
**UNIVERSIDAD DE TALCA**
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

# Plataforma web para crear, administrar, consultar y compartir planes de entrenamiento y de alimentación.

**FELIPE ALEJANDRO HERRERA HORMAZÁBAL**

Profesor Guía: DANIEL ANTONIO MORENO CÓRDOVA

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
mes, año

# CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.


Curicó, 2022

*Dedicado a ...*

## AGRADECIMIENTOS

A mi madre, por entregar siempre lo que tuvo a su alcance para proveer estabilidad a mi vida y oportunidades de formación en distintos ámbitos.

A los miembros de mi familia que me han apoyado de alguna u otra manera. A mi padre por procurar un ambiente libre de preocupaciones. A mi hermano por estar presente incondicionalmente. Y especial mención a Catalina Carvajal por ser un apoyo emocional importante en momentos en que resultara necesario.

A mis amigos y conocidos que han sido parte de mi proceso universitario, siendo hoy protagonistas de los mejores recuerdos de esta etapa de mi vida. En especial a Nicole P. Sánchez, Sofía Cornejo, Alejandro Naranjo, Patricio Castro y Reinaldo Jerez.

A aquellos docentes que han sabido orientarme tanto en el ámbito académico como fuera de éste: Daniel Moreno y Edgardo Ortiz.

A la comunidad del software libre que entrega herramientas e información. A la comunidad de Stackoverflow por atender dudas. Y a todo el que provee sus conocimientos de manera abierta.

Finalmente, agradecimientos a todos aquellos que formaron parte de este trabajo, participando en encuestas, revisiones, etc.

## TABLA DE CONTENIDOS

página

# ÍNDICE DE FIGURAS

página

# ÍNDICE DE TABLAS

página

# RESUMEN

Junto con el crecimiento de la comunidad fitness han surgido alternativas tecnológicas que permiten facilitar las distintas actividades asociadas a alimentación y ejercitación, o se ha adaptado el uso de otras herramientas con fines distintos, pero aún existe la oportunidad de optimizar la atención a dichas necesidades.

En este documento se presentan las especificaciones y el proceso de desarrollo de una aplicación web, que logra aunar distintas funcionalidades observadas en aplicaciones actuales, cubriendo además algunas nuevas necesidades detectadas.

Este desarrollo fue guiado por una adaptación de la metodología ágil *Scrum*. Para la captura de requisitos se realizó un diagnóstico del microentorno, encuestas, y evaluaciones continuas usando *Concurrent Think Aloud*. Además se hizo uso de *Test-driven development*.

Como resultado de este proyecto se obtuvo un producto viable mínimo que entregara valor a los usuarios. En concreto es una red social que permite registrar, ejecutar y compartir las actividades de alimentación y ejercicio físico, además de informar al respecto.

La aplicación fue evaluada mediante una encuesta que incluía la escala SUS de usabilidad y otra parte personalizada. En ella se expone que la aplicación cumple con un estándar de usabilidad (90 puntos sobre un mínimo aceptable de 80), y que los objetivos han sido cumplidos sobre un 90 % según validación de los usuarios.

Finalmente, se señala el trabajo futuro detectado, el cual incluye una puesta en marcha en producción, optimizaciones, atributos de calidad no funcionales, y algunas nuevas funcionalidades.

# 1. Introducción

En este capítulo se describe la propuesta de proyecto, junto a la información que permite comprender las motivaciones para su realización. Específicamente, se mencionan el contexto donde se desarrolla el proyecto, la definición del problema/oportunidad a atender, la propuesta como tal, objetivos general y específicos, además del alcance del proyecto.

## 1.1. Contexto

Los buenos hábitos alimenticios y la actividad física regular suelen ser renombrados al hablar de buena salud. Y no es casualidad, puesto que ha sido demostrado que aquellas personas que mantienen estos buenos hábitos de manera sostenida son más felices y saludables (tanto física como mentalmente) [26, 65]; todo esto debido a que se estimula la producción de hormonas y neurotransmisores (el cuarteto de la felicidad) relacionados con el equilibrio emocional, resultando incluso adictivo [13, 15, 35, 11, 62].

En conjunto a lo anterior, puede observarse muchos efectos somáticos y estéticos benéficos, tales como: disminución de la resistencia a la insulina, mejora del flujo sanguíneo, prevención de algunas enfermedades, retraso del envejecimiento, mejora de color y textura de la piel, reducción de sobrepeso y celulitis, entre otros [3, 14, 4].

En consecuencia, cada vez hay más gente poniendo en práctica como hábitos de vida saludable el mantener una nutrición adecuada y realizar ejercicio físico periódicamente, o lo que es lo mismo, cada vez más personas se suman a la comunidad **Fitness** [17]. (Más información en la Sección 4.1.2).

Sin embargo, dentro de la comunidad fitness existen muchos subgrupos (e incluso individuos) que practican distintas formas de entrenamiento, como por ejemplo

montar bicicleta, ir al gimnasio, defensa personal, etc. Así también existen comunidades dedicadas a la alimentación equilibrada en sus distintas formas, las cuales pueden o no coincidir con las anteriores. Pero estos hábitos deben ser planificados, variando en orden, cantidades o tiempos, los cuales dependen de cada persona, aunque pueden ser compartidos debido a los aspectos en común. También hay que tener en consideración que debe existir un equilibrio para que sea saludable [62]. Es por esto que resulta de gran utilidad contar con una herramienta adecuada al respecto, y efectivamente las hay.

En los últimos años han surgido distintas alternativas tecnológicas atendiendo estas necesidades, a la vez que se ha hecho uso de otras alternativas previamente existentes. Ejemplos de lo anterior son el uso de planillas de datos, la comunicación por redes sociales, y las distintas aplicaciones fitness que abordan la temática desde distintos enfoques. Es en este ámbito en que se desarrolla el presente proyecto.

## 1.2. Definición del problema

Como se mencionó anteriormente, existen distintas herramientas tecnológicas que atienden necesidades del fitness, las cuales son usadas en singular o en conjunto. No obstante, estas alternativas presentan ciertos contratiempos. A continuación se presentan los problemas tanto en aplicaciones de propósito general como en aplicaciones específicas.

Dentro de las aplicaciones de propósito general, parte de las más utilizadas son las redes sociales, las cuales se usan para mantenerse en contacto con otras personas, organizar eventos, conocer personas con los mismos intereses, y así también compartir sus avances, rutinas e incluso dietas, utilizando formatos genéricos. El problema de las redes sociales de propósito general en este contexto, además de la carencia de funcionalidades específicas, es el ruido generado por el contenido de otra índole.

Por otro lado están las distintas aplicaciones específicas, enfocadas en las necesidades referentes al cuidado físico. Dentro de toda la variedad existente, hay aplicaciones dedicadas a las distintas áreas de ejercicios, a la alimentación, o incluso ambas. Además están las aplicaciones de administración de gimnasios. El problema con ellas es que, al estar enfocadas en aspectos específicos, resultan en una experiencia de usuario desgastante, puesto que las características son limitadas a su especialización, y termina siendo necesario la utilización de una gran cantidad de aplicaciones

independientes.

Dado lo anterior, se identifica una oportunidad de desarrollo que permita unificar en una plataforma las distintas características que ya han sido implementadas y probadas en otras aplicaciones, siempre enfocándose en el público objetivo y sus necesidades particulares. (Más detalle respecto a las funcionalidades observadas en otras aplicaciones y propuestas de interesados en Sección 4.1.1, Sección 4.2 y Anexo A).

## 1.3. Trabajo relacionado

Como se mencionó en la Sección 1.2, existen por un lado aplicaciones sociales de propósito general utilizadas para organizar y compartir ejercicios y sus resultados, así como también aplicaciones específicas enfocadas en aspectos particulares del fitness. Además, existen opciones más tradicionales como planillas o alternativas no tecnológicas. A continuación se presentan ejemplos concretos de las dos primeras.

### 1.3.1. Redes sociales genéricas

Son aplicaciones creadas para la interacción social. Dentro de las más conocidas encontramos a Facebook, Instagram y Youtube.

Facebook es una gran red social multipropósito en la cual se pueden publicar eventos, fotografías, videos, texto, noticias, enlaces, entre otros, o incluso una mezcla de ellos. También se caracteriza por contar con redes tanto de amigos como de seguidores.

Instagram por su parte, está enfocada en la publicación de fotografías y videos, contando con una red de seguidores. Entre las utilidades que tienen ambas aplicaciones en el ámbito del fitness son las siguientes: compartir entrenamientos realizados y avance en forma de texto o imágenes, hacer publicidad personal o de productos, publicar eventos.

Youtube es una de las aplicaciones web más grandes en el mundo para compartir videos, y el mundo del fitness no se queda ajeno, puesto que hay diversos canales dedicados a este ámbito.

Debido a sus beneficios de comunicación y sus propios objetivos, las podemos considerar como herramientas facilitadoras en vez de competencia.

### 1.3.2. Aplicaciones específicas

Corresponde a aquellas en que su enfoque es la disposición de herramientas de uso personal, que satisfagan las necesidades detectadas de un aspecto particular. Entre ellas pueden encontrarse opciones como las siguientes:

- Technutri [43]: Red social que permite compartir alimentos consumidos, recetas, y su aporte nutricional.
- JEFIT [21]: Red social que permite ejecutar rutinas de ejercicios y compartir la rutina realizada.
- 8fit [5]: App de uso personal que permite el registro de actividades y comidas realizadas, basado en información predefinida.
- Runtastic [6]: App para registrar actividades deportivas que hace uso de los sensores del teléfono celular. Este tipo de aplicaciones no serán consideradas en el presente proyecto debido a su enfoque y objetivos.

En la Sección 4.1.1 se presenta un análisis comparativo entre siete alternativas, junto a la presente propuesta.

## 1.4. Propuesta de solución

En base a la oportunidad de desarrollo identificada anteriormente, se propone la implementación de una plataforma web en forma de red social temática, enfocada en el área del fitness. Esta plataforma deberá contar con distintos módulos o componentes, dentro de los cuales se identifican tres grandes ejes principales: ejercicio físico, alimentación, y socialización.

Con respecto al área de la actividad física, esta plataforma permitirá gestionar rutinas y planes de entrenamiento, los cuales serán creados a partir de un listado de ejercicios predefinidos, y luego podrán ser ejecutados y publicados. Adicionalmente se contará con un calendario personal para facilitar la organización de estos entrenamientos.

Con respecto a la alimentación, se dispondrá de un listado de alimentos básicos y recetas, a partir de los cuales se podrán crear nuevas recetas. Estas podrán ser

utilizadas para la creación de planes alimenticios, o ser agregadas a los planes de entrenamiento.

Con respecto a la socialización, se podrán compartir los distintos elementos creados o ejecutados en la aplicación. Para dicho fin se contará con una sección de noticias o actividades.

Por medio de estas características mencionadas se espera satisfacer las necesidades de la comunidad del fitness amateur. Además, se sentará una base para extender en el futuro esta plataforma a una mayor parte del mundo deportivo y nutricional. Para ver un mayor detalle de las características a desarrollar revisar Sección 1.6 (Alcance) y Capítulo 4 (Análisis y requisitos) del presente documento.

## 1.5. Objetivos

En esta sección se detallan los objetivos general y específicos que se espera concretar para el proyecto.

### Objetivo general

Facilitar la realización sostenida de hábitos de vida saludable asociados al fitness, mediante el desarrollo de una plataforma web que permita gestionar y compartir planes de entrenamiento y de alimentación.

### Objetivos específicos

1. Simplificar el seguimiento de planes de ejercicio, y el control de la ejecución de los mismos.

2. Simplificar el seguimiento de planes de alimentación, y la ejecución de los mismos.

3. Facilitar la generación y mantención de hábitos del fitness.

4. Promover la vida sana a través de la interacción entre aficionados al fitness, ya sea compartiendo rutinas y recetas, compitiendo entre ellos, o informándose de los distintos planes.

5. Proveer información oportuna referente a ejercitación física y alimentación.

## 1.6. Alcance

Para este proyecto se desarrollará un producto viable mínimo (MVP) de la propuesta, es decir, una base funcional con las características necesarias para ser utilizado por usuarios finales, pero sin perfeccionar detalles ni desarrollar la totalidad de las funcionalidades ideales. Esta base será diseñada de tal forma que facilite su escalabilidad, y estará limitada (al menos y no únicamente) según el siguiente listado:

- Se desarrollará una aplicación web escalable, que permita conectar fácilmente proyectos de la misma índole, aunque no se incluirán esos proyectos adicionales.
- La interfaz de la aplicación solo contará con una versión en idioma español latinoamericano.
- El contenido de la aplicación será información de ejemplo, por lo tanto será limitada y no necesariamente correcta.
- El diseño de la interfaz estará basado mayoritariamente en librerías de terceros, al igual que íconos y otros aspectos que son complementarios a la ingeniería de software.
- No se incluye un modelo de negocio.
- No se incluye una estructura física de servidores ni una puesta real en producción.

Las características a desarrollar se encuentran detalladas en el Capítulo 4 (Análisis y requisitos).

## 1.7. Conceptos básicos

Debido a que existen distintas acepciones para algunos términos, se propone una desambiguación para el actual proyecto, en la que destacan los siguientes conceptos que serán usados en todo el documento (el listado completo de definiciones puede verse en el Glosario):

- **Implementación/implemento.** Objetos, herramientas u equipos especializados, empleados para la realización de ejercicios físicos. Ejemplo: mancuerna, cuerda, entre otros.

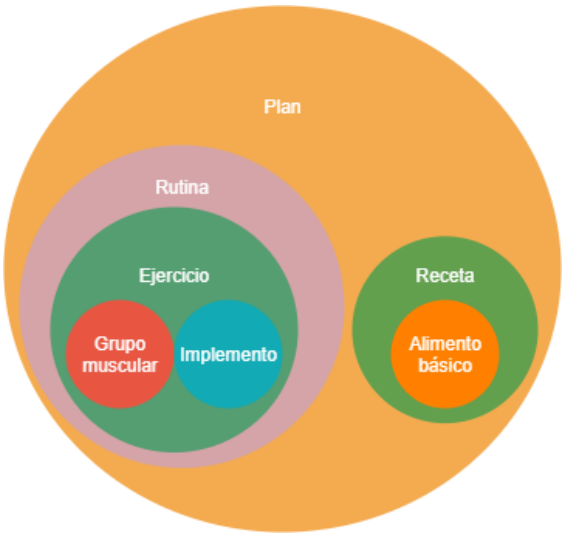


Figura 1.1: Relación conceptos básicos.

- **Grupo muscular.** Conjunto de músculos relacionados que actúan en conjunto para la realización de un movimiento. Ejemplo: Pectorales, glúteos, dorsales, entre otros.
- **Ejercicio.** Acción motora utilizada para el desarrollo de un músculo esquelético o un grupo muscular. Ejemplo: Sentadilla, peso muerto, dominada supina, entre otros.
- **Rutina.** Conjunto de ejercicios seleccionados y ordenados para su ejecución con un objetivo determinado. Ejemplo: rutina para desarrollo muscular de piernas que incluye sentadillas y peso muerto.
- **Plan de ejercicios.** Conjunto de rutinas seleccionadas y ordenadas en un periodo de varios días, con un objetivo determinado. Ejemplo: dos semanas con rutina de torso en días lunes y rutina de piernas en días jueves.
- **Alimento.** Representación de un alimento real, pudiendo ser básico (no contiene una receta para prepararlo) o preparado (contiene una receta y está compuesto de otros alimentos). Ejemplo de alimento básico: naranja. Ejemplo de alimento preparado: Cheesecake de maracuyá.
- **Plan alimenticio.** Conjunto de dietas seleccionadas y ordenadas para un periodo de varios días, con un objetivo determinado.

- **Plan de entrenamiento.** Rutinas y dietas seleccionadas y ordenadas para un periodo de varios días, es decir, mezcla de plan de ejercicios y plan alimenticio.

En la Figura 1.1 puede observarse cómo se conforman dichos conceptos al relacionarse entre ellos.

## 1.8. Descripción del documento

En el capítulo dos Antecedentes técnicos, se exponen los conceptos de desarrollo de software necesarios para comprender el resto del documento. Esto es, arquitectura, tecnologías utilizadas, metodologías y herramientas.

En el capítulo tres Metodología de trabajo, se muestran las metodologías utilizadas en los distintos procesos del proyecto, esto es, análisis, diseño, desarrollo y evaluación.

En el capítulo cuatro Análisis y requisitos, se presenta un breve estudio de mercado que considera competidores y usuarios, además de los requisitos resultantes.

En el capítulo cinco Arquitectura y diseño de software, se explican las decisiones de diseño, junto a la abstracción resultante en términos de arquitectura, base de datos, diagramas de clases y mapa de navegación.

En el capítulo seis Implementación, se expone el avance real en cada *sprint*, comparado con el objetivo de avance definido durante la planificación.

En el capítulo siete Verificación y evaluación, se explican las pruebas automatizadas ejecutadas, las pruebas de usuario realizadas, y los resultados de estas últimas.

En el capítulo ocho Conclusión y trabajo futuro, se presenta un compendio del proyecto que incluye resultado final obtenido, cumplimiento de objetivos, aprendizajes, oportunidades de mejora de la aplicación, y pasos siguientes para la puesta en marcha real del sistema en ambiente de producción.

En los Anexos se presentan los resultados completos de las encuestas realizadas.

# 2. Antecedentes técnicos

En este capítulo se entrega la información teórica y técnica necesaria para comprender el proceso de desarrollo de la aplicación y las decisiones tomadas durante este. En particular, se explican conceptos referentes a aplicaciones web, la arquitectura cliente-servidor, las tecnologías utilizadas, la metodología ágil Scrum, y así también de las pruebas utilizadas. Cabe mencionar que los conceptos más relevantes serán incluidos en el Glosario.

## 2.1. Aplicación web

Una aplicación web es aquella que se ejecuta en cualquier navegador web, tal que no es necesario instalarla, sino que hace uso de recursos alojados en algún servidor remoto. La principal ventaja frente a otro tipo de aplicaciones es su multiplataformidad, es decir, que pueden ser accedidas desde una amplia variedad de dispositivos, independiente de sus características [57]. En el desarrollo de aplicaciones web suelen encontrarse dos especializaciones principales:

- **Frontend**: Corresponde a la estructuración de páginas utilizando principios de diseño, junto a la lógica presente en la interfaz.
- **Backend**: Corresponde a la lógica del manejo de datos, tanto para almacenarlos, transferirlos, y otras funciones de gestión [10].

### 2.1.1. Single-Page Application (SPA)

Una aplicación de página única o SPA es una aplicación web que funciona en una única página, esto es, los recursos son cargados dinámicamente a través del

DOM (interfaz que permite modificar la estructura y el contenido de una página web), evitando así recargar la aplicación. Lo anterior tiene como finalidad mejorar la experiencia de usuario, disminuyendo los tiempos de carga, asimilando así a una aplicación de escritorio [18].

Para entregar las ventajas ya mencionadas, las SPAs suelen funcionar orientadas a eventos, es decir, responden frente a distintos sucesos, optimizando la experiencia del usuario. En otras palabras, suelen ser reactivas.

### 2.1.2. Programación reactiva

Los últimos años se ha establecido como tendencia la programación reactiva, debido a las necesidades que ha traído el crecimiento de la web [51].

Una aplicación reactiva se caracteriza por cumplir con 4 características definidas en el manifiesto reactivo [32]:

- Responsividad: El sistema provee tiempos de respuesta rápidos sin afectar la calidad.
- Resiliencia: Existe tolerancia a errores, manteniendo la responsividad.
- Elasticidad: Se mantiene la responsividad frente a situaciones de estrés.
- Orientación a mensajes: Los distintos componentes interactúan mediante mensajes asíncronos, para así minimizar el acoplamiento [54].

Es importante señalar que un sistema solo puede ser reactivo en la medida que sus componentes lo sean, por lo que deben aplicarse principios de diseño, además de utilizarse tecnologías que permitan el desarrollo de una arquitectura adecuada [32].

## 2.2. Arquitectura cliente-servidor

La arquitectura cliente-servidor es un tipo de arquitectura en la cual muchos clientes solicitan y reciben servicios desde un servidor centralizado [8].

Un cliente puede entenderse como un equipo final (computador, smartphone, etc), en el cual un usuario interactúa con una aplicación (**aplicación cliente**), la cual se encarga de todo lo referente a frontend. Cabe señalar que pueden existir distintas aplicaciones cliente consumiendo un mismo servicio.

Un servidor es una computadora que entrega información a través de una red. Normalmente no son usados directamente por personas, sino que son accedidos por medio de otros dispositivos. En estos se encuentra alojada la aplicación encargada de la lógica de negocios (backend) y las bases de datos.

A continuación se muestra el mapa físico de una arquitectura cliente-servidor (Figura 2.1).

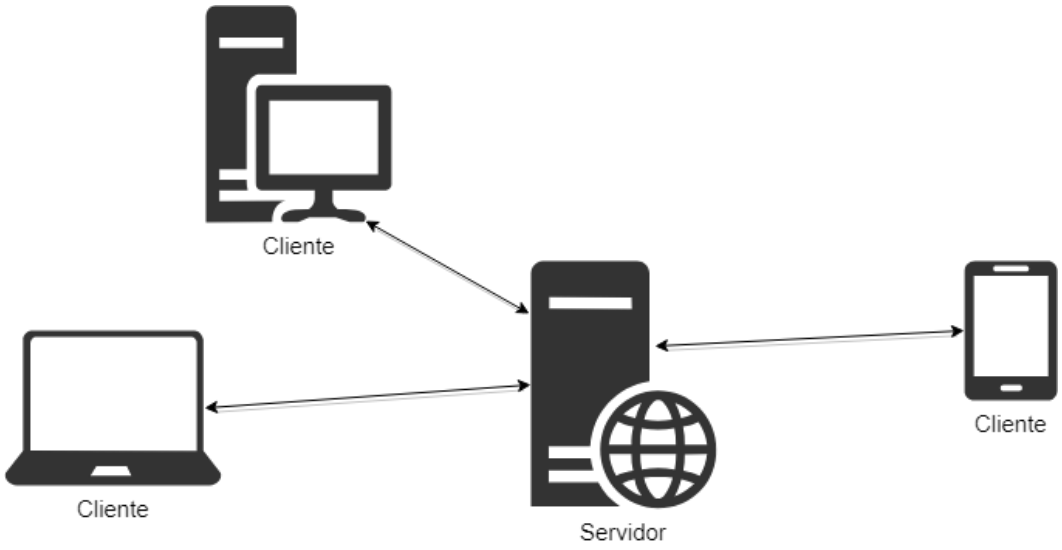


Figura 2.1: Arquitectura física Cliente-Servidor.

## 2.3. API RESTful

Una API es una interfaz de comunicación que una aplicación ofrece para ser usado por otro software, la cual consiste en un conjunto de subrutinas, funciones y procedimientos [19]. REST es un estilo de arquitectura usado para comunicar sin estado aplicaciones cliente-servidor, que suele transferir datos en formato JSON. Las APIs RESTful (APIs que hacen uso de REST) son adecuadas para el uso en internet debido a su bajo uso de ancho de banda. Son usadas especialmente en el caso de aplicaciones cliente-servidor porque disminuyen el acoplamiento. Se utilizan conectándose mediante una dirección única URI, y constan de varias operaciones, donde las más usadas son cuatro: GET, POST, PUT y DELETE [56, 52].

## 2.4. Autenticación por Json Web Token (JWT)

La autenticación por JSON Web Tokens, corresponde a un tipo de autenticación sin estado, es decir, donde no se mantiene información de sesión/estado, sino que los pares solicitud-respuesta son independientes.

Para comprender JWT es necesario saber previamente qué es un token. Un token es simplemente una cadena alfanúmerica generada en un sistema, para el cual presenta algún significado. Suelen generarse a partir de algún algoritmo de hashing.

Los JSON Web Tokens son un tipo de token con una estructura particular utilizada para la autenticación de aplicaciones web. Estos están formados por tres secciones separadas por un punto.

Las secciones identificadas son:

1. Header: Incluye el tipo de token y el algoritmo utilizado.
2. Payload: Contiene los datos que identifican al usuario, y datos de validación del token.
3. Signature: Firma digital creada a partir de las dos secciones anteriores, por medio de un algoritmo de encriptación. Se utiliza para verificar la autenticidad del token.

El flujo de datos con el cual funciona JWT es el siguiente (ver Figura 2.2):

1. El usuario envía sus datos en un *login* tradicional.
2. El servidor valida los datos y retorna un token.
3. El cliente guarda el token para incluirlo en las siguientes peticiones.
4. El cliente realiza las peticiones enviando el token en el header.
5. El servidor valida el token y responde la solicitud [7, 20].

## 2.5. Tecnologías utilizadas

En esta sección se describen las tecnologías utilizadas, mas no los fundamentos para dicha elección, los cuales pueden verse en el capítulo 5 (Arquitectura y diseño de software).

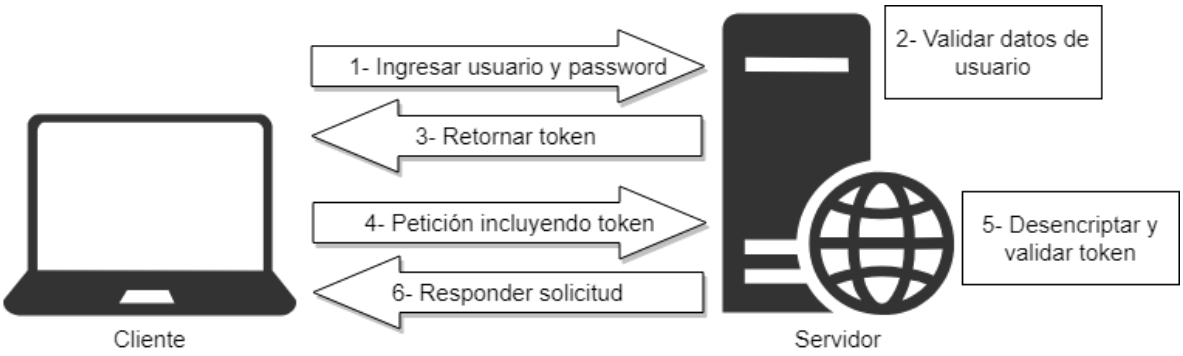


Figura 2.2: Flujo de datos con JWT.

### 2.5.1. MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos, de código libre y gratuito. Entre sus características se encuentran las siguientes:

- Los datos son almacenados en documentos flexibles con formato similar a JSON.
- Es una base de datos distribuída en su núcleo, ofreciendo escalabilidad.
- Eficiente.
- ACID *compliant*. Esto quiere decir que permite realizar transacciones manteniendo los principios de atomicidad, consistencia, aislamiento y durabilidad (ACID por sus iniciales en inglés), los cuales aseguran robustez en una base de datos [48, 50, 16, 25].

### 2.5.2. Spring

Spring es un framework Java (aunque soporta otros lenguajes), que incluye una amplia variedad de subproyectos con distintos objetivos, tanto de propósito general como específicos. Estos subproyectos pueden utilizarse en conjunto o por separado, y pueden determinar distintas arquitecturas [36].

#### Spring WebFlux

Spring WebFlux es un proyecto de Spring, el cual está encargado de dar soporte para aplicaciones web reactivas, entre ellas, servidores web reactivos. Este módulo

incluye dependencias a Spring-core (funcionalidades básicas) y a Spring-web, el que a su vez incluye implementaciones de APIs con JSON y XML (formatos para transferir datos), entre otras características [2, 47].

La base de WebFlux consiste en mantener flujos de datos (streams), los cuales pueden ser suscritos de manera asíncrona y sin bloqueos. Con la ayuda de Spring Data, esta reactividad se extiende hasta la base de datos [28].

#### Spring Data

Spring Data es un proyecto de Spring que facilita el uso de tecnologías de acceso a datos, incluyendo bases de datos relacionales y no relacionales, por medio del uso del patrón repository (capa de persistencia, entre la aplicación y la base de datos). Está compuesto de varios subproyectos, entre los que se encuentra Spring Data MongoDB [38, 1]. Spring Data MongoDB provee una abstracción a alto nivel para el uso de bases de datos MongoDB, incluyendo soporte en modo reactivo [39].

#### Spring Security

Spring Security es un framework altamente configurable de autenticación y control de acceso, con protección frente a ataques informáticos comunes [40]. Este framework soporta y facilita tanto autenticación *stateful* como *stateless* [41], esto es, manteniendo la información de la conexión en un estado o enviándola dentro de los mismos paquetes, respectivamente [9].

### 2.5.3. Angular

Angular es un framework JavaScript creado por Google utilizado en frontend, gratuito y de código abierto. Es sucesor del framework AngularJS (aunque no compatibles), pero a diferencia de este último, utiliza el lenguaje Typescript, que es un super-conjunto de JavaScript/ECMAScript.

Entre sus características se encuentran:

- Es ideal para la construcción de aplicaciones SPA.
- Promueve una arquitectura modular consistente que facilita la integración de nuevos desarrolladores a un proyecto.

- Debido a la modularización, la reutilización de componentes se vuelve esencial sin esfuerzos extras.
- Gracias al uso de TypeScript se pueden detectar errores tempranamente.
- Es un framework y no una biblioteca, por lo que entrega un mejor soporte para la construcción de aplicaciones [22].

A continuación se puede observar la estructura básica de una aplicación en angular (Figura 2.3).

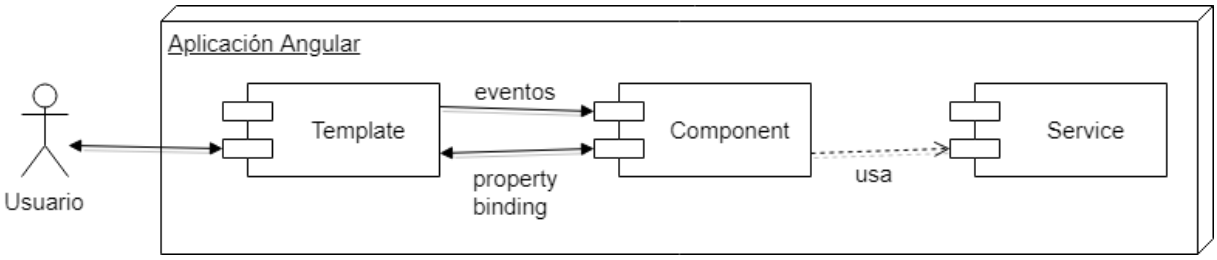


Figura 2.3: Arquitectura básica angular.

## 2.6. Metodología de desarrollo Scrum

Scrum es una metodología ágil de desarrollo de software. En consecuencia, pone énfasis en realizar entregas de valor constantemente, ser adaptable, y funcionar de manera colaborativa [24, 27].

Scrum funciona de manera iterativa en periodos cortos de tiempo (una a cuatro semanas), tras las cuales debe realizarse una **entrega** potencialmente definitiva, es decir, un producto parcial terminado. Estos periodos de tiempo iterativos son denominados **Sprints**.

Para llevar a cabo los sprints, es necesario tener definidos dos documentos. El primero contiene las historias de usuario, que son descripciones genéricas de los requerimientos del proyecto. Este documento es llamado **product backlog**. El segundo, creado a partir del product backlog, describe las tareas a realizar en el sprint, que permitan cumplir con un subconjunto de las características mencionadas en el product backlog. Este último es llamado **sprint backlog**.

Existen cuatro eventos o reuniones que deben realizarse al aplicar scrum:

- Planificación del sprint: Se realiza al comenzar un sprint, y en ella se define el sprint backlog.
- Reunión diaria (daily scrum o daily meeting): Reunión corta realizada a diario durante la ejecución del sprint, en la que cada integrante explica rápidamente qué hizo el día anterior, qué hará este día, y qué problemas ha tenido.
- Revisión de sprint o demostración: Se realiza al concluir un sprint, y antes de la retrospectiva. En ella se presenta el trabajo del sprint, y se define qué se completó y qué quedó pendiente.
- Retrospectiva del sprint: Esta reunión marca el final del sprint, y consiste en una dinámica de valoración del trabajo realizado. En ella deben identificarse los aspectos positivos, mejorables, y nuevas consideraciones por intentar [33].

En la Figura 2.4 se muestra el ciclo de vida de un proyecto utilizando la metodología Scrum.

Por último, para comprender esta metodología es necesario conocer los distintos roles definidos:

- Desarrolladores: Son los encargados de ejecutar las tareas del sprint. Se organizan en equipos de tres a nueve personas, los cuales deben tener las capacidades para entregar partes del producto terminadas, por lo que no suelen haber equipos de especialización.
- Scrum Master: Cada equipo de desarrollo tiene un Scrum Master, el cual puede ser desarrollador o no serlo. Su función es velar por el cumplimiento de la metodología y proporcionar soluciones a los impedimentos en los desarrolladores. Suele ser la voz del equipo de desarrollo y comunicarse con otros scrum master y el product owner. Adicionalmene, puede cumplir el papel de mentor.

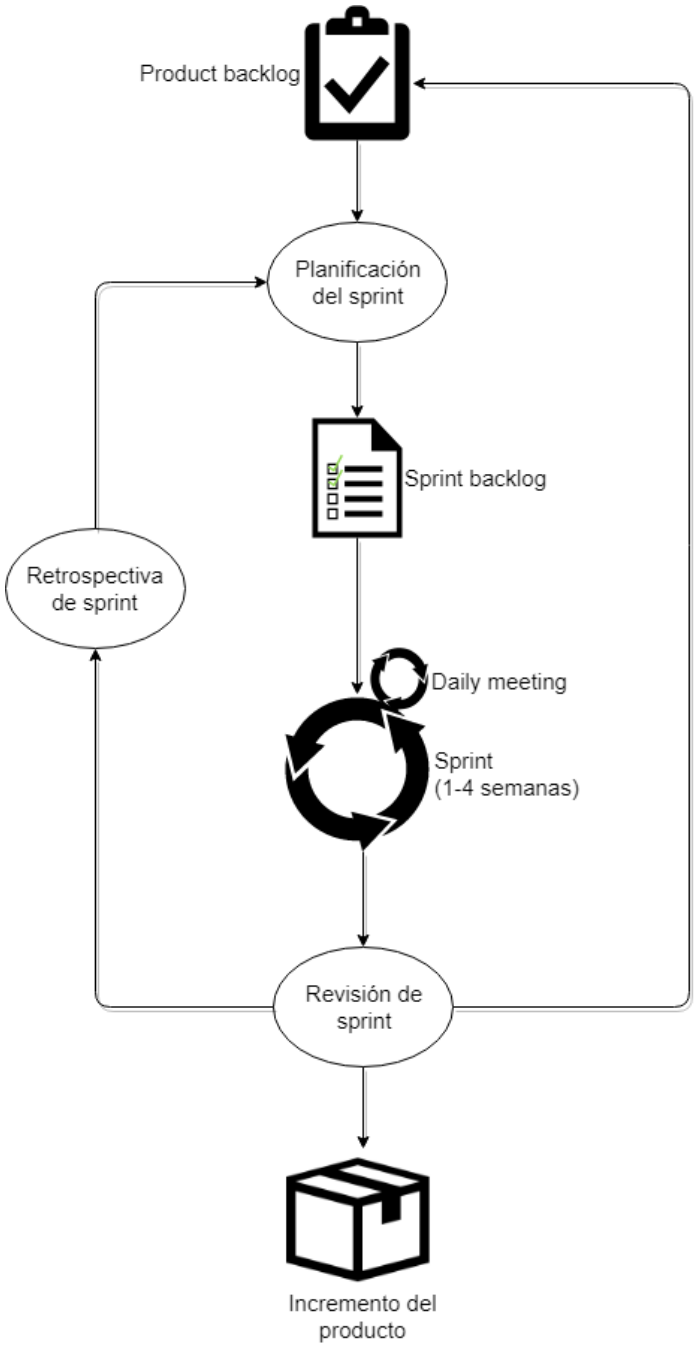


Figura 2.4: Ciclo de vida de Scrum.

- Product Owner: Es el encargado de representar los intereses de los distintos stakeholders dentro del equipo, con el objetivo detallar, priorizar y clarificar los requerimientos. Define hacia donde deben enfocarse los esfuerzos, y tiene

la última palabra en las decisiones del proyecto. El product owner define el product backlog y el sprint backlog (este último junto a cada equipo de desarrollo) [34].

Es importante destacar que la estructura mencionada no es estricta, ya que las metodologías ágiles son adaptables por definición.

## 2.7. Método Kanban

Kanban es un marco de trabajo que permite organizar tareas de manera visual, de acuerdo a un flujo de trabajo previamente definido. Originario de Japón, Kanban consiste en un tablero dividido en columnas, las cuales representan estados de tareas a lo largo de un proceso. Un tablero básico estaría compuesto de tres columnas: "por hacer", "en proceso" y "hecho". Las tareas se representan mediante tarjetas, las cuales son agregadas a la primera columna, para luego ser desplazadas según corresponda a su estado real [55, 29, 59]. En la Figura 2.5 puede verse un ejemplo de un tablero básico Kanban.

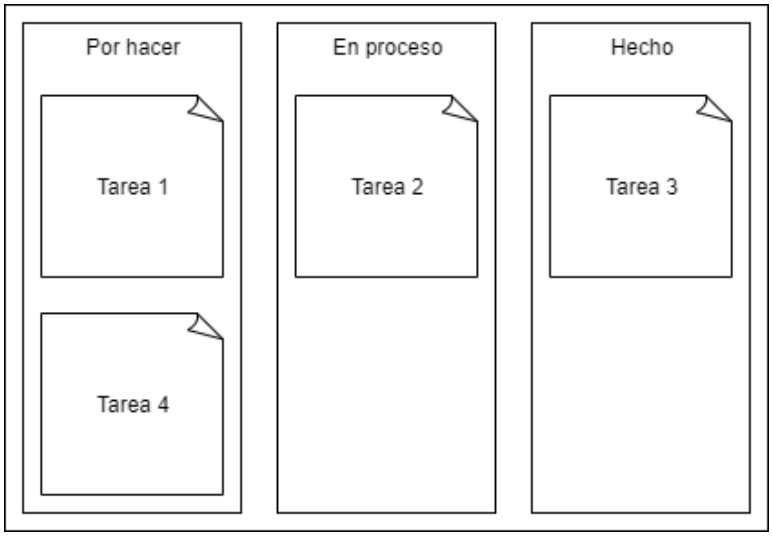


Figura 2.5: Tablero Kanban básico.

Entre los beneficios provenientes del uso de Kanban encontramos los siguientes:

- **Visualización:** La visualización es fundamental para una buena gestión, y Kanban permite visualizar de manera simple el flujo de trabajo, permitiendo así identificar fortalezas y debilidades, cuellos de botella, etc.

- **Enfoque de trabajo:** Mediante la limitación de tareas realizándose por cada miembro del equipo, se reducen las pérdidas de tiempo asociadas al *multitasking*, mejorando así su eficiencia.
- **Ahorro de tiempo:** Con la gestión de un flujo de trabajo continuo mediante el método Kanban, el tiempo asociado a reuniones o informes de progreso se verá disminuido, puesto que dicha información se encontrará disponible (hasta el nivel de detalle que se defina).
- **Definición de procesos:** Al utilizar Kanban, debe estar claramente definido cuando comienza un proceso, y cuando se da por terminado. Por ende, resulta clarificador para el equipo de trabajo, además de permitir identificar mejor responsabilidades.

## 2.8. Metodología de evaluación

Una aplicación puede ser evaluada de distintas maneras (no excluyentes), las cuales permiten verificar y validar el funcionamiento del sistema. A continuación se presentan algunas de estas pruebas. Para ver el detalle de las pruebas escogidas y formuladas ir al Capítulo 7 (Verificación y evaluación).

### 2.8.1. Pruebas unitarias

Las pruebas unitarias o *unit test* son métodos que permiten verificar de forma aislada el funcionamiento de unidades de código atingentes a una funcionalidad en específico [53].

Entre sus características se encuentran:

- Son automatizadas.
- Distintas ejecuciones debieran entregar el mismo resultado [31].

#### Pruebas de caja negra y caja blanca

Al verificar el funcionamiento de unidades de código, se pueden realizar tanto pruebas de caja negra como de caja blanca. Las pruebas de caja blanca se utilizan para evaluar la estructura del código, mientras que las pruebas de caja negra evalúan

el resultado obtenido tras aplicar una función sobre una entrada, sin importar cómo se llegó a ese resultado [49].

### 2.8.2. Pruebas de integración

Las pruebas de integración corresponden a pruebas que verifican el funcionamiento de los distintos componentes actuando en conjunto. En otras palabras, se prueban características individuales, pero sin aislar los componentes [53].

### 2.8.3. Pruebas de sistema

Las pruebas de sistema son el siguiente eslabón tras las pruebas de integración, y corresponden a pruebas que se realizan sobre el sistema completo e integrado, es decir, ya no se evalúan ni componentes aislados ni módulos parciales, sino el sistema funcional completo. En este tipo de pruebas, se puede evaluar tanto funcionalidad como rendimiento [53].

