Desarrollo de una aplicación móvil para recomendaciones de hábitos saludables
Development of a mobile application for healthy habits recommendations.

Trabajo de Fin de Grado
Curso 2024–2025

Autor
Rodrigo Souto Santos

Director
José Ignacio Hidalgo Pérez

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Desarrollo de una aplicación móvil para recomendaciones de hábitos saludables Development of a mobile application for healthy habits recommendations.

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Rodrigo Souto Santos

Director
José Ignacio Hidalgo Pérez

Convocatoria: *Junio 2025*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

10 de Mayo de 2025

Dedicatoria

A todas las personas que me han acompañado
durante esta etapa

Resumen

Desarrollo de una aplicación móvil para recomendaciones de hábitos saludables

En una época dominada por los alimentos procesados y el sedentarismo, una de las mayores amenazas a la integridad de la salud de las personas viene dada por los malos hábitos alimenticios y físicos.

Este trabajo de fin de grado presenta el desarrollo de la aplicación,**VitHabitus**. Enfocada en la generación de recomendaciones sobre hábitos saludables, con el objetivo de ayudar a las personas a seguir una rutina lo más saludable posible.

La aplicación VitHabitus se centra en una interfaz simple e intuitiva que facilita su uso y entendimiento. Mediante la integración de Firebase, se garantiza una gestión segura y eficiente de los datos de los usuarios.

Para su desarrollo, han sido empleadas tecnologías como Node Js, TypeScrpit y React native. Cuenta además, con una API REST enfocada en la gestión del recomendador de hábitos saludables y en su conexión con Firebase.

Palabras clave

Recomendador, Firebase, React Native, API, aplicación móvil, node JS, spring boot, base de datos, ORI.

Abstract

Development of a mobile application for healthy habits recommendations.

In a time dominated by processed food and sedentary lifestyles, one of the greatest threats people's health have comes from poor dietary and physical habits.

This bachelor's Thesis presents the development of the VitHabitus, an application focused on generating custom recommendations for healthy habits, with the aim of helping people follow the healthiest routine possible.

VitHabitus is built around a simple an intuitive interface that facilitates its use and understanding. Through the integration of Firebase, the application ensures secure and efficient managment of user data.

For the development, technologies such as Node JS, TypeScrpit, and React Native have been used. It also includes a REST APIdesigned to manage the healthy habits recommender and to connect with the Firebase.

Keywords

Recommender, Firebase, React Native, API, mobile application, node JS, spring boot, data base, ORI.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Cualquiera que sostenga una opinión verdadera sobre un tema que no entiende, es como un hombre ciego en el camino correcto"
— Sócrates

1.1. Motivación

En los ultimos años, la conciencia de las personas por mantener un estilo de vida saludable ha ido en aumento. Debido principalmente a que países con gran relevancia se han visto gravemente afectados por la creciente presencia de enfermedades como el sobrepeso, la obesidad y sus consecuencias en la salud a largo plazo. World Health Organization (2024)

Sin embargo, no para todas las personas ni en todas las situaciones es fácil identificar el problema o encontrar los hábitos que hay que moficar, así como la manera de hacerlo. Es por eso, que el campo de la nutrición ha experimentado un desarrollo en los últimos años, no solo con el aumento de profesionales si no también, con el avance de nuevas tecnologías que facilitan la tarea. Ismael San Mauro Martín1 (2014)

La proliferación de relojes, sensores, aplicaciones móviles,etc. Han proporcionado a las personas herramientas con las cuales pueden autoevaluarse a pesar de carecer de un enfoque realmente individualizado y personal. Es por ello que surge la necesidad de desarrollar, sistemas y algoritmos ajustables que permitan realizar recomendaciones precisas en base a la situación de cada individuo. Francesc Alòs (2021)

Este Trabajo de Fin de Grado surge con la motivación de fusionar esta rama de la salud basada en los hábitos de las personas, con el desarrollo de software. Creando una aplicación que otorge a los usuarios la capacidad de utilizar un algoritmo evolutivo para mejorar sus habítos semanales de la forma más personal posible.

1.2. Objetivos

El principal objetivo de este trabajo es diseñar e implementar una aplicación móvil compatible con software android y ios, que funcione como un asistente personalizable de hábitos saludables, combinando un diseño simple e intuitivo junto con un sistema de recomendación entrenado basado en un algoritmo evolutivo.

Entre los principales objetivos de este proyecto se encuentran:

- Desarrollar la estructura de la aplicación móvil con el framework React Native y el entorno de desarrollo Expo.
- Desarrollar una interfaz intuitiva y simple que divida en categorías los hábitos.
- Desarrollar un sistema de autenticación robusto mediante Firebase.
- Implementar una base de datos en Firestore para la gestión de información de los usuarios.
- Implementar una API REST en spring boot que integre el algoritmo evolutivo recomendador".
- Garantizar la sincronización segura de los datos y el funcionamiento en diferentes dispositivos.

1.3. Plan de trabajo

Para poder conseguir cada uno de los objetivos, se ha desarrollado un plan de trabajo para afrontarlos de la mejor manera posible:

1.3.1. Estrcuctura y entorno de desarrollo

- Con el fin de obtener el mejor resultado posible, se investigó sobre el mejor framework que recogiese las condiciones. React Native.(Axarnet (2025)).
- Se definió la estructura general de la app, componentes, apariencia general,etc.
- Uso de Expo para poder ejecutar y probar la aplicación.

1.3.2. Interfaz intuitiva

- Investigación de otras aplicaciones móviles para obtener referencias sobre interfaces simples e intuitivas.
- Uso de Figma para desarrollar componentes visuales aplicables al coódigo.

1.3.3. Autenticación

- Investigación sobre la plataforma que mejor se adapte al objetivo. Firebase
- Creación de una cuenta en la plataforma y sincronización con la aplicación
- Desarrollo de las diferentes pantallas y componentes necesarios para el correcto funcionamiento de la autenticación segura.

1.3.4. Base de Datos

- Uso de la cuenta de Firebase para la creación de la base de datos con Firestore con una base de datos noSql.
- implementación de la estructura de las colecciones y documentos que se almacenarán.

1.3.5. API REST

- Uso de Spring Boot Suite(STS) para desarrollar la API con formato el rest
- Implementación del código del recomendador, adaptandolo a los requisitos de la API y la conexión con la base de datos
- Vinculación e integración correcta y segura con la aplicación móvil.

