UNIVERSIDAD LAICA “ELOY ALFARO” DE MANABÍ
EXTENSIÓN EN EL CARMEN
CARRERA DE INGENIERÍA DE SOFTWARE
Creada Ley No. 10 – Registro Oficial 31 de noviembre 13 de 1985

PROYECTO INTEGRADOR

PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SOFTWARE

Sistema informático con BD Distribuidas para la gestión de inventario de herramientas del vivero de cacao Uleam El Carmen

ZAMBRANO BRAVO ALEX JEANPIERRE

AUTOR

ING. MENDOZA VILLAMAR ROCIO ALEXANDRA

TUTOR

EL CARMEN, AGOSTO 2026

CERTIFICACIÓN DEL TUTOR

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

CERTIFICACIÓN

En calidad de docente tutor(a) de la Extensión El Carmen de la Universidad Laica "Eloy Alfaro" de Manabí, CERTIFICO:

Haber dirigido, revisado y aprobado preliminarmente el Trabajo de Integración Curricular bajo la autoria del estudiante **ZAMBRANO BRAVO ALEX JEANPIERRE**, legalmente matriculado/a en la carrera de Ingeniería en Software, periodo académico 2025(2)-2026(1), cumpliendo el total de 288 horas, cuyo tema del proyecto **Sistema informático con BD Distribuidas para la gestión de inventario de herramientas del vivero de cacao Uleam El Carmen.**

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, y la originalidad del mismo, requisitos 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, El Carmen 3 de agosto del 2026.

Lo certifico,

Ing. Rocio Mendoza Villamar, Mgtr.
Docente Tutora
Área: Ingeniería de Software

TRIBUNAL DE SUSTENTACIÓN

Universidad Laica Eloy Alfaro de Manabí

Extensión El Carmen

Carrera de Ingeniería en Software

TRIBUNAL DE SUSTENTACIÓN

Título del Trabajo de Titulación:

Sistema informático con BD Distribuidas para la gestión de inventario de herramientas del vivero de cacao Uleam El Carmen

Modalidad:

Proyecto Integrador

Autor:

Zambrano Bravo Alex Jeanpierre

Tutor:

Ing. Mendoza Villamar Rocio Alexandra

Tribunal de Sustentación:

- **Presidente:** Ing. Mora Marcillo Alex Bladimir
- **Miembro:** Ing. Reascos Pinchao Raul Saed
- **Miembro:** Ing. López Rodríguez Carlos Vinicio

Fecha de Sustentación:

26 de agosto de 2026

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ
EXTENSIÓN EN EL CARMEN

DECLARACIÓN DE AUTORÍA

La responsabilidad del contenido de este Trabajo de titulación, cuyo tema es: Sistema informático con BD Distribuidas para la gestión de inventario de herramientas del vivero de cacao Uleam El Carmen, corresponde exclusivamente a: Zambrano Bravo Alex Jeanpierre CI.1313026849 los derechos patrimoniales de la misma corresponden a la Universidad Laica "Eloy Alfaro" de Manabí.

Zambrano Bravo Alex Jeanpierre
CI. 131302684-9

DEDICATORIA

Este proyecto se lo dedico, en primer lugar, a Dios, por ser mi fortaleza y guía en cada momento de mi vida. Gracias por darme la salud, la perseverancia y la sabiduría necesarias para llegar hasta aquí, por acompañarme en los momentos de dificultad, iluminar cada paso de mi camino y recordarme siempre que con fe y esfuerzo todo es posible.

A mis padres, quienes con su amor incondicional, paciencia y sacrificios me han brindado el apoyo necesario para alcanzar esta meta. Gracias por ser mi ejemplo de lucha, por creer en mí incluso cuando las circunstancias parecían adversas, por sus palabras de aliento en los momentos más difíciles y por enseñarme que el esfuerzo y la honestidad son los pilares para alcanzar cualquier sueño.

Extiendo también esta dedicatoria a mi familia, que con su cariño y comprensión ha sido una motivación constante para seguir adelante, y a todas las personas que, de una u otra manera, han contribuido con su apoyo, consejos o compañía durante este proceso. A la luna, que, aun en la distancia, estuvo presente en ciertas noches y me brindó su apoyo cuando más lo necesitaba, cada palabra de ánimo, cada gesto de confianza y cada momento compartido ha sido parte fundamental de este logro que hoy celebro.

Zambrano Bravo Alex Jeanpierre

AGRADECIMIENTO

Agradezco profundamente a todos mis docentes por haber compartido su conocimiento, su experiencia y también por dedicación durante mi formación profesional. En especial, a la Ing. Mendoza Villamar Rocio Alexandra Mg., por su valiosa guía como tutor de mi trabajo de titulación y por su constante apoyo y orientación a lo largo de este proceso. Extiendo también mi gratitud a mis compañeros de universidad, quienes con su colaboración y compañerismo hicieron de esta etapa una experiencia enriquecedora y memorable.

Zambrano Bravo Alex Jeanpierre

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

La gestión del inventario de herramientas del vivero de cacao de la Universidad Laica Eloy Alfaro de Manabí, Extensión El Carmen, requiere información confiable y disponible ante la conectividad intermitente del Campus Lastenia. Esta investigación tuvo como objetivo diseñar una arquitectura de base de datos distribuida e implementar su núcleo de sincronización bidireccional para el sistema informático de inventario.

Se desarrolló una investigación aplicada, documental y de campo mediante revisión bibliográfica, una encuesta dirigida a 60 estudiantes y una entrevista al responsable del vivero. La solución utiliza Laravel 12 y MySQL con InnoDB en los nodos Administrador y Central, mientras que para el Nodo Móvil se diseñó una base de datos SQLite con capacidad de operación sin conexión.

En la implementación se integran tablas que administran nodos, versiones, colas y conflictos, también los eventos de dominio recepta HTTP autentificada, reintentos, y repetición de versiones. Se generó 73 prueba en el repositorio general y un total de 322 aserciones, la comunicación fue validada en un ambiente controlado entre los nodos administrado y central; evidenciando como hallazgo principal que la comunicación y sincronización entre los nodos se ejecutan correctamente, incluyendo el manejo de reintentos, versiones y conflictos, sin presentar inconsistencias durante las pruebas realizadas.

Palabras clave: base de datos distribuida; sincronización bidireccional; consistencia eventual; idempotencia; versionado optimista.

ABSTRACT

The management of the tool inventory at the cocoa nursery of Universidad Laica Eloy Alfaro de Manabí, El Carmen Extension, requires reliable and accessible information despite the intermittent connectivity at the Lastenia Campus. This research aimed to design a distributed database architecture and implement its bidirectional synchronization core for the inventory management information system.

An applied, documentary, and field research approach was employed through a literature review, a survey administered to 60 students, and an interview with the nursery manager. The solution uses Laravel 12 and MySQL with InnoDB for the Administrator and Central nodes, while an SQLite database capable of operating offline was designed for the Mobile Node.

The implementation includes tables that manage nodes, versions, queues, and conflicts, as well as domain events such as authenticated HTTP requests, retries, and version repetition. A total of 73 tests were generated in the general repository, along with 322 assertions. Communication was validated in a controlled environment between the managed and central nodes; the main finding was that communication and synchronization between the nodes functioned correctly, including the handling of retries, versions, and conflicts, with no inconsistencies observed during the tests.

Keywords: distributed database; bidirectional synchronization; eventual consistency; idempotency; optimistic versioning.

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Introducción

Actualmente los avances tecnológicos son de gran importancia en todas las áreas es por este motivo que se analiza la situación administrativa del vivero de cacao en el Campus Lastenia de la ULEAM-Extensión El Carmen.

Los registros de la información al presente son manuales o aislados lo que dificulta su localización, y la asignación de responsabilidades del mantenimiento de estos.

El Campus Lastenia presenta conectividad interrumpida, de manera que una aplicación dependiente de un servidor remoto puede dejar de funcionar cuando falla la red. Asimismo, conservar una única base local limita el respaldo y la consulta consolidada. En consecuencia, se requieren nodos capaces de trabajar de manera autónoma y propagar sus cambios al recuperar la conexión.

En este contexto se desarrolla el proyecto “Sistema informático con BD Distribuidas para la gestión de inventario de herramientas del vivero de cacao Uleam El Carmen”. El sistema propuesto integra los procesos de planificación, asignación de tareas, control de inventario, logística y trazabilidad de las operaciones. No obstante, el alcance de esta investigación se concentró en el diseño de la base de datos distribuida y en la implementación del mecanismo de sincronización entre nodos. Para valorar la solución se consideraron criterios relacionados con la integridad y disponibilidad de los datos, la consistencia eventual, la tolerancia a fallos y la interoperabilidad entre diferentes motores de bases de datos.

La aplicación fue desarrollada utilizando **Laravel** bajo una arquitectura de monolito modular, en la que cada módulo funcional es responsable de administrar su propia información y gestionar la lógica asociada a sus procesos. Cuando se produce una modificación relevante, el módulo genera un evento de dominio que es capturado por el componente de sincronización y almacenado en una cola persistente. Una vez que existe disponibilidad de comunicación entre los nodos, estos eventos son enviados automáticamente al nodo de destino correspondiente. Al recibir una solicitud de sincronización, el sistema verifica la autenticidad del nodo emisor, valida la información recibida y la versión del registro involucrado, para posteriormente aplicar los cambios mediante el adaptador correspondiente. Finalmente, el resultado de la operación se registra, permitiendo mantener la trazabilidad del proceso y facilitar las tareas de monitoreo, auditoría y resolución de incidencias.

La arquitectura diseñada está compuesta por un Nodo Administrador con una base de datos MySQL local, un Nodo Central encargado de mantener la información consolidada y un Nodo Móvil planteado con SQLite para trabajar en condiciones de conectividad limitada o inexistente. En esta aplicación se implementó y validó, la comunicación bidireccional entre los nodos Administrador y Central. También se implementó el componente móvil y la migración de los identificadores a UUID quedaron definidas como incrementos posteriores.

1.2 Presentación del tema

El tema comprende el desarrollo de una capa de datos distribuida para un sistema de inventario agrícola. Su unidad tecnológica es el cambio sincronizable: una creación, actualización o eliminación identificada mediante UUID, entidad, versiones y nodo de destino.

La propuesta se basa en una replicación lógica implementada mediante contratos definidos en la aplicación, por lo que no es necesario copiar de manera física las bases de datos entre los distintos nodos. Esta elección facilita la comunicación entre los servidores que utilizan MySQL y la base de datos SQLite instalada en el dispositivo móvil. Además, permite que el sistema continúe funcionando cuando no existe conexión a Internet.

La información modificada se intercambia por medio de sobres de sincronización que no dependen de un motor de base de datos específico. Estos sobres pueden ser creados, enviados y procesados por cualquiera de los nodos que forman parte de la arquitectura: Administrador, Central o Móvil.

El alcance del trabajo comprende el diseño del modelo relacional, la identificación de los nodos, la gestión de la cola de sincronización, la autenticación de las comunicaciones, el control de operaciones duplicadas y los mecanismos de reintento. También se incluyen el manejo de eliminaciones mediante tombstones, la detección de conflictos, el diseño del outbox para el dispositivo móvil, los casos de uso y las pruebas realizadas.

No forman parte de esta investigación el desarrollo completo de las interfaces funcionales, la toma de decisiones relacionadas con el manejo agronómico ni las actividades administrativas que se realizan diariamente en el vivero.

1.3 Ubicación y contextualización de la problemática

El vivero de cacao se encuentra en el Campus Lastenia de la Universidad Laica Eloy Alfaro de Manabí, Extensión El Carmen. En este espacio se desarrollan actividades académicas y productivas que requieren el uso constante de distintas herramientas. Debido a que estos recursos

se trasladan entre las áreas de almacenamiento y las zonas de trabajo, resulta necesario mantener un registro de su ubicación, de la persona responsable y del estado en el que se encuentran.

La conectividad inestable del campus influyó directamente en el diseño de la arquitectura del sistema. Por esta razón, el nodo local fue planteado para que pueda registrar operaciones aun cuando no exista acceso a la red y enviar los cambios pendientes una vez que la conexión se restablezca, y antes de sincronizarse hace una comparación de las versiones disponibles.

1.4 Planteamiento del problema

La gestión manual del inventario de herramientas del vivero de cacao de la Universidad Laica Eloy Alfaro de Manabí, Extensión El Carmen, ocasiona que algunos datos se registren de manera incompleta, tardía o duplicada. Esta situación dificulta determinar con precisión el recorrido de cada herramienta y consultar su historial de préstamos, devoluciones, traslados y cambios de estado.

La información se encuentra distribuida en cuadernos, formularios y otros registros físicos, por lo que no siempre es posible comprobar si las cantidades anotadas coinciden con las existencias disponibles. Estas inconsistencias limitan la confiabilidad de la información y obligan al personal responsable a dedicar más tiempo a localizar las herramientas, verificar quién las utilizó y determinar su estado actual.

Asimismo, el control insuficiente de los préstamos y las devoluciones incrementa el riesgo de pérdidas, retrasos e inconsistencias en el inventario. Las actividades de mantenimiento suelen efectuarse después de que una herramienta presenta daños o deja de estar disponible, debido a que no se cuenta con información actualizada que permita realizar un seguimiento oportuno de su uso y condición.

La conectividad intermitente del Campus Lastenia agrava estas dificultades, ya que impide registrar y actualizar oportunamente las operaciones realizadas en las distintas áreas de trabajo. Como consecuencia, se recurre nuevamente a anotaciones provisionales en papel que posteriormente deben transcribirse, aumentando la posibilidad de pérdida de información, errores, registros duplicados y diferencias entre los datos disponibles.

Estas condiciones afectan la integridad, disponibilidad y trazabilidad de la información del inventario. También dificultan establecer responsabilidades sobre las operaciones realizadas y disponer de datos confiables para controlar las existencias, planificar el mantenimiento y tomar decisiones relacionadas con la administración de las herramientas.

1.4.1 Problematización

La gestión manual de las herramientas en el vivero de cacao de la ULEAM, Extensión El Carmen, ocasiona duplicidad de tareas, pérdida de información y errores en la identificación de los recursos disponibles. Estas dificultades limitan el control de las existencias y generan una administración ineficiente del inventario.

1.4.2 Génesis del problema

Las dificultades que actualmente presenta la gestión del inventario de herramientas en el vivero de cacao de la ULEAM, extensión El Carmen, tienen su origen en diversos factores de carácter operativo y organizacional que han afectado la eficiencia del proceso. Uno de los problemas es el control de inventario que se realiza de manera manual, en cuadernos o documentos físicos, este proceso lleva a cometer errores, duplicidad de registros y pérdida de datos esto hace que no se pueda controlar de una manera eficiente los procesos administrativos.

1.4.3 Estado actual del problema

De acuerdo con las causas identificadas en el análisis anterior, el estado actual del problema en la gestión del inventario de herramientas del vivero de cacao de la Uleam El Carmen presenta las siguientes consecuencias:

La información relacionada con la cantidad, el estado y la ubicación de las herramientas e insumos no siempre se encuentra actualizada ni refleja la situación real del inventario.

Falta de un sistema digital centralizado que cuente con la información actualizada de los procesos que se llevan en el vivero.

1.5 Diagrama causa – efecto del problema

Ilustración 1: Árbol del problema

1.6 Objetivos

1.6.1 Objetivo general

Desarrollar una arquitectura de base de datos distribuida y el mecanismo de sincronización bidireccional del sistema informático de inventario de herramientas del vivero de cacao de la ULEAM El Carmen, para mantener la integridad, trazabilidad y disponibilidad de los datos entre nodos autónomos en condiciones de conectividad intermitente.

1.6.2 Objetivos específicos

- Analizar los fundamentos y requisitos del diseño de bases de datos distribuidas, la operación sin conexión y la sincronización lógica, mediante la revisión bibliográfica y el levantamiento de las necesidades del vivero, para establecer los criterios técnicos de la arquitectura de datos.

- Diseñar el modelo lógico y físico de los datos funcionales y transversales, mediante técnicas de modelado relacional, la definición de la topología de nodos y la especificación del contrato portable de cambios, para garantizar la compatibilidad, integridad y portabilidad de los datos entre MySQL y SQLite.
- Implementar la infraestructura de sincronización mediante eventos, colas durables, adaptadores, autenticación entre nodos, idempotencia, reintentos, tombstones y versionado optimista, para asegurar el intercambio confiable, consistente y trazable de la información en condiciones de conectividad intermitente.
- Demostrar el funcionamiento de los procesos automatizados de la aplicación mediante pruebas unitarias, funcionales y de integración, para verificar la integridad de los datos, la comunicación entre los nodos y la continuidad de las operaciones ante fallos de conectividad.

1.7 Justificación

Actualmente, los procesos de registros de herramientas en el vivero de cacao del Campus Lastenia, extensión El Carmen, de la Uleam tiene múltiples desafíos debido a la dependencia de registros manuales. Estas condiciones pueden ocasionar inconsistencias, pérdida de información, demoras en la localización de herramientas y un control limitado sobre su disponibilidad, ubicación y estado, lo que afecta el desarrollo eficiente de las actividades académicas y productivas del vivero.

Por esta razón, se justifica el diseño de una base de datos distribuida y de un mecanismo de sincronización bidireccional que permita registrar y consultar la información desde los diferentes nodos del sistema. El almacenamiento local permitirá que las actividades continúen aun cuando no se disponga de conexión a Internet. Una vez restablecida la comunicación, los cambios

registrados podrán enviarse al nodo central mediante el proceso de sincronización, sin perder la integridad, el seguimiento ni la coherencia de la información.

La puesta en marcha de esta propuesta facilitará el control automatizado de las herramientas, disminuirá los errores producidos durante el registro y permitirá conservar un historial ordenado y confiable de cada movimiento realizado. Asimismo, facilitará la toma de decisiones, mejorará el seguimiento de préstamos y devoluciones y permitirá aprovechar de manera eficiente los recursos disponibles. De esta forma, el proyecto promoverá una gestión moderna, segura y organizada, acorde con las necesidades operativas del vivero y con los procesos de transformación digital de la Uleam.

1.8 Impactos esperados

1.8.1 Impacto tecnológico

El principal impacto tecnológico es una arquitectura de datos que tolera desconexiones sin abandonar las propiedades transaccionales locales.

La propuesta incorpora además prácticas de ingeniería que pueden comprobarse durante el desarrollo, entre ellas las migraciones versionadas, el uso de UUID en los metadatos distribuidos, los eventos de dominio y la ejecución de trabajos mediante colas. También incluye bloqueos que pueden recuperarse, reintentos con intervalos de espera progresivos, autenticación mediante tokens almacenados de forma segura a través de su hash, control de la cantidad de solicitudes y pruebas de integración. En conjunto, estas prácticas proporcionan una base para que el sistema pueda desplegarse de manera controlada, reproducible y con mecanismos que faciliten su seguimiento.

1.8.2 Impacto social

Contar con información actualizada y confiable contribuye a disminuir los desacuerdos ocasionados por registros inconsistentes o contradictorios, además de facilitar la asignación de responsabilidades con base en evidencia verificable. Los encargados del vivero pueden consultar el historial y el estado de cada herramienta, los estudiantes disponen de un procedimiento estandarizado para el préstamo y devolución de los implementos, y los docentes cuentan con información oportuna que les permite planificar de mejor manera las actividades académicas y prácticas.

Por otra parte, la posibilidad de operar sin conexión a Internet favorece la continuidad de las actividades y amplía el acceso a la tecnología, ya que el registro de la información no depende de una conectividad permanente o de alta calidad.

1.8.3 Impacto ecológico

La incorporación de medios digitales permite reducir el uso constante de formularios impresos y facilita conocer con mayor precisión qué herramientas están disponibles y cuáles requieren mantenimiento. Cuando un recurso puede localizarse con facilidad, recibir atención oportuna y volver a utilizarse, disminuye la necesidad de adquirir reemplazos sin que sean realmente necesarios.

Sin embargo, el sistema no elimina completamente el impacto ambiental asociado con el uso de equipos electrónicos y servicios en la nube. Por ello, se plantea ajustar la infraestructura a las necesidades reales del sistema, aprovechar los dispositivos que todavía se encuentren en buenas condiciones y definir periodos adecuados para conservar respaldos y eliminar información que ya no sea necesaria.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Antecedentes históricos

Los antecedentes de las bases de datos se remontan a la antigüedad, cuando las primeras civilizaciones crearon métodos básicos para almacenar y administrar información. Los registros estadísticos más antiguos se localizaron en asentamientos de Mesopotamia y Egipto, donde se empleaban tablillas de arcilla y papiros para registrar actividades agrícolas, censos de población y transacciones comerciales (Coronel y Morris, 2023).

En el siglo XIX, Herman Hollerith revolucionó el manejo de datos con la invención de la máquina de tarjetas perforadas, que automatizó y aceleró el procesamiento del censo en Estados Unidos. Este avance marcó el inicio de los sistemas de gestión de datos automatizados y sentó las bases para el desarrollo de las bases de datos modernas (Verduzco, 2020).

Desde la década de 1960, las bases de datos han evolucionado desde los sistemas jerárquicos y en red, como el IDS y el IMS, hasta el modelo relacional propuesto por Edgar F. Codd en 1970, que introdujo el lenguaje SQL y transformó la gestión de la información. En las décadas siguientes surgieron sistemas como Oracle, MySQL y PostgreSQL, impulsados por el crecimiento de internet. Con el avance del Big Data y la computación en la nube, tecnologías como Hadoop y servicios como Amazon RDS o Google Cloud SQL permitieron un manejo masivo y escalable de datos. Actualmente, la incorporación de la inteligencia artificial y la tecnología blockchain continúa fortaleciendo la eficiencia y seguridad en la administración de la información (Marqués, 2010).

2.2 Antecedentes de investigaciones relacionadas al tema presentado

El proyecto “Diseño e implementación de un sistema informático para el manejo de inventarios de la Distribuidora Mateo” automatizó el registro de entradas y salidas para reducir errores del control manual. Con la metodología en cascada, OOHDM y herramientas como NetBeans, XAMPP y MySQL, se implementó una solución la cual mejoro la disponibilidad del stock y facilitó la toma de decisiones (Argoti y Portilla, 2018).

La investigación “Gestión de inventarios y su incidencia en la rentabilidad de la cadena de suministros Manaquim de la ciudad de Manta” Reconoció el hallazgo de discrepancias de stock, deficiencias operativas y falta de capacitación. Por medio de entrevistas, observación y análisis FODA, se planteó una guía de procedimientos para estandarizar procesos, capacitar al personal y fijar indicadores de gestión (Marcillo, 2025).

El estudio “Control de inventarios y su incidencia en la rentabilidad de la empresa Fish y Dive de la ciudad de Manta” determinó que las deficiencias en el control afectaban la operación y los resultados económicos. Como solución, planteó un manual de políticas y procedimientos para organizar los registros, asignar responsabilidades y mejorar la medición de la rentabilidad (Gallo, 2025).

2.3 Definiciones conceptuales

2.3.1 Sistema informático con bases de datos distribuidas

Un sistema informático con bases de datos distribuidas integra personas, procedimientos, aplicaciones, datos y recursos tecnológicos para registrar y procesar información en más de un nodo conectado. Su propósito no se limita a almacenar registros, pues también debe mantener reglas de operación, seguridad, comunicación y recuperación que permitan utilizar la información

de manera confiable. En la presente investigación, esta variable comprende la arquitectura del sistema, la persistencia de los datos y los mecanismos utilizados para intercambiar cambios entre nodos cuando la conectividad no es permanente (van Steen y Tanenbaum, 2023).

2.3.1.1 Sistemas de información

Un sistema de información está compuesto por elementos relacionados que capturan datos, los procesan, los almacenan y los convierten en información útil para apoyar las operaciones y la toma de decisiones. En esta estructura intervienen componentes tecnológicos, como hardware, software y bases de datos, junto con personas y procedimientos que determinan cómo se recopila, valida y utiliza la información. Por esta razón, la calidad del sistema depende tanto del funcionamiento técnico como de la definición clara de responsabilidades y reglas para registrar los hechos que ocurren en la organización (Laudon y Laudon, 2022).

La utilidad de un sistema de información se observa cuando sus registros representan de forma oportuna lo que ocurre en el proceso real. Si los datos se encuentran incompletos, duplicados o desactualizados, la información resultante pierde valor para coordinar actividades y decidir. Aplicado al vivero, el sistema debe permitir conocer la existencia, ubicación, estado y responsable de cada herramienta mediante procedimientos uniformes de registro, de manera que la información apoye las labores diarias y no dependa únicamente de formularios dispersos o de la memoria de los usuarios (Laudon y Laudon, 2022).

2.3.1.2 Arquitectura modular y módulos de infraestructura

La arquitectura modular organiza un sistema en componentes con responsabilidades delimitadas y relaciones conocidas. Una separación adecuada busca alta cohesión dentro de cada módulo y bajo acoplamiento entre módulos, de modo que cada parte concentre funciones

relacionadas y pueda modificarse sin producir efectos innecesarios en el resto de la aplicación. Esta organización también favorece la comprensión del código, la reutilización controlada y la realización de pruebas sobre unidades funcionales claramente definidas (Newman, 2021).

Los servicios compartidos y los servicios de sincronización corresponden a responsabilidades transversales que deben mantenerse diferenciadas del dominio operativo. En el proyecto, el módulo Shared puede concentrar autenticación, usuarios, roles, permisos, auditoría y componentes base, mientras que Synchronization puede encargarse de nodos, eventos, colas, transporte, versiones y conflictos. La separación evita duplicar reglas comunes y permite que el intercambio de datos evolucione sin mezclarlo con la lógica de préstamos, devoluciones, traslados o mantenimiento de herramientas (Ford et al., 2021).

2.3.1.3 Bases de datos relacionales

Una base de datos relacional organiza la información en tablas formadas por filas y columnas, en las cuales cada registro representa una ocurrencia y cada atributo describe una característica. Las claves primarias identifican los registros y las claves foráneas establecen relaciones entre tablas, mientras que las restricciones ayudan a impedir valores inválidos o asociaciones incoherentes. El diseño parte de un modelo conceptual, continúa con un modelo lógico y se concreta en un modelo físico adaptado al gestor utilizado (Coronel y Morris, 2023).

Las transacciones permiten tratar varias operaciones relacionadas como una sola unidad de trabajo y se apoyan en las propiedades de atomicidad, consistencia, aislamiento y durabilidad. Estas propiedades reducen el riesgo de guardar cambios parciales ante errores o accesos simultáneos. En MySQL, el motor InnoDB incorpora transacciones, bloqueo y control multiversión; en un diseño que también contemple SQLite, el modelo lógico debe conservar el

mismo significado de entidades, claves y versiones aunque las decisiones físicas se adapten a cada motor (Botros y Tinley, 2021).

2.3.1.4 Bases de datos distribuidas

Una base de datos distribuida administra información almacenada en distintos nodos que cooperan por medio de una red. Aunque los datos se encuentren físicamente separados, el diseño procura mantener una visión lógica coordinada y reglas comunes para identificarlos, consultarlos y modificarlos. La distribución puede responder a necesidades de disponibilidad, proximidad, autonomía o continuidad, pero introduce decisiones sobre localización, replicación, propiedad de los datos y comunicación entre componentes (van Steen y Tanenbaum, 2023).

En un entorno distribuido pueden ocurrir fallos parciales: un nodo puede continuar activo mientras otro no responde o la red puede interrumpirse sin que las bases locales hayan fallado. Por ello, no es suficiente copiar tablas en varios equipos; es necesario definir cómo se detectan las interrupciones, qué operaciones siguen autorizadas, cómo se conservan los cambios pendientes y de qué manera se recupera la coordinación. Estas decisiones implican compromisos entre consistencia, disponibilidad, latencia y complejidad operativa (Ford et al., 2021).

2.3.1.5 Nodos distribuidos y operación sin conexión

Un nodo es una unidad de procesamiento y almacenamiento que participa en el sistema distribuido y posee una identidad dentro de la topología. Cada nodo puede desempeñar funciones diferentes, mantener datos propios y comunicarse con otros mediante mensajes. La autonomía local permite continuar ciertas operaciones cuando otro componente no está disponible; sin embargo, dicha autonomía debe encontrarse limitada por permisos, reglas de negocio y

mecanismos que permitan reincorporar los cambios cuando se restablezca la comunicación (van Steen y Tanenbaum, 2023).

La operación sin conexión requiere persistir los datos necesarios en el dispositivo y conservar de forma durable la evidencia de las operaciones pendientes de transmisión. Guardar los cambios solamente en memoria o en estructuras temporales no protege la información frente a cierres inesperados, recargas o fallos de energía. Un diseño orientado a la desconexión registra primero la operación autorizada en el nodo local y posteriormente coordina su envío, considerando que la comunicación puede fallar, retrasarse o repetirse (Burns, 2024).

