UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, Decana de América)

Facultad de Ingeniería de Sistemas e Informática

**"ENTREGABLE – MÓDULO DE COMUNICADOS"**

**PRÁCTICAS PRE PROFESIONALES**

**PROFESOR**

**William Martín, Enriquez Maguiña**

**PRESENTADO POR**

**Verde Huaynaccero, Erick Edison**

**Lima-Perú**

**2025**

## Informe del Módulo de Comunicados – Sistema de Comunicación Institucional

### Objetivo del Entregable

Desarrollar e implementar un sistema integral de comunicación institucional para la I.E.P. Las Orquídeas que permita a directivos y docentes autorizados publicar comunicados oficiales (académicos, administrativos, eventos, urgentes e informativos) dirigidos a padres de familia, docentes o toda la comunidad educativa, con segmentación automática por nivel, grado y curso según las relaciones institucionales de cada usuario.

El propósito fue resolver la problemática de comunicación dispersa y sin trazabilidad que enfrentaba la institución, estableciendo un canal formal que garantice que cada familia reciba únicamente los comunicados relevantes para el nivel académico de sus hijos, que los docentes accedan a información institucional y a los comunicados que ellos mismos publican, y que el director supervise toda la comunicación institucional con capacidad de gestión administrativa.

Este sistema incorpora controles de acceso diferenciados por rol, validación de permisos de publicación para docentes, sanitización automática de contenido para prevenir inyección de código malicioso, publicación inmediata o programada con ventana mínima de seguridad, marcado

automático de lectura para métricas de adopción, y funcionalidades administrativas de desactivación y eliminación de comunicados.

La implementación respeta los principios de Proporcionalidad (artículo 8) y Seguridad (artículo 9) de la Ley N.° 29733 – Ley de Protección de Datos Personales, asegurando que solo se trate la información estrictamente necesaria y con medidas técnicas de protección. Además, se alinea con los lineamientos de comunicación institucional establecidos en la Ley 28044 – Ley General de Educación, y aplica buenas prácticas de los controles de seguridad de la información de la ISO/IEC 27001 (específicamente A.9 Control de Acceso, A.12 Seguridad de las Operaciones, A.14 Seguridad en el Desarrollo, y A.18 Cumplimiento) y de trazabilidad de procesos de la ISO/IEC 12207 (gestión de configuración, documentación y verificación).

**Metodología Aplicada**

El desarrollo del módulo siguió un proceso estructurado de cuatro etapas con validación continua:

1. **Análisis y refinación de requisitos:**
   Estudié las necesidades de comunicación de la institución educativa y las traduje en tres historias de usuario principales con criterios de aceptación verificables: HU-COM-00 para la bandeja de comunicados con filtros y segmentación automática, HU-COM-01 para la lectura completa con marcado automático de registro de visualización, y HU-COM-02 para la creación y publicación con validaciones de permisos y contenido. Documenté cada historia con condiciones funcionales, validaciones de negocio, mensajes de éxito y error, y flujos de estados esperados.
2. **Diseño de arquitectura y contratos de API:**
   Diseñé veinticuatro endpoints REST organizados en seis secciones funcionales: bandeja de comunicados, lectura de comunicado completo, creación y publicación, gestión de comunicados propios, validaciones auxiliares, y gestión administrativa solo para director. Especifiqué para cada endpoint el método HTTP, ruta, parámetros de entrada (query, path, body), estructura de respuesta de éxito y error con códigos HTTP apropiados, y reglas de negocio que determinan el comportamiento. Apliqué principios de consistencia en formatos de respuesta y códigos de error estandarizados.
3. **Implementación modular backend:**
   Implementé el módulo en tres capas independientes: capa de ruteo que define los endpoints HTTP y aplica middlewares de autenticación JWT, autorización por rol y limitación de tasa; capa de controladores que validan parámetros de entrada, invocan servicios de dominio y formatean respuestas con manejo de errores; capa de servicios que ejecutan reglas de negocio, consultas a base de datos, validaciones de permisos y segmentación, y cálculos de destinatarios. Modelé en la base de datos la entidad Comunicado con campos de contenido, tipo, estado, segmentación

mediante arrays, fechas de trazabilidad y vinculación al autor; y la entidad ComunicadoLectura con restricción única por comunicado y usuario para evitar duplicados.

4. **Validación funcional y generación de datos de prueba:**
   Desarrollé un script automatizado que genera comunicados de ejemplo con diferentes tipos, estados, autores, públicos objetivo y segmentaciones, junto con registros de lectura simulados. Esto me permitió validar manualmente los flujos de bandeja (verificando que cada rol vea solo comunicados pertinentes), lectura (verificando que se marque automáticamente y actualice contadores), creación (verificando validaciones de longitud, tipo y segmentación), publicación programada (verificando restricción de tiempo mínimo), edición (verificando ventana de veinticuatro horas), y operaciones administrativas de desactivación y eliminación.