1.3.6. Sincronización

- Garantizar correcto funcionamiento constante de la API y de Firebase.
- Investigación sobre Expo y sus funcionalidades multiplataformas para un correcto funcionamiento en ambos sistemas operativos.

1.4. Recomendador de Hábitos Saludables

El sistema de recomendación utilizado en esta aplicación ha sido desarrollado y entrenado previamente en el contexto de un Trabajo de Fin de Grado de los grados de Ingenieria Informática y de Software en la Universidad Complutense de Madrid. Titulado **Recomendador de Hábitos para reducir el riesgo de padecer Sobrepeso y Obesidad"**, realizado por **Fan Ye y Daniel Martínez**, bajo la dirección de Jose Ignacio Hidalgo Pérez.

El recomendador conforma el núcleo funcional de la aplicación. Es el encargado de procesar los hábitos que el usuario haya itnroducido y generar en consecuencia ciertas recomendaciones personalizadas orientas en mejorar la salud y reducir la obesidad. Este sistema de recomendación está basado en un algoritmo evolutivo y en

el uso de ciertos modelos para obtener un coeficiente ORI (Obesity Risk Index,Indice de Riesgo de Obesidad) que represente la calidad de los hábitos.

Finalmente se realizó una integración de este recomendador en la aplicación mediante el desarrollo de la API rest en el entorno de Spring Boot Suite.

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter 1.

Motivation

In recent years, the awareness of people to maintain a healthy lifestyle has been increasing. This is mainly due to the fact that countries with high relevance have been seriously affected by the growing presence of diseases such as overweight, obesity and their long-term health consequences. World Health Organization (2024)

However, not for all people or in all situations it is easy to identify the problem or to find the habits to mitigate. That is why the field of nutrition has experienced a development in recent years, not only with the increase of professionals but also with the advance of new technologies that facilitate the task. Ismael San Mauro Martín1 (2014)

The proliferation of watches, sensors, mobile applications, etc. The time has provided people with tools with which they can evaluate themselves despite lacking a truly custom and personal approach. This is why there is a need to develop adjustable systems and algorithms that allow precise recommendations to be made based on each individual's situation.

Francesc Alòs (2021)

This bachelor's Thesis begins with the motivation to merge this branch of health based on the habits of people, with the development of software. Creating an application that gives users the ability to use an evolutionary algorithm to improve their weekly habits in the most personal way possible.

Objectives

The main objective of this work is to design and implement a mobile application compatible with android and iOS software, which works as a customizable healthy

habits assistant, combining a simple and intuitive design together with a trained recommendation system based on an evolutionary algorithm.

Between the main objectives of this project we hightlight:

- Develop the mobile app using the React Native framework and the Expo development environment.
- Develop an intuitive and simple interface that divides habits into categories.
- Develop a robust authentication system using Firebase.
- Implement a database in Firestore for managing user information.
- Implement a REST API in spring boot that integrates the evolutionary algorithm 'recommender'.
- Ensure the secure data synchronization and operation across devices.

Work Plan

In order to achieve the objectives, a work plan has been developed to get them in the best possible way:

Structure and development environment

- In order to obtain the best possible result, A research was done about React Native, the best framework that would approach the conditions wanted. (Axarnet (2025)).
- The general structure of the app, components, general appearance, etc. was defined.
- Use of Expo to be able to run and test the application.

Intuitive interface

- Research of other mobile applications to get references on simple and intuitive interfaces.
- Use of Figma to develop visual components applicable to the code.

Authentication

- Research on the platform that best suits the objective. Firebase.

- Creation of an account on the platform and synchronization with the application.
- Development of the different screens and components necessary for the correct work of the secure authentication.

Database

- Use of the Firebase account for the creation of the database with Firestore with a noSql database.
- Implementation of the structure of the collections and documents to be stored.

API REST

- Use of Spring Boot Suite(STS) to develop the API with rest format
- Implementation of the recommender code, adapting it to the requirements of the API and the connection with the database
- Correct and secure linking and integration with the mobile application.

Synchronisation

- Ensuring consistent correct functioning of the API and Firebase.
- Research on Expo and its cross-platform functionalities for a correct functioning in both operating systems.

Healthy Habits Recommender

The recommendation system used in this application has been previously developed and trained in the context of a bachelor's Thesis of the Computer and Software Engineering degrees at the Complutense University o Madrid. Entitled "Habits Recommender to reduce the risk of overweight and obesity", carried out by Fan Ye and Daniel Martínez, under the supervision of Jose Ignacio Hidalgo Pérez.

The recommender forms the functional core of the application. It is responsible for processing the habits that the user has produced and consequently generating certain custom recommendations aimed to improving health and reducing obesity This recommendation system is based on an evolutionary algorithm and the use of certain models to obtain an ORI (Obesity Risk Index) coefficient that represents the quality of the habits. Finally, this recommender was integrated into the application by developing the API rest in the Spring Boot Suite environment.

Capítulo 2

Descripción del Trabajo

Este capítulo contiene una descripción completa del trabajo realizado en este proyecto. Explica de forma detallada la estrcutura establecida para el desarrollo de la aplicacioón, asi como su arquitectura y todos los elementos necesarios para llevarla a cabo. Además incluye una descripción de la api utilizada para poder sincronizar y vincular la aplicación con la base de datos y nuestro recomendador.

2.1. Estructura general

Una de las primeros pasos en el desarrollo de esta aplicación móvil llamada VitHabitus es la estructura general seguida para el funcionamiento de la misma.

VitHabitus, tiene como objetivo ofrecer a los usuarios una herramienta adicional para poder mejorar sus hábitos saludables y llevar un registro de sus avances. Ha sido desarrollada como una aplicación móvil multiplataforma para android e iOS, con el objetivo de alcanzar al mayor número de usuarios con la mayor comodidad posible.

Para su desarrollo se ha utilizado el editor de código Visual Studio Code, el cuál cuenta con gran compatibilidad con el framework React Native y el entorno de desarrollo Expo, que son los pilares de esta aplicación.

Además, el sistema de la aplicación almacena todos sus datos en Firebase como solución backend integral tanto para autenticación y base de datos y almacenamiento; y una API REST desarrollada en Spring Boot para el cálculo de las recomendaciones de hábitos personalizadas con el recomendador.

Centrándonos en los requisitos funcionales podríamos listarlos de la siguiente manera:

2.1.1. Requisitos Funcionales

2.1.1.1. Autenticación

- Registro de usuarios mediante correo electrónico y contraseña.
- Funcionalidad de inicio de sesion y crear cuenta con opción de autenticación con provedores como google, apple, etc.
- Cerrado de sesión y manejo de errores en caso de correo incorrecto, contraseña inválida o de seguridad débil, falta de documentos en el creado de una cuenta, etc.

