UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Satisfacción y optimización de restricciones para la personalización de rutinas de gimnasio

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Mesas Martí, Darío

Tutor/a: Sapena Vercher, Oscar

CURSO ACADÉMICO: 2022/2023

Agradecimientos

Agradezco a mis padres y a mi hermana por todo el apoyo recibido desde que decidí iniciar este grado.

Por otro lado, también a los compañeros que he tenido en tercero y éste último año, principalmente a Aarón y a Josemi.

Satisfacción y optimización de restricciones para la personalización de rutinas de gimnasio

Resumen

Uno de los mayores problemas cuando se empieza en el gimnasio es familiarizarse con las máquinas. Durante las primeras semanas, y con la ayuda de amigos y entrenadores, es posible ya plantearse unos objetivos concretos a conseguir. Es ahí cuando llega otro problema: hay que empezar a organizar las rutinas de entrenamiento. Mediante algunas aplicaciones o una búsqueda por internet, es posible encontrar miles de rutinas distintas. La principal dificultad consiste en descubrir cuál es la que más conviene utilizar, cuál se adaptan mejor al horario disponible o a las limitaciones de tiempo y cuál respeta más las preferencias personales, como las de evitar cierto tipo de ejercicios.

Muchos de estos problemas se solucionan con una rutina personalizada, contratando a un entrenador personal. En este trabajo se propone el diseño de una herramienta para dispositivos móviles que sea capaz de generar rutinas altamente personalizadas en función del grupo de músculos que el usuario desee ejercitar, del tiempo máximo de entrenamiento disponible y de sus preferencias. Para ello, la herramienta contará con una base de datos de ejercicios, cada uno de los cuales especificará el tipo de ejercicio, qué grupos musculares trabaja y en qué medida, los tiempos de descanso tras su realización y el beneficio obtenido.

La base de datos de ejercicios junto con las restricciones y preferencias del usuario se utilizarán como entrada para generar, mediante algoritmos de optimización y satisfacción de restricciones, una rutina que logre los objetivos planteados al mismo tiempo que maximiza el beneficio conseguido dentro de las limitaciones temporales.

Palabras clave: gimnasio, rutina de entrenamiento, optimización, satisfacción de restricciones, aplicación móvil

Abstract

One of the most common problems when starting the gym is getting used to the machines. Over the first weeks, with the help of friends and trainers, it is possible to start thinking about the objectives to follow. That is when there is another problem: you must start doing your routines. Using some apps or with an internet search you will find a thousand of different routines. The most difficult part lies in choosing which routine to follow, which is more friendly on your schedule, and which one respects more your preferences, such as avoiding certain exercises.

Most of these problems are solved with a custom routine, paying for a personal trainer. This thesis proposes the design of a tool for mobile devices that can generate highly personalized routines based on the group of muscles that the user wants to exercise, the available time and your preferences. To do this, the app will have a database of exercises, each of which will specify the type of exercise, which muscle groups work and how much, rest times after completion and the benefit obtained.

The exercise database together with the restrictions and preferences of the user will be used as input to generate, through optimization algorithms and constraint satisfaction, a routine that achieves the stated objectives while maximizing the benefit achieved within the time constraints.

Keywords: gym, training routine, optimization, constraint satisfaction, mobile application

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

El gimnasio siempre ha sido un lugar donde poder hacer ejercicio sin necesidad de tener los materiales a mano, se paga una cuota y se es libre de acceder para hacer ejercicio. Cuando se entra, se observa que hay muchísimas máquinas a utilizar, y lo normal si no se tiene experiencia, es tener curiosidad de conocer el funcionamiento de estas.

Cuando ya se es más cercano al gimnasio, habiendo entrenado más veces, es cuando el usuario se da cuenta de que necesita una rutina. Hacer una rutina no es algo muy complicado, pero una rutina genérica no será tan efectiva como una rutina óptima que se adapte a tu tiempo de entrenamiento y a los músculos que te apetezca entrenar.

Dado que para crear una rutina efectiva es necesario tener un conocimiento de los posibles ejercicios que existen, así como saber a qué parte del cuerpo afectan, no es algo que pueda hacer todo el mundo sin informarse antes.

Es por eso por lo que he creado esta aplicación, que con saber lo que se quiera entrenar y el tiempo disponible, creará una rutina óptima.

1.1. Motivación

Yo empecé en el gimnasio después de las fallas de 2021. No había hecho nada de ejercicio en mucho tiempo, a parte de las clases de Educación Física en el instituto y jugar al fútbol ocasionalmente.

Convencí a un amigo para apuntarnos juntos, porque solo no me atrevía tanto a ir, y estuvimos unos días probando las máquinas para familiarizarnos un poco con ellas.

Cuando llevas ya unos días, te das cuenta de que vas a necesitar organizarte tus rutinas de una forma más eficaz, o al menos yo lo notaba así. Quería que me mereciera la pena ir al gimnasio, aprovechando el tiempo para obtener buenos resultados. Aquí venían los problemas: hay días que tenía menos tiempo para entrenar, por lo cual sabía que la rutina que había encontrado por internet no me iba a ser tan eficaz como pensaba, si tengo treinta minutos menos no sería capaz de hacer los mismos ejercicios y no hacía el entrenamiento entero.

Esto no me gustaba nada, dado que esos ejercicios ya estaban perdidos; si no los podía hacer el día que tocaban, no iba a hacerlos el día siguiente debido a que el día siguiente iba a ser un entrenamiento enfocado a otras partes del cuerpo, y ya tendría carga de los ejercicios del día anterior.

Esto puede parecer un caso puntual, pero era más frecuente de lo que parece. Al final, uno o dos días de la semana terminaban sin ser tan aprovechados como se debería por simple falta de tiempo, lo que era necesario cambiar.

En ese momento no tenía mucha idea, entonces tampoco le prestaba tanta atención, pero conforme vas progresando en el gimnasio te vas dando cuenta de que quieres progresar más, y junto a la alimentación, la rutina es lo más importante del gimnasio. Dejando todo el tema de

la alimentación, pasamos a centrarnos en la rutina, crear una rutina que se adapte dependiendo el tiempo del que dispongas.

En tercer año de la carrera, di una asignatura llamada Técnicas de Optimización, donde una de las partes que me pareció más interesante fue el tema de la Investigación Operativa, una rama de las matemáticas que, mediante modelos matemáticos, se modelan y resuelven problemas obteniendo la solución óptima. Esto me llevó a pensar todos los usos que puede traer la posibilidad de tener la capacidad de encontrar un problema cuotidiano, modelizarlo y poder resolverlo.

Ya que en varias asignaturas de la carrera hemos programado, me pareció una buena idea intentar crear una aplicación que fuera capaz de modelizar unos ejercicios, y dado que el gimnasio es un mundo en el que cada vez estoy más metido, pues decidí hacerlo como TFG.

1.2. Objetivos

Por un lado, el objetivo principal de esta aplicación es poder compartir con los usuarios una manera de hacer rutinas personalizadas. Tener una forma sencilla de hacer rutinas siempre va a quitar un gran problema de encima si se dispone de los conocimientos necesarios.

Por otro lado, también puede impulsar a la gente que está decidiendo si ir o no ir al gimnasio a finalmente lanzarse. A no ser que se sea una persona muy decidida, el hecho de apuntarse al gimnasio es un paso muy grande que tomar a veces. Es por eso por lo que cualquier impulso puede llevar a que una persona inicie su viaje en el gimnasio, y posiblemente un cambio en su vida a mejor. Cualquier cosa puede ayudar; si por ejemplo un usuario se lo está pensando y su padre es nutricionista, es claro que, aparte de ir al gimnasio, no se tendrá que preocupar por el tema de la alimentación, lo cual siempre quita un peso de encima porque se ve el camino como algo más sencillo. Es por eso, que facilitar una de las cosas más importantes en el gimnasio ayudará a la gente.

Otra parte de la aplicación es un sistema básico de registro e inicio de sesión en el que se podrá crear una cuenta propia. Dentro de esta cuenta, se podrán tener guardadas las rutinas que sean necesarias. Esto viene muy bien por el hecho de poder tener todas tus rutinas guardadas en un usuario. Cada rutina tendrá un nombre elegido, estando ordenadas por orden alfabético, y además del nombre, tendrán guardados los parámetros de los músculos que se ha elegido, y el tiempo disponible.

1.3. Impacto Esperado

Dependiendo del conocimiento de las máquinas y los ejercicios del gimnasio está claro que se verá la utilización de esta aplicación un poco diferente. Cuando ya se tiene un nivel y se ha estado muchas horas en el gimnasio, hasta el punto de que conocer bien los ejercicios, en ese momento ya no se va a necesitar tanta ayuda porque se suelen tener las rutinas aprendidas, ya que lo normal suele ser seguir la misma rutina la mayoría de las veces.

Pasa lo contrario si aún no se dispone de mucha experiencia. En ese momento sí que vendrá mejor una ayuda como puede ser la obtención de rutinas mediante esta aplicación, porque normalmente se tardará un poco más en acostumbrarse a todo el mundo del gimnasio. La aplicación no está pensada para que una persona la use siempre, llegará un momento en el que el usuario que se ha generado una rutina y la lleva utilizando desde hace semanas, o

meses, se vea capaz de quedarse con su rutina modificada de la forma que le haya parecido más cómoda.

Al menos por ahora la aplicación sólo tiene esa función, para cualquier tipo de mejora, será comentada con más detalle en el Capítulo 9.2. Posibles Ampliaciones.

2. Estado del arte

Las aplicaciones de gimnasio siempre han sido de dos diferentes tipos:

- Aplicaciones centradas en la alimentación
- Aplicaciones centradas en las rutinas

La mayoría de las aplicaciones repiten la misma fórmula porque funciona, una aplicación con una interfaz agradable, un apartado social en la que poder compartir cosas con tus amigos y un apartado donde realizar tu propio seguimiento.
Este seguimiento puede ser de dos tipos; seguimiento físico propio y seguimiento en las rutinas.

Por un lado, el seguimiento físico se basa en poder añadir datos propios como el peso, la altura, y otras medidas como podrían ser el pecho, la cintura y similares. Datos para tener en cuenta si persigues un objetivo como es conseguir que te crezca el pecho, o cualquier otro. Al escribir tu peso un día, el sistema lo guarda, y cuando hay varios días ya puede generar una gráfica para poder observar la variación de una forma visual.

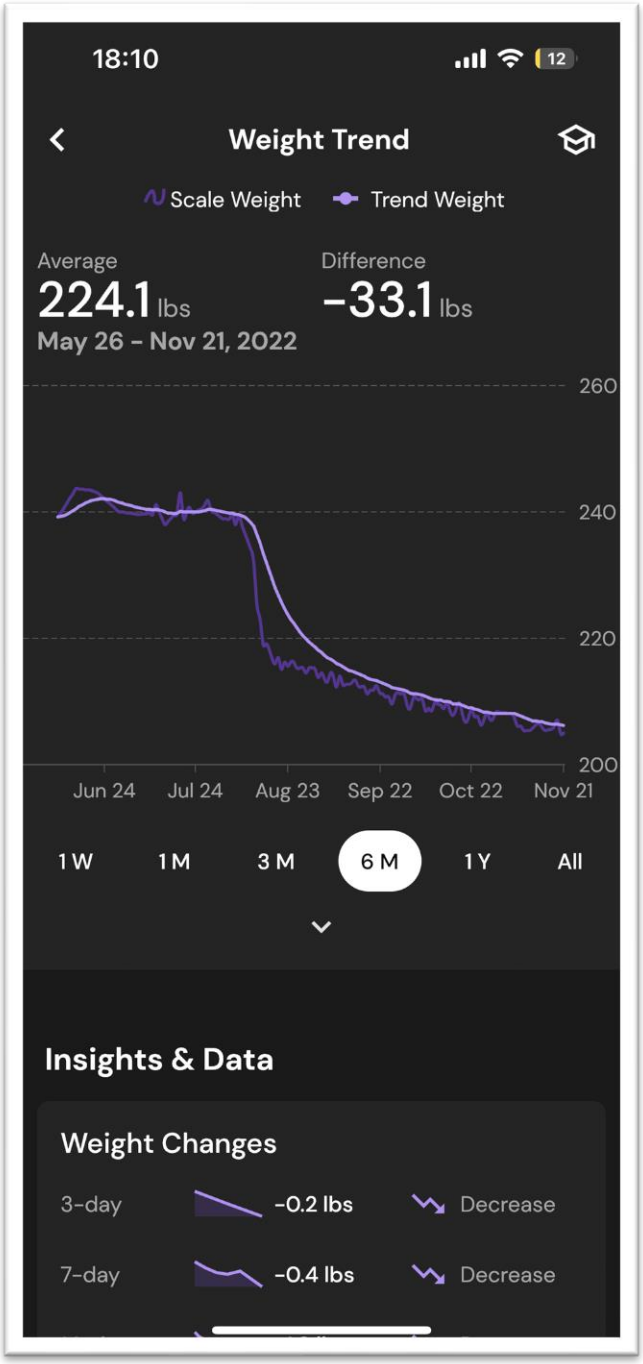

Figura 1: Seguimiento del peso

Como se puede observar en esta imagen, se puede elegir un intervalo de tiempo para modificar cómo se ven los datos. Esto es algo muy práctico debido a que en algún momento es posible cambiar el intervalo para visualizar los cambios que llevas de peso, o la medida seleccionada, de una forma más cómoda.

Por otro lado, el seguimiento de las rutinas necesita una rutina. Cada vez que se esté entrenando se puede añadir cuánto peso se ha usado para una serie de un ejercicio. De este modo, al día siguiente que se repita este mismo ejercicio, se podrá saber con cuánto peso exactamente se hizo el ejercicio, cuántas series y todos los datos necesarios.

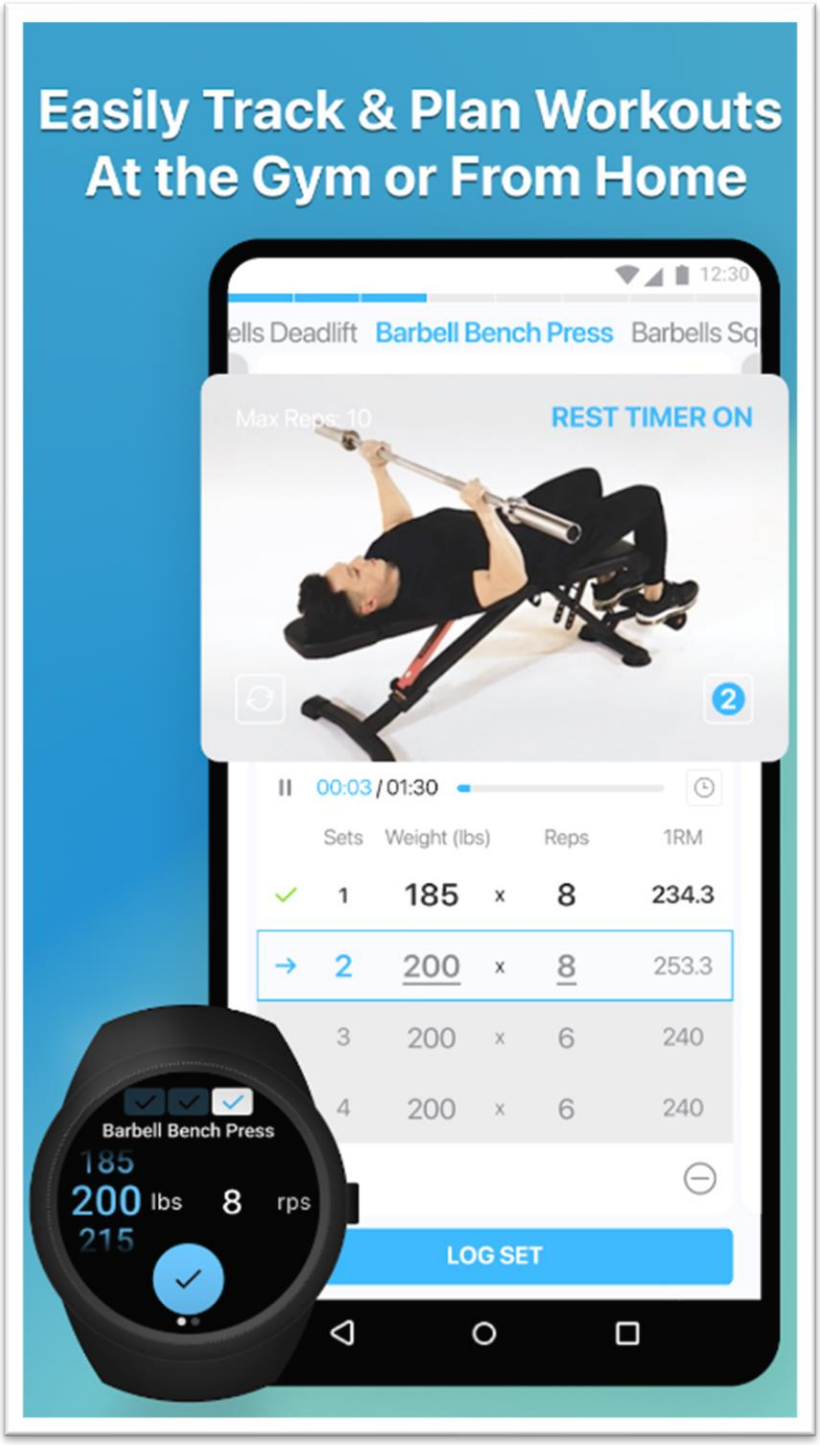

Figura 2: Seguimiento del ejercicio

Suele ser algo muy cómodo tener la comparativa del día anterior, ya que esto lleva a ser capaz de observar si ha habido diferencia en ejercicio respecto al peso del día anterior, o simplemente llevar un seguimiento.

Habiendo resumido un poco las funcionalidades básicas de la aplicación de gimnasio promedio, es hora de comentar sobre un par de las aplicaciones en el mercado:

-JEFIT[1]: JEFIT es una aplicación con una base de datos de ejercicios propia, lo que significa que no requiere conectarse a otro sitio para obtener los ejercicios. Se pueden personalizar las rutinas, y pueden ser iniciadas, es decir, empezar a contar lo que se tarda en los ejercicios y en los descansos, cronometrándolo, para así poder conseguir un gran seguimiento. Por otro lado, JEFIT tiene un apartado social, una comunidad en la que se puede seguir a gente, añadir amigos y realizar publicaciones.

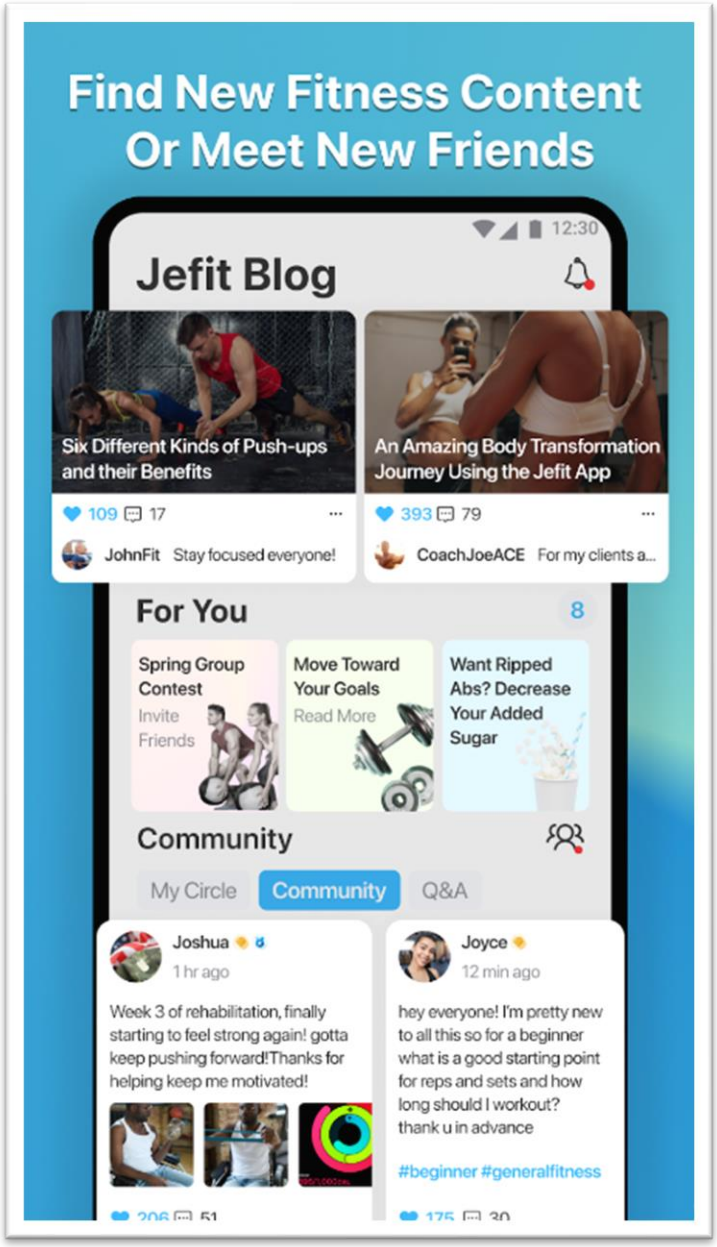

Figura 3: Apartado social de JEFIT

-Adidas Training: workout HIIT[2]: una aplicación oficial de la marca adidas, la cual no requiere ningún tipo de suscripción. A partir del material del cual se disponga, se puede entrenar desde

[1] https://www.jefit.com
[2] https://www.runtastic.com

casa, ya se puede añadir el material del que se dispone en caso de tener. Si se está en un gimnasio, también se puede añadir el material que haya para así que la aplicación busque los ejercicios. La aplicación dispone de planes de entrenamiento, retos mensuales e incluso una pantalla donde poder ver los entrenamientos y las estadísticas de estos.

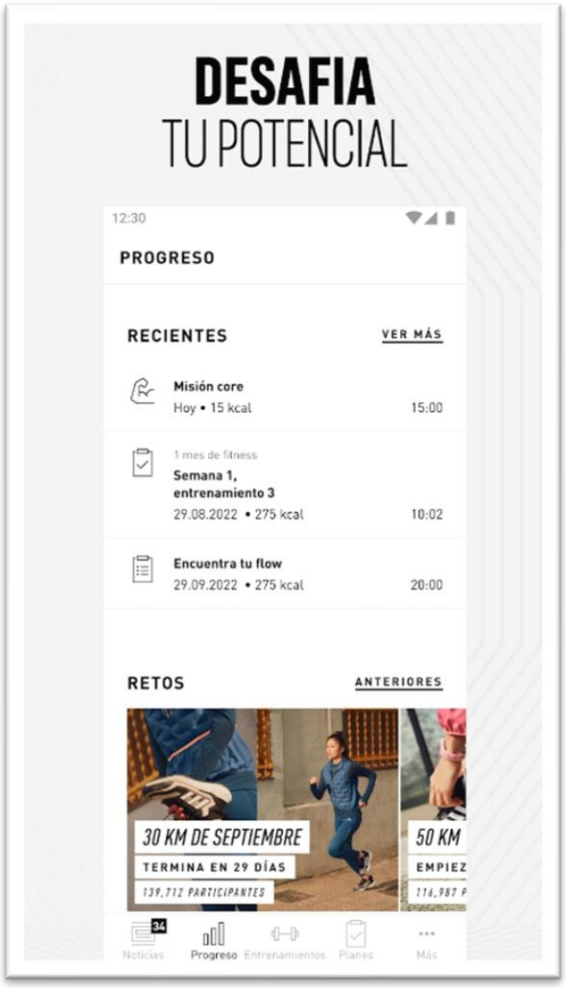

Figura 4: Planes de entrenamiento

A parte de las aplicaciones de gimnasio más centradas en las rutinas, hay también aplicaciones que se centran en la alimentación, que es otro tema muy importante del gimnasio. La aplicación más famosa en este aspecto es MyFitnessPal[3].

MyFitnessPal es una aplicación que bien utilizada es una de las formas más sencillas de llevar el seguimiento de las comidas y sus calorías. Introduciendo el peso del usuario se puede añadir un objetivo, si ganar o perder peso. A partir de ahí, la aplicación asigna un número de calorías y macronutrientes (hidratos, proteínas y grasas), y con esto ya se puede empezar a introducir lo que se come cada día. En la propia aplicación se puede registrar el código de barras para encontrar el alimento para comer, o añadirlo si no existe. Al estar ya en la base de datos, es una forma más fácil de medir los macronutrientes y poder llevar mejor el seguimiento. El hecho de poder añadir recetas que utilices añade bastante comodidad para llevar luego la cuenta de lo que se lleva ingerido. Hace poco añadieron también una función para seguir el ayuno intermitente, muy interesante en

[3] https://www.myfitnesspal.com/es

caso de estar interesado en este.
Por último, MyFitnessPal dispone de una comunidad en la que se pueden compartir tus recetas y hablar con los demás usuarios.

2.1. Propuesta

El desarrollo de esta aplicación se centrará en añadir la creación de rutinas óptimas mediante la simple elección de músculos y un tiempo. Comparando con las aplicaciones de gimnasio, no he visto ninguna que permita crearlas de esta forma. Por lo que he visto, no hay forma de crearlas así, pero sí que se pueden crear manualmente añadiendo los ejercicios que se vean pertinentes, teniendo así un tiempo aproximado de rutina.
A esto, también se le suma el aspecto social de poder compartir las rutinas con las amistades o en la propia comunidad de la aplicación. Debido a que el trabajo se centraba en crear una rutina óptima, no he visto necesario añadir nada social, ya que ya hay aplicaciones que hacen eso por ti, pero podría ser valorado como una mejora futura.

En lo que sí que coincide con las demás aplicaciones es en lo simple que es crear una cuenta para tenerlo todo registrado en un usuario, para así no tener ningún problema de que se mezcle con las rutinas de otro usuario.

A continuación, se muestra una tabla comparando las características de la aplicación creada en este trabajo, con otras aplicaciones del mercado.

	JEFIT	ADIDAS	MYFITNESSPAL	MI APLICACIÓN
REGISTRAR UN USUARIO	SÍ	SÍ	SÍ	SÍ
GUARDAR RUTINAS	SÍ	SÍ	NO	SÍ
CREACIÓN DE RUTINAS	SÍ	SÍ	NO	SÍ
OPTIMIZAR RUTINAS	NO	NO	NO	SÍ
CALENDARIO	SÍ	SÍ	SÍ	*
OPTIMIZAR CALENDARIO	NO	NO	NO	*
PERFIL	SÍ	SÍ	SÍ	*
SEGUIMIENTO DEL PESO	SÍ	SÍ	SÍ	*
SEGUIMIENTO DE LA COMIDA	NO	NO	SÍ	*
PAGOS EN LA APLICACIÓN	SÍ	SÍ	SÍ	NO
COMUNIDAD	SÍ	SÍ	SÍ	NO

Tabla 1: Comparación de Aplicaciones

* Se valora como posible futuro trabajo, más información en el Capítulo 9.2. Posibles Ampliaciones.

En esta tabla, como se puede observar, se comparan las tres aplicaciones mencionadas anteriormente. Por un lado, se puede ver que tanto JEFIT como la aplicación de Adidas están orientadas a ser una aplicación de gimnasio dado que tienen unas características muy similares. Por otro lado, se observa que MyFitnessPal está más orientada a la comida ya que puede hacer un seguimiento de la comida, pero no guarda ni las rutinas ni nada.

Además, las tres aplicaciones tienen varias características que sólo son desbloqueables después de realizar un pago en la aplicación. Para las dos primeras, es un pago que te permite acceder a un seguimiento más detallado, guardar un número determinado de rutinas. Para la otra, se basa en limitaciones a la hora de crear tus propias comidas y opciones de personalización en las cantidades de la comida.

Es decir, pagar da un uso más completo de la aplicación, aparte de quitar todos los anuncios. En mi aplicación esto no ocurre, dado que no hay ningún lugar donde pagar dinero ya que la aplicación es totalmente gratuita.

Por último, cabe destacar que las otras tres aplicaciones sí que tienen un apartado de comunidad, orientadas a poder compartir rutinas, crear posts o compartir recetas de comida.

3. Análisis del problema

A continuación, para continuar con la elaboración del proyecto, se observarán los diferentes tipos de autores y cómo se relacionan mediante casos de uso.

Un caso de uso es un diagrama de comportamiento en UML, utilizado para representar procesos y sistemas de programación orientada a objetos. Este diagrama es capaz de relacionar los actores (los usuarios) con las funciones que pueden realizar dentro del sistema. Para hacerlo más claro:

-Actor: se representa como una figura humana.
-Sistema: todo lo que está dentro del rectángulo.
-Caso de uso: una elipse que contiene el texto de la acción que realiza el usuario.

3.1. Actores

Para los actores utilizaremos dos tipos de usuarios diferentes, el usuario registrado y el usuario no registrado.
-Usuario registrado: usuario que ya dispone de una cuenta creada.
-Usuario no registrado: usuario que no tiene una cuenta todavía.

3.2. Casos de uso

A continuación, los casos de uso junto a la tabla descriptiva.

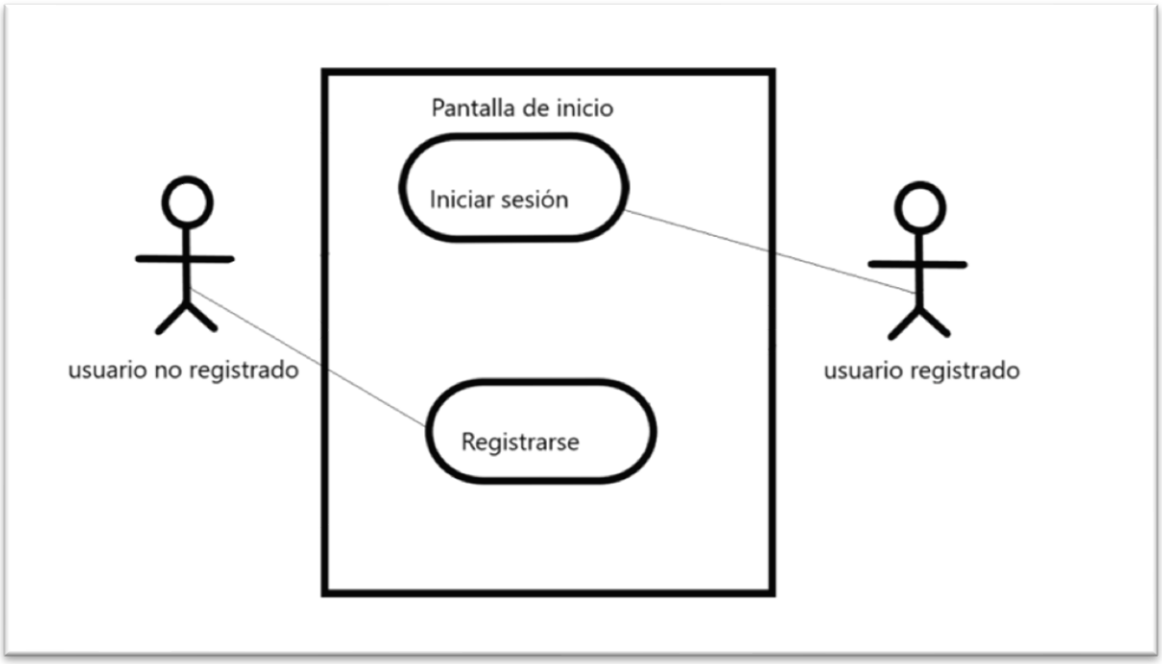

Figura 5: Caso de uso: Pantalla de inicio

Caso de uso	Iniciar sesión
Actores	Usuario registrado
Resumen	Dirige al usuario a una pantalla donde introducir el usuario y la contraseña.
Precondiciones	Tener ya una cuenta creada.
Postcondiciones	Que el usuario coincida con la contraseña y ambos sean válidos.

Caso de uso	Registrarse
Actores	Usuario no registrado
Resumen	Dirige al usuario a una pantalla donde se tiene que crear un usuario nuevo.
Precondiciones	-
Postcondiciones	Que el usuario no esté en uso. Que la contraseña cumpla los requisitos. Que la fecha de nacimiento sea válida. Que el correo electrónico no esté en uso.

A partir de aquí, el único actor que encontramos será el usuario registrado, dado que es necesario tener una sesión iniciada, ya sea tras iniciar sesión, o tras registrarte.

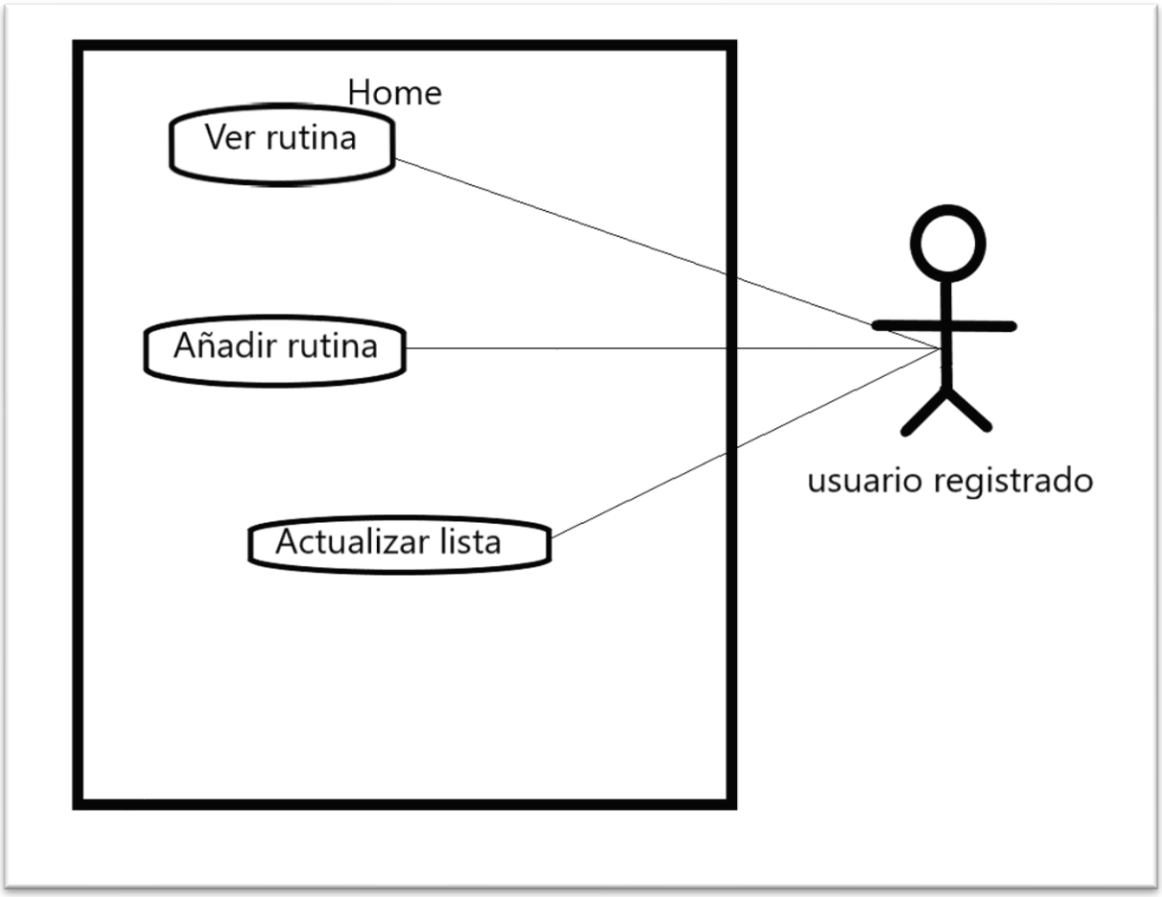

Figura 6: Caso de uso: Home

Caso de uso	Ver rutina
Actores	Usuario registrado
Resumen	Haciendo clic en el nombre de una rutina, se puede acceder a ver toda su información.
Precondiciones	Tener una sesión iniciada. Que esa sesión tenga alguna rutina creada.
Postcondiciones	-

Caso de uso	Añadir rutina
Actores	Usuario registrado
Resumen	Se abre una ventana para elegir un nombre de la nueva rutina.
Precondiciones	Tener una sesión iniciada
Postcondiciones	Que el nombre elegido no esté utilizado en otra rutina de la lista del propio usuario.

Caso de uso	Actualizar lista
Actores	Usuario registrado
Resumen	Refrescas la lista por el caso de haber añadido una nueva, así se puede ver.
Precondiciones	Tener una sesión iniciada
Postcondiciones	-

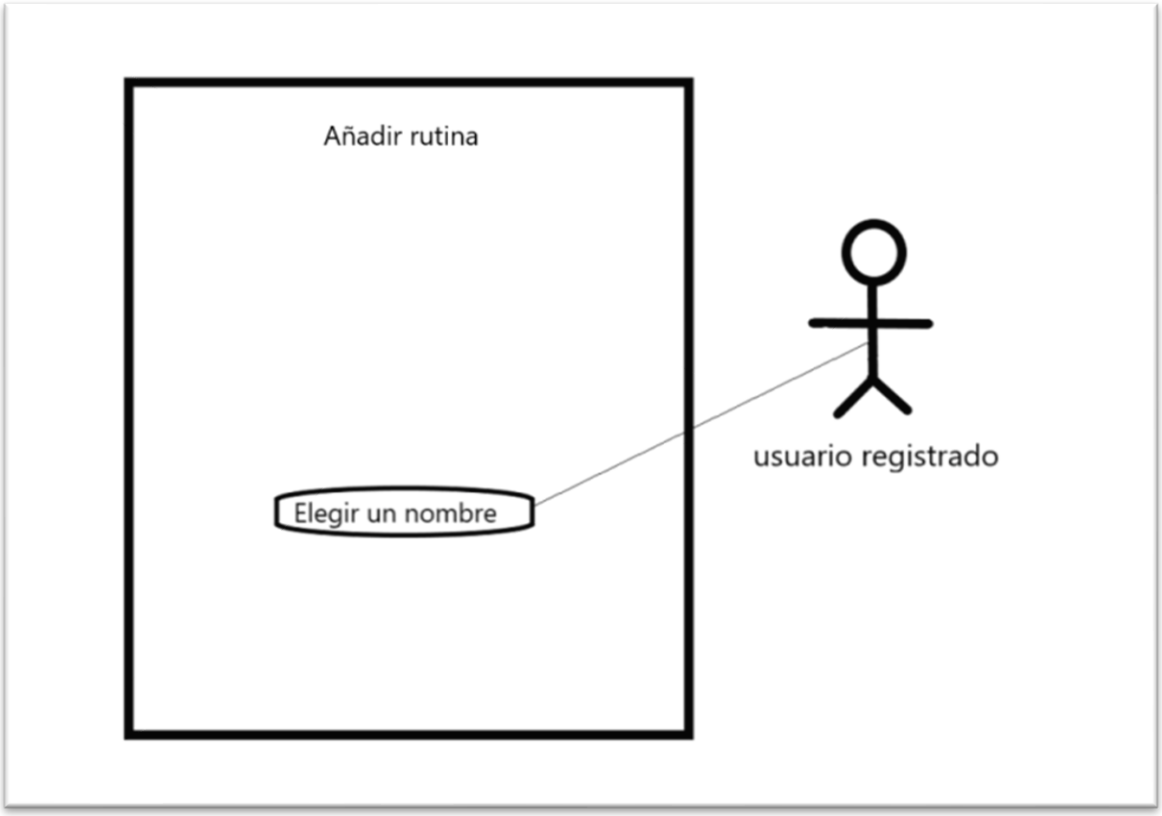

Figura 7: Caso de uso: Añadir rutina

Caso de uso	Elegir un nombre
Actores	Usuario registrado
Resumen	Elegir un nombre para la rutina.
Precondiciones	Tener una sesión iniciada.
Postcondiciones	-

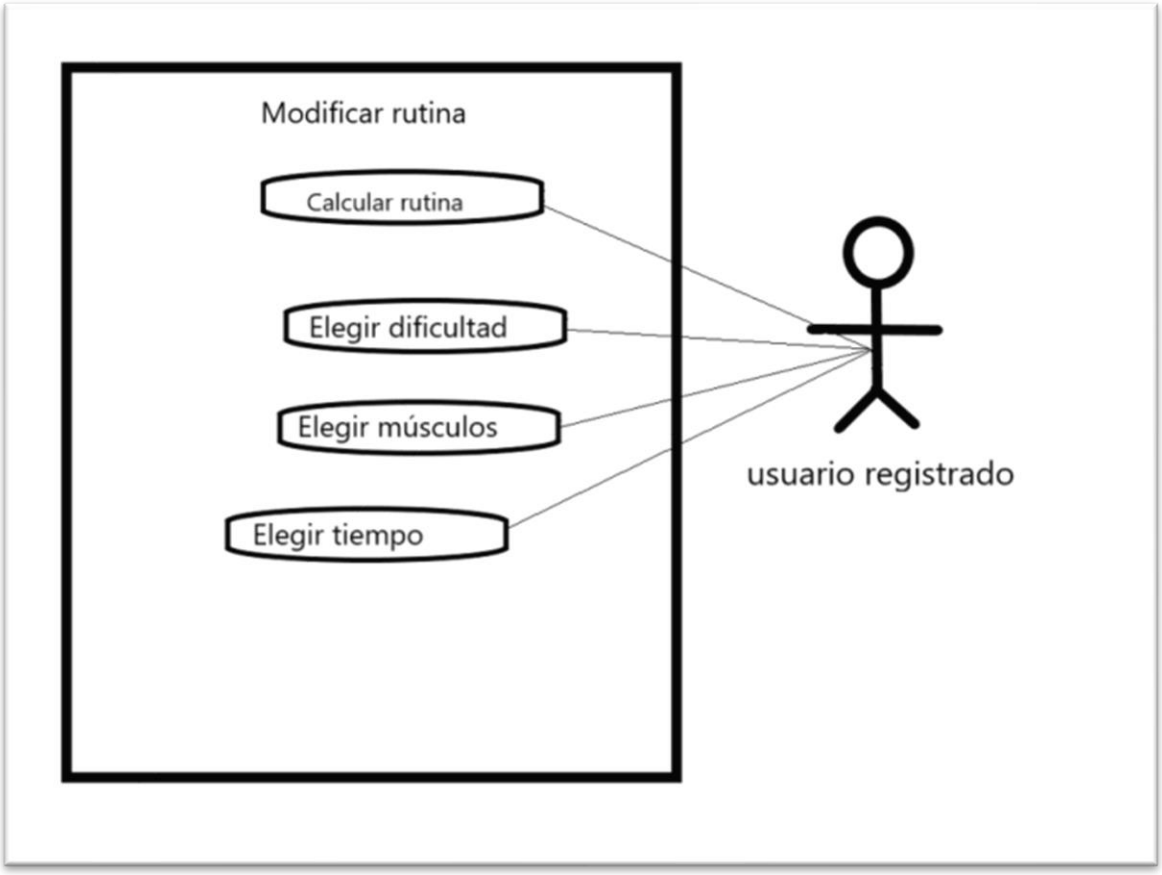

Figura 8: Caso de uso: Modificar rutina

Caso de uso	Calcular rutina
Actores	Usuario registrado
Resumen	Confirma los campos de la dificultad, los ejercicios y los músculos elegidos. Si no hay ningún error, guarda la rutina.
Precondiciones	Tener una sesión iniciada. Haber indicado un tiempo y uno o más músculos.
Postcondiciones	-

Caso de uso	Elegir dificultad
Actores	Usuario registrado
Resumen	Elige qué dificultad de ejercicios se prefiere. Puede quedarse en blanco.
Precondiciones	Tener una sesión iniciada.
Postcondiciones	-

Caso de uso	Elegir músculos
Actores	Usuario registrado
Resumen	Aquí es donde se eligen los músculos en los que se centra la rutina, pudiendo elegir desde un grupo a todos los que hay.
Precondiciones	Tener una sesión iniciada.
Postcondiciones	-

Caso de uso	Elegir tiempo

Actores	Usuario registrado
Resumen	Se elige el tiempo que va a durar la rutina (en minutos).
Precondiciones	Tener una sesión iniciada.
Postcondiciones	El valor añadido sea un integer.

3.3. Plantillas

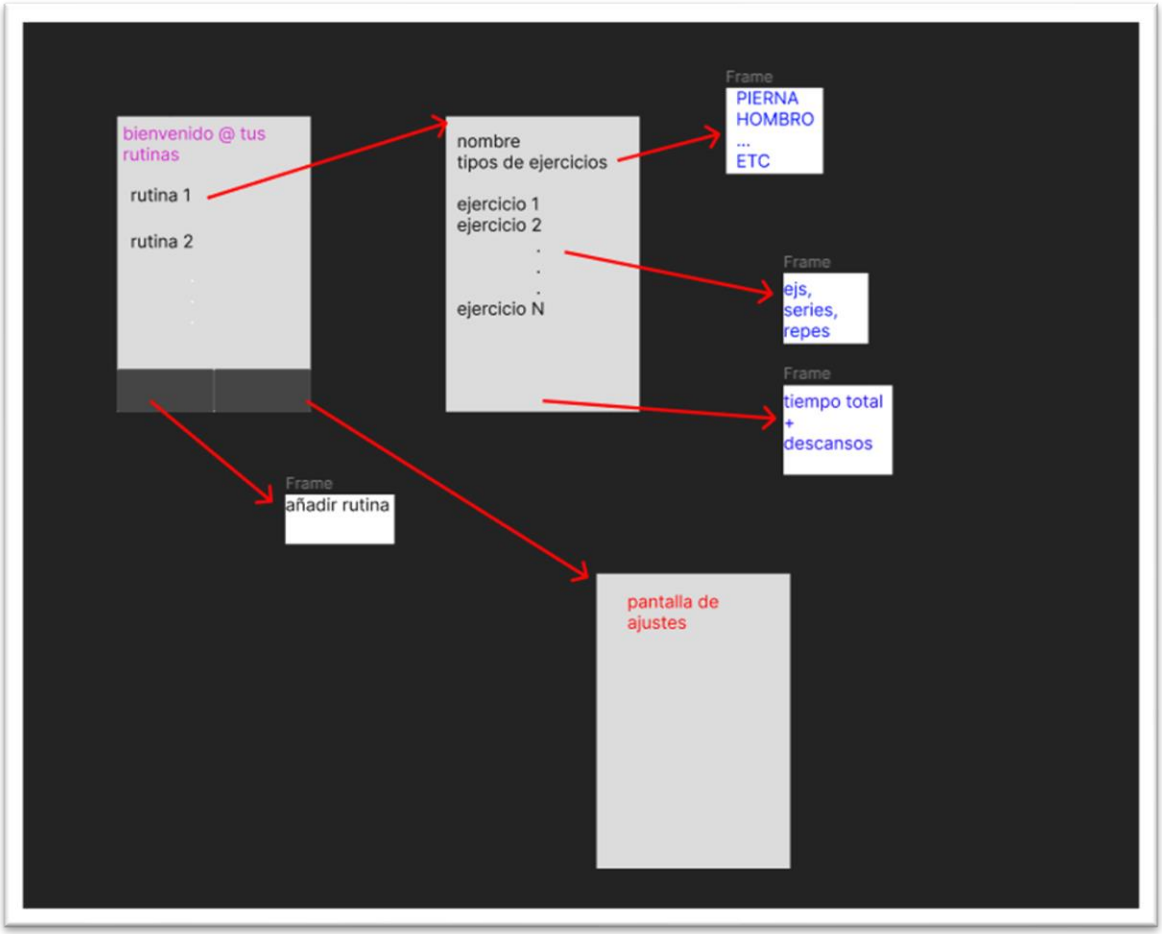

Figura 9: Primera plantilla del proyecto

Esta imagen que tenemos aquí fue una de mis primeras ideas sobre cómo empezar a orientar el trabajo.

Por un lado, la aplicación iniciaría en el home, siendo una pantalla muy simple con una lista de tus rutinas. Debajo de la lista, podemos observar dos botones, uno que llevaría a editar el perfil, y otro que llevaría a ajustes. Lo último de la pantalla home sería que al hacer click en una rutina, se podrían observar los detalles.

Primero, el botón añadir se cambió de lugar para darle una estética diferente, en lugar de estar abajo en un lateral, se acabó quedando en el centro. Este cambio de diseño radica en la importancia que tiene ese botón, pues dándole una posición central recae todo en él, siendo una de las partes más importantes de la aplicación, dado que ésta está enfocada en las rutinas.

Después, el botón de ajustes de momento no se ve necesario del todo, dado que la importancia actual de la aplicación es el tema de las rutinas, y programar un apartado de la

aplicación donde modificar los ajustes del perfil ahora mismo no sería necesario. Esto se comentará más adelante en el apartado de futuras ampliaciones.

Por último, la función de pulsar una rutina y la acción lleve a ella aún sigue teniendo la misma función que tenía en la plantilla. La mejor decisión desde el punto de vista de diseño fue dejarlo así, dado que hacer click suele ser más cómodo para ver la rutina.

Por otro lado, la visualización de la rutina ha cambiado un poco; muestra una pantalla con el nombre de la rutina. En ésta, se tienen que elegir los músculos que se quieren entrenar, pudiendo elegir un tiempo de entrenamiento. Como extra, se puede seleccionar una dificultad preferida, más detalles sobre esto en el capítulo 7.2. El resolutor, donde se entra más en detalle sobre todos los temas y cómo afecta la elección de la dificultad a la rutina.

Realmente, la lista de ejercicios no se muestra en la misma ventana, sino que se guarda en la base de datos y luego se muestra cuando la calculas, así al mínimo cambio realizado siempre va a influir en la lista de ejercicios actual. Esa es la razón por la que se muestra fuera de la ventana de creación.

De hecho, es la forma más cómoda de tenerla. Otra forma que pensé fue enseñar una alerta encima de la ventana nada más darle a confirmar como se muestra en la siguiente imagen.

Figura 10: Plantilla de forma de observar las rutinas

Esta idea fue rápidamente descartada, dado que cada vez que se muestra necesita volver a calcularlo todo por si se han realizado modificaciones. Por lo que en vez de hacer que se muestre así, la propia aplicación lo indica debajo de los ejercicios.

4. Satisfacción de restricciones y optimización

Este punto se utiliza principalmente para comentar todos los detalles sobre la parte más centrada en la optimización de los modelos del problema, dando así una visión más global sobre todas las partes del problema.

4.1. Introducción

La programación lineal es un campo de las matemáticas orientado a obtener un valor de una función lineal, donde las variables suelen estar restringidas por una serie de restricciones, donde todo se puede expresar como un sistema de ecuaciones o inecuaciones lineales.

La forma más simple de verlo es pensando en obtener el resultado óptimo en forma de un conjunto de soluciones. Estas soluciones, son valores que dan forma a un vector y cumplen unas restricciones.

4.2. Problemas de optimización

Un problema de optimización suele estar representado por un modelo. Un modelo debe contener tres elementos:

-Variables de decisión

-Restricciones

-Función objetivo

4.2.1. Variables de decisión

Las variables de decisión son las variables que nosotros, como personas operando el modelo, podemos modificar. Es decir, son las variables que están bajo nuestro control, tienen un valor que se le va a asignar, pese a que al principio son las incógnitas que se determinan en el momento en el cual se acaba resolviendo el problema. En una notación más firme, se suelen denominar Xi, tal que i es el número de variable de decisión a la que corresponde.

4.2.2. Restricciones

Las restricciones en un problema son las limitaciones a tener en cuenta. Son los límites que se encontrarán dentro del problema respecto a las variables de decisión.

Para obtener la solución de un problema, todas las restricciones han de ser satisfechas, dado que, de otra forma, la solución no cumpliría las restricciones del problema.

4.2.3. Función objetivo

La función objetivo es la función que se busca optimizar. Siempre es una función que depende de los valores de las variables de decisión.

4.3. Modelo matemático

Para empezar, asumiremos que $\hat{x}$ = (x_1, x_2,…,x_n) representa a un set de n variables, por lo que podemos expresar el modelo matemático como:

La función objetivo, maximizar o minimizar f($\hat{x}$).
Sujeto a las restricciones: [1]

$$g_i(\hat{x}) \leq g_{b_i:i} \in \{1, \dots, m\}$$

$$h_i(\hat{x}) \leq h_{b_i:i} \in \{1, \dots, m\}$$

Tal que g(x) y h(x) son funciones con el set de n variables. g(x) puede ser igual a h(x), lo único que diferencia de la restricción a la que pertenezca es el símbolo que tiene la igualdad, pero no tendría mucho sentido que perteneciera a ambas, ya que, si tiene que ser menor e igual y luego tiene que ser igual, siempre va a ser igual ya que ha de cumplir las dos.

4.3.1. Más en detalle

Después de haber explicado un poco por encima cuál es la notación del modelo matemático, es necesario entrar algo más en detalle.

Por un lado, la función objetivo hemos visto que se puede maximizar o minimizar. En este caso no es importante cual se elija, dado que el objetivo acaba siendo obtener el valor óptimo según la configuración elegida anteriormente.

A partir de ahora, la función objetivo se expresará como Z. Por lo que, suponiendo que maximizamos: [2]

$$MAX\ Z\ =\ C_i * x_1 + C_2 * x_2 + \cdots + C_{n^*} x_n$$

El objetivo principal del problema será encontrar los valores de $\hat{x}$ que maximice el valor de Z.

Por último, sin salir de la función objetivo, tenemos los coeficientes de la función objetivo. Estos coeficientes son valores enteros, los cuales pueden ser números positivos o números negativos.

Por otro lado, tenemos las restricciones, donde la parte izquierda es una función con los coeficientes de las restricciones y los valores de las variables. La parte derecha es un número natural.

4.4. Tipos de programación

En este punto se explicará un poco por encima los tipos de programación, así como los resolutores más utilizados para resolverlos.

4.4.1. Programación entera

La programación entera añade la posibilidad de limitar el tipo de algunas variables a que éstas sean enteras. Es útil para modelos donde estamos contando con unidades que no se pueden fraccionar. A parte de esto, también es posible limitar una variable a que sus posibles valores sean 0 o 1, que esto serían las variables binarias. Las variables binarias nos permiten representar situaciones imposibles de representar mediante programación lineal, como sería ver si algo ocurre o no.

4.4.2. Programación mixta

La programación mixta es una combinación de la programación lineal con la programación entera, parte de variables enteras y otra parte no. Se siguen siguiendo las mismas bases, las variables en la función objetivo han de ser linealmente independientes.

4.4.3. Programación cuadrática

La programación cuadrática tiene una diferencia muy importante respecto a los dos tipos de programación vistos anteriormente, la función objetivo es una función cuadrática, lo cual lleva a más complicaciones para resolverlo, como puede ser utilizar multiplicadores de Lagrange, pero siempre será una función lineal (es decir, las variables serán linealmente independientes).

4.5. Problema a resolver

Tras hablar un poco de la teoría sobre la programación lineal, es hora de centrarnos más en el problema que nos abarca. Como hemos visto antes, un modelo se suele dividir en tres partes, por lo que es hora de comentar las tres partes del problema relacionado con la aplicación de gimnasio.

Por un lado, las variables a tener en cuenta son las siguientes:

Cada ejercicio va a tener una variable profit, la cual se utiliza para calcular cuánto merece la pena escoger un ejercicio, es decir, lo que obtendríamos por utilizarla para el sumatorio. Es una variable real. En el capítulo 7.2. El resolutor, se comenta con más detalle cómo se obtiene la variable profit.

También cada ejercicio E tiene un número de series que se van a realizar, siendo éste E_x, es una variable entera.

Además, cabe destacar que cada ejercicio tiene un valor de tiempo asignado dependiendo del ejercicio, por lo que no todos los ejercicios aportarán de la misma forma a la variable del tiempo.

Por otro lado, las restricciones. Para calcularlo hay dos restricciones para tener en cuenta. La primera se basa en respetar el tiempo total. Cada ejercicio va a tener asignado un tiempo por serie, y ese tiempo es necesario que la suma de todos los tiempos sea menor que el tiempo dado. La primera se representaría así:

$$R_1: \sum_{1}^{n} EJ_x \,.\, tiempo \;\leq\; tiempo$$

Después, esta otra restricción se puede ver también como un grupo de restricciones, porqué no es sólo una, pero todas tienen la misma función. Se representarían así:

$$R_{x+1}: E_x \leq 3$$

Siendo x el número de ejercicios y E_x el número de series que se realizan del ejercicio x, hay que tener en cuenta que el máximo número de series de cada ejercicio que se pueden hacer. De esa forma, podemos limitar el número de series de forma que una rutina no devuelva sólo un ejercicio, haciendo todas las repeticiones posibles hasta que se acabe el tiempo.

Con este dato, aprovechamos también para modificar la primera restricción, dado que el tiempo de todos los ejercicios probablemente sea mayor que el tiempo asignado, por lo que hay que tener en cuenta en la limitación que sólo se compruebe con el número de series de cada ejercicio que vamos a realizar, lo cual quedaría así:

$$R_1: \sum_{1}^{n} EJ_x \,.\, tiempo \;*\; E_x \;\leq\; tiempo$$

Ahora mismo sí que estamos solamente comprobando que el tiempo de las series de los ejercicios que vamos a hacer se tenga en cuenta para la restricción del tiempo, y sí que es un valor correcto por el hecho de que, si un ejercicio **s** no va a ser añadido porque vamos a hacer cero series, $E_s = 0$, así que no añadiría al tiempo total.

Para terminar con esta restricción, es necesario añadir el tiempo de descanso entre series. Vamos a establecer el tiempo de descanso en 60 segundos, dado que, según estudios, es el tiempo óptimo para descansar [3], por lo que acabaría quedando así la restricción:

$$R_1: \sum_{1}^{n} EJ_x.tiempo * E_x + E_x * 60 \leq tiempo + 60$$

$$R_1: \sum_{1}^{n} (EJ_x.tiempo + 60) * E_x \leq tiempo + 60$$

Esta modificación lo que hace es tener en cuenta los sesenta segundos de descanso por cada serie que se haga, en la forma reducida de la restricción lo que hace es contar el tiempo que tarda el ejercicio y sumarle el tiempo de descanso también. A la derecha se le añaden 60 segundos, que serían los que se le añadirían para la última serie. Dado que ya el usuario se iría a los vestuarios o a su casa, esos 60 segundos no son contados como tiempo de gimnasio ya que no se puede controlar lo que un usuario puede tardar en ducharse y/o cambiarse.

Por último, la función objetivo es así:

$$Z = MAX \sum_{1}^{n} EJ_x.profit$$

Es decir, suponiendo que tenemos n ejercicios, sumaremos la variable profit de cada ejercicio.

4.6. Resolutores más utilizados

A continuación, se comentan una pequeña descripción de los resolutores más utilizados.

Para empezar, de los resolutores utilizados para resolver problemas de programación lineal, y programación mixta es GLPK (GNU Linear Programming Kit). Es una librería que se puede llamar, escrita en C [4].

Otro resolutor a destacar es lp_solve, que, de forma gratuita, resuelve modelos mixtos utilizando el método simplex y el método de ramificación y poda [5].

Después, también tenemos choco solver, una librería open source gratuita escrita en Java [6] para la programación con restricciones.

Los resolutores anteriores resuelven programación lineal mixta, programación lineal pura, sin importar el tamaño de los modelos, pero no son capaces de resolver modelos que no sean lineales.

Por último, tenemos CPLEX. CPLEX es un resolutor que resuelve problemas de programación lineal, entera y cuadrática [7]. Se dispone de un conjunto de librerías a utilizar como API para llamar, y estas librerías se pueden llamar desde C++, Java y .Net [8]

A continuación, tenemos una tabla comparativa resumiendo y comparando los atributos comentados anteriormente.

	GLPK	lp_solve	Choco solver	CPLEX
Programación entera	SÍ	SÍ	SÍ	SÍ

Programación mixta	SÍ	SÍ	SÍ	SÍ
Programación cuadrática	NO	NO	NO	SÍ
Usable en javascript	SÍ [4]	SÍ [5]	NO	NO

Tabla 2: Comparación de resolutores

Después de observar la tabla anterior, llegamos a la conclusión de que los resolutores que serían más adecuados para resolver el problema que se está planteando serían GLPK y lp_solve, pero no han sido utilizadas dado que no eran compatibles con el motor de Javascript utilizado por React Native. El hecho de que sean más adecuados radica en su usabilidad en Javascript, lenguaje sobre el cual está basada esta aplicación.

[4] https://github.com/jvail/glpk.js/
[5] https://www.npmjs.com/package/lp_solve

5. Metodología

Para este proyecto se ha utilizado una metodología ágil, ya que una de las principales características de la metodología ágil es dividir las tareas en pequeños incrementos, y normalmente la metodología ágil está recomendada para proyectos cortos [9].

Se han seguido las siguientes fases:

-Entender los requerimientos necesarios del proyecto: es necesario pensar en qué iba a ser necesario para empezar, visualizar la manera más cómoda de hacer una aplicación decidiendo hacerla en móvil, a partir de ahí decidir si Android, iOS o ambos. Después elegir el lenguaje de programación adecuado para todo lo necesario, la aplicación, el backend que pudiera ser utilizado, etc…

-Análisis: también es importante realizar un análisis a las aplicaciones similares que ya se encuentran en el mercado. Sin hacer mucha alusión a ellas, en el apartado del estado del arte están separadas por los dos tipos de aplicaciones que hay junto a una pequeña descripción, para observar un poco cómo es la competencia.

-Diseño: a partir de haber realizado un análisis previo a las aplicaciones, lo siguiente importante es el diseño. El diseño de la aplicación es algo necesario, porque lo mejor siempre es empezar a programar con una idea en la cabeza. Anteriormente, en el Capítulo 3.3. Plantillas, había un ejemplo de una plantilla de diseño de las primeras que hice sobre la interfaz, con una descripción de más o menos lo que tenía en la cabeza en ese momento.

-Hacer el código: Para hacer el código lo más importante fue tener claro en qué lenguajes se iba a programar. A partir de ahí, ya es simplemente hacer código.

-Testeo: Otro detalle muy importante es comprobar que el código funciona. Se ha hecho bastante testeo para detectar posibles bugs y todos los posibles casos en determinadas situaciones que podrían llevar a problemas.

Una de las ventajas más importantes de una metodología ágil es su rápida adaptación al hecho de que pueda cambiar sobre lo que se está trabajando. Ya que era posible que en algún momento mientras realizaba el trabajo cambiase de opinión en algún detalle, esto conllevaba un cambio en la dirección del código. Tras volver a hacer un análisis para evaluar los cambios y su posibilidad de ser implementados, se continuaba como se estaba funcionando anteriormente, pero modificado hacia el nuevo enfoque.

También fue muy importante la satisfacción del cliente. Esta aplicación era principalmente como un trabajo final de grado, pero si hay gente interesada en la aplicación, sus opiniones pueden ser útiles como usuarios valorando una aplicación, y esta metodología requiere feedback de los clientes [10].

Es cierto que cuenta con algunas desventajas, como por ejemplo el hecho de que, al confiar en el feedback de los usuarios, es necesario que haya usuarios pendientes, y eso como equipo (en este caso, sólo yo) puede llegar a crear situaciones en las que no pueda trabajar debido a que estoy esperando las respuestas de los usuarios.

Más adelante, se puede observar en el capítulo 8. Pruebas y Validación la ayuda que aportaron los usuarios para el desarrollo, junto a opiniones de la aplicación cuando eran necesarias.

5.1. Organización del trabajo

Siguiendo el punto de metodología, aquí es donde se entra más en los detalles de la forma de organización utilizada.

5.1.1. Trello

Una de las formas más cómodas para organizar el trabajo ha sido utilizar Trello[6]. Trello es una página web en la que se pueden organizar tableros.
En este caso, ha sido utilizado creando tres diferentes tableros, uno donde organizar las tareas pendientes, otro para las tareas en curso, y otra para las tareas finalizadas.

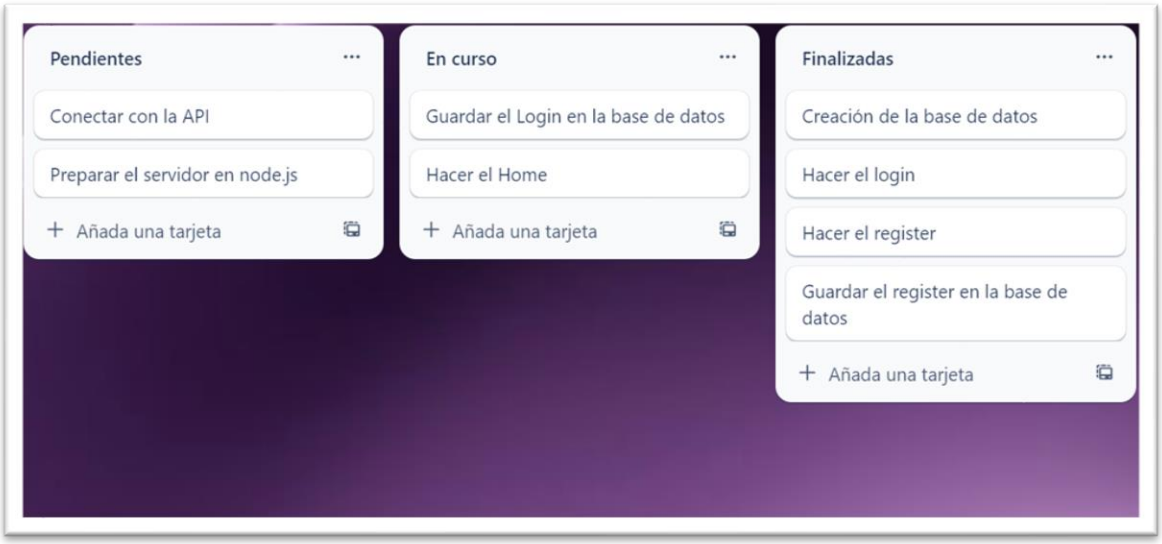

Figura 11: Trello

Tener Trello me ha permitido poder llevar una organización del estado de las tareas. Ha sido algo muy útil, pero tener las tareas así no te permite llevar las tareas tan organizadas si no tienes en cuenta todo el tema temporal, por lo que es necesario llevar una aproximación en la medida de lo posible sobre el inicio y fin de las tareas, junto a un tiempo esperado de duración de ésta. Es ahí donde entra el diagrama de Gantt.

5.1.2. Diagrama de Gantt

Un diagrama de Gantt es una herramienta gráfica utilizada en gestión de proyectos para observar el tiempo de dedicación y previsto de cada parte de las tareas del trabajo de una forma visual.

[6] https://trello.com/es

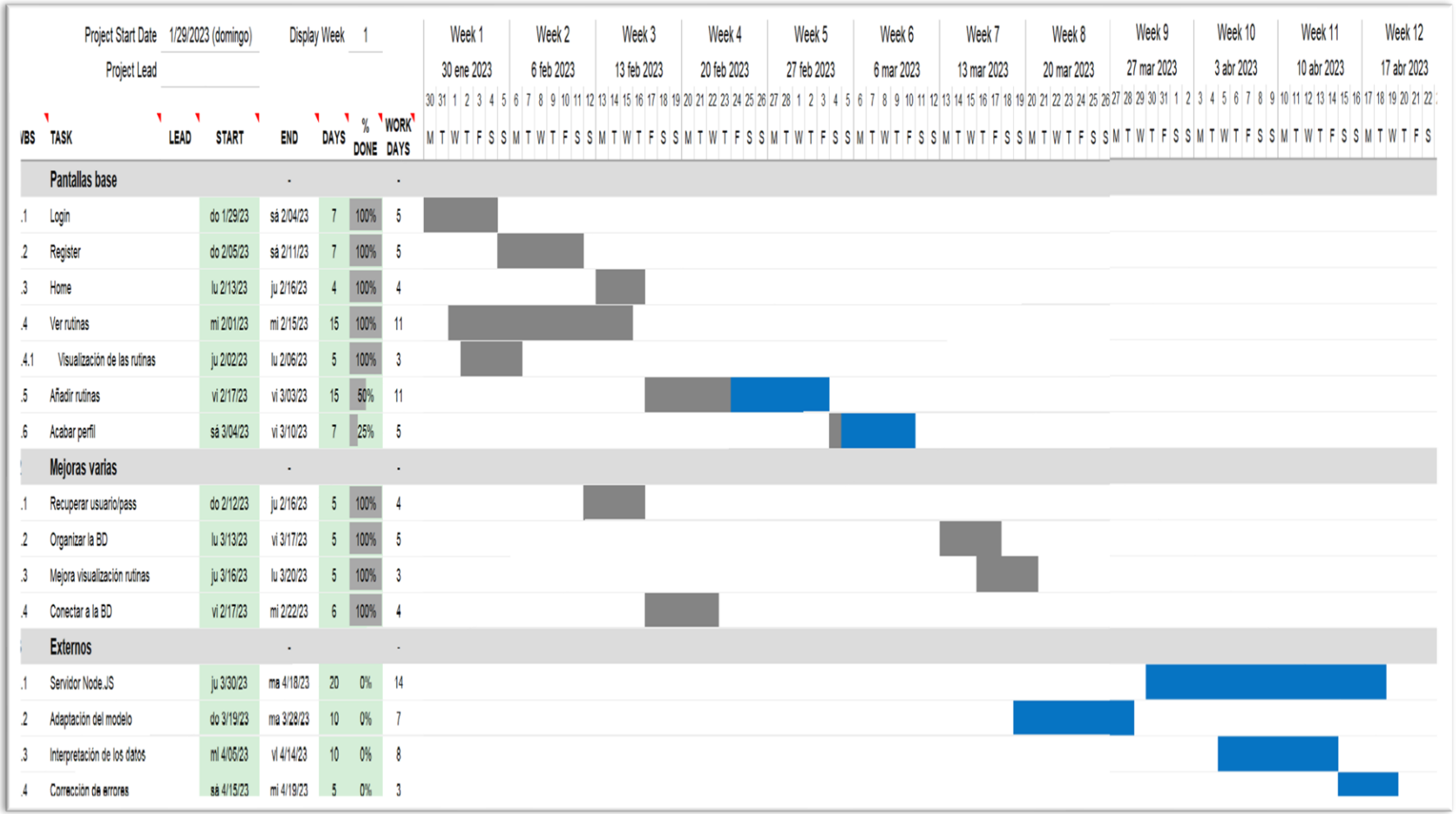

Figura 12: Diagrama de Gantt

Como podemos observar en la imagen anterior, aquí está el diagrama de Gantt, mostrando las doce semanas de duración aproximada del proyecto.

El proyecto de construir la aplicación consta de tres partes principales, siendo estas la creación de las pantallas base, mejoras a la conexión de las pantallas y los temas externos. Las pantallas base fueron lo más importante a implementar, necesitando ser conectadas entre sí para formar lo que es la aplicación. Esto fue muy importante para el trabajo, dado que fue necesario conectarlas entre sí mediante navegación entre ventanas.

Al tener unas ventanas funcionales con una funcionalidad básica, era el momento de proceder a realizar mejoras necesarias para la aplicación; por ejemplo, muy importante fue todo el tema de conectar con la base de datos, guardar todo en ella y poder acceder mediante consultas, debido a que los datos se guardaban ahí. Sobre este tema, hay más detalles en el capítulo 7.3. Base de Datos.

Por último, el apartado de externos se centra más en las cosas necesarias para la aplicación, pero que no entraban como tal en las mejoras básicas para la funcionalidad de la aplicación, ni estaban hechas sobre las ventanas principales. Un ejemplo de esto es la creación del servidor en node.js.

Cada parte principal consta de tareas, y estas tareas pueden tener subtareas. Cada tarea tiene asignada un tiempo de inicio, una duración de la tarea, un final de la tarea y un porcentaje de completado, siendo este último una aproximación.

En el gráfico de la derecha, cada tarea se representa desde su día de inicio hasta su día de fin, teniendo los días en los que se va a trabajar en esa tarea, pintados de gris o de azul. El gris simboliza que la tarea ya se ha hecho, y el azul indica que esa parte de la tarea está pendiente.

Hay tareas que se pueden hacer a la vez, como por ejemplo la tarea de la pantalla de ver rutinas, y su subtarea de visualizarlas. En pocas palabras, la visualización de las rutinas es una lista que mostraba todas las tareas, por lo que se podía hacer a la vez que la ventana de las rutinas, y esa es la razón por la que no se ha de esperar a que acabe la anterior para empezarla. Otras tareas sí que tienen este problema, para hacer una se ha de acabar la otra. Por ejemplo, para interpretar los datos recibidos es necesario haber adaptado el modelo. Esta es la razón por la cual la finalización de la tarea de la adaptación del modelo es anterior al inicio de la interpretación de datos recibidos, ya que sería imposible empezar a trabajar.

6. Análisis de la aplicación

En este punto se comentará sobre el coste energético de la aplicación. Por otra parte, también se realizará un pequeño análisis del marco legal y ético, dado que se manejan datos personales como contraseñas de los usuarios, o tarjetas de crédito / cuentas de paypal en caso de realizar una compra en la aplicación.

6.1. Análisis energético o de la eficiencia algorítmica

Para optimizar la aplicación de forma que sea eficiente es necesario que siga los siguientes pasos:

-Cargar los datos necesarios de la base de datos el mínimo de veces: Por un lado, ya que el perfil requiere tener datos cargados de la base de datos, la mejor opción sería cargarlos una vez nada más cargar la aplicación, ya que así se reduciría el número de veces que se consulta a la base de datos. Un gran número de consultas a la base de datos puede llevar a un peor rendimiento y mayor consumo de SQL, cosa a evitar ya que algo muy importante de la aplicación es el alto rendimiento que puede llegar a tener. Por otro lado, las rutinas también cargarían una vez y no sería necesario más, aunque se llegue a hacer un cambio de una rutina o añadir una nueva, hasta que no se guarden los cambios no se va a realizar ningún tipo de modificación a la base de datos para no guardar información que aún no haya sido confirmada por el usuario.

6.2. Análisis del marco legal y ético

Debido a que en esta aplicación las cuentas de usuario han de tener datos personales de los usuarios, es muy importante que los datos que se utilicen sean guardados y organizados de una forma que cumpla la ley de protección de datos personales. En concreto, la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [11].

Ya que en la base de datos se van a guardar datos personales como contraseñas, es necesario utilizar un método de encriptado para así evitar la posibilidad de una pérdida de contraseñas en caso de un acceso no deseado. Para realizar la encriptación, se utiliza un modelo [12] basado en un algoritmo AES [13].

Con este sistema de encriptación, los datos sensibles de los usuarios estarán a salvo. Además, cuando los usuarios tienen que decidir una contraseña, hay un mínimo de requerimientos para que la contraseña sea válida: ha de tener letras mayúsculas y minúsculas, una mínima longitud de 8 caracteres, mínimo un número y un símbolo.

Figura 13: Comparación del tiempo de hackear una contraseña

De esta forma, tras cumplir los requerimientos el usuario tiene una contraseña segura y protegida, dado que si es sencilla está más expuesta a hackers.

6.3. Presupuesto

Para empezar, comentando el presupuesto es necesario recalcar el hecho de que se trata de minimizar el coste pese a que ello conlleve a un retraso en el trabajo, dado que técnicamente no se trabaja hacia una fecha determinada, excepto si se sigue el diagrama de Gantt, el cual dispone de aproximaciones sobre la longitud de las tareas. Para hacerlo más sencillo, vamos a suponer que el trabajo se va a cumplir a rajatabla con las fechas puestas, lo cual sería empezar el 30 de enero y acabar el 21 de abril.

Entonces, de aquí podemos observar que es necesario un programador. El sueldo medio de un programador son unos 2375€ al mes [14].

Visto eso, nos queda también añadir costes de publicación de la aplicación, y el ordenador que usaría el programador. Para aproximar, podemos suponer un ordenador de unos 800€.

Por último, falta añadir el coste de publicar la aplicación en Play Store, que en este caso serían 25$ [15], lo cual actualmente son 23,29 €.

Sumando todo, obtenemos un coste de 7948,29€ para el diseño, programación y publicación de la aplicación en Play Store.

6.4. Diagrama de clases

Para continuar con el análisis de la aplicación, tenemos delante un diagrama de clases. Un diagrama de clases es un diagrama que describe la organización de un sistema mediante sus clases. Cada caja es una clase.
Dentro de cada clase, lo primero es el nombre, lo segundo es los atributos:

- Que empiece por “+” significa que es un atributo público
- Que empiece por “-“ significa que es un atributo privado

Por último, lo tercero es las acciones que puede realizar esa clase.

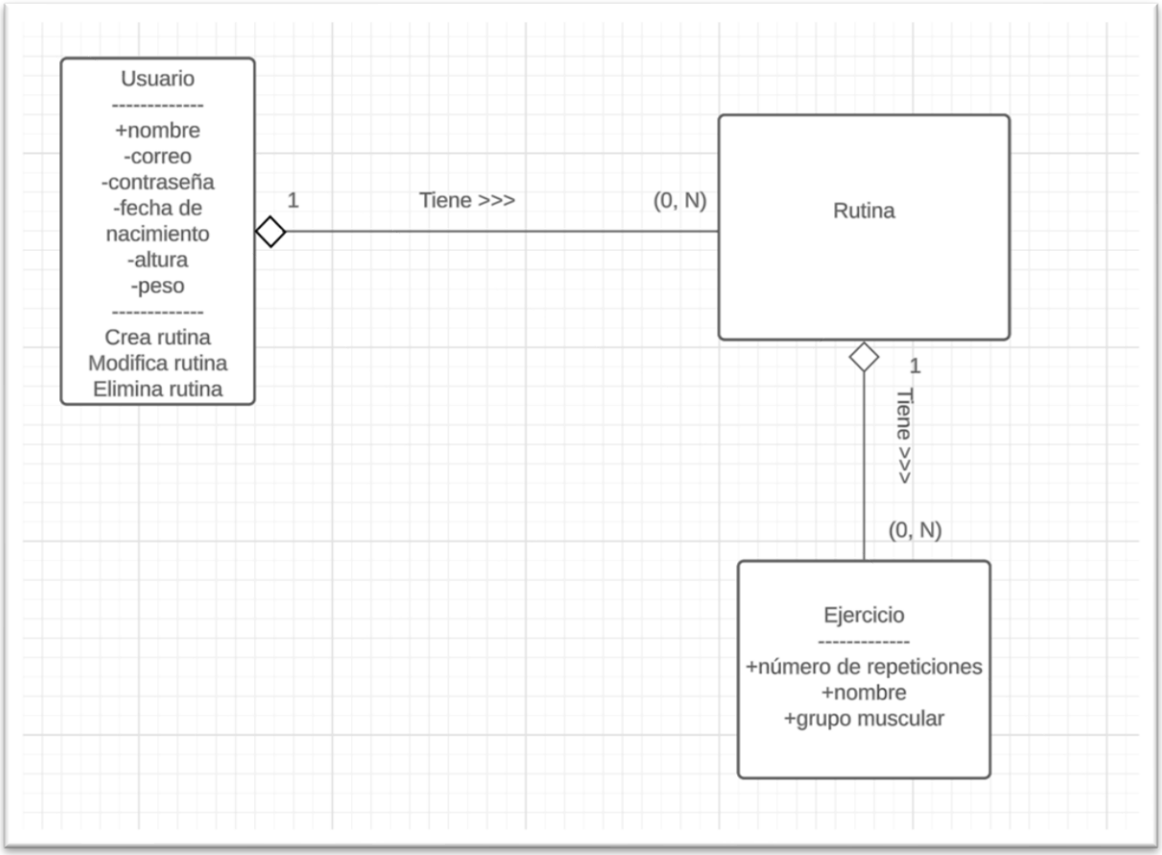

Figura 14: Diagrama de clases

Como se puede observar, según el diagrama de clases representado, un usuario tiene de ninguna rutina a N, siendo N un número indeterminado. Pasa algo parecido con la relación entre una rutina y los ejercicios, ya que una rutina puede estar vacía cuando se crea, por lo que es posible que tenga 0 ejercicios. Por último, el rombo blanco significa que una rutina es un conjunto de ejercicios, o que un usuario es un conjunto de rutinas.

7. Desarrollo de la aplicación

En este punto se entra más en detalle sobre las tecnologías utilizadas y la influencia que han tenido para hacer la aplicación.

7.1. Tecnologías utilizadas

En este punto se describirán las tecnologías utilizadas.

-Android Studio [7]

Figura 15: Logo de Android Studio

Android Studio es un entorno de desarrollo creado por Google para el diseño y creación de aplicaciones móviles para Android. También puede emular un móvil para crear una simulación y probar la aplicación desde ese móvil, comprobando así los cambios que le hagas al código en directo.

En este caso ha sido muy útil por la sencillez que ha sido una vez estaba preparado el proyecto, dado que se podía modificar el código y comprobar su efecto sobre la aplicación.

-TypeScript [8]/JavaScript

Figura 16: Logo de JavaScript y TypeScript

Por un lado, JavaScript es un lenguaje de programación interpretado, orientado a objetos, dinámico y débilmente tipado. Por otro lado, TypeScript es un lenguaje que es un superconjunto de JavaScript, pero con la diferencia que éste tiene tipos estáticos y objetos basados en clases.

Estos dos lenguajes son los mayormente usados en el trabajo.

---

[7] https://developer.android.com/studio
[8] https://www.typescriptlang.org

-React Native [9]

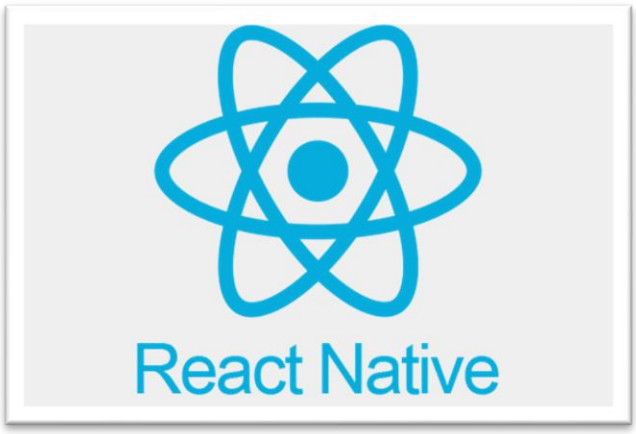

Figura 17: Logo de React Native

React Native es un framework de código abierto utilizado para desarrollar aplicaciones Android, iOS, Windows… ya que permite que los desarroladores usen react con las características nativas de estas plataformas.

React Native ha sido el framework principal utilizado para desarrollar la aplicación, con una única excepción, que veremos a continuación.

-Node.JS [10]

Figura 18: Logo de Node.JS

[9] https://reactnative.dev
[10] https://nodejs.org/en

Node.JS es un entorno en tiempo de ejecución multiplataforma, basado en JavaScript y asíncrono, que permite la creación de aplicaciones web escalables y de alta concurrencia.

En este caso, Node.JS ha sido utilizado como un servidor que recibe la información de la rutina, la calcula y la devuelve. Esto se explica más atrás en el capítulo 7.2. El resolutor.

7.2. El resolutor

Como resolutor, se necesitaba un resolutor de programación lineal. Tras estar barajando las diferentes opciones y no encontrar un resolutor compatible con React Native, por lo que me tocó innovar con una solución diferente. El mejor resolutor que encontré programado en JavaScript y listo para utilizarse no era compatible con React Native por Hermes, el motor que utiliza, por lo cual hubo que buscar otra forma de utilizarlo. Es ahí donde se me ocurrió la idea de seguir utilizando JavaScript, pero en Node.JS.

Dado que no hay una forma sencilla de juntar React Native con JavaScript, acabé creando un servidor con Node.JS. La idea es sencilla, cuando la aplicación en React Native necesite calcular una rutina, ésta hará una petición a la API de ejercicios, los devuelve y se le enviarán al servidor. El servidor lo recibe, las adapta en formato para el resolutor, se calcula todo y se devuelve a la aplicación.

En este caso, el resolutor utilizado es Javascript-lp-solver[11]

A continuación, un esquema de cómo funciona el proceso para adaptar los datos recibidos de un inicio a lo que recibe el cliente.

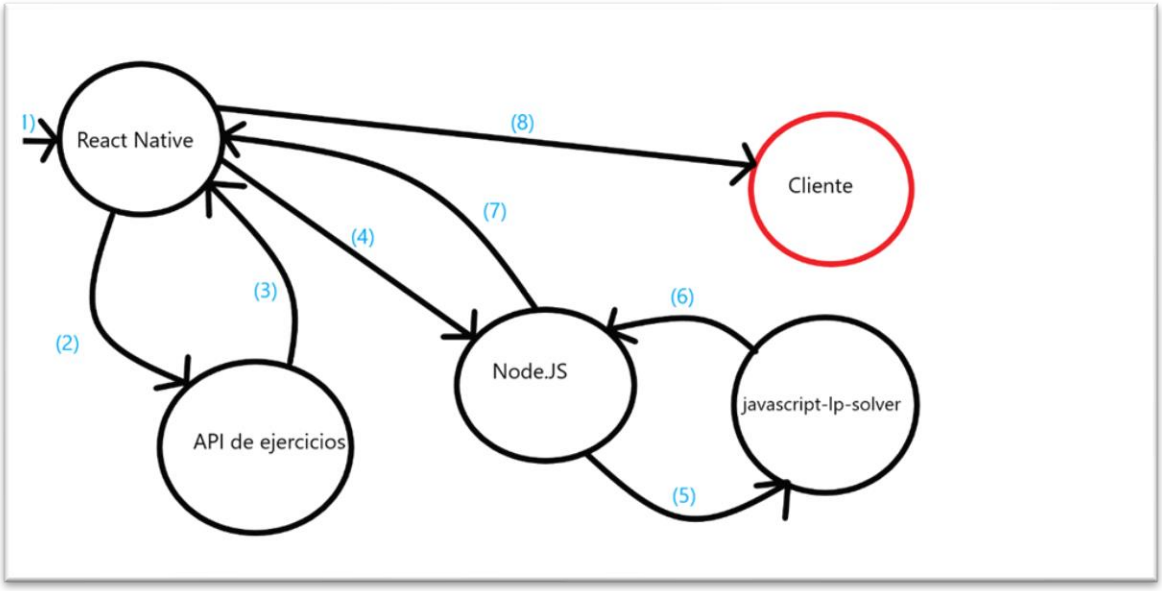

Figura 19: Grafo de proceso del cálculo de rutinas

Descripción de los pasos:

1: React Native recibe los datos. Dado que este proceso se centra en el resolutor, podemos simplificar diciendo que entramos en la ventana del resolutor.

[11] https://www.npmjs.com/package/javascript-lp-solver

2: Al introducir los datos necesarios, se hace una consulta a la API de ejercicios. Para hacer la consulta efectiva, se hace una consulta por cada músculo seleccionado. La API utilizada es una API gratuita llamada API Ninjas[12].

3: La API recibe la petición y envía 10 ejercicios del músculo seleccionado. Dado que normalmente se habrán elegido más de un músculo, los ejercicios seleccionados que se reciban se van guardando en un array para así poder más adelante adaptarlos al modelo de programación lineal adecuado para ser calculado.

```
[
  {
    "name": "Incline Hammer Curls",
    "type": "strength",
    "muscle": "biceps",
    "equipment": "dumbbell",
    "difficulty": "beginner",
    "instructions": "Seat yourself on an incline bench with a dumbbell in each hand. You should pres
  },
  {
    "name": "Wide-grip barbell curl",
    "type": "strength",
    "muscle": "biceps",
    "equipment": "barbell",
    "difficulty": "beginner",
    "instructions": "Stand up with your torso upright while holding a barbell at the wide outer hand
  },
  {
    "name": "EZ-bar spider curl",
    "type": "strength",
    "muscle": "biceps",
    "equipment": "barbell",
    "difficulty": "intermediate",
    "instructions": "Start out by setting the bar on the part of the preacher bench that you would n
  },
  {
    "name": "Hammer Curls",
    "type": "strength",
    "muscle": "biceps",
    "equipment": "dumbbell",
    "difficulty": "intermediate",
    "instructions": "Stand up with your torso upright and a dumbbell on each hand being held at arms
  },
```


Figura 20: Ejercicios obtenidos de la API

Este es un ejemplo de los ejercicios devueltos, la API devuelve diez ejercicios. A partir de ahí falta adaptar cada ejercicio; las instrucciones y el equipamiento necesario no son datos que sean posibles adaptar a un modelo de programación lineal, por lo que no los guardaremos.

Además, se enseña al usuario por pantalla la lista de ejercicios, para que éste pueda elegir en caso de que hay algún ejercicio que no quiera hacer.

[12] https://api-ninjas.com/api/exercises

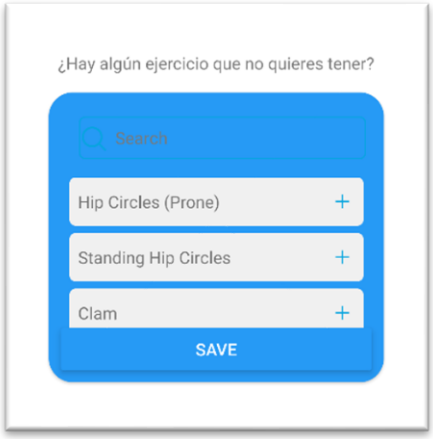

Figura 21: Elección de ejercicios a evitar

4: Al recibir los datos adaptados, se guardan con todo lo necesario y a partir de ahí le añadimos un tiempo a cada ejercicio de cada repetición (necesario en el cálculo temporal), se cambia la dificultad por un valor numérico y se hace lo mismo con el tipo de ejercicio.

LOG [{"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Triceps dip", "tiempoRepeticion": 4, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Decline EZ-bar skullcrusher", "tiempoRepeticion": 5, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 0.95, "muscle": "triceps", "name": "Dumbbell floor press", "tiempoRepeticion": 4, "type": "powerlifting"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Cable V-bar push-down", "tiempoRepeticion": 2, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Weighted bench dip", "tiempoRepeticion": 2, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "EZ-Bar Skullcrusher", "tiempoRepeticion": 5, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Reverse Grip Triceps Pushdown", "tiempoRepeticion": 5, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Push-Ups - Close Triceps Position", "tiempoRepeticion": 3, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Kneeling cable triceps extension", "tiempoRepeticion": 5, "type": "strength"}, {"difficulty": "intermediate", "multiplicadorDificultad": 1, "multiplicadorTipo": 1, "muscle": "triceps", "name": "Single-arm cable triceps extension", "tiempoRepeticion": 3, "type": "strength"}]

Figura 22: Datos de la API adaptados

Este es un ejemplo de cómo se quedan los ejercicios. A partir de esto, se hace una petición al servidor de Node.JS enviándole la lista de ejercicios completa, enviándole también el tiempo de entrenamiento disponible.

5: A partir de aquí, el código del servidor es relativamente simple. Primero adapta el texto recibido, guardando por una parte el listado de ejercicios para ser adaptado, y en otra parte el tiempo. Tras separarlo, se envía todo a la función solveLP del resolutor.

6: Aquí ya estamos en el resolutor, donde mediante una gran modificación de código, adapto el listado de ejercicios. Se entra más en detalle en el capítulo 4.5. Problema a resolver.

```
{
  optimize: 'profit',
  opType: 'max',
  constraints: {
    tiempo: { max: 1500 },
    E0: { max: 3 },
    E1: { max: 3 },
    E2: { max: 3 },
    E3: { max: 3 },
    E4: { max: 3 },
    E5: { max: 3 },
    E6: { max: 3 },
    E7: { max: 3 },
    E8: { max: 3 },
    E9: { max: 3 }
  },
  variables: {
    'Triceps dip': { profit: '1.000', tiempo: 200, E0: '1' },
    'Decline EZ-bar skullcrusher': { profit: '1.000', tiempo: 300, E1: '1' },
    'Dumbbell floor press': { profit: '0.950', tiempo: 400, E2: '1' },
    'Cable V-bar push-down': { profit: '1.000', tiempo: 300, E3: '1' },
    'Weighted bench dip': { profit: '1.000', tiempo: 300, E4: '1' },
    'EZ-Bar Skullcrusher': { profit: '1.000', tiempo: 200, E5: '1' },
    'Reverse Grip Triceps Pushdown': { profit: '1.000', tiempo: 500, E6: '1' },
    'Push-Ups - Close Triceps Position': { profit: '1.000', tiempo: 300, E7: '1' },
    'Kneeling cable triceps extension': { profit: '1.000', tiempo: 400, E8: '1' },
    'Single-arm cable triceps extension': { profit: '1.000', tiempo: 500, E9: '1' }
  },
  ints: {
    'Triceps dip': 1,
    'Decline EZ-bar skullcrusher': 1,
    'Dumbbell floor press': 1,
    'Cable V-bar push-down': 1,
    'Weighted bench dip': 1,
    'EZ-Bar Skullcrusher': 1,
    'Reverse Grip Triceps Pushdown': 1,
    'Push-Ups - Close Triceps Position': 1,
    'Kneeling cable triceps extension': 1,
    'Single-arm cable triceps extension': 1
  }
}
```

Figura 23: Modelo adaptado para ser optimizado

Este es un ejemplo de un modelo adaptado para ser calculado por el resolutor. Indicamos que vamos a optimizar la variable profit, calculada antes. Luego tenemos el tiempo, en este caso son veinticinco minutos, y por último de E0 a E9, siendo el máximo de cada ejercicio.

Más adelante en la lista de variables, cada ejercicio tiene un profit, un tiempo por serie y el valor de entero. El valor ese lo que hace es comprobar que el máximo de ejercicio es de tres, ya que cada ejercicio contaría como uno.

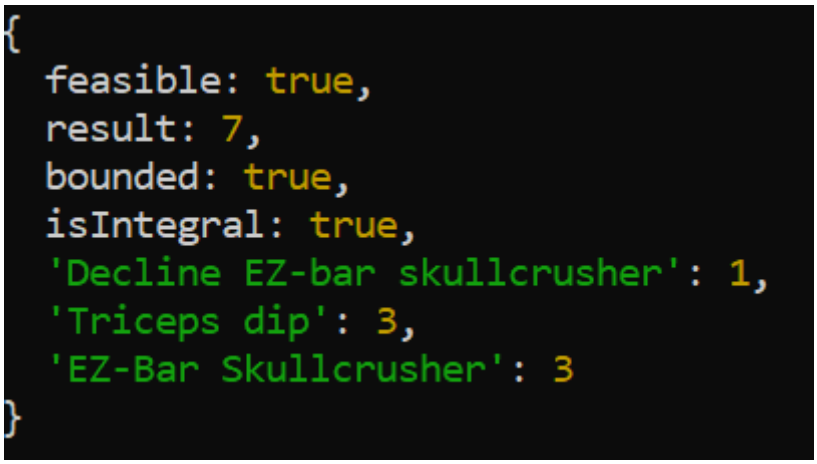

Figura 24: Resultado de optimizar el modelo

Tras ejecutar el resolutor esto es lo que se obtiene. Centrándonos en lo más importante en este momento, el resolutor ha calculado que en esos veinticinco minutos lo óptimo para hacer, dado los datos actuales, sería una serie de 'Decline EZ-bar skullcrusher', tres de 'Triceps dip' y otras tres de 'EZ-Bar Skullcrusher'.

7/8: El servidor devuelve estos datos de una forma ordenada, los cuales son mostrados al cliente por la pantalla.

Después de obtener los datos en node, lo que se hace es enviárselos otra vez a react, para que así se puedan comunicar al usuario y sepa lo que tiene que hacer.

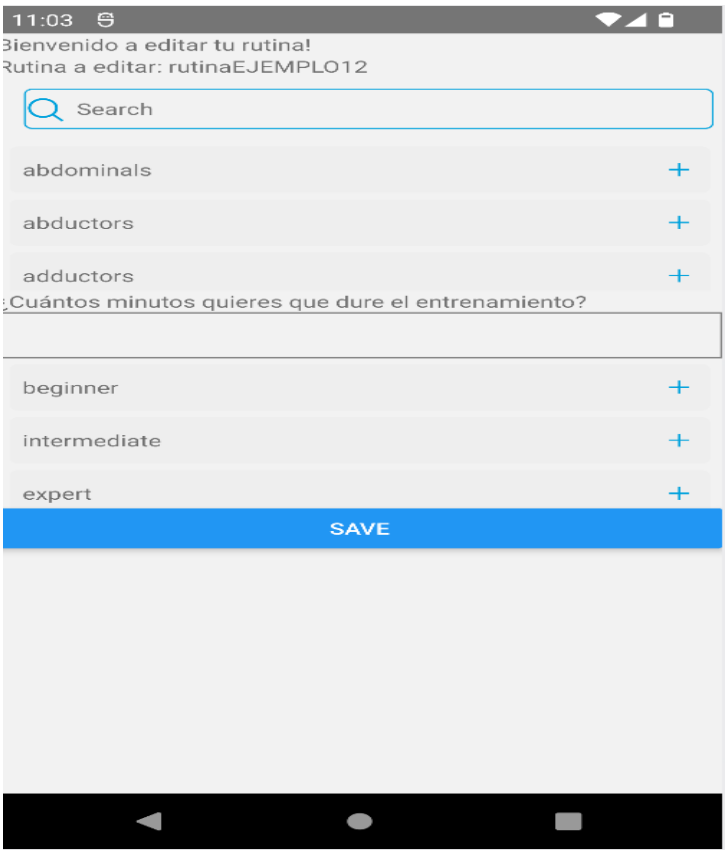

Figura 25: Visualización de la página para calcular la rutina

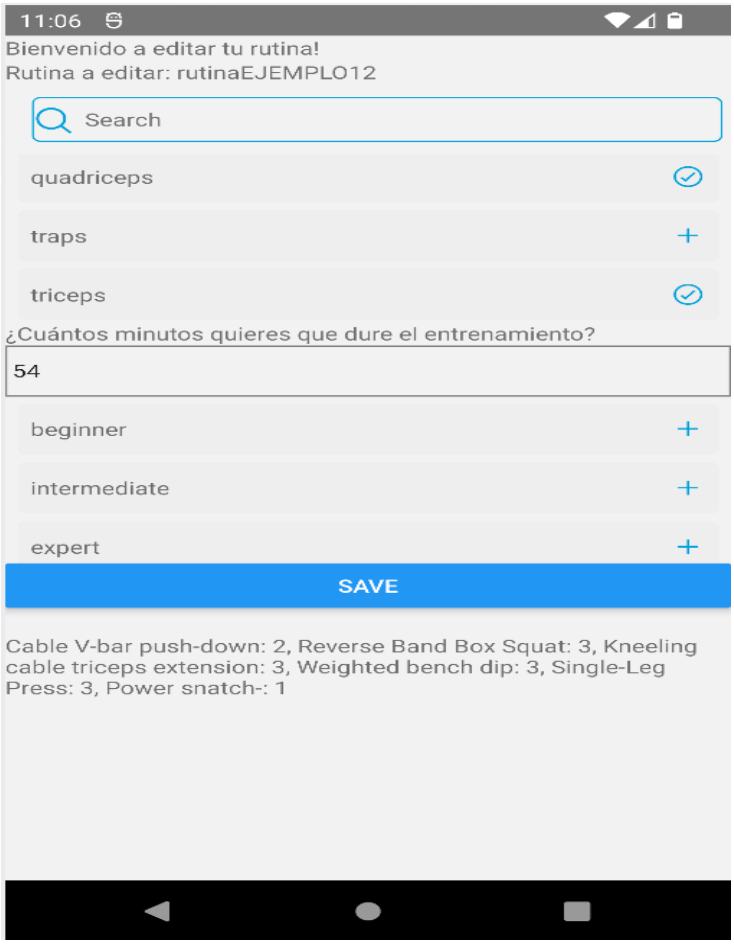

Figura 26: Rutina calculada

Como podemos observar en la figura anterior, debajo de la última barra se pueden ver los ejercicios elegidos, y las series que hay que hacer de cada uno.

7.2.1. Adaptación de la API al Resolutor

Para hablar de todo el tema de cómo funciona un resolutor, hay más información en el capítulo 4. Satisfacción de restricciones y Optimización.

Cuando se recibe la lista de ejercicios desde la API, es necesario modificarla un poco para enviársela al resolutor para que realice los cálculos, ya que no es lo mismo el formato que utiliza la API, que el formato que utiliza el resolutor para calcular.

El formato con el que se recibe la lista de ejercicios es la siguiente:

[{Ejercicio 1, Ejercicio 2, …, Ejercicio 10}]

Donde cada ejercicio sigue el siguiente formato:

{

“nombre”: nombre del ejercicio,

“tipo”: tipo del ejercicio,

“músculo”: músculo del ejercicio,

“equipamiento”: equipamiento necesario,

“dificultad”: la dificultad para hacer el ejercicio,

“instrucciones”: instrucciones para realizarlo,

}

Para empezar, se reciben todos los ejercicios y se van añadiendo a una lista, la cual se acabará enviando al resolutor. Esto es importante dado que, al darle al botón de aceptar, es necesario realizar una llamada a la API para cada ejercicio, ya que sólo se puede realizar una petición con un músculo. Para esto, se guardan los músculos elegidos, se hace un bucle para cada músculo, y al hacer la petición se guardan los ejercicios recibidos, así hasta tenerlos todos.

Cuando ya se tienen todos los ejercicios en una lista, es importante adaptar el formato para enviarlos correctamente. Cada ejercicio tiene datos que no nos son interesantes, como las instrucciones o el equipamiento. Es cierto que para una futura ampliación el hecho de saber qué equipamiento es necesario en un ejercicio podría ayudar a detectar ejercicios que necesiten el equipamiento del cual se dispone, pero no es ahora mismo algo que se contemple. Por otro lado, datos como el tipo de ejercicio o la dificultad son necesarios, pero no con los valores actuales, sino como multiplicadores.

Es decir, cuando se hace una rutina se puede elegir una dificultad. En caso de no elegirla, se queda como valor de la dificultad 1. En caso contrario, al elegir una dificultad la dificultad elegida se convierte en 1.25, y las otras dos se quedan como 0.75. Esto permite priorizar los ejercicios con la dificultad seleccionada.

Respecto al tipo de ejercicio, se le asigna también un multiplicador en función del tipo de ejercicio que sea, priorizando ejercicios más similares a la fuerza.

A parte de esto, se asigna un tiempo a cada ejercicio para tenerlo en cuenta luego al calcular que los ejercicios determinados se puedan realizar en el tiempo indicado por el usuario del entrenamiento.

Al final, se envía todo esto al resolutor junto al tiempo total. Ya se adapta el formato del todo, asegurándose de que todas las llaves estén bien puestas y todo.

La prioridad del resolutor es maximizar la variable profit. La variable profit es calculada mediante una multiplicación de todos los valores anteriores. Está adaptado todo de forma que solamente se escoja un ejercicio hasta tres veces. Por otro lado, es muy importante la restricción que el tiempo de los ejercicios elegidos sea menor que el tiempo total.

7.3. Base de datos

Para la base de datos se ha utilizado SQLite[13].

Figura 27: Logo de SQLite

SQLite es un sistema de gestión de bases de datos relacionales, siendo una librería escrita en el lenguaje de programación C.

Una de las principales razones por las que utilizar SQLite es debido a que es una base de datos muy sencilla de instalar, no requiere de instalar o configurar nada más de software adicional, y el código es dominio público [16]. Otra razón es que las consultas son atómicas, consistentes y aisladas [17].

Por último, hay que añadir que SQLite es el sistema de gestión de bases de datos más utilizados en el mundo. [18]

Por un lado, la organización utilizada en las bases de datos es la siguiente:

Name	Text primary key
Password	Text not null
Email	Text not null unique
Birthdate	Text not null
Height	Real
weight	Real

Tabla 3: Tabla SQL usuario

La tabla anterior es la tabla de usuario. Cuando se crea una nueva cuenta en la aplicación, se ha de comprobar parámetros como que no exista el usuario proporcionado ni el correo en la base de datos. Es por eso por lo que utilizar sqlite, ya que es bastante sencillo realizar las consultas desde el código.

[13] https://sqlite.org/index.html

Id	Primary key
Name	Text
User	Text
ejercicios	Text

Tabla 4: Tabla SQL rutinas

Esta tabla es la tabla de las rutinas. Dado que cada usuario va a tener unas rutinas propias, la forma de que una rutina sea asignada a un usuario es que el "user" de la rutina sea el mismo "name" de la tabla de usuarios. Esa es la razón por la cual el usuario ha de estar registrado para realizar acciones, ya que de otra forma no se podrían guardar las rutinas.
Por último, la entrada de ejercicios es un JSON, del tipo "{"Ejercicio 1": 2, "Ejercicio 2": 3}", que la aplicación leerá, tras haber cargado la lista de ejercicios, al clicar para visualizar una se observará que se tienen que hacer 2 series de "Ejercicio 1" y 3 series de "Ejercicio 2".

Por otro lado, pasando a hablar más del código, la librería de React Native Sqlite Storage ha facilitado muchísimo la implementación, permitiendo crear las bases de datos con dos líneas de código. Para hacer las consultas es similar, pero con algo más de dificultad:

Hay implementados un par de métodos, los necesarios para poder obtener datos de la base de datos, como podría ser la función que obtiene las rutinas mediante un usuario, necesaria para la obtención de los nombres de las rutinas del usuario que tiene la sesión iniciada. También hay otro método que comprueba si existe un usuario en la base de datos, lo cual es muy cómodo debido a que no puede haber dos usuarios utilizando el mismo nombre. Pasa lo mismo con el correo electrónico.

Ya, por último, también hay un método para insertar un usuario en la base de datos, lo cual se hace tras comprobar que todos los campos de registro son correctos. Pasa lo mismo con el botón de insertar una rutina. Ese botón es simple, sólo te pide un nombre, pero el nombre de la rutina no puede estar repetido en las propias rutinas del usuario. Es decir, dos usuarios pueden tener cada uno una rutina con el mismo nombre, pero un usuario no puede tener dos rutinas con el mismo nombre.

8. Pruebas y validación

8.1. Pruebas unitarias

Estas pruebas se van a centrar en la parte del código enfocada en la modificación de la base de datos. En total, hay 20 funciones relacionadas con la base de datos, de las cuales dos son de insertar, por lo que es necesario comprobar que el insertar funciona correctamente.

Por un lado, tenemos insertUser. Una función que necesita el usuario, la contraseña, el email y la fecha de nacimiento (en formato de string)

```
export async function insertUser(name: string, password: string, email: string, birthdate: string) : boolean {
  nombreRepetido = false;
  correoRepetido = false;
  const db = await getDbConnection();
  //const userCount = await countUsers("SELECT count(*) as count FROM users");
  //console.log(`There are ${userCount} users in the database`);
  const userExists = await checkUser(name);
  const emailExists = await checkEmail(email);

  if (userExists) {
    console.log("the user already exists");
    setErrorNombre();
    return false;
  }
  if (emailExists) {
     console.log("the email already exists");
     setErrorCorreo();
     return false;
    }
  const query = `INSERT INTO users (name, password, email, birthdate) VALUES (?,?,?,?)`;
  params = [name, password, email, birthdate]
  const result = await db.executeSql(query, params)
                         .then(result => {
                           //The query was successful
                           //console.log("Rows affected: " + result.rowsAffected);

                           return true;
                         })
                         .catch(error => {

                           // There was an error executing the query
                           console.log("Error message: " + error.message);
                           return false;
                         });;

  return true;
}
```

Figura 28: función insertUser

La base de datos tiene actualmente un usuario. Para esto, vamos a añadir otro usuario mediante el sistema de registro.

Figura 29: Crear un nuevo usuario

Tras confirmar el perfil, podemos observar que la base de datos tras comprobar todos los datos (comprobar que el usuario es válido y no está repetido, lo mismo con el correo, la validez del formato de la contraseña…) añade una entrada nueva. También se ha seleccionado un peso, una altura y un género.

{"birthdate": "12-03-2004", "email": "user123@us.com", "genre": "m", "height": 195, "name": "user123", "password": "aaaaAA!1", "weight": 75}
Por lo que podemos confirmar que insertar una persona funciona.

Por otro lado, tenemos insertRutina, una función que, con el nombre de la rutina, el usuario que la ha creado y los ejercicios, la guarda en la base de datos.

```
export async function insertRutina(name: string, user: string, ejercicios) : boolean {

  const db = await getDbConnection();
  const query = `INSERT INTO rutinas (name, user, ejercicios) VALUES (?,?,?)`;
  params = [name, user, ejercicios]
  const result = await db.executeSql(query, params)
                         .then(result => {
                           return true;
                         })
                         .catch(error => {
                           console.log("Error message: " + error.message);
                           return false;
                         });;

  return true;
}
```

Figura 30: función insertRutina

Para esto hay dos opciones, añadir una rutina sólo con el nombre, o añadir una rutina con el nombre y los ejercicios.

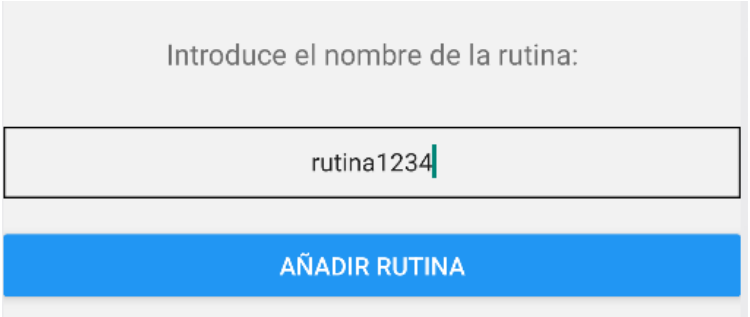

Figura 31: Nombre de la rutina a insertar

Para comprobar que se ha añadido, vamos a la pantalla donde se muestran las rutinas.

Figura 32: Rutina insertada

Actualmente en la base de datos, se muestra así:
{"ejercicios":"","name":"rutina1234","user":"root"}.

Si le queremos añadir ejercicios, se puede hacer desde la pantalla de editar una rutina, como se hace a continuación.

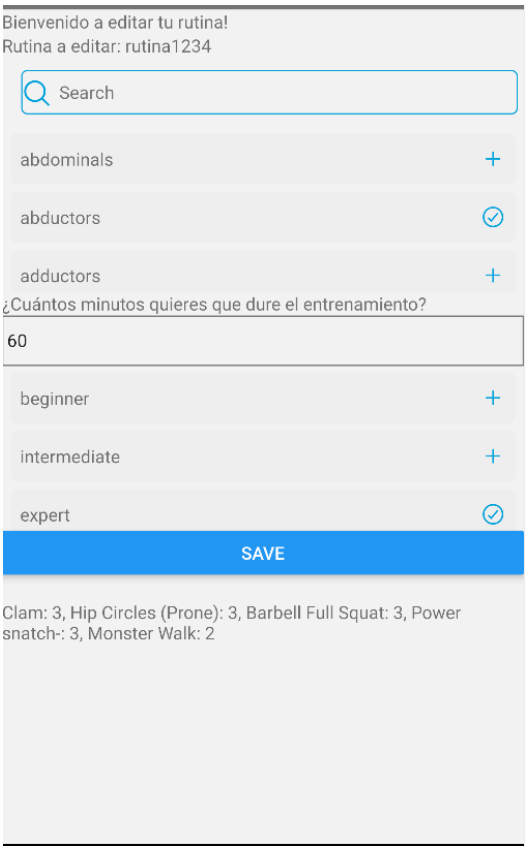

Figura 33: Ejercicios añadidos a la rutina

Ahora mismo, si comprobamos en la base de datos la rutina que hemos creado antes, podemos observar que se vería así:

{"ejercicios":"[{"name": "Clam", "value": 3}, {"name": "Hip Circles (Prone)", "value": 3}, {"name": "Barbell Full Squat", "value": 3}, {"name": "Power snatch-", "value": 3}, {"name": "Monster Walk", "value": 2}]","name":"rutina1234","user":"root"}.

La lista de ejercicios se queda registrada como un JSON, para así luego poder realizar una lectura más fácil y rápida, siendo “name” el nombre del ejercicio y “value” el número de series a realizar.

8.2. Encuestas

Para este punto, se han realizado encuestas a gente que ha podido probar la aplicación con el fin de obtener retroalimentación sobre las opiniones de la gente en lo que han usado la aplicación.

La encuesta ha sido creada mediante Google Forms. Tras recibir la aplicación y tener unos días de uso, la gente que probó la aplicación ha recibido la encuesta para valorar la satisfacción en relación con la aplicación y cualquier problema que haya podido surgir durante su utilización.

A continuación, se presentan las respuestas junto a un pequeño análisis de la aplicación.

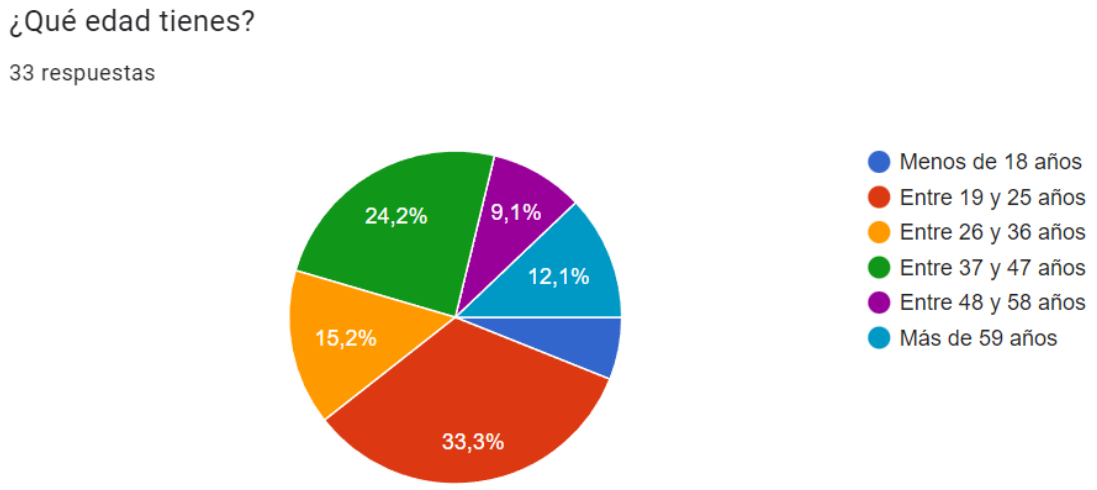

Figura 34: Pregunta sobre la edad de los encuestados

Para empezar, se pregunta la edad, un dato para tener en cuenta por tener una idea de que hay gente de diferentes edades, ya que la aplicación va a ser para todo el mundo y me interesa tener opiniones de todas las variedades posibles. El gráfico demuestra que hay un poco de todas las edades.

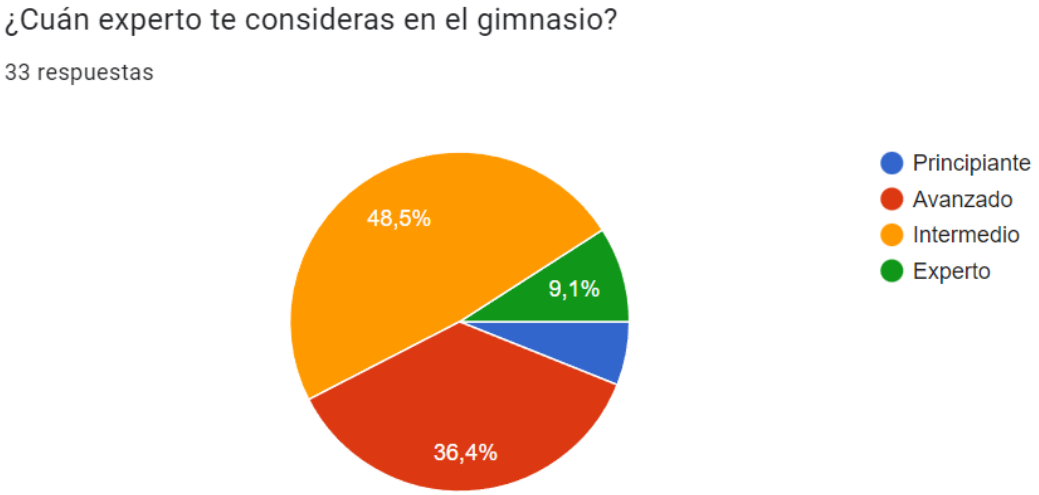

Figura 35: Pregunta sobre cuán expertos son los encuestados en el gimnasio

Otro dato que está bien tener en cuenta es saber un poco cómo se maneja la gente en el gimnasio. Está claro que va a ser una aplicación para todo el mundo, por lo que tu nivel no va a influir en nada, pero está bien tener en cuenta un poco cuál es el nivel que la gente considera que tiene.

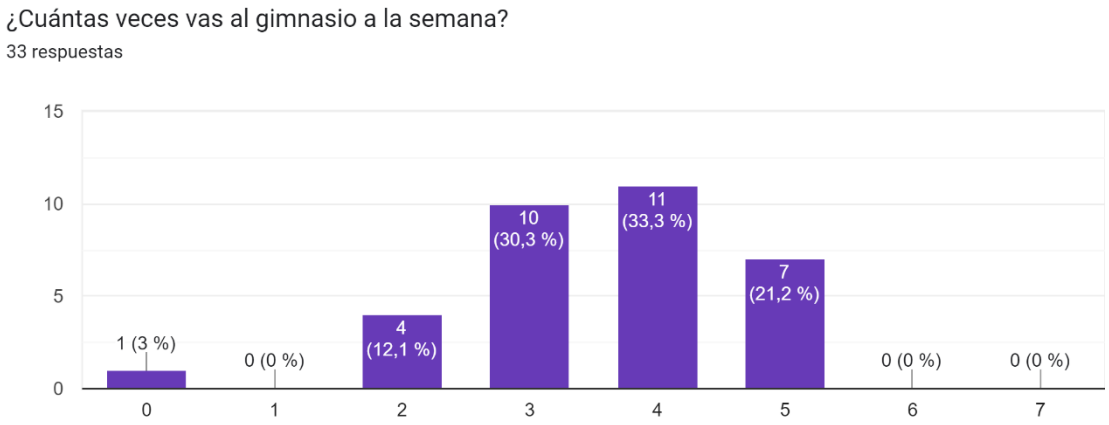

Figura 36: Pregunta sobre cuántos días al gimnasio van los encuestados

También es importante conocer cuánto va una persona normalmente al gimnasio, no para la aplicación en sí, pero para tener una rutina acertada hay que repartirse todos los músculos en un periodo de tiempo. En caso de decidir hacer una futura ampliación que también te intente repartir los días de gimnasio en una semana, este dato nos sería necesario. Más información en el capítulo 9.2. Futuras Ampliaciones.

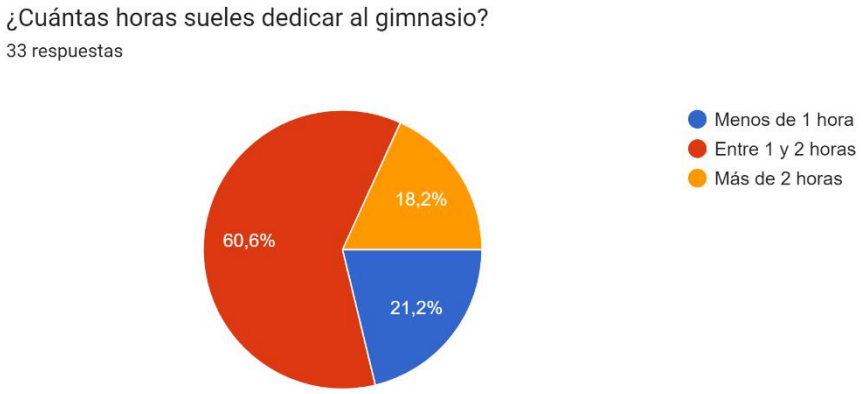

Figura 37: Pregunta sobre el tiempo de dedicación al gimnasio de los encuestados

Como podemos observar, lo normal de las rutinas de la gente que ha sido preguntada es que duren entre una y dos horas. Esto no afecta en nada a la aplicación, dado que, sin importar el tiempo elegido para la rutina, ésta generará las rutinas igual.

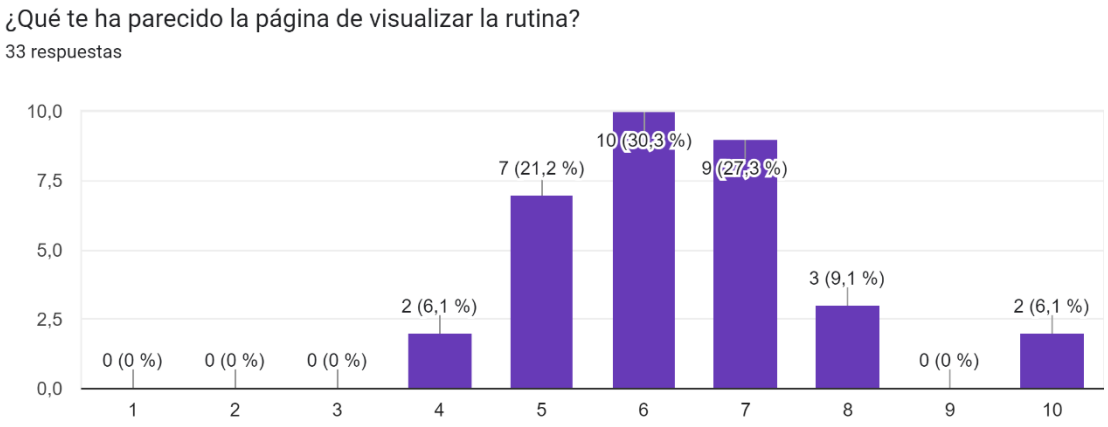

Figura 38: Pregunta sobre la opinión de la visualización de rutina

Ahora pasamos a una visión sobre la interfaz de la rutina. Para ser una interfaz muy básica, ya que el centro de la aplicación era lo de las rutinas no vemos tampoco unos datos muy negativos. Podrían ser mejores, pero eso ya se haría en caso del futuro de la aplicación, ahora mismo no es necesaria una interfaz bonita si simplemente es útil.

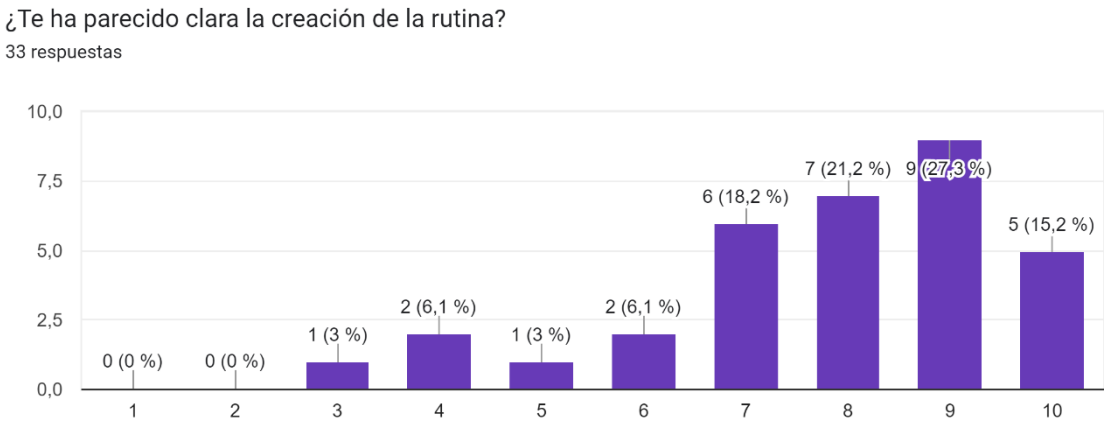

Figura 39: Pregunta sobre la claridad en la creación de una rutina

Aquí sí que hay unas opiniones más variadas en lo que podemos observar comparando con la figura anterior. La creación de rutinas ha de ser algo claro, por lo que estas valoraciones son mejorables, y es una de las cosas con más prioridad de la aplicación.
A parte de las mejoras a realizar después de esta encuesta, en el capítulo 9.2. Futuras ampliaciones, tenemos un ajuste que añadiría un tutorial para la aplicación, lo cual ayudaría a que fuese más sencilla la creación de la rutina.

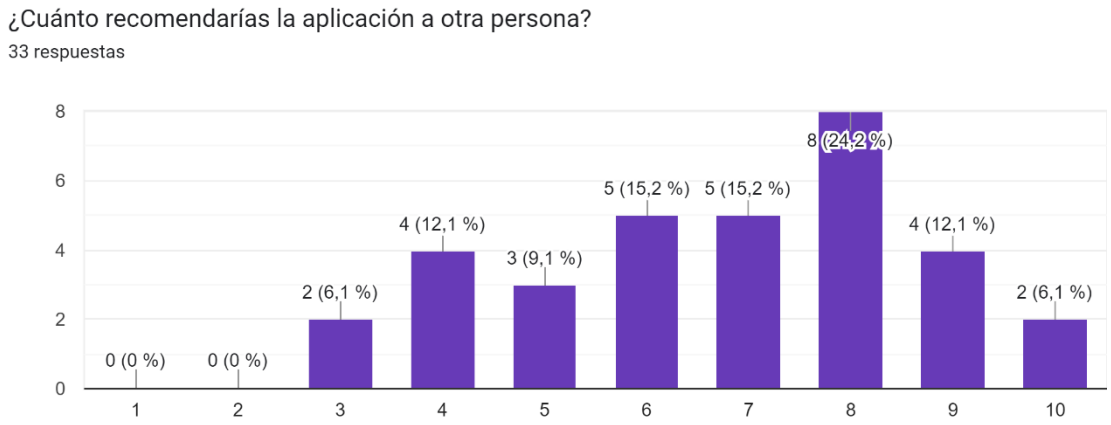

Figura 40: Pregunta sobre si se recomendaría la aplicación

Como se puede observar en esta figura, también vuelve a haber opiniones más variadas sobre la recomendación. Ya que la aplicación no está terminada desde un punto de vista de aplicación y simplemente tiene el apartado de las rutinas, es normal que no se recomiende al haber actualmente muchas otras aplicaciones mejores.

Figura 41: Pregunta sobre si la aplicación puede hacer descubrir a los encuestados mejores rutinas

Esta figura muestra que la mayoría de gente cree que la aplicación puede generar una buena rutina tras probarla. Esto está genial por ese 81.8% de la gente que está de acuerdo o valora la posibilidad.

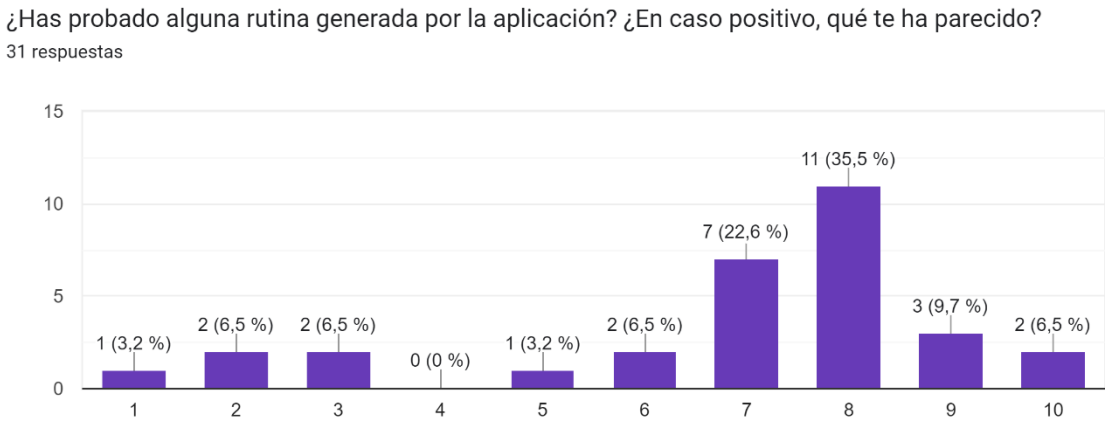

Figura 42: Pregunta sobre si se ha probado alguna rutina

Aquí es donde llegamos a uno de los puntos más importantes, de la gente que haya generado una rutina por la aplicación (31 de 33 personas), poder saber las opiniones de las rutinas. La mayoría de la gente parece estar bastante contenta, lo cual nos hace saber que las aplicaciones que han probado les han sido útiles. Esto de cara a la aplicación está muy bien, dado que podemos saber que las rutinas están siendo funcionales.

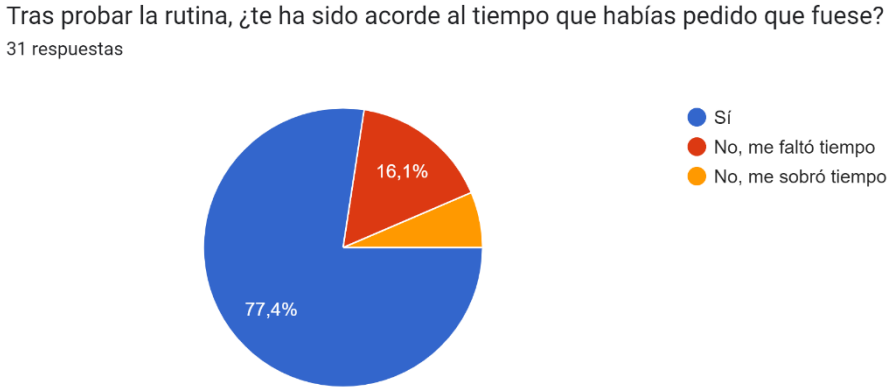

Figura 43: Pregunta sobre si el tiempo de las rutinas ha sido el indicado

Otro tema de la rutina que nos interesa es saber si el tiempo ha sido el mismo que el que tiempo con el que se genera la rutina. Gran parte de la gente sí que ha estado el tiempo que indicaba la rutina, y una pequeña parte no. He dividido en dos el "no" para poder saber la razón por la que no ha sido acorde al tiempo esperado de la rutina, y la gran mayoría de los "no" vienen por falta de tiempo. Por un lado, es posible que no haya dado tiempo porque la persona haya entrenado más lento, los descansos hayan sido más lentos de lo esperado, o cualquier otra cosa. Por otro lado, que haya sobrado tiempo puede ser debido a entrenar más rápido de lo esperado.
También puede haber sido por malos cálculos de la aplicación en ambos casos, es algo a seguir investigando.

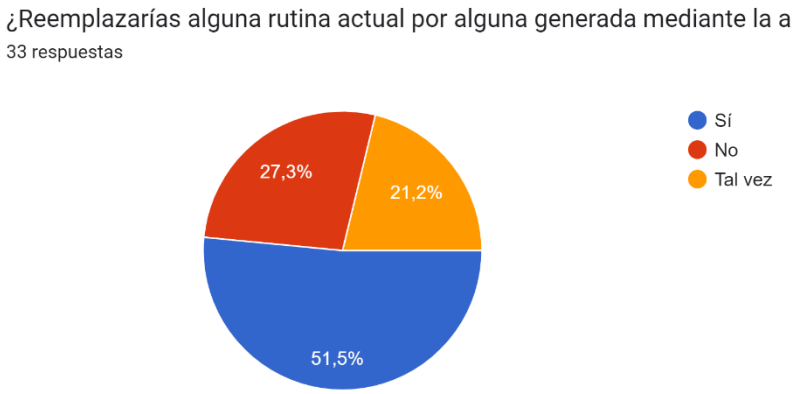

Figura 44: Pregunta sobre si los usuarios reemplazarían alguna rutina propia con alguna de la aplicación

Por último, podemos observar que casi un 75% de la gente se plantearía cambiar una rutina, siendo un 50% del total, gente que está segura de que lo haría. Para el tipo de persona sin mucha idea de las rutinas no significa tanto, pero para la gente que ya tiene sus rutinas hechas de hace tiempo y siempre sigue las mismas, significa más debido a que es más difícil cambiar una rutina si ya estás acostumbrado a ella y te gusta.

9. Conclusión

En conclusión, es momento de echar una mirada hacia el inicio del proyecto.

Primeramente, la idea principal del proyecto era crear una aplicación para el gimnasio. Esta aplicación al final ha acabado cumpliendo con el objetivo inicial, ser una aplicación que pueda crear rutinas mediante optimización lineal.

Uno de los matices más importantes de crear una aplicación, era el hecho de tener que meterme a programar mediante React Native, utilizando tecnologías como Javascript y Typescript, lo cual ha sido un reto ya que no soy ningún tipo de experto. Gracias a haber utilizado estos lenguajes, he podido indagar un poco más en la programación en Android, que es un tema que me parece bastante interesante, y que gracias a este trabajo siento que ya tengo más experiencia.

Además, haber usado de todas las librerías externas también me aporta un conocimiento que podré emplear en mi futuro, como es por ejemplo el uso de SQLite unido a Javascript, o de otras librerías propias de React Native.

Por último, también la necesidad que he tenido de utilizar node.js creando un servidor, el cual ha sido importante para poder mantener el resolutor en un entorno diferente al de React Native, debido a incompatibilidades ya comentadas anteriormente.

Resumiendo, la aplicación ha sido creada siguiendo lo comentado anteriormente en el trabajo, observando un poco tanto su arquitectura, como su estructura, y se ha alcanzado el objetivo principal, tanto como que se han cumplido objetivos temporales impuestos en el Trello.

9.1. Relación del trabajo desarrollado con los estudios cursados

Durante la realización de este trabajo, parte de las ideas que he tenido que han podido influir tanto en el diseño de la aplicación como las ideas para el trabajo, han sido influenciadas por los estudios cursados. A continuación, hay una lista con las asignaturas que han podido influenciar esto, con una explicación:

-ISW: Ingeniería del Software. En esta asignatura aprendí parte de lo visto en el análisis del problema, que son los casos de uso. A parte, vimos también una parte de la organización de los trabajos, donde se comentaron las metodologías ágiles (en este caso, las utilizadas para este trabajo)

-TOP: Técnicas de Optimización. En esta asignatura vi cómo funciona la obtención de un valor óptimo mediante la resolución de un sistema de ecuaciones, es decir, obtener la solución óptima. Mediante métodos como el método simplex [19] podemos lograrlo. En este caso, el resolutor implantado en el código para el cálculo de las rutinas óptimas, como dicho anteriormente, funciona como una optimización de un sistema de ecuaciones. Por otra parte, también se pueden solucionar problemas teniendo en cuenta una resolución de satisfacciones dentro de ello.

-GPR: Gestión de Proyectos. En esta asignatura pude conocer cuáles son las mecánicas más cómodas para seguir la creación de un proyecto y todo su seguimiento. Una de las partes donde se refleja la influencia recibida, es en el Diagrama de Gantt, dado que fue una de las formas más cómodas y visuales de ver el ritmo del trabajo.

-BDA: Bases de Datos. En esta asignatura aprendí el conocimiento necesario para operar sobre bases de datos, siendo esta nuestra base de SQL. SQL es necesario para operar sobre la base de datos que nuestra aplicación utiliza, siendo ahí donde se guardan los usuarios y las rutinas.

-ALT: Algorítmica. En esta asignatura obtuve el pensamiento crítico para resolver problemas, pensándolos bien para plantearlos de la mejor forma posible. Esto me ha hecho darle otro enfoque al problema que tenemos, y sin ese enfoque no habría sido capaz de acabar con la idea de poner el problema como una resolución de satisfacciones.

9.2. Posibles ampliaciones

En este punto veremos unas posibles mejoras que podrían ser aplicadas en futuras actualizaciones. Dado que el trabajo se centraba en la optimización de las rutinas y no en otras cosas, estas "posibles ampliaciones" serían necesarias en caso de ser una aplicación que saliera al mercado.

9.2.1. Ajustes

Una de las ampliaciones más importantes a realizar es un botón de ajustes para la cuenta del usuario. Dado que este trabajo se centra en la generación de las rutinas, para implementarlo no va a ser necesario ajustes, pero en caso de lanzarse la aplicación al mercado sería necesario un botón de ajustes con un mínimo de las siguientes acciones:

-Modo oscuro: poder elegir entre un modo claro y un modo oscuro en la aplicación, muy importante para los usuarios que prefieran el otro modo.

-Idioma: muy importante que cada persona pueda elegir el idioma deseado de la aplicación. Lo normal sería tener la aplicación en castellano, inglés y catalán. Para aspirar a un mercado más grande, estaría muy bien tener también la aplicación en idiomas como mandarín, francés, árabe e hindú, dado que son de los lenguajes más hablados en el mundo [20] (obviando el castellano y el inglés).

-Apartado de reportar bugs: en caso de localizar algún problema con la aplicación que impida la correcta utilización de ésta, hay un botón que enviará un mensaje directo desde el nombre del usuario que reporte al desarrollador donde se podrán añadir fotos y el texto que se necesite para describir cuál es el problema.

-Notificaciones: Se puede elegir el tipo de notificaciones que llegan, tanto por email como al teléfono. Dentro de esta opción se pueden pedir notificaciones los días que se crea que vas a poder entrenar, como recordatorio de que hoy es día de entrenamiento, así al recibirla te acordarás el usuario recordará que le toca entrenar.

-Tutorial de la aplicación: si esta opción está seleccionada, cuando se abra una nueva pantalla habrá una explicación de lo necesario para utilizar cada ventana nueva. Por defecto estaría desactivado.

9.2.2. Perfil

De normal el usuario ya tiene un perfil cuando se crea una cuenta, pero de momento no puedes acceder a él, está usado simplemente como una forma de tener las rutinas registradas.

Esta posible actualización se basaría en poder personalizar tu perfil con una foto, nombre (por defecto será tu nombre de usuario), y una pequeña descripción. Vendría también con imágenes por defecto, o te permitiría elegir una de tu galería.

9.2.3. Seguimiento

Como hemos observado en punto del estado del arte, la mayoría de las aplicaciones utilizan un seguimiento para poder guardar las marcas personales, tanto físicas como de las rutinas:

Sería implantar algo muy similar a lo que las aplicaciones actuales ya tienen, dado que es algo establecido como estándar en este tipo de aplicaciones, si la aplicación dispone de todas las funciones que tiene una aplicación normal, junto a lo de la generación de rutinas, la gente podría decidir sólo utilizar ésta. En caso contrario, sólo disponer de la generación de rutinas óptimas haría que la gente utilizara la aplicación para eso, pero luego si quieren llevar algún tipo de seguimiento habría que tener otra aplicación instalada.

9.2.4. Más de una rutina

A partir de obtener una rutina es posible que el usuario no le apetezca esa rutina, o quiera ver otra diferente. Con esta opción, se plantea la opción de pedir desde el primer momento más de una rutina, o pedir una diferente tras recibir la primera.

9.2.5. Demostración de los ejercicios

A partir de obtener una rutina, sería muy cómodo para la persona que lo está usando, que cada ejercicio que saliera tuviese una demostración de cómo hacerlo, ya que es muy posible que por el nombre no se sepa qué ejercicio es, o simplemente que no se esté seguro de si la técnica que se está empleando es la correcta.

9.2.6. Optimización de calendario

Otro detalle que no está en muchas aplicaciones sería el hecho de poder organizar una semana, que eligiendo los días que tienes libres para entrenar, la aplicación fuera capaz de organizar los entrenamientos optimizando los días de descanso y todo.

Referencias

[1] Sarker, R. A., & Newton, C. S. (2008). Optimization Modelling: A Practical Approach. Retrieved from https://books.google.es/books?id=_ZFnJ4hEZLsC Capítulo 1.4

[2] Sarker, R. A., & Newton, C. S. (2008). Optimization Modelling: A Practical Approach. Retrieved from https://books.google.es/books?id=_ZFnJ4hEZLsC Capítulo 1.4.1

[3] Freitas de Salles, B., Simão, R., Miranda, F. et al. Rest Interval between Sets in Strength Training. Sports Med 39, 765–777 (2009). https://doi.org/10.2165/11315230-000000000-00000

[4] GLPK (GNU Linear Programming Kit) https://www.gnu.org/software/glpk/ Consultado el 18/06/2023

[5] Introduction to lp_solve 5.5.2.11 https://lpsolve.sourceforge.net/5.5/ Consultado el 18/06/2023

[6] Considerations when using Choco-solver. https://choco-solver.org/docs/considerations/ Consultado el 18/06/2023

[7] Quadratic programming https://www.ibm.com/docs/en/icos/20.1.0?topic=areas-quadratic-programming Consultado el 18/06/2023

[8] Optimizer options https://www.ibm.com/docs/en/icos/20.1.0?topic=cplex-optimizer-options Consultado el 18/06/2023

[1] Sharma, Sheetal & Sarkar, Darothi & Gupta, Divya. (2012). Agile Processes and Methodologies: A Conceptual Study. International Journal on Computer Science and Engineering. 4. https://www.researchgate.net/publication/267706023_Agile_Processes_and_Methodologies_A_Conceptual_Study

[2] B. Boehm, "Anchoring the software process," in IEEE Software, vol. 13, no. 4, pp. 73-82, July 1996, doi: 10.1109/52.526834. https://ieeexplore.ieee.org/document/526834

[11] Jefatura del Estado. Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. Consultado el 19/05/2023. https://www.boe.es/eli/es/lo/2018/12/05/3/con

[12] Y. Liu et al., "Design of password encryption model based on AES algorithm," 2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Kunming, China, 2019, pp. 385-389, doi: 10.1109/ICCASIT48058.2019.8973003. https://ieeexplore.ieee.org/document/8973003

[13] N. Su, Y. Zhang and M. Li, "Research on Data Encryption Standard Based on AES Algorithm in Internet of Things Environment," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 2071-2075, doi: 10.1109/ITNEC.2019.8729488. https://ieeexplore.ieee.org/document/8729488

[14] https://es.talent.com/salary?job=programador Consultado el 26/05/2023

[15] Dennis Tobar. cuanto cuesta la publicación de una app https://support.google.com/googleplay/thread/17857232/cuanto-cuesta-la-publicación-de-una-app?hl=es Consultado el 26/05/2023

[16] D. Richard Hipp. About SQLite. https://www.sqlite.org/about.html Consultada el 12/05/2023

[17] D. Richard Hipp. SQLite Is Transactional. https://www.sqlite.org/transactional.html Consultada el 12/05/2023

[18] D. Richard Hipp. Most Widely Deployed SQL Database Engine. https://www.sqlite.org/mostdeployed.html Consultada el 12/05/2023

[19] Singh, Ajit, Linear Programming Problem Solving Simplex Method (February 25, 2022). Available at SSRN: https://ssrn.com/abstract=4043703 or http://dx.doi.org/10.2139/ssrn.4043703

[20] Berlitz. The most spoken languages in the world. 06 de febrero de 2023. Consultado el 19/05/2023. https://www.berlitz.com/blog/most-spoken-languages-world

Anexo 1: Manual del usuario

En este apartado del anexo, se realiza un pequeño manual para así poder ayudar al usuario a poder entender la aplicación.

Cuando abrimos la aplicación por primera vez, nos encontramos con la opción de iniciar sesión o registrarnos:

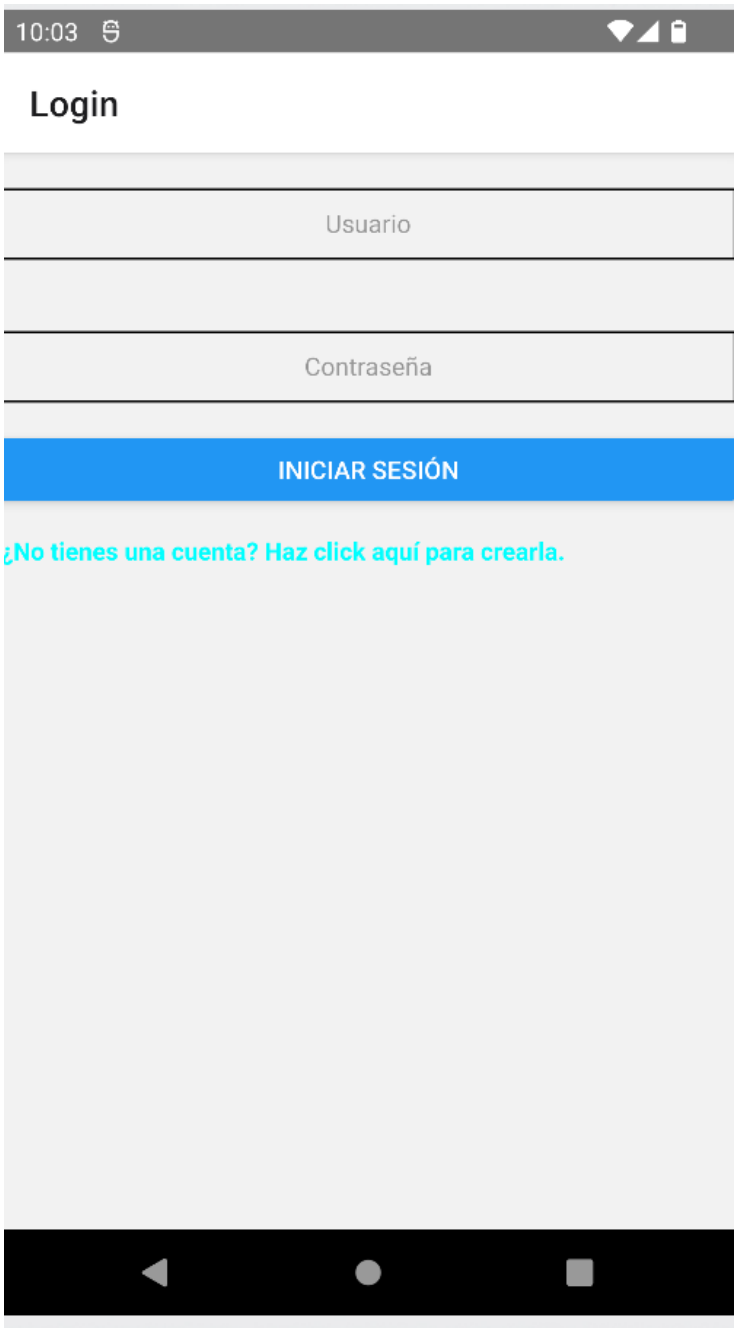

Figura 45: Captura de la pantalla del Login

Si por otro lado queremos registrarnos, hacemos click en registrarnos y rellenamos los siguientes campos.

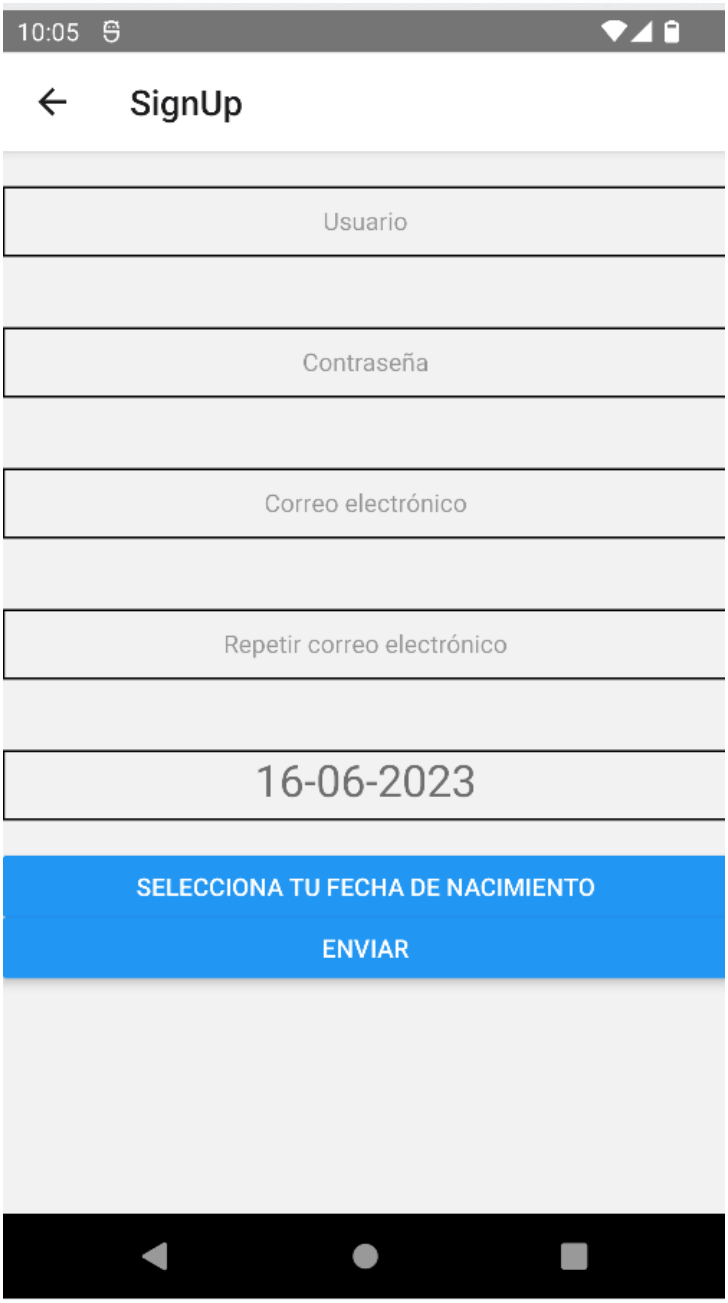

Figura 46: Captura de la pantalla de crear una cuenta

A partir de ahí, la pantalla de completar el registro si los datos son correctos.

Figura 47: Captura de la pantalla de completar el perfil

Ahora podemos confirmar los datos y nos llevará al home. Por otro lado, si hemos iniciado sesión, tendremos el home directamente.

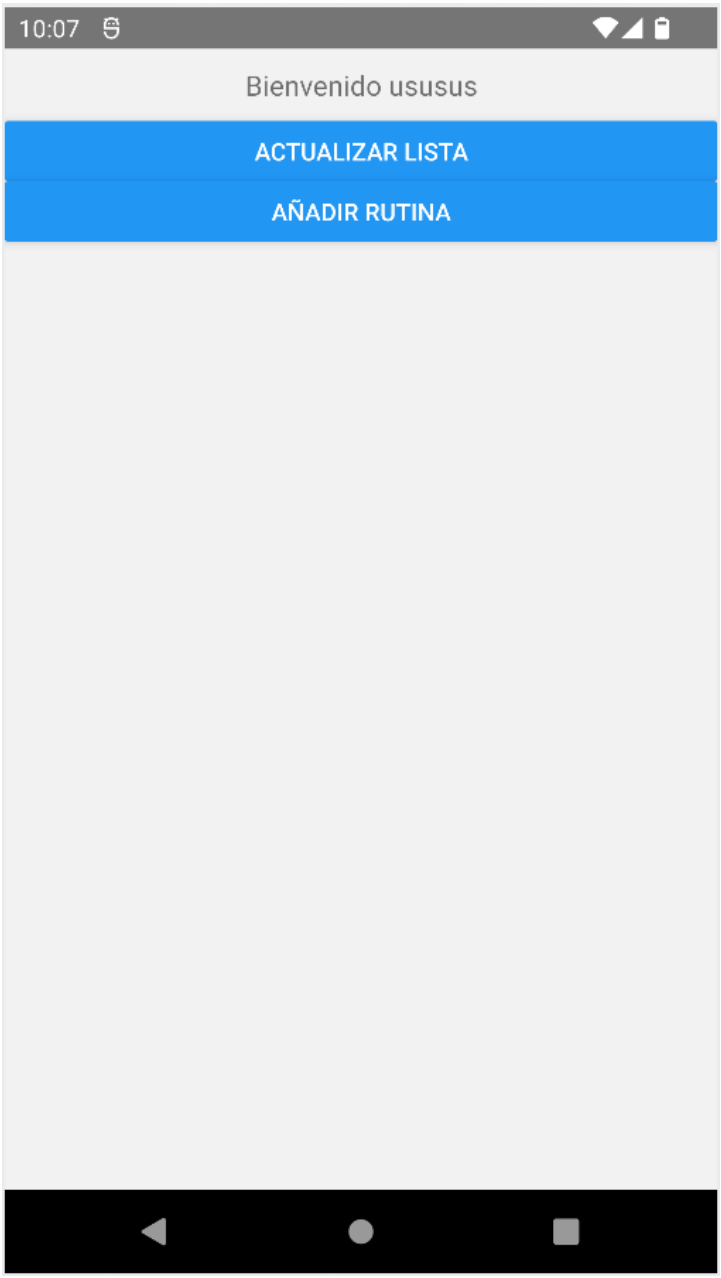

Figura 48: Captura de la pantalla Home

Como se puede observar, en esta ventana tenemos una lista de nuestras rutinas. Ahora mismo está vacía porque la acabamos de crear, pero podemos añadir una rutina haciendo click en el botón para crearla.

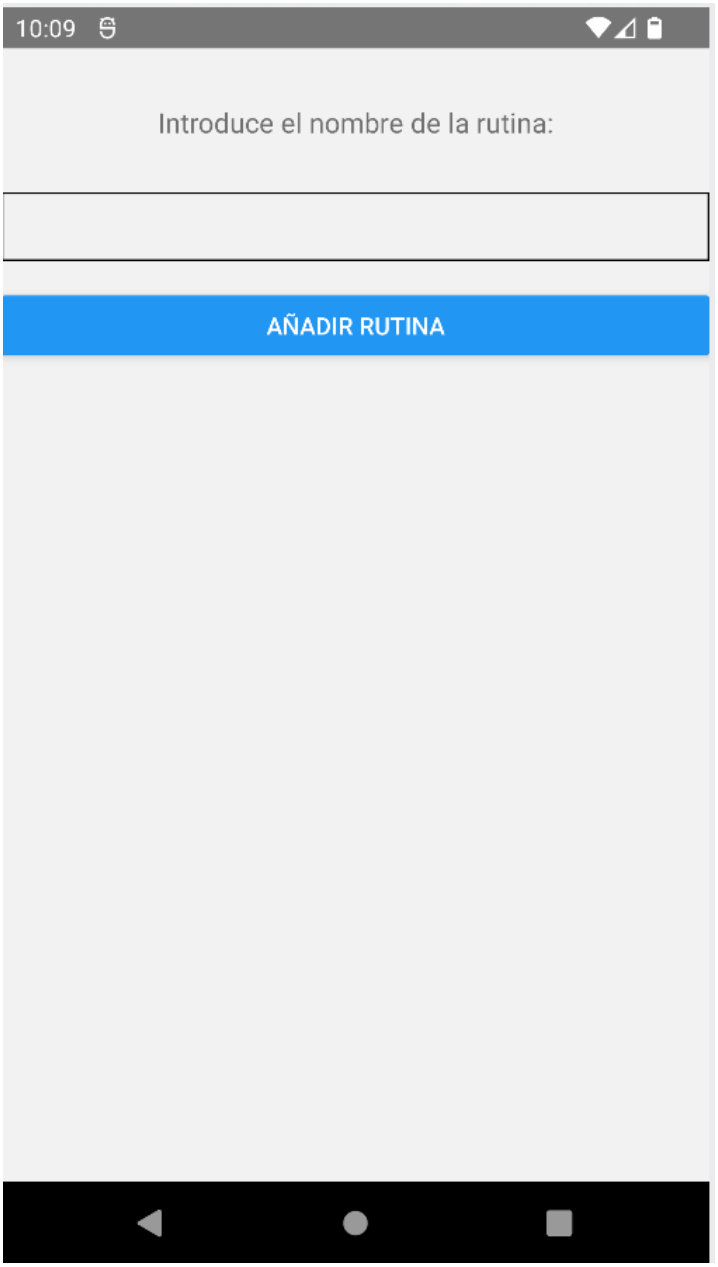

Figura 49: Captura de la pantalla de crear una rutina

Al escribir un nombre, si no está siendo utilizado en otra rutina se creará una con ese nombre.

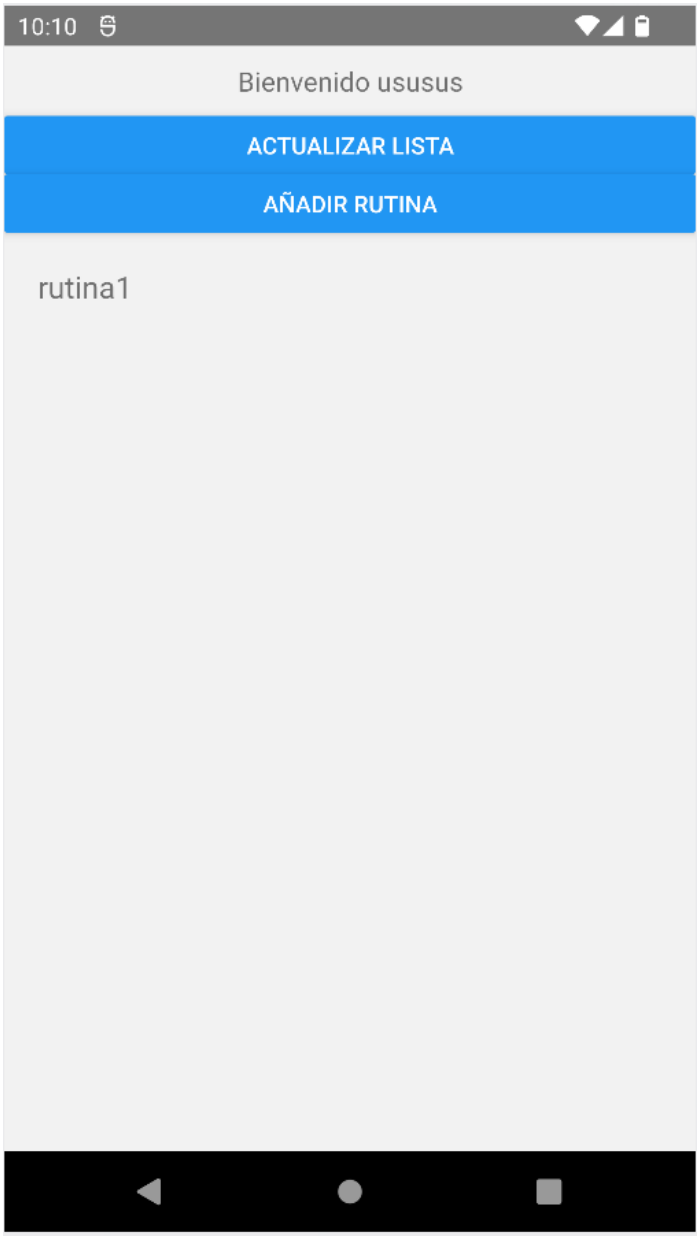

Figura 50: Captura de la pantalla Home con una rutina creada

En caso de no necesitar crearla, podemos pulsar cualquier rutina para editarla.

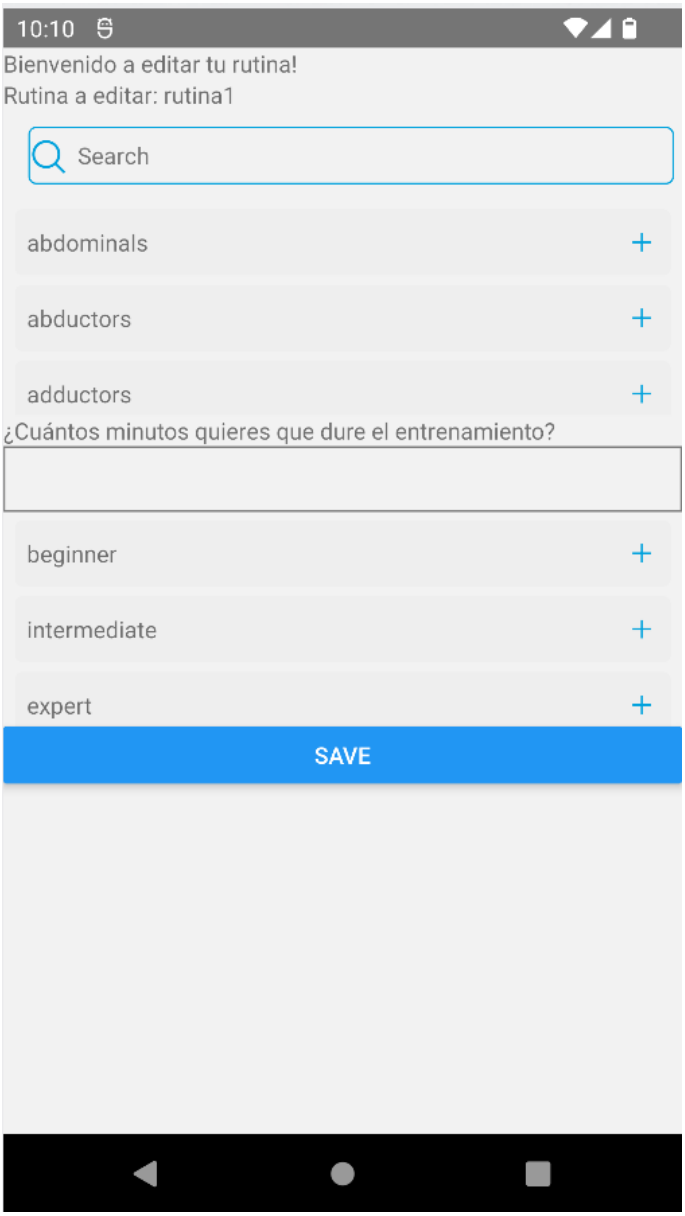

Figura 51: Captura de la pantalla de edición de rutinas

Cada rutina tiene sus parámetros de tiempo, dificultad y ejercicios elegidos. Tras seleccionar todo, podemos pulsar en generarla.

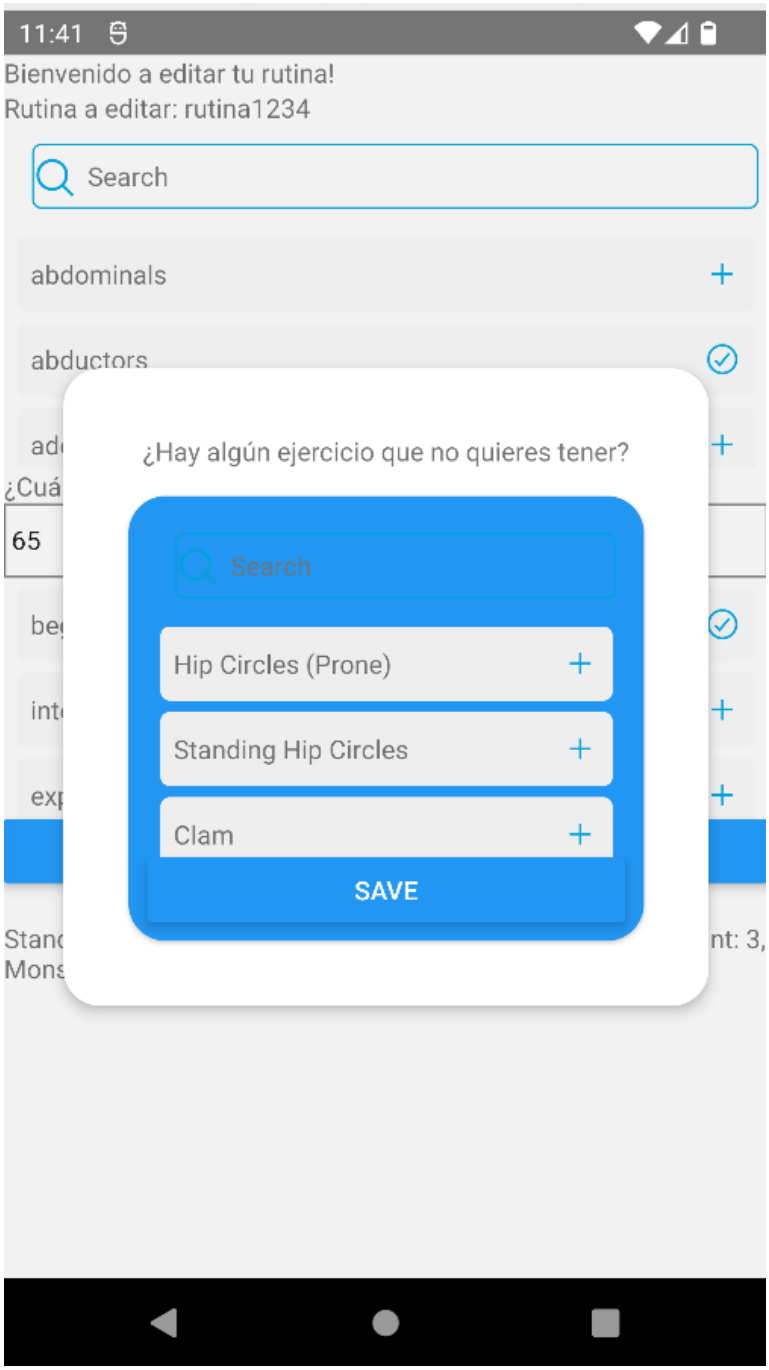

Figura 52: Captura de la pantalla para evitar ejercicios en una rutina

Ahora podemos observar la lista de ejercicios que nos devolvería. Se adjuntan dos capturas, una que marcaría el ejercicio llamado “Standing Hip Circles” y otra que no. La diferencia será que en la primera no aparecerá como ejercicio, y en la otra sí.

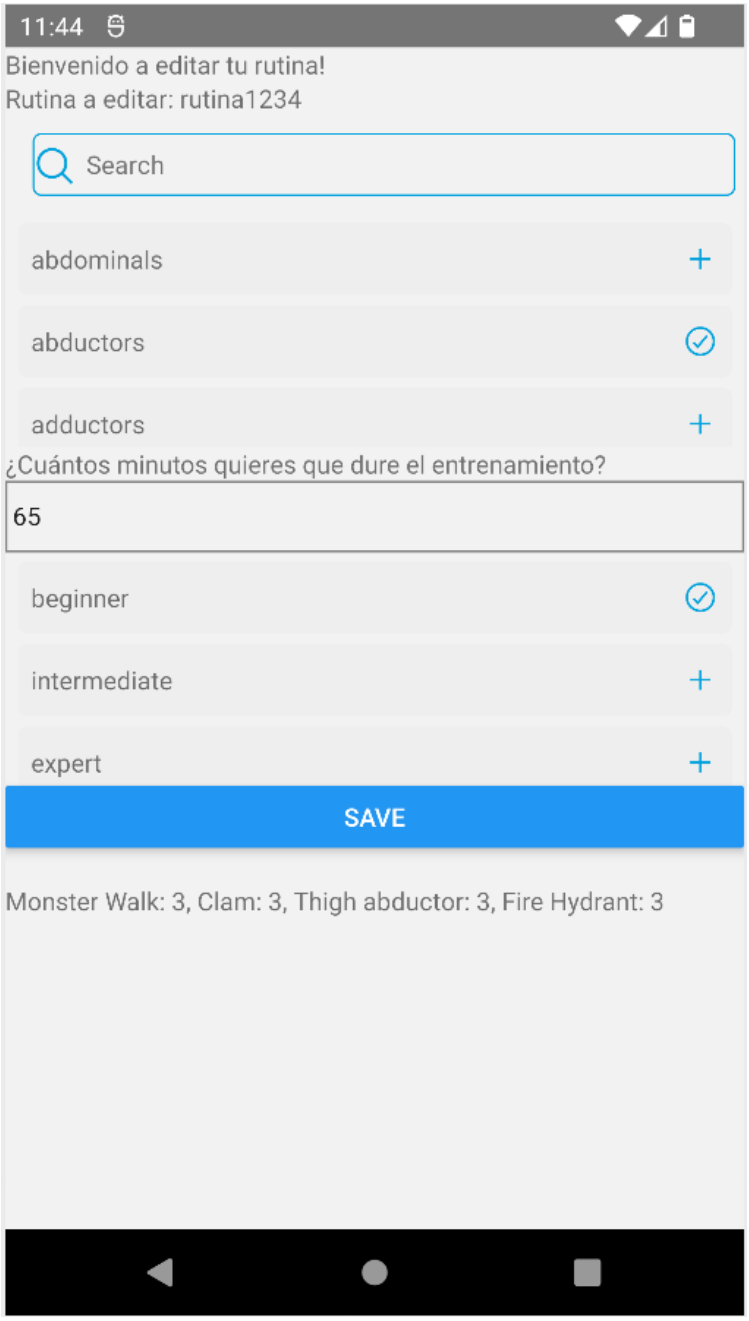

Figura 53: Captura de la rutina tras evitar un ejercicio

Por un lado, podemos observar cómo al haber elegido el ejercicio no sale luego en la rutina. Es una forma de vetar ese ejercicio.

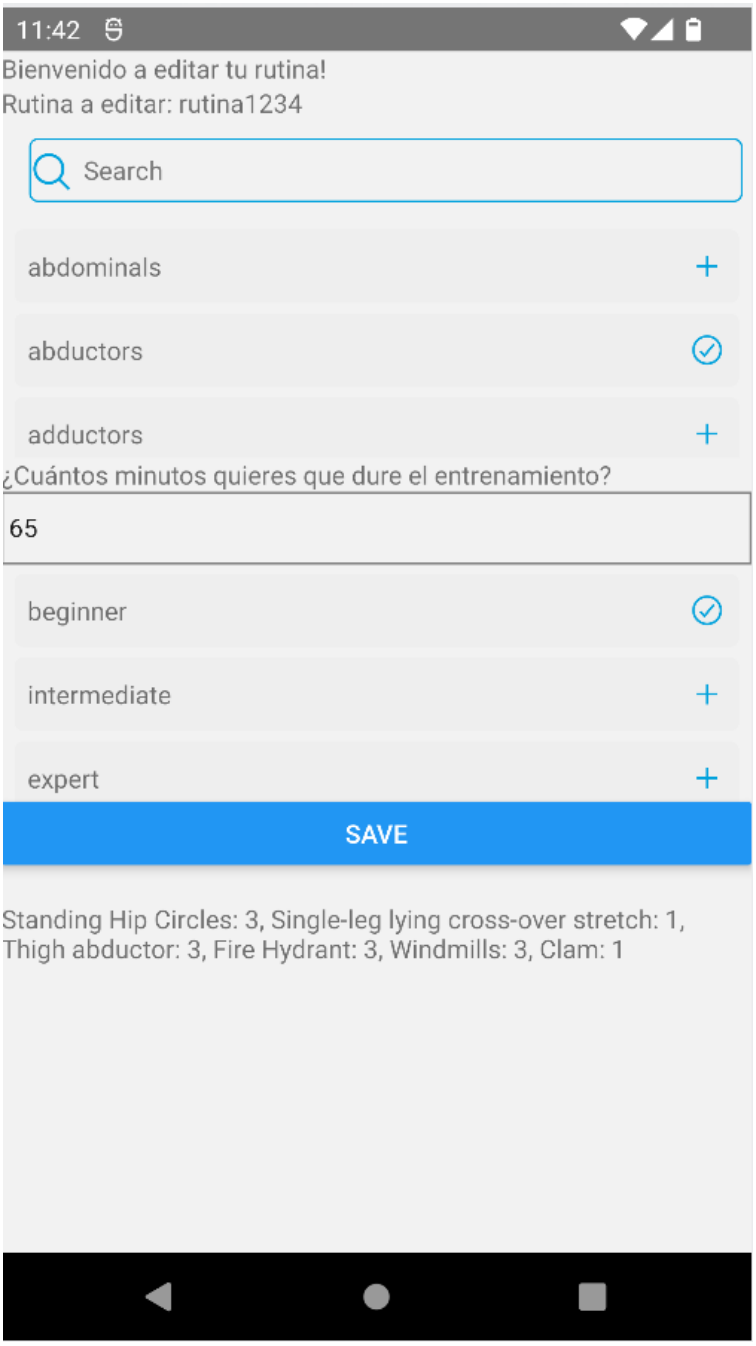

Figura 54: Captura de la rutina sin evitar ningún ejercicio

Por otro lado, podemos observar que aquí no se ha marcado, por lo que, al ser un ejercicio óptimo para la rutina por los parámetros comentados anteriormente, sí que sale en la rutina.

En general, tras haber creado la rutina, abajo obtenemos una lista de los ejercicios y el número de series a realizar.

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**			**X**	
ODS 5. **Igualdad de género.**			**X**	
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

-Salud y bienestar: el gimnasio es un lugar en el cual parte de la gente va para simplemente estar un poco en forma. Hacer ejercicio es algo muy importante para mantener el cuerpo ejercitado, ya que puede evitar muchos tipos de enfermedades que se podrían tener simplemente por el hecho de ser una persona muy sedentaria. Ahí es donde esta aplicación pretende ayudar a la gente a interesarse más por el gimnasio, o simplemente verlo como algo más posible debido a la facilidad añadida de tener la aplicación a tu lado para poder entrar. Es por eso por lo que salud y bienestar es donde más se enfoca el trabajo, pese a no encajar en la mayoría de los objetivos de desarrollo sostenible, salud y bienestar es donde más brilla el trabajo en sí, al estar centrado en el tema del gimnasio y comentar en parte, de forma breve, la alimentación relacionada con el gimnasio. Es por eso por lo que, personalmente, pienso que el grado de relación sería bastante alto.
-Educación de calidad: aquí sí que es algo más bajo dado que tampoco siento que el trabajo aporte una educación de calidad sobre ningún tema de los que se han comentado en estas setenta y pico páginas de trabajo, pero sí que añade un poco de unos comentarios sobre el gimnasio, que pese a alejarse de aportar mucho, sí que comentan un poco la base de todo, y al estar ligados al tema de la importancia del ejercicio en nuestras vidas y un mínimo de la alimentación, trata un poco de educar también a cualquier persona que lo lea con estos valores.
-Igualdad de género: no tiene mucho impacto en la aplicación, pero cuando escoges tu género es sólo un ajuste en tu perfil, no va a cambiar nada que seas un hombre, una mujer, u otro, ya que no afecta en ningún cálculo de la aplicación, excepto en datos como el IMC (Índice de Masa Corporal) que son fórmulas ya establecidas en la sociedad. Resumiendo, no importa cuál sea el género del usuario para utilizar esta aplicación, porque ésta no hace distinciones.