UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

TÍTULO PASANTÍA

Módulo web para la gestión de sistemas de información, programación de backups y reportes de mantenimiento del Hospital Universitario San Rafael de Tunja

PROPONENTE(S)

Santiago Sánchez Márquez

DIRECTOR

John Alexis Piracoca Arcos

Tunja

15 de mayo de 2026

CONTENIDO

1. FICHA TÉCNICA DE LA PASANTÍA

Título	Módulo web para la gestión de sistemas de información, programación de backups y reportes de mantenimiento del Hospital Universitario San Rafael de Tunja
Nombre Estudiante (s)	Santiago Sánchez Márquez — C.C. 1.005.162.273
Correo electrónico estudiante (s)	santiago.sanchezm@usantoto.edu.co
Director / codirector	John Alexis Piracoca
Línea de investigación	☒Ingeniería del software ☐ Inteligencia Artificial ☐ Transformación Digital
Empresa	Hospital Universitario San Rafael de Tunja (HUSRT)
Ciudad donde se ubica la empresa	Tunja, Boyacá
Modalidad de la pasantía	Híbrida
Tutor empresarial	David Guerrero
Duración	4 meses (15 de enero de 2026 — 15 de mayo de 2026)
Palabras claves	Gestión de sistemas de información, Programación de backups, Control de acceso basado en roles (RBAC), Calendario interactivo, Alertas reactivas, Mesa de servicios, Trazabilidad de equipos, Reportes de mantenimiento

2. ANTECEDENTES DE LA EMPRESA

El Hospital Universitario San Rafael de Tunja (HUSRT) es una Empresa Social del Estado (ESE) de carácter público, adscrita a la Gobernación de Boyacá. Fundado en 1956, se ha consolidado como el principal centro de referencia de alta complejidad del departamento, atendiendo una población de aproximadamente 1,3 millones de habitantes provenientes de los 123 municipios de Boyacá y departamentos vecinos. Cuenta con más de 30 servicios habilitados, entre ellos urgencias de alta complejidad, hospitalización, cirugía, unidad de cuidado intensivo adulto y neonatal, oncología, neurología, cardiología, imagenología, laboratorio clínico y farmacia hospitalaria.

2.1 Reseña histórica y ubicación estratégica

El Hospital Universitario San Rafael de Tunja fue fundado el 18 de octubre de 1956 por la comunidad de las Hermanas de la Caridad Dominicas de la Presentación, bajo el nombre de "Hospital San Rafael". Inicialmente funcionaba como un centro de baja complejidad con 30 camas y servicios básicos de medicina general y cirugía. En 1978 pasó a ser administrado por el Instituto de Seguros Sociales (ISS) y posteriormente, tras la reforma al sistema de salud colombiano de la Ley 100 de 1993, se transformó en Empresa Social del Estado (ESE) en 1996, adquiriendo su nombre actual.

En el año 2005, el hospital inició un proceso de modernización y expansión física que incluyó la construcción de los edificios de consulta externa y el área de urgencias de alta complejidad. En 2015 fue catalogado por el Ministerio de Salud como centro de referencia departamental de nivel III (alta complejidad).

Actualmente, el HUSRT se encuentra ubicado en la carrera 9 # 19-14, en el centro histórico de Tunja, con una extensión construida de aproximadamente 30.000 metros cuadrados distribuidos en tres edificios principales: el edificio administrativo, el edificio de consulta externa y el edificio de hospitalización central.

Geográficamente, el hospital concentra su atención en una población cercana a 1,3 millones de habitantes (según proyecciones del DANE 2025), abarcando la totalidad de los 123 municipios de Boyacá, así como zonas limítrofes de departamentos vecinos como Casanare, Arauca, Santander y Cundinamarca. Esta ubicación estratégica convierte al HUSRT en la principal puerta de entrada para pacientes remitidos desde hospitales de niveles I y II de la región.

2.2 Misión, visión y valores institucionales

De acuerdo con la información publicada en el sitio web institucional, la misión y visión del Hospital Universitario San Rafael de Tunja son las siguientes (Hospital Universitario San Rafael de Tunja, 2024):

Misión

“Somos un hospital público boyacense que brinda atención en salud integral al paciente y su familia, a través de un talento humano calificado, con enfoque inclusivo, digno y seguro, que contribuye al bienestar y satisfacción de la comunidad, con vocación académica y de mejoramiento continuo” (Hospital Universitario San Rafael de Tunja, 2024).

Visión

"A 2032 ser reconocido como un hospital universitario acreditado en salud, fortalecido en la prestación de servicios de alta complejidad, docencia e investigación. Comprometido con el bienestar del talento humano y una gestión tecnológica eficiente, brindando una atención más segura, amable, empática, respetuosa y con mejores resultados en salud" (Hospital Universitario San Rafael de Tunja, 2024).

Para cumplir con estos propósitos, la institución se apoya en valores como el respeto, la integridad, el compromiso, la calidad y el trabajo en equipo, los cuales orientan el comportamiento organizacional y son transversales a todas las áreas, incluyendo el Departamento de Tecnologías de la Información. Es en el marco de esta visión de "gestión tecnológica eficiente" que se enmarca y justifica el presente proyecto de desarrollo e implementación de un módulo web para la gestión de sistemas de información, programación de backups y reportes de mantenimiento.

Valores corporativos

Los valores que orientan el comportamiento organizacional son:

- Respeto: Reconocimiento de la dignidad del paciente, la familia y el equipo de trabajo.
- Integridad: Actuación ética, transparente y honesta en todas las gestiones.
- Compromiso: Entrega y dedicación en el cumplimiento de la misión institucional.
- Calidad: Búsqueda continua de la excelencia en los procesos y servicios.

- Trabajo en equipo: Colaboración interdisciplinaria para lograr los objetivos comunes.

Estos valores se reflejan también en el enfoque del Departamento de Tecnologías de la Información, que promueve la seguridad de la información como un pilar ético y operativo.

2.3 Diagnóstico situacional: backups, control de acceso y mantenimiento de equipos

2.3.1 Situación actual de los backups

Según la información suministrada el HUSRT no contaba con una herramienta centralizada que permitiera programar, ejecutar y monitorear los backups de los diferentes sistemas de información institucionales. Las prácticas de respaldo se caracterizaban por:

- Ejecución manual o semiautomática: Los responsables de cada sistema realizaban copias de seguridad mediante scripts aislados o procedimientos manuales, sin una planeación unificada.
- Falta de calendario visible: No existía un calendario accesible para todo el equipo del DTI que mostrara las fechas programadas, frecuencias y estados de los backups.
- Ausencia de alertas reactivas: Cuando un backup fallaba o no se ejecutaba, no se generaban notificaciones automáticas, lo que podía pasar desapercibido por días o semanas.
- Sin registro histórico ni métricas: No se llevaba un registro sistemático de la duración, tamaño, éxito/fallo de los respaldos.
- Riesgo de incumplimiento normativo: La Resolución 1995 de 1999 (Ministerio de Salud de Colombia, 1999) y la Ley 1581 de 2012 (Congreso de la República de

Colombia, 2012) exigen garantizar la integridad, disponibilidad y confidencialidad de la información. La ausencia de una estrategia formal de backups exponía al hospital a posibles sanciones.

2.3.2 Situación actual del control de acceso

De acuerdo con la observación directa del autor durante la pasantía (enero-mayo de 2026), el control de acceso a los sistemas de información se gestionaba de manera desarticulada:

- Perfiles independientes por sistema: Cada aplicación manejaba su propia tabla de usuarios y roles, dificultando la administración centralizada.
- Falta de control granular por sistemas asignados: No existía un mecanismo que permitiera asignar a un usuario acceso únicamente a determinados sistemas (permisos binarios o manuales).
- Inexistencia del rol SYSTEMUSER: El personal técnico del DTI y pasantes necesitaban un rol con permisos elevados para administrar backups y sistemas, sin ser administradores totales. Ese perfil no estaba definido.
- Auditoría limitada: No se registraba de forma centralizada quién accedía a qué recurso y en qué momento.

2.3.3 Situación actual del mantenimiento de equipos de sistemas e integración con mesa de servicios

La mesa de servicios interna, basada en GLPI, gestionaba aproximadamente 200 solicitudes mensuales. Sin embargo, según lo observado por el autor, se detectaron las siguientes deficiencias:

- Inventario de equipos desactualizado y disperso: Los equipos se inventariaban con hojas de cálculo y etiquetas físicas sin actualización sistemática.
- Falta de asociación entre casos de servicio y equipos: Al cerrar un caso, el técnico no podía vincular el incidente a un equipo específico en el inventario, lo que impedía generar reportes de mantenimiento por equipo.
- Ausencia de trazabilidad técnica: No existía un formulario estructurado para registrar el mantenimiento realizado (limpieza, cambio de piezas, reinstalación de software).
- Dificultades para la planeación de compras: Sin reportes confiables sobre la vida útil y fallos de equipos, el DTI carecía de justificación técnica para adquirir nuevos equipos.

2.3.4 Impacto de las brechas identificadas

Según el diagnóstico interno del DTI las limitaciones generaban consecuencias como:

Área	**Impacto**	**Frecuencia aproximada**
Backups	Pérdida de información clínica o administrativa por fallo no detectado	1-2 incidentes críticos reportados en el último año

Control de acceso	Acceso no autorizado a información sensible (riesgo de sanciones por Ley 1581)	Sin métricas por falta de auditoría
Mantenimiento de equipos	Tiempos de diagnóstico prolongados, compras sin justificación técnica	Tiempo promedio de resolución > 48 horas en casos críticos

2.3.5 Oportunidad de mejora

A partir de este diagnóstico, el DTI identificó la necesidad de desarrollar un módulo web integrado al sistema HUSRT-Biomédica que abordara estas tres brechas de forma articulada. El presente proyecto de pasantía se enmarca en esa oportunidad, proponiendo:

1. Un calendario visual de backups con alertas reactivas y programación automática por frecuencias configurables.
2. Un control de acceso granular por rol y por sistemas asignados, incluyendo el nuevo rol SYSTEMUSER.
3. La integración de la mesa de servicios con el inventario de equipos de sistemas.

El desarrollo y los resultados obtenidos se describen en la sección 6 (Descripción de actividades realizadas).

3. PLANTEAMIENTO DEL PROBLEMA

3.1 Descripción de la situación problemática

En el entorno hospitalario, la información clínica y administrativa es un activo crítico cuya pérdida o indisponibilidad puede comprometer la seguridad del paciente y la continuidad operativa de la institución. El Hospital Universitario San Rafael de Tunja, como centro de alta complejidad, administra un conjunto creciente de sistemas de información institucionales (asistenciales, administrativos, de soporte y de gestión biomédica), cada uno con políticas de respaldo, responsables y proveedores diferentes. Sin embargo, el Departamento de Tecnologías de la Información no cuenta con un módulo digital que centralice esta información ni que permita programar, ejecutar y auditar los backups de cada sistema bajo una misma plataforma.

3.2 Manifestaciones del problema

A partir del diagnóstico situacional presentado en la sección 2.3 y de la observación directa del autor durante la pasantía, se han identificado las siguientes manifestaciones concretas del problema:

Manifestación	**Descripción**	**Impacto directo**
Falta de inventario centralizado de sistemas de información	No existe un registro único que identifique cada sistema, su responsable, su proveedor, su tecnología base y su periodicidad de uso.	Dificultad para asignar responsabilidades y planificar actualizaciones o migraciones.

Programación de backups dispersa	Las copias de respaldo se gestionan de forma manual, en hojas de cálculo o por memoria del personal técnico, sin un calendario unificado que muestre el estado de cada respaldo	Backups olvidados, ausencia de trazabilidad y pérdida de información crítica para la operación hospitalaria.
Ausencia de alertas y trazabilidad	No se generan notificaciones automáticas cuando un backup está pendiente, vencido o no realizado.	Incremento del riesgo de pérdida de información clínica y administrativa.
Control de acceso insuficiente	La información de respaldos no diferencia entre roles (administrador, usuario asignado, observador), por lo que cualquier funcionario con acceso al sistema podía consultar o modificar registros que no le correspondían.	Riesgo de modificaciones indebidas, fuga de información y posible incumplimiento de la Ley 1581 de 2012 (protección de datos personales).
Visibilidad limitada para la	La falta de un calendario	Decisiones basadas en

toma de decisiones	visual interactivo y de exportación a Excel impide al jefe del DTI monitorear el cumplimiento de las políticas de respaldo y reportar indicadores a la gerencia.	información incompleta o desactualizada, y dificultad para demostrar el cumplimiento ante auditorías.

3.3 Formulación del problema

A partir de la situación problemática descrita y sus manifestaciones, se formula la siguiente pregunta central de investigación, junto con las preguntas específicas que guían el desarrollo del proyecto:

Pregunta central

¿De qué manera el desarrollo e implementación de un módulo web integrado al sistema HUSRT-Biomédica permite la parametrización de sistemas de información, la programación y seguimiento de backups mediante un calendario visual con alertas reactivas, el control de acceso granular por rol y sistemas asignados, ¿y la integración de la Mesa de Servicios con el inventario de equipos de sistemas en el Hospital Universitario San Rafael de Tunja?

Preguntas específicas

1. ¿Cómo diseñar e implementar un módulo de parametrización de sistemas de información que incluya modelo de datos en MariaDB, API REST en Express con Sequelize, y vista CRUD en Angular con PrimeNG, relacionando responsable y proveedor, ¿y control de estado?
2. ¿De qué forma se puede construir un modelo de datos, migración y API CRUD para la programación de backups asociados a cada sistema, con frecuencias configurables (Diario, Semanal, Quincenal, Mensual, Trimestral, Semestral, Anual), ¿generación automática de ocurrencias anuales y auto-transición a estado "No realizado" para backups vencidos?
3. ¿Cómo desarrollar un calendario visual interactivo de backups con codificación por color según estado (Programado, Completado, No realizado, Fallido), navegación mensual, tooltips, modal de gestión, exportación a Excel y diseño responsive?
4. ¿Qué mecanismo permite implementar un sistema de notificaciones y alertas reactivas mediante campana con badge, panel overlay de alertas pendientes, auto-regeneración del siguiente backup al completar el actual y sincronización en tiempo real entre calendario y panel de alertas?
5. ¿Cómo establecer un control de acceso granular al módulo combinando rol del usuario y sistemas asignados, incluyendo el nuevo rol SYSTEMUSER, endpoint centralizado de decisión de acceso, guard de Angular con caché invalidable, y restricciones diferenciadas en lectura y escritura?
6. ¿De qué manera integrar la Mesa de Servicios con el inventario de equipos de sistemas mediante asociación opcional entre caso de servicio y equipo, e implementar un flujo de cierre de caso que redirija a un formulario completo de reporte de mantenimiento, garantizando la trazabilidad del trabajo técnico realizado?

3.4 Árbol de problemas

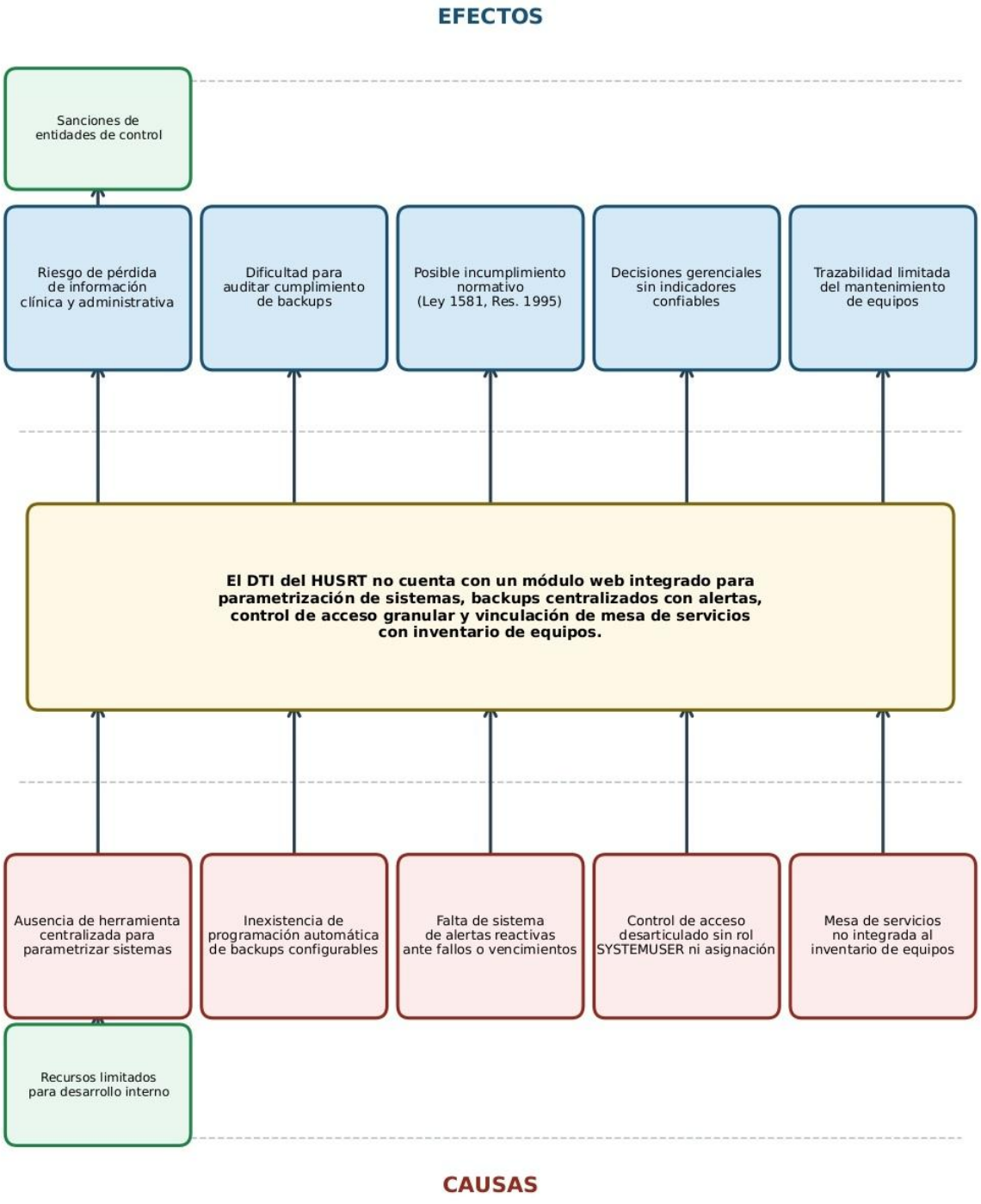


Figura 1. Árbol de problemas del Departamento de Tecnologías de la Información del HUSRT.

