UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

PlanLMS
Aplicación multiplataforma para la gestión de tareas de la plataforma Moodle

Presentado por Javier Pampliega García
en Universidad de Burgos — 4 de julio de 2025
Tutor: Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Javier Pampliega García, con DNI 71755086P, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado PlanLMS Aplicación multiplataforma para la gestión de tareas de la plataforma Moodle.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 4 de julio de 2025

Vº. Bº. del Tutor:

D. Raúl Marticorena Sánchez

Resumen

PlanLMS es una aplicación multiplataforma desarrollada con Flutter con el objetivo de facilitar la planificación de tareas a los usuarios de plataformas Moodle. Su funcionamiento es posible por la utilización de los servicios web de Moodle.

A través de una interfaz simple e intuitiva, se busca que el usuario logre planificar sus actividades por medio de un diagrama de Gantt que le permita visualizar la distribución de estas actividades en el tiempo. Por otra parte, incorpora una sección donde el usuario puede visualizar todas sus actividades de forma ordenada. Además, se incorpora otra funcionalidad que permite al usuario crear sus propias tareas asociadas a un curso.

Todas estas funcionalidades pueden ser reguladas mediante el sistema de filtrado que permite que el usuario visualice únicamente el contenido que necesite.

Descriptores

Gestión de tareas, Moodle, diagrama de Gantt, multiplataforma, Flutter, Dart.

Abstract

PlanLMS is a cross-platform application developed with Flutter, designed to make task planning easier for Moodle users. It operates by using Moodle´s web services.

Through a simple and intuitive interface, it allows the user to plan activities via a Gantt chart that visualizes their distribution over time. Likewise, it includes a section where the user can view all their activities in an organized list. Additionally, it offers a feature that lets the user create their own tasks linked to a course.

All of these functions can be managed through a filtering system that enables users to view only the content they need.

Keywords

Task management, Moodle, Gantt chart, cross-platform, Flutter, Dart.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Durante los últimos años la implementación de tecnologías en el ámbito educativo ha transformado los procedimientos de enseñanza. Las plataformas de gestión de aprendizaje (Learning Management Systems o LMS) como Moodle se han consolidado como herramientas imprescindibles en la enseñanza y en el seguimiento del progreso académico, ya que facilitan la interacción entre alumnos y docentes, organización del contenido académico, evaluación de actividades, etc. Aunque el uso de Moodle es cómodo para el alumnado, cabe recalcar que carece de algunas funcionalidades que pueden enriquecer el uso de la plataforma.

La motivación que impulsa el desarrollo de este proyecto viene de mejorar la gestión de las actividades de Moodle, para dar una visión más clara al alumnado de las tareas que debe realizar, previniendo que las labores pendientes queden desatendidas. Además, la aplicación incorpora un sistema de filtrado que proporciona un abanico de amplias posibilidades en el momento de filtrar el contenido que el usuario desee.

Por ello, en este Trabajo de Fin de Grado se propone la implementación de una aplicación multiplataforma desarrollada con Dart y Flutter, con el objetivo de complementar y mejorar la experiencia que ofrece Moodle. Para lograr esto último es necesario recurrir a los servicios web ofrecidos por Moodle, lo que permite la autenticación de usuarios, consulta de cursos, actividades a realizar o calificaciones obtenidas, entre decenas de funciones. Por otro lado, se presta especial atención al diseño de la interfaz de usuario, así como a la experiencia de usuario, mejorando la presentación y el uso de la aplicación.

Estructura del material entregado

A continuación se detallarán las estructuras de los materiales entregados. Desde la memoria y anexos, hasta la estructura de directorios y ficheros alojados en el repositorio de GitHub.

Estructura de la memoria

- **Introducción:** proporciona una visión general del contexto del proyecto, además de la estructura que lo compone.
- **Objetivos del proyecto:** describe los objetivos que se persiguen para alcanzar las metas establecidas.
- **Conceptos teóricos:** explica conceptos teóricos necesarios para la compresión total del proyecto y los objetivos que persigue.
- **Técnicas y herramientas:** describe las técnicas y herramientas empleadas para lograr el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** detalla los aspectos más destacados durante el desarrollo del proyecto, desde los inconvenientes hasta las decisiones tomadas para resolver los desafíos del proyecto.
- **Trabajos relacionados:** revisión de proyectos similares, realizando comparativas con dichos proyectos.
- **Conclusiones y líneas de trabajo futuras:** recoge las conclusiones finales del proyecto y especifica posibles mejoras a implementar en el futuro.

Estructura de los anexos

- **Plan de proyecto software:** detalla la planificación del proyecto y estudia las viabilidades económicas y legales del mismo.
- **Especificación de requisitos:** detalla los requisitos y casos de uso del proyecto.
- **Especificación de diseño:** especifica los aspectos de diseño, detallando la interfaz de usuario, diseño de datos, diagrama de clases, etc.

- **Documentación técnica de programación:** aporta documentación con la explicación de aspectos técnicos del proyecto para programadores.
- **Documentación de usuario:** aporta documentación relevante para que el usuario tenga conocimiento sobre el uso de la aplicación.

2. Objetivos del proyecto

El objetivo principal del proyecto es el desarrollo de una aplicación multiplataforma con Dart y Flutter, que permita al alumnado de distintas instituciones educativas planificar las actividades de su plataforma Moodle. Se busca potenciar la organización del usuario para incrementar su rendimiento académico, así como sus capacidades de planificación de tareas.

El proyecto se centra en uso de un diagrama de Gantt que facilita la visualización temporal de las actividades a realizar. Por otra parte, se implementa la creación de tareas con niveles de prioridad y ligadas a un curso matriculado. Estas funcionalidades pueden ser filtradas al gusto del usuario con el sistema de filtrado incorporado.

- **Visualización gráfica de actividades:** El objetivo principal es la implementación de un diagrama de Gantt interactivo que permita al usuario visualizar de forma clara y ordenada las fechas y duración de las tareas, de las actividades de Moodle. Además, el usuario podrá desplazarse a través del diagrama y podrá ver sus actividades ordenadas por cursos en un sección pareja al diagrama.
- **Creación de tareas propias:** La aplicación ofrecerá al usuario la posibilidad de definir sus propias tareas ligadas a un curso en el que se encuentre matriculado y podrá asignarles unas prioridad entre otras opciones. Todas las tareas creadas por parte del usuario se mostrarán por pantalla en función de su estado: Pendiente o Completada. Además, estarán ordenadas en función de su fecha de caducidad o su fecha de finalización.
- **Filtrado general de actividades y tareas:** Una de las funcionalidades más importantes será el filtrado de todos los eventos de ambas

funcionalidades mencionadas anteriormente. Esto será posible gracias al sistema de filtrado disponible en la pantalla inicial. Desde este, el usuario podrá filtrar por cursos, fechas, tipos de actividad y disponibilidad de fechas de apertura y cierre. Con ello se pretende que el usuario modifique el contenido de su aplicación a gusto propio y así logre mejorar su capacidad de organización.

Objetivos personales

En esta sección se establecen los objetivos personales que se pretenden superar en la realización de este proyecto:

- Aprendizaje en el desarrollo de aplicaciones, en este caso multiplataforma, para futuros proyectos.
- Mejorar el manejo de APIs.
- Entender el uso de la metodología SCRUM para su uso aplicado en proyectos reales.
- Mejorar la lógica de programación para desarrollos complejos.
- Descubrir y explorar nuevas herramientas que no había empleado con anterioridad.