2.1.1.2. Perfil

Todos los datos relacionados al perfil de un usuario.

- Campos comonombre de usuario y apellidos, teléfono móvil(opcional)
- Foto de perfil (opcional), género y otros campos que pueden almacenarse desde el perfil de usuario y que facilitan el autocompletado en futuras recomendaciones de hábitos. (Información general del usuario).
- Funcionalidades: Almacenar esos datos y poder exportarlos al reocomendador cuando sea necesario.

2.1.1.3. Hábitos

Pantalla “hábitos”:

- Formulario con odos los campos a rellenar para poder ejecutar el reocmendador.
- Posibilidad de autocompletar los campos según los descritos en el perfil.
- Manejo de errores en caso de introducir datos incorrectos, con una validación de entrada.
- Botón de carga de parametros alojados en la base de datos para ahorrar tiempo en el rellenado y otro botón para Ejecutar el recomendador y obtener la recomendación que se alojará en la base de datos.
- indicador de carga para evitar notificar al usuario del correcto funcionamiento de la aplicación a pesar de su tiempo de espera.
- Genera un historial en la base de datos sobre la nueva ejecución de hábitos.

2.1.1.4. Resultados

Pantalla “Resultados”, muestra una gráfica con la valoración de los hábtios de la última ejecución, con el valor ORI:

- Muestra una g´rafica para mayor entendimiento visual al usuario sobre la calidad de sus hábitos. Cuenta con un apartado con las recomendaciones de los cambios y un recuadro para poder indicar si se ha completado y en ese caso poder actualizar directamente los valores.
- Historial visible para que el usuario pueda ver sus avances con el paso del tiempo.

2.1.1.5. Notas

Pantalla de “Notas” con el objetivo de permitir a los usuarios tomar notas relacionadas con sus hábitos, rutinas de entrenamiento o cualquier otra cosa sin tener que salir de la aplicación ni usar otras externas. Funcionalidades:

- Crear notas con un título máximode 110 caracteres, contenido ilimitado, autoguardado tras escritura. Tras la creación de la nota se ordena cronologicamente según su creación. Además de la posiblidad de cambiar el orden de las notas arrastrandolas por la pantalla.
- Lectura del título de la nota y los primeros caracteres del contenido sin necesitad de abrirla.
- Actualización y borrado de las notas en la base de datos, tras su edición y borrado en la aplicación.

2.1.1.6. Calendario

Un calendario interactivo cpara poder ubir la fecha actual y poder vincular notas pequeñas a modo de recordatorios.

2.1.2. Especificaciones Técnicas

Para las especificaciones técnicas destaca Firebase y ciertos componentes de la arquitectura. Firebase:

- Usar Firebase authentication para el inicio de sesión, y Firebase Scurity Rules para la seguridad.
- Almacenar las notas en la base de datos con los campos: id único, user-id, titlenote, content, created-at y updated-at; lo que nos permita una mejor gestión de las mismas. De la misma manera se almacenaran los usuarios con su id único, su user-id con el nombre, apellidos, etc

2.1.3. Especificaciones no funcionales

- Rendimiento: Cargar los hábitos, y las notas en caché para un uso rápido gracias a Firebase y el uso de índices que ayuden a mejorar la velocidad de consultas en la base de datos.
- Seguridad: Con API keys que proporciona Firebase para que la api acceda de forma segura.
- Además es importanto llevar un cierto manejo de errores para evitar "crashes" de l aaplicación y mejorar la funcionalidad.(Global error boundary for crashes).

2.1.4. Riesgos

Dentro de los riesgos, destacan los problemas de sincronización de la Firebase con la API y la app, por lo que es necesario un seguimiento y un testing con diferentes usuarios.

2.2. Arquitectura

La arquitectura se ha centrado en priorizar la modularidad y escalabilidad de la aplicación. Principalmente destacan tres bloques que conforman la estrcutura del proyecto:

- **Backend** o servicios en la nube, siendo principalmente Firebase y sus respectivas funcionalidades (fireStore, authentication,etc.)
- **Frontend**, el código de la aplicación móvil que implementa las funcionalidades y la interaz de usuario de la aplicación.
- **API Rest**, estructura desarrollada en Spring Boot que contiene el recomendador con sus funcionalidades.

Dada la arquitectura que se puede ver en la (figura 2.1), es importante definir las bases del flujo de la aplicación en cada uno de los procesos.

El usuario abre la aplicación y se encuentra con una pantalla de inicio de sesión/-creación de cuenta. En este momento entra en juego Firebase Authentication, lo cual garantiza el acceso a los datos privados del usuario y sus seguridad y sincronización.

Una vez logueado en la aplicación se despliega la página de inicio home desde la cual puede navegar a las diferentes oportunidades que ofrece. Por un lado dando al botón de ajustes, tras abrirse un desplegable aparece la opción de cerrar sesión y la de editar el perfil. En el caso de cerrar la sesión se enviará a Firebase la notificación de fin de sesión para seguir manteniendo la privacidad y correcto funcionamiento de la aplicación. Por otro lado, en la parte de perfil, se navega a la pantalla perfil, en la cual está definida la foto y de perfil y los demás campos que representan los

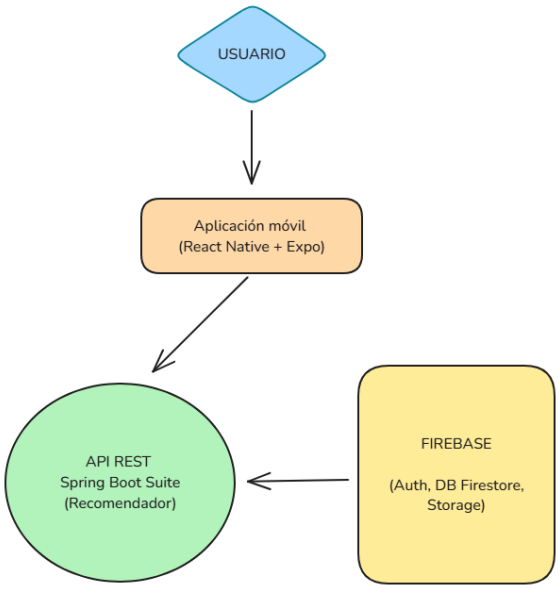

Figura 2.1: Arquitectura proyecto

datos personales del usuario. Desde aquí se pueden realizar las modificaciones y se guardarán automaticamente en la base de datos.