### 2.8.4. Evaluación de usabilidad

Las pruebas de usabilidad corresponden a validaciones por parte de potenciales usuarios sobre las características desarrolladas. El objetivo de estas pruebas es comprender, evaluar y mejorar la experiencia del usuario en la aplicación [30].

Pueden encontrarse dos tipos:

- **Cualitativas**: Son pruebas en que se privilegia la calidad de la información obtenida por sobre el espacio muestral. Suelen consistir en entrevistas u observación del usuario.
- **Cuantitativas**: Son pruebas realizadas a un espacio muestral mayor, al cual se aplican preguntas con respuestas estandarizadas (como encuestas), para obtener datos estadísticos [44].

#### Concurrent Think Aloud (CTA)

La metodología *Concurrent Think Aloud* consiste en que los usuarios interactúen con el sistema, a la vez que verbalizan sus pensamientos. De esta manera pueden identificarse en tiempo real los distintos defectos presentes en la experiencia de usuario [66, 63]. Corresponde a pruebas de tipo cualitativas.

### Retrospective Probing (RP)

La metodología *Retrospective Probing* consiste en realizar una serie de preguntas a los usuarios respecto a su apreciación de un sistema, tras haber realizado ciertas tareas asignadas. Las preguntas pueden ser tanto abiertas como cerradas, por lo que este tipo de prueba puede ser tanto cualitativa como cuantitativa [63]. Para la utilización de RP de forma cuantitativa se pueden utilizar escalas **Likert**, las cuales consisten en rangos de respuestas equivalentes a conceptos, principalmente para mostrar concordancia o discordancia frente a una declaración [60].

### Pruebas SUS

La Escala de Usabilidad de Sistemas (en inglés **SUS**) corresponde a una prueba de usabilidad desarrollada en 1986 por John Brooke, la cual consiste en 10 enunciados tipo likert, en que el usuario debe señalar el grado de acuerdo o desacuerdo en una escala de 1 a 5 [61].

Los enunciados son los siguientes:

1. Pienso que me gustaría usar esta aplicación con frecuencia.
2. Encuentro la aplicación innecesariamente compleja.
3. La aplicación me pareció fácil de usar.
4. Pienso que necesitaría de soporte técnico para poder utilizar la aplicación.
5. Me pareció que las funcionalidades de la aplicación estaban bien integradas.
6. Pienso que hay demasiadas inconsistencias en esta aplicación.
7. Creo que la mayoría de la gente podría aprender a usar la aplicación rápidamente.
8. Encontré la aplicación muy incómoda de usar.
9. Me sentí confiado usando la aplicación.
10. Tuve que aprender muchas cosas para poder usar la aplicación.

Cabe mencionar que los enunciados se han adaptado verbalmente para adaptarse al lenguaje y así facilitar el entendimiento de los mismos, procurando mantener el significado para así no afectar la confiabilidad [64].

Para calcular el puntaje de usabilidad debe puntuarse las preguntas positivas (1, 3, 5, 7 y 9) como el valor asignado menos 1. Para las preguntas negativas (2, 4, 6, 8 y 10) el puntaje será 5 menos el valor respondido. Luego se multiplica la suma de los puntajes por 2,5. De esta manera se obtendrá un valor entre 0 y 100.

## 2.9. Resumen del capítulo

Para la comprensión del documento es ideal contar con una base técnica sobre conceptos de ingeniería de software y desarrollo web. En este capítulo se describen tópicos de arquitectura y diseño, como arquitectura cliente-servidor, aplicaciones web, API RESTful y autenticación por JWT; tecnologías como los frameworks Angular, Spring y el sistema MongoDB; metodologías, métodos y técnicas, como Scrum, kanban, pruebas automatizadas y sus tipos, además de evaluaciones tipo likert, como SUS.

# 3. Metodología de trabajo

En este capítulo se describe la metodología de trabajo a utilizar durante el desarrollo del proyecto. Debido a las condiciones del proyecto, se definirá una metodología de análisis y diseño, una metodología de desarrollo y una metodología de evaluación.

## 3.1. Metodología de análisis y diseño

En este proyecto existe una ingente cantidad de posibles características, además de no contar con usuarios finales predefinidos, por lo cual resulta muy necesario el establecer una metodología de análisis y diseño. Esta metodología debe permitir acotar el alcance de manera apropiada, a la vez que otorga flexibilidad suficiente para modificaciones. Además, debe hacer converger las necesidades de posibles interesados con distintas características.

Es así que se define una metodología basada en la realización de un estudio de mercado y encuestas. El estudio de mercado está acotado al *microentorno*, analizando en particular competidores y posibles usuarios, tomando en consideración los comentarios realizados en aplicaciones similares. En base a lo anterior, las encuestas permiten priorizar las funcionalidades definidas, determinar algunas nuevas, esclarecer requisitos no funcionales y definir un *buyer persona* o arquetipo de usuario ideal. La existencia de este *buyer persona*, si bien en este caso no guía el proyecto completo, permite tomar decisiones de diseño de interfaz con mayor facilidad.

Adicionalmente, se aplica la técnica de experimentación controlada *Concurrent Think Aloud* (CTA), la cual es realizada durante el desarrollo, cada dos *sprints*, para obtener retroalimentación temprana, con unos pocos potenciales usuarios seleccionados. De esta manera los requerimientos y el diseño de la aplicación permiten dirigir los esfuerzos hacia una solución acertada.

## 3.2. Metodología de desarrollo

La metodología de desarrollo a utilizar, debe escogerse siempre considerando la naturaleza del proyecto. En este caso, la plataforma a implementar corresponde a una red social, por lo tanto, se espera que los requerimientos sean variables a lo largo de todo el proceso de desarrollo, puesto que debe guiarse por las necesidades de la comunidad. En razón de lo anterior, resulta de menor importancia el mantener procesos y documentación estrictos, frente a la capacidad de adaptación y la entrega continua de software funcionando. Esto sugiere el uso de una metodología ágil.

Se escoge *Scrum* por su enfoque en la entrega continua de valor. Esto dado a que, pese a no haber un tercero como cliente, sí existen muchos potenciales usuarios con los que se puede validar o mejorar tempranamente las distintas características a implementar, especialmente en cuanto a las interfaces humano-computador. No obstante, esta metodología debe ser modificada por no existir un equipo de desarrollo, sino un único individuo que estará a cargo de toda su ejecución. Debido a esto, se eliminarán algunas reuniones, manteniendo la revisión de cada sprint y la retrospectiva correspondiente, en conjunto con revisiones semanales con el profesor guía, en las que se verificará el avance para redirigir los esfuerzos de trabajo en caso de ser necesario.

La duración de los sprints se fija en tres semanas. Terminando cada uno de ellos se tendrán nuevas características implementadas, las cuales debieran funcionar al punto de poder ser sometidas a una validación.

Para la organización del trabajo, se utiliza el método *Kanban*, con el cual se facilita visualmente la trazabilidad del proceso, permitiendo reaccionar a tiempo en caso de algún percance, además de facilitar la priorización de las distintas tareas. En esta ocasión se divide el tablero (Figura 3.1) en cuatro columnas:

- **Product Backlog**: Se corresponde con el Product Backlog de *Scrum*, es decir, contiene las historias de usuario del proyecto. Las tarjetas de esta columna permanecen fijas.
- **Sprint Backlog**: Contiene las tareas a realizar durante el *sprint*, las cuales se corresponden con una o más historias de usuario.
- **In progress**: Contiene las tareas provenientes de *Sprint Backlog*, cuyo desarrollo ya haya sido comenzado.

- **Done**: Tareas provenientes de *In progress*, que han sido terminadas y probadas.

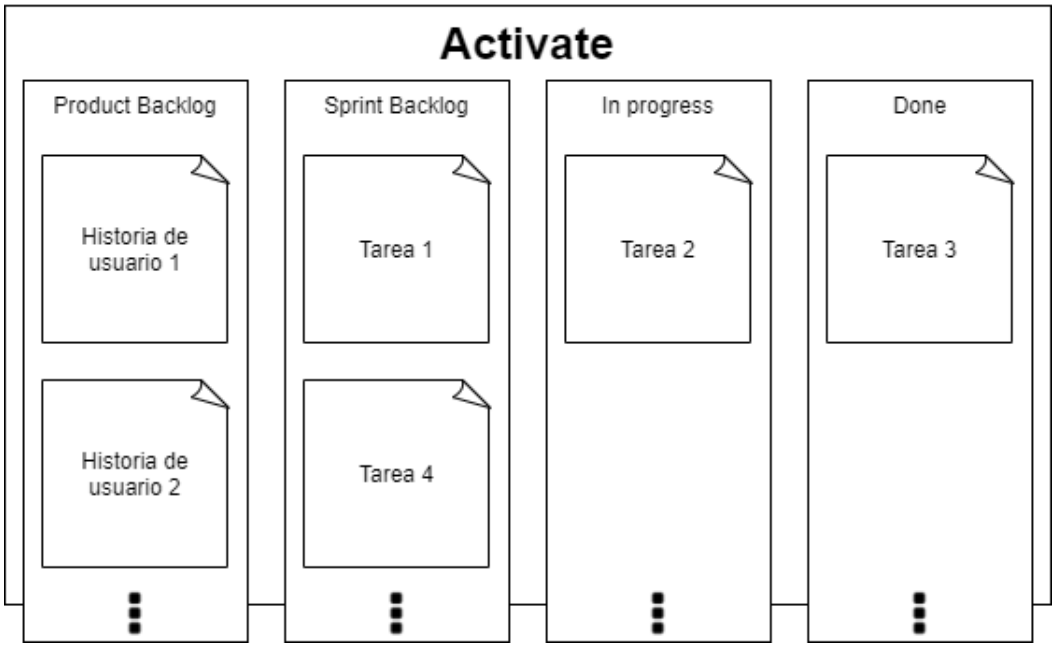


Figura 3.1: Tablero Kanban utilizado.

Para priorizar el desarrollo de los requerimientos capturados se ha optado por utilizar un valor de importancia y un valor de urgencia asociados a cada requisito, los cuales se actualizan durante el desarrollo en caso de resultar adecuado. Sus valores iniciales pueden observarse en la Sección 4.2.

Adicionalmente, y considerando que no existe un *velocity* conocido (cantidad de trabajo abarcable en un sprint), se define la siguiente estrategia para priorizar las tareas asociadas a un mismo caso de uso:

1. programar backend (entidades y *findAll*);
2. agregar datos a la base de datos con un script;
3. mostrar datos en el frontend;
4. agregar acciones sobre los datos; y
5. mejorar UI/UX.

De esta manera, se busca facilitar la verificación durante el desarrollo, permitiendo así el uso de *TDD*.

Finalmente, resulta oportuno decir que para evitar posibles incidentes de fuerza mayor, además de optimizar el desarrollo, es que se utilizan herramientas que permitan el trabajo de forma local a la vez que se cuenta con un respaldo en línea. Por ejemplo, se utiliza la herramienta de versionamiento *Git*, junto a la plataforma *GitLab*.

## 3.3. Metodología de evaluación

El producto desarrollado es evaluado tanto en su funcionalidad (verificación) como en su usabilidad (validación).

Para evaluar la correctitud de las funcionalidades, se han definido pruebas de caja negra para el servidor, las cuales han sido implementadas en forma de tests unitarios y tests de integración en las capas que se espera mayor probabilidad de fallos debido a las funciones que realizan.

Por razones de tiempo, no se realizan pruebas unitarias en la aplicación cliente. Sin embargo es verificada mediante las pruebas de sistema.

Para evaluar la usabilidad se hace uso de la técnica de experimentación controlada Retrospective Probing (RP). La técnica RP se ha utilizado con el MVP finalizado. Por limitaciones de recursos, estas pruebas se realizan con la aplicación ejecutándose en una red local, y no en un servidor abierto a Internet como sería ideal.

Para esta evaluación se selecciona una cantidad pequeña de personas pertenecientes al público objetivo, a los que se les solicita la realización de acciones específicas y luego la entrega de una retroalimentación mediante la completación de un formulario *SUS*.

Paralelamente se realiza otra encuesta basada en *Likert*, la cual busca conocer el estado de cumplimiento de los objetivos según la percepción de los usuarios. Esto debido a que para la intentar medirlo de manera cuantitativa, se requerirían herramientas de análisis avanzadas, además de un conocimiento mayor de los usuarios y los procedimientos que realizaba previamente.

Dadas las limitaciones de recursos, la aplicación no es sometida a pruebas de rendimiento.

## 3.4. Resumen del capítulo

Para el desarrollo del proyecto se define una metodología de trabajo que consiste a su vez en tres metodologías:

- **Metodología de análisis y diseño:** Investigar el microentorno y realizar encuestas a potenciales usuarios para definir requisitos, y luego refinarlos durante el desarrollo mediante la técnica *Concurrent Think Aloud*.
- **Metodología de desarrollo:** Uso de *Scrum* adaptado a una persona, ordenado mediante *Kanban*, y guiado por *TDD*.
- **Metodología de evaluación:** Verificación parcial mediante pruebas unitarias, de integración y de sistema. Validación de usabilidad mediante formulario *SUS*. Validación de cumplimiento de objetivos mediante encuesta tipo *Likert*.

# 4. Análisis y requisitos

En este capítulo se da a conocer un análisis del contexto desde el cual se obtienen los distintos requisitos del sistema. Junto a lo anterior se exponen los requisitos extraídos y plasmados en el product backlog, los cuales representan la totalidad de funciones deseadas identificadas, inclusive aquellas que no resulten factibles debido a los recursos disponibles para el proyecto.

## 4.1. Diagnóstico del microentorno

El microentorno corresponde al entorno competitivo donde una empresa –o en este caso la aplicación– actúa, es decir, las empresas que ofrecen los mismos productos o servicios. Dado que este proyecto no persigue fines comerciales actualmente, se hace énfasis en analizar a los posibles usuarios y competidores, y no así en los costos de una puesta en marcha real.

### 4.1.1. Competidores

Como se mencionó en el Capítulo 1.3, para los fines buscados pueden utilizarse tanto redes sociales genéricas como aplicaciones específicas de fitness. Solo se analizan estás últimas, y en particular aquellas que compartan funcionalidades similares a lo buscado, obviando así la competencia indirecta. La cantidad de aplicaciones disponibles respecto al fitness es ingente, por lo que resulta necesario seleccionar solo algunas de ellas. Es así que se eligió aplicaciones bien posicionadas en buscadores mediante palabras clave, las cuales fueron filtradas según si atendían las necesidades que se busca cubrir, revisando finalmente que cuenten con una buena evaluación. Estas aplicaciones fueron principalmente móviles; debido a esto y a los recursos disponibles,

se revisó aquellas que funcionaran al menos bajo el sistema operativo *Android*. Finalmente, se compararon las características de siete aplicaciones, cuyo detalle puede verse en el Cuadro 4.1, junto a una propuesta para este proyecto. Dichas aplicaciones son:

- **8fit**: Permite hacer uso de rutinas y recetas prediseñadas, incluyendo recomendaciones. Algunas funcionalidades son de pago. Destaca por su interfaz. [5].
- **Street Workouts Calisthenics: Trainer Fitness**: Ofrece un amplio abanico de características tanto para ejercicios como para nutrición. Sin embargo se detectan algunos errores y la interfaz de usuario es muy mejorable [42].
- **JEFIT**: Incluye todas las características deseables para ejercitación y permite interacción social [21].
- **Thenics**: Contiene rutinas de ejercicios con información útil, entregada en una interfaz sencilla y amigable [46].
- **TechNutri**: Red social especializada en nutrición con dietas de pago. Contiene planes de ejercicio predefinidos para usuaros premium (de pago). Partes de la aplicación están solo en portugués. [43].
- **Ejercicios en Casa**: Ofrece planes de alimentación y rutinas de ejercicios, ambos prediseñados. [12].
- **Madbarz**: Red social que incluye ejercicios y alimentación, con funcionalidades de pago. [23].

| **Características** | **8fit** | **SWC** | **JEFIT** | **Th** | **TN** | **EeC** | **MB** | **Prop** |
|---|---|---|---|---|---|---|---|---|
| Contiene información de ejercicios/musculatura | X | X | X | X | | X | X | X |
| Contiene ejercicios sin implementos | X | X | X | X | X | X | X | X |
| Contiene ejercicios con implementos | X | X | X | | | | X | X |

Cuadro 4.1: Características aplicaciones similares.

| **Características** | **8fit** | **SWC** | **JEFIT** | **Th** | **TN** | **EeC** | **MB** | **Prop** |
|---|---|---|---|---|---|---|---|---|
| Presenta ejercicios alternativos | X | | X | | | | | X |
| Contiene rutinas de ejercicios | X | X | X | X | X | X | X | X |
| Realizar rutinas de ejercicios | X | X | X | X | X | X | X | X |
| Crear rutinas de ejercicios | | X[1] | X | | | | X | X |
| Contiene planes de ejercicios | X | | X | | X | | X | X |
| Crear planes de ejercicios | | X[2] | X | | | | | X |
| Contiene información nutricional de alimentos | X | X | | | X | | X | X |
| Contiene recetas | X | | | | X | X | X | X |
| Crear recetas | | | | | X | | | X |
| Contiene planes alimenticios | X | X | | | X | X | | X |
| Crear planes alimenticios | | | | | | | | X |
| Crear planes de alimentación y ejercicios | | | | | | | | X |
| Ofrece sugerencias de acuerdo a objetivos u otros | X | | X | | | X | X | |
| Historial de estado físico | X | X | X | X | X | X | X | X |
| Permite interacción social mediante publicaciones | | | X | | X | | X | X |
| Contiene funcionalidades de pago | X | X | | | X | | X | |
| Publicidad invasiva | | X | X | | X | | | |

Cuadro 4.1: Características aplicaciones similares.

[1]Permite agregar una rutina para cada día de la semana.
[2]Permite definir una planificación semanal única.

### 4.1.2. Usuarios

Para el análisis de usuarios se consideran cifras de deportistas sean o no actualmente usuarios de aplicaciones. Según el reporte global del 2018 de la IHRSA (International Health, Racquet & Sportsclub Association), Chile cuenta con alrededor de 490.000 miembros de distintos gimnasios [58]. Por otro lado, en las aplicaciones presentadas en el punto anterior encontramos que aquella con menor cantidad de descargas, cuenta con más de 500 mil (ver Cuadro 4.2).

| **Aplicación** | **Descargas** |
|---|---|
| 8fit | 10.000.000+ |
| SWC:TF | 500.000+ |
| JEFIT | 5.000.000+ |
| Thenics | 1.000.000+ |
| Technutri | 5.000.000+ |
| Ejercicios en casa | 50.000.000+ |
| Madbarz | 1.000.000+ |

Cuadro 4.2: Descargas de aplicaciones similares.

Adicionalmente, se realizó una encuesta específica para esclarecer los intereses y prioridades, la cual fue respondida por 59 personas. De ella se desprende información que permitirá seleccionar y valorizar los requisitos a incluir en el MVP. Entre esta información destaca:

- La mayoría tienen entre 3 y 6 comidas diarias, con moda en 4.
- A la mayoría le resulta importante el poder acceder a información fácilmente, además de poder llevar un registro de su estado físico.
- El poder agregar actividades distintas a las rutinas tiene menor relevancia que otras funcionalidades.
- Resulta importante tener tanto elementos propios como de alguien con más experiencia.
- Deberá existir planes que incluyan ejercicios y alimentación.

Además, se ha recopilado información de utilidad para determinar la forma de la aplicación y para el trabajo futuro. En concreto, se define el siguiente arquetipo *buyer persona*:

- Edad: 24 años. *Millenial*.
- Actividad: estudiante y trabajador.
- Nivel socioeconómico: clase media.
- Cuida su estado físico personalmente, pero no es experto en el tema.
- Cuenta con equipamiento deportivo limitado.
- Sabe leer una etiqueta nutricional.
- No sigue una dieta estricta, pero es selectivo en términos nutricionales.
- Utiliza redes sociales jóvenes como Instagram.
- Le agrada compartir, pero le preocupa su seguridad en redes sociales.
- Le gustan los desafíos con dificultad progresiva
- A veces requiere motivación externa y una planificación e información prediseñada para hacer una actividad.

Pueden verse más detalles de la encuesta en el Anexo A.

## 4.2. Requisitos del sistema

A continuación se presentan los distintos requisitos obtenidos a partir del análisis anterior y que, en consecuencia, conforman el product backlog.

Estos requisitos se han definido como historias de usuario, las cuales llevan un código asignado. A estas historias se les ha añadido un valor de importancia definido en base a la encuesta inicial; así también un valor de urgencia, asignado de acuerdo a necesidades del desarrollo como prerequisitos. Estos dos valores tienen como objetivo facilitar la priorización de tareas a realizar durante construcción del software. Estos son valores iniciales y pueden variar según el interés de los usuarios en un momento dado. Su rango de valores es de 1 (poca importancia/urgencia) a 5 (mucha importancia/urgencia).

Cabe destacar que estos requisitos exceden el *MVP*, por lo que la implementación de aquellos con menor importancia dependerá de los recursos disponibles.

### 4.2.1. Deportista amateur

| Código | Como deportista amateur quiero... | Imp. | Urg. |
|---|---|---|---|
| DAM1 | Poder administrar mis propias rutinas, para tener un control sobre mis entrenamientos | 4 | 4 |
| DAM2 | Poder revisar/agregar rutinas ajenas, para reutilizar rutinas de alguien con más experiencia | 5 | 4 |
| DAM3 | Poder administrar mis propias recetas, para llevar un control sobre mis comidas | 4 | 4 |
| DAM4 | Poder revisar/agregar recetas ajenas, para reutilizar comidas de alguien con más experiencia | 4 | 3 |
| DAM5 | Poder administrar mis propios planes de entrenamiento, para llevar un control sobre mi desarrollo físico | 4 | 4 |
| DAM6 | Poder revisar planes de entrenamiento ajenos, para así reutilizar entrenamientos de alguien con más experiencia | 5 | 3 |
| DAM7 | Poder seguir a otros deportistas y su trabajo, para motivarme y aprender de ellos y su desarrollo físico | 3 | 2 |
| DAM8 | Poder ejecutar rutinas, para así simplificar el ponerme en forma | 5 | 3 |
| DAM9 | Poder hacer seguimiento de mi avance, para entender mejor mi progreso, aprender y motivarme | 4 | 2 |
| DAM10 | Poder ver el avance de otras personas, para así motivarme y aprender de su desarrollo físico | 3 | 2 |
| DAM11 | Registrar actividades físicas estándar, fuera de las rutinas definidas (como ir a bailar, trotar, etc), para llevar un registro más completo de mis actividades | 3 | 2 |
| DAM12 | Conocer los músculos que estoy trabajando, para organizar mejor mis entrenamientos | 5 | 4 |
| DAM13 | Tener rutinas con distintos objetivos, para organizar mejor mis entrenamientos | 5 | 4 |
| DAM14 | Poder subir fotos junto a mis historias, para así motivarme a mí mismo y a los demás | 2 | 1 |

| Código | Como deportista amateur quiero... | Imp. | Urg. |
|---|---|---|---|
| DAM15 | Saber cuanta fuerza tengo en relación al resto, para ver mejor mi progreso, aprender y motivarme | 1 | 1 |
| DAM16 | Realizar búsquedas de usuarios, rutinas y otros, para encontrar fácilmente lo que necesite | 5 | 4 |
| DAM17 | Administrar mi perfil/usuario, para controlar la seguridad de mis datos | 5 | 2 |
| DAM18 | Recibir sugerencias según mis características, para facilitar el uso de la aplicación a novatos | 3 | 1 |
| DAM19 | Evaluar los distintos elementos, para prevenir lesiones por hacer algo inadecuado según nivel y aprender | 3 | 2 |

Cuadro 4.3: Historias de usuario: Deportista Amateur

### 4.2.2. Entrenador

| Código | Como entrenador quiero... | Imp. | Urg. |
|---|---|---|---|
| ENT1 | Poder tener elementos públicos y privados, para así controlar el acceso a elementos en proceso o terminados | 1 | 1 |
| ENT2 | Poder comunicarme con mis deportistas, desde la misma aplicación. | 2 | 1 |
| ENT3 | Poder crear grupos de deportistas asociados a mi usuario, para proporcionarles elementos de manera diferenciada, de acuerdo a implementación u otros | 1 | 1 |
| ENT4 | Poder asociar elementos a un grupo de deportistas, para proporcionarles elementos de manera diferenciada, de acuerdo a implementación u otros | 1 | 1 |
| ENT5 | Relacionar a mis grupos de deportistas a un perfil especial como un gimnasio o centro deportivo, para proporcionarles elementos de manera diferenciada, de acuerdo a implementación u otros | 2 | 1 |

Cuadro 4.4: Historias de usuario: Entrenador

### 4.2.3. Administrador

| Código | Como administrador quiero... | Imp. | Urg. |
|---|---|---|---|
| ADM1 | Poder administrar ejercicios, para que los usuarios puedan usarlos en sus rutinas | 5 | 5 |
| ADM2 | Poder administrar alimentos, para que los usuarios puedan usarlos en sus recetas | 5 | 5 |
| ADM3 | Poder administrar usuarios, para así moderar el uso adecuado de la aplicación | 4 | 2 |
| ADM4 | Poder administrar elementos de implementación, para que los usuarios puedan usarlos en sus rutinas | 5 | 5 |

Cuadro 4.5: Historias de usuario: Administrador

## 4.3. Resumen del capítulo

En base a una selección de aplicaciones similares se definen los requisitos básicos, los cuales son complementados con una encuesta realizada a 59 personas. Se define un arquetipo de usuario ideal de 24 años, que se autogestiona con recursos limitados.

Los requisitos definidos están separados por tipo de usuario, y con índices de prioridad dado que exceden el MVP.

# 5. Arquitectura y diseño de software

En este capítulo se presentan las distintas decisiones de diseño realizadas junto a su fundamentación (cuando corresponda). En particular se expone la arquitectura de software y el diseño de software utilizado, además del mapa de navegación; elementos necesarios para una implementación correcta y ordenada.

## 5.1. Consideraciones generales

En el Capítulo 4 (Análisis y requisitos), se dieron a entender las funcionalidades a desarrollar. No obstante no se definen formalmente requerimientos no funcionales, se consideran algunos aspectos esenciales de calidad en el software: usabilidad, escalabilidad, seguridad, portabilidad y mantenibilidad. Algunos de estos cobran especial importancia al tratarse una aplicación destinada a un amplio espectro de usuarios.

Para favorecer la portabilidad, el sistema construido corresponde a una aplicación web, puesto que estas pueden ejecutarse sobre múltiples dispositivos y sistemas operativos. Además, dado el ámbito se espera mayoritariamente usos de larga duración y navegación, por sobre varios ingresos frecuentes, junto al uso de lógica en *frontend* para el manejo de planes y rutinas, entre otros. Con esto en consideración y en favor de la experiencia de usuario, resulta una buena opción hacer una aplicación de página única (SPA).

Para favorecer la escalabilidad y teniendo en cuenta que es una SPA, se opta por la arquitectura cliente-servidor con comunicación mediante una API REST, manteniendo así un buen rendimiento.

En pos de la usabilidad resulta conveniente el uso de programación reactiva, puesto que es esperable que una red social funcione de manera transparente, es decir, sin afectar la experiencia frente a situaciones de estrés o errores. Esto podría ser determinante para no perder usuarios.

Por último, dado que en una red social temática se manejan grandes cantidades de datos con múltiples asociaciones y la información no es uniforme, se opta por una base de datos de tipo NoSQL. Así además se facilita la escalabilidad, por ejemplo, permitiendo publicaciones polimórficas provenientes de alguna futura aplicación. Si bien existen diversos tipos de BBDD NoSQL, entre los cuales varios presentan sus propias ventajas para a este proyecto, se escoge el uso de MongoDB por flexibilidad, escalabilidad, eficiencia y afinidad del desarrollador.

En la Figura 5.1 se puede observar la arquitectura base. Más detalles respecto a la arquitectura en la Sección 5.5.

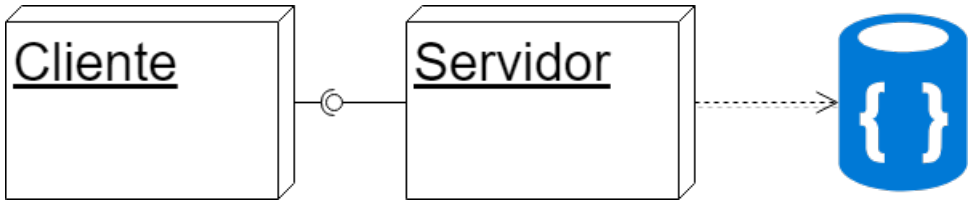


Figura 5.1: Arquitectura básica del sistema.

## 5.2. Mapa de navegación

A continuación se detallan las distintas vistas que tiene la aplicación, su descripción, y la relación entre ellas (Figura 5.2). Se excluyen vistas parciales.

La aplicación cuenta con tres vistas accesibles sin iniciar sesión:

- **Home:** Invita al usuario a formar parte de la comunidad. Incluye links directos a *Iniciar sesión* y *Registro*.
- **Iniciar sesión:** Muestra formulario de inicio de sesión. Cuenta con redirección a *Registro*. Si el inicio de sesión es efectivo se redirige al *Dashboard*.
- **Registro:** Muestra formulario básico de registro. Cuenta con redirección a *Iniciar sesión*. Una vez creadad una cuenta se accede a *bienvenida*, que consiste en una vista inicial que introduce al usuario al sistema.

Una vez iniciada una sesión, se puede acceder a las siguientes vistas desde el menú lateral o superior:

- **Dashboard:** Página principal. Muestra las actividades o posts de usuarios.
- **Perfil:** Perfil de usuario. Incluye su información básica, actividad, seguidores, elementos creados, entre otros.
- **Conectar:** Permite buscar a otros usuarios, y acceder a su perfil o seguirlos.
- **Seguidos/seguidores:** Similar al anterior, pero filtra los resultados según los seguidos o seguidores del usuario consultado.
- **Configuración:** Edición del perfil de usuario.
- **Ayuda:** Información estática de ayuda.
- **Notificaciones:** Listado de notificaciones autogeneradas que redirigen al *elemento* o usuario involucrado.
- **Vista general de *elementos*:** Listados de cada *elemento* (planes, rutinas, recetas, etc.), que incluye a todos los existentes.
- **Mis *elementos*:** Similar al anterior, limitado a aquellos asociados al usuario actual.
- ***Elementos* guardados:** Similar al anterior. Se muestran solo favoritos o guardados.

Asímismo, es posible acceder mediante links a las siguientes vistas:

- **Detalle de *elemento*:** Permite visualizar todo el contenido de un elemento seleccionado. En el caso de un administrador, o ser un elemento propio, permite el paso a edición.
- **Edición de *elemento*:** Contiene la edición básica del *elemento* indicado. Esto incluye textos, imágenes y otros *elementos* asociados.
- **Elaborar rutina:** Permite establecer el contenido de una rutina. Forma parte de una vista distinta a la de edición debido a la cantidad de elementos visuales.

- **Ejecutar rutina:** Permite la ejecución de una rutina de forma mostrando como mínimo el ejercicio a realizar y la cantidad de repeticiones.

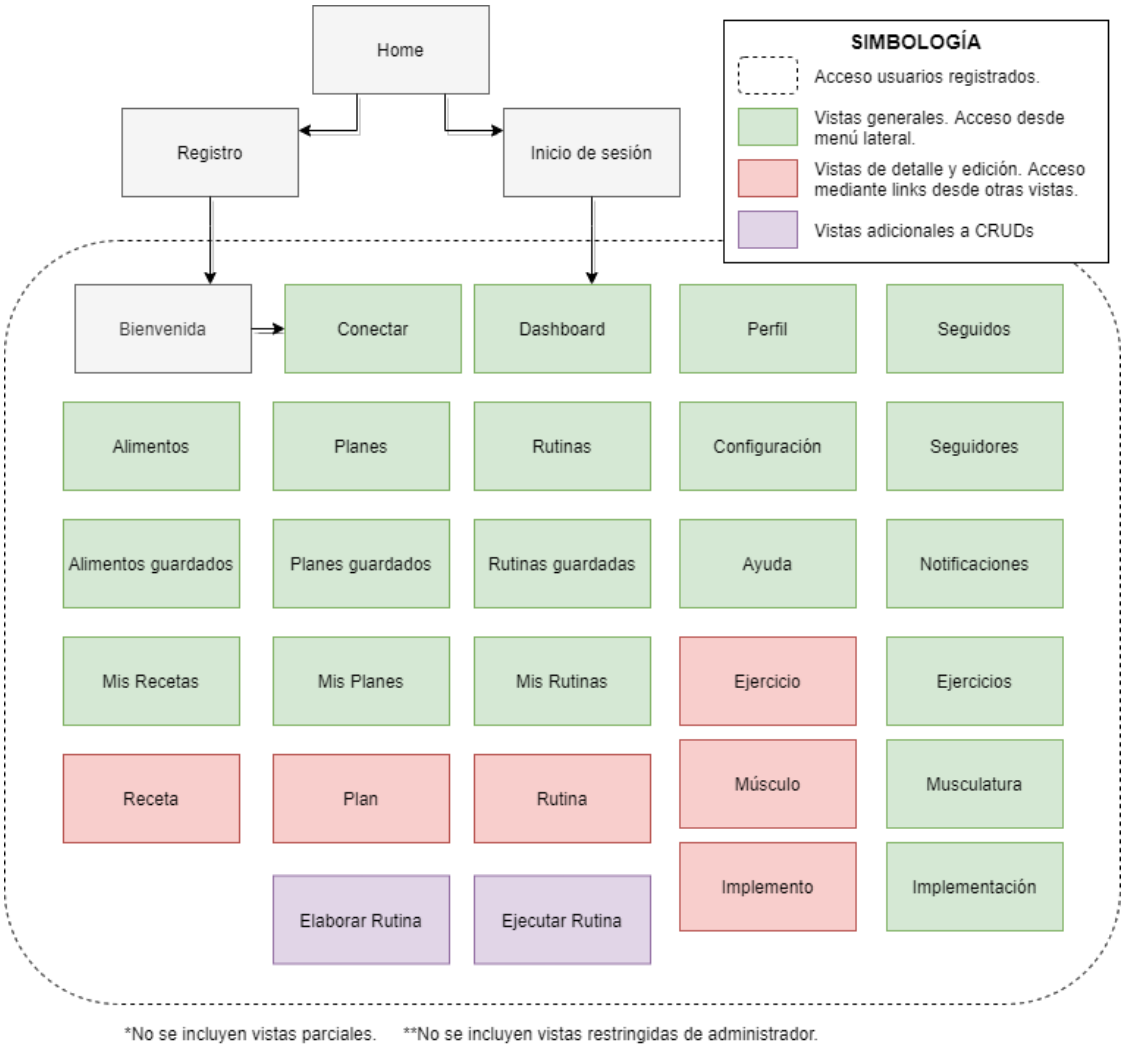


Figura 5.2: Mapa de navegación.

## 5.3. Abstracción del negocio

A continuación se presenta mediante diagrama de clases el modelo utilizado. Los nombres están presentados en español para facilitar su comprensión, no obstante la aplicación esté programada en inglés.

### 5.3.1. Perfiles y usuarios

En el paquete de perfiles se puede observar (Figura 5.3) el uso de dos clases: Usuario y Perfil; las cuales son usadas para autenticación y datos del usuario, respectivamente. Esto con objetivo de favorecer la escalabilidad del sistema; específicamente el caso de autenticación mediante JWT generado por terceros, o el caso de uso eventual de manejar desde una misma cuenta un perfil personal y uno de entrenador o institución.

Además, se cuenta con una lista de seguidores y seguidos.

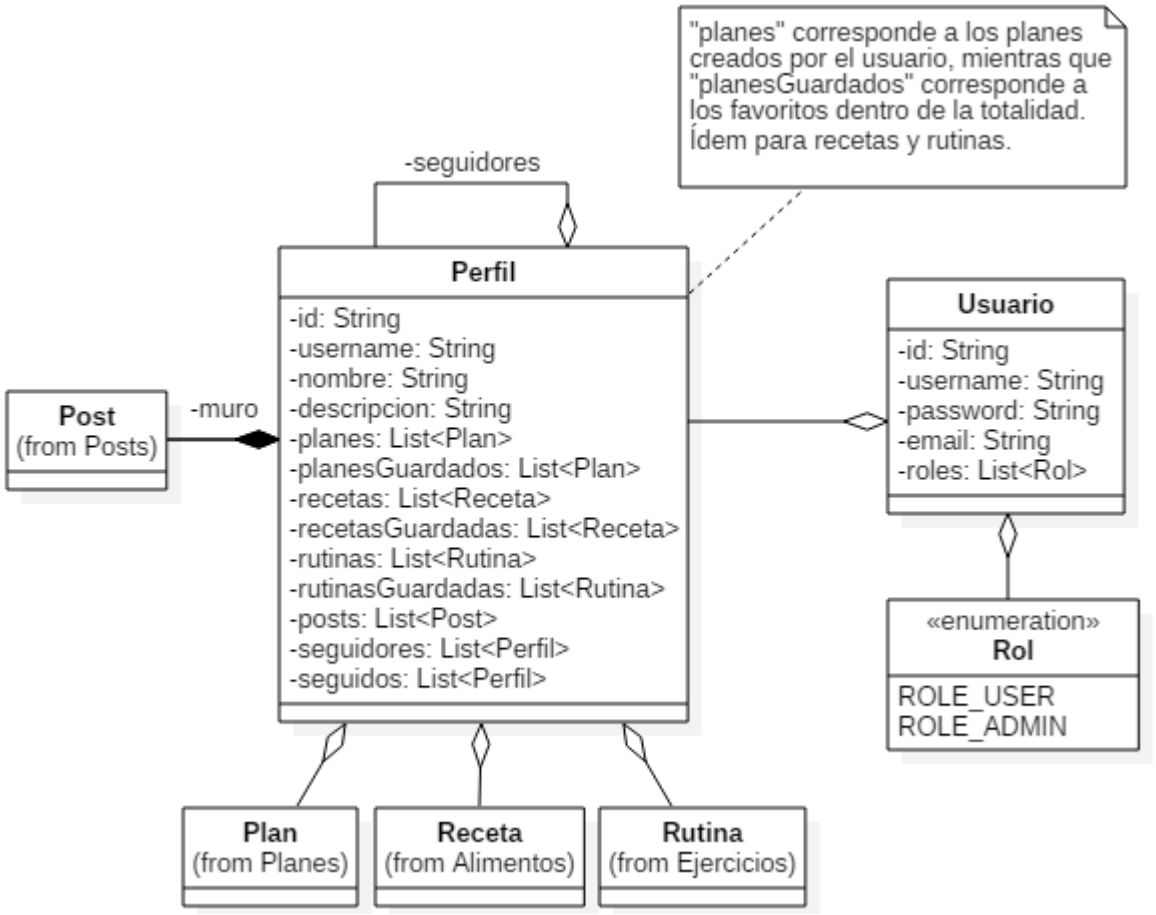


Figura 5.3: Diagrama de clases, paquete Perfiles.

### 5.3.2. Ejercicios

En el paquete referente a ejercicios se puede observar (Figura 5.4) las distintas asociaciones básicas: rutina contiene ejercicios, ejercicio trabaja músculos, entre otras que no requieren mayor explicación. Junto a ellas se presentan dos estructuras *composite* explicadas a continuación. Una rutina está compuesta de subrutinas, que a su vez pueden estar compuestas de otras subrutinas, o de ejercicios, permitiendo así la existencia de estructuras en forma de ciclos. Esto resulta especialmente beneficioso al considerar rutinas tipo tabata u otras que no están basadas en una secuencia lineal de ejercicios. En la Figura 5.5 puede verse esto gráficamente.

