Neiva, 17 de Octubre del 2023

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

JUAN FELIPE BUITRAGO VARGAS, con C.C. No. 1.004.157.027,

JUAN IGNACIO SOTO RAMIREZ, con C.C. No. 1.007.704.657,

_____, con C.C. No._____,

_____, con C.C. No._____,

Autor(es) de la tesis y/o trabajo de grado o _____

titulado Desarrollo De Aplicación De Gestión de Inventario Para Elemental Centro De Estética:

Un Enfoque Ágil Con Tecnologías MERN

presentado y aprobado en el año 2023 como requisito para optar al título de

INGENIERO DE SOFTWARE;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

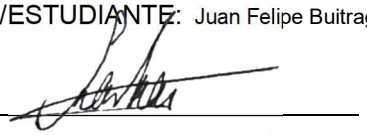
- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

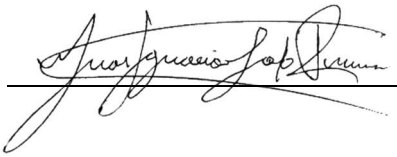
- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE: Juan Felipe Buitrago Vargas

Firma: 

EL AUTOR/ESTUDIANTE: Juan Ignacio Soto Ramirez

Firma: 

EL AUTOR/ESTUDIANTE:

Firma:_____

EL AUTOR/ESTUDIANTE:

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: Desarrollo De Aplicación De Gestión de Inventario Para Elemental Centro De Estética: Un Enfoque Ágil Con Tecnologías MERN

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
BUITRAGO VARGAS SOTO RAMIREZ	JUAN FELIPE JUAN IGNACIO

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
ZAMBRANO SANJUAN	JUAN PABLO

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: INGENIERO DE SOFTWARE

FACULTAD: INGENIERÍA

PROGRAMA O POSGRADO: INGENIERÍA DE SOFTWARE

CIUDAD: NEIVA **AÑO DE PRESENTACIÓN:** 2023 **NÚMERO DE PÁGINAS:** 123

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas_X_Fotografías_X_Grabaciones en discos___Ilustraciones en general_X_Grabados___
Láminas___Litografías___Mapas___Música impresa___ Planos___Retratos___ Sin ilustraciones___Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	Inventario	Inventory	6.	MongoDB	MongoDB
2.	Software	Software	7.	Express	Express
3.	MERN Stack	MERN Stack	8.	React	React
4.	Aplicación Web	Web Application	9.	Redux	Redux
5.	Node.js	Node.js	10.	Spa	Spa

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El presente proyecto está orientado al desarrollo de un prototipo de aplicación web de inventario para el centro de estética Elemental de la ciudad de Neiva, este fue desarrollado usando la pila de desarrollo conocida como MERN (Mongo, Express, React.JS, Node.JS). La metodología usada para el desarrollo fue ágil Scrum, iniciando con la elaboración de historias de usuarios, las cuales fueron obtenidas por medio de comunicación activa con los empleados de Elemental. Una vez obtenido los requerimientos se procede a construir un modelo de base de datos, así como los mockups de las interfaces a desarrollar, permitiendo tener una estructura clara de lo que se requiere construir, las historias de usuario fueron divididas en cuatro sprints según su complejidad, en cada sprint se desarrollaron las historias obteniendo partes así nuevas funcionalidades hasta obtener el producto completo. Finalmente, como resultado del presente proyecto, se entrega un prototipo probado totalmente funcional de aplicación web que permita la gestión de inventario por almacenes.

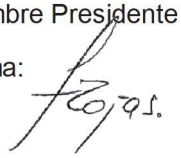
ABSTRACT: (Máximo 250 palabras)

This project is oriented to the development of a prototype inventory web application for the Elemental aesthetic center in the city of Neiva, this was developed using the development stack known as MERN (Mongo, Express, React.JS, Node.JS). The methodology used for the development was agile Scrum, starting with the development of user stories, which were obtained through active communication with the employees of Elemental. Once the requirements were obtained, we proceeded to build a database model, as well as the mockups of the interfaces to be developed, allowing to have a clear structure of what is required to build, the user stories were divided into four sprints according to their complexity, in each sprint the stories were developed obtaining parts of the product until the complete product was obtained. Finally, as a result of this project, we deliver a tested working prototype of a web application that allows inventory management by warehouses.

APROBACION DE LA TESIS

Nombre Presidente Jurado: FERNANDO ROJAS ROJAS

Firma:



Nombre Jurado: JUAN PABLO ZAMBRANO SANJUAN

Firma:

Nombre Jurado:

Firma:

UNIVERSIDAD
SURCOLOMBIANA

Desarrollo De Aplicación De Gestión de Inventario Para Elemental Centro De Estética: Un Enfoque Ágil Con Tecnologías MERN

Presentado para optar el título de:

Ingeniero de Software

Presenta:

Juan Ignacio Soto Ramírez

Juan Felipe Buitrago Vargas

Tutor: Ing. Juan Pablo Zambrano Sanjuán

Universidad Surcolombiana

Pregrado en Ingeniería de Software

Facultad de Ingeniería

Neiva, Colombia

2023

Página de Aceptación

_____
Director

_____
Jurado

_____
Jurado

Neiva, Fecha de entrega (28, septiembre, 2023)

Agradecimientos

Nos gustaría expresar nuestra profunda gratitud a nuestro tutor, el Ingeniero Juan Pablo Zambrano, cuya guía y apoyo han sido esenciales en nuestra formación y en el desarrollo de este proyecto. Agradecemos sinceramente a Elemental Centro de Estética y al cirujano plástico Juan Canencio por su colaboración y confianza en nosotros. Su participación ha sido clave en la creación de esta aplicación innovadora.

Nuestro agradecimiento a la Universidad Surcolombiana por proporcionarnos las herramientas necesarias para enfrentar los desafíos de la ingeniería de software y por ayudarnos a crecer como profesionales. A nuestros padres, su amor y apoyo constante son la base de todo lo que hemos logrado. Su sacrificio y dedicación no tienen comparación, y estamos eternamente agradecidos.

Finalmente, extendemos nuestro reconocimiento a amigos, compañeros y a todos aquellos que han sido parte de este logro. Cada contribución ha sido vital en nuestra trayectoria.

Resumen

El presente proyecto está orientado al desarrollo de un prototipo de aplicación web de inventario para el centro de estética Elemental de la ciudad de Neiva, este fue desarrollado usando la pila de desarrollo conocida como MERN (Mongo, Express, React.JS, Node.JS)

La metodología usada para el desarrollo fue ágil Scrum, iniciando con la elaboración de historias de usuarios, las cuales fueron obtenidas por medio de comunicación activa con los empleados de Elemental.

Una vez obtenido los requerimientos se procede a construir un modelo de base de datos, así como los mockups de las interfaces a desarrollar, permitiendo tener una estructura clara de lo que se requiere construir, las historias de usuario fueron divididas en cuatro sprints según su complejidad, en cada sprint se desarrollaron las historias obteniendo partes así nuevas funcionalidades hasta obtener el producto completo.

Finalmente, como resultado del presente proyecto, se entrega un prototipo probado totalmente funcional de aplicación web que permita la gestión de inventario por almacenes.

Palabras claves: Inventario, software, MERN Stack, Aplicación Web, Node.js, MongoDB, Express, React, SPA, Redux.

Abstract

This project is oriented to the development of a prototype inventory web application for the Elemental aesthetic center in the city of Neiva, this was developed using the development stack known as MERN (Mongo, Express, React.JS, Node.JS).

The methodology used for the development was agile Scrum, starting with the development of user stories, which were obtained through active communication with the employees of Elemental.

Once the requirements were obtained, we proceeded to build a database model, as well as the mockups of the interfaces to be developed, allowing to have a clear structure of what is required to build, the user stories were divided into four sprints according to their complexity, in each sprint the stories were developed obtaining parts of the product until the complete product was obtained.

Finally, as a result of this project, we deliver a tested working prototype of a web application that allows inventory management by warehouses.

Keywords: Inventory, software, MERN Stack, Web Application, Node.js, MongoDB, Express, React, SPA, Redux.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

1 Introducción

La empresa Elemental es un centro de estética, fundado en el año 2016 y ubicado en la calle 17 N° 5ª – 73 de la ciudad de Neiva, el cirujano plástico fundador Juan Canencio y sus familiares emprendieron este negocio con el cual han conseguido satisfacer las necesidades de sus clientes y establecerse como un centro de referencia en su área de especialización. A pesar de su éxito, el crecimiento y la expansión del negocio han traído consigo retos y problemas particulares.

Uno de los problemas enfrentados ha sido la gestión periódica inadecuada del inventario, esta problemática se evidencia en varios aspectos como un registro de datos en hojas de caculos en Google Sheets poco estructurado, carencia de uniformidad en los datos, un alto margen para errores humanos, así como obstáculos en la supervisión y manejo del inventario. Tales elementos han contribuido a una administración del inventario poco clara y no fiable, afectando tanto la eficiencia como la exactitud en la gestión de las existencias en los distintos almacenes del centro estético. En este contexto, Ehrhardt y Brigham subrayan la importancia de una administración eficiente del inventario para lograr dos objetivos clave. El primero es garantizar que haya suficiente inventario disponible (stock) para sostener las operaciones de la empresa de manera continua. El segundo objetivo es conservar niveles óptimos de inventario con la finalidad de minimizar los costos totales, que abarcan tanto los gastos asociados con la realización de pedidos como los costos de mantenimiento del inventario. Un inventario insuficiente puede elevar los costos de pedido, mientras que un exceso en el inventario puede incrementar los gastos relacionados con su almacenamiento (Ehrhardt & Brigham, 2007).

En respuesta a estas dificultades, se ha reconocido la necesidad de desarrollar una solución tecnológica personalizada para mejorar la gestión de inventario en Elemental Centro de Estética. Este proyecto aspira a resolver esta problemática al desarrollar un prototipo de aplicación web que facilite una gestión de inventario más precisa, segura y eficiente que reduzca el esfuerzo mensual. Para materializar este proyecto, se ha de llevar a cabo una serie de procesos de ingeniería de software, como el análisis de los requerimientos de Elemental, la identificación de las falencias de los procesos en uso del inventario actuales, el diseño y desarrollo de un prototipo de aplicación web. Todo ello será ejecutado bajo enfoques ágiles y utilizando tecnologías web de vanguardia.

Cabe destacar que el enfoque de este proyecto se limita a la optimización de la gestión de inventario en Elemental Centro de Estética, sin abordar otras áreas operativas de la empresa tales como la contabilidad. No obstante, se espera que esta iniciativa tenga un efecto positivo en el rendimiento general del centro de estética al abordar uno de sus problemas más urgentes.

2 Planteamiento del Problema

2.1 Descripción del Problema

Elemental Centro de Estética, situada en Neiva - Huila y dirigida por el cirujano plástico Juan Canencio, enfrenta dificultades en su gestión de inventarios debido a la ausencia de un sistema digital eficaz. Como se puede observar en la Figura 1, la actual dependencia de hojas de cálculos en Google Sheets se emplea para registrar la disponibilidad (stock) y supervisar los movimientos de los productos en los almacenes ha demostrado tener limitaciones significativas en términos de eficiencia, precisión y seguridad pues se requiere un sobre esfuerzo mensual por

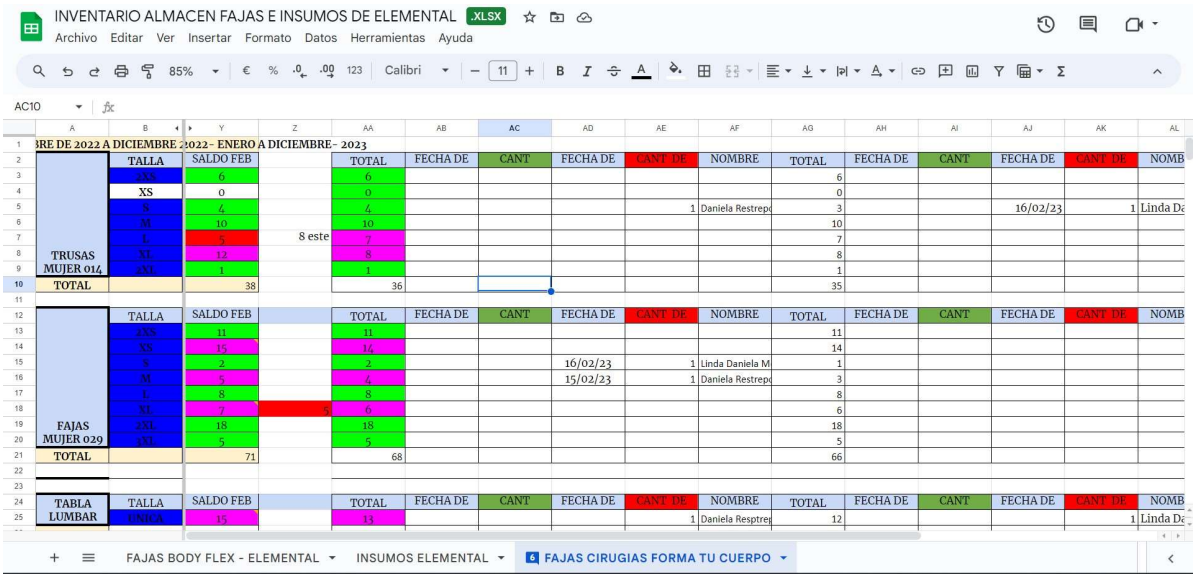
parte del personal de la empresa para la gestión del inventario, lo cual resulta en frecuentes revisiones manuales de los almacenes de la empresa.

Los problemas específicos de esta situación son:

- **Registro de datos del inventario (Libros escritos y Google Sheets):** Almacenar información en documentos físicos como se observa en la Figura 2, y en hojas de cálculo puede comprometer la seguridad de los datos, ya que no cuentan con medidas de seguridad robustas. Estos formatos son susceptibles a ser alterados y podrían estar al alcance de personas no autorizadas, lo que pone en riesgo la integridad y confiabilidad de la información. Además, en el formato de las hojas de cálculo que se usa no ofrece una estructura bien definida para el registro de inventario, lo que complica el análisis y la comprensión de los datos.
- **Falta de Estandarización:** Los datos ingresados por los empleados pueden ser inconsistentes debido a la falta de campos definidos y estandarizados, lo que genera confusión y puede afectar la confiabilidad de los registros.
- **Errores Humanos:** La entrada manual de datos en hojas de cálculo es propensa a errores humanos, tales como duplicaciones, omisiones, o entradas incorrectas, comprometiendo así la precisión y la fiabilidad de los registros.
- **Dificultades en el Seguimiento:** El control de existencia del inventario (stock) se vuelve complicado debido a las limitadas funcionalidades de seguimiento en Google Sheets, puesto que no está automatizado haciendo difícil mantener un estado actualizado del stock de productos.

Figura 1

Registro de movimientos de productos en hojas de calculo

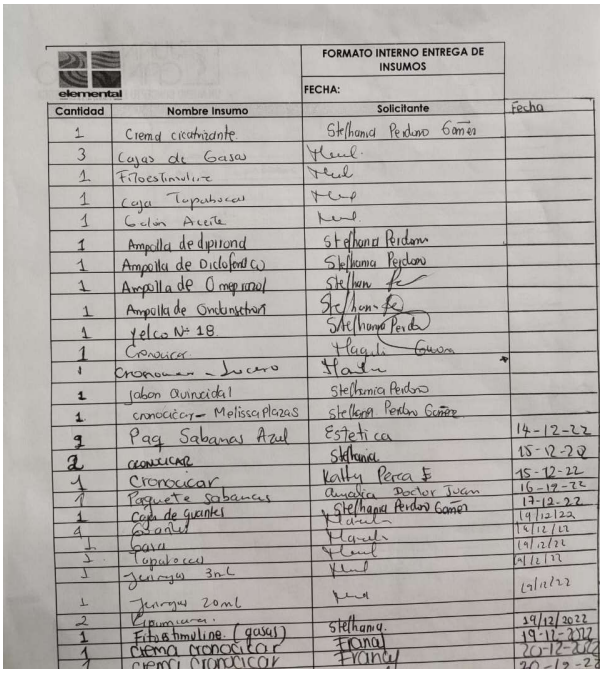


Nota. Fuente: https://docs.google.com/spreadsheets/d/1YPL4tyUtJ0huGwMa1ArgLvlN38_qOxEb/edit#gid=481066295

Para superar estos obstáculos, se plantea la necesidad de desarrollar un prototipo de aplicación web que se ajuste a las necesidades específicas de Elemental centro de estética. Esta herramienta permitiría un acceso inmediato a la información del inventario y mejoraría considerable en la gestión interna de los procesos de inventario.

Figura 2

Formato físico de entrega e ingreso de insumos al inventario



elemental

FORMATO INTERNO ENTREGA DE INSUMOS

FECHA:

Cantidad	Nombre Insumo	Solicitante	Fecha
1	Crema cicatrizante	Stefhanía Perdomo Gómez	
3	Cajas de Gasas	Yleul	
1	Fitoestimulina	Yleul	
1	Caja Tapabocas	Yleul	
1	Galón Aceite	Yleul	
1	Ampolla de dipirona	Stefhanía Perdomo	
1	Ampolla de Diclofenaco	Stefhanía Perdomo	
1	Ampolla de Omeprazol	Stefhan	
1	Ampolla de Ondansetron	Stefhanía	
1	Yelco N° 18	Stefhanía Perdo	
1	Cronocicar	Yluguis Gómez	
1	Cronocicar - Lucero	Yleul	
1	Jabon avinacida!	Stefhanía Perdomo	
1	cronocicar - Melissa Plazas	Stefhanía Perdomo Gómez	
1	Paq Sabanas Azul	Estetica	14-12-22
2	cronocicar	Stefhanía	15-12-22
1	Cronocicar	Kathy Perea E	15-12-22
1	Paquete Sabanas	Angelica Doctor Juan	16-12-22
1	Caja de guantes	Stefhanía Perdomo Gómez	17-12-22
1	Gasas	Yleul	19/12/22
1	Gasa	Yleul	19/12/22
1	Tapabocas	Yleul	19/12/22
1	Jeringas 3mL	Yleul	19/12/22
1	Jeringas 20mL	Yleul	
2	Agujas		
1	Fitoestimulina (gasas)	Stefhanía.	19/12/2022
1	Crema cronocicar	Yleul	19-12-2022
1	Crema cronocicar	Yleul	20-12-2022

Nota. Fuente: Autor.

2.2 Formulación del problema

¿Cómo se puede desarrollar un prototipo de aplicación web que permita a Elemental Centro de Estética superar las limitaciones de eficiencia, precisión y seguridad en sus actuales procesos de gestión de inventario basado en hojas de cálculo de Google Sheets?

3 Justificación

La gestión del inventario es un elemento crítico para el éxito de cualquier empresa, y Elemental Centro de Estética no es una excepción y es que según Martínez Orencio (2017), la información constituye uno de los activos más valiosos en el ámbito empresarial, dado que de ella se derivan numerosas decisiones tomadas por la gerencia y el personal. Una gestión inadecuada de esta información puede resultar en problemas significativos, incluyendo pérdidas económicas relevantes. En la era contemporánea, el hecho de ser una pequeña o mediana empresa (PYME) no es un obstáculo válido para la falta de software que facilite una gestión organizacional de inventario efectiva. Consecuentemente, es imperativo que todas las empresas se orienten hacia la automatización de sus procesos para organizar y acceder a la información de manera eficaz y eficiente.

En el mercado actual, existen múltiples soluciones de software diseñadas para asistir en la gestión de procesos empresariales como lo es el inventario. No obstante, no todas estas herramientas se ajustan a las especificaciones del negocio ni cumplen con todas las preferencias que cada empresa exige. Por tanto, se hace esencial llevar a cabo un análisis y diseño que permita el desarrollo una aplicación web personalizada que esté en consonancia con las necesidades específicas de la empresa.

Con este proyecto se propone abordar esta cuestión mediante la aplicación de gestión de almacenes, control de stock y movimientos de productos, adaptados al entorno específico de Elemental Centro de Estética. Lo anterior mediante el empleo de metodologías ágiles de desarrollo de software, como SCRUM, el proyecto no solo se adapta a las necesidades cambiantes del negocio, sino que también establece un marco para futuras investigaciones y aplicaciones en entornos similares.

Por otro lado, desde el punto de vista tecnológico, el proyecto aspira a desarrollar un prototipo de aplicación web utilizando tecnologías de vanguardia para ofrecer una solución robusta y escalable que supere las limitaciones de los procesos actuales de inventario hechos en Google Sheets. Con el respaldo de tecnologías avanzadas y mejores prácticas de la industria de software, se espera que esta aplicación no solo mejore la eficiencia en la gestión del inventario, sino que también minimice los errores, facilite el acceso y la adopción por parte del personal de Elemental.

La relevancia de este proyecto se amplifica por su enfoque en una necesidad real y urgente, con un propósito bien definido de optimizar un área operativa crítica para Elemental Centro de Estética. Además, al tratar los desafíos con una combinación de avances teóricos, innovaciones tecnológicas y aportes metodológicos, el proyecto tiene el potencial de servir como un caso de estudio valioso para otros centros de estética o pequeñas y medianas empresas que buscan modernizar sus operaciones a través de la tecnología.

4 Objetivos

4.1 Objetivo General:

Diseñar y desarrollar un prototipo de aplicación web basado en la pila de tecnologías MERN para registrar y controlar eficiente, precisa y segura del inventario en Elemental Centro de Estética, buscando optimizar su gestión.

4.2 Objetivos Específicos:

- Comprender y analizar los actuales procesos de inventarios de la empresa Elemental Centro de Estética. Esto implica entender y organizar la información sobre cómo se

realiza actualmente el registro y control del inventario, profundizar en los problemas y limitaciones existentes.

- Analizar y definir los requerimientos para el prototipo de la aplicación web de inventario de Elemental Centro de Estética, utilizando una metodología ágil.
- Diseñar y construir la base de datos que almacenará toda la información generada por la aplicación web.
- Diseñar los diagramas UML para el prototipo de la aplicación web de inventario de Elemental Centro de Estética.
- Desarrollar ágilmente el prototipo de la aplicación web para la gestión de inventario en Elemental Centro de Estética, haciendo uso de la pila MERN.
- Realizar pruebas de funcionamiento al prototipo de la aplicación web para garantizar la calidad y funcionalidad de la aplicación de gestión de inventario en Elemental Centro de Estética.

5 Alcances y Limitaciones

5.1 Alcance

El presente proyecto contempla el desarrollo de un prototipo de aplicación web para Elemental Centro de Estética, con el objetivo principal de mejorar y facilitar su gestión de inventario, enfocándose en registro y control. La aplicación será lo suficientemente flexible como para acomodar funciones esenciales, como registrar, actualizar, buscar y eliminar productos, almacenes, proveedores, clientes y usuarios. Aunque el prototipo ha sido diseñado para poder integrarse con otros sistemas en el futuro, en la primera versión no tendrá integración externa directa. Cabe señalar que el prototipo de aplicación web, en su primera versión no estará programado para recuperar automáticamente datos históricos de herramientas previas como

Google Sheets, aunque permitirá agregar estos datos manualmente. Aunque se ha proporcionado documentación, el soporte técnico continuo después del despliegue no está dentro del alcance de este proyecto, así como la capacitación o implementación inmediata de este.

5.2 Limitaciones

Las limitaciones identificadas para desarrollar el prototipo de aplicación web que optimizará los procesos de inventario en la empresa Elemental centro de estética son las siguientes:

- Existe un presupuesto limitado para la investigación y desarrollo de software. Por lo tanto, se optará por herramientas de software libre para la construcción del prototipo.
- Elemental Centro de Estética carece de metodologías y marcos de trabajo estandarizados en lo que a tecnología de información se refiere.

6 Antecedentes del Problema

Desde la antigüedad, ciertas civilizaciones, particularmente en Egipto, acostumbraban a almacenar grandes cantidades de alimentos para enfrentar periodos de sequía o desastres. "El almacenamiento de alimentos y bienes esenciales para sobrevivir en tales circunstancias, condujo al uso de sistemas de inventario para garantizar la supervivencia de la comunidad, permitiendo así la continuación de actividades necesarias para mantener un almacén de seguridad de sus recursos" (Durán, 2012). Con el advenimiento de las tecnologías y el crecimiento de los negocios a nivel nacional e internacional, las empresas en los últimos tiempos han tenido que transformar significativamente sus procesos de inventario. “Actualmente, muchas organizaciones comerciales están buscando nuevos métodos y herramientas que ayuden a administrar y controlar los inventarios, con el objetivo de tener información precisa para la elaboración de los Estados Financieros, lo que a su vez, facilitará la toma de decisiones informadas” (Cecibel, 2015). Varias

organizaciones renombradas han optado por actualizar y modernizar sus sistemas logísticos y de inventario utilizando tecnología. “A raíz de la crisis actual causada por el impacto del Coronavirus 19, la automatización de los procesos seguirá siendo de vital importancia, no solo porque permitirá a los empleados evitar el contacto con grupos de personas, sino también porque garantizará la reducción de errores y pérdidas de stock, ofreciendo un control más preciso” (Escudero, 2021). En Colombia, la situación no es diferente; las grandes organizaciones siguen necesitando tecnología avanzada para automatizar sus procesos y hay una necesidad palpable de transformar la cadena de suministro integrando recursos tecnológicos que favorezcan el crecimiento organizacional. “Por ejemplo, el uso de aplicaciones móviles, sistemas de radiofrecuencia e incluso la gestión de almacenes (Warehouse Management) están teniendo un impacto positivo en las empresas que los implementan” (Departamento Nacional de Planeación, 2020). Por lo tanto, es crucial en la actualidad considerar la incorporación de tecnologías en los procesos logísticos, ya que esto conlleva múltiples beneficios como una mejor administración de recursos y una toma de decisiones más ágil.

Una aplicación web es una solución de software diseñada para satisfacer las necesidades empresariales utilizando internet como plataforma de interconexión. Mateu (2004) señala que, inicialmente, la web consistía simplemente en una colección de páginas estáticas, documentos, etc., que podían ser consultados o descargados. Sin embargo, De Luca (2013) sugiere que las aplicaciones web son programas que operan en el navegador web y se ejecutan en el lado del cliente, interactuando con la tecnología del lado del servidor para intercambiar datos. Además, las aplicaciones web son simplemente otro tipo de programa que se comunica mediante el protocolo HTTP, están escritas en varios lenguajes de programación, operan en cualquier sistema operativo y pueden comportarse de maneras inimaginables. "En el corazón de cada aplicación web está el

hecho de que toda su funcionalidad se comunica a través de HTTP, sus salidas suelen estar formateadas en HTML y sus entradas se comunican a través de los métodos GET, POST y similares" (Hope & Walther, 2009). Por lo tanto, el uso de aplicaciones web como solución de software implica interactuar con internet para centralizar los procesos empresariales, y al aprovechar adecuadamente estas tecnologías, se pueden alcanzar los objetivos establecidos.

En cuanto a la metodología, Díaz (2009) exploró cómo los enfoques ágiles, utilizando procedimientos colaborativos, impactan en la calidad y otros aspectos del producto de software. El objetivo de la investigación fue demostrar que las metodologías ágiles producen resultados positivos, como la entrega de valor al cliente, la generación de negocio y el retorno de la inversión (ROI). La premisa planteada fue: ¿Cómo influyen las metodologías ágiles en el desarrollo de software? Tras realizar su investigación utilizando técnicas como la observación directa y encuestas, concluyeron que la única manera efectiva de producir software que funcione y genere buenos resultados para el negocio es de forma colaborativa. En este sentido, Navarro Cadavid, Fernández Martínez y Morales Vélez (2013) analizaron la utilidad del enfoque Scrum y su impacto en los proyectos de software. El objetivo principal era explicar el marco y determinar sus beneficios en los proyectos. Se utilizaron técnicas como la observación directa, encuestas y listas de verificación para evaluar su utilidad, concluyendo que el rol del cliente es más prominente en el proceso de desarrollo de las metodologías ágiles, un factor crucial para el éxito del proyecto. “Mike Beedle fue uno de los pioneros en adoptar Scrum y colaboró en su implementación en muchas empresas. Como se sabe, Scrum es básicamente el estándar de facto para el desarrollo ágil” (Griffiths, 2012). Por lo tanto, al elegir Scrum como marco de trabajo, se opta por un marco prácticamente de facto para el desarrollo de software utilizando metodologías ágiles.

7 Marco Teórico

7.1 Gestión de Inventarios

La gestión de inventario desempeña un papel esencial en la operación de las empresas, ya que asegura el suministro adecuado de productos a los clientes con el objetivo de generar beneficios. Empresas de gran envergadura, como IBM, lo definen de la siguiente manera:

> "El inventario es el conjunto de artículos o materiales que un negocio tiene la intención de vender a los clientes con fines lucrativos. La gestión de inventario, un elemento crítico de la cadena de suministro es el seguimiento del inventario desde el momento de su fabricación hasta los almacenes, y desde estas instalaciones hasta el punto de venta. El objetivo de la gestión de inventario es tener los productos correctos en el lugar adecuado y en el momento preciso. Esto requiere visibilidad de inventario: saber cuándo se debe hacer los pedidos, cuánto se debe pedir y dónde almacenar las existencias." (IBM, 2021)

Esta definición resalta la importancia de varios factores clave en la gestión de inventario, como la visibilidad del inventario y la toma de decisiones estratégicas sobre cuándo y cómo reponer los productos. Tales consideraciones son fundamentales para el éxito de cualquier sistema de gestión de inventario, incluido el que se abordará en el presente proyecto.

7.1.1 Tipos de Gestión de Inventario

Existen varios enfoques para manejar el inventario en el mundo empresarial, cada uno con sus propias ventajas y desafíos:

- **Gestión periódica de inventarios:** "El sistema periódico de inventarios es un método de valoración que se utiliza para los informes financieros en el que se realiza un recuento físico del inventario a intervalos específicos. Este método contable toma el inventario al

comienzo de un período, agrega nuevas compras de inventario durante el período y deduce el inventario final para derivar el costo de los bienes vendidos" (IBM, 2021).

- **Gestión de inventario de códigos de barras:** "Las empresas utilizan sistemas de gestión de inventario de códigos de barras para asignar un número a cada producto que venden. Pueden asociar varios puntos de datos al número, incluido el proveedor, las dimensiones del producto, el peso e incluso datos variables, como saber cuántos hay disponibles" (IBM, 2021).
- **Gestión de inventario RFID:** "La identificación por radiofrecuencia o RFID es un sistema que transmite de forma inalámbrica la identidad de un producto en forma de un número de serie único para rastrear artículos y proporcionar información detallada del mismo. El sistema de gestión de almacenes basado en RFID puede mejorar la eficiencia, aumentar la visibilidad del inventario y garantizar un registro automático rápido para la recepción y la entrega" (IBM, 2021).

7.1.2 Stock de Inventario

El término "stock" proviene del inglés y se refiere al conjunto de productos o mercancías que una empresa guarda en almacenamiento con la intención de venderlos o distribuirlos posteriormente. En este contexto, el término de lo que es un stock de inventarios, utilizado también como sinónimo de existencias, hace referencia al conjunto o cantidad de productos que una empresa tiene almacenados, lo que se traduce como una inversión realizada por la empresa para poder dar presta a la demanda o sus necesidades productivas, y desarrollar su activad con normalidad (J, P. P., & Merino, M., 2021).