2.3.1.6 Sincronización bidireccional

La sincronización bidireccional es el intercambio de cambios en ambos sentidos entre nodos autorizados. Un nodo puede originar una modificación y propagarla hacia otro, mientras que el receptor también puede generar operaciones que deban regresar o distribuirse a los demás participantes. Este proceso no equivale a reemplazar una base completa, sino a comunicar cambios identificables y aplicar reglas que permitan que las copias converjan después de periodos de desconexión (van Steen yTanenbaum, 2023).

Cuando la comunicación es intermitente, las réplicas pueden presentar diferencias temporales. La consistencia eventual admite esa situación siempre que, en ausencia de nuevos cambios y después de procesar los mensajes pendientes, los nodos alcancen un estado compatible. Para lograrlo, la sincronización debe reconocer el origen, el destino, la versión y la identidad de cada operación; además, debe impedir ciclos de reenvío y registrar qué mensajes fueron aceptados, rechazados o quedaron pendientes (Joshi, 2023).

2.3.1.7 Eventos de dominio y contratos de cambios

Un evento de dominio representa un hecho relevante que ya ocurrió dentro del contexto del sistema, como el registro de un préstamo, la devolución de una herramienta o un cambio de estado. Su nombre y contenido deben corresponder al lenguaje del negocio, ya que el evento comunica el significado de la operación y no solamente una instrucción técnica. El uso de eventos permite desacoplar el registro del hecho de los procesos posteriores que lo transportan, auditan o sincronizan (Khononov, 2021).

El contrato de cambios define la estructura portable con la que los nodos interpretan un evento. Puede incluir un identificador único, tipo de entidad, identificador del registro, acción realizada, datos modificados, versión, nodo de origen y fecha de ocurrencia. Mantener un contrato estable reduce la dependencia entre motores de base de datos y módulos internos; cuando la estructura evoluciona, el cambio debe versionarse para conservar compatibilidad con receptores que todavía utilizan una versión anterior (Ford et al., 2021).

2.3.1.8 Colas durables, idempotencia y reintentos

Una cola durable conserva en almacenamiento persistente los mensajes que todavía deben ser procesados o enviados. Cada elemento puede registrar su estado, número de intentos, destino, último error y momento previsto para el siguiente envío. Esta persistencia evita que una interrupción de la aplicación elimine el trabajo pendiente y permite retomar la comunicación desde un estado conocido cuando el nodo vuelve a estar disponible (Shapira et al., 2021).

La idempotencia permite recibir más de una vez el mismo mensaje sin aplicar repetidamente sus efectos. Esta propiedad es necesaria porque una entrega puede completarse aunque la confirmación se pierda, lo que provoca un reintento legítimo. El receptor debe reconocer

la identidad del evento y el contexto de destino antes de modificar los datos. Los reintentos deben espaciarse progresivamente y conservar su historial, pues repetir de forma inmediata una solicitud fallida puede aumentar la carga sin resolver la causa original (Burns, 2024).

2.3.1.9 Versionado optimista, tombstones y gestión de conflictos

El versionado optimista asocia un valor de versión con cada registro y comprueba ese valor antes de aceptar una actualización. Si el dato cambió desde la última lectura, la nueva operación no debe sobrescribirlo de forma silenciosa, sino identificarse como desactualizada o concurrente. En sistemas con varios escritores, los contadores y vectores de versión permiten distinguir una actualización posterior de dos cambios realizados de manera independiente, lo que proporciona una base para detectar conflictos (Joshi, 2023).

Un tombstone es una marca persistente que representa la eliminación de un registro dentro del flujo de replicación. Su conservación temporal impide que otro nodo interprete la ausencia física como un dato que debe recuperarse y permite comunicar la eliminación a participantes que estuvieron desconectados. Cuando se detecta un conflicto, la aplicación necesita una política explícita, como rechazar la versión antigua, priorizar una autoridad definida o enviar el caso a revisión; el criterio no debe depender únicamente del reloj del dispositivo, porque los relojes de nodos diferentes pueden no estar sincronizados (Shapira et al., 2021).

2.3.1.10 Seguridad, control de acceso, auditoría y observabilidad

La seguridad de un sistema distribuido comprende la autenticación de usuarios y nodos, la autorización de las operaciones y la protección de los datos durante su almacenamiento y transporte. La autenticación comprueba la identidad, mientras que la autorización determina qué acciones puede realizar esa identidad. El principio de mínimo privilegio recomienda otorgar solo

los permisos indispensables; asimismo, las entradas y mensajes recibidos deben validarse antes de ser procesados para reducir el riesgo de accesos indebidos, alteraciones o reenvíos maliciosos (Kohnfelder, 2021).

La auditoría conserva evidencia de quién realizó una acción, sobre qué registro, en qué momento y con qué resultado. La observabilidad amplía esta visión mediante eventos estructurados, registros y trazas que permiten comprender el comportamiento del sistema y analizar fallos que no se conocían de antemano. En la sincronización resulta útil registrar envíos, recepciones, reintentos, latencia, errores, versiones y conflictos (Majors et al., 2022).

Las pruebas automatizadas deben comprobar tanto las funciones normales de la aplicación como los escenarios repetidos, fallidos y autorizados que pueden presentarse durante la sincronización. Laravel dispone de herramientas para validar controladores, servicios, eventos, colas y API, lo que permite verificar el comportamiento de los módulos Shared y Synchronization ante diferentes condiciones de ejecución (Stauffer, 2023).

2.3.2 Gestión del inventario de herramientas

La gestión del inventario de herramientas comprende las actividades utilizadas para identificar, registrar, custodiar, ubicar, mantener y controlar los recursos físicos necesarios para la operación. Aunque una herramienta no se consume de la misma manera que una materia prima, su disponibilidad depende de conocer dónde se encuentra, quién la utiliza y cuál es su condición. En esta investigación, la variable se relaciona con la calidad de los registros y con la capacidad de reconstruir los movimientos que afectan las existencias del vivero (Hastings, 2021).

2.3.2.1 Gestión y control de inventarios

La gestión de inventarios planifica y coordina los recursos que una organización necesita para desarrollar sus actividades. Incluye decisiones sobre cantidades, disponibilidad, almacenamiento y reposición, mientras que el control verifica que las operaciones se realicen conforme a procedimientos establecidos. Un inventario útil debe mostrar no solo cuántos elementos existen, sino también su situación operativa y las restricciones que afectan su utilización (Heizer et al., 2023).

El control se apoya en registros de entradas, salidas, ajustes y verificaciones físicas. Cada movimiento debe responder a una responsabilidad definida y dejar evidencia suficiente para explicar las variaciones. En el vivero, estas prácticas permiten administrar las herramientas como activos de trabajo, evitar asignaciones incompatibles y comprobar si las cantidades registradas corresponden con las unidades que se encuentran físicamente disponibles (Hastings, 2021).

2.3.2.2 Identificación y clasificación de herramientas

La identificación asigna a cada herramienta un código estable que la diferencia de las demás y permite relacionar todos sus movimientos con el mismo recurso. Para que el registro sea confiable, los datos maestros deben contener una descripción uniforme, categoría, características relevantes y unidad de control. Una identificación ambigua puede generar duplicados o dificultar la conciliación entre el inventario documental y la existencia física (Hastings, 2021).

La clasificación agrupa las herramientas según criterios útiles para su administración, como tipo, función, frecuencia de uso, ubicación, nivel de riesgo o condición. Estas categorías facilitan la búsqueda, la organización del almacenamiento y la aplicación de controles diferenciados. La clasificación no debe sustituir la identidad individual cuando sea necesario conocer el historial de

una unidad específica; ambos elementos se complementan para mantener orden y trazabilidad (Richards, 2025).

2.3.2.3 Exactitud y disponibilidad de existencias

La exactitud del inventario expresa el grado de correspondencia entre las cantidades registradas y las existencias físicas. Para conservarla se requieren movimientos oportunos, conteos periódicos, conciliación de diferencias y autorización de los ajustes. Cuando una operación se registra tarde o se duplica, el sistema puede mostrar cantidades que no representan la realidad, lo que afecta la planificación y reduce la confianza de los usuarios (Richards y Grinsted, 2024).

La disponibilidad indica si una herramienta puede utilizarse en el momento en que se necesita. Este concepto no depende solamente de que la unidad exista: una herramienta prestada, trasladada, dañada o en mantenimiento no se encuentra disponible para una nueva asignación. Por ello, la información debe integrar cantidad, ubicación y estado, de modo que la existencia registrada represente la capacidad real del vivero para atender sus labores agrícolas (Slack et al., 2022).

2.3.2.4 Préstamos, devoluciones y traslados

El préstamo transfiere temporalmente la custodia de una herramienta a un usuario sin modificar su propiedad, mientras que la devolución restituye esa custodia y confirma la condición en la que se recibe. Ambas operaciones forman un mismo ciclo y deben quedar relacionadas para distinguir los préstamos activos, vencidos o finalizados. El registro debe identificar la herramienta, el responsable y las fechas correspondientes, evitando que una misma unidad aparezca disponible mientras continúa asignada (Hastings, 2021).

El traslado cambia la ubicación de una herramienta y debe conservar la relación entre el lugar de origen y el destino. Al igual que otras transacciones de inventario, necesita una fecha, un responsable y un estado resultante. Registrar estos datos permite explicar por qué una herramienta ya no se encuentra en su ubicación anterior y evita que el movimiento sea interpretado como pérdida. La estandarización de las transacciones facilita la conciliación y la rendición de cuentas (Richards, 2025).

2.3.2.5 Trazabilidad, custodia y localización de herramientas

La trazabilidad es la capacidad de reconstruir el historial de una herramienta a partir de registros enlazados. Para ello, cada evento debe relacionar la identidad del recurso con el usuario, la ubicación, la fecha y el tipo de movimiento. Una secuencia completa permite responder quién utilizó la herramienta, dónde estuvo, cuándo cambió de estado y qué acciones se realizaron durante su vida operativa (Hastings, 2021).

La custodia establece quién posee la responsabilidad temporal sobre el recurso, mientras que la localización indica el sitio físico en el que debe encontrarse. Estos dos datos deben actualizarse después de un préstamo, devolución o traslado. Una codificación clara de áreas y espacios de almacenamiento disminuye el tiempo de búsqueda y evita considerar extraviada una herramienta que se encuentra en una ubicación distinta de la registrada (Richards, 2025).

2.3.2.6 Estado, mantenimiento y vida útil de las herramientas

El estado describe la condición operativa de una herramienta y determina si puede ser utilizada. Categorías como disponible, prestada, en mantenimiento, fuera de servicio o dada de baja deben responder a reglas claras y actualizarse después de cada hecho relevante. Un estado

incorrecto puede ocasionar asignaciones indebidas o impedir el uso de una herramienta que sí se encuentra operativa (Hastings, 2021).

El mantenimiento reúne actividades de inspección, prevención y corrección destinadas a conservar la función del activo y reducir fallos. Registrar fechas, diagnósticos, responsables y trabajos realizados permite planificar intervenciones y conocer el costo y comportamiento del recurso durante su ciclo de vida. La decisión de reparar, sustituir o dar de baja debe considerar la condición, el riesgo, la utilidad y el historial del activo, no solamente la ocurrencia del último daño (Jacobs y Chase, 2021).

2.3.2.7 Indicadores de inventario y toma de decisiones

Los indicadores resumen los registros del inventario para evaluar su comportamiento y apoyar decisiones. Una medida útil debe tener una definición, una fuente de datos, una fórmula y un periodo de análisis conocidos. Los resultados deben compararse con objetivos o valores anteriores, porque una cifra aislada no permite determinar si el proceso mejora, se mantiene o presenta un deterioro (Slack et al., 2022).

En el contexto del vivero pueden analizarse la exactitud entre conteo físico y registro, el porcentaje de herramientas disponibles, los préstamos pendientes, la frecuencia de uso, las pérdidas, el tiempo de búsqueda y la cantidad de herramientas en mantenimiento. Estos indicadores ayudan a priorizar verificaciones, organizar el mantenimiento y justificar reposiciones; sin embargo, su confiabilidad depende de que los movimientos se registren de manera completa y oportuna (Heizer et al., 2023).

2.3.3 Metodología de desarrollo Scrum

Scrum es un marco de trabajo ágil que organiza el desarrollo en ciclos cortos denominados sprints, dentro de los cuales el equipo selecciona trabajo priorizado y produce un incremento evaluable. El marco define responsabilidades, eventos y artefactos que favorecen la transparencia, la inspección y la adaptación. El Product Owner orienta el valor del producto, el Scrum Master facilita la aplicación del marco y los desarrolladores construyen el incremento con base en un objetivo acordado (Tsui et al., 2022).

La aplicación de Scrum permite revisar progresivamente requisitos, diseño, implementación y pruebas en lugar de esperar hasta el final para evaluar todo el producto. La lista priorizada de trabajo puede dividirse en historias y tareas relacionadas con los módulos Shared y Synchronization; al término de cada sprint, la revisión proporciona retroalimentación y la retrospectiva permite ajustar la forma de trabajo. Este enfoque resulta pertinente cuando el producto necesita validar de manera incremental reglas de negocio y comportamientos ante fallos de comunicación (Breyter, 2022).

2.4 Conclusiones del marco teórico

La revisión teórica permitió determinar que un sistema informático con bases de datos distribuidas requiere una arquitectura organizada en nodos capaces de almacenar, procesar e intercambiar información. En contextos con conectividad intermitente, la operación local y la sincronización bidireccional son fundamentales para mantener la disponibilidad de los registros y lograr que los datos converjan cuando se restablezca la comunicación.

Asimismo, se estableció que la confiabilidad de la sincronización depende de mecanismos como las colas durables, la idempotencia, los reintentos controlados, el versionado optimista y la

gestión de conflictos. Estos elementos permiten conservar los cambios pendientes, reconocer operaciones repetidas y evitar que las actualizaciones concurrentes sobrescriban información sin un control previamente definido.

En relación con la gestión del inventario, se concluyó que el control de herramientas necesita registros completos y oportunos sobre su identificación, ubicación, estado, custodia y movimientos. La trazabilidad de préstamos, devoluciones, traslados y mantenimientos facilita la comprobación de las existencias y permite conocer el historial de cada herramienta durante su vida útil.

Finalmente, los fundamentos estudiados respaldan la separación de responsabilidades entre los módulos Shared y Synchronization. El primero concentra los servicios comunes de seguridad, usuarios, permisos y auditoría, mientras que el segundo administra los nodos, eventos, colas, versiones y conflictos. Esta organización proporciona una base teórica coherente para el desarrollo progresivo del sistema mediante Scrum y para la evaluación de su comportamiento ante interrupciones de conectividad.

CAPÍTULO III

3 MARCO INVESTIGATIVO

3.1 Introducción

El presente capítulo expone la metodología utilizada para el desarrollo de la investigación, detallando los métodos, tipos de investigación, técnicas e instrumentos de recolección de datos empleados para obtener información válida y confiable sobre la gestión del inventario de herramientas del vivero de cacao de la ULEAM El Carmen. Asimismo, se describen los procedimientos aplicados para el análisis e interpretación de la información recopilada, permitiendo fundamentar de manera científica el diseño y desarrollo del sistema informático propuesto como solución a la problemática identificada.

3.2 Tipo de investigación

3.2.1 Investigación bibliográfica o documental

La investigación bibliográfica recopila y analiza información de libros, artículos y documentos académicos para construir el marco teórico y contrastar diferentes enfoques sobre el objeto de estudio (Medina et al., 2023).

En este proyecto se utilizó para fundamentar los conceptos relacionados con gestión de inventarios, bases de datos distribuidas, sincronización y sistemas informáticos. La revisión permitió comprender la problemática del vivero e identificar modelos y procedimientos que sustentaron científica y metodológicamente el diseño de la solución propuesta.

3.2.2 Investigación de campo

La investigación de campo permite recopilar información directamente en el lugar donde se desarrolla el problema de estudio, utilizando técnicas como la observación, las entrevistas y las encuestas, sin intervenir ni modificar las variables analizadas. Tarrillo et al. (2024) indican que objetivo es obtener la certeza que facilite la comprensión de la realidad y el análisis objetivo del fenómeno dentro de su contexto específico.

Esta investigación, se llevó a cabo en el vivero de cacao de la ULEAM, extensión El Carmen, con la finalidad de hacer un estudio de los procesos relacionados con el vivero y detectar las necesidades de realizar un sistema digital. La información obtenida permitió conocer de manera precisa la situación actual y sirvió como base para diseñar un sistema informático ajustado a las necesidades reales de la institución y de los usuarios encargados de la gestión del inventario.

3.2.3 Investigación aplicada

La investigación aplicada utiliza conocimientos científicos para desarrollar soluciones prácticas orientadas a necesidades concretas, como herramientas, procesos o productos tecnológicos (Tarrillo et al., 2024).

En esta tesis se empleó para diseñar e implementar la infraestructura de base de datos distribuida y sincronización del sistema de inventario de herramientas del vivero de cacao. La propuesta integra los fundamentos teóricos y los datos obtenidos en campo para responder a las dificultades del control manual y la conectividad intermitente, proporcionando una solución tecnológica ajustada a las necesidades institucionales.

3.3 Métodos de investigación

3.3.1 Método Inductivo

Hadi et al., (2023) indican que el método inductivo es parte del análisis de hechos particulares para formular conclusiones generales e identificar patrones dentro del fenómeno estudiado.

Esta investigación se aplicó en el vivero de cacao de la ULEAM El Carmen, donde se identificaron pérdidas de herramientas, registros incompletos y deficiencias de control. Esta información permitió establecer la necesidad de implementar un sistema informático que mejore la gestión del inventario.

3.3.2 Método Deductivo

El método deductivo es un procedimiento lógico y sistemático que parte de principios generales para explicar situaciones particulares. Sus conclusiones son válidas cuando las premisas también lo son, ya que permite aplicar teorías y leyes previamente establecidas para analizar, interpretar y comprender fenómenos específicos de manera fundamentada (Gómez, 2012).

Se utilizó el método deductivo porque, luego de revisar teorías sobre gestión de inventarios, logística 4.0 y bases de datos distribuidas, estos conceptos generales se aplicaron al caso específico del vivero de cacao. Esto permitió diseñar una solución particular basada en fundamentos teóricos sólidos, transformando el conocimiento existente en elementos concretos del sistema informático, como el control de stock, la trazabilidad y la sincronización de datos.

3.3.3 Método Analítico

El método analítico descompone un fenómeno en sus elementos para examinar sus características, relaciones, causas y efectos (Hadi et al., 2023).

En esta investigación se utilizó para analizar por separado el registro, los préstamos, las devoluciones, el almacenamiento y el control de herramientas del vivero. Esto permitió identificar deficiencias como la falta de trazabilidad y la pérdida de recursos, proporcionando una base para diseñar las funciones del sistema informático.

3.3.4 Método Sintético

El método sintético es un proceso de razonamiento que integra de manera coherente los elementos previamente analizados, reconstruyendo el fenómeno como un todo. El libro señala que este método se desarrolla de manera progresiva, debido a que busca construir una visión integrada capaz de resumir y explicar los distintos elementos que conforman el fenómeno estudiado (Hadi et al., 2023).

En esta investigación, su aplicación permitió vincular la información del inventario con los datos obtenidos mediante las observaciones, encuestas y entrevistas. Esta integración ayudó a comprender el problema de forma global y a plantear una propuesta que reúne el registro, préstamo, control y sincronización de los datos del vivero.

3.4 Fuentes de información

3.4.1 Fuentes primarias

Las fuentes primarias corresponden a la información obtenida directamente del lugar donde ocurre el fenómeno investigado, permitiendo recopilar datos reales y específicos sobre la problemática estudiada. Según Barraza (2023), estas fuentes proporcionan información original obtenida mediante técnicas de investigación aplicadas directamente a los participantes o al contexto de estudio.

En la presente investigación se utilizaron como fuentes primarias la entrevista realizada al responsable del vivero de cacao y la encuesta aplicada a los estudiantes de vinculación de la carrera de Agropecuaria que hacen uso de las herramientas e instalaciones del vivero.

3.4.2 Fuentes secundarias

Las fuentes secundarias corresponden a información obtenida de documentos previamente elaborados, tales como libros, artículos científicos, tesis, revistas académicas y documentos digitales, los cuales permiten sustentar teóricamente la investigación. Según Barraza (2023), estas fuentes facilitan la recopilación de antecedentes, conceptos y fundamentos científicos relacionados con el tema de estudio.

En esta investigación se utilizaron libros, tesis, artículos científicos y documentación académica relacionados con sistemas informáticos, bases de datos distribuidas, gestión de inventarios y metodologías de investigación.

3.5 Estrategia operacional para la recolección de datos

3.5.1 Población – Segmentación – Técnica de muestreo – Tamaño de la muestra

3.5.1.1 Población

La población corresponde al conjunto total de individuos, elementos o unidades de análisis que comparten características relacionadas con el problema investigado. Su adecuada delimitación permite obtener información válida, confiable y representativa, facilitando el cumplimiento de los objetivos planteados y el correcto desarrollo de la investigación científica (Barraza, 2023).

Para efectos de esta investigación se consideró como población objetivo únicamente a los estudiantes y responsables de la carrera de agropecuaria. La población estuvo conformada por 61

participantes, correspondientes a 60 estudiantes y 1 profesor responsable de la carrera, quienes aportaron información relevante para el desarrollo y validación de la propuesta

3.5.1.2 Segmentación

La segmentación consiste en organizar la población en grupos que comparten características vinculadas con la investigación, lo que facilita el ordenamiento de la información y favorece la representatividad de los resultados (Barraza, 2023).

Para aplicar la encuesta se consideró a los 60 estudiantes de vinculación que hacen uso frecuente del vivero de cacao. Asimismo, el responsable del proyecto fue incluido como informante principal mediante una entrevista, debido a su participación directa en el control del inventario.

La población se segmentó de la siguiente manera:

Tabla 1 Segmentación de la población objeto de estudio del vivero de cacao de la ULEAM, extensión El Carmen (período académico 2025-2)

Segmentos	Descripción	Cantidad
Estudiantes	Estudiantes de vinculacion de la carrera de Agropecuaria que utilizan el vivero de cacao	60
Encargado	Responsable del proyecto vinculación con la sociedad (Pedro Eduardo Nivela Morante)	1
Total	Población total objeto de estudio	61

3.5.1.3 Técnica de muestreo

La técnica de muestreo es el procedimiento metodológico que permite seleccionar una parte representativa de la población para obtener información confiable sin estudiar a todos sus elementos (Barraza, 2023).

La técnica que se usó para este proyecto fue el muestreo censal debido a que se consideró la totalidad de los estudiantes que cumplían con los criterios establecidos para la investigación. La población estuvo conformada por 60 estudiantes de la carrera de Agropecuaria, quienes fueron seleccionados bajo los siguientes criterios:

- Estudiantes matriculados en la carrera de agropecuaria
- Estudiantes de practica y vinculación del periodo 2025-2

3.5.1.4 Tamaño de la muestra

El tamaño de la muestra es la cantidad de elementos seleccionados de la población para participar en la investigación. Su determinación es fundamental para garantizar resultados representativos, válidos y confiables, permitiendo generalizar los hallazgos y reducir el margen de error (Hadi et al., 2023).

En la presente investigación se trabajó con los estudiantes seleccionados en la muestra conformada por 60 estudiantes de vinculación y practicas de la carrera de Agropecuaria y el encargado del proyecto de vinculación con la sociedad del vivero de cacao de la Universidad Laica Eloy Alfaro de Manabí (ULEAM), extensión El Carmen. Por lo tanto, la población total estudiada estuvo conformada por 61 participantes.

3.5.2 Análisis de las herramientas de recolección de datos a utilizar

3.5.2.1 Encuesta

La encuesta es una técnica de recolección de información propia de la investigación cuantitativa que utiliza preguntas estructuradas para obtener datos de un grupo de personas. Permite recopilar información de forma sistemática sobre opiniones, comportamientos o

características, facilitando el análisis del problema mediante resultados medibles y comparables (Barraza, 2023).

Se aplicó un cuestionario estructurado con preguntas cerradas, con el objetivo de cuantificar y generalizar las percepciones, hábitos y experiencias de los usuarios del vivero en relación con el préstamo, uso, devolución y control de las herramientas. Esta técnica permitió identificar patrones de uso y problemáticas recurrentes asociadas al manejo manual del inventario.

3.5.2.2 Entrevista

La entrevista es una técnica de investigación cualitativa basada en la comunicación directa entre el investigador y el participante, que permite obtener información profunda sobre experiencias, opiniones y percepciones (Barraza, 2023).

Se realizo una entrevista al encargado del proyecto el Ing. Pedro Eduardo Nivela Morante, con preguntas abiertas que permitieron profundizar en aspectos cualitativos, obteniendo información detallada y contextualizada sobre los procesos administrativos, el control de herramientas y las limitaciones del sistema manual de inventario.

3.5.2.3 Estructura de los instrumentos de recolección de datos aplicados

A. Entrevista

Se aplicó una guía de entrevista de 13 preguntas abiertas al Ing. Pedro Eduardo Nivela Morante, responsable del proyecto de vinculación y del control de las bodegas. El instrumento permitió identificar cómo se gestionan el inventario, los préstamos y las devoluciones, así como las dificultades asociadas al registro manual y los requerimientos necesarios para implementar un sistema informático.

B. Encuesta estructurada

El segundo instrumento de recolección de información fue una encuesta estructurada, conformada por 13 preguntas cerradas y dirigida a los estudiantes de vinculación que realizan actividades en el vivero de cacao. Su objetivo fue recopilar información sobre la forma en que se lleva a cabo el control del inventario, el nivel de organización de las herramientas, los procedimientos de préstamo y devolución, así como conocer la percepción y disposición de los participantes frente a la implementación de un sistema informático que contribuya a optimizar la administración y el control de los recursos del vivero.

3.5.3 Plan de recolección de datos

CRONOGRAMA OPERATIVO			
Técnica	**Enfocado a**	**Responsable**	**Fecha**
Encuesta	Estudiantes de Agropecuaria	Alex Zambrano	12 al 15/11/2025
Entrevista	Responsable del proyecto de vinculación con la sociedad del vivero de cacao	Alex Zambrano	12/11/2025

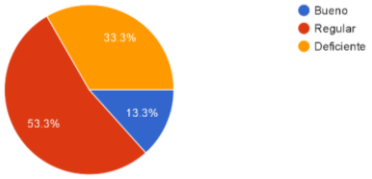
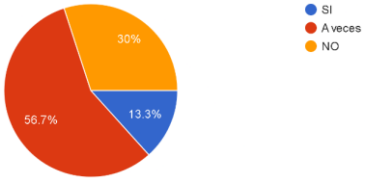
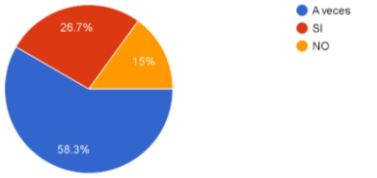
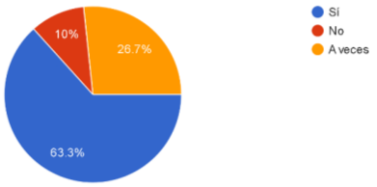
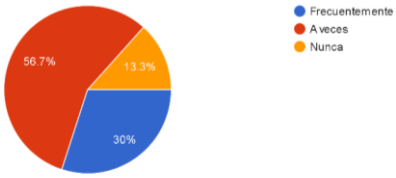
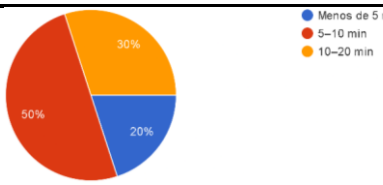
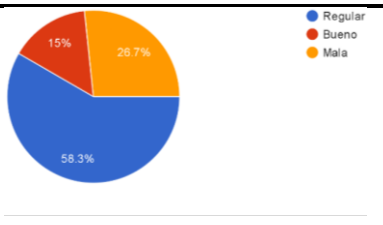
Tabla 2 Cronograma de recolección de datos de encuesta y entrevista

3.6 Análisis y presentación de resultados

3.6.1 Tabulación y análisis de los datos

- **Encuesta**

La encuesta fue aplicada a los estudiantes de la carrera de Agropecuaria de la Universidad Laica Eloy Alfaro de Manabí, extensión El Carmen, considerados los principales usuarios de las herramientas del vivero de cacao, con el fin de obtener información directa sobre el uso, control y disponibilidad del inventario.

Pregunta	Grafica	Análisis
1. ¿Cómo califica el control actual del inventario de herramientas?		La mayoría de los estudiantes indica que el control actual del inventario es regular y un porcentaje considerable dice que es deficiente
2. ¿Cree usted que el proceso de préstamo de herramientas es ordenado?		La mayoría de los estudiantes dijo que a veces estos procesos son ordenados y grupo considerable dijo que no
3. ¿La devolución incluye la revisión del estado de la herramienta?		Los estudiantes en su mayoría dijeron que a veces cuando hacen la devolución revisan el estado
4. ¿Ha experimentado errores, confusiones o pérdida de registros debido al manejo manual?		La mayoría de encuestados dice que si ha presentado errores en la perdida de los registros manuales
5. ¿Con qué frecuencia ha observado herramientas perdidas o dañadas?		Los estudiantes indicaron que las pérdidas o daños de herramientas ocurren ocasionalmente, situación relacionada con la falta de control y seguimiento adecuado de los recursos.
6. ¿Cuál es el tiempo promedio para encontrar una herramienta y que le sea entregada?		Los resultados indican que, si bien algunas entregas se realizan con rapidez, en determinadas ocasiones los estudiantes deben esperar por la dificultad para ubicar al bodeguero o identificar las herramientas que se encuentran disponibles.
7. ¿Cómo percibe el nivel de organización del área de almacenamiento?		La mayoría de los estudiantes considera que la organización del área de almacenamiento es regular, principalmente en la bodega de Lastenia, debido a que funciona en un espacio provisional con dificultades para mantener las herramientas ordenadas.