Volviendo a la página de inicio, home, tenemos cuatro opciones principales: hábitos, resultados, calendario y notas, todo ello recogido en una interfaz simple que mantiene los colores del logo.

Todas estas pantallas se encuentran agrupadas por un índice **(tabs)** que ayuda a gestionar la subapertura de pantallas desde el home.

La pantalla **Hábitos** se despliega tras pulsar su botón en home. En esta aparece un listado de todas los hábitos que acepta el recomendador con la intención de que el usuario los rellene si es su primera vez haciendolo o carga unos hábitos que tenga almacenados de veces pasadas. De todas formas para una mayor robustez, cada campo cuenta con un manejador de errores que evita introducir datos no reconocibles y lo indica al usuario para que resulte más intuitivo ver donde se encuentran los errores. Tras pulsar en el botón de ejecutar, la aplicación conecta con la API en la cual se encuentra definido el recomendador y que a su vez gestiona la base de datos. De esta forma los datos de hábitos se almacenaran en la base de datos de Firestore y a su vez se ejecutara el recomendador, lo que devuelve una lista de recomondaciones que también se almacenaán y un valor ORI que representa el indice de riesgo de obesidad que se puede inducir de dichos hábitos.

La pantalla **Resultados**, se despliega tras pulsar el botón "Resultados" en home. En esta pestaña aparece una gráfica similar a un velocímetro que indicará de forma visual los resultados ORI obtenidos por el recomendador. Desde aquí el usuario podra visualizar sus resultados y su lista de recomendaciones producida para poder así empezar una nueva rutina, o gestionar lo máximo posible para obtener la meta. El objetivo idílico sería ir modificando los hábitos según indiquen las recomendaciones hasta que no quede ninguna recomendación. En ese caso si el ORI sigue siendo

más alto de lo deseado, se puede volver a llamar al recomendador, introducir tus nuevos hábitos y solicitar nuevas recomendaciones. Desde esta pantalla "Resultados" además de poder ir marcando las recomendaciones completadas, podrás ejecutar la parte del recomendador que actúa como evaluador para ver el progreso. Además existe un apartado de "Historial" en el cual se pueden ver las últimas modificaciones en los hábitos para de alguna forma incitar al prograso y permitir al usuario ver sus avances. Volviendo a la parte del recomendador, desde la aplicación se empaquetan los datos en un fichero JSON el cual se envía a la API a través de una petición HTTP POST desarrollada en la propia API en Spring Boot.

La pantalla **Calendario**, muestra un calendario simple con el cual puedes hacer funciones propias de un calendario, siendo de las más útiles la de poder incluir notas que se guardarán en la base de datos.

La pantalla de **Notas**, es una gran funcionalidad añadida. En esta pantalla hay un completo gestor de notas con el cual el usuario puede crea, modificar y borrar notas a su prefencia. Estas notas las puede ordendar alfabeticamente por título o por orden de creación, modificación, etc. Con esta funcionalidad el usuario no necesita contar con otra aplicación de gestor de ficheros. Con VitHabitus , no solo obtiene las recomendaciones si no que también puede almacenar rutinas, listas de compra, cambios a realizar en hábitos de forma prograsiva,etc, y todo desde la misma aplicación.

Todas estas ventanas se encuentran vinculadas entre si, por lo que existe un panel inferior que permite una rápida navegación entre ellas sin necesidad de volver a la ventana home, con el objetivo de simplificar las conexiones.

Todo esto desencadena en la existencia de una arquitectura modular y con separación de responsabilidades que permite una gran escalabilidad en caso de querer realizar ampliaciones o extensiones de funciones, como se explica en el apartado CONCLUSIONES Y TRABAJO FUTURO al final de la memoria.

2.3. Desarrollo aplicación móvil

La aplicación ha sido desarrollada a través del editor de código Visual Studio Code, junto con el framework de React Native debido a que es una gran apuesta a la hora de realizar desarrollo de aplicaciones nativas gracias a su simplicidad y su alta compatibilidad de código único válido en multiplataformas. Además se ha usado el entorno Expo el cual es de gran ayuda en la depuración y puesta a punto de la aplicación. Gracias a Expo Go en móviles, con gran facilidad se puede probar la aplicación en el dispositivo en directo, con tan solo escanear el código qr generado por expo.(IBM (2025)).

Por otro lado los lenguajes más empleados en la elaboración de la aplicación han sido JavaScript para la creación de las pantallas, componentes, etc; y Java para la parte de la API en Spring Boot con el recomendador el cual también se encuentra programado en java.

En esta sección de desarrollo de la aplicación móvil se describe en mayor profun-

didad el funcionamiento de la applicación pero ya no a nivel de flujo si no a nivel de implementación de código.

Empezamos por la organización del proyecto. La raiz del proyecto app/ constituye el núcleo de la aplicación y en su interior se encuentran difernciadas las distintas pantallas principales y sus componentes reutilizables, así como dato, configuraciones, gestor de navegación...

Como deciamos antes, En primer nivel tenemos la pantalla index.tsx que en React Native es esencial para la indexación de pantallas en la navegación. Login.txs y register.tsx conforman las pantallas de acceso antes de autenticar el usuario. Anidado encontramos el (tabs)/ carpeta la cual incluye las subsecciones de la pantalla home.tsx, página de inicio. Dentro de (tabs)/ encontramos haits.tsx, parte fundamental y núcleo funcional de la aplicación, resultados.tsx, notes.tsx y calendar.tsx. Todas estas páginas responden a un sistema de navegación basado en pestañas (tabs) que facilita al usuario desplazarse.

Por otra parte tenemos los elementos paralelos como los componentes los cuales muchos de ellos son aplicados en las paginas descritas antes. En app/components/ se divin según su potencial uso componentes como casillas, botones con su determinadoestilo, y otros elementos. También hay elementos visuales que conforman la interfaz de usario, la idea es encapsulamiento y modulación para mantener un código más limpio.

Finalmente en lo que respecta a la organización, tenemos la carpeta data/ que recoge todos los archivos de configuración para formularios u otros procesos, como archivos json y datos esenciales. Tenemos también la carpeta services/ que recoge todos los fichero s necesarios para la implementación de la APi y de la Firebase; firebaseCOnfig.ts con su configuración de conexión. VerifyApi.ts para su verificación de conexión con la APi en este caso.

No hay que olvidar los ficheros, app.json, eas.json y package.json. El app.json es el archivo de configuración de expo. Basicamente, recoge el nombre de la aplicación, su logo, requisitos y permisos para que se pueda ejecutar en multiplataforma correctamente. El eas.json, EAS(Expo Application Services) que contiene la información esencial para la realización de builds (app.apk) a través de EXPO, que sea descargable y ejecutable (la aplicación en formato físico sin necesidad de soporte). Por último el package.json, uno de los más importantes, dado que define las versiones que utilza la aplicación. En caso de no saber que dependencias instalar, e nel package se indican aquellas que son necesarias.

Por otro lado, uno de los elementos más característicos de React Native es el React Navigation que permite realizar las transiciones y navegaciones entre pantallas de forma manual lo que aporta una gran personalización a la aplicación. En el caso de VitHabitus, se hace uso de Expo Router, una herramienta que en este caso genera las rutas de navegación de forma automática basandose en la estrcutura de los archivos del proyecto.

De esta forma, existen dos archivos layouts uno para app/ y otro para (tabs)/ dado que así cada uno de ellos define el comportamiento y apariencia de las pantallas. (Son como submenus dentro de las subpantallas). De esta forma cada pantalla tiene

su ruta bien definida y es más fácil identificar problemas a la hora del testing además de ser más sencillo el añadir nuevas pantallas y secciones.

2.3.1. Componentes reutilizables

Una de las partes más importantes en el desarrollo de una aplicación es la modularidad y el encapsulamiento. Son recursos necesarios para que el código sea lo más legible posible y lo menos repetitivo dado que muchas veces hay fragmentos de código que se repiten en diferentes lugares y que se pueden abstraer. Ese es el caso de las componentes que vamos a ver a continuación.

- Entre los componentes más reutilizados se encuentra el `PickerComponent.tsx` basicamente un selector desplegable que permite a l usuario poder elegir entre diferentes opciones predefinidas. Es uno de los componentes más importantes al ser usado tanto en la creación del formulario para introducir los hábitos y datos en general. La configuración se bada en estructuras de datos externas por lo que es más fácilmente implementable.

- Parecido a el picker encontramos el **AcordeonHabtis.tsx** usado principalmente en habits.tsx, sirve como un contenedor con la capacidad de plegar y desplegar las secciones, ahorrando mucho espacio visual a la hora de navegar por la pantalla. Cada una de estas secciones se coonstruyen a traves de data/-formHabitsStructure.ts que otorga gran flexibilidad al proceso de creación del formulario de hábitos. Es fácilmente modificable añadiendo nuevas secciones en dichgo fichero, sin modificar el componente.

- Siguiendo con la parte de texto, destaca el **FormInput.tsx**, un fichero que se encarga de definir un campo de texto sobre el que poder escribir, se caracteriza por ser modificable con la aplicación de estilos propios y por tener un gran manejo de errores. También header.tsx usado en el login o e HeaderHome.tsxel cual es menos aplicable a otras páginas, ya que representa u header bastante especifico.

- Por último a resaltar, a pesar de haber muchos más componentes estos eran los principales. Pero otros componentes visaules como el **Loader.tsx** el cual es muy útil para gestionar momentos de espera entre navegación de pantallas, cargado y guardado de datos, confirmaciones,etc. El loader.tsx muestra una animación de carga que es muy util y reutilizable en cualquier lugar. De la misma forma el **CFOmedidor.tsx**, el cual es una representación gráfica de un medidor para indicar el valor del ORI (Obesity Risk Index) de forma clara y visual.

En general, el hecho de escribir un programa modular facilita a la hora de modificar, añadir o eliminar elementos.

2.3.2. Validación del formulario de Hábitos

La validación del formulario es esencial para el correcto funcionamiento de la aplicación VitHabitus. La calidad de los datos ha de comprobarse para evitar tener problemas posteriores con el recomendador por valores input no reconocibles. Es por eso que para el desarrollo de esta parte es necesario ser muy minucioso e incorporar un sistema de validación robusto y fiable.

Se basa en una combinación de el fichero `ValidateHabits.tsx`, el cual evalua la validez de cada campo comprobando que cada uno está dentro de los dominios permitidos. No obstante, además de este fichero es necesario el `formHabtisStercuture.ts` que mencionamos antes, ya que contiene la definición y estructura concreta de cada una de las secciones y campos. Cuando el usuario introduce un valor, la aplicación evalua en tiempo real ese valor y emite una corrección. Si es válido se acepta pero en caso contrario reproduceun mensaje orientativo para solucionar el error.

Cuando todos los campos son correctos y se puede ejecutar el formulario, se construye un objeto JSON enviable a través de la API mediante una peticion HTTP POST.

2.4. Base de datos

Para la gestión de los datos de la aplicación, VitHabitus utilizada la base de datos NoSql de Firestore, parte de Firbase. Se trata de una base de datos organizada por colecciones y documentos que al formar parte del dominio de Firebase gestiona los datos con gran rendimiento en aplicaciones móviles multiplataforma. Siguiendo con la estructura, para cada usuario se le ha asignado un identificador único que se usa como una llave principal y distintos campos que conforman las subsecciones de cada bloque de datos.

En el caso de los usuarios, la estructura es:

- **Identificador único**. Cada usuario tiene un id único asociado.
- **Colección de hábitos**. Recoge toda la información de los formularios y sus respectivas recomendaciones generadas.
- **Colección de notas**. Son las notas asociadas al usuario. Cuya estrucutra se especifica después.
- **Campo apellido**. Apellido incluido al crear una cuenta de usuario, es modificable.
- **Campo created-at**. Momento de creación del usuario.
- **Campo email**. Correo electrónico introducido al crear una cuenta, es modificable.
- **Campo nombre**. Nombre introducido al crea la cuenta, es modificable.

- **Campo país**. País de nacimiento.
- **Campo teléfono móvil**. Teléfono introducido como posible contaco (opcional), es modificable.
- **Campo imagen**. Foto de perfil (opcional) introducida para personalizar la estática del perfil en la app.

Estructura de la subcolección hábitos:

- **formulario**. Recoge todas las variable correspondientes a los hábitos del recomendador, como se puede ver en la sección "Api Rest- Recomendador Hábitos" en la memoria. Cuenta también con un campo created-at que permita llevar un historial de creación.
- **Resultados**. Resultados de la recomendación. Lista de modificaciones de las variables de hábitos recomendadas. Cuenta con un created-at para poder llevar contabilidad del momento de creación (historial).

Según la configuración, un usuario puede tener un único formulario en activo, un solo conjunto de datos que puede cargar en cualquier momento. No obstante puede tener infinitas recomendaciones y evaluaciones de ese formulario. Por tanto el campo formulario es único pero el de Resultados se gestiona según su identificador y su variable created-at para ordenarlo.

Estructura de la colección de notas:

- **Identificador único**. Cada nota tiene un identificador único asociado que las distingue de las demás.
- **User-id** Identificador del usuario al que está vinculado la nota, al tratarse de una subcolección del usuario.
- **Título de la nota**. Título explicativo de la nota.
- **Contenido**. Texto completo que conforma el cuerpo principal de la nota.
- **Created-at y updated-at**. Campos para llevar la gestión de la creación y actualización de alguno de los campos título o contenido.

Para la protección de toda esta información se aplica Firebase Security Rules, que ayuda a el encapsulamiento de los datos, permitiendo que únicamente el usuario concreto pueda acceder a sus propios datos.

En un principio del desarrollo de la aplicación móvil, esta se conectaba directamente con la base de datos. No obstante, tras la creación e implementación de la API todas estas comunicaciones son mediadas a través de la conexión api. Esta solicita al usuario su token de identificador y se lo envía directamente a la Firebase la cual inicia en caso de encontrarse apagada.

Una de las ventajas de Firestore es que cuenta con una gestión automática de una caché local que permite el acceso offline por parte de la aplicación a diferentes contenidos de información.

2.5. Autenticación Usuarios

El sistema de autenticación está basado en el uso del correo electrónico y de la contraseña junto ocn la integración directa de Firebase Authentication.

Firebase Authentication proporciona diferentes funciones siendo las mas importantes, la asignación de un identificador único UID y un control de acceso por fecha y tiempo transcurrido en la app. Además proporciona la posibilidad de establecer la opción de verificar una dirección de correo electrónico para evitar creación masivas de cuentas de usuario falsas con mensajes al correo o via sms. Notificaciones varias e inicio de sesión a través de google, facebook, githug,etc; y la funcionalidad de restablecer la contraseña en caso de extraviarla.

Estas funcionalidades s ehan implementado en las pantallas `login.tsx` y `register.tsx`. Durante el flujo de procesamiento en estas paginas Firebase válida los campos rellenados y comprueba que por ejemplo la contraseña sea lo suficientemente segura.

La verificación del correo se establece en el fichero `EmailVerification.tsx` que se encarga de comprobar el correcto funcionamiento de la cuenta de correo y además verifica que el usuario tenga acceso a esta antes de crearla. Para evitar suplantaciones de identidad.

Tras la creación de un usuario y al iniciar sesión, Firebase emite un token de sesión que se mantiene activo hasta que el usuario cierra su sesión. Este token es muy importante para la sincronización y privacidad de la información entre la aplicación y la api y base de datos.

2.6. Api Rest

El sistema de recomendación utilizado en VitHabitus ha sido encapsulado en una API RESTful desarrollada como un servicio independiente gracias al uso del framework Spring Boot. El objetivo de esta API (Interfaz de Programación de Aplicaciones) es implementar el algoritmo del recomendador de la mejor forma posible para la aplicación móvil.(web official Spring Boot (2025b)). Tras investigar, se llegó a la conclusión de que implementar el algoritmo directamente en la app, provocaria un aumento excesivo del peso y complejidad de esta, provocando ralentizaciones y hasta errores inesperados. Es por ello que finalmente se decidió la realización de una API que posteriormente se encapsularía en un Docker para desplegarla en un servidor propio o en una nube, permitiendo una conexión constante y fluida con la aplicación sin ralentizarla.

La API cuenta con el patrón típico de capas de Spring Boot, con separación de controladores de controladores de entrada, de los modelos que representan las estructuras de datos y de también los servicios que se encargan de procesar la lógica de negocio. De esta manera, el @RestController, también conocido como controlador principal, expone los endpoint como el de tipo POST que recibe el conjunto de hábitos del usuario en formato JSON.

Dada la entrada de datos, se transforman los campos necesarios para poder introducirlos correctamente al recomendador y que este pueda funcionar correctamente. Para ello tenemos el @Service que contiene toda la lógica relacionada con la ejecución del recomendador. Con esto se consigue desacoplar el algoritmo del controlador HTTP.

Con respecto al recomendador, se ha modificado de cierta manera para poder integrarlo lo mejor posible, eliminando toda la capa de interfaz gráfica que tenía. Además hemos divido el reomendador en dos partes: evaluador de hábitos que genera el valor del ORI y busqueda de recomendaciones que realiza todo el proceso para obtener las recomendaciones finales más óptimas.

2.6.1. Recomendador de Hábitos

Entramos en el núcleo funcional de la aplicación VitHabitus, el recomendador de hábitos saludables desarrollado por Fan Ye, Daniel Martínez y recogido en el Trabajo de Fin de Grado: ‘‘`Recomendador de Hábitos para reducir el riesgo de padecer Sobrepeso y Obesidad`’’ de la Universidad Complutense de Madrid. (Ye Fan (2025)).

Este algoritmo se fundamenta en la generación de un algoritmo evolutivo, basado en simular procesos que replican la “selección natural” para buscar aquellas combinaciones de hábitos más oóptimas. Este proceso tiene en cuenta una serie de variables inmutables, como la edad, sexo y varaibles modificables como el ejercicio, enfermedades, alimentación, etc. De esta forma, a partir de los hábitos introducidos, el sistema genera una población inicial de soluciones que se representan como cromosomas. Estos son sometidos a varias iteraciones del ciclo evolutivo con procesos como mutación, cruce, selección, y finalmente la evaluación, a través de una función fitness que representa la calidad de los hábitos en relación con los modelos entrenados.

