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

Facultad de Ingeniería de Sistemas e Informática

"ENTREGABLE – MÓDULO DE ENCUESTAS"

PRÁCTICAS PRE PROFESIONALES

PROFESOR

William Martín, Enriquez Maguiña

PRESENTADO POR

Verde Huaynaccero, Erick Edison

Lima-Perú

2025

Informe del Módulo de Encuestas – Sistema de Recopilación de Feedback Institucional

Objetivo del Entregable

Implementar un sistema integral de encuestas institucionales que permita a directivos y docentes autorizados diseñar, publicar y analizar encuestas de feedback dirigidas a la comunidad educativa de la I.E.P. Las Orquídeas. El módulo debe habilitar la creación de encuestas con cinco tipos de preguntas (texto corto, texto largo, opción única, opción múltiple y escala de satisfacción uno a cinco), garantizar segmentación automática de destinatarios por rol y vínculos institucionales, registrar respuestas de forma inmutable con trazabilidad completa, y proporcionar análisis visual de resultados para toma de decisiones informada.

El propósito fue resolver la ausencia de mecanismos estructurados de recopilación de feedback de familias y docentes, estableciendo un canal formal que permita medir satisfacción, identificar necesidades y evaluar servicios educativos mediante encuestas segmentadas, anónimas o nominativas según configuración, con fechas de vencimiento automáticas y registro de participación para análisis de adopción.

Este sistema incorpora controles de acceso diferenciados por rol, validación de permisos de creación para docentes, constructor dinámico de preguntas con validaciones específicas por tipo,

segmentación automática de destinatarios, registro inmutable de respuestas con timestamp y dirección IP, validación de completitud de preguntas obligatorias, y funcionalidades de gestión administrativa para cerrar, extender, duplicar o eliminar encuestas según su estado.

La implementación respeta los principios de Proporcionalidad (artículo ocho) y Seguridad (artículo nueve) de la Ley número veintinueve mil setecientos treinta y tres – Ley de Protección de Datos Personales, asegurando que solo se recopilen datos necesarios para el feedback solicitado y con validaciones técnicas de protección. Además, se alinea con los lineamientos de participación de la comunidad educativa establecidos en la Ley veintiocho mil cuarenta y cuatro – Ley General de Educación, y aplica buenas prácticas de los controles de seguridad de la información de la ISO/IEC veintisiete mil uno (específicamente A.nueve Control de Acceso, A.doce Seguridad de las Operaciones, A.catorce Seguridad en el Desarrollo) y de trazabilidad de procesos de la ISO/IEC doce mil doscientos siete (gestión de requisitos, documentación y verificación).

Metodología Aplicada

El desarrollo del módulo siguió un proceso estructurado de cuatro etapas con validación continua y trazabilidad de requisitos a artefactos:

1. **Análisis y refinación de requisitos funcionales:**
   Estudié las necesidades de recopilación de feedback de la institución educativa y las traduje en seis historias de usuario con criterios de aceptación verificables: HU-ENC-00 para el panel de encuestas con tres pestañas de estado (activas, respondidas, vencidas) y filtros avanzados; HU-ENC-01 para el formulario dinámico de respuesta con cinco tipos de preguntas y validaciones específicas por tipo; HU-ENC-02 para la visualización de respuestas propias en modo solo lectura; HU-ENC-03 para el constructor de encuestas con wizard de tres pasos (información básica, preguntas, audiencia); HU-ENC-04 para la gestión de encuestas propias con operaciones administrativas; y HU-ENC-05 para el análisis de resultados con visualizaciones. Documenté cada historia con condiciones funcionales detalladas, validaciones de negocio numeradas, mensajes de éxito y error, y flujos de estados esperados.
2. **Diseño de arquitectura de datos y API:**
   Diseñé un modelo de datos normalizado con cinco entidades relacionadas: Encuesta con información general y configuración, PreguntaEncuesta con tipo y orden, OpcionPregunta para preguntas de selección, RespuestaEncuesta como contenedor principal por usuario, y RespuestaPregunta con valores específicos según tipo de pregunta. Especifiqué nueve endpoints REST organizados en cuatro secciones funcionales (panel, responder, ver respuestas propias, crear), documentando para cada endpoint el formato de entrada con validaciones Zod, estructura de respuesta con indicador de éxito, códigos de error HTTP y reglas de negocio aplicables.

Diseñé validaciones específicas por tipo de pregunta: longitud mínima y máxima para texto, mínimo dos opciones para selección, valor entre uno y cinco para escala.

3. **Implementación modular backend con validaciones defensivas:**
   Implementé el módulo en cuatro capas independientes aplicando separación de responsabilidades: capa de ruteo que define nueve endpoints con middlewares de autenticación obligatoria; capa de controladores con cinco funciones que validan parámetros y formatean respuestas; capa de middlewares con tres validadores especializados que verifican permisos de creación, permisos de gestión y estructura de respuestas por tipo de pregunta; y capa de servicios con reglas de negocio, consultas relacionadas a múltiples tablas, validaciones con esquemas Zod y cálculos de estadísticas. Implementé validación doble: middleware que verifica permisos de creación consultando tabla PermisoDocente, y servicio que valida estructura de cada pregunta según su tipo con mensajes descriptivos de error.
4. **Desarrollo de suite de pruebas automatizadas:**
   Desarrollé doce casos de prueba organizados en cinco grupos funcionales que validan: listado de encuestas con segmentación por rol, contador de pendientes excluyendo respondidas, creación de encuesta con validación de permisos (director puede, docente con permisos puede, padre no puede), validación de acceso según segmentación, obtención de formulario con preguntas y opciones ordenadas, envío de respuestas con validación de completitud, prevención de respuestas duplicadas con código de conflicto, y consulta de respuestas propias con error si no respondió previamente. Configuré limpieza automática de datos de prueba para evitar contaminación de base de datos.

Herramientas Utilizadas

- **Node.js con Express:** Plataforma backend para construcción de APIs REST con organización modular en cuatro capas (rutas, controladores, middlewares, servicios) y gestión de autenticación mediante tokens de sesión firmados.
- **Prisma ORM:** Herramienta de modelado de datos con generación automática de cliente tipado, definición de esquemas con enumeraciones estrictas, relaciones entre cinco entidades (Encuesta, PreguntaEncuesta, OpcionPregunta, RespuestaEncuesta, RespuestaPregunta), y soporte para migraciones controladas que evolucionan el esquema sin pérdida de datos.
- **Zod:** Biblioteca de validación de esquemas que facilita la definición de reglas de negocio verificables en tiempo de ejecución. Utilicé tres esquemas principales: crearEncuestaSchema con validación de estructura de preguntas por tipo, responderEncuestaSchema con validación de valores según tipo de pregunta, y obtenerEncuestasSchema con validación de parámetros de paginación y filtrado.
- **PostgreSQL (Neon):** Base de datos relacional gestionada en nube que almacena encuestas con preguntas en estructura normalizada, respuestas con valores específicos por tipo de pregunta,

usuarios con permisos, estudiantes y relaciones familiares. Aproveché restricciones únicas para prevenir respuestas duplicadas por usuario en la misma encuesta.

- **Vitest con Supertest:** Framework de testing para pruebas automatizadas de endpoints REST con aserciones sobre códigos HTTP, estructura de respuestas y validaciones de negocio. Configuré suite con doce casos que validan permisos, segmentación, validaciones y flujos críticos.
- **GitHub:** Sistema de control de versiones para gestionar cambios en código fuente, documentación técnica, esquemas de base de datos y scripts de prueba, manteniendo historial trazable de modificaciones por módulo.

Actividades Realizadas

- Refiné seis historias de usuario para el módulo de encuestas con criterios de aceptación detallados: panel con tres pestañas de estado y segmentación automática, formulario de respuesta con cinco tipos de preguntas y validaciones específicas, visualización de respuestas propias en modo solo lectura, constructor con wizard de tres pasos y validación de estructura, gestión administrativa con operaciones condicionales por estado, y análisis de resultados con visualizaciones por tipo de pregunta.
- Diseñé y documenté nueve endpoints REST agrupados por funcionalidad: obtener lista con paginación y filtros, contador de pendientes, búsqueda por término, polling de actualizaciones, validación de acceso, formulario con estructura de preguntas, envío de respuestas con validación de completitud, consulta de respuestas propias y creación de encuesta con estructura de preguntas.
- Modelé cinco entidades relacionadas en Prisma: Encuesta con campos de configuración (título, descripción, fechas, flags booleanos, relación al autor), PreguntaEncuesta con tipo mediante enumeración y orden, OpcionPregunta con texto y orden para preguntas de selección, RespuestaEncuesta con restricción única por encuesta y usuario, y RespuestaPregunta con campos específicos por tipo (valor_texto, valor_opcion_id, valor_opciones array, valor_escala).
- Implementé cinco controladores que gestionan las operaciones principales del módulo: crear encuesta validando permisos según rol, obtener lista aplicando segmentación automática por rol del usuario, obtener contador excluyendo encuestas ya respondidas, buscar por término en título y descripción, verificar actualizaciones para polling en tiempo real, validar acceso según segmentación de la encuesta, obtener formulario con preguntas ordenadas y validaciones por tipo, enviar respuestas validando completitud de obligatorias y formato por tipo, y obtener respuestas propias verificando que el usuario haya respondido previamente.
- Codifiqué lógica de negocio en servicio con validaciones defensivas: verificar permisos consultando tabla PermisoDocente con filtros por tipo de permiso, estado activo y año académico; aplicar segmentación automática consultando relaciones familiares para padres (filtrar por niveles y grados de hijos) o asignaciones docente-curso para docentes (filtrar por

grados asignados); validar estructura de cada pregunta según su tipo con esquemas Zod que especifican campos obligatorios, longitudes permitidas y valores válidos; crear encuesta con preguntas y opciones en transacción atómica; registrar respuestas con validación de unicidad, completitud de obligatorias y formato por tipo; y calcular estadísticas de participación, tiempo promedio de respuesta y completitud.

- Desarrollé tres middlewares especializados de validación: puedeCrearEncuestas que verifica que el usuario sea director o docente con permisos activos de tipo encuestas; puedeGestionarEncuesta que verifica que el usuario sea director sin restricciones o docente autor de la encuesta específica; y validarRespuestaEncuesta que verifica que la encuesta esté activa y no vencida, que el usuario no haya respondido previamente, que todas las preguntas obligatorias tengan respuesta válida, y que cada valor cumpla las validaciones específicas de su tipo.
- Construí suite de testing con doce casos de prueba organizados en cinco grupos funcionales: autenticación y autorización (tres roles diferentes intentan obtener encuestas), filtrado por estado y tipo, búsqueda con términos, contador de pendientes, creación validando permisos por rol, validación de acceso con segmentación, obtención de formulario con estructura completa, envío de respuestas con IDs reales obtenidos del formulario, prevención de respuestas duplicadas con código de conflicto cuatrocientos nueve, y consulta de respuestas propias con error si no respondió.
- Documenté especificación de API con ejemplos de uso mediante cURL, formatos de request y response en JSON, códigos de error estandarizados con mensajes descriptivos, y reglas de negocio numeradas que facilitan la trazabilidad entre requisitos y validaciones en código.
- Verifiqué manualmente los flujos críticos del módulo: listado de encuestas con segmentación automática (padre solo ve encuestas de grados de sus hijos, docente ve institucionales más propias, director ve todas), contador de pendientes excluyendo respondidas y vencidas, validación de acceso retornando motivo descriptivo, formulario con preguntas ordenadas y validaciones por tipo, envío de respuestas con validación de completitud de obligatorias, prevención de respuestas duplicadas mediante restricción única en base de datos, consulta de respuestas propias en modo solo lectura, y creación de encuesta con validación de permisos y estructura de preguntas.

Evidencias

- **Especificación técnica del módulo:** Documentación de seis historias de usuario refinadas con criterios de aceptación verificables y ciento setenta reglas de negocio numeradas; especificación de nueve endpoints REST con ejemplos de entrada, salida, validaciones y códigos de error; modelo de datos con cinco entidades relacionadas y enumeraciones estrictas de tres tipos (EstadoEncuesta, TipoPregunta, PermisoTipo).

- **Implementación backend verificable:** Archivo de rutas con definición de nueve endpoints y middleware de autenticación obligatoria, sin limitadores de tasa por estar en fase de desarrollo; cinco controladores que validan parámetros y formatean respuestas con estructura consistente; tres middlewares especializados que verifican permisos de creación, gestión y validez de respuesta; servicio completo con validaciones mediante esquemas Zod para tres operaciones críticas (crear encuesta, responder encuesta, obtener encuestas con filtros).
- **Estructura de datos normalizada:** Modelo de entidad Encuesta con enumeración de estados (borrador, activa, cerrada, vencida), flags booleanos para configuración (permite respuesta múltiple, es anónima, mostrar resultados), fechas de trazabilidad (creación, inicio, vencimiento, cierre), relación al usuario autor y año académico; entidad PreguntaEncuesta con enumeración de cinco tipos, campo texto, flag obligatoria y orden para renderizado; entidad OpcionPregunta con texto y orden para opciones de selección; entidad RespuestaEncuesta con restricción única compuesta por encuesta y usuario que previene duplicados; entidad RespuestaPregunta con campos específicos por tipo de pregunta que almacenan valores en formato adecuado.
- **Suite de pruebas automatizadas:** Doce casos de prueba organizados en cinco grupos que validan segmentación automática por rol, autorización de creación, búsqueda y filtros, contador de pendientes, validación de acceso, obtención de formulario, envío de respuestas con IDs reales, prevención de duplicados y consulta de respuestas propias. Suite con configuración de setup que crea usuarios de prueba (director, docente con permisos, padre) con números de documento únicos basados en timestamp para evitar conflictos, y teardown que limpia datos de prueba en orden inverso de dependencias.
- **Resultados de verificación funcional y de conformidad:**
  Validé que el listado de encuestas aplica segmentación automática según rol: ejecuté consulta como padre y verifiqué que solo retorna encuestas visibles para grados de hijos matriculados (consulta cruzada con tabla relaciones familiares); ejecuté consulta como docente y confirmé visibilidad de encuestas institucionales más las creadas por él (filtro por autor o público objetivo); ejecuté como director y verifiqué acceso total sin restricciones.
  Verifiqué que el contador de pendientes excluye encuestas respondidas: confirmé mediante consulta que existe restricción única en tabla RespuestaEncuesta por combinación de encuesta y usuario, y que el contador filtra encuestas cuyo identificador no esté en la lista de encuestas respondidas por el usuario autenticado.
  Confirmé que la validación de acceso retorna información descriptiva en lugar de código de error cuatrocientos tres: ejecuté validación de acceso a encuesta fuera de segmentación y verifiqué respuesta con success verdadero, tiene_acceso falso, motivo descriptivo ("La encuesta no está dirigida a tu rol o nivel"), puede_responder falso y segmentacion_valida falso.
  Validé que el formulario de respuesta incluye preguntas ordenadas con estructura completa: obtuve formulario de encuesta con cinco preguntas de tipos variados y confirmé que cada pregunta incluye identificador único, tipo mediante enumeración, texto, flag obligatoria, orden

numérico, opciones ordenadas (si aplica), etiquetas personalizables para escala (si aplica), y reglas de validación específicas por tipo (mínimo y máximo de caracteres para texto, mínimo y máximo de opciones para selección, rango de uno a cinco para escala).
Comprobé que el envío de respuestas valida completitud de preguntas obligatorias: envié respuesta con pregunta obligatoria faltante y recibí código cuatrocientos con mensaje descriptivo y lista de identificadores de preguntas faltantes; envié respuesta completa y recibí código doscientos uno con confirmación, identificador de respuesta, timestamp, tiempo de respuesta y estadísticas actualizadas.
Verifiqué prevención de respuestas duplicadas: respondí encuesta y recibí confirmación; intenté responder nuevamente con mismos datos y recibí código cuatrocientos nueve (conflicto) con mensaje "Ya has respondido esta encuesta anteriormente"; confirmé que el registro de respuesta tiene restricción única en base de datos sobre la combinación de encuesta_id y usuario_id.
Validé que la consulta de respuestas propias verifica existencia previa: consulté respuestas de encuesta no respondida y recibí código cuatrocientos con error RESPONSE_NOT_FOUND y mensaje "Aún no has respondido esta encuesta"; consulté respuestas de encuesta ya respondida y recibí código doscientos con encuesta, respuesta con metadatos (fecha, tiempo, completitud) y respuestas_detalle con valores formateados según tipo de pregunta.
Comprobé que la creación de encuesta valida permisos según rol: envié solicitud de creación como padre y recibí código cuatrocientos tres con error FORBIDDEN; envié como docente sin permisos y recibí cuatrocientos tres con mensaje descriptivo; envié como docente con permiso activo de tipo encuestas y recibí doscientos uno con encuesta creada incluyendo identificadores de preguntas y opciones generados; envié como director sin validación adicional y recibí doscientos uno confirmando privilegio total.

Dificultades Encontradas

- **Modelado de estructura flexible de preguntas:** Diseñar un modelo de datos que soporte cinco tipos diferentes de preguntas con requisitos variados (texto con longitudes, opciones con mínimos y máximos, escala con rango fijo) implicó decidir entre almacenar preguntas en JSON desnormalizado versus estructura normalizada con tablas relacionadas. Opté por normalización creando tres tablas (PreguntaEncuesta, OpcionPregunta, RespuestaPregunta) para facilitar consultas, validaciones y análisis, aceptando mayor complejidad en las inserciones. Implementé validaciones específicas por tipo en middleware y servicio que verifican estructura correcta antes de persistir.
- **Validación de completitud de preguntas obligatorias:** Asegurar que el usuario responda todas las preguntas obligatorias antes de enviar requirió validar tanto en frontend (para experiencia de usuario) como en backend (para seguridad real). Implementé verificación mediante conjuntos de identificadores: obtener conjunto de preguntas obligatorias de la encuesta, obtener conjunto de

preguntas respondidas del payload, y verificar que no haya diferencias. Retorno mensaje de error con lista exacta de identificadores faltantes para facilitar corrección.

- **Prevención de respuestas duplicadas en operaciones concurrentes:** Garantizar que un usuario no pueda responder la misma encuesta dos veces, incluso si envía solicitudes simultáneas o intenta reintentar tras error temporal, requirió usar restricción única en base de datos sobre la combinación de encuesta_id y usuario_id. Implementé manejo de error de violación de restricción única retornando código cuatrocientos nueve (conflicto) con mensaje descriptivo en lugar de error genérico de base de datos.
- **Cálculo de estadísticas de participación sin conocer destinatarios exactos:** Calcular el porcentaje de participación (respuestas dividido destinatarios por cien) requiere conocer el número exacto de destinatarios según la segmentación original. Como la segmentación aún no está completamente implementada en el módulo, implementé cálculo temporal con valor fijo de cien destinatarios y documenté la limitación para corrección futura cuando se implemente el sistema de segmentación completo similar al módulo de Comunicados.
- **Validación específica por tipo de pregunta con mensajes descriptivos:** Implementar validaciones diferentes para cada tipo de pregunta (longitudes para texto, opciones válidas para selección, rango para escala) con mensajes de error contextualizados que mencionen el texto de la pregunta específica requirió iterar sobre el array de respuestas, buscar la pregunta correspondiente en la estructura de la encuesta, validar según el tipo, y construir mensaje de error que incluya el texto completo de la pregunta para claridad. Esta validación detallada facilita la corrección de errores por parte del usuario.

Lecciones Aprendidas

- Reforcé la importancia de la inmutabilidad en datos críticos de auditoría. Aprendí que las respuestas de encuestas deben ser inmutables (sin opción de edición posterior) para garantizar integridad de análisis y prevenir manipulación de feedback. Implementé esta inmutabilidad mediante ausencia de endpoint de edición de respuestas y mensaje claro en interfaz: "No podrás modificar tus respuestas después de enviar". Esta decisión de diseño protege la confiabilidad de los resultados y se alinea con el principio de Integridad del artículo nueve de la Ley veintinueve mil setecientos treinta y tres.
- Consolidé mi comprensión de validación defensiva en múltiples capas. Implementé validación en cuatro niveles: validación en frontend con mensajes en tiempo real (longitud de texto, selección de opciones), validación en middleware que verifica estructura general y permisos, validación en servicio con esquemas Zod que verifican tipos y formatos, y validación en base de datos con restricciones únicas y relaciones de integridad referencial. Esta defensa en profundidad, alineada con el control A.catorce.dos.cinco (Principios de diseño seguro) de la ISO/IEC veintisiete mil uno, me enseñó que cada capa reduce riesgos residuales de la anterior.

- Aprendí el valor de la normalización versus desnormalización según patrones de acceso. Inicialmente consideré almacenar preguntas y opciones en campo JSON único de la encuesta para simplicidad, pero opté por normalización con tablas relacionadas al identificar que las consultas frecuentes requieren filtrado por tipo de pregunta, ordenamiento y validación de existencia de opciones. Esta decisión facilitó la validación de respuestas (verificar que opción_id corresponde a opción real de esa pregunta) y el análisis de resultados (contar respuestas por opción), aceptando mayor complejidad en la creación.
- Desarrollé capacidad para diseñar validaciones de negocio que reflejen restricciones institucionales reales. La restricción de que docentes solo pueden crear encuestas si tienen permiso activo de tipo encuestas no es arbitraria: responde al modelo de delegación de responsabilidades de la institución donde el director otorga permisos explícitos a docentes de confianza. Implementar esta restricción técnicamente mediante consulta a tabla PermisoDocente con filtros por tipo, estado y año me permitió aplicar concretamente el principio de Autorización Granular del control A.nueve.cuatro.uno (Restricción de acceso) de la ISO/IEC veintisiete mil uno.
- Comprendí la importancia de trazabilidad completa en recopilación de feedback. Registrar no solo las respuestas sino también timestamp, dirección IP, tiempo de respuesta y estudiante relacionado (para padres) habilita análisis de calidad: identificar respuestas apresuradas (tiempo menor a un minuto), detectar patrones por segmento (nivel o grado), y auditar participación real versus estimada. Esta trazabilidad facilita además la verificación de cumplimiento del principio de Legitimidad del artículo cinco de la Ley veintinueve mil setecientos treinta y tres: poder demostrar que cada dato fue recopilado con el consentimiento implícito del usuario al responder voluntariamente.
- Perfeccioné la habilidad de comunicar restricciones técnicas mediante mensajes orientados a acción. En lugar de mensajes genéricos ("Error de validación"), implementé mensajes específicos que describen el problema y sugieren la solución: "La respuesta a la pregunta 'Comentarios adicionales' debe tener al menos diez caracteres" (describe qué falta) o "Ya has respondido esta encuesta anteriormente" (explica por qué no puede continuar). Esta claridad en la comunicación mejora la experiencia de usuario y reduce solicitudes de soporte.

Competencias Desarrolladas en el Módulo de Encuestas

Competencias Genéricas

CG1 – Valores, compromiso ético y social

Durante el desarrollo del módulo de encuestas institucionales, apliqué rigurosamente el principio de Proporcionalidad establecido en el artículo ocho de la Ley número veintinueve mil setecientos treinta y tres al diseñar la recopilación de feedback. Cada encuesta solicita únicamente la información necesaria para su finalidad declarada (medir satisfacción, evaluar servicios, recopilar sugerencias),

sin solicitar datos personales innecesarios más allá del contexto educativo. Implementé configuración de encuestas anónimas mediante flag booleano es_anonima que, cuando está activo, omite el registro del usuario_id en las respuestas y solo almacena agregados estadísticos, protegiendo la privacidad de respondientes en temas sensibles.

Implementé el principio de Seguridad del artículo nueve mediante controles técnicos que protegen la integridad de las respuestas: inmutabilidad de respuestas enviadas (sin endpoint de edición), validación de completitud de preguntas obligatorias antes de aceptar el envío, sanitización de contenido de texto para prevenir inyección de código, y restricción única en base de datos que previene respuestas duplicadas incluso en operaciones concurrentes. Además, apliqué el control A.nueve.cuatro.tres (Gestión de acceso privilegiado) de la ISO/IEC veintisiete mil uno al restringir la visualización de resultados completos: solo el autor de la encuesta o el director pueden ver respuestas individualizadas, mientras que respondientes comunes solo ven resultados agregados si la configuración mostrar_resultados está activa.

Comprendí que en encuestas dirigidas a familias de menores de edad, la responsabilidad ética requiere transparencia en la finalidad. Cada encuesta declara en su descripción el objetivo y el uso previsto de los resultados, cumpliendo con el principio de Finalidad del artículo nueve de la Ley veintinueve mil setecientos treinta y tres. Implementé además registro de timestamp e IP para auditoría sin identificación directa en encuestas anónimas, equilibrando trazabilidad técnica con protección de privacidad de respondientes.

CG3 – Capacidad de análisis y pensamiento crítico

Ejercité el análisis sistemático al descomponer el problema de recopilación de feedback en requisitos funcionales verificables y restricciones de negocio numeradas. Identifiqué casos problemáticos y diseñé soluciones preventivas: ¿cómo garantizar que un usuario no responda dos veces la misma encuesta? (restricción única en base de datos más validación en servicio antes de insertar); ¿cómo validar que una respuesta de opción única realmente corresponde a una opción válida de esa pregunta? (búsqueda del identificador en el array de opciones de la pregunta específica, no de cualquier pregunta); ¿cómo determinar si una encuesta está vencida sin consultar manualmente? (comparación automática de fecha_vencimiento con fecha actual en cada consulta, más actualización masiva mediante job programado).

Apliqué pensamiento crítico al evaluar diferentes estrategias de modelado de datos. Evalúé tres opciones: almacenar preguntas en JSON desnormalizado (simple pero dificulta consultas y validaciones), estructura hibrida con encuesta y preguntas en JSON pero opciones normalizadas (balance intermedio), o normalización completa con cuatro tablas relacionadas (complejo pero robusto). Seleccioné normalización completa al identificar que los patrones de acceso requieren

consultas por tipo de pregunta, validación de existencia de opciones y análisis agregado de respuestas, justificando la complejidad adicional con beneficios de integridad y extensibilidad.

Desarrollé capacidad para cuestionar supuestos y validar decisiones mediante evidencia. Al diseñar la restricción de que encuestas publicadas no permiten edición de preguntas, cuestioné si limita demasiado la flexibilidad. Analicé el impacto: permitir edición después de que usuarios hayan respondido invalidaría las respuestas anteriores o crearía inconsistencias en el análisis. Concluí que la restricción es necesaria para integridad de datos, pero implementé alternativa: función de duplicar encuesta que copia la estructura como nuevo borrador, permitiendo ajustes sin afectar resultados de la encuesta original.

CG4 – Habilidad para la comunicación oral y escrita

Elaboré documentación técnica exhaustiva con cuatro niveles de audiencia diferenciados: desarrolladores frontend (especificación de endpoints con estructura de request/response, validaciones y códigos de error para consumo programático), desarrolladores backend (documentación de servicios con parámetros, valores de retorno y lógica de reglas de negocio), usuarios finales (mensajes de error y confirmación en lenguaje claro orientado a acción sin terminología técnica), y auditores o revisores académicos (trazabilidad de controles de seguridad y privacidad a normativas aplicables con citas de artículos específicos).

Redacté ciento setenta reglas de negocio numeradas con formato consistente que traducen requisitos institucionales en restricciones técnicas verificables. Ejemplos: "RN-ENC-cuarenta y tres: Verificar que haya al menos una pregunta obligatoria" facilita verificación en código mediante conteo de preguntas con flag obligatoria verdadero; "RN-ENC-cincuenta y dos: Una vez enviada, la respuesta no puede ser modificada ni eliminada (inmutable)" justifica ausencia de endpoint PUT o DELETE para respuestas y fundamenta mensaje al usuario sobre irreversibilidad del envío.

Documenté ejemplos de uso mediante comandos cURL con estructura de request completa, facilitando testing manual y comprensión de contratos de API. Cada ejemplo incluye headers de autenticación, formato JSON del body con todos los campos requeridos, y descripción de la respuesta esperada. Esta documentación práctica reduce ambigüedades y acelera la integración por parte del equipo de frontend.

Aprendí que la comunicación efectiva en proyectos de TI requiere no solo claridad en la explicación sino también consistencia en la organización. Mantuve numeración secuencial de reglas de negocio, estructura uniforme de respuestas API (success, data/error, mensaje), y formato estandarizado de códigos de error (BAD_REQUEST, FORBIDDEN, NOT_FOUND, CONFLICT) que facilitan la referencia cruzada entre documentación, código y pruebas.

Competencias Específicas

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

Apliqué un enfoque de desarrollo incremental con trazabilidad bidireccional entre historias de usuario, endpoints de API, funciones de código y entidades de base de datos, coherente con los procesos de gestión de configuración y documentación de la norma ISO/IEC doce mil doscientos siete. Seguí el ciclo Refinamiento de requisitos → Diseño de contratos → Implementación con validaciones → Pruebas automatizadas → Verificación funcional para cada componente del módulo.

Cómo lo hice: estructuré el desarrollo en fases con entregables verificables: primero refiné seis historias de usuario definiendo criterios de aceptación numerados y reglas de negocio específicas (ciento setenta reglas documentadas que traducen restricciones institucionales en validaciones técnicas); luego diseñé nueve endpoints REST especificando formatos de entrada con validaciones Zod, salida con estructura consistente y reglas de negocio aplicables; después implementé cuatro capas independientes (rutas con autenticación, controladores con formateo, middlewares con autorización, servicios con reglas de negocio) aplicando separación de responsabilidades que facilita testing unitario; finalmente desarrollé suite con doce casos de prueba que validan flujos críticos y condiciones de error.

Definí métricas de calidad objetivas que validé mediante suite automatizada: cobertura de validación de permisos en cien por ciento de endpoints de escritura (crear encuesta verifica permisos de docente, gestionar encuesta verifica autoría), cobertura de validación de entrada en cien por ciento de payload de creación y respuesta (cada campo tiene validación con Zod que especifica tipo, longitudes y valores permitidos), prevención de duplicados en respuestas mediante restricción única en base de datos verificada con caso de prueba que intenta responder dos veces, y consistencia de formato de respuesta en todos los endpoints (estructura unificada con success booleano, data o error con code y message).

Qué usé: metodología ágil de desarrollo incremental por historias de usuario con entregas semanales; patrón de arquitectura en cuatro capas (rutas, controladores, middlewares, servicios) para modularidad y testabilidad; esquemas de validación Zod con tipos estrictos y mensajes descriptivos de error; suite de testing con Vitest y Supertest que valida códigos HTTP, estructura de respuestas y reglas de negocio; versionamiento en GitHub con commits atómicos por funcionalidad; documentación técnica progresiva que se actualiza en paralelo con la implementación.

Aprendizaje: comprendí que la aplicación de estándares de calidad no es cumplimiento formal de procesos, sino práctica estructurada que reduce defectos y facilita mantenimiento. La trazabilidad Historia de Usuario → Endpoint → Controlador → Servicio → Entidad no es documentación redundante, sino mecanismo de verificación que asegura que cada requisito está implementado

completamente y que cada función en el código responde a una necesidad verificable. Esta disciplina permitió detectar inconsistencias tempranamente: una regla de negocio sin validación correspondiente en código señala brecha de implementación que debe corregirse antes de pasar a testing.

CT2.2 – Construcción y gestión de repositorios de datos

Diseñé y gestioné la estructura de datos del módulo de encuestas aplicando normalización selectiva que equilibra integridad con complejidad de consultas. Modelé cinco entidades relacionadas: Encuesta como contenedor principal con configuración y metadatos, PreguntaEncuesta con tipo mediante enumeración de cinco valores y orden para renderizado, OpcionPregunta para almacenar opciones de preguntas de selección con orden independiente del identificador, RespuestaEncuesta como contenedor por usuario con restricción única que previene duplicados, y RespuestaPregunta con campos específicos por tipo de pregunta que almacenan valores en formato adecuado (texto en string, opción única en identificador, opciones múltiples en array de identificadores, escala en entero).

Implementé validaciones de integridad referencial mediante relaciones en Prisma con eliminación en cascada: al eliminar encuesta se eliminan automáticamente preguntas, opciones y respuestas asociadas, manteniendo consistencia sin requerir lógica manual de limpieza. Configuré restricción única compuesta en RespuestaEncuesta sobre la combinación de encuesta_id y usuario_id que garantiza unicidad mediante constraint de base de datos, independiente de validaciones en capa de aplicación.

Cómo lo hice: apliqué normalización hasta tercera forma normal para entidades principales (Encuesta, Pregunta, Opción) evitando redundancia de datos: autor como relación a Usuario no como texto duplicado, tipo de pregunta como enumeración no como string libre, opciones en tabla separada no repetidas por respuesta. Implementé relaciones con integridad referencial: PreguntaEncuesta.encuesta_id referencia Encuesta.id con eliminación en cascada, OpcionPregunta.pregunta_id referencia PreguntaEncuesta.id con cascada, RespuestaPregunta.pregunta_id y respuesta_id referencian sus entidades padres. Configuré índices en campos de filtrado frecuente: índice en RespuestaEncuesta(encuesta_id, usuario_id) para consultas de "ya respondió", índice en PreguntaEncuesta(encuesta_id, orden) para obtención ordenada de preguntas.

Diseñé consultas relacionadas que minimizan número de llamadas a base de datos: uso de include en Prisma para cargar encuesta con autor, preguntas con opciones ordenadas en una sola consulta en lugar de N+uno consultas; uso de _count para calcular total de respuestas sin cargar registros completos; uso de findUnique con clave compuesta para verificar unicidad de respuesta en operación atómica.

Qué usé: Prisma ORM como herramienta de modelado con tipos estrictos, generación automática de cliente y soporte para relaciones complejas; PostgreSQL como motor de base de datos con soporte para restricciones únicas compuestas, enumeraciones nativas y eliminación en cascada; migraciones controladas que evolucionan el esquema sin pérdida de datos; transacciones para operaciones multi-paso que requieren atomicidad (crear encuesta con preguntas y opciones, enviar respuesta con respuestas a preguntas individuales); consultas con filtrado dinámico que construyen cláusulas WHERE según parámetros del usuario.

Aprendizaje: confirmé que la calidad de la estructura de datos determina directamente la complejidad de la lógica de negocio y la robustez del sistema. Modelar respuestas con restricción única previene duplicados de forma confiable sin requerir validaciones complejas en código. Separar opciones en tabla independiente facilita validación de existencia y análisis agregado de respuestas. Usar enumeraciones para tipo de pregunta garantiza valores válidos en toda la aplicación. Esta inversión en diseño de datos reduce defectos, facilita evolución y mejora mantenibilidad a largo plazo.

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

Implementé controles de seguridad en cinco niveles de protección aplicando defensa en profundidad: autenticación obligatoria mediante token JWT que verifica identidad del usuario antes de cualquier operación, autorización por endpoint que restringe acceso según rol declarado en el token (docentes y directores pueden crear, padres solo responder), validación de permisos específicos que consulta tabla PermisoDocente para verificar si el docente tiene autorización activa de tipo encuestas, validación de estructura de datos con esquemas Zod que verifican tipos, formatos, longitudes y valores permitidos antes de procesar, y validación de integridad en base de datos mediante restricciones únicas y relaciones de integridad referencial que previenen datos inconsistentes.

Apliqué el principio de Seguridad del artículo nueve de la Ley veintinueve mil setecientos treinta y tres mediante medidas técnicas que protegen confidencialidad (segmentación automática asegura que usuarios solo ven encuestas pertinentes), integridad (inmutabilidad de respuestas previene manipulación, validaciones previenen datos inválidos) y disponibilidad (ausencia de limitadores de tasa en fase de desarrollo, a implementar antes de producción). Implementé el control A.doce.dos.uno (Controles contra código malicioso) de la ISO/IEC veintisiete mil uno mediante sanitización de contenido de texto que elimina potenciales scripts o HTML malicioso antes de almacenar y mostrar.

Cómo lo hice: configuré autenticación JWT obligatoria en todas las rutas del módulo mediante middleware que valida firma del token, vigencia y estructura antes de permitir acceso a cualquier endpoint; implementé tres middlewares especializados de autorización que verifican permisos según contexto (puedeCrearEncuestas para endpoint POST, puedeGestionarEncuesta para operaciones PATCH/DELETE, puedeVerResultados para análisis); codifiqué validaciones con esquemas Zod que especifican reglas declarativas (título mínimo tres, máximo doscientos; preguntas mínimo uno, cada

una con tipo de enumeración válido; opciones mínimo dos si es selección) que se validan automáticamente antes de procesar; implementé inmutabilidad de respuestas mediante ausencia intencional de endpoint de edición y mensaje explícito al usuario; configuré restricción única en base de datos que intenta insertar y retorna error de conflicto si ya existe combinación de encuesta y usuario.

Implementé trazabilidad de acciones críticas registrando timestamp de creación de encuesta, timestamp de respuesta con dirección IP, y notificación al autor cuando se registra nueva respuesta. Diseñé la eliminación de encuestas como operación condicional: solo se permite si no tiene respuestas registradas (validación que consulta count de RespuestaEncuesta filtrado por encuesta_id), y si se ejecuta, elimina en cascada todas las entidades relacionadas (preguntas, opciones, notificaciones) dentro de transacción para garantizar atomicidad.

Qué usé: middleware de autenticación JWT integrado en capa de ruteo que rechaza tokens inválidos, expirados o sin firma verificable con código cuatrocientos uno; middlewares de autorización que consultan tablas de permisos y verifican autoría antes de permitir operaciones sensibles; esquemas de validación Zod con tipos estrictos, coerción automática y mensajes personalizados de error; transacciones de base de datos para inserción de encuesta con todas sus preguntas y opciones en operación atómica; restricciones únicas en base de datos que previenen duplicados a nivel de motor; relaciones con eliminación en cascada que mantienen integridad referencial automáticamente.

Aprendizaje: aprendí que la seguridad efectiva es resultado de decisiones técnicas correctas en cada capa del sistema, no de un solo control robusto. Si un usuario manipula el frontend para bypass validaciones de longitud, el middleware las verifica nuevamente. Si manipula el middleware, el servicio valida con Zod. Si manipula la inserción directa en base de datos, las restricciones únicas y enumeraciones previenen datos inválidos. Esta defensa en profundidad, alineada con el control A.catorce.uno (Requisitos de seguridad en desarrollo) de la ISO/IEC veintisiete mil uno, me permitió entregar un módulo resiliente ante errores humanos y intentos maliciosos.

CT4.3 – Identificación y análisis de problemas de investigación

Identifiqué como problema central la ausencia de mecanismos estructurados de recopilación de feedback de la comunidad educativa. La institución no contaba con herramientas para medir satisfacción de servicios, evaluar metodologías docentes, o recopilar sugerencias de mejora de forma organizada y analizable. La recopilación informal mediante formularios en papel o encuestas enviadas por mensajería generaba datos dispersos, difíciles de analizar, sin trazabilidad de participación y sin segmentación por destinatarios relevantes.

Cómo lo hice: formulé el problema como requisitos medibles y verificables: "habilitar creación de encuestas con preguntas de cinco tipos diferentes para capturar feedback estructurado y analizable",

"garantizar segmentación automática para dirigir encuestas solo a destinatarios pertinentes según vínculos institucionales", "registrar respuestas de forma inmutable con timestamp para análisis temporal", "validar completitud de preguntas obligatorias para asegurar calidad mínima de feedback", "calcular métricas de participación para evaluar representatividad de resultados". Traduje cada requisito en funcionalidades verificables: panel con tres estados claramente diferenciados, formulario dinámico que renderiza inputs específicos según tipo de pregunta, validaciones defensivas en cuatro capas, y cálculo automático de estadísticas.

Analicé las causas raíz del problema: falta de herramienta centralizada (resuelto con módulo dedicado de encuestas integrado al sistema institucional), ausencia de control de permisos (resuelto con tabla PermisoDocente y validaciones por rol), incapacidad de segmentar destinatarios (resuelto con segmentación automática similar a módulo de Comunicados, pendiente de implementación completa), falta de análisis estructurado de respuestas (resuelto con almacenamiento normalizado que facilita agregaciones y conteos), y ausencia de trazabilidad de participación (resuelto con registro de timestamp, IP y tiempo de respuesta).

Qué usé: historias de usuario para capturar necesidades diferenciadas de cada rol (padre responde encuestas de grados de hijos, docente responde institucionales y crea con permisos, director supervisa y analiza todas); criterios de aceptación con condiciones funcionales verificables que definen comportamiento esperado en cada escenario (qué sucede si usuario ya respondió, qué sucede si encuesta vence, qué sucede si falta pregunta obligatoria); especificación de API con ejemplos concretos de request/response que validan contratos de servicio; suite de testing con casos positivos y negativos que verifican cumplimiento de requisitos; documentación técnica que vincula cada funcionalidad al problema que resuelve con justificación de decisiones de diseño.

Aprendizaje: desarrollé capacidad para descomponer problemas complejos en componentes funcionales independientes y verificables. Identificar el problema de "recopilación de feedback" no es suficiente; debo descomponerlo en subproblemas específicos (¿cómo crear preguntas flexibles?, ¿cómo validar completitud?, ¿cómo prevenir duplicados?, ¿cómo analizar resultados?) y proponer solución técnica verificable para cada uno. Esta descomposición sistemática facilita estimación de esfuerzo, identificación de dependencias y validación progresiva de soluciones.

CT4.4 – Gestión de proyectos de TI

Planifiqué el desarrollo del módulo en cinco fases incrementales con entregables progresivos que entregan valor temprano: fase de análisis y refinación de historias de usuario con criterios de aceptación y reglas de negocio (entregable: especificación funcional con seis HU documentadas); fase de diseño de modelo de datos y contratos de API (entregable: esquema Prisma con cinco entidades y documentación de nueve endpoints); fase de implementación de backend con validaciones defensivas (entregable: código fuente con rutas, controladores, middlewares y servicios

operativos); fase de desarrollo de suite de pruebas automatizadas (entregable: doce casos de prueba que validan permisos, validaciones y flujos críticos); y fase de verificación funcional manual (entregable: validación de segmentación, prevención de duplicados e inmutabilidad de respuestas).

Prioricé funcionalidades según dependencias críticas y riesgo técnico: primero panel y validación de acceso (habilitadores que permiten verificar segmentación), luego formulario y envío de respuestas (valor directo para usuarios finales que habilita recopilación de feedback), después constructor de encuestas con validaciones (habilitador para autores que deben poder crear encuestas propias), luego visualización de respuestas propias (transparencia que genera confianza de usuarios), y finalmente gestión administrativa y análisis de resultados (optimización de flujo de trabajo y toma de decisiones). Esta priorización permitió validar segmentación y permisos tempranamente antes de invertir esfuerzo en funcionalidades dependientes.

Cómo lo hice: organicé el trabajo en sprints con entregables verificables que incrementan capacidad del sistema: primero entregué modelo de datos funcional con migraciones aplicadas que crean las tablas necesarias; luego endpoints de lectura (panel, formulario) que permiten verificar segmentación sin modificar datos; después endpoints de escritura (crear, responder) con validaciones exhaustivas; luego suite de testing que valida cada endpoint y middleware; finalmente verificación manual de flujos completos con usuarios de diferentes roles. Gestioné versionamiento en GitHub con commits descriptivos que vinculan cambios a historias de usuario (ejemplo: "Implementar HU-ENC-cero uno: formulario dinámico de respuesta con validaciones por tipo de pregunta"); documenté decisiones de arquitectura que justifican elecciones técnicas (normalización completa versus JSON desnormalizado, restricción única versus validación en código, inmutabilidad de respuestas versus flexibilidad de edición).

Implementé controles de aseguramiento de calidad progresivos: validaciones de entrada con Zod en cien por ciento de operaciones de creación y respuesta (cada campo tiene esquema que especifica tipo, formato y restricciones), autorización diferenciada en tres niveles de permisos (crear requiere permisos activos de encuestas, gestionar requiere autoría o rol director, ver resultados requiere autoría o participación), y trazabilidad de acciones críticas mediante timestamps (fecha_creacion de encuesta, fecha_respuesta de usuario, fecha_cierre de encuesta cerrada manualmente). Revisé cada función de servicio para verificar manejo de errores con códigos HTTP específicos (cuatrocientos para datos inválidos, cuatrocientos tres para permisos insuficientes, cuatrocientos cuatro para recurso no encontrado, cuatrocientos nueve para conflicto de unicidad) y mensajes descriptivos que facilitan corrección.

Qué usé: cronograma del proyecto para alinear módulo de encuestas con fase de comunicación y feedback; GitHub como repositorio central con estructura de carpetas por capa (routes, controllers, middlewares, services, tests); metodología de historias de usuario con criterios de aceptación

verificables como método de captura de requisitos; patrón de cuatro capas para organización modular que facilita desarrollo y testing independiente de cada capa; testing automatizado con Vitest que ejecuta casos de prueba en ambiente aislado con setup y teardown; documentación técnica con especificación de API, reglas de negocio y decisiones de arquitectura actualizadas progresivamente.

Aprendizaje: comprendí que la gestión efectiva de proyectos de TI requiere equilibrar velocidad de entrega con calidad verificable. Entregar rápido sin validaciones genera deuda técnica costosa de corregir; entregar con calidad excesiva retrasa el valor. El balance lo logré mediante priorización por riesgo: validaciones críticas de seguridad (permisos, prevención de duplicados) se implementan desde el inicio con testing exhaustivo, validaciones de experiencia de usuario (mensajes descriptivos, contadores en tiempo real) se refinan iterativamente según feedback. Mantener trazabilidad bidireccional entre requisitos y código no solo asegura cumplimiento, sino que facilita estimación de impacto de cambios y planificación de correcciones.

Conclusiones

El desarrollo del módulo de Encuestas habilitó un sistema estructurado, flexible y analizable de recopilación de feedback institucional para la I.E.P. Las Orquídeas. Se implementaron cinco tipos de preguntas que capturan feedback cualitativo (texto corto y largo para comentarios abiertos) y cuantitativo (opción única, opción múltiple y escala uno a cinco para métricas agregables), controles de acceso diferenciados por rol que respetan permisos de docentes, segmentación automática de destinatarios según vínculos institucionales, validaciones específicas por tipo de pregunta que aseguran calidad de respuestas, registro inmutable de respuestas con trazabilidad completa, y operaciones de gestión administrativa condicionales por estado de la encuesta.

La arquitectura en cuatro capas (rutas con autenticación, controladores con formateo, middlewares con autorización, servicios con reglas de negocio) y el modelo de datos normalizado con cinco entidades relacionadas facilitan el mantenimiento, la extensión de tipos de preguntas y el análisis de resultados mediante consultas agregadas. La aplicación de principios normativos (Ley veintinueve mil setecientos treinta y tres artículos ocho y nueve sobre proporcionalidad y seguridad, Ley veintiocho mil cuarenta y cuatro sobre participación de comunidad educativa) y buenas prácticas de seguridad (ISO/IEC veintisiete mil uno controles A.nueve, A.doce, A.catorce sobre acceso, operaciones y desarrollo seguro) fortalece el cumplimiento regulatorio y la confianza de la comunidad.

Este módulo contribuye directamente a mejorar la toma de decisiones institucionales basada en evidencia cuantitativa y cualitativa de familias y docentes, reduciendo la subjetividad en evaluaciones de servicios, mejorando la identificación de necesidades mediante preguntas abiertas de texto largo, y facilitando el seguimiento de satisfacción mediante encuestas recurrentes con escalas de valoración. La trazabilidad de respuestas con timestamp, tiempo de respuesta y estudiante relacionado (para padres) habilita análisis de calidad de participación y comparaciones por segmento.

En conclusión, este desarrollo consolidó mis competencias en gestión de repositorios de datos (diseño normalizado con integridad referencial, consultas relacionadas eficientes, índices estratégicos), seguridad de la información (validaciones en cinco niveles, inmutabilidad de datos críticos, trazabilidad de acciones), gestión de proyectos de TI (priorización por riesgo y dependencias, entregas incrementales con valor verificable, trazabilidad de requisitos a código), comunicación técnica (documentación multinivel para diferentes audiencias, mensajes descriptivos orientados a acción) y análisis de problemas (descomposición sistemática de necesidades complejas en requisitos funcionales verificables). Reforzó además mi compromiso ético en el tratamiento responsable de feedback de familias y docentes en contextos educativos.