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*
Calificación: *8.5*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

16 de junio 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 que provocan patologías como la obesidad y el exceso de grasa corporal.

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. y reducir el riesgo de padecer sobrepeso y obesidad.

La aplicación ofrece una interfaz sencilla y estructurada por secciones, que permite al usuario introducir sus hábitos actuales, recibir sobre ellos recomendaciones generadas por un algoritmo integrado en un sistema recomendador y visualizar su evolución. Para garantizar una gestión eficiente y segura de los datos, se ha integrado Firebase como backend principal encargado del funcionamiento, autenticación de usuarios y sincronización en tiempo real.

Como elemento central en la estructura de la aplicación, se encuentra una API REST, desarrollada en Spring Boot, que encapsula todas las funcionalidades del sistema recomendador. Este sistema, que da soporte a la aplicación, utiliza un algoritmo genético, una técnica de inteligencia artificial basada en la evolución natural, para buscar la mejor combinación de hábitos. A partir de los datos introducidos por el usuario, el algoritmo genera una población inicial de soluciones(cromosomas), cada una representando posibles combinaciones de los hábitos modificados. Estas soluciones se evalúan mediante una función de aptitud (fitness) que utiliza un conjunto 100 modelos predictivos previamente entrenados mediante técnicas de evolución gramatical (Grammatical Evolution). El resultado final es la selección de la mejor combinación de hábitos saludables cuya evaluación determina un índice de riesgo de Obesidad (ORI, Obesity Risk, Indicator) que resume el estado de la salud estimada del usuario.

Palabras clave

Obesidad, hábitos saludables, recomendador, aplicación móvil, código, usuario, 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, which cause conditions such as obesity and excess body fat.

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

The application offers a simple interface structured in sections, allowing the user to enter their current habits, receive recommendations generated by an algorithm integrated into a recommender system and visualize their progress. To ensure efficient and secure data management, Firebase has been integrated as the main backend, responsible for functionality, user authentication, and real-time synchronization.

As the main element in the structure of the application, there is a REST API developed in Spring Boot that encapsulates all the functionalities of the recommender system. This system, which supports the application, uses a genetic algorithm, a type of artificial intelligence technique inspired by natural evolution to search for the best combination of habits. Based on the data entered by the user, the algorithm generates an initial population of solutions (chromosomes), each one representing possible combinations of modified habits. These solutions are evaluated using a fitness function that relies on a set of 100 predictive models previously trained through grammatical evolution techniques. The final result is the selection of the best combination of healthy habits, whose evaluation determines an Obesity Risk Indicator (ORI) that summarizes the user's estimated health status.

Keywords

Obesity, healthy habits, recommender, mobile application, code, user, 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 modificar, 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 auto evaluarse 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 otorgue a los usuarios la capacidad de utilizar un algoritmo evolutivo para mejorar sus hábitos semanales de la forma más personalizable 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.

Los objetivos del proyecto son:

- Desarrollar la estructura de la aplicación móvil con el framework React Native y el entorno de desarrollo Expo. Dos herramientas esenciales en la creación y depuración de aplicaciones simples multiplataformas.
- Desarrollar una interfaz intuitiva y simple que facilite la introducción y la visualización de hábitos por parte del usuario.
- Desarrollar un sistema de autenticación robusto mediante Firebase.
- Implementar una base de datos en Firestore para la gestión de información de los usuarios como las notas, hábitos y recomendaciones.
- Desarrollar una API REST en Spring Boot que integre el sistema recomendador.
- Proporcionar funciones adicionales como historial de evolución, gestión de notas y calendario de hábitos.
- 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 estructurado en distintas etapas cada una centrada en un componente esencial en el desarrollo de la aplicación:

1.3.1. Investigación preliminar

Un estudio de diferentes aplicaciones móviles existentes en Apple Store y Play Store relacionadas con la nutrición, salud y gestión de hábitos o rutinas. Identificación de puntos fuertes, limitaciones, opciones de pago y oportunidades de mejora para aplicar a nuestra aplicación. Todo esto se encuentra desarrollado en el capítulo Estado Del Arte que se verá más adelante.

1.3.2. Estructura y entorno de desarrollo

Con el fin de obtener el mejor resultado posible, se estudió sobre el mejor framework que recogiese las condiciones exigidas. Entre las opciones se encontraron Flutter y React Native siendo este último el finalmente escogido gracias a su compatibilidad con Expo que presenta herramientas interesantes en el desarrollo y depuración de código para aplicaciones móviles.(Axarnet (2025)).

Por otro lado se diseñó la estructura general de la app. Componentes reutilizados que podrían modularizarse y pantallas por las que el usuario puede navegar en la aplicación.

1.3.3. Autenticación

- Estudio sobre la plataforma que mejor se adapte al objetivo. Concretamente Firebase de Google que proporciona todos los servicios básicos para una app móvil y permite gran escalabilidad a futuro.
- 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 de los usuarios y de la seguridad.

1.3.4. Base de Datos

- Al igual que en la autenticación, uso de la cuenta de Firebase para la creación de la base de datos con Firestore. Siendo esta una base de datos "noSql".
- Implementación de la estructura de las colecciones y documentos que se almacenarán.

1.3.5. Backend

- Construcción de una API REST con Spring Boot, con una integración del sistema recomendador y la definición de endpoints para la comunicación con la app.
- Implementación del código del recomendador adaptandolo a los requisitos de la API y la conexión con la base de datos buscando una vinculación correcta y segura con la aplicación móvil.

1.3.6. Sincronización

- Garantizar correcto funcionamiento constante de Firebase y de la API desplegando esta última en un servidor. Análisis sobre posibles cambios que favo-

rezcan la robustez de la aplicación, como el uso de una base de datos propia MySQL.

1.3.7. Interfaz intuitiva, frontend

- Creación y reutilización de elementos y componentes visuales open source, para mejorar la interfaz de la aplicación.
- Uso de Figma para desarrollar componentes visuales aplicables.

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 Ingeniería 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 **Daniel Martínez** y Ye Fan (2025), 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 orientadas en mejorar la salud y reducir la obesidad. Para evaluar la calidad de cada posible combinación de hábitos, se utiliza la función de aptitud (fitness) alimentada con 100 modelos predictivos ya entrenados construidos a partir de datos del proyecto GenObia (2025). Estos modelos permiten calcular el indicador denominado ORI (Obesity Risk Indicator), que representa el índice de riesgo de obesidad del usuario en función de sus hábitos actuales.

Finalmente se realizó una integración de este sistema recomendador en la aplicación mediante el desarrollo de la API REST en el entorno de Spring Boot Suite lo que permite una separación entre la interfaz móvil y el recomendador, facilitando su mantenimiento y escalabilidad en el futuro.

Introduction

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.

The objectives of this project are:

- 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:

Preliminary research

A study of various existing mobile applications on the Apple Store and Play Store related to nutrition, health, and habit or routine management. Identification of strengths, limitations, payment options, and opportunities for improvement to be applied to our own application. All of this is developed in the Estado Del Arte chapter, which will be presented later.

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.

Authentication

- Study 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.

Backend

- 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.

Synchronization

- 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.

Intuitive interface, fronted

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

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 Daniel Martínez and Ye Fan (2025), under the supervision of Jose Ignacio Hidalgo Pérez.