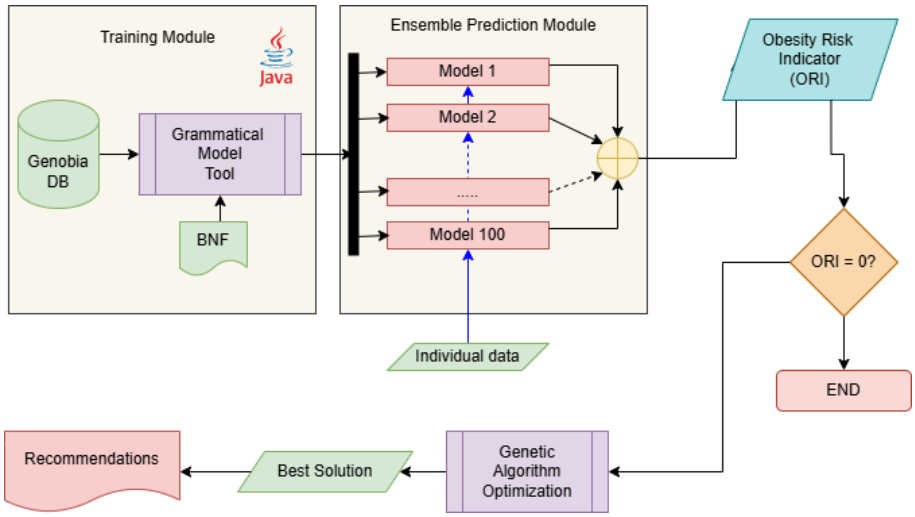

Figura 2.2: Arquitectura general del sistema de recomendación

Se trata de la unificación de dos esquemas presentes en el documento “An Evolu-

tionary Habit Recommender System to Reduce the Risk of Overweight and Obesity".

Para que el recomendador funcione correctamente, uno de los pasos más importantes a realizar, es el uso de los mejores modelos lógicos posibles, los cuales son los encargados de entrenar el algoritmo. Para ello, el recomendador cuenta con 100 modelos generados a partir de los datos del proyecto **GenObiA**, una iniciativa multidisciplinar centrada en el análisis del riesgo de obesidad mediante información socio-demográfica, hábitos de la vida y genética de más de 1000 personas.

El sistema se encarga de seleccionar de entre los modelos aquellos que tengan una mayor precisión para ser utilizados en la función fitness de evaluación. Se comparan las distintas soluciones y se selecciona la más prometedora.

Es importante también especificar las distintas variables que son tratadas a lo largo del código del recomendador, las cuales también son declaradas como campos únicos en la colección de formularios de la Base de Datos de Firebase.

Listado de variables:

1. **Sexo**: si la persona es hombre o mujer al nacer.
2. **Edad**.
3. **Población**: número de personas que viven en la localidad del individuo. Los rangos son: menos de 2.500 habitantes, entre 2.500 y 20.000, entre 20.000 y 50.000 y más de 50.000.
4. **Nivel educativo**: puede ser sin estudios, educación primaria, secundaria o universitaria.
5. **Ingresos**: situación económica del individuo. Los rangos son: menos de 1.000€ al mes, entre 1.000€ y 2.000€ al mes y más de 2.000€ al mes.
6. **Profesión**: tipo de trabajo del individuo, seleccionado de una lista de 16 ocupaciones.
7. **Estrés**: si la persona se considera estresada o no.
8. **Sueño (8 horas)**: si duerme al menos 8 horas diarias.
9. **Bebidas espirituosas**: si consume bebidas alcohólicas fuertes (como licores).
10. **Copas de licor**. Número de copas de licor a la semana.
11. **Vino o cerveza**: si consume vino o cerveza.
12. **Copas de cerveza**. Número de copas de cerveza a la semana.
13. **Copas de vino tinto**. Número de copas de vino tinto a la semana.
14. **Copas de vino blanco**. Número de copas de vino blanco a la semana.
15. **Copas de vino rosado**. Número de copas de vino rosado a la semana.

16. **Tabaco**: si fuma cigarrillos actualmente.
17. **Número de cigarrillo**. Número de cigarrillos por semana.
18. **Pipa**: número de pipas o cargas de pipa que fuma al día.
19. **Puros**: número de puros que fuma al día.
20. **Años como exfumador**: años desde que dejó de fumar.
21. **Exfumador (desconocido)**: si dejó de fumar, pero no sabe desde hace cuánto.
22. **Cáncer**: si padece algún tipo de cáncer.
23. **Cáncer de mama**.
24. **Cáncer de colon**.
25. **Cáncer de próstata**.
26. **Cáncer de pulmón**.
27. **Otros tipos de cáncer**.
28. **Infarto de miocardio**.
29. **Angina de pecho**.
30. **Insuficiencia cardíaca**.
31. **Diabetes tipo 2**.
32. **Síndrome metabólico**.
33. **Apnea del sueño**.
34. **Asma**.
35. **EPOC**: enfermedad pulmonar obstructiva crónica.
36. **Consumo de aceite de oliva**: cucharadas al día.
37. **Consumo de verduras**: porciones al día. La guarnición cuenta como media porción.
38. **Consumo de frutas**: piezas de fruta al día, incluyendo zumos naturales.
39. **Consumo de carne roja**: porciones de carne de vacuno o cerdo al día (hamburguesas, embutidos, fiambres). Cada porción equivale a 100-150g.
40. **Consumo de mantequilla o nata**: porciones al día. Cada porción equivale a unos 120g.
41. **Consumo de refrescos**: vasos de bebidas azucaradas o carbonatadas al día.

42. **Consumo de legumbres**: porciones por semana. Cada porción equivale a 150g.
43. **Consumo de pescado o marisco**: porciones por semana. Cada porción equivale a 100-150g o 4-5 piezas de marisco.
44. **Consumo de bollería industrial**: veces por semana.
45. **Consumo de frutos secos**: porciones por semana. Cada porción equivale a 30g.
46. **Consumo de carne blanca**: si prefiere carnes como pollo, pavo o conejo en lugar de carne roja.
47. **Consumo de sofritos**: veces que consume sofritos como acompañamiento de pasta, arroz u otros platos por semana.
48. **Consumo de lácteos**: veces que consume productos lácteos al día.
49. **Lácteos desnatados**: si los productos lácteos consumidos son desnatados o no.
50. **EIMS**: minutos semanales de ejercicio intenso.
51. **EMMS**: minutos semanales de ejercicio moderado.
52. **ECMS**: minutos semanales caminando.
53. **Minutos sentado**: tiempo medio que ha pasado sentado durante la última semana.