**Herramientas Utilizadas**

- **Node.js con Express:** Plataforma backend para construcción de APIs REST con organización modular, middlewares de autenticación mediante tokens de sesión, control de acceso por rol de usuario y limitación de tasa para protección contra abuso.
- **Prisma ORM:** Herramienta de modelado de datos y consultas que facilita la definición de esquemas con tipos estrictos, relaciones entre entidades y generación automática de cliente de base de datos. Me permitió trabajar con arrays de segmentación mediante tipo de dato JSON y consultas eficientes usando operadores de pertenencia.
- **PostgreSQL en Neon:** Base de datos relacional gestionada en la nube que almacena los comunicados, registros de lectura, usuarios, permisos, estudiantes y relaciones familiares. Aproveché el soporte nativo para tipo de dato JSONB para almacenar arrays de segmentación con índices y consultas optimizadas.
- **Validación con expresiones regulares:** Implementé funciones de sanitización de contenido HTML que detectan y eliminan etiquetas peligrosas (script, iframe, object, embed) mediante patrones de búsqueda, preservando etiquetas seguras para formato básico (párrafos, negritas, listas, enlaces).
- **GitHub:** Sistema de control de versiones para gestionar cambios en código fuente, documentación técnica y scripts de utilidad, con historial trazable de modificaciones por módulo.
- **Limitadores de tasa (Rate limiters):** Configuración de restricciones diferenciadas por tipo de operación: treinta solicitudes por minuto para operaciones de lectura y diez solicitudes por minuto para operaciones de escritura, protegiendo el servidor contra saturación y abuso.

**Actividades Realizadas**

- Refiné tres historias de usuario principales (bandeja de comunicados, lectura completa, creación y publicación) con criterios de aceptación detallados que especifican validaciones de entrada, reglas de visibilidad por rol, mensajes de confirmación y error, y flujos de estados esperados.

- Diseñé y documenté veinticuatro endpoints REST agrupados por funcionalidad, especificando para cada uno el formato de solicitud y respuesta, códigos HTTP de éxito y error, reglas de negocio aplicables y restricciones por rol de usuario.
- Implementé la capa de ruteo definiendo métodos HTTP, rutas con parámetros dinámicos y middlewares de seguridad: autenticación obligatoria mediante token de sesión, autorización específica por rol (padre, docente, director) y limitación de tasa diferenciada por criticidad.
- Desarrollé dieciocho funciones de controlador que gestionan las operaciones del módulo: obtener lista de comunicados con paginación y filtros, contar comunicados no leídos, buscar por término, verificar actualizaciones mediante sondeo, obtener comunicado completo, marcar como leído, validar acceso, crear comunicado nuevo, guardar borrador, editar comunicado existente, listar borradores propios, publicar borrador, listar comunicados programados, cancelar programación, validar contenido HTML, validar segmentación de destinatarios, desactivar comunicado y eliminar comunicado permanentemente.
- Codifiqué lógica de negocio en servicios: verificar permisos de usuario según su rol y permisos explícitos en tabla de permisos docentes, verificar acceso del usuario a segmentación solicitada según sus asignaciones de cursos, validar fecha programada con ventana mínima de treinta minutos futuros, calcular destinatarios estimados según arrays de segmentación cruzados con estudiantes y relaciones familiares, generar previews de contenido limitados a ciento veinte caracteres, formatear fechas en formato legible y relativo, sanitizar HTML eliminando etiquetas peligrosas, y generar texto descriptivo de destinatarios para confirmación.
- Modelé dos entidades principales: Comunicado con campos de título, contenido, tipo, estado, segmentación mediante cuatro arrays (público objetivo, niveles, grados, cursos), fechas de creación, publicación, programación y edición, indicador de edición, vinculación al autor y año académico; y ComunicadoLectura con vinculación a comunicado y usuario, fecha de lectura y restricción única para evitar duplicados.
- Construí script de generación automática de datos de prueba que crea veinticinco comunicados variados en tipo, estado, autor, público objetivo y segmentación, con fechas de publicación distribuidas en los últimos noventa días y programaciones futuras. El script también genera registros de lectura aleatorios para aproximadamente sesenta por ciento de los comunicados publicados, simulando adopción real.
- Documenté servicio frontend de integración con funciones cliente que consumen los endpoints principales: verificar permisos de docente, obtener jerarquía de niveles y grados, validar contenido HTML, calcular preview de destinatarios, crear comunicado publicado o borrador, marcar como leído, obtener comunicado por identificador y validar acceso del usuario.
- Verifiqué manualmente los flujos críticos: listado de comunicados con segmentación automática por rol, búsqueda por título y contenido con resaltado de término encontrado, contador de no leídos con actualización tras cada lectura, lectura de comunicado con marcado automático de registro, creación de comunicado con validaciones de longitud y permisos, publicación

programada con validación de tiempo mínimo, edición de comunicado dentro de ventana de veinticuatro horas, desactivación y eliminación de comunicado por director.