Diagrama ER – Sistema de información y backups

Figura 2. Diagrama ER – Sistema de información y backups

4. JUSTIFICACIÓN

La información clínica y administrativa del Hospital Universitario San Rafael de Tunja constituye un activo crítico para la prestación segura de los servicios de salud. Su pérdida, corrupción o indisponibilidad afecta directamente la atención al paciente, la facturación y el cumplimiento normativo. Por ello, contar con una herramienta que centralice la gestión de los sistemas de información y de sus copias de respaldo no es un lujo tecnológico sino una necesidad operativa y regulatoria.

4.1 Justificación técnica

El módulo desarrollado en esta pasantía se integra al sistema HUSRT-Biomédica ya en producción, reutilizando su infraestructura de autenticación (JWT), su base de datos MariaDB, su backend Express y su frontend Angular. Esto reduce los costos de licenciamiento, evita la introducción de stacks adicionales y facilita la adopción por parte de los usuarios del Departamento de TI, que ya conocen la interfaz del sistema.

Al apalancarse en componentes existentes (PrimeNG, AuthGuard, servicios HTTP), el desarrollo se concentra en el valor diferencial: el calendario visual de backups, las alertas reactivas y el control de acceso por sistemas asignados a través del nuevo rol SYSTEMUSER.

La elección de tecnologías específicas responde a criterios de madurez y productividad:

- Sequelize como ORM permite abstraer la base de datos MariaDB, simplificando las migraciones y las consultas relacionales entre sistemas, backups, responsables y proveedores.
- PrimeNG proporciona componentes listos para usar, como la tabla con paginación (`p-table`) y el calendario interactivo, lo que acelera el desarrollo frontend y garantiza una interfaz responsive.
- JWT (JSON Web Token) asegura la autenticación stateless, facilitando la escalabilidad del módulo sin necesidad de gestionar sesiones en servidor.
- Angular standalone components reduce la complejidad de módulos y mejora el rendimiento de carga, al tiempo que facilita el mantenimiento del código.

Esta combinación tecnológica no solo resuelve los problemas identificados, sino que deja una base sólida para futuras ampliaciones (por ejemplo, integración con sistemas externos de backup o generación de alertas por correo electrónico).

4.2 Justificación operativa

La implementación del módulo web desarrollado impacta positivamente la operación diaria del Departamento de Tecnologías de Información del HUSRT. A continuación, se describen los principales beneficios operativos, alineados con las manifestaciones del problema identificadas en la sección 3.2.

Centralización de la información: El módulo proporciona un inventario único de sistemas de información, con campos como responsable, proveedor, estado y frecuencia de backup. Esto elimina el uso de hojas de cálculo o registros dispersos, facilitando la consulta y actualización.

Automatización de la programación de backups: La solución permite configurar frecuencias (diario, semanal, quincenal, mensual, trimestral, semestral, anual) y genera automáticamente todas las ocurrencias del año en curso. Esto reduce la carga manual del equipo del DTI y minimiza el riesgo de backups olvidados.

Visibilidad mediante calendario interactivo: El calendario visual codificado por colores (Programado, Completado, No realizado, Fallido) permite a los técnicos y al jefe del DTI monitorear el estado de cada respaldo de un vistazo. La navegación mensual, los tooltips informativos y la capacidad de exportar a Excel facilitan la rendición de cuentas ante la gerencia y entes de control.

Alertas reactivas: El sistema de campana con badge y panel overlay de alertas pendientes notifica de forma oportuna los backups vencidos o próximos a vencer. La auto-regeneración del siguiente backup al completar el actual y la sincronización en tiempo real entre el calendario y las alertas garantizan que ningún respaldo quede sin seguimiento.

Control de acceso granular: La incorporación del rol SYSTEMUSER y la asignación de sistemas por usuario permiten que cada técnico acceda únicamente a la información relevante para su función. Esto mejora la seguridad de la información y facilita la auditoría de accesos.
Integración de mesa de servicios con inventario: Al asociar cada caso de servicio con un equipo de sistemas y al cerrar el caso con un formulario estructurado de reporte de mantenimiento, se logra una trazabilidad completa del trabajo técnico. El Departamento de TI puede generar reportes históricos por equipo, calcular indicadores como tiempo medio de reparación (MTTR) y justificar compras o renovaciones tecnológicas.

En conjunto, estas mejoras operativas reducen los tiempos de respuesta, aumentan la confiabilidad de los respaldos y fortalecen la capacidad del DTI para garantizar la continuidad de los servicios hospitalarios.

4.3 Justificación académica

Desde la perspectiva académica, la pasantía permite aplicar de forma integrada conocimientos adquiridos durante la formación en ingeniería de software y áreas afines. A continuación, se enumeran las principales competencias y temáticas abordadas:

- Modelado de datos relacional: Diseño de tablas, relaciones (claves foráneas), tipos de datos y migraciones en MariaDB, aplicando conceptos de normalización.
- Diseño de APIs REST: Implementación de endpoints con Express.js, manejo de códigos HTTP, validación de datos y autenticación mediante JWT.
- Arquitectura frontend basada en componentes: Uso de Angular standalone components, servicios con inyección de dependencias, y guards para control de rutas.

- Librerías de componentes (PrimeNG): Integración de tablas (`p-table`), calendarios, modales y tooltips para construir interfaces de usuario modernas y responsivas.
- ORM (Sequelize): Abstracción de la base de datos, definición de modelos, asociaciones `belongsTo` / `hasMany`, y ejecución de consultas complejas.
- Metodologías ágiles: Trabajo con historias de usuario (27 épicas y 68 historias implementadas), desarrollo iterativo, y uso de herramientas de control de versiones (Git).
- Seguridad y control de acceso: Implementación de middleware para decisión centralizada de acceso (rol + sistemas asignados), guards en Angular, y manejo de caché invalidable.
- Dominio hospitalario y normativa: Comprensión de la operación de un hospital de alta complejidad, identificación de sistemas de información clínica (HIS, PACS, LIS), y aplicación de requisitos de la Ley 1581 de 2012 y la Resolución 1995 de 1999.

Adicionalmente, el desarrollo de esta pasantía fortalece competencias transversales como la documentación técnica, la integración continua, la generación de reportes de pruebas y la comunicación con usuarios finales (personal del DTI y del hospital). El resultado no solo tiene valor práctico para la institución, sino que constituye un insumo para futuras investigaciones y trabajos de grado relacionados con gestión de respaldos, control de acceso y trazabilidad de mantenimiento en entornos hospitalarios.

4.4 Justificación legal y normativa

El desarrollo del módulo web se alinea con el marco normativo colombiano aplicable a la información en salud y a la protección de datos personales. A continuación, se describen las principales disposiciones que justifican la necesidad de contar con una herramienta centralizada de backups y control de acceso.

Ley 1581 de 2012 (Protección de datos personales): Esta ley establece el régimen general para la protección de datos personales en Colombia. En el contexto hospitalario, los sistemas de información clínica y administrativa contienen datos sensibles de pacientes y empleados. La ley exige garantizar la seguridad, confidencialidad e integridad de la información, así como la posibilidad de auditar el acceso y las modificaciones. El módulo desarrollado contribuye a este cumplimiento mediante el control de acceso granular (rol + sistemas asignados) y la trazabilidad de los backups (registro histórico de ejecución y estado). *Congreso de la República de Colombia. (2012, 17 de octubre). Ley 1581 de 2012, por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial n.° 48587. http://www.secretariasenado.gov.co/senado/basedoc/ley_1581_2012.html*

Resolución 1995 de 1999 (Manejo de la Historia Clínica): Esta resolución del Ministerio de Salud establece normas para la gestión de la historia clínica, incluyendo su custodia, conservación y disponibilidad. En su articulado, exige que las instituciones de salud implementen mecanismos que garanticen la integridad y disponibilidad de la historia clínica, así como la posibilidad de recuperarla ante eventuales pérdidas o daños técnicos. La programación automática de backups y el calendario de monitoreo implementados en este proyecto responden

directamente a esta exigencia. *Ministerio de Salud de Colombia. (1999, 8 de julio). Resolución 1995 de 1999, por la cual se establecen normas para el manejo de la Historia Clínica. https://www.minsalud.gov.co/Normatividad_Nuevo/RESOLUCI%C3%93N%201995%20DE%201999.pdf*

Circular Externa 000002 de 2016 (Superintendencia Nacional de Salud): Esta circular imparte instrucciones relativas a la gestión de la información en salud, incluyendo la obligación de adoptar planes de continuidad de negocio y respaldo de datos. La Superintendencia puede sancionar a las instituciones que no demuestren la implementación de medidas técnicas para proteger la información. El desarrollo del calendario visual de backups, las alertas reactivas y la generación de reportes exportables a Excel constituyen evidencia documentada de dichas medidas. (Superintendencia Nacional de Salud, 2016)

ISO/IEC 27001 (Seguridad de la información): Aunque no es una norma colombiana obligatoria, el hospital ha manifestado su interés en adoptar buenas prácticas internacionales. El control de acceso basado en roles y sistemas asignados (RBAC) que implementa este módulo se alinea con los controles de acceso (A.9) de la norma ISO 27001, específicamente con la política de control de acceso y la gestión de derechos de acceso. (International Organization for Standardization, 2022)

En consecuencia, el módulo desarrollado no solo resuelve un problema operativo, sino que fortalece el cumplimiento normativo del HUSRT, reduciendo el riesgo de sanciones y contribuyendo a la seguridad jurídica de la institución.

5. OBJETIVOS

5.1. Objetivo General

Construir un módulo web para el sistema HUSRT-Biomedica, mediante Angular, Node.js y TypeScript, con el fin de centralizar la gestión de los sistemas de información, la programación de backups y el control de acceso por roles del área de TI del Hospital Universitario San Rafael de Tunja.

5.2. Objetivos Específicos

1. Especificar los requerimientos de las funcionalidades de gestión de los sistemas de información y la programación de backups a través de calendario, alertas y notificaciones, mediante la documentación de historias de usuario validadas por el tutor empresarial y el diseño de la fracción del diagrama relacional correspondiente, con el fin de establecer el alcance funcional y estructural del desarrollo.
2. Desarrollar el módulo de gestión de los sistemas de información con control de acceso granular por rol, mediante Angular, Node.js y TypeScript, con el fin de centralizar su registro y seguimiento en el área de TI.
3. Implementar la funcionalidad de programación y seguimiento de backups, mediante la inclusión de un calendario interactivo, notificaciones reactivas y transición automática de estados, con el fin de automatizar el control y la trazabilidad de las rutinas de respaldo del área de TI.
4. Validar el módulo desarrollado mediante la ejecución de pruebas unitarias y funcionales y la aprobación del tutor empresarial, con el fin de garantizar la calidad del software y documentar su uso en el manual de usuario.

6. DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

6.1. Listado de actividades realizadas en el proyecto

A continuación, se presenta el listado de actividades ejecutadas durante la pasantía, organizadas por épicas funcionales que agrupan historias de usuario relacionadas. Para cada actividad se especifica la fecha de finalización (producto entregado), los entregables concretos (módulos, endpoints, componentes, integraciones) y la referencia a la sección o anexo de este documento donde se puede evidenciar su implementación. La tabla sigue el orden cronológico de desarrollo, desde la configuración del backend hasta la integración final de la mesa de servicios con el inventario de equipos. Todos los productos entregados corresponden a los planificados al inicio de la pasantía, e incluso se incorporaron funcionalidades adicionales (como la auto-transición a estado "No realizado" y la generación anual automática de backups) que superaron las expectativas iniciales.

Épica / Historia	Fecha de finalización	Productos entregados
EP-01: Backend – Sistemas de Información	31/01/2026	Tabla sistemasinformacion en MariaDB, modelo Sequelize, API REST completa (GET, POST, PUT, DELETE)
EP-02: Frontend – Sistemas de Información	05/02/2026	Componente Angular standalone con tabla PrimeNG (CRUD) y tarjeta de navegación "Sistemas de Información"
EP-03: Integración – Sistemas de Información	20/02/2026	Conexión backend-frontend con manejo de errores, tokens JWT y notificaciones SweetAlert2
EP-04: Relación responsable – Backend	25/02/2026	Claves foráneas responsableId y proveedorId en tabla sistemas, asociaciones Sequelize (belongsTo/hasMany), endpoints con datos completos
EP-05: Relación responsable – Frontend	28/02/2026	Dropdowns para seleccionar responsable (usuarios del sistema) y proveedor; visualización de nombres completos en tabla
EP-06: Modelo Backup	10/03/2026	Modelo BackupSistema con campos: sistemaInformacionId, fecha, tipo, estado, observacion, frecuencia_backup (ENUM)

EP-07: API Backups	20/03/2026	Endpoints CRUD para backups: GET por sistemaId (con filtro por mes), POST, PUT, DELETE. Asociaciones Sequelize
EP-08: Calendario Backups – Frontend	30/03/2026	Servicio BackupSistemaService, modal de gestión (selector de fecha, lista de backups)
EP-09: Alertas y notificaciones de backups	25/03/2026	Endpoint GET /backups/alertas, servicio con BehaviorSubject y polling, campana con badge, panel overlay de notificaciones
EP-10: Campo frecuencia backup	05/04/2026	Campo frecuencia_backup como ENUM al modelo. Dropdown en frontend con opciones: Diario, Semanal, Mensual, Anual (luego se amplió a 7 frecuencias)
EP-11: Edición de backups	06/04/2026	Modo de edición en formulario de backup: botón editar, precarga de campos, texto diferenciado en el botón guardar
EP-12: Notificaciones dinámicas	08/04/2026	Endpoint de alertas actualizado para usar frecuencia_backup. Las alertas del panel se eliminan al marcar backup como Completado
EP-13: Rediseño calendario		Codificación visual con chips/badges

backups	11/04/2026	individuales, barra lateral color, navegación prev/siguiente mes, tooltips, media queries responsive
EP-14: Modal detalle backup en calendario	12/04/2026	Al hacer clic en un chip del calendario se abre modal con todos los campos del backup y badge de estado
EP-15: Auto-regeneración de alertas	14/04/2026	Al marcar backup como Completado, el backend crea automáticamente el siguiente registro Pendiente. Botón "Marcar como Completado" en modal del calendario y en panel de alertas (con confirmación SweetAlert2)
EP-16: Mejoras UX calendario	16/04/2026	Exportación del calendario a Excel con título, timestamp, filas coloreadas por estado y conteos resumen. Se eliminó botón "Marcar como Completado" del panel de alertas (dejándolo solo informativo)
EP-17: Unificación de periodicidad	18/04/2026	Eliminado campo periodicidad del modelo SistemaInformacion; ahora se deriva del backup más reciente. En frontend se removió el campo editable
		Se añadió "No realizado" como cuarto

EP-18: Estado “No realizado"	19/04/2026	valor ENUM en estado del modelo BackupSistema. Soporte en calendario, modal, leyenda y exportación Excel (color gris inicial)
EP-19: Alertas mejoradas – auto-transición	21/04/2026	Backups Pendientes con fecha anterior a hoy se auto-transicionan a "No realizado" al consultar. Filtro de alertas limitado a fecha <= hoy
EP-20: Generación anual de backups	24/04/2026	Al crear un backup (POST), se generan automáticamente todas las ocurrencias Pendientes para el resto del año según la frecuencia
EP-21: Control de acceso por rol en backups	27/04/2026	Endpoints GET filtran por rol (administradores ven todos; otros roles solo sistemas asignados). POST/PUT/DELETE restringidos a SUPERADMIN y SYSTEMADMIN. Frontend oculta botones crud para no administradores
EP-22: Color naranja para "No realizado"	28/04/2026	Cambio del color del estado "No realizado" de gris a naranja en calendario, modal, leyenda y exportación Excel