Objetivos técnicos

En esta sección se especifican los objetivos técnicos que se pretenden alcanzar con la realización del proyecto:

- Uso de GitHub con GitBash para el control de versiones.
- Uso de Visual Studio Code para el desarrollo del proyecto.
- Uso de Dart y Flutter para el desarrollo de la aplicación multiplataforma.
- Uso de Postman para realizar peticiones a la API.
- Uso de AndroidEmulator para la simulación de un entorno Android.
- Uso de Figma para el prototipo de la aplicación.
- Uso de Overleaf para la realización de la documentación.

3. Conceptos teóricos

En este apartado se desarrollarán los conceptos teóricos necesarios para la comprensión total del proyecto. Se explicarán desde conceptos generales, hasta conceptos más específicos de las tecnologías empleadas.

Para entender el contexto de este proyecto, primero es necesario definir que es la gestión de tareas, así como todos los conceptos teóricos ligados a ellas.

3.1. Gestión de tareas

La gestión de tareas se define como un sistema estratégico y dinámico diseñado para la planificación y ejecución de tareas de forma eficaz, garantizando que la trayectoria de las mismas se lleve a cabo de forma ágil hasta su finalización. Cuando se trata de rutinas diarias, las tareas se completan en diversas fases, en las cuales se definen enfoques únicos. La dominancia de estas fases y la adaptación de la estrategia es la llave para desbloquear la productividad eficiente y el cumplimiento de los plazos establecidos.

Este concepto se puede confundir con la gestión de proyectos, la diferencia entre ambos se define en: la gestión de tareas se centra en la tareas individuales orientadas al ámbito profesional y laboral. Mientras que en la gestión de proyectos se centra en la ejecución exclusiva de objetivos profesionales.

Fases de la gestión de tareas

La gestión de tareas es un sistema que necesita de cuatro fases para ejecutarse de forma eficaz:

- **Etapa inicial:** se crea la tarea y se definen los objetivos de la misma.
- **Segunda fase:** se asigna a un responsable para llevar a cabo la tarea.
- **Tercera fase:** el responsable avanza con la realización de la tarea.
- **Cuarta fase:** se realiza una supervisión y control de la calidad de la tarea realizada.

Las tareas propuestas pueden ser eliminadas una vez se cumplan con los objetivos establecidos o también pueden ser abandonadas en caso de que los objetivos ya no sean necesarios. Además, la realización de las tareas se pueden realizar de forma conjunta o individual.

Pautas para ser eficiente en la gestión de tareas

Para lograr una gestión eficaz es necesario comprender todos los conceptos y herramientas, y emplearlas en el plan que se vaya a acometer. Se pueden emplear distintos pasos para lograrlo:

- En el caso de la tarea requiera de un equipo de trabajo, el plan se deberá de ajustar a los objetivos del equipo.
- Dedicar el tiempo necesario a cada tarea, estableciendo prioridades y realizando ajustes de tiempo.
- Priorizar la organización de las tareas, lo que ahorrará confusiones y centrará el trabajo en lo importante.
- Realizar un seguimiento de los progresos alcanzados a lo largo de las fases de la tarea. Esto ayudará a localizar que tarea requiere más atención.

Pasos clave en la gestión de tareas

La gestión de tareas requiere de algunos pasos críticos que actúan como papel vital en su funcionamiento.

- **Priorización:** es importante diferenciar la importancia de las tareas, una buena definición de prioridades permite que los plazos de entrega se completen con total eficacia. Unas buenas prácticas de priorización permiten reservar tiempo para la realización de otras tareas.

- **Seguimiento:** realizar un seguimiento de los objetivos y progresos de las tareas es vital para detectar la ausencia de componentes de gestión, es decir, lo que falta y lo que se necesita para la próxima tarea.
- **Programación:** este paso es muy importante, ya que permite definir las temporalidades de ejecución de las tareas y cuando deben de ser completadas para obtener los mejores resultados. Además, permite establecer un calendario robusto sobre el que trabajar.

Perfiles que hacen uso de la gestión de tareas

Cuando se trata de gestión de tareas, no existen restricciones de quien puede hacer uso de ello. Sin embargo, es recomendable que lo usen aquellos que tengan una agenda apretada y requieran de una organización rápida y eficaz. Los perfiles personas que más recurren a la gestión de tareas son:

- **Jefes de equipo:** desde jefe de proyectos hasta altos cargos de empresas. Generalmente, su agenda suele estar cargada con muchas tareas y a veces suelen colapsarse unas con otras. La gestión de tareas facilita la realización de tareas a tiempo.
- **Equipos de trabajo:** los equipos de trabajo se caracterizan por tener que realizar muchas tareas, repartidas entre los distintos miembros. Por ello, la gestión de tareas permite distribuir el trabajo a realizar entre los miembros, con el fin de agilizar el proceso de ejecución de las tareas.
- **Gestión personal:** permite a un individuo gestionar sus propias tareas que pueden ser: educativas, domésticas o profesionales. Con ello, se logra minimizar las perdidas de tiempo.

Ventajas de la gestión de tareas

- **Productividad:** la gestión de tareas potencia la productividad, ya que elimina tareas que son innecesarias de la rutina cotidiana.
- **Eficiencia:** cada tarea tendrá a su disposición los recursos necesarios para ser completada sin contratiempos. Además, la claridad en la organización de tareas, acelera el proceso de toma de decisiones.
- **Gestión del tiempo:** el hecho de planificar los acontecimientos con antelación previene perdidas de tiempo innecesarias. Por otro lado,

el tiempo sobrante puede ser empleado para la realización de otras tareas.

- **Toma de decisiones:** al dividir el trabajo en distintas tareas, facilita la toma de decisiones sobre las prioridades o tiempos de estas, dando una visión más amplia sobre las labores a realizar.
- **Reducción del estrés:** cuando se ha realizado la asignación de todas las tareas, se reduce el estrés de haber olvidado alguna tarea de vital importancia.

Herramientas de gestión de tareas

Para poder llevar a cabo una buena gestión de tareas es necesario hacer uso de distintas herramientas que nos ayuden a tener una mejor visión del trabajo que hay que completar.

- **Lista de tareas pendientes:** las tareas se estructuran en un listado que permite hacer una vista rápida de lo que hay que hacer.
- **Tableros:** las tareas se distribuyen de forma que se puedan distinguir por prioridades y ayudan a los usuarios a visualizarlas.
- **Diagramas:** sirven para representar gráficamente las tareas. Un ejemplo claro es el diagrama de Gantt, también usado en este proyecto. Este permite representar mediante líneas la fecha de inicio y fin de una tarea, así como la duración de la misma. Esto permite tener una visión clara de como se puede distribuir el tiempo para ejecutar las tareas de la forma más eficiente.
- **Calendarios:** los calendarios ayudan a ubicar las fechas de las tareas, así como a permiten añadir recordatorios o plazos personalizados.

Técnicas de gestión de tareas

- **Descartar tareas menos importantes:** aquellas tareas que no sean de vital importancia, deberán de ser descartadas para priorizar las más importantes. Una vez se ejecuten las de mayor importancia, se podrá destinar el tiempo restante a otras labores.
- **Evitar multitarea:** la realización de múltiples tareas supone que el esfuerzo será la mitad, además se producirán perdidas de tiempo. Lo

aconsejable es centrarse en la realización de un tarea y completarlas de una en una [29].

