UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación multiplataforma para la gestión de alumnos en academias

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Albero Belamendía, José María

Tutor/a: Izquierdo Doménech, Juan Jesús

CURSO ACADÉMICO: 2022/2023

Resumen

Este Trabajo Fin de Grado consiste en la realización de una aplicación multiplataforma para la gestión del alumnado matriculado en una academia, abarcando la parte de administrador (i.e., profesorado) y la parte de clientes (i.e., alumnado). La aplicación facilitará al profesorado la gestión de los alumnos, llevando un calendario de las clases programadas, así como una lista de los alumnos que tiene matriculados en sus clases con información sobre las tareas pendientes o trabajos a realizar. Los alumnos podrán subir archivos y escribir las tareas que tienen que hacer, además de tener un chat para poder hablar con el profesorado. Este proyecto no será desarrollado para una sola academia; pretende ser una aplicación genérica que se pueda distribuir a cualquier academia de refuerzo o aprendizaje. La aplicación será multiplataforma, pudiendo ser soportada en varios medios: web, android e iOS, abarcando así al mayor número de personas que pudiesen utilizar la aplicación. Uno de los objetivos principales de este proyecto será desarrollar una interfaz usable, eficiente y amigable, pudiendo ser usada por cualquier persona que tenga conocimientos básicos de las tecnologías. Las tecnologías seleccionadas para desarrollar este proyecto han sido el framework Flutter y el lenguaje de programación Dart. Flutter es un framework de desarrollo de aplicaciones móviles altamente eficiente y versátil, que permite la creación de interfaces de usuario atractivas y la implementación de funcionalidades complejas en múltiples plataformas y que, junto con el lenguaje de programación Dart, posibilita una integración perfecta entre la capa de presentación y la capa de lógica de negocio de la aplicación.

Palabras clave: aplicación multiplataforma; academia; Flutter; Dart; interfaz usable.

Abstract

This Bachelor's Thesis consists in the development of a cross-platform managing application for students enrolled in an academy, covering both the administrative (i.e., teaching staff) and customer (i.e., students) parts. The application will ease the management of students for the teaching staff, including a calendar of scheduled classes and a list of students enrolled in their classes with information on pending tasks or assignments to be completed. Students will be able to upload files and write their assignments, in addition to having a chat to communicate with the teaching staff. This project will not be developed for a single academy; it aims to be a generic application that can be distributed to any reinforcement or learning academy. The application will be cross-platform, supporting various mediums: web, Android, and iOS, thus covering the largest possible number of people who could use the application. One of the main objectives of this project will be to develop a usable, efficient, and user-friendly interface that can be used by anyone with basic knowledge of technology. The technologies selected to develop this project are the Flutter framework and the Dart programming language. Flutter is a highly efficient and versatile mobile application development framework that allows for the creation of attractive user interfaces and the implementation of complex functionality on multiple platforms. Together with the Dart programming language, they enable a perfect working integration between the presentation layer and the business logic layer of the application.

Keywords: cross-platform application, academy, Flutter, Dart, usable interface.

A mi familia, especialmente a mi madre por acompañarme en esta travesía.

A mi abuelo por ser mi fuente de inspiración a diario, en cada página está tu recuerdo.

Índice

Índice de tablas

Índice de ilustraciones

1. Introducción

La educación es un aspecto fundamental en el desarrollo de las personas y juega un papel crucial en su crecimiento intelectual y social. En la actualidad, muchos padres buscan dar a sus hijos una educación complementaria que les permita reforzar y ampliar sus conocimientos fuera del entorno escolar. Las academias de refuerzo se han convertido en una opción para aquellos padres y estudiantes que desean aprovechar al máximo su potencial académico.

Las academias de refuerzo ofrecen clases adicionales y personalizadas en diversas materias, con el objetivo de ayudar a los estudiantes a mejorar su rendimiento académico y fortalecer sus habilidades en áreas específicas. Estas academias se han vuelto especialmente relevantes después de la pandemia que sin duda tuvo un impacto en los estudiantes sobre todo en los más jóvenes produciendo un retraso en el aprendizaje y proceso educativo.

Sin embargo, la gestión de una academia de refuerzo puede presentar desafíos significativos. La coordinación de horarios, la asignación de profesores, el seguimiento individualizado de cada estudiante y la comunicación efectiva con los padres son sólo algunas de las tareas que requieren un enfoque organizado y eficiente. En este contexto, la tecnología puede desempeñar un papel fundamental para simplificar y mejorar estos procesos.

(posible párrafo hablando sobre la organización)

Este proyecto se ha realizado con la intención de ayudar a las academias a poder mantener una organización del alumnado además de brindar una atención más personalizada a cada alumno, haciendo que su proceso de aprendizaje sea más efectivo y satisfactorio.

Al ser una aplicación donde toda la información está digitalizada, los alumnos y profesores podrán tener una línea de comunicación organizada donde cada alumno podrá saber qué clases tiene próximamente, qué tareas debe completar o qué avisos importantes tiene que comunicar el profesor.

1.1 Motivación personal:

Este proyecto surge con la idea de profundizar en el desarrollo de aplicaciones.

Siempre he sentido una profunda pasión por la educación y el aprendizaje y he creído en el poder transformador de la educación para abrir puertas y generar oportunidades. Durante mi trayectoria académica, he sido testigo de la importancia de las clases de apoyo a los estudiantes para que pudieran alcanzar su máximo potencial. Esto me llevó a explorar el campo de las academias de refuerzo y su impacto en el desarrollo académico de los estudiantes.

Todo esto se enmarca en un desafío personal que he asumido con determinación, con el propósito de lograr el desarrollo exitoso de una aplicación funcional, poniendo un enfoque prioritario en la usabilidad del usuario. Mi motivación surge de mi afán por adentrarme en el mundo del desarrollo *Full Stack* y adquirir competencias en una tecnología emergente y prometedora como lo es Flutter. Este proyecto representa una valiosa oportunidad para ampliar mis habilidades y conocimientos en el ámbito del desarrollo de aplicaciones móviles, superando obstáculos y buscando la excelencia en cada etapa del proceso de desarrollo.

1.2 Motivación profesional:

Desde una perspectiva profesional, se reconoció la necesidad de abordar los desafíos que enfrentan las academias de refuerzo en su gestión diaria. Con la creciente demanda de apoyo educativo personalizado, estas instituciones requieren herramientas eficientes y efectivas para organizar horarios, asignar profesores y mantener una comunicación fluida con los estudiantes y los padres. Esta motivación llevó a profundizar en el uso de Flutter como una solución tecnológica para optimizar estos procesos.

Además, el objetivo profesional es contribuir al campo de la educación mediante el desarrollo de soluciones innovadoras y prácticas. A través de este proyecto, se ha tenido la oportunidad de adquirir habilidades técnicas en el desarrollo de aplicaciones móviles y comprender la importancia de la usabilidad y la experiencia del usuario. Al crear una aplicación intuitiva y de alto rendimiento, se busca brindar a las academias de refuerzo una herramienta que les permita ofrecer un servicio de calidad y adaptado a las necesidades individuales de cada estudiante.

1.3 Objetivos

El objetivo principal de este Trabajo Fin de Grado (TFG) es el diseño y desarrollo de una aplicación móvil dirigida a los profesores y alumnos que conforman una academia de refuerzo. Dado el amplio rango de edades que una academia de refuerzo puede tener, se pretende conseguir una aplicación intuitiva y fácil de usar que pueda manejar la gestión de alumnos y profesores. A continuación, se presentan los objetivos secundarios derivados de este objetivo principal:

- Desarrollar un sistema de organización y asignación de horarios que facilite la planificación de clases.
- Implementar un sistema de asignación y seguimiento de actividades que permita recordar a los alumnos que actividades les han sido asignadas.
- Implementar un sistema de documentos que permita el acceso rápido y organizado a los documentos o archivos subidos por los profesores.
- Integrar un sistema de comunicación por avisos que permita a los profesores publicar anuncios que afecten a todos los estudiantes.
- Lograr baja latencia en las operaciones que se efectúen dentro de la aplicación, minimizando los tiempos de espera del usuario.
- El código debe ser desarrollado siguiendo estándares y buenas prácticas que faciliten la mantenibilidad, escalabilidad e incorporación de nuevas funcionalidades a la aplicación.
- Realizar pruebas exhaustivas de la aplicación para identificar y corregir posibles errores o mejoras adicionales antes de su implementación final.

1.4 Estructura del documento

La memoria se estructura en 9 capítulos más la bibliografía. A continuación, se comentará cada uno de los capítulos detallando que se puede encontrar en cada uno de ellos.

El capítulo 1 es la introducción al proyecto. Se proporciona una visión general del proyecto, presentando el contexto en el que se enmarca, las motivaciones que lo impulsan y los objetivos que se pretenden alcanzar.

En el capítulo 2, el Estado del Arte, se realiza un análisis exhaustivo de las soluciones existentes en el mercado relacionadas con el problema planteado. Se analizarán las funcionalidades cubiertas y no

cubiertas por estos sistemas y que puntos de mejoras tendrían para implementarlos en nuestro proyecto.

El capítulo 3 se centra en el estudio de las distintas tecnologías disponibles para el desarrollo de aplicaciones móviles, incluyendo las opciones nativas, web apps y aplicaciones híbridas. Se analizan sus características, ventajas y limitaciones, con el objetivo de seleccionar la tecnología más adecuada para la implementación de la solución propuesta.

El capítulo 4 denominado Tecnologías Empleadas en el Desarrollo de la Aplicación se centra en analizar y describir las tecnologías utilizadas durante el proceso de desarrollo. Se abordan tanto las herramientas como los lenguajes de programación, frameworks y librerías utilizadas para implementar la solución propuesta.

En este capítulo, se detallan las razones que llevaron a la elección de cada tecnología y se explican sus características clave. Se proporciona información sobre cómo estas tecnologías permiten cumplir con los requisitos del proyecto de manera eficiente y efectiva.

Además, se destacan las ventajas y beneficios que aportan estas tecnologías en términos de escalabilidad, rendimiento, seguridad y facilidad de mantenimiento. Asimismo, se aborda la compatibilidad de las tecnologías empleadas con los dispositivos móviles y los sistemas operativos utilizados por los usuarios finales.

El capítulo 5 será el Análisis del Problema, donde se lleva a cabo un análisis detallado de los requisitos, identificando las funcionalidades y características necesarias para resolver el problema planteado. Se elaboran casos de uso, un modelo de dominio y se establece un plan de trabajo que guiará el desarrollo del proyecto.

El Diseño de la Solución se aborda en el capítulo 6, donde se describe la arquitectura y la estructura general de la aplicación. Se definen los componentes principales y se establecen las relaciones entre ellos, garantizando la coherencia y la eficiencia del sistema.

En el capítulo 7, Desarrollo de la Solución Propuesta, se detalla el proceso de implementación de la aplicación. Se describen las decisiones de diseño y se explican todas las funcionalidades desarrolladas e implementadas en la aplicación.

Las pruebas son abordadas en un capítulo 8, donde se presentan los diferentes tipos de pruebas realizadas, incluyendo pruebas unitarias, de integración y de aceptación. Se discuten los resultados obtenidos y se analiza la calidad y el rendimiento de la solución implementada.

Finalmente, en el capítulo 9 se encuentran las Conclusiones donde se realiza una recapitulación de los resultados alcanzados, se evalúa el éxito del proyecto y el grado de cumplimiento de los objetivos. También se relacionará los estudios cursados con el trabajo desarrollado, se plantean posibles líneas de trabajo futuro y resalta las lecciones aprendidas durante todo el proceso de desarrollo.

2. Estado del arte

En el presente apartado, se llevará a cabo un análisis exhaustivo del estado del arte en el ámbito de las aplicaciones y empresas que ofrecen servicios similares a la aplicación desarrollada en el marco de este TFG. Es fundamental comprender el panorama actual y conocer las propuestas existentes para identificar las fortalezas y oportunidades de mejora de la aplicación desarrollada. Para ello se ha seleccionado dos empresas que se dedican a la gestión de academias, que disponen de plataformas que ofrecen servicios parecidos a los que nuestra aplicación realizará.

A través de una breve descripción de algunas de estas aplicaciones y empresas destacadas, se establecerá un contexto comparativo que permitirá destacar las características únicas y el valor añadido de la aplicación desarrollada. Además, se analizará funcionalidades que nuestra aplicación no integra dando pie a futuras mejoras y actualizaciones que encajen con la propuesta y filosofía de nuestra aplicación. Este análisis permitirá evaluar de manera crítica el estado del arte y justificar la relevancia e innovación de la aplicación desarrollada en el ámbito de la gestión de academias de refuerzo.

Por último, este capítulo incluye secciones que examinan y proporcionan una breve visión general de las tecnologías clave utilizadas en este proyecto. A través de estas secciones, se brinda al lector una comprensión clara de las decisiones tomadas al seleccionar las tecnologías consideradas más apropiadas para cada componente de la solución. Esto permitirá apreciar el razonamiento detrás de nuestras elecciones tecnológicas y cómo se han aplicado de manera efectiva en el desarrollo del proyecto.

2.1 Análisis de soluciones existentes

Una plataforma para la gestión de alumnos de academias es una aplicación versátil, ya sea en formato web, móvil o de escritorio, que tiene como objetivo principal digitalizar y mejorar los servicios ofrecidos por las academias. Con esta herramienta, los alumnos pueden experimentar una gestión integral y personalizada de su progreso académico, recibiendo un apoyo individualizado para alcanzar sus metas educativas de manera eficiente y efectiva.

Actualmente, no son ampliamente conocidas las aplicaciones que se enfoquen en las necesidades del usuario y se desvinculen de los aspectos puramente empresariales. La mayoría de las empresas que brindan clases de refuerzo deben hacer uso de múltiples herramientas para gestionar horarios, actividades y documentos, lo cual puede generar una carga de trabajo considerable. Sin embargo, la aplicación que se propone en este proyecto integra todas estas herramientas en una única plataforma, simplificando enormemente la tarea de organizar y gestionar a decenas de alumnos.

Aunque existen diversas plataformas en línea que permiten la organización de academias, estas suelen estar principalmente orientadas a las necesidades empresariales, dejando de lado a los propios alumnos y careciendo de un espacio dedicado especialmente para ellos. Para corroborar esta afirmación, se presentarán a continuación varios ejemplos de dichas plataformas disponibles en el mercado.

2.1.1 Atenea App

Atenea [1] es una aplicación desarrollada por la empresa Ender que se enfoca en ofrecer una solución integral para la gestión de academias de refuerzo educativo. La aplicación brinda a las academias y a sus alumnos una plataforma digitalizada y personalizada que facilita la organización, el seguimiento y la comunicación en el ámbito educativo.

La aplicación Atenea cuenta con características y funcionalidades como; poder gestionar los horarios de clases, asignar profesores a cada grupo de alumnos o llevar un control detallado de la asistencia y el rendimiento académico. Además, proporciona un espacio donde los alumnos pueden acceder a materiales de estudio, tareas y ejercicios, lo que les permite llevar un seguimiento constante de su progreso y desarrollar sus habilidades de forma individualizada.

Una de las características destacadas de Atenea es su sistema de comunicación integrado, que facilita la interacción fluida entre alumnos, profesores y padres. A través de la aplicación, se pueden enviar mensajes, compartir archivos y recibir notificaciones sobre novedades y eventos importantes. Esto promueve una comunicación directa y efectiva, permitiendo resolver dudas y mantener a todos los involucrados informados de manera oportuna.

Como puntos negativos se podría decir que la aplicación falla en aspectos esenciales como una buena interfaz amigable e intuitiva que resulte atractiva para los usuarios. La aplicación también carece de una división más específica de actividades y documentos, por lo que no se podría distinguir entre una actividad asignada a un alumno y documentos subidos para todos como una circular.

En la Ilustración 1 se puede ver la funcionalidad de calendario que incorpora la aplicación Atenea App y en la Ilustración 2 el módulo de navegación.

Ilustración 1 Atenea App Calendario

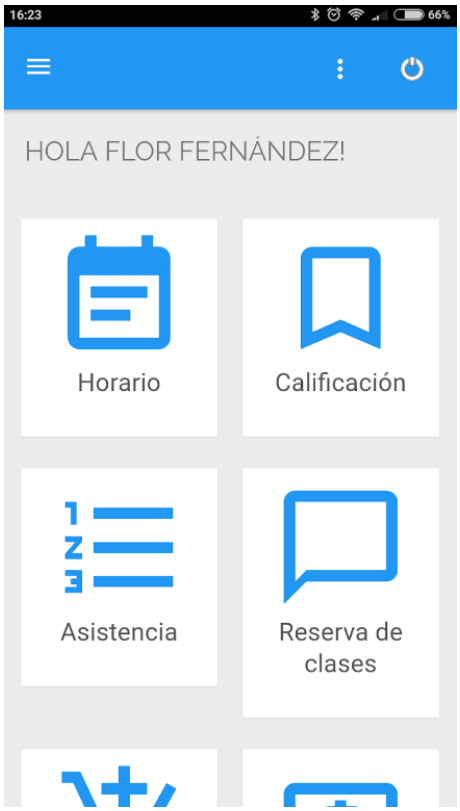

Ilustración 2 Atenea App Menú de navegación

2.1.2 Academity

La aplicación Academity [2], desarrollada por academity.es, es una plataforma educativa integral que se enfoca en la gestión eficiente de academias y centros educativos. Esta aplicación ofrece una amplia variedad de características y funcionalidades diseñadas para facilitar la organización, comunicación y seguimiento del progreso académico.

Una de las características destacadas de Academity es su sistema de gestión de horarios y asignación de profesores, que permite una planificación óptima de las clases y una distribución equilibrada de los recursos humanos. Además, la aplicación ofrece un portal de comunicación que conecta a estudiantes, padres y profesores, facilitando la interacción, la resolución de dudas y la notificación de eventos relevantes.

Otra funcionalidad destacada de Academity es su sistema de seguimiento académico, que permite a los profesores y administradores llevar un registro detallado del rendimiento de los estudiantes. Mediante esta herramienta, se pueden realizar evaluaciones, llevar un control de asistencia, y generar informes personalizados que brinden una visión completa del progreso de cada alumno.

Es cierto que la aplicación cuenta con funcionalidades interesante como la evaluación de alumnos, pero algunos puntos negativos a destacar serían:

- Demasiadas funcionalidades desvirtúan el propósito esencial de la aplicación.
- Muchas de sus funcionalidades están enfocada a una enseñanza online y no tanto presencial.
- Dispone de muchas herramientas que gestionan la parte empresarial del negocio, pudiendo no ser necesaria para negocios que efectúan este tipo de tareas con otros programas.

En la figura 3 se puede ver el módulo de avisos que incorpora la aplicación de Academity y en la figura 4 el perfil del usuario.

Ilustración 3 Academity vista de avisos

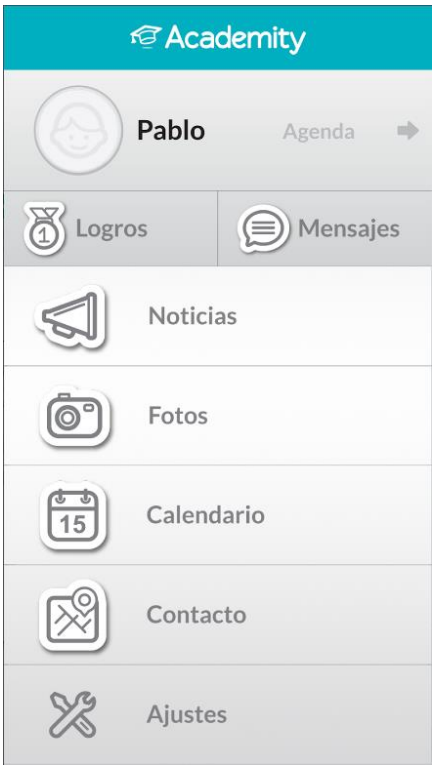

Ilustración 4 vista del perfil

3. Estudio de las distintas tecnologías

En el presente apartado se llevará a cabo un estudio de las diferentes tecnologías más relevantes a la hora de desarrollar aplicaciones móviles.

La mayoría de las tecnologías que se presentarán a continuación presentan ventajas e inconvenientes que deberán ser sopesados para seleccionar la mejor opción. A la hora de elegir una tecnología se deberá tener muy en cuenta estas ventajas e inconvenientes ya que tendrán un impacto crucial en el desarrollo de la solución final.

En cada una de estas tecnologías existen detractores y defensores con opiniones muy sesgadas que asegurarán que una tecnología es muy superior a otra, pero gracias a la experiencia se verá que cada tecnología puede ser una muy buena herramienta siempre que la utilicemos en el contexto y situación idónea.

Aunque existen más posibilidades a la hora de desarrollar una aplicación móvil vamos a centrarnos en las principales.

3.1. Aplicaciones nativas

3.1.1 Descripción

Las aplicaciones nativas [3] son aquellas desarrolladas específicamente para una plataforma móvil determinada, como iOS o Android. Estas aplicaciones se construyen utilizando los lenguajes de programación y las herramientas de desarrollo nativas de la plataforma objetivo, lo que les permite acceder a todas las características y funcionalidades del dispositivo, como la cámara, los sensores, el GPS y las notificaciones *push*.

Actualmente, muchas de las aplicaciones disponibles para móviles son nativas, es decir, están desarrolladas en el lenguaje nativo del móvil. En la tabla 1, se presentan los distintos lenguajes nativos para los dispositivos más usados del mercado.

Sistema operativo	**Fabricante**	**Lenguaje de programación**
iOs [4]	Apple	Objective C, Swift
Andoid [5]	Google	Java, Kotlin

Tabla 1 Lenguajes nativos y sus sistemas operativos

El desarrollo de aplicaciones nativas ofrece una serie de ventajas, como un rendimiento óptimo, una interfaz de usuario nativa y una mejor integración con las funcionalidades del dispositivo. Sin embargo, también presenta desafíos, como la necesidad de desarrollar y mantener versiones separadas para diferentes plataformas, lo que puede aumentar los costos y el tiempo de desarrollo.

En resumen, las aplicaciones nativas son una opción popular y poderosa para el desarrollo de aplicaciones móviles, permitiendo una experiencia más fluida y personalizada para los usuarios. Aunque requieren un enfoque específico para cada plataforma y pueden ser más costosas de desarrollar y mantener, ofrecen un alto rendimiento y acceso completo a las capacidades del dispositivo.

3.1.2 Ventajas e inconvenientes

Las aplicaciones nativas [3] ofrecen un rendimiento óptimo y una experiencia fluida para los usuarios, ya que están diseñadas específicamente para aprovechar al máximo las capacidades y características del sistema operativo y del dispositivo en el que se ejecutan. Además, al utilizar lenguajes de programación nativos y herramientas de desarrollo específicas de la plataforma, se logra una mejor integración con el entorno y una interfaz de usuario más intuitiva. Esto se traduce en una mayor velocidad de carga, tiempos de respuesta más rápidos y una navegación más suave.

Otra ventaja de las aplicaciones nativas es su capacidad para acceder a las funcionalidades del dispositivo, como la cámara, los sensores, el GPS y las notificaciones *push*. Esto permite crear experiencias interactivas y personalizadas, aprovechando al máximo las capacidades hardware del dispositivo móvil. Además, al estar disponibles en las tiendas de aplicaciones oficiales, las aplicaciones nativas tienen una mayor visibilidad y alcanzan a una amplia audiencia de usuarios potenciales.

El desarrollo de aplicaciones nativas puede implicar un mayor costo y tiempo de desarrollo en comparación con otras alternativas, como las aplicaciones web o híbridas. Esto se debe a que se requiere desarrollar versiones separadas de la aplicación para cada plataforma objetivo, como iOS y Android. Además, el mantenimiento y la actualización de las aplicaciones nativas pueden ser más complejos, ya que cualquier cambio o mejora debe realizarse en cada versión por separado.

Otro inconveniente de las aplicaciones nativas es la restricción a una sola plataforma. Si se desea llegar a una audiencia más amplia que utiliza diferentes sistemas operativos, se deberá desarrollar y mantener múltiples versiones de la aplicación. Esto implica un mayor esfuerzo y costos adicionales en comparación con las aplicaciones multiplataforma que se pueden desarrollar una vez y ejecutar en diferentes sistemas operativos.

3.2 Web app

3.2.1 Descripción

Las web apps, también conocidas como aplicaciones web, son aplicaciones accesibles a través de un navegador web y se ejecutan en servidores remotos en lugar de instalarse directamente en el dispositivo del usuario. Estas aplicaciones se construyen utilizando tecnologías web como HTML, CSS y JavaScript, y pueden ofrecer una amplia gama de funcionalidades y características similares a las aplicaciones nativas.

Las web apps son una alternativa a las aplicaciones nativas, ya que permiten a los usuarios acceder a ellas a través de un navegador web sin necesidad de descargar ni instalar nada en sus dispositivos. Estas aplicaciones se adaptan a diferentes tamaños de pantalla y sistemas operativos, lo que brinda una mayor flexibilidad y alcance para los usuarios.

3.2.2 Ventajas e inconvenientes

Una de las principales ventajas de las web apps es su compatibilidad multiplataforma. Al ser accesibles a través de un navegador web, las web apps se pueden utilizar en diferentes dispositivos y sistemas operativos, como ordenadores de escritorio, portátiles, tabletas y teléfonos móviles. Esto facilita su acceso y uso por parte de una amplia audiencia de usuarios.

Además, las web apps no requieren una instalación en el dispositivo del usuario, lo que las hace más convenientes y fáciles de utilizar. Los usuarios pueden acceder a ellas rápidamente a través de un

enlace o URL, sin tener que pasar por el proceso de descarga e instalación típico de las aplicaciones nativas.

Otra ventaja de las web apps es la facilidad de actualización. Al estar alojadas en servidores remotos, los cambios y actualizaciones realizados en la aplicación son instantáneamente visibles para los usuarios, sin necesidad de descargar una nueva versión. Esto permite una iteración más rápida y ágil en el desarrollo y mejora de la aplicación.

Aunque las web apps ofrecen una mayor compatibilidad y facilidad de acceso, también pueden tener limitaciones en términos de rendimiento y funcionalidad en comparación con las aplicaciones nativas. Debido a su naturaleza basada en navegador, las web apps pueden experimentar tiempos de carga más lentos y una respuesta menos fluida en comparación con las aplicaciones nativas optimizadas para el sistema operativo específico.

Además, las web apps pueden tener un acceso limitado a ciertas funcionalidades del dispositivo, como el hardware y los sensores, lo que puede restringir la capacidad de brindar experiencias interactivas y personalizadas.

En resumen, las web apps ofrecen una compatibilidad multiplataforma y un fácil acceso a través del navegador web, lo que las hace convenientes y ampliamente accesibles para los usuarios. Sin embargo, pueden tener limitaciones en términos de rendimiento y acceso a funcionalidades del dispositivo en comparación con las aplicaciones nativas optimizadas.

3.3 Aplicaciones híbridas

3.3.1 Descripción

En el desarrollo de aplicaciones móviles, una de las opciones que ha ganado popularidad en los últimos años son las aplicaciones híbridas [6]. Estas aplicaciones combinan elementos de las aplicaciones nativas y las aplicaciones web, permitiendo su funcionamiento en múltiples plataformas como iOS, Android y Windows. En este apartado, se explorará en detalle el concepto de aplicaciones híbridas, su funcionamiento y las ventajas e inconvenientes asociados a su uso.

Las aplicaciones híbridas son aquellas que se desarrollan utilizando tecnologías web como HTML, CSS y JavaScript, pero que se ejecutan dentro de un contenedor nativo en el dispositivo del usuario. Estas aplicaciones ofrecen una serie de beneficios al combinar la portabilidad y facilidad de desarrollo de las aplicaciones web con el acceso a características nativas del dispositivo, como la cámara, el GPS y las notificaciones.

3.3.2 Ventajas e inconvenientes

Portabilidad: Las aplicaciones híbridas pueden ejecutarse en múltiples plataformas, lo que ahorra tiempo y esfuerzo en el desarrollo de aplicaciones específicas para cada sistema operativo.

Costo-efectividad: Al tener un código base compartido entre plataformas, el desarrollo y mantenimiento de aplicaciones híbridas puede resultar más económico en comparación con las aplicaciones nativas.

Actualizaciones rápidas: Las actualizaciones y correcciones de errores se pueden implementar de forma rápida y sencilla, ya que se realizan en el servidor y no es necesario esperar la aprobación de las tiendas de aplicaciones.

Rendimiento: En comparación con las aplicaciones nativas, las aplicaciones híbridas pueden presentar un rendimiento ligeramente inferior, especialmente en tareas intensivas en recursos o que requieren un acceso rápido a características nativas del dispositivo.

Limitaciones de acceso a funciones nativas: Aunque las aplicaciones híbridas pueden acceder a muchas características del dispositivo a través de plugins, es posible que no tengan acceso completo a todas las funcionalidades nativas, lo que puede limitar ciertas capacidades.

Experiencia de usuario: Al intentar abarcar diferentes plataformas, es posible que las aplicaciones híbridas no ofrezcan la misma experiencia de usuario que las aplicaciones nativas, ya que pueden no aprovechar al máximo las características específicas de cada plataforma.

En resumen, las aplicaciones híbridas ofrecen una solución intermedia entre las aplicaciones nativas y las aplicaciones web, brindando ventajas como la portabilidad y el ahorro de costos, pero también presentando desafíos en términos de rendimiento y experiencia de usuario. La elección de utilizar aplicaciones híbridas dependerá de las necesidades específicas del proyecto y las prioridades del equipo de desarrollo.

3.4 Aplicaciones multiplataforma

3.4.1 Descripción

Las aplicaciones multiplataforma son aplicaciones de software que se desarrollan utilizando un único lenguaje de programación y pueden ejecutarse en múltiples plataformas, como iOS, Android y Windows. Estas aplicaciones se crean utilizando frameworks y herramientas que permiten la escritura de código compartido y la adaptación a las especificidades de cada plataforma. La principal característica de este tipo de aplicaciones es que pueden ser ejecutadas en diferentes plataformas sin necesidad de modificar significativamente el código.

3.4.2 Ventajas e inconvenientes

Las principales ventajas de las aplicaciones multiplataforma son:

- Ahorro de tiempo y recursos: Al utilizar un único código base, el desarrollo de aplicaciones multiplataforma puede ser más eficiente y rápido en comparación con el desarrollo de aplicaciones nativas para cada plataforma por separado.
- Amplio alcance: Al desarrollar una aplicación multiplataforma, se puede llegar a una mayor audiencia, ya que se puede lanzar en diferentes plataformas simultáneamente. Esto permite abarcar un espectro más amplio de usuarios y maximizar el alcance de la aplicación.
- Mantenimiento simplificado: Al tener un solo código base, el mantenimiento y la actualización de la aplicación se vuelven más sencillos. Los cambios y correcciones se pueden implementar en todo el conjunto de plataformas con menos esfuerzo y tiempo, lo que facilita la gestión a largo plazo.

Desventajas de las aplicaciones multiplataforma:

- Limitaciones de rendimiento: Aunque los frameworks multiplataforma han mejorado significativamente en términos de rendimiento, es posible que no alcancen el mismo nivel de rendimiento que las aplicaciones nativas en algunas situaciones. Esto puede ser especialmente relevante para aplicaciones que requieren un alto rendimiento.

- Acceso limitado a características del dispositivo: Al desarrollar una aplicación multiplataforma, es posible que no se tenga acceso completo a todas las características y capacidades específicas de cada plataforma. Algunas características avanzadas pueden no estar disponibles o requerir soluciones alternativas.
- Dependencia de terceros: Las aplicaciones multiplataforma suelen depender de frameworks y herramientas proporcionadas por terceros. Esto implica que los desarrolladores deben confiar en la actualización y el soporte continuo de estas herramientas para mantener la compatibilidad con las nuevas versiones de las plataformas.

3.5 Decisión motivada sobre la tecnología elegida para la aplicación

Después de haber realizado un estudio de las principales tecnologías para desarrollar aplicaciones móviles, se ha elegido la tecnología multiplataforma como la mejor candidata para desarrollar este proyecto.

La elección de desarrollar una aplicación multiplataforma para la implementación de este proyecto se basa en diversas consideraciones estratégicas. En primer lugar, las aplicaciones multiplataforma están experimentando un notable crecimiento en la industria, lo que brinda una oportunidad propicia para adquirir conocimientos y especialización en este ámbito en constante evolución.

El marco de desarrollo seleccionado para este proyecto es Flutter, el cual se abordará detalladamente en el siguiente capítulo. Flutter es un framework de desarrollo multiplataforma que ofrece una amplia gama de herramientas y recursos que facilitan la creación de aplicaciones para diferentes plataformas, como iOS, Android, web y escritorio.

Asimismo, es fundamental que la aplicación pueda ser distribuida en los principales mercados, como Play Store y Apple Store. La ventaja de optar por una aplicación multiplataforma radica en la posibilidad de desarrollarla una vez y poder lanzarla en múltiples plataformas, lo cual resulta más eficiente en términos de tiempo y recursos. Además, cabe destacar que el desarrollar una aplicación y poder lanzarla en los mercados de Play Store y Apple Store es una característica fundamental para el público al que va dirigido. El poder tener la aplicación en este formato facilita que los usuarios la descarguen y puedan usarla con más facilidad que si tuviesen que acceder a una aplicación web por medio de una URL, que supondría el recordar dicha dirección web.

Considerando las características y requisitos del proyecto, no se requiere un alto rendimiento ni una interacción compleja con los sensores o prestaciones del sistema operativo. Por lo tanto, no es necesario desarrollar la aplicación en lenguaje nativo, ya que las ventajas de una aplicación multiplataforma, como la flexibilidad y la agilidad en el desarrollo, se ajustan perfectamente a las necesidades planteadas.

En resumen, la decisión de utilizar una aplicación multiplataforma para el desarrollo de una aplicación para academias de refuerzo se basa en la oportunidad de crecimiento en el sector, la compatibilidad con el marco de desarrollo Flutter, la distribución en múltiples mercados y la adecuación a los requisitos de rendimiento y funcionalidad del proyecto.

4. Tecnologías empleadas en el desarrollo de la aplicación

Una vez vistas las principales tecnologías para realizar aplicaciones y elegida la tecnología híbrida, es momento de adentrarnos en los componentes clave empleados en el desarrollo de la aplicación. En este apartado, se analizarán el lenguaje de programación, el framework y la base de datos seleccionados para el proyecto, detallando las razones que fundamentaron estas elecciones.

4.1 Flutter

Flutter [7] es un framework de desarrollo de aplicaciones móviles de código abierto creado por Google. Desde su lanzamiento en 2017, Flutter ha ganado rápidamente popularidad en la comunidad de desarrolladores debido a su capacidad para crear aplicaciones de alta calidad y rendimiento tanto para Android como para iOS. Flutter utiliza un enfoque de desarrollo híbrido, lo que significa que una sola base de código puede ser utilizada para crear aplicaciones que se ejecuten en múltiples plataformas, lo que ahorra tiempo y recursos.

Flutter se distingue por su arquitectura basada en widgets, lo que permite crear interfaces de usuario atractivas de manera rápida y sencilla. Los widgets en Flutter son personalizables y altamente flexibles, lo que permite una gran libertad creativa en el diseño de la interfaz de usuario y cuenta con una amplia gama de widgets [8] predefinidos, lo que acelera el desarrollo y facilita la creación de interfaces coherentes y visualmente atractivas.

Contiene un gran número de bibliotecas y paquetes adicionales que amplían las funcionalidades de Flutter y facilitan el desarrollo de aplicaciones. Estas bibliotecas ofrecen soluciones predefinidas para tareas comunes, como el manejo de bases de datos, la integración con servicios en la nube, la animación de interfaces de usuario y mucho más. La comunidad de desarrolladores de Flutter es activa y contribuye constantemente con nuevas bibliotecas y paquetes, lo que brinda a los desarrolladores acceso a una amplia gama de recursos y herramientas para mejorar la eficiencia y calidad de sus aplicaciones.

Una de las características más destacadas de Flutter es su capacidad para ofrecer un rendimiento nativo. A diferencia de otros frameworks híbridos, Flutter no utiliza WebViews ni interpreta el código en tiempo de ejecución, lo que se traduce en un rendimiento rápido y fluido. Flutter utiliza su propio motor de renderizado de alto rendimiento llamado Skia, que aprovecha la aceleración de hardware para brindar una experiencia de usuario suave y receptiva.

Los puntos más destacables [9] de Flutter y por lo que se diferencia de otros frameworks es:

- **Productividad**: Flutter simplifica el desarrollo de aplicaciones al permitir la reutilización de código en diferentes plataformas, lo que acelera el proceso de desarrollo y reduce los costos.
- **Interfaz de usuario atractiva**: Los widgets personalizables y la arquitectura basada en widgets de Flutter permiten crear interfaces de usuario visualmente atractivas y coherentes en todas las plataformas.
- **Rendimiento nativo**: Flutter ofrece un rendimiento rápido y fluido gracias a su motor de renderizado de alto rendimiento. Esto garantiza una experiencia de usuario suave y sin interrupciones.
- **Hot Reload**: La función de Hot Reload de Flutter permite realizar cambios en tiempo real en el código y ver los resultados de inmediato, lo que acelera el proceso de desarrollo y facilita la depuración.

- **Comunidad activa y soporte**: Flutter cuenta con una comunidad de desarrolladores activa y en constante crecimiento, lo que ofrece un sólido soporte, tutoriales, bibliotecas y herramientas adicionales para facilitar el desarrollo de aplicaciones.

4.2 Dart

Dart es un lenguaje de programación versátil y poderoso que se ha convertido en la base del desarrollo de aplicaciones en Flutter. Diseñado por Google, Dart se destaca por su combinación única de características que lo hacen ideal para la creación de aplicaciones móviles modernas y de alto rendimiento. Con una sintaxis clara y concisa, Dart facilita la escritura de código legible y mantenible, lo que acelera el proceso de desarrollo y reduce la posibilidad de errores.

Una de las principales ventajas de Dart es su capacidad para realizar compilación en tiempo de ejecución (JIT) y compilación anticipada (AOT), lo que permite una ejecución rápida y eficiente de las aplicaciones en diferentes plataformas. Cuenta con un sistema de recolección de basura eficiente y un motor de ejecución optimizado, lo que garantiza un rendimiento fluido y una excelente experiencia de usuario.

Otro aspecto destacado de Dart es su enfoque en la programación orientada a objetos, lo que permite una estructuración clara y modular del código. El lenguaje ofrece soporte para características avanzadas como herencia, polimorfismo y interfaces, lo que facilita la creación de aplicaciones escalables y extensibles.

Dart también ofrece una amplia gama de bibliotecas y paquetes que cubren una variedad de necesidades, desde manipulación de datos y acceso a bases de datos hasta interacción con servicios web y desarrollo de interfaces de usuario. Estas bibliotecas, combinadas con la comunidad activa de desarrolladores de Dart, proporcionan un ecosistema sólido y en constante crecimiento que facilita la implementación de funcionalidades avanzadas en las aplicaciones.

4.3 Visual Studio Code vs Android Studio

Visual Studio Code y Android Studio [10] son dos entornos de desarrollo integrados (IDE) ampliamente utilizados para la creación de aplicaciones. Visual Studio Code, desarrollado por Microsoft, es un editor de código ligero y altamente personalizable que ofrece una amplia gama de características y extensiones para facilitar la escritura de código. Por otro lado, Android Studio, creado por Google, es un IDE específicamente diseñado para el desarrollo de aplicaciones Android, que proporciona herramientas especializadas y una integración profunda con el ecosistema de Android.

Visual Studio Code (Ilustración destaca por su rapidez y eficiencia en el desarrollo. Su interfaz minimalista y su rendimiento optimizado permiten una experiencia de codificación fluida y sin interrupciones. A lo que se añade su amplia colección de extensiones que permite a los desarrolladores personalizar su entorno de trabajo y adaptarlo a sus necesidades específicas.

Por otro lado, Android Studio se destaca por su funcionalidad y características especializadas para el desarrollo de aplicaciones Android. Proporciona un conjunto completo de herramientas para diseñar interfaces de usuario, depurar y analizar el rendimiento de la aplicación, así como para emular y probar en dispositivos virtuales

En cuanto a la elección, se ha optado por Visual Studio Code debido a su rapidez, funcionalidad y simplicidad. El entorno ligero y altamente personalizable de Visual Studio Code permite a los

desarrolladores trabajar de manera eficiente y adaptar el entorno a sus preferencias y necesidades específicas. Su amplia lista de extensiones y su capacidad de personalización ofrecen flexibilidad y posibilidades adicionales para mejorar la productividad y la experiencia de desarrollo.

4.4 Tecnología de base de datos: SQL vs NOSQL

En el ámbito del desarrollo de aplicaciones, la elección de la tecnología de base de datos es un aspecto fundamental que puede tener un impacto significativo en el rendimiento, la escalabilidad y la eficiencia de un sistema. En este punto, se realizará un análisis comparativo entre las tecnologías de base de datos SQL y NoSQL, examinando sus características, ventajas y desventajas, con el fin de determinar cuál es la más adecuada para el proyecto en cuestión.

La tecnología de base de datos SQL [11] (Structured Query Language) se basa en un modelo relacional, donde la información se organiza en tablas compuestas por filas y columnas. Utiliza consultas SQL para acceder y manipular los datos, garantizando una estructura predefinida y relaciones entre las tablas. Las bases de datos SQL son conocidas por su consistencia, integridad y soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que las hace ideales para aplicaciones que requieren una estricta estructura y coherencia de los datos.

Por otro lado, la tecnología de base de datos NoSQL [12] (Not Only SQL) es una alternativa a las bases de datos SQL tradicionales. Estas bases de datos se caracterizan por su flexibilidad en el esquema de datos, permitiendo almacenar y recuperar información no estructurada o semiestructurada de manera eficiente. Las bases de datos NoSQL se basan en diferentes modelos de datos, como clave-valor, documentos, columnas o grafos, y son altamente escalables y tolerantes a fallos. Además, suelen ofrecer un alto rendimiento y escalabilidad horizontal, lo que las convierte en una opción adecuada para aplicaciones que manejan grandes volúmenes de datos o requieren una rápida lectura y escritura.

La elección [13] entre una base de datos SQL y NoSQL dependerá de los requisitos específicos del proyecto. Las bases de datos SQL son ideales cuando se necesita una estructura rígida y relaciones complejas entre los datos, como en aplicaciones que gestionan transacciones financieras o información altamente relacionada. Proporcionan un alto nivel de consistencia y garantizan la integridad de los datos, pero pueden ser menos flexibles en términos de escalabilidad y adaptación a cambios en el esquema.

Por otro lado, las bases de datos NoSQL ofrecen flexibilidad y escalabilidad, lo que las hace adecuadas para aplicaciones con requisitos cambiantes o que manejan grandes volúmenes de datos. Son especialmente eficientes en entornos distribuidos y brindan un rendimiento rápido para operaciones de lectura y escritura. Sin embargo, las bases de datos NoSQL pueden carecer de algunas características de las bases de datos SQL, como la capacidad de realizar consultas complejas y mantener relaciones entre los datos.

En resumen, la elección entre SQL y NoSQL dependerá de las necesidades del proyecto, considerando factores como la estructura de los datos, los requisitos de escalabilidad, el rendimiento y la flexibilidad.

4.4.1 Google Firebase

Una vez vistos los puntos positivos y negativos de cada tecnología de base de datos, se ha decidido utilizar una base de datos del tipo NoSQL para el proyecto en cuestión. En este sentido, se ha optado

por la plataforma de desarrollo Google Firebase [14], que ofrece una solución integral y robusta para el almacenamiento y gestión de datos en aplicaciones móviles y web.

Firebase utiliza una estructura de base de datos NoSQL que permite una mayor flexibilidad y escalabilidad, lo cual resulta beneficioso para el desarrollo de la aplicación en términos de eficiencia y adaptabilidad a futuras necesidades. A continuación, se describirán en detalle las características y puntos positivos de Google Firebase como tecnología de base de datos.

Google Firebase es una plataforma desarrollada por Google que proporciona una amplia gama de servicios para la construcción y gestión de aplicaciones móviles y web. En cuanto a la base de datos, Firebase utiliza una estructura NoSQL denominada Firebase Realtime Database. Este enfoque se basa en la idea de almacenar los datos en forma de árbol JSON que ofrece una gran flexibilidad y agilidad en la manipulación de la información. Además, Firebase ofrece otras funcionalidades clave como la autenticación de usuarios, el almacenamiento en la nube, la mensajería en tiempo real y la integración con herramientas de análisis y rendimiento.

Una de las principales ventajas de utilizar Google Firebase como tecnología de base de datos es su escalabilidad y rendimiento. Firebase está diseñado para manejar grandes volúmenes de datos y soportar aplicaciones con una gran cantidad de usuarios concurrentes. Esto asegura que la aplicación pueda crecer y adaptarse a medida que aumenta la demanda y el número de usuarios.

Otro punto positivo de Firebase es su facilidad de uso y rápida implementación. La plataforma proporciona una interfaz intuitiva y herramientas de desarrollo que permiten a los desarrolladores agilizar el proceso de creación y despliegue de la aplicación. Firebase posee una amplia documentación y recursos de soporte que facilitan el aprendizaje y la resolución de problemas.

También cuenta con una amplia gama de características adicionales que complementan la funcionalidad de la base de datos, como la autenticación de usuarios, la gestión de notificaciones push, el almacenamiento en la nube y la integración con herramientas de análisis y rendimiento. Estas funcionalidades adicionales permiten enriquecer la aplicación y brindar una mejor experiencia de usuario.

Las herramientas de análisis y rendimiento puede ser un añadido muy interesante a la hora de elegir esta plataforma ya que ofrecerá información de cuántas personas están utilizando la aplicación y como estos usuarios interactúan con la misma proporcionando estadísticas sin tener que implementar herramientas y programas de terceros.

5. Análisis del problema

Tras haber explorado el contexto tecnológico y llevado a cabo un exhaustivo análisis de las diferentes soluciones disponibles, es momento de profundizar en los desafíos y problemas a los que se enfrentará este proyecto.

En el presente capítulo de la memoria, se establecerán de manera precisa todos los requisitos que el sistema deberá cumplir, así como una descripción detallada de los casos de uso que se pretenden abordar. Estos casos de uso, serán fundamentales para definir los diferentes flujos que seguirá un usuario al interactuar con la aplicación. Se dedicará una sección especial en este capítulo para la elaboración del diagrama de clases del proyecto, proporcionando así una representación visual de la estructura y las relaciones entre las distintas entidades del sistema.

5.1 Especificación de requisitos

La especificación de requisitos es una etapa fundamental en el proceso de desarrollo del proyecto, ya que permite establecer de manera clara y precisa que debe realizar el sistema. En este apartado, se detallarán los requerimientos funcionales y no funcionales, así como las restricciones y objetivos a alcanzar. Esta especificación brinda una base sólida para el diseño y la implementación, asegurando que el sistema cumpla con las expectativas de los usuarios y las metas del proyecto.

5.1.1 Requisitos funcionales

La especificación de los requisitos funcionales es esencial para que el sistema defina las funciones y características que debe tener. Estos requisitos describen detalladamente las acciones que debe realizar el sistema y cómo debe responder a las interacciones del usuario. Su importancia radica en que proporciona una guía clara y precisa para el diseño y desarrollo del sistema, permitiendo al equipo de trabajo comprender las necesidades y expectativas de los usuarios. Al definirse los requisitos funcionales de manera adecuada, se asegura que el sistema cumpla con las funcionalidades requeridas, dando una experiencia óptima y satisfactoria al usuario final.

A continuación, se procederá a definirlos:

Identificador	Descripción
F1	Creación de un sistema de inicio de sesión
F2	Permitir la creación de usuarios alumnos
F3	Permitir la creación de usuarios administradores
F4	Permitir la eliminación de usuarios
F5	Permitir la modificación de usuarios
F6	Permitir la creación horarios

F7	Permitir la asignación de horarios a usuarios
F8	Permitir la eliminación de horarios
F9	Permitir la modificación de horarios
F10	Permitir la creación de actividades
F11	Permitir la asignación de documentos a actividades
F12	Permitir la asignación de usuarios a actividades
F14	Permitir la modificación de actividades
F15	Permitir la eliminación de actividades
F16	Permitir la creación de documentos en la aplicación
F17	Permitir la asignación de usuarios a documentos
F18	Permitir la modificación de documentos
F19	Permitir la eliminación de documentos
F20	Permitir la descarga de documentos
F21	Permitir la creación de avisos en la aplicación
F22	Permitir la modificación de avisos
F23	Permitir la eliminación de avisos

Tabla 2 Requisitos funcionales

5.1.2 Requisitos no funcionales

Los requisitos no funcionales son una parte fundamental en la definición del sistema, ya que establecen los atributos y características que no están relacionados directamente con las funciones del sistema, pero que son igualmente importantes. Estos requisitos incluyen aspectos como el rendimiento, la seguridad, la usabilidad y la escalabilidad. Su importancia radica en que garantizan que el sistema cumpla con estándares de calidad, eficiencia y satisfacción del usuario. Al atender los requisitos no funcionales de manera adecuada, se logra un sistema confiable, seguro y eficiente, ofreciendo una experiencia positiva y cumpliendo con las expectativas de los usuarios.

A continuación, se procederá a definirlos:

Identificador	Descripción

F1	La aplicación debe ser capaz de manejar una carga de usuarios concurrentes sin experimentar retrasos significativos
F2	La aplicación debe de tener tiempos de respuesta rápidos y una buena capacidad de escalabilidad para adaptarse al crecimiento de nuevos usuarios.
F3	La aplicación debe garantizar la protección de los datos sensibles de los usuarios.
F4	La aplicación debe implementar medidas de seguridad robustas, como encriptación de datos y autenticación de usuarios.
F5	La interfaz de usuario debe ser intuitiva y fácil de usar, con una navegación clara y una experiencia fluida.
F6	Los usuarios deben poder acceder y utilizar las funciones de la aplicación sin dificultad, independientemente de su nivel de experiencia tecnológica.
F7	La aplicación debe estar disponible en todo momento, sin interrupciones significativas de servicio.
F8	La aplicación debe ser compatible con diferentes dispositivos.
F9	La interfaz debe tener una buena capacidad de adaptación a diferentes tamaños de pantalla y sistemas operativos.
F10	La aplicación debe ser fácil de mantener y actualizar. Debe contar con una arquitectura bien estructurada y un código limpio y documentado.

Tabla 3 Requisitos no funcionales

5.2 Casos de uso

En esta sección, se describirán de manera detallada las interacciones y flujos que los usuarios seguirán al utilizar la aplicación. Mediante una serie de escenarios, se analizarán las diversas funcionalidades y características que se espera que la aplicación ofrezca a los usuarios. Estos casos de uso proporcionarán una visión clara y concisa de cómo los diferentes actores interactúan con el sistema, permitiendo comprender las necesidades y expectativas de los usuarios. Esta información será fundamental para el diseño, desarrollo y pruebas de la aplicación, asegurando que cumpla con los requisitos establecidos y de una experiencia óptima para los usuarios finales.

Para ello comenzaremos mostrando los casos de usos generales que cubrirá la aplicación para posteriormente detallar cada uno de ellos más específicamente.

Un usuario de tipo profesor puede: crear, modificar y eliminar usuarios sin límites. Crear, modificar y eliminar clases, actividades y documentos, todos ellos pudiendo ser asignados a un usuario tipo alumno. Crear, modificar y eliminar avisos que serán enviados a todos los usuarios registrados en el sistema. Por último, podrá ver detalles de todos los apartados mencionados anteriormente: clases, actividades, documentos y avisos.

Los usuarios de tipo alumno únicamente podrán ver las clases, actividades, documentos y avisos que han sido asignados a su usuario además de los detalles de los mismos. Tendrán una funcionalidad especial en las actividades ya que podrán cambiar el estado de la actividad a “completada”.

5.2.1 Descripción de los actores

Los actores son entidades externas que interactúan con el sistema, desempeñan roles específicos y tienen ciertos objetivos o acciones que realizan dentro del sistema.

Dentro de este marco se describirán los siguientes actores:

1. **Alumnos:** Son los usuarios que acceden a la aplicación en calidad de estudiantes. Tienen la capacidad de iniciar sesión, navegar por los diferentes apartados (horarios, actividades, documentos, avisos) e interactuar con las actividades y documentos asignados.
2. **Profesores:** Son los usuarios con rol docente. Tienen acceso a funcionalidades adicionales, como crear, modificar y eliminar usuarios (alumnos y profesores), gestionar clases, actividades, documentos y avisos.

5.2.2 Diagrama de los casos de uso

Esta sección es una representación visual que permite comprender de manera clara y concisa las interacciones entre los diferentes actores y el sistema en desarrollo. En la ilustración 5 y 6 podremos ver la representación de este diagrama, donde se pueden identificar y visualizar los casos de uso, que describen las acciones y funcionalidades que los usuarios pueden realizar en el sistema. Proporciona una visión general de las relaciones entre los actores y los casos de uso, facilitando la comprensión de la lógica y el flujo de trabajo del sistema.

Ya que se disponen de dos actores en la aplicación se realizará un diagrama de los casos de uso de cada uno de los actores.

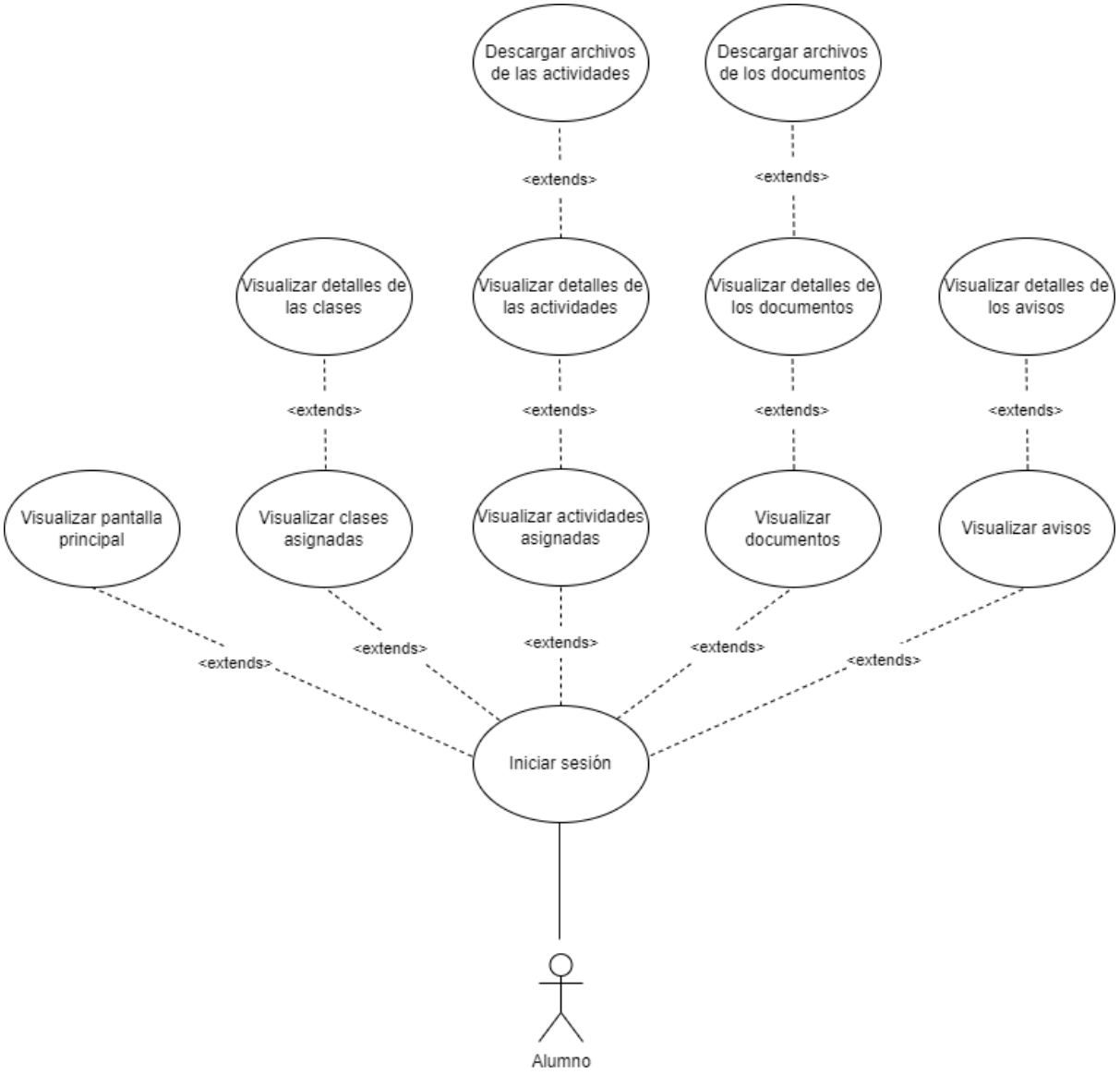

Ilustración 5 Diagrama de casos de uso de alumno

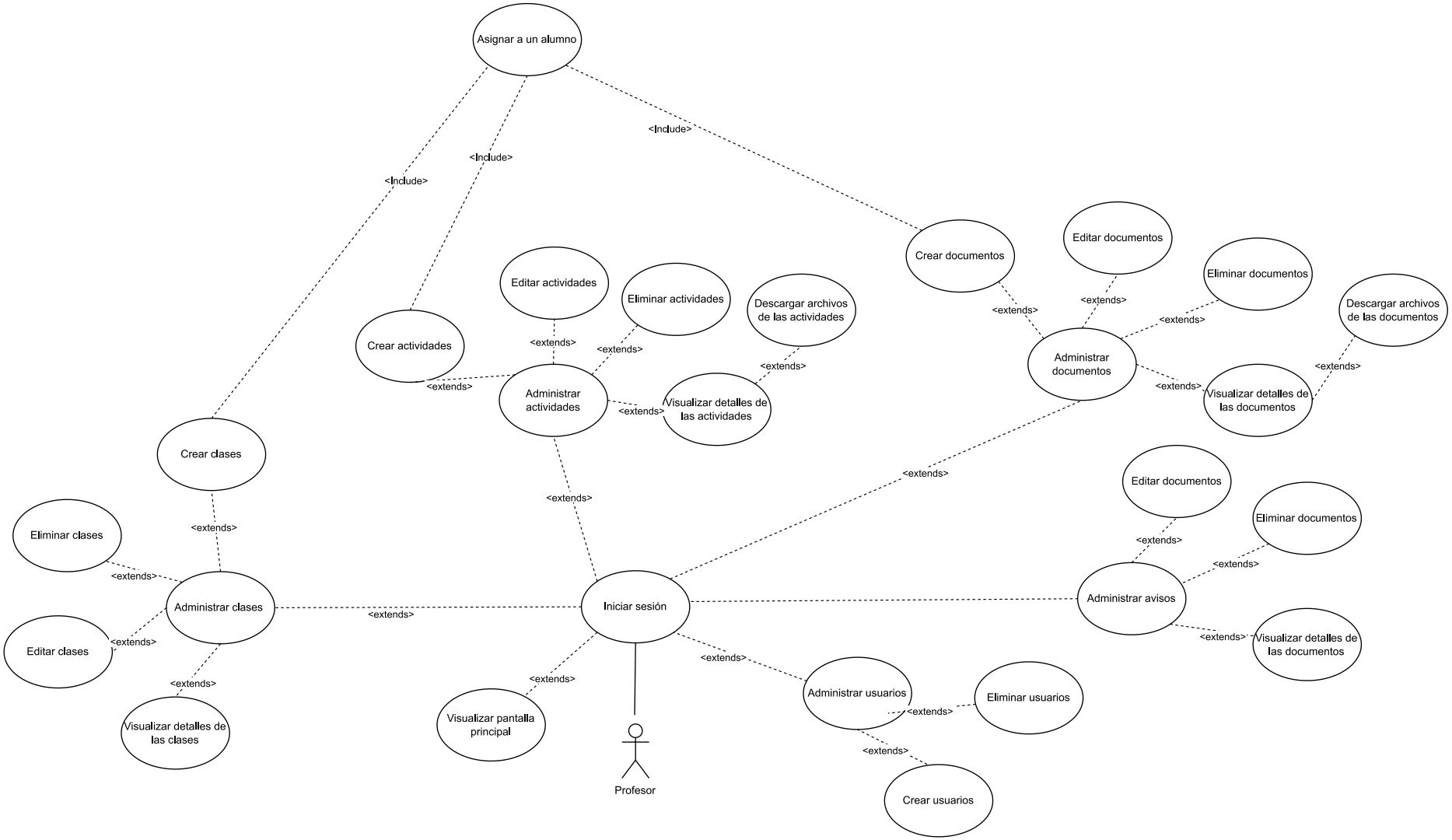

Ilustración 6 Diagrama de casos de uso de profesor

5.2.3 Descripción de los casos de uso

En este apartado se describirán los casos de uso generales y los que derivan de estos.

Casos de uso de usuario correspondiente a la Ilustración 1:

Nombre	Visualizar pantalla principal
Descripción	Un usuario visualizará la pantalla principal de la aplicación una vez que inicie sesión ya que será la primera pantalla que se muestre al usuario. En la pantalla principal se mostrará: La próxima clase que tenga el alumno La actividad más reciente que se le haya propuesto al alumno El último aviso que los profesores hayan publicado.
Secuencia de uso	A: 1. Estar autenticado en el sistema B: 2. Estar autenticado en el sistema 3. Presionar en el menú el icono de “pantalla principal”
Postcondición	
Excepciones	

Tabla 4 Caso de uso de alumno: visualización de pantalla principal

Nombre	Visualizar clases asignadas
Descripción	Un usuario podrá acceder a las clases asignadas una vez que haya iniciado sesión. En este punto el usuario podrá visualizar todas las próximas clases que le han sido asignadas por los profesores.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Calendario”
Postcondición	
Excepciones	

Tabla 5 Caso de uso de alumno: visualización de clases asignadas

Nombre	Visualizar actividades asignadas
Descripción	Un usuario podrá acceder a las actividades asignadas una vez que haya iniciado sesión. En este punto el usuario podrá visualizar todas las próximas clases que le han sido asignadas por los profesores. Si el usuario presiona en alguna actividad se le abrirá un cuadro de diálogo donde: - Podrá visualizar los detalles de la actividad - Podrá descargar documentos asociados a la actividad en el caso de que los tenga - Podrá marcar la actividad como “Realizada”
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Actividad” B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Actividad” 3. Presionar en el botón “Descargar” C: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Actividad” 3. Presionar en el botón “Realizada”
Postcondición	- Una vez presionado en el botón de “Descargar” se enviará una petición a la base de datos que descargará el archivo en el dispositivo móvil del alumno. - Una vez presionado el botón de “Realizada” se enviará una petición a la base de datos donde se actualizará el estado de la actividad siendo visible tanto para el profesor como para el alumno.
Excepciones	La referencia al archivo asociado a la actividad ha sido perdida y no se puede descargar el archivo.

Tabla 6 Caso de uso de alumno: visualización de actividades asignadas

Nombre	Visualizar documentos
Descripción	Un usuario podrá acceder a los documentos una vez que haya iniciado sesión. En este punto el usuario podrá visualizar todos los documentos que le han sido asignados. Si el usuario presiona en algún documento se le abrirá un cuadro de diálogo donde: - Podrá visualizar los detalles del documento - Podrá descargar el documento
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Documento”

	B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Actividad” 3. Presionar en el botón “Descargar”
Postcondición	Una vez presionado en el botón de “Descargar” se enviará una petición a la base de datos que descargará el archivo en el dispositivo móvil del alumno.
Excepciones	La referencia al archivo asociado ha sido perdida y no se puede descargar el archivo.

Tabla 7 Caso de uso de alumno: visualización de documentos

Nombre	Visualizar avisos
Descripción	Un usuario podrá acceder a los avisos una vez que haya iniciado sesión. En este punto el usuario podrá visualizar todos los avisos que los profesore hayan publicado. Si el usuario presiona en algún aviso se le abrirá un cuadro de diálogo donde: - Podrá visualizar los detalles del aviso.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “Avisos”
Postcondición	
Excepciones	

Tabla 8 Caso de uso de alumno: visualización de avisos

Casos de uso de profesor correspondiente a la Ilustración 2:

Nombre	Visualizar pantalla principal
Descripción	Un profesor visualizará la pantalla principal de la aplicación una vez que inicie sesión ya que será la primera pantalla que se muestre al usuario. En la pantalla principal se mostrará: - La próxima clase que tenga el profesor - La actividad más reciente que haya creado el profesor - El último aviso que los profesores hayan publicado.
Secuencia de uso	A: 1. Estar autenticado en el sistema B: 2. Estar autenticado en el sistema 3. Presionar en el menú el icono de “pantalla principal”

Postcondición	
Excepciones	

Tabla 9 Caso de uso de profesor: visualización de pantalla principal

Nombre	Administrar usuarios
Descripción	Un profesor podrá acceder a los ajustes del sistema presionando es su foto de perfil. Una vez dentro de los ajustes podrá administrar los usuarios: - Crear un nuevo usuario de tipo alumno o profesor especificando: nombre completo, correo electrónico, contraseña y rol. - Eliminar un usuario de tipo alumno o profesor.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en la foto de perfil 3. Presionar en crear un nuevo usuario 4. Rellenar todos los datos del cuadro de diálogo 5. Presionar en “Crear” B: 1. Estar autenticado en el sistema 2. Presionar en la foto de perfil 3. Presionar en “Eliminar usuario” 4. Seleccionar el usuario a eliminar 5. Presionar en “Eliminar”
Postcondición	- Una vez presionado en el botón de “Crear” se enviará una petición a la base de datos que creará el usuario y le permitirá iniciar sesión - Una vez presionado en el botón de “Eliminar” se enviará una petición a la base de datos que eliminará el usuario imposibilitando que este inicie sesión en la aplicación
Excepciones	- Formato erróneo (En cualquiera de los campos). - Campos requeridos no rellenados.

Tabla 10 Caso de uso de profesor: administración de usuarios

Nombre	Administrar clases
Descripción	Un profesor podrá acceder a la vista para administrar las clases y podrá: - Visualizar todas las clases que ha creado y aún no han sido impartidas. - Presionar en una clase para poder modificarla o eliminarla.

	- Presionar en el botón de “más” para crear una nueva clase donde tendrá que rellenar los datos de: título de la clase, alumno, día de la clase y hora.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “clases” 3. Presionar en una clase 4. Editar los datos de la clase seleccionada 5. Presionar en “Actualizar” B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “clases” 3. Presionar en una clase 4. Presionar en “Eliminar” C: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “clases” 3. Presionar en el botón “más” 4. Rellenar todos los datos de la nueva clase 5. Presionar en el botón de “Aceptar”
Postcondición	- Una vez presionado en el botón de “Crear” se enviará una petición a la base de datos que creará la nueva clase y podrá ser visualizada tanto por el profesor como por el alumno. - Una vez presionado el botón de “Actualizar” se enviará una petición a la base de datos que actualizará la clase seleccionada y se mostrará la clase actualizada tanto al profesor como al alumno. - Una vez presionado en el botón de “Eliminar” se enviará una petición a la base de datos que eliminará la clase seleccionada haciendo que esta desaparezca del listado de clases tanto del alumno como del profesor.
Excepciones	- Formato erróneo (En cualquiera de los campos). - Campos requeridos no rellenados.

Tabla 11 Caso de uso de profesor: administración de clases

Nombre	Administrar actividades
Descripción	Un profesor podrá acceder a la vista para administrar las actividades y se le mostrará un listado con todos los alumnos de la aplicación. Una vez que seleccione a un alumno viajará a una nueva pestaña donde: - Se le mostrarán todas las actividades que el alumno tiene asociadas. - Seleccionando alguna actividad se le abrirá un cuadro diálogo donde aparecerán detalles de la actividad y podrá descargar archivos asociados en caso de que los tenga. También podrá presionar en el botón de editar para poder

	editar los datos de la actividad y el botón de eliminar donde la actividad será borrada. - Presionar el botón de “más” para poder crear una nueva actividad.
Secuencia de uso	A: 5. Estar autenticado en el sistema 6. Presionar en el menú el icono de “actividades” 7. Presionar en el nombre de algún alumno 8. Presionar en el botón de “más” 9. Rellenar todos los campos del cuadro de diálogo 10. Presionar en “Crear” B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “actividades” 3. Presionar en el nombre de algún alumno 4. Presionar en alguna actividad que tenga el alumno 5. Presionar el botón de “Descargar” C: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “actividades” 3. Presionar en el nombre de algún alumno 4. Presionar en alguna actividad que tenga el alumno 5. Presionar el botón de “editar” 6. Cambiar los datos de la actividad 7. Presionar en el botón de “Actualizar” B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “actividades” 3. Presionar en el nombre de algún alumno 4. Presionar en alguna actividad que tenga el alumno 5. Presionar el botón de “Eliminar”
Postcondición	- Una vez presionado en el botón de “Crear” se enviará una petición a la base de datos que creará la nueva actividad y podrá ser visualizada tanto por el profesor como por el alumno. - Una vez presionado en el botón de “Descargar” se enviará una petición a la base de datos que descargará el archivo asociado a la actividad en el dispositivo del usuario. - Una vez presionado el botón de “Actualizar” se enviará una petición a la base de datos que actualizará la actividad seleccionada y se mostrará la clase actualizada tanto al profesor como al alumno. - Una vez presionado en el botón de “Eliminar” se enviará una petición a la base de datos que eliminará la actividad seleccionada haciendo que esta desaparezca del listado de clases tanto del alumno como del profesor.

Excepciones	- Formato erróneo (En cualquiera de los campos). - Campos requeridos no rellenados. - La referencia al archivo asociado ha sido perdida y no se puede descargar el archivo.

Tabla 12 Caso de uso de profesor: administración de actividades

Nombre	Administrar documentos
Descripción	Un profesor podrá acceder a la vista para administrar los documentos y se le mostrará un listado con todos los alumnos de la aplicación. Una vez que seleccione a un alumno viajará a una nueva pestaña donde: - Se le mostrarán todos los documentos que el alumno tiene asociados. - Seleccionando algún documento se le abrirá un cuadro diálogo donde aparecerán detalles del documento y podrá descargar el archivo asociado al documento. También podrá presionar en el botón de editar para poder editar los datos del documento y el botón de eliminar donde la actividad será borrada. - Presionar el botón de "más" para poder crear una nueva actividad.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de "documentos" 3. Presionar en el nombre de algún alumno 4. Presionar en el botón de "más" 5. Rellenar todos los campos del cuadro de diálogo 6. Presionar en "Crear" B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de "documentos" 3. Presionar en el nombre de algún alumno 4. Presionar en algún documento que tenga el alumno 5. Presionar el botón de "Descargar" C: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de "documentos" 3. Presionar en el nombre de algún alumno 4. Presionar en algún documento que tenga el alumno 5. Presionar el botón de "editar" 6. Cambiar los datos del documento 7. Presionar en el botón de "Actualizar" B: 6. Estar autenticado en el sistema 7. Presionar en el menú el icono de "documentos" 8. Presionar en el nombre de algún alumno 9. Presionar en algún documento que tenga el alumno 10. Presionar el botón de "Eliminar"

Postcondición	- Una vez presionado en el botón de “Crear” se enviará una petición a la base de datos que creará el nuevo documento y podrá ser visualizada tanto por el profesor como por el alumno. - Una vez presionado en el botón de “Descargar” se enviará una petición a la base de datos que descargará el archivo asociado al documento en el dispositivo del usuario. - Una vez presionado el botón de “Actualizar” se enviará una petición a la base de datos que actualizará el documento seleccionado y se mostrará el documento actualizado tanto al profesor como al alumno. - Una vez presionado en el botón de “Eliminar” se enviará una petición a la base de datos que eliminará el documento seleccionado haciendo que esta desaparezca del listado de clases tanto del alumno como del profesor.
Excepciones	- Formato erróneo (En cualquiera de los campos). - Campos requeridos no rellenados. - La referencia al archivo asociado ha sido perdida y no se puede descargar el archivo.

Tabla 13 Caso de uso de profesor: administración de documentos

Nombre	Administrar avisos.
Descripción	Un profesor podrá acceder a la vista para administrar los avisos y se le mostrará un listado con todos los avisos actualmente activos, además: - Seleccionando algún aviso se le abrirá un cuadro diálogo donde aparecerán detalles del aviso y podrá presionar en el botón de editar para poder editar los datos del aviso y el botón de eliminar donde el aviso será borrado. - Presionar el botón de “más” para poder crear un nuevo aviso.
Secuencia de uso	A: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “avisos” 3. Presionar en el botón de “más” 4. Rellenar todos los campos del cuadro de diálogo 5. Presionar en “Crear” B: 1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “avisos” 3. Presionar en el nombre de algún aviso 4. Presionar el botón de “editar” 5. Cambiar los datos del aviso 6. Presionar en el botón de “Actualizar” C:

	1. Estar autenticado en el sistema 2. Presionar en el menú el icono de “avisos” 3. Presionar en el nombre de algún aviso 4. Presionar el botón de “Eliminar”
Postcondición	- Una vez presionado en el botón de “Crear” se enviará una petición a la base de datos que creará el nuevo aviso y podrá ser visualizado por todos los usuarios de la aplicación. - Una vez presionado el botón de “Actualizar” se enviará una petición a la base de datos que actualizará el aviso seleccionado y se mostrará el aviso actualizado a todos los usuarios. - Una vez presionado en el botón de “Eliminar” se enviará una petición a la base de datos que eliminará el *aviso* seleccionado haciendo que esta desaparezca del listado de avisos para todos los usuarios de la aplicación.
Excepciones	- Formato erróneo (En cualquiera de los campos). - Campos requeridos no rellenados.

Tabla 14 Caso de uso de profesor: administración de avisos

5.3 Modelo de dominio

En esta sección se presenta el diagrama de clases, el cual es una representación visual de las entidades y las relaciones que existen en el dominio de la aplicación. Este diagrama proporciona una visión clara y estructurada de cómo se organizan y se relacionan las diferentes clases del sistema. A través del modelo de dominio, se puede comprender rápidamente la estructura y el comportamiento de la aplicación, facilitando así el diseño y la implementación.

Cabe destacar que estas clases pueden sufrir cambios a medida que el proyecto se vaya realizando ya que se encuentra en la fase de análisis y no en la de solución.

En la ilustración 7 se puede ver el diagrama de clases desarrollado.

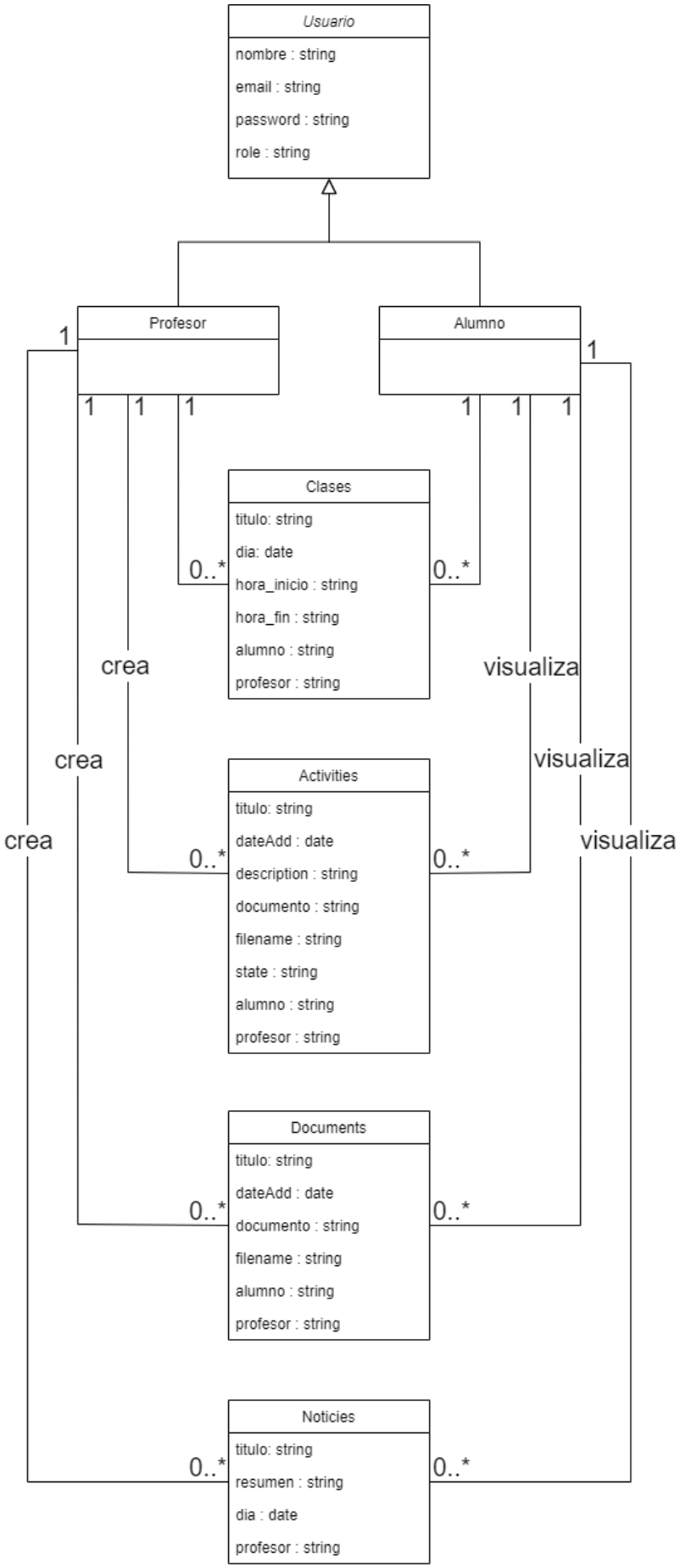

Ilustración 7 Diagrama de clases del análisis

5.4 Plan de trabajo

El plan de trabajo es una parte fundamental en la gestión de proyectos, ya que establece las actividades, plazos y recursos necesarios para su desarrollo exitoso. En este apartado, se presentará el plan de trabajo detallado para el proyecto, ofreciendo una visión clara y estructurada de las tareas a realizar.

En la realización de este proyecto, se ha utilizado la metodología de Proceso Unificado [15], que es un enfoque de desarrollo de software iterativo e incremental. Esta metodología se compone de diferentes fases que se suceden de manera secuencial, incluyendo el Inicio, Elaboración, Construcción, y Transición.

Sin embargo, se ha añadido una nueva fase previa a todas las demás, denominada "Adquisición de conocimientos". Esta fase se ha incorporado con el propósito de dedicar tiempo y recursos a la investigación, el estudio y la comprensión de las tecnologías, herramientas y conceptos necesarios para el desarrollo del proyecto. Esta fase inicial permite adquirir el conocimiento necesario antes de comenzar con las fases específicas del Proceso Unificado.

Una vez completada la fase de adquisición de conocimientos se pasará a la fase de inicio donde se establece el alcance del proyecto y se definen los objetivos.

El final de esta fase dará como resultado un documento donde aparezcan los requisitos y objetivos, los cuales no estarán expuestos a grandes cambios a lo largo del proyecto.

A continuación, comienza la fase de elaboración donde se realiza un análisis detallado de los requisitos, se elabora una arquitectura preliminar y se realiza un plan de desarrollo detallado. También se identifican los riesgos y se establece una base sólida para el proyecto.

La fase de construcción es una de las que más tiempo se requiere para completarla, debido a que se centra en la implementación de todos los requisitos descritos anteriormente.

En esta fase se desarrolla el sistema en incrementos iterativos. Se lleva a cabo la implementación, las pruebas y la integración de los componentes. Se realiza una validación continua y se van obteniendo versiones del sistema cada vez más funcionales.

Por último, tenemos la fase de Transición donde tenemos el producto completamente finalizado y funcional. Por lo tanto, en esta última fase se llevan a cabo las pruebas finales, se realiza la formación del usuario y se realiza la transición del sistema al entorno de producción. Se realiza un seguimiento del sistema en su etapa inicial de producción.

5.4.1 Diagrama de Gantt

El diagrama de Gantt [16] es una herramienta de gestión de proyectos que permite visualizar las tareas, plazos y dependencias de un proyecto en forma de gráfico de barras. Es una representación visual útil para planificar y programar las actividades a lo largo del tiempo.

El diagrama de Gantt muestra las diferentes tareas del proyecto en el eje horizontal, mientras que el eje vertical representa el tiempo. Cada tarea se representa como una barra en el gráfico, donde la longitud de la barra indica la duración de la tarea. Se pueden establecer dependencias entre las tareas, indicando qué tareas deben completarse antes de comenzar otras.

En la Ilustración 8 se muestra una de las imágenes del diagrama de Gantt de este proyecto. En la imagen se muestra la fase de 'Adquisición de conocimientos' e 'Inicio' así como las tareas para cada una de las fases.

Para más información, el diagrama se encuentra completo en el Anexo I.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Adquisición de conocimientos	**9 días**	**vie 03/02/23**	**mié 15/02/23**	
Lenguaje Dart	4 días	vie 03/02/23	mié 08/02/23	
Framework Flutter	3 días	jue 09/02/23	lun 13/02/23	2
Base de datos Firebase	2 días	mar 14/02/23	mié 15/02/23	3
Inicio	**6,5 días**	**jue 16/02/23**	**vie 24/02/23**	**4**
Definir el alcande del proyecto	**1 día**	**jue 16/02/23**	**jue 16/02/23**	
Identificar los objetivos del proyecto	0,5 días	jue 16/02/23	jue 16/02/23	
Establecer los límites del proyecto	0,5 días	jue 16/02/23	jue 16/02/23	7
Investigación preliminar	**1,5 días**	**jue 16/02/23**	**vie 17/02/23**	
Realizar un estudio de viabilidad técnica	0,5 días	vie 17/02/23	vie 17/02/23	8
Recopilar información sobre tecnologías y herramientas relevantes	0,5 días	jue 16/02/23	jue 16/02/23	
Analizar trabajos relacionados y proyectos similares	0,5 días	jue 16/02/23	jue 16/02/23	
Identificar los interesados	**1,5 días**	**vie 17/02/23**	**lun 20/02/23**	**10**
Identificar los posibles interesados	0,5 días	vie 17/02/23	vie 17/02/23	
Establecer comunicación y acordar las expectativas y necesidades	1 día	lun 20/02/23	lun 20/02/23	14
Definir los requisitos iniciales	**2,5 días**	**mar 21/02/23**	**jue 23/02/23**	
Documentar los requisitos iniciales del sistema	2 días	mar 21/02/23	mié 22/02/23	15
Priorizar los requisitos según su importancia y viabilidad	0,5 días	jue 23/02/23	jue 23/02/23	17
Elaborar el plan preliminar	**1 día**	**jue 23/02/23**	**vie 24/02/23**	**16**
Establecer un cronograma tentativo para cada fase del proyecto.	0,5 días	jue 23/02/23	jue 23/02/23	
Identificar los recursos necesarios, como hardware, software y datos	0,5 días	vie 24/02/23	vie 24/02/23	20

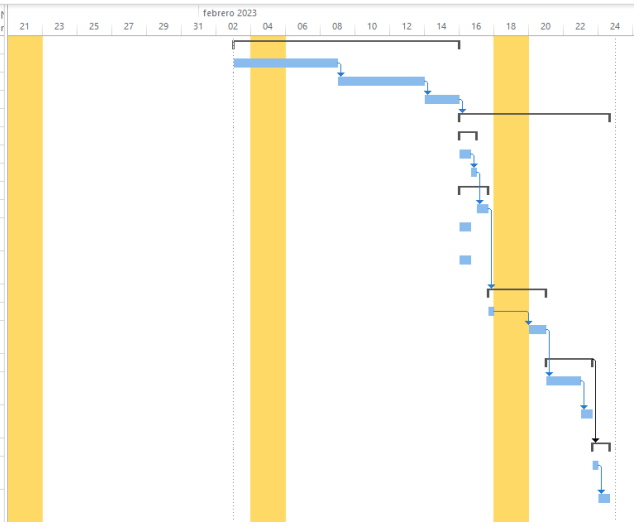

Ilustración 8 Diagrama de Gantt

5.4.2 Estimación del esfuerzo

Una vez que se ha realizado el diagrama de Gantt indicando todas y cada una de las tareas que se han considerado relevantes y fundamentales para la entrega del proyecto, se ha obtenido como resultado que el proyecto tendrá una duración aproximada de 452 horas, como se puede apreciar en la Ilustración 9.

Si bien es cierto que las estimaciones a la hora de medir cuanto tiempo se tardará en finalizar un proyecto software suelen ser erróneas y hay que estar preparado para los contratiempos que puedan surgir mientras se desarrolla el proyecto, los errores en las mediciones de tiempo suelen ser causadas por una mala planificación inicial. En la mayoría de planificaciones no se le da mucha importancia a reservar un conjunto del tiempo total a posibles contratiempos los cuales suelen ser la principal causa de retraso en un proyecto.

Por lo tanto, en el diagrama de Gantt se dejará reservado un conjunto de horas de trabajo que serán utilizadas únicamente en los momentos que surja algún imprevisto no contemplado que ocasione un retraso importante en el proyecto. Con esto se conseguirá un intervalo de tiempo donde la fecha para la finalización del proyecto estará entre la fecha optimista, la cual supondría no haber utilizado ninguna de estas horas reservadas para contratiempos y la fecha pesimista, que será el peor de los casos donde se haga uso de todas las horas reservadas para posibles problemas.

Con este sistema se asegurará que la fecha de finalización del proyecto siempre estará entre estos dos casos consiguiendo un intervalo donde se estará seguros que finalizará el proyecto.

Cabe también destacar que la experiencia juega un papel muy importante a la hora de poder asignar tiempos a tareas y esta será la que a la larga otorgará la capacidad de refinar la asignación de tiempos a las tareas siendo lo más precisa posible.

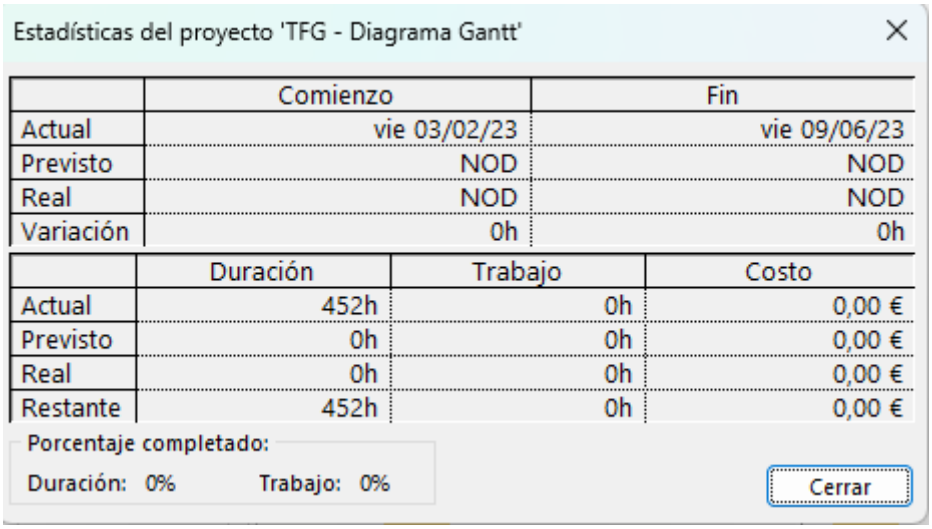

Ilustración 9 Estimación del esfuerzo

6. Diseño de la solución

En este apartado se aborda la fase crucial antes de la implementación en la cual se define la arquitectura y estructura de la aplicación. En esta sección se presentará una visión general de la solución propuesta, enfocándose en los aspectos clave del diseño. Se detallarán las decisiones arquitectónicas, la elección de tecnologías, la organización de los componentes y la interacción entre ellos a partir del análisis realizado en el capítulo anterior.

6.1 Diseño arquitectónico

En cuanto al diseño arquitectónico, se ha optado por la estructura de Cliente-Servidor [17] (Ilustración 10). Esta elección se basa en la necesidad de separar las responsabilidades entre el cliente, que es la aplicación en Flutter, y el servidor, representado por Google Firebase.

La arquitectura Cliente-Servidor se caracteriza por tener un cliente ligero que se encarga principalmente de la interfaz de usuario y la presentación de datos, mientras que el servidor se encarga del procesamiento y almacenamiento de datos. Una de las ventajas de esta estructura es la capacidad de escalabilidad, ya que permite gestionar un mayor número de clientes a través de un servidor centralizado. Además, facilita la actualización y mantenimiento del sistema, ya que los cambios realizados en el servidor pueden propagarse a todos los clientes de manera más sencilla.

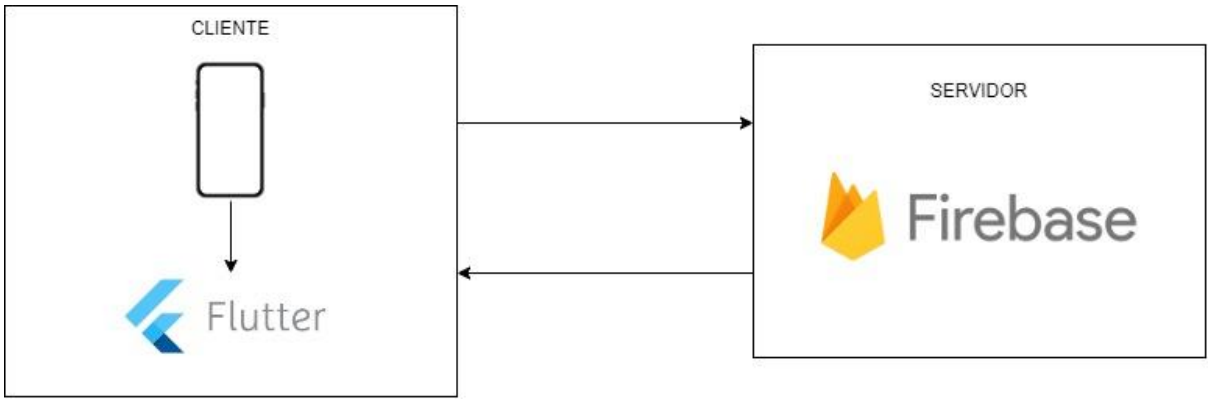

Ilustración 10 Diseño arquitectónico

6.2 Vista de arquitectura

Como se ha visto, se va a usar la arquitectura de Client-Servidor para la construcción del sistema. Además, como se ha comentado en apartados anteriores, se va a hacer uso de la base de datos Firebase, por lo tanto, estos dos elementos unidos formarán nuestra arquitectura como se puede apreciar en la Ilustración 11.

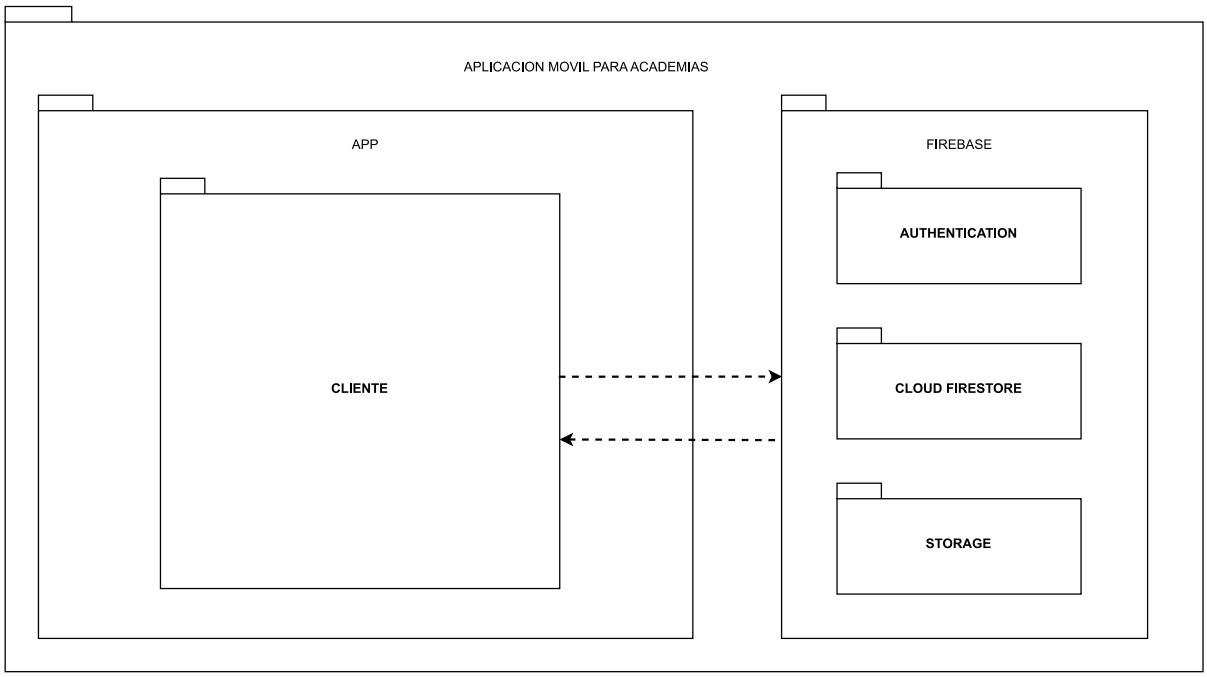

Ilustración 11 Vista de arquitectura del diseño

6.3 Diagrama de clases del diseño

En este apartado se presenta una representación visual de la estructura y las relaciones entre las distintas clases que componen la aplicación. Este diagrama proporciona una visión clara y concisa de cómo se organiza y relaciona cada componente del sistema. A través de este diagrama, se pueden identificar las clases principales, sus atributos y métodos, así como las asociaciones y dependencias entre ellas. Es una representación cercana y fidedigna al sistema final.

En la Ilustración 8 se puede ver un ejemplo de este diagrama, la ‘Gestión de sesión’, donde se podrá ver todas las clases que interactúan con la gestión de sesión y cómo todas ellas se enmarcan dentro de la arquitectura de Cliente-Servidor.

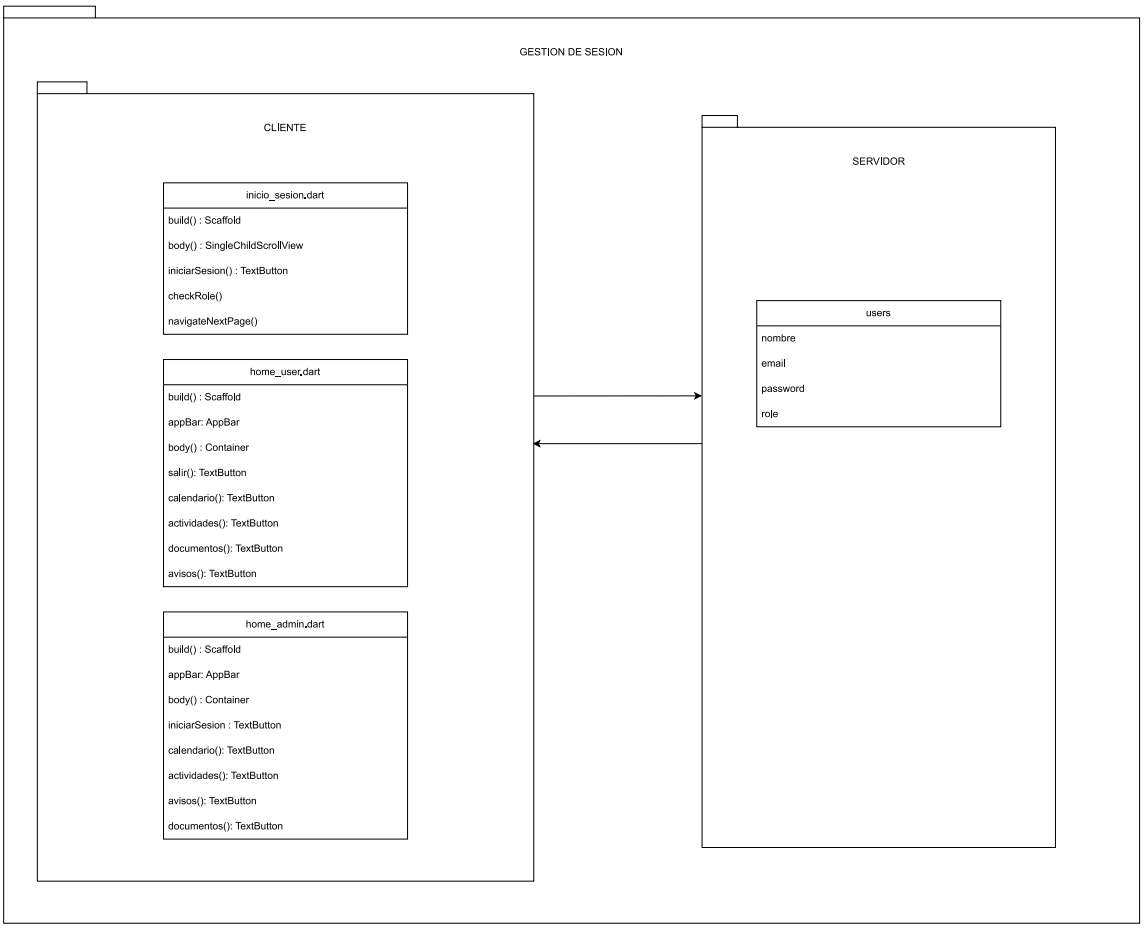

Ilustración 12 Diagrama de clases - Gestión de sesión

6.4 Diseño de datos

Como se ha comentado en apartados anteriores la base de datos escogida para guardar los datos de la aplicación ha sido Google Firebase Firestore. Ésta base de datos es de tipo NoSQL y se basa en un modelo de documentos, la cual la hace fácilmente escalabre y puede manejar grandes volúmenes de datos.

En la ilustración 13 se puede observar las distintas colecciones usadas para guardar la información de la aplicación.

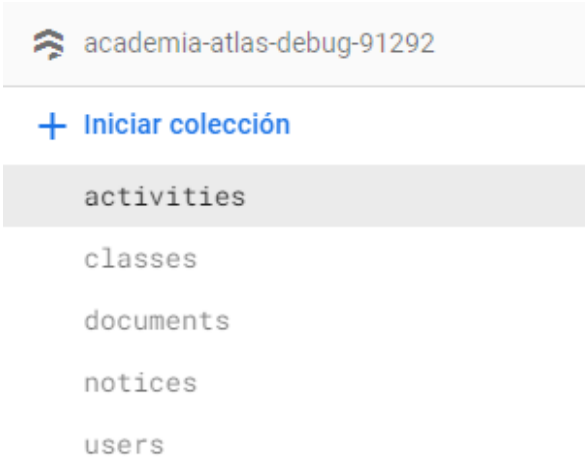

Ilustración 13 Diseño de la base de datos

En la ilustración 14 se puede observar la colección de datos de “activities” la cual guardará la información de todas las actividades creadas en la aplicación.

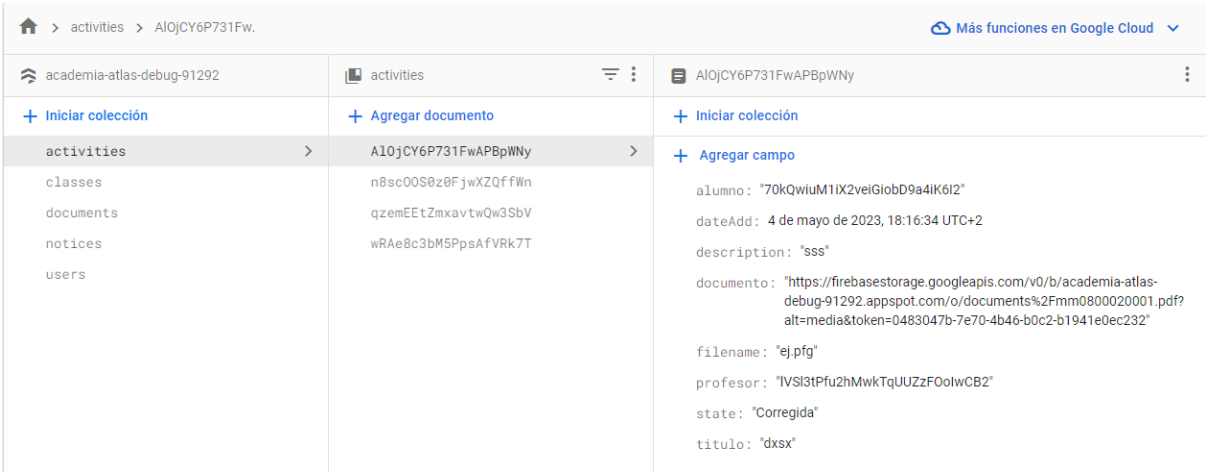

Ilustración 14 Diseño de la base de datos - Documento actividades

También se ha utilizado una funcionalidad de Google Firebase conocida como “storage”. Esta permite el almacenamiento y la gestión de archivos en la nube de forma sencilla y segura. Al utilizar el servicio de almacenamiento de Firebase, se ha facilitado el manejo de archivos multimedia, como imágenes, vídeos o documentos, dentro de la aplicación.

En la ilustración 15 se puede ver cómo hay diferentes documentos guardados en el almacenamiento proporcionado por Google Firebase. Cada uno de ellos tiene su propia URL mediante la cual se puede descargar o visualizar el documento que tiene asociado.

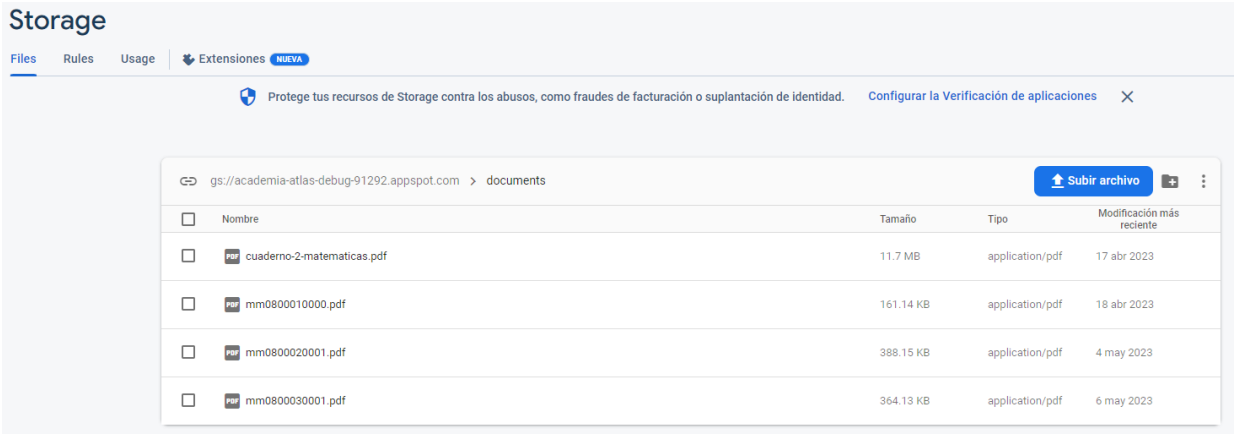

Ilustración 15 Google Firebase: Storage

6.5 Modelo de despliegue

El modelo de despliegue se refiere a la configuración y distribución de los componentes y recursos de software en un entorno de producción. En este apartado, se presentará el modelo de despliegue utilizado para implementar la aplicación, destacando la estructura física y lógica de los elementos involucrados.

A un lado se tiene el dispositivo móvil usado para poder ejecutar la aplicación, mientras que en el otro unido mediante la conexión a internet se encuentra la plataforma de Google Firebase.

En la Ilustración 16 se puede ver el modelo de despliegue planteado.

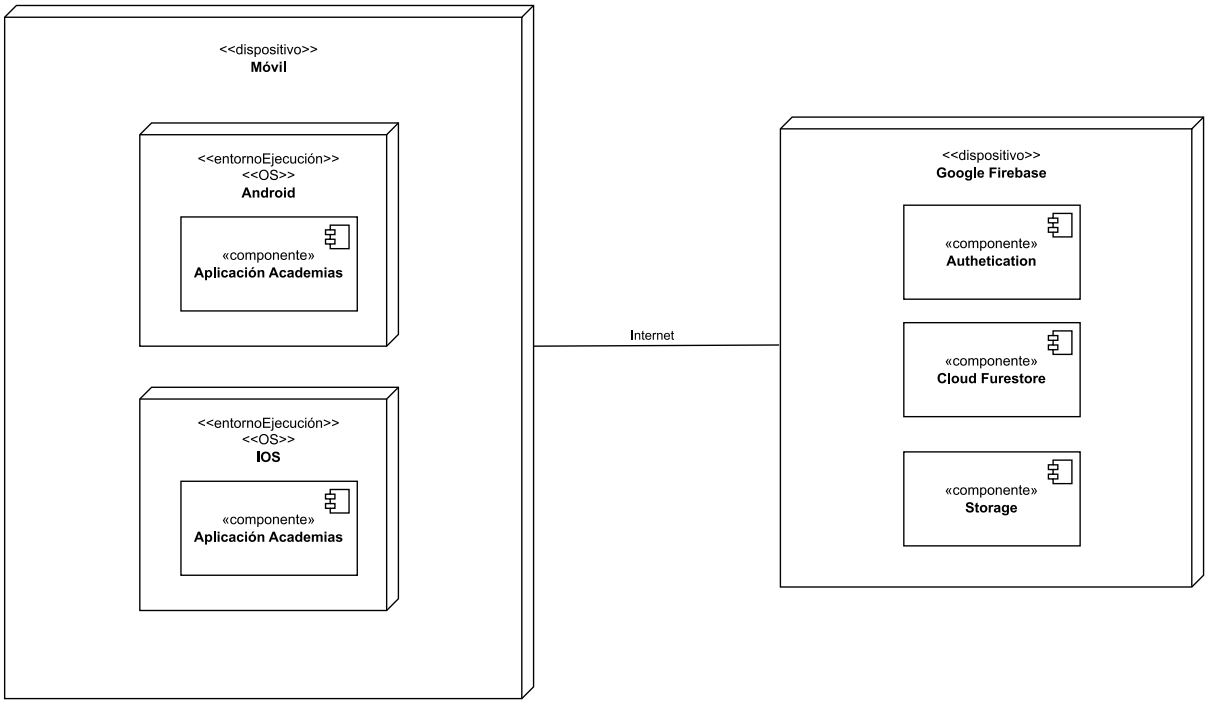

Ilustración 16 Modelo de despliegue

6.6 Diseño de interfaz

Antes de comenzar el desarrollo del código, es fundamental realizar un diseño detallado de la interfaz, ya que esto permite definir la estructura, la disposición de elementos y la experiencia de usuario deseada. Gracias a realizar un diseño previo a la implementación se puede visualizar y probar diferentes opciones de diseño, detectar posibles problemas de usabilidad y el desarrollo de código será mucho más liviano si se parte con una idea de lo que se tiene que desarrollar y cómo se debería ver la aplicación final.

En este caso se ha hecho uso de la plataforma de diseño Figma [18] para crear el diseño de interfaz. Figma es una herramienta colaborativa basada en la nube que permite crear, compartir y colaborar en diseños de interfaz de manera eficiente. Ofrece una amplia gama de funcionalidades, como la creación de diseños interactivos, la colaboración en tiempo real y la facilidad para compartir y recibir comentarios. El uso de Figma ha facilitado el proceso de diseño de interfaz, permitiendo visualizar el diseño de manera efectiva antes de proceder a la implementación.

En la ilustración 17 se puede ver el diseño planteado para la aplicación, donde estará tanto la parte de usuario (alumno) como de administrador (profesor).

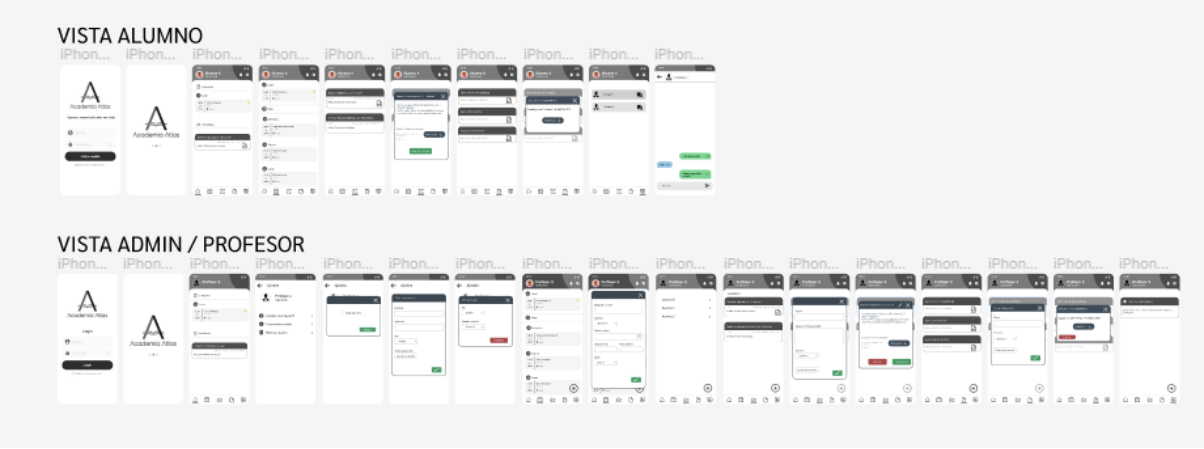

Ilustración 17 Diseño de la aplicación

7. Desarrollo de la solución propuesta

En este apartado se describirán aspectos relevantes del desarrollo de la aplicación, siguiendo el diseño previamente expuesto. Aquí se detallarán todas las vistas implementadas en la aplicación, mostrando cómo se ha llevado a cabo la materialización de la solución diseñada.

7.1 Estructura de paquetes

La estructura de paquetes (Ilustración 18) que se ha llevado a cabo en el código fuente se ha dividido en función del tipo de usuario que interactúa con el sistema. Por lo tanto, se puede apreciar cómo en la carpeta "pages" existen 3 carpetas; la carpeta de "admin" que contendrá todas las vistas y funcionalidades que interactúen con el usuario tipo administrador, "user" que interactuará con el tipo de usuario alumno y por último una carpeta "common" que será utilizada para guardar en ella funcionalidades que sean usadas por ambos tipos de usuario.

También se encuentran las carpetas "style" que guardará todo lo referente al estilo usado en la aplicación y la carpeta "querys" donde se guardarán las funciones que interactúen con el servidor de base de datos.

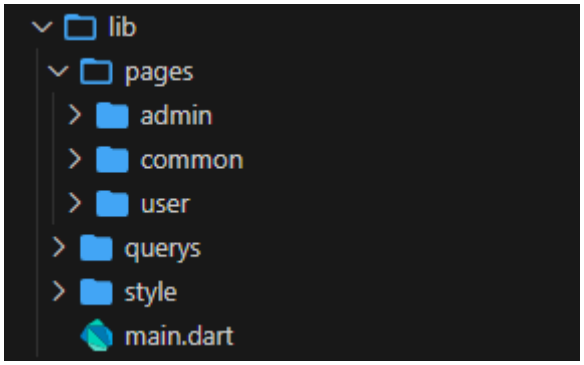

Ilustración 18 Estructura de paquetes

7.2 Paquetes usados en la aplicación

Uno de los archivos más importantes en cualquier aplicación Flutter es el "pubspec.yaml". El archivo pubspec.yaml en Flutter es un archivo de configuración fundamental que se utiliza para gestionar las dependencias y configuraciones del proyecto. Permite especificar las bibliotecas de terceros o paquetes que se utilizarán en la aplicación, así como sus versiones correspondientes.

Los paquetes pueden ser obtenidos desde la página pub.dev, como se muestra en la Ilustración 19, en la cual se encontrará un gran número, muchos de los cuales desarrollados por la comunidad, y con los cuales se podrá dotar de diversas funcionalidades al código.

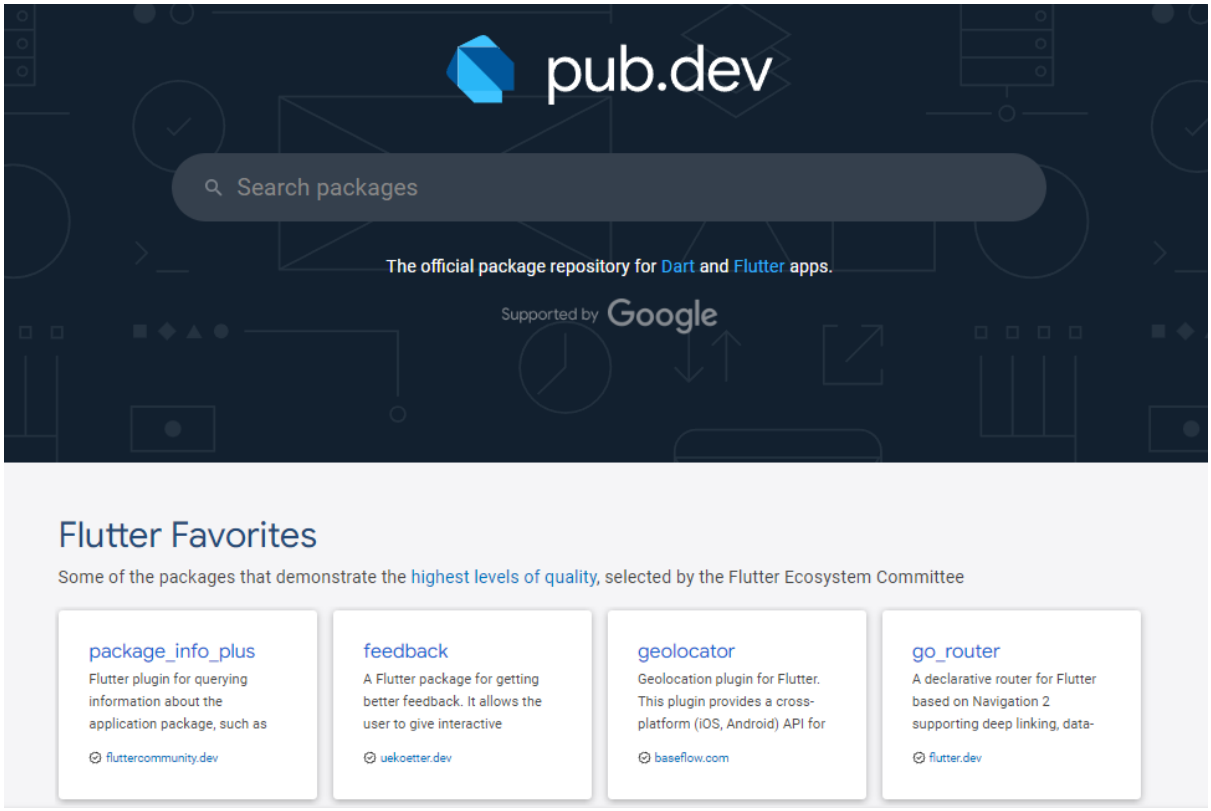

Ilustración 19 Paquetes disponibles en pub.dev

Los paquetes usados para esta aplicación se pueden encontrar en el siguiente fragmento de código:

```
firebase_core: ^2.7.1
firebase_auth: ^4.2.10
cloud_firestore: ^4.4.5
material_design_icons_flutter: ^6.0.7096
line_awesome_flutter: ^2.0.0
intl: ^0.17.0
icons_flutter: ^0.0.4
file_picker: ^5.2.9
firebase_storage: ^11.1.0
flutter_downloader: ^1.10.2
dio: ^5.1.2
path_provider: ^2.0.15
open_file: ^3.3.1
driver_extensions: ^2.0.0-nullsafety.1
```

Cada una de las líneas de código mostradas en la figura anterior consta del nombre del paquete seguido de dos puntos y la versión del paquete instalada en la aplicación. Cada paquete ha aportado funcionalidades distintas como: permitir la comunicación con el servicio de base de datos Google Firebase (*firebase_core, firebase_auth, firebase_storage, cloud_firestore*), ayudar con el diseño consiguiendo una interfaz de usuario más amigable (*material_design_icons_flutter,*

line_awesome_flutter, icons_flutter) o ayudar con la gestión de documentos (*file_picker, flutter_downloader, path_provider, open_file*).

7.3 Funcionalidades de la aplicación

En este apartado veremos todas las funcionalidades de la aplicación para la gestión de Academias.

Tiene dos funcionalidades principales: la primera enfocada a los profesores que actuarán con el rol de *admin*, llevarán la gestión de usuarios y crearán horarios, actividades, documentos y avisos; por otra parte, tenemos a los alumnos que podrán visualizar todos los horarios, actividades, documentos y avisos que se les han asignado y podrán interactuar con ellos.

7.3.1 Inicio de sesión

La aplicación cuenta con un sistema de inicio sesión basado en usuario y contraseña. Estos serán dados por parte de los profesores a los alumnos. En la Ilustración 20 se puede observar el inicio de sesión creado para la aplicación.

Ilustración 20 Inicio de sesión

7.3.2 Pantalla principal

La pantalla principal de la aplicación (Ilustración 21) será la primera vista que accederán tanto los alumnos como los profesores al iniciar sesión, en ambos casos serán idénticas. La pantalla muestra la próxima clase a la que el alumno o el profesor deban asistir, la última clase, en el caso del alumno, que le haya sido asignada y en el caso del profesor que haya asignado. También se mostrará la última alerta que algún profesor haya publicado en el sistema.

Por otra parte, se puede observar cómo hay un encabezado al inicio de la vista que se repetirá en todas las ventanas principales donde se mostrará la imagen del perfil del usuario además de su nombre. En el caso de ser un profesor al pulsar en la imagen se abrirá una nueva ventana de ajustes donde podrá realizar diferentes acciones, en el caso del alumno no tendrá ninguna interacción.

También la aplicación cuenta con una barra de navegación que ha sido implementada en todas las vistas principales, a través de ella se podrá navegar entre los módulos principales de la aplicación que son: Inicio, Calendario, Actividades, Documentos y Avisos.

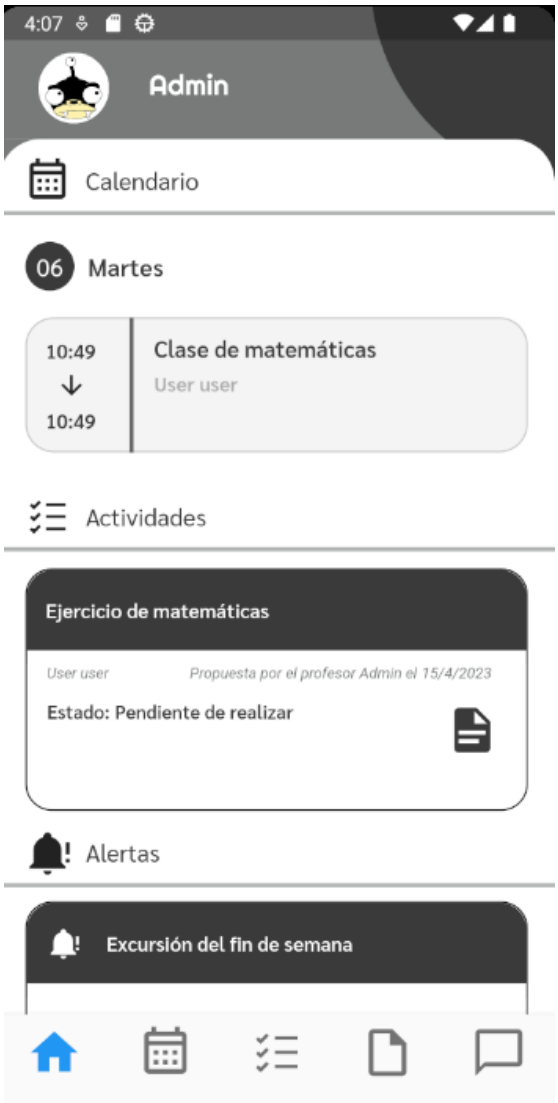

Ilustración 21 Pantalla principal

7.3.3 Pantalla de ajustes

Como se ha comentado en el punto anterior, al ser un usuario de tipo administrador y pulsar en la foto de perfil se abrirá una nueva ventana de ajustes (Ilustración 22). En la ventana de ajustes se podrán realizar 3 acciones:

- Crear un nuevo usuario (Ilustración 23)
- Eliminar usuario (Ilustración 24)
- Cambiar la foto de perfil de un usuario (Ilustración 25)

Todos los campos de los formularios cuentan con validación, si algún campo no es introducido o es introducido con datos erróneos saltará un error.

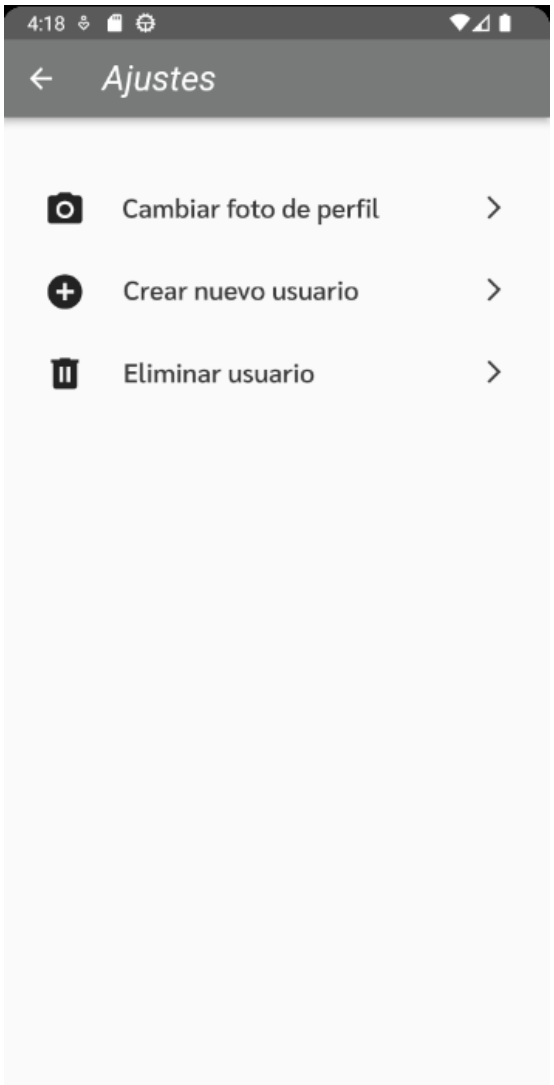

Ilustración 22 Pantalla de ajustes

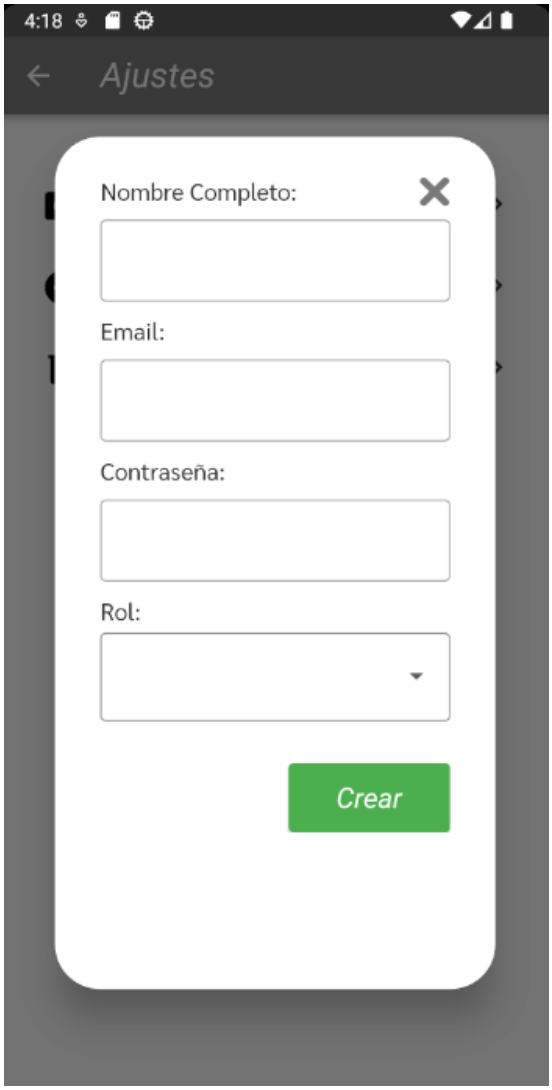

Ilustración 23 Pantalla de ajustes - Creación de usuario

Ilustración 24 Pantalla de ajustes - Eliminación de usuario

Ilustración 25 Pantalla de ajustes - Cambiar foto de perfil

7.3.4 Calendario

Al calendario se accederá pulsando en el icono de calendario del menú de navegación.

La vista mostrará las clases que tenga el usuario ordenadas por proximidad a la fecha del evento. Cada tipo de usuario (profesor y alumno) tendrán variaciones en la vista y su funcionalidad:

- La vista del alumno (Figura 26) simplemente mostrará las clases que le han sido asignadas mostrando: fecha y hora de la clase, titulo y el profesor que le ha asignado la clase.
- La vista del profesor (Figura 27) mostrará lo mismo que la vista del alumno, pero en este caso mostrará debajo del título de la clase el alumno con el cual tiene la clase. También si pulsa en alguna clase podrá editarla/eliminarla (Figura 28) y en la parte inferior de la vista se encuentra un botón que al pulsarlo se abrirá un cuadro de diálogo donde podrá crear una nueva clase (Figura 29).

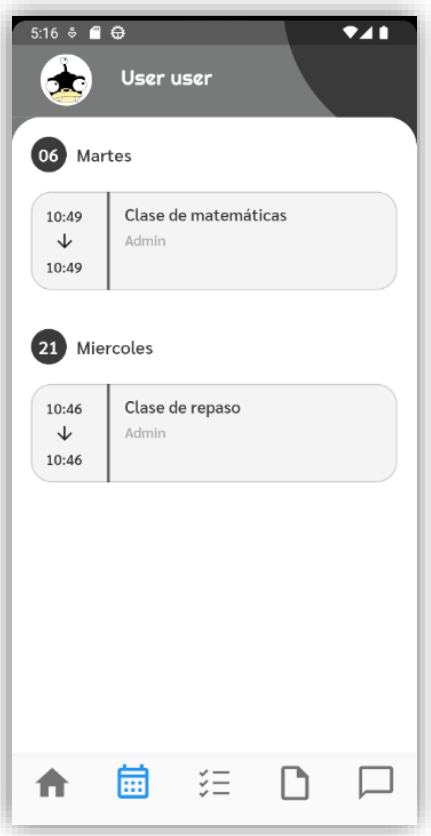

Ilustración 26 Pantalla calendario alumno

Ilustración 27 Pantalla calendario profesor

Ilustración 28 Pantalla calendario - edición de clases

Ilustración 29 Pantalla calendario - creación de clases

7.3.5 Actividades

La pantalla de actividades será donde los alumnos puedan ver las actividades o ejercicios que el profesor les ha mandado.

Al pulsar en el menú de navegación sobre el icono de Actividades dependiendo del tipo de usuario tendremos dos casos:

- Al acceder a la vista de actividades siendo un profesor, se abrirá una vista donde se mostrará una lista de todos los alumnos registrados en la aplicación (Ilustración 30) además al final de la vista se encuentra un botón mediante el cual podremos crear una nueva actividad. Al pulsar sobre algún alumno se abrirá una nueva ventana donde se le mostrará al profesor todas las actividades que el alumno tiene asignadas (Ilustración 31).
- Si por el contrario es un alumno el que accede a las actividades, se le mostrará directamente todas las actividades que le han sido asignadas (Ilustración 34).

Cada actividad que se muestra contiene el título, el alumno al que le ha sido asignada la actividad, el profesor que la creó y la fecha y el estado de la actividad. Si una actividad además tiene asignada un documento, se podrá apreciar el icono de un documento en la tarjeta de la actividad. Al presionar en la actividad se abrirá un cuadro de diálogo que tiene funcionalidades diferentes entre los dos tipos de usuario:

- Si un alumno presiona una actividad se le abrirá un cuadro de diálogo donde podrá ver los detalles de la actividad (Figura 35) además de poder marcar la actividad como “Realizada”. Si la actividad tiene un documento asociado el alumno pulsando en el botón de descargar podrá descargarse el documento en su almacenamiento local.
- Si un profesor pulsa en una tarea tendrá las mismas funcionalidades que un alumno, pero podrá cambiar el estado de la actividad a “Pendiente” o “Corregida” (Figura 32). También dispone de dos botones adicionales: uno le permitirá eliminar la tarea mientras que el botón de editar le abrirá un cuadro de diálogo donde podrá editar la tarea (Figura 33).

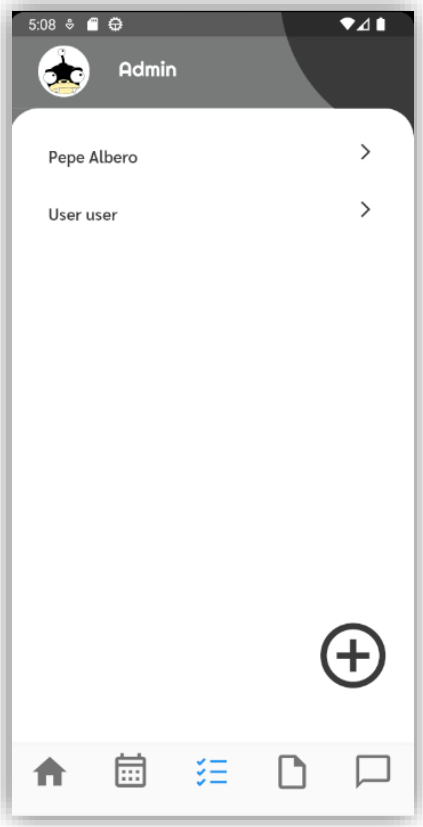

Ilustración 30 Pantalla de actividades profesor - selección de alumno

Ilustración 31 Pantalla de actividades profesor - listado de actividades

Ilustración 32 Pantalla de actividades - detalles de actividad

Ilustración 33 Pantalla de actividades - edición de actividad

Ilustración 34 Pantalla de actividades alumno

Ilustración 35 Pantalla de actividades alumno - detalles de actividad

7.3.6 Documentos

La pantalla de documentos será una vista donde los profesores puedan subir documentos relacionados con la actividad docente y los alumnos podrán verlos y descargarlos.

Documentos es una vista diferente que no guarda relación con actividades, será un apartado donde se puedan subir documentos de toda clase de índole y podrán ser asignados a los alumnos.

Igual que pasa en la vista de actividades, si un alumno presiona en el menú de navegación sobre el icono de documentos se le mostrará todos los documentos que le han sido asignados (Figura 40), si presiona en algún documento se le abrirá un cuadro de diálogo donde podrá descargar el documento (Figura 41). Por otra parte, si es un usuario administrador quien accede a la vista de documentos, primero se le mostrará un listado de los alumnos (Figura 36) y una vez que seleccione a un alumno se le mostrará todos los documentos que el alumno tiene asignados (Figura 38) y presionando en alguno podrá tanto descargar el documento, editar el documento o eliminar el documento (Figura 39).

Los administradores también contarán con un botón para poder crear un nuevo documento (Figura 37).

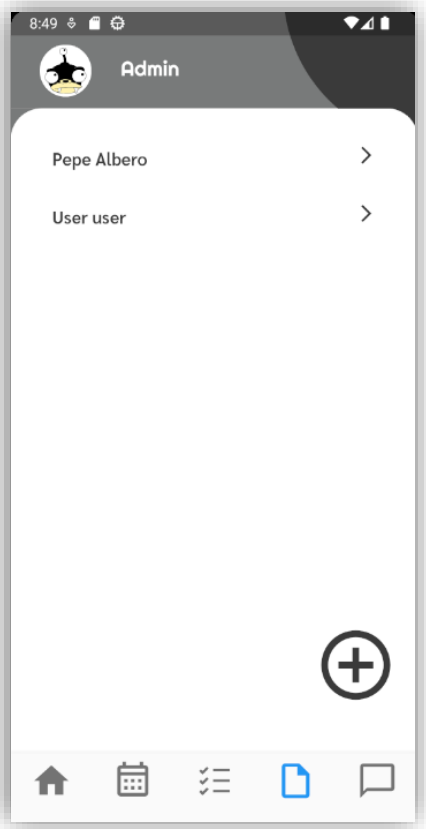

Ilustración 36 Pantalla de documentos profesor - selección de alumno

Ilustración 37 Pantalla de documentos profesor - creación de documento

Ilustración 38 Pantalla de documentos profesor - lista de documentos

Ilustración 39 Pantalla de documentos profesor - detalles de documento

Ilustración 40 Pantalla de documentos alumno - lista de documento

Ilustración 41 Pantalla de documentos alumno - detalles de documento

7.3.7 Avisos

La última de las vistas que tiene la aplicación es la vista de avisos donde los profesores podrán publicar anuncios que serán vistos por todos los usuarios de la aplicación.

Si un alumno presiona en el menú de navegación sobre el icono de avisos se le mostrará todos los avisos que han sido publicados en el sistema (Figura 46), si presiona en algún aviso se le abrirá un cuadro de diálogo donde podrá ver los detalles del aviso (Figura 47). Por otra parte, si es un usuario administrador quien accede a la vista de avisos se le mostrará todos los avisos publicados en la aplicación (Figura 42) y presionando en alguno se le abrirá un cuadro de diálogo con los detalles del aviso (Figura 44) y podrá eliminar el aviso o editar el aviso (Figura 45).

Los administradores también contarán con un botón para poder crear un nuevo aviso (Figura 43).

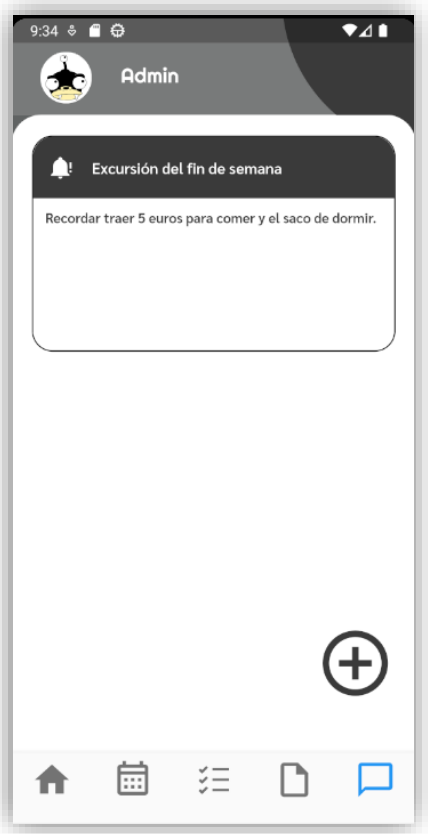

Ilustración 42 Pantalla de avisos profesor

Ilustración 43 Pantalla de avisos profesor - creación de aviso

Ilustración 44 Pantalla de avisos profesor - detalles de aviso

Ilustración 45 Pantalla de avisos profesor - edición de aviso

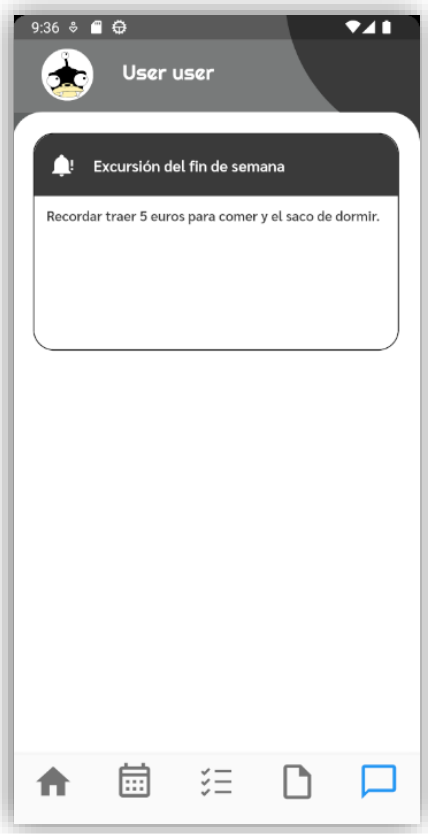

Ilustración 46 Pantalla de avisos alumno

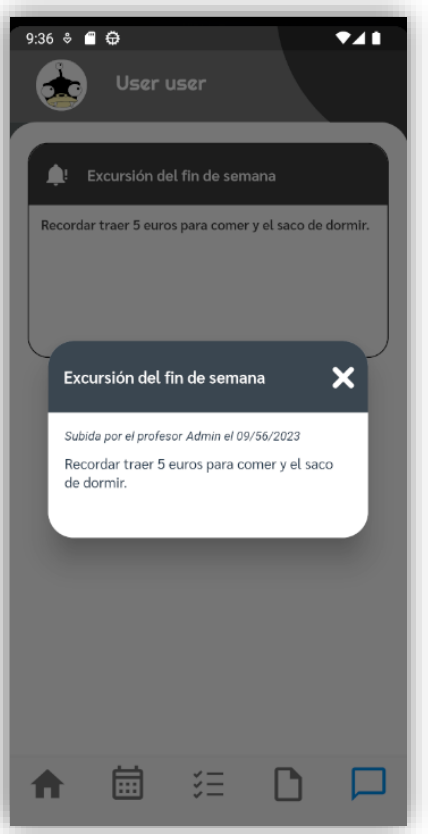

Ilustración 47 Pantalla de avisos alumno - detalles de aviso

7.4 Diseño responsive

En el contexto actual, el diseño responsive se ha vuelto fundamental en el desarrollo de aplicaciones, ya que los usuarios acceden a ellas desde una amplia variedad de dispositivos con diferentes tamaños de pantalla y resoluciones. El objetivo del diseño responsive es garantizar que la aplicación se adapte y se vea correctamente en cualquier dispositivo, brindando una experiencia de usuario consistente y atractiva.

Es crucial desarrollar aplicaciones con diseño responsive debido a varios factores. En primer lugar, permite llegar a un público más amplio, ya que los usuarios pueden acceder a la aplicación desde dispositivos móviles, tabletas y computadoras de escritorio. Además, proporciona una usabilidad mejorada, ya que los elementos de la interfaz se reorganizan y redimensionan automáticamente para adaptarse a diferentes pantallas, facilitando la navegación y la interacción.

En Las Ilustraciones 48 y 49 se pude ver dos vistas usando una resolución de pantalla mayor a la mostrada en los ejemplos anteriores.

Ilustración 48 Inicio de sesión en dispositivo tipo tablet

Ilustración 49 Calendario en dispositivo tipo tablet

8. Pruebas

El apartado de "Pruebas" es una etapa crucial en el desarrollo de una aplicación, ya que permite verificar y validar su funcionamiento de manera sistemática y exhaustiva. Durante esta fase, se realizan diferentes tipos de pruebas, como pruebas unitarias, de integración y de aceptación, con el objetivo de detectar y corregir posibles errores o fallos en la aplicación.

Las pruebas unitarias se centrarán en evaluar el correcto funcionamiento de unidades individuales de código, como funciones o métodos. Por otro lado, las pruebas de integración evaluarán la interacción entre diferentes componentes de la aplicación para asegurar que trabajen correctamente en conjunto. Por último, las pruebas de aceptación se enfocan en verificar que la aplicación cumpla con los requisitos y expectativas establecidos por los usuarios.

En el caso específico de este proyecto, se han llevado a cabo pruebas exhaustivas en todas las funcionalidades implementadas, incluyendo las vistas desarrolladas en la aplicación.

Cada vista tiene su conjunto de funcionalidades especificadas y desarrolladas, a partir de ellas se crean todos los casos posibles de uso tanto los que deberían de seguirse con normalidad como los que podrían llegar a ocasionar un problema. Todos ellos se ejecutan pensando en todas las posibles variantes de ellos y una vez que los resultados de las pruebas han sido satisfactorios podremos decir que la vista es correcta y segura para su funcionamiento en producción.

Se van a poner 3 ejemplos de cada una de las pruebas efectuadas en la aplicación:

Caso de Prueba Unitaria:

En el módulo de autenticación de la aplicación, se realiza una prueba unitaria para verificar que la función de inicio de sesión del usuario funcione correctamente. Se crean diferentes escenarios de prueba para comprobar el comportamiento de la función ante diferentes combinaciones de nombres de usuario y contraseñas. Se verifica si la función devuelve el resultado esperado al autenticar correctamente al usuario y si maneja adecuadamente los casos de autenticación fallida.

En el caso de Flutter existe un paquete que permite realizar diferentes pruebas sobre el código. En el siguiente fragmento de código se puede observar cómo se importa el paquete como “test.dart”.

Al ser una aplicación que hace uso del sistema de base de datos de Google Firebase, para poder realizar las pruebas se necesitará activar un emulador de móvil que será el que haga de intermediario para poder lanzar las peticiones a la base de datos.

Mediante código se simulará a un usuario introduciendo las credenciales e intentando iniciar sesión, una vez realizadas todas las acciones se comprobará si la página actual corresponde a la página principal del usuario lo que significaría un inicio de sesión satisfactorio o por el contrario la página actual no es la página principal lo que significaría que las credenciales no son correctas.

Para poder realizar todas estas acciones por código se necesitará dos archivos, el primero llamado app.dart el cual lanzará la aplicación en el emulador y app_test.dart el cual marcará que acciones se tienen que ejecutar en la aplicación.

Archivo app.dart:

```
import 'package:flutter_driver/driver_extension.dart';
import 'package:tfg_jmalbbel_app/main.dart' as app;

void main() {
 enableFlutterDriverExtension();
 app.main(); //Se ejectua la aplicación

}
```

Archivo app_test.dart:

```
…
 test("Credenciales correctas prueba", () async{
    await driver?.tap(emailField); //Simular presionar el campo de texto “Usuario”
    await driver?.enterText("user@academia.com"); //Simular introducir correo
    await driver?.tap(passwordField); //Simular presionar el campo de texto “Contraseña”
    await driver?.enterText("123456"); //Simular introducir contraseña
    await driver?.tap(signInButton); //Simular presionar botón de “Iniciar sesión”
    //Esperar a que acabe la comprobación de inicio de sesión
    await driver?.waitUntilNoTransientCallbacks();
    assert(homeUserPage != null); //Comprobar si la página actual corresponde a la principal

  });

 test("Credenciales incorrectas prueba", () async{
    await driver?.tap(emailField);
    await driver?.enterText("prueba@academia.com");
    await driver?.tap(passwordField);
    await driver?.enterText("123456");
    await driver?.tap(signInButton);
    await driver?.waitUntilNoTransientCallbacks();
    assert(homeUserPage != null);
  });
 });
}
```

Ejecución:

Para poder ejecutar las pruebas se tendrá que lanzar el siguiente comando en la terminal:

```
flutter drive --target=test_driver/app.dart
```

Salida:

Una vez que se acaba de ejecutar todas las pruebas la consola nos lanzará un mensaje diciendo que “todas las pruebas han sido pasadas”, como se puede apreciar en la figura 50.

```
00:01 +1: Login Test (tearDownAll)

00:01 +1: All tests passed!
```

Ilustración 50 Resultado de la ejecución de los tests

Caso de Prueba de Integración:

En el módulo de gestión de clases y actividades, se realiza una prueba de integración para verificar la comunicación entre diferentes componentes del sistema. Se simula la creación de una nueva clase, asignándole un profesor y un alumno. Luego, se comprueba si la información de la clase se almacena correctamente en la base de datos y si los alumnos asociados a la clase son actualizados adecuadamente en sus respectivas listas de clases.

En las Ilustraciones 51 y 52 se puede ver como los datos introducidos en la clase corresponde con los datos de la clase añadida a la vista de calendario.

Ilustración 51 Creación de una clase

Ilustración 52 Vista de clases

Caso de Prueba de Aceptación:

En la etapa de pruebas de aceptación, se simula un escenario completo de uso de la aplicación por parte de un usuario. Se realiza un caso de prueba en el que un alumno inicia sesión, navega por los diferentes apartados de la aplicación, visualiza las clases, actividades, documentos y avisos asignados a su perfil, y marca una actividad como completada. Se verifica si la aplicación responde correctamente a las interacciones del usuario, mostrando la información correcta y permitiendo realizar las acciones esperadas de manera fluida y sin errores.

Pruebas con usuarios reales:

El último paso en la fase de pruebas ha sido la realización de pruebas con usuarios reales. Para ello, se ha generado una versión de la aplicación en formato APK y se ha distribuido a un número reducido de usuarios seleccionados. Estos usuarios han tenido la oportunidad de trabajar con la aplicación y proporcionar comentarios y sugerencias, permitiendo identificar posibles errores, evaluar la usabilidad y recopilar información valiosa para mejorar la experiencia de usuario.

Las pruebas con usuarios son fundamentales para obtener una visión más amplia y realista del rendimiento y la calidad de la aplicación. La interacción directa con los usuarios proporciona información invaluable sobre la facilidad de uso, la intuición de la interfaz, la eficiencia de las funcionalidades y la satisfacción general del usuario.

A través de esta etapa de pruebas con usuarios, se han identificado errores o deficiencias no detectadas en las pruebas anteriores y se han realizado mejoras significativas en la aplicación. Estas pruebas han permitido afinar aún más la calidad y la experiencia del usuario, garantizando un producto final de alta calidad y adaptado a las necesidades y expectativas de los usuarios.

9. Conclusiones

Las conclusiones del trabajo se presentan como el cierre de todo el proceso de investigación, análisis, diseño y desarrollo de la aplicación. En este apartado, se van a analizar el grado de cumplimiento de los objetivos iniciales. Se analizarán las habilidades técnicas y personales que han sido necesarias para poder desarrollar este trabajo además de comentar las asignaturas cursadas a lo largo del grado que han sido necesarias para poder llevar el proyecto adelante.

Finalmente tendremos un apartado dedicado a las futuras mejoras que podrían ser añadidas a la aplicación mejorando así su funcionamiento. También se comentará los puntos que podrían haber sido mejorados, los errores cometidos a lo largo del proyecto y cómo estos podrían haber sido evitados.

9.1 Grado de cumplimiento de los objetivos

En cuanto al grado de cumplimiento de los objetivos planteados, se ha logrado satisfactoriamente alcanzar todas las metas establecidas. Los requisitos funcionales y no funcionales han sido implementados de manera efectiva, brindando al usuario final una experiencia óptima y cumpliendo con las expectativas planteadas. Se ha conseguido un sistema robusto y escalable lo que facilitará en el futuro expandir la aplicación y desarrollar nuevas funcionalidades de forma sencilla.

Recapitulando los objetivos planteados para el sistema:

- Se ha conseguido implementar un sistema de organización y asignación de horarios en formato calendario, lo que facilita la tarea organizativa de la academia haciendo que los alumnos y profesores puedan visualizar fácilmente las clases que tienen asignadas.
- Se ha implementado un sistema de asignación y seguimiento de actividades, así como de documentos facilitando la organización de los alumnos, teniendo un espacio donde pueden ver las actividades que le han sido asignadas.
- Se ha conseguido desarrollar satisfactoriamente un sistema de avisos donde los profesores podrán publicar anuncios relacionados con la tarea docente.
- El sistema desarrollado ofrece unos tiempos de latencia muy superiores a los esperados, haciendo que no haya pantallas de carga para poder acceder a los datos. Se ha prestado especial atención a la optimización de las consultas a la base de datos. En lugar de recuperar todos los datos y filtrarlos en la aplicación, se han realizado consultas específicas y precisas, lo que ha permitido obtener tiempos de respuesta más rápidos y eficientes.
- Los componentes de la aplicación han sido desarrollados siguiendo las pautas descritas por Robert C. Martín en el libro *"The Clean Code",* por lo que nuestro código es fácil de leer y entender lo que permite que pueda ser mantenido a lo largo del tiempo, además de poder realizar mejoras con facilidad.

Una de las claves a la hora de poder haber llevado este proyecto adelante ha sido la elección de las tecnologías. Flutter y Dart han permitido desarrollar la aplicación fácil y rápidamente. Gracias a la funcionalidad de *hot-reload* se han podido ver los cambios realizados en el código inmediatamente en la interfaz sin necesidad de volver a compilar el código, lo que ha permitido un flujo de trabajo mucho más rápido y eficiente. También la elección de Firebase como plataforma de base de datos ha sido un punto muy importante ya que al ser ambas tecnologías desarrolladas por Google ha permitido que la unión entre ellas se haya realizado de forma rápida y sin muchas complicaciones.

Durante el desarrollo del proyecto, se ha trabajado de manera constante y se ha seguido una metodología de desarrollo ágil, lo que ha permitido adaptarse a los cambios y realizar ajustes necesarios de manera oportuna. Asimismo, se ha contado con la participación de los usuarios en las pruebas, lo que ha brindado una retroalimentación valiosa y ha contribuido a la mejora de la aplicación.

9.2 Relación del trabajo desarrollado con los estudios cursados

Este proyecto es el resultado de aplicar los conocimientos obtenidos a lo largo del grado, de aplicar el conocimiento autodidacta conseguido gracias a las herramientas aprendidas en el grado y a la formación en prácticas de empresa que ha jugado un papel fundamental a la hora de enfrentarse a problemas.

Si nos centramos en el contenido del grado y qué asignaturas han tenido una relación directa a la hora de desarrollar este trabajo:

- Las asignaturas de “Introducción a la informática y la programación” y “programación” han sido la base con la cual partir a la hora de entender, desarrollar e interpretar código y como desarrollarlo de forma adecuada.
- La asignatura de “Estructura de datos y algoritmos” ha sido muy necesaria para profundizar y entender cómo se puede desarrollar código de manera más eficiente, realizando estructuras más complejas sin perder de vista los objetivos de optimización y eficiencia.
- Para desarrollar correctamente una estructura de base de datos han sido muy importantes asignaturas como “Base de datos” y “Sistemas de almacenamiento y procesado distribuido” con las cuales se ha obtenido el conocimiento necesario para poder elegir la mejor tecnología de base de datos y realizar una estructura de base de datos cohesionada y eficiente.
- La asignatura de “Interfaces persona computador” ha sido una de las que más relación guardan con el proyecto y una de las que más ha ayudado a que pueda ser realizado. En ella se nos presentó la tecnología de Flutter con la cual está desarrollado este proyecto además de darnos pautas sobre cómo desarrollar correctamente interfaces usables, eficientes y amigables con el usuario.
- Por último, mención especial al resto de asignaturas con las cuales se ha conseguido obtener un pensamiento crítico y una gran capacidad de solución de problemas lo cual ha permitido desarrollar este proyecto de la mejor forma posible. Además de desarrollar competencias como la responsabilidad y la toma de decisiones haciendo que cada decisión tomada en el proyecto tenga una explicación coherente y objetiva.

9.3 Trabajos futuros

Una vez finalizado el proyecto, existen mejoras que podrían ser integradas en la aplicación ampliando así sus funciones y dando una mejor experiencia al usuario.

Uno de los principales trabajos futuros será subir las aplicaciones a las tiendas de aplicaciones principales como son la Apple Store y Play Store. Esto facilitará que los usuarios descarguen las aplicaciones y sus futuras actualizaciones.

De cara a las mejoras sobre la aplicación estaría implementar un selector de paleta de colores, pudiendo así personalizar los colores de la aplicación permitiendo al usuario seleccionar el color que quiera.

También una mejora sería implementar un historial que permitiese a los usuarios ver las clases, actividades o documentos que han sido realizado y por lo tanto ya no se muestran en las interfaces de usuario.

En el lado del alumno estaría interesante que este pudiese subir en las actividades el documento de los ejercicios resueltos para que el profesor pueda descargárselo y corregirlo sin tener la necesidad de corregir los ejercicios de forma presencial.

Unas características que se tendría que agregar es un sistema de notificaciones que avisase al usuario de los cambios que se han producido en eventos relacionados con el sin tener la necesidad de acceder a la aplicación para comprobar si se han producido cambios en alguna de las funcionalidades. Es decir, si un profesor añade una nueva actividad a un alumno, la aplicación tendría que enviar una notificación que se mostrase en la barra de avisos del móvil indicando que se le ha añadido una nueva actividad para realizar.

Por último, una línea de mejora que sería de interés introducir en la aplicación sería conectar la aplicación con la plataforma de Google Calendar posibilitando así que los profesores subiendo las clases que tuviesen a su calendario personal apareciesen directamente en la aplicación.

10. Bibliografía

[1] Ender Aplicaciones *Portal de Atenea App* [en línea]. Disponible en: https://www.ender.es/app-para-academias/

[2] The Long Finger Company *Portal de Academity* [en línea]. Disponible en: https://academity.es/

[3] Aplicaciones nativas. *Abamobile* [en línea]. Disponible en: https://abamobile.com/web/que-son-aplicaciones-nativas-y-ventajas/#:~:text=Se%20llaman%20aplicaciones%20nativas%20debido,y%20App%20Store%20(iOS).

[4] Swift. *Apple* [en línea]. Disponible en: https://www.apple.com/es/swift/#:~:text=Swift%20es%20un%20intuitivo%20lenguaje,puede%20hacer%20realidad%20sus%20ideas.

[5] Android para desarrolladores. *Android* [en línea]. Disponible en: https://developer.android.com/?hl=es-419

[6] Sierra Yorman. (9 de agosto de 2019). Aplicaciones híbridas. *MediaCloud* [en línea]. Disponible en: https://blog.mdcloud.es/aplicaciones-hibridas-frameworks-ejemplos-y-ventajas/

[7] Que es flutter. *Aurestic* [en línea]. Disponible en: https://aurestic.es/que-es-flutter/

[8] Que son los widgets en Flutter. *Amazon* [en línea] Disponible en: https://aws.amazon.com/es/what-is/flutter/

[9] Pulido, Mónica. (11 de junio de 2019). Pros Y Contras De Flutter. Slashmobility [en línea]. Disponible en: https://slashmobility.com/blog/2019/06/pros-y-contras-de-flutter/

[10] Cianci, Lewis. (17 de marzo de 2020). Visual Studio Code vs Android Studio. *Codemagic* [en línea]. Disponible en: https://blog.codemagic.io/android-studio-vs-visual-studio-code/

[11] (30 de diciembre de 2021). ¿Qué es SQL y para que sirve? *Universidad Europea* [en línea] Disponible en: https://universidadeuropea.com/blog/lenguaje-programacion-sql/

[12] ¿Qué es NoSQL? *Amazon* [en línea] Disponible en: https://aws.amazon.com/es/nosql/

[13] Pandora FMS team. (29 de Septiembre de 2022). *Pandora FMS* [en línea]. NoSQL vs SQL. Disponible en: https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/

[14] Mora López, Sara. (17 de Mayo de 2020). *Digital55* [en línea]. Firebase: qué es, para qué sirve, funcionalidades y ventajas. Disponible en: https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/

[15] El proceso unificado. *Cybertesis* [en línea]. Disponible en: http://cybertesis.uach.cl/tesis/uach/2003/bmfcis718d/xhtml/TH.4.xml

[16] Martins, Julia. (12 de Septiembre de 2022). Diagrama de Gantt. *Asana* [en línea] Disponible en: https://asana.com/es/resources/gantt-chart-basics

[17] Schiaffarino, Andrés. (12 de Marzo de 2019). Arquitectura Cliente-Servidor. *Infranetworking* [en línea]. Disponible en: https://blog.infranetworking.com/modelo-cliente-servidor/

[18] Olmo, Juan Pablo. (Junio de 2020) ¿Qué es Figma y para qué sirve? *Juanpol* [en línea]. https://www.juanpol.com/que-es-figma/