EP-23: Acceso al módulo – rol SYSTEMUSER	02/05/2026	Endpoint GET /usuarios/me/acceso-modulo-sistemas. Middleware combinando validación de rol + sistemas asignados. Guard de Angular redirige a acceso denegado. Ocultamiento de enlaces en hub y navbars con caché. Integración del rol SYSTEMUSER en login y redirección
EP-24: Asignación de equipo a caso (mesa de servicios)	04/05/2026	FK opcional equipoId al modelo MesaCaso referenciando SysEquipo. Endpoints POST/PUT soportan equipoId. Dropdown opcional de equipo en formulario de casos. Campo de equipo editable en detalle del caso
EP-25: Casos cerrados por equipo	05/05/2026	Endpoint GET/casos/equipo/:id_sysequipo que retorna casos cerrados de un equipo. Modal que muestra los casos con título, descripción, prioridad, fecha y creador
EP-26: SysReporte mantenimiento	07/05/2026	Extensión del modelo SysReporteMantenimiento (hora_inicio, tipo_falla, motivo, calificación, etc.) y endpoint GET para consultar detalle

EP-27: Rediseño visual – Equipos sistemas	09/05/2026	Actualización del dropdown de equipos para navegar a /adminsistemas/nuevoreporte/:id y /adminsistemas/reportesequipo/:id, eliminando componente modal anterior

6.2. Herramientas utilizadas en el desarrollo del proyecto

Herramienta / Tecnología	Versión	¿Para qué se utilizó?
MariaDB	10.11	Base de datos relacional para almacenar los datos del módulo: sistemas de información, backups, responsables, proveedores, equipos, casos de servicio y reportes de mantenimiento. Se usó porque es la base de datos existente en el sistema HUSRT-Biomedica.
Sequelize	6.35	ORM (Object-Relational Mapping) para Node.js que permitió definir modelos, migraciones y asociaciones (belongsTo, hasMany) entre tablas, evitando escribir SQL manual y acelerando el desarrollo del backend.
		Framework web para Node.js utilizado para construir la API

Express.js	4.18	REST del módulo, exponiendo endpoints para sistemas, backups, alertas, control de acceso, mesa de servicios y reportes de mantenimiento.
Node.js	20.11	Entorno de ejecución de JavaScript del lado del servidor. Se eligió porque permite construir aplicaciones escalables, con un amplio ecosistema de librerías (npm) y porque el HUSRT-Biomedica ya lo utilizaba.
Angular	17.0 (standalone)	Framework frontend para construir la interfaz de usuario del módulo. Se usó porque el sistema HUSRT-Biomedica está desarrollado en Angular, garantizando integración, reutilización de servicios de autenticación y guards.
PrimeNG	17.0	Librería de componentes UI para Angular. Se empleó para construir la tabla de sistemas (p-table), el calendario visual de backups (con chips y badges), tooltips, modales y la exportación a Excel.

SweetAlert2	11.10	Biblioteca para modales personalizables. Se utilizó en las operaciones críticas: confirmar al marcar un backup como completado (con campo opcional de observaciones), confirmar eliminaciones y mostrar mensajes de éxito o error.
JWT (JSON Web Token)	9.0	Mecanismo de autenticación stateless. El token se genera en el login y se envía en cada petición al backend. Permite al backend identificar al usuario, su rol y los sistemas asignados, y aplicar el control de acceso granular.
Git	2.43	Sistema de control de versiones para gestionar el código fuente, mantener un historial de cambios y colaborar con otros desarrolladores (por ejemplo, al integrar el módulo en el repositorio del HUSRT-Biomedica).
Windsurf	2.2.17	Editor de código utilizado para escribir y depurar el código del backend y frontend, con extensiones

		para Angular, TypeScript, JavaScript y MySQL.
GitHub (o similar)	-	Plataforma de alojamiento de repositorios Git. Se utilizó para respaldar el código, mantener versiones y facilitar la integración con el sistema existente.

6.3. Metodología de desarrollo

El desarrollo del módulo se llevó a cabo bajo un enfoque de metodologías ágiles (Beck et al., 2001), adaptado al contexto de la pasantía con entregas incrementales y validación continua con el Departamento de Tecnologías de la Información del Hospital Universitario San Rafael de Tunja.

6.3.1. Evidencias Objetivo 1 – Requerimientos de software

Los requerimientos del software se especificaron a través de historias de usuario (User Stories; Cohn, 2004), una técnica propia de marcos ágiles como Scrum (Schwaber & Sutherland, 2020) y XP, que describe la funcionalidad deseada desde la perspectiva del usuario final. Cada historia siguió el formato estándar: "Como [rol], quiero [funcionalidad], para [beneficio]". Este enfoque garantizó que cada requerimiento estuviera orientado al valor real para el usuario y no solo a la solución técnica.

Las historias se organizaron en épicas, es decir, grupos temáticos de funcionalidades relacionadas que permitieron estructurar el trabajo en bloques de valor entregable de manera independiente. En total se definieron y ejecutaron 27 épicas y 69 historias de usuario, cubriendo

desde la parametrización de sistemas de información hasta la integración de la Mesa de Servicios con el inventario de equipos de sistemas.

Cada historia de usuario incluyó criterios de aceptación escritos en formato Gherkin (Given / When / Then), lo que facilitó la validación objetiva de cada funcionalidad antes de considerarla terminada y permitió detectar de forma temprana desviaciones respecto a los requerimientos originales. A modo de ejemplo, a continuación, se presenta una de las historias de usuario implementadas para la épica de backend de sistemas de información:

> ***Como** administrador del sistema, **quiero** que exista una tabla sistemasinformacion en la base de datos MariaDB, **para** poder almacenar y gestionar los registros de los sistemas de información del hospital.*
> *Los **criterios de aceptación** incluyeron que la tabla contuviera todos los campos requeridos, que el campo estado no admitiera valores nulos y que la tabla no contuviera registros de prueba al momento de su creación.*

6.3.2. Proceso iterativo e incremental

El trabajo se organizó en iteraciones cortas alineadas con las épicas (Schwaber & Sutherland, 2020), entregando funcionalidades completas y probadas al final de cada ciclo. Las validaciones con el equipo de TI del hospital retroalimentaron directamente las historias siguientes y, en algunos casos, motivaron el rediseño de funcionalidades ya implementadas. Entre los ajustes más significativos se encuentran el cambio de paradigma de alertas calculadas dinámicamente a registros persistentes en base de datos, y la sustitución del formulario de reportes embebido en un modal por una ruta Angular dedicada, con el fin de garantizar validaciones completas y uniformidad con el módulo biomédico ya existente en producción.

6.4 Desarrollo detallado del módulo

A continuación, se presenta el desarrollo detallado de cada actividad realizada durante la pasantía, organizado según los objetivos específicos planteados en la sección 5.2. Para cada objetivo, se describen las implementaciones técnicas, los componentes de software construidos (backend, frontend, base de datos) y las pruebas de validación correspondientes. Las evidencias de los productos entregables se muestran mediante figuras (capturas de pantalla, diagramas de flujo, modelos de datos) y tablas, todas numeradas y con título descriptivo. En los casos en que la información sea confidencial para la institución (por ejemplo, datos de usuarios reales o configuraciones de servidores), se sustituirá por diagramas explicativos o datos de prueba en la versión que se publique en el repositorio institucional CRAI. El manual de usuario del módulo desarrollado se incluye como Anexo B.

6.4.1 Evidencias objetivo 2 -Desarrollo del módulo de parametrización de sistemas de información

Se diseñó e implementó el módulo de parametrización de sistemas de información, que permite al Departamento de TI gestionar de forma centralizada los sistemas institucionales (asistenciales, administrativos y de soporte), asociándolos a un responsable (usuario del sistema) y a un proveedor externo.

Modelo de datos y migración en MariaDB

Se creó la tabla `sistemasinformacion` en la base de datos MariaDB del HUSRT-Biomédica. Los campos definidos incluyeron: identificador único (`id`), `nombre` (único), `descripcion`, `responsableId` (clave foránea a la tabla `usuarios`), `proveedorId` (clave foránea a `proveedores`), `estado` (activo, inactivo, en mantenimiento) y los timestamps `createdAt`, `updatedAt`. Se generó el script de migración correspondiente, asegurando la integridad referencial.

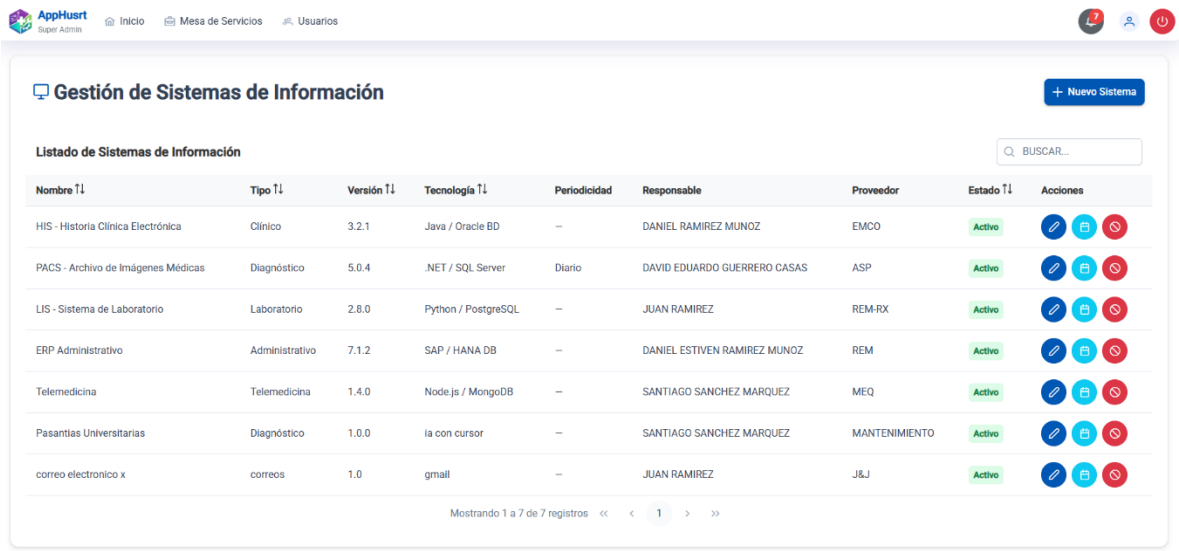
Backend con Sequelize y API REST

En el backend, se definió el modelo `SistemaInformacion` utilizando Sequelize, estableciendo las asociaciones `belongsTo` con los modelos `Usuario` (responsable) y `Proveedor`. Esto permitió que al consultar un sistema se obtuvieran automáticamente los datos completos del responsable y del proveedor, evitando consultas adicionales. Se construyó una API REST con endpoints para listar, obtener, crear, actualizar y eliminar sistemas. Cada endpoint incluyó validaciones de datos y manejo de errores. La implementación completa de la API se presenta en el Anexo A.2.

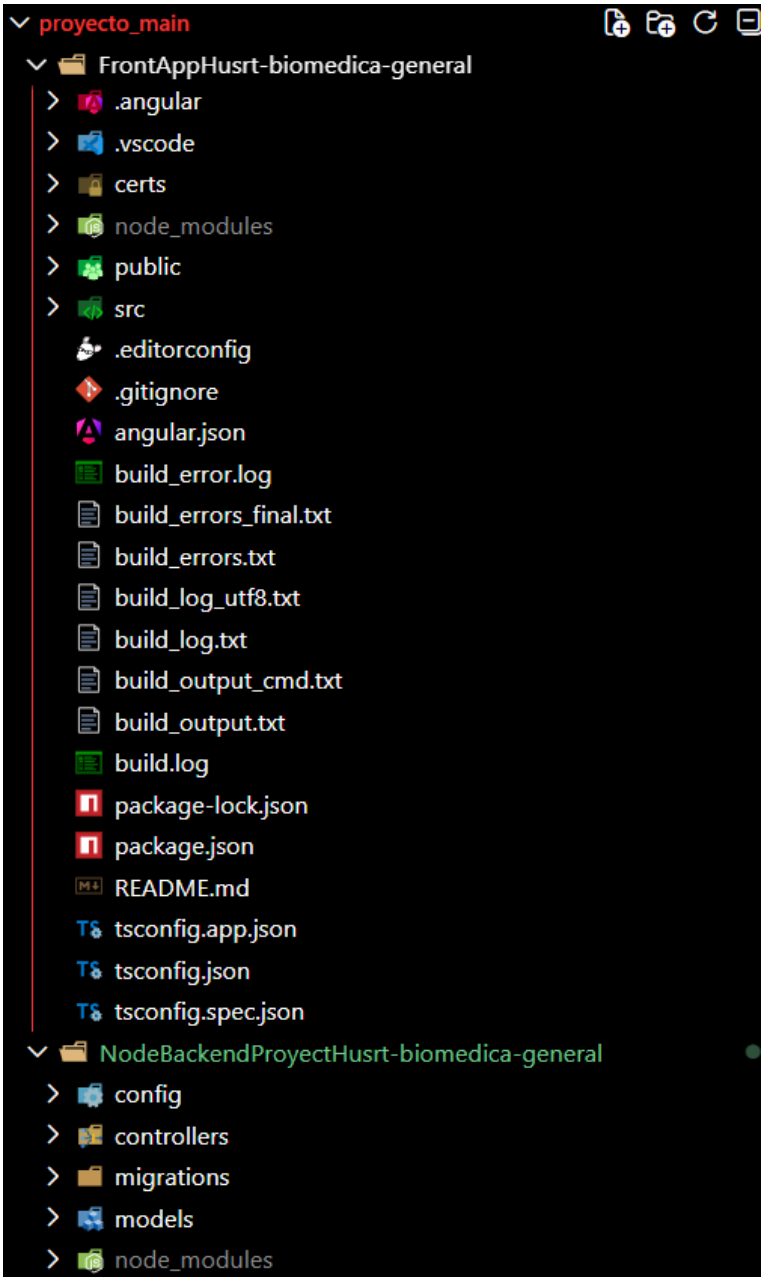
Frontend con Angular y PrimeNG

En el frontend, se creó un componente standalone que integra una tabla de PrimeNG (`p-table`) con las operaciones CRUD. La tabla muestra columnas de nombre, responsable (nombre completo del usuario), proveedor, estado y acciones (editar, eliminar). Se añadieron dos dropdowns (`p-dropdown`) para seleccionar responsable y proveedor, reemplazando los campos de texto libres. El componente maneja la paginación, el ordenamiento y los filtros de forma

automática. Las operaciones se conectan al servicio `SistemaService`, que consume la API REST con el token JWT adjunto. Se utilizó SweetAlert2 para confirmar acciones destructivas.



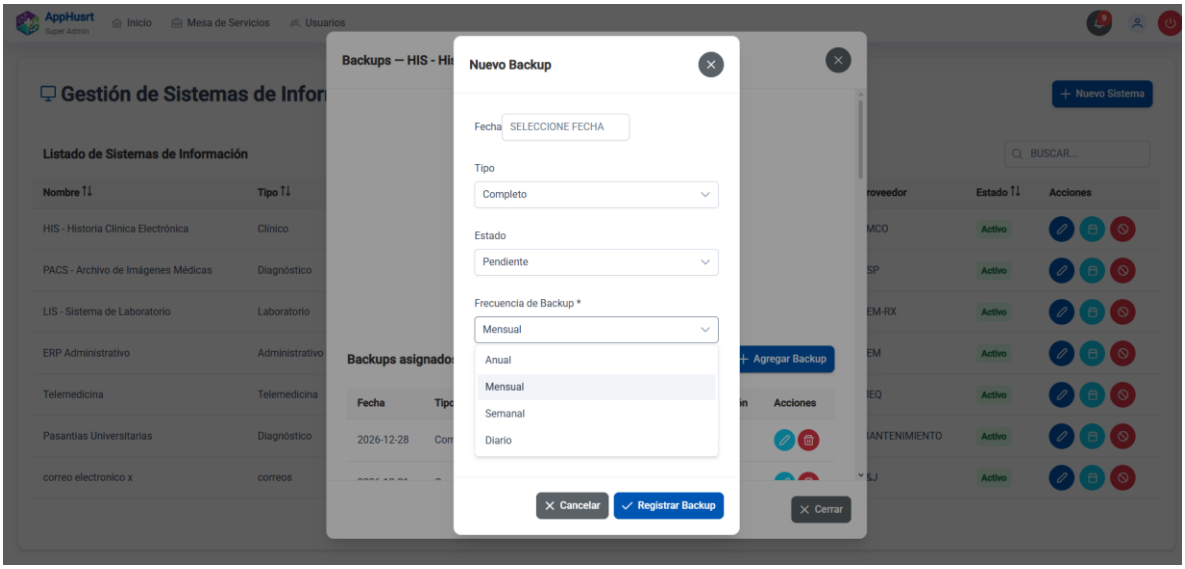
Evidencia 1. Vista del listado de sistemas de información en producción



Evidencia 2. Estructura del proyecto divida en backend y frontend respectivamente

Unificación de periodicidad (EP-17)

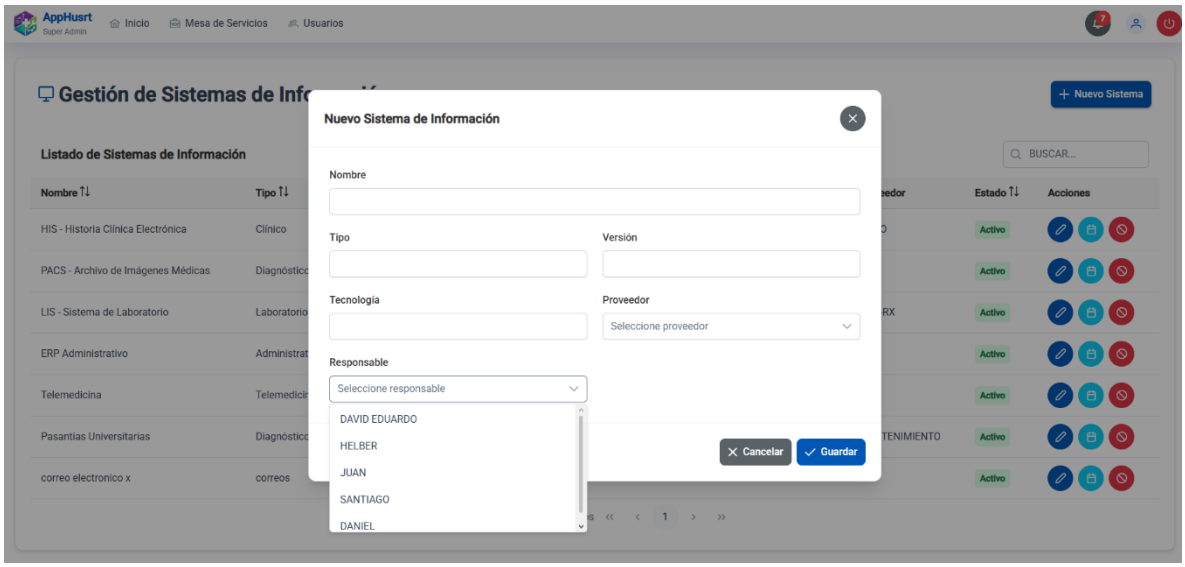
Posteriormente, en la épica 17, se eliminó el campo `periodicidad` del modelo `SistemaInformacion`, porque la periodicidad de backup pasó a derivarse exclusivamente del campo `frecuencia_backup` del backup más reciente de cada sistema. Esto simplificó la interfaz de usuario y evitó redundancia de información. En el frontend se retiró el campo de periodicidad editable y en la tabla se muestra un valor derivado (por ejemplo: "Diario", "Semanal", "Anual") consultando el último backup.



Evidencia 3. Dropdown de frecuencias en el formulario de backups.

Control de estado y relación con responsable y proveedor

El estado del sistema (activo, inactivo, en mantenimiento) se implementó como un enum en la base de datos y como un dropdown en el formulario frontend. La relación con responsable usuario del sistema y proveedor se validó mediante claves foráneas, y en la tabla se muestran los nombres completos.



Evidencia 4. Formulario de creación de sistema con dropdowns de responsable y proveedor.

6.4.2 Desarrollo del módulo de programación de backups

Consistió en construir el modelo de datos, la migración y la API CRUD para la programación de backups asociados a cada sistema de información, con frecuencias configurables (Diario, Semanal, Mensual, Anual), generación automática de todas las ocurrencias del año en curso y auto-transición a estado "No realizado" para backups vencidos sin ejecución.

Modelo de datos y migración

Se creó la tabla BackupSistema en MariaDB, con los siguientes campos: id (PK), sistemaInformacionId (FK hacia sistemasinformacion), fecha (DATE), frecuencia_backup (ENUM: 'Diario', 'Semanal', 'Mensual', 'Anual'), estado (ENUM: 'Programado', 'Completado', 'No realizado', 'Fallido'), observacion (TEXT, opcional), createdAt y updatedAt. Se definió la migración correspondiente y la asociación hasMany desde SistemaInformacion hacia BackupSistema.

API REST para backups

Se implementaron los siguientes endpoints, todos bajo el prefijo /api/backups:

- GET /api/backups?sistemaId=&mes=&anio= – lista backups filtrados por sistema, mes y año (útil para el calendario).
- GET /api/backups/:id – obtiene un backup específico.
- POST /api/backups – crea un nuevo backup para un sistema. Este endpoint dispara la generación automática de todas las ocurrencias futuras del año según la frecuencia seleccionada.
- PUT /api/backups/:id – actualiza un backup (estado, observación, fecha). Al cambiar el estado a "Completado", se invoca la lógica de auto-regeneración (épica 15).
- DELETE /api/backups/:id – elimina un backup (solo administradores).

Frecuencias configurables y generación anual automática (EP-10 y EP-20)

El campo frecuencia_backup se implementó como un enum con las siete opciones solicitadas. Al crear un backup mediante POST /api/backups, el backend calcula automáticamente todas las fechas siguientes para el resto del año según la frecuencia (por ejemplo: si se crea un backup semanal el 10 de abril, se generan registros con estado "Programado" para cada semana hasta el 31 de diciembre). Para las frecuencias "Quincenal", "Trimestral" y "Semestral" se implementaron funciones de cálculo específicas que respetan los intervalos exactos.

Auto-transición a estado "No realizado" (EP-18 y EP-19)

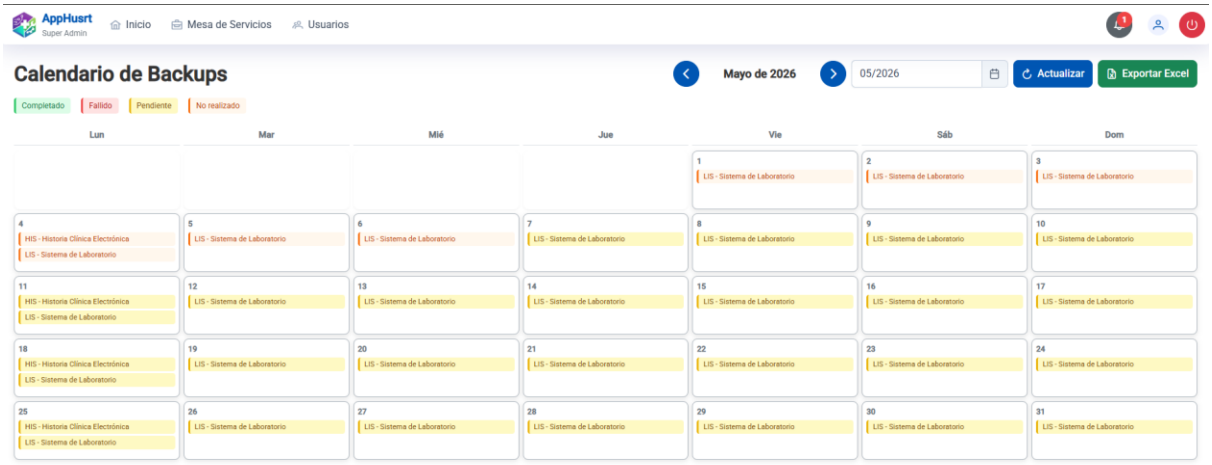
Se añadió el estado "No realizado" al enum del campo estado. En el endpoint GET /api/backups, antes de devolver los resultados, se ejecuta una rutina que identifica aquellos backups con estado "Programado" cuya fecha sea anterior al día actual. Dichos backups se actualizan automáticamente a "No realizado". Esta lógica también se aplica en el endpoint de alertas.
Con esto se garantiza que ningún backup vencido quede marcado como "Programado" incorrectamente.

Unificación de periodicidad (EP-17)

Como se mencionó en la subsección 6.4.1, el campo periodicidad del sistema de información fue eliminado. Ahora, cuando se necesita mostrar la periodicidad de un sistema (por ejemplo, en la tabla de sistemas), el frontend consulta el backup más reciente de ese sistema y extrae su frecuencia_backup. Esto evita la duplicación de información y mantiene la coherencia.

6.4.3 Evidencias objetivo 3 - Desarrollo del calendario visual interactivo

El tercer objetivo específico consistió en desarrollar un calendario visual interactivo que permita a los usuarios del DTI monitorear el estado de los backups de cada sistema de información, con codificación por color según estado (Programado, Completado, No realizado, Fallido), navegación mensual, tooltips informativos, modal de gestión con observación opcional, exportación a Excel y diseño responsive para múltiples dispositivos.



Evidencia 5. Captura del calendario principal mostrando la grilla mensual con los chips/badges de backups.

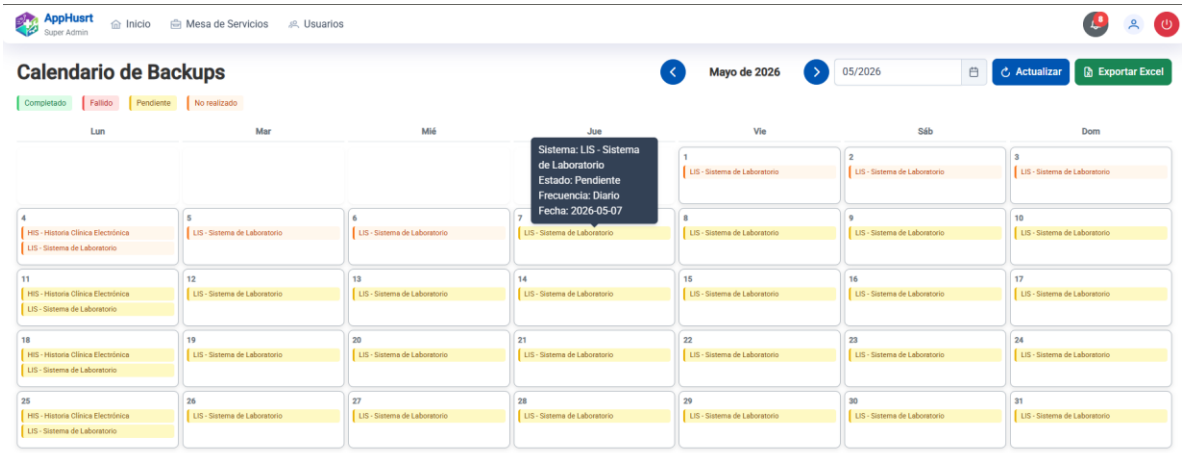
Rediseño del calendario (EP-13)

Inicialmente se exploró una representación con celdas de tabla completas coloreadas, pero resultaba confusa al mostrar múltiples backups por un mismo sistema en un día. Se optó por un diseño basado en chips o badges individuales, cada uno con una barra lateral del color correspondiente al estado del backup. El calendario se construyó con una grilla mensual donde cada día muestra los backups programados en esa fecha, agrupados visualmente por sistema. Se añadieron botones de navegón para avanzar y retroceder de mes, y se mejoró la tipografía del

encabezado. Se implementaron media queries para tablet y pantallas pequeñas, garantizando que los chips no se superpongan y que la legibilidad se mantenga.

Tooltips informativos (EP-13)

Al hacer hover sobre cada chip de backup, se muestra un tooltip con información detallada: nombre del sistema, frecuencia, fecha, estado y observación (si existe). Esto permite al usuario obtener una vista rápida sin necesidad de abrir el modal. La lógica de tooltips se implementó con PrimeNG (pTooltip).

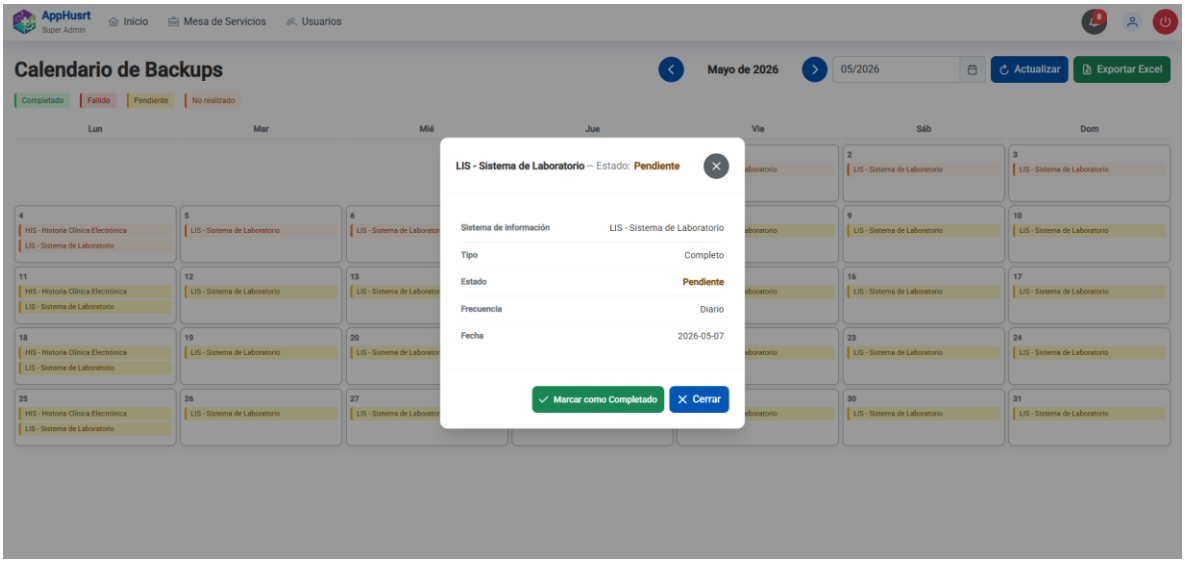


Evidencia 6. Captura del tooltip al hacer hover sobre un chip, mostrando detalles (sistema, fecha, estado, observación).

Modal de detalle y gestión (EP-14)

Al hacer clic en un chip del calendario, se abre un modal que muestra todos los campos del backup: sistema, fecha, frecuencia, estado (con un badge del color correspondiente) y observación. Dentro del modal se incluyen botones para editar el backup (si el usuario tiene permisos) y para marcar como completado (si está en estado Programado o No realizado). Al

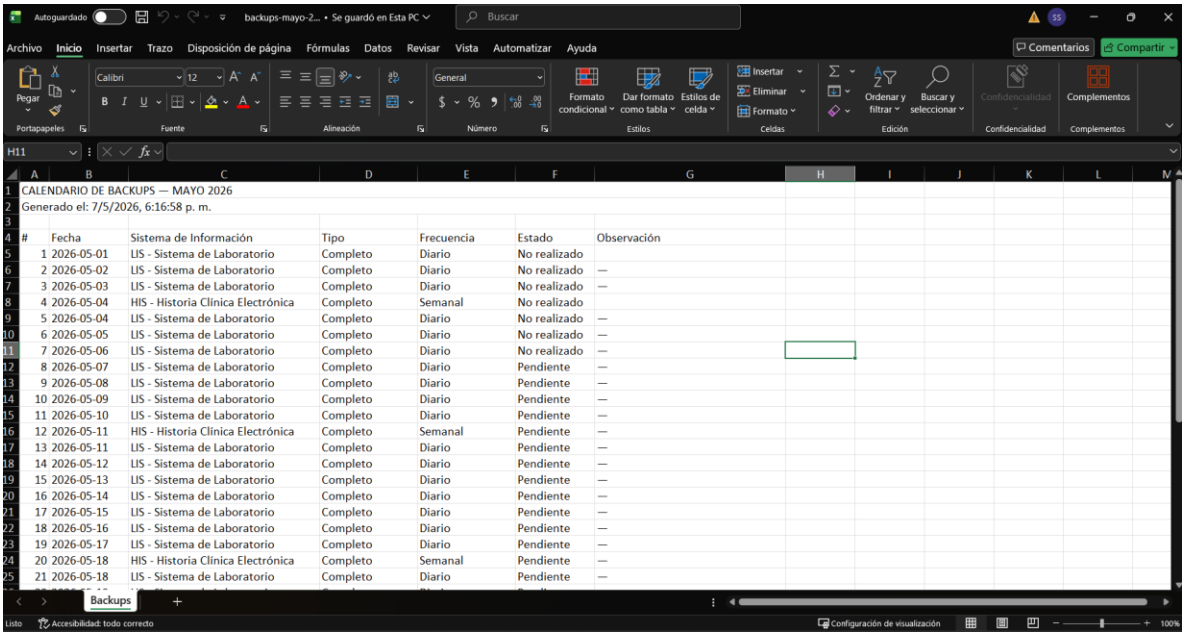
marcar como completado, se lanza una confirmación con SweetAlert2 que permite ingresar una observación opcional.



Evidencia 7. Captura del modal que aparece al hacer clic en un chip, mostrando todos los campos del backup (sistema, fecha, frecuencia, estado, observación) y los botones (editar, marcar como completado).

Exportación a Excel (EP-16)

Se implementó una funcionalidad de exportación del calendario a un archivo Excel. La exportación incluye: título del reporte, fecha y hora de generación, una tabla con las columnas (sistema, fecha, frecuencia, estado, observación) y filas coloreadas según el estado del backup (verde para Completado, naranja para No realizado, rojo para Fallido, azul para Programado). Además, se añaden conteos resumen por estado al final de la hoja. Esta funcionalidad permite al jefe del DTI generar reportes para auditorías o para la gerencia.



CALENDARIO DE BACKUPS — MAYO 2026
Generado el: 7/5/2026, 6:16:58 p. m.

#	Fecha	Sistema de Información	Tipo	Frecuencia	Estado	Observación
1	2026-05-01	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	
2	2026-05-02	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	—
3	2026-05-03	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	—
4	2026-05-04	HIS - Historia Clínica Electrónica	Completo	Semanal	No realizado	
5	2026-05-04	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	—
6	2026-05-05	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	—
7	2026-05-06	LIS - Sistema de Laboratorio	Completo	Diario	No realizado	—
8	2026-05-07	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
9	2026-05-08	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
10	2026-05-09	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
11	2026-05-10	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
12	2026-05-11	HIS - Historia Clínica Electrónica	Completo	Semanal	Pendiente	—
13	2026-05-11	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
14	2026-05-12	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
15	2026-05-13	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
16	2026-05-14	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
17	2026-05-15	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
18	2026-05-16	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
19	2026-05-17	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—
20	2026-05-18	HIS - Historia Clínica Electrónica	Completo	Semanal	Pendiente	—
21	2026-05-18	LIS - Sistema de Laboratorio	Completo	Diario	Pendiente	—

Evidencia 8. Muestra del archivo Excel generado al hacer clic al botón de exportación a Excel

Codificación por color y actualización del estado "No realizado" (EP-22)

Inicialmente el estado "No realizado" se mostraba en color gris, pero tras una revisión con el equipo del DTI, se cambió a color naranja para hacerlo más visible y diferenciarlo claramente de los estados "Programado" (azul), "Completado" (verde) y "Fallido" (rojo). Este cambio se aplicó

en el calendario (chips y barra lateral), en el modal (badge), en la leyenda del calendario y en la exportación a Excel. La leyenda del calendario se ubica en la parte superior derecha y explica cada color.

Diseño responsive

Se utilizaron media queries en CSS (con breakpoints para 768px, 1024px y 1280px) para adaptar la disposición de los chips y el tamaño de las celdas del calendario según el ancho de la pantalla. En dispositivos móviles (menos de 768px) los chips se apilan verticalmente dentro de cada día, y la navegación se simplifica con botones grandes.

6.4.4 Desarrollo del sistema de notificaciones y alertas reactivas

El desarrollo de esta funcionalidad consistió en implementar un sistema de notificaciones y alertas reactivas que permita al personal del DTI conocer de forma oportuna los backups pendientes o vencidos, mediante una campana con badge en el navbar, un panel overlay de alertas pendientes, auto-regeneración del siguiente backup al completar el actual y sincronización en tiempo real entre el calendario y el panel de alertas.

Endpoint de alertas y lógica de negocio (EP-09)

Se creó el endpoint GET /backups/alertas que evalúa el estado de los backups según la frecuencia configurada y la fecha actual. Este endpoint devuelve una lista de backups que cumplen alguna de estas condiciones: (a) backups con estado Programado cuya fecha sea igual o anterior al día actual (vencidos o por vencer hoy), (b) backups con estado No realizado (ya

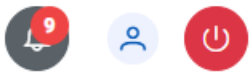
transicionados automáticamente) que aún no han sido atendidos. La lógica de auto-transición a No realizado ya descrita en 6.4.2 garantiza que las alertas solo muestren registros relevantes.

Servicio de notificaciones en frontend con polling (EP-09)

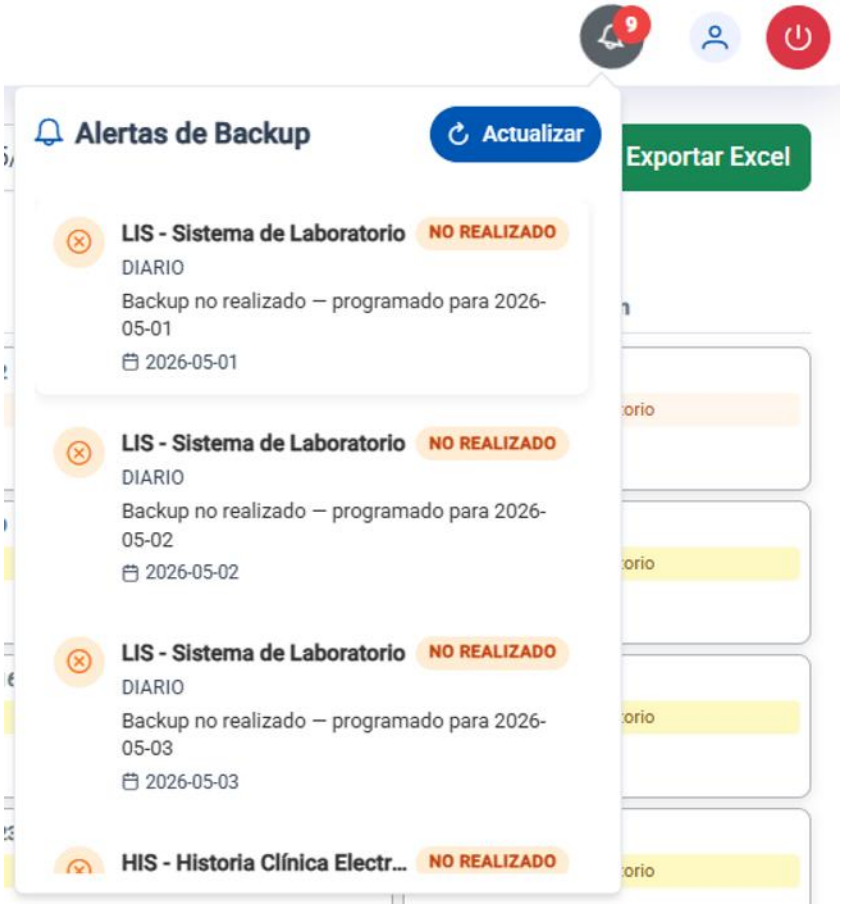
En el frontend se implementó el servicio NotificacionBackupService, que utiliza un BehaviorSubject para mantener el estado de las alertas y un mecanismo de polling (consulta periódica cada 30 segundos) al endpoint de alertas. Cada vez que se reciben nuevas alertas, el servicio actualiza el contador de la campana y la lista de alertas pendientes. El servicio se inyecta en el componente raíz y está disponible globalmente.

Componentes visuales: campana, badge y panel overlay (EP-09)

En el navbar (barra de navegación superior) se agregó un ícono de campana. Sobre él se muestra un badge (contador circular) con el número de alertas pendientes. Al hacer clic en la campana, se despliega un panel overlay que lista las alertas, mostrando para cada una: nombre del sistema, frecuencia, días de atraso (si aplica) y fecha del backup. Cada alerta tiene un botón para marcar como completado directamente desde el panel. El panel se cierra al hacer clic fuera.



Evidencia 9. Campana con badge en navbar mostrando contador de alertas pendientes.



Evidencia 10. Panel overlay con lista de alertas (sistema, días de atraso, fecha, acciones).

Auto-regeneración del siguiente backup (EP-15)

Cuando un usuario marca un backup como Completado (ya sea desde el modal del calendario o desde el panel de alertas), el backend ejecuta una lógica que crea automáticamente el siguiente registro Pendiente. El cálculo de la siguiente fecha se basa en la frecuencia del backup (por ejemplo: semanal suma 7 días, mensual suma un mes, etc.). Si la siguiente fecha cae en domingo o festivo (según calendario institucional), se ajusta al siguiente día hábil. Este proceso evita que el personal del DTI tenga que crear manualmente cada backup recurrente.

Sincronización en tiempo real entre calendario y panel de alertas (EP-12 y EP-15)

Para garantizar que el calendario y el panel de alertas estén siempre sincronizados, se utilizó el mismo BehaviorSubject del servicio de notificaciones. Cuando el usuario marca un backup como completado, se actualiza el estado en la base de datos y, tras la respuesta exitosa, el servicio notifica a todos los componentes suscritos. El componente del calendario refresca la vista (vuelve a cargar los backups del mes) y el panel de alertas actualiza su lista y el contador del badge. Además, el polling periódico actúa como mecanismo de respaldo para capturar cambios realizados desde otra sesión.

Eliminación de botón de completar en el panel (EP-16 - mejora UX)

En la épica 16, por petición del equipo del DTI, se eliminó el botón "Marcar como Completado" dentro del panel de alertas, dejando el panel únicamente como informativo. El usuario debe ir al calendario o al modal para completar un backup, lo que reduce la dispersión de acciones y evita errores. Esta decisión se tomó tras pruebas de usabilidad con el personal del área.

6.4.5 Desarrollo del control de acceso granular

Este desarrollo se enfocó en establecer un control de acceso granular al módulo, combinando el rol del usuario y los sistemas que tiene asignados. Se incluyó la creación del nuevo rol SYSTEMUSER, un endpoint centralizado de decisión de acceso, un guard de Angular con caché invalidable, y restricciones diferenciadas en lectura y escritura tanto en el backend como en la interfaz de usuario.

Creación del rol SYSTEMUSER

Se añadió el rol SYSTEMUSER al sistema de autenticación existente en HUSRT-Biomédica. Este rol está pensado para el personal técnico del DTI y para pasantes, otorgando permisos elevados sobre el módulo de gestión de sistemas y backups, pero sin ser un superadministrador del hospital.

Endpoint centralizado de decisión de acceso (EP-23)

Se implementó el endpoint GET /usuarios/me/acceso-modulo-sistemas, que retorna un objeto JSON con la siguiente estructura: { puedeAcceder: boolean, rol: string, sistemasAsignados: number[] }. Este endpoint verifica el token JWT del usuario, extrae su rol y la lista de sistemas de información que tiene asignados (si es un SYSTEMUSER o ADMIN, puede ver todos; si es un usuario normal, solo los sistemas que le hayan sido asignados). La lógica centralizada permite cambiar las reglas de acceso en un solo lugar, sin modificar cada endpoint del módulo.

Guard de Angular con caché invalidable (EP-23)

En el frontend se creó un guard de Angular (Route Guard) que se ejecuta antes de activar cualquier ruta del módulo (por ejemplo, /admin/sistemas, /calendario, /alertas). El guard consulta el endpoint centralizado y, según la respuesta, permite o deniega el acceso. Para evitar múltiples peticiones al backend mientras el usuario navega dentro del módulo, el guard implementa un sistema de caché simple: la primera consulta se almacena en memoria y las siguientes se sirven desde caché durante 30 segundos. Si el usuario cierra sesión o se produce un cambio de rol (por ejemplo, un administrador modifica sus permisos), el caché se invalida forzando una nueva consulta.

Restricciones diferenciadas en backend (EP-21)

En cada endpoint del módulo (sistemas, backups, alertas, mesa de servicios, reportes) se añadió un middleware que verifica el rol y los sistemas asignados. Las reglas son:

- SUPERADMIN: acceso total (lectura y escritura) sobre todos los sistemas.
- SYSTEMADMIN: acceso total (lectura y escritura) sobre todos los sistemas.
- SYSTEMUSER: acceso de lectura sobre todos los sistemas, pero solo puede escribir (crear, editar, eliminar) en los sistemas que tiene asignados. Además, puede marcar cualquier backup como completado (porque es una acción operativa).
    - Otros roles (ej: médico, enfermera): acceso denegado al módulo completo (redirigir a página de acceso denegado).

Los endpoints GET filtran los resultados de manera automática: si el usuario no es administrador, solo devuelven los sistemas asignados a él. Los endpoints POST, PUT y DELETE validan el permiso mediante el middleware antes de ejecutar la operación.

Restricciones diferenciadas en frontend (EP-21)

La interfaz de usuario oculta o deshabilita elementos según los permisos del usuario que ha iniciado sesión. Utilizando el mismo servicio que consume el endpoint centralizado (con caché), el frontend:

- Oculta los botones de "Crear sistema", "Editar sistema" y "Eliminar sistema" para usuarios que no son SUPERADMIN, SYSTEMADMIN o SYSTEMUSER con permiso sobre ese sistema específico.

- Oculta los botones de "Crear backup", "Editar backup" y "Eliminar backup" bajo la misma lógica.
- Para usuarios SYSTEMUSER, muestra el botón "Marcar como Completado" en el calendario y en el modal, pero no los botones de creación o edición de backups.
- Muestra un mensaje informativo "No tienes permisos para realizar esta acción" si el usuario intenta acceder a una ruta no autorizada (capturado por el guard).

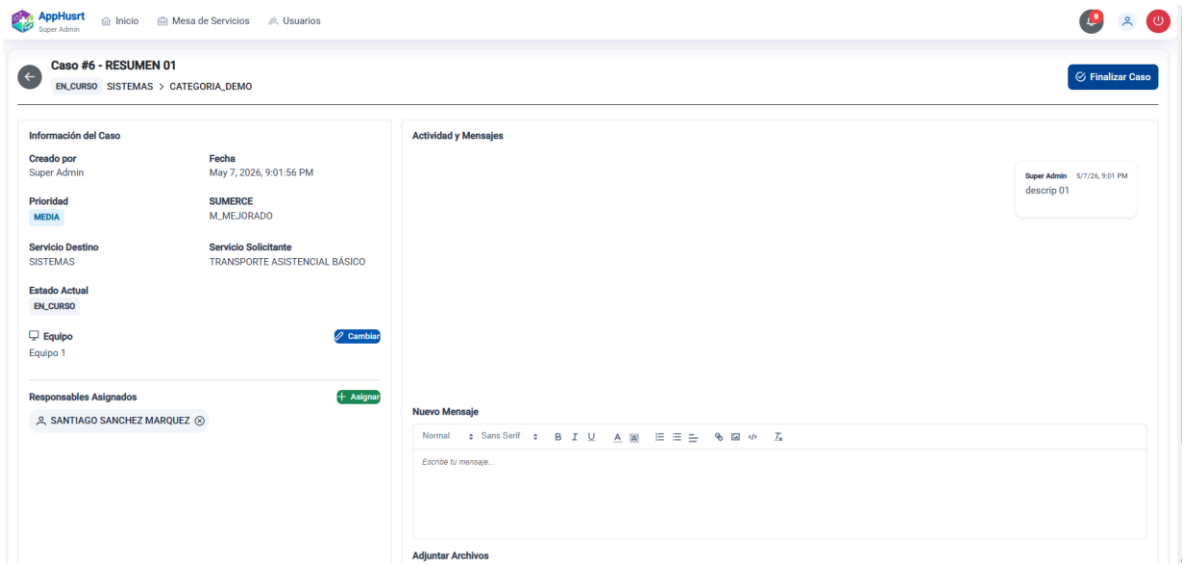
Ocultamiento de enlaces en navbar y hub (EP-23)

Además del guard en las rutas, se ocultaron los enlaces de navegación hacia el módulo en el navbar y en el hub (página principal) para usuarios sin permiso. Esto se logró mediante el mismo servicio de acceso, que expone un observable booleano (puedeAcceder). Los componentes del navbar y del hub se suscriben a este observable y muestran u ocultan los enlaces dinámicamente.

6.4.6 Desarrollo de la integración de Mesa de Servicios con inventario

Aquí se buscó integrar la Mesa de Servicios existente con el inventario de equipos de sistemas, mediante una asociación opcional entre cada caso de servicio y un equipo, e implementar un flujo de cierre de caso que redirige al técnico a un formulario completo de reporte de mantenimiento, garantizando la trazabilidad del trabajo técnico realizado sobre cada equipo de sistemas.

Evidencia 11. Listado de reportes por equipo en /reportesequipo.



Evidencia 12. Dashboard del caso donde se puede escoger responsable y equipos asignados

Extensión del modelo de datos para asociar equipo acaso (EP-24)

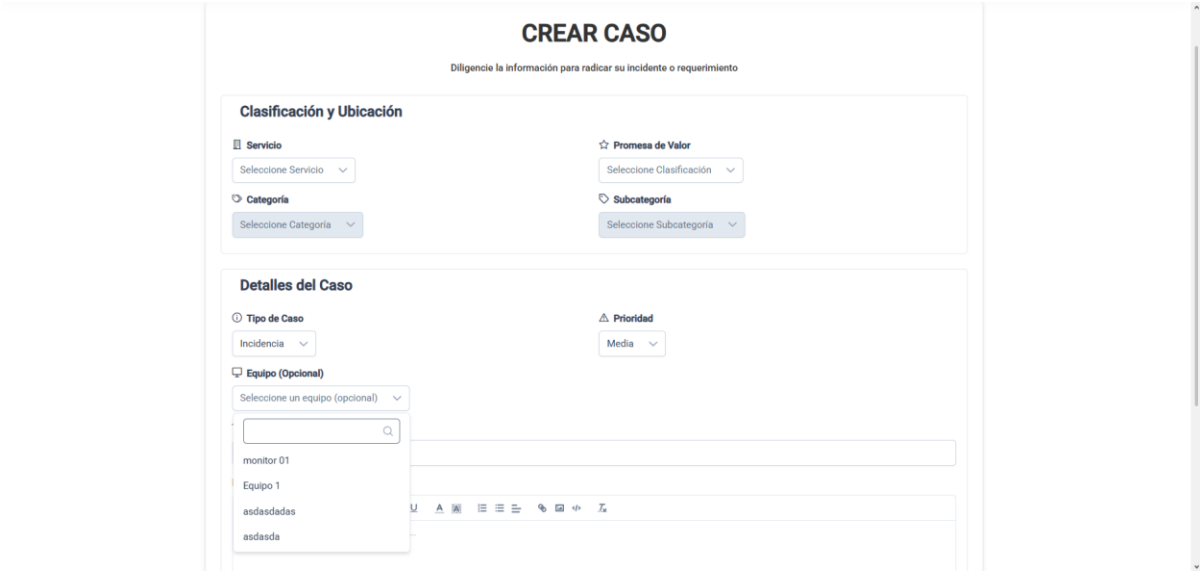
Se agregó la clave foránea opcional equipoId al modelo MesaCaso, referenciando la tabla SysEquipo (inventario de equipos de sistemas). La migración correspondiente se creó y ejecutó sin afectar los registros existentes (el campo puede ser nulo). Se actualizó el modelo Sequelize para definir la relación belongsTo entre MesaCaso y SysEquipo.

Endpoints modificados para soportar equipoId (EP-24)

Los endpoints POST /api/casos y PUT /api/casos/:id se modificaron para aceptar un campo opcional equipoId. Se agregaron validaciones para comprobar que el equipoId referencie un equipo existente en la tabla SysEquipo. Además, los endpoints GET /api/casos y GET /api/casos/:id ahora incluyen en la respuesta el objeto completo del equipo asociado (si existe). Esto permite mostrar el nombre, serial, ubicación y estado del equipo en la interfaz de usuario.

Frontend: dropdown de equipos en formulario de casos (EP-24)

En el formulario de creación y edición de casos de la Mesa de Servicios se agregó un dropdown (p-dropdown) que carga la lista de equipos de sistemas desde el backend. El dropdown es opcional: el usuario puede dejar el campo vacío si el caso no está relacionado con un equipo específico. Al seleccionar un equipo, el campo equipoId se envía junto con los demás datos del caso. En el detalle del caso (vista de solo lectura) se muestra la información del equipo asociado y se permite editarla (cambiar o eliminar la asociación) mediante un botón que abre un modal de selección.



Evidencia 13. Vista del formulario de caso con dropdown de equipos desplegado.

Endpoint de casos cerrados por equipo (EP-25)

Se implementó el endpoint GET /api/casos/equipo/:id_sysequipo que retorna todos los casos de servicio que estén asociados a un equipo específico y que tengan estado "Cerrado". Este endpoint es útil para generar reportes históricos de mantenimiento por equipo. La respuesta incluye: título del caso, descripción, prioridad (con badge), fechas de creación y cierre, y nombre del creador.

Modal de visualización de casos cerrados (EP-25)

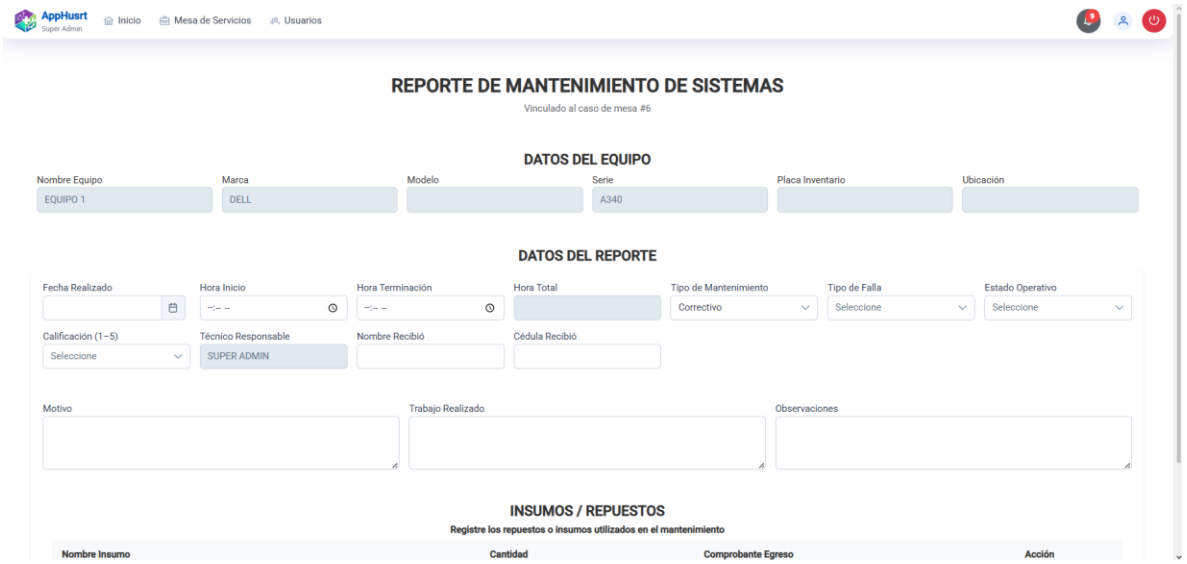
En la vista de detalle de un equipo (dentro del módulo de inventario), se agregó un botón "Ver casos cerrados". Al hacer clic, se abre un modal que consulta el endpoint anterior y muestra la lista de casos cerrados asociados a ese equipo. Cada caso se presenta con su título, prioridad (color: alta = rojo, media = amarillo, baja = verde), fecha de cierre y un enlace para ver el caso completo. El modal permite al técnico y al jefe del DTI revisar el historial de incidencias de un equipo, lo que facilita la toma de decisiones sobre reparaciones o reemplazos.

Extensión del modelo SysReporteMantenimiento (EP-26)

Se extendió el modelo SysReporteMantenimiento (que originalmente podía tener solo unos pocos campos) para incluir los siguientes campos obligatorios en el flujo de cierre de caso: hora_inicio, hora_fin, tipo_falla (eléctrica, mecánica, software, conectividad, otro), motivo (texto corto), descripción_detallada (texto largo), piezas_reemplazadas (texto), calificación (escala 1 a 5), y recomendaciones. Se creó la migración correspondiente y se añadió una relación belongsTo entre SysReporteMantenimiento y MesaCaso (un caso puede tener un reporte de mantenimiento, opcional).

Flujo de cierre de caso con redirección a formulario de reporte de mantenimiento (EP-26)

Cuando un técnico cierra un caso desde la Mesa de Servicios, en lugar de cerrarlo directamente se redirige automáticamente a un formulario dedicado en la ruta /adminsistemas/nuevoreporte/:id_caso. Este formulario contiene todos los campos del modelo SysReporteMantenimiento. El técnico debe completar los campos obligatorios antes de guardar. Al guardar, se crea un registro en la tabla SysReporteMantenimiento, se actualiza el estado del caso a "Cerrado", y se asocia el reporte al caso. De esta forma, se garantiza que cada cierre tenga un reporte completo y estructurado.



Evidencia 14. Formulario de reporte de mantenimiento al cerrar un caso (ruta /nuevoreporte).

Rediseño visual: eliminación de modal embebido por rutas dedicadas (EP-27)

En una iteración posterior (épica 27), se rediseñó la experiencia de usuario: se eliminó el modal que antes contenía el formulario de reporte (visto en versiones anteriores del módulo) y se crearon dos rutas Angular dedicadas:

- /adminsistemas/nuevoreporte/:id_caso – para crear un nuevo reporte al cerrar un caso.
- /adminsistemas/reportesequipo/:id_equipo – para ver el historial de reportes de un equipo (listado y detalle).

Los dropdowns de equipos ahora enlazan directamente a estas rutas en lugar de abrir modales. Este cambio mejoró la estabilidad del formulario (evitando pérdida de datos por cierres accidentales del modal) y unificó la experiencia con el resto del módulo biomédico.

Evidencias objetivo 4 - Validación y pruebas del módulo desarrollado

Para garantizar la calidad del software y verificar el cumplimiento de los requisitos funcionales y técnicos, se realizó un proceso de validación que combinó pruebas unitarias sobre componentes críticos del backend, pruebas funcionales sobre la interfaz de usuario, y la aprobación formal por parte del tutor empresarial.

Pruebas funcionales: validación de requerimientos

Con el fin de asegurar que el módulo cumpliera con todos los requisitos planteados al inicio del proyecto, se elaboró una matriz de requerimientos funcionales donde se especificó para cada uno su criterio de éxito y el resultado de la validación. Esta tabla permitió realizar un seguimiento sistemático de cada funcionalidad y verificar que todas operaban conforme a lo esperado antes de proceder con el despliegue.

Módulo / Área	Requerimientos cubiertos	Criterio de éxito general	Validado
Parametrización de sistemas de información	Registro, edición y eliminación de sistemas con atributos de criticidad, tipo y responsable técnico	El sistema permite gestionar el catálogo completo de sistemas con sus metadatos operativos	Sí
Programación de backups	Definición de frecuencia, tipo (completo/incremental/diferencial), ventana horaria y política de retención por sistema	Cada sistema registrado puede tener una política de backup activa, editable y trazable	Sí

Calendario visual de backups	Visualización mensual/semanal de ejecuciones programadas con diferenciación por estado (pendiente, exitoso, fallido)	El técnico puede identificar el estado de cualquier backup del período sin consultar registros en texto	Sí
Alertas y notificaciones reactivas	Generación automática de alertas ante fallos, vencimientos de retención y backups no ejecutados en su ventana	Las alertas se generan sin intervención manual y quedan registradas con marca de tiempo y sistema afectado	Sí
Control de acceso granular	Asignación de roles diferenciados (administrador, técnico, auditor) con permisos por módulo y restricción de vistas	Cada usuario accede únicamente a las funciones y datos correspondientes a su rol asignado	Sí
Integración de Mesa de Servicios con inventario	Vinculación de tickets de soporte con equipos del inventario, registro de repuestos utilizados e historial por técnico	Un ticket puede trazarse desde su apertura hasta el cierre con el equipo, técnico y materiales asociados	Sí

En la tabla anterior se puede observar que la totalidad de los requerimientos funcionales definidos para el módulo fueron validados exitosamente, incluyendo la parametrización de sistemas, la programación de backups, el calendario visual, las alertas reactivas, el control de acceso granular y la integración con la Mesa de Servicios.

Pruebas unitarias: validación de formularios críticos

Las pruebas unitarias se enfocaron en los formularios más críticos del sistema, donde la integridad de los datos es fundamental para la operación. En particular, se validó el formulario de creación y edición de sistemas de información, que incluye las siguientes reglas de negocio:

- El campo nombre es obligatorio y debe ser único en el sistema, para evitar duplicidades.
- El campo estado no puede quedar vacío y solo admite los valores Activo, Inactivo o En mantenimiento.
- Los campos responsable y proveedor deben corresponder a registros existentes en las tablas de usuarios y proveedores respectivamente, garantizando la integridad referencial.
- El campo URL (cuando se proporciona) debe tener un formato válido.

Estas validaciones se implementaron tanto en el backend (mediante validaciones con Sequelize y en los controladores de la API) como en el frontend (mediante validaciones reactivas en Angular), asegurando una doble capa de protección contra datos incorrectos.

```
Archivo: NodeBackendProyectHusrt-biomedica-general/routes/biomedica/sistemaInformacionRoutes.js

Validación de rol y acceso (líneas 9–44)

// líneas 9–13 – solo admins pueden crear/editar/eliminar
function requireAdminRole(req, res, next) {
    if (!ROLES_ADMIN.includes(req.user?.rol)) {
        return res.status(403).json({ error: 'No tiene permisos para realizar esta acción' });
    }
    next();
}
```

Evidencia 15. Middleware requireAdminRole que valida que el usuario tenga rol de administrador (líneas 9-13 del archivo sistemaInformacionRoutes.js). Si no lo tiene, retorna un error 403 con el mensaje "No tiene permisos para realizar esta acción".

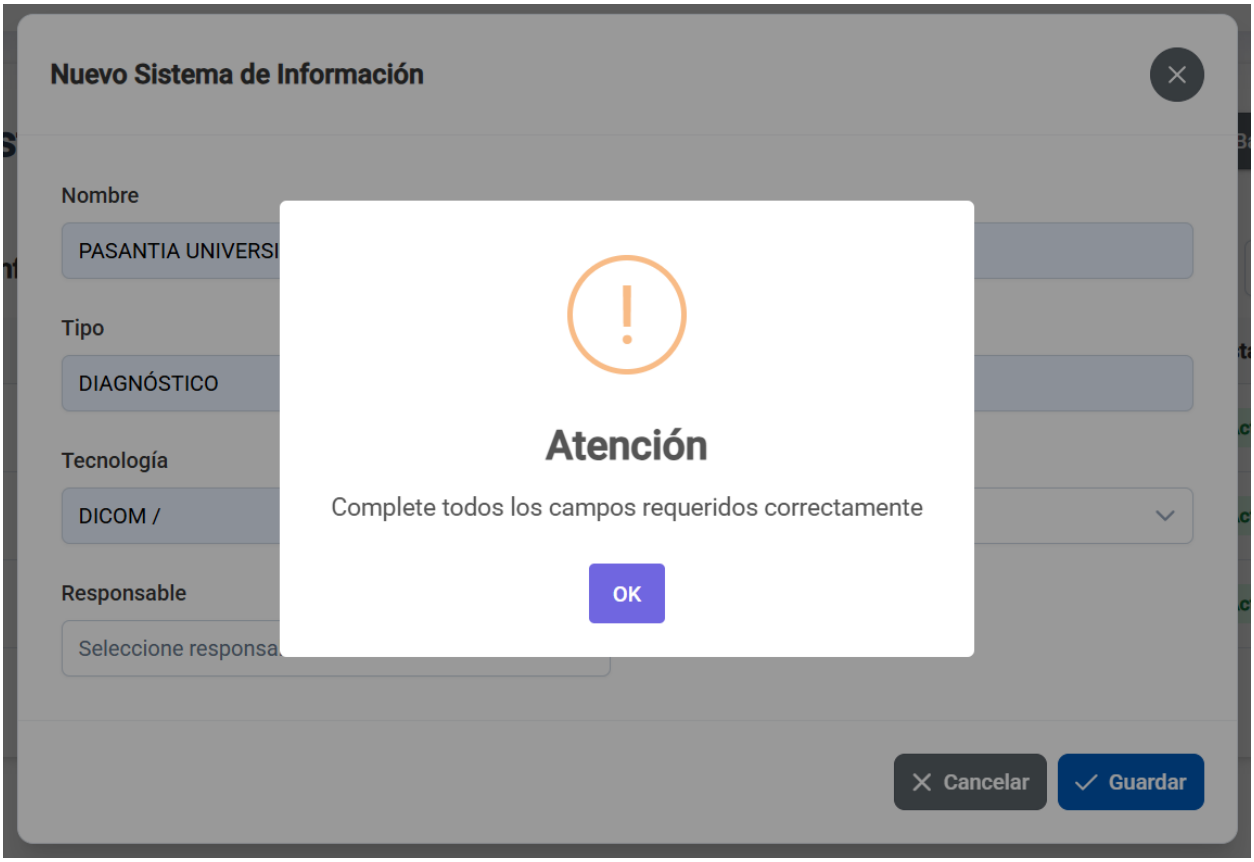
```
// líneas 19–44 – acceso al módulo
async function requireSistemasModuloAccess(req, res, next) {
    if (!ROLES_MODULO_SISTEMAS.includes(rol))
        return res.status(403).json({ error: 'Rol no autorizado para el módulo de sistemas' });
    // ...usuarios sin rol admin deben tener al menos 1 sistema asignado
    if (tieneSistemas === 0)
        return res.status(403).json({ error: 'No tiene sistemas de información asignados' });
}
```

Evidencia 16. Middleware requireSistemasModuloAccess que valida dos condiciones: que el usuario tenga un rol autorizado para el módulo de sistemas (ROLES_MODULO_SISTEMAS) y que, si no es administrador, tenga al menos un sistema de información asignado. Si no cumple alguna, retorna error 403.

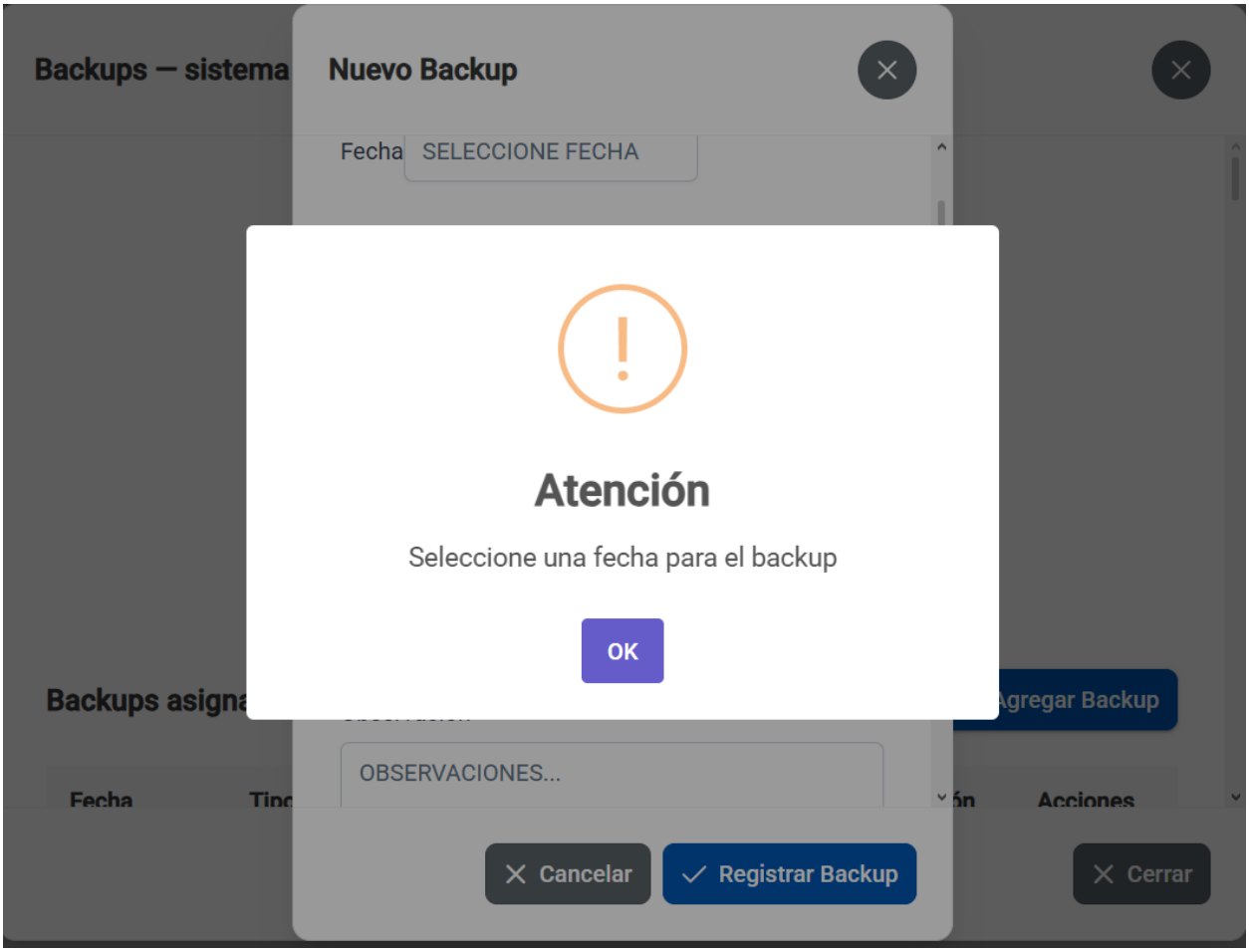
```
Verificación de existencia al editar (líneas 136–139)

// PUT /sistemainformacion/:id – líneas 134–145
const sistema = await SistemaInformacion.findByPk(req.params.id);
if (!sistema) {
    return res.status(404).json({ error: 'Sistema de información no encontrado' });
}
```

Evidencia 17. Verificación de existencia del sistema de información al momento de editarlo (líneas 136-139). Si el sistema no existe en la base de datos, retorna un error 404 con el mensaje "Sistema de información no encontrado".



Evidencia 18. Validación en el frontend: mensaje de error al intentar guardar un sistema sin completar todos los campos requeridos.



Evidencia 19. Validación en el frontend: mensaje de error al intentar registrar un backup sin seleccionar una fecha.

Capa	Archivo	Mecanismo	Campos validados
Frontend	admsistemasinformacion.component.ts L.92–99	Validators.required en FormGroup	nombre, tipo, version, tecnologia, responsableId, proveedorId
Frontend	admsistemasinformacion.component.ts L.147–150	formGroup.invalid → SweetAlert2	(todos los campos)
Backend (rutas)	sistemaInformacionRoutes.js L.9–44	Middleware de rol y acceso	rol de usuario, sistemas asignados
Backend (rutas)	sistemaInformacionRoutes.js L.136–139	findByPk + respuesta 404	existencia del registro al editar
Modelo Sequelize	SistemaInformacion.js	allowNull: false solo en estado	estado (booleano)

Pruebas de la lógica de backups: diagrama de flujo

Para explicar visualmente el comportamiento del subsistema de backups, se elaboró un diagrama de flujo que representa el ciclo de vida completo de un backup, desde su programación inicial hasta su finalización o vencimiento. El diagrama ilustra los siguientes pasos:

1. El usuario programa un backup con una frecuencia determinada (Diaria, Semanal, Quincenal, Mensual, Trimestral, Semestral o Anual).

2. El sistema genera automáticamente todas las ocurrencias programadas para el año en curso.
3. Cada backup permanece en estado Programado hasta que se ejecuta o vence.
4. Si la fecha del backup es anterior al día actual y no ha sido ejecutado, el sistema lo transiciona automáticamente al estado No realizado.
5. Cuando el usuario marca un backup como Completado, el sistema genera automáticamente el siguiente registro según la frecuencia configurada (auto-regeneración).
6. Los estados posibles son: Programado, Completado, No realizado y Fallido.

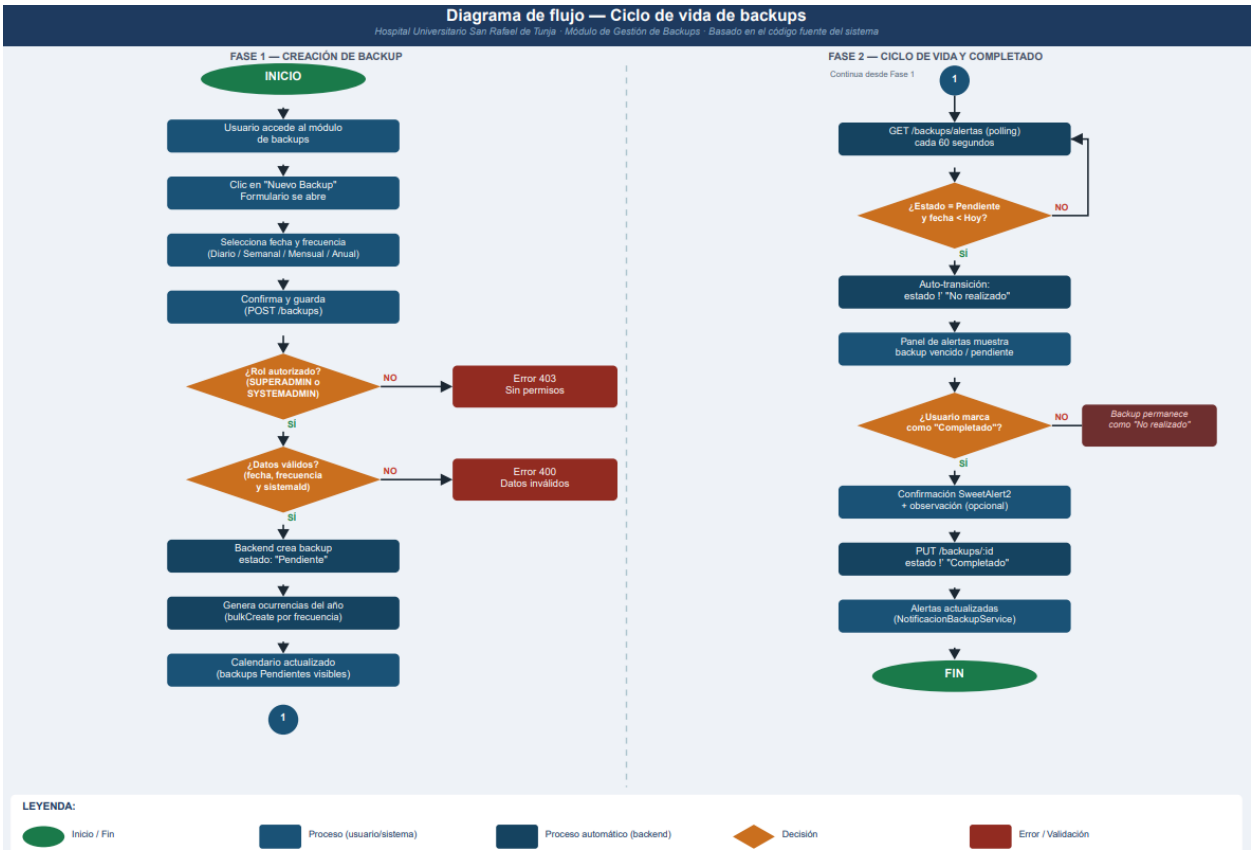


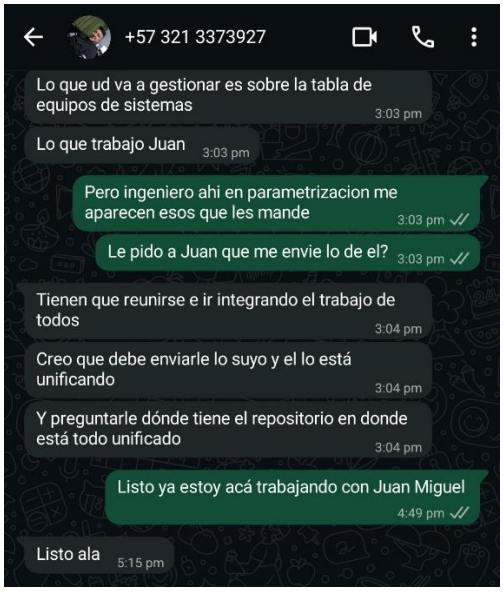
Figura 3. Diagrama de flujo – Ciclo de vida de backups

Este flujo garantiza que ningún backup quede sin seguimiento y que el personal del DTI siempre tenga visibilidad del estado de cada respaldo programado.

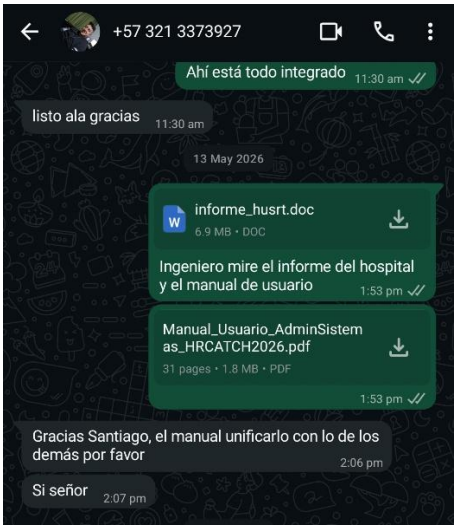
Aprobación del tutor empresarial

El módulo fue sometido a una validación final por parte del tutor empresarial, Ing. David Guerrero, del Departamento de Tecnologías de la Información del HUSRT. Durante esta validación, se verificó el cumplimiento de los requerimientos funcionales, la usabilidad del sistema y la correcta integración con el entorno de producción del HUSRT-Biomédica.

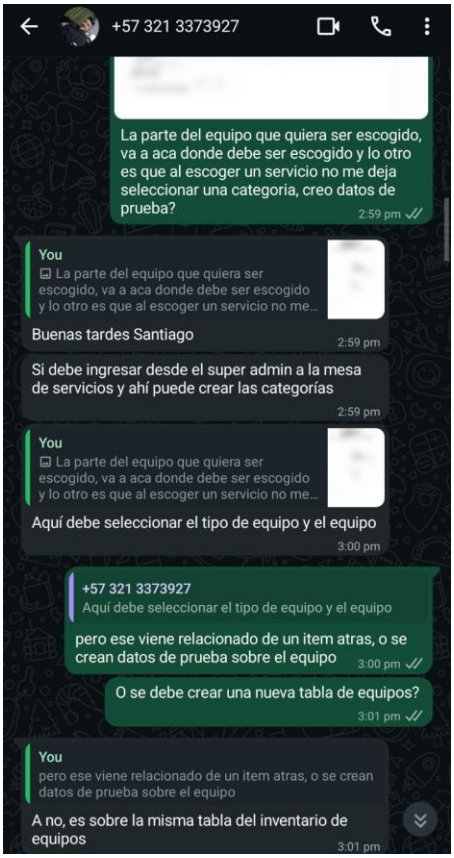
Solicitud del tutor para coordinar con los compañeros Daniel y Diego la definición de indicadores de mantenimiento correctivo a partir de los reportes generados por la Mesa de Servicios.



Instrucciones del tutor para integrar los trabajos de los diferentes pasantes en un repositorio unificado, coordinandocon Juan Miguel la consolidación del código y la documentación.



Conversación con el tutor empresarial, Ing. David Guerrero, en la que se envía el informe del proyecto y el manual de usuario para su revisión. El tutor solicita unificar el manual con los demás módulos del sistema.



Consulta al tutor sobre la integración de la Mesa de Servicios con el inventario de equipos. El tutor explica que el equipo debe seleccionarse desde la tabla de equipos de sistemas (inventario) y que las categorías deben crearse desde el rol de super administrador.

El tutor empresarial aprobó el módulo, confirmando que cumple con los objetivos planteados y que está en condiciones de ser utilizado por el personal del DTI.

Manual de usuario

Como parte del proceso de validación y para facilitar la adopción del módulo por parte de los usuarios finales, se elaboró un manual de usuario que describe paso a paso el uso de cada funcionalidad.

HOSPITAL UNIVERSITARIO SAN RAFAEL DE TUNJA

SISTEMA DE GESTIÓN TECNOLÓGICA (HRCATCH)

MANUAL DE USUARIO

Módulo de Sistemas

Juan Miguel Ramírez Mancilla

Santiago Sanchez Marquez

Daniel Estiven Ramírez

Diego Alexis Moreno Valero

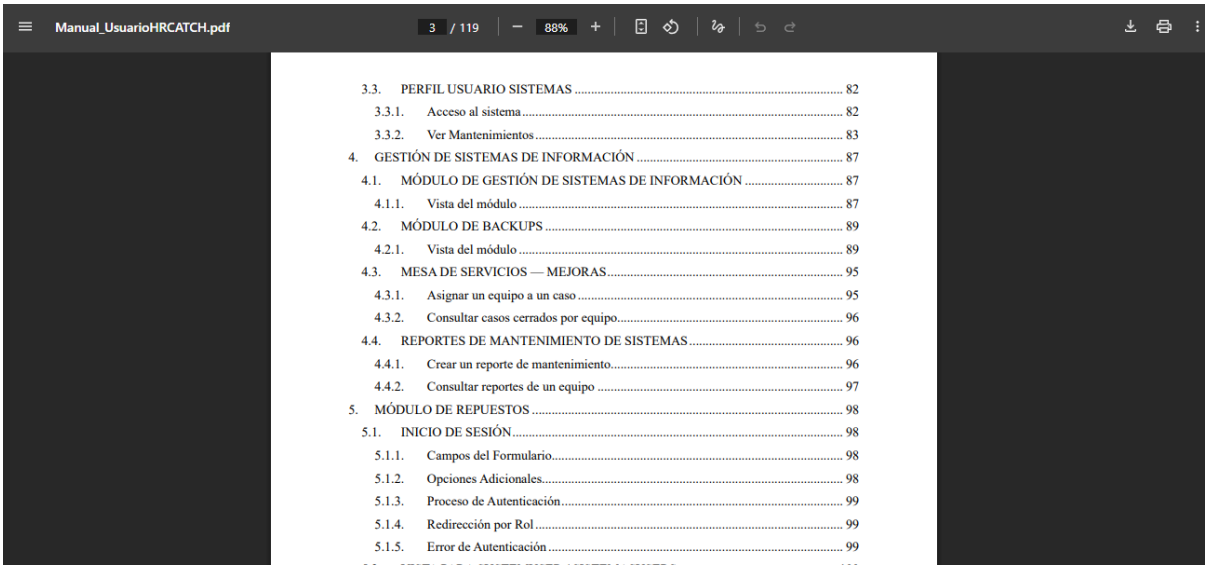
Perfiles: AdminSistemas • Técnico Sistemas • UsuarioSistemas

Anexo. Manual de Usuario

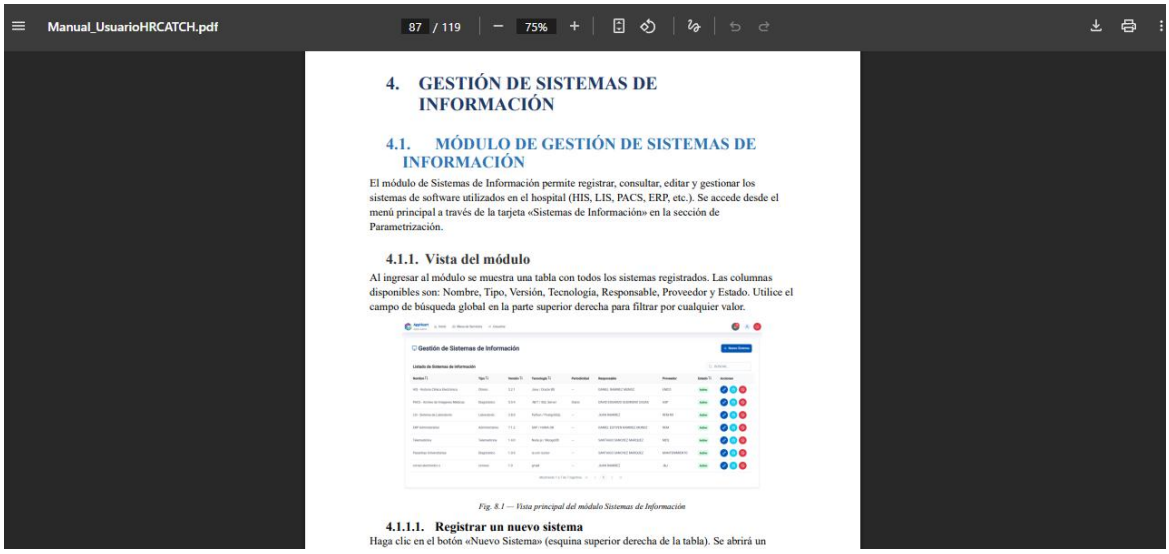
El manual de usuario completo, desarrollado por los miembros de la pasantia, se adjunta como un archivo PDF independiente junto con este documento. En él se describen paso a paso las funcionalidades del módulo: parametrización de sistemas, programación de backups, uso del calendario, gestión de alertas y cierre de casos en la Mesa de Servicios, junto a las demás funcionalidades hechas por los demás compañeros de la pasantía.

El manual incluye capturas de pantalla y explicaciones detalladas de:

- Cómo acceder al módulo y gestionar sistemas de información.
- Cómo programar y monitorear backups.
- Cómo utilizar el calendario visual y las alertas.
- Cómo cerrar casos de la Mesa de Servicios con reporte de mantenimiento.



Índice del manual de usuario del módulo desarrollado, que incluye: perfil de usuario, gestión de sistemas de información, gestión de backups, mejoras en la Mesa de Servicios, reportes de mantenimiento y módulo de repuestos.



Vista del módulo de gestión de sistemas de información según el manual de usuario. Muestra la tabla con los sistemas registrados, sus columnas (Nombre, Tipo, Versión, Tecnología, Responsable, Proveedor, Estado), el campo de búsqueda y los botones de acción (Guardar, Editar, Eliminar, Cancelar, Salir).

Las pruebas unitarias y funcionales, junto con la aprobación del tutor empresarial, confirman que el módulo desarrollado cumple con los requisitos establecidos, opera de manera estable y está listo para su uso en el Hospital Universitario San Rafael de Tunja. La documentación generada (matriz de requerimientos, diagrama de flujo y manual de usuario) respalda la calidad del software y facilitará su mantenimiento futuro.

7. CONCLUSIONES

Cumplimiento de los objetivos

Se cumplieron la totalidad de los objetivos específicos planteados en la sección 5.2. En primer lugar, se levantaron los requerimientos del software mediante la metodología ágil llamada historias de usuario buscando lo que quería que el tutor empresarial para el desarrollo del módulo se diseñó e implementó el módulo de parametrización de sistemas de información, incluyendo el modelo de datos en MariaDB, la API REST en Express con Sequelize y la vista CRUD en Angular con PrimeNG, estableciendo la relación con responsable y proveedor, así como el control de estado. En segundo lugar, se construyó el modelo de datos, la migración y la API CRUD para la programación de backups con siete frecuencias configurables (diario, semanal, mensual y anual), junto con la generación automática de todas las ocurrencias del año en curso y la auto-transición a estado "No realizado" para los backups vencidos sin ejecución.

Asimismo, se desarrolló un calendario visual interactivo de backups con codificación por color según el estado, navegación mensual, tooltips informativos, modal de gestión con observación opcional, exportación a Excel y diseño responsive para múltiples dispositivos. De igual forma, se implementó un sistema de notificaciones y alertas reactivas mediante una campana con badge en el navbar, un panel overlay de alertas pendientes, la auto-regeneración del siguiente backup al completar el actual y la sincronización en tiempo real entre el calendario y el panel de alertas.

También se estableció un control de acceso granular al módulo, combinando el rol del usuario y los sistemas asignados. Esto incluyó la creación del nuevo rol SYSTEMUSER, un endpoint centralizado de decisión de acceso, un guard de Angular con caché invalidable, y restricciones diferenciadas en lectura y escritura tanto en el backend como en la interfaz de usuario. Finalmente, se integró la Mesa de Servicios con el inventario de equipos de sistemas mediante una asociación opcional entre cada caso de servicio y un equipo, y se implementó un flujo de cierre de caso que redirige al técnico a un formulario completo de reporte de mantenimiento, garantizando la trazabilidad del trabajo técnico realizado sobre cada equipo de sistemas.

Todos los entregables fueron validados por el Departamento de TI del HUSRT y se encuentran operativos en el sistema HUSRT-Biomédica en producción.

Dificultades y retos enfrentados

Durante la ejecución de la pasantía se presentaron varias dificultades que requirieron ajustes constantes y trabajo en equipo. Una de ellas fue la integración del código desarrollado con los cambios realizados simultáneamente por otros pasantes en el mismo repositorio del sistema HUSRT-Biomédica. Los conflictos de merge fueron frecuentes, especialmente en archivos compartidos como los servicios de autenticación y los componentes base del navbar. Sin embargo, mediante una comunicación continua, reuniones breves diarias para coordinar las áreas de trabajo, y el uso disciplinado de ramas en Git, se logró unificar los avances sin pérdida de funcionalidades.

Otra dificultad importante surgió al implementar el formulario de reporte de mantenimiento que aparece al cerrar un caso de la Mesa de Servicios. El formulario no se mostraba correctamente debido a una redundancia en la lógica del frontend: el componente estaba intentando reinicializarse con datos de un tipo de equipo que no existía en el contexto del cierre de caso. La solución consistió en reiniciar explícitamente el estado del formulario desde el servicio de tiposEquipo, asegurando que cada nuevo reporte partiera de un objeto vacío y con las validaciones activas. Este error, aunque costó varias horas de depuración, permitió comprender mejor el ciclo de vida de los componentes en Angular.

En el sistema de notificaciones reactivas se presentó un problema inicial: el panel de alertas mostraba absolutamente todos los backups programados para el futuro, incluyendo aquellos con fechas muy lejanas. Esto saturaba la interfaz y restaba utilidad a las alertas. Se modificó la lógica del endpoint para que solo retornara los backups cuya fecha programada fuera igual al día actual o anterior (es decir, backups pendientes para hoy o vencidos). Con este ajuste, las alertas se volvieron relevantes y accionables, mejorando la experiencia del usuario.

Finalmente, el mayor reto técnico fue asumir el desarrollo con Angular para un proyecto de tamaño real. Antes de la pasantía, el conocimiento del framework era básico, basado en tutoriales y proyectos pequeños. Enfrentarse a un sistema en producción con múltiples módulos, guards, servicios, componentes standalone y comunicación con una API REST

compleja requirió una curva de aprendizaje intensiva. Gracias a este desafío, se adquirió una competencia sólida en Angular, incluyendo el manejo de observables (RxJS), la inyección de dependencias, el enrutamiento con guards y la integración de librerías como PrimeNG.

Aprendizajes y competencias desarrolladas

La pasantía representó una experiencia de aprendizaje integral tanto en el plano técnico como en el profesional. En el ámbito técnico, se fortaleció el modelado de datos relacional (diseño de tablas, claves foráneas, migraciones en MariaDB) y el desarrollo de APIs REST con Express.js y Sequelize, incluyendo la lógica de negocio para generación recurrente de registros y auto-transición de estados. En el frontend, se adquirió un dominio práctico de Angular, especialmente en componentes standalone, servicios con inyección de dependencias, guards de rutas, y el manejo de observables (RxJS) para la sincronización en tiempo real entre el calendario y el panel de alertas. También se aprendió a integrar librerías como PrimeNG y SweetAlert2 para construir interfaces de usuario robustas y responsivas.

En cuanto a competencias transversales, se desarrolló la capacidad de trabajar en un repositorio compartido con otros desarrolladores, gestionando conflictos de merge mediante ramas y comunicación continua. Se fortaleció la habilidad para depurar errores complejos (como el bug en el formulario de reporte de mantenimiento) utilizando herramientas del navegador y logs del backend. Además, se aprendió a adaptarse a cambios en los requerimientos sin perder el rumbo del proyecto, replanificando sprints y reescribiendo

partes del código cuando fue necesario. Desde el punto de vista profesional, la interacción diaria con el equipo del Departamento de TI y la validación de cada funcionalidad con usuarios reales permitió comprender la importancia de escuchar al cliente y de documentar las decisiones técnicas. Finalmente, trabajar en un entorno hospitalario de alta complejidad sensibilizó sobre el valor crítico de la información clínica y la responsabilidad que implica desarrollar software para este sector.

Impacto concreto en el HUSRT

La implementación del módulo generó un impacto positivo y medible en la operación del Departamento de Tecnologías de la Información. En primer lugar, se resolvió la ausencia de un inventario centralizado de sistemas de información: actualmente se encuentran registrados sistemas (entre asistenciales, administrativos y de soporte), cada uno con su responsable, proveedor y estado, lo que ha facilitado la asignación de responsabilidades y la planificación de actualizaciones.

En cuanto a los backups, se automatizó por completo la programación y el seguimiento. El sistema genera automáticamente todas las ocurrencias anuales según la frecuencia configurada (diaria, semanal, etc.), eliminando la necesidad de registros manuales o recordatorios en hojas de cálculo. El calendario visual y las alertas reactivas han reducido el tiempo que el personal del DTI dedica a verificar el estado de los respaldos, pasando de una revisión manual dispersa (que podía tomar varias horas a la semana) a una consulta centralizada de menos de 30 minutos diarios. Además, la auto-transición a estado "No

realizado" y la generación del siguiente backup al completar el actual garantizan que ningún respaldo quede sin ejecutar, disminuyendo el riesgo de pérdida de información clínica y administrativa.

El control de acceso granular, con el nuevo rol SYSTEMUSER y la asignación de sistemas por usuario, ha mejorado la seguridad de la información. Ahora el personal técnico solo puede modificar los sistemas que le corresponden, y los accesos no autorizados se bloquean tanto en el backend como en la interfaz, en línea con los requisitos de la Ley 1581 de 2012.

La integración de la Mesa de Servicios con el inventario de equipos de sistemas ha permitido asociar cada caso de servicio a un equipo específico y generar reportes de mantenimiento estructurados al cerrar el caso. Esto ha proporcionado trazabilidad completa del trabajo técnico, facilitando el cálculo de indicadores como el tiempo medio de reparación (MTTR) y la identificación de equipos con fallos recurrentes, lo que a su vez justifica decisiones de compra o reemplazo.

Los principales beneficiarios son el Departamento de TI (que cuenta con una herramienta unificada y eficiente), los técnicos de soporte (que disponen de alertas oportunas y trazabilidad de su trabajo), y la gerencia del hospital (que puede auditar el cumplimiento de las políticas de respaldo y mantenimiento). En última instancia, el paciente se beneficia indirectamente, pues la disponibilidad y la integridad de la información clínica son esenciales para una atención segura y continua.

8. REFERENCIAS

- AXELOS. (2019). ITIL Foundation: ITIL 4 Edition. TSO (The Stationery Office).
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for agile software development. Agile Alliance. https://agilemanifesto.org
- Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley Professional.
- Congreso de la República de Colombia. (2012). Ley 1581 de 2012, por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial n.° 48587.
- Coronel, C., Morris, S., & Rob, P. (2019). Database systems: Design, implementation, & management* (13th ed.). Cengage Learning.
- Departamento Administrativo Nacional de Estadística. (2025). Proyecciones de población departamentales 2025. DANE. https://www.dane.gov.co
- Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, 4(3), 224–274. https://doi.org/10.1145/501978.501980

- Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures [Doctoral dissertation, University of California, Irvine]. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Google. (2024). Angular documentation: Introduction to the Angular framework. https://angular.dev/overview
- Hospital Universitario San Rafael de Tunja. (2024). Misión, visión, funciones y deberes. https://hospitalsanrafaeltunja.gov.co/mision-vision-funciones-y-deberes/
- International Organization for Standardization. (2022). ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection — Information security management systems — Requirements. ISO.
- Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT) (RFC 7519). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc7519
- Ministerio de Salud de Colombia. (1999). Resolución 1995 de 1999, por la cual se establecen normas para el manejo de la Historia Clínica*. Ministerio de Salud de Colombia.
- OpenJS Foundation. (2024a). Express.js: Fast, unopinionated, minimalist web framework for Node.js. https://expressjs.com
- OpenJS Foundation. (2024b). Node.js documentation: A JavaScript runtime built on Chrome's V8 JavaScript engine. https://nodejs.org/en/docs
- PrimeTek. (2024). PrimeNG: The most complete UI suite for Angular. https://primeng.org

- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. IEEE Computer, 29(2), 38–47. https://doi.org/10.1109/2.485845
- Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The definitive guide to Scrum: The rules of the game. Scrum.org. https://scrumguides.org/scrum-guide.html
- Sequelize Contributors. (2024). Sequelize v6 documentation: Promise-based Node.js ORM para bases de datos relacionales. https://sequelize.org/docs/v6/
- Superintendencia Nacional de Salud. (2016). Circular Externa 000002 de 2016: Instrucciones relativas a la gestión de la información en salud. Supersalud.

9. ANEXOS

Name sistemasinformacion Primary id

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	nombre	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
3	descripcion	text	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
4	tipo	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
5	estado	tinyint(1)	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
6	version	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
7	fecha_implementacion	date	NULL	NULL	YES	NULL	EMPTY	EMPTY	EMPTY
8	proveedor	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
9	responsable	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
10	tecnologia	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
11	url	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
12	createdAt	datetime	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
13	updatedAt	datetime	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
14	responsableId	int	NULL	NULL	YES	NULL	EMPTY	usuario(id)	EMPTY
15	proveedorId	int	NULL	NULL	YES	NULL	EMPTY	responsable(id)	EMPTY

index_name	index_algorithm	is_unique	column_name
PRIMARY	BTREE	TRUE	id
sistemasinformacion_proveedorId_fk	BTREE	FALSE	proveedorId
sistemasinformacion_responsableId_fk	BTREE	FALSE	responsableId

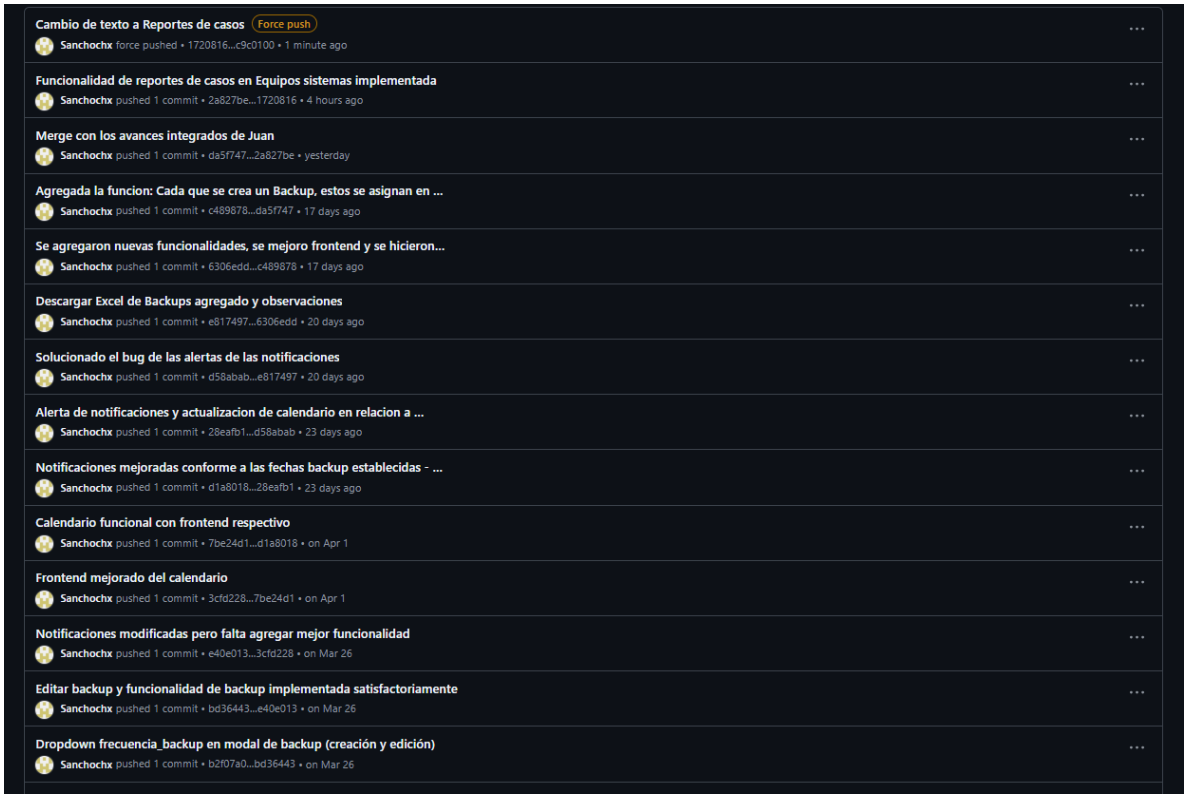
Anexo A.1 EP-01, EP-04 (Backend sistemas)

Tabla "sistemasinformacion" mostrada en el programa TablesPlus

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	EMPTY
2	sistemaInformacionId	int	NULL	NULL	NO	NULL	EMPTY	sistemasinformacion(id)	EMPTY
3	fecha	date	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
4	tipo	varchar(255)	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
5	estado	enum('Pendiente','Completado','Fallido','No realizado')	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
6	frecuencia_backup	enum('Anual','Mensual','Semanal','Diario')	utf8mb4	utf8mb4_unicode_ci	NO	Mensual	EMPTY	EMPTY	EMPTY
7	observacion	text	utf8mb4	utf8mb4_unicode_ci	YES	NULL	EMPTY	EMPTY	EMPTY
8	createdAt	datetime	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY
9	updatedAt	datetime	NULL	NULL	NO	NULL	EMPTY	EMPTY	EMPTY

Anexo A.2 EP-06, EP-07, EP-10, EP-20 (Modelo Backup, API CRUD, frecuencia y generación anual de backups)

Estructura de la tabla BackupSistema en MariaDB (clave foránea sistemaInformacionId).



Anexo A. Avance del proyecto mediante commits al repositorio de Github