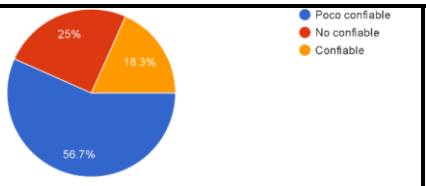
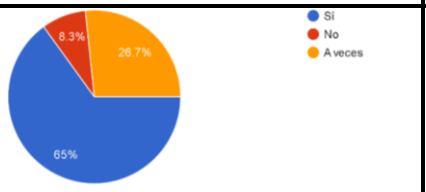
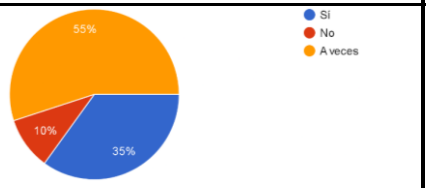
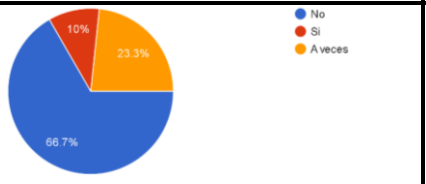
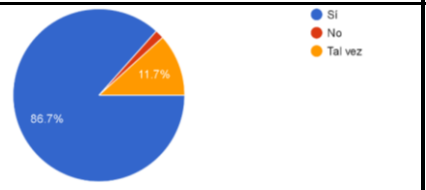
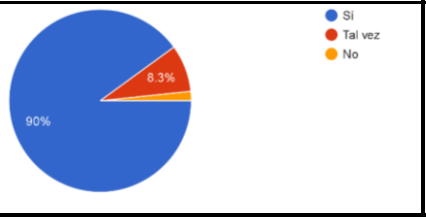
8. ¿Cree que la información del inventario es confiable?		Los estudiantes consideran que la información del inventario no siempre es completamente confiable ni actualizada, debido a que gran parte de los registros se realizan manualmente.
9. ¿El proceso manual de atención genera demoras?		Los resultados evidencian que el proceso manual sí genera demoras durante la atención, principalmente cuando el responsable de la bodega se encuentra realizando otras actividades.
10. ¿Ha tenido problemas por falta de herramientas disponibles?		Una parte importante de los encuestados manifestó haber tenido inconvenientes por falta de herramientas, especialmente durante mingas o actividades grupales donde existe mayor demanda.
11. ¿El sistema actual le parece eficiente?		Para la mayoría de los estudiantes el sistema no les parece eficiente
12. ¿Usaría un sistema digital para préstamos y devoluciones?		Los estudiantes en su mayoría dicen que si usarías un sistema digital para los procesos
13. ¿Considera que implementar un sistema informático de inventario mejoraría la organización del inventario de herramientas?		La mayoría de los estudiantes considera que la implementación de un sistema informático contribuiría a mejorar la organización, el control y la disponibilidad de las herramientas, al mismo tiempo que permitiría disminuir errores y agilizar el tiempo de atención.

Tabla 3 Tabulación y análisis de los datos de encuesta

- **Entrevista**

La entrevista fue realizada al Ing. Pedro Eduardo Nivela Morante, responsable del proyecto de vinculación y de la administración de las bodegas del vivero de cacao. A través de sus respuestas se obtuvo información relevante sobre la forma en que actualmente se gestionan, supervisan y organizan las herramientas e insumos utilizados en las actividades del vivero. Los datos recopilados permitieron conocer de primera mano las principales dificultades del proceso, así

como identificar las necesidades que sirvieron de base para el diseño de la propuesta tecnológica planteada en esta investigación.

Preguntas	Pedro nivela responsable del proyecto del vivero de cacao	Análisis
1. ¿Cómo se realiza actualmente el registro y control del inventario de herramientas?	Manifestó que existe un inventario general manejado por la coordinación de carrera de la universidad y un Kardex de herramientas. En la granja, el bodeguero registra manualmente en un cuaderno quién retira las herramientas y solicita la cédula como garantía. Sin embargo, en Lastenia no existe un registro formal y el préstamo de herramientas se realiza únicamente de forma verbal.	Se evidencia que el control del inventario es manual en una bodega y en la otra no se lleva ningún registro, generando inconsistencias y falta de estandarización en los procesos de registro.
2. Describa el procedimiento de préstamo y devolución de herramientas a estudiantes.	El encargado indico que el estudiante pide la herramienta al bodeguero, y la anota manualmente y conserva la cédula hasta que la herramienta sea devuelta.	El proceso que se realiza para el préstamo de herramientas es manual
3. ¿Qué mecanismos utilizan para verificar el estado físico de las herramientas al momento de la devolución?	Señaló que las herramientas deben entregarse limpias y se revisa visualmente su estado antes de ser almacenadas nuevamente.	La revisión del estado físico de las herramientas se efectúa de forma manual y elemental, sin documentos o registros establecidos que permitan conservar un historial de los daños y mantenimientos realizados.
4. ¿Cuáles son las principales dificultades o limitaciones del manejo manual del inventario?	Manifestó que la falta de tiempo constituye una de las principales dificultades, ya que los bodegueros también desempeñan labores de conserjería y vigilancia. La cantidad de funciones asignadas limita la posibilidad de supervisar continuamente las herramientas.	La carga excesiva de trabajo y la falta de un sistema automatizado disminuyen la eficiencia en la gestión del inventario y dificultan que las herramientas estén disponibles cuando se necesitan.
5. ¿Con qué frecuencia se presentan pérdidas, daños o inconsistencias en el inventario?	Señaló que cada semana se presentan casos de herramientas que no han sido devueltas, documentos que permanecen olvidados y diferencias en los registros de préstamos y devoluciones.	La presencia recurrente de inconsistencias refleja deficiencias en el método actual utilizado para controlar y dar seguimiento a las herramientas.
6. ¿Cuánto tiempo, en promedio, toma localizar y entregar una herramienta solicitada?	Indicó que la entrega generalmente es inmediata; sin embargo, cuando el bodeguero se encuentra realizando otras actividades, es necesario localizarlo, ya que no permanece en un lugar fijo.	Aunque la entrega puede ser rápida, la falta de organización y disponibilidad permanente del responsable ocasiona retrasos en determinados momentos.
7. ¿Cómo evalúa la organización y distribución del área de almacenamiento?	Indicó que la bodega de Lastenia funciona en condiciones semiprovisionales y cambia de lugar con frecuencia, por lo que actualmente se encuentra instalada en un espacio improvisado. Por el contrario, la bodega de la granja conserva una ubicación fija y organizada desde hace cerca de 15 años.	La condición provisional de la bodega de Lastenia limita la adecuada organización del inventario y dificulta mantener un control estable de las herramientas.
8. ¿Considera que la información registrada del inventario es precisa y se mantiene actualizada? Explique.	Señaló que se dispone de información general sobre el inventario, aunque esta no siempre se mantiene actualizada por el uso de registros manuales y las limitaciones operativas del personal responsable.	El registro manual de la información aumenta la posibilidad de cometer errores y disminuye la confiabilidad de los datos disponibles.
9. ¿Se generan demoras durante la atención a los usuarios? ¿Cuáles son las causas principales?	Mencionó que sí pueden presentarse demoras cuando el bodeguero se encuentra realizando otras funciones fuera de la bodega.	Las múltiples responsabilidades asignadas al personal afectan la rapidez y eficiencia en la atención de solicitudes de herramientas.
10. ¿Con qué frecuencia no hay herramientas suficientes para todos los estudiantes?	Dijo que durante las mingas o actividades grupales sabe haber escasez de herramientas, debido a que varios cursos las utilizan al mismo tiempo.	Al no haber suficientes herramientas en las mingas es el inconveniente
11. Desde su experiencia, ¿considera eficiente el sistema actual? ¿Qué mejoras propone para optimizar el control y la gestión de herramientas?	Considera que el método actual resulta poco eficiente y que, con los recursos tecnológicos disponibles, estas actividades deberían realizarse de forma automatizada. Además, manifestó que el vivero requiere con urgencia un sistema informático que permita fortalecer el control y mejorar la administración del inventario.	Se evidencia la necesidad inmediata de actualizar este proceso mediante recursos tecnológicos que mejoren la organización y permitan dar un seguimiento adecuado al inventario.
12. ¿Estaría dispuesto a utilizar un sistema digital	Señaló que estaría dispuesto a emplear un sistema digital y añadió que los estudiantes de vinculación también podrían utilizar la plataforma.	Se observa una respuesta favorable ante la incorporación de herramientas

para registrar préstamos y devoluciones?		digitales destinadas a fortalecer el control del inventario.
13. ¿Cree que un sistema informático mejoraría la organización y gestión de las herramientas? ¿Por qué?	Señaló que la implementación de un sistema informático contribuiría a fortalecer el control, disminuir las pérdidas, agilizar los procesos y mejorar la organización de las herramientas utilizadas por los estudiantes de vinculación de sexto y séptimo semestre.	La propuesta del sistema se considera una alternativa factible para mejorar la eficiencia operativa y administrativa del vivero de cacao.

Tabla 4 Tabulación y análisis de los datos de entrevista

3.6.2 Presentación y descripción de los resultados obtenidos

Los resultados de la encuesta coinciden con la entrevista. Las preguntas 1 y 4 evidencian que el control del inventario es manual, poco confiable y propenso a errores, situación confirmada en las preguntas 1 y 5 de la entrevista. Asimismo, la pregunta 6 relaciona las demoras en la localización de herramientas con la ausencia frecuente del bodeguero, señalada en la pregunta 6 de la entrevista.

La pregunta 8 muestra problemas de confiabilidad ocasionados por traslados de herramientas sin notificación, mientras que la pregunta 11 confirma la ineficiencia del proceso manual. Finalmente, las preguntas 12 y 13 reflejan una alta aceptación de un sistema digital, necesidad respaldada por el Ing. Pedro Nivela en la pregunta 13 de la entrevista.

3.6.3 Informe final del análisis de los datos

El análisis de la encuesta aplicada a los 60 estudiantes y de la entrevista realizada al responsable del vivero permitió contrastar las causas planteadas en el árbol del problema. La existencia de registros manuales se evidencia en la pregunta 4 de la encuesta, en la que el 63,3 % de los participantes indicó haber experimentado errores, confusiones o pérdida de registros y el 26,7 % señaló que esto ocurre algunas veces. Asimismo, en la pregunta 8, el 56,7 % consideró que la información es poco confiable y el 25 % manifestó que no es confiable. Estos resultados coinciden con las respuestas de las preguntas 1 y 2 de la entrevista, donde se confirmó que los

préstamos y las devoluciones se registran manualmente y que, en el Campus Lastenia, algunas operaciones se realizan de manera verbal.

La falta de un protocolo formal y persistente para el registro de las operaciones se comprueba en las preguntas 2 y 3 de la encuesta. El 86,7 % de los estudiantes señaló que el proceso de préstamo solamente algunas veces es ordenado o que no lo es, mientras que el 73,3 % indicó que la revisión del estado de las herramientas durante la devolución se realiza solo algunas veces o no se realiza. De igual manera, en las preguntas 1, 2 y 3 de la entrevista se indicó que no existe un registro formal en el Campus Lastenia, que la información se anota manualmente y que la condición física de las herramientas se revisa visualmente, sin documentos que permitan conservar su historial.

Las inconsistencias en la información se reflejan en las preguntas 4 y 8 de la encuesta, así como en las preguntas 5 y 8 de la entrevista. El responsable manifestó que se presentan herramientas no devueltas, documentos olvidados y diferencias entre los registros de préstamos y devoluciones. También señaló que la información no siempre se mantiene actualizada. Estos resultados demuestran la presencia de datos contradictorios y dificultades para efectuar el seguimiento; sin embargo, los instrumentos no preguntaron directamente si existe una política para resolver conflictos entre registros, por lo que la ausencia de dicha política solo puede considerarse parcialmente comprobada.

Los efectos del problema también fueron confirmados. En la pregunta 9 de la encuesta, el 91,7 % de los estudiantes manifestó que el proceso manual genera demoras o que estas se producen algunas veces. Además, la pregunta 6 muestra que el 80 % debe esperar entre cinco y veinte minutos para localizar y recibir una herramienta. Esta situación coincide con las preguntas 4, 6 y

9 de la entrevista, en las que se relacionan las demoras con la falta de una ubicación fija, la organización provisional de la bodega y las múltiples funciones asignadas al personal responsable.

Finalmente, la conectividad intermitente, establecida como una de las causas del árbol del problema, no fue evaluada directamente mediante las preguntas de la encuesta o de la entrevista. Por consiguiente, los datos presentados no permiten afirmar que esta causa quedó comprobada con los instrumentos aplicados. Para verificarla se requiere incorporar una ficha de observación o un registro técnico que detalle la frecuencia, duración y efectos de las interrupciones de Internet en el Campus Lastenia.

CAPÍTULO IV

4 MARCO PROPOSITIVO

4.1 Introducción

La propuesta técnica convierte los requisitos en una arquitectura de datos distribuida compatible con el producto actual. Se presenta como un incremento dentro del ERP Lastenia y no como una aplicación independiente. Su frontera comienza después de que un módulo funcional confirma una escritura y termina cuando el destino aplica o registra el cambio, dejando evidencia de cada estado.

El capítulo sigue una estructura comparable con la tesis aprobada de referencia: descripción, recursos, metodología Scrum, historias, casos de uso, requisitos, arquitectura, datos, sincronización, sprints, definición de terminado, pruebas e incremento. La diferencia es la concentración exclusiva en persistencia y distribución.

4.2 Descripción de la propuesta

La propuesta se denomina “Infraestructura distribuida de datos y sincronización del ERP Lastenia”. La arquitectura está compuesta por una base funcional de carácter modular y un plano transversal encargado de la sincronización. La base funcional mantiene la fuente local de verdad correspondiente a cada dominio, mientras que el plano transversal identifica los nodos, administra las versiones lógicas, registra las entregas y conserva las dos versiones cuando se presenta un conflicto.

Este enfoque permite preservar las reglas de negocio, ya que el receptor procesa cada evento a través de un SyncEntityAdapter, el cual invoca el contrato correspondiente al módulo

propietario. La misma aplicación desarrollada en Laravel puede configurarse para operar como Administrador o como Central. Por su parte, el módulo Móvil se integra como un tercer emisor y receptor compatible, aunque utiliza un esquema reducido basado en SQLite.

El objetivo operacional consiste en permitir que el usuario confirme un movimiento de forma local, sin depender de la disponibilidad de una conexión a Internet. Por otro lado, el objetivo distribuido es garantizar que, una vez restablecida la comunicación, cada evento sea entregado al destino al menos una vez y genere como máximo un único efecto; en caso contrario, el conflicto queda registrado de forma explícita. El objetivo de auditoría es reconstruir qué se intentó, desde dónde, con qué versión y cuál fue el resultado.

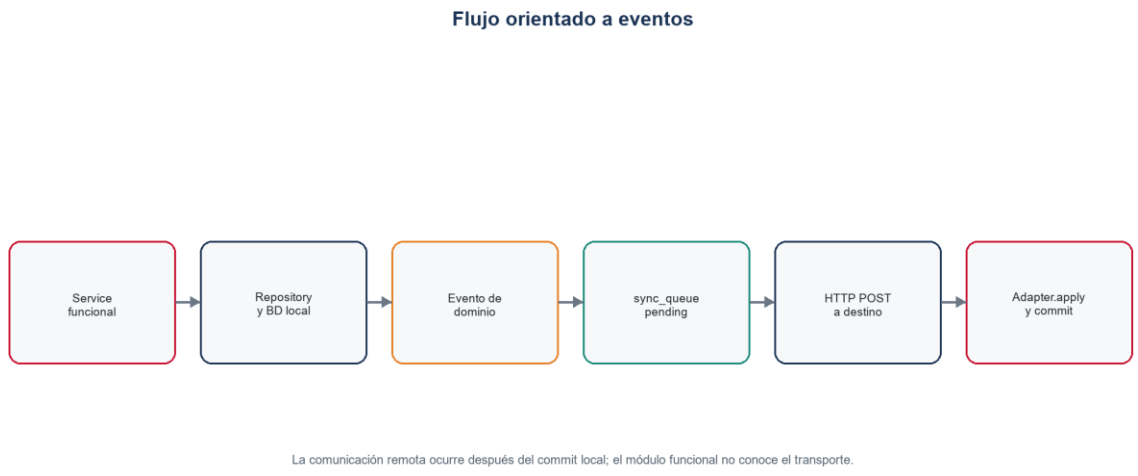


Ilustración 2 Flujo del cambio desde el módulo hasta el destino

4.3 Recursos del proyecto

4.3.1 Recursos humanos

Recurso	Función	Actividades
Zambrano Bravo Alex Jeanpierre	Desarrollador del módulo	Responsable del modelado conceptual, lógico y físico de la base de datos; creación de migraciones, restricciones e índices; diseño de los contratos de integración;

		implementación de la sincronización bidireccional; gestión de eventos, colas, versiones y conflictos; ejecución de pruebas y elaboración de la documentación técnica.
Ing. Mendoza Villamar Rocio Alexandra, Mg.	Tutora del proyecto	Encargada de supervisar el avance metodológico y técnico del proyecto, revisar la documentación, orientar las decisiones relacionadas con la arquitectura de datos y verificar el cumplimiento de los objetivos establecidos para el trabajo de titulación.
Equipo de arquitectura y propietarios de los módulos funcionales	Revisores e integradores funcionales	Definen los DTO, eventos y reglas de negocio que deben utilizar los adaptadores de Planning, Tasks, Inventory, Logistics y Tracking.
Responsable del vivero	Validador funcional	Comprueba que los procesos de registro, préstamo, devolución, mantenimiento y control de herramientas se ajusten al funcionamiento real del vivero. Asimismo, valida que los requisitos, los procedimientos y los resultados del sistema sean consistentes con las necesidades operativas.
Zambrano Bravo Alex Jeanpierre	Administrador de nodos y seguridad	Registra, configura y desactiva los nodos de sincronización, asimismo, administra las direcciones, credenciales y tokens; supervisa las colas de procesamiento, los errores y los conflictos; controla la ejecución de los procesos programados y verifica los respaldos, la conectividad y las condiciones necesarias antes del despliegue del sistema.

Tabla 5 recursos humanos

4.3.2 Recursos tecnológicos

Recursos de software

Capa	Recurso	Versión o condición	Función y uso en el proyecto
Backend	PHP	8.2 o superior	Lenguaje de ejecución de la lógica del módulo de base de datos y sincronización.
Framework	Laravel	12.x	Organiza servicios, repositorios, eventos, listeners, colas, jobs, comandos y tareas programadas del módulo.
Persistencia	Eloquent ORM	Incluidas en Laravel 12	Mapea modelos y relaciones y permite ejecutar consultas y transacciones sobre las entidades funcionales y de sincronización.

Esquema	Migraciones de Laravel	Incluidas en Laravel 12	Crean y actualizan tablas, claves foráneas, índices, restricciones y estructuras transversales de sincronización.
BD de servidor	MySQL / InnoDB	8.4 objetivo	Persistencia transaccional de los nodos Administrador y Central, con restricciones, índices y claves foráneas.
BD local y móvil	SQLite	Versión del entorno	Se utiliza en pruebas locales y se proyecta como persistencia nativa del futuro nodo móvil con funcionamiento offline.
Transporte	Cliente HTTP de Laravel	Incluidas en Laravel 12	Envía los sobres JSON entre los nodos, interpreta las respuestas obtenidas mediante HTTP y registra el resultado de cada operación, ya sea como éxito, reintento o conflicto.
Procesamiento	Laravel Queue y Jobs	Incluidas en Laravel 12	Procesan las entregas pendientes fuera de la transacción funcional y aplican reintentos controlados.
Programación	Laravel Scheduler	Incluidas en Laravel 12	Se encarga de ejecutar periódicamente el comando sync:run cuya función es seleccionar y despachar los cambios disponibles para su posterior sincronización.
Pruebas	PHPUnit	11.5.x	Verifica migraciones, eventos, cola, endpoint receptor, idempotencia, versionado, conflictos y adopción por módulos.
Nodo móvil	Capacitor	Versión por fijar	Runtime aprobado para empaquetar la aplicación Android que utilizará SQLite, outbox local y sincronización push/pull; su integración permanece pendiente.

Tabla 6 Recursos de software del módulo de base de datos y sincronización

Infraestructura para producción

Recurso	Condición	Función en producción
Servidor web	Compatible con PHP 8.2 y Laravel 12	Se encarga de publicar la API y habilitar la aplicación en un entorno de producción utilizando una configuración endurecida que contemple medidas de seguridad, estabilidad y rendimiento.
Variables de entorno	Archivo. env protegido por nodo	Separar la identidad local, el destino remoto, la conexión a la base de datos y los secretos de configuración, evitando su incorporación directa en el código fuente.
Scheduler	Cron cada minuto	Ejecutar el planificador de Laravel e invocar el comando sync: run para identificar y seleccionar las entregas disponibles.
Worker de colas	Proceso permanente	Son procesados de manera automática para gestionar el envío y la recepción de información entre los nodos, mantienen un historial de los errores y conflictos detectados, lo que facilita el monitoreo, la auditoría y la resolución de incidencias durante el proceso de sincronización.
Supervisor	Supervisor o servicio equivalente	Mantener en ejecución el worker para garantizar su reinicio automático en caso de fallos o interrupciones.
Certificado TLS	Certificado vigente	Garantizar la confidencialidad de la comunicación entre los nodos mediante conexiones seguras HTTPS.
Respaldos	Política periódica y restauración probada	Garantizar la protección de las bases de datos mediante mecanismos que permitan recuperar la información ante situaciones de pérdida, alteración o corrupción de datos.
Monitoreo	Métricas, registros y alertas	Supervisar los elementos pendientes, fallidos y en conflicto, así como la antigüedad de la cola, el último contacto registrado y el estado de ejecución de los procesos.

Tabla 7 Recursos de infraestructura requeridos para producción

Recursos de hardware y red

Recurso	Condición o característica	Función en la topología
PC Administrador	Equipo ubicado en el Campus Lastenia, conectado a la LAN local y con capacidad para ejecutar el backend y MySQL.	Mantener la operación local del sistema, registrar cambios y conservar la cola de sincronización cuando no exista conexión con el nodo Central.
Servidor Central	Servidor accesible mediante HTTPS, con MySQL/InnoDB y disponibilidad para recibir solicitudes de los nodos autorizados.	Consolidar la información sincronizada, mantener el estado autorizado de las entidades y responder adecuadamente a los procesos de intercambio de datos entre los nodos.
Dispositivo Android	Contar con un equipo compatible con el runtime móvil aprobado, almacenamiento privado disponible y conectividad mediante Wi-Fi o datos móviles.	Permitir la ejecución del nodo móvil, el registro de operaciones en ausencia de conexión y la sincronización de los datos generados una vez recuperada la conectividad con la red.
Almacenamiento de servidores	Contar con un espacio dimensionado de acuerdo con la carga actual medida y las proyecciones de crecimiento real del sistema.	Archivar la información en la base de datos funcional, garantizando su disponibilidad e integridad.
Almacenamiento móvil privado	Espacio interno protegido del dispositivo Android destinado al almacenamiento de la base de datos SQLite y la outbox local.	Evitar que las carpetas sean compartida mediante la exposición directa del archivo de la base de datos SQLite, este se debe realizar mediante los mecanismos de sincronización implementados.
Red LAN de Lastenia	Conectividad local estable entre la interfaz de usuario, el backend y la base de datos del nodo Administrador.	Permitir que el sistema continúe funcionando de manera local, aun cuando la conexión a Internet sea intermitente o se encuentre temporalmente indisponible, garantizando la continuidad de las operaciones y el almacenamiento de la información hasta que sea posible restablecer la sincronización con los demás nodos.
Enlace a Internet	Conexión intermitente disponible entre el nodo Administrador y el nodo Central.	Implementar mecanismos de reanudación automático y garantizar la protección de la información.
Wi-Fi o datos móviles	Conectividad disponible para el dispositivo Android durante su operación.	Garantizar que el sistema funcione de forma local en los tiempos de desconexión.
Medio de respaldo	Respaldos en un almacenamiento seguro e independiente, con una	Garantizar la recuperación de las bases de datos y de la información operativa en caso de pérdida de datos, corrupción de la información

	política de retención y restauración previamente validada.	o fallas del equipo principal, asegurando la continuidad del servicio y la integridad de la información mediante mecanismos adecuados de respaldo y restauración.

Tabla 8 Recursos de hardware y red

4.3.3 Recursos económicos

Cantidad	Recurso	Precio unitario (USD)	Subtotal (USD)
1	Equipo de desarrollo	1000.00	1000.00
6 meses	Servicio de Internet	25.00	150.00
6 meses	Servidor Central en la nube	20.00	120.00
1	Dispositivo Android para pruebas	250.00	250.00
1	Cable USB para pruebas	2.00	2.00
1	Medio externo de respaldo	70.00	70.00
200 horas	Diseño, desarrollo, pruebas y documentación	10.00	2000.00
1	Licencias de PHP, Laravel, MySQL Community, SQLite y PHPUnit	0.00	0.00
		TOTAL:	**3592.00**

Tabla 9 Recursos económicos

4.4 Desarrollo según metodología Scrum

El Product Backlog se construyó con historias enfocadas en riesgos de datos. Los sprints generan incrementos demostrables, cuya revisión permite comparar los resultados obtenidos con los criterios de aceptación definidos, asimismo, la retrospectiva documenta las decisiones de arquitectura y la deuda técnica pendiente. La tabla de sprints diferencia los elementos implementados de los dos incrementos requeridos para completar la topología del sistema.

El Product Owner establece las prioridades relacionadas con la continuidad e integridad del sistema; el equipo de desarrollo se encarga de la implementación y las pruebas; mientras que el Scrum Master facilita la ejecución del proceso. En un proyecto académico, una misma persona puede asumir más de una responsabilidad, siempre que los compromisos adquiridos y los criterios definidos permanezcan claramente establecidos.

4.4.1 Descripción del producto

4.4.1.1 Propósito del producto

Nuestro objetivo es crear un sistema de bases de datos inteligente y repartido que permita al equipo del Vivero de Cacao Lastenia registrar, consultar y guardar información de sus herramientas desde el panel de Administración o desde un Celular. La gran ventaja es que todo esto se puede hacer incluso si el internet falla o es muy limitado. Apenas hay señal, los dispositivos se comunican en ambas direcciones con un Servidor Central en la nube para actualizarse mutuamente. Con esto logramos que nunca más se pierdan datos, se anoten cosas dos veces o haya información contradictoria. Además, el sistema ya viene preparado para todo: tiene medidas de seguridad, guarda el historial de quién hizo cada cambio, avisa si hay un choque de información y es capaz de recuperarse por sí solo ante cualquier fallo técnico.

4.4.1.2 Funcionalidades clave

- Construir la estructura de almacenamiento de datos para los servidores y el dispositivo móvil, definiendo las tablas, relaciones, restricciones e índices necesarios para garantizar una organización consistente de la información durante todo el ciclo de vida del sistema.
- Habilitar el funcionamiento local del sistema en los nodos Administrador y Móvil, de manera que los usuarios puedan continuar registrando y consultando información aun cuando no exista acceso a Internet.

- Asignar una identidad propia a cada nodo de la arquitectura distribuida, diferenciando al Administrador, Central y Móvil mediante identificadores únicos, estados operativos y credenciales independientes de las utilizadas por los usuarios del sistema.
- Registrar cada modificación realizada sobre la información como una operación pendiente de sincronización, almacenándola de forma temporal hasta que el nodo receptor confirme que el cambio fue procesado correctamente.
- Acceder continuamente a la información entre el Nodo Central y los nodos Administrador y Móvil.
- Usar servicios web como HTTPS para el envío y recepción de información, aplicando mecanismos de autenticación y validación que garanticen que solo los nodos autorizados puedan ingresar al proceso de sincronización.
- Incluir controles que permitan detectar cada evento de sincronización como una operación única, para evitar duplicidad en los envíos.
- Conservar las operaciones que aún no han sido enviadas y retomarlas automáticamente cuando la comunicación vuelva a estar disponible.
- Comparar la versión más reciente para determinar si otro nodo realizó cambios previamente. Cuando exista una diferencia entre ambas versiones, el sistema debe identificarla como un conflicto y conservar la información necesaria para su análisis y resolución.
- Difundir entre todos los nodos mediante un mecanismo la eliminación y así permita reconocer que el elemento ya no forma parte de la información válida.
- Realizar de manera incremental la actualización de los datos, recuperando únicamente los registros creados o modificados desde la última sincronización exitosa.
- Conservar un registro de las operaciones procesadas, los intentos efectuados, las respuestas recibidas y cualquier error o conflicto detectado, facilitando posteriormente las labores de seguimiento y auditoría.
- La sincronización entre los nodos debe ejecutarse sin intervención permanente del usuario mediante procesos automáticos en segundo plano.
- Aplicar medidas de seguridad para proteger la información almacenada y transmitida, incluyendo el uso de almacenamiento privado en los dispositivos móviles, comunicaciones cifradas, autenticación de nodos, controles de acceso, copias de seguridad y mecanismos de recuperación ante fallos.

