ULPGC Universidad de Las Palmas de Gran Canaria | Escuela de Ingeniería Informática

Grado en Ingeniería Informática

Trabajo de Fin de Grado

Nutrain: aplicación web de nutrición, entrenamiento y otros aspectos de la salud.

Daniel Suárez Cáceres

Tutor: Luis Hernández Acosta

Las Palmas de Gran Canaria

Junio de 2020.

Índice general

Índice de cuadros y figuras

Agradecimientos

En primer lugar, agradecer a mi familia, que siempre ha estado ahí para apoyarme y animarme, tanto en los mejores días como en los peores, que es especialmente cuando uno más lo necesita. Gracias de corazón, de verdad.

En segundo lugar, agradecer a mi tutor D. Luis Hernández, un gran profesor que ha sabido guiarme durante todo el desarrollo y del cual he podido aprender muchas cosas gracias a su gran experiencia en este tipo de proyectos.

Por último, y no por ello menos importante, agradecer a aquellos profesores que durante la carrera nos enseñaron muchas cosas más allá de aprobar el examen de su asignatura, y que, de alguna manera u otra, consiguieron mantener y avivar mi interés por este maravilloso mundo de la informática.

Muchas gracias a todos.

Resumen

El mundo de la nutrición, el entrenamiento y la salud en general lleva mucho tiempo presente con nosotros. Empezó hace décadas con el culturismo, el cual practicaba un porcentaje muy pequeño de la población, pero con el paso de las décadas se ha ido extendido a toda la población, asentándose firmemente en nuestra sociedad.

Sin embargo, gracias a las nuevas tecnologías, ahora más que nunca es increíblemente sencillo tener acceso a cualquier contenido relacionado con tu salud: buscar rutinas de entrenamiento en Internet, vídeos de YouTube sobre cómo hacer un ejercicio en concreto, recetas y dietas de todos los tipos que puedas imaginar, y un largo etcétera. Y con ello, por supuesto, surge la necesidad de llevar un control y seguimiento de todo lo anteriormente nombrado.

La motivación de este Trabajo de Fin de Grado es, por tanto, permitir al usuario mantener un seguimiento y cuidado de su salud (dividiéndola en cuatro bloques: nutrición, entrenamiento, psicología y fisioterapia) a través de una aplicación web, y, a su vez, permitir a los profesionales de estas áreas darse a conocer y prestar sus servicios a través de la misma.

Abstract

The world of nutrition, training and health has been present in our society for a long time. It started a few decades ago with a movement called bodybuilding, which was exclusive to a few people, but it has become more popular through the years, and now is well stablished in our society.

However, thanks to new technologies, it has never been easier to access and learn about this world: you can find raining routines in many blogs, thousands of videos in YouTube about how to make an exercise, websites dedicated to recipes and diets and so on. With this, the need of tracking these different aspects became important.

For that reason, the purpose of this final project is to allow users to keep track of their health (dividing it into four blocks: nutrition, training, psychology and physiotherapy) through the web application, and at the same time, allow professionals of these fields to make themselves known and give their service.

Capítulo 1: Introducción

1.1 Estado actual

La salud y el bienestar han sido siempre objeto de atención y debate. La gente cada vez se interesa más por saber cómo puede mejorar salud: qué hay que comer y que no, cuánto tengo que entrenar, cuantos "pasos diarios" tengo que hacer, cuál es la mejor manera de recuperarme de una lesión.

En la última década, y especialmente, los últimos años, la industria del fitness ha experimentado un crecimiento considerable. Podemos tomar algunos datos de diferentes estadísticas recopiladas:

- La industria del fitness y de la salud genera en torno a 100.000 millones de dólares a nivel global. (Fuente: BusinessInsider [1])
- Sólo en Estados Unidos, la industria del fitness genera en torno a treinta mil millones de dólares y ha experimentado un crecimiento constante de en torno a un 5% anualmente durante los últimos 5 años. (Fuente: GloFox [2])
- En el período del año 2000 al 2017, las afiliaciones a gimnasios y centros deportivos se duplicó, superando los 60 millones en 2017. (Fuente: Statista [3])
- La industria de la nutrición fue valorada en 15.600 millones de dólares. (Fuente: GrandViewResearch [4])

Como podemos observar, son números que nos hacen pensar que estamos ante una industria de gran importancia y reconocimiento a nivel mundial.

La tecnología y el crecimiento de los sistemas digitales han provocado un gran impacto en esta industria (Fuente: ITBrief [5]) de diferentes maneras: el crecimiento e influencia de las redes sociales (los famosos *fitness influencers*) y de una grandísima comunidad online, incorporación de televisores y pantallas táctiles en los gimnasios, crecimiento continuo de la venta de dispositivos *wearables* (pulseras de actividad mayormente) y similares, así como la aparición de aplicaciones de monitorización, entre otros.

Un *wearable* es un dispositivo electrónico, que, como sugiere su término en inglés (*wear*), se puede "vestir". Un ejemplo de *wearable* son las pulseras de actividad.

En este último grupo es donde entra este proyecto, en las aplicaciones de monitorización. Actualmente, existen diversas alternativas en el mercado, como por ejemplo MyFitnessPal, Cronometer o FatSecret [7], entre otras. No obstante, este proyecto incluye dos bloques olvidados por estas aplicaciones, que son la fisioterapia y la psicología, además de aunar también las funcionalidades propias de cualquier software de gestión de clientes, que permita gestionar contratos o concretar citas.

1.2 Motivación

La principal motivación de este proyecto era desarrollar una idea propia y además en un área relacionada con la salud, como es la nutrición, el entrenamiento, la fisioterapia y la psicología.

Se considera que es importante desarrollar proyectos dentro del ámbito de la informática y la industria software, pero también lo es el hecho de aplicar dichos conocimientos a otros campos, como en este caso, la salud. La informática, los sistemas digitales y la industria software están presentes en todos los aspectos de nuestra vida, y el principal objetivo debería ser siempre mejorar la calidad de esta y generar cambios positivos en la sociedad.

También se consideraba importante poner a prueba los conocimientos adquiridos durante la carrera y tratar desarrollar una aplicación de principio a fin, donde se tuvieran que desarrollar todas las fases: idea, análisis, diseño, desarrollo, documentación.

Por último, cabe destacar que la elección del nombre de la aplicación refleja en cierto aspecto el objetivo de la misma: “Nutrain”, diseñada para ayudar a nutrir y entrenar tu salud cuatro aspectos: nutrición, entrenamiento, psicología y fisioterapia.

1.3 Objetivos

El proyecto de la aplicación web “Nutrain” tiene como objetivo ofrecer al usuario una herramienta con la que pueda monitorizar y controlar diferentes aspectos de su salud, además de ofrecer una plataforma a través de la cual clientes y profesionales puedan comunicarse y establecer relaciones contractuales.

Por ello, los principales objetivos de la aplicación son los siguientes:

- Permitir al usuario gestionar y monitorizar su nutrición: guardar dietas, registrar las comidas que realiza cada día, visualizar gráficos e informes de las mismas y la posibilidad de descargar toda esta información.
- Permitir al usuario gestionar sus entrenamientos: crear rutinas de entrenamiento, registrar las sesiones de entrenamiento que realiza, descargar informe de las mismas.
- Permitir al usuario almacenar, visualizar y descargar diferentes ejercicios de psicología.
- Permitir al usuario almacenar, visualizar y descargar diferentes ejercicios de fisioterapia.
- Como cliente, permitir al usuario encontrar profesionales expertos en estas cuatro áreas (nutrición, entrenamiento, fisioterapia y psicología) a través de la plataforma, pudiendo crear contratos para interaccionar entre ellos y gestionar las citas.
- Como profesional, ofrecer la oportunidad para darse a conocer en la plataforma y que los usuarios puedan contactar, permitiendo iniciar contratos y concretar citas con ellos, además de asignarles rutinas, dietas, etc.

1.4 Aportaciones

Con el continuo crecimiento del interés de la población en áreas como la nutrición, la salud o el bienestar, surge la inherente necesidad de controlar y monitorizar de alguna manera todo lo que esté relacionado con ello.

Es por esta razón por la que surgen muchas iniciativas, entre las cuales se encuentra “Nutrain”, cuyo objetivo principal es ofrecer al usuario la posibilidad de gestionar y monitorizar diferentes aspectos de su salud a través de una aplicación, de una manera sencilla y eficaz. Como se comprobará más adelante en el manual de usuario, se ha hecho hincapié en los informes y los gráficos. Se considera que es una manera sencilla y visual de ofrecer al usuario información sobre su salud.

Otro aspecto a destacar es que algunas alternativas ya existentes, como las anteriormente nombradas (MyFitnessPal, Cronometer o FatSecret), no incluyen las áreas de la fisioterapia y la psicología. Este proyecto sí

los considera importantes también y por ello los incluye dentro de su abanico de funcionalidades.

Por último, también conviene mencionar que la aplicación “Nutrain” permite a los usuarios relacionarse como clientes y como profesionales, permitiéndoles establecer contratos y gestionar citas entre ellos.

En definitiva, este proyecto plantea soluciones similares a algunas alternativas ya existente, a lo que añade las áreas de psicología y nutrición y también la posibilidad de manejar relaciones cliente-profesional a través de la gestión de contratos y citas.

Capítulo 2: Hardware, Software, configuración e instalación

En esta sección se explicarán los diferentes recursos utilizados para el desarrollo del proyecto. Se distinguen principalmente tres apartados:

1. Recursos hardware
2. Software (sistema operativo, programas, herramientas)
3. Lenguajes de programación, librerías, *frameworks*[1].

2.1 Hardware

El hardware que se ha utilizado para el completo desarrollo de este proyecto ha sido un ordenador portátil MacBook Pro, cuyas características principales son las siguientes:

-Pantalla 'Retina'con resolución de 2880x1800 píxeles
-Procesador Intel i7 4820HQ
-Disco de estado sólido (SSD) de 512 GB.
-Memoria RAM DDR3 de 16 GB a 1600 MHz
-Gráficos Intel Iris Pro (integrados), además de gráficos dedicados con una NVIDIA GeForce GT 750M de 2GB.

2.2 Programas software y herramientas

2.2.1 macOS

Es el sistema operativo por defecto de los ordenadores Apple, y el que, por comodidad y afinidad, se ha utilizado.

[1] En el ámbito software, un *framework* es una estructura conceptual y un conjunto de prácticas y módulos que sirven como base para el desarrollo de aplicaciones específicas.

2.2.2 Visual Studio Code

Es el editor de código que se ha utilizado para el desarrollo del proyecto. Desarrollado por Microsoft, es un editor multiplataforma (disponible para Windows, MacOS y Linux), gratuito y de código abierto.

Incluye numerosas funcionalidades como resaltado de sintaxis, finalización inteligente de código, refactorización, integración con Git para control de versiones, depuración, compartir código, y muchas otras funcionalidades gracias a una extensa colección de plugins que crece día a día.

Está basado en Electron, un *framework* que hace uso de Chromium y Node.js para desarrollar aplicaciones de escritorio usando tecnologías web.

2.2.3 pgAdmin4

Desarrollado con Flask (*framework* de Python), Javascript, jQuery y Bootstrap, es una interfaz gráfica desarrollada para el motor de base de datos PostgreSQL. pgAdmin es una aplicación que nos permite realizar todas las funcionalidades de una base de datos: administración y gestión de bases de datos, realización de consultas, manipulación de datos, volcados y restauraciones, etc.

2.2.4 GitHub

Plataforma web utilizada para manejar proyectos y que utiliza Git. Está disponible tanto en su versión web como en su versión de escritorio, GitHub Desktop. Como dato curioso, cabe destacar que está basado en Ruby on Rails, el mismo *framework* que se ha utilizado para el desarrollo de este proyecto.

2.2.5 StarUML

Herramienta UML gratuita que permite realizar el modelado de un software a través de numerosos diagramas: diagramas de casos de uso, de actividades, de flujo, de clases, etc.

2.2.6 Trello

Trello es un software de administración de proyectos, disponible en versión web, en versión escritorio tanto para Windows como para macOS y en versión móvil para Android e iOS. Es conocido por su sistema de tarjetas, listas y tableros, que permiten trabajar de una manera sencilla, flexible y productiva.

2.3 Lenguajes, librerías y *frameworks*

2.3.1 Git

Git es un sistema de control de versiones diseñado por Linus Torvalds (el creador de Linux). Es de código abierto y está pensado para manejar proyectos tanto de pequeña como de gran escal

2.3.2 PostgreSQL

También conocido como Postgres, es un sistema de gestión de bases de datos relacional, de código abierto y de los más utilizados actualmente.

2.3.3 Ruby on Rails

Desarrollado originalmente por David Heinemeier Hansson, Ruby on Rails es un *framework* de aplicaciones web, de código abierto y escrito en el lenguaje de programación Ruby. Está pensado para aplicaciones que siguen el patrón Modelo-Vista-Controlador (MVC).

Se caracteriza por ser un *framework* donde prima la simplicidad y que se rige principalmente por dos principios: Don't Repeat Yourself (DRY) y Convención sobre Configuración.

La versión utilizada para el desarrollo de este proyecto de fin de grado es la 6.0.2.

2.3.4 Ruby

Lenguaje de programación interpretado y orientado a objetos desarrollado por el japonés Yuhikiro Matsumoto. Se caracteriza por estar enfocado a la simplicidad y tener una sintaxis elegante y muy legible.

La versión utilizada para el desarrollo de este proyecto de fin de grado es la 2.6.3.

2.3.5 NodeJS

NodeJS es un entorno de ejecución multiplataforma, de código abierto, basado en el motor V8 de Chrome y que permite la ejecución de JavaScript fuera del navegador.

2.3.6 JavaScript

Lenguaje de programación interpretado, débilmente tipado, dinámico y orientado a objetos, caracterizado por dar interactividad y dinamismo a las páginas web. Actualmente se utiliza tanto para el desarrollo *front-end* como para el *back-end*, aunque para este proyecto se utilizará solo para el *front-end.*

2.3.7 HTML5

Del inglés *HyperText Markup Language,* es el lenguaje de marcado utilizado para el desarrollo web y que se usa para dar estructura a las páginas web.

2.3.8 CSS3

Del inglés *Cascading Style Sheets,* o hojas de estilo en cascada en español. Es un lenguaje de diseño gráfico para definir el estilo y la presentación de documentos HTML o XML. Se caracteriza principalmente por el uso de propiedades.

2.3.9 jQuery

Librería multiplataforma basada en JavaScript que permite la manipulación de documentos HTML, manejo de eventos, animaciones u otros de una manera sencilla y efectiva.

2.3.10 Bootstrap

Librería de código abierto pensada para el desarrollo de interfaces web, donde prima el desarrollo *responsive* y *mobile-first*. Está enfocado únicamente a la parte *front-end* de una aplicación web, y contiene numerosos componentes como botones, cuadros, tarjetas, menús, etc.

2.4 Configuración e instalación del proyecto

2.4.1 Instalación Git

El primer paso sería instalar el sistema de control de versiones Git. En el siguiente enlace se proporcionan instrucciones de instalación para los distintos sistemas operativos:

Guía instalación de Git [6]: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Lo siguiente, sería crear una carpeta donde vayamos a descargar nuestro proyecto. Una vez dentro de la carpeta, abrimos una terminal y ejecutamos los siguientes comandos:

```
1- $ git init
2- $ git config --global user.name "Tu nombre"
3- $ git config --global user.email tuemail@dominio.com
4- $ git clone @enlace_del_repositorio
```

El primer comando inicializa esa carpeta como un proyecto git (de este modo, puedes utilizar todos los comandos y hacer uso de las funciones de git en ese directorio). Los dos siguientes comandos configuran tu usuario (nombre y email), y el último comando sirve para clonar el repositorio del proyecto en tu directorio local.

2.4.2 Instalación Ruby

En primer lugar, será necesario instalar el lenguaje Ruby:

Guía de instalación Ruby [7]: https://www.ruby-lang.org/es/downloads/

En dicha página se dan las respectivas instrucciones para instalar Ruby según el sistema operativo (Windows, macOS o Linux y sus distribuciones).

Como se ha indicado previamente, la versión a instalar sería la 2.6.3, que es la que se ha utilizado para el desarrollo del proyecto.

2.4.3 Instalación Rails

El siguiente paso sería instalar el *framework* Ruby on Rails, para el cual necesitaremos tener instalado Ruby, descrito en el punto anterior.

En el siguiente enlace se proporcionan instrucciones para instalar Rails en los distintos sistemas operativos:

Guía para instalar Rails [8]: https://www.tutorialspoint.com/ruby-on-rails/rails-installation.htm

La versión a instalar en este caso sería la 6.0.2.3 que es la que se ha utilizado para el desarrollo del proyecto. Una vez instalado, en directorio del proyecto ejecutaremos el siguiente comando:

```
$ bundle install
```

Este comando instalará todas las librerías y dependencias necesarias (lo que en Rails se conoce como gemas) a través del bundler de Rails.

2.4.4 Instalación NodeJS

En este paso, se instalará NodeJS, entorno de ejecución necesario para JavaScript. En el siguiente enlace se proporcionan diferentes

instaladores según el sistema operativo. Una vez descargado, sería seguir los pasos del instalador.

Guía para instalar NodeJS [9]: https://nodejs.org/es/

2.4.5 PostgreSQL

Para la base de datos, instalaremos PostgreSQL. Para ello, iremos al siguiente enlace:

Guía para instalar PostgreSQL [10]: https://www.postgresql.org/download/

Escogeremos la opción *"Interactive installer by EDB"*, que ya trae PostgreSQL y pgAdmin. La versión de PostgreSQL a instalar será la 11.

Durante la instalación de PostgreSQL, en uno de los pasos se nos pedirá introducir un usuario y contraseña, que serán los que se utilizarán para conectarse a la base de datos.

Dentro de la carpeta raíz del proyecto, crearemos un archivo local_env.yml.

En dicho archivo, pondremos el siguiente contenido:

```
1. DB_USER_NAME: usuario_postgres
2. DB_PASSWORD: contraseña_postgres
```

En "usuario_postgres" y "contraseña_postgres" deberás poner los valores que previamente introduciste durante la instalación de PostgreSQL.

2.4.6 Configuración del editor VSCode y setup del proyecto

Por último, nos quedará instalar Visual Studio Code, que es dónde se ha desarrollado el proyecto en su totalidad.

Cabe destacar que no es obligatorio utilizar VSCode, se pueden utilizar otras opciones como Ruby Mine, Sublime Text, Atom u otros. No obstante, yo recomiendo utilizar VSCode por su usabilidad, cantidad de *plugins* y la gran comunidad que tiene.

Para instalarlo, accedemos al siguiente enlace y descargamos la opción para nuestro sistema operativo:

Guía para instalar VSCode [11]: https://code.visualstudio.com/download

Una vez instalado, abriremos la aplicación, y en el menú lateral de los *plugins*, se recomienda instalar los siguientes:

- Ruby
- Ruby on Rails
- Simple Ruby ERB
- VSCode Ruby
- HTML CSS Support
- HTML Snippets
- endwise
- ERB Formatter/Beautify
- GitLens
- Git History
- Auto Close Tag

Todos estos *plugins* proporcionan una gran ayuda a la hora de desarrollar la aplicación: autocompletado de código, soporte para Git, formatear código, *code snippets,* entre otras.

Si se ha instalado todo correctamente, solo quedaría arrancar inicializar la base de datos y arrancar el servidor. Para ello, con una terminal en el directorio del proyecto, ejecutamos los siguientes comandos:

```
$ rails db:create db:migrate db:seed
```

Este comando son tres comandos separados pero que se pueden ejecutar en uno sólo. Lo que hace es crear la base de datos (*db:create)*, realizar las migraciones *(db:migrate)* e inicializar la aplicación con algunos valores, lo que se conoce como fichero *seeder (db: seed)*.

2 *Code snippet* es una función para reutilizar código. En los editores, lo más común es que escribiendo un comando corto te lo sustituya por un trozo de código más grande.

Este último es opcional si lo que se quiere es la base de datos completamente vacía. Para añadir o quitar más información con la que se quiera relleanr inicialmente la base de datos, basta con editar el fichero db/seed.rb

Por último, levantamos el servidor:

```
$ rails server
```

Este comando inicia un servidor que ya viene instalado junto con Rails, y si accedemos a http://localhost:3000 deberíamos visualizar la aplicación si hemos realizado todos los pasos correctamente.

Capítulo 3: Plan de trabajo

En este capítulo se explicará la planificación del trabajo, desglosando las diferentes partes del desarrollo y las horas dedicadas a cada una, las metodologías utilizadas para el desarrollo del proyecto y un análisis del plan de negocio.

3.1 Planificación del proyecto

3.1.1 Temporalización del proyecto

A continuación, se muestra en el cuadro 1 las diferentes fases del proyecto y el desglose de horas y tareas principales correspondiente a cada fase.

Fases	***Duración Estimada*** *(horas)*	***Tareas*** *(nombre y descripción, obligatorio al menos una por fase)*
Estudio previo / Análisis	30	Tarea 1.1: Análisis del dominio del problema y de la posible funcionalidad de la aplicación. Análisis sobre las distintas tecnologías a utilizar en cada caso.
		Tarea 1.2: Análisis y especificación de los requisitos y alcance de la aplicación, así como diseño de los bocetos de sus interfaces.
Diseño / Desarrollo / Implementación	150	Tarea 2.1: Diseño de las interfaces de usuario de la app, la navegación entre ellas e implementación del código necesario.
		Tarea 2.2: Diseño de la base de datos a utilizar en el back-end donde se analizarán las tablas necesarias y los campos que la forman así como las relaciones entre estas tablas.
		Tarea 2.3: Desarrollo del back-end e implementación del acceso al mismo desde el front-end (app).
Evaluación / Validación / Prueba	60	Tarea 3.1: Evaluación del rendimiento y funcionalidad de la aplicación y optimización de la misma en caso de ser necesario
		Tarea 3.2: Batería de pruebas y tests para la validación de la aplicación.
Documentación / Presentación	60	Tarea 4.1: Documentación final del TFG.
		Tarea 4.2: Preparación de la presentación y defensa del TFG.

Cuadro 1: Tabla de planificación del proyecto desglosada por horas. Fuente: elaboración propia.

3.1.2 Metodología

Para el desarrollo del proyecto se ha aplicado una metodología ágil utilizando principalmente la metodología Kanban, cuya definición es la siguiente:

"Kanban es una palabra japonesa que significa "tarjetas visuales", donde Kan es "visual", y Ban corresponde a "tarjeta".

