CAPPTYON

Aplicación para entrenadores personales

Memoria

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD
D SALAMANCA

Julio de 2021

Autora

Marta Cano Belamendia

Tutores

Juan Francisco De Paz Santana

Gabriel Villarrubia González

Certificado de los tutores

D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

Hacen constar:

Que el trabajo titulado "Capptyon" ha sido realizado por Marta Cano Belamendia, con DNI 72827029J y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 1 de julio de 2021

DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)

Firmado digitalmente por DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)
Nombre de reconocimiento (DN): c=ES, serialNumber=76125754D, sn=DE PAZ, givenName=JUAN FRANCISCO, cn=DE PAZ SANTANA, JUAN FRANCISCO (AUTENTICACIÓN)
Fecha: 2021.07.02 11:03:32 +02'00'

D. Juan F. De Paz Santana

VILLARRUBIA GONZALEZ GABRIEL - 71932845T

Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-71932845T, givenName=GABRIEL, sn=VILLARRUBIA GONZALEZ, cn=VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2021.07.02 11:48:50 +02'00'

D. Gabriel Villarrubia González

Resumen

En el mundo del *fitness* siempre ha existido la figura del entrenador personal, pero este profesional siempre ha sido visto como un servicio alcanzable sólo a los bolsillos de los más afortunados. Pero hoy en día, y aunque ciertamente los mejores entrenadores no puedan ser permitidos por personas corrientes, el mundo del entrenamiento personal se ha extendido por todo el mundo, dejando de ser ese lujo del que pocos podían disfrutar.

Cada vez más personas que antes buscaban un mero escape de su realidad en el deporte están contratando los servicios de entrenadores personales para aprender a mejorar su forma o para tener un *planning* diseñado para ellos que les ayude a lograr sus objetivos.

Además, debido a la pandemia mundial y a que el mundo se está enfocando cada vez más hacia una era digital, la demanda de los entrenadores personales –y los entrenamientos– online están en alza.

Este proyecto se ha desarrollado con motivo del Trabajo de Fin de Grado, y busca presentar una aplicación para dispositivos móvil que ayude tanto a entrenadores personales como a sus clientes a gestionar entrenamientos online.

La aplicación dispone de dos perfiles: entrenador y cliente. Una de las grandes características de la aplicación es que cada entrenador dispondrá de un número único en el sistema, el cual proporcionará a sus clientes para unirlos a su cuenta. De este modo se crean 'grupos' que constan de un entrenador y todos sus clientes.

El entrenador tendrá un listado de sus clientes en su página principal desde la cual podrá acceder a visualizar la progresión y las actividades de sus clientes, y además tendrá la capacidad de subir documentos personalizados a las necesidades de sus clientes, como entrenamientos o artículos, así como documentos generales, que podrán ver todos sus clientes.

Por otro lado, el cliente podrá llevar una progresión de su evolución, pudiendo introducir datos que posteriormente son presentados visualmente por medio de gráficas, en las que se destacan las últimas dos entradas añadidas. Asimismo, podrá llevar un diario de las actividades que realiza a lo largo del año en el calendario que se le presenta en su página principal.

Ambos usuarios podrán gestionar su perfil, cambiando los datos pertinentes e incluso la foto en el menú principal, pudiendo elegir de entre una selección de fotografías rotativas. Por lo demás, ambos usuarios podrán hacer uso de un apartado de microblogueo en el que podrán compartir posts con el entrenador y todos aquellos clientes que compartan el 'grupo', pudiendo ordenar estos posts según el criterio de las opciones presentadas.

El desarrollo de este proyecto se enmarca en el ámbito de las tecnologías híbridas. Para su realización, se ha utilizado el *framewok* de Google Flutter, y el lenguaje que lo acompaña, Dart. Al mismo tiempo, se ha hecho uso de los servicios de otra de las plataformas de Google, Firebase, y sus servicios.

Para seguir un proceso comprensible y conseguir un producto robusto se ha empleado la metodología del Proceso Unificado.

Palabras clave: Entrenador, Cliente, Flutter, Actividades, Progresión

Abstract

The figure of a personal trainer has always existed in the world of fitness, but this professional has always been seen as a service attainable only for those with money on their pockets. But nowadays, and although the best personal trainers are still not obtainable for the general public, the world of personal training has become widespread worldwide, no more to be that luxury few people could enjoy.

More and more people that would only seek sports as an escape from their lives are hiring the services of personal trainers to learn how to better their form or as a way to have a plan designed for them that will help them achieve their goals.

More so, due to the global pandemic and the world being more directed into a digital era, the demand for personal trainers –and training– online are rising.

This project has been developed as a Final Degree Project, and aims to present a mobile phone app to help both the personal trainer and their clients in managing online trainings.

The app has two profiles: trainer and client. One of the big characteristics of the app is that each trainer will have a unique number in the system that they will have to supply to their clients to link them to their account. This way, 'groups' are created that consist of a trainer and all their clients.

The trainer will have a list of their clients in their main page, from which they will be able to visualize their clients' progression and activities, and they will have the ability to upload documents personalized to their clients' needs, be them new training schedules or articles, as well as general documents, which will be able to be accessed by all their clients.

On the other hand, the client will be able to keep a progression of their evolution, being able to add data that will be presented via graphics in which the last two added entries will be highlighted. More so, they will have at their disposal a calendar on their main page where they will be allowed to mark any day they carry out an activity.

Both types of user will be able to manage their profile, changing their information and even the picture in the main menu, which consists of a selection of images in a rotation. In addition, both users will be able to use the microblogging part of the app, in which they'll be able to share posts with the trainer and all their clients in the 'group', and they will be able to rearrange the order in which the posts appear in the timeline.

The development of this project is within the field of hybrid technologies. For its execution, Google's Flutter framework has been used, as well as the language that accompanies it, Dart. At the same time, the services of another one of Google's platforms have been used, Firebase, and its services.

In order to follow a comprehensible process and obtain a solid product, the methodology of Unified Process has been followed.

Keywords: Trainer, Client, Flutter, Activities, Progression

Tabla de contenido

Lista de tablas

Lista de ilustraciones

1. Introducción

La actividad física es importante para tener un sistema inmune fuerte, y ayuda a reducir los riesgos de ciertas enfermedades. Los gimnasios siempre han sido un sitio donde las personas iban a hacer deporte, pero con la situación actual de la pandemia y la digitalización de la sociedad, hay gente que busca más de lo que un gimnasio pueda ofrecerles.

Es en esos gimnasios donde las empresas de entrenadores personales ofertan sus servicios a tanto clientes frecuentes como a nuevos, pero las nuevas tendencias listadas por The American College of Sports Medicine en su *Worldwide Survey of Fitness Trends*[1], publicó a principios de este año que los entrenamientos online dominarían por encima de entrenamientos de fuerza o actividades al aire libre.

Y no es para menos, ya que las plataformas de *streaming* o redes sociales, donde *celebrities* hacen su vida, ayudan a visibilizar el trabajo de estos profesionales. Por ello, muchas personas ven el entrenamiento personal como una herramienta personalizada a las necesidades de cada uno para poder mantener un estado óptimo de salud y no tanto como solución estética.[2]

El entrenamiento personal online está creciendo, ya que gracias a ello no es necesario ajustarse a horarios imposibles limitados por gimnasios o las jornadas laborales de los entrenadores personales, ya que el cliente puede realizar estos ejercicios en su casa o en el exterior en el momento que mejor le convenga.

Este proyecto está diseñado para ayudar a los entrenadores personales a gestionar sus clientes, tanto para empresas de entrenadores personales como para los entrenadores autónomos, y para que estos clientes puedan visualizar sus resultados mediante una progresión.

Al ser una herramienta digitalizada donde la información se transmite de manera online, los clientes y entrenadores podrán tener una comunicación directa y sencilla, donde los entrenamientos estarán guiados por documentos y actividades realizadas, y las dudas podrán resolverse casi al minuto.

Esta memoria recoge la documentación relativa al proyecto de fin de grado 'Capptyon', y está acompañada de la documentación que la completa. Se divide en los siguientes apartados.

En primer lugar se dará una explicación de los objetivos del proyecto de forma general como punto de partida del desarrollo.

A continuación, se hablará de las técnicas y herramientas que han hecho posible la creación de este proyecto.

Posteriormente, se hará una breve exposición de los anexos adjuntados, dando una explicación de los aspectos relevantes. Los anexos son:

- **Anexo I. Planificación Temporal.** Se compone de la planificación temporal y estimación del esfuerzo.
- **Anexo II. Especificación de Requisitos Software.** Formado por el catálogo de requisitos del sistema, descripción de los objetivos, casos de uso, y matrices de rastreabilidad.
- **Anexo III. Análisis de Requisitos del Sistema.** Describe el modelo del dominio del problema en diagrama de clases del análisis, paquetes, y clases siguiendo la vista de arquitectura inicial.
- **Anexo IV. Especificación de Diseño.** Recoge la explicación del diseño arquitectónico a seguir en la implementación. Se detalla el diagrama de clases de la fase anterior refinándolo, y se obtiene una vista de arquitectura con la base de datos a utilizar. Se realizan los casos de uso y el diseño de datos.

- **Anexo V. Documentación Técnica.** Explica el lenguaje y *framework* escogidos y las configuraciones básicas para la ejecución de la aplicación. Se resumen la organización del código y las clases más importantes.
- **Anexo VI. Manual de Usuario.** Describe paso por paso las acciones a realizar por un usuario para tener una completa idea de cómo manejar la aplicación.

Se continúa con las conclusiones del proyecto y por último se expondrán las posibles líneas de trabajo futuras.

2. Objetivos del proyecto

El objetivo principal de este proyecto es realizar una aplicación móvil dirigida a entrenadores personales trabajando autónomamente o a empresas de entrenamiento personal. Por ello, se pretende conseguir una aplicación de uso intuitivo y fácil aprendizaje que pueda manejar la gestión de entrenadores y clientes.

2.1. Objetivos del sistema

Dentro del sistema, el objetivo principal del entrenador es poder visualizar las actividades y progresión de su cliente, además de poder subir documentos donde estarán indicados los ejercicios a realizar. Por ello, es importante que se disponga de un apartado en el que el entrenador pueda subir PDFs individualizados, pero también dispondrá de una sección en la que estos documentos puedan ser compartidos entre todos sus clientes. Además, visualizar la progresión de su cliente así como sus actividades diarias, podrá permitir al entrenador refinar los entrenamientos acorde a las necesidades del cliente.

Este cliente podrá llevar un diario de sus actividades en su calendario, pudiendo añadir todas sus actividades realizadas en cada día. Mantener un registro de su progresión le permitirá saber si sigue las indicaciones de su entrenador o necesita cambiar sus entrenamientos. Estas entradas son el peso y la grasa corporal, y podrá visualizar las marcas a modo de gráficas, donde se le mostrará la fecha de los dos últimos datos introducidos.

Por supuesto, el sistema deberá hacer una buena gestión del perfil de cada usuario, permitiendo que puedan editar sus datos personales. Los usuarios contarán en su perfil con un apartado opcional de 'objetivos', donde podrán escribir la razón o razones por las que contrataron un entrenador; ya que al avanzar en el entrenamiento las razones de seguir en él pueden variar.

La comunicación es otra de las grandes funcionalidades, ya que sin ella el cliente y el entrenador no podrán comunicarse, y el sistema deberá encargarse de que esta se hace sin interrupciones y de manera fluida, así como la notificación de mensajes nuevos.

Finalmente, los clientes de un entrenador y este, podrán interactuar unos con otros en el apartado de microblogueo, en el que podrán compartir experiencias o información. Esta aplicación no tendrá como objetivo principal que los clientes puedan intercambiar mensajes entre sí, pero aun así es importante hacerles saber que no son los únicos en ello. Este objetivo permitirá una socialización limitada donde los usuarios de un grupo puedan intercambiar información relevante.

Otro de los objetivos clave que el sistema debe tener es tener una interfaz clara y concisa, ya que el rango de edades de las personas que pudiera utilizar esta aplicación podría ser dispar y ha de abarcar un rango de edad lo más amplio.

Es muy esencial que la aplicación sea robusta y escalable; al ser el entrenamiento online un área que está en constante evolución, es necesario que la aplicación pueda evolucionar también. Por tanto, la escalabilidad será un aspecto clave.

2.2. Objetivos personales

Este proyecto ha supuesto un reto en aprender un nuevo lenguaje de programación partiendo desde cero, pero ha sido un reto interesante de completar. Conocer de primera mano las tecnologías y herramientas que se utilizan en el mundo laboral y poder crear algo desde la nada ha sido una gran satisfacción.

Las aplicaciones de móvil están en auge, y poder aventurarme en ese sector aplicando los conocimientos aprendidos a lo largo de la carrera ha sido curioso ya que sin más fecha de límite que la entrega de este proyecto, me ha servido para planificar, diseñar, e implementar un sistema que de otra manera no hubiera sido posible.

3. Conceptos teóricos

3.1. Actividad física

La OMS (Organización Mundial de la Salud) define la actividad física como cualquier movimiento corporal producido por los músculos esqueléticos, con el consiguiente consumo de energía[3]. Por ello, cualquier desplazamiento que realice una persona, ya sea en bicicleta o andando, se considera actividad física, así como la práctica de deportes o juegos.

Se ha demostrado que la actividad física regular ayuda a prevenir enfermedades cardíacas, diabetes, e incluso varios tipos de cáncer, y no sólo mejora la calidad de vida y bienestar general, sino que puede llegar a mejorar la salud mental.

La cantidad de actividad física que requiere cada persona depende de su grupo de edad y necesidades de su cuerpo, pero se coincide en que los adultos deberían ejercitarse por lo menos dos veces por semana, con actividades físicas aeróbicas moderadas y actividades de fortalecimiento muscular que pueden llegar a prevenir caídas.

3.2. ¿Qué es un entrenamiento personal?

Ilustración 1. Entrenador personal y cliente

Un entrenador personal es un profesional del *fitness*, cualificado en actividad física, que prescribe ejercicios, motiva, y fija metas de forma individualizada teniendo en cuenta las condiciones físicas y los objetivos de cada uno de sus clientes.[4]

Un entrenamiento personal es un programa de entrenamiento elaborado por el entrenador basado en el tiempo que disponga su cliente y los objetivos que quiera alcanzar. Los objetivos de un

cliente pueden variar en el tiempo y gracias a su entrenador, pero generalmente las personas buscan mantener su cuerpo, tonificarlo, o la pérdida de grasa.

Una de las grandes ventajas de entrenamientos personalizados es poder llevar un programa de manera constante y consistente para poder así invertir mejor su tiempo. Además, un entrenador personal se asegurará de mejorar la técnica de su cliente para entrenar de manera más segura. Como se puede observar en la Ilustración 1, un entrenador personal corrige la forma en la que se realizan los ejercicios constantemente, dando indicaciones o sugerencias.

3.3. Material Design

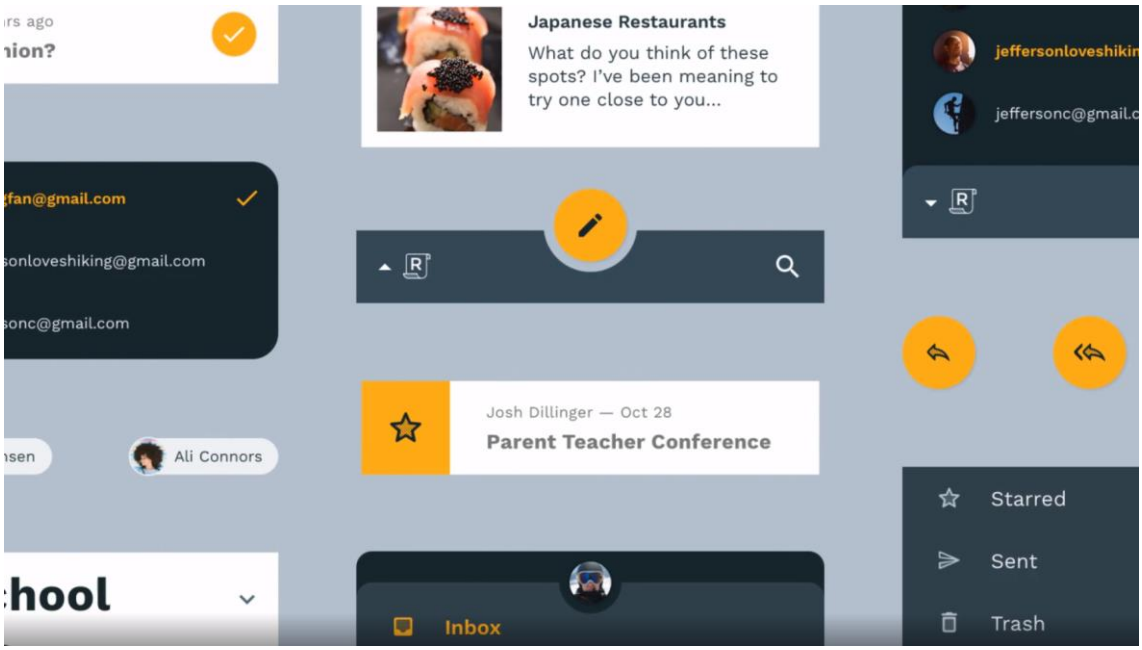


Ilustración 2. Concepto de Material Design

El Material Design es una nueva filosofía que empezó enfocada para el diseño Android, pero por sus pautas respecto a los colores, sombras, y profundidad para mostrar la realidad material en una aplicación, su implementación comenzó a extenderse a toda la web. Por ello, es fácil encontrar esta filosofía tanto en aplicaciones Android como en páginas web y las diferentes plataformas software.

Es un sistema de diseño creado por Google para incorporar coherencia estética y funcional en todos sus productos. Según su definición oficial: "es un lenguaje que combina los principios innovadores de la tecnología con las normas clásicas del diseño."[5]

En la Ilustración 2 se pueden ver ejemplos de Material Design. Este es un diseño con tipografía clara, casillas bien ordenadas, colores, e imágenes llamativas para no perder el foco y un sentido del orden con la jerarquía marcada. El juego de luces y sombras proporciona indicios de cómo se comportará un elemento y en qué nivel se encuentra, puesto que un cerebro humano entiende que el objeto que recibe la sombra de otro, se encuentra en un nivel inferior.[6]

El movimiento también toma gran importancia en esta filosofía, ya que centra la atención del usuario en él y mantiene la continuidad de la aplicación a través de las transiciones.

3.4. Widgets

```
class Car extends StatelessWidget {
  const car({ Key key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(child: Text("Red car"));
  }
}
```

Ilustración 3. Ejemplo de Stateless Widget

Los widgets se pueden considerar como bloques de código reusables que describen la apariencia de la interfaz de usuario (UI, User Interface). Los widgets describen cómo debería ser su vista dada su configuración y estados actuales, ya que cuando el estado de uno cambia, este tiene que reconstruir su descripción realizando los cambios mínimos necesarios en el árbol de renderizado para la transición de un estado al siguiente.

Los widgets pueden ser de dos tipos: Stateless Widgets, que como su nombre indica, no tienen estado, y los Stateful Widgets, que sí disponen de él. Como se ve en la Ilustración 3, un widget sin estado muestra elementos estáticos, en este caso un *Container* con un texto, y en el caso de Stateful Widget, el widget será dinámico, por lo que realizan el seguimiento de los cambios producidos y llegar a modificar la interfaz en función de estos cambios.[8]

4. Técnicas y herramientas utilizadas

En este apartado se expondrán las técnicas y herramientas de desarrollo utilizadas para la compleción del proyecto.

4.1. Ingeniería del software

Este proyecto se ha realizado bajo el marco del Proceso Unificado (Unified Process, UP) y el Lenguaje Unificado de Modelado (Unified Modeling Language, UML). El Proceso Unificado[9] es un proceso de desarrollo de software formado por un conjunto de actividades necesarias que transforman los requisitos del usuario en un sistema software.

Como se puede ver en la Ilustración 4, el UP está dividido en fases que realizan el mismo tipo de tareas aunque en las distintas fases se le dé más importancia a unas labores que a otras.

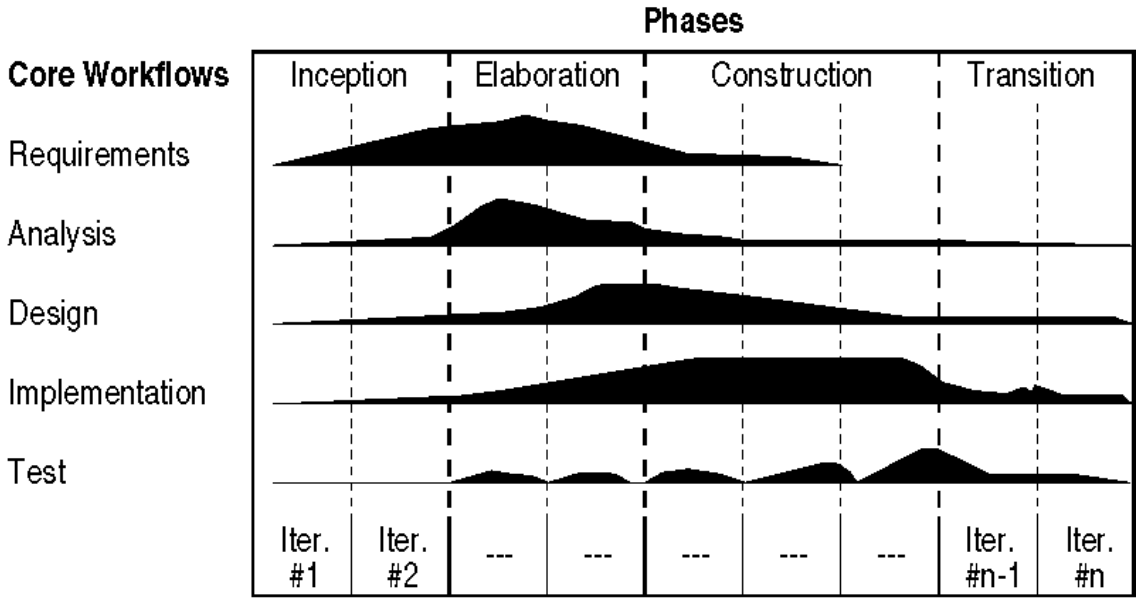


Ilustración 4. Fases del Proceso Unificado

Esta metodología tiene una serie de ventajas:

Está dirigido por los casos de uso. Los casos de uso guían el proceso de desarrollo, y basándose en ellos, se crean una serie de modelos de diseño e implementación. De este modo, no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor a través de flujos de trabajo que parten de los casos de uso.

Centrado en la arquitectura. Se necesita una arquitectura para comprender el sistema y organizar el desarrollo. Además, fomenta la reutilización y hace evolucionar el sistema ya que se desarrolla mediante iteraciones.

Es un proceso iterativo e incremental. Cada iteración consta de cuatro fases: inicio, elaboración, construcción, y transición, en las cuales se centran en las distintas disciplinas en mayor o menor medida, que obtiene como resultado una versión interna del sistema.

4.1.1. Tom's Planner

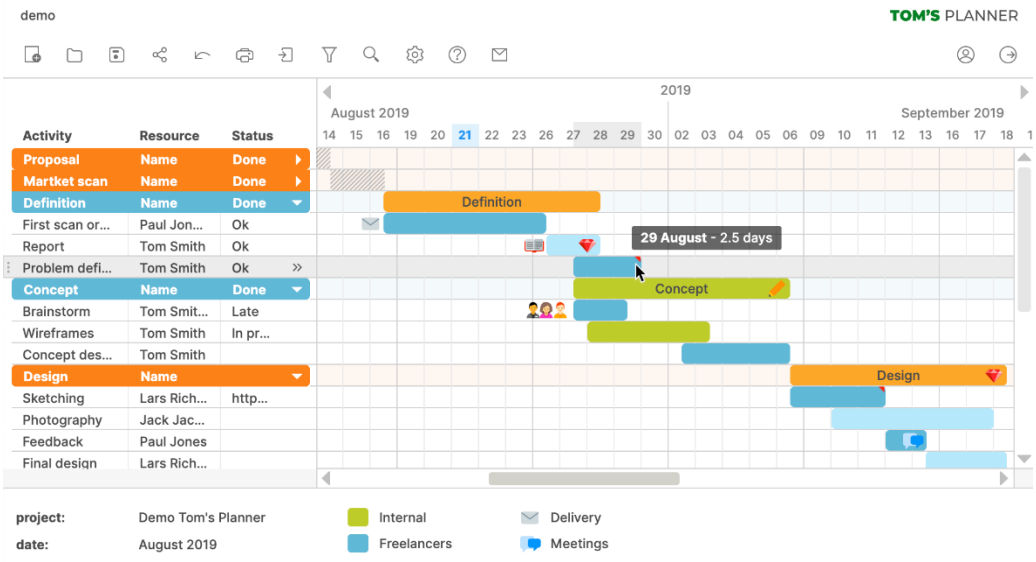


Ilustración 5. Herramienta Tom's Planner

El Diagrama de Gantt es una herramienta gráfica para la gestión de proyecto. Su función es visualizar las diferentes tareas o actividades de un proyecto y las relaciones que se establecen entre ellas de acuerdo con su duración o ubicación en el calendario.

Tom's Planner[10] es un software dedicado a la elaboración de los Diagramas de Gantt, que permite a los usuarios crear calendarios de proyectos en línea sin necesidad de instalación y con un uso fácil. Como se puede ver en la Ilustración 5, en Tom's Planner se puede añadir la longitud de tareas diferenciadas unas de otras por colores, así como el estado de las tarea realizadas o quién las ha realizado.

Se ha elegido está herramienta por su curva de aprendizaje eficiente y al ser sugerida por uno de los tutores.

4.1.2. EZEstimate

Es una herramienta software para la realización de la estimación del esfuerzo de un proyecto a partir de los puntos de caso de uso. Incluye en su estimación a los actores que toman parte en el sistema así como a los factores técnicos y de entorno.

Esta herramienta se dio a conocer en la asignatura de Gestión de Proyectos.

4.1.3. REM

Por el concepto iterativo de la Gestión de Requisitos, del proyecto, y el propio ciclo de vida del desarrollo, se consideran las herramientas CASE (Computer Aided Software Engineering) como la mejor opción que integra el proceso.[11]

Para ello, se ha utilizado la herramienta REM, conocida en la asignatura de Gestión de Proyectos.

REM (REquirements Management) es una herramienta diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software. Fue presentada por Dr. Amador Durán en septiembre de 2000 y ha sido obtenida gracias a la Universidad de Sevilla.[12]

Esta herramienta da como resultado un documento ordenado donde se encuentra la captura de requisitos en un formato estándar.

4.1.4. Visual Paradigm

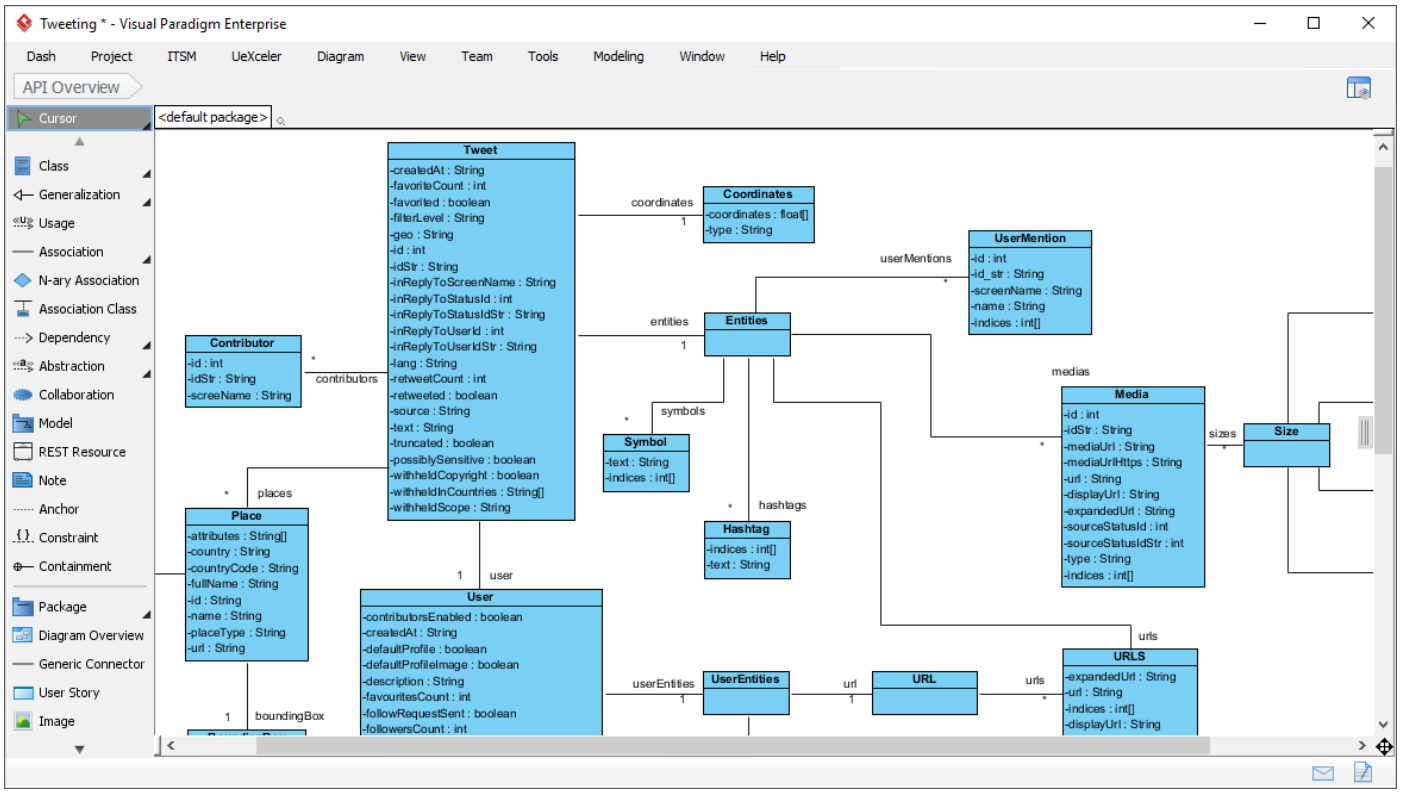


Ilustración 6. Herramienta Visual Paradigm

Es otra herramienta CASE para el modelado visual con UML. Esta sí es de pago, pero gracias a la licencia proporciona por la Universidad de Salamanca, se puede hacer completo uso de ella. Esta herramienta fue presentada en la asignatura de Ingeniería del Software.

Como se puede ver en la Ilustración 6, Visual Paradigm[13] soporta el lenguaje de modelado UML (Unified Modeling Language)[14], un lenguaje gráfico para visualizar, especificar, construir, y documentar un sistema. Los diagramas presentados en este proyecto han sido en su mayoría diseñados gracias a esta herramienta, incluyendo los diagramas de clases hasta los diagramas de secuencia entre otros.

4.2. Framework y Base de datos

4.2.1. Flutter

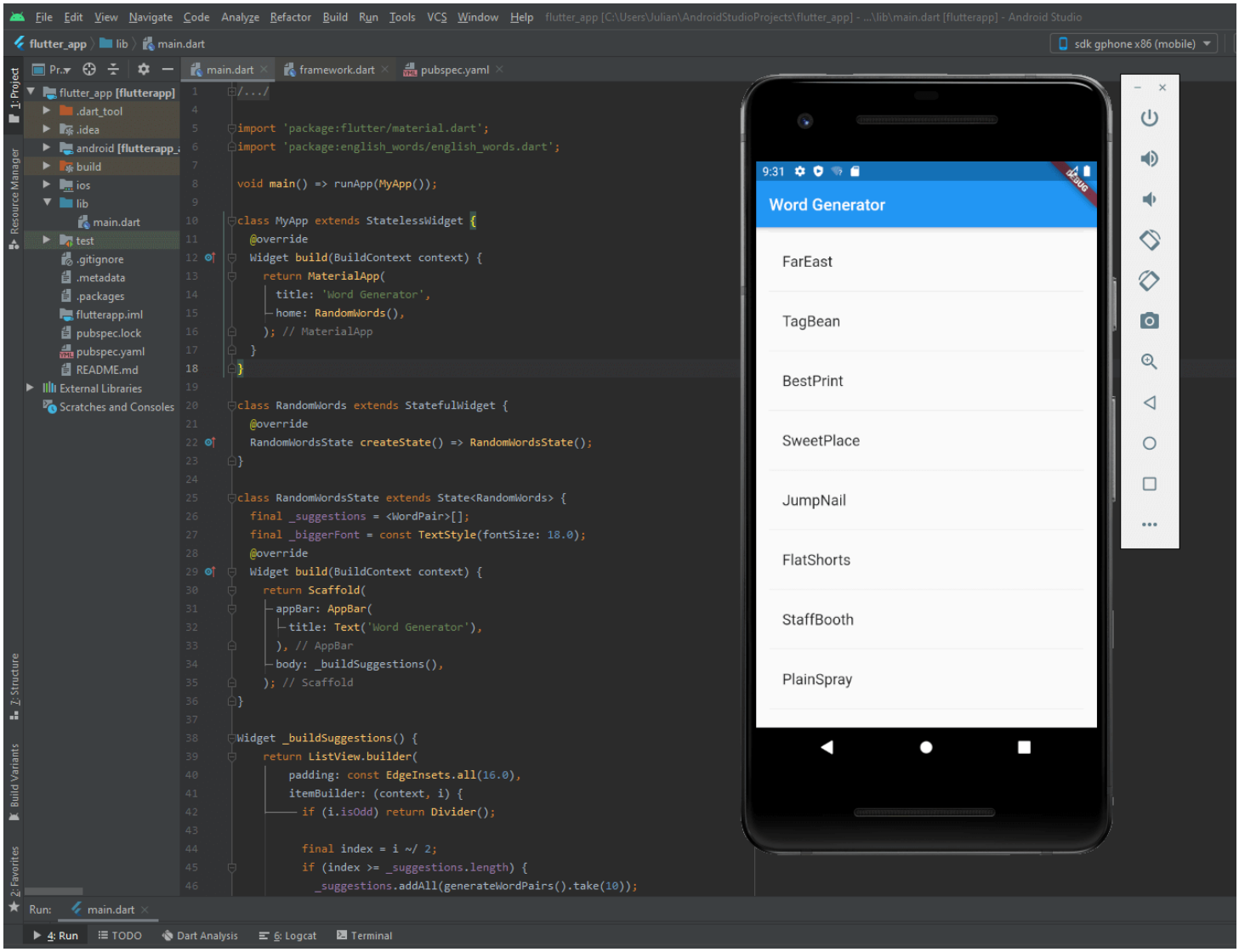


Ilustración 7. Flutter

Flutter es un *framework* elaborado por Google y publicado por primera vez como proyecto de código abierto a finales de 2018 para desarrollar aplicaciones para diferentes plataformas. Como gran característica tiene su habilidad de crear aplicaciones multiplataforma con un único código.

Flutter[15] ofrece un gran número de bibliotecas para elementos de las interfaces de Android e iOS, pero mientras que se comportan como se espera en cada sistema, relativo a la programación, no es necesario prestar atención a las particularidades de cada uno.

Flutter cuenta con una extensa documentación y catálogo de widgets que hace que su programación sea sencilla. Además, consta de unas claras ventajas[16] respecto a otros *frameworks*:

- **Rapidez en el desarrollo.** Permite desarrollar código a una gran velocidad gracias a su 'hot reload', que permite aplicar cambios de forma dinámica sin perder el contexto de la aplicación.
- **Interfaz de usuario expresiva y flexible.** Flutter permite hacer uso de elementos gráficos de cada uno de los sistemas operativos existentes, y se puede encontrar Material Design para formato Android y Cupertino Style para iOS. Como se puede observar en la Ilustración 7, línea 30, la aplicación creada con Flutter sigue la filosofía de Material Design, con Scaffold.
- **Características nativas.** Para poder desarrollar aplicaciones nativas multiplataforma, Flutter utiliza el lenguaje de programación Dart, creado por Google y compatible con todos los demás sistemas operativos.

4.2.2. Dart

Dart[17] es un lenguaje de *open source* desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos.

A diferencia de muchos lenguajes, Dart se diseñó para hacer el proceso de desarrollo lo más cómodo y rápido posible. Por ello, viene con un conjunto de herramientas integrado como su propio gestor de paquetes, compiladores, un analizador, y un formateador. Además, la máquina virtual de Dart y la compilación *Just-in-Time* hacen que los cambios realizados en el código se puedan ejecutar inmediatamente.

En cuanto a su sintaxis, es muy similar a lenguajes como JavaScript, Java, y C++, lo que hace que su aprendizaje sea muy fácil de aprender. Respecto a los otros lenguajes, Dart es más fácil de leer también, ya que su sintaxis se acerca al lenguaje humano. Hay menos comandos pero más posibilidades.

En 2019, la plataforma de desarrollo Stack Overflow[18] determinó que Dart era uno de los lenguajes de programación más populares únicamente por detrás de JavaScript.

Se ha usado el editor de código Android Studio junto con Flutter y Dart por su ligereza y soporte de depuración.

4.2.3. Plataforma Firebase

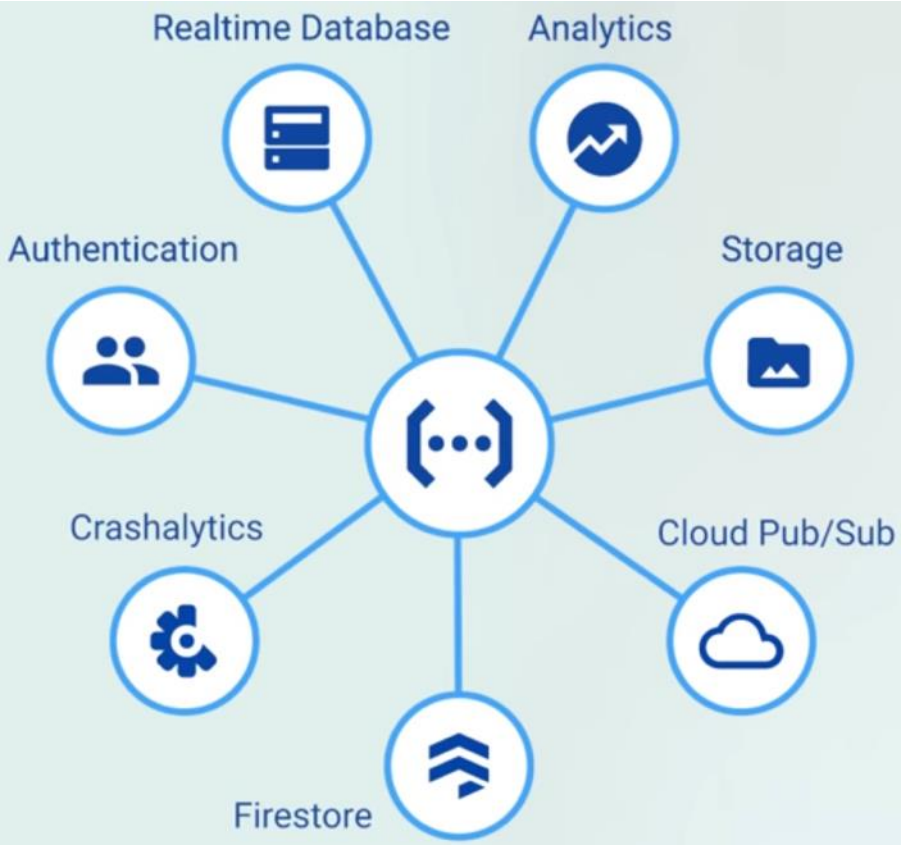


Ilustración 8. Funciones ofrecidas por Firebase

Firebase[19] es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para las distintas plataformas como Android, la web, o iOS, con lo que es más rápido trabajar en el desarrollo.

Creada en 2011, pasó a ser parte de Google en 2014 comenzando como una base de datos en tiempo real, mas al continuar añadiendo funcionalidades, permite agrupar los SDK de productos Google con distintos fines, facilitando su uso.

No sólo sirve para el almacenamiento de datos; entre otras cosas, las funcionalidades de Firebase se dividen en cuatro grupos[20]: Desarrollo (Develop), Crecimiento (Grow), Monetización (Earn), y Analítica (Analytics). En la Ilustración 8 se pueden ver una serie de funciones que ofrece Firebase, pero para este proyecto se han utilizado las referentes al grupo de Desarrollo, siendo estas las siguientes:

Firebase Firestore. Es una base de datos NoSQL creada para apps globales. Permite almacenar, sincronizar, y consultar fácilmente datos para las aplicaciones móviles y web a escala global.

Authentication. Ofrece un sistema de autenticación que permite tanto el registro mediante email y contraseña como el acceso utilizando perfiles de otras plataformas externas como Google o Twitter.

Cloud Storage. Firebase cuenta con un sistema de almacenamiento donde se pueden guardar los ficheros de las aplicaciones (vinculándolas con referencias a un árbol de ficheros) y sincronizarlos.

4.3. dartdoc

Dartdoc[21] es una herramienta incluida en el SDK de Dart que permite la creación de documentación de referencia de API a partir del código fuente desarrollado. Los ficheros generados serán estáticos en formato HTML con la documentación.

El comando a utilizar en los directorios indicados es *$ dartdoc* en la terminal, habiendo utilizado previamente comentarios de documentación /// o /**/ para identificar a estos.

5. Aspectos relevantes del desarrollo

A continuación se exponen los detalles y consideraciones de las partes más representativas de las fases del ciclo de vida del proyecto de final de carrera.

5.1. Marco de trabajo y planificación

En el apartado de Técnicas y herramientas utilizadas, apartado 4.1, se comentó que el proyecto ha sido realizado siguiendo la metodología del Proceso Unificado[22]. Esta metodología consta de cuatro fases siendo estas Inicio, Elaboración, Construcción, y Transición, que se dividen en iteraciones que evolucionan basándose en la iteración anterior hasta conseguir el producto final.

No obstante, en este proyecto se ha añadido al comienzo una fase más, **Adquisición de conocimientos**, para adquirir los conocimientos necesarios.

Tras ello comienza la **fase de inicio**, donde se define el alcance de proyecto y se representan las metas del proyecto y conceptos de dominio. Además, se crean unos inicios de prototipo que ayudarán a encaminar la aplicación a la visión del proyecto.

El final de la fase de inicio es un documento completo con los requisitos y objetivos que no estará supuesto a más grandes cambios a lo largo del proyecto.

A continuación comienza la **fase de elaboración**, donde el énfasis está en la refinación de la visión del proyecto, y por ello es probable la identificación de nuevos requisitos o alcances.

La **fase de construcción** es la fase que más tiempo requiere porque se centra en la implementación de los paquetes obtenidos anteriormente, siguiendo los distintos diagramas que los forman.

En esta fase se construye el sistema completo, y por ello se pone en foco las pruebas realizadas cada vez que se añadía una nueva funcionalidad, comprobando que el sistema no quedaba comprometido con cada nueva actualización.

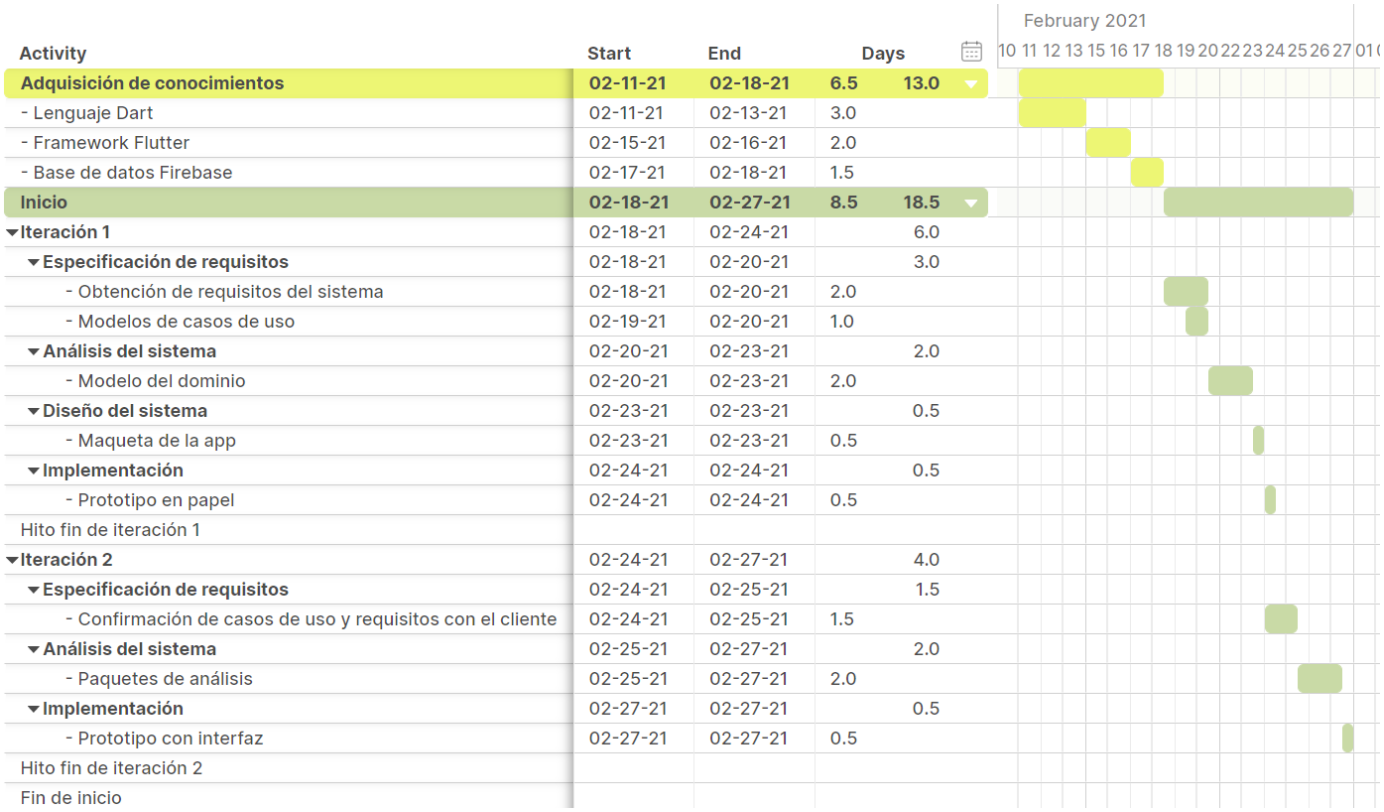
Por último se encuentra la **fase de transición**, en la que el producto es completamente terminado y funcional. El mayor esfuerzo se ha centrado en el refinamiento de la aplicación y pequeños detalles, así como con pruebas con usuarios reales.

5.1.1 Diagrama de Gantt

La planificación temporal es la identificación de tareas, asignación de tiempos y recursos a dichas tareas, y planificación de la secuencia de ejecución para que el tiempo de desarrollo sea mínimo. Es fundamental para tener una visión completa del calendario y el horario a seguir para evitar retrasos.

A continuación, Ilustración 9, se muestra una de las imágenes del diagrama de Gantt de este proyecto, la imagen referente a las fases de 'Adquisición de conocimientos' e 'Inicio' y las tareas realizadas en cada apartado. A la izquierda se muestran las tareas seguidas mientras que a la derecha se encuentra el tiempo.

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



Activity	Start	End	Days	
Adquisición de conocimientos	**02-11-21**	**02-18-21**	**6.5**	**13.0**
- Lenguaje Dart	02-11-21	02-13-21	3.0	
- Framework Flutter	02-15-21	02-16-21	2.0	
- Base de datos Firebase	02-17-21	02-18-21	1.5	
Inicio	**02-18-21**	**02-27-21**	**8.5**	**18.5**
Iteración 1	02-18-21	02-24-21		6.0
Especificación de requisitos	02-18-21	02-20-21		3.0
- Obtención de requisitos del sistema	02-18-21	02-20-21	2.0	
- Modelos de casos de uso	02-19-21	02-20-21	1.0	
Análisis del sistema	02-20-21	02-23-21		2.0
- Modelo del dominio	02-20-21	02-23-21	2.0	
Diseño del sistema	02-23-21	02-23-21		0.5
- Maqueta de la app	02-23-21	02-23-21	0.5	
Implementación	02-24-21	02-24-21		0.5
- Prototipo en papel	02-24-21	02-24-21	0.5	
Hito fin de iteración 1				
Iteración 2	02-24-21	02-27-21		4.0
Especificación de requisitos	02-24-21	02-25-21		1.5
- Confirmación de casos de uso y requisitos con el cliente	02-24-21	02-25-21	1.5	
Análisis del sistema	02-25-21	02-27-21		2.0
- Paquetes de análisis	02-25-21	02-27-21	2.0	
Implementación	02-27-21	02-27-21		0.5
- Prototipo con interfaz	02-27-21	02-27-21	0.5	
Hito fin de iteración 2				
Fin de inicio				

Ilustración 9. Diagrama de Gantt - Adquisición de conocimientos e Inicio

5.2. Estimación del esfuerzo

Antes de iniciar el proyecto se realizó una estimación de coste y esfuerzo con la herramienta EZEstimate, siguiendo la métrica UCP (Use Case Points) que considera actores, escenarios, y factores técnicos y de entorno para calcularlo.

Los UCP se calculan mediante la expresión:

$$UCP = UUCP * TCF * ECF$$

donde las siglas significan:

UUCP: Unadjusted Use Case Points, basado en el peso de los casos de uso UUCW (*Unadjusted Use Case Weight*) y el peso de los actores UAW (*Unadjusted Actor Weight*).

TCF: Technical Complexity Factor, o factores de complejidad técnicos.

ECF: Environment Complexity Factor, o factores de complejidad de entorno.

Los detalles de este apartado se encuentran en el Anexo I.

Tras introducir los valores necesarios (Ilustración 10) se obtiene un resultado estimado de 835,7 horas de persona, en línea con lo estimado en la planificación temporal.

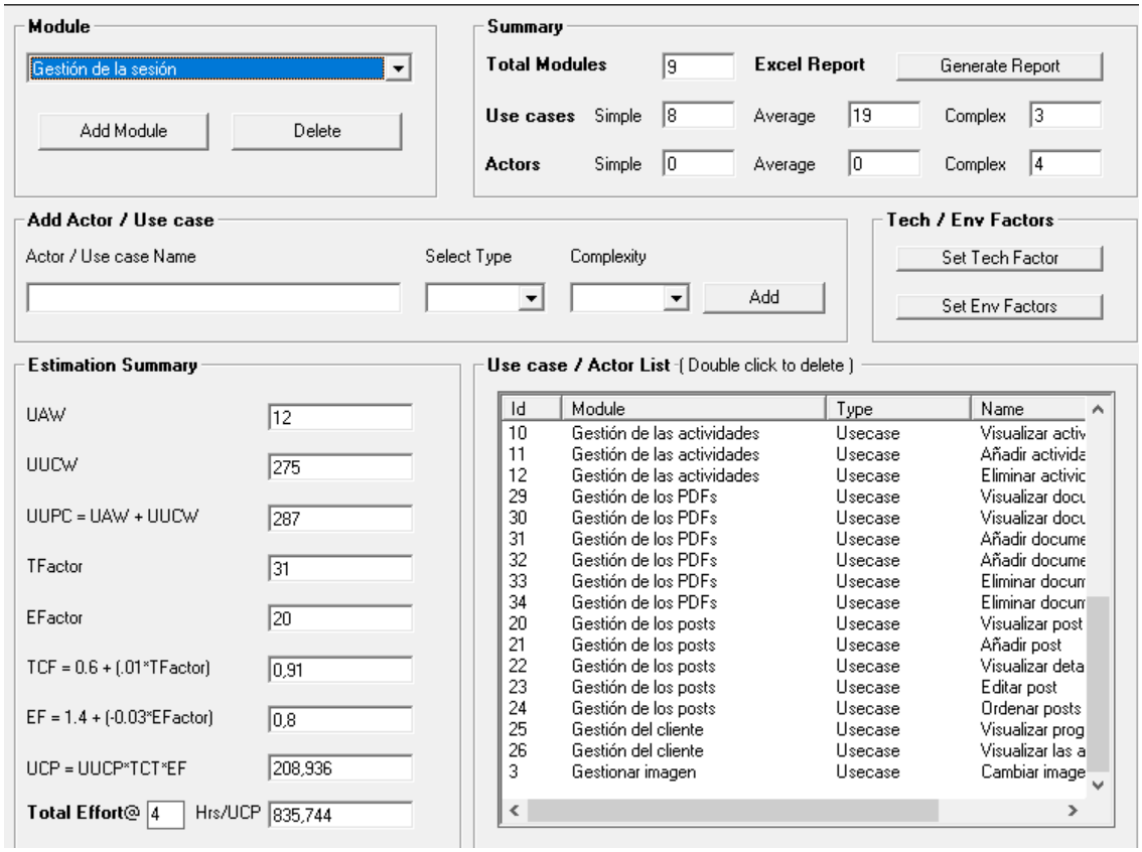


Ilustración 10. Captura del programa EZEstimate

5.3. Catálogo de requisitos

Para poder capturar los requisitos y la información del dominio del problema se ha seguido el método de Durán y Bernárdez, y se han utilizado los diagramas de casos de uso para reconocer los requisitos funcionales. La herramienta REM ha sido usada para crear las tablas.

El catálogo debe definir de forma precisa y completa los requisitos del sistema, debiendo estar abierta a cambios puntuales al fin de ajustarse a la visión.

El Anexo II habla más extensamente de este apartado, en el cual se describirán las diferentes secciones del catálogo, no obstante, se describirá a continuación los tipos que lo componen.

5.3.1. Objetivos

Los objetivos describen qué tiene que hacer el sistema. En este sistema, los objetivos son:

- Gestión de usuarios
- Gestión de la comunicación
- Gestión de posts
- Gestión de documentos
- Gestión de actividades
- Gestión de gráficas
- Gestión de la visualización de cliente
- Gestión de autenticación

Como ejemplo se presenta el cuarto objetivo, la tabla de 'Gestión de documentos'.

OBJ-0004	**Gestión de documentos**
Versión	1.0 (11/05/2021)
Autores	• Cano Belamendia, Marta
Fuentes	• De Paz Santana, Juan Francisco • Villarrubia González, Gabriel
Descripción	El sistema deberá *ser capaz de gestionar la subida y correcta visualización de PDFs.*
Subobjetivos	Ninguno
Importancia	importante
Urgencia	puede esperar
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 1. Objetivo - Gestión de documentos

5.3.2. Requisitos de información

Los requisitos de información son los que el sistema debe almacenar para que el sistema pueda funcionar acorde a las funcionalidades especificadas. En esta aplicación, los requisitos necesarios son:

- Información sobre usuarios
- Información sobre sala de chat
- Información sobre mensaje de chat
- Información sobre sala de post
- Información sobre post
- Información sobre documentos
- Información sobre el peso
- Información sobre la grasa corporal
- Información sobre actividades

El requisito de información del post servirá como ejemplo.

IRQ-0005	**Información sobre post**
Versión	1.0 (11/05/2021)
Autores	• Cano Belamendia, Marta
Fuentes	• De Paz Santana, Juan Francisco • Villarrubia González, Gabriel
Dependencias	Ninguno
Descripción	El sistema deberá almacenar la información correspondiente a *los post*. En concreto:
Datos específicos	• Nombre • Apellidos • uid • entrenadorCliente • numeroEntrenador • Fecha • Titulo del post • Cuerpo del post
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 2. Requisito de información - Información sobre post

5.3.3. Requisitos no funcionales

Estos requisitos no están ligados con la funcionalidad directamente, pero servirán para hacer el sistema más fácil de usar y más atractivo. Ellos son:

- Interfaz sencilla
- Seguridad
- Funcionamiento tiempo real

El requisito primero servirá como ejemplo.

FRQ-0001	**Interfaz sencilla**
Versión	1.0 (18/05/2021)
Autores	• Cano Belamendia, Marta
Fuentes	• De Paz Santana, Juan Francisco • Villarrubia González, Gabriel
Dependencias	Ninguno
Descripción	El sistema deberá *disponer de una interfaz sencilla para que el usuario navegue sin problemas.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

Tabla 3. Requisito no funcional - Interfaz sencilla

5.3.4. Requisitos funcionales

Representan la funcionalidad que el sistema debe poseer. Los requisitos funcionales incluyen los actores y los casos de uso.

Como actores en el sistema se encontrarán:

- Entrenador
- Cliente
- Usuario no identificado
- Usuario identificado

Donde los actores 'Entrenador' y 'Cliente' serán una especialización del actor 'Usuario identificado'.

La representación de la tabla del actor 'Entrenador' será la siguiente.

ACT-0001	Entrenador
Versión	1.0 (10/05/2021)
Autores	• Cano Belamendia, Marta
Fuentes	• De Paz Santana, Juan Francisco • Villarrubia González, Gabriel
Descripción	Este actor representa *un usuario que se ha logueado como entrenador y ejecutará las acciones destinadas a los entrenadores.*
Comentarios	Ninguno

Tabla 4. Actor - Entrenador

Por otro lado están los casos de uso. Para su representación se ha usado la herramienta Visual Paradigm mencionada anteriormente.

Primero se organizaron por paquetes, dando como resultado el diagrama de paquetes mostrado más abajo, Ilustración 11, para describir un esquema general y una mejor visión del sistema.

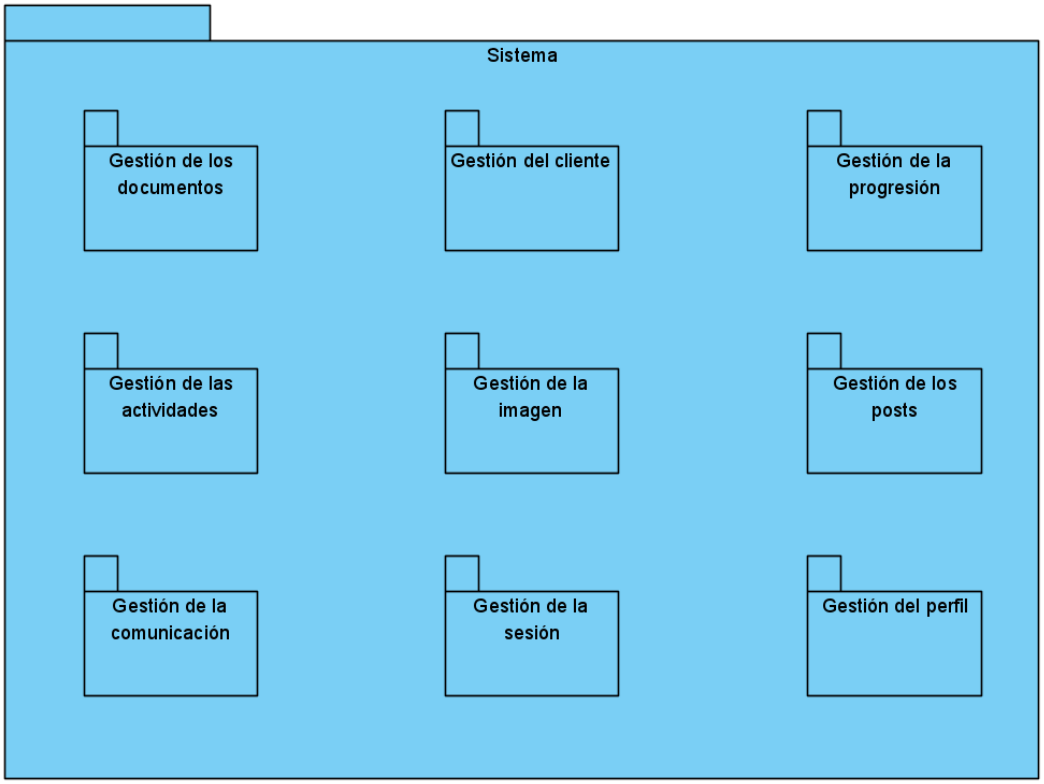


Ilustración 11. Diagrama de paquetes del sistema

En estos paquetes se pueden encontrar los siguientes casos de uso

Gestión de la sesión	Gestión de las actividades	Gestión del perfil
Información	Visualizar actividad	Visualizar perfil
Registrarse	Añadir actividad	Editar perfil
Inicio de sesión	Eliminar actividad	Eliminar cuenta
Salir		

Tabla 5. Casos de uso 1/3

Gestión del cliente	Gestión de los posts	Gestión de los documentos
Visualizar progresión de cliente	Visualizar posts	Añadir documentos
Visualizar las actividades del cliente	Visualizar detalles de post	Visualizar documentos generales
	Editar post	Eliminar documentos
	Eliminar post	Visualizar documentos personalizados
	Añadir post	Eliminar documentos personalizados
	Ordenar posts	

Tabla 6. Casos de uso 2/3

Gestión de la imagen	Gestión de la comunicación	Gestión de la progresión
Cambiar imagen de perfil	Enviar mensaje	Visualizar progresión
	Hablar con entrenador	Añadir progresión
	Hablar con cliente	

Tabla 7. Casos de uso 3/3

Como ejemplo, en la Ilustración 12 se mostrará el paquete de 'Gestión de los posts' y el caso de uso que le pertenece 'Ordenar posts'.

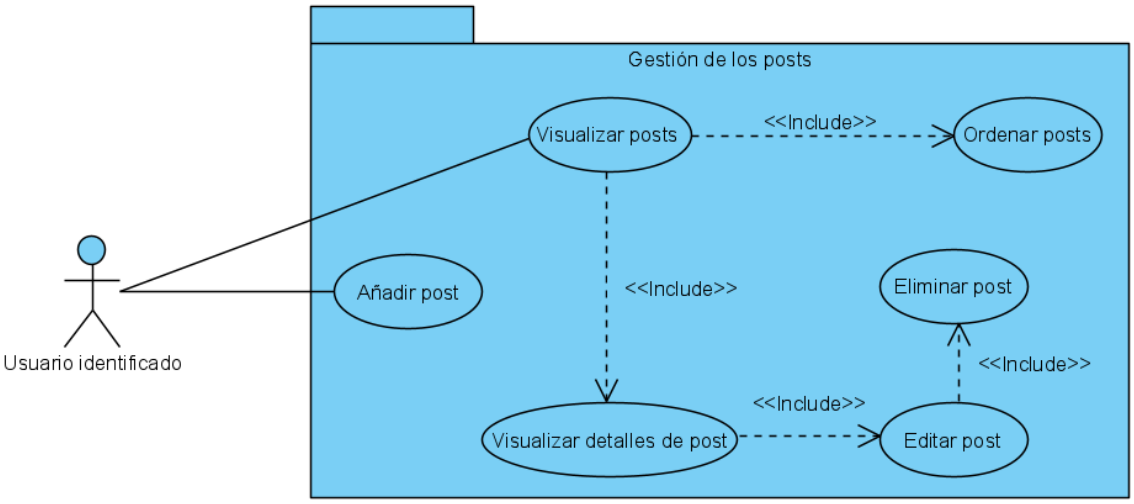


Ilustración 12. Paquete de Gestión de los posts

UC-0018	**Ordenar posts**	
Versión	1.0 (13/05/2021)	
Autores	• Cano Belamendia, Marta	
Fuentes	• De Paz Santana, Juan Francisco • Villarrubia González, Gabriel	
Dependencias	• [IRQ-0001] Información sobre usuarios • [IRQ-0004] Información sobre sala de posts	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario identificado solicite ordenar los post.*	
Precondición	Listado de posts disponible.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario identificado (ACT-0004) *pulsa la opción de ordenar*
	2	El sistema *muestra las opciones de ordenación.*
	3	El actor Usuario identificado (ACT-0004) *elige el nuevo método de ordenación.*
	4	El sistema *ordena los post según la nueva elección.*
	5	El sistema *muestra el listado de posts en la nueva ordenación.*
Postcondición	Listado de posts ordenados según la elección del usuario identificado.	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 8. Caso de uso - Ordenar posts

5.3.5. Matrices de rastreabilidad

Para relacionar los requisitos entre sí, se incluyen las matrices de rastreabilidad. La primera relaciona los casos de uso y los requisitos de información mientras que la segunda los objetivos y los requisitos de información.

Las tablas de las matrices de rastreabilidad se encuentran al final del Anexo II.

5.4. Análisis

En la fase de análisis se realizó la descomposición en paquetes, separando y refinando las clases de análisis y las funcionalidades del sistema, y de ese modo conseguir una visión más próxima al producto final.

Este apartado llega a obtener un modelo del sistema describiendo el problema de forma clara y concisa, modulando también la arquitectura a seguir. Como último paso, los casos de uso serán presentados como diagramas de secuencia.

Una visión más profunda de este apartado se puede encontrar en el Anexo III.

5.4.1. Modelo de dominio

En el modelo de dominio se recogen las representaciones conceptuales de las clases, que se obtienen a partir de los requisitos de información. Estas clases pueden sufrir cambios a medida que el proyecto evoluciona ya que el análisis se centra en el dominio del problema y no en la solución.

La Ilustración 13 que aparece a continuación muestra el modelo de dominio de este proyecto.

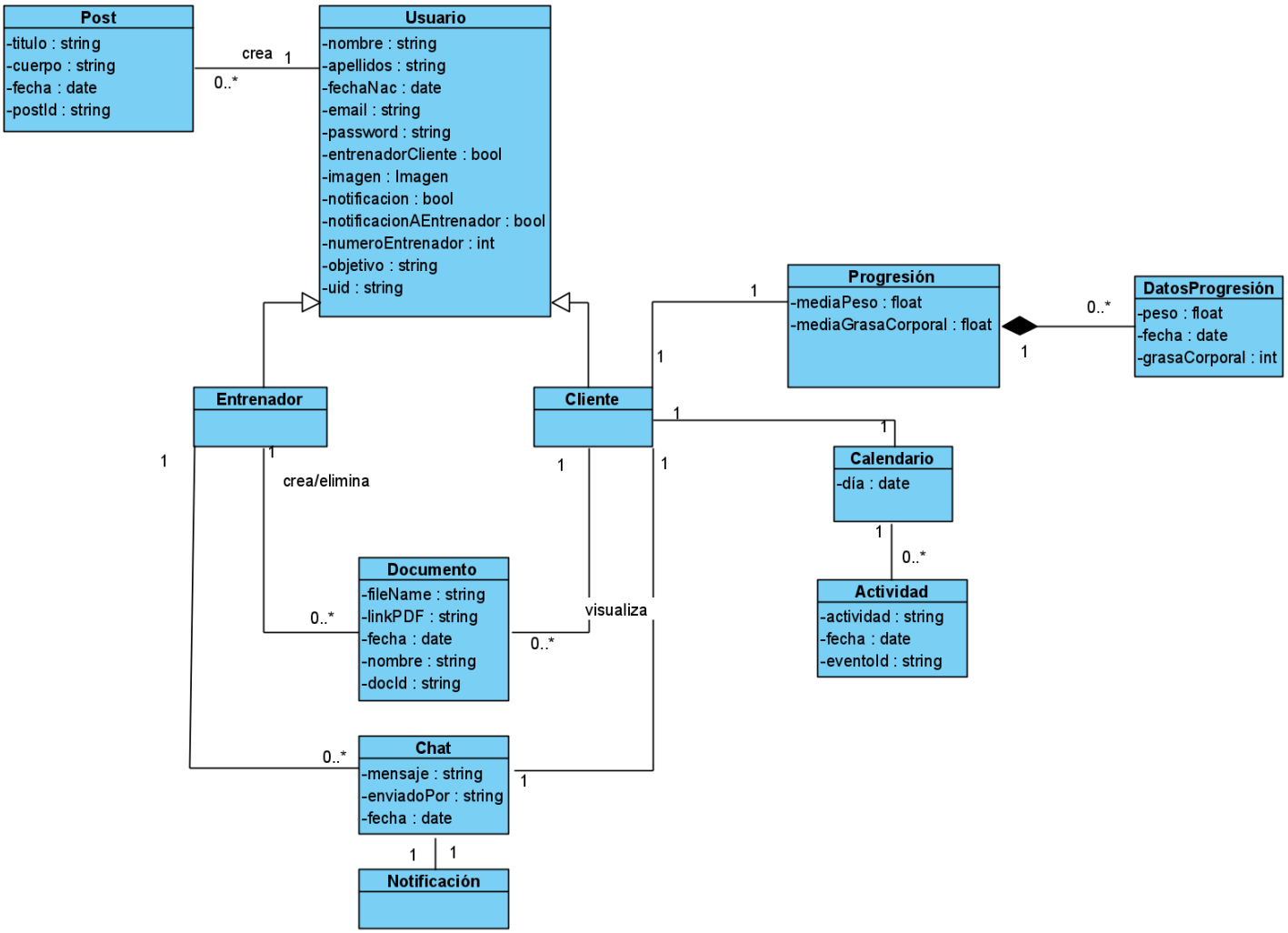


Ilustración 13. Diagrama de clases del análisis

5.4.2. Paquetes de análisis y arquitectura

Los paquetes de análisis contienen representaciones visuales de los objetos que toman parte en la realización de un caso de uso. Estos se dividen en tres objetos: interfaces, que interactúan con usuarios o servicios externos; controladores, los que procesan peticiones y sirven como puente entre las interfaces y entidades; y las entidades, los objetos que representan datos del sistema.

En la Ilustración 14 se evidencia uno de esos paquetes, el representante a los usuarios.

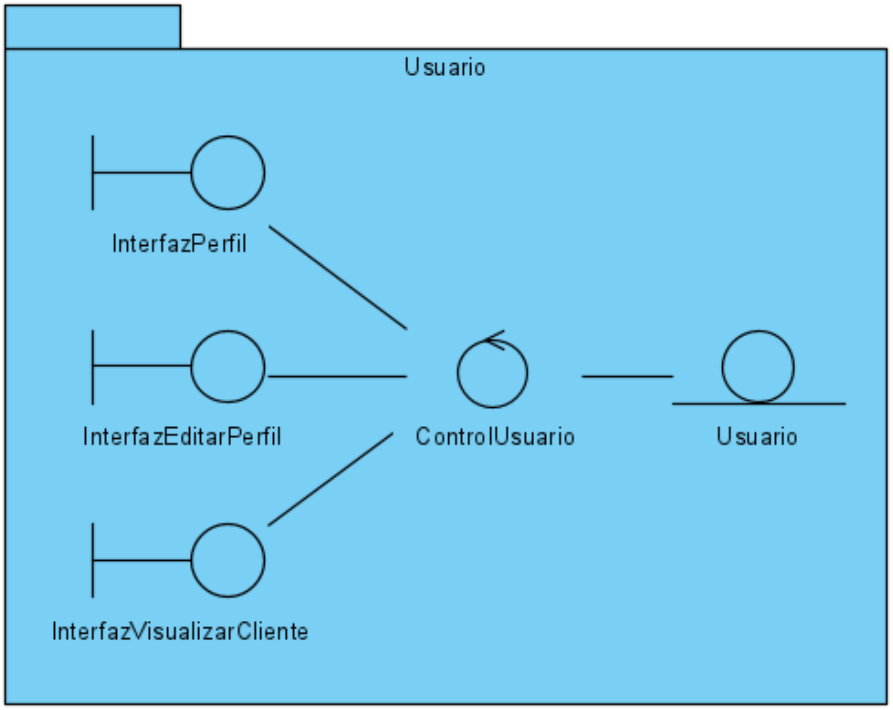


Ilustración 14. Paquete de análisis - Usuario

Estos paquetes junto a las dependencias que los unen sirven para mostrar una primera aproximación de la arquitectura que tendrá el sistema, Ilustración 15, la cual se adjunta en la página siguiente.

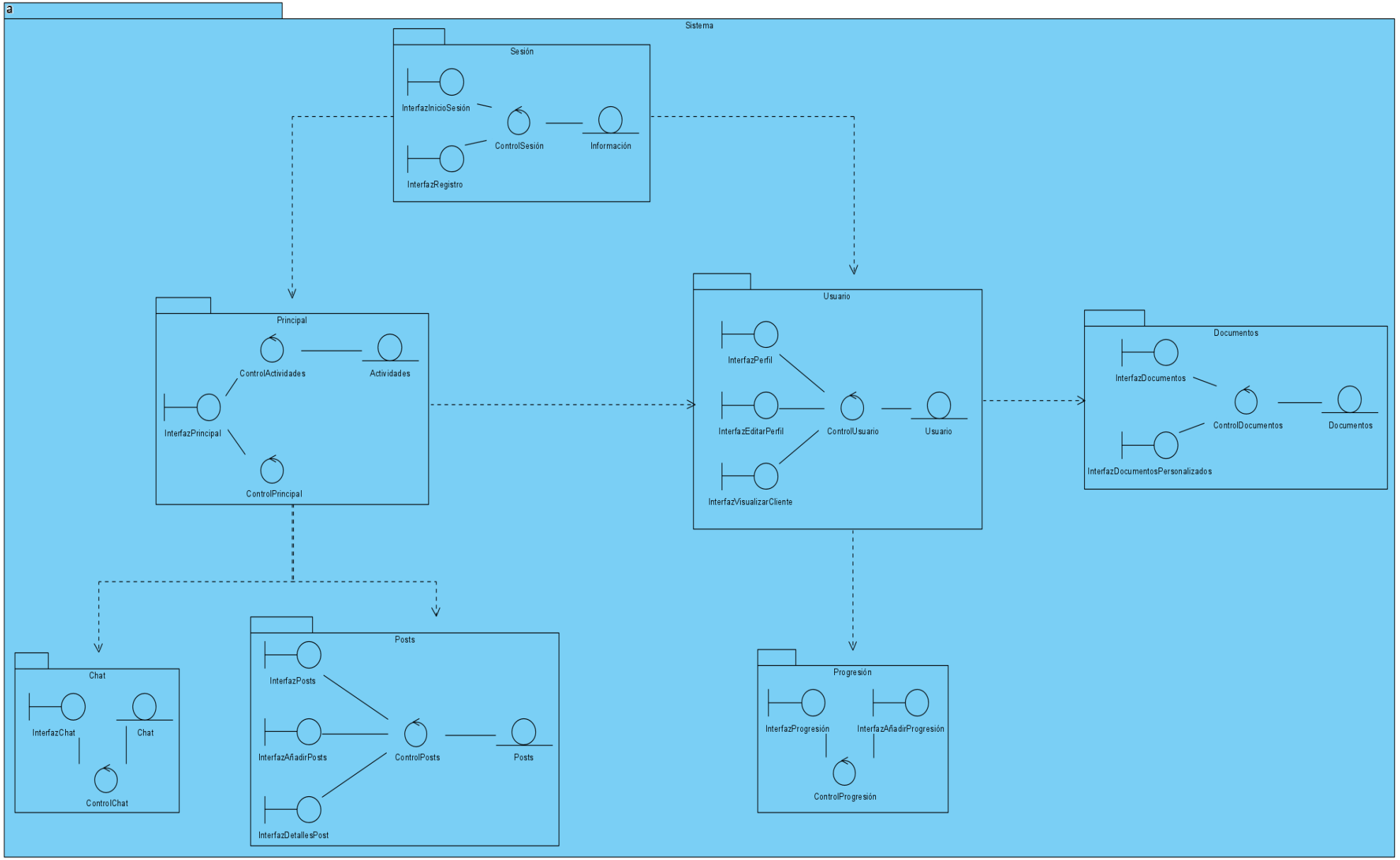


Ilustración 15. Vista de arquitectura del modelo de análisis

5.4.3. Vista de interacción

Los diagramas de secuencia sirven para mostrar cómo interactúan los objetos para llevar a cabo un caso de uso, ya que describen los mensajes intercambiados entre objetos e instancias de actores.

A continuación, en la Ilustración 16, se muestra el diagrama de secuencia para el caso de uso 'Registrarse' del paquete de 'Sesión'.

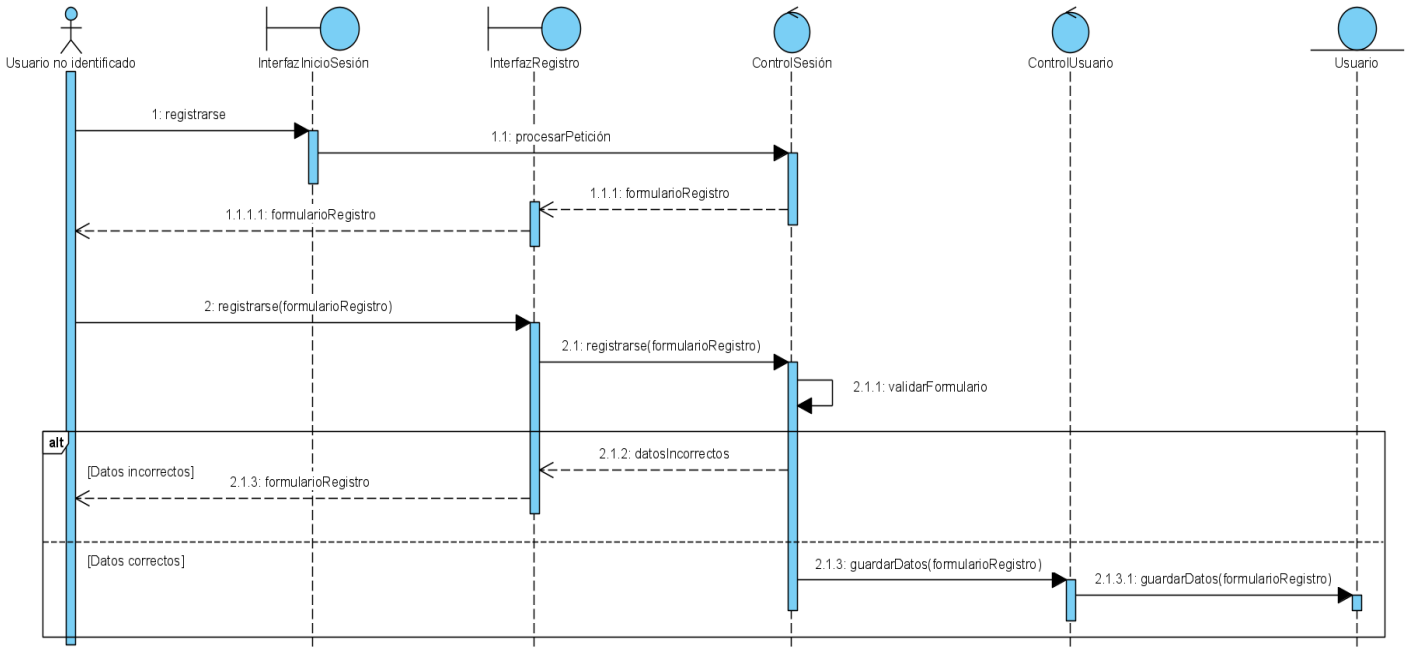


Ilustración 16. Diagrama de secuencia en análisis - Registrarse

Este diagrama muestra los mensajes que se intercambian para que un usuario no identificado pueda registrarse en el sistema. Como se puede observar, interfaces, controladores y entidades se relacionan entre sí, con las interfaces siendo quienes presentan las vistas al usuario.

5.5. Diseño

El diseño no sólo define la arquitectura final del sistema, sino que también es la última fase antes de la implementación.

En esta parte se expone el modelado del dominio de la solución; es por ello que en esta fase se refinarán los elementos identificados en iteraciones pasadas.

Para más información en detalle sobre el diseño se puede consultar el Anexo IV.

5.5.1. Diseño arquitectónico

Mientras que el análisis daba una vista de arquitectura formada por los paquetes, es en el diseño donde se elabora completamente, ya que el análisis mostraba una abstracción del patrón MCV[23], Modelo-Vista-Controlador. Se puede ver un esquema de este patrón en la Ilustración 17.

El patrón MVC tiene una gran variedad de ventajas, entre otras, que consigue que los sistemas creados siguiéndolo son más limpios, simples, y robustos. Además, facilita la realización de pruebas unitarias y permite tener múltiples vistas a partir de un mismo modelo, pudiendo así aprovechar los desarrollos y asegurando consistencia entre ellas.

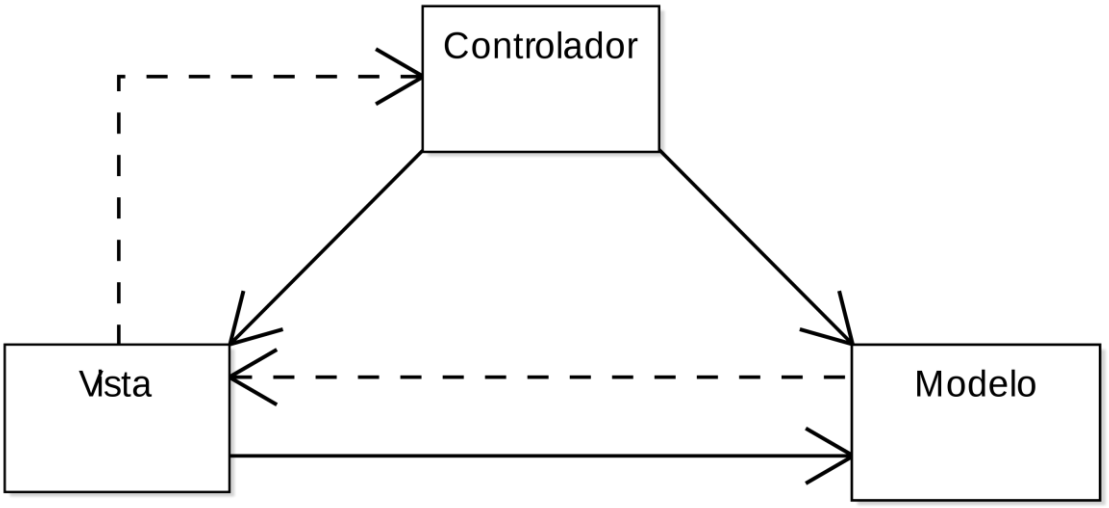


Ilustración 17. Patrón Modelo-Vista-Controlador

5.5.2. Vista de arquitectura

Como ya se ha comentado, se ha seguido el patrón MVC para construir el sistema. Unido a ello, y para que la aplicación pueda hacer uso de todas sus funcionalidades, se encuentra parte de la plataforma de Firebase con sus componentes; estas dos partes unidas formarán la vista de arquitectura general del sistema expuesta en la Ilustración 18.

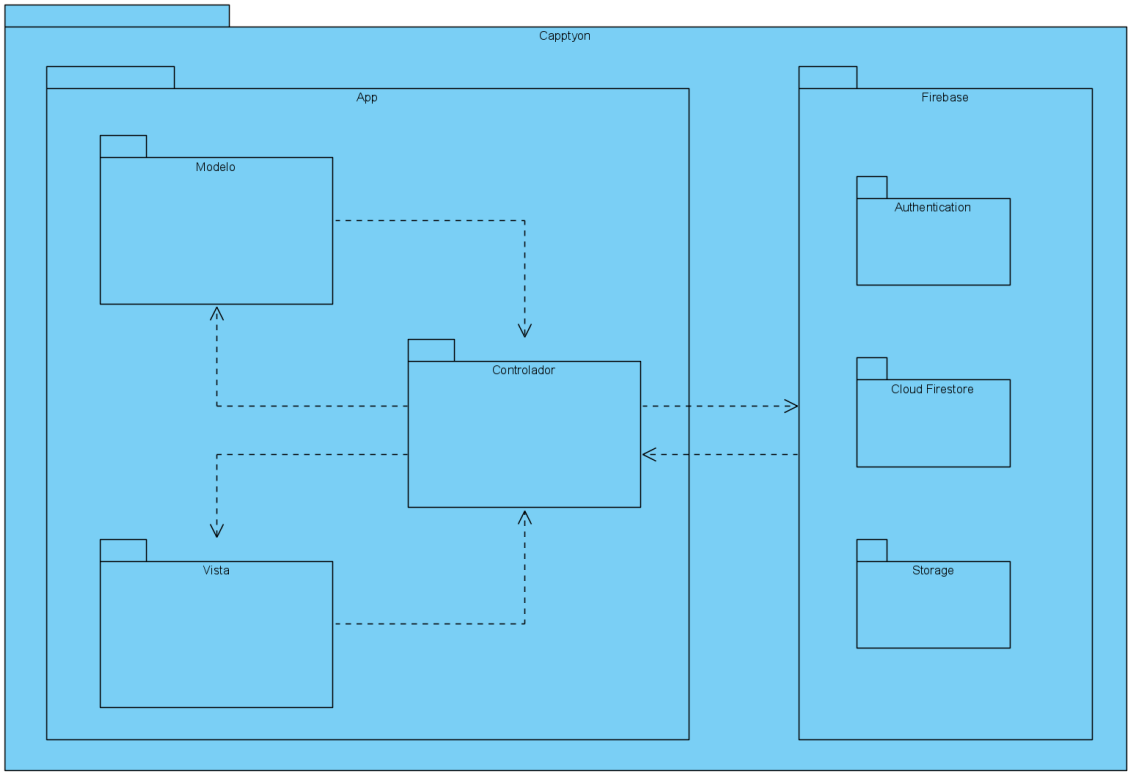


Ilustración 18. Vista de arquitectura del diseño

5.5.3. Diagrama de clases del diseño

El diagrama de clases del diseño es una implementación cercana del sistema final. En este momento del proceso se refinan las entidades mientras se tiene en cuenta la comunicación con las vistas y controladores.

A continuación se presenta en la Ilustración 19 un ejemplo de los diagramas de clase de diseño, la 'Gestión de sesión'. El resto de diagramas siguen un esquema similar y se encuentran, como se ha comentado anteriormente, en el Anexo IV.

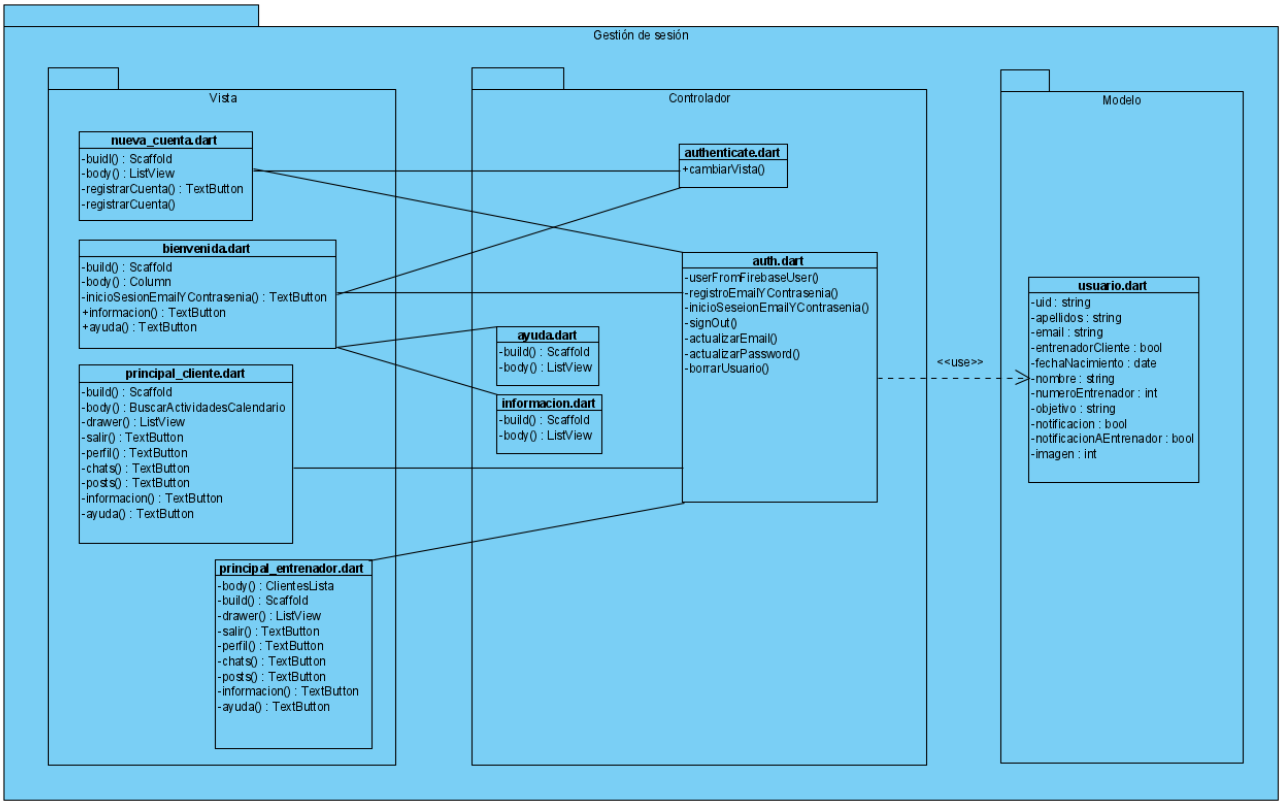


Ilustración 19. Diagrama de clases - Gestión de sesión

5.5.4. Realización de los casos de uso

La realización de los casos de uso en el diseño se aproxima a la realidad de la implementación, ya que incluye llamadas de métodos reales y los componentes que los forman. Al igual que en el análisis, se representen mediante diagramas de secuencia.

En la siguiente imagen (Ilustración 20) se muestra la evolución del caso de uso 'Registrarse' expuesto en la parte de análisis.

Se puede observar que en este momento se especifican clases y métodos concretos a implementar posteriormente.

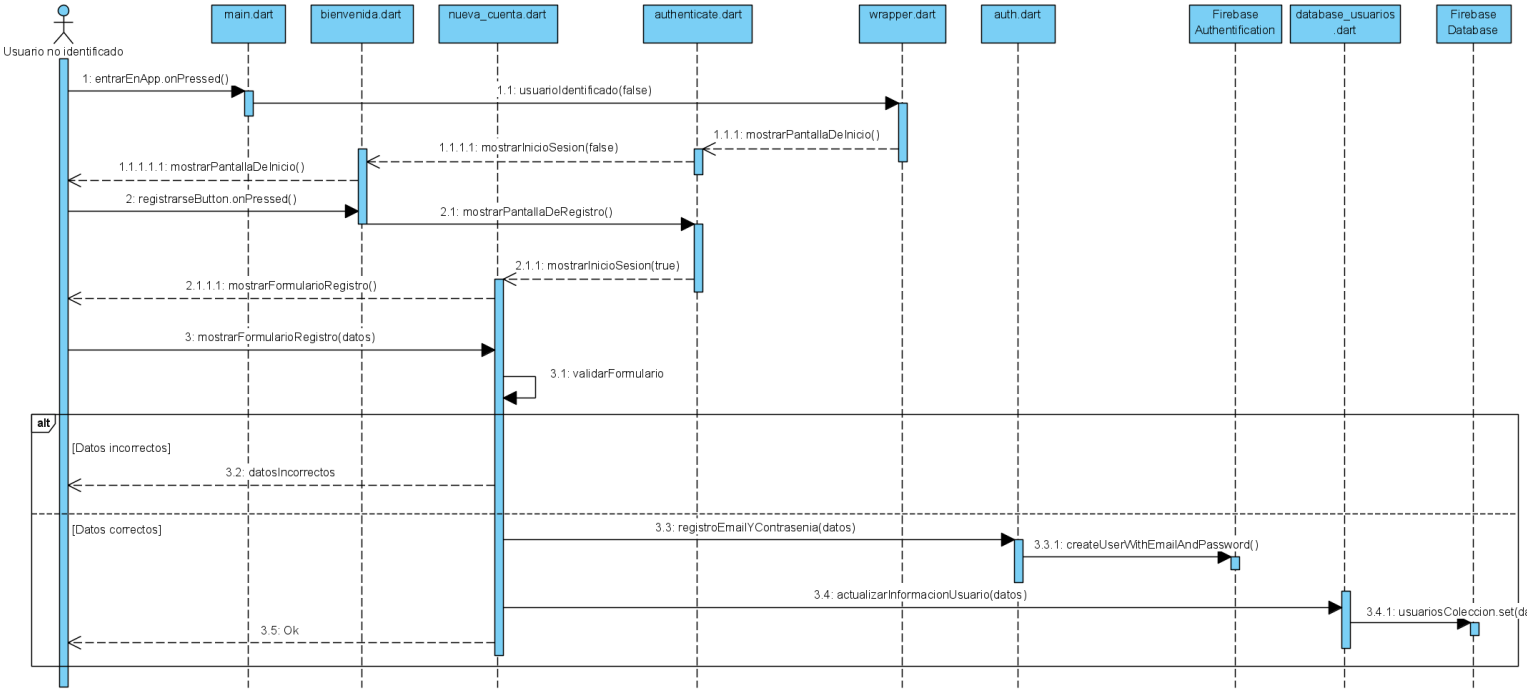


Ilustración 20. Diagrama de secuencia en diseño - Registrarse

5.5.5. Diseño de datos

La base de datos escogida para almacenar la información de la aplicación ha sido Firebase Firestore. En ella, los datos se guardan como colecciones de documentos de manera similar a JSON, pero también permite construir una fácil jerarquía con las subcolecciones dentro de los documentos.

En la Ilustración 21 que sigue, se puede apreciar la estructura general utilizada y a continuación una vista más profunda de la colección de los usuarios.

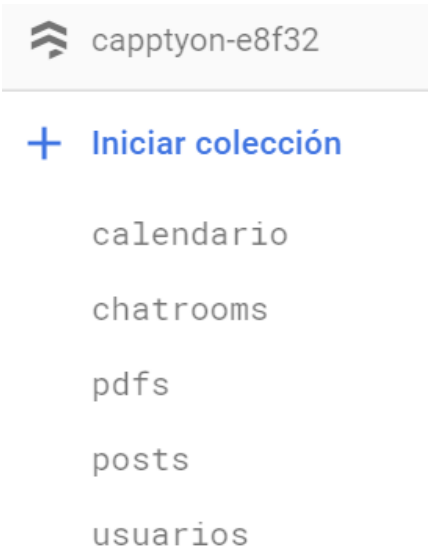


Ilustración 21. Diseño de la base de datos

En la Ilustración 22 se puede ver la colección de 'usuarios', que contiene en su interior documentos relativos a los usuarios registrados en el sistema.

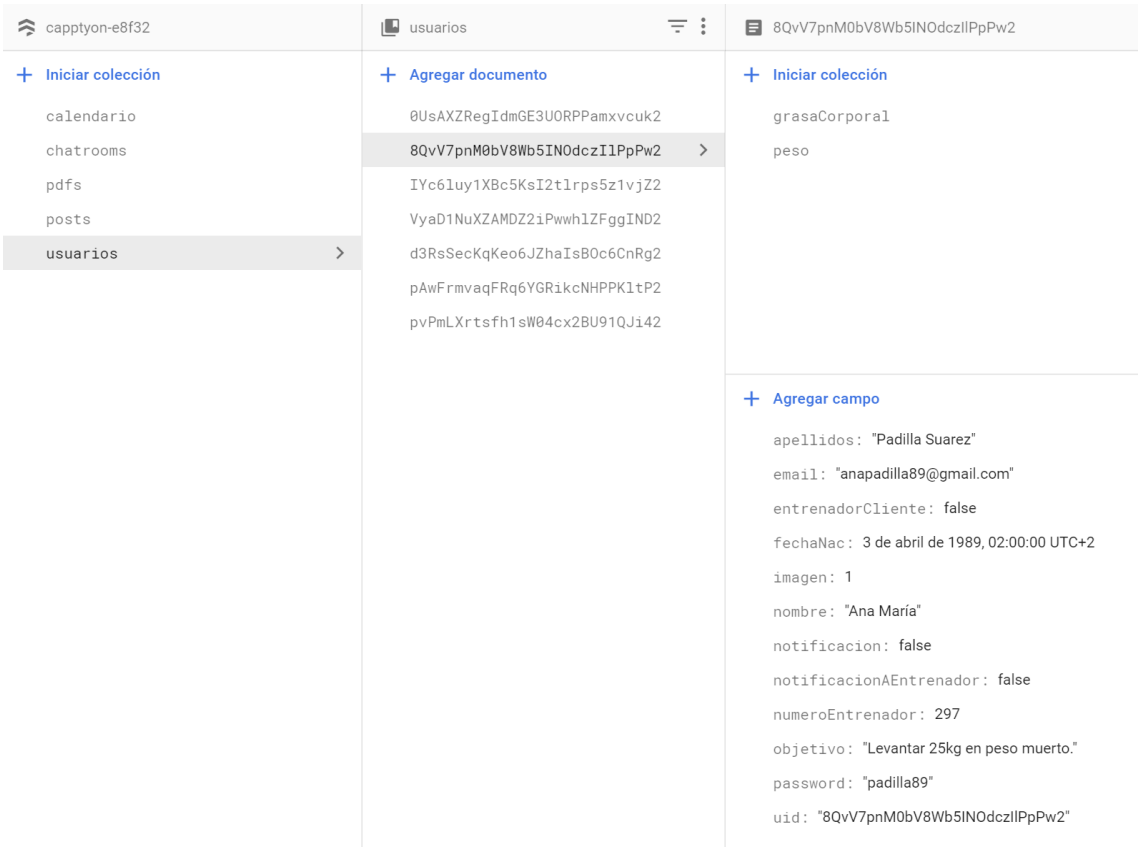


Ilustración 22. Diseño de la base de datos - Documento usuarios

5.5.6. Modelo de despliegue

El modelo de despliegue representa los distintos nodos y conexiones que forman el sistema completo, así como la distribución de los componentes físicos.

A un lado se tiene el dispositivo móvil para ejecutar la aplicación y por otro lado, unido mediante la conexión a internet, se encuentra la plataforma Firebase, que funciona en la nube.

En la Ilustración 23 se puede ver el esquema de esta aplicación.

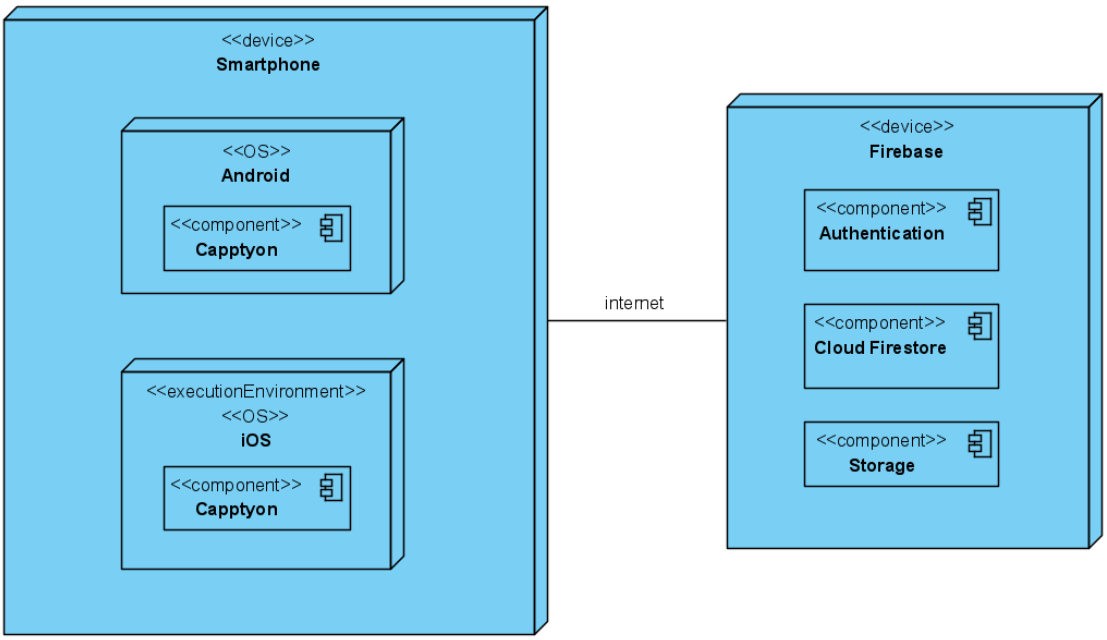


Ilustración 23. Modelo de despliegue

5.6. *Código fuente de la aplicación*

Como se ha comentado anteriormente, para el desarrollo de la aplicación se ha utilizado el entorno de Android Studio. Gracias a la extensa documentación de Flutter, se encuentran en su página las instrucciones para su descarga e instalación. En la Ilustración 24 se puede ver las instrucciones ofrecidas.

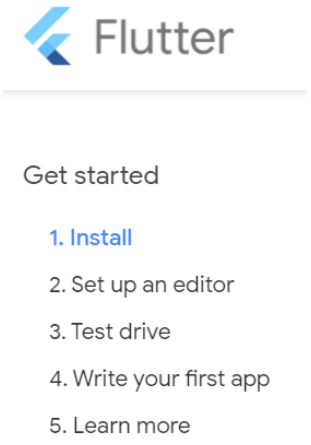


Ilustración 24. Inicio en Flutter

Lo primero, hay que descargar y extraer la última versión del SDK (*Software Development Kit)* de Flutter, extraerlo en un sitio adecuado en el ordenador, preferentemente en uno en el que no vaya a pedir permisos de administrador, y agregar la variable PATH del entorno para poder usarlo desde cualquier sesión del terminal.

Con la ejecución de *>flutter doctor* se puede ver si falta algo por instalar o si ha habido algún error. En la Ilustración 25 puede verse la salida correcta si todo está bien.

```
C:\Windows\System32>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 2.0.3, on Microsoft Windows [VersiÃ³n 10.0.19041.1052], locale es-ES)
[√] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[√] Chrome - develop for the web
[√] Android Studio (version 4.1.0)
[√] Connected device (2 available)

• No issues found!
```

Ilustración 25. Salida correcta de 'flutter doctor' en terminal

5.6.1. Configurar la plataforma

A continuación hay que instalar la plataforma deseada, Android Studio. Se debe configurar siguiendo las instrucciones, y otra ejecución de *flutter doctor* determinará si hay errores o no. Como la captura de más arriba, sin errores, se puede continuar.

Posteriormente se deben configurar los emuladores de dispositivos físicos para poder ejecutar las pruebas de las aplicaciones en ellos. Android Studio ofrece un catálogo de dispositivos virtuales desde teléfonos hasta televisiones o tablets, pero en este caso se configurarán únicamente para móviles.

5.6.2. Configurar el editor

Flutter sugiere ciertos editores para crear las aplicaciones. En este caso se ha escogido Android Studio, y para ello se siguen las instrucciones detalladas.

Después de instalar Android Studio, realizado en el apartado anterior, la instalación de Flutter es sencilla; basta con dirigirse al *Marketplace* en la opción de *Plugins* y seleccionar Flutter. En la Ilustración 26 aparecen las instrucciones de la página de Flutter que hay que seguir y en la Ilustración 27 instalados en este proyecto.

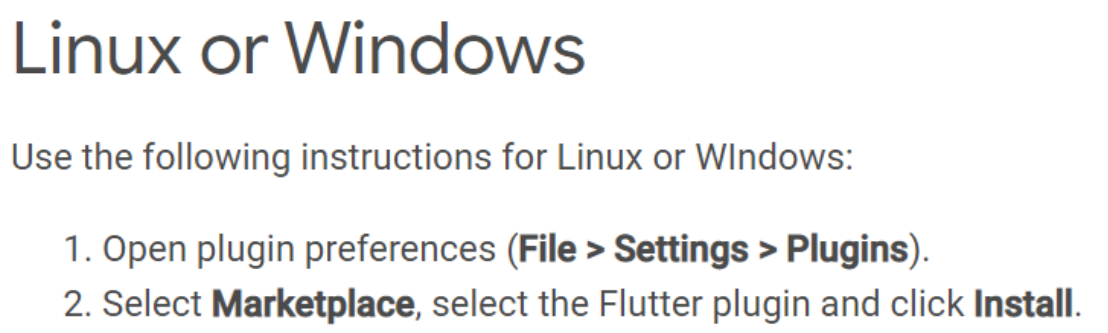


Ilustración 26. Instrucciones para instalación de plugins

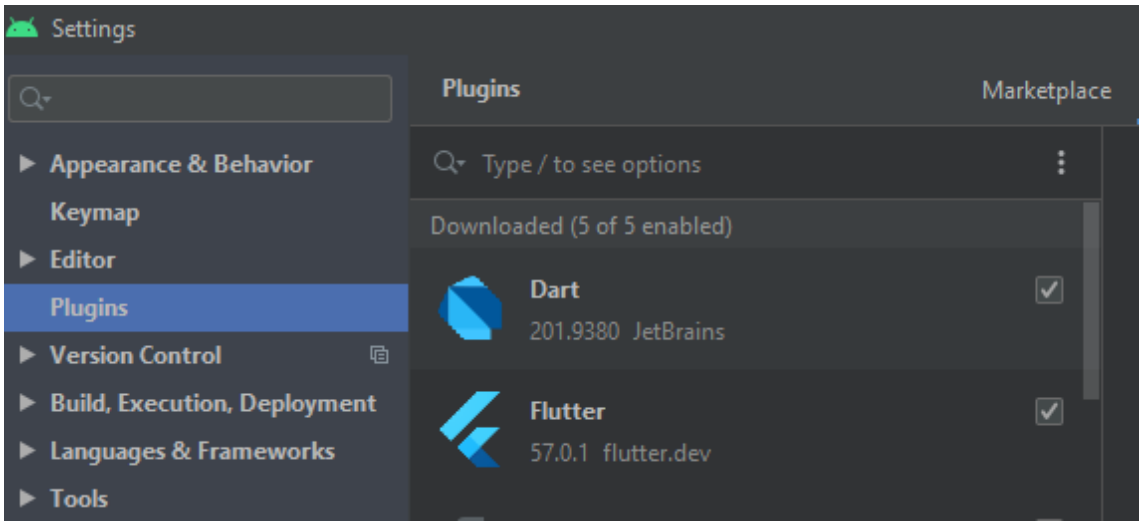


Ilustración 27. Plugins instalados correctamente

Con ello ya estarían Flutter y Dart instalados.

5.6.3. Primera aplicación

Creando un nuevo proyecto en Flutter y añadiendo un nombre, se mostrará una aplicación de demostración simple. Esta cuenta con un botón en la esquina inferior derecha, y en medio de la pantalla un mensaje estático y un número que cambiará con el número de pulsaciones en el botón. Es simplemente una aplicación contador que muestra el número de veces que se ha hecho click en el botón.

En la Ilustración 28se puede observar la barra superior con las distintas funciones, donde aparece el dispositivo usado, el código principal, el botón de ejecutar, de debuggear, o el de parar.

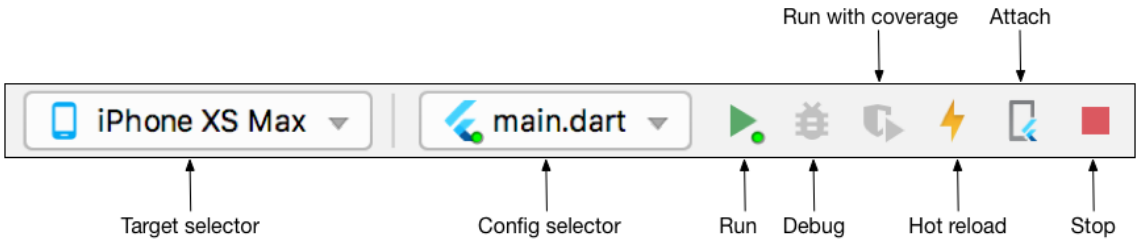


Ilustración 28. Barra de opciones en Android Studio

Una de las opciones más interesantes de esta ilustración es el botón de *Hot reload*, que permite recargar el código sin necesidad de reiniciar la aplicación o compilar de nuevo el código para aplicar rápidamente cualquier cambio en este.

5.6.4. En la aplicación

En este proyecto, la división del código fuente (Ilustración 29) se ha llevado a cabo en función de los usuarios que interactuarán en el sistema, es decir, usuarios registrados, clientes, y entrenadores.

Por ello existen las carpetas de *cliente*, *entrenador*, y *compartido*, que contendrán las funcionalidades especializadas para los roles Cliente, y Entrenador, mientras que *compartido* contendrá aquellas funcionalidades ejecutadas por ambos.

La carpeta *paginas* contendrá los archivos necesarios para las acciones realizadas por un usuario no identificado, y la carpeta *servicios* contiene los ficheros de comunicación con la base de datos.

Los otros dos archivos al mismo nivel que las carpetas servirán para hacer la primera bifurcación si se trata de un usuario identificado, con lo que se le presentará su página privada, o si se trata de un usuario no identificado, lo que mostrará la página de inicio de sesión.

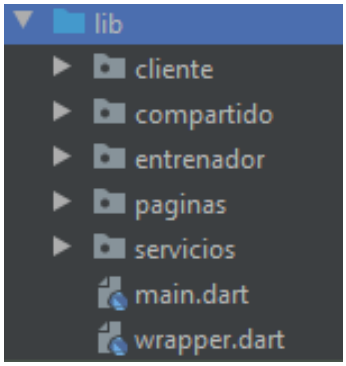


Ilustración 29. Estructura del código en Android Studio

Para más detalles se puede consultar el Anexo V de documentación técnica.

Por otro lado se encuentra el archivo *pubspec.yaml*. Este archivo contiene información sobre los metadatos de la aplicación como el nombre del proyecto, su versión, su entorno, tipografías, y lo más importante, las dependencias.

Las dependencias son paquetes específicos a Flutter que, entre otras cosas, ayudan a incrementar las funcionalidades de la aplicación.

Estos paquetes pueden obtenerse desde la página de pub.dev, que contiene un extenso catálogo de paquetes creados por la comunidad. En la Ilustración 30 se puede ver un ejemplo de los paquetes disponibles.

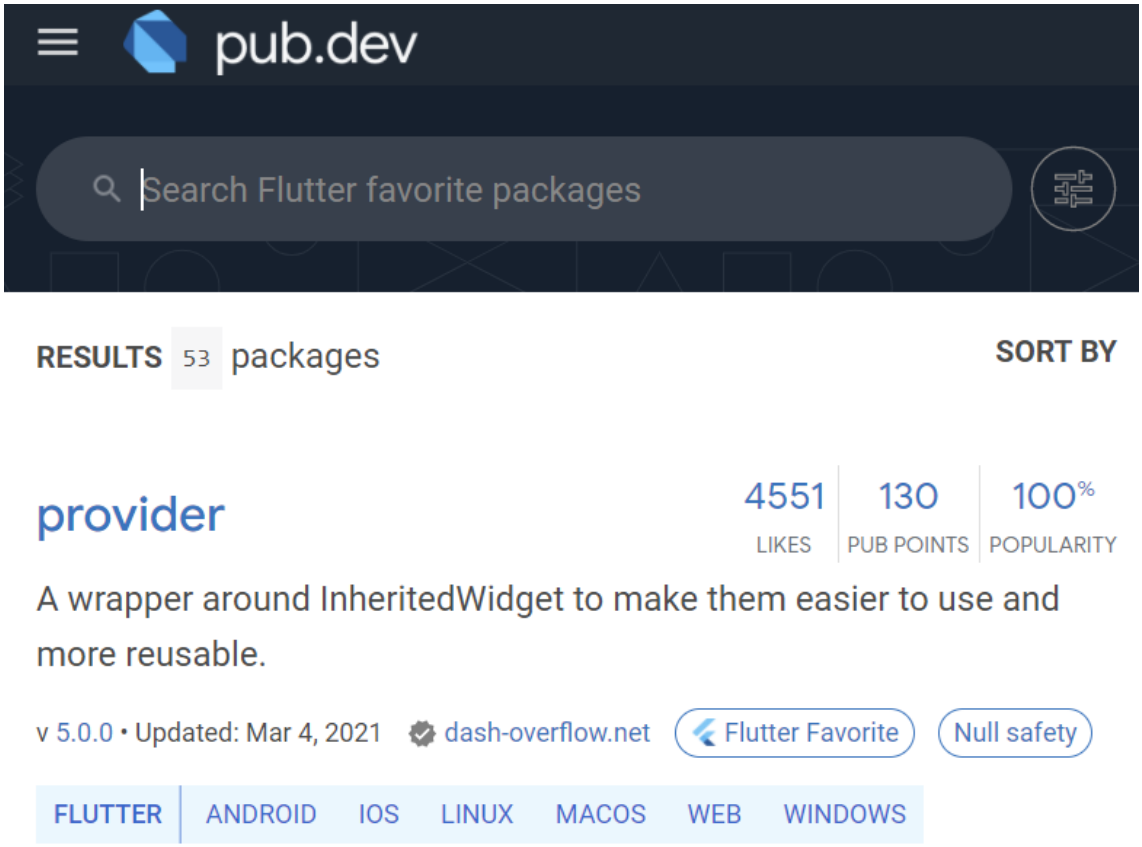


Ilustración 30. Paquetes de Dart en pub.dev

Para el desarrollo de esta aplicación se pueden ver en la Ilustración 31 las dependencias utilizadas.

```
dependencies:
  passwordfield: ^0.0.82
  flutter:
    sdk: flutter
  flutter_localizations:
    sdk: flutter
  intl: ^0.17.0
  cloud_firestore: ^1.0.3
  email_validator: ^2.0.1
  firebase_auth: ^1.0.1
  firebase_core: ^1.0.2
  flutter_spinkit: ^5.0.0
  flutter_sparkline: ^0.1.0
  provider: ^5.0.0
  table_calendar: ^3.0.0
  file_picker: ^3.0.1
  firebase_storage: ^8.0.2
  flutter_pdfview: ^1.1.0
  path_provider: ^2.0.1
  flutter_phoenix: ^1.0.0
  flutter_slidable: ^0.6.0
```

Ilustración 31. Dependencias de Capptyon

Cada línea del código consta del nombre del paquete seguido de dos puntos y la versión instalada. Estos paquetes han ayudado a la realización del proyecto, proporcionando funciones para la comunicación con los servicios Firebase (*cloud_firestore, firebase_auth, firebase_core, firebase_storage*) o efectos visuales de carga de datos (*flutter_spinkit*).

Uno de los aspectos más rompecabezas en la aplicación surgió a la hora de eliminar usuarios. No a la eliminación per se, sino que una vez eliminado el usuario, la aplicación se quedaba en un estado de carga ya que se encontraba en una zona de usuario sin usuario.

Por fortuna, se encontró solución en la dependencia de *flutter_phoenix* que recarga la aplicación a nivel de aplicación, no a nivel de sistema operativo, eliminando el estado anterior. Por consiguiente, la aplicación 'se olvidaba' de buscar el usuario ya eliminado y como no encontraba ninguno, mostraba entonces la página de inicio sesión.

Otro de los paquetes importantes ha sido *provider*, que permite Widgets de nivel inferior puedan acceder a datos de los de niveles superiores sin que estos tenga que explícitamente pasar los datos.

5.7. Funcionalidades de la aplicación

En este apartado se resumen las funcionalidades de la aplicación Capptyon.

Tiene dos funcionalidades principales: la primera, por parte del cliente, es poder llevar la evolución de su progresión respecto a su peso y grasa corporal; la segunda, respecto al entrenador, poder visualizar dicha progresión y actividades realizadas de modo en que pueda modificar los entrenamientos.

5.7.1. Registro e inicio de sesión

Al registrarse en la aplicación, se le pedirá al usuario que introduzca sus datos básicos. Una cuenta de Entrenador o de Cliente se diferencia en una mera opción, en la cual si el usuario escoge la opción 'Cliente' tendrá que introducir un número de entrenador válido (Ilustración 32), mientras que si escoge la de 'Entrenador', se le asignará un número de entrenador automáticamente (Ilustración 33).

Es este número el que deberá facilitar a sus futuros clientes.

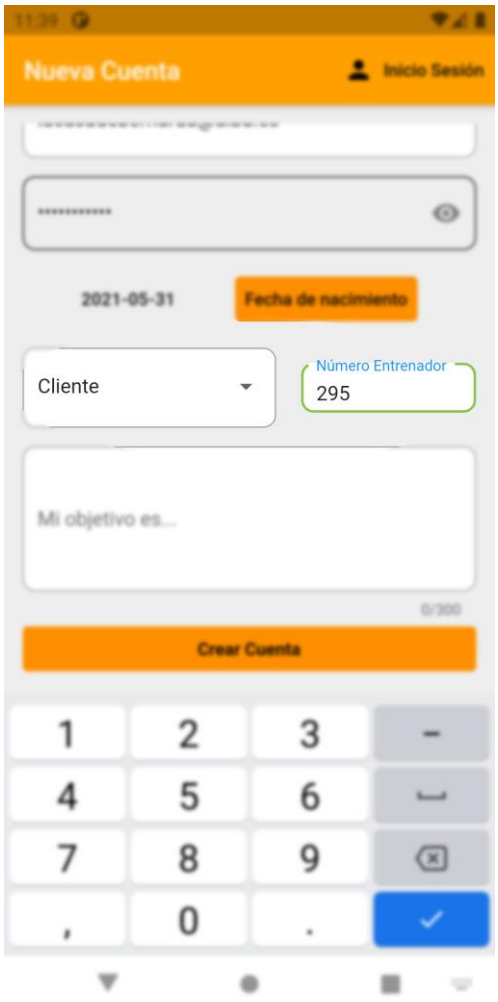


Ilustración 32. Nueva Cuenta - Cliente

Ilustración 33. Nueva Cuenta - Entrenador

Para iniciar sesión será suficiente con introducir el correo electrónico y la contraseña elegida.

5.7.2. Gestión del perfil

Todos los usuarios identificados podrán cambiar la imagen de su menú de una selección rotativa (Ilustración 35), y accediendo a su perfil podrán ser capaces de modificar la mayoría de los datos introducidos (Ilustración 34). Entre los que no pueden ser modificados estará el número de entrenador, ya que es imprescindible para el correcto funcionamiento de la aplicación.

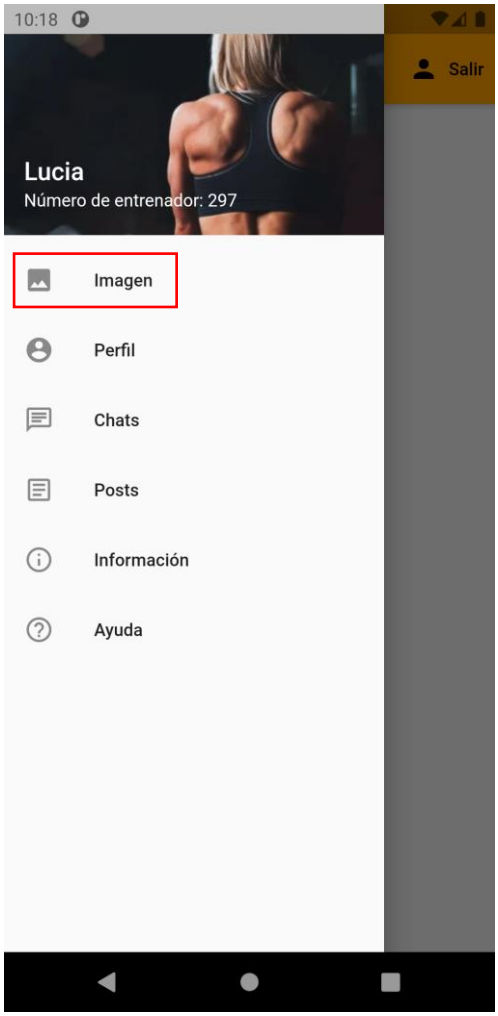


Ilustración 35. Menú principal - Imagen y Perfil

Ilustración 34. Página de perfil

5.7.3. Documentos

Desde la misma página de su perfil, los usuarios podrán visitar la página de documentos donde sólo el entrenador podrá subir o eliminar documentos. Aquí, los clientes tienen dos opciones: visualizar los documentos personalizados, Ilustración 36, documentos que el entrenador sube a cada uno de sus clientes accesibles únicamente por ellos, o documentos generales, (Ilustración 37) que el entrenador pone a disposición de todos sus clientes.

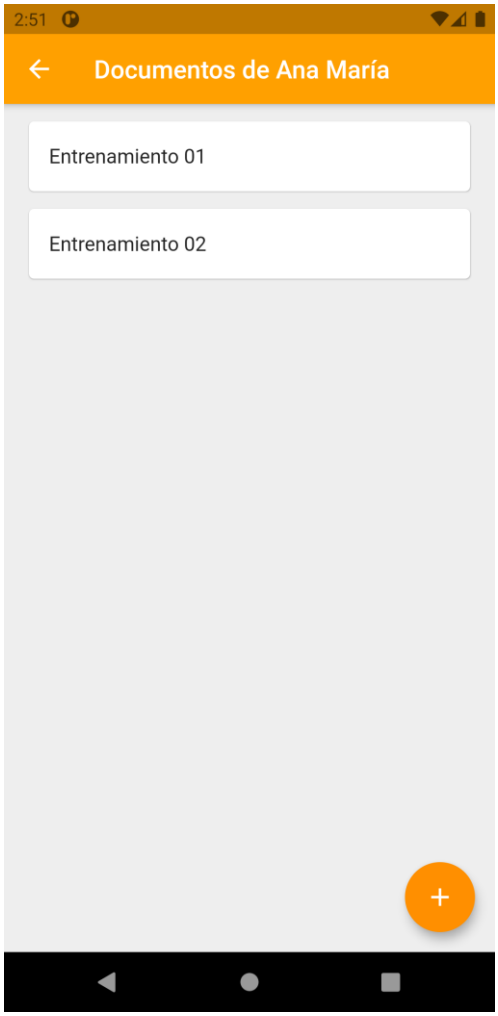


Ilustración 36. Documentos personales

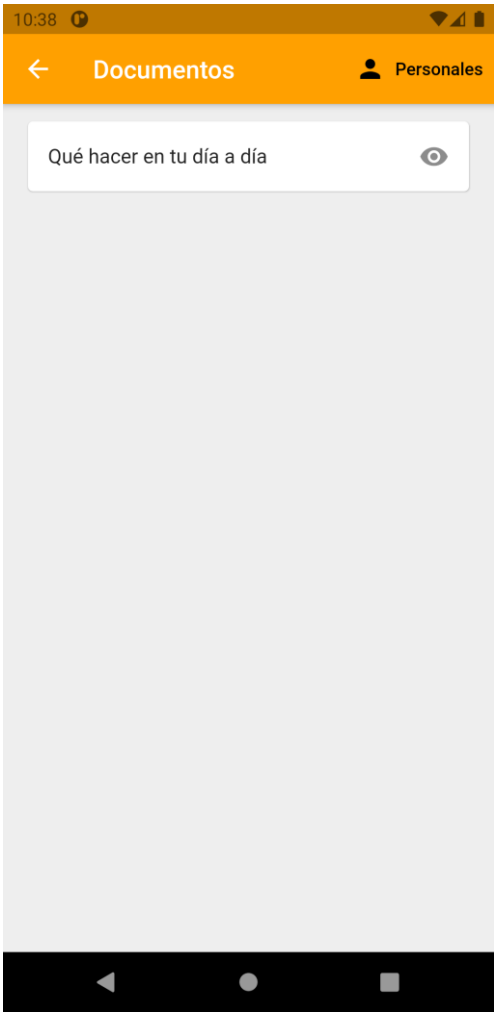


Ilustración 37. Documentos generales

5.7.3. Chat

La comunicación entre cliente y entrenador es fundamental, y por ello otra de las grandes funcionalidades. En el caso de cliente, un acceso directo (Ilustración 39) a la conversación con su entrenador (Ilustración 38) le espera en el menú de la izquierda, mientras que en el caso del entrenador (Ilustración 40), tendrá un listado de conversaciones con sus clientes (Ilustración 41), de los cuales podrá elegir la conversación a iniciar.

Una vez elegido el chat, ambos usuarios funcionarán de modo similar.

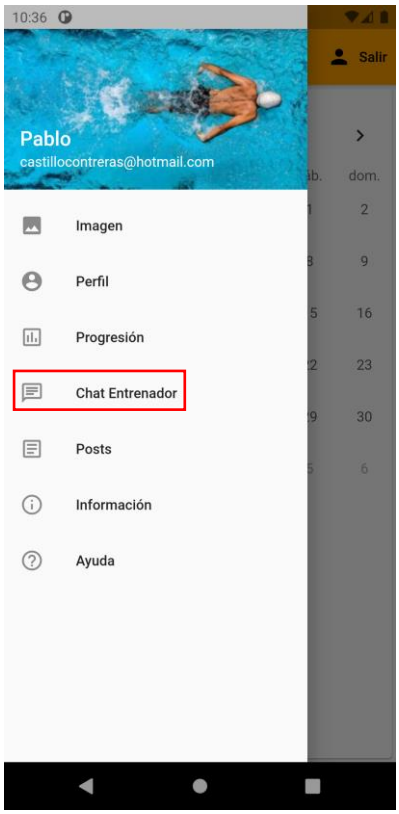


Ilustración 39. Menú principal - Chat Entrenador

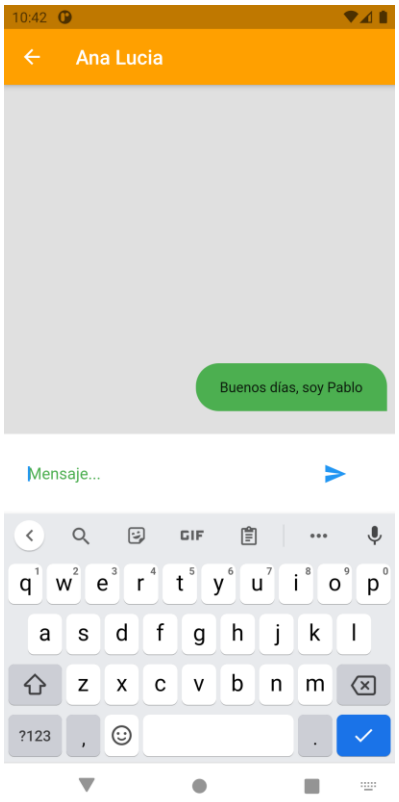


Ilustración 38. Chat privado con entrenador

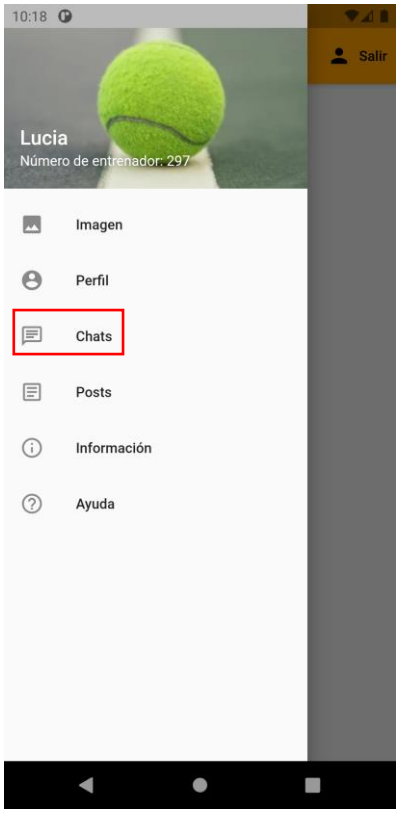


Ilustración 40. Menú principal - Chats

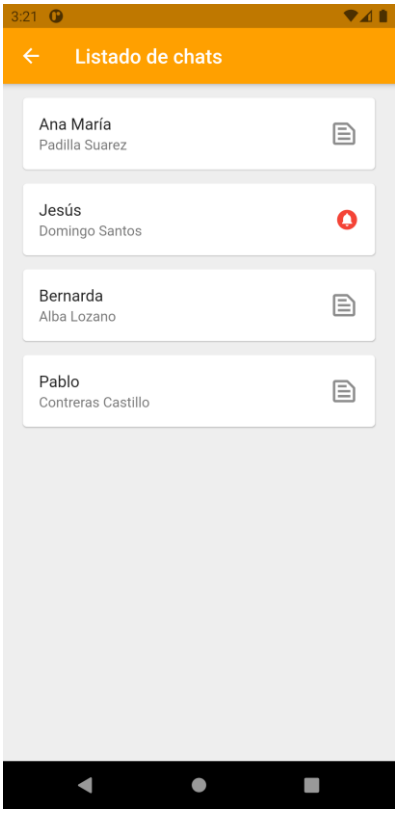


Ilustración 41. Listado de chats

5.7.4. Actividades

Otra de las funcionalidades por parte del cliente será el poder tener un calendario donde apuntar las actividades presentes, pasadas, o las futuras (Ilustración 42). Este listado de actividades podrá ser visualizado por el entrenador (Ilustración 43, Ilustración 44).

Ilustración 42. Página principal del cliente - Calendario

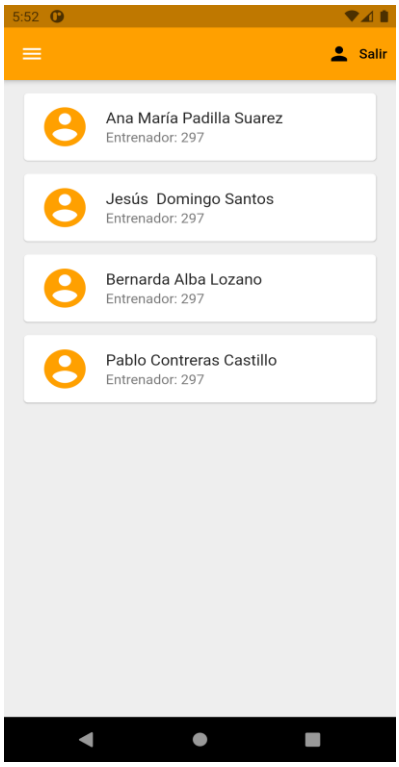


Ilustración 44. Página principal del entrenador - Listado de clientes

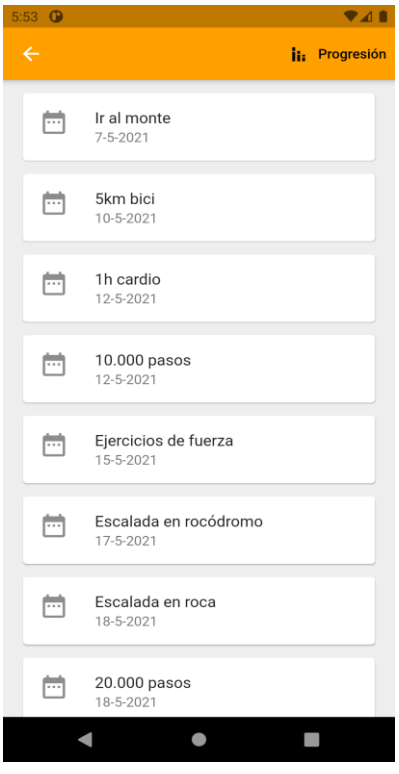


Ilustración 43. Vista de entrenador - Listado de actividades de cliente

5.7.5. Progresión

Esta es una de las principales funcionalidades de la aplicación por parte del cliente, el poder llevar su progresión y añadir nuevas entradas (Ilustración 45, Ilustración 46). El entrenador podrá visualizar esta progresión y modificar los entrenamientos a su vez (Ilustración 47, Ilustración 48).

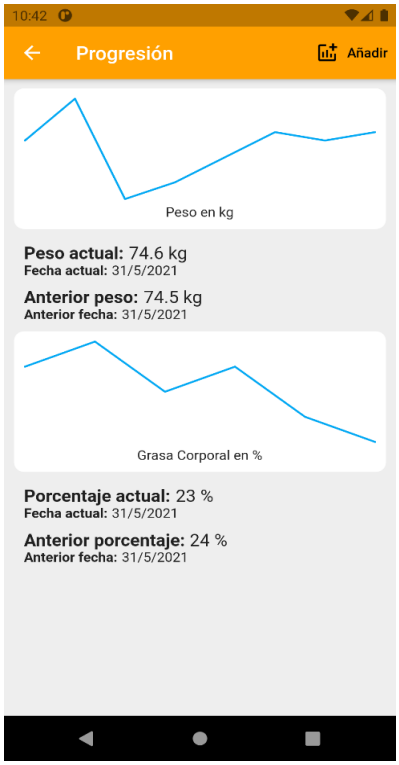


Ilustración 46. Vista del cliente - Gráficos de progresión

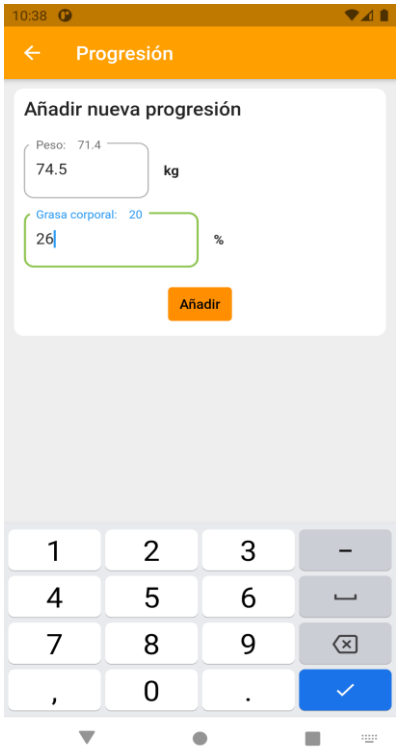


Ilustración 45. Añadir nueva progresión

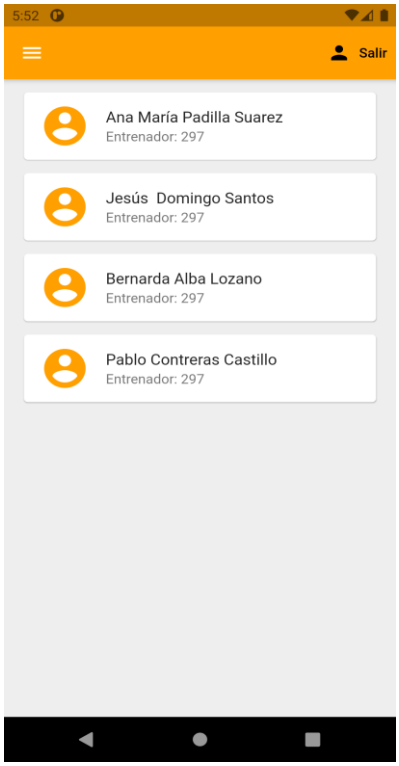


Ilustración 48. Página principal del entrenador - Listado de clientes

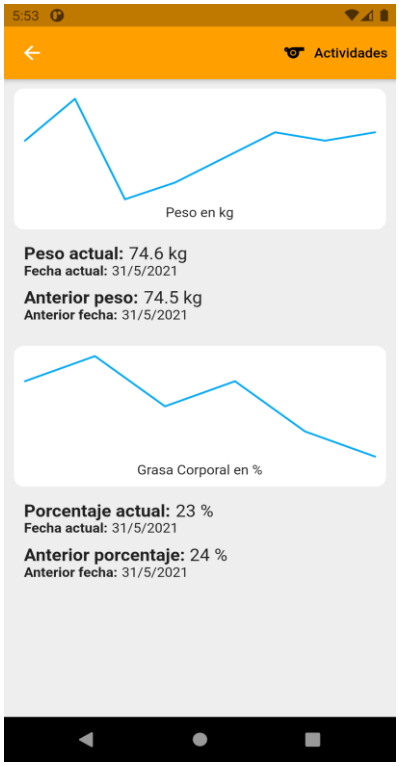


Ilustración 47. Vista del entrenador - Gráficos de progresión

5.7.6. Posts

La aplicación cuenta, además, con un apartado de posts (Ilustración 49, Ilustración 50) en la que el entrenador y todos sus clientes pueden interactuar.

A parte de poder añadir, editar, y eliminar posts, se puede también editar la cronología de los posts (Ilustración 51) para visualizar los creados por el entrenador (Ilustración 52), los clientes, o los propios. Por defecto, la cronología de estos es por fecha de creación.

Ilustración 50. Listado de posts

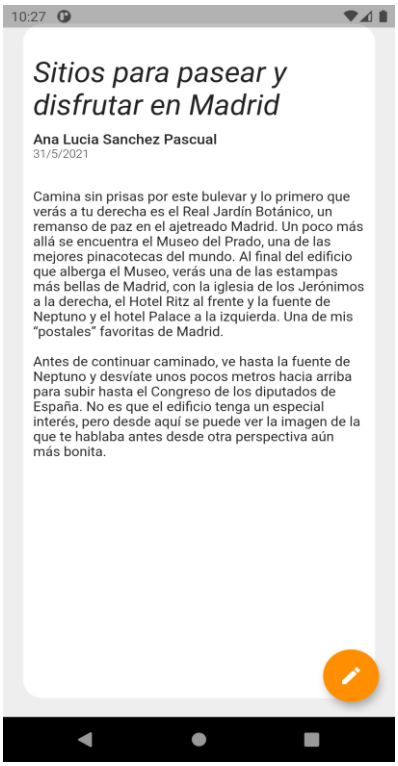


Ilustración 49. Detalles de post

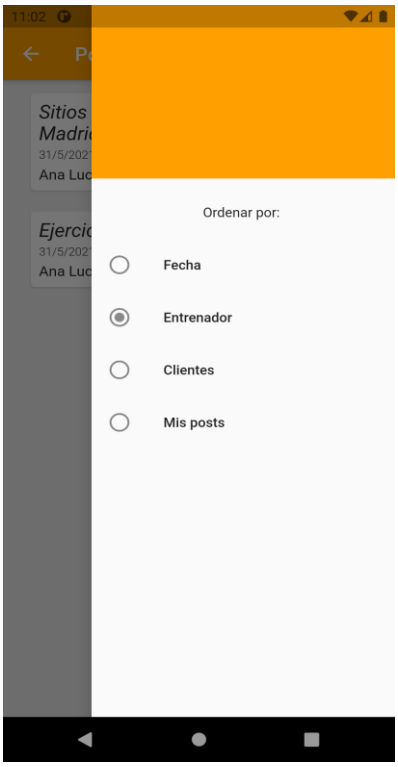


Ilustración 52. Opciones de ordenación

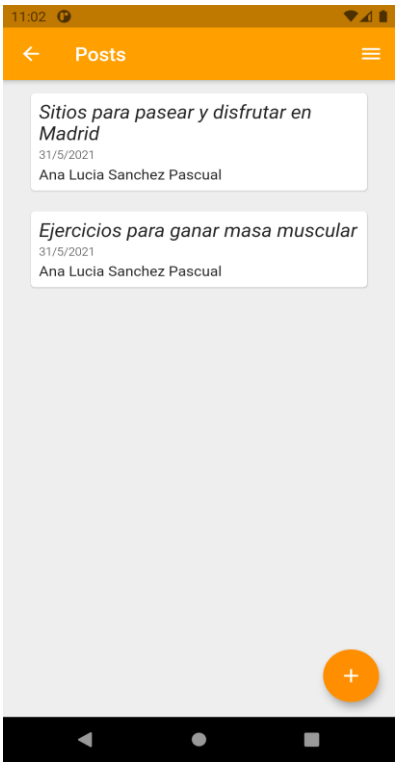


Ilustración 51. Nueva ordenación - Entrenador

5.8. Pruebas

Las pruebas de la aplicación se realizaban cada vez que se añadía una funcionalidad nueva y regularmente, para asegurarse de que todo funcionaba correctamente. Estas pruebas, a lo largo de la implementación, se hicieron en el emulador que proporciona Android Studio.

Por otro lado, una vez de que la aplicación estaba completamente terminada, se creó la apk de ella para probarla en sistemas reales y modificar pequeños errores que de otro modo no se hubieran podido encontrar.

Esto ayudó a confirmar que la aplicación puede ser usada sin mayores problemas, ya que es intuitiva y fácil de usar.

6. Conclusiones

Una vez concluido el proyecto, es posible afirmar que se han cumplido los objetivos principales propuestos.

Se planteaba la realización de una aplicación de móvil que pudiera gestionar dos tipos de usuarios, permitiéndoles actualizar sus datos en todo momento, e incluso la eliminación completa de estos si así lo quisiera el usuario. El marco de trabajo Flutter y Dart, así como las herramientas ofrecidas por Firebase, han permitido un desarrollo fluido, centrado en la escalabilidad.

De cara a los objetivos propuestos, se ha conseguido implementar una comunicación entre usuarios mediante chats privados con actualizaciones en tiempo real; además, gracias a la ayuda de Firebase Authentication, se ha conseguido un sistema de gestión de usuarios para controlar el registro y el inicio de sesión o la salida de la aplicación.

Respecto a los clientes, se ha conseguido que puedan disponer de un calendario en su página principal que sirva como diario de las actividades realizadas, y que puedan llevar una gestión de su progresión, añadiendo nuevas entradas que actualicen los gráficos cada vez que son introducidas.

Siguiendo con las funcionalidades, se ha logrado que el entrenador pueda subir documentos de dos maneras: para sus clientes en general y personalizados para cada persona. En un primer lugar, los documentos iban a estar únicamente disponibles como documentos individualizados, pero tras una investigación de prueba y error, se consiguió que la aplicación permitiera subir archivos generales, ya que hay momentos en los que la información requiere ser compartida con un mayor número de personas.

Otro de los objetivos propuestos era la creación de un apartado de microblogueo en la que clientes y entrenador pudieran crear entradas que el resto pudieran ver. En un principio, esta página no contaba con una opción que dejara ordenar los posts, pero tras una reunión con el tutor surgió la idea de cómo llevarlo a cabo.

Según la tecnología utilizada, Flutter y Dart permite desarrollar aplicaciones fácil y rápidamente. Ha sido sorprendente el poder realizar cambios en la interfaz y verlos tomar efecto gracias a su hot-reload, ya que ha salvado mucho tiempo a la hora de decidir la estética y posicionamiento de elementos en la interfaz.

Además, al ser tanto Firebase como Flutter dos plataformas o herramientas diseñadas y creadas por Google, la unión de ambas en la aplicación no ha tenido mayores problemas. Mientras que lo más complicado pudo haber sido la integración de la plataforma en el proyecto, una vez hecho eso, manejar los datos desde o hacia Firebase y sus servicios ha sido cosa sencilla. Definitivamente son opciones que contemplaré usar en el futuro ya que no sólo son cómodas de usar por todas las ventajas que ofrecen, sino que están siendo utilizadas más y más por personas en el campo de la informática.

La aplicación ha sido construida superando con creces las expectativas, y ha abierto una puerta en un sector tan aclamado como el de las aplicaciones de móvil. Ha sido, sin duda, un gran logro y una gran experiencia que ha permitido poner en práctica los conocimientos y habilidades adquiridas a lo largo de la carrera.

7. Líneas de trabajo futuras

Dado por concluido el proyecto, existen líneas de trabajo futuras para optimizar la funcionalidad de la aplicación o añadir nuevas ideas.

De cara a mejoras sobre la aplicación, una mejora futura sería la posibilidad de añadir imágenes en los posts, y que estos pudieran ser respondidos por otros clientes para así crear una especie de red social.

En el lado cliente, una mejora sería poder ver en más detalle su propia progresión; en estos momentos los únicos datos que se pueden leer son los dos últimos introducidos, y esta línea de trabajo futura permitiría visualizar los datos anteriores.

Por el lado de entrenador, se podría añadir la posibilidad de añadir actividades al calendario de sus clientes, indicadas en un color distinto, para recordar los entrenamientos o completar su deber como entrenador.

Una primera línea de trabajo para añadir nuevas funcionalidades sería integrar la herramienta de Google Google Fit para que el cliente no necesite introducir los datos de su progresión manualmente sino que se actualicen gracias a esta aplicación.

Siguiendo esa línea, se podría añadir el inicio de sesión o creación de cuenta con una cuenta existente de Google, ya que la mayoría de las personas, por no decir todas, cuentan con una cuenta de este tipo.

Otra funcionalidad que se podría añadir sería el incluir una opción de videollamada al chat para sesiones presenciales o dudas de ejercicios, así como la posibilidad de un apartado similar al de documentos donde el entrenador podría subir vídeos para que sus clientes puedan ver cómo realizar correctamente un ejercicio para disminuir la posibilidad de lesión.

Sería interesante también convertir parte de la aplicación en un servicio de búsqueda para conectar entrenadores con nuevos clientes; en estos momentos, la aplicación depende de que el entrenador y el cliente se hayan conocido previamente para así intercambiar el número de entrenador. Con esta nueva funcionalidad, el círculo de posibilidades para encontrar nuevos clientes aumentaría para el entrenador, y al estar basados en entrenamientos online, la distancia no sería un impedimento.

En este caso, el cliente mandaría una petición al entrenador, que aceptaría o denegaría esta petición de nuevo cliente. Además, esta funcionalidad sería optativa para quienes prefieren buscar a los clientes por su cuenta.

8. Referencias

[1] Thompson WR. "Worldwide Survey of Fitness Trends for 2021",

https://journals.lww.com/acsm-healthfitness/Fulltext/2021/01000/Worldwide_Survey_of_Fitness_Trends_for_2021.6.aspx?context=FeaturedArticles&collectionId=1&__cf_chl_jschl_tk__=3451d69c6930b29ec85835ac20a1f034e1715e00-1612176595-0-ATSG7fV11v-kORunXhwHY56dNjkG3WlQsxGQ1uTGwmSu4-t7b67SNRrpOobTUpivSRsfYdRgpquyHRIliaQZnm0aWOw-pASnNKmjCaUW4ScGbncbhRy8HGVyQKmZ6EPl7bK7XMy5cxpsTG7EQVINiWbvH3rbE0my-pmrjDlb54IFLPntS7c2dBbah8yspOaHWby-hebHUWxBeCBl4C4R29pu6PPZUHgSp39eX6C8xRb0sDaiK7EsYldbcoK66NCt6LZfiKRbT0gexQPU6avLJJZ7uGtWIJ9PL4_Ec_cYjSP7dzU8rQxChmYWBFyd-bIG3T0ZFNbTeH-0inKVaJvYPOYkW-aZzjtMp-CsVCu9fnTxFACQydiE9XWm4mu7_xDTab0oMZ6pZhmvR7yBnX04rRe7meFzz0V-DDpW_pgiq9I-sL_fpiKmhpUCyZVBvq_CNP06QZFINfmdxeGj8tYyxS61g-pENg45i1Kgx2w9HO6NEo-GPHynC9xxfIOSfvvJy8OhKvhX3wpU5by_qfL42Lgo03cRiZs-dlcNgUGyC2RVjzUo

[2] Irene Sierra. "La demanda de entrenadores personales crece desde el estallido de la pandemia",

https://www.womenshealthmag.com/es/salud-bienestar/a36752840/demanda-entrenadores-personales-pandemia/

[3] OMS. "Actividad física",

https://www.who.int/es/news-room/fact-sheets/detail/physical-activity

[4] Cuidate Plus. "Entrenador personal",

https://cuidateplus.marca.com/ejercicio-fisico/diccionario/entrenador-personal.html

[5] NextU. "Material Design: ¿por qué todos hablan de eso?,

https://www.nextu.com/blog/material-design-que-es/

[6] Enrique Pérez. "Qué es Material Design?",

https://www.elespanol.com/elandroidelibre/20141109/material-design/2000127_0.html

[7] Flutter. "Introducción a los Widgets",

https://esflutter.dev/docs/development/ui/widgets-intro

[8] Raúl Ferrer. "Conoce cómo se construye en Flutter: los Widgets",

https://www.raulferrergarcia.com/conoce-como-se-construye-en-flutter-los-widgets/

[9] A.U.S. Gustavo Torossi. "El Proceso Unificado de Desarrollo de Software",

http://dsc.itmorelia.edu.mx/~jcolivares/courses/pm10a/rup.pdf

[10] Joan Teran. "Breve introducción a Tom's Planner",

https://masterenux.com/2020/07/31/breve-introduccion-a-toms-planner/

[11] Junta de Andalucía. "Herramientas para la Gestión y Documentación de Requisitos",

http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/420

[12] Amador Durán Toro. "Herramienta REM",

http://www.lsi.us.es/descargas/descarga_programas.php?id=3

[13] Wikipedia. "Visual Paradigm",

https://en.wikipedia.org/wiki/Visual_Paradigm

[14] Wikipedia. "Lenguaje unificado de modelado",

https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[15] Desarrollo web. "Flutter: introducción al framework multiplataforma",

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-flutter/

[16] Appandweb. "¿Qué es Flutter?",

https://www.appandweb.es/blog/ventajas-flutter/

[17] Victor Divi. "¿Qué es el lenguaje de programación Dart?",

https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart

[18] Desarrollo web. "Darat de Google: Una introducción al lenguaje Dart",

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/lenguaje-de-programacion-dart-de-google/

[19] Google. "Firebase",

https://firebase.google.com/products-build

[20] Sara López. "Firebase: qué es, para qué sirve, funcionalidades y ventajas",

https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

[21] Moises Belchin. "Generar documentación en dart con dartdoc",

https://dartgoogle.wordpress.com/2014/02/25/generar-documentacion-en-dart-con-dartdoc/

[22] Unknown. "Proceso Unificado de Desarrollo de Software",

http://pnfiingenieriadesoftwaregrupocuatro.blogspot.com/2012/07/proceso-unificado-proceso-unificado-de.html

[23] José María Aguilar. "¿Qué es el patrón MVC en programación y por qué es útil?,

https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx