Informe Final Proyecto Ingeniería de Software

# “Prácticas UBB”

**Integrantes:** Carlos Gómez
Anyelo Gática
Matias Reyes
Nicolás Muñoz
Cristopher Alarcón

**Fecha:** 29 de Diciembre de 2025

# CAPÍTULO I

## I.1. DEFINICIONES, SIGLAS Y ABREVIACIONES

● **JWT (JSON Web Token):** Estándar para la creación de tokens de acceso que permiten la transmisión segura de información entre partes.
● **CRUD:** Acrónimo de Crear, Leer, Actualizar y Borrar (las operaciones básicas de base de datos).
● **Stakeholders:** Actores interesados en el proyecto (Alumnos, Docentes, Jefes de Carrera).
● **Bitácora:** Registro cronológico de actividades que el estudiante debe completar (funcionalidad de compañero).
● **Cupo:** Vacante disponible en una oferta de práctica.

## I.2. PRESENTACIÓN DEL CONTEXTO

El presente proyecto se desarrolla en el contexto de la **Educación Superior**, específicamente orientado a la gestión académica de las **Prácticas Profesionales**. Esta etapa es un hito crítico en la formación de los estudiantes, ya que constituye el primer acercamiento formal al mundo laboral y es un requisito obligatorio para la obtención del título profesional.

En el entorno actual de la institución, el proceso de práctica involucra a tres actores fundamentales que deben coordinarse eficientemente:

1. **El Estudiante:** Quien debe buscar, postular y validar su centro de práctica, además de reportar su progreso.
2. **El Docente Encargado:** Responsable de buscar oportunidades en el mercado, publicarlas, validar las postulaciones y supervisar el desempeño académico del alumno.
3. **La Administración Académica:** Que requiere un registro fidedigno de que el estudiante ha cumplido con los requisitos curriculares.

Actualmente, las instituciones educativas se encuentran en un proceso de **transformación digital**, buscando migrar sus procesos administrativos manuales o análogos hacia plataformas web centralizadas. En este escenario, la gestión de la información relativa a las prácticas (ofertas, cupos, bitácoras de avance y retroalimentación) requiere de un sistema que garantice la integridad de los datos, la inmediatez en la comunicación y la transparencia en la asignación de cupos, reemplazando los métodos informales como el correo electrónico, tableros físicos o planillas de cálculo desconectadas.

Este software se inserta precisamente en este ecosistema, actuando como el puente tecnológico que digitaliza y estandariza el flujo completo de la práctica profesional, desde la publicación de la oferta hasta el cierre del proceso.

## I.3. DESCRIPCIÓN DE LA PROBLEMÁTICA

Actualmente, la gestión de prácticas profesionales se realiza de manera dispersa. Los docentes publican ofertas por correo o tablones físicos, lo que provoca que la información se pierda. No existe un control centralizado de quién postula a qué, ni validación automática de cupos. Además, los estudiantes que consiguen prácticas por fuera (externas) no tienen dónde registrarlas formalmente, y la comunicación (feedback) queda aislada en correos personales.

## I.4. PROPUESTA DE SOLUCIÓN

Desarrollar una plataforma web centralizada que gestione el ciclo de vida completo de la práctica. Desde la publicación de ofertas con control de cupos en tiempo real, pasando por la postulación del estudiante, hasta el seguimiento mediante bitácoras y comunicación directa docente-alumno vía comentarios.

## I.5. ANÁLISIS DE LAS SOLUCIONES SIMILARES DISPONIBLES

Para justificar el desarrollo de la presente solución tecnológica, se realizó un análisis comparativo de las herramientas y métodos que actualmente se utilizan para la gestión de prácticas profesionales, identificando sus fortalezas y carencias frente a los requerimientos específicos de la institución.

A continuación, se detallan las tres categorías principales de soluciones disponibles en el mercado:

### 1. Gestión Manual (Hojas de Cálculo y Correo Electrónico)

Es el método tradicional utilizado por muchas coordinaciones académicas antes de la digitalización.

- Descripción: Uso de Microsoft Excel o Google Sheets para llevar el registro de alumnos y correos electrónicos para el envío de documentos (cartas de aceptación, informes).
- Limitaciones:
    - Descentralización de la información: Los documentos quedan dispersos en bandejas de entrada, dificultando la auditoría.
    - Falta de seguimiento en tiempo real: No permite al estudiante ver el estado de su solicitud ("Revision_Pendiente", "Aprobada") sin preguntar manualmente.

    - Error humano: Alta probabilidad de duplicidad de datos o pérdida de archivos adjuntos.

**2. Plataformas de Gestión de Aprendizaje (LMS genéricos como Moodle o Canvas)**

Herramientas diseñadas para la gestión de cursos académicos que a menudo se adaptan forzosamente para gestionar prácticas.

- Descripción: Se crean "cursos" virtuales donde la entrega de la práctica es una "tarea".
- Limitaciones:
    - Rigidez en el flujo: No están diseñadas para flujos de trabajo específicos de prácticas (ej: validar primero la empresa, luego la carta de aceptación, luego iniciar bitácoras).
    - Ausencia de roles externos: No suelen permitir la gestión eficiente de datos de supervisores externos o empresas sin crear cuentas de usuario complejas.
    - Bitácoras limitadas: La funcionalidad de bitácora semanal con seguimiento de horas y validación docente no es nativa y requiere adaptaciones complejas.

**3. Software de Reclutamiento y RRHH (SaaS comerciales)**

Plataformas como LinkedIn Talent Solutions, JobTeaser o Symplicity.

- Descripción: Software robusto enfocado en conectar empresas con estudiantes.
- Limitaciones:
    - Alto Costo: Suelen requerir licencias anuales costosas en dólares.
    - Enfoque incorrecto: Están diseñadas para que la empresa "contrate", no para que la universidad "evalúe académicamente".
    - Falta de adaptación local: No contemplan validaciones locales como el formato de RUT chileno o la distinción académica entre "Práctica Propia" (autogestionada) vs. "Práctica Publicada" (convenio).

**Conclusión del Análisis:**
Si bien existen herramientas en el mercado, ninguna cubre el flujo completo del ciclo de vida de la práctica profesional (Postulación $\to$ Aceptación $\to$ Bitácoras $\to$ Evaluación) con la especificidad requerida por la normativa interna.

La solución desarrollada ("Sistema de Gestión de Prácticas") cubre este vacío al integrar:

1. Un módulo de Ofertas donde los docentes publican vacantes y los alumnos postulan.
2. Un módulo de Validación de Prácticas Propias, permitiendo al estudiante autogestionar su cupo con subida de evidencia (Cartas de aceptación en PDF).
3. Un sistema de Bitácoras Digitales, que reemplaza el cuaderno físico, permitiendo al docente supervisar el avance semanal y las horas trabajadas de forma remota y centralizada.

## I.6. JUSTIFICACIÓN DEL PROYECTO

Este sistema optimiza el tiempo de los docentes al automatizar la selección (listados de postulantes) y notificaciones por correo. Para el estudiante, democratiza el acceso a las oportunidades y centraliza sus antecedentes académicos en un solo lugar.

# Capítulo II. DEFINICIÓN DEL PROYECTO

## II.1. OBJETIVO GENERAL

Implementar un sistema web para la gestión, administración y seguimiento de prácticas profesionales, facilitando la interacción entre estudiantes y docentes encargados.

## II.2. OBJETIVOS ESPECÍFICOS

1. Gestionar ofertas de práctica (CRUD) con control automático de cupos. 2. Implementar un módulo de postulación segura que notifique vía correo electrónico a los encargados.
3. Permitir el registro de prácticas externas conseguidas autogestionadamente por los alumnos (Aporte compañero).
4. Habilitar un sistema de bitácoras para el seguimiento del avance del estudiante (Aporte compañero).
5. Establecer un canal de comentarios para feedback directo en la plataforma (Aporte compañero).

## II.3. METODOLOGÍA DE DESARROLLO

Para el desarrollo del presente proyecto "Sistema de Gestión de Prácticas", se ha seleccionado la metodología ágil **Scrum**. Esta elección se justifica debido a la necesidad de realizar entregas incrementales y funcionales del software, permitiendo una rápida adaptación a los cambios en los requisitos y una división clara de tareas entre los integrantes del equipo.

El ciclo de desarrollo se organiza en iteraciones cortas denominadas *Sprints*, con una duración de [ej: 2 semanas], donde se abordan los siguientes hitos:

1. **Product Backlog:** Se definió una lista priorizada de funcionalidades (Historias de Usuario), tales como "Registro de estudiante", "Publicación de oferta", "Subida de bitácora", etc.
2. **Sprint Planning:** Al inicio de cada iteración, el equipo selecciona las tareas a desarrollar basándose en la prioridad y complejidad.
3. **Desarrollo Incremental:** Se trabaja en paralelo separando las responsabilidades de *Frontend* y *Backend*, realizando integraciones continuas mediante control de versiones.
4. **Revisión (Review):** Al finalizar cada sprint, se verifica que las funcionalidades (como el envío de correos o la validación de RUT) cumplan con los criterios de aceptación.

## II.4. DESCRIPCIÓN DE LAS ACTIVIDADES PARA LOGRAR LOS OBJETIVOS

## ESPECÍFICOS

Para dar cumplimiento a los objetivos planteados, se han definido las siguientes actividades técnicas agrupadas por fases del ciclo de vida del software:

**Objetivo Específico 1:** Analizar y diseñar la arquitectura del sistema.

- Recolección y refinamiento de requisitos funcionales y no funcionales con los stakeholders (docentes y administrativos).
- Diseño del Modelo Entidad-Relación (MER) normalizado para la base de datos PostgreSQL.
- Diseño de diagramas de flujo y casos de uso para los procesos de postulación y validación de prácticas.
- Prototipado de interfaces de usuario (UI) utilizando herramientas de diseño [ej: Figma/Adobe XD].

**Objetivo Específico 2:** Implementar el servicio Backend y la Base de Datos.

- Configuración del entorno de servidor con **Node.js** y el framework **Express**.
- Implementación del ORM **TypeORM** para la gestión de modelos y migraciones hacia PostgreSQL.
- Desarrollo de módulos de seguridad: autenticación vía **JWT** (JSON Web Tokens) y encriptación de contraseñas.
- Creación de API RESTful siguiendo el patrón MVC (Controladores, Servicios y Rutas) para entidades clave: Usuarios, Ofertas, Prácticas y Bitácoras.
- Implementación de servicios auxiliares: subida de archivos (Multer) y notificaciones por correo electrónico (Nodemailer).

**Objetivo Específico 3: Implementar la interfaz de usuario (Frontend) e integración.**

- Desarrollo de una SPA (Single Page Application) utilizando **React** y **Vite**.
- Implementación de gestión de estado y consumo de API mediante **Axios**.
- Desarrollo de componentes reutilizables (Navbar, Formularios Dinámicos, Tablas de gestión).
- Implementación de rutas protegidas según roles de usuario (Estudiante, Docente, Administrador).

**Objetivo Específico 4: Validar y desplegar la solución.**

- Pruebas unitarias y de integración de los endpoints (API Testing).
- Pruebas de aceptación de usuario (UAT) para verificar flujos críticos como la subida de documentos PDF.
- Corrección de errores y optimización de consultas a la base de datos.
- Documentación final y manual de usuario.

## II.5. ESTÁNDARES DE DOCUMENTACIÓN

Para garantizar la mantenibilidad y legibilidad del código fuente, se han adoptado los siguientes estándares:

1. **Estándar de Código (Coding Style):**
    - Se utiliza la convención **CamelCase** para variables y funciones en JavaScript (ej: crearPractica, obtenerOfertas).
    - Se utiliza **PascalCase** para nombres de Componentes en React y Entidades en el Backend (ej: PracticaSchema, Navbar.jsx).
    - El código está estructurado modularmente, separando claramente la lógica de negocio (Servicios) de la lógica de control (Controladores).
2. **Documentación de API:**
    - Los endpoints se documentan detallando: Método HTTP, URL, Parámetros requeridos (Body/Query), y Respuestas esperadas (Códigos de estado 200, 400, 500).
3. **Control de Versiones (Git):**
    - Se utiliza el estándar **Conventional Commits** para el historial de cambios, facilitando la trazabilidad. Ejemplos:
        - feat: Nueva funcionalidad (ej: feat: agregar subida de archivos).
        - fix: Corrección de errores (ej: fix: error en validación de fecha).
        - refactor: Mejoras de código sin cambio de funcionalidad.

## II.6. TÉCNICAS Y NOTACIONES

El desarrollo del proyecto se apoya en técnicas de ingeniería de software modernas y notaciones estandarizadas:

**Técnicas:**

- **Arquitectura MVC (Modelo-Vista-Controlador):** Utilizada en el Backend para desacoplar la lógica de acceso a datos (TypeORM Entities), la lógica de negocio (Services) y la interfaz de respuesta HTTP (Controllers).
- **API RESTful:** Diseño de interfaces de comunicación basadas en los verbos HTTP estándar (GET, POST, PUT, DELETE) y transferencia de datos en formato JSON.
- **Validación de Esquemas:** Uso de la librería **Joi** para asegurar la integridad de los datos de entrada antes de procesarlos en el servidor.

**Notaciones:**

- **UML (Lenguaje Unificado de Modelado):**
    - *Diagrama de Clases:* Para representar la estructura de la base de datos y las relaciones entre entidades (Estudiante, Práctica, Docente, Bitácora).

    - *Diagrama de Secuencia:* Para modelar la interacción entre el Cliente (Frontend), el Servidor (API) y la Base de Datos en procesos complejos como el Login o la Postulación.
- **DER (Diagrama Entidad-Relación):** Para el modelado lógico de la base de datos relacional en PostgreSQL.

## II.7. HERRAMIENTAS, FRAMEWORK, LENGUAJE

- **Frontend:** React (Vite), CSS3, SweetAlert2 (para alertas), Axios (conexión API).
- **Backend:** Node.js con Express.
- **Base de Datos:** PostgreSQL (o la que estés usando) gestionada con TypeORM.
- **Seguridad:** Bcrypt (encriptación claves) y JWT (autenticación).
- **Servicios:** Nodemailer (envío de correos SMTP Google).

# Capítulo III. DEFINICIÓN DEL SOFTWARE

## III.1. OBJETIVO GENERAL DEL SOFTWARE PROPUESTO

Desarrollar e implementar una plataforma web para la Gestión Integral de Prácticas Profesionales, que permita centralizar, automatizar y supervisar los procesos de postulación, validación, seguimiento de bitácoras y evaluación final, facilitando la interacción entre estudiantes, docentes supervisores y administrativos de la institución.

## III.2. OBJETIVOS ESPECÍFICOS DEL SOFTWARE PROPUESTO

1. Digitalizar el flujo de inscripción: Permitir a los estudiantes registrar sus prácticas, ya sean gestionadas por la universidad (Ofertas publicadas) o autogestionadas (Prácticas propias), eliminando el uso de formularios en papel.
2. Optimizar la difusión de vacantes: Proveer a los docentes un módulo para publicar y administrar ofertas de práctica disponibles, gestionando los cupos y requisitos de manera transparente.
3. Facilitar el seguimiento académico: Implementar un sistema de Bitácoras Digitales que permita al estudiante reportar semanalmente sus actividades y al docente visualizar el avance y las horas realizadas en tiempo real.
4. Centralizar la documentación: Gestionar el almacenamiento seguro de documentos críticos (Cartas de aceptación, informes de evaluación) vinculados directamente al expediente digital de cada práctica.

## III.3. LÍMITES

El sistema se limitará exclusivamente a la gestión académica y administrativa del proceso de práctica dentro de la institución educativa.

- No incluye: Gestión de nóminas o pagos de sueldos por parte de la empresa, generación legal de contratos laborales, ni seguimiento de tareas internas propias de la empresa empleadora.
- Alcance Geográfico: El sistema está diseñado para operar vía web, permitiendo el acceso remoto para prácticas en modalidades presencial, virtual o híbrida.

## III.4. RESTRICCIONES TÉCNICAS

Para el correcto funcionamiento y despliegue del software, se establecen las siguientes restricciones:

1. Conectividad: Requiere conexión a Internet estable para el acceso a la interfaz web y la comunicación con la base de datos en la nube.

2. Navegadores: Optimizado para navegadores modernos (Google Chrome, Mozilla Firefox, Edge) en sus versiones actualizadas.
3. Infraestructura Backend: El servidor debe soportar un entorno Node.js y contar con librerías específicas para la gestión de archivos (Multer) y seguridad (Bcrypt, JWT).
4. Base de Datos: El sistema requiere un motor de base de datos relacional PostgreSQL compatible con el ORM TypeORM para la integridad referencial de los datos.
5. Formatos de Archivo: El sistema restringe la subida de documentos probatorios exclusivamente a formatos estándar no editables (PDF) o de imagen, con un límite de tamaño predefinido por el servidor.

## III.5. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

A continuación, se detallan las funcionalidades agrupadas por módulos, integrando las responsabilidades de todos los roles (Estudiante, Docente, Admin):

### RF de Comentarios

El sistema debe permitir al estudiante enviar un comentario a su profesor guía, requiriendo obligatoriamente la selección de una Prioridad (Normal o Alta) y un Tipo de Alerta ('Problema de Empresa' o 'Problema General o Personal') antes de procesar el envío.

El sistema debe proporcionar la opción de subir documentos adjuntos (como licencias médicas o certificados) al formulario de consulta, permitiendo que el estudiante envíe el comentario incluso si no adjunta ningún archivo (campo opcional).

El sistema debe permitir el envío de comentarios únicamente si valida que el estudiante tiene una práctica registrada y activa en el sistema, bloqueando la funcionalidad si no existe esta asociación.

### RF de Entregas y evaluaciones finales

El estudiante deberá entregar su informe final y su autoevaluación al concluir el periodo de prácticas, únicamente si la práctica se encuentra activa o en progreso. El informe será evaluado por el docente o evaluador asignado, deberá estar en formato PDF o Word, con el diseño predeterminado por la universidad, y cumplir con los límites de tamaño y contenido (no vacío). No se aceptarán documentos fuera de los plazos establecidos. El informe debe basarse en un resumen de las bitácoras, y conforme al reglamento institucional. La evaluación consistirá en una calificación

diaria de las actividades realizadas y una reflexión o comentario final sobre la práctica. El docente o evaluador deberá registrar las calificaciones correspondientes en el sistema, una ves verificado los archivos y revisado si está conforme con el cumplimiento de las normas de la universidad.

**RF de Inscripción de práctica**

El software permitirá al estudiante inscribirse a una práctica profesional conseguida por cuenta propia. El estudiante deberá ingresar la información correspondiente adjuntando los documentos requeridos, asegurando que la práctica cumpla con los requisitos mínimos establecidos por la Universidad, tales como empresa, periodo y supervisor. Una vez enviada la solicitud, ésta quedará registrada con estado “Revisión pendiente” y deberá ser revisada, aprobada o rechazada por el encargado de prácticas.

**RF de Ofertas de practica**

Publicar ofertas de prácticas: El encargado de prácticas podrá ofrecer (publicar) una o varias prácticas con n cupos para los estudiantes, completando un formulario con los siguientes campos: título de la oferta, descripción del cargo, requisitos, duración, modalidad (presencial u online), jornada, ubicación, cupos y fecha límite. Cada uno de los campos del formulario debe estar completo para que la oferta pueda ser publicada. Una vez publicada, la oferta quedará visible para los estudiantes en el apartado de "Ofertas"

**RF de Bitácoras**

El sistema deberá permitir que el estudiante con una práctica activa registre semanalmente una bitácora digital con la información correspondiente a la semana, incluyendo la descripción de las actividades realizadas, las horas trabajadas, los resultados o aprendizajes obtenidos y, de forma opcional, archivos de respaldo en formato pdf o docx con un tamaño máximo de x mb. Solo se permitirá el registro de una bitácora por semana y por práctica activa. El sistema deberá validar la existencia de una práctica activa antes de habilitar el registro. La bitácora deberá cumplir con el formato establecido, la extensión mínima definida y el límite de espacio permitido. Una vez enviada, la bitácora no podrá ser modificada por el estudiante lo que sí tiene opción es que al momento de subir un posible archivo este lo puede eliminar antes de subirlo, al igual que el profesor docente puede aceptar, confirmar o rechazar las bitácoras de el estudiante este también puede descargar el archivo pdf para poder evaluar de mejor manera al estudiante con los archivos pedidos

### III.6. REQUERIMIENTOS NO FUNCIONALES

● Seguridad: Las contraseñas no se guardan en texto plano.
● Usabilidad: Interfaz responsiva y feedback visual inmediato (alertas de carga, éxito y error).
● Rendimiento: Carga de listados optimizada mediante paginación
● Memoria: Debe registrar y guardar todo documento subido a la plataforma

### III.7. INTERFACES EXTERNAS DE ENTRADA

Corresponden a los mecanismos mediante los cuales ingresa información al sistema:

● Formularios Web: Interfaces gráficas diseñadas en React para la captura de datos alfanuméricos (Datos de empresa, descripción de ofertas, registro de usuarios.

●Carga de Archivos (File Upload): Interfaz tipo “Drag & Drop” o selector de archivos para la ingesta de documentos PDF (Carta de aceptación, evaluaciones)

● Credenciales de Acceso: Campos de entrada segura para la autenticación de usuarios mediante tokens cifrados.

### III.8. INTERFACES EXTERNAS DE SALIDA

Corresponden a la información que el sistema entrega al usuario o a otros sistemas:

1. Vistas de Tablero (Dashboards): Tablas dinámicas que muestran listados de ofertas, estado de prácticas y resumen de bitácoras.
2. Notificaciones Emergentes (SweetAlerts): Mensajes visuales de confirmación de éxito (ej: "Práctica creada exitosamente") o alertas de error de validación.
3. Archivos Descargables: Enlaces seguros para visualizar o descargar los documentos PDF previamente almacenados en el servidor.
4. Correos Electrónicos: Notificaciones automáticas enviadas a la bandeja de entrada de los usuarios para informar cambios de estado (Aprobación de práctica, Nueva oferta).

# CAPÍTULO IV. FACTIBILIDAD

El estudio de factibilidad tiene como propósito determinar la viabilidad del proyecto "Sistema de Gestión de Prácticas" desde tres perspectivas críticas: técnica, operativa y económica. Este análisis permite prever riesgos y asegurar que los recursos disponibles son suficientes para el éxito del desarrollo e implementación.

## IV.1. FACTIBILIDAD TÉCNICA

La factibilidad técnica evalúa si se cuenta con la infraestructura, el hardware, el software y los conocimientos necesarios para desarrollar el sistema.

- **Hardware:** El equipo de desarrollo cuenta con computadores personales con capacidad de procesamiento suficiente (procesadores Intel Core i5/i7 o equivalentes, mínimo 8GB RAM) para ejecutar entornos de desarrollo local, servidores de base de datos y herramientas de virtualización. No se requiere la adquisición de hardware especializado adicional.
- **Software y Herramientas:** El proyecto se basa en tecnologías de código abierto (Open Source), lo que elimina costos de licenciamiento:
    - Frontend: React.js con Vite (Licencia MIT).
    - Backend: Node.js con Express (Licencia MIT).
    - Base de Datos: PostgreSQL (Código abierto).
    - ORM: TypeORM para la abstracción de datos.
    - Control de Versiones: Git y GitHub.
    - Editor de Código: Visual Studio Code (Gratuito).
- **Conocimientos del Equipo**: Los integrantes poseen las competencias técnicas necesarias en el stack de desarrollo seleccionado (MERN/PERN Stack), así como conocimientos en ingeniería de software para aplicar patrones de diseño (MVC) y metodologías ágiles.

**Conclusión Técnica:** Dado que se dispone de los equipos necesarios y se utiliza software libre sin costos de licencia, el proyecto es técnicamente factible.

## IV.2. FACTIBILIDAD OPERATIVA

La factibilidad operativa analiza si el sistema será aceptado y utilizado eficazmente por los usuarios finales (estudiantes, docentes y administrativos) y si mejora los procesos actuales.

- **Adopción del Usuario:** La interfaz ha sido diseñada pensando en la usabilidad (UX/UI), facilitando la transición desde el manejo manual de papeles hacia lo digital. Al ser una aplicación web, no requiere instalación en los dispositivos de los usuarios, garantizando acceso universal desde cualquier navegador.

- **Mejora de Procesos:** Actualmente, la gestión manual implica riesgos de pérdida de documentos y lentitud en la retroalimentación. El sistema automatiza validaciones (fechas, RUT) y centraliza la información, resolviendo "cuellos de botella" administrativos reales.
- **Soporte Institucional:** El sistema se alinea con las políticas de modernización y transformación digital de la institución educativa.

Conclusión Operativa: El sistema resuelve problemas críticos de gestión y burocracia, por lo que se prevé una alta tasa de adopción y resistencia mínima al cambio. El proyecto es operativamente factible.

### IV.3. FACTIBILIDAD ECONÓMICA

Este análisis determina si los beneficios económicos (o el ahorro de costos) superan la inversión necesaria para desarrollar y mantener el sistema.
Nota: Para este cálculo, se consideran "Costos Evitados" (ahorro de horas-hombre de secretaría y docentes) como "Ingresos", ya que es un proyecto interno.

#### IV.3.1. FLUJO DE CAJA ( solo teórico, ofrece vista de cuanto se ahorraria con la utilización de este proyecto)

Se proyecta un flujo de caja a 12 meses.

- **Inversión Inicial ($I_0$):** Considera el tiempo de desarrollo estimado (valorado) y configuración de servidores.
- **Costos Operativos:** Hosting en la nube (ej. Render/AWS) y dominio anual.
- **Beneficios:** Ahorro estimado de 20 horas mensuales de gestión administrativa valorizadas a $5.000 CLP/hora.

| Concepto | Inversión (Mes 0) | Mes 1 al 11 | Mes 12 |
|---|---|---|---|
| INGRESOS (Ahorro) | | $100.000 | $ 100.000 |
| Ahorro Administrativo | | $100.000 | $ 100.000 |
| EGRESOS (Costos) | $500.000 | $ 15.000 | $ 15.000 |
| Desarrollo (Valorizado) | $ 500.000 | - | - |
| Hosting y Dominio | - | $15.000 | $ 15.000 |
| FLUJO NETO | ($ 500.000) | $ 85.000 | $ 85.000 |

(Los montos son referenciales. $500.000 representa el costo ficticio de desarrollo si se pagara a los estudiantes, y $15.000 el costo mensual de un servidor básico).

### IV.3.2. CÁLCULO DEL V.A.N (Valor Actual Neto)

El V.A.N. permite traer al valor presente los flujos futuros para saber si la inversión es rentable hoy. Se utilizará una Tasa de Descuento ($k$) del 10% anual (0.83% mensual aprox).

La fórmula utilizada es:

$$VAN = -I_0 + \sum_{t=1}^{n} \frac{F_t}{(1+k)^t}$$

Donde:

- I0 = 500.000$ (Inversión Inicial)
- Ft = 85.000$ (Flujo Neto Mensual)
- k = 0.0083$ (Tasa mensual)
- n = 12$ (Meses)

Cálculo:

$$VAN = -500.000 + \frac{85.000}{(1,0083)^1} + \frac{85.000}{(1,0083)^2} + \ldots + \frac{85.000}{(1,0083)^{12}}$$

Al realizar la sumatoria de los flujos descontados:

- Valor Presente de los Flujos = $968.000 CLP
- Resta de la Inversión: $968.000 - 500.000$

Van =468.000 CLP (aprox)

Interpretación:

Al obtener un VAN > 0, significa que el proyecto recupera la inversión inicial (valorada) y genera un excedente económico equivalente a $468.000 CLP en el primer año, gracias al ahorro de tiempos administrativos.

### IV.4. CONCLUSIÓN DE LA FACTIBILIDAD

Tras analizar las tres dimensiones de la factibilidad, se concluye lo siguiente:

1. Tecnológicamente, el equipo posee las herramientas y el know-how para completar el desarrollo sin dependencias externas costosas.
2. Operativamente, existe una necesidad real y urgente de digitalizar el proceso de prácticas, asegurando su uso.
3. Económicamente, el proyecto presenta un VAN positivo, demostrando que la implementación del sistema resulta más eficiente y económica para la institución que mantener el proceso manual actual.

En consecuencia, el proyecto "Sistema de Gestión de Prácticas" se declara totalmente viable para su ejecución.

# CAPÍTULO V. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

**V.1. ESPECIFICACIÓN DE LA SOLUCIÓN** Aplicación Web tipo SPA (Single Page Application) basada en arquitectura cliente-servidor RESTful.

**V.2. OPTIMIZACIÓN DEL RENDIMIENTO** Se implementó **paginación** en el listado de ofertas públicas para evitar sobrecarga de datos en el cliente. Se utiliza useEffect
y useCallback en React para evitar renderizados innecesarios y consumo excesivo de memoria.

**V.3. COMPONENTE: MODELO DE DATOS**

- Entidad User: Roles y datos personales.
- Entidad OfertaPractica: Relación "Many-to-Many" con User (tabla intermedia de postulaciones) y "Many-to-One" con User (encargado).
- Entidad Documentos Relación "Many-to-One" con User y "Many-to-One" con Practica
- Entidad EvaluaciónFinal Relación "Many-to-One" con User y "Many-to-One" con Documento
- Entidad Práctica Relación "Many-to-One" con User y "Many-to-One" con User
- Entidad Postulación Relación "Many-to-One" con User y "Many-to-One" con OfertaPractica
- Entidad Cometario Relación "Many-to-One" con User y “Many-to-One con Comentario

**V.3.1. MODELO DE DETECCIÓN Y VALIDACIÓN DE ARCHIVOS**

Debido a que el sistema permite la carga de documentación externa (Cartas de Aceptación y Evaluaciones), se ha implementado un **Modelo de Detección de Tipos de Archivo (MIME Type Detection)** basado en firmas digitales, para garantizar la integridad y seguridad del servidor.

Este modelo no confía únicamente en la extensión del archivo (ej: .pdf), la cual es fácilmente falsificable, sino que inspecciona la cabecera binaria del documento.

1. Estrategia de Detección:

El sistema utiliza el método de "Números Mágicos" (Magic Numbers). Cada formato

de archivo posee una firma hexadecimal única en sus primeros bytes. El modelo de detección opera interceptando el flujo de datos entrante (Buffer) antes de guardar el archivo en el disco.

**2. Algoritmo de Detección Implementado:**

- **Paso 1: Intercepción.** El middleware de carga (Multer) recibe el flujo de datos multipart/form-data proveniente del Frontend.
- **Paso 2: Lectura de Cabecera.** El sistema lee los primeros 4 a 8 bytes del archivo (el "encabezado").
- **Paso 3: Comparación de Patrones.** Se compara la firma hexadecimal leída con una lista blanca de firmas permitidas en el sistema.
    - *Patrón PDF:* 25 50 44 46 (ASCII: .PDF)
    - *Patrón DOCX:* 50 4B 03 04 (Formato ZIP/Office Open XML)
- **Paso 4: Decisión.**
    - **Coincidencia Positiva:** Si la firma hexadecimal coincide con el tipo de contenido declarado, el archivo se procesa y se almacena en la carpeta /uploads.
    - **Detección de Anomalía:** Si la extensión dice .pdf pero la firma hexadecimal corresponde a un ejecutable (4D 5A), el modelo detecta la amenaza, rechaza la carga y retorna un error 400 (Bad Request).

**3. Diagrama Lógico de Detección:**

Usuario sube archivo ->Extracción de Bytes (Header)-> ¿Coincide con Firma .PDF?

- **SÍ** -> Almacenar archivo ->Guardar ruta en BD
- **NO** -> Eliminar temporal -> Emitir Alerta de Seguridad

**V.3.2. MÉTRICAS**

Para garantizar la calidad y el rendimiento del software desarrollado, se han definido métricas clave de desempeño (KPIs) divididas en dos categorías: métricas de proceso (durante el desarrollo) y métricas de producto (funcionamiento del sistema).

**Métricas de Calidad del Código:**

- **Cobertura de Pruebas (Code Coverage):** Porcentaje de líneas de código ejecutadas durante las pruebas unitarias. Se establece un objetivo mínimo del 80% para la lógica crítica del negocio (Servicios y Controladores).
- **Deuda Técnica:** Evaluación estática del código mediante *linters* (ESLint) para identificar malas prácticas, variables no utilizadas o complejidad ciclomática excesiva.

**Métricas de Rendimiento del Sistema:**

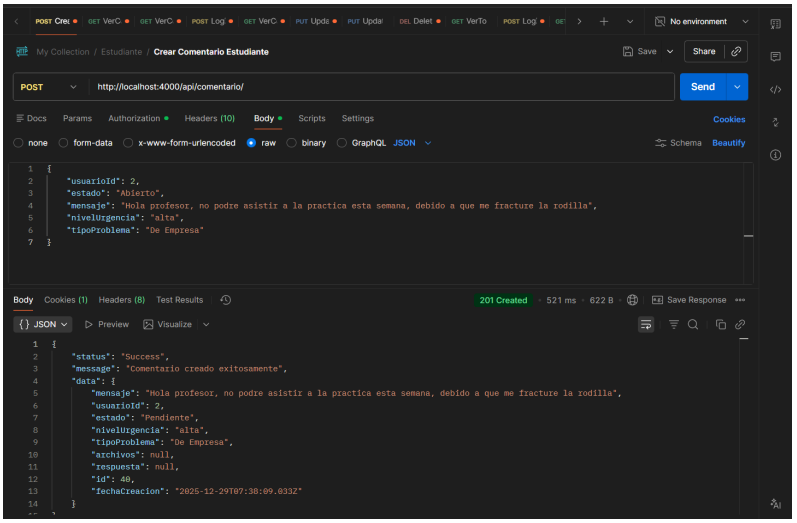
- **Tiempo de Respuesta de la API:** El tiempo promedio de respuesta del servidor (Backend) para operaciones de lectura (GET) debe ser inferior a 200ms y para operaciones de escritura/subida de archivos (POST con Multer) inferior a 2 segundos.
- **Tasa de Errores HTTP:** Monitoreo del porcentaje de respuestas con códigos de estado 4xx (Error de Cliente) y 5xx (Error de Servidor). El umbral aceptable de errores 500 es inferior al 1%.

**3.3. Diagrama de flujo de eventos:**

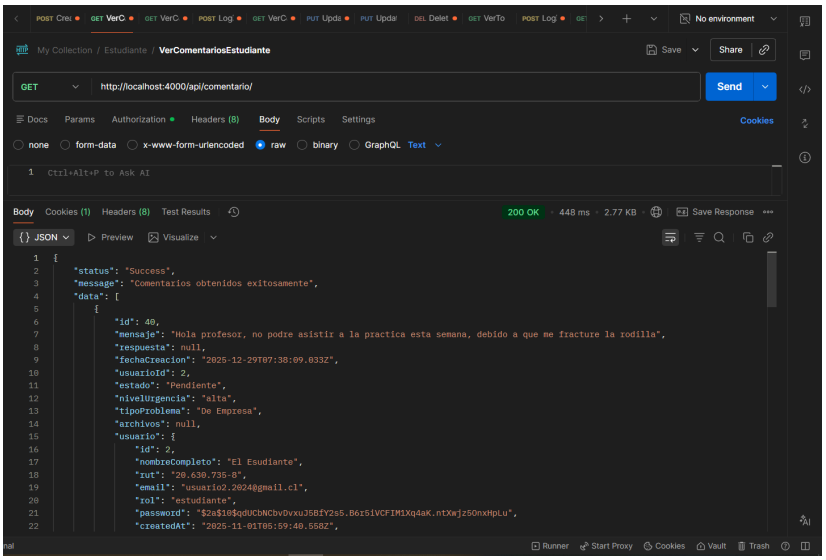
**3.6. PRUEBAS: [https://github.com/EstebanBra/Proyecto-Practicas.git](https://github.com/EstebanBra/Proyecto-Practicas.git)**

**Comentarios:**

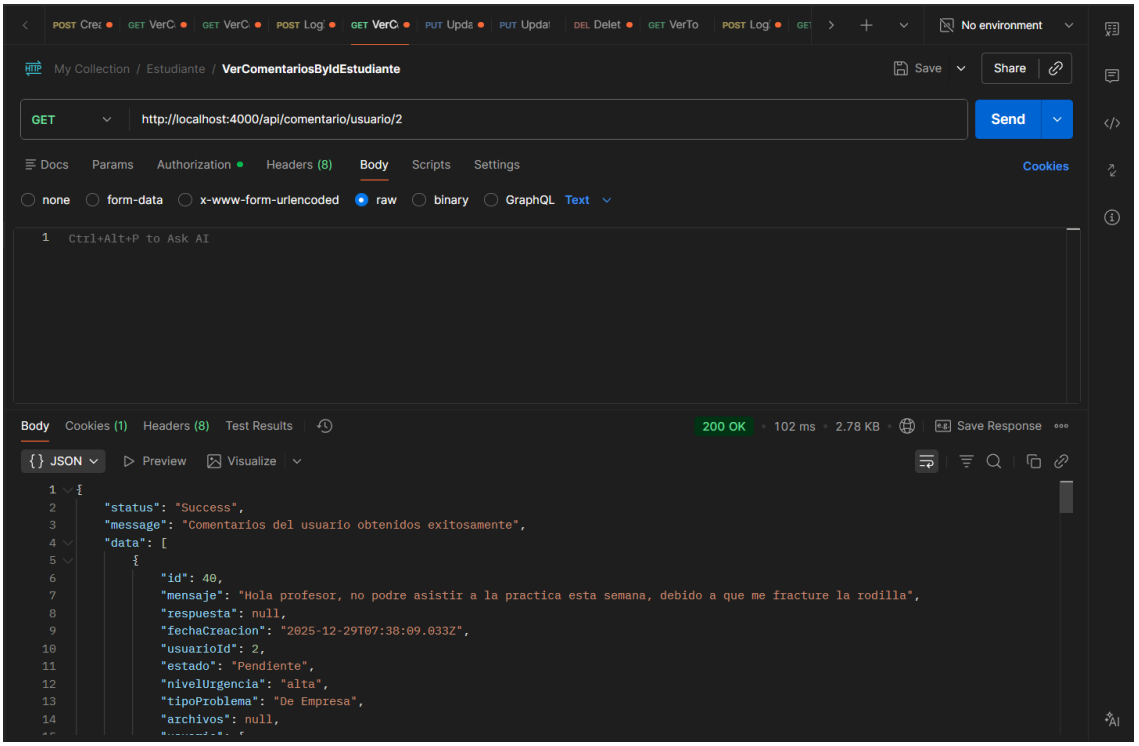
**Crear Comentario:**



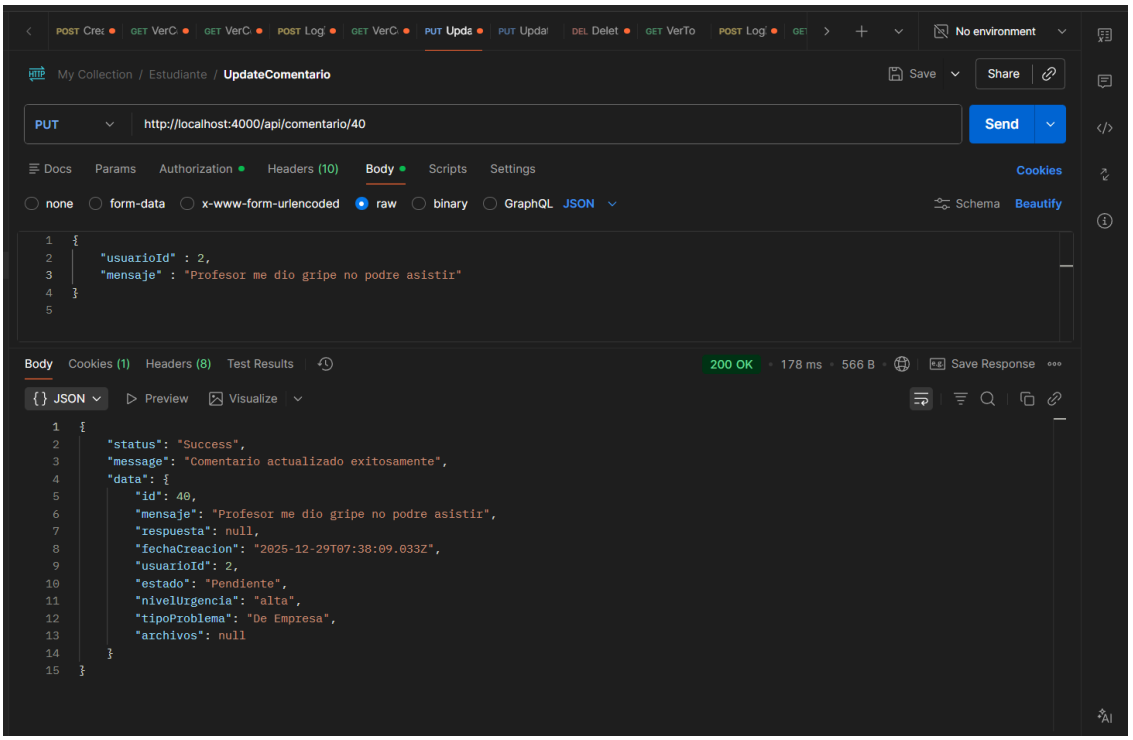

**Ver Comentarios del Estudiante:**



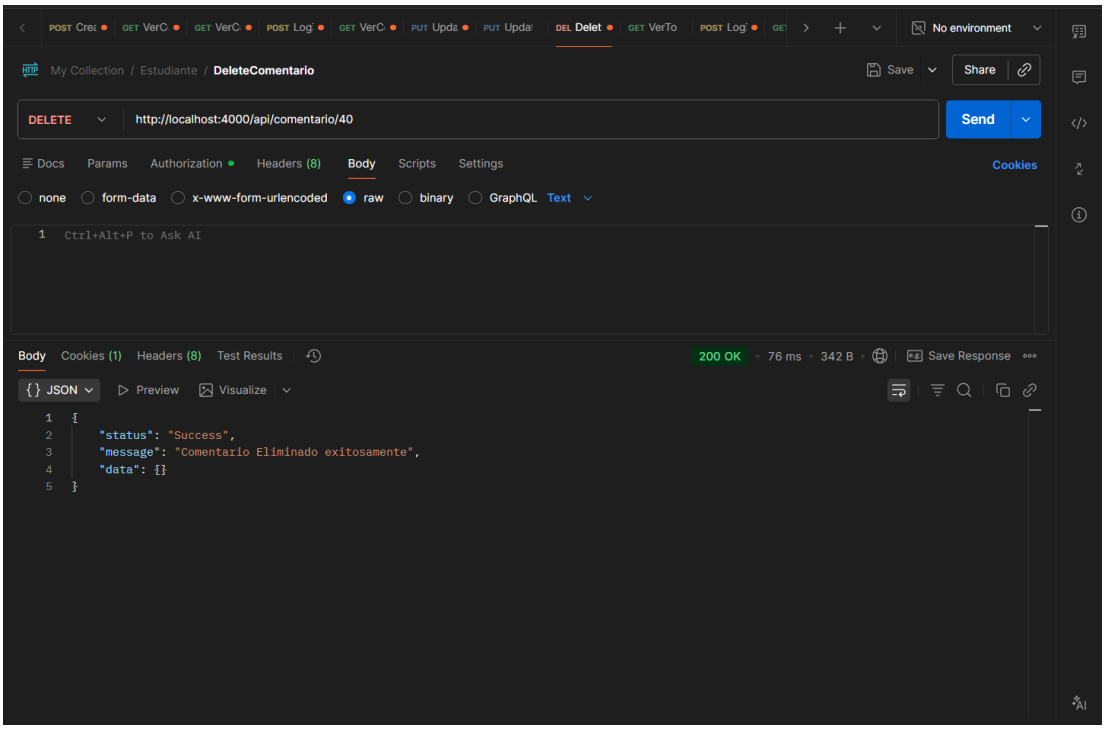

## Ver Comentarios por ID del Estudiante:



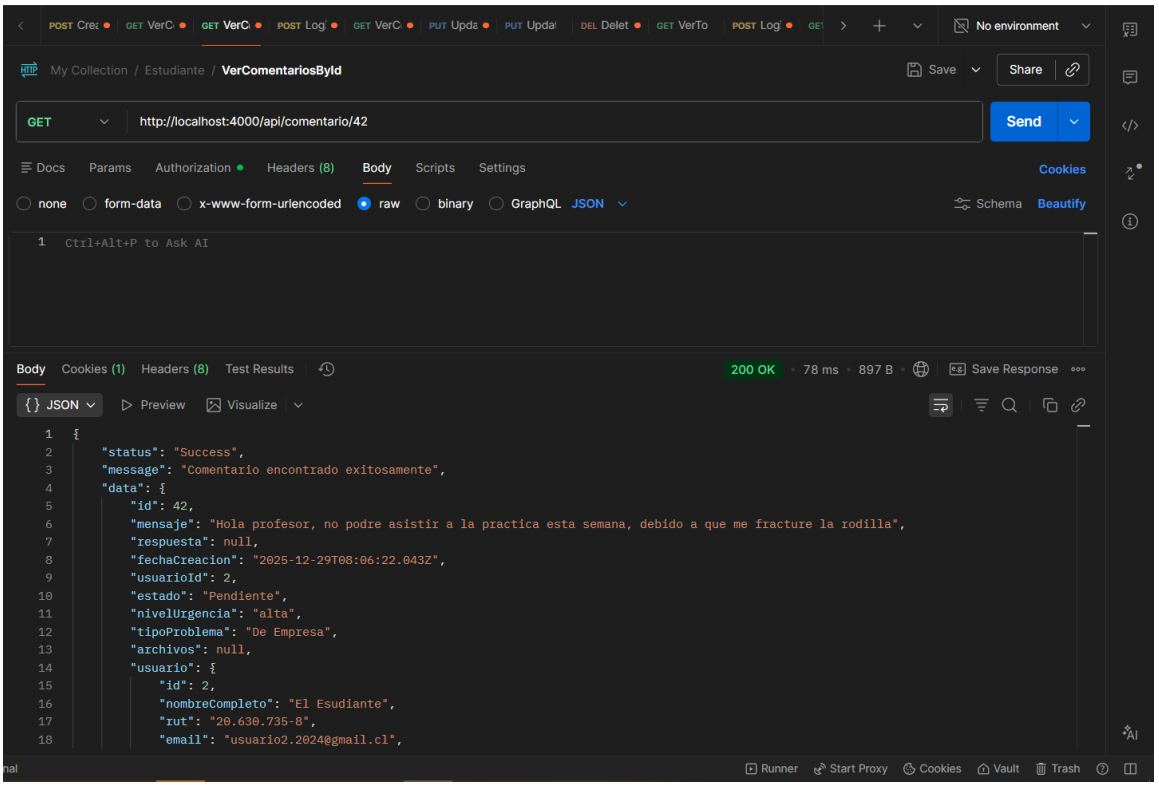

## Actualizar Comentario:



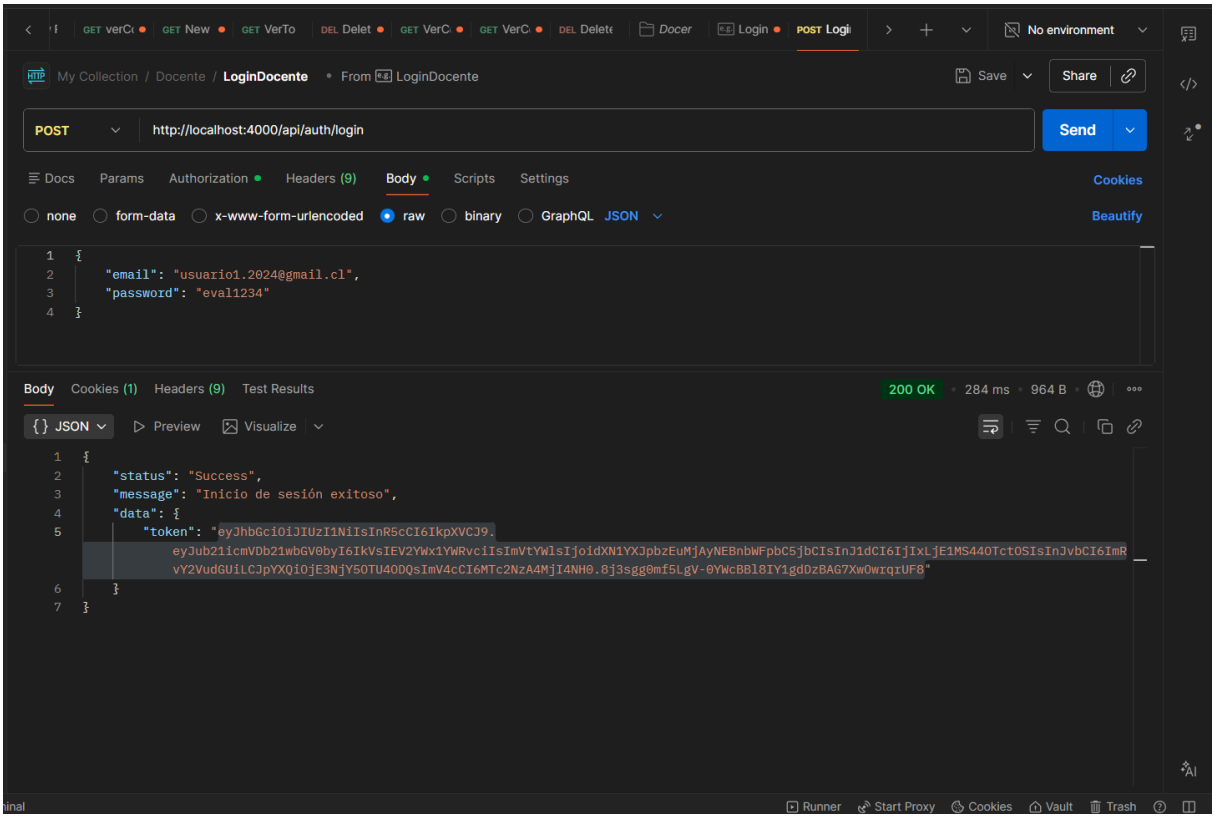

## Eliminar Comentario:



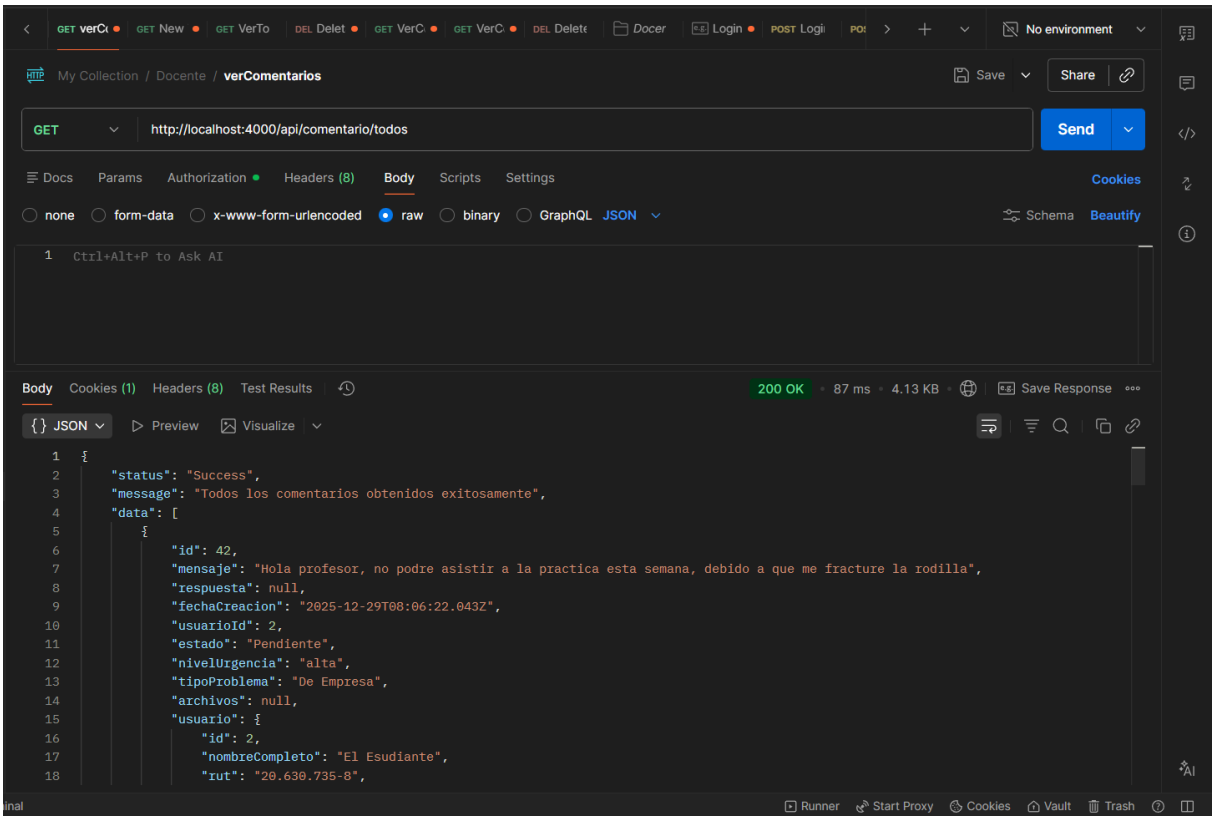

## Ver Comentario por Id:



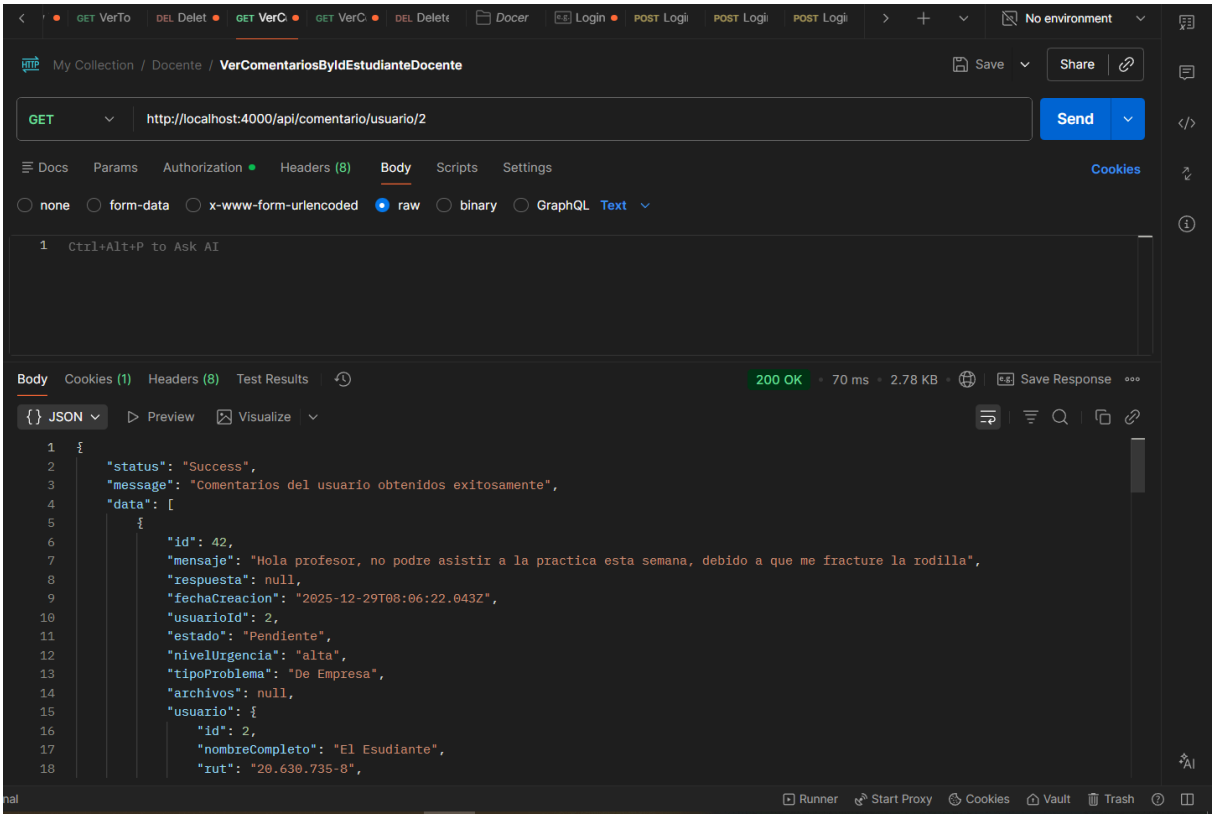

## Inicio Sesion de Docente:



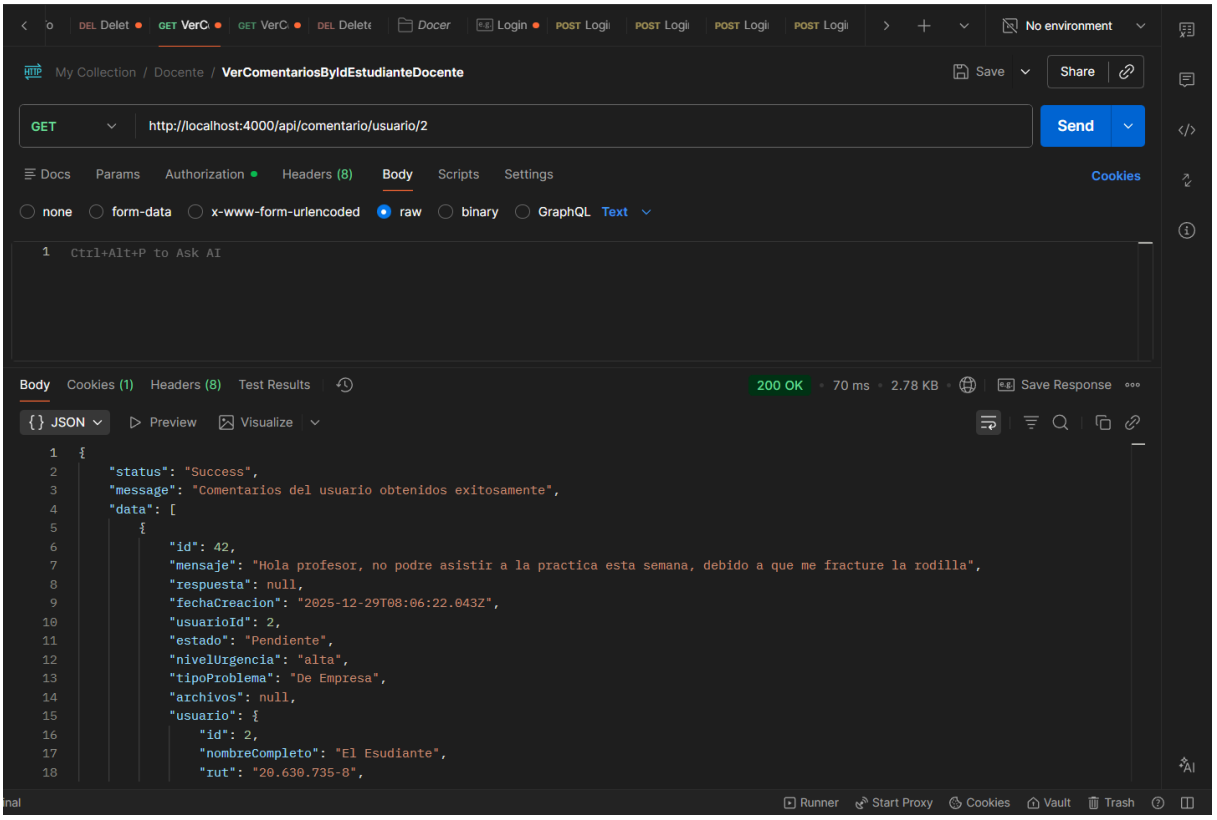

## Ver Todos los Comentarios:



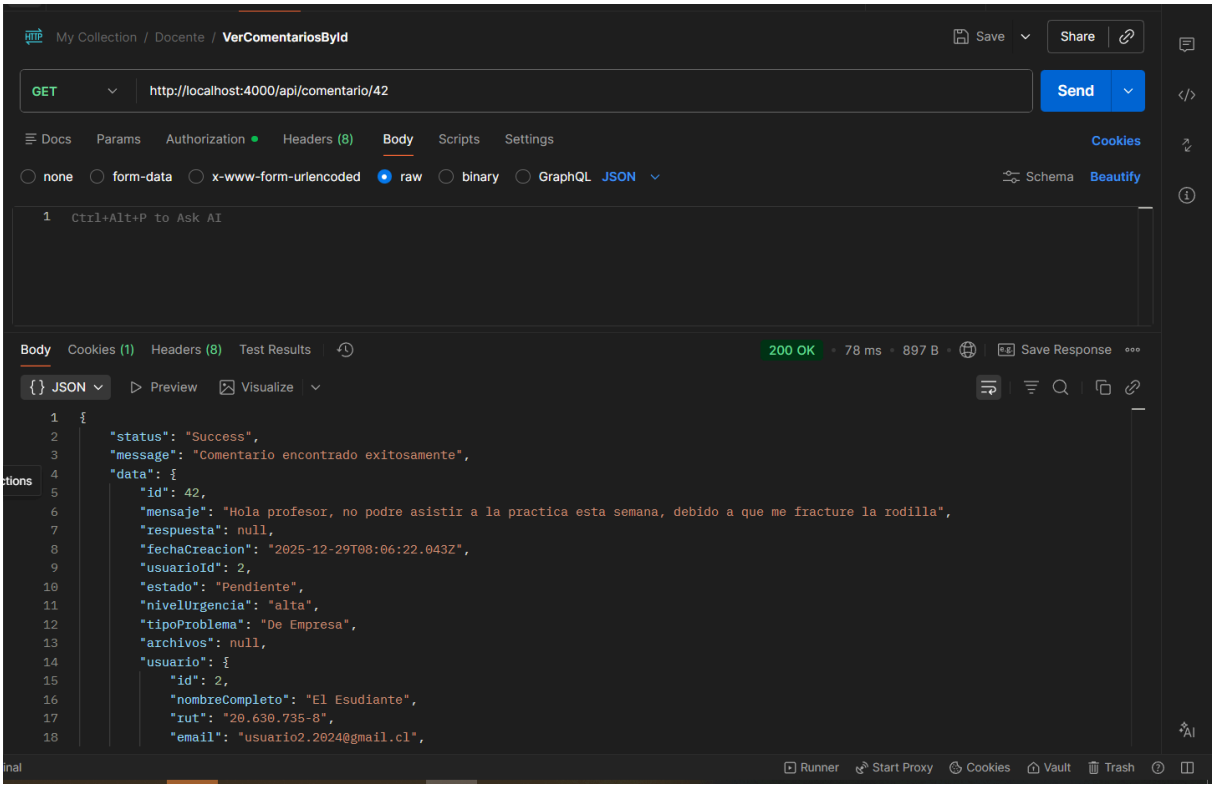

## Ver los comentarios de un Estudiante:



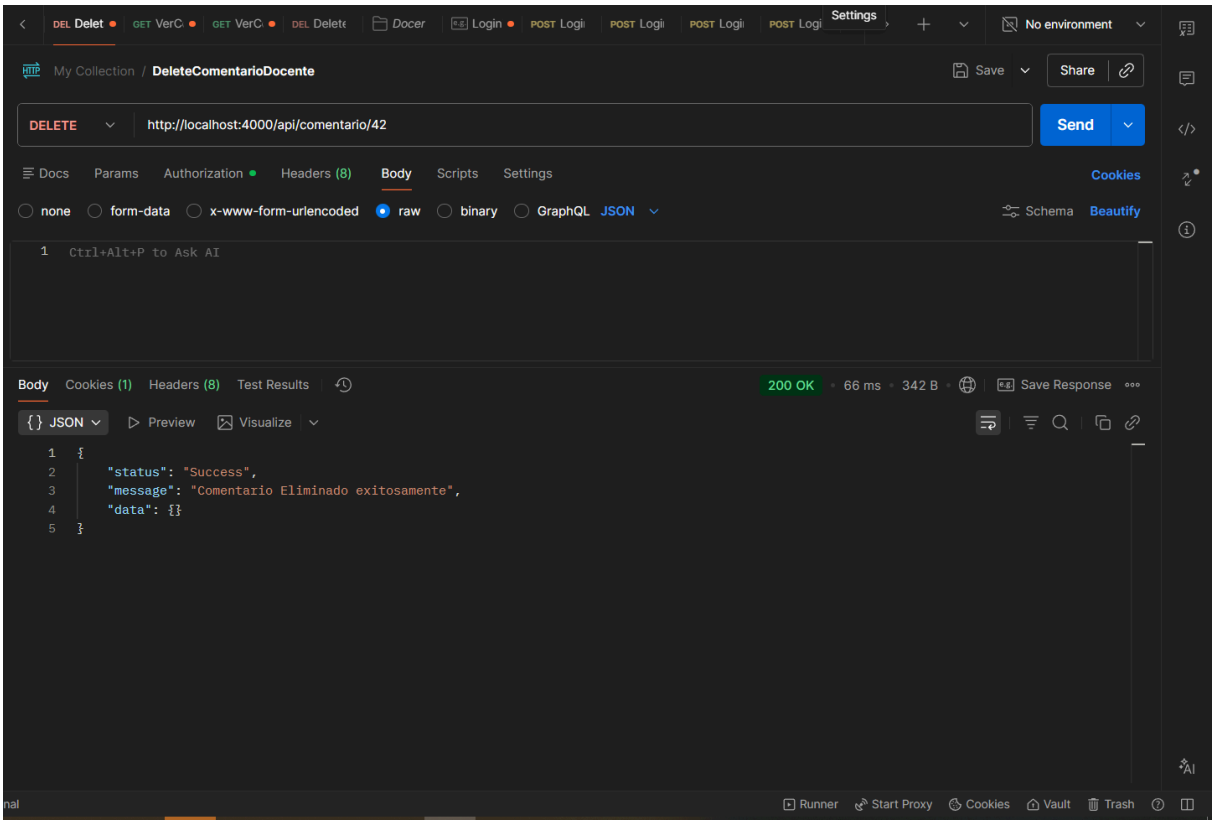
## Ver Comentario By IdEstudiante:



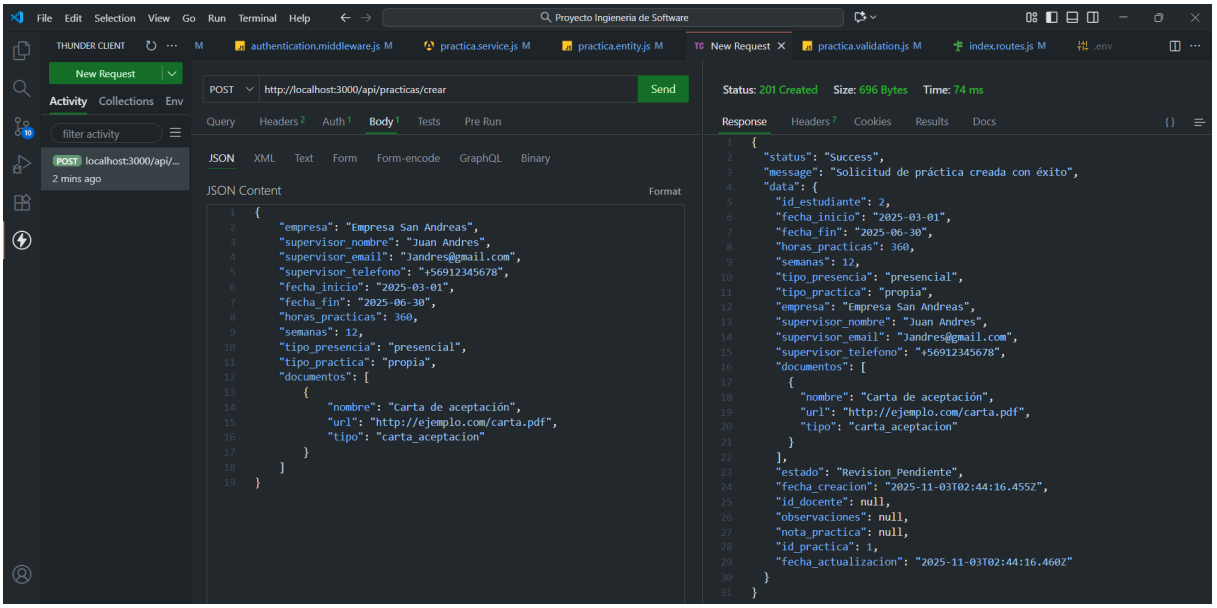

## Ver Comentarios por ID:




## Eliminar Comentarios por parte del docente:

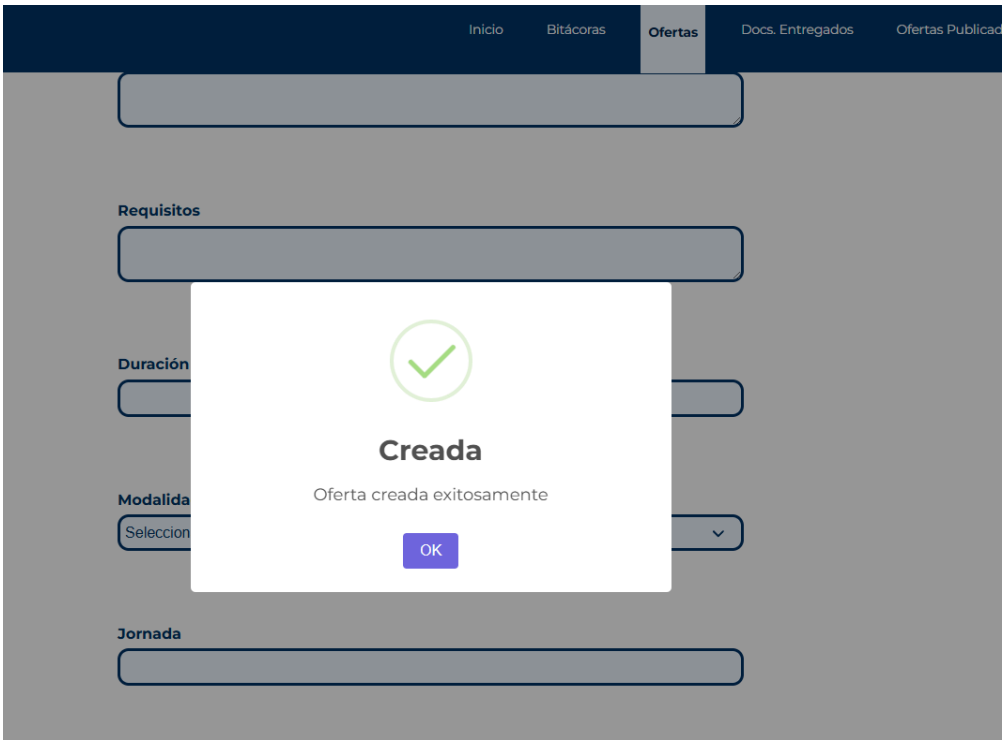


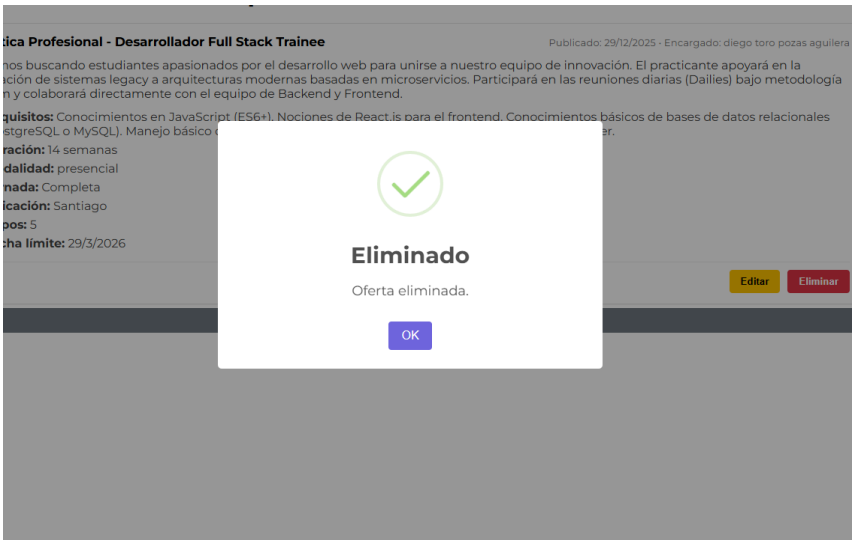

## Prueba de Creación de Practica:

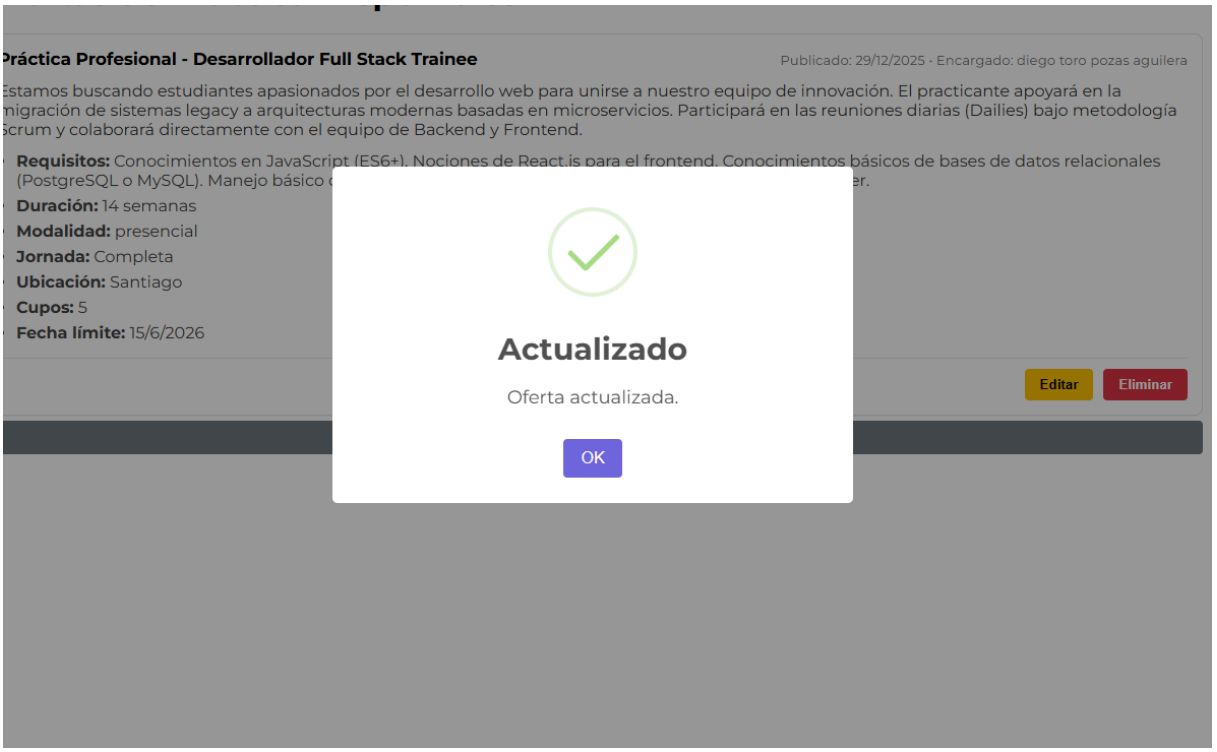



## subida Informes finales y evaluación de documentos

**Crear, Ver, Editar y Eliminar Ofertas de Práctica, además de estudiantes que postulen a la oferta de práctica, además del sistema de correos**



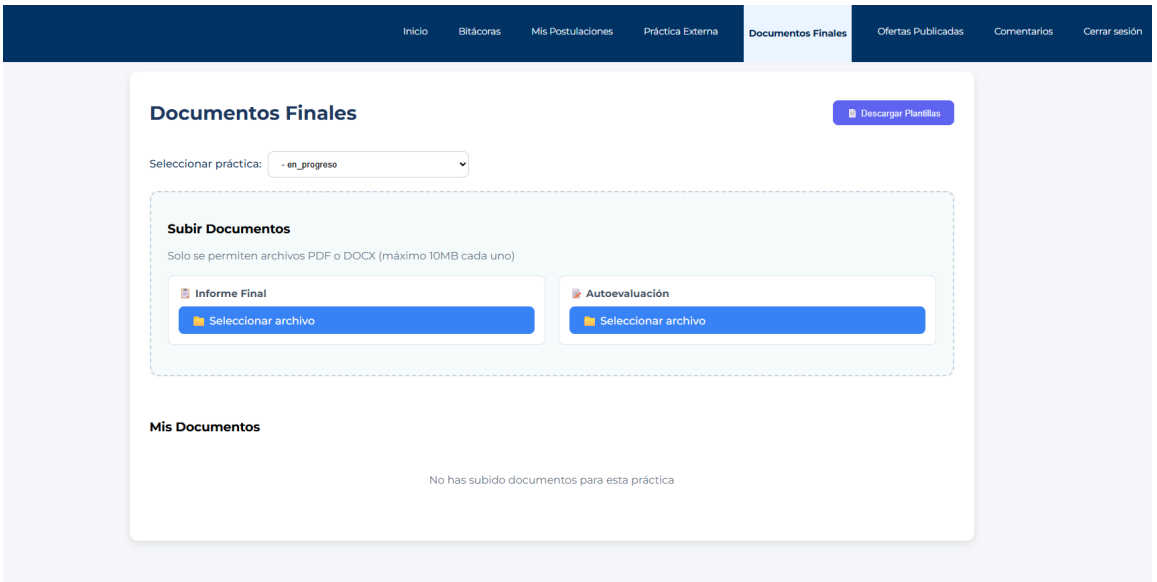



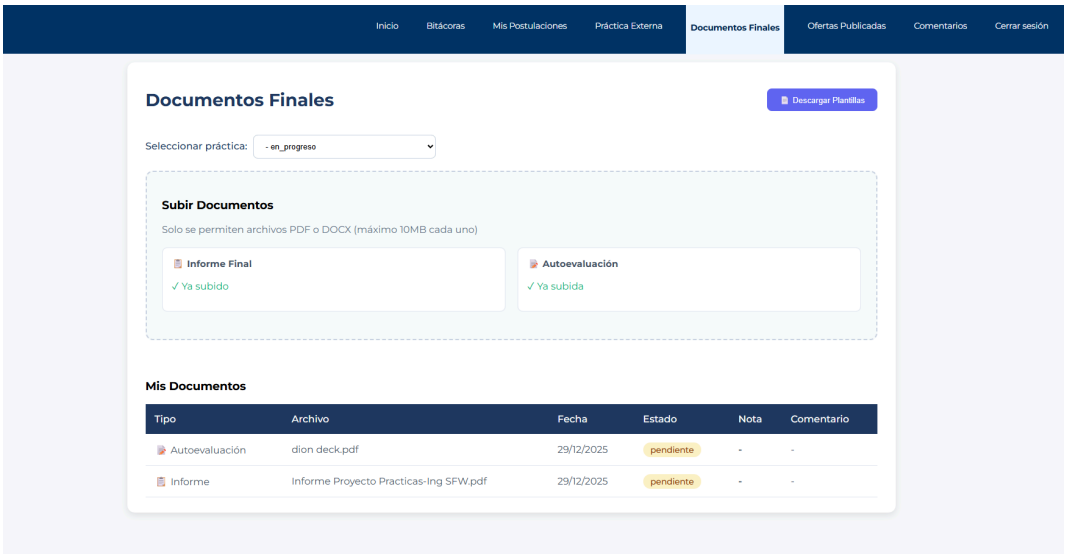

Práctica Profesional - Desarrollador Full Stack Trainee
Publicado: 29/12/2025 · Encargado: diego toro pozas aguilera
Estamos buscando estudiantes apasionados por el desarrollo web para unirse a nuestro equipo de innovación. El practicante apoyará en la migración de sistemas legacy a arquitecturas modernas basadas en microservicios. Participará en las reuniones diarias (Dailies) bajo metodología Scrum y colaborará directamente con el equipo de Backend y Frontend.
Requisitos: Conocimientos en JavaScript (ES6+). Nociones de React.js para el frontend. Conocimientos básicos de bases de datos relacionales (PostgreSQL o MySQL). Manejo básico
Duración: 14 semanas
Modalidad: presencial
Jornada: Completa
Ubicación: Santiago
Cupos: 5
Fecha límite: 15/6/2026
Editar
Eliminar
Actualizado
Oferta actualizada.
OK

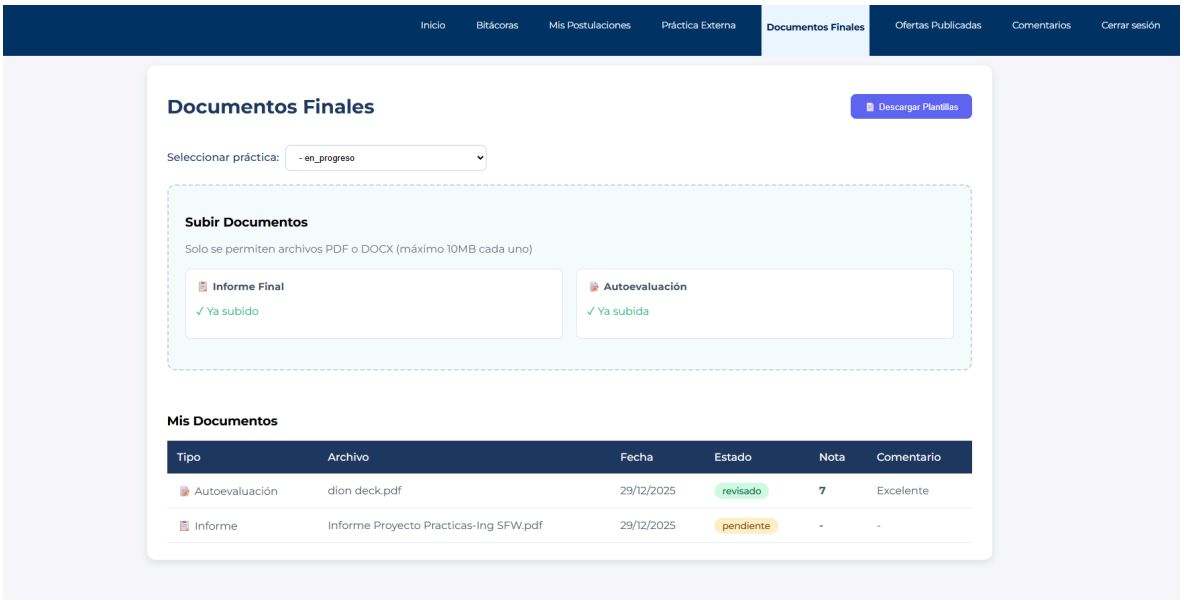

Ofertas de Práctica Disponibles
Práctica Profesional - Desarrollador Full Stack Trainee
Publicado: 29/12/2025 · Encargado: diego toro pozas aguilera
Estamos buscando estudiantes apasionados por el desarrollo web para unirse a nuestro equipo de innovación. El practicante apoyará en la migración de sistemas legacy a arquitecturas modernas basadas en microservicios. Participará en las reuniones diarias (Dailies) bajo metodología Scrum y colaborará directamente con el equipo de Backend y Frontend.
Requisitos: Conocimientos en JavaScript (ES6+). Nociones de React.js para el frontend. Conocimientos básicos de bases de datos relacionales (PostgreSQL o MySQL). Manejo básico de Git para control de versiones. Proactividad y ganas de aprender.
Duración: 14 semanas
Modalidad: presencial
Jornada: Completa
Ubicación: Santiago
Cupos: 5
Fecha límite: 15/6/2026
Editar
Eliminar
Ver Postulantes


Ofertas de Práctica Disponibles
Práctica Profesional - Desarrollador Full Stack Trainee
Publicado: 29/12/2025 · Encargado: diego toro pozas aguilera
Estamos buscando estudiantes apasionados por el desarrollo web para unirse a nuestro equipo de innovación. El practicante apoyará en la migración de sistemas legacy a arquitecturas modernas basadas en microservicios. Participará en las reuniones diarias (Dailies) bajo metodología Scrum y colaborará directamente con el equipo de Backend y Frontend.
Requisitos: Conocimientos en JavaScript (ES6+). Nociones de React.js para el frontend. Conocimientos básicos de bases de datos relacionales (PostgreSQL o MySQL). Manejo básico de Git para control de versiones. Proactividad y ganas de aprender.
Duración: 14 se
Modalidad: pr
Jornada: Com
Ubicación: San
Cupos: 4
Fecha límite:
Postulantes (1)
Nombre
RUT
Estado
Acciones
carlos williams gómez zavala
21.317.581-5
Pendiente
✓ Aceptar
✕ Rechazar
Cerrar
Eliminar

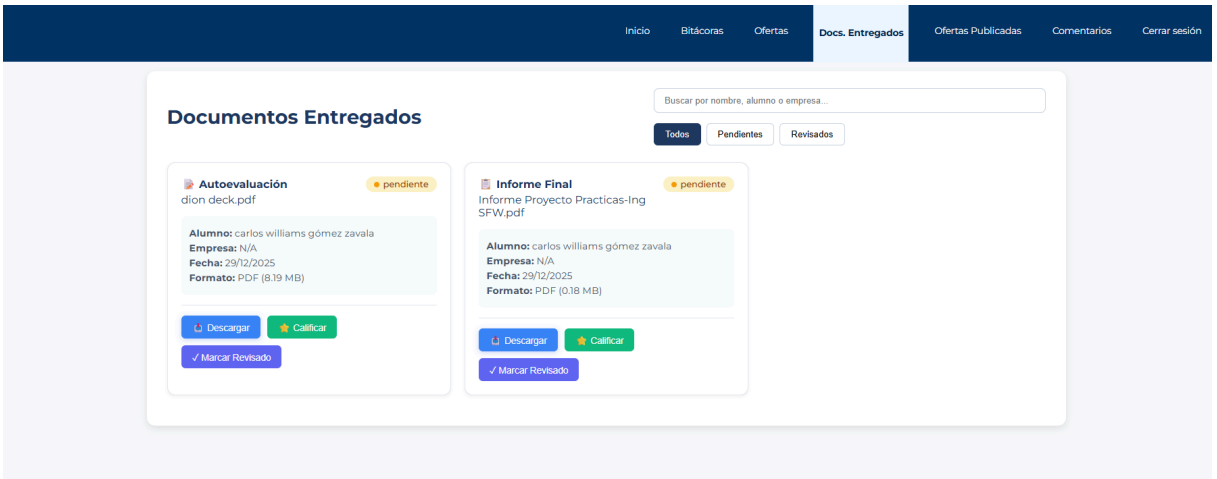

**prueba entrega documentos / vista estudiante**

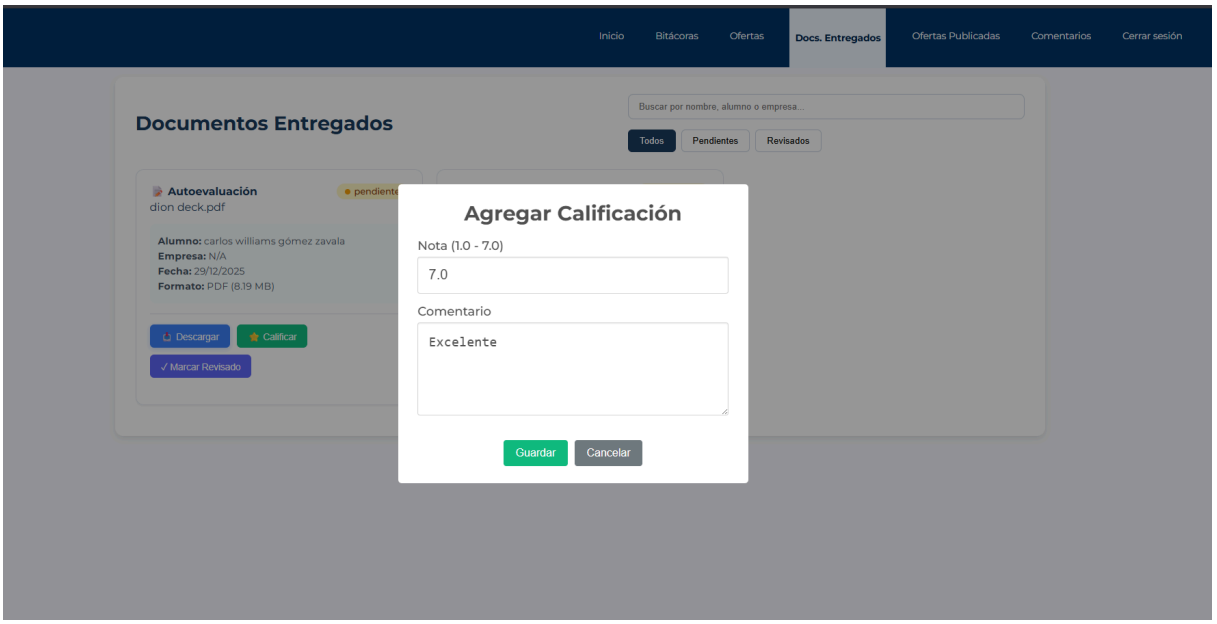


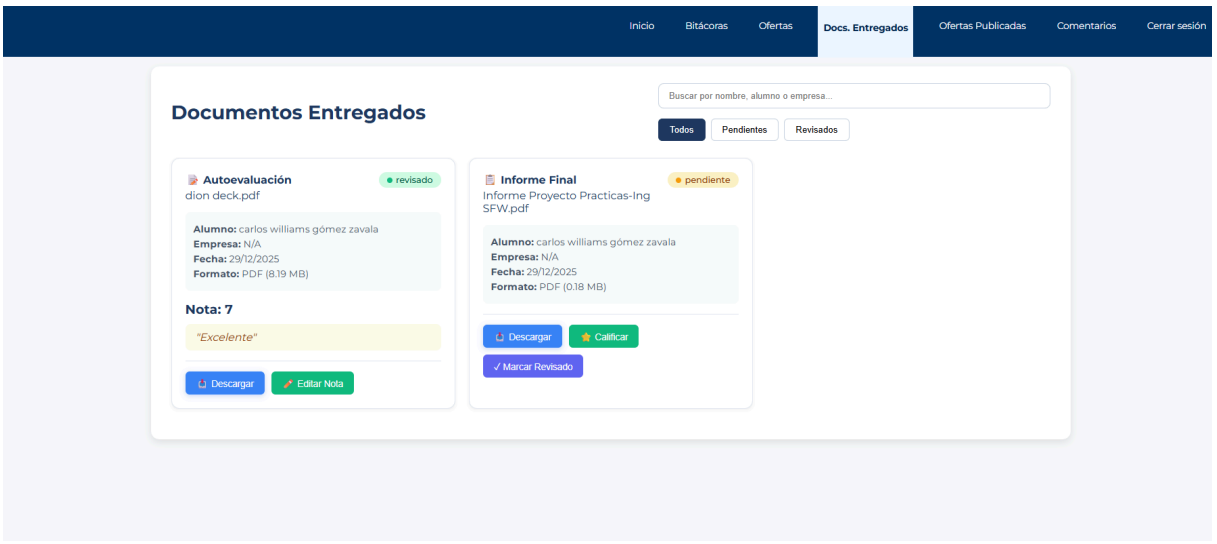



**prueba documentos entregados / vista docente**






### V.5.1. MODELO DE DATOS

El almacenamiento de la información se gestiona a través de una base de datos relacional **PostgreSQL**, estructurada bajo un esquema normalizado para garantizar la integridad referencial. La interacción con la base de datos se realiza mediante el ORM **TypeORM**, lo que permite manipular los datos utilizando clases y objetos en lugar de consultas SQL crudas.

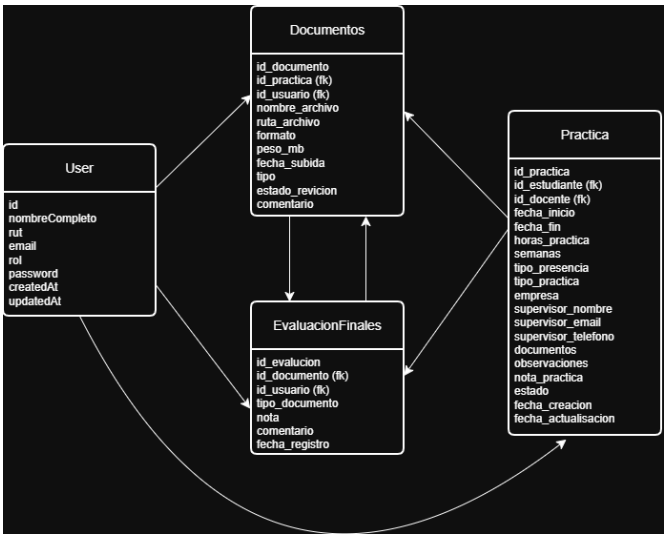
**Entidades Principales:**

1. **Usuarios (users):** Tabla maestra que almacena las credenciales, roles (Estudiante, Docente, Admin) y datos personales (RUT, Nombre).
2. **Prácticas (practicas):** Entidad central que registra la solicitud del estudiante.

    - *Atributos clave:* id_estudiante, empresa, estado (Enum: Revision_Pendiente, Aprobada, etc.), tipo_practica (Propia/Publicada) y el array de documentos (documentos).
3. **Ofertas (ofertas):** Almacena las vacantes publicadas por los docentes.
4. **Bitácoras (bitácoras):** Registros semanales vinculados a una práctica aprobada, donde se detallan las actividades y horas realizadas.

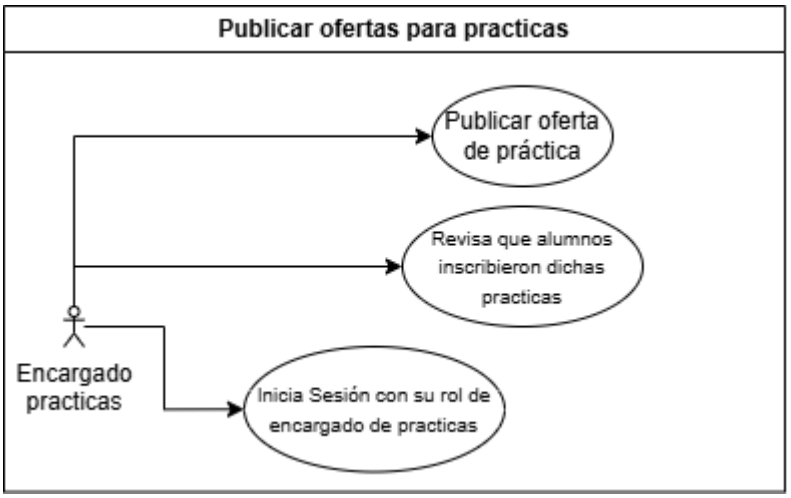
**Relaciones:**

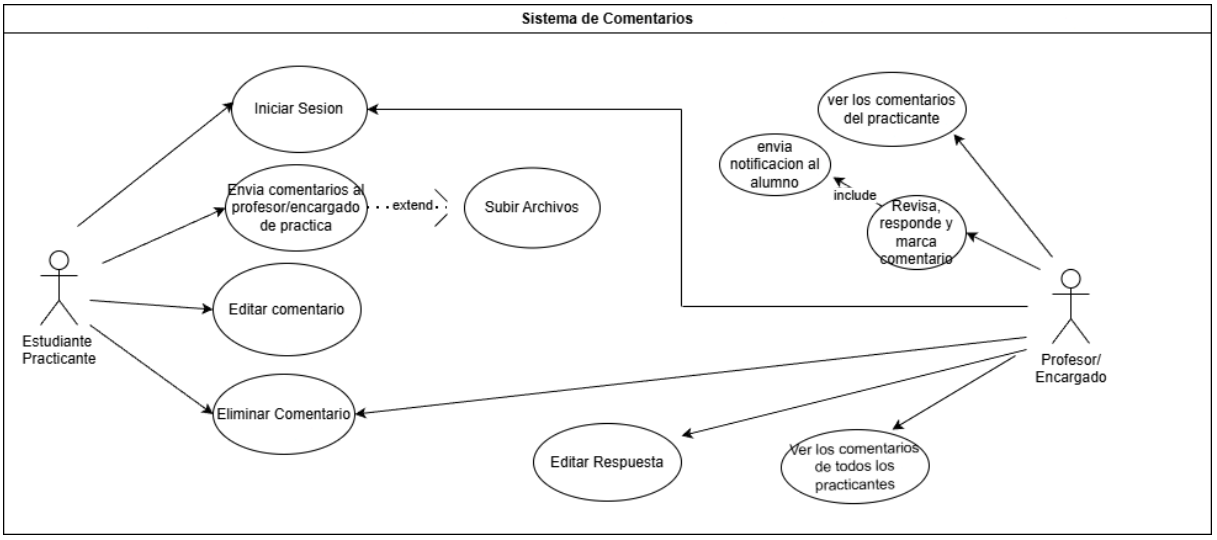
- Un **Usuario (Estudiante)** tiene una relación de *uno a muchos* con **Prácticas** (historial de postulaciones).
- Una **Práctica** tiene una relación de *uno a muchos* con **Bitácoras** (una práctica genera múltiples reportes semanales).
- Un **Usuario (Docente)** tiene una relación de *uno a muchos* con **Ofertas** creadas.
- MR relacion documentos - evaluación



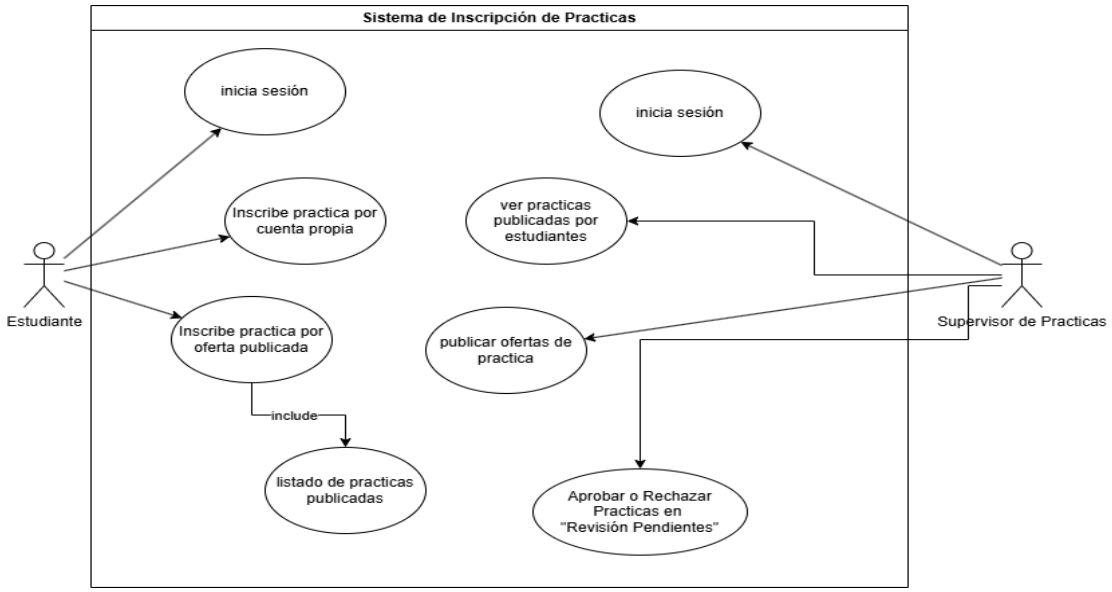

## V.5.2. CASOS DE USO

A continuación, se detallan los diagramas de comportamiento del sistema, especificando las interacciones entre los actores (Estudiante, Docente, Administrador) y los procesos del software.

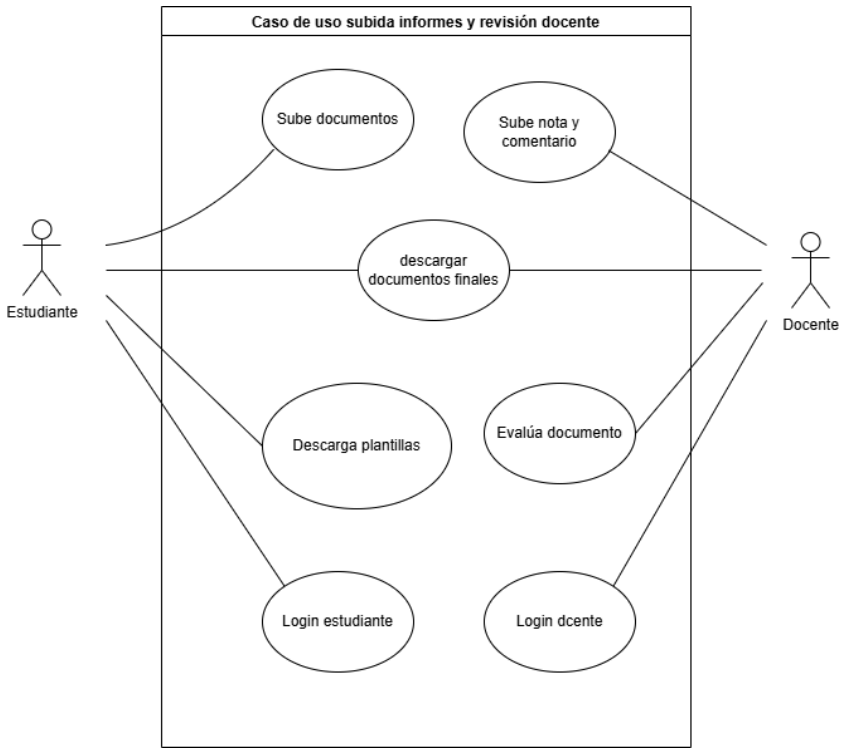



## Caso de Uso de Inscripcion de prácticas:




## Caso de uso: Entrega de informes y Revisión




## CU: sube documentos

Precondiciones: usuario debe estar logueado, debe tener rol estudiante
Include:

Flujo de acciones:
El estudiante selecciona el informe y autoevaluación que quiera subir, y sube los archivos a la plataforma.
Sistema registra la subida de ambos archivos y solo si la práctica está finalizada
Postcondiciones: se guardan los archivos en la carpeta Uploads/documentos, también le agrega caracteres para evitar archivos repetidos.

**CU: Descarga plantillas**
Precondiciones:usuario debe estar logueado, debe tener rol estudiante
Include:
Flujo de acciones: estudiante descarga documentos
Postcondiciones: se descargan documentos con el link de adeca

**CU: Sube nota y comentario**
Precondiciones: usuario debe estar logueado, debe tener rol docente y alumno que está enlazado debe haber entregado documentos finales, y tienen que haber una nota ingresada en la web
Include:
Flujo de acciones:
el estudiante debe haber entregado y subido con antelación sus informes, el docente puede descargar los archivos que el estudiante a entregar, debe agregar una nota en la página y registrarla, esta nota es modificable
Postcondiciones: Se guardan los datos ingresados en la base de datos.

**CU: Evalúa documento**
Precondiciones: el estudiante debe haber entregado y subido con antelación sus informes, el docente puede descargar los archivos que el estudiante a entregar
include:
Flujo de acciones: debe agregar una nota en la página y el comentario es opcional
postcondiciones:

**CU: descargar documentos finales**
Precondiciones: usuario debe estar logueado, debe tener rol docente o alumno, alumno debe haber entregado documentos finales
Include:
Flujo de acciones: el sistema busca los documentos en la base de datos y en la carpeta y genera un link para acceder al archivo
postcondiciones: se señala si hubo un error por la página o se lanza una descarga del archivo

**V.5.3. DISEÑO INTERFAZ Y NAVEGACIÓN**

El diseño de la interfaz de usuario (UI) se ha desarrollado bajo el concepto de

**Single Page Application (SPA)** utilizando la biblioteca **React.js**. Esto permite una navegación fluida sin recargas completas de la página, mejorando la experiencia de usuario (UX).

**Arquitectura de Navegación:** La navegación se gestiona mediante react-router-dom y se estructura en base a roles, utilizando un componente de seguridad denominado ProtectedRoute:

1. **Rutas Públicas:**
    - /auth: Login y acceso al sistema.
    - /register: Registro de nuevos estudiantes.
2. **Rutas Privadas (Estudiante):**
    - /home: Panel de control general.
    - /crear-practica: Formulario dinámico para inscripción de práctica propia y subida de archivos.
    - /bitacoras: Gestión de reportes semanales.
3. **Rutas Privadas (Docente/Admin):**
    - /users: Gestión de usuarios (Solo Admin).
    - /crear-oferta: Publicación de vacantes (Solo Docente).

**Diseño Visual:** Se utiliza un diseño responsivo mediante CSS modular, asegurando que los formularios de inscripción y las tablas de visualización sean accesibles tanto desde dispositivos de escritorio como móviles. Se implementan alertas visuales (*SweetAlert2*) para dar feedback inmediato al usuario sobre el éxito o fracaso de sus operaciones (ej: "Práctica registrada con éxito").

### V.5.4. SISTEMA WEB - BACKEND

La lógica del servidor se ha implementado utilizando **Node.js** con el framework **Express**, siguiendo el patrón de arquitectura **MVC (Modelo-Vista-Controlador)** para asegurar la separación de responsabilidades y la mantenibilidad del código.

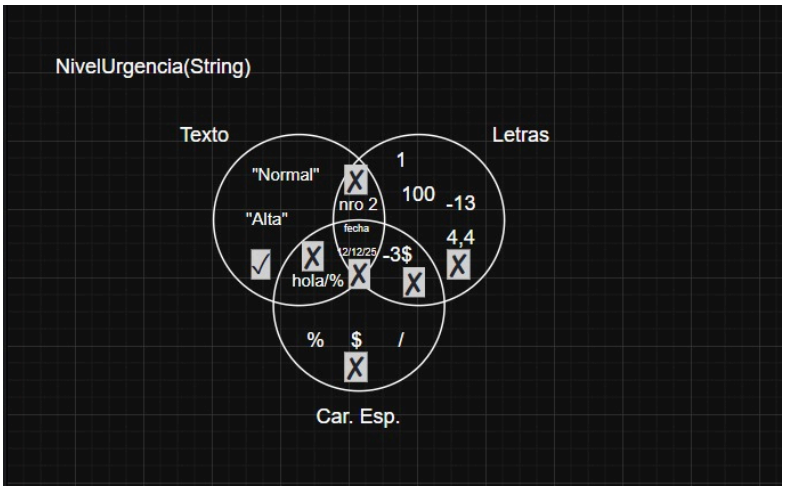
**Componentes de la Arquitectura:**

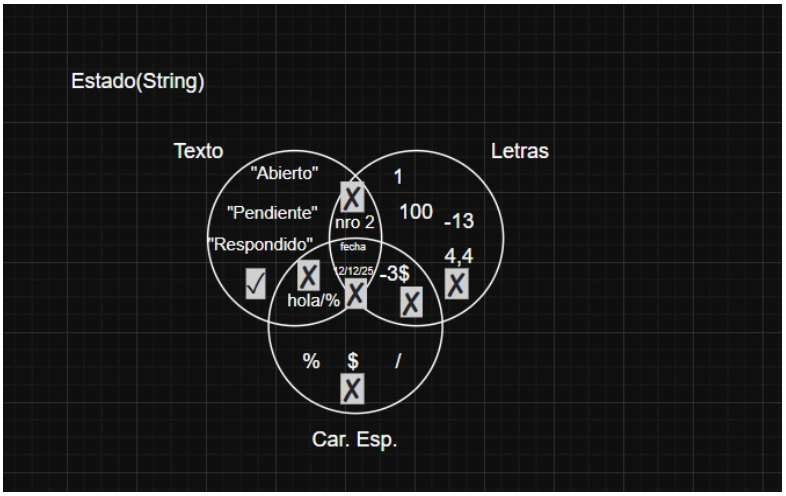
1. **Rutas (Routes):** Definen los puntos de entrada de la API (Endpoints). Se encargan de dirigir las peticiones HTTP (GET, POST, PUT, DELETE) hacia el controlador correspondiente.
    - *Ejemplo:* La ruta /api/prácticas/crear intercepta la petición y aplica primero el middleware de autenticación (authenticateToken) y luego el de subida de archivos (upload.array).
2. **Controladores (Controllers):** Manejan la lógica de entrada y salida. Reciben los datos del req.body o req.files, validan la información básica y llaman a los servicios. Se encargan de enviar las respuestas estandarizadas al cliente (JSON con códigos HTTP 200, 400, 500).

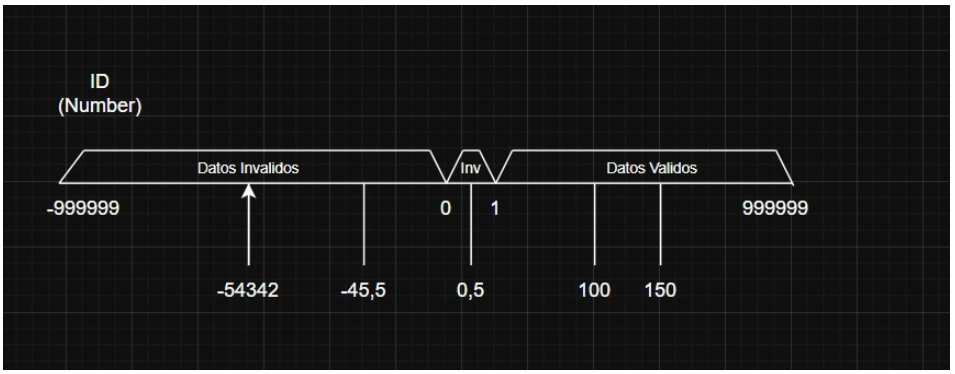
3. **Servicios (Services):** Contienen la lógica de negocio pura. Interactúan con la base de datos a través de los repositorios de TypeORM. Aquí se ejecutan las validaciones complejas, como verificar que un alumno no tenga dos prácticas activas simultáneamente o calcular el estado inicial de una solicitud (Revision_Pendiente).
4. **Seguridad y Middlewares:**
    - **Autenticación:** Uso de JSON Web Tokens (JWT) para validar la sesión en cada petición protegida.
    - **Manejo de Archivos:** Implementación de Multer para la gestión de datos multipart/form-data, permitiendo recibir archivos binarios (PDF) y almacenarlos en el servidor local, guardando solo la referencia (ruta) en la base de datos.

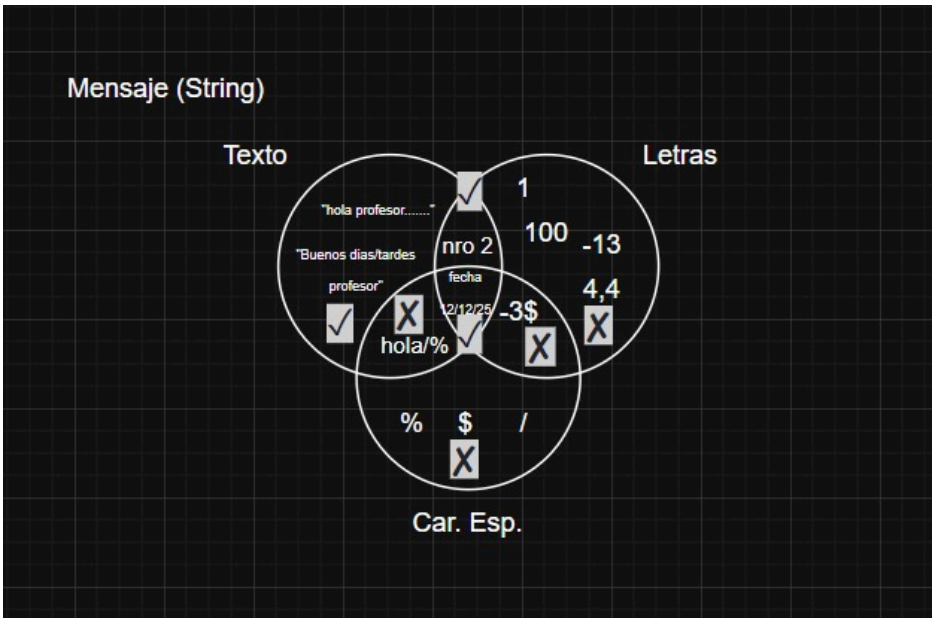
### V.5.5. DIAGRAMA CAJA NEGRA - CASOS DE PRUEBAS

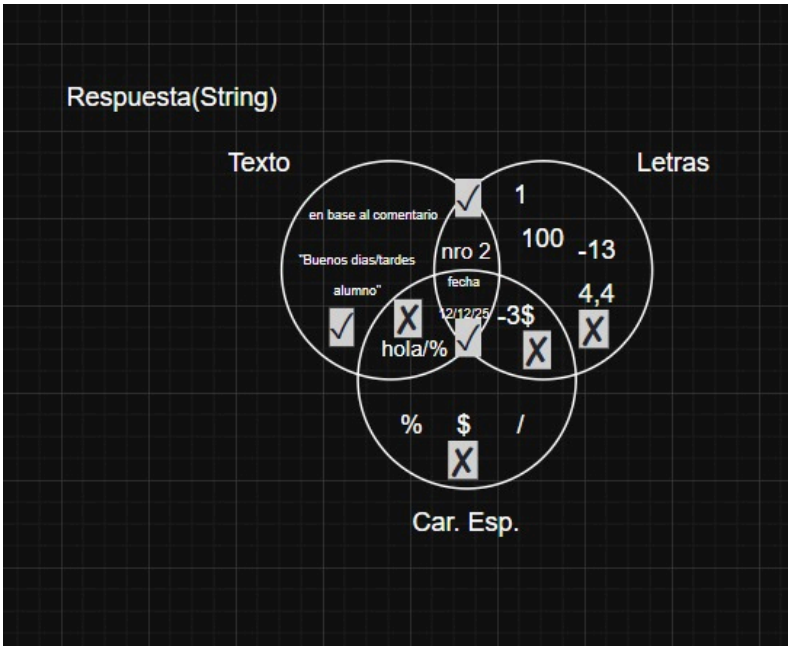
**Caoso de prueba comentarios**



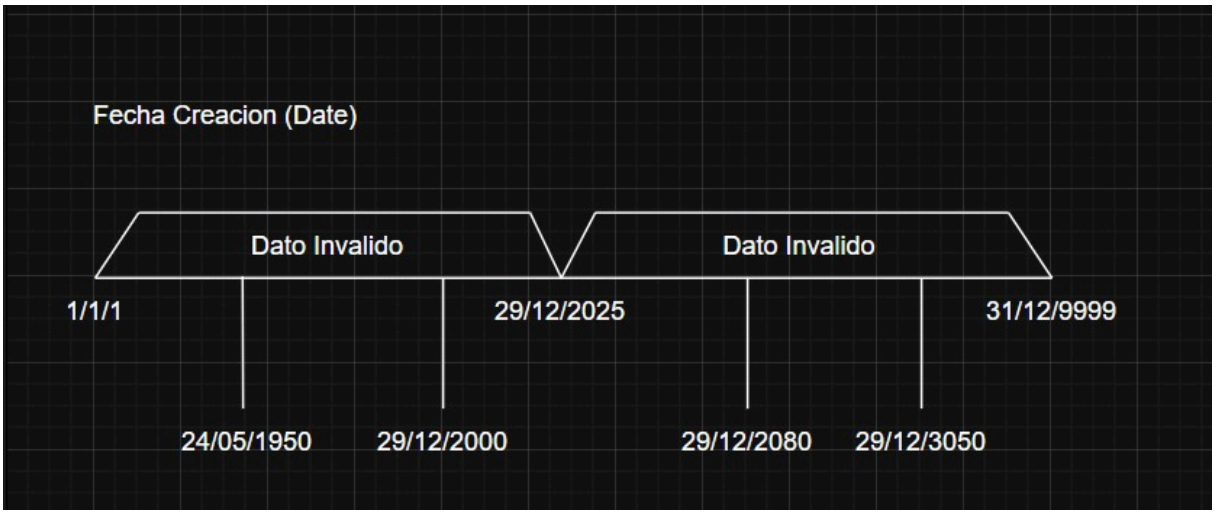




ID
(Number)
Datos Invalidos
Inv
Datos Validos
-999999
0
1
999999
-54342
-45,5
0,5
100
150

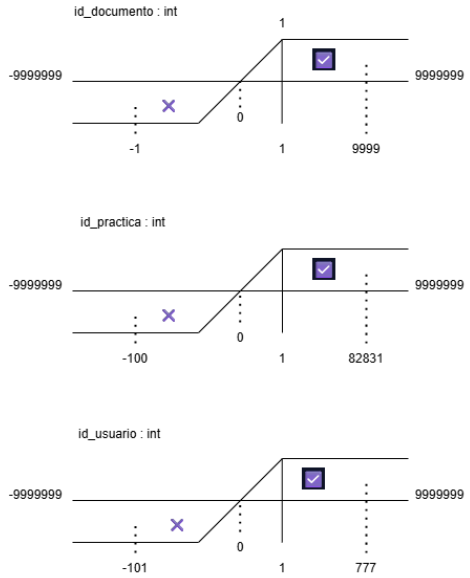

Mensaje (String)
Texto
Letras
"hola profesor....."
"Buenos dias/tardes
profesor"
nro 2
fecha
12/12/25
hola/%
1
100
-13
4,4
-3$
%
$
/
Car. Esp.

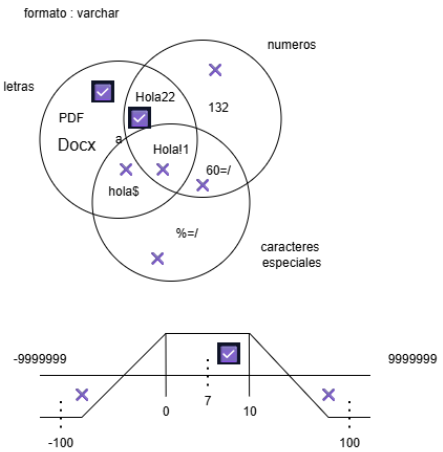

Respuesta(String)
Texto
Letras
en base al comentario
"Buenos dias/tardes
alumno"
nro 2
fecha
12/12/25
hola/%
1
100
-13
4,4
-3$
%
$
/
Car. Esp.

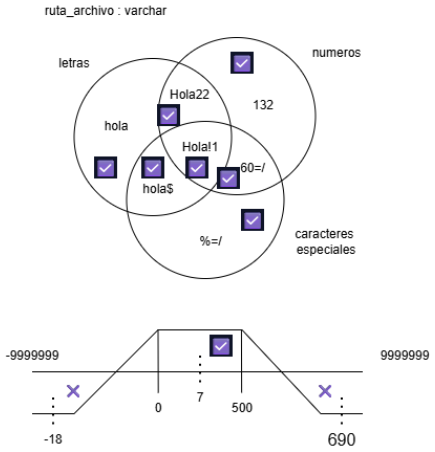

IdUsuario (Number)
Datos Invalidos
inv
Dato Invalido
-99999
0
1
1000
99999
-420
-100,5
0,5
700
3000
95674

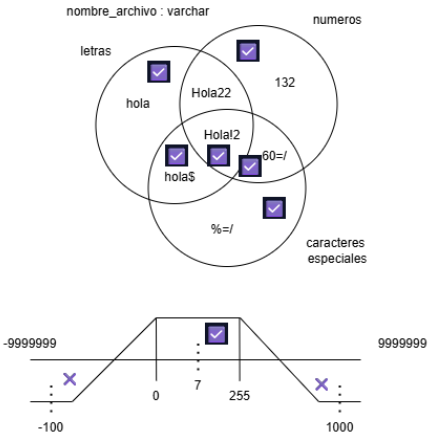

Fecha Creacion (Date)
Dato Invalido
Dato Invalido
1/1/1
29/12/2025
31/12/9999
24/05/1950
29/12/2000
29/12/2080
29/12/3050

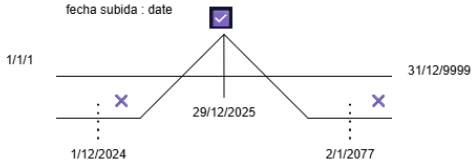
# Caso de pruebas documentos finales

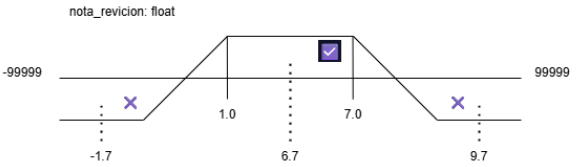


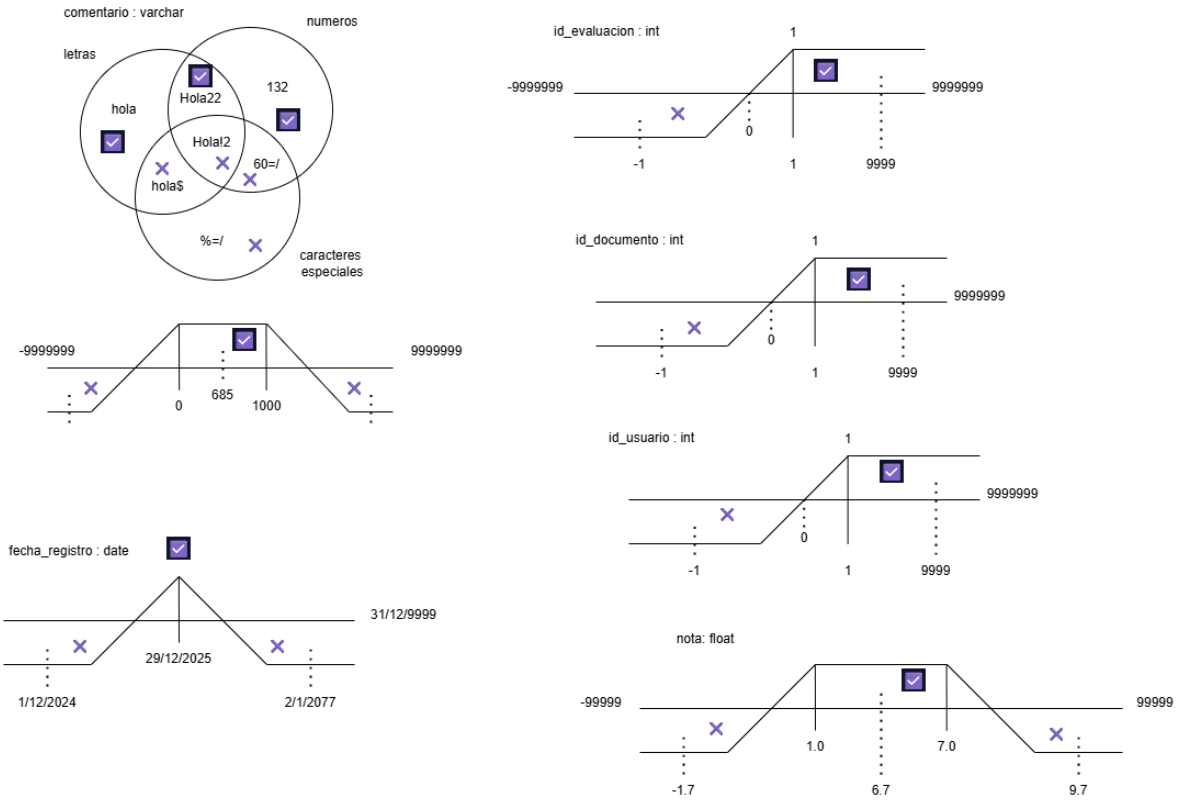



Caso de prueba evaluación




## V.4. SERVIDORES

**V.4.1. SERVIDOR CON EXPRESS** Se utiliza Express.js como framework de backend para manejar las rutas, middlewares de autorización (isAdmin, isEstudiante) y controladores.

## V.5. SISTEMA WEB DE GESTIÓN DE PRÁCTICAS

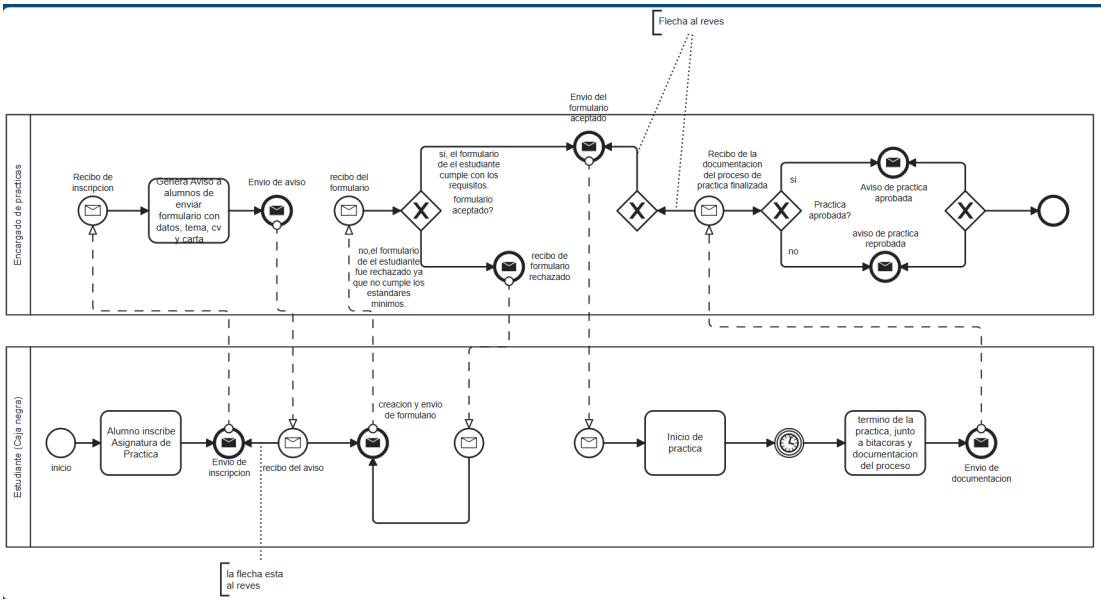
**V.5.4. BACKEND** Arquitectura en capas:

1. **Rutas:** Definición de endpoints.
2. **Middlewares:** Validación de JWT y Roles.
3. **Controladores:** Lógica de negocio (ej: lógica de descontar cupos).
4. **Servicios:** Comunicación con la BD (Repositories).

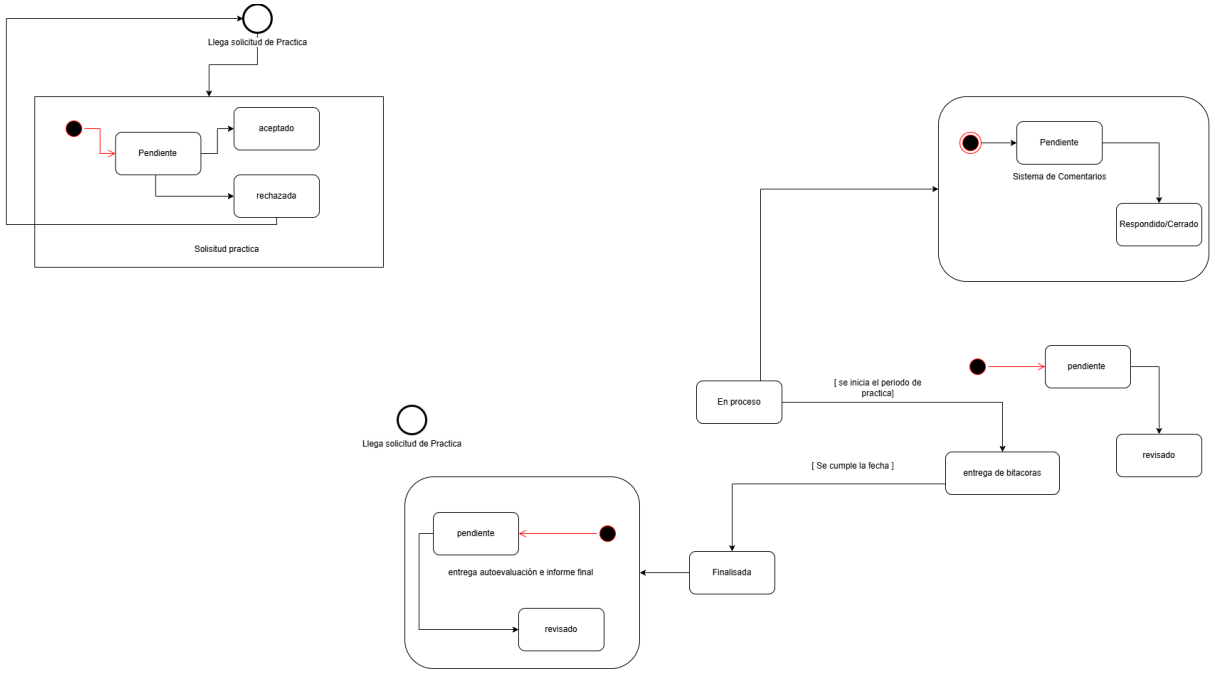
### V.5.5. FRONTEND

Desarrollado en React. Uso de **Hooks** personalizados (useUsers, useOfertas) para separar la lógica de la vista. Implementación de **Rutas Protegidas** para evitar que estudiantes entren a paneles de administración.

## V.5.6. BPMN




## V.5.7. DIAGRAMA DE ESTADOS




# Capítulo VI. EVALUACIONES Y RESULTADOS

Aquí puedes mencionar las pruebas que acabamos de hacer:

- **Prueba de Postulación:** Se verificó que al postular, el cupo disminuye en la base de datos y en la vista sin recargar (React State).
- **Prueba de Unicidad:** El sistema impide postular dos veces a la misma

oferta.
- **Prueba de Notificación:** Se comprobó la recepción del correo vía Nodemailer.
- **Prueba de Roles:** Un estudiante no puede ver el botón "Eliminar Oferta", solo "Postular".

# Capítulo VII. PLAN DE IMPLANTACIÓN

Este capítulo describe cómo el sistema pasará del entorno de desarrollo (tu computador) al entorno real de la institución, asegurando que los usuarios sepan usarlo.

## VII.1. PLAN DE CAPACITACIÓN/ENTRENAMIENTO

Para asegurar la correcta adopción del "Sistema de Gestión de Prácticas", se ha diseñado un plan de capacitación diferenciado según el perfil de usuario:

- **Administradores y Jefes de Carrera:**
  - **Modalidad:** Taller presencial práctico (2 horas).
  - **Contenidos:** Gestión de usuarios (CRUD), restauración de contraseñas, supervisión global de ofertas y generación de reportes. ○ **Material de Apoyo:** Manual técnico en PDF y acceso a un usuario "Super Admin" de prueba.
- **Docentes Encargados:**
  - **Modalidad:** Webinar sincrónico o cápsulas de video (tutoriales). ○ **Contenidos:** Cómo crear una oferta de práctica, revisar la lista de postulantes, descargar datos de alumnos y utilizar el módulo de comentarios para feedback.
  - **Objetivo:** Que el docente pueda publicar una oferta y gestionar sus cupos sin asistencia técnica.
- **Estudiantes:**
  - **Modalidad:** Autoinstrucción digital.
  - **Contenidos:** Infografía paso a paso enviada al correo institucional y un video corto ("Reel" o "Short") en las redes sociales de la carrera explicando cómo postular, subir bitácoras y registrar prácticas externas.
  - **Enfoque:** Uso intuitivo de la interfaz, resaltando la importancia de mantener su perfil actualizado.

### VII.1.1. ESTRATEGIA DE IMPLANTACIÓN

La puesta en marcha del sistema se realizará bajo una estrategia de **"Implantación en Fases"** (Phased Rollout) para minimizar riesgos:

**Fase 1: Piloto Controlado (Semana 1-2):**

- Se desplegará el sistema solo para una carrera específica o un grupo pequeño de docentes "Beta Testers".
- **Objetivo:** Detectar errores no vistos en desarrollo y validar la usabilidad del flujo de postulación.

**Fase 2: Ajustes y Feedback (Semana 3):**

- Se recopilarán los comentarios del grupo piloto para corregir bugs críticos o mejorar textos confusos en la interfaz.

**Fase 3: Masificación (Semana 4 en adelante):**

- Apertura del sistema a todas las carreras planificadas.
- Migración de datos antiguos (si aplica) y apagado progresivo de los métodos anteriores (correos masivos, planillas Excel).

## Capítulo VIII. CONCLUSIÓN DEL PROYECTO

El desarrollo del **Sistema de Gestión de Prácticas Profesionales** ha permitido dar solución a una problemática administrativa latente en la institución, modernizando un proceso que, hasta ahora, se caracterizaba por la dispersión de la información y la gestión manual.

A través de la implementación de una arquitectura de software robusta basada en el stack **MERN** (en este caso usando PostgreSQL con Node.js y React), se logró cumplir con los objetivos planteados: centralizar las ofertas, automatizar el control de cupos y transparentar el proceso de postulación para los estudiantes.

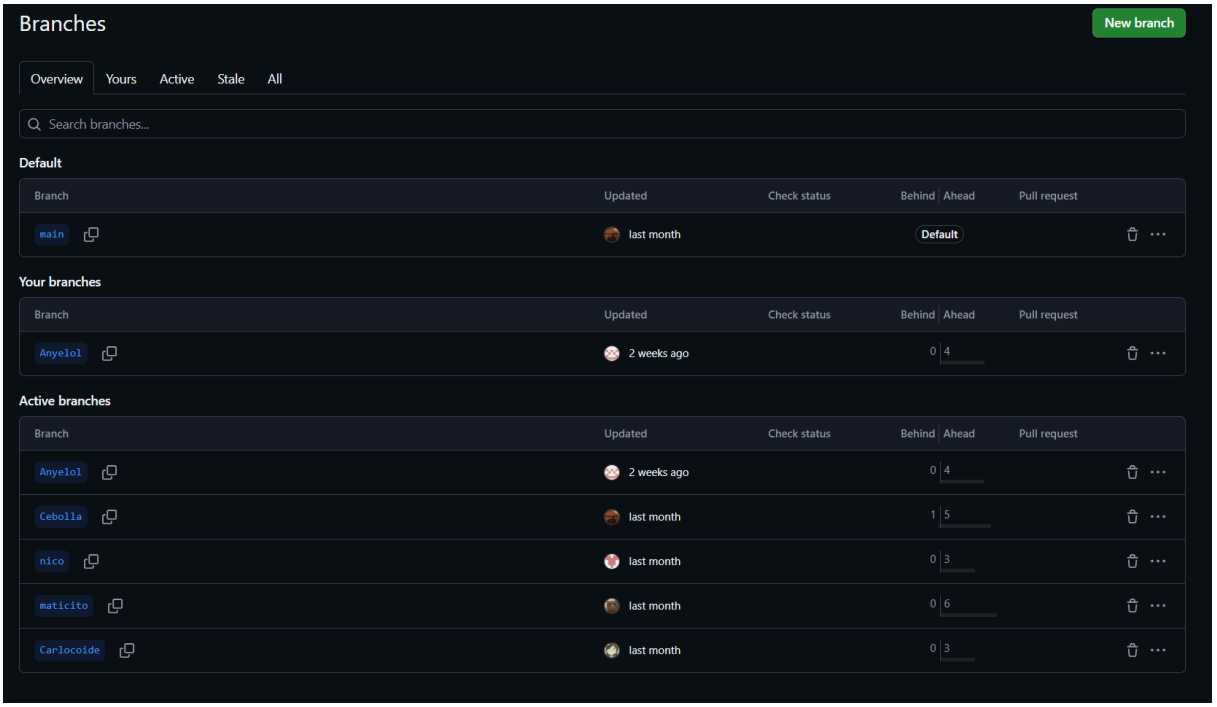
Técnicamente, el proyecto demostró la viabilidad de integrar funcionalidades complejas como el envío de correos transaccionales (Nodemailer), la seguridad mediante tokens (JWT) y la gestión de relaciones de base de datos (TypeORM) en una interfaz amigable y responsiva.

En conclusión, este software no solo optimiza tiempos operativos para los docentes, sino que también entrega a los estudiantes una herramienta formal y digna para gestionar uno de los hitos más importantes de su formación profesional. El sistema queda preparado para escalar y recibir futuros módulos, consolidándose como un activo digital valioso para la comunidad académica.

## CAPÍTULO IX. REFERENCIAS

1. React Documentation. (2024). Quick Start. Recuperado de: https://react.dev/
2. Node.js Foundation. (2024). Node.js Documentation. Recuperado de: https://nodejs.org/en/docs/
3. TypeORM. (2024). TypeORM Documentation: Relational Database Object Mapping. Recuperado de: https://typeorm.io/
4. Express.js. (2024). Fast, unopinionated, minimalist web framework for Node.js. Recuperado de: https://expressjs.com/
5. Nodemailer. (2024). Send emails from Node.js. Recuperado de: https://nodemailer.com/
6. JWT.io. (2024). Introduction to JSON Web Tokens. Recuperado de: https://jwt.io/introduction
7. PostgreSQL. (2024). PostgreSQL: The World's Most Advanced Open Source Relational Database. Recuperado de: https://www.postgresql.org/docs/

Aclaración de doble subisa de proyecto por errores




aparte del repositorio git actual teniamos este:
https://github.com/DonCebollin/Proyecto_Ingerieria_de_Software.git