Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería de Sistemas

Implementación de un sistema de información web integrado impulsado por IA conversacional para empresas en el sector farmacéutico

Tesis presentada por el Bachiller:

Falcon Escalante, Paolo Fabrizio

ORCID: 0000-0002-3728-5586

para optar el Título Profesional de Ingeniero de Sistemas

Asesor:

Dr. Sulla Torres, José Alfredo

ORCID: 000-0001-5129-430X

Arequipa - Perú

2024

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA DE SISTEMAS

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 25 de Junio del 2024

Dictamen: 009350-C-EPIS-2024

Visto el borrador del expediente 009350, presentado por:

2016801771 - FALCON ESCALANTE PAOLO FABRIZIO

Titulado:

IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN WEB INTEGRADO IMPULSADO POR IA CONVERSACIONAL PARA EMPRESAS EN EL SECTOR FARMACÉUTICO

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO DE SISTEMAS

29302116 - DELGADO DELGADO FREDY RAMIRO
DICTAMINADOR

29393323 - ZUÑIGA CARNERO MANUEL MARIANO
DICTAMINADOR

29244573 - PAREDES MARCHENA FERNANDO GERMAN
DICTAMINADOR

Implementación de un sistema de información web integrado impulsado por IA conversacional para empresas en el sector farmacéutico

INFORME DE ORIGINALIDAD

11%	11%	3%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.ug.edu.ec Fuente de Internet	1%
2	Submitted to Universidad Católica de Santa María Trabajo del estudiante	1%
3	tesis.ucsm.edu.pe Fuente de Internet	<1%
4	www.coursehero.com Fuente de Internet	<1%
5	issuu.com Fuente de Internet	<1%
6	repositorio.uci.cu Fuente de Internet	<1%
7	dokumen.pub Fuente de Internet	<1%
8	sedici.unlp.edu.ar Fuente de Internet	<1%

DEDICATORIA

A mis amados padres, quienes con su inagotable esfuerzo y generosidad me brindaron la oportunidad de acceder a una educación de calidad. A mi querida abuela, cuya incondicional confianza e interés en mi desarrollo profesional han sido un faro en mi camino. A los miembros de mi familia, pilar fundamental, cuyo tiempo y disposición han sido esenciales para el éxito de proyectos como este.

AGRADECIMIENTOS

Agradezco sinceramente a quienes han sido fundamentales en mi camino académico y profesional. En primer lugar, a mis padres, cuyo generoso apoyo me permitió acceder a una educación de calidad. Mi gratitud se extiende a mi abuela, quien ha sido un pilar constante en mi vida, brindándome su apoyo y siguiendo de cerca mi desarrollo profesional. En segundo lugar, quiero expresar mi agradecimiento a mis seres queridos, quienes dedicaron su tiempo y disposición para contribuir al éxito de proyectos como este. Un reconocimiento especial a mis amigos, compañeros de viaje en esta travesía académica, que compartieron el mismo compromiso de avanzar y culminar esta carrera. Especial mención merecen aquellos amigos más cercanos, con quienes he compartido vivencias dentro y fuera de las aulas. En tercer lugar, mi reconocimiento a los docentes, cuya dedicación y esfuerzo fueron fundamentales para transmitirme conocimientos profesionales y valiosas experiencias de vida. Por último, pero no menos importante, agradezco a mis compañeros de trabajo y amigos en la empresa, quienes contribuyeron significativamente proporcionándome la experiencia necesaria en el ámbito profesional para alcanzar mis metas personales y profesionales.

RESUMEN

Los recientes avances y popularidad de los modelos de IA generativos permiten que más negocios busquen integrar este tipo de soluciones a sus operaciones diarias. Siguiendo esta tendencia, en este informe se describen los procesos de diseño, desarrollo e implementación de un sistema web integrado para el sector farmacéutico apoyado por un modelo de procesamiento de lenguaje generativo a modo de *chatbot* conversacional. La aplicación implementada automatiza los procesos de negocio de compras, ventas, inventario, catálogo y reportes de farmacias, además de estar construido con un enfoque multiempresa y multisucursal. La integración de la aplicación con un *chatbot* impulsado por IA Conversacional permite a los usuarios consultar sobre información directamente desde la base de datos utilizando lenguaje natural, esta herramienta demostró mejorar la usabilidad de la aplicación, facilitando el acceso rápido e intuitivo a información actual. Se lograron satisfacer con éxito los requerimientos del cliente, además de mejorar muchos aspectos relevantes de la anterior versión de la plataforma. Finalmente, el producto obtenido cumplió con los estándares de calidad y seguridad, logrando optimizar los procesos de negocio y mejorando la toma de decisiones de múltiples cadenas farmacéuticas nacionales.

Palabras clave: soluciones TI en farmacia, metodología ingeniería web, sistema de información web, modelos de procesamiento de lenguaje, OpenAI API, IA generativa.

ABSTRACT

Recent advances and popularity of generative AI models are allowing more businesses to seek to integrate this type of solution into their daily operations. Following this trend, this report describes the design, development, and implementation processes of an integrated web system for the pharmaceutical sector supported by a generative language processing model in the form of a conversational chatbot. The implemented application automates the business processes of purchasing, sales, inventory, catalog, and pharmacy reports, in addition to being built with a multi-company and multi-branch approach. The integration of the application with a chatbot powered by Conversational AI allows users to query information directly from the database using natural language, this tool has proven to improve the usability of the application, facilitating quick and intuitive access to updated information. The client's requirements were successfully met, in addition to improving many relevant aspects of the previous version of the platform. Finally, the product obtained met quality and safety standards, achieving optimization of business processes and improving the decision-making of multiple national pharmaceutical chains.

Keywords: pharma IT solutions, web engineering methodology, web information system, large language model, OpenAI API, generative AI.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, la integración de soluciones tecnológicas es indispensable para el éxito de las empresas, siendo el sector farmacéutico un claro ejemplo de esta tendencia. Las Tecnologías de la Información y la Comunicación (TIC) desempeñan un papel crucial al buscar la optimización de los procesos de negocio y la aplicación de una gestión eficiente en diversas empresas. En el rubro de las farmacias, las operaciones abarcan actividades de compras, ventas, cotizaciones, gestión de deudas, reportes, productos, proveedores y listas de precios, entre otros. Por lo tanto, es altamente recomendable implementar un sistema de información moderno capaz de respaldar todas estas operaciones.

El presente proyecto de tesis documenta el proceso de desarrollo e implementación de un sistema integrado basado en web orientado al rubro farmacéutico, con el objetivo de mejorar la eficacia y eficiencia de los procesos de negocio de empresas en este sector, además de implementar una solución innovadora adicional para contribuir con la interacción humano-computador. El sistema propuesto busca abordar los desafíos y necesidades específicas de las farmacias, tales como la gestión de inventario, el control de ventas, compras, deudas, trámites de facturación y demás procesos; además de implementar una herramienta basada en procesamiento de lenguaje natural para que los usuarios puedan interactuar con información actual de una manera intuitiva y rápida.

El primer capítulo de esta tesis se centra en la exposición de los objetivos del proyecto, el alcance de este, antecedentes del proyecto, fundamentos teóricos, técnicas y herramientas utilizadas para la investigación, la metodología elegida para el desarrollo de *software* y los aspectos relevantes de los procesos de análisis, diseño, codificación, documentación, pruebas e implementación del sistema integrado.

En el segundo capítulo se abordará a detalle cada aspecto relevante de forma técnica incluyendo la planificación temporal del proyecto, el estudio de viabilidad, definición de los

requisitos funcionales y no funcionales del sistema, el proceso de diseño del sistema, diseño de la base de datos, codificación de módulos y medidas de seguridad en el sistema.

En el tercer capítulo se expondrá la adición de una característica diferenciadora frente al mercado, la integración de una herramienta de IA Conversacional a las operaciones del sistema. Se describirá su propósito e importancia, los requisitos del proyecto para adaptar esta solución, la arquitectura de la integración, el proceso de implementación, alcance y limitaciones de esta herramienta.

Finalmente, en el cuarto capítulo se documentarán los resultados de las pruebas realizadas a la aplicación desde enfoques como la seguridad, integración rendimiento y usabilidad; también se incluirán las pruebas realizadas al módulo del *chatbot*. Se documentará la evaluación del proyecto utilizando las normas ISO 25010:2023 y 27001:2022, para finalmente describir los KPIs del proyecto y las mediciones tomadas para cada uno de estos.

La relevancia de este proyecto radica en la mejora de la optimización de procesos de negocio de farmacias mediante una aplicación moderna, utilizando tecnologías y metodologías contemporáneas, además de incluir la característica diferenciadora de un *chatbot* impulsado por IA conversacional para consultas sobre información actual. El sistema integrado propuesto tiene como objetivos mejorar la toma de decisiones, la usabilidad y seguridad del sistema, optimizar los procesos de ventas, compras, inventario, catálogos, clientes y generación de reportes. Además, se espera que este proyecto sirva como punto de partida para futuras investigaciones y desarrollos en el campo de las TIC aplicadas al rubro farmacéutico.

CAPÍTULO I

1 DESCRIPCIÓN DEL PROYECTO

1.1 Objetivos

1.1.1 Objetivo General

Implementar un innovador sistema web integrado destinado a empresas pertenecientes al sector farmacéutico, impulsado por modelos de procesamiento de lenguaje natural para la construcción de un *chatbot* que facilite el acceso a información actual en la base de datos.

1.1.2 Objetivos Específicos

1. Recopilar y documentar los requerimientos funcionales y no funcionales del proyecto comprendiendo las necesidades de los usuarios, planificación temporal y estudio de viabilidad de este.
2. Ejecutar las etapas de diseño e implementación de un sistema web integrado para procesos de negocio propios de farmacias mientras se lleva un desarrollo basado en iteraciones y actividades definidas en la metodología ágil Ingeniería Web.
3. Integrar un *chatbot* impulsado por IA conversacional apoyado por un modelo de la familia GPT 3, donde los usuarios realicen consultas en lenguaje natural para obtener información actual desde la base de datos de forma rápida e intuitiva, mejorando así la usabilidad del sistema.
4. Definir pruebas para medir distintas características de la aplicación como la seguridad, integración, usabilidad y rendimiento. Registrar los resultados de estas con la finalidad de evaluar la calidad del producto *software*.
5. Hacer uso de las normas ISO para mejorar la calidad y seguridad del producto de *software*, además de diferenciar el sistema integrado de otras soluciones similares para empresas en el sector farmacéutico.

1.2 Caracterización del Problema

La empresa Peruana de Consultoría Digital E.I.R.L. tiene como clientes a cadenas de farmacias a lo largo de la zona sur del país, el sistema integrado que esta desarrolladora les ofrece se encuentra en funcionamiento y hace el trabajo fundamental, sin embargo, las tecnologías sobre las que fue construido se hallan desactualizadas, algunas han quedado obsoletas y muchos de los requerimientos actuales no pueden ser cubiertos con actualizaciones periódicas al *software*.

Entre las limitaciones de la aplicación se encuentran inconsistencias en el modelado de los procesos de negocio, dificultad de mantenimiento debido a la pobre optimización del código y poca fluidez en la navegación en la interfaz de usuario. Esto trae como consecuencias que la corrección de errores sea lenta, las oportunidades de mejora se vean limitadas por tecnología arcaica, el control sobre los procesos sea mínimo y el mantenimiento tenga un coste mayor en tiempo y recursos.

1.3 Justificación del Proyecto

Las aplicaciones web son la forma más popular y rápida para desarrollar soluciones *software* para diversos tipos de negocio, por esta razón, el catálogo de recursos compatibles con este tipo de tecnología es amplio, actualizado y accesible. Las principales características de estos sistemas son la facilidad de uso, bajos costos en *hardware* y *software*, escalabilidad, integridad de datos, seguridad, accesibilidad y facilidad de uso (International Organization for Standardization, 2022, 2023). Por este motivo, se dio inicio al proyecto de una plataforma web que funcionara como un sistema integrado especializado en el sector farmacéutico, para reemplazar la anterior versión de la aplicación.

Llevando un proceso de desarrollo ágil, apoyado por la metodología IWeb, haciendo uso de tecnología contemporánea, priorizando la calidad y seguridad del producto mediante

estándares ISO 25010 y 27001 (International Organization for Standardization, 2022, 2023), la construcción de una aplicación de gestión para farmacias fue posible.

Además, con el desarrollo de este proyecto se presentaría una oportunidad de mejora significativa como lo es la integración con modelos conversacionales de inteligencia artificial, específicamente, con *large language models (LLMs)*, siendo los más populares los modelos de la familia GPT. Esta integración tuvo el objetivo de mejorar en gran medida la usabilidad del sistema, para que el acceso a la información por parte de los usuarios fuera intuitivo, realizando preguntas en lenguaje natural en una ventana de chat convencional, pero a su vez, una potente herramienta para funcionalidades algo más avanzadas dentro del sistema.

1.4 Alcances y Limitaciones

El proyecto se concibió como un sistema integrado con un enfoque multiempresa y multisucursal para negocios del servicio farmacéutico, definiendo cada tipo de empresa dentro de este sector de la siguiente manera:

- Farmacia – son establecimientos gestionados por un profesional químico farmacéutico, autónomos en administración, atención y consejería de los usuarios (Perú21, 2017).
- Botica – son establecimientos farmacéuticos propiedad de empresarios, se contratan profesionales químicos farmacéuticos, sin embargo, no tienen capacidad de decisión sobre negociaciones comerciales (Perú21, 2017).
- Droguería – son negocios dedicados al comercio de productos farmacéuticos, dispositivos médicos y productos sanitarios. Tiene una modalidad de venta a comercios, es decir, boticas y farmacias (DIGEMID, 2023).

De forma general, los módulos que esta plataforma engloba son: almacén, ventas, compras, movimientos en la caja contable, cuentas por cobrar, cuentas por pagar, productos, marcas, categorías, laboratorios, listas de precios, clientes, proveedores, empresas y

sucursales. Adicionalmente, ofrece opciones de emisión reportes para DIGEMID (Dirección General de Medicamentos, Insumos y Drogas), lotes de productos, compras, ventas y el kárdex valorizado. También, es capaz de conectar con la plataforma de declaración de comprobantes de la SUNAT y otros servicios adicionales.

Por la parte de la integración con modelos conversacionales, este desarrollo abarcó una primera versión del *chatbot*, donde el enfoque central fue el diseño y construcción de una arquitectura que permitiese obtener información actualizada desde la base de datos por medio de preguntas simples en lenguaje natural a través de un modelo de IA conversacional.

Otros puntos que describen el alcance y las limitaciones de este proyecto se listan a continuación.

- **Lugar.** La empresa desarrolladora tiene como sede la ciudad de Arequipa, sin embargo, los clientes de este proyecto se encuentran en distintas regiones del Perú. Debido a que se trata de un proyecto TIC, la posibilidad de llevar un proceso de desarrollo remoto es factible, agendando reuniones con clientes tanto de forma presencial como virtual.
- **Tiempo.** Debido a que las farmacias contaban con una plataforma de gestión anterior, este proyecto podía aplazarse indefinidamente, sin embargo, se estableció que la duración adecuada fuese de menos de un año, finalmente resultando en una planificación de nueve meses y medio, como se presenta en el punto **2.1.1**.
- **Financiamiento.** La desarrolladora cuenta con la experiencia y el personal necesario para desarrollar un proyecto de esta magnitud. En términos monetarios, los costos más significativos a lo largo del desarrollo del sistema son el pago al equipo de desarrollo, el pago de servidores y gastos adicionales por reuniones o actividades similares.

1.5 Fundamentos Teóricos

1.5.1 Antecedentes del proyecto

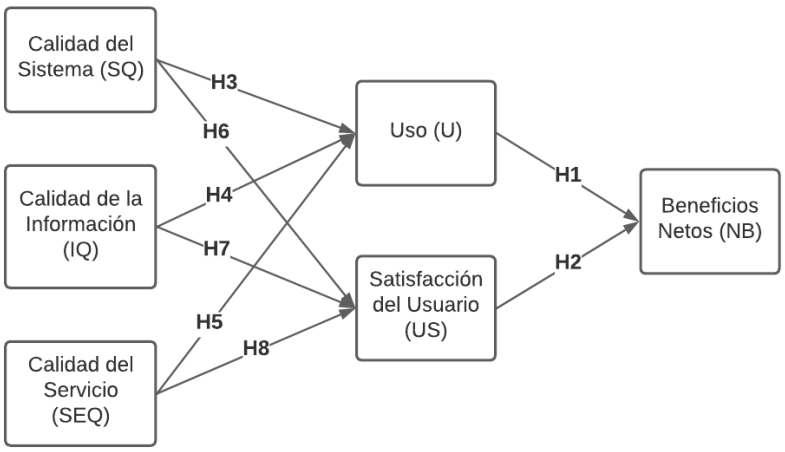
1.5.1.1 Implementación de Sistemas de Información en Farmacias.

1.5.1.1.1 Análisis de Factores Críticos de Éxito.

La implementación de un sistema de información en cualquier tipo de organización requiere de un estudio del entorno con la finalidad de que este sea exitoso y cumpla con los requerimientos del usuario. El Modelo de Éxito de Sistemas de Información (ISSM) describe las variables que tendrán un grado de influencia sobre el proyecto de implementación de acuerdo con el rubro al cual éste se encuentre orientado, como se muestra en **Figura 1**.

Figura 1

Modelo de Éxito de Sistemas de información



Nota. Modelo propuesto por DeLone y McLean donde se muestran nombres, abreviaciones e hipótesis que relacionan las variables entre sí. Adaptada de (Lubis et al., 2019).

Los autores Lubis et al. plantearon una serie de hipótesis en su estudio sobre los factores críticos de éxito en la implementación de un ERP en farmacias, con esta investigación acometieron identificar qué variables predominaban (Lubis et al., 2019). Diseñaron entonces un modelo de medición basado en lo descrito en el ISSM, haciéndose preguntas básicas sobre si existían uno o más indicadores ocultos, si estos indicadores aparecían en más de una variable, la frecuencia con las que estos indicadores se presentaban y

si estos afectaban o no a cada una de las variables; sobre estas preguntas se utilizaron técnicas que corroboren las hipótesis como: consistencia interna, validez convergente y divergente.

El resultado del análisis reafirmó tres de las hipótesis planteadas por los autores; en primer lugar, que la variable “uso del sistema” influye fuertemente a la variable “beneficios netos”. En segundo lugar, la “satisfacción del usuario” también favorece significativamente a los “beneficios netos”. Y, en tercer lugar, la “calidad del sistema” tiene un alto grado de influencia sobre el “uso del sistema” (Lubis et al., 2019).

1.5.1.1.2 Proyectos de Implementación de Sistemas de Información.

Los proyectos de desarrollo de plataformas TI para farmacias son variados y tienen alcances distintos, ya que se pueden enfocar en una sola área o grupo de procesos, pero también son capaces de alcanzar una magnitud mayor abarcando toda la organización y procesos en los que actúan entidades externas.

En el proceso de análisis, diseño e implementación de un sistema de control de inventarios para la farmacia “Danafarma”, el equipo de desarrollo hizo uso de la metodología tradicional RUP con el fin de elaborar un producto de calidad que se ajuste a los requerimientos del usuario a la vez que llevaba un correcto control de los avances. Este proyecto surgió de la necesidad de integrar una solución de TI a un negocio donde los procesos se llevaban de forma manual, lo que acarreaba desventajas como: deficiencias de abastecimiento, pérdidas de stock, búsqueda de medicamentos y productos vencidos (Navarro, 2019). El sistema de información logró dar solución a estos errores permitiendo que el usuario final mejore su desempeño, también se logró apreciar que la información que el *software* proporcionaba era precisa y oportuna, por lo que la toma de decisiones mejoró; además, ha permitido reducir drásticamente el tiempo de búsqueda, acceso y abastecimiento de medicamentos en almacén (Navarro, 2019).

Otro ejemplo actual es el proyecto de implementación de un sistema de gestión para la empresa “Megafarma”, donde se evaluó y documentó el grado de mejora en la gestión una vez la plataforma se encontró disponible. El equipo de desarrollo utilizó las metodologías RUP y UML con el fin de preservar la calidad; el alcance del proyecto abarcaba principalmente los procesos de compra, venta e inventarios. El producto final fue una aplicación de escritorio funcional en Windows 10 e integrada con MySQL para la conexión con la base de datos. Finalmente, los autores verificaron que el uso del sistema favoreció de forma significativa en la optimización de procesos de negocio, reduciendo el tiempo de venta en 4.33 minutos, el tiempo de generación de una orden de compra en 0.89 horas y el proceso de gestión de inventario en una media de 0.38 días (Delao, 2018). La conclusión a la que los autores llegaron fue que contar con un sistema de información es una ventaja competitiva en el negocio farmacéutico, ya que hace posible la disposición de información actualizada y en tiempo real; como consecuencia, se mejora la toma de decisiones (Delao, 2018).

1.5.1.2 Sistemas de Información Apoyados por IA Generativa.

La reciente popularidad de modelos de inteligencia artificial generativos en conjunto con el desarrollo acelerado de *hardware* especializado en cálculos computacionales complejos ha permitido que todos los sectores de mercado se vean beneficiados de alguna forma por este tipo de tecnología. *Chatbots* conversacionales, generación de imágenes de extrema calidad, conducción autónoma y avatares virtuales realistas son sólo unos ejemplos de los muchos casos de uso que han adoptado este tipo de modelos de IA. Igualmente, han surgido estudios recientes explorando las aplicaciones adicionales que una tecnología podría tener, comparando el desempeño de estas frente a métodos más tradicionales.

Como primer ejemplo, los autores Smoliński et al. hicieron uso del modelo de generación de imágenes StableDifusssion 1.5 para crear *posters* publicitarios artificiales tomando como referencia *posts* de campañas de marketing reales con el fin de comparar el

impacto de cada uno en los usuarios (Smoliński et al., 2023). Un primer estudio reunió a 55 estudiantes universitarios a los cuales se les presentaron 10 anuncios publicitarios (5 reales y 5 generados artificialmente), utilizando un cuestionario especializado los investigadores lograron calificar la percepción, respuesta emocional y compromiso del consumidor para cada una de estas imágenes (Smoliński et al., 2023). Los resultados fueron positivos, el impacto en la actitud, intención y satisfacción de los usuarios por parte de los *posts* generados artificialmente fue mayor que los anuncios publicitarios hechos por profesionales (Smoliński et al., 2023). Un segundo estudio de los mismos autores logró expandir los resultados del primero, se buscó evaluar la capacidad de los modelos para generar publicidad personalizada sobre distintos grupos de personas clasificándolos por sus rasgos de personalidad: extrovertidos e introvertidos. En este caso se utilizaron los modelos GPT-4 y StableDifussion 1.5 para generar publicaciones que uno encontraría normalmente en redes sociales, 53 estudiantes universitarios fueron encuestados utilizando tres *posts* (1 real y 2 generados artificialmente), posteriormente se les hizo un *test* de personalidad (Smoliński et al., 2023). Nuevamente los resultados fueron favorables para los modelos de IA generativa, 44 de los 53 estudiantes eligieron publicaciones generadas artificialmente como sus favoritas (Smoliński et al., 2023). Finalmente, los autores concluyeron que la construcción de un sistema de información especializado sería un gran avance, donde se almacenaría información detallada del consumidor para generar publicidad personalizada artificialmente, además de llevar un control de la retroalimentación por parte de los usuarios, esto agilizaría la creación de campañas publicitarias y permitiría contar con una aplicación segura, mantenible y de alto rendimiento (Smoliński et al., 2023).

En un segundo ejemplo, los autores Acharya et al. hicieron uso del LLM Alpaca-Lora y un el *dataset* público MovieLens 1m, el cual contiene un millón de reseñas de usuarios sobre cuatro mil películas, con el fin de entrenar un LLM que alimente sistemas de

recomendación personalizada de películas generando descripciones precisas sobre estas (Acharya et al., 2023). La principal motivación para la implementación de un modelo de este tipo en la arquitectura de sistemas de recomendación fue que, tradicionalmente las descripciones de películas son extraídas de sitios web como IMDB mediante *web-scraping*, donde son escritas manualmente (Acharya et al., 2023). Sin embargo, se considera que los LLM realizan una mejor labor evitando sesgos, errores humanos y ahorrando recursos (Acharya et al., 2023). Al aplicar diversas métricas de rendimiento de los sistemas de recomendación de películas, este estudio demostró tener resultados nuevamente favorables para esta tecnología, las descripciones generadas por LLMs obtuvieron puntajes competitivos en sistemas de recomendaciones de películas frente a las descripciones hechas manualmente por escritores profesionales (Acharya et al., 2023).

En un último ejemplo, los autores Freire et al. compararon sistemas basados en LLMs con sistemas basados en intenciones. Un sistema de procesamiento de lenguaje natural basado en intenciones es aquel que se entrena con frases comunes de usuarios y se clasifican en distintos tópicos o intenciones, por ejemplo: un usuario escribe “Quiero hacer una reserva en el restaurante para las 4 de la tarde”, esta oración podría clasificarse como “reserva_hotel” en un sistema basado en intenciones (Freire et al., 2024). Debido a la naturaleza rígida de este tipo de sistemas, se buscó comparar su desempeño con aquellas aplicaciones basadas en un LLM, para esto se construyeron dos aplicaciones de *chatbots* basados en un asistente virtual para una fábrica industrial, uno donde las consultas del usuario se clasificarían como intenciones y el otro donde tendrían la libertad de conversar naturalmente para obtener respuestas. Se tomaron 55 estudiantes de maestría de diseño industrial, se les pidió que hicieran uso de ambos sistemas por un tiempo determinado con la libertad de poder realizar cualquier tipo de preguntas, se evaluaron campos como la realización de tareas, usabilidad del sistema y experiencia de usuario (Freire et al., 2024). Los resultados del experimento

clarificaron que el desempeño de los LLM es significativamente mayor a los modelos basados en intenciones, con argumentos como que los sistemas de intenciones son rígidos, fallan en dar una respuesta fácilmente y muchas veces se llegan a callejones sin salidas en la conversación; a diferencia de los sistemas basados en LLMs, donde la conversación fue mucho más natural e intuitiva y las consultas fueron respondidas siempre (Freire et al., 2024).

1.5.1.3 Sistemas de Información Basados en IWeb.

La metodología IWeb, es una de las formas más eficientes cuando de desarrollar aplicaciones web se trata. Por este motivo, numerosos autores han documentado el proceso de implementación de estos sistemas utilizando IWeb como la base del proyecto.

Los investigadores Barrientos E. y Rincón M. redactaron un informe detallando el proceso de desarrollo de una plataforma *ecommerce* para la compañía alimenticia "Tu Pan Gourmet SAS", debido a que el proyecto se basaría completamente en internet, el equipo optó por hacer uso de la metodología IWeb para brindarle al usuario accesibilidad, alto rendimiento, seguridad y ergonomía en la aplicación (Barrientos & Rincón, 2020). Este proyecto se dividió en iteraciones donde en cada una se cumplían con las seis etapas de IWeb, mismas que se explican en el punto **1.5.2.2**, repitiendo así este proceso hasta llegar a un producto mínimo viable. El resultado final fue una aplicación basada enteramente en web, la cual integra distintos módulos como: ventas, reportes, inventario, productos, etc. Además, incluye un *dashboard* en tiempo real el cual muestra información sobre el total de las compras, ventas, clientes registrados y productos vendidos.

Los autores concluyeron de este proyecto que lograron implementar el marco de trabajo de Pressman sobre la ingeniería web, tanto los empleados de la panificadora como el equipo de desarrollo se apoyaron de esta metodología ágil para trabajar en forma conjunta e iterativa, a la vez que se tomaban decisiones y se modificaba la plataforma conforme a las necesidades del cliente (Barrientos & Rincón, 2020). Finalmente, adicionaron un aplicativo

móvil al cual se le realizaron pruebas unitarias, rectificando que la calidad del *software* era apta para trabajar en multiplataforma (smartphones, tables y equipos de escritorio).

Un caso similar se expone en el proyecto "Diseño de un Sistema Web para el Control de Curriculum Vitae", en el que el equipo de desarrollo optó por hacer uso de IWeb con el fin de obtener un producto de calidad, actualizado y que satisfaga los requisitos de los usuarios. Esta aplicación se diseñó para que permitiese elaborar un Curriculum Vitae electrónico de docentes universitarios almacenando información como: datos personales, experiencia académica, experiencia profesional, logros y grados académicos (Castillo Estrada et al., 2022). Con esta información, el *software* sería capaz de generar gráficos estadísticos y reportes en formato PDF con el resumen del profesor evaluado.

La aplicación fue programada utilizando tecnologías actuales como lo son construcción de interfaces por componentes y orientada a servicios (APIs REST); también, se caracteriza por su escalabilidad, disponibilidad y estabilidad (Castillo Estrada et al., 2022). Finalmente, los autores observaron que el tiempo de organización de evidencias documentales se redujo significativamente, la generación reportes fue automatizada, la arquitectura de *software* orientado a servicios facilitó su implementación y se planea que el sistema crezca a futuro incorporando un aplicativo móvil (Castillo Estrada et al., 2022)

1.5.2 Bases Teóricas del proyecto

1.5.2.1 IA Generativa y Modelos de Lenguaje Extensos.

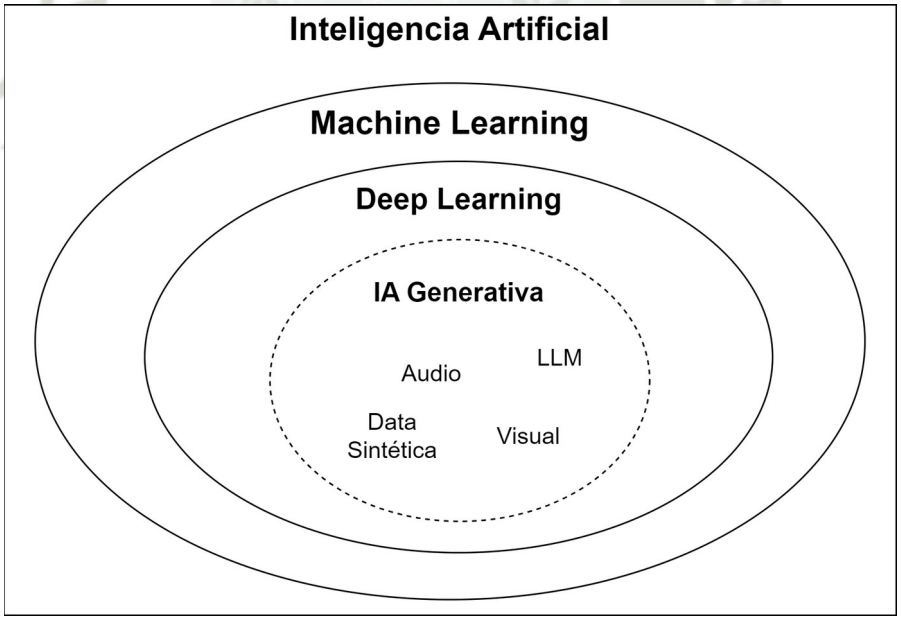
Hoy en día, gracias al incremento del poder computacional de los últimos años, las herramientas de IA disponibles se han visto potenciadas a tal punto de formar parte de nuestra vida diaria como los son los vehículos autónomos, generación de imágenes y video, sistemas de recomendación y asistentes virtuales para sistemas de información. Con el lanzamiento de GPT 3 en 2021 y GPT 4 en 2023 la utilización de *large language models (LLMs)* se

popularizó alrededor del mundo, surgieron oportunidades de éxito, propuestas de mejora y casos de uso tecnológicos basados en IA Generativa (Dhoni, 2023).

¿Pero qué es realmente la inteligencia artificial? Pues según autores se define como la percepción, síntesis e inferencia de información ejecutada por un computador, distinto a los procesos cognitivos que realizan los humanos y animales (Dhoni, 2023). El esquema general de las disciplinas más populares dentro del campo de la inteligencia artificial se estructura de la siguiente forma **Figura 2**. Podemos observar que la IA Generativa contiene una variedad de campos de estudio como lo son el procesamiento de video, audio, texto y data sintética (datos generados artificialmente para entrenamiento de modelos) (Dhoni, 2023; Nvidia Corporation, 2023).

Figura 2

Campos de estudio de la inteligencia artificial y disciplinas de la IA Generativa



Nota. La IA Generativa es un subconjunto del *deep learning*, contiene campos de investigación en procesamiento de audio, generación de imágenes y video, generación de data sintética y procesamiento de lenguaje natural (Mahmood, 2023; Nvidia Corporation, 2023).

El concepto de IA Generativa hace referencia a la capacidad de algunos modelos para crear nuevo contenido a partir de *inputs* (Dhoni, 2023; Mahmood, 2023; Nvidia Corporation,

2023), por ejemplo, describir una escenario de una pareja bajo la lluvia y recibir como *output* una imagen realista de esta escena, o pedir que se realicen correcciones ortográficas sobre un párrafo proporcionado y obtener una respuesta correctamente redactada por parte del modelo. Además, se definen tres características clave para asegurar la calidad de un modelo de IA generativa:

- **Calidad.** La calidad del contenido generado es fundamental, y es un factor de mayor peso para los casos en los que los usuarios interactúan directamente con el modelo, por ejemplo, la calidad de audio de un discurso generado o la capacidad de una imagen generada por IA de ser indistinguible de una imagen real.
- **Diversidad.** Los mejores modelos generativos son capaces de capturas la diversidad presente en los datos, sin perder calidad y evitando contenido sesgado.
- **Velocidad.** Al igual que con la calidad del contenido, la velocidad de generación también debe considerarse una prioridad cuando se interactúa con usuarios, por ejemplo, un modelo que requiere extraer datos clave en tiempo real de un audio enviado por un usuario. (Nvidia Corporation, 2023)

Por otra parte, los LLM son un tipo de IA Generativa **Figura 2**, se considera el campo de investigación donde se han realizado los mayores avances hasta la fecha y son la base de muchos otros modelos generativos populares. Los casos de uso para este tipo de modelos son diversos: filtrado de spam, publicidad online, diagnósticos médicos, investigación de medicamentos y análisis de sentimientos (Nvidia Corporation, 2023).

Los LLM son capaces de recibir una variedad de tipos de datos como *input*, mejor conocido como *prompt*, es capaz de incluir documentos, imágenes o almacenes de vectores. La forma en la que los LLM generan contenido es completando el *prompt* recibido con palabras aprendidas durante su entrenamiento, una a una en un proceso secuencial (Mahmood, 2023; Nvidia Corporation, 2023). Por ejemplo, un LLM en entrenamiento recibe como

entrada la oración “Mi deporte favorito son las artes marciales mixtas”, en este caso el modelo aprendería que las palabras que siguen a “Mi deporte favorito” serían “son las”, luego también podría deducir que la siguiente frase podría ser “artes marciales” y finalmente la palabra “mixtas”. Es por este mismo motivo que la cantidad de datos de entrenamiento necesarios para construir un LLM es masiva, se cuentan por millones o trillones de palabras (Mahmood, 2023).

1.5.2.2 Uso de Metodologías Ágiles en Proyectos de Desarrollo de Software.

Las metodologías aplicadas a proyectos *software* juegan un rol crítico en estos, ya que son capaces de determinar tanto la calidad del producto final como la adecuada gestión del proceso de desarrollo mientras incorporan los planes, métodos, actores y controles correspondientes. Existen dos grupos cuando se habla de tipos de metodologías: tradicionales y ágiles.

Las metodologías tradicionales garantizan resultados de alta calidad mediante un riguroso proceso de recopilación de requerimientos previo a las etapas de análisis y diseño. Tienen un flujo lineal no iterativo por lo que, encontrándose en determinada etapa de desarrollo en la que se requiere un cambio, no es posible realizar avances en la siguiente o volver a la anterior sin haber trazado un nuevo plan con la gestión de cambios pertinente (López-Mora et al., 2019).

Por otra parte, en la práctica, las metodologías ágiles se distinguen de las tradicionales por conceptos como la priorización de los individuos y sus interacciones antes que las herramientas y procesos involucrados; es decir, el principal objetivo es construir un producto *software* funcional sujeto a cambios más que redactar documentación detallada o seguir las políticas impuestas para el proyecto (López-Mora et al., 2019). Con este tipo de metodologías los tópicos como la colaboración con el cliente, la gestión de cambios y comunicación activa son más relevantes que el seguimiento estricto de un plan o las negociaciones de un contrato.

Las características de los proyectos ágiles son las siguientes:

- Cada proyecto ágil es independiente y se compone por subproyectos con características similares.
- La comunicación con el cliente es indispensable, constantemente se tienen reuniones con él o un representante de este.
- Se adaptan a los cambios con facilidad ya que estos son esperados y están diseñados específicamente para ellos.
- Las entregas al cliente y la retroalimentación son comunes y necesarias para conseguir un producto de calidad.
- Son de gran valor para grupos pequeños (menos de 10 miembros) trabajando en los mismos proyectos y haciendo uso de canales de comunicación no formales (López-Mora et al., 2019).

Sin embargo, las metodologías ágiles no son perfectas y cuentan con claras desventajas cuando se las compara con las tradicionales; por ejemplo, la posibilidad de no existencia de contratos prefijados, poco control sobre el proyecto y baja o nula normalización en el desarrollo (López-Mora et al., 2019). Por otra parte, los beneficios de este tipo de metodologías son los siguientes:

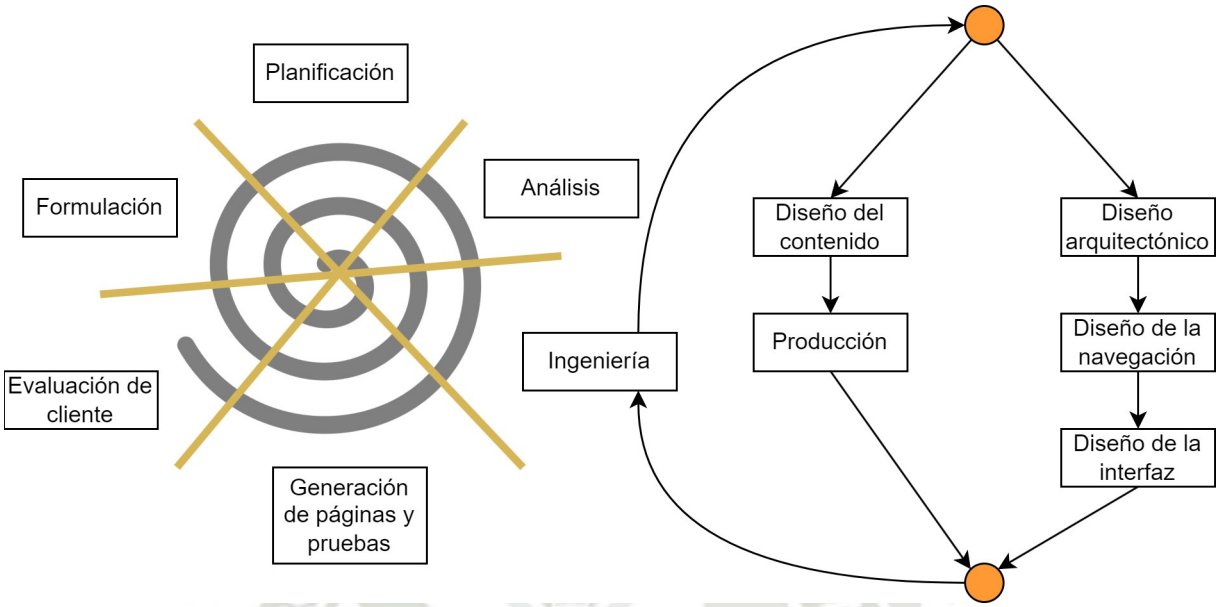
- Son funcionales en un ambiente y contexto donde el equipo de desarrollo necesita trabajar con los clientes, involucrando a estos últimos en la toma de decisiones sobre los requerimientos del proyecto.
- Son simples de aprender y aplicar, por lo que un equipo nuevo no tendrá mayor inconveniente al hacer uso de ellas.
- Minimizan costos relacionados con el equipo de desarrollo y capacitación de estos.
- Hacen uso eficiente del tiempo, concentrándose más en el desarrollo que en la documentación rigurosa (López-Mora et al., 2019).