The recommender forms the functional core of the application. It is responsible for processing the habits entered by the user and consequently generating personalized recommendations aimed at improving health and reducing obesity. To evaluate the quality of each possible combination of habits, a fitness function is used, powered by 100 pre-trained predictive models built using data from the proyect GenObia (2025). These models enable the calculation of the ORI (Obesity Risk Indicator), which represents the user's obesity risk index based on their current habits.

Finally, this recommender system was integrated into the application through the development of a REST API using the Spring Boot Suite environment, allowing for a separation between the mobile interface and the recommender system, facilitating future maintenance and scalability.

Capítulo 2

Estado Del Arte

Uno de los primeros pasos para la creación de la aplicación VitHabitus ha sido analizar el mercado para poder entrar en sintonía con aquellas aplicaciones más populares. Esta investigación permite extraer ideas, modelos, interfaces, estructuras, componentes y mucha más información que facilitan enormemente el desarrollo de una aplicación competitiva para el mercado.

De esta manera, este capítulo recoge una revisión de varias de las aplicaciones más usadas relacionadas con el ámbito de la salud y de la planificación de hábitos, identificando tanto las fortalezas y debilidades de cada una, como las características comunes que resultan esenciales para el atractivo de una aplicación.

Entre las aplicaciones que se analizaron se encuentran “Habitify”, “Quitzilla”, “Way of life”, “INDYA”, “HabitNow”, “Habitica”, “Todoist”.

A la hora de evaluar las aplicaciones se han seguido ciertos criterios y parámetros con el fin de obtener una visión crítica y equilibrada. En primer lugar, se ha analizado la experiencia de usuario, valorando la facilidad de navegación, la claridad de las instrucciones y explicaciones de uso y la fluidez general del recorrido de navegación. También se ha tenido en cuenta el diseño y los efectos visuales, examinando el atractivo estético, la calidad del diseño y la presentación visual. Siendo estos factores importantes para fomentar el compromiso del usuario hacia la aplicación.

Otro aspecto a considerar ha sido la flexibilidad de cada aplicación, evaluando su capacidad para adaptarse a circunstancias cambiantes, como la modificación de objetivos o la reprogramación de actividades de un día para otro. Surge también el término “Gamificación”, básicamente consiste en examinar de que maneras la aplicación incorpora elementos de juego como recompensas, insignias o sistemas de rachas que motivan y atrapan a los usuarios a seguir consumiendo la aplicación. Se trata de un elemento interesante, que bien implementado tiene mucho éxito.

En cuanto a componentes técnicos, muchas de estas aplicaciones se encuentran desplegadas en dispositivos móviles y en ordenadores, por lo que un factor importante para la escalabilidad es la diferencia de implementación entre los diferentes entornos (aunque de momento VitHabitus tiene como único objetivo los teléfonos móviles).

Por último, se analizó la parte de comercialización. Todas estas aplicaciones que se analizarán a continuación, son de uso gratuito en una primera instancia, pero algo que también todas tienen en común, es la existencia de un plan superior al gratuito, el "plan premium", en el cual a cambio de pagar una cuota mensual o anual, se ofrecen ciertos privilegios adicionales. En términos generales, se analizó el coste de esos planes de pago y la relación calidad-precio entre servicios y coste. Además, muchas aplicaciones se apoyan en la premisa de ser aplicaciones gratuitas a pesar de que su plan gratuito cuenta con tantas restricciones que el usuario poco puede hacer en ella sin invertir en un plan superior.

2.1. Habitica

La primera de las aplicaciones a mencionar es Habitica. Se trata de una aplicación basada en la gestión de hábitos pero con un enfoque más interactivo y con juegos de rol. Mezcla la productividad con la diversión lo que lo convierte en una aplicación que combina a la perfección la tarea de gestión de hábitos y la retención del usuario.

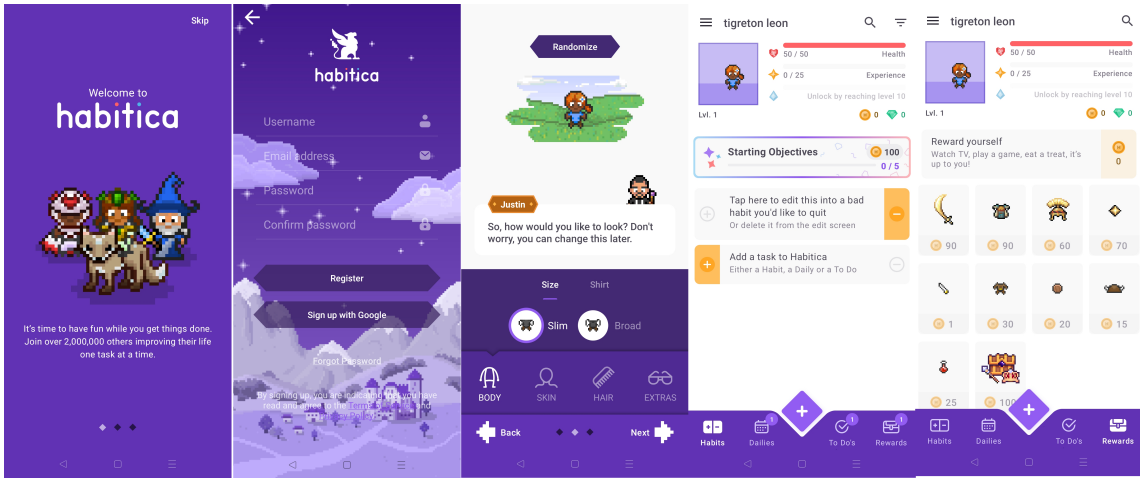

Figura 2.1: Flujo de funcionamiento de Habitica

Como aspectos a destacar, es necesaria la creación de una cuenta de usuario para poder acceder. Desde la pantalla de inicio explica de forma amena la finalidad de la aplicación para facilitar al usuario su integración con la aplicación. Una vez hecho el log-in, se presenta la pantalla de inicio en la cual se puede acceder a los distintos componentes y servicios de la aplicación. Para mayor inmersión juega con las recompensas, las tareas diarias y la personalización de personaje para que, de alguna forma, el usuario se sienta más apegado a los resultados y evolución del mismo. Es un claro ejemplo de gamificación con recompensas.

No obstante, hay ciertos aspectos que consideramos negativos sobre todo para utilizar en VitHabitus. Se trata de una aplicación sobrecargada de estímulos, tanto por los elementos llamativos como por los servicios innecesarios de una aplicación de gestión de hábitos. Eso provoca que en cierto punto el usuario pierda el objetivo principal de la misma, haciéndola ineficiente.

Por último a pesar de considerarse una aplicación gratuita, tiene la posibilidad de suscribirse mensualmente con pocos privilegios extras. Algo a considerar para la sostenibilidad de la aplicación.

2.2. Habitify

La siguiente aplicación se trata de Habitify. A diferencia de lo visto en Habitica, en Habitify existe la posibilidad de crear una cuenta como invitado, lo que permite a los usuarios utilizar la aplicación en un periodo de prueba para ver si realmente les interesa. Es un práctica interesante para alcanzar a un mayor público.