4.4.1.3 Usuarios de la propuesta

Tipo de usuario	Principales funcionalidades
Usuario operativo del vivero	Registra y consulta la información autorizada desde el Nodo Administrador o el dispositivo móvil; continúa trabajando sin conexión; identifica si un cambio está pendiente, sincronizado o en conflicto; y recibe la información actualizada cuando se restablece la conectividad.
Responsable del vivero	Consulta la información consolidada y trazable del vivero, verifica que los datos sincronizados correspondan con la operación real y valida, cuando sea necesario, cuál versión de negocio debe conservarse ante un conflicto.

Tabla 10 Usuarios objetivo del módulo de base de datos y sincronización

4.4.1.4 Condiciones de éxito del producto

Cuando se restablece la conexión a Internet, los cambios pendientes se sincronizan automáticamente, permitiendo que todos los nodos actualicen su información y mantengan el mismo estado.

Los cambios realizados en los nodos Administrador y Central se sincronizan de forma bidireccional, mientras que la información generada en el nodo Móvil se envía y recibe a través del Nodo Central.

El sistema garantiza que una misma solicitud procesada más de una vez no genere registros duplicados ni provoque efectos repetidos sobre la base de datos.

Cuando existen modificaciones concurrentes sobre un mismo registro, el sistema detecta las diferencias de versión y genera un conflicto que queda registrado para su posterior revisión, evitando la pérdida o sobrescritura silenciosa de la información.

Las eliminaciones se sincronizan entre todos los nodos mediante un mecanismo de borrado lógico, impidiendo que los registros eliminados vuelvan a aparecer en futuras sincronizaciones.

La cola de sincronización conserva las operaciones pendientes cuando ocurren fallos de red, reinicios o interrupciones temporales del servicio, reanudando automáticamente el proceso una vez restablecidas las condiciones normales de funcionamiento.

El intercambio de información se realiza únicamente entre nodos activos y debidamente autenticados mediante conexiones HTTPS, rechazando y registrando cualquier intento de acceso no autorizado, uso de credenciales inválidas o alteración de los datos transmitidos.

Las migraciones permiten mantener de forma consistente la estructura de las bases de datos MySQL y SQLite, preservando claves, relaciones, restricciones e índices necesarios para garantizar la integridad de la información.

Los mecanismos de respaldo y restauración permiten recuperar las bases de datos, las colas de sincronización, el historial de versiones y los conflictos registrados, sin afectar las operaciones previamente confirmadas.

Antes de poner el sistema en funcionamiento, se realizaron diferentes escenarios de prueba para comprobar su comportamiento en condiciones normales y ante posibles fallos. Estas evaluaciones incluyeron pruebas unitarias, de integración, pérdida y recuperación de la conexión, reenvío de operaciones, resolución de conflictos y recuperación del servicio, permitiendo verificar que el mecanismo de sincronización respondiera de forma confiable.

Se implementó un proceso de monitoreo que permite conocer en todo momento el estado de los nodos, controlar las operaciones que permanecen pendientes, detectar oportunamente posibles errores y comprobar que la sincronización se complete dentro de los tiempos de respuesta esperados, de acuerdo con los resultados obtenidos durante las pruebas de rendimiento.

4.4.2 Historias de usuario

HISTORIA DE USUARIO			
Id de historia:	HU-01	**Título:**	Gestión segura de nodos
Prioridad:	Alta	**Usuario:**	Administrador técnico
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Administrador técnico del ERP Lastenia		
Quiero:	Registrar, activar, desactivar y renovar las credenciales de los nodos Administrador, Central y Móvil.		
Para:	Garantizar que únicamente las instancias identificadas y autorizadas intervengan en la sincronización de los datos.		
Criterios de aceptación:	Cada nodo cuenta con una identidad propia que lo diferencia del resto de la infraestructura. Esta identidad incluye un UUID, un código, un nombre, el tipo de nodo, su dirección y el estado en que se encuentra. Las credenciales se crean una sola vez. Para proteger la seguridad del sistema, el token original no se almacena; únicamente se conserva su hash. Solo los nodos habilitados pueden intercambiar información. Cualquier solicitud proveniente de un nodo deshabilitado o con credenciales incorrectas es rechazada. El sistema mantiene un registro de las operaciones administrativas realizadas sobre los nodos. De esta manera es posible consultar el historial de cambios cuando sea necesario.		
Dependencias:	Tabla sync_nodes y migraciones implementadas. Permiso de administración, conexión HTTPS y almacenamiento protegido de secretos.		

Tabla 11 Historia de usuario HU-01: Gestión segura de nodos

HISTORIA DE USUARIO			
Id de historia:	HU-02	**Título:**	Captura de cambios sincronizables
Prioridad:	Alta	**Usuario:**	Responsable del módulo funcional
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Encargado de un módulo funcional		
Quiero:	Generar un evento sincronizable después de confirmar una creación, actualización o eliminación en la base de datos local.		
Para:	Agregar las entidades con la distribución de datos sin vincular el módulo directamente con las tablas internas de Synchronization.		
Criterios de aceptación:	Este evento se crea únicamente después de confirmar la transacción funcional. Incluye event_id, entity_type, entity_id, operación, fecha, versión y nodo de origen. Una transacción revertida no genera eventos ni registros pendientes. La aplicación de un cambio remoto no produce un evento de retorno hacia el nodo de origen.		
Dependencias:	Identificadores permanentes, eventos de dominio y nodo local configurado. SyncEntityAdapter y contrato de datos establecidos por el módulo responsable.		

Tabla 12 Historia de usuario HU-02: Captura de cambios sincronizables

HISTORIA DE USUARIO			
Id de historia:	HU-03	**Título:**	Cola durable de sincronización
Prioridad:	Alta	**Usuario:**	Administrador técnico
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Administrador técnico		
Quiero:	Mantener cada cambio pendiente en una cola durable con su destino, operación, versiones y payload.		
Para:	Cuando el servidor este en espera prevenir la pérdida de información, o en el proceso de sincronización		
Criterios de aceptación:	La entrada se registra en estado pending dentro de una transacción. La combinación de event_id y target_node_id vita entregas duplicadas. El payload se mantiene sin cambios desde el primer intento para que todos los reintentos sean equivalentes. Los eventos pendientes continúan disponibles después de reiniciar el backend o el worker.		
Dependencias:	Tablas sync_queue y sync_entity_states. Listener de eventos, base de datos operativa y nodo de destino registrado.		

Tabla 13 Historia de usuario HU-03: Cola durable de sincronización

HISTORIA DE USUARIO			
Id de historia:	HU-05	**Título:**	Reintentos seguros e idempotencia
Prioridad:	Alta	**Usuario:**	Usuario operativo del vivero
Riesgo de desarrollo:	Medio	**Tamaño de la tarea:**	Mediana
Como:	Usuario operativo del vivero		
Quiero:	Que el sistema permita reintentar los envíos fallidos sin duplicar el efecto de una operación previamente recibida.		
Para:	Que las operaciones se mantengan ante fallos temporales de red sin generar registros ni movimientos duplicados.		
Criterios de aceptación:	Un fallo temporal devuelve la entrega al estado pending con un tiempo de espera progresivo. Al llegar al máximo de intentos, la entrega pasa a failed y conserva el error registrado. Un event_id previamente procesado se reconoce como duplicado sin aplicar otra vez el cambio. Los reintentos continúan después de reiniciar el worker y quedan registrados.		
Dependencias:	Cola durable, event_id único y receptor con comportamiento idempotente. Scheduler, worker, política de reintentos y registro de fallos.		

Tabla 14 Historia de usuario HU-04: Sincronización bidireccional automática

HISTORIA DE USUARIO			
Id de historia:	HU-04	**Título:**	Sincronización bidireccional automática
Prioridad:	Alta	**Usuario:**	Usuario operativo del vivero
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Usuario operativo del vivero		
Quiero:	Que se sincronice los cambios realizados en los nodos Administrador, Central y Móvil se envíen y reciban automáticamente cuando exista conectividad.		
Para:	Analizar la información actualizada sin necesidad de ingresar nuevamente los datos registrados durante la interrupción de Internet.		
Criterios de aceptación:	La operación autorizada se registra primero en la base de datos local. El Administrador y el Central intercambian cambios en ambos sentidos mediante HTTPS. El Móvil envía su outbox y descarga cambios incrementales desde el Central. Después de procesar las colas, los nodos alcanzan el mismo estado o registran un conflicto.		
Dependencias:	Nodos registrados, API de sincronización, scheduler y worker operativos. Adaptadores funcionales, outbox móvil, cursor de descarga y conexión disponible.		

Tabla 15 Historia de usuario HU-05: Reintentos seguros e idempotencia

HISTORIA DE USUARIO			
Id de historia:	HU-07	**Título:**	Trabajo móvil sin conexión
Prioridad:	Alta	**Usuario:**	Operador del vivero
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Operador del vivero que trabaja en campo		
Quiero:	Registrar y consultar en Android la información autorizada, incluso cuando no haya conexión a Internet.		
Para:	Garantizar la continuidad de las actividades del vivero y sincronizar los cambios al restablecerse la conectividad.		
Criterios de aceptación:	La aplicación consulta y guarda los datos en una base SQLite ubicada en el almacenamiento privado. Cada operación local crea una entrada en el outbox móvil dentro de la misma transacción. Los datos y eventos pendientes se conservan después de cerrar o reiniciar la aplicación. Al recuperar la conexión, el dispositivo envía los pendientes y descarga cambios sin compartir el archivo SQLite.		
Dependencias:	Aplicación React integrada con Capacitor y complemento SQLite nativo. Esquema móvil, UUID, outbox, cursor, permisos locales y control de conectividad.		

Tabla 16 Historia de usuario HU-06: Detección de conflictos de versión

HISTORIA DE USUARIO			
Id de historia:	HU-06	**Título:**	Detección de conflictos de versión
Prioridad:	Alta	**Usuario:**	Responsable del dato
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Responsable de la información del vivero		
Quiero:	Permitir que el sistema identifique y conserve las versiones diferentes de una entidad modificada en varios nodos.		
Para:	Que impida que una actualización válida sea sobrescrita o eliminada sin dejar registro.		
Criterios de aceptación:	Una base_version distinta de la versión registrada impide aplicar el cambio y genera un conflicto. Se conservan las versiones local y remota, junto con sus hashes, nodos, fechas y causa. La entrega y la entidad permanecen en estado conflict hasta su resuelva. Las eliminaciones concurrentes también se revisan mediante versionado y tombstones.		
Dependencias:	Versionado optimista en sync_entity_states y registro de conflictos en sync_conflicts. Adaptadores de entidad, trazabilidad y estados de sincronización visibles.		

Tabla 17 Historia de usuario HU-07: Trabajo móvil sin conexión.

HISTORIA DE USUARIO			
Id de historia:	HU-09	**Título:**	Resolución controlada de conflictos
Prioridad:	Alta	**Usuario:**	Responsable autorizado del vivero
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Responsable autorizado del vivero		
Quiero:	Revisar las versiones en conflicto y elegir o integrar la información que debe mantenerse.		
Para:	Recuperar la convergencia de los nodos sin perder el registro de la decisión adoptada.		
Criterios de aceptación:	La interfaz presenta ambas versiones con su nodo, autor, fecha y diferencias principales. Solo un usuario autorizado puede seleccionar una versión o registrar una combinación válida. La resolución genera una nueva versión y la distribuye a los nodos relacionados. El conflicto conserva el actor, la fecha, la decisión y el estado resolved sin eliminar el historial.		
Dependencias:	HU-06 implementada, tabla de conflictos y auditoría de usuarios. Servicio de resolución, interfaz de comparación y adaptador de la entidad.		

Tabla 18 Historia de usuario HU-08: Consulta del estado de sincronización

HISTORIA DE USUARIO			
Id de historia:	HU-08	**Título:**	Consulta del estado de sincronización
Prioridad:	Media	**Usuario:**	Responsable del vivero
Riesgo de desarrollo:	Medio	**Tamaño de la tarea:**	Mediana
Como:	Responsable del vivero		
Quiero:	Verificar si cada cambio se encuentra pendiente, en proceso, sincronizado, en conflicto o fallido.		
Para:	Identificar la confiabilidad y el alcance real de la información disponible en cada nodo.		
Criterios de aceptación:	El estado se muestra con información clara sobre la fecha, el nodo y el último resultado. La consulta permite filtrar los cambios pendientes, conflictos y fallos por entidad o período. Los estados se actualizan después de cada intento sin alterar los datos funcionales. La interfaz no muestra tokens, credenciales ni payloads sensibles al usuario operativo.		
Dependencias:	API de monitoreo, colas, conflictos, registros y permisos de consulta. Componentes de interfaz y actualización periódica de los estados.		

Tabla 19 Historia de usuario HU-09: Resolución controlada de conflictos

HISTORIA DE USUARIO			
Id de historia:	HU-10	**Título:**	Monitoreo, respaldo y recuperación
Prioridad:	Alta	**Usuario:**	Administrador técnico
Riesgo de desarrollo:	Alto	**Tamaño de la tarea:**	Grande
Como:	Administrador técnico		
Quiero:	Monitorear el estado de los nodos y las colas, recibir alertas y realizar respaldos y restauraciones verificadas.		
Para:	Identificar fallos a tiempo y restablecer la operación sin perder los cambios confirmados.		
Criterios de aceptación:	El monitoreo presenta la última comunicación, los pendientes, su antigüedad, los conflictos, fallos y el estado de los procesos. Los límites establecidos generan alertas sin mostrar información sensible. Los respaldos abarcan las bases funcionales, colas, versiones, conflictos, auditoría y configuración requerida. La restauración se verifica y permite continuar la sincronización sin duplicar los efectos ya confirmados.		
Dependencias:	Logs, métricas, scheduler, worker, supervisor y sistema de alertas. Almacenamiento seguro para respaldos y procedimiento de recuperación documentado.		

Tabla 20 Historia de usuario HU-10: Monitoreo, respaldo y recuperación

4.4.3 Diseño del Sistema / Descripción Técnica

El diseño técnico del módulo se organiza en tres casos de uso principales: la gestión de nodos, la sincronización bidireccional de los datos y la gestión de conflictos con monitoreo. Esta agrupación integra los casos técnicos internos sin trasladar al módulo de sincronización las reglas funcionales pertenecientes a inventario, planificación, tareas o trazabilidad.

4.4.3.1 Caso de uso: Gestión de nodos de sincronización

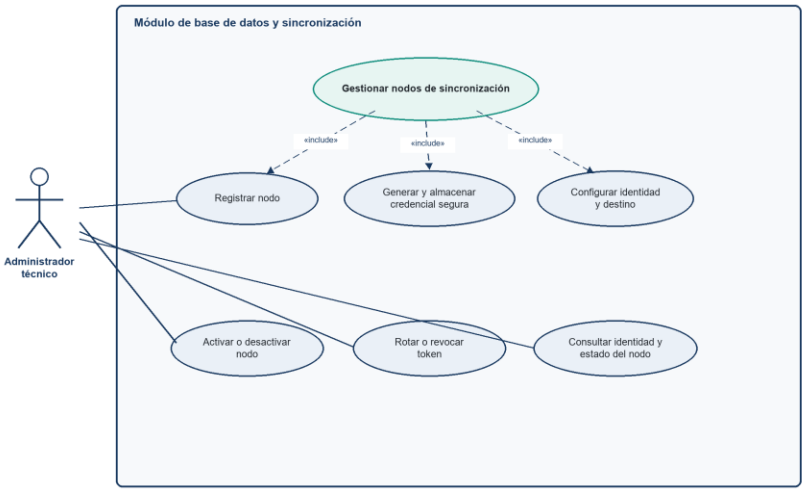


Ilustración 3 Caso de uso: Gestión de nodos de sincronización

4.4.3.1.1 Documentación del caso de uso: Gestión de nodos de sincronización

Documentación del caso de uso: Gestión de nodos de sincronización	
Caso de uso	Caso de uso N.° 001: Gestión de nodos de sincronización
Fecha y elaboración	01/08/2026 | responsable del módulo de base de datos y sincronización
Actores	Administrador técnico.
Objetivo	Registrar y gestionar las identidades de los nodos Administrador, Central y Móvil, creando credenciales seguras para su comunicación.
Precondiciones	El administrador ingresa a un entorno autorizado; el código del nodo es único, el tipo y la URL son válidos, y la base de datos está disponible.
Postcondiciones	El nodo se almacena en sync_nodes, permanece activo y cuenta con una identidad UUID. El hash del token se guarda y la credencial inicial se muestra una sola vez.
Medios para realizar la acción	Consola administrativa del backend y configuración protegida mediante variables de entorno.
Pasos	1. Ingresar al entorno autorizado del servidor. 2. Realizar el registro indicando código, nombre, tipo de nodo y URL. 3. Comprobar que el código sea único y que el tipo de nodo sea válido. 4. Crear la identidad UUID y un token aleatorio. 5. Generar y guardar el hash y el prefijo del token. 6. Registrar la identidad, la URL y el estado activo en sync_nodes. 7. Presentar la credencial inicial una sola vez. 8. Configurar la identidad y el destino fuera del repositorio de código. 9. Confirmar la autenticación y el estado del nodo.
Situaciones excepcionales	1. El código del nodo ya se encuentra registrado. 2. El tipo, la URL o los datos ingresados no son válidos. 3. La base de datos no está disponible. 4. La credencial se pierde o queda comprometida, por lo que debe renovarse o revocarse. 5. El nodo se encuentra desactivado y no puede autenticarse.

Tabla 21 Documentación del caso de uso: Gestion de nodos de sincronización

Revisado por: Tutor del proyecto y equipo de arquitectura

Tabla 1. Documentación del caso de uso: Gestión de nodos de sincronización

4.4.3.2 Caso de uso: Sincronización bidireccional de datos

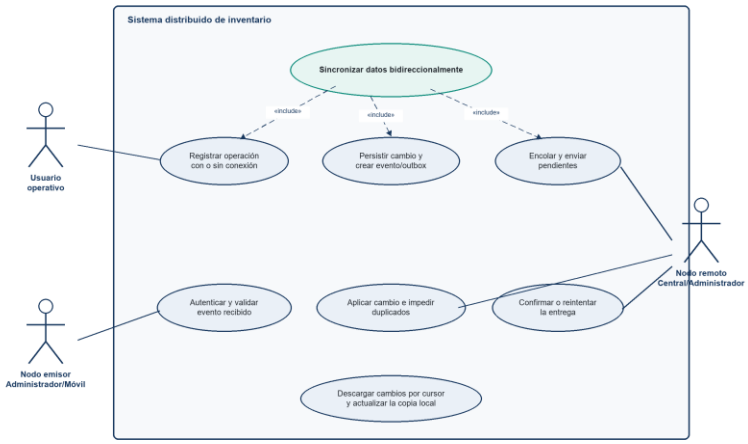


Ilustración 4 Caso de uso: Sincronización bidireccional de datos

Documentación del caso de uso: Sincronización bidireccional de datos	
Caso de uso	Caso de uso N.º 002: Sincronización bidireccional de datos
Fecha y elaboración	01/08/2026 | Responsable del módulo de base de datos y sincronización
Actores	Usuario operativo, nodo emisor Administrador o Móvil y nodo remoto Central o Administrador.
Objetivo	Mantener localmente las operaciones realizadas con o sin conexión y transferirlas en ambos sentidos hasta que las copias autorizadas alcancen consistencia eventual.
Precondiciones	Los nodos están registrados y activos; las bases se encuentran migradas; existe una credencial válida; el worker y el scheduler están operativos; el esquema móvil dispone de SQLite y outbox local.
Postcondiciones	La entidad queda almacenada con su versión, el evento se confirma mediante event_id y la cola pasa al estado synced. Los cambios descargados se aplican y el cursor del nodo avanza.
Medios para realizar la acción	Aplicación web o móvil, base local MySQL o SQLite, worker de sincronización y API HTTPS del nodo remoto.
Pasos	1. El usuario registra, modifica o elimina un dato desde la aplicación. 2. La aplicación verifica la operación mediante el servicio del módulo responsable. 3. La entidad y su evento de salida se almacenan dentro de una transacción local. 4. La aplicación confirma la operación local, incluso cuando no existe conexión. 5. El motor de sincronización obtiene los eventos pending según el orden definido. 6. El nodo emisor transmite event_id, versiones, operación, payload y hash al receptor mediante HTTPS. 7. El receptor autentica el nodo, verifica el hash y valida la idempotencia. 8. El receptor aplica el cambio mediante el adaptador y confirma la nueva versión. 9. El emisor cambia el evento a estado synced al recibir una respuesta satisfactoria. 10. El nodo solicita los cambios posteriores a su cursor, los aplica localmente y actualiza dicho cursor.
Situaciones excepcionales	1. No hay conectividad: el evento continúa en estado y se reintenta con el mismo event_id. 2. El token es inválido o está revocado: el receptor rechaza la solicitud. 3. El evento ya fue procesado: se devuelve el resultado anterior sin repetir sus efectos. 4. La versión base no coincide y se genera un conflicto. 5. El payload, el hash o el tipo de entidad no son válidos. 6. La comunicación se interrumpe después de aplicar el evento: el reintento se reconoce de manera idempotente.

Tabla 22 Documentación del caso de uso: Sincronización bidireccional de datos

4.4.3.2.1 Documentación del caso de uso: Sincronización bidireccional de datos

Revisado por: Tutor del proyecto y equipo de arquitectura

Tabla 2. Documentación del caso de uso: Sincronización bidireccional de datos

4.4.3.3 Caso de uso: Gestión de conflictos y monitoreo

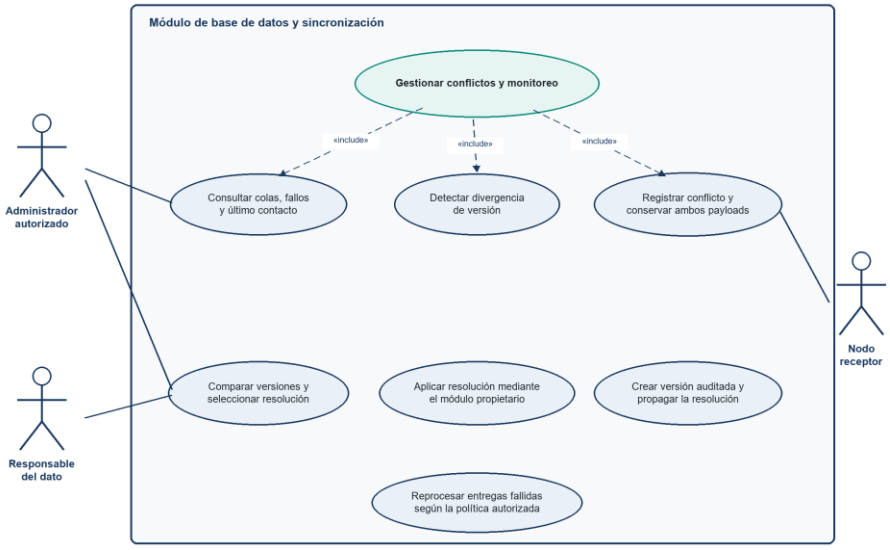


Ilustración 5 Caso de uso: Gestión de conflictos y monitoreo

Documentación del caso de uso: Gestión de conflictos y monitoreo	
Caso de uso	Caso de uso N.° 003: Gestión de conflictos y monitoreo
Fecha y elaboración	01/08/2026 | Responsable del módulo de base de datos y sincronización
Actores	Administrador autorizado, responsable del dato y nodo receptor.
Objetivo	Detectar diferencias, conservar las versiones local y recibida, resolver los conflictos con auditoría y supervisar el estado de las colas y los nodos.
Precondiciones	Existe un conflicto open o una entrega pendiente de revisión; el usuario cuenta con autorización, ambos payloads están disponibles y el servicio del módulo responsable puede validar la decisión.
Postcondiciones	El conflicto pasa a estado resolved con responsable, fecha y tipo de resolución; se genera una nueva versión auditada y su evento se distribuye a los demás nodos.
Medios para realizar la acción	Panel administrativo de sincronización, API protegida, tablas sync_queue y sync_conflicts y servicio del módulo responsable.
Pasos	1. Consultar el estado de los nodos, las colas, los fallos y el último contacto. 2. Seleccionar un conflicto abierto y revisar la entidad, las versiones y la causa. 3. Comparar el payload local con el payload recibido. 4. Elegir entre conservar la versión local, aceptar la entrante o realizar una fusión. 5. Comprobar nuevamente la versión vigente antes de aplicar la decisión. 6. Validar la resolución mediante el servicio del módulo responsable. 7. Generar una nueva versión y un evento de resolución. 8. Registrar resolution, resolved_by y resolved_at. 9. Propagar la versión resultante y actualizar los indicadores de monitoreo.
Situaciones excepcionales	1. La entidad fue modificada durante la revisión: el conflicto continúa abierto y debe analizarse nuevamente. 2. La fusión incumple una regla del módulo responsable. 3. El usuario no cuenta con permisos para resolver conflictos. 4. El nodo remoto está desconectado: la resolución permanece en cola para enviarse posteriormente. 5. El worker o la base de datos no están disponibles y la operación no puede confirmarse.

Tabla 23 Documentación del caso de uso: Gestión de conflictos y monitoreo

4.4.3.3.1 Documentación del caso de uso: Gestión de conflictos y monitoreo

Revisado por: Tutor del proyecto y equipo de arquitectura

Tabla 3. Documentación del caso de uso: Gestión de conflictos y monitoreo

4.4.3.4 Diagramas de Secuencia

Los diagramas siguientes representan la interacción temporal entre actores, servicios, colas y bases de datos para los tres casos de uso principales.