7.1.3 Sistema de Gestión de Inventario

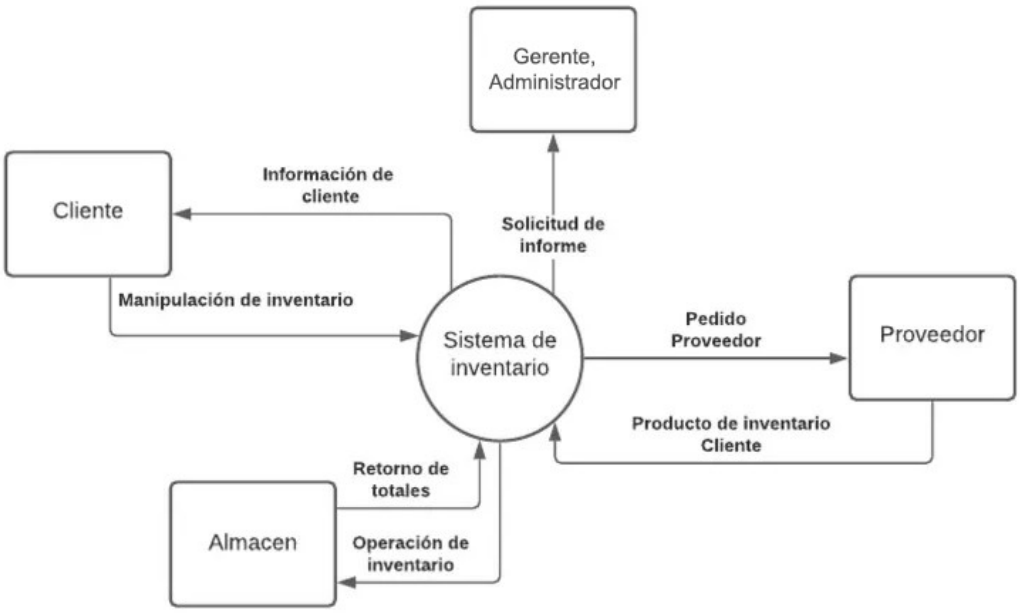
"Las hojas de cálculo, los niveles de existencias contados a mano y la realización manual de pedidos han sido reemplazados en gran medida por un software avanzado de seguimiento de inventario. Un sistema o aplicación de gestión de inventarios puede simplificar el proceso de realización de pedidos, almacenamiento y uso del inventario mediante la producción end-to-end, la gestión del negocio, el pronóstico de la demanda y la contabilidad." (IBM, 2021).

En síntesis, la gestión de inventario moderna no es una operación aislada, sino que parte de un ecosistema entonces un sistema de gestión de inventario debe abarcar diversos actores y componentes clave para su funcionamiento eficiente. Como se puede observar en la Figura 3, el contexto del sistema involucra a múltiples partes interesadas como el cliente, el proveedor, el almacén y el gerente o administrador de inventario.

- El cliente es quien demanda los productos y, por lo tanto, su comportamiento de compra afecta directamente los niveles de inventario.
- El proveedor es la fuente de los productos que se almacenan en el inventario. La calidad del proveedor y la eficiencia de su logística son factores críticos para el sistema.
- El almacén es donde se guarda físicamente el inventario. Requiere una administración eficiente para minimizar los costos y maximizar la eficacia del inventario disponible (stock).
- El administrador de inventario, o gestor, es quien se encarga de coordinar todas las actividades de inventario, desde la adquisición hasta la distribución.

Figura 3

Diagrama de contexto de un sistema de inventario.



Nota. Fuente: https://es.scribd.com/document/518071583/Inventario-Diagrama-de-Contexto

7.2 Metodología de Desarrollo

7.2.1 Metodologías de Desarrollo Ágil

Cuando se habla de enfoques de desarrollo ágiles, es fundamental también abordar las metodologías de desarrollo convencionales, ya que las ágiles surgieron como una respuesta a estas últimas. Las metodologías convencionales suelen estar centradas en la planificación y son especialmente adecuadas para proyectos de gran envergadura. Usualmente, las fases iniciales de estos métodos son el Análisis y el Diseño, seguidos de la validación de requisitos.

Por otro lado, los métodos ágiles ofrecen una mayor flexibilidad en comparación con los enfoques tradicionales. Estos enfoques descomponen un proyecto en segmentos más manejables y ponen el enfoque en las personas más que en los procesos.

La Tabla 1 presenta una comparación entre las características distintivas de las metodologías ágiles y tradicionales.

Tabla 1

Metodologías tradicionales vs agiles

Metodologías tradicionales	***Metodologías agiles***
Predictivos	Adaptativos
Orientados a procesos	Orientado a personas
Proceso rígido	Proceso flexible
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación
Se concibe como un proyecto	Un proyecto es subdivido en varios más pequeños

Nota. Fuente: Cadavid et al. (2013)

Entre las metodologías de desarrollo ágil más comunes y usadas actualmente se encuentran: SCRUM, Extreme Programming, Crystal Methodologies, Adaptive Software Development (ASD).

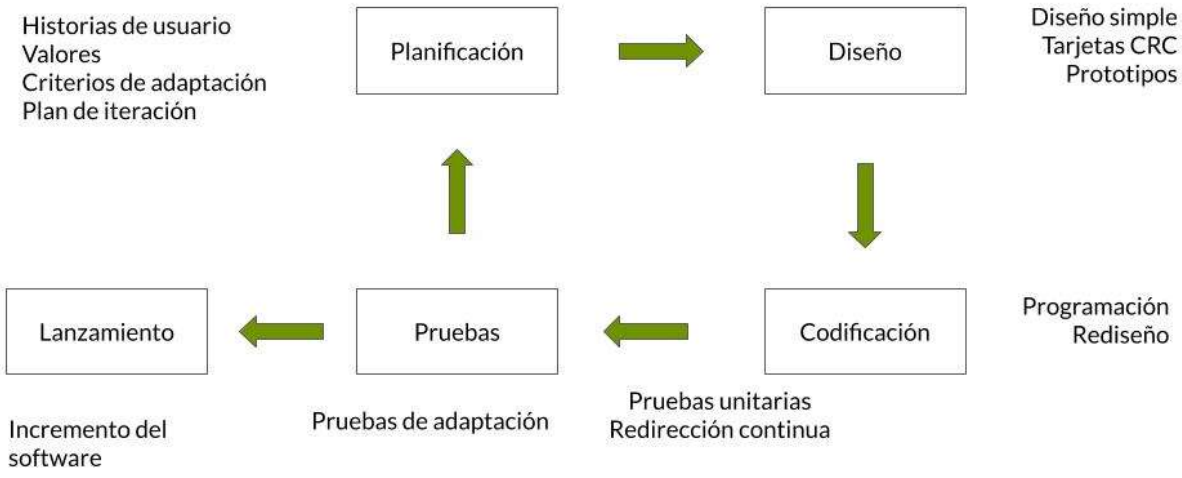
En este proyecto, se empleará Scrum como la metodología de desarrollo, ya que facilita la gestión organizada y controlada de las actividades involucradas en el proyecto, lo que conlleva beneficios como la reducción del tiempo necesario. Scrum promueve la participación de todas las partes interesadas, lo que facilita una integración óptima.

7.2.2 Extreme Programming (XP)

En español programación extrema, es un enfoque ágil para la administración de proyectos que prioriza la rapidez y simplicidad a través de ciclos de desarrollo cortos. Este enfoque se articula en torno a cinco valores fundamentales, cinco normas y doce prácticas de codificación. Aunque es un sistema rigurosamente estructurado, su enfoque en sprints intensivos y en la integración continua aspira a entregar un producto de alta calidad (Raeburn,2022). En la Figura 4 se observa el ciclo ágil la metodología XP.

Figura 4

Diagrama de metodología XP



Nota. Fuente: https://www.sinnaps.com/blog-gestion-proyectos/metodologia-xp

7.2.3 Kanban

La metodología Kanban se implementa utilizando tableros visuales conocidos como tableros Kanban. Esta es una estrategia visual para administrar proyectos que facilita a los equipos ver tanto sus flujos laborales como la cantidad de trabajo en curso. En este tablero, que se organiza mediante columnas, se despliega el progreso del trabajo en un proyecto. Generalmente,

cada columna simboliza una fase diferente del proceso laboral. Como se observa en la Figura 5, un tablero Kanban básico podría tener columnas tituladas como 'Que hacer', 'En curso' y 'Hecho'. Las actividades específicas, ilustradas por medio de tarjetas visuales en el tablero, pasan de una columna a otra hasta que se concluyen (Martins, 2022).

Figura 5

Tablero Kanban

Nota. Fuente: https://tecnosoluciones.com/10-razones-para-usar-la-metodologia-kanban-en-tu-organizacion

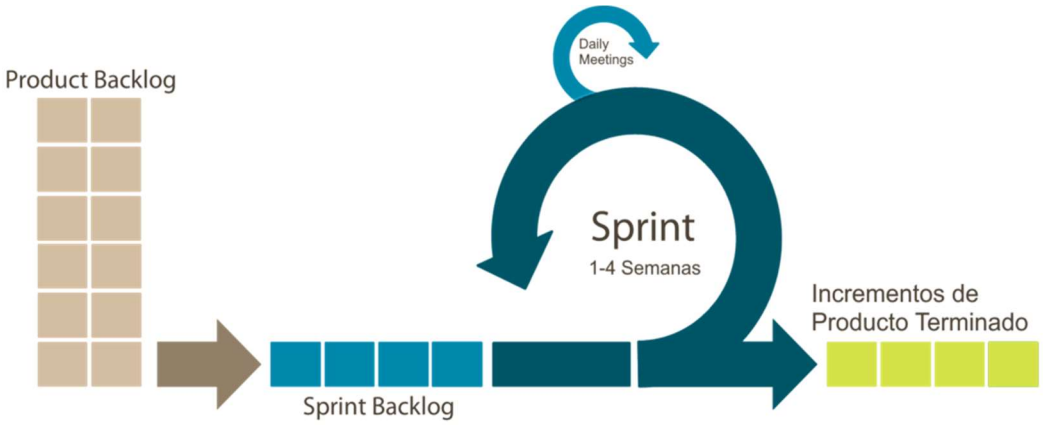
7.2.4 Scrum

Scrum es una metodología ágil para la gestión de proyectos asociados al desarrollo de software. Su creciente aceptación se debe a su flexibilidad, que lo ha transformado en un conjunto de buenas prácticas para la colaboración efectiva en equipo

"Scrum es una metodología para el desarrollo de software iterativa e incremental, debe su nombre a la jugada de rugby llamada de la misma manera, se dice que es iterativa ya que se ejecuta en bloques temporales cortos y fijos" (Cohn, 2010 citado en Salazar et al., 2018, p. 32). Estos bloques temporales se conocen como "Sprints", y de manera incremental generan una porción del producto final en cada iteración como se observa en la Figura 6.

Figura 6

Registro de movimientos de productos



Nota. Fuente: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum

7.2.4.1 Roles de Scrum

Conocer y entender los roles de Scrum es fundamental para asegurar que se implementa de forma exitosa en un proyecto, los principales roles son los siguientes:

- **El Scrum Máster**: responsable de asegurar los procesos.
- **El Dueño del Producto (Product Owner):** responsable de maximizar el valor del producto.
- **El Equipo de Desarrollo (Development Team):** responsable de realizar el trabajo.

Ademas de los tres roles principales tenemos tambien al Stakeholder, "un Stakeholder es cualquier individuo u organización cuyo interés puede verse afectado positiva o negativamente por el proceso de desarrollo de un producto o el producto resultante". Esto se basa en la definición tradicional de "parte interesada" o cliente de proyectos que puede servir como punto de partida. (*Alaimo Labs*, 2022).

7.2.4.2 Artefactos y Herramientas

“Scrum, propone tres herramientas o "artefactos" para mantener organizados los proyectos: Estos artefactos, ayudan a planificar y revisar cada uno de los Sprint, aportando medios ineludibles para efectuar cada una de las ceremonias”. (Bahit, 2012, p. 39)

- **Backlog de Producto:** Esta es una lista en constante evolución y abierta a todos los participantes del proyecto. El Dueño de Producto es el encargado de mantenerla. Aquí es donde se incluyen las conocidas historias de usuario, definidas como requisitos del sistema expresados en el lenguaje que el usuario comprende.
- **Backlog de Sprint:** Según Bahit (2012, p. 45), este backlog es una versión más corta del Backlog de Producto, creada al comienzo de cada sprint. En ella se registran las tareas que el equipo se compromete a completar. La lista se actualiza diariamente, mostrando tareas pendientes, en proceso y terminadas, además del esfuerzo restante para cada tarea no finalizada y quién está encargado de ella.
- **Scrum Taskboard:** Este tablero visual, también conocido como pizarra, sirve para controlar el estado de las tareas pendientes, en proceso y terminadas.
- **Diagrama de Burndown:** Es un gráfico que ilustra visualmente el avance en la realización de los ítems del Backlog de Producto.

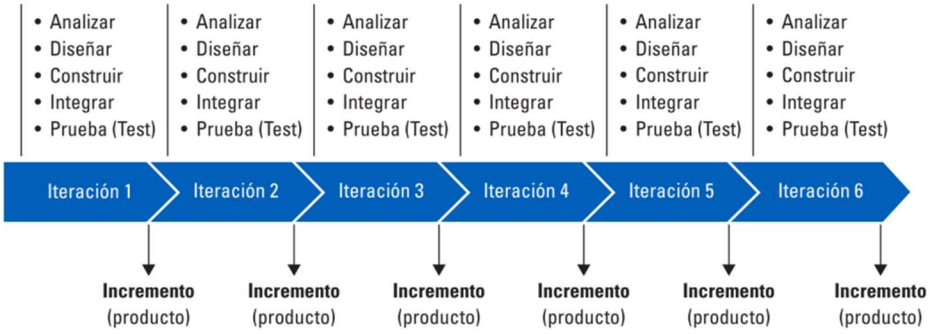
7.2.4.3 Sprint

El sprint es el núcleo de la metodología Scrum. Bahit (2012, p. 30) explica que su objetivo principal es convertir un conjunto de requisitos del cliente en funcionalidades completamente operativas. Los sprints son secuenciales con una duración de entre una y cuatro semanas. Como se puede observar en la Figura 7, en cada iteración o sprint se ejecutar tareas de

análisis, diseño, desarrollo y pruebas, generando asi nuevas funcionalidades que se traducen en el incremento del producto, sistema o aplicación.

Figura 7

Conjunto de Sprint o Interacciones



Nota. Fuente: https://www.belikesoftware.com/agil-verdad-reduce-los-costes-proyecto

7.2.4.4 Ceremonias en Scrum

Bahit (2012) describe distintas ceremonias dentro de Scrum que suceden iterativamente en cada Sprint:

- **Planificación del Sprint:** Aquí, el Dueño del Producto presenta las historias de usuario prioritarias del Backlog de Producto para que el equipo las entienda y se comprometa con ellas (Bahit, 2012, p. 50).
- **Reunión Diaria (Daily):** Son conversaciones cortas, no superiores a 15 minutos, conducidas por el Scrum Máster con cada miembro del equipo.
- **Revisión:** En esta ceremonia, el equipo muestra al Dueño del Producto las funcionalidades desarrolladas. Este último tiene la opción de sugerir mejoras o aprobarlas (Bahit, 2012, p. 54).
- **Retrospectiva:** Aquí se evalúa lo realizado para identificar fortalezas y áreas de mejora, considerada por Bahit (2012, p. 55) como una especie de aprendizaje activo.

Actualmente, Scrum es una de las metodologías más populares no solo en el campo del desarrollo de software, sino también en diversos tipos de proyectos. Su versatilidad para adaptarse a distintas necesidades lo convierte en una opción viable incluso para proyectos personales. En este tipo de iniciativas, una misma persona podría asumir múltiples roles.

7.3 Servicios Web

Con los progresos de la tecnología, los sistemas han tenido que adaptarse y evolucionar, migrando frecuentemente hacia plataformas web. En este entorno, han surgido diversas oportunidades, como la posibilidad de tener servidores de bases de datos y servidores de Backend geográficamente separados y aun así acceder a la información necesaria en cuestión de segundos. Este fenómeno ha sido impulsado en gran medida por los Servicios Web, que han contribuido a la popularización de arquitecturas distribuidas. Según la W3C (2010), los servicios web son "un diseño centrado en mensajes comúnmente encontrado tanto en la web como en el software empresarial, basado en tecnologías como HTTP, XML, SOAP, WSDL, SPARQL, entre otros".

7.3.1 Servicios Web RESTFUL

Es común confundir la arquitectura REST con el servicio web RESTful, pero son conceptos diferentes. Según Red Hat (2020), "Una API RESTful es una interfaz de programación que sigue los principios de la arquitectura REST y permite interactuar con servicios web RESTful". En este contexto, se introduce el término API, que es esencial para la comunicación entre diferentes partes de una aplicación, particularmente en la arquitectura REST bajo el servicio RESTful. Red Hat (2019) define una API como "un conjunto de definiciones y protocolos utilizado para el desarrollo e integración de aplicaciones de software". En resumen, la API pone a disposición los servicios construidos en el servidor (Backend) para ser utilizados por el lado del cliente (Frontend), asegurando así el funcionamiento integral del sistema.

7.3.2 Servicios Web SOAP

En la categoría de servicios web, uno de los más prominentes es SOAP, siglas en inglés para "Simple Object Access Protocol". Según IBM (2019), se define como "un formato de mensaje XML utilizado en interacciones de servicios web. Los mensajes SOAP suelen enviarse sobre HTTP o JMS, aunque otros protocolos también son aplicables". A diferencia de REST, SOAP es un protocolo que impone reglas específicas, aumentando la complejidad y la sobrecarga, lo que puede resultar en tiempos de carga de página más lentos (Red Hat, 2019).

7.4 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual unificado que es rico tanto en sintaxis como en semántica, con el objetivo de facilitar la arquitectura, el diseño y la puesta en marcha de sistemas de software complejos (Lucidchart, 2023). Aunque UML es especialmente útil en el ámbito del desarrollo de software, también se emplea en otros contextos, como la gestión de procesos de fabricación.

Funciona de manera similar a cómo los planos operan en diversas industrias, y se compone de varios tipos de diagramas que sirven para esclarecer los límites del sistema, su estructura y su comportamiento, así como los objetos que lo integran. Aunque UML no es un lenguaje de programación en sí mismo, hay herramientas disponibles que permiten convertir los diagramas UML en código en varios lenguajes de programación. En la actualidad existen 14 tipos de diagramas UML (Lucidchart. 2023), sin embargo, en esta revisión se menciona lo aplicados en la fase de diseño del aplicativo web.

7.4.1 Diagrama de Casos de Usos

El diagrama de casos de uso ilustra cómo un cliente (conocido como Actor) interactúa con el sistema en desarrollo, detallando la naturaleza, categoría y secuencia de estas interacciones (que

se denominan operaciones o casos de uso). Este tipo de diagrama se compone de varios componentes:

- **Actor:** Es relevante subrayar el término "rol" al hablar de un Actor, ya que se refiere a la función que desempeña un usuario en relación con el sistema, más que a un individuo específico.

Figura 8

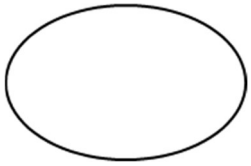
Actor en Casos de Uso

Nota. Fuente: Actor

- **Casos de uso:** Se trata de una actividad o función específica desencadenada por una solicitud de un actor externo, ya sea directamente por un actor o mediante la invocación de otro caso de uso.

Figura 9

Caso de Uso

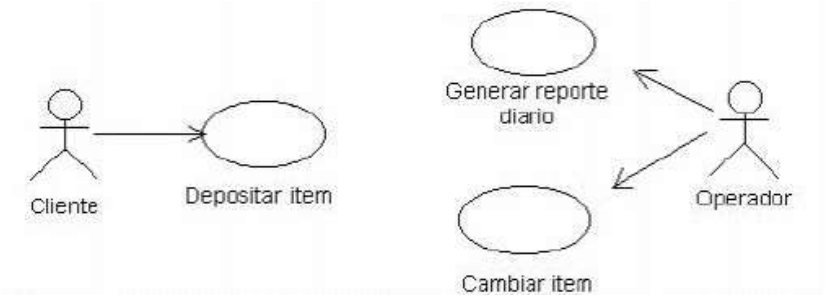


Nota. Fuente: Actor

- **Relaciones como Uso, Herencia y Comunicación:** Es la conexión entre los elementos del modelo.

Figura 10

Ejemplo de un Diagrama de Casos de Uso.



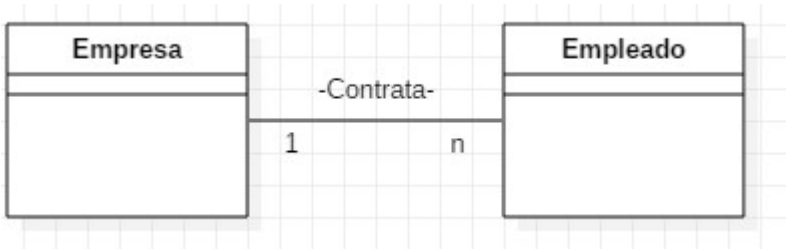
Nota. Fuente: https://unadzsurlab.com/UML/U1/diagramas_de_casos_de_uso.html

7.4.2 Diagrama de Clases

Un diagrama de clases es un instrumento utilizado para transmitir la arquitectura de un programa diseñado con enfoque en la programación orientada a objetos, facilitando la representación de las conexiones entre distintas entidades.

Figura 11

Ejemplo de un Diagrama de Clase



Nota. Fuente https://diagramasuml.com/diagrama-de-clases

7.5 Arquitectura de Software

Aunque no haya un acuerdo unánime sobre la definición exacta de arquitectura de software, todas las interpretaciones convergen en la idea de que se trata de la estructuración del sistema, sus componentes y las interacciones entre ellos. Según Blancarte Iturralde (2020, p. 30), "la arquitectura de software representa el nivel más elevado del diseño de un sistema y se compone de una serie de patrones y conceptos abstractos que ofrecen un marco coherente para la construcción del sistema".

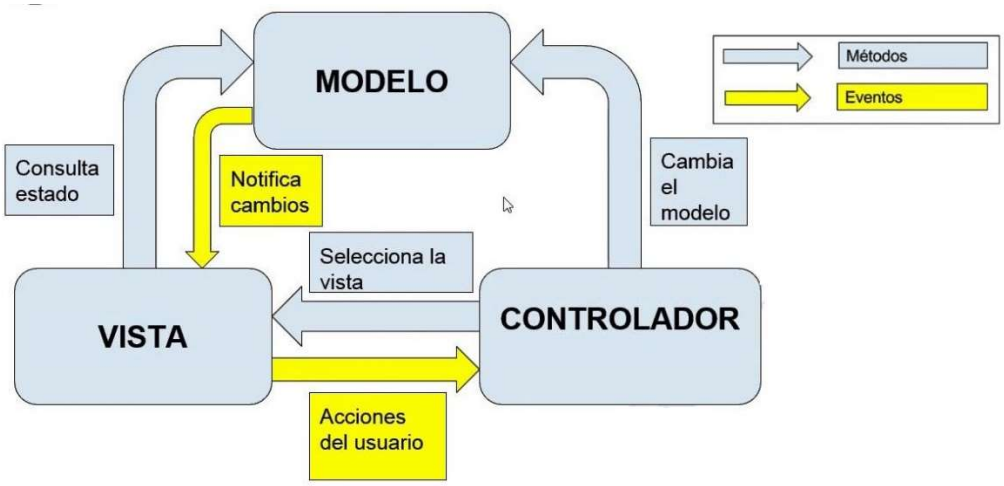
7.5.1 Modelo Vista Controlador (MVC)

El Modelo Vista Controlador (MVC) es un enfoque arquitectónico que fue conceptualizado por Trygve Reenskaug en los laboratorios de investigación de Xerox a finales de la década de 1970 y posteriormente incorporado en la versión Smalltalk-80 del lenguaje de programación Smalltalk. Este patrón se desarrolló con el objetivo de minimizar el esfuerzo en la programación al implementar sistemas múltiples y sincronizar datos. Como se observa en la Figura 12, se distingue por separar las entidades del Modelo, las Vistas y los Controladores, permitiendo que cualquier modificación en el Modelo se refleje de manera automática en todas las Vistas vinculadas a él. Este marco arquitectónico es especialmente útil en aplicaciones que requieren la representación gráfica de datos en diferentes escalas y en ventanas separadas.

Como señala Eslava Muñoz (2013), "MVC es un patrón de diseño que desacopla los componentes de datos y la lógica de negocio de una aplicación del módulo que maneja la interfaz de usuario, eventos y comunicaciones".

Figura 12

Arquitectura MVC



Nota. Fuente: https://i.ytimg.com/vi/z6WppAQ3LUg/maxresdefault.jpg

7.6 Aplicaciones Web

7.6.1 Definición

Una aplicación web o sistema web se define como "una variedad especial de aplicación cliente/servidor en la que tanto el cliente (navegador o explorador) como el servidor (servidor web) y el protocolo de comunicación (HTTP) están estandarizados y no necesitan ser creados" (Luján Mora, 2002, p. 48). Esto implica que un sistema web puede ser accedido desde cualquier navegador. Además, otra característica distintiva de las aplicaciones web son los protocolos que utilizan para operar. Según Lerma Blasco et al., (2013):

> HTTP, el Protocolo de Transferencia de Hipertexto, es el protocolo principal utilizado en la Web. Fue desarrollado en 1989 en el CERN (Laboratorio Europeo de Física de Partículas) como una forma de compartir datos científicos internacionalmente de manera rápida y económica (p.10).

7.6.2 Lenguaje de programación web

En el ámbito de la programación web, la interoperabilidad ha simplificado considerablemente la vida de los desarrolladores, al permitir la integración de servicios escritos en diversos lenguajes en un único servicio web, sin afectar la experiencia del usuario final. Esta flexibilidad en la elección de tecnologías está usualmente a cargo del arquitecto de software, quien toma las decisiones en función de las necesidades del proyecto. En el caso de este proyecto, se optó por una pila tecnológica basada en JavaScript.

- **JavaScript:** JavaScript es principalmente conocido como un lenguaje de programación utilizado para desarrollar sitios web interactivos. Según Pérez (2009, p. 5), "es un lenguaje interpretado, lo que significa que no hay necesidad de compilar el código antes de ejecutarlo, permitiendo que los programas se prueben directamente en cualquier navegador sin pasos adicionales". Aunque su nombre pueda confundirse con Java, son lenguajes independientes. Se utiliza sobre todo en el desarrollo Front-end con librerías como Vue y React, aunque su uso en el desarrollo Back-end también ha ganado terreno gracias al entorno en tiempo de ejecución multiplataforma Node.js.

7.7 Base de Datos

Las bases de datos actúan como depósitos permanentes de información para aplicaciones, permitiendo un acceso rápido y eficiente a los datos. Esta necesidad de almacenar información de forma segura y accesible tiene una larga historia, que se remonta a la invención de la escritura. Ha evolucionado desde el uso de papiro y papel, pasando por cintas magnéticas, hasta llegar a medios digitales como discos duros y la tecnología de almacenamiento en la nube. En este contexto, una base de datos se define como "una colección estructurada y organizada de datos,

generalmente almacenada electrónicamente en un sistema informático, y manejada por un Sistema de Gestión de Bases de Datos (DBMS)" (Oracle, 2022).

7.7.1 Tipos de Bases de Datos

Las tecnologías de bases de datos han avanzado significativamente, dando lugar a una variedad de tipos que se adaptan a distintas necesidades industriales y empresariales. Los más comunes son los modelos relacionales y no relacionales.

- **Base de Datos Orientada a Objetos:** Este tipo de base de datos se centra en el uso de objetos en lugar de tablas. Aquí, las clases actúan como relaciones, las variables como atributos, y los métodos como procedimientos almacenados.
- **Base de Datos Jerárquicas:** Estas son unas de las primeras bases de datos y organizan la información en una estructura de árbol invertido, donde cada nodo puede tener registros secundarios considerados como hijos.
- **Base de Datos Relacionales:** Son las más comunes y fáciles de usar, fundamentadas en el modelo relacional. "Una base de datos relacional consiste en elementos de datos con relaciones preestablecidas entre ellos, organizados en tablas con columnas y filas" (Amazon, 2020). Utilizan conceptos como llaves primarias y foráneas para identificar y relacionar registros únicos en diferentes tablas.
- **Base de Datos no Relacionales (NoSQL):** Han ganado popularidad debido a la necesidad de mayor flexibilidad y escalabilidad. "Están diseñadas para modelos de datos específicos, tienen esquemas flexibles y son reconocidas por su facilidad de desarrollo y rendimiento a gran escala" (Amazon, 2020). Dentro de las bases de datos NoSQL, existen subtipos como:

- **Base de datos de documentos:** En estas se asocia cada clave con una estructura de datos complejas denominada como documento que normalmente se los agrupa en colecciones de documentos similares, ademas es el subtipo que se usa en este proyecto de desarrollo para la empresa Elemental centro estética.
- **Almacenes de grafos:** Son usadas para almacenar información sobre redes de datos, tales como conexiones sociales.
- **Almacenes de clave-valor:** En estas, cada elemento se almacena como un nombre de atributo o clave, junto con su valor.
- **Base de datos orientadas a columnas:** Como Cassandra o HBase, permiten realizar consultas en grandes conjuntos de datos y almacenan los datos en columnas, en vez de filas.

Estos tipos de bases de datos permiten un desarrollo más ágil, una estructuración esquemática sencilla, y la flexibilidad para hacer cambios sin desorganizar el esquema existente. Además, tienen la capacidad de escalar horizontalmente, a diferencia de las bases de datos relacionales que suelen escalar verticalmente.

7.8 Herramientas de Desarrollo

7.8.1 Librería

Una librería o biblioteca en el contexto del desarrollo de software es esencialmente una colección de archivos que sirven para auxiliar en la creación de programas. Estos archivos, que suelen contener tanto código como datos, están diseñados para ser incorporados en otros programas de manera independiente (Gómez, 2021). Las librerías pueden ser caseras o externas, en función de si están creadas por el propio programador de un proyecto o por programadores

externos ajenos al proyecto, pero la principal distinción es entre librerías estáticas y librerías dinámicas.

- **Librerías estáticas:** estas se graban en un programa como ejecutables. Sirven exclusivamente para esto; después, es posible borrarlas sin problemas, ya que el programa seguirá funcionando con la función necesaria.
- **Librerías dinámicas:** son distintas a las estáticas en tanto en cuanto no se copian en el programa al compilarlas. Las subrutinas son cargadas en tiempo de ejecución, en vez de enlazarse en tiempo de compilación.

7.8.2 Framework

Un marco de trabajo, o framework, actúa como una infraestructura sobre la cual los desarrolladores pueden construir aplicaciones para una plataforma en particular. Este suele incluir elementos como clases y funciones ya diseñadas que facilitan tareas como la gestión de entradas de usuario, la interacción con hardware y la comunicación con el software del sistema operativo. Al hacerlo, se elimina la necesidad de que los programadores tengan que codificar cada componente desde cero cada vez que inician un nuevo proyecto, acelerando así el ciclo de desarrollo (Emmita,2017).

7.8.3 Front End

El front end es la parte del desarrollo web que se dedica a la parte frontal de un sitio web, en otras palabras, el diseño de un sitio o aplicación web. A continuación, se presentan las librerías integradas en el proyecto:

7.8.3.1 React.JS

React es una biblioteca de código abierto escrita en JavaScript, creada por Facebook en 2013 con el objetivo de simplificar el desarrollo de componentes interactivos y reutilizables para interfaces de usuario. Uno de los aspectos que lo hace versátil es su capacidad para funcionar tanto en el lado del cliente como en el servidor, lo que facilita una colaboración fluida entre ambos. Esta biblioteca es especialmente eficaz para el desarrollo de aplicaciones web y móviles, así como para la creación de aplicaciones de una sola página, conocidas como Single Page Applications (SPA). Su robusto ecosistema, repleto de componentes, herramientas y módulos, permite el rápido desarrollo de funciones complejas.

Para entender el funcionamiento de React, es importante tener en cuenta el contexto del desarrollo web tradicional, que generalmente implica el conocimiento de tres tecnologías fundamentales: HTML para la estructura de la página, CSS para su diseño, y JavaScript como el 'cerebro' que controla la interactividad. Antes de la llegada de React, estos elementos solían estar separados en distintos archivos y carpetas, lo que complicaba la reutilización y escalabilidad del código. Sin embargo, React resuelve este problema al introducir el concepto de "componente", donde la estructura HTML y el código JavaScript están intrínsecamente vinculados y pueden combinarse con CSS. Esto ha dado lugar al surgimiento de JSX (JavaScript XML), una notación que optimiza el desarrollo de aplicaciones al hacerlas más escalables y manejables (Next U, 2022).

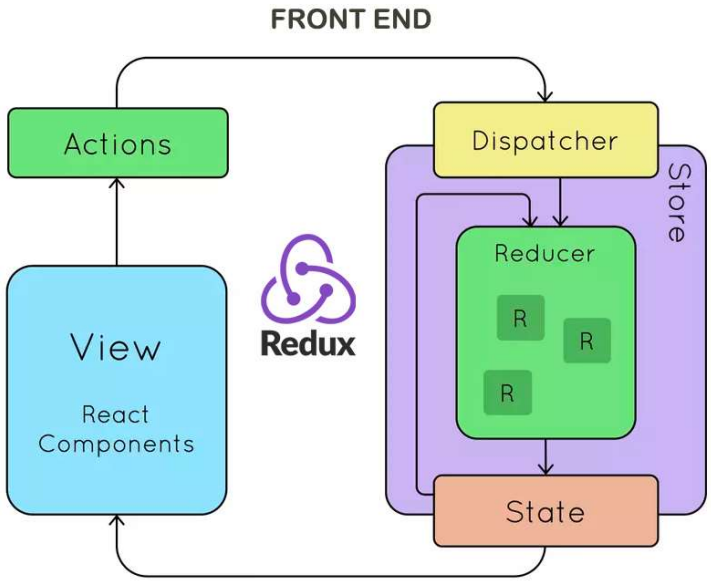
7.8.3.2 Redux

Redux sirve como un depósito confiable para gestionar el estado de aplicaciones creadas en JavaScript. Entre sus fortalezas destacan la inmutabilidad y el control global del estado de la aplicación, así como una arquitectura de datos que se puede escalar. A pesar de su pequeño

tamaño de solo 2KB, Redux ofrece una solución robusta para administrar el estado en diversas aplicaciones de JavaScript (Mirra, 2020). Como se observa en la Figura 13, su enfoque para manejar cambios de estado se basa en tres principios fundamentales: un almacén (Store) como la única verdad absoluta, un estado (State) que es de solo lectura y la modificación del estado solo a través de acciones definidas (actions) y funciones puras (reducers).

Figura 13

Diagrama del Funcionamiento de Redux



Nota. Fuente: https://techriders.tajamar.es/react-con-redux

7.8.3.3 Material UI

Material UI se presenta como una biblioteca de componentes para interfaces de usuario, específicamente diseñada para ser utilizada con React. Su desarrollo se basa en los principios del Material Design, el esquema de diseño desarrollado por Google. Esto se manifiesta en una gama extensa de componentes ya preconfigurados que incluye elementos como botones, formularios, iconos y cuadros de diálogo, entre otros. Lo que distingue a Material UI es su modularidad y configurabilidad. Cada componente es altamente adaptable, permitiendo a los desarrolladores

modificar tanto su funcionalidad como su estética según las necesidades del proyecto. Esta personalización se ve facilitada por un conjunto de utilidades de estilo, diseño y animación, lo que elimina la necesidad de crear hojas de estilo CSS desde cero.

En cuanto a la arquitectura, Material UI emplea una estructura de componentes que se derivan de los componentes básicos de React, como 'div', 'button' y 'input'. Estos se ajustan y personalizan mediante una serie de propiedades que definen su comportamiento y apariencia. En lo que respecta a la estilización, la biblioteca utiliza un enfoque que combina CSS in JS y JSS (JavaScript Style Sheets). JSS es un enfoque para gestionar hojas de estilo directamente dentro de JavaScript, permitiendo que los estilos se definan como objetos JavaScript y luego se apliquen a los componentes de la interfaz. Esta metodología ofrece a los desarrolladores una mayor flexibilidad para personalizar estilos, manteniendo a su vez coherencia estilística a lo largo de la aplicación (Gonzales, J. 2023).

7.8.4 Back End

El back end es la capa de programación que procesa la información, es decir la capa de acceso a los datos y no esta visible para el usuario. A continuación, se presenta el entorno de ejecución y framework empleados en el proyecto:

7.8.4.1 Node.JS

Node.JS es un ecosistema que actúa en el lado del servidor y fue “ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables.” (OpenJS, 2021). Fue desarrollado en el año 2009 por Ryan Dahl y basado en el motor V8 de Google, Node.js es una tecnología en constante evolución que ha ganado relevancia en el ámbito del desarrollo web. Se destaca por ser de código abierto y multiplataforma.

7.8.4.2 Express.JS

Express es el framework para Node más popular y sirve como base para numerosos otros marcos web de Node populares. Ofrece mecanismos para:

- Crear controladores para manejar solicitudes con distintos verbos HTTP en diferentes rutas de URL.
- Integrarse con motores de renderizado de "vistas" para generar respuestas mediante la inserción de datos en plantillas.
- Establecer configuraciones comunes de aplicaciones web, como el puerto para la conexión y la ubicación de las plantillas utilizadas para renderizar la respuesta.
- Añadir "middleware" de procesamiento de solicitudes adicionales en cualquier momento del flujo de manejo de solicitudes.

Aunque Express en sí es bastante minimalista, los desarrolladores han creado paquetes de middleware, también conocido como lógica de intercambio de información entre aplicaciones que son compatibles para abordar casi cualquier problema de desarrollo web. Existen bibliotecas para trabajar con cookies, sesiones, inicio de sesión de usuarios, parámetros de URL, datos POST, encabezados de seguridad y mucho más. Puede encontrar una lista de paquetes de middleware mantenidos por el equipo de Express en Express Middleware (Express.js, 2022).

7.8.4.3 Mongoose

Esta es una biblioteca en JavaScript que simplifica la interacción con la base de datos MongoDB, haciendo más sencillos procesos como añadir, leer, actualizar y borrar datos. Conocida también como un ODM (Object Data Modeling) para MongoDB y Node.js, la

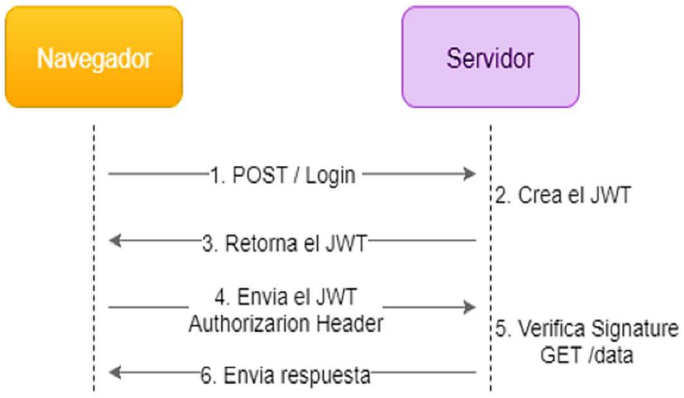
biblioteca maneja las relaciones entre los datos, valida esquemas y actúa como un traductor entre los objetos definidos en el código y cómo estos son representados en MongoDB" (Karnik, 2018).

7.8.4.4 JSON Web Token

El JSON Web Token (JWT) es un estándar abierto (RFC 7519) fundamentado en JSON (JavaScript Object Notation) que se utiliza para la generación de tokens. Estos tokens facilitan la transferencia segura y confiable de datos entre diversas aplicaciones o servicios. Una de las aplicaciones más frecuentes de los JWT es la autenticación de usuarios en entornos web (Auth0 Inc., 2020). En el proceso de autenticación como se observa en la Figura 14, el usuario envía sus credenciales al servidor, que a su vez genera un JWT y lo remite al cliente. Posteriormente, este token se incluye en cada solicitud al servidor para confirmar la identidad del usuario. Sin embargo, los JWT no se limitan a escenarios de autenticación; también pueden usarse para intercambiar cualquier tipo de datos entre diferentes servicios de una aplicación, asegurando su integridad. En términos de estructura, un JWT se compone de tres segmentos: 'header', 'payload' y 'signature'. El 'header' y el 'payload' son cadenas en base64 que se derivan de objetos JSON.

Figura 14

Diagrama de Secuencias del Funcionamiento del JSON Web Token



Nota. Fuente: Autor

` El 'signature' es una encriptación, generalmente mediante el algoritmo SHA-256, de las dos primeras partes. El 'header' generalmente contiene información sobre el algoritmo de encriptación utilizado y el tipo de token. Por su parte, el 'payload' puede incluir cualquier tipo de datos, aunque hay ciertas propiedades estándar, como el emisor ('iss'), la audiencia ('aud') o el tiempo de expiración ('exp'). La 'signature' se crea utilizando una clave secreta y sirve para verificar la autenticidad del token. La implementación de JWTs es especialmente útil en la autenticación de aplicaciones, aunque presenta el desafío de no poder ser revocado una vez emitido, lo cual obliga a establecer tiempos de expiración cortos para mitigar posibles riesgos de seguridad.

7.8.5 Base de Datos

7.8.5.1 MongoDB

MongoDB es una base de datos que se centra en el almacenamiento de datos en formato de documentos, utilizando BSON como su formato binario para representar JSON. A diferencia de las bases de datos relacionales, MongoDB ofrece flexibilidad en la estructura de los datos, permitiendo que documentos dentro de una colección tengan esquemas distintos. Este enfoque es particularmente beneficioso para sistemas que necesitan escalar de forma eficiente, ya que MongoDB facilita procesos como la replicación y el fragmentado (sharding) para lograr una escalabilidad horizontal. Una de las diferencias claves con respeto a las bases de datos relacionada (SQL), es que no tiene que seguir un esquema. Los documentos de una misma colección o concepto similar pueden presentar esquemas diferentes (MongoDB, s.f.).

7.8.6 MERN Stack

Los stacks tecnológicos, también conocidos como 'ecosistemas de datos', son agrupaciones de servicios tecnológicos que se emplean para desarrollar y operar una aplicación

específica, entonces tenemos la pila de tecnologías “MERN Stack” que es un conjunto integrado de tecnologías de software que facilita el desarrollo de aplicaciones. Como se observa en la Figura 15, este stack incluye Mongo DB para la base de datos, Express.js como marco de servidor, React.js para la interfaz de usuario y Node.js para el entorno de ejecución del servidor. Su nombre se deriva de las iniciales de estas cuatro tecnologías.

Esta pila de tecnologías brinda a los desarrolladores la capacidad de construir aplicaciones web o móviles completas. Utiliza React, que puede ser implementado con JavaScript o TypeScript, para gestionar el front-end, mientras que Node.js se encarga del back-end. De esta manera, se logra un control completo sobre tanto el aspecto visual y la experiencia del usuario como la lógica y los algoritmos del servidor (Parada, 2020).

Figura 15

Acrónimo de MERN Stack

Nota. Fuente: https://www.javatpoint.com/mern-stack

7.8.7 Control de Cambios y Manejo de Versiones

7.8.7.1 Git

Git es un sistema de manejo de versiones que opera de forma distribuida, lo cual implica que cada clon local de un proyecto sirve como un repositorio de control de versiones completo e independiente. Esto permite a los programadores trabajar de forma desconectada o en ubicaciones remotas sin inconvenientes. Los desarrolladores efectúan confirmaciones de sus cambios localmente antes de sincronizar sus versiones del repositorio con la del servidor central. Este

enfoque se diferencia del control de versiones centralizado, en el que los usuarios deben coordinar sus cambios con un servidor central antes de generar nuevas versiones del código. La versatilidad y el reconocimiento generalizado de Git lo convierten en una herramienta altamente recomendable para equipos de desarrollo (Mijacobs, 2023).

7.8.7.2 GitHub

GitHub es una plataforma en línea sin costo que ofrece gestión de proyectos y control de versiones para código fuente. Es una herramienta popular entre los programadores para conservar y compartir sus proyectos, permitiendo la colaboración global. Se podría considerar a GitHub como una especie de comunidad en línea diseñada específicamente para los desarrolladores, y es uno de los servicios de repositorio más ampliamente utilizados en todo el mundo (GitHub, s.f.).

7.8.8 Pruebas

En el proceso de desarrollo de sistemas, las pruebas son cruciales porque permiten evaluar de manera detallada las distintas funcionalidades del sistema y, al mismo tiempo, facilitan la detección de errores. Es importante tener en cuenta que existen múltiples tipos de pruebas aplicables al sistema para garantizar su calidad y eficiencia.

Pruebas Unitarias: Las pruebas unitarias están diseñadas para evaluar la funcionalidad de un componente individual del software, como una función o un método (Beck & Gamma, 2000). Su objetivo es confirmar que cada pieza del software funcione según lo diseñado.

- **Pruebas Unitarias:** Las pruebas unitarias están diseñadas para evaluar la funcionalidad de un componente individual del software, como una función o un método (Beck & Gamma, 2000). Su objetivo es confirmar que cada pieza del software funcione según lo diseñado.

- **Pruebas de Humo:** También conocidas como "Smoke Tests," estas pruebas son un conjunto reducido de pruebas que se ejecutan para asegurar que las funciones más críticas del sistema funcionan como se espera (Pressman & Maxim, 2014). Generalmente se llevan a cabo después de cada nueva compilación o versión del software y suelen ser informales o poco documentadas.
- **Pruebas de Portabilidad:** Estas pruebas garantizan que el software se ejecutará correctamente en diferentes entornos de hardware y software. Son esenciales para aplicaciones destinadas a funcionar en múltiples plataformas o dispositivos (International Organization for Standardization, 2005).
- **Pruebas de Aceptación:** Se llevan a cabo para evaluar si el sistema cumple con los requisitos y expectativas del usuario. Estas pruebas son generalmente la última etapa del ciclo de pruebas y son decisivas para la aceptación del producto por parte del cliente (Dustin, 2002).

8 Metodología

El presente proyecto se realiza mediante el tipo de investigación aplicada, puesto que el objetivo es resolver situaciones que se presentan en la realidad aplicando todos los conocimientos adquiridos durante la carrera e investigación básica. Para llevar a cabo este proyecto se emplea una metodología ágil que permita un desarrollo flexible y centrado para pequeños proyectos. En consecuencia, se plantean las siguientes etapas:

8.1 Análisis y Levantamiento de Requerimientos:

- Revisión bibliográfica de las metodologías ágiles: Se efectuó una revisión exhaustiva de diferentes enfoques ágiles, como Scrum, XP, Kanban, y Lean, para identificar el marco de trabajo más apropiado para este proyecto.

- Estudio de proyectos similares: Se examinarán proyectos similares que hayan abordado problemas de gestión de inventario en contextos parecidos.
- Levantamiento de requerimientos: Utilizando la metodología ágil seleccionada, se llevarán a cabo reuniones con el personal de la empresa y analizar los procesos actuales de inventario que realizan en sus almacenes.
- Síntesis de información: Toda la información recolectada se acoplará de forma coherente, alineados con la estructura de la metodología seleccionada.

8.2 Diseño del Prototipo:

- Definición de Escenarios y Funcionalidades del Sistema: Se esquematizará la operación del sistema a través de diagramas de casos de uso y la vista lógica con el diagrama de clases.
- Modelado de la Base de Datos: Se diseñará la estructura de la base de datos acorde a los requisitos identificados.
- Diseño de la Estructura de la Interfaz de Usuario: Se crearán wireframes para el diseño de la interfaz de usuario, tomando en cuenta los requisitos especificados.

8.3 Desarrollo del Prototipo:

- Definición de la arquitectura de software para el desarrollo de la aplicación web.
- Definición de tecnologías y herramientas de desarrollo.
- Creación de la base de datos en base al modelamiento elaborado
- Codificación e integración versionada con Github del back end y front end en paralelo.

8.4 Pruebas de Calidad:

- Construcción de pruebas unitarias para validar cada componente individual del sistema.

- Creación de pruebas de compatibilidad para asegurar la portabilidad del aplicativo.
- Desarrollo de pruebas de validación del usuario para confirmar que el sistema cumple con los requisitos especificados.

9 Desarrollo del Proyecto

El desarrollo de la aplicación web se organizó en torno a cuatro fases: análisis, diseño, desarrollo y pruebas.

9.1 Análisis

Durante esta fase inicial, se estableció la metodología a seguir y se identificaron las necesidades del usuario.

9.1.1 Definición de la Metodología

Tras el análisis bibliográfico sobre las metodologías ágiles descritas en el marco teórico, se eligió Scrum como el enfoque más conveniente para este proyecto. Scrum resulta particularmente efectivo en proyectos de menor escala y con equipos de tamaño reducido. Además, su flexibilidad inherente permite la personalización de sus artefactos, llegando incluso a la posibilidad de omitir algunos en contextos de proyectos pequeños y equipos compactos. Los "sprints", o ciclos de desarrollo cortos, dividen el producto final en entregables múltiples. Esta estructura favorece una interacción constante con el cliente, lo cual facilita la aprobación progresiva de cada componente del proyecto. El equipo Scrum se constituyó y los roles fueron asignados de la siguiente manera:

- **Product Owner:** Juan Ignacio Soto, quien ha mantenido relaciones laborales anteriores con la empresa Elemental.
- **Scrum Master:** Juan Felipe Buitrago.
- **Equipo de Desarrollo:** Integrado por Juan Felipe Buitrago y Juan Ignacio Soto.

- **Stakeholder:** Paola Matajira, quien se desempeña como subdirectora comercial y es la responsable de recursos humanos en la empresa Elemental.

9.1.2 Levantamiento de Requerimientos

Conforme a la metodología Scrum, los requerimientos se extraen de las historias de usuario, que se organizan siguiendo la estructura: "como [rol], quiero [evento] para [funcionalidad]". A partir de estas, se definen los criterios de aceptación pertinentes. Como se observa en la Tabla 2, el proyecto se compone de 17 historias de usuario, cuyos requerimientos se obtuvieron en tres reuniones: dos virtuales de 15 minutos cada una, llevadas a cabo a través de Google Meet, y una reunión presencial que tuvo lugar en el centro de estética. En todas estas reuniones, la participación de Paola Matajira, la Stakeholder y encargada del inventario en el centro de estética, fue crucial. Para cada historia de usuario, se documentan los siguientes elementos:

- **ID:** Funciona como el identificador único de la historia de usuario.
- **Usuario:** Especifica el rol del usuario que protagoniza la historia.
- **Puntos estimados**: Sirven para evaluar la complejidad asociada a la historia.
- **Descripción:** Proporciona un resumen detallado de la funcionalidad o acción que se desea realizar.
- **Criterios de aceptación:** Enumeran las condiciones que deben satisfacerse para que se considere completada la implementación de la historia de usuario.

Tabla 2

Historias de Usuario

	Historia de Usuario
ID: HU-1	**Usuario:** Administrador/Colaborador
Nombre historia: Iniciar sesión	
Puntos estimados: 2	
Iteración asignada: 1	
Descripción: Como usuario, quiero poder iniciar sesión con mi correo y clave asignada para autenticarme ante el sistema, y que en caso de olvidar mi clave que me indique que debo hacer.	
Criterios de aceptación: • El usuario debe ver un formulario donde pueda ingresar su nombre de usuario y contraseña. • El sistema debe validar que ambos campos estén completos antes de permitir el inicio de sesión. • El sistema debe tener la opción de indicar al usuario que debe hacer en caso de olvidar la clave. • Si la autenticación es exitosa, el usuario debe ser redirigido a la página principal. • Si la autenticación falla, el usuario debe recibir una notificación de error.	

	Historia de Usuario
ID: HU-2	**Usuario:** Administrador/Colaborador
Nombre historia: Autenticación y manejo de sesiones	
Puntos estimados: 8	
Iteración asignada: 1	
Descripción: Como usuario, quiero poder autenticarme de forma segura en el sistema y que permita mantener mi sesión de usuario activa hasta que cierre sesión para acceder las funcionales del sistema según mi rol y verificar mi usuario.	
Criterios de aceptación: • Si el inicio de sesión es exitoso, el usuario debe ser autenticado y tener acceso a las funcionalidades del sistema según su rol. • El usuario debe ser capaz de visualizar su nombre de usuario en todo momento mientras la sesión este activa. • El usuario debe ser capaz de cerrar la sesión y desautenticarse del sistema en cualquier momento.	

Historia de Usuario	
ID: HU-3	**Usuario:** Administrador
Nombre historia: Administrar usuarios del sistema	
Puntos estimados: 8	
Iteración asignada: 1	
Descripción: Como administrador, quiero gestionar las cuentas de usuario en el sistema para verificar los usuarios del sistema o crear un nuevo usuario.	
Criterios de aceptación: • El administrador debe ser capaz de ver una lista de todos los usuarios del sistema. • El administrador debe ser capaz de buscar usuarios específicos por nombre. • El administrador debe ser capaz de agregar, editar o eliminar usuarios. • El administrador debe ser capaz de editar las contraseñas de los usuarios.	

Historia de Usuario	
ID: HU-4	**Usuario:** Administrador
Nombre historia: Crear un nuevo usuario	
Puntos estimados: 3	
Iteración asignada: 1	
Descripción: Como administrador, quiero crear un nuevo usuario con nombre, correo y clave en el sistema para otorgar acceso a nuevos encargados y asignarles un rol.	
Criterios de aceptación: • El administrador debe ser capaz de registrar en un formulario el correo electrónico, el nombre, la contraseña y el rol. • El sistema debe validar que los campos requeridos estén completos. • Una vez que se ingresen todos los datos, el administrador debe ser capaz de registrar el usuario en el sistema.	

Historia de Usuario	
ID: HU-5	**Usuario:** Administrador
Nombre historia: Administrar productos	
Puntos estimados: 7	
Iteración asignada: 3	
Descripción: Como administrador, quiero administrar la lista de productos disponibles en el inventario total de todos los almacenes para tener la visión general de todos los productos.	

Criterios de aceptación: • El administrador debe ser capaz de ver una lista/tabla de todos los productos. • El administrador debe ser capaz de buscar productos específicos por diferentes criterios. • El administrador debe ser capaz de agregar, editar o eliminar productos. • Cada acción (agregar, editar, eliminar) debe ser confirmada con una notificación o alerta.

	Historia de Usuario
ID: HU-6	**Usuario:** Administrador
Nombre historia: Crear un nuevo producto	
Puntos estimados: 6	
Iteración asignada: 3	
Descripción: Como administrador, quiero crear un nuevo producto en el inventario para registrarlo en los almacenes con su nombre, presentación, proveedor y categoría.	
Criterios de aceptación: • El administrador debe ser capaz de acceder a un formulario donde pueda ingresar todos los detalles relevantes del producto. • El administrador debe ser capaz de seleccionar categorías, almacenes y proveedores existentes o agregar nuevos si es necesario. • Una vez que se ingresen todos los detalles, el administrador debe ser capaz de guardar el producto en el sistema. • Tras guardar el producto, el administrador debe recibir una notificación confirmando que el producto se creó con éxito o si hubo algún error.	

	Historia de Usuario
ID: HU-7	**Usuario:** Administrador
Nombre historia: Administrar categorías de productos	
Puntos estimados: 4	
Iteración asignada: 3	
Descripción: Como administrador, quiero gestionar las categorías de productos que son etiquetas para saber si son fajas, insumos, cronocicar, mesoterapia y pevonia en el sistema y clasificar adecuadamente los productos en diferentes categorías.	
Criterios de aceptación: • El administrador debe ser capaz de ver una lista/tabla de todas las categorías. • El administrador debe ser capaz de buscar categorías específicas por diferentes	

criterios.

- El administrador debe ser capaz de agregar, editar o eliminar categorías.
- Cada acción (agregar, editar, eliminar) debe ser confirmada con una notificación o alerta.

Historia de Usuario	
ID: HU-8	**Usuario:** Administrador
Nombre historia: Crear una nueva categoría de productos	
Puntos estimados: 4	
Iteración asignada: 3	
Descripción: Como administrador, quiero crear una nueva categoría en el sistema para organizar y clasificar adecuadamente los productos en diferentes categorías.	
Criterios de aceptación: • El administrador debe ser capaz de acceder a un formulario donde pueda ingresar el nombre y la descripción de la nueva categoría. • El sistema debe validar que los campos requeridos estén completos antes de permitir la creación de la categoría. • Una vez que se ingresen todos los detalles, el administrador debe ser capaz de guardar la categoría en el sistema. • Tras guardar la categoría, el administrador debe recibir una notificación confirmando que la categoría se creó con éxito o si hubo algún error.	

Historia de Usuario	
ID: HU-9	**Usuario:** Administrador
Nombre historia: Administrar almacenes	
Puntos estimados: 5	
Iteración asignada: 2	
Descripción: Como administrador, quiero gestionar la lista de almacenes disponibles en el sistema para asegurarme de que toda la información de los almacenes esté actualizada y sea correcta.	
Criterios de aceptación: • El administrador debe ser capaz de ver una lista/tabla de todos los almacenes. • El administrador debe ser capaz de buscar almacenes específicos por diferentes criterios. • El administrador debe ser capaz de agregar, editar o eliminar almacenes. • Cada acción (agregar, editar, eliminar) debe ser confirmada con una notificación o alerta.	

Historia de Usuario	
ID: HU-10	**Usuario:** Administrador
Nombre historia: Crear un nuevo almacén	
Puntos estimados: 6	
Iteración asignada: 2	
Descripción: Como administrador, quiero crear un nuevo almacén en el sistema para gestionar adecuadamente los productos en diferentes ubicaciones.	
Criterios de aceptación: • El administrador debe ser capaz de acceder a un formulario donde pueda ingresar el nombre y la ubicación del nuevo almacén. • El sistema debe validar que los campos requeridos estén completos antes de permitir la creación del almacén. • Una vez que se ingresen todos los detalles, el administrador debe ser capaz de guardar el almacén en el sistema. • Tras guardar el almacén, el administrador debe recibir una notificación confirmando que el almacén se creó con éxito o si hubo algún error.	

Historia de Usuario	
ID: HU-11	**Usuario:** Administrador
Nombre historia: Administrar productos por almacén	
Puntos estimados: 8	
Iteración asignada: 4	
Descripción: Como administrador, quiero gestionar y visualizar la lista de productos disponibles en un almacén específico, con opción de filtrar la lista por categoría o nombre de producto para asegurarme de que la información de cada producto en el almacén esté correcta y tener la opción de trasladar los productos entre almacenes o exportar a excel la lista de productos y su stock del almacén.	
Criterios de aceptación: • El administrador debe ser capaz de seleccionar un almacén específico y ver una lista de todos los productos con la información del producto en ese almacén. • El administrador debe ser capaz de buscar productos específicos por nombre o categoría dentro del almacén seleccionado. • El administrador debe tener la opción de exportar la lista de productos en una hoja de cálculo. • El administrador debe ser capaz de trasladar productos entre almacenes.	

Historia de Usuario	
ID: HU-12	**Usuario:** Administrador

Nombre historia: Administrar proveedores de productos
Puntos estimados: 7
Iteración asignada: 2
Descripción: Como administrador, quiero buscar por nombre y gestionar los proveedores de productos en el sistema para asegurarme de que toda la información de los proveedores esté actualizada y sea correcta.
Criterios de aceptación: • El administrador debe ser capaz de ver una lista de todos los proveedores. • El administrador debe ser capaz de buscar proveedores por nombre. • El administrador debe ser capaz de agregar, editar o eliminar proveedores. • Cada acción de eliminar o editar proveedores debe ser confirmada con una notificación o alerta.

	Historia de Usuario
ID: HU-13	**Usuario:** Administrador
Nombre historia: Crear nuevo proveedor de productos	
Puntos estimados: 6	
Iteración asignada: 2	
Descripción: Como administrador, quiero crear un nuevo proveedor en el sistema para registrarlo en sistema con su nombre y contacto.	
Criterios de aceptación: • El administrador debe ser capaz de acceder a un formulario donde pueda ingresar el nombre y contacto del nuevo proveedor. • El sistema debe validar que los campos requeridos estén completos antes de permitir la creación del proveedor. • Una vez que se ingresen todos los detalles, el administrador debe ser capaz de guardar el proveedor en el sistema. • Tras guardar el proveedor, el administrador debe recibir una notificación confirmando que el proveedor se creó con éxito o si hubo algún error.	

	Historia de Usuario
ID: HU-14	**Usuario:** Administrador
Nombre historia: Administrar pacientes (clientes)	
Puntos estimados: 3	
Iteración asignada: 7	
Descripción: Como administrador, quiero buscar por nombre y gestionar los pacientes en el sistema para asegurarme de que la información de los pacientes esté actualizada y sea	

correcta.
Criterios de aceptación: • El administrador debe ser capaz de ver una lista de todos los pacientes. • El administrador debe ser capaz de buscar pacientes específicos por nombre. • El administrador debe ser capaz de agregar, editar o eliminar pacientes. • Cada acción de editar o eliminar debe ser confirmada con una notificación o alerta.

Historia de Usuario	
ID: HU-15	**Usuario:** Administrador
Nombre historia: Crear nuevo paciente (cliente)	
Puntos estimados: 6	
Iteración asignada: 3	
Descripción: Como administrador, quiero crear un nuevo paciente con su nombre y contacto en el sistema para mantener un registro actualizado de todos los pacientes.	
Criterios de aceptación: • El administrador debe ser capaz de acceder a un formulario donde pueda ingresar el nombre y contacto del nuevo paciente. • El sistema debe validar que los campos requeridos estén completos antes de permitir la creación del paciente. • Una vez que se ingresen todos los detalles, el administrador debe ser capaz de guardar el paciente en el sistema. • Tras guardar el paciente, el administrador debe recibir una notificación confirmando que el paciente se creó con éxito o si hubo algún error.	

Historia de Usuario	
ID: HU-16	**Usuario:** Administrador / Colaborador
Nombre historia: Realizar un movimiento en el inventario	
Puntos estimados: 9	
Iteración asignada: 4	
Descripción: Como usuario administrador o colaborador, quiero realizar movimientos de productos para registrar entrada y salida de productos de cada almacén, en el registro deseo que pueda escribir la factura, proveedor o paciente, notas y verificar la existencia en inventario.	
Criterios de aceptación: • El usuario debe ser capaz de seleccionar una fecha para la transacción. • El usuario debe ser capaz de verificar detalles relacionados con el producto, como el nombre, la talla/presentación, el almacén y el stock. • El usuario debe ser capaz de digitar la cantidad a ingresar o egresar. • El usuario debe ser capaz de seleccionar el paciente al momento de realizar egresos.	

• El usuario debe ser capaz de digitar la factura y agregar notas al movimiento. • El movimiento debe ser registrado y reflejado en el histórico de movimientos de inventario. • El usuario debe recibir una notificación si el movimiento ha sido realizado exitosamente.

Historia de Usuario	
ID: HU-17	**Usuario:** Administrador / Colaborador
Nombre historia: Visualizar movimientos de inventario	
Puntos estimados: 4	
Iteración asignada: 4	
Descripción: Como usuario administrador y colaborador, quiero visualizar una lista detallada del registro de movimientos en el inventario que no pueda ser editada con opciones de búsqueda para verificar los ingresos y egresos de los productos realizados por cada empleado, y que se pueda sacar en excel e imprimir.	
Criterios de aceptación: • El usuario debe ser capaz de ver una tabla de los movimientos con columnas para la fecha, tipo de movimiento, nombre del producto, presentación/talla, cantidad, almacén, paciente o proveedor, factura, usuario y notas. • La tabla debe ser capaz de mostrar múltiples movimientos y permitir la navegación entre diferentes páginas si hay muchos registros. • La tabla debe ser capaz de ser filtrada por columnas para facilitar la búsqueda de un movimiento especifico. • La tabla debe ser capaz de ser exportada en una hoja de cálculo. • La tabla debe tener la opción de ser impresa.	

Nota. Fuente: Autor

9.1.3 Síntesis de la Información

Una vez obtenidos todas las historias de usuarios, estas se ordenan en el Product Backlog, como se observa en la Tabla 4, esto permite tener una visión de todo lo que se quiere hacer, tener claras las prioridades del desarrollo y el estado de cada tarea. Entre el equipo, se estipula la evaluación de cada historia de usuario por prioridad clasificadas en la Tabla 3 y puntos estimados.

Tabla 3

Clasificación de Prioridades en Historias de Usuario

Prioridad	Concepto
Máxima	La historia de usuario es crucial para el funcionamiento de la aplicación y debe abordarse lo más pronto posible.
Alta	La historia de usuario es importante para el funcionamiento general de la aplicación y su implementación afecta a otras historias.
Media	a historia de usuario representa una funcionalidad no esencial del aplicativo y su desarrollo podría posponerse si fuera necesario.
Baja	La historia de usuario cumple con una función que no altera el funcionamiento del aplicativo, pero se requiere para cumplir con todas las expectativas del usuario.

Nota. Fuente: Autor

Los puntos estimados o "story points" se asignan en base a una estimación del esfuerzo requerido para completar cada historia de usuario. Se utiliza una escala de 1 a 10, donde '1' indica una historia de baja complejidad y '10' representa una historia altamente compleja que requeriría un nivel de conocimiento y esfuerzo significativos para su desarrollo.

Tabla 4

Product Backlog del Proyecto

ID	Enunciado de la Historia	Puntos Estimados	Estado	Iteración (Sprint)	Prioridad
HU-1	Iniciar sesión	2	Finalizada	1	Alta
HU-2	Autenticación y manejo de sesiones	8	Finalizada	1	Maxima
HU-3	Administrar usuarios del sistema	8	Finalizada	1	Media
HU-4	Crear un nuevo usuario	3	Finalizada	1	Maxima
HU-5	Administrar productos	7	Finalizada	3	Media
HU-6	Crear un nuevo producto	6	Finalizada	3	Media
HU-7	Administrar categorías de productos	4	Finalizada	2	Alta
HU-8	Crear una nueva categoría de productos	4	Finalizada	2	Alta
HU-9	Administrar almacenes	5	Finalizada	2	Media
HU-10	Crear un nuevo almacén	6	Finalizada	2	Alta
HU-11	Administrar productos por almacén	8	Finalizada	4	Baja
HU-12	Administrar proveedores de productos	7	Finalizada	2	Alta
HU-13	Crear nuevo proveedor de productos	6	Finalizada	2	Alta
HU-14	Administrar pacientes (clientes)	7	Finalizada	3	Media
HU-15	Crear nuevo paciente (cliente)	6	Finalizada	3	Media
HU-16	Realizar un movimiento en el inventario	9	Finalizada	4	Media
HU-17	Visualizar movimientos de inventario	4	Finalizada	4	Baja

Nota. Fuente: Autor

9.1.3.1 Plan de Entrega (Sprint BackLog)

El empleo del Product Backlog facilitó la determinación del número total de sprints necesarios para el desarrollo completo del proyecto. Se estableció que serían cuatro sprints, cada uno con una duración de dos semanas (13 días). Debido a la variabilidad en los horarios de trabajo a lo largo del proyecto, optamos por crear un Sprint Backlog que fuera tanto simplificado como flexible, ilustrado en la Tabla 5, en donde destaca la estimación de esfuerzo y los periodos de duración de cada tarea.

Tabla 5

Sprint Backlog del proyecto

Sprint	ID	Descripción de tarea	Estimación	Fecha Inicio	Fecha Fin
1	HU-1	Iniciar sesión	2	3/04/2023	12/04/2023
	HU-2	Autenticación y manejo de sesiones	8	5/04/203	10/04/2023
	HU-4	Crear un nuevo usuario	5	3/04/2023	7/04/2023
	HU-3	Administrar usuarios del sistema	8	10/04/2023	15/04/2023
2	HU-9	Administrar almacenes	5	17/04/2023	22/04/2023
	HU-10	Crear un nuevo almacén	6	17/04/2023	18/04/2023
	HU-12	Administrar proveedores de productos	7	18/04/2023	25/04/2023
	HU-13	Crear nuevo proveedor de productos	6	18/04/2023	23/04/2023
	HU-7	Administrar categorías de productos	4	19/04/2023	27/04/2023
	HU-8	Crear una nueva categoría de productos	8	22/04/2023	29/04/2023
3	HU-5	Administrar productos	8	1/05/2023	11/05/2023
	HU-6	Crear un nuevo producto	6	1/05/2023	4/05/2023
	HU-15	Crear nuevo paciente (cliente)	6	8/05/2023	10/05/2023
	HU-14	Administrar pacientes (clientes)	7	8/05/2023	13/05/2023
4	HU-16	Realizar un movimiento en el inventario	9	15/05/2023	26/05/2023
	HU-17	Visualizar movimientos de inventario	4	25/05/2023	27/05/2023
	HU-11	Administrar productos por almacén	8	19/05/2023	27/05/2023

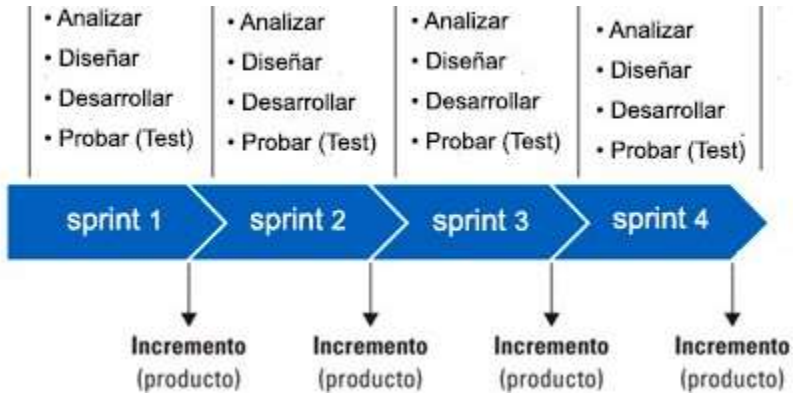
Nota. Fuente: Autor

En los cuatro sprints del proyecto se realizaron las actividades de ingeniería, tal como se ilustra en la Figura 16. En cada sprint, el objetivo es alcanzar un desarrollo incremental del producto, en este caso, un prototipo de aplicación web de gestión de inventario. La stakeholder, Paola Matajira, se mostró disponible durante 15 minutos (daily) de cada sábado por medio de reuniones virtuales en Google Meet para observar los avances de las funcionalidades y diseño del

producto, lo anterior expuesto por el equipo de desarrollo que a su vez al cumplir paralelamente los roles de product owner y scrum master al finalizar cada sprint se reunían para la revisión de este. Durante cada sprint, el equipo de desarrollo realizó pruebas de humo de manera informal para verificar las funcionalidades clave, basándose en los criterios de aceptación de cada historia de usuario. No obstante, las pruebas unitarias, de portabilidad y de aceptación se llevaron a cabo luego del ciclo Scrum. Esto se hizo para documentar de manera formal los resultados logrados en el producto final, permitiendo que la Stakeholder y equipo de desarrollo evaluara de manera formal todas las funcionalidades y dieran su aprobación final para la entrega del prototipo que se encuentra documentado en la fase de prueba.

Figura 16

Actividades en cada sprint del proyecto.



Nota. Fuente: Autor.

9.2 Diseño

En esta fase se abordan varios aspectos esenciales que servirán de guía para la implementación posterior. El diseño del prototipo se segmenta en tres componentes clave: diagramas de casos de uso, diagrama de clases y diseño de la estructura de la interfaz de usuario a través de wireframes.

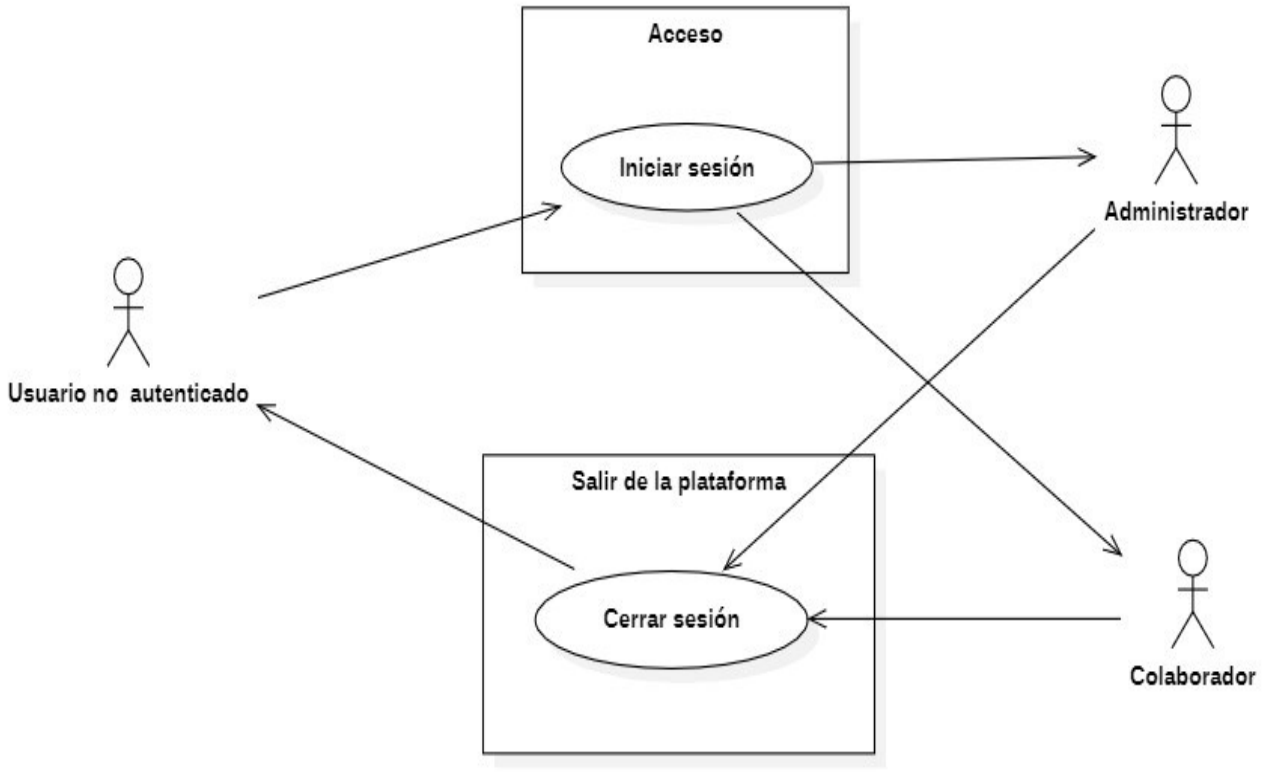
9.2.1 Definición de Escenarios y Funcionalidades del Sistema

Conforme a las historias de usuario, se definió los escenarios y funcionalidades del sistema a través de dos técnicas de modelado UML, diagramas de casos de usos y diagrama de clases elaborados con la herramienta StarUML.

Las Figuras 17 a 23 ilustran la operatividad del sistema en sus diversos módulos, detallando los actores, que corresponden a los roles asignados, así como las actividades que ejecutan, mientras que la Figura 24 ilustra una vista lógica de las funcionales.

Figura 17

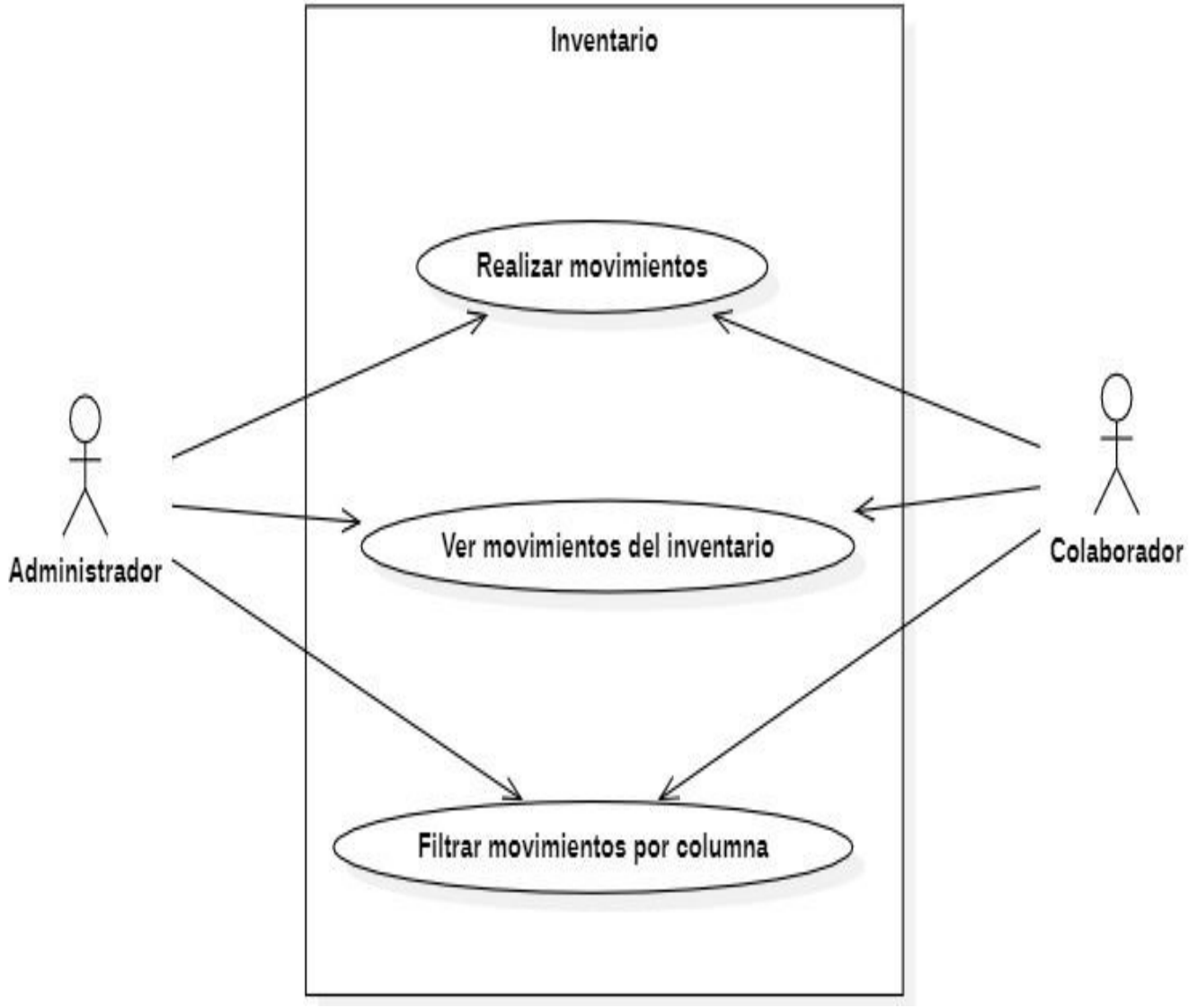
Diagrama de Caso de Uso: Inicio y Cierre de Sesión



Nota. En el inicio y cierre de sesión se muestra la expresión de 'Usuario no autenticado', sin embargo, no representa ningún rol en el sistema. Fuente: Actor.

Figura 18

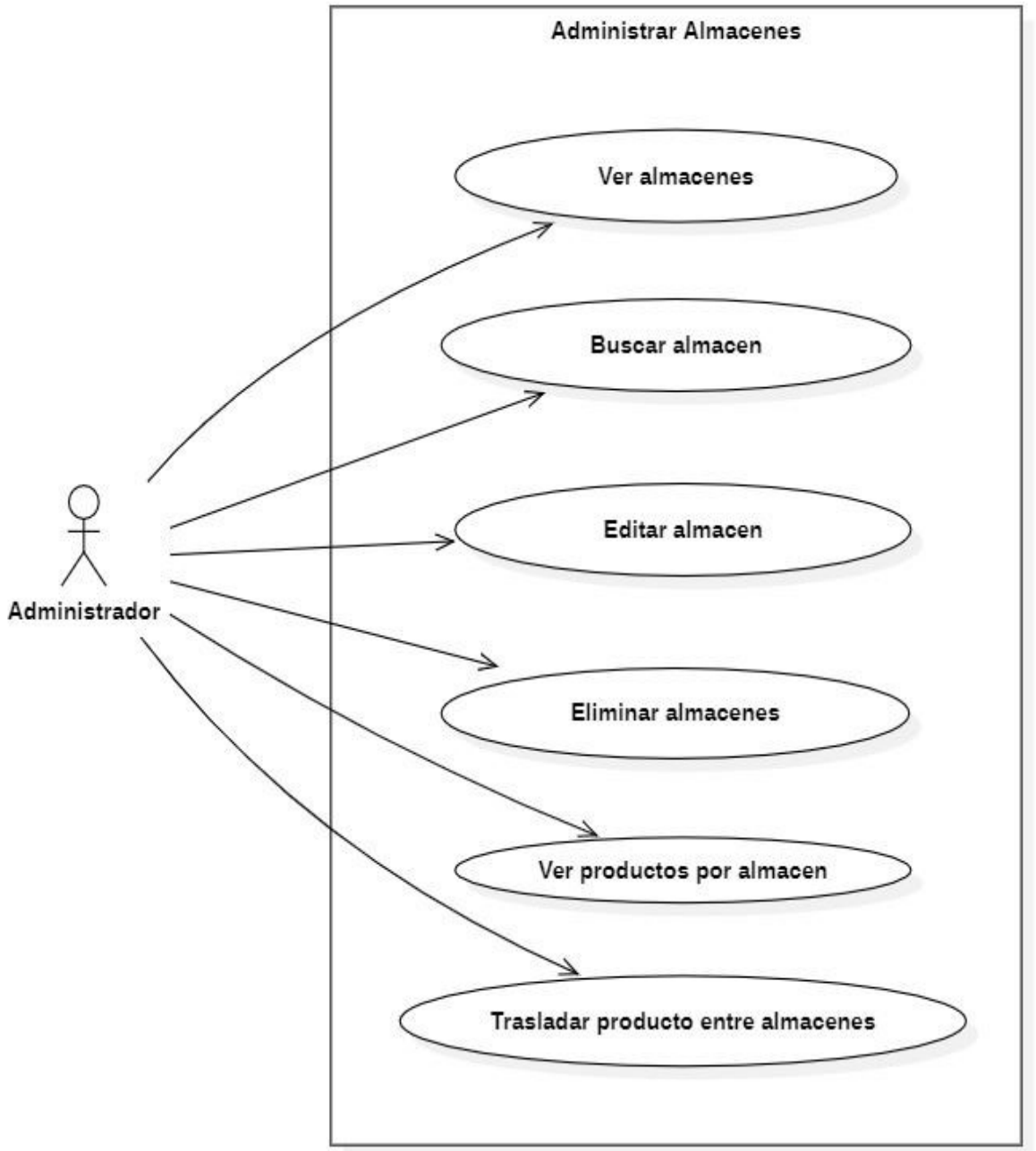
Diagrama de Caso de Uso: Movimientos de Inventario.



Nota. Fuente: Actor

Figura 19

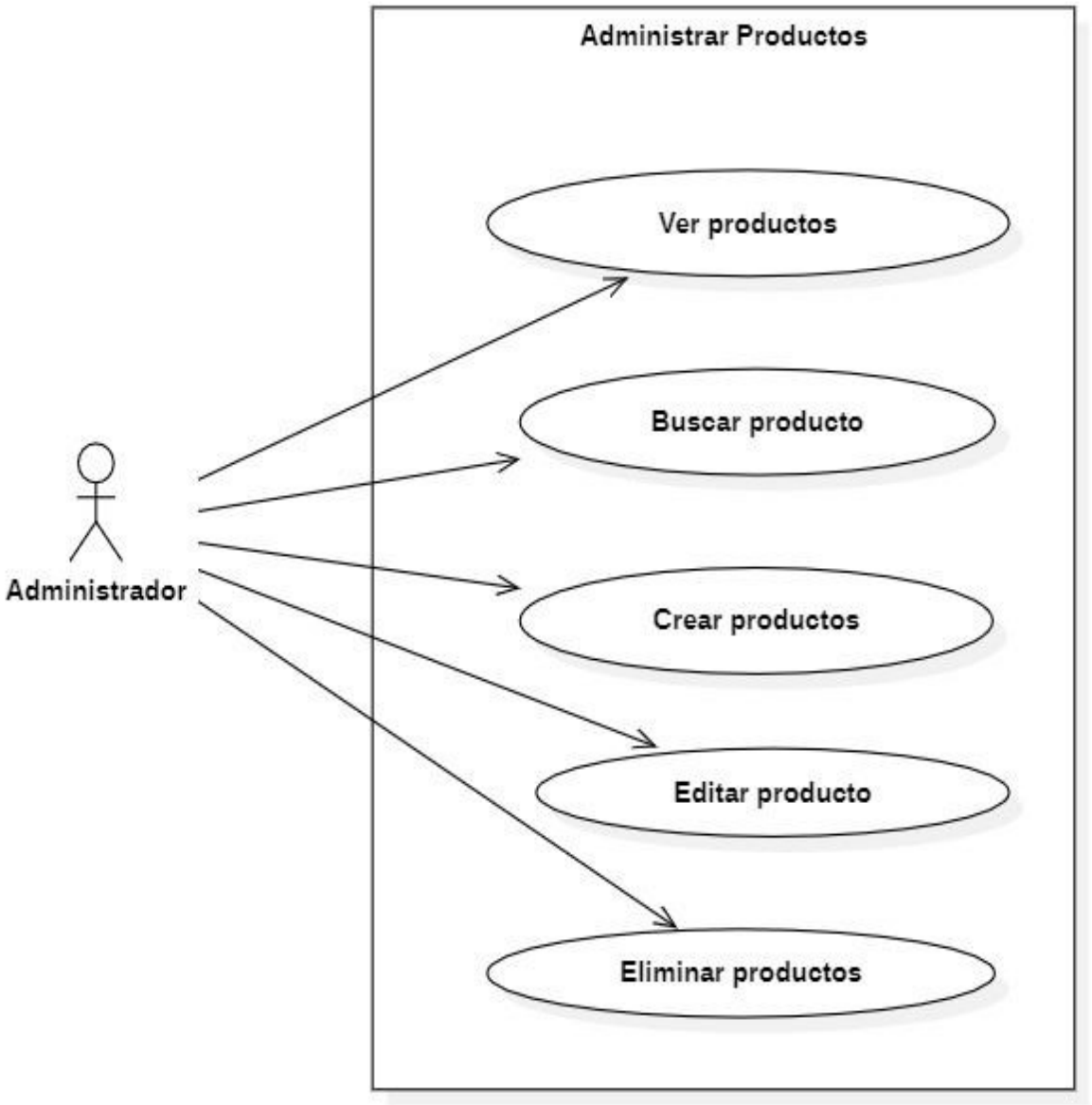
Diagrama de Caso de Uso: Administración de Almacenes.



Nota. Fuente: Actor

Figura 20

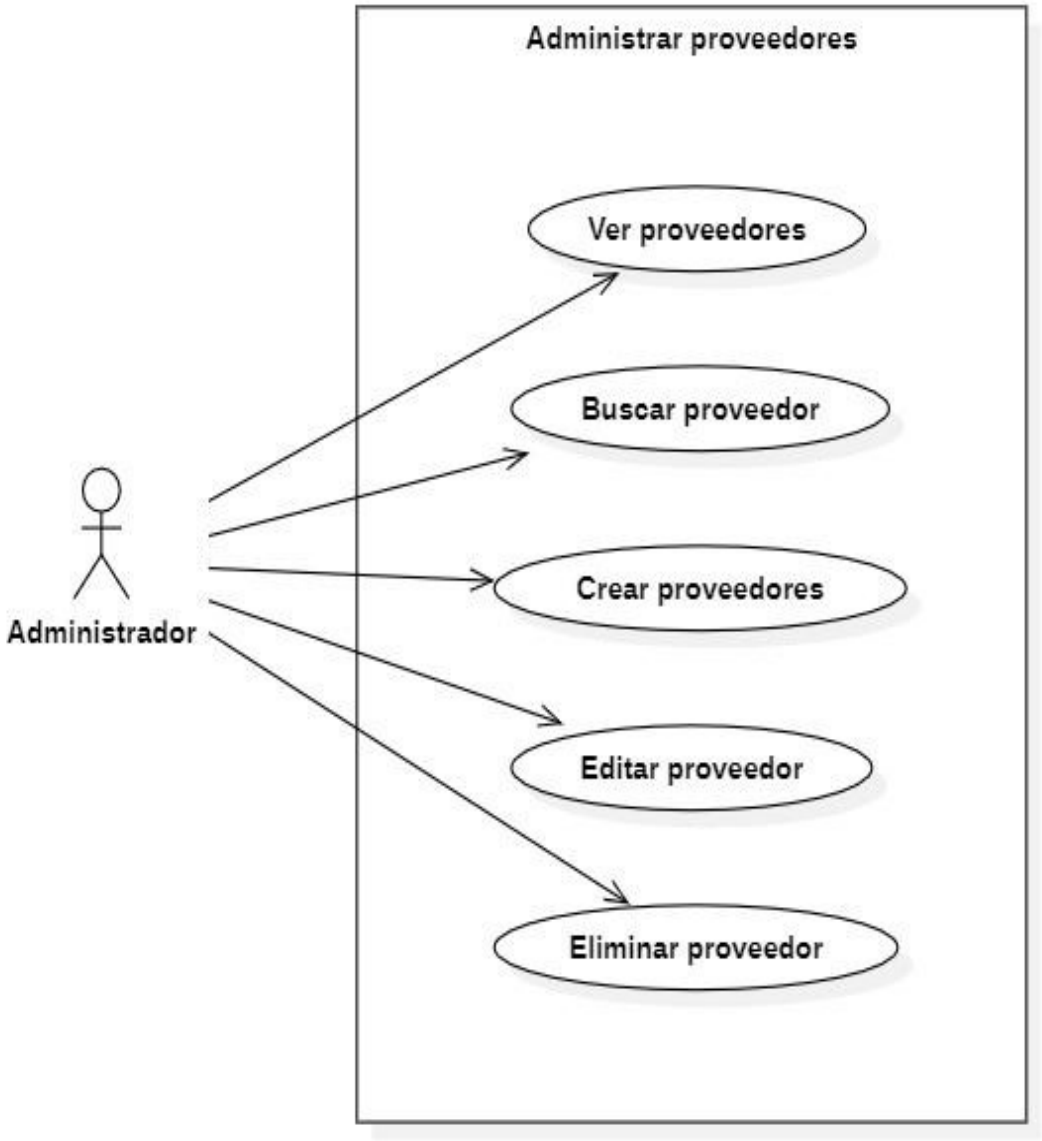
Diagrama de Caso de Uso: Administración de Productos.



Nota. Fuente: Actor

Figura 21

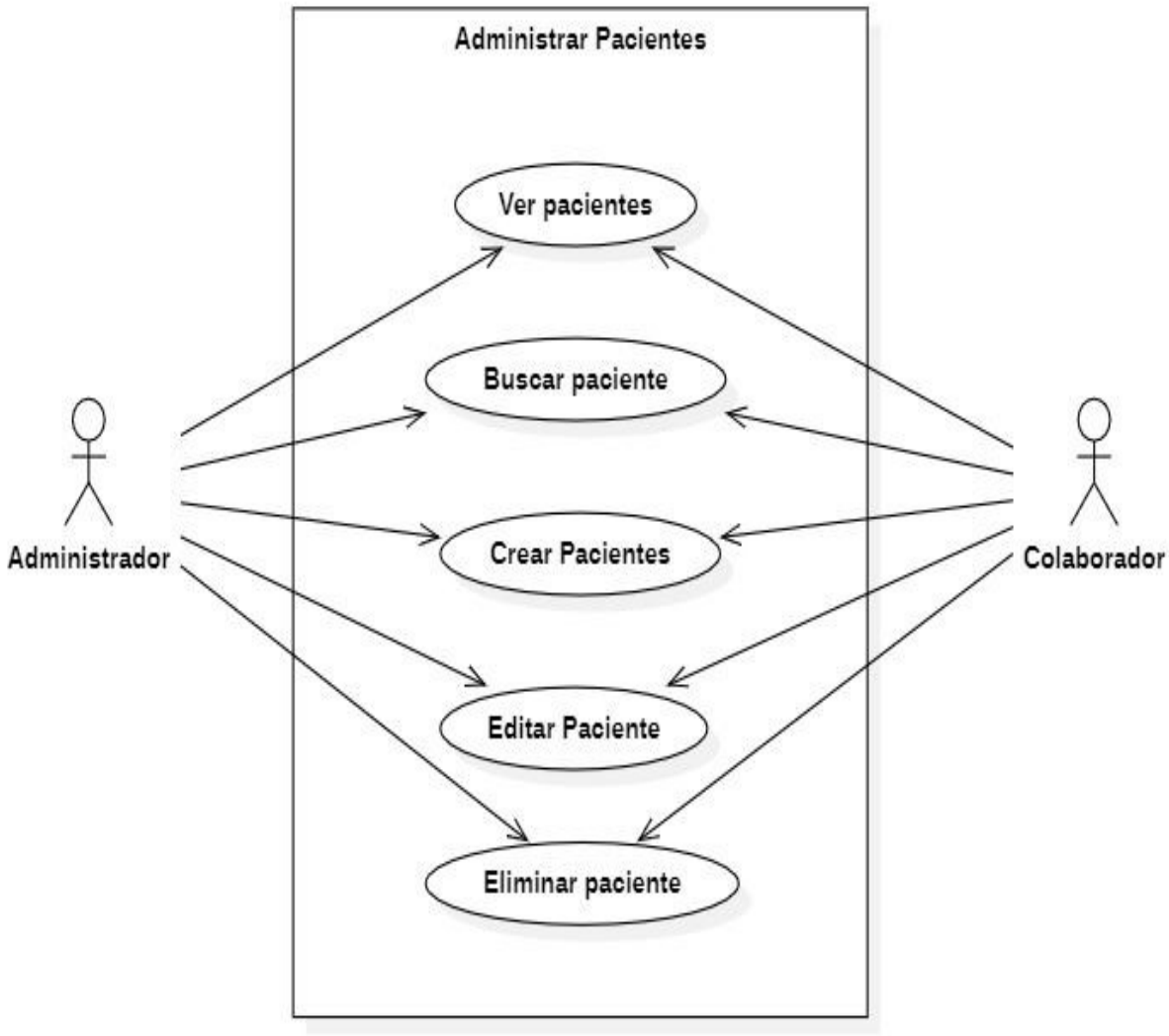
Diagrama de Caso de Uso: Administración de Proveedores.



Nota. Fuente: Actor

Figura 22

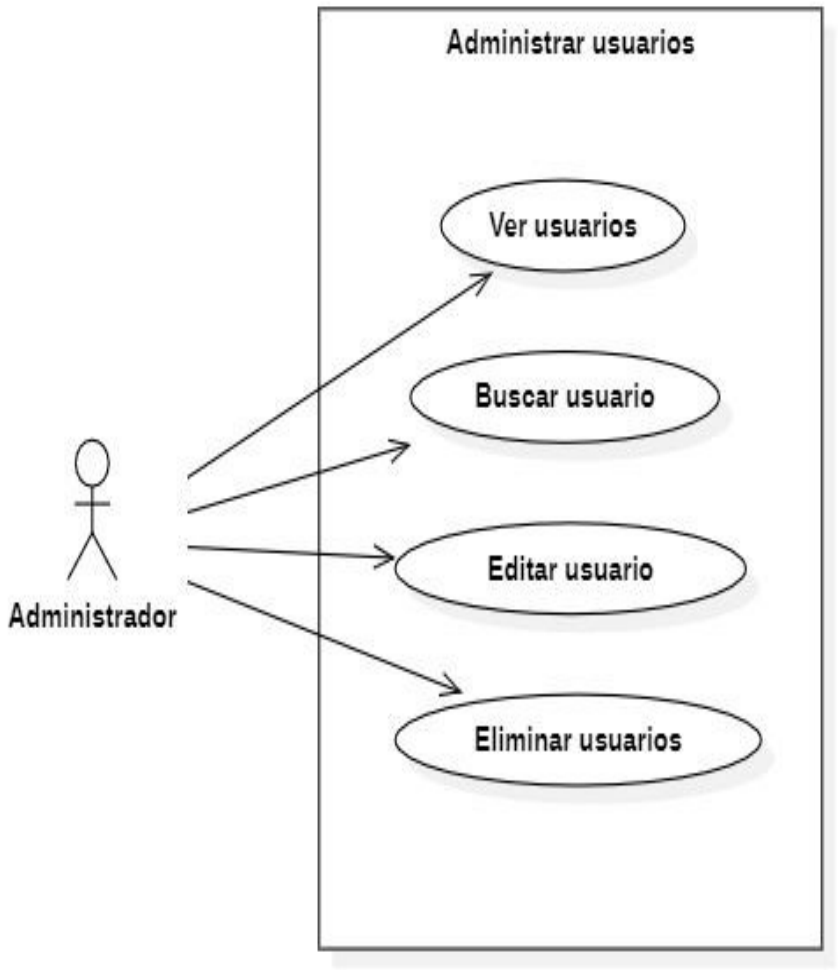
Diagrama de Caso de Uso: Administración de Pacientes.



Nota. Fuente: Actor

Figura 23

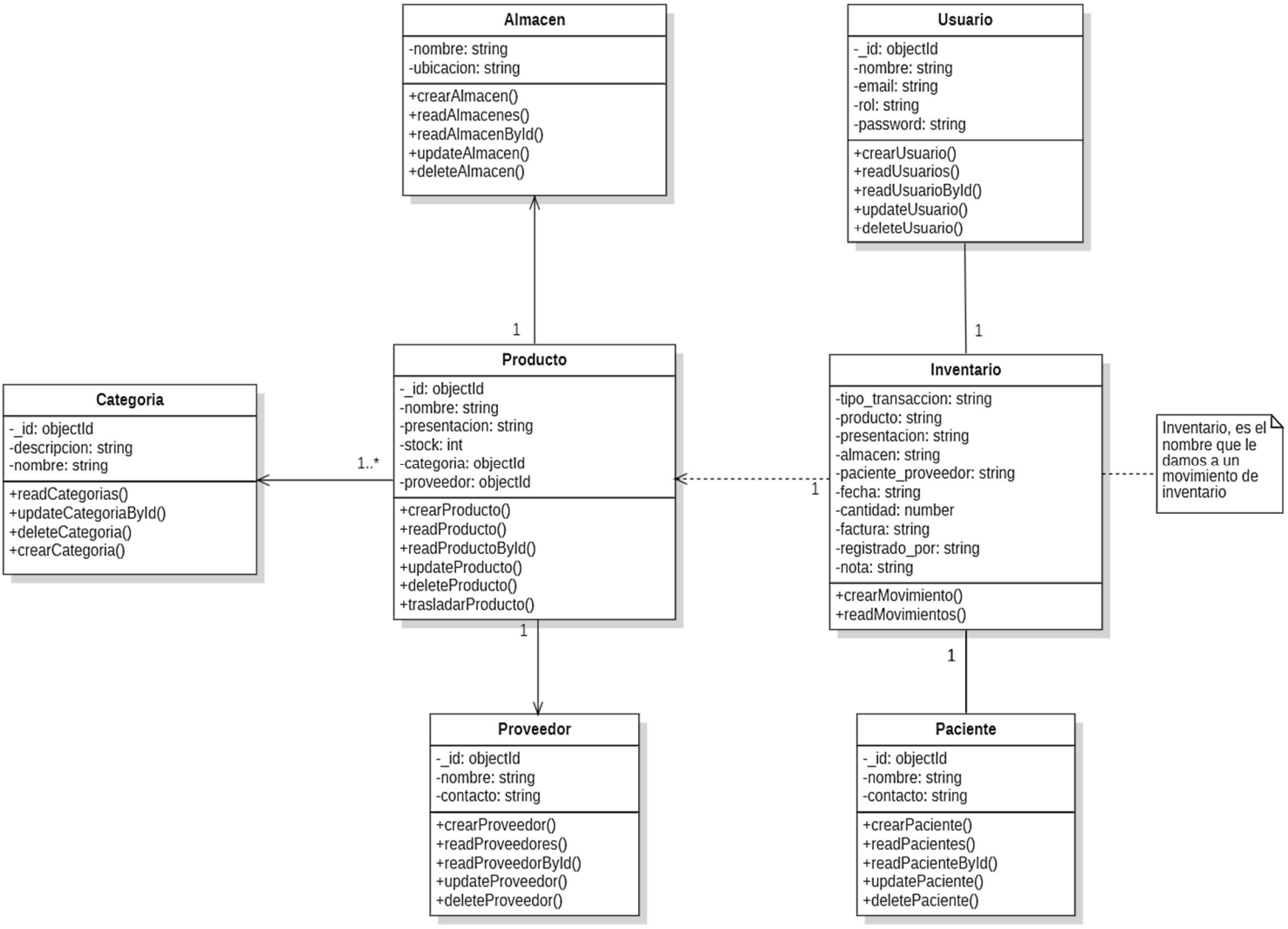
Diagrama de Caso de Uso: Administración de Usuarios.



Nota. Fuente: Actor

Figura 24

Diagrama de Clases



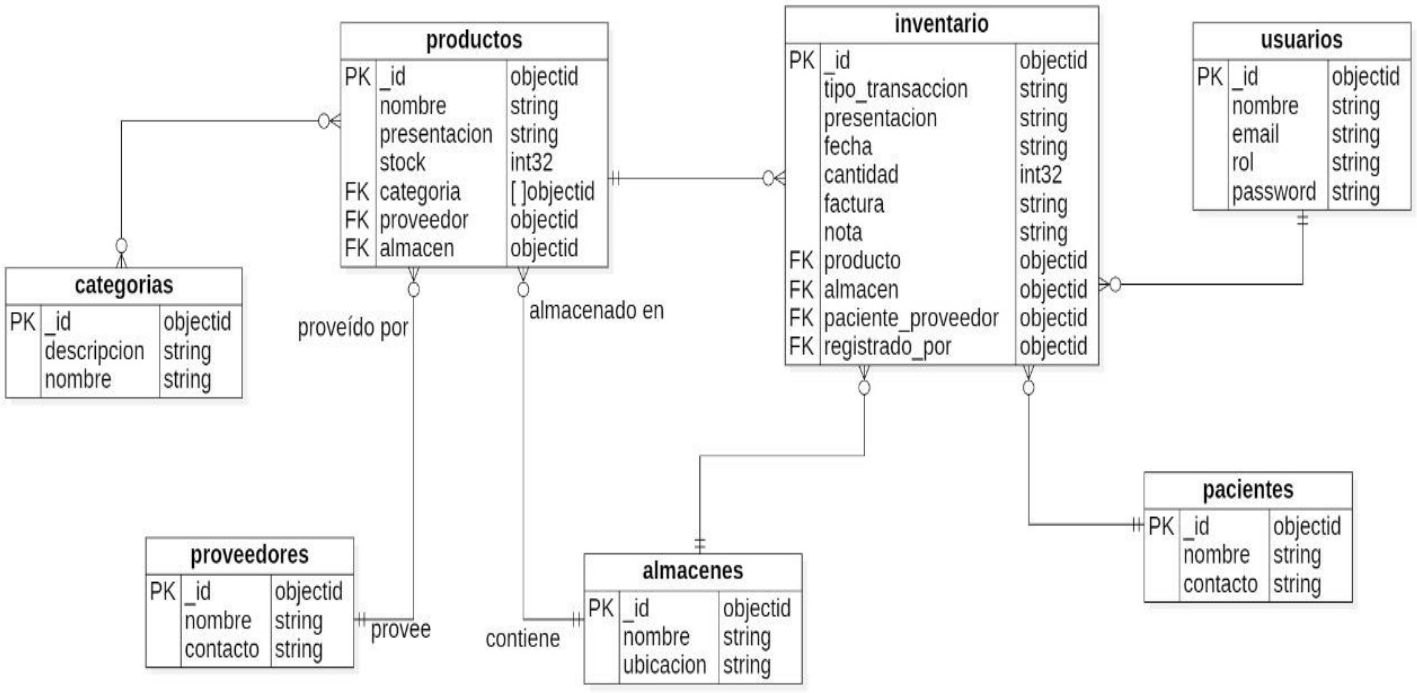
Nota. Fuente: Actor

9.2.2 Modelado de la Base Datos

Para lograr una gestión eficiente de la información y permitir una interacción fluida entre las distintas partes del sistema, se modeló una base de datos flexible y escalable. La Figura 25 muestra el diseño del esquema de la base de datos, que se compone de diversas colecciones para almacenar los datos de manera organizada y accesible.

Figura 25

Esquema de la base datos (Diagrama de colecciones)



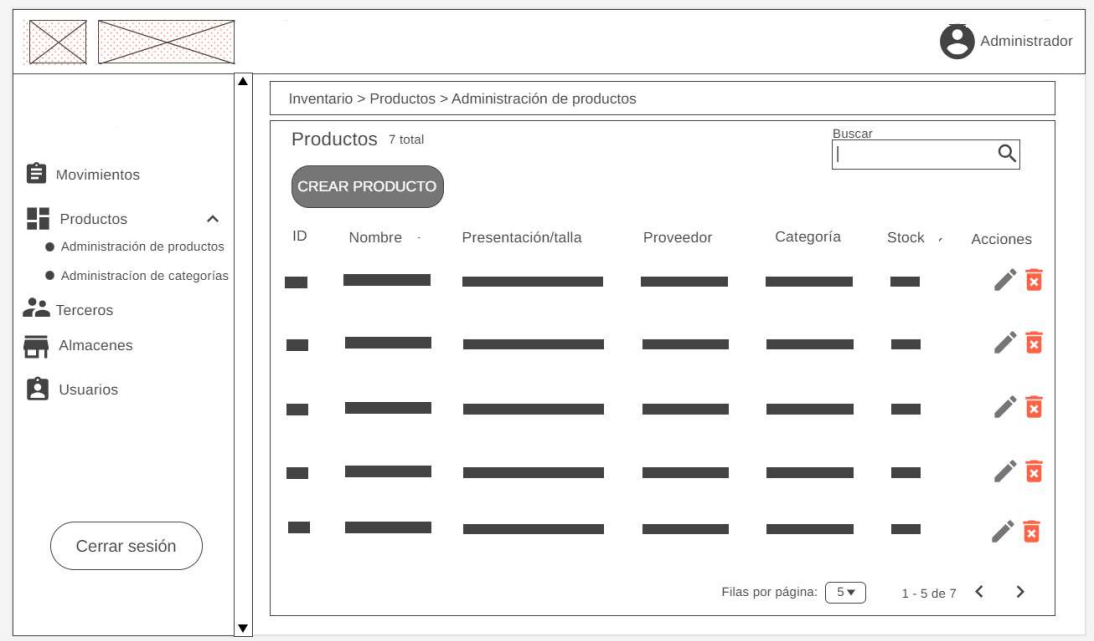
Nota. Fuente: Actor

9.2.3 Diseño de la Estructura de Interfaz de Usuario

Para el diseño de la interfaz de usuario, se optó por trabajar con wireframes interactivos en lugar de mockups estáticos, ya que el enfoqué fue mostrarle al Stakeholder la estructura y navegabilidad de la interfaz de usuario más que en realizar un sobre esfuerzo en la estética de la interfaz de usuario. Esta decisión se alineó con la naturaleza flexible de la metodología de desarrollo empleada, permitiéndonos hacer ajustes rápidos y eficientes durante los sprints. Cabe destacar que, aunque los wireframes no presentan colores, ya se tenía una guía de colores institucionales establecida por la empresa, que se compone principalmente de tonos de blanco y negro. Las Figuras 26 y 27 ilustran los wireframes interactivos correspondientes a las vistas de 'Administración de Productos' y 'Crear Productos', la herramienta usada fue Wireframe CC.

Figura 26

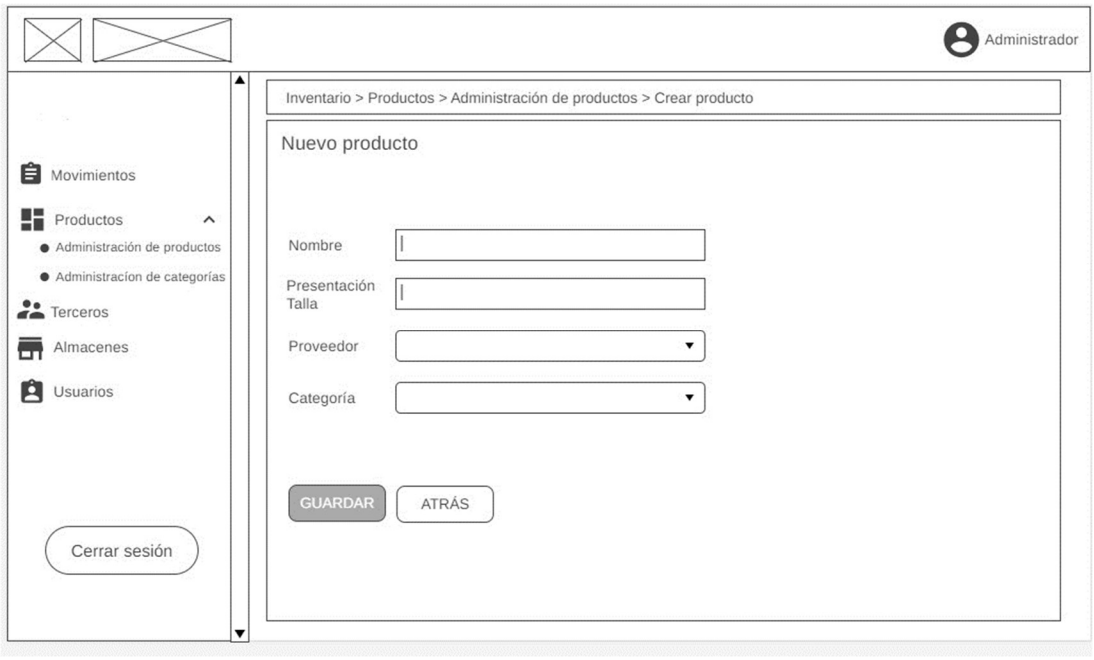
Wireframe de la Vista Administración de Productos



Nota. Fuente: Actor

Figura 27

Wireframe de la Vista Crear Productos.



Nota. Fuente: Actor

Los wireframes completos por cada vista, pueden ser consultados en el siguiente enlace: https://wireframe.cc/pro/pp/5e435e0a8643420 . Esto con el fin de no extender demasiado el documento.

9.3 Desarrollo del Prototipo.

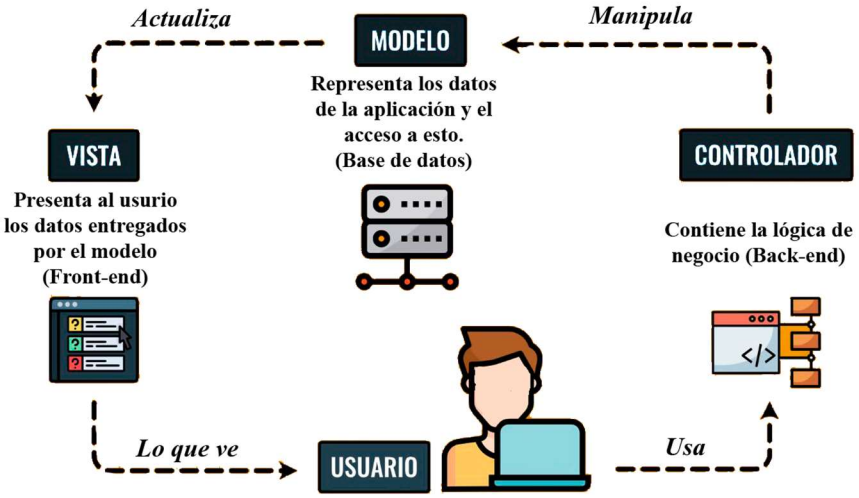
En esta fase se abordan los aspectos relacionados con la construcción efectiva del prototipo de aplicación web para la gestión de inventario, siguiendo las directrices y requerimientos definidos en las etapas anteriores del proyecto. La meta es llevar a la práctica las ideas conceptualizadas, transformándolas en un producto funcional que cumpla con los requerimientos establecidos.

9.3.1 Definición de la Arquitectura de Software para el Desarrollo

Previo al desarrollo de la base datos y la codificación, se realizó el análisis para definir la arquitectura de software, se tuvieron en cuenta tres factores para ellos, el primero fue la revisión bibliográfica, los requerimientos del usuario y los conocimientos previos del equipo de desarrollo. Se decidió emplear el patrón arquitectónico MVC (Modelo-Vista-Controlador), porque como se observa en la Figura 28, MVC permite desacoplar la lógica de negocio de la interfaz de usuario, lo que facilita tanto el mantenimiento como la escalabilidad del sistema.

Figura 28

Arquitectura del Modelo Vista Controlador



Nota. Fuente: Autor

Esta separación entre la lógica de negocio y la interfaz de usuario permitió dividir el desarrollo del prototipo en dos proyectos distintos: el Front End, encargado de la interfaz de usuario, y el Backend, responsable de la lógica de negocio. Este enfoque se considera una buena práctica y contribuyó a la eficiencia del desarrollo.

9.3.2 Definición de Tecnologías y Herramientas de Desarrollo

Dada la familiaridad del equipo de desarrollo con JavaScript, y teniendo en cuenta que se seleccionó el patrón arquitectónico Modelo-Vista-Controlador (MVC), la selección de tecnologías y herramientas se realizó de manera estratégica. Además, se buscó una solución que facilitara la escalabilidad del prototipo de aplicación web para adaptarse a necesidades futuras y que tambien ofreciera un rendimiento rápido en la usabilidad del usuario. Para el marco de trabajo se optó por el stack MERN (MongoDB, Express.js, React.js, Node.js) que, siendo completamente en JavaScript, también fomenta las mejores prácticas de desarrollo de software.

Para potenciar el proceso de desarrollo, se utilizaron las siguientes herramientas y tecnologías, cada una seleccionada por sus ventajas específicas:

- **Visual Studio Code:** Elegido de entorno de desarrollo integrado (IDE), por su eficiencia, extensibilidad y amplio soporte para tecnologías web. Ofrece una variedad de extensiones y una interfaz de usuario intuitiva que acelera el desarrollo.
- **GitHub**: Utilizado para el control de versiones, facilitando la colaboración en equipo y el mantenimiento de un historial detallado de cambios en el código. Es una herramienta robusta que ofrece diversas funcionalidades para manejar múltiples versiones del proyecto.
- **MongoDB Atlas y MongoDB Compass:** Se optó por esta combinación para gestionar nuestra base de datos, proporcionando una solución en la nube que es tanto escalable como fácil de administrar, y permitiendo que durante todo el

desarrollo cada programador tuviese acceso a la base datos sincronizadamente desde su computador. Compass sirve como la interfaz gráfica, facilitando la visualización y manipulación de datos.

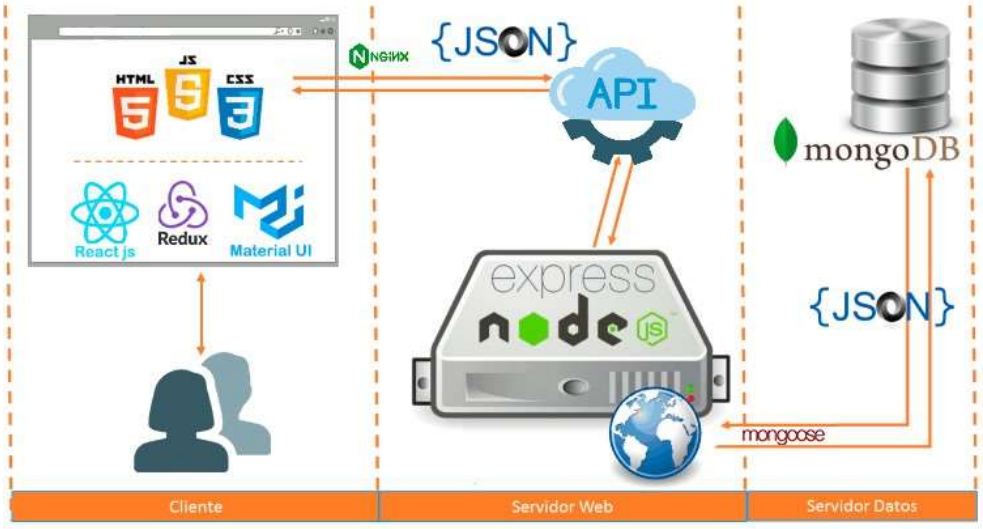
- **Material-UI**: Esta librería de diseño fue seleccionado para acelerar la estilización del frontend, ofreciendo una amplia gama de componentes preconstruidos y permitiendo que el equipo se centrara en la funcionalidad más que en los detalles estéticos.

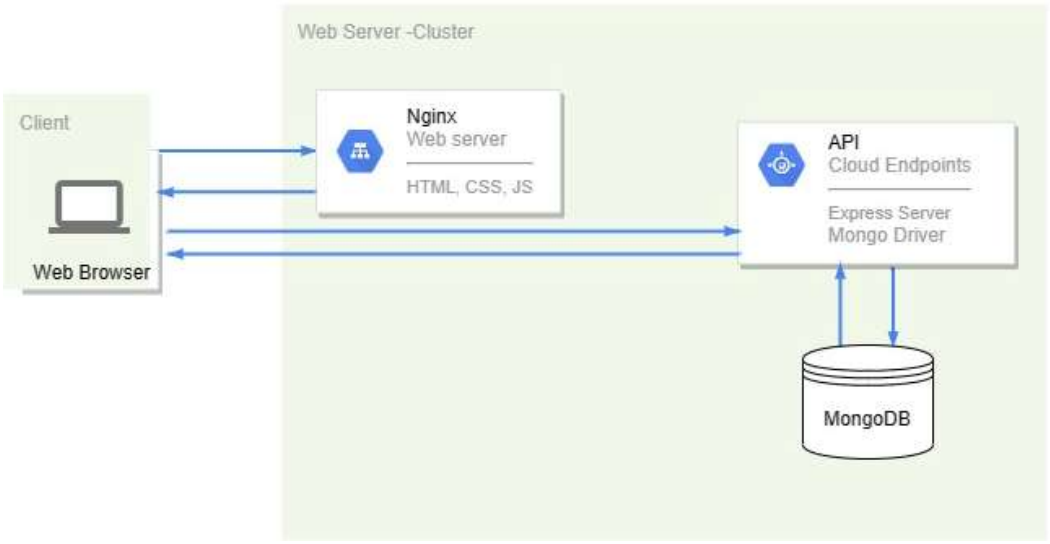
- **Redux:** Incorporado para mejorar la gestión del estado en la aplicación React, Redux hace más sencillo el manejo del estado global, lo que facilita el seguimiento de cambios y depuración.

- **Adobe Illustrator:** Utilizado para el diseño del logo, por sus capacidades avanzadas en la edición y creación de gráficos vectoriales.

Figura 29

Tecnologías y Arquitectura General de la Aplicación





Nota. Fuente: Autor

9.3.3 Creación de la Base Datos

Se establecieron los esquemas para diferentes colecciones de la base de datos, cada una correspondiente a una entidad específica de la aplicación web de inventario, esto en base al diseño de la base datos siguiendo el diagrama de colecciones, Figura 25. Las Tablas 6 a 12 presentan los detalles de estos esquemas, incluyendo los campos, tipos de datos, descripciones y si un campo es requerido o no, que tiene cada modelo.

Tabla 6

Esquema de Inventarios

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
tipo_transaccion	String	Tipo de transacción, movimiento de ingreso o egreso.	Si
producto	String	Nombre del producto involucrado en el movimiento de inventario.	Si
presentacion	String	Presentación o talla del producto involucrado en el movimiento de inventario.	Si
almacen	String	Nombre del almacen en donde está el producto.	Si
paciente_proveedor	String	Nombre de paciente o proveedor, dependiendo del tipo de movimiento.	Si
fecha	String	Fecha del movimiento de inventario.	Si
cantidad	Number	Cantidad del producto a mover.	Si

factura	String	Factura relacionada con el movimiento de inventario.	No
registrado_por	String	Nombre del usuario que realiza el movimiento.	Si
nota	String	Nota relacionada con el movimiento de inventario.	No

Nota. Se les llama inventarios a los movimientos de inventarios. Fuente: Autor

Tabla 7

Esquema de Productos

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
nombre	String	Nombre del producto involucrado en el movimiento de inventario.	Si
presentacion	String	Presentación o talla del producto involucrado en el movimiento de inventario.	Si
stock	Number	Nombre del almacen en donde está el producto	Si
categoria	ObjectId[]	Arreglo de objetos con referencia a la colección de categorías.	Si
proveedor	ObjectId	Objeto con referencia a la colección de proveedor.	Si
almacen	ObjectId	Objeto con referencia a la colección de almacén.	Si

Nota. Fuente: Autor

Tabla 8

Esquema de Proveedores.

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
nombre	String	Nombre del proveedor.	Si
contacto	String	Contacto del proveedor, puede ser correo electrónico o número telefónico.	Si

Nota. Fuente: Autor

Tabla 9

Esquema de Categorías

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
nombre	String	Nombre de la categoría de productos.	Si
descripcion	String	Descripción de la categoría de productos.	Si

Nota. Fuente: Autor

Tabla 10

Esquema de almacenes.

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
nombre	String	Nombre del almacén o bodega de productos.	Si
ubicacion	String	Ubicación del almacén o bodega de productos.	Si

Nota. Fuente: Autor

Tabla 11

Esquema de Pacientes

Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
nombre	String	Nombre del paciente, que es el cliente.	Si
contacto	String	Contacto del proveedor, puede ser correo electrónico o número telefónico.	Si

Nota. Fuente: Autor

Tabla 12

Esquema de Usuarios

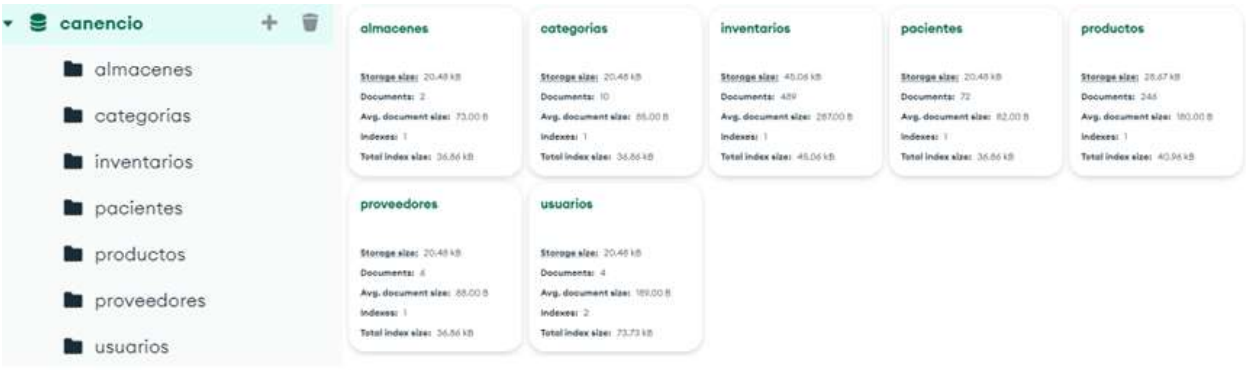
Campo	Tipo	Descripción	Requerido
_id	ObjectId	Identificador único de registro.	Si
name	String	Nombre del usuario.	Si
email	String	Correo electrónico del usuario.	Si
password	String	Contraseña del usuario, se encripta en la base datos.	Si
rol	String	Rol del usuario.	Si

Nota. Fuente: Autor

Posteriormente, se procedió a la creación de la base de datos utilizando Mongo Atlas, una base de datos como servicio que permite alojar bases de datos MongoDB en la nube. La Figura 30 ilustra la creación de la base de datos en esta plataforma.

Figura 30

Creación de la Base de Datos en Mongo Atlas.



Nota. Fuente: Autor

9.3.4 Backend con Express.js y MongoDB

9.3.4.1 Modelos y Esquemas

La estructuración de los datos se realizó a través de esquemas Mongoose, que definen cómo se almacenan los datos en MongoDB. A continuación, las Figuras 31 a 33 se presentan algunos de los esquemas de Mongoose para las colecciones como usuarios, productos, e inventarios.

Figura 31

Documento de MongoDB en Formato JSON de la Colección Productos.

Nota. Fuente: Autor

Figura 32

Documento de MongoDB en Formato JSON de la Colección Inventarios.

Nota. Fuente: Autor

Figura 33

Documento de MongoDB en Formato JSON de la Colección Usuarios.

Nota. Fuente: Autor

9.3.4.2 Controladores y Rutas

En el backend de la aplicación web de inventario, se ha empleado el framework Express.js, un marco de aplicación web altamente versátil para Node.js. Como se observa en la Figura 34, en Express.js las rutas (router) desempeñan un papel esencial, determinando cómo se deben procesar las solicitudes entrantes y qué controladores deben ser invocados en respuesta. Cada definición de ruta se asocia a una función específica, generalmente proporcionada por un controlador (controllers).

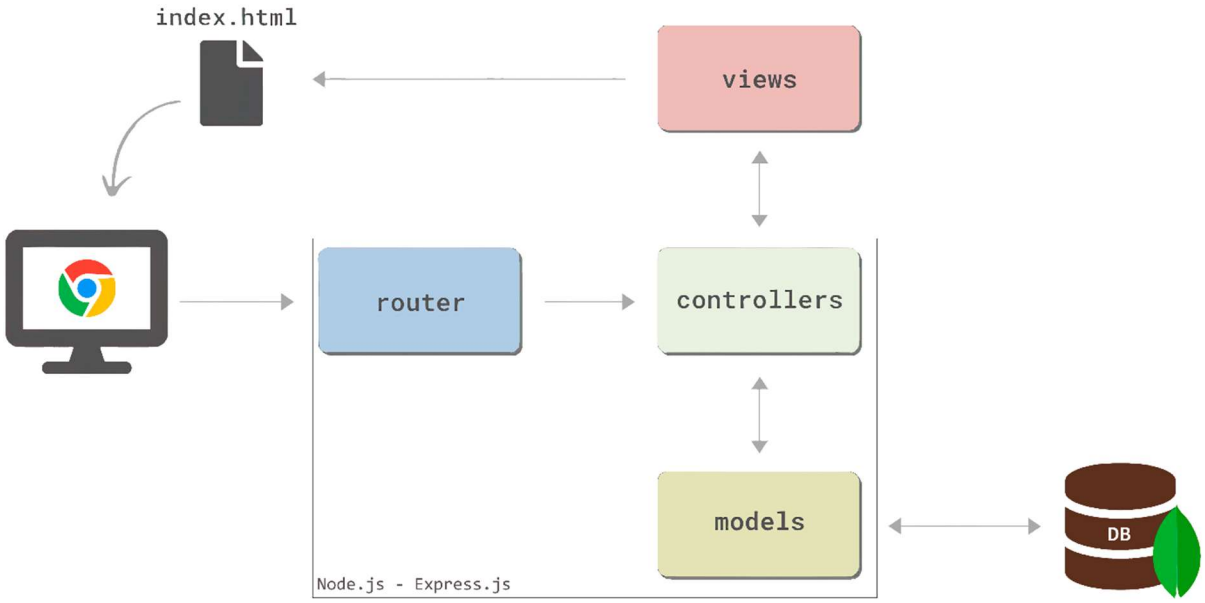
Los controladores, tales como authCtrl, almacenCtrl, productoCtrl, entre otros, contienen la lógica necesaria para manejar solicitudes específicas relacionadas con autenticación, gestión de almacén, productos y más. Tambien como se observa en la Figura 34, estas funciones de

controlador interactúan con la base de datos a través del modelo (model), realizan operaciones específicas de lógica de negocio y finalmente devuelven una respuesta al cliente.

Por otro lado, las rutas en la aplicación definen qué controlador debe ser invocado para una URL específica. Un enfoque común, que se ha adoptado en esta aplicación web de inventario, es segmentar las rutas en diferentes archivos o módulos, basados en su funcionalidad. Esto no solo ayuda a mantener el código organizado y limpio, sino que también permite una estructura modular, lo cual es fundamental para la escalabilidad y mantenimiento a largo plazo.

Figura 34

Flujo de la Aplicación Web con Express.js Basada en el Patrón MVC



Nota. Fuente: Autor.

9.3.4.3 Autenticación y Autorización

La autenticación se gestiona mediante tokens JWT. Cuando un usuario se registra o inicia sesión, se le proporciona un token que debe enviar con sus solicitudes subsiguientes para ser

autenticado. Esta técnica garantiza que solo los usuarios autenticados y autorizados puedan acceder a ciertas rutas y funcionalidades.

Autenticación: La autenticación verifica la identidad de un usuario. En otras palabras, asegura que el usuario es quien dice ser. La autenticación en esta aplicación web se realizó mediante el uso de JSON Web Tokens (JWT). Cuando un usuario se registra o inicia sesión en la aplicación, se verifica su información, por ejemplo, comparando un correo electrónico y una contraseña ingresada con una almacenada en la base de datos. Si la autenticación es exitosa, el servidor genera un JWT, que es un token codificado que contiene información sobre el usuario, cómo su _id. Este token se envía al cliente y se espera que el cliente lo almacene, por ejemplo, en el almacenamiento local del navegador. Para futuras solicitudes que requieran autenticación, el cliente debe enviar este token al servidor en los encabezados de la solicitud.

Verificación de JWT: Cuando el servidor recibe una solicitud con un JWT, lo decodifica y verifica su validez (por ejemplo, asegurándose de que no haya expirado y que haya sido firmado con la clave secreta adecuada). Si el token es válido, la solicitud se procesa. Si no es válido, se rechaza la solicitud y se informa al cliente.

Autorización: Mientras que la autenticación verifica la identidad de un usuario, la autorización determina qué se le permite hacer a ese usuario. Por ejemplo, un usuario con el rol de "administrador" puede tener más permisos que un usuario con un rol de "cliente".

Roles y Permisos: Los usuarios pueden tener dos roles, administrado o colaborador, y cada rol puede tener un conjunto específico de permisos. Al momento en que un usuario intenta llevar a cabo una acción, como por ejemplo eliminar un producto, el servidor no solo valida su autenticidad mediante el JWT, sino que también comprueba que el usuario cuente con el permiso

adecuado para esa acción específica. La Tabla 13 detalla los permisos asociados a las rutas según el rol del usuario. Estos permisos, a su vez, determinan el acceso a los controladores y, dentro de estos, a los métodos CRUD correspondientes a cada módulo.

Tabla 13

Accesos por Rol de Usuario

Rol	Rutas
Administrador	- Inventario. - Producto. - Categoría. - Almacen. - Proveedor. - Paciente. - Usuario.
Colaborador	- Inventario. - Paciente.

Nota. Fuente: Autor.

Middleware de Autorización: En el backend, se implementó middlewares específicos que se ejecutan antes de los controladores para verificar la autorización. Por ejemplo, si la ruta de productos solo debe ser accesible para usuarios con rol administrador, un middleware de autorización verifica el rol del usuario (a partir del JWT) antes de permitir el acceso. El middleware validarJWT se utiliza para validar JSON Web Tokens (JWT) en las solicitudes entrantes. Es una función intermedia que se ejecuta antes de que la solicitud llegue al controlador correspondiente.

Extracción del Token: El token se espera en un encabezado llamado x-token. Si no se proporciona un token, se devuelve una respuesta con un código de estado 401 (No Autorizado) y un mensaje indicando que no se proporcionó un token.

Verificación del Token: Cuando se encuentra un token, se utiliza jwt.verify para verificar su validez. Esto implica descifrarlo usando la clave secreta(process.env.SECRET_JWT_SEED).

Si el token es inválido (por ejemplo, si fue firmado con una clave diferente o si ha expirado), se devuelve una respuesta con un código de estado 401 y un mensaje que indica que el token es falso. Si el token es válido, se extraen el uid y el name del token verificado.

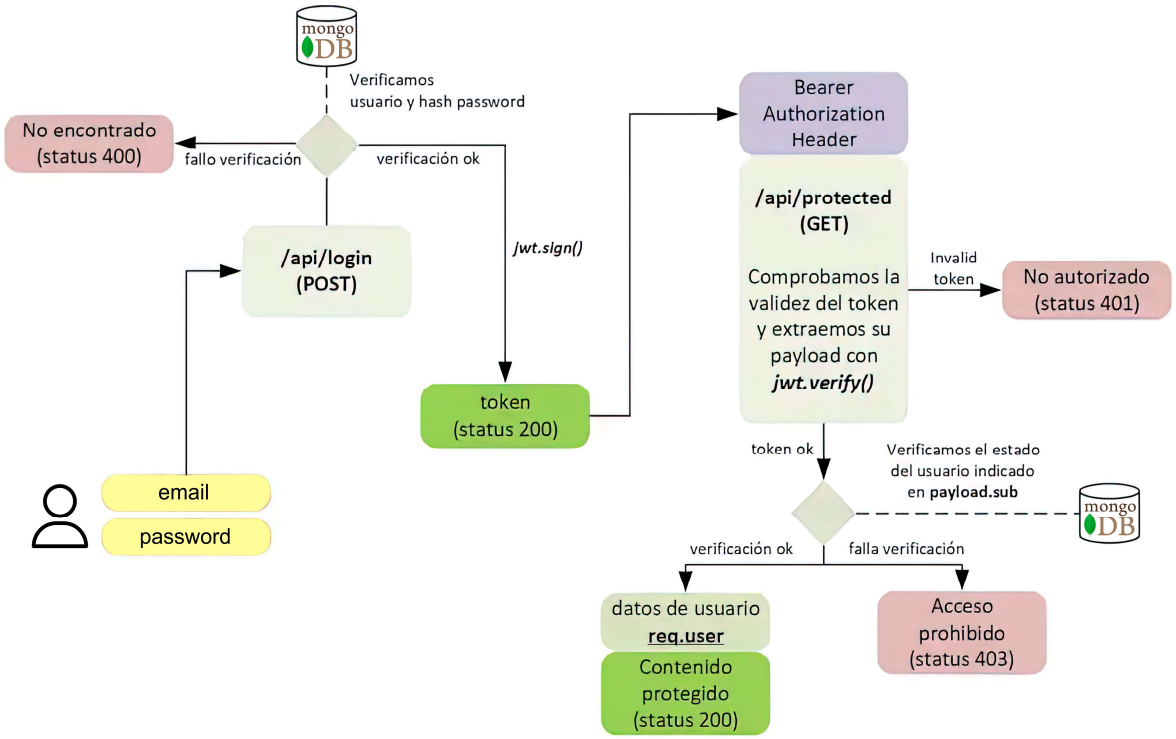
Logging: Se imprime en la consola información relevante para la solicitud, incluyendo el nombre del usuario, la IP desde la que se originó la solicitud, la URL y el método de la solicitud, la fecha y hora actuales. Esto fue útil para propósitos de seguimiento y pruebas.

Adjuntar Información al Objeto de Solicitud: Una vez que se ha verificado el token, se adjunta el uid y el name al objeto req, lo que permite que las funciones subsiguientes accedan a esta información. Por ejemplo, el controlador authCtrl.js puede usar req.uid para identificar al usuario que hizo la solicitud.

Avance al siguiente Middleware o Controlador: Si todo es exitoso y el token es válido, la función next() se invoca, permitiendo que la solicitud avance al siguiente middleware o al controlador correspondiente. Este middleware se puede agregar a cualquier ruta o conjunto de rutas que requieran autenticación. Al hacerlo, se garantiza que solo las solicitudes con un token JWT válido puedan acceder a recursos protegidos.

Figura 35

Diagrama de Flujo del Proceso de Autenticación con JWT



Nota. Fuente: Autor

9.3.5 Front End con React, Redux y Material UI

La arquitectura del front-end de la aplicación web de inventario fue construida sobre la integración de React, Redux y Material UI, conformando un ecosistema cohesivo y eficiente para la construcción de interfaces de usuario dinámicas y reactivas.

9.3.5.1 React

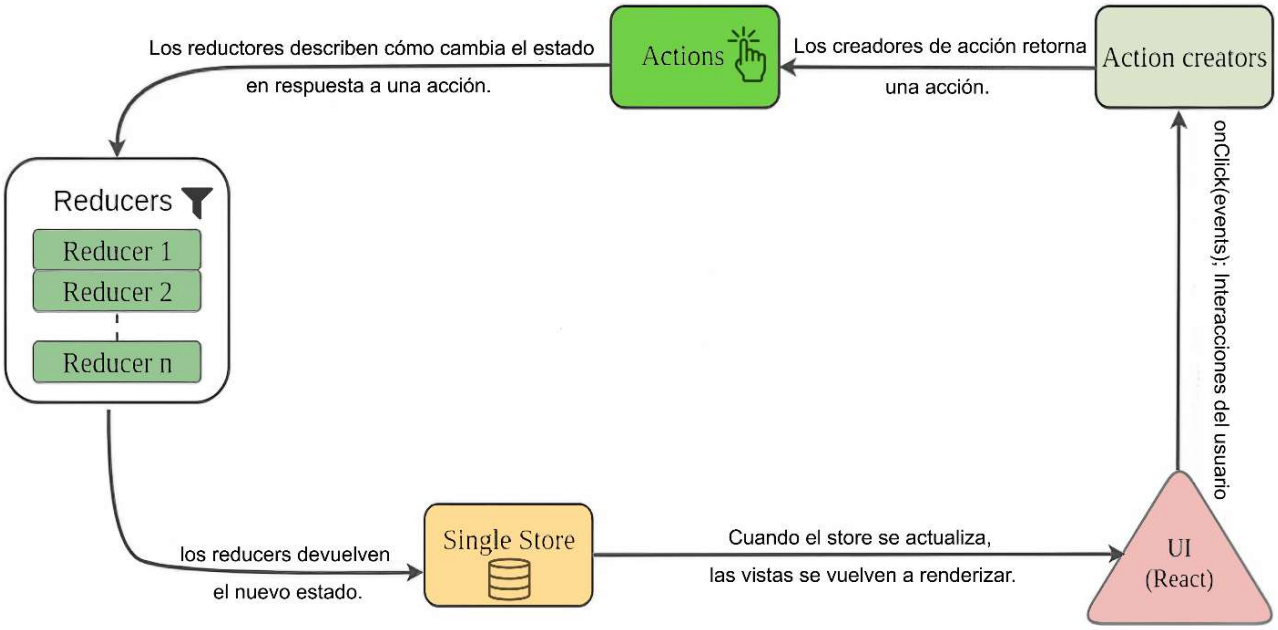
React se implementó como la librería fundamental para el desarrollo de la interfaz de usuario (UI) de la aplicación. Facilitó la creación de componentes reutilizables y gestionó de manera eficiente el renderizado y la actualización de la interfaz de usuario (UI) en respuesta a los cambios en el estado de la aplicación.

9.3.5.2 Redux

Redux fue implementado para gestionar el estado global de la aplicación. Como se muestra en la Figura 36, Redux utiliza un 'store' centralizado, garantizando así un flujo de datos predecible y uniforme a lo largo de toda la aplicación. Este modelo facilitó la manipulación del estado a través del uso de 'actions' y 'reducers'. En el flujo de trabajo de Redux, cuando se despacha una acción, el 'store' convoca a los 'reducers' para determinar el nuevo estado. A continuación, los componentes de React suscritos a dicho estado se actualizan automáticamente, manifestando los cambios en la interfaz de usuario y permitiendo el seguimiento de las modificaciones del estado a lo largo del tiempo.

Figura 36

Fujo de Trabajo de la Arquitectura Redux.



Nota. Fuente: Autor.

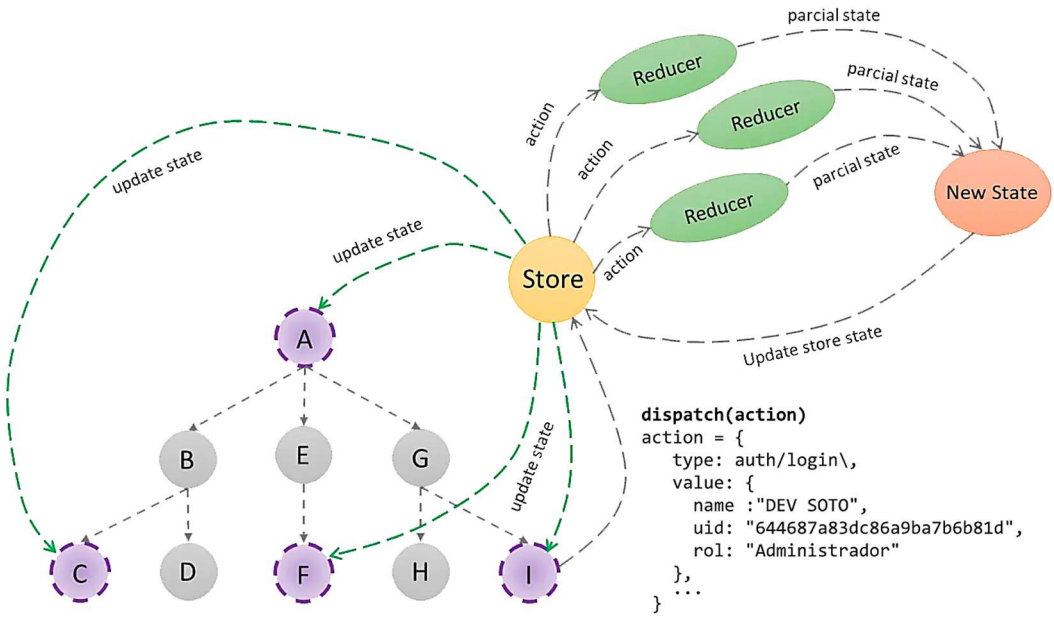
Los componentes que conforma Redux y como estos encajan para administrar el estado de la aplicación web inventario son:

- **Store**: En donde se ha representado el estado de la aplicación, Es el elemento central en la arquitectura redux puesto que contiene el estado (state) completo y actualizado de la aplicación web de inventario en un solo lugar.
- **Reducers**: Son funciones puras de JavaScript que se han definido para determinar cómo debe actualizarse el store en función de las acciones.
- **Actions**: Son los objetos JavaScript en los que se describen una intención para cambiar el estado del store.
- **Dispatch**: Es una función implementada para permitir el envío de acciones (actions) al store, con el objetivo de modificar el estado.

Como se observa en la Figura 37, Redux permite gestionar el estado desde una estructura externa a la jerarquía de componentes de React, lo que facilita la manipulación de datos comunes que son relevantes para una página determinada. Este proceso se destaca en la actualización de los componentes de React cuando ocurre un cambio en el estado de la aplicación, utilizando hooks para acceder y actualizar dicho estado de forma eficiente y concisa.

Figura 37

Actualizando los componentes con el nuevo estado



Nota. Fuente: Autor.

Cabe mencionar que la suscripción al store se realizó mediante el uso del hook useSelector. De este modo, cuando el store se actualiza, el hook se activa, ejecutando la función definida, y el nuevo estado es retornado y asignado a la variable correspondiente. Simultáneamente, el componente se actualiza para reflejar los cambios recientes."

9.3.5.3 Material UI

Material UI se implementó para establecer un diseño coherente y estéticamente agradable. Proporcionó una serie de componentes predefinidos y personalizables que adhieren a los principios de Material Design, facilitando así la creación de una interfaz de usuario intuitiva y atractiva. Componentes específicos como App bar, Accordion, Modal, Dialog, Grid V2, Date & Time Picker, Breadcrumbs, Data Grid, entre otros y junto con Iconos SVG de Material con tema Outlined, fueron integrados para optimizar la experiencia de usuario.

9.3.5.4 Integración y Flujo de Trabajo

- **Componentes y Estado**: Los componentes de React se conectaron al "store" de Redux, permitiendo el acceso y la modificación del estado global mediante el despacho de "actions". La suscripción a cambios en el "store" permitió la re-renderización de los componentes afectados.

- **Diseño y Estilo**: Material-UI suministró los estilos y componentes utilizados en la construcción de la UI. La integración de componentes de Material-UI dentro de los componentes de React aseguró una interfaz uniforme y agradable visualmente.

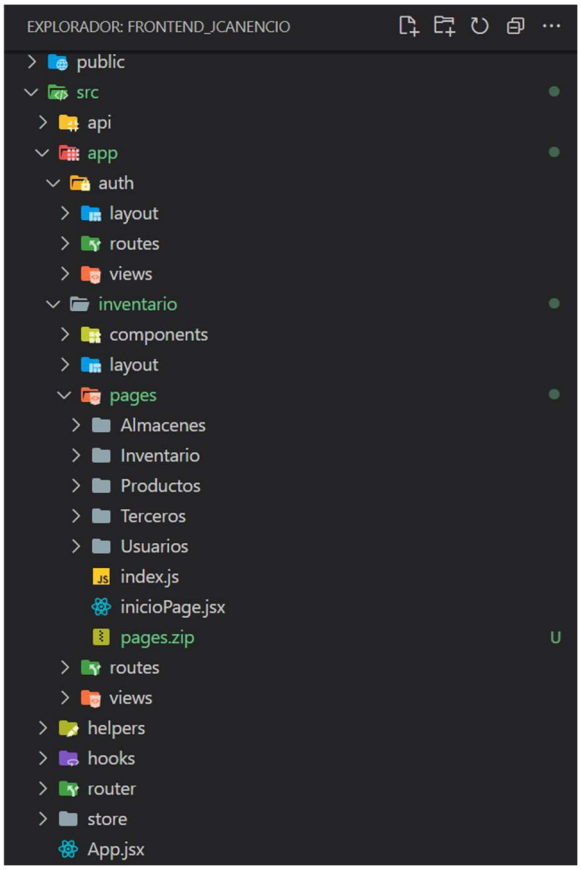
- **Interacción y Actualización**: Las interacciones del usuario con la UI, tales como clics de botón o entradas de formulario, desencadenaron el despacho de "actions" en Redux, provocando actualizaciones en el estado global y, consecuentemente, en la vista renderizada por React.

9.3.6 Resultados del Desarrollo Front End y Back End

El desarrollo del Front End y Back End se realizó en paralelo, utilizando GitHub para el manejo de versiones y colaboración entre los desarrolladores. Este enfoque permitió un seguimiento detallado de los cambios, facilitó la resolución de conflictos y aseguró la integridad del código a lo largo del desarrollo. A continuación, de la Figura 38 a la 57 se muestran los resultados del desarrollo:

Figura 38

Organización del Proyecto Front End



Nota. Esta figura muestra la estructura de directorios y archivos del proyecto Front End, evidenciando una organización modular y escalable, captura tomada de Visual Studio Code. Fuente: Autor

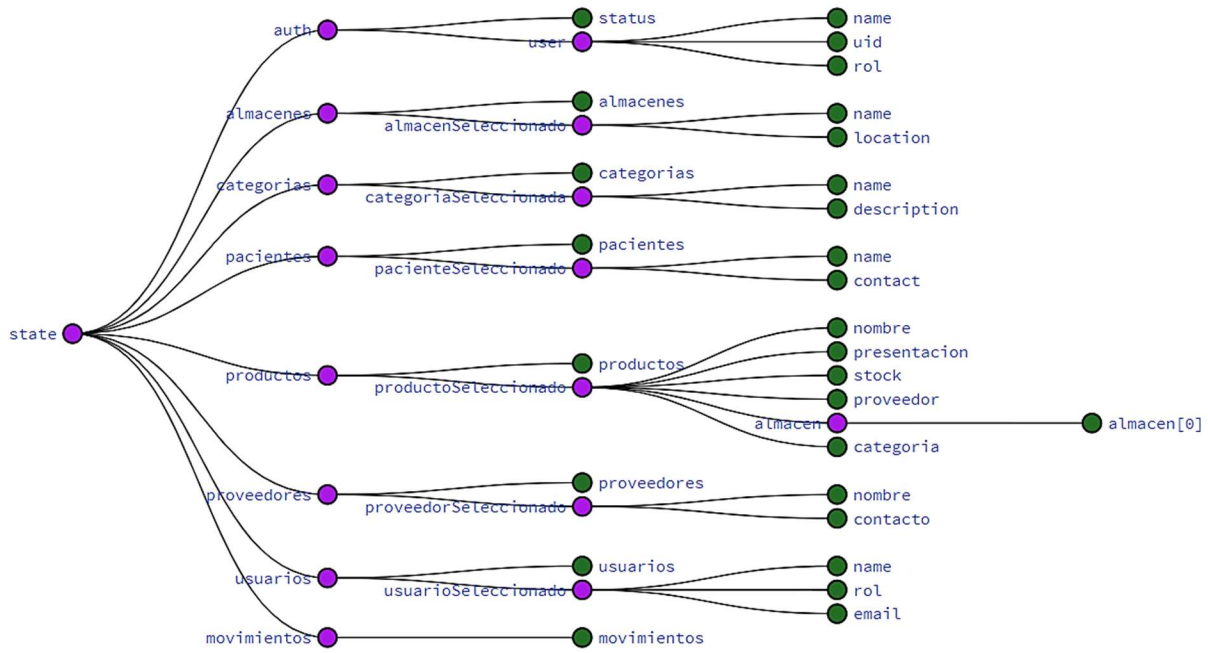
En la Figura 39 se observa el resultado de la representación visual del "State Tree" de la aplicación web de inventario, este árbol de estado es la estructura de datos central en Redux, donde se almacena y gestiona el estado global de la aplicación desde un único Store. En el nivel más alto del State Tree, se tiene varias ramas o nodos principales, que representan diferentes rebanadas (slices) o áreas de la aplicación web de inventario. A continuación, se detalla cada una:

- **auth**: Gestiona el estado relacionado con la autenticación de usuarios.

- **almacenes**: Almacena la información y estado de los almacenes en la aplicación de inventario.
- **categorias**: Maneja la información y estado de las diferentes categorías de productos.
- **pacientes**: Contiene el estado relacionado con los pacientes.
- **productos**: Administra el estado e información de los productos.
- **proveedores**: Maneja la información y estado de los proveedores.
- **usuarios**: Gestiona la información y estado de los usuarios.
- **movimientos**: Almacena el estado relacionado con los movimientos de inventario.

Figura 39

State Tree View de la Aplicación Web Inventario

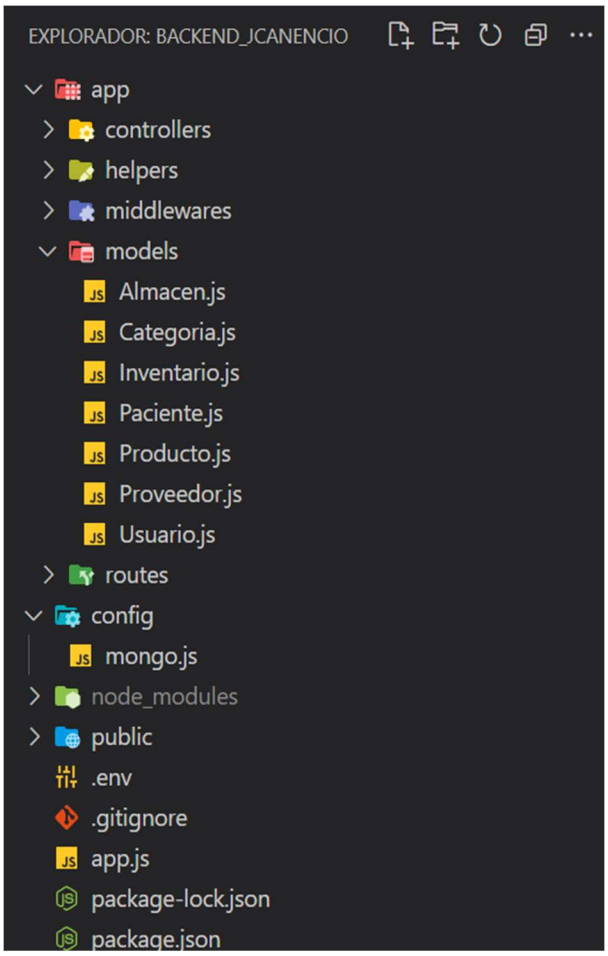


Nota. Generada por Redux DevTools. Fuente: Autor.

Cada uno de estos nodos principales se desglosa en subnodos que representan diferentes piezas de estado dentro de esa rebanada.

Figura 40

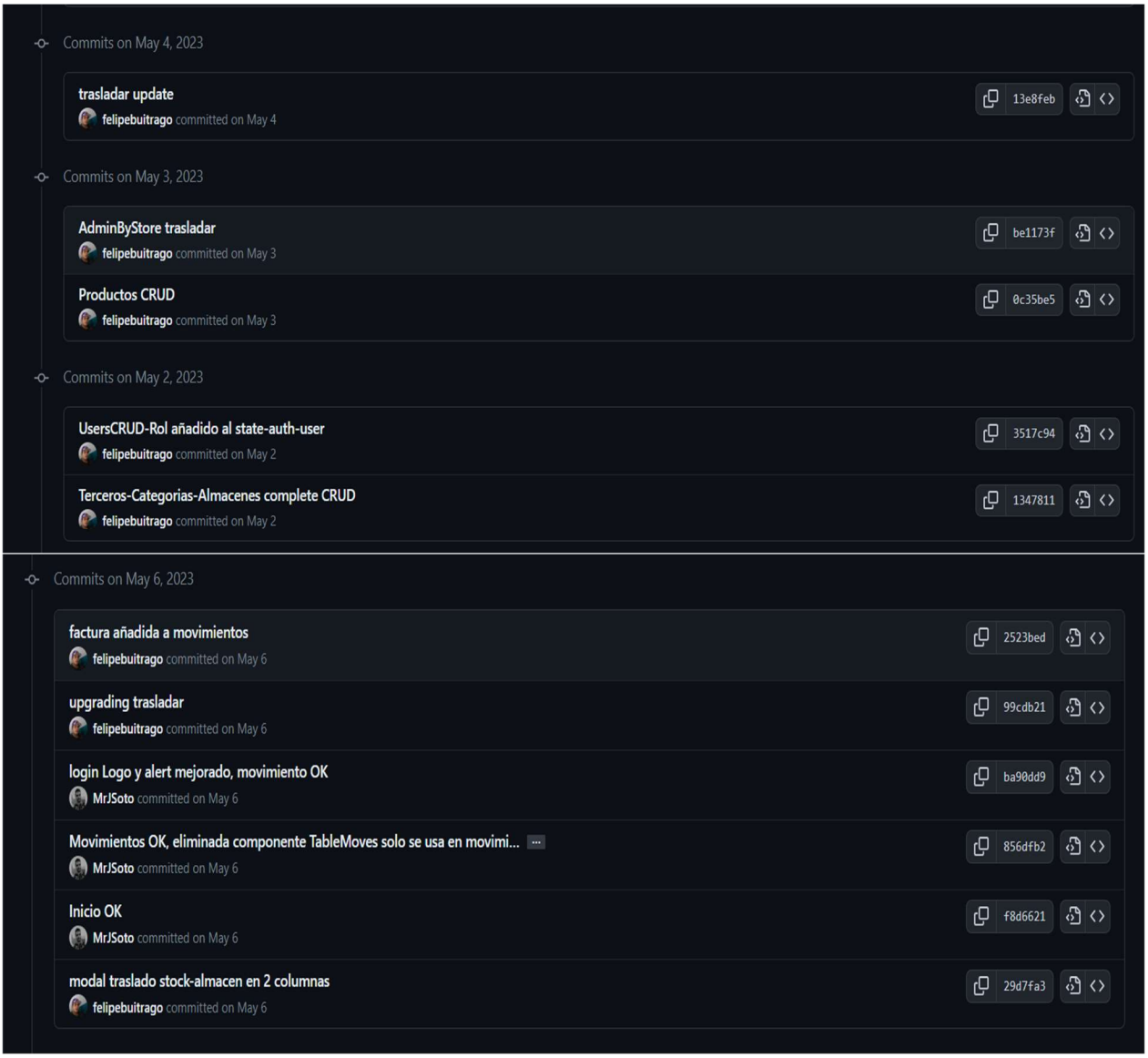
Organización del Proyecto Back End



Nota. Esta figura muestra la organización del proyecto Back End, destacando la separación de responsabilidades y la estructura de los archivos de la lógica de negocio, captura tomada de Visual Studio Code. Fuente: Autor.

Figura 41

Historial de Commits GitHub



Nota. Esta figura muestra el historial de commits en GitHub, reflejando el progreso del desarrollo, la colaboración entre los desarrolladores y la frecuencia de las actualizaciones en cada sprint. Fuente: Autor

El código fuente de la aplicación está disponible para su consulta a través de los siguientes enlaces, que corresponden a los repositorios del front end y del back end, respectivamente:

https://github.com/felipebuitrago/fend-canencio

https://github.com/felipebuitrago/backend_canencio

La aplicación web de inventario fue diseñada y desarrollada con un enfoque en la simplicidad y la facilidad de uso, pues en si es un prototipo bien pulido. Desde el inicio de sesión hasta la interacción con diferentes módulos, cada elemento ha sido cuidadosamente pensado para ofrecer una experiencia de usuario intuitiva.

Una vez que los usuarios acceden a la aplicación web, son recibidos por una pantalla de inicio que les brinda una visión general de la aplicación y las opciones disponibles. Esta interfaz sirve como punto de partida para explorar las diversas funcionalidades de la aplicación dependiendo del rol del usuario, en el caso del administrador se incluyó el seguimiento detallado de los movimientos del inventario, la administración de productos, terceros, almacenes y usuarios.

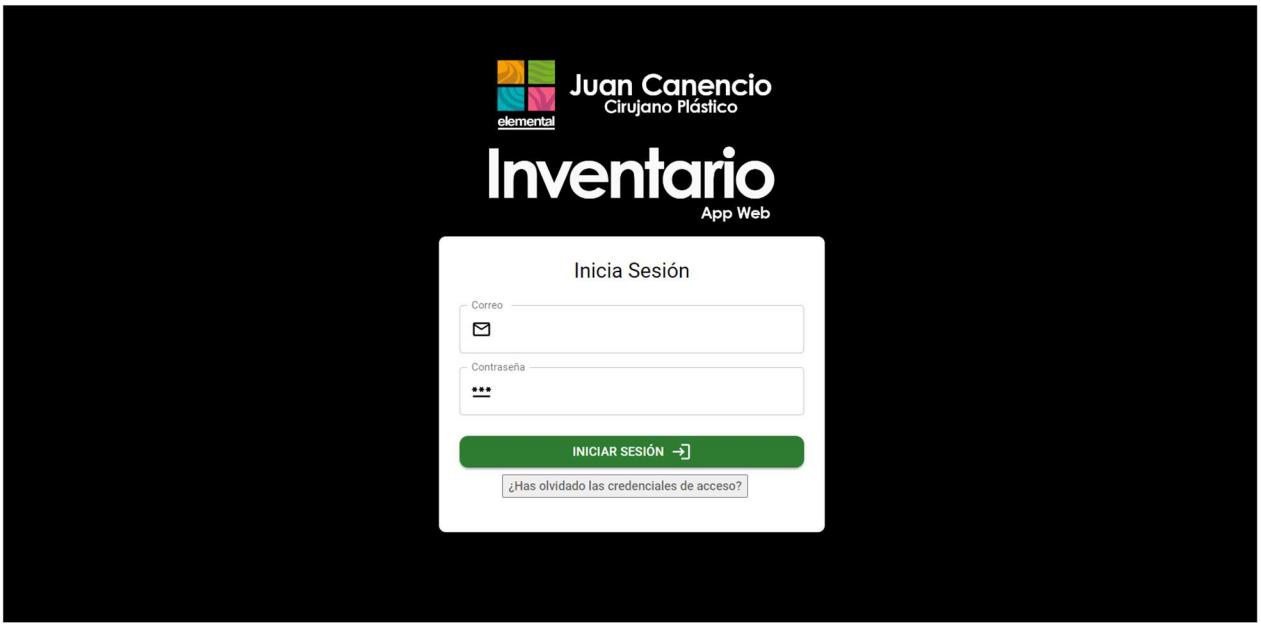
Los formularios y diálogos modales se han implementado para facilitar la realización de movimientos en el inventario, tanto de ingreso como de egreso. Estos elementos demuestran la consideración por la experiencia del usuario y el deseo de crear interacciones fluidas y sin complicaciones.

Las vistas de administración permiten a los usuarios gestionar diferentes aspectos la aplicación web de inventario, desde productos hasta almacenes, resaltando la modularidad y organización de la aplicación. Además, las notificaciones integradas en la aplicación web aseguran que los usuarios estén informados.

Lo anterior, como resultado de la integración exitosa de tecnologías y prácticas de desarrollo modernas se refleja en la cohesión y funcionalidad de cada componente y vista, tal como se ilustra desde la Figura 42 hasta la 57.

Figura 42

Vista del Log In



Nota. Fuente: Autor

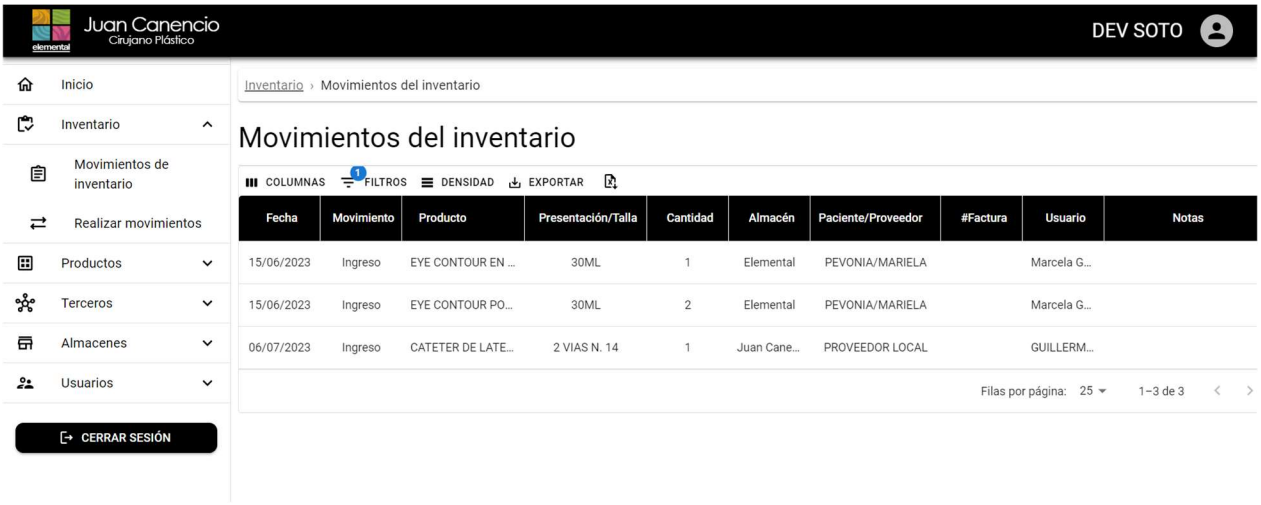
Figura 43

Vista del Inicio de la Aplicación.

Nota. Fuente: Autor

Figura 44

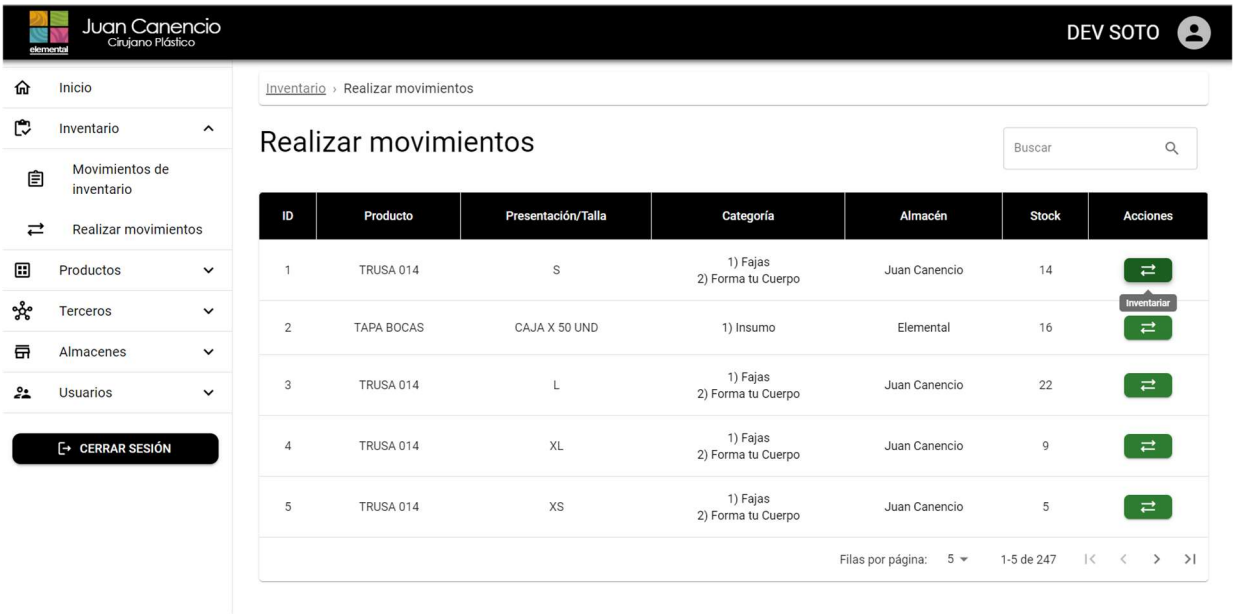
Vista del Historial de Movimientos del Inventario.



Nota. Fuente: Autor

Figura 45

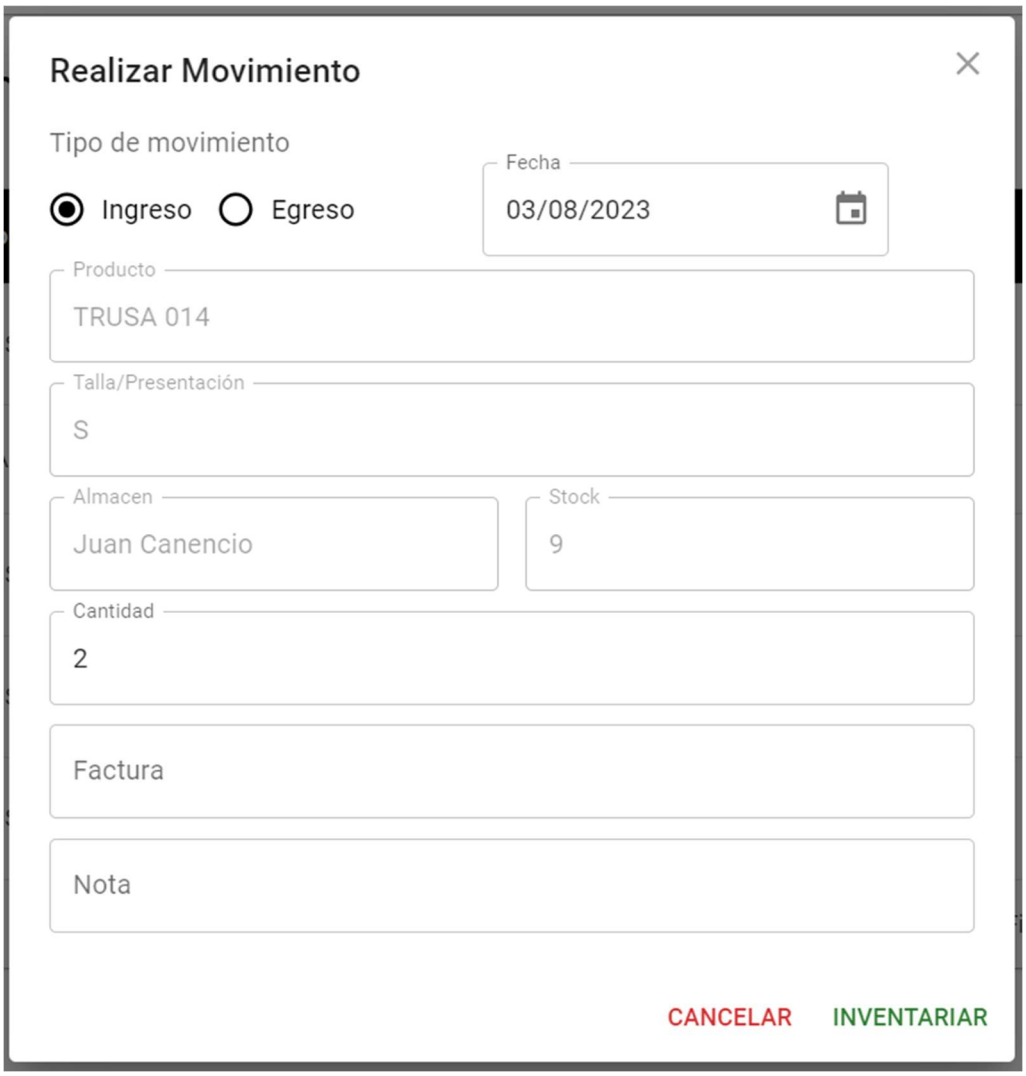
Vista de Realización de Movimiento en Inventario



Nota. Fuente: Autor

Figura 46

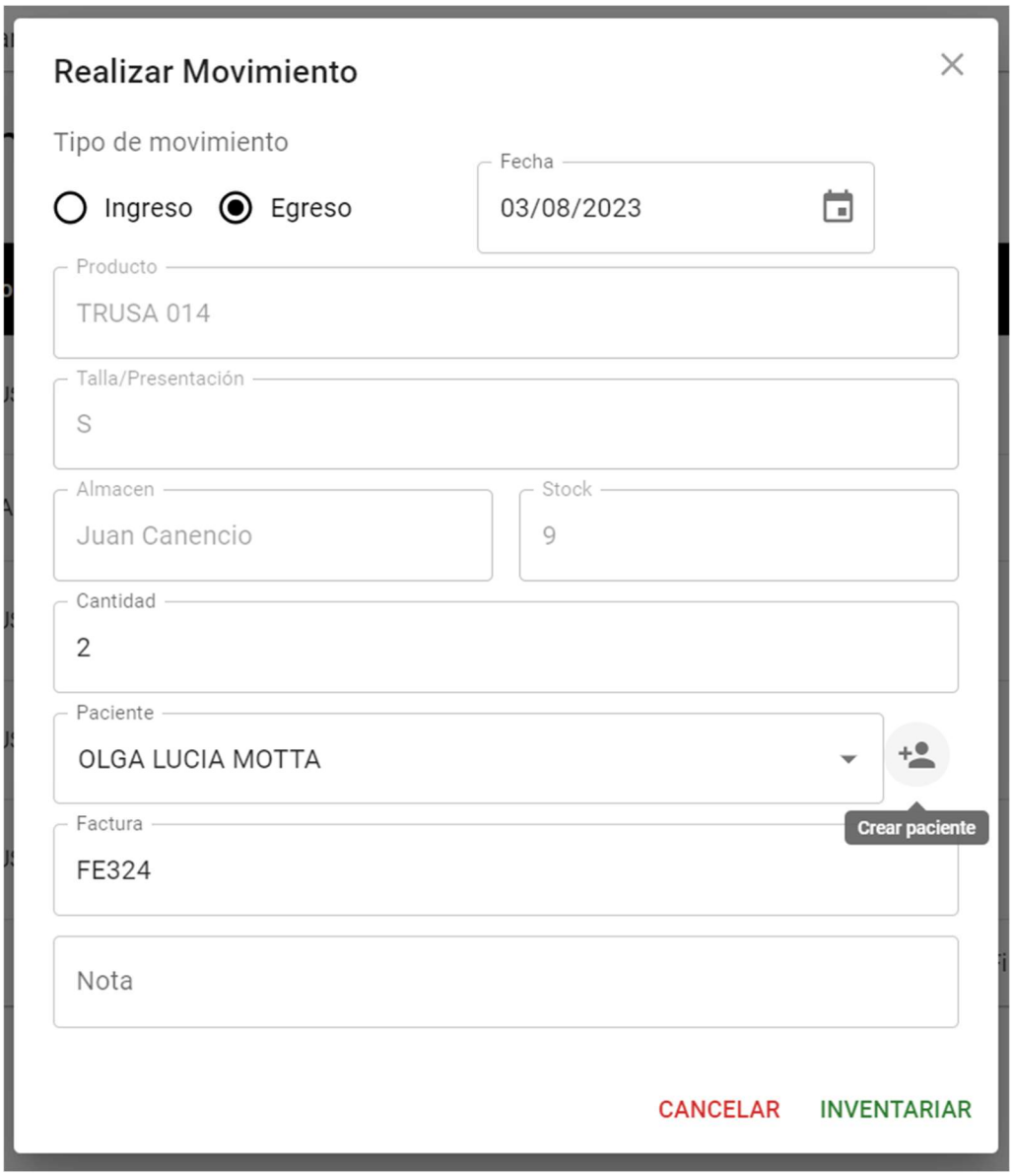
Dialog Modal del Formulario para Realizar Movimientos, en caso de Ingreso.



Nota. Fuente: Autor

Figura 47

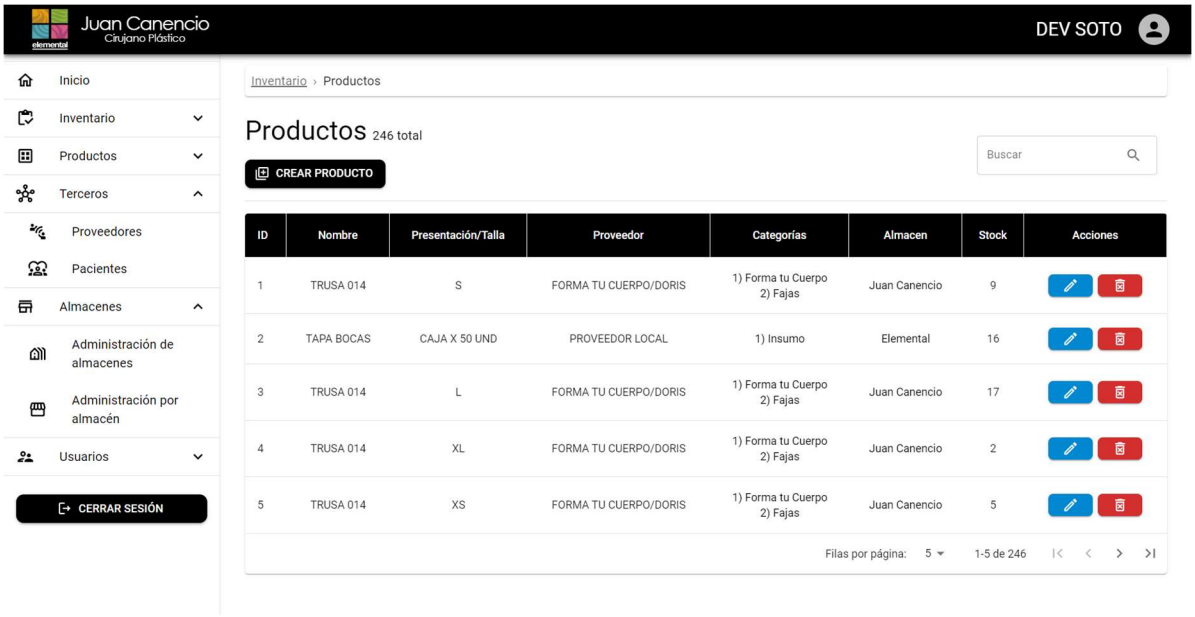
Dialog Modal del Formulario para Realizar Movimientos, en caso de Egreso.



Nota. Fuente: Autor

Figura 48

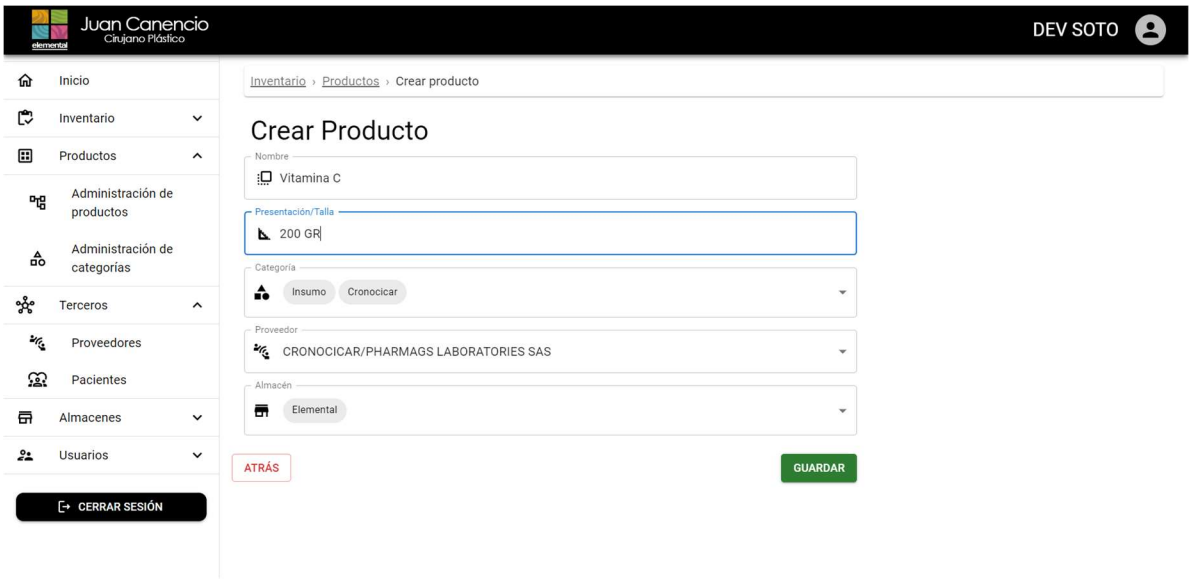
Vista de Administración de Productos.



Nota. Fuente: Autor

Figura 49

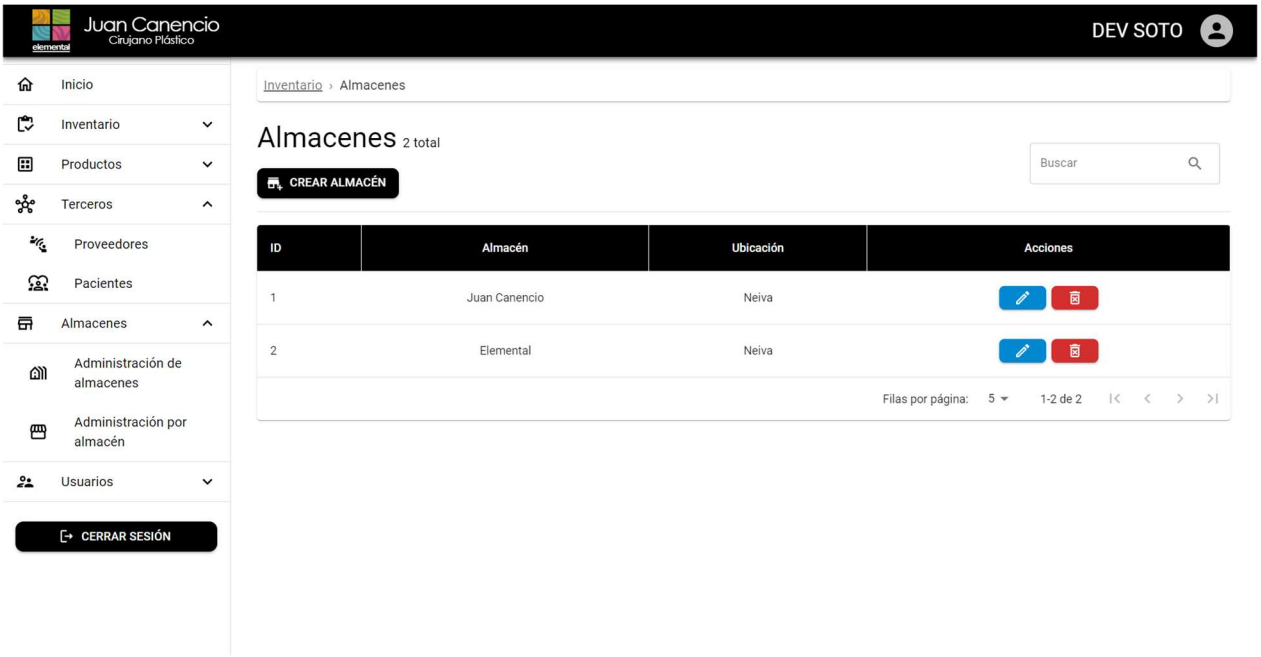
Vista del Formulario para Crear un Producto



Nota. Fuente: Autor

Figura 50

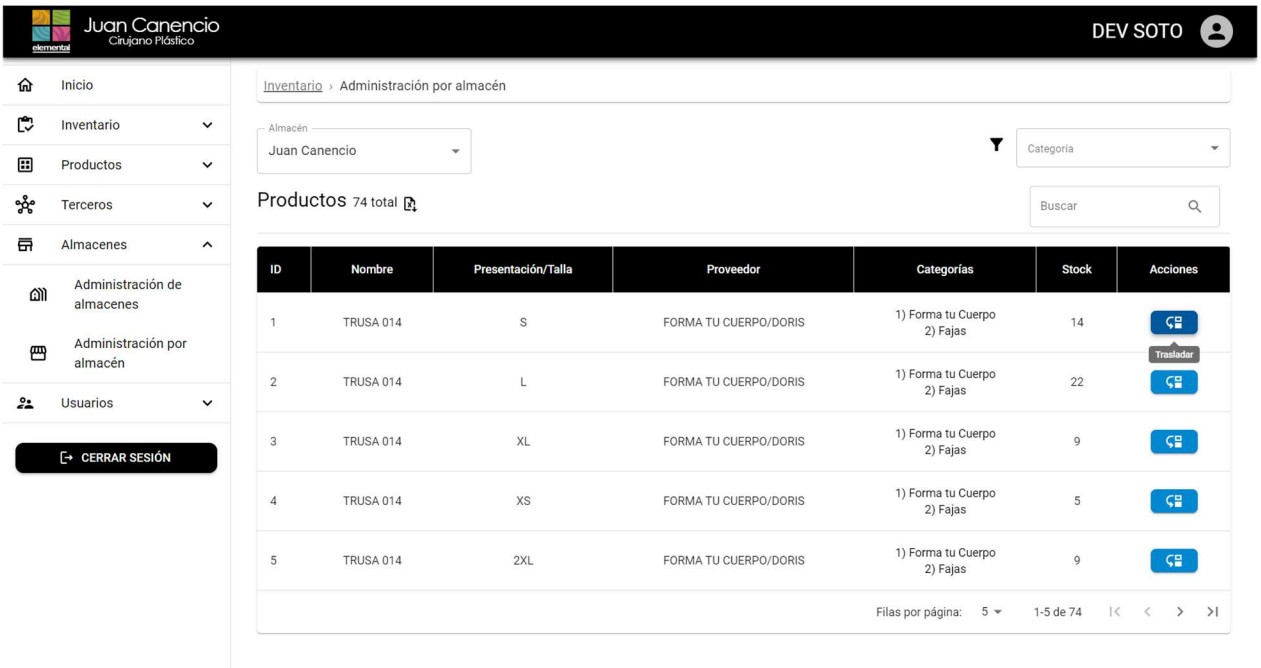
Vista de Administración de Almacenes.



Nota. Fuente: Autor

Figura 51

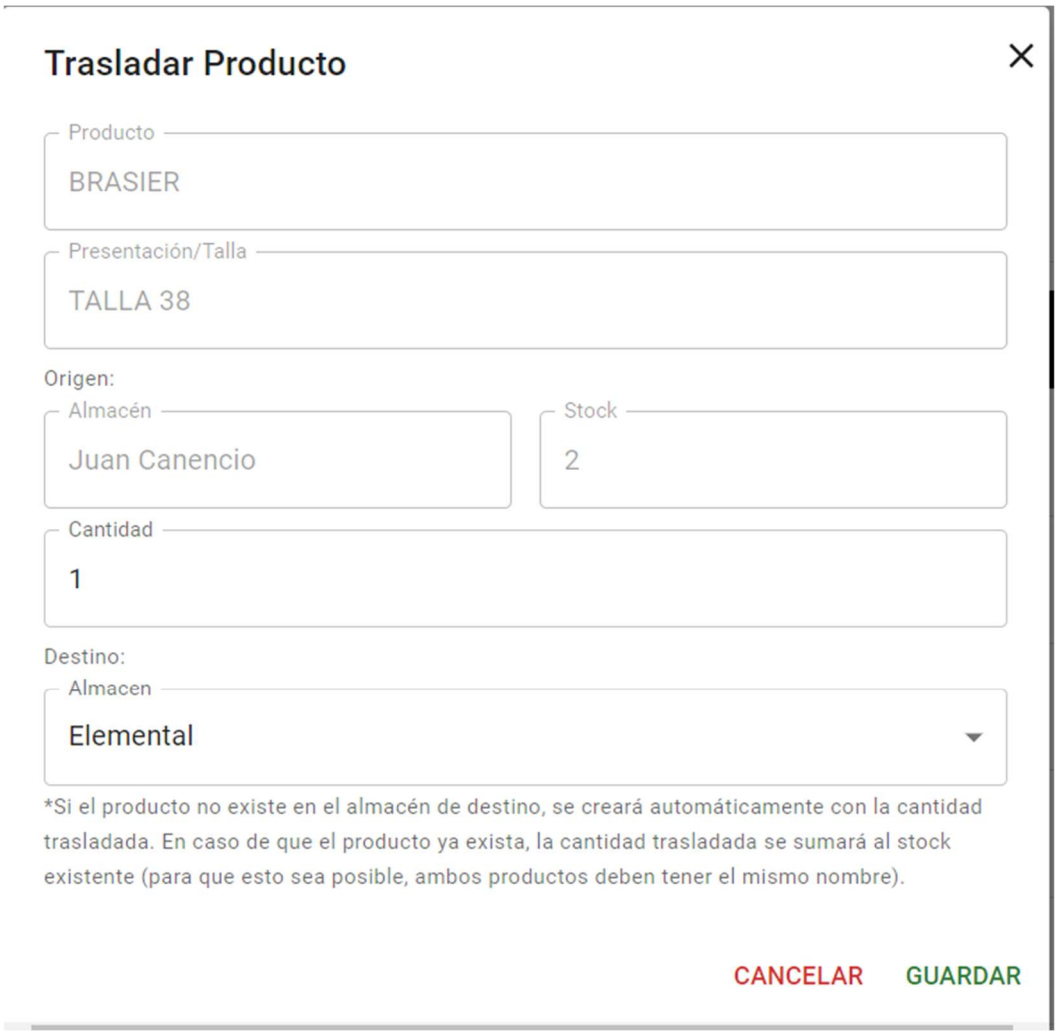
Vista de Administración por Almacén



Nota. Fuente: Autor.

Figura 52

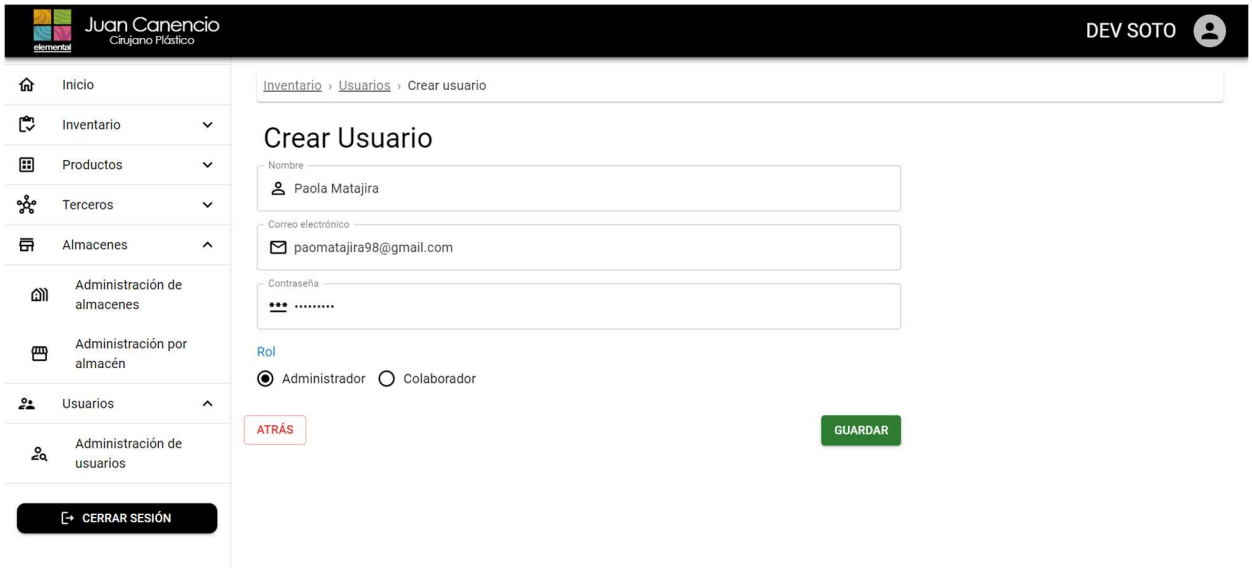
Dialog Modal del Formulario para Traladar Productos



Nota. Fuente: Autor.

Figura 53

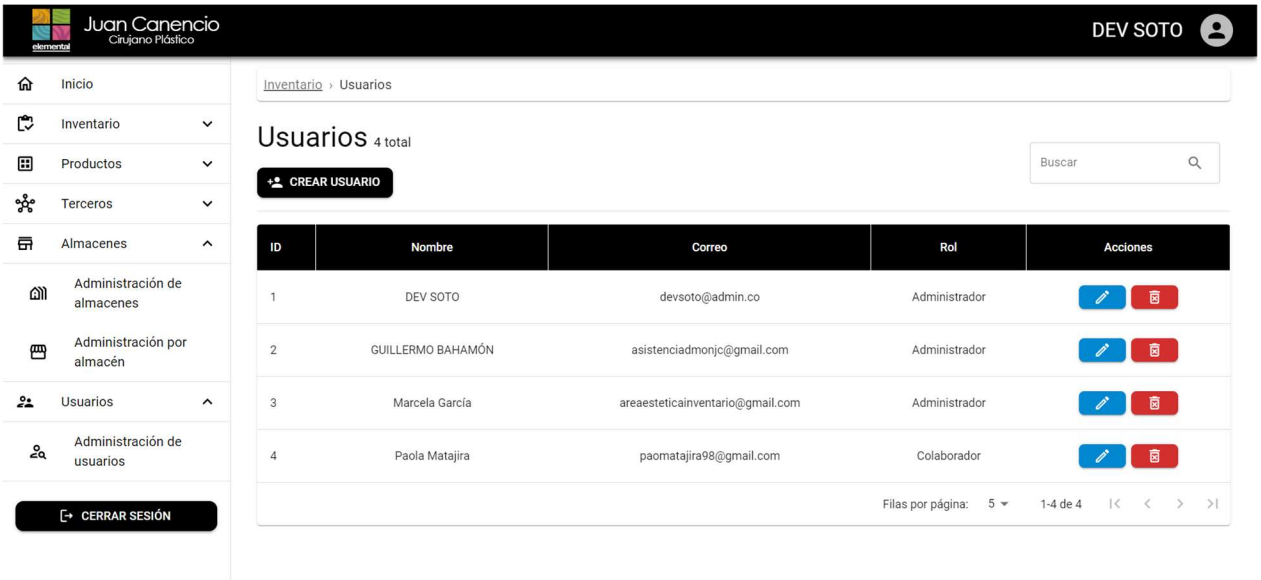
Vista del Formulario para Crear un Nuevo Usuario



Nota. Fuente: Autor

Figura 54

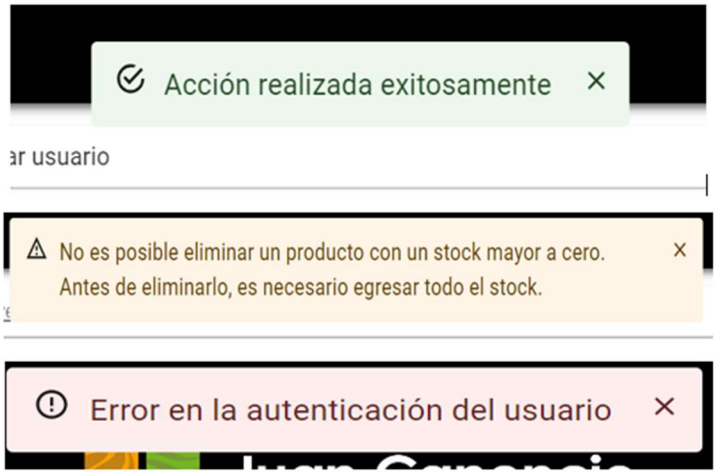
Vista de Administración de Usuario.



Nota. Fuente: Autor

Figura 55

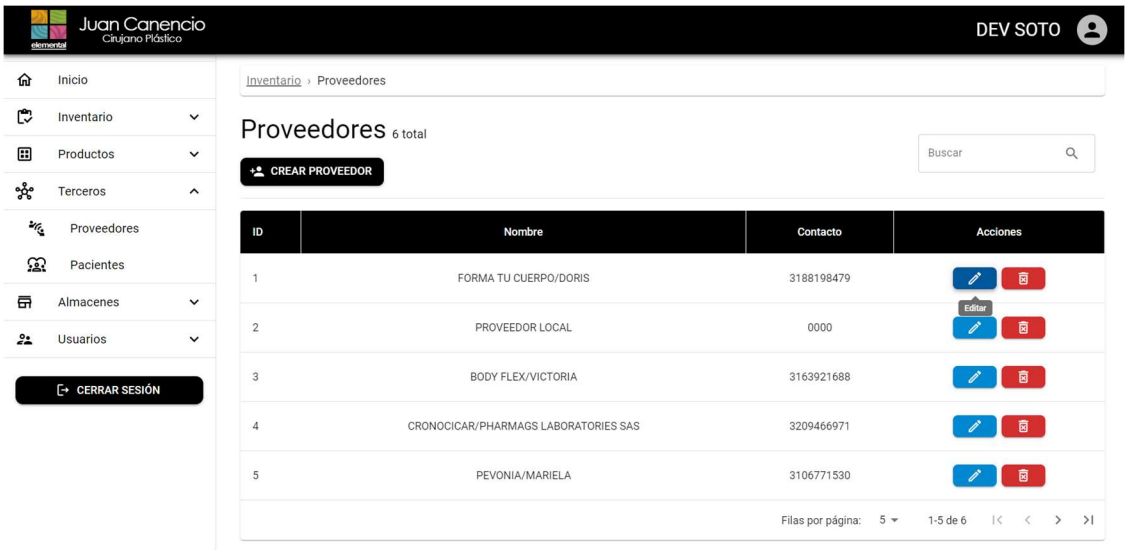
Notificaciones de la Aplicación Web.



Nota. Fuente: Autor

Figura 56

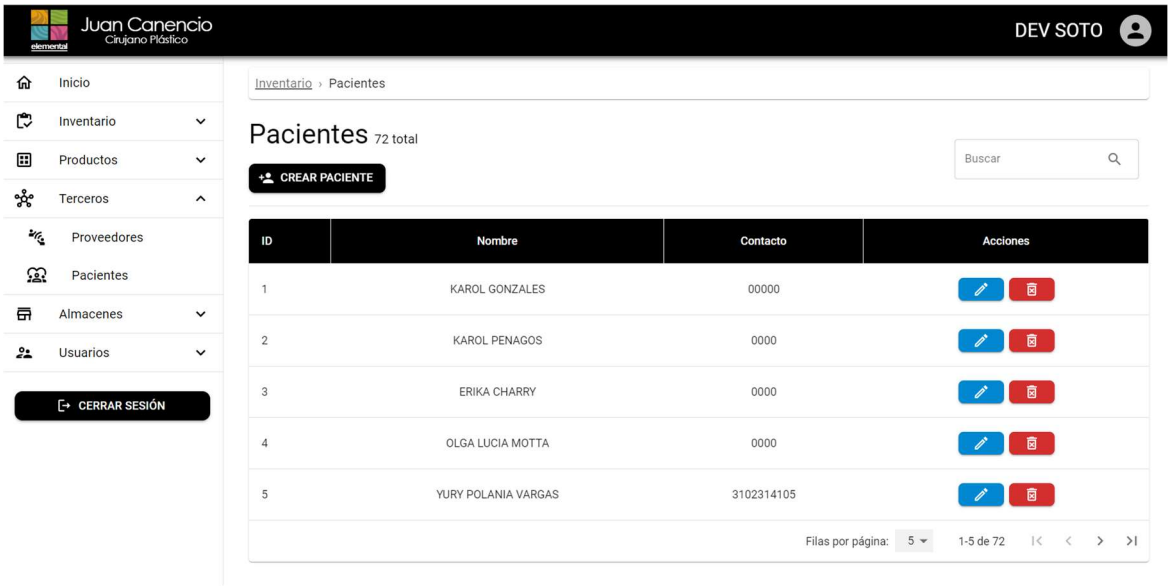
Vista de Administración de Proveedores



Nota. Fuente: Autor

Figura 57

Vista de Administración de Pacientes.



Nota. Fuente: Autor

9.4 Pruebas

En la fase de pruebes, se implementó un plan específico para llevar a cabo tres tipos distintos de evaluaciones: pruebas unitarias, pruebas de portabilidad y pruebas de aceptación.

9.4.1 Objetivo de las Pruebas

El principal objetivo es garantizar que la aplicación web funcione correctamente en términos de operabilidad, configuración y usabilidad.

9.4.2 Descripción general de la Aplicación Web

La aplicación web está diseñada para administrar diversas operaciones relacionadas con el inventario, incluyendo ingresos y egresos de productos. También permite la creación, consulta, edición y eliminación de pacientes, proveedores, categorías, productos, usuarios y almacenes. Las funcionalidades disponibles varían según el rol del usuario en el sistema.

9.4.3 Módulos de la Aplicación Web

- Almacenes.
- Inventario.
- Proveedores.
- Usuarios.
- Pacientes.
- Productos.
- Categorías.
- Sesión.

9.4.4 Formularios

- Crear almacén.
- Crear categoría.
- Crear proveedor.
- Crear producto.
- Crear usuario.
- Crear paciente.
- Editar almacén.
- Editar categoría.
- Editar proveedor.
- Editar producto.
- Editar usuario.
- Editar paciente.
- Trasladar producto.

- Iniciar sesión.

9.4.4.1 Cronograma de pruebas

En la Tabla 14 se observa el cronograma de pruebas.

Tabla 14

Cronograma de Pruebas

	Cronograma de pruebas			
	Mayo		Junio	
	30	31	2	3
Tipo de prueba	Martes	Miercoles	Viernes	Sabado
Unitaria				
Portabilidad				
Aceptación				

Nota. Fuente: Autor

9.4.5 Ejecución de las Pruebas

9.4.5.1 Pruebas Unitarias

Se detallan a continuación los campos que se evidencian en la Tabla 15 de pruebas unitarias:

- Id: Identificador único de la prueba.
- Proceso: Ámbito o funcionalidad a la que está destinada la prueba.
- Actividad: Tarea específica dentro del proceso que se está probando.
- Descripción: Explicación concisa del objetivo de la prueba.
- Resultado Esperado: Qué se espera que ocurra una vez que se realice la prueba.

- Cumplió: Indicación de si la prueba fue exitosa o no.
- Encargado: Persona responsable de realizar la prueba.
- Riesgos: Posibles riesgos asociados con la ejecución de la prueba.

Tabla 15

Pruebas Unitarias

Pruebas Unitarias								
ID	**Proceso**	**Actividad**	**Descripción**	**Resultado esperado**	**¿Cumplido? SI/NO**	**Encargado**	**Riesgos**	**Observaciones**
PU1	Crear	Almacén	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU2			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU3			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU4		Categoría	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU5			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU6			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU7		Proveedor	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU8			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU9			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU10		Paciente	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU11			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU12			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU13		Producto	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU14			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU15			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU16		Movimiento	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU17			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU18			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU19		Usuario	Datos vacíos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU20			Datos erroneos	Que no cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU21			Datos correctos	Que cree	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU22	Editar	Almacén	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU23			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU24			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU25		Categoría	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU26			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU27			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU28		Proveedor	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU29			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU30			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU31		Paciente	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU32			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU33			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU34		Producto	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU35			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU36			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU37		Usuario	Datos vacíos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU38			Datos erroneos	Que no edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU39			Datos correctos	Que edite	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU40	Eliminar	Categoría	Datos correctos	Que elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU41		Proveedor	Datos correctos	Que elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU42		Paciente	Datos correctos	Que elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU43		Categoría	Datos correctos	Que elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU44		Producto	Stock en cero	Que elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU45			Stock mayor a cero	Que no elimine	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU46	Consultar	Almacén	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU47			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU48		Categoría	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU49			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU50		Proveedor	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU51			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU52		Paciente	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU53			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU54		Producto	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU55			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU56		Movimiento	Datos erroneos	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU57			Datos correctos	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU58		Usuario	Nombre erroneo	Que no consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU59			Nombre correcto	Que consulte	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU60	Imprimir	Movimientos	Datos correctos	Que imprima	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU61	Login	Iniciar sesión	Datos vacíos	Que no inicie sesión	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU62			Datos erroneos	Que no inicie sesión	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU63			Datos correctos	Que inicie sesión	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU64	Trasladar	Producto	Datos vacíos	Que no traslade	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU65			Datos erroneos	Que no traslade	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	
PU66			Datos correctos	Que traslade	SI	Juan Ignacio Soto	Fallos en el servidor local de pruebas	

Nota. Fuente: Autor

9.4.5.2 Pruebas de portabilidad

Para evaluar la portabilidad, se consideraron tres tipos distintos de pantallas: computadora, celular y tablet. En estos dispositivos, se examinaron cinco elementos clave de la aplicación: imágenes, títulos, menús, formularios y botones. Se incluyó además una sección para anotar observaciones. Los tamaños exactos de las pantallas de los dispositivos móviles se detallan en la tabla 16.

Tabla 16

Pruebas de Portabilidad

Tipo de pantalla	Imágenes	Títulos	Menús	Formularios	Botones	Observaciones
Computador	SI	SI	SI	SI	SI	
Celular(914x412)	SI	SI	SI	SI	SI	El aplicativo se debe usar en horizontal.
Tablet (820 x 1180)	SI	SI	SI	SI	SI	

Nota. Fuente: Autor

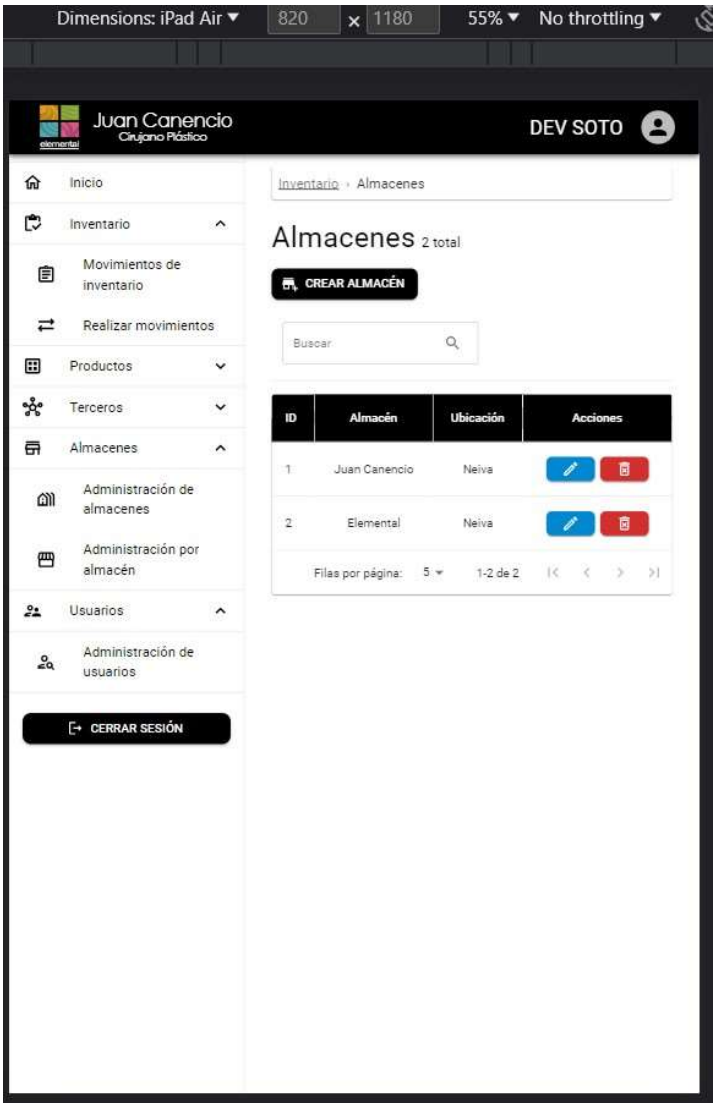
Figura 58

Visualización de la Aplicación Web en un Celular Samsung Galaxy A51

Nota. Fuente: Autor

Figura 59

Visualización de la Aplicación Web en una Tablet iPad Air



Nota. Fuente: Autor

9.4.5.3 Pruebas de Aceptación

Las pruebas de aceptación fueron ejecutadas por Paola Matajira, quien está a cargo del inventario en la empresa Elemental Centro de Estética. Como se observa en la Tabla 17 y 18, estas pruebas se llevaron a cabo evaluando cada uno de los requerimientos específicos, previamente definidos durante la fase de análisis, en función de los distintos roles de usuario.

Tabla 17

Prueba de Aceptación en el Rol de Colaborado

Nombre de Quien Evalúa	Paola Matajira		
Rol	Colaborador		
Fecha	2/06/2023		
HU a Evualuar	**Valoración**		**Observaciones**
	Aceptada	**No Aceptada**	
HU-1	SI		
HU-2	SI		
HU-14	SI		
HU-15	SI		
HU-16	SI		
HU-17	SI		

Nota. Fuente: Autor.

Tabla 18

Prueba de Aceptación en el Rol de Administrador

Nombre de Quien Evalúa	Paola Matajira		
Rol	Administrador		
Fecha	2/06/2023		
HU a Evualuar	**Valoración**		**Observaciones**
	Aceptada	**No Aceptada**	
HU-1	SI		
HU-2	SI		
HU-3	SI		
HU-4	SI		
HU-5	SI		
HU-6	SI		
HU-7	SI		
HU-8	SI		
HU-9	SI		
HU-10	SI		
HU-11	SI		
HU-12	SI		
HU-13	SI		
HU-14	SI		
HU-15	SI		
HU-16	SI		
HU-17	SI		

Nota. Fuente: Autor.

10 Conclusiones

- Se desarrolló el prototipo de aplicación web tras un detallado análisis y levamiento de requerimientos para el proyecto “Desarrollo De Aplicación De Gestión de Inventario Para Elemental Centro De Estética: Un Enfoque Ágil Con Tecnologías MERN” con el objetivo de automatizar procesos internos, lo cual permite un registro y control más eficiente del inventario, facilitando la gestión de almacenes de Elemental.
- Crear un diseño de base de datos a partir de requisitos bien definidos ofrece una comprensión clara del sistema que se planea construir. Este enfoque ayuda a evitar la repetición innecesaria de información y hace que los procesos sean más fáciles de llevar a cabo y monitorear.
- La aplicación de la metodología Scrum facilita una colaboración continua entre todos los participantes del proyecto, incluido el cliente. Este enfoque es clave para entender y atender las necesidades de los usuarios. Mediante un proceso formal, se identifican y documentan los requerimientos, que luego se reflejan en el Product Backlog, una herramienta esencial en Scrum que lista las funciones que se van a desarrollar.
- La selección de herramientas tecnológicas juega un papel importante en el desarrollo del prototipo de aplicación web. Por eso es crucial optar por una pila de desarrollo que se adapte al tipo de sistema que se va a crear. En este caso, la decisión de usar la pila MERN (Mongo, Express, React, Node.JS), ha resultado ser apropiada pues facilitó el proceso de trabajar con la arquitectura modelo vista controlador (MVC).
- La utilización de herramientas como GitHub y Visual Studio Code fue instrumental para el éxito del proyecto. GitHub facilitó la colaboración y el seguimiento del progreso, mientras que Visual Studio Code proporcionó un entorno de desarrollo robusto y versátil. La combinación de estas herramientas, junto con las tecnologías React, Redux y Material-

UI, resultó en un desarrollo fluido y en una aplicación final funcional, intuitiva y visualmente atractiva.

Bibliografía

Ehrhardt, M. y Brigham, E. (2007). Finanzas corporativas. Segunda edición. México: Editorial Thomson

Martínez Orencio, A. (2017). *La información en la organización, su gestión y auditoría*. GestioPolis. https://www.gestiopolis.com/la-informacion-en-la-organizacion-su-gestion-y-auditoria

Durán, Y. (2012). *Administración del inventario: elemento clave para la optimización de las utilidades en las empresas*. Redalyc.org. https://www.redalyc.org/articulo.oa?id=465545892008

J, P. P., & Merino, M. (2021). Stock - Qué es, en la geología, definición y concepto. *Definición.de*. https://definicion.de/stock

Escudero, E. (2021, 5 julio). Los 10 startups de tecnología que revolucionarán la logística (Primera parte). *THE LOGISTICS WORLD | Conéctate e inspírate.* https://thelogisticsworld.com/innovacion/las-10-startups-de-tecnologia-que-revolucionaran-la-logistica-primera-parte

Departamento Nacional de Planeación (2020). *Encuesta Nacional Logística 2020.* Colombia https://plc.mintransporte.gov.co/Portals/0/News/Encuesta%20Nacional%20Logi%CC%81stica%202020.pdf?ver=2021-09-24-211753-007

Mateu, C. (2004, 1 marzo). *Desarrollo de aplicaciones web*. https://openaccess.uoc.edu/bitstream/10609/224/1/Desarrollo%20de%20aplicaciones%20web.pdf

De Luca, D. (2013). *APSS HTML5 para móviles - Desarrollo de aplicaciones para smartphones y tablets basado en tecnologías web*. Alfaomega Grupo Editor.

Cecibel, S. C. (2015). *El Control de los Inventarios y su Aporte en los Estados Financieros de la Empresa.* Machala. http://repositorio.utmachala.edu.ec/bitstream/48000/3100/1/TTUACE-2015-CA-CD00070.pdf

Hope, P., & Walther, B. (2009). *Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast*. «O'Reilly Media, Inc.»

Navarro Cadavid A., Fernández Martínez, J., Morales Vélez, J., Andrés, N., Juan, F., & Morales, J. (2013). *Revisión de metodologías ágiles para el desarrollo de software.* Bogotá, Colombia: Universidad Autónoma del Caribe, Prospectiva. http://repositorio.uac.edu.co/handle/11619/1212

Griffiths, M. (2012). *PMI-ACP exam prep: Rapid Learning to Pass the PMI Agile Certified Practitioner (PMI-ACP) Exam - on Your First Try!* Conran Octopus.

Diaz, J. (2009). *Las metodologías ágiles como garantía de calidad del software*. Madrid, España: REICIS. Revista Española de Innovación. https://www.redalyc.org/pdf/922/92217181006.pdf

Silva, P. T. (2019). *Ecosistema digital de comercio electrónico y marketing digital para la empresa Elemental Centro de Estética*. Universidad Nacional Abierta y a Distancia. https://repository.unad.edu.co/handle/10596/28249

IBM. (2021*). ¿Qué es la gestión de inventarios y cómo funciona?* IBM

https://www.ibm.com/es-es/topics/inventory-management

Cohn, M. (2010). *Succeeding with Agile: Software Development Using Scrum*. Pearson Education.

Raeburn, A. (2022). ¿Qué es la programación extrema (XP)? [2022] • Asana. *Asana*. https://asana.com/es/resources/extreme-programming-xp

Salazar, J. C., Tovar, Á., Linares, J. C., Lozano, A., & Valbuena, L. (2018). Scrum versus XP: similitudes y diferencias Scrum vs XP: Similarities and Differences. *Tia*, *6*(2), 29–37. https://revistas.udistrital.edu.co/index.php/tia/article/view/10496/14690

Rad, N. K., & Turley, F. (2019). *Los fundamentos de Agile Scrum*. Van Haren.

¿Quiénes son los stakeholders? | Alaimo Labs. (2022). Alaimo Labs. https://alaimolabs.com/es/self-learning/agile-product-management/quienes-son-los-stakeholders

Bahit, E. (2012). *Scrum & Extreme Programming Para Programadores*. La web del programador. https://www.lawebdelprogramador.com/pdf/8829-Scrum-&-Extreme-Programming-para-programadores.html

Lucidchart. (2021). *Qué es el Lenguaje Unificado de Modelado (UML)*. Lucidchart. https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml#:~:text=un%20diagrama%20UML-,%C2%BFQu%C3%A9%20es%20UML%3F,en%20estructura%20como%20en%20comportamiento.

Manzanares, N. A. F. (2018). *Diagramas de casos de uso | LENGUAJE DE MODELADO UNIFICADO UML*. UNAD https://unadzsurlab.com/UML/U1/diagramas_de_casos_de_uso.html

Gómez, P. (2021). *Qué es una librería en programación - DevCamp*. DevCamp. https://devcamp.es/que-es-libreria-programacion

Emmita. (2017, 14 octubre). Frameworks - Emmita - medium. *Medium*. https://medium.com/@Emmitta/frameworks-541da921ddcc

Pérez, J. E. (2009*). Introducción a JavaScript*. Librosweb.es http://dis.um.es/~lopezquesada/documentos/IES_1314/IAW/curso/UT7/libroswebjavascript/www.librosweb.es/javascript/index.html

MongoDB. (s. f.). *¿Qué es MongoDB?* https://www.mongodb.com/es/what-is-mongodb

Blancarte Iturralde, O. J. (2020). *Introducción a la arquitectura de software – Un enfoque práctico.* O. J. Blancarte Iturralde (ed.); Primera Ed, Oscar Blancarte blog software architect.

Luján Mora, S. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web.* En Programación de aplicaciones web: historia, principios básicos y clientes web (Editorial, p. 48). https://sergiolujanmora.es/verpdf/42

Blasco, R. V. L., Talón, E. M., & Andrés, J. A. M. (2013). *Aplicaciones web, grado medio*. Editorial McGraw-Hill Ciclos Formativos.

Eslava Muñoz, V. J. (2013). *El nuevo PHP. Conceptos avanzados*. España: Bubok. Publishing S.L.

Red Hat. (2019). *¿Qué es una API?* Documentación extraída de Red Hat acerca del Concepto de API (Application Programming Interface). https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Red Hat. (2019). *Diferencias entre REST y SOAP*. Red Hat. https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest

Red Hat. (2020). *¿Qué es una API de REST?* Red Hat. https://www.redhat.com/es/topics/api/what-is-a-rest-api

W3C. (2010). *Web of Services. W3C web site*. https://www.w3.org/standards/webofservices

Oracle. (2022). *¿Qué es una base de datos?* Oracle Colombia. https://www.oracle.com/co/database/what-is-database

Amazon. (2020). *Bases de datos SQL | AWS*. Amazon AWS. https://aws.amazon.com/es/relational-database

Amazon. (2019). *MongoDB en AWS: Quick Start. Amazon AWS*. Amazon AWS . https://aws.amazon.com/es/quickstart/architecture/mongodb-atlas

Amazon. (2020). *Bases de datos no relacionales | Bases de datos de gráficos | AWS*. Amazon AWS. https://aws.amazon.com/es/nosql

Next U. (2022). *¿Qué es React JS y cómo funciona?* Blog | NextU LATAM. https://www.nextu.com/blog/que-es-react-js-como-funciona-rc22

Gonzales, J. (2023*). Styled Components y Material UI. ¿Qué son y cómo utilizarlas en tu proyecto REACT?* Alura. https://www.aluracursos.com/blog/styled-components-material-ui-que-son-como-utilizarlas#:~:text=%C2%BFQu%C3%A9%20es%20Material%20UI%3F,de%20dise%C3%B1o%20creado%20por%20Google

OpenJS. (2023). *Node.js*. Nodejs.Org. https://nodejs.org/es/about

Express.js. (2022). *Express - Infraestructura de aplicaciones web Node.js*. Express. https://expressjs.com/es

Parada, M. (2020). *MERN Stack: qué es y qué ventajas ofrece*. OpenWebinars. https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece

Beck, K., & Gamma, E. (2000). JUnit.

International Organization for Standardization. (2005). *ISO/IEC 25000:2005, Software engineering — Software product Quality Requirements and Evaluation* (SQuaRE).

Dustin, A. A. E. (2002*). Automated Software Testing: Introduction, Management, and Performance*. Artech House.

Pressman, R. S., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

Karnik, N. (2018). *Introduction to Mongoose for MongoDB*. FreeCodeCamp. https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57

Auth0 Inc. (2020). *JWT.IO - JSON Web Tokens Introduction*. JSON Web Tokens - jwt.io. https://jwt.io/introduction

Mijacobs. (2023). *¿Qué es Git? - Azure DevOps*. Microsoft Learn. https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

GitHub (s. f.). *Build software better, together.* GitHub. https://github.com/about