UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación móvil para la gestión de un gimnasio

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Mora Buitrago, Kevin Danilo

Tutor/a: Esparza Peidro, Javier

CURSO ACADÉMICO: 2021/2022

Título

Aplicación móvil para la gestión de un gimnasio.

Resumen

La finalidad en este proyecto es crear una aplicación móvil que sirva de apoyo para gestionar las reservas de un gimnasio. Entre las herramientas con las que contará la app serán: permitir el registro de nuevos usuarios, reservar las diferentes actividades/horarios disponibles, consultar información sobre el gimnasio, seguir las dietas, entrenamientos personales y un apartado para ponerse en contacto con los entrenadores.

Abstract

The purpose of this project is to create a mobile application that serves as support to manage the reservations of a gym. Among the tools with which they will have the app will be: allowing the registration of new users, reserving the different activities/schedules available, consulting information about the gym, following diets, personal training and a section to contact the trainers.

Palabras Clave

App; Gestión; Gimnasio; Móvil; iOS; Swift; Xcode

Key Words

App; Management; Gym; Mobile; iOS; Swift; Xcode

Agradecimientos

Me gustaría agradecer primero a mi familia, que siempre me ha apoyado durante esta etapa en la universidad. No han importado las horas, meses o años, siempre han confiado en mí y en que con trabajo y esfuerzo podría obtener el título.

A mis profesores ya que sin ellos ninguno de nosotros a día de hoy sabríamos lo que sabemos. La dedicación que ponen día a día para que podamos aprender y sentar la bases para poder desarrollar esta profesión una vez salgamos de la universidad.

Por último, pero no por ello menos importante, a mis compañeros. La verdadera clave de toda la etapa universitaria. Mis compañeros me han dado la motivación directa o indirectamente para esforzarme durante estos años y al final conseguir el objetivo que es obtener el título. Sin su ayuda la gran mayoría de veces no habría podido sobrepasar los obstáculos que han ido surgiendo a lo largo del tiempo.

Índices

Contenido

Índice de tablas, imágenes, esquemas

1. Introducción

El objetivo de este proyecto es la realización de una app que, de manera sencilla, sirva de apoyo a un negocio, en este caso en concreto un gimnasio. Con esta app se busca gestionar diferentes funciones que se llevan a cabo en dentro de un gimnasio diariamente.

Cualquier negocio cuenta con un gran numero clientes por lo que en nuestro caso esta aplicación estará orientada a la parte del cliente, donde este podrá encontrar información sobre el gimnasio, sus novedades o promociones, seguir las dietas o rutinas, apuntarse a actividades, etc.

Todo aquel proceso de negocio que tenga que ver con los clientes y su gestión, se tendrá en cuenta para incluirlo en la aplicación. El objetivo final es darle al cliente una única herramienta con la que poder realizar cualquier acción dentro del gimnasio.

1.1. Motivación

La verdadera razón por la que elegí empezar este proyecto es porque tenía un gran interés en investigar y aprender sobre el desarrollo de aplicaciones para dispositivos móviles. Creo que actualmente este sector de la informática es uno de los más importantes y atractivos por lo que desde que empecé el grado siempre me ha llamado la atención el desarrollo de software y en especial el mundo de las apps. Durante el grado realmente no vemos nada de contendido a cerca de este sector. Tenemos asignaturas como ingeniería del software o alguna optativa, pero de resto no se profundiza tanto sobre este tema. Bajo mi punto de vista, es uno de los sectores más atractivos y entretenidos que hay en la informática.

Por otra parte, el tipo de negocio lo elegí ya que yo también formo parte de un gimnasio y entreno activamente en él. En este gimnasio no contamos con una herramienta especializada que nos ayude a poder ver las rutinas de manera sencilla o a apuntarnos a una actividad u hora concreta. La forma que tenemos de ver las rutinas o actividades es a través de google drive, aplicación la cual muchas veces no es muy cómoda de manejar en dispositivos móviles. Para apuntarnos a una hora concreta, se lo tenemos que comunicar al entrenador y allí en el gimnasio se nos reserva la plaza de forma escrita. Cualquier comunicado o noticia que se nos quiera dar como vacaciones ausencias, cuotas, etc. se hace o bien presencialmente o por aplicaciones de mensajería como WhatsApp. Por lo tanto, creo que una app para realizar todas estas acciones sería de gran ayuda.

Juntando mi interés sobre el desarrollo de apps para móviles y la idea de crear una herramienta para facilitar la gestión de un negocio, surgió esta idea. He tenido que aprender muchas desde la base, pero creo que al final ha merecido la pena ya que ha despertado aún más interés mi por seguir trabajando en este campo de la informática y el desarrollo de software.

1.2. Objetivos del proyecto

Para este TFG los objetivos que me marque al principio, los cuales yo consideraba más importantes o primarios fueron:

- Crear una UI intuitiva, sencilla y atractiva para el usuario.
- Hacer uso de una parte servidor donde poder gestionar todos los datos de los clientes, así como sus dietas rutinas o actividades.
- Juntar en una sola herramienta, todas las funcionalidades que lleva a cabo un cliente habitual de un gimnasio.
- Crear una aplicación donde cada cliente tuviera únicamente contenido, información y datos de su interés personal.

Como objetivos secundarios tenia:

- Poder darse de alta como usuario y visualizar el contenido personal de cada uno (dietas, rutinas, actividades).
- Incluir un apartado de información sobre el gimnasio donde se encontrarán los diferentes entrenadores, redes sociales, localización imágenes y demás.
- Añadir una ventana de novedades donde se pueda comunicar a todos los clientes promociones, noticias o cualquier evento en especial.
- Otra venta o ventanas donde se puedan visualizar las dietas y rutinas exclusivas de cada cliente.
- Una ventana donde se encentren todas las actividades que se llevan a cabo en el gimnasio y donde poder apuntarse a cada una de ellas.
- Finalmente, una última ventana donde de manera representativa se puedan ver las actividades que tiene cada cliente.

1.3. Impacto esperado

Una vez finalizado el proyecto se espera que el funcionamiento de un gimnasio mejore considerablemente ya que se agrupan varios de los procesos cotidianos con una sola herramienta. Se puede obtener un mejor control de los usuarios inscritos en el gimnasio, de las dietas/rutinas de cada uno. Qué actividades se van a realizar y cuando, además de las personas inscritas en ellas. En general dar un salto de calidad de cara al cliente ya que es más atractivo el negocio y aparte mejorar la forma en la que se trabaja día a día en el gimnasio.

Por otra parte, para el cliente es más sencillo y rápido el gestionar sus actividades, estar al tanto de las noticias acerca del gimnasio, desarrollar sus ejercicios y ponerse en contacto con los entrenadores.

1.4. Estructura de la memoria

A continuación, se definirá cuál será la estructura de la memoria del proyecto y cómo se han dividido cada una de las partes a partir de este punto.

- Punto 2. En este punto se repasará el mercado y el estado actual de las aplicaciones de este estilo con sus puntos fuertes y débiles de cada una. Se desarrollará la propuesta detallada del proyecto.

- Punto 3. Se hará un repaso por todas las tecnologías usadas para la realización del TFG, los lenguajes de programación y la explicación de la parte servidor o base de datos.

- Punto 4. En este punto se entrará a explicar el diseño de la solución de forma conceptual y a grandes rasgos. Arquitectura de la aplicación y cada una de las capas. Además, se abordará cada capa por separado y el diseño llevado a cabo.

- Punto 5. Continuando con el punto anterior se explica cómo se ha desarrollado la propuesta siguiendo el diseño comentado anteriormente, dividiendo por cada una de las capas de la aplicación.

- Punto 6. Resultado de la aplicación. Se muestra finalmente cual es el aspecto de la aplicación a modo de tour por ella.

- Punto 7. Se finalizará la parte de diseño y desarrollo del software realizando una valoración final y unas conclusiones. Problemas durante el desarrollo y relación con los estudios cursados.

- Punto 8. Trabajos futuros. Brevemente se explica que aspectos podrían ser mejorados si hiciera falta y que otros desarrollos se podrían incluir en la aplicación.

- Punto 9. Referencias bibliográficas y glosario de términos.

2. Estado del arte

Comenzaremos este apartado realizando un análisis y estudio del mercado de los posibles competidores de la aplicación.

El mercado de las aplicaciones para móviles es muy amplio por lo que podremos encontrar gran cantidad de aplicaciones que sirven de soporte para la gestión de un negocio. En mi caso concreto he acortado mucho más la búsqueda y me he centrado en este tipo de aplicaciones pero que estuvieran enfocadas a los gimnasios o el ejercicio físico.

- Ufit365

Ufit365 es un proveedor de este servicio, es más que una única aplicación independiente. Esta empresa ofrece su aplicación a diferentes clientes para que cada uno de ellos la puedan customizar en base a sus necesidades y así poder llevar a cabo la gestión de su negocio desde esta aplicación.

IMAGEN 1: UFIT365 LOGO

La empresa cuenta con una app base en la que cada negocio podrá añadir o quitar módulos con forme deseen de tal forma que será personalizable para cada gimnasio, pero al final todas tendrán la misma estructura y diseño.

Los módulos con los que cuenta esta aplicación son: Entrenamiento, Plan Nutricional, Análisis Corporal, Eventos, Reserva de Clases, Cursos, Sistema de Reservas de Servicios Profesionales...

A parte de esto, desde la web oficial de la empresa [1] comentan también que se pueden diseñar las rutinas de cada cliente y montar sus entrenamientos, aparte de contar con un sistema de reserva de actividades personalizable.

Al no ser cliente de ningún gimnasio no he logrado tener acceso a muchos de estos módulos, pero si he podido visualizar varios de ellos y analizar su funcionamiento.

IMAGEN 2: UFIT365 APP

Como se observa en las diferentes capturas de pantalla (Imagen 2), el diseño parece bastante sencillo y se ve bastante bien, sin embargo, la forma de pasar de una pantalla

a otra es un tanto confusa. Cuenta con tres menús diferentes con una disposición de vistas distinta por lo que bajo mi punto de vista la navegación se hace lenta en algunos momentos. A parte de esto, como ya he comentado antes, la aplicación tiene el mismo diseño y estructura para todas las empresas por lo que ninguno de los módulos y ni siquiera el aspecto es modificable. Simplemente te proveen un servicio al que los negocios se pueden amoldar o no.

De todas formas, el producto me parece muy bueno ya que para la mayoría de los casos funciona correctamente y cumple el objetivo de permitir a los gimnasios gestionarse a través de una única herramienta.

- TCP-MATCHPOINT

Un caso muy parecido al anterior. Empresa proveedora de un software de gestión enfocado a clubes deportivos, en este caso no solo gimnasios. [2]

IMAGEN 3: TCP-MATCHPOINT

Uno de los aspectos que diferencian las dos opciones ya comentadas es que, en este caso, las posibilidades de flexibilidad y adaptación a los requerimientos de cada negocio son mayores.

Como ya he dicho nos centraremos solo en perfiles como gimnasios ya que gestionan más clubes deportivos. A parte de esto la empresa ofrece gran cantidad de servicios, pero solo he analizado la aplicación.

Desgraciadamente no he podido comprobar el funcionamiento de la app debido a la misma razón de antes, el acceso. Sin embargo, si se pueden visualizar capturas de pantalla y por otra parte he podido comprobar mínimamente el servicio en una app de un gimnasio en concreto que hace uso de esta empresa.

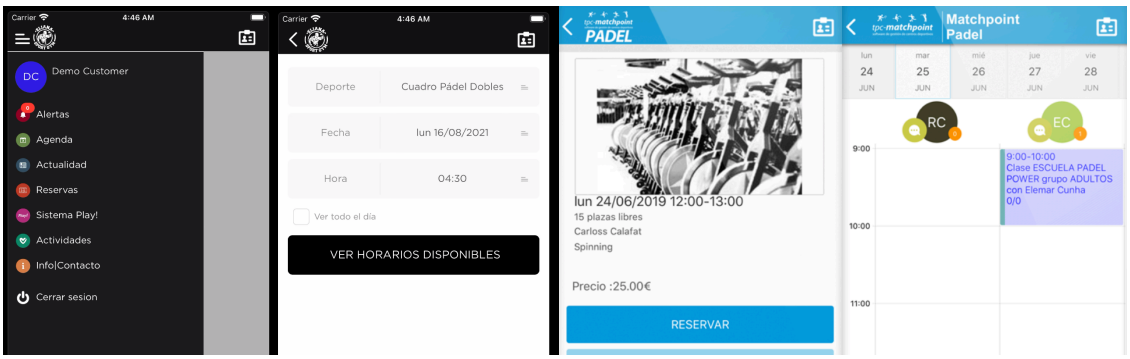

IMAGEN 4: TPC-MATCHPOINT APP

Esta opción me parece relativamente mejor ya que uno de los puntos débiles de la anterior opción se ven mejorados aquí y ese es el de la personalización. Creo que esta opción es más potente y ofrece a cada negocio un poco más de flexibilidad.

Realmente no he podido observar los módulos y funcionalidades por lo que me baso en lo que se ofrece y el aspecto que debería de tener una app finalizada. En este caso posiblemente la navegación sea mejor y el aspecto perece sencillo y fácil de usar.

- Apps Independientes

Otras opciones del mercado son las apps creadas para un negocio en concreto. En este caso he encontrado dos que voy a analizar brevemente ya que se asemejan bastante a las anteriores opciones.

La primera de ella es para un gimnasio bastante conocido, ubicado en la ciudad de valencia (Titan Fitness) y la segunda es para un gimnasio de mi localidad (Agón Gym). Las dos apps son muy similares, pero se diferencian en que a priori gestionan un número de distinto de clientes.

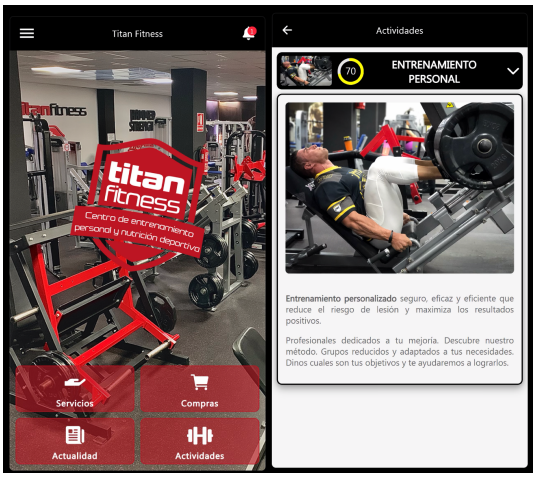

IMAGEN 5: TITAN FITNESS APP

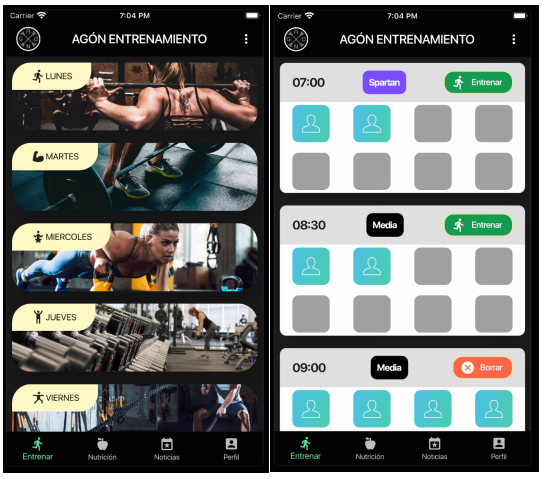

IMAGEN 6: AGON GYM APP

La primera app cuenta con un estilo sencillo fácil de usar. Incluye módulos muy básicos pero importantes como es el de actividades, actualidad o servicios. No se ha podido testear cada uno de estos módulos por no estar dado de alta, pero a simple vista, el módulo de actividades incluye información sobre cada una de ellas y los usuarios pueden apuntarse a través de la vista de servicios en las horas que se ofrecen diariamente.

La segunda app tiene un estilo aún más sencillo y fácil de usar, donde en la primera vista se encuentran los entrenamientos separados por días de la semana. En la segunda imagen podemos ver como el entrenamiento se separa por horas y en este caso está pensado para poder realizarlo de forma remota e individual por cada cliente. Del resto de módulos no se da mucha más información, pero sirven para seguir las dietas y estar al tanto de las noticias.

Como punto débil puedo decir que estas dos apps son más básicas ya que cuentan con funcionalidades menos complejas que las anteriores. Sin embargo, son totalmente personalizables ya que el diseño es único para cada gimnasio y se estructura en base a las necesidades que tenga cada uno de ellos.

3. Análisis de requisitos

Una vez realizado todo este análisis de mercado toca elegir como se empezará a confeccionar la propuesta de trabajo.

La idea principal era crear una aplicación basándome en todo lo visto anteriormente y de alguna forma incluir de cada uno de los competidores sus puntos fuertes, para así poder crear una app que reuniera todas estas ventajas.

Por lo tanto, la app deberá de incluir varias vistas y de una forma sencilla poder navegar entre ellas. Cada una de estas vistas ofrecerá al gimnasio la posibilidad de cubrir sus necesidades principales. Las vistas por lo tanto serán: una vista para visualizar la información acerca del gimnasio, otra para poder estar al día de las noticias y novedades, otra para que cada usuario pueda seguir las diferentes dietas y rutinas que tenga, y por último dos vistas más que podrían estar relacionadas como son las actividades y un horario donde representar de forma visual a las que están apuntado el usuario.

Para la vista de información simplemente se incluye una foto o fotos, la descripción clara y breve del gimnasio, información acerca del entrenador como sus redes sociales, correo, teléfono, etc. es decir, poder ponerse en contacto con el entrenador. Finalmente, la localización del gimnasio con un mapa interno en la app. La funcionalidad de esta vista no es más que poder tener un punto de referencia o inicial donde se encuentren los datos del gimnasio y entrenador/es.

La siguiente vista es la de las novedades. En esta vista es realmente sencilla, el usuario podrá visualizar una foto y breve descripción por cada novedad. Estas novedades son comunes para todos los usuarios por lo tanto se les debería mostrar a todos por igual.

La funcionalidad aquí es la de mantener al día a los usuarios del gimnasio. Si se quiere comunicar algo, desde esta vista podrá ser posible.

Sobre las dietas y rutinas hay diferentes opciones. Separar las vistas o juntarlas en una única vista. Al principio valoré juntarlas en una única vista ya que el funcionamiento es muy similar, pero finalmente decidí separarlas para que la visualización fuera más cómoda para el usuario. Dentro de esta vista pensé de qué manera se podría representar las dietas o rutinas y la mejor opción dentro de mi punto de vista era incluir un lector de documentos (imágenes, pdf). De esta forma, si la dieta o la rutina es demasiado compleja la visualización será mucho mejor y más fácil de incluir dentro de la app. Además de esto, cada usuario tiene su propio contendido por lo que para manejar los datos resulta más sencillo de esta forma.

En cuanto a su funcionalidad un usuario podrá acceder a esta vista y elegir en una lista que dieta o rutina desea ver. Una vez seleccione una de ellas se le abrirá una venta donde podrá ver el contenido de esa dieta/rutina específica.

En las actividades y horarios fue donde más cambie de idea y busque la mejor forma para conseguir que funcionara correctamente. Lo principal y creo que bastante bueno para el usuario es que se puedan representar las actividades en un calendario. De esta manera se puede gestionar mejor el tiempo y a simple vista ver mejor la rutina semanal o diaria. Para ello se puede utilizar o replicar un calendario donde al seleccionar una actividad automáticamente se represente ahí. En cuanto a las actividades como tal finalmente las decidí separar entre los diferentes días de la semana. Cada día tendrá sus actividades concretas en el horario concreto y al pulsar sobre una de ellas se abrirá una ventana donde elegir el día deseado. Las actividades contarán con un número máximo de asistentes por lo que si se alcanza ese número ya no se podrán apuntar más clientes.

La funcionalidad por tanto es, primero acceder a las actividades y seleccionar un día de la semana. Dependiendo del día que seleccione se mostraran una lita de actividades u otra con sus horarios y título. Pulsando sobre una de estas actividades se abre otra vista, la cual da la posibilidad al usuario de elegir el día concreto en el que se quiere apuntar. Mediante un selector de fechas o algo parecido el usuario elige su día y selecciona apuntarse. Las actividades contaran con un máximo de asistentes por lo que, si ese máximo se cumple, se le avisa al usuario indicándole que no es posible al completarse el aforo.

Si se permite apuntarse a una actividad, esa actividad se verá representada en el horario. Accede a la vista de horarios y en el calendario, pulsando sobre el día en el que se había apuntado aparecerá la actividad a la hora que toca. En este calendario se puede la rutina diaria dividida entre horas y donde aparecen todas las actividades que tenga.

A continuación, se mostrará un diagrama de casos de uso [3] que refleja al primer actor: usuario que utiliza la aplicación y desea realizar cualquiera de las actividades que se ofrecen dentro de ella (Imagen 7).

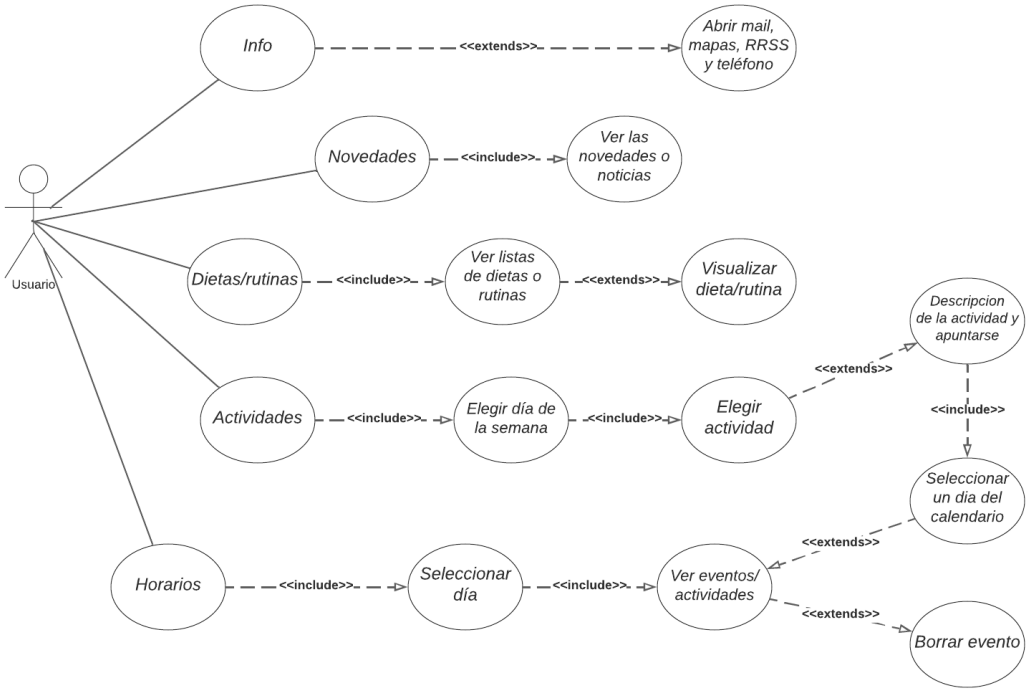

IMAGEN 7: DIAGRAMA DE CASOS DE USO

Una vez ideada la aplicación ahora se decide para que plataforma. Las opciones son sencillas y conocidos por todo el mundo, Android o iOS.

En mi caso me decante por iOS ya que mi ordenador personal es de la marca Apple por lo que podía elegir entre las dos variantes y la de Apple me pareció más interesante.

Es difícil decir cuál es mejor plataforma para desarrollar aplicaciones, pero lo que está claro es que Apple ofrece un producto más cerrado, en una cantidad concreta de dispositivos. Debido a esto, el desarrollo se hace relativamente más sencillo y con una documentación más específica.

En el siguiente punto entrare más en detalle a cerca del entorno de trabajo, lenguajes herramientas, etc.

4. Entorno de desarrollo

En este punto indagaré a cerca de las tecnologías que he usado para este TFG.

4.1. Entorno de desarrollo integrado

- Xcode:

Xcode es la herramienta que se usa en la mayoría de casos para desarrollar aplicaciones para iOS, MacOS, watchOS o tvOS. Distribuida por Apple cuanta con todas las herramientas necesarias para poder desarrollar aplicaciones de la mejor manera. Existen más herramientas de desarrollo multiplataforma como pueden ser Flutter [4] o KotlinMM [5].

IMAGEN 8: XCODE LOGO

Para la finalidad de este proyecto no es necesario desarrollar la aplicación para distintos dispositivos ya que no es lo que ese busca. Es por ello que la elección fue sencilla y me decante directamente por el framework nativo de iOS en su versión 13.2.1.

Esta herramienta es bastante sencilla de usar y provee a los desarrolladores de todas las herramientas necesarias para poder crear las apps.

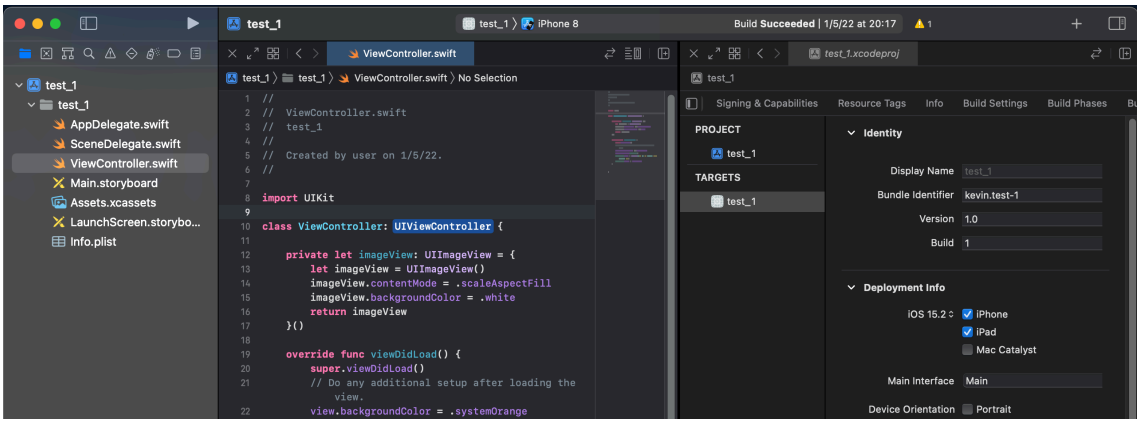

IMAGEN 9: XCODE

Xcode permite elegir entre diferentes formas de desarrollar una app como pueden ser mediante un Storyboard donde se podrá dar el diseño a la interfaz de usuario mediante diferentes componentes o crear las vistas de forma programática. En mi caso elegí la segunda ya que ofrece más posibilidades de personalización y control sobre las vistas.

Además de esto, tuve que hacer uso de diferentes librerías externas para poder incluir varias herramientas que me eran necesarias.

Hay varias formas de incluir estas librerías. Mediante Podfiles o añadiendo los paquetes directamente en Xcode. Para la primera opción se tiene que hacer uso del intérprete de comandos (terminal) y mediante comandos se importa y se instalan las librerías en nuestro espacio de trabajo, es decir nuestro proyecto de Xcode. En el punto 5 indagaré más acerca del uso de los Podfiles.

4.2. Dispositivo móvil

Para realizar las diferentes pruebas y funcionamiento de la app se necesita un dispositivo móvil de la marca Apple ya que no es posible desplegarlas si no se tiene el mismo sistema operativo.

Ya se ha visto la primera herramienta Xcode y como he dicho este entorno ofrece todo lo es necesario para el correcto desarrollo de las apps. Xcode cuenta con sus propios simuladores de diferentes dispositivos. Según la versión incluirá unos u otros y el sistema operativo también variará. En mi caso al estar utilizando una de las últimas versiones tenia disponible todos los modelos actuales del mercado con la última versión de iOS integrada. No hay que hacer ninguna configuración adicional porque las apps se despliegan por defecto en el simulador que nosotros escojamos.

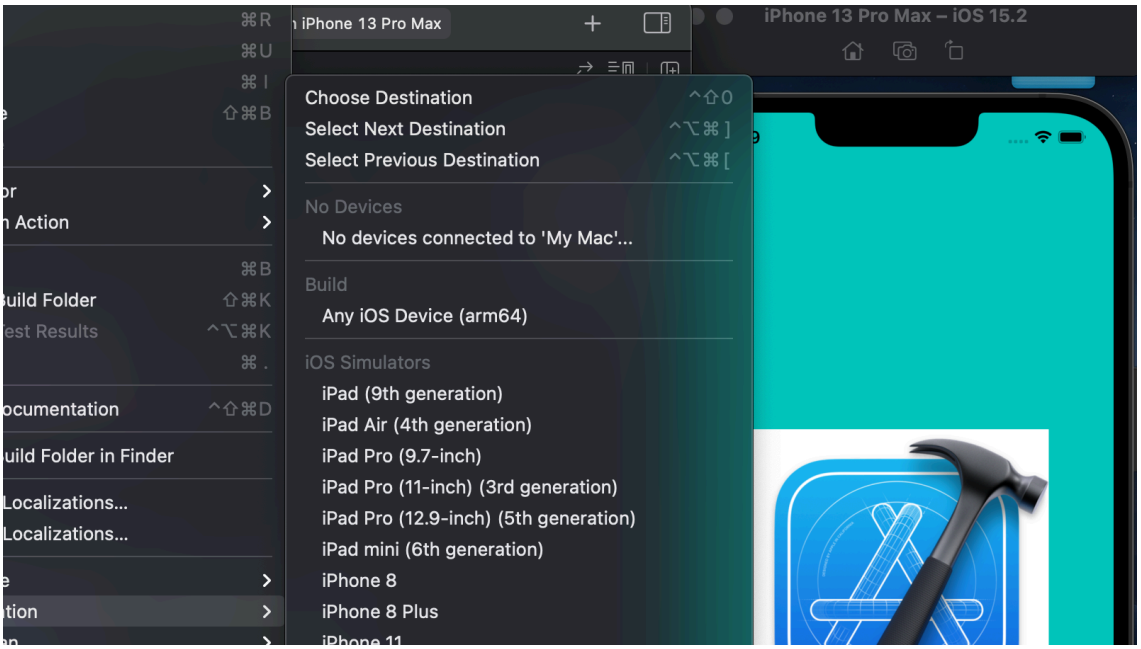

IMAGEN 10: SIMULADOR XCODE

Sin embargo, y como se puede ver en la última imagen se puede utilizar un dispositivo personal conectándolo al ordenador y desplegando la app en ese dispositivo. Incluso se permite desplegar las apps a través de wifi en nuestro móvil.

4.3. Lenguajes de programación

- Swift

Si se elige desarrollar la app con el entorno nativo de Apple se debe usar por tanto el lenguaje de programación nativo de la compañía.

Originalmente las aplicaciones de iOS y en general de Apple se programaban en Objective-C. Actualmente el lenguaje estándar de desarrollo ha pasado a ser Swift. Swift es un lenguaje multiparadigma creado por la propia Apple y que hereda directamente bibliotecas programadas en Objective-C y puede llamar a funciones de C. La compatibilidad con Objective-C por tano es completa, pero bajo ciertas condiciones.

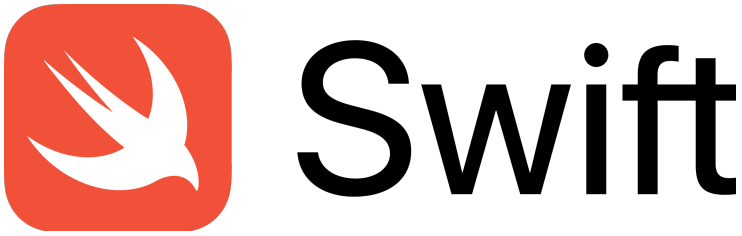

IMAGEN 11: SWIFT LOGO

En mi caso he usado la última versión que es Swift 5. He de decir que hay una gran cantidad de contenido e información acerca de este lenguaje y esa es una gran ventaja a la hora de buscar información por internet.

Por último, añadir que a partir del año 2019 se lanzó SwiftUI, una framework de este mismo lenguaje que se utiliza de forma distinta. Este framework es totalmente exclusivo de Swift por lo que la compatibilidad con los otros lenguajes ya no es posible. SwiftUI intenta mejorar muchos de los conceptos de Swift 5 y además de esto unifica mucho más la programación para entornos Apple.

Mi elección fue por tanto Swift 5 ya que, bajo mi punto de vista, al tener más tiempo desde su lanzamiento, la cantidad de documentación e información es mayor a la de SwiftUI.

4.4. Servidor de bases de datos

- Firebase

En cuanto a la persistencia de los datos existen varias opciones.

Entre todas estas opciones encontramos, por ejemplo, SQLite. Esta opción es una biblioteca que implementa una base de datos relacional. Es un sistema más rápido al estar integrado dentro de la misma aplicación y no necesitar de acceso a un servidor externo.

Otra opción es Core Data, que no es en sí una base de datos si no es una solución de mapeo relacional de objetos creada y mantenida por Apple. Dentro de Xcode hay un editor de modelos de Core Data. Este editor permite a los desarrolladores modelar el

modelo de datos de la aplicación a través de una interfaz gráfica. A pesar de esto, administrar una gran cantidad de datos puede tener un peor rendimiento que SQLite.

En mi caso me decante por Firebase. Firebase es una plataforma desarrollada por Google que sirve para desarrollar y facilitar la creación de aplicaciones para dispositivos móviles. La plataforma se encuentra alojada en la nube y está disponible para diferentes plataformas como Android, iOS, y web.

IMAGEN 12: FIREBASE LOGO

Firebase [6] es una variante que permite no tener que dedicarle tanto tiempo al backend y aun así seguir mantenido una calidad de desarrollo y mantenimiento muy buena. Algunos de los objetivos principales de la plataforma son:

- Análisis
- Desarrollo
- Crecimiento
- Monetización

La principal ventaja y razón de uso para este proyecto es su cloud storage el cual permite desarrollar sin recurrir a bases de datos propias.

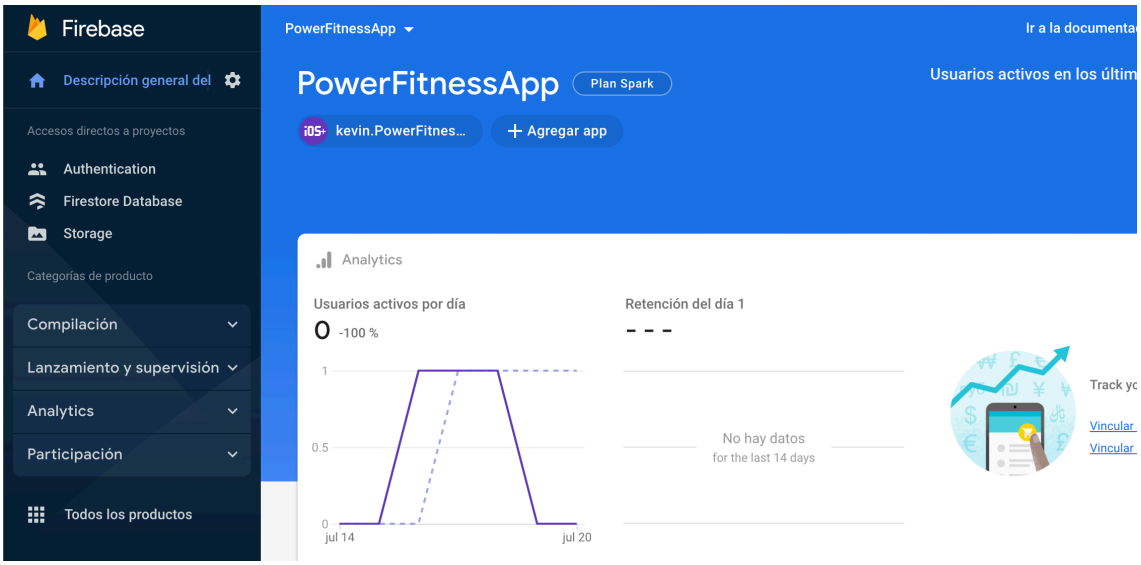

IMAGEN 13: FIREBASE

Simplemente hay que enlazar nuestra aplicación con el proyecto de firebase y como se observa en la imagen podemos hacer uso de una grandísima variedad de módulos que nos permitirán crear bases de datos, gestionar a los usuarios, guardar archivos; aparte de esto también se pueden obtener estadísticas como son análisis al desplegar la aplicación, usuarios activos, etc. Para este proyecto solo haremos uso de tres módulos.

Para acabar, simplemente destacar otra ventaja y es que la gran mayoría de herramientas que nos ofrece la aplicación son totalmente gratuitas.

5. Diseño de la solución

En este punto se entrará a explicar la arquitectura de la aplicación y cada una de las capas. Además, se abordará cada capa por separado y el diseño llevado a cabo.

5.1. Arquitectura de la aplicación

Para el desarrollo de esta aplicación he decidido seguir el modelo de tres capas (presentación, negocio y datos) visto durante el grado en la asignatura de ingeniería de software.

Este modelo cuenta con tres capas como son la de presentación, la de negocio y la de datos:

- Capa de presentación: Qué verá el usuario al utilizar la app. Incluye los controles y el código para la interacción de este usuario con el sistema. Se comunica con la capa de negocio.

- Capa de negocio: Conecta con la anterior y es donde se reciben las peticiones de los usuarios y se les devuelve las respuestas. Las peticiones pueden estar relacionadas con los datos o no, si las peticiones requieren de acceso a los datos entonces conectará con la capa de datos para recibir la información. Capa intermedia que hace de puente entre el resto de capas.

- Capa de datos: Mediante un gestor de bases de datos (o varios) aloja la información para posteriormente poder recuperarla y devolverla en la capa de negocio.

La arquitectura que he utilizado basándome en este modelo es cerrada, es decir, las capas solo podrán acceder a funcionalidades de otras capas siempre y cuando sean inmediatamente inferiores. Con una arquitectura cerrada tendremos mejor separación entre las capas y manejarlas de forma independiente es mucho mejor. Para que se entienda esto mejor he incluido el siguiente diagrama [7] (Imagen 14).

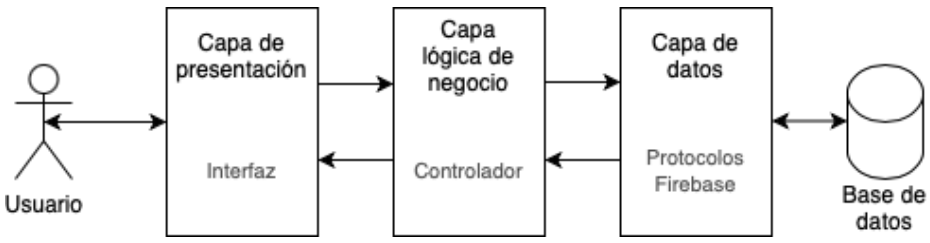

IMAGEN 14: MODELO DE TRES CAPAS

En un proceso habitual dentro de nuestra aplicación, el usuario interactuará con la interfaz gráfica (presentación) accionando los diferentes controles que tenga en la vista y visualizando la información que haya por defecto.

Suponiendo que quiera acceder al contendido a cerca de sus dietas, entonces será la capa intermedia (lógica de negocio) la que restrinja esta información dependiendo del usuario activo.

Finalmente, para mostrar esta información, la capa intermedia conecta con la de datos para de esa manera poder solicitar los documentos asociados al usuario y una vez obtenidos los resultados poder presentarlos en la vista o interfaz gráfica.

Una vez vista la arquitectura de la aplicación que vamos a seguir pasare a explicar el diseño de cada una de las capas individualmente.

5.2. Capa 1 Presentación

En la capa de presentación es necesario dividir la interfaz en distintas vistas, así como la navegación entre ellas. Para el diseño de esta capa directamente hay que basase en los objetivos que se buscan y en la propuesta inicial comentada anteriormente en esta memoria.

La parte más importante a ceca de este punto es el aspecto visual y la disposición de los controles. Las vitas deben ser intuitivas, claras y fáciles de usar. En mi caso la aplicación no requería de una alta complejidad por lo que los diseños eran más sencillos de idear. Sin embargo, hay que tener muy en cuenta que el usuario debe poder hacer uso de todas las herramientas que se le ofrecen sin ningún tipo de adaptación previa. La primera vez que se accede se entiende el funcionamiento de la aplicación de forma rápida y sencilla.

Por lo tanto, la aplicación debe tener un total de ocho vistas, sin contar con la navegación entre ellas:

- Inicio de sesión.
- Creación de usuarios.
- Sobre nosotros (también llamada *Info* a lo largo de la memoria).
- Novedades.
- Dietas.
- Rutinas.
- Actividades.
- Horarios.

Para una mejor explicación en este punto he hecho uso de una herramienta para la creación de Mockups [8]. Además, de que visualmente se entiende mejor, también es necesario ya que de esta forma se pueden ver los cambios que han ido surgiendo en la aplicación en cuanto a la interfaz de usuario, mayormente en la navegación.

Pasamos por lo tanto a explicar la navegación y cada vista por separado.

- Inicio de sesión

Para el inicio de sesión el diseño era muy sencillo y simplemente me base en los inicios de sesión típicos de una aplicación para móviles. Introducir el usuario, su contraseña, un botón para acceder a la aplicación y otro para crear un nuevo usuario si se desea.

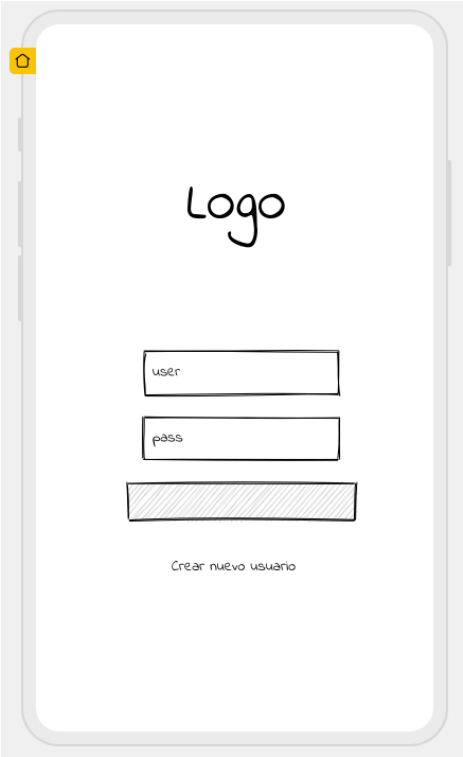

IMAGEN 15: INICIO DE SESIÓN

- Creación de usuarios

Como se ha podido observar en la vista anterior, en la parte inferior debería de incluirse un pequeño botón por si el usuario que accede a la aplicación no tiene cuenta o no esta dado de alta.

Esta vista no tiene su propio Mockup ya que el diseño es muy parecido al de inicio de sesión. Lo único que incluye es un campo extra para añadir el nombre del usuario.

- Navegación

He comentado que la aplicación tenía un total de ocho vitas. Sin embargo, la navegación, aunque no forma parte directamente del grupo de vitas de la interfaz, también se debe de diseñar con cuidado ya que es la encargada del paso entre una vista y otra.

En este punto es donde más dudas tuve a la hora de implantar el diseño final. Primero mostrare la primera idea que tuve y después finalmente como decidí que sería el diseño.

Inicialmente pensé en crear una vista de inicio que sirviera para la navegación. Con esto me refiero a que un usuario que inicie sesión, lo primero que vería es una serie de iconos donde pulsando en cada uno de ellos pudiera acceder a las distintas vistas. Además, si quisiera cerrar la sesión tendría un icono en la parte superior derecha que lo sacaría de la app.

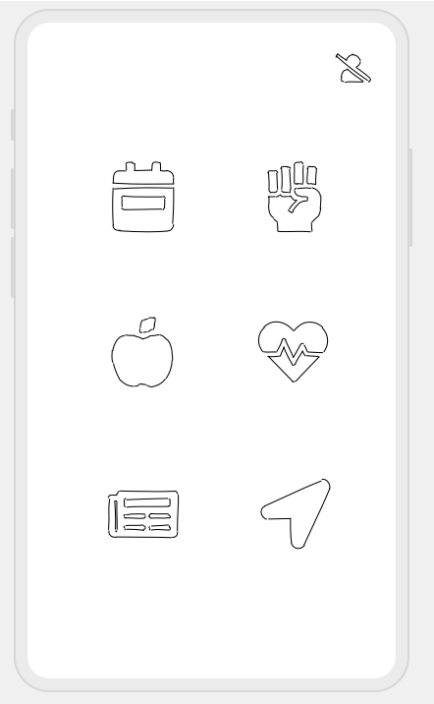

IMAGEN 16: MODELO DE NAVEGACIÓN INICIAL

A partir de aquí, si se selecciona un icono se pasa a la vista en concreto y con otro botón en la parte superior izquierda se vuelve al inicio.

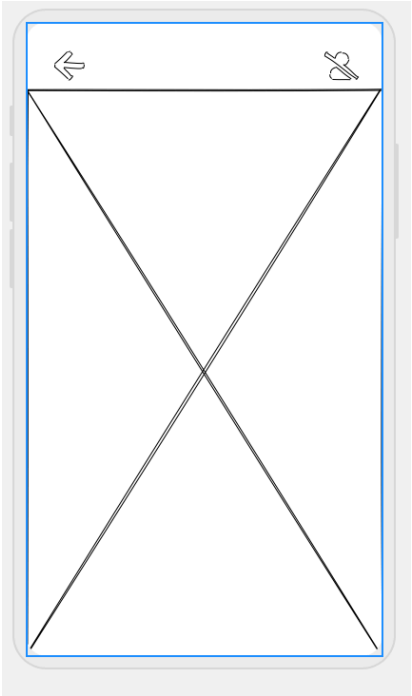

IMAGEN 17: MODELO DE VISTA INICIAL

La razón por la que al final deseche esta opción fue que bajo mi punto de vista consideraba que la navegación se hacía muy lenta para el usuario. Primero se accedía al inicio, de ahí a una vista concreta y para volver se tenía que pasar de nuevo por el inicio. Esto era uno de los puntos negativos que le vi a la app de Ufit365 por lo que decidí que necesitaba cambiar de idea.

La segunda opción que idee fue la de crear un menú lateral que se desplegara en cualquier vista. Ahí sí sería posible cambiar más cómodamente entre vistas y me ahorraría una de inicio que no tenía ningún funcionamiento en especial.

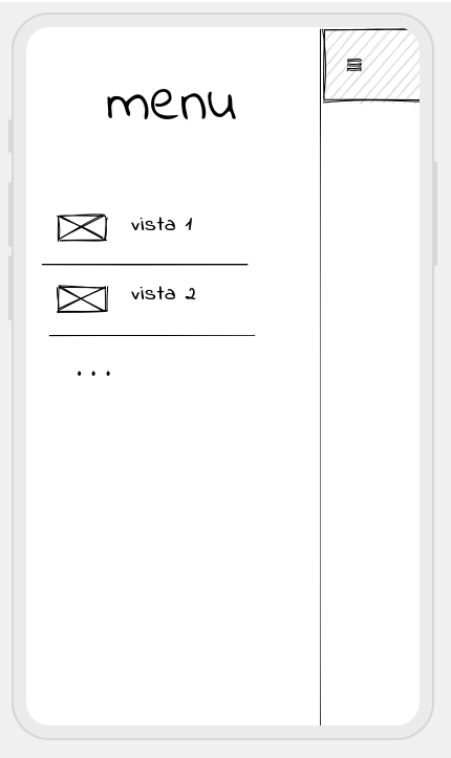

IMAGEN 18 NAVEGACIÓN FINAL

Ahora si se accede de forma más rápida y al iniciar sesión lo que se muestra es una vista más funcional como es la de 'Sobre nosotros' por ejemplo.

- Sobre nosotros

En esta vista se incluye la información básica del gimnasio como puede ser una foto, la localización, descripción corta, horarios, etc.

Para la vista pensé en primero incluir una imagen o varias imágenes del gimnasio, con una breve descripción del mismo y los horarios.

Además de esto también información del entrenador. Su foto, redes sociales, teléfono o mail.

Al final de la vista lo único que quedaba era incluir la localización ya fuera con un link o directamente añadir un mapa en la vista si se pudiera.

IMAGEN 19: VISTA SOBRE NOSOTROS

Comentar también que en la parte superior se añade una barra donde de izquierda a derecha encontramos: un botón para abrir el menú, el título de la vista y un botón para cerrar sesión. Todas las vistas deberían de tener la misma barra.

- Novedades

En esta vista la información que se muestra es la de las novedades, anuncios o promociones que el gimnasio quiera comunicar a los usuarios.

La disposición de esta vista es por lo tanto muy sencilla ya que no hay interacción realmente con ningún controlador de la vista más allá de la barra superior (común para todas).

Para visualizar las noticias pensé en que la mejor manera de hacerlo era incluyendo un título, una imagen (si se desea) y una descripción de cada una de ellas.

Por lo tanto, la vista finalmente tendría este diseño:

IMAGEN 20: VISTA NOVEDADES

- Dietas

Para las dietas como he comentado anteriormente, debía de buscar una forma sencilla de poder presentarle al usuario toda la información. Ya de por si montar una dieta requiere de mucha información y datos, aparte de esto la visualización en la app puede llegar a ser confusa y difícil de seguir. Finalmente decidí que la forma de hacerlo podía ser mediante un lector de documentos.

En la vista se listarían todas las dietas en orden y el usuario podría pulsar sobre cada una de ellas y mediante otra vista presentar el documento donde poder navegar de forma más fácil y rápida. Este documento puede ser un pdf que contenga la información necesaria para seguir una dieta concreta.

Otro aspecto que valore en este punto fue el de juntar o separar las diferentes dietas y rutinas en una o varias vistas. Juntas tendrían estas listas ocultas y al pulsar en una opción u otra se mostraría cada lista. Al final decidí separarlas en vistas independientes ya que visualmente es mejor porque se ve a simple vista la cantidad de dietas o rutinas que hay.

Primero mostraré la idea inicial que era juntar las vistas.

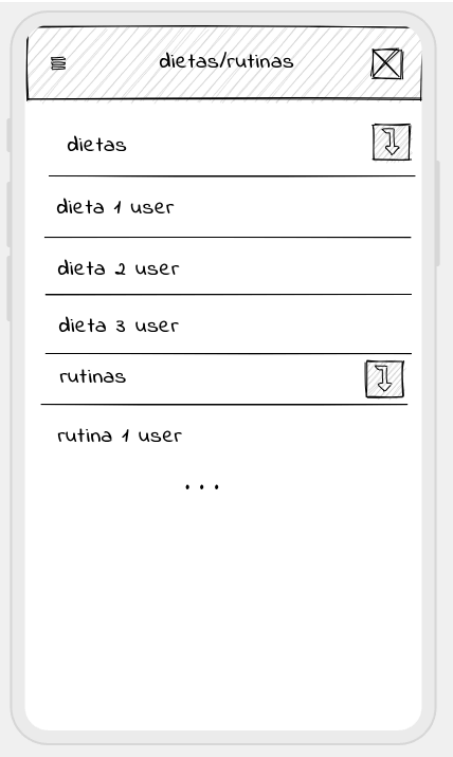

IMAGEN 21: MODELO INICIAL DIETAS/RUTINAS

Como se puede apreciar, pulsando en las flechas se expande y se muestra la lista, al pulsar de nuevo se ocultarían de nuevo.

Finalmente, al separarlas, las vistas se ven de la siguiente manera.

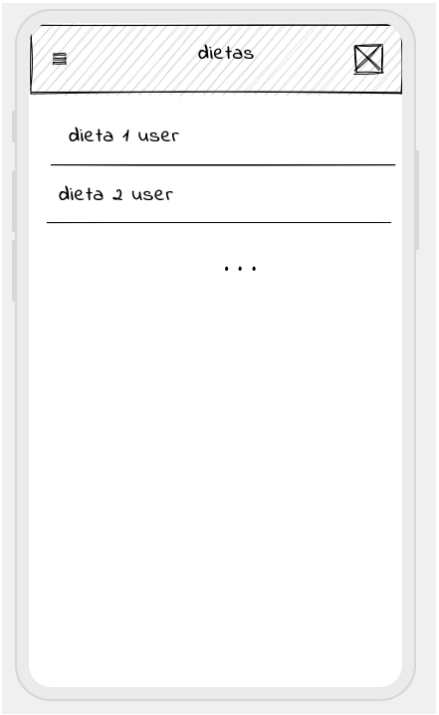

IMAGEN 22: VISTA DIETAS/RUTINAS

Y como he comentado, la forma de visualizar el contenido es pulsando sobre cada una de ellas. Para esto se utiliza una vista a parte.

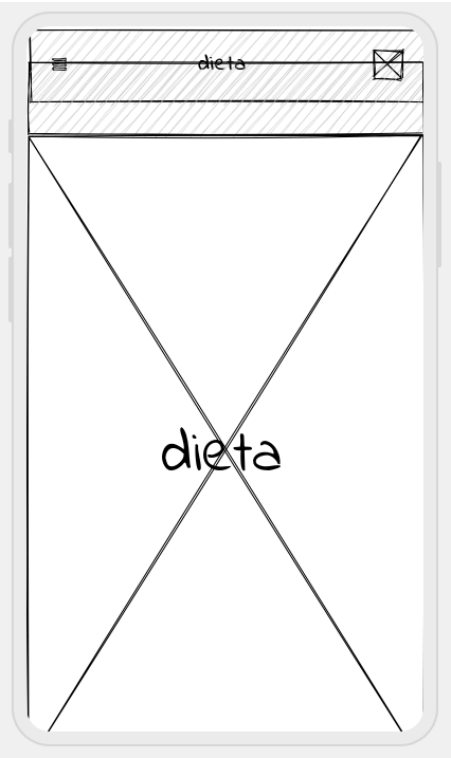

IMAGEN 23: LECTOR DIETAS/RUTINAS

- Rutinas

En cuanto a las rutinas, el diseño es idéntico al de las dietas. El aspecto es igual, listado de las diferentes rutinas y pulsando sobre cada una de ellas un lector que muestre el contenido.

Por esta razón no he completado el Mockup de las rutinas, es igual que el punto anterior, pero en la app el contenido debería ser diferente.

- Actividades

La vista de actividades también fue variando su diseño, aunque inicialmente ya tenía una idea bastante clara de cómo se podría diseñar.

Para las actividades habría justo debajo de la barra superior, una lista de días para poder elegir el que se desee y a partir de ahí mostrar las actividades de ese día de la semana.

En la lista de actividades tuve dos opciones, la primera era simplemente ordenarlas por hora, una debajo de otra y con un switch en la parte derecha poder apuntarse o desapuntarse.

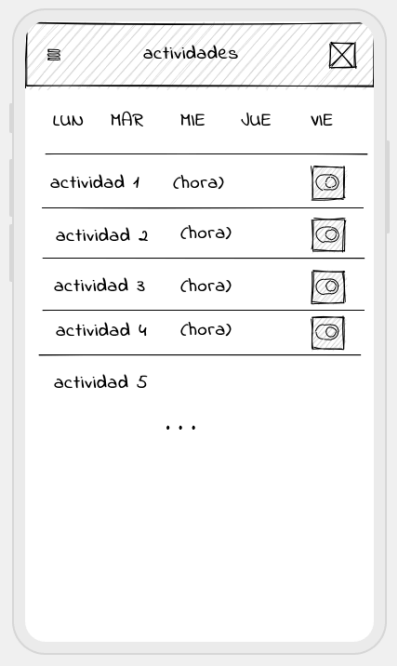

IMAGEN 24: MODELO INICIAL ACTIVIDADES

Encontré un problema a este diseño y es que la información era muy mínima. Por espacio en la vista no se puede visualizar ninguna descripción de la actividad ya que si no quedaría todo demasiado junto.

Para ello pensé en una solución similar a la de las dietas/rutinas. En este caso en vez de un lector, al pulsar la actividad se mostraría otra vista con controladores independientes e información relativa solo a la actividad en concreto que se haya seleccionado.

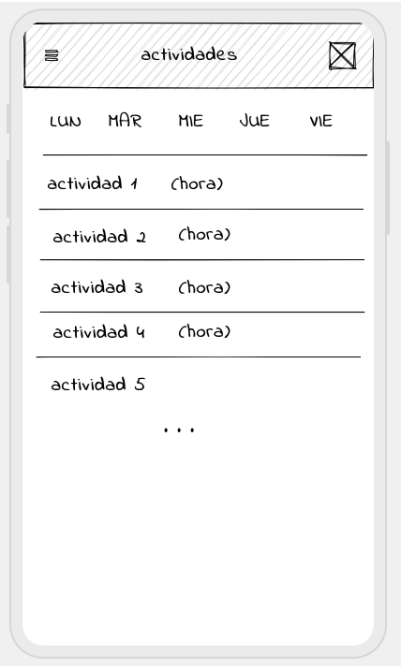

IMAGEN 25: VISTA ACTIVIDADES

El diseño final es por lo tanto muy similar a diferencia de que, en vez de pulsar sobre un switch, se nos abre otra vista con más información y donde poder apuntarnos.

IMAGEN 26: VISTA ACTIVIDAD

- Horarios

La ultima vista de la aplicación es la de los horarios. Esta vista cumple la función de representar en un calendario, las actividades a las que el usuario se ha apuntado.

Para lograr este objetivo es necesario replicar un calendario con las horas del día y los días de la semana. El usuario al pulsar sobre el día en concreto pude visualizar las actividades a las que esta apuntado y a qué horas.

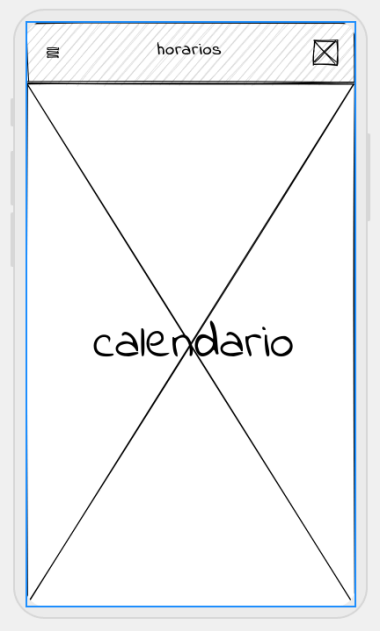

IMAGEN 27: VISTA HORARIOS

5.3. Capa 2 Lógica de negocio

Una vez tenemos el diseño de la interfaz gráfica del usuario podemos pasar a diseñar como se gestionará de forma lógica el funcionamiento de la app.

Recordar que esta es la capa puente entre la 1ª de presentación y la 3ª de datos.

En esta capa, primero hay que pensar cuales son las funcionalidades que queremos permitirle al usuario realizar con la aplicación.

Separando por vistas se puede ver que necesitamos controlar en cada una de ellas de forma individual.

- Inicio de sesión o creación del usuario

En estas dos vistas el funcionamiento es claro. Primero, desde la interfaz se recogen los datos del usuario y posteriormente se le pasan a la base de datos para comprobar si el usuario esta dado de alta.

Si esta comprobación resultara fallida entonces no se le podría dar acceso al usuario y por lo tanto no se le mostrara la primera vista.

Si el usuario no está dado de alta, entonces se puede pasar a la vista de creación de un nuevo usuario y los datos recibidos se introducirán en la base de datos.

- Sobre nosotros (1ª vista)

En esta vista la capa lógica juega un papel muy simple. La información que se muestra aquí no viene de la base de datos, esta información es parte del código de la aplicación por lo que no es necesario hacer ninguna conexión con la capa de persistencia.

Lo que debemos controlar aquí es que cuando el usuario pulse sobre alguno de los botones como el de las redes sociales o la localización, desde la capa lógica se redireccione al navegador o a la aplicación de mapas del dispositivo.

Un aspecto importante que si hay que controlar con cuidado es el cierre de sesión. Al pulsar sobre el botón indicado, la vista nos redireccionará a la de inicio de sesión, donde para volver a acceder se debe realizar todo el proceso anterior.

Mediante un diagrama de procesos BPMN [9] (Imagen 28) se puede visualizar como están enlazadas estas primeras vistas y la conexión con el servidor de bases de datos.

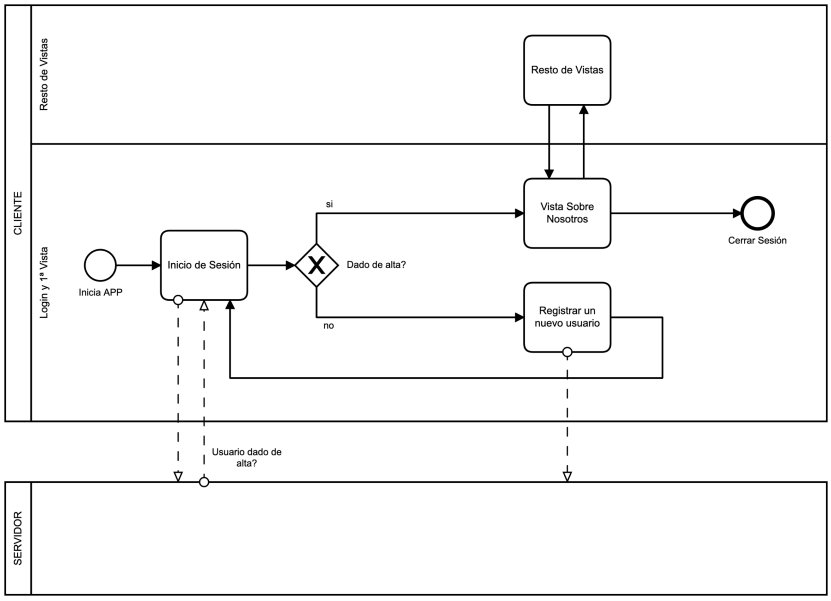

IMAGEN 28: PROCESO INICIO DE SESIÓN Y 1ªVISTA

- Novedades

En esta vista la capa lógica es más importante. Desde la interfaz de usuario se crea la vista con la estructura de cada novedad, pero los datos vienen desde la base de datos. La capa lógica lo que hace es que una vez se cargue la vista, solicita los datos específicos a la base de datos y los recupera para pasárselos a la capa de presentación.

En este caso no se controla si la información es de un usuario en específico o no ya que la vista muestra el mismo contenido a todos por igual.

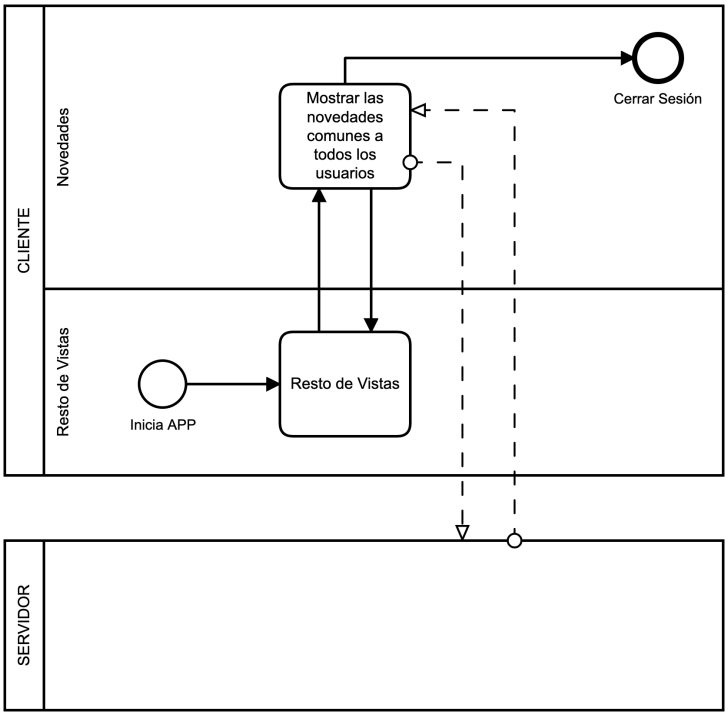

IMAGEN 29: PROCESO NOVEDADES

- Dietas

Para la vista de dietas el funcionamiento se asemeja en gran parte al de novedades. Sin embargo, en este caso el contenido a mostrar dependerá del usuario activo. No todos los usuarios tendrán las mimas dietas.

Para controlar esto la capa lógica debe de conocer en todo momento cual es el usuario activo.

Primero la GUI crea la estructura de la vista, pero los datos a mostrar deben de venir de la base de datos. La capa lógica de nuevo es la encargada de conectar con la de persistencia obtenerlos y pasárselos de nuevo a la 1ª capa. En este caso, los datos obtenidos tienen que ser exclusivamente de un usuario en concreto por lo que la capa lógica debe de comunicar a la de datos cual es el usuario activo y así poder obtener su información.

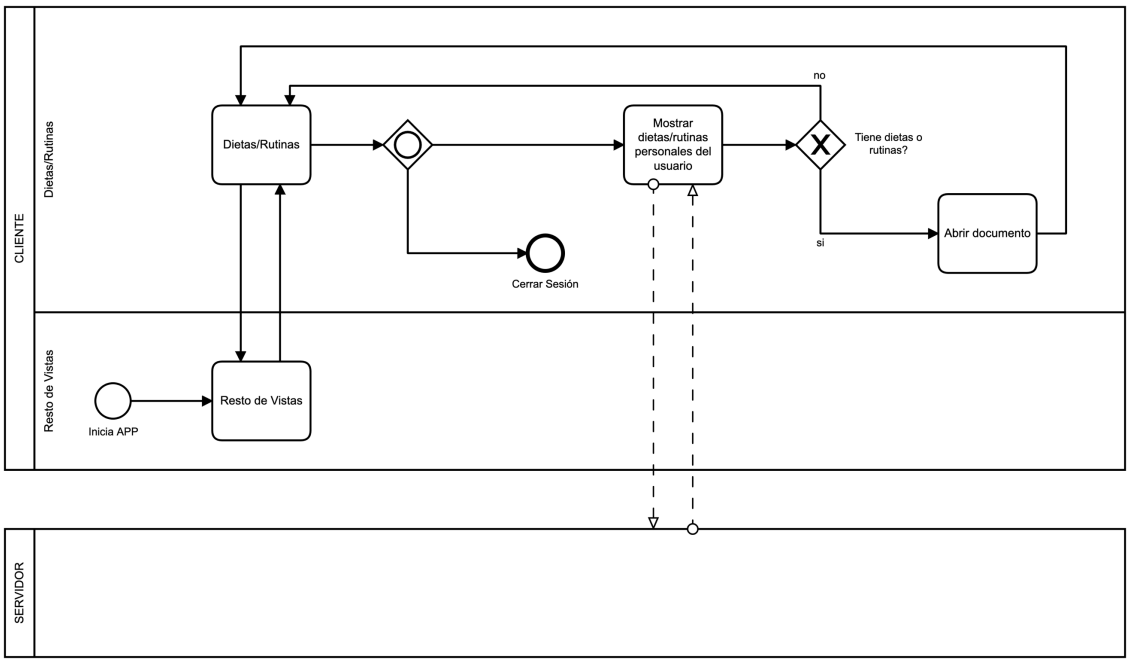

IMAGEN 30: PROCESO DIETAS/RUTINAS

- Rutinas

Para las rutinas el proceso es el mismo que para las dietas.

La información obtenida de la base de datos depende directamente del usuario que este activo y solo se le mostraran sus datos.

- Actividades

En cuanto a las actividades lo primero es que la GUI permitirá al usuario elegir el día de la semana. Dependiendo de este día la capa lógica mostrará unas actividades u otras.

Cuando se pulse en una actividad y el usuario proceda a apuntarse es cuando se hará la conexión con la capa de datos.

Una vez elegida la actividad y el día, la capa lógica le pedirá toda la información acerca de las actividades a la base de datos. La capa de lógica tiene que comprobar si la actividad ya estaba creada para modificar esa actividad o bien crear una nueva entrada en la base de datos.

En el caso de que la actividad ya estuviera registrada entonces comprueba el número de asistentes primero. Si el número de asistentes ha llegado al máximo el usuario entonces no puede asistir a esa actividad. La capa lógica no pedirá a la base de datos modificar esa actividad y mediante la GUI se mostrará una alerta indicándole al usuario que se ha alcanzado el aforo máximo.

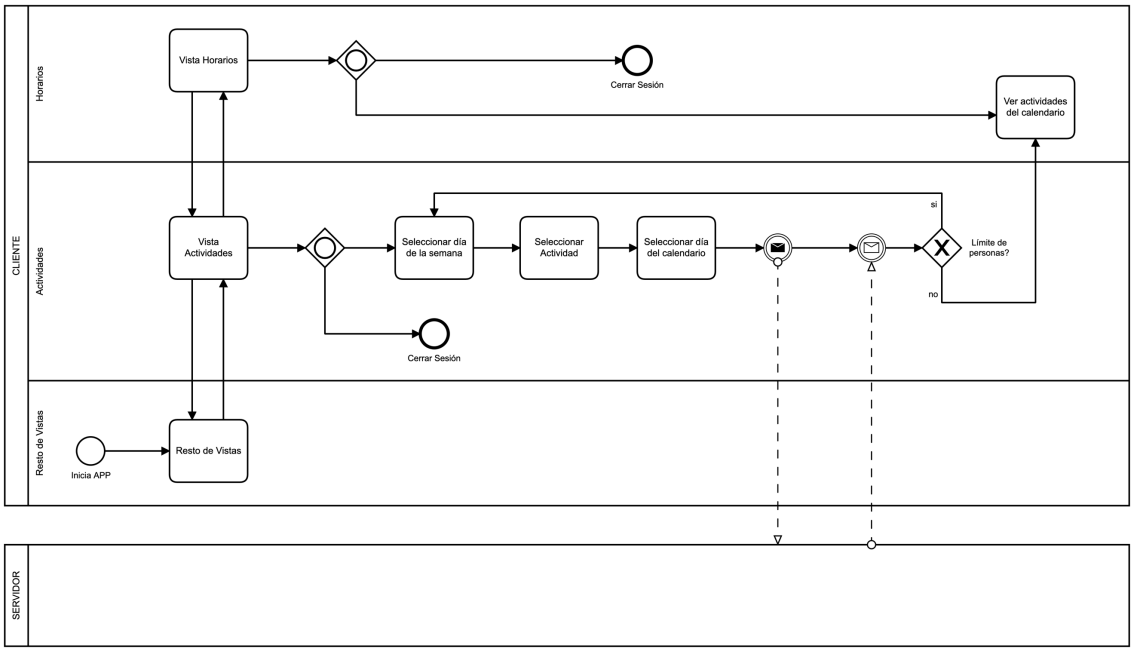

IMAGEN 31: PROCESO ACTIVIDADES Y HORARIOS

- Horarios

La vista de horarios simplemente re representan las actividades que se han dado de alta en la vista anterior.

La interfaz gráfica de usuario hará el mapeo de las actividades que la capa lógica registre en la base de datos. Por lo tanto, la conexión que existe en esta vista es la de las dos primeras capas, la de lógica de negocio conecta con la de presentación para mostrarle al usuario cuales son las actividades a las que esta apuntado.

No he creado un diagrama especifico de este proceso porque ya se puede visualizar en el diagrama anterior como se conectan las dos vistas, Horarios y Actividades.

5.4. Capa 3 Datos

Por último, queda el diseño de la base de datos que será donde se aloje toda la información de la aplicación y los ficheros necesarios.

Al estar haciendo uso de Firebase no es necesario montar el esquema ni la arquitectura del servidor de base de datos ya que su funcionamiento es diferente y más sencillo.

Dejando de lado el diseño del esquema de la base de datos, me he centrado principalmente en qué datos es necesario gestionar.

Los datos que se deben guardar y persistir son aquellos que tengan que ver primero con la autentificación de los usuarios. Se debe de guardar su correo electrónico o nombre de usuario y su contraseña.

A parte de esto, se puede guardar más información de los usuarios como es el nombre o características físicas, etc.

Por otra parte, debemos de guardar la información de las novedades. Cada novedad cuenta con una imagen, su título y una descripción.

Las dietas y rutinas vienen a ser muy parecido a las novedades, pero en este caso se deben de guardar los documentos pdf. Además de esto cada dieta o rutina debe de estar relacionada con un usuario ya que son exclusivas para cada cliente.

Tanto las novedades como las dietas/rutinas se le entregarán a la capa lógica de negocio para poder mostrarlas en la capa de presentación. La única diferencia es la ya comentada, novedades son generales y las otras dos estarán restringidas dependiendo del usuario activo.

Finalmente quedan las actividades. Para estas actividades guardaremos el nombre de la actividad, la hora y el día del calendario, además del número actual de apuntados. Este número de apuntados nos servirá para comprobar en la capa lógica si el usuario se puede apuntar a una actividad o no.

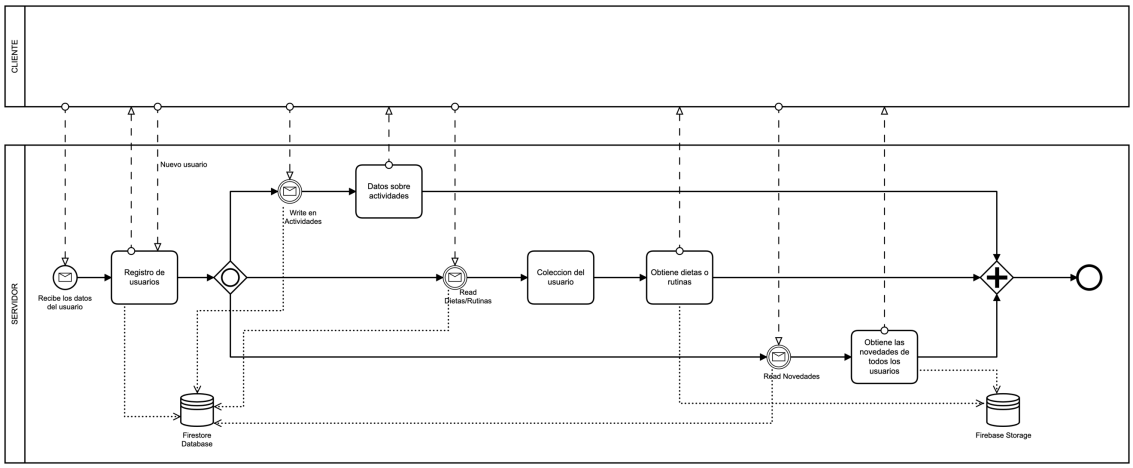

IMAGEN 32: PROCESO CAPA DE DATOS

6. Desarrollo de la solución propuesta

Partiendo del punto anterior se desarrollará cual ha sido el proceso a seguir para llevar a cabo la implantación de las ideas de diseño.

Siguiendo la misa estructura se dividirá la explicación entre las distintas capas de la arquitectura.

Antes de comenzar con cada capa, me gustaría explicar de manera gráfica cual es la estructura del proyecto, concretamente en xcode (ficheros y archivos).

Xcode y swift no obligan a tener una estructura concreta y a seguir unos protocolos de nombramiento de archivos o directorios. Sin embargo, el único consejo es utilizar la primera letra en mayúscula ya durante el desarrollo puede llegar a ser confuso si se utilizan funciones y demás.

La estructura por tanto es la siguiente:

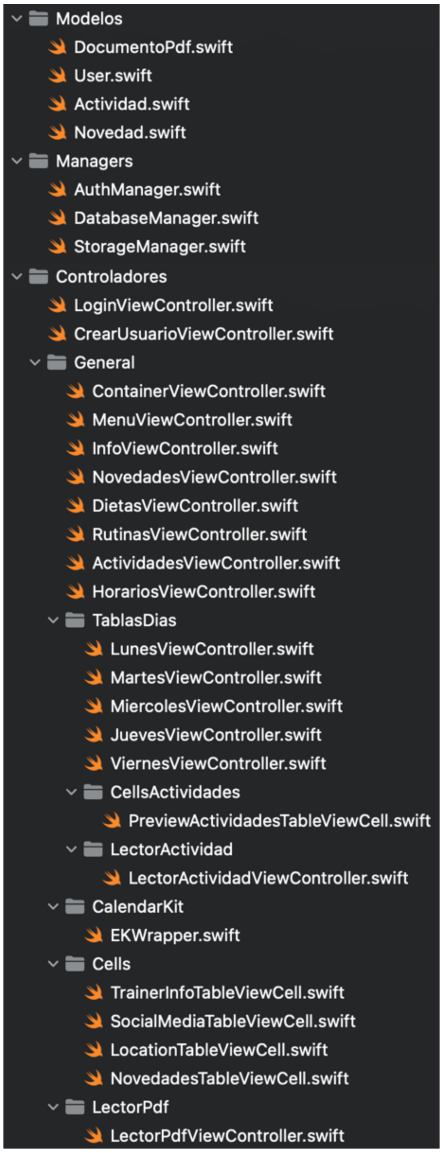

IMAGEN 33: ESTRUCTURA GENERAL DEL PROYECTO

Como se puede observar hay tres directorios principales, Modelos, Managers y Controladores.

En el primero, Modelos, irán los modelos de datos necesarios para la recepción de los datos provenientes de la base de datos. Pueden ser un documento pdf, datos de un usuario, actividades o novedades. Cada modelo cuenta por tanto con una estructura y datos distintos.

A continuación vienen los Managers. Estos archivos son necesarios para la conexión de la capa 2 con la capa de datos. En ellos se encuentran funciones que recuperan datos de la base de datos, insertan en esta misma, añaden nuevos usuarios o descargan archivos que tengamos alojados en Firebase.

El ultimo directorio principal es el de los Controladores.

Aquí se encuentran todos los scripts relacionados con la creación de la GUI, dándole la composición de los controles y su aspecto. Además de esto, en los scripts también está el código que se encarga de la capa 2. En él se conecta la interfaz con la capa de datos. Un ejemplo puede ser Login, donde el usuario introduzca los datos a través de la interfaz y la capa lógica llame a AuthManager para comprobar en la base de datos si ese usuario esta dado de alta o no.

6.1. Capa 1 Presentación

De la estructura anterior (Imagen 33) los ficheros que tienen que ver con cada una de las vistas son los siguientes:

- *LoginViewController* → 'Inicio de sesión
- *CrearUsuarioViewController* → 'Creación de usuarios'
- *MenuViewController* → 'Menú'
- *InfoViewController* → 'Sobre nosotros'
- *NovedadesViewController* → 'Novedades'
- *DietasViewController* → 'Dietas'
- *RutinasViewController* → 'Rutinas'
- *ActividadesViewController* → 'Actividades'
    - *LunesViewController*
    - *MartesViewController*
    - *MiercolesViewController* → unas actividades u otras depende del día
    - *JuevesViewController*
    - *ViernesViewController*
- *HorariosViewController* → 'Horarios'

Además de estos ficheros también se usan *LectorActividadViewController* y *LectorPdfViewController*. Estos dos archivos están relacionados con las ventanas emergentes a la hora de visualizar una rutina/dieta o cuando se pulsa sobre una actividad. Se explicará con más detalle a continuación.

El resto de ficheros con terminación *Cell* son subclases de otros componentes (tablas, colecciones) por lo que no se les considera vistas de la interfaz como tal. En el fichero *EKWrapper* están alojadas todas las funciones que se utilizan con CalendarKit y que permiten sincronizar el calendario del móvil con el de la app.

Para la capa de presentación primero tuve que decidir entre dos variantes que nos ofrece Xcode. Como he comentado antes yo he utilizado Swift 5. El entorno de desarrollo permite utilizar una herramienta llamada Storyboard. Esta herramienta no es más que un 'lienzo' donde podemos crear las interfaces de forma más visual incluyendo todos los controles necesarios como botones, etiquetas, imágenes, etc. Además de esto también la navegación de la aplicación por las distintas vistas.

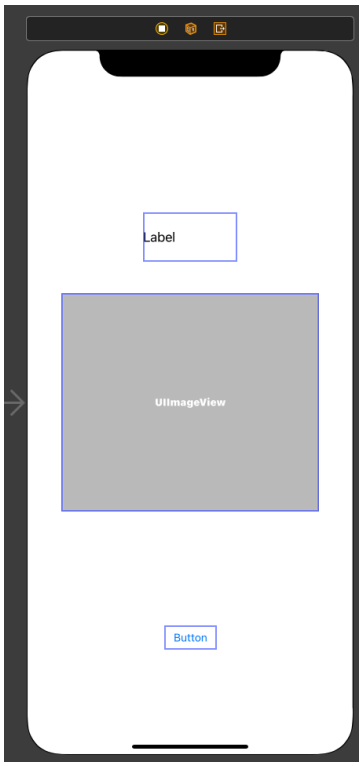

IMAGEN 34: STORYBOARD

La otra variante es no usar esta herramienta y eliminarla del proyecto. Con esta decisión ahora se tendrá que crear la interfaz de la vista de forma programática.

Puede parecer una desventaja ya que el trabajo es mayor. Sin embargo, el aspecto positivo de hacer la interfaz de forma programática es que se tiene un mayor manejo sobre los controles que se usen en la vista y por lo tanto sobre las interfaces en general.

La diferencia es simplemente que con el uso de Storyboard solamente hace falta incluir una referencia a ese control, por la otra parte, de forma programática debemos de ser nosotros los que primero creemos el control, le asignemos una posición y lo añadamos a la vista.

Mi decisión finalmente fue hacerlo de forma programática.

En la imagen siguiente (Imagen 35) se aprecia la diferencia entre las dos variantes donde a la izquierda se ve el código que se utiliza con el uso de storyboard y a la derecha la forma programática.

```
@IBOutlet weak var imagen: UIImageView!
```

```
private let imageView: UIImageView = {
    let imageView = UIImageView()
    imageView.contentMode = .scaleAspectFill
    imageView.backgroundColor = .white
    return imageView
}()
```

IMAGEN 35: STORYBOARD VS PROGRAMÁTICA

Ya decidido cómo se empezaría a trabajar inicié el desarrollo por las primeras vistas como son la de 'Login' y 'Creación del usuario'.

Para estas dos vistas se incluyen una foto que será el logo del gimnasio, dos campos de texto para el nombre y la contraseña, y finalmente dos botones para acceder o crear el usuario.

En la vista de creación de usuario se sigue la misma estructura ya que son muy parecidas.

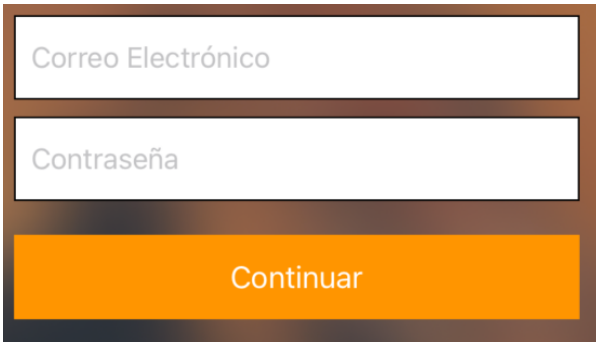

IMAGEN 36: INICIO DE SESIÓN

Una vez se accede a la aplicación la primera vista que parece es la de 'Sobre nosotros'. Esta vista tiene una gran cantidad de controles y objetos. En swift existe un objeto llamado TableView. Este objeto no es más que una tabla a la que le podemos incluir el número de filas y gestionar cada fila de forma independiente.

Para la vista necesitaba incluir imágenes, textos y el mapa. Una característica de estas tablas es que a cada fila se puede customizar de forma independiente. A las filas en swift se les denomina 'Celdas'.

El objeto TableView contiene una subclase que permite gestionar las celdas y darles el formato que nosotros deseemos. Por tanto, al utilizar una tabla, si se desea se puede crear una subclase de esta tabla que sea de tipo celda y ahí programar como queremos que sea su estructura.

Esta subclase se incluye en un fichero independiente al de la vista principal que contiene la tabla. A este fichero se le debe de indicar el tipo de subclase que es (tipo celda) (Imagen 37) y posteriormente podremos llamar a esta subclase desde la tabla y utilizar la estructura de las celdas customizadas que hemos creado.

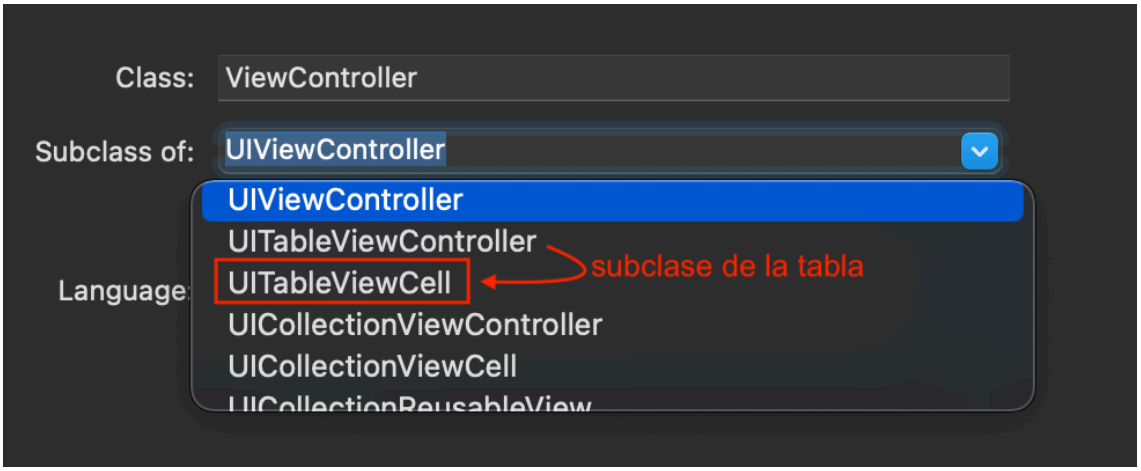

IMAGEN 37: CELDAS

Este fichero sigue una estructura muy parecida al de las vistas normales y es aquí donde se le va a dar posición a las imágenes, textos, botones, etc. pero no se incluyen los datos como tal. Únicamente se busca definir la estructura de la celda para después en la tabla utilizar esa estructura e incluir los datos.

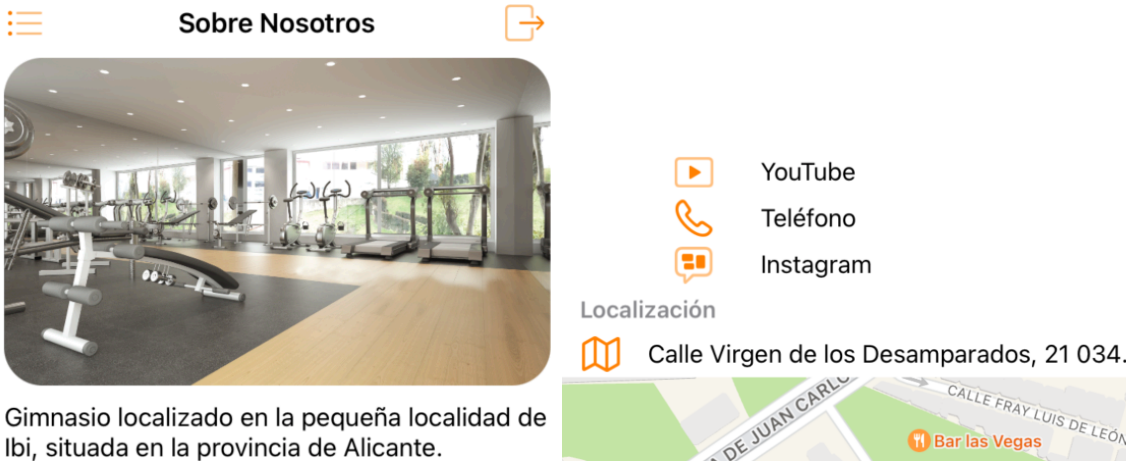

IMAGEN 38: VISTA PRINCIPAL

Básicamente tenemos una vista, que a su vez contiene una tabla y esta tabla contiene celdas que nosotros podemos programar en ficheros aparte.

Para esta primera vista también se crea la barra de navegación superior que se puede apreciar en la imagen anterior. Esta barra tendrá un botón para abrir el menú, el título de la vista y en la parte derecha un botón para cerrar sesión.

La barra de navegación se crea en esta vista inicial y el resto de vistas la heredan manteniendo el mismo funcionamiento de los botones cambiando únicamente el título de la vista.

En cuanto a la vista 'Novedades' la estructura de la interfaz es muy parecida. Se utiliza una tabla en la que cada celda es una novedad. A estas celdas se les incluye una

etiqueta para el texto, otra para el título y una imagen. Por tanto, si hay 3 novedades, el total de celdas será de 3.

En este caso, los datos que se mostraran en la vista vienen de la base de datos. En las siguientes capas explicare como es el proceso de obtención de los datos para mostrarlos en la interfaz.

La vista 'Dietas/Rutinas' también hace uso de una tabla con la diferencia de que aquí no es necesario customizar las celdas. Para esta vista la tabla es dinámica, esto quiere decir que según la cantidad de dietas que tenga ese usuario, el número de filas variara. Cada fila mostrará el nombre de la dieta y pulsando sobre ella se llamará a otra vista donde se muestra el documento.

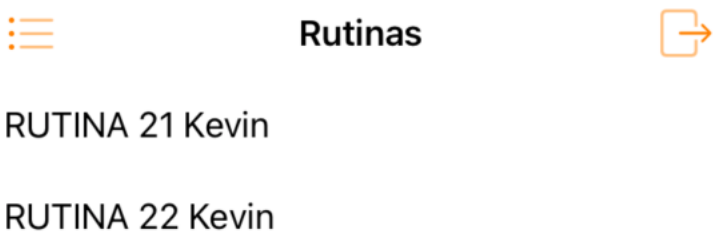

IMAGEN 39: VISTA DIETAS/RUTINAS

La vista para mostrar el documento es principalmente un lector de archivos, en este caso pdf y en la parte superior cuenta con una barra donde se muestra el nombre del archivo que se está leyendo.

Los datos de nuevo deberán de venir de la base de datos.

Para la vista 'Actividades' se utiliza primero una colección en la parte superior. Una colección es similar a la tabla, la diferencia es que en la colección no tenemos filas ni una estructura determinada. Lo que sí es parecido es que en las colecciones también se usan celdas, en este caso se especifica que la celda es subclase de una colección y no de una tabla.

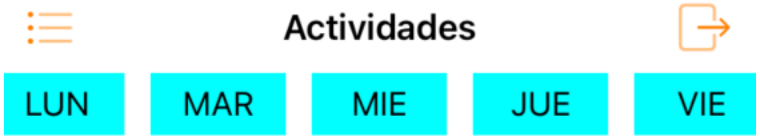

IMAGEN 40: COLECCIÓN DÍAS DE LA SEMANA

A estas celdas les podemos dar una estructura o posición determinada. A diferencia de la tabla aquí pueden estar dos a la misma altura, un debajo de otra, en general como se desee.

Al pulsar sobre cada una de estas celdas también se le puede indicar que realice una acción. En este caso, al elegir el día se mostrará justo debajo de la colección una tabla con las actividades asociadas a ese día.

Las tablas de las actividades se gestionan de la misma manera que las anteriores. En este caso se vuelve a hacer uso de una celda customizada ya que queremos incluir varios campos de texto, imágenes e iconos. Al pulsar sobre estas celdas llamaremos a otra vista independiente que nos va a mostrar la información referente a esa actividad.

En esta vista independiente se incluye una foto en la parte superior, la descripción de la actividad y el horario, y finalmente un botón para apuntarse.

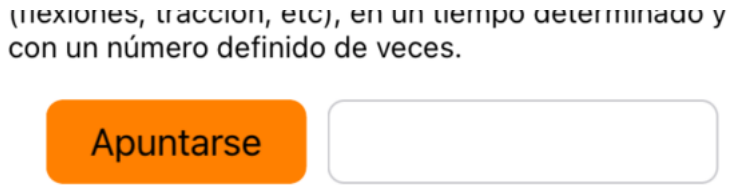

IMAGEN 41: APUNTARSE A ACTIVIDAD

Junto al botón tenemos un campo de texto. Pulsando sobre este campo se abrirá un date picker, que no es más que un pequeño calendario donde poder seleccionar el día para apuntarse a la actividad.

Para la última vista 'Horarios' he incluido un paquete de GitHub desarrollado por un usuario de esa plataforma. El nombre del paquete es CalendarKit [10] y con el podremos incluir en nuestro proyecto un calendario. En este calendario se pueden crear eventos, modificarlos o borrarlos. Gracias a este paquete la vista simplemente contendrá este calendario donde irán representas todas las actividades.

Para crear el calendario correctamente primero debemos de enlazarlo al calendario del móvil y de esa manera nos importara todos los eventos de nuestro calendario personal. Lo que creemos en nuestra app se nos mostrará en el calendario personal y viceversa.

IMAGEN 42: VISTA CALENDARIO

Importar un paquete a nuestro proyecto es muy sencillo. Simplemente debemos de ir a los ajustes del proyecto y en 'paquetes y dependencias' incluir un nuevo paquete.

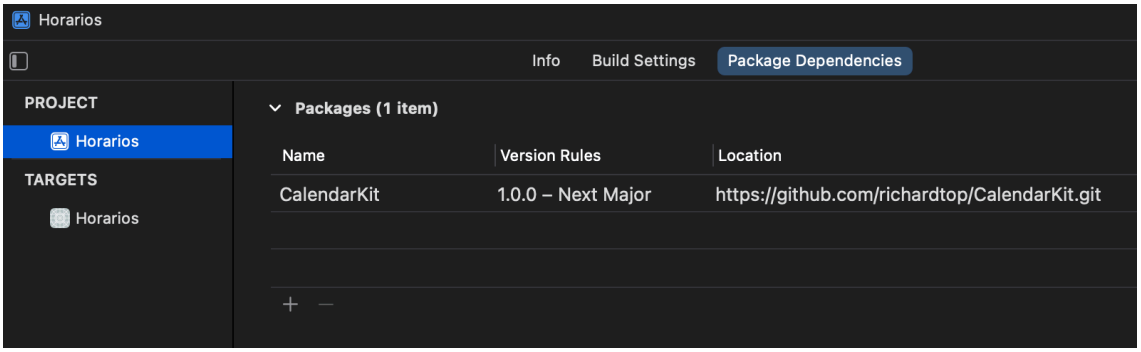

IMAGEN 43: PAQUETES XCODE

Pulsamos en el botón '+' y se nos abre una ventana donde pegaremos la URL de paquete y lo podremos instalar.

La ultima vista a implementar es la del menú. El menú se crea como una vista normal, igual que las anteriores, pero con un aspecto y funcionamiento distinto.

Al menú lo podemos llamar desde cualquiera de las vistas y dentro de él estarán las diferentes opciones que serán las vistas de la app.

Para el menú he utilizado una tabla con celdas customizadas ya que se incluye un pequeño icono y el nombre de la vista.

Al pulsar el botón izquierdo de la barra de navegación se nos desplegara el menú el dónde según la celda de la tabla que elijamos nos lleva a una vista u otra.

El funcionamiento del manu se explica en la capa de lógica de negocio ya que es la encargada de la navegación entre vistas.

IMAGEN 44: VISTA MENÚ

6.2. Capa 2 Lógica de negocio

Una vez se ha creado la interfaz gráfica del usuario, queda dotar de funcionamiento a estas interfaces.

El modelo de tres capas que he seguido permite separar cada capa para gestionarlas individualmente. Al no hacer uso del storyboard, las capas 1 y 2 van a tratarse casi al mismo tiempo, es decir, en el mismo fichero donde yo creo los controles y objetos de la interfaz estará todo el código de la capa lógica.

Si se hiciera uso del storyboard, la separación entre capas seria mayor ya que con los storyboards creamos la vista y después solo tenemos que referenciarla para poder conectar la capa lógica con la de presentación. Aun así, nada de esto supone un mayor problema ya que para el proceso de desarrollo se siguen un orden; primero creo la interfaz y después incluyo el código para dotarla de funcionamiento.

En esta segunda capa tenemos que conectar la interfaz con la base de datos. Para poder hacer uso de la base de datos primero debemos de importar los paquetes de firebase a nuestro proyecto. Como se ha visto en los horarios podemos incluir paquetes de terceros. En este caso el proceso de importación e instalación será diferente.

Para trabajar con firebase lo primero es conectar nuestro proyecto de xcode al proyecto de firebase. Siguiendo la documentación oficial de la herramienta, incamente hay que incluir unas líneas de código y un fichero que nos descargaremos del proyecto de firebase llamado ‘GoogleService-Info.plist’. Este fichero contiene la información necesaria para hacer el enlace entre los dos proyectos y comenzar a trabajar con las herramientas de Firebase.

Firebase permite trabajar con distintos módulos o herramientas. En mi caso solamente he necesitado hacer uso de tres.

- Firebase Authentication: Proporciona servicios de backend, SDK y bibliotecas de IU para autenticar a los usuarios en la app. Admite la autenticación mediante contraseñas, números de teléfono o proveedores de identidad como Google, Facebook y Twitter.
- Firestore Database: Es una base de datos flexible y escalable. Mantiene los datos sincronizados entre apps cliente a través de objetos de escucha en tiempo real y ofrece soporte sin conexión para dispositivos móviles.
- Firebase Storage: Servicio de almacenamiento de objetos potente y simple. Agrega la seguridad de Google a las operaciones de carga y descarga de archivos de las apps. Se pueden almacenar imágenes, audio, video, etc.

Una vez nos hemos enlazado a Firebase es hora de instalar estos módulos que vamos a utilizar. En este caso el proceso es diferente al de los paquetes anteriores. Para instalar estos módulos tenemos que hacer uso de los archivos Pod.

Para ello abrimos el intérprete de comandos o terminal de macOS. Nos dirigimos hasta la carpeta donde tenemos nuestro proyecto. Aquí vamos a introducir un comando para indicar que este proyecto va a hacer uso de archivos Pod para la instalación de paquetes.

- `$ pod init`

Una vez se ejecute este comando se crea un archivo llamado 'Podfile'. En él se introducen todos los paquetes que queremos instalar en nuestro proyecto. Basta con escribir *" pod 'nombre_del_paquete' "* y al finalizar ejecutar este archivo que comenzara la descarga e instalación.

- `$ open Podfile`
- `$ pod install`

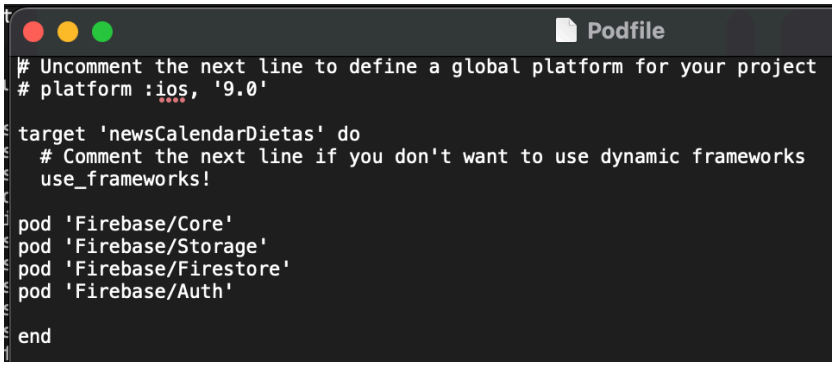

IMAGEN 45: PODFILES

El primer módulo que se utiliza es el de Auth, que permite gestionar las cuentas de los usuarios de una forma muy sencilla. En la vista 'Inicio de sesión' la capa lógica recogerá los datos del usuario y se los enviara a firebase para comprobar que, primero el usuario exista y segundo que sus credenciales son correctas. En el caso de que el usuario no este dado de alta entonces la vista 'Crear usuario' recogerá los datos y en este mismo modulo se creará un nuevo usuario con su correo y contraseña.

Adicionalmente a esto, si se crea el nuevo usuario en Auth la capa lógica también llamara al módulo Firestore Database. Es necesario crear una nueva entrada en la base de datos con la información del usuario ya que más adelante hará falta. La estructura de la base de datos se explica en la tercera capa.

Para la vista 'Sobre nosotros' no es necesaria ninguna conexión con la base de datos. La capa lógica en este aspecto solamente controla el momento en el que se pulsa una celda, ya que por ejemplo al pulsar el mapa se abrirá la aplicación de mapas del teléfono o al pulsar sobre las redes sociales se abrirá la aplicación en concreto.

En el resto de vistas, 'Novedades, 'Dietas', 'Rutinas' la capa lógica juega un papel más importante.

Primero para las novedades hay que hacer una conexión con la base de datos solicitándole todas las novedades que haya en ese momento. Las novedades incluyen imágenes y esas imágenes se guardan en el módulo de firebase llamado Storage.

Al iniciar la vista, la interfaz espera a recibir los datos que tiene que mostrar. La capa lógica por tanto llamará a la base de datos y pedirá la información de cada novedad (título, descripción y localización donde está alojada esa imagen). Una vez la reciba la devolverá a la interfaz para que se le pueda mostrar al usuario.

Lo mismo ocurre con las dietas o rutinas. La única diferencia en este punto es que la capa lógica mantendrá en todo momento cual es el usuario activo. Cuando llame a la base de datos pedirá la información de ese usuario en concreto.

Para guardar el usuario activo he utilizado los sistemas predeterminados de usuario que incluye swift. Estos sistemas predeterminados no son más que listas de propiedades que yo puedo crear al construir la aplicación. Estas listas funcionan como diccionarios en la que las entradas tienen una clave y su valor.

Para el usuario activo yo creo una clave llamada ‘user’ y el valor se introduce en la vista de ‘Inicio de sesión’. Por lo tanto, para que la capa lógica de negocio sepa cuál es el usuario activo tendrá que pedirle a esta lista que le devuelva el valor que hay en la clave ‘user’.

Una vez tenga ese valor, es decir el usuario, llamará a la base de datos y solicitara la información de únicamente este cliente.

En la vista ‘Actividades’ la conexión viene al seleccionar el día en el que se quiere apuntar a esa actividad. En este caso la capa lógica enviara la información relativa a esa actividad para que se escriban en la base de datos.

Además de esto, antes de escribir la actividad, la capa 2 primero comprueba de la base de datos la cantidad de apuntados. Si el número es igual al máximo entonces no pedirá la escritura en la base de datos e indicara al usuario que no es posible apuntarse.

Para la recepción de los datos he creado una serie de modelos que son útiles a la hora de gestionar esta información proveniente de firebase. Estos modelos los usará la capa lógica y en ellos se encuentra la estructura que deben de tener los datos de las actividades, dietas/rutinas, usuarios y novedades.

Si los datos recibidos no siguen esta estructura entonces la capa lógica no pude recibirlos. Dependiendo de la vista, los datos tienen que tener obligatoriamente la misa estructura para poder manejarlos correctamente.

Además de los modelos, en el proyecto también se han creado unos ficheros que sirven de manejadores para la base de datos. Es decir, en estos ficheros se encuentran todas las funciones que tienen que ver con la comunicación entre la capa lógica y la capa de datos. Si en alguna vista se solicitan X datos, entonces se hace uso de las funciones de estos manejadores para llamar a firebase.

La ventaja de usar los manejadores es que se estructura mejor el código evitando tener que duplicar partes de código por todos los ficheros.

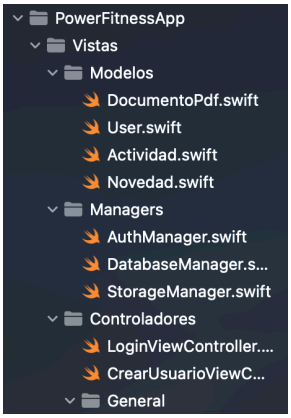

IMAGEN 46: ESTRUCTURA DEL PROYECTO

Lo último que quiero comentar acerca de la capa lógica es la navegación.

Como se ha visto en la interfaz hay varias vistas a parte del menú. En la lógica de negocio se controla el paso de una vista a otra. La primera vista en aparecer será la de inicio de sesión. De aquí se puede acceder a la vista de creación de un nuevo usuario y volver al inicio de sesión. Si el usuario introduce correctamente las credenciales se le dará paso al resto de vistas.

Para controlar el paso de una vista a otra se hace uso del menú por lo que todas las vistas deben de contar con un botón para abrir o cerrar este menú.

Las vistas de la aplicación (menos login y crear usuario) están encapsuladas en una clase llamada 'Container'. El uso que se le da a esta clase es el de permitir que todas las vistas puedan utilizar el menú lateral. Al abrir el menú podemos elegir a que vista del 'Container' queremos dirigirnos, por lo que dentro de esta clase se gestionara la navegación.

6.3. Capa 3 Datos

En la última capa el trabajo será montar la estructura de la base de datos, así como el almacenamiento de archivos y credenciales del usuario.

Empezando por la de las credenciales, he comentado que se usa un módulo específico para ello, por lo que en cuanto a la autentificación solamente debemos de centrarnos en un módulo concreto. Este módulo no es una base de datos convencional, en él solo se guardan los datos de los usuarios de la aplicación. Cuando desde la capa lógica se envíe un nuevo usuario se dará de alta en este módulo.

IMAGEN 47: FIREBASE AUTH

El segundo módulo es el del almacenamiento de archivos. En él se alojarán todos los archivos que se vayan a usar en la aplicación como las imágenes, las dietas o las rutinas.

Para las dietas y rutinas he creado una carpeta independiente para cada usuario ya que son exclusivas. La estructura por tanto es así de sencilla.

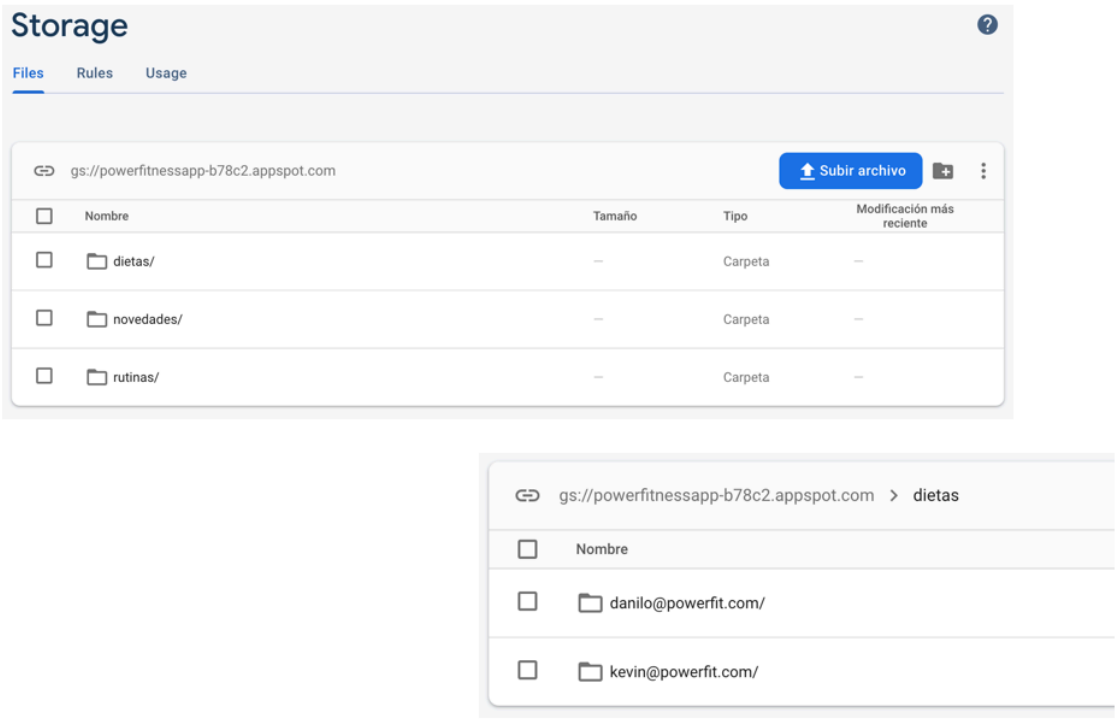

IMAGEN 48: FIREBASE STORAGE

El último módulo, pero no por ello menos importante es el de Firestore Database, la base de datos.

Esta base de datos no sigue le estructura típica de una base de datos relacional con tablas y relaciones. En firebase se trabaja por colecciones y documentos. Una colección puede tener varios documentos y en cada documento guardar información. A su vez los documentos pueden guardar colecciones.

La estructura que se sigue es por lo tanto en forma de árbol. Las colecciones raíces alojan documentos y estos documentos pueden alojar datos o más colecciones que alojan más documentos, …

Para esta aplicación son necesarias tres colecciones raíces.

La primera para guardar la información de los usuarios. Esta colección tendrá tantos documentos como usuarios haya. El nombre de la colección será el email del usuario. De esa manera desde la capa lógica al solicitar información de un usuario se accederá a su documento independiente.

Para cada documento de usuario se guardará información relacionada a ese cliente. Además de esto también habrá tres colecciones más, una para las actividades, otra para las dietas y otra para las rutinas. La cantidad de documentos de estas colecciones dependerá de las actividades a las que se haya apuntado o las dietas/rutinas que tenga.

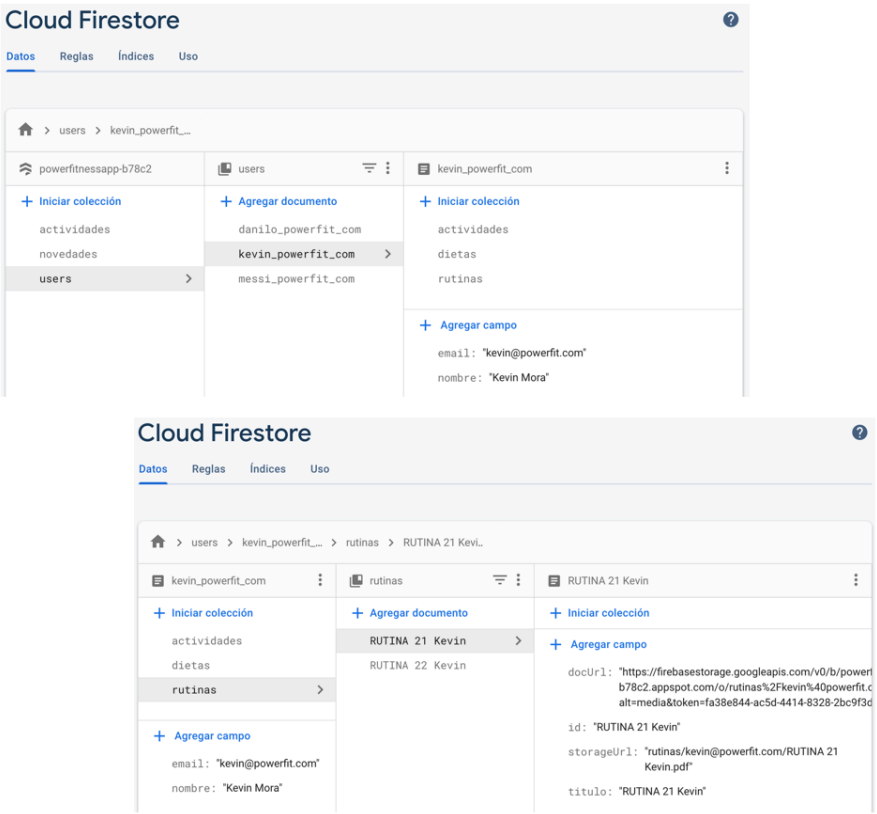

IMAGEN 49: COLECCIÓN USUARIOS

Para las otras dos colecciones raíces (novedades y actividades) no es necesario guardar un documento por usuario ya que los datos son iguales para todos los clientes.

En estos documentos se guarda la información específica de cada actividad o de cada novedad.

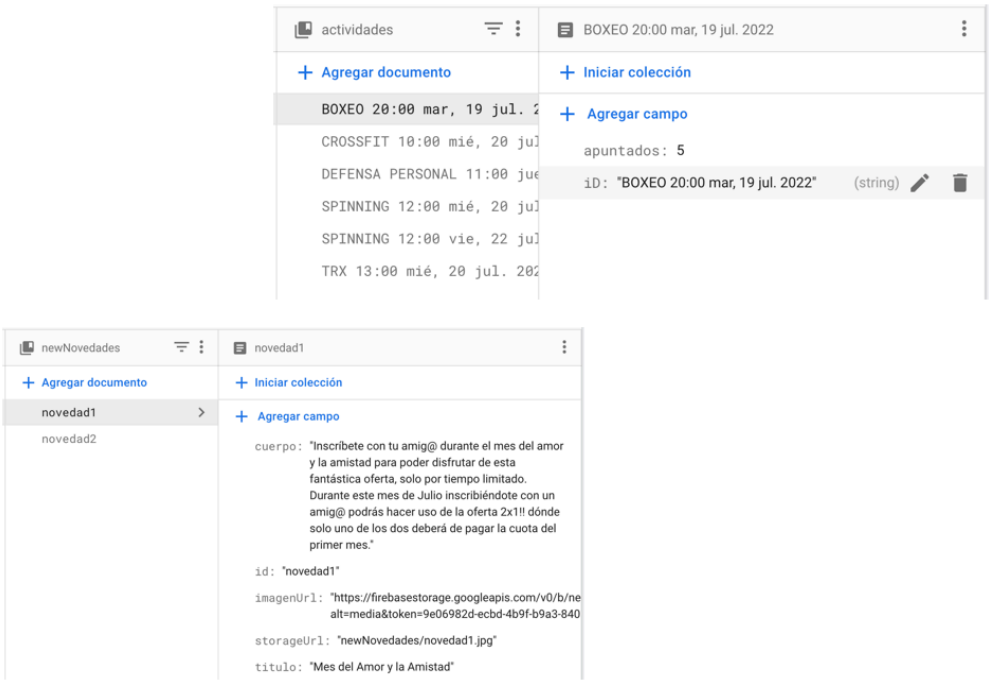

IMAGEN 50: COLECCIÓN ACTIVIDADES/NOVEDADES

Para que se vea de una forma más clara las colecciones tendrán la siguiente estructura. Entre paréntesis se indican los atributos de cada documento.

- users
    - usuario_1
        - actividades
            - actividad_1 *(id, título, fecha y hora)*
            - …
        - dietas
            - dieta_1 *(id, rutaStorage, título)*
            - …
        - rutinas
            - rutina_1 *(id, rutaStorage, título)*
            - …
        - información del usuario *(email, nombre)*
    - usuario_2
        - …

- novedades
    - novedad_1 *(id, rutaStorage, cuerpo, título)*
    - novedad_2
    - …

- actividades
    - actividad_1 *(id, número de apuntados)*
    - actividad_2
    - …

Si hablamos de la conexión de la capa 3 con la capa 2 se hace desde los Managers vistos anteriormente. Aquí hay varias funciones como son insertar un usuario, obtener las dietas, acceder los datos de una actividad, obtener las novedades, etc.

La capa de datos realmente no hace ningún control, el control lo lleva la capa lógica de negocio. Esta es la encargada de identificar el usuario activo para así obtener los datos concretos o controlar que no se pase del límite de apuntados en una actividad.

A la base de datos únicamente se indica de donde se quiere obtener información y donde se quiere insertar información. Para ello la estructura debe de ser tal y como hemos visto en este último punto.

7. Resultado

En este apartado realizare una demostración del estado final de la aplicación realizando un recorrido por las distintas vistas que hay.

7.1. Demostración de la aplicación

La primera vista al iniciar la app es la de ‘Inicio de sesión’ y tendrá el aspecto de la siguiente captura (Imagen 50). La vista para crear un nuevo usuario es realmente muy similar por lo que he optado por no incluirla.

IMAGEN 51: VISTA FINAL INICIO DE SESIÓN

Una vez iniciada la sesión se abrirá la primera ventana de la aplicación como tal y esa es la de ‘Sobre nosotros’ con el aspecto siguiente.

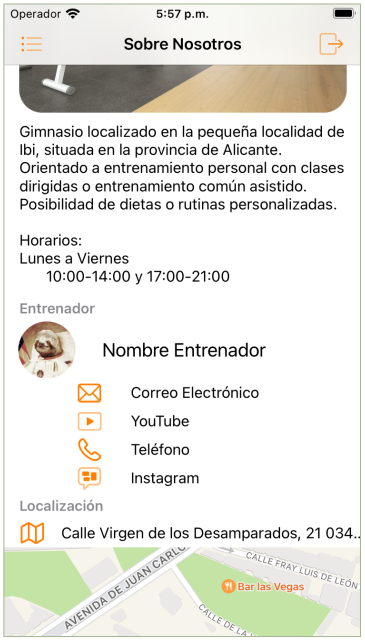

IMAGEN 52: VISTA FINAL SOBRE NOSOTROS

Pulsando sobre el icono superior izquierdo (*lista con viñetas*) se puede abrir el menú con el que navegar al resto de vistas. A su vez, con el de la parte derecha (*cuadrado con flecha*) se puede cerrar la sesión, se presentará una alerta para confirmar que así se desea y sacara al usuario al inicio de sesión. El menú tiene el siguiente aspecto.

IMAGEN 53: VISTA FINAL MENÚ

Siguiendo el orden del menú, la vista que sigue es la de 'Novedades' (Imagen 53).

IMAGEN 54: VISTA FINAL NOVEDADES

Las próximas vistas de la aplicación son las de ‘Dietas’ y ‘Rutinas’. Para esta demostración solo he incluido una (Imagen 54).

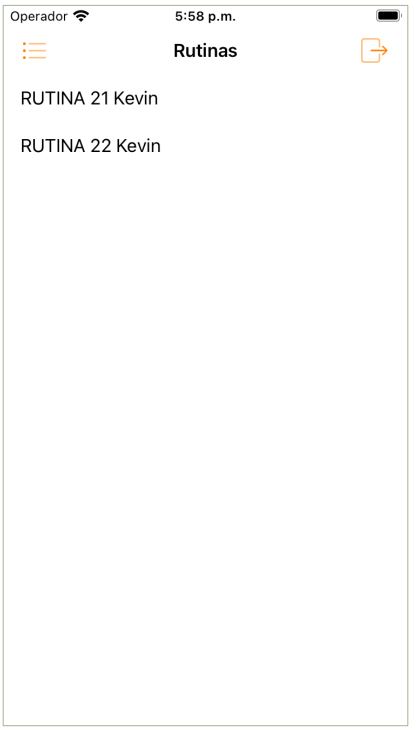

IMAGEN 55: VISTA FINAL RUTINAS

Pulsando sobre una de las rutinas se abre el lector de pdf (Imagen 55) con el que se puede visualizar y seguir la rutina que se desee.

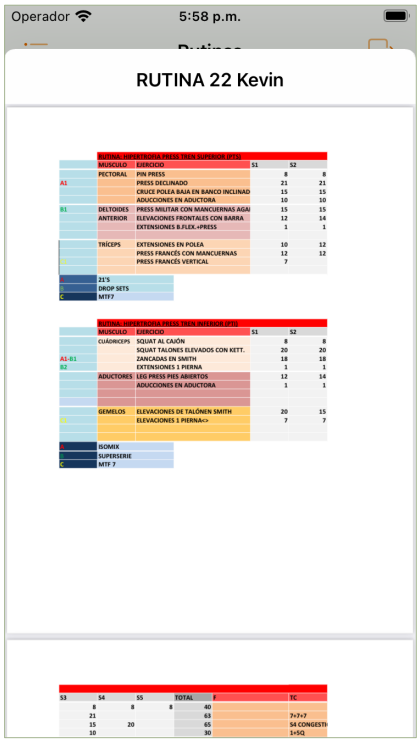

IMAGEN 56: LECTOR RUTINAS

Continuamos por la vista de ‘Actividades’. En ella se puede seleccionar un día de la semana donde aparecerán todas las actividades disponibles para ese día (Imagen 56).

IMAGEN 57: VISTA FINAL ACTIVIDADES

Si se pulsa sobre alguna actividad se abre una vista con infamación sobre esa actividad en concreto y donde se puede elegir el día y apuntarse (Imagen 57).

IMAGEN 58: INFORMACIÓN ACTIVIDAD

Para elegir el día basta con pulsar sobre el campo de techo a la derecha del botón ‘apuntarse’ y aparecerá un mini calendario como es el siguiente.

IMAGEN 59: ELEGIR FECHA PARA ACTIVIDAD

Finalmente, una vez se hayan elegido las actividades y la app haya permitido apuntarse a ellas, se puede pasar a la última vista ‘Mi Horario’, donde se pueden ver representadas todas las actividades que tenga ese usuario.

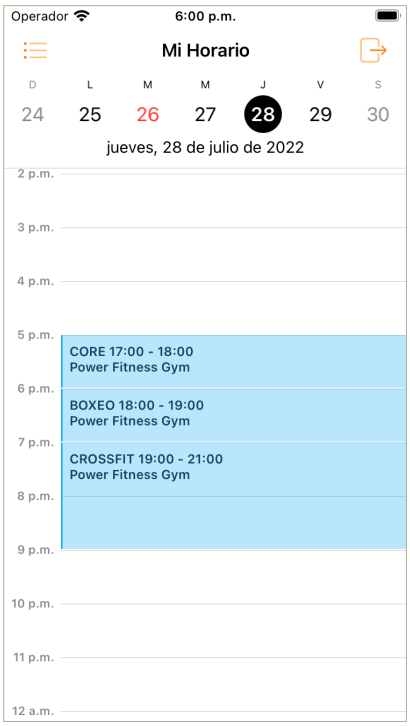

IMAGEN 60: VISTA FINAL HORARIO

8. Conclusiones

Gracias a este proyecto he podido aprender mucho sobre el desarrollo de apps para dispositivos móviles y puedo decir que he sentado unas bases iniciales para en un futuro poder continuar con proyectos mucho más ambiciosos. Si bien solo he trabajado con una tecnología en específico como es la de Apple, el conocimiento obtenido me sirve para poder tomar diferentes caminos y desarrollar en un futuro aplicaciones multiplataforma o nativas para Android.

He de decir que realmente he disfrutado desarrollando la aplicación ya que es algo que durante el grado no se ve en profundidad y desde hace tiempo tenía ganas de aprender. Si comparo los conocimientos que tenía sobre esto cuando inicie el proyecto con los que tengo ahora, la verdad que no pensaría que en relativamente poco tiempo podía ser capaz de desarrollar una app de principio a fin.

Tanto con el lenguaje Swift como con los entornos o herramientas empecé de cero. Lo aprendido en el grado ayuda a comprender mucho mejor el funcionamiento de todo esto, y aunque la etapa de aprendizaje se me hizo un poco cuesta arriba, al final he sido capaz de trabajar mucho mejor y con más soltura.

Hay una gran comunidad de desarrolladores por internet y una gran cantidad de información sobre este campo de la informática, por lo que estoy seguro de que este es el punto de partida para poder seguir ampliando mis conocimientos sobre la materia.

8.1. Problemas durante el desarrollo

Al iniciar de cero en el mundo del desarrollo de aplicaciones para móviles me han surgido una gran cantidad de dudas y errores durante todo el proyecto. Principalmente me he apoyado de internet ya que no hay mejor lugar para buscar información acerca de lo que sea. De entre todas las páginas que he usado para corregir errores me gustaría comenta algunas de ellas.

Primero para cualquier duda acerca del código la mejor web y de donde más información he utilizado es Stackoverflow [11]. La web permite a los usuarios realizar preguntas sobre cualquier tema así que muchos de mis fallos de código tenían solución ahí. Además de esta página también he buscado información en diferentes repositorios de GitHub [12], que cuenta con muchísimos desarrolladores que comparten sus proyectos y de donde he podido descubrir como estructuran su código de swift o como implementan algunas de las funcionalidades necesarias para mi proyecto.

Para comentar uno de los mayores problemas que tuve y que más retraso mi progreso fue al instalar los Podfiles por primera vez en un proyecto de xcode.

El problema venia al necesitar de la instalación de Homebrew [13] en mi sistema. Homebrew es un sistema de gestión de paquetes para sistemas operativos como macOS o Linux. Gracias a este sistema podemos instalar cocoapods que es el

administrador de dependencias de Swift y con el que vamos a crear los Podfiles e instalarlos en el proyecto.

Por alguna razón, la última versión de Homebrew no me permitía instalar la última versión de cocoapods y sin esta versión no podía incluir las dependencias necesarias para trabajar con Firebase. Al final, y después de muchos intentos, me instalé una versión especifica de Homebrew anterior a la que se instalaba por defecto con la que ya pude instalar la versión de cocoapods que necesitaba.

Teóricamente, al haber instalado todo correctamente no me debería de haber dado ningún problema. El caso es que no encontré en ningún sitio que esa versión primera de Homebrew no permitía instalar cocoapods correctamente. Al final decidí probar con otras versiones y si funcionó. Quizás el problema era más de mi sistema operativo, al que le faltaban instalar otras herramientas, pero no indague mucho más.

8.2. Relación del trabajo desarrollado con los estudios cursados

En el grado hay varias asignaturas que a la hora de realizar el proyecto me han servido de gran ayuda.

Una de ellas es bases de datos que, aunque para mi aplicación no se haga uso de las bases de datos relacionales, tener un conocimiento teórico me ha servido para estructurar mejor Firebase. Además de esto entender cómo funciona una base de datos relacional ayuda a poder entender el resto de bases de datos como es este caso o el de bases de datos por grafos, etc.

En cuanto al frontend he cursado alguna asignatura especifica que trataba las interfaces gráficas de usuario. Estas asignaturas no tienen mucha relación con las interfaces de swift, pero realmente el valor que me han aportado es a la hora de realizar el diseño. La disposición de los controles o como hacer una interfaz más intuitiva para el usuario.

Sin dudas una de las asignaturas o bloque de asignaturas más importantes son las de programación. En estas asignaturas se aprende desde la base toda la parte teórica de la programación, base que después ayuda mucho a la hora de aprender un lenguaje nuevo. El aprender la teoría y modelos de programación ayuda a que si se aprende un lenguaje nuevo solo necesitemos de familiarizarnos con la sintaxis y no tanto con su funcionamiento en general.

Para finalizar comentar la asignatura ingeniería del software. Para este proyecto no hay asignatura que aporte más que esta. Es verdad que el desarrollo lo he llevado yo solo, pero las estrategias de trabajo como SCRUM son útiles a la hora de comenzar con cualquier proyecto que se base en el desarrollo de software.

Una de las herramientas que se enseñan en ingeniería del software es gitlab (similar a GitHub), para el control de versiones, la cual es un apoyo muy significativo a la hora de desarrollar software. En mi caso no he utilizado ninguna de estas herramientas, pero está claro que es algo que me podría haber ayudado considerablemente.

9. Trabajos futuros

Para finalizar la memoria me gustaría comentar algunos desarrollos que se podrían incluir en la aplicación en un futuro.

Una vista que considero que podría ser útil incluirla puede ser la de un chat grupal con todos los miembros del gimnasio.

La vista contendría un chat como puede ser el de un grupo de WhatsApp o Telegram donde los usuarios podrían realizar preguntas o comentar temas de interés general del gimnasio. De esta manera no haría falta de ninguna aplicación de mensajería típica y la aplicación sería mucho más completa, donde todo lo necesario y relacionado con el gimnasio estaría incluido ahí.

La vista podría tener un único chat con todos los clientes/entrenadores o incluso chats individuales de cada cliente con los entrenadores. Incluir una foto de perfil, otra vista en el menú para ver la información del perfil, etc.

Otro desarrollo bastante útil es el de usuario administrador o incluso app de administrador.

Actualmente para incluir la información en la base de datos se debe de hacer uso de la consola de Firebase directamente. Honestamente el funcionamiento de esta consola no es nada difícil de manejar. Incluir los ficheros en los directorios es muy sencillo, habilitar o deshabilitar a los usuarios también es sencillo y para la base de datos, el incluir nuevas colecciones o documentos se necesita ir con cuidado, pero tampoco es extremadamente difícil.

De todas formas, siendo totalmente realista, los entrenadores de un gimnasio quizás no vayan a utilizar directamente Firebase. Pueden o eliminar datos delicados o insertarlos con errores. Por ello creo que una cuenta de administrador seria de ayuda a la hora de incluir las dietas/rutinas, novedades o modificar las actividades.

Podría haber dos opciones, o bien se crea una cuenta de administrador que tenga acceso a vistas especiales para añadir esta información e incluirla en la base de datos, o bien se puede crear otra app a parte que solo este enfocada en esto.

Quizás bajo mi punto de vista la segunda opción es mejor ya que se separa el código y se gestiona de forma independiente la una y la otra. Esta app por ejemplo podría tener un listado de los usuarios dados de alta, una vista para subir las dietas o rutinas, también una herramienta para crear novedades a modo de publicaciones como un blog, etc.

10. Referencias

10.1. Bibliografía

[1] Página web de Ufit365: https://ufit365.com

[2] Página web de TCP-MATCHPOINT: https://tpcmatchpoint.com/index.html

[3] Lucid App – Diagrama de casos de uso: https://lucid.app/lucidspark/

[4] Flutter: https://flutter.dev/multi-platform/mobile

[5] KoltlinMM: https://kotlinlang.org/docs/multiplatform.html

[6] Firebase: https://firebase.google.com/docs/ios/setup

[7] Draw.io: https://app.diagrams.net

[8] Uizard – Mockups: https://app.uizard.io/p/sbgsUEO--

[9] Camunda Modeler – BPMN: https://bpmn.io/modeler/

[10] Calendar Kit, Richard Topchii, GitHub - https://github.com/richardtop/CalendarKit.git

[11] Stackoverflow - https://stackoverflow.com/questions

[12] GitHub - https://github.com/topics/swift

[13] Homebrew - https://brew.sh/index_es

10.2. Acrónimos

1: Android; Sistema operativo móvil basado en el núcleo Linux y otros softwares de código abierto.

2: iOS; Sistema operativo móvil de la multinacional Apple Inc.

3: Storyboard; Función integrada en Xcode que permite ensamblar visualmente las distintas pantallas que componen una aplicación de iOS y la ruta de navegación a través de esas pantallas.

4: Interprete de comandos; el intérprete de comandos es el programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el sistema operativo.

5: Podfiles; Archivos de texto que se utilizan para el desarrollo de los códigos fuente de programación de software. Este formato hace uso del lenguaje Perl.

6: SQLite; SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C.

7: Framework; estructura conceptual y tecnológica de asistencia definida, que puede servir de base para la organización y desarrollo de software.

8: Core Data; Core Data es un marco de persistencia y gráfico de objetos proporcionado por Apple en los sistemas operativos macOS e iOS.

9: Backend; Procesos que se realizan internamente para programar las distintas funciones de un programa y que no son visibles para los usuarios.

10: Cloud Storage; modelo de almacenamiento de datos que se alojan en uno o más servidores de forma virtual.

11: GUI; Interfaz gráfica de usuario o en inglés *Grafical User Interface*

12: Mockup; Un mockup es un modelo o un prototipo a mano alzada que se utiliza para exhibir o probar el diseño de un software.

13: BPMN; Notación gráfica estandarizada que permite el modelado de procesos de negocio.