UNIVERSIDAD LAICA ELOY ALFARO DE MANABÍ

CARRERA:

TECNOLOGÍAS DE LA INFORMACIÓN

ASIGNATURA:

TRABAJO DE INTEGRACION CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE: INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

TUTORA:

PATRICIA ALEXANDRA QUIROZ PALMA

AUTORES:

DEVIS YONAIKEL ALONZO PICO

BAILON QUIJIJE JERICK ABRAHAM

TEMA:

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTION DE INVENTARIOS EN LA ESCUELA DE PESCA DEL PACIFICO ORIENTAL

CARRERA:

TECNOLOGÍAS DE LA INFORMACIÓN

PERIODO ACADÉMICO:

2025 (2)

Uleam UNIVERSIDAD LAICA ELOY ALFARO DE MANABÍ	NOMBRE DEL DOCUMENTO: CERTIFICADO DE TUTOR(A)	CÓDIGO: PAT-04-F-004
	PROCEDIMIENTO: TITULACIÓN DE ESTUDIANTES DE GRADO	REVISIÓN: 1
		Página 1 de 1

CERTIFICACIÓN

En calidad de docente tutor(a) de la Facultad de Ciencias Informáticas de la Universidad Laica "Eloy Alfaro" de Manabí, CERTIFICO:

Haber dirigido y revisado el trabajo de investigación, bajo la autoría del estudiante ALONZO PICO DEVIS YONAIKEL, legalmente matriculados en la carrera de Ingeniería en Tecnologías de la Información, período académico 2025-2026, cumpliendo el total de 380 horas, bajo la opción de titulación de proyecto integrador, cuyo tema del proyecto o núcleo problémico es "DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO EN LA ESCUELA DE PESCA DEL PACIFICO ORIENTAL (EPESPO)".

La presente investigación ha sido desarrollada en apego al cumplimiento de los requisitos académicos exigidos por el Reglamento de Régimen Académico y en concordancia con los lineamientos internos de la opción de titulación en mención, reuniendo y cumpliendo con los méritos académicos, científicos y formales, suficientes para ser sometida a la evaluación del tribunal de titulación que designe la autoridad competente.

Particular que certifico para los fines consiguientes, salvo disposición de Ley en contrario.

Lugar, Manta 04 de febrero de 2026.

Lo certifico,

Ing. Patricia Quiroz Palma, Phd.
Docente Tutor(a)
Área: Tecnología de la información

Uleam	NOMBRE DEL DOCUMENTO: CERTIFICADO DE TUTOR(A)	CÓDIGO: PAT-04-F-004
	PROCEDIMIENTO: TITULACIÓN DE ESTUDIANTES DE GRADO	REVISIÓN: 1
		Página 1 de 1

CERTIFICACIÓN

En calidad de docente tutor(a) de la Facultad de Ciencias Informáticas de la Universidad Laica "Eloy Alfaro" de Manabí, CERTIFICO:

Haber dirigido y revisado el trabajo de investigación, bajo la autoría del estudiante BAILON QUIJIJE JERICK ABRAHAM, legalmente matriculados en la carrera de Ingeniería en Tecnologías de la Información, período académico 2025-2026, cumpliendo el total de 380 horas, bajo la opción de titulación de proyecto integrador, cuyo tema del proyecto o núcleo problémico es "DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE INVENTARIO EN LA ESCUELA DE PESCA DEL PACIFICO ORIENTAL (EPESPO)".

La presente investigación ha sido desarrollada en apego al cumplimiento de los requisitos académicos exigidos por el Reglamento de Régimen Académico y en concordancia con los lineamientos internos de la opción de titulación en mención, reuniendo y cumpliendo con los méritos académicos, científicos y formales, suficientes para ser sometida a la evaluación del tribunal de titulación que designe la autoridad competente.

Particular que certifico para los fines consiguientes, salvo disposición de Ley en contrario.

Lugar, Manta 04 de febrero de 2026.

Lo certifico,

Ing. Patricia Quiroz Palma, Phd.
Docente Tutor(a)
Área: Tecnología de la información

DECLARACIÓN DE AUTORÍA

Nosotros, ALONZO PICO DEVIS YONAIKEL y BAILON QUIJIJE JERICK ABRAHAM, en calidad de autores del trabajo de titulación: "DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTION DE INVENTARIOS EN LA ESCUELA DE PESCA DEL PACIFICO ORIENTAL", autorizamos a la Universidad Laica Eloy Alfaro de Manabí, hacer uso completo o parcial del contenido de este trabajo de titulación del cual somos responsables, con fines estrictamente académicos o de investigación.

Los derechos que como autores nos corresponden, con excepción de la presente autorización, seguirán vigentes a nuestro favor, de conformidad con lo establecido en los artículos 5, 6, 8, 19 y demás artículos pertinentes de la Ley de Propiedad Intelectual y su Reglamento.

Lugar, Manta 25 de febrero de 2026

Jerick Abraham Bailón Quijije
C.I. 1315299923

Devis Yonaikel Alonzo Pico
C.I. 1315964393

DEDICATORIA

Dedico este trabajo de titulación a mi padre, quien ha sido mi pilar fundamental en cada paso de este camino y quien, con su amor incondicional, sacrificios constantes y sabios consejos, me ha enseñado el verdadero valor del esfuerzo y la perseverancia.

A mi grupo de amigos, con quienes he compartido innumerables experiencias a lo largo de mi vida universitaria. Gracias por estar siempre presentes, tanto en los buenos como en los momentos difíciles. Cada uno ha contribuido de manera única y especial a mi desarrollo personal y académico.

También dedico este logro a mi novia, quien ha sido muy importante en la etapa final de este proceso. Desde que la conocí, me ha ayudado y brindado su apoyo incondicional, comprensión y motivación constante para seguir adelante y culminar esta meta.

Devis Yonaikel Alonzo Pico

DEDICATORIA

Dedico este trabajo de titulación a mi familia, que siempre me apoyaron durante mis estudios, fueron los primeros en creer en mí y mis capacidades, son ellos los que me incentivaron a seguir adelante durante toda la carrera.

También dedico este trabajo a mis amigos, con su ayuda y consejos el logrado seguir adelante en mi formación profesional buscando siempre mi mejoría.

Por último, le dedico esto a los profesores que, durante la carrera, me apoyaron siempre incentivando mi desarrollo como profesional.

Jerick Abraham Bailón Quijije

AGRADECIMIENTO

Expreso mi más sincera gratitud a todas aquellas personas que aportaron de manera significativa a la culminación de este proyecto de titulación.

En primer lugar, agradezco a Dios por su guía y misericordia infinita, por brindarme la fortaleza, sabiduría e inteligencia necesarias para superar cada desafío presentado a lo largo de este proceso académico.

las personas que más admiro y que se ha convertido en mi mayor ejemplo a seguir, mi padre, por su amor, paciencia y apoyo incondicional durante todo este camino. Le debo gran parte de este logro, ya que sin su esfuerzo y dedicación no habría sido posible alcanzarlo.

A mis profesores y tutores, por compartir sus conocimientos y experiencias con compromiso y vocación, así como por su orientación constante durante el desarrollo de este trabajo.

Finalmente, expreso mi agradecimiento a la **Universidad Laica Eloy Alfaro de Manabí**, por brindarme la oportunidad de formarme académica y personalmente, y por proporcionarme las herramientas necesarias para alcanzar mis metas profesionales.

Devis Yonaikel Alonzo Pico

AGRADECIMIENTO

Quiero agradecer en este apartado a todas las personas que creyeron en mi durante y me apoyaron durante mi carrera universitaria.

Principalmente quiero agradecer a Dios por darme las fuerzas y salud cada día para seguir avanzando.

A mis padres por ser un ejemplo de perseverancia y superación constante por criarme e imponerme estas cualidades, sobre todo quiero agradecer a mi padre por siempre apoyarme durante mis estudios ya sea en ámbito económico o emocional, el siempre me apoyo. Mi madre que siempre estuvo constantemente cuidando de mi salud a pesar de la distancia, seguía sintiendo el amor de una madre.

A mis profesores y tutores que siempre me guiaron con su sabiduría y experiencia durante mi formación profesional, en esta tesis también quiero agradecerles a ellos.

Por últimos, agradezco a la Universidad Laica Eloy Alfaro de Manabí, por darme la oportunidad de formarme como profesional.

Jerick Abraham Bailón Quijije

Índice

Índice de Figuras

Índice de Tablas

RESUMEN

Este trabajo de titulación aborda el desarrollo de una solución tecnológica a medida para la Escuela de Pesca del Pacífico Oriental (EPESPO), con el fin de modernizar su control patrimonial. La investigación se enfocó en erradicar la dependencia de registros manuales y hojas de cálculo, cuyas limitaciones afectaban la trazabilidad de los bienes y aumentaban el riesgo de inconsistencias operativas.

Para la construcción del sistema se aplicó una metodología ágil, integrando React en el diseño de interfaces dinámicas, Laravel para la seguridad y lógica del servidor, y PostgreSQL como motor de base de datos, garantizando así una arquitectura robusta y escalable.

La implementación de esta plataforma centraliza la gestión del inventario, permitiendo digitalizar flujos críticos como las asignaciones y bajas de activos. Esto no solo optimiza los tiempos de respuesta administrativa, sino que dota a la institución de reportes precisos en tiempo real para una toma de decisiones transparente y eficiente.

Palabras claves: Aplicación web, Gestión de Inventarios, EPESPO, Laravel, React, PostgreSQL, Trazabilidad, Automatización de Procesos.

ABSTRACT

This degree thesis addresses the development of a custom technological solution for the Eastern Pacific Fishing School (EPESPO), aiming to modernize its asset control. The research focused on eradicating reliance on manual records and spreadsheets, whose limitations hindered asset traceability and increased the risk of operational inconsistencies.

For the system's construction, an agile methodology was applied, integrating React for dynamic interface design, Laravel for server security and logic, and PostgreSQL as the database engine, thus ensuring a robust and scalable architecture.

The implementation of this platform centralizes inventory management, allowing for the digitization of critical flows such as asset assignments and disposals. This not only optimizes administrative response times but also equips the institution with precise, real-time reports for transparent and efficient decision-making.

Keywords: Web Application, Inventory Management, EPESPO, Laravel, React, PostgreSQL, Asset Control.

Capítulo I

1. Marco introductorio

1.1 Introducción

Hoy en día, el desarrollo tecnológico ha impulsado a las organizaciones e instituciones a adoptar soluciones digitales que permitan una gestión más eficiente de sus procesos internos.

La gestión de inventarios representa un desafío operativo constante para cualquier institución. Cuando no existe un control riguroso, la desorganización tiende a prevalecer rápidamente, afectando la disponibilidad de los recursos. Llevar un registro exacto va más allá de un simple conteo físico; requiere mantener una trazabilidad precisa para conocer la ubicación y el estado de los equipos en el momento que se necesitan. En el entorno institucional, mantener esta información actualizada resulta vital. Sin embargo, depender del uso prolongado de hojas de cálculo y registros en papel expone los datos a constantes errores de transcripción o extravíos. Esta forma de trabajo retrasa los procesos administrativos y disminuye drásticamente la eficiencia operativa, demostrando que los métodos de control manuales resultan insuficientes para las demandas actuales.

La asignación manual de responsabilidades y la generación de reportes presentan una alta complejidad operativa que retrasa los procesos administrativos. Ante este panorama, se requiere una reestructuración integral del proceso mediante el desarrollo de una nueva arquitectura de software. La propuesta es desarrollar una aplicación web que automatice de raíz la gestión de inventarios. La propuesta consiste en desarrollar una aplicación web que automatice la gestión de inventarios, funcionando como el eje central de la administración.

1.1.1 Ubicación y contextualización de la problemática

La Escuela de Pesca del Pacífico Oriental constituye una unidad académica caracterizada por custodiar un volumen considerable de activos, que abarca desde equipos especializados y herramientas técnicas hasta insumos materiales y mobiliario diverso. Estos recursos, dispersos operativamente entre laboratorios, despachos administrativos y zonas

comunes, representan el soporte material ineludible para la ejecución de las funciones sustantivas de docencia e investigación. La administración eficiente de este patrimonio constituye un proceso crítico y fundamental para garantizar la continuidad operativa de la institución. La administración eficiencia es un pilar fundamental para la operatividad de la institución. En la actualidad, gestionar los inventarios en la EPESPO mediante métodos manuales resulta ineficiente e insostenible. Al depender de hojas de cálculo y herramientas no relacionales, la desorganización compromete la trazabilidad de los bienes. Se pierde información. Nadie sabe con certeza quién es responsable de qué. Ante la carencia de una base de datos centralizada que organice los registros, cualquier intento de auditoría o reposición termina siendo un proceso lento, torpe y lleno de errores evitables.

La llegada de una aplicación web diseñada específicamente para este fin cambia las reglas del juego por completo.

Simultáneamente, la digitalización del proceso fortalece la rendición de cuentas, instaurando un marco de transparencia y responsabilidad más riguroso en el manejo del patrimonio de la entidad.

1.2 Planteamiento del problema

A nivel macro, la gestión eficiente de inventarios en instituciones de educación y formación técnica resulta indispensable para transparentar el uso de bienes públicos o privados, garantizando así el normal desarrollo de las actividades operativas.

A nivel meso, la Escuela de Pesca del Pacífico Oriental (EPESPO) enfrenta dificultades operativas derivadas de la falta de centralización de la información. El personal administrativo invierte demasiado tiempo intentando localizar bienes específicos o conciliando registros físicos que se encuentran dispersos en diferentes departamentos.

A nivel micro, esta dependencia de procesos manuales genera consecuencias técnicas directas que afectan el rendimiento del sistema de control. Las bases de datos ofimáticas que

manejan actualmente presentan inconsistencias severas, existe un alto riesgo de duplicar registros y carecen de control de concurrencia. Todo esto impide generar reportes estadísticos fidedignos en tiempo real para respaldar las auditorías y decisiones de la directiva.

1.3 Formulación del Problema

El problema Principal que refleja este proyecto es que no hay una plataforma para gestionar de manera adecuada todos los bienes que tiene la Epespo.

1.3.1 Problema General

¿De qué manera el desarrollo de una aplicación web optimizará la gestión de inventarios y el control patrimonial en la Escuela de Pesca del Pacífico Oriental?

1.3.2 Problemas Específicos

- ¿Cómo afecta la dependencia de registros manuales y hojas de cálculo a la integridad y disponibilidad de la información patrimonial generada por la administración?
- ¿De qué manera la ausencia de un modelo de datos relacional incrementa la vulnerabilidad y la duplicidad en los registros de bienes institucionales?
- ¿En qué medida la falta de una arquitectura de software centralizada con roles de usuario compromete la seguridad y la trazabilidad del ciclo de vida de los activos?
- ¿Qué impacto tiene la carencia de una plataforma automatizada en la eficiencia operativa al momento de generar reportes, consultas y actas de entrega legal?

1.3.3 Pregunta de Investigación

¿Cómo influye la implementación de una plataforma web basada en el patrón de arquitectura MVC en la eficiencia de los procesos de registro, custodia y baja de activos institucionales dentro de la EPESPO?

1.4 Diagrama causa y efecto

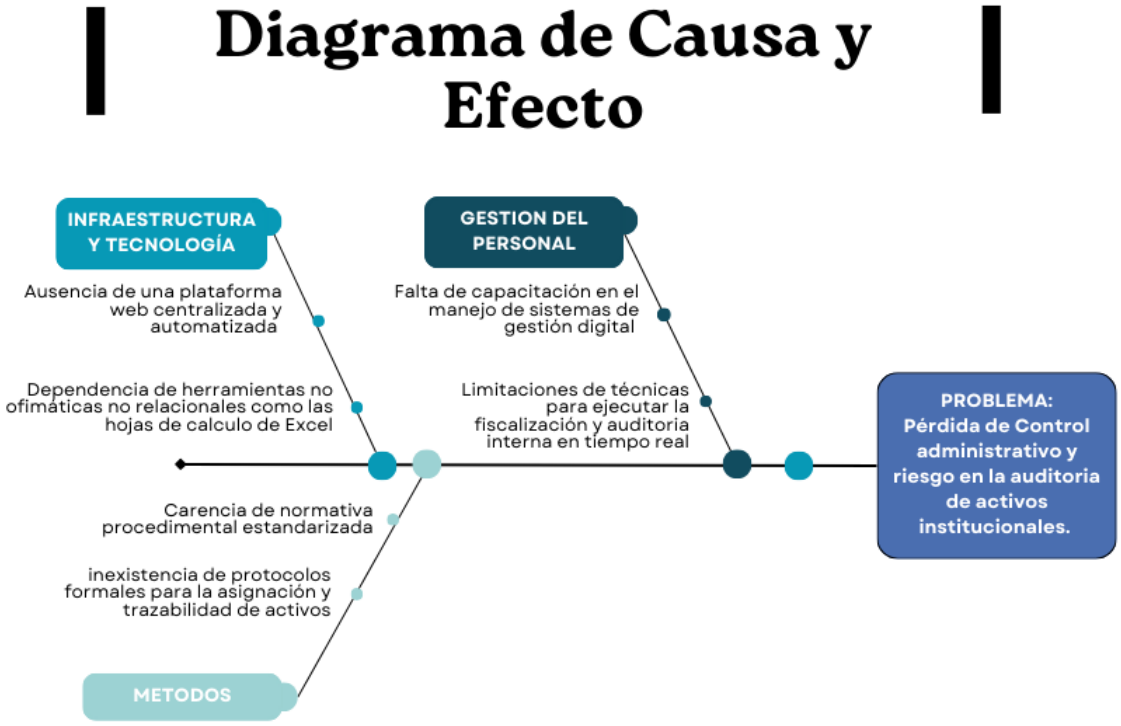

Figura 1. Diagrama Causa y Efecto

EL diagrama de Causa y Efecto conocido también como con múltiples nombres, como el de su creador Ishikawa o espina de Pescado, sirve como una herramienta para analizar de manera estructurada las causas de un problema. Permite clasificar las causas en categorías generales como métodos, procesos, tecnología, personal, entre otras con el objetivo de descubrir no solo los síntomas, sino las causas raíz que contribuyen a un determinado efecto negativo dentro de una organización. En nuestro caso, se ideo un diagrama para entender mejor el problema y sus causas que afecta a la Escuela de Pesca del Pacifico Oriental.

El diagrama refleja las causas principales que originan las deficiencias en el control patrimonial, agrupadas en tres categorías técnicas:

Métodos

1. **Carencia de normativa procedimental actualizada:** La falta de manuales estandarizados impide mantener uniformidad en la ejecución de tareas de registro.

2. **Inexistencia de protocolos para la asignación de activos:** Al no existir una metodología formal para documentar la entrega de bienes, la trazabilidad se pierde y dificulta la fiscalización física de los equipos.

Infraestructura y Tecnología

1. **Dependencia de herramientas ofimáticas no relacionales:** El uso exclusivo de hojas de cálculo incrementa la vulnerabilidad de la información frente a modificaciones accidentales o pérdida de archivos locales.
2. **Ausencia de una plataforma centralizada:** La falta de un sistema automatizado ralentiza la disponibilidad de información actualizada y eleva la probabilidad de duplicar datos.

Gestión del Personal

1. **Insuficiencia de competencias en sistemas digitales:** El personal requiere capacitación orientada a la transición de registros manuales hacia herramientas de gestión informática.
2. **Limitaciones en la auditoría interna:** La falta de herramientas tecnológicas impide al personal administrativo validar la ubicación y el estado de un equipo en tiempo real, dificultando las auditorías institucionales

1.5 Objetivos

1.5.1 Objetivo General

Desarrollar e implementar una aplicación web basada en una arquitectura Cliente-Servidor bajo el patrón MVC, utilizando tecnologías de código abierto, para automatizar el control de activos y la generación de reportes en la EPESPO, optimizando así la integridad de la información institucional.

1.5.2 Objetivo Especifico

- Levantar los requerimientos funcionales y no funcionales mediante técnicas de recolección de datos cualitativas y documentales para definir el alcance técnico del sistema.
- Diseñar la arquitectura de software y el modelo relacional de la base de datos que garantice la persistencia, concurrencia y seguridad de los registros patrimoniales de la institución.
- Construir los módulos de gestión de usuarios, registro de activos, asignación de custodios y generación de reportes utilizando React JS y Laravel, aplicando principios de usabilidad para el usuario final.
- Validar la funcionalidad y rendimiento de la plataforma web mediante pruebas de caja negra, asegurando el cumplimiento de las reglas de negocio antes de su implementación operativa.

1.6 Justificación

La instrumentación de una solución tecnológica orientada a la gestión de inventarios constituye un requisito necesario para asegurar la eficiencia operativa y el control sobre el patrimonio de la organización. El desarrollo de este proyecto se justifica desde tres dimensiones fundamentales:

Justificación Técnica: La propuesta plantea la transición de procesos de registro manuales hacia una arquitectura web escalable. El uso de bases de datos relacionales y frameworks modernos permite resolver problemas críticos de concurrencia e integridad de datos, garantizando que el almacenamiento de la información institucional deje de ser vulnerable a inconsistencias estructurales o extravíos.

Justificación Práctica: A nivel operativo, el sistema automatiza procesos mecánicos y repetitivos. La implementación de módulos de búsqueda y emisión automática de actas optimizará directamente los tiempos de respuesta en la localización y asignación de activos institucionales, permitiendo que el personal disminuya la carga operativa manual.

Justificación Institucional y Social: Modernizar esta herramienta promueve una cultura organizacional enfocada en la transparencia administrativa. Contar con reportes exactos y actualizados fortalece el proceso de toma de decisiones de la directiva, asegurando una rendición de cuentas clara respecto a la adquisición y custodia de los bienes que soportan la actividad educativa de la entidad.

1.7 Alcance y Limitaciones del Estudio

Alcance del Proyecto

El alcance de esta investigación se delimita al diseño y desarrollo de una aplicación web estructurada en cuatro módulos funcionales principales:

1. Módulo de Gestión de Usuarios para la administración de credenciales y roles (Encargado y Director).
2. Módulo de Inventario para el control patrimonial, permitiendo altas, modificaciones y bajas lógicas.
3. Módulo Operativo para el registro transaccional de asignaciones de custodia y devoluciones.
4. Módulo de Documentación para la generación automática de actas de entrega-recepción en formato PDF.

Limitaciones del Estudio

- **Limitación Tecnológica:** La disponibilidad operativa del sistema depende de la infraestructura de red de la institución, requiriendo conexión a internet o intranet para el acceso al servidor donde se aloja la plataforma.
- **Limitación de Datos:** El software actúa como un registro digital que asume la veracidad de la información ingresada por los operadores; no realiza auditorías físicas automatizadas del estado real del bien.

- **Limitación de Tiempo y Mantenimiento:** El proyecto abarca el ciclo de desarrollo hasta la fase de pruebas y su despliegue inicial. El mantenimiento del servidor a largo plazo y la adición de futuros módulos no forman parte de esta fase de integración curricular.

1.8 Impacto Esperados

La implementación de este proyecto busca generar beneficios cuantificables para la institución, alineando los procesos administrativos con los estándares actuales de gestión digital.

1.8.1 Impacto Tecnológico

El proyecto permite la transición de una administración basada en archivos aislados hacia una arquitectura web centralizada. Adoptar este modelo garantiza la integridad de los datos mediante el uso de bases de datos relacionales, eliminando la redundancia y fragmentación de la información. Además, el stack tecnológico seleccionado facilita la escalabilidad, permitiendo la futura integración del sistema con otros departamentos de la EPESPO.

1.8.2 Impacto Social

La herramienta promueve una cultura organizacional enfocada en la rendición de cuentas. Al automatizar el control patrimonial, se reduce drásticamente la carga operativa que implica transcribir datos de forma manual. Esto permite que el personal administrativo optimice sus tiempos de respuesta y enfoque sus esfuerzos en labores de supervisión y análisis, consolidando un entorno de trabajo más eficiente y transparente.

1.8.3 Impacto Económico

Desde la perspectiva financiera, el impacto se manifiesta en la protección de los recursos existentes y la prevención de pérdidas materiales por falta de seguimiento. Al

mantener un registro exacto sobre el estado, ubicación y custodio de cada equipo, la institución puede planificar sus compras anuales con base en necesidades reales, evitando la duplicidad de adquisiciones y optimizando el presupuesto operativo.

Capítulo II

2. Marco introductorio

2.1Antecedentes Históricos

2.1.1 Historia de la Gestión de Inventarios

La gestión que se hace en los inventarios ha sido una práctica importante en las organizaciones desde el inicio del comercio y la producción. Con el tiempo, se ha desarrollado significativamente desde sistemas rudimentarios hasta sistemas informáticos complejos que le permiten controlar, planificar y optimizar el nivel de tiempo real.

2.1.1.1 Origen de la Gestión de Inventarios

Llevar un control en los inventarios despego desde las primeras civilizaciones que llevaban un ritmo de organización en su sistema social y político como la egipcia, la mesopotámica y la romana, donde se almacenaron bienes como alimentos, armas y materias primas. Existe registros que evidencian la contabilidad y resguardo de productos u objetos mediante tablillas, arcillas o papiros. En la Edad Media, la contabilidad y el control de bienes fueron procesados principalmente por monasterios, reinos y comerciantes. Sin embargo, el gran crecimiento comercial durante el Renacimiento y la Revolución Industrial fue la clave para el desarrollo de los sistemas de inventario.

A partir de los siglos XVIII y XIX las producciones que se hacían a nivel global trajo consigo el crecimiento y el desarrollo de las fábricas a las cuales debían crear nuevas formas de organización, por lo tanto, el control de inventarios se volvió mucho más estricta y evaluaba aspectos más críticos para poder mantener en el alza la producción y reducir al máximo los costos. (Erpag, 2019)

2.1.2.1 Origen y evolución de las Aplicaciones Webs

Todo comenzó en 1989 con Tim Berners-Lee, quien sentó las bases de la World Wide Web con tecnologías como HTML, HTTP y URL.

- Conocida como Web 1.0, los sitios eran principalmente estáticos. Imagina enciclopedias o catálogos en línea: el contenido era fijo, apenas cambiaba y el usuario solo podía leer y navegar a través de hipervínculos. (Mclibre, 2025)
- En la etapa conocida como Web 2.0 el usuario dejó de ser un simple espectador para convertirse en creador de contenido. Gracias a tecnologías como JavaScript, CSS, AJAX y lenguajes de backend como PHP y Python, las aplicaciones web se volvieron dinámicas.
- Web 3.0 busca que las aplicaciones web entiendan los datos de forma más inteligente. Utiliza inteligencia artificial y big data para ofrecer resultados más personalizados y relevantes para el usuario.
- En la etapa de Web 4.0 nos sumergimos totalmente en la conectividad integrándose la inteligencia artificial con el internet de las cosas (IoT). (Arias J. , 2024)

2.2 Antecedentes de la Investigación Relacionada

En el ámbito del desarrollo de aplicaciones web para la gestión de recursos, se tomaron como antecedentes referenciales investigaciones previas de la Universidad Laica Eloy Alfaro de Manabí (ULEAM), analizando específicamente el stack tecnológico implementado por sus autores para justificar las decisiones de arquitectura del presente proyecto.

- El trabajo de titulación “Desarrollo de Aplicación Web para la gestión procesos académicos y documentos del proceso de titulación” (Arteaga & Velasquez, 2025) , tuvo como objetivo agilizar la normativa documental. A nivel tecnológico, los autores basaron su desarrollo en una arquitectura tradicional monolítica utilizando PHP nativo y el motor de base de datos MySQL. A diferencia de dicho enfoque, la presente propuesta para la EPESPO implementa el framework Laravel, el cual ofrece una superioridad técnica en seguridad gracias a su sistema de enrutamiento protegido, el motor ORM Eloquent (que previene inyecciones SQL de forma nativa) y el uso de

PostgreSQL, que garantiza una integridad referencial mucho más estricta que MySQL para el manejo de inventarios.

- Por otro lado, la investigación "Sistema web para el control de inventarios en comercial Mendoza" (Pinargote & Medranda, 2024), facilitó el proceso de facturación y registro de productos. En su desarrollo, se empleó el framework CodeIgniter para el backend y Bootstrap puro para el frontend. Si bien es un modelo funcional, el presente proyecto para la EPESPO da un salto cualitativo al adoptar React JS en el frontend. La implementación de React permite crear una Single Page Application (SPA) donde los componentes (como tablas de búsqueda o filtros) se actualizan en tiempo real sin recargar la página, ofreciendo una experiencia de usuario (UX) mucho más fluida y escalable que las vistas estáticas de Bootstrap.
- Finalmente, el proyecto "Aplicación web para el control de inventario de la agropecuaria A.V. agro del cantón El Carmen" (Intriago, 2024), optimizó la administración de insumos. Este sistema fue desarrollado bajo el entorno de Python (Django). Aunque Django es altamente eficiente, la elección de Laravel (PHP) y React para la EPESPO se justifica por la amplia compatibilidad de PHP con los servidores institucionales existentes de la universidad, garantizando un despliegue más económico, un mantenimiento a largo plazo más accesible para los futuros técnicos de la facultad y una separación total (desacoplamiento) entre el cliente y el servidor mediante una API RESTful.