Las principales ventajas de esta herramienta es que es muy fácil de utilizar, actualizar y asumir por parte del equipo. Además, destaca por ser una técnica de gestión de las tareas muy visual, que permite ver a golpe de vista el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

- **Calidad garantizada.** Todo lo que se hace debe salir bien a la primera, no hay margen de error. De aquí a que en Kanban no se premie la rapidez, sino la calidad final de las tareas realizadas. Esto se basa en el hecho que muchas veces cuesta más arreglarlo después que hacerlo bien a la primera.

- **Reducción del desperdicio.** Kanban se basa en hacer solamente lo justo y necesario, pero hacerlo bien. Esto supone la reducción de todo aquello que es superficial o secundario.

- **Mejora continua.** Kanban no es simplemente un método de gestión, sino también un sistema de mejora en el desarrollo de proyectos, según los objetivos a alcanzar.

- **Flexibilidad.** Lo siguiente a realizar se decide del *backlog* (o tareas pendientes acumuladas), pudiéndose priorizar aquellas tareas entrantes según las necesidades del momento (capacidad de dar respuesta a tareas imprevistas)."

Fuente: IEBSchool [13]

Como puede verse, Kanban es una metodología realmente cómoda y útil a la hora de trabajar, porque en un vistazo rápido tienes una visión general del estado de las tareas y es bastante sencilla de utilizar y entender.

En la figura 1 podemos observar un ejemplo de cómo se ha utilizado dicha herramienta en el desarrollo del proyecto haciendo uso de la aplicación Trello:

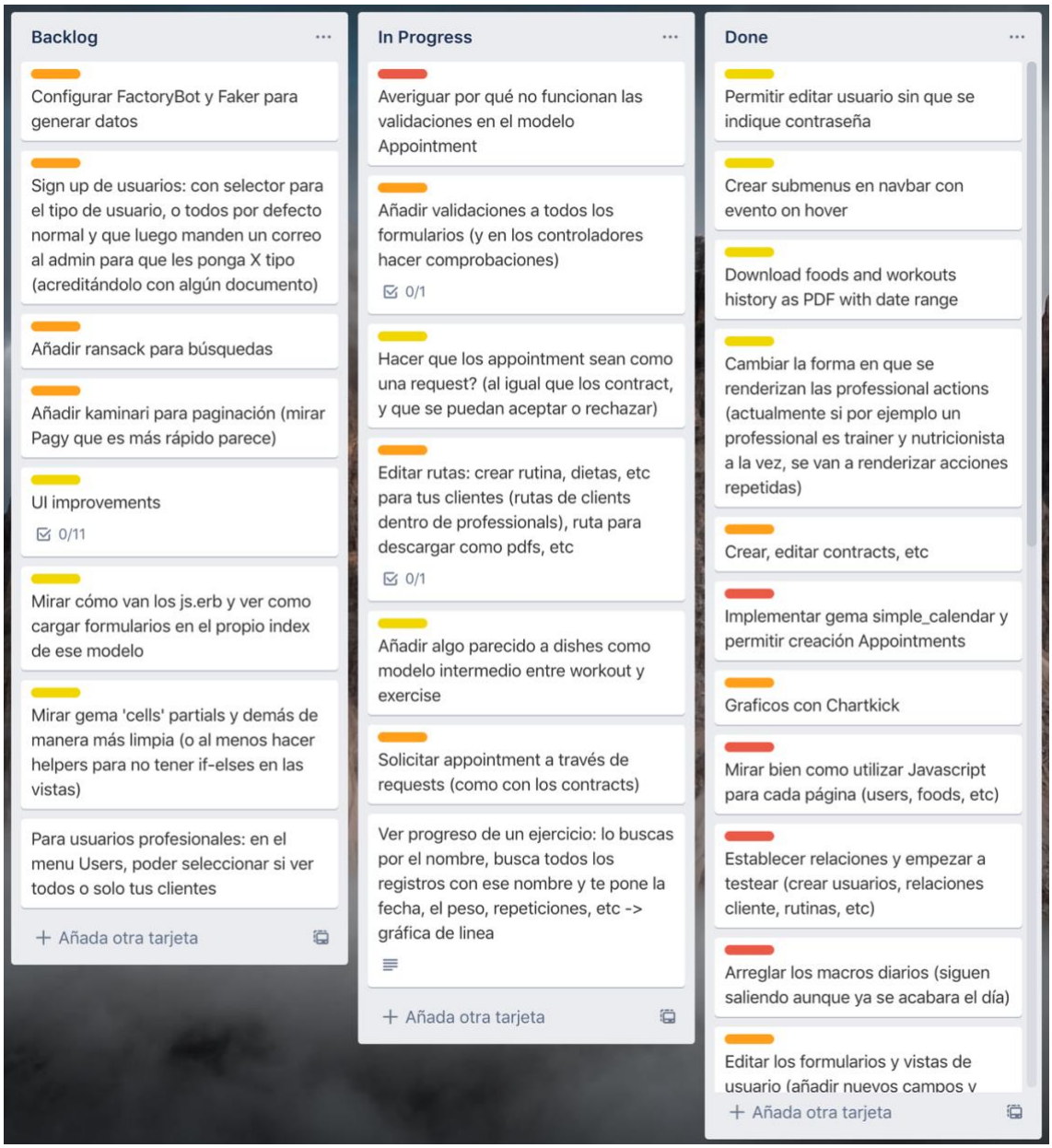

Figura 1: ejemplo de Kanban durante el proyecto en la aplicación Trello. Fuente: elaboración propia.

Se ha utilizado un código de colores según la prioridad de la tarea:

- Rojo: prioridad alta.
- Naranja: prioridad normal.
- Amarillo: prioridad baja.

3.2 Plan de negocio

A continuación, se expondrán los costes relacionados con el desarrollo del proyecto, desglosando y explicando cada uno de ellos.

3.2.1 Costes hardware

En el cuadro 2 se desglosan los costes hardware, que consisten en todo el equipamiento informático necesario para que el equipo de desarrollo pueda realizar su actividad de manera eficiente, buscando el mejor equilibrio posible entre precio y rendimiento/prestaciones.

Equipo hardware	**Precio (€)**
Ordenador x 4	900 x 4 = 3600
Monitor x 4	100 x 4 = 400
Teclado + Ratón x 4	30 x 4 = 120
Total	4120

Cuadro 2: Tabla de costes hardware. Fuente: elaboración propia.

3.2.2 Costes software

Los costes relacionados con el software son 0, dado que para cada una de las partes implicadas en el desarrollo del proyecto se han utilizado herramientas gratuitas. No obstante, conviene tener presente la posibilidad de utilizar algún software de pago si fuera necesario (IDE's, herramientas de *testing* u otros).

3.2.3 Costes personal

A continuación, se muestra en el cuadro 3 los costes relacionados con el personal contratado para poder desarrollar el proyecto.

Puesto	**Sueldo (€)**
Jefe de proyecto	3500
Analista	2500
Diseñador	1800
Programador	2000
Total	9800

Cuadro 3: Tabla de costes de personal. Fuente: elaboración propia.

3.2.4 Otros costes

En este apartado se incluirán los gastos considerados como básicos, tales como agua, luz, internet y alquiler de oficinas o algún tipo de espacio correctamente habilitado para el trabajo, tal y como se observa en el cuadro 4.

Concepto	Coste (€)
Agua	40
Luz	90
Internet	80
Oficina y material oficina	1500
Total	1710

Cuadro 4: Tabla de costes de otros conceptos. Fuente: elaboración propia.

El resumen de todos los costes es el siguiente que podemos observar en el cuadro 5:

Concepto	Coste
Costes hardware	4120
Costes software	0
Costes personal	9800
Otros costes	1710
Total	15630

Cuadro 5: Tabla de costes totales. Fuente: elaboración propia.

Esta cantidad total equivale a un mes, por lo que habría que multiplicarlo por el periodo de tiempo en que se extendiera este proyecto. Se propone una extensión de 6 meses, por lo que el gasto final sería el siguiente: 6 x 15630 = 93780 euros.

Para obtener ingresos a través de la aplicación, se plantean principalmente dos vías:

1. **Publicidad:** se deberá incluir anuncios publicitarios en la página web, y, a ser posible, de compañías relacionadas con el ámbito de la aplicación (por ejemplo, anuncios de páginas web para comprar suplementos deportivos, anuncios de institutos de profesionales del sector, etc.).

2. **Suscripción Premium:** sería interesante integrar una alternativa Premium, donde el usuario disfrute de funcionalidades extras que las que se incluyen por defecto (por ejemplo, análisis mucho más detallados de los macronutrientes, opciones de personalización, etc.).

También conviene considerar otras opciones, como por ejemplo campañas de financiación como Kickstarter, GoFundMe o CrowdFunder. Otra opción podría ser investigar las diferentes subvenciones a nivel nacional para comprobar si el desarrollo y el ámbito del proyecto encaja en algún tipo de subvención.

Capítulo 4: Desarrollo del proyecto

En este capítulo se abordará una parte importante del proyecto, que es el desarrollo. Se detallará toda la información necesaria para la comprensión del desarrollo del proyecto, tales como: lista de requisitos (*product backlog),* diagrama de base de datos, diagrama de casos de uso, diagrama de clases u otros esquemas, diagramas e información que se requiera para la total comprensión del capítulo.

4.1 Requisitos

4.1.1 Product Backlog

En esta sección, se detallarán la lista de requisitos obtenidos durante la fase de diseño y análisis del software, los cuales contienen las funcionalidades implementadas en la aplicación web y que se recogen en el cuadro 6.

Es importante destacar que existen tres tipos de roles, que son el Usuario (entidad principal), y Cliente y Profesional (entidades derivadas de Usuario), por lo que todas las funciones con rol Usuario la podrán realizar ambos, y las funciones con rol Cliente o Profesional son específicas de ese modelo.

También se debe tener en cuenta que, como se explicará en el apartado 4.2.1, correspondiente al diseño de la base de datos, en los usuarios tipo Profesional hay cuatro campos que tienen importancia: *nutritionist* (nutricionista), *trainer* (entrenador), *psychologist* (psicólogo), *physiotherapist* (fisioterapeuta). De este modo, cada profesional podrá realizar las acciones correspondientes a cada tipo de profesional (si por ejemplo un profesional es nutricionista y entrenador, entonces sólo puede hacer acciones relacionadas con nutrición y entrenamiento, pero no con psicología o fisioterapia).

Nombre	**Rol**	**Resultado**
Registrarse	Usuario	El usuario podrá crear una cuenta con la que

		accederá a la página para poder hacer uso de todas sus funcionalidades.
Iniciar sesión	Usuario	El usuario podrá iniciar sesión con la cuenta con la que se haya registrado.
Cerrar sesión	Usuario	El usuario podrá cerrar la sesión en cualquier momento.

Borrar cuenta	Usuario	El usuario podrá cancelar su cuenta si no va a hacer más uso de ella.
Editar perfil	Usuario	El usuario podrá editar la información de su perfil.
Mostrar perfil	Usuario	El usuario podrá visualizar los datos de su perfil.
Buscar profesionales	Cliente	El cliente podrá buscar profesionales que estén registrados en la página.
Pedir cita profesional	Cliente	El cliente podrá pedir una cita al profesional en el que esté interesado.
Solicitar contrato	Cliente	El cliente podrá solicitar el inicio de una relación contractual con el profesional que le interesa que le mentorice.
Ver detalles profesional	Cliente	El cliente podrá ver todos los detalles del usuario profesional.

Buscar clientes	Profesional	El profesional podrá ver la lista de posibles clientes registrados en la página.
Pedir cita cliente	Profesional	El profesional podrá pedir una cita con un cliente en caso de que le interese.
Editar cita	Usuario	El usuario podrá editar los datos de una cita.
Borrar cita	Usuario	El usuario podrá borrar una cita ya creada.
Ver listado citas	Usuario	El usuario podrá ver un listado de sus citas.
Descargar informe citas	Usuario	El usuario podrá descargar un informe PDF de sus citas.
Solicitar contrato	Profesional	El profesional podrá solicitar un contrato a un cliente en el que esté interesado mentorizar.
Ver detalles clientes	Profesional	El profesional podrá ver los detalles del usuario cliente.
Ver lista contratos	Usuario	El usuario podrá ver la lista de contratos que tiene, tanto si están pendientes como si están activos.
Descargar informe contratos	Usuario	El usuario podrá descargar un informe PDF de sus contratos.

Crear dieta	Usuario	El usuario podrá crear una dieta.
Crear dieta cliente	Profesional	El profesional podrá crearle una dieta a un cliente con el que tenga contrato.
Editar dieta	Usuario	El usuario podrá editar una dieta creada.
Editar dieta cliente	Profesional	El profesional podrá editar la dieta de un cliente con el que tenga contrato (solamente la que tenga activa).
Borrar dieta	Usuario	Un usuario podrá borrar una dieta, independientemente de si es creada por él o por un profesional.
Ver listado dietas	Usuario	El usuario podrá visualizar el listado de dietas que ha registrado.
Descargar informe dietas	Usuario	El usuario podrá descargar un informe PDF con información de las dietas.
Crear rutina	Usuario	Un usuario podrá crear una rutina de entrenamiento.
Editar rutina	Usuario	Un usuario podrá editar una rutina creada.
Borrar rutina	Usuario	Un usuario podrá borrar una rutina creada, independientemente de si es creada por él o por un profesional.

Ver listado rutinas	Usuario	El usuario podrá visualizar el listado de rutinas que ha registrado.
Descargar informe rutinas	Usuario	El usuario podrá descargar un informe PDF con información sobre las rutinas.
Crear rutina cliente	Profesional	Un profesional podrá crearle una rutina de entrenamiento a un cliente.
Editar rutina cliente	Profesional	Un profesional podrá editar la rutina de su cliente (solamente la que tenga activa)
Añadir comida	Usuario	Un usuario podrá registrar una comida.
Editar comida	Usuario	El usuario podrá editar una comida ya creada.
Borrar comida	Usuario	El usuario podrá borrar una comida ya creada.
Ver historial comidas	Usuario	El usuario podrá visualizar el listado de todas las comidas registradas.
Descargar informe comidas	Usuario	El usuario podrá descargar un informe PDF con los datos de las comidas.
Crear alimento	Usuario	El usuario podrá crear un alimento.
Editar alimento	Usuario	El usuario podrá editar un alimento ya creado.
Borrar alimento	Usuario	El usuario podrá borrar un alimento.
Ver listado alimentos	Usuario	El usuario podrá ver un listado de todos los

		alimentos que ha registrado.
Descargar informe alimentos	Usuario	El usuario podrá descargar un informe PDF con información de los alimentos que ha registrado.
Añadir entrenamiento	Usuario	El usuario podrá registrar una sesión de entrenamiento.
Editar entrenamiento	Usuario	El usuario podrá editar un entrenamiento ya creado.
Borrar entrenamiento	Usuario	El usuario podrá registrar un entrenamiento.
Ver listado entrenamientos	Usuario	El usuario podrá visualizar la lista de entrenamientos que ha registrado.
Descargar informe entrenamientos	Usuario	El usuario podrá descargar un informe PDF con información sobre los entrenamientos.
Crear ejercicio	Usuario	El usuario podrá crear un ejercicio.
Editar ejercicio	Usuario	El usuario podrá editar un ejercicio ya creado.
Borrar ejercicio	Usuario	El usuario podrá borrar un ejercicio.
Ver listado ejercicios	Usuario	El usuario podrá ver un listado de los ejercicios que ha registrado.
Descargar informe ejercicios	Usuario	El usuario podrá descargar un informe

		PDF con información de los ejercicios.
Crear ejercicio psicología	Usuario	El usuario podrá crear un ejercicio de psicología.
Editar ejercicio psicología	Usuario	El usuario podrá editar un ejercicio de psicología ya creado.
Borrar ejercicio psicología	Usuario	El usuario podrá borrar un ejercicio de psicología.
Ver listado ejercicios psicología	Usuario	El usuario podrá ver un listado de sus ejercicios de psicología.
Descargar informe ejercicios psicología	Usuario	El usuario podrá descargar un informe PDF sobre los ejercicios de psicología.
Crear ejercicio psicología a cliente	Profesional	El profesional podrá crear un ejercicio de psicología a su cliente.
Crear ejercicio fisioterapia	Usuario	El usuario podrá crear un ejercicio de fisioterapia.
Editar ejercicio fisioterapia	Usuario	El usuario podrá editar un ejercicio de fisioterapia.
Borrar ejercicio fisioterapia	Usuario	El usuario podrá borrar un ejercicio de fisioterapia.
Ver listado ejercicios fisioterapia	Usuario	El usuario podrá ver un listado de los ejercicios de fisioterapia creados por él.
Descargar informe ejercicios fisioterapia	Usuario	El usuario podrá descargar un informe PDF de los ejercicios de fisioterapia.

Crear ejercicio fisioterapia cliente	Profesional	El profesional podrá crear un ejercicio de fisioterapia a un cliente con el que tenga contrato.
Añadir medidas corporales	Usuario	El usuario podrá crear añadir un conjunto de medidas corporales.
Editar medidas corporales	Usuario	El usuario podrá editar unas medidas corporales ya creadas.
Borrar medidas corporales	Usuario	El usuario podrá borrar medidas corporales.
Ver listado medidas corporales	Usuario	El usuario podrá ver un listado de sus medidas corporales.
Descargar informe medidas corporales	Usuario	El usuario podrá descargar un informe PDF de sus medidas corporales.
Ver gráficos medidas corporales	Usuario	El usuario podrá ver distintos gráficos de sus medidas corporales, seleccionado diferentes intervalos de tiempo.
Enviar correo página	Usuario	El usuario podrá contactar con la página a través de un correo.

Cuadro 6: Tabla de requisitos del software. Fuente: elaboración propia.

4.1.2 Casos de uso de la aplicación

A continuación, se mostrarán los diferentes diagramas de casos de uso de la aplicación, según el usuario que los realiza. Por comodidad y para facilitar la visualización, dividiremos el diagrama en tres partes: Usuario, Profesional y Cliente (por esta razón, en la figura 2 se observan dos líneas hacia abajo, que son precisamente las que vienen de las entidades derivadas de Usuario, que son las entidades Profesional y Cliente).

En primer lugar, tenemos el diagrama de casos de uso para el Usuario general. Como ya se ha explicado previamente, aquí se contemplan las funciones que son comunes a todos los usuarios.

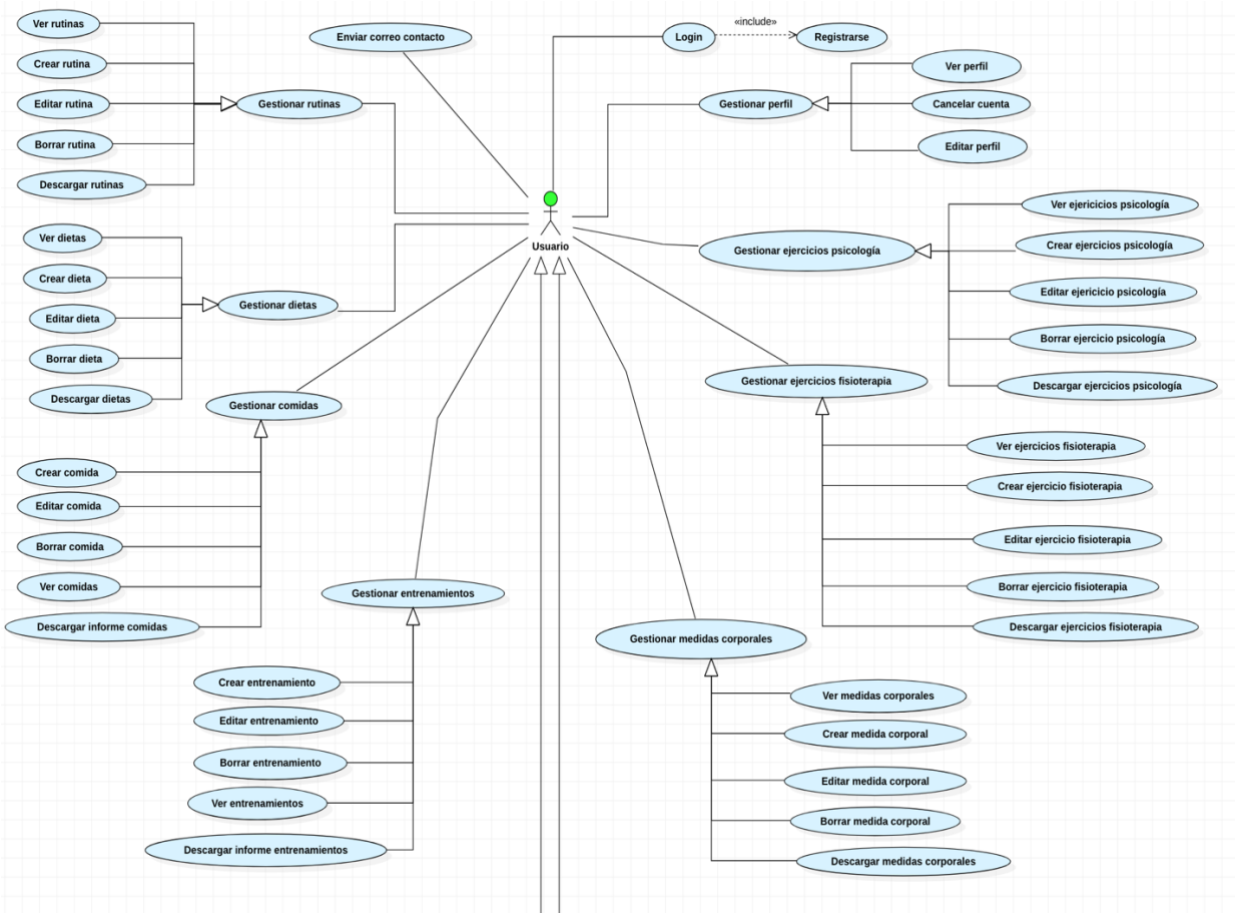

Figura 2: Diagrama de casos de uso del Usuario. Fuente: elaboración propia.

En segundo lugar, tenemos el diagrama de casos de uso del usuario Profesional. En la figura 3 se puede observar que del usuario Profesional salen cuatro "usuarios" más: como ya se ha explicado en el anterior

apartado, son cuatro campos que posee el usuario Profesional y que según cuáles tenga activos, tendrá unas funcionalidades u otras.

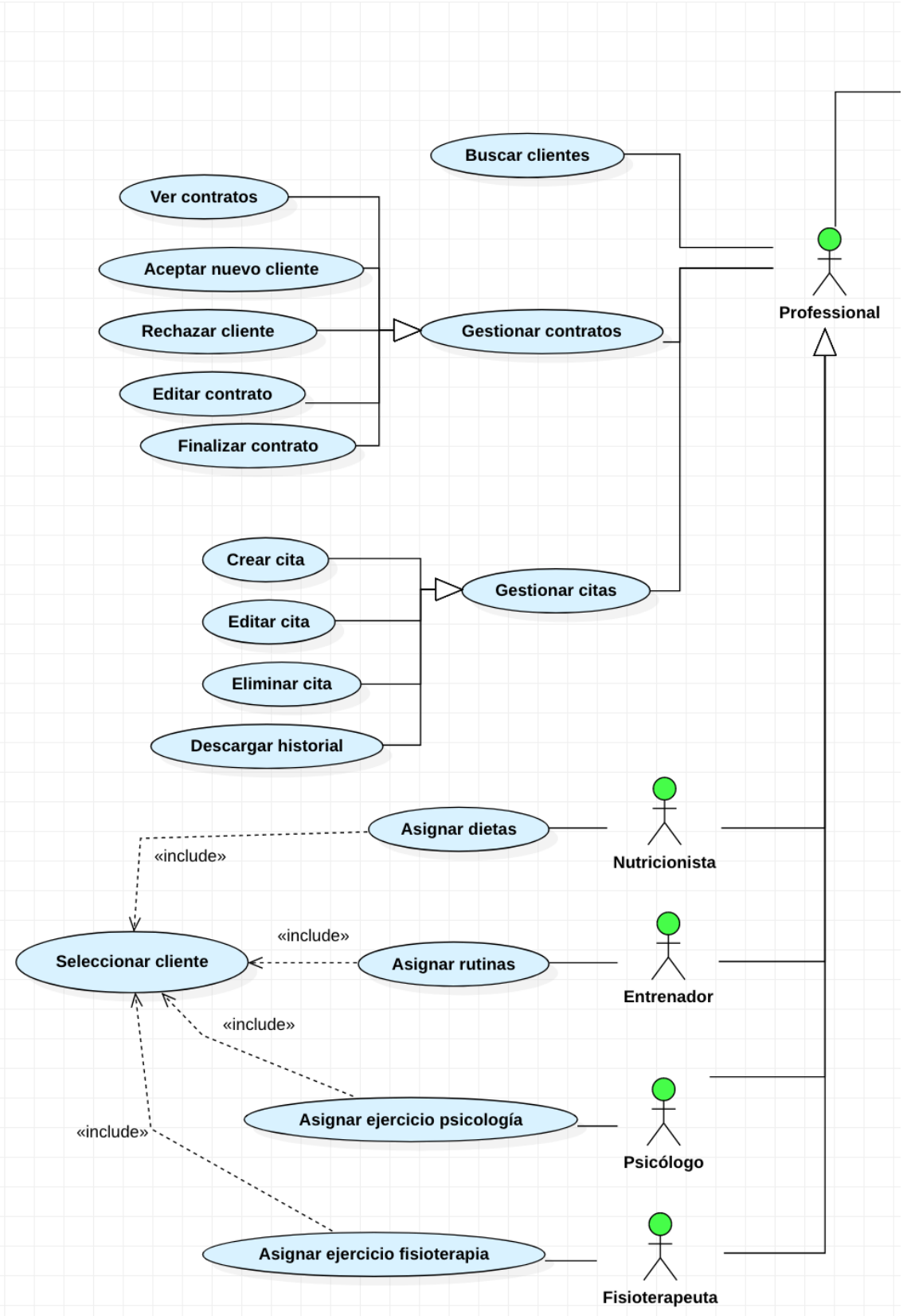

Figura 3: Diagrama de casos de uso del Profesional. Fuente: elaboración propia.

Por último, tenemos el diagrama de casos de uso del usuario Cliente, como se puede observar en la figura 4.

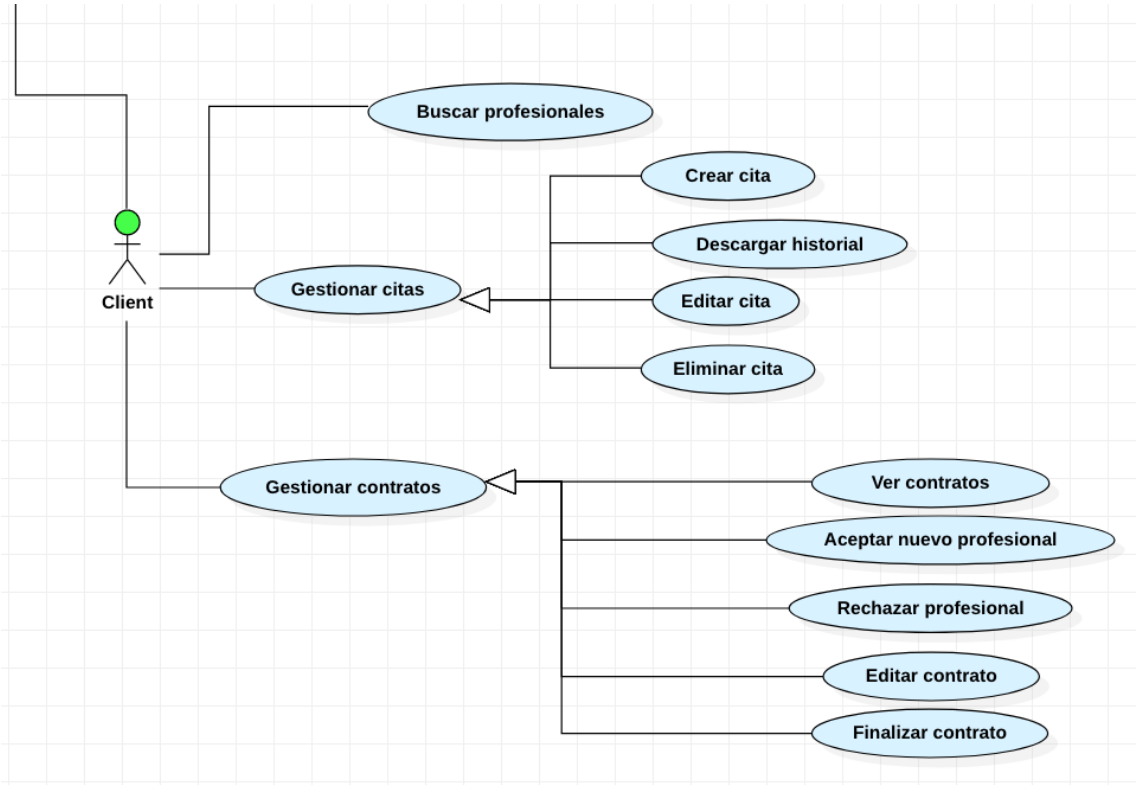

Figura 4: Diagrama de casos de uso del Cliente. Fuente: elaboración propia.

4.2 Diseño

4.2.1 Base de datos

A continuación, se procederá a explicar las tablas de la base de datos que podemos observar en la figura 5. Se explicará por qué se han relacionado las tablas de dicha manera, además de qué tipo de información guardan los campos que incluyen y por qué se han hecho algunas elecciones en el diseño.

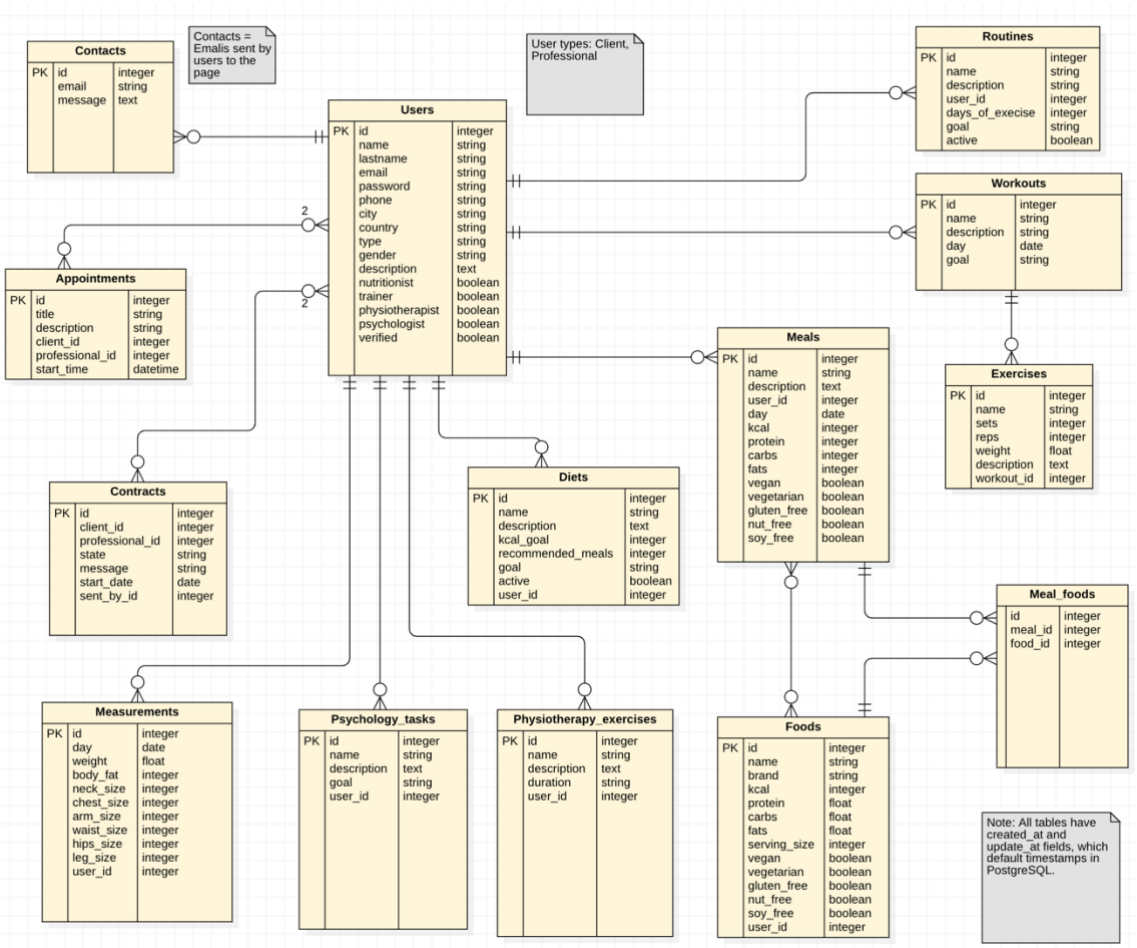

Figura 5: Diagrama de la base de datos. Fuente: elaboración propia.

Tabla *Users* (Usuarios)

Dentro de los usuarios, distinguimos dos tipos: clientes y profesionales. Ambos tienen los mismos campos en la base de datos, pero con la diferencia de que los campos *nutritionist, trainer, psychologist, physiotherapist* y *verified* en un usuario de tipo cliente siempre van a tener valor "null", mientras que en un usuario profesional estarán a "true" según el tipo de profesional que sea dicho usuario. Por ejemplo, un profesional nutricionista tendrá el campo *nutritionist* con valor "true".

Este planteamiento lógico es muy común y se conoce como *STI* (*Single Table Inheritance*) y consiste en utilizar una sola tabla para las entidades derivadas (Cliente y Profesional) de una entidad padre (Usuario).

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre del usuario.
- Lastname: apellidos del usuario.
- Email: email del usuario. Es un campo único para cada usuario.
- Password: contraseña del usuario.
- Gender: género (masculino o femenino).
- Age: edad del usuario.
- Phone: número de teléfono para el usuario.
- City: ciudad donde reside el usuario.
- Country: país donde reside el usuario.
- Description: campo que se reserva para que el usuario aporte la información que crea conveniente que los profesionales deban saber (alergias, lesiones, algún dato importante de su historial, preferencias, objetivos, etc).
- Trainer: estará a "true" si el profesional es un entrenador personal.
- Nutritionist: estará a "true" si el profesional es un nutricionista.
- Physiotherapist: estará a "true" si el profesional es un fisioterapeuta.
- Psychologist: estará a "true" si el profesional es un psicólogo.
- Verified: estará a "true" si el profesional ha enviado un correo a la plataforma con la respectiva documentación que acredita que es un profesional en su campo.

Tabla *Measurements* (Medidas corporales)

En esta tabla se almacena la información de las medidas corporales que introduzca el usuario. Un usuario puede introducir muchas medidas corporales, pero cada medida corporal pertenece a un único usuario. Cada registro de esta tabla contiene la siguiente información:

- Id: campo autonumérico y clave primaria de la tabla.
- Day: día en el que se ha registrado un nuevo conjunto de medidas corporales.
- Weight: peso del usuario en ese día.
- Neck_size: diámetro del cuello medido en centímetros en ese día.
- Chest_size: diámetro del pecho medido en centímetros en ese día.
- Arm_size: diámetro del brazo medido en centímetros en ese día.

- Waist_size: diámetro de la cintura medido en centímetros en ese día.
- Hips_size: diámetro de las caderas medido en centímetros en ese día.
- Leg_size: diámetro de la pierna (generalmente en la zona del cuádriceps) medido en centímetros en ese día.
- User_id: id del usuario al que pertenece dicho conjunto de medidas corporales.

Tabla *Routines* (Rutinas de entrenamiento)

Esta tabla almacena información sobre las rutinas que quiera guardar el usuario. Un usuario puede guardar todas las rutinas que quiera y cada rutina pertenece a un único usuario. En ella se almacena la siguiente información:

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre de la rutina.
- Description: descripción de la rutina (cuánto tiempo vas a estar con la rutina, ejercicios y su explicación, etc).
- User_id: id del usuario al que pertenece la rutina.
- Days_of_exercise: número de días de entrenamiento de dicha rutina.
- Goal: objetivo de la rutina (hipertrofia, aumento de fuerza, etc).
- Active: indica si la rutina es la que está usando el usuario actualmente. Un usuario sólo puede tener una rutina como "Active", puesto que solo debería estar siguiendo una única rutina de entrenamiento.

Tabla *Workouts* (Entrenamiento)

Almacena información sobre el propio entrenamiento en sí: nombre, número de ejercicios, etc. Un usuario puede guardar todos los *workouts* que desee y cada *workout* pertenece a un único usuario. Un *workout* se compone de ejercicios.

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre del entrenamiento (ejemplo: entrenamiento de espalda y bíceps).

- Description: descripción del entrenamiento (anotaciones, descanso entre ejercicios, etc.)
- Day: día en que se realizó el entrenamiento.
- User_id: id del usuario al que pertenece.
- Number_of_exercises: número de ejercicios que tiene el entrenamiento
- Goal: objetivo del entrenamiento

Tabla *Exercises* (Ejercicios)

Esta tabla almacena información sobre cada uno de los ejercicios que componen un entrenamiento.

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre del ejercicio (ejemplo: dominadas)
- Description: descripción del ejercicio (cómo hacerlo, advertencias, etc)
- Sets: número de series del ejercicio.
- Reps: número de repeticiones del ejercicio.
- Weight: peso con el que se ha realizado el ejercicio.
- Workout_id: id del workout al que pertenece.

Tabla *Diets* (Dietas)

Esta tabla almacena información sobre las dietas que quiera guardar el usuario. Un usuario puede guardar todas las dietas que quiera y cada dieta pertenece a un único usuario. En ella se almacena la siguiente información:

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre de la dieta (ejemplo: dieta cetogénica).
- Descripción: descripción de la dieta (ejemplo: dieta alta en proteínas y grasas, con muy baja cantidad de carbohidratos).
- User_id: id del usuario al que pertenece la rutina.
- Kcal_goal: cantidad de calorías diarias que se deben ingerir en esta dieta (ejemplo: 2500 kcal).

- Recommended_meals: número de comidas que se recomienda hacer siguiendo esta dieta (por ejemplo, si una dieta es alta en calorías, lo lógico será repartirlo en muchas comidas).
- Goal: objetivo de la dieta (ganar peso, perder peso, mantenimiento).
- Active: indica si la dieta es la que está siguiendo el usuario actualmente. Un usuario sólo puede tener una dieta como "Active", puesto que solo debería estar siguiendo una única dieta.

Tabla *Meals* (Comidas)

En esta tabla se almacena la información de cada comida que introduzca el usuario. Un usuario podrá introducir muchas comidas, pero cada comida pertenece a un único usuario. Cada comida se compone de uno o más alimentos/ingredientes. En dicha tabla, se guarda la siguiente información:

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre de la comida.
- Description: descripción que quiera añadir el usuario.
- User_id: id del usuario al que pertenece.
- Day: día en el que realizó dicha comida.
- Kcal: kilocalorías totales de la comida. Se calcula como la suma de las kilocalorías de cada alimento/ingrediente que compone dicha comida.
- Protein: proteínas totales de la comida. Se calcula como la suma de las proteínas de cada alimento/ingrediente que compone dicha comida.
- Carbs: carbohidratos totales de la comida. Se calcula como la suma de los carbohidratos de cada alimento/ingrediente que compone dicha comida.
- Fats: grasas totales de la comida. Se calcula como la suma de las grasas de cada alimento/ingrediente que compone dicha comida.
- Vegan: indica si una comida es vegana (sin alimentos de origen animal). Estará a "true" si todos los alimentos que componen la comida también tienen dicho atributo a "true".
- Vegetarian: indica si una comida es vegetariana (sólamente los siguientes alimentos de origen animal: lácteos y derivados, huevos). Estará a "true" si todos los alimentos que componen la comida también tienen dicho atributo a "true".

- Gluten_free: indica si una comida no contiene gluten. Estará a "true" si todos los alimentos que componen la comida también tienen dicho atributo a "true".
- Nut_free: indica que la comida no contiene frutos secos. Estará a "true" si todos los alimentos que componen la comida también tienen dicho atributo a "true".

Tabla *Foods* (Alimentos/Ingredientes)

Esta tabla es similar a la tabla de Meals (Comidas). Los alimentos forman comidas y cada alimento puede aparecer en diferentes comidas. Almacena la siguiente información:

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre de la comida.
- Brand: marca del alimento.
- User_id: id del usuario al que pertenece.
- Kcal: kilocalorías totales del alimento por 100 gramos.
- Protein: proteínas totales del alimento por 100 gramos.
- Carbs: carbohidratos totales del alimento por 100 gramos.
- Fats: grasas totales del alimento por 100 gramos.
- Vegan: estará a true si el alimento es vegano (sin alimentos de origen animal).
- Vegetarian: estará a true si el alimento es vegetariano (sólamente los siguientes alimentos de origen animal: lácteos y derivados, huevos).
- Gluten_free: estará a true si el alimento no contiene gluten.
- Nut_free: estará a true si el alimento no contiene frutos secos.
- Soy_free: estará a true si el alimento no contiene soja.

Tabla *Psychology_Exercises* (Ejercicios psicología)

Esta tabla contiene los registros de ejercicios de psicología que crean los usuarios.

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre del ejercicio.
- Description: descripción del ejercicio.
- User id: id del usuario al que pertenece.

Tabla *Physiotherapy Exercises* (Ejercicios fisioterapia)

Esta tabla contiene los registros de ejercicios de fisioterapia creados por los usuarios.

- Id: campo autonumérico y clave primaria de la tabla.
- Name: nombre del ejercicio.
- Description: descripción del ejercicio.
- Duration: duración del ejercicio.

Tabla *Contracts* (Contratos)

Almacena la información de los contratos que se establecen entre clientes y profesionales.

- Client id: id del cliente que participa en el contrato.
- Professional id: id del profesional que participa en el contrato.
- State: estado del contrato (Pendiente o Activo).
- Message: mensaje inicial que se manda en la solicitud del contrato.
- Sent by id: id del usuario que mandó la solicitud.

Tabla *Appointments* (Citas)

Esta tabla almacena información sobre las citas que se fijan entre clientes y profesionales.

- Client id: id del cliente.
- Professional id: id del profesional.
- Title: título para la cita (ejemplo: revisión trimestral con X cliente).
- Description: descripción o información adicional necesaria (ejemplo: quedamos en mi consulta, llámame si no funciona el timbre).
- Start time: hora a la que empieza la cita.

Tabla *Contacts* (Correos de contacto)

En esta tabla se guardan todos los mensajes que envían los usuarios a través de la sección de contacto, por ejemplo, para comunicar errores, problemas, preguntar dudas, etc.

- Email: email del usuario que lo envía.
- Message: contenido del correo.

4.3 Implementación

En esta sección se abordarán las diferentes partes de las que se compone una aplicación Ruby on Rails, explicando la estructura MVC (Modelo-Vista-Controlador) y algunas partes relevantes del código.

4.3.1 ¿Qué es MVC?

Según Wikipedia [14], "el Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.[12] Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento."

En la figura 6 podemos observar un esquema de cómo funciona el patrón MVC:

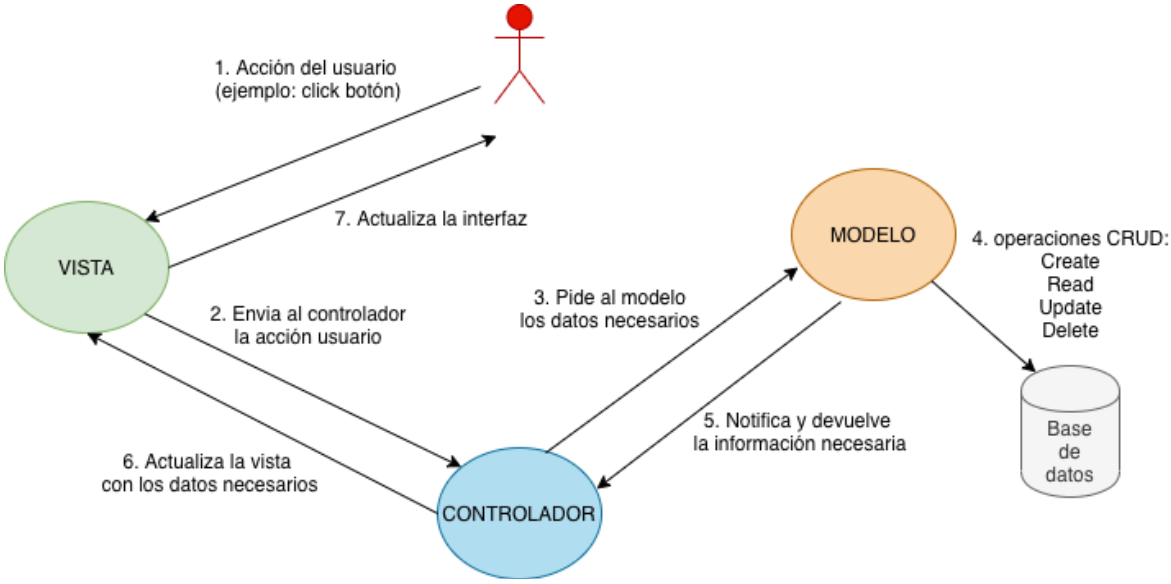

Figura 6: Diagrama del patrón MVC. Fuente: elaboración propia.

De manera general, este es el esquema de funcionamiento de un patrón MVC. El usuario realiza una acción (enviar un formulario, clicar un botón para pedir información, etc.), la vista detecta la acción y se la pasa al controlador, el controlador pide al modelo los datos necesarios (si la acción es por ejemplo enviar un formulario, probablemente necesite almacenar los datos enviados), el modelo realiza las operaciones correspondientes con la base de datos, notifica y devuelve información en caso de que sea necesario, el controlador transmite la actualización a la vista y la vista hace las actualizaciones necesarias en la interfaz.

4.3.2 Estructura de un proyecto Rails

Al crear un proyecto nuevo de Ruby on Rails con el siguiente comando, se genera la estructura de directorios que podemos ver en el cuadro 7:

```
$ rails new nombre_proyecto
```

Archivo/Carpeta	Objetivo
app/	Contiene los controladores, modelos, vistas, assets, *helpers, mailers* u otros. Tu código irá principalmente en esta carpeta.
bin/	Contiene los *scripts* Rails para iniciar tu aplicación, así como otros posibles ficheros de inicialización, actualización o despliegue.

Archivo/Carpeta	Objetivo
config/	Este archive contiene la configuración de las rutas, base de datos y otros aspectos. Para más información, visitar Configuring Rails Applications.
config.ru	Configuración de Rack para servidores basados en Rack. Para más información, visitar la Rack website.
db/	Contiene el esquema de la base de datos y las migraciones.
Gemfile Gemfile.lock	Estos dos archivos te permiten especificar las dependencias de gemas que necesita tu aplicación Rails. Son los fichero que utiliza el Bundler de Rails. Para más información, puedes visitar su web: Bundler website.
lib/	Extensión de módulos para tu aplicación.
log/	Ficheros log de la aplicación.
package.json	Este archive te permite especificar las dependencias npm (*node package manager*) de tu aplicación. Es el fichero utilizado por Yarn. Para más información, visita la Yarn website.
public/	Contiene archivos estáticos y assets compilados.
Rakefile	Este fichero localiza y carga tareas que pueden ser ejecutadas desde la línea de comandos. Si necesitaras personalización, no cambias este archive sino que añades el tuyo propio dentro de la carpeta lib/tasks.
README.md	Breve introducción/manual para tu aplicación. Debes editar este fichero para indicarle a otros de qué va tu aplicación, cómo configurar el proyecto, etc.
storage/	Archivos de Active Storage para el Disk Service. Para más información, visitor Active Storage Overview.
test/	Test unitarios y otros aspectos relacionados con el testing de la aplicación. Más información en Testing Rails Applications.
tmp/	Archivos temporales.
vendor/	Carpeta para código de terceros.
.gitignore	Este archivo le dice a Git que ficheros o patrones debe ignorar (de este modo, no se subirán al repositorio remoto).
.ruby-version	Este fichero contiene la version por defecto de Rails.

Cuadro 7: Estructura de un proyecto Ruby on Rails. Fuente: Web official Rails, Ruby on Rails Guide.

En el Cuadro 7 podemos observar diferentes ficheros y directorios dentro de la estructura, pero los que más nos interesan son estos tres:

1. La carpeta *models,* que contiene los modelos de la aplicación.
2. La carpeta *controllers,* que contiene los controladores de la aplicación.
3. La carpeta *views,* que contiene las vistas de la aplicación.

Las tres carpetas se encuentran dentro del directorio principal /app, y, a continuación, se explicarán estas tres partes a las que nos referimos.

4.3.3 Modelos

Un modelo de Rails (y en general en cualquier *framework* MVC o similar) es una clase que se encarga de la interacción con la base de datos (creación, actualización, búsquedas, etc).

Para generar un nuevo modelo en Rails, se ejecuta el siguiente comando en el directorio del proyecto:

```
$ rails generate model nombre_modelo atributo1:tipo_dato atribut
o2:tipo_dato …
```

Este comando nos generaría un archivo nombre_modelo.rb.

Es importante destacar que dentro de los modelos hay que establecer relaciones entre ellos (acorde al diseño de la base de datos). De este modo, dentro del modelo Usuario (user.rb) tendríamos lo siguiente:

```
class User < ApplicationRecord
  has_many :measurements
  has_many :routines
  has_many :workouts
  has_many :exercises, dependent: :destroy
  has_many :diets
  has_many :meals, dependent: :destroy
  has_many :foods, dependent: :destroy
  has_many :physio_exercises
  has_many :psychology_tasks
  has_many :appointments
  .
  .
  .
end
```

Como se puede observar, el usuario tiene muchas (*has_many)* rutinas, dietas, etc., tal y como se había establecido en el diseño de la base de datos. Esto nos permite hacer cosas como:

`User.diets`

Lo que nos devolvería las dietas pertenecientes a un usuario. Por el lado contrario, en el modelo Dieta tendríamos lo siguiente:

```
class Diet < ApplicationRecord
  belongs_to :user
  .
  .
  .
end
```

De este modo, un usuario tiene muchas dietas pero cada dieta pertenece a un único usuario (*belongs_to)*, tal y como se establece en las relaciones de la base de datos. Este tipo de relaciones se establecen también para el resto de modelos según sea necesario.

A continuación, se listan los diferentes modelos dentro de la aplicación Nutrain, así como algunos de sus métodos más importantes.

- **User model:** modelo Usuario.
    - **only_professionals():** devuelve una lista únicamente de los usuarios profesionales.

```
def self.only_professionals
    where(type: 'Professional')
end
```

    - **only_clients():** devuelve una lista únicamente de los usuarios clientes.

```
def self.only_clients
    where(type: 'Client')
end
```

    - **get_current_routine():** devuelve la dieta actual del usuario (la que tiene marcada como “active”).

```
def get_current_routine
    self.routines.where(active: true).first
end
```

- **get_current_diet():** devuelve la dieta actual del usuario.

```
def get_current_diet
  self.diets.where(active: true).first
end
```

- **Client model:** modelo Cliente (hereda del modelo User).
  - **current_professional_list():** devuelve la lista de usuarios profesionales con los que el usuario tiene contrato.

```
def current_professionals_list
    current_professionals = []
    self.contracts.each do |contract|
      current_professionals << User.find(contract.professional_id)
    end
    current_professionals
end
```

  - **professionals_without_contract():** devuelve la lista de usuarios profesionales con los que el cliente aún no tiene contrato.

```
def professionals_without_contract
    all = Professional.all
    current_professionals = current_professionals_list
    not_my_professionals = all - current_professionals
    not_my_professionals
end
```

  - **has_contract?(id):** revisa si el usuario cliente tiene un contrato con el profesional pasado por parámetro como id.

```
def has_contract?(user_id)
    result = false
    self.contracts.each do |contract|
      if contract.professional_id == user_id && contract.state ==
'Active'
        result = true
      end
    end
    result
end
```

- **Professional model:** modelo Profesional (hereda del modelo User).

- **current_clients_list():** devuelve la lista de clientes con los que el profesional tiene contrato. La lógica es la misma que para el método current_professional_list() del modelo Cliente, solamente cambiando los nombres de las variables.
- **clients_without_contract():** devuelve la lista de usuarios clientes con los que el profesional aún no tiene contrato. La lógica es la misma que para el método professionals_without_contract() del modelo Cliente, solamente cambiando los nombres de las variables.
- **has_contract?(id):** revisa si el usuario profesional tiene un contrato con el cliente pasado por parámetro como id. La lógica es la misma que para el método has_contract_id?() del modelo Cliente.
- **is_nutritionist?():** verifica si el profesional es un nutricionista.

```
def is_nutritionist?
    self.nutritionist ? 'Yes' : 'No'
end
```

- **is_trainer?():** verifica si el profesional es un entrenador. La lógica es la misma que en is_nutritionist?() pero cambiando el nombre de la variable.
- **is_physiotherapist?():** verifica si el profesional es un fisioterapeuta. La lógica es la misma que en is_nutritionist?() pero cambiando el nombre de la variable.
- **is_psychologist?():** verifica si el profesional es un psicólogo. La lógica es la misma que en is_nutritionist?() pero cambiando el nombre de la variable.

- **Routine model:** modelo Rutina de entrenamiento.
- **Workout model:** modelo Entrenamiento.
- **Exercise model:** modelo Ejercicio.
- **Diet model:** modelo Dieta.
- **Meal model:** modelo Comida.
  - **calculate_total_macros():** calcula los macronutrientes totales (kcal, proteínas, carbohidratos y grasas) de la comida.

```
def calculate_total_macros_food
    self.kcal = self.protein = self.carbs = self.fats = 0.0
    self.vegan = self.vegetarian = self.nut_free = self.gluten_free =
self.soy_free = false
```

```
    if foods.empty?
      return
    else
      foods.each do |food|
        serving_size = (food.serving_size / 100.0)
        self.kcal += (serving_size * food.kcal).round
        self.protein += (serving_size * food.protein).round
        self.carbs += (serving_size * food.carbs).round
        self.fats += (serving_size * food.fats).round
        self.vegan = food.vegan
        self.vegetarian = food.vegetarian
        self.nut_free = food.nut_free
        self.gluten_free = food.gluten_free
        self.soy_free = food.soy_free
      end
    end
  end
```

- **total_kcal_macros_today():** calcula los macronutrientes totales de un usuario durante el día en curso.

```
def self.total_kcal_macros_today(user_id)
    total_macros = Hash.new
    total_macros["kcal"] = total_macros["protein"] =
total_macros["carbs"] = total_macros["fats"] = 0
    User.find(user_id).meals.where('DATE(day) = ?', Date.today).each do
|meal|
      total_macros["kcal"] += meal.kcal
      total_macros["protein"] += meal.protein
      total_macros["carbs"] += meal.carbs
      total_macros["fats"] += meal.fats
    end
    total_macros
end
```

- **total_kcal_today():** calcula las kilocalorías totales consumidas por un usuario durante el día en curso.

```
def self.total_kcal_meals_today(user_id)
    total_kcal = 0
    User.find(user_id).meals.where('DATE(day) = ?', Date.today).each do
|meal|
      total_kcal += meal.kcal
    end
    total_kcal
end
```

Para las proteínas, carbohidratos y grasas se calcularía de la misma manera, simplemente cambiando la variable a la que se accede.

- **total_proteins_today():** calcula las proteínas totales consumidas por un usuario durante el día en curso.

```
def self.total_protein_meals_today(user_id)
    total_protein = 0
    User.find(user_id).meals.where('DATE(day) = ?', Date.today).each do
|meal|
      total_protein += meal.protein
    end
    total_protein.round
end
```

- **total_carbs_today():** calcula los carbohidratos totales consumidos por un usuario durante el día en curso. La lógica es la misma que en el ejemplo total_proteins_today(), solamente cambiando el nombre de las variables.
- **total_fats_today():** calcula las grasas totales consumidas por un usuario durante el día en curso. La lógica es la misma que en el ejemplo total_proteins_today(), solamente cambiando el nombre de las variables.

- **Food model:** modelo Alimento/Ingrediente.
- **Physiotherapy exercise model:** modelo Ejercicio de fisioterapia.
- **Psychology exercise model:** modelo Ejercicio de psicología.
- **Measurement model:** modelo Medida corporal.
- **Contract model:** modelo Contrato.
- **Appointment model:** modelo Cita.

4.3.4 Vistas

Son la representación visual de la información (páginas, formularios, menús, etc.).

Dentro del proyecto, cada modelo tiene su propia carpeta dentro del directorio /views. Es decir, para el modelo *Diets*, hay una carpeta views/diets, y así con los demás. Salvo algunas excepciones, todos los modelos tienen al menos las siguientes vistas:

- **new.html.erb:** vista para crear una instancia de un modelo.
- **index.html.erb:** vista para mostrar la lista de instancias de un modelo.
- **show.html.erb:** vista para mostrar un elemento.
- **edit.html.erb:** vista para editar un elemento.

- **form.html.erb:** vista que contiene el formulario para crear/editar una instancia.

La extensión del fichero (.html.erb) indica que el archivo contiene tanto código HTML como código Ruby (erb = embedded ruby). Existen otras variaciones para utilizar diferentes lenguajes, como por ejemplo haml.erb o js.erb.

Además de los anteriores, dentro del directorio /views también se encuentran otros elementos de la vista, como por ejemplo:

- **application.html.erb:** fichero que contiene la estructura general de la página (head, links a los archivos CSS y JS, footer…)
- **_navigation.html.erb:** archivo que contiene el código de la barra superior de navegación.
- **_profile.html.erb:** fichero que contiene los elementos del menú "Perfil".
- **pdf.html.erb:** fichero que contiene la estructura general de los informes PDF que pueden generar y descargar los usuarios. El contenido específico de cada PDF se encuentra en el fichero "index.pdf.erb" dentro de la respectiva carpeta de cada modelo.

A continuación, se muestra el código del fichero application.html.erb:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Nutrain</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>
    <%= stylesheet_link_tag 'application', media: 'all', 'data-
turbolinks-track': 'reload' %>
    <%= javascript_pack_tag 'application', 'data-turbolinks-track':
'reload' %>
    <%= wicked_pdf_stylesheet_link_tag "pdf" %>
    <%= wicked_pdf_javascript_include_tag "number_pages" %>
    <link
href="https://fonts.googleapis.com/css?family=Catamaran|Comfortaa|Roboto|
Muli|Poppins|Sriracha|Quicksand&display=swap" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.13.0/css/all.min.css"
    integrity="sha256-h20CPZ0QyXlBuAw7A+KluUYx/3pK+c7lYEpqLTlxjYQ="
crossorigin="anonymous" />
  </head>
  <body>
  <div class="content">
```

```
  <header>
    <%= render 'layouts/messages' %>
    <%= render 'layouts/navigation' %>
  </header>
    <div class="container-fluid">
      <%= yield %>
    </div>
    </div>
  <div class="footer-box">
    <footer class="footer text-center">&copy; 2020 Daniel Suárez Cáceres.
All rights reserved.</footer>
  </div>
  </body>

</html>
```

Como se puede comprobar, contiene la estructura general para toda la aplicación: link a las hojas de estilo CSS, a los ficheros JavaScript, a la librería wicked_pdf (permite generar los archivos PDF), las fuentes, etc., y también elementos comunes a todas las páginas, como las notificaciones (messages) o la barra de navegación.

4.3.5 Controladores

El Controlador es el intermediario entre el Modelo y la Vista. Se encarga de recibir las acciones del usuario en la página web, interaccionar con el Modelo si fuera necesario (por ejemplo, recuperar información de la base de datos) y comunicarle a la Vista dicha información.

Rails dispone de un comando para generar controladores y es el siguiente:

```
$ rails generate controller NombreModeloEnPlural
```

Por ejemplo: `rails g controller Users`

Todos los controladores disponen de los siguientes siete métodos (además de los que decida añadir el usuario, pero estos siete son los principales y los que genera por defecto Rails):

- **index:** obtener la lista de instancias de un modelo.
- **show:** mostrar una única instancia de un modelo.
- **new:** mostrar el formulario o la vista correspondiente para crear una nueva instancia de un modelo.

- **create:** realiza la acción de crear una instancia (new es para renderizar, create es para crear)
- **edit:** mostrar el formulario o la vista correspondiente para editar una instancia de un modelo.
- **update:** realiza la acción de actualizar una instancia (edit es para renderizar, update para actualizar)
- **destroy:** destruye la instancia de un modelo.

Son las siete rutas RESTful que Rails provee por defecto (el usuario puede añadir más por su cuenta pero estas son las básicas), y que coinciden además con los nombres de las vistas como vimos antes (index, new, etc) y también con los métodos HTTP (GET, POST, PUT, PATCH, DELETE) y con las operaciones CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) sobre la base de datos.

Dentro de la aplicación, como se observa en el Cuadro 8, se distinguen los siguientes controladores:

Modelo	**Controlador**
Usuario	*UsersController*
Rutina	*RoutinesController*
Dieta	*DietsController*
Entrenamiento	*WorkoutsController*
Comida	*MealsController*
Ejercicio	*ExercisesController*
Alimento	*FoodsController*
Ejercicio psicología	*PsychologyExercisesController*
Ejercicio fisioterapia	*PhysioExercisesController*
Cita	*AppointmentsController*
Contrato	*ContractsController*
Medida corporal	*MeasurementsController*
Correo contacto	*ContactsController*

Cuadro 8: Tabla de Controladores de la aplicación.

Para entender mejor el funcionamiento de un Controlador, se muestra a continuación el Controlador del modelo Dieta *(DietsController.rb):*

```
class DietsController < ApplicationController

  def index
    if params[:client_id]
      @pagy, @diets = pagy(User.find(params[:client_id]).diets)
    else
      @pagy, @diets = pagy(User.find(params[:user_id]).diets)
      respond_to do |format|
        format.html
        format.pdf do
          render pdf: "Diets_#{Date.today.strftime("%Y_%m_%d")}",
          page_size: 'A4',
          template: "diets/index.pdf.erb",
          title: "Diets_#{Date.today.strftime("%Y_%m_%d")}",
          locals: { :diets => @diets }
        end
      end
    end
  end

  def show
    @diet = Diet.find(params[:id])
  end

  def new
    @diet = Diet.new
  end

  def edit
    @diet = Diet.find(params[:id])
  end

  def create
    if params[:diet][:user_id].to_i != current_user.id
      @client = User.find(params[:diet][:user_id].to_i)
      @diet = @client.diets.new(diet_params)
      if @diet.valid?
        @diet.save
        redirect_to user_clients_path(user_id: params[:user_id]), notice:
'Diet created for your client successfully'
      else
        render :new
      end
    else
      @user = current_user
      @diet = @user.diets.new(diet_params)
      if @diet.valid?
        @diet.save
        redirect_to user_diets_path(@user), notice: 'Diet created
successfully'
      else
        render :new
```

```
      end
    end
  end

  def update
    @diet = Diet.find(params[:id])
    if @diet.update(diet_params)
      if params[:diet][:user_id].to_i != current_user.id
        redirect_to user_clients_path(current_user), notice: 'Diet
updated successfully'
      else
        redirect_to user_diets_path(current_user), notice: 'Diet updated
successfully'
      end
    else
      render :edit
    end
  end

  def destroy
    @diet = Diet.find(params[:id])
    if @diet.destroy
      redirect_to user_diets_path(current_user), notice: 'Diet deleted
successfully'
    else
      flash[:error] = @diet.errors.full_messages
      redirect_to user_diet_path(@diet)
    end
  end

  def delete_all
    if current_user.diets.delete_all
      redirect_to user_diets_path(current_user), notice: 'All diets
deleted successfully'
    else
      redirect_to user_dietss_path(current_user), flash[:error] =
'Something went wrong while deleting diets'
    end
  end

  private

  def diet_params
    params.require(:diet).permit(
      :id,
      :name,
      :description,
      :user_id,
      :client_id,
      :kcal_goal,
      :recommended_meals,
      :active,
      :goal
    )
  end
end
```

Lo primero que se puede observar son los métodos que coinciden con los nombres de las vistas, como indicábamos antes (*index, new, create, edit, update, delete).* Por ejemplo, el método *show()* lo que hace es buscar el id de una dieta. El método *create()* hace unas comprobaciones antes de guardar la instancia, y si es válida, pide al modelo que haga una operación (*diet.save),* muestra un mensaje (*notice)* y redirige a la correspondiente página *(redirect_to),* donde la vista mostrará los cambios necesarios. Tal y como se había explicado en la sección 4.3.1 sobre el funcionamiento de una aplicación Modelo-Vista-Controlador.

También cabe destacar que todos los controladores deben tener un método *nombreModelo_params,* donde se definen los parámetros que acepta ese modelo (lo normal es que al menos tengan los mismos que los campos en la base de datos, pero se pueden añadir más según sea necesario). Esta lista de métodos que se conoce como *whitelist* sirve para que solamente esos parámetros que llegan al controlador a través de una petición HTTP puedan guardarse en la base de datos. En el caso de que le llegasen otros parámetros, si no se encuentra dentro de la *whitelist,* no se guardará.

4.3.6 Archivos de configuración u otros

Dentro de la carpeta "*config*", hay varios archivos importantes. Uno que habría que destacar es el routes.rb, el cual contiene las rutas de la aplicación.

```
Rails.application.routes.draw do
  get 'welcome/index'
  root 'welcome#index'

  devise_for :users, controllers: {
      registrations: 'users/registrations'
  }

  resources :professionals, controller: 'users', type: 'Professional',
only: %i[index show]
  resources :clients, controller: 'users', type: 'Client', only: %i[index
show]

  resources :users do
    resources :clients, only: %i[index show], controller: 'users', type:
'Client' do
      resources :routines
      resources :diets
      resources :psychology_tasks
      resources :physio_exercises
      resources :appointments
```

```
    resources :measurements
  end
  resources :measurements do
    collection do
      get 'delete_all'
    end
  end
  resources :routines do
    collection do
      get 'delete_all'
    end
  end
  resources :workouts do
    collection do
      get 'delete_all'
    end
  end
  resources :exercises do
    collection do
      get 'delete_all'
    end
  end
  resources :diets do
    collection do
      get 'delete_all'
    end
  end
  resources :meals do
    collection do
      get 'delete_all'
    end
  end
  resources :foods do
    collection do
      get 'delete_all'
    end
  end
  resources :psychology_tasks do
    collection do
      get 'delete_all'
    end
  end
  resources :physio_exercises do
    collection do
      get 'delete_all'
    end
  end
  resources :appointments do
    collection do
      get 'delete_all'
    end
  end
  resources :contracts do
    collection do
      get 'delete_all'
    end
```

```
    end
  end

  resources :contacts, only: %i[new create]

end
```

Como se puede observar, la palabra clave *resources* sirve para indicar un recurso, que se corresponde en este caso con cada uno de los modelos de la aplicación (usuario, dieta, rutina, etc). Esta *keyword* genera todas las rutas HTTP para ese recurso: GET, POST, PUT, PATCH y DELETE. También es importante destacar que se pueden anidar recursos, de modo que en el navegador aparecía en la URL de la siguiente manera:

```
http:localhost:3000/recurso_principal/id/recurso_anidado
```

En el proyecto, esto se utiliza para los usuarios. Como ya se ha explicado en secciones anteriores, del modelo Usuario derivan los modelos Cliente y Profesional.

En este caso, se utiliza para anidar al recurso Usuario el recurso Cliente, y a su vez anidar al recurso Cliente los demás recursos (dietas, rutinas, etc). Con esto se consigue que un usuario profesional pueda editar recursos de su cliente, como por ejemplo editar su dieta (si es nutricionista en este caso).

Capítulo 5: Conclusiones finales, competencias específicas y líneas futuras

5.1 Conclusiones finales

Una vez finalizado el proyecto "Nutrain: aplicación web para la nutrición, entrenamiento y otros aspectos de la salud", podemos llegar a la conclusión de que se han cumplido los objetivos que se tenían en mente desde el inicio.

La aplicación permite almacenar, gestionar y mantener un seguimiento de los cuatro aspectos de la salud (nutrición, entrenamiento, psicología y fisioterapia) que se plantearon en un principio, aunque sí es cierto que los bloques de nutrición y entrenamiento están un tanto más desarrollados que los de fisioterapia y psicología. En ese aspecto, haría falta realizar más investigación para que estos dos bloques (fisioterapia y psicología) estuvieran más completos.

En segundo lugar, respecto al desarrollo del proyecto, se concluye que ha ayudado a afianzar conceptos aprendidos durante la carrera, y, sobre todo, a tener más confianza al haber profundizado en todos los aspectos relativos al desarrollo de un proyecto software: idea inicial, diseño de la interfaz, diseño de la base de datos, lógica. Se considera que este tipo de proyectos te hacen crecer, tanto personal como profesionalmente, porque te enseñan disciplina, trabajo duro y constancia.

Sobre las tecnologías utilizadas, podemos decir que Ruby on Rails es un *framework* muy maduro y cómodo. Sus conceptos como *DRY* (*Don't Repeat Yourself*) o *Convention over Configuration* hacen que el flujo de trabajo sea bastante cómodo. Es ideal para proyectos que requieran un tiempo de desarrollo especialmente rápido. También cabe destacar la librería Bootstrap, la cual podría considerarse bastante útil para todo aquel que no disponga de conocimientos suficientes de HTML-CSS como para hacer su propia interfaz desde cero. Bootstrap permite generar diseños de manera rápida y cómoda, atentiendo sobre todo a aspectos como usabilidad y diseño adaptativo (*Responsive Design).*

En resumen, se puede concluir que hay satisfacción con el trabajo realizado, no sólo por el hecho de haber completado el proyecto en sí, sino porque además abarca áreas de gran interés, como son la nutrición y el entrenamiento, y en general, la salud.

5.2 Competencias específicas cubiertas

Con la realización de este proyecto se han cubierto una serie de competencias de la especialidad de Ingeniería del Software. Se listan a continuación:

- CII016: Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Las distintas fases del desarrollo de este Trabajo de Fin de Grado (análisis, planificación, diseño, implementación y documentación), basándose en buenas prácticas de desarrollo software y de programación, hacen que esta competencia quede cubierta.

- IS01: Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

En general, esta competencia queda cubierta con la propia realización del proyecto, puesto que se ha planeado, analizado y desarrollado una aplicación software, teniendo en cuenta los requisitos y necesidades del usuario, asegurando un producto de calidad y aplicando diferentes métodos de ingeniería del software.

- IS03: Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

Con la elección e integración de los diferentes lenguajes, programas y librerías/*frameworks* queda cubierta esta competencia.

- IS04: Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

A medida que se desarrollaba el proyecto, han ido surgiendo diversos problemas, por ejemplo, al implementar una nueva funcionalidad o al hacer cambios en la lógica de la aplicación. Para ello, se ha intentado buscar una solución profesional basándose en buenas prácticas de software ya afianzadas, patrones de diseño, y, sobre todo, siguiendo las buenas prácticas de desarrollo de cada lenguaje/librería/*framework*.

5.3 Líneas futuras

Desde un principio estaba claramente definido hasta dónde iba a abarcar la aplicación, pero también se tenía en mente que esto era la base de lo que podría ser algo mucho más grande en un futuro. De hecho, durante el desarrollo del proyecto han ido surgiendo aún más ideas para un futuro. A continuación, se explican algunas de las ideas y conceptos que deberían implementarse en un futuro:

- **Implementación de una API de alimentos:** registrarse y hacer uso de una API de pago, que permita acceder a una base de datos profesional de alimentos, puesto que no sólo mejoraría la funcionalidad de la aplicación, sino que además permitiría un análisis mucho más detallado en el aspecto de nutrición. Si se diera el caso, también podría considerarse una API para alguno de los otros tres bloques de la salud.

- **Implementación de un sistema de mensajería:** sería interesante implementar un chat o similar para que los usuarios pudieran comunicarse a través de la aplicación, independientemente de que luego lo utilicen más o menos, pero al menos ofrecer esa opción.

- **Mejorar sistema de verificación de títulos:** es importante para el usuario que se pueda verificar que efectivamente, un entrenador posee algún título oficial de entrenador, o un nutricionista de nutricionista, etc. Actualmente el usuario

profesional tendría que enviar un correo a la página, indicando el título de colegiado o algún código identificativo, y los administradores de la página se encargarían de verificar dicha información con las correspondientes entidades oficiales. Sería interesante investigar y desarrollar algún sistema o lógica que permitiera realizar esto de una manera más automática.

- **Considerar el uso de un *framework* para el *front-end*:** aunque haciendo uso de HTML, CSS y JavaScript se pueden conseguir buenos resultados, se considera que sería beneficioso utilizar alguno de los *frameworks* para *front-end* más conocidos actualmente, como por ejemplo React o Vue, ambos *frameworks* de JavaScript. Dichos *frameworks* mejorarían en gran medida la usabilidad y experiencia general de la aplicación, dado que son tecnologías conocidas y valoradas muy positivamente en la actualidad.

- **Aplicación móvil:** aunque la aplicación web se ha realizado teniendo en cuenta que pudiera ser utilizada desde pantallas más pequeñas, adaptando el diseño e intentando hacerlo lo más *responsive* posible, sería interesante desarrollar una aplicación móvil para los sistemas operativos iOS y Android. En este caso, podría considerarse o bien desarrollar las aplicaciones por separado en sus respectivas plataformas (es decir, haciendo uso de Swift para iOS o de Java/Kotlin para Android), o bien utilizar *frameworks* donde el desarrollo es común para ambas plataformas, como por ejemplo Ionic, React Native, o el más reciente Dart.

Apéndice A: Manual de usuario

A continuación, se mostrará la aplicación web que resulta de este Trabajo de Fin de Grado. Se explicarán las distintas funcionalidades (de manera general las comunes a todos los usuarios, y puntualizando donde las acciones difieran según el tipo de usuario).

Para el desarrollo de este manual, se ha rellenado previamente la base de datos con algunos usuarios, rutinas, dietas, etc.

A.1 Login, registro, perfil

Lo primero que nos encontramos al acceder a la pagina es la página para iniciar sesión (si ya se ha registrado previamente), como se puede ver en la Figura A1, para registrarse, como se puede ver en la Figura A2, o para que te reenvíen tu contraseña en caso de que se te haya olvidado, como se puede ver en la Figura A3.

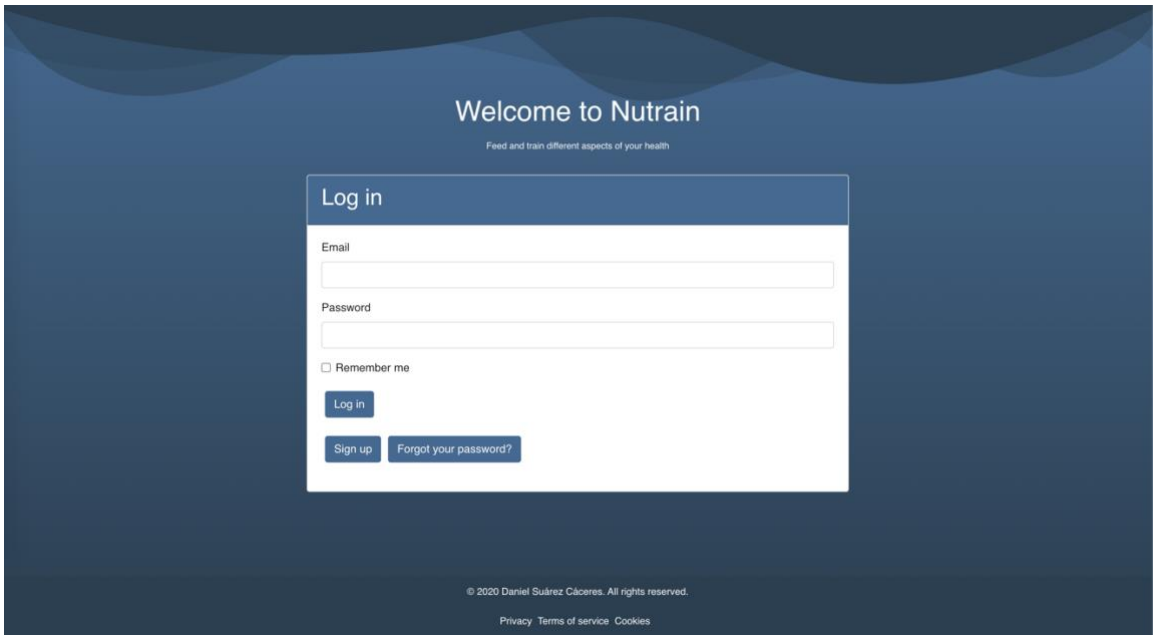

Figura A 1: Página inicio de sesión. Fuente: elaboración propia.

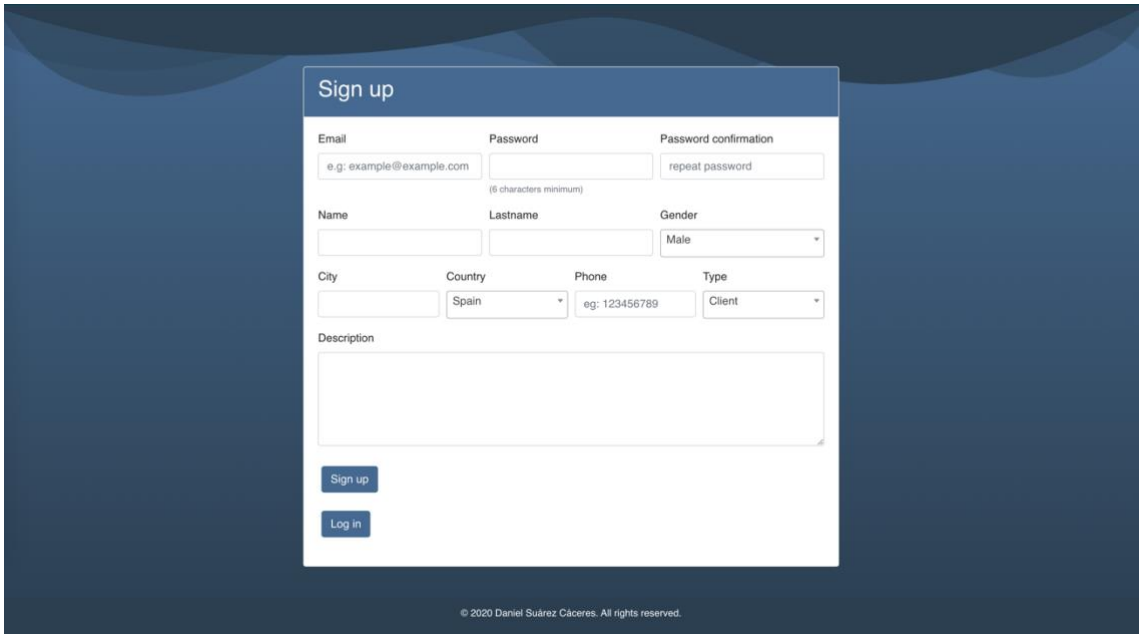

Figura A 2: Pantalla para registrarse. Fuente: elaboración propia.

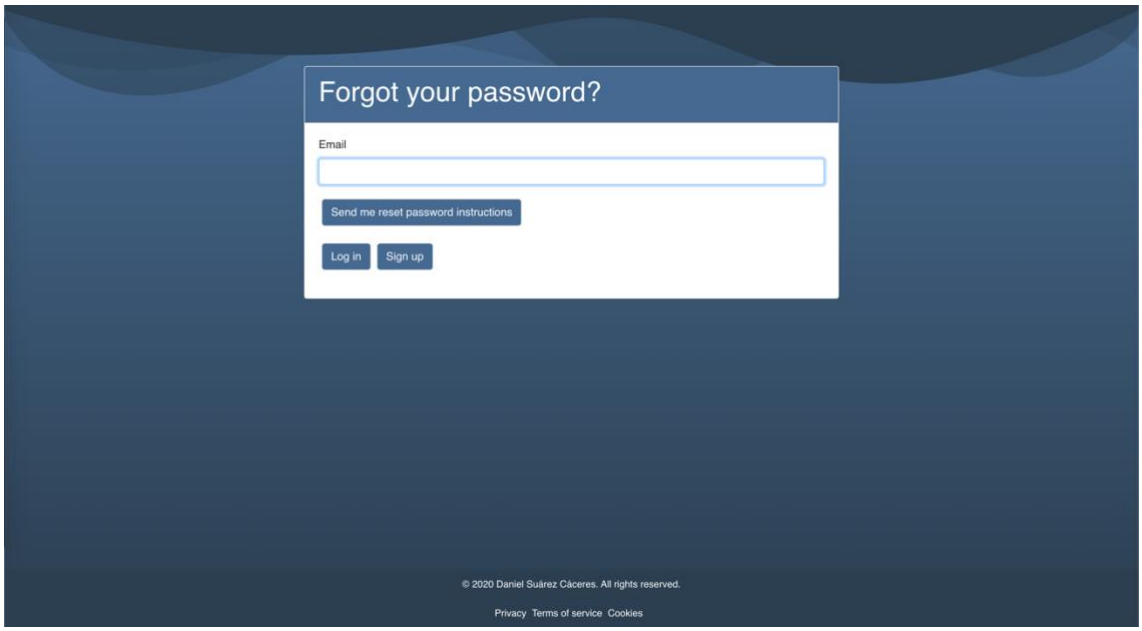

Figura A 3: Pantalla para reenviar contraseña. Fuente: elaboración propia.

Destacar que, una vez ha iniciado sesión, el usuario podrá ver la información de su perfil, así como editar los datos que crea convenientes.

A.2 *Dashboard*

Nada más iniciar sesión, el usuario verá una pantalla inicial a modo de *dashboard,* como se observa en la figura A4, donde puede ver datos y gráficos principalmente sobre su nutrición y también sobre su entrenamiento.

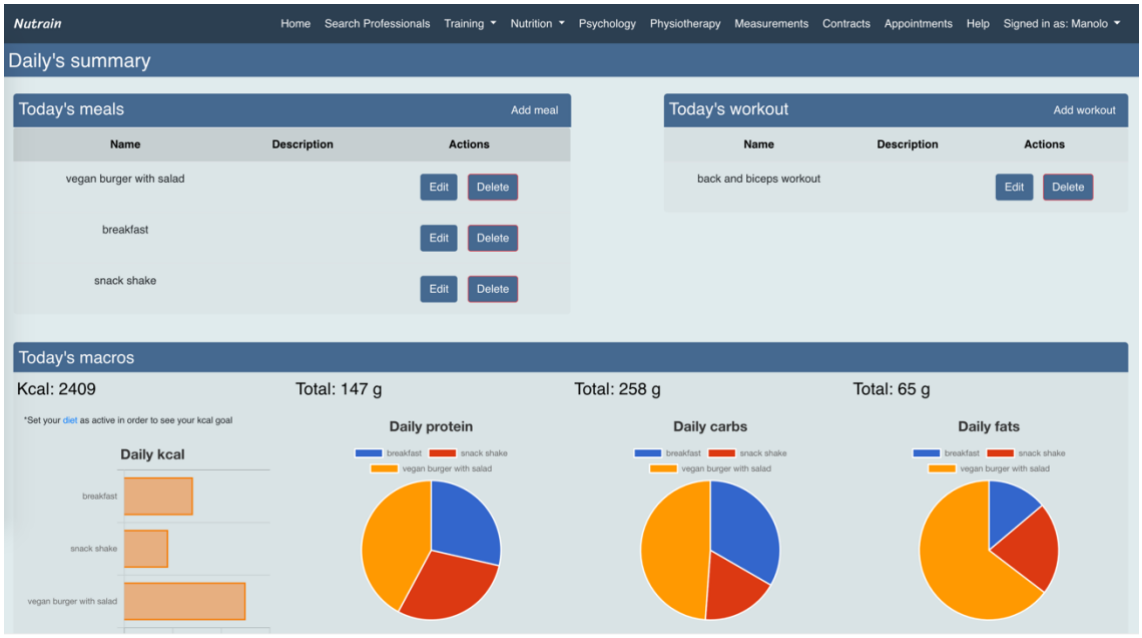

Figura A 4: Dashboard en página principal. Fuente: elaboración propia.

En la parte superior, tenemos dos tablas donde el usuario podrá añadir las comidas y entrenamientos respectivos al día en curso.

En la parte media, tenemos un resumen de los macronutrientes del día en curso, donde se informan de las kilocalorías totales ingeridas (respecto a las que tenemos como objetivo según nuestra dieta), las proteínas, los carbohidratos y las grasas totales. Se puede observar la distribución de todos estos parámetros a través de los respectivos gráficos de barras y de sectores.

Como se observa en la figura A5, en la parte inferior tenemos el menú *Nutrition history,* donde el usuario podrá observar un histórico de los macronutrientes y las calorías durante la última semana, el último mes o los últimos tres meses.

Figura A 5: Dashboard, resumen nutrición. Fuente: elaboración propia.

Todos los gráficos (al igual que los del resto de la página que se irán viendo) pueden ser descargados pulsando en la flecha que aparece en la parte superior derecha de cada gráfico.

Como se podía ver en la figura A4, podemos añadir comidas y entrenamientos desde esta pantalla inicial. Si pulsamos en *Add workout,* se mostrará el siguiente formulario donde el usuario puede rellenar los datos del entrenamiento (nombre, descripción, etc.) y añadir los ejercicios correspondientes (aparecerá un formulario para introducir los datos del ejercicio, como por ejemplo nombre, series, repeticiones, etc.), como se puede observar en las figuras A6 y A7.

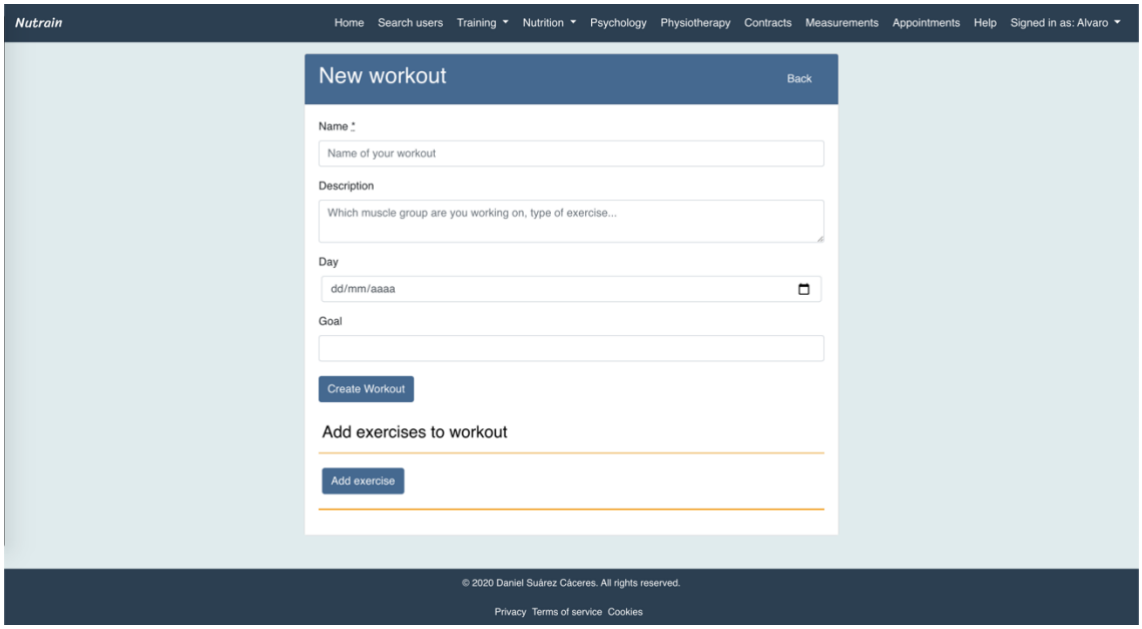

Figura A 6: Creación de una sesión de entrenamiento. Fuente: elaboración propia.

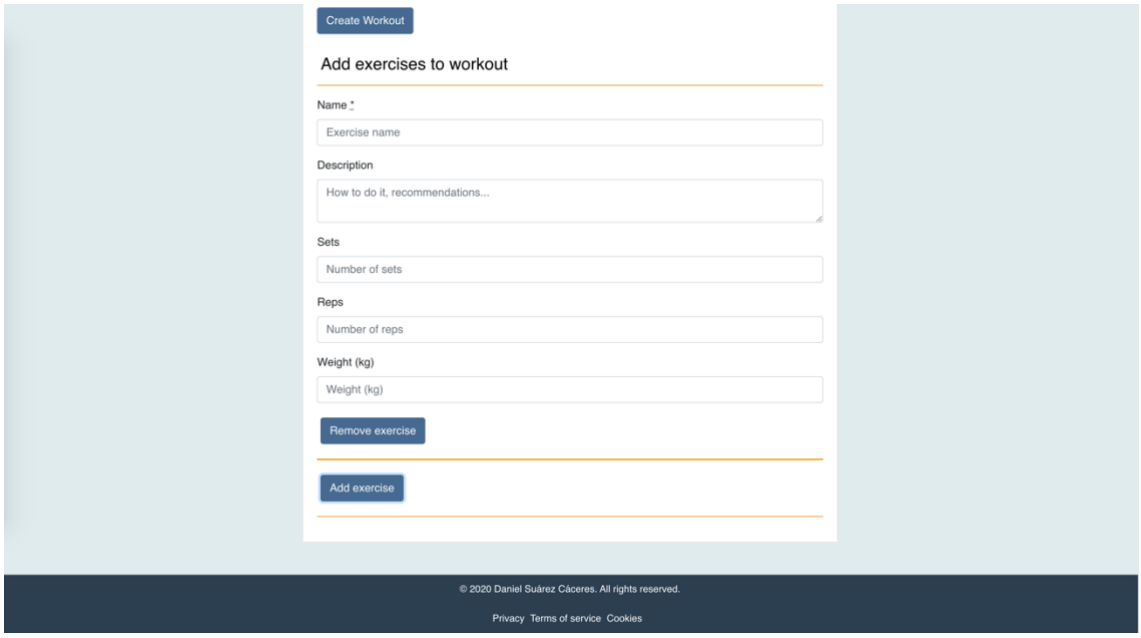

Figura A 7: Añadir ejercicios al entrenamiento. Fuente: elaboración propia.

Por otra parte, si en la pantalla principal pulsamos en *Add meal,* se mostrará el formulario para añadir una nueva comida y sus respectivos ingredientes, como se puede observar en la figura A8.

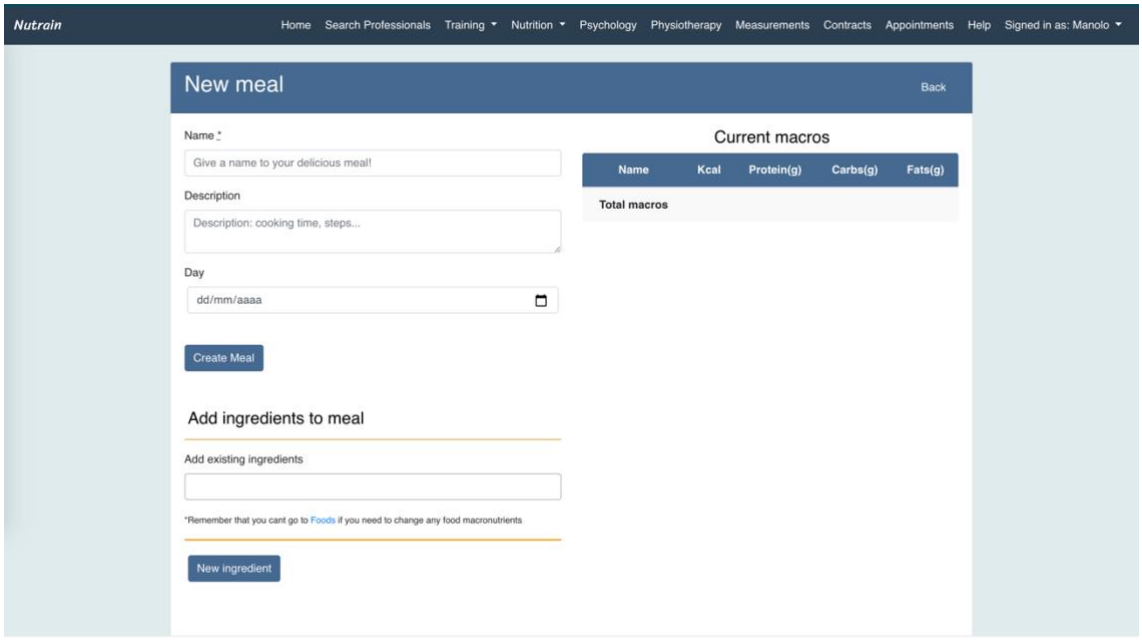

Figura A 8: Creación de una comida. Fuente: elaboración propia.

Podemos introducir nombre, descripción, añadir alimentos con el buscador. Si creamos una comida, la siguiente vez que entremos en el formulario veremos algo similar a la figura A9.

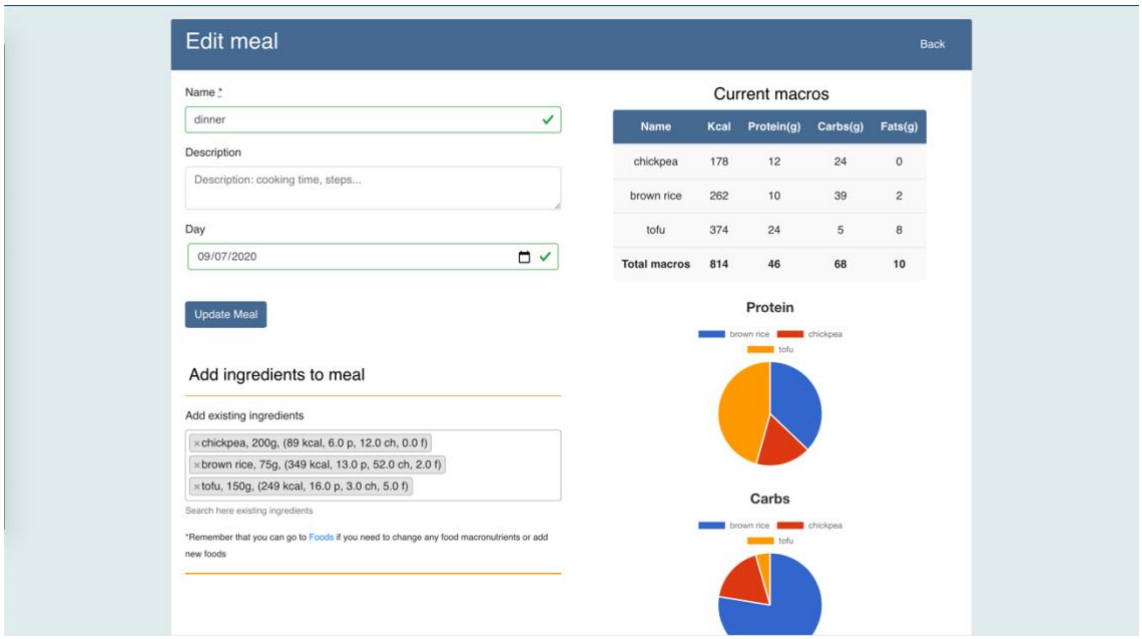

Figura A 9: Visualizar una comida ya creada. Fuente: elaboración propia.

Podemos observar los ingredientes añadidos, los macronutrientes de cada alimento, etc.

A.3 Almacenamiento y *tracking* de tu salud

En este apartado se mostrará cómo el usuario puede gestionar los distintos aspectos de su salud a través de la creación de dietas, gestión de informes, visualización de información y gráficos, etc.

A.3.1 Gestión entrenamiento

En la barra de navegación hay un menú desplegable *Training,* que tiene los submenús *Routines* y *Workouts.* Si pinchamos en *Workouts,* se nos llevará a la pantalla que hemos visto previamente para la creación de un workout. Si vamos al menú *Routines,* nos mostrará la lista de rutinas de entrenamiento que hayamos creado, tal y como se muestra en la figura A10.

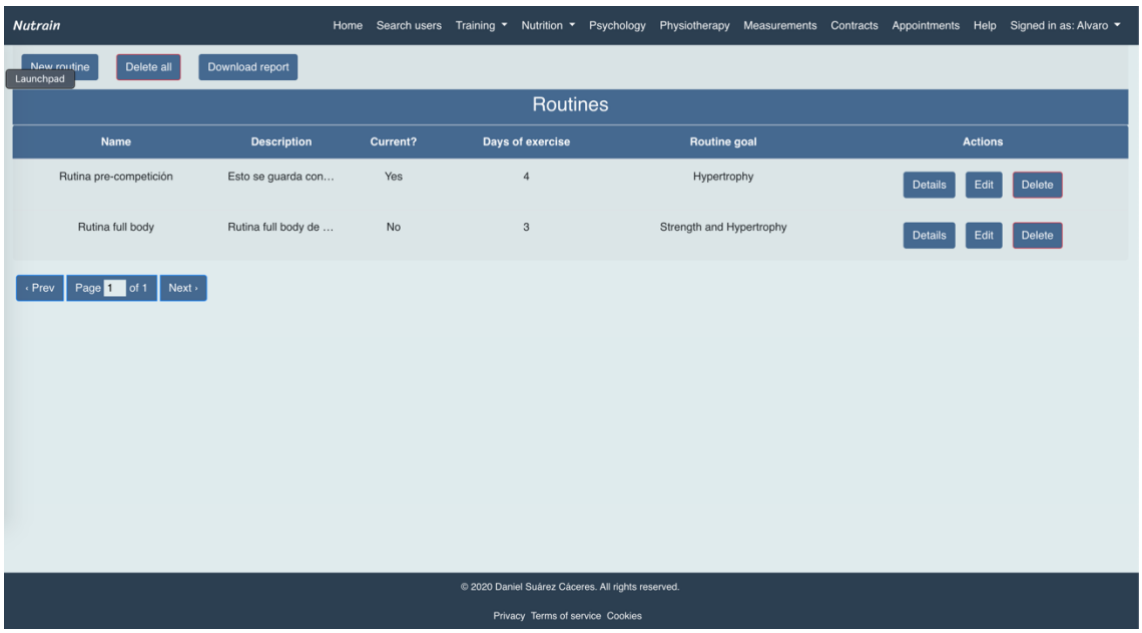

Figura A 10: Lista de rutinas de entrenamiento del usuario. Fuente: elaboración propia

Desde esta pantalla podemos ver información básica sobre la rutina (nombre, descripción, objetivo…), y a la derecha disponemos de una serie de opciones para entrar en los detalles *(Details),* editar la rutina de entrenamiento *(Edit)* o eliminarla *(Delete).*

En la parte superior tenemos las opciones de crear una nueva rutina, borrarlas todas (se mostrará un mensaje para que confirmemos si queremos borrarlas todas) y un botón para descargar un informe.

Si le damos a crear una nueva rutina, nos aparecerá el correspondiente formulario, como se observa en la figura A11:

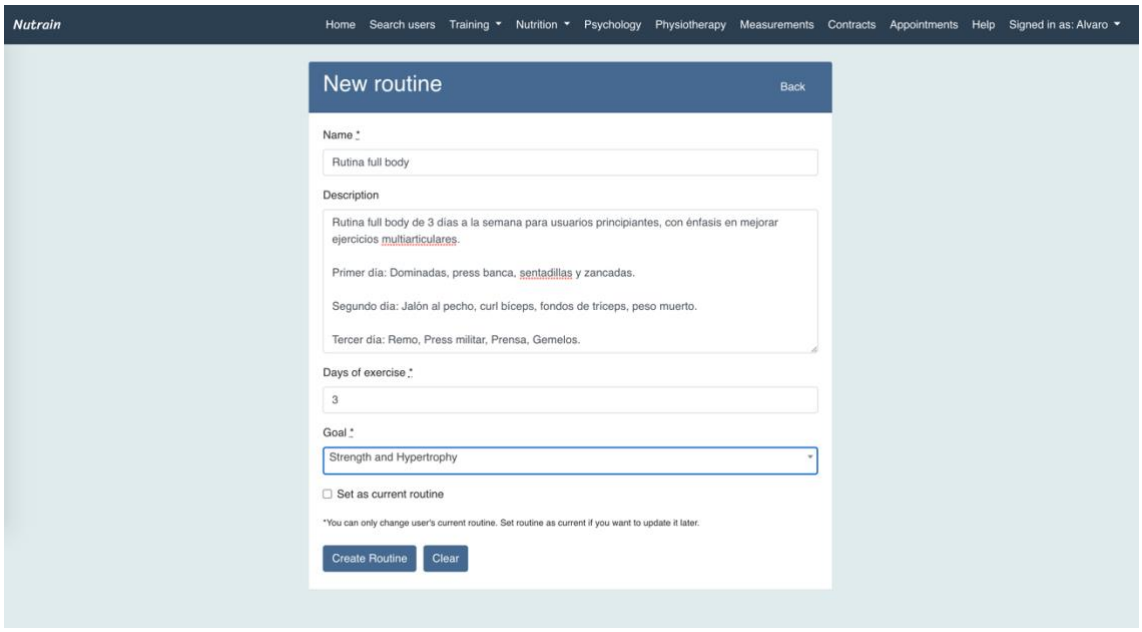

Figura A 11: Formulario para crear una nueva rutina. Fuente: elaboración propia.

Si le damos a descargar informe, se nos abrirá una nueva pestaña donde podremos ver un informe PDF con todas nuestras rutinas, como en la figura A12.

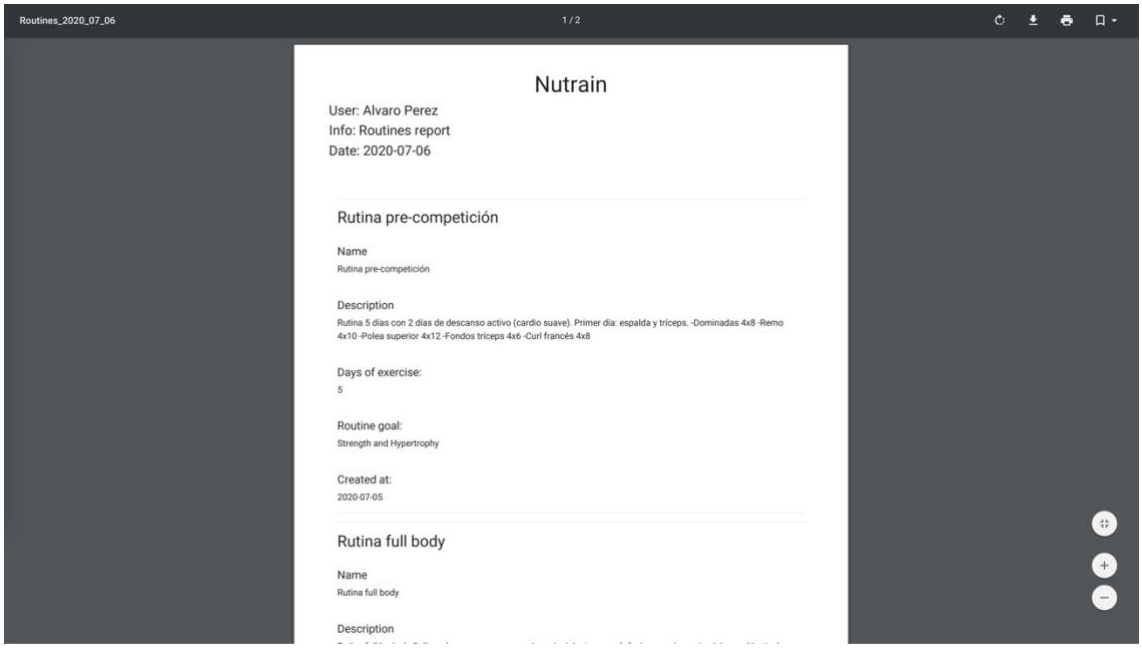

Figura A 12: Informe PDF de las rutinas del usuario. Fuente: elaboración propia.

Si accedemos al submenú de los *workouts* (sesiones de entrenamientos), tendremos una vista similar a la de las rutinas. Veremos un listado con todos los entrenamientos realizados de los cuales podemos ver detalles, editar o borrar. En la parte superior disponemos de botones para crear uno nuevo, borrar todos o descargar un informe.

Al pulsar el botón para descargar el informe, no se abre directamente, como en el caso de las rutinas, sino que se nos pide un intervalo de fechas, como se observa en la figura A13, de modo que el informe solo contendrá los entrenamientos realizados entre esas fechas, como se observa en la figura A14. Si no ponemos nada, se mostrarán todas.

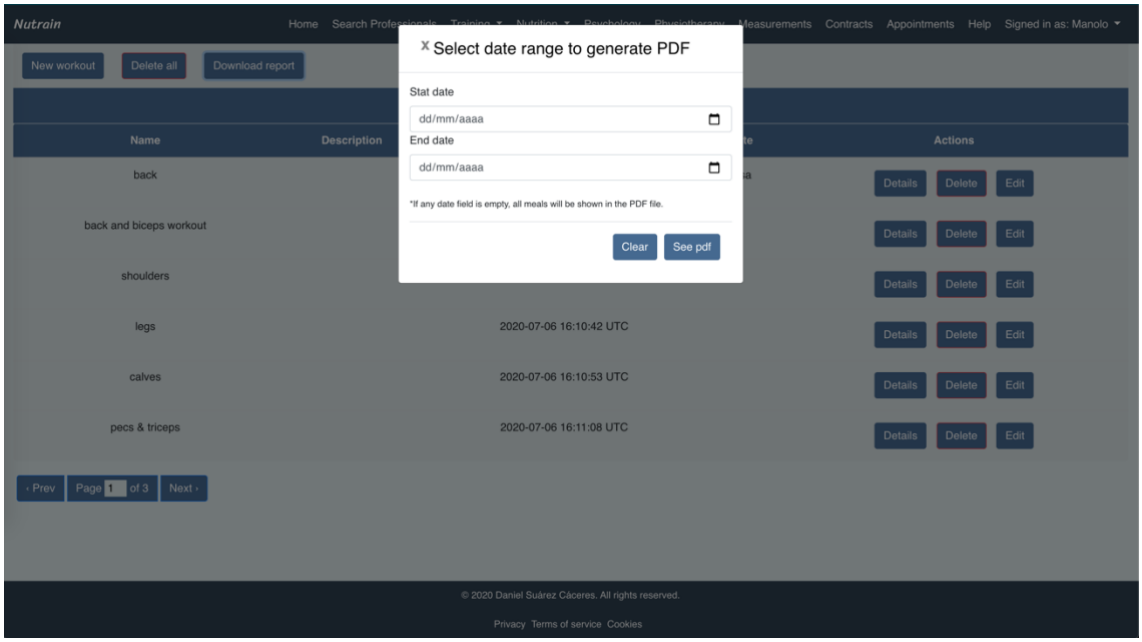

Figura A 13: Descargar informe PDF con intervalo de fechas. Fuente: elaboración propia.

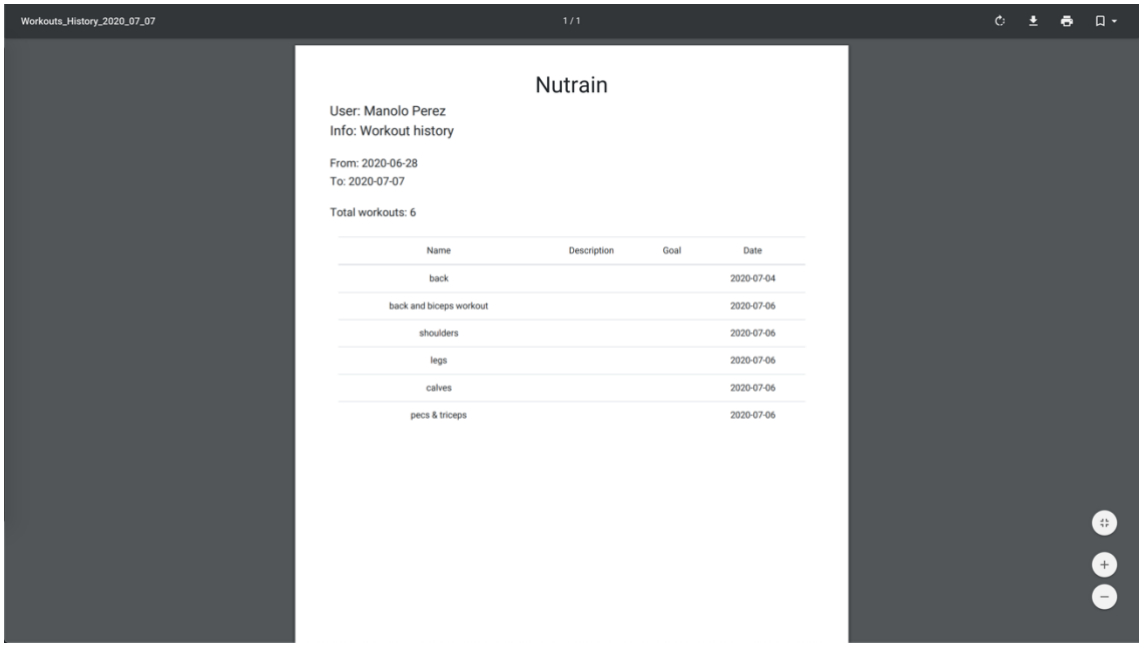

Figura A 14: Informe PDF de los entrenamientos realizados. Fuente: elaboración propia.

A.3.1.1 Gestión de rutinas de entrenamiento a tus clientes como usuario Profesional

El usuario Profesional tiene las mismas funcionalidades anteriormente nombradas, pues son comunes a todos los usuarios, pero además puede realizar dos funciones extras:

- Visualizar información extra sobre sus clientes
- Crear y editar una rutina de entrenamiento a sus clientes.

Si vamos a los detalles de uno de nuestros clientes, no sólo podremos ver sus datos básicos (lo cual ya podía hacer, aunque no fuera cliente suyo), sino que además podremos visualizar cierta información sobre la salud del usuario, como por ejemplo gráficos sobre las kilocalorías y macronutrientes del usuario en el último mes, sus últimos entrenamientos, ejercicios de psicología y fisioterapia, o un histórico de sus medidas corporales, tal y como podemos ver en las figuras A15, A16 y A17.

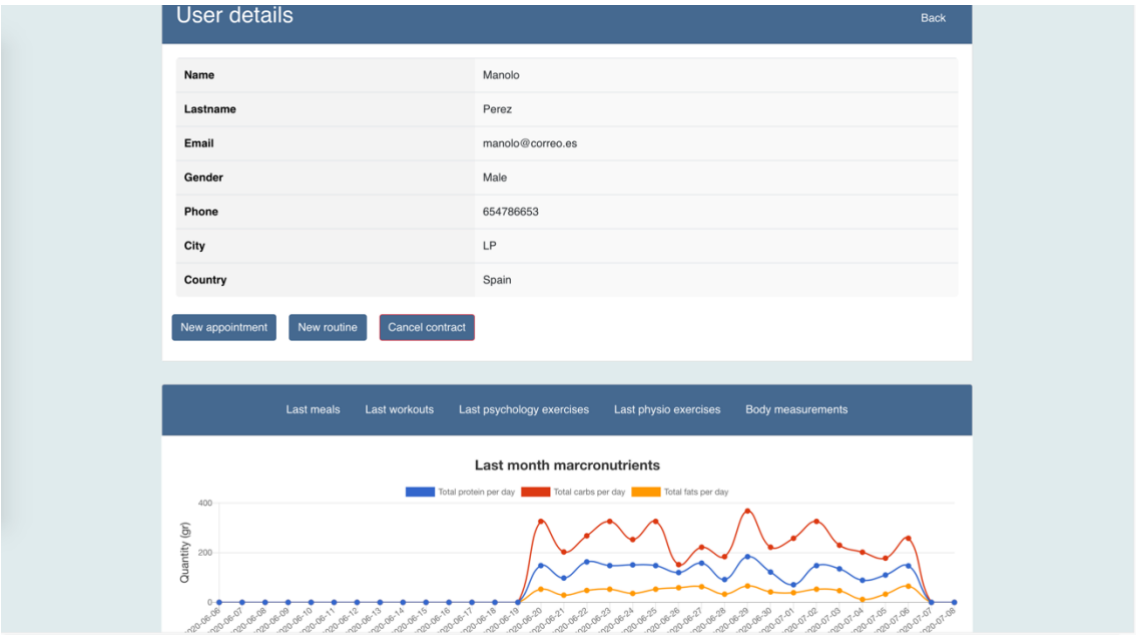

Figura A 15: Visualizar datos de un cliente como usuario profesional. Fuente: elaboración propia.

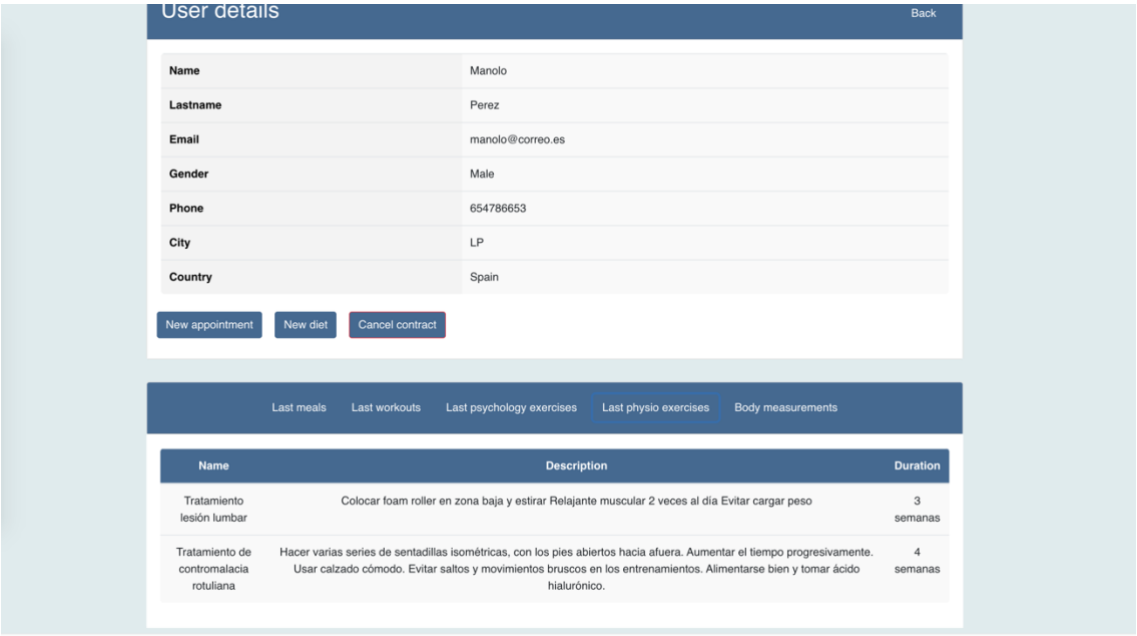

Figura A 16: Visualizar datos de un cliente como usuario profesional, parte dos. Fuente: elaboración propia.

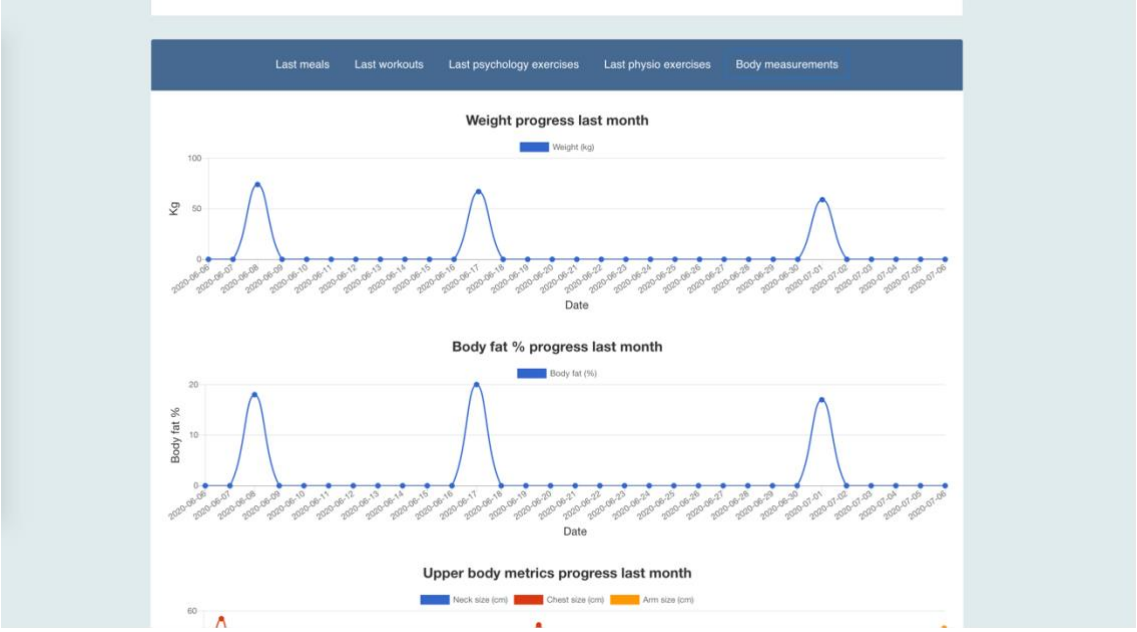

Figura A 17: Visualizar datos de un cliente como usuario profesional, parte tres. Fuente: elaboración propia.

En este caso, como somos un usuario profesional, podemos ver siempre datos relativos a los cuatro aspectos de la salud de nuestro cliente (nutrición, entrenamiento, psicología, fisioterapia), pero sólo podemos crear/asignar elementos relacionados a nuestra profesión (que viene determinado por los campos de la base de datos que se comentaba en el capítulo 4. En este caso, como el usuario profesional es entrenador, sólo

podemos crear/editar la rutina de entrenamiento de nuestro cliente. Si por ejemplo, fuéramos un nutricionista, podríamos editar su dieta, y así para los demás.

A.3.2 Gestión nutrición

En la barra de navegación disponemos de un menú *Nutrition,* con los correspondientes submenús *Diets* (dietas) y *Meals* (comidas). Si accedemos al menú de dietas, se nos muestra un listado de las dietas que hemos creado, similar a como vimos con las rutinas de entrenamiento.

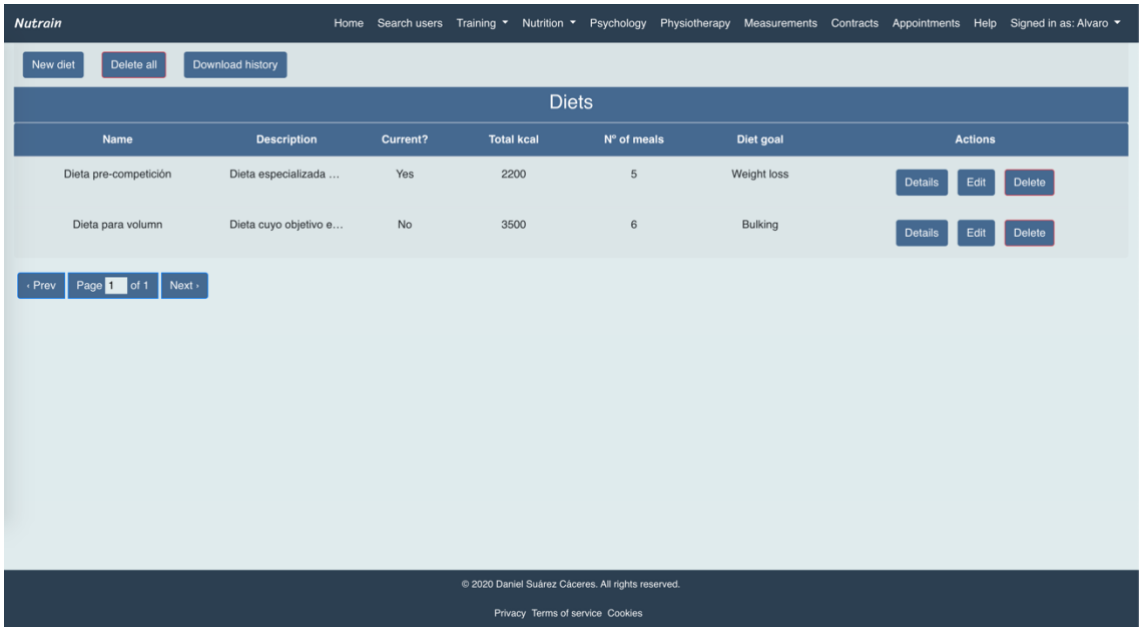

Figura A 18: Lista de dietas del usuario. Fuente: elaboración propia.

En la figura A18, podemos visualizar información básica sobre la dieta, como por ejemplo el nombre, descripción, objetivo de kilocalorías, etc. Si entramos en los detalles, veremos más información de la descripción, como en la figura A19.

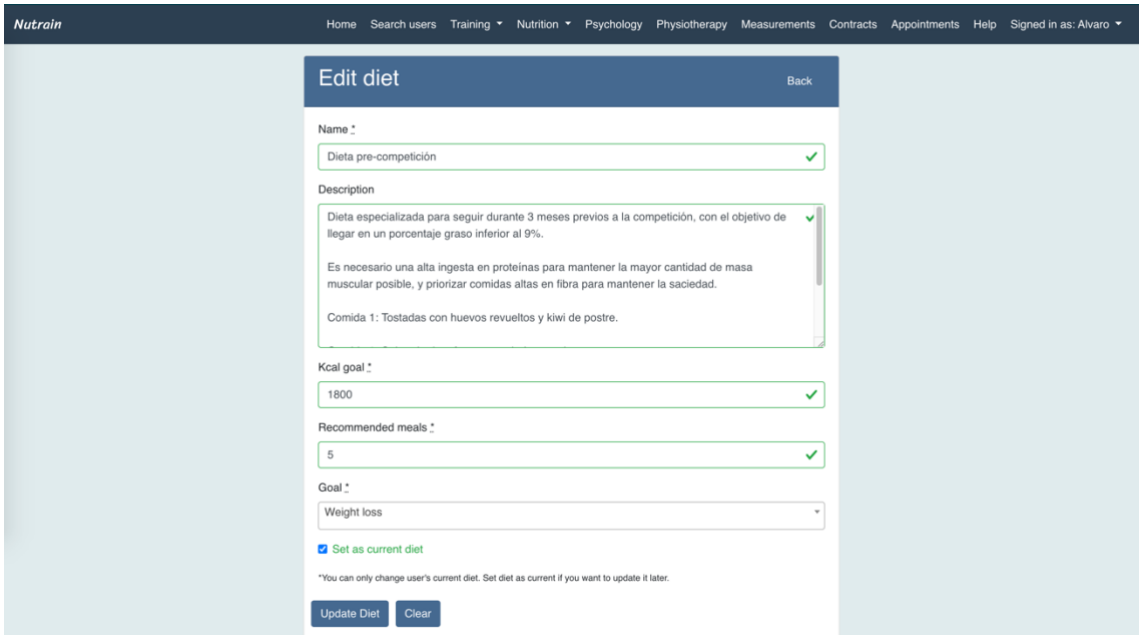

Figura A 19: Detalles sobre una dieta. Fuente: elaboración propia.

También podemos crear una nueva, borrar, editar, ver detalles, tal y como se hacía con las rutinas. También podremos descargar un informe PDF donde se nos mostrará información sobre las dietas.

En el menú *Meals* (comidas), podremos observar un listado de todas las comidas que hemos realizado. Podremos visualizar detalles, editar, borrar, etc. También podremos descargar un informe PDF que tendrá toda la información sobre las comidas, eligiendo un intervalo de fecha si lo deseamos, tal y como se hacía con los entrenamientos.

A3.2.1 Gestión de nutrición de tus clientes como usuario Profesional

Al igual que ocurría con las rutinas de entrenamientos, si en este caso el usuario profesional es un nutricionista, podremos crear y editar las dietas de nuestros clientes.

El formulario sería el mismo que se ha visto anteriormente, sólo que la rutina estaría siendo creada para otro usuario (en este caso, el cliente correspondiente).

A.3.3 Gestión psicología

En el menú superior, si hacemos click en *Psychology,* nos llevará al listado de nuestros ejercicios de psicología, como en la figura A20.

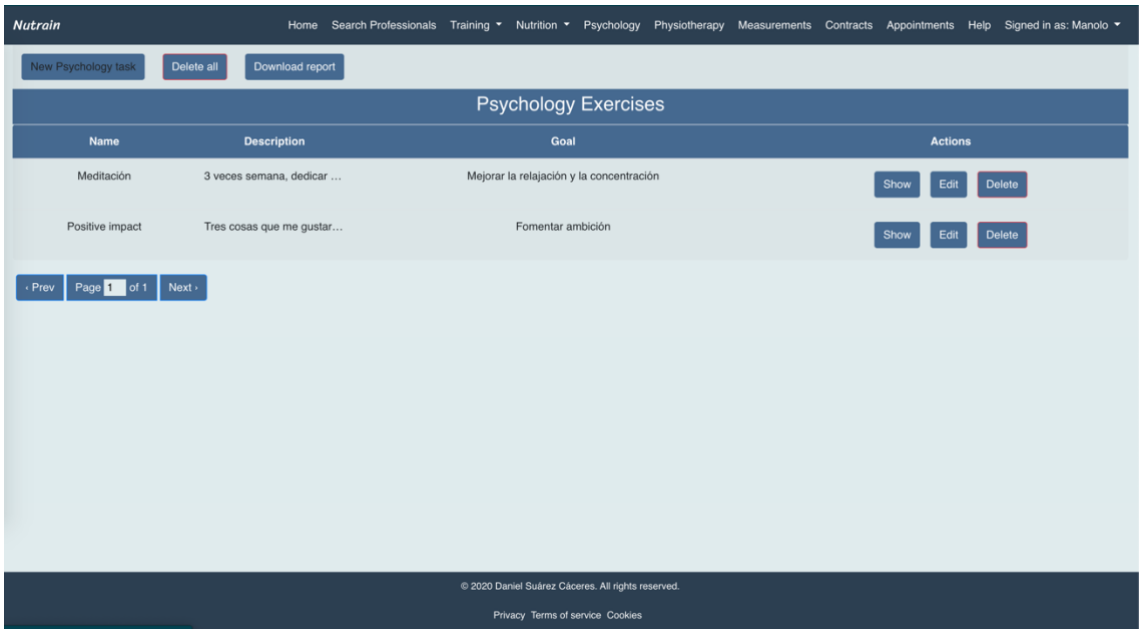

Figura A 20: Listado de ejercicios de psicología del usuario. Fuente: elaboración propia.

Del mismo modo que con los demás apartados, podremos visualizar los detalles, editar un ejercicio, borrar uno o borrar todos, crear uno nuevo o también descargar un informe PDF, en el cual también se pueden introducir un intervalo de fechas según nos interese.

El formulario sería como el que se observa en la Figura A21 (la apariencia del formulario es la misma tanto si lo creamos para uno de nuestros clientes como si lo creamos para nosotros mismos,).

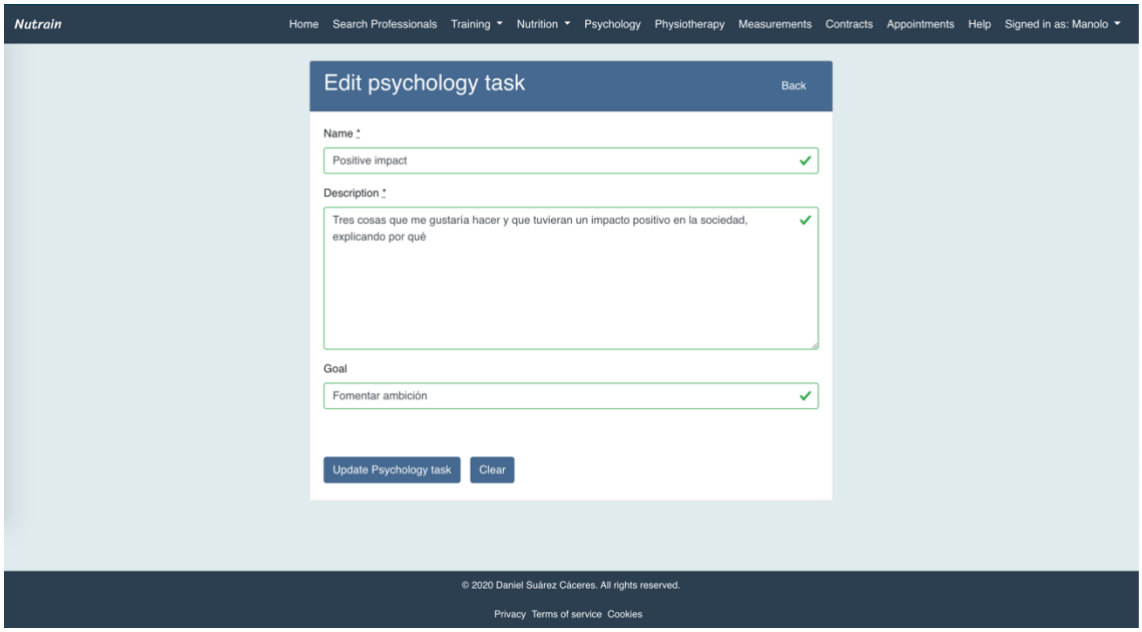

Figura A 21: Creación de un ejercicio de psicología. Fuente: elaboración propia.

A.3.3.1 Gestión de psicología de tus clientes como usuario profesional

Si somos un usuario profesional psicólogo, al igual que pasaba con las rutinas de entrenamiento o con las dietas si el profesional era entrenador o nutricionista, podremos crear ejercicios de psicología para nuestros clientes.

El formulario sería como el que se observaba antes en la Figura A21.

A.3.4 Gestión fisioterapia

En el apartado de fisioterapia (accediendo a través de la barra de navegación, en *Physiotherapy),* nos encontraremos con lo mismo que para los aspectos anteriores (nutrición, salud y psicología): podemos ver un listado, crear nuevos ejercicios, editarlos o descargar informes PDF.

A.3.4.1 Gestión de fisioterapia de tus clientes como usuario profesional

Al igual que ocurría con los demás tipos profesionales, si somos un fisioterapeuta podremos crear ejercicios y asignárselos a nuestros clientes. El formulario sería como el que se observa en la Figura A22.

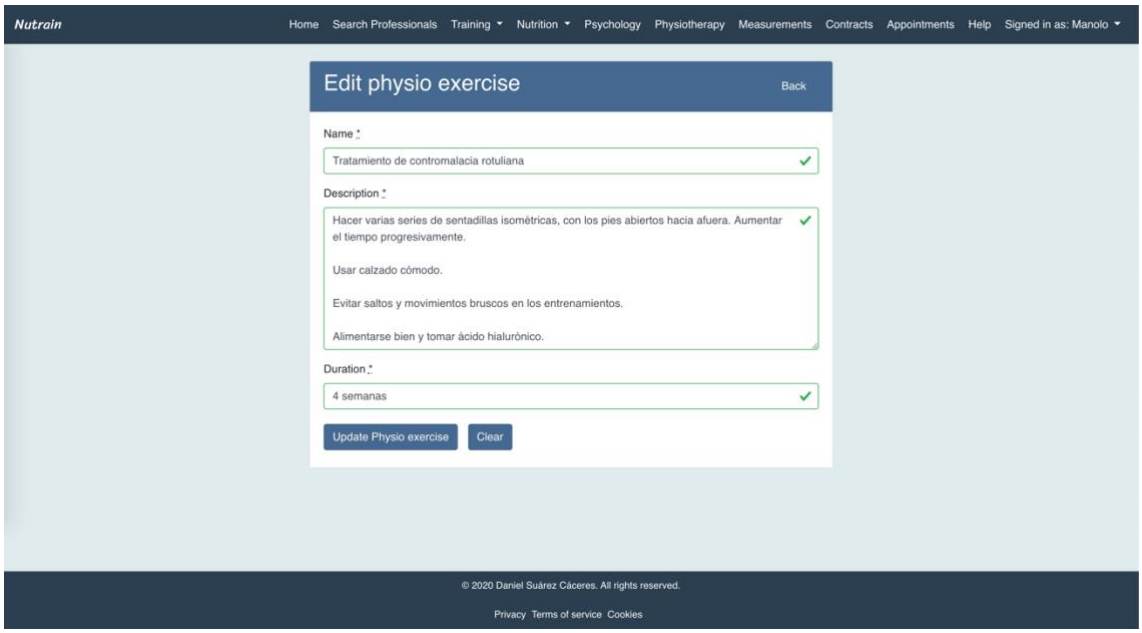

Figura A 22: Creación ejercicio fisioterapia. Fuente: elaboración propia.

A.3.5 Gestión medidas corporales

El último apartado en lo que a la salud del usuario respecta, sería el de las medidas corporales, accediendo a través de la barra de navegación, en *Measurements.* El usuario puede introducir un conjunto de medidas corporales (peso, porcentaje de grasa, diámetro de brazos, pecho, etc.) y anotando el día correspondiente.

Al igual que con los demás apartados, podremos ver un listado de las medidas corporales, crear una nueva, editarlas, borrarlas y descargar informes PDF. En la figura A23 se observa un formulario de ejemplo para introducir un conjunto de medidas corporales.

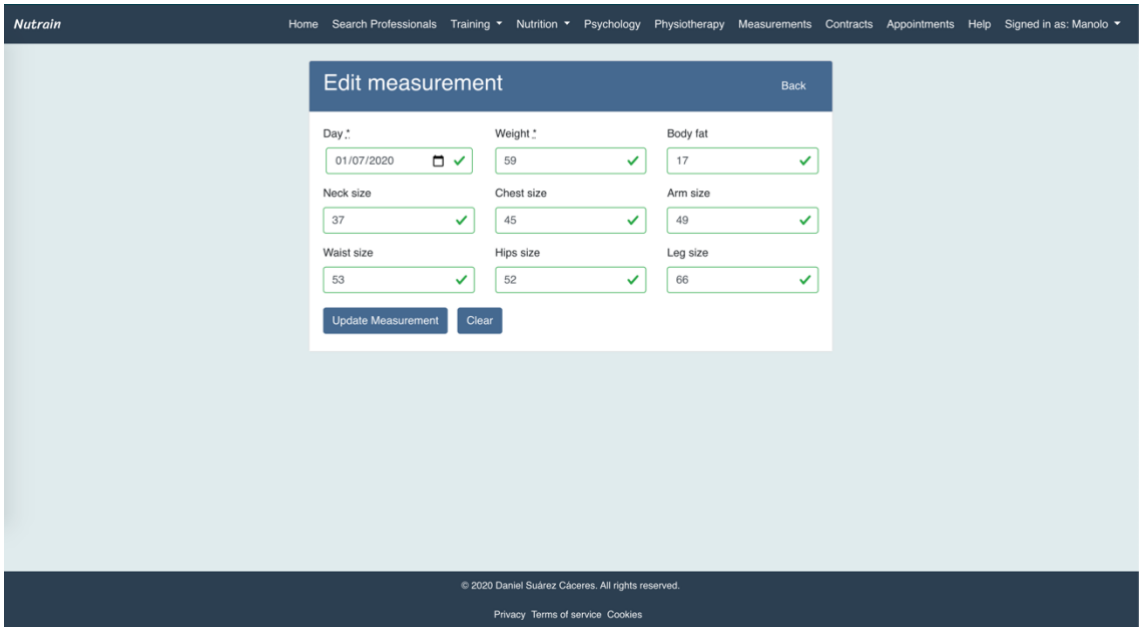

Figura A 23: Creación de conjunto de medidas corporales. Fuente: elaboación propia.

En la pantalla a la que se nos lleva tras hacer click en *Measurements,* no sólo podremos ver un listado (como en la figura A23), sino que también se presenta al usuario diferentes informes de diferentes intervalos de tiempo. En este caso, el usuario puede ver un histórico de sus medidas corporales en el último mes, los últimos tres meses o los últimos seis meses, tal y cómo se observa en la figura A24. Para cada intervalo de tiempo, hay cuatro gráficos: uno para el peso, otro para el porcentaje graso, otro para las medidas del tren superior (cuello, pecho, brazos) y otro para las medidas del tren inferior (cintura, cadera, piernas), como se observa en la figura A25.

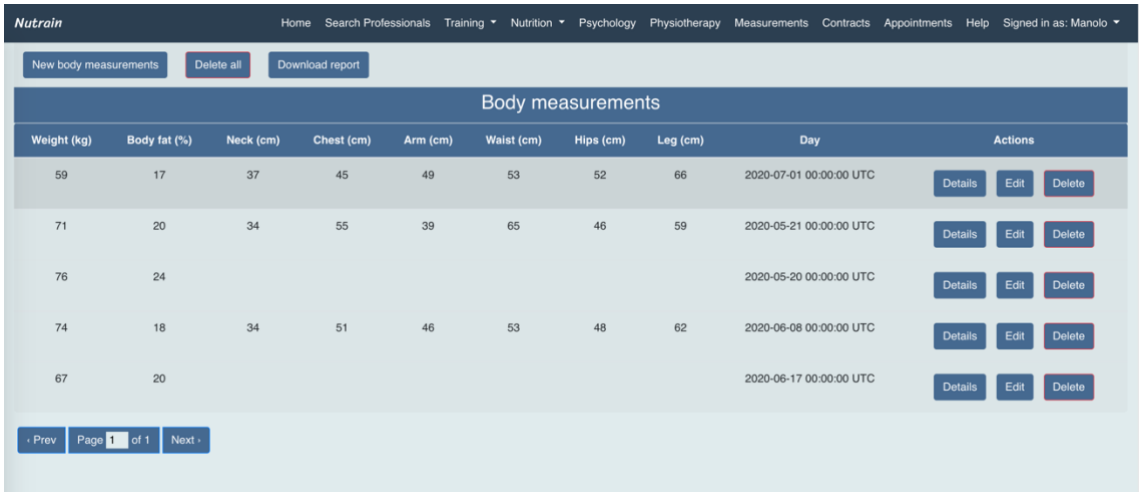

Figura A 24: Listado de medidas corporales del usuario. Fuente: elaboración propia.

Figura A 25: Gráficos históricos de las medidas corporales. Fuente: elaboración propia.

A.4 Buscar clientes/profesionales

En la barra de navegación tenemos una opción *Search users*, que nos permitirá ver el listado de clientes que están inscritos en la plataforma (si somos un profesional), o el listado de profesionales que están inscritos en la plataforma (si somos un cliente).

En este listado, podremos ver información sobre los usuarios, y solicitar citas y también mandar una solicitud para iniciar un contrato, como se observa en la figura A26.

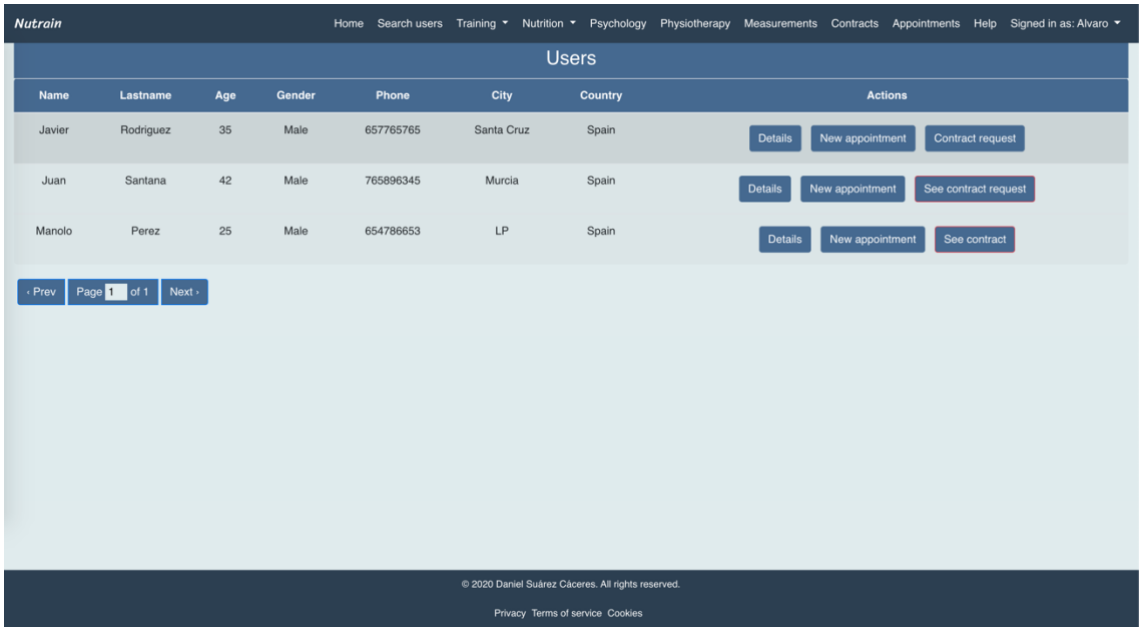

Figura A 26: Listado de clientes. Fuente: elaboración propia.

Como se puede observar, para los clientes con los que no tengamos contrato, nos aparecerá una opción *Contract request.* Con los que tengamos un contrato pendiente de aceptar/rechazar, aparecerá la opción *See contract request,* y con los que ya tengamos un contrato aceptado, aparecerá la opción *See contract.* Para un cliente, la vista sería igual, solo que lo que veríamos serían usuarios profesionales en vez de usuarios clientes.

A.5 Gestión contratos

Similar al punto anterior, disponemos de una página donde exclusivamente veremos los contratos y los respectivos clientes con los que los tenemos (es decir, el resto de usuarios de la plataforma no se visualizan en este apartado). En esta vista, como se observa en la figura A27, podemos ver los contratos que tenemos, el estado en que se encuentran (“*Pending”* si la solicitud aún no ha sido aceptada/rechazada, *“Active”* si la solicitud fue aceptada y el contrato ya está activo), los clientes con los que los tenemos y la fecha de inicio en caso de que ya haya sido aceptado. Si somos un usuario cliente, tendríamos la misma vista, cambiando los clientes por profesionales.

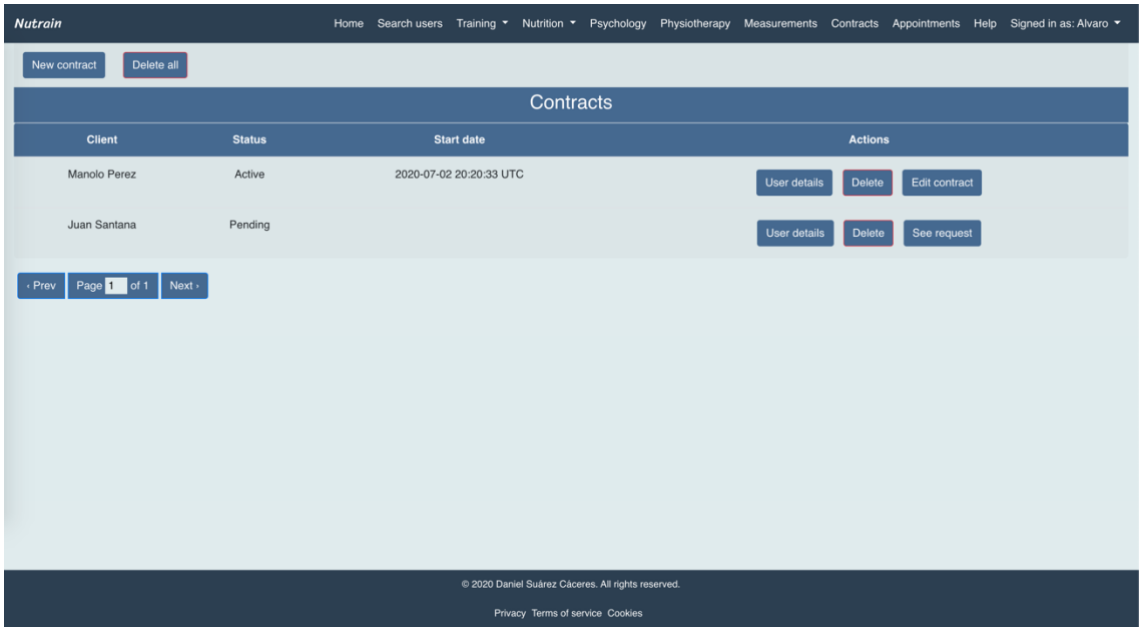

Figura A 27: Lista de contratos. Fuente: elaboración propia.

Para crear un nuevo contrato, se mostraría un formulario como el de la Figura A28, donde podemos seleccionar el cliente y escribirle un mensaje.

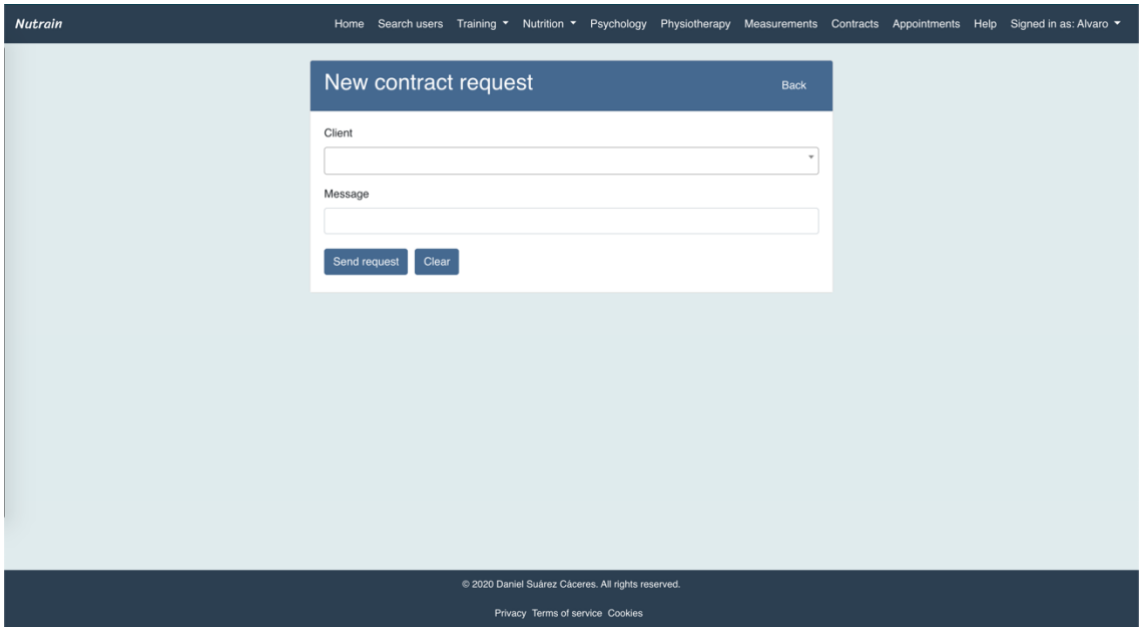

Figura A 28: Creación de un contrato. Fuente: elaboración propia.

A.6 Gestión citas

El último apartado respecto a las relaciones cliente-profesional, sería la gestión de citas. Podemos solicitar citas con clientes (si somos un profesional) o con profesionales (si somos un cliente) desde la vista de usuarios que veíamos en la figura A26. No obstante, disponemos de una vista más adecuada a la que podemos acceder a través de la barra de navegación en *Appointments,* que nos llevaría a la pantalla que se ve en la figura A29.

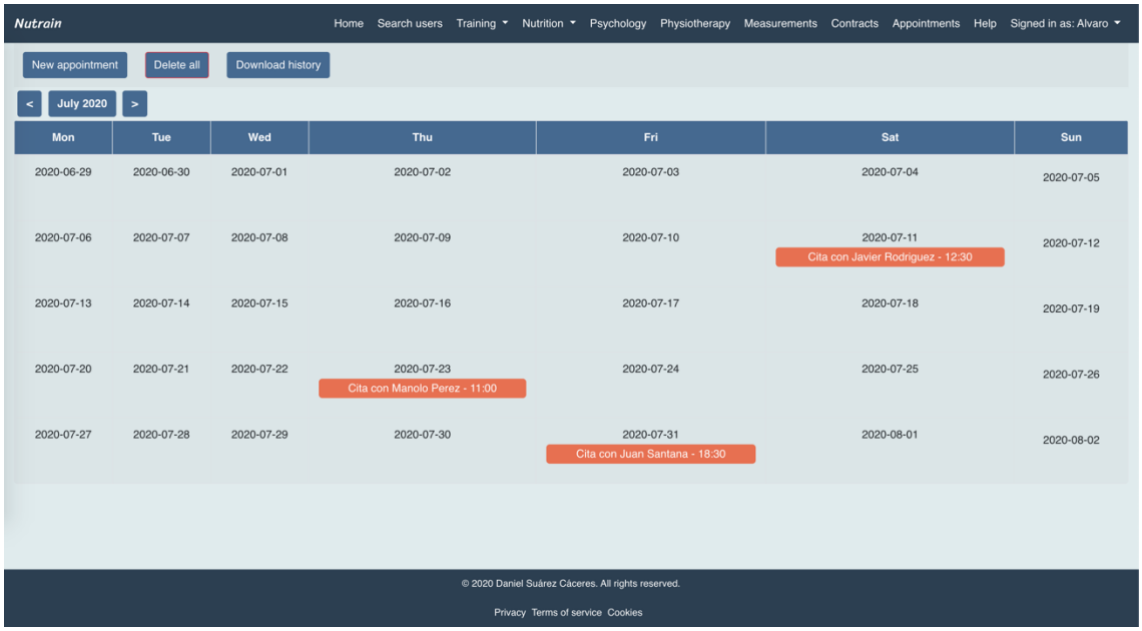

Figura A 29: Vista de gestión de citas. Fuente: elaboración propia.

Tenemos una vista de calendario, donde podemos navegar entre los diferentes meses del año. Podemos crear una nueva cita, borrar, editar, descargar informes PDF, al igual que ocurría con los demás apartados.

A.7 Contactar con la página

Por último, todo usuario dispone de un apartado para contactar con la página, accediendo a través de la barra de navegación, en *Help.* Esta página está destinada a que el usuario envíe correos comunicando algún tipo de error, o por ejemplo si eres un profesional, para mandar tu código de colegiado y que la página verifique que efectivamente estás en

posesión de ese título. El formulario sería el que se observa en la figura A30.

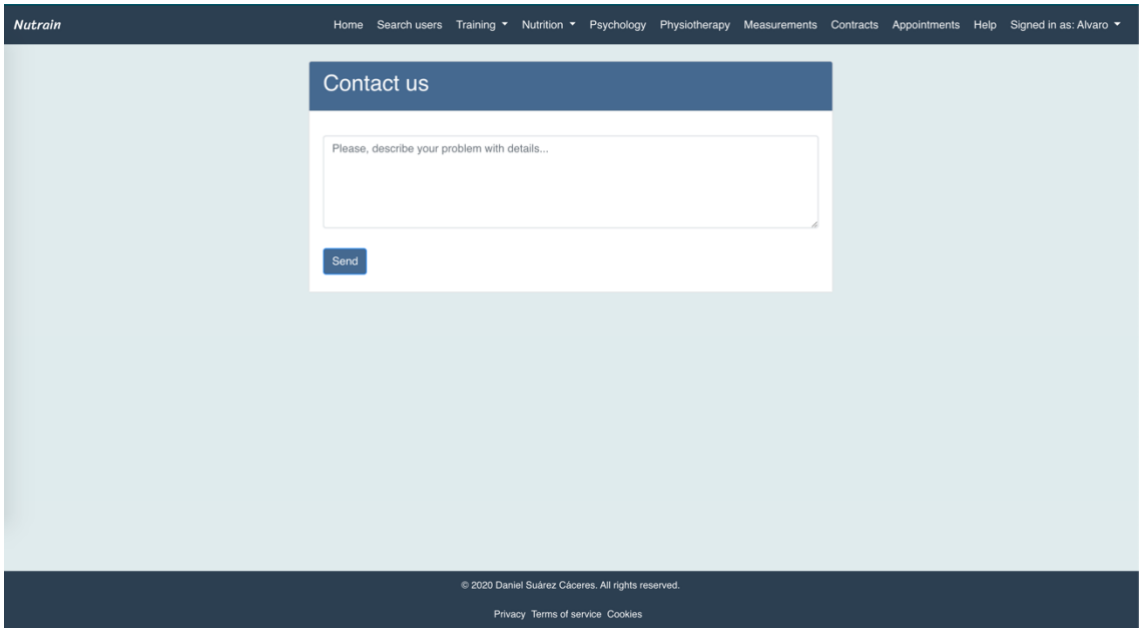

Figura A 30: Formulario de contacto/ayuda. Fuente: elaboración propia.

Apéndice B: Legislación relativa al proyecto.

B.1 Ley de protección de datos

Como todo proyecto de software que trabaje con datos personales del usuario, la aplicación desarrollada también debe cumplir una serie de requisitos legales, con el fin de garantizar la seguridad y privacidad de los datos del usuario.

Dichas garantías se encuentran recogidas en el documento del Boletín Oficial del Estado (BOE) en la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos y Garantía de los Derechos Digitales (LOPDGDD).

La ley obliga a las empresas, organismos e instituciones a tomar las medidas necesarias para garantizar la protección de los datos que el usuario ha confiado a la página.

También se deberá realizar un registro de las actividades de tratamiento de datos, indicando el tipo de datos que recopilas, la finalidad del tratamiento y la política de tratamiento de estos datos.

Otro punto importante es garantizar los derechos de los usuarios, tales como: acceso a sus datos personales, rectificación en caso de que algunos de los datos sean inexactos, limitar el tratamiento de los datos, derecho a exigir la eliminación de los datos cuando estos no se tratan correctamente o para otra finalidad diferente a la definida, entre otras.

El objetivo final es garantizar que las actividades que realicen las empresas y organismos se hagan siempre dentro de un marco legal, garantizando y protegiendo en todo momento la información de sus usuarios.

Bibliografía

[1] Fitness Industry cashing $100.000 million dollars globally, Business Insider.

https://www.businessinsider.com/fitness-has-exploded-into-a-nearly-100-billion-global-industry-2019-9?IR=T

[2] Why fitness industry is growing so fast, GloFox.

https://www.glofox.com/blog/fitness-industry/#:~:text=The%20global%20health%20club%20industry,hit%20%24106%20billion%20in%202020.

[3] Total number of memberships to health centers in the US from 2000 to 2017, Statista.

https://www.statista.com/statistics/236123/us-fitness-center--health-club-memberships/#:~:text=U.S.%20fitness%20center%20%2F%20health%20club%20memberships%202000%2D2017&text=This%20statistic%20shows%20the%20number,total%20membership%20of%2060.87%20million.

[4] Overview of Sport and Nutrition market in 2019, Grand View Research.

https://www.grandviewresearch.com/industry-analysis/sports-nutrition-market

[5] How technology is changing the fitness industry, IT Brief.

https://itbrief.com.au/story/how-technology-is-changing-the-fitness-industry

[6] Algunas de las alternativas existentes en el mercado: MyFitnessPal, Cronometer, FatSecret.

https://www.myfitnesspal.com/es

https://cronometer.com/

https://www.fatsecret.es/

[7] Guía de instalación Git, web oficial Git.

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

[8] Guía para descargar Ruby, web oficial Ruby.

https://www.ruby-lang.org/es/downloads/

[9] Guía para descargar Rails, Tutorialspoint.com.

https://www.tutorialspoint.com/ruby-on-rails/rails-installation.htm

[10] Guía para descargar e instalar NodeJS, web oficial NodeJS.

https://nodejs.org/es/

[11] Guía para descargar e instalar PostgreSQL, web oficial PostgresSQL.

https://www.postgresql.org/download/

[12] Guía para descargar e instalar Visual Studio Code, web oficial VSCode.

https://code.visualstudio.com/download

[13] Definición de Kanban, IEBSchool.

https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/

[14] Definición de patrón MVC, Wikipedia.

https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[15] Estructura de un proyecto Ruby on Rails, web official Rails.

https://guides.rubyonrails.org/getting_started.html