4.4.3.4.1 Diagrama de secuencia: Registro de nodo

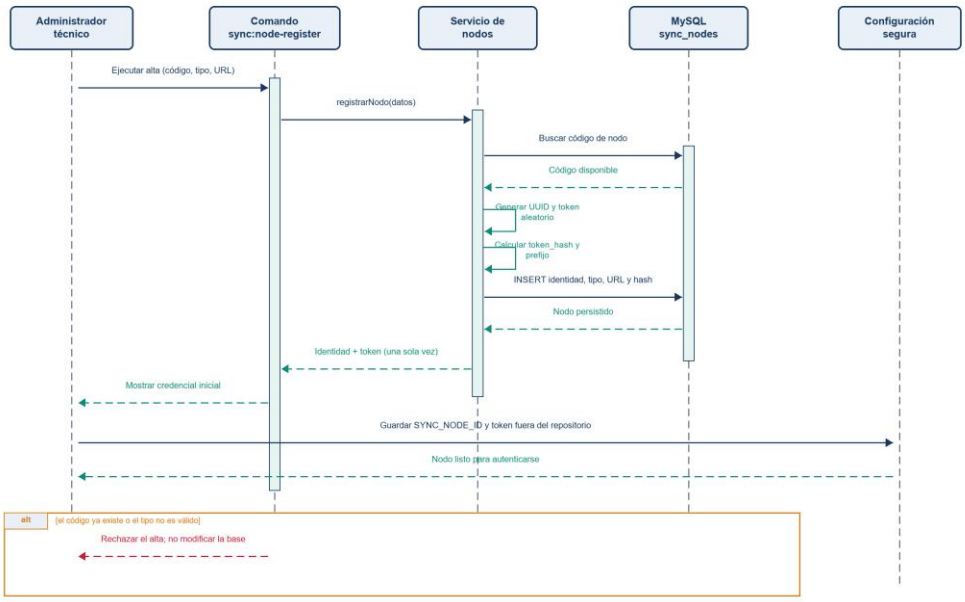


Ilustración 6 Diagrama de secuencia: Registro de nodo

4.4.3.4.2 Diagrama de secuencia: Sincronización bidireccional:

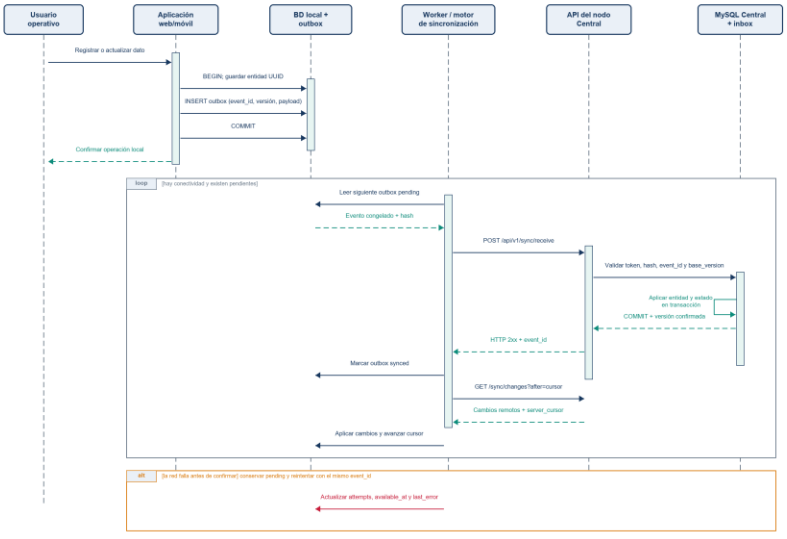


Ilustración 7 Diagrama de secuencia: Sincronización bidireccional

4.4.3.4.3 Diagrama de secuencia: Resolución de conflictos

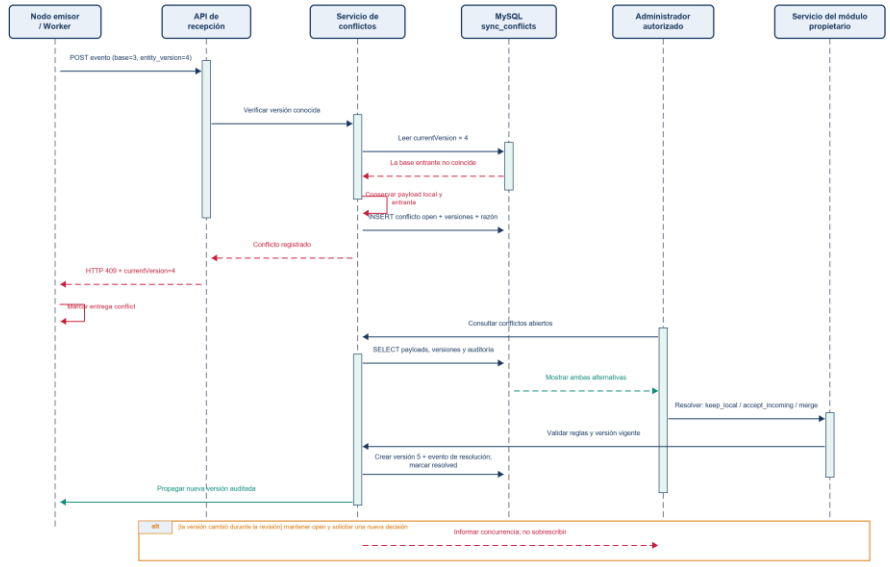


Ilustración 8 Diagrama de secuencia: Detección y resolución de conflictos

4.4.3.5 Diagramas de Estado

Los diagramas de estado describen las condiciones y transiciones controladas por el módulo. Los estados de las entregas se persisten en sync_queue. En el ciclo del conflicto, OPEN y RESOLVED se almacenan en sync_conflicts, mientras que revisión, propagación y convergencia representan etapas operativas del procedimiento.

4.4.3.5.1 Diagrama de estado: Nodo de sincronización

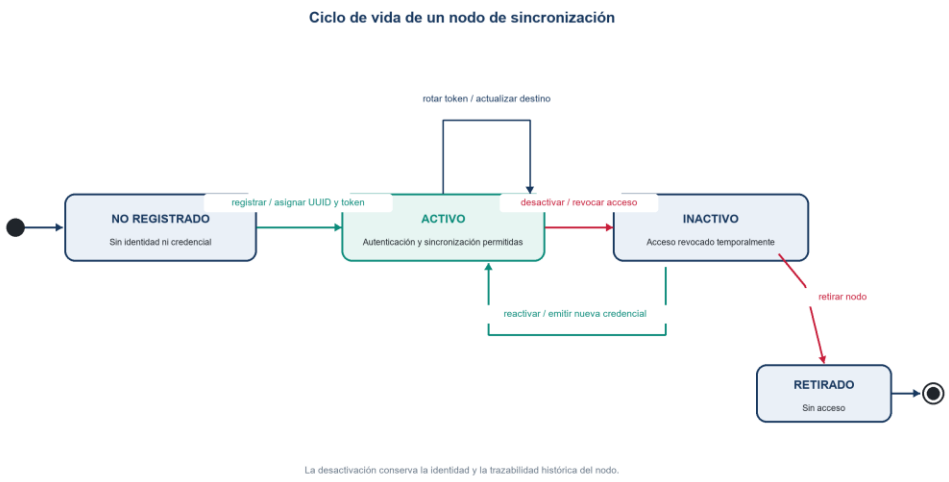


Ilustración 9 Diagrama de estado: Ciclo de vida de un nodo de sincronización

4.4.3.5.2 Diagrama de estado: Entrega de sincronización

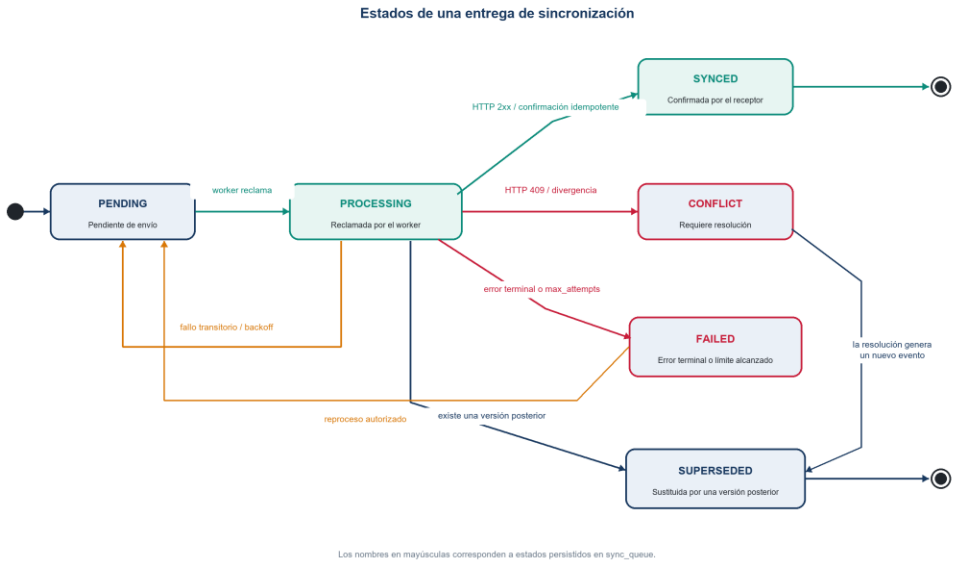


Ilustración 10 Diagrama de estado: Entrega de sincronización

4.4.3.5.3 Diagrama de estado: Conflicto de sincronización

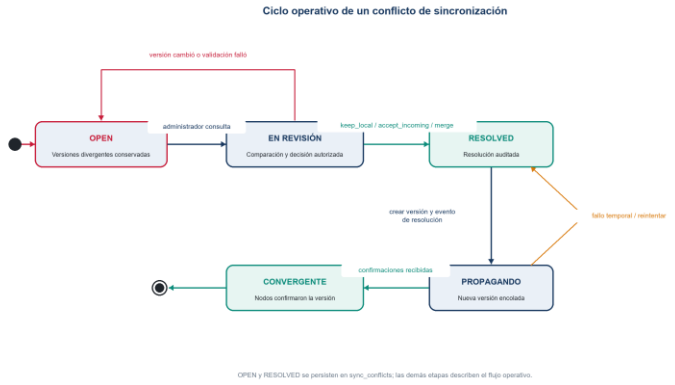


Ilustración 11 Diagrama de estado: Ciclo operativo de un conflicto de sincronización

4.4.3.6 Diagrama de Base de Datos

El modelo separa las tablas técnicas de sincronización de las entidades funcionales. Los nodos Administrador y Central utilizan MySQL; el dispositivo móvil conserva SQLite en almacenamiento privado, una outbox local, el cursor de descarga y los conflictos pendientes. Las líneas discontinuas representan el contrato HTTPS entre bases que nunca comparten directamente sus archivos.

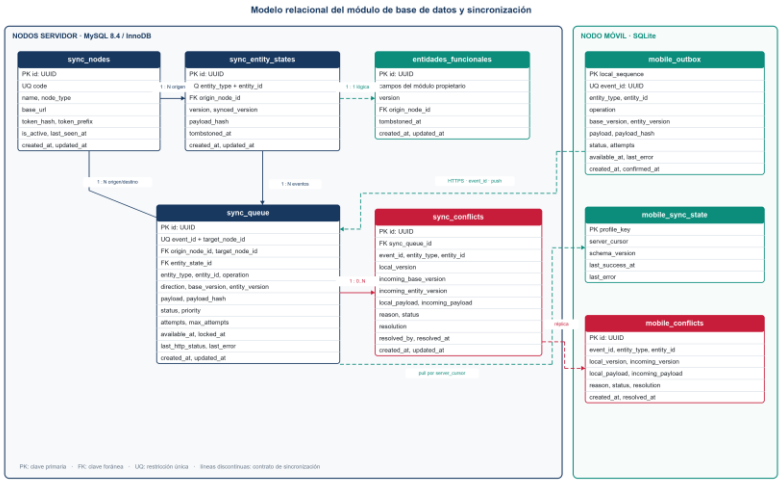


Ilustración 12 Diagrama de base de datos distribuida del módulo

4.4.4 Descripción Técnica / Arquitectura del Sistema

Esta sección describe la arquitectura final del módulo de base de datos y sincronización del sistema de gestión del vivero de cacao. El alcance se concentra en la persistencia de los datos, la identidad de los nodos, el intercambio bidireccional de cambios, el tratamiento de fallos de conectividad y la detección de conflictos; por ello, las tecnologías exclusivas de presentación se mencionan únicamente cuando intervienen en la persistencia móvil.

4.4.4.1 Arquitectura del Sistema

La solución mantiene la arquitectura oficial de monolito modular del proyecto ERP. Los módulos de planificación, inventario, logística, tareas y trazabilidad conservan sus propias reglas, servicios y repositorios, mientras que Synchronization actúa como un módulo transversal.

La propuesta se basa en tres nodos mediante una arquitectura distribuida, cada uno tiene su función específica dentro del proceso de sincronización. El Nodo Administrador, ejecuta la aplicación web desarrollada en Laravel y utiliza una base de datos MySQL para gestionar las operaciones locales, el Nodo Central funciona como punto de integración de la información, empleando la misma aplicación y el mismo esquema de base de datos en un servidor accesible mediante conexiones HTTPS, la identificación de este nodo, junto con su dirección y credenciales, se define a través de la configuración del entorno, finalmente el Nodo Móvil corresponde a la aplicación para Android desarrollada con React y empaquetada mediante capacitor, la cual dispone de una base de datos SQLite privada donde almacena temporalmente las operaciones realizadas cuando el dispositivo no cuenta con acceso a Internet.

El procesamiento de la información inicia cuando un usuario registra, modifica o elimina un dato desde cualquiera de los módulos del sistema. Antes de guardar el cambio, la operación es validada por la lógica de negocio y posteriormente almacenada en la base de datos local mediante

las capas Service y Repository. Una vez completada esta acción, el sistema genera un evento de dominio que es atendido por el componente de sincronización, encargado de actualizar la versión lógica del registro y crear una nueva operación pendiente dentro de la cola sync_queue. Posteriormente, el comando sync:run se ejecuta automáticamente en intervalos programados, evitando ejecuciones simultáneas, mientras que el worker procesa cada operación pendiente utilizando el mismo event_id y el mismo hash durante todos los reintentos, garantizando la consistencia del proceso de sincronización.

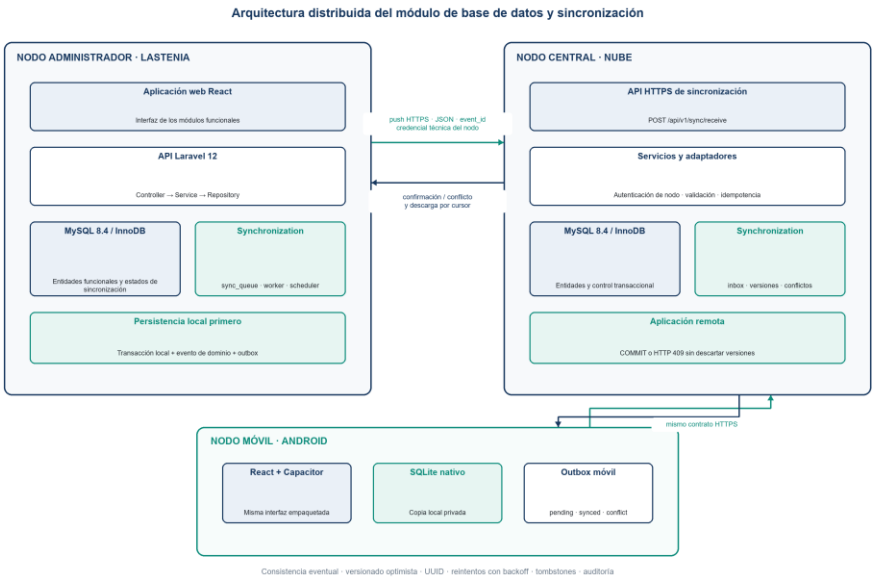


Ilustración 13 Arquitectura distribuida del módulo de base de datos y sincronización

4.4.4.1.1 Tecnologías Utilizadas

En el desarrollo de la propuesta se seleccionaron únicamente las tecnologías necesarias para soportar la arquitectura distribuida, la persistencia de la información y el proceso de sincronización entre los diferentes nodos del sistema. La elección de cada herramienta respondió a los requerimientos funcionales y técnicos identificados durante el análisis de la solución.

El backend del sistema fue implementado con PHP 8.2 y Laravel 12, debido a que este framework proporciona una estructura organizada para desarrollar aplicaciones de gran tamaño y

facilita la separación de responsabilidades mediante controladores, servicios, repositorios, eventos y procesos ejecutados en segundo plano. Sobre esta plataforma se desarrollaron los componentes encargados de la sincronización, incluyendo el servicio que administra los nodos, el receptor de solicitudes, los adaptadores de entidades y el mecanismo de intercambio de información entre los diferentes puntos de la arquitectura (Laravel, 2025a).

Para la gestión de los datos se utilizó Eloquent ORM, lo que permitió trabajar con las entidades del sistema a través de modelos sin perder la organización establecida por el patrón Repository. De igual manera, las migraciones de Laravel fueron empleadas para mantener el control de versiones del esquema de la base de datos y garantizar que su estructura pudiera reproducirse de forma uniforme en los nodos Administrador y Central. Gracias a este mecanismo fue posible administrar de manera controlada tablas como sync_nodes, sync_entity_states, sync_queue y sync_conflicts, necesarias para el funcionamiento del proceso de sincronización (Laravel, 2025b).

Como gestor de datos en los servidores se eligió MySQL 8.4 con el motor InnoDB, ya que ofrece soporte para transacciones, recuperación ante errores, bloqueo a nivel de filas y consistencia en las operaciones. Estas características permiten que los cambios realizados sobre la información funcional y los registros asociados a la sincronización se almacenen como una sola unidad de trabajo, evitando inconsistencias cuando ocurre una interrupción inesperada (Oracle, 2024).

En el caso del dispositivo móvil se empleó SQLite, una base de datos ligera que permanece almacenada dentro del espacio privado de la aplicación. Esta decisión permitió que los usuarios continuaran registrando información aun cuando no existiera conexión a Internet. Las operaciones realizadas en ese período se conservan localmente junto con su correspondiente registro de sincronización hasta que sea posible enviarlas al Nodo Central.

El procesamiento de las tareas pendientes se apoyó en Laravel Queue y Task Scheduler, herramientas que automatizan la ejecución de trabajos en segundo plano. Su utilización permitió programar el envío de información, controlar los reintentos cuando se presentaban fallos temporales de comunicación, registrar errores y mantener el funcionamiento continuo del servicio de sincronización sin afectar la interacción del usuario con la aplicación (Laravel, 2025c).

La comunicación entre los nodos se implementó mediante una API REST, utilizando mensajes en formato JSON transmitidos a través de conexiones HTTPS. Cada nodo cuenta con una identidad propia representada por un UUID y un token técnico, cuyo valor se almacena únicamente como hash para fortalecer la seguridad del sistema. Este mecanismo de autenticación funciona de manera independiente al acceso de los usuarios y permite que únicamente los nodos autorizados participen en el intercambio de información. La autenticación de los usuarios continúa siendo gestionada mediante Laravel Sanctum, mientras que la sincronización entre servidores emplea credenciales exclusivas para este propósito (Fielding et al., 2022; Madden, 2020).

Para garantizar la unicidad de los registros y controlar las modificaciones concurrentes se recurrió al uso de UUID y al versionado optimista. Los identificadores únicos se utilizan tanto en las entidades como en los eventos de sincronización, mientras que el control de versiones permite determinar si un registro ha sido modificado por otro nodo antes de aplicar una actualización. Este mecanismo contribuye a prevenir conflictos y favorece la consistencia de la información distribuida (IETF, 2024).

La aplicación destinada a los dispositivos móviles fue desarrollada con React y posteriormente empaquetada mediante Capacitor, lo que permitió convertir la aplicación web en una aplicación nativa para Android sin mantener proyectos independientes. A través del complemento nativo de SQLite fue posible integrar la base de datos local utilizada durante el

funcionamiento sin conexión, manteniendo una única lógica de negocio y un mismo mecanismo de sincronización para los nodos Administrador, Central y Móvil (Ionic, 2026).

Para asegurarnos de que el sistema funcione sin problemas en el día a día (en producción), tomamos muy en cuenta la seguridad y la estabilidad. ¿Cómo lo logramos? Usamos certificados de seguridad TLS, programamos tareas para que se ejecuten solas, y mantenemos vigilado todo el tiempo el proceso que trabaja en segundo plano. Además, hacemos copias de seguridad de las bases de datos de forma regular, actualizamos las contraseñas técnicas constantemente, y no le quitamos el ojo de encima a las colas de sincronización, los conflictos, los errores y el estado de los equipos (NIST, 2022; Oracle, s. f.).

4.4.4.2 Requerimientos Funcionales

Los requerimientos funcionales delimitan las operaciones que debe proporcionar el módulo de base de datos y sincronización. No incluyen reglas propias de inventario, planificación, logística, tareas o trazabilidad; esas reglas permanecen bajo responsabilidad de sus módulos propietarios.

Código	Requerimiento verificable
RF-SIN-01	Registrar los nodos Administrador, Central y Móvil con identidad UUID, código único, nombre, tipo, URL, estado y credencial técnica. El token inicial se muestra una sola vez y únicamente se almacena su hash.
RF-SIN-02	Configurar la identidad local y el nodo de destino mediante variables de entorno, sin guardar credenciales ni direcciones de producción en el repositorio de código.
RF-SIN-03	Registrar los cambios sincronizables mediante eventos de dominio generados después de que el módulo responsable confirme una creación, actualización o eliminación.
RF-SIN-04	Guardar la versión actual del dato y dejar preparado un paquete en la cola de envíos. Este paquete debe tener su ID único, de dónde viene, a dónde va, qué se hizo y arrancar con estado "pendiente".
RF-SIN-05	Acceder al Nodo Móvil para que registre operaciones sin conexión en SQLite y en mobile_outbox, confirmando la escritura local sin depender de la disponibilidad del Nodo Central.
RF-SIN-06	Que la información fluya en ambas direcciones (entre el Administrador y el Central, y entre el Móvil y el Central) usando un mismo formato y conexiones seguras.
RF-SIN-07	Certificar el nodo emisor y verificar event_id, tipo de entidad, identificador, operación, hash, base_version y entity_version antes de aplicar una entrega recibida.

Código	Requerimiento verificable
RF-SIN-08	Respaldar la estabilidad mediante la combinación de event_id y target_node_id, devolviendo el resultado registrado cuando una entrega se repita y evitando efectos duplicados.
RF-SIN-09	Tratar los estados pending, processing, synced, conflict, failed y superseded; registrar intentos, bloqueo, disponibilidad, código HTTP, error y fechas de confirmación o conflicto.
RF-SIN-10	Reiterar automáticamente los fallos temporales con espera progresiva y conservar en estado failed las entregas que alcancen el máximo de intentos para su revisión o reproceso autorizado.
RF-SIN-11	Descubrir diferencias de versión, responder con HTTP 409 y registrar en sync_conflicts las versiones, payloads y causa, sin eliminar silenciosamente ningún cambio concurrente.
RF-SIN-12	Resolver un conflicto conservando la versión local, aceptando la entrante o fusionando ambas; la decisión debe registrar responsable y fecha, generar una nueva versión y distribuirse a los nodos.
RF-SIN-13	Descargar los cambios posteriores al cursor del nodo, aplicarlos mediante el adaptador del módulo responsable y actualizar el cursor únicamente después de confirmar la transacción local.
RF-SIN-14	Representar las eliminaciones mediante tombstones, supervisar nodos, colas, conflictos y último contacto, y permitir consultar o reprocesar entregas fallidas sin modificar directamente otra base de datos.

Tabla 24 Requerimientos funcionales del módulo de base de datos y sincronización

4.4.4.3 Requerimientos No Funcionales

Los siguientes requisitos establecen atributos medibles de calidad. Los tiempos se evaluarán con el percentil 95 en el ambiente de pruebas definido para la tesis, con la red disponible y una carga nominal documentada; una prueba de carga determinará la capacidad máxima sin sustituir estos criterios de aceptación.

Código	Requerimiento verificable
RNF-SIN-01	Disponibilidad offline: el Nodo Móvil debe permitir consultar y registrar la información autorizada incluso sin conexión; al recuperarse la red, la sincronización debe continuar sin intervención del usuario.
RNF-SIN-02	Evento de salida debe confirmarse localmente en un tiempo máximo de 2 segundos en el percentil 95.
RNF-SIN-03	Una conexión estable, al menos el 95 % de las entregas pendientes debe confirmarse o clasificarse como conflicto dentro de los 75 segundos posteriores a su creación.
RNF-SIN-04	Validar y responder una entrega ordinaria en un tiempo máximo de 2 segundos en el percentil 95, sin considerar la latencia externa de la red.
RNF-SIN-05	Ninguna interrupción de red debe causar la pérdida o duplicación lógica de un cambio; todos los reintentos deben mantener el mismo event_id y payload_hash.

Código	Requerimiento verificable
RNF-SIN-06	Toda comunicación en producción debe realizarse mediante HTTPS con TLS 1.2 o superior; las solicitudes sin una credencial de nodo válida deben rechazarse sin aplicar modificaciones.
RNF-SIN-07	Los tokens deben generarse con aleatoriedad criptográfica, almacenarse como hash, permitir su rotación o revocación y no mostrarse en logs, respuestas posteriores ni repositorios.
RNF-SIN-08	Las escrituras relacionadas deben ejecutarse mediante transacciones ACID y cumplir con claves primarias, foráneas, restricciones únicas, validación de versiones y verificación del hash del payload.
RNF-SIN-09	EL sistema debe alcanzar consistencia eventual y utilizar versionado optimista; no se permitirá resolver automáticamente una divergencia mediante last-write-wins.
RNF-SIN-10	El archivo de la BD SQLite debe conservarse en el almacenamiento privado de la aplicación y no debe compartirse directamente por red ni guardarse en almacenamiento público sin cifrado.
RNF-SIN-11	Los nodos del servidor deben funcionar con MySQL 8.4 e InnoDB, mientras que el nodo móvil debe utilizar SQLite nativo, conservando el mismo esquema lógico, vocabulario de estados y contrato de sincronización.
RNF-SIN-12	Los cambios del esquema deben implementarse mediante migraciones; cada entidad sincronizable debe incorporarse mediante eventos y un adaptador del módulo responsable, sin realizar consultas directas a repositorios externos.
RNF-SIN-13	Se deben registrar el identificador del evento, nodo, entidad, estado, intentos, latencia, código HTTP y error; el monitoreo debe generar alertas ante colas detenidas, conflictos abiertos y nodos sin comunicación.
RNF-SIN-14	Se deben realizar copias de seguridad automáticas, con un RPO máximo de 24 horas y un RTO máximo de 4 horas; el proceso de restauración debe verificarse al menos una vez cada trimestre.
RNF-SIN-15	Operación continua: el scheduler debe ejecutarse mediante cron y el worker mantenerse activo con un supervisor de procesos; el reinicio del servicio no debe eliminar las entregas almacenadas ni los locks vencidos.

Tabla 25 Requerimientos no funcionales del módulo de base de datos y sincronización

4.4.5 Roles y Responsabilidades

TIPO DE USUARIO	ROL	DESCRIPCIÓN
Responsable del vivero	Product Owner	Representa las necesidades operativas del vivero, define las prioridades del módulo y verifica que el almacenamiento y la sincronización de los datos cumplan con el procedimiento establecido.
Tesista responsable del módulo	Scrum Master / Development Team	Responsable de planificar los sprints y desarrollar el modelo de datos, las migraciones, los contratos de intercambio, la configuración de nodos, la sincronización bidireccional, la gestión de conflictos, las pruebas y la documentación técnica.
Propietarios de los módulos funcionales	Development Team	Establecen las reglas de negocio, los DTO, los eventos de dominio y los adaptadores relacionados con sus entidades.

TIPO DE USUARIO	ROL	DESCRIPCIÓN
Tutor y equipo de arquitectura	Stakeholders	Valoran la estructura de la base de datos, la migración a UUID, los mecanismos de seguridad y la integración del módulo de sincronización con los demás componentes del sistema.
Administrador técnico	Stakeholder operativo	Gestionan los nodos, las credenciales técnicas, las variables de entorno, los certificados TLS, el scheduler, el worker, los respaldos y el monitoreo de colas, conflictos y disponibilidad del sistema.

Tabla 26 Roles y responsabilidades del módulo de base de datos y sincronización.

4.4.6 Product Backlog

El Product Backlog reunió las actividades necesarias para diseñar, implementar y validar la infraestructura de base de datos distribuida y sincronización. Los elementos se ordenaron por dependencia técnica y fueron seleccionados progresivamente al inicio de cada sprint. Para evitar la repetición acumulativa del listado completo, en los apartados posteriores se presenta únicamente el Sprint Backlog correspondiente a cada iteración.

N.º	Elemento del Product Backlog	Prioridad	Estado inicial
1	Levantamiento de entidades y flujos de datos del ERP	Alta	Por hacer
2	Diseño del modelo lógico y relacional de la base de datos	Alta	Por hacer
3	Definición de UUID, versionado, auditoría y tombstones	Alta	Por hacer
4	Diseño de tablas para nodos, estados, colas y conflictos	Alta	Por hacer
5	Elaboración del diagrama entidad-relación y diccionario de datos	Alta	Por hacer
6	Creación de migraciones MySQL 8.4 con InnoDB	Alta	Por hacer
7	Implementación de modelos Eloquent y repositorios de sincronización	Alta	Por hacer
8	Registro de nodos y generación segura de credenciales técnicas	Alta	Por hacer
9	Configuración de identidad local y nodo destino por entorno	Alta	Por hacer
10	Pruebas de migración, restricciones, índices y reversión	Alta	Por hacer
11	Implementación del contrato de eventos sincronizables	Alta	Por hacer
12	Integración de adaptadores para los módulos funcionales	Alta	Por hacer
13	Implementación de la cola durable y congelamiento del payload	Alta	Por hacer
14	Configuración del scheduler, comando sync:run y worker	Alta	Por hacer
15	Implementación de reintentos, backoff, locks y estados terminales	Alta	Por hacer
16	Desarrollo del endpoint HTTPS versionado de sincronización	Alta	Por hacer

N.º	Elemento del Product Backlog	Prioridad	Estado inicial
17	Implementación del cliente HTTP y validación del sobre JSON	Alta	Por hacer
18	Recepción idempotente y aplicación transaccional de cambios	Alta	Por hacer
19	Detección y registro de conflictos mediante versionado optimista	Alta	Por hacer
20	Resolución auditada de conflictos y propagación de la nueva versión	Alta	Por hacer
21	Configuración de SQLite nativo y esquema móvil privado	Alta	Por hacer
22	Implementación transaccional del outbox móvil con UUID	Alta	Por hacer
23	Envío de cambios móviles con confirmación y reintentos	Alta	Por hacer
24	Descarga incremental por cursor y aplicación local transaccional	Alta	Por hacer
25	Servicio de consulta de estados y monitoreo de conectividad	Alta	Por hacer
26	Aplicación de TLS, rotación de tokens y límites de seguridad	Alta	Por hacer
27	Ejecución de pruebas unitarias, feature, integración y reconexión	Alta	Por hacer
28	Pruebas E2E con nodos Administrador, Central y Móvil	Alta	Por hacer
29	Configuración de respaldos, restauración, cron, supervisor y monitoreo	Alta	Por hacer
30	Despliegue productivo, documentación operativa y aceptación final	Alta	Por hacer

Tabla 27 Product Backlog inicial

4.4.7 Definición de terminado

La definición de terminado estableció las condiciones mínimas que debía cumplir cada incremento antes de considerarse completo. Estos criterios se utilizaron durante las revisiones para comprobar la integridad de los datos, la seguridad, la trazabilidad y el comportamiento de la sincronización.