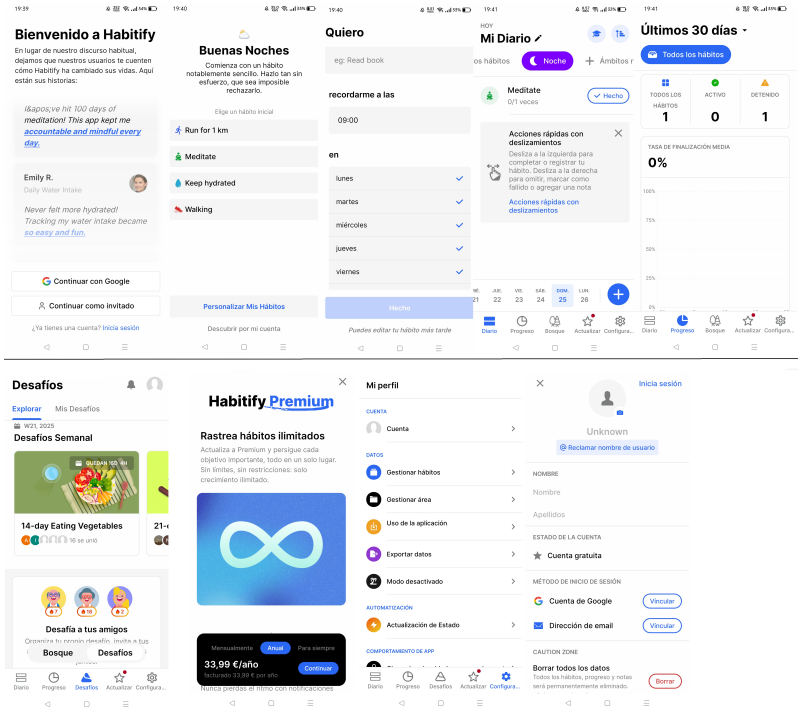

Figura 2.2: Flujo de funcionamiento de Habitify

Su estructura es bastante sencilla como se puede ver en la figura 2.2. Se trata de una pantalla de inicio en la cual se puede seleccionar el(los) hábito(s) que se desea gestionar. Permite la selección de fecha y hora y una interfaz sencilla de colores blanco y azul que integra muy bien la finalizad de la aplicación. Por último un apartado de progreso donde el usuario puede ver sus avances, algo esencial para la retención de personas.

Por último, al igual que la gran mayoría de las aplicaciones de este listado, cuenta con un plan premium y con una iniciativa de recompensas mediante desafíos que enganche al usuario a permanecer en la aplicación. Con respecto a los ajustes y el perfil, es bastante simple con las funciones necesarias para un gran funcionamiento.

2.3. HabitNow

HabitNow se trata de nuevo de una aplicación centrada en la gestión de tareas y de hábitos pero en este caso con ciertas funcionalidades extras que la convierten en una aplicación muy versátil. Carece de inicio de sesión ni creación de cuenta pero presenta autoguardado por usuario, lo que permite mantener todos los datos a pesar de no registrarse. Tiene una interfaz muy limpia y elegante, a diferencia de Habitify y HabitNow, cuenta con una gama de colores opuesta que favorece mucho a aquellos usuarios amantes de los temas oscuros. Presenta una pantalla principal sencilla y sin elementos innecesarios y una gran variedad de funciones que permiten la personalización de las tareas y notas creadas.

Una de las mejores características de HabitNow es su personalización, lo que permite a un mayor número de usuarios configurarla a su antojo.

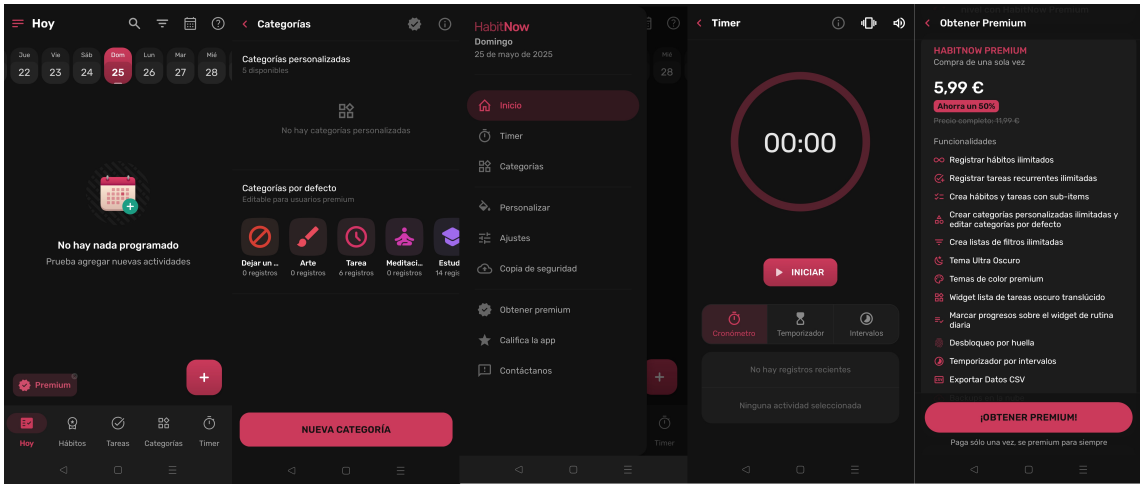

Figura 2.3: Flujo de funcionamiento de HabitNow

2.4. INDYA

Dejando a un lado las aplicaciones de gestiones de hábitos, nos encontramos con INDYA, una de las aplicaciones de hábitos y rutinas saludables más potentes y usadas en teléfonos móviles.

En este caso empezaremos con la parte más negativa y, a la vez la que garantiza su éxito. Se trata de una aplicación exclusivamente de pago. Una vez te registras, el flujo de la aplicación consiste en una serie continua de pantallas para poder ir introduciendo tus hábitos, rutinas y objetivos a conseguir. Con ello se consigue, de una forma muy amena y visual, que el usuario introduzca grandes cantidades de datos. Una vez terminado este recorrido, la aplicación gestiona los datos introducidos para buscar a un profesional apropiado para el usuario. De esta forma sirve como gestor para la búsqueda rápida de un entrenador personal al que se deberá pagar. Por tanto se trata de una aplicación de pago.

A pesar de ello, su interfaz y su forma de afrontar el reto de solicitar al usuario rellenar formularios de información es muy interesante. Supone un sistema interactivo y que sin duda se puede aplicar en la aplicación VitHabitus.

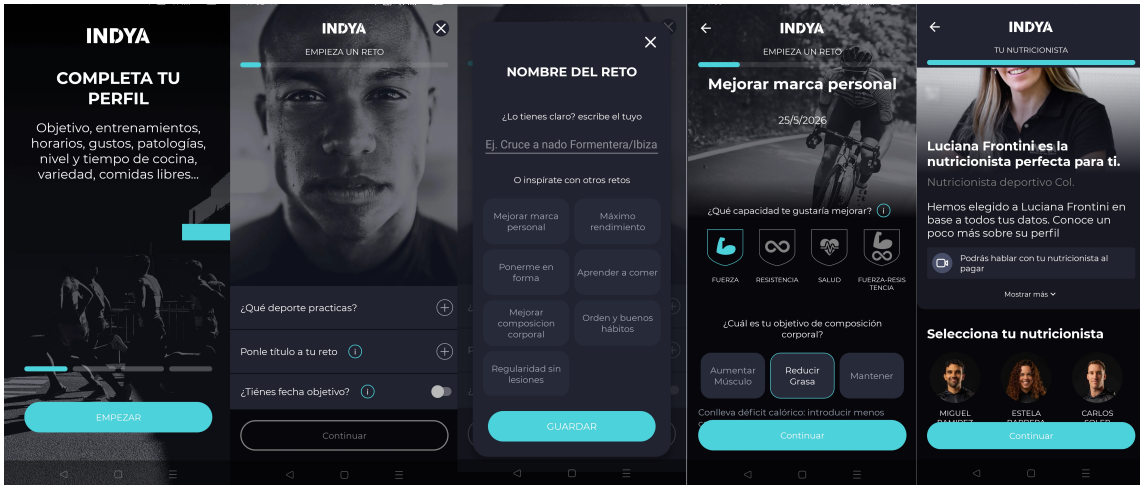

Figura 2.4: Flujo de funcionamiento de INDYA

2.5. Way of Life

Way of Life es de entro todas las aplicaciones mencionadas, la mas sencilla pero no por ello la peor. Se trata de una aplicación simple de seguimiento de hábitos, el usuario crea un hábito y lleva un registro de cuando lo cumple y cuando no. La aplicación solo se encarga de evaluarlo y mostrarlo gráficamente al usuario. (Ver en la figura 2.5).

A pesar de no contar con una interfaz muy elaborada ni con unas funcionalidades apabullantes, se trata de un referente al que se aspira en el desarrollo de nuestra aplicación de hábitos saludables. Way of life combina a la perfección la sencillez con la funcionalidad al igual que HabitNow, son dos aplicaciones que ofrecen lo que prometen cuando las descargas de una forma sencilla y visual.

2.6. Todoist

Por último y para no hacer más hincapié en aplicaciones de gestión de tareas, tenemos Todoist. Se trata de una de las aplicaciones de gestión de objetivos y tareas más relevantes en el mercado, no solo de aplicaciones móviles, si no también de ordenadores.

El motivo de la presencia de esta aplicación es la forma de lidiar con las tareas tan peculiar y diferente a los anteriores gestores. En este caso, Todoist abruma por su personalización y su cantidad de funcionalidades y servicios. No solo permite al

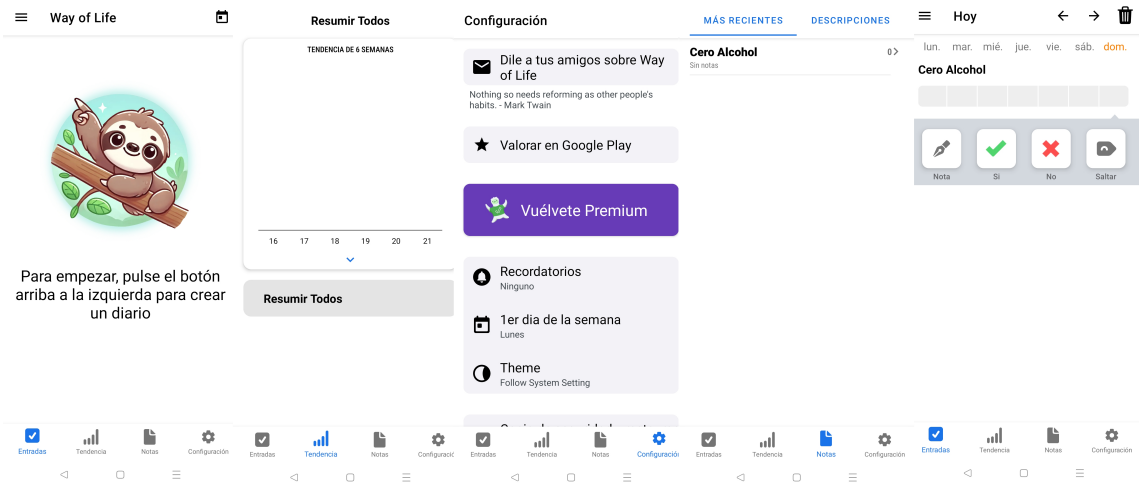

Figura 2.5: Flujo de funcionamiento de Way of Life

usuario crear notas, permite crear las plantillas de dichas notas, algo que aumenta la experiencia del usuario. Por ello Todoist es una herramienta muy potente y fácil de usar.

2.7. Conclusiones

A raíz del estudio realizado sobre diversas aplicaciones móviles disponibles en el mercado, se ha llevado a cabo un análisis centrado principalmente en herramientas de hábitos y tareas, más que en aquellas aplicaciones centradas en el control nutricional o en planes de entrenamiento para pérdida de grasa. Esto se debe a que VitHabitus, aunque tiene como objetivo final la mejora de la salud y reducción del riesgo de obesidad, el enfoque que queremos transmitir es que lo haga a través de la modificación progresiva de hábitos cotidianos y no mediante el conteo de calorías o rutinas de ejercicios.

De esta manera, las aplicaciones analizadas Habitica, Habitify, HabitNow, Way of Life, INDYA y Todoist, han aportado una visión variada a los diferentes enfoques que se pueden tomar.

Con este análisis se han identificado elementos positivos e imprescindibles para el desarrollo de VitHabitus como la necesidad de implementar una interfaz sencilla y amigable, evitando la sobrecarga de elementos y aumentando la simplicidad del uso de la aplicación. Por otro lado, se intenta evitar el exceso de funciones innecesarias y el uso de técnicas de enganche como premios, recompensas diarias, etc.; elementos que por el momento no se priorizan en el desarrollo.

Capítulo 3

VitHabitus

Este capítulo contiene una descripción completa de la investigación y el trabajo realizado en este proyecto. Explica de forma detallada la estructura establecida para el desarrollo de la aplicación, el flujo de funcionamiento y 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.

3.1. Diseño del sistema

Una vez analizado el mercado de aplicaciones y con una idea en mente sobre como afrontar la creación de la aplicación, viene el proceso de listado de todas aquellas funcionalidades y requisitos que nuestra aplicación debe cumplir. Se trata de un listado que se ha ido modificando durante el desarrollo de la práctica hasta terminar siendo el siguiente:

3.1.1. Requisitos Funcionales

3.1.1.1. Autenticación

- Registro de usuarios mediante correo electrónico y contraseña.
- Funcionalidad de inicio de sesión y crear cuenta con opción de autenticación con proveedores 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.

3.1.1.2. Perfil

Todos los datos relacionados al perfil de un usuario.

- Campos como nombre 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 recomendador cuando sea necesario.

3.1.1.3. Hábitos

Pantalla "hábitos":

- Formulario con todos los campos a rellenar para poder ejecutar el recomendado.
- 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 parámetros 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 indicar al usuario sobre el 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.

3.1.1.4. Resultados

Pantalla "Resultados", muestra una gráfica con la valoración de los hábitos de la última ejecución, con el valor ORI:

- Presencia de una gráfica que muestre el valor ORI de los hábitos del usuario hasta ahora seguido del listado de recomendaciones y la opción mediante un botón de actualizar los hábitos con nuevos valores y recalcular su ORI.

3.1.1.5. 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 máximo de 110 caracteres, contenido ilimitado, autoguardado tras escritura. Tras la creación de la nota se ordena cronológicamente según su creación. Además da la posibilidad de cambiar el orden de las notas arrastrándolas 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.

3.1.1.6. Calendario

Un calendario interactivo para poder revisar de forma sencilla todas las recomendaciones generadas y los valores ORI calculados. Estos registros, quedan vinculados a las fechas en las que se obtuvieron siendo en este apartado donde se pueden consultar.

3.1.2. 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 Security 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 permite 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

3.1.3. 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 importante llevar un cierto manejo de errores para evitar "crashes" de la aplicación y mejorar la funcionalidad.(Global error boundary for crashes).

3.1.4. 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.

3.2. Arquitectura

La arquitectura de VitHabitus se basa en un modelo cliente-servidor, donde la aplicación móvil actúa como cliente principal y se comunica con los distintos servicios de back-end y almacenamiento. Se trata de una estructura modular que separa la lógica para un mejor mantenimiento y persistancia de datos Principalmente destacan tres bloques que conforman la estructura 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. Realmente conforma también la backend.

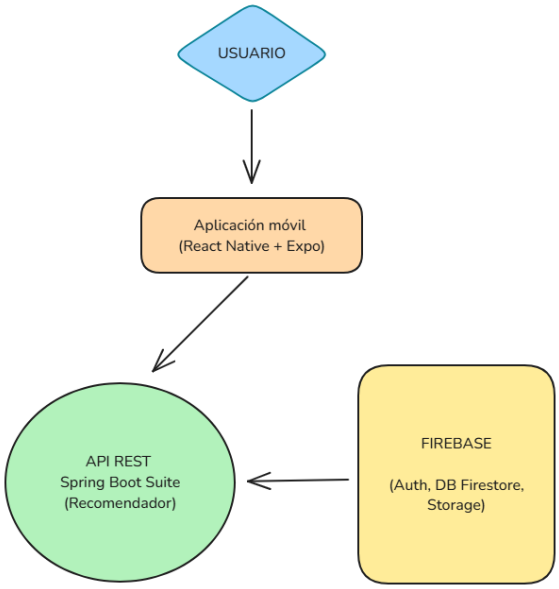

Figura 3.1: Arquitectura proyecto

Dada la arquitectura que se puede ver en la (figura 3.1), es importante definir las bases del flujo de la aplicación en cada uno de los procesos.

Desde el punto de vista estructural, el proceso de autenticación está completamente gestionado por **Firebase Authentication**, que permite al usuario registrarse, iniciar sesión, cerrar sesión, verificar cuenta de correo electrónico y solicitar recuperación de contraseña. Además se ha configurado el cierre automático de sesión cuando la aplicación se cierra, reforzando la protección de la información lo máximo posible.

Una vez el usuario se ha autenticado, este accede al conjunto de pantallas funcionales de la aplicación a través de un sistema de navegación estructurado en pestañas (tabs) y funciones de navegación router-expo. Los datos por otra parte se cargan y almacenan dinámicamente desde Firebase o mediante peticiones HTTP a la API REST.

Concretamente, la aplicación utiliza Firebase Firestore como base de datos principal. Todos los datos del usuario (hábitos, recomendaciones, notas, historial ORI) se almacenan en documentos dentro de colecciones organizadas por identificador único. Firestore ofrece sincronización en tiempo real, lo que permite que los cambios sean visibles al instante en todos los dispositivos y sesiones del usuario.

Las operaciones CRUD (crear, leer, actualizar y eliminar) se realizan de forma directa desde el cliente, mediante el SDK de Firebase gracias a las validaciones locales y a las alertas visuales para garantizar una buena experiencia de usuario. Esta comunicación directa con la base de datos simplifica enormemente la arquitectura al reducir la necesidad de intermediarios (API) para la mayoría de las operaciones que no afectan al recomendador.

Uno de los procesos más relevantes, desde el punto de vista arquitectónico de la aplicación, es la ejecución del sistema de recomendación. Este se activa cuando el usuario pulsa el botón de "Ejecutar recomendador" desde la pantalla de hábitos como ya hemos visto antes. En ese momento, la aplicación empaqueta los datos actuales en un objeto JSON estructurado y los envía a la API mediante una petición HTTP POST.

La API REST (desarrollada con Spring Boot Suite) recibe esta petición, valida los datos y activa internamente el algoritmo evolutivo que compone el recomendador. Este sistema trabaja sobre el conjunto de 100 modelos predictivos previamente entrenados y devuelve dos resultados clave:

- Una lista de hábitos modificados, recomendados para el usuario para obtener un mejor coeficiente ORI, junto con su ORI objetivo.

- Un valor **ORI** (Obesity Risk Indicator) que representa el nivel de riesgo actual del usuario.

Dado un conjunto de características F_i de una persona, cada modelo de predicción M_i proporciona una predicción binaria $p_i \in \{0, 1\}$ sobre el riesgo de ser obeso o tener sobrepeso, es decir, tener un Índice de Masa Corporal (IMC) superior a 25, sin conocer directamente los datos relacionados con el IMC. Con esas predicciones, definimos el Indicador de Riesgo de Obesidad (ORI), expresado como:

$$\texttt{ORI} = \sum_{i=1}^{n} w_i \cdot p_i \tag{3.1}$$

donde $w_i \in [0, 1]$ es el peso de cada modelo M_i en el modo conjunto (ensemble). En este trabajo, se establecen todos los valores de $w_i = 1$ para $i = 1$ hasta n. Esto hace que el valor del ORI esté en el rango $[0, 100]$, siendo 100 el mayor riesgo de obesidad o sobrepeso.

Estos resultados se almacenan automáticamente en Firestore (conexión Firestore-API REST) y se asocian al historial del usuario, quedando accesibles tanto desde la pantalla de resultados como desde el calendario.

Por último, con respecto al recomendador, tras presionar el botón de ejecución si el usuario cierra el mensaje de alerta de "proceso en ejecución" se activa un sistema de notificación persistente en la parte superior que indica que el proceso sigue ejecutándose en segundo plano. Esta funcionalidad ha sido diseñada para ofrecer una experiencia no bloqueante mientras se realiza un proceso computacional intensivo en el backend.

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 Conclusions and Future Work al final de la memoria.

3.3. Estructura general

En este capítulo se tratará de forma detallada el flujo del usuario a través de la aplicación VitHabitus. Paso a paso, se explica la navegación que el usuario podrá realizar por las diferentes pantallas. La aplicación cuenta con una estructura pensada en facilitar la navegación al usuario a las distintas funcionalidades principales de forma intuitiva y rápida, manteniendo una interfaz simple y explicativa.

Desde el punto de vista funcional, el flujo es lineal con ciertos accesos secundarios desde la pantalla de inicio, como veremos a continuación.

Nada más abrir la aplicación nos encontramos en la primera etapa de **Inicio y Autenticación** (Ver figura 3.2). El usuario se encuentra con una pantalla de inicio que le da acceso al registro o al inicio de sesión. Además de la funcionalidad de recuperación de contraseña, se aprecian, en la ventana de "Creación de cuenta", todos los campos necesarios para poder registrarse junto con los términos y condiciones. Esta primera fase es muy importante para asegurar que cada usuario pueda tener su propio entorno de datos privado. El proceso de autenticación se realiza mediante Firebase, configurado para un log-out automático cada vez que se abandona la aplicación.

Una vez registrado el usuario avanza a la pantalla principal en la cual puede acceder a todas las funcionalidades de la aplicación. Entre ellas se encuentran las principales, "Mis hábitos", "Resultados", "Notas" y "calendario".

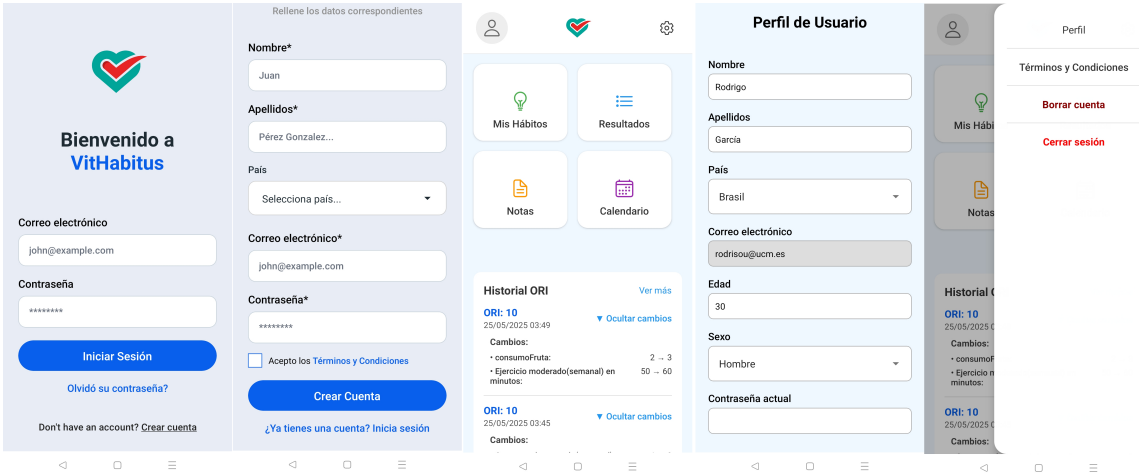

Figura 3.2: Registro, pantalla principal y perfil

Además de las funcionalidades principales, en la pantalla de inicio aparece un historial de ORI en el cual se muestran los valores ORI de las últimas evaluaciones de hábitos. Además, existe el desplegable de configuración, el cual permite al usuario ir al perfil, borrar su cuenta por completo y cerrar sesión, lo que lo devolverá a la pantalla de registro.

En la pantalla de perfil, se encuentran los datos personales del usuario, nombre, apellido, etc. Siendo el correo electrónico el único no modificable por motivos de robustez de usuarios y evitar fraudes. Por otro lado el usuario puede modificar su contraseña en todo momento.

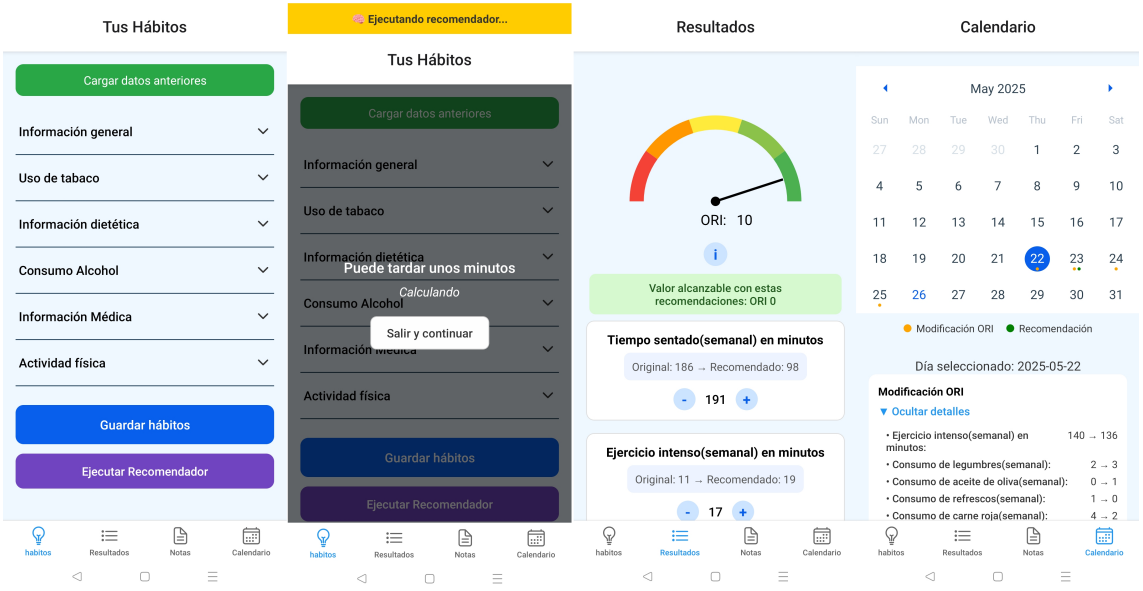

Figura 3.3: Hábitos, pantalla de Resultados y calendario

Empezando por la funcionalidad principal, nos encontramos en la pantalla de Hábitos tras seleccionar “Mis Hábitos” (Ver figura 3.3). Esta pantalla está formada por un formulario divido en distintas secciones temáticas con un sistema de acordeón

para facilitar la visualización de los datos en pantallas pequeñas como las de los teléfonos móviles. Cada una de estas secciones del formulario deben ser rellenadas para que el usuario pueda ejecutar el recomendador. Para ello, cuentan con un sistema de validaciones para evitar introducir datos no reconocibles para el sistema. Además del formulario, existe el botón de cargar datos procedentes de la base de datos y el botón de guardar los datos del formulario en la base de datos sobrescribiendo los anteriores. Toda la aplicación esta compuesta de mensajes de alerta para notificar en todo momento al usuario si en algún momento sus acciones pueden comprometer los datos.

Por último, tras rellenar el formulario, el usuario puede ejecutar el sistema de recomendación (botón de Ejecutar recomendador). Esto lanza una solicitud a la API REST la cual procesa los hábitos introducidos y devuelve una lista de recomendaciones junto con un valor numérico ORI (Obesity Risk Indicator) a la base de datos. Tras finalizar la ejecución, un mensaje de alerta se despliega indicando el valor ORI de los hábitos del usuario y el valor ORI de los hábitos objetivo tras completar las recomendaciones.

Como funcionalidad añadida, en período de ejecución se mostrará un mensaje en una pantalla de carga acerca de la posible duración del proceso. En cualquier momento el usuario puede decidir salir de dicha pantalla y dejar que el proceso se ejecute en segundo plano como puede verse en el mensaje que se despliega en la zona superior de la pantalla.

Por otro lado, en el caso de pulsar en “Resultados” en la pantalla de inicio (Ver figura 3.3), el usuario irá a una pantalla con una gráfica de tipo velocímetro que representa visualmente el valor ORI de sus hábitos. Bajo el velocímetro se encuentra el valor ORI objetivo y el listado de recomendaciones generadas con sus valores de hábito anteriores y los sugeridos. De esta manera podrá ir realizando los cambios oportunos y calculando de nuevo el coeficiente ORI de los nuevos hábitos (pulsando en el botón de calcular y guardar).

No hay que olvidar la ventana “Calendario” (Ver figura 3.3), accesible desde la pantalla de inicio. Se trata de una funcionalidad destacable de la aplicación en la cual, además de mostrar un calendario completo y funcional, el usuario puede visualizar las acciones realizadas cada uno de los días como la ejecución del recomendador con sus recomendaciones o la evaluación de unos nuevos hábitos. Al seleccionar el día concreto, se despliegan cada una de las acciones antes mencionadas que se hayan producido dicho día. Esto refuerza la idea de seguimiento e historial que favorece al usuario a seguir utilizando la aplicación.

Cabe mencionar que para poder navegar a estas pantallas, además de los botones de la pantalla de inicio, una vez te encuentras en una de estas pantallas, existe una barra horizontal inferior con cada una de las secciones para un acceso rápido.

Por último, se encuentra la pantalla de “Notas” en la cual el usuario se encuentra con un gestor completo de notas. Pudiendo crear notas con su título y contenido, borrarlas, editarlas y buscarlas por nombre del título en caso de haber demasiadas.

Con esto termina el flujo del usuario a través de la aplicación VitHabitus. Es importante tener en cuenta que el funcionamiento de la aplicación depende de la

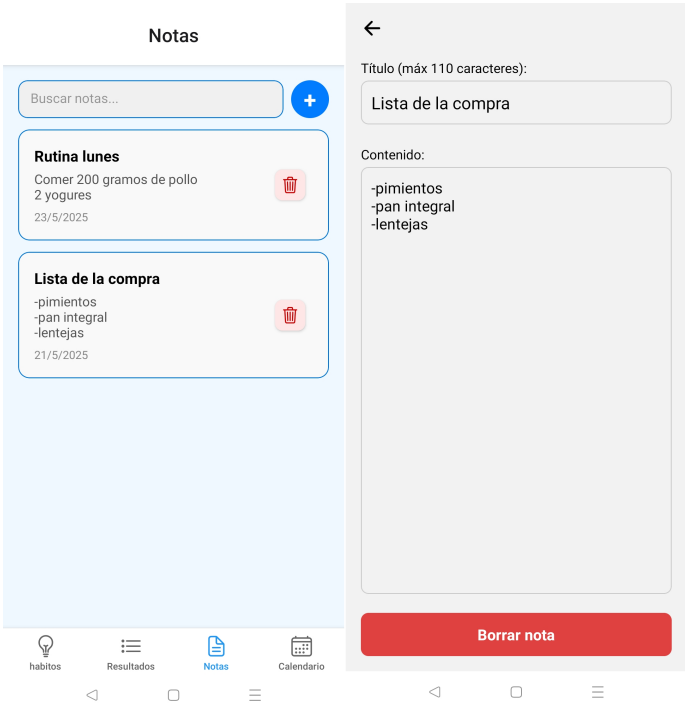

Figura 3.4: Pantalla de Notas

API REST y del correcto funcionamiento de la base de datos Firebase. A lo largo de todo el flujo se ha ido haciendo comprobaciones de identificación de usuario y de conexión con el backend para evitar fisuras.

3.4. Desarrollo de la 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 profundidad el funcionamiento de la aplicación 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/ constitu-

ye el núcleo de la aplicación y en su interior se encuentran diferenciadas las distintas pantallas principales y sus componentes reutilizables, así como dato, configuraciones, gestor de navegación, etc.

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 habits.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, muchos de los cuales son aplicados en las páginas descritas antes. En app/components/ se dividen, segun su potencial uso, componentes como casillas y botones con sus determinados estilos. También hay elementos visuales que conforman la interfaz de usuario, la idea es buscar el encapsulamiento y la 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 ficheros necesarios para la implementación de la API y de la Firebase; firebaseConfig.ts con su configuración de conexión y VerifyApi.ts necesario para la verificación de conexión con la API.

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 utiliza la aplicación. En caso de no saber que dependencias instalar, en el 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 basándose en la estructura 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. 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.

3.4.1. 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`, básicamente un selector desplegable que permite al 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 basa en estructuras de datos externas por lo que es más fácilmente implementable.

- Parecido al `PickerComponent.tsx`, encontramos el **AcordeonHabits.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 construyen a través de data/formHabitsStructure.ts que otorga gran flexibilidad al proceso de creación del formulario de hábitos. Es fácilmente modificable añadiendo nuevas secciones en dicho 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 HeaderHome.tsx que es menos aplicable a otras páginas, ya que representa un header bastante específico.

- Por último, hemos de resaltar otros componentes visuales 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 útil 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.

3.4.2. 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 evalúa 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 evalúa en tiempo real ese valor y emite una corrección. Si es válido, se acepta, pero en caso contrario, reproduce un 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 petición HTTP POST.

3.5. Servicios backend

3.5.1. 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 estructura se especifica después.
- **Campo apellido**: Apellido incluído 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 contacto (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 se 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.

3.5.2. 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.
- Función de verificación de una dirección de correo a través de un correo electrónico para evitar creaciones masivas de cuentas de usuario falsas.
- Notificaciones varias de inicio de sesión a través de Google, Facebook, Github, etc; y la funcionalidad de restablecer la contraseña en caso de extraviarla

Estas funcionalidades se han implementado en las pantallas login.tsx y register.tsx. Durante el flujo de procesamiento en estas páginas Firebase valida los campos rellenados y comprueba que 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.

3.5.3. 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 provocaría 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 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 dividido el recomendador en dos partes: evaluador de hábitos que genera el valor del ORI y búsqueda de recomendaciones que realiza todo el proceso para obtener las recomendaciones finales más óptimas.

3.5.3.1. 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)).

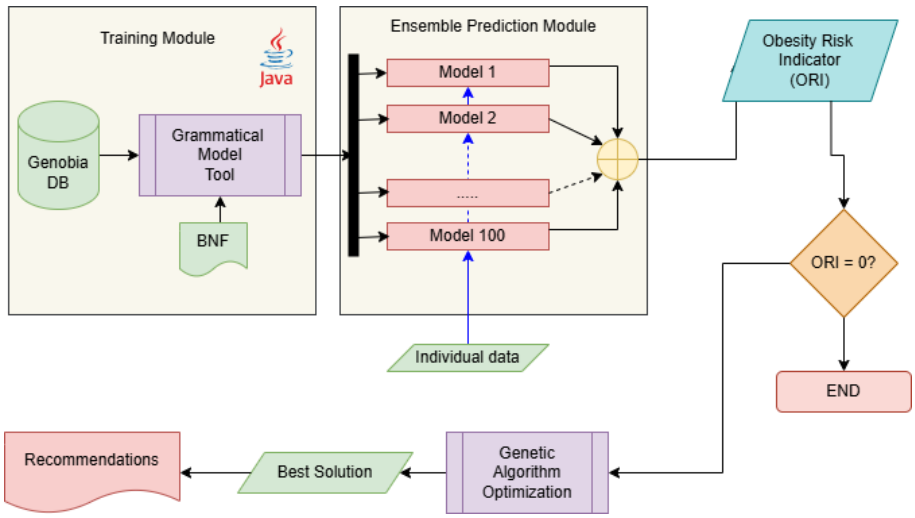

Figura 3.5: Arquitectura general del sistema de recomendación

Este sistema se basa en un algoritmo evolutivo, una técnica de inteligencia artificial inspirada en los procesos de selección natural. El objetivo del algoritmo es generar combinaciones de hábitos que reduzcan el riesgo de padecer obesidad, respetando tanto variables inmutables como edad o sexo y variables modificables como ejercicio físico, alimentación, etc. De esta forma, a partir de los hábitos introducidos, el sistema genera una población inicial de soluciones representadas mediante 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.