3.2. Diagrama de Gantt

Un diagrama de Gantt es una herramienta gráfica que permite representar el tiempo o duración de distintas tareas a lo largo del tiempo.

El formato más común de diagrama se basa en la enumeración de tareas de forma vertical de izquierda a derecha. Se establecen dos puntos, el comienzo de la tarea y el fin de la misma. Entre ambos puntos se traza una linea horizontal que representa la duración de cada tarea.

El diagrama de Gantt permite revisar las tareas pendientes de hacer, cuándo se esperan realizar o como dependen unas de otras.

En 1896 el ingeniero polaco, Karol Adamiecki, creó el armonograma (predecesor del diagrama de Gantt). Años más tarde, el ingeniero y consultor estadounidense, Henry Gantt, desarrolló su propia versión que unificaba a ambos gráficos, con el fin de evaluar la productividad de su fábrica [32].

Componentes de un diagrama de Gantt

Un diagrama de Gantt de forma simple se compone de una lista de tareas, una línea de tiempo horizontal y barras horizontales que representan las tareas a realizar. Todos los componentes que lo componen son:

- **Lista de tareas:** lista vertical de las tareas a realizar.
- **Linea de tiempo:** linea horizontal que muestra los días, semanas o meses en las que transcurren las tareas.
- **Bares:** marcadores horizontales que representan a las tareas con su duración y fechas de inicio y fin.
- **Fecha:** línea vertical que representa la fecha actual.

Cuándo utilizar un diagrama de Gantt

Se debe de hacer uso de un diagrama de Gantt cunado se requiera de la visualización de las tareas a realizar. Permite tener una visión centrada en el tiempo del trabajo que se debe de realizar. También, da una visión de las siguientes tareas a la realización de otras.

Un diagrama de Gantt es útil en los siguientes casos:

- **Planificación de tareas:** cuando se quieren visualizar todas las tareas, incluso las más pequeñas.
- **Programación de tareas:** cuando se quiere ver la duración de las tareas y cuando se finalizarán por completo.

Ventajas e inconvenientes de los diagramas de Gantt

El uso de este tipo de diagramas implica ciertas ventajas, pero también algunos inconvenientes. Las ventajas de su uso son:

- Representación clara de las tareas del proyecto, indicación de plazos, etc.
- Facilidad para compartir con las partes implicadas en las tareas

Algunos de los inconvenientes que presenta son:

- Cuando existen muchas tareas, pueden ser más compleja la creación de un diagrama de Gantt.
- Se debe de disponer de todas las fechas de las tareas para que puedan ser representadas [11].

3.3. API

El término *API* hace referencia a *Application Programming Interfaces*, es decir, interfaz de programación de aplicaciones. Consiste en un conjunto de definiciones y protocolos que se utilizan para integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones software mediante un conjunto de reglas. Podemos entender una *API* como un puente que facilita el intercambio de información y funcionalidades entre dos aplicaciones distintas [14].

Objetivo de una API

El objetivo principal de una *API* es facilitar el trabajo a los desarrolladores, además de tiempo y dinero. Por ejemplo, se puede implementar un servicio de pago sin la necesidad de integrar un servicio de pago desde cero.

Funcionamiento de una API clientes-servidor

La arquitectura de las *API* se basa en el modelo *cliente-servidor*. La aplicación que envía una solicitud se denomina cliente, y la que envía la respuesta se denomina servidor. Las *API* pueden funcionar de cuatro formas diferentes [2]:

- **API de SOAP:** utiliza el protocolo simple de acceso a datos. El cliente y el servidor intercambian mensajes a través de *XML*.
- **API de RPC:** se denominan llamadas a procedimientos remotos. El cliente completa un procedimiento en el servidor, y el servidor retorna el resultado al cliente.
- **API de WebSocket:** esta *API* utiliza objetos *JSON* para transmitir datos. Una ventaja de estas es la comunicación bidireccional entre aplicaciones cliente y el servidor. El servidor tiene la capacidad de enviar mensajes de devolución de llamada a los clientes.
- **API de REST:** en esta *API* el cliente envía las solicitudes al servidor como datos. El servidor hace uso de la entrada del cliente para inicializar las funciones internas y retorna los datos de salida del cliente. REST hace referencia a la transferencia de estado representacional. En esta se definen un conjunto de funciones como *GET, PUT, POST, DELETE, etc.* que los clientes pueden hacer uso de para acceder a los datos que se encuentren alojados en el servidor. Todos estos datos son intercambiados mediante *HTTP*.

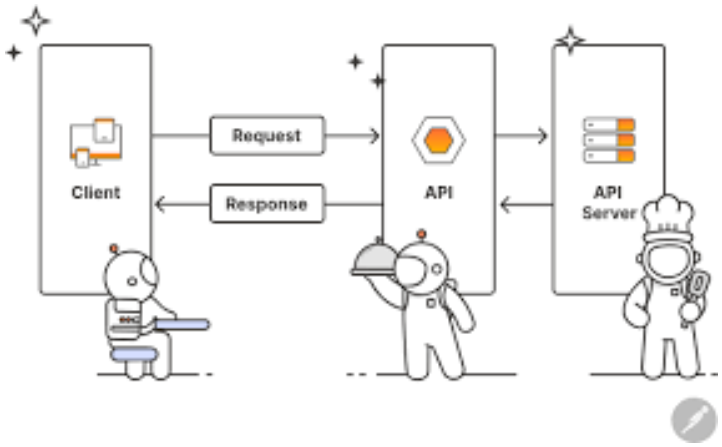

Figura 3.1: Esquema del funcionamiento de una API

3.4. APK

Los archivos con extensión *.apk* son ejecutables diseñados para el sistema móvil *Android*, este término se refiere a *Android Application Package*. Se utiliza para distribuir e instalar aplicaciones en dispositivos que hagan uso del sistema operativo Google [15]. La creación de los archivos *APK* transcurre en dos fases:

- Compilación de un programa para *Android*.
- Empaquetamiento de todas sus partes en un solo archivo.

Posteriormente el archivo *APK* contendrá todo el código, recursos, activos, certificados y archivo *manifest*.

Los distintos componentes de un archivo *APK* son [3][31]:

- **AndroidManifest.xml:** archivo de manifiesto adicional que define el nombre, versión, permisos requeridos, actividades y servicios que utiliza , versiones compatibles de Android y otros metadatos necesarios.
- **assets:** directorio que contiene archivos que pueden ser accedidos por la aplicación en tiempo de ejecución, estos pueden ser *archivos de configuración, fuentes, HTML o multimedia*. Estos archivos deben de ser accedidos a través de AssetManager.
- **classes.dex:** clases compiladas en el formato de archivo dex, legibles por la máquina virtual Dalvik.
- **lib:** contiene bibliotecas nativas compiladas y que son especificas de una capa de software de procesador. Son empleadas cuando la aplicación de depende de código en C o C++ requerido para operaciones críticas de rendimiento.
- **META-INF:** almacena el certificado, la firma, y un archivo manifiesto de la aplicación, empleado para verificar la integridad y autenticidad de la *APK*.
- **res:** directorio que contiene recursos sin compilar, como pueden ser archivos XML, imágenes, iconos y componentes de interfaz de usuario.
- **resources.arsc:** archivo que contiene recursos precompilados, permite que la aplicación cargue contenido relacionado con el idioma.