2.7. Tecnologías Utilizadas

Las principales tecnologías empleadas en este proyecto son las siguiente:

- **Firebase**: Plataforma elegida debido a su amplia gama de servicios y gran compatibilidad con aplicaciones móviles. Se usan principalmente los servicios de Atutenticación de usuarios para la gestión de los usuarios y Firestore para el almacenamiento de datos en la nube.(Firebase (2025))
- **React Native**: Framework empleado para el desarrollo de la parte frontend, interfaz de la aplicación VitHabitus. Destaca por la facilidad de crear aplicaciones tanto para Andorid como para iOS en un mismo proyecto. Además de ofrecer similitudes con estilos como css y ser un framework sencillo e intuitivo para aquellos desarrolladores primerizos. Esto se debe gracias a sus librerias que facilitan en gran medida las conexiones backend como a una API o Base de Datos. React native, Calendars, compatibilidad con Expo, etc. (web official React Native (2025))

- **Expo**: Entorno de desarrollo utilizado para aplicaciones móviles basadas en React Native. Expo proporciona una experiencia completa en la craeción prueba y despliegue de una aplicación en diferentes dispositivos móviles android y iOS, gacias a sus numerosas herramientas que resultan de gran utilidad. Aplicaciones como Expo GO que permiten previsualizar la app en tiempo real desde un dispositivo físico sin compilar el proyecto. Eas de Expo(EXPO (2025a)) que permite la generación de binarios, builds listos para la instalación o publicación. Librerias como Expo router que implementan de forma sencilla la navegación entre pantallas basandose en la estructura de archvios y haciendo el proyecto más escalable. Entre otras.(EXPO (2025b))

- **NodeJS**: Es un entorno de ejecución para JavaScript. Se utilzia en el proyecto como base para el entorno de desarrollo de React native, sinedo su función principal permitir la ejecución de herramientas y scripts que son necesarias para que la aplicación funcione. Tiene además herramientas como Express para la cración de API-Rest.(web officialNodeJS (2025))

- **NPM (Node Package Manager**: Es un gestor de paquetes asociado al escosistema de JavaScript. NPM es imprescindible en el funcionamiento de la app, siendo el comando "npm install" necesario siempre para confirmar la instalación de todas las dependencias correctas y necesarias. Tiene scripts definidos muy útiles en el archivo package.json que automatizan varios procesos.

- **TypeScript**: Es preferible su uso sobre JavaScript debido a su tipado estático, lo que garantiza mayor seguridad y claridad en el código. Especialmente para desarrolladores acostumbrados a lenguajes tipados.

- **Github y git**: Para el control de versiones y modificaciones del código se ha utilizado la herramienta git junto con la interzad de github con el uso de un repositorio.

- **Spring Boot Suite**: Se trata de un entorno de desarrollo integrado basado en eclipse que se utiliza en la creación de la API REST. Es muy utilizado para construir servicios web de forma rápida, modular y con las configuraciones necesarias. Tiene gran valor gracias a su compatibilidad con Maven(gestión de dependencias, archivo pom.xml) y Java, lenguaje en el que está implementado e recomendador.(web official Spring Boot (2025a)).

Capítulo 3

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas generales de trabajo futuro.

Teniendo encuenta el proceso completo, el desarrollo de VitHabitus ha supuesto un reto técnico e interesante al ser mi presentación al mundo de las aplicaciones móviles.

Al final se ha conseguido cumplir gran parte de los objetivos propuestos, con el desarrollo del frontend y el backend con la API y la base de datos de Firebase. Integración del algoritmo de recomendación de hábitos y un correcto flujo de funcionamiento entre todos los elementos.

Queda como resultado una aplicación que supone de gran utilidad para muchas personas y que demuestra una vez más, el increible potencial que tiene la inteligencia artificial aplicada a la salud. Además demuestra como el desarrollo de aplicaciones móviles puede actuar como un canal eficaz para trasladar tecnologías complejas a la vida cotidiana de las personas, gracias a una interfaz intuitiva y de fácil uso a través de un teléfono móvil.

3.0.1. Trabajo Futuro

Independientemente del resultado. Siempre hay posibilidad de mejora y avances. De esta forma, hay algunos ideas que surgieron durante la realización del proyecto. Una de las más importantes sería la ampliación del formulario de hábitos, permitiendo ser aún más flexible con más hábitos que representen relación con la obesidad.

- Implementación de conectividad con dispositivos wearables que contienen sensores que podrían calcular datos como pulsaciones, horas de sueño, calorias quemadas, que pueden ser de gran utilidad para el recomendador.

- Publicar la aplicación de forma oficial en google play y App Store, lo que supondría cumplir con los estandares solicitados y mantener la aplicación actualizada

- Traducir la aplicación a varios idiomas o al menos al inglés para tener futuro en el mercado internacional.
- Por otro lado en la parte visual, hay ciertos partes que se podrían mejorar para obtener un acabado más profesional.

.

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter 3.

Taking into account the entire process, the development of VitHabitus has been a technical and a rewarding challenge, as it was my first step into the world of mobile applications.

In the end, most of the initial objectives were successfully achieved, including the development of both the frontend and backend, the implementation of the API and the integration with Firebase as the database. The habit recommendation algorithm was also successfully incorporated, resulting in a coherent and functional flow between all components.

The final product is an application that can be highly useful for many users and it clearly showcases the incredible potential of artificial intelligence when applied to health. Also, it demonstrates how mobile app development can serve as an effective channel for bringing complex technologies into people's everyday lives, thanks to an intuitive and easy-to-use interface accessible from any smartphone

Future Work

Regardless of the current outcome, there is always room for improvement and growth. Some ideas emerged throughout the development process and some of the mos relevant are this ones:

- Implement connectivity with wearable devices equipped with sensors that could track valuable data such as heart rate, sleep duration, calories burned... variables that could been applied to the recommendation system.
- Officially publish the app on Google Play and Apple Store, which involve setting the required standards and ensuring continuous updates.
- Translating the app into multiple languages, or at least into English, to increase its potential to reach the international market.
- In the visual aspects, improving some aspects of the user interface to achieve more polished and visual finish.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

Axarnet. Plataformas de desarrollo de aplicaciones móviles. 2025.

EXPO. Expo application services. 2025a.

EXPO. Guides: Overview. 2025b.

Firebase, G. Documentación firebase. 2025.

Francesc Alòs, A. P.-R. Uso de wearables y aplicaciones móviles. 2021.

IBM. ¿qué es el desarrollo de aplicaciones móviles? 2025.

Ismael San Mauro Martín1, M. G. F. y. L. C. Y., 2. Aplicaciones móviles en nutrición, dietética y hábitos saludables. 2014.

web officialNodeJS. Manual de nodejs. 2025.

web official React Native. Manual de react native. 2025.

web official Spring Boot. Manual de spring boot. 2025a.

web official Spring Boot. Manual de spring boot rest. 2025b.

World Health Organization, W. Obesity and overweight. 2024.

Ye Fan, M. D. Recomendador de hábitos para reducir el riesgo de padecer sobrepeso y obesidad. 2025.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes