Desarrollo de sistema de gestión para actas de evaluación de trabajo de grado en ingeniería

Juan Fernando Vergara Londoño

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Ingeniería de Sistemas y Computación
Cali
2026

Desarrollo de sistema de gestión para actas de evaluación de trabajo de grado en ingeniería

Juan Fernando Vergara Londoño

Trabajo de grado

*D*irector: Dr. Gerardo Mauricio Sarria

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Ingeniería de Sistemas y Computación
Cali
2026

Santiago de Cali, Enero 15 de 2026

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo M. Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo

Por medio de la presente me permito informarle que el estudiante de Ingeniería de Sistemas y Computación Juan Fernando Vergara Londoño (cod: 8932280) trabaja bajo mi dirección en el proyecto de grado titulado **"Desarrollo de sistema de gestión para actas de evaluación de trabajo de grado"**.

Atentamente,

Dr. Gerardo Mauricio Sarria
Director de trabajo de grado
Pontificia Universidad Javeriana Cali

Santiago de Cali, Enero 15 de 2026

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo M. Sarria
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo

Me permito presentar a su consideración el proyecto de grado titulado: **"Desarrollo de sistema de gestión para actas de evaluación de trabajo de grado"**, con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Atentamente,

Juan Fernando Vergara Londoño
Código: 8932280

Índice

1. Introducción **11**

2. Descripción del Problema **12**
2.1. Planteamiento del Problema 12
2.1.1. Formulación 12
2.1.2. Sistematización 12
2.2. Objetivos 13
2.2.1. Objetivo General 13
2.2.2. Objetivos Específicos 13

3. Marco de Referencia **14**
3.1. Marco Teórico 14
3.1.1. La ingeniería de software 15
3.1.2. El ciclo de vida del software 15
3.1.3. Patrón de arquitectura MVC 16
3.1.4. FastApi 16
3.1.5. Node.js 16
3.1.6. MySQL 16
3.1.7. Sequelize 17
3.1.8. phpMyAdmin 17
3.1.9. Trabajos Relacionados en el Ámbito Estudiantil 17
3.1.10. Trabajos Relacionados en el Ámbito Empresarial 18

4. Requisitos del sistema de gestión **19**
4.1. Metodología de Obtención de Requisitos 19
4.2. Diagrama de Contexto 20
4.3. Diagrama de actividades 21
4.4. Diagrama de casos de uso 24
4.5. Requisitos funcionales 25
4.6. Requisitos no funcionales 27
4.7. Limitaciones 27

5. Diseño del sistema de gestión **28**
5.1. Diseño general de la arquitectura del sistema 28
5.1.1. Diseño propuesto de arquitectura para el sistema 29
5.2. Diseño de la base de datos 30
5.2.1. Selección de la base de datos 30
5.2.2. Modelo de la base de datos 31

5.3. Diseño de las vistas 32
5.3.1. Selección de tecnologías para las vistas 32
5.3.2. Mock-ups para las vistas 33

6. Desarrollo del sistema de gestión 40
6.1. Selección de la tecnología para el prototipo 40
6.1.1. Docker en el desarrollo del prototipo 40
6.2. Configuraciones iniciales 41
6.2.1. Backend: 41
6.2.2. Frontend: 42
6.2.3. Docker: 43
6.2.4. Base de datos: 44
6.3. Desarrollo de los modelos 45
6.4. Definición de las rutas 46
6.5. Desarrollo de los controladores 48
6.5.1. Controlador de autenticación 48
6.5.2. Controlador de actas 49
6.6. Desarrollo de las vistas 50

7. Pruebas del sistema de gestión 53
7.1. Pruebas de caja negra 53
7.2. Pruebas con el usuario 54

8. Conclusiones 55

9. Trabajo futuro 56

10. Bibliografía 57

11. Anexos 59

Índice de figuras

1. Acta de evaluación referencia (Estudiante Juan Esteban Amaya) . . . 20
2. Diagrama de Contexto . . . 21
3. Diagrama de Actividades - Proceso general del sistema . . . 22
4. Diagrama de Actividades - Acceso a funcionalidades por roles . . . 22
5. Diagrama de casos de uso . . . 24
6. Diagrama general de la arquitectura del sistema . . . 28
7. Diagrama propuesto para la arquitectura del sistema . . . 29
8. Modelo base de datos . . . 31
9. Diseño Inicio de sesión . . . 33
10. Diseño página principal . . . 34
11. Diseño Crear acta . . . 35
12. Diseño editar acta . . . 37
13. Diseño Ver acta . . . 38
14. Diseño Crear usuario . . . 39
15. Código para iniciar backend . . . 42
16. Código para configurar el Frontend . . . 43
17. Código para Docker . . . 44
18. Código para configurar conexión de la base de datos . . . 45
19. Código con ejemplo de endpoints . . . 47
20. Código para iniciar sesión o registrar usuario . . . 48
21. Código para generar el JWT . . . 49
22. Código con lógica de creación de un acta . . . 50
23. Vista Final Inicio de sesión (Mensajes de error) . . . 51
24. Vista Final Página principal . . . 51
25. Vista Final Crear usuario . . . 52
26. Pruebas de caja negra . . . 53
27. Formulario con retroalimentación del usuario . . . 54
28. Diseño final del Inicio de sesión . . . 59
29. Diseño final de página principal . . . 59
30. Diseño final de crear usuario . . . 60
31. Diseño final de ver acta . . . 61
32. Diseño final de crear acta . . . 62
33. Diseño final de editar actas . . . 63

Resumen

En esta tesis se presenta el desarrollo de una plataforma web diseñada para facilitar la gestión y evaluación de trabajos de grado. El proyecto tiene como objetivo principal crear una herramienta eficiente que permita a los evaluadores registrar, consultar y gestionar actas de evaluación de manera centralizada, optimizando el proceso y asegurando un acceso rápido a la información relevante.

La plataforma contará con un sistema de creación de actas en línea, donde los evaluadores podrán registrar información clave sobre los trabajos de grado. Además, incluirá funcionalidades para buscar, filtrar y visualizar actas previamente registradas, así como la posibilidad de generar reportes y exportar actas.

Se describe el proceso completo de desarrollo de la plataforma, desde la identificación de las necesidades hasta la implementación técnica. Se detallan las metodologías utilizadas, las tecnologías empleadas en el desarrollo (como el framework web y el sistema de bases de datos), y se discuten los resultados esperados en términos de usabilidad y eficiencia del sistema. Finalmente, se presentan las pruebas de validación realizadas y los futuros alcances de la plataforma en el ámbito académico.

Palabras Clave: Plataforma, Trabajo de grado, Acta de evaluación, Framework web, Sistema de bases de datos, Optimización.

Summary

This thesis presents the development of a web platform designed to facilitate the management and evaluation of undergraduate theses. The main objective of the project is to create an efficient tool that enables evaluators to record, consult, and manage evaluation reports in a centralized manner, optimizing the process and ensuring quick access to relevant information.

The platform will feature an online report creation system, allowing evaluators to register key information about the theses. Additionally, it will include functionalities to search, filter, and view previously registered reports, as well as the ability to generate and export reports.

The thesis describes the complete development process of the platform, from the identification of needs to technical implementation. It details the methodologies used, the technologies employed (such as the web framework and database system), and discusses the expected outcomes in terms of system usability and efficiency. Finally, it presents the validation tests carried out and the future potential of the platform within the academic field.

Keywords: Platform, undergraduate thesis, evaluation report, web framework, database system, optimization.

1. Introducción

Los trabajos de grado en el pregrado representan una fase crucial durante la formación del estudiante pues esta exige una profundización en el campo disciplinar y un compromiso activo en la construcción de conocimiento y resolución de problemas complejos por medio del pensamiento crítico y la capacidad de trabajar autónoma y colaborativamente.

En este contexto, la gestión efectiva de los trabajos de grado en el pregrado se consolida como la evidencia tangible para asegurar el éxito en la formación investigativa. La planificación cuidadosa, la comunicación fluida y la evaluación meticulosa de estos trabajos son elementos clave que además de garantizar un seguimiento claro y conciso del progreso, se consideran primordiales para la creación de un entorno propicio para la investigación avanzada. No obstante, en ocasiones esta tarea se ve obstaculizada si no se cuenta con una organización eficiente de los datos, si existen posibles problemas de comunicación entre estudiantes y tutores, o si se experimentan problemas en la evaluación de los trabajos de grado.

Teniendo en cuenta que la administración eficaz de los trabajos de grado en el pregrado se configura como un pilar central para asegurar el éxito en la formación académica, las instituciones de educación superior como la Universidad Javeriana de Cali, enfrentan la necesidad de diseñar e implementar un software especializado que permita gestionar, revisar y consolidar eficazmente los procesos asociados a los trabajos de grado en el pregrado. Un sistema integral desarrollado en respuesta a estas necesidades específicas facilita la centralización de la información, la estandarización de procedimientos y fortalece el ecosistema investigativo, garantizando de esta forma la calidad académica.

2. Descripción del Problema

2.1. Planteamiento del Problema

Para el ámbito académico, contar con sistemas que faciliten la evaluación y gestión de los trabajos de grado resulta fundamental, dado que estos documentos requieren un proceso de revisión riguroso y bien documentado. Sin embargo, el proceso de evaluación de trabajos de grado puede ser tedioso, especialmente cuando se realiza de manera física o mediante sistemas como formularios, lo cual puede derivar en pérdida de información, dificultades en la organización y tiempos prolongados para la toma de decisiones.

En la actualidad, existen pocos sistemas que se adapten a las necesidades de evaluación de un trabajo de grado, además que faciliten la gestión de actas de evaluación en el ámbito educativo, lo que limita la eficiencia del proceso y la accesibilidad a la información por parte de los evaluadores, directores y otros agentes involucrados. Esta carencia de herramientas tecnológicas adecuadas plantea una oportunidad significativa para el desarrollo de una plataforma web que permita a los evaluadores acceder a una solución completa e intuitiva para gestionar las actas de manera ordenada y accesible.

La falta de un sistema dedicado a la evaluación de trabajos de grado no solo ralentiza el proceso, sino que también dificulta el seguimiento de los trabajos y sus evaluaciones a lo largo del tiempo, afectando la transparencia y el registro de información relevante en la trayectoria académica del estudiante. Por esta razón, desarrollar una plataforma que centralice la gestión de evaluaciones y permita un acceso ágil y seguro a los datos resulta una alternativa viable para apoyar la eficiencia, la organización y la fiabilidad en el proceso de evaluación académica.

2.1.1. Formulación

¿Cómo desarrollar una plataforma web que facilite la gestión y evaluación de trabajos de grado, optimizando el proceso de registro y consulta de actas para evaluadores y directores académicos?

2.1.2. Sistematización

¿Cómo se estructurará la información de los trabajos de grado y sus evaluaciones dentro de la plataforma para permitir un registro y consulta eficientes?

¿Qué herramientas tecnológicas se emplearán y cómo se diseñará una interfaz intuitiva y accesible para los evaluadores y directores?

¿Cómo se garantizará la seguridad, la confidencialidad de los datos y la correcta gestión de la autenticación y permisos de los usuarios?

¿Qué mecanismos se implementarán para la documentación, generación de reportes y gestión de actas de evaluación?

¿Qué tipos de pruebas se realizarán para validar la funcionalidad, usabilidad y eficiencia del sistema desarrollado?

2.2. Objetivos

2.2.1. Objetivo General

Desarrollar una plataforma web que mejore la gestión de las actas de evaluación de los trabajos de grado de ingeniería, proporcionando una herramienta eficiente y segura para registrar, consultar y administrar actas de evaluación académica.

2.2.2. Objetivos Específicos

Diseñar e implementar una interfaz web intuitiva que permita a evaluadores y directores registrar, consultar y modificar actas de evaluación de manera eficiente.

Desarrollar una arquitectura de base de datos robusta que facilite la gestión centralizada y optimizada de la información relacionada con los trabajos de grado.

Incorporar mecanismos de seguridad y control de acceso para proteger la confidencialidad de los datos y garantizar la integridad del sistema.

Validar el prototipo mediante pruebas funcionales de caja negra, incluyendo la verificación del control de acceso por roles, y elaborar una guía de usuario y documentación técnica que describa su arquitectura y configuración.

3. Marco de Referencia

3.1. Marco Teórico

El desarrollo de software constituye hoy en día la columna vertebral para la optimización de procesos dentro de las compañías, especialmente en instituciones educativas que requieren de sistemas seguros, eficaces y de fácil acceso para la gestión de información académica. El sistema de gestión de actas de grado actual permite realizar el registro de la información de forma correcta, pero en cuanto la interacción entre el usuario y la plataforma, el software es poco intuitivo dificultando de esta manera su uso y complicando la experiencia de la persona que lo utiliza.

En este sentido, se ha desarrollado un sistema mejorado y avanzado de gestión de actas de grado buscando que se permita la creación, edición y visualización de estas desde una interfaz más amigable y accesible para los usuarios, facilitando de esta forma su uso y mejorando la eficiencia de la gestión. El principal objetivo de este sistema es permitir un almacenamiento de datos más eficiente, facilitar la interacción del usuario con las actas y agilizar el seguimiento de los trabajos de grado calificados. Esta herramienta resulta bastante útil y genera un valor agregado a la Facultad de Ingeniería y Ciencias teniendo en cuenta la gran cantidad de trabajos de grado que se entregan cada semestre.

Considerando los beneficios de este sistema, es importante resaltar algunos de los conceptos clave que permiten la formación de este software, entre ellos encontramos:

- **JWT:**un estándar abierto utilizado para transmitir información de forma compacta, autónoma y segura entre dos partes que puede implementarse mediante un JWS (JWS: Firma web JSON) o un JWE (JWE: Cifrado web JSON).
- **Docker:** plataforma de software que permite la creación e implementación de aplicaciones rápidamente, para el desarrollo de este software se utilizó MySQL para crear la base de datos.
- **Postman:**plataforma que permite la prueba, diseño, simulación, depuración, supervisión y publicación de APIs desde una interfaz, en este caso, utilizada para pruebas de registro y login.
- **Frontend:** es la parte visible de un software y aquella con la que el usuario puede interactuar directamente.
- **Backend:** es la parte interna del software que permite la conexión con la base de datos del aplicativo o software.
- **Fastapi:** es el framework para crear APIS con python que se utilizó en este caso para crear la API del backend.

Teniendo presente lo descrito anteriormente, en el contexto de este proyecto, se busca unificar en un solo software todas las operaciones relacionadas con la gestión de un trabajo de pregrado. Es importante señalar que la Facultad de Ingeniería y Ciencias alberga ocho programas de pregrado que requieren trabajo de grado, por ende, debido al alto flujo de trabajos de pregrado que se reciben semestralmente, la meta principal que se espera con este proyecto es facilitar el almacenamiento de los datos, mejorar la comunicación, agilizar el seguimiento y facilitar al usuario todo el proceso que conllevan los trabajos de pregrado. Por el momento se busca que el software se ejecute localmente con la meta a futuro de que pueda escalarse hasta realizar un despliegue online.

El diseño de una plataforma de este tipo requiere la comprensión y aplicación de diversos conceptos teóricos fundamentales, por lo anterior, resulta imperativo definir algunos conceptos clave, entre ellos, la ingeniería de software, el ciclo de vida del software y las herramientas implementadas durante su desarrollo.

3.1.1. La ingeniería de software

"Es una disciplina de la ingeniería que se ocupa de todos los aspectos de la producción de software desde las primeras etapas, partiendo de la especificación del sistema, hasta el mantenimiento del mismo cuando ya ha sido entregado. En esta definición, hay dos frases clave. La primera es 'Disciplina de ingeniería', la cual indica que los ingenieros aplican de manera selectiva teorías, métodos y herramientas de manera que logren realizar sistemas funcionales. Además, siempre intentan descubrir soluciones a los problemas, incluso cuando no existen teorías o métodos aplicables. En segundo lugar, los ingenieros reconocen que deben trabajar con recursos organizativos y financieros determinados; a la hora de buscar soluciones deben tener en cuenta dichas limitaciones." [1].

3.1.2. El ciclo de vida del software

El ciclo de vida del software es un proceso estructurado que describe las fases necesarias para crear una aplicación de software. Este ciclo busca garantizar que el aplicativo haya sido desarrollado de manera controlada, planificada y eficiente, cumpliendo con los requisitos solicitados por el usuario. El ciclo de vida del software se compone por las siguientes etapas:

- **Planificación:** en esta fase, se definen las necesidades del sistema mediante reuniones con los usuarios, clientes y partes interesadas. Se determinan los objetivos y los requisitos funcionales y no funcionales.
- **Diseño del sistema:** con los requisitos establecidos se procede a diseñar el sistema definiendo los componentes principales y planificando el desarrollo y la arquitectura del software.

- **Implementación (codificación):** se escribe el código fuente basado en el diseño, desarrollando un prototipo funcional con ayuda de toda la información recopilada anteriormente.
- **Verificación del sistema:** tras finalizar el desarrollo, se corrigen errores, se optimiza el rendimiento y se agregan nuevas funcionalidades hasta que el ciclo cumpla con los requisitos inicialmente establecido.

Estos pasos se pueden repetir a lo largo del ciclo de vida del software.

3.1.3. Patrón de arquitectura MVC

El Modelo-Vista-Controlador (MVC) es un patrón arquitectónico creado en 1979 por Trygve Reenskaug, con el objetivo de simplificar la interacción entre usuarios y sistemas complejos. El MVC separa los datos y la lógica de negocio (modelo) de su representación visual (vista) y del módulo que gestiona las comunicaciones y eventos (controlador). Esta separación de conceptos es fundamental para crear aplicaciones robustas y fáciles de mantener.

3.1.4. FastApi

FastAPI es un framework moderno para el desarrollo de APIs web en Python, orientado a la creación de servicios backend de manera rápida y estructurada. Se caracteriza por facilitar la definición de endpoints mediante un enfoque declarativo, soportar validación automática de datos de entrada y salida, y permitir la generación de documentación interactiva de la API. Además, está diseñado para trabajar de forma eficiente con aplicaciones web que se comunican mediante peticiones HTTP (por ejemplo, desde un frontend), promoviendo una arquitectura clara y escalable para sistemas basados en servicios.

3.1.5. Node.js

El Node.js es un entorno de ejecución JavaScript de un solo hilo, de código abierto y multiplataforma que permite ejecutar códigos de este lenguaje fuera del navegador web, es decir, del lado del servidor. Funciona ejecutando el motor JavaScript V8 y el núcleo de Google Chrome, además combina tecnologías de desarrollo front-end y back-end, ejecutando las aplicaciones dentro de su propio tiempo de ejecución en el servidor, lo que permite manejar grandes volúmenes de datos.

3.1.6. MySQL

MySQL es un sistema de gestión de bases de datos de código abierto utilizado principalmente para almacenar, gestionar y recuperar datos de forma organizada.

Se basa en el lenguaje SQL para interactuar con la información permitiendo consultarlos, manipularlos y controlarlos.

3.1.7. Sequelize

Sequelize es un ORM (Object-Relational Mapping) para Node.js que facilita la interacción entre una aplicación desarrollada en javascript y una base de datos relacional como MySQL. Sequelize, en lugar de realizar consultas directas a SQL, permite manipular los datos mediante objetos y métodos de Javascript simplificando así el trabajo con bases de datos en el desarrollo backend.

3.1.8. phpMyAdmin

phpMyAdmin es una herramienta web de administración para MySQL que permite gestionar bases de datos mediante una interfaz gráfica en el navegador. A través de ella es posible crear y modificar bases de datos y tablas, ejecutar consultas SQL, administrar usuarios y permisos, e importar o exportar información, facilitando estas tareas sin necesidad de usar exclusivamente la línea de comandos.

3.1.9. Trabajos Relacionados en el Ámbito Estudiantil

- **Sistema para el seguimiento de trabajos de grado**.

  La gestión manual de trabajos de grado genera alta carga administrativa, errores y dispersión de información en correos y archivos. El trabajo propone un sistema integral para centralizar el proceso, automatizar la comunicación y facilitar seguimiento de estado y apoyar la toma de decisiones(Martínez, P. H. and Mena, S. (2024).). La diferencia con el proyecto propuesto, principalmente es que este se busca implementar para los trabajos de grado en pregrado y sus respectivas actas de evaluación.

- **Sistema de gestión para trabajos de grado en posgrado**

  Este proyecto es una herramienta web para la gestión de trabajos de grado en posgrados. También permite realizar seguimiento y control a cada proyecto, lo que ayuda a simplificar los tiempos en la entrega de documentos, asignación de jurados, elaboración de notificaciones, entre otras. (Amaya, J. E. (2025).). La diferencia con el proyecto propuesto, principalmente es que este se busca implementar para los trabajos de grado en pregrado y sus respectivas actas de evaluación, además permitir a estudiantes y evaluadores la posibilidad de mantenerse al día frente a las actas de evaluación en el sistema.

- **Sistema de información Web que asiste el proceso de radicación y seguimiento de proyectos de grado de la Especialización en Ingeniería de Software de la Universidad Distrital Francisco José de Caldas**

  Este proyecto aborda todas las etapas del ciclo de vida básico del software para el desarrollo de un sistema de información web que apoya el proceso de radicación de proyecto de grado de los estudiantes. Asimismo, respalda el proceso de asignación de directores, revisores, el proceso de seguimiento y ofrece herramientas funcionales que apoyen la gestión y búsqueda de trabajos ya existentes. El proyecto propuesto se diferencia en que se realizará específicamente para la Facultad de Ingeniería y Ciencias de la Universidad Javeriana Cali.

3.1.10. Trabajos Relacionados en el Ámbito Empresarial

- **ASANA**

  Es una herramienta de gestión de proyectos que permite crear informes ilimitados en tiempo real sobre los datos de los proyectos de una empresa automatizando los flujos para aumentar la productividad y asignación sobre quien se encarga de las tareas. Aunque es una plataforma que permite la centralización de trabajos, el proyecto mencionado anteriormente se enfoca en evaluaciones académicas formales mediante actas y roles definidos por la universidad.

- **ClickUp**

  Es una herramienta de gestión de proyectos y colaboración en equipo que centraliza el trabajo de equipos y personas y reemplazando el uso de múltiples herramientas. Permite la organización de tareas en jerarquías con vistas personalizables y facilitando el seguimiento y colaboración entre los integrantes de un equipo, siendo una herramienta adaptable a cualquier tipo de proyecto o necesidad. Se diferencia con el proyecto propuesto en que este último se trata de un sistema de especializado para gestionar actas de evaluación académica, por lo que tendrá las funciones necesarias y pertinentes para realizar con éxito dicha gestión.

4. Requisitos del sistema de gestión

Para este capítulo, se realizó una obtención de requisitos que se basó en reuniones, análisis de documentos y validaciones con usuarios, luego se pasó la creación de un diagrama de contexto para comprender el alcance y la interacción del sistema de gestión con su entorno. Además, se elaboraron diagramas de actividades para representar los pasos y estados que conforman el proceso dentro del sistema. También se diseñó un diagrama de casos de uso, el cual ofrece una visión general de las funciones que desempeñará cada actor dentro del sistema. Finalmente, toda esta información se consolidó en los requisitos funcionales, no funcionales y limitaciones, proporcionando una base clara para el desarrollo del prototipo funcional.

4.1. Metodología de Obtención de Requisitos

Para la definición y validación de los requisitos del sistema, se ejecutó un proceso de recolección de información que involucró a los actores clave del proceso administrativo y académico. La metodología se dividió en tres fases:

- **Entrevistas con el director del programa:** Se realizaron sesiones de trabajo con el Dr. Gerardo Mauricio Sarria, Director de la Carrera de Ingeniería de Sistemas y Computación. El objetivo de estas sesiones fue comprender el flujo general del trabajo de grado, definir el alcance del sistema y establecer las reglas, tales como los roles permitidos y las restricciones en el acceso a la información.
- **Análisis Documental y Modelado de Datos:** Se tomó como insumo principal el formato oficial actual de las Actas de Evaluación de Trabajo de Grado, para esto se utilizó como referencia real actas históricas, analizando su estructura, como el acta del estudiante Juan Esteban Amaya 1. A partir de este documento, se realizó un análisis para identificar:
  - **Entidades de datos:** Títulos, autores, códigos, jurados y directores.
  - **Lógica de evaluación:** Se mapearon los campos de calificación cuantitativa y cualitativa, identificando la necesidad de calcular promedios automáticos y convertir notas numéricas a texto, replicando la estructura exacta del documento físico en la base de datos.

Facultad de Ingeniería y Ciencias

ACTA DE EVALUACIÓN DE TRABAJO DE GRADO
Carrera de Ingeniería de Sistemas y Computación

Información del proyecto

Número acta	419
Fecha	lunes, marzo 3, 2025
Programa académico	Ingeniería de Sistemas y Computación
Título proyecto de grado	Sistema de gestión para trabajos de grado en posgrado.
Autor 1	Juan Esteban Amaya Ramirez
Código autor 1	8949439
Doc Identidad autor 1	1001349447
Email autor 1	amaya2001@javerianacali.edu.co
Nombre Director	Juan Carlos Martínez
Correo electrónico director	juancmartinez@javerianacali.edu.co
Jurado Uno	Luisa Rincón
Correo electrónico - Jurado Uno	lfrincon@javerianacali.edu.co
Jurado Dos	Diego Linares
Correo electrónico - Jurado Dos	dlinares@javerianacali.edu.co

Figura 1: Acta de evaluación referencia (Estudiante Juan Esteban Amaya)

- **Validación con Usuario Final:** Se realizó una sesión de validación y pruebas de usabilidad con Andrea del Pilar Gamboa, Secretaria de la Facultad de Ingeniería. Al ser ella la usuaria potencial encargada de la gestión operativa de las actas, su retroalimentación permitió ajustar aspectos de la interfaz y confirmar que el flujo de creación de actas se alineaba con sus necesidades diarias de operación y eficiencia.

4.2. Diagrama de Contexto

El primer paso en todos los desarrollos de software es entender el contexto en el que el sistema se va a desenvolver; de este análisis podremos conocer las personas o entidades que van a hacer uso del sistema, así como las necesidades que necesita cubrir. Con esto en mente, se llevó a cabo una reunión con el director de la carrera Ingeniería de Sistemas y Computación de la Universidad Javeriana Cali, Gerardo Mauricio Sarria. A partir de esta se elaboró el siguiente diagrama de contexto en el que se concluyó quienes son los stakeholders, las entradas y salidas de información, y los límites del sistema de gestión. Esta visión general que proporciona el diagrama ayuda a dar una primera idea de los requisitos que requiere el sistema.

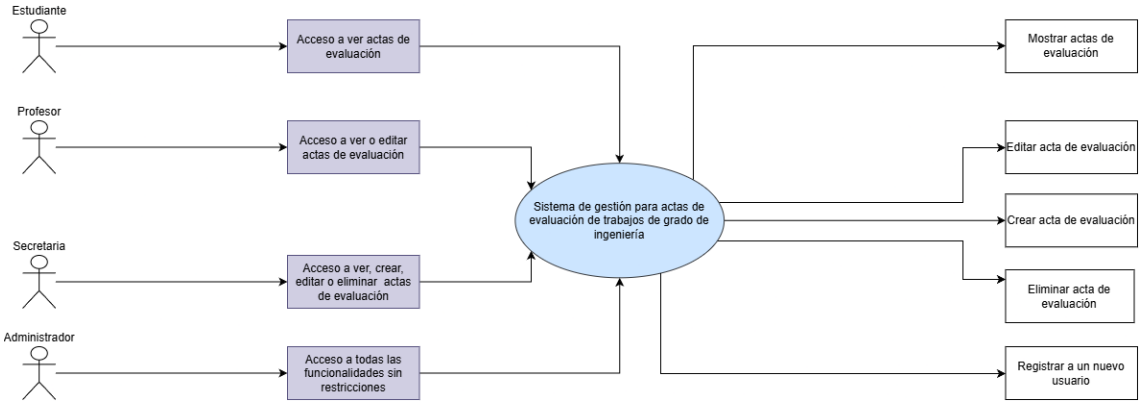

Figura 2: Diagrama de Contexto

Como se puede ver en el diagrama de la **Figura 2**, el sistema involucra a 4 actores, cuyos roles se describen a continuación:

- **Estudiante:** Es una persona que hace parte de un programa de pregrado de la facultad de ingeniería y ciencias. Este a diferencia de los demás roles solo podrá hacer uso de una funcionalidad, la cuál será observar las actas de evaluación cargadas en el sistema.
- **Profesor:** Es un colaborador de la facultad de ingeniería y ciencias que puede ser profesor o director académico de una carrera. Este puede hacer uso de las funcionalidades para ver o editar actas de evaluación cargadas en el sistema.
- **Secretaria:** Es un colaborador de la facultad de ingeniería y ciencias, normalmente este rol se le asigna a la secretaria de la facultad. Este es el encargado de utilizar la funcionalidad para crear las actas de evaluación que se cargarán al sistema, además cuenta con los permisos para utilizar las funcionalidades para ver, editar o eliminar actas de evaluación cargadas en el sistema.
- **Administrador:** Es la persona con acceso a todas las funcionalidades que existen el sistema, normalmente este rol solo se le asigna a el desarrollador del software. No obstante, como los demás roles no pueden registrar usuarios, este rol es vital que siempre exista, ya que es el único que puede acceder a la funcionalidad de registro de nuevos usuarios.

4.3. Diagrama de actividades

Después de entender el contexto del sistema, se consideró realizar un diagrama de actividades para lograr comprender de manera general el sistema de gestión de actas de evaluación. Durante la implementación del diagrama, se identificó una oportunidad para lograr una comprensión profunda del sistema, la cual se basaba en implementar un diagrama general del sistema y otro diagrama que detalla los roles

y las funcionalidades del sistema. A partir de este análisis, se tomó la decisión de desarrollar dos diagramas de actividades.

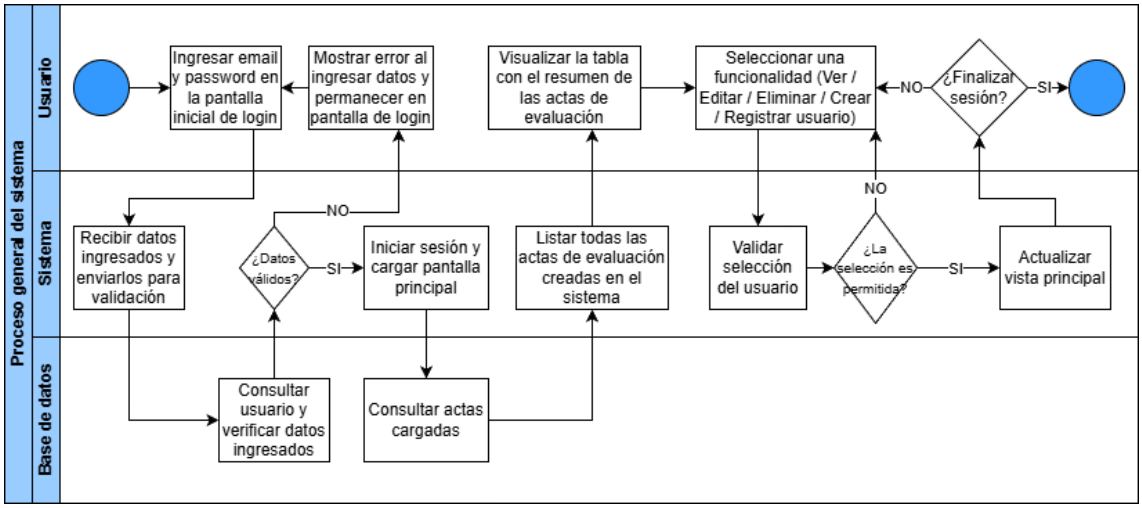

Figura 3: Diagrama de Actividades - Proceso general del sistema

La **Figura 3** presenta el diagrama de actividades del proceso general del sistema, organizado en tres carriles (Usuario, Sistema y Base de datos). El flujo inicia en la pantalla de inicio de sesión, cuando el usuario ingresa sus credenciales. El sistema recibe estos datos y consulta la base de datos para validarlos; si son incorrectos, se muestra un mensaje de error y el usuario permanece en la pantalla de login. Si la validación es exitosa, el sistema inicia la sesión, carga la pantalla principal y lista las actas de evaluación registradas. Luego, el usuario selecciona una funcionalidad (ver, editar, eliminar, crear acta o registrar usuario) y el sistema valida si dicha acción está permitida según sus permisos; si no lo está, el usuario regresa a la vista principal, y si lo está, el sistema actualiza la vista. El proceso se repite mientras el usuario continúe interactuando con el sistema y termina cuando decide finalizar la sesión.

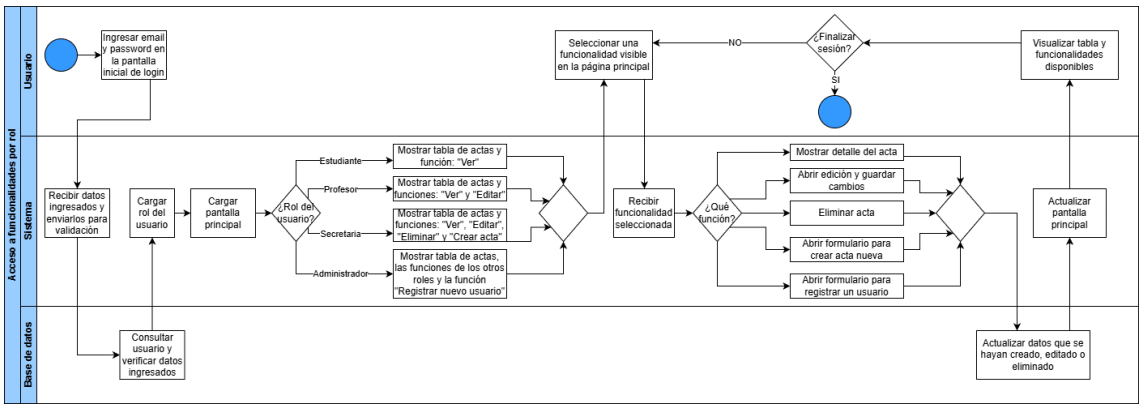

Figura 4: Diagrama de Actividades - Acceso a funcionalidades por roles

Como se puede ver en el diagrama de la **Figura 4**, el sistema utiliza un control

de acceso a funcionalidades por roles. Este diagrama, en complemento al anterior, detalla como el sistema identifica el rol del usuario autenticado y, basado en eso, renderiza la pantalla principal mostrando solamente las funciones a las que tiene permiso cada rol. En consecuencia, el rol Estudiante visualiza el listado de actas y la opción de ver el detalle de un acta; el rol Profesor visualiza además la opción de edición; el rol Secretaría dispone de opciones para ver, editar, eliminar y crear actas; y el rol Administrador incluye, adicionalmente, la opción de registrar nuevos usuarios.

4.4. Diagrama de casos de uso

Los diagramas previos permitieron ver de manera detallada la forma en que se implementó el sistema de gestión de actas y la manera en que se puede utilizar, dependiendo del rol al que pertenece el usuario. A continuación, se consideró necesario la vista del sistema desde una perspectiva de alto nivel, para esto se decidió realizar un diagrama de casos de uso.

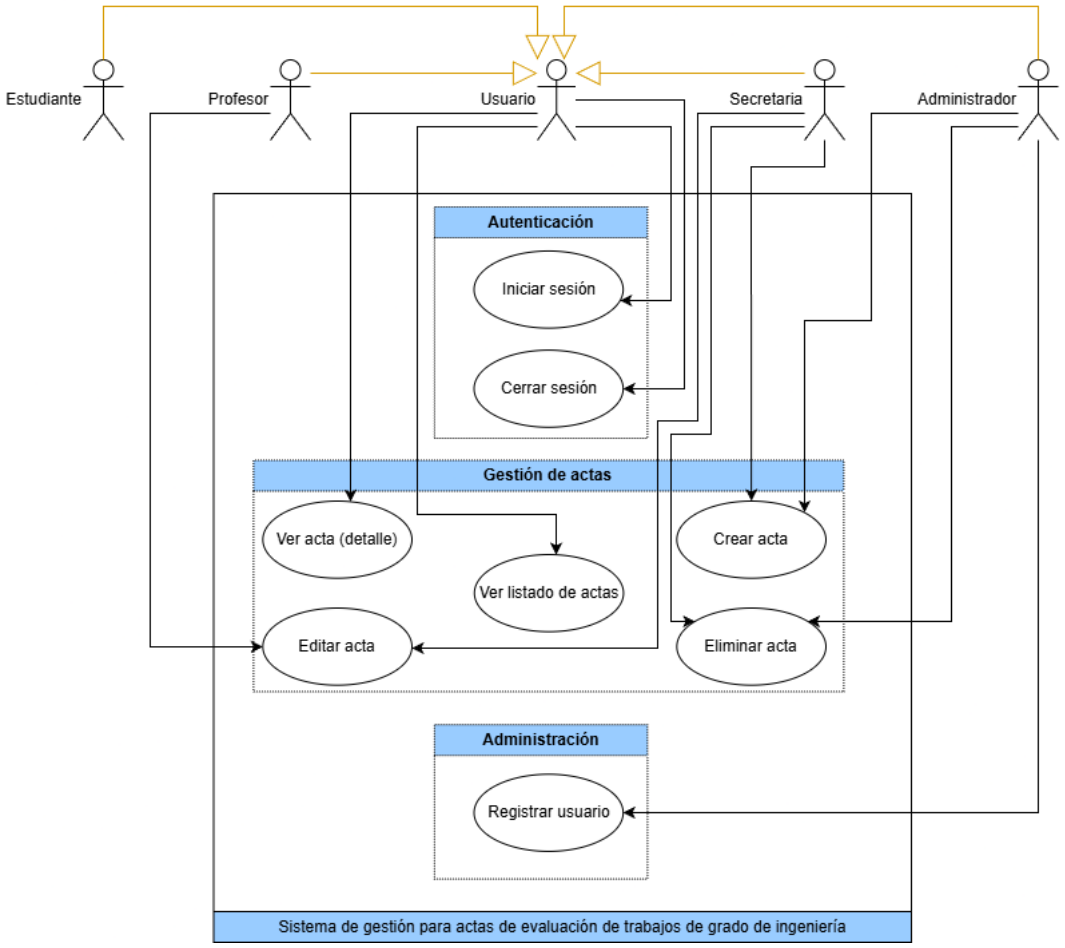

Figura 5: Diagrama de casos de uso

En el diagrama de casos de uso de la **Figura 5** se identificaron tres grupos que corresponden a los tres usos principales del sistema, cada uno con sus funciones bien definidas. La decisión de agrupar las funciones es importante ya que permite una mejor organización y gestión del sistema. También, se identificaron cuatro roles con funciones "únicas" y un rol que solo se utilizó en el diagrama de casos de uso para la representación de un usuario general, y su respectivo comportamiento que puede ser generalizado mediante la herencia de funciones de los roles originales a este usuario general. Con este diseño, se logró crear un diagrama que permite conectar las funciones "únicas" con su respectivo rol, y las que pueden realizar todos los roles con el rol "usuario".

El propósito de los diagramas es ayudar a entender cómo se maneja el sistema de gestión de actas de trabajos de grado. Teniendo claridad sobre el funcionamiento del sistema, ahora se pueden establecer los requisitos, que serán la base para el desarrollo del prototipo del sistema. A continuación, se especificará los requisitos funcionales, no funcionales y las limitaciones que van a definir el correcto funcionamiento del prototipo del sistema de gestión.

4.5. Requisitos funcionales

Autenticación de usuarios

- **RF-001:** El sistema debe permitir a cualquier usuario (Estudiante, Profesor, Secretaria y Administrador) iniciar sesión desde la pantalla de login mediante el ingreso de correo electrónico y contraseña. El sistema debe validar las credenciales consultando la base de datos SQL local y, en caso de éxito, redirigir al usuario a la pantalla principal cargando su rol correspondiente.
- **RF-002:** El sistema debe impedir el acceso a la pantalla principal cuando las credenciales ingresadas sean incorrectas o el usuario no exista en la base de datos. En este caso, el sistema debe mostrar un mensaje de error y mantener al usuario en la pantalla de login.

Estudiante

- **RF-003:** El sistema debe mostrar al Estudiante, al acceder a la pantalla principal, una tabla con el listado de actas registradas y un conjunto de campos resumen.
- **RF-004:** El sistema debe permitir al Estudiante consultar el detalle completo de un acta seleccionada desde el listado mediante la opción "Ver". Al seleccionar esta opción, el sistema debe abrir la vista correspondiente del acta y mostrar toda la información almacenada para dicho registro.

Profesor

- **RF-005:** El sistema debe permitir al Profesor realizar las mismas acciones de consulta disponibles para el Estudiante (visualizar listado de actas en la pantalla principal y visualizar el detalle mediante "Ver").
- **RF-006:** El sistema debe permitir al Profesor editar un acta existente mediante la opción "Editar" disponible en la pantalla principal. Al seleccionar "Editar", el sistema debe redirigir a la pantalla de edición cargando los datos actuales del acta, permitir la modificación de los campos y, al presionar "Guardar", actualizar el registro en la base de datos SQL local y retornar a la pantalla principal.

Secretaria

- **RF-007:** El sistema debe permitir a la Secretaria realizar las mismas acciones de consulta y edición disponibles para el Profesor (visualizar listado, ver detalle y editar actas).
- **RF-008:** El sistema debe permitir a la Secretaria eliminar un acta existente mediante la opción "Eliminar" disponible en la pantalla principal. Al ejecutar la eliminación, el sistema debe remover el registro de la base de datos SQL local y actualizar el listado mostrado en la pantalla principal para que el acta eliminada no vuelva a aparecer.
- **RF-009:** El sistema debe permitir a la Secretaria crear un acta de evaluación mediante un botón "Crear acta" disponible en la pantalla principal. Al seleccionar esta opción, el sistema debe redirigir a la pantalla de creación de acta, permitir el diligenciamiento de los campos requeridos y, al presionar "Crear", registrar el acta en la base de datos SQL local y retornar a la pantalla principal donde se podrá ver el acta creada en el listado.

Administrador

- **RF-010:** El sistema debe permitir al Administrador realizar todas las funcionalidades disponibles para la Secretaria (visualizar listado, ver detalle, editar, eliminar y crear actas), dado que este rol tiene acceso completo a las pantallas del sistema.
- **RF-011:** El sistema debe permitir al Administrador registrar nuevos usuarios desde un botón exclusivo de "Registrar usuario" disponible en la pantalla principal. Al seleccionarlo, el sistema debe redirigir a la pantalla de registro de usuario nuevo, permitir el registro de nombre, correo, contraseña y rol, y almacenar el nuevo usuario en la tabla de usuarios de la base de datos SQL local, habilitándolo para iniciar sesión posteriormente.

Sistema de gestión

- **RF-012:** El sistema debe restringir la visibilidad y acceso a funcionalidades en la pantalla principal (/acta) según el rol del usuario autenticado. En consecuencia, el sistema debe mostrar únicamente los botones permitidos para cada rol:
  - **Estudiante:** Ver
  - **Profesor:** Ver, Editar
  - **Secretaria:** Ver, Editar, Eliminar y Crear acta
  - **Administrador:** Ver, Editar, Eliminar, Crear acta y Registrar usuario

  De esta manera, el usuario solo podrá seleccionar acciones disponibles conforme a sus permisos.

4.6. Requisitos no funcionales

- **RNF-001:** La interfaz de usuario debe ser intuitiva y consistente, permitiendo a los diferentes roles (Estudiante, Profesor, Secretaria y Administrador) completar sus tareas principales con facilidad. El sistema debe presentar mensajes claros ante errores y proporcionar retroalimentación al completar acciones como crear, editar o eliminar un acta.
- **RNF-002:** El sistema debe proteger el acceso a la información mediante autenticación (correo y contraseña) y autorización basada en roles. Adicionalmente, la interfaz debe mostrar únicamente las funcionalidades permitidas para cada rol, evitando el acceso a operaciones no autorizadas.
- **RNF-003:** El código del sistema debe estar organizado de forma modular, con una estructura clara que facilite el mantenimiento, la corrección de errores y futuras mejoras.
- **RNF-004:** El sistema debe ser compatible con navegadores web modernos, garantizando su correcto funcionamiento al menos en la versión actual de Google Chrome (visualización de la pantalla principal, formularios y navegación entre vistas).

4.7. Limitaciones

El sistema desarrollado corresponde a un prototipo orientado exclusivamente a la gestión de actas de evaluación de trabajos de grado en ingeniería, por lo cual su alcance funcional se limita a la visualización, creación, edición y eliminación de actas, así como al registro de usuarios y la asignación de permisos mediante roles. En consecuencia, no contempla procesos adicionales asociados a la gestión integral del trabajo de grado, tales como la radicación de documentos complementarios, control de versiones, generación automática de documentos oficiales o el envío de notificaciones.

Adicionalmente, la solución fue implementada para operar en un entorno local, utilizando una base de datos SQL local para el almacenamiento de usuarios y actas. Por esta razón, el sistema no incorpora mecanismos de despliegue en la nube, acceso remoto institucional, integración con directorios de autenticación ni sincronización con otros sistemas académicos (correo institucional, one gate o brightspace). En este contexto, la creación de usuarios depende del módulo de registro habilitado únicamente para el rol Administrador.

Finalmente, los permisos y funcionalidades visibles en la interfaz están definidos de acuerdo con la lógica implementada en el prototipo; por lo tanto, cualquier cambio en los criterios de evaluación, estructura del acta o políticas de roles requerirá ajustes en la configuración y/o en el código del sistema para garantizar su correcta adaptación.

5. Diseño del sistema de gestión

En este capítulo se presenta el diseño del Sistema de gestión para actas de evaluación de trabajos de grado de ingeniería, con el propósito de describir la forma en que se estructuró e implementó la solución a partir de los requisitos definidos. Se detallan los componentes principales del sistema, la arquitectura adoptada y las decisiones de diseño que permiten soportar las funcionalidades de autenticación, gestión de actas y control de acceso por roles.

5.1. Diseño general de la arquitectura del sistema

El sistema de gestión se desarrolló siguiendo una separación de responsabilidades inspirada en el patrón Modelo Vista Controlador (MVC). El diagrama de la **Figura 6** presenta la arquitectura general implementada y el flujo de interacción entre sus partes.

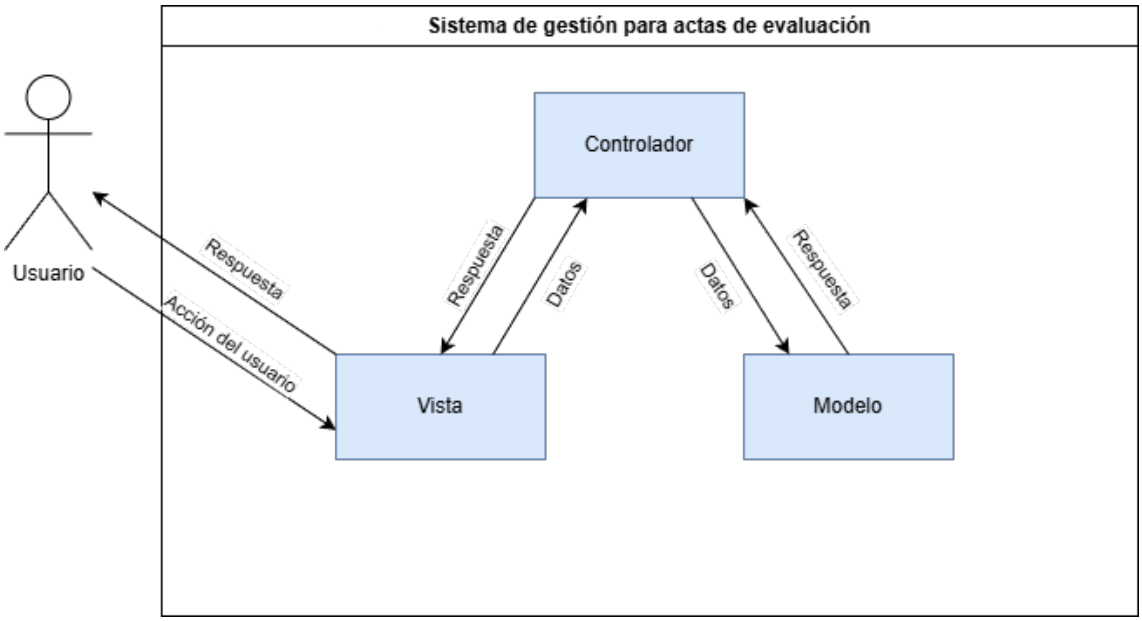

Figura 6: Diagrama general de la arquitectura del sistema

Como se puede observar en el diagrama de la **Figura 6**, La Vista corresponde al frontend, encargado de la interfaz y la navegación. El Controlador está representado por el backend implementado, enfocado en sus routers y endpoints, que reciben las solicitudes HTTP y coordinan la ejecución de las operaciones. El Modelo agrupa la lógica de negocio y el acceso a datos, implementados mediante las capas de service y repository, las cuales interactúan con la base de datos. Esta arquitectura permite mantener el código organizado y facilita la mantenibilidad y evolución del sistema.

5.1.1. Diseño propuesto de arquitectura para el sistema

La **Figura 7**, presenta el mapeo de la arquitectura propuesta bajo el enfoque MVC. La Vista está implementada en React (Frontend), encargada de la interacción con el usuario y la presentación de formularios y listados. El Controlador corresponde a la API desarrollada en FastAPI (Backend), que expone endpoints REST, recibe las solicitudes HTTP provenientes del frontend y coordina la ejecución de operaciones. El Modelo agrupa la lógica de negocio y persistencia mediante una estructura Service/Repository, la cual consulta y almacena la información en la base de datos MySQL (Database).

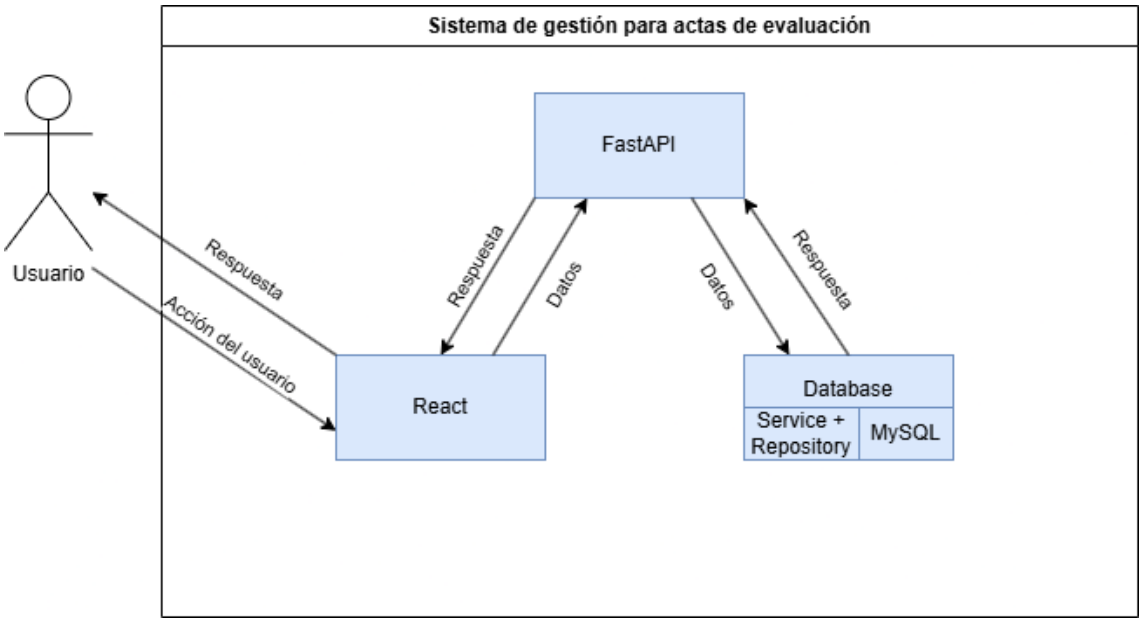

Figura 7: Diagrama propuesto para la arquitectura del sistema

Flujo general de la arquitectura presentada

- Primero, el usuario interactúa con la vista.
- Segundo, React envía solicitudes al backend.
- Tercero, FastAPI delega la lógica al servicio correspondiente.

- Cuarto, el repositorio ejecuta consultas SQL en MySQL.
- Quinto, la respuesta retorna al frontend para actualizar la interfaz.

5.2. Diseño de la base de datos

En esta sección se describe el diseño de la base de datos utilizada por el Sistema de gestión para actas de evaluación de trabajos de grado de ingeniería, la cual permite almacenar y consultar la información necesaria para el funcionamiento del sistema. La base de datos soporta principalmente dos conjuntos de datos: los usuarios (incluyendo sus credenciales y rol) y las actas de evaluación, las cuales contienen la información académica y los campos requeridos para el registro y gestión de la evaluación.

5.2.1. Selección de la base de datos

Para este proyecto se seleccionó MySQL como sistema gestor de base de datos debido a su estabilidad, amplio uso en aplicaciones web y disponibilidad de herramientas que facilitan su administración. Al tratarse de un sistema que requiere almacenar información estructurada (usuarios, roles y actas de evaluación), el enfoque relacional permite organizar los datos de manera consistente y realizar consultas eficientes para las operaciones de listado, creación, edición y eliminación.

Adicionalmente, la base de datos se ejecuta en un entorno local mediante Docker, lo cual facilita la instalación y configuración del entorno de desarrollo, permitiendo replicar el mismo esquema y condiciones de ejecución sin depender de instalaciones manuales en cada equipo. Para la administración y verificación de los datos se empleó DataGrip, herramienta que permitió inspeccionar tablas, ejecutar consultas y validar la integridad de la información almacenada durante el desarrollo y las pruebas del sistema.

5.2.2. Modelo de la base de datos

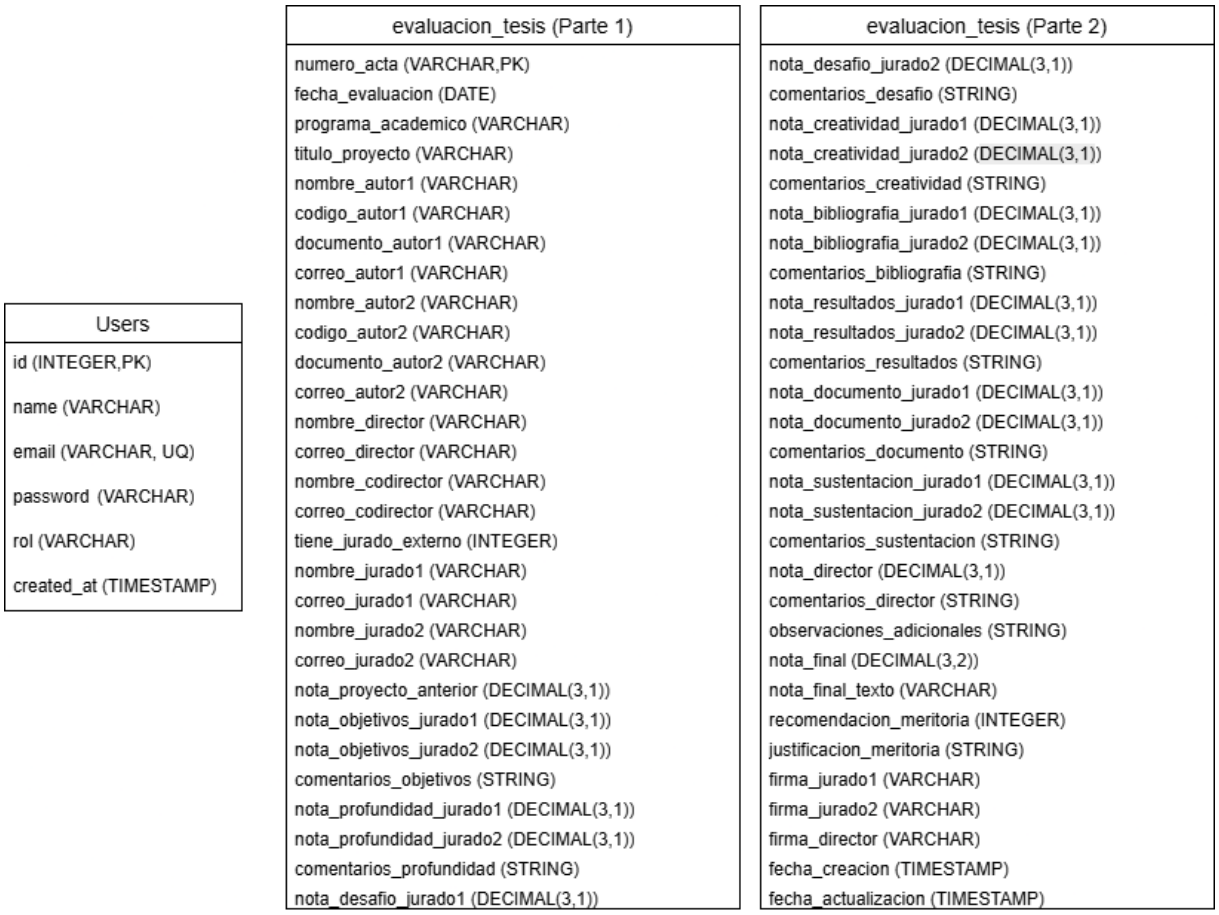

Figura 8: Modelo base de datos

El modelo de la base de datos en la **Figura 8** se compone de dos tablas principales: **Users** y **evaluacion tesis**. La tabla **Users** almacena la información necesaria para la autenticación y autorización del sistema, incluyendo los datos básicos del usuario y su rol. En esta tabla, el campo **id** se utiliza como clave primaria (PK) y el campo email se considera único (UQ), dado que permite identificar a cada usuario de forma individual dentro del sistema.

Por otra parte, la tabla **evaluacion tesis** almacena la información completa asociada a cada acta de evaluación. La clave primaria de esta entidad es el campo **numero acta**, el cual identifica de manera única cada registro. Esta tabla concentra los metadatos del acta, la información del proyecto y de los autores, los datos del director y jurados, así como los campos de calificación y comentarios correspondientes a los criterios de evaluación. Debido a que la estructura del acta incluye un conjunto amplio de datos (58 columnas), el diagrama del modelo se representó en dos secciones con el fin de mejorar la legibilidad, manteniendo la consistencia de una única entidad.

Finalmente, el modelo se implementó de manera que las operaciones del sistema (crear, consultar, actualizar y eliminar actas; y gestionar usuarios) se realicen mediante consultas directas a estas dos tablas en la versión actual del prototipo.

5.3. Diseño de las vistas

En esta sección se describe el diseño de las vistas del Sistema de gestión para actas de evaluación de trabajos de grado de ingeniería, enfocándose en la estructura de la interfaz, la navegación entre pantallas y la forma en que cada usuario interactúa con las funcionalidades del sistema. Las vistas fueron diseñadas con el objetivo de presentar la información de manera clara y permitir la ejecución de las acciones principales según el rol del usuario autenticado.

5.3.1. Selección de tecnologías para las vistas

Para el desarrollo de las vistas del sistema se utilizó React, debido a su enfoque basado en componentes, lo cual facilita la construcción de interfaces reutilizables, mantenibles y escalables. Este enfoque permitió implementar de forma organizada las pantallas principales del sistema (inicio de sesión, listado de actas, creación, edición, visualización y registro de usuarios) y gestionar el comportamiento de la interfaz según el rol autenticado.

La navegación entre pantallas se implementó mediante React Router DOM, permitiendo definir rutas específicas y controlar el acceso a las vistas de acuerdo con los permisos establecidos a cada rol. Para el manejo de formularios se empleó React Hook Form, herramienta que permitió capturar y validar los datos ingresados por el usuario en procesos como login, creación y edición de actas, optimizando el control del estado del formulario y la validación de campos.

Finalmente, la comunicación con el backend se gestionó mediante TanStack React Query, con el fin de realizar solicitudes a los endpoints del sistema y mantener sincronizada la información mostrada en pantalla con los datos almacenados en la base de datos. Esta librería facilitó el manejo de consultas, estados de carga, y actualización automática de información tras operaciones como crear, editar o eliminar actas.

5.3.2. Mock-ups para las vistas

En esta sección se presentan los mock-ups de las principales vistas del sistema, con el fin de ilustrar la estructura de la interfaz y la forma en que el usuario interactúa con cada pantalla. Estas representaciones permiten evidenciar la distribución de los componentes visuales, los elementos de navegación y los controles disponibles para ejecutar las funcionalidades del sistema.

Los mock-ups se crearon utilizando la herramienta de diseño gráfico Canva. Para los diseños se utilizó el logo de la institución, además de estilos y colores, similares a los que se usan en la universidad Javeriana Cali en sus aplicaciones y páginas web.

Inicio de sesión

La **Figura 9** corresponde al diseño del inicio de sesión, el cual se realizó con base a los requisitos RF-001 y RF-002, donde se especifica los datos necesarios para ingresar al prototipo (correo electrónico y contraseña).

Figura 9: Diseño Inicio de sesión

Página principal

El diseño de la **Figura 10** corresponde al diseño de la página principal del sistema, desde la perspectiva del rol Administrador para poder hacer visibles todas las acciones posibles. Esta vista se realizó con base en los requisitos del sistema de gestión RF-012, dado que permite visualizar una tabla con el listado de actas registradas y acceder al detalle completo de una acta seleccionada mediante la opción "Ver" RF-003 y RF-004. Adicionalmente, se implementaron las funcionalidades asociadas a los roles con mayores permisos: el rol Profesor puede acceder a la opción

"Editar" (RF-005 y RF-006), mientras que el rol Secretaria cuenta además con la opción "Eliminar" y el botón para "Crear acta" (RF-007, RF-008 y RF-009). Finalmente, el rol Administrador dispone de todas las acciones anteriores y del botón exclusivo para "Registrar usuario" (RF-010 y RF-011). Conforme a lo establecido en RF-012, el sistema muestra en esta pantalla únicamente los botones y acciones permitidos según el rol autenticado, evitando que el usuario acceda a funciones no autorizadas.

Figura 10: Diseño página principal

Crear acta

El diseño en la **Figura 11** corresponde al diseño de la vista "Crear acta", la cual se implementó con base en el requisito RF-009, dado que permite a la Secretaria diligenciar la información requerida para registrar una nueva acta de evaluación en el sistema. En esta interfaz se organiza el formulario por secciones, con el fin de facilitar el ingreso estructurado de la información.

Debido a la cantidad de campos asociados al formato de evaluación, la vista se presenta en dos partes las partes principales de esa vista: en la primera se incluye el botón "Regresar a página principal", que permite retornar al panel principal sin crear una nueva acta; y en la segunda se encuentra el botón "Guardar evaluación", el cual confirma la creación del acta y almacena los datos en la base de datos. Conforme a lo definido en RF-012, el acceso a esta vista y la visualización del botón "Crear acta" en la página principal se restringen únicamente a los roles autorizados (Secretaria y Administrador).

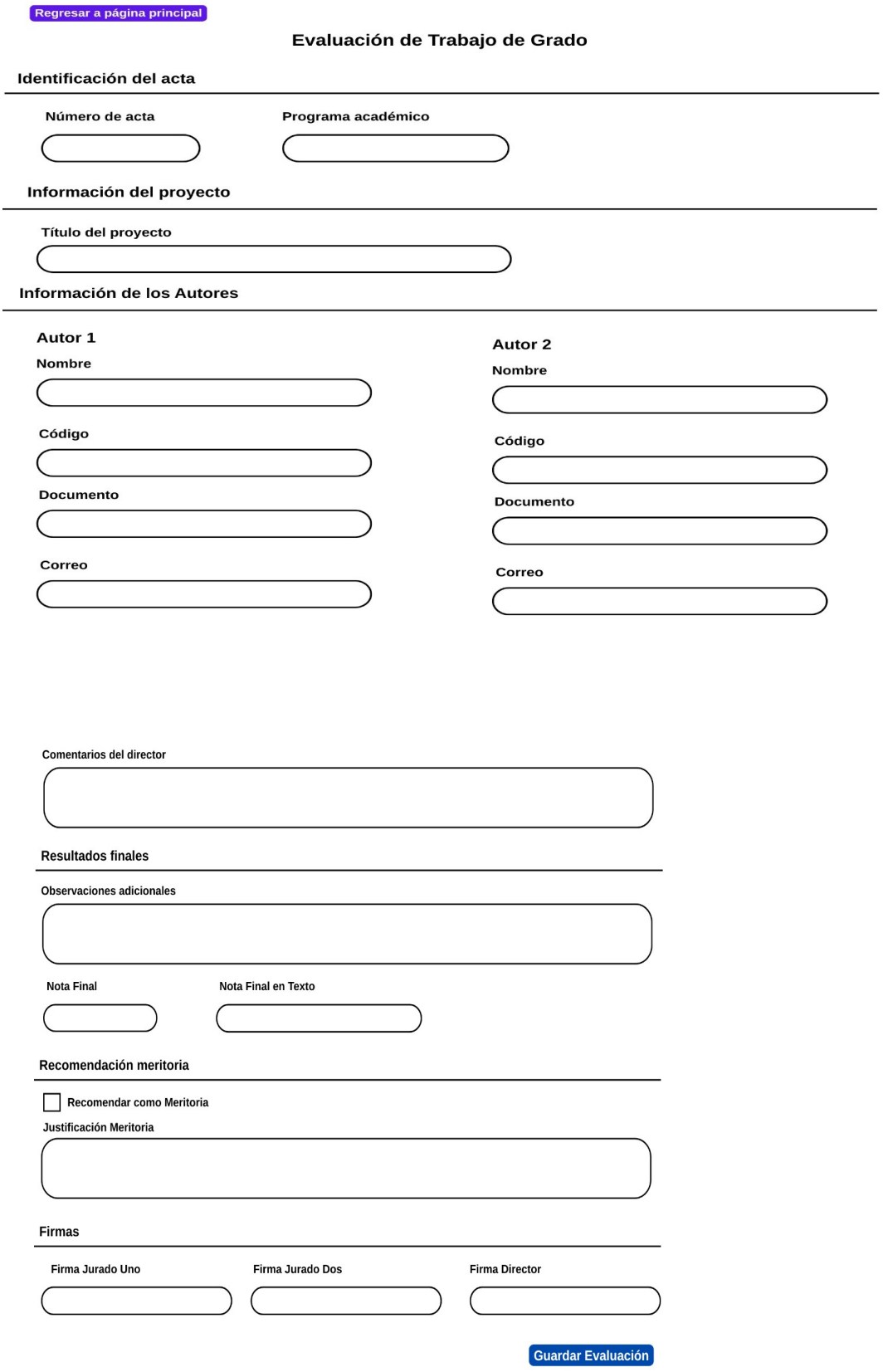

Figura 11: Diseño Crear acta

Editar acta

El diseño en la **Figura 12** corresponde al diseño de la vista "Editar acta", la cual mantiene una estructura similar a la de Çrear acta", pero con la diferencia de que los campos se cargan previamente diligenciados con la información del acta seleccionada desde la pantalla principal. Esta vista se sustenta en RF-006 y RF-007 y en RF-010, garantizando que únicamente los roles autorizados puedan acceder a la opción "Editar". El botón "Actualizar evaluación" confirma los cambios y guarda la información actualizada en la base de datos, retornando posteriormente al listado de actas para continuar con la gestión, en coherencia con el control de acceso por rol definido en RF-012.

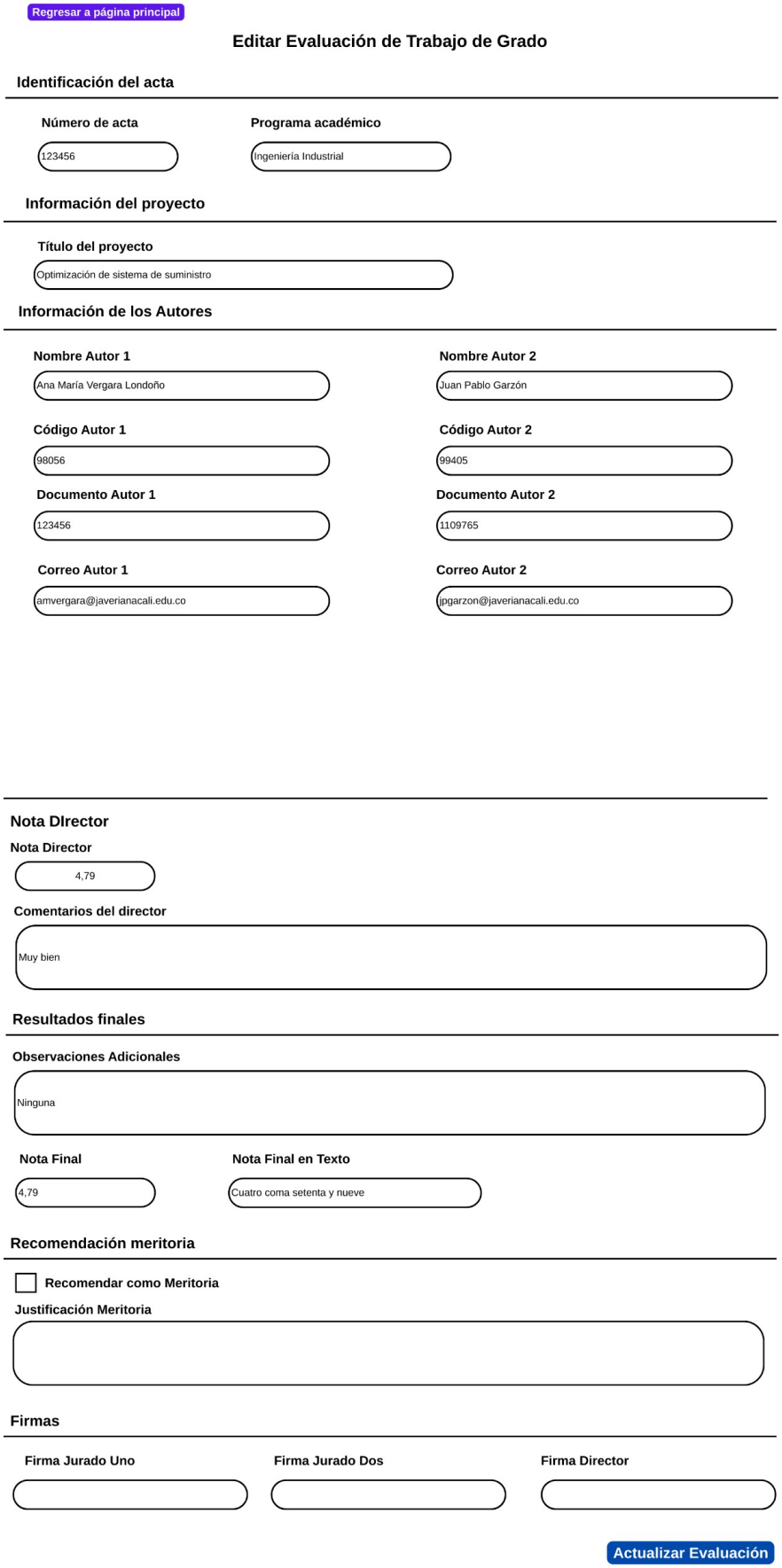

Figura 12: Diseño editar acta

Ver acta

El diseño en la **Figura 13** corresponde al diseño de la vista "Ver acta", para el cual se toma como referencia los requisitos funcionales RF-003 y RF-004, ya que estos describen el comportamiento general del sistema para consultar el listado y acceder al detalle de un acta. En esta interfaz se presenta la información organizada por secciones, facilitando la consulta y lectura del contenido. Adicionalmente, el botón "Regresar a página principal" permite retornar al panel de actas sin modificar el registro, manteniendo el flujo de navegación entre el listado y el detalle.

Figura 13: Diseño Ver acta

Crear usuario

El diseño en la **Figura 14** corresponde al diseño de la vista “Crear usuario”, para el cual se toma como referencia el requisito funcional RF-011, donde se establece que el Administrador debe contar con una opción exclusiva para registrar usuarios en el sistema. En esta pantalla se solicitan los datos necesarios para crear el registro en la base de datos, permitiendo asignar el rol adecuado a cada usuario desde el momento del registro. Finalmente, la vista incluye el botón “Registrar” para confirmar el ingreso de la información y la opción “Volver a actas” para retornar a la pantalla principal sin realizar cambios.

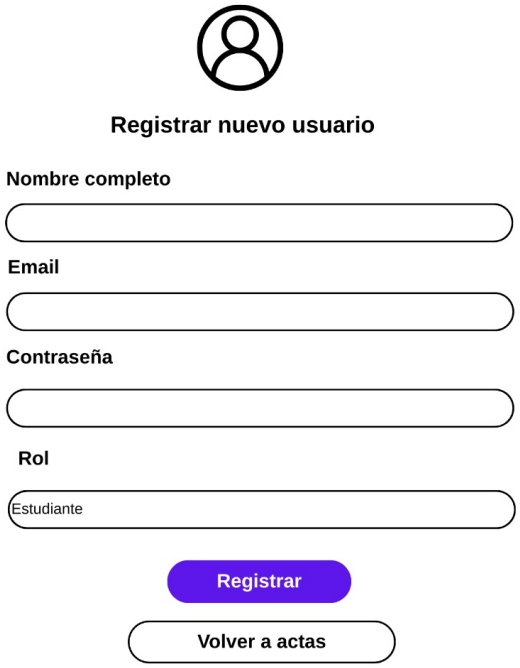

Figura 14: Diseño Crear usuario

6. Desarrollo del sistema de gestión

En este capítulo se abordará la implementación de sus componentes principales y la forma en que se integran para cumplir los requisitos definidos previamente. Para ello, se presentará la construcción de la estructura general del proyecto, detallando desarrollo del backend y el frontend, definiendo las tecnologías utilizadas, la lógica detrás de la implementación y la creación de las vistas.

6.1. Selección de la tecnología para el prototipo

Se realizó la implementación del prototipo utilizando las tecnologías FastApi para el backend y React para el frontend. Esta combinación permitió separar las responsabilidades del sistema, facilitando el mantenimiento del código y la evolución del proyecto a medida que se incorporaron nuevas funcionalidades.

En el backend, FastAPI se eligió por su enfoque en la construcción de APIs REST, su buen rendimiento y la facilidad para definir endpoints de forma clara y modular mediante routers. Además, el uso de modelos con Pydantic permitió validar la información recibida desde el cliente, reduciendo errores y asegurando consistencia en los datos gestionados por el sistema. Para la persistencia de la información se utilizó MySQL, ejecutado en un contenedor Docker, lo cual hizo más sencillo configurar el entorno de desarrollo y administrar la base de datos de manera controlada.

En el frontend, React se seleccionó por su orientación a componentes y por facilitar la creación de interfaces reutilizables y escalables. Adicionalmente, el uso de un sistema de rutas permitió organizar las vistas del prototipo y restringir el acceso según el rol del usuario autenticado. Con esto, se logró construir una interfaz alineada con los mock-ups planteados y conectada al backend mediante solicitudes HTTP para ejecutar las operaciones principales del sistema.

6.1.1. Docker en el desarrollo del prototipo

En el prototipo, Docker se utilizó como soporte de infraestructura para estandarizar el entorno de ejecución y facilitar el despliegue local de los servicios necesarios durante el desarrollo. Su adopción permitió ejecutar componentes del sistema en contenedores configurados de forma reproducible, evitando inconsistencias entre equipos y reduciendo el tiempo asociado a instalaciones manuales y dependencias específicas del sistema operativo.

En particular, Docker fue fundamental para la implementación de la capa de persistencia, ya que la base de datos MySQL se ejecuta como un servicio contenido y aislado. Adicionalmente, se integró phpMyAdmin como herramienta de administración desde el navegador, lo que facilitó la inspección de tablas, validación de registros y ejecución de consultas durante las pruebas del sistema.

Finalmente, para garantizar que la información no se perdiera al detener o recrear contenedores, se configuraron volúmenes (Volumes), permitiendo que los datos de la base de datos se almacenen de forma persistente fuera del ciclo de vida del contenedor. Esto aseguró continuidad en el almacenamiento de usuarios y actas durante el proceso de desarrollo y verificación del prototipo.

6.2. Configuraciones iniciales

El primer paso en el desarrollo del prototipo fue la instalación y configuración de las tecnologías necesarias para cada uno de sus componentes: back-end, front-end y base de datos.

6.2.1. Backend:

Para el componente back-end se utilizó FastAPI, y la configuración inicial del servicio se centralizó en el archivo main.py. En este archivo se crea la instancia principal de la aplicación (FastAPI()), la cual actúa como punto de entrada del servidor y permite integrar los módulos del sistema.

Posteriormente, se configura el middleware CORS (Cross-Origin Resource Sharing), con el fin de habilitar la comunicación entre el front-end (ejecutado en el navegador) y el back-end. Esto es necesario debido a que el front-end y el back-end se ejecutan en orígenes distintos durante el desarrollo (por ejemplo, diferentes puertos), por lo que se define una lista de orígenes permitidos y se habilitan métodos y encabezados para las peticiones HTTP.

Finalmente, en main.py se registran los routers del sistema mediante include_router, separando las funcionalidades por módulos. En particular, se integra el router de autenticación bajo el prefijo /auth y el router de actas bajo el prefijo /acta, permitiendo organizar los endpoints y mantener una estructura modular en el back-end.

```python
from fastapi import FastAPI
from app.router.authentication_router import router as authentication_router
from fastapi.middleware.cors import CORSMiddleware
from app.router.acta_router import router as acta_router

CORS_CONFIG = {
    "allow_origins": ['http://localhost:5173', 'http://localhost', '*'],
    "allow_credentials": True,
    "allow_methods": ["*"],
    "allow_headers": ["*"],
}
/*Crear instancia FastApi*/
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=CORS_CONFIG["allow_origins"],
    allow_credentials=CORS_CONFIG["allow_credentials"],
    allow_methods=CORS_CONFIG["allow_methods"],
    allow_headers=CORS_CONFIG["allow_headers"],
)

app.include_router(authentication_router, prefix="/auth", tags=["login"])
app.include_router(acta_router, prefix="/acta", tags=["acta"])
```

Figura 15: Código para iniciar backend

6.2.2. Frontend:

Para el componente front-end se configuró un entorno de desarrollo basado en Vite, el cual permite iniciar la aplicación web en un servidor local y optimiza el proceso de construcción y recarga automática durante el desarrollo. La interfaz fue implementada con React, organizando la aplicación en componentes reutilizables para facilitar el mantenimiento y la escalabilidad.

En cuanto al diseño visual, se integró TailwindCSS como framework de estilos, permitiendo definir la apariencia de cada vista mediante clases utilitarias directamente en los componentes. Esto agiliza el desarrollo de la interfaz, reduce la necesidad de hojas de estilo extensas y mantiene consistencia visual en todo el sistema.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react-swc'
import tailwindcss from '@tailwindcss/vite'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    tailwindcss(),
    react()
  ],
})
```

Figura 16: Código para configurar el Frontend

6.2.3. Docker:

En el desarrollo del prototipo se utilizó Docker como herramienta clave para estandarizar el entorno de ejecución y facilitar el despliegue local de los componentes del sistema, especialmente la base de datos. Su uso permitió encapsular servicios y dependencias en contenedores ligeros, evitando configuraciones manuales extensas y reduciendo problemas de compatibilidad entre equipos o sistemas operativos.

Para este proyecto se definió la infraestructura describiendo los servicios necesarios para la persistencia y administración de datos. Como se muestra en la **Figura 17**, se levantan dos contenedores principales: uno correspondiente al motor de base de datos MySQL y otro para phpMyAdmin, herramienta utilizada para gestionar la base de datos desde el navegador. Esta configuración establece la comunicación entre ambos servicios, garantizando que phpMyAdmin pueda conectarse directamente al contenedor de MySQL.

Adicionalmente, la configuración expone los puertos necesarios para acceder a los servicios desde el equipo anfitrión: por un lado, se habilita el acceso web a phpMyAdmin y, por otro, se permite la conexión al motor MySQL cuando se requiera interacción externa. De esta manera, Docker aporta un entorno reproducible y controlado que agiliza el desarrollo, pruebas y mantenimiento del prototipo.

```yaml
version: '3'

services:
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin-v1
    ports:
      - "9091:80"  /* Puerto para acceder a phpMyAdmin */
    environment:
      - PMA_HOST=mysql  /* Apunta al contenedor MySQL */
      - PMA_PORT=3306
    networks:
      - server_network
    depends_on:
      - mysql

  mysql:
    image: mysql:8.0
    container_name: mysql-db-v1
    environment:
      - MYSQL_ROOT_PASSWORD= ******
      - MYSQL_DATABASE=mydb
    ports:
      - "9092:3306"
    volumes:
      - mysql_data:'/var/lib/mysql'
    networks:
      - server_network

networks:
  server_network:
    driver: bridge

volumes:
  mysql_data:
    driver: local
```

Figura 17: Código para Docker

6.2.4. Base de datos:

Para la conexión entre el backend y la base de datos, el sistema implementa un mecanismo centralizado que crea y administra la sesión de comunicación con MySQL. Esta configuración permite que los distintos módulos del servidor realicen operaciones de consulta y actualización sobre la base de datos sin duplicar lógica de

conexión, manteniendo el acceso a datos como un componente reutilizable dentro del proyecto.

Como se muestra en la **Figura 17**, el proceso de conexión se realiza indicando los parámetros esenciales: el host (dirección donde está disponible el servicio), el puerto, las credenciales de acceso (usuario y contraseña) y el nombre del esquema utilizado por el sistema. Con estos datos, el servidor establece la comunicación con MySQL para poder ejecutar las operaciones necesarias sobre las tablas del sistema.

```python
import mysql.connector
from mysql.connector import Error
from contextlib import contextmanager
import os
from typing import Generator

@contextmanager
def get_db_connection() -> Generator[mysql.connector.MySQLConnection, None, None]:
    conn = None
    try:
        conn = mysql.connector.connect(
            host="localhost",
            port=9092,
            user="root",
            password="******",
            database="mydb"
        )
```

Figura 18: Código para configurar conexión de la base de datos

6.3. Desarrollo de los modelos

Durante el desarrollo del prototipo, los modelos del sistema se definieron directamente a nivel de base de datos mediante scripts SQL. La creación de los modelos se basaron en la **Figura 8**.Esta decisión permitió establecer de forma explícita la estructura de las entidades principales y sus restricciones, asegurando que el almacenamiento de la información quedara alineado con los datos requeridos por el sistema.

Se utilizaron dos archivos SQL independientes para inicializar el esquema. El archivo SECRET.sql contiene la definición del modelo de usuarios (Users), incluyendo los campos necesarios para el inicio de sesión y la administración de roles dentro del sistema. Por su parte, el archivo SECRECTEVALUATION.sql define el modelo correspondiente a las actas de evaluación (evaluacion_tesis), incorporando los campos que componen el formulario de evaluación, así como restricciones orientadas a mantener la integridad de la información almacenada.

Una vez elaborados estos scripts, se importaron directamente a la base de datos SQL para crear las tablas y dejar el esquema listo para su uso por parte del backend. De esta manera, el prototipo parte de una estructura de datos consistente desde el inicio, facilitando el registro, consulta y actualización de la información según las funcionalidades implementadas.

6.4. Definición de las rutas

En el prototipo, las rutas representan los puntos de acceso a las funcionalidades del sistema mediante endpoints HTTP. Cada endpoint define una URL y un método (GET, POST, PUT o DELETE) que permite al cliente web ejecutar una acción específica, como en la **Figura 19** se puede observar los métodos usados para crear, obtener, listar, actualizar o eliminar, un acta en el modelo Evaluacion_Tesis.

Para mantener una estructura modular, las rutas se agrupan por funcionalidad en dos módulos principales: autenticación y actas. Estos módulos se integran en la aplicación con prefijos que facilitan su identificación: /auth para operaciones de autenticación y /acta para operaciones relacionadas con evaluaciones. De esta manera, el sistema expone endpoints organizados y coherentes, lo cual facilita su mantenimiento y escalabilidad.

```python
router = APIRouter()

@router.post("/evaluaciones/")
async def crear_evaluacion(evaluacion: EvaluacionTesis):
    try:
        await ActaService.crear_acta(evaluacion)
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=str(e)
        )

@router.get("/evaluaciones/{numero_acta}" )
async def obtener_evaluacion(numero_acta: str):
    return await ActaService.obtener_acta(numero_acta)

@router.get("/evaluaciones")
async def listar_evaluaciones():
    return await ActaService.listar_actas()

@router.put("/evaluaciones/{numero_acta}")
async def actualizar_evaluacion(numero_acta: str, evaluacion: EvaluacionTesis):
    try:
        return await ActaService.actualizar_acta(numero_acta, evaluacion.dict())
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=str(e)
        )

@router.delete("/evaluaciones/{numero_acta}")
async def eliminar_evaluacion(numero_acta: str):
    try:
        return await ActaService.eliminar_acta(numero_acta)
    except Exception as e:
        raise HTTPException(
            status_code=500,
            detail=str(e)
        )
```

Figura 19: Código con ejemplo de endpoints

Express maneja las solicitudes HTTP internamente y siempre necesita saber qué petición recibió y cómo se debe responder. Por esta razón, siempre que se define una función en una ruta, Express recibe automáticamente los parámetros “req” (solicitud) y “res” (respuesta). Estos parámetros permiten capturar los datos que envía el cliente y transformarlos de acuerdo con lo que necesite.

6.5. Desarrollo de los controladores

En el prototipo, authentication_service cumple el rol de controlador para el módulo de autenticación, ya que concentra la lógica encargada de validar el acceso de los usuarios al sistema y habilitar el uso de funcionalidades según su rol. Este componente actúa como enlace entre el frontend y el backend en el proceso de inicio de sesión: recibe las credenciales enviadas desde la vista de login, verifica su validez y genera la respuesta que permite continuar el flujo de navegación dentro del sistema.

6.5.1. Controlador de autenticación

Dentro de sus responsabilidades se encuentra la verificación de credenciales contra la información almacenada en la base de datos, así como el manejo de errores cuando los datos ingresados no son correctos, como se puede ver en la **Figura 20** que define las funciones para iniciar sesión con un usuario o registrar un nuevo usuario.

```
def login(email: str, password: str):
    user = user_repository.find_user_by_email(email)

    if user is None:
        raise ValueError("Username or password incorrect")

    stored_password = user['password'].encode('utf-8')
    if not bcrypt.checkpw(password.encode('utf-8'), stored_password):
        raise ValueError("Username or password incorrect")

    token = generate_jwt_token(user)

    return {
        "access_token": token,
        "token_type": "bearer", "user": user
    }

def register_user(email: str, name: str, password: str, rol: str):
    salt = bcrypt.gensalt()
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), salt)

    existing_user = user_repository.find_user_by_email(email)

    if existing_user:
        raise ValueError("User already exists")

    user_id = user_repository.create_user(email, name, hashed_password, rol)
```

Figura 20: Código para iniciar sesión o registrar usuario

Adicionalmente, se encarga de la emisión de un token JWT, el cual permite mantener la sesión activa y asociar el usuario autenticado (incluyendo su rol) a las solicitudes posteriores que realiza el cliente. De esta forma, el sistema puede controlar el acceso a funcionalidades protegidas y mantener coherencia con la restricción de permisos definida en los requisitos funcionales. La **Figura 21** muestra la implementación del código para la generación de JSON Web Token (JWT).

```
import jwt

def generate_jwt_token(user_data: Dict) -> str:
    """Generate JWT token for user"""
    payload = {
        'user_id': user_data['id'],
        'email': user_data['email'],
        'rol': user_data['rol'],
        'exp': datetime.utcnow() + JWT_EXPIRATION_DELTA
    }
    return jwt.encode(payload, JWT_SECRET, algorithm=JWT_ALGORITHM)
```

Figura 21: Código para generar el JWT

6.5.2. Controlador de actas

En el prototipo, el archivo acta_service cumple el rol de controlador para el módulo de actas, ya que centraliza la lógica necesaria para atender las solicitudes provenientes del front-end relacionadas con la gestión de evaluaciones. Su función principal es orquestar el flujo entre las rutas del backend y la capa de acceso a datos: recibe los datos enviados desde la interfaz, aplica reglas de validación y coherencia, y delega las operaciones de persistencia al repositorio correspondiente.

En este controlador se implementan las operaciones principales del sistema sobre el recurso evaluacion_tesis, tales como crear, obtener, listar, actualizar y eliminar actas. De esta manera, acta_service actúa como un componente intermedio que garantiza que cada acción iniciada por el usuario en la interfaz (por ejemplo, guardar una evaluación o actualizar un acta existente) se traduzca en una operación controlada en el backend, manteniendo separada la lógica del sistema respecto al acceso directo a la base de datos.

```
class ActaService:
    @staticmethod
    async def crear_acta(datos_acta) -> Dict:
        try:
            datos_dict = datos_acta.model_dump()

            acta_existente = ActaRepository.buscar_acta_por_numero
            (datos_dict['numeroActa'])
            if acta_existente:
                raise HTTPException(status_code=400,
                detail="Ya existe un acta con este número")

            if 'notaFinal' not in datos_dict:
                datos_dict['notaFinal'] =

                ActaService._calcular_nota_final(datos_dict)

            if 'notaFinalTexto' not in datos_dict:
                datos_dict['notaFinalTexto'] =

                ActaService._convertir_nota_a_texto
                (datos_dict['notaFinal'])

            nueva_acta = ActaRepository.crear_acta(datos_dict)
            if not nueva_acta:
                raise HTTPException(status_code=500,
                detail="Error al crear el acta")

            return ActaService._convertir_a_pydantic_model
            ([nueva_acta])[0].dict()
```

Figura 22: Código con lógica de creación de un acta

6.6. Desarrollo de las vistas

En este capítulo se describe la implementación de las vistas del prototipo, las cuales fueron construidas a partir de los mock-ups presentados previamente en 5.3.2 (Mock-ups para las vistas) y alineadas con los requisitos funcionales definidos en 4.5 (Requisitos funcionales). Las interfaces se desarrollaron con un enfoque de navegación por rutas, permitiendo el acceso secuencial a las pantallas del sistema según el siguiente flujo: Inicio de sesión, panel principal y funcionalidades administrativas.

La vista de inicio de sesión permite la autenticación mediante correo y contraseña, validando la necesidad de los campos antes de continuar. En caso de credenciales inválidas o datos incompletos, el sistema presenta mensajes de error que impiden el acceso y mantienen al usuario en la misma pantalla, cumpliendo con los requisitos

de autenticación y control de acceso. La **Figura 23** muestra el formulario de inicio de sesión y evidencia la retroalimentación al usuario cuando existen errores de validación.

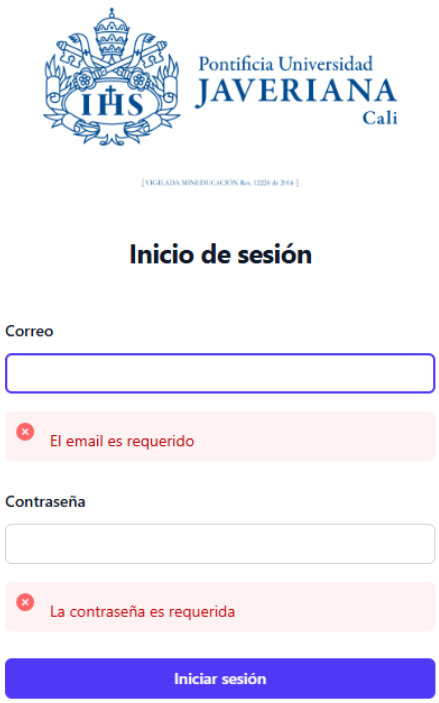

Figura 23: Vista Final Inicio de sesión (Mensajes de error)

Posteriormente, la página principal funciona como punto central del sistema, presentando una tabla con el listado de actas registradas y las acciones disponibles para cada registro. Adicionalmente, desde esta vista se habilitan accesos directos a funciones como la creación de actas, el registro de usuarios y el cierre de sesión. La habilitación de botones y acciones se ajusta al rol del usuario autenticado, garantizando que cada perfil visualice únicamente las opciones permitidas. La **Figura 24** presenta el panel principal implementado.

Figura 24: Vista Final Página principal

Finalmente, la vista de registro de usuario permite el ingreso de la información

requerida para almacenar un nuevo usuario en el sistema, funcionalidad reservada para el rol administrador. La **Figura 25** muestra el formulario final de registro.

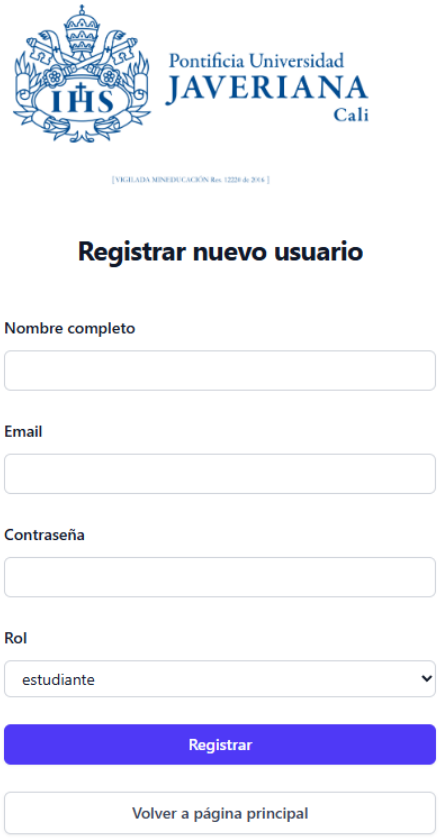

Figura 25: Vista Final Crear usuario

Las demás vistas finales en el prototipo funcional se realizaron con base en los mock-ups creados en la sección de diseño. Estas vistas finales son las de crear, editar y ver acta, debido a su longitud, se podrán encontrar en el capítulo de anexos del documento. 11

7. Pruebas del sistema de gestión

En este capítulo se presenta la validación del prototipo del sistema de gestión, con el fin de comprobar que sus funcionalidades operan correctamente y que el comportamiento del sistema coincide con los requisitos funcionales definidos. Para ello, se realizaron pruebas enfocadas en la interacción del usuario con el sistema y en los resultados observables durante la ejecución de las principales acciones.

7.1. Pruebas de caja negra

En este capítulo se presenta la estrategia de pruebas de caja negra aplicada a la plataforma, con el objetivo de verificar que cada funcionalidad cumpla con los requerimientos funcionales (RF) definidos, sin considerar la implementación interna (código, estructuras o lógica interna). Este enfoque se centra en evaluar el sistema desde la perspectiva del usuario, validando el comportamiento observable a partir de entradas, acciones y resultados.

La **Figura 26** resume el conjunto de casos ejecutados, organizados por identificador (ID), Módulo, precondición, pasos, resultado obtenido y su clasificación como válido (V) o no válido (NV), dependiendo de si el comportamiento del sistema coincide o no con lo esperado según los requisitos asociados.

Con esto se puede identificar de manera clara qué requerimientos fueron verificados exitosamente y cuáles presentan desviaciones que requieren ajustes o correcciones.

ID	Módulo	RF	Caso de prueba	Precondición	Pasos	Resultado obtenido	Tipo
L-01	Login	RF-001	Iniciar sesión con email y contraseña válidos	Usuario registrado	Ingresar email válido; Ingresar contraseña correcta; Presionar Iniciar sesión	Acceso exitoso a la página principal	V
L-02	Login	RF-002	Email válido y contraseña incorrecta	Email registrado	Ingresar email válido; Ingresar contraseña incorrecta	"Error inesperado al autenticar"	NV
L-03	Login	RF-002	Email no registrado	Ninguna	Ingresar email inexistente; Ingresar contraseña	"Error inesperado al autenticar"	NV
L-04	Login	RF-002	Email vacío	Ninguna	Dejar email vacío; Ingresar contraseña	"El email es requerido"	NV
L-05	Login	RF-002	Contraseña vacía	Ninguna	Ingresar email válido; Dejar contraseña vacía	"La contraseña es requerida"	NV
L-06	Login	RF-002	Iniciar sesión sin datos	Ninguna	Dejar email y contraseña vacíos; Presionar iniciar sesión	"El email es requerido" y "La contraseña es requerida"	NV
R-01	Registro	RF-011	Registrar usuario con datos válidos	Administrador logueado	Completar formulario y guardar	Registro exitoso y regreso a la página principal	V
R-02	Registro	RF-011	Registrar usuario sin email	Administrador logueado	Dejar campo email vacío	"El email es requerido"	NV
R-03	Registro	RF-011	Registrar usuario sin contraseña	Administrador logueado	Dejar campo contraseña vacío	"La contraseña es requerida"	NV
R-04	Registro	RF-011	Registrar usuario sin nombre	Administrador logueado	Dejar campo nombre vacío	"El nombre es requerido"	NV
PP-01	Página principal	RF-012	Cerrar sesión	Usuario cualquiera logueado	Presiona el botón "Cerrar sesión"	Regreso a la vista Login	V
PP-02	Página principal	RF-003	Visualizar listado actas	Usuario cualquiera logueado	Iniciar sesión	Vista del listado de actas	V
PP-03	Página principal	RF-004	Ver detalle de un acta	Usuario cualquiera logueado	Presionar la acción "Ver" del acta deseada	Vista de acta detallada	V
PP-04	Página principal	RF-008	Eliminar acta	Secretaria o administrador logueado	Presionar la acción "Eliminar" del acta deseada	Actualiza página principal y acta deseada desaparece	V
PP-05	Página principal	RF-005	Acceder a editar acta	Profesor, secretaria o administrador logueado	Presionar la acción "Editar" del acta deseada	Vista para editar acta	V
EA-01	Editar acta	RF-006	Profesor edita acta	Profesor logueado	Profesor edita campos de acta y presiona "Actualizar evaluación"	Regreso a la vista página principal; datos actualizados	V
EA-02	Editar acta	RF-007	Secretaria edita acta	Secretaria logueado	Secretaria edita campos de acta y presiona "Actualizar evaluación"	Regreso a la vista página principal; datos actualizados	V
EA-03	Editar acta	RF-010	Administrador edita acta	Administrador logueado	Administrador edita campos de acta y presiona "Actualizar evaluación"	Regreso a la vista página principal; datos actualizados	V
EA-04	Editar acta	RF-012	Editar acta con campos requeridos vacíos	Profesor, secretaria o administrador logueado	Borrar datos de campos requeridos y presionar "Actualizar evaluación"	Mensaje "es requerido" sobre los campos vacíos	NV
EA-05	Editar acta	RF-012	Editar acta con campos no requeridos vacíos	Profesor, secretaria o administrador logueado	Borrar datos de campos no requeridos y presionar "Actualizar evaluación"	Regreso a la vista página principal; datos actualizados	V
CA-01	Crear acta	RF-009	Secretaria crea acta	Secretaria logueado	Secretaria completa el formulario y presiona "Guardar evaluación"	Crea un acta y regresa la página principal	V
CA-02	Crear acta	RF-010	Administrador crea acta	Administrador logueado	Administrador completa el formulario y presiona "Guardar evaluación"	Crea un acta y regresa la página principal	V
CA-03	Crear acta	RF-012	Crear acta con campos vacíos	Secretaria o administrador logueado	Presiona el botón "Crear acta"; Guardar acta con campos vacíos	Mensaje "es requerido" sobre los campos vacíos	NV
S-01	Seguridad	RF-012	Estudiante usa acciones distintas a "Ver"	Estudiante logueado	Presionar una acción distinta a "Ver"	No aparecen las acciones distintas a "Ver"	NV
S-02	Seguridad	RF-012	Profesor usa acciones distintas a "Ver" y "Editar"	Profesor logueado	Presionar una acción distinta a "Ver" o "Editar"	No aparecen las acciones distintas a "Ver" o "Editar"	NV
S-03	Seguridad	RF-012	Secretaria trata de registrar un usuario nuevo	Secretaria logueado	Presionar la acción "Crear usuario"	No aparece acción "Crear usuario"	NV
S-04	Seguridad	RF-012	Administrador ve todas las acciones	Administrador logueado	Iniciar sesión; observar en página principal acciones disponibles	Aparecen todas las acciones posibles	V

Figura 26: Pruebas de caja negra

7.2. Pruebas con el usuario

Como prueba final, se presentó el prototipo a la secretaria de la facultad de ingeniería, quien tiene el perfil de usuario del rol "Secretaria.[en] el prototipo. Este perfil se seleccionó debido a que cuenta con acceso a las funcionalidades principales del prototipo, incluyendo la gestión de actas y, adicionalmente, el registro de nuevos usuarios.

Antes de empezar la presentación del prototipo se le explicó la funcionalidad de registro de usuarios y el perfil autorizado para acceder a ella. El objetivo de esta prueba fue simular el proceso completo de gestión de un acta de evaluación, permitiendo que la usuaria interactuara con el prototipo y ejecutara las acciones disponibles, con el fin de evaluar su experiencia y recoger observaciones sobre aspectos positivos y oportunidades de mejora.

Para recopilar esta información, al finalizar la prueba se solicitó a la secretaria responder un conjunto de preguntas. Sus respuestas, junto con los comentarios realizados durante el desarrollo de la actividad, permitieron consolidar la retroalimentación obtenida mediante un formulario. Las respuestas obtenidas están documentadas en la imagen de la **Figura 27**.

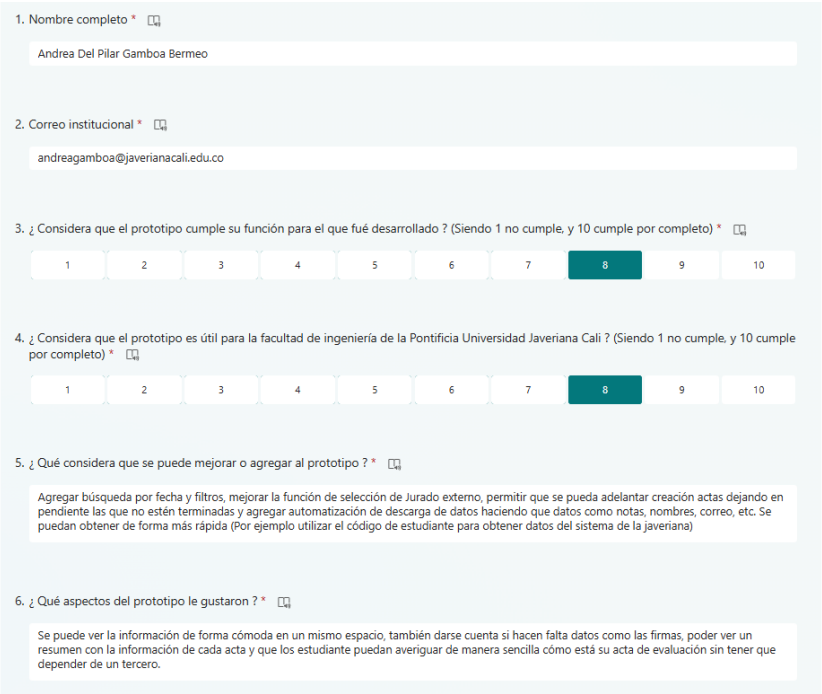

Figura 27: Formulario con retroalimentación del usuario

8. Conclusiones

A partir del desarrollo del prototipo y del análisis de los resultados obtenidos durante su implementación y validación, en esta sección se presentan las conclusiones del trabajo realizado. Estas recogen los principales aportes del proyecto en relación con la gestión de actas de evaluación en ingeniería, así como los aprendizajes que resultaron del proceso de desarrollo del sistema.

Primero, se desarrolló la plataforma web que permitió mejorar la gestión de las actas de evaluación en ingeniería, al proporcionar una herramienta eficiente y segura para registrar, consultar y administrar actas de evaluación académica. La solución propuesta centraliza la información asociada a las evaluaciones, facilita su acceso por parte de estudiantes, evaluadores y directores académicos, secretarias de facultad, y contribuye a mejorar el seguimiento de los procesos evaluativos.

Además, la plataforma incorpora una interfaz web intuitiva que permite a los perfiles de usuario con autorización registrar, consultar y modificar actas de evaluación de manera eficiente. El diseño de las vistas y la estructura de navegación favorecen una interacción clara con el sistema, lo que permite tener una experiencia de uso más accesible y una ejecución ágil de las tareas principales relacionadas con la evaluación de los trabajos de grado.

También se desarrolló una arquitectura de base de datos robusta que facilita la gestión centralizada y optimizada de la información relacionada con los trabajos de grado. Este modelo soporta el almacenamiento estructurado de usuarios y actas de evaluación, garantizando la consistencia, disponibilidad y facilidad de actualización de los datos, y estableciendo una base técnica adecuada para el mantenimiento y la escalabilidad del sistema.

Asimismo, el sistema incorpora mecanismos de seguridad y control de acceso que permiten proteger la confidencialidad de los datos y garantizar la integridad del sistema. A través de procesos de autenticación y autorización basados en roles, se restringen las funcionalidades disponibles según el perfil del usuario, asegurando que cada actor interactúe únicamente con las operaciones correspondientes a sus responsabilidades dentro del proceso académico.

Finalmente, la validación de la plataforma mediante las pruebas funcionales de caja negra y verificación de control de acceso por roles, junto con la elaboración de documentación técnica y una guía de usuario, permitió comprobar el correcto funcionamiento del prototipo y su alineación con los requisitos definidos.

9. Trabajo futuro

En esta sección se hace un análisis de la implementación y desarrollo del prototipo funcional, y con este se describen propuestas que pueden ampliar el alcance del sistema y mejorar aspectos técnicos en versiones posteriores.

Realización del despliegue y configuración, creando un flujo mediante el cuál se pueda levantar el sistema de gestión completo en entornos distintos al local.En especial, el objetivo sería buscar un entorno web que esté relacionado con uno de los sistemas de la universidad Javeriana Cali, como one gate o brightspace.

Por último, incluir funcionalidades adicionales al sistema de gestión. Incluir una forma de exportar las actas de evaluación en diferentes formatos, una barra de busqueda avanzada y filtros para cuando el sistema crezca a un gran volumen de datos. Y notificaciones al correo registrado en el sistema cuando se realicen cambios sobre actas relacionadas con el usuario.

10. Bibliografía

Referencias

[1] Ian Sommerville, *Software Engineering [Ninth Edition]*, 2011, pp. 24 - 73. Disponible en: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[2] Amaya, J. E. (2025). *Sistema de gestión para trabajos de grado en posgrado.*, 2009. Disponible en: http://hdl.handle.net/11522/4666

[3] Martínez, P. H. and Mena, S. (2024). *Sistema para el seguimiento de trabajos de grado.*, 2009. Disponible en: http://hdl.handle.net/11522/1898

[4] S. L. Raul, P. Kevadiya, Z. Khan and J. Pereira, .ª React.js-Based Approach for Enhancing Content Management and User Experience in Dynamic Web Applications,"2025 International Conference on Engineering Innovations and Technologies (ICoEIT), Bhopal, India, 2025, pp. 89-94. Disponible en: https://ieeexplore.ieee.org/document/11211620

[5] N. Dahmani, S. A. Alex, S. G. Sadhana, S. G. Jayasree and T. P. A. Jinu, "Welcome Wagons: A Block Chain based Web Application for Car Booking,"2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 2022, pp. 1-6. Disponible en: https://ieeexplore.ieee.org/document/10017821

[6] A. Neumann, N. Laranjeiro and J. Bernardino, .ªn Analysis of Public REST Web Service APIs,ïn IEEE Transactions on Services Computing, vol. 14, no. 4, pp. 957-970, 1 July-Aug. 2021. Disponible en: https://ieeexplore.ieee.org/document/8385157

[7] D. Bernstein, Çontainers and Cloud: From LXC to Docker to Kubernetes,ïn IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014. Disponible en: https://ieeexplore.ieee.org/document/7036275

[8] R. Deari, X. Zenuni, J. Ajdari, F. Ismaili and B. Raufi, .ªnalysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL,"2018 International Conference on Information Technologies (InfoTech), Varna, Bulgaria, 2018. Disponible en: https://ieeexplore.ieee.org/document/8510719

[9] E. Mabotha, N. E. Mabunda and A. Ali, "Performance Evaluation of a Dynamic RESTful API Using FastAPI, Docker and Nginx,"2024 Global Energy

Conference (GEC), Batman, Turkiye, 2024, pp. 174-181. Disponible en: https://ieeexplore.ieee.org/document/10881712

[10] S. Ahmed and Q. Mahmood, .An authentication based scheme for applications using JSON web token,"2019 22nd International Multitopic Conference (INMIC), Islamabad, Pakistan, 2019, pp. 1-6. Disponible en: https://ieeexplore.ieee.org/document/9022766

11. Anexos

Figura 28: Diseño final del Inicio de sesión

Figura 29: Diseño final de página principal

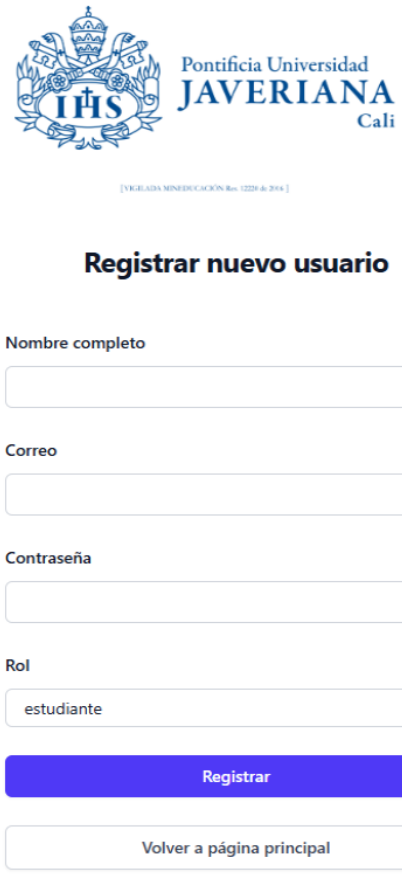

Figura 30: Diseño final de crear usuario

Figura 31: Diseño final de ver acta

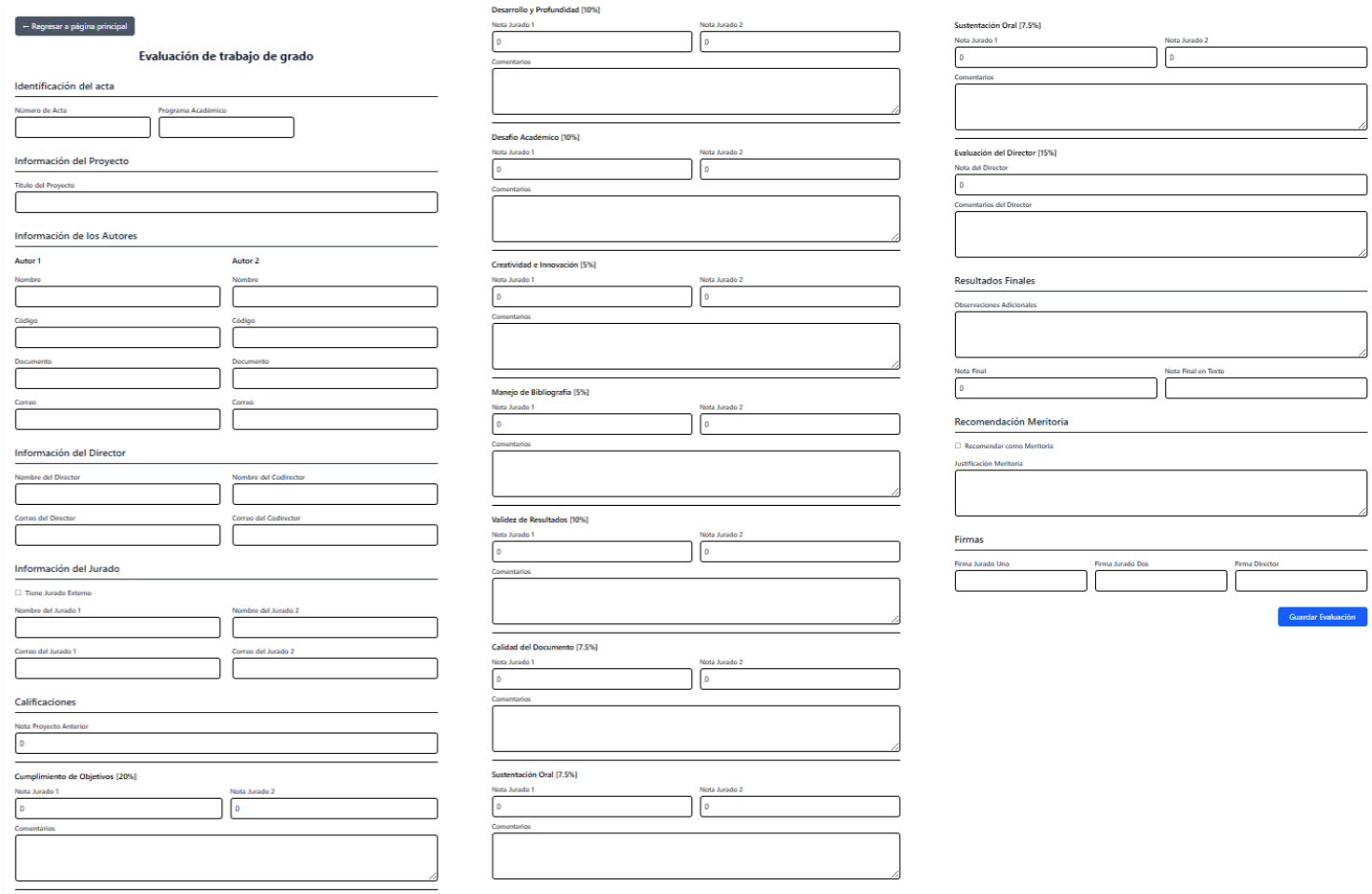

Figura 32: Diseño final de crear acta

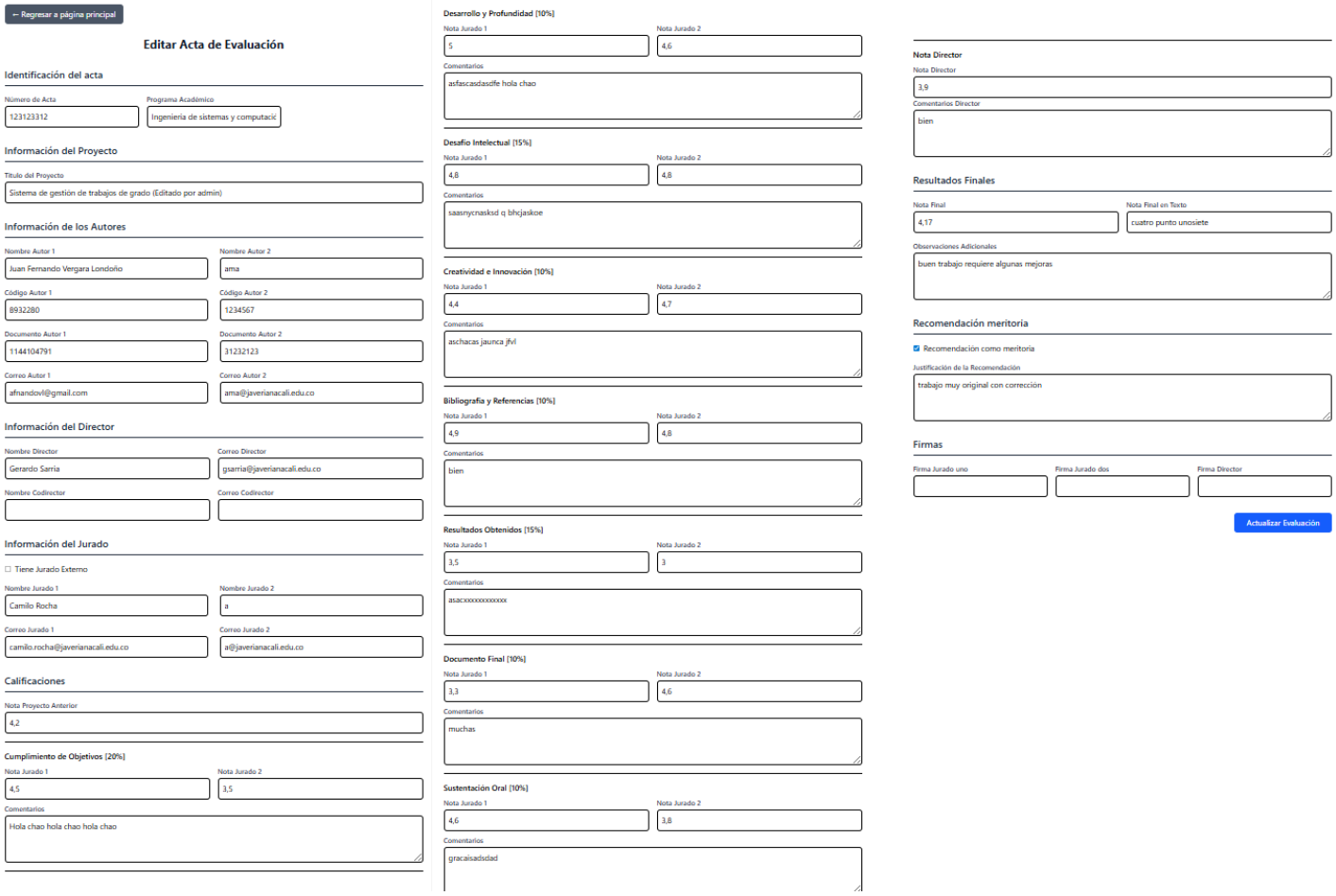

Figura 33: Diseño final de editar actas