3.5. Desarrollo multiplataforma

El desarrollo multiplataforma consiste en la codificación de aplicaciones que pueden ser ejecutadas en distintos sistemas operativos o plataformas. Cada sistema operativo o plataforma, de forma habitual, requiere de un lenguaje de programación en concreto para la creación de software. Esto se debe a que cada sistema hace uso de una interfaz de programación de aplicaciones propia, es decir, el conjunto de instrucciones que indican como debe de interactuar el software con el sistema [19].

El desarrollo multiplataforma minimiza el proceso de crear una versión software específica para cada sistema operativo, esto se logra usando un lenguaje que pueda funcionar con diferentes interfaces de programación de aplicaciones. Por ello, con la codificación de un solo código fuente se puede crear un software ejecutable en distintos sistemas.

¿Cuáles son los beneficios del desarrollo multiplataforma?

El desarrollo multiplataforma brinda múltiples beneficios a aquellos programadores que recurren a su uso. Estos beneficios son:

- Mayor alcance de usuario en el mercado.
- Reducción de costes de producción.
- Aumento de la reutilización de código.

3.6. Persistencia

La persistencia hace referencia a la propiedad de los datos por la cual estos permanecen sin llegar a desaparecer. Si se hace referencia a este término en programación, indica la acción de mantener la información de un objeto de forma permanente y que puede ser recuperada en cualquier momento. Los datos pueden tener una duración efímera, lo que quiere decir que en el momento en el que estos cambian su valor ya no hay persistencia.

Tipos de persistencia

Persistencia de aplicación

Capacidad mediante la cual los datos sobreviven a la ejecución del programa que los ha creado. Sin esta capacidad, los datos desaparecen en cuanto el equipo se queda sin energía, ya que estos residirían en la memoria RAM, y esta es volátil. Por ello, es necesario que los datos se almacenen en un medio de almacenamiento secundario, que no pierda los datos en cuanto se quede sin energía. Por ejemplo, se puede guardar en disco un fichero con la configuración de un programa para que el usuario no tenga que volver a configurarlo la próxima vez que lo ejecute.

Persistencia de objetos

La persistencia de objetos se basa en la inicialización de objetos con sus atributos [30]. Esto se logra de dos formas:

- Un medio de almacenamiento fijo guarda un conjunto de datos que son recuperados cuando el tipo de objeto es creado, posteriormente esos datos se traspasan a las propiedades del objeto creado.
- Un objeto almacena los datos que serán transferidos al nuevo objeto cuando sea creado. Por lo tanto, los datos están almacenados en memoria.

Para almacenar los datos en un medio físico se recurre a un mecanismo llamada serialización, que contiene en una secuencia de bytes todos los datos del objeto.

La persistencia, en el ámbito de programación, permite almacenar, transferir y recuperar el estado de los objetos. Existen varias formas de hacerlo:

- Serialización.
- Mapeo relacional de objetos.
- Bases de datos orientadas a objetos.

3.7. Programación asíncrona

La programación asíncrona permite evitar retrasos durante la ejecución de un programa. Cuando las ejecuciones son síncronas se pueden producir

bloqueos durante la ejecución debido a la necesidad de esperar a que se ejecute algún proceso, lo que puede provocar un bloqueo total del programa hasta que termine de ejecutarse dicho proceso [5].

Las operaciones asíncronas más comunes son [8]:

- Obtención de datos de la red.
- Escritura de datos en una base de datos.
- Lectura de datos de un archivo.

Las principales ventajas del uso de programación asíncrona son:

- Permite la ejecución de múltiples tareas, garantizando que no se van a producir bloqueos.
- Evita el bloqueo de interfaces por la ejecución de operaciones largas.

4. Técnicas y herramientas

En este apartado se procederá a explicar las distintas técnicas y herramientas que han sido empleadas durante el desarrollo del proyecto.

4.1. Metodología ágil

Para la gestión del proyecto se ha recurrido a la metodología ágil denominada Scrum. Este tipo de metodología se basa en un enfoque iterativo, mediante *sprints* de intervalos de tiempos similares, en los que se realizan entregas rápidas y continuas del producto. La metodología Scrum permite incorporar cambios conforme finalizan los *sprints*, ya que posteriormente de realizar la entrega, se llevan a cabo reuniones para definir los objetivos del próximo *sprint*. Además, en estas reuniones se valora la calidad de la entrega, con el objetivo de realizar mejoras sobre la misma [13].

En una metodología de este tipo, es necesario definir tareas que permitan al desarrollador comprender qué funcionalidades debe de implementar durante el *sprint*. Todo ello, favorece la organización de las partes del proyecto y la priorización de la tareas a realizar.

Para el seguimiento tanto de *sprints* como de sus respectivas tareas, se ha empleado la herramienta web Zube. Esta herramienta permite definir los diferentes *sprints* del proyecto, estableciendo el marco temporal de estos, asi como una descripción en formato texto. Otra funcionalidad que ofrece Zube, de cara a la organización de *sprints*, es la creación de *cards*, es decir, la definición de las tareas planteadas para el *sprint*. Una ventaja destacable de Zube es la compatibilidad que ofrece con GitHub, ya que las tareas creadas pasarán a ser *issues* en el repositorio del proyecto en GitHub.

4.2. Documentación

Como herramienta para la redacción de la documentación se ha optado por usar LaTeX. Esta decisión se ve reforzada por la alta calidad en la elaboración de documentos técnicos, ya que LaTeX ofrece un estilo elegante, así como una presentación más organizada. Además, LaTeX aporta una gran flexibilidad para la creación de documentos de forma personalizada.

4.3. Control de versiones

Para el control de las versiones del proyecto, se han empleado dos herramientas que se utilizan de forma conjunta:

- **GitHub:** Es una plataforma que permite alojar repositorios Git en la nube. Además, incorpora funcionalidades muy interesantes relacionadas con el desarrollo de proyectos, entre ellas destacan el historial de *commits*, el control de ramas, control de *issues*, integración aplicaciones de terceros, etc. Se ha elegido como herramienta principal por su comodidad y por la experiencia previa utilizándola los últimos años.
- **Git Bash:** Es un herramienta que funciona como terminal de linea de comandos para manejar repositorios Git. Se ha elegido como herramienta por la experiencia en su uso, así como por su integración con el editor de código empleado.

4.4. Editor de código

En cuanto al editor de código, existen diversas opciones disponibles. Sin embargo, para este proyecto se ha optado por Visual Studio Code, herramienta que he usado durante mis estudios universitarios. Este editor por la comodidad de uso, la gran cantidad de extensiones disponibles y su flexibilidad para ser configurado según las preferencias del programador.

4.5. Lenguaje de programación y *framework*

Dart

Dart es el lenguaje de programación elegido para este proyecto. Desarrollado por Google, se caracteriza por una sintaxis familiar para programar

aplicaciones de forma rápida. Una de sus principales ventajas es que puede compilarse a código nativo, esto permite que se pueda ejecutar en diferentes entornos [7].

En este proyecto, la lógica de negocio, la comunicación con la API y el desarrollo del *front-end* se llevan a cabo en Dart, además este proceso se facilita con las herramientas integradas en su SDK de Flutter.

Flutter

Flutter es un *Software Development Kit* (SDK) creado por Google que permite crear interfaces gráficas a través de *widgets* predefinidos o personalizados. Una gran ventaja de Flutter es su capacidad de lograr que una interfaz sea reactiva, además de adaptarla a cada plataforma.[16]

Además, Flutter introduce la funcionalidad *hot-reload*, que refresca la interfaz de forma inmediata al realizar cambios en el código, acelerando así el proceso de desarrollo.

4.6. API de Moodle

La API de Servicios Web de Moodle permite que sistemas externos tengan acceso a funciones que son unicamente accesibles desde una plataforma Moodle. Estas funciones van desde servicio de autenticación hasta obtención de datos del usuario, entre decenas de operaciones. Para su uso en una aplicación externa, es necesaria la creación un token que permita acceder a los servicios de la API. Este token será usado en las llamadas a las funciones junto a otros parámetros requeridos. La respuesta a estas llamadas, se retorna en formato JSON con los datos solicitados [21].

Las ventajas de usar la API de Moodle son:

- Facilita la integración de los servicios de Moodle en aplicaciones móviles.
- Permite conectar Moodle con otros LMS.

4.7. Base de datos

En un principio se planteó utilizar *Firebase*, sin embargo, la ausencia de algunos ficheros impidió la integración. Como alternativa se recurrió a

Supabase, una plataforma *Backend as a Service* en la nube que permite crear y gestionar servicios *backend*. En este proyecto se emplea unicamente la herramienta de gestión de base de datos, aprovechando sus ventajas:

- Compatibilidad con SQL y políticas de seguridad a nivel de fila (RLS).
- Sencilla integración con Flutter, ya que se dispone de una dependencia que facilita el proceso de gestión de datos.

4.8. Prototipado

Para el prototipado del proyecto se barajaron varias alternativas como *Pencil*, *Canva*, *Adobe XD*, etc. Al final se optó por usar *Figma*, que permite crear prototipos interactivos y visualizarlos de forma sencilla en distintos dispositivos. Además, cuenta con el respaldo de una amplia comunidad que publica componentes y recursos que se pueden reutilizar en proyectos propios.

4.9. Emuladores para debug

Durante el desarrollo de la aplicación se emplearon diferentes emuladores para verificar como se comportaba la app en distintos entornos.

- **Android Emulator:** Para la emulación de la app en entornos de dispositivos Android.
- **Windows Emulator:** Para la emulación de la app en entornos Windows.

4.10. Dependencias del proyecto

Para facilitar y agilizar el proceso de desarrollo se ha recurrido al uso *dependencias*, estas permiten emplear funcionalidades desarrolladas por otro desarrolladores de la comunidad. Las dependencias empleadas son:

http

Dependencia que permite consumir recursos HTTP mediante peticiones *GET* ó *POST*. El uso de esta dependencia permite consumir las funciones de la API de Moodle para crear los modelos de la app [9].

shared_preferences

Dependencia que permite almacenar de forma local datos en distintos tipos de formato [17].

intl

Dependencia que contiene código para lidiar con mensajes internacionales, formateo de fechas y números, y otro tipo de incidencias de internacionalización [10].

flutter_localizations

Dependencia que facilita el soporte de internacionalización de la app. Permite traducir textos, adaptar formatos de fecha, hora y más en función de la localización del usuario [28].

supabase_flutter

Dependencia que facilita la integración de Supabase en una app de Flutter. Supabase se usa de forma alternativa a Firebase y permite la implementación de bases de datos, autenticación de usuarios, almacenamiento de archivos, etc [24].

flutter_dotenv

Dependencia empleada para configurar las variables globales de una app Flutter, mediante el uso de un archivo .env. Esto permite mantener la privacidad de las *API KEYS* [27].

url_launcher

Dependencia que permite lanzar la página a una URL indicada, es decir, permite el correcto funcionamiento de urls. Ofrece soporte web, móvil y SMS [18].

flutter_widget_from_html

Dependencia que permite renderizar el formato html y transformarlo en un widget que ofrece soporte *hyperlink*, imagen, audio, video y otras cuantas etiquetas [6].

sliding_up_panel

Dependencia que permite implementar en la aplicación un panel deslizante que se superpone al contenido principal y puede arrastrarse de forma vertical para mostrar su contenido [1].

calendar_date_picker2

Dependencia que implementa un calendario con una interfaz mucho más flexible y con opciones más avanzadas que *DatePicker*. Permite escoger una fecha, varias fechas o rangos de fechas, mediante un calendario con una gran variedad de personalizaciones, como colores, formatos de fechas, estilos para los distintos días de la semana, etc [26].

flutter_slidable

Dependencia que permite añadir a cualquier elemento la acción de deslizamiento horizontal para mostrar distintas acciones, como pueden ser *editar* o *eliminar* [23].

5. Aspectos relevantes del desarrollo del proyecto

En este apartado se explicaran todos los aspectos y decisiones que han supuesto un punto importante en el desarrollo del proyecto. Además, se hará mención de aquellos puntos críticos que han llevado a tomar esas decisiones.

5.1. Ciclo de vida

Para este proyecto se hizo uso de la metodología Scrum mencionada anteriormente. El motivo de haber empleado esta metodología y no otra, es la continua entrega de resultados, lo que permite una retroalimentación de los mismos, así como las mejoras a implementar para la siguiente entrega. Es un modelo de trabajo que permite llevar el proyecto al día, evitando así cargas de trabajo mayores en la entrega final del proyecto.

Al principio del proyecto se realizaban *sprints*/reuniones cada dos semanas ya que el proyecto se encontraba en fase de desarrollo. En estas semanas se proponían nuevas funcionalidades a implementar, así como, se revisaban los resultados de las propuestas de anteriores reuniones. A su vez, las tareas a llevar a cabo se anotaban como *issues* en el repositorio de GitHub, todo bajo el control de Zube. Cada *commit* relacionado con la *issue* que se lanzaba a la nube, lo hacia con la referencia a la *issue* que resolvía, con el objetivo de lograr un control de versiones sencillo de comprender.

Posteriormente, una vez finalizado el desarrollo principal, las reuniones tenían lugar cada semana. En estas, se daba una guía de como realizar la documentación completa del proyecto, además de, los pequeños cambios que se debían realizar para mejorar algunos aspectos de PlanLMS.

5.2. Tipo de arquitectura

Para el desarrollo del proyecto se optó por utilizar *Clean Architecture*, patrón utilizado en el diseño software que promueve la mantenibilidad del código, así como la escalabilidad del mismo. Se caracteriza por separar las distintas responsabilidades que componen el proyecto [12].

Este patrón se caracteriza por dividir el desarrollo en distintas capas en función de las responsabilidades que cada una aborda. Estas capas son [4]:

- **Capa de entidades:** alberga los objetos de negocio del dominio, es decir, clases que albergan los datos principales del negocio.
- **Capa de Casos de Uso:** contiene clases que representan acciones de negocio, es decir, contiene la lógica de la aplicación.
- **Capa de adaptadores:** se encarga de convertir los datos del exterior en un formato legible por las capas internas.
- **Capa de Frameworks:** esta capa contiene todo el código relativo a las bibliotecas, *frameworks* y herramientas externas.

En el caso de este proyecto, se ha divido el código en los siguientes directorios:

- ***backend*:** contiene toda la lógica de la aplicación.
- ***config*:** contiene todas las configuraciones generales de toda la aplicación, en este caso configuración de temas.
- ***models*:** contiene todas las entidades de negocio.
- ***presentation*:** contiene la interfaz gráfica de la aplicación.
- ***services*:** contiene la lógica que obtiene datos externos al proyecto.
- ***utils*:** contiene recursos útiles para todo el proyecto.

La división de responsabilidades se ha realizado de esta forma para facilitar el mantenimiento del código, así como la interpretación del funcionamiento del mismo. Por otra parte, esto favorece que el código pueda escalar en el futuro con nuevas implementaciones.

5.3. Almacenamiento de datos

La aplicación aborda dos métodos para el almacenamiento de datos, estos se usan dependiendo de la funcionalidad.

Almacenamiento de filtros

Un punto fuerte que ofrece la aplicación es la posibilidad de filtrar todo el contenido que el usuario desee visualizar. El problema surgía en que estos filtros que el usuario escogía, no eran persistentes, por lo tanto, se plantearon dos posibilidades: almacenarlos en la nube o almacenarlos localmente en el dispositivo del usuario. Se optó por la segunda opción por la comodidad que suponía su implementación, además de que permite al usuario de un dispositivo iniciar sesión con múltiples cuentas y almacenar los filtros de todas ellas.

Almacenamiento de tareas personales

La creación de tareas personales por parte del usuario requería del almacenamiento de las mismas. En este caso, se optó por alojarlas en una base de datos en la nube. El motivo de esta decisión reside en que el usuario debería de ser capaz de tener acceso a todas sus tareas desde cualquier dispositivo, ventaja que carece el almacenamiento en local.

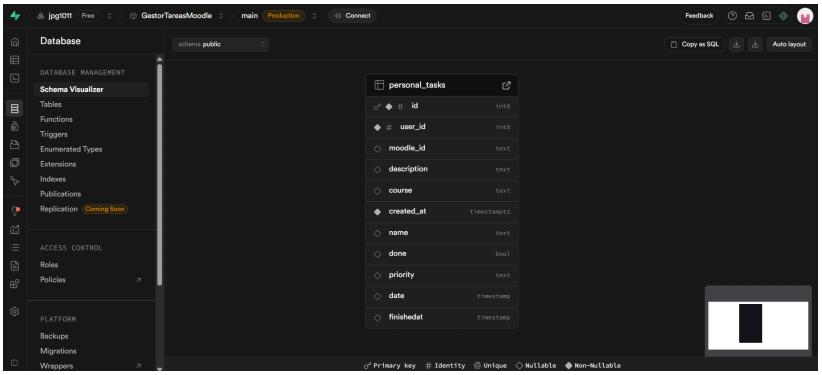

Figura 5.1: Dashboard de la base de datos de Supabase

5.4. Representación de tareas en un diagrama de Gantt

El diagrama de Gantt representa las tareas de forma que el usuario visualmente puede hacerse a la idea de la duración de dichas tareas. El problema surge cuando existe la ausencia de una de las dos fechas que el diagrama necesita para poder mostrar la duración de las tareas, así como el inicio y el fin de las mismas. En Moodle las tareas y los cuestionarios disponen de una configuración de fechas, pero ese factor depende del responsable que las configure. Por lo tanto, para poder realizar una adaptación de tal forma que se pudieran visualizar las tareas se planteó la asignación de fechas a las tareas en función de qué fechas se disponía. Para entender mejor esto último, se recoge en una tabla la asignación de fechas:

Disponibilidad		Asignación	
Fecha Apertura	**Fecha Cierre**	**Fecha Apertura**	**Fecha Cierre**
X	X	Fecha Apertura	Fecha Cierre
X	-	Fecha Apertura	Fecha Apertura
-	X	Fecha Cierre/Actual	Fecha Cierre/Actual
-	-	Fecha Actual	Fecha Actual

Tabla 5.1: Fechas disponibles y su respectiva asignación en el diagrama de Gantt

En los casos que existen dos posibles fechas para asignar, se realiza una comprobación. Si la fecha de cierre es anterior a la fecha actual, se asigna la fecha de cierre como fecha de apertura. En caso contrario, se asigna la fecha actual. Lo mismo ocurre en el caso de la fecha de cierre.

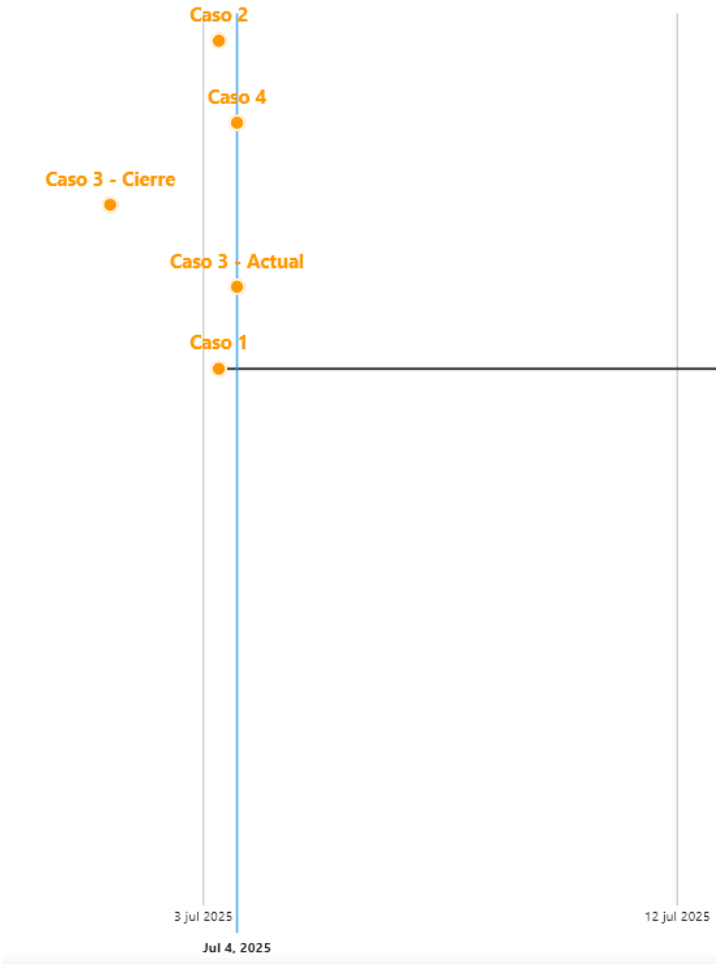

Figura 5.2: Ejemplo de la asignación que hace PlanLMS con las fechas

5.5. Librería del diagrama de Gantt

Una vez seleccionada la librería que iba a ser utilizada para la representación del diagrama de Gantt, se puso a prueba con casos prácticos de lo que sería un alumno. En este punto surgieron múltiples problemas:

- **Ausencia de fecha actual:** la visualización del diagrama de Gantt era bastante completa, pero le faltaba un componente muy importante, y era indicar al usuario en qué parte del diagrama se encontrar. Es decir, indicarle en que parte se encontraba la fecha actual, como solución se dibujó una línea azul perpendicular con un texto debajo que muestra la fecha.

- **Definición de alto y ancho:** el diagrama necesitaba una altura y un ancho específico para ser dibujado sobre la pantalla. Cuando se introducía una carga alta de tareas se producían errores que no permitían ver todas las tareas porque el tamaño del diagrama lo limitaba. Como solución se creó una función que calculaba el alto en función del número de tareas que se debían de mostrar. En cuanto al ancho, este puede ser configurado por el usuario a través de los filtros del diagrama.

- **Desajustes generales:** existían algunos desajustes relacionados con la visualización. Algunos desajustes encontrados son:

  - Textos cortados por falta de espacio en el diagrama.
  - Formato de fechas e idioma desconfigurados.

  Todos estos problemas mencionados fueron solucionados modificando el código fuente de la librería del diagrama de Gantt. Como se trata de una modificación, se consultó la licencia, que en este caso es MIT, lo que permitía su modificación y distribución bajo responsabilidad. Posteriormente se alojó localmente la librería con los nuevos cambios que solventaban todos los problemas.

- **Problemas al carecer de tareas:** uno de los primeros problemas encontrados consistía en que al no disponer de tareas para mostrar en el diagrama, se lanzaban errores en la ejecución de la aplicación. Para ello se eliminó la comprobación que limitaba el uso del diagrama sin tareas.

5.6. Sistema de filtrado

Como se ha mencionado anteriormente, el sistema de filtrado es uno de los puntos más fuertes de PlanLMS. La idea de hacer un filtro cuyos cambios afecten en toda la aplicación parte de la comodidad de permitir al usuario visualizar el contenido que más le interese, en el momento que desee. El sistema de filtrado se divide en cuatro filtros:

- **Filtrado de cursos:** permite al usuario seleccionar los cursos cuyas actividades o tareas personales quiera visualizar. Por defecto, si no selecciona ningún curso, se mostrarán todos los cursos en los que se encuentre matriculado.

- **Filtrado de tipo de actividad:** existen dos tipos de actividades en el diagrama de Gantt: los cuestionarios y las tareas. Por lo tanto, se permite al usuario seleccionar cual de los dos tipos desea visualizar. Por defecto, se muestran los dos tipos de actividades.
- **Filtrado de fechas:** permite al usuario filtrar las tareas que se encuentren entre un rango de fechas. También se da la posibilidad de seleccionar una de las dos fechas. La ventaja de poder filtrar actividades o tareas personales de esta forma, es que el usuario puede eliminar de su visualización aquellas actividades que ya no se encuentren en su marco temporal.
- **Filtrado de tipo de fecha:** consiste en dos casillas seleccionables que filtran las actividades de Moodle de forma que solo se muestren aquellas que dispongan de una fechas de apertura y de cierre o viceversa. La necesidad de implementar este filtro parte de la ausencia de alguna de estas fechas en la configuración de estas actividades de Moodle.

5.7. Resolución de errores

Cuando se planteó el almacenamiento de tareas personales en la nube, la primera opción a implementar fue Firebase. Se realizó la instalación siguiendo la documentación proporcionada por Google Firebase, pero a la hora de realizar la depuración de la aplicación surgió un problema. El compilador indicaba la ausencia de unos ficheros que eran necesarios para poder depurar la aplicación. Se trató de resolver este error, pero la documentación en internet era escasa, por lo tanto, se optó por usar Supabase en sustitución.

6. Trabajos relacionados

En este apartado se presentan distintos trabajos o proyectos similares a la aplicación desarrollada. El objetivo es realizar una breve comparativa de las opciones que ofrecen distintas alternativas en comparación a la desarrollada, con la finalidad de dar una visión al usuario de las opciones que puede utilizar en función de sus necesidades.

6.1. Aplicación móvil de Moodle

La aplicación oficial de Moodle ofrece todas las funcionalidades que ofrece la versión web, pero adaptada a dispositivos móviles [20]. La aplicación de Moodle incorpora las siguientes funciones:

- Acceso a los cursos.
- Comunicación con otros participantes de los cursos.
- Envío de tareas.
- Seguimiento del progreso.
- Realización de actividades.

6.2. Cuckoo

Se trata de un proyecto desarrollado en Flutter cuya finalidad es ofrecer un rendimiento mejorado en comparación a la aplicación oficial de Moodle. Además, implementa funcionalidades que pueden resultar muy útiles, así

como puede ser modificado para dar soporte a otras plataformas de Moodle [25].

Algunas funcionalidades que ofrece son:

- **Autenticación:** hace uso del mismo sistema de autenticación de Moodle con el fin de garantizar la seguridad de la información de los usuarios.
- **Estado de finalización:** Cuckoo se sincroniza con Moodle para comprobar si los eventos con fecha limite están completados y, según corresponda, marcarlas en las lista de eventos de la propia aplicación.
- **Eventos personalizados:** los usuarios pueden crear sus propios eventos y vincularlos con un curso en el que se encuentre inscrito.
- **Acceso a los recursos de Moodle:** dispone de acceso a los contenidos descargables de los cursos, y permite descargarlos sin necesidad de acceder a Moodle.

6.3. Trello

Trello es un software de gestión en linea, que hace uso de la metodología Kanban. Esta metodología se basa en el uso de tarjetas de trabajo en un tablero, similar a una línea de producción de tareas con un *status* [22].

Trello sirve para organizar y gestionar tareas, ya sean proyectos o tareas del día, entre otros. Trello dispone de una gran cantidad de características que hacen de su uso una comodidad:

- Sistema de trabajo colaborativo.
- Asignación de tareas.
- Organizar tareas por etiquetas.
- Multiplataforma.
- Fechas de vencimiento de tareas.
- Personalización de tableros.

6.4. Tabla de comparativas

A continuación se presenta una tabla comparando distintas funcionalidades de las aplicaciones presentadas anteriormente. El cometido de esta tabla es realizar una comparativa de los puntos fuertes de este proyecto, comparando las posibilidades del resto.

Funcionalidad \App	**Moodle**	**Cuckoo**	**Trello**	**Proyecto**
Autenticación	X	X	X	X
Conexión Moodle	X	X	-	X
Diagrama Gantt	-	-	-	X
Filtro General	-	-	-	X
Tareas Personales	X	X	X	X

Tabla 6.1: Comparativa de puntos fuertes del proyecto con otros proyectos

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Una vez finalizado el desarrollo del proyecto de la aplicación multiplataforma para gestionar tareas de Moodle, puedo afirmar que se han cumplido con todos los objetivos propuestos y, además, se han incorporado funcionalidades que no estaban previstas desde comienzo del proyecto. Por lo tanto, se han completado con todos los objetivos del proyecto. El hecho de haber alcanzado las expectativas previstas supone un logro gratificante, que es fruto de una planificación en la que se realizaban constantes revisiones así como nuevas propuestas.

En el aspecto personal, me siento satisfecho con el resultado final, ya que se ofrece una aplicación al alumnado que puede usar de forma que le sirva para la planificación de su tiempo en el momento de realizar sus tareas. El hecho de haber resuelto todos los obstáculos presentes durante la realización del proyecto, ha supuesto un logro personal, donde me he demostrado a mi mismo que con constancia todo es posible.

Por otra parte, ha sido positivo adquirir nuevos conocimientos relacionados con el desarrollo de aplicaciones multiplataforma. Además, me ha motivado a seguir ampliando mis conocimientos acerca de esta tecnología. Estos conocimientos podrán ser utilizados en el futuro para el desarrollo de nuevos proyectos o la continuación de este mismo si se diera la oportunidad.

7.2. Líneas de trabajo futuras

En esta sección se expondrán las posibles futuras implementaciones que podrían enriquecer el uso de la aplicación así como atraer a nuevos usuarios.

Resumen diario

Se podría incorporar una nueva funcionalidad que informe al usuario sobre su estado en la plataforma, este resumen le brindaría al usuario un resumen diario con información como:

- Tareas pendientes para el día actual o los siguientes días.
- Nuevas calificaciones en tareas o cuestionarios.

Esta funcionalidad nos proporcionaría un enfoque de lo útil que sería usar esta aplicación, ya que el usuario estaría en todo momento informado de sus labores o resultados de las mismas.

Visualización de estadísticas

El hecho de permitir que los usuarios visualicen el progreso en sus tareas o cursos puede servir de ayuda de cara al usuario para concienciarlo en el cumplimiento de sus actividades. Es decir, si un usuario no esta dedicando mucho tiempo a un curso, esta funcionalidad le puede incentivar a ser mas constante y atento.

Dentro de esta funcionalidad se pueden incluir muchos tipos de estadísticas, estas pueden ser:

- Porcentaje de actividades completadas respecto del total.
- Tiempo medio dedicado en la realización de actividades.
- Media ponderada de las calificaciones de actividades.

Internacionalización a otro idiomas

Actualmente la aplicación se encuentra en castellano, por lo que limita su uso a las regiones donde se haga uso de este idioma. El hecho de expandirse por otros horizontes requiere de adaptarse a las necesidades de los usuarios. Es por ello, que uno de los puntos cruciales para poder llegar a más público, es la integración de nuevos idiomas en la aplicación. Como consecuencia se podría comercializar a nivel global.

Implementación nuevos diagramas

Para mejorar la experiencia de gestión de tareas, se propone la integración de nuevos diagramas, gráficos o formas de representación. El hecho de recurrir a nuevas formas de representación de tareas, puede proporcionar al usuario distintas perspectivas para la gestión y planificación de sus labores. Los gráficos a implementar podrían ser:

- **Tablero Kanban:** consiste en un tablero divido en distintas secciones: "Por hacer", "En progreso" y "Hecho", entre otras varias posibilidades. Esta forma de trabajo permite desplazar las tareas entre las distintas secciones dependiendo de su estado.
- **Calendario o agenda:** herramientas que permiten asignar fechas a las tareas con la finalidad de programar plazos o bloques de trabajo.
- **Checklist:** se trata de una lista simple de tareas donde las realizadas quedan marcadas con una casilla, indicando su estado de finalización.

Sistema de notificaciones

Un aspecto muy importante a implementar de cara al futuro es la notificación de los eventos a realizar. Es muy importante que el usuario reciba avisos para no perder la constancia en la realización de sus labores. Este sistema se podría plantear de forma que el usuario pueda configurar las notificaciones según sus necesidades o intereses.

Implementación de nuevos filtros

El hecho de que la aplicación se pueda expandir hacia todos los horizontes, hace que el sistema de filtrado cobre mayor importancia. Por ello, la implementación de nuevos filtros podría suponer una interfaz más limpia así como una organización potenciada por dichos filtros. Los filtros que podrían ser implementados son:

- Filtro por estado de entrega.
- Filtro por prioridad de la tarea (Tareas personales).
- Filtro por calificaciones.
- Filtro por duración de las tareas.

Integración con otras plataformas

Actualmente la aplicación se conecta con las plataformas Moodle, función que permite internacionalizar la aplicación en distintas instituciones que utilicen Moodle. Para llegar a más usuarios, se plantea implementar la compatibilidad con otras plataformas de gestión de tareas y cursos. Algunas plataformas son:

- Blackbox
- Canvas
- Totara

Bibliografía

[1] akshathjain.com. sliding_up_panel flutter package. https://pub.dev/packages/sliding_up_panel.

[2] AWS Amazon. ¿qué es una interfaz de programación de aplicaciones (api)? https://aws.amazon.com/es/what-is/api/.

[3] BrowserStack. Apk file: What it is and how does it work? https://www.browserstack.com/guide/what-is-an-apk-file, 2025.

[4] Diego C. 'introducción a las 'clean architectures". https://medium.com/@diego.coder/introducci%C3%B3n-a-las-clean-architectures-723fe9fe17fa, 2023.

[5] Cassio. ¿qué es la programación asincrónica y cómo utilizarla? https://www.softplan.com.br/es/tech-writers/o-que-e-programacao-assincrona-e-como-utiliza-la/, 2022.

[6] daohoangson.com. flutter_widget_from_html flutter package. https://pub.dev/packages/flutter_widget_from_html.

[7] Dart. Dart programming language. https://dart.dev/.

[8] Dart. Asyncronous programming: futures, async, await. https://dart.dev/libraries/async/async-await, 2025.

[9] dart.dev. http dart package. https://pub.dev/packages/http.

[10] dart.dev. intl dart package. https://pub.dev/packages/intl.

[11] Equipo de Canva. Crear un diagrama de gantt: Cómo hacerlo, plantillas y consejos. https://www.canva.com/es_es/pizarra-online/diagrama-gantt/, 2024.

[12] DomainLogic. Clean architecture: qué es, importancia y beneficios para tu empresa. https://domainlogic.io/clean-architecture-que-es-importancia-y-beneficios-para-tu-empresa/.

[13] Claire Drumond. Qué es scrum y cómo empezar. https://www.atlassian.com/es/agile/scrum.

[14] Yúbal Fernández. Api: qué es y para qué sirve. https://www.xataka.com/basics/api-que-sirve, 2019.

[15] Yúbal Fernández. Apk de android: qué son estos archivos y cómo se instalan. https://www.xataka.com/basics/apk-android-que-estos-archivos-como-se-instalan, 2019.

[16] Flutter. Flutter - build apps for any screen. https://flutter.dev/.

[17] flutter.dev. shared_preferences flutter package. https://pub.dev/packages/shared_preferences.

[18] flutter.dev. url_launcher flutter package. https://pub.dev/packages/url_launcher.

[19] Claudia López. Desarrollo multiplataforma: Te explicamos cómo el software y las apps pueden funcionar en múltiples dispositivos. https://www.northware.mx/blog/desarrollo-multiplataforma/, 2022.

[20] Moodle. Moodle app. https://download.moodle.org/mobile.

[21] Moodle. Web service api functions. https://docs.moodle.org/dev/Web_service_API_functions, 2023.

[22] Yanina Muradas. Qué es trello. https://openwebinars.net/blog/que-es-trello/, 2020.

[23] romainrastel.com. flutter_slidable flutter package. https://pub.dev/packages/flutter_slidable.

[24] supabase.io. supabase_flutter flutter package. https://pub.dev/packages/supabase_flutter.

[25] thermitex. cuckoo-flutter. https://github.com/thermitex/cuckoo-flutter, 2015.

[26] Unverified uploader. calendar_date_picker2 flutter package. https://pub.dev/packages/calendar_date_picker2.

[27] Unverified uploader. flutter_dotenv flutter package. https://pub.dev/packages/flutter_dotenv.

[28] Unverified uploader. flutter_localization flutter package. https://pub.dev/packages/flutter_localization.

[29] Paul VanZandt. ¿qué es la gestión de tareas? definición, pasos, importancia, beneficios, herramientas y técnicas. https://ideascale.com/es/blogs/definicion-de-gestion-de-tareas/, 2022.

[30] Wikipedia. Persistencia (informática). https://es.wikipedia.org/wiki/Persistencia_(inform%C3%A1tica), 2024.

[31] Wikipedia. Apk (formato). https://es.wikipedia.org/wiki/APK_(formato), 2025.

[32] Wikipedia. Diagrama de gantt. https://es.wikipedia.org/wiki/Diagrama_de_Gantt, 2025.