Aplicativo móvil para la gestión por parte del estudiante de su trabajo de grado en la plataforma Comunidad Académica (COMA)

Brajhan Javier Rivera González

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director
Luis Ignacio González Ramírez
Magíster en Informática

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingeniería de Sistemas e Informática
Ingeniería de Sistemas
Bucaramanga
2026

Dedicatoria

Dedico este trabajo de grado, primero que todo a Dios, por permitirme terminarlo y darme la fortaleza necesaria en momentos difíciles en donde estuve a punto de desfallecer.

A mis padres, Marigen González y Javier Rivera, por su amor incondicional, su apoyo constante en todo momento, por inculcarme el valor del esfuerzo y la perseverancia. Su confianza en mí fueron un pilar fundamental en esta fase de mi vida.

A mis hermanos, por su alegría y cariño hacía mí, que me ven como un ejemplo a seguir.

A mi amigo, Cristian Jaimes, por su apoyo constante, por estar en las buenas y en las malas durante todo mi proceso académico.

A ti, por haber sido una parte muy importante en mi vida, también fuiste crucial para poder cumplir esta meta.

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a las y personas que fueron un pilar fundamental durante mi formación y en la realización de este trabajo de grado.

En primer lugar, a la Universidad Industrial de Santander y a la Escuela de Ingeniería de Sistemas e Informática, por ser mi casa de estudios y por brindarme una fase académica enriquecedora que ha sido clave en mi crecimiento personal y profesional.

Extiendo mi gratitud a todos los docentes que, a lo largo de mi carrera, compartieron generosamente sus conocimientos y experiencias. Cada una de sus enseñanzas ha contribuido a forjar las bases sobre las cuales hoy construyo mi futuro como ingeniero.

Asimismo, quiero ofrecer un agradecimiento especial a Ovy por brindarme la oportunidad de crecer profesionalmente. La experiencia práctica adquirida durante mi tiempo en la compañía fue invaluable y me permitió aplicar dichos conocimientos directamente en el desarrollo de este trabajo de grado. Agradezco a mis mentores y colegas por su confianza y por todo lo que aprendí a su lado.

Tabla de Contenido

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Resumen

Título: Aplicativo móvil para la gestión por parte del estudiante de su trabajo de grado en la plataforma Comunidad Académica (COMA)*

Autor: Brajhan Javier Rivera González**

Palabras Clave: Aplicativo móvil, comunidad académica, coma, trabajo de grado, COMA, gestión.

Descripción:

El presente trabajo de grado documenta el desarrollo de un aplicativo móvil orientado a la gestión del trabajo de grado por parte de los estudiantes en la plataforma Comunidad Académica (COMA) de la Universidad Industrial de Santander. Actualmente, dicho proceso se realiza exclusivamente a través de la versión web de la plataforma, lo que limita la accesibilidad de los estudiantes.

Para el desarrollo del proyecto se utilizó Flutter, un framework de código abierto de Google que permite crear aplicaciones nativas para Android a partir de un único código base programado en el lenguaje Dart. La arquitectura de la aplicación se compone de un front-end, que corresponde a la interfaz con la que interactúa el usuario, y un back-end, que gestiona la lógica de negocio y los datos. La comunicación entre ambos componentes se establece a través de una API REST, que funciona como un puente para garantizar una sincronización efectiva y segura con la plataforma COMA.

Entre los objetivos del desarrollo se incluyó la implementación de funcionalidades esenciales para el estudiante, tales como la consulta del estado del trabajo de grado, la inscripción del tema y la carga de documentos como el plan de trabajo y el libro final. Adicionalmente, se facilitó la interacción con el director y los coautores en las diferentes etapas del proceso. Finalmente, el proyecto contempló el diseño de una interfaz de usuario intuitiva y accesible, junto con la implementación de mecanismos de seguridad para proteger la integridad de la información académica.

---

* Trabajo de Grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de Sistemas. Director: Luis Ignacio González Ramírez. Magíster en Informática.

Abstract

Title: Mobile Application for Student Management of their Degree Project on the Comunidad Académica (COMA) Platform*

Author: Brajhan Javier Rivera González**

Key Words: Mobile app, academic community, COMA, thesis, management

Description:

This degree project presents the development of a mobile application for student management of their degree project on the Comunidad Académica (COMA) platform at the Universidad Industrial de Santander. Currently, the management of these projects is carried out exclusively through the web version of the COMA platform, which limits student access.

For the project's development, Flutter was used, an open-source framework by Google that allows for the creation of native applications for both Android and iOS from a single codebase, programmed in the Dart language. The application's architecture is composed of a front-end, which is the interface the user interacts with, and a back-end, which manages the business logic and data. Communication between both components is established through an API, which acts as a bridge to ensure effective and secure synchronization with the COMA platform.

Among the specific development objectives is the implementation of essential functionalities for the student. The application will allow users to check the status of their degree project in real-time, register their topic, and upload documents such as the work plan and the final paper. Additionally, it will be essential to facilitate interaction with the director and authors throughout the different stages of the process. Finally, the project includes designing an intuitive and accessible user interface, along with implementing security mechanisms to protect the integrity of academic information.

*Degree Work

**Faculty of Physical-Mechanical Engineering. School of Systems Engineering and Informatics. Systems Engineering. Advisor: Luis Ignacio González Ramírez. Master in Computer Science.

Introducción

La gestión del trabajo de grado representa una fase crucial en la formación de los estudiantes de la Universidad Industrial de Santander, siendo un proceso académico de alta importancia para culminar su carrera profesional. Históricamente, la universidad ha soportado este proceso a través de la plataforma Comunidad Académica (COMA), una herramienta web integral desarrollada y mantenida desde el año 2003 por el Grupo de Innovación y Desarrollo CALUMET. En respuesta a las necesidades actuales de acceso móvil, el propósito fundamental de este trabajo fue trasladar la funcionalidad de gestión de trabajos de grado a un aplicativo móvil plenamente funcional e integrado con la plataforma COMA.

Dentro del ámbito académico, el aprovechamiento de las Tecnologías de la Información y Comunicación (TIC) es fundamental, ya que ofrece una amplia gama de oportunidades para optimizar procesos que en épocas pasadas eran complejos. Entre sus beneficios se encuentran la facilitación del acceso a la información, la promoción de la participación activa de estudiantes y docentes, y la personalización de los procesos de enseñanza.

Por lo tanto, el desarrollo de esta aplicación se enfocó en materializar funcionalidades esenciales que permitan al estudiante consultar el estado de su trabajo de grado, realizar la inscripción del tema y cargar documentos clave como el plan de trabajo y el informe final, facilitando a su vez la interacción con el director y los coautores.

1. Planteamiento y Justificación del Problema

La plataforma COMA, si bien ha sido una herramienta fundamental, presenta una limitación significativa en la era digital actual, ya que la gestión de los trabajos de grado se realiza exclusivamente mediante un ordenador de escritorio. Esta dependencia genera una barrera de accesibilidad para los estudiantes, quienes, en una sociedad caracterizada por la necesidad de acceso a la información desde cualquier lugar, requieren mayor flexibilidad para gestionar sus proyectos académicos. Esta situación conduce a la siguiente pregunta de investigación: *¿Cómo se puede mejorar la accesibilidad y disponibilidad del servicio de gestión del trabajo de grado?*

La justificación para desarrollar esta aplicación móvil radica en el impacto positivo y tangible que genera. El proyecto otorga a los estudiantes la capacidad de revisar el estado de su trabajo de grado en tiempo real, permitiéndoles tomar decisiones pertinentes y realizar ajustes oportunos, lo que se traduce en un aumento de su productividad y en la agilización de los procesos académicos. Para la universidad, esta iniciativa contribuye a la modernización y evolución de sus servicios, mejorando la experiencia estudiantil al facilitar el acceso a recursos académicos relevantes en cualquier momento y lugar.

2. Objetivos

2.1 Objetivo General

Llevar a un aplicativo móvil la funcionalidad de gestión de trabajos de grado por parte del estudiante en la plataforma COMA.

2.2 Objetivos Específicos

- Desarrollar las funcionalidades necesarias en el aplicativo móvil para permitir a los estudiantes gestionar su trabajo de grado, incluyendo:
  - La consulta del estado del trabajo de grado.
  - La inscripción del tema.
  - Subir el plan del trabajo de grado y la interacción tanto con el director como con los autores.
  - Subir el documento final y la interacción entre los autores y el director o calificadores asignados por el comité.
- Integrar el aplicativo móvil con la plataforma COMA, garantizando una sincronización efectiva y segura de los datos y permitiendo una experiencia de usuario fluida y consistente.
- Implementar mecanismos de seguridad y privacidad en el aplicativo móvil, asegurando la protección de la información sensible relacionada con los trabajos de grado y el resguardo de la integridad de los datos.
- Diseñar la interfaz de usuario del aplicativo móvil para la gestión del trabajo de grado, asegurando su usabilidad y accesibilidad para los estudiantes.
- Realizar pruebas del aplicativo móvil, verificando su funcionalidad, rendimiento y compatibilidad.

- Documentar de manera clara y detallada todo el proceso de desarrollo del aplicativo móvil, incluyendo la arquitectura, las tecnologías utilizadas, los desafíos enfrentados y las soluciones implementadas, con el fin de contar con un recurso de referencia para futuras actualizaciones y mejoras.

3. Marco de Referencia

3.1 Fundamentos Teóricos

En el marco de referencia de este proyecto, se abordarán los fundamentos teóricos relacionados con el desarrollo de aplicativos móviles, la gestión de trabajos de grado y la importancia de la usabilidad.

- **Desarrollo de aplicativos móviles**: Es el proceso de creación de software para distintos dispositivos móviles tales como, celulares, tabletas o asistentes digitales, con mayor frecuencia para sistemas operativos Android o iOS. La aplicación puede venir instalada por defecto en el dispositivo o simplemente descargarse desde un gestor de aplicaciones. Los lenguajes de programación utilizados para este tipo de desarrollo de software incluyen Java, Swift, C# y lenguajes de marcación como HTML5.
    - **Modelo híbrido:** Dentro del desarrollo de aplicativos móviles existe el modelo híbrido, el cual permite adoptar el enfoque “escribir una vez, ejecutar en cualquier lugar”. Estás aplicaciones utilizan un único código fuente que puede funcionar en cualquier plataforma o sistema operativo.
    - **Arquitectura**: Según Microsoft (2023), la arquitectura consta de tres capas:

- **Capa de presentación:** La que el usuario ve, con la que interactúa, es decir, la interfaz.
- **Nivel de negocio:** Lógica que gobierna los flujos de trabajo. Donde se gestiona la seguridad, almacenamiento, registro y gestión de excepciones.
- **Nivel de datos:** Donde se validan y mantienen los datos, dando utilidad a los mismos y soporte a las transacciones de estos.

- **Front-end:** Se relaciona con la capa de presentación, las herramientas y lenguajes que se pueden utilizar depende de los dispositivos en donde se implementará la aplicación.
- **Back-end:** Donde se incluye la base de datos y lógica que reside en el servidor.
- **Api**: Es el puente de comunicación entre el front-end y el back-end, las apis permiten que la aplicación pueda interactuar con diversos servicios.
- **Flutter:** Es un framework de desarrollo de código abierto creado por Google que se ha convertido en una herramienta poderosa para la creación de aplicaciones móviles multiplataforma. Una de sus principales ventajas es la rapidez de desarrollo, ofrece una experiencia de usuario excepcional, por medio de interfaces de usuario personalizables y atractivas. Además, permite a los desarrolladores crear aplicaciones móviles nativas para Android y iOS desde un solo código base. Esto ahorra tiempo y recursos, ya que no es necesario desarrollar y mantener dos aplicaciones separadas. El SDK de

Flutter está basado en el lenguaje de programación Dart, el cual, también fue desarrollado por Google (Google, s.f.).

  - **Dart**: Es un lenguaje de programación open source creado por Google en el año 2011, con el objetivo de hacer el proceso de desarrollo más fácil y rápido para los desarrolladores. Por eso cuenta con ciertas herramientas que lo hacen especial, como su propio gestor de paquetes, varios compiladores, un analizador y formateador. Además, posee características tan útiles como la compilación Just-in-time que permite observar en tiempo real los cambios realizados en el código. Por esta razón Flutter optó por Dart como su lenguaje de programación, ya que ofrece una sintaxis clara y concisa, facilitando el desarrollo de aplicaciones eficientes y confiables. (Felgo, s.f.).

- **Gestión de proyectos de grado:** La gestión de proyectos es la disciplina responsable de la planificación, organización, dirección y control de los recursos y actividades necesarios para lograr los objetivos específicos del proyecto dentro de un marco de tiempo determinado. Implica el uso de técnicas, herramientas y habilidades de gestión para lograr resultados exitosos.
- **Usabilidad e interfaz de usuario**: Según la ISO 25000 (2014), la usabilidad es la capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:
  - **Reconocibilidad de la adecuación:** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades

- **Aprendizabilidad**: Capacidad del producto que permite al usuario aprender su aplicación.
- **Operabilidad:** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Protección contra errores de usuario**: Capacidad del sistema para proteger a los usuarios de errores.
- **Estética de la interfaz de usuario**: Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad:** Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

3.2 Antecedentes del Tema

En cuanto a los antecedentes sobre sistemas de gestión de trabajos de grado podemos referenciar:

3.2.1 Plataforma web COMA

La plataforma web Comunidad Académica, desarrollada y soportada por el Grupo de Innovación y Desarrollo CALUMET de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander desde el año 2003, es un sistema integral que ha sido diseñado para facilitar una amplia gama de procesos académicos. Entre sus funciones principales, destaca la gestión de los trabajos de grado, un aspecto crítico en el proceso de formación de los estudiantes y de investigación de los profesores.

Desde su creación, Comunidad Académica ha evolucionado continuamente para adaptarse a las necesidades cambiantes de la comunidad educativa. A través de esta plataforma, tanto estudiantes como profesores pueden llevar a cabo una variedad de actividades relacionadas

con los trabajos de grado, incluyendo la presentación de los temas, el seguimiento de avances, la recepción de retroalimentación, etc.

La plataforma ofrece una interfaz intuitiva y fácil de usar, diseñada para garantizar que todos los usuarios puedan acceder y utilizar sus funciones de manera efectiva.

A lo largo de los años, Comunidad Académica se ha convertido en una herramienta fundamental para la comunidad académica de la Universidad Industrial de Santander, facilitando la colaboración entre estudiantes y profesores, promoviendo la excelencia en la investigación y contribuyendo al desarrollo académico y profesional de sus usuarios.

4. Metodología

Para el desarrollo del aplicativo móvil para la gestión del trabajo de grado en la plataforma Comunidad Académica (COMA), se utilizó la metodología ágil Scrum, adecuada para el modelo de desarrollo iterativo e incremental que se quiere seguir, enfocando fundamentalmente en el desarrollo colaborativo, la adaptabilidad y entrega de valor de forma temprana y eficiente.

Esta metodología se basa en ciclos cortos (sprints), que suelen tener una duración de 1 a 4 semanas. Cada sprint se inicia con una reunión de planificación, donde se seleccionan las tareas a realizar, y finaliza con una revisión del trabajo realizado en una reunión de revisión del sprint.

Al utilizar Scrum, se puede dividir el desarrollo del aplicativo móvil en iteraciones más pequeñas y manejables, lo que facilita la adaptación a cambios en los requisitos y permite una entrega continua de funcionalidades. Esto garantiza una mayor flexibilidad y capacidad de respuesta a medida que se obtiene retroalimentación de los usuarios y se identifican nuevas necesidades.

La metodología Scrum se adapta bien al desarrollo de aplicativos móviles, ya que permite una mayor interacción con los usuarios y una retroalimentación constante durante todo el proceso. Además, promueve la transparencia y la comunicación efectiva dentro del equipo de desarrollo y con las partes interesadas, lo que facilita la identificación y resolución temprana de problemas.

4.1 Implementación de Scrum

- **Definición de la lista de trabajo**: Se identifican y priorizan las funcionalidades del aplicativo móvil, creando una serie de tareas que guiarán el desarrollo.
- **Planificación del sprint**: Se seleccionan las tareas a realizar en el sprint, considerando la capacidad en tiempo y las prioridades de la lista de trabajo. Se definen los objetivos y los entregables esperados al final del sprint.
- **Desarrollo iterativo**: Se trabaja en la implementación de las funcionalidades seleccionadas para el sprint. Se realizan reuniones programadas para mantener la comunicación y la sincronización.
- **Reunión de revisión del sprint**: Al finalizar el sprint, se presenta el trabajo realizado a las partes interesadas y se obtiene su retroalimentación. Se evalúa si se cumplen los objetivos establecidos y se realizan ajustes necesarios para los próximos sprints.
- **Reunión de retrospectiva**: Se reflexiona sobre el sprint realizado, identificando aspectos que se pueden mejorar y definiendo acciones concretas para el siguiente sprint. Se busca continuamente la mejora del proceso y la optimización del trabajo en equipo.

5. Desarrollo del Proyecto

5.1 Arquitectura

Para el desarrollo del aplicativo, se implementó una arquitectura cliente-servidor basada en el modelo de tres capas: presentación, negocio y datos. Esta estructura permite una clara separación de responsabilidades, facilitando el mantenimiento y la escalabilidad del sistema.

5.1.1 Servidor

El servidor, denominado *coma_movil_server*, fue construido sobre el entorno de ejecución **Node.js** y el framework **Express.js**. Su diseño sigue un estricto patrón de Arquitectura en Capas lo que asegura una separación de responsabilidades bien definida. La estructura del código se organiza en los siguientes directorios principales:

Figura 1

Estructura principal de directorios del servidor.

```
src/
├── controllers/     # Lógica de negocio y manejo de requests
├── middlewares/     # Middleware de autenticación y autorización
├── models/          # Modelos de datos con Sequelize ORM
├── repositories/    # Capa de acceso a datos
├── routes/          # Definición de rutas y endpoints
├── schemas/         # Validación de datos con Zod
├── helpers/         # Funciones utilitarias
├── app.js          # Configuración principal de Express
├── database.js     # Gestión de conexiones a BD
└── db-config.js    # Configuración multi-tenant
```

- **Controllers:** Manejan las solicitudes (requests) y respuestas (responses) HTTP, orquestando la lógica de negocio.

- **Repositories:** Abstraen el acceso a la base de datos, centralizando todas las consultas y operaciones de datos.
- **Models:** Definen la estructura de los datos utilizando el ORM **Sequelize**.
- **Middlewares:** Son las piiezas del software que procesan las solicitudes de forma secuencial, utilizadas principalmente para la autenticación y autorización.
- **Routes:** Definen los endpoints de la API y los asocian con sus respectivos controladores.
- **Schemas:** Contienen las reglas de validación de datos de entrada utilizando la librería **Zod**.

5.1.1.1 Diseño de Arquitectura Multi-Tenant. Una de las características más avanzadas del servidor es su arquitectura multi-tenant, diseñada para dar servicio a las diferentes escuelas académicas de la Universidad de forma aislada y segura. Cada escuela opera con su propia base de datos independiente en MySQL, lo que garantiza la integridad y privacidad de los datos.

Este diseño se implementa a través de un sistema de configuración centralizado (db-config.js) que mapea cada "tenant" (escuela) a sus credenciales de base de datos. Se gestiona un pool de conexiones inteligente que crea y reutiliza conexiones de manera eficiente y bajo demanda para cada escuela, optimizando el rendimiento y el uso de recursos.

5.1.1.2 Patrones de Diseño Implementados. Para garantizar un código robusto, mantenible y escalable, se implementaron varios patrones de diseño clave:

- **Patrón Repositorio**: Se utiliza para desacoplar la lógica de negocio del acceso a los datos. Clases como *ThesisProjectRepository* centralizan las consultas a la base de datos, permitiendo cambiar la fuente de datos sin afectar otras partes del sistema.
- **Patrón Factory**: Se emplea para la creación dinámica de instancias de modelos de Sequelize. La función *getUserModel()* es un ejemplo que permite generar un modelo de usuario conectado a la base de datos correcta según la escuela del tenant que realiza la solicitud.
- **Patrón Middleware**: Fundamental en Express.js, se usa para construir una cadena de procesamiento para la seguridad. Un ejemplo es la secuencia authenticateToken > studentAuthMiddleware > requireServices, donde cada paso verifica un aspecto diferente (validez del token, rol del usuario, permisos específicos) antes de llegar al controlador.
- **Patrón Strategy**: Se aplica en la selección dinámica de la conexión a la base de datos. El sistema elige la "estrategia" de conexión correcta basándose en el identificador del tenant.
- **Patrón Singleton**: Se utiliza para gestionar el pool de conexiones de la base de datos, asegurando que solo exista una instancia del pool por cada escuela, evitando la sobrecarga de crear conexiones repetidamente.

5.1.1.3 Seguridad, Autenticación y Autorización. La seguridad es un pilar del sistema, implementada a través de múltiples mecanismos:

- **Autenticación Stateless**: Se basa en JSON Web Tokens (JWT). Tras un inicio de sesión exitoso, el servidor genera un token firmado que el cliente almacena y envía en cada solicitud posterior para autenticarse.
- **Autorización Granular**: Se implementó un sistema de autorización por servicios, donde cada ruta de la API está mapeada a un código de servicio específico (ej. S123 para crear una tesis). El middleware de autorización verifica que el rol del usuario tenga permiso para acceder a dicho servicio antes de procesar la solicitud.
- **Validación de Entrada**: Se utiliza Zod para definir esquemas estrictos que validan la estructura y el tipo de todos los datos que ingresan a la API, previniendo ataques de inyección y datos corruptos.

5.1.1.4 Calidad y Mantenibilidad. Para asegurar la calidad a largo plazo del código, se adoptaron prácticas de desarrollo profesional:

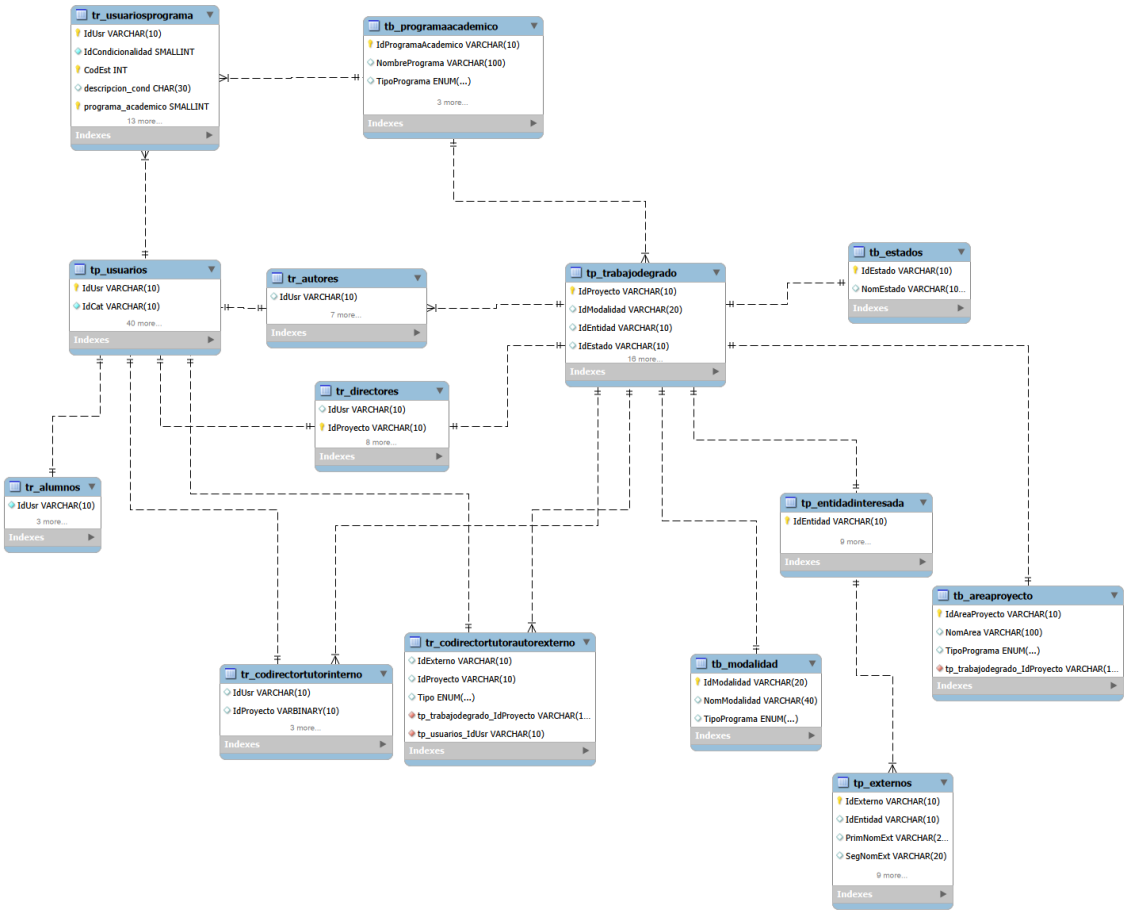
- **Calidad de Código:** Se utilizó el linter **StandardJS** para mantener un estilo de código consistente y limpio en todo el proyecto.
- **Modularidad:** El código está organizado en módulos con responsabilidades únicas, siguiendo el principio de "Separation of Concerns".
- **Gestión de Configuración:** Se utiliza el paquete dotenv para gestionar las variables de entorno, permitiendo configurar la aplicación para diferentes entornos (desarrollo, producción) sin modificar el código fuente.
- **Containerización:** El proyecto incluye un Dockerfile para crear una imagen de Docker con Node.js Alpine, facilitando el despliegue y la portabilidad del servidor.

5.1.1.5 Modelo de datos. Como se observa en la **Figura 2**, el diagrama Entidad-Relación (ER) detalla la estructura de las tablas principales, tales como tp_trabajodegrado, que actúa como la entidad central vinculando el estado del proyecto, la modalidad y el área académica. Asimismo, se destacan las tablas transaccionales como tr_autores y tr_directores, las cuales permiten asociar múltiples actores a un mismo proyecto.

Este modelo refleja las entidades principales del sistema y sus interrelaciones, permitiendo la gestión eficiente de usuarios, roles y el ciclo de vida de los proyectos de grado.

Figura 2

Modelos entidad relación



5.1.2 Cliente

El cliente fue desarrollado con el framework Flutter, diseñada para ofrecer una experiencia de usuario nativa, fluida y reactiva en sistemas operativos tales como Android y iOS. La arquitectura de la aplicación se centró en la modularidad, la separación de responsabilidades y un manejo de estado eficiente para garantizar su escalabilidad y mantenibilidad.

Figura 3

Estructura principal de directorios del cliente.

```
lib/
│   main.dart                    # Punto de entrada de la aplicación
│
├── common/                      # Componentes reutilizables
├── controllers/                 # Controladores de estado (GetX)
├── models/                     # Modelos de datos
├── screens/                    # Pantallas de la aplicación
└── utils/                     # Utilidades y helpers
```

5.1.2.1 Gestión de Estado con GetX. La arquitectura de la aplicación sigue un patrón que combina elementos de MVC (Modelo-Vista-Controlador) y MVVM (Modelo-Vista-VistaModelo), estructurado principalmente por la biblioteca GetX. GetX fue seleccionada no solo como una solución para la gestión de estado, sino como un micro-framework integral que simplifica la inyección de dependencias, la gestión de rutas y la programación reactiva.

5.1.2.2 Comunicación con el Back-end y Manejo de Errores. La comunicación entre la aplicación Flutter y el servidor Node.js se gestiona a través de una capa de red centralizada, robusta y estructurada. Esta capa fue diseñada no solo para realizar peticiones HTTP, sino para

automatizar la autenticación, interceptar respuestas y manejar errores de manera consistente en toda la aplicación, garantizando una clara separación de responsabilidades y una experiencia de usuario fluida.

- **Capa de Red Centralizada**: El núcleo de la comunicación reside en JHttpHelper, un mixin implementado con métodos estáticos que funciona bajo el patrón Singleton. Este diseño asegura que exista un único punto de acceso global para todas las operaciones de red, lo que permite centralizar la configuración, el estado de autenticación y la lógica de las peticiones. Las características principales de JHttpHelper son:
  - **Gestión de Autenticación Automática y Transparente**: El cliente HTTP gestiona de forma interna y privada un _accessToken. Antes de enviar cualquier petición, añade automáticamente este token a la cabecera Cookie, liberando a los controladores y a otras partes de la aplicación de la responsabilidad de manejar manualmente la autenticación.
  - **Soporte Completo de Métodos HTTP**: Proporciona métodos estáticos para todas las operaciones estándar (GET, POST, PATCH, DELETE), así como una función especializada uploadFiles. Esta función soporta la subida de archivos multipart/form-data, permitiendo enviar campos de texto adicionales junto con los archivos y detectando automáticamente el tipo MIME correcto.
- **Sistema de Manejo de Errores Personalizado**: Para manejar los errores de forma estructurada y consistente con el back-end, se creó la clase AppException.

Esta clase encapsula los errores devueltos por la API en un mapa, lo que permite manejar múltiples errores asociados a campos específicos (por ejemplo, errores de validación en un formulario). Cuando una respuesta HTTP indica un error que no es un simple mensaje informativo, JHttpHelper lanza una AppException, permitiendo que el código que originó la llamada (generalmente un controlador) la capture y reaccione de manera específica, como mostrar un mensaje de error debajo de un campo de texto.

- **Interceptores de Respuesta y Notificaciones Automáticas:** Para proporcionar feedback inmediato al usuario, se implementó un sistema de interceptores que actúa como un middleware sobre las respuestas HTTP. La clase ResponseInterceptor tiene la responsabilidad de procesar automáticamente todas las respuestas exitosas.

  Su funcionalidad principal es la detección automática de flash messages. El back-end puede incluir en su respuesta JSON un objeto flash con mensajes de éxito, error, advertencia o información. El interceptor transforma este objeto y utiliza un sistema de notificaciones (MySnackbar) para mostrar un mensaje visual al usuario.

  Las notificaciones son dinámicas e inteligentes, puesto que, hasta que el ciclo de vida del widget haya sido completado la notificación no se mostrará para así evitar conflictos en el renderizado.

  Los snackbars también adaptan su apariencia (color de fondo, icono, color del texto) según el tipo de mensaje (éxito, error, etc.), garantizando una experiencia visual consistente y accesible.

5.1.2.3 Programación Reactiva y Componentes de UI. La reactividad de la interfaz se logró mediante el patrón observador, implementado por GetX a través de sus variables observables (.obs). Tipos como RxBool, RxString y RxList permiten que los widgets se reconstruyan automáticamente cuando el valor de una variable de estado cambia, sin necesidad de llamar actualizar su estado manualmente.

Adicionalmente, se siguió un patrón de componentes, creando una biblioteca de widgets personalizados y reutilizables (con el prefijo J, como JAppBar, encargado de mostrar una app bar personalizado), lo que garantiza la consistencia visual y acelera el desarrollo de nuevas pantallas.

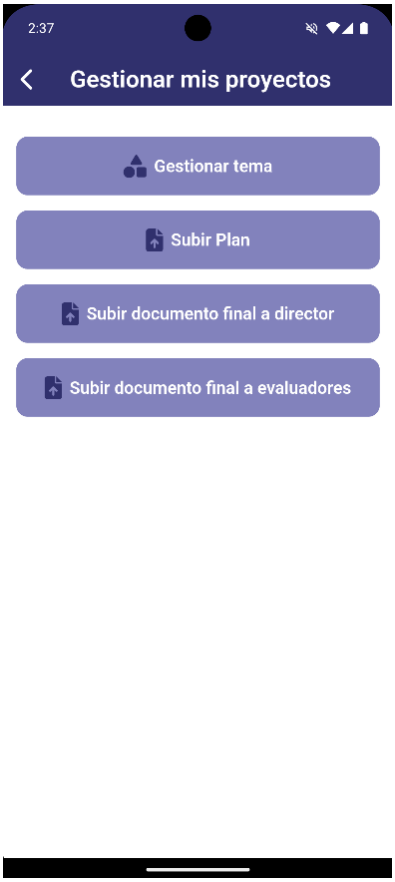
5.1.2.4 Sistema de Diseño y Dependencias Clave. Se implementó un sistema de tematización centralizado (JAppTheme) basado en Material Design 3, lo que permite una gestión unificada de la paleta de colores para las distintas escuelas de la Universidad, tipografías y estilos de los widgets.

5.2 Módulos y Funcionalidades Principales

El objetivo principal de la aplicación es permitir a los estudiantes de la Universidad Industrial de Santander gestionar su propio proyecto de grado, para llevar a cabo esta gestión se desarrollaron las siguientes funcionalidades principales:

Figura 4

Menús de "Gestionar mis proyectos".



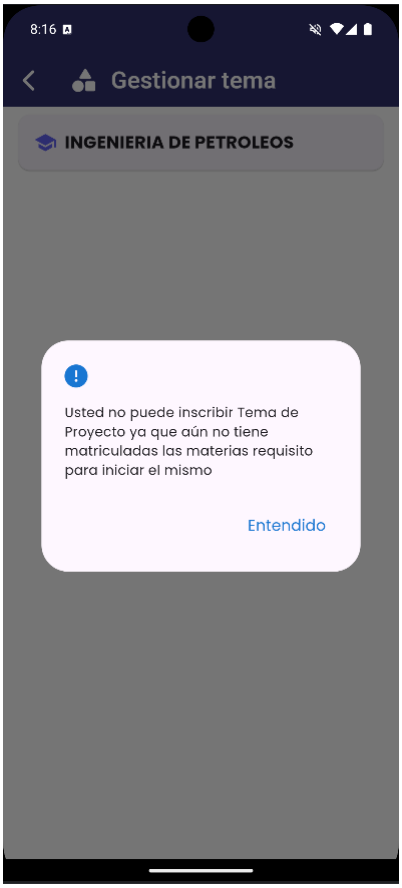
5.2.1 Módulo de Gestión de Temas de Proyecto

Este es el módulo central donde comienza el ciclo de vida del trabajo de grado. Permite al estudiante proponer, desarrollar y gestionar la aprobación de su tema de proyecto.

5.2.1.1 Verificación de Elegibilidad E Inscripción del Tema. El proceso inicia con una verificación de elegibilidad automática que consulta al servidor para determinar el estado del estudiante. El sistema valida si este cumple los requisitos académicos para inscribir un tema, si ya tiene uno en proceso o si no es elegible, mostrando en cada caso la interfaz apropiada.

Figura 5

No elegible para inscribir tema.



Si el estudiante está habilitado, accede a un proceso de inscripción guiado por pasos que estructura la captura de información de manera organizada e intuitiva. A lo largo de este asistente, el estudiante debe detallar lo siguiente:

- **Información General:** El proceso inicia con los datos básicos, solicitando el **título** del proyecto y permitiendo definir su **modalidad** y los **coautores** que participarán.
- **Fundamentación y Metodología:** En pasos sucesivos, se profundiza en el núcleo del proyecto, detallando el **objetivo general y los específicos**, la **justificación**, el plan de **actividades** a desarrollar, los **resultados o productos** esperados y las **palabras clave** para su clasificación.

- **Equipo de Apoyo Académico:** Se designa el equipo que guiará el trabajo. Mediante selecciones múltiples avanzadas, se vincula al **director**, **codirectores** (internos y externos), **tutores** y se especifica el **área de conocimiento** a la que pertenece el proyecto.
- **Revisión y Confirmación:** Al finalizar, el sistema presenta un **resumen completo** de toda la información ingresada, permitiendo al estudiante verificar todos los datos antes de confirmar y enviar la propuesta.

Figura 6

Información general

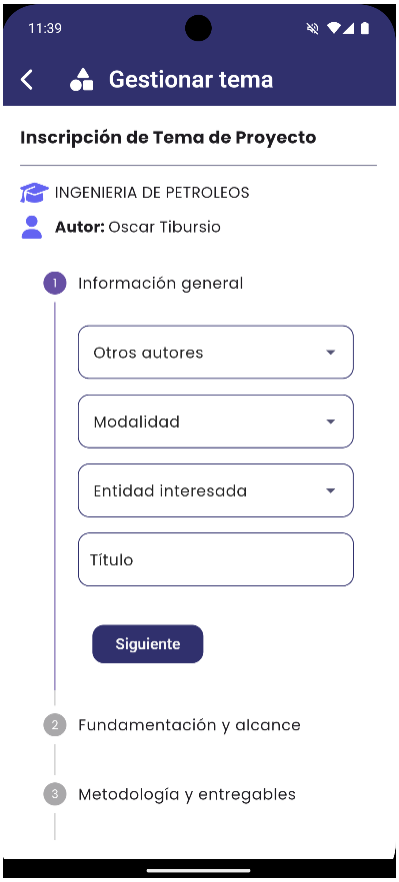


Figura 7

Fundamentación y alcance.

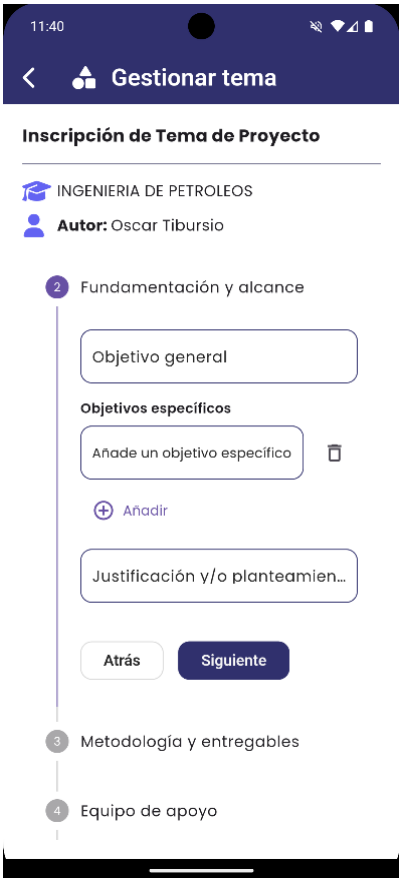


Figura 8

Metodología y entregables.

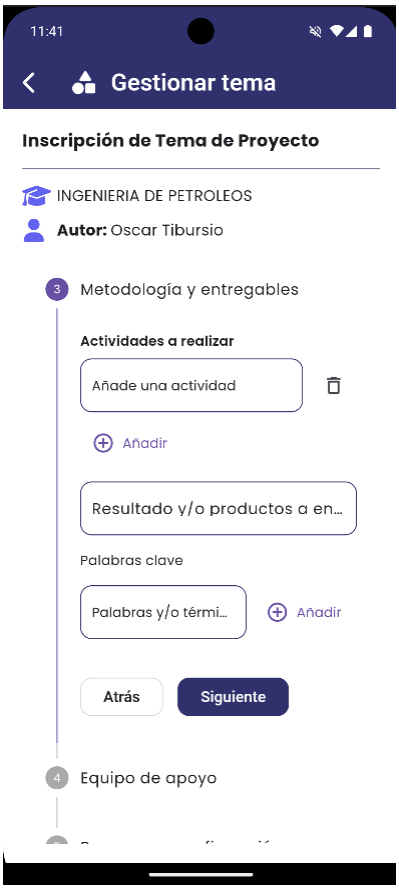
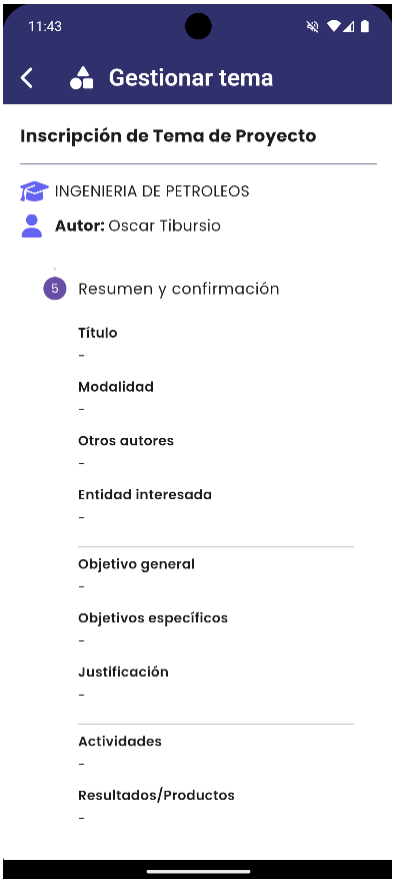


Figura 9

Equipo de apoyo.

Figura 10

Resumen y confirmación.



El formulario integra validaciones para todos los campos obligatorios, asegurando la integridad de los datos antes de su creación.

Figura 11

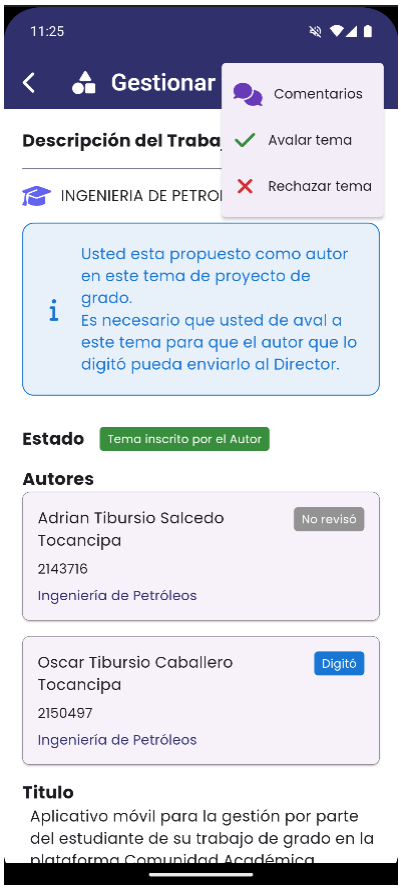
*Campo **Modalidad** es requerido.*

5.2.1.2 Gestión de Coautores y Proceso de Aval. Cuando un proyecto incluye coautores, se activa un flujo de aval por parte de este coautor, fundamental para el proceso.

- **Notificación y Solicitud**: Al ser añadido a una propuesta, cada coautor recibe una notificación vía correo electrónico y la solicitud de avalar su participación.
- **Revisión y Decisión**: El coautor puede revisar todos los detalles del tema propuesto y decidir si **avala** o **rechaza** su inclusión.

Figura 12

Pantalla del coautor propuesto a un tema y las acciones disponibles sobre este.

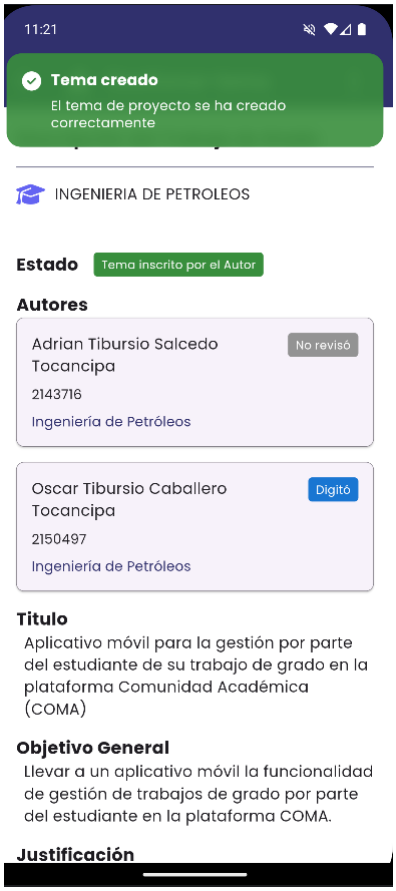


- **Bloqueo Condicional**: El autor principal no puede enviar el tema a revisión del director hasta que todos los coautores propuestos hayan otorgado su aval. Esto asegura el consenso del equipo desde el inicio.

5.2.1.3 Flujo de Gestión.

Figura 13

Tema inscrito correctamente.



Una vez el tema está completo y con todos los avales, se habilitan las siguientes acciones:

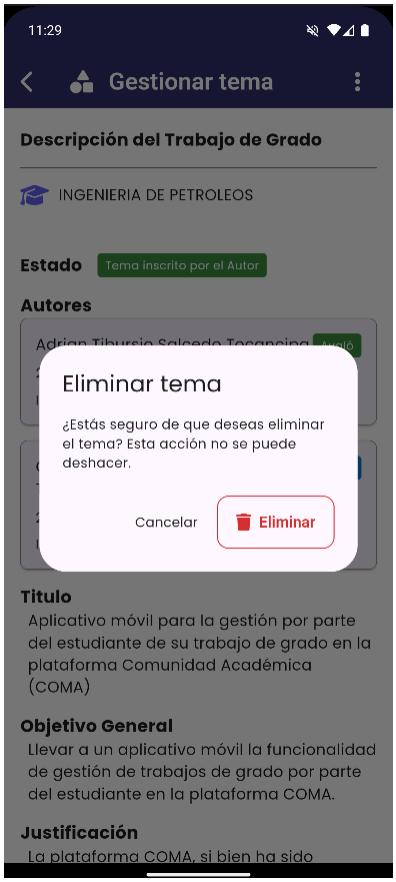
- **Envío a director**: El estudiante envía la propuesta, lo que cambia su estado a "Tema para aval del director" y bloquea la edición para prevenir modificaciones durante la evaluación.
- **Retiro de revisión**: Si el estudiante necesita realizar una corrección importante antes de que el director evalúe, tiene la opción de retirar el tema de revisión, lo que lo devuelve al estado "Tema inscrito por el Autor".
- **Comunicación bidireccional**: Se habilita un sistema de comentarios contextualizado, que sirve como canal de comunicación oficial entre el

estudiante y el director para discutir observaciones, realizar consultas y solicitar correcciones.

- **Modificación del tema**: El autor que digitó el tema puede modificar algunos campos específicos del tema según el estado de este.
- **Eliminación del tema**: El autor que digitó el tema tiene la opción de eliminarlo si así lo requiere. En caso de tener coautores, al eliminar el tema se les notificará a estos sobre la acción realizada.

Figura 14

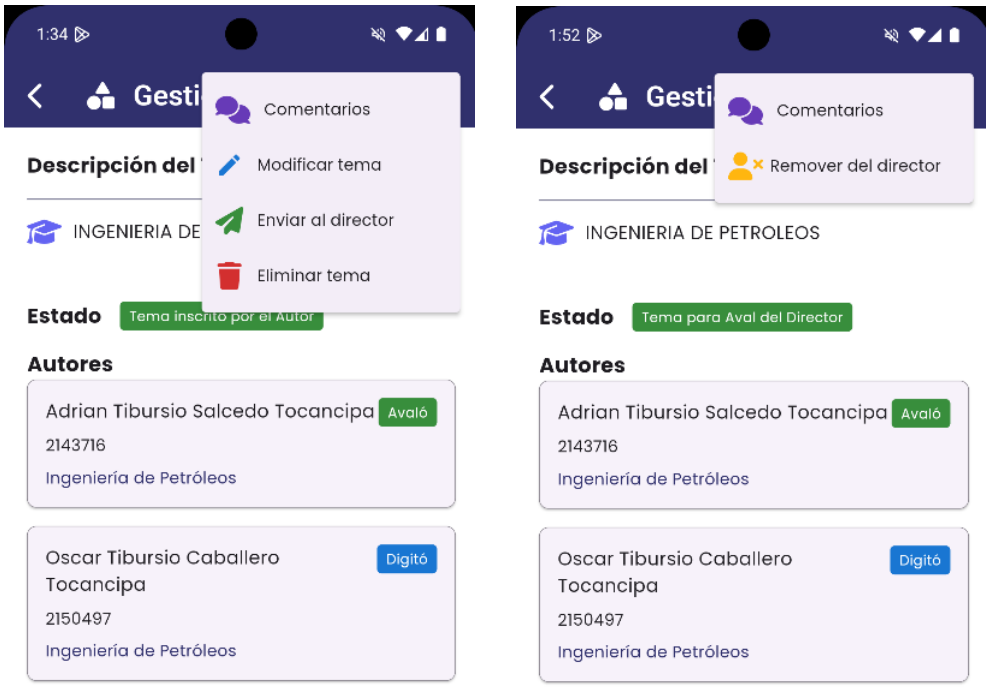
Mensaje de confirmación sobre la acción "Eliminar tema".



Cabe resaltar que cada acción realizada lanza un evento en donde se notifica a los coautores o director (acción *Enviar al director*) vía correo electrónico.

Figura 15

Acciones disponibles para la gestión del tema



5.2.2 Módulo Subir Plan

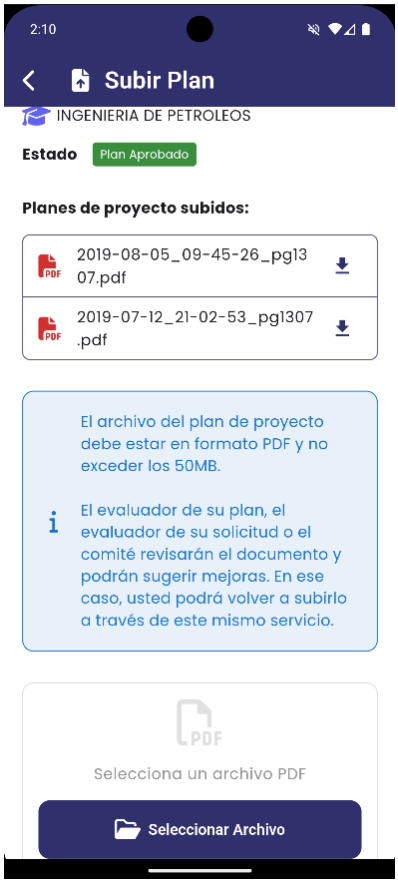
Este módulo constituye la segunda fase fundamental en el ciclo de vida del trabajo de grado, se puede realizar una vez que el tema del estudiante ha sido aprobado por el comité. El propósito de este módulo es proporcionar una interfaz centralizada y controlada para que el estudiante cargue, gestione y versione el documento de su plan de proyecto. También cuenta con la funcionalidad de comentarios para la comunicación entre el director y el autor.

5.2.2.1 Gestión de Archivos. La pantalla principal está diseñada para ser intuitiva y funcional para el usuario, proporcionando toda la información necesaria de manera clara.

Se incluye un panel informativo estático que comunica claramente las reglas y el proceso: se especifican las restricciones técnicas (formato PDF, tamaño máximo de 50MB) y se explica que el documento será revisado y que se podrán subir nuevas versiones basadas en comentarios recibidos por parte del director.

Figura 16

Interfaz Subir Plan



- **Selección y Subida del Plan:** La interfaz presenta un selector de archivos configurado para aceptar únicamente documentos en formato PDF, previniendo

errores de formato desde el origen. Una vez seleccionado, el servidor gestiona el proceso de subida, mostrando un indicador de carga que se activa y desactiva de forma reactiva, informando al usuario del progreso.

- **Listado Reactivo y Versionado:** La pantalla muestra una lista de todos los planes de proyecto que se han subido previamente. Esta lista es reactiva; gracias al gestor de estado GetX, se actualiza automáticamente en la interfaz de usuario inmediatamente después de una subida exitosa, sin necesidad de recargar la pantalla. Esto funciona como un versionado implícito.
- **Descarga de Documentos:** Cada elemento en la lista de planes subidos es descargable. La lógica de descarga incluye optimizaciones como la verificación de caché local para evitar descargar un archivo que ya existe en el dispositivo y la capacidad de abrir el documento automáticamente con la aplicación predeterminada una vez finalizada la descarga.

5.2.3 Módulo Subir Documento Final

Una vez que el plan de proyecto ha sido aprobado, el desarrollo del trabajo de grado entra en su fase final, la cual culmina con la entrega del documento definitivo. Para gestionar este proceso se implementa dos módulos secuenciales y complementarios: uno para la revisión del director y otro para la evaluación de los calificadores.

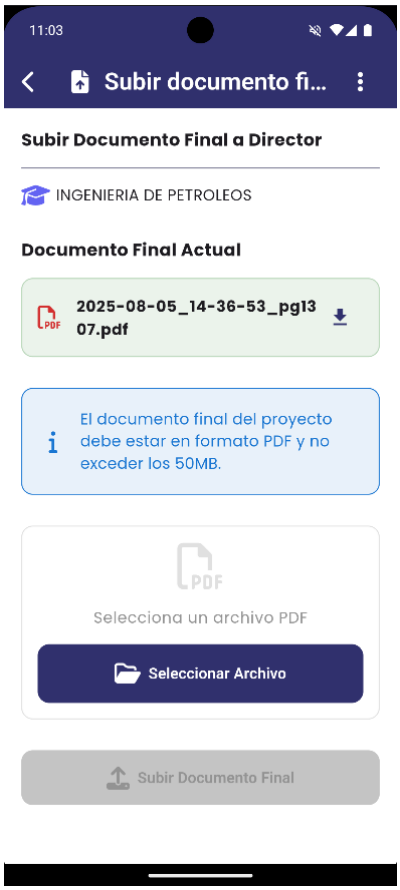
5.2.3.1 Subida del Documento Final para Revisión del Director. Este módulo tiene como propósito principal facilitar la entrega del documento final del trabajo de grado por parte del estudiante a su director para una revisión exhaustiva. Esta etapa es un control de calidad intermedio y obligatorio antes de que el documento sea presentado al comité calificador.

Las funcionalidades claves de este módulo son:

- **Gestión del Documento**: Permite al estudiante subir un único archivo en formato PDF (con un tamaño máximo de 50MB). El sistema gestiona el versionado de manera implícita, de modo que cada nueva subida reemplaza a la anterior, asegurando que el director siempre revise la versión más reciente.
- **Canal de Comunicación**: Se habilita un foro de comunicación bidireccional exclusivamente entre el autor (o autores) y el director. Este espacio está diseñado para discutir correcciones, solicitar aclaraciones y realizar comentarios sobre el documento.
- **Validaciones y Seguridad**: El sistema verifica rigurosamente que el usuario que intenta subir un archivo o comentar sea efectivamente el autor o el director asignado a ese proyecto, garantizando la integridad del proceso.
- **Notificaciones Automáticas**: Tanto el estudiante como el director reciben notificaciones por correo electrónico cuando se sube una nueva versión del documento o se realiza un comentario, manteniendo a ambos informados y agilizando el ciclo de revisión.

Figura 17

Interfaz para subir documento final al director.



5.2.3.2 Subida del Documento Final para Evaluación de Calificadores. Una vez que el director ha revisado y otorgado su aprobación al documento final, el proyecto avanza a la etapa de evaluación formal. Este segundo módulo se activa para gestionar la entrega del documento final al comité calificador asignado. Su propósito es formalizar la entrega para la evaluación que precederá a la sustentación.

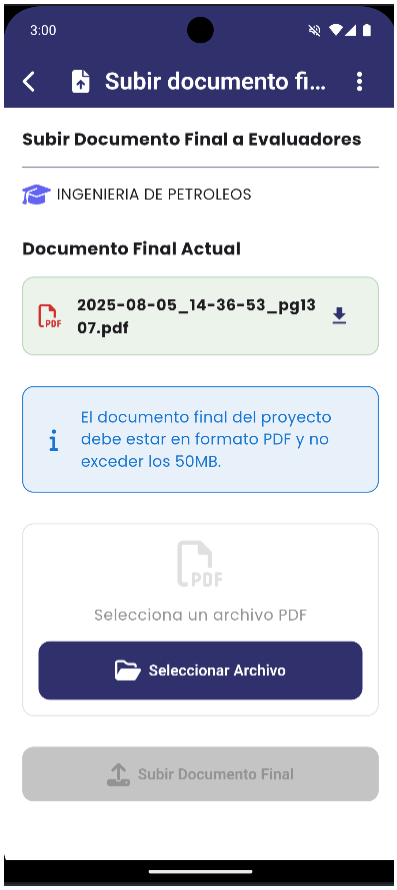
Este módulo comparte una base funcional con el anterior, pero con diferencias clave en su propósito y participantes:

- **Gestión del Documento para Evaluación**: Al igual que en la fase anterior, el estudiante sube la versión final y aprobada por el director en formato PDF. Este documento será la base sobre la cual los calificadores realizarán su evaluación.

- **Canal de Comunicación con el Comité**: Se establece un foro de comunicación entre el autor (o autores) y todos los miembros del comité calificador. Este espacio permite a los evaluadores realizar observaciones finales o solicitar aclaraciones antes de la sustentación.
- **Validaciones de Rol**: El sistema valida que solo los autores y los calificadores asignados a ese trabajo de grado puedan interactuar en este módulo.
- **Notificaciones al Comité**: El sistema notifica automáticamente a todos los miembros del comité calificador cuando el documento final ha sido cargado, asegurando que todos los evaluadores tengan acceso oportuno al material para su revisión.

Figura 18

Interfaz para subir documento final a evaluadores.



5.2.4 Módulo de Comentarios

El Módulo de Comentarios es un componente transversal y fundamental de la aplicación, diseñado para servir como un canal de comunicación contextualizado entre los estudiantes, directores y evaluadores a lo largo de las distintas fases del trabajo de grado. Su diseño se basa en una arquitectura centralizada y reutilizable, lo que garantiza una experiencia de usuario consistente y un mantenimiento de código eficiente.

La principal característica de este módulo es su reutilización. En lugar de construir sistemas de chat separados para cada fase, se implementó un conjunto único de componentes que se adaptan según el contexto:

- **Pantalla Genérica (CommentsScreen)**: Una única pantalla que se configura dinámicamente al ser invocada, recibiendo parámetros como el identificador de la tesis (thesisId) y el tipo de conversación (commentsType).
- **Controlador Único (CommentsController)**: Un solo controlador que gestiona toda la lógica de negocio, incluyendo la obtención (fetchComments) y el envío (addComment) de mensajes, comunicándose con los endpoints correspondientes de la API.
- **Widget de Interfaz Reutilizable (CommentBox)**: Un componente de interfaz de usuario que renderiza la lista de mensajes y proporciona el campo de texto para el envío. Este widget maneja estados de carga y deshabilita la entrada de datos durante las operaciones de red para prevenir envíos duplicados.

Esta estrategia de reutilización reduce la duplicación de código y asegura que cualquier mejora o corrección aplicada al módulo se propague a todos los flujos de trabajo donde se utiliza.

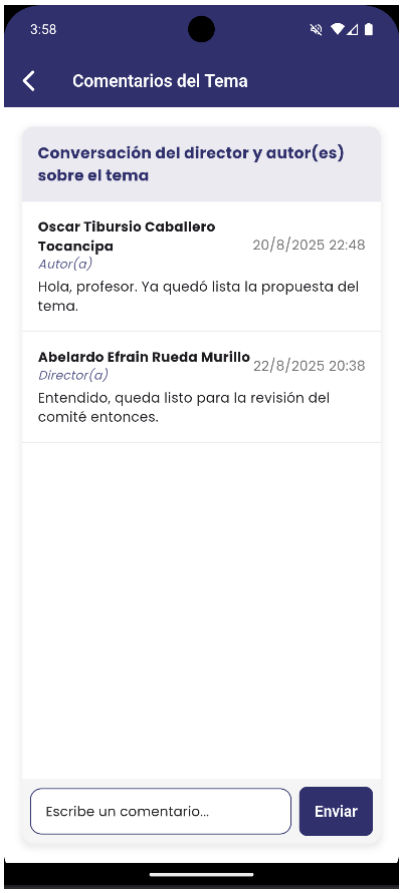
5.2.4.1 Integración Contextual en el Flujo del Trabajo de Grado. El módulo se integra en cuatro etapas clave del proceso académico, cada una con un hilo de conversación independiente y un propósito específico:

- **Discusión del Tema del Proyecto (commentsType: 'topic')**: Durante la fase de propuesta, este canal permite una comunicación directa entre el estudiante y el director para discutir, refinar y aprobar el tema del trabajo de grado.

- **Revisión del Plan de Proyecto (commentsType: 'plan')**: Una vez aprobado el tema, se habilita un nuevo hilo de conversación dedicado exclusivamente al feedback sobre el documento del plan de proyecto, manteniendo las discusiones organizadas.
- **Revisión del Documento Final por el Director (commentsType: 'final')**: Facilita un canal de comunicación privado entre el estudiante y su director para los ajustes finales del documento antes de ser enviado a evaluación.
- **Evaluación del Documento Final por Calificadores (commentsType: 'final')**: Permite la interacción entre el estudiante y el comité calificador para observaciones y consultas previas a la sustentación.

Figura 19

Ejemplo de una conversación entre autor y director en la gestión del tema.



5.2.5 Módulo de Notificaciones Snackbar

Para proporcionar retroalimentación al usuario de manera efectiva, se implementó un sistema de notificaciones basado en el componente de interfaz de usuario conocido como Snackbar. Una Snackbar es un mensaje breve que aparece temporalmente en la pantalla, generalmente en la parte superior, para informar al usuario sobre el resultado de una operación. A diferencia de un diálogo modal, no interrumpe el flujo de trabajo del usuario y desaparece automáticamente al cabo de unos segundos, lo que lo convierte en el mecanismo ideal para confirmaciones, advertencias o errores puntuales.

5.2.5.1 Implementación de un Componente Centralizado. Con el fin de garantizar la consistencia visual y funcional en toda la aplicación, se desarrolló un componente centralizado y reutilizable denominado MySnackbar. En lugar de invocar notificaciones directamente desde cada pantalla, este componente actúa como una única fuente de verdad, lo que simplifica su mantenimiento y permite aplicar estilos de manera global.

La implementación de MySnackbar se construyó sobre la funcionalidad Get.snackbar de la biblioteca de gestión de estado **GetX**. Esta elección técnica permite mostrar notificaciones de forma superpuesta en la interfaz, lo que otorga una gran flexibilidad para invocar notificaciones desde cualquier capa de la lógica de la aplicación, incluidos los controladores o los servicios de red.

5.2.5.2 Variantes Semánticas para una Comunicación Intuitiva. El componente MySnackbar fue diseñado con cuatro variantes semánticas preconfiguradas que comunican el propósito del mensaje de manera inmediata a través de un código de colores e iconografía estándar:

- **Éxito (Success)**: Utiliza un fondo verde y un icono de verificación para confirmar que una operación se ha completado correctamente, como "El archivo se subió con éxito".
- **Error (Danger)**: Emplea un fondo rojo y un icono de advertencia para informar sobre fallos o acciones que no se pudieron completar, como "No se pudo conectar con el servidor".
- **Información (Info)**: Con un fondo azul, comunica mensajes de estado neutral o guías contextuales.

- **Advertencia (Warning)**: Usa un fondo amarillo para llamar la atención sobre información importante que no constituye un error crítico.

Esta clasificación semántica mejora significativamente la experiencia de usuario, ya que el significado del mensaje se transmite visualmente antes incluso de leer el texto.

5.2.5.3 Características Avanzadas y Experiencia de Usuario. El componente MySnackbar fue desarrollado con características adicionales para asegurar una alta calidad en la experiencia de usuario y el cumplimiento de buenas prácticas de diseño:

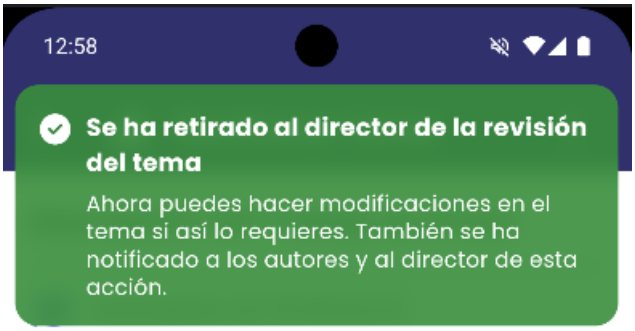
- **Cálculo de Contraste Automático**: Para garantizar la legibilidad y la accesibilidad, el componente incluye una función (_getOptimalTextColor) que calcula automáticamente si el texto del mensaje debe ser blanco o negro en función del color de fondo de la notificación. Esto asegura un contraste adecuado en todos los temas y variantes.
- **Duración Configurable y Notificaciones Persistentes**: Por defecto, las notificaciones desaparecen automáticamente después de unos segundos. Sin embargo, el componente permite configurar una duración personalizada o, si se incluye un botón de "Cerrar", la notificación se vuelve persistente, permaneciendo en pantalla hasta que el usuario interactúe directamente con ella, lo cual es útil para mensajes de error importantes.

5.2.5.4 Integración con la Lógica de la Aplicación. Este sistema de notificaciones está integrado en la arquitectura de la aplicación. Su uso más notable es en conjunto con el **ResponseInterceptor** de la capa de comunicación HTTP. Cuando el servidor responde a una

petición de la API con un "flash message" (un mensaje de estado incluido en la respuesta JSON), el interceptor lo procesa automáticamente e invoca la variante correspondiente de MySnackbar para informar al usuario del resultado de la operación. Este diseño crea un bucle de retroalimentación automatizado y consistente, mejorando la robustez y la usabilidad del sistema.

Figura 20

Notificación snackbar de éxito al retirar al director de la revisión del tema.



5.2.6 Módulo de Autenticación

El módulo de autenticación es el portal de entrada a la aplicación y el responsable de verificar la identidad del usuario, establecer una sesión segura y personalizar la experiencia de la aplicación según su rol y su afiliación académica. Su diseño se basa en una arquitectura robusta que combina una interfaz de usuario clara, una gestión de estado centralizada con **GetX** y una capa de comunicación segura que maneja la sesión a través de cookies.

5.2.6.1 Interfaz de Usuario y Flujo de Inicio de Sesión. La experiencia de autenticación comienza en la pantalla de inicio de sesión (LoginScreen), la cual presenta al usuario un formulario con tres campos esenciales:

- Selector de Escuela, que permite al usuario elegir su escuela académica de una lista predefinida.
- Nombre de Usuario
- Contraseña, con una funcionalidad para alternar su visibilidad que mejora la usabilidad.

Al presionar el botón "Iniciar Sesión", se activa un flujo orquestado que proporciona retroalimentación constante al usuario. Primero, se muestra un diálogo de carga no bloqueante para indicar que el proceso está en curso. A continuación, se envían las credenciales al servidor a través del cliente HTTP. Si la autenticación es exitosa, el sistema guarda la información del usuario, establece el estado de la sesión y navega al usuario a la pantalla principal de la aplicación.

En caso de que el servidor devuelva errores de validación (por ejemplo, contraseña incorrecta o usuario inexistente), el sistema está diseñado para proporcionar retroalimentación contextual. La respuesta de error del servidor es capturada y utilizada para mostrar mensajes específicos directamente debajo del campo correspondiente en el formulario, guiando al usuario para que corrija la información de manera precisa.

Figura 21

Pantalla inicio de sesión.

5.2.6.2 Gestión de Estado y Sesión con GetX. La gestión del estado de la sesión y la información del usuario se centraliza en dos controladores principales de GetX, garantizando una única fuente de verdad en toda la aplicación:

- **AuthController**: Su responsabilidad principal es almacenar en memoria reactiva el objeto User una vez que la autenticación ha sido exitosa. Esto permite que otras partes de la aplicación accedan a la información del usuario (nombre, correo, etc.) de manera consistente.
- **SchoolController**: Este controlador gestiona la escuela seleccionada por el usuario y el estado booleano isLoggedIn. Una de sus funciones más importantes es la persistencia: guarda el identificador de la escuela seleccionada en

SharedPreferences, permitiendo que la aplicación recuerde esta selección entre sesiones. Además, el tema visual de la aplicación se adapta dinámicamente a la paleta de colores definida para la escuela seleccionada, personalizando la experiencia visual del usuario.

5.2.6.3 Proceso de Cierre de Sesión. El cierre de sesión es un proceso controlado que revierte el estado de la aplicación a su punto inicial. Cuando el usuario selecciona la opción "Cerrar Sesión", la aplicación realiza los siguientes pasos:

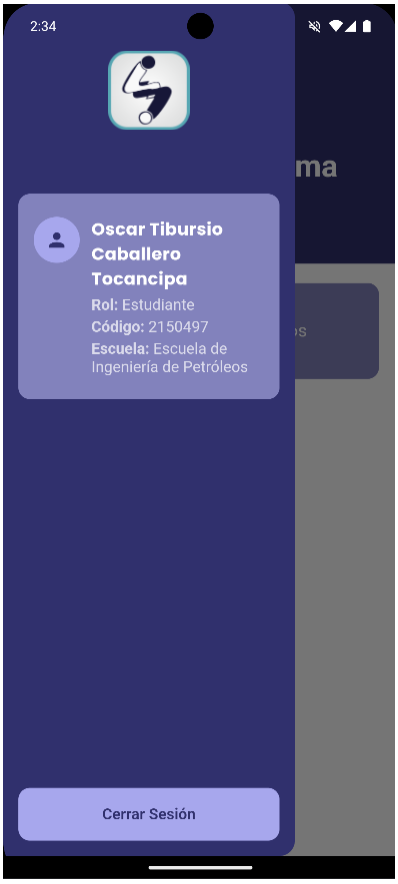
- Envía una solicitud al endpoint *api/auth/logout* para invalidar la sesión en el servidor.
- Limpia la información del usuario del AuthController.
- Reinicia el estado de sesión y la escuela seleccionada en el SchoolController, eliminando también la persistencia en SharedPreferences.
- Navega al usuario de vuelta a la LoginScreen, completando el ciclo.

5.2.7 Pantallas Comunes y Flujo de Navegación Principal

5.2.7.1 Menú de Navegación Lateral. El menú de navegación lateral (Drawer), es un persistente accesible desde la pantalla del inicio. Su propósito es servir como un centro de identidad y gestión de la sesión para el usuario. Visualmente, está diseñado con los colores institucionales y el logo de la aplicación. Desde aquí el usuario puede cerrar su sesión.

Figura 22

Menú lateral.



5.2.7.2 Pantalla de Inicio. Tras una autenticación exitosa, la pantalla de inicio (HomeScreen) es la primera interfaz que recibe al usuario. Funciona como un portal de bienvenida y un punto de partida dinámico hacia las funcionalidades más relevantes.

- **Bienvenida Personalizada**: La parte superior de la pantalla (AppBar) está diseñada para ocupar un espacio visual prominente, mostrando un saludo personalizado con el nombre del usuario y el título de la aplicación.
- **Navegación Basada en Roles**: El cuerpo de la pantalla es dinámico y su contenido se adapta al rol del usuario autenticado. Para el usuario con rol de estudiante, se presenta un único botón de acción principal: *Gestionar mis proyectos*.

Figura 23

Pantalla de inicio.

5.3 Preparación para el Despliegue en Producción

Como parte del desarrollo del proyecto, se preparó y documentó la viabilidad técnica para un futuro despliegue de la aplicación en el entorno de producción de la universidad. Aunque la puesta en marcha final constituye una fase posterior (ver Capítulo 7: Trabajo Futuro), en esta etapa se establecieron las bases de infraestructura necesarias para garantizar una transición fluida y segura.

5.3.1 Viabilidad Técnica del Servidor (Back-end)

Un resultado clave del proyecto fue la preparación del servidor para su eventual despliegue. Para ello, se abordó la containerización de la aplicación (back-end) utilizando Docker.

El proyecto incluye un Dockerfile optimizado que facilita la creación de una imagen ligera y segura de la aplicación. Este contenedor está diseñado para ser integrado dentro de la red de contenedores interna de la plataforma COMA, una ubicación estratégica que alberga los contenedores de las bases de datos de cada escuela académica.

La preparación de esta configuración es crucial, ya que asegura que, en un futuro despliegue, el servidor pueda establecer una comunicación directa, segura y de baja latencia con las bases de datos, lo cual es un pilar para el correcto funcionamiento de la arquitectura multi-tenant implementada.

6. Conclusiones

El presente trabajo de grado tuvo como objetivo principal llevar a un aplicativo móvil la funcionalidad de gestión de trabajos de grado de la plataforma COMA. Para abordar este reto, se realizó un análisis detallado de las tecnologías disponibles, decantándose por una arquitectura cliente-servidor robusta, compuesta por un cliente móvil desarrollado en **Flutter** y un servidor construido con **Node.js y Express**. La arquitectura implementada demostró ser eficaz, con una clara separación entre la interfaz de usuario (front-end) y la lógica de negocio y gestión de datos (back-end), cumpliendo así con los objetivos de desarrollo propuestos.

La contribución más significativa de este proyecto es la solución directa al problema de accesibilidad de la plataforma COMA. Al desarrollar esta aplicación móvil, se rompe la dependencia de un ordenador de escritorio, otorgando a los estudiantes la flexibilidad para gestionar su proceso académico más importante en cualquier momento y lugar. Sin embargo, la contribución más relevante desde una perspectiva técnica y de sistema fue garantizar la interoperabilidad y coexistencia con la plataforma web. Un pilar fundamental del diseño fue asegurar que la aplicación móvil funcionara como un complemento y no como un reemplazo, garantizando que cualquier acción registrada desde el móvil se refleje de manera consistente en la web, y viceversa. Esto asegura una experiencia de usuario unificada y sin inconsistencias de datos.

Entre otras contribuciones se encuentran:

- **Modernización de Servicios Académicos**: El proyecto representa un paso adelante en la modernización de las herramientas tecnológicas de la Universidad Industrial de Santander.

- **Mejora de la Experiencia del Estudiante**: La aplicación agiliza los procesos académicos al ofrecer retroalimentación en tiempo real a través de sus módulos de comunicación.
- **Implementación de una Arquitectura Escalable**: El diseño del servidor, especialmente su capacidad **multi-tenant**, proporciona una base técnica sólida que puede ser extendida en el futuro.

Finalmente, es importante destacar que en este trabajo se logró desarrollar un aplicativo móvil funcional, con módulos probados y una arquitectura multi-tenant preparada para su integración con la plataforma COMA. No obstante, aunque se documentó y preparó la viabilidad técnica del despliegue en un entorno de producción mediante containerización, la puesta en marcha institucional excede el alcance definido en los objetivos del proyecto. Este proceso involucra factores externos como la gestión de infraestructura, la aplicación de políticas de seguridad y el soporte institucional, por lo cual no fue considerado como un objetivo obligatorio del presente trabajo, dejando a la institución un producto listo para ser implementado.

7. Trabajo a Futuro

El desarrollo de esta aplicación móvil ha sentado las bases para un ecosistema de gestión académica más robusto y accesible en la Universidad Industrial de Santander. Si bien el proyecto actual cumplió con sus objetivos centrados en el rol del estudiante, su arquitectura modular y escalable abre la puerta a numerosas oportunidades de expansión y mejora. A continuación, se detallan las líneas de trabajo futuro recomendadas para dar continuidad y potenciar el impacto de esta iniciativa.

7.1 Puesta en Producción Institucional

Como trabajo futuro, se recomienda la puesta en producción del aplicativo móvil dentro de la infraestructura tecnológica de la Universidad Industrial de Santander. Si bien en este proyecto se dejó preparada la viabilidad técnica mediante la containerización del servidor y la arquitectura multi-tenant, el despliegue real requiere la coordinación con los equipos de infraestructura institucional, la adopción de políticas de seguridad y privacidad, así como la definición de mecanismos de soporte a largo plazo. Estas tareas, al estar fuera del alcance académico definido en los objetivos, constituyen una etapa posterior que garantizaría la integración adecuada del sistema en la plataforma COMA y su sostenibilidad en el tiempo.

7.2 Ampliación de Funcionalidades para el Rol Docente

La recomendación principal es la ampliación de la aplicación para incluir las funcionalidades del rol docente. Actualmente, la aplicación se centra exclusivamente en la perspectiva del estudiante. El siguiente paso es desarrollar los módulos necesarios para que los directores puedan gestionar sus responsabilidades directamente desde sus dispositivos móviles.

7.3 Mejoras Técnicas y de Experiencia de Usuario

Para refinar la versión actual de la aplicación y mejorar su robustez, se proponen las siguientes mejoras:

- **Añadir Notificaciones Push:** Integrar un servicio de notificaciones push para alertar a los usuarios en tiempo real sobre eventos importantes, como la recepción de un nuevo comentario, un cambio de estado en el proyecto o la proximidad de una fecha límite.

- **Implementar Persistencia de Sesión Segura:** Añadir una funcionalidad de "Recordarme" para que los usuarios no tengan que iniciar sesión cada vez que abren la aplicación. Esto se puede lograr mediante el almacenamiento seguro del token de sesión o la implementación de un sistema de *refresh tokens*.

7.4 Migración de Otros Servicios de COMA

Además de la gestión de trabajos de grado, la plataforma web de COMA ofrece una variedad de servicios académicos que son viables para migrar a la aplicación móvil. Se recomienda realizar un análisis para identificar y priorizar otras funcionalidades que beneficiarían a la comunidad universitaria.

Referencias Bibliográficas

Atlassian. (s.f.). *Scrum*. https://www.atlassian.com/es/agile/scrum

Zod

Colinhacks. (s.f.). *Zod: TypeScript-first schema validation with static type inference*. https://zod.dev/

Cormoran. (s.f.). CALUMET, Grupo de Innovación y Desarrollo. http://ingsistemas.uis.edu.co/eisi/grupo/calumet/#views/gm1/inicio

Delía, Lisandro Nahuel. (2017). *Desarrollo de aplicaciones móviles multiplataforma.* UNIVERSIDAD NACIONAL DE LA PLATA. http://sedici.unlp.edu.ar/handle/10915/60497

Express.js

Express. (s.f.). *Express: Fast, unopinionated, minimalist web framework for Node.js*. https://expressjs.com/

Google. (2023). *Todo lo que necesitas para realizar tareas, ahora en un solo lugar.* Google Workspace. https://workspace.google.com/intl/es

Dart

Google. (s.f.). *Dart: A client-optimized language for fast apps on any platform*. https://dart.dev/

Flutter

Google. (s.f.). *Flutter: Build apps for any screen*. https://flutter.dev/documentation

IBM. (2023). *¿Qué es el desarrollo de aplicaciones móviles?* https://www.ibm.com/es-es/topics/mobile-application-development

JSON Web Tokens (JWT)

Internet Engineering Task Force (IETF). (2015). *RFC 7519: JSON Web Token (JWT)*. https://www.jwt.io/introduction#what-is-json-web-token

ISO 25000. (s.f.). *Usabilidad*. https://iso25000.com/index.php/normas-iso-25000/iso-25010/23-usabilidad

Laudon, K. C., & Laudon, J. P. (2016). *Sistemas de información gerencial* (14.ª ed.). Pearson.

GetX

Law, J. (s.f.). *GetX: The solution for state management, dependency injection, and route management*. Pub.dev. https://pub.dev/packages/get

Microsoft. (2023). *¿Qué es el desarrollo de aplicaciones móviles?* https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-mobile-app-development/#definition

Ministerio de Tecnologías de la Información y las Comunicaciones. (s.f.). *Política General de Gobierno en línea 2013-2018*. https://www.mintic.gov.co/gestionti/615/articles-5482_G2_Politica_General.pdf

Piattini Velthuis, M., et al. (2007). *Análisis y diseño detallado de aplicaciones informáticas de gestión*. RA-MA.

Ruiz Rivera, M. E., Torres Dávila, G., & Ruiz Lizama, E. (2021). *Diseño y desarrollo de un aplicativo móvil educativo para optimizar la comunicación e interacción entre los miembros de las instituciones educativas en tiempo real*. Centro Institucional de Publicaciones y revistas especializadas - Fundación universitaria San mateo. https://www.redalyc.org/journal/816/81668400013/html/

Sequelize (ORM

Sequelize. (s.f.). *Sequelize: Promise-based Node.js ORM for Postgres, MySQL, MariaDB, SQLite and SQL Server*. https://sequelize.org/

Apéndices

Apéndices A. Guía de Instalación y Despliegue Local

En este apéndice se proporcionan las instrucciones técnicas necesarias para configurar y ejecutar el aplicativo móvil en un entorno de desarrollo local. Se detallan la ubicación de los repositorios de código fuente, los prerrequisitos de software y los pasos para la instalación tanto del servidor (back-end) como del cliente (front-end).

A.1 Repositorios de Código Fuente

El código fuente del proyecto está dividido en dos repositorios independientes, alojados en GitHub:

- **Servidor (Back-end)**: El código del servidor desarrollado en Node.js se encuentra disponible en el siguiente enlace: https://github.com/calumet/COMA_movil_server
- **Cliente (Front-end):** El código de la aplicación móvil desarrollada en Flutter se encuentra disponible en el siguiente enlace: https://github.com/calumet/coma-movil

Para tener acceso a estos repositorios debes tener acceso al grupo de desarrollo Calumet en GitHub.

A.2 Prerrequisitos de Software

Antes de proceder con la instalación, es necesario tener instalado el siguiente software en el equipo local:

- **Para el Servidor**:
    - Flutter SDK (versión 3.x o superior).

- Android Studio para el SDK de Android y la gestión de emuladores.
- Un editor de código o IDE (ej. Visual Studio Code, Android Studio).
- Un emulador de Android o un dispositivo físico para pruebas.

A.3 Configuración del Entorno del Servidor (Back-end)

Para ejecutar el servidor en un entorno local, siga los siguientes pasos:

- **Clonar el repositorio** del servidor desde la URL proporcionada.
- **Instalar las dependencias** navegando al directorio raíz del proyecto y ejecutando el comando: npm install.
- **Configurar las variables de entorno**. Cree un archivo .env en la raíz del proyecto. Este archivo debe contener las claves de configuración necesarias, como el puerto (PORT), las credenciales de la base de datos por defecto (DB_HOST, DB_NAME, DB_USER, DB_PASSWORD) y el secreto para la firma de tokens (JWT_SECRET_KEY).
- **Configurar la arquitectura multi-tenant**. El archivo src/db-config.js contiene las configuraciones de conexión para cada escuela académica. Para un entorno local, es necesario editar este archivo y modificar los host de cada escuela para que apunten a la dirección del servidor MySQL local (generalmente localhost).
- **Ejecutar el servidor**. Inicie el servidor en modo de desarrollo con el comando: npm run dev o node .\src\app.js

A.4 Configuración del Entorno del Cliente (Front-end)

Para ejecutar la aplicación móvil, siga estos pasos:

- **Clonar el repositorio** del cliente desde la URL proporcionada.

- **Instalar las dependencias** del proyecto ejecutando el comando flutter pub get en el directorio raíz.
- **Configurar la URL del servidor (Paso crítico)**. Abra el archivo lib/utils/helpers/http/http_client.dart y modifique la constante _baseUrl para que apunte a la dirección IP del servidor back-end que se configuró en el paso anterior.
    - Si se utiliza un **emulador de Android**, la dirección debe ser: http://10.0.2.2:<PUERTO>.
    - Si se utiliza un **dispositivo físico** en la misma red, la dirección debe ser la IP local de la máquina que ejecuta el servidor (ej. http://192.168.1.100:<PUERTO>).
- **Ejecutar la aplicación**. Con un emulador en ejecución o un dispositivo conectado, utilice el comando flutter run para compilar e instalar la aplicación.

A.5 Generación del Paquete de Instalación (APK)

Para generar un paquete de instalación de Android (.apk) para pruebas de desarrollo, se debe ejecutar el siguiente comando en la raíz del proyecto del cliente:

flutter build apk –debug

El archivo .apk resultante se encontrará en el directorio build/app/outputs/flutter-apk/. Este archivo puede ser instalado directamente en un dispositivo Android para realizar pruebas funcionales.