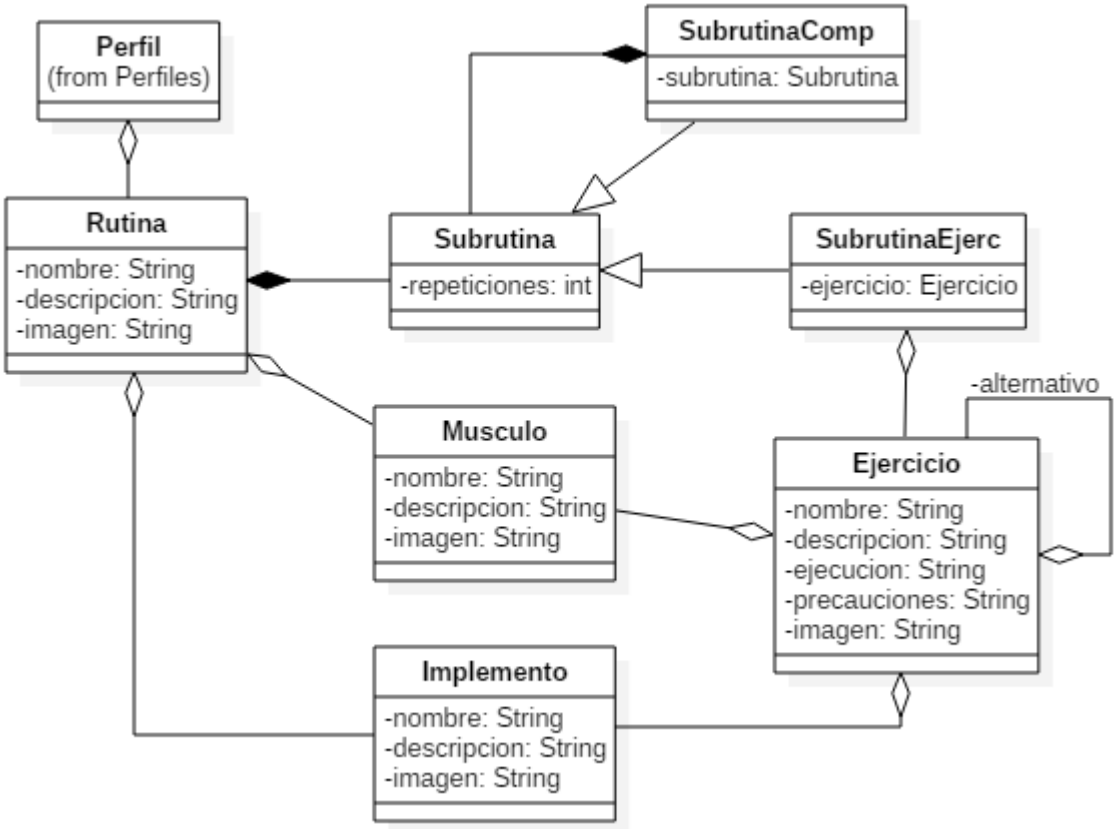


Figura 5.4: Diagrama de clases, paquete Ejercicios.

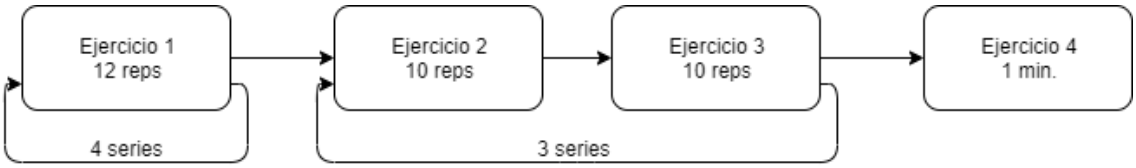


Figura 5.5: Ejemplo rutina de ejercicios.

### 5.3.3. Alimentos

En el paquete referente a alimentación se puede observar (Figura 5.6) la relación existente entre alimentos y recetas. Cabe destacar que solo las recetas cuentan con un autor asociado.

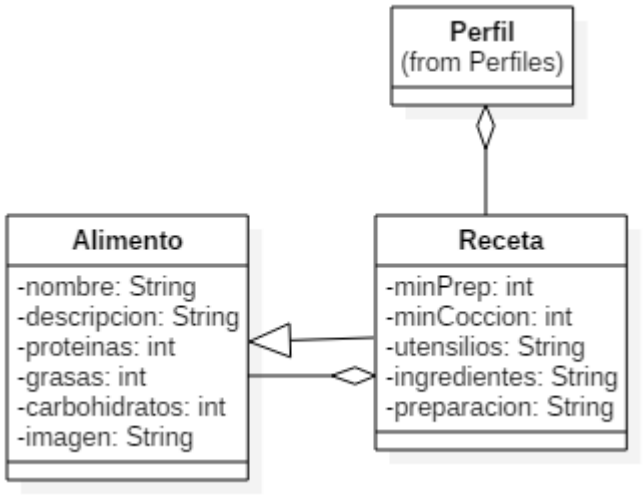


Figura 5.6: Diagrama de clases, paquete Alimentos.

### 5.3.4. Planes

En el paquete referente a planes se puede observar (Figura 5.7) que no hay diferenciación entre planes de ejercicios y de alimentación, sino que hay una generalización que cuenta con los atributos que permitan ambos. De esta manera se entrega flexibilidad a la hora de hacer transformación entre ellos. También puede observarse la existencia de un diccionario *dia*, que permite la agregación dinámica de las distintas comidas y rutinas de un día, además de permitir la modificación de la duración de un plan, todo esto evitando el uso excesivo de recursos.

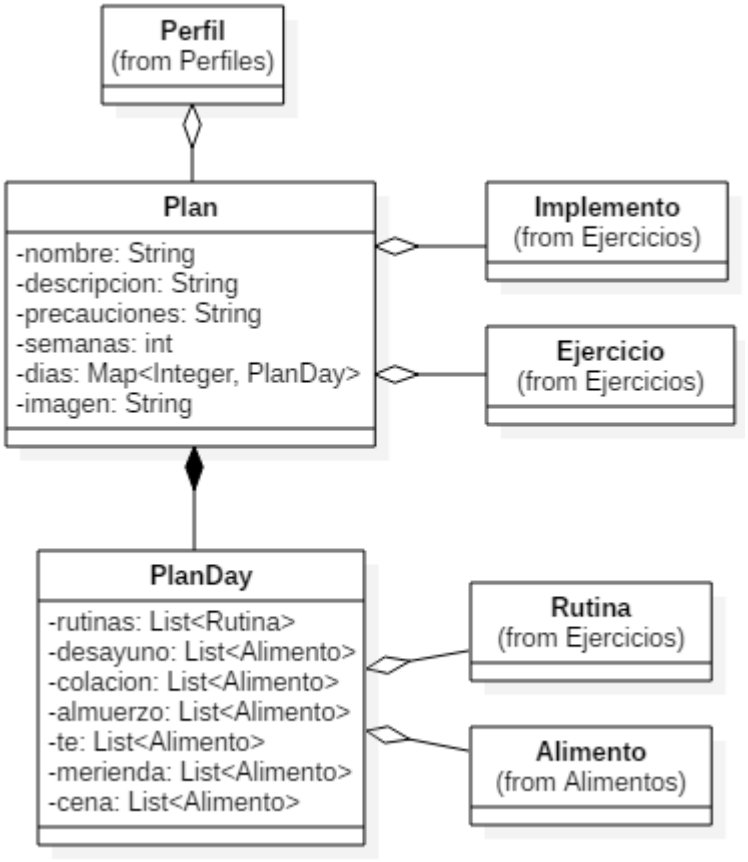


Figura 5.7: Diagrama de clases, paquete Planes.

### 5.3.5. Posts

En la Figura 5.8 puede observarse la existencia de *posts*, siempre asociados a un perfil, sea este el del autor o no. Además existen tipos de posts para distintas acciones, a saber:

- Post básico o *por default*.
- Crear receta, rutina o plan. (3 distintos).
- Evaluar receta, rutina o plan. (3 distintos).
- Compartir alimento, receta, rutina o plan. (4 distintos).
- Compartir post.

El tipo de post resulta de utilidad a la hora de especializar la vista.

Además, puede observarse la interfaz Compartible, mediante la cual se realiza *inversión de dependencias* con el fin de estandarizar el contenido compartido en un post.

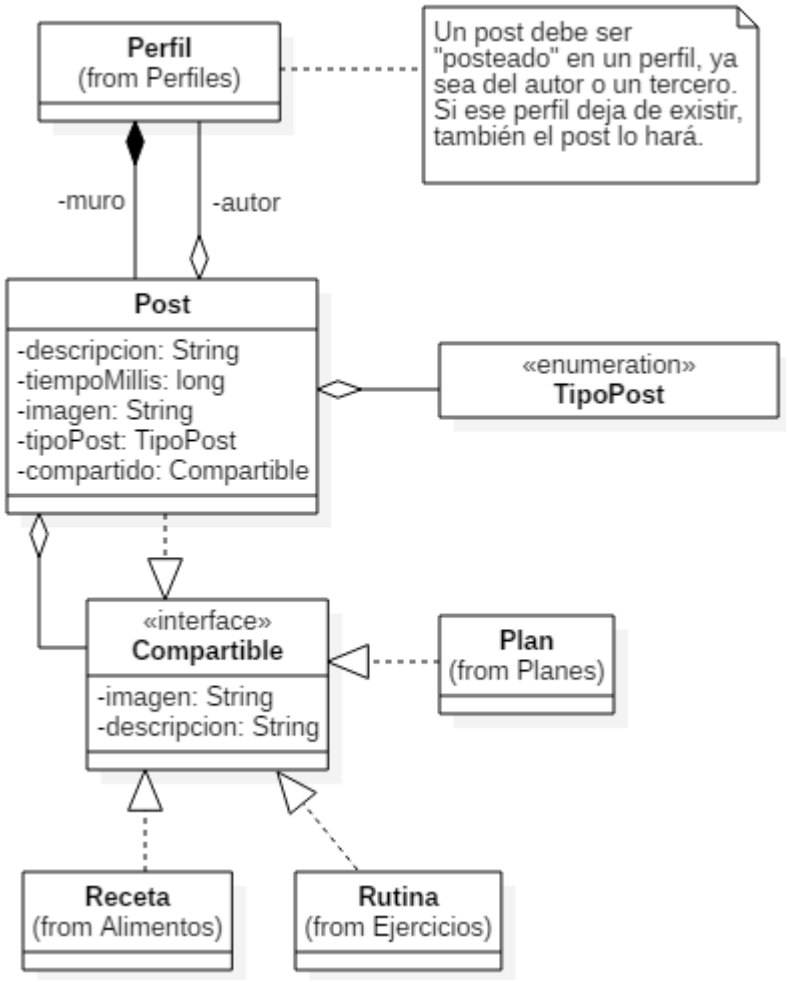


Figura 5.8: Diagrama de clases, paquete Posts.

### 5.3.6. Comentarios

En la Figura 5.9 se ve el uso de comentarios para las clases Receta, Rutina, Plan y Post; esto debido a que son todos los elementos creados por usuarios. Para ese fin se crea la interfaz Comentable. No se dispone de respuestas a los comentarios.

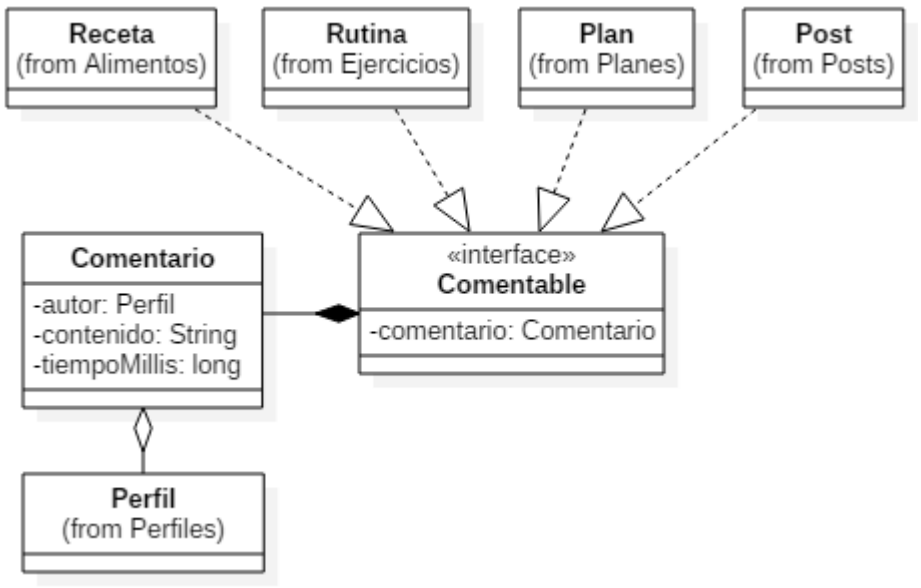


Figura 5.9: Diagrama de clases, paquete Comentarios.

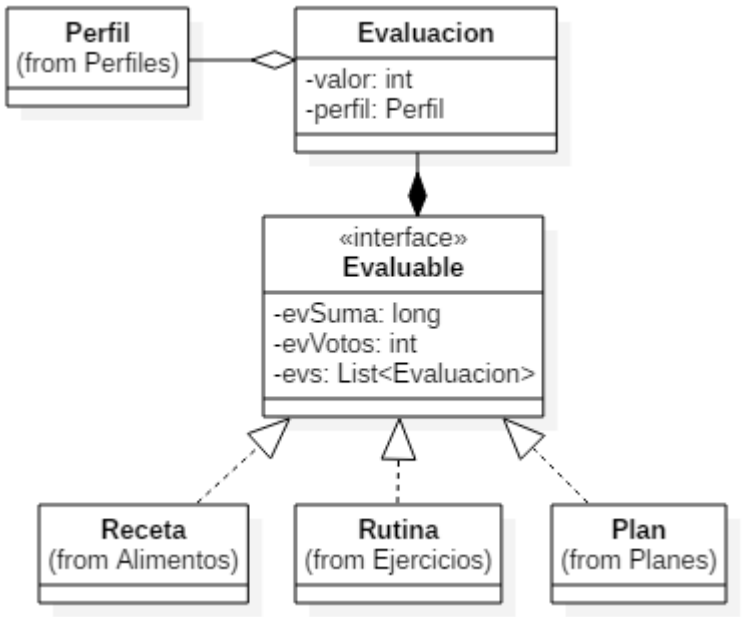


Figura 5.10: Diagrama de clases, paquete evaluaciones.

### 5.3.7. Evaluaciones

Las evaluaciones o valoraciones (Figura 5.10) corresponden a una nota de 1 a 5 realizada por los usuarios, y está disponible para recetas, rutinas y planes, que son elementos creados por usuarios. No se incluye evaluación de posts, dado que no entrega valor para el usuario.

El diseño escogido permite mantener coherencia en el valor de un elemento frente a los siguientes casos de uso:

- Realizar una evaluación.
- Mostrar a un usuario su evaluación y limitar esta a una sola.
- Borrar una evaluación realizada.

### 5.3.8. Likes

Para permitir dar "me gusta" o *like*, se crea la interfaz Likeable (Figura 5.11) para ser implementada por aquellas clases que deban permitir likes. Actualmente, solo Post hace uso de ella.


Figura 5.11: Diagrama de clases, paquete Likes.

### 5.3.9. Notificaciones

Finalmente, para la creación de notificaciones, se dispone de la interfaz Notificable para referenciar al elemento de donde surge la notificación, el cual es diferenciado según el tipo de notificación. Además, estas cuentan con un perfil del autor y un perfil del notificado (Figura 5.12).

Los tipos de notificaciones definidos son:

- Seguir usuario.
- Comentar receta, rutina, plan o post. (4 distintos).

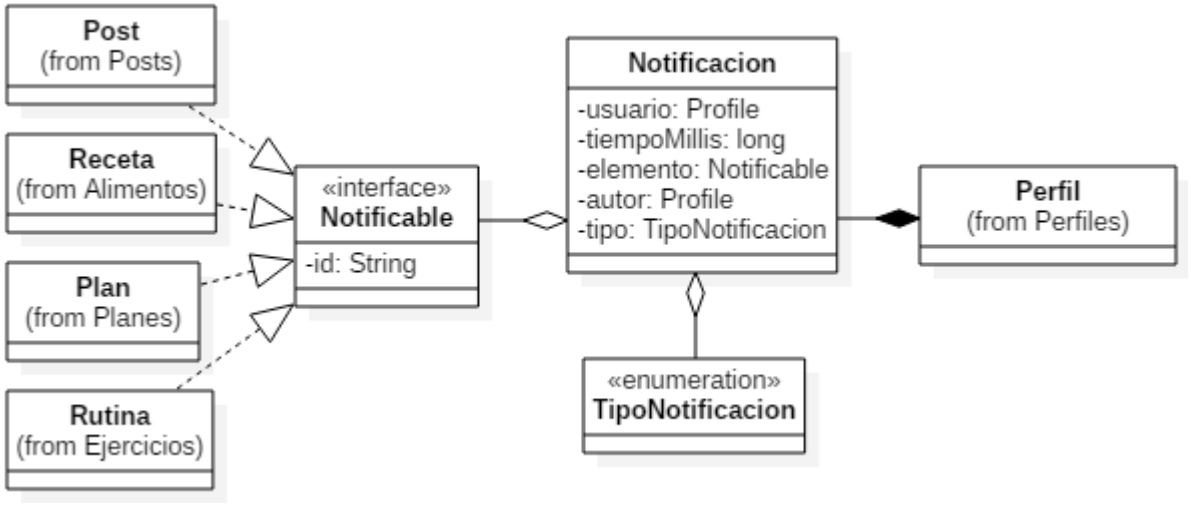


Figura 5.12: Diagrama de clases, paquete Notificaciones.

- Valorar receta, rutina o plan. (3 distintos).
- Compartir receta, rutina, plan o post. (4 distintos).
- Duplicar receta, rutina o plan. (3 distintos).
- Dar "me gusta" a un post.

## 5.4. Base de datos

A continuación se expone una representación de la base de datos (Figura 5.13, 5.14, 5.15 y 5.16). Si bien es una base de datos MongoDB, donde se suele usar duplicación de datos, se han mantenido documentos referenciados en otros, para mantener la consistencia frente a acciones de los usuarios. Esta técnica se ha usado principalmente en datos repetitivos como comidas de un plan, de tal manera que no se afecte la eficiencia gracias a la caché de MongoDB, o también en listas donde se requerirá el objeto referenciado completo.

Cabe mencionar que a pesar de no existir claves foráneas en este tipo de base de datos, se señalan las referencias más importantes a *ID* de otros documentos como *FK*, y otros datos duplicados como *FM*; esto únicamente con el fin de facilitar la lectura. Además aquellos campos que deberían estar presentes en cada documento insertado en una colección están marcados con un asterisco.

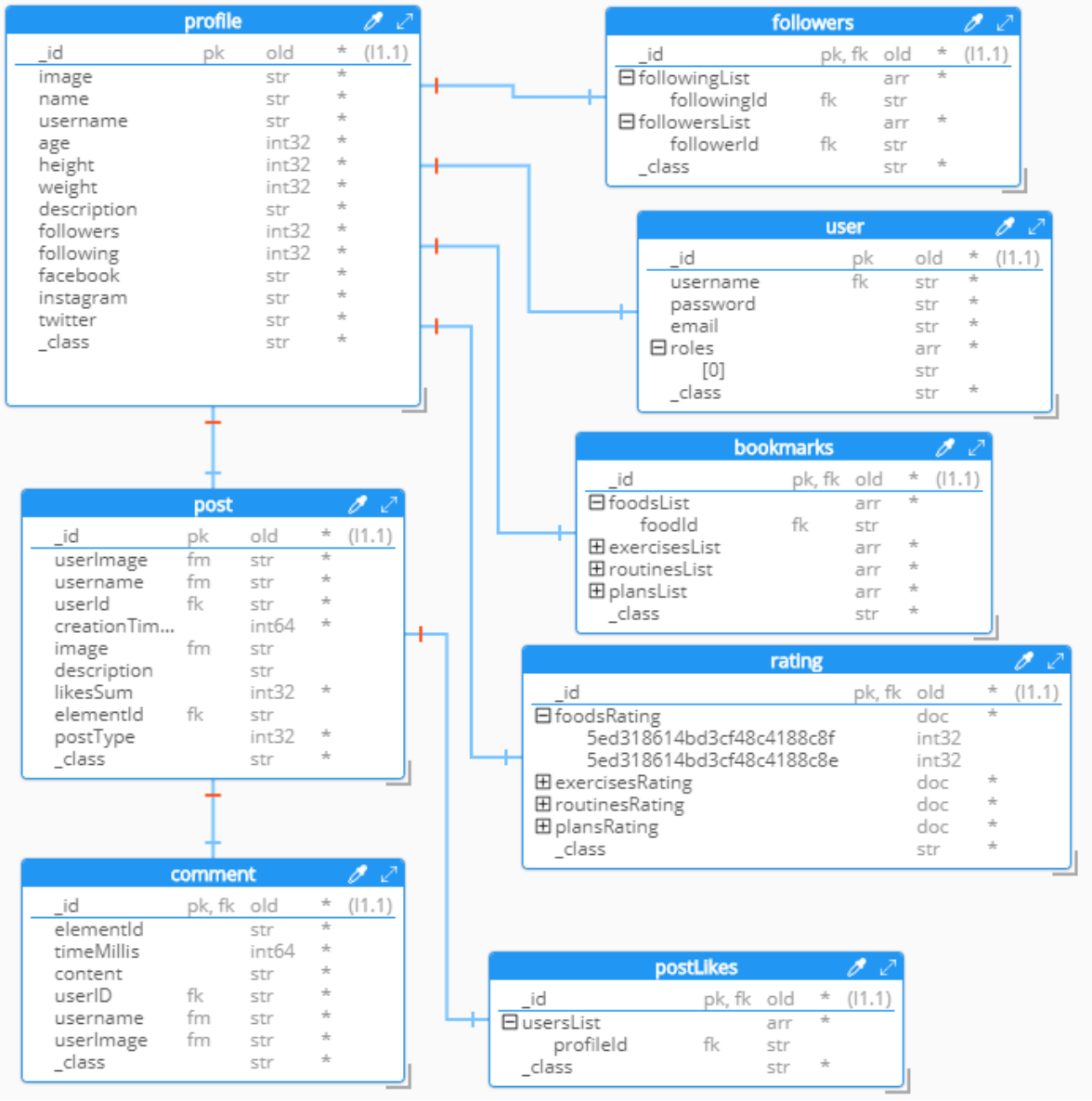


Figura 5.13: Diagrama de base de datos, parte 1.

En la Figura 5.13, puede observarse que los documentos de la colección *rating* existen campos con nombres que aparentan ser tokens. Ellos corresponden a diccionarios con la forma

$$ObjectId : rate$$

donde *rate* es la nota asignada por el perfil *_id* al objeto referenciado por *ObjectId*.

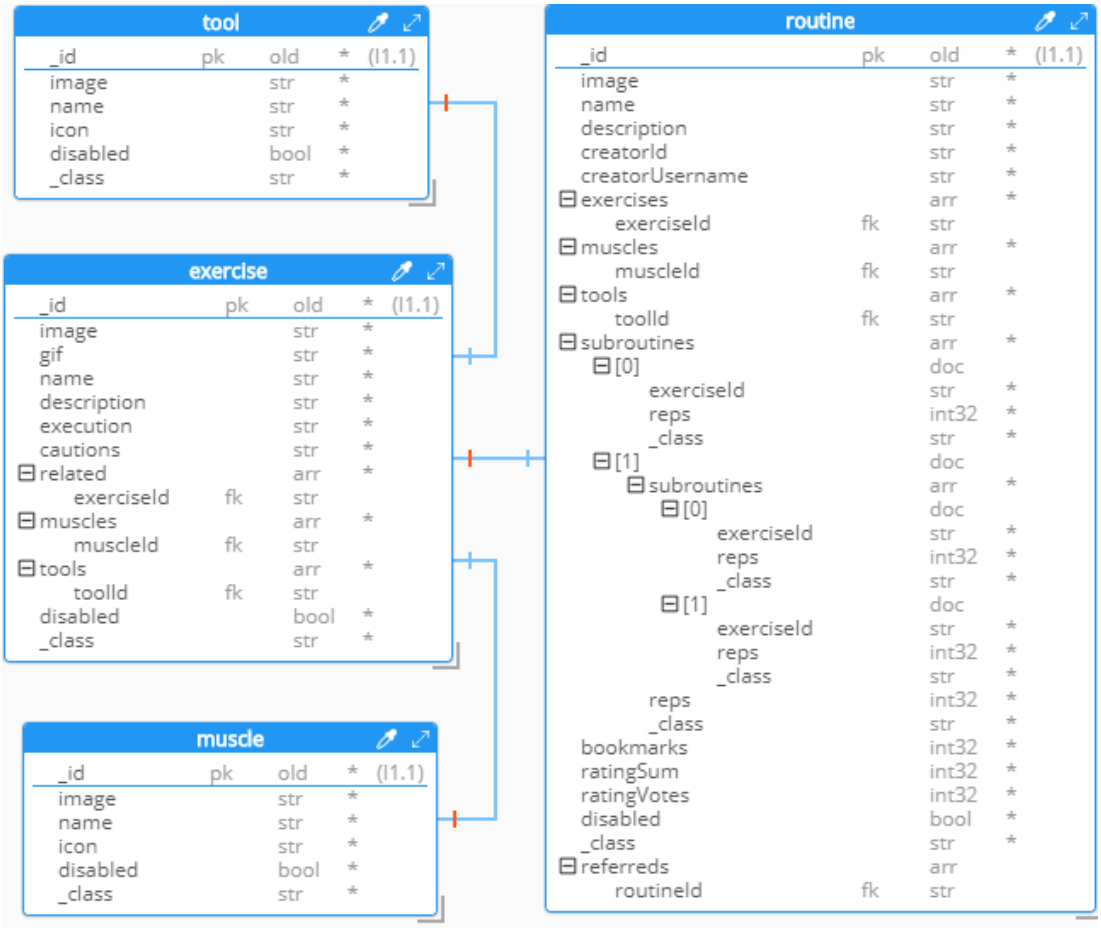


Figura 5.14: Diagrama de base de datos, parte 2.

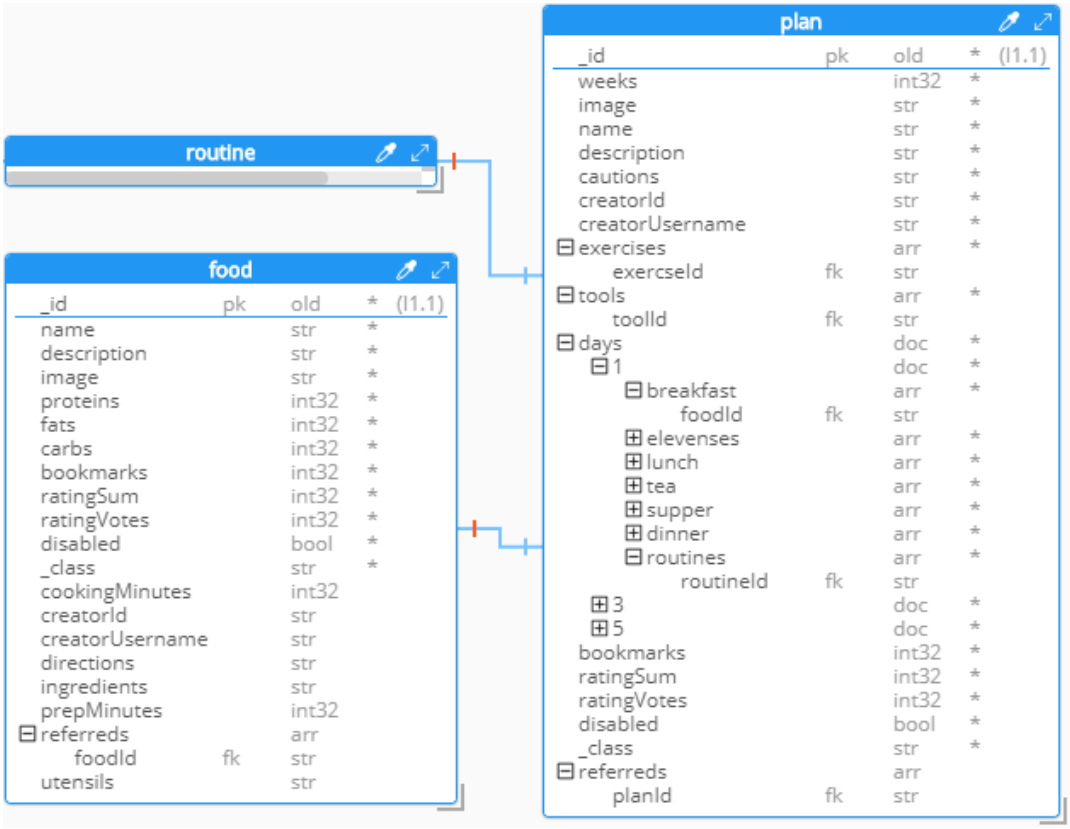


Figura 5.15: Diagrama de base de datos, parte 3.

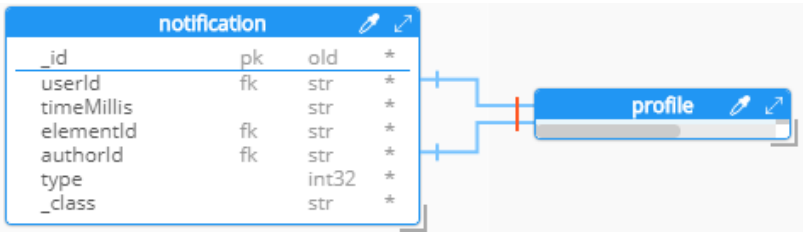


Figura 5.16: Diagrama de base de datos, parte 4.

## 5.5. Arquitectura

A continuación se presenta la arquitectura lógica del sistema, la cual es de tipo cliente-servidor con una base de datos MongoDB, tal como se mencionó en la Sección 5.1 y se ve en la Figura 5.1.

Considerando las características de la aplicación y todo lo mencionado en el capítulo se escogen las siguientes tecnologías: **Spring** y **Angular**.

### 5.5.1. Servidor

En el caso del servidor se utiliza el lenguaje Java junto a su framework Spring, debido a su gran cantidad de herramientas disponibles. En particular se usan los siguientes subproyectos:

- Spring WebFlux: Provee la base reactiva (basada en *streams*) de la aplicación. Esto significa un funcionamiento no bloqueante de buen desempeño [37]. Como implicancia está el uso de programación declarativa.
- Spring Data JPA: Facilita la conexión y uso de la base de datos, mediante el uso de interfaces especializadas.
- Spring Security: Facilita y mejora la autenticación, proveyendo seguridad a la aplicación.

Para realizar la trasformación de los datos desde su forma en la base de datos (**entidades**) a su forma en la aplicación cliente (**modelos**), privilegiando la escalabilidad y facilitando la verificación de procesos, se utilizarán cuatro capas en el servidor:

- **Repositorios**: Encargados de la conexión a la base de datos. Utilizan únicamente entidades. Su función se limita a extraer, guardar y eliminar datos, o marcarlos como eliminados según sea el caso.
- **Servicios**: Capa intermedia encargada de la transformación de entidades a modelos y viceversa, valiéndose de los distintos repositorios para esto. Es aquí donde se realizan las acciones necesarias para mantener la integridad entre los datos y algunas acciones automáticas como la creación de notificaciones. También hay un servicio para autenticación, además de un servicio de imágenes, encargado del almacenamiento de estas en un directorio.
- **Controladores**: Capa encargada de proveer una API RESTful a la cual se conecta la aplicación cliente. Trabaja solo con modelos. Se vale de los distintos servicios para obtener los datos a entregar. En esta capa se realiza la autorización. En el caso del controlador para imágenes, este funciona con

- **Seguridad**: Capa encargada de la autenticación. La autenticación inicial es mediante *authentication basic*, con la cual se genera un token JWT, el cual se usará posteriormente para una autenticación por *bearer* (Figura 5.17). Los endpoints de la api se encuentran securizados, con excepción de la descarga de imágenes. Cabe decir que los passwords se almacenan encriptados mediante el algoritmo de hashing *Bcrypt*.

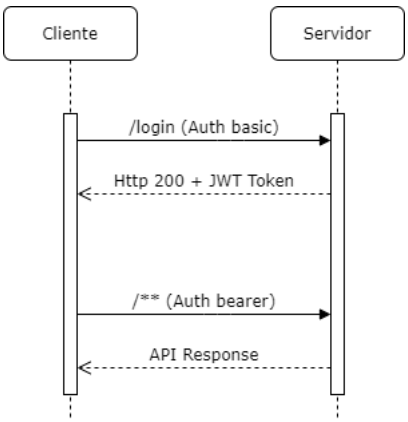


Figura 5.17: Secuencia de autenticación.

En la Figura 5.18 se muestra un ejemplo del funcionamiento del backend para la solicitud de un plan por parte de un usuario con sesión iniciada.

Como se observa en la Figura 5.18, la descarga de imágenes se realiza de manera independiente a las otras peticiones, y no requiere de un chequeo de autenticación. El objetivo de esto es evitar algunos problemas de ancho de banda, mejorando así la experiencia de usuario. En complemento a lo anterior, las imágenes son guardadas en un directorio, con un nombre generado mediante hashing, basado en datos como el nombre de usuario y el momento actual en milisegundos.

### 5.5.2. Cliente

En el caso del cliente se utiliza Angular, ya que, dadas sus características, resulta bueno en términos de mantenibilidad y escalabilidad. Además, provee seguridad, por ejemplo en la sanitización de código. Angular está diseñado para funcionar de manera reactiva, permite la detección temprana de errores al estar basado en Typescript, y provee un marco de trabajo MVVM. Adicionalmente, se cuenta con los

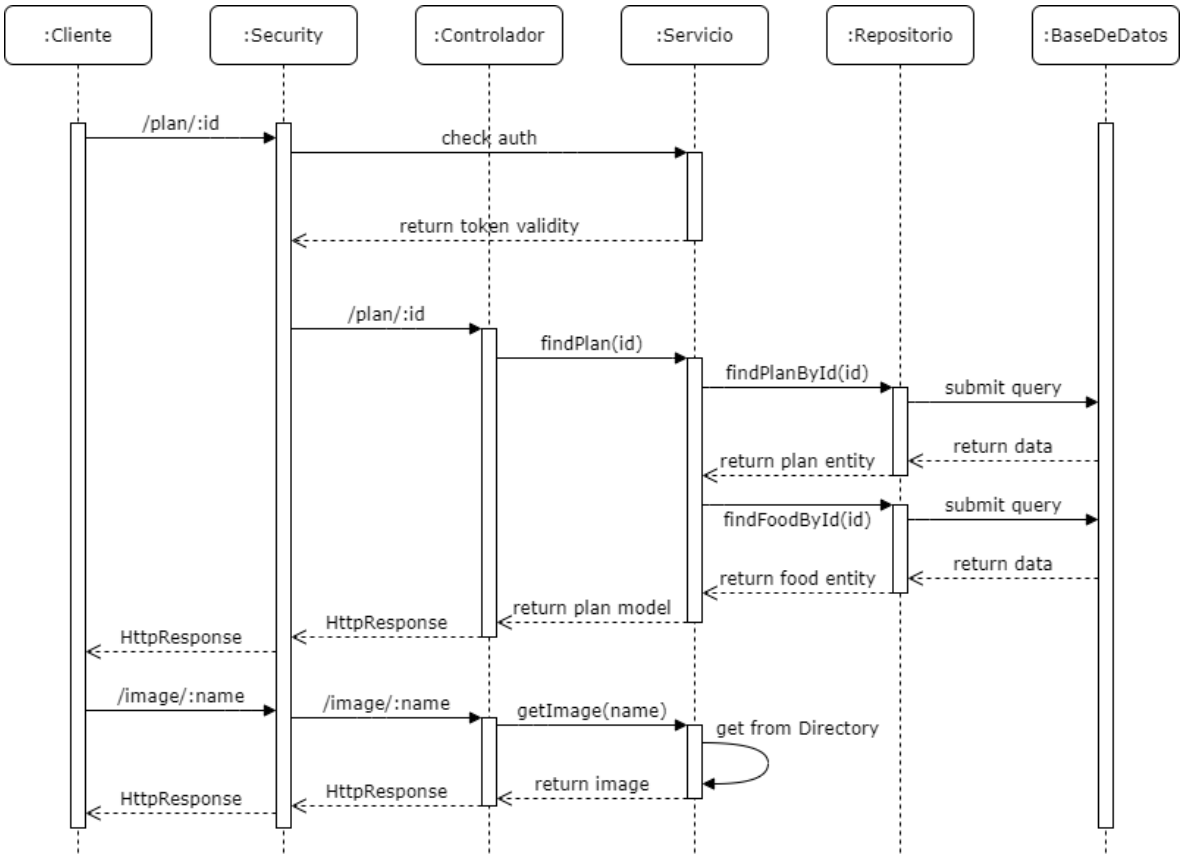


Figura 5.18: Secuencia backend para cargar un plan.

distintos paquetes disponibles para Node Package Manager (NPM). Los principales componentes de arquitectura que existirán en el cliente son:

- **Interceptor**: Capa que inspecciona y modifica las peticiones http, usada principalmente para autenticación, agregando el *bearer* antes de que sean realizadas, y tomando acciones si corresponde tras recibir una respuesta.

- **Servicios**: Capa encargada principalmente de generar las peticiones http para consumir la API REST. También existen servicios para características trascendentes en la aplicación, como los mensajes emergentes o el gestionar los datos del usuario que inicio sesión.

- **Componentes**: Capa intermedia que gestiona el flujo de información y las estructuras necesarias, entre servicios y templates. Se corresponde con *View-Model* de una arquitectura MVVM.

- **Templates**: Capa encargada de presentar la información al usuario. Se corresponde con las vistas de una arquitectura MVVM.
- **Decoradores**: Conjunto de herramientas utilizadas por los templates. Incluye *pipes* y *directivas*.

Cabe mencionar que junto a las clases del modelo, se han agregado interfaces que permitan cumplir el **principio de inversión de dependencias** entre componentes, favoreciendo así la reutilización de código.

### 5.5.3. Arquitectura lógica completa

En virtud de todo lo anterior, se obtiene la arquitectura lógica del sistema presentada en la Figura 5.19, la cual está implementada sobre stack reactivo, ofreciendo así un mejor rendimiento frente a la concurrencia, en comparación a alternativas bloqueantes. Las distintas capas favorecen la escalabilidad y mantenibilidad al seguir el **principio de responsabilidad única**, maximizando la cohesión y minimizando el acoplamiento. Como se mencionó anteriormente, los componentes Models y Entities en el diagrama corresponden a los modelos del dominio en la forma requerida por el frontend y por la base de datos, respectivamente.

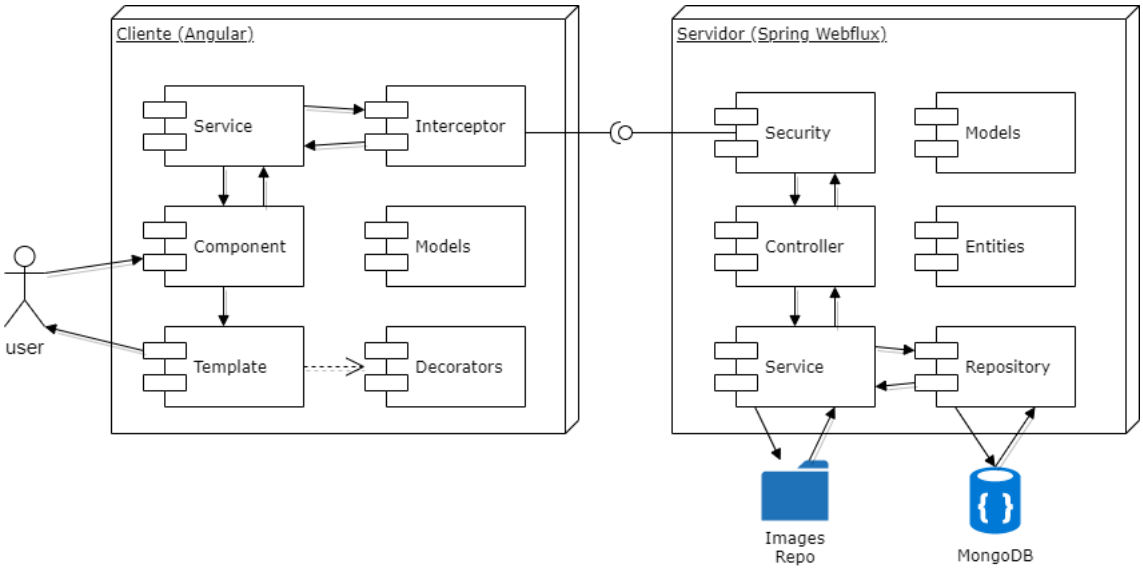


Figura 5.19: Arquitectura lógica del sistema.

## 5.6. Resumen del capítulo

En base a los requisitos del proyecto se ha definido el mapa de navegación y el modelo del sistema. Una vez definidos, se han especificado las colecciones de la base de datos, teniendo en consideración la integridad de los datos y la eficiencia. Conocido lo anterior se han escogido tecnologías adecuadas: Angular y Spring; y se ha concretado la arquitectura lógica del sistema.

# 6. Implementación

En este capítulo se detalla el proceso de implementación separado según los *sprints* de tres semanas. Para cada sprint se expone un objetivo de desarrollo actualizado según las vicisitudes del proceso, junto con el estado del producto, es decir, los avances logrados. Este proceso estuvo basado en los requisitos definidos en el Capítulo 4, los cuales fueron divididos en tareas con una dificultad e importancia asociadas (dichas tareas son específicas y no están detalladas en este documento).

## 6.1. Sprint 0

### Objetivo

Definir las bases para el desarrollo. En particular la arquitectura del sistema y documentos útiles para la clarificación de requisitos.

**Historias a abordar:** No aplica

### Estado

En la primera semana se realizaron los wireframes iniciales junto al mapa de navegación. Durante las otras dos semanas se definió la versión inicial del product backlog, la arquitectura lógica y las tecnologías a utilizar.

## 6.2. Sprint 1

### Objetivo

Desarrollar la base de la aplicación.

### Estado

En la primera semana se estudiaron algunas plantillas para el *frontend*, y una de ellas fue aplicada. La segunda semana se implementó la aplicación servidor con su estructura base y fue conectada al frontend. La última semana de este *sprint* se implementaron las acciones básicas (CRUD) para la entidad **Músculo** a través de todo el stack.

**Reajustes en los requerimientos:** Agregado DAM20 "mostrar mensajes informativos al realizar acciones". Se sugieren cambios a la interfaz.

## 6.3. Sprint 2

### Objetivo

Permitir subir y mostrar imágenes.

### Estado

- Se corrigen algunos aspectos de la interfaz, además de agregar un logo provisorio.
- Se agregan mensajes de información tipo *toast* (mensajes emergentes de notificación de eventos).
- Se implementa un servicio que permite subir imágenes, guardándolas como archivo de nombre generado por *hashing*, con una referencia en base de datos. Esto junto a la contraparte en frontend.
- Se agregan dos CRUDs: Implemento y Ejercicio.

## 6.4. Sprint 3

### Objetivo

Implementar funcionalidades básicas de rutinas, además de asociaciones entre elementos existentes.

**Historias a abordar:** DAM2, DAM1, DAM8.

#### Estado

- Agregado CRUD básico de rutinas. En *frontend* queda pendiente construir la rutina.
- Se pueden ejecutar rutinas agregadas directamente en backend.
- Agregadas asociaciones en los formularios de edición.
- Mejorada interfaz de vista de detalles.

**Historias terminadas:** DAM2.
**Historias no terminadas:** DAM1, DAM8.
**Reajustes en los requerimientos:** Módulo de alimentación sube en prioridad.

## 6.5. Sprint 4

#### Objetivo

Implementar características referentes a alimentos y recetas.
**Historias a abordar:** DAM3, DAM4, ADM2.

#### Estado

- Agregado CRUD de Alimento.
- Agregado CRUD de Receta con texto enriquecido. Dicho texto es sanitizado.
- Agregado en frontend el módulo SharedModule con funcionalidades compartidas en toda la aplicación.
- Agregado mockup básico de la página principal y perfil de usuario.

**Historias terminadas:** DAM3, DAM4, ADM2.
**Reajustes en los requerimientos:** Agregado DAM21 "Permitir realizar comentarios en los distintos elementos". Se sugieren cambios a la interfaz.

## 6.6. Sprint 5

### Objetivo

Implementar CRUD de planes y construcción de rutinas.
**Historias a abordar:** DAM1, DAM6, DAM8.

### Estado

- Terminada la construcción de rutinas de ejercicios en el frontend, admitiendo solo rutinas secuenciales.
- Se permite mostrar planes creados directamente en el backend.
- Queda pendiente la edición de planes; específicamente el poder modificar la duración del plan y el contenido de los días.

**Historias terminadas:** DAM1, DAM6, DAM8.
**Historias no terminadas:** DAM5.

## 6.7. Sprint 6

### Objetivo

Implementar la edición de planes y el perfil de usuario.
**Historias a abordar:** DAM5.

### Estado

- Terminado módulo de planes.
- Agregado mockup de login.
- Vista de perfil de usuario avanzada. Pendientes las vistas parciales.

**Historias terminadas:** DAM5.
**Reajustes en los requerimientos:** (Cambios de prioridades, nuevos requerimientos)

## 6.8. Sprint 7

### Objetivo

Implementar la autenticación de usuario. Filtrar el contenido mostrado según el usuario que inició sesión. Agregar a un perfil listas de *seguidos* y *seguidores*.

**Historias a abordar:** DAM9, DAM10, DAM17.

### Estado

- Usuario puede autenticarse.
- Implementación mínima de registro de usuario.

**Historias no terminadas:** DAM9, DAM10, DAM17.

**Reajustes en los requerimientos:** Se sugiere agregar un recortador de imágenes para fotos de perfil.

## 6.9. Sprint 8

### Objetivo

Completar tareas sprints 6 y 7: perfil de usuario y autenticación. Implementar seguidores. Dar compatibilidad a usuarios y características anteriores. Registrar usuario.

**Historias a abordar:** DAM9, DAM10, DAM17

### Estado

- Perfil de usuario actualizado y terminado.
- Se adecúan las características existentes para que funcionen de acuerdo al usuario autenticado.
- Se permite seguir a otros usuarios.
- Las características disponibles son reordenadas, corregidas y terminadas.
- Interfaz actualizada según las pruebas CTA y RP previas.

- Se agregan funcionalidades menores, tales como favoritos, valoración y crear elemento en base a uno existente.

**Historias terminadas:** DAM19.

**Historias no terminadas:** DAM9, DAM10, DAM17, DAM21.

**Reajustes en los requerimientos:** Agregado DAM22 "Enviar notificaciones frente a acciones realizadas".

## 6.10. Sprint 9

### Objetivo

Mejorar el registro de usuario. Dar funcionalidad al mockup de la vista principal, es decir, agregar historias a un *feed*. Realizar las correcciones pertinentes.

**Historias a abordar:** DAM9, DAM10, DAM17.

### Estado

- Se implementa un registro de usuario que considere la experiencia de usuario.
- Se finaliza la vista principal (dashboard), el cual carga los posts mediante un "scroll infinito".
- Se implementa la generación de posts según acciones de usuarios.

**Historias terminadas:** DAM9, DAM10, DAM17.

**Reajustes en los requerimientos:** Se sugieren cambios a la interfaz. Se señala la importancia de DAM22.

## 6.11. Sprint 10

Tras realizar las últimas pruebas de usuario preliminares se decidió agregar un décimo sprint de duración reducida (1 semana), para corregir aquellas necesidades coincidentes entre los usuarios.

**Objetivo**

- Agregar notificaciones.
- Mostrar contenido de las rutinas al ejecutarlas.
- Corregir defectos menores.

**Historias a abordar:** DAM22

**Estado**

- Se generan notificaciones, cuya cantidad se muestra en la barra de navegación.
- Se muestra el contenido de las rutinas en forma resumida.
- Mejorada la interfaz de edición de planes.
- Correcciones menores.
- Agregada más y mejor información de ejemplo.

**Historias terminadas:** DAM22

## 6.12. Resultado

Tras el proceso de desarrollo abarcado en el presente proyecto, se ha obtenido una aplicación que cumple con características suficientes para ser competitiva según el Cuadro 4.1. En el Anexo B, se pueden observar registros visuales de dicho MVP. Este cumple con la mayoría de requisitos de usuario de tipo deportista amateur (se excluyen DAM11, DAM13, DAM14, DAM15 y DAM18) y tipo administrador (todos implementados). No se implementaron los requisitos de entrenador, dada la menor importancia definida. Por otro lado, en las sesiones de CTA surgieron nuevas características que resultaban importantes para usuarios o para un mejor funcionamiento, como las siguientes:

- Mensajes de información tras realizar algunas acciones.
- Editor de texto enriquecido para algunos campos.

- Recortador de imágenes para que cumplan con proporciones. Por ejemplo, 1 : 1 para fotos de perfil.
- Comentarios para posts y elementos.
- Likes en posts.
- Notificaciones.

## 6.13. Resumen del capítulo

En este capítulo se ha señalado un resumen del proceso de implementación del MVP mediante la metodología definida, y el resultado obtenido, que incluye los requisitos de tipo deportista amateur (excluyendo DAM11, DAM13, DAM14, DAM15 y DAM18) y requisitos de administrador, junto a otros surgidos durante el desarrollo, donde destacan comentarios y notificaciones.

# 7. Verificación y evaluación

En este capítulo se presenta la forma de las pruebas de verificación, las cuales se han aplicado durante el desarrollo. También se presentan las pruebas de validación/evaluación junto a los resultados obtenidos en ellas y su interpretación. Finalmente, se expone una retrospectiva de todo el proceso de desarrollo expuesto.

## 7.1. Pruebas automatizadas

El desarrollo estuvo guiado mediante la técnica *test-driven development* (TDD), es decir, implementando pruebas en primer lugar, para así evitar en lo posible la presencia de errores. Lo casos de usos a verificar, basado en desarrolladores experimentados [45], fueron los siguientes:

- Caso más común esperado.
- Casos extremos permitidos y no permitidos.
- *Bugs* complejos por resolver.

Se realizaron pruebas automatizadas unitarias y de integración. Se consideró realizar pruebas de sistema, pero se descartó debido a los recursos necesarios para desarrollar pruebas que den valor al producto.

### 7.1.1. Pruebas unitarias

A continuación se expone el formato de las pruebas unitarias de caja negra. No se realizaron pruebas formales de caja blanca, debido al tiempo disponible y el hecho de que el sistema no es crítico.

Figura 7.1: Prueba unitaria: análisis de rutina

```
describe('RoutineSetupComponent', () => {
    //Let's assume some boilerplate code
    beforeEach(() => {
        //...//
        component = new RoutineSetupComponent(...);
    }

    var routineAsArray = [
        {serieStart: true, reps: 3},
        {exercise: {...}, reps: 8}, // exercise is a complex object
        {serieStart: true, reps: 3},
            {exercise: {...}, reps: 10}
        {serieEnd: true},
        {serieEnd: true},
        {exercise: {...}, reps: 12}
    ];
    var expected = [
        {
            reps: 3,
            subroutines: [
                {reps: 8, exerciseId: "eId1"},
                {
                    reps: 3,
                    subroutines: [
                        {reps: 10, exerciseId: "eId2"}
                    ]
                }
            ]
        },
        {reps: 12, exerciseId: "eId3"}
    ]

    it('should parse the routine', () => {
        result = component.parseRoutine(routineAsArray);
        expect(result).toEqual(expected);
    });
}
```

Las pruebas unitarias de caja negra se implementaron mediante el uso de JUnit en el backend y, con Jasmine y Karma en el caso del frontend. Estas pruebas se aplicaron en aquellas funciones que podían verse alteradas frente a un cambio en el modelo, o aquellas propensas a fallar dada su complejidad. Cabe mencionar que para la realización de estas pruebas, se hace uso de *Mocks* que simulan de forma estática el comportamiento de las otras capas en caso de ser necesario.

La estructura de las pruebas es la siguiente:

- Valor de entrada.
- Salida esperada.
- Salida obtenida.

En la Figura 7.1 se puede observar el test para una función que analiza sintácticamente (a un alto nivel) una lista de elementos para convertirlos en una rutina con subrutinas anidadas.

## 7.2. Pruebas de integración

Las pruebas de integración realizadas tienen la misma forma de las pruebas unitarias, pero abarcan varias capas a la vez, enfocadas en una funcionalidad. Solo fueron implementadas en backend debido a los recursos disponibles.

En la Figura 7.2 puede observarse las pruebas a un componente tipo *repository* en el cual se verifica el uso correcto de la base de datos.

Además, durante el desarrollo se realizaron algunas pruebas no automatizadas para verificar el funcionamiento de la API del backend mediante el uso de la herramienta Postman, la cual facilita la realización de peticiones http.

## 7.3. Concurrent Think Aloud

La metodología CTA fue utilizada durante el desarrollo con dos objetivos principales:

- Determinar interfaces y funcionalidades intuitivas para la aplicación.
- Detectar defectos tempranamente, para así corregirlos con prontitud.

Figura 7.2: Prueba de integración: capa repository

```
1  // Imports and context configuration
2  public class MuscleRepositoryTest
3  {
4      @Autowired
5      private MuscleRepository muscleRepository;
6      private Muscle original = new Muscle(...);
7      private Muscle muscle1;
8      private Muscle muscle2;
9
10     @Test
11     public void shouldCreate() {
12         muscle1 = muscleRepository.save(original).block();
13         assertEquals("Pectoral", muscle1.getName());
14     }
15
16     @Test
17     public void shouldRead() {
18         muscle2 = muscleRepository.findById(muscle1.getId()).block();
19         assertEquals("Pectoral", muscle2.getName());
20     }
21
22     @Test
23     public void shouldUpdate() {
24         muscle2.setName("Pectoral mayor");
25         muscle2 = muscleRepository.save(muscle2).block();
26         assertEquals("Pectoral mayor", muscle2.getName());
27         assertNotEquals("Pectoral mayor", muscle1.getName());
28         muscle1 = muscleRepository.findById(muscle1.getId()).block();
29         assertEquals("Pectoral mayor", muscle1.getName());
30     }
31
32     @Test
33     public void shouldDelete() {
34         muscleRepository.deleteById(muscle1.getId()).block();
35         muscle2 = muscleRepository.findById(muscle1.getId()).block();
36         assertNull(muscle2);
37     }
38 }
```

Tras las distintas instancias de CTA se logró mejorar la usabilidad, mediante la corrección de algunos aspectos visuales, tales como:

- Uso de colores.
- Mensajes de retroalimentación frente a acciones del usuario.
- Forma y ubicación de elementos.
- Navegación dentro del sitio.

Junto a esto hubieron pequeñas modificaciones de los atributos de algunas clases, como el uso de textos con formato enriquecido.

Por último, estas revisiones fueron de utilidad para encontrar errores de funcionamiento que se habían pasado por alto en las pruebas realizadas en el frontend.

## 7.4. Retrospective Probing

La metodología RP fue utilizada tras tener finalizada la aplicación. Para su realización se dio acceso al sistema a varias personas, con un listado de posibles tareas a realizar, pero con libertad de acción. Estas tareas se dividieron en 4 grupos.

Tareas asociadas al ámbito social:

- Crear un post.
- Compartir elementos.
- Dar like a un post y comentar.
- Seguir y dejar de seguir a un usuario.

Tareas asociadas a ejercitación:

- Revisar información de musculatura, implementación, etc.
- Ver los ejercicios asociados a un músculo.
- Revisar una rutina.
- Ejecutar una rutina.

- Crear, duplicar, eliminar y editar una rutina.

Tareas asociadas a alimentación:

- Revisar información de alimentos y recetas.
- Crear, duplicar, eliminar y editar una receta.

Otras tareas:

- CRUD planes.
- Ver notificaciones.
- Guardar elementos como favoritos.
- Valorar elementos.
- Actualizar perfil.

Luego fue aplicada una encuesta con cuatro partes (ver Anexo C):

- Datos básicos.
- Escala SUS de usabilidad.
- Percepción de cumplimiento de objetivos del proyecto.
- Preguntas abiertas.

Esta encuesta fue respondida por 16 personas, entre las cuales se identifican dos grupos: aquellos relacionados con el fitness con anterioridad o **interesados** y, aquellos **no interesados**.

### 7.4.1. Evaluación de usabilidad SUS

En la escala SUS (ver Figura 7.5, y detalle en el Anexo C), tras calcular el promedio de los resultados se obtuvo un puntaje de 80,3 (resumen de resultados en Figura 7.3). Esto significa que se está en el límite inferior para una buena usabilidad, en que se comprende el 100 % de las funcionalidades probadas, quedando alrededor de un tercio de las personas bajo dicho límite. Sin embargo, al revisar los resultados solo de los interesados, los cuales se acercan al arquetipo buyer persona definido en

el Capítulo 4, el valor SUS aumenta a 90 (resumen de resultados en Figura 7.4), con lo cual se puede decir que la usabilidad es muy buena.

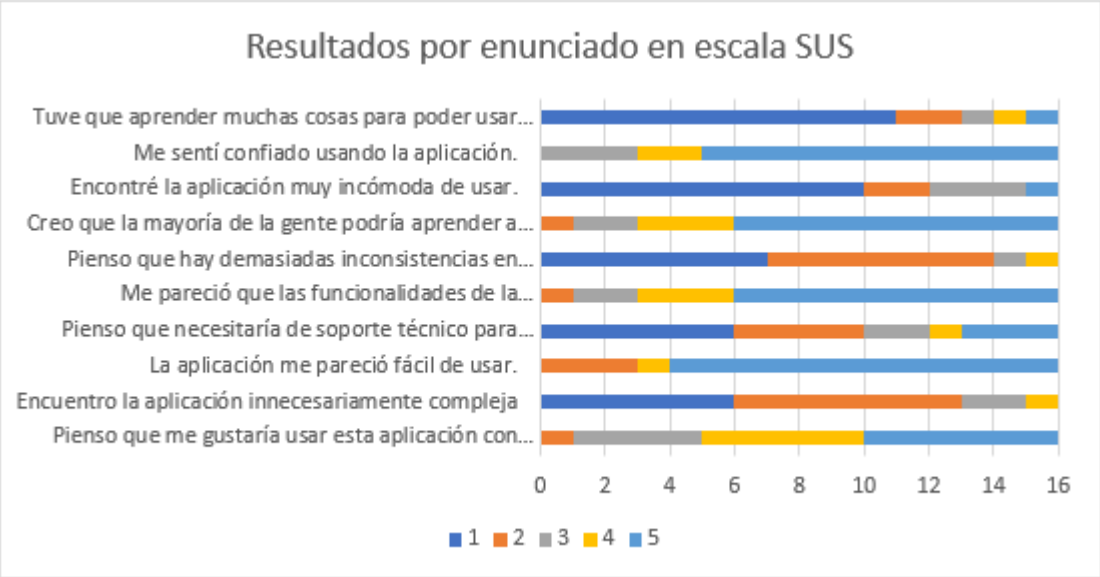


Figura 7.3: Resultados por enunciado en escala SUS: todos los usuarios.

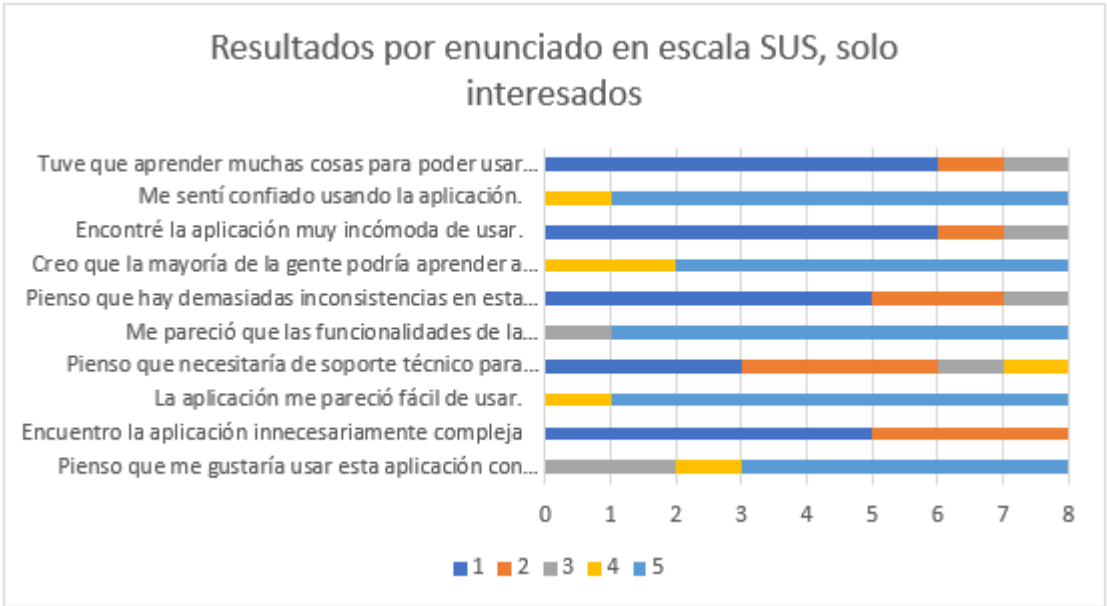


Figura 7.4: Resultados por enunciado en escala SUS: interesados.

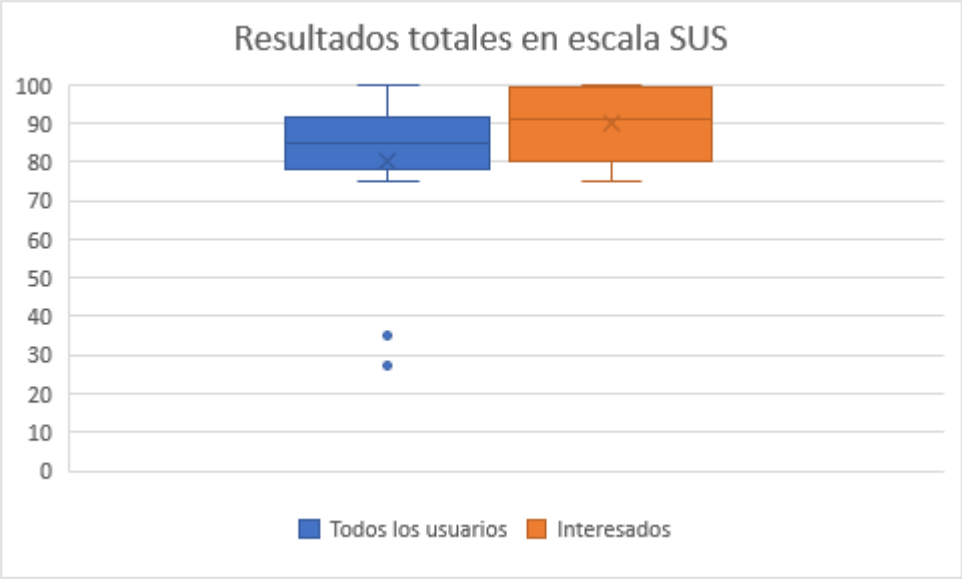


Figura 7.5: Resultados totales en escala SUS.

### 7.4.2. Evaluación de cumplimiento de objetivos

En la escala de cumplimiento de objetivos (detalle en el Anexo C), tras promediar y normalizar[1] las respuestas, se obtiene una percepción general de cumplimiento sobre el 80 % en cada objetivo específico, mientras que al considerar solo a los usuarios interesados en el fitness, el cumplimiento aumenta por sobre el 90 % (ver Figura 7.6).

Esto quiere decir que los objetivos se cumplen según la percepción de los usuarios. No obstante, en la encuesta se hace notar diversas mejoras a la aplicación, tales como las siguientes:

- Ofrecer mejor contenido prediseñado para novatos.
- Incluir un tutorial.
- Uso de tags y/o colores para clasificar elementos.
- Interacción y navegación por medio de hashtags.
- Historial de avances y acciones.
- Uso de videos.

[1]Se normalizó utilizando (valor-1)/4, obteniendo así un resultado en el rango de [0,1]

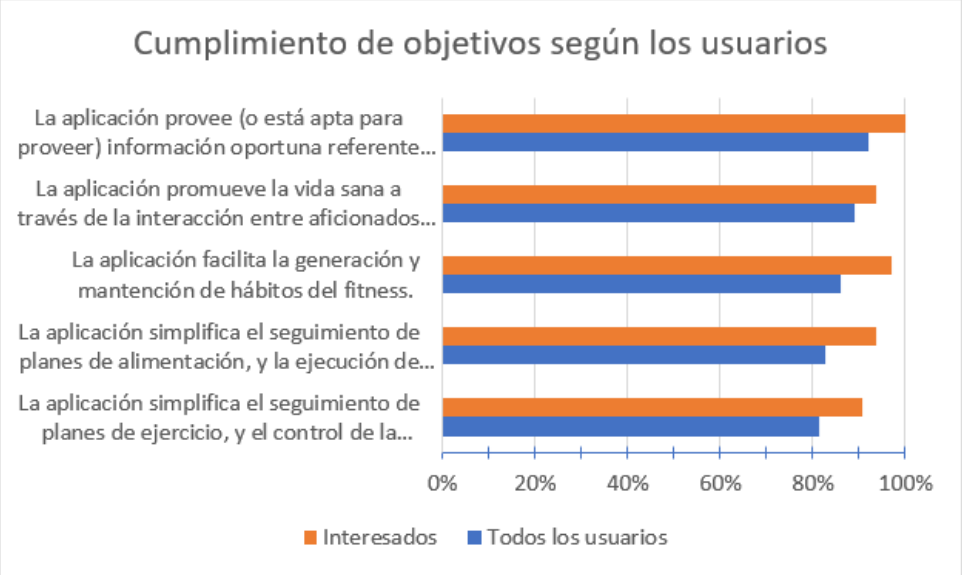


Figura 7.6: Cumplimiento de objetivos específicos según usuarios.

## 7.5. Interpretación de resultados

Tras evaluar la aplicación implementada, se puede apreciar la aceptación por parte de los usuarios, tanto en términos de usabilidad como de cumplimiento de objetivos, con lo cual se puede señalar que el proyecto es útil. No obstante, existen claras diferencias entre usuarios que ya son parte de la comunidad fitness con usuarios nuevos.

Este resultado era esperable, puesto que el diseño de las interacciones estuvo guiado mediante un arquetipo buyer persona. Sin embargo, evidencia algunos aspectos que podrían mejorarse para llegar a más usuarios, siendo el principal la entrega de información oportuna en forma de inducción a la aplicación previo a crear cuenta de usuario. También deberá completarse el tutorial de bienvenida y las páginas de ayuda. Por otro lado, se espera que con el aumento de usuarios y sus publicaciones mejoren automáticamente las malas evaluaciones basadas en el poco contenido.

## 7.6. Retrospectiva del proceso y aprendizajes

En todo este proceso de desarrollo se han percibido aprendizajes de distinta índole: metodología, tecnología, experiencia de usuario, planificación y estimación,

entre otras.

En cuanto a la metodología de desarrollo, se confirma que una metodología ágil funciona muy bien para proyectos como este. No obstante, el caso de Scrum puede significar una sobrecarga al intentar aplicarla para una persona, por lo que una metodología de desarrollo más simple como XP (extreme programming) podría ser una buena alternativa a considerar. Respecto a la metodología de evaluación, resultó muy importante el uso de pruebas tempranas observando las acciones de un usuario mientras usan la aplicación, a la vez que explican lo que piensan durante el proceso (Concurrent Think Aloud); sería interesante aplicar estas pruebas a mayor escala mediante herramientas de análisis como las de Google Analitycs. Por último, sobre la metodología de análisis y diseño, si bien se intentó simplificar para no entorpecer el desarrollo, de manera natural se acercó muchas veces a Design Thinking, la cual es una excelente opción para concretar nuevas características.

En cuanto a las tecnologías es importante notar como una aplicación web no monolítica sin estado (cliente y servidor independientes) resulta en una mejora de escalabilidad directamente, al desacoplar cliente y servidor gracias al uso de una API Restful. Junto a esto se reforzaron conocimientos como los siguientes: la utilidad de una base de datos no relacional frente a problemas que requerirían desnormalización, el proceso de securización de una aplicación web y las buenas prácticas aplicadas en todas las herramientas que otorgan los frameworks Spring y Angular.

Sobre la experiencia de usuario, fue muy importante el definir previamente el arquetipo de usuario, puesto que desarrollar teniendo en mente a alguien específico, permite obtener mejores resultados. También resultó muy importante el hacer partícipe a usuarios y hacer pruebas tempranamente, para no implementar una interfaz poco utilizable. Por último, se aprendió que ante la duda, siempre es bueno tomar en consideración el ejemplo de diseñadores (UI/UX) experimentados y seguir estándares aceptados.

Finalmente, en cuanto a planificación y estimación, se reconoce de mejor manera el tiempo usado en los distintos roles ejecutados, y los posibles contratiempos de cada uno de ellos. También se evidenció la importancia de considerar riesgos de distinto nivel, y tener un plan de acción para enfrentarlos.

## 7.7. Resumen del capítulo

En este capítulo se han presentado las pruebas automatizadas y evaluaciones de usuario realizadas a la aplicación. Los resultados señalan correctitud y aceptación. Además, se muestra que los objetivos del proyecto han sido cumplidos.

Por último, se señalan los aprendizajes provenientes de la observación de todo el proceso de desarrollo, incluyendo aprendizajes de metodología, tecnología, experiencia de usuario, y la planificación de proyectos.

# 8. Conclusión y trabajo futuro

En este último capítulo se presentan los resultados y conclusiones obtenidos tras todo este proceso de desarrollo. Junto a esto, se expone el trabajo futuro que debiera proseguir a este proyecto, clasificado en cinco tópicos, donde se incluyen tanto atributos funcionales como no funcionales.

## 8.1. Resultado

En el Capítulo 1, se planteó el siguiente objetivo: "Facilitar la realización sostenida de hábitos de vida saludable asociados al fitness, mediante el desarrollo de una plataforma web que permita gestionar y compartir planes de entrenamiento y de alimentación".

Con este fin se realizó el proceso de ingeniería descrito en todo el documento, durante el cual se usaron las siguientes metodologías:

- Metodología de análisis y diseño: Para definir los requerimientos y tareas a realizar, se realizó un diagnóstico del microentorno, un análisis de los usuarios y sus necesidades basado en una encuesta a 59 interesados, y adaptaciones continuas basadas en evaluaciones tempranas usando Concurrent Think Aloud.

- Metodología de desarrollo: Para la implementación, se hizo uso de una adaptación de Scrum para una persona, ordenando las tareas en un tablero kanban.

- Metodología de evaluación: Para la verificación, se utilizó test driven development, implementando tests unitarios y de integración en aquellas funcionalidades que estuvieran propensas a fallos. Por otro lado, para la validación se

realizó una encuesta que incluía usabilidad y cumplimiento de los objetivos del proyecto.

El resultado de este proceso fue un producto viable mínimo, que consistió en la siguientes características:

- Aplicación web de página única en idioma español latinoamericano, implementada en Angular, Spring (java) y MongoDB.
- Permite visualizar información referente a alimentos, musculatura, implementación deportiva y ejercicios.
- Permite gestionar, compartir y ejecutar rutinas de ejercicio.
- Permite gestionar y compartir recetas.
- Permite gestionar y compartir planes de entrenamiento y/o alimentación.
- Permite interactuar con otros usuarios mediante publicaciones, comentarios, likes y valoraciones.

La usabilidad del software desarrollado ha sido evaluada mediante escala SUS, obteniéndose como resultado un puntaje promedio de 80,3 entre todos los usuarios participantes, y de 90 al considerar a quienes tenían interés en el fitness. Estos resultados, complementados con el detalle de las respuestas, indican que la experiencia de usuario es buena, pero puede ser mejorada mediante una inducción a la aplicación.

En pos de cumplir el objetivo general, se plantearon cinco objetivos específicos, los cuales fueron validados mediante una escala Likert en la encuesta final, debido a la dificultad de medirlos cuantitativamente.

Los usuarios participantes estuvieron en su mayoría de acuerdo o muy de acuerdo con el cumplimiento de cada uno de los objetivos específicos. Al normalizar todas sus respuestas a la unidad, se obtiene un valor promedio para cada uno de los objetivos superior al 90 %. Esto quiere decir que, de acuerdo a la apreciación de los usuarios, se puede afirmar con seguridad que los objetivos del proyecto han sido exitosamente cumplidos.

## 8.2. Trabajo futuro

El potencial de crecimiento de este proyecto es ingente, y además puede ser mejorado en distintos aspectos. A continuación se mencionarán algunas de las próximas tareas, las cuales deberán ser evaluadas y priorizadas cuando corresponda. Estas se dividen en los siguientes grupos:

- Puesta en marcha o paso a producción.
- Optimizaciones.
- Otros atributos de calidad no funcionales.
- Herramientas de análisis.
- Nuevas funcionalidades.

Además, resulta oportuno definir un modelo de negocio, el cual mantenga con vida la aplicación realizada.

### 8.2.1. Puesta en marcha

Para una puesta en marcha real, es necesario contar con un servidor, un dominio y un certificado SSL. Resulta conveniente el uso de Docker para asegurar compatibilidad, creando tres contenedores orquestados mediante Docker Compose. Eventualmente podría resultar necesario el uso de un balanceador de carga, si la cantidad de usuarios llegase a ser muy grande. Se sugiere el uso de herramientas de integración continua para facilitar la mantención. Una vez publicado, sería útil registrarlo en los motores de búsqueda más populares.

### 8.2.2. Optimizaciones

Para mejorar la experiencia de usuario es recomendable realizar algunas optimizaciones:

- Optimización automática de imágenes, manteniendo versiones de distintos tamaños y resoluciones.
- Uso de un *ServiceWorker* para mejorar el rendimiento mediante un caché en frontend.

- Transformar el sistema en una Aplicación Web Progresiva (PWA) para permitir el uso de la aplicación sin conexión a Internet.

### 8.2.3. Atributos de calidad no funcionales

También existen otros atributos de calidad no funcionales a considerar:

- Mejoras de accesibilidad para personas con discapacidades, como el uso del atributo *aria* para personas con discapacidad visual.
- Soporte multilenguaje para usuarios no hispanohablantes.
- Documentación y publicación de API para que aplicaciones externas puedan hacer uso de funcionalidades como la publicación de actividades realizadas.
- Optimizaciones SEO, es decir, mejoras de posicionamiento en motores de búsqueda mediante el uso de palabras clave.
- Uso de *Metatags* html, para generar una vista previa al compartir en la web.

### 8.2.4. Herramientas de análisis

Es recomendable el uso de herramientas de análisis para poder tomar decisiones de implementación que respondan al comportamiento de los usuarios. También se pueden realizar tests A/B con facilidad gracias a ellas. Y además pueden funcionar como una herramienta de detección de errores.

### 8.2.5. Nuevas funcionalidades

Finalmente, existe un gran abanico de posibles nuevas funcionalidades, tanto definidas con anterioridad como surgidas durante el desarrollo, entre las que destacan:

- Uso de un calendario en la aplicación.
- Guardar estado físico progresivamente, y mostrar estadísticas en forma de gráficos.
- Entregar sugerencias de actividad y contactos de acuerdo a las características de la persona.

- Tutoriales para principiantes y profesionales con gente bajo su cargo.
- Creación de cuentas especializadas para gimnasios.
- Conexión con APIs de otras aplicaciones como Google Calendar para otorgar flexibilidad de uso.
- Chat.
- Reportes/bloqueos de usuarios y configuraciones de privacidad.
- Creación y publicación de eventos sociales.
- *Pipes* para adaptar textos, como por ejemplo acortar números o nombres muy largos.

# Glosario

## Fitness

**Alimento:** Representación de un alimento real, pudiendo ser básico (no contiene una receta para prepararlo) o preparado (contiene una receta y está compuesto de otros alimentos). Ejemplo de alimento básico: naranja. Ejemplo de alimento preparado: Cheesecake de maracuyá.

**Ejercicio:** Acción motora utilizada para el desarrollo de un músculo esquelético o un grupo muscular. Ejemplo: Sentadilla, peso muerto, dominada supina, entre otros.

**Fitness:** estado generalizado de bienestar y salud física logrado a partir de buenos hábitos alimenticios y la práctica del ejercicio físico.

**Grupo muscular:** Conjunto de músculos relacionados que actúan en conjunto para la realización de un movimiento. Ejemplo: Pectorales, glúteos, dorsales, entre otros.

**Implementación/implemento:** Objetos, herramientas u equipos especializados, empleados para la realización de ejercicios físicos. Ejemplo: mancuerna, cuerda, entre otros.

**Plan alimenticio:** Conjunto de dietas seleccionadas y ordenadas para un periodo de varios días, con un objetivo determinado.

**Plan de ejercicios:** Conjunto de rutinas seleccionadas y ordenadas en un periodo de varios días, con un objetivo determinado. Ejemplo: dos semanas con rutina de torso en días lunes y rutina de piernas en días jueves.

**Plan de entrenamiento:** Rutinas y dietas seleccionadas y ordenadas para un periodo de varios días, es decir, mezcla de plan de ejercicios y plan alimenticio.

**Rutina:** Conjunto de ejercicios seleccionados y ordenados para su ejecución con un objetivo determinado. Ejemplo: rutina para desarrollo muscular de piernas que incluye sentadillas y peso muerto.

## Desarrollo de software

**Acoplamiento:** Interdependencia entre módulos de software.

**Backend:** Parte de una aplicación con la que no interactúa el usuario. Contiene la lógica de negocio.

**Basic authentication:** Autenticación http básica. Agrega usuario y contraseña en la cabecera de un request.

**BBDD:** Bases de datos.

**Bearer authentication:** Autenticación http basada en token.

**Buyer persona:** Arquetipo de cliente ideal.

**Cohesión:** Es la medida en que las partes de una unidad de software pueden trabajar unidas.

**Composite, patrón:** Patrón de diseño estructural en que objetos complejos son construidos recursivamente a partir de otros del mismo tipo o similar.

**CTA:** Concurrent think aloud.

**Diccionario:** Estructura de datos basada en tuplas de tipo (llave, valor).

**Framework:** Esquema, herramienta o estructura con estándares definidos a cumplir, dentro de la cual se puede trabajar.

**Frontend:** Parte de una aplicación con la que interactúa el usuario.

**Hashing:** Proceso de convertir un valor en otro usualmente ininteligible mediante una función de hash.

**Inversión de dependencias, principio de** : Principio SOLID que establece que las clases de alto nivel no deberían depender de las clases de bajo nivel, sino que ambas deberían depender de abstracciones.

**JSON:** JavaScript Object Notation. Es un formato de texto sencillo para el intercambio de datos.

**Microentorno:** Ámbito en que se desenvuelve una empresa o proyecto. Incluye competencia y clientes, entre otros.

**MVP:** Producto viable mínimo.

**MVVM:** Patrón de arquitectura model-view-viewmodel.

**NoSQL:** Bases de datos no relacionales.

**Producto viable mínimo:** Producto con suficientes características para satisfacer a los clientes iniciales, y proporcionar retroalimentación para el desarrollo futuro.

**Programación declarativa:** Paradigma de programación no imperativo.

**Repository, patrón:** Patrón de diseño en que existe una capa encargada de las conexiones a la base de datos.

**Responsabilidad única, principio de:** Principio SOLID que establece que cada módulo o clase debe tener responsabilidad sobre una sola parte de la funcionalidad proporcionada por el software y esta responsabilidad debe estar encapsulada en su totalidad por la clase.

**RP:** Retrospective probing.

**Stack de tecnologías:** Lista de servicios tecnológicos utilizados para construir y ejecutar una aplicación

**TDD:** Test driven development.

**Token:** Cadena alfanumérica usualmente creada por hashing, que suele ser usada para efectos de autenticación.

# Bibliografía


[1] 1. working with spring data repositories. https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html, Consultado el 23 de junio de 2018.

[2] 23. webflux framework. https://docs.spring.io/spring/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/web-reactive.html, Consultado el 23 de junio de 2018.

[3] 5 hormonas que se activan cuando haces ejercicio — salud180. https://www.salud180.com/nutricion-y-ejercicio/5-hormonas-que-se-activan-cuando-haces-ejercicio, Consultado el 26 de septiembre de 2018.

[4] 8 razones para llevar a una dieta saludable — salud180. https://www.salud180.com/nutricion-y-ejercicio/8-razones-para-llevar-una-dieta-saludable, Consultado el 8 de octubre de 2018.

[5] 8fit - fitness y nutrición - apps en google play. https://play.google.com/store/apps/details?id=com.eightfit.app&hl=es_CL, Consultado el 12 de octubre de 2018.

[6] adidas running by runtastic - correr y fitness - apps en google play. https://play.google.com/store/apps/details?id=com.runtastic.android&hl=es_CL, Consultado el 12 de octubre de 2018.

[7] Autenticación con json web tokens - oscar blancarte blog. https://www.oscarblancarteblog.com/2017/06/08/autenticacion-con-json-web-tokens/, Consultado el 20 de junio de 2018.

[8] Client-server architecture — computer science — britannica.com. https://www.britannica.com/technology/client-server-architecture, Consultado el 20 de junio de 2018.

[9] Defining stateful vs stateless web services — nordic apis. https://nordicapis.com/defining-stateful-vs-stateless-web-services/, Consultado el 7 de agosto de 2018.

[10] Desarrollador web: Front-end, back-end y full stack. ¿quién es quién? https://www.campusmvp.es/recursos/post/Desarrollador-web-Front-end-back-end-y-full-stack-Quien-es-quien.aspx, Consultado el 7 de agosto de 2018.

[11] Dopamina: 7 funciones esenciales de este neurotransmisor. https://psicologiaymente.com/neurociencias/dopamina-neurotransmisor, Consultado el 25 de septiembre de 2018.

[12] Ejercicios en casa - sin equipo - apps en google play. https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment&hl=es_CL, Consultado el 12 de octubre de 2018.

[13] El "cuarteto de la felicidad": cómo desatar los efectos positivos de la endorfina, serotonina, dopamina y oxitocina. https://www.bbc.com/mundo/noticias-39333917, Consultado el 25 de septiembre de 2018.

[14] El ejercicio físico realza la belleza. https://www.lavanguardia.com/estilos-de-vida/20140103/54398581039/el-ejercicio-fisico-realza-la-belleza.html, Consultado el 26 de septiembre de 2018.

[15] Endorfinas (neurotransmisores): funciones y características. https://psicologiaymente.com/neurociencias/endorfinas-neurotransmisores, Consultado el 25 de septiembre de 2018.

[16] Faq — mongodb. https://www.mongodb.com/faq, Consultado el 23 de junio de 2018.

[17] Fitness - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Fitness, Consultado el 28 de septiembre de 2018.

[18] How single-page applications work – paul sherman – medium. https://medium.com/@pshrmn/demystifying-single-page-applications-3068d0555d46, Consultado el 15 de junio de 2018.

[19] Interfaz de programación de aplicaciones - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Interfaz_de_programación_de_aplicaciones, Consultado el 9 de octubre de 2018.

