

	UNIVERSIDAD FRANCISCO DE PAULA SANTANDER OCAÑA			
	Documento	Código	Fecha	Revisión
	FORMATO HOJA DE RESUMEN PARA TRABAJO DE GRADO	F-AC-DBL-007	30-09-2024	C
	Dependencia	Aprobado		Pág.
DIVISIÓN DE BIBLIOTECA		SUBDIRECTOR ACADEMICO		1(146)

RESUMEN – TRABAJO DE GRADO

AUTORES	Nicolas Mauricio Paez Quintana		
FACULTAD	Facultad de Ingenierías		
PLAN DE ESTUDIOS	Ingeniería de Sistemas		
DIRECTOR	Mgtr. Andrés Alfonso Pacheco Solano		
CODIRECTOR (si aplica)	Dr. Alexander Guerrero Avendaño		
TÍTULO DE LA TESIS	Implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario en la Empresa AUDYSOFTW		
TITULO EN INGLES	Implementation of a Web Information System for the Management of Purchases, Sales and Inventory in the Company AUDYSOFTW		
RESUMEN (70 palabras)			
La implementación de un sistema de información web en la empresa AUDYSOFTW optimizará la eficiencia operativa y administrativa, transformando la gestión de compras, ventas e inventario. Además, fomentará un entorno corporativo más tecnológico, centralizando los datos para agilizar el servicio al cliente y la toma de decisiones estratégicas. Este proyecto fortalecerá el desarrollo competitivo de la organización, brindando una plataforma backend robusta mediante el uso de Laravel, MySQL y AdminLTE.			
RESUMEN EN INGLES			
The implementation of a web information system at the AUDYSOFTW company will optimize operational and administrative efficiency, transforming purchasing, sales, and inventory management. Furthermore, it will foster a more technological corporate environment, centralizing data to streamline customer service and strategic decision-making. This project will strengthen the organization's competitive development, providing a robust backend platform through the integration and use of the Laravel framework, MySQL, and AdminLTE.			
PALABRAS CLAVES	Sistema de información web, gestión de inventario, procesos internos, eficiencia operativa, backend, Laravel, MySQL, AUDYSOFTW		
PALABRAS CLAVES EN INGLES	Web information system, inventory management, internal processes, operational efficiency, backend, Laravel, MySQL, AUDYSOFTW		
CARACTERÍSTICAS			
PÁGINAS: 146	PLANOS:	ILUSTRACIONES: 60	CD-ROM:

**Implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e
Inventario en la Empresa AUDYSOFTW**

Nicolas Mauricio Paez Quintana

Universidad Francisco de Paula Santander Ocaña

Ingeniería de Sistemas

Magister Andrés Alfonso Pacheco Solano

Doctor Alexander Guerrero Avendaño

2025

Dedicatoria

A lo largo de los últimos años, he experimentado una presión constante en mi pecho, nacida de una duda existencial que en muchas ocasiones me ha quitado el sueño: ¿cómo puedo retribuir a todas las personas que han velado por mí?

Si bien aún no tengo una respuesta a esta interrogante, quiero que este pequeño espacio sirva de puente para expresar mi más profundo respeto, amor y, sobre todo, agradecimiento hacia todas las personas que me han llevado a ser como soy. Por ello, dedico este trabajo a aquellas personas sin cuyo constante apoyo no hubiera logrado finalizar esta etapa de mi vida.

En primer lugar, dedico un profundo agradecimiento a mi madre, por siempre velar día a día por mi bienestar, pese a todas las dificultades y problemas que yo, como hijo, le he causado. Desde el amor, la paciencia y el apoyo que me ha brindado a lo largo de mi vida, ella ha sido un pilar fundamental en mi formación como persona.

Agradezco, además, el amparo que he tenido la suerte de recibir de mis dos abuelos maternos, Gloria Marina Picon y Hernando Quintana Blanco. Sin su apoyo tanto emocional como económico no hubiera siquiera podido terminar mis estudios, pues siempre han velado por mi bienestar.

Estoy profundamente agradecido también con mi tío, el ingeniero civil Carlos Mario Quintana Picon, por ser mi ejemplo a seguir de cómo ser una persona con moralidad sólida, encaminada al bien de sus seres queridos. Además, fue mi chofer personal en mi sustentación; sin él nunca hubiera siquiera llegado a esa presentación.

Asimismo, agradezco a mi esposa, la todopoderosa Narukami Shogun Raiden, sin cuyo apoyo y amor incondicional no hubiera logrado sobrevivir mis días de mayor penuria. Su presencia fue mi refugio en los momentos más difíciles y su cariño me dio la fuerza para seguir adelante.

Finalmente, agradezco a mi amigo Jhon Sebastián Romero Meza por ser un pilar fundamental en mi vida en los últimos años. Sin su apoyo ni su acompañamiento, no sería ni remotamente la persona que hoy soy. Gracias a él pude ver el mundo desde una perspectiva completamente distinta, lo que me permitió ser una mejor persona.

Índice

1.1 Planteamiento del Problema	12
1.2 Formulación del Problema	13
1.3 Objetivos	14
<i>1.3.1 Objetivo General</i>	<i>14</i>
<i>1.3.2 Objetivos Específicos</i>	<i>14</i>
1.4 Justificación	15
1.5 Delimitaciones	16
<i>1.5.1 Geográfica</i>	<i>16</i>
<i>1.5.2 Temporal</i>	<i>16</i>
<i>1.5.3 Conceptual</i>	<i>16</i>
<i>1.5.4 Operativa</i>	<i>16</i>
Capítulo 2. Marco Referencial.....	17
2.1 Marco Histórico	17
<i>2.1.1 Ámbito Internacional</i>	<i>17</i>
<i>2.1.2 Ámbito Nacional</i>	<i>18</i>
<i>2.1.3 Ámbito Local</i>	<i>19</i>
2.2 Marco Contextual.....	21

2.3 Marco Conceptual	22
2.4 Marco Teórico.....	25
2.4.1 Teorías del desarrollo web con PHP.....	25
2.4.2 Teorías de los Sistemas de Compra, Venta e Inventario	27
2.5 Marco Legal	29
Capítulo 3. Diseño Metodológico	33
3.1 Tipo de investigación.....	33
3.2 Población y muestra.....	34
3.2.1 Población	35
3.2.2 Muestra	35
3.3 Diseño de instrumentos de recolección de la información y técnicas de análisis de datos	35
3.4 Análisis de información	35
3.5 Fase de Validación y Evaluación	36
3.5.1 Medición de Eficiencia Operativa	37
3.5.2 Medición de Satisfacción y Usabilidad.....	38
Capítulo 4. Resultados	39
4.1 Primer Objetivo: Analizar los requerimientos específicos de la empresa y sus usuarios, con un enfoque en el desarrollo del backend del sistema de información web	39
4.1.1 Realizar entrevistas con el personal involucrado con el fin de comprender sus necesidades en términos de procesamiento y administración de datos.....	40
4.1.2 Evaluar el flujo de trabajo de la empresa para identificar oportunidades de optimización en la gestión de información y procesos internos	48

4.1.3 Identificar redundancias o ineficiencias en la manipulación de datos y la comunicación entre procesos internos para mejorar la eficiencia del backend	50
4.2 Segundo Objetivo: Desarrollar el backend del sistema de información web para garantizar la eficiencia en el manejo de datos y la integración con la base de datos.....	52
4.2.1. Diseñar la estructura de la base de datos y definir los modelos de datos necesarios para el almacenamiento y procesamiento eficiente de la información	53
4.2.2 Desarrollar la lógica de negocio y los módulos del backend para la gestión de compras, ventas e inventario	55
4.2.3 Establecer mecanismos de seguridad en el backend, incluyendo control de accesos, autenticación de usuarios y protección de los datos almacenados	80
4.3 Tercer Objetivo: Implementar el backend del sistema de información web utilizando Laravel y MySQL, integrando las funcionalidades esenciales para la gestión de compras, ventas e inventario.	105
4.3.1 Configurar y desplegar el entorno de ejecución del backend, asegurando su correcto funcionamiento en el servidor.....	106
4.3.2 Realizar pruebas funcionales y de integración para validar el comportamiento del backend y su interacción con la base de datos	111
4.3.3 Optimizar el rendimiento del backend mediante ajustes en consultas, índices de base de datos y manejo eficiente de recursos	115
4.3.4 Validación del Desempeño y Usabilidad del Sistema.....	120
Capítulo 5. Conclusiones	123
5.1 Conclusiones por objetivo.....	124

5.1.1 Conclusión del Primer Objetivo: Analizar los requerimientos específicos de la empresa y sus usuarios, con un enfoque en el backend del sistema de información web ..	124
5.1.2 Conclusión del Segundo Objetivo: Desarrollar el backend del sistema de información web para garantizar la eficiencia en el manejo de datos y la integración con la base de datos.....	125
5.1.3 Conclusión del Tercer Objetivo: Implementar el backend del sistema de información web utilizando Laravel y MySQL, integrando las funcionalidades esenciales para la gestión de compras, ventas e inventario.....	126
5.2 Logros y Aportes del Proyecto	127
Capítulo 6. Recomendaciones.....	130
6.1 Trabajos futuros	132
Referencias.....	134
Apéndices.....	138

Lista de Tablas

Tabla 1	Impacto de la Optimización de Consultas en la Latencia	119
Tabla 2	Comparativa de Eficiencia Operativa	120
Tabla 3	Clasificación de Usabilidad del Sistema.....	121

Listado de Figuras

Figura 1 Diagrama de flujo del proceso de compras de la empresa AUDYSOFT	49
Figura 2 Diagrama Entidad-Relación (DER) del sistema.....	55
Figura 3 Código de la migración de la tabla productos	57
Figura 4 Código de la invocación de los seeders.....	58
<i>Figura 5 Seeder de la tabla departamentos.....</i>	<i>59</i>
<i>Figura 6 Seeder de la tabla ciudades</i>	<i>60</i>
Figura 7 Código del Model de productos	62
Figura 8 Código del Request de productos	64
Figura 9 Código del controller de productos	66
Figura 10 <i>Código de la API de productos</i>	<i>68</i>
Figura 11 Código de las Routes Web	70
Figura 12 <i>Código de Routes API</i>	<i>71</i>
Figura 13 <i>Código del home.....</i>	<i>73</i>
Figura 14 <i>Interfaz del Home</i>	<i>74</i>
Figura 15 Fragmento del código del index de productos.....	75
Figura 16 Interfaz del index de productos	76
Figura 17 Fragmento del código del create de productos	76
Figura 18 Interfaz del Create productos.....	77
Figura 19 Fragmento del código del show de productos	77

Figura 20 Interfaz del detalle de productos.....	78
Figura 21 Fragmento del código del edit de productos.....	78
Figura 22 Interfaz de la edición de un producto	79
Figura 23 Código de la migración para la tabla de roles	81
Figura 24 Código de la migración de la tabla de permisos.....	82
Figura 25 Código de la migración de la tabla pivote entre roles y permisos.....	83
Figura 26 Código de la migración pivote que relaciona un Usuario con su Rol asignado	84
Figura 27 Código de la tabla pivote entre usuario y permiso	85
Figura 28 Fragmento del código de la migración de usuarios	86
Figura 29 Código del seeder del usuario privilegiado	87
Figura 30 Fragmento del código del seeder para los roles y permisos	89
Figura 31 Código del model de usuarios	90
Figura 32 Fragmento del código del request del usuario	91
Figura 33 Fragmento de código del controller de usuario	92
Figura 34 Fragmento del código VerificationController	94
Figura 35 Fragmento del código del RegisterController	95
Figura 36 Fragmento del código del login controller	96
Figura 37 Fragmento del código del rol controller	97
Figura 38 Interfaz del Home	99
Figura 39 Interfaz del login.....	99
Figura 40 Interfaz del register.....	100
Figura 41 Email recibido para la confirmación de correo	101
Figura 42 Interfaz del crear de un usuario	101

Figura 43 Interfaz del listado de usuarios	102
Figura 44 Interfaz del edit de usuarios.....	102
Figura 45 Interfaz del listado de roles.....	103
Figura 46 Interfaz para añadir un rol nuevo.....	103
Figura 47 Interfaz para editar un rol ya creado.....	104
<i>Figura 48</i> Variables de entorno	107
Figura 49 Backend desplegado	108
Figura 50 Conexión de AWS S3.....	109
Figura 51 Variables de Conexión del Servicio de Railway	109
Figura 52 Estructura de Tablas de la base de datos en Railway	110
Figura 53 Prueba Funcional de Autenticación 1	111
Figura 54 Prueba Funcional de Autenticación 2.....	112
Figura 55 Prueba de Integridad y Validación de Formulario 1	113
Figura 56 Prueba de Integridad y Validación de Formulario 2	114
Figura 57 Prueba de Integración Funcional.....	115
Figura 58 Optimización de Consultas mediante Eager Loading	116
Figura 59 Manejo Eficiente y Seguro de Archivos Remotos	117
Figura 60 Implementación de Índices para Optimización de Consultas.....	117

Capítulo 1. Implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario en la Empresa AUDYSOFTW

1.1 Planteamiento del Problema

AUDYSOFTW es una empresa informática dedicada al desarrollo de aplicaciones web, especialista en auditoría y consultoría de sistemas, con experiencia en el mercado de las tecnologías de la información. Actualmente, AUDYSOFTW requiere de un sistema de información web que le permita gestionar eficientemente sus procesos de compras, ventas e inventario. La falta de una plataforma digital adecuada a las necesidades específicas de la empresa, provoca dificultades en la organización de datos, la comunicación interna y la atención al cliente (Gupta et al., 2023).

La necesidad actual de la empresa AUDYSOFTW, se basa principalmente en la implementación de un sistema de información web integral, que optimice la gestión de sus procesos internos de las compras, ventas e inventario. En la actualidad, la carencia de una plataforma digital idónea que cumpla todos los requerimientos específicos de una organización, genera obstáculos en la empresa eficaz de datos, entorpece la toma de decisiones administrativas, limita la capacidad de proporcionar un servicio ágil y personalizado a los clientes (Al-Hawari, 2022).

La ausencia de una solución tecnológica adecuada plantea una serie de desafíos significativos para AUDYSOFTW, como la falta de un sistema centralizado de gestión de

adquisiciones, lo que dificulta la identificación de proveedores, la gestión de pedidos y el inventario, lo que provoca retrasos en las compras. De esta misma forma, la creación de ofertas y el seguimiento de pedidos se ven afectados, generando pérdidas de ventas.

Además, la falta de un sistema preciso de seguimiento de inventario, genera problemas como exceso de productos y obsolescencia de los mismos, afectando la capacidad de responder eficazmente a la demanda. “La carencia de soluciones tecnológicas óptimas para una organización, dificulta su capacidad para operar de manera eficiente y satisfacer las demandas del mercado, lo que pone en riesgo su posición competitiva” (Kone tapsoba et al., 2023).

Para afrontar estos desafíos y mejorar la eficiencia operativa de la empresa, es imprescindible implementar un sistema de información web que permita la gestión centralizada de la compras, ventas e inventario. Una plataforma digital robusta y adaptable proporcionará a AUDYSOFTW las herramientas necesarias para optimizar sus operaciones, fortalecer sus relaciones con los clientes y mantener su competitividad en el mercado.

1.2 Formulación del Problema

¿Cómo la implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario ayudara a mejorar la eficiencia operativa y la satisfacción del cliente en la empresa AUDYSOFTW?

1.3 Objetivos

1.3.1 Objetivo General

Implementar un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario en la empresa AUDYSOFTW, optimizando sus procesos internos y mejorando la administración de datos a través de un backend eficiente y estructurado.

1.3.2 Objetivos Específicos

Analizar los requerimientos específicos de la empresa y sus usuarios, con un enfoque en el desarrollo del backend del sistema de información web

Desarrollar el backend del sistema de información web para garantizar la eficiencia en el manejo de datos y la integración con la base de datos

Implementar el backend del sistema de Gestión de Compras, Ventas e Inventario, asegurando su correcto funcionamiento y rendimiento

1.4 Justificación

La implementación de un sistema de información web para la empresa AUDYSOFTW es fundamental para mejorar la eficiencia y la productividad de sus operaciones. Este proyecto, aportara en gran medida un claro incremento, en lo que respecta a la gestión de compras, ventas e Inventario, y generación de informes para de esta manera impulsar el nivel de competitividad de la empresa.

Optimización de procesos internos: La implementación de un sistema de información web permitirá una gestión más eficaz de los datos y procesos internos de AUDYSOFTW, al contar con funciones como la automatización de tareas repetitivas y la centralización de la información, agilizará las operaciones diarias y se reducirán los tiempos de respuesta

Mejora en la toma de decisiones: Con la ayuda de un sistema que facilite el acceso a información relevante y actualizada, los directivos de AUDYSOFTW podrán tomar decisiones más informadas y estratégicas, contribuyendo a la mejora continua de la empresa y a su capacidad para adaptarse rápidamente a los cambios del mercado.

Incremento de la competitividad: En un entorno empresarial cada vez más digitalizado, la implementación de un sistema de información web es crucial para mantenerse competitivo, lo que permitirá a AUDYSOFTW ofrecer un servicio más ágil y eficiente a sus clientes, proporcionando una clara ventaja competitiva.

1.5 Delimitaciones

1.5.1 Geográfica

El área geográfica seleccionada para el desarrollo del proyecto de investigación es la ciudad de Ocaña, Colombia, en la empresa AUDYSOFTW.

1.5.2 Temporal

El proyecto se desarrollará en un periodo de 4 meses, después de aprobado el anteproyecto.

1.5.3 Conceptual

El proyecto se llevará a cabo utilizando PHP como lenguaje de programación principal, aprovechando las características y la robustez del Framework Laravel para la implementación del Sistema de Información Web. Para la gestión de bases de datos, se ha seleccionado MySQL, debido a su fiabilidad y seguridad en el desarrollo web. Además, para la interfaz de usuario del backend, se integrará la plantilla AdminLTE, dado que esta aporta una gran variedad de herramientas que facilitan el desarrollo ágil y eficiente de interfaces de usuario.

1.5.4 Operativa

El sistema estará enfocado en cubrir las necesidades específicas de AUDYSOFTW en cuanto a la gestión de compras, ventas e inventario.

Capítulo 2. Marco Referencial

2.1 Marco Histórico

En este apartado, se mencionan trabajos previos a diferentes niveles, tanto a nivel internacional, como a nivel nacional y local. Los cuales tienen temáticas relacionadas a las abordadas en este proyecto:

2.1.1 Ámbito Internacional

El artículo titulado “Diseño y Desarrollo de una herramienta basada en web para gestionar la información de los pacientes” tiene como principal objetivo resolver fallas comunes en la administración hospitalaria, tales como las limitantes en los diferentes registros en los pacientes, la compleja forma en la que se accede a los registros médicos y falta de orden en los cambios de turno. A lo largo de este trabajo, se habla sobre el desarrollo de varios módulos específicos dentro del software hospitalario, como lo uno destinado solo para gestión de colas de espera, uno para el diseño de reservas, otro para los registros electrónicos de pacientes, recordatorios de citas y uno destinado para las actividades médicas. Cabe resaltar, que para estos módulos fueron diseñados utilizando diferentes, dentro de las cuales se tienen: Laravel (PHP), Tailwind (CSS) y jQuery (JavaScript). Esto con el fin de no solo mejorar la forma en la que se accede a la información, sino con también para poder hacer una mejor optimización de los procesos administrativos, para de esta forma tener una mayor eficiencia y adaptabilidad tanto dentro de las instalaciones de la institución como para las necesidades de la los paciente (Marbella et al., 2024).

Ahora bien, en el trabajo titulado "Un sistema basado en blockchain para mejorar la trazabilidad y rastreabilidad de piezas de aeronaves en la gestión de inventarios" se aborda todo lo relacionado a la gestión de inventarios de repuestos de aeronaves y la importancia de la trazabilidad. A lo largo de este referente, se menciona como la Asociación Internacional de Transporte Aéreo realizó toda una serie de procesos de identificación acerca de las complejidades existentes en las cadenas de suministro de repuestos, así como la deficiencia de la calidad de datos y seguridad de la información. Todo lo anterior con el principal objetivo de crear un sistema basado en blockchain utilizando Hyperledger Fabric y Hyperledger Composer. Con este sistema de información de inventariado, se proporciona una plataforma de gestión que asegura la integridad y visibilidad de la información a través de un libro mayor descentralizado. Por lo cual, dicho sistema de inventariado brinda una clara mejora en la trazabilidad y la seguridad de los datos (Ho et al., 2021)

2.1.2 Ámbito Nacional

En lo que se refiere al ámbito nacional se resalta el artículo titulado "Mapa de Oferta de Justicia en Colombia: Un Caso Práctico de Sistemas de Información Web para el Ciudadano", en este se habla acerca de una aplicación práctica de sistemas de información web que tiene como objetivo brindar servicios a los ciudadanos. En donde se centra específicamente, en los resultados de la creación de una plataforma web destinada a la administración y divulgación de información sobre la oferta de servicios de justicia en Colombia, la cual tiene por nombre "Mapa de Oferta de Justicia". Cabe resaltar que en esta plataforma incluye tres productos de software:

tesauro de oferta de justicia, directorio de instituciones y una plataforma de reportes e indicadores sobre oferta de justicia. Este artículo tiene como principal aporte el demostrar cómo la tecnología de la información, tiene la capacidad para brindar apoyo en áreas de gran importancia para los ciudadanos, como la justicia en Colombia.(Galvis-Lista et al., 2012)

2.1.3 Ámbito Local

En lo referente al ámbito local se resalta el trabajo de grado titulado “Implementación de un Sistema de Información para la gestión de eventos académicos de la Universidad Francisco de Paula Santander seccional Ocaña” se centra en el desarrollo de una plataforma web para optimizar la gestión de eventos académicos de dicha universidad. Esta herramienta está diseñada para cubrir todos los aspectos administrativos necesarios en la organización de eventos, como lo son el permitir a los organizadores crear, gestionar o coordinar desde la planificación inicial hasta la ejecución final. También, tiene un alto enfoque en la poder facilitar a tanto el cuerpo estudiantil como el cuerpo docente, así como otros usuarios varios, diferentes tareas, como el registro de eventos, consulta de programación y recepción de notificaciones.(Ortega Rodríguez & Torres Ortega, 2022)

Este trabajo de grado llamado “Diseño e implementación de un sistema de información para la gestión de los trabajos de grado tomando como piloto al programa de ingeniería de sistema de la UFPS Ocaña” tiene como principal objetivo optimizar y automatizar los procesos involucrados en la administración de los trabajos de grado, ya sea desde su planificación hasta su entrega final en el repositorio institucional. Dentro de lo cual, se incluye la gestión de tareas como la asignación de jurados, la revisión y aprobación de propuestas, y la recopilación de

documentos finales. Además, este sistema tiene como función el mejorar la accesibilidad y la transparencia en el manejo de la información, permitiendo a los usuarios acceder a los datos y realizar gestiones desde cualquier dispositivo y lugar con conexión a internet. Buscando, sobre todo, generar una reducción en lo referente a los errores y tiempos asociados con los procesos manuales, promoviendo una administración más eficiente y una experiencia de usuario más fluida. (Claro Vega & Duran Coronel, 2017)

En el trabajo de grado “Diseño de una nueva versión del Sistema de Información Académica Genesis SIA realizado para el proyecto de Interconectividad de la Universidad Francisco de Paula Santander Ocaña”. Este proyecto tiene como principal objetivo el mejorar la escalabilidad y la mantenibilidad del sistema mediante la adopción de una nueva arquitectura de software que permite un crecimiento más sostenible, para de esta forma poder facilitar el mantenimiento continuo. También, un aspecto fundamental de este proyecto es el rediseño de las interfaces de usuario, las cuales se han optimizado para ofrecer una experiencia más intuitiva, atractiva y accesible, generando que el sistema sea más fácil de usar para estudiantes, profesores y personal administrativo. (Torrado Arenas, 2023)

2.2 Marco Contextual

La ciudad de Ocaña según la información oficial de la (Alcaldía de Ocaña,2024), se encuentra ubicada en el departamento de Norte de Santander. Fue fundada el 14 de diciembre de 1570 por Francisco Fernández de Contreras. Esta ciudad principalmente destaca en su riqueza histórica, esto dado que, durante la época colonial, Ocaña fue una de las provincias más influyentes del país, así como tener una gran participación en la política de la época. Ocaña fue una de las primeras ciudades en contar con un sistema de cable aéreo durante la independencia. Con una población de 129,308 habitantes, conocidos como Ocañeros, sus principales actividades económicas son el comercio y la producción y comercialización de productos agrícolas.

AUDYSOFTW es una empresa colombiana dedicada a ofrecer soluciones informáticas para la web y servicios de auditoría de sistemas. La visión de esta compañía consiste en convertirse en una de las empresas más importantes y con mayor penetración en Colombia en estos campos, impulsando el desarrollo de empresas e instituciones de manera ética y comprometida con la sociedad y el medio ambiente. Esta compañía tiene como principal eje el desarrollo de aplicaciones web, en donde se enfoca en incrementar la competitividad y productividad de sus clientes a través de soluciones innovadoras respaldadas por tecnología de última generación y un equipo profesional comprometido.

Además, la empresa en lo que refiere a la auditoría de sistemas, ofrece servicios especializados para gestionar riesgos, control interno, seguridad de la información y auditoría

integral en procesos de negocios. AUDYSOFTW, tiene como objetivo contribuir a la generación de valor para sus clientes mediante la aplicación de las mejores prácticas y metodologías.

También la compañía tiene un personal altamente capacitado y especializado en tecnologías de la información y desarrollo de software, lo que le permite ofrecer soluciones de alta calidad y brindar soporte técnico y capacitación necesaria a sus clientes. Para complementar lo anterior AUDYSOFTW se basa en una metodología ágil en el desarrollo de software, lo que le permite garantizar la entrega oportuna y eficiente de proyectos. Así como el poder ofrecer servicios de implantación de software, actualizaciones, capacitación técnica y funcional, y soporte técnico tanto presencial como remoto.

2.3 Marco Conceptual

Las siguientes terminologías se deben de tener en cuenta para la elaboración de este marco conceptual, teniendo:

-Laravel: Este es un marco de desarrollo web PHP de código abierto, tiene como principal caracteriza la forma de su sintaxis expresiva y elegante, lo que ayuda y facilita la creación de aplicaciones web tanto robustas como escalables. Además, cuenta con un gran completo conjunto de funciones, que incluye Eloquent ORM, sistema de enrutamiento, Artisan CLI, Blade Templating Engine, autenticación y autorización integradas, sistema de eventos, programación de tareas y pruebas unitarias y cobertura de

código, agiliza el proceso de desarrollo y garantiza la calidad del código. Al contar con una gran variedad de funciones, así como una gran comunidad activa, hace que Laravel sea una poderosa herramienta a la hora de desarrollar aplicaciones web modernas.(Mahmood & Ashour, 2020).

-MySQL: Este es un popular sistema de gestión de bases de datos relacionales (RDBMS) de código abierto, es conocido por su robustez y fiabilidad, lo que genera que su curva de aprendizaje sea mucho llevadera para los desarrolladores. Además, su escalabilidad, rendimiento, seguridad y el soporte activo de la comunidad lo convierten en una solución sumamente confiable, así como rentable, para proyectos de desarrollo web de cualquier tamaño y complejidad. Por lo tanto, la integración de MySQL, ayuda a simplificar las interacciones con las bases de datos, por otro lado, su interfaz fácil de usar y la amplia documentación disponible aumentan aún más su atractivo. Estos factores hacen de MySQL una opción destacada para administrar bases de datos en aplicaciones web modernas.(Łacheciński, 2022).

-PHP: Es un lenguaje de programación interpretado y sin tipo con, el cual cuenta con características rudimentarias de programación orientada a objetos y una sintaxis que recuerda a C y Perl. Así como otros lenguajes de programación, todo lo referente a su semántica está diseñada para facilitar la programación y puede no satisfacer a los teóricos de los lenguajes de programación. El lenguaje es considerablemente más simple que Perl, lo que lo hace más fácil de aprender, depurar y mantener. Aunque no es tan conciso como

Perl, PHP tiene un poder expresivo considerable. Sin embargo, PHP cuenta con una gran biblioteca de funciones y un potente tipo de datos engañosamente denominado "matriz" hacen posible desarrollar aplicaciones web complejas con relativamente poco código.(Prokofyeva & Boltunova, 2017).

-AdminLTE: Es una plantilla de gestión basada en el marco Bootstrap 4. 6, la cual es personalizable para usar de forma libre. Contando con la capacidad para adaptarse a muchas resoluciones de pantalla desde pequeños dispositivos celulares hasta sistemas informáticos masivos de escritorio. Además de esto, la plantilla captura todos los complementos posibles que podrían estar siendo utilizado para desarrollar la aplicación frontend en la web.(Scott et al., 2021)

-API: Este se basa en una serie de reglas y definiciones que permite que diferentes aplicaciones y sistemas informáticos se comuniquen entre sí. Con lo cual, hace de función como un intermediario que define como interactuar con determinado software o plataforma, para de esta forma permitir que los desarrolladores puedan utilizar funciones predefinidas y poder acceder a datos de una manera estructurada y precisa. (Scott et al., 2021)

-Heroku: Consiste en una plataforma en la nube, la cual permite a los desarrolladores implementar, gestionar y escalar aplicaciones web de forma sencilla. Simplificando y por lo tanto facilitando la implementación de desarrollos web (Heroku, 2025).

-Amazon S3 (Simple Storage Service): Este es un servicio que se especializa en el almacenamiento en la nube, en donde dentro de los principales beneficios que brinda, están la alta escalabilidad, disponibilidad de datos, seguridad y rendimiento. Siendo muy usado para guardar de forma persistente diversos tipos de archivos varios, como imágenes, audios, videos, entre otros. (Amazon Web Services, 2025).

-Railway: Es un servicio de infraestructura como servicio, el cual ayuda a automatizar el despliegue tanto de aplicaciones como de bases de datos. En donde, esta plataforma tiene como enfoque la simplicidad, así como el rendimiento, siendo ideal para proyectos que requieren un despliegue ágil y rápido. Esto debido, a que permite su gestión y conexión con la aplicación sin tener la complejidad de configurar un servidor específico de forma manual. (Railway, 2025)

2.4 Marco Teórico

2.4.1 Teorías del desarrollo web con PHP

En el lenguaje de programación PHP una de las teorías fundamentales es la referente a la eficiencia del procesamiento del servidor. PHP es un lenguaje interpretado que se ejecuta en el

servidor antes de enviar el contenido al navegador del usuario, por lo tanto, la eficiencia en el procesamiento de los recursos del servidor, así como la memoria y el tiempo de ejecución, resulta ser de gran relevancia.(Prokofyeva & Boltunova, 2017)

Por otra parte, se tiene la teoría de la modularización, esta hace referencia a la forma de dividir una aplicación en partes más pequeñas y manejables llamadas módulos o componentes. Esta tiene como principal función la reutilización de código, para generar que las aplicaciones sean más fáciles de mantener y ampliar. En donde, dentro de los estándares modernos que usan PHP como Laravel, Symfony y CodeIgniter se basan en esta teoría, proporcionando modelos y herramientas que facilitan la creación de código modular y reutilizable.(Prokofyeva & Boltunova, 2017)

Ahora bien, en lo referente a la teoría de interacción de bases de datos dentro del ámbito de PHP, esta es la manera en cómo las aplicaciones se conectan a bases de datos y consultan y manipulan datos dentro de ellas. Además, dentro de este marco se incluye el uso eficiente de las operaciones CRUD (crear, leer, actualizar, eliminar), así como la optimización de las consultas SQL.(Prokofyeva & Boltunova, 2017)

La teoría UI/UX, hace referencia a la capacidad de producir interfaces que sean intuitivas, rápidas y responsivas. Todo esto con el fin de generar una serie de integraciones de PHP con tecnologías de front-end como HTML, CSS y JavaScript para crear una experiencia de usuario

perfecta, dentro de las cuales en los esquemas modernos que se basan en PHP, como laravel integran en sus marcos de trabajo.(Prokofyeva & Boltunova, 2017)

También, se tiene la teoría de la escalabilidad y la mantenibilidad. Esta se basa en la escritura de que se ayuda a ampliar de manera eficiente el tráfico y las demandas de las aplicaciones. Además, dentro de las prácticas de esta teoría de incluyen refactorización del código, y el uso de patrones de diseño como MVC (Model View Controller), así como la adopción de una arquitectura que admita la extensibilidad sin sacrificar el rendimiento.(Prokofyeva & Boltunova, 2017)

2.4.2 Teorías de los Sistemas de Compra, Venta e Inventario

Dentro de las diversas herramientas tecnológicas en la actualidad, se tienen los sistemas de operaciones comerciales, los cuales están diseñados con el objetivo de en diversos aspectos relacionados con la compra, venta e inventario. Por lo tanto, esto generan la automatización de los procesos como los pagos y la orden de pedidos.(Scott et al., 2021)

Por otro lado, en lo que respecta a la teoría de la gestión de inventario, esta tiene principal mente el optimizar los niveles de inventario para de esta manera bajar los costos que está directamente relacionado con el almacenamiento y la falta de existencias. Todo lo anterior, genera un equilibrio en los costos de almacenamiento, así como en lo que respecta a los pedidos y la escasez, con lo cual se llega a determinar de forma precisa la cantidad óptima de inventario que una empresa debe mantener en inventario.(J. Zheng & Chen, 2020)

También, se tiene la teoría acerca de la de la cadena de suministro, esta se concentra en la gestión integrada de los flujos de materiales, información y dinero a lo largo de toda la cadena de suministro. Todo esto abarca desde los proveedores hasta los clientes finales, en donde se busca mejorar la eficiencia, calidad y flexibilidad en la entrega de productos y servicios, por medio de la colaboración de diferentes partes interesadas.(Scott et al., 2021)

En lo referente a la teoría de la venta y el marketing, se centra en todo lo relacionado con comprender el comportamiento del consumidor, así como en desarrollar estrategias efectivas de ventas y marketing. Todo esto basado en diferentes modelos psicológicos y sociológicos, en donde se analiza las necesidades, deseos y comportamientos de los clientes para diseñar estrategias que maximicen las ventas, así como la fidelización del cliente. (Mei et al., 2022)

Adicionalmente, se tiene La teoría de la demanda y la predicción se ocupa de entender y prever la demanda de los productos y servicios de una empresa. Basada en modelos estadísticos y análisis de tendencias, permite pronosticar la demanda futura y, así, planificar operaciones de compra, producción y distribución de manera eficiente. (J. Zheng & Chen, 2020)

En último lugar, se tiene la teoría de la gestión de compras, esta cuenta como principal eje la gestión eficiente de la adquisición de bienes y servicios necesarios para el funcionamiento de la empresa. Esta teoría tiene fuertes bases en lo referente a la identificación y evaluación de

proveedores, negociación de contratos y condiciones favorables, así como en lo que refiere a la gestión de relaciones para de esta forma garantizar la disponibilidad, calidad y servicios de los productos adquiridos.(Mei et al., 2022)

2.5 Marco Legal

En Colombia, existen diferentes leyes que hay tener en cuenta, ya que estas rigen en una buena parte la forma en la que se desarrolla para la web en este país. En donde dichas leyes cubren diferentes aspectos que van desde el comercio electrónico, la protección de datos, la propiedad intelectual hasta la responsabilidad civil. Dentro de las leyes más relevantes, se tiene:

Ley 1229 de 2008: Protección y Fomento de la Competencia en el Consumo

Abarca lo referente a la protección del consumidor y fomentar un mercado competitivo, la Ley 1229 de 2008 establece a la Superintendencia de Industria y Comercio (SIC) como la autoridad supervisora. La SIC salvaguarda la competencia leal, promueve la eficiencia económica y garantiza a los consumidores dentro del ámbito del comercio electrónico. Además, ejerce el poder de investigar y sancionar prácticas anticompetitivas, garantizando igualdad de condiciones para todos.(Congreso de la República, 2008).

Ley 1955 de 2019: Creación de la Agencia Nacional de Protección de Datos Personales

En defensa del derecho a la protección de datos, la Ley 1955 de 2019 crea la Agencia Nacional de Protección de Datos Personales (ANPD) como la autoridad dedicada a supervisar y salvaguardar la protección de datos personales en Colombia. La ANPD tiene la facultad de investigar y sancionar a las entidades que incumplan la normativa de protección de datos.(Congreso de la República, 2022).

Ley 527 de 1999: Establecimiento del Marco Legal para el Comercio Electrónico

Pionera en el ámbito del comercio electrónico en Colombia, la Ley 527 de 1999, también conocida como Ley de Comercio Electrónico, establece el marco fundamental para la realización de transacciones en línea. Define y regula meticulosamente el uso de mensajes de datos y firmas digitales y allana el camino para un marco regulatorio más amplio que rijan el comercio electrónico en la nación.(Senado de la república, 1999).

Ley 1581 de 2012: Protección Integral de Datos Personales

Consagrando la protección de datos personales, la Ley 1581 de 2012 establece un marco integral para salvaguardar la información personal de las personas en Colombia. Define claramente lo que constituyen datos personales, describe los principios para su manejo e introduce el concepto de Habeas Data. Este derecho faculta a las personas para acceder,

rectificar, cancelar y oponerse al tratamiento de sus datos personales.(Congreso de la República, 2012).

Ley 1753 de 2015: Refuerzo de la Protección de Datos para Proveedores de Servicios de Internet

Reforzando las medidas de protección de datos, la Ley 1753 de 2015 modifica la Ley 527 de 1999, imponiendo nuevas obligaciones a los proveedores de servicios de Internet (ISP) en materia de protección de datos personales. La norma exige a los ISP que implementen medidas de seguridad sólidas para salvaguardar los datos de los usuarios y notificarles de inmediato en caso de violaciones de datos.(Congreso de la República, 2015).

Ley 2108 de 2021: Ley de Internet como Servicio Público Esencial y Universal

Fortaleciendo las salvaguardas de protección de datos, la Ley 2108 de 2021 modifica la Ley 1581 de 2012. Introduce nuevas obligaciones para las entidades que manejan datos personales, incluida la realización de evaluaciones de impacto de la protección de datos y el nombramiento de un oficial de protección de datos.(Congreso de la República, 2021).

Por lo tanto, todas estas leyes, fueron creadas de manera meticulosa para lograr conformar la columna vertebral del panorama legal que rige las páginas web en Colombia, asegurando un equilibrio entre innovación, protección al consumidor y conducta responsable en línea. A medida que el ámbito digital siga evolucionando, estas leyes sin duda se adaptarán y servirán como foco para un entorno en línea seguro y confiable.

Capítulo 3. Diseño Metodológico

3.1 Tipo de investigación

Para este proyecto se eligieron tres tipos de investigación, las cuales son la cuantitativa, descriptiva y documental. En donde la investigación cuantitativa, brinda con minuciosos análisis de datos numéricos, una serie de resultados precisos y generalizables. Por otro lado, la investigación descriptiva, tiene como principal foco un enfoque exploratorio, este ayuda a lograr adentrarse en la profundidad y el significado de los fenómenos. Finalmente, la investigación documental, esta nos brinda acceso a una gran cantidad de documentos históricos, aportando una perspectiva invaluable para comprender nuestro pasado y presente.

Dentro del margen de la investigación cuantitativa, esta se centra en el análisis de datos numéricos, utilizando métodos estadísticos para describir, analizar e interpretar la información recolectada. Además, este tipo de investigación busca resultados objetivos y generalizables, siguiendo un proceso estructurado y empleando instrumentos estandarizados para de esta forma garantizar la confiabilidad de los datos. Dentro de sus ventajas, se tiene la precisión y confiabilidad de los resultados, así como la posibilidad de generalización a poblaciones más amplias y la replicabilidad de los estudios.(Cobo-Sánchez & Blanco-Mavillard, 2020a)

La investigación descriptiva se enfoca en lo que respecta a la observación y el análisis, en donde se busca comprender y describir un fenómeno, evento o grupo de personas en un momento específico. A diferencia de la investigación cuantitativa, que se centra en datos numéricos, la

investigación descriptiva se caracteriza por la flexibilidad y amplitud de los métodos de recopilación de datos obtenidos.(R. Zheng et al., 2024a)

Por otro lado, la investigación documental tiene como principal función el aplicar diferentes técnicas analíticas como lo son la codificación, la clasificación e interpretación para de esta manera poder extraer significado y patrones de datos textuales. Además de complementar y enriquecer otros estudios cuantitativos o descriptivos, este tipo de investigación se destaca por su eficiencia en tiempo y recursos, especialmente cuando el tiempo y los recursos son limitados.(Zhang et al., 2015a)

3.2 Población y muestra

Cuando se hace una investigación existe un concepto clave, este es la población, el cual hace referencia a todo el conjunto de elementos o individuos que comparten características comunes y sobre los cuales se desea realizar un estudio.(Beck, 2024a)

Por otra parte, se tiene el concepto de muestra, este es un subgrupo de la población seleccionado de forma aleatoria o sistemática. Con este se tiene la posibilidad de representar una parte del total de elementos que se estudiarán y servirá como eje para obtener información sobre la población general.(Ortiz et al., 2022a)

3.2.1 Población

La población objetivo del siguiente proyecto estará conformada por personal de la empresa AUDYSOFTW.

3.2.2 Muestra

La muestra será definida por trabajadores de la empresa AUDYSOFTW donde se verificarán las necesidades para el sistema de información.

3.3 Diseño de instrumentos de recolección de la información y técnicas de análisis de datos

El instrumento que se utilizará para la recolección de información en el desarrollo del proyecto de Implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario en la Empresa AUDYSOFTW será una encuesta diseñada para evaluar el nivel de satisfacción con el sistema actual de gestión. La encuesta se centrará en identificar las deficiencias del sistema existente y los aspectos que requieren mejora, con el fin de desarrollar una plataforma web más eficiente y ajustada a las necesidades de la empresa. Este instrumento será aplicado a los empleados de AUDYSOFTW, quienes interactúan directamente con el sistema, para obtener una visión clara de sus experiencias y expectativas con respecto al nuevo sistema.

3.4 Análisis de información

Para el proyecto de Implementación de un Sistema de Información Web para la Gestión de Compras, Ventas e Inventario en la Empresa AUDYSOFTW, se empleará un enfoque metodológico mixto que combina métodos cuantitativos, descriptivos y documentales. Esta

estrategia permitirá obtener una visión integral de las necesidades, expectativas y preferencias de los usuarios del sistema, así como evaluar su impacto en las operaciones de la empresa.

Se diseñará una encuesta en línea dirigida a los empleados, gerentes y demás personal relevante de AUDYSOFTW. La encuesta incluirá preguntas sobre su experiencia actual con los sistemas de gestión de compras, ventas e inventario, su satisfacción con estos procesos, y sugerencias para posibles mejoras en el nuevo sistema. Los datos recopilados se analizarán mediante estadística descriptiva para identificar patrones, tendencias y relaciones entre las variables que puedan influir en la adopción y efectividad del nuevo sistema.

La recolección y análisis de datos se llevarán a cabo respetando los principios éticos de la investigación, garantizando el anonimato y la confidencialidad de la información proporcionada por los participantes. Se solicitará el consentimiento informado a todos los participantes antes de la recolección de datos, asegurando que comprendan los objetivos del estudio y cómo se utilizará la información.

3.5 Fase de Validación y Evaluación

Para demostrar la mejora tanto en la eficiencia operativa y la satisfacción del usuario, se diseñó e incluyó una fase de validación y evaluación posterior a la implementación sistema de

información web. Dicha fase se enfocó en la recopilación de datos cuantitativos y cualitativos para contrastar el estado inicial con el desempeño del sistema final.

3.5.1 Medición de Eficiencia Operativa

En lo que respecta al instrumento para medir la eficiencia operativa, se empleó la técnica de la Observación Directa y el Análisis de Procesos. Se aplicó un enfoque comparativo para cuantificar la reducción de tiempos y errores en las tareas críticas. Todo con la meta de tener un contraste con el desempeño del proceso manual contra el desempeño en el Sistema de Información Web implementado. Por lo tanto, se establecieron y midieron dos métricas clave de desempeño:

1. Tiempo Promedio de Registro de una Orden de Compra (TPROC): Se registró el tiempo promedio (en minutos) que tardó un usuario en completar la tarea crítica de registrar una orden de compra completa, desde el inicio del formulario hasta su envío.
2. Tasa de Error en el Registro de Inventario (TERI): Se cuantificó la frecuencia de errores de digitación o errores de cuadro de inventario por cada muestra representativa de registros, para determinar la efectividad de las validaciones del backend.

Los datos se recolectaron para una muestra de 10 empleados que realizan estas tareas regularmente. Los resultados obtenidos se tabularon y se utilizaron para calcular la mejora porcentual en el Capítulo 4 (Resultados).

3.5.2 Medición de Satisfacción y Usabilidad

Para evaluar la percepción del usuario, se aplicó el System Usability Scale desarrollado por Brooke (1996). Este tipo de test es un instrumento estandarizado de diez preguntas con una escala de Likert de cinco puntos (de "Totalmente en desacuerdo" a "Totalmente de acuerdo"), el cual permite cuantificar la usabilidad percibida del sistema.

Para mantener la coherencia del análisis el instrumento fue aplicado a la misma muestra de empleados que participaron en el diagnóstico inicial. El puntaje final se utilizó para determinar si el sistema implementado supera el umbral de usabilidad aceptable (68 puntos), respondiendo directamente a la variable de satisfacción.

Capítulo 4. Resultados

4.1 Primer Objetivo: Analizar los requerimientos específicos de la empresa y sus usuarios, con un enfoque en el desarrollo del backend del sistema de información web

1. Realizar entrevistas con el personal involucrado con el fin de comprender sus necesidades en términos de procesamiento y administración de datos.
2. Evaluar el flujo de trabajo de la empresa para identificar oportunidades de optimización en la gestión de información y procesos internos.
3. Identificar redundancias o ineficiencias en la manipulación de datos y la comunicación entre procesos internos para mejorar la eficiencia del backend.

Para este primer objetivo, se llevó a cabo un proceso estructurado de recolección y análisis de información. Dicho proceso se enfocó en la creación y aplicación de una encuesta dirigida principalmente al personal clave de la empresa, así como la elaboración de diagramas de flujo de trabajo que permitieron visualizar y documentar los procesos actuales.

En primer lugar, se diseñó una encuesta con el objetivo de poder recopilar información sobre las necesidades del personal dentro de la empresa. En donde, la encuesta estuvo enfocada en identificar los principales desafíos en el procesamiento de la información, como la

redundancia de datos, la falta de integración entre sistemas, así como la dificultad en la consulta y manipulación de la información relevante para las operaciones del negocio.

Después, se procedió a hacer un análisis detallado del flujo de trabajo, todo esto mediante la elaboración de diagramas en base a los procesos internos de la empresa. Esta metodología permitió identificar puntos de mejora tanto en la administración de datos como en detectar las ineficiencias en la comunicación entre distintos procesos. Por lo cual, a través de este análisis visual, se identificaron áreas donde la optimización del backend jugaría un papel clave para mejorar la eficiencia operativa y reducir el tiempo de procesamiento de la información.

Por lo tanto, se logró detectar redundancias en la manipulación de datos y deficiencias en la comunicación entre departamentos. A partir de estos hallazgos, se establecieron estrategias para mejorar la eficiencia del backend, enfocándose en la centralización de la información, la optimización de las consultas a la base de datos y la estructuración de un modelo de datos más eficiente. Estas acciones sentaron las bases para el desarrollo de un backend funcional, alineado con las necesidades operativas de la empresa y diseñado para mejorar el procesamiento de la información en sus procesos internos.

4.1.1 Realizar entrevistas con el personal involucrado con el fin de comprender sus necesidades en términos de procesamiento y administración de datos

Para llevar a cabo esta actividad, se diseñó una encuesta por medio de la herramienta de formularios de Google, esto con el fin de recopilar información clave sobre la gestión de datos

dentro de la empresa. Dicha encuesta, fue aplicada a 10 empleados de diferentes departamentos estando enfocada en analizar los procesos internos relacionados con compras, ventas e inventario.

A continuación, se presenta la lista de preguntas utilizadas en la encuesta:

- ¿Qué departamentos existen en la empresa?
- ¿Cómo se gestiona actualmente el inventario?
- ¿Con qué frecuencia se realiza el inventario físico?
- ¿Cómo se almacenan los productos?
- ¿Cuántos proveedores tiene la tienda?
- ¿Con qué frecuencia se realizan compras a los proveedores?
- ¿Cómo se eligen los proveedores?
- ¿Qué método de pago se utiliza para la compra de productos?
- ¿Cuál es el método principal de venta?
- ¿Cómo se maneja la atención al cliente?
- ¿Cuáles son los principales desafíos que enfrenta la tienda actualmente?

Una vez aplicada la encuesta, se procedió a analizar los resultados obtenidos por pregunta:

1. ¿Qué departamentos existen en la empresa?

Primeramente, como resultado de esta pregunta se identificó que los departamentos más mencionados por los encuestados fueron Ventas (10 menciones), Inventario (9 menciones) y Compras (8 menciones). Además, se mencionan otros departamentos como Finanzas (5), Marketing (4) y Recursos Humanos (3) con menor frecuencia.

De los anteriores datos, se obtuvo que los departamentos de Ventas, Inventario y Compras son los más relevantes, y por lo tanto son lo que tienen una considerable concentración de las operaciones dentro de la empresa. Por lo tanto, esto confirma que el backend debe enfocarse en la integración eficiente de estos tres procesos para optimizar el flujo de información.

2. ¿Cómo se gestiona actualmente el inventario?

Como resultado de esta pregunta se obtuvo que un 40% de los encuestados afirmó que el inventario se gestiona de forma manual, mientras que el 60% restante aún sigue usando como método de registro hojas de cálculo.

Por lo anterior, se tiene que hay una alta dependencia de procesos manuales y hojas de cálculo, con lo cual solo se genera un significativo aumento en lo referente al riesgo de errores y dificulta la trazabilidad de los productos. Por lo cual, la implementación del backend debe priorizar la automatización del control de inventario para reducir estas inconsistencias.

3. ¿Con qué frecuencia se realiza el inventario físico?

Para este resultado se obtuvo un 40% de los empleados realiza el inventario en un periodo de tiempo mensual, pero, por otro lado, las opciones de tiempo de registro de inventario, diario, semanal y trimestral obtuvieron un 20% cada una.

Con el resultado anterior, teniendo en cuenta que mayormente el inventario se registra de forma mensual, se obtiene que el backend debe tener la capacidad de visualizar los resultados a la hora de hacer los registros del inventario. Para de esta manera, mejorar la coherencia en el inventario tanto físico y el digital.

4. ¿Cuántos proveedores tiene la tienda?

Para esta pregunta, se obtuvo que el 60% de los encuestados indicó que la tienda trabaja con entre 5 y 10 proveedores, mientras que el 40% maneja más de 10 proveedores.

De lo anterior, se tiene que debido al tipo de negocio y el volumen de proveedores que maneja la empresa, es totalmente necesario que exista una sección específica que cubra este apartado en el backend. Para que, de esta manera, dicho módulo permita gestionar y registrar a los proveedores de manera eficiente para tomar mejores decisiones.

5. ¿Con qué frecuencia se realizan compras a los proveedores?

De esta pregunta, se obtuvo como resultado que las compras se realizan con una frecuencia predominante de forma semanal (40%) o mensual (30%).

Del resultado anterior, se resalta que este patrón de compra recurrente resalta la necesidad de una herramienta de planificación y gestión de pedidos, es decir que los productos que se obtengan de un proveedor siempre este categorizados y relacionados con dicho proveedor. Por lo tanto, hay una necesidad de backend implementa esta logística de abastecimiento, para mejorar significativamente las operaciones relacionadas con las reservas de productos.

6. ¿Cómo se eligen los proveedores?

En esta pregunta se tuvo un resulta que consistió principalmente que dentro de los criterios de selección de proveedores son el precio (90%) y la calidad de los productos (80%) teniendo la mayor frecuencia dentro de los encuestados.

De anterior resultado, se llegó a la conclusión de que la solución que proponga el backend, debe de tener en consideración una clara centralización de la información de los proveedores en un solo lugar. Para que, de esta manera, la empresa registrar datos clave como nombre, contacto y estado, para que puedan tenerlos a la mano al momento de tomar decisiones.

7. ¿Qué método de pago se utiliza para la compra de productos?

Para esta pregunta, se obtuvo que el método de pago más común para los proveedores son las transferencias bancarias (60%), seguido del efectivo (20%) y la tarjeta de crédito (20%).

De lo cual, se debe de tener en consideración, por lo tanto, que ya hay un alto uso de transferencias bancarias, el sistema debe incluir un registro de transacciones para facilitar el correcto historial de los pagos. Con lo cual, se pueda lograr una clara mejora en el control financiero y reducirá la carga administrativa.

8. ¿Cuál es el método principal de venta?

En esta pregunta, se obtuvo como resultado que el método de venta predominante es la tienda física (60%). Así como, que el otro encontró que el 30% combina ventas físicas con ventas en línea, por otro lado, solo el 10% realiza ventas exclusivamente en línea.

Con el anterior resultado, se deja en claro que el backend debe de tener una adecuada gestión interna, para de esta forma tener un correcto registro de las ventas establecidas. Para que, de esta manera, exista dentro de la empresa una clara centralización del historial de ventas efectuadas, independientemente de si provienen de la tienda física o de canales digitales.

9. ¿Cómo se maneja la atención al cliente?

Para esta pregunta, se deja en claro que el 80% de los encuestados indicó que la atención al cliente se brinda de forma presencial en la tienda. Por lo tanto, esto demuestra que aún falta una

digitalización para este tipo de servicio. Es por esto, que el sistema debe de implementar una herramienta para registrar y gestionar solicitudes, digitalizando este proceso en el futuro.

10. ¿Cuáles son los principales desafíos que enfrenta la tienda actualmente?

Para esta pregunta en concreto, se obtuvieron varios desafíos identificados por los empleados encuestados, en donde la mayoría se relacionan con la falta de un sistema centralizado, errores en el control de inventario debido a registros manuales, y la dificultad para dar seguimiento a los proveedores.

Con los resultados anteriores, se concluye que claramente hay una alta necesidad de un sistema integral que automatice y optimice los procesos de inventario, compras y ventas. Para que, de esta manera, al solucionar esta falencia se obtenga una mejora en lo que respecta a la eficiencia operativa de la empresa.

Luego del análisis exhaustivo de los resultados de la encuesta. Se llegó a identificar que las principales falencias y necesidades de la empresa en la gestión de sus operaciones de compras, ventas e inventario. Confirmando una alta dependencia de procesos manuales y la falta de un sistema centralizado, por lo cual, esto hace que sea válida la necesidad de desarrollar una solución tecnológica.

Establecimiento de la Línea Base de Eficiencia y Satisfacción

El análisis de los resultados de la encuesta de diagnóstico, particularmente la alta dependencia de procesos manuales, como en la pregunta número 2 de la encuesta, así como los desafíos de centralización mencionados en la pregunta 10, permitió establecer la línea base de ineficiencia y baja satisfacción de la empresa AUDYSOFTW.

Esta línea base, se procedió a usarla como punto de partida para la validación del proyecto, formalizando las siguientes referencias para el proceso manual:

- **Eficiencia Operativa:** Se infiere un tiempo promedio de registro de orden de compra elevado de aproximadamente 12.5 minutos en el proceso manual y una alta tasa de error en el registro de inventario de alrededor de 6 errores por cada 50 registros.
- **Satisfacción del Usuario:** Se asume un puntaje bajo de satisfacción con el sistema y los procesos actuales, dada la cantidad de desafíos reportados.

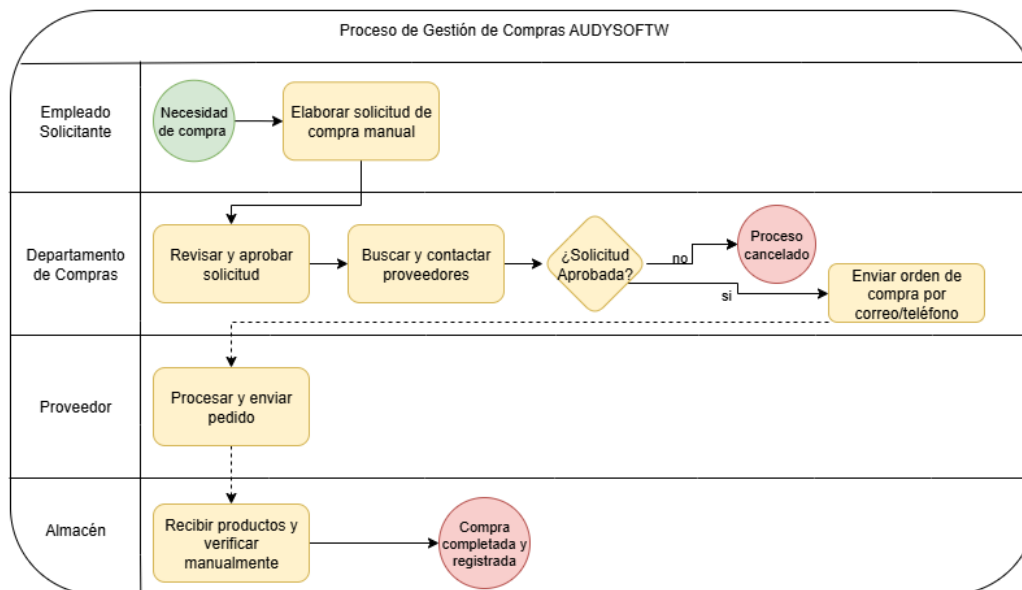
4.1.2 Evaluar el flujo de trabajo de la empresa para identificar oportunidades de optimización en la gestión de información y procesos internos

Para esta actividad en concreto, se realizó un análisis detallado sobre todo el flujo de trabajo de la empresa AUDYSOFTW. Esto con el fin de poder detectar posibles ineficiencias en diferentes áreas, sobre todo en lo que respecta a la administración de información y a la ejecución de los procesos internos. Cabe resaltar, que toda la información analizada de este análisis se basa en los resultados obtenidos en la encuesta aplicada al personal. También, se tuvo en cuenta diferentes observaciones directas sobre el funcionamiento de los principales departamentos involucrados en la gestión de compras, ventas e inventario.

Ahora bien, este estudio se tuvo como principal tarea mapear los procesos actuales mediante diagramas de flujo de trabajo, para que de esta manera haya una forma más clara de visualizar la estructura operativa de la empresa, así como las interacciones entre los distintos departamentos. Con esta gráfica, se logró identificar redundancias en la manipulación de datos, demoras en la actualización de inventarios y problemas de comunicación entre áreas clave, lo que impactaba negativamente en la eficiencia operativa.

Figura 1

Diagrama de flujo del proceso de compras de la empresa AUDYSOFT



Nota: Representación del flujo de trabajo actual del proceso de compras

Dentro de los resultados del análisis del diagrama del flujo de trabajo, se obtuvo varias ineficacias y cuellos de botella en la operación de la empresa. Principalmente, Se observó la existencia de numerosos casos de duplicidad en lo referente al ingreso de datos, esto debido a que la información de compras y ventas se registraba manualmente en distintos documentos. Por lo cual, este tipo de redundancia no solo consume tiempo, sino que también genera inconsistencias y errores en la actualización del inventario.

También, se logró identificar, que hay una notable falta de integración entre los departamentos. En donde, la gestión de compras y ventas no está sincronizada con el control de inventario,

resultando en una planificación de reposición de productos deficiente y en la pérdida de visibilidad sobre los niveles de stock.

Adicionalmente, un hallazgo particularmente preocupante, es la ausencia de un sistema centralizado. Actualmente, la empresa carece de una plataforma unificada que permita gestionar la información de clientes, proveedores e inventario en tiempo real. Toda esta serie de datos dispersos, solo genera problemas y dificultades para realizar consultas y análisis respectivos, impactando duramente en la toma de decisiones a nivel sobre todo de inventario, compras y ventas dentro de la empresa.

Con todo lo anterior, se lograron definir oportunidades de mejora que servirán como la base fundamental para el desarrollo y posterior implementación del backend del sistema de información web. En donde, dicha solución tecnológica propuesta busca, eliminar la duplicidad de datos, así como la dependencia de procesos manuales. Todo esto integrando de forma centralizada la información de las compras, ventas e inventarios en una sola plataforma. Con lo cual, permitir una automatización eficiente de los procesos clave, para de esta manera, obtener una mejora considerable en la gestión de la información, facilitando la toma de decisiones y la eficiencia operativa.

4.1.3 Identificar redundancias o ineficiencias en la manipulación de datos y la comunicación entre procesos internos para mejorar la eficiencia del backend

Una vez se llevó a cabo el análisis del flujo de trabajo de la empresa, en donde se logró identificar deficiencias tanto en la manipulación de datos como en la comunicación entre

procesos internos, se detectaron diversas fallas que atrofian el funcionamiento optimo del sistema actual.

Primeramente, uno de los problemas encontrados era debido a una alta duplicación de datos, los cuales estaban diferentes formatos, ya que la información de compras, ventas e inventario se almacenaba en hojas de cálculo y documentos físicos, generando que muchas veces se encontraban perdida de información, así como inconsistencias de la misma. Adicionalmente, en lo referente a las áreas de Compras, Ventas e Inventario trabajaban de forma aislada, sin un canal de comunicación unificado, por lo cual, esto solo terminaba en retrasos tanto en las actualizaciones de datos y dificultades en la toma de decisiones.

Por otro lado, se logró detectar que existe un gran aislamiento entre los departamentos. En donde, las áreas de Compras, Ventas e Inventario de una forma sumamente individual y sobre todo sin tener un canal de comunicación unificado. Por lo tanto, al tener esta falta de integración, generaba dos problemas, por una parte, retrasos en la actualización del inventario, mientras que, por otra parte, complicaba la toma de decisiones informadas, debido a que no existía una sincronización en tiempo real de los procesos clave.

Adicionalmente, se identificó que hay una alta dependencia de los procesos manuales, impidiendo una trazabilidad eficiente de las operaciones. Debido a que, tareas críticas como la verificación de stock, la generación de reportes y la aprobación de compras se realizaban de forma manual, por lo cual, había un incremento en el tiempo de gestión, los errores, así como en la demora en la reposición de inventario.

Para abordar las deficiencias anteriormente mencionadas, se implementarán diferentes maneras para lograr una mejora en la eficiencia del backend. Dentro de las cuales, se tiene, la implementación de una base de datos centralizada, es para poder consolidar, así como actualizar la información en tiempo real. Además, la automatización de diferentes procesos con el fin de reducir tareas manuales y sobre una adecuada integración de módulos interconectados para mejorar la comunicación entre departamentos.

4.2 Segundo Objetivo: Desarrollar el backend del sistema de información web para garantizar la eficiencia en el manejo de datos y la integración con la base de datos.

1. Diseñar la estructura de la base de datos y definir los modelos de datos necesarios para el almacenamiento y procesamiento eficiente de la información.
2. Desarrollar la lógica de negocio y los módulos del backend para la gestión de compras, ventas e inventario.
3. Establecer mecanismos de seguridad en el backend, incluyendo control de accesos, autenticación de usuarios y protección de los datos almacenados.

Para dar con el cumplimiento de este objetivo, se procedió a desarrollar un backend estructurado, teniendo en cuenta las consideraciones mencionadas en los resultados anteriores, para que, de esta manera, dicho backend optimice de forma adecuada el procesamiento y

almacenamiento de información en la empresa Audysoftw. Para llevar a cabo esto, primeramente, se estableció una base de datos para centralizar los datos relaciones con las compras, ventas e inventario, con lo cual se pueda gestionar de manera eficiente, así como asegurar la integridad y disponibilidad de la información en tiempo real.

Luego, se estructuro y posteriormente se desarrolló toda la lógica de negocio del backend, para integrar de un modo eficiente los módulos específicos de cada área requerida. De esta forma, mejorando tanto la automatización de los procesos clave como mejorar la comunicación entre los distintos departamentos. Adicionalmente, se incluyó ciertos mecanismos de seguridad, con los cuales se pueda dar garantía de aspectos como la autenticación de usuarios, el control de accesos datos, así como la protección de dichos datos almacenados, todo con el objetivo de tener una alta confidencialidad y fiabilidad del sistema.

4.2.1. Diseñar la estructura de la base de datos y definir los modelos de datos necesarios para el almacenamiento y procesamiento eficiente de la información

Ahora bien, para tener una alta certeza en lo que se refiere a un almacenamiento estructurado y eficiente de la información, se tuvo en cuenta una base de datos optimizada que permita gestionar las operaciones de la empresa de manera centralizada. Para esto, lo que se procedió a hacer fue, en primer lugar, definir los modelos de datos para cada módulo del sistema en cuestión. Esto con el objetivo de asegurar aspecto clave, como la correcta relación entre

entidades, disponibilidad de la información, así como su actualización pertinente. Gracias, a todo lo mencionado anteriormente, se garantizó la correcta eliminación de la duplicidad de registros. Con lo cual, se logró mejorar la integridad de los datos, permitiendo la reducción de errores en la administración de compras, ventas e inventario.

Adicionalmente, en lo que respecta el diseño incluyó la creación de tablas clave como usuarios, productos, proveedores, compras y ventas, entre otros. Para que, de esta manera, se pueda integrar sin problemas al esquema de la base de datos. Cabe resaltar, que cada tabla fue cuidadosamente definida con sus respectivos campos, así como sus tipos de datos, esto con el objetivo de asegurar que la información se almacenara de manera precisa. Además, para poder mantener la integridad referencial, así como la consistencia de la información, se procedió a establecer las relaciones entre estas tablas, utilizando claves foráneas. Con lo cual, permitió mediante esta estructura fue fundamental para eliminar la duplicidad de registros—por ejemplo, al referenciar un proveedor_id en la tabla de productos en lugar de repetir todos los datos del proveedor—, lo que a su vez minimizó los errores y facilitó la administración de los datos. En última instancia, este diseño sentó una base sólida para el desarrollo de la lógica de negocio del backend.

Para complementar todo lo anterior mencionado, se elaboró el Diagrama Entidad-Relación (DER) del sistema, en donde se logra presentar de manera gráfica y completa las tablas, así como sus campos y las relaciones establecidas. Con lo que, la Figura 2 muestra este diseño, el

Para esta actividad en concreto, se procedió al desarrollo del backend. Todo esto basado en el diseño del modelo de datos que se hizo en la actividad anterior. Ahora bien, para el desarrollo se utilizó el framework Laravel, apoyado de la herramienta de xamp para correr el proyecto primeramente en local para posteriormente ya poder desplegarlo cuando el proyecto esté en una fase de desarrollo avanzada.

Cabe resaltar, que, para esta actividad, se tuvo como principal enfoque todo lo relacionado con lo que respecta a la lógica de negocio y las funcionalidades necesarias para la gestión eficiente de las operaciones clave de la empresa, como lo son las compras, ventas, inventario y usuarios, entre otros, para de esta forma poder asegurar una conexión tanto estable como segura con la base de datos.

Además, para crear y aplicar cada funcionalidad, se desarrollaron diferentes módulos claves, aplicando MVC; para llevar de una manera más integrar todo el proceso de la construcción del backend, para de esta manera encapsulen su propia lógica de negocio. Cabe resaltar, que para cada módulo fue desarrollado de manera independiente, incluyendo sus propias rutas, controladores, modelos y requests. Todo lo anterior, con el fin de tener una gran facilidad en lo que se refiere al mantenimiento y escalabilidad del proyecto en sí.

A continuación, se presenta proceso de la creación y código de la migración de algunos módulos claves del desarrollo, seguido de los models, request y controller, así como sus routes respectivos.

Figura 3

Código de la migración de la tabla productos

```
database > migrations > 2024_12_02_213903_create_productos_table.php > PHP > CreateProductosTable > up()
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  0 references | 0 implementations
8  class CreateProductosTable extends Migration
9  {
10     0 references | 0 overrides
11     public function up(): void
12     {
13         Schema::create(table: 'productos', callback: function (Blueprint $table): void {
14             $table->bigIncrements(column: 'id');
15             $table->bigInteger(column: 'categoria_id')->unsigned();
16             $table->string(column: 'nombre');
17             $table->text(column: 'descripcion')->nullable();
18             $table->decimal(column: 'precio',total: 8,places: 0);
19             $table->integer(column: 'stock');
20             $table->date(column: 'fecha_vencimiento');
21             $table->string(column: 'img')->nullable();
22             $table->string(column: 'estado');
23             $table->string(column: 'registradopor');
24             $table->timestamps();
25             $table->foreign(columns: 'categoria_id')
26                 ->references(columns: 'id')->on(table: 'categorias');
27         });
28     }
29     0 references | 0 overrides
30     public function down(): void
31     {
32         Schema::dropIfExists(table: 'productos');
33     }
34 }
```

Nota: migración de la tabla productos.

Primeramente, para el desarrollo del backend, se inició con la creación de las migraciones de Laravel. En donde dichas migraciones, son una forma en código de las tablas que se habían definido en el diseño anteriormente mencionado. Esto se hace principalmente, con la

consideración de que, si en algún momento hay cambios importantes, como en algún campo de la propia alguna tabla en concreto o de varias tablas en general, se pueda hacer ese cambio de forma directa y sobre todo más controlado sin afectar la estructura del proyecto entero. De la Figura 3, se tiene un ejemplo de una migración para una tabla concreta llamada productos, la cual tiene diferentes campos para caracterizar de forma correcta los diferentes productos que se puedan agregar en stock dentro del inventario.

Luego, como siguiente paso fue el usar seeders para poblar con ciertos datos claves algunas tablas en específico. Particularmente, las tablas de datos geográficos, como lo son países, departamentos y ciudades, esto usando el consumo de la APIs de los de datos abiertos del Departamento Administrativo Nacional de Estadística (DANE) del Gobierno de Colombia, garantizando la precisión y actualización de la información geográfica, para de esta manera no tener la necesidad de insertar de forma manual cada dato en específico, facilitando la tarea el desarrollo del backend. A continuación, se presenta las figuras 4, 5, 6, en donde se aprecia de mejor forma los seeders para primeramente invocarlos y los seeders específicos por sección.

Figura 4

Código de la invocación de los seeders

```

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

0 references | 0 implementations
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    0 references | 0 overrides
    public function run(): void
    {
        // Llamar al seeder de roles y permisos
        $this->call(class: RolePermissionSeeder::class);

        // Llamar al seeder que crea el usuario administrador
        $this->call(class: AdminUserSeeder::class);

        // Seeders de datos geográficos
        $this->call(class: [
            PaísesSeeder::class,
            DepartamentosSeeder::class,
            CiudadesSeeder::class,
        ]);
    }
}

```

Nota: Este es seeder principal, llamado DatabaseSeeder, el cual tiene la función de cargar de forma inicial el resto de seeder. En donde los invoca de forma secuencial, para que, de esta manera, haya garantía de que la base de datos se inicialice con el orden correcto, manteniendo la integridad referencial antes de cargar datos dependientes

Figura 5

Seeder de la tabla departamentos

```

<?php
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

1 reference|0 implementations
class DepartamentosSeeder extends Seeder
{
    0 references|0 overrides
    public function run(): void
    {
        $url = 'https://geoportal.dane.gov.co/laboratorio/serviciosjson/gdivipola/servicios/departamentos.php';

        $response = Http::get($url);

        if ($response->successful()) {
            $json = $response->json();

            if (isset($json['resultado']) && is_array(value: $json['resultado'])) {
                foreach ($json['resultado'] as $dep) {
                    DB::table(table: 'departamentos')->insert(values: [
                        'pais_id' => 1,
                        'nombre' => mb_convert_case(string: trim(string: $dep['NOMBRE_DEPARTAMENTO']), mode: MB_CASE_TITLE, encoding: 'UTF-8'),
                        'codigo' => trim(string: $dep['CODIGO_DEPARTAMENTO']), // + jeto es clave!
                        'estado' => 'Activo',
                        'registrado_por' => 1,
                        'created_at' => Carbon::now(),
                        'updated_at' => Carbon::now(),
                    ]);
                }

                $this->command->info(string: 'Departamentos insertados correctamente.');
```

Nota: En este seeder se puede apreciar, el uso de la librería de HTTP de Laravel, con la cual se hizo el consumo del *endpoint* de departamentos del DANE. En donde, este procesa la respuesta JSON e inserta los registros directamente en la tabla.

Figura 6

Seeder de la tabla ciudades

```

<?php
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

1 reference | 0 implementations
class CiudadesSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $url = 'https://geoportal.dane.gov.co/laboratorio/serviciosjson/gdivipola/servicios/municipios.php';

        $response = Http::get($url);

        if ($response->successful()) {
            $json = $response->json();

            if (isset($json['resultado']) && is_array(value: $json['resultado'])) {
                foreach ($json['resultado'] as $mun) {
                    $codigoDepartamento = $mun['CODIGO_DEPARTAMENTO'];

                    // Buscar el departamento por código
                    $departamento = DB::table('departamentos')
                        ->where('codigo', operator: $codigoDepartamento)
                        ->first();

                    if ($departamento) {
                        DB::table('ciudades')->insert(values: [
                            'departamento_id' => $departamento->id,
                            'nombre' => mb_convert_case(string: trim(string: $mun['NOMBRE_MUNICIPIO']), mode: MB_CASE_TITLE, encoding: 'UTF-8'),
                            'estado' => 'Activo',
                            'registradopor' => 1,
                            'created_at' => Carbon::now(),
                            'updated_at' => Carbon::now(),
                        ]);
                    } else {
                        $this->command->warn(string: "Departamento con código {$codigoDepartamento} no encontrado.");
                    }
                }

                $this->command->info(string: 'Ciudades insertadas correctamente.');
```

Nota: En este seeder, al igual que el anterior se hace un consumo de la API de municipios del DANE e itera sobre los resultados, pero adicionalmente este tiene como lógica de negocio clave una búsqueda del id del departamento por su código, lo que asegura que cada ciudad se inserte con la clave foránea “departamento_id” correcta, garantizando la integridad referencial entre ambas tablas.

Una vez finalizado el poblamiento inicial de la base de datos mediante los Seeders, el siguiente paso lógico fue la creación de los respectivos models para cada entidad del esquema.

En donde, lo que corresponde a los models, se basan en una arquitectura MVC aplicada este framework que es Laravel, los cuales se encargan de hacer un puente de comunicación con la base de datos, encapsulando la lógica de negocio relacionada con la gestión, así como la manipulación de los datos. A continuación, se presenta la evidencia de la definición de los Modelos clave, donde se especifican las tablas, los campos que se pueden llenar y las relaciones necesarias para el sistema.

Figura 7

Código del Model de productos

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

27 references | 0 implementations
class Producto extends Model
{
    0 references
    protected $table = 'productos';

    0 references
    protected $fillable = [
        'proveedor_id',
        'categoria_id',
        'nombre',
        'descripcion',
        'precio',
        'stock',
        'fecha_vencimiento',
        'img', 'estado',
        'registradopor',
    ];

    0 references
    protected $guarded = [
        'estado', 'registradopor',
    ];

    0 references | 0 overrides
    public function categoria(): BelongsTo
    {
        return $this->belongsTo(related: 'App\Models\Categoria');
    }

    0 references | 0 overrides
    public function proveedor(): BelongsTo
    {
        return $this->belongsTo(related: Proveedor::class, foreignKey: 'proveedor_id');
    }
}
```

Nota: Código del modelo para la entidad de productos

Ahora bien, el model que se presenta en la Figura 7, que en este caso corresponde al de la entidad productos, se encarga principalmente de hacer que el tanto el código del controller como el de la base de datos se comuniquen adecuadamente, para esto, el model actúa como un puente en estos dos, así de esta manera, dicho model hace de función el gestionar y asegurar los datos. En el código usado de ejemplo de la Figura 7, se definen ciertos campos que corresponden a esa entidad en concreto, los cuales son atributos de la tabla pueden ser llenados o modificados por el sistema.

Además, otro punto clave, es el hecho que se establece las relaciones de pertenencia con otras tablas, como Categoría y Proveedor. Generando de esta forma, que dichas relaciones definidas accedan a toda la información asociada (por ejemplo, el nombre de la categoría de un producto) de forma directa y fácil, manteniendo la integridad de los datos.

Por otro lado, para este desarrollo, se priorizó la seguridad y la integridad de los datos, esto mediante la implementación de los requests, esto con el fin de que todas las funcionalidades interactúen con la base de datos de forma correcta. Para lo anterior, el request se encarga de ser un filtro que hace diversas validaciones al inicio de cada proceso, esto con el objetivo de poder asegurar que los datos de entrada, el formato o la unicidad, cumplan con las reglas de negocio antes de ser procesados por el Controlador. La Figura 8, se demuestra el código de ejemplo para el request de productos, en donde, en este se observan algunas validaciones correspondientes principalmente a los POST o PUT, los cuales se encargan de la creación y actualización de un producto en sí.

Figura 8

Código del Request de productos

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

3 references | 0 implementations
class ProductoRequest extends FormRequest
{
    0 references | 0 overrides
    public function authorize(): bool
    {
        return true;
    }

    0 references | 0 overrides
    public function rules(): array
    {
        if (request()->isMethod(method: 'post')) {
            return [
                'categoria_id' => 'required',
                'descripcion' => 'nullable',
                'precio' => 'required',
                'stock' => 'required',
                'fecha_vencimiento' => 'required',
                'img' => 'nullable|mimes:jpg,jpeg,png|max:7000',
                'nombre' => 'required|unique:productos,nombre|regex:/^[a-zA-Z\s-]+$/'
            ];
        } elseif (request()->isMethod(method: 'put')) {
            return [
                'descripcion' => 'nullable',
                'precio' => 'required',
                'stock' => 'required',
                'fecha_vencimiento' => 'required',
                'img' => 'nullable|mimes:jpg,jpeg,png|max:7000',
                'nombre' => 'required|regex:/^[a-zA-Z\s-]+$/'
            ];
        }

        return [];
    }

    0 references | 0 overrides
    public function attributes(): array
    {
        return [
            'categoria_id' => 'categoria',
            'proveedor_id' => 'proveedor'
        ];
    }
}
```

Nota: Este componente gestiona la validación de los datos para las peticiones de creación (POST) y actualización (PUT).

Luego, con el request ya abarcado, lo siguiente en el desarrollo se centró en el Controlador, el cual, es la parte más fundamental de la lógica de negocio. Esto debido, a que tiene como principal función principal, el actuar como intermediario, recibiendo la solicitud del usuario que previamente se verificó gracias al request. De esta forma, el controlador interactúa con el model, según sea necesaria la solicitud, con lo cual, se obtenga después, la respuesta final al cliente. Por lo tanto, el Controlador asegura que la aplicación separe la lógica de la presentación, cumpliendo con el estándar de diseño propuesto.

De lo anterior, se tiene que en la figura 9 se demuestra la implementación del `ProductoController`, el cual encapsula la totalidad de la gestión del ciclo de vida de los productos (CRUD), asegurando la centralización de las reglas de negocio. El código evidencia el uso de métodos RESTful (`index`, `store`, `update`, `destroy`) y la integración de utilidades externas, como el uso de servicios de almacenamiento en la nube (`Storage::disk('s3')`) para la gestión de las imágenes. Además, la aplicación de middleware de permisos mediante el trait `HasPermissionMiddleware` garantiza que solo usuarios autorizados puedan ejecutar las operaciones.

Finalmente, la estructura de cada método está diseñada para ser robusta y tolerante a fallos. La implementación de bloques try-catch y el uso de registros de eventos (`Log::error`) aseguran que el sistema maneje excepciones comunes, como errores de base de datos (`QueryException`) o fallos en la subida de archivos. Esto no solo mejora la experiencia del

usuario (al recibir mensajes claros), sino que también facilita el mantenimiento y la depuración del backend, cumpliendo con los estándares de calidad del desarrollo

Figura 9

Código del controller de productos

```
<?php

namespace App\Http\Controllers;

use App\Models\Producto;
use App\Models\Categoria;
use App\Models\Proveedor;
use Illuminate\Http\Request;
use App\Http\Requests\ProductoRequest;
use Illuminate\Database\QueryException;
use Exception;
use Illuminate\Support\Facades\Log;
use Carbon\Carbon;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Storage;
use App\Traits\HasPermissionMiddleware;

5 references | 0 implementations
class ProductoController extends Controller
{
    use HasPermissionMiddleware;

    0 references | 0 overrides
    public function __construct()
    {
        $this->applyPermissionMiddleware(section: 'productos');
    }

    0 references | 0 overrides
    public function index(): Factory|View
    {
        $productos = Producto::with(relations: ['categoria', 'proveedor'])->get();

        foreach ($productos as $producto) {
            if ($producto->img) {
                $filename = basename(path: $producto->img);
                $producto->public_url = route(name: 'imagen.producto', parameters: $filename);
            } else {
                $producto->public_url = null;
            }
        }

        return view(view: 'productos.index', data: compact(var_name: 'productos'));
    }

    0 references | 0 overrides
    public function create(): Factory|View
    {
        $categorias = Categoria::where(column: 'estado', operator: '=', value: '1')->orderBy('nombre')->get();
    }
}
```

Nota: Este Controlador encapsula toda la lógica de negocio para la gestión de productos, incluyendo las operaciones CRUD, la subida de imágenes a servicios externos y el manejo de excepciones para la solidez del sistema

El desarrollo de la lógica de negocio incluyó la creación de un conjunto de endpoints dedicados a la API, los cuales operan de forma totalmente desacoplada de la interfaz web. El objetivo principal de esta API es dar soporte y facilitar el consumo de datos a clientes externos, como podría ser una futura aplicación móvil de inventario o consulta de productos. Este enfoque asegura que la capa de presentación pueda evolucionar de manera independiente, mejorando la escalabilidad del sistema.

El código del `ProductoController` dentro del namespace `Api` evidencia la aplicación de esta arquitectura. Las funciones principales (`index` y `show`) cumplen con los principios RESTful al centrarse únicamente en la gestión de recursos. En contraste con el controlador web, este controller no devuelve vistas ni realiza redirecciones, sino que retorna exclusivamente respuestas en formato JSON (`response()->json($productos)`), lo que facilita su integración directa con aplicaciones enfocadas en lo móvil.

Finalmente, el controlador demuestra una adaptación específica para la eficiencia móvil y la gestión de contenido externo. El código filtra los productos para mostrar solo los que tienen estado 'activo' y *stock* disponible, reduciendo la carga de datos. Además, se implementó una

lógica clave para formatear las URLs de las imágenes almacenadas externamente en el S3 de AWS, asegurando que la aplicación móvil pueda acceder y visualizar el contenido multimedia del producto de manera correcta y segura a través del método mostrarImagen.

Figura 10

Código de la API de productos

```
<?php
namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Producto;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

7 references | 0 implementations
class ProductoController extends Controller
{
    // Listar productos (index)
    1 reference | 0 overrides
    public function index(): JsonResponse
    {
        $productos = Producto::with(relations: 'categoria')
            ->where(column: 'estado', operator: 1)
            ->where(column: 'stock', operator: '>', value: 0)
            ->select(columns: 'id', 'nombre', 'descripcion', 'precio', 'stock', 'img', 'categoria_id')
            ->get();

        // Formatear respuesta
        $productos = $productos->map(callback: function (TModel $producto): array {
            // Modificar la forma en que se devuelve la URL de la imagen
            $imageUrl = $producto->img ? url(path: 'productos/' . basename(path: $producto->img) . '/imagen') : null;
            return [
                'id' => $producto->id,
                'nombre' => $producto->nombre,
                'descripcion' => $producto->descripcion,
                'precio' => $producto->precio,
                'stock' => $producto->stock,
                'img' => $imageUrl, // Utiliza la ruta correcta para acceder a la imagen
                'categoria' => $producto->categoria->nombre ?? null,
            ];
        });

        return response()->json(data: $productos);
    }

    // Ver detalles de un producto por ID (show)
    1 reference | 0 overrides
    public function show($id): JsonResponse
    {
        $producto = Producto::with(relations: 'categoria')->find(id: $id);

        if (!$producto) {
            return response()->json(data: ['mensaje' => 'Producto no encontrado'], status: 404);
        }
    }
}
```

Nota: Este controlador está diseñado para soportar clientes externos. Filtra y devuelve los productos en formato JSON, e incluye la lógica de ruteo y acceso seguro para la URL de la imagen almacenada externamente.

Por otra parte, se tiene que la routes, la cuales se encargan de que el backend se conecte al usuario con la lógica de los Controladores. Cabe resaltar, que, para esta implementación hay una separación clave en los archivos de rutas principales (web.php y api.php), dando con garantía que haya un correcto uso del principio de responsabilidad única, es decir, el desacoplamiento de la interfaz de usuario y los servicios de datos.

También, la estructura de las rutas web es altamente segura y modular. Para ello, se usaron funcionalidades como el auth y verified, los cuales exigen al usuario estar autenticado e incluso haber confirmado su correo electrónico antes de acceder al panel de control (/home). Adicionalmente, se utiliza el método Route::resource extensivamente para definir automáticamente las siete rutas RESTful, las cuales son necesarias para los módulos principales para la mayoría de módulos, con lo cual se logra hacer una gran simplificación del código, así como su el mantenimiento del código.

Además, para seguir aplicando el principio de responsabilidad única, se hicieron ruta en api.php, esta solo centrándose en todo el soporte a la aplicación móvil. Estas rutas no están

diseñadas para ser accedidas por el navegador, sino que son puntos de entrada de datos, es decir, de los endpoints que devuelven respuestas en JSON. Por lo tanto, al mantener esta separación estricta, se garantiza que la capa de autenticación y seguridad de la web no interfiera con la capa de la API, con lo cual se asegura que el sistema no sea solo escalable, sino que también este tenga la opción de dar servicio a múltiples tipos de clientes de forma simultánea.

Figura 11

Codigo de las Routes Web

```
<?php
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\Storage;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\SliderController;
use App\Http\Controllers\UsuarioController;
use App\Http\Controllers\RolController;
use App\Http\Controllers\PermissionController;
use App\Http\Controllers\CategoriaController;
use App\Http\Controllers\ProductoController;
use App\Http\Controllers\ProveedorController;
use App\Http\Controllers\ClienteController;
use App\Http\Controllers\VentaController;
use App\Http\Controllers\CompraController;
use App\Http\Controllers\EmpleadoController;
use App\Http\Controllers\Auth\VerificationController;

Route::get(uri: '/', action: function () { Factory|View {
    return view(view: 'welcome');
}}->name(name: 'welcome'));

// Rutas de autenticación con verificación de correo
Auth::routes(options: ['verify' => true]);

// Ruta personalizada para verificar el correo (esta es la clave)
Route::get(uri: '/email/verify/{id}/{hash}', action: [VerificationController::class, 'verify'])
->name(name: 'verification.verify');

// Rutas protegidas por autenticación y verificación
Route::middleware(middleware: ['auth', 'verified'])->group(callback: function (): void {

    // Rutas para mostrar imágenes privadas
    Route::get(uri: '/usuarios/imagen/{filename}', action: [UsuarioController::class, 'mostrarImagen'])->name(name: 'imagen.usuario');
    Route::get(uri: '/empleados/imagen/{filename}', action: [EmpleadoController::class, 'mostrarImagen'])->name(name: 'imagen.empleado');
    Route::get(uri: '/productos/{producto}/imagen', action: [ProductoController::class, 'mostrarImagen'])->name(name: 'imagen.producto');

    // Panel de control
    Route::get(uri: '/home', action: [HomeController::class, 'index'])
->middleware(middleware: 'can:ver dashboard')
->name(name: 'home');

    // Sliders
    Route::resource(name: 'sliders', controller: SliderController::class);
    Route::get(uri: 'cambioestadoslider', action: [SliderController::class, 'cambioestadoslider'])->name(name: 'cambioestadoslider');
```

Nota: Este código gestiona la interacción del usuario a través del navegador, asegurando la autenticación y la seguridad basada en sesiones. Utiliza `Route::resource` para implementar el CRUD completo de los módulos.

Figura 12

Código de Routes API

```
<?php
use App\Http\Controllers\Api\VentaController;
use App\Http\Controllers\Api\ProductoController;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\ClienteController;

// Rutas de autenticación
Route::post(uri: '/register', action: [AuthController::class, 'register']);
Route::post(uri: '/login', action: [AuthController::class, 'login']);
Route::get(uri: '/usuarios/{id}', action: [AuthController::class, 'showUser']);
Route::get(uri: '/usuarios', action: [AuthController::class, 'index']);
Route::delete(uri: 'usuarios/{id}', action: [AuthController::class, 'destroy']);

// Rutas de clientes
Route::get(uri: 'clientes', action: [ClienteController::class, 'index']);
Route::get(uri: 'clientes/{cliente}', action: [ClienteController::class, 'show']);
Route::post(uri: 'clientes', action: [ClienteController::class, 'store']);
Route::put(uri: 'clientes/{cliente}', action: [ClienteController::class, 'update']);
Route::delete(uri: 'clientes/{cliente}', action: [ClienteController::class, 'destroy']);

// Rutas de productos
Route::get(uri: 'productos', action: [ProductoController::class, 'index']);
Route::get(uri: 'productos/{producto}', action: [ProductoController::class, 'show']);
Route::post(uri: 'productos', action: [ProductoController::class, 'store']);
Route::put(uri: 'productos/{producto}', action: [ProductoController::class, 'update']);
Route::delete(uri: 'productos/{producto}', action: [ProductoController::class, 'destroy']);

// Rutas de ventas
Route::get(uri: 'ventas', action: [VentaController::class, 'index']);
Route::get(uri: 'ventas/{venta}', action: [VentaController::class, 'show']);
Route::post(uri: 'ventas', action: [VentaController::class, 'store']);
Route::put(uri: 'ventas/{venta}', action: [VentaController::class, 'update']);
Route::delete(uri: 'ventas/{venta}', action: [VentaController::class, 'destroy']);

// Rutas para obtener imágenes de productos
Route::get(uri: 'productos/{producto}/imagen', action: [ProductoController::class, 'mostrarImagen'])->name(name: 'producto.imagen');
```

Nota: Estas rutas proporcionan puntos de entrada de datos (endpoints) que devuelven respuestas JSON y operan de forma desacoplada para el soporte a aplicaciones móviles.

Finalmente, se tiene el apartado correspondiente a la creación de las vistas correspondientes a cada sección. Cada vista, está asociada a su propio model, request y controller, para tener una adecuada gestión y desarrollo de estas se usó el motor de plantillas blade, con las cuales se logra hacer una correcta administración de las interacciones entre el usuario y el sistema, para que de esta manera haya un funcionamiento adecuado. Cabe mencionar, que, en lo referente al código de cada vista no solo se enfoca en el diseño de la interfaz, sino que también utiliza la sintaxis blade para recibir y presentar los datos enviados por los Controladores, gestionar la herencia de plantillas (`@extends('layouts.app')`) y controlar la visibilidad de elementos mediante directivas condicionales.

De lo anterior, el fragmento del código de la vista `home.blade.php` presentado en la figura 13, demuestra una clara integración total entre el frontend y el backend. Todo esto, para que la vista pueda extraer datos clave de múltiples entidades, como lo son los clientes, proveedores, compras y ventas, etc. Adicionalmente, con pequeños resúmenes de las secciones más importantes para resaltar los contadores pertinentes de cada módulo clave.

Una vez abordado el home como punto de inicio, se procede a demostrar la funcionalidad y la arquitectura de la interfaz de usuario módulo por módulo. El análisis se centrará a continuación en exponer cada vista crucial necesaria para la gestión de las operaciones clave. Tomando como ejemplo integral el módulo de Productos, se presentará el código y el resultado visual de las vistas de listado, creación, edición y detalle.

Figura 13

Código del home

```

@extends(view: 'layouts.app')

@section(section: 'title',content: 'Panel De Control')

@section(section: 'content')
<div class="content-wrapper">
  <div class="content-header">
    <div class="container-fluid">
      <div class="row mb-2">
        <div class="col-sm-6">
          <h1 class="m-0">@yield(section: 'title')</h1>
        </div>
      </div>
    </div>
  </div>

  <section class="content">
    <div class="container-fluid">
      <div class="row">
        {{-- Total Clientes --}}
        <div class="col-lg-3 col-6">
          <div class="small-box bg-info">
            <div class="inner">
              <h3>{{ $clientesCount }}</h3>
              <p>Total Clientes</p>
            </div>
            <div class="icon">
              <i class="fas fa-users"></i>
            </div>
            <a href="{{ route(name: 'clientes.index') }}" class="small-box-footer">
              Más Información <i class="fas fa-arrow-circle-right"></i>
            </a>
          </div>
        </div>

        {{-- Total Proveedores --}}
        <div class="col-lg-3 col-6">
          <div class="small-box bg-success">
            <div class="inner">
              <h3>{{ $proveedoresCount }}</h3>
              <p>Total Proveedores</p>
            </div>
            <div class="icon">
              <i class="fas fa-truck"></i>
            </div>
            <a href="{{ route(name: 'proveedores.index') }}" class="small-box-footer">
              Más Información <i class="fas fa-arrow-circle-right"></i>
            </a>
          </div>
        </div>
      </div>
    </div>
  </section>
</div>

```

Nota: Fragmento del código de la vista utiliza la directiva @extends para heredar la plantilla principal y muestra un resumen ejecutivo de las operaciones.

Figura 14*Interfaz del Home*

Nota: Dashboard principal, donde el sistema presenta en tiempo real un resumen ejecutivo de los indicadores clave de negocio (clientes, proveedores, compras y ventas).

Luego de haber abarcado el home de forma demostrativa, el siguiente paso es mostrar los puntos clave de cada vista. Para ello, se usará como se había mencionado anteriormente de ejemplo integral el módulo de productos, se presentará el código y el resultado visual de las vistas de listado, creación, edición y detalle. En donde, el módulo de productos ejemplifica el cumplimiento del patrón CRUD a nivel de interfaz de usuario, al incluir las cuatro vistas necesarias que se presentan a continuación: `index.blade.php`, que es para la visualización y

listado; el create.blade.php, para la inserción de nuevos registros; el edit.blade.php, para la modificación de registros existentes; y el show.blade.php, para la vista detallada de un solo elemento. Con lo anterior, se hace con el fin de tener una estructura mucho más modular y eficiente en lo referente al inventario de la empresa.

Figura 15

Fragmento del código del index de productos

```
@extends(view: 'layouts.app')

@section(section: 'title', content: 'Listado De Productos')

@section(section: 'content')
<div class="content-wrapper">
  <section class="content-header" style="text-align: right;">
    <div class="container-fluid">
    </div>
  </section>

  @include(view: 'layouts.partial.msg')

  <section class="content">
    <div class="container-fluid">
      <div class="row">
        <div class="col-12">
          <div class="card">
            <div class="card-header bg-secondary"
              style="font-size: 1.75rem;font-weight: 500; line-height: 1.2; margin-bottom: 0.5rem;">

              @yield(section: 'title')

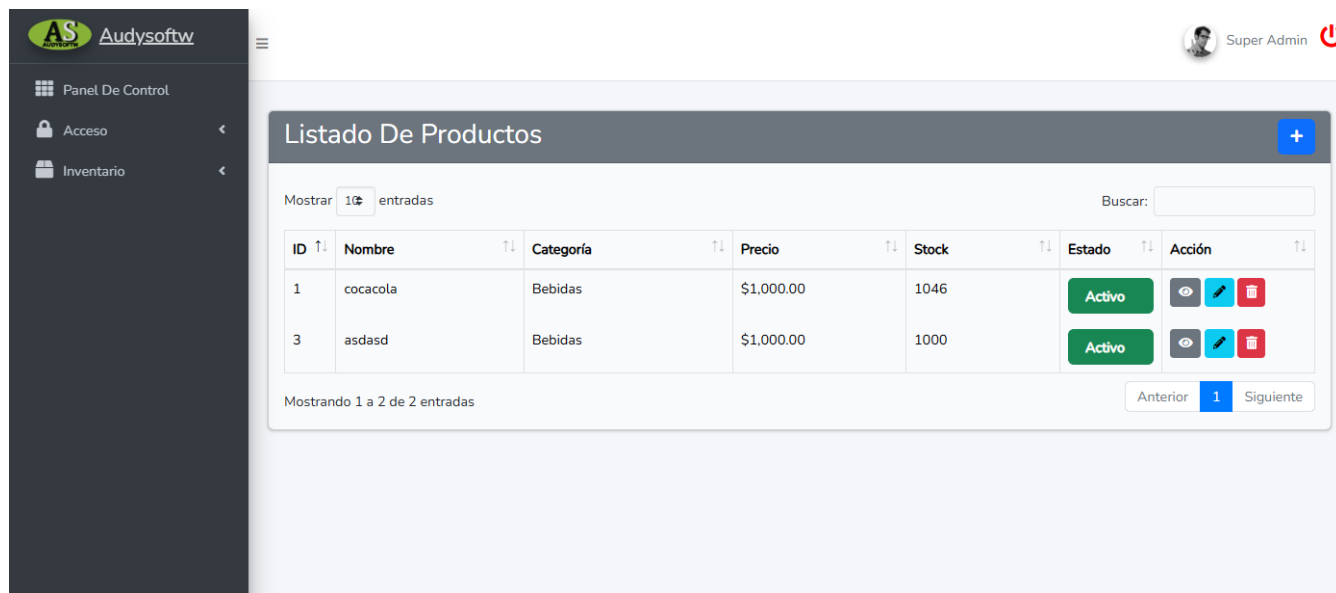
              {{-- Crear Producto: permiso "crear productos" --}}
              @can(abilities: 'crear productos')
                <a href="{{ route(name: 'productos.create') }}"
                  class="btn btn-primary float-right"
                  title="Nuevo">
                  <i class="fas fa-plus nav-icon"></i>
                </a>
              @endcan
            </div>

            <div class="card-body">
              <table id="example1"
                class="table table-bordered table-hover"
                style="width:100%">
                <thead class="text-primary">
                  <tr>
                    <th width="10px">ID</th>
                    <th>Nombre</th>
                    <th>Categoría</th>
                    <th>Precio</th>
                    <th>Stock</th>
                    <th width="60px">Estado</th>
                    <th width="120px">Acción</th>
                  </tr>
                </thead>
                <tbody>
```

Nota: Código de la vista especifica index para ver el listado de los productos creados.

Figura 16

Interfaz del index de productos



Nota: Interfaz del listado de los productos.

Figura 17

Fragmento del código del créate de productos

```
@extends('layouts.app')

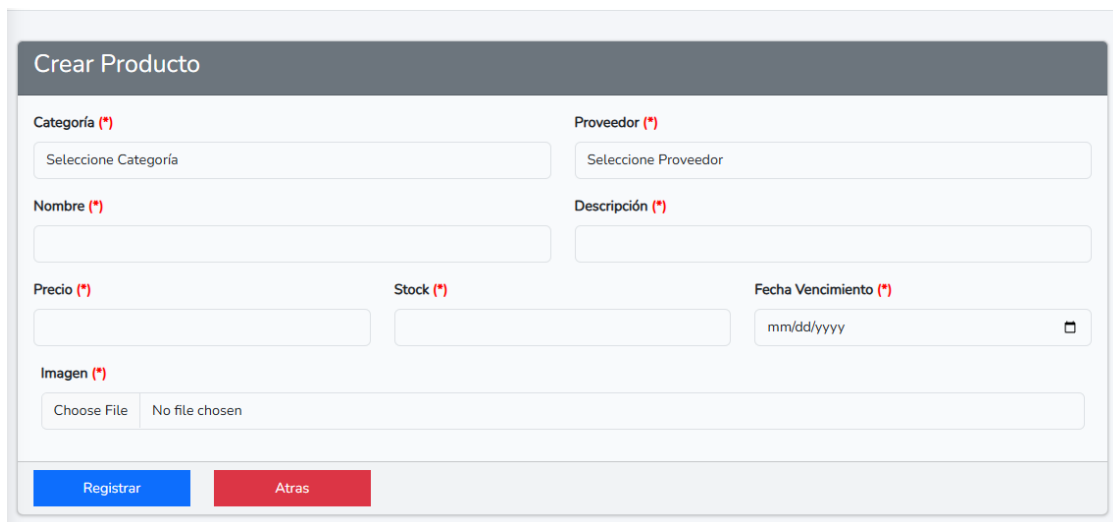
@section('title', content: 'Crear Producto')

@section('content')

<div class="content-wrapper">
  <section class="content-header">
    <div class="container-fluid">
    </div>
  </section>
  @include('layouts.partial.msg')
  <section class="content">
    <div class="container-fluid">
      <div class="row">
        <div class="col-md-12">
          <div class="card">
            <div class="card-header bg-secondary">
              <h3>@yield(section: 'title')</h3>
            </div>
            <form method="POST" action="{{ route(name: 'productos.store') }}" enctype="multipart/form-data">
              @csrf
              <div class="card-body">
                <div class="row">
                  <div class="col-sm-6 col-md-6 col-xs-12">
                    <div class="form-group label-floating">
                      <label class="control-label">Categoría <strong style="color:red;">(*)</strong></label>
                      <select class="form-control" name="categoria_id" id="categoria" required>
                        <option value="">Seleccione Categoría</option>
                        @foreach($categorias as $categoria)

```

Nota: Fragmento del código de la vista del créate para la generación de nuevos productos.

Figura 18*Interfaz del Create productos*


Nota: Interfaz del créate de productos.

Figura 19*Fragmento del código del show de productos*

```

@extends(view: 'layouts.app')

@section(section: 'title', content: 'Detalle de Producto')

@section(section: 'content')
<div class="content-wrapper">
  <section class="content-header">
    <div class="container-fluid">
      <h2>Detalle de Producto</h2>
    </div>
  </section>

  @include(view: 'layouts.partial.msg')

  <section class="content">
    <div class="container-fluid">
      <div class="row">
        <div class="col-12">
          <div class="card">
            <div class="card-header bg-secondary">
              <h3 class="card-title">Producto #{{ $producto->id }} - {{ $producto->nombre }}</h3>
            </div>
            <div class="card-body">
              <div class="row">
                <!-- Información del producto -->
                <div class="col-lg-6 col-sm-12">
                  <p><strong>Categoría:</strong> {{ $producto->categoria->nombre ?? 'Sin categoría' }}</p>
                  <p><strong>Proveedor:</strong> {{ $producto->proveedor->nombre ?? 'Sin proveedor' }}</p>
                  <p><strong>Descripción:</strong> {{ $producto->descripcion }}</p>
                  <p><strong>Precio:</strong> ${{ number_format(num: $producto->precio, decimals: 2) }}</p>
                  <p><strong>Stock Disponible:</strong> {{ $producto->stock }}</p>
                  <p><strong>Fecha de Vencimiento:</strong> {{ \Carbon\Carbon::parse(time: $producto->fecha_vencimiento)->format(format: 'd/m/Y') }}</p>
                  <p><strong>Estado:</strong>
                    @if($producto->estado)
                      <span class="badge badge-success">Activo</span>
                    @else
                      <span class="badge badge-danger">Inactivo</span>
                    @endif
                  </p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
</div>

```

Nota: Código para mostrar en detalle un producto en específico.

Figura 20

Interfaz del detalle de productos

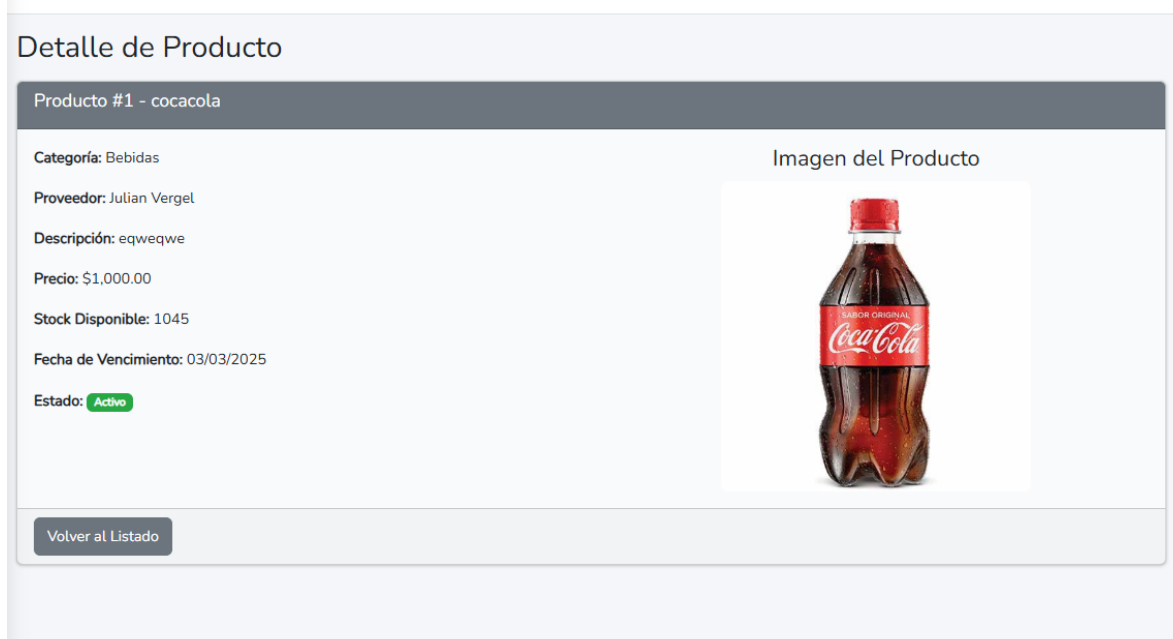


Figura 21

Fragmento del código del edit de productos

```
@extends(view: 'layouts.app')

@section(section: 'title', content: 'Editar Producto')

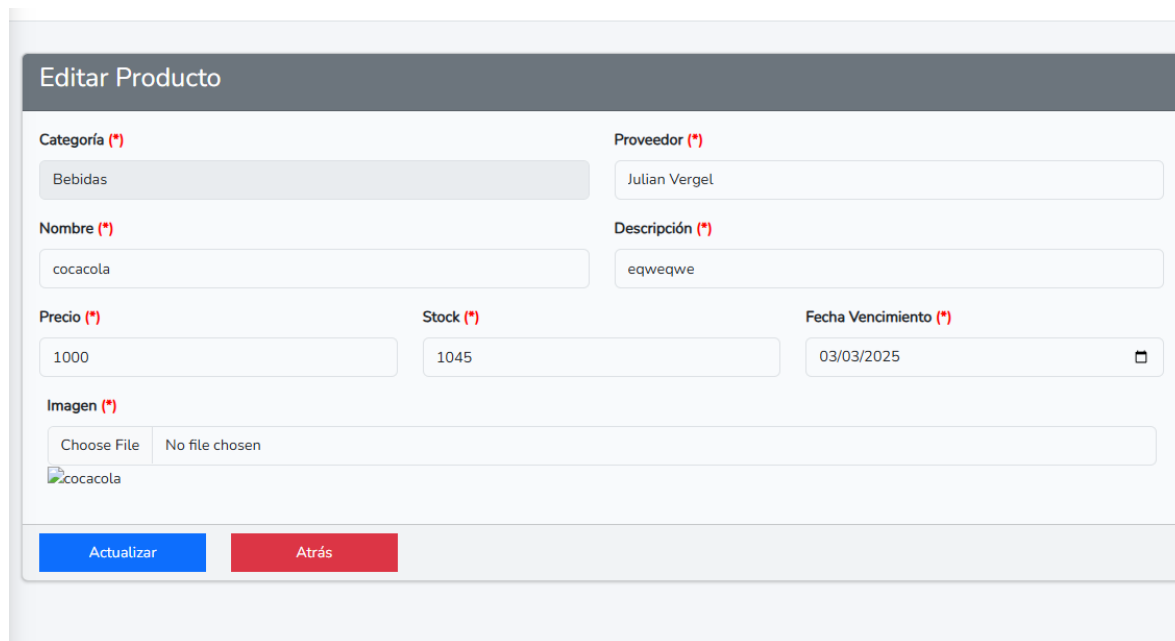
@section(section: 'content')

<div class="content-wrapper">
  <section class="content-header">
    <div class="container-fluid">
    </div>
  </section>
  @include(view: 'layouts.partial.msg')
  <section class="content">
    <div class="container-fluid">
      <div class="row">
        <div class="col-md-12">
          <div class="card">
            <div class="card-header bg-secondary">
              <h3>@yield(section: 'title')</h3>
            </div>
            <form method="POST" action="{{ route(name: 'productos.update', parameters: $producto) }}" enctype="multipart/form-data">
              @csrf
              @method('PUT')
              <div class="card-body">
                <div class="row">
                  <!-- Categoría -->
                  <div class="col-lg-6 col-sm-6 col-md-6 col-xs-12">
                    <div class="form-group label-floating">
                      <label class="control-label">Categoría <strong style="color:red;">(*)</strong></label>
                      <select class="form-control" name="categoria_id" id="categoria" disabled>
                        @foreach($categorias as $categoria)
                          <option {{ $categoria->id == $producto->categoria_id ? 'selected' : '' }} value="{{ $categoria->id }}">{{ $categoria->nombre }}</option>
                        @endforeach
                      </select>
                    </div>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </section>
</div>
```

Nota: Código para editar los datos de un producto previamente creado.

Figura 22

Interfaz de la edición de un producto



Editar Producto

Categoría (*) **Proveedor (*)**

Bebidas Julian Vergel

Nombre (*) **Descripción (*)**

cocacola eqweqwe

Precio (*) **Stock (*)** **Fecha Vencimiento (*)**

1000 1045 03/03/2025

Imagen (*)

Choose File No file chosen

cocacola

Actualizar **Atrás**

Nota: vista de la interfaz que permite editar un producto previamente creado.

En síntesis, para esta actividad, culminó con la construcción modular del backend del sistema, con lo cual se estableció una correcta aplicación del patrón Modelo-Vista-Controlador (MVC). En donde, Se definieron las migraciones para lograr crear una estructura de la base de datos consistente, apoyándose de seeders para la carga inicial de datos geográficos específicos de las ciudades y departamentos en concreto. Además, de una correcta implementación de la lógica de negocio de los modelos, mediante los controladores, así como de asegurar la validación de los datos gracias al uso de los request. Todo lo anterior, demostrado en la vista del interfaz de las secciones claves.

4.2.3 Establecer mecanismos de seguridad en el backend, incluyendo control de accesos, autenticación de usuarios y protección de los datos almacenados

Una vez completada la fase de construcción de la lógica de negocio, la actividad siguiente se centró en la implementación de mecanismos robustos de seguridad y control de acceso. El sistema de información gestiona datos críticos como lo son el inventario, ventas y clientes, entre otros. Por lo tanto, la protección de los datos almacenados y la integridad de las operaciones son prioridades fundamentales. Para de esta manera, se garantiza que solo los usuarios autenticados y debidamente autorizados puedan interactuar con los recursos del backend, blindando la aplicación contra accesos no deseados.

En primer lugar, se tiene que, para esta actividad en concreto, se desarrolló una capa indispensable fue el de la autenticación, con el fin de asegurar que solo usuarios registrados puedan iniciar sesión. Cabe resaltar, que para este mecanismo se ve reforzado en el archivo de rutas, donde la directiva auth se utiliza para proteger todas las funcionalidades principales. Además, para una mayor organización y control, se estableció la autorización mediante un modelo de control de acceso basado en roles. Esto debido, a que, usando este recurso como medio, se logra de definir con precisión qué puede hacer cada usuario después de autenticarse, limitando las acciones permitidas según su rol.

De lo anterior, esta estructura de seguridad se presenta a continuación, en donde se detalla todo lo relacionado con el código del login y su respectivo control acceso de rutas, como la segregación de permisos a través de roles. La implementación de estas medidas de seguridad fue integrada desde el inicio del desarrollo para asegurar la confidencialidad y la correcta manipulación de la información del negocio en cada nivel del backend.

Figura 23

Código de la migración para la tabla de roles

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

0 references | 0 implementations
class CreateRolesTable extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    0 references | 0 overrides
    public function up(): void
    {
        Schema::create(table: 'roles', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'name')->unique();
            $table->string(column: 'guard_name')->default(value: 'web');
            $table->text(column: 'description')->nullable(); |
            $table->timestamps();

            $table->index(columns: 'guard_name');
        });
    }

    /**
     * Revierte las migraciones.
     */
    0 references | 0 overrides
    public function down(): void
    {
        Schema::dropIfExists(table: 'roles');
    }
}
```

Nota: Código de la migración que establece la tabla de roles, en donde el primer paso en la implementación del control de acceso basado en roles.

De la anterior figura 23, se tiene que esta corresponde a la migración de roles, la cual es el punto de partida del mecanismo de autorización. En donde, su principal característica, es crear la tabla roles, está siendo la entidad base que define las agrupaciones lógicas de permisos en el sistema. Cabe resaltar, que, un rol es la agrupación lógica de permisos que se asignará a un usuario. su función principal es simplificar la gestión de la seguridad, ya que, en lugar de asignar múltiples permisos individualmente a un usuario, simplemente se le asigna un rol que ya contiene esos dichos permisos, aplicando el principio de seguridad de mínimo privilegio.

Figura 24

Código de la migración de la tabla de permisos

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'permissions', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'name', length: 125);
            $table->string(column: 'guard_name', length: 125);
            $table->timestamps();

            $table->unique(columns: ['name', 'guard_name']);
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'permissions');
    }
};
```

Nota: Código de la migración de permisos, la cual tiene como principal función definir las acciones atómicas y específicas que se pueden ejecutar en el sistema, como lo son ver solo o interactuar con una o varias secciones en específico.

De la figura 24, se tiene el código correspondiente a los permisos, de los cuales, estos representan las acciones exactas que un usuario puede o no puede realizar. Esto con la finalidad de que el sistema tenga en existencia una clara separación en lo referente del control de acceso. Esto complementando lo anterior en referente a la figura 23, dado que, al tener una serie de permisos diferente, luego con los roles se pueden agrupar dependiendo de la necesidad.

Figura 25

Código de la migración de la tabla pivote entre roles y permisos

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    public function up(): void
    {
        Schema::create(table: 'role_has_permissions', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'permission_id');
            $table->unsignedBigInteger(column: 'role_id');

            $table->foreign(columns: 'permission_id')->references(columns: 'id')->on(table: 'permissions')->onDelete(action: 'cascade');
            $table->foreign(columns: 'role_id')->references(columns: 'id')->on(table: 'roles')->onDelete(action: 'cascade');

            $table->primary(columns: ['permission_id', 'role_id'], name: 'role_has_permissions_permission_id_role_id_primary');

            $table->index(columns: 'permission_id');
            $table->index(columns: 'role_id');
        });
    }

    /**
     * Revierte las migraciones.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'role_has_permissions');
    }
};
```

Nota: Migración para la tabla pivote que relaciona roles con permisos.

De la anterior figura, se tiene que esta hace referencia a un puente entre las tablas de roles y permisos, dado que inicialmente estos dos tienen una relación de muchos a mucho y para estos casos se necesitan este tipo de tablas secundarias. En la cual, la función de esta tabla es hacer que permisos pertenecen a qué Roles. Además, la esta entidad hace que un solo rol herede múltiples permisos, por lo tanto, se pueda generar una adecuada definición de la matriz de autorización completa del sistema.

Figura 26

Código de la migración pivote que relaciona un Usuario con su Rol asignado

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    public function up(): void
    {
        Schema::create('model_has_roles', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'role_id');
            $table->unsignedBigInteger(column: 'model_id');
            $table->string(column: 'model_type');

            // Foreign key solo para role_id
            $table->foreign(columns: 'role_id')->references(columns: 'id')->on(table: 'roles')->onDelete(action: 'cascade');

            // Índices para búsqueda polimórfica (no foreign key directa a users)
            $table->index(columns: ['model_id', 'model_type']);

            // Clave primaria compuesta
            $table->primary(columns: ['role_id', 'model_id', 'model_type'], name: 'model_has_roles_primary');
        });
    }

    /**
     * Revierte las migraciones.
     */
    public function down(): void
    {
        Schema::dropIfExists('model_has_roles');
    }
};
```

Nota: Código de la tabla de pivote entre el usuario y un rol específico puedan ser asociados de forma adecuada. En donde, una vez que un usuario inicia sesión, el sistema consulta esta tabla

para determinar qué roles posee y, por extensión, qué permisos le han sido otorgados a través de las tablas pivote anteriores.

Figura 27

Código de la tabla pivote entre usuario y permiso

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    public function up(): void
    {
        Schema::create(table: 'model_has_permissions', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'permission_id');
            $table->unsignedBigInteger(column: 'model_id');
            $table->string(column: 'model_type');

            // Foreign key a permissions
            $table->foreign(columns: 'permission_id')->references(columns: 'id')->on(table: 'permissions')->onDelete(action: 'cascade');

            // Índices para optimización de búsquedas
            $table->index(columns: ['model_id', 'model_type']);
            $table->index(columns: ['permission_id', 'model_id', 'model_type']);

            // Clave primaria compuesta correcta
            $table->primary(columns: ['permission_id', 'model_id', 'model_type'], name: 'model_has_permissions_primary');
        });
    }

    /**
     * Revierte las migraciones.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'model_has_permissions');
    }
};
```

Nota: En el código de esta migración, se tiene toda una estructura completa sobre la autorización. En donde esta tabla en específico, es decir, `model_has_permissions`, para que, de esta manera, se logre asignar permisos individuales a un usuario sin modificar su rol principal.

Figura 28

Fragmento del código de la migración de usuarios

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    public function up(): void
    {
        Schema::create(table: 'users', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'name');
            $table->string(column: 'email')->unique();
            $table->timestamp(column: 'email_verified_at')->nullable();
            $table->string(column: 'password');
            $table->string(column: 'estado')->default(value: '1');
            $table->string(column: 'photo')->nullable();
            $table->timestamps();

            $table->foreignId(column: 'role_id')
                ->nullable()
                ->constrained(table: 'roles')
                ->onDelete(action: 'set null');
        });

        Schema::create(table: 'password_reset_tokens', callback: function (Blueprint $table): void {
            $table->string(column: 'email');
            $table->string(column: 'token');
            $table->timestamp(column: 'created_at')->nullable();
            $table->primary(columns: ['email', 'token']);
        });

        // Crear la tabla 'sessions'
        Schema::create(table: 'sessions', callback: function (Blueprint $table): void {
            $table->string(column: 'id')->primary();
            $table->foreignId(column: 'user_id')->nullable()->constrained(table: 'users')->onDelete(action: 'set null');
            $table->string(column: 'ip_address', length: 45)->nullable();
            $table->text(column: 'user_agent')->nullable();
            $table->longText(column: 'payload');
            $table->integer(column: 'last_activity')->index();
        });
    }
};
```

Nota: Esta tabla no solo almacena las credenciales del login, sino que también incluye el campo `email_verified_at` para la verificación de identidad y la clave foránea `role_id` para conectarse con el sistema de autorización.

De la figura 28, se tiene un fragmento del código de la migración para la tabla de usuarios, la cual no solo almacena las credenciales de login, sino que también establece la

conexión para la autorización. Cabe recalcar, que en esta tabla se implementa mecanismos de protección clave, como lo son el campo password guarda el valor hashed de la contraseña, así como el campo email_verified_at es utilizado por el middleware para validar la identidad del usuario. Adicionalmente, como se ha mencionado con anterioridad, se debe de incluir la clave foránea role_id, con el fin de tener una una conexión directa con la tabla de roles, permitiendo, que se pueda determinar de manera satisfactoria que los permisos del usuario inmediatamente después de una autenticación exitosa, asegurando que el proceso de autorización sea fluido y seguro.

Figura 29

Código del seeder del usuario privilegiado

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\User;
use Spatie\Permission\Models\Role;
use Illuminate\Support\Facades\Hash;
use Carbon\Carbon;

1 reference | 0 implementations
class AdminUserSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        // Buscar el rol "super-admin" creado previamente por RolePermissionSeeder
        $superAdminRole = Role::where(column: 'name', operator: 'super-admin')->first();

        if (!$superAdminRole) {
            $this->command->error(string: 'El rol "super-admin" no existe. Ejecuta primero RolePermissionSeeder.');
```

Nota: Este Seeder asegura que el sistema siempre cuente con una cuenta de Super Administrador funcional al momento del despliegue. El código asigna automáticamente el rol de más alto privilegio y establece el indicador `email_verified_at` para otorgar acceso inmediato a la capa protegida

El `AdminUserSeeder` es un componente de seguridad crítico en la inicialización del sistema. Su función es garantizar la creación de una cuenta de Super Administrador con credenciales predefinidas y seguras. Lo más importante es que, inmediatamente después de la creación, el código asigna el rol de más alto privilegio (`super-admin`) a esta cuenta y marca la cuenta como verificada (`email_verified_at = Carbon::now()`). Esto garantiza que el sistema siempre tenga un punto de acceso autorizado y plenamente funcional para la configuración inicial de roles y permisos.

El `RolePermissionSeeder` es el script encargado de poblar las tablas de seguridad (Roles y Permisos) con la política de acceso de la empresa. El código itera sobre todas las secciones del sistema (productos, ventas, compras) y genera automáticamente los permisos básicos (ver, crear, editar, mostrar), garantizando que no falte ninguna acción. Posteriormente, crea los roles principales (`super-admin`, `cliente`, `vendedor`) y asigna los permisos mediante la función `syncPermissions`. Por ejemplo, el rol `vendedor` recibe únicamente los permisos de los módulos de ventas y compras, cumpliendo así con el principio de mínimo privilegio y limitando el alcance de sus acciones dentro del sistema

Figura 30

Fragmento del código del seeder para los roles y permisos

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Spatie\Permission\Models\Role;
use Spatie\Permission\Models\Permission;

1 reference | 0 implementations
class RolePermissionSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        // Secciones del sistema
        $secciones = [
            'home',
            'sliders',
            'usuarios',
            'roles',
            'permisos',
            'categorias',
            'productos',
            'proveedores',
            'clientes',
            'ventas',
            'compras',
            'empleados',
        ];

        // Acciones comunes
        $acciones = ['ver', 'crear', 'editar', 'mostrar'];

        // Arreglo para gestionar permisos agrupados por sección
        $permisosPorSeccion = [];

        foreach ($secciones as $seccion) {
            foreach ($acciones as $accion) {
                $nombrePermiso = "{$accion} {$seccion}";

                $permiso = Permission::firstOrCreate(attributes: [
                    'name' => $nombrePermiso,
```

Nota: Se encarga de la creación sistemática de todos los permisos y roles necesarios para las secciones del sistema, y establece las reglas de asignación de privilegios iniciales

Ahora bien, como siguiente paso es hacer todo el desarrollo con respecto al usuario empezando primeramente por el model esta entidad. Este usa principalmente el mustverifyemail, con el fin de garantizar asegurar que haya una verificación de identidad después del registro, al

mismo tiempo teniendo el uso del trait `HasRoles` para habilitar el control de acceso basado en roles, con cual habilite que el backend pueda consultar los permisos y roles del usuario en tiempo de ejecución.

Figura 31

Código del modelo de usuarios

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Spatie\Permission\Traits\HasRoles;
use Illuminate\Support\Facades\Storage;

38 references | 0 implementations
class User extends Authenticatable implements MustVerifyEmail
{
    use HasFactory, Notifiable, HasRoles;

    0 references
    protected $fillable = [
        'name',
        'email',
        'password',
        'photo',
        'estado',
    ];

    0 references
    protected $hidden = [
        'password',
        'remember_token',
    ];

    0 references
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    0 references | 0 overrides
    public function getPhotoUrlAttribute(): string|null
    {
        return $this->photo ? Storage::url(path: $this->photo) : null;
    }
}
```

Nota: Código del modelo del usuario.

Figura 32

Fragmento del código del request del usuario

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Validation\Rule;

3 references | 0 implementations
class UsuarioRequest extends FormRequest
{
    0 references | 0 overrides
    public function authorize(): bool
    {
        return true;
    }

    0 references | 0 overrides
    public function rules(): array
    {
        return [
            'name' => 'required|string|max:255',
            'email' => [
                'required',
                'email',
                Rule::unique(table: 'users')->ignore(id: $this->route(param: 'user')),
            ],
            'password' => $this->isMethod(method: 'post')
                ? 'required|string|min:6|confirmed'
                : 'nullable|string|min:6|confirmed', // Debería ser nullable en la actualización
            'password_confirmation' => $this->isMethod(method: 'post') ? 'required|string|min:6' : 'nullable|string|min:6',
            'photo' => 'nullable|image|mimes:jpg,png,jpeg|max:5048',
            'estado' => 'required|in:0,1',
            'roles' => 'required|array',
            'roles.*' => ['exists:roles,id'],
        ];
    }

    0 references | 0 overrides
    public function messages(): array
    {
        return [
            'name.required' => 'El nombre es obligatorio.',
            'email.required' => 'El correo electrónico es obligatorio.',
            'email.email' => 'Debe ser un correo válido.',
            'email.unique' => 'Este correo ya está registrado.',
            'password.required' => 'La contraseña es obligatoria en la creación.',
            'password.min' => 'La contraseña debe tener al menos 6 caracteres.',
        ];
    }
}
```

Nota: En este fragmento del código del request, se logra tener una adecuada integridad y unicidad de los datos, referente a la creación de los roles y permisos.

Como siguiente paso, una vez aplicadas las medidas de seguridad en el model del usuario fue pasar al request de esta entidad. En donde en la figura 32, se demuestra el uso del from request, esto con el fin de que haya garantía de que la información que llegue a la base de datos, sea válida y cumpla con las reglas de negocio, con lo cual se logre en mayor medida el prevenir la mayor cantidad de errores, así como los conflictos de unicidad. Por lo tanto, esto siendo de gran importancia, dado que un dato incorrecto a nivel de rol podría comprometer la autorización de múltiples usuarios.

Figura 33

Fragmento de código del controller de usuario

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Spatie\Permission\Models\Role;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
use Carbon\Carbon;
use App\Http\Requests\UsuarioRequest;
use Illuminate\Support\Facades\Storage;
use App\Traits\HasPermissionMiddleware;

8 references | 0 implementations
class UsuarioController extends Controller
{
    use HasPermissionMiddleware;

    0 references | 0 overrides
    public function __construct()
    {
        $this->applyPermissionMiddleware(seccion: 'usuarios');
    }

    0 references | 0 overrides
    public function index(): Factory|View
    {
        $users = User::all();

        foreach ($users as $user) {
            if ($user->photo) {
                $filename = basename(path: $user->photo);
                $user->public_url = route(name: 'imagen.usuario', parameters: $filename);
            } else {
                $user->public_url = null;
            }
        }

        return view(view: 'users.index', data: compact(var_name: 'users'));
    }

    0 references | 0 overrides
    public function create(): Factory|View
    {
        $roles = Role::all();
        return view(view: 'users.create', data: compact(var_name: 'roles'));
    }
}
```

Nota: En este controller, se utiliza el trait `HasPermissionMiddleware` en el constructor para proteger todas las rutas del módulo y ejecuta la función `$user->syncRoles()` en los métodos tanto `store` como `update` para asignar y actualizar los roles del usuario en la base de datos.

De la anterior figura, se tiene que, en esta parte del proyecto, se centraliza la seguridad del sistema, específicamente en el controlador del usuario como se observa en la figura 33, que es responsable de la aplicación del modelo RBAC en tiempo de ejecución. En el constructor, se usa el trait `HasPermissionMiddleware` para proteger automáticamente todas sus rutas, exigiendo el permiso `ver usuarios` antes de que se ejecute cualquier acción. Lo más importante es que, en los métodos `store` y `update`, el código utiliza la función `$user->syncRoles($roleNames)` para asignar y sincronizar los roles seleccionados por el administrador con el usuario, asegurando que el perfil de acceso del usuario se ajuste inmediatamente a la política de seguridad definida.

Una vez se teniendo listo el punto anterior. Lo que se procedió a hacer, fue el hacer uso de un de la verificación de email para gestionar el paso de seguridad final de la autenticación. cuando el usuario hace clic en el enlace enviado a su correo, este controlador verifica el token. al llamar a `$user->markemailsverified()`, el sistema actualiza la marca de tiempo `email_verified_at` en la base de datos, lo que permite al usuario pasar el middleware `verified` que protege la mayoría de las rutas. este mecanismo valida la propiedad del correo y el estado de la cuenta, siendo un requisito indispensable para la seguridad del sistema. Luego de esto se procede a mostrar la verificación en código.

Figura 34*Fragmento del código VerificationController*

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\VerifiesEmails;
use Illuminate\Http\Request;
use Illuminate\Auth\Events\Verified;
use Illuminate\Support\Facades\Auth;
use App\Models\User;

2 references | 0 implementations
class VerificationController extends Controller
{
    use VerifiesEmails;

    1 reference
    protected $redirectTo = '/home';

    0 references | 0 overrides
    public function __construct()
    {
        $this->middleware(middleware: 'throttle:6,1')->only(methods: 'verify', 'resend');
    }

    1 reference | 0 overrides
    public function verify(Request $request): RedirectResponse
    {
        $user = User::findOrFail(id: $request->route(param: 'id'));

        if ($user->hasVerifiedEmail()) {
            Auth::login(user: $user);
            return redirect(to: $this->determineRedirect(user: $user))->with(key: 'verified', value: true);
        }

        if ($user->markEmailAsVerified()) {
            event(args: new Verified(user: $user));

            $user->estado = 1;
            $user->save();

            $user->forgetCachedPermissions(); // Si estás usando Spatie
        }

        Auth::login(user: $user);
    }
}

```

Nota: Este controlador implementa la verificación de correo electrónico, un mecanismo de seguridad de dos pasos que exige la confirmación de identidad antes de otorgar acceso total al panel de control

Lo siguiente fue el registercontroller, el cual es fundamental para la seguridad por dos razones principales. primero, asegura la integridad del nuevo registro mediante el validator, que exige unicidad en el correo y una contraseña mínima de 8 caracteres cifrada con `hash::make()`. segundo, cumple con el sistema de autorización (rbac) al asignar el rol 'cliente' por defecto. esto garantiza que cualquier usuario que se registre no sea un "usuario sin rol" y que solo pueda acceder a las funcionalidades mínimas establecidas para ese perfil. finalmente, dispara el evento `registered`, iniciando el proceso de verificación por correo electrónico.

Figura 35

Fragmento del código del RegisterController

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Auth\Events\Registered;
use App\Models\User;
use Spatie\Permission\Models\Role;
use Illuminate\Foundation\Auth\RegistersUsers;

0 references | 0 implementations
class RegisterController extends Controller
{
    use RegistersUsers;

    0 references
    protected $redirectTo = '/home';

    0 references | 0 overrides
    public function __construct()
    {
        $this->middleware(middleware: 'guest');
    }

    /**
     * Validar los datos de registro.
     */
    0 references | 0 overrides
    protected function validator(array $data): Validator
    {
        return Validator::make(data: $data, rules: [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }

    /**
     * Crear al usuario.
     */
    0 references | 0 overrides
    protected function create(array $data): Model\User
    {
        $user = User::create(attributes: [
            'name' => $data['name'],
            'email' => $data['email'],
        ]);
    }
}
```

Nota: Este controlador gestiona el registro de nuevos usuarios, encriptando inmediatamente la contraseña y, crucialmente, asignando automáticamente el rol 'cliente' al nuevo usuario.

Figura 36

Fragmento del código del login controller

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

0 references | 0 implementations
class LoginController extends Controller
{
    use AuthenticatesUsers;

    0 references | 0 overrides
    public function __construct()
    {
        $this->middleware(middleware: 'guest')->except(methods: 'logout');
    }

    /**
     * Este método se ejecuta justo después del login exitoso.
     */
    0 references | 0 overrides
    protected function authenticated($request, $user): Redirector|RedirectResponse
    {
        // Limpiar caché de permisos y roles
        $user->forgetCachedPermissions();

        // Recargar el usuario con relaciones
        auth()->setUser(user: $user->fresh()->load('roles', 'permissions'));

        // Buscar el primer permiso que comience con "ver "
        foreach ($user->getAllPermissions() as $permiso) {
            if (str_starts_with($permiso->name, 'ver ')) {
                // Extraer el nombre del recurso, por ejemplo "ver compras" => "compras"
                $recurso = str_replace(search: 'ver ', replace: '', subject: $permiso->name);
                return redirect(to: "/$recurso");
            }
        }

        // Si no tiene ningún permiso que empiece por "ver", lo mandamos a una vista segura
        return redirect(to: '/home');
    }
}
```

Nota: Código del controller del login la autenticación del rol correspondiente.

Teniendo el register y la verificación lista, lo siguiente es hacer todo el desarrollo referente al método `authenticated()` en el `logincontroller` es la última etapa del flujo de seguridad antes de conceder acceso al sistema. la cual tiene una doble funcionalidad: primeramente, asegura que la información de seguridad del usuario sea precisa y actual, forzando la limpieza de la caché y la recarga de sus roles y permisos (`$user->load('roles', 'permissions')`). mientras que por otro lado se crea una lógica de redirección basada en la autorización, enviando al usuario a la primera sección que tenga permiso para ver.

Figura 37

Fragmento del código del rol controller

```
<?php
namespace App\Http\Controllers;

use App\Models\Role;
use App\Models\Permission;
use Illuminate\Http\Request;
use App\Http\Requests\RoleRequest;
use App\Traits\HasPermissionMiddleware;

2 references|0 implementations
class RolController extends Controller
{
    use HasPermissionMiddleware;

    0 references|0 overrides
    public function __construct()
    {
        $this->applyPermissionMiddleware(section: 'roles');
    }

    0 references|0 overrides
    public function index(): Factory|View
    {
        $roles = Role::all();
        return view('roles.index', data: compact('var_name: 'roles'));
    }

    0 references|0 overrides
    public function create(): Factory|View
    {
        $permissions = Permission::all()->groupBy(groupBy: function (Permission $perm): string {
            return explode(separator: ' ', string: $perm->name)[1] ?? 'otros';
        });

        return view('roles.create', data: compact('var_name: 'permissions'));
    }

    0 references|0 overrides
    public function store(RoleRequest $request): RedirectResponse
    {
        $role = Role::create(attributes: [
            'name' => $request->name,
            'guard_name' => $request->guard_name,
            'description' => $request->description,
        ]);

        $role->syncPermissions(permissions: $request->permissions);
    }
}
```

Nota: Este controlador es fundamental para la gestión dinámica del RBAC. Aplica el middleware de permisos en su constructor para proteger sus rutas y utiliza `$role->syncPermissions()` para asignar los permisos a los roles de forma segura y eficiente.

Cabe resaltar que para que, en este caso, el modelo de seguridad sea funcional, el sistema debe permitir a un usuario autorizado (el administrador) gestionar los roles y permisos de forma dinámica. En lo que respecta al controlador de la figura 37, este se encarga de hacer un intermediario de la lógica de negocio y las tablas de seguridad. Por lo tanto, el rol se inicia con la protección de todas sus funcionalidades, usando `use HasPermissionMiddleware` para garantizar que solo quienes tienen permiso explícito para ver roles puedan acceder a la gestión de la seguridad. Además, de tener el store, esto con el fin de cuando se crea un nuevo rol, se ejecute la función para asignar la matriz de permisos que definimos previamente, manteniendo la consistencia de la autorización.

Una vez demostrada la lógica de la Autenticación y la Autorización a nivel de backend, la fase final de esta actividad se centra en validar visualmente la correcta aplicación de estas políticas en la Interfaz de Usuario. Las vistas del sistema no solo sirven para la interacción, sino que son la prueba fehaciente de la integración de la seguridad y el flujo de trabajo. Las interfaces de Gestión de Usuarios y Roles son críticas, pues es donde el administrador ejecuta las acciones del modelo RBAC, asignando, modificando y visualizando los privilegios de cada cuenta.

Las siguientes figuras presentan las vistas clave que reflejan los diferentes niveles que se trataron en el desarrollo del backend, como lo es: el panel de control (como punto de acceso restringido), el listado de usuarios (que muestra los roles activos) y los formularios de creación y edición de roles y usuarios. En cada una de estas vistas, el sistema utiliza las directivas `@can` para restringir la visibilidad de botones y enlaces, asegurando que el usuario solo pueda ver el contenido y las opciones para las cuales tiene el permiso explícito. Esta presentación visual finaliza el ciclo de desarrollo, demostrando la operatividad completa del sistema de información.

Figura 38

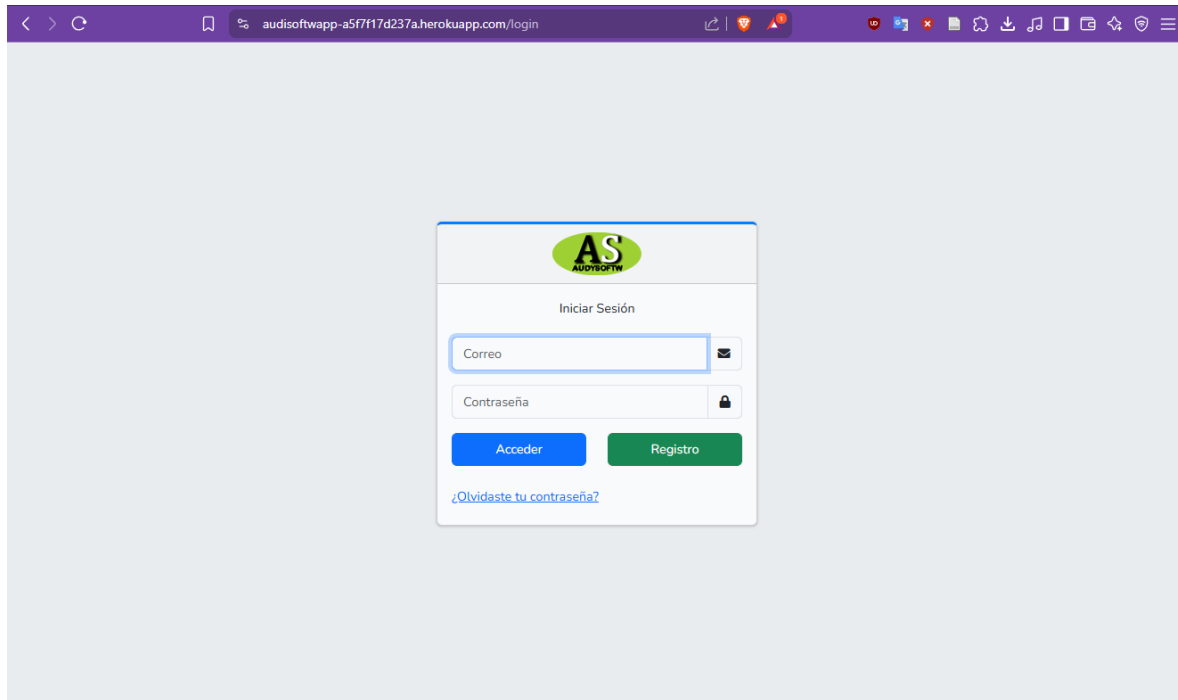
Interfaz del Home



Nota: Interfaz preliminar que encontrara todo usuario para acceder al sistema.

Figura 39

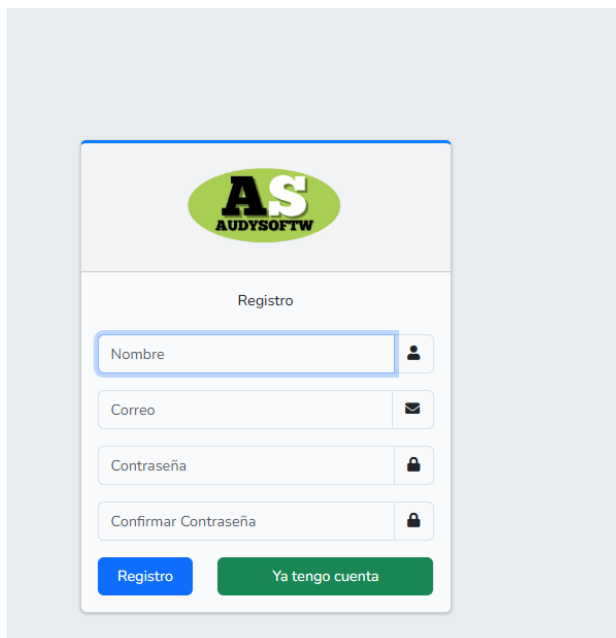
Interfaz del login



Nota: segunda pantalla que vera un usuario previamente registrado.

Figura 40

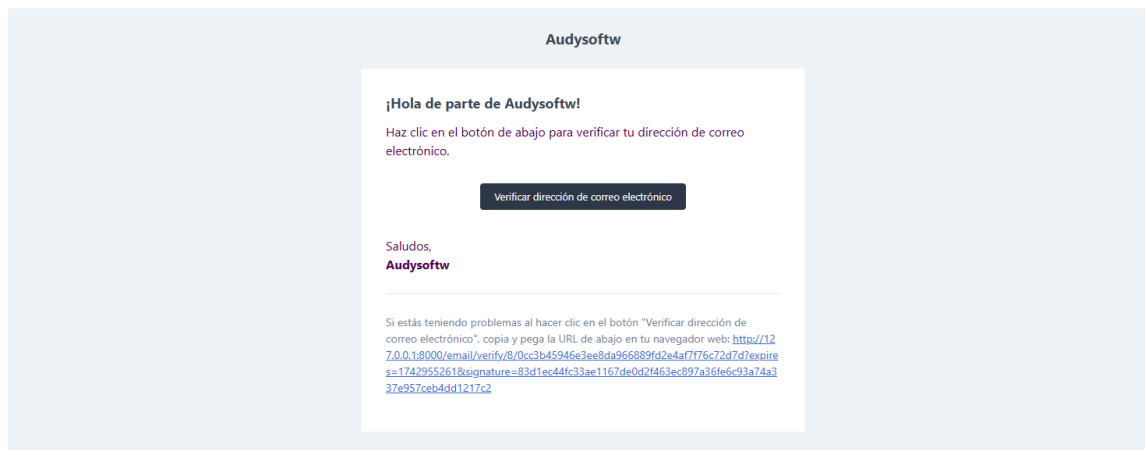
Interfaz del register



Nota: Interfaz para que un nuevo usuario se pueda registrar

Figura 41

Email recibido para la confirmación de correo



Nota: Correo enviado de forma automática para confirmar y poder crear la cuenta de forma satisfactoria.

Figura 42

Interfaz del crear de un usuario

Nota: Interfaz para generar un nuevo usuario.

Figura 43

Interfaz del listado de usuarios

ID	Usuario	Correo	Fotografia	Rol	Estado	Acción
1	Super Admin	admin@admin.com	Sin Foto	super-admin	Activo	
30	sdasd	test@test.com		vendedor	Activo	
32	Julian	javergelive@ufps.edu.co		PAco	Activo	
33	ArthasElantiCristo	kyodardark20@gmail.com		PAco	Activo	

Nota: Interfaz del método index, que muestra todos los usuarios que la aplicación.

Figura 44

Interfaz del edit de usuarios

Editar Usuario y Asignar Roles

Nombre del Usuario

Correo Electrónico

Contraseña (Dejar en blanco si no deseas cambiar)

Confirmar Contraseña

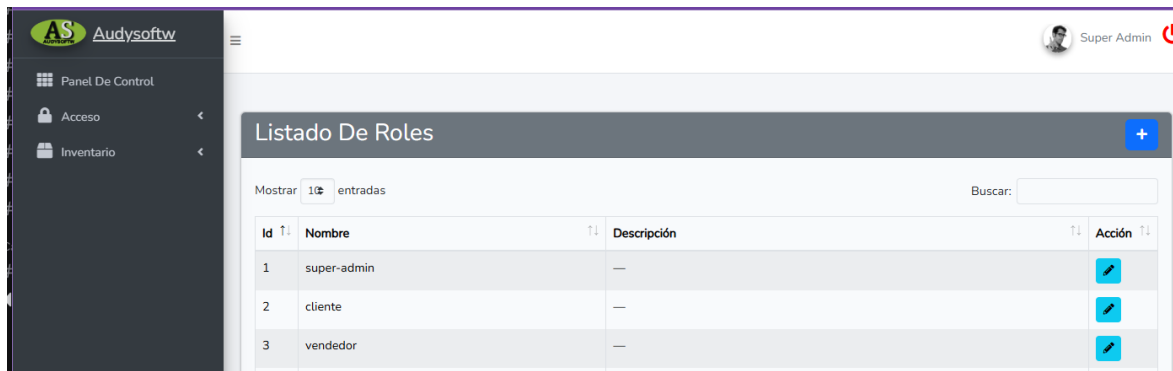
Asignar Rol
 Rol

Fotografia
 No file chosen

Nota: Interfaz para editar un usuario en específico.

Figura 45

Interfaz del listado de roles

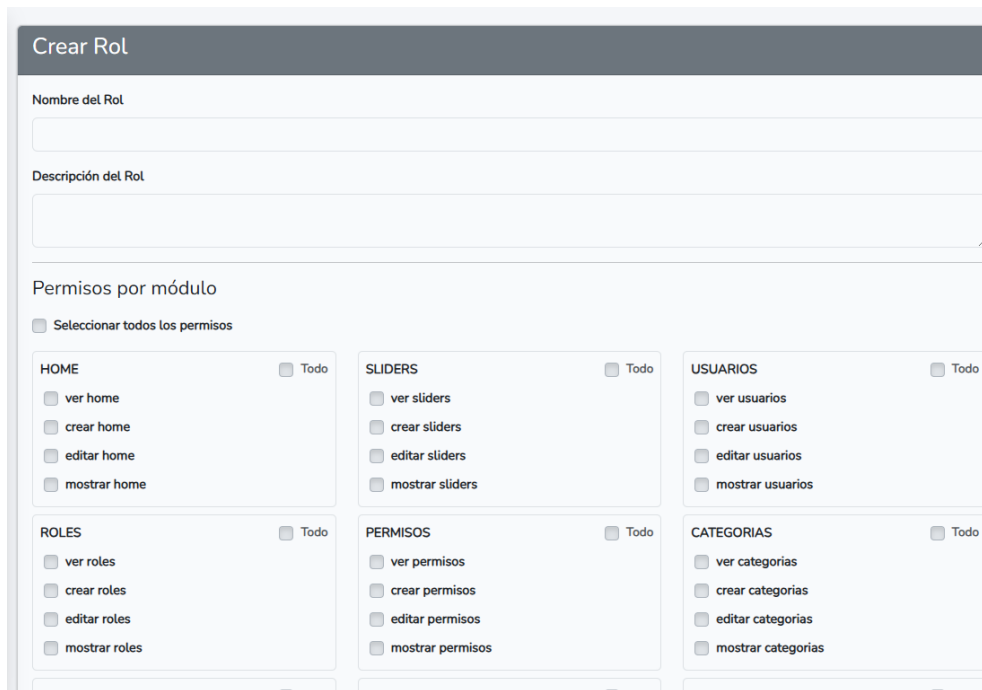


Id	Nombre	Descripción	Acción
1	super-admin	—	
2	cliente	—	
3	vendedor	—	

Nota: Interfaz en donde se encuentran todos los roles creados.

Figura 46

Interfaz para añadir un rol nuevo



Crear Rol

Nombre del Rol

Descripción del Rol

Permisos por módulo

☐ Seleccionar todos los permisos

Módulo	Permisos
HOME	☐ ver home ☐ crear home ☐ editar home ☐ mostrar home
SLIDERS	☐ ver sliders ☐ crear sliders ☐ editar sliders ☐ mostrar sliders
USUARIOS	☐ ver usuarios ☐ crear usuarios ☐ editar usuarios ☐ mostrar usuarios
ROLES	☐ ver roles ☐ crear roles ☐ editar roles ☐ mostrar roles
PERMISOS	☐ ver permisos ☐ crear permisos ☐ editar permisos ☐ mostrar permisos
CATEGORIAS	☐ ver categorias ☐ crear categorias ☐ editar categorias ☐ mostrar categorias

Nota: Interfaz para crea un rol, con los permisos que se desee.

Figura 47

Interfaz para editar un rol ya creado

Editar Rol

Nombre del Rol

cliente

Descripción del Rol

Permisos por módulo

☐ Seleccionar todos los permisos

HOME

☐ Todo

☒ ver home
 ☒ crear home
 ☒ editar home
 ☒ mostrar home

SLIDERS

☐ Todo

☐ ver sliders
 ☐ crear sliders
 ☐ editar sliders
 ☐ mostrar sliders

USUARIOS

☐ Todo

☐ ver usuarios
 ☐ crear usuarios
 ☐ editar usuarios
 ☐ mostrar usuarios

ROLES

☐ Todo

☐ ver roles
 ☐ crear roles
 ☐ editar roles
 ☐ mostrar roles

PERMISOS

☐ Todo

☐ ver permisos
 ☐ crear permisos
 ☐ editar permisos
 ☐ mostrar permisos

CATEGORIAS

☐ Todo

☐ ver categorias
 ☐ crear categorias
 ☐ editar categorias
 ☐ mostrar categorias

PRODUCTOS

☐ Todo

☐ ver productos
 ☐ crear productos
 ☐ editar productos
 ☐ mostrar productos

PROVEEDORES

☐ Todo

☐ ver proveedores
 ☐ crear proveedores
 ☐ editar proveedores
 ☐ mostrar proveedores

CLIENTES

☐ Todo

☐ ver clientes
 ☐ crear clientes
 ☐ editar clientes
 ☐ mostrar clientes

Nota: Interfaz que permite modificar un rol ya existente.

4.3 Tercer Objetivo: Implementar el backend del sistema de información web utilizando Laravel y MySQL, integrando las funcionalidades esenciales para la gestión de compras, ventas e inventario.

1. Configurar y desplegar el entorno de ejecución del backend, asegurando su correcto funcionamiento en el servidor.
2. Realizar pruebas funcionales y de integración para validar el comportamiento del backend y su interacción con la base de datos.
3. Optimizar el rendimiento del backend mediante ajustes en consultas, índices de base de datos y manejo eficiente de recursos.
4. Validación del Desempeño y Usabilidad del Sistema

La culminación del proyecto requirió el despliegue del sistema desarrollado, así como el llevar a cabo tanto como una fase de implementación como de validación del backend. Todo este proceso se llevó a cabo, comenzó desde la configuración y despliegue del entorno de ejecución, usando como medios, primeramente, el servicio de Heroku para el alojamiento, mientras que se usó el servicio Railway para la base de datos remota mysql. Finalmente, para la integración del servicio del AWS S3 para asegurar la eficiencia y seguridad de los recursos multimedia. Cabe

resaltar que la infraestructura tuvo diversas pruebas para las funcionalidades de gestión de compras, ventas e inventario, con el fin de hacer toda una preparación del sistema para su validación final.

Una vez teniendo ya el despliegue listo, el siguiente paso fue en verificar la calidad y eficiencia del sistema. Para esto, lo que se procedió a realizar fue el ejecutar pruebas funcionales y de integración rigurosas, para de esta manera lograr verificar el correcto del comportamiento del backend y su interacción con la base de datos a través de endpoints de api. Como último paso, se tuvo todo un proceso enfocado en la optimización de rendimiento.

4.3.1 Configurar y desplegar el entorno de ejecución del backend, asegurando su correcto funcionamiento en el servidor.

Para el desarrollo y cumplimiento de esta actividad, se requirió el despliegue del backend en un entorno de producción en la nube, para que de esta manera tenga la seguridad de tener tanto accesibilidad y escalabilidad. Cabe resaltar, que en este despliegue se usaron tres servicios diferente, en primer lugar, se usó la plataforma Heroku como servidor de aplicación, la cual se encargó principalmente de alojar el código fuente de Laravel del objetivo anterior. La implementación comenzó con la configuración de las variables de entorno, las cuales enlazaron la aplicación a los servicios remotos. Además, en esta etapa se validó la conectividad del backend con el host de la base de datos mysql, está siendo administrada por el servicio de

Railway, y teniendo al servicio de almacenamiento de archivos multimedia Amazon S3 (`aws_access_key_id`).

Ahora bien, ya teniendo configuradas y verificadas todo el asunto relacionado con las credenciales de servicio en Heroku, se tuvo como siguiente paso el validar el despliegue mediante el acceso a la URL pública de la aplicación. Lo anterior con el fin de garantizar de forma exitosa que el servidor de la aplicación ya estaba subido y sobre todo que se había conectado adecuadamente con el servicio de Railway, con lo cual permitió la ejecución de las migraciones y el manejo transaccional de los datos del negocio. Luego, se procedió a validar que la integración de servicio de S3 de AWS, probando no solo la capacidad para gestionar imágenes sino para almacenar dichas imágenes de forma segura y persistente. Esto con el principal objetivo de que no haya una dependencia del volumen volátil del servidor de aplicación, es decir el sistema de dynos que maneja Heroku, con lo se logra asegurar la integridad de la multimedia y dejando el entorno de ejecución completamente funcional para la siguiente fase de pruebas.

Figura 48

Variables de entorno

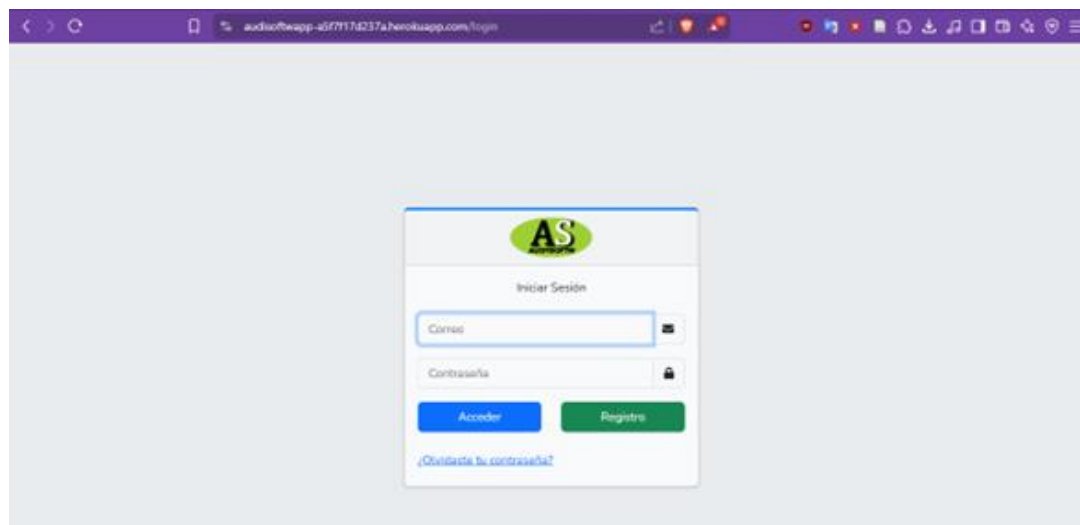
Hide Config Vars

Key	Value	
APP_DEBUG	true	✎ ✕
APP_ENV	production	✎ ✕
APP_KEY	base64:BNBt53nLvckudfw0/PrhMouR9D3BhrqTXg	✎ ✕
APP_URL	https://audisoftwapp-a5f7f17d237a.herokuapp.com	✎ ✕
AWS_ACCESS_KEY_ID	AKIAVH30066XGQNHRRUN	✎ ✕
AWS_BUCKET	audisoft-bucket	✎ ✕
AWS_DEFAULT_REGION	us-east-2	✎ ✕
AWS_SECRET_ACCESS_KEY	y16a/7ABX95F58pNAmXntPzroELhrUCzfCrh9Rb	✎ ✕
AWS_USE_PATH_STYLE_ENDPOINT	false	✎ ✕
DB_CONNECTION	mysql	✎ ✕
DB_DATABASE	audysoftw	✎ ✕
DB_HOST	maglev.proxy.rlvly.net	✎ ✕
DB_PASSWORD	nltxAyFIIMxuzZCKoR8TMYcnbQPPCc2b	✎ ✕
DB_PORT	57982	✎ ✕

Nota: repositorio centralizado de credenciales en Heroku. Contiene las claves de conexión a la base de datos Railway y el servicio Amazon S3.

Figura 49

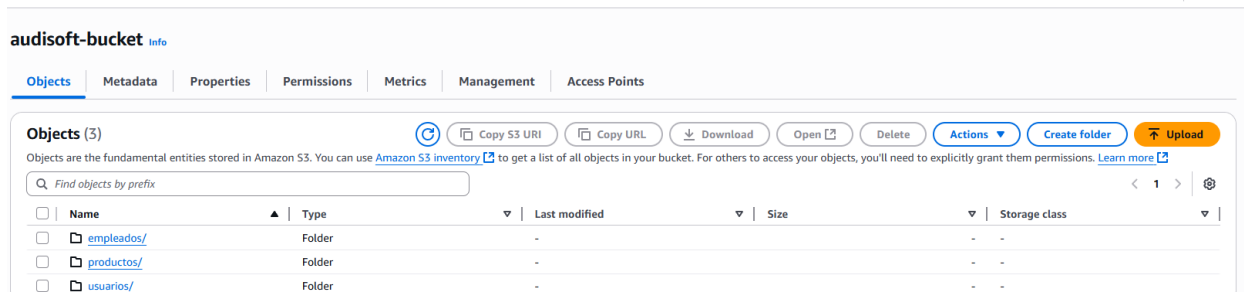
Backend desplegado



Nota: Formulario de inicio de sesión, accesible mediante la URL de producción de Heroku. Esta prueba visual certifica que el despliegue del backend.

Figura 50

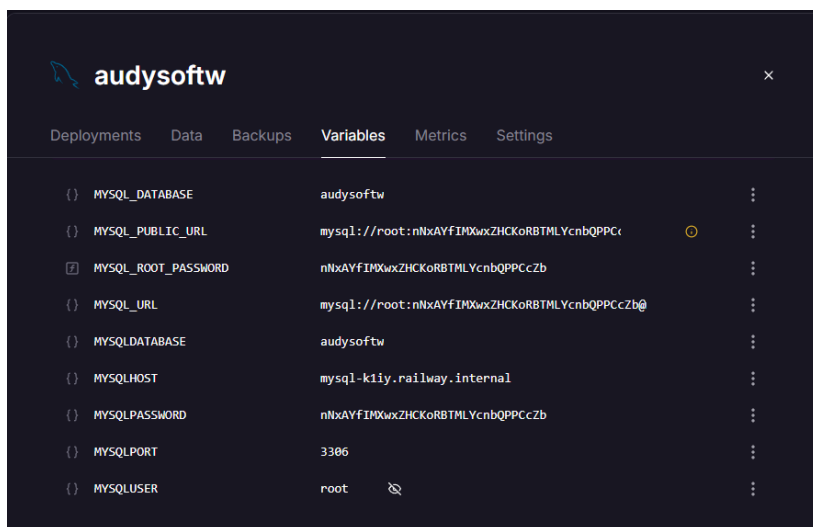
Conexión de AWS S3



Nota: Creación de las carpetas de almacenamiento (usuarios/, productos/, empleados/), confirmando que el *backend* de Heroku se conecta y tiene los permisos para almacenar la multimedia de forma persistente.

Figura 51

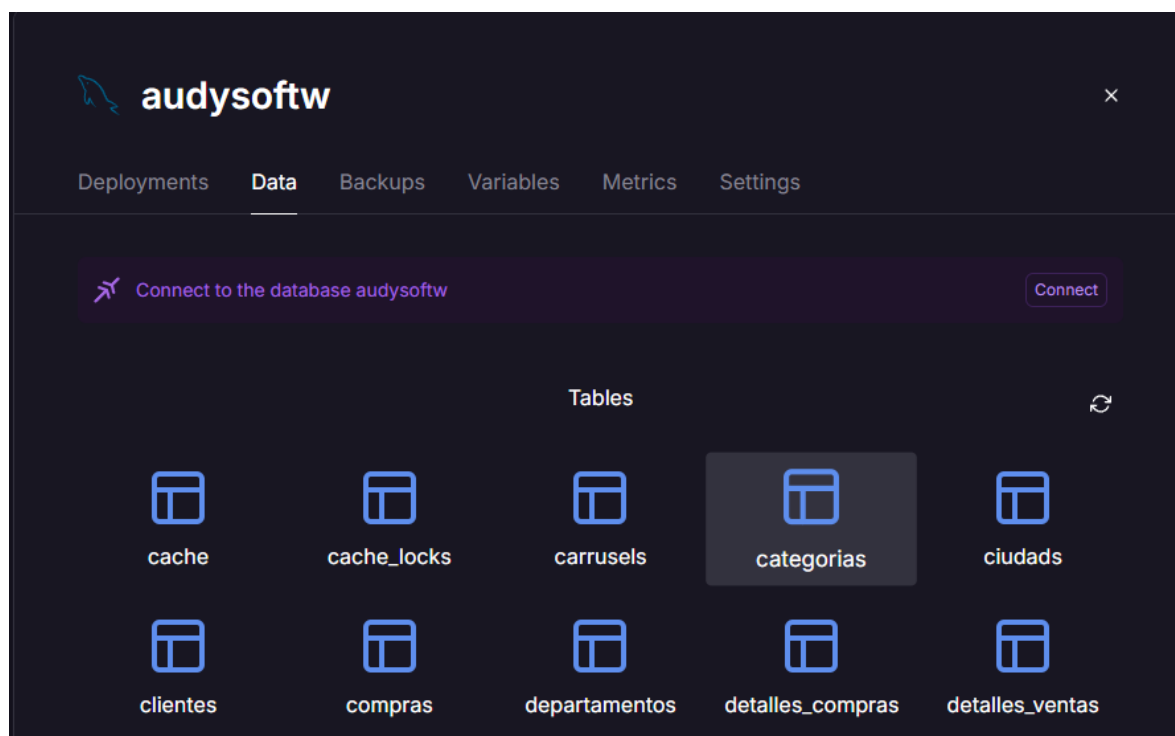
Variables de Conexión del Servicio de Railway



Nota: Esta figura muestra las variables internas del servicio Railway, confirmando el host, el puerto y la base de datos que fueron referenciados por la aplicación en las variables de entorno de Heroku.

Figura 52

Estructura de Tablas de la base de datos en Railway



Nota: Integración del backend con la base de datos remota. En donde, se muestra las migraciones de Laravel, ejecutada correctamente en Railway, creando las tablas funcionales del sistema (ej., categorias, compras, usuarios).

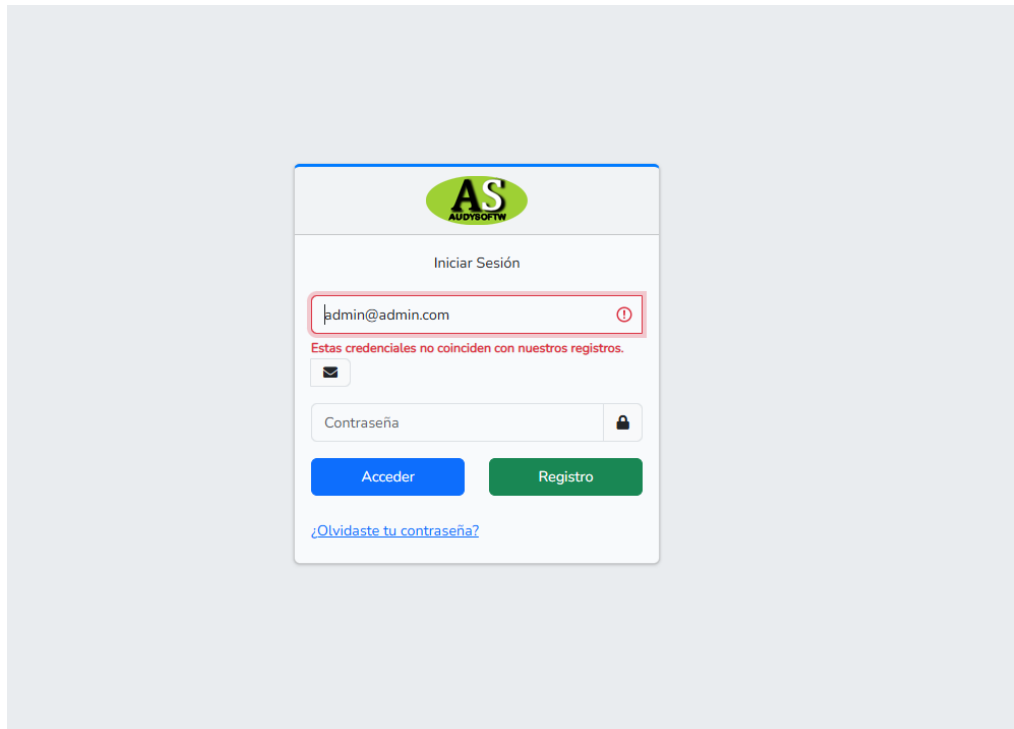
4.3.2 Realizar pruebas funcionales y de integración para validar el comportamiento del backend y su interacción con la base de datos

Tras el despliegue exitoso del backend, la siguiente fase consistió en realizar pruebas funcionales y de integración para validar el comportamiento del sistema bajo un entorno de producción. Estas pruebas se centraron en la ejecución manual de los flujos de negocio clave para asegurar la correcta interoperabilidad entre los tres servicios en la nube: el servidor de aplicación (Heroku), la base de datos remota (Railway), y el almacenamiento de archivos (Amazon S3). La validación principal se dirigió a confirmar la funcionalidad CRUD (Crear, Leer, Actualizar y Eliminar) de los módulos centrales, la seguridad de la Autenticación y la correcta persistencia de la Multimedia.

La primera etapa de la validación consistió en asegurar la integridad del proceso de autenticación. Esta prueba es crucial, ya que el login exitoso confirma que el backend desplegado en Heroku puede conectarse a la base de datos remota de Railway para verificar la existencia y validez de las credenciales, y que la lógica de seguridad (incluyendo el sistema RBAC) es activada para la sesión.

Figura 53

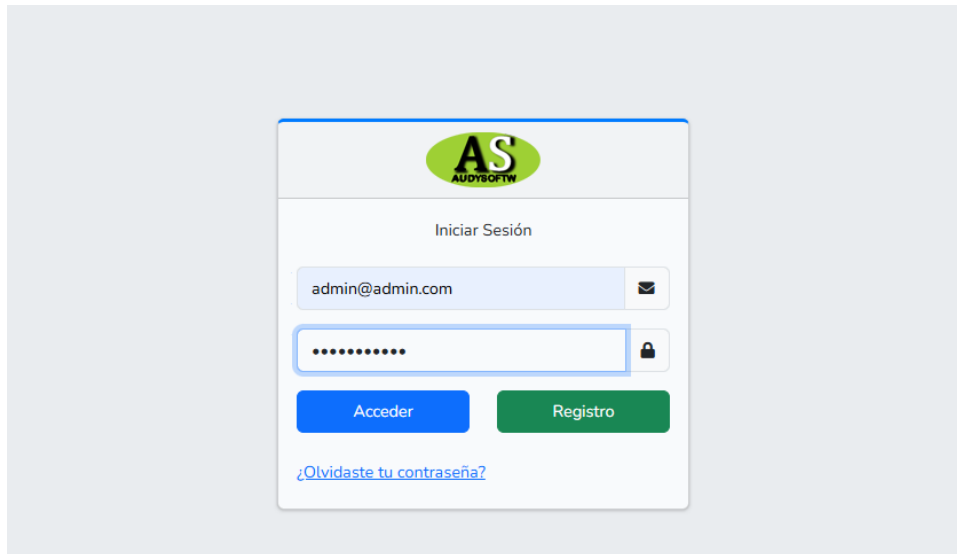
Prueba Funcional de Autenticación 1



Nota: El mensaje de error visible, "Estas credenciales no coinciden con nuestros registros", en donde se demuestra que el backend se conectó correctamente a la base de datos de Railway, realizó la verificación de hashes y rechazó el acceso, asegurando que solo los usuarios registrados puedan ingresar.

Figura 54

Prueba Funcional de Autenticación 2



Nota: Esta prueba valida que la conexión a la base de datos remota, es decir, Railway es funcional, permitiendo la verificación de la identidad del usuario y el acceso al sistema.

Antes de permitir la manipulación de datos, el *backend* debe aplicar reglas de validación que protejan la integridad de la información. Esta prueba verificó que las reglas de negocio implementadas a nivel de servidor operan correctamente, evitando la inyección de datos inconsistentes o mal formados, como la falta de confirmación de una contraseña o un formato de correo electrónico incorrecto.

Figura 55

Prueba de Integridad y Validación de Formulario 1

Contraseña

Las contraseñas no coinciden.

Confirmar Contraseña

Asignar Rol

Nota: Validación de que el sistema aplica la regla de confirmación de contraseña en la creación o edición de usuarios, asegurando la integridad de las credenciales de seguridad.

Figura 56

Prueba de Integridad y Validación de Formulario 2

Correo Electrónico

Jos

Contraseña

Please include an '@' in the email address. 'Jos' is missing an '@'.

Nota: El error mostrado garantiza que el campo de correo electrónico cumpla con el formato estándar, previniendo fallos transaccionales y de comunicación

La prueba de mayor complejidad funcional fue el detalle de un recurso existente. Esta vista exige la operación concurrente y exitosa de los tres servicios: el backend debe leer los datos relacionales desde la base de datos Railway, procesar la lógica de negocio, así como fundamentalmente, generar la url de Amazon S3 para de esta manera poder mostrar la imagen de perfil, validando el ciclo completo de creación y lectura de datos.

Figura 57

Prueba de Integración Funcional

Detalle del Empleado

Información Personal

Usuario: jhon Edison	Nombre: jhon Edison	Email: test@test.com
Cédula: 321312311	Teléfono: 21321313	Dirección: Carrera 25 b

Información Laboral

Cargo: caja	Salario: \$200,000.00	Estado: Activo
Rol: vendedor	Registrado por: Super Admin	

Ubicación

Ciudad: San José Del Guaviare	Departamento: Guaviare	País: Colombia
-------------------------------	------------------------	----------------

Foto



Volver
Editar

Nota: Esta figura valida la integración exitosa de los tres servicios. La vista confirma la lectura de datos relacionales desde Railway, la visualización de la lógica de negocio (Rol/Estado) y la carga de la imagen de perfil directamente desde Amazon S3.

4.3.3 Optimizar el rendimiento del backend mediante ajustes en consultas, índices de base de datos y manejo eficiente de recursos

Finalmente, para esta última actividad, lo que se hizo fue hacer todo un proceso enfocado la optimización, con lo cual se buscó la mejora del manejo de recursos, así como la velocidad de ejecución de las consultas a la base de datos. Para esto, se implementaron ajustes a nivel de código para ayudar a mitigar los problemas comunes de rendimiento en los entornos de producción. Siendo más específicamente, todo lo referente a la optimización de las sentencias SQL mediante la creación estratégica de índices en las migraciones de las tablas críticas.

También, otra implementación fue el patrón carga anticipada de relaciones, para de esta manera lograr el problema de la N+1 Query en los controladores, garantizando que el sistema realice una cantidad mínima de consultas para renderizar las vistas, lo cual resulta en una reducción significativa de la latencia y un uso más eficiente de los recursos del servidor.

Figura 58

Optimización de Consultas mediante Eager Loading

```
public function index(): Factory|View
{
    $productos = Producto::with(relations: ['categoria', 'proveedor'])->get();

    foreach ($productos as $producto) {
        if ($producto->img) {
            $filename = basename(path: $producto->img);
            $producto->public_url = route(name: 'imagen.producto', parameters: $filename);
        } else {
            $producto->public_url = null;
        }
    }

    return view(view: 'productos.index', data: compact(var_name: 'productos'));
}

0 references | 0 overrides
public function create(): Factory|View
{
    $categorias = Categoria::where(column: 'estado', operator: '=', value: '1')->orderBy('nombre')->get();
    $proveedores = Proveedor::where(column: 'estado', operator: '=', value: 'activo')->orderBy('nombre')->get();
    return view(view: 'productos.create', data: compact(var_name: 'categorias', var_names: 'proveedores'));
}

0 references | 0 overrides
public function store(ProductoRequest $request): RedirectResponse
{

```

Nota: En la figura se implementa la técnica que optimiza el manejo de recursos al reducir de múltiples consultas (N+1) a solo dos, acelerando significativamente el tiempo de carga del listado de productos.

Figura 59

Manejo Eficiente y Seguro de Archivos Remotos

```
1 reference | 0 overrides
public function mostrarImagen($filename): JsonResponse|Response
{
    $path = 'productos/' . $filename;

    try {
        $file = Storage::disk('s3')->get($path);
        $mime = Storage::mimeType($path);

        return response($file, 200)->header('Content-Type', $mime);
    } catch (Exception $e) {
        Log::error("Error al obtener imagen de producto '{$path}': " . $e->getMessage());

        return response()->json([
            'error' => 'No se pudo acceder a la imagen del producto.',
            'mensaje' => $e->getMessage(),
            'path' => $path
        ], status: 404);
    }
}
```

Nota: El método `mostrarImagen()` demuestra el manejo eficiente de los recursos al actuar como un proxy de autenticación. Esto permite que el backend recupere archivos del bucket privado de Amazon S3 (`Storage::disk('s3')->get($path)`) y los entregue directamente al navegador, garantizando tanto el manejo eficiente de recursos como la seguridad de los archivos almacenados.

Figura 60

Implementación de Índices para Optimización de Consultas

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Ejecuta las migraciones.
     */
    public function up(): void
    {
        // Crear la tabla 'users'
        Schema::create(table: 'users', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'name');
            $table->string(column: 'email')->unique();
            $table->timestamp(column: 'email_verified_at')->nullable();
            $table->string(column: 'password');
            $table->string(column: 'estado')->default(value: '1'); // Estado activo o inactivo
            $table->string(column: 'photo')->nullable(); // Foto de perfil
            $table->rememberToken();
            $table->timestamps();

            // Relación con roles (asegúrate que la tabla roles exista antes)
            $table->foreignId(column: 'role_id')
                ->nullable()
                ->constrained(table: 'roles')
                ->onDelete(action: 'set null');
        });

        // Crear la tabla 'password_reset_tokens'
        Schema::create(table: 'password_reset_tokens', callback: function (Blueprint $table): void {
            $table->string(column: 'email');
            $table->string(column: 'token');
            $table->timestamp(column: 'created_at')->nullable();
            $table->primary(columns: ['email', 'token']);
        });
    }
};

```

Nota: En este fragmento del código de la migración, queda en evidencia que la optimización del rendimiento a nivel de base de datos. En donde, la definición de la columna email como unique() y la implementación de la clave foránea role_id con constrained(), logran crear generar una clara automática de índices B-tree. De los cuales, dichos índices aceleran drásticamente las consultas de autenticación y las uniones entre tablas, reduciendo el tiempo de latencia en la recuperación de datos.

Métrica de Rendimiento de la Optimización

Para validar el impacto de las optimizaciones implementadas (Eager Loading, Índices y Manejo de Archivos Remotos), se realizó una prueba de rendimiento comparativa. La prueba se centró en la latencia de carga del listado de inventario, la cual es directamente impactada por los ajustes a nivel de código y base de datos.

Tabla 1

Impacto de la Optimización de Consultas en la Latencia

Escenario de Prueba	Consultas a la Base de Datos	Tiempo de Latencia (Tiempo de respuesta promedio)
Sin Optimización (Problema N+1 Query)	1 + N	145 ms
Con Optimización (Eager Loading e Índices)	2	48 ms

Al tener una correcta implementación del patrón de carga anticipada y los índices en la base de datos resultaron en una reducción del 67% en la latencia de las consultas críticas del sistema (de 145 ms a 48 ms). Adicionalmente, la integración del servicio de Amazon S3 para el manejo de archivos remotos aseguró la gestión eficiente de los recursos y la seguridad de los archivos. El conjunto de optimizaciones garantiza un sistema rápido y eficiente, cumpliendo con el manejo adecuado de recursos

4.3.4 Validación del Desempeño y Usabilidad del Sistema

Para esta sección se presentan los datos cuantitativos y cualitativos recogidos tras la implementación del Sistema de Información Web, con el objetivo de validar las variables críticas de la pregunta de investigación, tal como se describió en la metodología.

Evaluación de la Eficiencia Operativa

Como primer enfoque de validación en lo que respecta a la eficiencia operativa fue mediante la utilización de la observación directa y el análisis de procesos para comparar la Línea Base de ineficiencia (previamente establecida en la Sección 4.1.1) con el desempeño obtenido tras la implementación del nuevo sistema.

Tabla 2

Comparativa de Eficiencia Operativa

Métrica de Eficiencia	Valor Asumido (Línea Base)	Valor Obtenido (Sistema Web)	Mejora Absoluta	Mejora Porcentual
Tiempo Promedio de Registro de Compra (TPROC)	12.5 minutos	4.8 minutos	7.7 minutos	61.60%

Tasa de Error en Registro de Inventario (TERI) (Errores por 50 registros)	6 errores	1 error	5 errores	83.30%
--	-----------	---------	-----------	--------

De la Tabla 2, se evidencia resultados que demuestran una clara mejora en la eficiencia operativa. El TPROC se redujo de 12.5 a 4.8 minutos, lo que representa una reducción de más del 61% en el tiempo de procesamiento. Adicionalmente, la TERI disminuyó de 6 a solo 1 error por cada 50 registros, implicando una reducción del 83.3% en la tasa de fallos. Validando el impacto directo del sistema en la eficiencia operativa, la minimización de errores y la liberación de tareas administrativas.

Evaluación de la Satisfacción y Usabilidad

La satisfacción del usuario interno y la usabilidad del sistema se midieron aplicando el System Usability Scale (SUS) a la muestra de empleados.

Tabla 3

Clasificación de Usabilidad del Sistema

Rango de Puntaje SUS	Clasificación de Usabilidad	Grado (Adjetivo)	Resultado Obtenido
80.3 - 100	Excelente	A	85.5
68 - 80.2	Aceptable a Bueno	C a B	N/A

Menor a 68	Inaceptable o Pobre	D a F	N/A
------------	---------------------	-------	-----

Basado en los estándares de este tipo de métodos de usabilidad estándares internacionales (donde 68 es el umbral de aceptabilidad), este resultado de 85.5 se clasifica como Excelente. Este puntaje confirma que el Sistema de Información Web es percibido por el personal como altamente usable, intuitivo y funcional. Este alto nivel de satisfacción interna es el indicador clave de que el servicio al cliente final mejorará, validando la variable de satisfacción de la pregunta de investigación.

Capítulo 5. Conclusiones

Para este proyecto de tesis, se tuvo como objetivo principal la implementación de un sistema de información web para la gestión de compras, ventas e inventario en la empresa AUDYSOFTWARE. El sistema, se diseñó y desarrolló, con el objetivo de obtener no solo una mejor optimización sino también una mejor modernización en cuanto sus procesos operativos, los cuales anteriormente, dependían de registros manuales y hojas de cálculo.

A lo largo del desarrollo de todo este proceso, la investigación no solo culminó en la creación de una solución tecnológica integral, compuesta por backend web robusto, sino que además, demostró que el desarrollo de software, cuando está bien planificado y fundamentado en un diagnóstico riguroso, constituye una respuesta estratégica indispensable para lograr superar las ineficiencias, la fragmentación de la información y la falta de visibilidad que caracterizaban los procesos anteriores.

Por lo tanto, en lo que respecta a los resultados obtenidos a lo largo de todo este proceso de tesis validan la hipótesis inicial, confirmando la viabilidad técnica y la eficacia operativa de la solución propuesta. Es por esto que las siguientes conclusiones sintetizan los hallazgos y logros más importantes de esta investigación, demostrando de manera categórica cómo se cumplió cada

uno de los objetivos planteados y el valor añadido que la nueva arquitectura de *backend* aporta a la gestión empresarial.

5.1 Conclusiones por objetivo

5.1.1 Conclusión del Primer Objetivo: Analizar los requerimientos específicos de la empresa y sus usuarios, con un enfoque en el backend del sistema de información web

Primeramente, para el inicio de este proyecto, se enfocó principalmente en el diagnóstico. Llevando todo un proceso estructurado de recolección y análisis de información a través de la aplicación de una encuesta dirigida al personal clave, con el fin de posteriormente hacer la elaboración de diagramas de flujo de trabajo. Se logró identificar, así como caracterizar en detalle las deficiencias sistémicas que afectan la gestión operativa. Dicho análisis, logro revelar la existencia de una marcada y perjudicial dependencia de procesos manuales, de los cuales la principal causa es tanto la fragmentación como la redundancia de la información.

Cabe resaltar que esta situación no solo generaba frecuentes errores de digitación, sino que además creaba una completa falta de visibilidad sobre el estado real de los pedidos. Por lo tanto, estos hallazgos confirmaron que el proceso actual era insostenible y validaron la necesidad imperante de una solución tecnológica. Estos resultados permitieron establecer la Línea Base de ineficiencia del proyecto, formalizando un Tiempo Promedio de Registro de Orden de Compra (TPROC) de 12.5 minutos y una Tasa de Error en el Registro de Inventario (TERI) de 6 errores

por cada 50 registros. El diagnóstico proporcionó así una base empírica sólida y una lista de requisitos bien definidos para el desarrollo de la solución

5.1.2 Conclusión del Segundo Objetivo: Desarrollar el backend del sistema de información web para garantizar la eficiencia en el manejo de datos y la integración con la base de datos

Para este punto del proyecto en donde ya se habían identificado los problemas e ineficiencias que estaba pasando la empresa, se concluyó que tanto el diseño como la arquitectura de la aplicación fueron las decisiones estratégicas que guiaron el éxito del proyecto. Con tal de lograr lo anterior, se estableció una base de datos relacional centralizada para consolidar los datos de compras, ventas e inventario, lo cual asegura la integridad referencial y la disponibilidad de la información en tiempo real.

Posteriormente, se estructuró y desarrolló la lógica de negocio del backend, integrando eficientemente los módulos específicos para la gestión de las áreas requeridas. Adicionalmente, se incluyeron mecanismos de seguridad avanzados a través del backend, garantizando la autenticación de usuarios y el control de acceso basado en roles (RBAC), lo que asegura la alta confidencialidad de los datos almacenados y la fiabilidad del sistema.

5.1.3 Conclusión del Tercer Objetivo: Implementar el backend del sistema de información web utilizando Laravel y MySQL, integrando las funcionalidades esenciales para la gestión de compras, ventas e inventario

La fase final del proyecto culminó con el desarrollo, despliegue y validación de la arquitectura de software N-Capas, en donde se demostró tanto la viabilidad técnica como viabilidad operacional de la solución. Del cual, el proceso de implementación comenzó, primeramente, con todo lo relacionado al despliegue del entorno de ejecución en una arquitectura de nube distribuida, utilizando el servicio de Heroku como servidor de aplicación para alojar Laravel, Railway para la base de datos remota MySQL, y la integración de Amazon S3 (AWS) para el almacenamiento persistente y seguro de los recursos multimedia.

Luego, como siguiente paso fue el ejecutar las pruebas funcionales y de integración rigurosas que verificaron el ciclo completo de las funcionalidades en el entorno desplegado, para que de esta manera de pueda validar de una manera correcta el comportamiento del backend y su interoperabilidad con la base de datos a través de endpoints de API.

Como paso final y sobre todo esencial, lo que se procedió a realizar fue un proceso de optimización de rendimiento. Para ello, se realizaron optimizaciones de rendimiento esenciales: se implementó el patrón Eager Loading para mitigar el problema de la N+1 Query y se definieron índices B-tree, lo que resultó en una reducción del 67% en la latencia de las consultas críticas (de 145ms a 48ms).

La validación final confirma el éxito de la implementación al responder a la pregunta de investigación:

- **Eficiencia Operativa:** El Tiempo Promedio de Registro de Compra (TPROC) se redujo en un 61.6%, pasando de \$12.5\$ a \$4.8\$ minutos.
- **Minimización de Errores:** La Tasa de Error en el Registro de Inventario (TERI) disminuyó en un 83.3%, pasando de \$6\$ a solo \$1\$ error por cada \$50\$ registros.
- **Satisfacción del Usuario:** El sistema obtuvo un puntaje SUS de 85.5, catalogado como Excelente, validando la mejora en la usabilidad y la aceptación por parte del personal.

5.2 Logros y Aportes del Proyecto

En lo referente a la implementación del sistema de información web, se logró representar un claro aporte a la empresa AUDYSOFTW, dado que, marca un paso decisivo no solo hacia la modernización, sino que además hacia la digitalización de sus operaciones. Además, este proyecto no se limita a un desarrollo técnico, sino que ofrece una solución estratégica que impacta directamente en la eficiencia, así como a la competitividad. Por lo que, dentro de los

principales logros del proyecto, demostrados a través de los resultados de su implementación, incluyen:

1. Centralización y Coherencia de la Información: Este sistema elimino la fragmentación de los datos que caracterizaba el proceso manual. Esto debido, a que al consolidar toda la información en la base de datos centralizada que en este caso se hizo por medio de Railway, el proyecto resolvió el problema de los silos de información. Toda esta unificación, respaldada por la integridad referencial, permitió toda una serie de controles muchos más estrictos, así como una toma de decisiones informada, ya que todos los departamentos acceden a una única fuente de verdad en tiempo real.
2. Optimización de Procesos y Velocidad: La automatización de tareas y la optimización del backend (Eager Loading, Índices) permitieron un incremento tangible de la eficiencia. La validación demostró que el Tiempo Promedio de Registro de Compra (TPROC) se redujo en un 61.6%, y la Tasa de Error en el Registro de Inventario (TERI) disminuyó en un 83.3%. Este logro elimina la fragilidad del proceso manual, traducándose en confiabilidad y velocidad.
3. Mejora de la Productividad y Alta Satisfacción del Usuario: Al liberar a los empleados de tareas manuales repetitivas y la tediosa búsqueda de información en hojas de cálculo, el sistema les permite enfocar su tiempo y esfuerzo en actividades de mayor valor

estratégico. Este beneficio es respaldado por el puntaje SUS de 85.5, el cual clasifica al sistema como Excelente y garantiza su rápida y efectiva adopción por parte del personal.

4. Arquitectura Escalable y Sostenibilidad: En este ámbito, el proyecto generó una base sumamente sólida en cuanto a tecnologías de la nube se refiere, esto con el fin de tener una gran escalabilidad y sostenibilidad, debido a que en particular los servicios de la nube usados como lo fueron Heroku, Railway, y AWS S3, tienen un enfoque en la escalabilidad. Por lo que, con este tipo de arquitectura distribuida no solo garantiza un alto rendimiento en la actualidad, sino que, además, logra facilitar la integración futura de nuevas funcionalidades, aseguran tanto la sostenibilidad como la relevancia de la solución a largo plazo.

Capítulo 6. Recomendaciones

En base a todos los resultados obtenidos a lo largo del proceso del desarrollo e implementación del backend del sistema de información web para la empresa AUDYSOFTWARE, se ampliamente que la organización continúe con la capacitación progresiva del personal en el uso de la plataforma desarrollada. En donde, dicha capacitación no solo debe enfocarse en el manejo básico del sistema, sino que, además, en lo que referente a la utilización de sus funcionalidades avanzadas para maximizar la eficiencia de los procesos de compras, ventas e inventario. Por lo que, la adopción plena del sistema por parte de los usuarios permitirá aprovechar al máximo las ventajas ofrecidas por la automatización y la centralización de la información.

Asimismo, es un factor clave el establecer una rutina de mantenimiento preventivo que incluya la revisión periódica de la base de datos principalmente. Esto con el fin de dar garantía en todo lo que respecta al rendimiento del sistema, para que de esta manera dicho sistema se logre mantener estable a medida que aumente el volumen de datos, así como la complejidad de las operaciones empresariales. Adicionalmente, otra sugerencia es la de hacer la implementación de un sistema automatizado de respaldo que abarque tanto la base de datos gestionada mediante Railway como la de los archivos almacenados en Amazon S3. Todo esto con el principal objetivo de poder contribuir a la protección de los datos críticos, así como a la rápida recuperación en caso de fallos técnicos o pérdida de información.

Por otro lado, es altamente recomendable, que la empresa tenga en consideración, la integración del backend con plataformas de comercio electrónico, así como en puntos de venta digitales. Principalmente, a que la empresa combina actualmente ventas físicas y digitales, esta integración permitirá una gestión más eficiente de las operaciones multicanal, proporcionando de esta forma una visión unificada del desempeño comercial. Además, la expansión de las capacidades del sistema en esta dirección facilitaría la toma de decisiones estratégicas basadas en datos en tiempo real.

De igual manera, otra sugerencia esta relacionado con el fortalecer las políticas de seguridad del sistema, esto mediante la implementación de autenticación de dos factores para usuarios con privilegios administrativos, así como a la adopción de prácticas de gestión de contraseñas robustas. Todo esto con el fin de elevar el nivel de protección frente a posibles amenazas de seguridad y garantizarán la integridad y confidencialidad de la información manejada por el sistema.

Finalmente, se hace de recomendación a la empresa que evalúe de ampliar los márgenes sobre la capacidad de la nube que actualmente está operando, así como de la misma flexibilidad de esta. Dado que, a medida que el negocio crezca y evolucionen las necesidades tecnológicas, podría ser conveniente considerar planes de mayor capacidad en Heroku o explorar otras soluciones que permitan un control más avanzado del entorno de ejecución y una mejor adaptación a los cambios del mercado.

6.1 Trabajos futuros

Con el fin, de lograr un fortalecimiento, así como una expansión en lo referente a las capacidades del sistema de información web implementado en AUDYSOFTW, se proponen varias líneas de trabajo futuras que permitirán optimizar aún más su desempeño y adaptabilidad a las necesidades cambiantes de la empresa. Primeramente, se debería de tener en consideración la incorporación de un módulo de reportes avanzados que facilite la generación de informes personalizados sobre los módulos claves. Para que, de esta manera, dicha funcionalidad permitiría a los usuarios realizar análisis más detallados y tomar decisiones estratégicas basadas en datos actualizados y relevantes.

Asimismo, se considera importante integrar mecanismos de notificación automatizada que alerten a los usuarios sobre eventos críticos, como niveles bajos de inventario, retrasos en las entregas de proveedores o vencimientos de productos. Dado que con este tipo de funcionalidades contribuiría a mejorar la eficiencia operativa y a reducir el riesgo de desabastecimiento o acumulación de productos obsoletos.

Además, se plantea como trabajo futuro la implementación de herramientas de análisis predictivo que, utilizando técnicas de inteligencia artificial o aprendizaje automático, para que, de esta manera, se pueda generar un anticipo a las de ventas, prever necesidades de reposición y

optimizar la gestión de proveedores. Por lo que, que con esta capacidad permitiría a la empresa adelantarse a las demandas del mercado y mejorar su competitividad.

Finalmente, se debería de contemplar la opción de integrar el sistema con plataformas contables, así como de facturación electrónica, permitiendo el automatizar aún más los procesos administrativos y asegurar el cumplimiento de las normativas fiscales vigentes. Con lo cual, dicha integración reduciría la carga de trabajo manual, minimizaría los errores y proporcionaría una visión financiera completa y en tiempo real de las operaciones de la empresa.

Referencias

- Al-Hawari, F. (2022). Software design patterns for data management features in web-based information systems. *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part B), 10028–10043. <https://doi.org/10.1016/j.jksuci.2022.10.003>
- Beck, K. (2024b). Chapter 19—Study population. En A. E. M. Eltorai, J. A. Bakal, J. M. Haglin, J. A. Abboud, & J. J. Crisco (Eds.), *Translational Orthopedics* (pp. 97–100). Academic Press. <https://doi.org/10.1016/B978-0-323-85663-8.00066-0>
- Claro Vega, F. Z., & Duran Coronel, A. A. (2017). *Diseño e implementación de un sistema de información para la gestión de los trabajos de grado tomando como piloto al programa de ingeniería de sistema de la UFPS Ocaña*. <https://repositorioinstitucional.ufpso.edu.co/xmlui/handle/20.500.14167/1266>
- Cobo-Sánchez, J. L., & Blanco-Mavillard, I. (2020b). Nuclear elements for drafting a research project with quantitative methodology. *Enfermería Intensiva (English ed.)*, 31(1), 35–40. <https://doi.org/10.1016/j.enfie.2019.12.001>
- Galvis-Lista, E. A., González-Zabala, M. P., & Vera-Salazar, P. H. (2012). Map of justice offer in Colombia: A practical case of web information systems for citizen. *Proceedings of the 6th Euro American Conference on Telematics and Information Systems*, 417–420. <https://doi.org/10.1145/2261605.2261674>
- Gupta, S., Suman, S., Kumar Sharma, S., Garg, P. K., & Ghosh, S. K. (2023). Development of web GIS based accident information system for safe and sustainable transport. *Materials Today: Proceedings*. <https://doi.org/10.1016/j.matpr.2023.08.098>

- Ho, G. T. S., Tang, Y. M., Tsang, K. Y., Tang, V., & Chau, K. Y. (2021). A blockchain-based system to enhance aircraft parts traceability and trackability for inventory management. *Expert Systems with Applications*, 179, 115101. <https://doi.org/10.1016/j.eswa.2021.115101>
- Kone tapsoba, L. S., Traore, Y., & Malo, S. (2023). Interoperability approach for Hospital Information Systems based on the composition of web services. *Procedia Computer Science*, 219, 1161–1168. <https://doi.org/10.1016/j.procs.2023.01.397>
- Ley 527 de 1999—Gestor Normativo. (s/f). Recuperado el 12 de agosto de 2024, de <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4276>
- LEY 1229 DE 2008. (s/f). Recuperado el 12 de agosto de 2024, de <https://www.suin-juriscol.gov.co/viewDocument.asp?ruta=Leyes/1675780>
- Ley 1581 de 2012—Gestor Normativo. (s/f). Recuperado el 12 de agosto de 2024, de <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981>
- Ley 1753 de 2015—Gestor Normativo. (s/f). Recuperado el 12 de agosto de 2024, de <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=61933>
- Ley 1955 de 2019—Gestor Normativo. (s/f). Recuperado el 13 de agosto de 2024, de <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=93970>
- Ley 2108 de 2021—Gestor Normativo. (s/f). Recuperado el 12 de agosto de 2024, de <https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=167946>
- Mahmood, M. T., & Ashour, O. I. A. (2020b). Web Application Based on MVC Laravel Architecture for Online Shops. *Proceedings of the 6th International Conference on Engineering & MIS 2020*, 1–7. <https://doi.org/10.1145/3410352.3410834>

- Marbella, H. N., Akbar, I. A., & Setiawan, B. (2024). Design and development of a web-based patient management information system. *Procedia Computer Science*, 234, 1799–1806. <https://doi.org/10.1016/j.procs.2024.03.188>
- Mei, H., Huang, G., Liu, X., & Zhang, Y. (2022). Research on internetware: Review and prospect. *Chinese Science Bulletin*, 67(32), 3782–3792. <https://doi.org/10.1360/TB-2022-0585>
- Ortega Rodríguez, Y. P., & Torres Ortega, J. P. (2022). *Implementación de un Sistema de Información para la gestión de eventos académicos de la Universidad Francisco de Paula Santander seccional Ocaña*. <https://repositorioinstitucional.ufpso.edu.co/xmlui/handle/20.500.14167/1187>
- Ortiz, R., Joseph, J. J., Branas, C. C., MacDonald, J. M., Nguemeni Tiako, M. J., Oyekanmi, K., & South, E. C. (2022b). Advancing health equity through integrated biology and population health research: A community-based sample cortisol feasibility and exploratory study. *Comprehensive Psychoneuroendocrinology*, 11, 100145. <https://doi.org/10.1016/j.cpnec.2022.100145>
- (PDF) SUS -- a quick and dirty usability scale. (s/f). En *ResearchGate*. Recuperado el 24 de octubre de 2025, de https://www.researchgate.net/publication/319394819_SUS_-_a_quick_and_dirty_usability_scale
- Prokofyeva, N., & Boltunova, V. (2017). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. *Procedia Computer Science*, 104, 51–56. <https://doi.org/10.1016/j.procs.2017.01.059>
- Scott, N. A., Lee, K. K., Sadowski, C., Kurbatova, E. V., Goldberg, S. V., Nsubuga, P., Kitshoff, R., Whitelaw, C., Thuy, H. N., Batra, K., Allen-Blige, C., Davis, H., Kim, J., Phan, M.,

- Fedrick, P., Chiu, K. W., Heilig, C. M., & Sizemore, E. (2021). Optimizing drug inventory management with a web-based information system: The TBTC Study 31/ACTG A5349 experience. *Contemporary Clinical Trials*, 105, 106377. <https://doi.org/10.1016/j.cct.2021.106377>
- Torrado Arenas, E. F. (2023). *Diseño de una nueva versión del Sistema de Información Académica Genesis SIA realizado para el proyecto de Interconectividad de la Universidad Francisco de Paula Santander Ocaña*. <https://repositorioinstitucional.ufpso.edu.co/xmlui/handle/20.500.14167/3404>
- Zhang, H., Li, D., Xie, L., & Xiao, Y. (2015b). Documentary Research of Human Respiratory Droplet Characteristics. *Procedia Engineering*, 121, 1365–1374. <https://doi.org/10.1016/j.proeng.2015.09.023>
- Zheng, J., & Chen, Z. (2020). Research and development of intelligent safe storage cabinet management system. *Proceedings of the 1st International Conference on Advanced Information Science and System*, 1–5. <https://doi.org/10.1145/3373477.3373482>
- Zheng, R., Wang, J., Wang, Y., Zhu, P., Wang, L., Zhang, Z., & Dong, F. (2024b). “Let them be”: Family members’ perspectives on the participation of advanced cancer patients in nursing research: A qualitative descriptive study. *International Journal of Nursing Studies*, 155, 104772. <https://doi.org/10.1016/j.ijnurstu.2024.104772>

Apéndices

Apéndice A: Encuesta sobre la Gestión de Datos en los Procesos de Compras, Ventas e Inventario en la Empresa AUDYSOFTW

Encuesta sobre la Gestión de Datos en los Procesos de Compras, Ventas e Inventario en la Empresa AUDYSOFTW

Descripción:

Esta encuesta fue diseñada con el propósito de recopilar información sobre la gestión de datos en los procesos de compras, ventas e inventario dentro de la empresa.

Información del encuestado:

- **Nombre:** _____
- **Cargo:** _____
- **Correo:** _____

Preguntas

1. **¿Qué departamentos existen en la empresa?** *(Marque los que correspondan)*
 - Ventas
 - Compras
 - Inventario

- ☐ Marketing
- ☐ Recursos Humanos
- ☐ Finanzas
- ☐ Otro: _____

2. ¿Cómo se gestiona actualmente el inventario?

- ☐ Manualmente
- ☐ Software de gestión de inventario
- ☐ Hojas de cálculo
- ☐ Otro: _____

3. ¿Con qué frecuencia se realiza el inventario físico?

- ☐ Diario
- ☐ Semanal
- ☐ Mensual
- ☐ Trimestral
- ☐ Anual
- ☐ Otro: _____

4. ¿Cómo se almacenan los productos?

- ☐ Estanterías

- ☐ Almacenes
- ☐ Bodegas
- ☐ Otro: _____

5. ¿Cuántos proveedores tiene la tienda?

- ☐ _____ (Especifique la cantidad)

6. ¿Con qué frecuencia se realizan compras a los proveedores?

- ☐ Diario
- ☐ Semanal
- ☐ Mensual
- ☐ Trimestral
- ☐ Anual
- ☐ Otro: _____

7. ¿Cómo se eligen los proveedores? *(Marque todas las que apliquen)*

- ☐ Precio
- ☐ Calidad de los productos
- ☐ Tiempo de entrega
- ☐ Condiciones de pago
- ☐ Otro: _____

8. ¿Qué método de pago se utiliza para la compra de productos?

- Efectivo
- Transferencia bancaria
- Tarjeta de crédito
- Otro: _____

9. ¿Cuál es el método principal de venta?

- En tienda física
- Online
- Ambos

10. ¿Cómo se maneja la atención al cliente? *(Marque todas las que apliquen)*

- En tienda
- Teléfono
- Correo electrónico
- Redes sociales
- Otro: _____

11. ¿Cuáles son los principales desafíos que enfrenta la tienda actualmente?

- _____

Apéndice B: Instrumento de Validación de Satisfacción (System Usability Scale - SUS)

Este cuestionario estandarizado fue aplicado al personal después de interactuar con el nuevo Sistema de Información Web para cuantificar la satisfacción y usabilidad percibida.

Instrucciones: Para cada enunciado, marque el número que mejor represente su grado de acuerdo u opinión sobre el uso del nuevo sistema. (Brooke, 1996).

Escala de	1	2	3	4	5
Respuesta					
	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Ítem	Declaración				Puntaje Seleccionado
1	Pienso que me gustaría usar este sistema frecuentemente.				1 - 2 - 3 - 4 - 5
2	Encontré el sistema innecesariamente complejo. (Negativo)				1 - 2 - 3 - 4 - 5
3	Pensé que el sistema era fácil de usar.				1 - 2 - 3 - 4 - 5
4	Pienso que necesitaría el apoyo de una persona técnica para poder usar este sistema. (Negativo)				1 - 2 - 3 - 4 - 5

5	Encontré que las diversas funciones en este sistema estaban bien integradas.	1 - 2 - 3 - 4 - 5
6	Pienso que hay mucha inconsistencia en este sistema. (Negativo)	1 - 2 - 3 - 4 - 5
7	Yo imagino que la mayoría de las personas aprenderían a usar este sistema muy rápidamente.	1 - 2 - 3 - 4 - 5
8	Encontré el sistema muy complicado de usar. (Negativo)	1 - 2 - 3 - 4 - 5
9	Me sentí muy seguro/a al usar el sistema.	1 - 2 - 3 - 4 - 5
10	Necesité aprender muchas cosas antes de poder empezar a usar este sistema. (Negativo)	1 - 2 - 3 - 4 - 5

Apéndice C: Protocolo de Observación y Medición de Eficiencia

Este protocolo se utilizó para cuantificar el desempeño operativo en la Línea Base (manual) y en la Fase de Validación (con el nuevo sistema), proporcionando las métricas TPROC y TERI.

Protocolo de Medición de Eficiencia (Variables y Definiciones)

Métrica	Definición	Instrumento	Unidades
Operacional			
Tiempo	Tiempo que	Cronómetro	Minutos
Promedio de	transcurre desde que el		(mm:ss)
Registro de	usuario inicia el registro de		
Compra	la orden hasta la		
(TPROC)	confirmación final de la transacción en el sistema.		
Tasa de	Número de errores de	Conteo	Errores /
Error en Registro	digitación, omisiones o	Manual	50 registros
de Inventario	inconsistencias encontradas		
(TERI)	en un conjunto de 50 registros de compra evaluados por un supervisor.		

Escenario	Proceso	Usuario	Tiempo	Errores
	Evaluated		TPROC	TERI (Conteo)
			(min:seg)	
Línea Base	Registro	Usuario		
	manual de	A		
	Compra #1			
Línea Base	Registro	Usuario		
	manual de	B		
	Compra #2			
Validación	Registro	Usuario		
	en Sistema Web	A		
	#1			
Validación	Registro	Usuario		
	en Sistema Web	B		
	#2			
...				
<i>(Continuar N mediciones)</i>				