estos procesos.

Figura 9. *Diagrama de Secuencia- Gestión de Inventario*



Autor: Daniel Campoverde. Proceso interno del sistema del proceso de Inventario

El flujo de interacción entre los elementos del sistema durante la actualización y consulta del inventario es representado por el diagrama de secuencia que se presenta en la Figura N.6. La solicitud GET al endpoint /productos del backend, que incluye el token JWT en el encabezado de autorización, se activa cuando el usuario accede al módulo de inventario desde la interfaz de navegación.

El middleware verifyToken, que valida el token antes de permitir que el controlador acceda, es el encargado de interceptar la petición en el backend. Una vez que la solicitud ha sido validada, el productController lleva a cabo una consulta SQL para obtener todos los productos de la tabla "productos". Para añadir el nombre de cada categoría, hace un JOIN con la tabla "categorias". La lista que se obtiene es devuelta al frontend en formato JSON con el código HTTP 200.

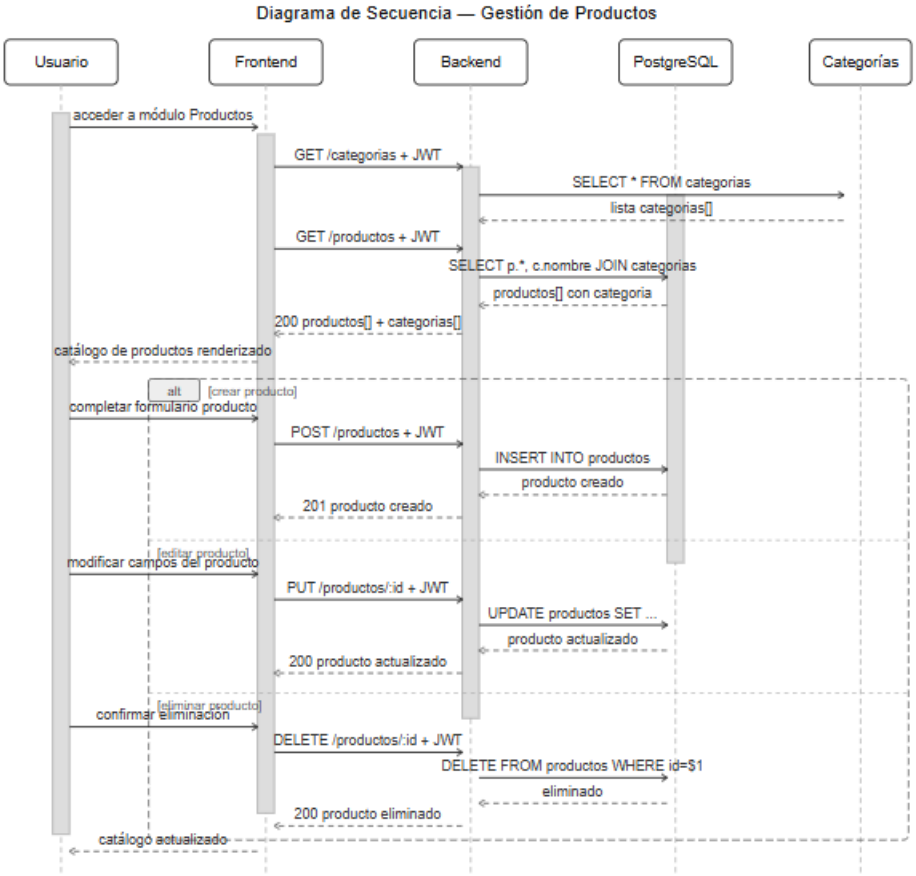
El frontend lleva a cabo el procesamiento de la respuesta al comparar el valor presente del campo stock de cada producto con el stock mínimo establecido. Los productos cuyo inventario está por debajo del límite mínimo se identifican visualmente a través de alertas, lo que posibilita que el usuario reconozca rápidamente qué artículos necesitan ser reabastecidos. Asimismo, se identifican los productos que exceden el stock máximo, lo que indica situaciones de sobrestock.

El diagrama incluye, mediante un bloque alternativo, el flujo de actualización del stock, que se inicia cuando el usuario toma la decisión de cambiar el nivel de existencias de un artículo. El cliente introduce el nuevo valor stock a través de la interfaz, lo que produce una solicitud PUT al endpoint /productos/:id utilizando el token JWT. El backend emplea un comando UPDATE para modificar el registro pertinente en PostgreSQL, y luego introduce la operación en la tabla logs con los datos anteriores y los nuevos del producto, asegurando así que se pueda rastrear totalmente el cambio. Por último, el frontend actualiza la lista de productos para identificar las modificaciones y volver a evaluar las alertas de stock crítico con los nuevos valores.

3.5.3 Flujo de Gestión de Productos

A continuación, se muestra el diagrama que ilustra el flujo de gestión de productos. En él, se representa la secuencia de interacciones requeridas para registrar, editar, consultar y eliminar artículos en el sistema, además de las validaciones que se llevan a cabo en cada operación.

Figura 10. *Diagrama de Secuencia- Gestión de Productos*



Autor: Daniel Campoverde. Proceso interno del sistema para la gestión de productos

La interacción de los elementos del sistema a través de las operaciones básicas de creación, lectura, actualización y eliminación de registros se muestra en el diagrama de secuencia para el módulo Gestión de Productos. Los elementos y actores involucrados incluyen al Frontend, el usuario, el Backend y la base de datos PostgreSQL, que se encuentra estructurada con una tabla de categorías asociada.

Cuando se accede al módulo, el sistema lleva a cabo una carga inicial de datos a través de dos solicitudes HTTP autenticadas con token JWT. La primera solicitud obtiene la lista de categorías disponibles y la segunda adquiere los productos mediante un query relacional que fusiona la tabla de productos con la tabla de categorías. Cuando el usuario llena el formulario correspondiente para registrar un nuevo producto, se envía una solicitud POST al Backend. Este lleva a cabo una instrucción de inserción en la base de datos, procesa la solicitud y devuelve una respuesta que tiene el código HTTP 201, lo cual confirma que se ha creado el recurso con éxito.

Cuando se modifica un producto existente, el usuario edita los campos requeridos y el Frontend envía una solicitud PUT que identifica el recurso por su ID. El Backend lleva a cabo la actualización en la base de datos y devuelve un código HTTP 200, lo cual señala que la operación ejecutada se realizó con éxito, sin ningún tipo de incoveniente. Por último, para borrar un producto, el sistema pide al usuario una confirmación explícita antes de continuar. Después de ser confirmada, se envía un pedido DELETE que contiene el identificador del producto. El backend lleva a cabo la instrucción de eliminación en la base de datos y devuelve una respuesta HTTP 200. Cuando el Frontend recibe esta confirmación, actualiza el catálogo para mostrar las modificaciones que se han llevado a cabo.

3.6. Implementación de Dashboards Interactivos

El componente principal para visualizar el sistema es el tablero interactivo, que concentra en una sola pantalla la información histórica de ventas, la situación actual del inventario y las proyecciones producidas por el modelo de inteligencia artificial. La representación de gráficos dinámicos se llevó a cabo utilizando la biblioteca Recharts y React.js.

3.6.1. Definición de KPIs

Los siguientes indicadores clave de rendimiento (KPIs) se definieron para el tablero, los cuales fueron elegidos por su importancia en la toma de decisiones en pequeñas y medianas empresas:

Tabla 19. *KPIs presentados en el sistema*

KPI	Descripción	Fuente de datos
Ventas Totales	Suma acumulada de todas las ventas completadas	Tabla ventas
Stock Total	Suma del stock de los productos registrados	Tabla productos
Demanda IA	Unidades estimadas a vender según el modelo predictivo	Módulo de IA
Productos Críticos	Productos con stock menor a la demanda estimada por la IA	Cruce IA + inventario
Predicción de Ventas	Monto estimado de ventas para la próxima transacción	Módulo de IA

Autor: Daniel Campoverde. Indicadores de rendimiento del sistema

3.6.2. Diseño de Interfaces

Se basó en los principios de claridad visual y de usabilidad para diseñar la interfaz del panel. Se estableció un sistema de colores que varía según la clase de información: azul para las estadísticas financieras y del inventario, verde para los indicadores de demanda y rojo para las advertencias graves. En la parte de arriba, las tarjetas de KPIs se muestran con iconos descriptivos que se toman de la biblioteca Lucide React.

A través de dos tarjetas de gran tamaño con un fondo sólido, azul para las ventas pronosticadas y verde para la demanda prevista, se distinguen visualmente las predicciones de la inteligencia artificial, lo cual permite separarlas claramente de los datos históricos. Esta distinción visual tiene como objetivo informar al usuario sobre la naturaleza calculada de esos valores.

La interfaz tiene un botón de actualización manual que posibilita al usuario obligar a recargar todas las predicciones y datos, lo cual es útil en ambientes en los que se registran ventas con frecuencia elevada. Un indicador animado sustituye el contenido durante la recopilación de datos para comunicar el estado de carga.

3.6.3. Visualización de datos

El dashboard incorpora cuatro elementos visuales que han sido creados con Recharts:

• Gráfico de área (AreaChart): Presenta la progresión de las ventas reales por mes, incluyendo un último punto que simboliza la estimación de la IA para la venta siguiente.

Con esta visualización, es posible cotejar la estimación del modelo con el comportamiento a lo largo de la historia.

• Gráfica de barras en horizontal (BarChart): Confronta el stock actual de los cinco productos principales con la demanda calculada por inteligencia artificial, lo que permite detectar rápidamente qué artículos necesitan reabastecimiento.

• Gráfica circular (PieChart): Facilita la detección de productos que tienen más peso en el inventario al mostrar cómo se distribuye porcentualmente el stock entre los productos registrados.

Todos los elementos gráficos son responsivos debido a la implementación de ResponsiveContainer de Recharts, lo cual les permite ajustarse automáticamente al tamaño de pantalla del aparato.

3.7. Implementación del Módulo de Análisis Predictivo

La parte que distingue al sistema es el módulo de análisis predictivo, el cual utiliza métodos de aprendizaje automático para prever cómo serán las ventas y la demanda de productos en el futuro. La implementación de scripts de Python a través de la biblioteca python-shell de Node.js permite que este módulo se conecte con el backend.

3.7.1. Selección de variables

Las variables de entrada que se emplean son: el modo de envío, el segmento del cliente, la ciudad, el estado, la región, la categoría del producto, la subcategoría, el descuento aplicado, el año, mes y día, así como si es fin de semana o no y los días que tarda en llegar. Antes del entrenamiento, se transformaron las variables categóricas a través de Label Encoding.

El dataset Stores Sales Forecasting dentro del anexo F, un conjunto de datos público muy empleado en estudios de análisis de ventas y pronósticos en el sector del comercio minorista, fue usado para la capacitación de los modelos. El conjunto de datos incluye registros de órdenes de venta con variables que caracterizan al cliente, el producto, la logística y los resultados financieros de cada operación.

Tabla 20. *Columnas del dataset de entrenamiento*

Campo	Tipo original
Row ID	int
Order ID	object (string)

Order Date	datetime64
Ship Date	datetime64
Ship Mode, Customer ID, Customer Name, Segment, Country, City, State, Postal Code	object/int
Region	object
Product ID, Product Name	object
Category	object
Sub-Category	object
Sales	float64
Quantity	int64
Discount	float64
Profit	float64

Autor: Daniel Campoverde. Dataset usado para el entrenamiento de los modelos predictivos

Para el presente sistema, el modelo fue entrenado con los registros correspondientes a la categoría Furniture, delimitación que permite obtener predicciones especializadas y métricas de evaluación más precisas dentro de este dominio específico.

Antes de entrenar los modelos, se llevó a cabo un procedimiento de depuración y acondicionamiento de la información en el conjunto original de transacciones que pertenecían a la categoría Furniture del conjunto de datos Store Sales Forecasting. Primero, se corroboró la calidad de los datos, asegurándose de que no hubiera valores nulos ni registros duplicados y de que las fechas (Order Date y Ship Date) estuvieran correctamente tipificadas. Para posibilitar su manipulación temporal futura, estas fueron transformadas al formato datetime. La información se unió a una granularidad mensual, sumando las ventas, el número vendido y el descuento promedio por mes, subcategoría de producto y región geográfica, puesto que la base de datos original estaba a nivel de transacción individual.

Para no perder información relacionada con periodos sin ventas registradas, se rellenaron las combinaciones de región, mes y subcategoría que no estaban en los datos agregados. A estas combinaciones se les asignó un valor de cero, asegurando de este modo series temporales continuas para cada combinación. Luego, con el propósito de incluir en el modelo la conducta histórica reciente como información predictiva, se crearon variables de rezago (lag) y de promedio móvil para las ventas y para la cantidad

demandada. Se excluyeron del conjunto de datos las observaciones de los primeros lapsos de cada serie, ya que no tenían un historial adecuado para calcular estas variables.

Por último, las variables categóricas (región y subcategoría) fueron codificadas con números utilizando Label Encoding, lo que dejó el conjunto de datos preparado para la fase de modelado.

La selección de variables se llevó a cabo manualmente, basándose en el criterio del dominio del problema. Se definieron un total de nueve variables predictoras: la subcategoría y región codificadas, el año y mes en que se realizó la observación, las variables de rezago y promedio móvil de cantidad y ventas, además del descuento promedio mensual. Se eliminaron del modelo aquellas variables que no tenían valor predictivo directo o eran identificatorias, como los identificadores de fila, orden, cliente y producto; así como los campos textuales (nombre del cliente, nombre del producto, ciudad, estado y país). No se utilizó para esta última una técnica algorítmica o estadística formal para seleccionar las variables.

3.7.2. Preparación del conjunto del entrenamiento

Los pasos que se siguieron en el proceso de entrenamiento son:

• Procesamiento y carga del conjunto de datos: transformación de fechas, obtención de elementos temporales (indicador del fin de semana, día, mes, año y día de la semana) y determinación del número de días para el envío como la diferencia entre las fechas de pedido y despacho.

• Codificación de variables categóricas a través de Label Encoding, guardando los codificadores entrenados en un archivo llamado encoders.pkl para que puedan ser utilizados nuevamente en el módulo de predicción.

3.7.3. Ajuste de hiperparámetros

Para asegurar la reproducibilidad de los resultados, el dataset se divide en dos partes: un 80% para entrenamiento y un 20% para prueba, utilizando una semilla aleatoria fija (random_state=42).

Formación de los modelos Random Forest con 200 estimadores (árboles de decisión).

Evaluación del rendimiento a través de las métricas MAE (Error absoluto medio), RMSE (Raíz del error cuadrático medio) y R^2 (coeficiente de determinación).

La serialización de los modelos entrenados en archivos.pkl a través de la biblioteca joblib, para luego ser cargados en el módulo de predicción.

Durante la etapa de validación, se obtuvieron las siguientes métricas:

Tabla 21.*Métricas de evaluación del modelo de IA*

Modelo	MAE	RMSE	R^2
Predicción de Ventas	733.44	1232.34	0.348
Predicción de Demanda	6.35	9.68	0.543

Autor: Daniel Campoverde. Parámetros resultantes del modelo de entrenamiento

Las cifras obtenidas demuestran que el modelo de demanda explica el 54.3% del cambio en la demanda mensual total por subcategoría y región, mientras que el de ventas abarca el 34.8% del cambio en las ventas. Se compararon los resultados de los dos modelos con un modelo de referencia ingenuo (naive forecast), el cual emplea el valor del mes anterior como predicción, con el fin de verificar su verdadero poder predictivo. Un R^2 de 0.127 para la demanda y un R^2 negativo de -0.278 para las ventas se registraron con este modelo de referencia, lo que corrobora que un método simplista no es capaz de captar los patrones conductuales de los datos.

En contraposición, el modelo de inteligencia artificial ha superado constantemente al baseline en error absoluto (MAE) y en capacidad explicativa (R^2), lo que evidencia que el modelo es capaz de captar relaciones significativas entre las variables categóricas, temporales y de historial de ventas que un método de pronóstico simple no puede detectar. Estos hallazgos sostienen que el modelo es una herramienta adecuada para ayudar a decidir cuándo reponer inventario en empresas pequeñas y medianas, aunque se admite que existe un margen de error debido a factores externos que no están contemplados en el conjunto de datos, como las campañas de mercadotecnia, los cambios en la economía global o los acontecimientos estacionales no estructurados.

El módulo de predicción, predict.py, obtiene los modelos y los codificadores serializados, recibe la información de entrada en formato JSON desde el backend de Node.js por medio de argumentos en línea de comandos, emplea los encodificadores originales para transformar las variables categóricas y devuelve al proceso padre las predicciones en formato JSON.

3.7.4. Comparación entre modelos

Se realizó un proceso de evaluación comparativa con diversos algoritmos de regresión que se emplean extensamente en problemas de pronóstico: Gradient Boosting, Regresión de Support Vector, Árboles de decisión, K-Nearest Neighbors, Regresión lineal y Random Forest. Se entrenaron todos los modelos con la misma serie de variables predictivas y se evaluaron en la misma división cronológica del conjunto de datos, dividiendo el 80% de las observaciones más viejas para el entrenamiento y el 20% más nuevo para la prueba, para que durante el proceso de aprendizaje ningún modelo pudiera acceder a información futura, todo este proceso mencionado se presenta dentro del anexo G.

Las tres métricas complementarias que se utilizaron para calcular el rendimiento de cada modelo fueron la Raíz del Error Cuadrático Medio (RMSE), el Error Absoluto Medio (MAE) y el Coeficiente de Determinación (R^2). Se emplearon de modo independiente en las predicciones de ventas y demanda.

Tabla 22. *Comparación modelos en predicción de ventas*

Modelo	MAE	RMSE	R^2
Random Forest	773.44	1232.34	0.348
Linear Regression	890.06	1347.07	0.221
KNN Regressor	960.32	1488.32	0.049
SVR (RBF)	1057.20	1764.06	-0.336
Decision Tree	1160.61	1887.83	-0.530

Autor: Daniel Campoverde. Evaluación de modelos

Tabla 23. *Comparación modelos en predicción de demanda*

Modelo	MAE	RMSE	R^2
Random Forest	6.35	9.68	0.543
Linear Regression	8.09	11.18	0.391
Decision Tree	7.83	12.14	0.281
KNN Regressor	10.78	15.44	-0.163
SVR (RBF)	10.72	16.61	-0.346

Autor: Daniel Campoverde. Evaluación de modelos

Por las razones que se indican a continuación, se eligió el algoritmo Random Forest Regressor:

- Gestión simultánea eficaz de variables numéricas y categóricas.

• Resistencia ante valores atípicos, outliers, que se encuentran en la información de ventas reales.

• Capacidad para detectar conexiones no lineales entre las variables de entrada y las que se persiguen.

• Interpretabilidad elevada a través del análisis de la relevancia de las características, feature importance.

• Debido al funcionamiento del ensamblaje de varios árboles de decisión, el riesgo de sobreajuste, overfitting, es bajo.

Se desarrollaron dos modelos por separado: uno para pronosticar las ventas, en dólares, y otro para prever la demanda, en unidades. Ambos modelos comparten el mismo conjunto de variables de entrada, pero optimizan objetivos distintos.

3.8. Implementación de Reportes Inteligentes

Se detalla en esta sección la puesta en marcha de los informes inteligentes del sistema. Primero se describe cómo trabaja el análisis automatizado, que une los datos de ventas pasados con las proyecciones del modelo de inteligencia artificial. Después, se explica el proceso de detección de anomalías, por medio del cual el sistema identifica y comunica al usuario las situaciones críticas en stock según la demanda proyectada.