[20] Introducción a json web tokens (jwt). https://platzi.com/blog/introduccion-json-web-tokens/, Consultado el 20 de junio de 2018.

[21] Jefit: Gym trainer, diario gimnasio, ejercicios - apps en google play. https://play.google.com/store/apps/details?id=je.fit&hl=es_CL, Consultado el 12 de octubre de 2018.

[22] Las 5 principales ventajas de usar angular para crear aplicaciones web. https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx, Consultado el 23 de junio de 2018.

[23] Madbarz - bodyweight workouts - apps en google play. https://play.google.com/store/apps/details?id=com.madbarz.madbarzApp&hl=es_CL, Consultado el 12 de octubre de 2018.

[24] Manifiesto por el desarrollo Ágil de software. http://agilemanifesto.org/iso/es/manifesto.html, Consultado el 24 de junio de 2018.

[25] Mongodb architecture — mongodb — mongodb. https://www.mongodb.com/mongodb-architecture, Consultado el 23 de junio de 2018.

[26] Physical activity may strengthen children's ability to pay attention — illinois. https://news.illinois.edu/view/6367/205988, Consultado el 29 de mayo de 2018.

[27] Principios del manifiesto Ágil. http://agilemanifesto.org/iso/es/principles.html, Consultado el 24 de junio de 2018.

[28] Programación reactiva con spring webflux - parte 1. https://windoctor7.github.io/Programacion-Reactiva-Spring5.html, Consultado el 23 de junio de 2018.

[29] Qué es kanban: Fundamentos — kanbanize. https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban/, Consultado el 15 de mayo de 2019.

[30] Qué es un test de usabilidad y porqué debería aplicarlo a mi sitio web. https://www.hostgator.mx/blog/que-es-un-test-de-usabilidad/, Consultado el 2 de agosto de 2018.

[31] Qué es un unit test. https://msdn.microsoft.com/es-es/communitydocs/alm/unit-test, Consultado el 1 de agosto de 2018.

[32] The reactive manifesto. https://www.reactivemanifesto.org/, Consultado el 15 de junio de 2018.

[33] Scrum, desarrollo ágil por excelencia — nateevo. http://www.nateevo.com/scrum-la-metodologia-de-desarrollo-agil-por-excelencia/, Consultado el 24 de junio de 2018.

[34] Scrum: roles y responsabilidades — deloitte españa. https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html, Consultado el 24 de junio de 2018.

[35] Serotonina: 6 efectos de esta hormona en tu cuerpo y mente. https://psicologiaymente.com/neurociencias/serotonina-hormona, Consultado el 25 de septiembre de 2018.

[36] Spring. https://spring.io/, Consultado el 23 de junio de 2018.

[37] Spring boot performance battle: blocking vs non-blocking vs reactive. https://medium.com/@filia.aleks/microservice-performance-battle-spring-mvc-vs-webflux-80d39fd81bf0, Consultado el 12 de junio de 2020.

[38] Spring data. https://spring.io/projects/spring-dataoverview, Consultado el 23 de junio de 2018.

[39] Spring data mongodb - reference documentation. https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/, Consultado el 23 de junio de 2018.

[40] Spring security. https://spring.io/projects/spring-security, Consultado el 23 de junio de 2018.

[41] Spring security reference. https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/, Consultado el 23 de junio de 2018.

[42] Street workouts calistenia : Entrenador fitness - aplicaciones en google play. https://play.google.com/store/apps/details?id=com.jleoapps.calistenia, Consultado el 12 de octubre de 2018.

[43] Technutri - dieta cetogénica para perder peso - apps en google play. https://play.google.com/store/apps/details?id=br.com.tecnonutri.app&hl=es_CLL, Consultado el 12 de julio de 2020.

[44] Test de usabilidad: qué es y por qué debes hacerlo - agencia waka. https://www.somoswaka.com/blog/2018/01/test-de-usabilidad/, Consultado el 2 de agosto de 2018.

[45] testing - what should you test with unit tests? - software engineering stack exchange. https://softwareengineering.stackexchange.com/questions/750/what-should-you-test-with-unit-tests, Consultado el 12 de junio de 2019.

[46] Thenics - apps en google play. https://play.google.com/store/apps/details?id=com.abad.thenics&hl=es_CL, Consultado el 12 de octubre de 2018.

[47] Web on reactive stack. https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.htmlspring-webflux, Consultado el 23 de junio de 2018.

[48] What is acid (atomicity, consistency, isolation, and durability)? - essential sql. https://www.essentialsql.com/what-is-meant-by-acid/, Consultado el 23 de junio de 2018.

[49] What is black box testing? techniques, example & types. https://www.guru99.com/black-box-testing.html, Consultado el 1 de agosto de 2018.

[50] What is mongodb? — mongodb. https://www.mongodb.com/what-is-mongodb, Consultado el 23 de junio de 2018.

[51] What is reactive programming? – redelastic. https://blog.redelastic.com/what-is-reactive-programming-bc9fa7f4a7fc, Consultado el 15 de junio de 2018.

[52] What is restful api? - definition from whatis.com. https://searchmicroservices.techtarget.com/definition/RESTful-API, Consultado el 9 de octubre de 2018.

[53] ¿pruebas de integración, funcionales, de carga...? ¿qué diferencias hay? — seidor technologies. https://www.seidortechnologies.cl/tipos-de-pruebas/, Consultado el 2 de agosto de 2018.

[54] ¿qué es la programación reactiva? una introducción 8211; consultoría y servicios it para empresas — profile software services. https://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/, Consultado el 15 de junio de 2018.

[55] ¿qué es un tablero kanban? definición y detalles — kanbanize. https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban/, Consultado el 15 de mayo de 2019.

[56] ¿qué es una api rest? - idento. https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/, Consultado el 9 de octubre de 2018.

[57] ¿qué es una aplicación web? - blog neosoft sistemas. https://www.neosoft.es/blog/que-es-una-aplicacion-web/, Consultado el 7 de agosto de 2018.

[58] The 2018 ihrsa global report. https://www.ihrsa.org/improve-your-club/industry-news/latin-america-news-gyms-expand-in-uruguay-colombia-chile/, 2018.

[59] Carmichael A. Anderson, D. J. Essential kanban condensed. *Blue Hole Press.*, 2016.

[60] Dane Bertram. Likert Scales. *CPSC 681 – Topic Report*, 2007.

[61] John Brooke. A “quick and dirty” usability scale. 1996.

[62] Psicólogos en Línea. Autoestima total v1.0. http://www.shalon.edu.ec/aula/pluginfile.php/900/mod_resource/content/1/autoestimatotal.pdf, 2012.

[63] Jonathan F. Leclerc Christine Andrews Paulsen Julie H. Birns, Kristen A. Joffre. Collecting Usability Data Using Two Methods—Concurrent Think Aloud and Retrospective Probing. *Eleventh Anual Meeting of the Usability Professional's Association*, 2002.

[64] James R. Lewis and Jeff Sauro. The factor structure of the system usability scale. 2009.

[65] Elena Miró Ana I. Sánchez Manuel G. Jiménez, Pilar Martínez. Bienestar psicológico y hábitos saludables: ¿están asociados a la práctica de ejercicio físico? *Universidad de Granada, España*, 2007.

[66] Maaike J. van den Haak and Memo D.T. de Jong. Exploring Two Methods of Usability Testing: Concurrent versus Retrospective Think-Aloud Protocols. *Professional Communication Conference, IPCC 2003. Proceedings. IEEE International*, pages 285–287, 2003.

# ANEXOS

# A. Encuesta requisitos

A continuación se presentan los resultados obtenidos de la encuesta realizada para la definición de requisitos. El objetivo era priorizar los requisitos presentados y obtener otros nuevos. En esta encuesta participaron 60 personas, una de las cuales fue descartada por respuestas atípicas y con falta de coherencia.

Por motivos de transparencia las respuestas textuales se han mantenido tal como fueron escritas, agregando solo la mayúscula inicial y el punto final según fuera necesario.

## A.1. Información general

En esta sección se agregaron parámetros que permitieran clasificar a los participantes. Tras aplicar filtros a la información no se detectaron diferencias sustanciales entre los distintos grupos, pero fue de utilidad para conocer a los potenciales interesados. Se consideraron tres factores: edad (figura A.1), género (figura A.2) y relación con el fitness (figura A.3).

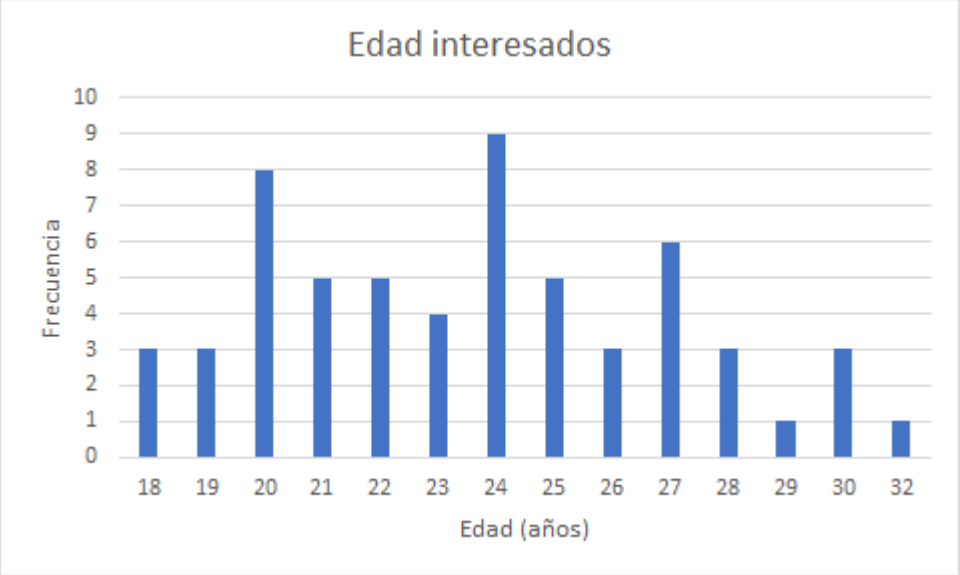


Figura A.1: Encuesta requisitos: edad

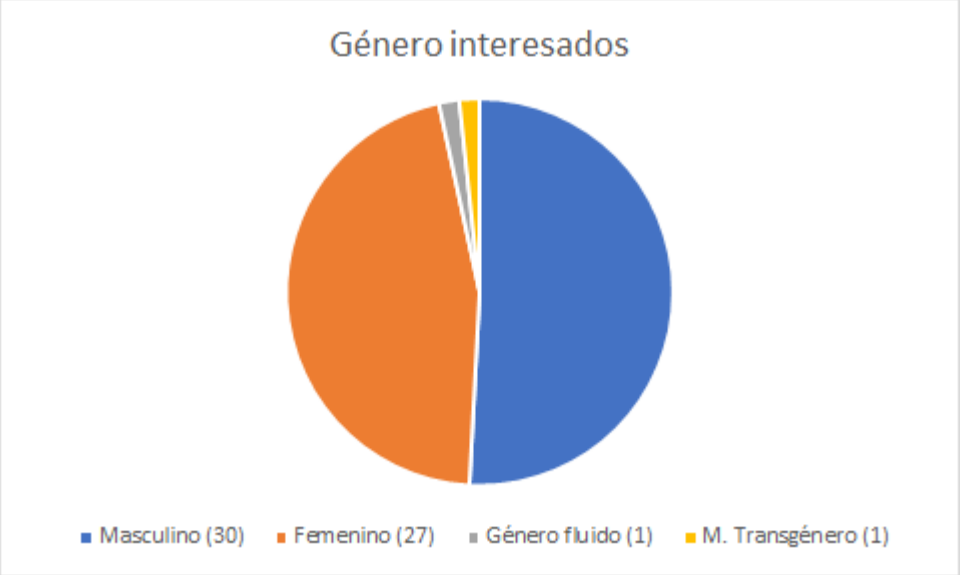


Figura A.2: Encuesta requisitos: género

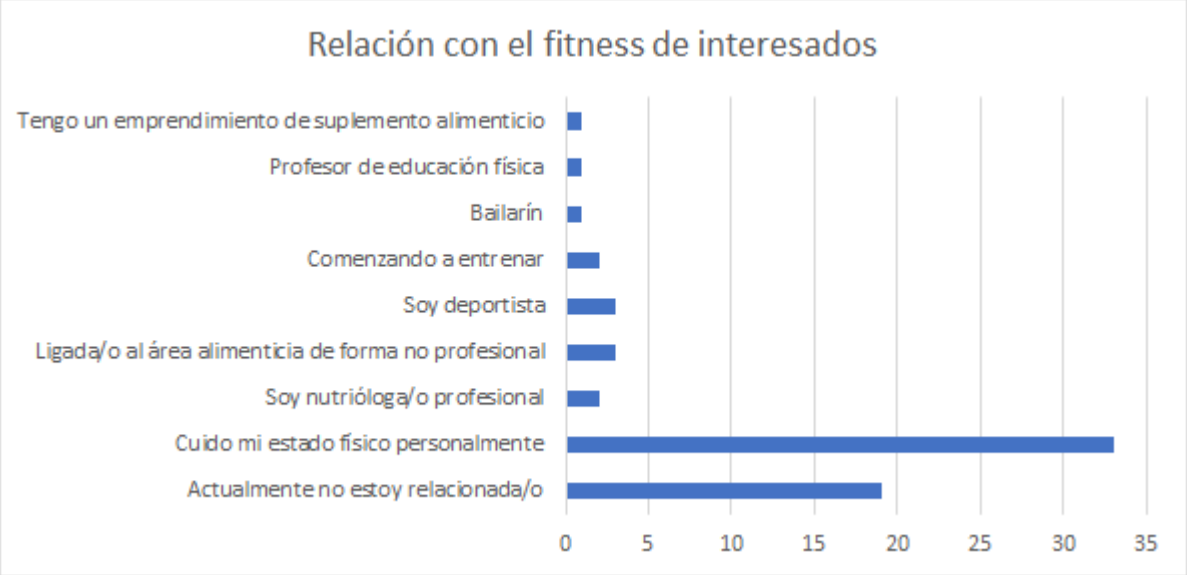


Figura A.3: Encuesta requisitos: relación con el fitness

## A.2. Ejercicios

En este apartado se preguntó por el nivel de actividad física (figura A.4).

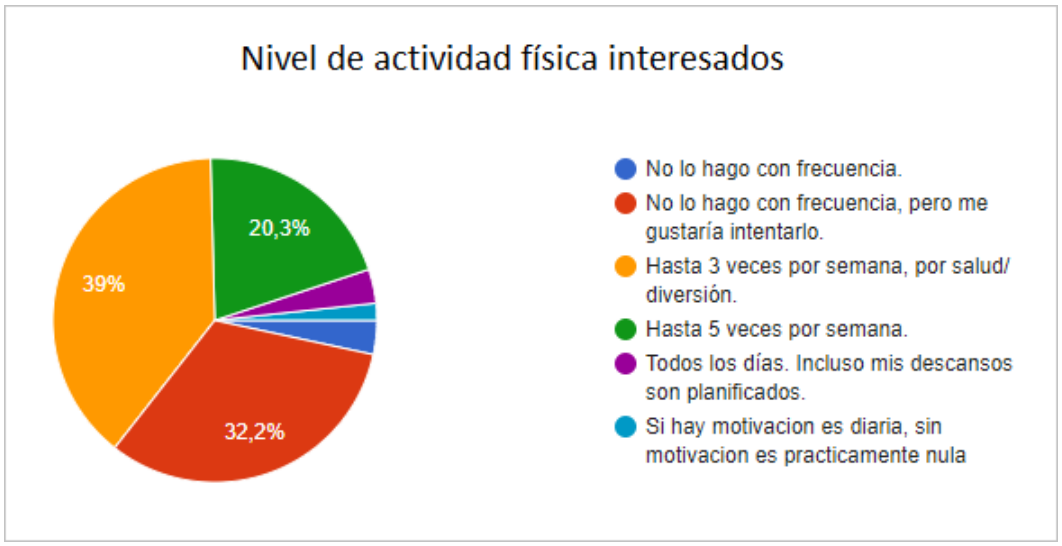


Figura A.4: Encuesta requisitos: nivel de actividad física

En segundo lugar se pidió asignar un puntaje (figura A.5) a las características presentadas según la siguiente escala:

0. ¡Elimínalo!
1. Irrelevante.
2. Prescindible.
3. Importante.
4. Muy importante.
5. Imprescindible.

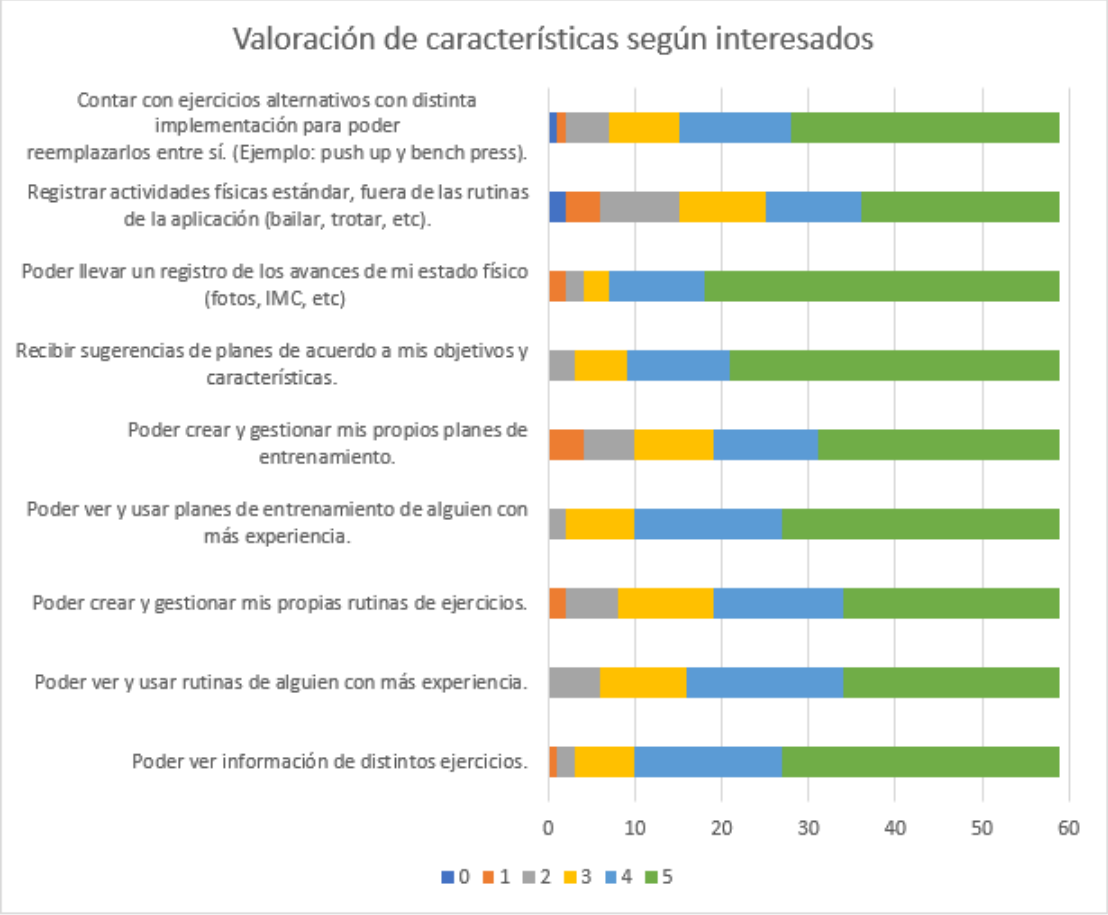


Figura A.5: Encuesta requisitos: Valoración características de Ejercicios.

Finalmente, se dejó un espacio abierto a sugerencias, del cual se obtuvieron las siguientes propuestas (sic):

- Vídeos que muestran al menos 3 o 4 alternativas para ejercitar un grupo muscular específico, sugerencias de tiempos de pausa entre una sesión sesión de ejercicios y entre repeticiones de la sesión, estructuración de ejercicios en una sesión para trabajar un grupo muscular ejem: 8 repeticiones de press banca al 80 % luego inmediatamente hacer 6 flexiones de brazos con aplauso, luego pausa de 20seg y repetir por 4 o 5 veces.

- Cantidad de calorias quemadas por minuto de tal X ejercicio, por ejm realizar saltos de cuerda por un minuto(100 saltos x min) que man 24 calorias por ejemplo. Graficos de progreso, tendencias de pronóstico en los gráficos, recomendaciones si disminuyen los progresos, como subida de pesos inesperadas, las rutina/nutrición empleada no funciona y que ofrezca ejercicios alternativos, etc.

- Estadísticas de tiempo dedicado a cada actividad. Actualmente no es frecuente encontrar app que te indiquen el tiempo que dedicas a cada ejercicio (específicamente al levantamiento de pesas), podría ser un factor de control que aportaría a controlar la sobre ejercitación por fatiga, o a ver si realmente estás dedicando mucho tiempo a un músculo y descuidas otro.

- Un apartado donde poder encontrar mas informacion de los ejercicios, sitios para los instrumentos necesarios o para las recomendaciones, ademas del añadido de alerta de hacer demasiado ejercicio para evitar desmayos o daños fisicos.

- Contar con una función de video, donde puedas ver cómo se realiza el ejercicio y para cuándo lo estés realizando, los demás usuarios puedan decir que estás haciendo bien o mal.

- Agregar instrucciones para aquellos que quieren iniciar a practicar deporte, pero desconocen del tema.

- Sugerencias de alimentación para distintos tipos de entrenamiento, recuperación muscular, etc.

- Ejercicios que se puedan hacer en casa en un tiempo limitado por ejemplo 30 minutos.

- Que te motive o recuerde hacer los ejercicios según cada plan de ejercicios.

- Tener la posibilidad de hacer ejercicio en casa y con pocos implementos.
- Crear una rutina o plan de ejercicios compartido entre mis amigos.
- Rutinas semanales para trabajar todp el cuerpo en dias diferentes.
- Que monitoree señales de pulso, etc.
- Musculatura implicada en hacer el ejercicio.
- Planificación específica para cada deporte.
- Ejercicios que se pueden realizar en casa.
- Yoga y calistenia.

## A.3. Alimentación

En este apartado se preguntó por el cuidado de la alimentación (figura A.6).

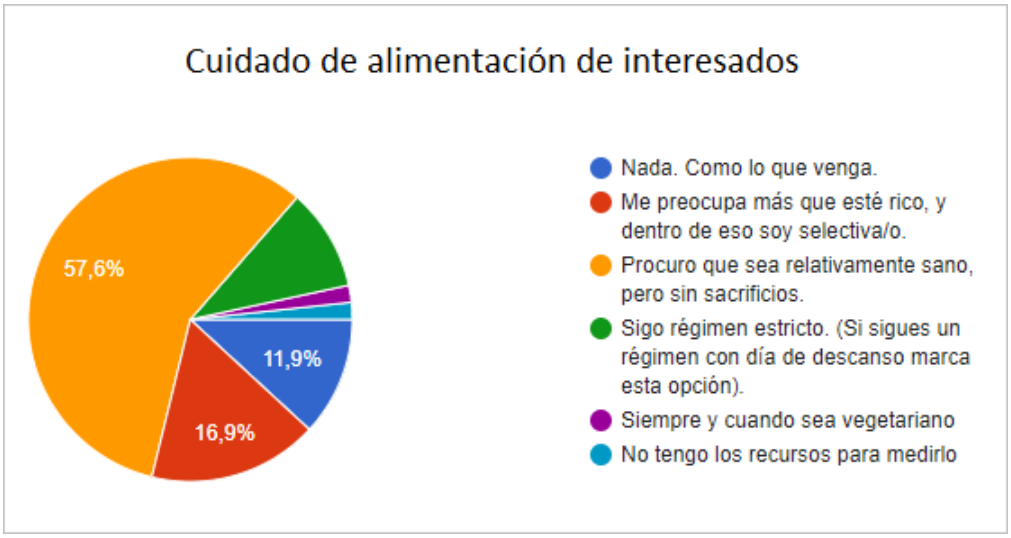


Figura A.6: Encuesta requisitos: Cuidado de la alimentación.

En segundo lugar se pidió asignar un puntaje (figura A.7) a las características presentadas según la siguiente escala:

0. ¡Elimínalo!

1. Irrelevante.
2. Prescindible.
3. Importante.
4. Muy importante.
5. Imprescindible.

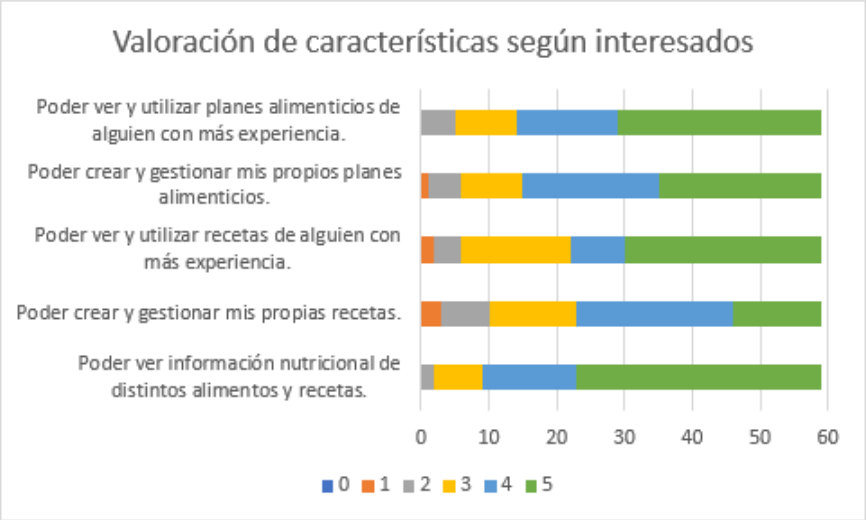


Figura A.7: Encuesta requisitos: Valoración características de Alimentos.

Finalmente, se dejó un espacio abierto a sugerencias, del cual se obtuvieron las siguientes propuestas (sic):

- Calendario semanal como un horario donde pueda ingresar qué comí al desayuno, merienda, almuerzo, colación tarde, cena, pre y post entreno y tipos de horario, por ejm normal(5 comidas), para entrenamientos hypertrofia(5 comidas, más pre y post entreno), aportes en proteínas, grasas, carbohidratos, kcal, si con lo que voy consumiendo llego a mis requerimientos de kcal diarios o estoy en déficit calórico, por ejemplo.
- Evaluar si el plan de alimentación creado por uno mismo está dentro de la posibilidad de que sea adecuado, debido a que con el tiempo podría ser un usuario experto y su dieta no sea la adecuada para el resto. En resumen validar la receta creada por un experto en el área.

- Poder ver información sobre locales/supermercado/restorants donde pueda encontrar tipos específicos de comida saludable. Por ejemplo, un dato de un local donde comprar un tipo determiando de quinoa o productos vegetarianos.
- Contadores de kilocalorías, grasas, proteínas y que estas sean mostrada en una comparación con un aproximado de las gastadas con el total de el ejercicio que fue realizado, para poder complementarlo.
- Básicamente hablar de la calidad del alimento. Por ejemplo, ¿Queso fresco es sano? El artesanal se hace en base de leche entera. ¿La parte negra del pescado se come? Ahí se concentran lo ácidos grasos Omega.
- Recetas accesibles para todo tipo de bolsillo, ya que existen planes nutricionales fuera del alcance de varios personas por temas económicos.
- Información sobre cada alimento, su interacción con el organismo, beneficios; de igual forma con los suplementos alimenticios.
- Estadísticas por aporte nutricional del alimento, así como yazio, donde puedes agregar la info nutricional de lo que comes.
- Asociar costos a distintos tipos de recetas, encontrar alternativas naturales, recetas mas baratas, etc...
- Que los alimentos estén relacionados a la zona donde me encuentre e idealmente de bajo costo.
- Evaluar el Estado nutricional periódicamente y el seguimiento del plan de alimentación.
- Recomendaciones de planes alimenticios dependiendo del objetivo a lograr.

## A.4. Ámbito social

En este apartado se pidió asignar un puntaje (figura A.8) a las características presentadas según la siguiente escala:

0. ¡Elimínalo!

1. Irrelevante.
2. Prescindible.
3. Importante.
4. Muy importante.
5. Imprescindible.

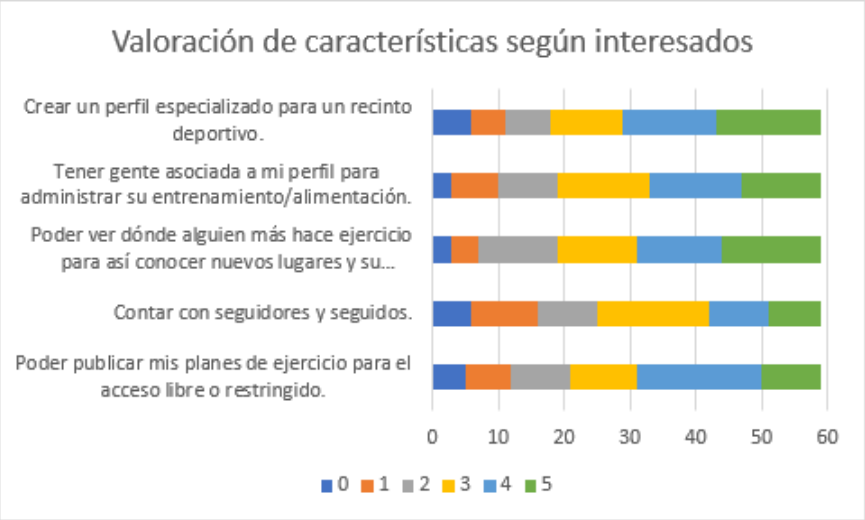


Figura A.8: Encuesta requisitos: Características de Social.

Finalmente, se dejó un espacio abierto a sugerencias, del cual se obtuvieron las siguientes propuestas (sic):

- Dar consejos nutricionales asociados al ámbito casero.
- Tipos de cuentas: Cuentas de personas, que permita crear cuentas de grupos asociar cuentas de marcas, asociar cuentas de grupos(crear grupos), cuentas de gimnasios, clubes deportivos generales y específicos con emojis de tipos de deportes que se imparten en estos clubes/gymnasios, las cuentas comerciales que permitan poder asignarles horarios, telefonos, web, etc. Que estas cuentas permitan ser evaliadas con puntuaciones como play store y en vez de número de descargas diga número de usuarios que usaron ese gym/club v/s puntuación. Y lo tipico poder publicar en otras cuentas instagram, facebook, twitter, Snapchat, twitcht, etc. Eso xd

- Que la administración del gimnasio o entrenador tenga rutinas o planes de ejercicios guardados para él y que pueda asignarlos a un usuario.
- Un chat, de esta manera conocer mas personas, un sistema de niveles que aumente segun los ejercicios, el nivel de dificultad, el nivel que ayuda a los demas en sus planes, la responsabilidad; y la posibilidad de reportar a cierta persona/cuenta, para tener encuenta si se trata de una persona que podria dañar la comunidad.
- Importante incluir planes con objetivos puntuales en el caso de deportivas con alguna especialidad, sobre todo en atletismo si fuera el caso.
- Por temas de seguridad de las personas considero que saber en donde se encuentran podria producirse para acceso al acoso, sicopatía, entre otros. Si seria bueno decir lugares donde serian buenos para hacer deporte como sugerencia, pero que quede en el anonimato. A no ser que sea algun contacto de amistad en la red social.
- Los recintos deberían tener sus tarifas en público y claras, de modo de hacer un benchmarking con tranquilidad.
- Foros, poder añadir eventos, como organizar un partido, o entrenamiento grupal.

# B. Aplicación terminada

A continuación se presentan algunas capturas de la aplicación funcionando, en pantallas de distinto tamaño según corresponda. Se obvian componentes/partes de menor importancia y/o repetitivas, tales como diálogos de confirmación.


Figura B.1: Aplicación terminada: Página de inicio tamaño grande.


Figura B.2: Aplicación terminada: Página de inicio tamaño mediano.

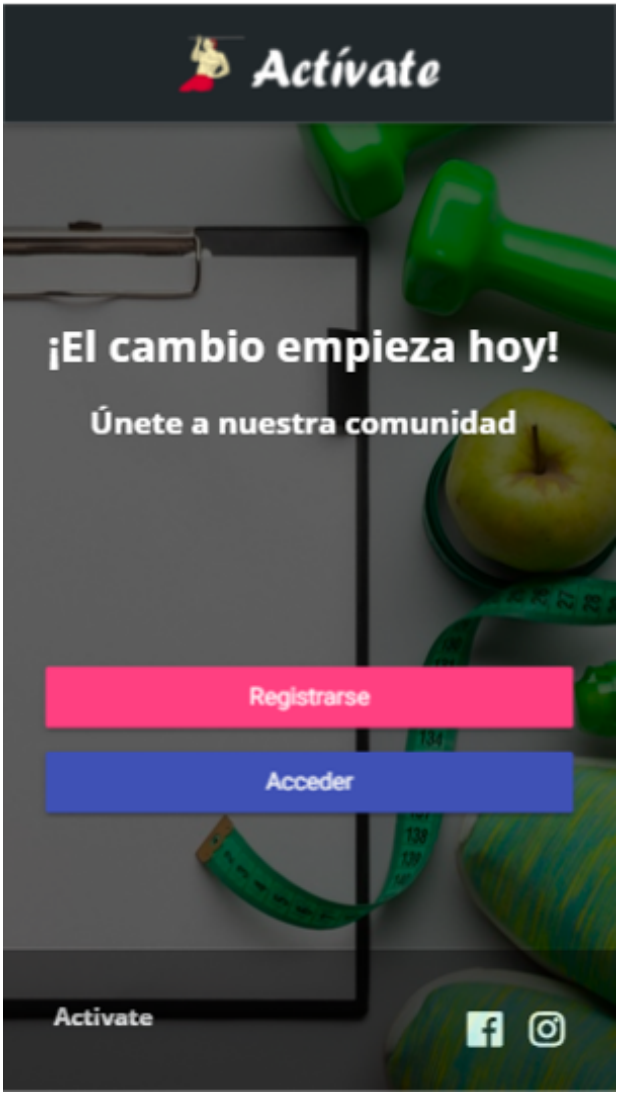


Figura B.3: Aplicación terminada: Página de inicio tamaño pequeño.

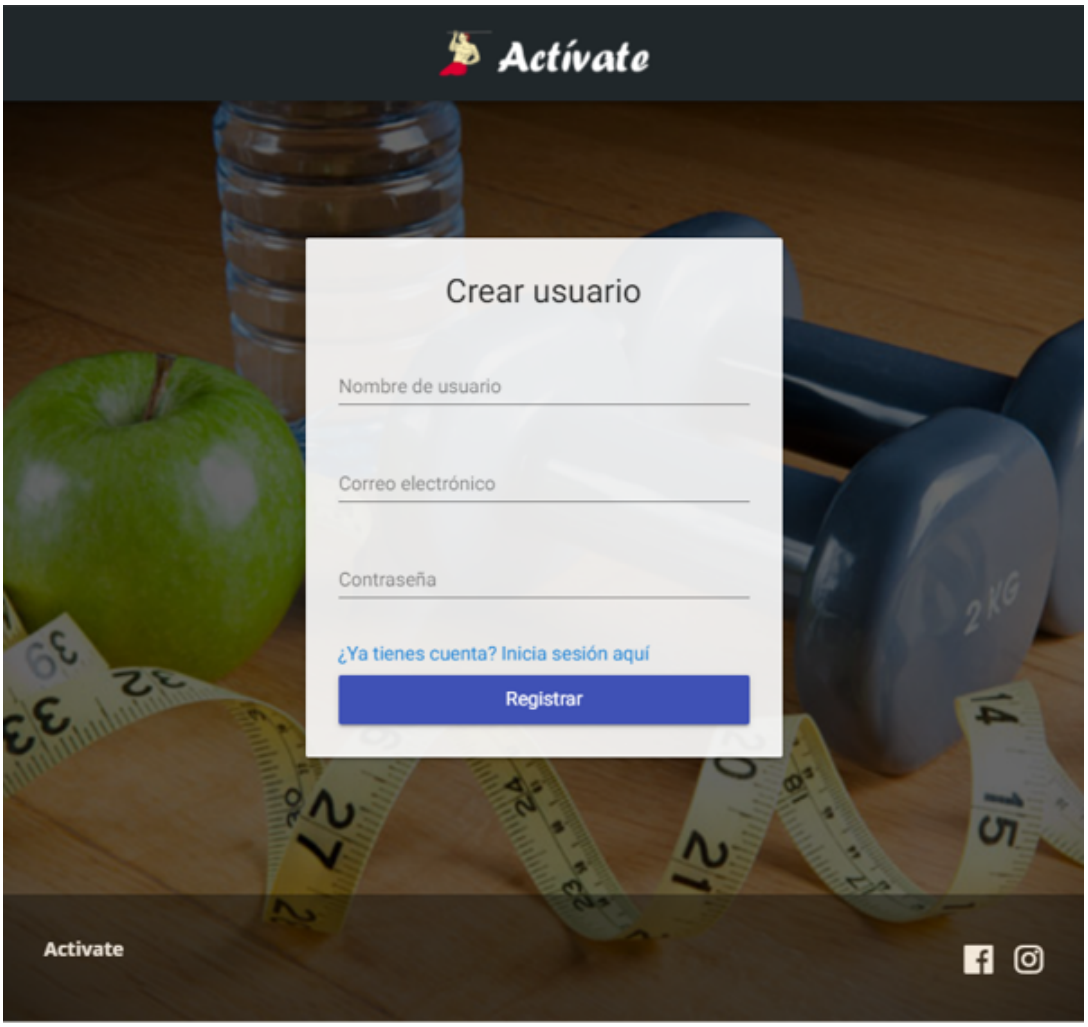


Figura B.4: Aplicación terminada: Crear nuevo usuario.

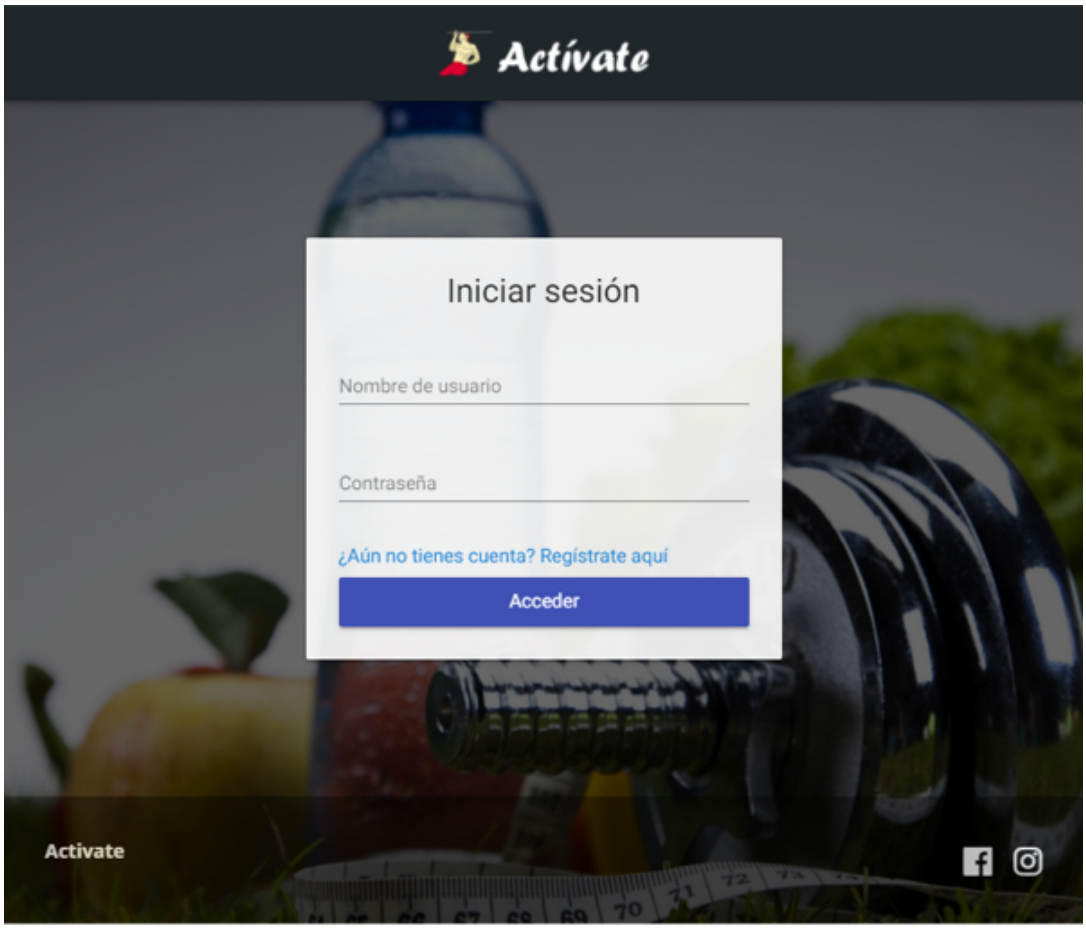


Figura B.5: Aplicación terminada: Iniciar sesión.

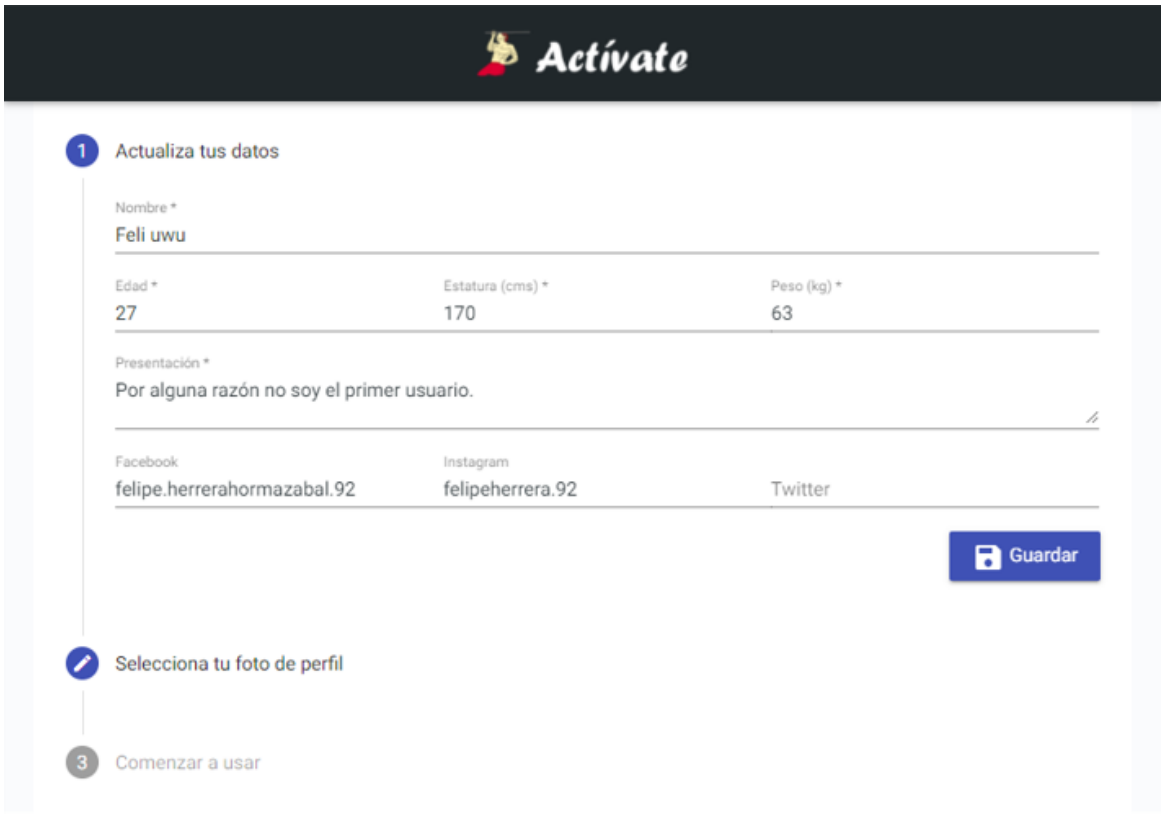


Figura B.6: Aplicación terminada: Formulario nuevo usuario.

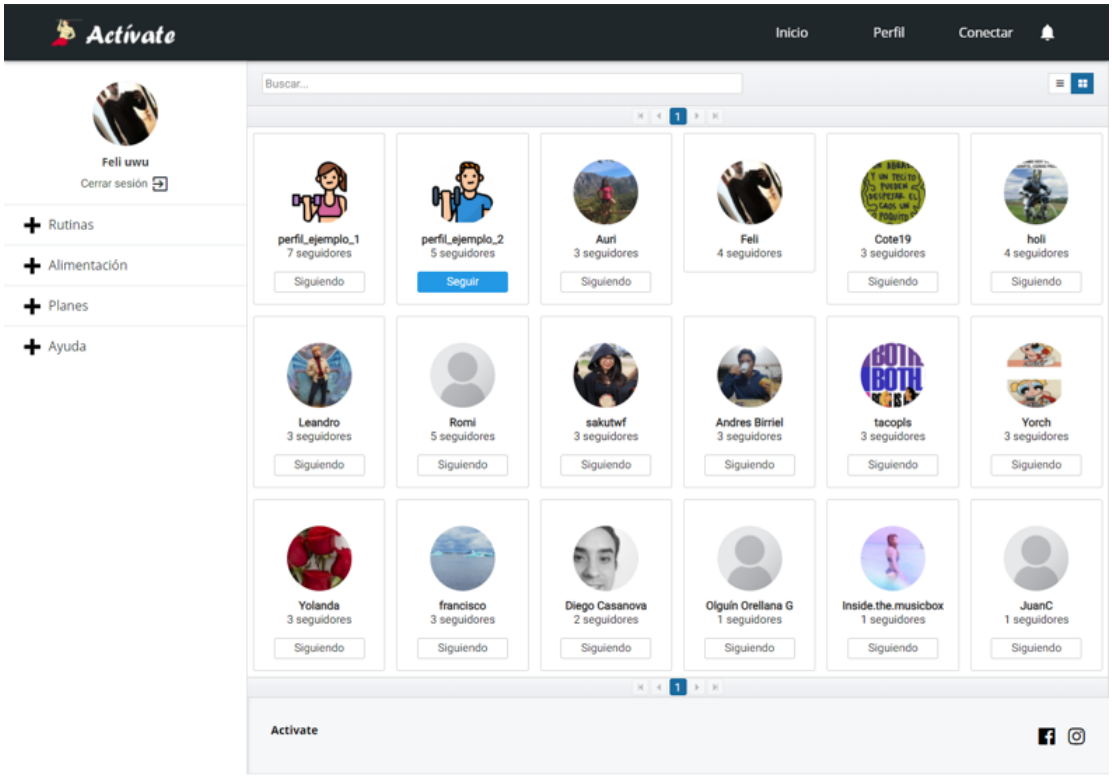


Figura B.7: Aplicación terminada: “Conectar” en pantalla grande.

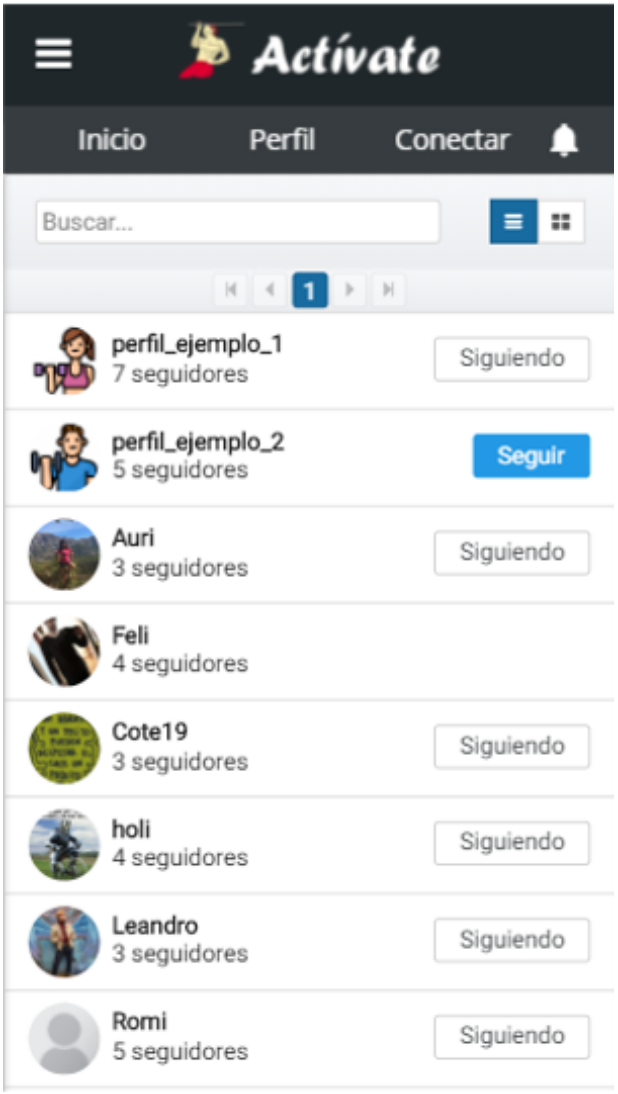


Figura B.8: Aplicación terminada: "Conectar" en pantalla pequeña.

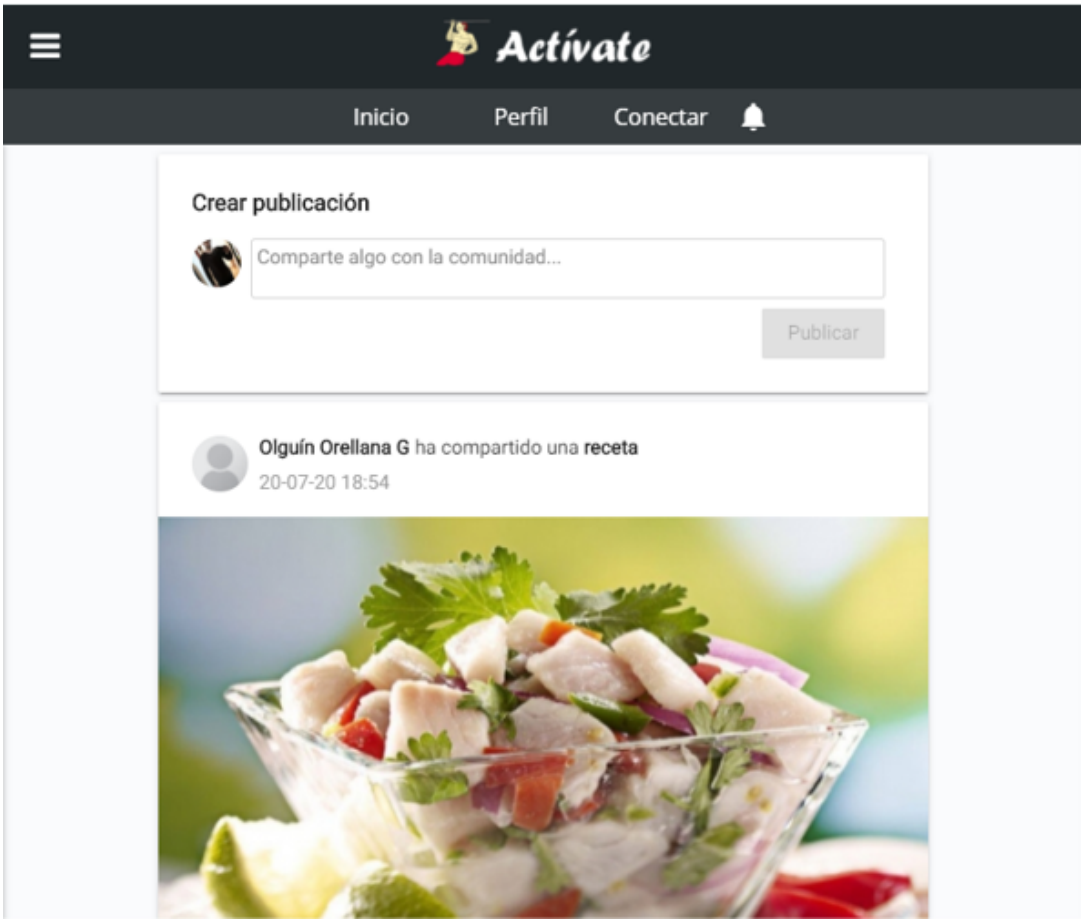


Figura B.9: Aplicación terminada: “Inicio” en pantalla mediana.

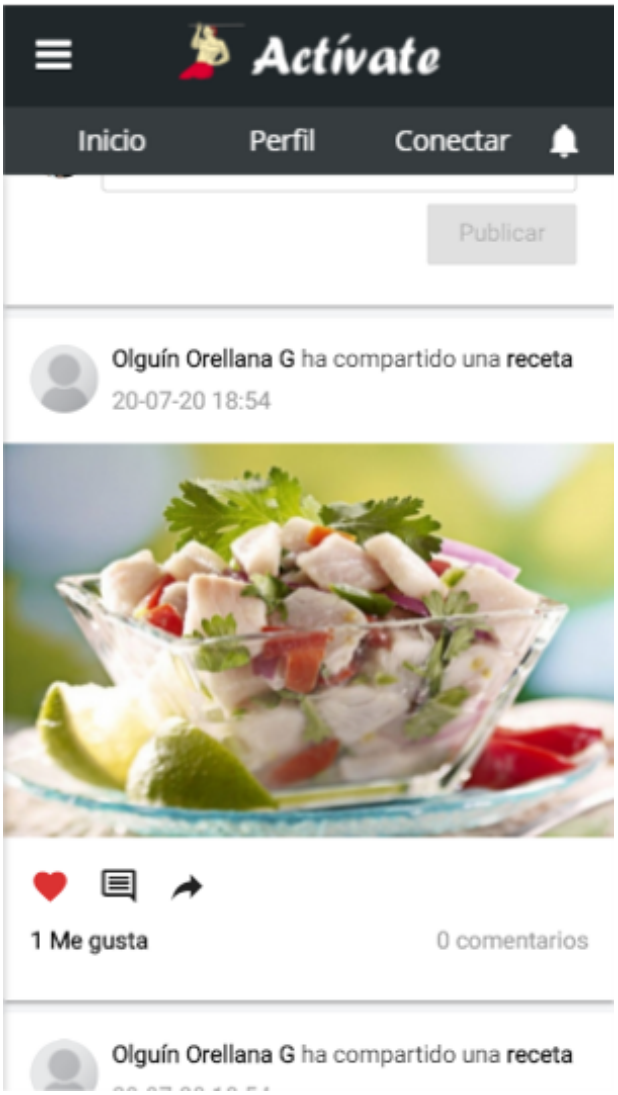


Figura B.10: Aplicación terminada: “Inicio” en pantalla pequeña.

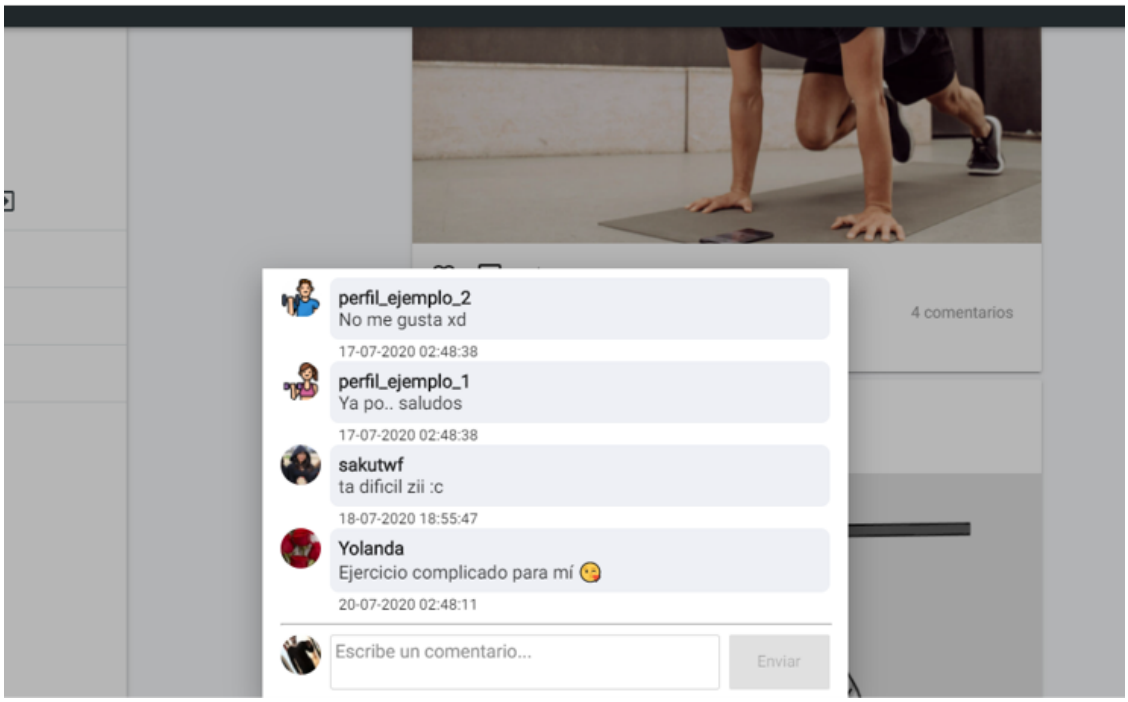


Figura B.11: Aplicación terminada: Comentarios en "Inicio" pantalla grande.

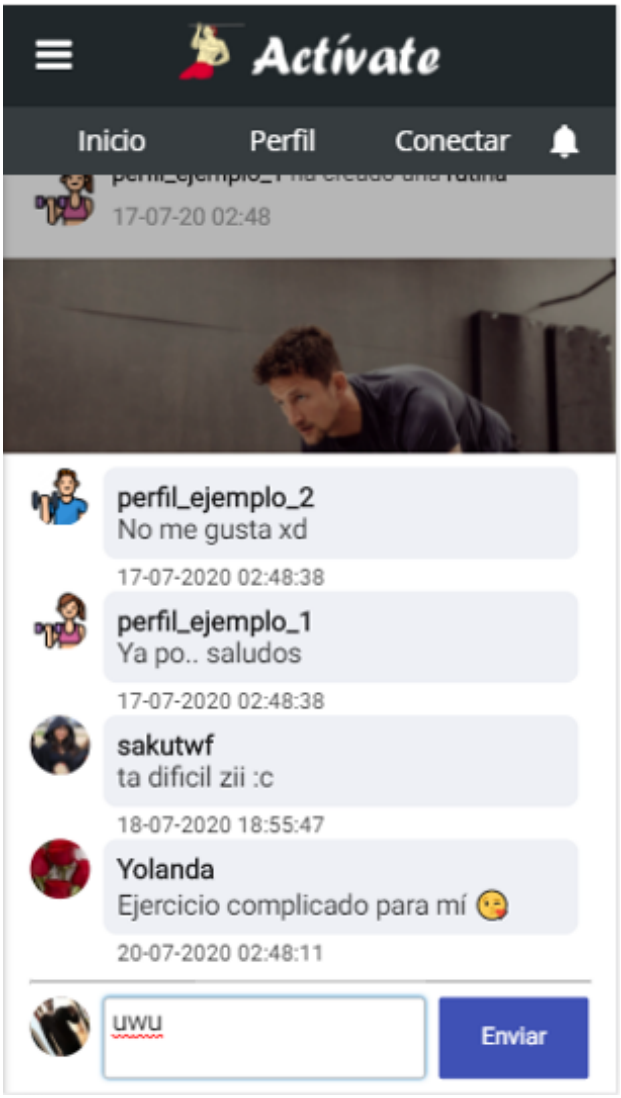


Figura B.12: Aplicación terminada: Comentarios en “Inicio” pantalla pequeña.

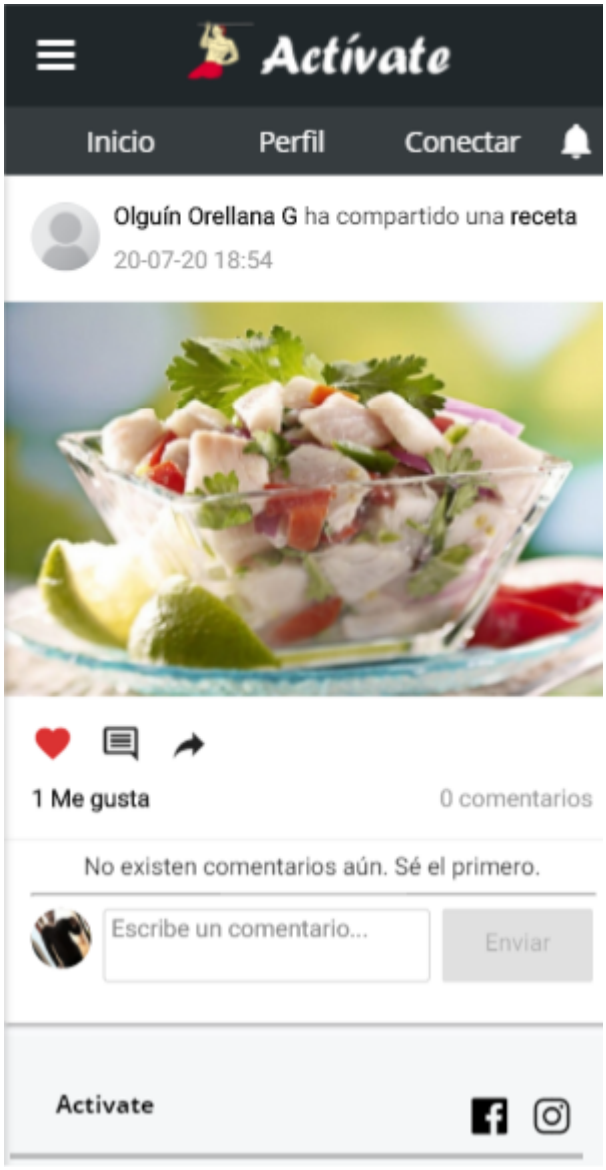


Figura B.13: Aplicación terminada: Detalle de un post.

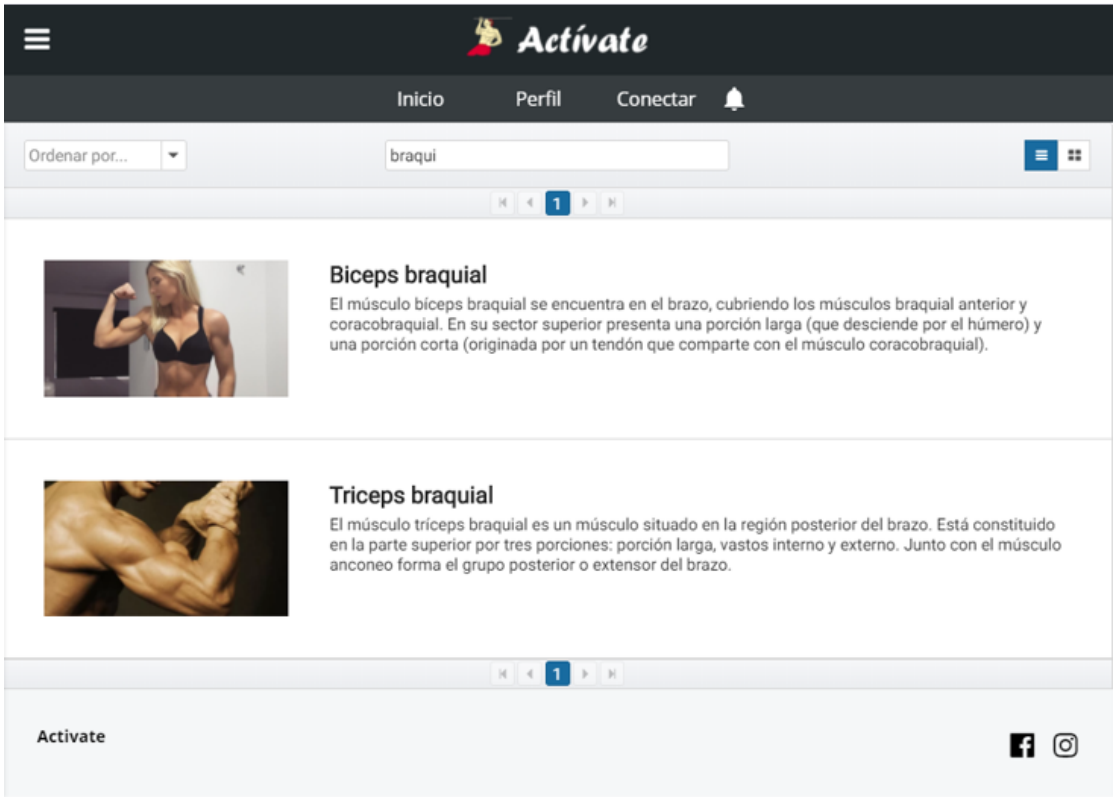


Figura B.14: Aplicación terminada: Lista de músculos pantalla mediana.

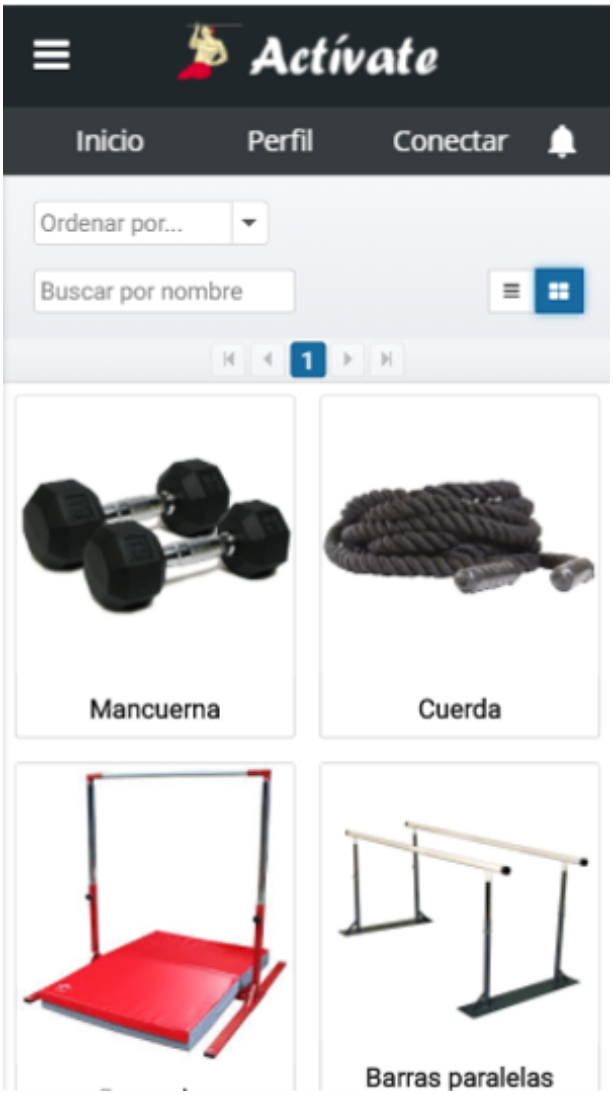


Figura B.15: Aplicación terminada: Lista de implementos en pantalla pequeña.


Figura B.16: Aplicación terminada: Detalle de músculo en pantalla mediana.

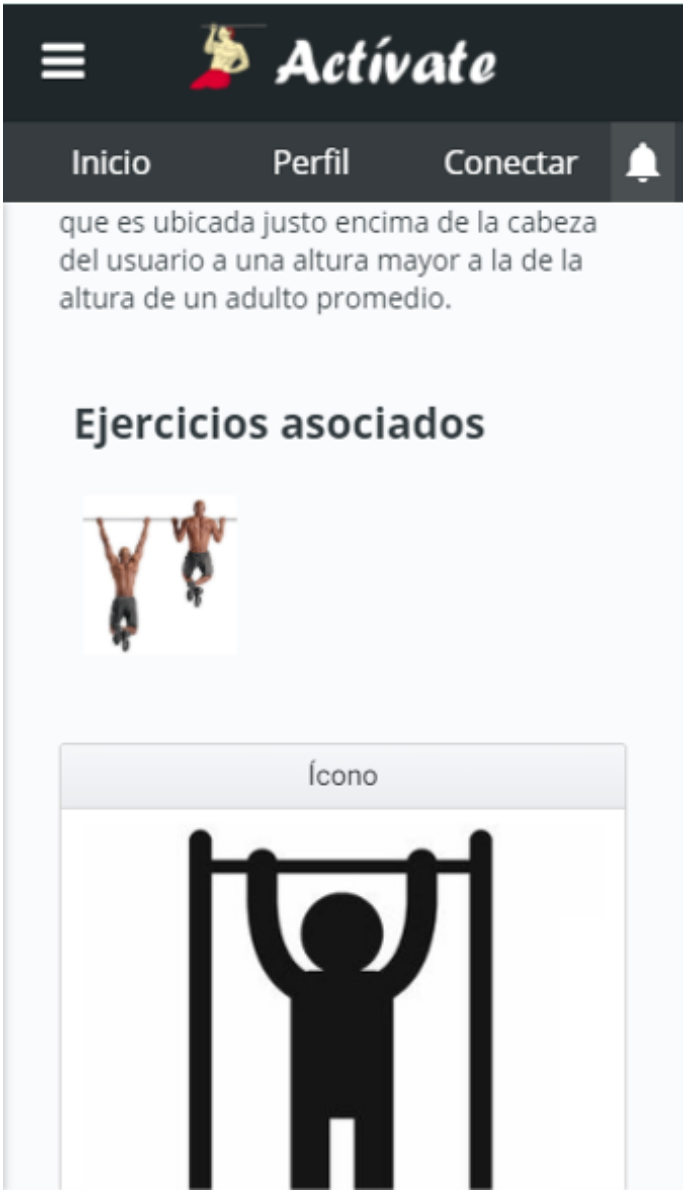


Figura B.17: Aplicación terminada: Detalle de implemento en pantalla pequeña.

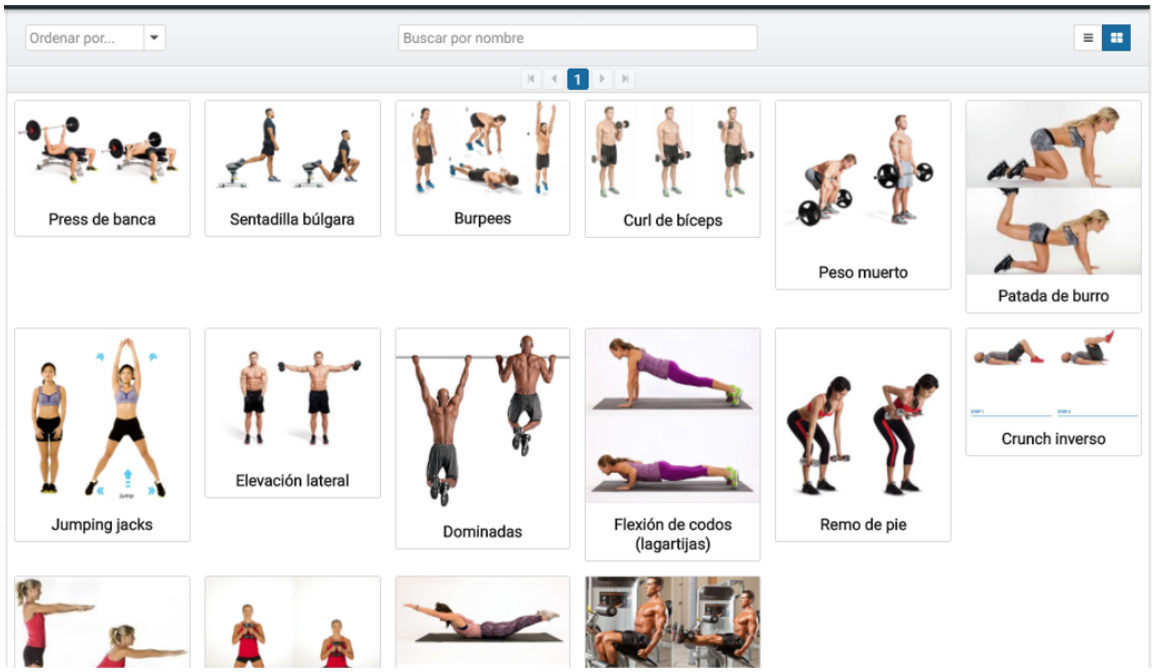


Figura B.18: Aplicación terminada: Lista de ejercicios.

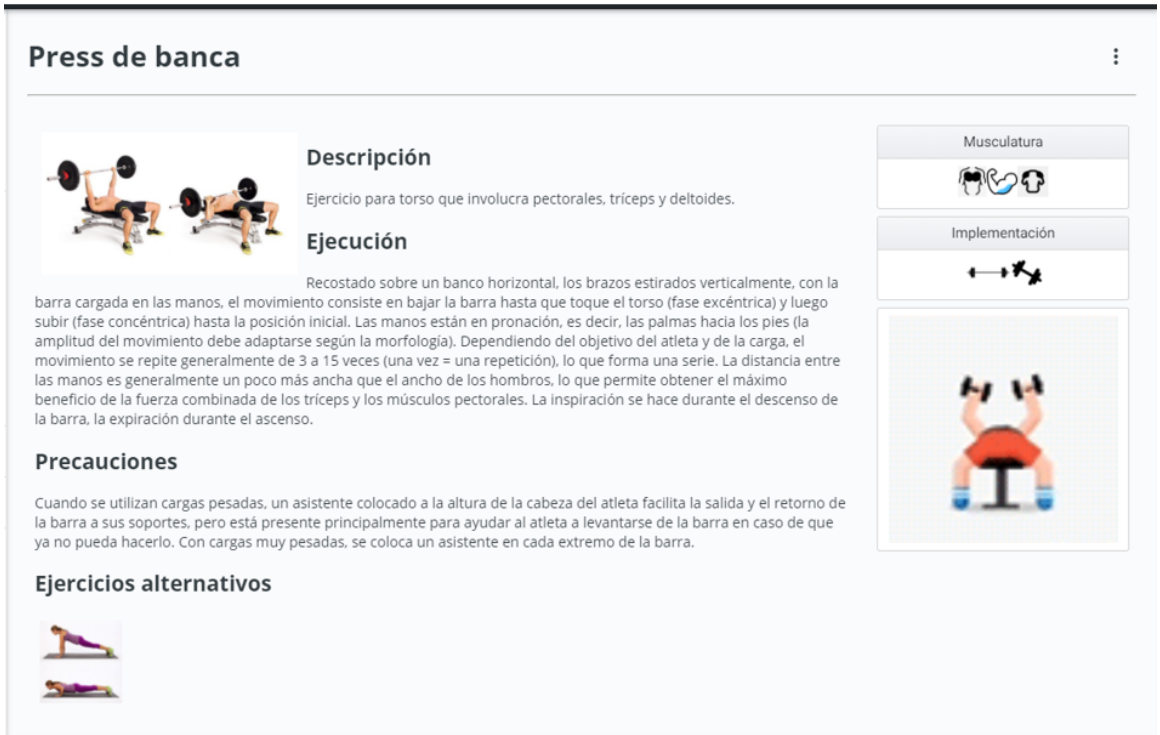


Figura B.19: Aplicación terminada: Detalle de ejercicio.

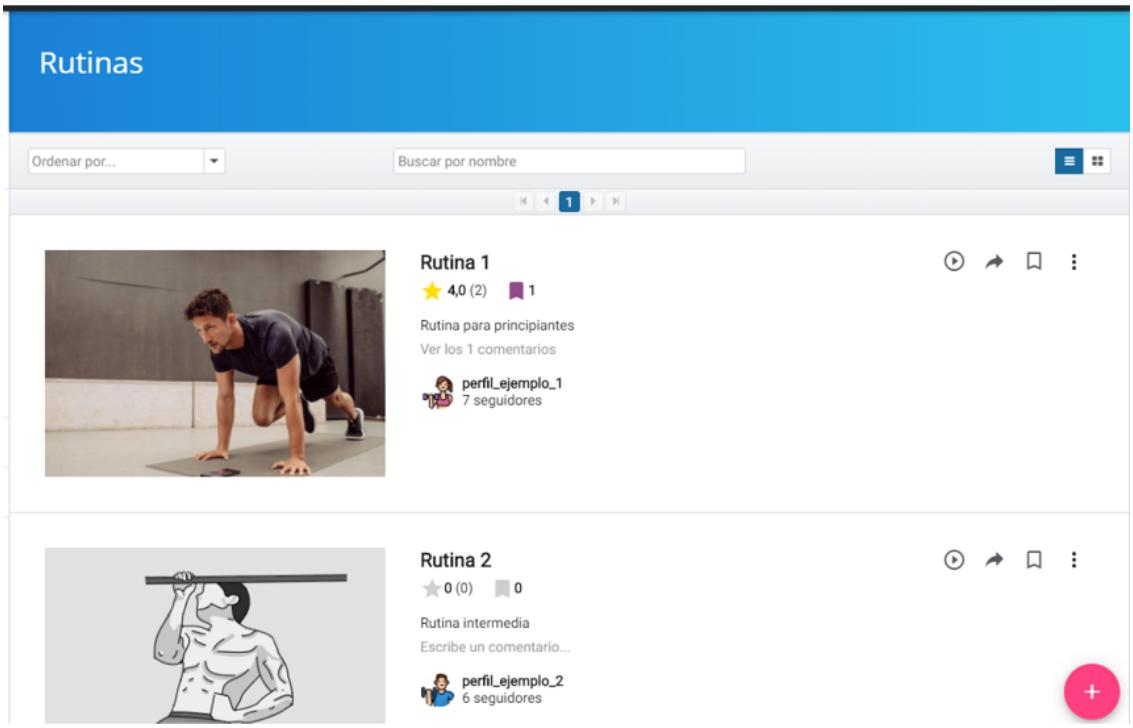


Figura B.20: Aplicación terminada: Lista de rutinas en pantalla grande.

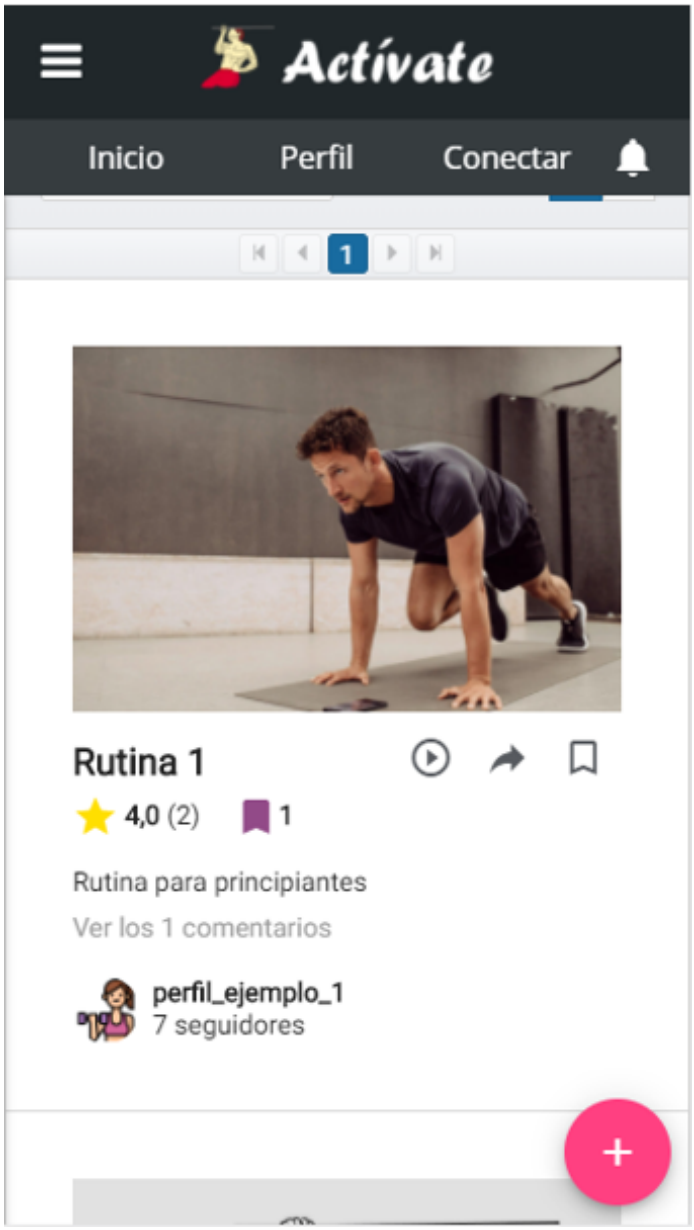


Figura B.21: Aplicación terminada: Lista de rutinas en pantalla pequeña.