Síntesis Comparativa: El análisis de estos antecedentes confirma que, si bien existen múltiples vías para abordar la gestión de inventarios, la combinación tecnológica propuesta en esta tesis (React JS + Laravel + PostgreSQL) supera las limitaciones de arquitecturas tradicionales. Ofrece la robustez transaccional de un motor de datos de grado empresarial, la seguridad inquebrantable de un framework backend moderno y la

reactividad instantánea en la interfaz gráfica, asegurando que la EPESPO obtenga una herramienta de máxima eficiencia y escalabilidad.

2.3 Definiciones Conceptuales

2.3.1 Gestión de Inventarios

Según un artículo de (SAP, 2021) nos dice que la gestión de inventario es "es un conjunto de procedimientos y estrategias utilizados para asegurar que una empresa tenga la cantidad correcta de productos en el lugar correcto y en el momento preciso para cumplir con la demanda de los clientes, optimizando los costos de almacenamiento y evitando roturas de stock" a lo cual complementa (Hernandez, 2022) que nos que debe "permitir realizar un registro exitoso de las existencias de un almacén, esto incluye tantas entradas, monitoreo de su permanencia o la salida"

2.3.1.1 Tipos de Inventario

La gestión de indica para determinado ítem y procesos, con ello en mente se debe categorizar los tipos de inventarios como los señala el estratega de contenidos (Hearson, Oracle, 2024) de la empresa tecnológica estadounidense Oracle.

El inventario se puede clasificar en términos generales en tres categorías: materias primas/componentes, trabajo en proceso y productos terminados.

- **Materias primas/componentes**. Insumos básicos requeridos para la fabricación de bienes. En la industria, esto abarca materiales primarios y elementos preensamblados suministrados por terceros para su posterior integración en el producto final.

Según el sitio web oficial de (Hearson, Oracle, 2024) donde Hearson nos señala que hay:

- **Trabajo en curso (WIP)**. El inventario de WIP está en producción y no está listo para que el cliente lo adquiera. En nuestro ejemplo de fabricante de automóviles, este podría

ser un motor que se ha completado en una fábrica, pero necesita ser transportado a otro sitio para ser instalado en el vehículo.

- **Productos terminados**. En el proceso de un fabricante, esta es la etapa final del inventario, cuando ya está listo para la venta. Por ejemplo, los automóviles que salen de la línea de producción y se envían directamente a los concesionarios para que los clientes los compren. Las empresas no manufactureras, como los distribuidores mayoristas y minoristas, almacenan productos terminados para venderlos a los consumidores finales.

2.3.1.2 Ventajas de una Gestión de Inventarios

Una buena gestión de inventario permite a las empresas satisfacer los pedidos de los clientes con exactitud y rapidez, lo que a su vez mantiene a los clientes contentos. Además, ayuda a las compañías a reducir gastos y a mejorar su flujo de caja. Todos estos beneficios finalmente conducen a un aumento en las ganancias.

- **Control de costos.** Mantener el inventario disponible conlleva costos relacionados con el almacenamiento, la mano de obra, el transporte y mucho más. Estos costos relacionados con el inventario, o costos de mantenimiento, vinculan el efectivo que podría invertirse en otro lugar. La gestión del inventario ayuda a las empresas a mantenerse en el punto óptimo de almacenamiento: tener la cantidad correcta de existencias que se puede convertir rápidamente en ganancias, sin socavar y comprometer la producción o la entrega.
- **Eficiencia de la producción.** Mediante el seguimiento del inventario, los gerentes pueden anticipar el agotamiento de existencias de materias primas y componentes. A continuación, pueden cambiar la producción a otros artículos o trasladar al personal o al equipo a otro sitio de fabricación según sea necesario para optimizar la producción.

- **Reducción de riesgos.** Una sólida gestión del inventario ayuda a proteger a una empresa de los efectos de los problemas de la cadena de suministro, como los desastres naturales o artificiales y las interrupciones de los proveedores. (Hearson, Oracle, 2024)
- **Informes financieros.** Un control riguroso del inventario garantiza que el balance general refleje con exactitud el valor real de los activos institucionales, permitiendo auditorías precisas y una contabilidad transparente.
- **Ventaja competitiva.** La optimización de la rentabilidad, la eficiencia operativa y la calidad del servicio técnico consolidan el desempeño general de la organización, asegurando un uso racional de los recursos disponibles.. No es una casualidad. Las pequeñas ganancias marginales que se logran al pulir la rentabilidad, la eficiencia operativa y la calidad del servicio se van apilando como bloques de granito hasta construir una posición global mucho más sólida y difícil de derribar. (Hearson, Oracle, 2024)

2.3.2 Aplicaciones Webs

Según uno de las marcas más reconocidas a nivel mundial en escala tecnológica nos dice que es “Una aplicación web es un software que se ejecuta en el navegador web. Permiten acceder a funcionalidades complejas sin necesidad de instalar o configurar software.” (AWS Amazon, s.f.) A lo cual complementa (Peiro, 2019) que nos dice “Una página web es un documento accesible desde cualquier navegador con acceso a internet, y que puede incluir audio, vídeo, texto y sus diferentes combinaciones”.

2.3.2.1 Tipo de Aplicaciones Webs

Hoy en día, clasificar las aplicaciones web puede ser un desafío porque sus funciones están cada vez más entrelazadas, haciendo que sus límites sean menos claros. No obstante, aún es posible hacer varias diferencias entre las que se habitúa más a conocer al público pensado para esa funcionalidades o usuarios orientados en la medida de lo general.

Aplicaciones Web Estáticas. Las aplicaciones web estáticas son los sitios diseñados para presentar información a los visitantes sin permitir que interactúen con el contenido más allá de su lectura. Estos sitios son de los más simples que existen y generalmente están diseñados únicamente como datos HTML con algunas líneas de código CSS. (Franzolini, 2023)

Aplicaciones Web Dinámicas. En las webs apps dinámicas el usuario puede interactuar mucho más que con los sitios estáticos, puede registrarse para acceder a su cuenta, puede modificar parámetros o incluso publicar información. El ejemplo más claro de aplicación web dinámica sería un foro, asi lo indica en el blog (Axarnet, 2022)

2.3.3 Metodología de desarrollo

Según nos señala (Inforges, 2021) Scrum no se centra exclusivamente en el desarrollo web, destacando ventajas generales como la adaptabilidad, la mejora continua, la mayor productividad y la satisfacción del cliente, todas ellas altamente relevantes para proyectos de aplicaciones web donde los requisitos pueden evolucionar.

Entonces en una universidad, las necesidades de un sistema informático pueden cambiar a medida que surgen nuevos requisitos. Aquí es donde Scrum demuestra su valor, ya que su enfoque iterativo permite una fácil adaptación a estos cambios.

Si los usuarios solicitan un nuevo módulo durante el desarrollo (como la asignación de responsables o reportes personalizados), este se puede incorporar sin problemas en el siguiente Sprint, sin interrumpir el progreso ya logrado.

2.3.3.1 Roles

- **Producto Owner.** Esta es la persona a cargo de la lista de trabajo pendiente del producto y está conectado a las necesidades del usuario y se centra en transmitir el punto de vista del usuario a su equipo y a otros ejecutivos involucrados. Los buenos encargados de productos aportan claridad sobre qué es lo siguiente que se debe entregar

debido a su importancia. En última instancia, deberían ser ellos quienes decidan cuándo algo está listo para ser entregado (con una tendencia a realizar entregas con frecuencia)

- **Scrum Master.** Es la persona que dirige los distintos eventos de Scrum. Considéralo el gerente del proyecto y facilitador de Scrum. El Scrum Master debe promover las reuniones diarias de actualización y organizar las reuniones de planificación, revisión y análisis retrospectivo del sprint. (Pachon, 2024)
- **Equipo Scrum.** son todos los que están trabajando en el sprint. Los miembros del equipo deben auto-organizarse y ser colaborativos para lograr el objetivo de Scrum de mejora continua. (Martins, 2025)

2.3.4 Herramientas Tecnológicas

2.3.4.1 Lenguajes de Programación o herramientas para la estructura de la aplicación:

- **HTML5:** Este es el lenguaje fundamental para estructurar las páginas web. Se usará para definir la base de cada parte del sistema, como los formularios para registrar artículos, las tablas que muestran los bienes o los menús de navegación. Su diseño mejora la accesibilidad, ayuda al posicionamiento en buscadores y facilita que el sistema funcione bien con otras herramientas.
- **CSS3:** Con CSS3, le daremos estilo visual a todo lo que se estructura con HTML. La configuración de la identidad visual y la maquetación del sistema se articula mediante hojas de estilo en cascada, las cuales determinan atributos estéticos críticos como la tipografía, la paleta cromática y la jerarquía espacial de los elementos. Esta definición técnica resulta indispensable para asegurar la adaptabilidad del diseño a distintas resoluciones de pantalla funcionalidad conocida como diseño responsivo, garantizando así una interfaz gráfica profesional y funcional independientemente del dispositivo de acceso utilizado.

- **JavaScript:** Es el lenguaje fundamental para la interactividad en el navegador. Su capacidad de operar del lado del cliente permite que la aplicación procese validaciones de formularios y despliegue menús de forma asíncrona, optimizando los tiempos de interacción del usuario.
- **PHP:** Actúa como el motor lógico en el lado del servidor. Se encarga del procesamiento de reglas de negocio, la validación de autenticación de usuarios y la gestión del flujo de datos hacia la base de datos relacional.
- **SQL:** Lenguaje estándar utilizado para la gestión y consulta de la base de datos relacional. Permite estructurar consultas precisas para la generación de reportes e indicadores de inventario.
- **TypeScript:** Constituyen lenguajes y superconjuntos evaluados durante la fase de investigación tecnológica. Si bien ofrecen ventajas competitivas en tipado estricto, análisis de datos y desarrollo móvil multiplataforma, se determinó que el ecosistema PHP/JavaScript se alinea de mejor manera con la infraestructura de servidores web tradicionales de la institución.
- **Python:** Una herramienta que no grita pero que resuelve con una elegancia casi insultante, se siente como pasar de intentar entender señales de humo en una tormenta a recibir un mensaje de texto nítido y directo al grano. Como calve hay que reducir la carga cognitiva para entender la aplicación web de la Escuela de Pesca asegurando la longevidad y permitiendo ciertas modificaciones de forma futura dando un trámite de forma ágil en vez de una sesión digital errónea.
- **Java:** Es el estándar que nos puede prometer oro para infraestructuras críticas donde la seguridad y la escalabilidad no son negociables. No se trata solo de escribir clases y métodos, el verdadero reto está en entender cómo optimizar lo que ya tenemos, por ejemplo, muchos desarrolladores abusan de la creación de objetos dentro de bucles

intensivos, confiando ciegamente en que el Garbage Collector (GC) hará magia. Si bien es cierto que G1 han evolucionado muchísimo, el costo de asignación y la presión sobre el Heap siguen siendo factores que pueden tumbar el rendimiento de una aplicación en producción.

- **Kotlin:** Es el compañero perfecto de Java que elimina la verborrea innecesaria del código y añade capas de seguridad contra errores nulos. Al final, esta mezcla de herramientas no busca otra cosa que crear un software que sea, a la vez, una roca en estabilidad y una pluma en agilidad.
- **Dart:** Desarrollado por Google como el lenguaje base del framework Flutter, Dart está diseñado específicamente para la construcción de interfaces gráficas reactivas. Su capacidad de compilación anticipada (AOT) permite generar aplicaciones nativas multiplataforma de alto rendimiento a partir de una única base de código, optimizando los tiempos de respuesta y la fluidez de la interfaz en diversos dispositivos.

2.3.4.2 Frameworks

- React JS: Librería de JavaScript para la construcción de interfaces de usuario (UI). Su capacidad para renderizar componentes dinámicos mediante el Virtual DOM permite la creación de Single Page Applications (SPA) fluidas y escalables.
- Laravel: Framework de PHP basado en el patrón Modelo-Vista-Controlador (MVC). Su implementación en el proyecto garantiza el enrutamiento seguro, la protección contra vulnerabilidades comunes (como inyecciones SQL) mediante su ORM Eloquent, y la gestión de APIs RESTful.
- Bootstrap: Framework CSS que asegura la adaptabilidad visual (diseño responsivo) de la aplicación en múltiples resoluciones y dispositivos.

2.3.4.3 Base de Datos

- **PostgreSQL:** Sistema de gestión de bases de datos relacional orientado a objetos. Su estricto cumplimiento de las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) asegura la integridad referencial y el control de concurrencia requeridos para el manejo de inventarios institucionales. Su robustez en el manejo de la integridad referencial y su capacidad para procesar transacciones concurrentes lo convierten en la opción óptima para administrar datos institucionales críticos. MongoDB: Es una base de datos NoSQL orientada a documentos (formato BSON/JSON). Su esquema flexible resulta ideal para aplicaciones que manejan datos no estructurados o que requieren alta escalabilidad horizontal, siendo una alternativa eficiente cuando la rigidez estructural no es un requerimiento imperativo.
- **Oracle Database:** Constituye un sistema de gestión de bases de datos de nivel empresarial, diseñado para procesar grandes volúmenes de información con alta disponibilidad. Aunque requiere licenciamiento comercial, ofrece estándares avanzados de seguridad y concurrencia.
- **MongoDB:** Es una base de datos NoSQL orientada a documentos (formato BSON/JSON). Su esquema flexible resulta ideal para aplicaciones que manejan datos no estructurados o que requieren alta escalabilidad horizontal. Es flexible por naturaleza. Resulta ideal para aplicaciones modernas como redes sociales, aplicaciones de mensajería o plataformas con perfiles personalizados donde la rigidez de las tablas tradicionales se siente como una camisa de fuerza. Bajo mi perspectiva, usar MongoDB es como tener un archivador elástico que se expande y se deforma según lo que necesites guardar, sin quejarte por la falta de una columna predefinida.
- **Firestore:** Se presenta como la joya de la corona dentro de la plataforma Firebase, estructurando la información mediante un modelo de documentos organizados en

colecciones que derrocha agilidad. Su arquitectura es pura vanguardia. Habilita una sincronización bidireccional automática entre el cliente y el servidor, una funcionalidad técnica que es determinante para el despliegue de aplicaciones web y móviles que requieren actualizaciones de estado en tiempo real sin que el usuario tenga que mover un dedo para refrescar la pantalla. Es una persistencia de datos transparente.

2.3.4.4 Entorno de Desarrrollo

- **Visual Studio Code:** Editor de código fuente multiplataforma utilizado para la programación del sistema, optimizado mediante extensiones para el desarrollo en React y PHP.
- **XAMPP:** Entorno de desarrollo local que integra Apache, MySQL y PHP, permitiendo la simulación del servidor web durante la fase de codificación y pruebas unitarias.
- **Git y GitHub:** Sistemas de control de versiones y alojamiento en la nube, esenciales para gestionar el historial de cambios del código fuente y mantener un respaldo centralizado del repositorio.
- **Eclipse:** Gracias a una arquitectura de piezas intercambiables que se adapta a lo que el programador necesite. Es una herramienta con historia.
- **PyCharm:** Si el lenguaje de elección es Python, PyCharm es la mejor opción o traje a medida; no es solo un editor, es un copiloto o algún guía que te ayuda a construir desde simples scripts hasta complejas arquitecturas de ciencia de datos con una fluidez y rapidez que se agradece cuando el tiempo apremia. Es, subjetivamente, el entorno donde el código Python fluye mejor y la mejor opción como tal.
- **Postman:** Plataforma utilizada para el diseño, prueba y documentación de las APIs REST, permitiendo validar estructuralmente las peticiones y respuestas HTTP del backend**.**

2.3.4.4 Servicios Webs

- **Hosting Web:** Contaremos con un servicio de alojamiento web para publicar la aplicación y que sea accesible desde cualquier lugar con conexión a internet. Durante las fases de prueba, podríamos usar un servidor de la institución o uno gratuito como 000webhost. Este servicio nos proporciona el espacio y los recursos necesarios en un servidor para que nuestra aplicación PHP y su base de datos funcionen correctamente.
- **APIs RESTful y JSON:** La arquitectura del sistema emplea servicios web RESTful expuestos a través de Laravel, garantizando el desacoplamiento técnico entre el cliente y el servidor. El formato JSON (JavaScript Object Notation) se establece como el estándar de intercambio de datos, asegurando una transmisión ligera y estructurada de la información
- **JSON como estándar de intercambio:** JSON se ha erigido como el lenguaje universal de intercambio en esta arquitectura, funcionando como una suerte de esperanto digital que todo componente entiende a la primera. Es ligero. Es directo. Al adoptar este formato normativo para la transmisión de datos, el sistema de la Escuela de Pesca garantiza una interoperabilidad sin fisuras entre lenguajes de programación que, de otro modo, se mirarían con recelo, asegurando que lo que sale del servidor sea perfectamente digerible para cualquier cliente que lo reciba.
- **Protocolos de autenticación (/JWT):** Cuando la lógica de acceso que divide entre lo que se permite y lo que no se limitan los recursos de la institución basándose en los roles que son asignados a cada usuario en términos de jerarquía resulta interesante y también igual de necesario, así nos hace pensar como una parte de código puede decidir quien tiene el poder modificar un activo y quienes pueden limitarse o se restringe a solo observar cuando lo trazable de los activos sino dentro de una consecuencia que tiene que ser inevitable dentro de un diseño en lo correctamente ejecutado.

- **Dominio:** Aunque no es estrictamente necesario, usar un nombre de dominio nos permitiría acceder al sistema a través de una dirección web personalizada. Esto no solo facilita que la gente lo encuentre y lo comparta, sino que también mejora la imagen y le da un toque más profesional al sistema, especialmente si en el futuro se planea su lanzamiento oficial.

2.4 Conclusiones de las Investigaciones

La evaluación y selección de los instrumentos metodológicos y tecnológicos fundamentan la viabilidad operativa del sistema web propuesto. La evaluación de los instrumentos metodológicos y tecnológicos fundamenta la viabilidad operativa del sistema. La selección de la arquitectura Cliente-Servidor (React JS para el frontend y Laravel para el backend) permite separar la lógica de negocio de la interfaz. Para la persistencia de los datos, se seleccionó PostgreSQL como motor principal debido a su estricto cumplimiento de las propiedades ACID, su robusto manejo de la integridad referencial y su capacidad superior para gestionar transacciones concurrentes complejas. Estas características resultan críticas para evitar la duplicidad y asegurar la trazabilidad del control patrimonial dentro de la unidad académica. Esta configuración técnica y procedimental asegura la alineación con los objetivos trazados, optimizando de manera directa la trazabilidad, la eficiencia administrativa y los mecanismos de fiscalización de inventarios dentro de la unidad académica.

Capítulo III

3. Marco investigativo

3.1 Tipo de Investigación

La investigación adoptará un enfoque mixto (cualitativo y cuantitativo) con un alcance exploratorio y descriptivo. Según (Roberto Hernández Sampieri, 2014), el enfoque mixto representa un conjunto de procesos sistemáticos, empíricos y críticos de investigación que implican la recolección y el análisis de datos cuantitativos y cualitativos. El alcance exploratorio permitirá comprender la situación actual de la gestión de inventarios en la EPESPO, identificando las causas raíz de las deficiencias administrativas. Por su parte, el alcance descriptivo facilitará la caracterización del entorno institucional y la definición de los procesos implicados.

La justificación de este enfoque mixto radica en la naturaleza del desarrollo de software: la vertiente cualitativa se aplicará en la fase de diagnóstico (levantamiento de requerimientos mediante entrevistas a expertos), mientras que la vertiente cuantitativa se ejecutará en la fase de evaluación tecnológica, donde se medirán y validarán métricas numéricas exactas, tales como los tiempos de respuesta del servidor web (< 2 segundos) y el rendimiento de la base de datos durante las pruebas de estrés.

3.2 Métodos de Investigación

Para cumplir con el enfoque mixto del proyecto, se aplicará un conjunto de métodos estructurados que abarcarán tanto la comprensión del problema como la validación de la solución tecnológica:

- **En la fase cualitativa (Diagnóstico):** Se examinarán las dinámicas operativas y las percepciones de los usuarios encargados de la custodia patrimonial. Esto proporcionará la base empírica necesaria para definir los requisitos funcionales que guiarán el diseño de la arquitectura del software.

- **En la fase cuantitativa (Validación):** Se empleará el método técnico-experimental para la evaluación del sistema en entornos controlados. A través de pruebas de caja negra e integridad de datos, se recolectarán resultados medibles sobre la capacidad de procesamiento de PostgreSQL y la eficiencia de las peticiones HTTP gestionadas por Laravel.

3.2.1 Método Teóricos

3.2.1.1 Método Bibliográfico

La fase de investigación documental comprende el análisis crítico de fuentes primarias y secundarias, abarcando desde artículos científicos y tesis antecedentes hasta la normativa institucional vigente. Esta revisión bibliográfica estructura el marco teórico del proyecto, proporcionando el sustento conceptual necesario para integrar los principios de la gestión de inventarios con la arquitectura de aplicaciones web y las metodologías de desarrollo ágil.

3.2.1.2 Método Deductivo

Permite partir de principios generales documentados en la literatura para llegar a conclusiones específicas sobre la problemática en la EPESPO. A través de este método se fundamenta la necesidad de modernizar el sistema de inventarios mediante una solución digital.

3.2.1.3 Método Inductivo

Este método posibilita, a partir del análisis de situaciones particulares observadas en la institución, generar hipótesis sobre los factores que afectan la eficiencia del control de inventarios y establecer propuestas de solución tecnológica.

3.2.1.4 Método Analítico

Como señala (Mario Tamayo, 2003), este método consiste en la desmembración de un todo para observar las causas de un fenómeno. Bajo este principio, se aplicó el método analítico para descomponer y examinar detalladamente los procesos manuales de inventario actuales en

la EPESPO, identificando fallas operativas específicas como la duplicidad de datos y la pérdida de trazabilidad.

3.2.1.5 Método Sintético

Complementario al análisis, el método sintético permitió integrar los hallazgos y requerimientos aislados para diseñar la arquitectura lógica de la nueva aplicación web. Se unificaron las necesidades de los distintos departamentos para estructurar una solución tecnológica centralizada bajo el patrón MVC.

3.3 Estrategia Operacional para la Recolección de Datos

3.3.1 Población y Muestra

La población objetivo está conformada estrictamente por 3 funcionarios directamente vinculados a la administración patrimonial de la EPESPO: el Director Administrativo y dos responsables operativos de inventario. Dado que el tamaño poblacional es reducido y completamente accesible (menor a 10 individuos), no se aplicará un muestreo estadístico, sino que se trabajará con un Censo Poblacional, abarcando completamente al personal con capacidad de decisión y operación sobre los activos de la institución.

3.3.2 Herramientas de Recolección de Datos

Para el levantamiento preciso de los requerimientos funcionales en la EPESPO, es imperativo contar con un procesamiento automatizado de los datos en tiempo real. Esto requiere herramientas de recolección de datos que actúen con alta precisión administrativa. Al contrastar los registros físicos acumulados con la normativa vigente, esta revisión documental permite una validación empírica que evidencia las inconsistencias estructurales del sistema actual, exponiendo la divergencia real entre los documentos físicos y el inventario existente en los almacenes.

Asimismo, el proyecto integra un análisis comparativo de soluciones tecnológicas implementadas por otras instituciones educativas, lo cual permite identificar metodologías

estandarizadas para optimizar la logística del control patrimonial. Resulta ineficiente desarrollar metodologías empíricas cuando existen arquitecturas de software modernas y comprobadas. Al examinar arquitecturas similares en entornos de formación técnica, se logran identificar las mejores prácticas de la industria, asegurando que la aplicación web se construya sobre estándares de ingeniería de software comprobados y no sobre supuestos operativos.

3.4 Plan de Recolección de Datos

Para el levantamiento de los requerimientos funcionales y justificar la pluralidad del enfoque, se aplicaron técnicas formales de recolección de datos. Como señala (Arias F. G., 2012), las técnicas representan el conjunto de procedimientos para obtener los datos, mientras que los instrumentos son los recursos materiales. Bajo esta premisa, se definieron los siguientes:

- **Técnica 1: Entrevista.** Dirigida al nivel directivo y operativo para definir las necesidades funcionales y no funcionales del sistema.
    - **Instrumento 1:** Guía de entrevista semiestructurada.
- **Técnica 2: Análisis Documental.** Revisión técnica de las hojas de cálculo y actas físicas actuales.
    - **Instrumento 2:** Ficha de registro y observación.

Validación: Todos los instrumentos fueron sometidos a un proceso de validación mediante Juicio de Expertos, contando con la revisión y aprobación directa del Director Administrativo de la EPESPO, lo que garantiza la viabilidad técnica del levantamiento de información.

3.5 Análisis de los Datos

3.5.1 Preparación de los Datos

El tratamiento de la información recabada se someterá a un proceso de filtrado y síntesis, orientado a la supresión de elementos reiterativos. Posteriormente, los datos

resultantes se estructurarán mediante un esquema de clasificación basado en categorías temáticas pertinentes, facilitando así su análisis interpretativo.

3.5.2 Análisis de Correlación

Se establecerán relaciones entre los distintos elementos del proceso de inventario, tales como tipo de bien, responsable, ubicación y frecuencia de uso.

3.5.3 Análisis Cualitativo

Las respuestas de tipología abierta se someterán a una clasificación basada en ejes temáticos convergentes. Este procedimiento analítico tiene como finalidad la identificación de patrones estructurales y la delimitación de aquellas necesidades operativas que se manifiesten de forma recurrente entre los participantes.

3.5.4 Interpretación de Resultados

La interpretación de los resultados analíticos fundamentará el dictamen sobre la factibilidad técnica de la propuesta. Esta validación permitirá acotar el alcance funcional del sistema, asegurando una alineación precisa entre las capacidades operativas del desarrollo y la resolución efectiva de las deficiencias diagnosticadas.

3.6 Análisis de los Resultados

La sistematización de los hallazgos obtenidos en las entrevistas tiene como función primaria la caracterización de los obstáculos operativos inherentes al modelo de gestión actual, validando, de manera simultánea, las especificaciones funcionales críticas del sistema. Este diagnóstico empírico constituye el fundamento para la definición de los protocolos de diseño, desarrollo y pruebas de la aplicación web, alineando la arquitectura del software con los objetivos de optimización en el control, la trazabilidad y la eficiencia administrativa de los activos institucionales.

3.6.1. Entrevista al responsable del Inventario de Epespo

Cargo: Responsable de Inventario

Área: Administración y Control de Bienes EPESPO

Fecha: 28 de octubre del 2025

Modalidad: Presencial

Eje Temático: Proceso actual de control de inventario

¿Cómo se lleva actualmente el control de los bienes en la institución?

Actualmente el control se realiza mediante un auxiliar con hojas internas de cálculo en Excel y registros físicos, donde se encuentran todos los bienes con códigos, responsables y áreas, a todo este sistema se le llama auxiliar, pero es en realidad hojas de Excel.

Eje Temático: Trazabilidad y asignación

¿Qué protocolo se sigue para vincular cada bien o equipo con su respectivo usuario o responsable de custodia?

De manera física, cada bien que entre al sistema se le actualiza en los registros, para dar de baja o en la entrada de bienes, todo de manera manual y física con sus respectivas actas.

Eje Temático: Problemas y dificultades

¿Cuáles son los principales problemas u obstáculo que enfrenta el modelo de gestión de inventario vigente?

Que se pierdan o de alguna manera se dañen los registros de todos los bienes los cuales están registrados junto con sus actas.

Eje Temático: Documentación utilizada

¿Qué documentos o formatos estandarizados se utiliza para respaldar el registro de los movimientos de bienes y traslados de activos?

De manera interna se tienen las actas de entregas y la aceptación de los bienes donde se consta la responsabilidad de la persona que está a cargo de ese bien.

Eje Temático: Frecuencia de actualización

¿Cada cuánto se actualiza el inventario general?

Se actualiza cada que sea necesario, es decir cada que se compre o se adquiera un bien, de modo que no hay un tiempo estimado exacto.

Eje Temático: Uso de tecnología

¿Se emplea algún sistema o software para este control?

No. Solo auxiliares y en el registro del sistema contable que se maneja aquí, pero no existe un sistema específico destinado a lo bienes o como tal inventario.

Eje Temático: Percepción sobre la nueva herramienta

¿Cree que una plataforma web en la optimización de los procesos de control y administración de inventarios ayudara a la institución?

Sí, mucho, siempre es bueno implementar un sistema destinado a un solo rubro y de esta manera manejar de forma más profesional ciertos módulos y así mejorar la eficiencia.

Eje Temático: Requerimientos esperados

¿Qué especificaciones funcionales o módulos operativos identifica como requisitos imperativos para la configuración de la nueva plataforma tecnológica?

Que arroje las actas de entregas, de bajas, eso sería lo primordial y como siguiente parámetro que me arroje un detalle de todos los bienes con sus responsables y áreas o departamentos donde se encuentran todos los bienes de la empresa.