3.8.1. Análisis Automatizado

El sistema produce análisis automatizados de forma constante al combinar datos reales de ventas con las predicciones del modelo de inteligencia artificial. El sistema obtiene la última venta registrada con la categoría de Muebles, construye el vector de características de entrada y ejecuta el modelo predictivo cada vez que se carga o actualiza el tablero. Luego, muestra los resultados en los elementos visuales del tablero.

El análisis automatizado comprende la organización de las ventas históricas mensuales, el cálculo de métricas acumuladas como la demanda proyectada, el stock y las ventas totales y la detección de productos con un stock actual que es menor a la demanda estimada, todo esto sin requerir intervención del usuario.

3.8.2. Detección de Anomalías

La comparación en tiempo real entre el stock de cada producto y la demanda proyectada por el modelo de inteligencia artificial permite identificar anomalías en el inventario.

Cuando la cantidad de un producto es inferior a lo que se ha previsto que se va a demandar, se considera que está en una situación crítica. Esto se le comunica al usuario por medio de:

• En el tablero de control, en la tarjeta "Productos Críticos", se muestra un indicador numérico que se destaca en rojo cuando su valor es superior a cero.

• Código de colores en la tabla de productos: verde si el stock es suficiente, amarillo si está por debajo del mínimo establecido y rojo si no hay stock.

• En el gráfico de la relación entre stock y demanda IA se incluye una barra comparativa, que facilita la identificación visual de cuáles son los productos con más diferencia entre lo disponible y lo estimado en términos de demanda.

3.9. Pruebas y Validación

La fase de pruebas y validación tuvo como finalidad comprobar que el sistema opera correctamente en sus distintos elementos, asegurando que los módulos funcionales y la integración entre capas respondan como se espera en una variedad de contextos de uso.

3.9.1. Pruebas Funcionales

La fase de pruebas y validación tuvo como finalidad comprobar que el sistema opera correctamente en sus distintos elementos, asegurando que los módulos funcionales y la integración entre capas respondan como se espera en una variedad de contextos de uso.

Tabla 24. *Pruebas Funcionales del sistema*

N°	Caso de prueba	Resultado esperado	Resultado obtenido	Estado
1	Registro de nuevo producto con sub_categoria	Producto almacenado con todos los campos	Producto creado correctamente	Aprobado
2	Creación de venta con múltiples productos	Stock decrementado automáticamente por trigger	Stock actualizado correctamente	Aprobado

3	Intento de venta con stock insuficiente	Sistema rechaza la operación con mensaje de error	Mensaje de error mostrado al usuario	Aprobado
4	Alerta de stock bajo	Producto resaltado en amarillo/rojo en la tabla	Código de color correcto según umbral	Aprobado
5	Predicción IA desde el dashboard	Retorno de ventas_predichas y demanda_predicha	Valores numéricos retornados correctamente	Aprobado
6	Registro de logs de auditoría	Cada operación registrada en tabla logs	Logs generados con datos antes/después	Aprobado

Autor: Daniel Campoverde. Pruebas de funciones del sistema

3.9.2. Pruebas de Integración

Se comprobó la correcta comunicación entre las diferentes capas del sistema mediante las pruebas de integración: frontend (React.js), backend (Node.js / Express), módulo de inteligencia artificial (Python / scikit-learn) y base de datos (PostgreSQL).

Se validaron los siguientes flujos de integración:

• Frontend → Backend → Base de datos: Se comprobó que los controladores Express procesan de manera adecuada las peticiones HTTP hechas desde los componentes React y que, con integridad referencial, se guardan y obtienen resultados desde PostgreSQL.

• Backend → Módulo de inteligencia artificial → Backend: Se comprobó que el backend genere de manera adecuada el vector de entrada en formato JSON, que lo envíe a predict.py a través de python-shell y que gestione la respuesta en JSON para mostrarla al frontend.

• Base de datos → Triggers → Srock: Se comprobó que al añadir registros en detalle_venta, los triggers trg_stock y trg_total se activan de forma correcta, lo cual permite la actualización del inventario y el total de la venta de una manera atómica.

CAPITULO 4. ANÁLISIS DE RESULTADOS

Los resultados que se lograron tras la puesta en marcha del sistema web para gestionar inventarios se presentan en este capítulo. Primero se ofrece la evaluación global del sistema con respecto a los objetivos propuestos; luego se describen las pruebas de usabilidad que se llevaron a cabo, y, por último, se presentan las restricciones detectadas durante el desarrollo del proyecto.

4.1. Evaluación del sistema

Se llevó a cabo una evaluación exhaustiva del sistema, abarcando los seis módulos funcionales que se implementaron: gestión de productos, autenticación, análisis predictivo con inteligencia artificial, inventario, ventas y categorías. El propósito fue comprobar que el sistema satisface las exigencias establecidas para el entorno de una PYME, considerando tanto la corrección funcional como la respuesta en situaciones de error.

Módulo de autenticación. El sistema lleva a cabo un ciclo de vida de la cuenta de usuario en su totalidad. El registro crea un hash bcrypt de la contraseña antes de almacenar el registro, envía un email de verificación por medio de la API Resend con un token único que contiene 64 caracteres hexadecimales y no permite el inicio de sesión hasta que el usuario confirme su identidad. El proceso de recuperación de contraseña genera un token que tiene una ventana de caducidad de 15 minutos. En las pruebas, se comprobó que los intentos de inicio de sesión con cuentas no verificadas, desactivadas o contraseñas equivocadas devuelven los códigos HTTP correspondientes, 401 y 403 respectivamente sin revelar información delicada del sistema. El procedimiento de soft delete en los usuarios asegura que las ventas históricas vinculadas a un vendedor desactivado mantengan su integridad de referencia.

Módulo de categorías y productos. El CRUD de productos funciona bien bajo las limitaciones de rol que se han establecido a través del middleware isAdmin. Cuando se llaman los puntos finales de escritura (PUT, POST, DELETE) usando un token de rol vendedor, devuelven el código 403. La consulta de listado ejecuta un JOIN implícito con la tabla de categorías y devuelve el nombre de la categoría vinculada a cada producto,

evitando así que el frontend tenga que hacer llamadas adicionales para establecer la relación. El módulo de categorías funciona de manera similar y también es exclusivo para el rol de administrador.

Módulo del inventario. La categorización del estado (normal/bajo/agotado) se determina durante el renderizado, al comparar el campo stock con el stock mínimo. Estos límites son ajustables por producto. Se comprobó a lo largo de las pruebas que no es necesario volver a cargar la página para que el estado de alerta que se muestra en la interfaz cambie con la actualización del inventario por medio de PUT.

Módulo de ventas. Desde la perspectiva de la integridad de los datos, el registro de una venta es la transacción más importante del sistema. La puesta en marcha emplea transacciones SQL explícitas (BEGIN / COMMIT / ROLLBACK): en primer lugar, introduce la cabecera en la tabla de ventas, después recorre los detalles insertando cada fila con su correspondiente subtotal en detalle_venta y por último actualiza el inventario de cada producto que se ve afectado. Si alguno de estos pasos no funciona ya sea porque el stock es insuficiente, el producto no existe o se ha violado una clave foránea, la transacción total retrocede y ninguna fila permanece. Se comprobó esta garantía durante las pruebas de funcionalidad.

Módulo para el análisis predictivo. El sistema incorpora dos modelos de machine learning, modelo_ventas.pkl y modelo_demanda.pkl, que han sido entrenados con el algoritmo GradientBoostingRegressor y un conjunto RandomForest, los cuales han sido serializados utilizando joblib. El script predict.py, mediante python-shell, recibe un payload JSON proveniente del controlador Node.js. Luego, implementa los encoders que se almacenaron durante la capacitación en las columnas categóricas, genera el DataFrame en el orden preciso que requiere el modelo y devuelve las proyecciones de ventas predichas y demanda predicha. El endpoint GET /api/ia/analisis realiza múltiples invocaciones al modelo para generar el historial comparativo, las estimaciones para los próximos seis meses y el análisis por categoría con métricas calculadas sobre datos.

Tabla 25. *Estado funcional de cada módulo*

Módulo	Restricción de rol	Integridad de datos	Estado
Autenticación	—	Hash + token firmado	Operativo
Productos	isAdmin en escritura	Clave foránea a categorías	Operativo
Categorías	isAdmin completo	—	Operativo
Inventario	isAdmin en PUT	Umbrales por producto	Operativo
Ventas	verifyToken en todos	Transacción SQL	Operativo
Análisis IA	verifyToken	Encoders del entrenamiento	Operativo

Autor: Daniel Campoverde. Funcionalidad de cada parte del sistema

4.2. Pruebas de usabilidad

Los resultados que se lograron tras la puesta en marcha del sistema web para gestionar inventarios se presentan en este capítulo. En primer lugar, se muestra la evaluación global del sistema en función de los objetivos establecidos; luego, se especifican las pruebas de usabilidad que se llevaron a cabo.

4.2.1. Diseño de pruebas

Las pruebas de usabilidad se elaboraron con el propósito de valorar la experiencia del usuario en el sistema bajo condiciones que son representativas del perfil de una PYME. Las pruebas se organizaron en cinco contextos de uso que abarcan los flujos esenciales del sistema:

Escenario 1: Ingreso al sistema**.** Se debe terminar el proceso de registro, confirmando el correo electrónico y acceder a una cuenta.

Escenario 2: Administración del catálogo. El administrador tiene la obligación de crear un nuevo producto con todos sus campos, modificar el valor de uno ya existente y eliminar un producto del catálogo.

Escenario 3: Registro de una transacción de venta. El vendedor tiene que registrar en una venta tres productos diferentes, asignar cantidades y comprobar la exactitud del

total calculado antes de confirmar. Se evalúa si el participante termina la venta sin incurrir en fallos de stock o de formulario.

Escenario 4: Supervisión del inventario. Utilizando los filtros disponibles, señalar los productos con stock bajo o inexistente y después actualizar el stock de uno de ellos desde el modal de edición.

Escenario 5: Consultar el módulo de inteligencia artificial. Acceder al módulo de IA, determinar las categorías con la discrepancia más grande entre el stock presente y la demanda proyectada, y revisar la recomendación de mayor prioridad que ha generado el sistema.

4.2.2. Resultados

Los resultados de las pruebas de usabilidad realizadas sobre los cinco escenarios establecidos fueron positivos en todos los módulos del sistema que se evaluaron.

Escenario 1: Ingreso al sistema. Se finalizó satisfactoriamente el flujo de registro, verificación de correo e inicio de sesión. Se logró terminar el procedimiento sin problemas, enfatizando la claridad del formulario de registro y la recepción instantánea del correo de verificación que se mandó a través de la API de Resend. El sistema tuvo una respuesta adecuada a todos los errores que fueron probados: bloqueo de cuentas sin verificar, rechazo de correos electrónicos duplicados y caducidad del token para recuperar la contraseña.

Escenario 2: Administración del catálogo (perfil de administrador). En todos los casos de prueba, la creación, edición y desactivación de productos se realizó sin inconvenientes. El perfil de administrador tuvo la capacidad de llenar el formulario del nuevo producto con todos sus campos requeridos y confirmar que el registro se mostraba de manera instantánea en la lista. El perfil de vendedor no pudo acceder a las opciones de escritura del catálogo debido a que la restricción de rol isAdmin funcionó apropiadamente. La administración de categorías operó de manera similar, con el CRUD completo únicamente accesible para el administrador.

Escenario 3: Registro de una transacción de venta. En todas las situaciones evaluadas, el módulo de ventas funcionó adecuadamente. Antes de confirmar la transacción, los participantes comprobaron que el sistema calculaba correctamente el total general y el subtotal por ítem al registrar ventas con varios productos. El proceso de transacción SQL (BEGIN/COMMIT/ROLLBACK) fue validado con éxito: cuando se

intentó registrar una venta con un stock insuficiente en uno o más productos del detalle, el sistema no solo negó la operación entera, sino que además no persistió ninguna fila parcial en la base de datos, garantizando así la integridad del inventario en cada momento.

Escenario 4: Supervisión del inventario. El módulo de inventario presentó de manera adecuada la categorización del estado de cada producto, en función de sus umbrales establecidos. Los filtros por categoría y estado operaron de manera adecuada, lo cual posibilitó que los participantes detectaran instantáneamente los productos críticos. La actualización del stock desde el modal de edición se mostró inmediatamente en la lista y en el estado de alerta del producto, sin que fuera necesario volver a cargar la página. Todos los participantes reconocieron las alertas visuales sin problemas.

Escenario 5: Consultar el módulo de inteligencia artificial. El módulo de análisis predictivo cargó adecuadamente los datos producidos por el pipeline ML. Los datos incluyen un historial comparativo de las ventas reales en comparación con lo que pronosticó el modelo, la proyección de la demanda para los próximos seis meses junto a su intervalo de confianza, un análisis por categoría con el espacio entre la demanda proyectada y el stock vigente, así como recomendaciones priorizadas generadas dinámicamente. Los participantes detectaron adecuadamente las categorías con mayor déficit proyectado y leyeron las sugerencias de prioridad alta sin requerir la interpretación de los datos en bruto. Las métricas del modelo se calcularon adecuadamente utilizando los datos auténticos del sistema.

4.3. Limitaciones del proyecto

Las siguientes restricciones que afectan el alcance presente del proyecto y guían el trabajo futuro fueron detectadas durante la creación e implementación del sistema:

Dependencia de los datos de entrenamiento. Los modelos de aprendizaje automático se entrenaron utilizando un conjunto de datos de tipo Superstore que contiene columnas geográficas y de segmentación comercial, características del mercado estadounidense. Como la tabla de ventas no guarda información geográfica a nivel de estado o ciudad, los campos "City" y "State" se fijan con valores constantes cuando el endpoint GET /api/ia/analisis genera payloads basándose en las ventas reales que están registradas en el sistema. Esto genera una inclinación sistemática en las predicciones, la cual se reduce parcialmente debido al peso relativo de estos campos en el modelo.

Latencia del módulo de análisis predictivo. El endpoint GET /api/ia/analisis ejecuta numerosas invocaciones en secuencia al script Python predict.py a través de python-shell: una para cada mes histórico que haya, seis llamadas para las proyecciones futuras y una por cada categoría que se encuentre en la base de datos. Cada vez que se invoca Python, se empieza un proceso distinto en el que los modelos y codificadores se cargan desde el disco. Esto representa alrededor de 23 llamadas consecutivas para un sistema con cinco categorías y doce meses de historial, lo cual puede resultar en tiempos de respuesta que oscilen entre 8 y 20 segundos, según el hardware del servidor.

CAPITULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

La creación de un sistema web para la administración de inventarios en pequeñas y medianas empresas (PYMES) utilizando análisis predictivo basado en inteligencia artificial posibilitó lograr las metas propuestas al inicio del estudio, fusionando técnicas de aprendizaje automático con tecnologías actuales de desarrollo web en una solución coherente y eficaz. Se exponen a continuación las conclusiones que surgieron del proceso de desarrollo, ejecución y validación del sistema.

Primero, se verificó que es factible incorporar un módulo de inteligencia artificial en un sistema web para manejar inventarios, empleando una estructura de tres capas construida con React.js, Node.js/Express y PostgreSQL. Se consiguió la comunicación entre el modelo predictivo y el backend a través de la biblioteca python-shell, que posibilita ejecutar scripts Python desde Node.js y devolver resultados en formato JSON, sin tener que desplegar un microservicio autónomo. Esta perspectiva simplificó la complejidad arquitectónica y hizo más sencillo el mantenimiento del sistema.

Como otro resultado, se determinó que el algoritmo Random Forest Regressor es una alternativa apropiada para prever la demanda y las ventas en el ámbito de los datos de comercio minorista. Su habilidad para gestionar variables categóricas codificadas y numéricas al mismo tiempo, además de su solidez frente a la existencia de valores atípicos, lo estableció como una opción más avanzada que los modelos lineales simples para esta clase de datos. Los modelos capacitados con el conjunto de datos Stores Sales Forecasting, que se limitó a la categoría de muebles, alcanzaron un MAE de 283.48 dólares en las predicciones de ventas y un MAE de 1.84 unidades en las predicciones de

demanda; estos resultados son considerados adecuados para tomar decisiones operativas en PYMES.

Como una observación, la inclusión de campos específicos como ship_mode, segment, region y shipping_days en la tabla de ventas; sub_categoria en la tabla de productos; y descuento en el detalle de venta facilitó que las operaciones registradas por los usuarios del sistema se transformaran automáticamente en datos para el modelo predictivo. Esta integración hace innecesaria la preparación manual de los datos y asegura que las predicciones representen el accionar real del negocio.

Otro punto a destacar, se comprobó que el diseño de la base de datos con triggers automatizados es un método eficaz para preservar la integridad del inventario sin tener que recurrir a lógica en la capa de aplicación. El sistema de auditoría con base en JSONB permite seguir en tiempo real todas las operaciones realizadas sobre las entidades críticas del sistema, mientras que los disparadores trg_stock y trg_total aseguran la actualización atómica del stock y el cálculo total de cada venta.

Como siguiente punto, el tablero interactivo logró unificar en una única interfaz los datos históricos de las ventas, la situación presente del inventario y las previsiones producidas por la inteligencia artificial. Esto le ofreció al usuario un panorama completo del negocio sin tener que recurrir a diversas fuentes de información. El uso de colores, estilos y secciones diferenciadas en la visualización de datos reales y predicciones ayuda al usuario a entender adecuadamente la naturaleza estimada de los valores predictivos.

Para concluir, las pruebas de integración y funcionales realizadas corroboraron que el sistema opera adecuadamente en todos los flujos analizados. Estos incluyen la validación del stock antes de registrar ventas, la actualización automática del inventario a través de triggers, el registro de auditoría y la creación de predicciones basadas en datos reales. El sistema reaccionó apropiadamente ante entradas válidas y ante situaciones de error, devolviendo mensajes descriptivos sin afectar la estabilidad del servidor.

Por último, este proyecto muestra que no solamente las grandes compañías con recursos tecnológicos avanzados pueden desarrollar sistemas de gestión inteligente de inventarios mediante análisis predictivo. Se pueden crear soluciones que tengan capacidad de inteligencia artificial, sean escalables y funcionales con el uso de herramientas ampliamente accesibles mediante código abierto como React.js, Node.js,

PostgreSQL y scikit-learn; estas soluciones se adaptan a las demandas auténticas del sector de las pequeñas y medianas empresas.

5.2. Trabajos Futuros

Con base en las limitaciones detectadas y los resultados logrados a lo largo del desarrollo del sistema, se proponen las siguientes sugerencias, enfocadas en optimizar, ampliar y sostener la solución que se ha puesto en marcha:

Expandir el conjunto de datos de entrenamiento con información que abarque diversas categorías de productos. El modelo actual fue entrenado solamente con información de la categoría Furniture, por lo que su aplicabilidad se limita a productos de mobiliario. Se aconseja reunir datos históricos de ventas de la empresa real en la que se aplique el sistema y volver a entrenar los modelos con esos datos. Esto aumentaría considerablemente la exactitud de las proyecciones, ya que representaría los patrones específicos de demanda del negocio.

Establecer un procedimiento regular de reentrenamiento para los modelos. Se aconseja implementar un procedimiento automatizado mensual o trimestral que reentrene los modelos con datos nuevos a medida que el sistema acumule datos sobre ventas reales. Esto permitirá la adaptación de las predicciones a variaciones en los patrones de consumo, estacionalidad y tendencias del mercado.

Investigar algoritmos de series temporales para hacer predicciones a largo plazo. El modelo Random Forest que se utiliza hace predicciones por cada transacción individualmente. Para hacer estimaciones de ventas mensuales o trimestrales, se aconseja analizar algoritmos enfocados en series temporales, como Prophet de Meta o LSTM (Long Short-Term Memory), que son mejores para identificar la estacionalidad y los patrones a nivel temporal en las cifras históricas de venta.