1.5.2.3 Ingeniería Web (IWeb).

Ingeniería Web o IWeb es una metodología ágil propuesta por el ingeniero Roger S. Pressman que surge a partir de la aplicación de fundamentos ya establecidos en la Ingeniería de *Software* adicionando determinados procesos, métodos, técnicas y modelos que satisfagan las características propias de las aplicaciones web modernas (Castillo Estrada et al., 2022). Muchas de estas necesidades se originan debido a que el desarrollo web, a comparación del desarrollo de *software* tradicional, involucra a personas las cuales no necesariamente están familiarizadas con la programación, sino que se especializan en otros campos como lo son el diseño gráfico, diseño de bases de datos, análisis de sistemas, ciberseguridad y gestión de proyectos (Pinzon & Rodríguez, 2017). A diferencia del desarrollo de *software* tradicional, las aplicaciones web suelen ser más dinámicas, interactivas y requieren una colaboración estrecha entre equipos multidisciplinarios. IWeb reconoce estas necesidades y proporciona un marco flexible y adaptable que facilita la comunicación, colaboración y entrega continua de valor. Una forma clara en la que IWeb habilita la interacción entre desarrolladores y el resto de participantes del proyecto se encuentra en su estructura de organización en “regiones de tareas”, como se muestra **Figura 3** (Barrientos & Rincón, 2020; Molina et al., 2018).

Figura 3

Fases e iteraciones de la metodología IWeb



Nota. Actividades de IWeb y la propiedad iterativa de esta metodología de desarrollo. Adaptada de (Molina et al., 2018).

Las actividades descritas por Roger S. Pressman sobre la Ingeniería Web son aplicables a cualquier proceso de desarrollo de aplicaciones web sin considerar su magnitud, complejidad o alcance (Molina et al., 2018; Pinzon & Rodríguez, 2017). Se especifica el detalle de cada una de las fases a continuación.

1. **Formulación.** Se establecen los objetivos, el alcance, las metas y una primera versión de la entrega, además, es necesario definir la motivación, las necesidades por las cuales se desarrolla la aplicación y los beneficiados de esta.
2. **Planificación.** Se analizan los riesgos y se crean planes de contingencia, se diseña el cronograma junto a las fechas de entrega, el costo global es estimado, se asignan responsables y se listan los recursos.
3. **Análisis.** Se registran los requerimientos técnicos y de diseño del sistema, el contexto de cada uno de ellos y como se integrará el contenido; se realizan cuatro tipos de análisis distintos: contenido, interacción, funcionalidad y configuración.

4. **Ingeniería.** Esta etapa se segmenta en diseño de arquitectura, donde surgen múltiples modelos (datos, interacción, configuración, funcional, etc.), y diseño producción del contenido, donde se modelan interfaces y navegación del usuario.
5. **Generación de páginas y pruebas.** Se construye el sistema haciendo uso herramientas especializadas en desarrollo de aplicaciones web, se integran los modelos obtenidos de la etapa anterior. Finalmente, se realizan pruebas de interfaz, navegación y componentes de las páginas web con el fin de comprobar el buen funcionamiento y encontrar posibles errores.
6. **Evaluación del cliente.** Permite identificar oportunidades de mejora, cambios y errores. Se presentan entregas parciales y el proyecto es desarrollado involucrando al cliente en el proceso asegurando su satisfacción según los requerimientos solicitados (Barrientos & Rincón, 2020; Castillo Estrada et al., 2022; Molina et al., 2018; Pinzon & Rodríguez, 2017).

Las actividades listadas son entonces incluidas en cada una de las iteraciones que el proyecto requiera; al concluir cada iteración se tendrán identificados los errores y mejoras, se obtendrán modelos actualizados del sistema, se agregarán o modificarán requerimientos y lo más importante, la aplicación web será refinada de forma gradual.

1.6 Técnicas y Herramientas

1.6.1 Técnicas

Tabla 1

Listado y descripciones técnicas

Técnica	Descripción
Técnicas de ingeniería de requisitos	Las técnicas en ingeniería de requisitos comprenden: entrevistas, tormentas de ideas, diagramas de clases, diagramas de casos de uso, diagramas de secuencia, etc. Esto con el fin de optimizar la recopilación de los requisitos en cada etapa del proyecto, además de permitir gestionar las líneas base y solicitudes de cambio (Toro & Peláez, 2016).

Juicio de expertos	Hace referencia al juicio que pueden desarrollar las distintas personas involucradas en el proyecto de acuerdo con su grado de conocimiento en un área de aplicación. Para el presente proyecto, esta técnica será usada tanto al inicio como a lo largo del proceso de monitoreo con el fin de identificar oportunidades de mejora, riesgos y beneficios (Project Management Institute, 2021).
Análisis de alternativas	Técnica usada para aplicar acciones correctivas y preventivas frente a sucesos inesperados (Project Management Institute, 2021) se hará uso de esta para dar soluciones tecnológicas a requerimientos de alto nivel.
Análisis de causa raíz	Útil para identificar el origen de un problema y factores que influyen en este con el fin de tomar una decisión en base a los objetivos del proyecto (Project Management Institute, 2021).
Análisis de variación	Esta técnica implica registrar los cambios en tiempo, costo, recursos y técnicas respecto al plan de dirección del proyecto. La identificación de estas variaciones en métricas habilita al equipo de desarrollo iniciar medidas preventivas y correctivas (Project Management Institute, 2021).

Nota. Elaboración propia.

1.6.2 Herramientas

Tabla 2

Listado y descripciones de herramientas

Herramienta	Descripción
Reuniones	Las reuniones serán cruciales para informar a los interesados clave sobre el estado del proyecto, identificación de objetivos, entregables e hitos. También son indispensables para la recopilación de requisitos de alto nivel, cambios en el proyecto y definición de criterios de éxito (Project Management Institute, 2021).
Diagrama Gantt	Útil para la gestión de proyectos, es una representación gráfica de la planificación de este incluyendo tareas, fechas y cronograma (Meardon, 2022). Se hará uso de esta herramienta para definir el plan de trabajo del proyecto, puede visualizarse en el punto **2.1.1**.
Método Kanban	Utilizando el *software* de gestión de trabajo “Trello” se dará seguimiento al proyecto de forma visual mediante el sistema Kanban, también se incluirán

	características colaborativas, de planificación y ejecución del proyecto para contribuir con el seguimiento y control de forma compartida (Atlassian, 2022).
Notion	Herramienta flexible de organización de tareas, horarios, recursos, etc. En el caso específico de este proyecto, es utilizada para la toma de notas en reuniones: tomas de decisiones, cambios en planes, incidentes y riesgos (Calebio & Notion Labs Inc, 2022).

Nota. Elaboración propia.

1.7 Aspectos Relevantes del Desarrollo

1.7.1 Análisis y Diseño

1.7.1.1 Recopilación de Requisitos.

Este proyecto se planteó como una nueva versión mejorada de un anterior sistema orientado a farmacias, por este motivo, muchos de los requisitos para este sistema se enfocan más en la mejora de procesos y procedimientos en lugar del procesamiento específico de los datos. En esta etapa se definió el alcance del proyecto, los roles de cada miembro del equipo de desarrollo, las reuniones con los clientes y finalmente la recopilación de requerimientos. De forma resumida, el resultado de la recopilación de información desde los clientes se muestra a continuación:

1. El proyecto debe ser construido en base a tecnologías vigentes que contribuyan con la facilidad de uso de la plataforma.
2. El sistema en conjunto con las tecnologías elegidas debe garantizar la estricta seguridad de la información.
3. La plataforma debe incluir todos los procesos de negocio de farmacias, también mejorar las características de la anterior versión del sistema con el fin de garantizar que el procesamiento de la información sea preciso.

4. Los procesos de negocio implementados en el *software* deben estar actualizados al presente año, ya que muchas reglamentaciones a farmacias se han modificado y esto se ve reflejado en la presentación de reportes, formatos, descuentos, etc.
5. El sistema debe contar con la suficiente ergonomía para que un usuario pueda navegar por él de una forma intuitiva, sin requerir de conocimiento técnico ni abrumarse por la complejidad de la interfaz.

Posteriormente, se redactó la especificación de requisitos donde se listaron de forma descriptiva, puede encontrar más detalles técnicos sobre este proceso en el punto **2.2**, también puede visualizar el documento completo en **Anexo 5**.

1.7.1.2 Diseño de la Base de Datos.

Para la construcción de la base de datos se tomó en cuenta la forma en la que se almacenaba la información en las tablas de la anterior versión del sistema, se usaron las tres primeras formas normales sobre ellas, se hicieron cambios significativos y se agregaron nuevas tablas adicionales que mejoraron la estructura de almacenamiento de determinados procesos; también se tomaron en cuenta las modificaciones legales por parte de entidades públicas orientadas al sector farmacéutico. Puede observar el proceso de normalización de tablas de la base de datos en el punto **1.7.2.2**.

Como resultado de los procesos de análisis y diseño, la estructura de la base de datos se orientó a las actividades de compras, ventas, clientes, proveedores, facturación, reportes y almacén. Además, también almacena información de módulos adicionales como lo son el registro de deudas, guías de remisión, tipo de cambio de moneda, gestión de series de comprobantes, gestión de caja y correcciones en ingreso o egresos.

En cuestión a reglas de negocio, se implementaron dos procedimientos almacenados para ventas, útiles a la hora de mostrar la información de estas con un formato adecuado para

los reportes que pueden ser generados a través del sistema; mismos formatos que son requeridos por entidades gubernamentales.

1.7.1.3 Análisis y Diseño de la Plataforma Web.

En esta fase se elaboraron distintos diagramas UML como lo son de secuencia, casos de uso, de componentes, además también otro tipo de diagramas como lo es el BPMN. También se diseñaron algunas de las interfaces del sistema, prototipos que mostraron cómo estas interactuarían unas con otras, los enlaces que tendría cada una, cómo se organizaría cada página, qué datos serian requeridos según para que módulos y qué mejoras se podrían implementar respecto a la versión anterior de la plataforma gracias a las nuevas tecnologías de las que se haría uso. Todo el detalle del proceso de análisis y diseño, al igual que los diagramas puede ser encontrado en el punto **2.3**.

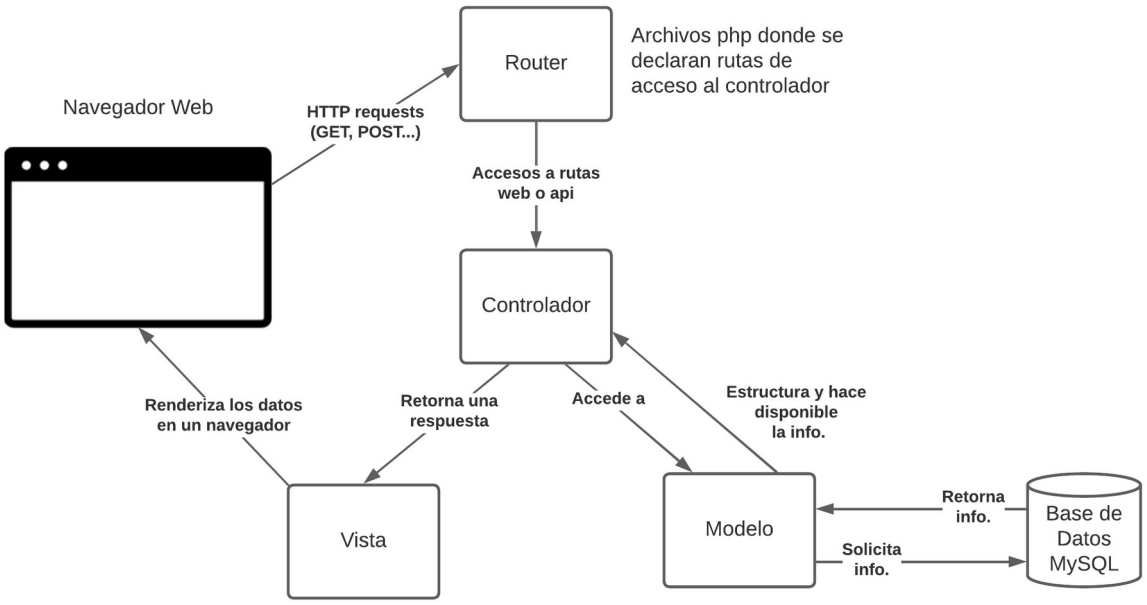
1.7.2 Codificación

1.7.2.1 Tecnologías y Programación del Software.

El proyecto se construyó sobre el *framework* Laravel 9 ya que este ofrece todos los recursos necesarios para desarrollar productos *software* a gran escala; además, es compatible con múltiples tecnologías de *frontend.* bases de datos, plantillas y estilos para interfaces, etc. Laravel utiliza como pilar central la arquitectura Modelo Vista Controlador (MVC), como se observa en **Figura 4** esta establece una división clara entre el modelo, la vista y el controlador que permite una organización estructurada y mantenible del código. La modularidad inherente al MVC agiliza la reutilización de componentes en diferentes partes del sistema y facilita la escalabilidad de este, también contribuyó con la metodología ágil elegida IWeb.

Figura 4

Interacción de la arquitectura MVC en el proyecto Laravel

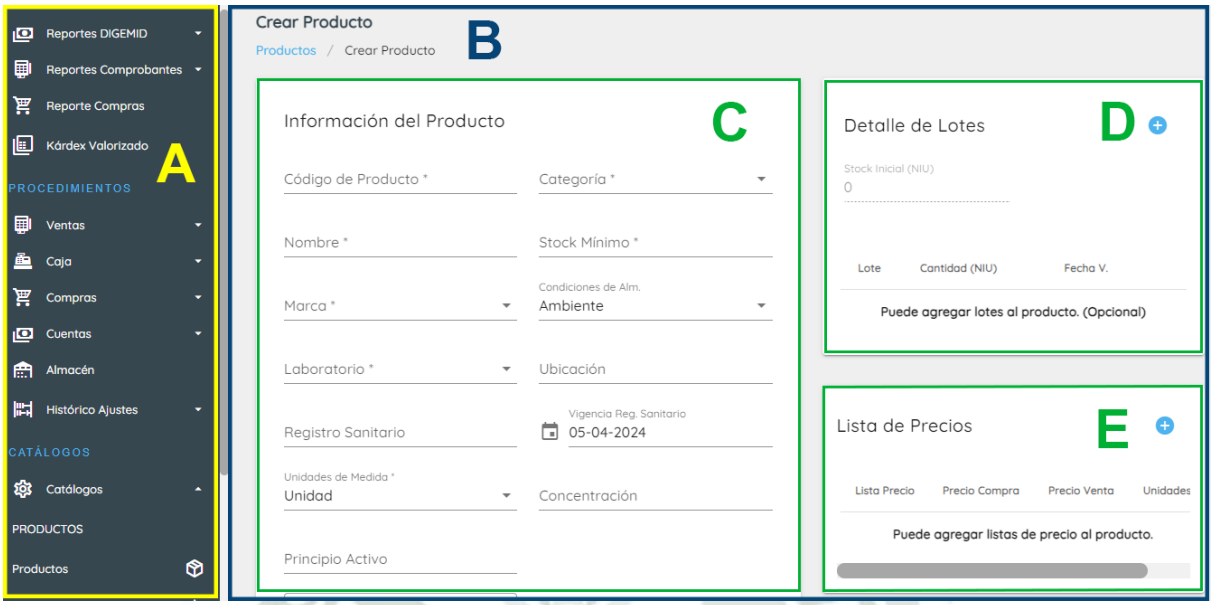


Nota. El proceso da inicio al momento en que el usuario realiza una petición *http* mediante el navegador web, estas pueden ser de los tipos GET, POST, PUT o DELETE y el acceso a ellas se determinará por el archivo de rutas definido y los permisos del usuario.

Para el desarrollo del *frontend* se hizo uso del *framework* VueJs, ya que brinda una forma de trabajo cómoda, al mismo tiempo, está especializado en la construcción de interfaces por componentes, es decir, ya que en las vistas de la plataforma contamos con un menú lateral, una barra superior y la ventana principal del módulo; podemos separar estas tres secciones principales en distintos componentes *.vue*, con el fin de evitar recargar toda la página y únicamente actualizar el componente que lo requiera. Cada uno de estos componentes es independiente, lo que significa que es posible modificarlos sin afectar al resto. En **Figura 5** se muestran los componentes que forman parte de la vista “Editar Producto” en el módulo Productos.

Figura 5

Distinción de los distintos componentes en la interfaz del sistema



Nota. Donde A es el menú lateral, B el contenedor dinámico de la interfaz de usuario, C la información del producto, D los lotes del producto y E la lista de precios de este.

Además de estas tecnologías, se optó por incorporar un sistema de roles y permisos para los usuarios, este sistema abarca tanto el *frontend* como el *backend* añadiendo restricciones a los usuarios que envían determinadas solicitudes; con esto obtenemos un control total sobre las capacidades que tiene cada usuario y agrega una capa adicional de seguridad al sistema.

Hablando de seguridad, es importante especificar que Laravel cubre completamente esta área, este *framework* permite que los desarrolladores definan las rutas de modo que requieran un token de sesión para acceder a ellas, también gestiona la autenticación de cada usuario mediante un archivo configurable, lo cual da mucha libertad al momento de elegir según qué métodos son convenientes en esta cuestión (Otwell, 2023). También, es posible hacer uso de *hashing* para cadenas de caracteres, específicamente, para las contraseñas de usuarios, el cual es un algoritmo de encriptación de una sola vía (Otwell, 2023). Existen características de seguridad adicionales que Laravel ofrece a los desarrolladores, sin embargo,

en este proyecto no se hace uso del resto de ellas, por este motivo no se mencionaron en el informe.

1.7.2.2 Normalización de Base de Datos.

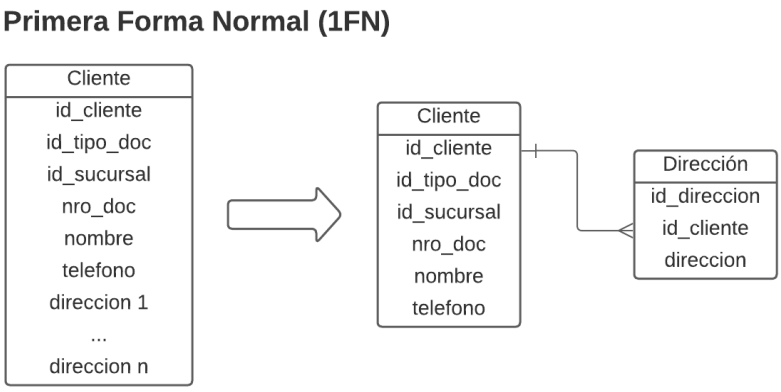
El diseño de la base de datos de un sistema integrado mayormente cuenta con una cantidad de tablas considerable que requieren ser normalizadas para que estas no generen inconsistencias en el tratamiento de la información. En este proyecto en específico, se hizo uso de las tres primeras formas normales debido a que muchas de las tablas almacenan datos de empresas, personas, productos, listas de precios, compras, ventas, etc. Este tipo de información implica que muchos de los datos tienden a ser redundantes por naturaleza ya que los procesos se encuentran interconectados. A continuación, se demuestra con un ejemplo práctico sobre algunas tablas del sistema el proceso de normalización de la base de datos.

1.7.2.2.1 Primera forma normal (1FN).

En la base de datos se guarda información de clientes tales como: tipo documento, sucursal, nombre, número de documento, teléfono y direcciones. La 1FN nos dice que todos los valores deben ser atómicos, los campos deben identificarse por la clave primaria y debe existir una independencia en el orden de los datos, es decir, no importará en qué orden se presenten los datos esto no cambiará su significado (Jaque, 2015). Siguiendo estas reglas para la tabla clientes, debido a que cada uno de ellos puede contar con múltiples direcciones se aplicó la 1FN como se muestra en **Figura 6**; este mismo principio fue aplicado a la tabla proveedores y sus direcciones. Otro caso similar pero un tanto más complejo sucede con la tabla productos, los cuáles pueden tener distintos precios y presentaciones por lo cual se crearon las tabla adicionales listas_precios y listas_precios_detalle; estas tablas contienen registros del precio de compra, precio de venta y unidad en la que se vende cada producto.

Figura 6

Primera forma normal de tabla clientes e información de dirección

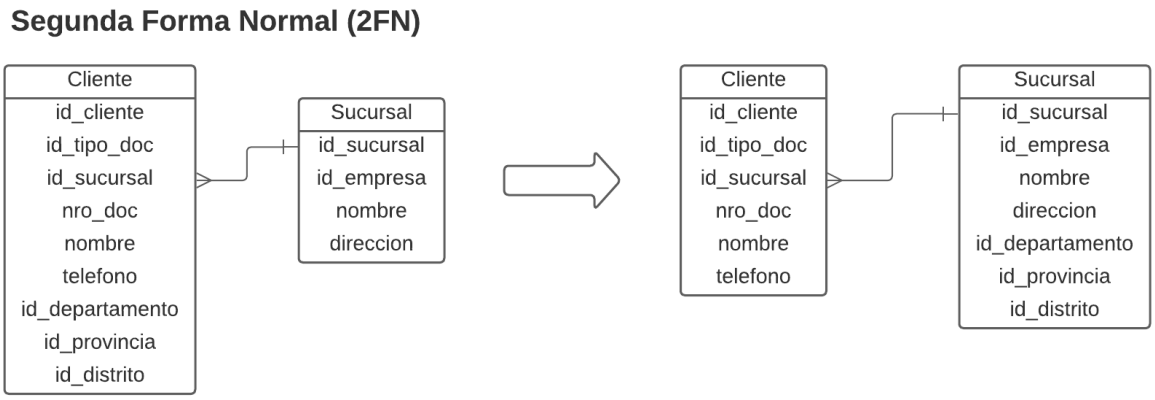


1.7.2.2.2 Segunda forma normal (2FN).

Para el registro de clientes en el sistema se toma en cuenta la sucursal a la que pertenecen, además del departamento, provincia y distrito de este. En un principio se consideraba que estos datos dependían de la tabla cliente, sin embargo, era conveniente aplicar la segunda forma normal sobre esta relación de tablas, debido a que la condición para la 2FN es que "los atributos no clave dependen de la clave principal" (Jaque, 2015). Por el motivo que una misma sucursal puede recibir varios clientes, teniendo estos la misma información de locación, al aplicar la 2FN tenemos como resultado que los datos de ubigeo dependen de la clave primar id_sucursal, como se observa en **Figura 7.**

Figura 7

Segunda forma normal de tabla clientes y sucursales

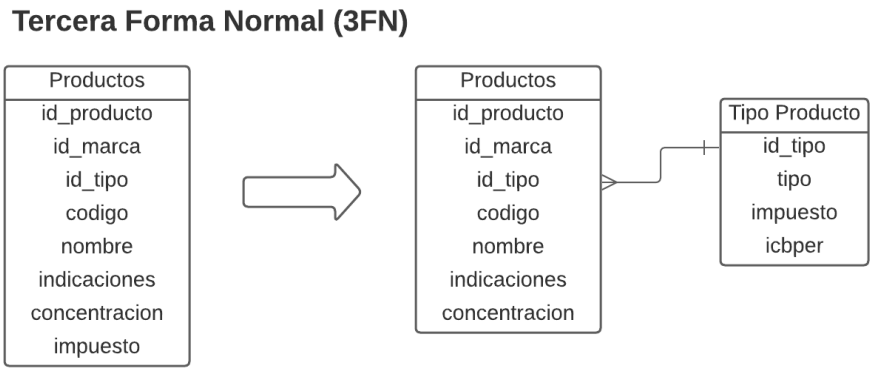


1.7.2.2.3 Tercera forma normal (3FN).

La tercera forma normal especifica que no debe existir ninguna relación de dependencia transitiva entre los atributos que no son clave, es decir, todos los atributos no clave deben aportar información sobre la clave primaria de la tabla (Jaque, 2015). En nuestra base de datos, el caso más claro para aplicar la 3FN fue en la tabla Productos y la columna "impuesto", ya que este atributo depende del "tipo de producto" y no del producto en sí. Por este motivo, esta columna se extrajo de la tabla Productos y se insertó en la tabla Tipo Productos, véase en **Figura 8**. Con esto podemos evitar inconsistencias en la información del impuesto de cada producto, ya que estos valores pueden cambiar con frecuencia.

Figura 8

Tercera forma normal de tabla productos y tipo



1.7.3 Documentación

Al tratarse de un proceso de desarrollo ágil, debido al uso de la metodología IWeb, la documentación del proyecto no fue rigurosa, más sí detallada en aquellos aspectos que se consideraron relevantes. Tomando como referencia buenas prácticas de la documentación de proyectos *software* desde diversas fuentes, se elaboró un documento con los puntos más notables para el proyecto, se listan a continuación las secciones de este:

1.7.3.1 Planificación del Proyecto.

En este subtítulo se documentaron la necesidad de la organización, la demanda, condiciones de mercado, los requisitos legales y las expectativas de los interesados. También

se elaboró el plan de trabajo, donde se listaron las tareas dentro del proyecto en conjunto con las fechas asignadas a cada una de ellas, esta información fue plasmada en un Diagrama Gantt (Project Management Institute, 2021; Universidad Nacional del Sur, 2017). El detalle técnico de este punto puede ser encontrado en **2.1**.

1.7.3.2 Definición y Especificación de Requerimientos.

En estos documentos se registraron los requerimientos funcionales y no funcionales del proyecto, se describieron a detalle y destacaron aspectos técnicos en aquellos que lo requerían. Se incluyó para cada uno de los requisitos su definición, descripción detallada y pruebas que se realizarían sobre la plataforma en determinado entorno para corroborar su cumplimiento (Project Management Institute, 2021; Universidad Nacional del Sur, 2017). Ambos documentos pueden ser visualizados de forma pública desde **Anexo 5** y **Anexo 6**.

1.7.3.3 Diseño y Arquitectura del Software.

Ya que el proyecto consistió en la construcción de un sistema integrado, implicó que este sea compuesto por distintas partes que interactúen entre sí. En esta sección se describieron cuáles serían esos módulos, el rol de cada uno de ellos dentro de la plataforma, su jerarquía y la forma en la que se organizaron e interconectaron para dar sentido a los procesos de negocio. Se describió a detalle cada módulo, el papel que cumpliría, su alcance, restricciones y requisitos; además, de ser necesario, se especificaron los servicios externos de los que este haría uso (Project Management Institute, 2021; Universidad Nacional del Sur, 2017).

1.7.3.4 Descripción de Servicios.

Aquí se listaron los servicios que el sistema ofrece, se describieron los procesos de tratamiento de la información y como es que estos son invocados desde la aplicación. Se incluyeron pseudo algoritmos enriquecidos de comentarios y descripciones de alto nivel del funcionamiento de cada servicio, esto con el fin de dar a entender de una forma simplificada

el propósito de cada uno de ellos (Project Management Institute, 2021; Universidad Nacional del Sur, 2017).

1.7.3.5 Mantenimiento y Soporte.

Aquí se especificaron las indicaciones más relevantes cuando se requiere hacer mantenimiento al sistema. Se describieron los pasos necesarios para la configuración del proyecto *software*, instrucciones para gestionar las versiones del proyecto mediante git y manuales de desarrollo que sirvan como guía para llevar un trabajo estandarizado. También se llevó un registro de las tareas necesarias para el mantenimiento de la plataforma, gestión de cambios, implementación de actualizaciones y posibles mejoras futuras (Project Management Institute, 2021; Universidad Nacional del Sur, 2017).

1.7.3.6 Anexos.

En esta sección se listaron todos los documentos, diagramas, manuales, links a páginas web y demás recursos necesarios para el sustento de la documentación del proyecto (Project Management Institute, 2021; Universidad Nacional del Sur, 2017).

1.7.4 Pruebas

Es preciso resaltar que estas pruebas se ejecutaron al mismo tiempo que se desarrollaba el proyecto, esto gracias a las iteraciones que establece la metodología IWeb. A continuación, se listan los distintos tipos de pruebas consideradas y ejecutadas a lo largo del desarrollo de este proyecto.

1.7.4.1 Pruebas Unitarias.

Fueron las primeras pruebas que se realizaron directamente sobre extractos de código fuente, estas se tuvieron como objetivo principal garantizar el correcto funcionamiento de cada una de las funciones del sistema, por más pequeñas o sencillas que fueran. Debido a la naturaleza de las pruebas unitarias, fue responsabilidad de los programadores escribir test automatizados y ejecutarlos conforme la aplicación se iba desarrollando e implementando. El

efecto de estas pruebas sobre el proyecto fue la detección temprana de errores, evitando así que estos escalaran y afectaran el funcionamiento esperado del sistema (Natalie Rodgers, 2022; Pittet, 2023). Los resultados de estas pruebas ser encontrados en **4.1.1**.

1.7.4.2 Pruebas de Integración.

Estas se utilizaron una vez concluyeron las pruebas unitarias y su principal objetivo fue comprobar que los módulos que integran el sistema funcionaran bien en conjunto (Natalie Rodgers, 2022; Pittet, 2023). Por ejemplo, se culminó con el desarrollo del módulo de compras de forma aislada, sin embargo, se requería generar compras a partir de órdenes de compra. Entonces se encontró que, al hacer este proceso, las cantidades de los productos en el detalle se importaban de forma incorrecta. Finalmente se halló que era un error de conversión de texto a número, por lo tanto, se requirió normalizar los datos de ambos módulos. Como producto de la utilización de las pruebas de regresión se obtuvo una colección de módulos integrados, funcionales entre sí y que garantizaron la consistencia de la información que se procesaba dentro del sistema. Los resultados de estas pruebas ser encontrados en **4.1.2**.

1.7.4.3 Pruebas de Seguridad.

Estas pruebas tuvieron el objetivo de detectar posibles vulnerabilidades en el sistema que pudieran afectar su funcionamiento normal, la integridad de los datos o la disponibilidad de la información (Natalie Rodgers, 2022). Respecto a las vulnerabilidades en sistemas hechos en Laravel, se estima que las más recurrentes son aquellas que dependen del programador, es decir, errores en la configuración de permisos, inclusión de credenciales en el código fuente, incluir el archivo de configuración "*.env*" en el repositorio compartido (Otwell, 2023). Se ejecutaron entonces las pruebas de seguridad verificando que los tipos de usuarios tenían distintos niveles de permisos dentro de la aplicación, que no era posible acceder a funciones internas siendo un agente externo al sistema y que credenciales de todo tipo son tratadas como variables de entorno. Los resultados de estas pruebas ser encontrados en **4.1.3**.

1.7.4.4 Pruebas de Rendimiento.

Fueron ejecutadas sobre la aplicación luego de la implementación de cada módulo, estas consistieron en registrar los tiempos de respuesta, consumo de recursos y estabilidad del sistema con cargas de trabajo determinadas que simulaban condiciones reales (Natalie Rodgers, 2022; Pittet, 2023). La principal conclusión a la que se llegó fue que los tiempos de ejecución son óptimos, menores a 1 segundo y en ocasiones específicas hasta 2 segundos, lo que permite a los usuarios disfrutar de un uso de la aplicación fluida. Los resultados de estas pruebas ser encontrados en **4.1.4**.

1.7.4.5 Pruebas de Usabilidad.

Se utilizaron para evaluar el grado de facilidad de uso de la plataforma por parte de los usuarios desde las perspectivas de atractivo visual, diseño intuitivo, coherencia de elementos visuales, acceso a la información y aporte del *chatbot* a la usabilidad del sistema (Natalie Rodgers, 2022). Estas pruebas fueron ejecutadas luego de culminar cada iteración del proceso de desarrollo, lo cual facilitó el descubrimiento de fallos en el diseño del sistema, oportunidades de mejora y determinar si el producto final es útil (Pursell, 2023). Estas pruebas arrojaron como resultado una serie de requerimientos adicionales que debían ser incluidos en el sistema para la conformidad de los usuarios finales.

1.7.5 Implementación

El despliegue del sistema integrado implicó una serie de pasos previos que sirvieron como preparación para que este proceso fuera exitoso. Como primer paso, durante la etapa de desarrollo del *software* se eligieron el dominio y *hosting* para la aplicación. Se consideraron muchas opciones con distintos rangos de precios, características técnicas variadas y enfocados a distintos tipos de negocios, la elección final consistió en un *web hosting* ofrecido por la compañía OVH ya que su precio es cómodo, las características técnicas se ajustaban al

negocio, cuenta con una buena reputación en soporte técnico y la localización de sus servidores era conveniente.

Una vez elegido el *hosting*, el siguiente paso fue desplegar la aplicación con las respectivas configuraciones, lo cual no tomó mucho tiempo como tal, sin embargo, el proveedor de *hosting* requirió de tres días para configurar y entregar un *web hosting* operativo. Es necesario recordar que los pasos descritos fueron ejecutados al mismo tiempo que se construía la aplicación, lo que quiere decir que los avances posteriores sobre el *software* fueron actualizados tanto en el repositorio Git como en el *web hosting* mediante el protocolo FTP al culminar cada iteración de la metodología IWeb.

Finalmente, una vez el proyecto fue concluido, se requirió brindar capacitación a los usuarios finales, en esta etapa se otorgó el manual de usuario a cada cliente y se programaron reuniones cortas de una hora a lo largo de cuatro días para dar las indicaciones necesarias sobre el uso de la aplicación; además, se estableció un canal de comunicación directo entre el cliente y la desarrolladora para la aclaración de cualquier duda posterior a la culminación del proyecto.

CAPÍTULO II

2 DOCUMENTACIÓN TÉCNICA

2.1 Plan de Proyecto Informático

2.1.1 Planificación Temporal del Proyecto

El presente proyecto tiene una estimación aproximada de nueve meses y medio siendo la fecha de inicio el 01/09/2022 y como fecha de finalización el 31/08/2023. La jornada de trabajo consta de 8 horas de lunes a viernes, contando con 245 días hábiles dentro de las fechas del proyecto y habiendo asignado dos programadores podemos calcular que las horas/hombre necesitadas para culminarlo son 980 horas. A continuación, en **Tabla 3** podemos observar el listado de actividades con la fecha asignada a cada una; igualmente en **Anexo 2** es posible visualizar el diagrama Gantt asociado.

Tabla 3

Cronograma de tareas relacionadas con el proyecto y fechas de ejecución

Cronograma de Tareas	Fecha de inicio	Fecha de fin
Etapa 1: Análisis	**01/09/22**	**20/09/22**
Estudio de viabilidad	01/09/22	06/09/22
Definición de requisitos	07/09/22	12/09/22
Modelado funcional	13/09/22	15/09/22
Modelado estructural	16/09/22	20/09/22
Etapa 2: Diseño	**21/09/22**	**18/10/22**
Diseño de arquitectura	21/09/22	27/09/22
Diseño de datos	28/09/22	04/10/22
Diseño de interfaces	05/10/22	11/10/22
Diseño de base de datos	12/10/22	18/10/22
Etapa 3: Desarrollo	**19/10/22**	**30/05/23**
Creación y configuración del proyecto	19/10/22	25/10/22
Integración de seguridad de acceso y permisos	26/10/22	01/11/22
Desarrollo de módulos de tablas maestras	02/11/22	13/12/22
Desarrollo de módulo de compras	14/12/22	27/12/22
Desarrollo de módulo de ventas	28/12/22	24/01/23
Desarrollo de módulo de deudas	25/01/23	21/02/23

Cronograma de Tareas	Fecha de inicio	Fecha de fin
Desarrollo de módulo de inventario	22/02/23	14/03/23
Funciones de facturación	15/03/23	28/03/23
Funciones de *reporting*	29/03/23	18/04/23
Corrección de Errores	19/04/23	09/05/23
Implementación de mejoras	10/05/23	30/05/23
Etapa 4: Implementación	**31/05/23**	**20/06/23**
Capacitación de usuario	31/05/23	06/06/23
Despliegue de sistema	07/06/23	20/06/23
Etapa 5: Pruebas	**21/06/23**	**21/07/23**
Documentación del proceso de pruebas	21/06/23	27/06/23
Prueba de integración	28/06/23	30/06/23
Prueba de sistema	03/07/23	05/07/23
Prueba de interfaz	06/07/23	10/07/23
Prueba de seguridad	11/07/23	13/07/23
Prueba de rendimiento	14/07/23	18/07/23
Prueba de usabilidad	19/07/23	21/07/23
Etapa 6: Integración del módulo de *chatbot*	**24/07/23**	**31/08/23**
Planificación de la actualización y estimación de costos	24/07/23	28/07/23
Actualización de la versión de Laravel	31/07/23	03/08/23
Desarrollo del algoritmo	04/08/23	24/08/23
Pruebas en producción	25/08/23	31/08/23

Nota. Elaboración propia.

2.1.2 Estudio de Viabilidad del Proyecto

2.1.2.1 Factibilidad Técnica.

Se centró en evaluar las distintas opciones tecnológicas disponibles para este proyecto, en este caso, el equipo de desarrollo en ese entonces contaba con experiencia en Laravel, pero también se incluyeron tecnologías nuevas para ellos con el fin de agilizar el proceso de desarrollo de *software* y mejorar la calidad del producto final para el usuario. Puede visualizar el resultado de este análisis en **Tabla 4**,

adicionalmente, **s**i desea conocer más sobre el *software* utilizado en este proyecto puede consultar el punto **1.7.2.1**.

Tabla 4

Recursos hardware y software empleados en el proyecto

Recurso	Descripción
Software	
Laravel 9.x	Es un *framework* de *backend* fundamental para el proyecto, ya que conforma la base de este. Gestiona librerías, dependencias, interfaces de usuario, lógica interna y comunicación con la base de datos mientras mantiene la estructura MVC a lo largo del proyecto de *software* (Otwell, 2023).
Navegador Web (Chromium)	Para el proceso de desarrollo del sistema de información es necesario utilizar un navegador web, en esta ocasión, la elección será cualquier navegador comercial basado en Chromium, ya que es la arquitectura más popular.
Visual Studio Code	Editor de código fuente popular y compatible con Windows, Linux y macOS. La distribución base tiene soporte para Node.js, JavaScript y TypeScript, pero también es flexible por lo cual tiene extensiones para otros lenguajes de programación (Microsoft, 2022).
Servidor Apache y MySQL local	Tecnologías utilizadas como servidor local para el alojamiento del proyecto, cuenta con herramientas para alojar la base de datos y la aplicación en un ambiente local. Aquí veremos de forma inmediata los cambios que se realicen sobre el código del proyecto.
Github	La plataforma más popular de gestión de repositorios basada en *git* para el control de versiones de proyectos. A medida que el sistema de información sea desarrollado, aquí se subirán los avances gradualmente y además se podrán asignar las correcciones pendientes. Puede consultar en **Anexo 4**.
Hardware	
Equipo de computación	Ordenador de sobremesa o portátil el cual tendrá instalado el *software* necesario para el desarrollo del proyecto, conteniendo las versiones de *software* específicas.

Recurso	Descripción
Servidor de despliegue	Servidores habilitados para poder soportar las tecnologías incorporadas en el sistema de información, deberá estar disponible las 24 horas además de tener servidores de respaldo para conservar la disponibilidad del servicio.

Nota. Elaboración propia.

2.1.2.2 Factibilidad Económica.

Se elaboró un presupuesto para el proyecto listando los recursos económicos, en esta ocasión, estuvieron relacionados principalmente con los equipos personales de cómputo y el alquiler de servidores para poner en producción el sistema web; también se tomó en cuenta el total de los gastos en servicios como luz, *hosting*, salarios de los trabajadores y servicios externos calculados en el plazo de un año. En **Tabla 5** puede observar el detalle de los recursos económicos necesarios para el proyecto.

Tabla 5

Recursos económicos empleados en el proyecto

Recurso	Cantidad	Precio Unitario ($)	Precio Total ($)
Laptop	2	600	1200
Servicio de luz oficinas por mes	12	28	336
Propiedad del dominio	1	40	40
OVHCloud Web Hosting Professional	1	92	92
Abono a saldo de OpenAI API	12	5	60
Salario de programadores (x2)	24	280	6720
Salario de *Project Manager*	12	600	7200
		Total	**15648**

Nota. Elaboración propia.

2.1.2.3 Factibilidad Legal.

Se evaluaron las actualizaciones a las regulaciones de farmacias, especialmente aquellas que afectaban el proceso de facturación, algunos formatos de reportes y registro de productos. Cabe resaltar que, al tratarse de un sistema de gestión interno y no de un

ecommerce, muchas leyes que aplicarían a negocios electrónicos no se tomaron en cuenta para este proyecto, sin embargo, las que sí se consideraron fueron las leyes 29733 sobre la protección de datos personales (Ley de Protección de Datos Personales N° 29733, 2013) y la 1033 sobre la protección de propiedad intelectual y derechos de autor, específicamente en este punto existen términos relacionados con el desarrollo de *software*, la gestión de la posesión del código fuente donde se evalúa la cesión de uso al cliente y si la entrega del trabajo será total o por lo contrario el producto se entregará como un compilado (Ley de Organización y Funciones Del Instituto Nacional de Defensa de La Competencia y de La Protección de La Propiedad Intelectual, 2008).

2.1.2.4 Factibilidad Operacional.

Tomamos en cuenta los perfiles de cada involucrado en el proyecto, la estrategia que se tomaría para el avance, la calidad de la definición de objetivos, los requerimientos operativos, estrategias y procedimientos para identificar distintos aspectos de la factibilidad operativa del proyecto. A continuación, en **Tabla 6** puede observar a los interesados en el proyecto junto a la descripción de cada uno de ellos. A su vez, en **Tabla 7** se observa la descripción de la estrategias y procedimientos tomados para el proceso de avance del proyecto

Tabla 6

Interesados del proyecto

Interesado	Expectativas	Grado de influencia	Grado de interés
Gerente de proyecto	Es el líder y responsable principal de asegurar que el proyecto se complete con éxito, dentro del alcance definido, dentro del presupuesto asignado y en el plazo establecido.	Alto	Alto
Equipo de desarrollo	Está compuesto por los programadores del proyecto, cuentan con los conocimientos técnicos, coordinan en conjunto con el	Alto	Alto

Interesado	Expectativas	Grado de influencia	Grado de interés
	gerente del proyecto los próximos avances y están encargados de llevar los requerimientos a funcionalidades en el sistema.		
Propietario de cadena de farmacias	Tienen un interés directo en el éxito del proyecto y en la mejora de la gestión de los procesos de negocio de la cadena de farmacias.	Alto	Alto
Departamento operaciones y logística	Están interesados en optimizar los procesos de abastecimiento, inventario y distribución de productos farmacéuticos.	Medio	Medio
Personal de farmacias	Cumplirán el rol de usuarios finales del sistema, estarán interesados en contar con una herramienta que facilite sus tareas diarias además de que el producto final sea intuitivo y simple.	Medio	Alto
Proveedores	Los proveedores de productos farmacéuticos estarán interesados en una cadena de suministro eficiente, sin inconsistencias en el registro de las facturas y los movimientos en almacén.	Medio	Bajo
Cliente de farmacias	Los clientes finales de las farmacias se beneficiarán indirectamente del proyecto, ya que el sistema integrado mejorará la atención al cliente y almacenará sus datos personales, por lo que estarán especialmente interesados en la seguridad de estos.	Medio	Bajo
Entidades reguladoras	Las autoridades de regulación y supervisión en el ámbito farmacéutico están interesadas en garantizar que el sistema cumpla con las regulaciones y normativas pertinentes. Además, entidades tributarias están	Alto	Bajo

Interesado	Expectativas	Grado de influencia	Grado de interés
	interesadas en la correcta declaración de comprobantes.		

Nota. Elaboración propia.

Tabla 7

Estrategias y procedimientos

Estrategia/Procedimiento	Aplicación en el proyecto
Identificación de interesados en el proyecto	Identificar y listar a los *stakeholders* del proyecto, registrar sus expectativas grado de influencia e interés con el fin de tener una perspectiva sobre el proyecto desde todos los ángulos.
Análisis de influencia e interés de *stakeholders*	Con el fin de agrupar los *stakeholders* se hizo uso de entrevistas, reuniones y análisis de documentación para identificar a los interesados clave.
Análisis de requerimientos operativos	Identificar y definir claramente los requerimientos y necesidades operativas del proyecto, incluyendo recursos humanos, tecnológicos, financieros y de infraestructura.
Análisis de riesgos operacionales	Identificar y evaluar los riesgos operacionales que podrían afectar la ejecución del proyecto y desarrollar planes de contingencia para mitigarlos.
Evaluación de costos operativos	Calcular los costos operativos estimados para la ejecución del proyecto, incluyendo gastos de personal, insumos, mantenimiento, entre otros.
Evaluación de normativas y regulaciones	Verificar que el proyecto cumpla con todas las normativas y regulaciones aplicables en su industria o sector.

Nota. Elaboración propia.

2.2 Especificación de Requisitos del Proyecto de TICs

Para la tarea de documentar los requisitos del proyecto se hizo uso de ingeniería de requisitos, donde se completaron las fases de captura y análisis de requisitos; luego de la culminación de estas actividades se estructuró el documento “Especificación de Requisitos”

donde se incluyó toda la información necesaria para el diseño y posterior verificación del producto final en relación con la especificación. Para visualizar el documento completo y detallado puede dirigirse a **Anexo 5**, ya que en este punto sólo haremos un resumen de lo que aquel informe contiene.

La intención con la que se elaboró el documento "Especificación de Requisitos" fue porque se requería un informe de referencia disponible tanto para interesados, equipo de desarrollo y clientes finales. Este tiene la capacidad de establecer una base clara y compartida de comprensión respecto a las características, funcionalidades y limitaciones del sistema. Esto también garantiza que todas las partes involucradas tengan una visión común y precisa de lo que se espera lograr con el producto final, lo que a su vez facilita la toma de decisiones informadas y asegura la alineación con los objetivos y necesidades del proyecto (Visure, 2023).

En primer lugar, se especificaron los requisitos mínimos y recomendados del sistema para seguir con el proceso de desarrollo y ejecución de una plataforma basada en Laravel 9. Esto hace referencia a las características de *hardware* que debería tener el servidor donde se alojará la aplicación, tanto en un entorno local como en uno en la nube al pasar a la etapa de producción. También se registraron los requisitos funcionales y no funcionales, cada uno con un código de identificación y respectiva descripción. Puede observar todos estos detalles especificados en las tablas a continuación.

Tabla 8

Requisitos mínimos del sistema

Requisitos mínimos	
Procesador	Procesador de doble núcleo a 2.5 GHz o superior
Memoria RAM	4 GB de RAM
Espacio en Disco	Al menos 3 GB de espacio libre en disco duro
Conexión a Internet	Acceso a Internet de banda ancha para descargar librerías y recursos

Nota. Elaboración propia.

Tabla 9

Requisitos recomendados del sistema

Requisitos recomendados	
Procesador	Procesador de cuatro núcleos a 2.8 GHz o superior
Memoria RAM	8 GB de RAM o más
Espacio en Disco	5 GB de espacio libre en disco duro o SSD
Conexión a Internet	Conexión de banda ancha con alta velocidad de descarga y carga

Nota. Elaboración propia.

Tabla 10

Requisitos funcionales

Código	**Descripción**
Registro y Gestión de Productos	
RF01	El sistema permitirá a los usuarios registrar nuevos productos en la base de datos, ingresando detalles como código, nombre, descripción, categoría, laboratorio, marca, etc.
RF02	Los usuarios podrán actualizar la información de los productos existentes, incluyendo cambios en el código, nombre, descripción, categoría, laboratorio, marca, etc.
RF03	Cada producto contará con listas de precio, esto con el fin de gestionar el precio unitario en las diferentes unidades de medida de sus presentaciones (caja, blíster, unidad).
RF04	También podremos abastecer un producto al momento de su creación con una opción para crear lotes, lo cual generará un movimiento de entrada en almacén.
Compras y Gestión de Inventario	
RF05	Se podrán emitir cotizaciones de compras, también estarán disponibles en formato PDF.
RF06	Las órdenes de compra podrán ser generadas independientemente o a partir de una cotización previa y también podrán ser visualizadas en formato PDF.
RF07	Las compras podrán ser generadas independientemente o a partir de una orden de compra previa, generarán un movimiento en almacén, actualizarán el stock del producto/s, se registrarán nuevos lotes de compra y podrán ser enviadas por *email* en formato PDF.
RF08	Los registros de compra podrán ser anulados luego de su creación, esto generará

Código	Descripción
	un movimiento de devolución con las cantidades de cada producto que fueron previamente descontados de almacén.
RF09	Los usuarios podrán visualizar el nivel de inventario actualizado y recibirán notificaciones cuando los niveles estén por debajo del stock mínimo.
Ventas y Facturación	
RF10	Los usuarios podrán registrar ventas seleccionando el tipo de comprobante, cliente, moneda, condición de pago, productos, unidades de medidas y cantidades respectivas.
RF11	Luego de realizar la venta el sistema enviará los datos del comprobante al servicio de facturación electrónica de SUNAT, luego de este proceso, se actualizará el estado del comprobante (aprobado, en espera, rechazado).
RF12	En caso el proceso de facturación electrónica del comprobante haya fallado, el botón para enviar a SUNAT siempre estará disponible desde la pantalla de visualización de este.
RF13	A partir de un comprobante creado, se podrá asociar una nota de crédito la cuál será se visualizada desde vista dentro del menú ventas.
RF14	A partir de un comprobante creado, se podrá generar una guía de remisión para acreditar el traslado de los productos tomando información como la fecha, el motivo del traslado, peso total, direcciones de partida y llegada, datos del transportista y productos.
Generación de Reportes	
RF15	El sistema permitirá a los usuarios generar informes detallados sobre las ventas realizadas en un período específico, a un cliente en concreto. También cumplirán con el formato 14.1 establecido por la SUNAT.
RF16	El sistema permitirá a los usuarios generar informes sobre los lotes de productos mostrando el stock, cantidad de salidas, fecha de vencimiento y el producto al que hacen referencia.
RF17	El sistema permitirá a los usuarios generar informes para DIGEMID, en “Observatorio de Productos” se listará cada producto con el precio por empaque y precio unitario.
RF18	El sistema permitirá a los usuarios generar informes para DIGEMID, en “Listado de Precios” se listará cada producto con su precio mínimo, precio

Código	Descripción
	máximo y la mediana de acuerdo con el formato requerido.
Deudas por Cobrar y Pagar	
RF19	Al generar una deuda en una venta, esta creará una deuda asociada tanto a al comprobante como al cliente.
RF20	Cada cliente contará con una cuenta corriente desde la cual se podrá acceder a todas sus deudas, se visualizará el estado de estas (deuda pendiente, deuda saldada) y se podrán registrar los pagos.
RF21	Al generar una deuda en una compra, esta creará una deuda asociada tanto a la compra como al proveedor.
RF22	Cada proveedor contará con una cuenta corriente desde la cual se podrá acceder a todas sus deudas, se visualizará el estado de estas (deuda pendiente, deuda saldada) y se podrán registrar los pagos.
Apertura y Cierre de Caja	
RF23	El sistema controlará la apertura de caja antes de realizar cualquier operación donde exista intercambio monetario.
RF24	La caja será abierta o cerrada manualmente, sin embargo, el sistema deberá cerrar esta automáticamente al terminar el día.
RF25	En la lista de cajas será posible visualizar la fecha, hora, monto de apertura, monto de cierre y la división de estos por medio de pago (efectivo, tarjeta y depósito).
Seguridad y Autenticación	
RF26	Los usuarios deberán autenticarse mediante un nombre de usuario y una contraseña para acceder al sistema.
RF27	El sistema implementará niveles de acceso, otorgando diferentes permisos a diferentes roles de usuario, como trabajador, administrador y superadministrador.
RF28	Todos los niveles de acceso y permisos serán creados, modificados o deshabilitados desde un módulo propio.
Interfaz de Usuario	
RF29	La interfaz de usuario será intuitiva y fácil de usar, con un diseño amigable que permita a los usuarios navegar y utilizar las funcionalidades sin dificultad.
RF30	Contará con un menú lateral donde se encontrarán todas las opciones de los

Código	Descripción
	módulos del sistema, serán agrupados de forma lógica de ser necesario y los nombres de cada uno serán cortos y precisos.

Nota. Elaboración propia.

Tabla 11

Requisitos no funcionales

Código	Descripción
Usabilidad	
RNF01	La interfaz de usuario debe ser intuitiva y de fácil navegación, permitiendo a los usuarios realizar tareas sin requerir capacitación excesiva.
RNF02	El tiempo de respuesta del sistema para realizar acciones y generar informes no deberá comprometer la fluidez del sistema.
RNF03	Los módulos del sistema deberán ser estructurados de forma coherente, los botones para aceptar, cancelar, eliminar o buscar deben tener los mismos colores, íconos y ubicación a lo largo de la plataforma.
Rendimiento	
RNF04	El sistema debe ser capaz de manejar simultáneamente 30 usuarios realizando ventas y consultas sin degradación significativa del rendimiento.
RNF05	Las tecnologías en uso no deben perjudicar el rendimiento del sistema, se tendrán que seleccionar librerías que no incorporen animaciones o complementos costosos a nivel de recursos.
Seguridad	
RNF06	Los datos de los usuarios, ventas, inventario y facturación estarán protegidos mediante cifrado y medidas de seguridad para evitar accesos no autorizados.
RNF07	Las contraseñas de los usuarios se almacenarán de manera segura utilizando técnicas de *hash*.
Disponibilidad	
RNF08	El sistema estará disponible para su uso las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad programado de no más de 2 horas al mes en caso se requiera una actualización mayor.
Escalabilidad	
RNF09	El sistema debe ser escalable, capaz de manejar un aumento en el número de usuarios, productos, compras o ventas sin una degradación significativa del

Código	Descripción
	rendimiento.
Documentación	
RNF10	Se proporcionará una documentación completa que describa el despliegue, configuración y operación del sistema, así como los procedimientos de solución de problemas.
RNF11	El código fuente del sistema estará bien estructurado y documentado, facilitando su mantenimiento y futuras actualizaciones.

Nota. Elaboración propia.

2.3 Especificación de Diseño

El principal objetivo de la redacción de este informe fue proporcionar una visión detallada y técnica del diseño del sistema de farmacias, estableciendo las bases para su implementación y brindando un enfoque profundo en la arquitectura, componentes y funcionalidades del sistema. La estructura inicial de este documento consiste en una breve introducción, los objetivos del proyecto, el cronograma, la arquitectura del sistema y el diseño de la base de datos. Los siguientes puntos se centraron en el diseño de las interfaces de usuario, el diseño de componentes y consideraciones en la implementación del proyecto.

A continuación, mencionaremos los puntos más relevantes de este documento, muchos de ellos se encuentran especificados a lo largo de este informe, sin embargo, existen consideraciones adicionales. Al igual que en el anterior subtítulo, podrá encontrar el documento de “Especificación de Diseño” completo en **Anexo 6**.

2.3.1 Interfaces de Usuario

2.3.1.1 Diseño Responsivo.

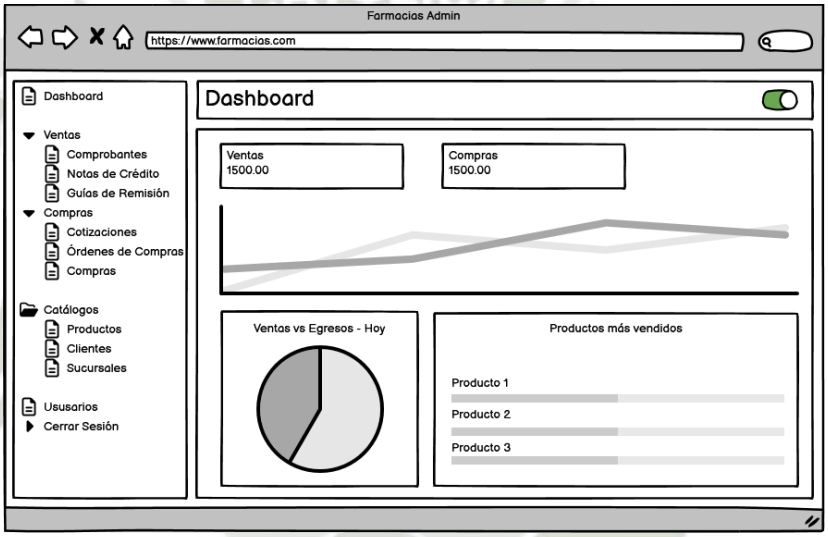
Todas las interfaces se han diseñado teniendo en cuenta la adaptabilidad y la usabilidad en dispositivos de sobremesa como computadoras de escritorio o laptops. Esto garantiza que los usuarios puedan acceder y utilizar el sistema de manera eficiente desde los dispositivos para los que este estará adaptado.

2.3.1.2 Interfaz de Dashboard.

El dashboard es la página de inicio del sistema y proporciona una visión general de las operaciones y estadísticas clave. Su diseño se centra en la presentación clara y concisa de información relevante, incluyendo gráficos de ventas por mes, resumen de los productos más vendidos y ventas vs compras del día. Es la primera página que los usuarios verán al ingresar al sistema y les brindará un resumen inmediato de las transacciones del negocio.

Figura 9

Diseño de interfaz del dashboard

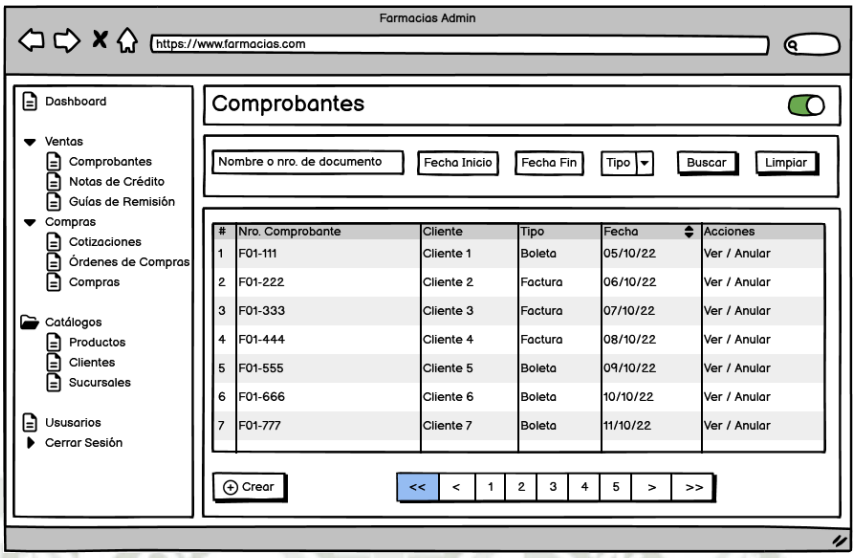


2.3.1.3 Interfaz Gestión de Ventas.

La interfaz de gestión de ventas permite a los usuarios registrar, buscar y gestionar las ventas realizadas en la farmacia. Se sigue el enfoque de CRUD (crear, leer, actualizar, eliminar) para brindar un flujo de trabajo completo. Los usuarios pueden agregar nuevas ventas, ver detalles de ventas existentes, actualizar determinada información y anular comprobantes si es necesario.

Figura 10

Diseño de interfaz de gestión de ventas, listado de comprobantes

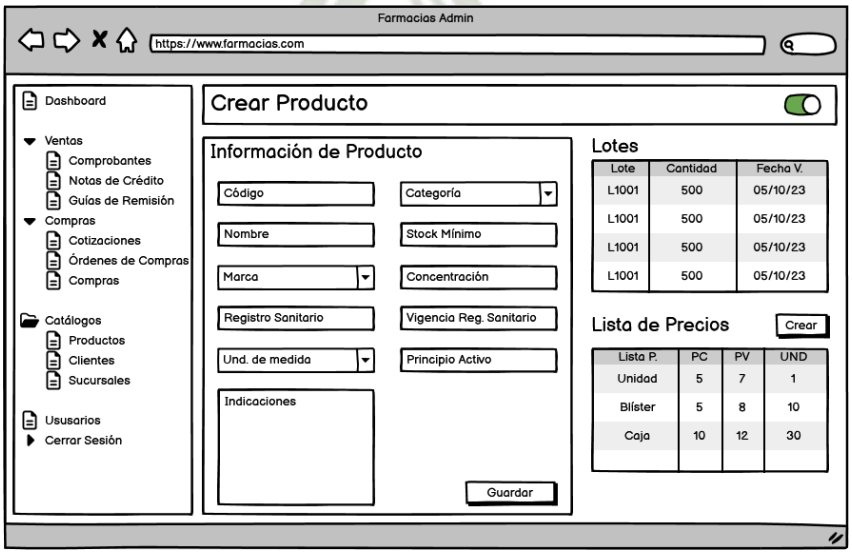


2.3.1.4 Interfaz Creación de Productos.

La interfaz de creación de productos permite a los usuarios agregar nuevos registros al inventario de la farmacia. Incluye campos para ingresar detalles como nombre, código, categoría, registro sanitario y stock mínimo; adicionalmente se incluyen la visualización los lotes de compra y las listas de precio por unidad. Se ha diseñado para ser intuitiva y eficaz, facilitando la incorporación de nuevos productos al sistema.

Figura 11

Diseño de interfaz de creación de producto



2.3.2 Diagrama de Componentes

2.3.2.1 Componentes.

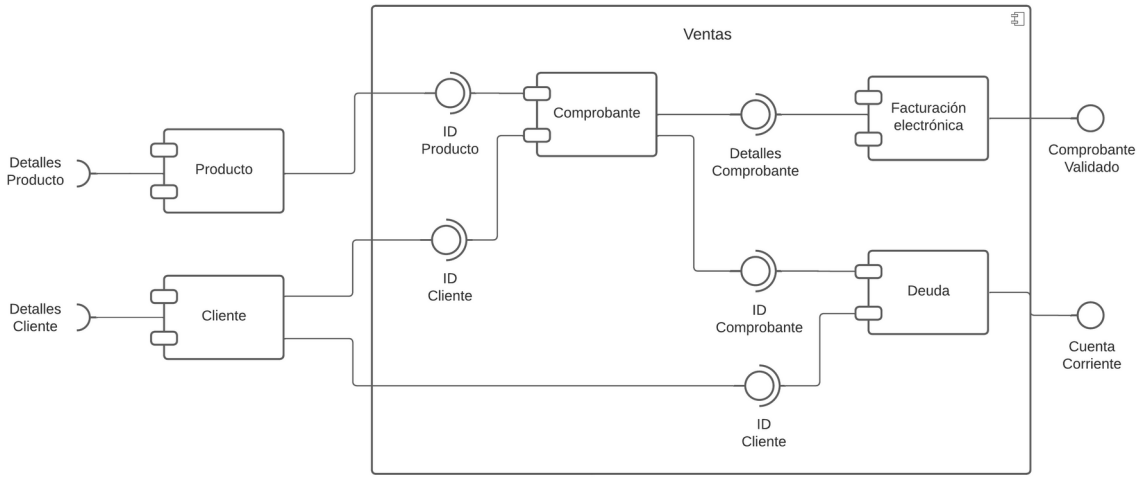
1. **Gestión de Productos.** Permitirá a los usuarios agregar, editar y eliminar productos del inventario de la farmacia. Incluirá funcionalidades como la especificación de nombre, código, categoría, y lista de precios y especificaciones de cada producto.
2. **Gestión de Clientes.** Permitirá a los usuarios registrar información sobre los clientes que se encuentren en procesos de ventas o generen deudas. Se incluirán detalles como nombre, tipo de documento, correo electrónico, direcciones y datos de contacto para facilitar futuras transacciones.
3. **Gestión de Deudas.** Aquí se registrarán todas las deudas por cobrar y pagar de la farmacia, permitirá a los usuarios agregar pagos para saldar deudas y ver el detalle de estos. Incluirá campos como la relación con la deuda, relación con la venta/compra, fecha de pago, monto y observaciones. Es útil para la gestión de deudas de clientes y deudas con proveedores.
4. **Gestión de Ventas.** Será un componente central del sistema, que permitirá a los usuarios registrar ventas, calcular total de compras y gestionar pagos. También se integrará con la gestión de inventario para mantener un registro en tiempo real de las existencias y con la gestión de deudas donde se listarán los pagos pendientes y saldados por parte de clientes.
5. **Facturación Electrónica.** Será un componente crítico para generar comprobantes de venta electrónicos válidos y legales ante SUNAT. Se integrará con la gestión de ventas para automatizar este proceso, se enviarán los campos requeridos por la entidad tributaria mediante un servicio API y se validará el comprobante asegurando así el cumplimiento de las regulaciones fiscales.

2.3.2.2 Diagrama.

Los componentes mencionados anteriormente interactuarán entre sí para ofrecer una experiencia completa y coherente al usuario, en **Figura 12** a continuación se muestra el diagrama de componentes, especificando cada uno y mostrando la relación entre ellos.

Figura 12

Diagrama de componentes del proceso de ventas



Nota. Cada componente se ha diseñado para cumplir con requisitos específicos y se integrará de manera eficiente para brindar una solución coherente y eficaz.

2.3.3 Diagrama de Casos de Uso

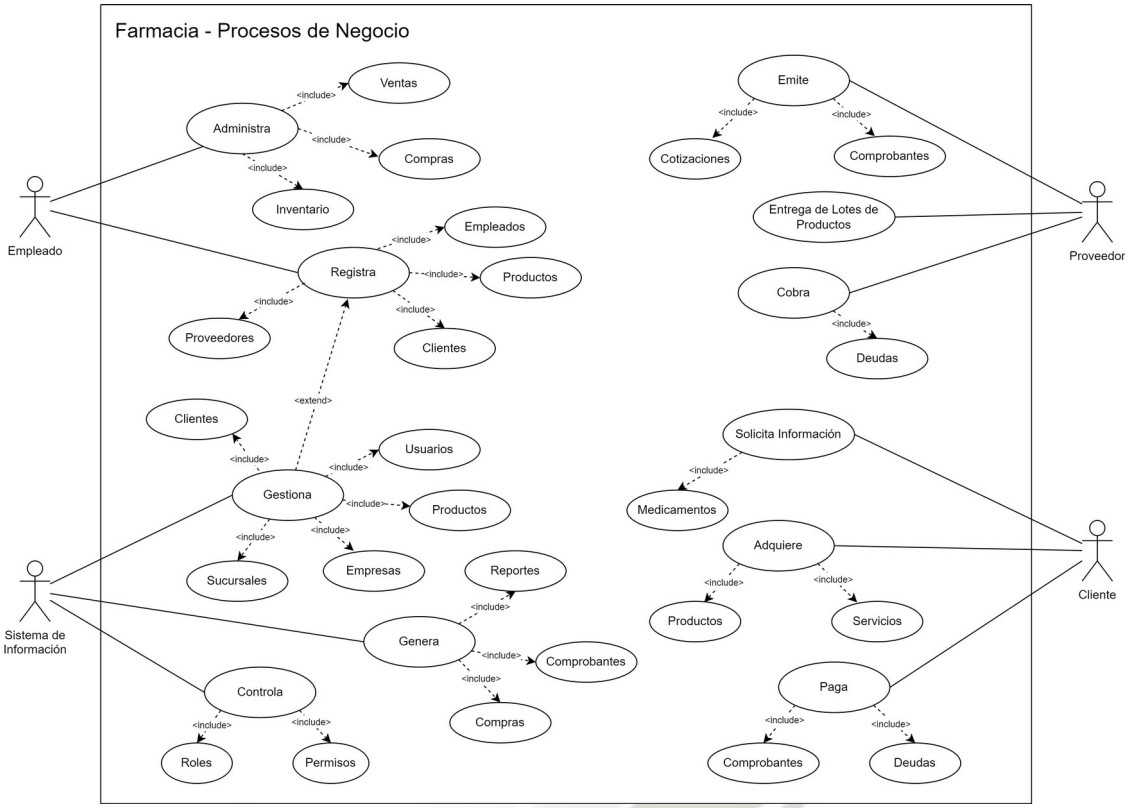
Para el enfoque del diagrama de Casos de Uso se identifican cuatro actores principales:

- **Empleado.** Personal que desempeña actividades de atención al cliente, gestión de ventas, compras, administración inventario además de registrar información sobre clientes, usuarios, proveedores y productos.
- **Proveedor.** Empresa que suministra medicamentos y diversos productos por lotes a la farmacia. Emite cotizaciones, comprobantes y realiza el cobro de deudas.
- **Cliente.** Persona o empresa que solicita información, adquiere productos o servicios en la farmacia y paga por ellos al contado o al crédito.

- **Sistema de información.** *Software* principal de optimización de procesos, gestiona información de clientes, usuarios, empresas, sucursales. Genera reportes de compras, y ventas. Controla el acceso a la aplicación mediante roles y permisos.

Figura 13

Diagrama de casos de uso



2.3.4 Diagrama de Secuencia

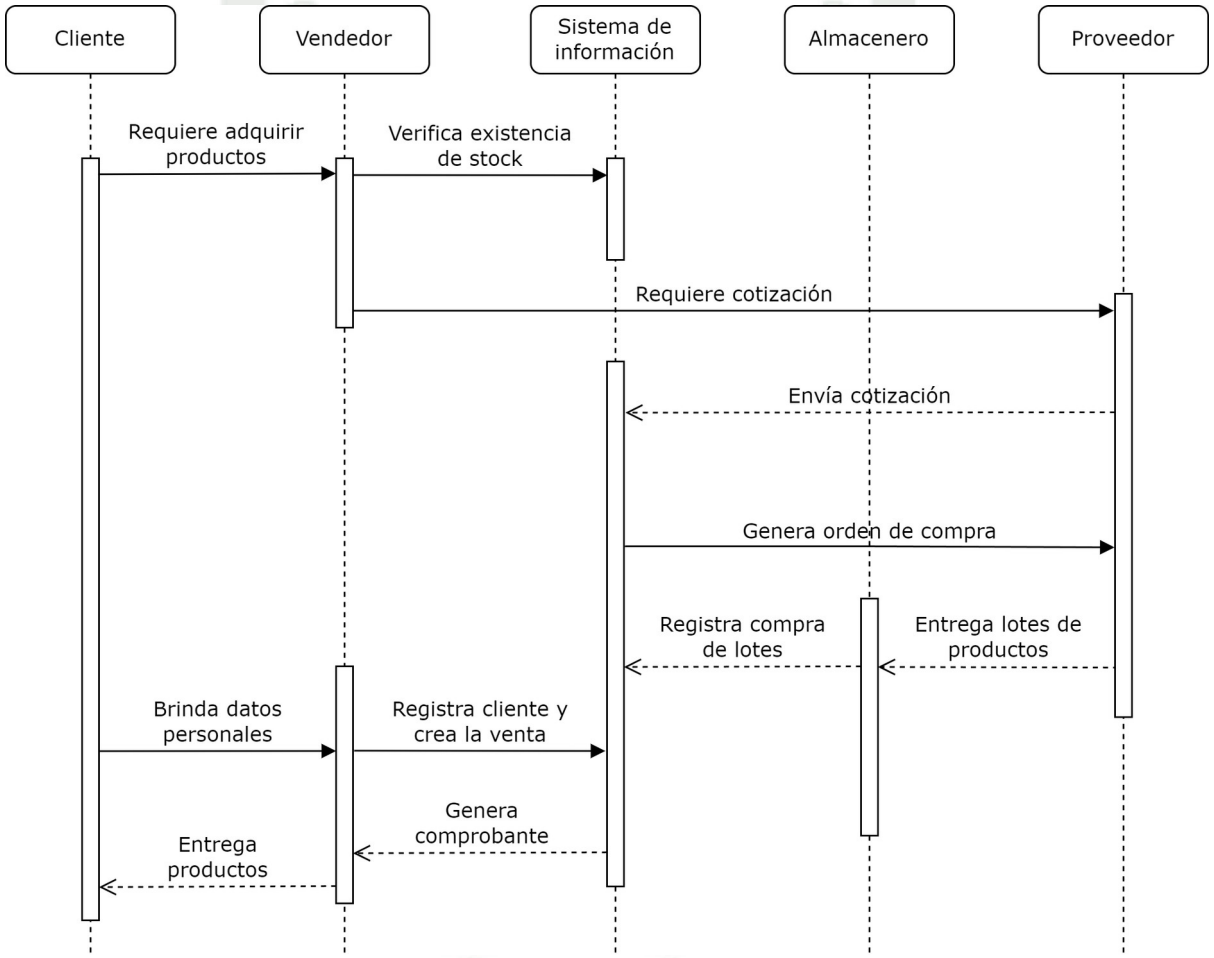
El diagrama de secuencia que se muestra a continuación describe el flujo de trabajo para los procesos de compra y venta en una farmacia.

1. El proceso inicia cuando el cliente ingresa a la farmacia y requiere adquirir productos.
2. El vendedor verifica la disponibilidad de estos en el sistema de información.
3. Dependiendo de la existencia o no del producto, el vendedor requerirá una cotización por parte del proveedor.
4. Se generará una orden de compra a partir de la cotización de compra.

5. El proveedor entregará los lotes de productos, se agregarán al almacén.
6. El vendedor generará el comprobante de venta tomando los datos personales del cliente.
7. Los productos son entregados al cliente y la venta es completada.

Figura 14

Diagrama de secuencia proceso de compra y venta



2.3.5 Diagrama BPMN

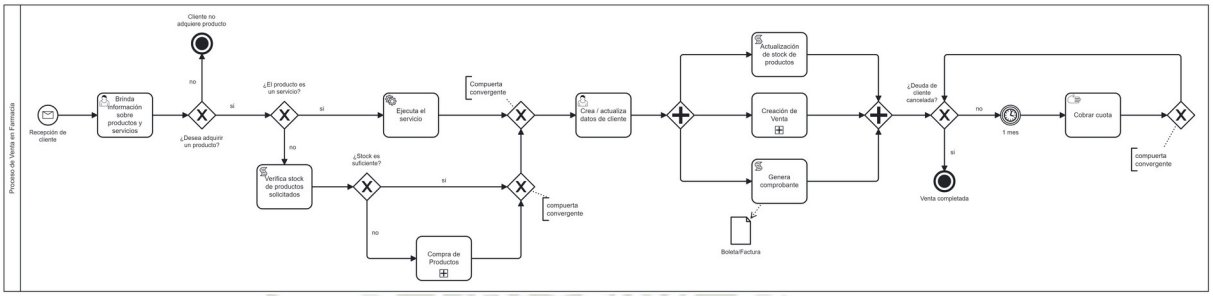
Al igual que en los diagramas UML presentados anteriormente, el diagrama BPMN representa los procesos de venta, compra y abastecimiento de inventario para las farmacias, abarcando más detalles del proceso como compuertas exclusivas, eventos y procesos ejecutados por el sistema.

1. El proceso da inicio cuando un cliente solicita información sobre un producto o servicio. El vendedor le proporciona la información solicitada y el cliente puede decidir en adquirir o no el producto.
2. Si se requiere adquirir un servicio, este es ejecutado, se genera un comprobante y la venta es completada.
3. Si se requiere adquirir un producto, primero se necesita verificar la existencia de stock de éste, si hay suficiente stock la venta es realizada, si el stock mínimo del producto es alcanzado, se requerirá realizar una compra.
4. Para el proceso de compra, el vendedor solicita una cotización al proveedor, este envía la cotización y es registrada en el sistema.
5. Si la cotización es aceptada, se emite una orden de compra que es enviada al proveedor, esto generará entonces un comprobante con el detalle de los lotes de productos.
6. Si la cotización no fue aceptada, se volverá a solicitar una cotización a otro proveedor.
7. La compra es registrada en el sistema, los lotes son recibidos registrando una entrada en almacén y el stock del producto es actualizado.
8. En caso de generarse una deuda con el proveedor, esta se registrará en el sistema y se pagará periódicamente.
9. Para completar la venta se requiere solicitar información personal del cliente para crearlo / actualizarlo en la base de datos.
10. El cliente elegirá el tipo de pago: al crédito o al contado, si es al crédito se generará una deuda la cual se irá cobrando mes a mes.
11. Un comprobante de venta será generado, se entregará la boleta o factura al cliente en conjunto con los productos adquiridos.

12. Finalmente, el stock del producto será actualizado y se registrará la salida de almacén.

Figura 15

Diagrama BPMN proceso de venta



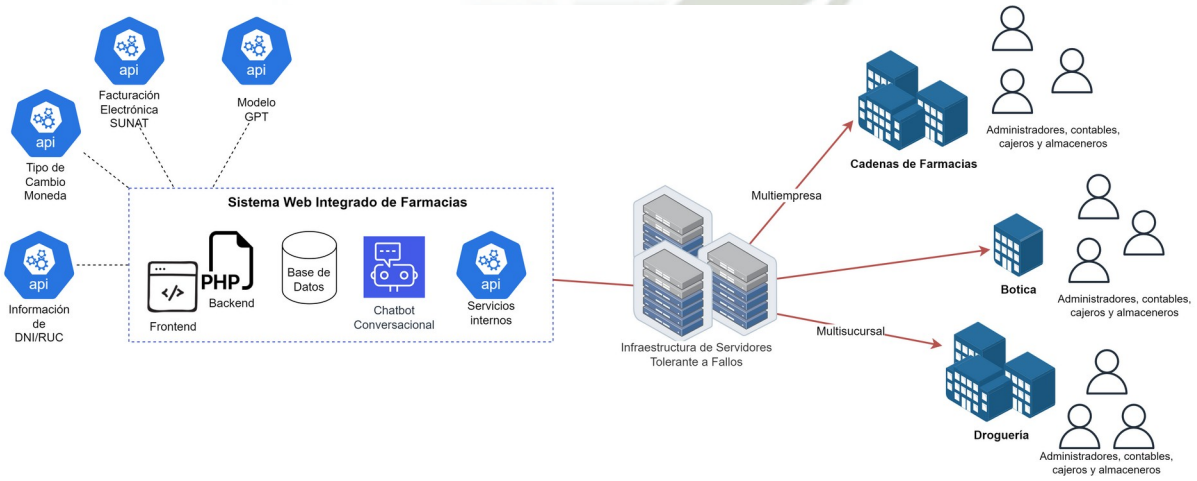
Nota. Para visualizar el BPMN del proceso de venta completo puede consultar en **Anexo 3**.

2.3.6 Diagrama General del Producto Final

Luego de describir los diagramas que intervienen en los distintos procesos del proyecto, es acertado brindar una visión general del producto final, una arquitectura donde el sistema integrado para farmacias se relaciona con otros componentes para dar lugar a una solución completa. Puede visualizar el diagrama elaborado en **Figura 16**.

Figura 16

Arquitecutra General del Producto Final



Nota. Podemos ver los distintos elementos que conforman el sistema integrado de farmacias (stack de la aplicación, base de datos, *chatbot* y servicios API internos), estos a su vez consumen APIs externas para sus operaciones. A la derecha podemos observar los servidores web que alojan la aplicación y que dan respuesta a los tipos de clientes.

2.4 Desarrollo de Software

En este punto se especificarán todos los detalles técnicos pertenecientes a la programación del *software* y el enfoque que el equipo de desarrollo tuvo para cada una de las partes que componen el proyecto, tomando como ejemplo el módulo “Productos”.

2.4.1 Estructura del Directorio del Proyecto

Antes de entrar de lleno a la codificación del proyecto, explicaremos la estructura de carpetas y archivos de este. El proyecto cuenta con tres grandes grupos de archivos los cuales fueron accedidos y modificados con frecuencia durante todo el proceso de desarrollo del *software*, tener en cuenta que esta estructura se basa en la arquitectura MVC. Las partes se listan a continuación:

1. **Carpeta *app*.** En esta carpeta se guardan los archivos de modelos en la subcarpeta “*Models*” y los controladores en la subcarpeta “*Http/ Controllers /API*”; ambos tipos de archivos se almacenan en formato *php*. También podemos encontrar, en la ruta “*Http/Controllers/Auth*”, los archivos necesarios para el registro e inicio de sesión de usuarios; estos últimos controladores son incluidos automáticamente al crear el proyecto Laravel. Finalmente, en la ruta “*Http/Controllers/Middleware*” encontramos los algoritmos de inspección y filtrado *http* que utiliza Laravel, esto como medida de seguridad para solicitudes a la aplicación.
2. **Carpeta *resources*.** Aquí se almacenan todos los archivos necesarios para la construcción de las vistas (*.vue, .css y .blade.php*). Dentro de la subcarpeta “*js*” encontramos los componentes y las páginas construidas en VueJs, además, en la ruta “*js/routes/index.js*” se ubican las rutas *JavaScript* necesarias para que la plataforma redireccione a cada una de las páginas codificadas en archivos *.vue*. Finalmente, la parte más crucial se encuentra en la ruta “*js/pages*”, donde se

organizan en subcarpetas los archivos de cada página, estas carpetas se nombran en referencia a los módulos del sistema.

3. **Carpeta *routes*.** Aquí se encuentran los archivos "*web*" y "*api*" en formato *php*, los cuales sirven para especificar las rutas y métodos de acceso a ellas para cada función en los controladores. Estas rutas pueden ser incluidas o no en un *middleware* que permita el acceso a ellas sólo si se cumplen determinadas condiciones.
4. **Carpeta *database.*** Esta carpeta se divide en tres subcarpetas relacionadas a operaciones ejecutadas directamente sobre la base de datos. En primer lugar, la carpeta *migrations* es usada para crear las migraciones en formato *php* para cada tabla de la base de datos, esto permite una forma de trabajo ordenada y ágil para modificar o crear nuevas tablas y registros, con un simple comando podemos migrar estas instrucciones a nuestra base de datos operativa. En segundo lugar, la carpeta *seeds* la cual es útil para insertar registros predefinidos que serán creados al momento de hacer migraciones a la base de datos, esto es útil por ejemplo para definir usuarios de prueba mientras la aplicación se encuentra en desarrollo. Finalmente, la carpeta *factories* es utilizada para crear registros de prueba en masa, por ejemplo, podemos definir que se creen 100 usuarios aleatorios para hacer pruebas sobre la aplicación.

2.4.2 Diseño de Base de Datos

El diseño de base de datos desempeña un papel crucial en la estructuración y organización eficiente de la información dentro del sistema de farmacias. Este capítulo se enfoca en la definición detallada de las tablas, campos, relaciones y restricciones que regirán la gestión de datos, asegurando la integridad y la consistencia de la información en todo momento.

2.4.2.1 Diseño de Tablas.

El sistema de farmacias requiere un conjunto de tablas bien definidas para gestionar los aspectos esenciales de las operaciones y satisfacer los requisitos del sistema, cada una de estas se relaciona con los procesos de compras, ventas, gestión de inventario, productos, gestión de negocio y catálogos.

Además, se deben tomar en cuenta tablas adicionales que den sustento el propio funcionamiento del *software* como lo son los usuarios, roles y permisos, rutas de archivos almacenados, tokens de autorización, etc.

2.4.2.2 Relaciones y Restricciones.

Las relaciones entre las tablas son fundamentales para mantener la coherencia de los datos, por este motivo se establecieron los siguientes lineamientos:

1. Todas las tablas maestras que estén relacionadas con los procesos de compra o venta deben también tener relación con la tabla sucursales, esto con el fin de diferenciar las operaciones por cada empresa y sucursal.
2. La tabla productos deben contener relaciones muchos a uno con las tablas marcas, unidades de medida, categorías, tipos y sucursales.
3. La tabla de detalle del listado de precios debe relacionarse con sucursales, productos y listas de precios.
4. Las tablas roles y permisos deben poseer una relación de muchos a muchos, por lo que se requiere una tabla intermedia para almacenar estos registros.
5. Toda sucursal debe estar relacionada con una empresa, y esta a su vez debe relacionarse con la tabla *file_paths* para almacenar la ruta del logo de esta.

Además de las relaciones, se han implementado restricciones para asegurar la precisión de los datos. Por ejemplo:

- Restricciones de clave primaria y foránea se han aplicado para prevenir registros duplicados y garantizar la integridad referencial.
- Valores como precios y cantidades se someten a restricciones para asegurar que sean válidos y coherentes.
- Los campos de email, teléfono y número de documento cuentan con validaciones que verifican su estructura y autenticidad.
- Los campos número de documento y código también cuentan con restricciones que evitan registros duplicados.

2.4.2.3 Normalización de Datos.

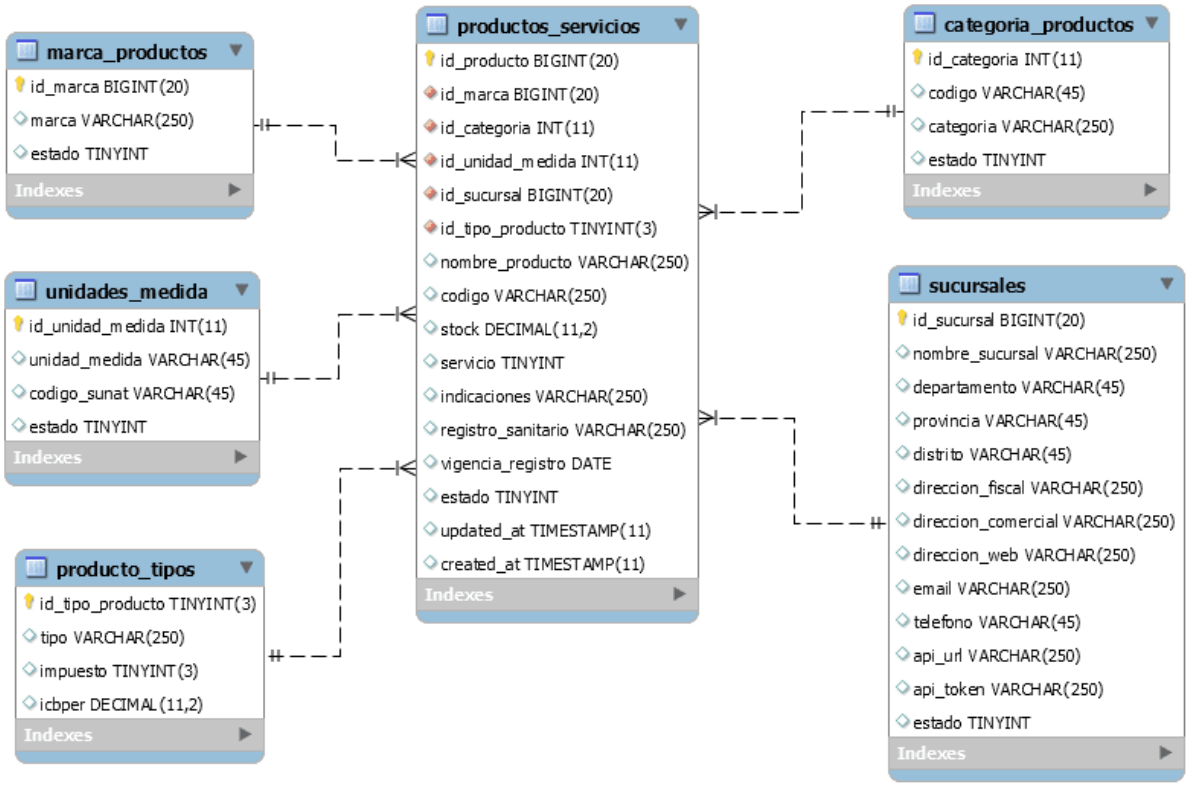
La normalización de datos es esencial para eliminar redundancias y anomalías, asegurando la integridad y eficiencia del sistema. Se ha aplicado la normalización hasta la tercera forma normal (3NF), garantizando que los datos estén organizados de manera lógica y óptima; si desea conocer más detalles sobre este proceso puede dirigirse al punto **1.7.2.2**.

2.4.2.4 Diagrama Entidad-Relación.

El módulo productos se sustenta en la tabla maestra *productos_servicios* que posee relaciones con las tablas: *marca_productos*, *unidades_medida*, *producto_tipos*, *categoría_productos* y sucursales; estas relaciones dan sentido a cada uno de los registros de la tabla y reflejan la complejidad real que estos tienen dentro del negocio tal como se muestra en **Figura 17**. Adicionalmente, para observar la totalidad del diagrama entidad-relación de la base de datos del proyecto consulte el **Anexo 1**.

Figura 17

Diagrama entidad-relación de la tabla productos_servicios

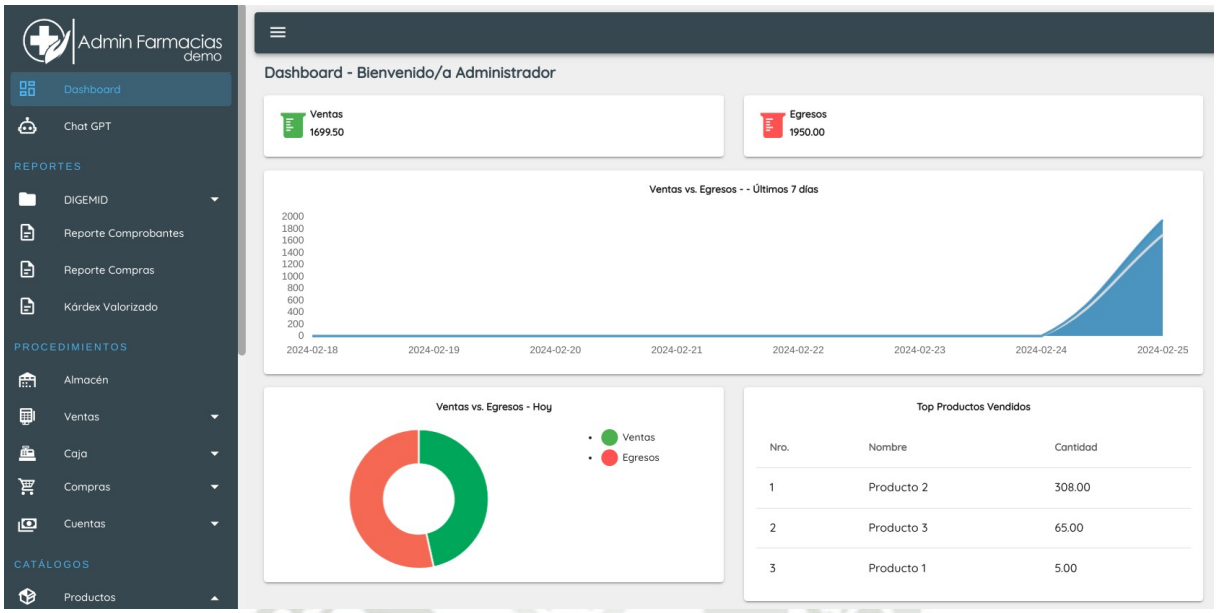


2.4.3 Interfaz de la Aplicación

La siguiente captura es una muestra de cómo se implementaron las vistas diseñadas en **2.3.1** a la interfaz y funcionalidad de la aplicación final. En **Figura 18** podemos observar la pantalla del bienvenida que a su vez cumple el rol de representar un *dashboard*. También puede volver a revisar en **Figura 5**, los distintos componentes que conforman una vista en la aplicación, y cómo estos pueden ser actualizados de forma independiente sin que esto implique una recarga de la página en su totalidad que afecte al resto de componentes.

Figura 18

Interfaz del dashboard de la aplicación



Nota. En la pantalla del *dashboard* de la aplicación podemos observar: ventas totales, egresos totales, ventas vs egresos de los últimos 7 días, ventas vs egresos del presente día y el top productos vendidos.

2.4.4 Modelo Producto

Haciendo uso de la arquitectura MVC, el primer paso para referenciar esta tabla es mediante la creación del modelo para el producto, consiste en un archivo denominado *Producto.php* localizado dentro de la carpeta *app* del proyecto, en el cual se creará una clase Producto, se listarán su clave primaria, atributos, relaciones y métodos, tal y como se muestra en el siguiente código.

```
class Producto extens Model {
      protected $table = 'productos_servicios';
      protected $primaryKey = 'id_producto';
      protected $fillable = [
            id_producto',
            id_categoria',
            id_marca',
      }
      public $timestamps = true;
```

```
        protected $with = array('sucursal', 'categoria', 'marca');


        public function sucursal(){
              return $this->belongsTo('App\Sucursal', 'id_sucursal');
        }
        public function categoria(){
              return $this->belongsTo('App\Categoria','id_categoria');
        }
        public function categoria(){
              return $this->belongsTo('App\Categoria','id_categoria');
        }
        public function marca(){
              return $this->belongsTo('App\Marca','id_marca');
        }
}
```

El modelo Producto cuenta con variables, identificadas con el símbolo $, que hacen referencia al nombre de la tabla, clave primaria, campos y relaciones; estas últimas mediante métodos públicos donde se especifica el tipo de relación, el modelo y la clave foránea.

2.4.5 Controlador de Productos

En cuanto a la estructura de un controlador en la arquitectura MVC, para el caso de productos, se encuentra en un archivo denominado "*ProductoController.php*" el cual contiene las siguientes funciones:

- *Index*, función utilizada para listar los registros de la tabla *productos_servicios*, éstos se pueden filtrar mediante parámetros definidos tales como término de búsqueda, fechas, tipos, etc.
- *Store*, recibe como parámetro un dato del tipo *Request* el cual contiene la información que el usuario introdujo desde la vista. Implementa métodos de validación de campos, también almacena lotes y listas de precios del producto.

- *Show*, recibe el *id* de un único producto que se desea visualizar y devuelve la información de este, es útil cuando se desea cargar toda la información de un solo registro en una ventana o página para su edición.
- *Update*, recibe como parámetros un *Request* y el *id* del registro que se desea actualizar, su funcionamiento en productos es similar al método *Store* con la diferencia de que no se crean nuevos registros en productos ni en lotes.
- *Destroy*, recibe como parámetro el *id* del registro y ejecuta un borrado lógico sobre el registro seleccionado, es decir, los registros se desactivan de la base de datos más no se eliminan por completo de esta. También implementa un control que no permite la acción de borrado en caso el producto se encuentre registrado en operaciones de otros módulos del sistema.
- *GetReporteProductosLotes*, esta función no forma parte del módulo productos, sino que está destinada a la generación de reportes. Similar al método *index* lista los registros de la tabla implementado filtros, con la diferencia que ejecuta operaciones *join* sobre las tablas: lotes, productos, comprobantes y detalle de comprobantes para mostrar las relaciones entre estos datos requeridas en el reporte final.

A continuación, se muestra un extracto del código fuente de *ProductoController.php* a modo de ejemplificación y resumen de lo anterior expuesto:

```
class ProductoController extends Controller {
      public function index(){}
      public function store(Request $request){}
      public function show($id){}
      public function update(Request $request, $id){}
      public function destroy($id){}
      public function getProductos(Request $request){}
      public function getReporteProductoLotes(Request $request){}
```

```
}
```

Con estas funciones implementadas será posible entonces que el usuario final acceda a ellas desde la vista del módulo, esto mediante una simple llamada desde el cliente especificando el método *http* (GET, POST, PUT o DELETE), la ruta y los parámetros si así se requiere.

2.4.6 Declaración de Rutas

Para que sea posible acceder a cada una de las funciones declaradas en el controlador desde la vista "Productos" es necesario declarar las rutas de acceso a cada función mediante los archivos *routes* que Laravel pone a disposición. Para este proyecto en concreto se asignaron dos tipos de rutas:

- Rutas *web*: se encuentran registradas en el archivo *web.php*, son exclusivas para usuarios finales, es decir, sólo pueden ser accedidas de forma interna en el sistema y con los permisos necesarios. Utilizan el *middleware* web proporcionado por Laravel, el cual cuenta con una serie de características como CSRF, sesiones y gestiona la información de manera *stateful*, es decir, la información útil para próximos accesos es persistente en el cliente (Otwell, 2023).
- Rutas *api*: están habilitadas para hacer uso de ellas a modo de servicios, es decir, reciben parámetros y retornan una respuesta al igual que otras rutas con la diferencia que se puede acceder a ellas de forma externa, siempre y cuando se posean los permisos necesarios. Utilizan el *middleware api* proporcionado por Laravel, el cual cuenta con la única protección de *throttling* que sirve para limitar la cantidad de llamadas a una función en determinado periodo de tiempo. Gestiona la información de forma *stateless*, es decir, no almacena ningún tipo de información (Otwell, 2023).

En el caso del módulo Productos las rutas de acceso a sus métodos CRUD fueron declaradas en el archivo *api.php*, cabe resaltar que este tipo de rutas se configuran como

apiResources, es decir, con tan sólo la línea 'producto' => '*Api\ProductoController*' se incluyen de forma automática los métodos CRUD. Por otra parte, las funciones para generar los reportes en formato Excel fueron incluidas como rutas *web*. Puede visualizar la declaración de las rutas Producto en la siguiente porción de código.

```
//--- MAESTROS ---
Route::apiResources([
      //--- Productos ---
      'producto' => ProductoController::class,
      'categoria' => CategoryController::class,
      'marca' => MarcaController::class,
      'laboratorio' => LaboratorioController::class,
      'condicion_almacenamiento' => CondicionAlmController::class,
      'unidades_medida' => UnidadesMedidaController::class,
      'price_list' => ListaPreciosController::class,
      'pricelist_detail' => ListaPreciosDetalleController::class,
      //--- Fin ---
]);
//--- FIN ---
//--- REPORTES ---
//--- Exports Controller ---
Route::get('/exportarKardexValorizado/{data}', [ExportsController::class,
'exportarKardexValorizado'])->middleware('auth');
Route::get('/exportarReporteProductLote/{data}', [ExportsController::class,
'exportarReporteProductLote'])->middleware('auth');
Route::get('/exportarReporteAlmacen/{data}', [ExportsController::class,
'exportarReporteAlmacen'])->middleware('auth');
Route::get('/exportarReporteComprobanteFormat/{data}',
[ExportsController::class, 'exportarReporteComprobanteFormat'])-
>middleware('auth');
```

```
Route::get('/exportarReporteComprobanteGeneral/{data}',
[ExportsController::class, 'exportarReporteComprobanteGeneral'])-
>middleware('auth');
Route::get('/exportarReporteCompraFormat/{data}',
[ExportsController::class, 'exportarReporteCompraFormat'])-
>middleware('auth');
Route::get('/exportarCuentasPagar/{data}', [ExportsController::class,
'exportarCuentasPagar'])->middleware('auth');
Route::get('/exportarCuentasCobrar/{data}', [ExportsController::class,
'exportarCuentasCobrar'])->middleware('auth');
//--- Fin ---
//--- FIN ---
```

2.4.7 Vista Producto

Una vez explicada la estructura del *backend* de la aplicación web, es pertinente agrupar estos conceptos en el segmento más cercano al usuario final, la vista del usuario. En este proyecto el *framework* principal para la construcción de interfaces de usuarios fue VueJs, el cual es un *framework* de *frontend* para Javascript especializado en la construcción de interfaces y basado en la programación por componentes. Hace que la navegación sea más fluida, las interfaces dinámicas y también brinda una gran facilidad a la hora de programar cada una de las páginas (You et al., 2022), más detalles en sobre esta tecnología en **1.7.2.1**. La forma de trabajo de VueJs es que cada página se programa como un componente (archivos *.vue*), estos cuentan con distintas propiedades tales como: cambios de estado, *declarative rendering*, y reactividad. Cada uno de estos componentes requiere ser registrado en un archivo de rutas para las páginas nombrado "*routes.js*"; a continuación, se muestra una porción del código de este archivo.

```
//--- CATALOGOS ---
//--- Productos ---
```

```
{path:'/productos',name:'products',component:require('../pages/Productos/
ProductPage.vue').default},
{path:'/crear_productos,component:require('../pages/Productos/
ProductCreatePage.vue').default},
{path:'/editar_productos/:id,component:require('../pages/Productos/
ProductEditPage.vue').default},
{path:'/categorias',name:category,component:require('../pages/Categorias/
CategoryPage.vue').default},
{path:'/marcas',component:require('../pages/Marcas/index.vue').default},
{path:'/lista_precios',component:require('../pages/ListaPrecios/
ListaPrecioPage.vue').default},
{path:'/lista_precios_detalle/:id',component:require('../pages/
ListaPrecios/ListaPrecioDetallePage.vue').default},
//--- Fin ---
```

Donde se listan cada una de las rutas a las que el usuario puede ser redireccionado, se componen por el *path* que es el nombre de la ruta web a crear, un sobrenombre si se requiere y la ruta del componente *.vue* en el directorio del proyecto. Adicionalmente algunas de estas rutas pueden recibir parámetros, como por ejemplo la ruta "*editar_productos*" recibe el *id* del producto seleccionado.

Cabe volver a recalcar que Vue no recarga la página entera, sino que sólo actualiza el componente, por lo que al cambiar de ruta de página sólo se estaría reemplazando el componente de la vista, mientras que los componentes del menú lateral o superior seguirían intactos.

El archivo *app.vue* funciona como la estructura base de las vistas, en el siguiente fragmento puede observar la codificación de este y los distintos componentes que lo conforman. Donde podemos notar que contamos con un componente *<navbar></navbar>* para el menú lateral, *<app-header></app-header>* para la barra superior y finalmente un *<v-*

container></v-container> con un componente *<router-view></router-view>* que es donde se actualizará la vista de cada módulo.

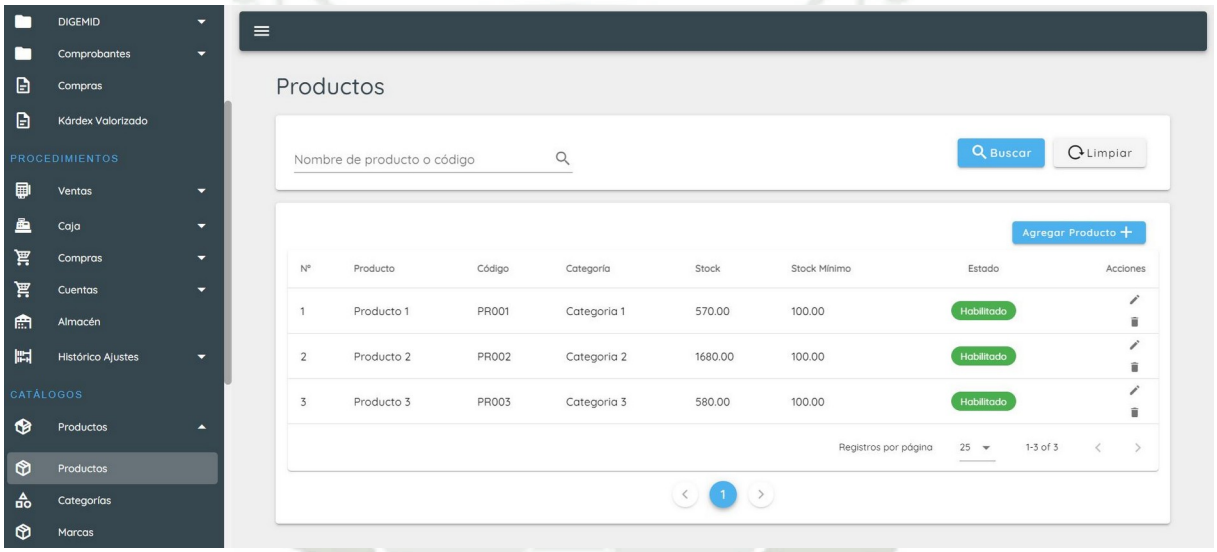
```
<template>
      <v-app id="main" style="background: #efefef;">
            <--menú lateral-->
            <navbar ref="navbar"></navbar>
            <--barra superior-->
            <app-header @toggle-drawer="$refs.navbar.drawer = !
            $refs.navbar.drawer">
            </app-header>
            <-- vista principal -->
            <v-main>
                  <v-container style="padding: 10px 30px;">
                        <--componente encargado de redireccionar a vistas-->
                        <router-view></router-view>
                  </v-container>
            </v-main>
            <-- pie de página -->
            <app-footer></app-footer>
      </v-app>
</template>
<script>
import Navbar from './components/Navbar'
import Header from './components/Header'
import Footer from './components/Footer'
export default {
      name: 'App',
      components: {
            'navbar': Navbar,
            app-header': Header,
```

```
            app-footer': Footer
        }
}
</script>
```

Ahora podríamos dirigirnos a la vista inicial del módulo Productos haciendo clic en el menú lateral, tal y como se muestra en **Figura 19**.

Figura 19

Vista principal del módulo Productos



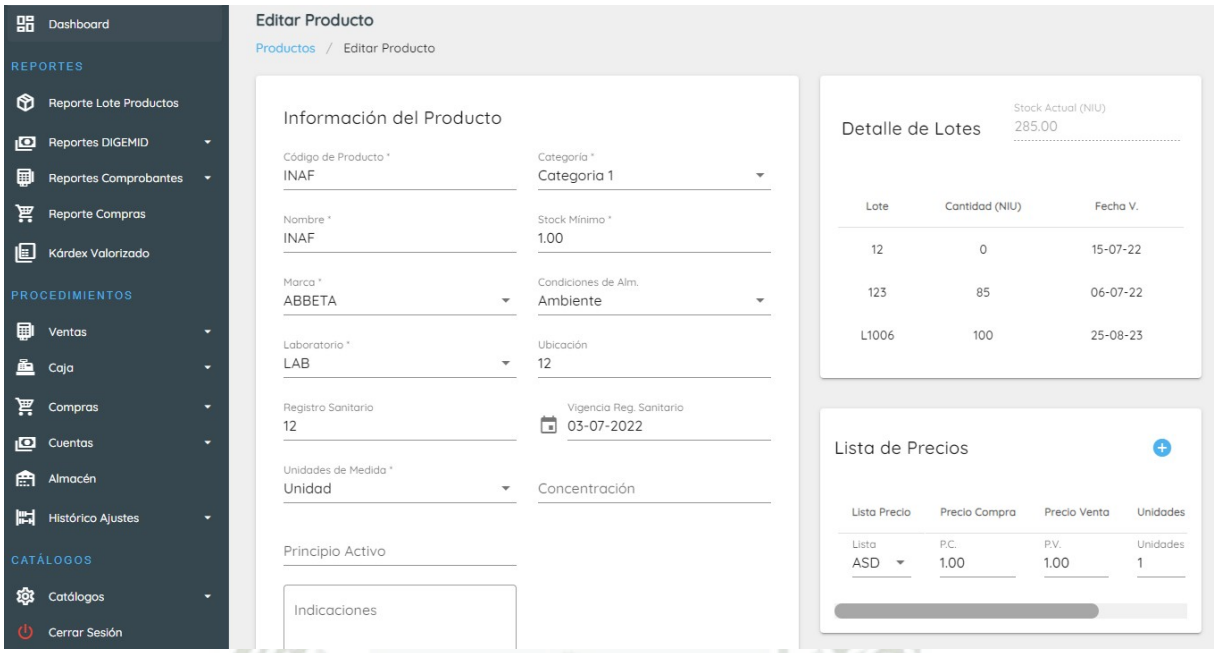
Nota. La pantalla principal de Productos está formada por el filtro de nombre o código, la lista completa de registros y botones para editar o borrar cada uno de ellos. También es destacable el menú lateral del sistema desde el cual todos los módulos son accesibles.

La interfaz del módulo Productos se planteó en base a tres vistas: la página principal, la página de creación y la página de edición. Esta decisión fue tomada debido a que las relaciones de la tabla producto son numerosas, de modo que, cuando se requiere crear o editar un nuevo registro es necesario contar con el espacio necesario en la interfaz para distribuir la información.

En **Figura 20** se puede observar la vista de creación de productos, sin embargo, es pertinente aclarar que la pantalla de edición es similar, simplemente se agrega una restricción al intentar registrar nuevos lotes de producto desde este módulo.

Figura 20

Vista de creación/edición del módulo Productos

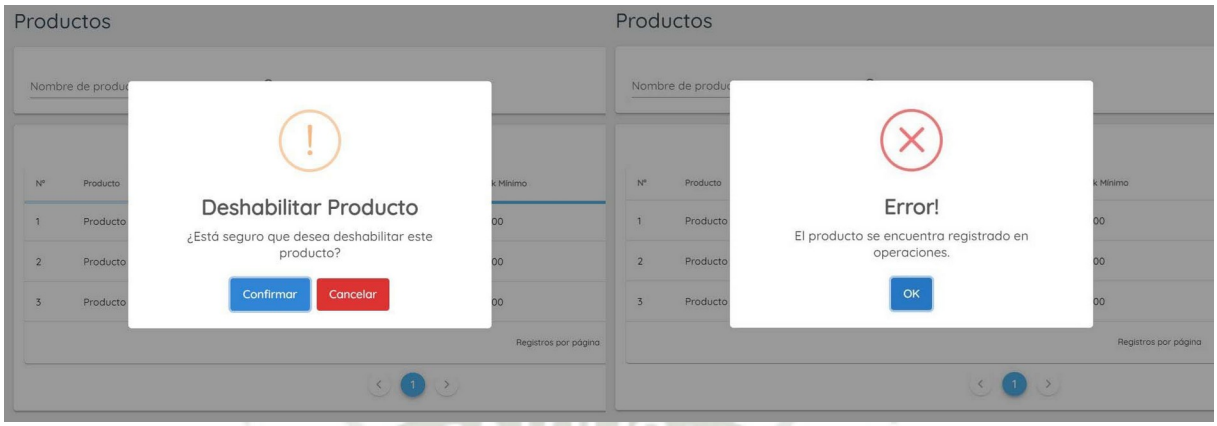


Nota. La pantalla de creación/edición de producto se divide en tres secciones: información de producto, donde se encuentran los campos tabla siendo gran parte de ellos obligatorios; detalle de lotes, que sólo es habilitado al crear un nuevo registro y determinará el stock del producto y finalmente el listado de precio, en el cual se especifican los precios de compra y venta para las distintas presentaciones del producto (unidad, blíster o caja).

Posterior a la creación del nuevo producto, este será accesible desde todos los módulos que lo permitan en el sistema con la consideración de que, en caso el producto sea inhabilitado, no será visible ni accesible en muchos de ellos. También hay que destacar que la inhabilitación del producto sólo será posible si este no se encuentra registrado en operaciones, es decir, no se encontró el *id_producto* en registros de otras tablas; de lo contrario un mensaje de error se mostrará al pretender borrar el registro como se muestra en **Figura 21**.

Figura 21

Mensaje de error al borrar producto registrado en operaciones



Para finalizar, todo lo anteriormente explicado sobre la codificación del módulo productos es aplicable al resto de páginas en el sistema. El resto de los módulos del sistema comparten características como su construcción sobre la arquitectura MVC, también cuentan con funciones similares como el listar registros mediante filtros, crear registros, editar registros y borrar registros (funciones CRUD). Algunos módulos operan los datos de formas muy específicas, sin embargo, no es necesario detallarlos aquí ya que simulan procesos específicos del negocio farmacéutico.

2.4.8 Roles y Permisos de Usuarios

Una vez construidas las páginas de la plataforma de farmacias fue necesario implementar el sistema de roles y permisos a todas las partes del *software* consideradas. Este es un módulo extenso que tiene impacto directo en la totalidad de la plataforma, cobra especial relevancia al contar con diversos tipos de usuarios interactuando con la aplicación. En esta ocasión se hizo uso de la librería *laravel-permission* perteneciente a la colección de recursos *Spatie*. Una vez instalada esta librería en el proyecto, todos los complementos necesarios para la implementación del sistema de roles y permisos estuvieron disponibles para su uso.

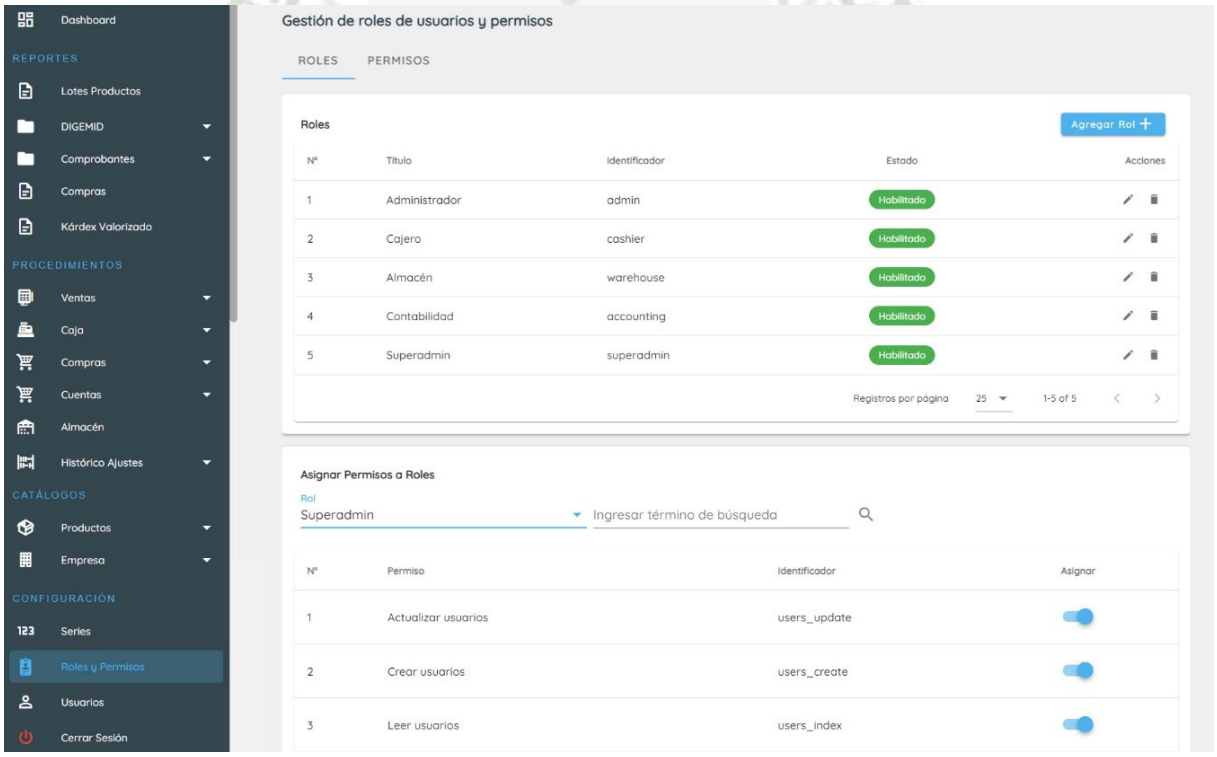
La forma de implementar el sistema de permisos al proyecto mediante *laravel-permission* fue, en primer lugar, migrar las tablas prefabricadas *roles*, *permissions* y

role_has_permissions a la base de datos. Estas tablas se importan de la librería e incluyen sus archivos de modelos propios, por lo que no requieren mayor configuración en el proyecto.

El siguiente paso consistió en desarrollar el CRUD de roles y permisos donde se habilitaron las acciones de creación y edición de roles, permisos y asignación de permisos a roles. La codificación de este módulo es similar a la forma en la que se programó el módulo de productos por lo que no se requiere mayor explicación en ese sentido, sin embargo, un detalle diferencial en esta vista es que se trabajaron las distintas listas de registros en una misma página divida por secciones y pestañas, esto con el fin de tener mayor comodidad. Observe la **Figura 22** donde se muestra la totalidad de la vista "Roles y Permisos".

Figura 22

Vista principal del módulo Roles y Permisos



Nota. Se divide en tres secciones, la primera para el CRUD de roles, la segunda para el CRUD de permisos y la tercera para la asignación de permisos por rol de usuario.

Puede observar que cada uno de los permisos fueron creados con el patrón "*módulo_permiso*" para conservar el orden en estos. Una vez se crearon los permisos estos

fueron incluidos en las distintas partes de las vistas del sistema mediante el código que observa a continuación:

```
<v-card class="mb-4" light style="padding: 15px">
      <v-row dense class="pa-2 align-center">
            <v-col v-if="$can('users_create', 'all')" class="text-right">
                  <v-btn small color="primary" class="mr-2"
                  @click="addDialog=true;">
                        Agregar Usuario<v-icon>mdi-plus</v-icon>
                  </v-btn>
            </v-col>
      </v-row>
</v-card>
```

Donde *v-if* es el condicional que incluye VueJs para el código en *frontend*, *$can()* es una función propia de la librería *laravel-permission* que permite identificar si el usuario que inició sesión tiene determinado permiso y *user_create* es el permiso que se quiere comprobar. Se incluyeron entonces condiciones similares a lo largo de todas las vistas del sistema.

Finalmente, el proceso de importación del rol y permisos de determinado usuario se hace cada vez que este inicia sesión o cuando los roles/permisos son modificados; de esta forma nos aseguramos de que siempre el acceso del usuario a la plataforma esté sincronizado con los condicionales de permisos. La primera parte del código, el archivo *Login.vue*, se muestra a continuación, donde se observa la función *login()* en *javascript* en la que se almacenan de forma local los datos del usuario que inicia sesión y los permisos de este importados desde el modelo *User*.

```
login(){
      this.preloader = true;
      axios.post('/login', {
            email: this.user.email,
            password: this.user.password
```

```
        }).then(response => {
                this.$router.push('/');
                localStorage.setItem('user_data',
                JSON.stringify(response.data.user_data))
                localStorage.setItem('user_permissions',JSON.stringify(response
                .data.permissions));
                location.reload();
        }).catch(e => {
                console.error(e);
        }).finally(()=>(this.preloader = false));
},
```

La segunda parte del código sirve para la comprobación de los permisos en los componentes *.vue* mediante la consulta *$can()*. Cada permiso del usuario que inició sesión se almacena en un arreglo, posteriormente se recorren los elementos y se registran usando la función *can(permission, 'all')* como se muestra a continuación.

```
import { defineAbility } from '@casl/ability';
export default defineAbility((can, cannot) => {
        let permissions = [];
        permissions = JSON.parse(localStorage.getItem('user_permissions'));
        if(permissions){
                for (let index = 0; index < permissions.length; index++) {
                        const element = permissions[index];
                        can(element.name, 'all');
                }
        }
});
```

CAPITULO III

3 INTEGRACIÓN DE IA GENERATIVA A LAS OPERACIONES DEL SISTEMA

La integración de los modelos de IA Generativa con la base de datos del sistema de farmacias representa un logro significativo en la evolución de la interacción entre el usuario y la aplicación. Este capítulo se enfoca en establecer el propósito y resaltar la importancia de esta integración clave, ya que, al fusionar la potencia del procesamiento del lenguaje natural con la variedad de datos almacenados, se abre un abanico de posibilidades capaz de mejorar la experiencia del usuario. Abordaremos temas como la arquitectura de la integración, el diseño de la interfaz del chat, la lógica del algoritmo, los resultados, ventajas y limitaciones que están asociados a esta tecnología.

3.1 Propósito e Importancia

ChatGPT pertenece a la familia Generative Pretrain Transformer (GPT), se trata de un modelo de lenguaje basado en la arquitectura GPT 3.5, o su versión más reciente GPT 4, y su principal característica es la capacidad de llevar una conversación fluida con el usuario elaborando respuestas, escribiendo código, resumiendo párrafos de artículos, corrigiendo texto, etc. (OpenAI, 2022b). Por otro lado, la familia GPT está conformada por un tipo específico de Large Language Models (LLMs) los cuales se describen como modelos de procesamiento de lenguaje natural a gran escala capaces de generar texto o predecir palabras en una oración (Chase, 2023).

Es necesario aclarar que la integración de aplicaciones con los servicios de OpenAI no se realizan con ChatGPT, ya que este título hace referencia al modelo basado en GPT 3.5 que cualquier persona puede utilizar si crease una cuenta e ingresase a la plataforma desde el navegador. Al contrario, el consumo de OpenAI API se basa en los diferentes modelos de la familia GPT disponibles, cada uno con sus ventajas y desventajas, los cuáles podremos seleccionar de acuerdo con nuestras necesidades y presupuesto.

El propósito de la integración del sistema de gestión de farmacias con los distintos modelos de GPT radica en la capacidad de contribución a la usabilidad y accesibilidad en la aplicación, adicionando un *chatbot* conversacional que permite al usuario realizar consultas en lenguaje natural, y acceder a información actual en la base de datos a través de un modelo de lenguaje adaptado a sus requerimientos. Al aprovechar la información almacenada en la base de datos, el modelo GPT puede comprender y responder a las consultas de manera más precisa y relevante obteniendo respuestas personalizadas y contextualizadas sobre los resultados del negocio, ventas del día, productos más vendidos, etc.

Los beneficios son tangibles y abarcan desde un incremento en la eficiencia de la búsqueda de información hasta una mayor confiabilidad en las recomendaciones y decisiones proporcionadas por el sistema. También, la integración con este modelo permite abrir un mundo de posibilidades para la plataforma como lo serían la generación de contenido, soporte al cliente o asistentes virtuales, lo que agregaría más niveles de accesibilidad a esta misma.

3.2 Prerrequisitos de la Integración

3.2.1 Actualización de la Versión del Framework

Inicialmente, el proyecto fue diseñado para funcionar en la versión 7 de Laravel, no obstante, las herramientas disponibles hoy en día para integrar este *framework* con los modelos GPT están desarrolladas para versiones más recientes. Por este motivo, se tomó la decisión de migrar el proyecto a una versión más actualizada, en concreto, Laravel 9. Aunque esta actualización representó una tarea crítica que demandó un tiempo considerable, es importante destacar que la documentación de Laravel aborda de manera exhaustiva escenarios como este en cada una de sus versiones, lo cual facilita el trabajo.

Los cambios más significativos se dieron en la actualización de algunas dependencias de Laravel, versión de PHP, nombres de funciones, variables y la forma de registrar las rutas

en los archivos *api.php* y *web.php*. A continuación, se puede ver el cambio en la declaración de las rutas en el sistema:

```
// Anterior forma de declaración de rutas
Route::get('getUserRoles', 'Api\UserRolesController@getUserRoles');
// Nueva forma de declaración de rutas
Route::get('getUserRoles', [UserRolesController::class, 'getUserRoles']);
```

También, como consecuencia de la actualización de dependencias y paquetes, surgieron conflictos con muchos otros de ellos, por lo que el equipo de desarrollo tuvo que actualizar, configurar y reparar cualquier problema que surgiera debido al cambio de dependencias.

Sin embargo, estos cambios no son los únicos que la documentación de Laravel incluye, también se describen modificaciones en los directorios de migraciones de la base de datos, características de Eloquent, eventos del *backend*, validaciones, *testing* y más (Otwell, 2023). Sin embargo, dado que no se emplearon todas las características en el proyecto, no fue necesario dedicar parte del tiempo asignado a la actualización de la versión del *framework* para realizar estas modificaciones.

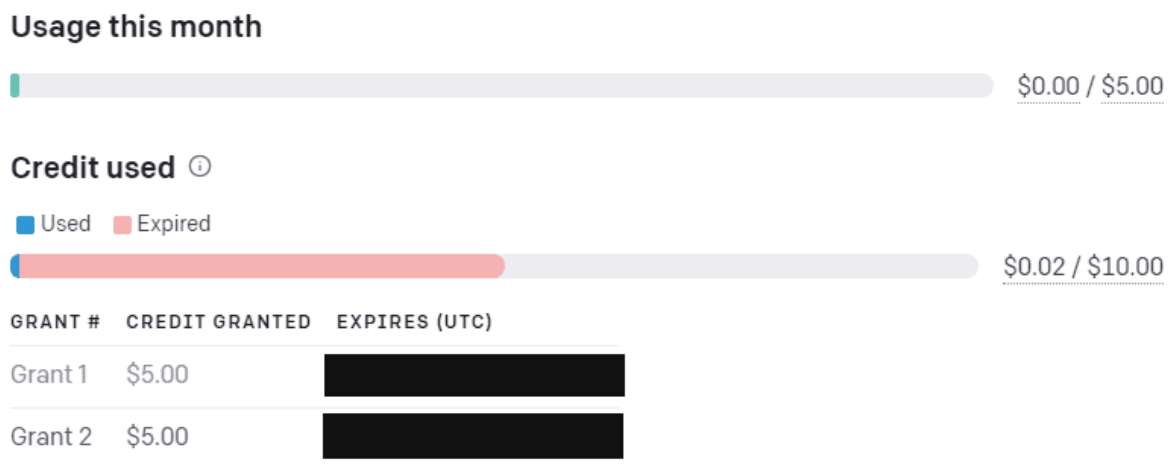
El resultado fue un proyecto actualizado, mejor estructurado y codificado en comparación con su versión anterior. Además, surgió la posibilidad de incorporar diversas herramientas externas modernas que pueden resultar muy beneficiosas para una plataforma de este tipo. Asimismo, la migración a una versión más reciente del *framework* garantiza una mayor seguridad y compatibilidad con las tecnologías y estándares actuales del sector.

3.2.2 Creación de Cuenta y Configuración de API keys

El servicio disponible para hacer uso de los distintos modelos de GPT desde una aplicación o sistema se basa en el envío de solicitudes por API. Para acceder a este es necesario crear una cuenta en la plataforma de OpenAI y abonarla con mínimo 5 dólares americanos, los cuáles se consumirán como crédito por cada cierta cantidad de tokens usados en las consultas; puede consultar en **Anexo 12** para obtener la lista de precios de OpenAI API por modelo. También la plataforma nos brinda un gráfico en el cual nos muestra el crédito usado por mes y el historial de pagos, observe **Figura 23**.

Figura 23

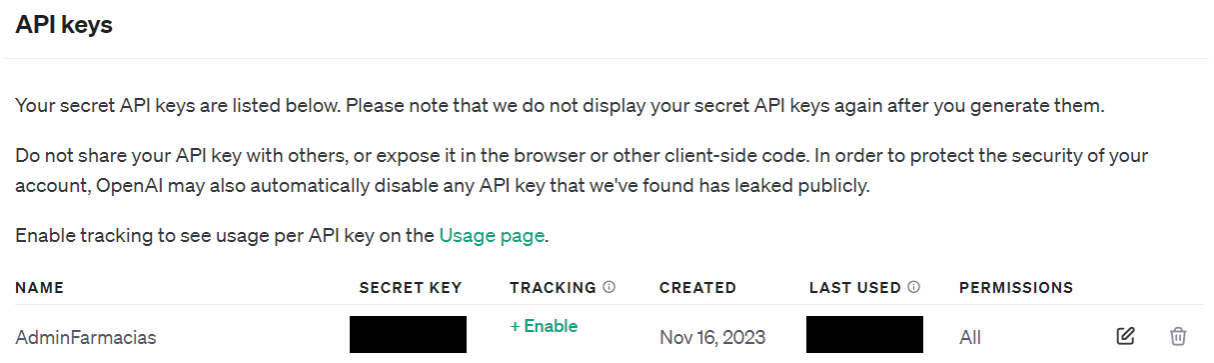
Cantidad de crédito usado por mes e historial de pagos en la plataforma de OpenAI



Una vez hecho el pago en la plataforma, el siguiente paso consiste en crear una *API key* y registrarla en las variables de entorno de nuestro proyecto, esta servirá para autenticar las solicitudes que enviaremos desde el sistema y obtener una respuesta elaborada por el modelo elegido. Puede visualizar en **Figura 24** el listado de *keys* creadas, por el momento contamos sólo con una para el sistema de farmacias.

Figura 24

Lista de las API keys creadas en OpenAI



3.2.3 Configuración del Paquete OpenAI

El último paso de los prerrequisitos consistió en, dentro del proyecto Laravel, instalar el paquete "*openai-php/laravel*" mediante *Composer*. Este es un paquete que facilita la interacción con la API de OpenAI desde aplicaciones basadas en Laravel, ofreciendo características propias que simplifican el proceso de comunicación con el modelo. Puede dirigirse a **Anexo 11** para encontrar el repositorio original de *openai-php/laravel*.

Al momento de instalar esta dependencia en el proyecto se crean scripts en el *backend* de la aplicación, específicamente en el archivo *config/openai.php*, donde podemos observar las siguientes líneas de código:

```
'api_key' => env('OPENAI_API_KEY'),
'organization' => env('OPENAI_ORGANIZATION'),
```

Las cuáles invocan a las variables de entorno del proyecto incluidas en el archivo *.env*, donde podemos definir los tres siguientes parámetros según las necesidades de nuestra arquitectura, esto con el fin de proteger estos datos privados y evitar el *hardcoding*, ya que este archivo no se incluye en el repositorio GitHub.

```
// Variables de entorno OpenAI-Laravel
OPENAI_API_KEY= sk-…
OPENAI_ORGANIZATION= pharmacy
OPENAI_REQUEST_TIMEOUT= 30
```

Finalmente, con las configuraciones anteriormente descritas, el proyecto se encontró apto para hacer uso del servicio de OpenAI API y los modelos GPT tanto desde la lógica interna del código del *backend* como en el *frontend* mediante la interfaz y diseño de la pantalla del chat.

3.3 Arquitectura de la Integración

3.3.1 Diseño de la Arquitectura

Para habilitar la comunicación entre el modelo de IA, el sistema de información y la base de datos, se seleccionaron herramientas y tecnologías especializadas. Se hizo uso de un sistema de gestión de bases de datos relacional como MySQL, el cual ofrece una estructura eficiente para almacenar y recuperar información de forma estructurada y rápida. Además, se implementó una estructura de comunicación que facilita la interacción del LLM con el sistema de información, permitiendo consultas y actualizaciones en tiempo real.

La interconexión entre el modelo de IA y la base de datos se logra gracias a una capa de comunicación gestionada por la arquitectura interna en la aplicación y los modelos GPT, los pasos se muestran a continuación:

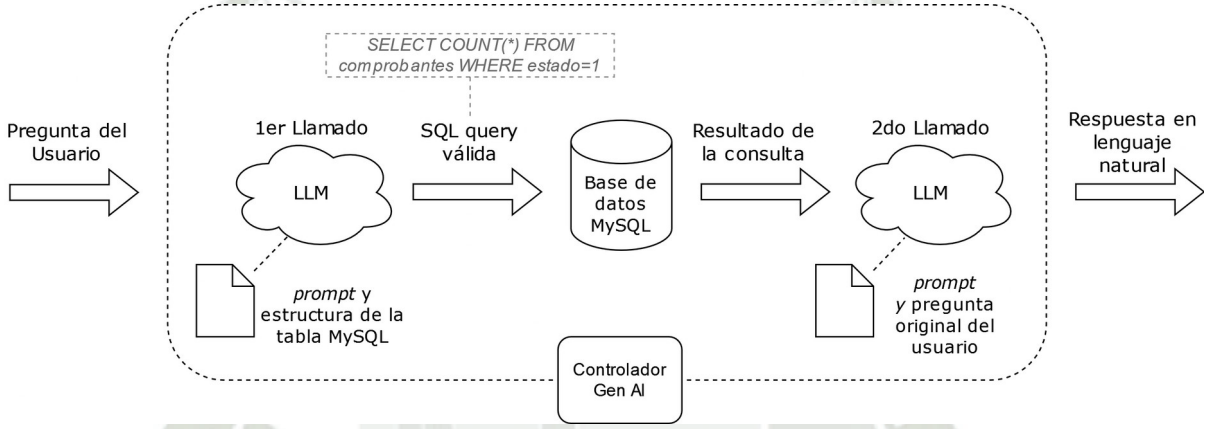
1. El usuario abre la ventana del *chat*, selecciona un tópico y realiza una pregunta o solicitud en lenguaje natural.
2. Un primer llamado al LLM realiza una transformación de la pregunta del usuario a una consulta SQL utilizando la estructura de la base de datos y un *prompt* personalizado con instrucciones específicas.
3. La consulta SQL es validada para evitar acciones indeseadas, y ejecutada en la base de datos para obtener un resultado.
4. Un segundo llamado al LLM transforma el resultado de la consulta SQL en una respuesta en lenguaje natural, utilizando la pregunta original del usuario y un *prompt* personalizado para limitar la información mostrada.

5. Finalmente, la respuesta es mostrada en una burbuja del chat para que el usuario pueda visualizarla.

Esta comunicación bidireccional se gestiona de manera segura y se optimiza para minimizar la latencia y garantizar una experiencia de usuario fluida, puede observar en **Figura 25** la arquitectura de esta comunicación.

Figura 25

Arquitectura de la interacción de la plataforma web, LLM y la base de datos



Nota. La arquitectura fue adaptada a Laravel, sin embargo, el diseño original de esta puede ser encontrado en la documentación de LangChain para agentes de bases de datos SQL (Chase, 2023).

Podemos apreciar que es un proceso simple mediante el cual se podrán elaborar respuestas inmediatas a consultas del usuario. Cada envío de información al LLM se contextualiza mediante la estructura comprimida de una tabla de la base de datos, la pregunta del usuario, unas instrucciones breves para el formato en que queremos al respuesta y parámetros extra que definen el modelo elegido, temperatura de la respuesta, cantidad de tokens, etc. Todos estos puntos se explican más adelante en **3.4.2** donde describiremos de forma técnica la solución implementada.

3.3.2 Características de la Arquitectura

Las características principales de una solución basada en una arquitectura como esta son las siguientes:

- *Context-awareness*, lo que significa que se toma en cuenta el escenario y contextualiza cada solicitud enviada modelo de acuerdo con instrucciones, *embeddings*, ejemplos, etc (Chase, 2023). Es importante recalcar que no se trata de un proceso de entrenamiento como tal, por el contrario, muchos de los modelos de la familia GPT son productos pre-entrenados y por lo pronto, el proceso *fine-tunning* de un modelo personalizado es una tarea compleja que requiere de un gasto de recursos de tiempo y dinero significativo cuando de bases de datos se trata.
- Permite ajustar el "razonamiento" del modelo dando indicaciones de cómo responder según que escenarios, que acciones tomar, si puede agregar creatividad a su respuesta o ceñirse al resultado de la consulta (Chase, 2023). Y es más personalizable en una arquitectura como la nuestra, debido a que tenemos acceso total a las indicaciones del modelo, pudiendo insertar información adicional, requerimientos específicos, etc.
- Ofrece diversas opciones para elegir cuando se habla de la variedad de los modelos, cada uno de ellos se ajusta a distintos escenarios, en una variedad de precios y capacidades adaptados a las demandas del usuario.

También es necesario destacar que esta solución es adaptable a la gran mayoría de aplicaciones modernas, sim importar la escala de estas, debido a que la interacción con estos modelos es por solicitudes simples a APIs públicas.

3.4 Implementación de la Integración al Sistema

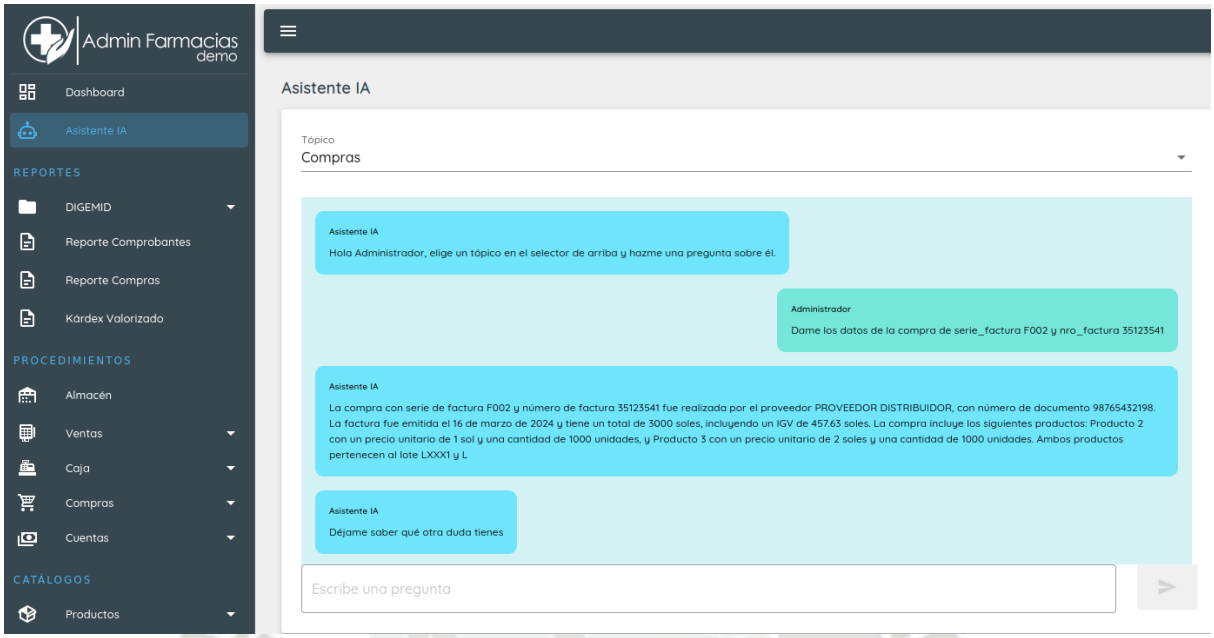
3.4.1 Diseño de la Interfaz del Chat

La interfaz del chat es simple y directa, los diferentes elementos de este se encuentran muy bien delimitados con el fin de ofrecer facilidad de uso a los usuarios, tomando como

referencia el diseño de interfaz minimalista que la mayoría de servicios de IA conversacional ofrecen.

Figura 26

Interfaz de chatbot en sistema de farmacias



Como se observa en **Figura 26**, la interfaz del chat está construida en una ventana completa y el módulo puede ser accedido desde el menú lateral del sistema, comprende los siguientes elementos:

- **Selector de tópicos:** este selector se carga con la información de tablas comprimidas provenientes de *compressed_tables*, observe **Figura 28**. Este elemento es usado para contextualizar la consulta del usuario, si el usuario elige el tópico compras y realiza una consulta sobre el stock de productos el chat responderá con la información solicitada, en cambio, si el usuario elige el tópico ventas pero realiza una consulta sobre productos comunicará que no puede procesar la consulta.
- **Ventana de chat:** es el contenedor principal del chat, incluye un fondo color azul oscuro y contiene elementos como la caja de texto de entrada, un botón de envío y las burbujas en el chat.

- **Caja de texto y botón de envío:** conformado por un *input* de texto simple.
- **Burbujas de texto:** contenedores que muestran el autor del mensaje (LLM o usuario), el contenido del mensaje y un color distintivo.

3.4.2 Lógica Interna del Algoritmo

Tomando como referencia la arquitectura presentada en la **Figura 25**, se construyeron cuatro componentes esenciales para la implementación de esta solución: la compresión de tablas de la base de datos, el modelo *CompressedTables*, el controlador de *GenAI* y las rutas *api* para acceder a las funciones.

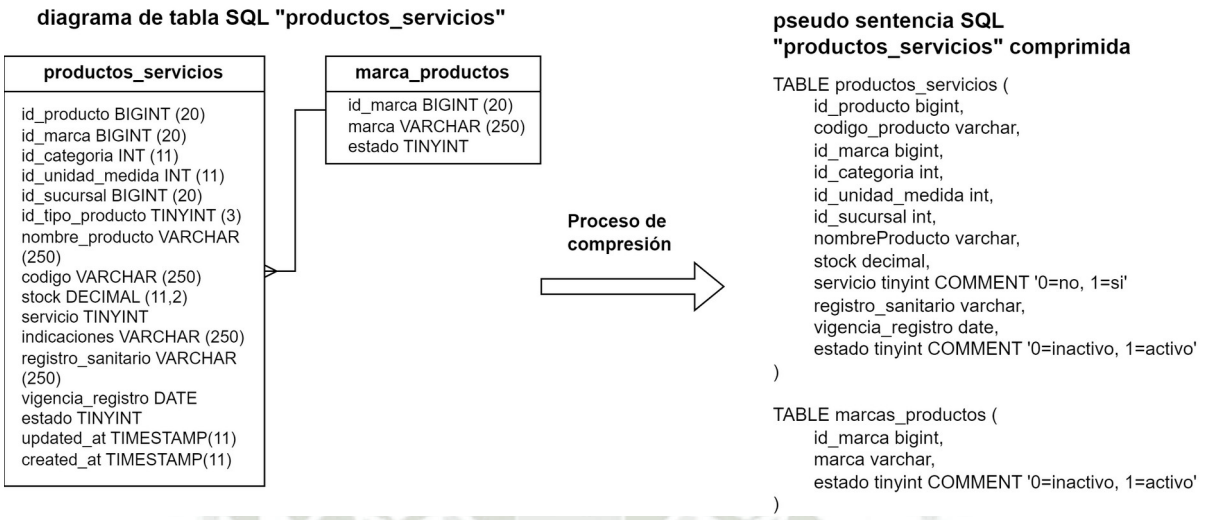
3.4.2.1 Compresión de Tablas.

Este proceso de compresión de tablas consistió en llevar a cabo una transformación de determinadas tablas de la base de datos a una versión resumida, esto con el fin de reducir la longitud del *prompt* enviado al LLM en el primer llamado **Figura 25**.

Por ejemplo, puede observar en **Figura 27** el proceso compresión hecho para la tabla *productos_servicios*. Este mismo proceso de compresión se aplicó a las tablas comprobantes y compras con sus respectivas relaciones.

Figura 27

Compresión de tabla Productos para su uso en la integración con el modelo de IA Generativa

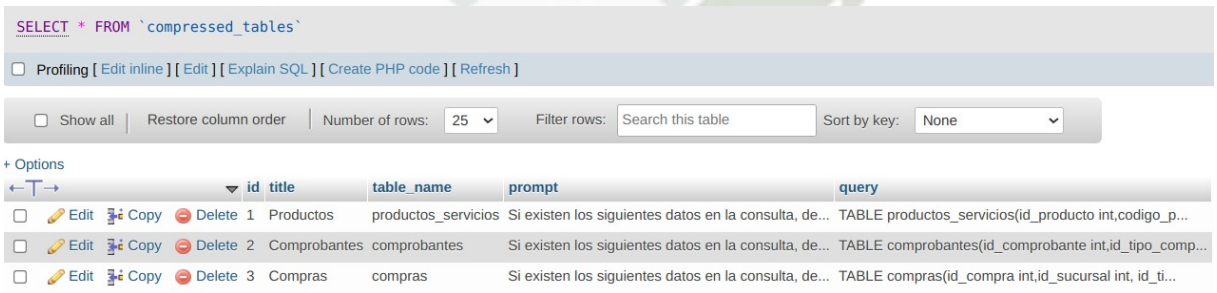


Nota. La tabla expresada como pseudo-sentencia SQL especificando campos, tipos, comentarios y relaciones con otras tablas. Algunos campos que no son de interés gpara la consulta se eliminaron y la longitud de los tipos de datos dejó de especificarse.

Una vez culminado el proceso de conversión, este dato es almacenado en la tabla *compressed_tables*, puede observar en **Figura 28**.

Figura 28

Estructura de tabla compressed_tables en base de datos MySQL.



```
SELECT * FROM `compressed_tables`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Restore column order | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	id	title	table_name	prompt	query
Edit Copy Delete	1	Productos	productos_servicios	Si existen los siguientes datos en la consulta, de...	TABLE productos_servicios(id_producto int,codigo_p...
Edit Copy Delete	2	Comprobantes	comprobantes	Si existen los siguientes datos en la consulta, de...	TABLE comprobantes(id_comprobante int,id_tipo_comp...
Edit Copy Delete	3	Compras	compras	Si existen los siguientes datos en la consulta, de...	TABLE compras(id_compra int,id_sucursal int, id_ti...

- **title.** Título del tópico, es el elemento visual que se muestra en el selector de la interfaz del chat.
- **table_name.** Nombre de la base de datos, se utiliza como código identificador y es el valor del selector de tópicos.
- **prompt.** Instrucción personalizada para tópico, especifica los campos de la tabla que deberán mostrarse y se adiciona al *prompt* enviado al LLM en el segundo

llamado **Figura 25**. Cumple el papel de limitar la información mostrada sobre los registros de las tablas.

- **query.** Almacena el resultado del proceso de compresión de cada tabla **Figura 27**. Se especifican sus datos, tipos y relaciones con otras tablas también comprimidas.

3.4.2.2 Modelo CompressedTable.

La recuperación de información de esta tabla se realiza mediante un componente modelo de Laravel, como se explica en el punto **2.4.4**, hacemos referencia a la tabla *compressed_tables*, listamos sus campos, relaciones y funciones adicionales de ser necesario. En la siguiente porción de código puede observar el contenido del archivo *CompressedTable.php*.

```
class CompressedTable extends Model
{
    // Nombre de Tabla y primary key
    protected $table = 'compressed_tables';
    protected $primaryKey = 'id';
    // Campos de la tabla
    protected $fillable = [
        'table_name',
        'prompt',
        'query',
    ];
    public $timestamps = false;
}
```

3.4.2.3 Controlador ChatbotController.

Este controlador fue creado en la ruta "*app/Http/Controller/API/Common/*", contiene la lógica necesaria para comunicar la aplicación con el LLM y retornar las respuestas que el usuario requiere en un lenguaje comprensible. El controlador incluye las siguientes dos funciones.

3.4.2.3.1 CompressedTablesCombo.

Es una función simple de una sola línea que recupera los registros de la tabla *compressed_tables* y los retorna a la vista del módulo de chat para ser seleccionado por el usuario, a modo de tópicos. Visualice el código a continuación:

```
public function compressedTablesCombo(){
     return response()->json(CompressedTable::all());
}
```

3.4.2.3.2 SendQuery.

Es la función principal de la interacción con el LLM, implementa la arquitectura de la solución diseñada en **Figura 25.**

1. Recibe como parámetro el tópico seleccionado y pregunta del usuario.
2. Utiliza esta información para recuperar información de la tabla *compressed_tables*.
3. Construye el *prompt* para el primer llamado al LLM mediante OpenAI API, este expresará la pregunta del usuario en una consulta SQL.
4. Verifica si la consulta es válida, de serlo, se ejecuta sobre la base de datos.
5. Construye un segundo *prompt* utilizando la pregunta del usuario, resultado de la consulta SQL y las instrucciones personalizadas de *compressed_tables*.
6. Procesa la cadena resultante del segundo llamado del LLM y retorna la respuesta.

```
$inputText = $request["text"];
$table = CompressedTable::where('table_name', $request["dbTable"])-
>first();
// 1. Primer llamado a modelo GenAI, se construye la consulta SQL en
base a la tabla seleccionada y pregunta del usuario.
$sqlQueryPrompt = "
     Dada la tabla: {$table->query}
     Genera una consulta MySQL a partir de: {$inputText}
";
$sqlQuery = OpenAI::completions()->create([
```

```
        'model' => 'gpt-3.5-turbo-instruct',
        'prompt' => $sqlQueryPrompt,
        'max_tokens' => 125,
        'temperature' => 0,
        'n'=> 2
]);
$sqlQuery = $sqlQuery['choices'][0]['text'];
$sqlQuery = trim(preg_replace('/\s\s+/', ' ', $sqlQuery));

// 2. Se detectan comandos con riesgo de SQL Injection
$forbidden_commands = ["INSERT", "UPDATE", "DELETE", "DROP"];
foreach ($forbidden_commands as $command){
        if(strpos($sqlQuery, $command) !== FALSE){
                return response()->json(['success'=>false, 'message'=>"Las
consultas que modifiquen o agreguen información a la base de datos no
están permitidas"], 500);
        }
}

// 3. Se ejecuta la consulta SQL generada por el modelo sobre la base
de datos
$queryResult = DB::select(DB::raw("$sqlQuery"));
$queryResult = json_encode($queryResult);

// 4. Segundo llamado a modelo GenAI, expresa la respuesta de la base
de datos en lenguaje natural.
$naturalLangPrompt = "
        Dada una consulta del usuario: {$inputText}\n
        Se retornó este resultado desde la base de datos:{$queryResult}\
n
```

```
        Devuelve el resultado en una muy corta respuesta en lenguaje
    natural, sin decir que proviene de una base de datos.
        {$table->prompt}
    ";
    $chatResponse = OpenAI::completions()->create([
        'model' => 'gpt-3.5-turbo-instruct',
        'prompt' => $naturalLangPrompt,
        'max_tokens' => 150,
        'temperature' => 0,
        'n'=> 2
    ]);
    $chatResponse = $chatResponse['choices'][0]['text'];
    $chatResponse = trim(preg_replace('/\s\s+/', ' ', $chatResponse));
    return response()->json(['success'=>true, 'message'=>$chatResponse]);
```

3.4.2.4 Declaración de las Rutas.

La declaración de las rutas, como se explicó en el punto **2.4.6** de este informe, son útiles para permitir acceder a las funciones del controlador mediante su inclusión en los archivos de rutas de Laravel. Para el caso específico de rutas de acceso a *GenAIController*, se establecieron como rutas API debido a que se requiere que pueda ser accedida de forma externa y limitar el número de llamadas por minuto a la función. A continuación, puede observar la declaración de estas rutas en el archivo *api.php*.

```
//--- Chatbot---
Route::post('genAI/SendQuery', [GenAIController::class, 'SendQuery']);
Route::get('compressedTablesCombo',[GenAIController::class,
'compressedTablesCombo']);
//--- End ---
```

3.5 Alcance y Limitaciones

Esta integración estuvo orientada a facilitar la obtención de información desde la base de datos mediante preguntas simples hechas por el usuario, sin embargo, no es posible hacer

uso del *chatbot* como normalmente se hace en la plataforma ChatGPT, es decir, las preguntas sueltas o fuera del contexto no tendrán un resultado.

Por otro lado, es necesario darle importancia a una serie de detalles que surgieron cuando el equipo de desarrollo indagó a fondo sobre tecnología, como los que se describen a continuación.

3.5.1 Entrenamiento y Contextualización

Para este proyecto se consideraron dos tipos de modelos dentro de la familia GPT: *fine-tuning* y *gpt-3.5 turbo*; la diferencia entre ambos reside en que los modelos *fine-tuning* sí pueden ser entrenados con información propia, mientras que los modelos *gpt-3.5 turbo* tan sólo pueden ser contextualizados. El motivo por el cual se desestimó el uso de modelos *fine-tuning* fue por el tiempo y costo de recursos que estos suponen. Tal y como se explica en la documentación de OpenAI, este tipo de modelos requieren de la creación de un *dataset* con preguntas y respuestas esperadas, y realizar esta tarea con la estructura de una base de datos es significativamente complejo además de requerir de mucho tiempo más del contemplado para el proyecto (OpenAI, 2022a).

3.5.2 Tokenización y Precio de Consultas

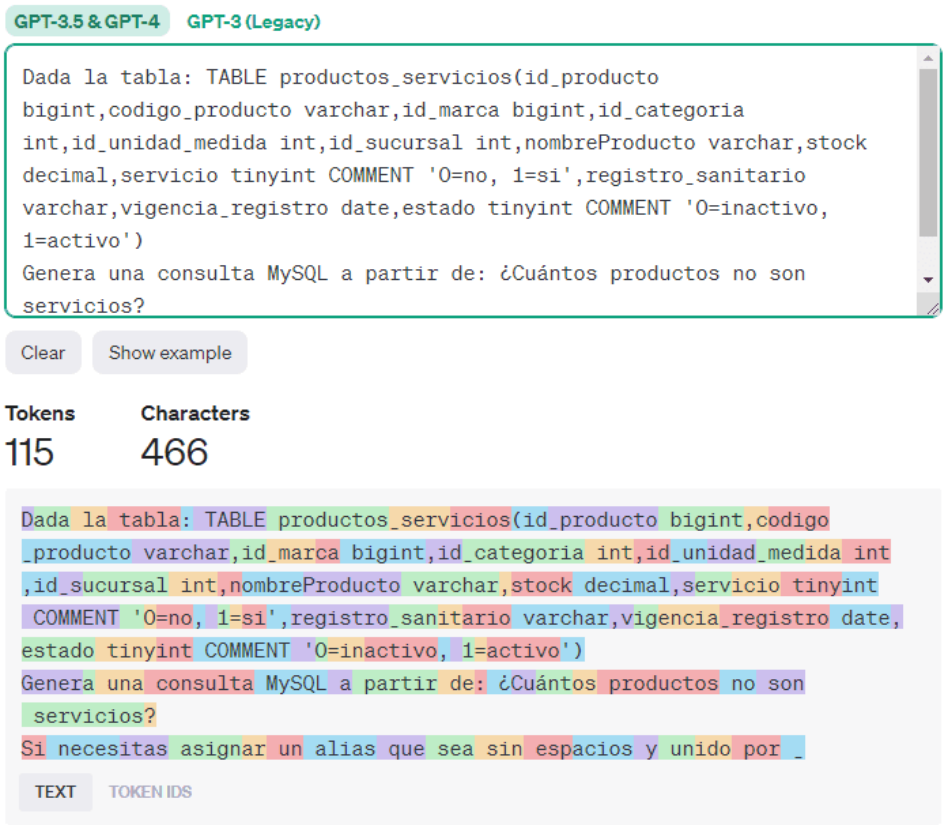
Los modelos de procesamiento de lenguaje natural utilizan un proceso de "tokenización" para dividir un texto en unidades más pequeñas, con el fin de facilitar su análisis y procesamiento (OpenAI, 2022a). Cada cierta cantidad de tokens enviados a OpenAI API tendrá asignado un precio dependiendo del modelo GPT elegido. Estos tokens serán calculados en base a la cantidad de caracteres de entrada (consultas de usuario) y salida (respuesta del modelo).

En este proyecto el modelo elegido fue *gpt-3.5-turbo-instruct*, el cual tiene por defecto un límite de 4097 tokens y un costo de $0.0020 cada mil tokens. Es pertinente recordar que la lista actualizada de precios de los modelos GPT las puede encontrar en **Anexo 12**.

Tomando en cuenta que la gran mayoría de solicitudes no requieren el contexto total del modelo relacional de la base de datos, fue ventajoso para el equipo de desarrollo y para el ahorro en costos utilizar versiones reducidas de esta. Es decir, para consultas sobre ventas se hace uso de tablas relacionadas con ventas únicamente, para consultas sobre productos se toman en cuenta sólo la tabla maestra y la misma lógica aplica para compras. En **Figura 29** puede observar el total de tokens que OpenAI calculó con una consulta de ejemplo.

Figura 29

OpenAI Tokenizer para el cálculo de una consulta a la tabla productos



CAPÍTULO IV

4 PRUEBAS Y VALIDACIÓN

4.1 Pruebas de la Aplicación

Crear pruebas en una aplicación basada en Laravel 9 implica abordar diferentes aspectos de la plataforma, desde pruebas de integración hasta pruebas de rendimiento y usabilidad. En los siguientes puntos se expone el proceso y los resultados de las distintas pruebas aplicadas sobre la presente aplicación.

4.1.1 Pruebas Unitarias

Mediante la ejecución de archivos test que Laravel pone a nuestra disposición, podemos programar pruebas automatizadas para una variedad de funciones del *backend* de la aplicación, esto ofrece la principal ventaja de ahorro en tiempo y esfuerzo. Existen dos tipos de pruebas automatizadas en Laravel: *Unit*, las cuales tienen un enfoque en pequeñas porciones individuales del código y *Feature*, que abarcan mayor cantidad elementos como llamadas HTTP, múltiples objetos, comunicación con base de datos, etc (Otwell, 2023). En este escenario, todas las pruebas automatizadas incluidas en este proyecto son del tipo *Feature*.

4.1.1.1 Generación de Documentos.

La aplicación ofrece la capacidad de generar documentos a partir de las operaciones realizadas, por ejemplo, documentos PDF para compras, órdenes de compras y cotizaciones; o reportes en formato Excel para el kárdex valorizado, comprobantes, compras, movimientos en almacén, etc.

Teniendo esto en cuenta, se creó el archivo DocsGenerationTest.php donde se definieron funciones que evaluaron la correcta generación de distintos tipos de documentos, más no el contenido de estos. Observe la siguiente porción de código.

```
public function test_caja_pdf(){
     $user = User::find(2);
     $this->actingAs($user);
```

```
        $mockMpdf = $this->createMock(Mpdf::class);
        $mockMpdf->method('WriteHTML');
        $mockMpdf->expects($this->once())->method('Output')->with($this-
>anything(), 'I');

        $controller = new DocGenerationController();
        $controller->generarCajaPDF(1, $mockMpdf);
}

public function test_kardex_export(){
        $user = User::find(2);
        $this->actingAs($user);
        $mockWriter = $this->createMock(Xlsx::class);
        $mockWriter->expects($this->once())
            ->method('save')
            ->with('php://output');
        $params = '{"fechaInicio":"2024-01-01","fechaFin":"2024-03-03",
"searchTerm":""}';
        $controller = new ExportsController();
        $controller->exportarKardexValorizado($params, $mockWriter);
}
```

En el código mostrado puede notar que las funciones de *test* tienen una estructura similar, primero se busca un usuario existente, utilizamos la función *actingAs()* para simular una sesión activa, luego se crea un *mock* el cual es un objeto simulado a partir de una clase determinada y, finalmente, se llama a la función sobre la que se requieren realizar pruebas.

- *test_caja_pdf* – verifica que el formato de documento pdf que hace referencia al cierre de una caja sea de la clase *Mpdf* y del tipo *Output 'I'*.
- *test_kardex_export* – verifica que el resultado de un reporte generado del kárdex valorizado sea de la clase *Xlsx* y el método *save*.

4.1.1.2 Operaciones CRUD de Módulo.

El mismo tipo de pruebas se utilizaron para realizar pruebas sobre las operaciones CRUD de los módulos, por ejemplo, pruebas sobre el listado, creación, actualización y eliminación de productos. Observe las dos siguientes funciones *test* de ejemplo sobre productos.

```
public function test_product_create(){
      $user = User::find(2);
      Passport::actingAs($user);
      $response = $this->post('/api/producto', [
          "data" => [
              "id_producto" => 4,
              "codigo_producto" => "PRODTEST",
              "nombreProducto" => "PRODTEST",
              "id_marca" => 2,
              "id_laboratorio" => 1,
              "id_tipo_producto" => 10,
              "ubicacion" => "Almacen",
          ],
          "lotes"=>[],
          "list_detail"=>[
              [
                  "id_lista_precio" => 1,
                  "precio_compra" => 1000,
                  "precio_venta" => 2000,
                  "unidades" => 1000,
              ]
          ]
      ]);
      $response->assertStatus(200);
}
```

```
public function test_product_delete(){
      $user = User::find(2);
      Passport::actingAs($user);
      $id_producto = 4;
      $response = $this->delete('/api/producto/'.$id_producto);
      $response->assertStatus(200);
}
```

Estas otras dos funciones fueron definidas dentro del archivo *CrudsTest.php* el cual se centra en realizar pruebas sobre las operaciones CRUD de módulos. En el primer ejemplo, el *test* se centra en verificar la correcta creación de un nuevo producto y para el segundo ejemplo, se pone a prueba la eliminación de un producto. Existen diferencias clave con respecto a las pruebas sobre generación de documentos:

- Las pruebas aquí se realizan sobre una ruta definida *"api/producto"*, la cual implementa el *middleware "auth:api"*, por este motivo la creación de una sesión activa se realiza mediante la función *Passport::actingAs()*.
- Las pruebas son realizadas sobre las rutas definidas en lugar de sobre las funciones directamente.
- Se utiliza la función *assertStatus()* para definir un código de respuesta esperado al ejecutar la función, en este caso el código 200 hace referencia a una solicitud completada con éxito.

4.1.1.3 Resultados de las Pruebas Unitarias.

Una vez definidas las pruebas unitarias para los distintos escenarios elegidos, podemos ejecutarlas de forma automatizada con tan sólo el comando *php artisan test*. Observe en **Figura 30**, al ejecutar esta instrucción obtenemos la lista de resultados de las pruebas, el estado de cada una y el tiempo total de ejecución. Puede notar que tanto las pruebas sobre el CRUD de productos como las pruebas hechas sobre la generación de documentos pasaron de

forma exitosa. En caso alguna de estas fallara, se mostraría aquí el estado de la función con una equis roja y también nos especificaría el tipo de error ocurrido.

Figura 30

Resultados de ejecución de pruebas unitarias automatizadas.

Nota. Al ejecutar el comando *php artisan test* obtendremos el resultado de todos los *tests* programados divididos por el archivo en el que fueron incluidos con la palabra PASS o FAIL según el resultado. Además de realizar el conteo total de pruebas y el tiempo.

4.1.2 Pruebas de Integración

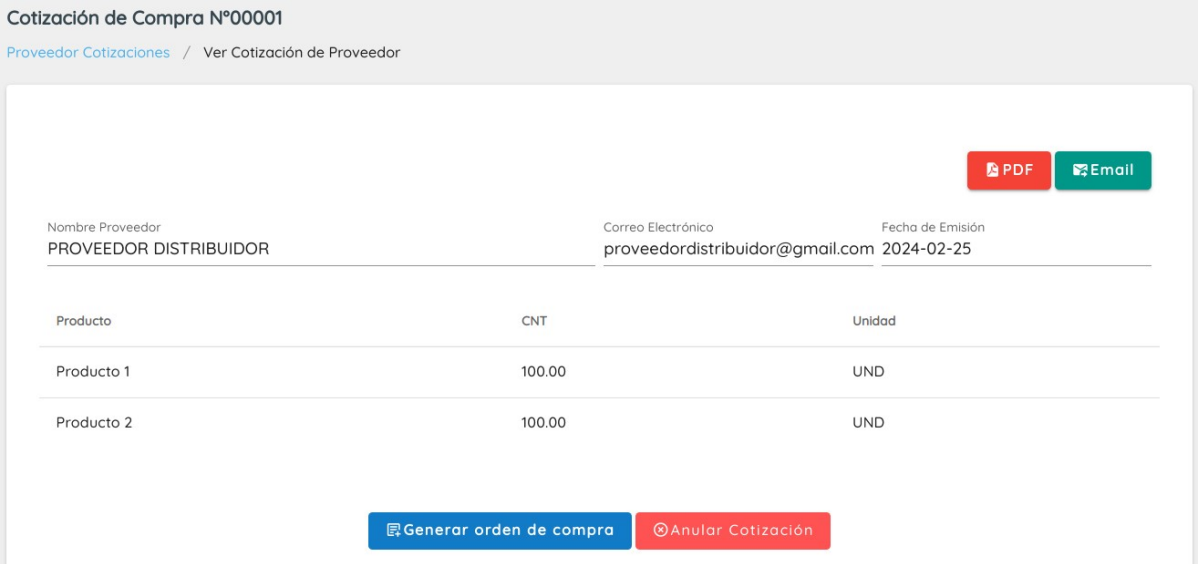
Las pruebas de integración fueron útiles para verificar la correcta interacción entre los módulos del sistema, cómo es que se integran para optimizar los procesos de negocio de una farmacia.

4.1.2.1 Proceso de Creación de Compras.

Como primer ejemplo, el proceso de “Compras” se divide en tres fases: creación de “Cotización”, generación de “Orden de Compra” y, finalmente, creación y ejecución de la Compra. Puede observar en **Figura 31** una cotización ya creada, podemos ver que se asocia a un proveedor, contiene una fecha de emisión y lista una serie de productos con sus respectivas cantidades y unidades. También se puede observar un botón para generar una “Orden de Compra”.

Figura 31

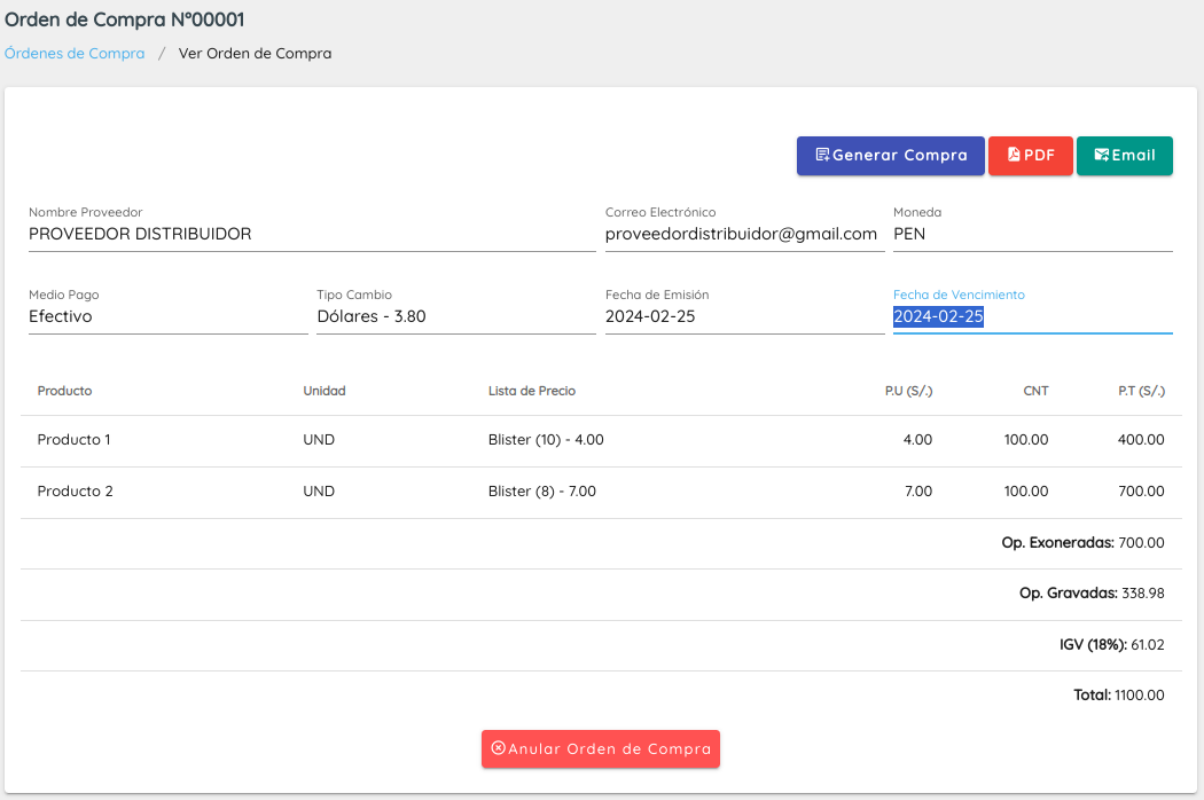
Cotización creada para futura compra a un proveedor



Al presionar este botón seremos dirigidos a la pantalla de creación de una "Orden de Compra" **Figura 32**, cabe resaltar que este tipo de documento sólo puede generarse a partir de una cotización y no de forma independiente. Ahora podemos observar más información como la moneda, medio de pago, tipo de cambio, fecha de emisión y fecha de vencimiento. Igualmente observamos en el detalle de la orden la lista de productos en conjunto con su precio de compra y totales. A partir de este documento podemos generar una compra utilizando el botón azul de la esquina derecha superior.

Figura 32

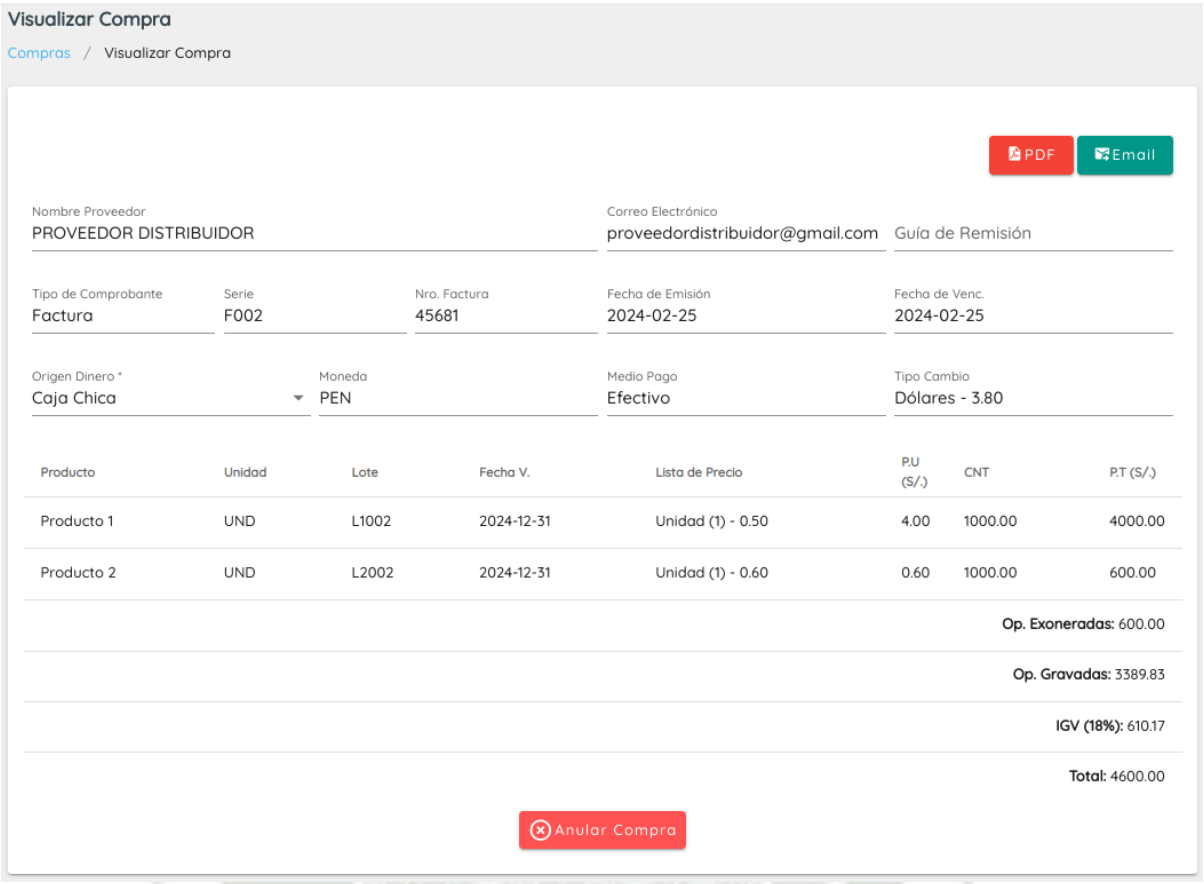
Orden de Compra generada a partir de una Cotización



Finalmente, nos dirigiremos a la pantalla de creación de “Compras”, en este caso estas pueden generarse a partir de una orden de compra o crearse de forma independiente. Puede notar en **Figura 33** que contamos con más datos a comparación de la orden, tanto en la cabecera como en el detalle; por ejemplo, se requiere el identificador de la factura de compra, opcionalmente la guía de remisión, en el detalle el nombre de cada lote adquirido, fecha de vencimiento, lista de precio y cantidad de unidades.

Figura 33

Compra generada a partir de una Orden de Compra



4.1.2.2 Productos y Listas de Precio.

Para la gestión de precios de productos se requieren guardar múltiples tipos de precio para un solo producto, por este motivo, en el módulo de creación de productos podemos editar y agregar “Listas de Precio” donde se especificará la lista, precio de compra, precio de venta y unidades del producto.

Figura 34

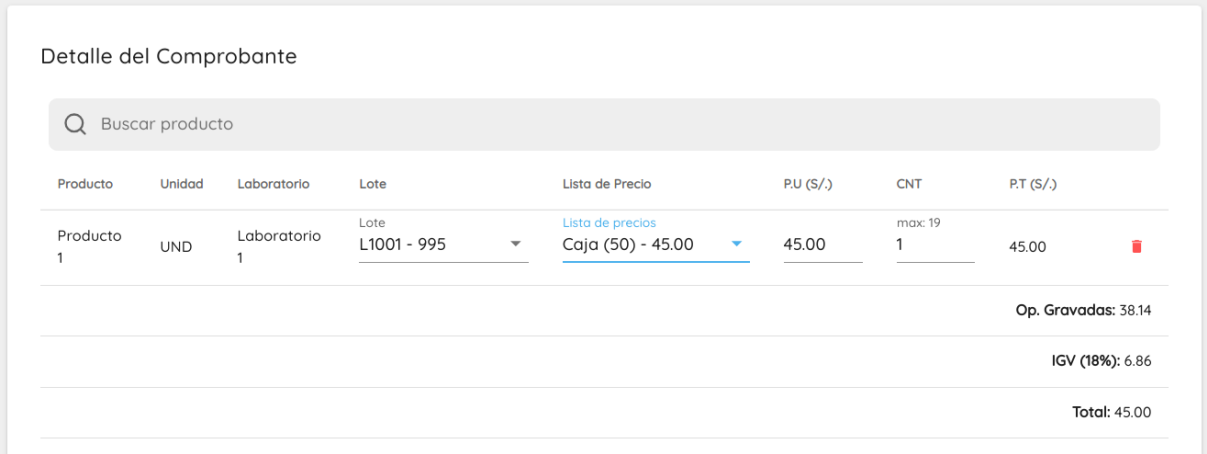
Creación de listas de precios para productos

La unidad de medida para cada lista de precio dependerá de las presentaciones de los productos: unidad, blíster o caja. Los distintos precios para cada producto serán visibles a lo largo del sistema, por ejemplo, anteriormente en compras el precio que visualizamos pertenece al “precio de compra” del producto.

Otro claro ejemplo del uso de listas de precio se da en la creación de un comprobante, observe **Figura 35**. Podemos notar que, en el detalle del comprobante, al agregar productos tenemos la posibilidad de elegir la lista de precio de venta para cada uno. A su vez, dinámicamente se cargará la cantidad establecida para cada unidad de medida; por ejemplo, para el caso de “producto 1”, elegimos la lista de precio de “Caja” donde cada una tiene 50 unidades y el costo es de S/. 45, podemos observar que en el campo “CNT” nos especifica el valor máximo 19, esto significa que sólo podemos adquirir esa cantidad de cajas ya que es el stock que tenemos actualmente.

Figura 35

Lista de precio para un producto en detalle de comprobante



4.1.3 Pruebas de Seguridad

4.1.3.1 Seguridad de la Aplicación.

Para medir la seguridad de la aplicación en general, se utilizó una herramienta especializada denominada “Enlightn”, la cual es capaz de analizar el código del *software*, compararlo con buenas prácticas específicas y agrupar estas pruebas en las categorías de: seguridad, rendimiento, fiabilidad y seguridad (Enlightn, 2020).

La prueba fue ejecutada mediante el comando *“php artisan englightn”* obteniendo como resultados los mostrados en **Tabla 12**, donde observará que el 71% de pruebas de seguridad fueron aprobadas satisfactoriamente; y siendo los aspectos desaprobados una serie de atributos configurados de una forma determinada para el correcto funcionamiento de la aplicación.

Tabla 12

Resultados de pruebas con Laravel Englightn sobre la aplicación

Estado	Rendimiento	Fiabilidad	Seguridad	Total
Aprobado	11 (61%)	27 (96%)	15 (71%)	53 (79%)
Fallido	4 (22%)	1 (4%)	4 (19%)	9 (13%)
No aplicable	3 (17%)	0 (0%)	2 (10%)	5 (7%)
Error	0(0%)	0 (0%)	0 (0%)	0 (0%)

Nota. Elaboración propia

Donde las evaluaciones más relevantes fueron las siguientes:

- Encriptación de cookies – Aprobado
- Archivo .env no público – Aprobado
- Configuración de PHP segura – Aprobado
- Dependencias actualizadas – Aprobado
- Dependencias con problemas de seguridad conocidos no incluidas – Aprobado

4.1.3.2 Roles y Permisos de Usuario.

Una de las medidas de seguridad más útiles en el manejo del sistema es la implementación de roles y permisos para usuarios, puede consultar la explicación técnica de este módulo en **2.4.8.** El principal objetivo de esta medida de seguridad es limitar el acceso a herramienta del sistema a los distintos tipos de usuario: administrador, cajero, almacén y contabilidad. Observe la siguiente entre **Figura 36** y **Figura 37**, puede notar que la vista "Sucursales" para los roles de administrador y cajero son distintas, donde cajero tan sólo puede visualizar la lista de sucursales mientras que el administrador puede realizar todas las funciones CRUD sobre esta tabla.

Figura 36

Vista sucursales para usuarios con rol de administrador

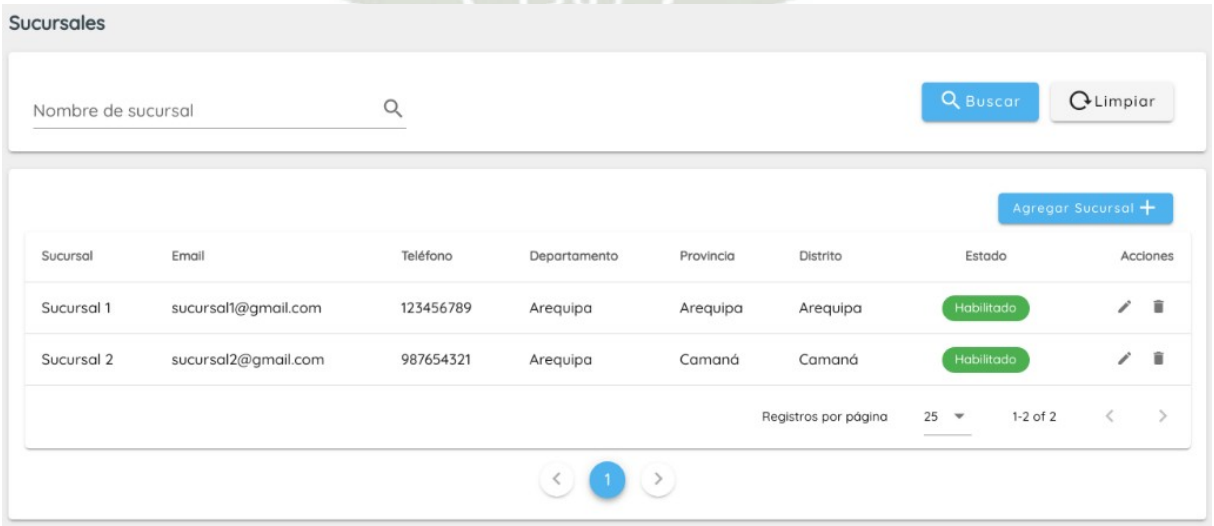
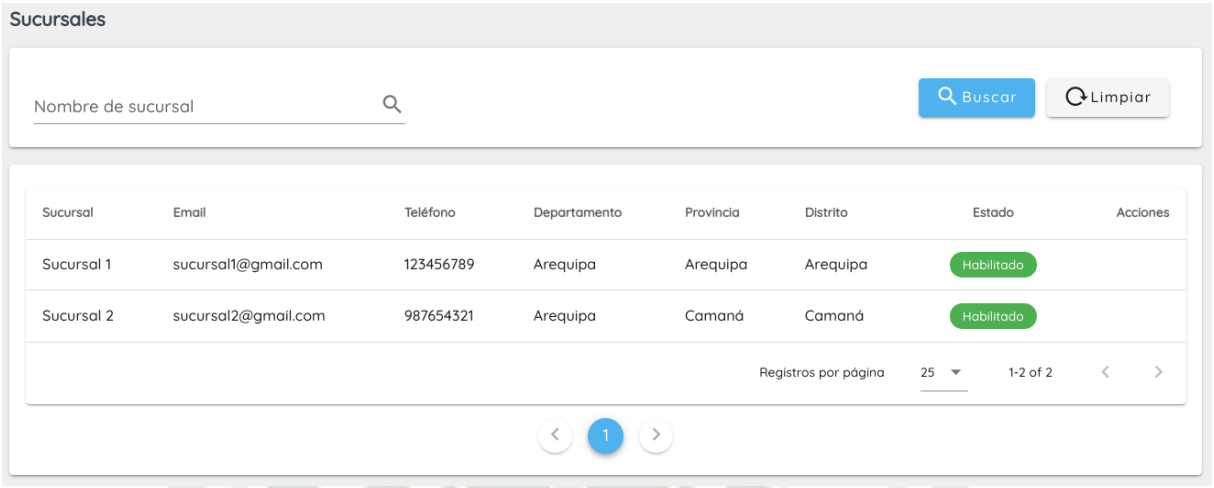


Figura 37

Vista sucursales para usuarios con rol de cajero



Otro ejemplo se da en la vista “Apertura de Caja”, observe en **Figura 38**, el usuario con rol de almacén puede visualizar el histórico de cajas e incluso ver el detalle de cada registro desde el sistema o en formato pdf; sin embargo, este usuario no tiene los permisos necesarios para abrir o cerrar una caja. Por el contrario, puede ver en **Figura 39** que el usuario con rol cajero, además de poder consultar el detalle de cada caja, tiene controles adicionales sobre estas.

Figura 38

Comparación vista apertura de caja para usuarios con rol de almacén

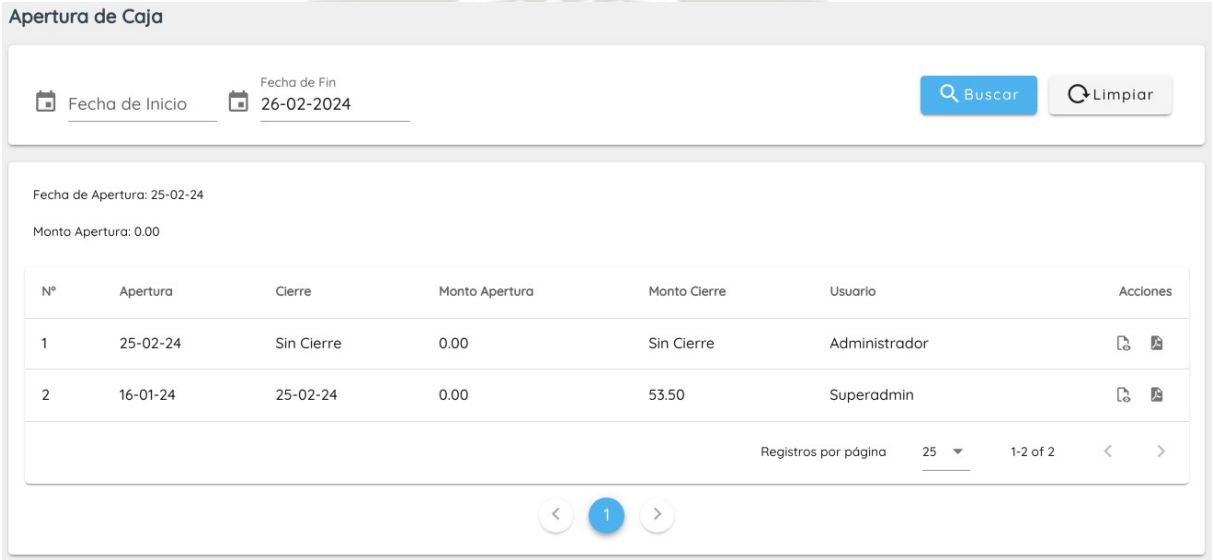
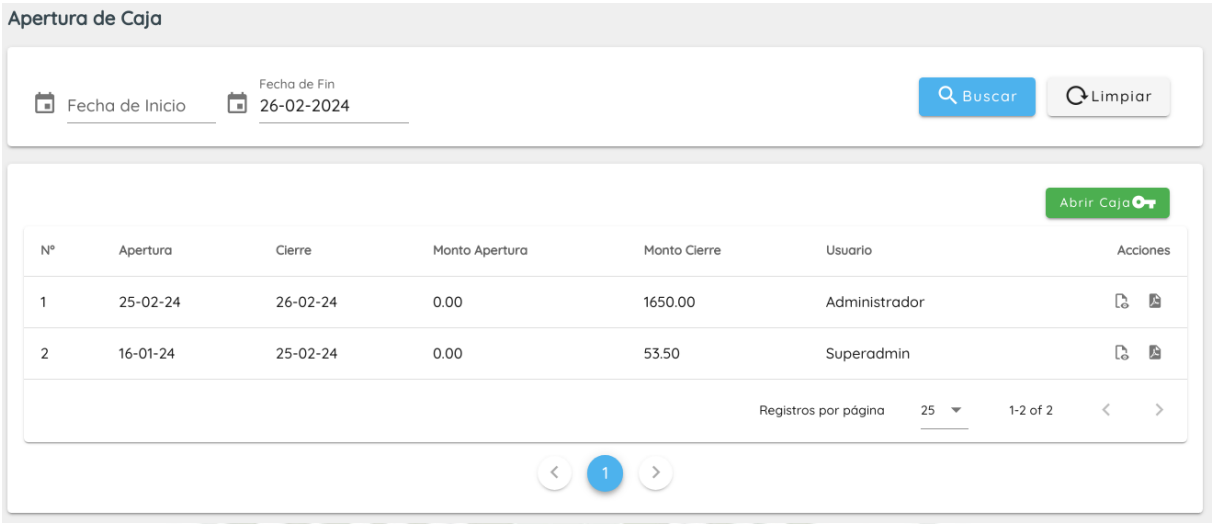


Figura 39

Comparación vista apertura de caja para usuarios con rol de cajero



4.1.4 Pruebas de Rendimiento

4.1.4.1 Tiempo de Ejecución de Consultas CRUD.

Para medir el rendimiento de la aplicación fue necesario registrar el tiempo de ejecución de distintas funciones con una variedad de carga de datos, es decir, agregar múltiples productos al detalle de compras, ventas, agregar listas de precios a productos, creación de lotes en la pantalla de productos, etc. Se registró el tiempo en tres distintas ocasiones para cada función considerada, puede observar los resultados en **Tabla 13** y la representación gráfica de estos en **Figura 40**. Como puede notar, las funciones CRUD de un módulo complejo como lo es “Productos”, en promedio no tardan más de un segundo en ejecutarse, lo cual se traduce en un procesamiento optimizado de los datos. Para el caso de procesos de “Compras” y “Ventas”, el tiempo de ejecución varía entre uno y dos segundos debido a la complejidad de procesamiento de datos que se debe hacer en el *backend* de la aplicación.

Tabla 13

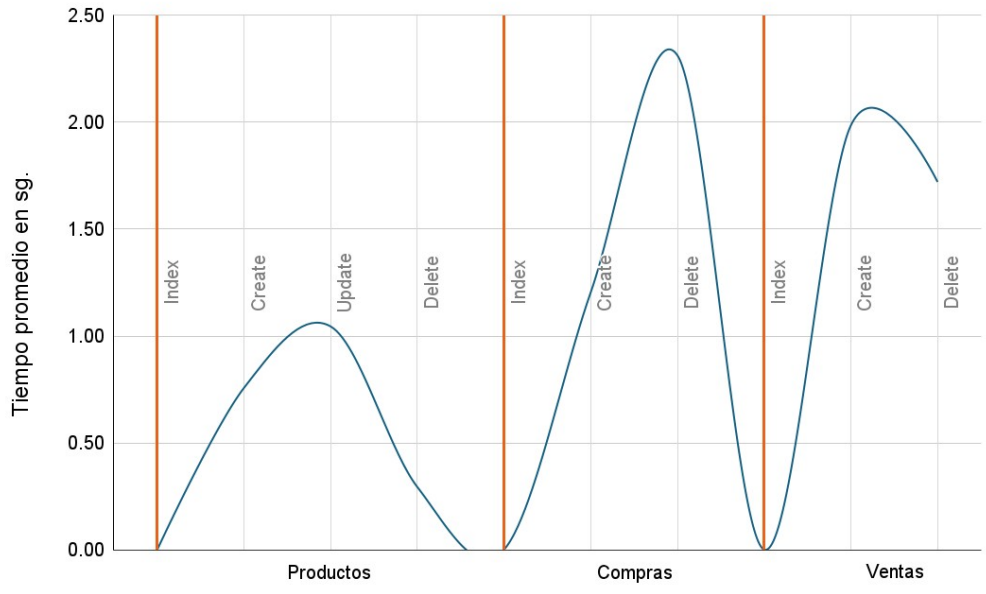
Resultados de pruebas de rendimiento sobre funciones CRUD

	Función	Test 1	Test 2	Test 3	Promedio
Productos	*Index*	0,000240	0,000217	0,000232	0,000230
	Create	0,960271	0,655427	0,656085	0,757261
	Update	1,020489	1,021617	1,092186	1,044764
	Delete	0,282680	0,304655	0,292707	0,293347
Compras	*Index*	0,000564	0,000804	0,000814	0,000727
	Create	1,141416	1,146354	1,326577	1,204782
	Delete	2,580567	2,244757	2,113918	2,313081
Ventas	*Index*	0,000716	0,000581	0,000542	0,000613
	Create	1,725157	1,746057	2,484855	1,985356
	Delete	1,293694	1,346954	2,522026	1,720891

Nota. Elaboración propia.

Figura 40

Promedio de pruebas de rendimiento sobre funciones CRUD



4.1.4.2 Tiempo de Generación de Documentos.

Al igual que en el punto anterior, se tomó el tiempo de generación de documentos en tres distintas ocasiones para cada una de las funciones. Se consideró incluir complejidad en la

construcción de cada documento agregando más registros, rangos de fecha, términos de búsqueda. Puede observar los resultados en **Tabla 14** y la representación gráfica de estos en **Figura 41**, notará que el rendimiento de cada función es mucho más optimizado a comparación de funciones que interactúan de forma más directa con la base de datos, a pesar de que se realizan acciones como *joins* de tablas, búsquedas de términos, filtros por fecha.

Tabla 14

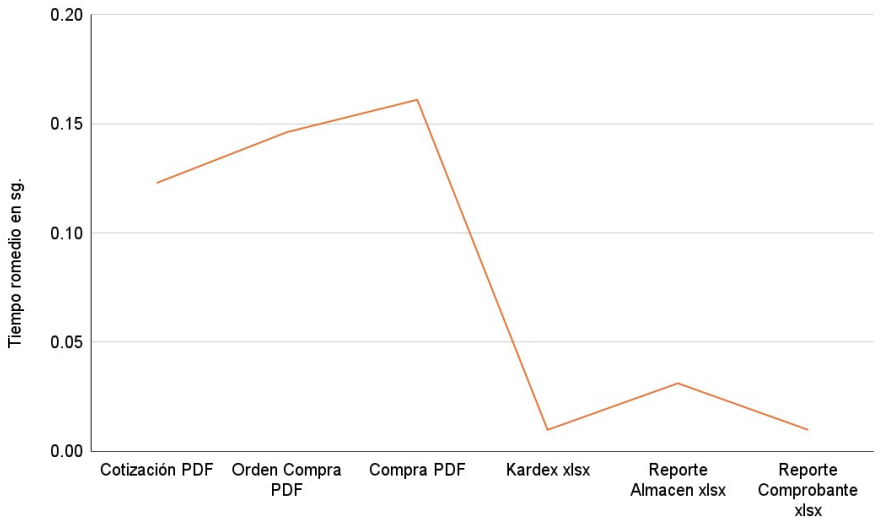
Resultados de pruebas de rendimiento sobre generación de documentos

Funciones	Test 1	Test 2	Test 3	Promedio
Cotización PDF	0,128554	0,116995	0,123120	0,122890
Orden Compra PDF	0,140760	0,160157	0,137704	0,146207
Compra PDF	0,181329	0,141014	0,160798	0,161047
Kardex Xlsx	0,009818	0,009282	0,010148	0,009749
Reporte Almacen Xlsx	0,031752	0,029318	0,032265	0,031112
Reporte Comprobante Xlsx	0,009936	0,010320	0,008985	0,009747

Nota. Elaboración propia.

Figura 41

Promedio de pruebas de rendimiento sobre generación de documentos



En conclusión, una vez realizadas las pruebas de rendimiento para las distintas funciones dentro de nuestra aplicación, notamos que los tiempos de ejecución para funciones

en las que se interactúa con la base de datos son mayores que aquellas operaciones de generación de documentos. Podemos asegurar que el rendimiento general es óptimo, con tiempos de ejecución generalmente menores a un segundo, lo que garantiza una interacción fluida con el usuario.

4.1.5 Pruebas de Usabilidad

Para validar la usabilidad del sistema se realizó una encuesta a 26 usuarios del sistema, estos se encontraban entre los 20 y 35 años al momento de realizar la encuesta, además de estar familiarizados con las funciones del sistema de farmacias, debido a que los participantes pertenecían a áreas de ventas, compras, contabilidad y administración. La encuesta original puede ser encontrada en **Anexo 7**, los resultados registrados a partir de esta encuesta pueden ser encontrados en **Anexo 8**.

Se evaluaron cinco aspectos de la aplicación: diseño intuitivo, coherencia de elementos visuales, atractivo visual, acceso a la información y nivel de aporte de los modelos de IA Generativa en la usabilidad del sistema. La encuesta comenzó con un breve video demostración, donde se presentan las capacidades principales del sistema: creación/edición de producto, visualización de compra, creación de comprobante, movimientos en almacén y el uso del módulo de *chatbot*. Se formularon cinco preguntas, una por cada aspecto de usabilidad considerado, finalizando con una caja de texto donde los usuarios pudieron expresar sus recomendaciones de forma voluntaria. Detallamos cada aspecto considerado para la usabilidad de la aplicación a continuación.

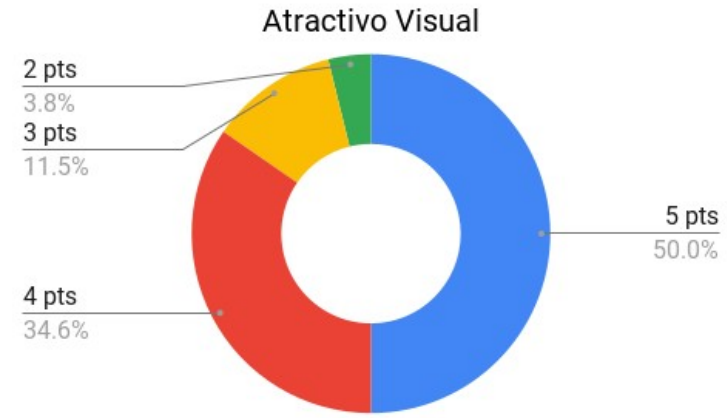
4.1.5.1 Atractivo Visual.

El aspecto de atractivo visual se evaluó mediante la pregunta *"¿Considera que la aplicación es visualmente atractiva?"*. El 50% de usuarios puntuó con la nota más alta este aspecto, un 34% con 4 puntos y un 3.8% con 2 puntos apenas, puede visualizar el gráfico a continuación en **Figura 42**. Podemos concluir que es un aspecto evaluado de forma positiva

en su mayoría, pero a la vez, es donde se obtuvieron mayor variedad de respuestas, lo cual podría indicar que requiere un mayor trabajo.

Figura 42

Resultados del aspecto "Atractivo Visual" de la encuesta de usabilidad

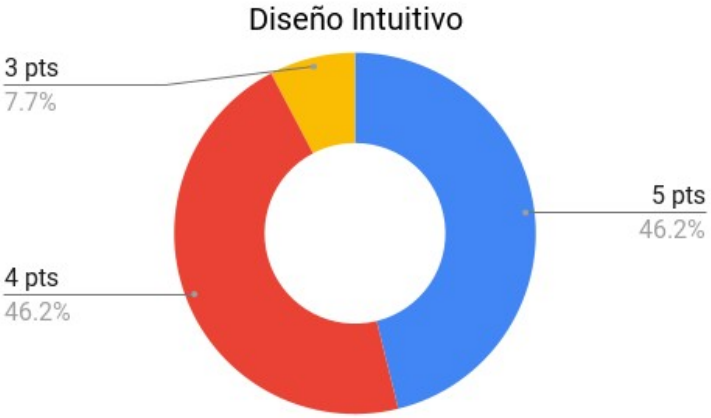


4.1.5.2 Diseño Intuitivo.

El aspecto de diseño intuitivo se evaluó mediante la pregunta *"¿Es el diseño de la aplicación intuitivo y fácil de entender?"*. El 46,2% de usuarios puntuó con la nota más alta este aspecto, otro 46,2% con 4 puntos y un 7.7% con 3 puntos, puede visualizar el gráfico a continuación en **Figura 43**. Con estos resultados concluimos que es un aspecto evaluado de forma positiva en su mayoría, debido a que la satisfacción del usuario es alta, las mejoras que se deben implementar son pequeñas y de poca urgencia.

Figura 43

Resultados del aspecto "Diseño Intuitivo" de la encuesta de usabilidad

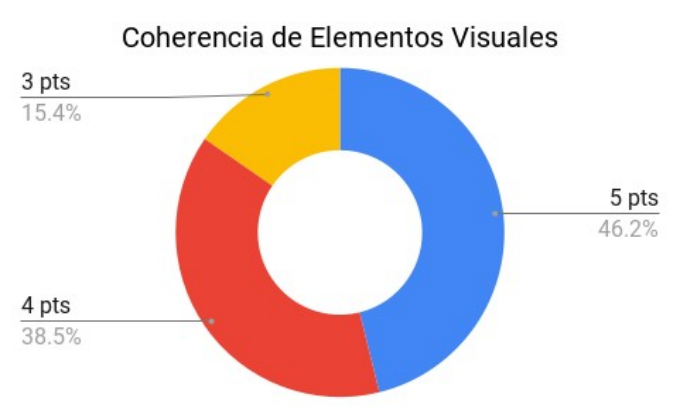


4.1.5.3 Coherencia de Elementos Visuales.

El aspecto de coherencia de elementos visuales se evaluó mediante la pregunta *"¿Cómo calificaría la coherencia de elementos visuales? (ubicación de menús, botones, colores, divisiones de vistas de usuario, etc)"*. El 46,2% de usuarios puntuó con la nota más alta este aspecto, el 38,5% con 4 puntos y un 15.4% con 3 puntos, puede visualizar el gráfico a continuación en **Figura 44**. Podemos apreciar que la evaluación demostró un puntaje alto para este aspecto, lo cual se traduce en una satisfacción alta del usuario y las mejoras que se realicen a partir de ahora serán pequeñas modificaciones.

Figura 44

Resultados del aspecto "Coherencia de Elementos Visuales" de la encuesta de usabilidad



4.1.5.4 Acceso a la Información.

El aspecto de acceso a la información se evaluó mediante la pregunta *"¿Considera que es fácil encontrar la información que se busca dentro de la aplicación?"*. En este caso el 53,8% de usuarios calificaron con un puntaje de 4 este aspecto, 42,3% calificó con 5 puntos y tan sólo un 3.8% con 3 puntos, puede visualizar el gráfico a continuación en **Figura 45**. Podemos deducir que el acceso a la información bueno pero mejorable, tanto dentro del sistema como también otorgando recursos de orientación o capacitaciones al usuario.

Figura 45

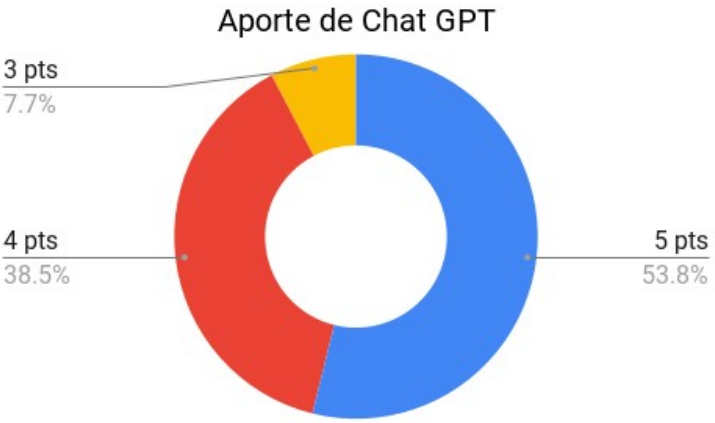
Resultados del aspecto "Acceso a la Información" de la encuesta de usabilidad

4.1.5.5 Aporte de IA Generativa.

El aspecto del aporte del módulo del *chatbot* conversacional se evaluó mediante la pregunta *"¿Cómo calificaría el grado de aporte a la usabilidad de la aplicación la implementación del módulo 'Chatbot Gen AI?"*. Donde el 53,8% de usuarios calificó este aspecto con 5 puntos, 38,5% con 4 puntos y tan sólo un 7,7% con 3 puntos. Puede visualizar el gráfico a continuación en **Figura 46**. Concluimos que el aporte de la IA Generativa es significativo, sin embargo, al encontrarse en una etapa temprana es posible que los usuarios no perciban los beneficios que una tecnología como esta ofrece, ni el abanico de oportunidades que podrían implementarse a futuro.

Figura 46

Resultados del aspecto "Aporte de IA Generativa" de la encuesta de usabilidad



4.1.5.6 Recomendaciones de Usuarios.

Finalmente, en una pregunta opcional se pidieron recomendaciones que mejoren la usabilidad de la aplicación, aquí fue posible obtener retroalimentación de los usuarios. Las recomendaciones más frecuentes o que más impacto podrían tener fueron las siguientes.

1. Envío de informe mensual automático al correo de administrador, con el fin de automatizar procesos de reportes sin necesidad de generarlos manualmente.
2. Opciones de personalización para el usuario para poder modificar la ubicación del menú, tamaño de la fuente o idioma. Los usuarios consideran que esto aportaría a la ergonomía y accesibilidad de la aplicación.
3. Una ventana de instrucciones para el uso del módulo Chatbot Gen AI, esto con el fin de que tanto usuarios con experiencia como sin experiencia puedan hacer uso de esta herramienta.
4. Integrar la ventana del *chatbot* como una opción disponible en cualquier lugar de la plataforma, es decir en modalidad *widget,* y no como una vista separada de la aplicación.
5. Normalización de colores, ubicación de íconos de botones etiqueta asignada a los datos mostrados.

4.2 Pruebas Módulo Chatbot Gen AI

Los integrantes de la familia GPT son, hoy en día, los modelos de procesamiento de lenguaje natural más confiables; no obstante, son falibles. Muchas de las consultas que los usuarios realicen obtendrán respuestas no del todo acertadas o con información que no fue la que solicitaron. Sin embargo, las fallas que podrían ocurrir en un enfoque como el de este proyecto se lograron mitigar gracias a la contextualización de nuestros parámetros.

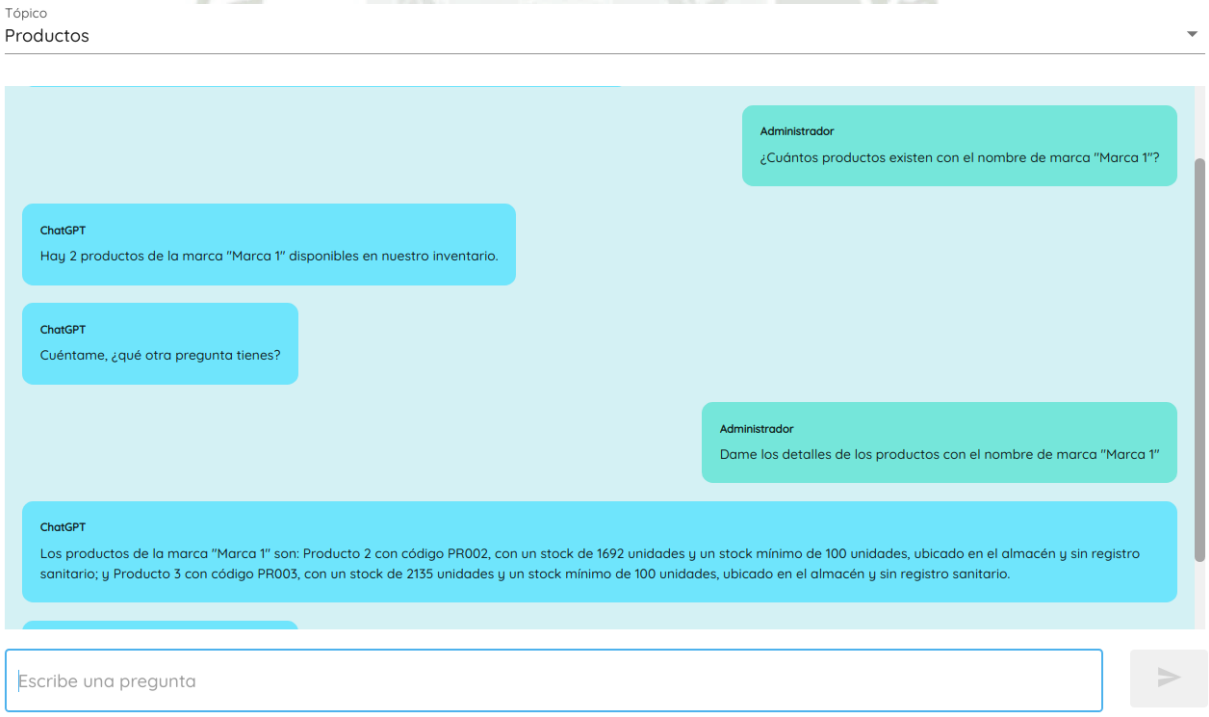
4.2.1 Pruebas de Seguridad

4.2.1.1 Acceso a Información Interna.

Debido a que la arquitectura del *chatbot* interactúa directamente con la base de datos, consulte en **3.3.1**, fue imprescindible limitar el acceso a la información que una solución como ésta tendría. Por este motivo, fue conveniente incluir el selector de tópicos, esto obliga al usuario de cierta forma a realizar consultas relacionadas a un solo tema a la vez. Por ejemplo, en **Figura 47**, podemos observar que la tabla "Productos" fue seleccionada, y se realizan consultas sobre los productos que no son servicios. También puede notar que la cantidad de información mostrada por cada producto es acotada, esto gracias al *prompt* personalizado relacionado con el tópico "Productos", consulte **Figura 28**.

Figura 47

Interacción con chat exitosa mediante selector de tópicos.

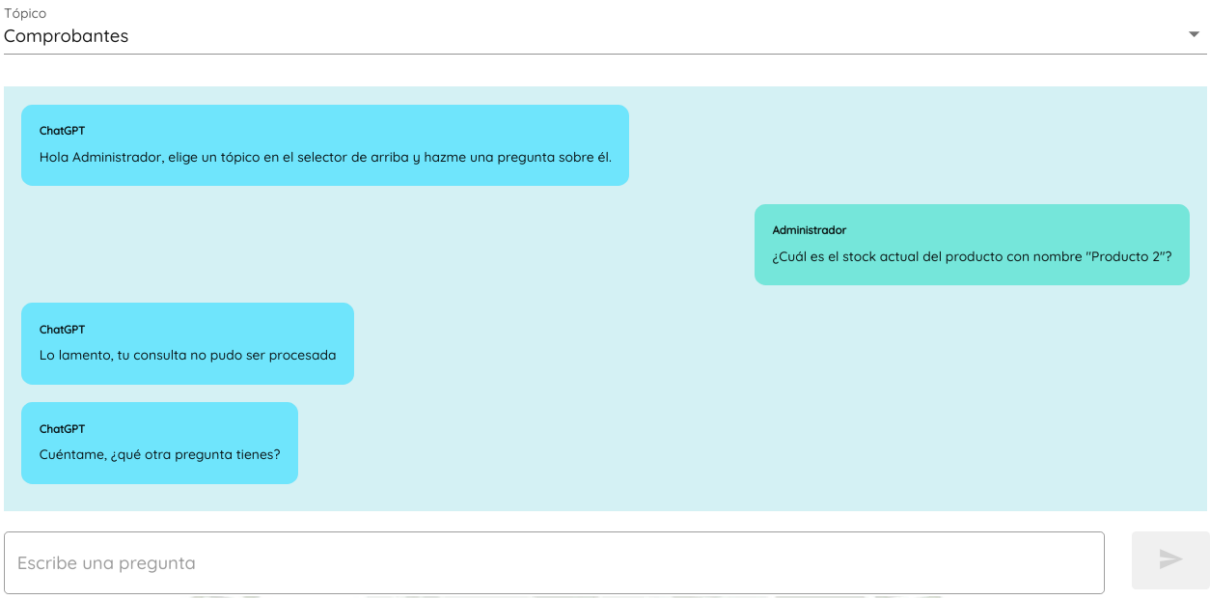


Otro ejemplo sobre las restricción de las consultas según el tópico se hace notorio cuando realizamos una consulta a un tópico equivocado. Observe **Figura 47**, puede notar que se seleccionó el tópico "Comprobantes", pero la consulta fue sobre el tópico "Productos" lo que lleva al *bot* a no retornar una respuesta, por el contrario, le comunica al usuario que no

puede procesar la consulta. Esto debido a que internamente, la conversión de la pregunta a una consulta SQL no pudo ser ejecutada sobre la base de datos.

Figura 48

Interacción con chat fallida mediante selector de tópicos.



4.2.1.2 Riesgo de SQL Injection.

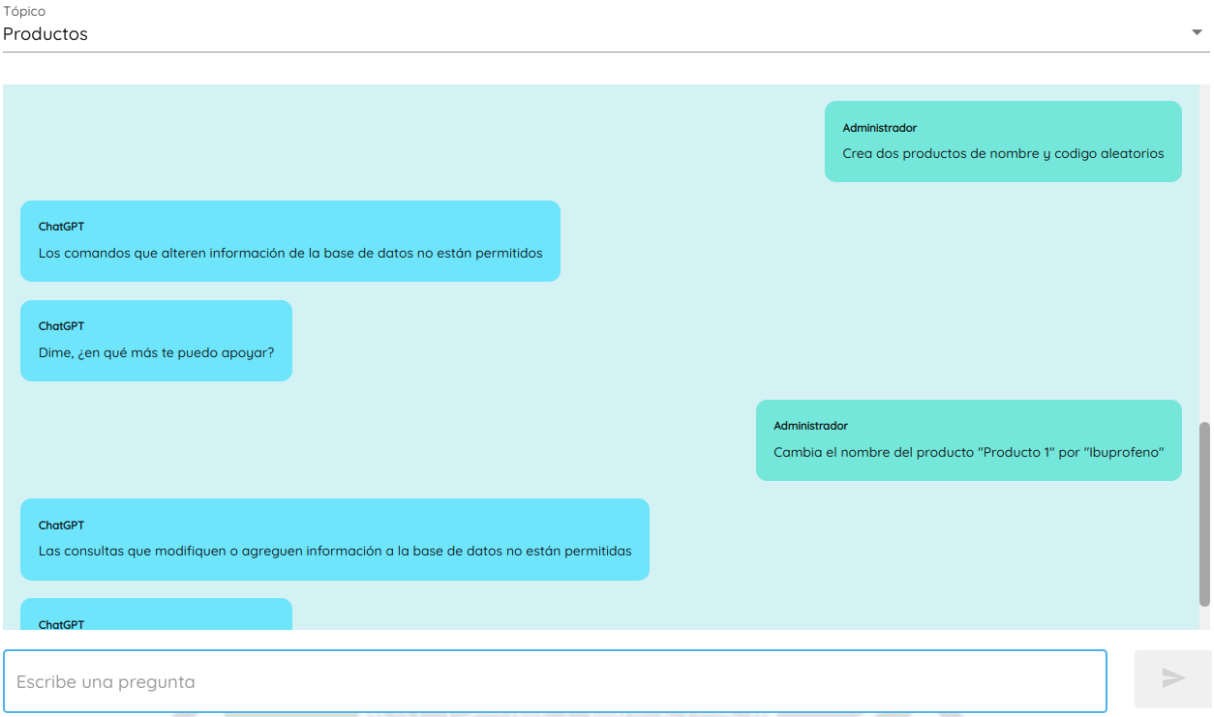
Para evitar comandos que alteren la información de la base de datos o accedan a más información de la esperada, implementamos dos enfoques. El primero, explicado en el anterior punto, fue dividir el chat por tópicos que accederán a una versión muy reducida de la base de datos. El segundo enfoque se centró en detectar comandos que alteren la información de la base de datos, la solución más directa fue implementar una validación donde se buscan instrucciones como INSERT, UPDATE, DROP o DELETE antes de ejecutar la consulta SQL sobre la base de datos.

Por ejemplo, como se observa en **Figura 49**, la primera solicitud del usuario es crear dos nuevos productos con nombre y código aleatorios, sin embargo, el *bot* responde con un mensaje de advertencia comunicando que ese tipo de consultas no están permitidas. En la segunda interacción, el usuario pide modificar el nombre de un producto, pero nuevamente

esta consulta es rechazada. Ninguna de estas solicitudes logró su objetivo debido a que se detectaron los comandos INSERT y UPDATE.

Figura 49

Interacción con chat no permitida debido a detección de SQL Injection.



4.2.2 Pruebas de Fiabilidad de la Información

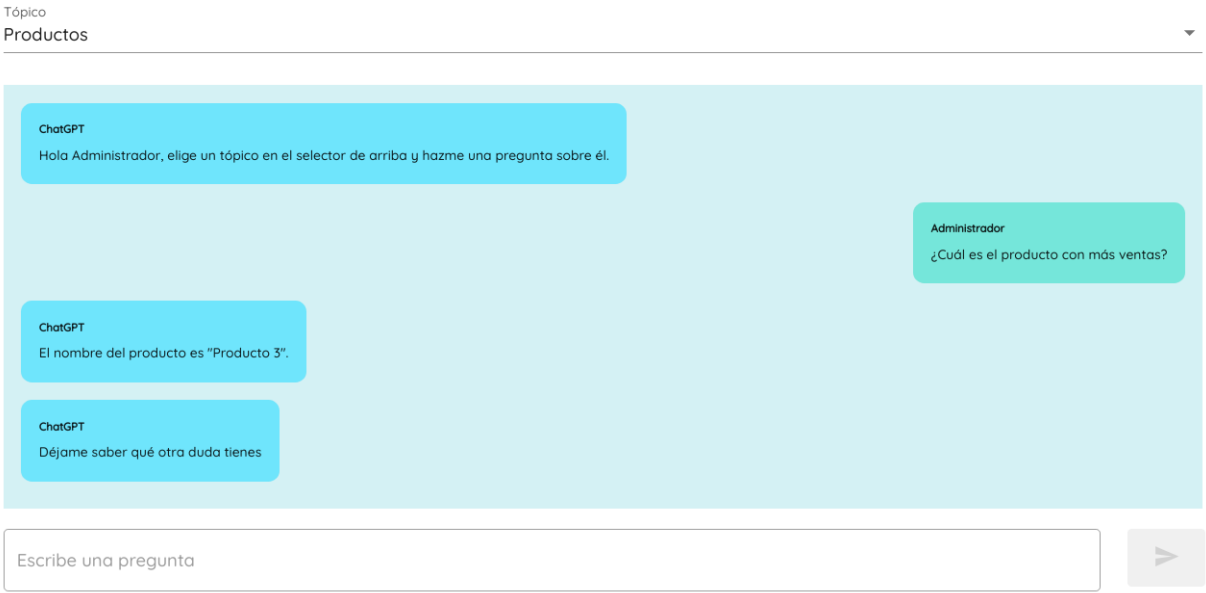
4.2.2.1 Respuestas del Bot vs Información en Base de Datos.

Es común que los modelos de lenguaje disponibles actualmente brinden información no del todo acertada, inventen algún dato o argumenten más de lo debido al realizar una consulta. Debido a que nuestra arquitectura propuesta utiliza una versión comprimida de la base de datos como fuente de información para abordar las consultas, el riesgo de que los datos retornados en la respuesta sean inventados es mínimo. Sin embargo, es posible que el modelo malinterprete la pregunta del usuario, lo que puede conllevar obtener respuestas equivocadas con datos reales.

Observe un ejemplo sobre una respuesta con datos fuera de lugar en **Figura 50**, note que el tópico “Productos” fue seleccionado, sin embargo, se consulta sobre el producto con más ventas. El chat devolvió como respuesta que el producto “Producto 3” es el más vendido.

Figura 50

Respuesta del chat equivocada al malinterpretar la pregunta del usuario.



Si investigamos lo sucedido detrás de escena, podremos obtener la consulta SQL generada para esta pregunta. Notará que en realidad la consulta generada ordena los productos de mayor stock a menor stock, lo cual es completamente erróneo si lo que se quiere recuperar es el producto más vendido.

```
SELECT nombreProducto FROM productos_servicios ORDER BY stock DESC LIMIT 1;
```

4.2.2.2 Consultas Válidas No Procesadas.

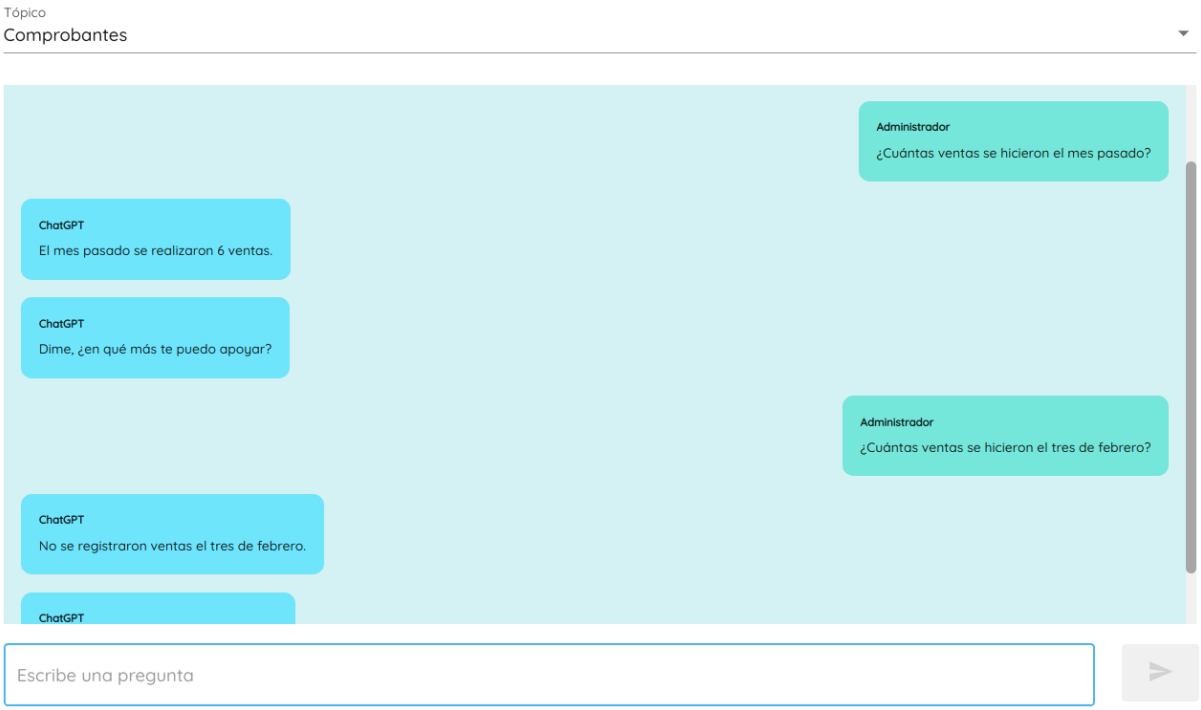
También existe la posibilidad que consultas válidas no sean correctamente procesadas. Esto es debido a que actualmente estos modelos, a pesar de contar con la especificación de una tarea de forma detallada, pueden malinterpretar la solicitud y el contexto dado en ciertas ocasiones. Para reducir el riesgo de este tipo de situaciones es necesario trabajar en el *prompt* brindado al modelo, sin embargo, no está totalmente garantizado que estos casos se solucionen en su totalidad.

Como ejemplo, podemos ver en **Figura 51** que se pregunta por las ventas hechas el mes pasado (febrero 2024), el chat responde satisfactoriamente. La segunda consulta es por

una fecha en específico, ventas hechas el tres de febrero del mismo año, sin embargo, en este caso el modelo expresa que no existieron ventas en ese día, cuando en realidad sí.

Figura 51

Interacción con chat válida pero no contestada



4.3 Juicio de Expertos

Para validar el estudio realizado, metodología, herramientas empleadas, diagramas, arquitectura de la solución, pruebas y calidad de la misma fue necesario realizar un cuestionario a expertos cercanos a los temas involucrados en el proyecto: soluciones de TI impulsadas por IA generativa.

4.3.1 Perfil de los Expertos Encuestados

Los 6 expertos consultados contaban con experiencia en proyectos de TI desde los 5 años hasta más de 15 años. Entre los cargos que los expertos ocupaban se encontraban los roles de *data scientist*, gerente de TI, *Chief Strategy Officer* y asesor técnico *senior.* Además, se recogió información sobre áreas de especialización en distintos proyectos como infraestructura de TI y Blockchain. Observe la información sobre el tiempo de experiencia en

proyectos de los expertos en **Figura 52** y las áreas generales de especialización de los mismos **Figura 53**.

Figura 52

Años de experiencia de expertos

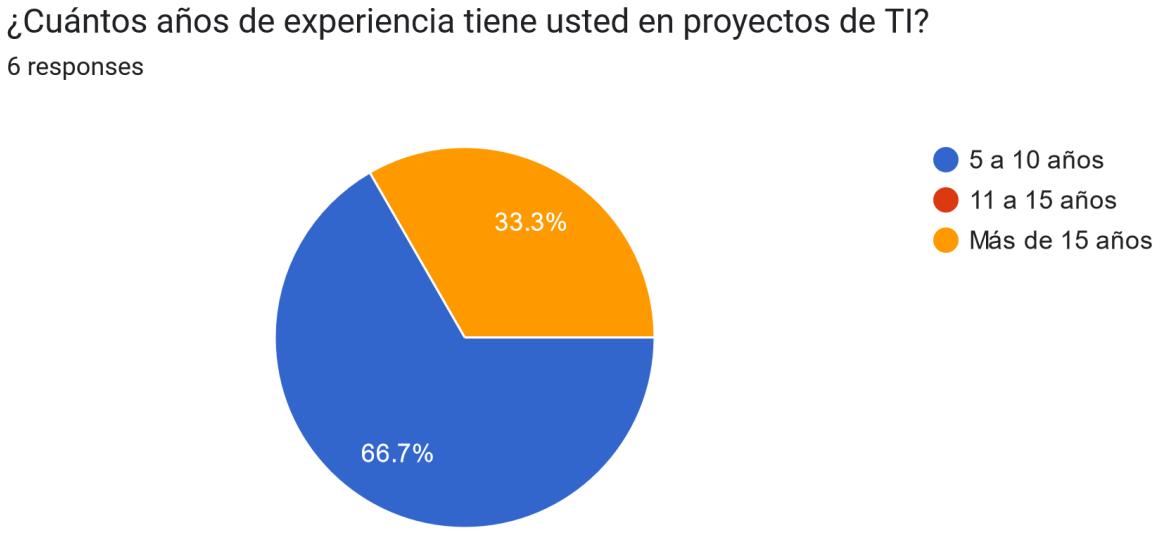
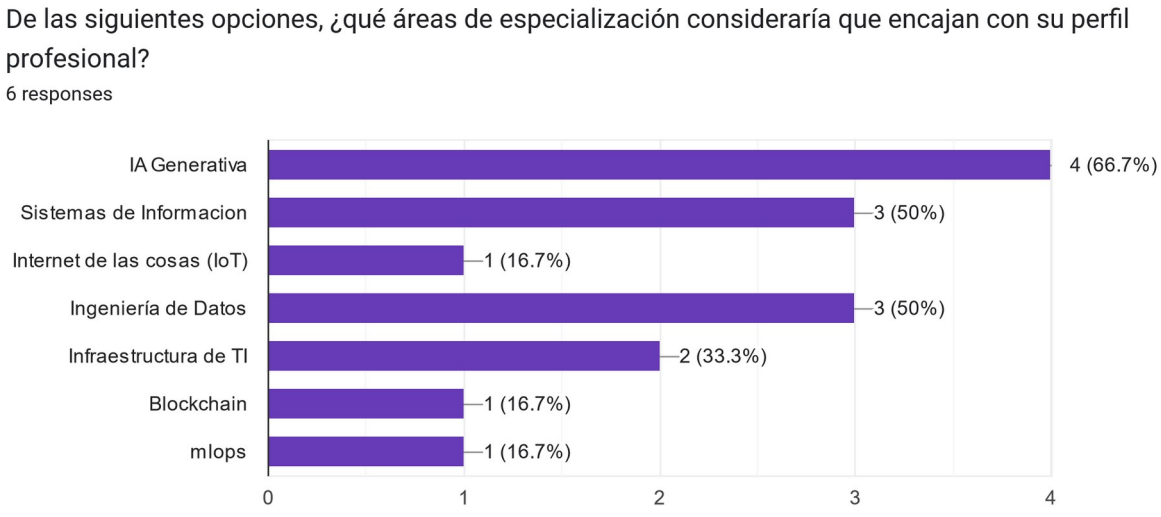


Figura 53

Áreas de especialización de los expertos



4.3.2 Estructura General del Cuestionario

El cuestionario se dividió en siete secciones, agrupadas en dos conjuntos: secciones de presentación y de evaluación. Las dos primeras secciones de presentación se estructuraron de la siguiente forma:

1. En una primera sección se solicitaron datos del experto para armar su perfil, tomando datos como el nombre, cargo actual, años de experiencia en proyectos de TI y áreas de especialización.
2. La siguiente sección consistió en la presentación del proyecto, donde se expuso una introducción en conjunto con un breve video explicativo del uso de la aplicación web. También se incluyó el diagrama general del producto, mismo que podemos encontrar en **Figura 15**.

Las seis secciones restantes de evaluación se enfocaron en distintos aspectos del proyecto mediante tres preguntas simples. En cada sección de evaluación se realizaron variaciones de las siguientes tres preguntas:

¿La elección del recurso fue la adecuada?

¿La aplicación del recurso se ejecutó correctamente?

¿Recomendaría algún otro recurso? ¿Por qué?

Entendiendo como “recurso” los distintos elementos presentados como el enfoque principal de cada sección. La estructura original de la encuesta puede ser encontrada en **Anexo 9**, igualmente, la intención y resultados obtenidos en cada sección evaluada se explica en el siguiente punto.

4.3.3 Resultados

4.3.3.1 Arquitectura de la Solución.

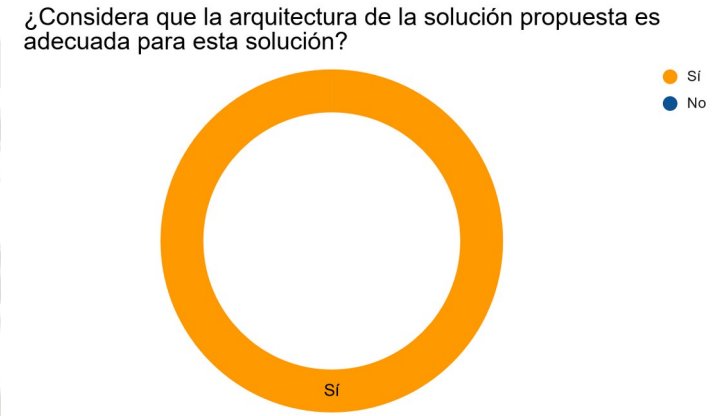
Se presentó la arquitectura de la solución de la parte generativa del producto, junto a la descripción paso a paso del flujo que esta seguía para procesar las preguntas del usuario y ser

capaz de responderlas. Puede observar un resumen de la lista de respuestas a continuación, incluyendo la citada en **Figura 54**.

¿Considera que la arquitectura de la solución propuesta es adecuada para esta solución?

Figura 54

Resultados de pregunta sobre selección de recurso adecuada en "Arquitectura de la Solución"



¿Qué aspectos de la arquitectura considera que son más fuertes? ¿Y cuáles podrían mejorarse?

Aspectos Fuertes:

- Automatización del flujo de consulta.
- Interfaz de usuario amigable, intuitiva y fácil de usar; inspirada en productos similares más populares.
- Flexibilidad y escalabilidad de la arquitectura, es posible adaptarla diferentes escenarios y volúmenes de datos.
- Etapas y secuencia de acciones claras para obtener la respuesta deseada a partir de una entrada en lenguaje natural.
- La arquitectura separa y controla cada proceso de la manipulación de Gen, lo que mejora la eficiencia y la confiabilidad.
- Interacción con SQL para obtener información actual sin la necesidad de contener la totalidad de la información en un *embedding*.

Aspectos a Mejorar:

- Explicar mejor a los usuarios los beneficios que obtienen al utilizar esta solución, tanto en términos económicos como de reducción de tiempo.
- La implementación de disparadores con palabras clave podría facilitar la navegación y la búsqueda de información específica.
- La precisión de las consultas depende en gran medida del LLM utilizado. Se recomienda explorar otras alternativas para mejorar la precisión y gestionar ambigüedades.
- Se deben implementar medidas de seguridad y validación de consultas más robustas para proteger la información.
- En el caso de consultas complejas, sería útil proporcionar más detalles sobre los esquemas de la base de datos o el contexto suficiente para procesar consultas.

¿Recomendaría alguna otra arquitectura para este tipo de soluciones? ¿Por qué?

En general, las respuestas indicaron que la arquitectura actual con el modelo LLM está cumpliendo su función adecuadamente y no se recomienda un cambio. Se destacan algunos puntos a considerar:

- **Necesidad de mayor detalle:** Se sugiere proporcionar más información sobre los LLM, especialmente en referencia a la seguridad se necesita detallar el acceso privado al *bot*.
- **Énfasis en la adaptabilidad:** Se resalta la capacidad del LLM para adaptarse a las necesidades específicas del negocio. Además, se hace mención a que la arquitectura podría variar según el público objetivo al que se dirige la solución, ofreciendo así una solución más personalizada.
- **Eficiencia en tiempo y recursos:** Se considera que la arquitectura actual es la óptima en términos de tiempo y microprocesos.

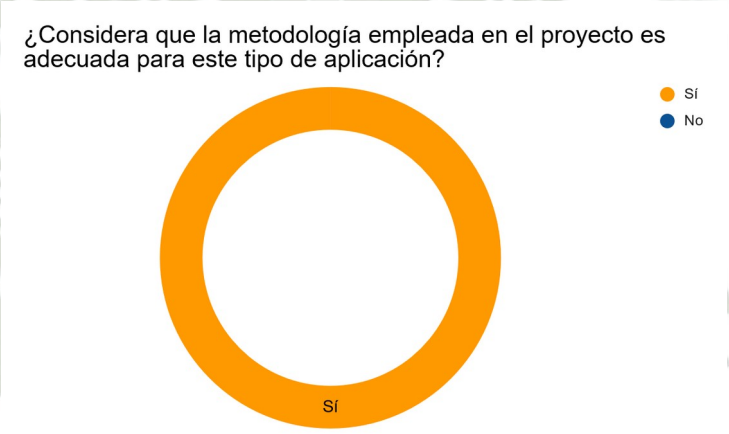
4.3.3.2 Metodología de Desarrollo.

Se presentó la metodología de desarrollo del producto, explicando su naturaleza iterativa, ventajas y etapas consideradas para cada iteración. Se solicitó a los encuestados que dieran su perspectiva de la elección y aplicación de esta metodología. Puede observar un resumen de la lista de respuestas a continuación, incluyendo la citada en **Figura 55.**

¿Considera que la metodología empleada en el proyecto es adecuada para este tipo de aplicación?

Figura 55

Resultados de pregunta sobre selección de recurso adecuada en "Metodología de Desarrollo"



¿Qué aspectos de la metodología considera que son más fuertes? ¿Y cuáles podrían mejorarse?

Aspectos fuertes:

- Ciclo iterativo: Permite la retroalimentación continua y la adaptación a los cambios.
- Integración de diseño y desarrollo: Facilita la comunicación y la colaboración entre equipos.
- Evaluación continua que garantiza la calidad del producto.
- Visión global sobre el proyecto, permite la comprensión de los casos de uso.

- Metodología completa: Abarca todo el ciclo de vida del proyecto, desde la concepción hasta la entrega del producto final.

Aspectos a mejorar:

- Gestión de riesgos: a pesar de que la metodología IWeb es completa, se requiere prestar especial atención en la gestión de riesgos.
- Se requieren detalles técnicos sobre el uso de entornos de trabajo, lenguajes de programación y su interrelación.

¿Recomendaría alguna otra metodología para el desarrollo de este tipo de aplicaciones? ¿Por qué?

Cinco de los seis expertos encuestados no recomendaron una metodología diferente al modelo actual de desarrollo web para este tipo de aplicaciones. Consideran que el modelo actual es idóneo y acertado para este tipo de proyectos.

Uno de los expertos sugirió incorporar el método CRISP-DM en caso de que se utilicen modelos de Machine Learning en la solución.

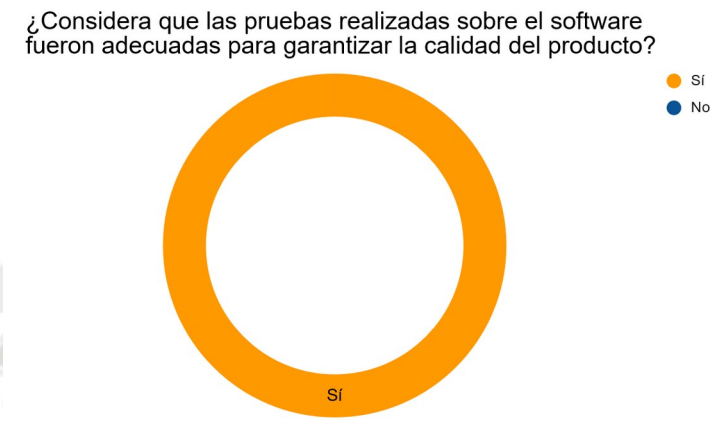
4.3.3.3 Pruebas Realizadas.

Se listaron las pruebas realizadas sobre la aplicación: unitarias, de integración, seguridad, rendimiento y usabilidad. Además, se adjuntaron los resultados de cada una de ellas. Puede observar un resumen de la lista de respuestas a continuación, incluyendo la citada en **Figura 56**.

¿Considera que las pruebas realizadas sobre el software fueron adecuadas para garantizar la calidad del producto?

Figura 56

Resultados de pregunta sobre selección de recurso adecuada en "Pruebas Realizadas"



¿En su opinión, las pruebas cubrieron los suficientes casos de uso y funcionalidades de la aplicación?

Las respuestas a esta pregunta dejaron en claro que las pruebas cubrieron los casos de uso y funcionalidades más importantes de la aplicación. Los expertos consideraron que el alcance de las pruebas fue adecuado y que se abordaron las necesidades fundamentales. Sin embargo, también dejaron en claro que siempre hay margen de mejora y que no se pueden anticipar errores. También sugirieron que una mayor cantidad de usuarios para las pruebas de usabilidad habría sido pertinente.

¿Recomendaría algún otro tipo de prueba para la aplicación? ¿Por qué?

Cinco de los seis expertos no recomendaron realizar pruebas adicionales a las cinco pruebas mencionadas en la encuesta, debido a que consideran que estas cubren los aspectos fundamentales para una evaluación de QA adecuada.

Sin embargo, dos de los encuestados sí sugieren pruebas adicionales:

- Pruebas de estrés para evaluar el comportamiento de la aplicación bajo una carga alta de información, solicitudes y usuarios.
- Pruebas A/B para comparar diferentes versiones de la aplicación y determinar cuál tiene un mejor rendimiento o impacto en los usuarios.

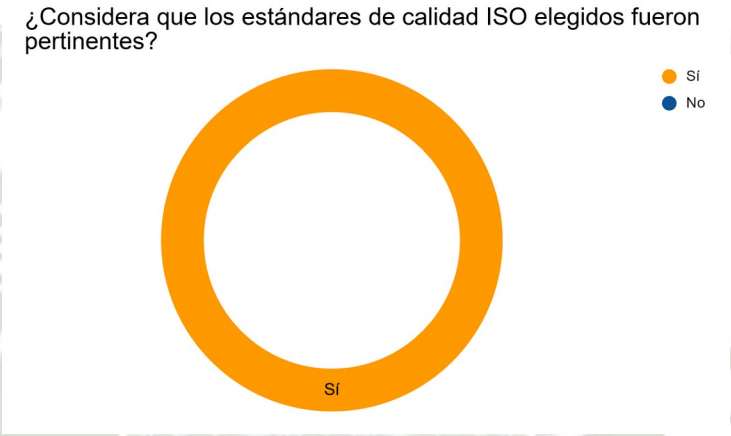
4.3.3.4 Estándares de Calidad ISO Aplicados.

Sección tuvo el objetivo de presentar los estándares ISO 25010 y 27001 aplicados al proyecto. Se expusieron los puntos relevantes tomados en cuenta de cada uno de ellos y se adjuntó el proceso completo del uso de estos sobre el producto software. Puede observar un resumen de la lista de respuestas a continuación, incluyendo la citada en **Figura 57**.

¿Considera que los estándares de calidad ISO elegidos fueron pertinentes?

Figura 57

Resultados de pregunta sobre selección de recurso adecuada en "Estándares de Calidad ISO Aplicados"



¿Considera que los estándares de calidad ISO se aplicaron de forma correcta sobre el proyecto? ¿Por qué o por qué no?

La mayoría de los participantes consideraron que los estándares se aplicaron de forma correcta, o al menos, se han realizado esfuerzos significativos para implementarlos correctamente. Los expertos mencionaron algunos aspectos específicos que respaldan su valoración:

- Implementación de procesos y prácticas específicas para cumplir con los requisitos de calidad.
- Realización de mediciones y evaluaciones regulares de la calidad del software y los sistemas de información.
- Aplicación de los puntos de cada norma ISO a cabalidad.

¿Recomendaría algún otro estándar de calidad para este tipo de sistemas? ¿Por qué?

Cinco de los seis expertos no recomendaron ningún otro estándar, la principal razón fue porque el sistema ya ha sido validado con una norma ISO reciente. Se considera que esto es suficiente para garantizar la calidad del sistema y que estándares adicionales no serían relevantes para el proyecto.

4.4 Estándares ISO

4.4.1 Estándar ISO/IEC 25010:2023

El estándar ISO/IEC 25010 tiene como enfoque valorar la usabilidad de un producto *software* evaluando su calidad de este desde distintos criterios, definiendo calidad como el grado de satisfacción de los requisitos de los usuarios por parte de la aplicación (International Organization for Standardization, 2023).

4.4.1.1 Adecuación Funcional.

Tiene como principal enfoque verificar la capacidad del producto *software* para satisfacer los requerimientos de los clientes (International Organization for Standardization, 2023). Contando con el documento de “Especificación de Requisitos” Anexo 5, fue posible listar cada requerimiento para clasificarlo según sus características y funcionalidad de la aplicación, observe **Tabla 15**.

Tabla 15

Evaluación de requerimientos según su adecuación funcional

Código	Descripción	Evaluación
RF-01	El sistema permitirá a los usuarios crear, editar o eliminar registros relacionados a productos (marcas, laboratorios, categorías y unidades de medida) ingresando detalles como código, nombre, descripción, categoría, vigencia de registro sanitario, etc.	El requisito fue alcanzado satisfactoriamente utilizando el criterio de **completitud funcional**.
RF-03	Cada producto contará con listas de precio, esto con el fin de optimizar la gestión de precios	El requisito fue alcanzado satisfactoriamente utilizando

	unitarios (venta y compra) del producto en sus diferentes presentaciones (caja, blíster, unidad).	el criterio de **completitud funcional**.
RF-06	Las órdenes de compra podrán ser generadas únicamente a partir de una cotización previa y también podrán ser visualizadas en formato PDF.	El requisito fue alcanzado satisfactoriamente utilizando el criterio de **pertinencia funcional**.
RF-07	Las compras podrán ser generadas independientemente o a partir de una orden de compra previa, generarán un movimiento en almacén, actualizarán el stock del producto/s, se registrarán nuevos lotes de compra y podrán ser enviadas por *email* en formato PDF.	El requisito fue alcanzado satisfactoriamente utilizando el criterio de **corrección funcional**.
RF-19	Al generar una deuda en una venta, ésta se encontrará asociada tanto a al comprobante como al cliente.	El requisito fue alcanzado satisfactoriamente utilizando el criterio de **pertinencia funcional**.
RF-20	Cada cliente contará con una cuenta corriente, desde aquí se visualizarán sus deudas y el estado de estas (pendiente o saldada). Se podrán registrar los pagos, los cuales actualizarán el monto pendiente de la deuda.	El requisito fue alcanzado satisfactoriamente utilizando el criterio de **corrección funcional**.

Nota. Elaboración propia.

4.4.1.2 Eficiencia de Desempeño.

Se evalúa el rendimiento de la aplicación en determinado tiempo y tomando en cuenta los recursos como lo son CPU, memoria y almacenamiento (International Organization for Standardization, 2023). Tomando en cuenta lo anteriormente descrito en **4.1.4** para las pruebas de rendimiento de la aplicación podemos asegurar que los tiempos de ejecución de la aplicación son favorables. Para la medición de utilización de recursos por parte de nuestra aplicación se tomaron como referencia los requerimientos recomendados listados en **Anexo 5**. Observe la **Tabla 16** a continuación, notará el poco uso de los recursos por parte de nuestra aplicación.

Tabla 16

Medición de utilización de recursos de hardware por parte de la aplicación

Recurso	Requisito Recomendado	Utilización del Recurso
CPU	4 núcleos a 2.8 GHz	3% a 25% de uso
Memoria	8GB RAM	12.5 MB a 29.8 MB
Almacenamiento	5GB HDD o SSD	627,7 MB

Nota. Elaboración propia.

4.4.1.3 Compatibilidad.

Este criterio es descrito como la capacidad del producto *software* para interactuar con plataformas externas con el fin de completar tareas y funciones solicitadas (International Organization for Standardization, 2023).

Tabla 17

Capacidad de compatibilidad de la aplicación

Coexistencia	El producto *software* es una aplicación web desplegada en un hosting compartido, lo que quiere decir que, comparte recursos del servidor con otros sitios web sin causar un conflicto.
Interoperabilidad	La aplicación utiliza herramientas de terceros para, por ejemplo, generar reportes *xlsx* y documentos *pdf*. Estos mismos reportes pueden ser enviados por *email* utilizando un proveedor externo. Finalmente, también se hace uso de APIs de SUNAT para la validación de comprobantes de venta y actualización del cambio de dólar a sol.

Nota. Elaboración propia

4.4.1.4 Capacidad de Interacción.

Hace referencia a la capacidad del sistema para permitir que los usuarios completen tareas interactuando con la interfaz de la aplicación (International Organization for Standardization, 2023). Para la evaluación de este aspecto tomamos en cuenta la encuesta hecha para las pruebas de usabilidad del sistema **4.1.5**, donde podemos notar que para los distintos criterios considerados la calificación por parte de los usuarios fue mayormente

positiva. Por otra parte, también obtuvimos retroalimentación de los usuarios al utilizar la plataforma en sus distintas etapas de desarrollo. Clasificamos los atributos de nuestra aplicación en las siguientes subcaracterísticas.

Tabla 18

Subcaracterísticas de la capacidad de interacción del producto

Reconocibilidad de la adecuación	Hace referencia a la capacidad del producto para hacer comprender al usuario que sus necesidades están cubiertas (International Organization for Standardization, 2023). Debido a que este proyecto es una nueva versión de un aplicación anterior, los usuarios ya están familiarizados con los términos, módulos, operaciones, etc. Las nuevas funcionalidades incluidas también son reconocibles por usuarios ya que reflejan procesos de negocio reales.
Aprendibilidad	Hace referencia al aprendizaje del uso de la aplicación en un periodo de tiempo determinado (International Organization for Standardization, 2023). La nueva versión del sistema fue diseñada de una forma más intuitiva **4.1.5.2**, los usuarios pueden ver un video demostración del uso de la plataforma y en un promedio de media hora son capaces de realizar compras, ventas, deudas y reportes.
Operabilidad	Verifica la facilidad de uso del sistema, está relacionado a los anteriores criterios y podemos comprobar el grado de influencia dentro de nuestro sistema web observando los resultados de la encuesta a los usuarios sobre usabilidad **4.1.5**.
Protección contra errores de usuario	Validación de datos según su tipo al interactuar con los módulos del sistema, restricción de modificación de datos como fechas de emisión, prevención de eliminación de registros ya utilizados en operaciones y pantalla de carga preventiva para cuando el usuario deba esperar a que se complete una operación.

Nota. Elaboración propia

4.4.1.5 Fiabilidad.

Se centra en la capacidad del producto para completar funciones solicitadas bajo ciertas condiciones y periodos de tiempo específicos (International Organization for Standardization, 2023). Para validar este criterio nos apoyamos de la pruebas realizadas sobre la aplicación **4.1**, sobre todo las pruebas unitarias, de integración, seguridad y rendimiento nos revelan resultados relevantes para esta característica.

- Podemos garantizar en cierta medida la ausencia de fallos en la aplicación mediante las pruebas unitarias y de seguridad. Estas pruebas fueron automatizadas y pueden ser consultadas en los puntos **4.1.1** y **4.1.3**.
- Utilizar una forma de desarrollo clara, optimizada y basada en buenas prácticas contribuye con la disponibilidad de la aplicación. Por ejemplo, programar actividades exigentes en tiempo y recursos para que se ejecuten diariamente en horas inactivas.
- La tolerancia a fallos de *software* es cubierta por un diseño de funciones que pueden volver a invocarse si fallan. Un ejemplo es la creación de una venta: la información se envía a la API de SUNAT para la facturación electrónica, y si esta falla, se puede reintentar la declaración del comprobante tantas veces como sea necesario.
- Para el caso de tolerancia a fallos de *hardware*, el producto *software* en producción se encuentra alojado en el servicio de web *hosting* del proveedor OVH, el cual cuenta con opciones de replicación de información en caso de fallos (OVHcloud, 2022).
- Finalmente, la capacidad de recuperación de información del sistema web es posible gracias al *backup* diario de la base de datos, reduciendo así la cantidad de información perdida en caso de algún error inesperado.

4.4.1.6 Mantenibilidad.

Busca garantizar la suficiente capacidad del producto para ser actualizado efectiva y eficientemente de acuerdo con las necesidades evolutivas de los usuarios, también se busca corregir y perfeccionar la aplicación de forma continua (International Organization for Standardization, 2023).

Tabla 19

Subcaracterísticas de la capacidad de mantenibilidad del producto

Modularidad	Los cambios en un componente no deberían afectar al resto, esto puede ser cubierto gracias a la arquitectura de la aplicación. VueJs permite separar las vistas de usuario en componentes independientes, además, en los casos en los que interactúan entre sí, las variables y funciones fueron normalizadas para evitar posibles errores de datos indefinidos. Por otra parte, en el *backend* cada función realiza una sola acción, lo que significa que son independientes incluso cuando interactúan entre sí, muchas de estas devuelven resultados en un formato estándar para que este pueda ser procesado en otras instancias de la aplicación.
Reusabilidad	La aplicación está construida utilizando tecnologías como Laravel, VueJs, lenguajes como php y javascript. Las funciones, componentes y vistas pueden ser exportados a otros productos, sin embargo, se deberán realizar algunos cambios dependiendo de las tecnologías involucradas.
Analizabilidad	Gracias a las pruebas automatizadas implementadas sobre el *software* **4.1.1**, es posible detectar los problemas del *backend* de una forma rápida. Para el *frontend* de la aplicación, al no contar con pruebas automatizadas aún, se requieren pruebas manuales.
Capacidad para ser modificado	El uso de buenas prácticas, organización del código, inclusión de comentarios descriptivos, estructuración del directorio del proyecto y normalización de funciones/variables facilita la modificación de la aplicación aún cuando el desarrollador no conoce del todo el proyecto.
Capacidad para ser	El *framework* Laravel ofrece un completo sistema de pruebas para

probado	la aplicación, esto la hace altamente capacitada para pruebas automatizadas (Otwell, 2023). El diseño intuitivo y la facilidad de acceso a la información también permite que las pruebas manuales sean accesibles por distintos tipos de usuarios.

Nota. Elaboración propia.

4.4.2 Estándar ISO/IEC 27001:2022

El estándar ISO/IEC 27001 se centra en la gestión de seguridad de la información. Para nuestro sistema esta es una actividad de la alta criticidad, debido a que la información con la que se trabaja en la plataforma es utilizada en actividades que involucran transacciones monetarias como lo son procesos de ventas, compras o deudas. Por otra parte, también se almacena información personal como lo son los datos de clientes, proveedores, números de documentos de identificación, correos electrónicos, números de teléfono, etc.

4.4.2.1 Planificación.

El primer paso consistió en la planificación, donde se identificaron los riesgos de seguridad que podrían influir de forma negativa en el uso normal de la aplicación. En **Tabla 20** puede observar el detalle de los riesgos identificados junto a su impacto y probabilidad en el presente sistema de información.

Tabla 20

Identificación de riesgos de la aplicación, impacto y probabilidad

Riesgos de la aplicación			
Código	**Descripción**	**Impacto**	**Probabilidad**
RK001	Usuario no autenticado ejecuta una función en el sistema	Bajo	Media
RK002	Un usuario sin el permiso requerido puede visualizar un componente en el sistema	Bajo	Media
RK003	Se ejecuta un comando de *SQL Injection* en la base de datos	Medio	Baja
RK004	Las credenciales de un usuario administrador son	Alto	Media

Riesgos de la aplicación			
Código	**Descripción**	**Impacto**	**Probabilidad**
	filtradas y la información dentro del sistema es alterada		
RK005	Una llamada a una función de *backend* es interrumpida y la operación no es concretada	Bajo	Alta
RK006	Un agente externo logra replicar la sesión de un usuario y manipula la información a la que tiene acceso	Alto	Baja

Nota. Tomado de (International Organization for Standardization, 2022).

4.4.2.2 Implementación.

Se establecieron políticas de seguridad en base a los riesgos identificados. Estas políticas buscan prevenir, mitigar o reducir el impacto y probabilidad de los riesgos identificados, puede observar el detalle de estas políticas a continuación.

Tabla 21

Implementación de políticas de seguridad para los riesgos identificados

Políticas de Seguridad		
Código	**Descripción**	**Riesgo al que hace referencia**
PS001	Las rutas de acceso a las funciones del sistema deben estar correctamente aseguradas mediante un *middleware* que verifique la sesión del usuario	RK001
PS002	Cada vez que un componente nuevo sea agregado debemos evaluar si este requiere o no un permiso asociado	RK002
PS003	Todas las consultas a la base de datos deben estar programadas utilizando, el cual es un ORM proporcionado por Laravel que incluye controles de seguridad para sentencias de SQL Injection (Otwell, 2023)	RK003
PS004	Se deben realizar *backups* diarios de la base de datos, además, los usuarios dependiendo su rol deben tener acceso	RK004

Políticas de Seguridad		
Código	**Descripción**	**Riesgo al que hace referencia**
	limitado a una parte de la información.	
PS005	Se debe procurar activar la pantalla de carga cada vez que se llamen funciones de *backend*, de esta forma, evitamos que el usuario sature el sistema realizando otras acciones	RK005
PS006	Las contraseñas de usuario deben ser confidenciales, el cifrado en base de datos es obligatorio y las contraseñas de administradores deben cumplir con el nivel de complejidad necesario.	RK006

Nota. Tomado de (International Organization for Standardization, 2022).

Cada una de estas políticas de seguridad están asociadas a uno o varios procesos de negocio para los cuáles se establecen controles con el objetivo tomar en cuenta criterios como la gestión de accesos, permisos, seguridad de la red, gestión de incidentes, etc. Puede observar a continuación en **Tabla 22**.

Tabla 22

Implementación de controles de seguridad a los procesos de negocio

Controles de Seguridad	
Criterio	**Descripción**
Gestión de Accesos	• Implementar controles de seguridad robustos limitando el acceso a funciones de *backend* a usuarios no autenticados. • Limitar el tiempo de sesión de los usuarios que permanezcan inactivos por determinado tiempo. • Sólo administradores pueden agregar, modificar o eliminar usuarios en la aplicación.
Gestión de Permisos	• Los permisos hacen referencia a una única función/característica en el sistema. • Los permisos están asociados a roles dentro del sistema, cada usuario cuenta con un sólo rol • Los roles son modificados sólo por administradores

Controles de Seguridad	
Criterio	**Descripción**
	• Los permisos son modificados sólo por administradores
Gestión de Empresas/Sucursales	• Implementar restricciones de acceso a información entre empresas • Implementar restricciones de acceso a información entre sucursales • Implementar restricción de aislamiento de información, gestión de procesos e inventario entre empresas
Gestión de Incidentes	• La gestión de incidentes comprende las etapas de detección, investigación, respuesta y mejoras de seguridad • La detección e investigación de incidentes se apoyan de registros guardados en la base de datos y *logs* generados para cada venta.
Seguridad de red	• Los servidores donde se aloja la aplicación garantizan una infraestructura que contiene ataques DDoS • Los servidores están configurados con el protocolo de seguridad SSL permitiendo adquirir un certificado de este tipo • Se aplica también un Web Application Firewall para evitar otro tipo de solicitudes maliciosas a la aplicación (OVHcloud, 2022)

Nota. Tomado de (International Organization for Standardization, 2022).

4.4.2.3 Evaluación.

Se definieron KPIs para la evaluación del rendimiento de las políticas y controles tomados para el sistema tomando como base las prácticas del estándar ISO 27001 anteriormente descrito. Con el objetivo de poder mesurar de forma precisa el porcentaje de mejora una vez implementado cada control, estas cifras nos dan una visión más precisa de la seguridad del sistema. Los KPIs especializados en la seguridad de la aplicación se muestran en **Tabla 23**, adicionalmente, puede encontrar la lista del resto de KPIs considerados en **4.4.3**.

Tabla 23

Lista de resultados trimestrales de KPIs de seguridad

KPIs Seguridad - Trimestral			
Criterio	**Descripción**	**Límite estimado**	**Medición actual**
Incidentes de riego	Cantidad de incidentes relacionados a manipulación, filtración o pérdida de datos mensualmente	3	1,5
Tasa de éxito de ataques	Mide el porcentaje de intentos de ataque que logran vulnerar la seguridad del sistema	20%	10%
Tiempo de respuesta a incidentes	Tiempo que tarda el equipo de seguridad en detectar, investigar y responder a un incidente de seguridad	45 min	45 min
Empleados con formación en seguridad	Porcentaje de empleados que han recibido formación en seguridad informática, saben cómo reconocer y prevenir vulnerabilidades	40%	90%

Nota. Elaboración propia

4.4.2.4 Mejora Continua.

El objetivo principal de implementar un estándar de seguridad como el ISO 27001:2022 es aminorar la cantidad de incidentes de seguridad y su impacto en el flujo habitual del sistema. Cada evento relacionado con la seguridad debe documentarse con el fin de aprender de él, ésta es la base de la mejora continua.

En el registro de cada incidente almacenaremos información sobre los detalles del evento, sus consecuencias, controles que se aplicaron para resolverlo y la efectiva de la acción tomada. Además, determinaremos la causa raíz del evento mediante preguntas como:

1. ¿Qué acción realizada dio paso al incidente de seguridad?
2. ¿Con qué formación contaban los empleados acerca de un incidente como este?
3. ¿En qué controles de seguridad recae este incidente? ¿No está considerado?

Así la mejora continua, en términos de seguridad, será aplicada de forma organizada y con la suficiente eficiencia.

4.4.3 KPIs del Proyecto

Posterior a la implementación en producción de nuestra aplicación, se consideró fundamental sustentar la mejora en los procesos de negocio que este software aporta a las farmacias.

Con este objetivo, se realizaron y documentaron evaluaciones del rendimiento de dichos procesos en cada uno de los criterios evaluados, comparando el desempeño antes y después de la implementación del renovado sistema de información.

Tabla 24

Lista de resultados trimestrales de KPIs del sistema de información

KPIs Sistema de Información - Trimestral			
Criterio	**Descripción**	**Límite estimado**	**Medición actual**
Tiempo de resolución de *tickets*	Tiempo de implementación de ajustes requeridos por el cliente tanto en la etapa de desarrollo como en la etapa de mantenimiento.	45 min	45 min
Tiempo medio de permanencia de usuarios	Mide el tiempo de sesión de los usuarios al utilizar el sistema en su jornada diaria	7 h	6.83 h
Tasa de conversión de usuarios	Porcentaje de usuarios que completan una acción deseada en el sistema	30%	90%

Nota. Elaboración propia

Tabla 25

Lista de resultados trimestrales de KPIs del área de ventas

KPIs Ventas - Trimestral			
Criterio	**Descripción**	**Límite estimado**	**Medición actual**
Ventas totales	Monto total de ventas trimestral	S/. 180,000	S/. 243,000

Crecimiento de ventas	Incremento de las ventas totales del trimestre anterior al actual	3%	7,5%
Tasa de conversión	Proporción de visitas que finalizan en una venta	45%	80%

Nota. La referencia para el valor mínimo esperado es tomada de (Mendoza Bernedo & Anchiraico Bernaola, 2018)

Tabla 26

Lista de resultados trimestrales de KPIs del área de compras y almacén

KPIs Compras - Trimestral			
Criterio	**Descripción**	**Límite estimado**	**Medición actual**
Coste de adquisición promedio	Mide el coste promedio de adquirir un producto desde proveedores.	<= S/. 10	S/. 5.2
Margen bruto	Describe la diferencia entre el precio de venta y el coste de adquisición	30%	48%
Coste de productos en almacén	Costo de almacén pequeño en zona céntrica por producto en un trimestre	S/. 1.5	S/. 0.90

Nota. La referencia para el valor mínimo esperado es tomada de (Mendoza Bernedo & Anchiraico Bernaola, 2018)

CONCLUSIONES

1. Para la etapa de análisis del proyecto, se ejecutaron la recopilación y documentación de los requerimientos funcionales y no funcionales, comprendiendo las necesidades de los usuarios utilizando técnicas de obtención de información como entrevistas, ingeniería de requisitos, análisis de alternativas, entre otros.
2. Se logró realizar un estudio de viabilidad del proyecto planificando los plazos de tiempo para cada tarea y se evaluaron la factibilidad técnica, económica, legal y operacional de este.
3. Se completó el proceso de diseño de la aplicación utilizando diagramas de componentes, casos de uso, secuencia y BPMN; además de también diseñar la base de datos e interfaces de usuario.
4. Se diseñó y desarrolló un *chatbot* conversacional impulsado por modelos de lenguaje de la familia GPT, para que usuarios realicen consultas en lenguaje natural con el fin de obtener información actual de la base de datos de forma rápida e intuitiva.
5. La arquitectura diseñada e implementada para la integración IA Generativa es compatible con otras aplicaciones, adaptable a la mayoría de los lenguajes de programación, *frameworks* y arquitecturas de *software* actuales.
6. Se desarrollaron y ejecutaron pruebas automatizadas evaluando la integración, seguridad y rendimiento de la aplicación, se registraron los resultados y se realizaron correcciones en el *software*.
7. Para garantizar la calidad, cumplir con los requisitos y mejorar la seguridad del producto *software*, se implementaron las normas ISO/IEC 25010 y 27001. La adopción de estas normas diferencia al producto final de otras soluciones en el mercado, brindándole mayor calidad, seguridad y confiabilidad.

RECOMENDACIONES

1. Generar un *embedding* para la estructura de la base de datos, alimentando de mejor manera la base de conocimientos del LLM, ahorrando costos y unificando las consultas para todos los tópicos en lugar de dividirlos por compras, ventas y productos.
2. Mejorar la interacción con el asistente de IA incluyéndolo como un *widget* de chat disponible a lo largo de toda la aplicación. También agregar más funcionalidades como la creación, edición y eliminación de registros mediante solicitudes directas al *chatbot* autorizando estas acciones según el rol del usuario.
3. Desarrollar un subsistema para obtención automática de información desde el Observatorio Peruano de Productos Farmacéuticos (OPM) con el fin de capturar los precios de productos actualizados e importarlos a la base de datos de la aplicación.
4. Desarrollar módulos adicionales en el sistema para cubrir los procesos de la gestión de recursos humanos: administración de planillas, gestión de relaciones laborales, planificación de fuerza laboral, gestión de desempeño y gestión de talento.
5. Implementar un subsistema para la analítica de datos generados a partir de las interacciones con el *chatbot*, con el fin de obtener *insights* de valor que permitan mejorar su usabilidad y accesibilidad.

REFERENCIAS BIBLIOGRÁFICAS

Acharya, A., Singh, B., & Onoe, N. (2023). LLM Based Generation of Item-Description for Recommendation System. In *Proceedings of the 17th ACM Conference on Recommender Systems* (pp. 1204–1207). ACM. https://doi.org/10.1145/3604915.3610647

Atlassian. (2022). *Acerca de Trello*. https://trello.com/about

Barrientos, E., & Rincón, M. (2020). Diseño de una Plataforma de E-commerce para el Sector de Alimentos. In E. Serna (Ed.), *Desarrollo e Innovación en Ingeniería* (5ta ed., pp. 180–186). https://doi.org/10.5281/zenodo.4031208

Calebio, J., & Notion Labs Inc. (2022). *¿Qué es Notion?* https://www.notion.so/Qu-es-Notion-bd5c88a4ec254793b4da2e9cdc31f770

Castillo Estrada, C. M., Cancino Villatoro, K., Benavides García, V., & de la Cruz Vázquez, A. (2022). Diseño de un Sistema Web para el Control de Curriculum Vitae Electrónico de Personal Docente Basado en una Arquitectura Orientada a Servicios (API REST). *Revista de Investigación En Tecnologías de La Información*, *10*(20), 28–42. https://doi.org/10.36825/riti.10.20.003

Chase, H. (2023). *LangChain Docs*. https://python.langchain.com/docs/get_started/introduction

Delao, I. (2018). *Implementación de un sistema de información para la mejora de la gestión de la Farmacia Megafarma* [Universidad Nacional del Centro del Perú]. https://repositorio.uncp.edu.pe/bitstream/handle/20.500.12894/5306/T010_44528339_T.pdf?sequence=1

Dhoni, P. (2023). Unleashing the Potential: Overcoming Hurdles and Embracing Generative AI in IT Workplaces: Advantages, Guidelines, and Policies. *Athorea Preprints*. https://doi.org/10.36227/techrxiv.23696709.v1

DIGEMID. (2023). *DIGEMID Preguntas Frecuentes*. https://www.digemid.minsa.gob.pe/webDigemid/preguntas-frecuentes/

Enlightn. (2020). *Laravel Enlightn Docs*. https://www.laravel-enlightn.com/docs/getting-started/installation.html

Freire, S. K., Wang, C., & Niforatos, E. (2024). *Chatbots in Knowledge-Intensive Contexts: Comparing Intent and LLM-Based Systems*. ArXiv Preprint ArXiv. https://arxiv.org/pdf/2402.04955.pdf

International Organization for Standardization. (2022). *ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. https://www.iso.org/obp/ui/en/#iso:std:iso-iec:27001:ed-3:v1:en

International Organization for Standardization. (2023). *ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model*. https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25010:ed-2:v1:en

Jaque, M. (2015, November 7). *Formas Normales (1FN, 2FN, 3FN y FNBC)*. https://19e37.com/blog/formas-normales-1fn-2fn-3fn/

Ley de Organización y Funciones Del Instituto Nacional de Defensa de La Competencia y de La Protección de La Propiedad Intelectual, Pub. L. No. 1033, Diario Oficial El Peruano (2008).

Ley de Protección de Datos Personales N° 29733, Pub. L. No. 003-2013-JUS, Diario Oficial El Peruano (2013).

López-Mora, D., Villamar-Coloma, M., Bravo-Pino, Á., & Lozano-Rodríguez, E. (2019). El Uso de las Metodologías Ágiles y su Importancia para el Desarrollo de Software. *Killkana Técnica*, *3*(1), 25–30. https://doi.org/10.26871/killkana_tecnica.v3i1.473

Lubis, M., Witjaksono, W., & Syafiraliany, L. (2019). Analysis of Critical Success Factors from ERP System Implementation in Pharmaceutical Fields by Information System Success Model. *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 1–5. https://doi.org/10.1109/ICIC47613.2019.8985678

Mahmood, S. (2023, December). *Understanding Text Generation with Generative AI*. https://www.linkedin.com/pulse/understanding-text-generation-generative-ai-dr-sajjad-mahmood-8shif/

Meardon, E. (2022). *Todo sobre los diagramas de Gantt*. https://www.atlassian.com/es/agile/project-management/gantt-chart

Mendoza Bernedo, J. F., & Anchiraico Bernaola, W. R. (2018). *Determinación de patrones de ventas en boticas independientes para mejorar las ventas*. https://hdl.handle.net/20.500.14005/8591

Microsoft. (2022). *Documentation for Visual Studio Code*. https://code.visualstudio.com/docs

Molina, J. R., Zea, M. P., Contento, M. J., García, F. G., De Metodologías, C., Rolando, J., Ríos, M., Paola, M., Ordóñez, Z., José, M., Segarra, C., Gustavo, F., & Zerda, G. (2018). Comparación de metodologías en aplicaciones web. *3C Tecnología: Glosas de Innovación Aplicadas a La Pyme*, *7*, 1–19. https://doi.org/10.17993/3ctecno.2018.v7n1e25.1-19

Natalie Rodgers. (2022). *Diferencias entre Testing Funcional y no Funcional*. https://cl.abstracta.us/blog/diferencias-testing-funcional-no-funcional/

Navarro, K. (2019). *Análisis, diseño e implementación de un sistema de control de inventarios para la farmacia "Danafarma."*

Nvidia Corporation. (2023). *Generative AI – What is it and How Does it Work?* https://www.nvidia.com/en-us/glossary/generative-ai/

OpenAI. (2022a). *OpenAI Documentation*. https://platform.openai.com/docs/guides/gpt-best-practices

OpenAI. (2022b, November 30). *Introducing ChatGPT*. https://openai.com/blog/chatgpt

Otwell, T. (2023). *Laravel 9.x Documentation*. https://laravel.com/docs/9.x

OVHcloud. (2022). *Why Choose OVH?* https://www.ovh.com/world//us/about-us/why-choose-ovh.xml

Perú21. (2017, July 6). *Esto debes saber sobre las diferencias entre farmacia y botica para cuidar mejor tu salud*. Esto debes saber sobre las diferencias entre farmacia y botica para cuidar mejor tu salud

Pinzon, O., & Rodríguez, K. (2017). Ingeniería Web: Una Metodología para el Desarrollo de Aplicaciones Web Escalables y Sostenibles. *The Fifteen LACCEI International Multi-Conference for Engineering, Education Technology*, 19–21. http://www.laccei.org/LACCEI2017-BocaRaton/student_Papers/SP277.pdf

Pittet, S. (2023). *Los Distintos Tipos de Pruebas en Software*. https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

Project Management Institute. (2021). *A guide to the Project Management Body of Knowledge (PMBOK)* (7th ed.).

Pursell, S. (2023). *Pruebas de Usabilidad: Definición, Tipos y Ejemplos*. https://blog.hubspot.es/website/pruebas-usabilidad

Smoliński, P., Januszewicz, J., & Winiarski, J. (2023). Towards Completely Automated Advertisement Personalization: An Integration of Generative AI and Information Systems. *31st International Conference on Information Systems Development (ISD 2023)*. https://doi.org/10.62036/ISD.2023.60

Toro, A., & Peláez, L. E. (2016). Ingeniería de Requisitos: de la especificación de requisitos de software al aseguramiento de la calidad. Cómo lo hacen las Mipymes desarrolladoras de software de la ciudad de Pereira. *Entre Ciencia e Ingeniería*, *10*(20), 117–123.

Universidad Nacional del Sur. (2017). *Guía para la Documentación de Proyectos de Software*.

Visure. (2023). *What is Requirements Specification: Definition, Best Tools & Techniques*. https://visuresolutions.com/blog/requirements-specification/

You, E., Gore, A., & Kyriakidis, A. (2022). *Introduction | Vue.js*. https://vuejs.org/guide/introduction.html

ANEXOS

Anexo 1

Diagrama entidad-relación de la base de datos de la aplicación

https://www.dropbox.com/s/0p1hbmdk0hok7cj/DatabaseDiagram.png?dl=0

Anexo 2

Diagrama Gantt de la planificación temporal del proyecto

https://www.dropbox.com/scl/fi/laxl9y1ibua8jfntda3fy/Gantt_Cronograma.png?rlkey=lb0f7v5k7ultvifojg2a8xy8p&dl=0

Anexo 3

Diagrama BPMN del proceso de venta

https://www.dropbox.com/scl/fi/odnq5dfvpjvcj73yjnq5q/BPMN_ProcesoVenta.jpg?rlkey=d7zmiyqq4tajw2tsu667zxk82&dl=0

Anexo 4

Repositorio en GitHub de la versión demo de la aplicación web

https://github.com/paolo-fabrizio/SistemaFarmaciasDemo

Anexo 5

Documento especificación de requisitos del proyecto

https://www.dropbox.com/scl/fi/wfmmii7etknjsnhn75w6s/EspecificacionDeRequisitos.pdf?rlkey=knuvsgt16w5too5gjzmma03a7&dl=0

Anexo 6

Documento especificación de diseño del proyecto

https://www.dropbox.com/scl/fi/2usnkffhzl2ljq4l4yzrb/EspecificacionDeDiseno.pdf?rlkey=3i1i2wtqx5efwngj687pisv7q&dl=0

Anexo 7

Formulario de encuesta sobre usabilidad de la aplicación

https://forms.gle/VyiZFudAK9joxpAY9

Anexo 8

Resultados de encuesta sobre usabilidad de la aplicación

https://docs.google.com/spreadsheets/d/1y91_4gfP5N3iN1zMcCUtqkFUqrgnmvoXVo7KVPb1pxM/edit?usp=sharing

Anexo 9

Formulario de encuesta para juicio de expertos del proyecto

https://forms.gle/1Yd6Pgj8wkkG4dfU9

Anexo 10

Resultados de encuesta para juicio de expertos del proyecto

https://docs.google.com/spreadsheets/d/10hfS12pQzWq714d2VGocJgYxsOj-szmHsDlqUhK8hvQ/edit?usp=drive_link

Anexo 11

Repositorio en GitHub del paquete de integración de Laravel con OpenAI

https://github.com/openai-php/laravel

Anexo 12

Lista de precios de modelos GPT de OpenAI API

https://openai.com/pricing