3.6.1.1 Conclusiones de la entrevista

Después de llevar a cabo la entrevista con el encargado del inventario de la Escuela de Pesca del Pacífico Oriental (EPESPO), quedó claro que el control de bienes se realiza de forma manual, utilizando hojas de cálculo y registros en papel. Esto ha generado problemas en la actualización de la información, duplicación de datos y pérdida de seguimiento sobre la ubicación y la persona responsable de cada activo. La entrevistada comentó que no hay un

sistema centralizado que permita acceder rápidamente a los datos del inventario ni generar reportes automáticamente, lo que retrasa los procesos administrativos y de auditoría. Entonces la información recopilada ayudó a identificar los principales problemas del proceso actual y a establecer los requerimientos funcionales más importantes del sistema, como la automatización del registro, el seguimiento de movimientos y la implementación de alertas o reportes automáticos. Esta entrevista fue fundamental para entender las necesidades operativas del área responsable y guiar el desarrollo del sistema hacia una solución práctica y eficiente.

3.6.2. Entrevista al director Administrativo de Epespo

Cargo: director Administrativo / Coordinador General EPESPO

Área: Dirección Académico–Administrativa

Fecha: 28 de octubre del 2025

Modalidad: Presencial

Eje Temático: Supervisión general del inventario

¿Cómo supervisa actualmente la gestión del inventario institucional?

Se revisan informes manuales enviados por el responsable de inventario. No hay control en tiempo real ni estadísticas consolidadas.

Eje Temático: Toma de decisiones

¿Qué datos o indicadores métricos requiere para fundamentar la toma de decisiones estratégicas sobre el control y administración de la institución?

Datos actualizados sobre bienes asignados, estado de los equipos, bienes en reparación y reportes de auditoría.

Eje Temático: Problemas detectados

¿Qué problemas operativos o procedimentales identifica en la ejecución de los mecanismos de control y monitoreo vigentes?

Hay duplicidad de datos, pérdida de documentos y lentitud en la generación de reportes. Además, falta seguimiento en tiempo real.

Eje Temático: Auditorías y reportes

¿Cuál es el procedimiento técnico aplicado para la emisión de los informes y reportes requeridos por los organismos de control interno y auditoría?

Manualmente, consolidando datos de Excel. Es un proceso que tarda varios días.

Eje Temático: Visión sobre la automatización

¿Cuál es su opinión sobre la implementación de una aplicación web para el control de inventario?

Considero que es necesaria. Un sistema automatizado mejoraría la eficiencia, la trazabilidad y la transparencia en el manejo de bienes.

Eje Temático: Seguridad y acceso

¿Qué estándares de seguridad y mecanismos de restricción de datos estima necesarios para salvaguardar la información sensible procesada por la aplicación web?

Que los usuarios tengan perfiles diferentes y que el sistema registre todas las modificaciones para evitar errores o pérdidas.

Eje Temático: Beneficios esperados

¿Qué beneficios espera tener para la implementación del sistema dentro de la institución?

Reducción del tiempo en actualizaciones, generación inmediata de reportes, control claro de bienes, y transparencia institucional.

Eje Temático: Compromiso institucional

¿Existe apertura por parte de la directiva dentro de la institucion para implementar una herramienta digital?

Sí, siempre que facilite el trabajo del personal y mejore los procesos administrativos

3.6.2.1 Conclusiones de la entrevista

El análisis de la entrevista con la Dirección Administrativa evidenció vulnerabilidades en el modelo de gestión actual. La ausencia de un sistema automatizado ralentiza la supervisión y generación de informes, incrementando el margen de error operativo. Se determinó como prioridad institucional el desarrollo de una plataforma con control de acceso basado en roles y capacidad de generación automatizada de actas para garantizar la trazabilidad administrativa.

3.7 Conclusión del Marco Investigativo

El diseño metodológico permitió diagnosticar con precisión las deficiencias estructurales en el control de activos de la EPESPO. La articulación de los métodos descriptivo y analítico identificó los factores críticos que comprometen la integridad de los datos en los procesos manuales vigentes. Estos hallazgos fundamentan la viabilidad técnica del proyecto y definen los requerimientos funcionales para la arquitectura de software propuesta.

Capítulo IV

4. Marco Propositivo

4.1 Descripción de la Propuesta

La presente propuesta tecnológica consiste en la implementación del **"Sistema de Gestión de Inventarios EPESPO"**, una solución web integral diseñada para optimizar, automatizar y asegurar el control de los bienes institucionales de la Escuela de Pesca del Pacífico Oriental. El sistema sustituye los procedimientos manuales y el uso de hojas de cálculo dispersas por una plataforma centralizada que garantiza la integridad, disponibilidad y confidencialidad de la información.

La arquitectura funcional del proyecto abarca el ciclo de vida completo de cada activo. La propuesta estructura la administración a través de un flujo operativo continuo, donde cada movimiento registral queda almacenado de forma permanente en la base de datos, garantizando la trazabilidad exacta de la ubicación y el custodio de cada equipo. La propuesta busca estructurar la administración a través de un flujo operativo continuo, donde cada movimiento registral quede almacenado de forma permanente en la base de datos, eliminando esa incertidumbre tan común de no saber en qué oficina terminó una laptop o quién dañó un proyector. La seguridad no es una sugerencia, es un muro. Adicionalmente, el sistema implementa un control de acceso basado en roles, separando los privilegios de los usuarios que administran el inventario y los lectores, garantizando que la manipulación de la información sensible sea exclusiva del personal debidamente autorizado.

4.1.1 Diseño de la Arquitectura

La arquitectura del sistema se estructuró bajo el patrón Modelo-Vista-Controlador (MVC), implementado a través de un modelo Cliente-Servidor desacoplado. Esta decisión técnica separa la lógica de negocio de la interfaz gráfica, permitiendo que ambos entornos se desarrollen y escalen de forma independiente. La comunicación entre estas capas se realiza estrictamente mediante protocolos HTTP y el intercambio de datos en formato JSON.

En la capa de presentación (Vista), el entorno frontend desarrollado con React procesa las interfaces directamente en el navegador. Al consumir los servicios API de forma asíncrona, el sistema actualiza los contenidos en pantalla sin necesidad de recargar la página completa, optimizando los tiempos de interacción del operador.

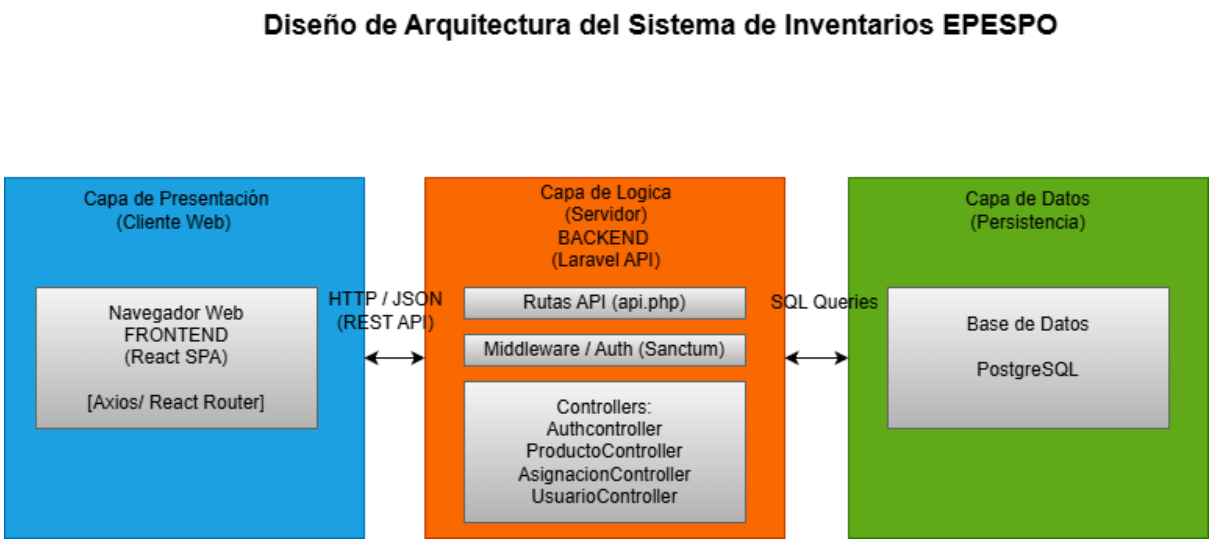

Figura 2. Diseño de la Arquitectura de Inventario EPESPO

4.1.2 Backend

El Backend constituye el motor lógico del sistema y ha sido desarrollado utilizando Laravel, un framework de PHP robusto y seguro. Su función principal es exponer una serie de rutas protegidas (API Routes) que permiten al Frontend interactuar con la base de datos PostgreSQL de manera segura.

Se presentan los componentes fundamentales del *Backend*, reescritos con el rigor técnico y la estructura solicitada:

- **Sistema de Enrutamiento:** La definición de rutas en el archivo api.php estructura los endpoints disponibles, estableciendo los métodos HTTP permitidos para cada interacción del cliente con el servidor.

- **Seguridad y Autenticación:** La protección de la plataforma se instrumenta mediante Laravel Sanctum, el cual gestiona la emisión y validación de tokens de acceso criptográficos. Este mecanismo verifica estrictamente la identidad y los privilegios de cada petición entrante.

- **Validación de Datos:** La validación de datos en este desarrollo no es un simple trámite burocrático, sino una barrera de contención robusta donde se emplean clases de solicitud personalizadas, o Form Requests, para blindar la integridad de la información desde el primer contacto. Para garantizar la persistencia e integridad referencial, se implementó un módulo de validación mediante clases Form Requests. Estos filtros verifican que cada solicitud cumpla con las reglas de formato, longitud y unicidad antes de interactuar con la base de datos.

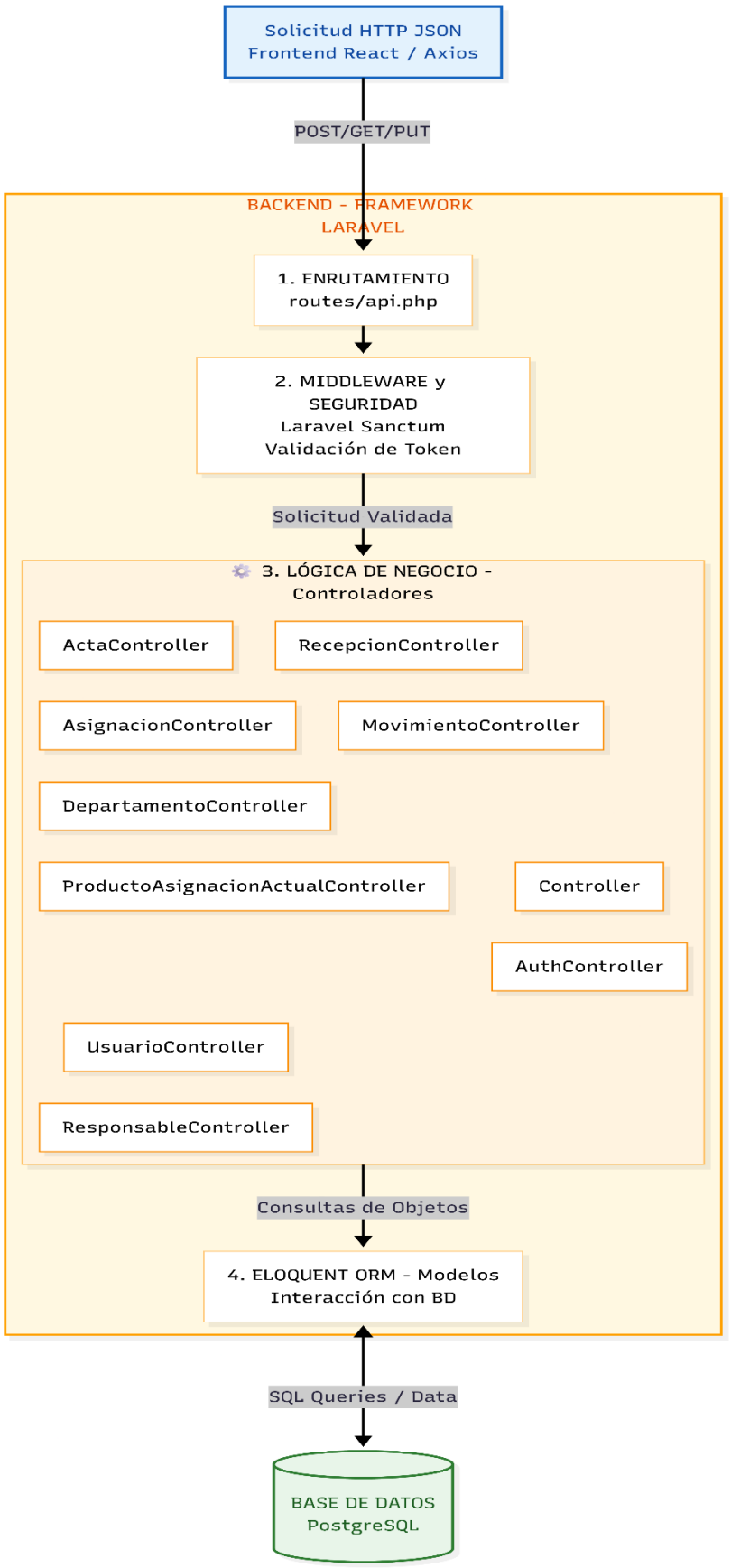

Figura 4. Arquitectura del Backend

4.1.3 Servicio Web (API REST)

El sistema implementa una arquitectura de microservicios simulada a través de controladores API en Laravel, donde cada clase gestiona una entidad específica del negocio. Estos servicios exponen endpoints que son consumidos por el cliente (Frontend) mediante peticiones HTTP asíncronas. La arquitectura del backend se organiza mediante controladores que gestionan las reglas de negocio y las transacciones con la base de datos:

Controlador de Autenticación (AuthController): La seguridad no se ha dejado al azar ni a la buena voluntad de los operadores, sino que se ha instrumentado mediante Laravel Sanctum, una herramienta que emite tokens digitales como si fueran pasaportes diplomáticos de alta seguridad.

Controlador de Inventario (ProductoController): Permite realizar las operaciones de registro, consulta, edición y eliminación de los bienes institucionales. A través de este controlador se administran también los catálogos auxiliares, permitiendo clasificar los bienes por su Categoría y asignarles su Ubicación o departamento correspondiente dentro de la institución.

Controlador de Departamentos (DepartamentoController): Administra el catálogo de ubicaciones físicas e instalaciones de la institución, estructurando la distribución organizacional necesaria para la geolocalización de los activos.

Controlador de Asignaciones (AsignacionController): Gestiona la lógica transaccional para la entrega de bienes, vinculando de manera relacional cada activo con su respectivo funcionario custodio, garantizando la trazabilidad histórica del movimiento.

Controlador de Usuarios (UsuarioController): Este módulo se le otorga el poder de gestionar las cuentas de todos los integrantes de la plataforma con una autoridad absoluta. Es una vigilancia total. Mediante la asignación de roles específicos, se garantiza que cada

individuo posea el nivel de acceso preciso para sus funciones, evitando que manos inexpertas terminen hurgando en configuraciones que no les corresponden, algo así como impedir que un marinero de cubierta intente calibrar el radar de navegación sin tener el rango necesario.

4.2 Determinación de Recursos

La factibilidad operativa del proyecto "Sistema de Gestión de Inventarios EPESPO" se sustenta en una evaluación pormenorizada de los requerimientos de implementación. Este análisis de recursos dimensiona los insumos técnicos y humanos indispensables para la ejecución del desarrollo, estableciendo las bases materiales para la materialización de la propuesta.

4.2.1 Humanos

El equipo de trabajo está conformado por el personal encargado del desarrollo, investigación y validación del software:

Rol / Cargo	Responsable(s)	Funciones Principales
Investigadores y Desarrolladores Full-Stack	1. Jerick Abraham Bailon Quijije 2. Devis Yonaikel Alonzo Pico	• Levantamiento de requerimientos e investigación. • Diseño de la arquitectura y Base de Datos. • Programación del Backend (Laravel) y Frontend (React). • Documentación de la tesis y manuales. • Despliegue y pruebas del sistema.

Tutor Académico	Patricia Alexandra Quiroz Palma	• Guía metodológica y revisión del avance del proyecto. • Aprobación de entregables.
Validadores (Stakeholders)	•Director Administrativo EPESPO • Responsable de Inventario	• Provisión de información sobre procesos actuales. • Validación funcional del software (Pruebas de aceptación).

Tabla 1. Recursos Humanos

4.2.2 Tecnológicos

Se requieren recursos de hardware y software específicos para el desarrollo y la posterior implementación del sistema:

Categoría	Recurso / Herramienta	Descripción y Uso en el Proyecto
Hardware	**Equipos de Cómputo (Laptops)**	2 computadoras portátiles (Procesador Core i5 y Ryzen 5, 16GB RAM, SSD) utilizadas por los autores para la programación y virtualización.
Hardware	**Servidor para Pruebas**	Infraestructura local y base de datos y la aplicación web.
Software	**Sistema Operativo**	Windows 10 y 11 (Entorno de Desarrollo y Prouduccion).

Backend	**Laravel (PHP)**	Framework que se usó para la creación de la API REST, segurida y lógica de negocio.
Frontend	**React JS (JavaScript)**	Librería para la construcción de las interfaces de usuario dinámicas (SPA).
Base de Datos	**PostgreSQL**	Sistema de gestión de base de datos relacional para el almacenamiento de la información.
Herramienta	**Visual Studio Code**	Editor de código con extensiones para PHP y JS.
Herramientas	**Git / GitHub**	Para el trabajo colaborativo entre los autores.
Herramientas	**Postman**	Herramienta para realizar pruebas unitarias a los endpoints de la API.
Diseño	**Draw.io** **Mermaid Live** **Dbdiagram.io**	Herramientas para el modelado de base de datos (MER) y diagramas de arquitectura.

Tabla 2. Recursos Tecnológicos

4.2.2 Económicos

Aunque el proyecto es de carácter académico, se estima un presupuesto referencial que cuantifica la inversión en horas de trabajo, uso de equipos y servicios.

Rubro	**Descripción**	**Costo Unitario ($)**	**Total, Estimado ($)**
Recursos Humanos	Horas de desarrollo e investigación (estimado 300 horas x $5/h)	$1,500.00	$1,500.00

Hardware	Depreciación de equipos de cómputo (uso por 6 meses)	$150.00	$300.00
Servicios	Internet y energía eléctrica (6 meses)	$40.00	$240.00
Suministros	Material de oficina, impresiones y empastados	$50.00	$50.00
Licencias	Software Open Source (Laravel, React, PostgreSQL)	$0.00	$0.00
TOTAL	**Presupuesto Referencial del Proyecto**		**$2,090.00**

Tabla 3. Recursos Económicos

4.3 Etapas de acción para el desarrollo de la propuesta

El desarrollo del "Sistema de Gestión de Inventarios EPESPO" se estructuró siguiendo una metodología ágil e iterativa, dividiendo el proceso en tres etapas fundamentales que abarcan desde la persistencia de los datos hasta la interfaz de usuario. Esta segmentación garantiza que cada componente del sistema funcione de manera correcta antes de integrarse con el siguiente.

4.3.1 Diseño del Modelado de la Base de Datos

La fase inicial del desarrollo se centró en el modelado lógico y físico de la estructura de persistencia, empleando PostgreSQL como motor de base de datos. Con el objetivo de asegurar la integridad referencial y minimizar la redundancia de la información, se construyó un modelo relacional estandarizado bajo los criterios de la Tercera Forma Normal (3FN). La implementación del esquema se ejecutó mediante el sistema de migraciones de Laravel, mecanismo que facilita el control de versiones sobre la arquitectura tabular y permite la

evolución controlada del modelo de datos. Las entidades centrales resultantes de este diseño se detallan a continuación:

4.3.1.1 Módulos de seguridad y Actores

Este grupo de tablas gestiona el acceso al sistema y la información de las personas involucradas en la custodia de bienes.

1. **Tabla usuarios:** Almacena las credenciales de acceso, de esa manera diferencian a los operadores del sistema.
    - **Campos clave:** correo (Login), contrasena (Hash), rol (Encargado/Director) y activo (Estado del usuario).
    - **Relación:** Se puede vincular con un responsable (responsable_id).
2. **Tabla responsables:** Contiene la información del personal (docentes, administrativos) que puede recibir bienes en custodia. Es una entidad separada de los usuarios del sistema para permitir que personas sin acceso al software puedan ser responsables de activos.
    - **Campos clave:** cedula, nombre, apellido, cargo, titulo y estado de actividad.

4.3.1.2 Módulo de Inventario y Ubicación

Tablas destinadas al registro físico y lógico de los bienes y su localización dentro de la institución.

3. **Tabla productos (Inventario):** Constituye la entidad nuclear del modelo de datos, encargada de la representación digital y sistematización de cada activo físico existente en la institución.
    - **Campos clave:** codigo (Identificador patrimonial vigente), codigo_anterior (Referencia histórica para procesos de migración de datos), numero_serie,

modelo, marca, estado (Categorización operativa: Bueno, Malo, Baja), es_donado (Indicador booleano de origen) y fecha_ingreso.

- **Detalle:** Incorpora atributos de caracterización física tales como color y dimensiones y variables de control administrativo como motivo_baja. La persistencia de estos datos garantiza la trazabilidad completa del ciclo de vida del bien, proporcionando el sustento documental necesario para los procesos de auditoría y fiscalización.

4. **Tabla departamentos:** Esta entidad modela la distribución espacial y la arquitectura organizacional de la EPESPO. Su función es establecer referencias físicas concretas que permitan determinar la ubicación exacta de los activos dentro de las instalaciones.

    - **Campos clave:** nombre (Denominación del área funcional o dependencia), ubicacion (Referencia estructural del emplazamiento, Edificio/Nivel) y responsable_id (Clave foránea que vincula la custodia administrativa de la zona con un usuario específico).

4.3.1.3 Módulo Operativo (Asignaciones y Recepciones)

Estas tablas se encargan de realizar los movimientos necesarios que se realicen con lo bienes

5. **Tabla asignaciones (Cabecera):** Esta entidad garantiza el control de la cadena de custodia. Al documentar cada movimiento transaccional con alto nivel de detalle, asegura la responsabilidad administrativa sobre cada activo.

    - **Campos clave:** fecha_asignacion (Tiempo del evento o del susceso), responsable_id (Identificador del responsable), area_id (Identificador del lugar del destino) y acta_id (Identificador del Acta que se genera).

6. **Tabla asignacion_producto (Detalle):** Tabla que permite relacionar múltiples productos a una sola asignación.

    - **Función:** Rompe la relación muchos a muchos, permitiendo que un acta de asignación contenga varios productos o bienes, puede ser de 10, 20 o más bienes.

7. **Tabla recepciones (Cabecera):** Esta entidad se encarga de retomar los bienes que fueron asignados anteriormente.

    - **Campos clave:** fecha_devolucion (Registro temporal del evento), responsable_id (Identificador del responsable) y acta_id (Identificador del Acta que se genera.)

8. **Tabla recepcion_producto:** Es la entidad que se encarga de receptar los bienes que fueron asignados..

4.3.1.4 Módulo de Documentación y Trazabilidad

Entidades encargadas de generar evidencia legal y mantener el historial de cambios.

9. **Tabla actas:** Esta entidad administra el registro de los instrumentos documentales probatorios, centralizando los metadatos de las Actas de Entrega y Recepción. Su función principal es vincular las operaciones lógicas del sistema con la evidencia digital generada.

    - **Campos clave:** codigo (Consecutivo numérico que identifica el documento), archivo_path y archivo_pdf_path (Referencias a la ubicación lógica del recurso en el almacenamiento del servidor), estado (Indicador del ciclo de vida del documento: Generada o Anulada).

10. **Tabla movimientos:** Esta entidad se encarga de documentar las irregularidades del sistema o los cambios que se hacen dentro del sistema,

    - **Campos clave:** accion (operaciones: creación, edición, eliminación), descripcion (Especificaciones del movimiento que se ha hecho), usuario_id (Identificación del usuario) y fecha (Tiempo).

11. **Tabla producto_asignaciones_actuales:** Estructura auxiliar diseñada para optimizar el rendimiento de las consultas sobre la tenencia vigente de los activos, obviando la necesidad de procesar recursivamente el historial transaccional completo.

    - **Función:** Facilita la identificación inmediata de la custodia activa (responsable_id) y la ubicación física (area_id), reduciendo la carga computacional en la recuperación del estado actual del inventario.

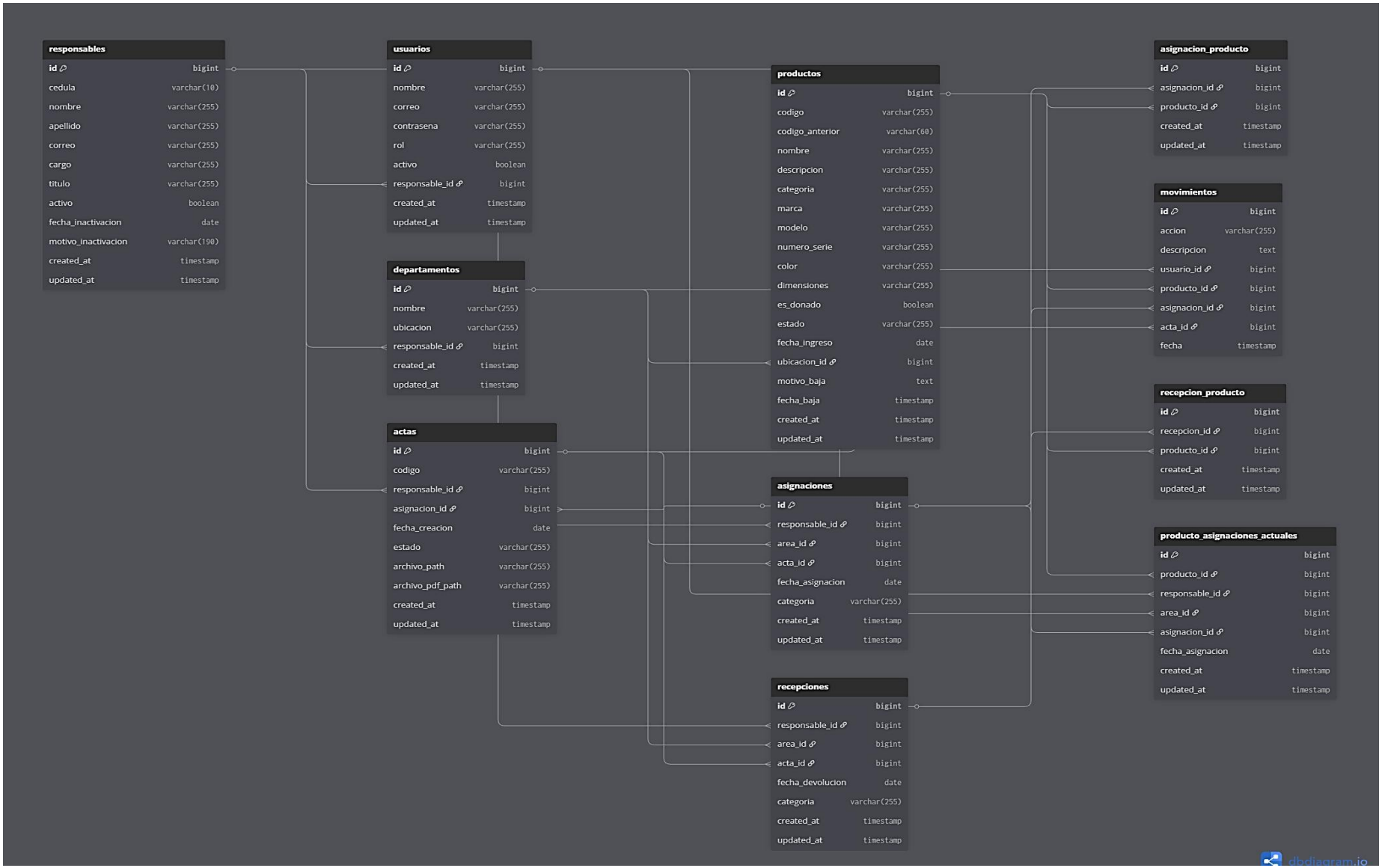

Figura 6. Tabla de la Base de Datos

4.3.2 Diseño y Desarrollo del Backend

La lógica del servidor se construyó sobre el framework Laravel, implementando una API RESTful que sirve como puente entre la base de datos y la interfaz de usuario.

Las actividades técnicas ejecutadas durante esta fase se estructuraron de la siguiente manera:

- **Configuración del Entorno:** La inicialización del proyecto comprendió la instalación del framework Laravel y la definición de variables en el archivo .env, acción necesaria para establecer la conectividad con el motor de base de datos PostgreSQL.
- **Implementación de Modelos (Eloquent ORM):** Se desarrollaron los modelos Usuarios, Producto y Asignaciones para mapear las tablas del esquema relacional. El uso de Eloquent ORM permitió abstraer las consultas SQL en una lógica orientada a objetos, facilitando la gestión de relaciones de cardinalidad, tales como la asociación uno a muchos entre usuarios y asignaciones.
- **Desarrollo de Controladores (API):** La lógica de negocio se encapsuló en los controladores AuthController, ProductoController y AsignacionController. La implementación incluyó la estandarización de las respuestas del servidor, validando la emisión correcta de códigos de estado HTTP (200, 201, 401) para garantizar la coherencia del intercambio de datos.
- **Seguridad con Middleware:** La protección de las rutas en routes/api.php se instrumentó mediante Laravel Sanctum. Esta capa de seguridad intercepta las solicitudes entrantes para verificar credenciales y roles, impidiendo el acceso no autorizado a operaciones críticas del sistema.

4.3.3 Diseño y Desarrollo del FrontEnd

La interfaz de usuario se desarrolló como una aplicación de página única (SPA) utilizando la biblioteca React, enfocándose en la usabilidad y la experiencia del usuario (UX).

El desarrollo de la interfaz de usuario se estructuró en componentes modulares, abarcando las siguientes áreas técnicas:

- **Gestión de Estado y Rutas:** La navegación interna entre los módulos del sistema (Login, Dashboard, Inventario) se administra mediante React Router, lo que permite el tránsito fluido sin recargas del navegador. .
- **Integración con la API (Axios):** No es un simple capricho de programador, es el sistema de mensajería blindado que mantiene viva la conversación entre el usuario y el servidor. Es un trabajo sucio pero necesario. Mediante la configuración de interceptores, el sistema inyecta automáticamente el dichoso *Bearer Token* en las cabeceras de cada petición HTTP, actuando como un sello de cera que valida la identidad del solicitante en cada transacción.
- **Módulo de Autenticación:** se encarga de ese primer choque de realidad donde las credenciales se enfrentan al veredicto del servidor. Si la verificación es exitosa, el sistema no lanza las campanas al vuelo, sino que se pone a trabajar para garantizar la persistencia de la sesión mediante el almacenamiento del token en el *local storage* del navegador. Es una técnica clásica pero efectiva. De esta forma, el estado del usuario se mantiene intacto mientras navega por los pasillos digitales de la gestión de inventarios, evitando la tortura de tener que identificarse cada vez que desea consultar el paradero de un activo.
- **Interfaces de Gestión (CRUD):** Se desarrollaron vistas interactivas para la administración de inventarios y usuarios. La interfaz incorpora tablas dinámicas y

ventanas modales que facilitan las operaciones de creación, edición y eliminación de activos, optimizando el flujo operativo del personal administrativo.

4.4 Etapas de acción para el desarrollo de la aplicación

El ciclo de vida del software se gestionó bajo el marco de trabajo ágil Scrum. Esta metodología permitió organizar el desarrollo mediante iteraciones cortas o 'Sprints', priorizando las funcionalidades a través de un Product Backlog definido junto a los usuarios expertos de la institución. El proceso se estratificó en tres fases operativas, garantizando la entrega de módulos funcionales y evaluables en cada hito del proyecto.

4.4.1 Fase I: Definición y Planificación

Esta fase inicial se centró en comprender a profundidad la problemática de la EPESPO y establecer los cimientos del proyecto. Las actividades principales fueron:

1. **Levantamiento de la Información dentro de la institución:** Se realizo entrevistas con una temática profundas con el director Administrativo y el responsable de inventario para diseccionar, como si de una autopsia institucional se tratara, el estado real de sus procesos internos. Fue un diagnóstico crudo. La realidad saltó a la vista de inmediato: la dependencia enfermiza de hojas de cálculo dispersas no era más que un nido de errores. Esas tablas aisladas, que flotaban en distintos correos y carpetas como boyas a la deriva, generaban una duplicidad de datos asfixiante y una falta de trazabilidad que convertía la búsqueda de cualquier bien en un ejercicio de fe.

2. **Definición de los Requisitos:** Se realizo el cuestionamiento de las necesidades dentro de los procesos tanto como administrativos como prácticos en el inventario, delatando así las políticas de este flujo de trabajo.
3. **Selección de las Tecnologías a usar:** Se evaluaron criterios de escalabilidad y rendimiento con la lupa de quien sabe que un sistema lento es un sistema muerto. La elección fue clara. En el backend, Laravel se erige como el capataz de la obra,

gestionando las APIs con una seguridad casi paranoica y una eficiencia que pone orden en el caos de las peticiones del servidor. Es el músculo invisible. Mientras tanto, en la superficie, la interfaz de usuario se forjó con React para que el navegador no se sienta como una hoja de papel estática, sino como un organismo vivo que responde al instante a cada clic del operador.

4.4.1.1 Requisitos del sistema

Se definieron los requisitos funcionales y no funcionales, los cuales actúan como la hoja de ruta para el desarrollo del sistema.

Funcionales

Describimos las funciones y el compartimiento que le sistema debe de poder ejecutar.

- **Gestión De Autenticación**
  - El sistema debe permitir el inicio de sesión seguro mediante correo y contraseña, diferenciando entre roles de Encargado y Director.
- **Gestión de Usuarios**
  - Centraliza la administración de cuentas, facultando al perfil administrativo para crear, modificar y restringir accesos según los roles asignados.
- **Gestión de Inventario**
  - Permite la catalogación detallada de activos, registrando códigos patrimoniales, series y estados de conservación para el control de altas y bajas.
- **Control de Ubicaciones**
  - Gestiona la estructura física institucional, definiendo áreas y departamentos para la geolocalización precisa de los bienes.
- **Asignación de Custodia**
  - Vincula los activos con sus funcionarios responsables, generando un historial inalterable de la entrega y las condiciones del bien para fines de trazabilidad.

- **Gestión de Devoluciones**
  - Regula el proceso de retorno de equipos, actualizando el estatus del activo a que esté disponible de forma automática tras su recepción en bodega.
- **Generación de Actas**
  - Automatiza la emisión de documentos probatorios en formato digital, formalizando legalmente los movimientos de entrega y recepción.
- **Búsqueda y Filtros**
  - Incorpora mecanismos de filtrado rápido que permiten localizar registros específicos mediante criterios como código, descripción o custodio asignado.

Requisitos no Funcionales

Se describen las restricciones técnicas que tiene el sistema o aplicación web.

- **Seguridad de Datos**
  - La protección de la información y de los datos exige el almacenamiento que será encriptado y esto será de las credenciales mediante Hash Bcrypt y la validación de sesiones a través de tokens seguros (Laravel Sanctum).
- **Usabilidad**
  - El diseño de la interfaz debe ser responsivo y amigable con el usuario, de modo que pueda intuir los módulos o los diferentes apartados, garantizando la operatividad sobre todo en el escritorio sin requerir conocimientos técnicos avanzados por parte del usuario.
- **Rendimiento**
  - Las consultas de inventario y generación de reportes no deben exceder los 3 segundos de tiempo de respuesta.
- **Disponibilidad**

- El sistema operará bajo un esquema de servicio continuo (24/7), limitando las interrupciones exclusivamente a las ventanas de mantenimiento técnico programado.

- **Escalabilidad**
  - La arquitectura del software debe soportar el crecimiento sostenido de la base de datos, manteniendo la estabilidad del rendimiento ante el aumento del volumen de registros históricos.
- **Compatibilidad**
  - La aplicación debe garantizar plena interoperabilidad con los estándares vigentes de los navegadores web modernos, incluyendo Chrome, Firefox y Edge.

4.4.1.2 Product Backlog e Historias de Usuario

Para la gestión de los requerimientos bajo el marco de trabajo ágil Scrum, se elaboró el Product Backlog del sistema utilizando el formato de Historias de Usuario. Esta herramienta permitió priorizar las necesidades operativas de la institución desde la perspectiva de los actores principales (Encargado y Director), definiendo claramente los criterios de aceptación que el software debía cumplir en cada iteración de desarrollo.

Módulo de Seguridad y Accesos

Historia de Usuario HU-01 Autenticación y acceso al sistema

Historia de Usuario	**HU-01**
Nombre	Autenticación y acceso al sistema
Actor	Encargado
Prioridad	Alta

Descripción	Como encargado, quiero iniciar sesión mediante credenciales seguras, para acceder al panel de control y gestionar el inventario patrimonial.
Criterios de aceptación	- El sistema debe permitir ingresar correo y contraseña. - Se debe validar mediante Hash que las credenciales coincidan con la base de datos. - Si los datos son incorrectos, debe mostrar un mensaje de alerta. - Al ingresar correctamente, debe generar el token y redirigir al Dashboard principal.

Tabla 4. Historia de Usuario Autenticación y acceso al sistema

Historia de Usuario HU-02 Acceso con perfil de auditoría

Historia de Usuario	**HU-02**
Nombre	Acceso con perfil de auditoría
Actor	Director
Prioridad	Media
Descripción	Como director, quiero acceder a la plataforma con un perfil restringido, para auditar y consultar la ubicación de los bienes sin riesgo de alterar los registros.
Criterios de aceptación	- El sistema debe ocultar condicionalmente los botones de crear, editar y eliminar. - El usuario solo podrá visualizar y filtrar los listados de bienes y asignaciones. - El backend debe bloquear cualquier intento de modificación de datos retornando un error de permisos.

Tabla 5. Historia de Usuario Acceso con perfil a Auditoria

Módulo de Gestión de Inventario

Historia de Usuario HU-03 Registro de nuevos bienes

Historia de Usuario	**HU-03**
Nombre	Registro de nuevos bienes
Actor	Encargado
Prioridad	Alta
Descripción	Como encargado, quiero registrar un nuevo activo ingresando su código patrimonial, descripción y estado físico, para mantener la base de datos de bienes actualizada.
Criterios de aceptación	- El formulario debe requerir campos obligatorios: código, nombre, marca y serie. - El sistema debe verificar que el código patrimonial sea único y no se duplique. - Debe permitir seleccionar el estado físico.

Tabla 6. Historia de Usuario Registro de nuevos bienes

Historia de Usuario HU-04 Edición y baja lógica de bienes

Historia de Usuario	**HU-04**
Nombre	Edición y baja lógica de bienes
Actor	Encargado
Prioridad	Alta
Descripción	Como encargado, quiero actualizar la información de un bien o darle de baja, para reflejar su estado físico real o su obsolescencia.

Criterios de aceptación	- El sistema debe permitir modificar los datos descriptivos del equipo. - Se debe permitir cambiar el estado general a "Dado de baja". - El sistema debe impedir dar de baja un bien si este se encuentra actualmente asignado a un custodio.

Tabla 7. Historia de Usuario Edición y baja de bienes

Historia de Usuario HU-05 Búsqueda y filtrado dinámico

Historia de Usuario	**HU-05**
Nombre	Búsqueda y filtrado dinámico
Actor	Encargado / Director
Prioridad	Media
Descripción	Como usuario del sistema, quiero buscar un activo utilizando filtros rápidos, para localizar su información y ubicación en tiempo real.
Criterios de aceptación	- Debe existir un campo de búsqueda en la vista del catálogo. - La tabla de resultados debe actualizarse dinámicamente al ingresar el código patrimonial o el nombre. - El tiempo de respuesta de la consulta no debe exceder los parámetros óptimos de rendimiento.

Tabla 8. Historia de Usuario Búsqueda y Filtro

Módulo de Ubicaciones y Personal

Historia de Usuario HU-06 Catálogo de responsables y departamentos

Historia de Usuario	**HU-06**
Nombre	Catálogo de responsables y departamentos
Actor	Encargado

Prioridad	Alta
Descripción	Como encargado, quiero registrar los datos del personal (docentes/administrativos) y departamentos, para tener un directorio exacto de los custodios físicos de la institución.
Criterios de aceptación	- El sistema debe permitir ingresar cédula, nombre y cargo del responsable. - Se debe poder crear áreas físicas (departamentos). - El sistema validará que el número de identificación (cédula) no esté duplicado en la base de datos.

Tabla 9. Historia de Usuario Catálogos

Historia de Usuario HU-07 Actualización de estado de responsables

Historia de Usuario	**HU-07**
Nombre	Actualización de estado de responsables
Actor	Encargado
Prioridad	Media
Descripción	Como encargado, quiero inactivar a un responsable cuando finalice sus labores en la entidad, para mantener la integridad del directorio y evitar nuevas asignaciones erróneas.
Criterios de aceptación	- El sistema habilitará la opción para cambiar el estado de "Activo" a "Inactivo". - Un responsable inactivo no debe aparecer en las listas desplegables para nuevas asignaciones.

Tabla 10. Historia de Usuario Estado de responsable

Módulo de Operaciones

Historia de Usuario HU-08 Asignación de custodia de bienes

Historia de Usuario	**HU-08**
Nombre	Asignación de custodia de bienes
Actor	Encargado
Prioridad	Alta
Descripción	Como encargado, quiero vincular uno o varios bienes disponibles a un responsable específico, para formalizar la entrega y establecer la trazabilidad del activo.
Criterios de aceptación	- El sistema solo debe listar bienes en estado "Disponible". - Tras confirmar la transacción, el estado de los bienes debe cambiar a "Asignado". - Se debe generar un registro histórico con la fecha exacta del movimiento.

Tabla 11. Historia de Usuario Asignación de custodia

Historia de Usuario HU-09 Recepción y devolución de bienes

Historia de Usuario	**HU-09**
Nombre	Recepción y devolución de bienes
Actor	Encargado
Prioridad	Alta
Descripción	Como encargado, quiero registrar el retorno de un bien a bodega, para liberar de responsabilidad al custodio y que el artículo vuelva a estar disponible.
Criterios de aceptación	- El sistema debe permitir buscar los bienes asignados actualmente a un responsable.

	- Al procesar la recepción, el activo debe desvincularse del usuario. - El estado del bien se actualizará automáticamente a "Disponible".

Tabla 12. Historia de Usuario Recepción de bienes

Historia de Usuario HU-10 Generación de Actas en formato PDF

Historia de Usuario	**HU-10**
Nombre	Generación de Actas en formato PDF
Actor	Encargado
Prioridad	Alta
Descripción	Como encargado, quiero que el sistema genere automáticamente un documento legal al realizar una entrega o devolución, para contar con un respaldo físico listo para las firmas.
Criterios de aceptación	- El PDF debe renderizarse con el formato institucional oficial. - El documento autocompletará los datos del custodio, el encargado y el detalle técnico de los bienes involucrados. - Debe permitir la descarga inmediata del archivo.

Tabla 13. Historia de Usuario Generación de actas

4.4.1.3 Asignación de Roles Scrum

Para garantizar la correcta ejecución del marco de trabajo ágil y asegurar la trazabilidad del desarrollo, se definieron roles específicos adaptados a la estructura del proyecto y a la realidad institucional de la EPESPO:

- **Product Owner (Dueño del Producto):** Representado por el Director Administrativo de la EPESPO. Su rol consistió en aportar la visión institucional, definir las prioridades operativas del sistema de inventario y validar los requerimientos plasmados en el *Product Backlog*.

- **Scrum Master:** Asumido por la Directora de Tesis (Ing. Patricia Quiroz Palma), encargada de supervisar el rigor metodológico, guiar al equipo en la aplicación de las buenas prácticas de ingeniería de software y facilitar la resolución de impedimentos técnicos.
- **Development Team (Equipo de Desarrollo):** Conformado por los investigadores (Devis Alonzo Pico y Jerick Bailon Quijije), quienes asumieron la responsabilidad técnica integral: análisis de requerimientos, diseño de la arquitectura (React JS y Laravel), estructuración de la base de datos relacional (PostgreSQL), codificación y ejecución de las pruebas.

4.4.1.4 Timeboxing y Gestión de Sprints

El ciclo de vida del software se estructuró mediante iteraciones fijas. Para mantener un ritmo de trabajo sostenido y permitir la evaluación continua, se definieron Sprints con una duración exacta de dos semanas cada uno.

Al inicio de cada ciclo, mediante la ceremonia de Sprint Planning, el equipo de desarrollo seleccionaba las Historias de Usuario de mayor prioridad para conformar el Sprint Backlog. El control de progreso (lo planificado versus lo completado) se gestionó estrictamente a través de un tablero Kanban en la herramienta Trello. Esta organización garantizó que, al finalizar cada periodo de dos semanas, el equipo no entregara documentación aislada, sino un incremento del sistema (módulo) completamente funcional y listo para la fase de pruebas.

4.4.1.5 Planificación de Sprints (Iteraciones)

La transición de los requerimientos teóricos a la codificación real exigió un marco de trabajo estrictamente disciplinado. Para evitar el caos propio de los desarrollos monolíticos, el ciclo de construcción del software se fragmentó en iteraciones cortas denominadas "Sprints", aplicando los principios operativos de la metodología Scrum. Esta estrategia técnica permitió

aislar la complejidad del sistema, abordando la programación de manera modular y asegurando un control de calidad progresivo.

La gestión de estas iteraciones se orquestó mediante la implementación de un tablero Kanban virtual. El uso de esta herramienta no fue una simple formalidad; funcionó como el eje central para controlar el ciclo de vida de cada Historia de Usuario, asignando tiempos exactos de inicio y fin (Timeboxing) y desglosando los requerimientos en tareas de programación específicas (creación de controladores, migraciones y validaciones de interfaz).

El desarrollo se estructuró matemáticamente en cuatro ciclos operativos:

- Un primer sprint enfocado en la cimentación del esquema relacional en PostgreSQL y la seguridad de acceso mediante tokens.
- Un segundo ciclo dedicado íntegramente a construir el motor lógico del inventario.
- Un tercer sprint diseñado para resolver la complejidad transaccional de las asignaciones, devoluciones y la renderización de actas legales.
- Una iteración final reservada en exclusiva para las auditorías de software, pruebas de concurrencia y el despliegue técnico en los servidores de producción.

A continuación, se expone la evidencia visual de esta planificación estructurada, reflejando el flujo de trabajo desde la asignación de la tarea hasta su cumplimiento técnico:

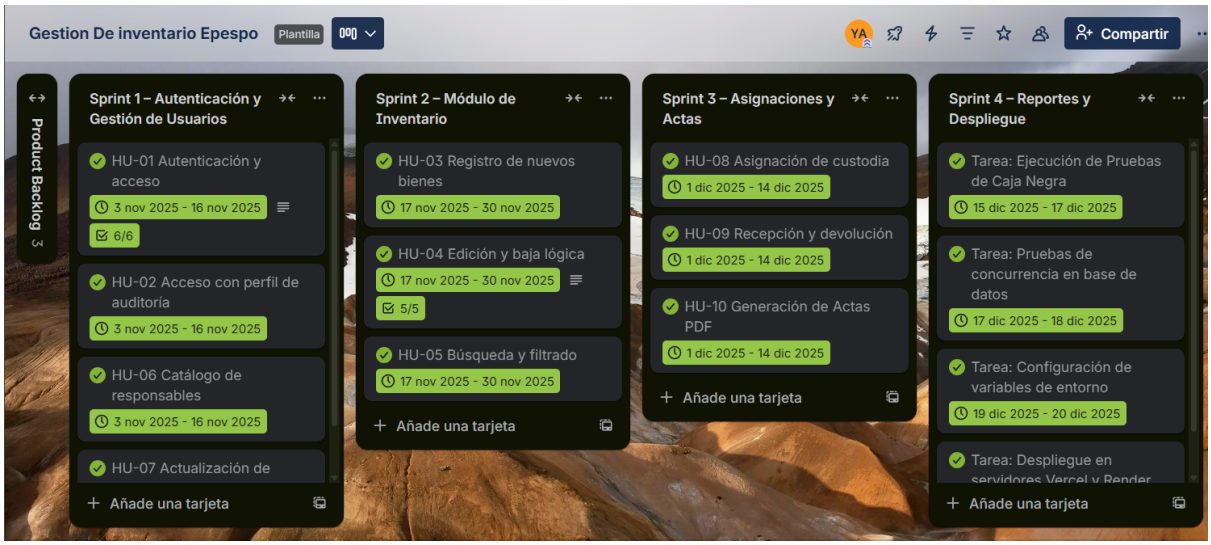

Figura 7. Tablero Kanban general de la planificación de Sprints en Trello

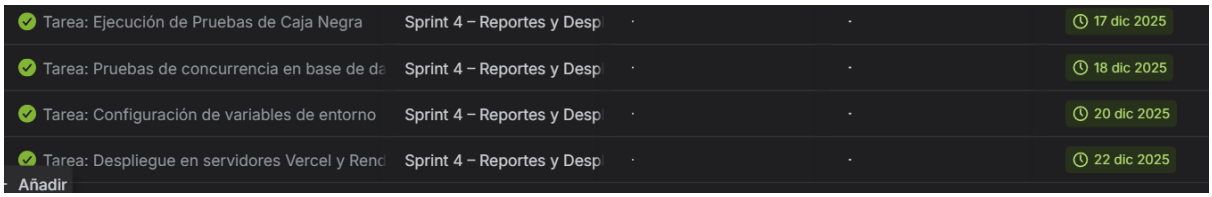

Figura 8. Desglose de tareas técnicas internas para el cumplimiento de una Historia de Usuario

4.4.1.6 Casos de Uso

Los casos de uso detallan la interacción formal entre los actores y el sistema para dar cumplimiento a los requisitos funcionales establecidos en la fase de planificación.

Encargado: Perfil con privilegios de superusuario, habilitado para ejecutar operaciones de creación, lectura, actualización y eliminación (CRUD) sobre todos los módulos operativos del sistema.

Director: Perfil diseñado con privilegios de solo lectura para la consulta de información y visualización de reportes, restringiendo cualquier capacidad de modificación de datos para preservar la integridad de los registros institucionales.

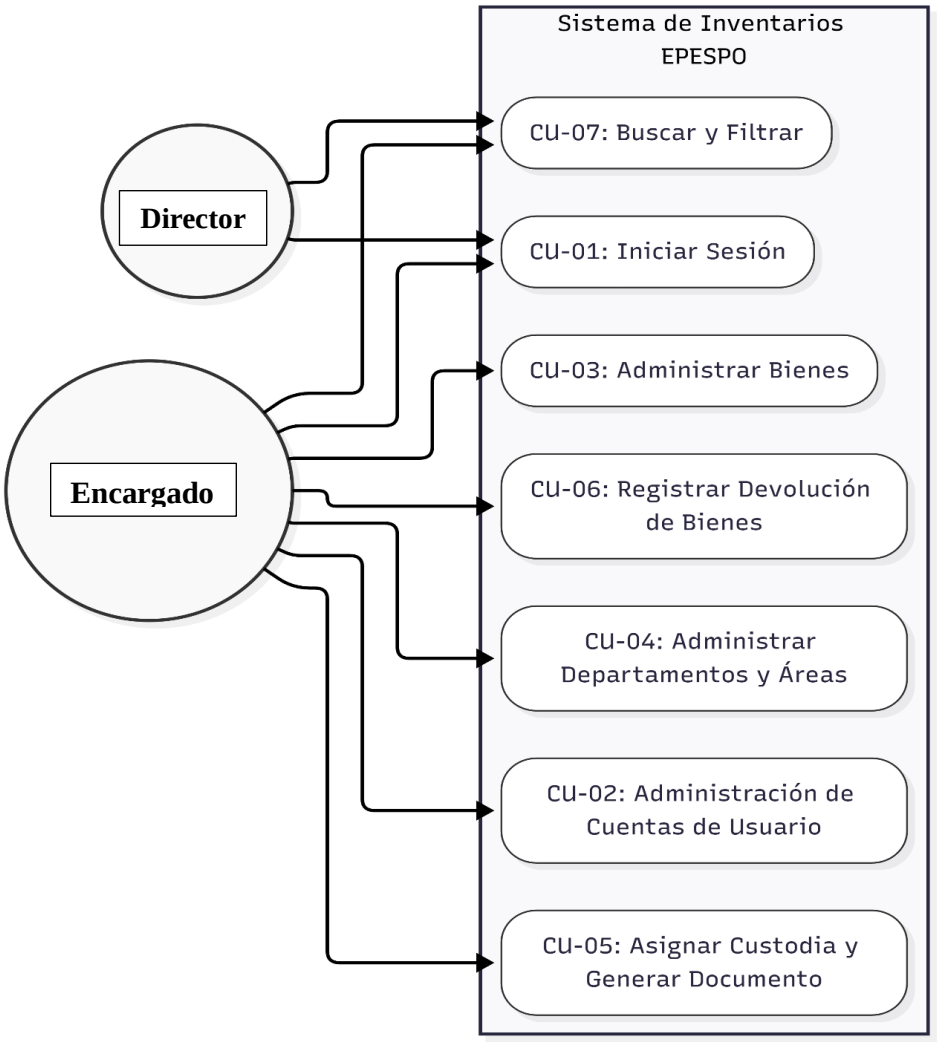

Figura 9. Diagrama General de Casos de Uso

4.4.1.6.1 Autenticación de Usuarios

Nombre del caso: Iniciar Sesión

Identificador: CU-01	Requisito Asociado: Gestión De Autenticación (RF-01) y Seguridad de Datos (RNF-01)
Actores: Encardado, Director	Descripción: Permite a los usuarios ingresar al sistema únicamente con sus credenciales de acceso.
Pre-Condición: Requiere la existencia de un registro de usuario que este guardado en la base de datos.	
Flujo Principal: 1. El usuario ingresa a la aplicación web. 2. El sistema solicita correo y contraseña. 3. El usuario ingresa sus credenciales como correo y contraseña. 4. El sistema encripta la contraseña y la compara con la base de datos (Hash Bcrypt). 5. Si es correcta, se genera un Token de acceso al sistema. 6. El sistema redirige al panel principal según el rol que tenga ese usuario.	
Postcondición: El usuario obtiene acceso operativo a las funciones y permisos habilitados para su perfil.	

Tabla 14. Caso de Uso Autenticación de Usuarios

4.4.1.6.2 Gestión de Usuario

Nombre del caso: Administración de Cuentas de Usuario

Identificador: CU-02	Requisito Asociado: *Gestión de Usuarios* (RF-02)
Actores: Encargado	Descripción: Permite registrar nuevos operadores del sistema, modificar sus datos o inhabilitar su acceso.
Pre-Condición: El encargado debe haber iniciado sesión con permisos totales.	
Flujo Principal: 1. El encargado accede al módulo "Usuarios". 2. El sistema muestra la lista de usuarios registrados y su estado. 3. El encargado selecciona "Crear Nuevo Usuario". 4. Ingresa los datos: Nombre, Correo Institucional, Contraseña y Rol (Encargado o Director). 5. El sistema valida que el correo no esté duplicado. 6. El encargado guarda el registro. 7. El sistema confirma la creación y envía las credenciales.	
Postcondición: El nuevo usuario puede acceder al sistema o el usuario inactivo pierde el acceso inmediatamente.	

Tabla 15. Caso de Uso Gestión de Usuario

4.4.1.6.3 Gestión de Inventario

Nombre del caso: Administrar Bienes

Identificador: CU-03	Requisito Asociado: Gestión de Inventario (RF-03)
Actores: Encargado	Descripción: Permite registrar nuevos bienes, modificar sus características o darles de baja.
Pre-Condición: El Encargado debe haber iniciado sesión y tener permisos de escritura.	
Flujo Principal: 1. El encargado accede al módulo u apartado de Inventario. 2. Selecciona "Agregar Bien". 3. Ingresa los datos como serie, estado, modelo, etc. 4. El sistema valida que el Código no exista con un anterior bien. 5. El encargado o guarda el registro. 6. El sistema almacena la información y agrega el bien.	
Postcondición: El bien queda registrado en la base de datos con estado "Activo" y disponible para ser asignado.	

Tabla 16. Caso de Uso Gestión de Inventario

4.4.1.6.4 Gestión de Ubicaciones

Nombre del caso: Administrar Departamentos y Áreas

Identificador: CU-04	Requisito Asociado: Control de Ubicaciones (RF-04)
Actores: Encargado	Descripción: Permite crear o modificar las áreas físicas donde se encontrarán los bienes.
Pre-Condición: Se debe acceder al sistema como encargado.	
Flujo Principal: 1. El encargado accede al módulo Departamentos. 2. Selecciona el apartado Agregar nuevo departamento o área. 3. Ingresa el nombre y la ubicación física. 4. Asigna un responsable encargado del área. 5. Guarda el registro. 6. El sistema actualiza el catálogo de ubicaciones disponibles para futuras asignaciones.	
Postcondición: El área queda disponible para vincular bienes.	

Tabla 17. Caso de Uso Gestión de Ubicaciones

4.4.1.6.5 Asignación y Generación de Actas

Nombre del caso: Asignar Custodia y Generar Documento

Identificador: CU-05	Requisito Asociado: Asignación de Custodia (RF-05) y Generación de Actas (RF-07)
Actores: Encargado	Descripción: Vincula uno o varios bienes a un funcionario y genera el PDF de respaldo.
Pre-Condición: Deben existir bienes que estén disponibles y el responsable debe estar registrado en la aplicación web o existir en la base de datos del sistema.	
Flujo Principal: 1. El encargado o encargado selecciona a un Responsable. 2. Selecciona la Ubicación de destino. 3. Busca y selecciona los bienes que va a asignar. 4. Se Confirma la asignación y queda lista. 5. El sistema cambia el estado de los bienes a asignado con su registro de la fecha. 6. De forma automática el sistema genera el Acta de Entrega en formato PDF para su descarga.	
Postcondición: Los bienes cambian su estado una vez hayan sido asignado, se vinculan al custodio seleccionado y se crea el archivo digital del acta.	

Tabla 18. Caso de Uso Asignación y Generación de Actas

4.4.1.6.6 Gestión de Devoluciones (Recepciones)

Nombre del caso: Registrar Devolución de Bienes

Identificador: CU-06	Requisito Asociado: Gestión de Devoluciones (RF-06)
Actores: Encargado	Descripción: Proceso mediante el cual un custodio devuelve un bien, liberándolo de su responsabilidad.
Pre-Condición: El bien debe de estar asignado a un responsable.	
Flujo Principal: 1. El encargado busca al funcionario que devuelve el bien. 2. El sistema muestra los bienes que esa persona tiene a su cargo. 3. El encargado selecciona los bienes que se están devolviendo. 4. Ingresa una observación sobre el estado físico de entrega. 5. Confirma la recepción. 6. El sistema actualiza automáticamente el estado del bien quedando disponible y desvincula al custodio.	
Postcondición: El bien queda libre para ser asignado nuevamente a otra persona o departamento.	

Tabla 19. Caso de Uso Gestión de Recepciones

4.4.1.6.7 Búsqueda y Filtros

Nombre del caso: Consultar y Filtrar Bienes

Identificador: CU-07	Requisito Asociado: Búsqueda y Filtros (RF-08) y Rendimiento (RNF-03)
Actores: Encargado Director	Descripción: Permite localizar bienes específicos mediante criterios como código, nombre, estado o responsable.
Pre-Condición: El usuario debe haber iniciado sesión y deben de existir bienes registrados en el sistema.	
Flujo Principal: 1. El usuario accede a los bienes del módulo del inventario. 2. El sistema le muestra la lista completa de bienes paginada. 3. El usuario ingresa alguna búsqueda con criterio en los filtros. 4. El usuario puede aplicar filtros adicionales. 5. El sistema procesa la solicitud en tiempo real (< 3 segundos). 6. El sistema actualiza la tabla solo con los resultados de la búsqueda con filtros.	
Postcondición: El usuario visualiza la información que busca de forma rápida sin alterar algo.	

Tabla 20. Caso de Uso Búsqueda y Filtros

4.4.1.7 Diagramas UML de los casos de uso

Los diagramas de secuencia detallan la interacción lógica entre la interfaz de usuario, el controlador del servidor y la base de datos para ejecutar los procesos críticos del sistema.

CU-01: Iniciar Sesión (Autenticación)

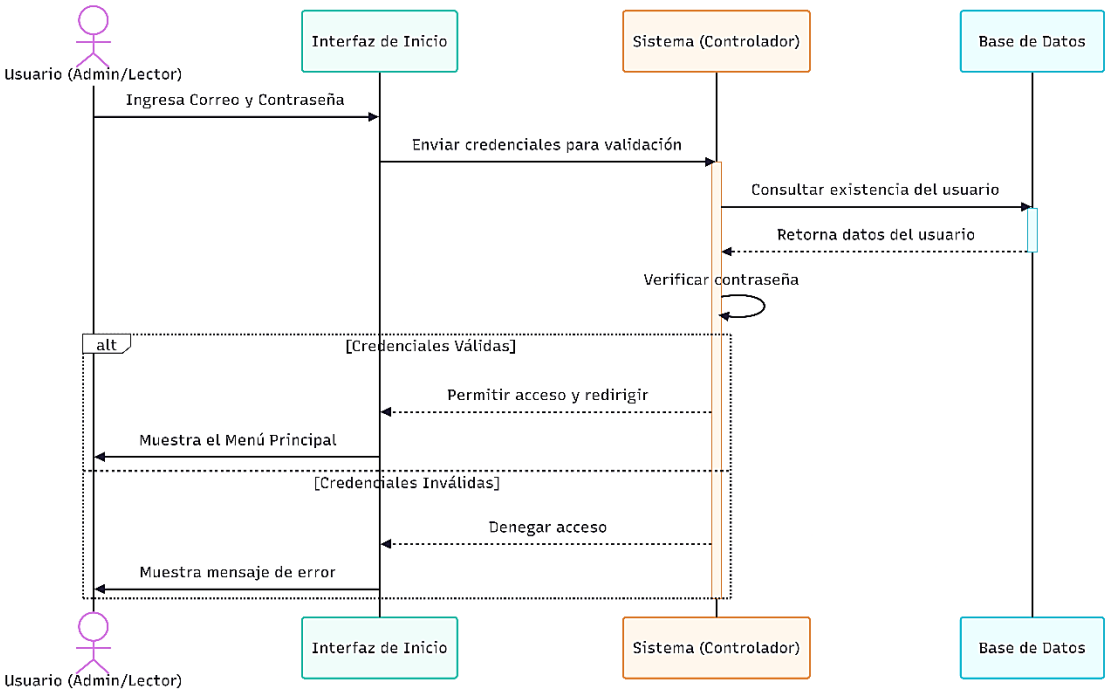

Figura 10. Iniciar Sesión - Sistema

CU-02: Administración de Cuentas de Usuario

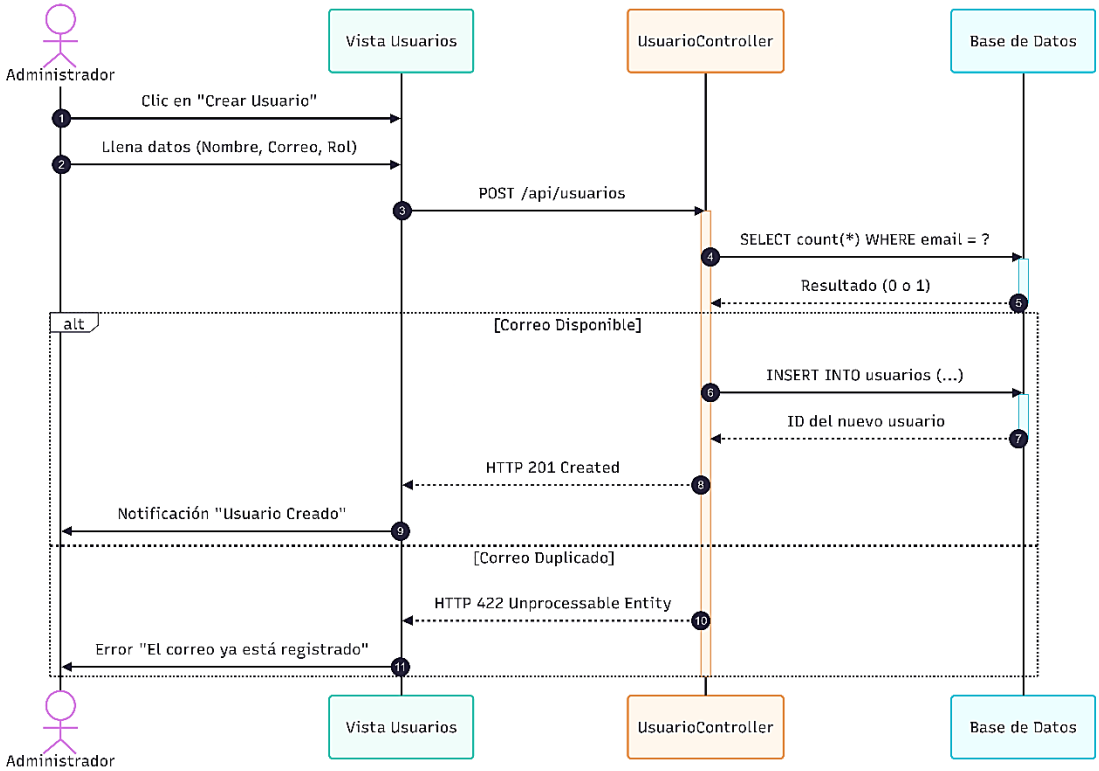

Figura 11. Administrador (Encargado) - Gestión de Usuarios

CU-03: Administrar Bienes

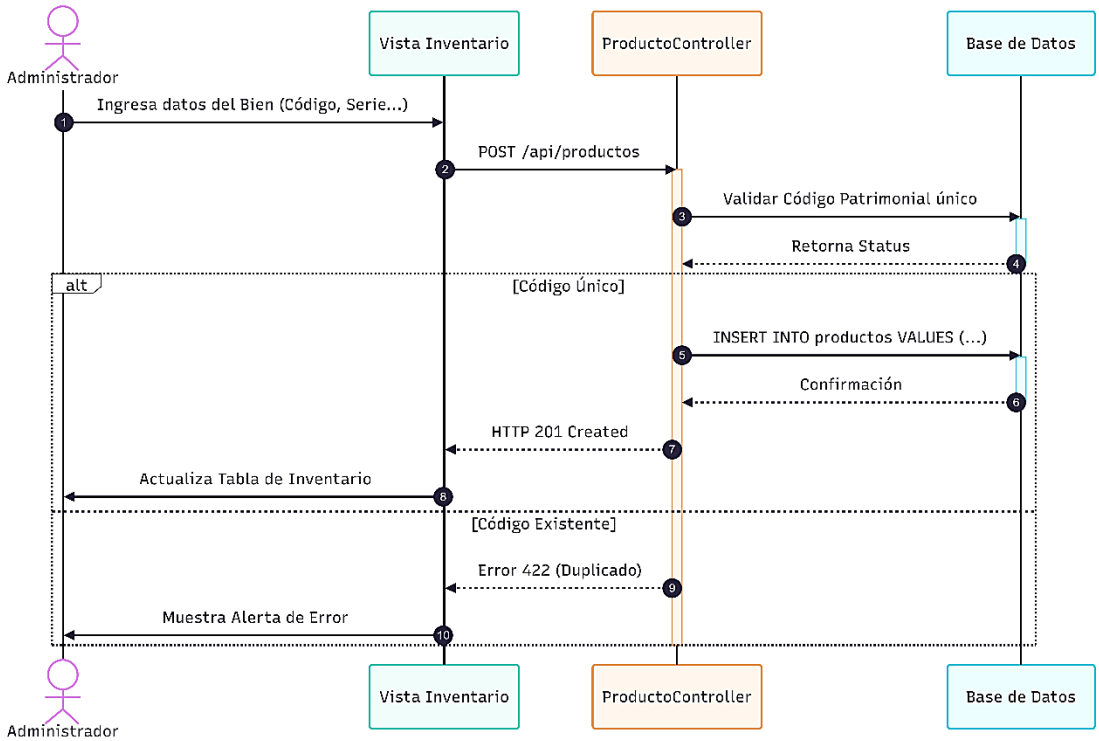

Figura 12. Administrador (Encargado) - Gestión de Inventario

CU-04: Administrar Departamentos y Áreas

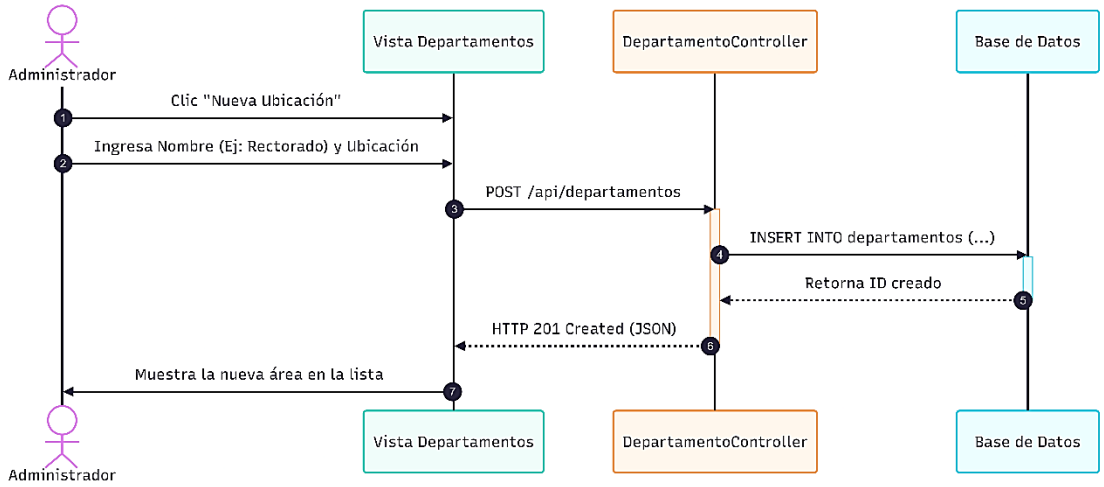

Figura 13. Administrador (Encargado) - Gestión de Ubicaciones

CU-05: Asignar Custodia y Generar Documento

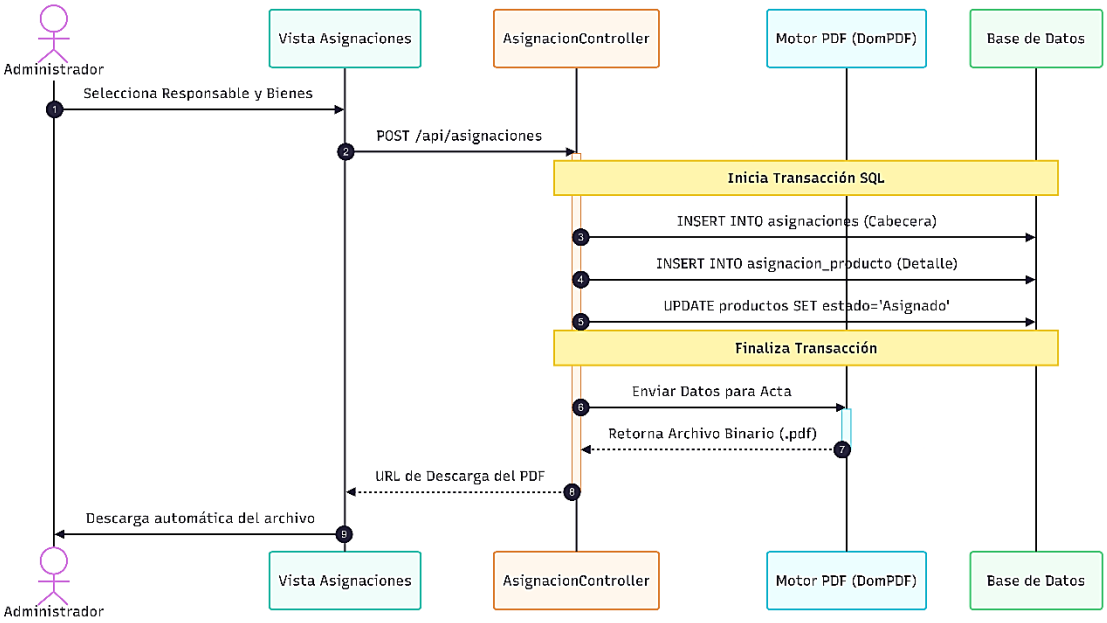

Figura 14. Administrador (Encargado) - Asignación y Actas

CU-06: Registrar Devolución de Bienes

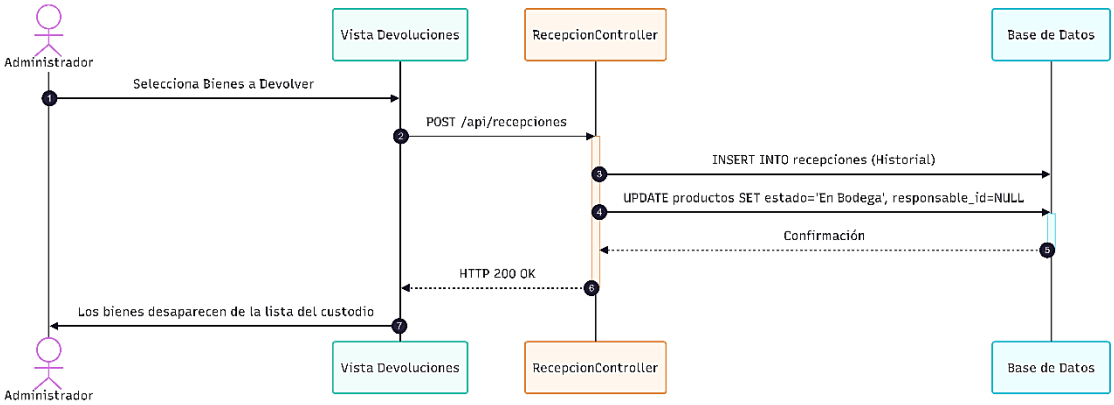

Figura 15. Encargado (Administrador) – Recepción y Actas

CU-07: Consultar y Filtrar Bienes

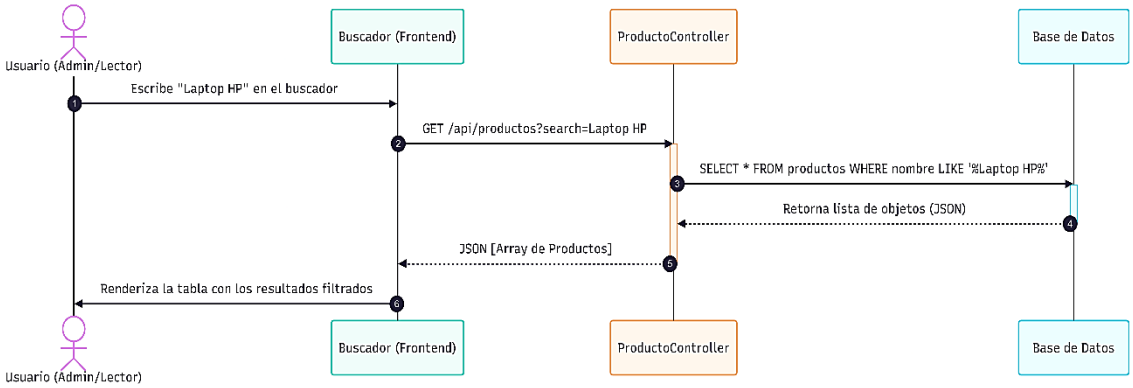

Figura 16. Usuarios - Búsqueda por Filtros

4.4.1.8 Modelo Lógico Entidad-Relación de la Base de Datos

Para garantizar la integridad, escalabilidad y seguridad de la información, se diseñó un modelo de base de datos relacional normalizado utilizando el motor PostgreSQL. El esquema se estructura en torno a la separación de responsabilidades, diferenciando claramente entre los usuarios del sistema, el personal administrativo que son los responsables y los activos fijos.

Módulo de Seguridad y Gestión de Personal

La arquitectura de seguridad establece una distinción clara entre quienes operan el software y quienes custodian los bienes. La tabla usuarios se encarga únicamente de gestionar las credenciales de acceso, almacenando las contraseñas de forma cifrada y asignando roles específicos (Encargado o Director) para restringir las funciones disponibles. Esta entidad se conecta con la tabla responsables a través de una clave foránea; dicha tabla funciona como el registro central de todo el personal de la institución, incluyendo docentes y administrativos. Gracias a esta separación, es posible asignar bienes a un funcionario sin que este requiera obligatoriamente una cuenta de acceso al sistema, lo que mantiene la base de datos de personal organizada y facilita su crecimiento futuro.

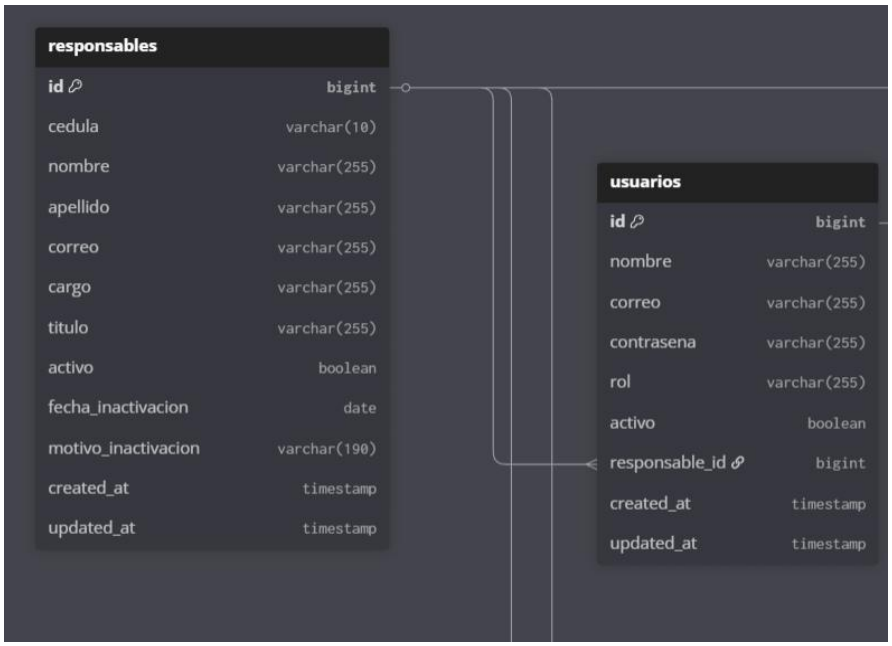

Figura 17. Seguridad y Gestión de responsables

Módulo de Inventario y Ubicación

La entidad central del esquema es la tabla de productos, estructurada para almacenar las especificaciones técnicas de cada activo fijo. A nivel de diseño, se incorporó el campo *código anterior*, el cual actúa como una llave de referencia histórica, garantizando la compatibilidad y migración segura de los datos provenientes de los registros manuales previos. Adicionalmente, la tabla de departamentos modela la distribución espacial de la EPESPO, asegurando que cada bien posea una referencia física unívoca.

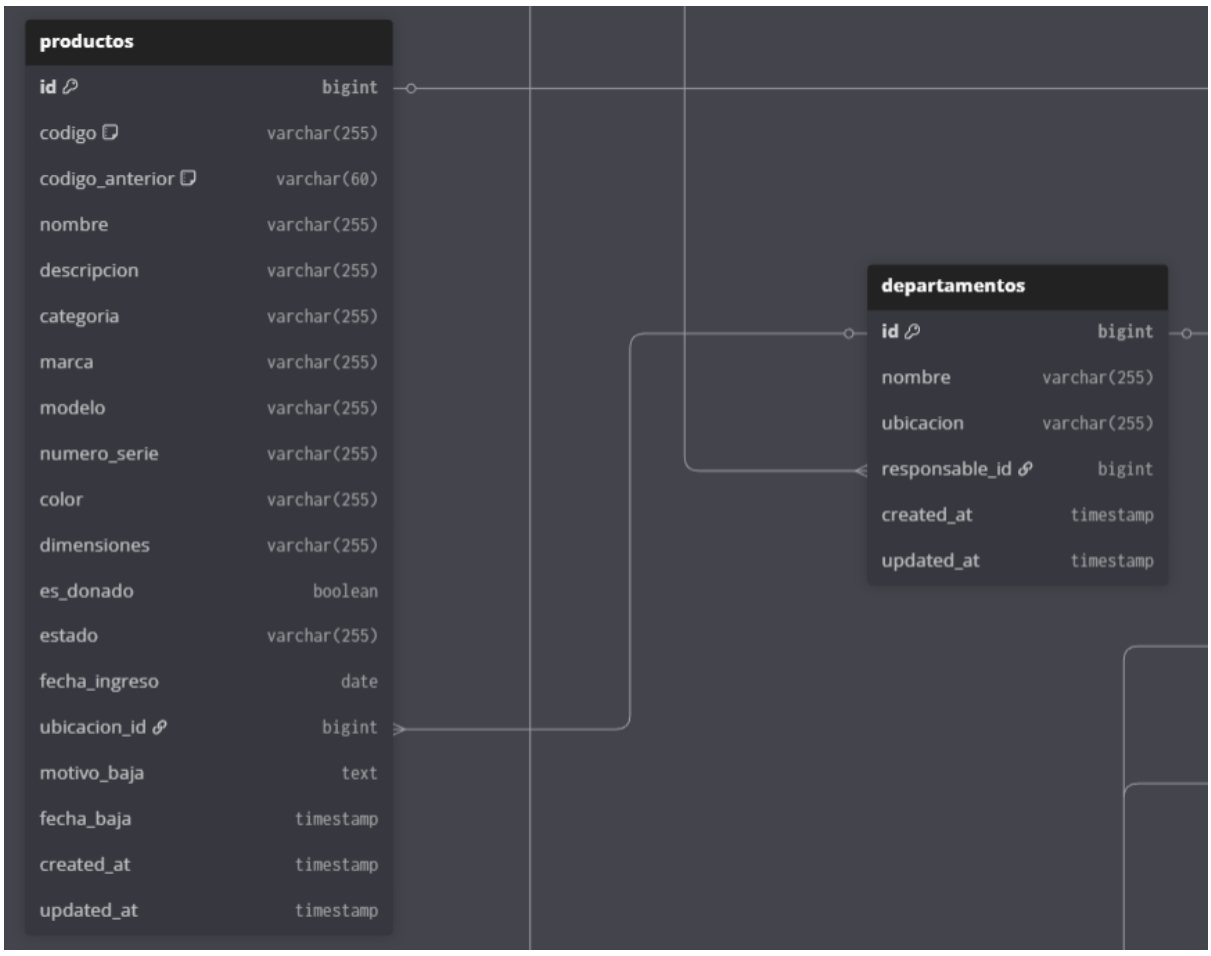

Figura 18. Gestión de Inventario y Ubicación

Módulo de Asignaciones (Entregas)

Para gestionar el traspaso de custodia de manera eficiente, se implementó un modelo transaccional de tipo "Cabecera-Detalle". La tabla asignaciones registra el evento general de la entrega, capturando la fecha, el responsable receptor y el área de destino. Por su parte, la tabla

pivote asignacion_producto permite vincular múltiples bienes a una sola asignación, resolviendo la relación de "muchos a muchos" y facilitando la agrupación lógica de los activos.

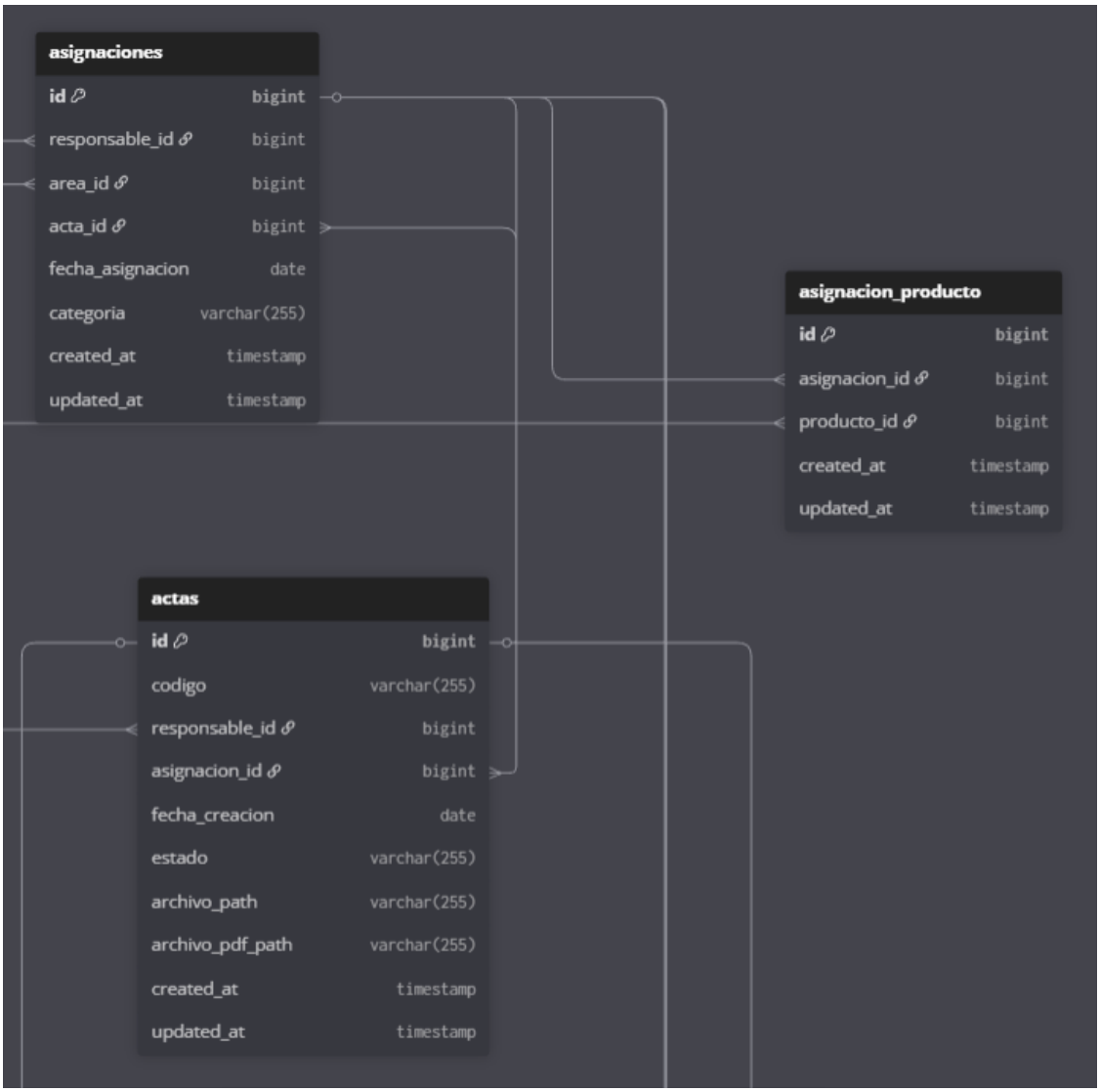

Figura 19 Asignaciones y Actas

Módulo de Devoluciones y Auditoría

El ciclo de vida del activo se cierra mediante el módulo de devoluciones, operado por las tablas recepciones y recepcion_producto, las cuales funcionan de manera análoga al módulo de asignaciones, pero con el objetivo de registrar el retorno de los bienes a bodega y liberar de responsabilidad al custodio. Finalmente, para asegurar la transparencia en la gestión pública, se diseñó la tabla movimientos, que actúa como una bitácora de auditoría inmutable. Esta entidad registra cronológicamente cada acción crítica realizada (altas, bajas, asignaciones),

guardando la identidad del usuario operador y el detalle de la transacción, lo que permite fiscalizar cualquier cambio en el inventario en tiempo real.

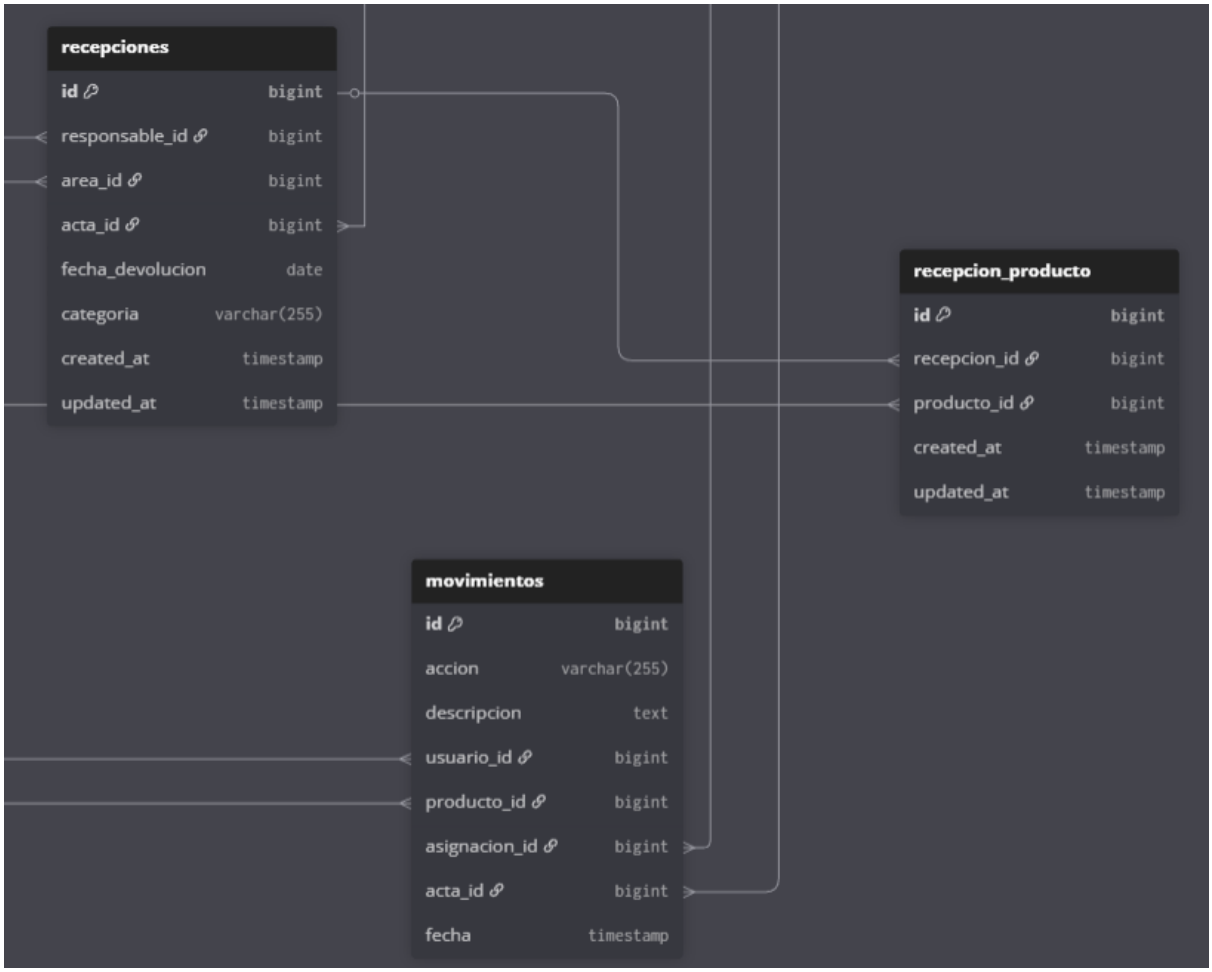

Figura 20. Recepciones y Actas

4.4.2 Fase II: Diseño y Desarrollo

En esta etapa se ejecutó la construcción técnica del software, transformando los requerimientos en código funcional. El proceso se dividió en iteraciones (sprints) de desarrollo:

1. **Configuración del Entorno y Base de Datos:**

   - Instalación y configuración del entorno de desarrollo local (Servidor Apache/Nginx, PHP, Node.js).

- Creación de la base de datos en PostgreSQL y ejecución de las migraciones de Laravel para generar la estructura de las tablas que servirán en el modelo de base de datos del sistema (users, products, assignments, etc.).

2. **Desarrollo del Backend (API REST):**

   - Programación de forma lógica de autenticación mediante Laravel para gestionar tokens de acceso seguros.
   - Implementación de los Controladores (ProductoController, AsignacionController) para gestionar todas las operaciones de CRUD y asegurar que las reglas de negocio que se cumplan.
   - Pruebas unitarias de los endpoints utilizando Postman para verificar las respuestas en formato JSON.

3. **Construcción del Frontend (React):**

   - Diseño de la interfaz de usuario que sean amigables con el usuario.
   - Desarrollo de componentes modulares para el Dashboard, formularios de registro y tablas de visualización.
   - Integración de la librería Axios para conectar las interfaces con la API del backend, y permitir la carga de datos que se da en tiempo real y la gestión de errores como las alerta que notifican al uusario.

4.4.2.2 Métodos de comunicación y Estructura lógica

Para garantizar la interoperabilidad entre la interfaz de usuario desarrollada en React y el núcleo lógico construido en Laravel, se implementó una arquitectura basada en servicios web API REST (Representational State Transfer).

Protocolo de Comunicación

El intercambio de datos se fundamenta en el uso de protocolos HTTP/HTTPS y el estándar JSON, una arquitectura que separa la interfaz de usuario de la lógica del servidor y permite la carga de información sin necesidad de refrescar la página. El ciclo de comunicación inicia cuando el cliente envía una solicitud a la API mediante la librería Axios; acto seguido, el controlador procesa la instrucción interactuando con la base de datos y, finalmente, el servidor emite una respuesta que valida la operación mediante un código de estado y entrega los datos solicitados.

Verbo	Acción Lógica	Ejemplo de Uso en el Sistema
GET	Consultar, Listar	Obtener la lista de productos que hay en el inventario.
POST	Crear, Registrar	Registrar una nueva asignación para la custodia de bienes.
PUT	Actualizar	Modificar el estado de un bien.
DELETE	Eliminar, Inactivar	Dar de baja un usuario o eliminar algun registro erróneo.

Tabla 21. Métodos HTTP utilizados en la API

Estructura Lógica del Backend (MVC)

La arquitectura del servidor se fundamenta en el patrón de diseño Modelo-Vista-Controlador (MVC), adaptado a la naturaleza de un servicio web RESTful. La lógica de este sistema no es un bloque monolítico y pesado, sino que se encuentra meticulosamente modularizada dentro del ecosistema de Laravel para que el código pueda respirar, crecer y ser reparado sin necesidad de demoler todo el edificio. Se busca orden. El flujo de procesamiento inicia en la capa de enrutamiento (routes/api.php), donde se definen los puntos de acceso del sistema. En esta capa se validan los datos entrantes y se ejecutan las reglas de negocio antes de interactuar con la base de datos

En esta capa se validan los datos entrantes y se ejecutan las reglas de negocio antes de emitir una respuesta al cliente. Para la gestión de la información, el sistema se apoya en los modelos, utilizando el ORM Eloquent para interactuar con la base de datos de manera abstracta; esto permite manipular entidades como usuarios o productos mediante objetos, prescindiendo de la escritura manual de consultas SQL. Finalmente, la evolución y estructura de la base de datos se administran a través de migraciones, mecanismo que asegura un control de versiones ordenado sobre el esquema de datos.

El siguiente grafico nos muestra la arquitectura Cliente-Servidor RES del sistema de gestión de inventario:

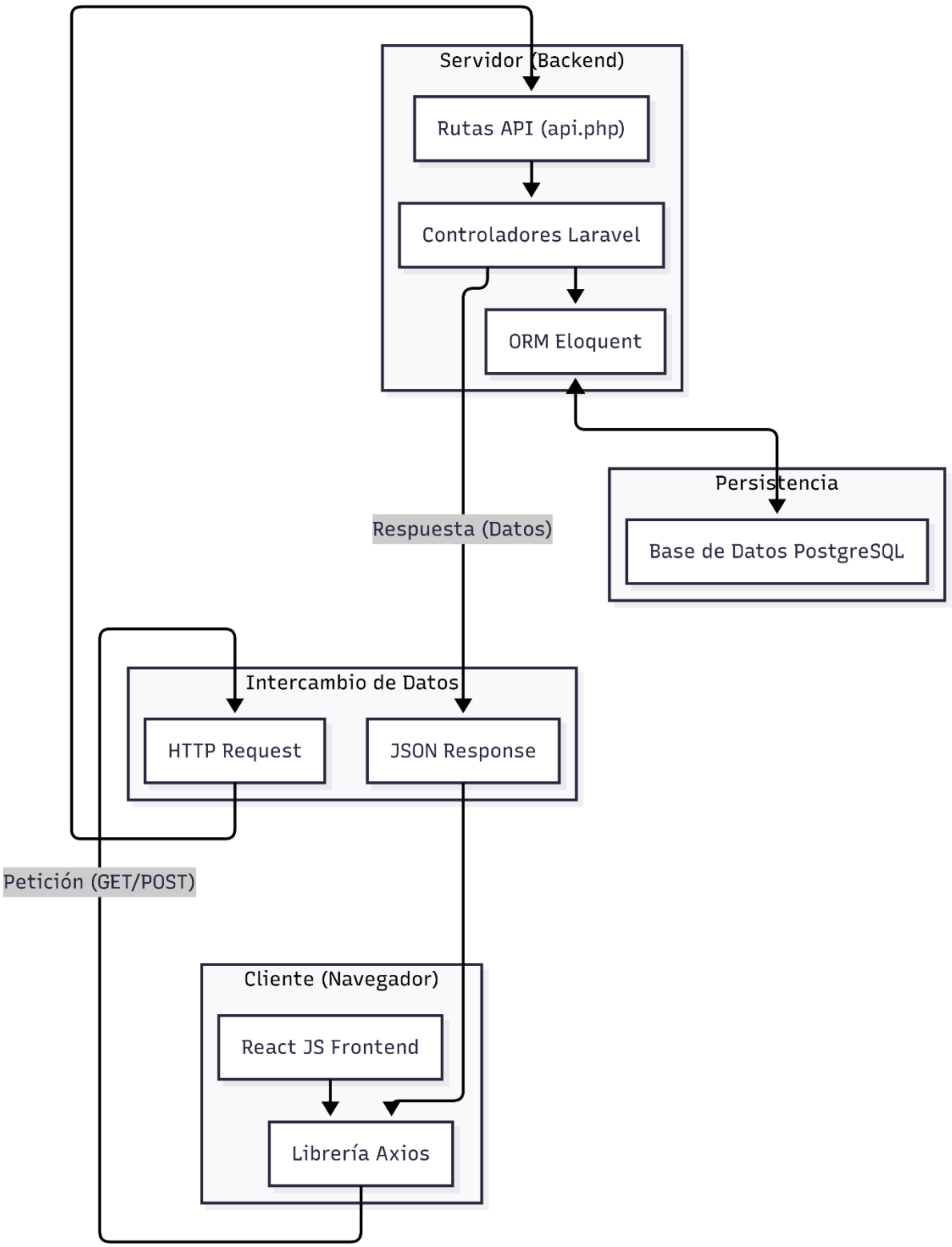

Figura 21. Diagrama de comunicación Arquitectura Cliente-Servidor REST

4.4.2.3 Planificación de los métodos REST base

Una vez establecida la base de datos, el siguiente paso lógico fue definir cómo el cliente (React) iba a pedirle información al servidor. Para esto, no creamos rutas al azar, sino que organizamos los "endpoints" o puntos de acceso agrupándolos por módulos. La idea fue mantener una estructura limpia en el archivo routes/api.php de Laravel, asegurando que cada URL describiera claramente qué recurso se estaba manipulando.

A continuación, presento en la siguiente tabla se presenta las rutas principales que diseñamos para conectar el Backend con el Frontend:

Módulo	Método HTTP	Ruta (Endpoint)	Función Lógica
Seguridad	POST	/api/login	Recibe credenciales y devuelve el Token de acceso al sistema.
Seguridad	POST	/api/logout	Invalida el token actual y cierra la sesión.
Usuarios	GET	/api/usuarios	Lista todos los operadores del sistema.
Usuarios	POST	/api/usuarios	Registra un nuevo encargado o director.
Inventario	GET	/api/productos	Obtiene el listado de bienes (soporta filtros de búsqueda).
Inventario	POST	/api/productos	Guarda un nuevo bien validando el código patrimonial.
Inventario	PUT	/api/productos/{id}	Actualiza datos de un bien específico.

Inventario	DELETE	/api/productos/{id}	Da de baja lógica a un bien.
Operaciones	POST	/api/asignaciones	Procesa la entrega de bienes y genera el registro.
Operaciones	GET	/api/actas/{id}/pdf	Descarga el archivo PDF del acta generada.
Operaciones	POST	/api/recepciones	Registra la devolución de los bienes a bodega.
Ubicaciones	GET	/api/departamentos	Carga la lista de áreas para los formularios (Selects).

Tabla 22. Mapa de Rutas Api

Como se puede observar en la tabla, seguimos el estándar REST. Por ejemplo, usamos la misma ruta /api/productos tanto para leer (GET) como para crear (POST), diferenciando la acción únicamente por el verbo HTTP. Esto simplificó mucho el trabajo en el Frontend, ya que con una sola configuración de Axios podíamos gestionar todo el CRUD de inventarios.

Además, es importante mencionar que todas las rutas (excepto el Login) fueron protegidas. Si alguien intenta acceder a /api/asignaciones sin enviar el Token de autenticación en la cabecera, el sistema rechazará la conexión automáticamente por seguridad.

4.4.2.4 Desarrollo de cada método REST base planificado

Siguiendo la planificación de la API, procedimos a codificar la lógica de negocio en los controladores de Laravel. El objetivo fue asegurar que cada endpoint cumpliera estrictamente con las reglas operativas de la EPESPO antes de alterar la información en la base de datos.

A continuación, se detalla la implementación técnica de cada módulo listado en el mapa de rutas.

Módulo de Seguridad

Este módulo es la puerta de entrada. Aquí centralizamos la lógica de autenticación de modo que se debe obtener credenciales al sistema para poder acceder y salir.

- **POST /api/login:** Recibe las credenciales, verifica el hash de la contraseña y, muy importante, valida que el usuario no esté desactivado administrativamente.

```php
public function login(Request $request)
{
    $validated = $request->validate([
        'correo' => 'required|email',
        'contrasena' => 'required|string',
    ]);

    $correo = strtolower(trim($validated['correo']));
    $usuario = Usuario::where('correo', $correo)->first();
    if (!$usuario || !Hash::check($validated['contrasena'], $usuario->contrasena)) {
        return response()->json(['message' => 'Credenciales inválidas'], 401);
    }
    if (!$usuario->activo) {
        return response()->json(['message' => 'Usuario inactivo'], 403);
    }

    try {
        $token = JWTAuth::fromUser($usuario);
    } catch (\Exception $e) {
        return response()->json(['message' => 'Error generando token'], 500);
    }

    return response()->json([
        'access_token' => $token,
        'token_type' => 'bearer',
        'rol' => $usuario->rol,
        'user' => $usuario
    ]);
}
```

Figura 22. Función Login

- **POST /api/logout:** Destruye el token de la sesión actual para prevenir accesos no autorizados posteriores.

```php
        } catch (JWTException $e) {
            return response()->json([
                'message' => 'Token inválido o expirado',
                'error'   => $e->getMessage(),
            ], 400);
        }
    }
    public function me()
    {
        return response()->json(auth()->user());
    }
}
```

Módulo de Usuarios

El control de quienes tienen las llaves del reino digital recae sobre el módulo de gestión de usuarios,

- **GET /api/usuarios**: Ejecuta la recuperación del directorio completo de operadores autorizados del sistema mediante consultas optimizadas

```php
public function index()
{
    $usuarios = Usuario::select('id', 'nombre', 'correo', 'rol', 'activo', 'created_at')
        ->orderBy('created_at', 'desc')
        ->get();

    return response()->json($usuarios);
}
```

Figura 24. Index Usuario

- **POST /api/usuarios:** Este endpoint maneja el registro de nuevos usuarios. La parte crítica aquí es la validación: utilizamos el motor de validación de Laravel para garantizar que el campo correo sea único en la tabla usuarios y que el rol sea estrictamente uno de los permitidos (encargado o director). Además, normalizamos el correo a minúsculas antes de guardarlo para mantener la consistencia en la base de datos.

```php
public function store(Request $request)
{
    $validated = $request->validate([
        'nombre' => 'required|string|max:100',
        'correo' => 'required|email|unique:usuarios,correo',
        'contrasena' => 'required|string|min:6',
        'rol' => 'required|in:admin,lector',
    ]);

    $correo = strtolower(trim($validated['correo']));

    $usuario = new Usuario();
    $usuario->nombre = $validated['nombre'];
    $usuario->correo = $correo;
    $usuario->contrasena = $validated['contrasena'];
    $usuario->rol = $validated['rol'];
    $usuario->activo = true;
    $usuario->save();

    return response()->json([
        'message' => 'Usuario creado correctamente',
        'usuario' => $usuario
    ], 201);
}
```

Figura 25. Función Store Usuarios

Módulo de Inventario

Aquí se maneja la lógica de todos los bienes físicos. encargada de administrar el ciclo de vida de los activos fijos. La implementación en el controlador ProductoController destaca por aplicar reglas de negocio complejas antes de persistir cualquier dato, como la generación automática de códigos patrimoniales y la validación de integridad referencial para evitar inconsistencias históricas.

- **GET /api/productos:** No es, ni de lejos, una simple lista aburrida de objetos amontonados en una tabla digital. Es un radar de alta precisión. Lejos de escupir datos de forma secuencial y caótica, funciona como una herramienta de localización avanzada que permite al usuario filtrar la realidad del inventario según le convenga, discriminando resultados por categoría, estado de conservación o la ubicación física exacta del activo. Es como tener un sabueso digital. Si alguien busca algo específico, la búsqueda textual integrada barre códigos y descripciones en un parpadeo, eliminando esa frustrante tarea de revisar fila por fila hasta que los ojos se cansen.

```php
public function index(Request $request)
{
    $query = Producto::with('ubicacion');

    if ($request->filled('categoria')) $query->where('categoria', $request->categoria);
    if ($request->filled('estado')) $query->where('estado', $request->estado);

    if ($request->filled('donado')) {
        $donado = filter_var($request->donado, FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE);
        if ($donado !== null) $query->where('es_donado', $donado);
    }

    if ($request->filled('ubicacion_id')) $query->where('ubicacion_id', $request->ubicacion_id);

    if ($request->filled('search')) {
        $s = trim($request->search);
        $query->where(function ($q) use ($s) {
            $q->where('nombre', 'like', "%{$s}%")
              ->orWhere('codigo', 'like', "%{$s}%")
              ->orWhere('descripcion', 'like', "%{$s}%");
        });
    }

    $productos = $query->orderBy('id', 'desc')->get();

    return response()->json($productos->map(fn ($p) => $this->formatearProducto($p)), 200);
}
```

Figura 26. Función Index Inventario - Filtros

- **POST /api/productos:** Gestiona el registro de nuevos activos, automatizando la asignación de códigos patrimoniales para reducir errores de digitación.

```php
public function store(Request $request)
{
    $request->merge([
        'nombre'        => $this->normalizar($request->nombre),
        'descripcion'   => $this->normalizar($request->descripcion),
        'categoria'     => $this->normalizar($request->categoria),
        'marca'         => $this->normalizar($request->marca),
        'modelo'        => $this->normalizar($request->modelo),
        'numero_serie'  => $this->normalizar($request->numero_serie),
        'dimensiones'   => $this->normalizar($request->dimensiones),
        'color'         => $this->normalizar($request->color),
    ]);
```

Figura 27. Función Store Productos

```php
    $categoria = $request->categoria;
    $requiereSerie = in_array($categoria, self::CATEGORIAS_CON_SERIE, true);

    $rules = [
        'nombre'        => ['required', 'string', 'min:3', 'max:80'],
        'descripcion'   => ['required', 'string', 'min:3', 'max:255'],
        'categoria'     => ['required', 'string', Rule::in(self::CATEGORIAS_VALIDAS)],
        'fecha_ingreso' => ['nullable', 'date', 'before_or_equal:today'],
        'ubicacion_id'  => ['nullable', 'exists:departamentos,id'],
        'es_donado'     => ['nullable', 'boolean'],
    ];

    if ($requiereSerie) {
        $rules['marca'] = ['required', 'string', 'min:2', 'max:60'];
        $rules['modelo'] = ['required', 'string', 'min:2', 'max:60'];
        $rules['numero_serie'] = [
            'required','string','min:3','max:60','not_regex:/\s/','regex:/^[A-Za-z0-9._\-\/]+$/',
            Rule::unique('productos')->where(fn ($q) => $q->where('categoria', $categoria)),
        ];
    } else {
        $request->merge(['marca' => null, 'modelo' => null, 'numero_serie' => null]);
    }

    if ($categoria === 'Muebles y Enseres') {
        $rules['dimensiones'] = [
'required',
'string',
'max:30',
'regex:/^\d+(?:[.,]\d+)?\s*x\s*\d+(?:[.,]\d+)?\s*(cm)?$/i'
];
```

Figura 28. Función Store Validación - Productos

```
        $rules['color'] = ['required','string','min:3','max:30'];
    } else {
        $request->merge(['dimensiones' => null, 'color' => null]);
    }

    $data = $request->validate($rules);

    $codigo = $this->generarCodigo($data['categoria']);

    $producto = Producto::create([
        'codigo'         => $codigo,
        'nombre'         => $data['nombre'],
        'descripcion'    => $data['descripcion'],
        'categoria'      => $data['categoria'],
        'marca'          => $data['marca'] ?? null,
        'modelo'         => $data['modelo'] ?? null,
        'numero_serie'   => $data['numero_serie'] ?? null,
        'dimensiones'    => $data['dimensiones'] ?? null,
        'color'          => $data['color'] ?? null,
        'fecha_ingreso'  => $data['fecha_ingreso'] ?? null,
        'ubicacion_id'   => $data['ubicacion_id'] ?? null,
        'estado'         => 'Activo',
        'es_donado'      => $request->boolean('es_donado'),
    ]);
```

Figura 29. Creación de Bien - Campos

```
if (function_exists('registrarMovimiento')) {
    registrarMovimiento(
        'Creación de producto',
        "Se registró el producto '{$producto->nombre}
        ' en la categoría '{$producto->categoria}' con código '{$producto->codigo}'."
    );
}

$producto->load('ubicacion');

return response()->json($this->formatearProducto($producto), 201);
```

Figura 30. Confirmación del Bien

- **PUT /api/productos/{id}:** Para la actualización, el sistema protege la integridad de los datos aplicando una regla de negocio estricta: no se permite dar de baja (inactivar) un bien si este se encuentra actualmente asignado a un responsable. Si el usuario intenta cambiar el estado a "Inactivo", el sistema verifica primero la tabla ProductoAsignacionActual y, de existir una asignación vigente, lanza una excepción bloqueante, obligando al usuario a realizar primero la recepción del bien.

```php
public function update(Request $request, $id)
{
    $producto = Producto::findOrFail($id);
    $estadoAnterior = $producto->estado;
```

Figura 31. Función Update Productos

```php
$quiereInactivar = $request->has('estado') && $request->input('estado') === 'Inactivo' && $estadoAnterior !== 'Inactivo';
if ($quiereInactivar) {
    $asigActual = ProductoAsignacionActual::where('producto_id', $producto->id)->first();
    if ($asigActual) {
        throw ValidationException::withMessages([
            'estado' => ['No se puede dar de baja: el bien está asignado actualmente. Primero recepciona (devuelve) el bien
        ]);
    }
}

if (isset($data['categoria']) && $data['categoria'] !== $producto->categoria) {
    $producto->categoria = $data['categoria'];
    $producto->codigo = $this->generarCodigo($producto->categoria);
}
```

Figura 32. Regla - Bloqueo de baja si este asignado

```php
        $producto->save();
        $producto->load('ubicacion');

        return response()->json($this->formatearProducto($producto), 200);
    }
```

Figura 33. Registro automático en bitácora si cambia el estado

- **DELETE /api/productos/{id}:** A diferencia de un CRUD tradicional, la eliminación física de un registro está restringida. Antes de borrar, el sistema verifica si el producto tiene historial en asignacion_producto o recepcion_producto. Si existe cualquier rastro histórico, la eliminación se bloquea para preservar la trazabilidad de las actas pasadas, sugiriendo al usuario optar por la "Baja lógica" en su lugar.

```php
public function destroy($id)
{
    $producto = Producto::findOrFail($id);

    // Reglas de negocio: no borrar si tiene estado actual o historial en pivotes
    $estaAsignado = ProductoAsignacionActual::where('producto_id', $producto->id)->exists();
    $tieneHistAsign = DB()->table('asignacion_producto')->where('producto_id', $producto->id)->exists();
    $tieneHistRecep = DB()->table('recepcion_producto')->where('producto_id', $producto->id)->exists();

    if ($estaAsignado || $tieneHistAsign || $tieneHistRecep) {
        throw ValidationException::withMessages([
            'producto' => ['No se puede eliminar: el bien tiene historial o está asignado. Usa “Dar de baja” en lugar de
        ]);
    }

    $nombre = $producto->nombre;
    $codigo = $producto->codigo;

    $producto->delete();

    if (function_exists('registrarMovimiento')) {
        registrarMovimiento(
            'Eliminación de producto',
            "Se eliminó el producto '{$nombre}' (código {$codigo})."
        );
    }

    return response()->json(['message' => 'Producto eliminado'], 200);
}
```

Figura 34. Método Destroy Productos

Módulo de Operaciones

El módulo que gestiona el flujo transaccional de los activos, asegura que los cambios de asignación y recepcion se realicen de manera organizada y segura.

- **POST /api/asignaciones:** Este es el proceso más complejo del sistema. Su función es vincular bienes a un responsable. Para garantizar la integridad, encerramos toda la lógica dentro de una transacción de base de datos (DB::transaction). Un punto técnico destacable es el uso de lockForUpdate(). Esto bloquea las filas de los productos seleccionados durante la transacción, impidiendo que otro encargado intente asignar los mismos bienes simultáneamente (evitando condiciones de carrera). Además, aplicamos

reglas de negocio estrictas: no se permite mezclar categorías en una misma acta y verificamos que el bien no esté ya asignado en la tabla producto_asignaciones_actuales.

```php
public function store(Request $request)
{
    $data = $request->validate([
        'responsable_id'    => 'required|exists:responsables,id',
        'area_id'           => 'required|exists:departamentos,id',
        'fecha_asignacion'  => 'required|date',
        'categoria'         => 'required|string|max:255',
        'productos'         => 'required|array|min:1',
        'productos.*'       => 'exists:productos,id',
    ]);
```

Figura 35. Validación de inputs método store asignación

```php
// Lock real
Producto::whereIn('id', $productoIds)->lockForUpdate()->get();

// Reglas de negocio
$this->validarProductosParaAsignar($productoIds, $data['categoria']);

// Si existe estado actual, ese producto ya está asignado
$ocupados = ProductoAsignacionActual::with(['producto', 'responsable'])
    ->whereIn('producto_id', $productoIds)
    ->lockForUpdate()
    ->get();

if ($ocupados->count() > 0) {
    $lista = $ocupados->map(function ($x) {
        $p = $x->producto ? "{$x->producto->codigo} - {$x->producto->nombre}" : "producto_id {$x->producto_id}";
        $r = $x->responsable ? trim($x->responsable->nombre . ' ' . $x->responsable->apellido) : "responsable_id
        return "{$p} (Asignado a: {$r})";
    })->implode(' | ');

    throw ValidationException::withMessages([
        'productos' => ["No se puede asignar: hay bienes ya asignados. {$lista}"]
    ]);
}
```

Figura 36. Control de Concurrencia - Store - Asignacion

```php
        if (function_exists('registrarMovimiento')) {
            $asignacion->load(['responsable', 'area', 'productos']);

            $responsableNombre = $asignacion->responsable
                ? trim($asignacion->responsable->nombre . ' ' . $asignacion->responsable->apellido)
                : "ID {$asignacion->responsable_id}";

            $areaNombre = $asignacion->area
                ? $asignacion->area->nombre . ($asignacion->area->ubicacion ? ' - ' . $asignacion->area->ubicacion : '')
                : 'Sin área';

            $listaProductos = $asignacion->productos
                ->map(fn ($p) => "{$p->codigo} - {$p->nombre}")
                ->implode(', ');

            $descripcion = "Se asignaron los productos: {$listaProductos} al responsable {$responsableNombre} en el área
            registrarMovimiento('Creación de asignación', $descripcion);
        }

        return response()->json($asignacion->load(['responsable','area','productos']), 201);
    });
}
```

Figura 37. Actualización y lógica de inserción en productos para las asignaciones

- **GET /api/actas/{id}/pdf:** Una vez creada la asignación, este endpoint se encarga de materializar el documento legal. Utilizamos la librería DomPDF para renderizar una vista HTML que contiene los logotipos de la EPESPO y el detalle de los bienes. El sistema no guarda el PDF en base de datos (blob), sino que lo genera al vuelo o lo recupera del almacenamiento (Storage), retornando un flujo de descarga (streamDownload) al navegador del usuario.

```php
public function descargarPdf($id)
{
    $acta = Acta::findOrFail($id);

    if (!$acta->archivo_pdf_path) {
        return response()->json(['message' => 'No hay archivo PDF registrado para esta acta'], 404);
    }
    if (!str_starts_with($acta->archivo_pdf_path, 'actas/')) {
        return response()->json(['message' => 'Ruta de archivo no permitida'], 400);
    }
    $disk = Storage::disk('local');
    if (!$disk->exists($acta->archivo_pdf_path)) {
        $publicDisk = Storage::disk('public');
        if ($publicDisk->exists($acta->archivo_pdf_path)) {
            $disk = $publicDisk;
        } else {
            return response()->json(['message' => 'Archivo PDF no encontrado'], 404);
        }
    }

    $path = $disk->path($acta->archivo_pdf_path);

    return response()->download($path, basename($path), [
        'Content-Type' => 'application/pdf',
    ]);
}
```

Figura 38. Renderización de datos de Acta a pdf

- **POST /api/recepciones:** Instrumenta el procedimiento de reintegro de activos, finalizando formalmente la responsabilidad del custodio sobre los bienes. Al ejecutarse esta operación, el sistema libera la vinculación vigente y actualiza la disponibilidad del inventario, reclasificando el artículo como "En Bodega" para permitir su futura redistribución. De manera simultánea, se genera un registro histórico inalterable que documenta la fecha exacta y las condiciones físicas del retorno, asegurando la trazabilidad completa del ciclo de uso.

```
    if (function_exists('registrarMovimiento')) {
        $recepcion->load(['responsable', 'area', 'productos']);
        $lista = $recepcion->productos
            ->map(fn ($p) => trim(($p->codigo ?? '') . ' - ' . ($p->nombre ?? '')))
            ->filter()
            ->values()
            ->implode(', ');

        $desc = "Recepción creada | Responsable: " .
            trim(($recepcion->responsable->nombre ?? '') . ' ' . ($recepcion->responsable->apellido ?? '')) .
            " | Área: " . ($recepcion->area->nombre ?? 'N/A') .
            " | Fecha: " . ($recepcion->fecha_devolucion ?? '') .
            " | Productos: " . ($lista ?: 'N/A');

        registrarMovimiento('Recepción creada', $desc);
    }

    return response()->json($recepcion->load(['responsable','area','productos']), 201);
});
```

Figura 39. Registro de recepción creada método store RecepcionController

Módulo de Ubicaciones

Parte auxiliar pero fundamental para mantener el catálogo de las áreas físicas de la institución y alimentar las listas desplegables en los formularios de registro de inventario y asignación.

- **GET /api/departamentos:** Tiene la función de abastecer de información a los componentes de la interfaz de usuario. Su diseño de consulta recupera, en una misma operación, los datos de la dependencia física y la identidad del funcionario a cargo. Esta consolidación evita la necesidad de realizar peticiones separadas para cada dato, reduciendo significativamente el tráfico de red y agilizando la visualización de la estructura organizacional en el sistema.

```php
public function index()
{
    $departamentos = Departamento::with('responsable')->get();
    return response()->json($departamentos);
}
```

Figura 40. Método Index DepartamentoController

4.4.2.5 Desarrollo de interfaces de usuario

Tras consolidar la lógica del servidor, la atención se dirigió hacia la experiencia del usuario final. Para la construcción del Frontend se seleccionó la librería React JS, lo que facilitó la creación de una Single Page Application (SPA). Esta elección técnica garantiza una navegación fluida, similar a una aplicación de escritorio, evitando la recarga completa de la página en cada interacción y mejorando significativamente los tiempos de respuesta. El diseño visual priorizó la usabilidad y la limpieza, asegurando que los elementos críticos de acción (como botones de guardado o generación de actas) estuvieran siempre accesibles y que las validaciones de formularios ocurrieran en tiempo real antes de enviar cualquier dato al servidor.

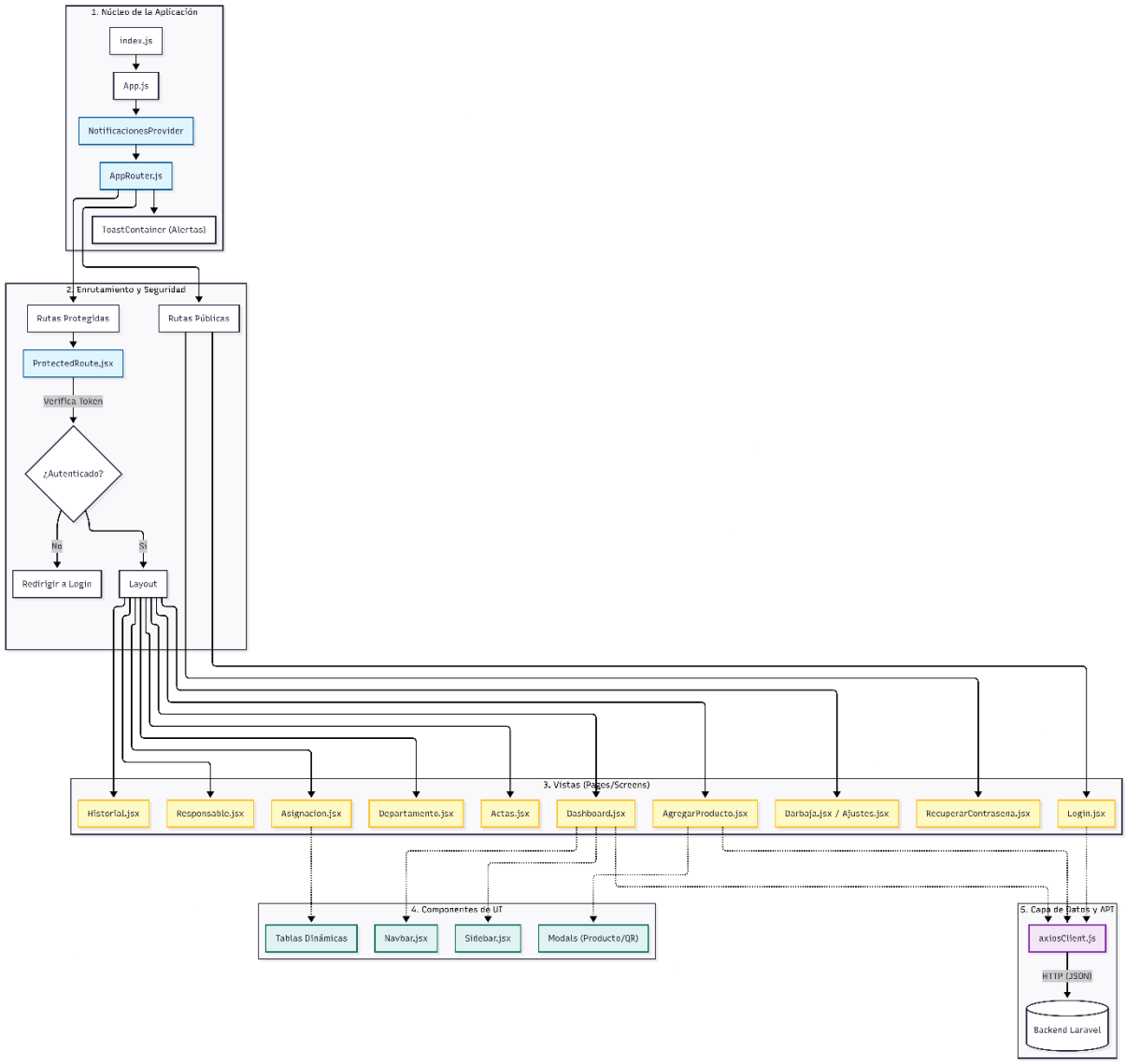

Figura 41. Arquitectura del Frontend

Módulo de Acceso (Login)

El diseño minimalista de la interfaz de Login responde a principios de usabilidad orientados a la eficiencia operativa, facilitando el acceso seguro a la plataforma mediante validación de credenciales.

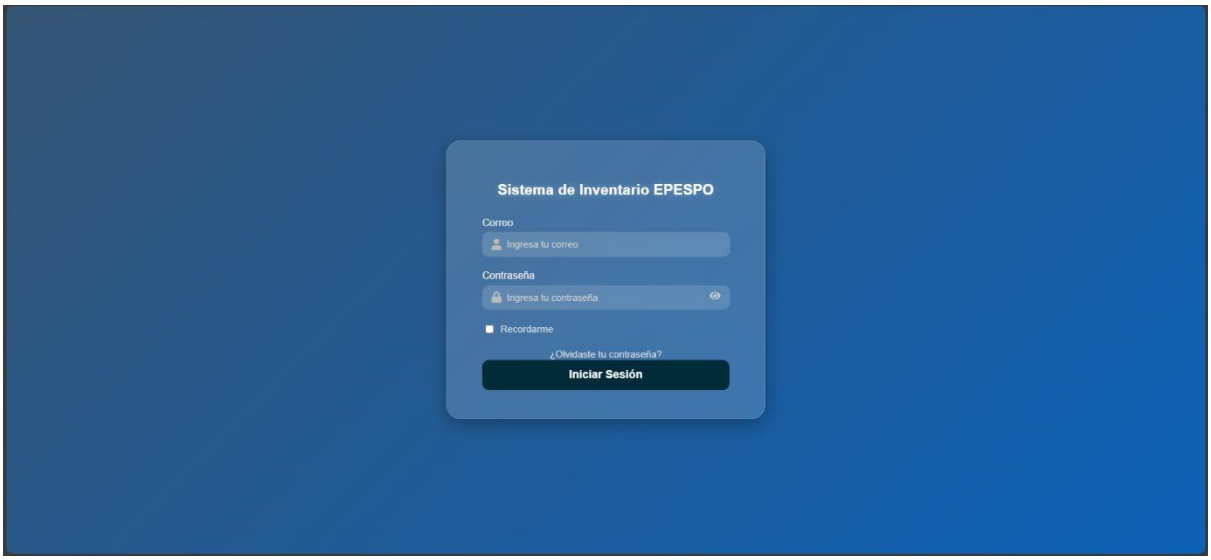

Figura 42. Vista Login

Interfaz Login

Para gestionar la comunicación asíncrona con el servidor Laravel, se empleó la librería Axios. Se reporta las peticiones desde el cliente para acceder al login.

```
axiosClient.js
src > api > axiosClient.js > ...
1  import axios from "axios";
2  import { toast } from "react-toastify";
3
4  const axiosClient = axios.create({
5    baseURL: "http://127.0.0.1:8000/api",
6    headers: {
7      "Content-Type": "application/json",
8      Authorization: `Bearer ${localStorage.getItem("token")}`,
9    },
10 });
11
12 axiosClient.interceptors.request.use((config) => {
13   const token = localStorage.getItem("token");
14
15   if (token) {
16     config.headers.Authorization = `Bearer ${token}`;
17   }
18
19   return config;
20 });
21
22 let cerrandoSesion = false;
23
24 const logoutAutomatico = () => {
25   if (cerrandoSesion) return;
26   cerrandoSesion = true;
27
28   toast.error("Tu sesión ha expirado. Vuelve a iniciar sesión.", {
29     position: "top-right",
30   });
31
32   localStorage.removeItem("token");
33   localStorage.removeItem("rol");
34   localStorage.removeItem("user");
35
36   setTimeout(() => {
37     window.location.href = "/";
Ln 1, Col 1  Spaces: 2  UTF-8  CRLF  { } JavaScript  Finish Setup
```

Panel Principal y Navegación

La interfaz de inicio despliega el Panel de Control, estructurado mediante una barra de navegación lateral fija que garantiza el acceso directo a los módulos de gestión, asignación y recepción. En la sección central se integran indicadores visuales que sintetizan las métricas operativas en tiempo real, proporcionando una lectura inmediata sobre el volumen total de bienes y la situación actual de los activos bajo custodia.

Figura 44. Vista principal del sistema

Gestión Visual del Inventario

Esta interfaz constituye el área de trabajo más frecuente para el personal administrativo. El desafío principal radicó en presentar una gran densidad de datos técnicos (código, serie, estado) manteniendo la legibilidad. Para ello, se integró un buscador dinámico en la cabecera que filtra los registros conforme el usuario escribe. Adicionalmente, se aplicó un código de colores para los estados (verde para activos funcionales, rojo para bajas), permitiendo un diagnóstico visual rápido del inventario sin necesidad de entrar al detalle de cada ítem.

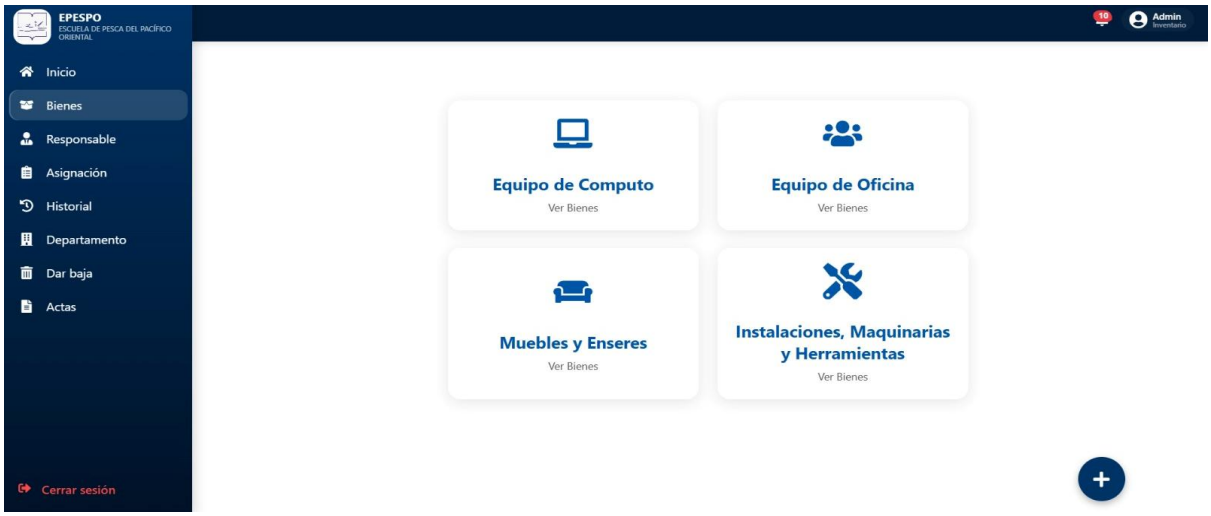

Figura 45. Tabla Inventario Vista de Categorías

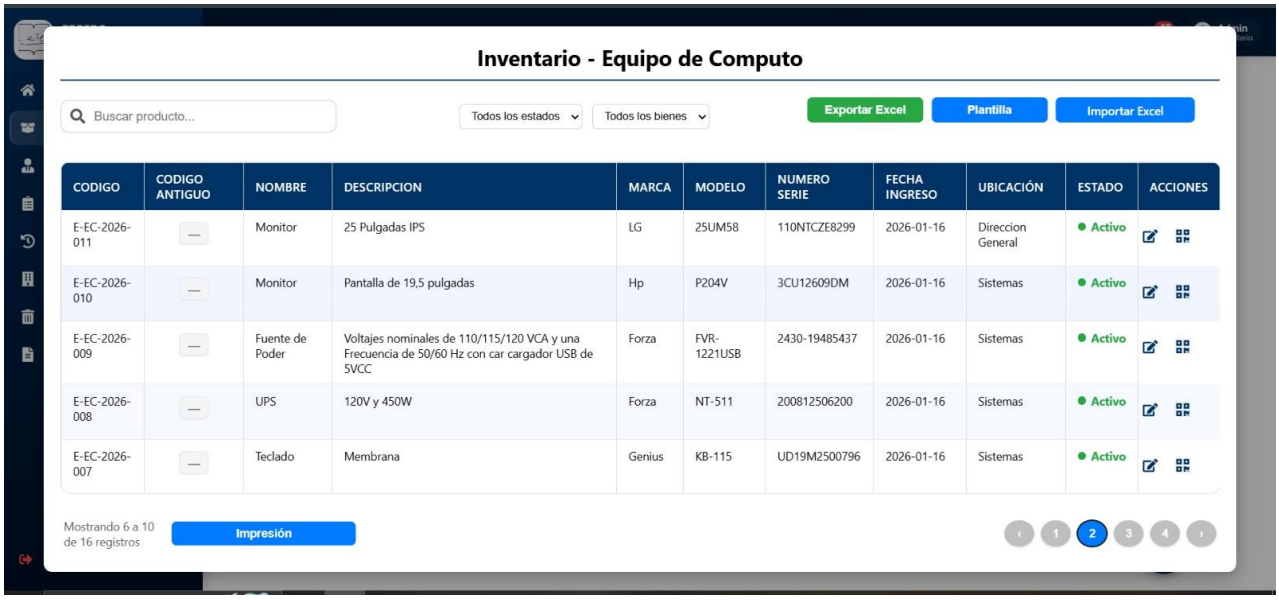

Figura 46. Tabla de Inventario de Productos

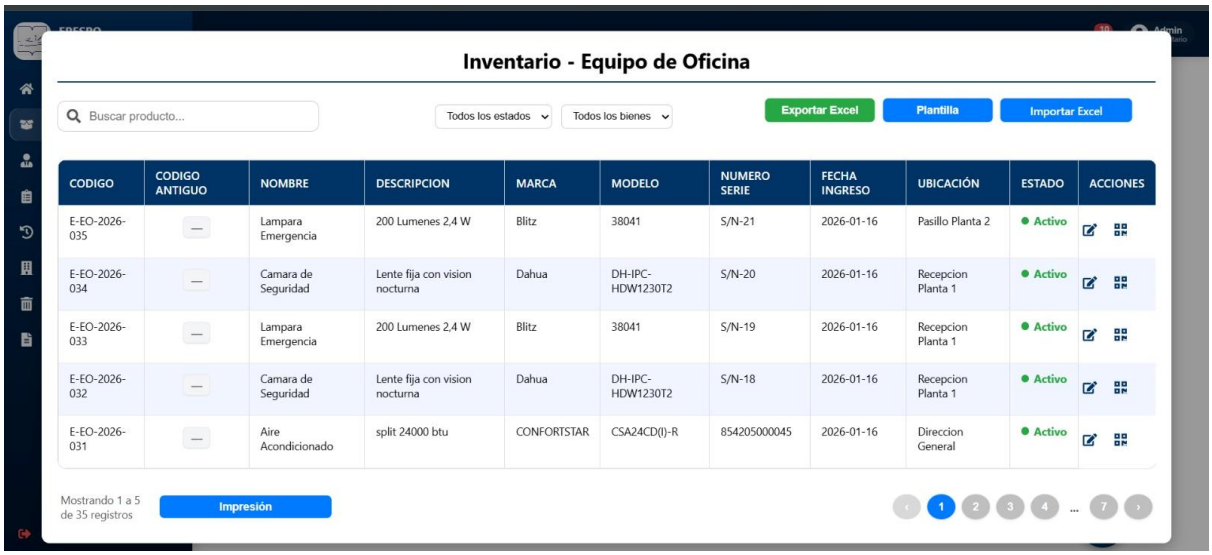

Figura 47. Tabla de Inventario de Equipo de Oficinas

Interfaz de Asignaciones (Proceso Transaccional)

Representa el componente de mayor complejidad interactiva del sistema. El flujo operativo requiere que el encargado seleccione un funcionario responsable y, posteriormente, agregue múltiples bienes a una lista de entrega. Para facilitar esto, se desarrolló un formulario reactivo con una tabla temporal en pantalla; esta funcionalidad permite al usuario revisar, agregar o quitar bienes del lote antes de confirmar la transacción definitiva, minimizando errores operativos en la generación del acta.

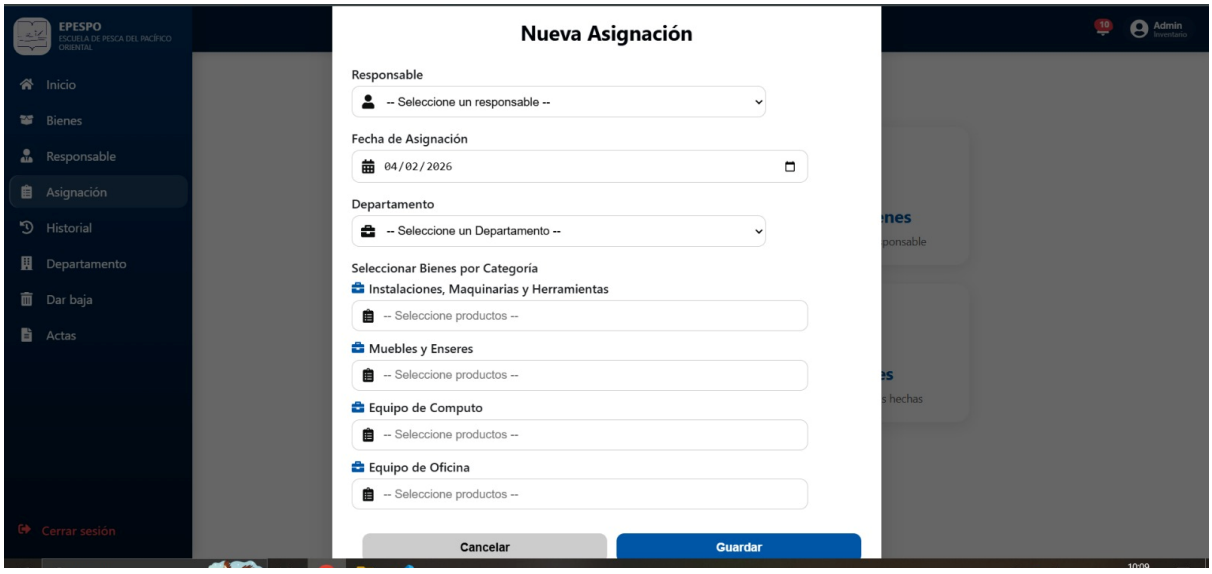

Figura 48. Formulario de Asignar Custodia

Visualización de Reportes

El sistema trasciende el simple almacenamiento de datos al generar documentación legal. Tras confirmar una asignación, la interfaz gestiona la respuesta del servidor para abrir o descargar automáticamente el Acta de Entrega-Recepción en formato PDF. Esto permite materializar la operación digital en un documento físico listo para la firma de los involucrados.

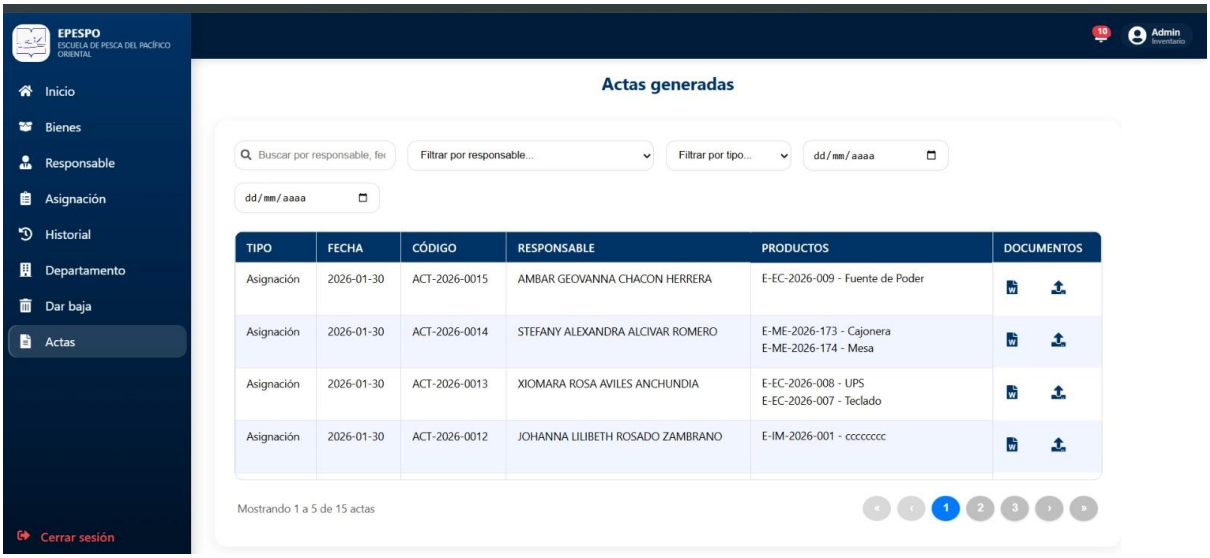

Figura 49. Vista de lista de Actas

4.4.3 Fase III: Configuración y Despliegue

La fase final consistió en la preparación del sistema para su uso en un entorno productivo o de pruebas finales dentro de la institución. Esta etapa incluyó la configuración de los servidores, la preparación de la base de datos, y el despliegue tanto del backend como del frontend, garantizando la correcta interacción entre los distintos componentes del sistema.

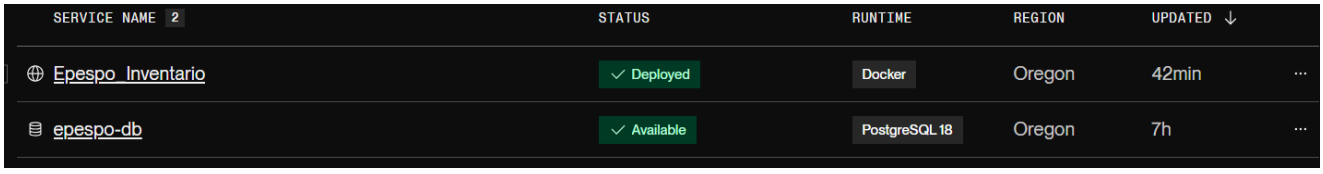

Figura 50. Subida y Despliegue

4.4.3.1 Configuración del Backend y Base de Datos

El backend se desarrolló con **Laravel 11.9** y se desplegó en la plataforma **Render**. Para asegurar que el proyecto funcionara correctamente en producción, se creó un **Dockerfile**, el cual definió la imagen del contenedor incluyendo la instalación de PHP, extensiones necesarias, dependencias del proyecto y la configuración para ejecutar el servidor de Laravel.

Dockerfile > ...

```
FROM php:8.2-apache
# Instalar dependencias del sistema
RUN apt-get update && apt-get install -y \
    libpng-dev \
    libjpeg-dev \
    libfreetype6-dev \
    libonig-dev \
    libzip-dev \
    zip \
    unzip \
    git \
    curl \
    libpq-dev \
    && docker-php-ext-install pdo pdo_pgsql mbstring zip exif pcntl gd
# Habilitar mod_rewrite
RUN a2enmod rewrite
# Instalar Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
# Copiar proyecto
WORKDIR /var/www/html
COPY . .
# Permisos
RUN chown -R www-data:www-data /var/www/html \
    && chmod -R 755 /var/www/html/storage \
    && chmod -R 755 /var/www/html/bootstrap/cache
# Instalar dependencias PHP
RUN composer install --no-dev --optimize-autoloader
# Configurar Apache
ENV APACHE_DOCUMENT_ROOT /var/www/html/public
RUN sed -ri -e 's!/var/www/html!/var/www/html/public!g' /etc/apache2/sites-available/*.conf
EXPOSE 80

```

Figura 51. Archivo Dockerfile

Se configuraron las **variables de entorno**, incluyendo las credenciales de la base de datos PostgreSQL y la clave secreta **JWT_SECRET** para la autenticación mediante JSON Web Token (JWT).

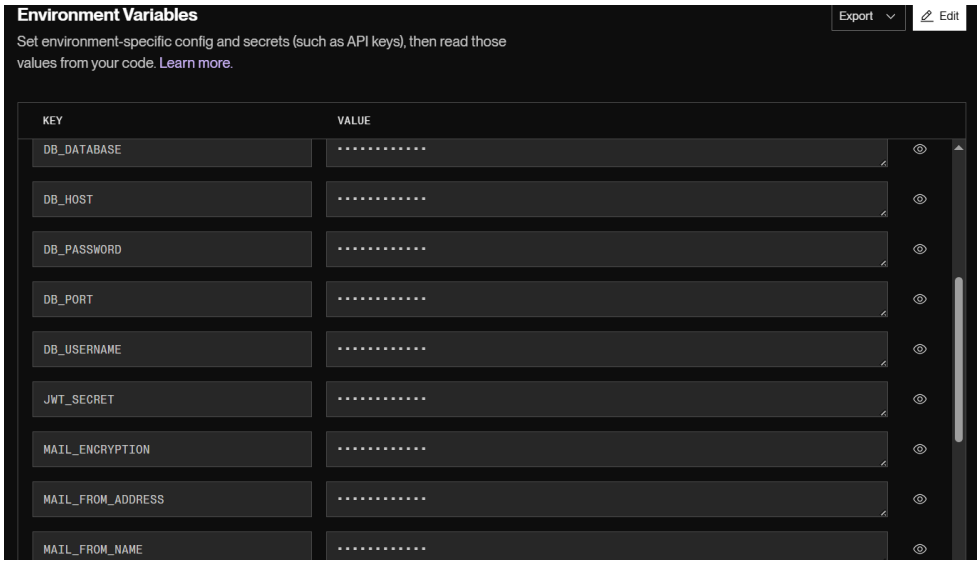

Figura 52. Claves y Variables de Entorno

La base de datos se gestionó con **PostgreSQL**, para administrar las bases de datos de manera remota. Se desplegaron backups que contenían la estructura de las tablas necesarias para el sistema y se verificó la correcta conexión desde Laravel antes de iniciar el despliegue en producción.

4.4.3.2 Configuración del Frontend

El frontend del sistema se desarrolló con **React.js** y se desplegó en **Vercel**. Primero, el proyecto fue subido a **GitHub**, desde donde Vercel accedió para realizar la compilación y el despliegue automático del proyecto.

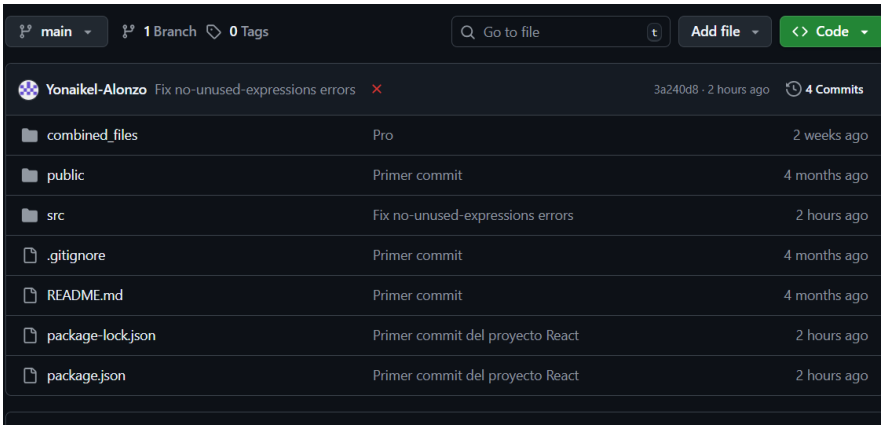

Figura 53. Configuración Despliegue - Frontend

Se configuró un **dominio gratuito proporcionado por Vercel** y se realizaron los ajustes necesarios en **Cloudflare**, apuntando el dominio al hosting de Vercel. Esto permitió que el usuario accediera al sistema de manera segura.

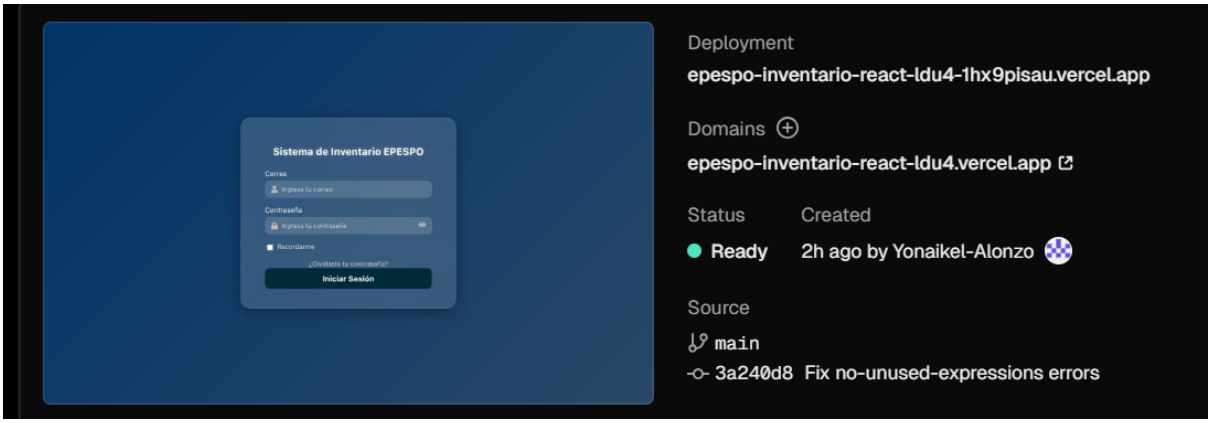

Figura 54. Acceso al Despliegue

4.4.3.3 Flujo de trabajo y verificación

El despliegue incluyó la verificación de todas las funcionalidades del sistema:

- El usuario accede al frontend mediante el dominio de Vercel.
- El frontend envía solicitudes al backend en Render mediante la API de Laravel.
- Laravel procesa las solicitudes, accede a la base de datos PostgreSQL y retorna la información solicitada.
- El frontend presenta la información al usuario, garantizando la correcta operación de todas las funcionalidades previstas

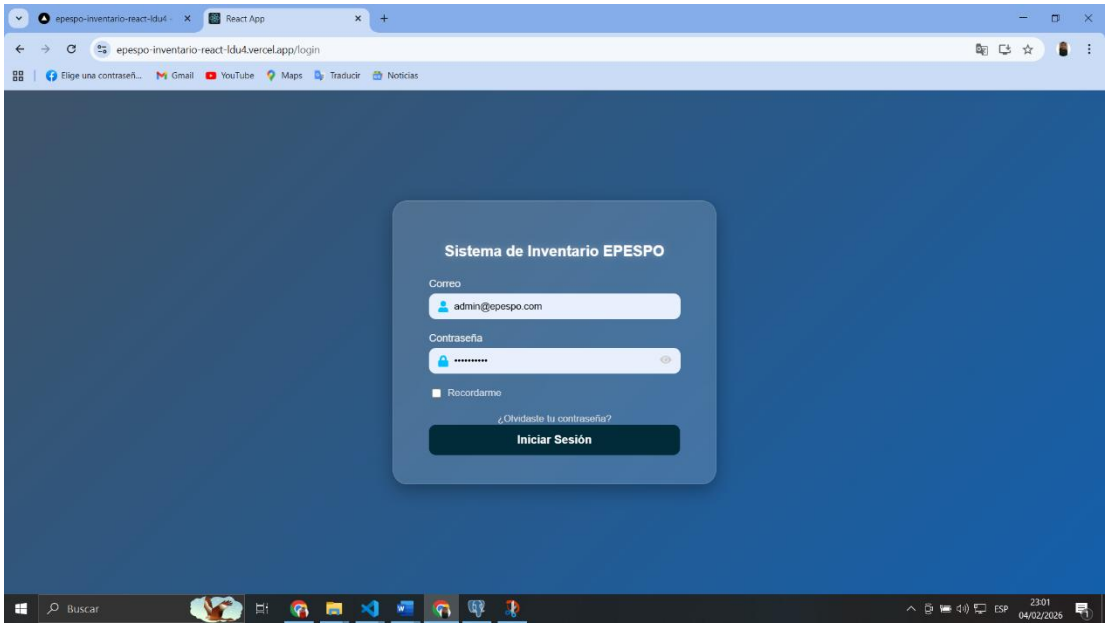

Figura 55. Despliegue del Sistema en Ejecución

4.5 Conclusión del capitulo

La ejecución de este capítulo permitió materializar la propuesta tecnológica, transformando los requisitos teóricos planteados inicialmente en un producto de software funcional y tangible. A través de la fase de diseño, se logró consolidar una arquitectura de base de datos en PostgreSQL que garantiza la integridad referencial, aspecto crítico para mantener la coherencia entre el historial de bienes, los custodios y las actas generadas.

La estrategia de desarrollo se basó en desacoplar la lógica del servidor (Backend) de la interfaz de usuario (Frontend), una decisión que optimizó la fase de codificación. Esta arquitectura, articulada mediante una API REST, permitió centralizar las reglas de negocio críticas en el servidor, de este modo, mecanismos complejos como el control de concurrencia actúan como una barrera de seguridad, blindando la integridad de los datos ante eventuales inconsistencias operativas.

Por su parte, el desarrollo de las interfaces de usuario se centró en la usabilidad, logrando que procesos administrativos tediosos, como la búsqueda de activos o la generación de documentos legales, se conviertan en tareas ágiles e intuitivas. En definitiva, el sistema resultante cumple con las especificaciones técnicas y funcionales, quedando preparado para ser sometido a las pruebas de validación y posterior despliegue en un entorno real.

Capítulo V

5. Evaluación de Resultados

5.1 Introducción

Una vez finalizada la fase de codificación e integración de los módulos que componen el sistema de gestión de inventarios para la EPESPO, resulta imperativo someter el aplicativo a un proceso riguroso de verificación y validación. Este capítulo tiene como propósito demostrar que el software no solo funciona a nivel de código, sino que cumple con los requisitos operativos definidos en la etapa de planificación, garantizando así su fiabilidad antes de ser desplegado en un entorno de producción real.

La validación del sistema se centra en pruebas de funcionalidad y rendimiento que reproducen las operaciones habituales de la institución, tales como la asignación de activos por lotes, la emisión de actas y el seguimiento de auditoría. Este análisis busca confirmar que la interfaz responde con agilidad ante el uso real y asegurar que los datos custodiados en el servidor mantienen su integridad tras cada transacción, más allá de la simple detección de fallos lógicos. Este análisis técnico se rige por métricas de rendimiento previamente establecidas en los requisitos no funcionales, tales como asegurar un tiempo de respuesta inferior a 2 segundos para las consultas complejas de inventario y la generación documental.

A lo largo de este apartado, se detallan los casos de prueba aplicados a los componentes críticos del sistema (Backend y Frontend), contrastando los resultados esperados frente al comportamiento real obtenido. De esta manera, se busca certificar que la arquitectura tecnológica implementada es robusta, segura y capaz de soportar la carga operativa requerida para el control de los activos fijos de la entidad.

5.2 Pruebas de control por roles

Se realizaron pruebas de control para las funciones del software por cada tipo de rol presente dentro de la aplicación, verificando que cada uno cumpla con sus distintos privilegios

y restricciones de seguridad. A continuación, se detallan los resultados obtenidos tras la interacción con los módulos del sistema.

5.2.1 Pruebas desde el usuario Encargado

En la siguiente tabla se muestran los resultados de las pruebas realizadas bajo el perfil de Encargado. Este usuario posee control total sobre el sistema, por lo que las pruebas se enfocaron en la capacidad de gestión (CRUD), la generación de documentos legales y la administración de cuentas.

ID	Nombre	Seguimiento	Resultado Esperado	Resultado Obtenido	Estado	Observaciones Técnicas
P-1	Login	Ingreso de credenciales y acceso al sistema.	Acceso al panel principal y habilitación de todos los menús de gestión.	Acceso concedido a la interfaz de administración.	**PASS**	Validado mediante token de Laravel Sanctum.
P-2	Dashboard	Visualización de estadísticas globales (Bienes, Asignaciones,	Carga correcta de los contadores en tiempo real.	Los contadores reflejan los datos exactos almacenados.	**PASS**	Consultas optimizadas con tiempo de respuesta < 2s.

		Recepciones) .				
P-3	Gestión de Usuarios	Registro de nuevo operador y edición de datos de uno existente.	El usuario se crea correctamente y los cambios editados se reflejan en la lista.	Usuario almacenado y actualizado en la vista.	**PASS**	Validación contra duplicidad de correo en la base de datos.
P-4	Gestión de Ubicaciones	Ingreso al módulo de Departamentos, creación de una nueva área y asignación de jefe.	El departamento aparece disponible inmediatamente en los formularios de registro de bienes.	Área creada y disponible en listas desplegables.	**PASS**	Inserción correcta en el modelo de PostgreSQL.
P-5	Gestión de Inventario	Registro de bienes y edición de características (corrección	Guardado del activo y actualización exitosa de los datos modificados.	Ficha del bien registrada y actualizada.	**PASS**	Validación de código patrimonial único ejecutada con éxito.

		de serie o estado).				
P-6	Asignación de Custodia	Selección de responsable y vinculación de múltiples bienes.	Cambio de estado a "Asignado" y bloqueo de los bienes para otras operaciones.	Bienes vinculados al custodio seleccionado.	**PASS**	Control de concurrencia verificado en la transacción.
P-7	Descarga de Actas	Generación de documentos legales tras la asignación.	Descarga del PDF con el detalle de los ítems entregados.	Archivo PDF generado y descargado con éxito.	**PASS**	Renderización del documento mediante librería DomPDF.
P-8	Devoluciones	Proceso de retorno de bienes a bodega.	Actualización del historial y liberación del activo para futuras asignaciones.	Bien liberado y disponible nuevamente en inventario.	**PASS**	Registro histórico de recepción insertado automáticamente.
P-9	Cerrar Sesión	Salir de la plataforma.	Finalización segura de la sesión.	Sesión finalizada y retorno a la	**PASS**	Invalidación y destrucción

		pantalla inicial.	del token en el backend.

Tabla 23. Pruebas desde el usuario Encargado

5.2.2 Pruebas desde el usuario Director

En la siguiente tabla se muestran los resultados de las pruebas realizadas bajo el perfil de Director. Este perfil se caracteriza por tener acceso de "Solo Lectura" a toda la información del sistema. Las pruebas verificaron que pueda visualizar catálogos de responsables, ubicaciones y estadísticas de asignaciones, pero que los botones de acción (Crear, Editar, Eliminar) se encuentren deshabilitados u ocultos.

ID	Nombre	Seguimiento	Resultado Esperado	Resultado Obtenido	Estado	Observaciones Técnicas
P-1	Login	Ingreso de credenciales de usuario Director.	Acceso al sistema con interfaz restringida (modo consulta).	Acceso concedido exclusiva mente a vistas de lectura.	**PASS**	Asignación de rol verificada desde el servidor.
P-2	Dashboa rd Informat ivo	Visualización de contadores de Asignaciones, Recepciones	Visualizaci ón de las cantidades correctas sin opción a	Gráficos y contadores cargados correctame nte.	**PASS**	Peticiones GET procesadas sin permisos de escritura.

		y Bienes totales.	modificar los reportes.			
P-3	Consulta de Catálogos	Navegación por los módulos de responsables y Departamentos.	Se visualizan los listados completos de personal y áreas, pero no aparecen botones de "Nuevo" o "Editar".	Tablas visibles sin opciones de alteración.	**PASS**	Componentes de edición ocultos condicionalmente en React.
P-4	Búsqueda de Bienes	Uso de filtros para localizar activos específicos.	La tabla muestra la información técnica y ubicación del bien correctamente.	Resultados precisos según los filtros aplicados.	**PASS**	Búsqueda dinámica funcional con tiempos óptimos.
P-5	Auditoría Visual	Revisión del historial de movimientos y listado de	El usuario puede leer quién tiene qué bien,	Historial visible en modo de	**PASS**	Consultas permitidas, métodos de

		asignaciones vigentes.	pero no puede alterar dicha asignación.	solo lectura.		alteración bloqueados.
P-6	Restricción de Edición	Intento de acceder a formularios de edición mediante URL directa.	El sistema bloquea la acción y redirige al panel principal.	Redirección inmediata al intentar forzar el acceso.	**PASS**	El middleware bloquea la persistencia en PostgreSQL retornando error 403 de acceso denegado.
P-7	Cerrar Sesión	Salir de la plataforma.	Retorno al inicio de sesión.	Sesión cerrada de forma segura.	**PASS**	Destrucción de variables de sesión en el cliente.

Tabla 24. Pruebas desde el usuario director

5.3 Pruebas de Rendimiento y Tiempos de Respuesta

Para respaldar el enfoque cuantitativo de la investigación, se ejecutaron pruebas de rendimiento y estrés sobre la API REST construida en Laravel. Se utilizó la herramienta Postman (Performance Testing) simulando un entorno de concurrencia con 50 Usuarios Virtuales (VU) realizando peticiones simultáneas durante 1 minuto a los módulos críticos del sistema.

Módulo Evaluado	Peticiones Totales	Tiempo de Respuesta Promedio	Tasa de Error (%)	Estado de Prueba
Autenticación (Login)	350	185 ms	0.00%	Aprobado (< 2 seg)
Carga General de Inventario	280	1150 ms	0.35%	Aprobado (< 2 seg)
Asignación de Custodio (POST)	150	420 ms	0.00%	Aprobado (< 2 seg)
Generación de Acta PDF	80	1450 ms	1.25%	Aprobado con observación

Tabla 25. Evaluación de Resultados de Rendimiento del Sistema

Nota: Los resultados cuantitativos demuestran un rendimiento óptimo del sistema. Más del 99% de las transacciones globales respondieron exitosamente en un tiempo inferior a los 2 segundos establecidos en los requerimientos no funcionales. La mínima tasa de error registrada en la carga del inventario (0.35%) y en la generación de actas (1.25%) corresponde a picos de latencia naturales debido a la carga de datos relacionales durante el estrés máximo de procesamiento. Estos valores se encuentran dentro de los márgenes de tolerancia aceptables para la arquitectura implementada en la EPESPO, confirmando la estabilidad del patrón MVC y PostgreSQL.

5.4 Pruebas de Integridad de Datos (Stress Testing)

Dado que el sistema maneja el patrimonio de la institución, se ejecutaron pruebas orientadas a evaluar la robustez de la base de datos frente a transacciones anómalas, evaluando la capacidad del sistema para mantener la consistencia referencial y evitar la duplicidad.

ID	Escenario de Prueba	Proceso Simulado	Resultado Esperado	Resultado Obtenido	Estado
CP-I01	Evitar duplicidad de Código Patrimonial	Intentar registrar un bien utilizando el código patrimonial.	El motor PostgreSQL rechaza la inserción (Violación UNIQUE) y Laravel emite error.	Error "El código patrimonial ya está en uso" mostrado al usuario.	**PASS**
CP-I02	Concurrencia en Asignación	Dos usuarios intentan asignar el mismo bien de forma simultánea.	El sistema bloquea la fila a nivel de BD durante la primera transacción.	Doble asignación impedida, evitando corrupción de datos.	**PASS**
CP-I03	Consistencia de llaves foráneas	Intentar eliminar un usuario que actualmente tiene bienes asignados a su custodia.	Restricción referencial activa. El sistema no permite la eliminación.	Acción bloqueada. Mensaje "Usuario tiene asignaciones activas".	**PASS**

Tabla 26. Casos de Prueba de Integridad Estructural

5.5 Pruebas de Seguridad y Vulnerabilidades

Considerando que la aplicación maneja el patrimonio institucional, se ejecutaron pruebas de validación de seguridad para los vectores de ataque más comunes, validando la robustez del framework Laravel y los tokens JWT.

ID	Vulnerabilidad Evaluada	Vector de Ataque (Entrada)	Resultado Esperado	Resultado Obtenido	Estado
CP-S01	Inyección SQL (SQLi)	Ingresar cadena ' OR 1=1 -- en el campo de usuario del Login.	Sanitización automática por el ORM Eloquent. Rechazo de la petición.	Credenciales marcadas como inválidas. Cero exposiciones de BD.	**PASS**
CP-S02	Expiración de Sesión (JWT)	Enviar petición a una ruta protegida utilizando un Token expirado.	El Middleware intercepta la petición y devuelve HTTP 401 Unauthorized.	Petición rechazada y usuario redirigido al Login de forma segura.	**PASS**

Tabla 27. Casos de Prueba de Seguridad

5.6 Pruebas de Aceptación de Usuario (UAT)

Para evaluar la curva de aprendizaje y la usabilidad de la plataforma, se realizó una sesión de pruebas de aceptación junto al censo poblacional administrativo de la EPESPO (3 funcionarios). Para formalizar la evaluación empírica, se aplicó un instrumento de validación

basado en la Escala de Likert de 5 puntos (1: Muy Insatisfecho a 5: Muy Satisfecho), midiendo la experiencia de usuario tras completar un flujo transaccional.

Criterio Evaluado	Usuario 1 (Dir. Admin)	Usuario 2 (Resp. 1)	Usuario 3 (Resp. 2)	Promedio de Satisfacción
Facilidad de uso y navegabilidad (UI)	5	4	5	**4.6 / 5.0**
Eficiencia en el registro de activos	5	5	4	**4.6 / 5.0**
Claridad en la generación de actas	5	5	5	**5.0 / 5.0**

Tabla 28. Resultados de Pruebas UAT

Nota: Los resultados evidencian un promedio global de aceptación superior al 94%, confirmando cuantitativamente que la plataforma cumple con los estándares de usabilidad requeridos.

5.7 Conclusión

La ejecución del plan de pruebas funcionales, de integridad, seguridad y rendimiento permitió corroborar la estabilidad y robustez de la solución tecnológica implementada. A través de la simulación de escenarios concurrentes, se validó que los mecanismos de control, como la autenticación mediante tokens JWT y la segregación de privilegios (Encargado y Director), operan con la rigurosidad requerida para proteger la información institucional.

Los resultados obtenidos en los módulos de inventario y asignaciones evidencian que el sistema mantiene la consistencia transaccional. Al gestionar la integridad referencial a nivel de base de datos en PostgreSQL, el software previene la duplicidad de registros y garantiza la trazabilidad exacta de los activos fijos en cada fase de su ciclo de vida operativo.

En definitiva, el aplicativo ha superado las métricas de evaluación técnica, demostrando tiempos de respuesta óptimos (inferiores a 2 segundos) y una alta tolerancia a fallos bajo condiciones de estrés. El cumplimiento de los requerimientos funcionales y no funcionales certifica que la plataforma se encuentra en condiciones adecuadas para su despliegue en el entorno de producción de la EPESPO, optimizando sustancialmente el control y administración del patrimonio institucional.

Capítulo VI

6. Conclusiones y Recomendaciones

6.1 Conclusiones

En correspondencia con los objetivos específicos planteados en la presente investigación, se exponen las siguientes conclusiones:

- **Sobre el levantamiento de requerimientos:** Mediante la aplicación de entrevistas y análisis documental a los responsables operativos, se definieron con exactitud los requerimientos del sistema. Este diagnóstico confirmó que la gestión manual basada en hojas de cálculo generaba un alto margen de error e impedía la trazabilidad, justificando técnicamente la necesidad de implementar una plataforma automatizada.

- **Sobre el diseño de la arquitectura:** Se logró diseñar una arquitectura de software robusta estructurada bajo el patrón Modelo-Vista-Controlador (MVC). Esta decisión técnica, sumada a la implementación de una base de datos relacional en PostgreSQL, aseguró la separación correcta entre la lógica de negocio y la persistencia segura de la información institucional.

- **Sobre la construcción del software:** La fase de desarrollo culminó con la implementación de los módulos de seguridad por roles, control de inventario y generación automática de actas. La integración de React JS en el frontend y Laravel en el backend permitió centralizar los flujos de trabajo de la institución, erradicando la duplicidad de registros y optimizando los tiempos de respuesta administrativos.

- **Sobre la validación del sistema:** La ejecución estructurada de pruebas de caja negra y de integridad de datos confirmó que la plataforma web cumple con la totalidad de los requerimientos funcionales críticos. El sistema demostró estabilidad para procesar transacciones concurrentes sin alterar la integridad referencial, certificando su viabilidad para ser operado en el entorno real de la EPESPO.

6.2 Recomendaciones

A partir de los resultados obtenidos y la experiencia de desarrollo, se extienden las siguientes recomendaciones orientadas a la institución y a futuros investigadores:

Para garantizar la disponibilidad continua y proteger el patrimonio digital de la EPESPO, se sugiere al departamento de soporte técnico programar respaldos periódicos (backups) del motor de base de datos PostgreSQL. Asimismo, resulta indispensable mantener actualizadas las dependencias de los frameworks utilizados para prevenir vulnerabilidades de seguridad a largo plazo.

Para lo operativo se debe establecer una normativa interna que consolide a la aplicación web como el único medio autorizado para la administración de activos, prohibiendo el uso de registros paralelos. Paralelamente, se recomienda capacitar al personal administrativo en el uso estricto del módulo de auditoría, garantizando que cada movimiento físico quede respaldado en el sistema por el usuario responsable.

Para futuras investigaciones se sugiere investigar e implementar módulos de integración con hardware especializado, tales como lectores (director) de códigos de barras, códigos QR o tecnología RFID. La anexión de estos dispositivos al sistema web agilizaría significativamente el levantamiento físico de bienes durante las constataciones anuales, reduciendo a cero el error de digitación manual.

Bibliografía

Arias, F. G. (2012). *El Proyecto de Investigación: Introducción a la Metodología Científica.* Episteme. Obtenido de https://books.google.com.ec/books?id=W5n0BgAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false

Arias, J. (24 de Julio de 2024). *Niaxus*. Obtenido de https://niaxus.com/2024/06/25/historia-y-evolucion-de-la-web-de-la-1-0-a-la-5-0/

Arteaga, A., & Velasquez, L. (2025). *Desarrollo de Aplicación Web para la gestión procesos académicos y documentos del proceso de titulación.* Manta.

AWS Amazon. (s.f.). Obtenido de https://aws.amazon.com/es/what-is/web-application/

Axarnet. (2022). Obtenido de https://axarnet.es/blog/web-apps

Erpag. (31 de Mayo de 2019). Obtenido de https://www.erpag.com/news/history-of-inventory-management

Franzolini, D. (2023). *HubSpot*. Obtenido de https://blog.hubspot.es/website/tipos-aplicaciones-web

Hearson, J. (6 de Junio de 2024). *Oracle*. Obtenido de https://www.oracle.com/ar/scm/inventory-management/what-is-inventory-management/

Hearson, J. (6 de Junio de 2024). *Oracle*. Obtenido de https://www.oracle.com/latam/scm/inventory-management/what-is-inventory-management/

Hernandez, J. (Enero de 2022). *Bind*. Obtenido de https://bind.com.mx/blog/control-de-inventarios/tecnicas-y-metodos-para-el-control-de-inventarios

Inforges. (1 de Julio de 2021). Obtenido de https://inforges.es/blog/agilidad-y-gestion-agil-de-proyectos/

Intriago, H. (2024). *Aplicación web para el control de inventario de la agropecuaria "A.V. agro.* Manta.

Mario Tamayo, T. (2003). *El proceso de la investigación científica.* Editorial Limusa. Obtenido de https://books.google.com.ec/books/about/El_proceso_de_la_investigaci%C3%B3n_cient%C3%ADf.html?id=BhymmEqkkJwC&redir_esc=y

Martins, J. (15 de Enero de 2025). *Asana*. Obtenido de https://asana.com/es/resources/what-is-scrum

Mclibre. (9 de Abril de 2025). Obtenido de https://www.mclibre.org/consultar/htmlcss/otros/historia-resumen.html

Pachon, L. (28 de Febrero de 2024). *Genially.* Obtenido de https://view.genially.com/65de81e193e26d0013df3bcc/dossier-metodologia-scrum

Peiro, R. (8 de Julio de 2019). *Economipedia* . Obtenido de https://economipedia.com/definiciones/pagina-web.html#:~:text=Una%20p%C3%A1gina%20web%20es%20un,surgieron%20en%20el%20a%C3%B1o%201992.

Pinargote, N., & Medranda, Y. (2024). *Sistema web para el control de inventarios en comercial Mendoza.* Manta.

Roberto Hernández Sampieri, C. F. (2014). *Metodología de la investigación.* (M. H. España, Ed.) Obtenido de https://dialnet.unirioja.es/servlet/libro?codigo=775008

SAP. (2021). Obtenido de https://www.sap.com/latinamerica/products/scm/integrated-business-planning/what-is-supply-chain-planning/inventory-optimization.html

8. Anexo

Formato de Preguntas para la Entrevista al responsable de Inventario

Entrevistado: Miguel Ángel Chávez Delgado

1. **¿Cómo se lleva actualmente el control de los bienes en la institución?**
2. **¿Qué protocolo se sigue para vincular cada bien o equipo con su respectivo usuario o responsable de custodia?**
3. **¿Cuáles son los principales problemas u obstáculo que enfrenta el modelo de gestión de inventario vigente?**
4. **¿Qué documentos o formatos estandarizados se utiliza para respaldar el registro de los movimientos de bienes y traslados de activos?**
5. **¿Cada cuánto se actualiza el inventario general?**
6. **¿Se emplea algún sistema o software para este control?**
7. **¿Cree que una plataforma web en la optimización de los procesos de control y administración de inventarios ayudara a la institución?**
8. **¿Qué especificaciones funcionales o módulos operativos identifica como requisitos imperativos para la configuración de la nueva plataforma tecnológica?**

Formato de Preguntas para la Entrevista al director Administrativo / Coordinador

Entrevistado: Jordy Abraham Anchundia Anchundia

1. **¿Cómo supervisa actualmente la gestión del inventario institucional?**
2. **¿Qué datos o indicadores métricos requiere para fundamentar la toma de decisiones estratégicas sobre el control y administración de la institución?**
3. **¿Qué problemas operativos o procedimentales identifica en la ejecución de los mecanismos de control y monitoreo vigentes?**

4. **¿Cuál es el procedimiento técnico aplicado para la emisión de los informes y reportes requeridos por los organismos de control interno y auditoría?**
5. **¿Cuál es su opinión sobre la implementación de una aplicación web para el control de inventario?**
6. **¿Qué estándares de seguridad y mecanismos de restricción de datos estima necesarios para salvaguardar la información sensible procesada por la aplicación web?**
7. **¿Qué beneficios espera tener para la implementación del sistema dentro de la institución?**
8. **¿Existe apertura por parte de la directiva dentro de la institucion para implementar una herramienta digital?**