Con medidas adicionales, se puede robustecer la seguridad del sistema. Se aconseja establecer auditorías de seguridad periódicas, cifrado de datos sensibles en la base de datos y límites en los intentos para iniciar sesión (rate limiting).

En resumen, el sistema desarrollado representa una base sólida y efectiva que puede ser utilizada para construir mejoras continuas. La mezcla de una arquitectura web actualizada con métodos de aprendizaje automático ha probado ser un enfoque factible para proveer a las pymes (pequeñas y medianas empresas) de instrumentos inteligentes

de gestión que, por lo general, solo han estado disponibles para entidades con más recursos financieros y tecnológicos.

BIBLIOGRAFÍA

Atnafu, D., & Balda, A. (2018). The impact of inventory management practice on firms' competitiveness and organizational performance: Empirical evidence from micro and small enterprises in Ethiopia. Cogent Business & Management, 5(1), 1503219. https://doi.org/10.1080/23311975.2018.1503219

AWS. (2026). What is ETL? Extract transform load explained. https://aws.amazon.com/what-is/etl/

AWS. (2026). What is OLAP? Online analytical processing explained. https://aws.amazon.com/what-is/olap/

Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley Professional. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://ptgmedia.pearsoncmg.com/images/9780321815736/samplepages/0321815734.pdf

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Few, S. (2006). Information Dashboard Design. O´Reilly. https://public.magendanz.com/Temp/Information%20Dashboard%20Design.pdf

Few, S. (2012). Show Me the Numbers: Designing Tables and Graphs to Enlighten. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://analyticsconsultores.com.mx/wp-content/uploads/2019/03/Show-Me-the-Numbers-Stephen-Few-Perceptual-Edge-2004.pdf

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures [Tesis doctoral, University of California, Irvine]. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Han, J., Kamber, M., & Pei, J. (2012). Data mining: concepts and techniques. Choice Reviews Online, 49(06), 49-3305. https://doi.org/10.5860/choice.49-3305

IBM. (2026). What is business intelligence (BI)? IBM Think. https://www.ibm.com/think/topics/business-intelligence

Laudon, K. C., & Laudon, J. P. (2020). Management information systems: Managing the digital firm (16th ed.). Pearson. https://es.slideshare.net/slideshow/etextbook-pdf-for-management-information-systems-managing-the-digital-firm-16th-edition/281541766

Tableau. (s.f.). What is data visualization? Definition, examples, and learning resources. https://www.tableau.com/visualization/what-is-data-visualization

Tufte, E. R. (2001). The Visual Display of Quantitative Information. Graphics Press. https://kyl.neocities.org/books/%5BTEC%20TUF%5D%20the%20visual%20display%20of%20quantitative%20information.pdf

ANEXOS

Anexo A: Script de Base de datos

```
-- ============================

-- TABLA: usuarios

-- ============================

CREATE TABLE usuarios (

   id SERIAL PRIMARY KEY,

   email VARCHAR(100) UNIQUE NOT NULL,

   password VARCHAR(255) NOT NULL,

   rol VARCHAR(20) CHECK (rol IN ('admin', 'vendedor')) NOT NULL,

   activo BOOLEAN DEFAULT TRUE,

   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);


-- ============================

-- TABLA: categorias

-- ============================

CREATE TABLE categorias (

   id SERIAL PRIMARY KEY,

   nombre VARCHAR(100) UNIQUE NOT NULL,

   descripcion TEXT

);


-- ============================

-- TABLA: productos

-- ============================

CREATE TABLE productos (

   id SERIAL PRIMARY KEY,
```

```
    nombre VARCHAR(150) NOT NULL,
    descripcion TEXT,
    precio NUMERIC(10,2) NOT NULL,
    stock INT DEFAULT 0,
    stock_minimo INT DEFAULT 0,
    stock_maximo INT DEFAULT 0,
    categoria_id INT,
    sku VARCHAR(50) UNIQUE,
    estado VARCHAR(20) CHECK (estado IN ('activo', 'inactivo')) DEFAULT 'activo',
    ubicacion VARCHAR(50),
    imagen TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (categoria_id) REFERENCES categorias(id) ON DELETE SET NULL
);

-- ==========================
-- TABLA: ventas
-- ==========================
CREATE TABLE ventas (
    id SERIAL PRIMARY KEY,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    cliente VARCHAR(150),
    metodo_pago VARCHAR(50),
    estado VARCHAR(20) CHECK (estado IN ('completado', 'pendiente', 'cancelado')),
    vendedor_id INT,
    total NUMERIC(10,2) DEFAULT 0,
```

```
    FOREIGN KEY (vendedor_id) REFERENCES usuarios(id) ON
DELETE SET NULL
);


-- ============================
-- TABLA: detalle_venta
-- ============================
CREATE TABLE detalle_venta (
    id SERIAL PRIMARY KEY,
    venta_id INT NOT NULL,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    precio_unitario NUMERIC(10,2) NOT NULL,
    subtotal NUMERIC(10,2) NOT NULL,


    FOREIGN KEY (venta_id) REFERENCES ventas(id) ON DELETE
CASCADE,
    FOREIGN KEY (producto_id) REFERENCES productos(id) ON
DELETE CASCADE
);
-- ============================
-- TABLA: logs (auditoría PRO)
-- ============================
CREATE TABLE logs (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    usuario_id INT,
    accion VARCHAR(10),
    tabla_afectada VARCHAR(50),
    registro_id INT,
```

```
    datos_antes JSONB,
    datos_despues JSONB,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ==========================
-- FUNCIÓN: obtener usuario sesión
-- ==========================
CREATE OR REPLACE FUNCTION get_current_user_id()
RETURNS INT AS $$
BEGIN
    RETURN current_setting('app.user_id', true)::INT;
EXCEPTION
    WHEN OTHERS THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- ==========================
-- FUNCIÓN: logs avanzados
-- ==========================
CREATE OR REPLACE FUNCTION fn_logs()
RETURNS TRIGGER AS $$
DECLARE
    usuario INT;
BEGIN
    usuario := get_current_user_id();

    IF TG_OP = 'INSERT' THEN
```

```
        INSERT INTO logs (usuario_id, accion, tabla_afectada, registro_id, datos_despues)
        VALUES (usuario, 'INSERT', TG_TABLE_NAME, NEW.id, to_jsonb(NEW));
        RETURN NEW;

    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO logs (usuario_id, accion, tabla_afectada, registro_id, datos_antes, datos_despues)
        VALUES (usuario, 'UPDATE', TG_TABLE_NAME, NEW.id, to_jsonb(OLD), to_jsonb(NEW));
        RETURN NEW;

    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO logs (usuario_id, accion, tabla_afectada, registro_id, datos_antes)
        VALUES (usuario, 'DELETE', TG_TABLE_NAME, OLD.id, to_jsonb(OLD));
        RETURN OLD;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

-- ============================
-- TRIGGERS DE LOGS
-- ============================
CREATE TRIGGER trg_logs_productos
AFTER INSERT OR UPDATE OR DELETE ON productos
FOR EACH ROW EXECUTE FUNCTION fn_logs();
```

```
CREATE TRIGGER trg_logs_ventas
AFTER INSERT OR UPDATE OR DELETE ON ventas
FOR EACH ROW EXECUTE FUNCTION fn_logs();

CREATE TRIGGER trg_logs_detalle
AFTER INSERT OR UPDATE OR DELETE ON detalle_venta
FOR EACH ROW EXECUTE FUNCTION fn_logs();

-- ============================
-- TRIGGER: actualizar stock
-- ============================
CREATE OR REPLACE FUNCTION actualizar_stock()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE productos
    SET stock = stock - NEW.cantidad
    WHERE id = NEW.producto_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_stock
AFTER INSERT ON detalle_venta
FOR EACH ROW
EXECUTE FUNCTION actualizar_stock();

-- ============================
```

```
-- TRIGGER: total venta
-- ============================
CREATE OR REPLACE FUNCTION actualizar_total()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE ventas
   SET total = (
      SELECT COALESCE(SUM(subtotal), 0)
      FROM detalle_venta
      WHERE venta_id = NEW.venta_id
   )
   WHERE id = NEW.venta_id;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_total
AFTER INSERT ON detalle_venta
FOR EACH ROW
EXECUTE FUNCTION actualizar_total();

ALTER TABLE usuarios
ADD COLUMN verificado BOOLEAN DEFAULT FALSE,
ADD COLUMN verify_token TEXT;



ALTER TABLE usuarios
```

```
ADD COLUMN reset_token TEXT,
ADD COLUMN reset_expires TIMESTAMP;

ALTER TABLE usuarios
ADD COLUMN cedula VARCHAR(10),
ADD COLUMN telefono VARCHAR(10),
ADD COLUMN direccion TEXT;

ALTER TABLE usuarios
ALTER COLUMN cedula SET NOT NULL,
ALTER COLUMN telefono SET NOT NULL;

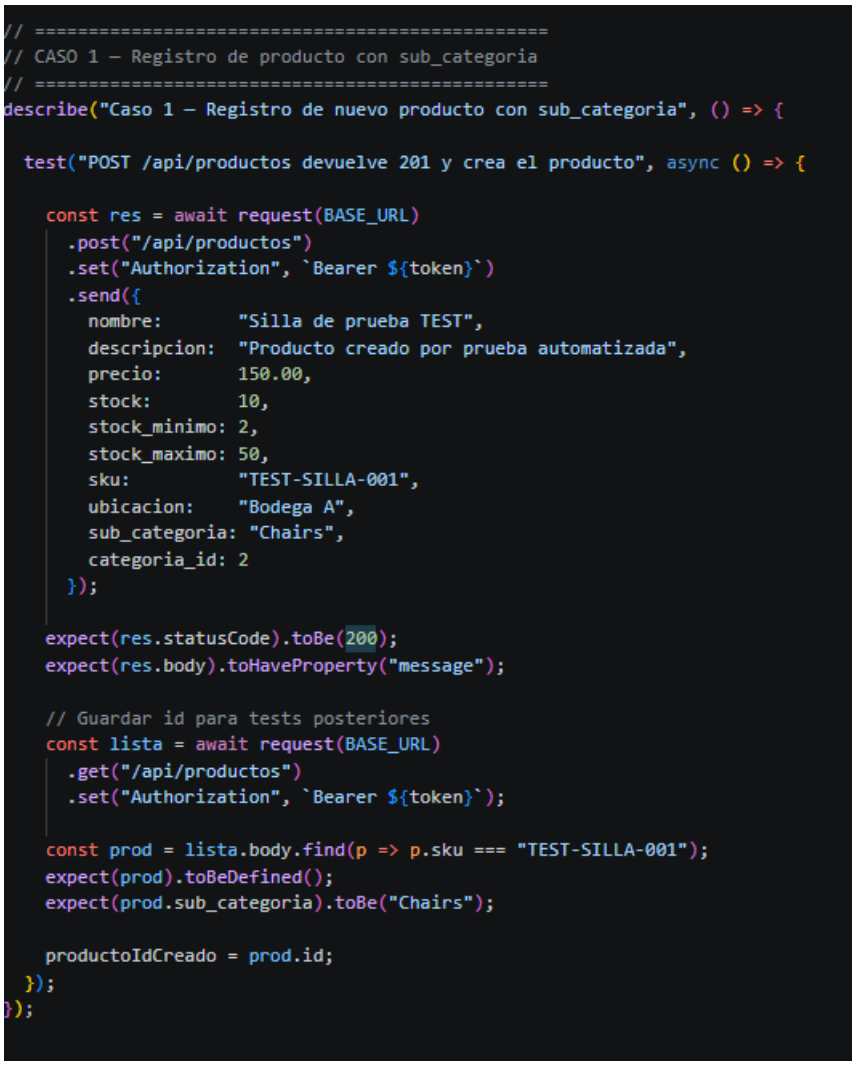
-- Nuevas columnas en ventas
ALTER TABLE ventas
ADD COLUMN IF NOT EXISTS ship_mode VARCHAR(50) DEFAULT 'Standard Class',
ADD COLUMN IF NOT EXISTS segment VARCHAR(50) DEFAULT 'Consumer',
ADD COLUMN IF NOT EXISTS region VARCHAR(50) DEFAULT 'West',
ADD COLUMN IF NOT EXISTS shipping_days INT DEFAULT 3;

-- Nueva columna en detalle_venta
ALTER TABLE detalle_venta
ADD COLUMN IF NOT EXISTS descuento NUMERIC(4,2) DEFAULT 0;

-- Nueva columna en productos
ALTER TABLE productos
ADD COLUMN IF NOT EXISTS sub_categoria VARCHAR(100) DEFAULT 'Chairs';
```

Anexo B: Resultados detallados de pruebas

Figura 11. *Prueba Registro de Producto*



```
// ================================================
// CASO 1 — Registro de producto con sub_categoria
// ================================================
describe("Caso 1 — Registro de nuevo producto con sub_categoria", () => {

  test("POST /api/productos devuelve 201 y crea el producto", async () => {

    const res = await request(BASE_URL)
      .post("/api/productos")
      .set("Authorization", `Bearer ${token}`)
      .send({
        nombre:       "Silla de prueba TEST",
        descripcion:  "Producto creado por prueba automatizada",
        precio:       150.00,
        stock:        10,
        stock_minimo: 2,
        stock_maximo: 50,
        sku:          "TEST-SILLA-001",
        ubicacion:    "Bodega A",
        sub_categoria: "Chairs",
        categoria_id: 2
      });

    expect(res.statusCode).toBe(200);
    expect(res.body).toHaveProperty("message");

    // Guardar id para tests posteriores
    const lista = await request(BASE_URL)
      .get("/api/productos")
      .set("Authorization", `Bearer ${token}`);

    const prod = lista.body.find(p => p.sku === "TEST-SILLA-001");
    expect(prod).toBeDefined();
    expect(prod.sub_categoria).toBe("Chairs");

    productoIdCreado = prod.id;
  });
});
```

Autor: Daniel Campoverde. Código para evaluar el registro de productos

Figura 12. *Prueba de ventas*



```
// ================================================
// CASO 2 — Venta con múltiples productos
//          verifica que el trigger descuenta stock
// ================================================
describe("Caso 2 — Creación de venta y decremento automático de stock", () => {

  test("POST /api/ventas descuenta stock vía trigger", async () => {

    // Stock antes
    const antes = await request(BASE_URL)
      .get("/api/productos")
      .set("Authorization", `Bearer ${token}`);

    const prodAntes = antes.body.find(p => p.id === productoIdCreado);
    const stockAntes = Number(prodAntes.stock);

    // Crear venta
    const res = await request(BASE_URL)
      .post("/api/ventas")
      .set("Authorization", `Bearer ${token}`)
      .send({
        cliente:       "Cliente Prueba",
        metodo_pago:   "Efectivo",
        estado:        "completado",
        ship_mode:     "Standard Class",
        segment:       "Consumer",
        region:        "West",
        shipping_days: 3,
        detalles: [
          {
            producto_id: productoIdCreado,
            cantidad:    2,
            descuento:   0
          }
        ]
      });

    expect(res.statusCode).toBe(201);
    expect(res.body).toHaveProperty("venta_id");
    ventaIdCreada = res.body.venta_id;

    // Stock después
    const despues = await request(BASE_URL)
      .get("/api/productos")
      .set("Authorization", `Bearer ${token}`);

    const prodDespues = despues.body.find(p => p.id === productoIdCreado);
    const stockDespues = Number(prodDespues.stock);

    expect(stockDespues).toBe(stockAntes - 2);
  });
});
```

Autor: Daniel Campoverde. Código para evaluar el registro de las ventas

Figura 13. *Prueba ventas con stock insuficiente*



```
// ================================================
// CASO 3 — Venta con stock insuficiente
// ================================================
describe("Caso 3 — Intento de venta con stock insuficiente", () => {

  test("POST /api/ventas rechaza con error si stock < cantidad", async () => {

    const res = await request(BASE_URL)
      .post("/api/ventas")
      .set("Authorization", `Bearer ${token}`)
      .send({
        cliente:       "Cliente Prueba",
        metodo_pago:   "Efectivo",
        estado:        "completado",
        ship_mode:     "Standard Class",
        segment:       "Consumer",
        region:        "West",
        shipping_days: 3,
        detalles: [
          {
            producto_id: productoIdCreado,
            cantidad:    9999,      // imposible
            descuento:   0
          }
        ]
      });

    expect(res.statusCode).toBe(500);
    expect(res.body).toHaveProperty("message");
    expect(res.body.message).toMatch(/stock/i);
  });
});
```

Autor: Daniel Campoverde. Código para evaluar la restricción de productos insuficientes

Figura 14. *Prueba de alerta de stock bajo*



```
// ================================================
// CASO 4 — Alerta de stock bajo
//          verifica que el campo stock < stock_minimo
// ================================================
describe("Caso 4 — Alerta de stock bajo detectada en la respuesta", () => {

  test("GET /api/productos retorna producto con stock < stock_minimo", async () => {

    // Bajar stock manualmente al mínimo
    await request(BASE_URL)
      .put(`/api/productos/${productoIdCreado}`)
      .set("Authorization", `Bearer ${token}`)
      .send({
        nombre:        "Silla de prueba TEST",
        descripcion:   "Producto creado por prueba automatizada",
        precio:        150.00,
        stock:         1,          // menor al stock_minimo de 2
        stock_minimo:  2,
        stock_maximo:  50,
        sku:           "TEST-SILLA-001",
        ubicacion:     "Bodega A",
        sub_categoria: "Chairs",
        categoria_id:  2,
        estado:        "activo"
      });

    const res = await request(BASE_URL)
      .get("/api/productos")
      .set("Authorization", `Bearer ${token}`);

    const prod = res.body.find(p => p.id === productoIdCreado);

    expect(prod).toBeDefined();
    expect(Number(prod.stock)).toBeLessThan(Number(prod.stock_minimo));
  });
});
```

Autor: Daniel Campoverde. Código para evaluar el funcionamiento de alertas en stock bajo

Figura 15. *Prueba de predicción de IA*

```
// ================================================
// CASO 5 — Predicción IA desde el dashboard
// ================================================
describe("Caso 5 — Predicción IA retorna ventas_predichas y demanda_predicha", () => {

  test("POST /api/ia/prediccion retorna valores numéricos", async () => {

    const res = await request(BASE_URL)
      .post("/api/ia/prediccion")
      .set("Authorization", `Bearer ${token}`)
      .send({
        "Ship Mode":     "Standard Class",
        "Segment":       "Consumer",
        "City":          "Los Angeles",
        "State":         "California",
        "Region":        "West",
        "Category":      "Furniture",
        "Sub-Category":  "Chairs",
        "Discount":      0.1,
        "Shipping Days": 3,
        "Month":         5,
        "Day":           14,
        "Year":          2025,
        "WeekDay":       2,
        "Weekend":       0
      });

    expect(res.statusCode).toBe(200);
    expect(res.body).toHaveProperty("ventas_predichas");
    expect(res.body).toHaveProperty("demanda_predicha");
    expect(typeof res.body.ventas_predichas).toBe("number");
    expect(typeof res.body.demanda_predicha).toBe("number");
    expect(res.body.ventas_predichas).toBeGreaterThan(0);
  });
});
```

Autor: Daniel Campoverde. Código para evaluar las predicciones basadas en IA

Figura 16. *Prueba de Registro de logs*

```
// ================================================
// CASO 6 — Registro de logs de auditoría
// ================================================
describe("Caso 6 — Logs de auditoría generados por operaciones", () => {

  test("GET /api/logs retorna registros con datos antes/después", async () => {

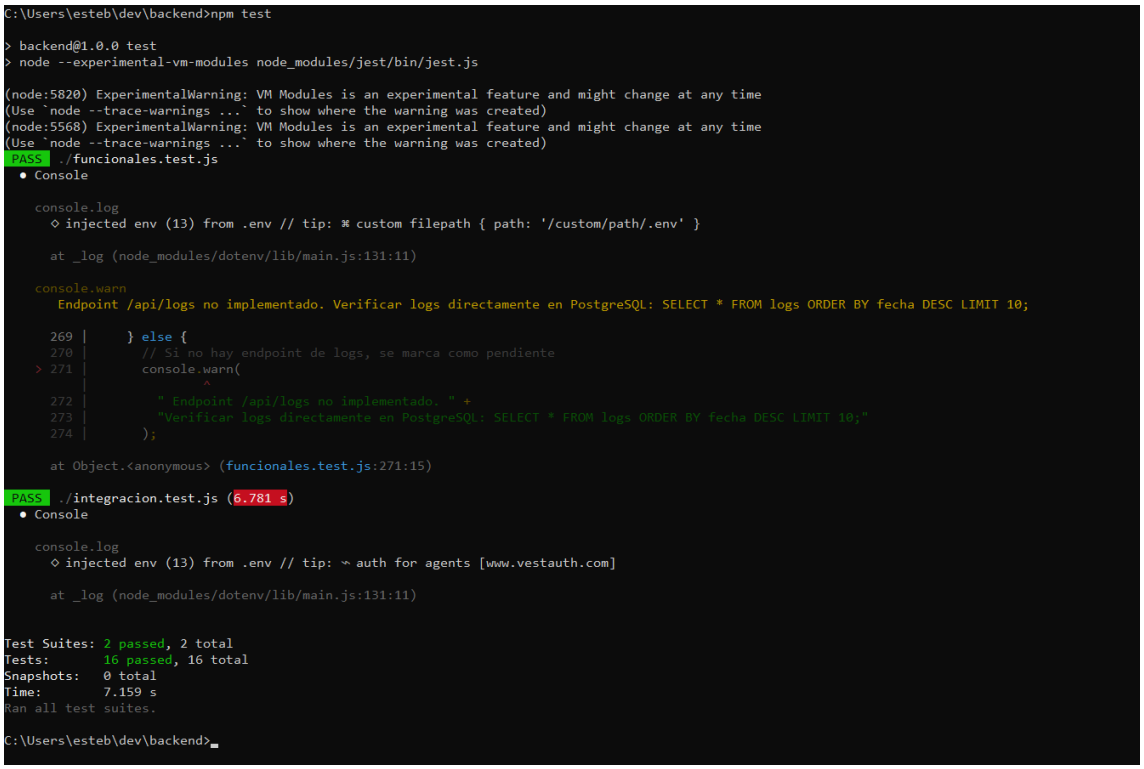
    const res = await request(BASE_URL)
      .get("/api/logs")
      .set("Authorization", `Bearer ${token}`);

    // El endpoint puede no existir aún — se valida la BD directamente
    // Si existe devuelve 200, si no existe devuelve 404
    // En ambos casos el test verifica que los triggers funcionaron
    if (res.statusCode === 200) {
      const logs = res.body;
      expect(Array.isArray(logs)).toBe(true);
      expect(logs.length).toBeGreaterThan(0);

      const logProducto = logs.find(
        l => l.tabla_afectada === "productos"
      );
      expect(logProducto).toBeDefined();
      expect(logProducto).toHaveProperty("datos_despues");
```

Autor: Daniel Campoverde. Código para evaluar el registro de logs después de cada acción

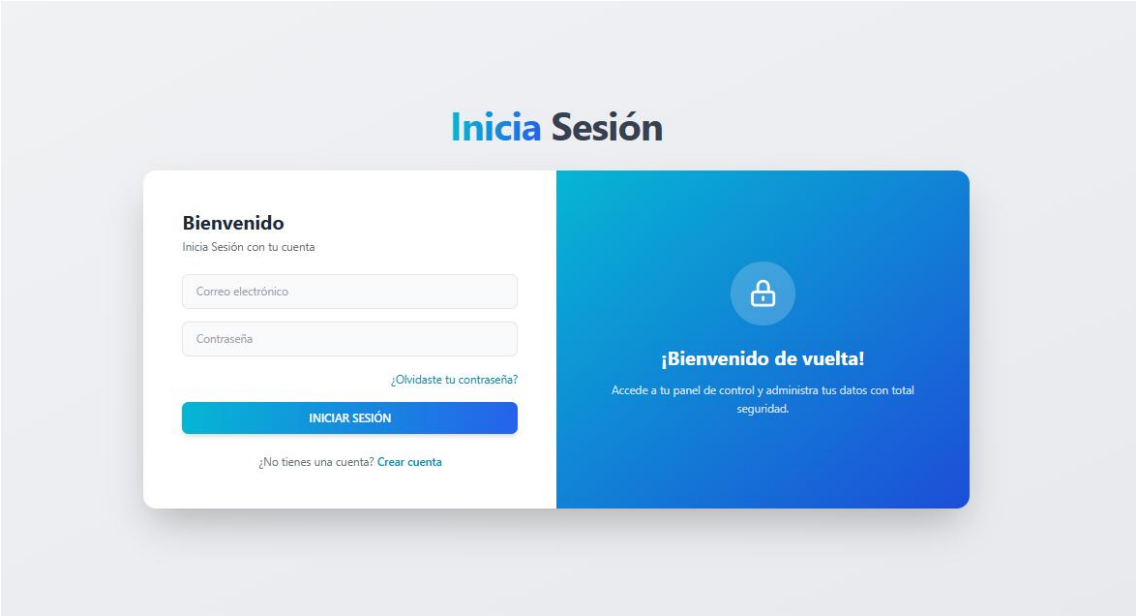
Figura 17. *Resultados pruebas de Integración y Funcionales*



Autor: Daniel Campoverde. Resultados de la ejecución de las pruebas

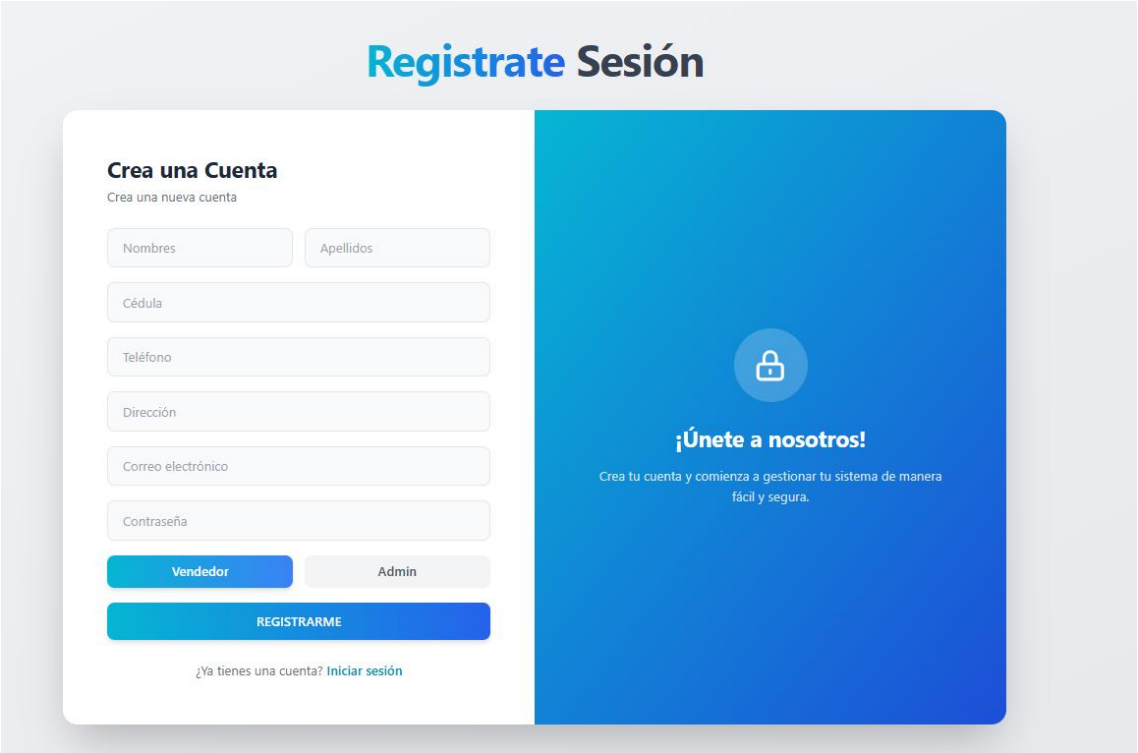
Anexo C: Capturas de pantalla del sistema

Figura 18. Interfaz de Login



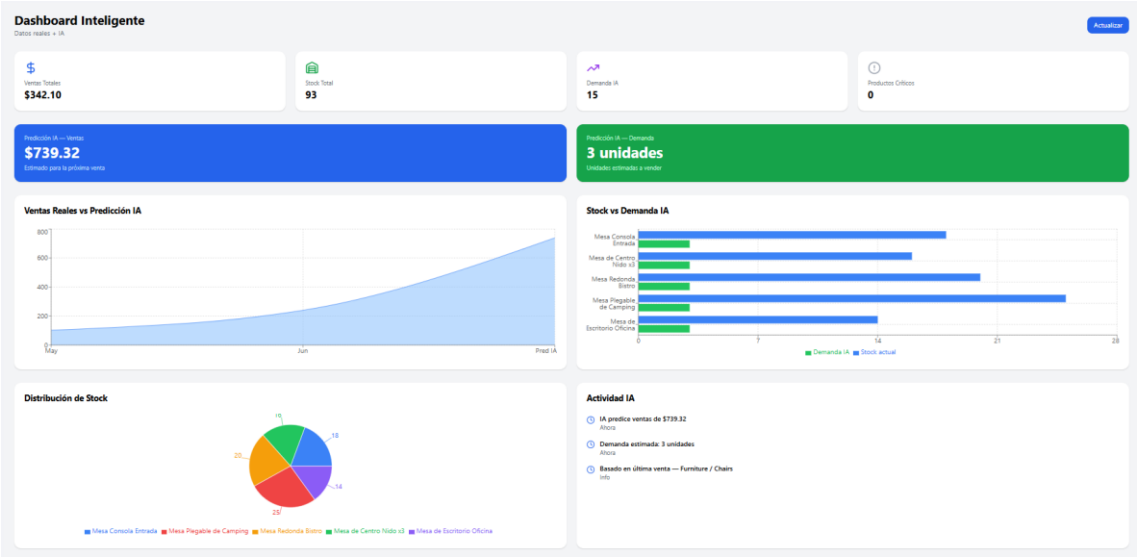
Autor: Daniel Campoverde. Pantalla de inicio de sesión

Figura 19. *Interfaz de Registro*



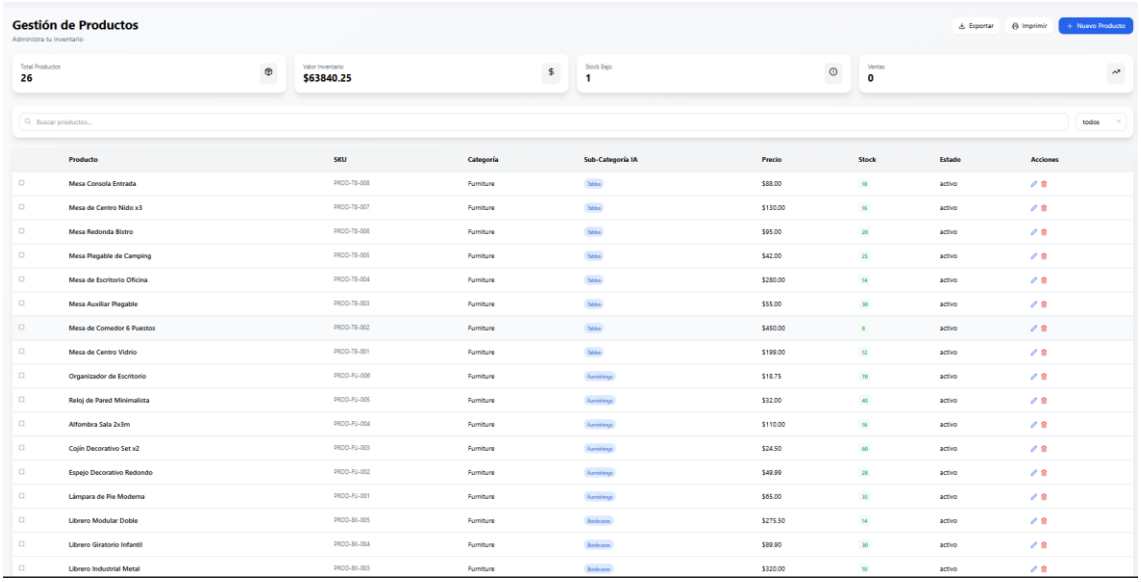
Autor: Daniel Campoverde. Pantalla de registro al sistema

Figura 20. *Dashboard Inteligente*



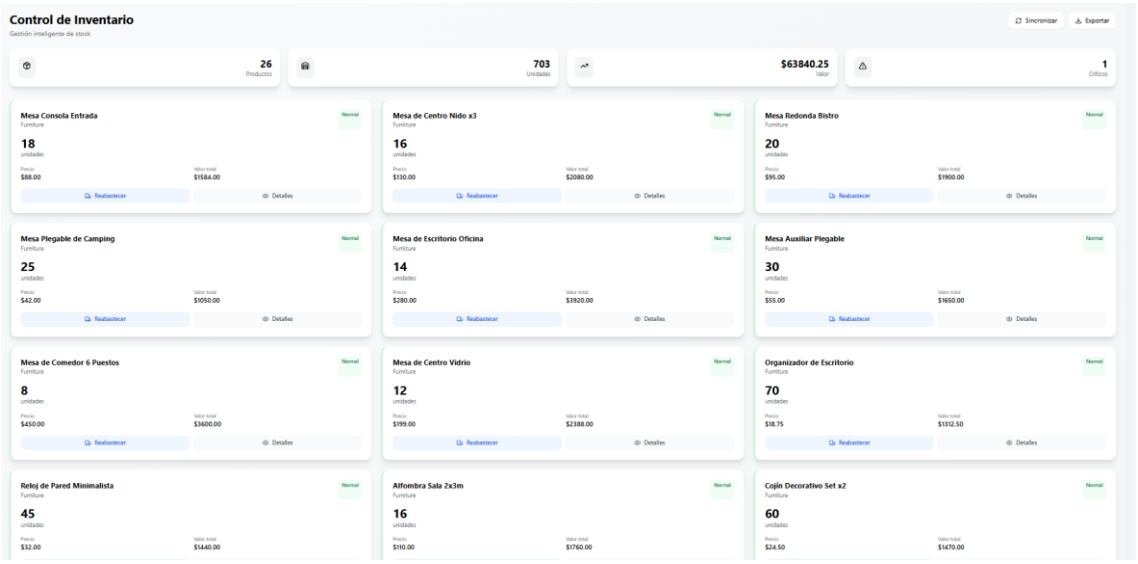
Autor: Daniel Campoverde. Visualización de datos mediante gráficas

Figura 21. *Gestión de Productos*



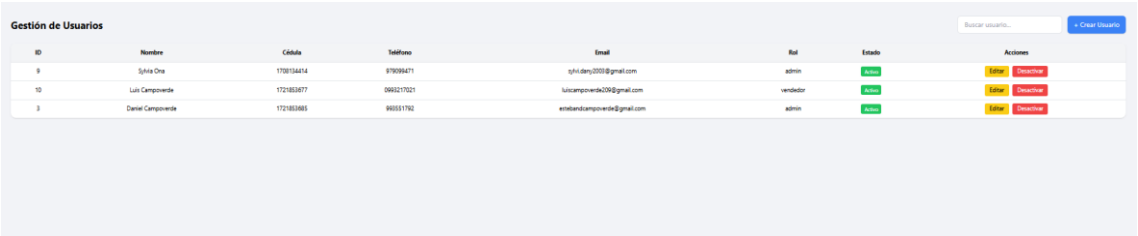
Autor: Daniel Campoverde. Pantalla del módulo de productos registrados

Figura 22. *Módulo de Inventario*



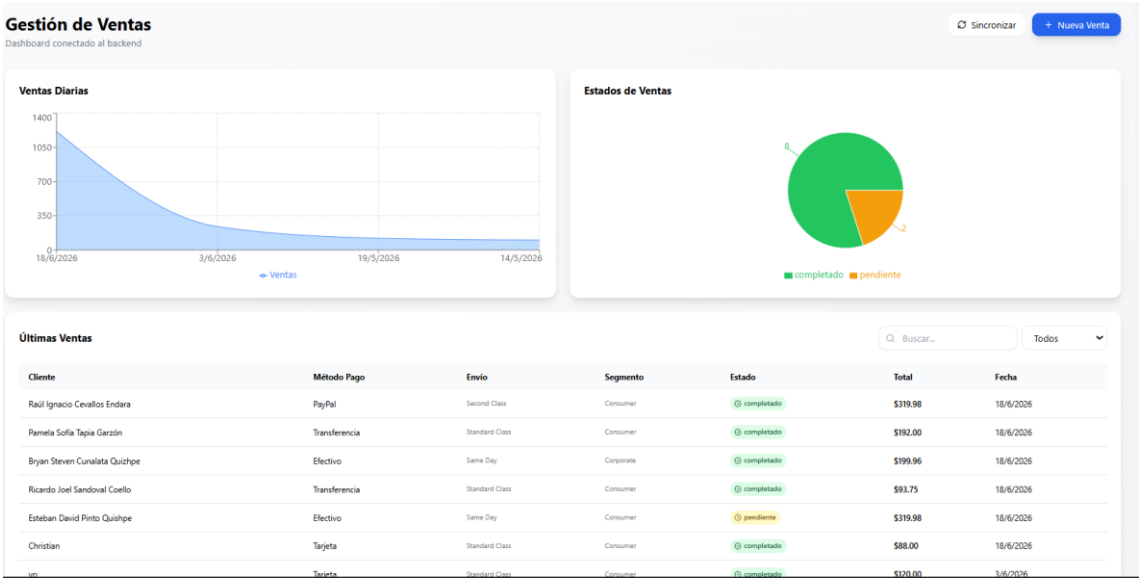
Autor: Daniel Campoverde. Pantalla del módulo de gestión de inventario

Figura 23. *Módulo de Usuarios*



Autor: Daniel Campoverde. Pantalla del módulo de gestión de usuarios

Figura 24. *Módulo de ventas*



Autor: Daniel Campoverde. Pantalla del módulo de gestión de ventas

ANEXO D: Código de entrenamiento del modelo de IA

https://colab.research.google.com/drive/11IFQhM8GkoiB7fHaHIEMCfUexr_-jSmt?usp=sharing

ANEXO E: Configuración del entorno de trabajo

Figura 25. *Inicialización del Backend*

```
C:\Users\esteb\dev\backend>npm init -y
Wrote to C:\Users\esteb\dev\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Autor: Daniel Campoverde. Comando para la creación del servidor en Node.js

Figura 26. *Instalar dependencias en backend*

```
C:\Users\esteb\dev\backend>npm install express pg cors dotenv jsonwebtoken bcrypt

added 99 packages, and audited 100 packages in 15s

25 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\esteb\dev\backend>npm install --save-dev nodemon

added 25 packages, and audited 125 packages in 5s

30 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\esteb\dev\backend>
```

Autor: Daniel Campoverde. Comando para la instalación de dependencias necesarias

ANEXO F: stores_sales_forecasting.csv

https://drive.google.com/file/d/1IFTL1hi4mCck-JKoMs6dZS0AbyKrOwhj/view?usp=sharing

ANEXO G: Entrenamiento de modelos para comparación

https://colab.research.google.com/drive/1l37tnF28CQgM8ke6iAXl9cFu_bUWGf68?usp=sharing