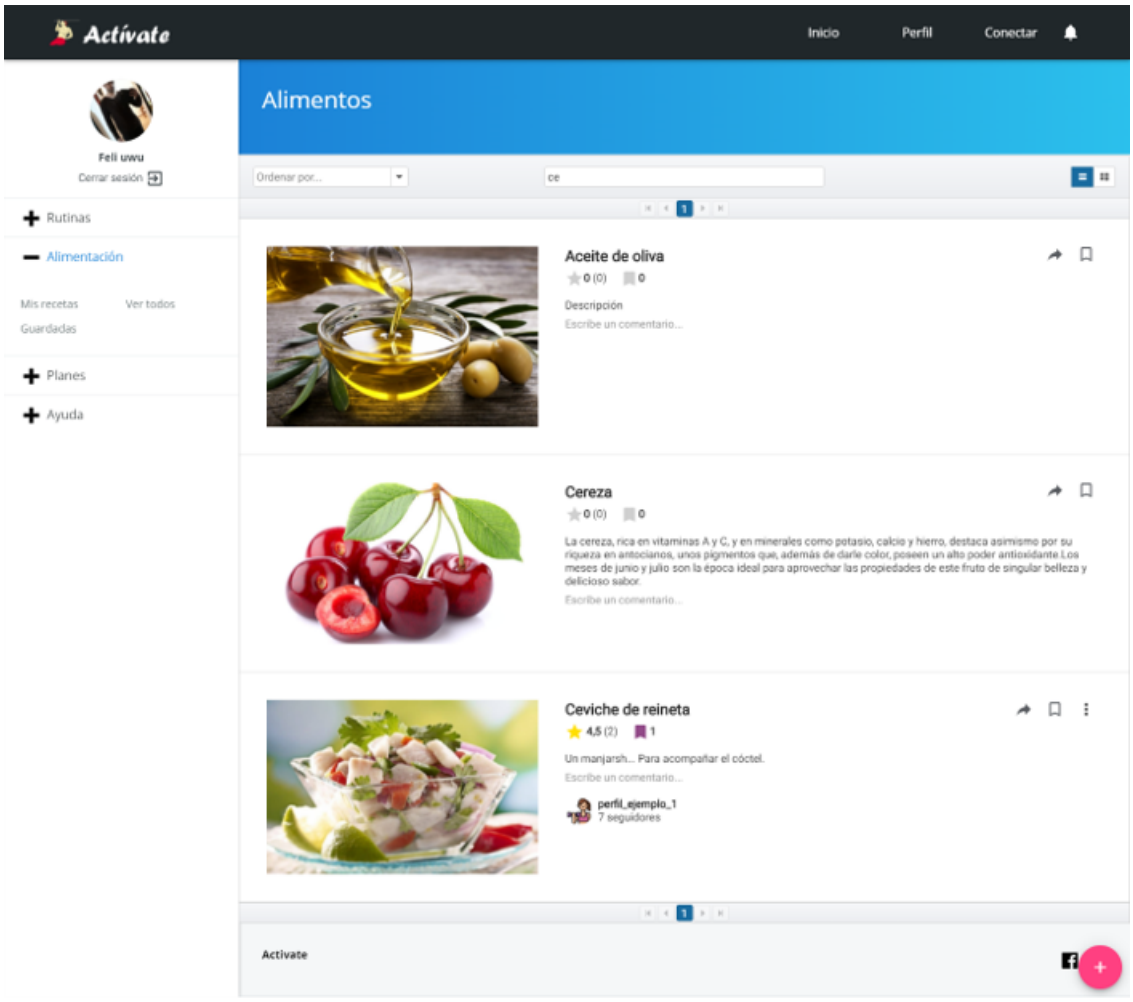


Figura B.22: Aplicación terminada: Lista de alimentos.

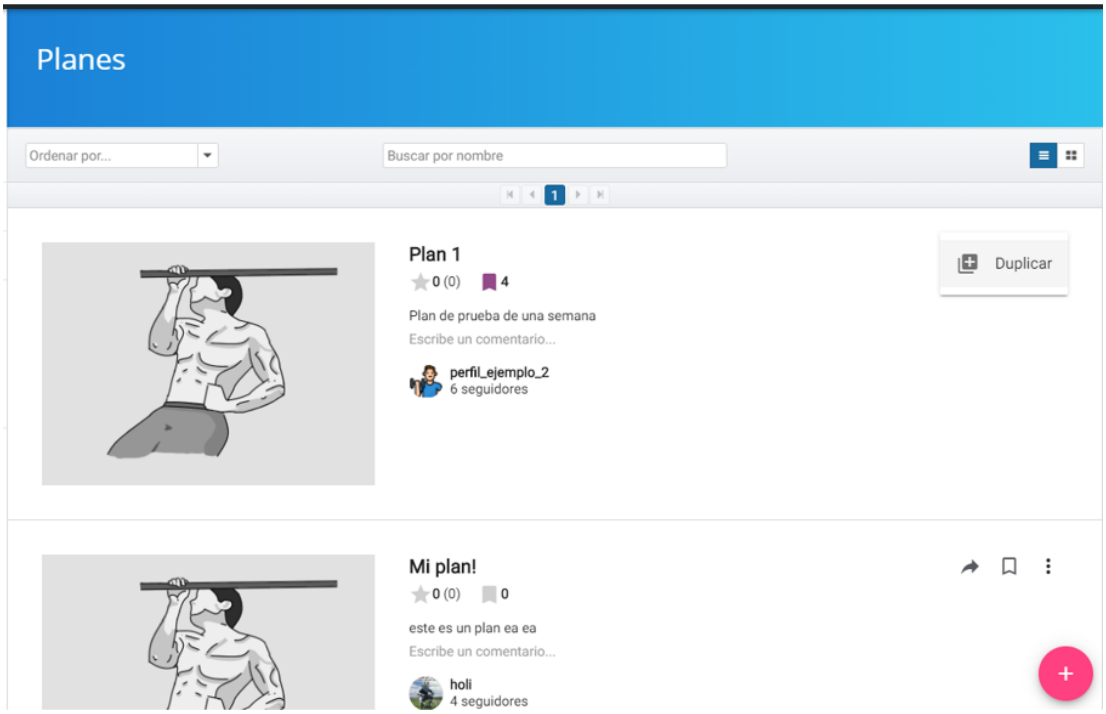


Figura B.23: Aplicación terminada: Lista de planes.

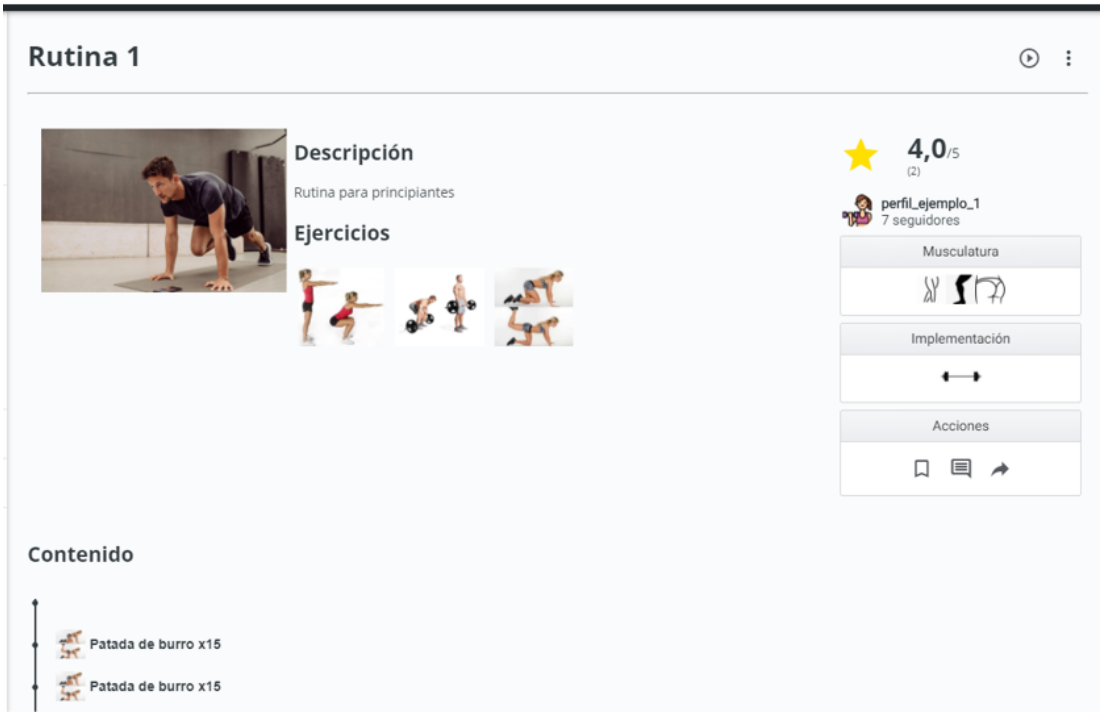


Figura B.24: Aplicación terminada: Detalle de rutina.


Figura B.25: Aplicación terminada: Ejecución de rutina en pantalla mediana.


Figura B.26: Aplicación terminada: Ejecución de rutina en pantalla pequeña.

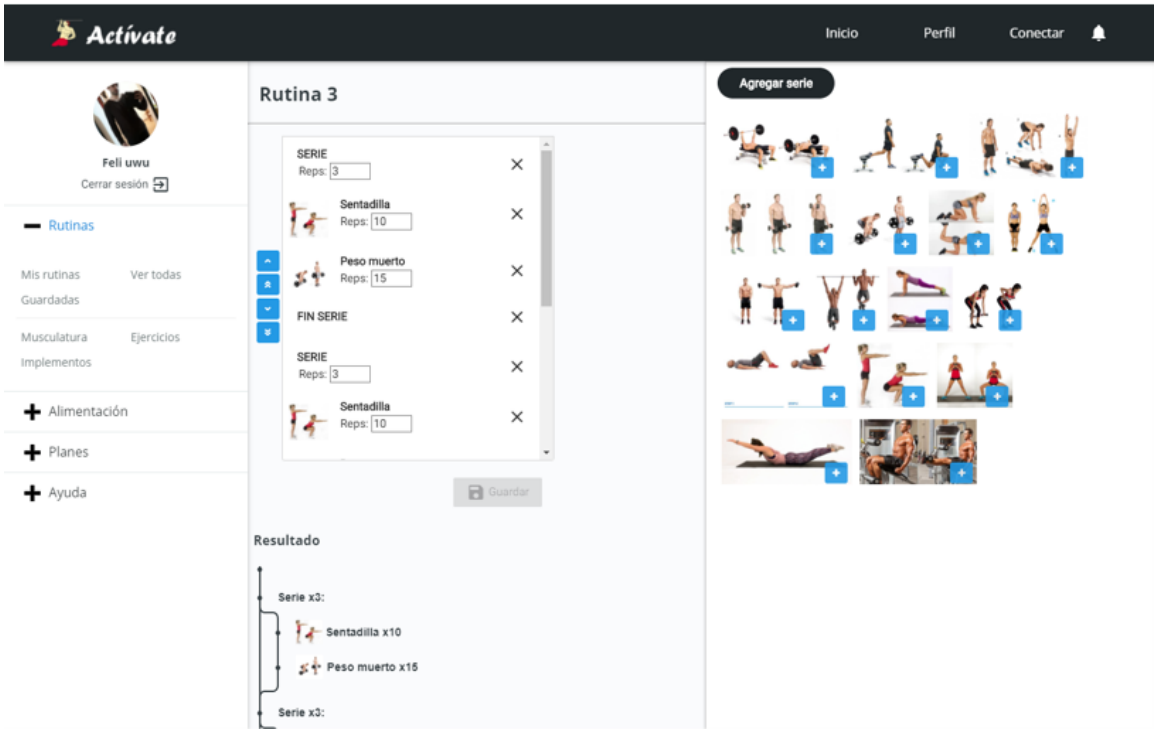


Figura B.27: Aplicación terminada: Construcción de rutina en pantalla grande.

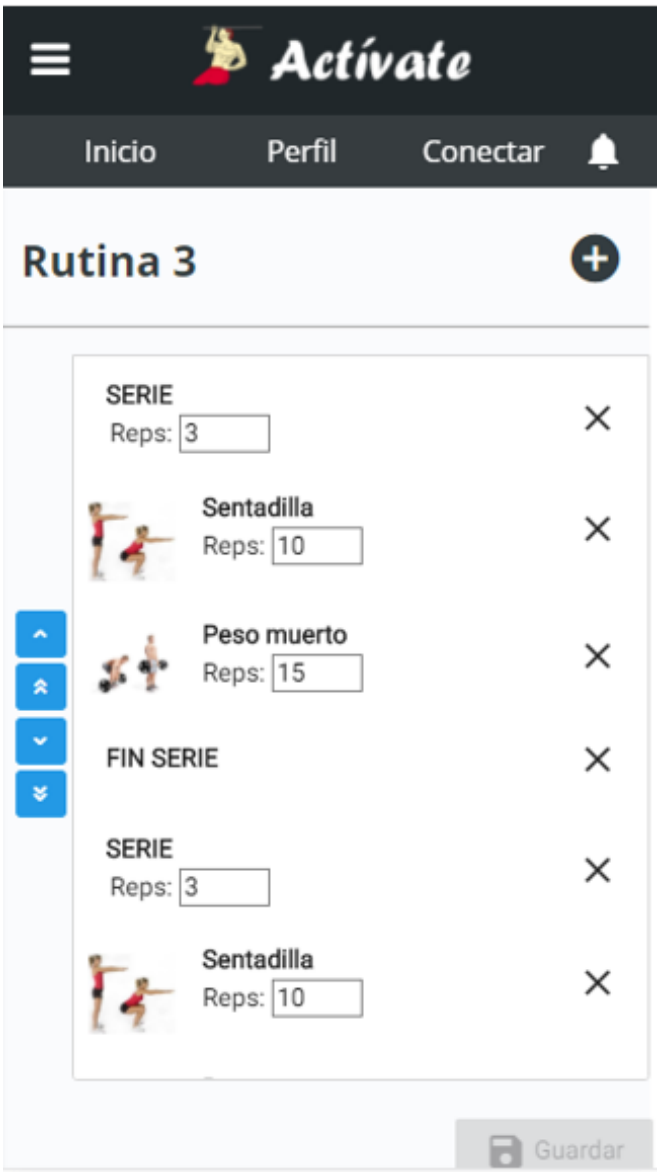


Figura B.28: Aplicación terminada: Construcción de rutina en pantalla pequeña.

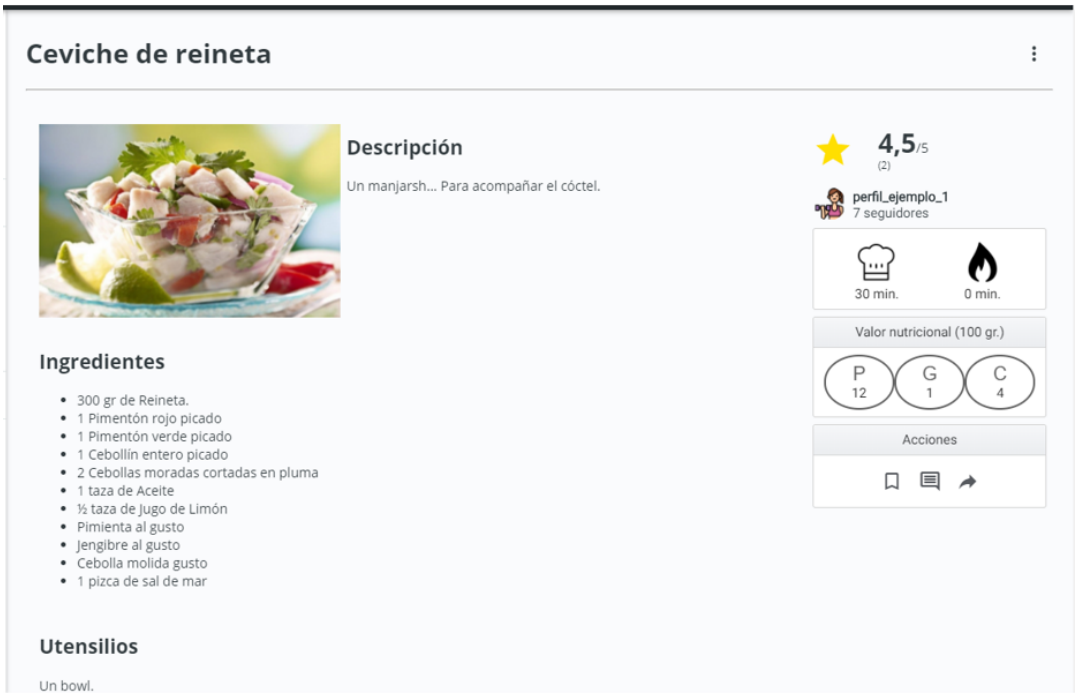


Figura B.29: Aplicación terminada: Detalle de receta.

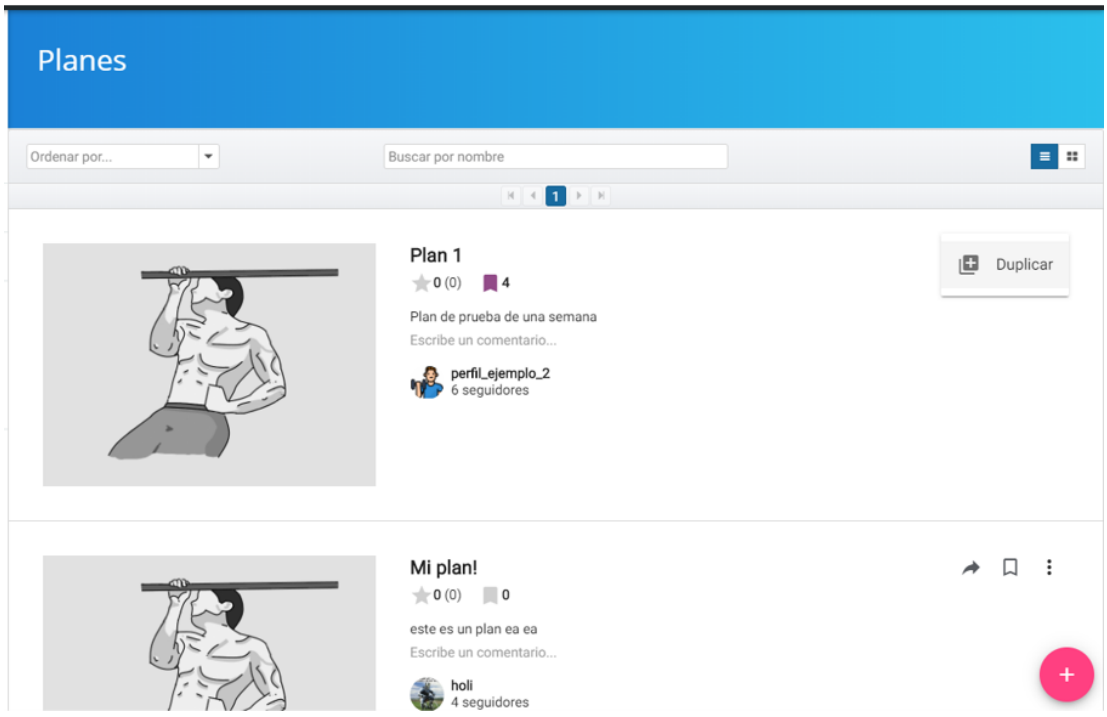


Figura B.30: Aplicación terminada: Lista de planes.

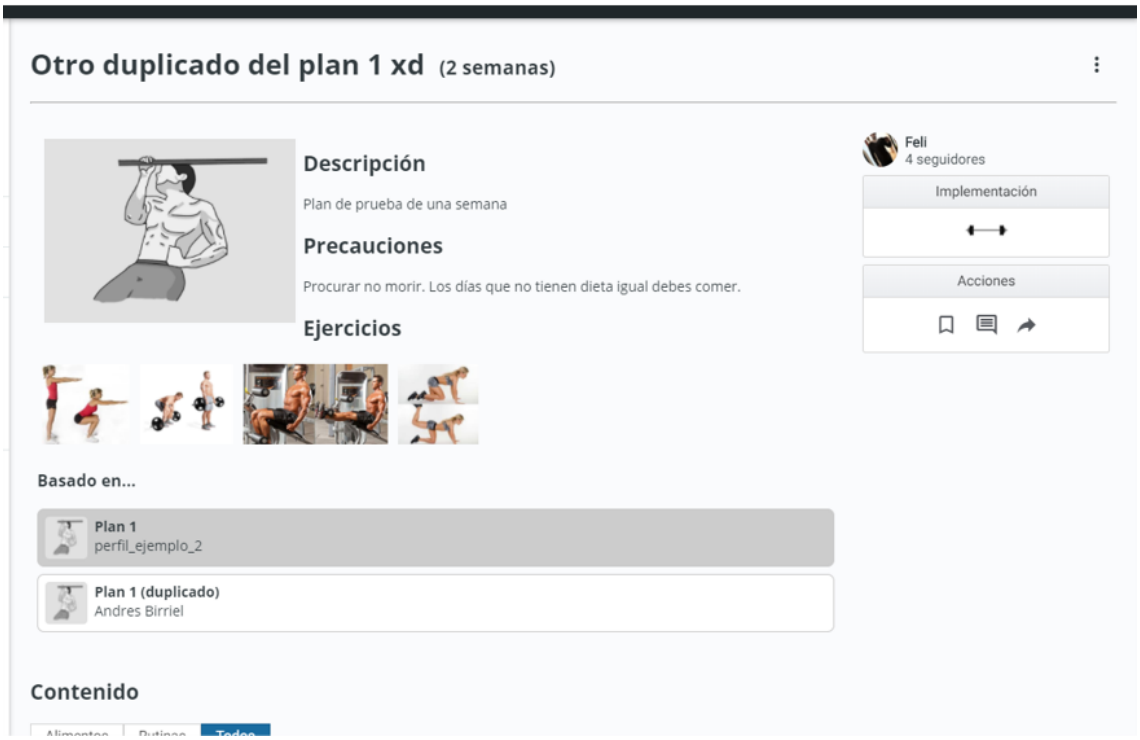


Figura B.31: Aplicación terminada: Detalle de plan creado por duplicación 1.

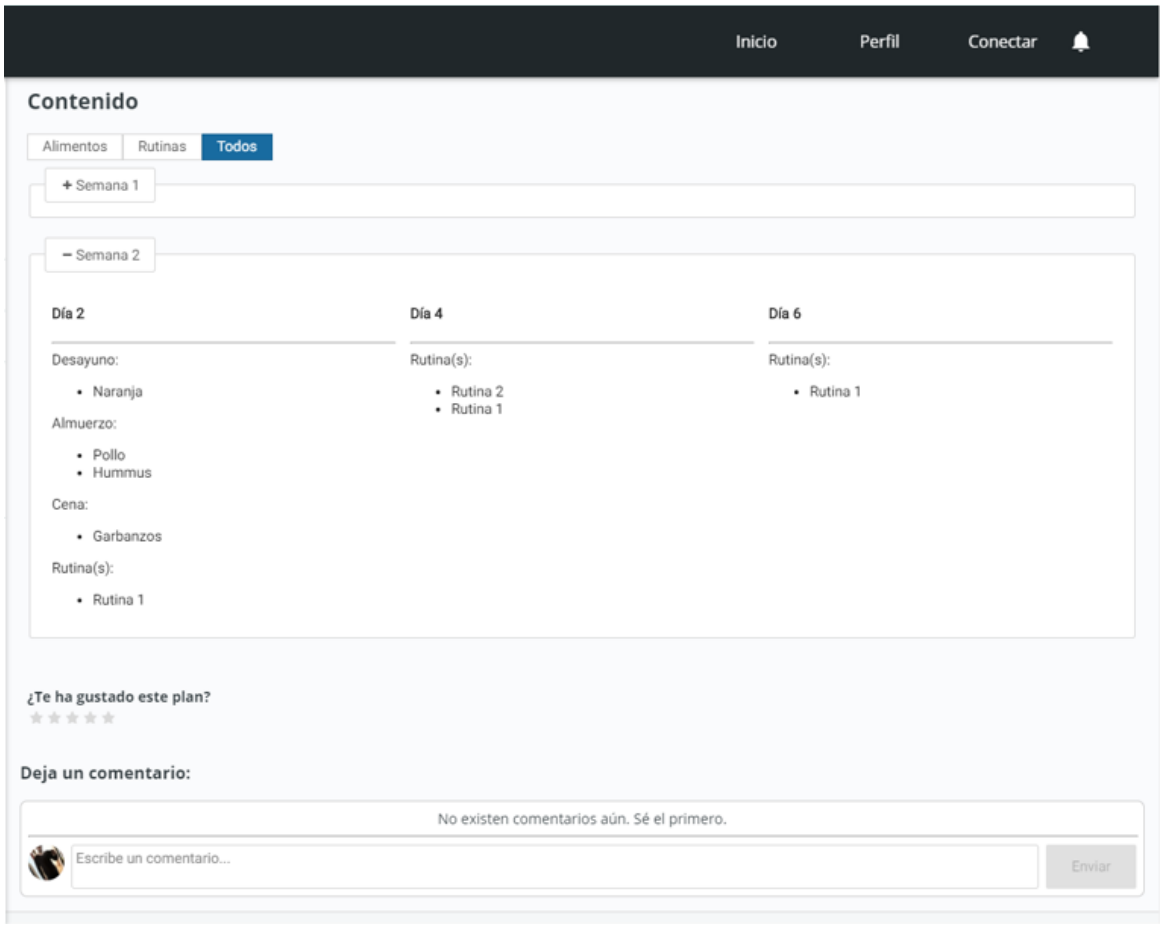


Figura B.32: Aplicación terminada: Detalle de plan (continuación).

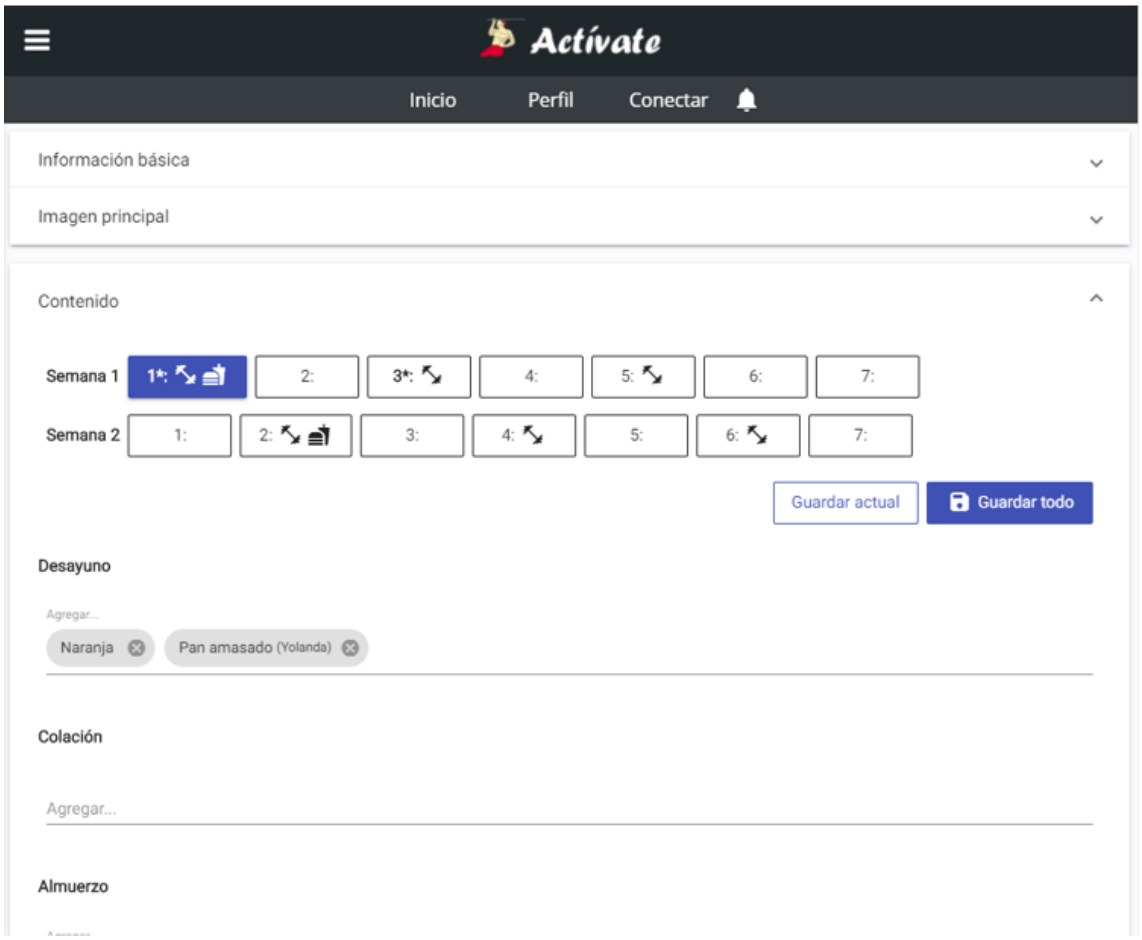


Figura B.33: Aplicación terminada: Edición de contenido de un plan en pantalla mediana.


Figura B.34: Aplicación terminada: Edición de contenido de un plan en pantalla pequeña.

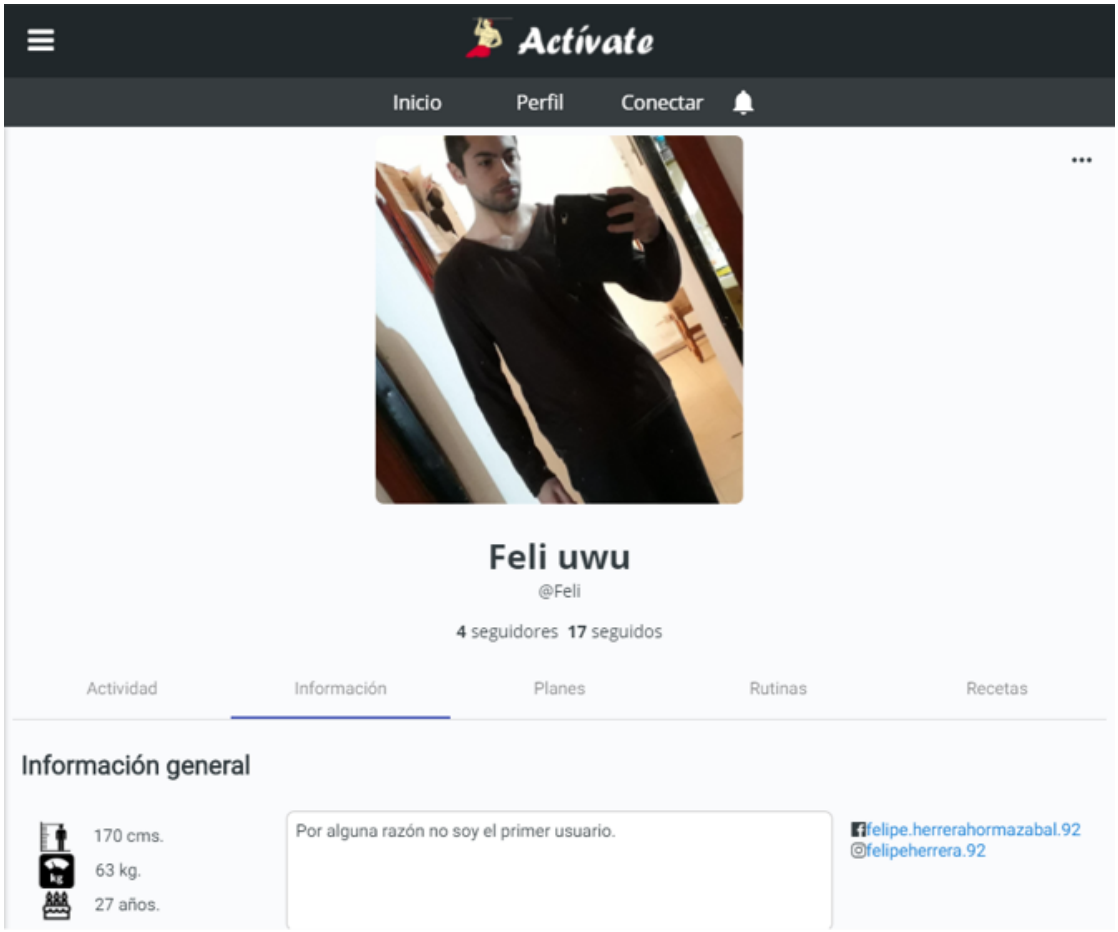


Figura B.35: Aplicación terminada: Perfil de usuario en pantalla mediana.

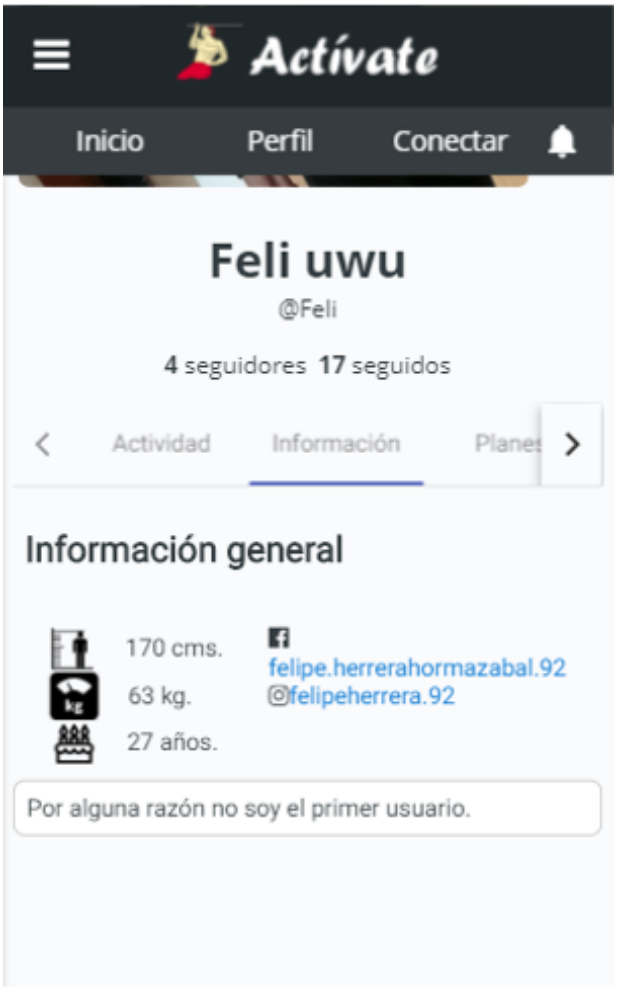


Figura B.36: Aplicación terminada: Perfil de usuario en pantalla pequeña.

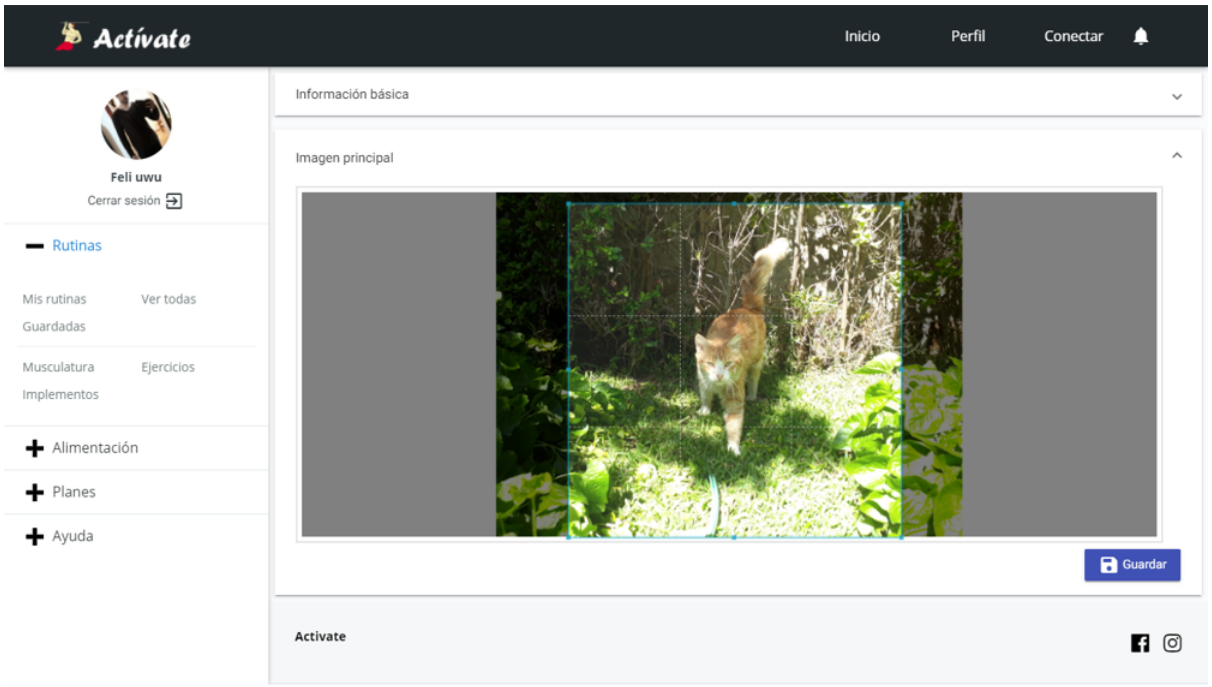


Figura B.37: Aplicación terminada: Cambio de foto de perfil.

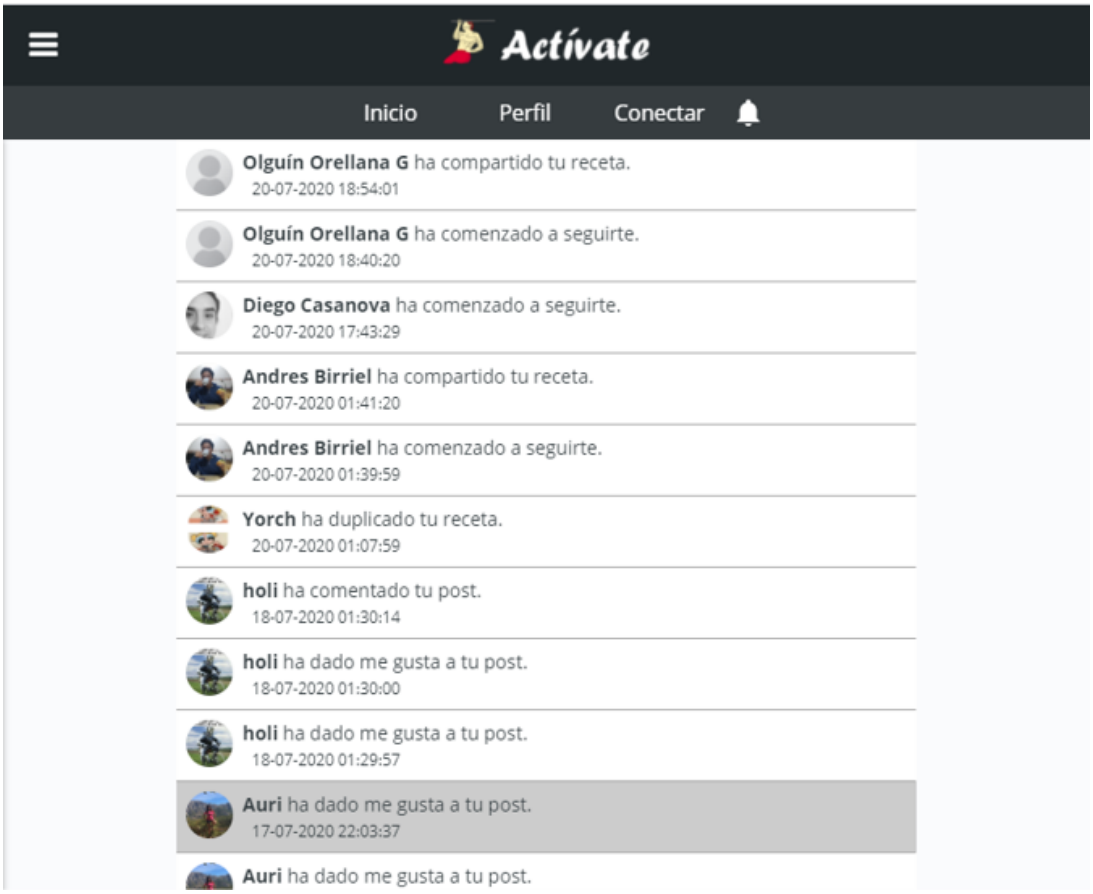


Figura B.38: Aplicación terminada: Notificaciones.

# C. Encuesta de validación

A continuación se presenta un resumen de los resultados obtenidos de la encuesta realizada para validar la aplicación. En esta participaron un total de 16 personas.

## C.1. Información general

En esta sección se consultan parámetros que permitan clasificar a los participantes. Tras aplicar filtros a la información, se hizo notar una diferencia entre quienes forman parte del mundo fitness y aquellos actualmente no relacionados. Llamaremos "interesados" al primer grupo, con el fin de aunarlos y mostrar sus resultados en las siguientes secciones.

Los factores consultados en esta sección son cuatro: edad (Figura C.1), relación con el fitness (Figura C.2), características usadas en la aplicación (Figura C.3) y herramientas usadas con anterioridad para estos fines (Figura C.4).

## C.2. Usabilidad (SUS)

Esta sección corresponde a la encuesta estandarizada SUS. Debido a las diferencias existentes, se muestran los resultados tanto del total de encuestados como de los interesados únicamente (Figuras C.5 hasta C.14).

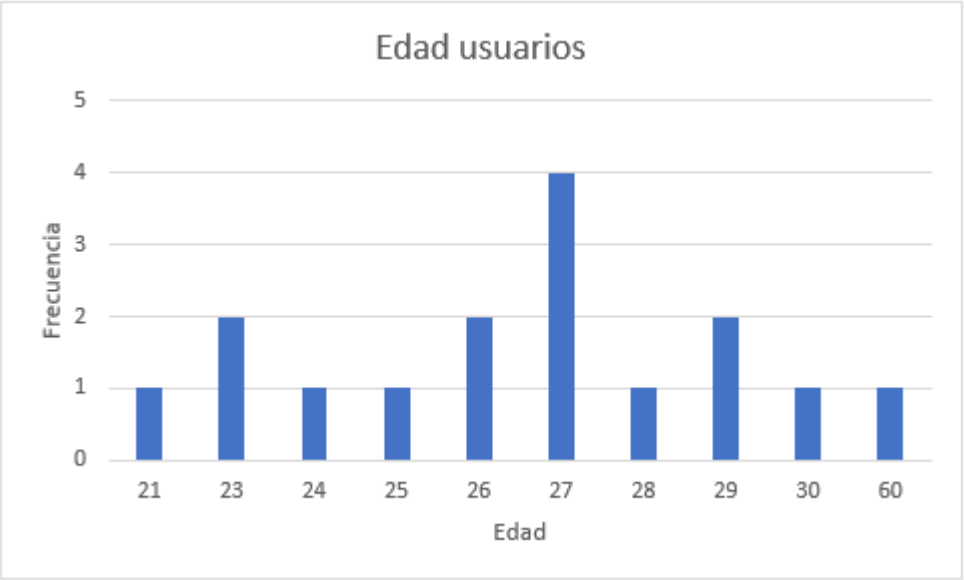


Figura C.1: Encuesta validación: edad.

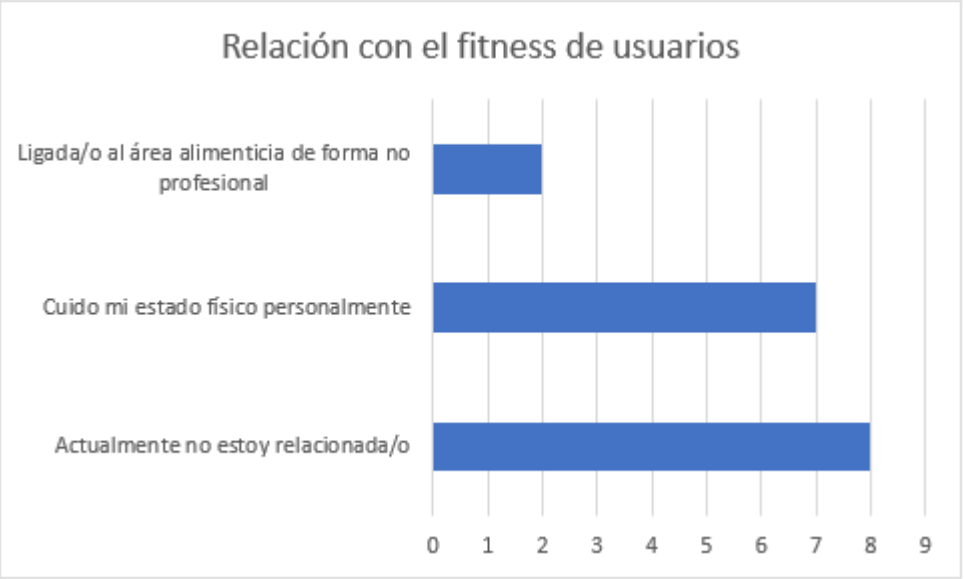


Figura C.2: Encuesta validación: relación con el fitness.

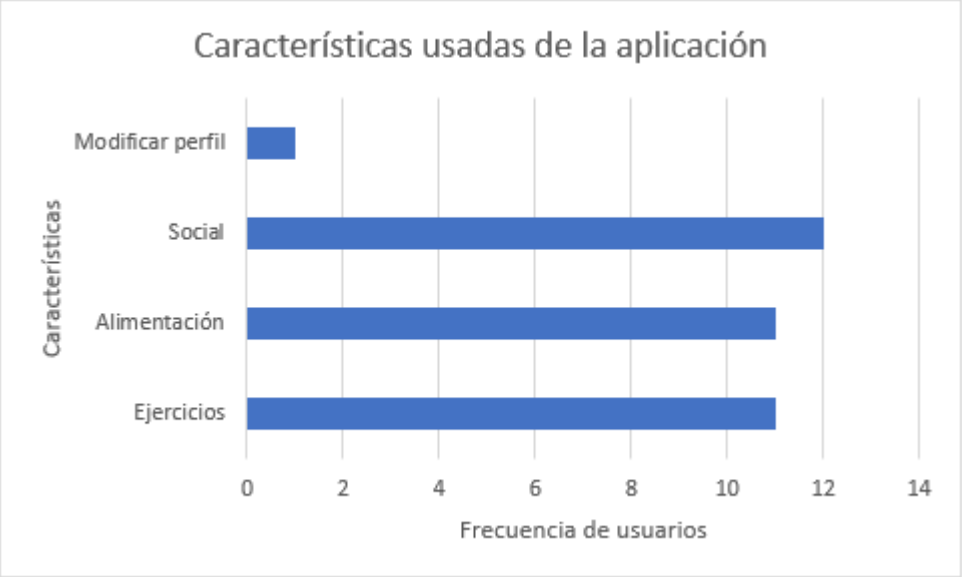


Figura C.3: Encuesta validación: características usadas.

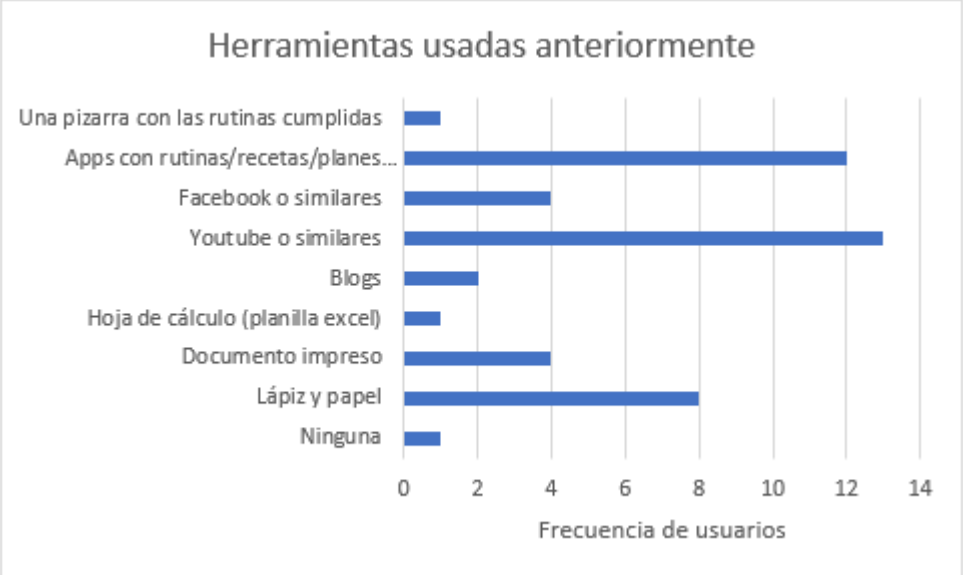


Figura C.4: Encuesta validación: herramientas usadas anteriormente.

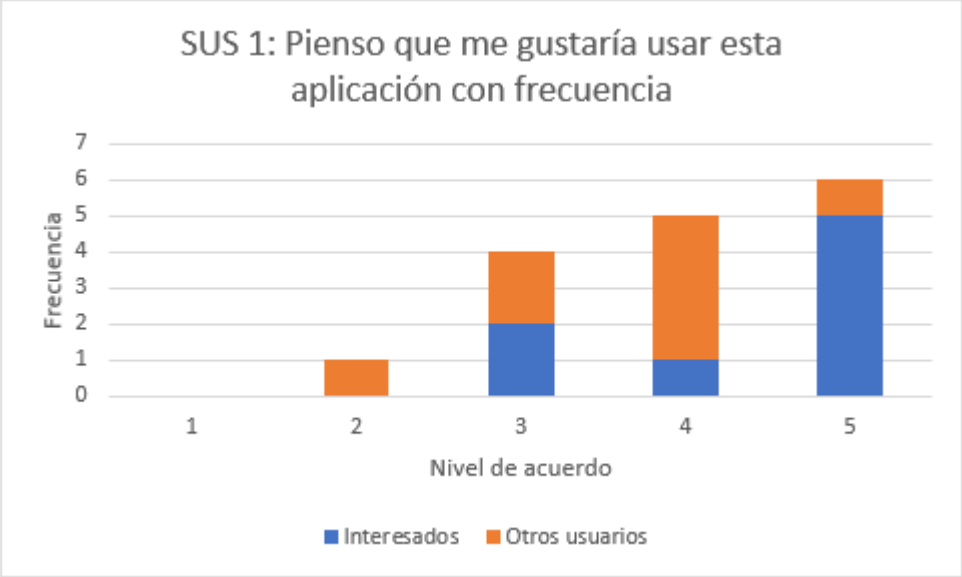


Figura C.5: Encuesta validación: respuestas acumuladas pregunta 1 escala SUS.

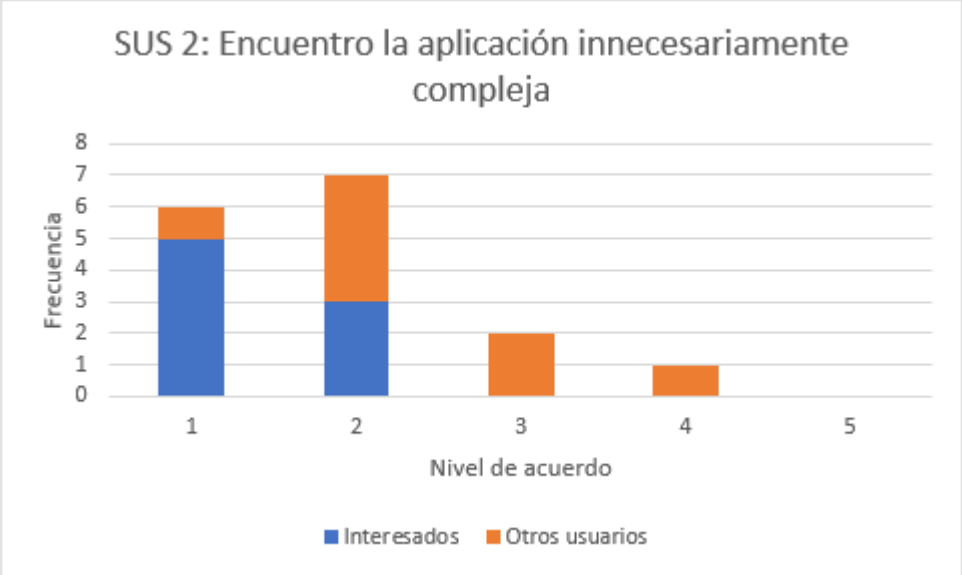


Figura C.6: Encuesta validación: respuestas acumuladas pregunta 2 escala SUS.

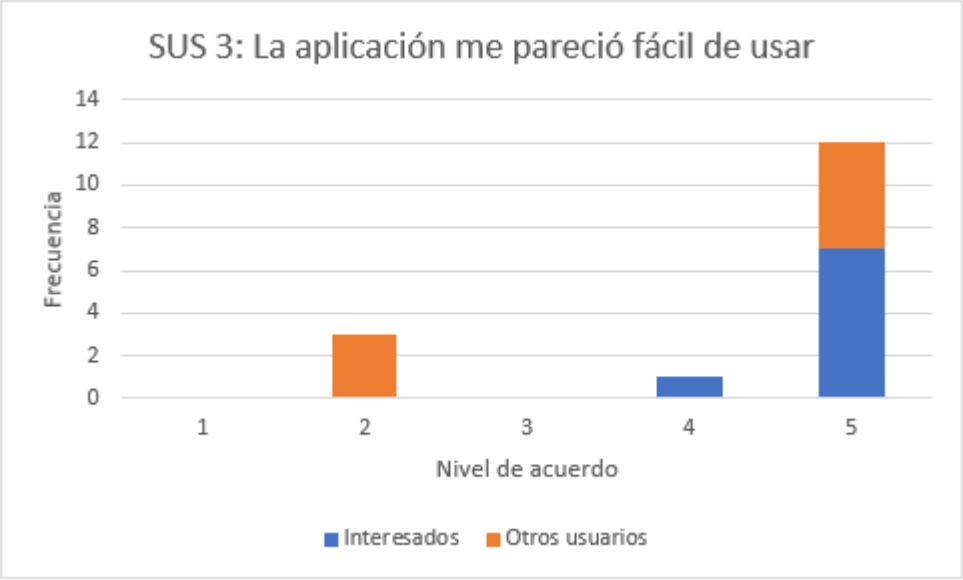


Figura C.7: Encuesta validación: respuestas acumuladas pregunta 3 escala SUS.

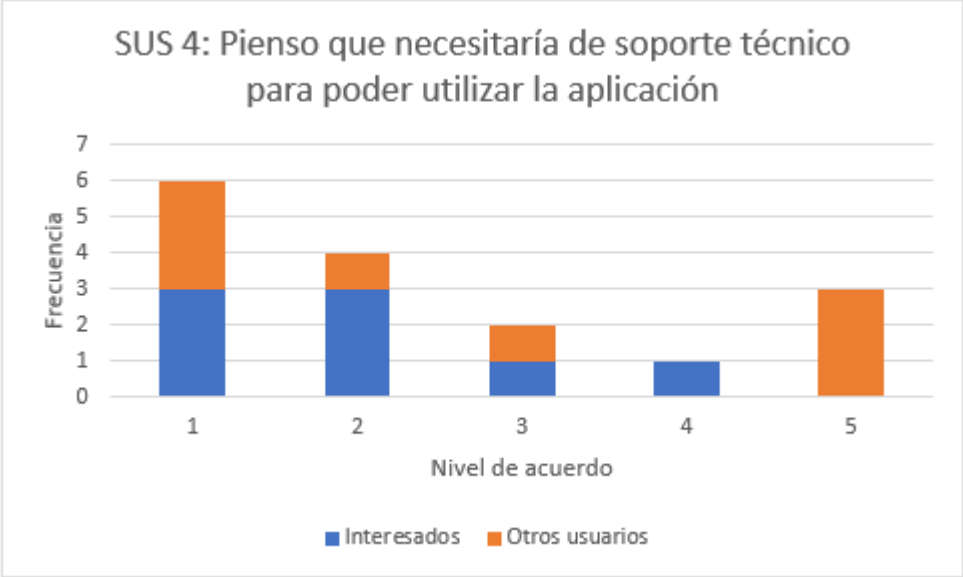


Figura C.8: Encuesta validación: respuestas acumuladas pregunta 4 escala SUS.

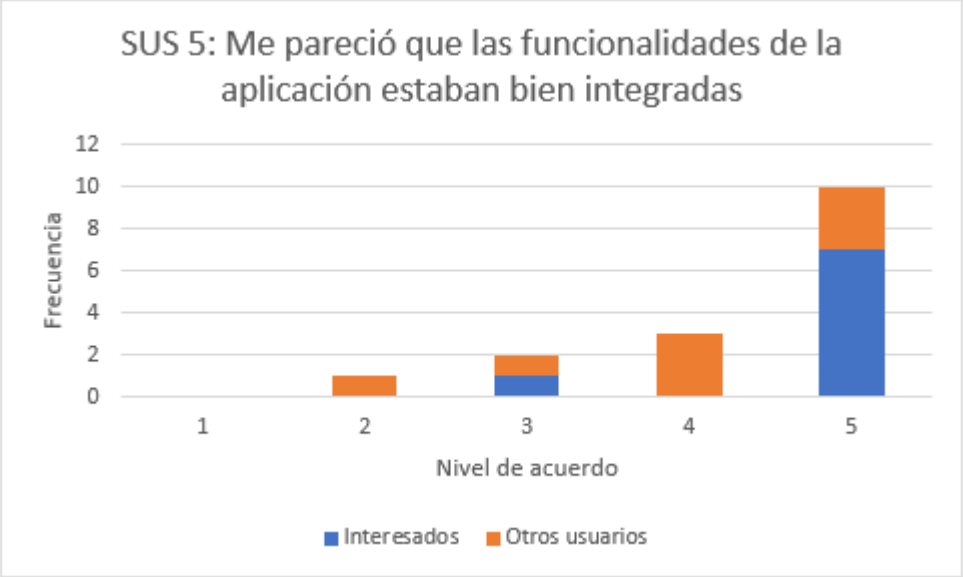


Figura C.9: Encuesta validación: respuestas acumuladas pregunta 5 escala SUS.

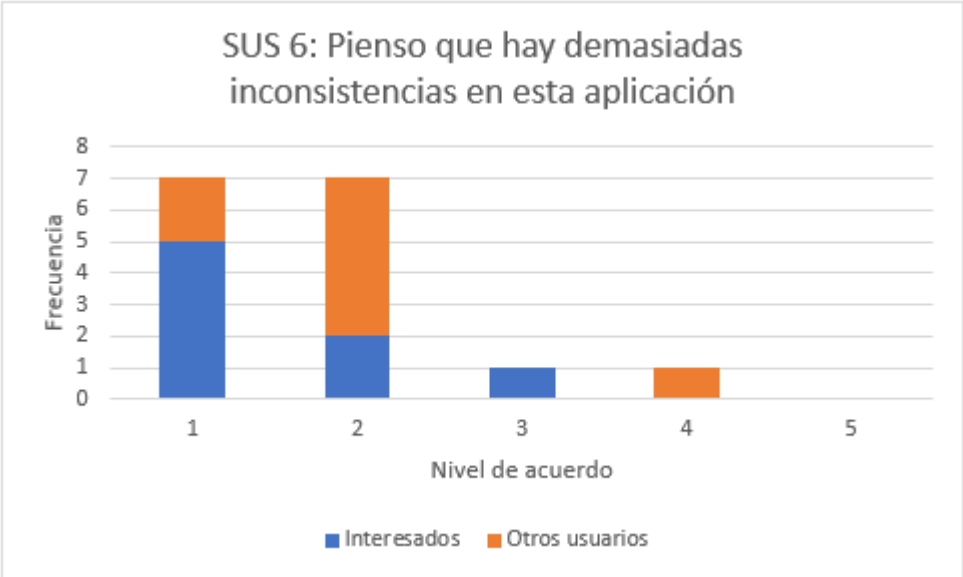


Figura C.10: Encuesta validación: respuestas acumuladas pregunta 6 escala SUS.

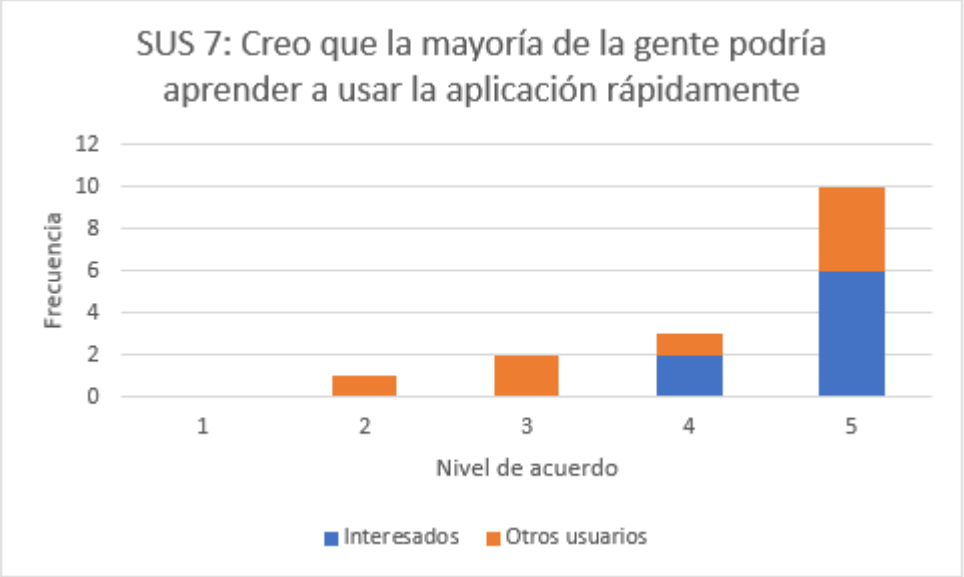


Figura C.11: Encuesta validación: respuestas acumuladas pregunta 7 escala SUS.

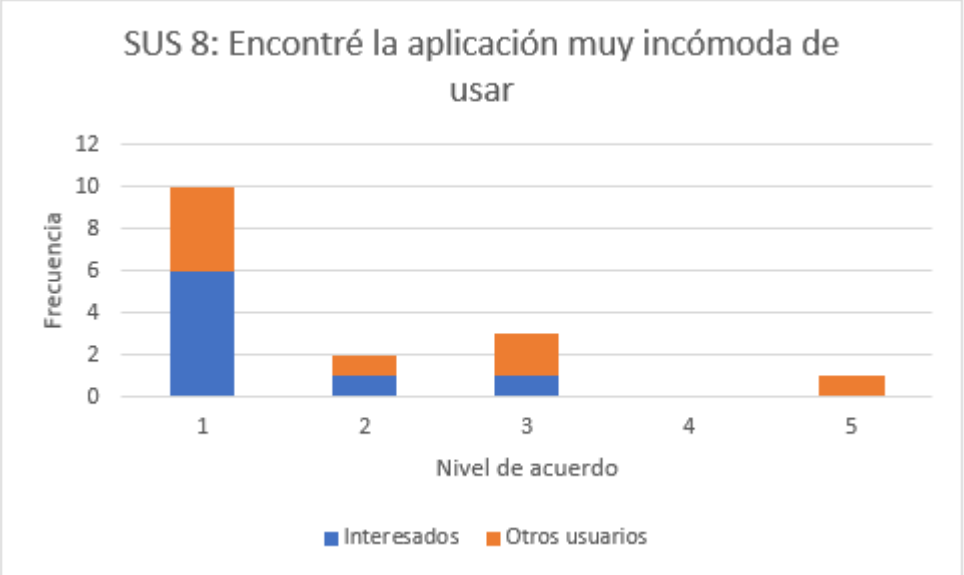


Figura C.12: Encuesta validación: respuestas acumuladas pregunta 8 escala SUS.

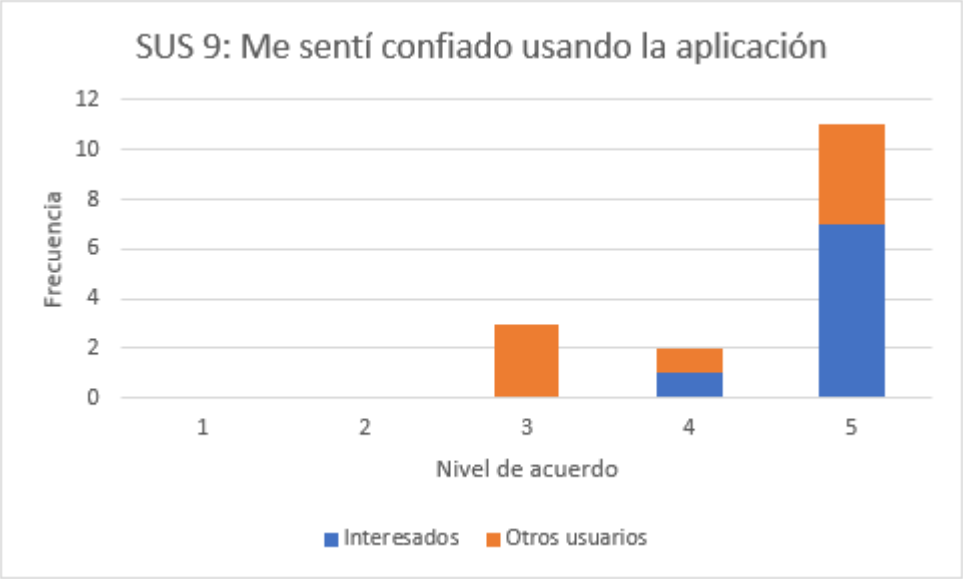


Figura C.13: Encuesta validación: respuestas acumuladas pregunta 9 escala SUS.

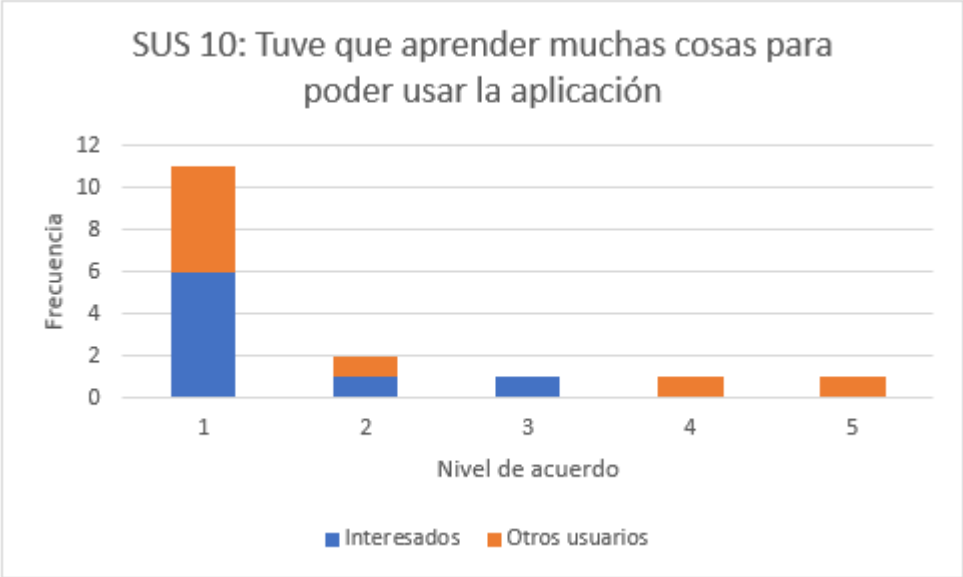


Figura C.14: Encuesta validación: respuestas acumuladas pregunta 10 escala SUS.

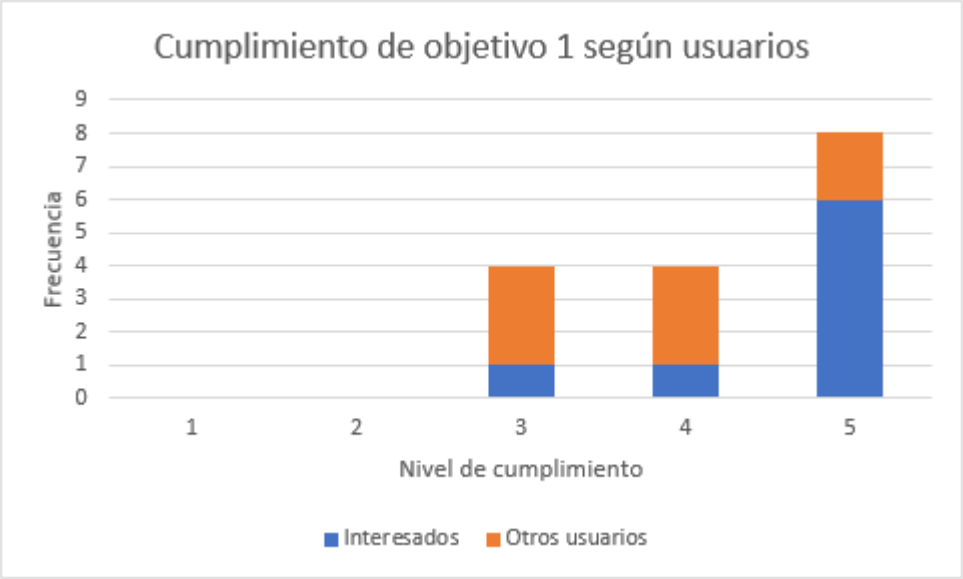


Figura C.15: Encuesta validación: Cumplimiento de objetivo 1 según usuarios.

## C.3. Cumplimiento de objetivos

En esta sección se consulta sobre la percepción de los usuarios respecto al nivel de cumplimiento de los objetivos del proyecto. Debido a las diferencias, se muestran los resultados del total de encuestados y de los interesados (Figuras C.15 hasta C.19).

Por motivos de transparencia las respuestas se han mantenido tal como fueron escritas, agregando solo la mayúscula inicial y el punto final según fuera necesario.

## C.4. Comentarios

Finalmente, se incluyeron las siguientes preguntas abiertas para interpretar mejor los resultados:

- ¿Qué cosas te parece que están bien?
- ¿Qué cosas deberían ser corregidas/mejoradas?
- ¿Qué cosas nuevas te gustaría ver?
- Espacio libre para comentar.

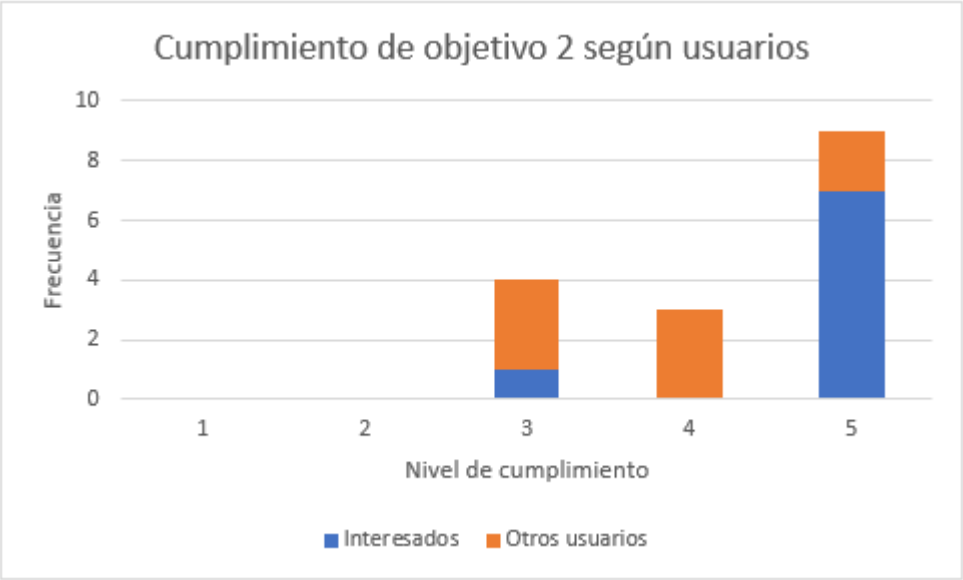


Figura C.16: Encuesta validación: Cumplimiento de objetivo 2 según usuarios.

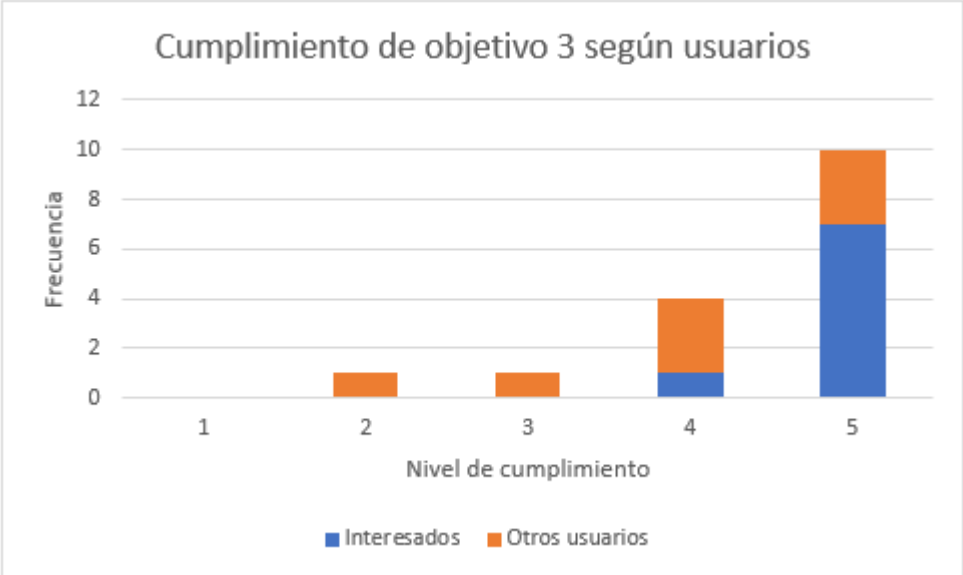


Figura C.17: Encuesta validación: Cumplimiento de objetivo 3 según usuarios.

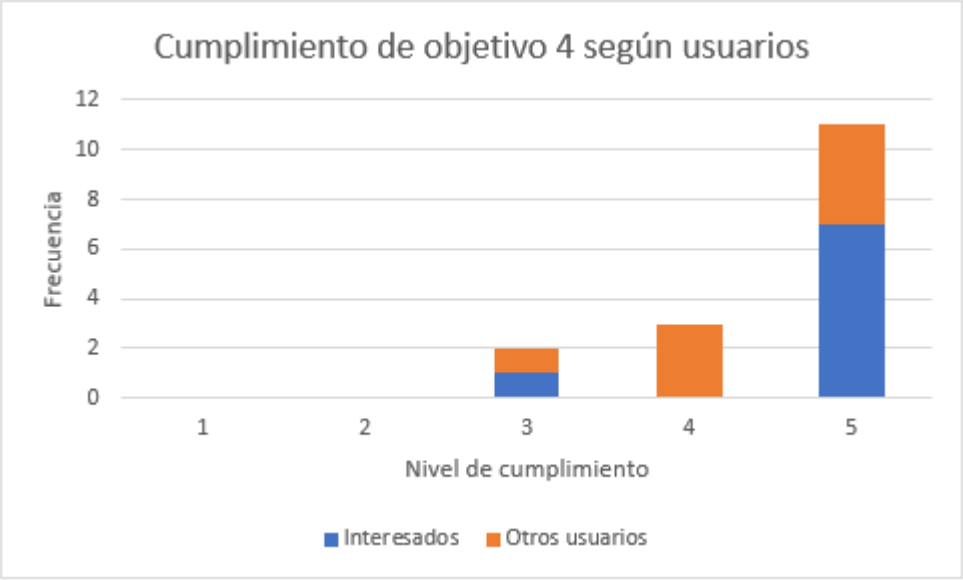


Figura C.18: Encuesta validación: Cumplimiento de objetivo 4 según usuarios.

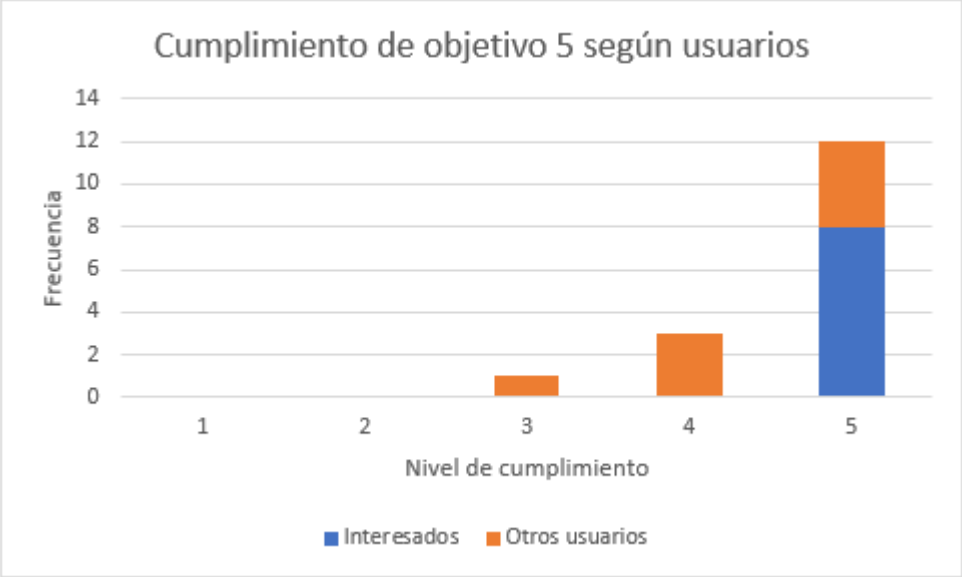


Figura C.19: Encuesta validación: Cumplimiento de objetivo 5 según usuarios.

**¿Qué cosas te parece que están bien?**

- Reúne distintas partes del fitness en un solo sitio.
- La interfaz, la información provista.
- Realizar la rutina, agregar recetas, agregar planes.
- Incluir rutinas, permitir sociabilidad progresos.
- El inicio, el perfil, "conectar" para conocer a más gente, crear las rutinas, la alimentación. Además de poder ver en inicio las actividades de los demás y que puedan compartir recetas.
- El orden, casi todo encuentro que está bien.
- La forma de compartir experiencias mientras uno va viendo cambias en si mismo gracias a las rutinas.
- La gráfica.
- Que sea tipo red social pero que no se enfoque solo en una cosa.
- Me encanta la aplicación la.manera de acceder a ella.
- Creo que en general la aplicación está bastante bien, debido a que facilita el acceso a la vida fitness incluyendo la alimentación que es una de las cosas más importantes.
- La forma en que están integradas las funciones.
- La posibilidad de formar parte de una comunidad con otro usuarios con los mismos intereses.
- La funcionalidad de la aplicación esta muy bien se estructura de forma fluida, las secciones le dan un aspecto sencillo y entendible por cualquiera, las sugerencias textual a los iconos también la hacen intuitiva, la velocidad de refresco es aceptable.
- Lo completo que está.
- Todo correcto.

**¿Qué cosas deberían ser corregidas/mejoradas?**

- El contenido.
- Más tipos de ejercicios.
- Rutinas realizadas, cumplimiento de alimentación, instructivo de la aplicación.
- Rutinas con videos.
- Tal vez un video explicativo de como funciona la aplicación para gente que no se maneje mucho en este ambiente tecnológico, pero por el resto esta bien.
- Como es harta la información que se comparte, sería importante generar logos o diferencias de colores que permita incitar al usuario a investigar de forma intuitiva y aprender así de forma autónoma como usar la red social.
- El manejo y cambio de secciones dentro de la aplicación, cuando navegué en un principio no me pareció tan intuitiva.
- No entendí en sí la app.
- Un poco de más colores vivos (para llamar más la atención ).
- Pienso que se podrían incluir mas ejercicios y recetas inicialmente.
- Poner un servidor adecuado, corregir los ingresos de información, como peso, edad y estatura ( puse 0,1 g y 1700 cm).
- La velocidad de conexión y algunas notificaciones de tareas realizadas.
- Los editores de contenidos, donde iría la publicidad de terceros?, en los iconos falta que existan representación textual como la que existe en los iconos de proteinas carbohidratos cuando pasas el cursor por encima, que en las rutinas en el seguimiento al lado derecho vaya marcando con otro color cuando esta en 'x' parte de la rutina.
- Más información de ejemplos.

**¿Qué cosas nuevas te gustaría ver?**

- Una línea del tiempo de avances y un calendario para seguir los planes. Tags.
- Rutinas pre hechas, para gente que se inicia.
- Videos realizando el ciclo del ejercicio que se está realizando, tiempo en que demoro haciendo cada ejercicio.
- Interacción con desconocidos a través de hashtags.
- Crear un foro de preguntas mas frecuentes a medida que más gente la ocupe.
- No se pudo apreciar si se podian subir videos, pero me gustaria poder subir videos!
- Poder descargar las planillas en formato pdf para que sean impresas.
- Podrían ser videos de muestra sobre los ejercicios.
- Una guía rápida de introducción, para saber cómo usar la app.
- Que haya un "entrenador virtual" que recomiende cosas dependiendo de lo que usas.
- Videos relacionados con el tema.
- Algún indicador de que implementos se deben utilizar en los ejercicios.
- Agregar rutinas de ciclismo, u otros deportes, una aplicación de control de peso.
- Faltan animaciones en los cambios de pestañas son super planos, falta historial de busquedas anteriores, falta que tenga cuentas verificadas, faltan estadisticas de visualización de perfil, hora y alcance de las publicaciones, tipos de cuentas, planes y que características tendrán esas cuentas para diferenciarlas, cuanto tiempo estuve conectado, registro de dispositivos en que estoy conectado aún, dispositivos confiables, formas de pago de los planes, sistemas de pago dentro de la aplicación para transacciones entre usuarios, geolocalización en tiempo real usando la API de Google o visualización del mapa propios. administrador/contenedor para las notificaciones, para silenciarlas eliminarlas o bloquear

al usuario, usuarios públicos privados, donaciones para un café o para campañas. Cuando se si un usuario esta conectado podría hacer match con alguien si quiero, como puedo denunciar un perfil o desactivar mi perfil.

- Recetas.

**Espacio libre para comentar.**

- Encuentro que es una aplicación que está en constante mejora y ayudaría bastante a las personas que lo usan aun así falta mejoras pero es una buena aplicación. Buen trabajo!!!
- Debería haber un inicio que permita ver las fotos recientes de todos los que participan en la red social.
- Solo pondria un poco mas de enfasis en la parte de navegacion.. por ejemplo no supe como visitar otros perfiles.
- Felicito al creador ya que hace que sea más fácil y didáctico el tema fitness.
- Esta bastante buena la aplicación.
- Esta súper buena la aplicación!!
- Felicitaciones por el proyecto! genera expectativas :D
- Quisiera agradecer la confianza como usuario de prueba y el tiempo de trabajo empleado en esta memoria/aplicación se vienen tiempos de trabajo constante para mejorar su funcionalidad futura que es ilimitada.
- Está muy bien elaborado, las rutinas de ejercicios y la manera adecuada de alimentación .felicitaciones !!!