**Evidencias**

- **Especificación técnica del módulo:** Documentación de historias de usuario con tres historias principales refinadas, veinticuatro endpoints REST especificados con ejemplos de entrada, salida y reglas de negocio, y modelo de datos con dos entidades y sus relaciones.
- **Implementación backend verificable:** Archivo de definición de rutas con configuración de autenticación, autorización y limitadores aplicados a cada grupo de endpoints; archivo de controladores con dieciocho funciones que gestionan validación de parámetros, invocación de servicios y formateo de respuestas; archivo de servicios con funciones de validación de permisos, cálculo de destinatarios, sanitización de HTML y gestión de estados de comunicado.
- **Estructura de datos normalizada:** Modelo de entidad Comunicado con enumeraciones de tipo, estado y prioridad, arrays para almacenar segmentación de destinatarios, campos de trazabilidad de fechas y relación al usuario autor; modelo de entidad ComunicadoLectura con clave única compuesta por identificador de comunicado e identificador de usuario para garantizar unicidad de registros de lectura.
- **Datos de prueba generados:** Script automatizado que crea veinticinco comunicados de ejemplo con distribución variada de tipos, estados, autores y segmentaciones, junto con registros de lectura simulados que representan aproximadamente sesenta por ciento de adopción en comunicados publicados.
- **Integración frontend documentada:** Servicio cliente con funciones de consumo de API que incluyen validaciones de entrada en el lado del cliente, normalización de payloads según especificación de API, y manejo estructurado de errores con códigos de respuesta HTTP.
- **Resultados de verificación funcional manual:**
  Validé que la bandeja de comunicados filtra automáticamente según el rol del usuario autenticado: un padre visualiza solo comunicados dirigidos a los niveles y grados donde tiene hijos matriculados (verificado mediante consulta de relaciones familiares); un docente visualiza comunicados institucionales dirigidos a todos o a docentes, más los comunicados que él mismo creó (verificado mediante filtro por autor o público objetivo); un director visualiza todos los comunicados sin restricción (verificado con consulta sin filtros de segmentación).
  Verifiqué que el marcado de lectura se registra una sola vez por usuario mediante la inserción con cláusula de conflicto que previene duplicados, y que el contador de no leídos se actualiza correctamente restando las lecturas registradas del total de comunicados visibles.
  Confirmé que la creación de comunicados valida la longitud del título (entre diez y doscientos caracteres), la longitud del contenido (entre veinte y cinco mil caracteres), el tipo de comunicado según rol (docentes solo académico y evento, director todos los tipos), y la segmentación

solicitada según las asignaciones del docente (verificado con consulta de asignaciones a cursos activas).
Validé que la publicación programada requiere una fecha y hora con al menos treinta minutos de antelación respecto al momento actual, y que el comunicado queda en estado programado hasta que se alcance la fecha establecida.
Comprobé que la edición de comunicados publicados solo se permite dentro de las primeras veinticuatro horas posteriores a la publicación, y que los borradores se pueden editar en cualquier momento sin restricción.
Verifiqué que la desactivación de comunicados (solo director) oculta el comunicado de las bandejas de todos los usuarios pero mantiene el registro en la base de datos para auditoría, y que la eliminación permanente (solo director con confirmación explícita) remueve el comunicado y sus registros de lectura asociados.

**Dificultades Encontradas**

- **Complejidad de segmentación automática por rol y relaciones institucionales:** Diseñar la lógica para determinar qué comunicados ve cada usuario según su rol, vínculos familiares y asignaciones docentes implicó consultas con múltiples tablas relacionadas (usuarios, estudiantes, relaciones familiares, asignaciones docente-curso, niveles y grados). Resolví esto construyendo consultas dinámicas que filtran por arrays de segmentación usando operadores de pertenencia, y validando manualmente los resultados con diferentes combinaciones de roles y segmentaciones mediante el script de datos de prueba.
- **Sanitización de contenido HTML preservando formato:** Enfrenté el desafío de eliminar elementos peligrosos del contenido HTML sin destruir el formato visual que los usuarios esperan (negritas, listas, enlaces). Implementé un sanitizador que usa patrones de búsqueda para detectar y remover etiquetas inseguras (script, iframe, object, embed) mientras preserva etiquetas de formato básico (párrafos, strong, em, ul, ol, li, a). Verifiqué que la sanitización protege contra inyección de código sin afectar la legibilidad del contenido final.
- **Gestión de estados y transiciones de comunicado:** Diseñar las reglas de transición entre estados (borrador puede pasar a publicado o programado; publicado puede pasar a desactivado; programado puede pasar a publicado automáticamente o cancelarse a borrador) requirió definir validaciones claras en cada operación. Documenté cada regla de transición posible y las validaciones que la permiten o bloquean, y verifiqué los flujos mediante pruebas manuales de cada caso.
- **Ventana de edición y control de versiones:** Implementar la restricción de edición solo dentro de veinticuatro horas post-publicación implicó calcular el tiempo transcurrido desde la fecha de publicación y compararlo contra la ventana permitida. Decidí que los borradores no tienen restricción de edición, pero los comunicados publicados sí, para equilibrar flexibilidad con

estabilidad de información ya comunicada. Registré la fecha de edición y un indicador booleano para trazabilidad de cambios.
- **Trazabilidad única de lecturas sin duplicados:** Asegurar que cada usuario pueda marcar un comunicado como leído una sola vez, sin generar registros duplicados en operaciones simultáneas o reintentos, requirió usar una restricción única en la base de datos sobre la combinación de identificador de comunicado e identificador de usuario. Implementé esta restricción en el modelo de datos y usé una instrucción de inserción condicional que ignora el intento si ya existe el registro.

**Lecciones Aprendidas**

- Reforcé la importancia de diseñar controles de acceso granulares que vayan más allá de la simple autenticación. Aprendí que verificar que un usuario esté autenticado (tiene sesión válida) no es suficiente; debo verificar también que su rol le permita ejecutar la operación (autorización de endpoint), y además que tenga permisos específicos para los recursos solicitados (autorización de segmentación). Este enfoque de defensa en profundidad, alineado con el control A.9.1 (Requisitos de negocio para control de acceso) de la ISO/IEC 27001, me permitió aplicar el principio de mínimo privilegio en cada capa del sistema.
- Consolidé mi comprensión del principio de Proporcionalidad establecido en el artículo 8 de la Ley 29733. Comprendí que este principio no solo significa "no pedir datos innecesarios", sino también "no mostrar información que no corresponde". Al segmentar comunicados automáticamente según el rol y vínculos del usuario, garanticé que un padre no vea comunicados dirigidos solo a docentes o a niveles donde no tiene hijos matriculados. Esta aplicación concreta del principio me enseñó a integrar requisitos legales en decisiones de diseño técnico.
- Aprendí que la seguridad del contenido generado por usuarios requiere validación doble: sanitización en el frontend para mejorar la experiencia (prevenir envío de contenido claramente inválido) y sanitización obligatoria en el backend para garantía real de protección (prevenir bypass maliciosos). Implementar esta doble barrera me permitió entender las buenas prácticas del control A.14.2 (Seguridad en los procesos de desarrollo) de la ISO/IEC 27001, donde cada entrada de usuario debe considerarse no confiable hasta validación técnica.
- Desarrollé capacidad para diseñar validaciones de negocio que reflejen restricciones operativas reales. La ventana de edición de veinticuatro horas post-publicación no es arbitraria, sino que balancea la necesidad de corregir errores recientes con la estabilidad de información ya comunicada y potencialmente leída por múltiples usuarios. Esta decisión de diseño refleja un entendimiento de que las reglas técnicas deben servir necesidades institucionales concretas.
- Comprendí el valor de la trazabilidad de acciones para auditoría y análisis. Registrar cada lectura con fecha y usuario, cada edición con indicador booleano y fecha, cada desactivación con estado y timestamp, facilita no solo métricas de adopción (cuántos usuarios leyeron cada comunicado),

sino también auditoría de cambios (quién modificó qué y cuándo). Esta trazabilidad es fundamental para cumplir con el control A.12.4 (Registro y Monitoreo) de la ISO/IEC 27001.
- Perfeccioné la habilidad de documentar decisiones técnicas con justificación normativa. Cada control implementado (sanitización, validación de permisos, segmentación automática, ventana de edición) está vinculado a un principio legal (Ley 29733) o buena práctica de seguridad (ISO/IEC 27001), lo que fortalece la trazabilidad entre requisitos regulatorios y controles operativos. Esta práctica facilita auditorías y demuestra cumplimiento ante revisiones externas.

**Competencias Desarrolladas en el Módulo de Comunicados**

**Competencias Genéricas**

**CG1 – Valores, compromiso ético y social**

Durante el desarrollo del módulo de comunicación institucional, apliqué con rigor el principio de Proporcionalidad establecido en el artículo 8 de la Ley N.° 29733 al diseñar la segmentación automática de comunicados. Cada usuario visualiza únicamente los comunicados que le conciernen según su rol y vínculos institucionales: un padre solo accede a comunicados dirigidos a los niveles y grados donde tiene hijos matriculados, un docente solo accede a comunicados institucionales o a los que él mismo creó, y el director supervisa todos sin restricción. Esta segmentación técnica operacionaliza el mandato legal de tratar solo datos estrictamente necesarios para la finalidad declarada (comunicación institucional relevante).

Implementé el principio de Seguridad del artículo 9 de la misma ley mediante controles técnicos en tres capas: autenticación obligatoria con tokens de sesión firmados, autorización diferenciada por rol en cada endpoint, y sanitización automática de contenido HTML para eliminar elementos peligrosos que podrían comprometer la integridad del sistema o la confidencialidad de otros usuarios. Además, apliqué el control A.9.4.1 (Restricción de acceso a la información) de la ISO/IEC 27001 al validar que docentes solo puedan publicar comunicados de tipos autorizados (académico y evento) y solo dirigidos a grados donde tienen asignaciones activas.

Comprendí que en un entorno educativo donde se gestionan datos de menores de edad y se emiten comunicaciones oficiales a familias, la responsabilidad ética trasciende el cumplimiento formal de normativas. Cada control de acceso que implementé (padre no ve comunicados de otros grados, docente sin permisos no puede publicar, director supervisa todo con trazabilidad) protege el derecho de cada familia a privacidad informativa y comunicación pertinente. Esta aplicación práctica del compromiso ético refuerza mi conciencia profesional sobre el impacto social de las decisiones técnicas en sistemas que sirven a comunidades vulnerables.

**CG3 – Capacidad de análisis y pensamiento crítico**

Ejercité el análisis sistemático al descomponer el problema de comunicación institucional en requisitos funcionales y no funcionales verificables. Identifiqué casos problemáticos: ¿qué sucede si un docente intenta publicar un comunicado urgente sin tener permisos explícitos? (respuesta: el sistema valida permisos en tabla PermisoDocente y bloquea la operación con mensaje claro); ¿cómo garantizar que un padre no acceda a comunicados dirigidos solo a docentes? (respuesta: filtrado automático por rol y público objetivo en la consulta de bandeja); ¿cómo prevenir que se programen comunicados en el pasado o con menos de treinta minutos de anticipación? (respuesta: validación de fecha programada que calcula diferencia con hora actual y rechaza la operación si no cumple ventana mínima).

Apliqué pensamiento crítico al evaluar la efectividad de controles solo documentales versus controles técnicos automatizados. No bastaba documentar que "los docentes solo pueden crear comunicados académicos y de eventos"; implementé la validación tanto en el servicio de verificación de permisos (que consulta tipo de permiso en base de datos) como en el controlador de creación (que verifica el tipo solicitado contra los tipos permitidos para el rol). Esta doble validación previene errores humanos y intentos de bypass, alineándose con las buenas prácticas del control A.14.1 (Requisitos de seguridad en el desarrollo) de la ISO/IEC 27001.

Desarrollé capacidad para cuestionar supuestos y proponer mejoras realistas. Al detectar que la sanitización básica con expresiones regulares podría no cubrir todos los vectores de ataque XSS, documenté la limitación y propuse ampliar las pruebas de seguridad con un validador robusto externo (DOMPurify). Al notar que la segmentación manual por grado podría generar errores de destinatarios, propuse un preview calculado de destinatarios antes de publicar para confirmación explícita del usuario.

**CG4 – Habilidad para la comunicación oral y escrita**

Elaboré documentación técnica exhaustiva con tres niveles de audiencia: desarrolladores (especificación detallada de endpoints con ejemplos de código de solicitud/respuesta), usuarios finales (mensajes de error y confirmación en lenguaje claro orientado a acción), y auditores o supervisores (trazabilidad de controles de seguridad a normativas aplicables). Adapté el lenguaje técnico según el contexto: usé terminología precisa en documentación interna (middleware, JSONB, sanitización, limiters) y lenguaje natural en mensajes a usuarios ("No tienes permisos para crear comunicados de tipo urgente. Contacta al director para solicitar este permiso").

Redacté ejemplos de entrada y salida para cada endpoint con formato JSON estructurado y comentarios explicativos, facilitando la integración por parte del equipo de frontend. Documenté reglas de negocio en lenguaje declarativo ("Docentes solo pueden crear comunicados académicos y de eventos", "La fecha programada debe ser al menos treinta minutos en el futuro") que traducen restricciones técnicas en políticas comprensibles para stakeholders no técnicos.

Aprendí que la comunicación efectiva en proyectos de TI requiere no solo claridad lingüística, sino también estructura y trazabilidad. Organicé la documentación de API en secciones funcionales con numeración consistente, vinculé cada endpoint a la historia de usuario que lo requiere, y mantuve formatos de respuesta estandarizados que facilitan el consumo programático. Esta disciplina en la comunicación escrita reduce ambigüedades, acelera la integración y previene malentendidos costosos.

**Competencias Específicas**

**CT1.1 – Aplicación de metodologías, estándares y métricas de calidad**

Apliqué un enfoque de desarrollo incremental estructurado en refinación de requisitos, diseño de contratos, implementación modular y validación continua, coherente con los procesos de desarrollo de software de la norma ISO/IEC 12207. Seguí el ciclo Análisis → Diseño → Construcción → Verificación para cada funcionalidad del módulo, manteniendo trazabilidad bidireccional entre historias de usuario, endpoints de API, funciones de controlador y servicio, y entidades de base de datos.

**Cómo lo hice:** estructuré el desarrollo en fases secuenciales con entregables claros: primero refiné historias de usuario definiendo criterios de aceptación verificables (qué condiciones debe cumplir cada funcionalidad para considerarse completa); luego diseñé los contratos de API especificando formatos de entrada, salida y reglas de negocio estandarizadas; después implementé el código en capas independientes (rutas, controladores, servicios) aplicando separación de responsabilidades; finalmente validé cada flujo con datos de prueba generados automáticamente que simulan escenarios reales de uso.

Definí métricas de calidad objetivas para validar el módulo: cobertura de validación de entrada en cien por ciento de endpoints (cada endpoint valida formato, longitud y contenido de parámetros antes de procesarlos), cobertura de autorización en cien por ciento de endpoints (cada endpoint verifica rol del usuario y permisos específicos), tasa de sanitización en cien por ciento de contenido HTML (todo contenido pasa por eliminación de elementos peligrosos antes de almacenarse), y consistencia de formato de respuesta en todos los endpoints (estructura unificada con indicador de éxito, datos o código de error).

**Qué usé:** metodología ágil de desarrollo incremental por historias de usuario; patrón de arquitectura en tres capas (presentación, negocio, datos) para separación de responsabilidades; especificación de API REST con formato OpenAPI informal que documenta contratos verificables; versionamiento de código en GitHub con commits descriptivos por funcionalidad; validación manual iterativa con datos de prueba generados.

**Aprendizaje:** comprendí que la aplicación de metodologías no es seguir mecánicamente un proceso documentado, sino adaptar principios de organización, trazabilidad y calidad al contexto específico del proyecto. La trazabilidad Historia de Usuario → Endpoint → Código → Validación no es burocracia, sino mecanismo de control que asegura que cada requisito se implementa completamente y que cada línea de código responde a una necesidad verificable. Esta disciplina reduce defectos, facilita el mantenimiento y permite demostrar cumplimiento ante auditorías.

**CT2.2 – Construcción y gestión de repositorios de datos**

Diseñé y gestioné la estructura de datos del módulo de comunicados aplicando principios de normalización, integridad referencial y eficiencia de consultas. Modelé la entidad Comunicado con campos que capturan toda la información relevante: identificador único, título y contenido, tipo mediante enumeración (académico, administrativo, evento, urgente, informativo), estado mediante enumeración (borrador, publicado, programado, desactivado, cancelado), segmentación de destinatarios mediante cuatro arrays (público objetivo, niveles, grados, cursos), fechas de trazabilidad (creación, publicación, programación, edición), indicador de edición booleano, relación al autor mediante identificador de usuario, y año académico para particionamiento temporal.

Modelé la entidad ComunicadoLectura con clave única compuesta por identificador de comunicado e identificador de usuario, garantizando unicidad de registros y permitiendo consultas eficientes de estados de lectura. Vinculé esta entidad con eliminación en cascada para mantener integridad referencial cuando se elimina un comunicado permanentemente.

**Cómo lo hice:** apliqué normalización para evitar redundancia de datos del autor (almacenando solo el identificador y resolviendo el nombre completo mediante relación); usé arrays de segmentación en tipo de dato JSONB para permitir consultas de pertenencia eficientes sin necesidad de tablas relacionadas adicionales; definí enumeraciones estrictas para tipo, estado y prioridad que garantizan validez de datos y facilitan consultas categóricas; configuré índices en campos de filtrado frecuente (estado, fecha_publicación, autor_id) para optimizar performance de consultas de bandeja.

Implementé consultas dinámicas que construyen filtros según el rol del usuario autenticado: para padres, cruzar arrays de segmentación del comunicado con niveles/grados de estudiantes vinculados mediante relaciones familiares; para docentes, filtrar por autoría propia o público objetivo institucional; para director, sin filtros adicionales. Estas consultas optimizan el acceso sin comprometer seguridad.

**Qué usé:** Prisma ORM como herramienta de modelado de esquemas con tipos estrictos y generación automática de cliente de consultas; PostgreSQL como motor de base de datos con soporte nativo para JSONB y operadores de pertenencia en arrays; migraciones controladas para evolución segura del esquema; restricciones de unicidad y claves foráneas para integridad referencial; consultas con inclusión de relaciones (include) para reducir número de llamadas a base de datos.

**Aprendizaje:** confirmé que la calidad de la estructura de datos impacta directamente en la simplicidad y mantenibilidad de la lógica de negocio. Modelar la segmentación como arrays simplificó las consultas de visibilidad (un solo filtro de pertenencia en lugar de múltiples joins), pero requirió validaciones más cuidadosas en la creación (verificar que los valores de los arrays corresponden a niveles/grados reales). Este balance entre normalización y optimización requiere evaluar patrones de acceso y priorizar según criticidad: en este caso, prioricé consultas rápidas de lectura (más frecuentes) sobre simplicidad en escritura (menos frecuentes).

**CT2.3 – Fundamentos de gestión de calidad y seguridad en sistemas**

Implementé controles de seguridad en cuatro niveles de protección: autenticación obligatoria mediante tokens de sesión que verifican identidad del usuario, autorización por rol que restringe acceso a endpoints según el rol declarado en el token (padre, docente, director), validación de permisos específicos que verifica en base de datos si el docente tiene permiso explícito para crear comunicados, y sanitización de contenido que elimina elementos peligrosos del HTML antes de almacenarlo y mostrarlo.

Apliqué el principio de Seguridad del artículo 9 de la Ley 29733 mediante medidas técnicas que protegen la confidencialidad (solo usuarios autorizados ven comunicados según segmentación), integridad (contenido sanitizado no puede contener código malicioso) y disponibilidad (limitadores de tasa previenen saturación del servicio). Implementé además el control A.12.6.1 (Gestión de vulnerabilidades técnicas) de la ISO/IEC 27001 al validar que todo contenido HTML pase por sanitización antes de persistirse, eliminando vector de ataque de inyección de código.

**Cómo lo hice:** configuré autenticación JWT obligatoria en todas las rutas del módulo mediante middleware que valida firma, vigencia y estructura del token antes de permitir acceso; implementé autorización granular por endpoint con middleware que verifica lista de roles permitidos (ej.: solo director puede eliminar comunicados); codifiqué validaciones de permisos en servicios que consultan tabla PermisoDocente para verificar que el docente tiene permiso activo de tipo comunicados antes de permitir creación; desarrollé función de sanitización que detecta y elimina etiquetas peligrosas (script, iframe, object, embed) mediante patrones de expresiones regulares, preservando etiquetas seguras de formato; configuré limitadores de tasa diferenciados que bloquean temporalmente direcciones IP que exceden el umbral permitido de solicitudes por minuto.

Implementé trazabilidad de acciones administrativas sensibles registrando fecha de desactivación y eliminación para facilitar auditorías. Diseñé la eliminación de comunicados como operación de dos pasos: primero eliminación de registros relacionados en tabla de lecturas (para mantener integridad referencial), luego eliminación del comunicado propiamente, ambas dentro de transacción para garantizar atomicidad.

**Qué usé:** middleware de autenticación JWT integrado en capa de ruteo; middleware de autorización por rol que rechaza solicitudes de usuarios no autorizados con código HTTP 403; consultas de verificación de permisos en tabla PermisoDocente con filtros por tipo de permiso, estado activo y año académico; función de sanitización con patrones de expresiones regulares que detectan elementos peligrosos; limitadores de tasa configurados con umbrales diferenciados por criticidad (escritura más restrictiva que lectura); transacciones de base de datos para operaciones multi-paso que requieren atomicidad.

**Aprendizaje:** aprendí que la seguridad efectiva no es resultado de un solo control robusto, sino de capas superpuestas donde cada una mitiga riesgos residuales de la anterior. Si un atacante bypasa sanitización en el cliente, el servidor la aplica nuevamente. Si manipula su token para cambiar el rol, la autorización por endpoint lo detiene. Si bypass la validación de rol, la verificación de permisos específicos en base de datos lo bloquea nuevamente. Esta defensa en profundidad, alineada con el control A.14.2.5 (Principios de ingeniería y diseño seguro de sistemas) de la ISO/IEC 27001, me permitió entender que la seguridad es propiedad emergente de múltiples decisiones técnicas correctas, no de una sola tecnología milagrosa.

**CT4.3 – Identificación y análisis de problemas de investigación**

Identifiqué como problema central la ausencia de trazabilidad y segmentación en la comunicación institucional de la I.E.P. Las Orquídeas. Los directivos y docentes comunicaban avisos mediante canales informales (mensajes grupales, llamadas telefónicas, avisos en papel) sin registro de confirmación de lectura, sin segmentación automática por destinatarios relevantes, y sin organización histórica que permita consultar comunicaciones pasadas.

**Cómo lo hice:** formulé el problema como requisitos medibles: "habilitar publicación de comunicados con segmentación automática por nivel/grado/curso", "registrar confirmación de lectura de cada usuario", "permitir búsqueda y filtrado de comunicados históricos por tipo y fecha", "garantizar que cada usuario vea solo comunicados que le conciernen". Traduje cada requisito en funcionalidades verificables: bandeja con filtros de tipo y fecha, marcado automático de lectura con timestamp, búsqueda por título y contenido, segmentación automática mediante consultas que cruzan arrays de destinatarios con vínculos institucionales del usuario.

Analicé las causas raíz del problema: falta de herramienta centralizada (resuelto con módulo dedicado de comunicados), ausencia de control de permisos (resuelto con tabla PermisoDocente y validaciones), incapacidad de segmentar mensajes (resuelto con arrays de destinatarios y filtrado automático), y falta de métricas de lectura (resuelto con tabla ComunicadoLectura y contadores).

**Qué usé:** historias de usuario para capturar necesidades desde la perspectiva de cada rol; criterios de aceptación para definir qué significa "comunicado exitosamente publicado y leído"; especificación

de API para validar contratos de servicio; datos de prueba simulados para verificar que las soluciones resuelven los casos identificados; documentación técnica que vincula cada funcionalidad al problema que resuelve.

**Aprendizaje:** desarrollé capacidad para fundamentar problemas con evidencia verificable (falta de trazabilidad medida por ausencia de registros históricos, falta de segmentación medida por quejas de sobrecarga informativa) y para proponer soluciones preventivas con controles técnicos (validación previa de segmentación, sanitización automática, publicación programada con ventana mínima). Comprendí que identificar problemas es insuficiente; debo traducirlos en requisitos accionables y validar que las soluciones realmente resuelven las causas raíz mediante verificación empírica con datos reales o simulados.

**CT4.4 – Gestión de proyectos de TI**

Planifiqué el desarrollo del módulo en cuatro fases incrementales con entregables progresivos: fase de análisis y refinación de historias de usuario con criterios de aceptación (entregable: especificación funcional); fase de diseño de arquitectura y contratos de API (entregable: documentación de veinticuatro endpoints); fase de implementación backend en capas (entregable: código fuente con rutas, controladores, servicios y modelo de datos); fase de validación funcional y generación de datos de prueba (entregable: script de semillas y verificación manual de flujos críticos).

Prioricé funcionalidades según dependencias críticas y valor entregado: primero bandeja y lectura (habilitadores fundamentales), luego creación y publicación (valor directo para usuarios), después gestión de borradores y programación (optimización de flujo de trabajo), y finalmente operaciones administrativas (control y supervisión). Esta priorización aseguró que el módulo entregue valor incremental en cada fase, permitiendo uso temprano de funcionalidades básicas mientras se desarrollan las avanzadas.

**Cómo lo hice:** organicé el trabajo en entregables verificables (documentación técnica firmada por criterios de aceptación, backend funcional con endpoints accesibles, semillas de datos que permiten validación manual); gestioné el versionamiento de código en GitHub con commits atómicos que reflejan una funcionalidad específica (ej.: "Implementar validación de permisos docente para creación de comunicados"); apliqué separación de responsabilidades en tres capas independientes que pueden desarrollarse en paralelo (rutas con seguridad, controladores con validación de parámetros, servicios con reglas de negocio); documenté el alcance comprometido (qué funcionalidades entrega este módulo) y el alcance pendiente (qué funcionalidades quedan para siguientes iteraciones) con criterios claros de priorización.

Implementé controles de aseguramiento de calidad: validaciones de entrada en cien por ciento de endpoints, sanitización obligatoria de contenido HTML, limitadores de tasa para prevenir abuso, y

trazabilidad de acciones administrativas mediante timestamps y estados. Revisé cada función de controlador y servicio para verificar manejo de errores con códigos HTTP apropiados y mensajes descriptivos.

**Qué usé:** cronograma del proyecto general para alinear el módulo con las fases de desarrollo; GitHub como repositorio central de código, documentación y scripts; metodología de historias de usuario con criterios de aceptación como método de captura de requisitos; patrón de tres capas (MVC adaptado) para organización modular; testing iterativo manual con datos de semilla para verificar funcionalidades conforme se desarrollan; documentación técnica progresiva que se actualiza en paralelo con la implementación.

**Aprendizaje:** comprendí que la gestión efectiva de proyectos de TI no consiste únicamente en completar tareas dentro de plazos, sino en entregar valor incremental con calidad verificable y trazabilidad completa. Priorizar funcionalidades según dependencias (autenticación antes que comunicados, permisos docentes antes que creación) reduce bloqueos y riesgos de retrabajos. Documentar el alcance comprometido y el alcance pendiente con claridad previene expectativas incorrectas y facilita la planificación de iteraciones futuras. Mantener trazabilidad bidireccional entre requisitos y artefactos no solo mejora la calidad del producto final, sino que fortalece la capacidad de respuesta ante cambios de alcance o correcciones de defectos.

**Conclusiones**

El desarrollo del módulo de Comunicados habilitó un sistema estructurado, seguro y trazable de comunicación institucional para la I.E.P. Las Orquídeas. Se implementaron controles de acceso diferenciados por rol, segmentación automática de destinatarios según relaciones institucionales, sanitización obligatoria de contenido HTML, validación de permisos de publicación para docentes, publicación inmediata o programada con ventana de seguridad, registro automático de lectura para métricas de adopción, y operaciones administrativas de desactivación y eliminación para el director.

La arquitectura en tres capas (rutas con middlewares de seguridad, controladores con validación de parámetros, servicios con reglas de negocio) y la trazabilidad bidireccional entre historias de usuario y artefactos técnicos facilitan el mantenimiento, la extensión y la auditoría del módulo. La aplicación de principios normativos (Ley 29733 artículos 8 y 9, Ley 28044) y buenas prácticas de seguridad (ISO/IEC 27001 controles A.9, A.12, A.14, A.18) fortalece el cumplimiento regulatorio y la confianza de la comunidad educativa.

Este módulo contribuye directamente a mejorar la comunicación efectiva entre la institución y las familias, reduciendo la dispersión informativa, mejorando la oportunidad de los avisos mediante publicación programada, y facilitando el seguimiento mediante registro de lectura. La trazabilidad de

comunicados, lecturas y acciones administrativas habilita métricas de adopción y análisis de efectividad de comunicación que informarán mejoras futuras.

En conclusión, este desarrollo consolidó mis competencias en gestión de repositorios de datos (diseño de estructuras eficientes, consultas optimizadas, integridad referencial), seguridad de la información (controles en múltiples capas, sanitización, principio de mínimo privilegio), gestión de proyectos de TI (priorización por valor y dependencias, entregables incrementales, trazabilidad) y comunicación técnica (documentación multinivel, mensajes claros a usuarios). Reforzó además mi compromiso ético en el tratamiento responsable de datos de menores y familias en contextos educativos.