4.4.7.1 Criterios generales

- El modelo relacional, las migraciones y las restricciones de integridad fueron revisados y reproducidos en una instancia limpia.
- Las entidades sincronizables utilizan identificadores UUID, versionado lógico, auditoría y tombstones para representar las eliminaciones distribuidas.
- Los contratos de eventos y los adaptadores respetan las reglas definidas por los responsables de los módulos funcionales.

- Cada cambio confirmado se registra en una cola durable dentro de una transacción y conserva el mismo event_id durante sus reintentos.
- El código del módulo cumple la sintaxis de PHP 8.2 y las convenciones de Laravel 12, sin secretos incorporados en el repositorio.
- Las credenciales técnicas se transmiten mediante HTTPS, se muestran una sola vez y únicamente se almacena su hash SHA-256.

4.4.7.2 Criterios específicos del proyecto

- La sincronización funciona en ambos sentidos entre los nodos Administrador y Central mediante POST /api/v1/sync/receive.
- El receptor reconoce la combinación event_id y target_node_id y responde a eventos duplicados sin repetir sus efectos.
- La comparación entre base_version y la versión local detecta divergencias, responde HTTP 409 y conserva ambos payloads en sync_conflicts.
- Los conflictos se resuelven mediante keep_local, accept_incoming o merge; la decisión queda auditada y origina una nueva versión propagable.
- Los paneles de nodos, cola, conectividad y conflictos no muestran tokens completos ni payloads sensibles a usuarios sin autorización.
- Las copias de seguridad, la restauración, el monitoreo del último contacto y las alertas se encuentran documentados y verificados.
- La validación automatizada registró 8 pruebas aprobadas y 38 aserciones para el núcleo de sincronización y los adaptadores funcionales.

4.4.8 Desarrollo iterativo e incremental de los sprints

El desarrollo de los módulos Shared y Synchronization se ejecutó mediante seis sprints. En cada iteración se seleccionaron los elementos prioritarios del Product Backlog, se estableció un objetivo y se produjo un incremento verificable. Esta organización permitió incorporar progresivamente el modelo de datos, la administración de nodos, la cola durable, la sincronización bidireccional, la operación móvil y los controles de seguridad.

En las revisiones participaron el responsable del módulo, el responsable del vivero, los propietarios de los módulos funcionales, la tutora y el administrador técnico. En los siguientes apartados se integran las actividades ejecutadas, las interfaces, las pruebas, el incremento obtenido y la adaptación definida para la iteración siguiente.

4.4.8.1 Sprint 1: Análisis y diseño del modelo de datos distribuido

Duración: 2 semanas (05/01/2026 - 18/01/2026).

Prioridad: Alta.

Objetivo del sprint: Analizar los flujos de información del ERP Lastenia y diseñar el modelo de datos requerido para la persistencia local, la identidad de los nodos y el intercambio controlado de cambios entre las bases distribuidas.

Historias de usuario seleccionadas: HU-01: Gestión segura de equipos, fase de diseño; HU-02: Captura de los cambios que deben sincronizarse, estructura de datos.

4.4.8.1.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
1	Levantamiento de entidades y flujos de datos del ERP	Alta	Completada

N.º	Tarea seleccionada	Prioridad	Estado al cierre
2	Diseño del modelo lógico y relacional de la base de datos	Alta	Completada
3	Definición de UUID, versionado, auditoría y tombstones	Alta	Completada
4	Diseño de tablas para nodos, estados, colas y conflictos	Alta	Completada
5	Elaboración del diagrama entidad-relación y diccionario de datos	Alta	Completada

Tabla 28 Sprint Backlog del Sprint 1

4.4.8.1.2 Actividades ejecutadas y evidencias

Durante el sprint se identificaron las entidades, relaciones y responsables de los datos de los módulos Planning, Tasks, Inventory, Logistics y Tracking. Posteriormente, se elaboró el modelo lógico y relacional, se definieron identificadores UUID, campos de versionado y auditoría, y tombstones para representar eliminaciones. También se diseñaron las tablas sync_nodes, sync_entity_states, sync_queue y sync_conflicts, junto con el diccionario de datos y el mapa de navegación del componente transversal.

Las evidencias principales fueron el modelo entidad-relación, el diccionario de datos y el diseño de la arquitectura distribuida presentados previamente en los apartados técnicos. Como evidencia de la organización funcional de las interfaces se obtuvo el mapa de navegación del módulo.

Ilustración 14 Mapa de navegación del módulo de base de datos y sincronización

4.4.8.1.3 Incremento obtenido

El incremento consistió en el modelo de datos distribuido y la estructura transversal requerida para identificar nodos, controlar versiones, conservar eventos pendientes y registrar conflictos. El diseño permitió establecer qué información debía intercambiarse y proporcionó una base compatible con MySQL y SQLite.

4.4.8.1.4 Revisión y adaptación

En la revisión del 18 de enero de 2026 se comprobó la identificación de entidades, relaciones y propietarios de los datos. El modelo lógico, los UUID, el versionado, la auditoría, los

tombstones y las cuatro tablas transversales fueron revisados por el tutor y los responsables de los módulos.

A partir de la revisión se priorizó para el siguiente sprint la materialización del diseño mediante migraciones reproducibles, modelos Eloquent y servicios para administrar de forma segura la identidad de los nodos.

4.4.8.2 Sprint 2: Esquema MySQL, migraciones e identidad de nodos

Duración: 2 semanas (19/01/2026 - 01/02/2026).

Prioridad: Alta.

Objetivo del sprint: Implementar el esquema transversal en MySQL 8.4 con InnoDB, reproducirlo mediante migraciones y habilitar el registro seguro de los nodos Administrador, Central y Móvil.

Historias de usuario seleccionadas: HU-01: Gestión segura de equipos, fase de implementación; HU-03: Cola de sincronización durable, fase de almacenamiento.

4.4.8.2.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
6	Creación de migraciones MySQL 8.4 con InnoDB	Alta	Completada
7	Implementación de modelos Eloquent y repositorios de sincronización	Alta	Completada
8	Registro de nodos y generación segura de credenciales técnicas	Alta	Completada
9	Configuración de identidad local y nodo destino por entorno	Alta	Completada
10	Pruebas de migración, restricciones, índices y reversión	Alta	Completada

Tabla 29 Sprint Backlog del Sprint 2

4.4.8.2.2 Actividades ejecutadas y evidencias

Se implementaron migraciones reversibles para MySQL 8.4 con InnoDB, claves foráneas, restricciones únicas e índices. Se desarrollaron los modelos Eloquent, enumeraciones, repositorios y servicios para nodos, estados, entregas y conflictos. El servicio de nodos permitió registrar, activar, desactivar y renovar identidades, mostrar la credencial inicial una sola vez y conservar únicamente su hash. La identidad local, la URL de destino y las credenciales se separaron mediante variables de entorno.

La evidencia funcional incluyó la pantalla de gestión de nodos y las verificaciones aplicadas al formulario y al servicio interno SyncNodeService.

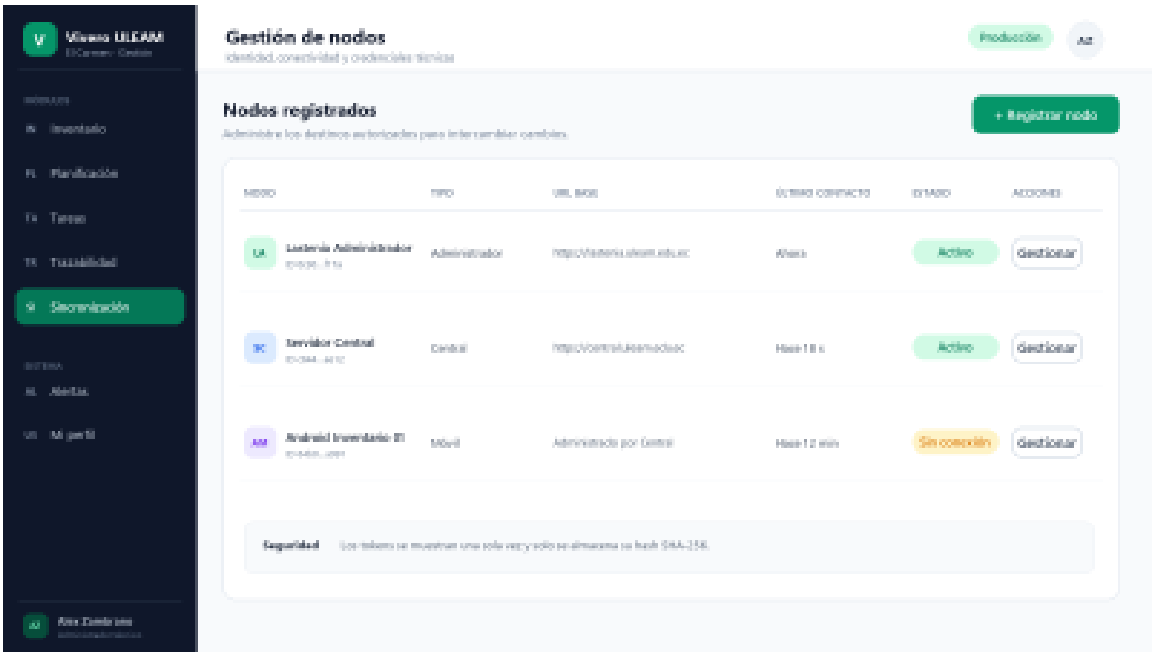


Ilustración 15 Pantalla de gestión de nodos

Campo	Tipo	Valor permitido	Resultado de la verificación
Código del nodo	Texto	Único; entre 3 y 50 caracteres	Rechazó valores vacíos o duplicados.
Nombre	Texto	Entre 3 y 120 caracteres	Se almacenó y mostró correctamente.

Campo	Tipo	Valor permitido	Resultado de la verificación
Tipo	Lista	administrator, central o mobile	Aceptó únicamente los tipos definidos.
URL base	URL	HTTPS para nodos de servidor	Impide registrar destinos de servidor sin HTTPS.
Token técnico	Generado	Cadena aleatoria visible una sola vez	Se almacenó únicamente el hash SHA-256 y el prefijo.

Tabla 30 Verificación del formulario de gestión de nodos

Método	Acción esperada	Resultado obtenido	Evaluación
register()	Crear o actualizar un nodo y emitir un token irrepetible.	Generó 64 caracteres, guardó el hash SHA-256 y devolvió el token una sola vez.	Ruta aprobada; el secreto no quedó en texto claro.
authenticate()	Aceptar únicamente una identidad activa y un token coincidente.	Autenticó con hash_equals y actualizó last_seen_at.	Las credenciales inválidas fueron rechazadas.
ensureConfigured Nodes()	Materializar la identidad local y el destino configurado.	Creó o actualizó ambos nodos sin duplicar el identificador.	Operación idempotente aprobada.

Tabla 31 Pruebas internas del servicio de nodos

4.4.8.2.3 Incremento obtenido

El incremento obtenido fue un esquema transversal reproducible y un servicio de administración de nodos capaz de registrar, activar, revocar y autenticar identidades sin almacenar secretos en texto claro. El último contacto de cada nodo quedó disponible para su monitoreo.

4.4.8.2.4 Revisión y adaptación

En la revisión del 1 de febrero de 2026 se verificaron las migraciones, las restricciones, los modelos Eloquent y los repositorios. También se comprobó la generación segura de tokens y la separación de identidad, destino y credenciales por entorno.

La revisión permitió concentrar el siguiente sprint en la captura automática de cambios, la integración de adaptadores y la persistencia de las entregas antes de iniciar el transporte entre nodos.

4.4.8.3 Sprint 3: Captura de cambios, cola y procesamiento asíncrono

Duración: 3 semanas (02/02/2026 - 22/02/2026).

Prioridad: Alta.

Objetivo del sprint: Capturar las escrituras confirmadas por los módulos funcionales, conservar cada cambio en una cola durable y procesar los envíos mediante scheduler y worker sin perder información ante reinicios o fallos temporales.

Historias de usuario seleccionadas: HU-02: Captura de cambios sincronizables; HU-03: Cola durable de sincronización; HU-05: Reintentos seguros desde el nodo emisor.

4.4.8.3.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
11	Implementación del contrato de eventos sincronizables	Alta	Completada
12	Integración de adaptadores para los módulos funcionales	Alta	Completada
13	Implementación de la cola durable y congelamiento del payload	Alta	Completada
14	Configuración del scheduler, comando sync:run y worker	Alta	Completada
15	Implementación de reintentos, backoff, locks y estados terminales	Alta	Completada

Tabla 32 Sprint Backlog del Sprint 3

4.4.8.3.2 Actividades ejecutadas y evidencias

Se implementó el contrato SyncableDomainEvent, el listener transversal y los adaptadores SyncEntityAdapter para las entidades autorizadas. Cada escritura confirmada generó una entrega con event_id, origen, destino, operación, versiones, payload, hash y estado pending. Se configuraron el comando sync:run, el scheduler, el worker, los bloqueos, el backoff, los intentos y los estados pending, processing, synced, conflict, failed y superseded.

La cola de sincronización permitió consultar y filtrar las entregas, reconocer sus intentos y reactivar únicamente los fallos recuperables sin modificar event_id ni payload_hash.

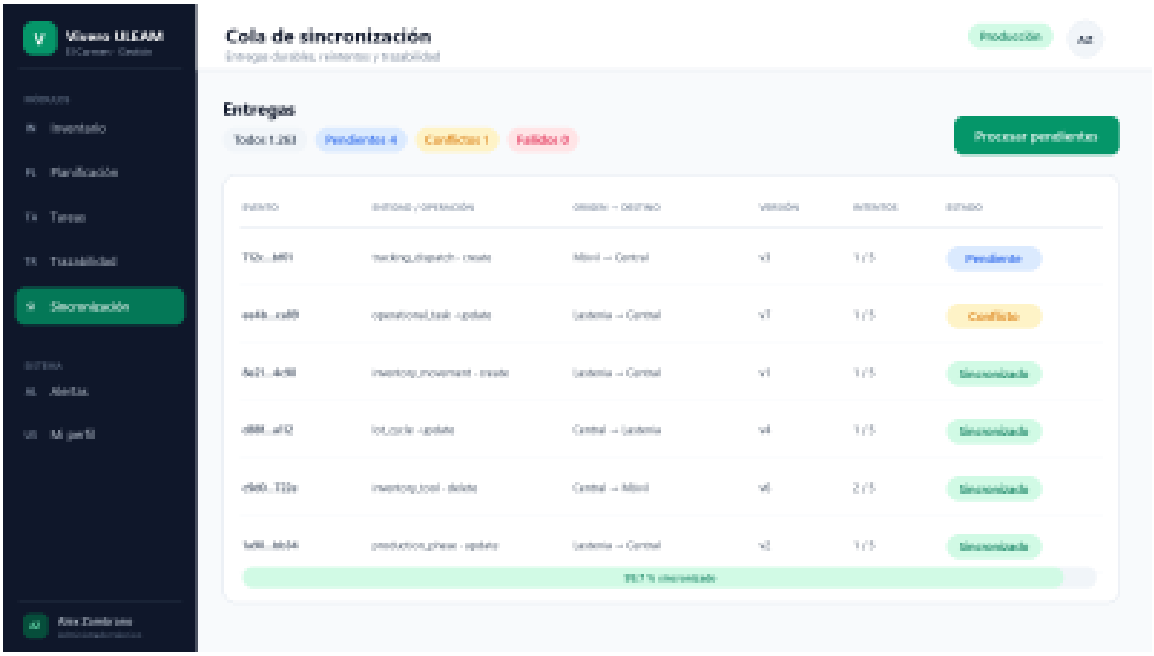


Ilustración 16 Pantalla de la cola de sincronización

Elemento	Tipo	Valor permitido	Resultado de la verificación
Estado	Filtro	pending, processing, synced, conflict, failed o superseded	El listado mostró únicamente las entregas del estado seleccionado.
Dirección	Filtro	inbound u outbound	Diferenció correctamente los cambios recibidos y enviados.
Intentos	Número	Entero entre 0 y el máximo configurado	Se incrementó una vez por cada intento reclamado.
Reintentar	Botón	Solo para fallos recuperables autorizados	Devolvió la entrega a pending sin modificar event_id ni payload_hash.

Tabla 33 Verificación del panel de cola de sincronización

Método	Acción esperada	Resultado obtenido	Evaluación
enqueue(): identidad	Comprobar que origin_node_id coincida con el nodo local.	Bloqueó los intentos cuyo origen no correspondía con la identidad local.	Control de autoría aprobado.
enqueue(): nodos activos	Verificar que el emisor y el destino se encuentren activos.	Impide encolar cambios cuando alguno de los nodos está inactivo.	Control de estado aprobado.
enqueue(): duplicado	Reutilizar una entrega ya existente.	Reconoció event_id y target_node_id sin duplicar la fila.	Protección contra duplicados aprobada.
enqueue(): transacción	Actualizar el estado y crear la entrega de forma atómica.	Actualizó sync_entity_states y sync_queue dentro de DB::transaction.	Prueba transaccional aprobada.

Tabla 34 Pruebas internas del servicio de cola

4.4.8.3.3 Incremento obtenido

El incremento fue un mecanismo durable de captura y procesamiento asíncrono. Cada cambio confirmado quedó almacenado antes de su transporte y pudo recuperarse después de una interrupción. Los reintentos conservaron el mismo evento y el mismo contenido normalizado.

4.4.8.3.4 Revisión y adaptación

En la revisión del 22 de febrero de 2026 se verificó la integración de eventos y adaptadores, la creación de entregas durables y la ejecución del scheduler y del worker. También se comprobaron los bloqueos, el número de intentos, el backoff y los estados terminales.

Con la persistencia y el procesamiento local estabilizados, el siguiente sprint se enfocó en el transporte HTTPS, la recepción idempotente, el control de versiones y la resolución de conflictos.

4.4.8.4 Sprint 4: Sincronización bidireccional, idempotencia y conflictos

Duración: 3 semanas (23/02/2026 - 15/03/2026).

Prioridad: Alta.

Objetivo del sprint: Implementar el intercambio bidireccional por HTTPS entre los nodos Administrador y Central, aplicar los cambios de forma idempotente y conservar las versiones divergentes hasta su resolución controlada.

Historias de usuario seleccionadas: HU-04: Sincronización automática en ambas direcciones; HU-05: Reintentos seguros en el receptor; HU-06: Detección de divergencias; HU-09: Resolución controlada de conflictos.

4.4.8.4.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
16	Desarrollo del endpoint HTTPS versionado de sincronización	Alta	Completada
17	Implementación del cliente HTTP y validación del sobre JSON	Alta	Completada
18	Recepción idempotente y aplicación transaccional de cambios	Alta	Completada
19	Detección y registro de conflictos mediante versionado optimista	Alta	Completada
20	Resolución auditada de conflictos y propagación de la nueva versión	Alta	Completada

Tabla 35 Sprint Backlog del Sprint 4

4.4.8.4.2 Actividades ejecutadas y evidencias

Se implementaron el endpoint HTTPS versionado y el cliente de transporte. El receptor autenticó el nodo emisor, validó el sobre JSON, verificó el hash y reconoció las entregas duplicadas sin repetir sus efectos. La comparación entre base_version y la versión local permitió detectar divergencias, responder HTTP 409 y conservar los payloads local y entrante. La resolución admitió keep_local, accept_incoming y merge, dejó evidencia de la decisión y generó una nueva versión propagable.

Las evidencias incluyeron el contrato de recepción bidireccional, el formulario de resolución y la interfaz que compara las versiones local y entrante antes de aplicar una decisión autorizada.

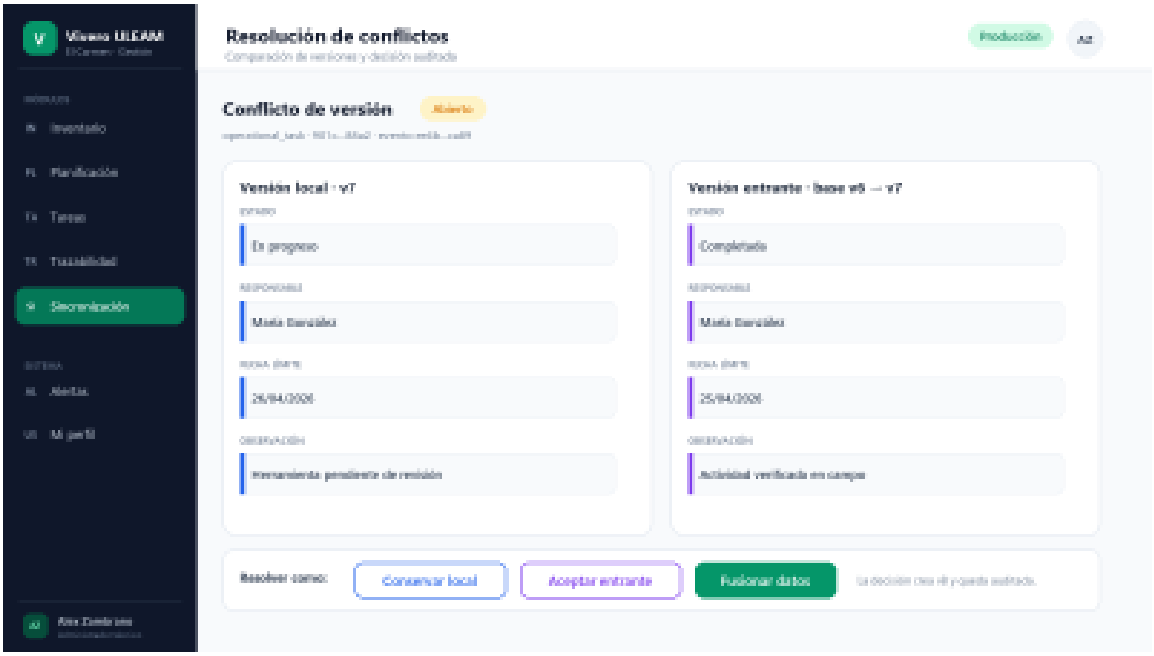


Ilustración 17 Pantalla de resolución de conflictos

Campo	Tipo	Valor permitido	Resultado de la verificación
event_id	UUID	UUID válido y obligatorio	El duplicado devolvió el resultado registrado sin repetir la escritura.
entity_type / operation	Texto / Enum	Entidad registrada; create, update o delete	Las entidades u operaciones desconocidas fueron rechazadas.
base_version	Entero	Versión no negativa	Una base diferente de la versión local generó HTTP 409 y un conflicto.
entity_version	Entero	Mayor que base_version	La versión válida se aplicó y actualizó el estado de la entidad.
payload_hash	Hash	SHA-256 del payload normalizado	Un hash inconsistente impidió aplicar el cambio.

Tabla 36 Contrato de recepción bidireccional

Elemento	Tipo	Valor permitido	Resultado de la verificación
Estrategia	Selección	keep_local, accept_incoming o merge	Obligó a seleccionar una estrategia permitida.
Payload fusionado	Editor JSON	Obligatorio para merge	Impidió fusionar campos inválidos o ajenos a la entidad.
Observación	Texto	Hasta 1000 caracteres	Conservó responsable, fecha y justificación.
Confirmar	Botón	Usuario técnico autorizado	Cerró el conflicto, creó una versión y encoló su propagación.

Tabla 37 Verificación del formulario de resolución de conflictos

Método	Acción esperada	Resultado obtenido	Evaluación
claim()	Reclamar una entrega pending disponible.	Cambió el estado a processing, incrementó attempts y asignó lock_token.	Evitó el procesamiento concurrente.
freezePayload()	Congelar el payload y el hash en el primer intento.	Conservó el mismo contenido durante los reintentos.	Integridad del evento verificada.
handleResponse(): 2xx	Marcar la entrega como synced.	Guardó la versión del receptor, la fecha de éxito y liberó el bloqueo.	Entrega confirmada.
handleResponse(): 409	Registrar el conflicto de versiones.	Conservó las versiones local y entrante.	Disponible para resolución auditada.
markTransportFailure()	Reintentar con backoff o finalizar según el error.	Programó available_at y cambió a failed al alcanzar el límite.	Intento terminal aprobado.

Tabla 38 Pruebas internas del servicio de envío

Método	Acción esperada	Resultado obtenido	Evaluación
receive(): autenticación	Validar la credencial técnica del nodo emisor.	Exigió la credencial antes de consultar o modificar la base.	Acceso inválido rechazado.
receive(): duplicado	Reconocer un evento procesado previamente.	Devolvió el estado anterior sin repetir la modificación.	Idempotencia aprobada.
receive(): integridad	Comparar el hash con el payload normalizado.	Rechazó el evento cuando el hash no coincidió.	Integridad validada.
receive(): versión	Comparar base_version con la versión local bloqueada.	Aplicó la coincidencia y registró conflicto ante la divergencia.	Conflicto aprobado.
receive(): transacción	Aplicar adaptador, actualizar estado y registrar inbound.	Confirmó las tres operaciones de forma atómica y evitó eventos de eco.	Núcleo validado.

Tabla 39 Pruebas internas del servicio de recepción

4.4.8.4.3 Incremento obtenido

El incremento obtenido fue un servicio bidireccional entre los nodos Administrador y Central, con autenticación técnica, recepción idempotente y gestión auditada de conflictos. Las versiones divergentes permanecieron disponibles hasta su resolución y no se sobrescribieron mediante una política automática de último cambio.

4.4.8.4.4 Revisión y adaptación

En la revisión del 15 de marzo de 2026 se comprobó el transporte HTTPS, la autenticación entre nodos, la aplicación idempotente y el registro de duplicados. También se validaron la detección por versionado optimista y las tres estrategias autorizadas de resolución.

Una vez validada la comunicación entre servidores, el siguiente sprint extendió el mismo contrato hacia la persistencia SQLite y el outbox del Nodo Móvil para soportar periodos sin conectividad.

4.4.8.5 Sprint 5: Persistencia móvil sin conexión y sincronización con el Nodo Central

Duración: 3 semanas (16/03/2026 - 05/04/2026).

Prioridad: Alta.

Objetivo del sprint: Habilitar la operación sin conexión en Android mediante SQLite privado, conservar los cambios en el outbox móvil y descargar actualizaciones incrementales desde el Nodo Central al recuperarse la conectividad.

Historias de usuario seleccionadas: HU-04: Sincronización automática desde la aplicación móvil; HU-07: Operación móvil sin conexión; HU-08: Consulta del estado de sincronización.

4.4.8.5.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
21	Configuración de SQLite nativo y esquema móvil privado	Alta	Completada
22	Implementación transaccional del outbox móvil con UUID	Alta	Completada
23	Envío de cambios móviles con confirmación y reintentos	Alta	Completada
24	Descarga incremental por cursor y aplicación local transaccional	Alta	Completada
25	Servicio de consulta de estados y monitoreo de conectividad	Alta	Completada

Tabla 40 Sprint Backlog del Sprint 5

4.4.8.5.2 Actividades ejecutadas y evidencias

Se configuró SQLite en el almacenamiento privado del dispositivo y se implementó mobile_outbox dentro de la misma transacción utilizada para confirmar las operaciones funcionales. Se desarrollaron el envío por lotes, la conservación de event_id, los reintentos, la

descarga incremental por server_cursor y la aplicación transaccional de los cambios. La interfaz mostró la conectividad, las operaciones pendientes, los conflictos y la fecha del último contacto.

La pantalla móvil permitió reconocer si el dispositivo trabajaba en línea o sin conexión y mostró la cantidad de cambios pendientes antes de iniciar la sincronización.

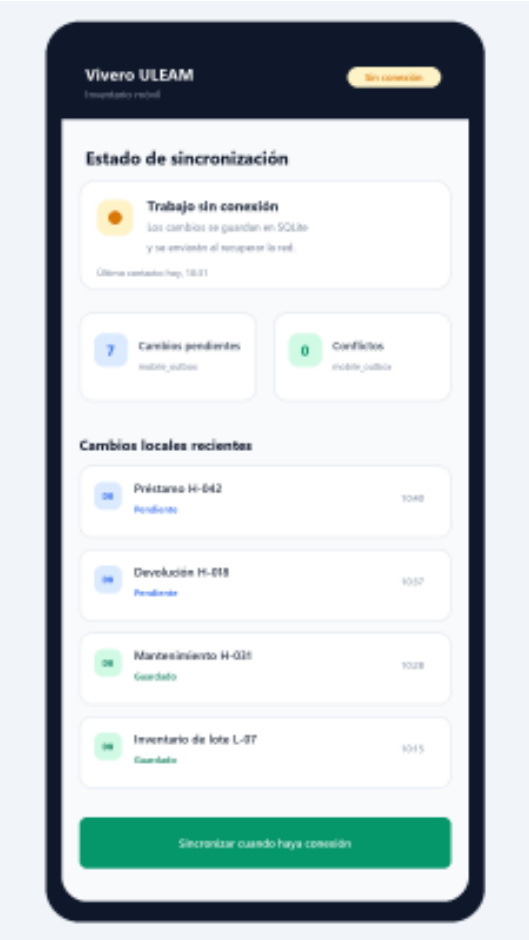


Ilustración 18 Pantalla de estado de sincronización móvil

Elemento	Tipo	Valor permitido	Resultado de la verificación
Conectividad	Indicador	online u offline	La pérdida de red no bloqueó el registro local.
Cambios pendientes	Contador	Entradas activas en mobile_outbox	Se actualizó después de cada escritura y confirmación.
Sincronizar	Botón	Con conectividad y sesión técnica válida	Envió pendientes, conservó event_id y descargó desde server_cursor.
Último contacto	Fecha y hora	Marca UTC convertida a la zona local	Permitió reconocer la actualización de los datos.

Tabla 41 Verificación del estado de sincronización móvil

4.4.8.5.3 Incremento obtenido

El incremento fue una persistencia móvil privada con capacidad de registrar operaciones sin conexión. Cuando se restableció la red, el outbox transmitió los cambios pendientes y el dispositivo descargó las actualizaciones posteriores a su cursor sin compartir directamente el archivo SQLite.

4.4.8.5.4 Revisión y adaptación

En la revisión del 5 de abril de 2026 se verificaron la base SQLite, la confirmación transaccional del outbox, el envío de lotes, la conservación del event_id y la descarga incremental. También se revisaron los estados pending, synced, conflict y failed.

El último sprint se destinó a fortalecer la seguridad, ejecutar las pruebas integrales, comprobar la recuperación ante fallos y documentar los procedimientos de respaldo y monitoreo.

4.4.8.6 Sprint 6: Seguridad, pruebas, respaldo y monitoreo

Duración: 3 semanas (06/04/2026 - 26/04/2026).

Prioridad: Alta.

Objetivo del sprint: Validar integralmente la base de datos distribuida y la sincronización, aplicar controles de seguridad y dejar documentados el monitoreo, los respaldos y el procedimiento de recuperación.

Historias de usuario seleccionadas: HU-10: Monitoreo, respaldo y recuperación; HU-01 a HU-09: Validación integral del incremento.

4.4.8.6.1 Sprint Backlog

N.º	Tarea seleccionada	Prioridad	Estado al cierre
26	Aplicación de TLS, rotación de tokens y límites de seguridad	Alta	Completada
27	Ejecución de pruebas unitarias, feature, integración y reconexión	Alta	Completada
28	Pruebas E2E con nodos Administrador, Central y Móvil	Alta	Completada
29	Configuración de respaldos, restauración, cron, supervisor y monitoreo	Alta	Completada
30	Despliegue productivo, documentación operativa y aceptación final	Alta	Completada

Tabla 42 Sprint Backlog del Sprint 6

4.4.8.6.2 Actividades ejecutadas y evidencias

Se aplicaron TLS, rotación y revocación de tokens, límites de solicitudes y de payload, y exclusión de secretos en registros y respuestas. Se ejecutaron pruebas unitarias, feature, de integración, reconexión, duplicados y divergencias de versión. También se configuraron el scheduler, el worker supervisado, los respaldos, la restauración, las métricas, las alertas y el monitoreo del último contacto.

Las evidencias de cierre incluyeron el inicio de sesión institucional, el panel de monitoreo y el resumen de la validación automatizada y operativa.

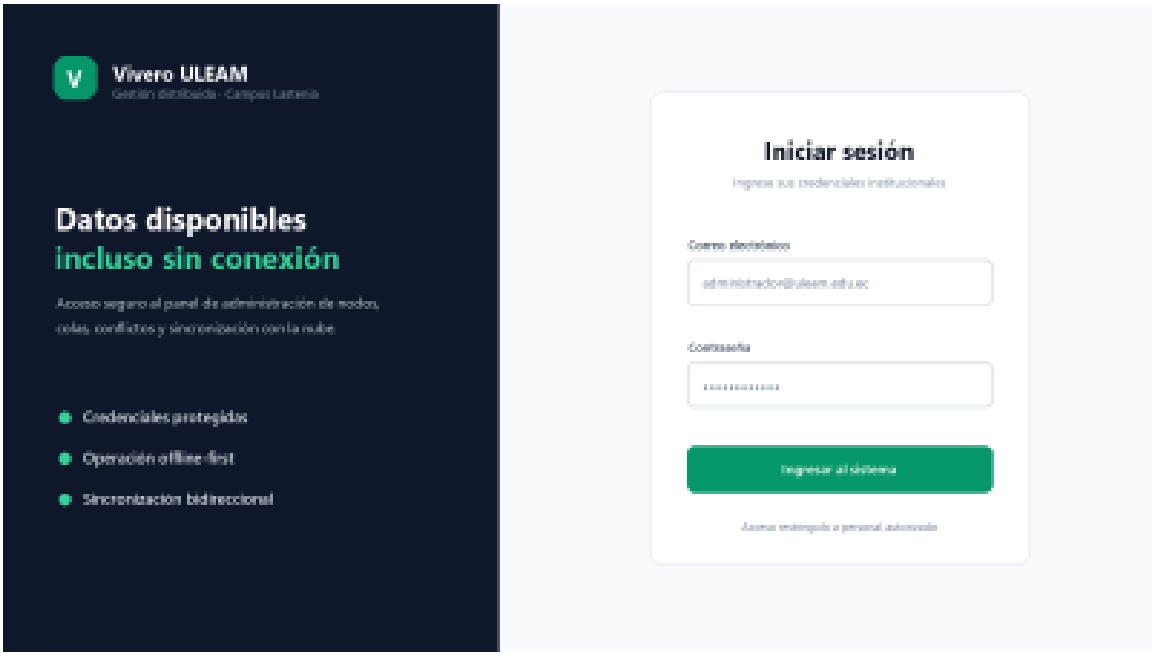


Ilustración 19 Pantalla de inicio de sesión

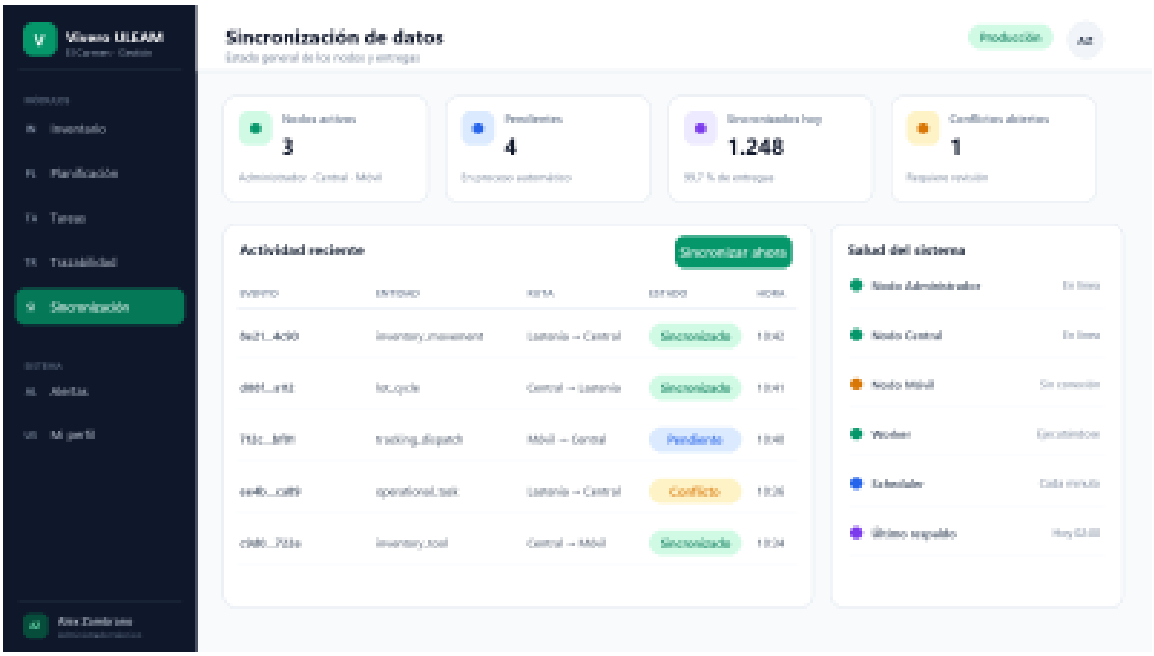


Ilustración 20 Panel de monitoreo de sincronización

Indicador	Resultado	Evaluación
Pruebas generales del repositorio	73 aprobadas	Satisfactorio

Indicador	Resultado	Evaluación
Aserciones generales	322 aprobadas	Satisfactorio
Pruebas del núcleo de sincronización	8 aprobadas	Satisfactorio
Aserciones del núcleo	38 aprobadas	Satisfactorio
Comunicación Administrador-Central	Bidireccional	Operativa
Nodo Móvil con SQLite	Outbox y cursor	Operativo
Respaldo, restauración y monitoreo	Procedimiento validado	Operativo

Tabla 43 Resumen de la validación del Sprint 6

4.4.8.6.3 Incremento obtenido

El incremento final integró la infraestructura de datos distribuida, la sincronización automática, los controles de seguridad, las pruebas y los procedimientos operativos. La validación general del repositorio registró 73 pruebas y 322 aserciones aprobadas; el núcleo de sincronización registró 8 pruebas y 38 aserciones aprobadas.

4.4.8.6.4 Revisión y adaptación

En la revisión del 26 de abril de 2026 se comprobaron los escenarios de reconexión, duplicados y conflictos, junto con los mecanismos de seguridad, respaldo, restauración y monitoreo. El responsable del vivero revisó el procedimiento operativo correspondiente al módulo.

Las mejoras posteriores deberán gestionarse como nuevos elementos del Product Backlog, manteniendo la observación de colas, conflictos, errores, respaldos y último contacto de los nodos durante la operación.

CAPÍTULO V

5 EVALUACIÓN DE RESULTADOS

5.1 Introducción

Este capítulo presenta los resultados de la evaluación técnica del módulo de base de datos y sincronización del sistema informático para la gestión del inventario de herramientas del vivero de cacao. La revisión se concentró en la estructura del esquema, la integridad de los datos, la captura de cambios, la cola durable, la recepción autenticada, la idempotencia, los reintentos, el versionado optimista y la comunicación entre los nodos Administrador y Central.

La evaluación se realizó mediante la inspección del repositorio, la revisión de las migraciones, la ejecución documentada de pruebas automatizadas y el análisis de escenarios funcionales en un ambiente controlado. El Nodo Móvil se consideró únicamente en el nivel de diseño, porque en el repositorio revisado no se identificaron una aplicación Android con Capacitor, el complemento SQLite nativo, la tabla mobile_outbox ni el servicio de descarga incremental por cursor. Por esta razón, dichos componentes no se presentan como funciones implementadas o validadas.

Los resultados técnicos se separaron de los efectos operativos observados en el vivero. La encuesta inicial permitió establecer la incidencia de las causas del problema, pero no se dispone de una medición posterior a la utilización del sistema. En consecuencia, el capítulo informa los resultados comprobados y no atribuye porcentajes de mejora que todavía no cuentan con evidencia de campo.

5.2 Presentación y monitoreo de resultados

5.2.1 Planificación de la evaluación

La evaluación se organizó de acuerdo con los componentes desarrollados, los criterios de aceptación definidos en el capítulo IV y la evidencia disponible en el repositorio.

Proceso evaluado	Método de validación	Resultado esperado
Diseño de la base de datos	Revisión de migraciones, tablas, claves, restricciones e índices.	Comprobar que el esquema puede reproducirse de manera consistente.
Captura y cola de cambios	Revisión de eventos de dominio, adaptadores, estados de entidad y registros de la cola.	Conservar cada cambio pendiente con identidad, versión, destino y trazabilidad.
Sincronización Administrador-Central	Pruebas automatizadas de envío, recepción, autenticación, validación e idempotencia.	Procesar cambios en ambos sentidos sin duplicar sus efectos ni generar ciclos de eco.
Fallos, seguridad y conflictos	Escenarios con reintentos, eventos duplicados, token inválido, hash alterado y versiones divergentes.	Rechazar datos inválidos, recuperar fallos transitorios y registrar conflictos.
Diseño del Nodo Móvil	Revisión de la arquitectura prevista para SQLite, outbox y cursor incremental.	Determinar el alcance diseñado y diferenciarlo de la implementación comprobada.
Impacto sobre las causas	Comparación de la encuesta diagnóstica con una medición posterior equivalente.	Calcular la variación porcentual de cada causa con datos obtenidos después del uso del sistema.

Tabla 44. Planificación de la evaluación

5.2.2 Ejecución del monitoreo

5.2.2.1 Evaluación de la base de datos

La revisión identificó 42 archivos de migración que contienen 44 instrucciones de creación de tablas. Al ejecutarse, Laravel incorpora además la tabla de control de migraciones, por lo que el esquema completo alcanza 45 tablas. Dentro de este conjunto, la infraestructura transversal implementada se concentra en sync_nodes, sync_entity_states, sync_queue y sync_conflicts.

La tabla sync_nodes identifica los nodos autorizados; sync_entity_states conserva la versión conocida de cada entidad; sync_queue registra las entregas, sus estados, intentos y bloqueos; y sync_conflicts almacena las divergencias detectadas. Las tablas transversales utilizan UUID, claves foráneas, restricciones únicas e índices orientados al procesamiento de la cola y a la auditoría.

La relación con los datos funcionales se realiza mediante el tipo de entidad, su identificador y el adaptador correspondiente. Se verificó la configuración de siete adaptadores para Lot, LotCycle, OperationalTask, Movement, TrackingClient, TrackingMovement y Dispatch. No se comprobó una migración general de los identificadores de todas las entidades funcionales a UUID; por ello, esa actividad debe mantenerse como incremento pendiente y no como resultado alcanzado.

5.2.2.2 Evaluación de la sincronización Administrador-Central

Las pruebas comprobaron que una operación funcional confirmada puede generar un evento de dominio y una entrega durable en estado pending. El proceso de envío obtiene la entrega, construye el payload y la remite al endpoint configurado. El receptor autentica el nodo de origen, verifica la estructura y el hash, compara las versiones y aplica la operación mediante el adaptador de la entidad dentro de una transacción.

Cuando una entrega es aceptada, su estado cambia a synced y se actualiza la versión registrada. Los eventos repetidos se reconocen mediante la combinación de event_id y target_node_id, por lo que un reintento devuelve el resultado previo sin ejecutar nuevamente el adaptador. SynchronizationContext impide que un cambio recibido se vuelva a publicar como un evento nuevo y evita el ciclo Administrador-Central-Administrador.

La comunicación Administrador-Central fue validada mediante pruebas automatizadas con solicitudes HTTP simuladas y una base SQLite de pruebas. Esta evidencia demuestra el comportamiento lógico del núcleo en un ambiente controlado, pero no equivale a una validación productiva con dos servidores MySQL independientes, certificados TLS, workers permanentes y una red real.

5.2.2.3 Evaluación de fallos, seguridad y conflictos

Los errores temporales conservan la entrega para un nuevo intento y registran el número de ejecuciones, el último error y la fecha de disponibilidad. La recepción rechaza credenciales inválidas y payloads cuyo hash no corresponde con el contenido. Cuando base_version no coincide con la versión local, el sistema evita la sobrescritura silenciosa, marca la entrega como conflict y registra las versiones local y entrante en sync_conflicts.

La evidencia del repositorio permite comprobar la detección y el registro de conflictos. Sin embargo, no se identificó un flujo implementado para que un usuario autorizado resuelva el conflicto mediante las estrategias keep_local, accept_incoming o merge. La pantalla presentada en el capítulo IV debe considerarse un prototipo hasta que exista el servicio, la ruta, la autorización y las pruebas correspondientes.

5.2.2.4 Estado del Nodo Móvil y de la operación sin conexión

El Nodo Móvil fue diseñado con una base SQLite privada, una cola de salida local y un cursor para la descarga incremental. El diseño establece que cada operación funcional y su evento pendiente deben confirmarse en una misma transacción, conservar event_id durante los reintentos y avanzar el cursor únicamente después de aplicar los cambios recibidos.

No obstante, la revisión del repositorio no evidenció dependencias de Capacitor, un proyecto Android, el complemento SQLite nativo, migraciones móviles, mobile_outbox ni servicios de envío y descarga. En consecuencia, la operación móvil sin conexión permanece como diseño técnico y no puede calificarse como operativa, desplegada o validada.

5.2.2.5 Resultados de las pruebas automatizadas

La ejecución documentada del repositorio registró 73 pruebas aprobadas y 322 aserciones. Dentro de este conjunto, ocho pruebas con 38 aserciones estuvieron relacionadas directamente con la infraestructura de sincronización y su integración con los módulos Planning, Tasks, Inventory y Tracking. Estos resultados respaldan el núcleo implementado, pero deben interpretarse dentro del ambiente y del alcance descritos anteriormente.

Indicador	Resultado	Evaluación
Pruebas generales del repositorio	73 aprobadas	Satisfactorio según la ejecución documentada
Aserciones generales	322 aprobadas	Satisfactorio según la ejecución documentada
Pruebas del núcleo de sincronización	8 aprobadas	Satisfactorio en ambiente controlado
Aserciones del núcleo	38 aprobadas	Satisfactorio en ambiente controlado
Tablas transversales	4 implementadas	Comprobado en las migraciones
Tipos de entidad integrados	7 adaptadores	Integración parcial del dominio funcional
Comunicación Administrador-Central	HTTP simulado	Validada en ambiente de pruebas
Detección de conflictos	Implementada	Registro comprobado; resolución manual pendiente
Nodo Móvil con SQLite	Diseño documentado	Pendiente de implementación y validación

Tabla 45. Resumen de los resultados técnicos obtenidos

5.2.3 Comparación de resultados

La comparación distingue las limitaciones del control manual, las capacidades verificadas en el núcleo implementado y los componentes que permanecen en diseño. Esta separación evita atribuir al sistema funciones que todavía no cuentan con evidencia de implementación.

Aspecto	Situación inicial	Resultado comprobado o alcance actual
Persistencia	Registros manuales o datos aislados.	Esquema servidor reproducible mediante migraciones y cuatro tablas transversales.
Trazabilidad	No existía un historial técnico verificable de los cambios.	La cola conserva evento, entidad, operación, origen, destino, versiones, estado e intentos.
Recuperación ante fallos	Los registros podían perderse, retrasarse o duplicarse.	Las entregas pendientes se conservan y los errores temporales admiten reintentos controlados.
Duplicados	Una operación repetida podía registrarse más de una vez.	La idempotencia reconoce event_id y target_node_id y evita efectos duplicados.
Cambios simultáneos	Las diferencias podían pasar inadvertidas.	El versionado detecta y registra conflictos; la resolución manual permanece pendiente.
Intercambio entre servidores	No existía sincronización automática.	El flujo Administrador-Central fue validado con solicitudes HTTP simuladas.
Operación móvil	No existía una alternativa para trabajar sin conexión.	La arquitectura SQLite y outbox está diseñada, pero no implementada ni validada.
Impacto operativo	Se dispone de una línea base de las causas del problema.	No existe medición posterior; el porcentaje de mejora todavía no puede determinarse.

Tabla 46. Comparación entre la situación inicial y el alcance comprobado

5.2.4 Cumplimiento de los objetivos

El grado de cumplimiento se estableció comparando cada objetivo específico con la evidencia disponible. Se utilizó la categoría parcial cuando una parte del resultado fue diseñada o implementada, pero todavía requiere integración o validación adicional.

Objetivo específico	Evidencia disponible	Estado
Analizar los fundamentos y requisitos del diseño de bases de datos distribuidas, la operación sin conexión y la sincronización lógica.	Marco teórico, diagnóstico del vivero y criterios técnicos de autonomía, integridad, consistencia eventual y tolerancia a fallos.	Cumplido
Diseñar el modelo lógico y físico, la topología de nodos y el contrato portable para MySQL y SQLite.	Modelo de tres nodos, esquema servidor, diseño SQLite, metadatos, versiones, cola, conflictos y contrato JSON.	Cumplido a nivel de diseño
Implementar eventos, cola durable, adaptadores, autenticación, idempotencia, reintentos, tombstones y versionado optimista.	Núcleo servidor implementado con cuatro tablas transversales y siete adaptadores; Nodo Móvil y resolución manual pendientes.	Cumplido parcialmente
Demostrar el funcionamiento de los procesos automatizados mediante pruebas unitarias, funcionales y de integración.	73 pruebas y 322 aserciones documentadas; ocho pruebas y 38 aserciones del núcleo; comunicación Administrador-Central simulada.	Cumplido parcialmente

Tabla 47. Cumplimiento de los objetivos específicos

5.3 Interpretación objetiva

La encuesta diagnóstica aplicada a 60 estudiantes permitió establecer la incidencia inicial de tres causas incluidas en el árbol del problema. El 86,7 % indicó haber observado herramientas perdidas o dañadas de manera frecuente u ocasional; el 90,0 % manifestó haber experimentado errores, confusiones o pérdida de registros por el manejo manual; y el 73,3 % señaló que la revisión del estado de las herramientas durante la devolución se realizaba solamente algunas veces o no se efectuaba. Este último indicador se utilizó como aproximación al desconocimiento del estado de los materiales.

Causa del problema	Incidencia inicial	Incidencia posterior	Porcentaje de mejora
Extravío frecuente de herramientas	86,7 %	No medida	No calculable
Errores humanos en el registro manual de insumos	90,0 %	No medida	No calculable

Causa del problema	Incidencia inicial	Incidencia posterior	Porcentaje de mejora
Desconocimiento del estado de los materiales	73,3 %	No medida	No calculable

Tabla 48. Línea base y estado de la medición de las causas del problema

El porcentaje de mejora debe calcularse únicamente después de aplicar el mismo instrumento, o uno metodológicamente equivalente, tras un periodo definido de utilización del sistema. La medición posterior debe mantener la población objetivo, las categorías de respuesta y la forma de agrupar las respuestas, con el fin de que la comparación sea válida.

Mejora (%) = ((incidencia inicial - incidencia posterior) / incidencia inicial) x 100

Las 73 pruebas y 322 aserciones demuestran el comportamiento técnico documentado del software, pero no prueban por sí mismas una reducción del extravío, de los errores de registro o del desconocimiento del estado de los materiales. Por tanto, con la evidencia disponible puede afirmarse la viabilidad técnica del núcleo de sincronización y su funcionamiento controlado entre Administrador y Central, pero no puede establecerse todavía el impacto porcentual sobre las causas operativas del vivero.

Para completar la evaluación exigida por la rúbrica, se deberá ejecutar un piloto con usuarios, registrar la fecha y duración de la intervención, aplicar nuevamente las preguntas asociadas a cada causa y sustituir en la Tabla 48 las expresiones «No medida» y «No calculable» por los valores obtenidos y calculados. Hasta que esa evidencia exista, cualquier porcentaje de mejora sería una estimación sin respaldo empírico.

CAPÍTULO VI

6 CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- En relación con el primer objetivo específico, la revisión bibliográfica y el diagnóstico realizado mediante la encuesta y la entrevista permitieron establecer los fundamentos y requisitos necesarios para la arquitectura distribuida. Se determinó que el sistema debía garantizar persistencia local, integridad, trazabilidad, consistencia eventual y recuperación de los cambios ante interrupciones de conectividad, considerando las deficiencias identificadas en el registro manual del inventario.
- Respecto al segundo objetivo específico, se diseñó el modelo lógico y físico mediante la separación de los datos funcionales y los componentes transversales de sincronización. La definición de la topología conformada por los nodos Administrador, Central y Móvil, junto con el uso de MySQL, SQLite, identificadores UUID, restricciones, índices, transacciones y un contrato portable de cambios, permitió establecer una estructura compatible e íntegra para el intercambio de información.
- En cumplimiento del tercer objetivo específico, se implementó la infraestructura de sincronización mediante eventos de dominio, adaptadores, colas durables, autenticación entre nodos, idempotencia, reintentos controlados, tombstones y versionado optimista. Estos mecanismos permiten conservar las operaciones pendientes, evitar efectos duplicados, detectar modificaciones concurrentes y registrar los conflictos sin sobrescribir silenciosamente la información.

- En cuanto al cuarto objetivo específico, las pruebas automatizadas y funcionales permitieron verificar el comportamiento de los componentes desarrollados. La validación general registró 73 pruebas y 322 aserciones aprobadas, de las cuales 8 pruebas y 38 aserciones correspondieron directamente al núcleo de sincronización. Asimismo, la comunicación bidireccional entre los nodos Administrador y Central fue comprobada en un ambiente controlado, incluyendo escenarios de recepción, reintentos, duplicados y conflictos de versión. Estos resultados demuestran el cumplimiento técnico del objetivo general dentro del alcance validado de la investigación.

6.2 Recomendaciones

- Se recomienda al administrador técnico del sistema realizar un monitoreo periódico de las colas de sincronización, los conflictos abiertos, los errores registrados, la antigüedad de las operaciones pendientes y el último contacto de cada nodo, con la finalidad de detectar oportunamente interrupciones o fallos en el intercambio de información.
- Se recomienda al administrador técnico, con el apoyo del personal de Tecnologías de la Información de la ULEAM, establecer y ejecutar un plan de respaldos para las bases de datos MySQL y SQLite. Las copias deben incluir la información funcional, las colas, los estados de las entidades, los conflictos y la configuración necesaria para restablecer el servicio. Asimismo, se deben efectuar pruebas periódicas de restauración para comprobar la validez de los respaldos.
- Se recomienda al administrador técnico gestionar las credenciales de los nodos mediante procedimientos de renovación, rotación y revocación. También deberá resguardar los secretos fuera del repositorio del proyecto, mantener actualizados los certificados TLS y evitar que los

tokens o datos sensibles se expongan en registros, respuestas o archivos de configuración compartidos.

- Se recomienda al equipo de desarrollo y al responsable de pruebas ejecutar evaluaciones periódicas después de cada modificación relevante del sistema. Estas pruebas deben incluir escenarios de desconexión, reconexión, reenvío de operaciones, duplicados, conflictos de versión, reinicio del servicio, cierre inesperado de la aplicación móvil y restauración de respaldos.
- Se recomienda al responsable del vivero, con el apoyo del equipo investigador o de la coordinación de la carrera, aplicar nuevamente la encuesta después de un periodo definido de utilización del sistema. Esta medición permitirá comparar los resultados con la línea base y calcular objetivamente el porcentaje de reducción del extravío de herramientas, los errores de registro y el desconocimiento del estado de los materiales.

BIBLIOGRAFÍA

Argoti Caiza, J. D., y Portilla Román, J. G. (2018). Diseño e implementación de un sistema informático para el manejo de inventarios de la Distribuidora “Mateo” [Trabajo de titulación, Universidad Politécnica Salesiana]. Repositorio Institucional de la Universidad Politécnica Salesiana. https://dspace.ups.edu.ec/handle/123456789/15470

Barraza Macías, A. (2023). Metodología de la investigación cualitativa: Una perspectiva interpretativa. Benessere. Centro de Intervención para el Bienestar Físico y Mental A. C. https://www.benesseredgo.org/wp-content/uploads/2024/10/MetodologiaInvestigacion.pdf

Botros, S., y Tinley, J. (2021). High performance MySQL (4th ed.). O’Reilly Media. https://www.oreilly.com/library/view/high-performance-mysql/9781492080503/

Capacitor. (s. f.). Capacitor: Cross-platform native runtime for web apps. Recuperado el 2 de agosto de 2026, de https://capacitorjs.com/docs

Coronel, C., y Morris, S. (2023). Database systems: Design, implementation, & management (14th ed.). Cengage. https://www.cengage.com/c/database-systems-design-implementation-management-14e-coronel-morris/9780357673034/

Davis, K., Peabody, B., y Leach, P. (2024). Universally Unique IDentifiers (UUIDs) (RFC 9562). Internet Engineering Task Force. https://doi.org/10.17487/RFC9562

Fielding, R., Nottingham, M., y Reschke, J. (2022). HTTP semantics (RFC 9110). Internet Engineering Task Force. https://doi.org/10.17487/RFC9110

Gallo Cevallos, J. A. (2025). Control de inventarios y su incidencia en la rentabilidad de la empresa Fish & Dive de la ciudad de Manta [Proyecto de investigación, Universidad Laica Eloy Alfaro de Manabí]. Repositorio Institucional de la Universidad Laica Eloy Alfaro de Manabí. https://repositorio.uleam.edu.ec/handle/123456789/8497

Gómez Bastar, S. (2012). Metodología de la investigación. Red Tercer Milenio.

Hadi, M., Martel, C., Huayta, F., Rojas, R., y Arias, J. (2023). Metodología de la investigación: Guía para el proyecto de tesis. Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú. https://doi.org/10.35622/inudi.b.073

Laravel. (s. f.-a). Events: Laravel 12.x documentation. Recuperado el 2 de agosto de 2026, de https://laravel.com/docs/12.x/events

Laravel. (s. f.-b). HTTP client: Laravel 12.x documentation. Recuperado el 2 de agosto de 2026, de https://laravel.com/docs/12.x/http-client

Laravel. (s. f.-c). Queues: Laravel 12.x documentation. Recuperado el 2 de agosto de 2026, de https://laravel.com/docs/12.x/queues

Laravel. (s. f.-d). Rate limiting: Laravel 12.x documentation. Recuperado el 2 de agosto de 2026, de https://laravel.com/docs/12.x/rate-limiting

Laravel. (s. f.-e). Task scheduling: Laravel 12.x documentation. Recuperado el 2 de agosto de 2026, de https://laravel.com/docs/12.x/scheduling

Laudon, K. C., y Laudon, J. P. (2022). Management information systems: Managing the digital firm (17th ed.). Pearson.

Madden, N. (2020). API security in action. Manning Publications. https://www.manning.com/books/api-security-in-action

Marcillo Vargas, R. J. (2025). Gestión de inventarios y su incidencia en la rentabilidad de la cadena de suministros Manaquim de la ciudad de Manta [Proyecto de investigación, Universidad Laica Eloy Alfaro de Manabí]. Repositorio Institucional de la Universidad Laica Eloy Alfaro de Manabí. https://repositorio.uleam.edu.ec/handle/123456789/8531

Medina, M., Rojas, R., Bustamante, W., Loaiza, R., Martel, C., y Castillo, R. (2023). Metodología de la investigación: Técnicas e instrumentos de investigación. Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú. https://doi.org/10.35622/inudi.b.080

National Institute of Standards and Technology. (2022). Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities (NIST SP 800-218). https://doi.org/10.6028/NIST.SP.800-218

Oracle. (s. f.). The InnoDB storage engine. MySQL 8.4 Reference Manual. Recuperado el 2 de agosto de 2026, de https://dev.mysql.com/doc/refman/8.4/en/innodb-storage-engine.html

Stauffer, M. (2023). Laravel: Up y running (3rd ed.). O'Reilly Media. https://www.oreilly.com/library/view/laravel-up/9781098153250/

Tarrillo Saldaña, O., Mejía Huamán, J., Dávila Mego, J. S., Pintado Castillo, C. A., Tapia Idrogo, C. E., Chilón Camacho, W. M., y Vélez Escobar, S. B. (2024). Metodología de la investigación: Una mirada global: Ejemplos prácticos. CID–Centro de Investigación y Desarrollo. https://doi.org/10.37811/cli_w1078

van Steen, M., y Tanenbaum, A. S. (2023). Distributed systems (4th ed.). Distributed-Systems.net. https://www.distributed-systems.net/index.php/books/ds4/

Botros, S., y Tinley, J. (2021). *High performance MySQL* (4th ed.). O'Reilly Media. https://www.oreilly.com/library/view/high-performance-mysql/9781492080503/

Breyter, M. (2022). *Agile product and project management: A step-by-step guide to building the right products right*. Apress. https://doi.org/10.1007/978-1-4842-8200-7

Burns, B. (2024). *Designing distributed systems: Patterns and paradigms for scalable, reliable services* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/designing-distributed-systems/9781098156343/

Coronel, C., y Morris, S. (2023). *Database systems: Design, implementation, & management* (14th ed.). Cengage. https://www.cengage.com/c/database-systems-design-implementation-management-14e-coronel-morris/9780357673034/

Ford, N., Richards, M., Sadalage, P., y Dehghani, Z. (2021). *Software architecture: The hard parts: Modern trade-off analyses for distributed architectures*. O'Reilly Media. https://www.oreilly.com/library/view/software-architecture-the/9781492086888/

Hastings, N. A. J. (2021). *Physical asset management: With an introduction to the ISO 55000 series of standards*. Springer. https://doi.org/10.1007/978-3-030-62836-9

Heizer, J., Render, B., y Munson, C. (2023). *Operations management: Sustainability and supply chain management* (14th ed.). Pearson. https://www.pearson.com/en-gb/subject-catalog/p/operations-management-sustainability-and-supply-chain-management-global-edition/P200000009988

Jacobs, F. R., y Chase, R. B. (2021). *Operations and supply chain management* (16th ed.). McGraw Hill. https://www.mheducation.com/unitas/highered/changes/jacobs-operations-and-supply-chain-management-16e.pdf

Joshi, U. (2023). *Patterns of distributed systems*. Addison-Wesley Professional. https://www.pearson.com/en-us/subject-catalog/p/patterns-of-distributed-systems/P200000011305

Khononov, V. (2021). *Learning domain-driven design: Aligning software architecture and business strategy*. O'Reilly Media. https://www.oreilly.com/library/view/learning-domain-driven-design/9781098100124/

Kohnfelder, L. (2021). *Designing secure software: A guide for developers*. No Starch Press. https://nostarch.com/designing-secure-software

Laudon, K. C., y Laudon, J. P. (2022). *Management information systems: Managing the digital firm* (17th ed.). Pearson. https://www.pearson.com/en-us/subject-catalog/p/management-information-systems-managing-the-digital-firm/P200000001392

Majors, C., Fong-Jones, L., y Miranda, G. (2022). *Observability engineering: Achieving production excellence*. O'Reilly Media. https://www.oreilly.com/library/view/observability-engineering/9781492076438/

Newman, S. (2021). *Building microservices: Designing fine-grained systems* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/

Richards, G. (2025). *Warehouse management: The definitive guide to improving efficiency and minimizing costs in the modern warehouse* (5th ed.). Kogan Page. https://www.koganpage.com/logistics-supplychain-operations/warehouse-management-9781398618701

Richards, G., y Grinsted, S. (2024). *The logistics and supply chain toolkit: Over 100 tools for transport, warehousing and inventory management* (4th ed.). Kogan Page.

Botros, S., y Tinley, J. (2021). *High performance MySQL* (4th ed.). O'Reilly Media. https://www.oreilly.com/library/view/high-performance-mysql/9781492080503/

Breyter, M. (2022). *Agile product and project management: A step-by-step guide to building the right products right*. Apress. https://doi.org/10.1007/978-1-4842-8200-7

Burns, B. (2024). *Designing distributed systems: Patterns and paradigms for scalable, reliable services* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/designing-distributed-systems/9781098156343/

Coronel, C., y Morris, S. (2023). *Database systems: Design, implementation, & management* (14th ed.). Cengage. https://www.cengage.com/c/database-systems-design-implementation-management-14e-coronel-morris/9780357673034/

Ford, N., Richards, M., Sadalage, P., y Dehghani, Z. (2021). *Software architecture: The hard parts: Modern trade-off analyses for distributed architectures*. O'Reilly Media. https://www.oreilly.com/library/view/software-architecture-the/9781492086888/

Hastings, N. A. J. (2021). *Physical asset management: With an introduction to the ISO 55000 series of standards*. Springer. https://doi.org/10.1007/978-3-030-62836-9

Heizer, J., Render, B., y Munson, C. (2023). *Operations management: Sustainability and supply chain management* (14th ed.). Pearson. https://www.pearson.com/en-gb/subject-catalog/p/operations-management-sustainability-and-supply-chain-management-global-edition/P200000009988

Jacobs, F. R., y Chase, R. B. (2021). *Operations and supply chain management* (16th ed.). McGraw Hill. https://www.mheducation.com/unitas/highered/changes/jacobs-operations-and-supply-chain-management-16e.pdf

Joshi, U. (2023). *Patterns of distributed systems*. Addison-Wesley Professional. https://www.pearson.com/en-us/subject-catalog/p/patterns-of-distributed-systems/P200000011305

Khononov, V. (2021). *Learning domain-driven design: Aligning software architecture and business strategy*. O'Reilly Media. https://www.oreilly.com/library/view/learning-domain-driven-design/9781098100124/

Kohnfelder, L. (2021). *Designing secure software: A guide for developers*. No Starch Press. https://nostarch.com/designing-secure-software

Laudon, K. C., y Laudon, J. P. (2022). *Management information systems: Managing the digital firm* (17th ed.). Pearson. https://www.pearson.com/en-us/subject-catalog/p/management-information-systems-managing-the-digital-firm/P200000001392

Majors, C., Fong-Jones, L., y Miranda, G. (2022). *Observability engineering: Achieving production excellence*. O'Reilly Media. https://www.oreilly.com/library/view/observability-engineering/9781492076438/

Newman, S. (2021). *Building microservices: Designing fine-grained systems* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/

Richards, G. (2025). *Warehouse management: The definitive guide to improving efficiency and minimizing costs in the modern warehouse* (5th ed.). Kogan Page. https://www.koganpage.com/logistics-supplychain-operations/warehouse-management-9781398618701

Richards, G., y Grinsted, S. (2024). *The logistics and supply chain toolkit: Over 100 tools for transport, warehousing and inventory management* (4th ed.). Kogan Page. https://www.koganpage.com/logistics-supplychain-operations/the-logistics-and-supply-chain-toolkit-9781398613379

Shapira, G., Palino, T., Sivaram, R., y Petty, K. (2021). *Kafka: The definitive guide: Real-time data and stream processing at scale* (2nd ed.). O'Reilly Media. https://www.oreilly.com/library/view/kafka-the-definitive/9781492043072/

Slack, N., Brandon-Jones, A., y Burgess, N. (2022). *Operations management* (10th ed.). Pearson. https://www.pearson.com/en-au/subject-catalog/p/operations-management/P200000005430

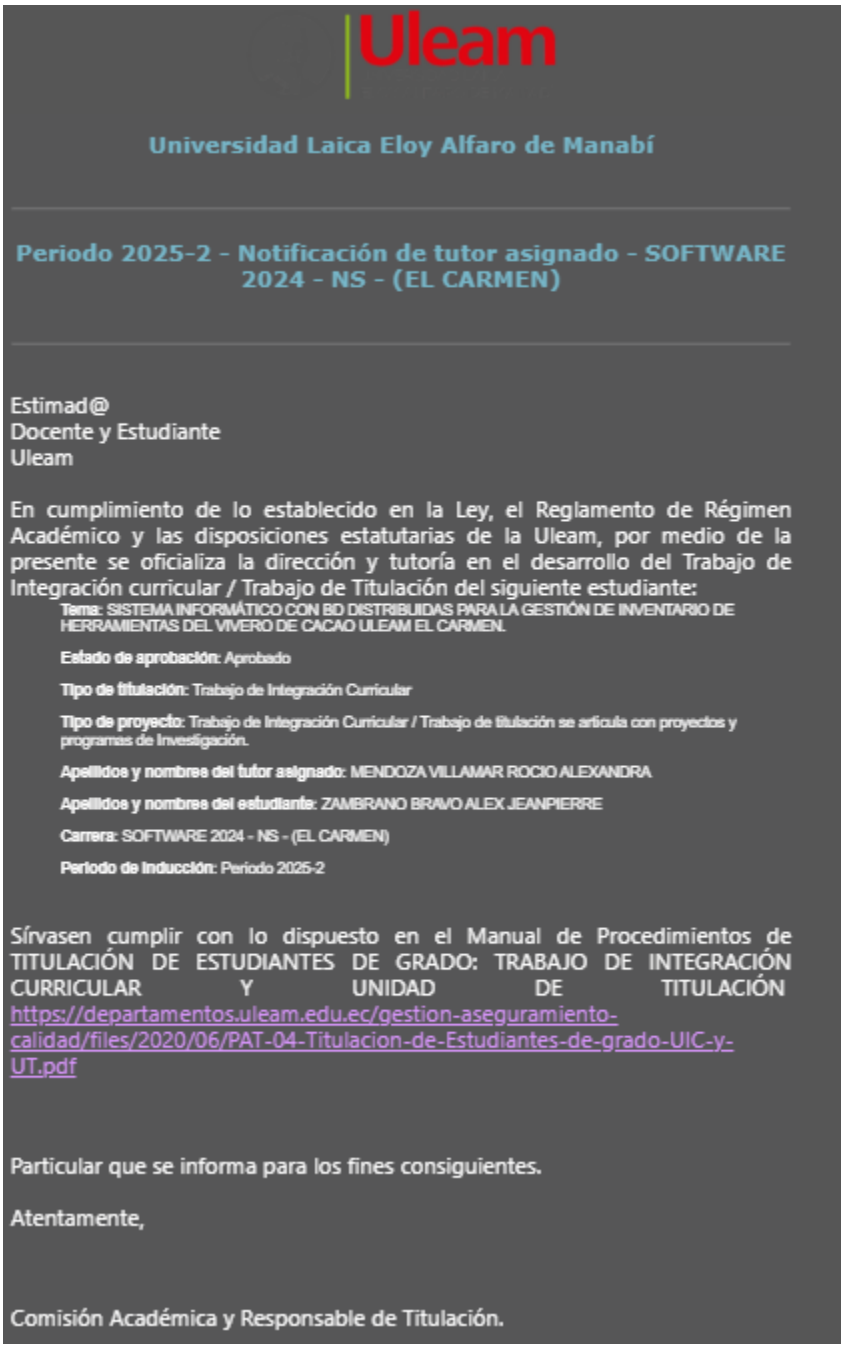
Stauffer, M. (2023). *Laravel: Up & running: A framework for building modern PHP apps* (3rd ed.). O'Reilly Media. https://www.oreilly.com/library/view/laravel-up/9781098153250/

Tsui, F., Karam, O., y Bernal, B. (2022). *Essentials of software engineering* (5th ed.). Jones & Bartlett Learning. https://www.oreilly.com/library/view/essentials-of-software/9781284229004/

van Steen, M., y Tanenbaum, A. S. (2023). *Distributed systems* (4th ed.). distributed-systems.net. https://www.distributed-systems.net/index.php/books/ds4/

ANEXOS

Anexo A: Asignación de tutor



Universidad Laica Eloy Alfaro de Manabí

Periodo 2025-2 - Notificación de tutor asignado - SOFTWARE 2024 - NS - (EL CARMEN)

Estimad@
Docente y Estudiante
Uleam

En cumplimiento de lo establecido en la Ley, el Reglamento de Régimen Académico y las disposiciones estatutarias de la Uleam, por medio de la presente se oficializa la dirección y tutoría en el desarrollo del Trabajo de Integración curricular / Trabajo de Titulación del siguiente estudiante:

Tema: SISTEMA INFORMÁTICO CON BD DISTRIBUIDAS PARA LA GESTIÓN DE INVENTARIO DE HERRAMIENTAS DEL VIVERO DE CACAO ULEAM EL CARMEN.

Estado de aprobación: Aprobado

Tipo de titulación: Trabajo de Integración Curricular

Tipo de proyecto: Trabajo de Integración Curricular / Trabajo de titulación se articula con proyectos y programas de Investigación.

Apellidos y nombres del tutor asignado: MENDOZA VILLAMAR ROCIO ALEXANDRA

Apellidos y nombres del estudiante: ZAMBRANO BRAVO ALEX JEANPIERRE

Carrera: SOFTWARE 2024 - NS - (EL CARMEN)

Periodo de inducción: Periodo 2025-2

Sírvasen cumplir con lo dispuesto en el Manual de Procedimientos de TITULACIÓN DE ESTUDIANTES DE GRADO: TRABAJO DE INTEGRACIÓN CURRICULAR Y UNIDAD DE TITULACIÓN https://departamentos.uleam.edu.ec/gestion-aseguramiento-calidad/files/2020/06/PAT-04-Titulacion-de-Estudiantes-de-grado-UIC-y-UT.pdf

Particular que se informa para los fines consiguientes.

Atentamente,

Comisión Académica y Responsable de Titulación.

Anexo B: Reporte del sistema anti plagio.

Informe de análisis
Compilatio Magister+ | ULEAM-ECU

Tesis_ALEX ZAMBRANO
ID : 63ed1f5b48c560647e7445850338d3bb26e3526f

8%
Textos sospechosos

Nombre del fichero : Tesis_ALEX ZAMBRANO.txt
Tamaño del archivo original : 4,24 MB
Número de palabras : 25.656

Depositante : ROCIO MENDOZA VILLAMAR
Fecha de depósito : 5 de agosto de 2026
Tipo de carga : interface
fecha de fin de análisis : 5 de agosto de 2026

Resumen (sección 1/3)

Localización de los textos sospechosos en el documento :

Incluido en el porcentaje de textos sospechosos :

Similitudes 2%
Pasajes con similitudes a fuentes encontradas en diferentes colecciones.

Detección de IA 6%
Textos estilísticamente próximos a un texto generado por una IA.
Este índice es un indicador y no una prueba. Comprueba con el autor si domina los conocimientos mencionados en el documento.

Anexo C: Fotografías

Anexo D: Evidencia de aplicación de encuestas y entrevista

Anexo E: Formato de la encuesta

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ EXTENSIÓN EL CARMEN

CARRERA DE INGENIERÍA DE SOFTWARE

Encuesta dirigida a los estudiantes de vinculación de la carrera de Agropecuaria usuarios del vivero de cacao

Objetivo: Recopilar información sobre el control actual del inventario de herramientas del vivero de cacao, con la finalidad de identificar deficiencias en los procesos de préstamo, devolución, organización y disponibilidad de los recursos, a fin de sustentar la propuesta de un sistema informático para mejorar su gestión.

1. ¿Cómo califica el control actual del inventario de herramientas? Pregunta sin título

○ Bueno ○ Regular ○ Deficiente

2. ¿Cree usted que el proceso de préstamo de herramientas es ordenado?

○ Si ○ A veces ○ No

3. ¿La devolución incluye revisión del estado de la herramienta?

○ Si ○ A veces ○ No

4 ¿Ha experimentado errores, confusiones, pérdidas de registros debido al manejo manual?

○ Si ○ A veces ○ No

5. ¿Con que frecuencia ha observado herramientas perdidas y dañadas?

○ Frecuentemente ○ A veces ○ No

6. Tiempo promedio para encontrar una herramienta y se la entreguen

○ Menos de 5 min ○ 5–10 min ○ 10–20 min

7. Cómo percibe el nivel de organización del área de almacenamiento

○ Regular ○ Mala ○ Buena

8. ¿La información del inventario es confiable?

○ Confiable ○ Poco confiable ○ No confiable

9. El proceso manual de atención genera demoras?

○ Si ○ No ○ A veces

10. ¿Ha tenido problemas por falta de herramientas disponibles?

○ Si ○ No ○ A veces

11. ¿El sistema manual actual le parece eficiente?

○ Si ○ No ○ A veces

12. ¿Usaría un sistema digital para préstamos y devoluciones?

○ Si ○ No ○ Tal vez

13. ¿Considera que implementar un sistema informático de inventarios mejoraría la organización del inventario de herramientas?

○ Si ○ No ○ Tal vez

Anexo F: Formato de la entrevista

- **Entrevista:**

UNIVERSIDAD LAICA "ELOY ALFARO" DE MANABÍ EXTENSIÓN EL CARMEN
CARRERA DE INGENIERÍA DE SOFTWARE

Entrevista dirigida al responsable del proyecto de vinculación del vivero de cacao de la carrera de Agropecuaria

Objetivo: Obtener información detallada sobre los procedimientos actuales de control y administración del inventario de herramientas e insumos del vivero de cacao, identificando las principales dificultades, necesidades y oportunidades de mejora en los procesos de préstamo, devolución, registro y seguimiento de los recursos, con el propósito de sustentar el desarrollo de un sistema informático para su gestión.

1. ¿Cómo se realiza actualmente el registro y control del inventario de herramientas?

2. Describa el procedimiento de préstamo y devolución de herramientas a estudiantes.

3. ¿Qué mecanismos utilizan para verificar el estado físico de las herramientas al momento de la devolución?

4. ¿Cuáles son las principales dificultades o limitaciones del manejo manual del inventario?

5. ¿Con qué frecuencia se presentan pérdidas, daños o inconsistencias en el inventario?

6. ¿Cuánto tiempo, en promedio, toma localizar y entregar una herramienta solicitada?

7. ¿Cómo evalúa la organización y distribución del área de almacenamiento?

8. ¿Considera que la información registrada del inventario es precisa y se mantiene actualizada? Explique.

9. ¿Se generan demoras durante la atención a los usuarios? ¿Cuáles son las causas principales?

10. ¿Con qué frecuencia no hay herramientas suficientes para todos los estudiantes?

11. Desde su experiencia, ¿considera eficiente el sistema actual? ¿Qué mejoras propone para optimizar el control y la gestión de herramientas?

12. ¿Estaría dispuesto a utilizar un sistema digital para registrar préstamos y devoluciones?

13. ¿Cree que un sistema informático mejoraría la organización y gestión de las herramientas? ¿Por qué?

GLOSARIO

ACID: Conjunto de propiedades de atomicidad, consistencia, aislamiento y durabilidad que permiten ejecutar transacciones de base de datos de manera confiable.

Adaptador de sincronización: Componente que transforma una entidad funcional en una representación intercambiable y aplica los cambios recibidos mediante las reglas del módulo responsable.

API REST: Interfaz de comunicación que permite intercambiar información entre aplicaciones mediante solicitudes HTTP dirigidas a recursos identificados por rutas.

Backoff: Estrategia que aumenta progresivamente el tiempo de espera entre reintentos después de un fallo temporal de comunicación o procesamiento.

Base de datos distribuida: Conjunto de bases de datos ubicadas en nodos diferentes que cooperan para almacenar, consultar y actualizar información como parte de un mismo sistema.

Base de datos local: Base de datos almacenada en el dispositivo o servidor donde se ejecuta la aplicación y que permite continuar determinadas operaciones sin depender de una conexión permanente.

Base de datos relacional: Sistema de almacenamiento que organiza la información en tablas vinculadas mediante claves y restricciones de integridad.

base_version: Versión de una entidad que el nodo emisor considera confirmada antes de generar y transmitir un nuevo cambio.

Capacitor: Entorno de ejecución que permite empaquetar una aplicación web como aplicación móvil y utilizar funcionalidades nativas mediante complementos.

Cola durable: Estructura persistente que conserva los eventos pendientes de envío o procesamiento, incluso después de reiniciar la aplicación o el servidor.

Commit: Operación que confirma definitivamente los cambios efectuados dentro de una transacción de base de datos.

Conflicto de sincronización: Situación que se produce cuando dos nodos modifican una misma entidad a partir de versiones incompatibles y el sistema no puede aplicar automáticamente uno de los cambios.

Consistencia eventual: Modelo en el que los nodos pueden mantener temporalmente datos diferentes, pero convergen después de restablecerse la comunicación y procesarse los cambios pendientes.

Cursor de sincronización: Valor que identifica el último cambio descargado y aplicado correctamente, permitiendo solicitar solamente los eventos posteriores.

DTO (Data Transfer Object): Objeto utilizado para transferir datos entre módulos o nodos sin exponer directamente la estructura interna de las tablas.

Endpoint: Dirección específica de una API que recibe una solicitud y ejecuta la operación asociada con esa ruta.

Entidad sincronizable: Registro funcional autorizado para crearse, actualizarse o eliminarse mediante el mecanismo de sincronización.

entity_version: Número que representa la versión resultante de una entidad después de aplicar correctamente un cambio.

event_id: Identificador único asignado a un evento de sincronización para reconocer reintentos y evitar que sus efectos se apliquen más de una vez.

Evento de dominio: Representación de un hecho relevante que ocurrió después de completarse una operación válida dentro de un módulo funcional.

Hash: Huella digital calculada a partir de un contenido y utilizada para comprobar que la información no fue modificada durante su almacenamiento o transmisión.

HTTPS: Protocolo de comunicación que utiliza TLS para proteger la confidencialidad e integridad de la información intercambiada mediante HTTP.

Idempotencia: Propiedad que permite procesar repetidamente una misma solicitud sin producir efectos adicionales después de su primera aplicación correcta.

Inbound: Evento o entrega que ingresa a un nodo procedente de otra instancia autorizada del sistema.

InnoDB: Motor de almacenamiento de MySQL que proporciona transacciones, claves foráneas, bloqueo por fila, recuperación ante fallos y control de concurrencia.

JSON: Formato de texto utilizado para representar y transmitir datos estructurados entre los nodos del sistema.

Laravel: Framework de desarrollo para PHP utilizado para organizar la lógica, los servicios, las rutas, los eventos, las colas y el acceso a los datos del sistema.

Laravel Queue: Mecanismo de Laravel que permite ejecutar trabajos en segundo plano, administrar reintentos y registrar tareas que no pudieron completarse.

Migración de base de datos: Archivo versionado que permite crear o modificar tablas, columnas, índices y restricciones de forma controlada y reproducible.

Monolito modular: Arquitectura en la que el sistema se despliega como una sola aplicación, pero mantiene sus funcionalidades separadas en módulos con responsabilidades definidas.

Módulo Shared: Componente transversal que concentra servicios comunes, como usuarios, roles, permisos, autenticación, auditoría y elementos compartidos por los demás módulos.

Módulo Synchronization: Componente transversal encargado de administrar nodos, eventos, colas, versiones, transporte, reintentos y conflictos de sincronización.

MySQL: Sistema gestor de bases de datos relacionales utilizado para almacenar la información de los nodos Administrador y Central.

Nodo: Instancia autónoma del sistema que posee identidad, configuración, almacenamiento y capacidad para emitir o recibir cambios.

Nodo Administrador: Instancia ubicada cerca de la operación del vivero que ejecuta la aplicación web y mantiene una base de datos MySQL local.

Nodo Central: Instancia accesible mediante HTTPS que recibe, consolida y distribuye los cambios autorizados entre los demás nodos.

Nodo Móvil: Aplicación Android que utiliza una base de datos SQLite privada y puede registrar operaciones durante periodos sin conectividad.

Observabilidad: Capacidad de comprender el estado interno del sistema mediante registros, métricas y trazas relacionadas con eventos, errores, latencia, reintentos y conflictos.

Operación sin conexión: Acción registrada y almacenada localmente cuando el dispositivo no dispone de comunicación con el Nodo Central.

Outbound: Evento o entrega generada localmente que permanece pendiente de transmisión hacia otro nodo.

Outbox: Tabla local que conserva de forma durable los cambios pendientes de sincronización generados después de una operación realizada sin conexión.

Payload: Contenido funcional de un evento de sincronización, generalmente representado mediante una estructura JSON.

payload_hash: Hash calculado sobre el payload normalizado para comprobar su integridad antes de aplicar el cambio recibido.

Pull: Proceso mediante el cual un nodo solicita y descarga los cambios disponibles desde otro nodo.

Push: Proceso mediante el cual un nodo envía sus cambios pendientes al receptor configurado.

Rate limiting: Mecanismo que limita la cantidad de solicitudes aceptadas durante un periodo para reducir abusos y proteger los recursos del sistema.

Reintento: Nueva ejecución de una entrega que no pudo completarse debido a un error temporal de comunicación o procesamiento.

Replicación lógica: Intercambio de cambios de entidades mediante eventos y contratos, sin copiar directamente los archivos físicos de la base de datos.

Repositorio: Componente que concentra las operaciones de consulta y persistencia de las entidades pertenecientes a un módulo.

Rollback: Operación que revierte los cambios de una transacción cuando ocurre un error o no se cumplen las condiciones requeridas.

Scheduler: Programador de tareas que ejecuta periódicamente el comando encargado de localizar y despachar eventos pendientes.

Sincronización bidireccional: Proceso mediante el cual dos nodos alternan los roles de emisor y receptor para intercambiar cambios en ambos sentidos.

SQLite: Motor de base de datos integrado que almacena la información en un archivo local y permite operar sin un servidor independiente.

sync_conflicts: Tabla que conserva las versiones local y entrante de los conflictos detectados durante la sincronización, junto con su estado y resolución.

sync_entity_states: Tabla que registra la versión actual, la versión sincronizada y el estado de eliminación de cada entidad sincronizable.

sync_nodes: Tabla que almacena la identidad, el tipo, la configuración, el estado y las credenciales protegidas de los nodos autorizados.

sync_queue: Tabla durable que registra las entregas entrantes y salientes, sus intentos, estados, bloqueos, errores y tiempos de disponibilidad.

TLS: Protocolo criptográfico utilizado por HTTPS para proteger la confidencialidad e integridad de la comunicación entre nodos.

Token de nodo: Credencial técnica utilizada para identificar y autenticar una instancia durante el intercambio de información con otro nodo.

Tombstone: Marca lógica que representa la eliminación de una entidad y permite propagarla sin borrar inmediatamente la evidencia del cambio.

Transacción: Conjunto de operaciones de base de datos que se confirma completamente o se revierte si ocurre un error.

UUID: Identificador universal de 128 bits que puede generarse en nodos diferentes sin depender de un contador central.

Versionado optimista: Estrategia de control de concurrencia que compara versiones antes de aplicar una modificación y registra un conflicto cuando la versión esperada no coincide con la almacenada.

WAL (Write-Ahead Logging): Modo de registro de SQLite que escribe primero los cambios en un archivo adicional para favorecer la recuperación y la concurrencia entre lecturas y escrituras.

Worker: Proceso que permanece en ejecución para atender los trabajos pendientes de la cola de sincronización.