Se trata de la unificación de dos esquemas presentes en el documento "An Evolutionary 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. Para ello, el recomendador cuenta con 100 modelos predictivos entrenados previamente sobre 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.

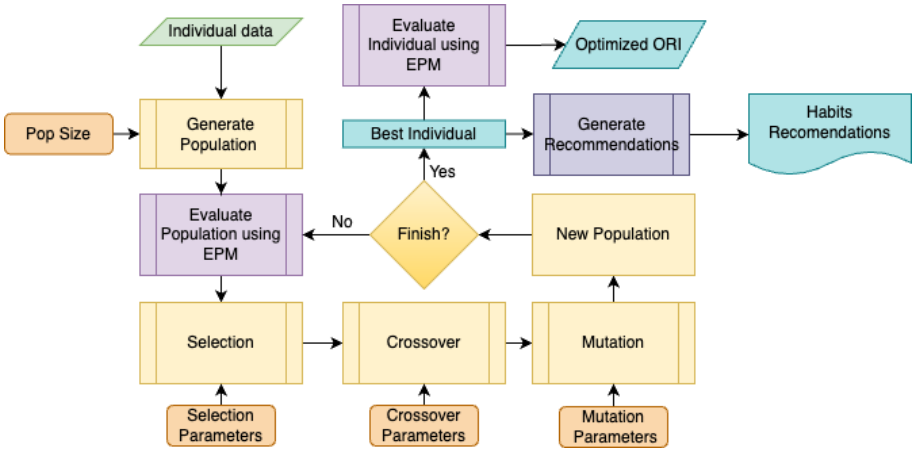

Figura 3.6: Arquitectura general del sistema de recomendación

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.

La solución final seleccionada es aquella que presenta un mejor equilibrio en conjunto de hábitos saludables, teniendo el coeficiente ORI (Obesity Risk Indicator) más bajo que el resto.

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.

3.6. Caso de uso: Ejemplo completo de evaluación y recomendación

Este capítulo tiene como objetivo ilustrar el funcionamiento real del sistema de recomendación implementado en la aplicación. A través de un caso práctico, se describe paso a paso cómo un usuario tras su inicio de sesión con su correo electrónico y contraseña, introduce sus hábitos, cómo se ejecuta el sistema recomendador y qué tipo de resultados obtiene, incluyendo las recomendaciones personalizadas y el cálculo del índice de riesgo de obesidad (ORI).

3.6.1. Perfil del usuario

Para este ejemplo, se ha creado un usuario ficticio con las siguientes características generales:

- **Nombre:** Juan García.
- **Edad:** 49 años.
- **Sexo:** Hombre.
- **Nacionalidad:** Argentina.

3.6.2. Hábitos introducidos en la aplicación

Desde la pantalla "Hábitos" de la aplicación móvil, el usuario accede a un formulario estructurado por secciones donde introduce la siguiente información:

- Tiene estrés.
- Duerme 8 horas diarias.
- Nivel educativo de secundaria, viviendo en una población de más de 50.000 habitantes.
- Ingreso entre 1000 y 2000 euros como cajero y dependiente.
- Actividad física semanal:
  - Ejercicio moderado: 60 minutos.
  - Ejercicio intenso: 150 minutos.
  - Caminata: 435 minutos.
  - Sentado: 240 minutos.
- Alimentación:

- Aceite de oliva: 3 raciones al día.
- Fruta: 3 pieza al día
- Verdura: 2 raciones al día
- Carne roja: 6 porciones al día
- Mantequilla: 0 al día
- Legumbres: 0 veces por semana
- Pescado: 5 raciones a la semana
- Sofrito : 0 raciones a la semana
- Consumo mayor de carne blanca que carne roja
- Lácteos: 4 raciones al día y mayor consumo de productos desnatados que normales.
- Bollería industrial: 0 veces por semana
- Refrescos azucarados: 1 vasos al día
- Aceite de oliva: 0 cucharadas al día

- Consumo de sustancias:
  - Tabaco: 1 cigarrillo al día
  - Alcohol (licor): 0 copas de todo tipo a la semana

- Enfermedades:
  - Ningún tipo de enfermedad

Una vez completado el formulario y validada la entrada, el usuario pulsa el botón *Ejecutar recomendador*.

3.6.3. Ejecución del recomendador

La aplicación empaqueta los hábitos introducidos en un objeto JSON y lo envía mediante una petición HTTP POST a la API REST desarrollada en Spring Boot. Esta API procesa los datos usando el sistema recomendador basado en algoritmos evolutivos y modelos predictivos previamente entrenados.

El sistema evalúa el estado actual del usuario y genera una propuesta personalizada de mejora, obteniendo los siguientes resultados:

- **ORI inicial:** 88
- **ORI Objetivo tras aplicar todas las recomendaciones:** 39

3.6.4. Recomendaciones generadas

El sistema devuelve una lista de modificaciones sugeridas para los hábitos introducidos, con el objetivo de reducir el riesgo de obesidad. A continuación se muestra un resumen de las recomendaciones obtenidas:

Hábito	**Valor actual**	**Recomendado**
Frutos secos	0	2
Carne roja	6	4
Refrescos	1	0
Lácteos	4	5
Legumbres	0	2
Ej. Caminando	435	324
Aceite de Oliva	3	1
Fruta	3	2
Consumo tabaco	1	0
Pescado	5	7
Ej. moderado	60	72
Tiempo sentado	240	228
Consumo verdura	2	1
Horas de sueño	8	0
Ej. intenso	150	145

Estas recomendaciones representan una modificación progresiva de los hábitos actuales del usuario, ajustada a un perfil realista y factible a corto o medio plazo.

3.6.5. Visualización en la aplicación

Una vez recibida la respuesta del recomendador, la aplicación redirige al usuario a la pantalla de "Resultados". Allí se muestra un velocímetro con el valor actual del ORI, el ORI recomendado y un listado de todas las sugerencias generadas, incluyendo los valores anteriores y propuestos (Ver Figura 3.7).

Además, los resultados se almacenan automáticamente en el historial y pueden ser consultados posteriormente desde la funcionalidad de "Calendario". Cada ejecución del recomendador queda vinculada a la fecha correspondiente, permitiendo al usuario hacer un seguimiento de su evolución a lo largo del tiempo.

3.6.6. Conclusiones del caso

Este caso práctico pone de manifiesto la capacidad de VitHabitus para analizar los hábitos del usuario, identificar áreas de mejora y generar recomendaciones personalizadas. La reducción significativa del valor ORI demuestra que el sistema puede ser una herramienta útil para promover hábitos más saludables y prevenir enfermedades relacionadas con el sobrepeso y la obesidad. Además, el proceso puede repetirse de forma iterativa si el usuario desea seguir optimizando sus hábitos con el objetivo de alcanzar un ORI aún menor.

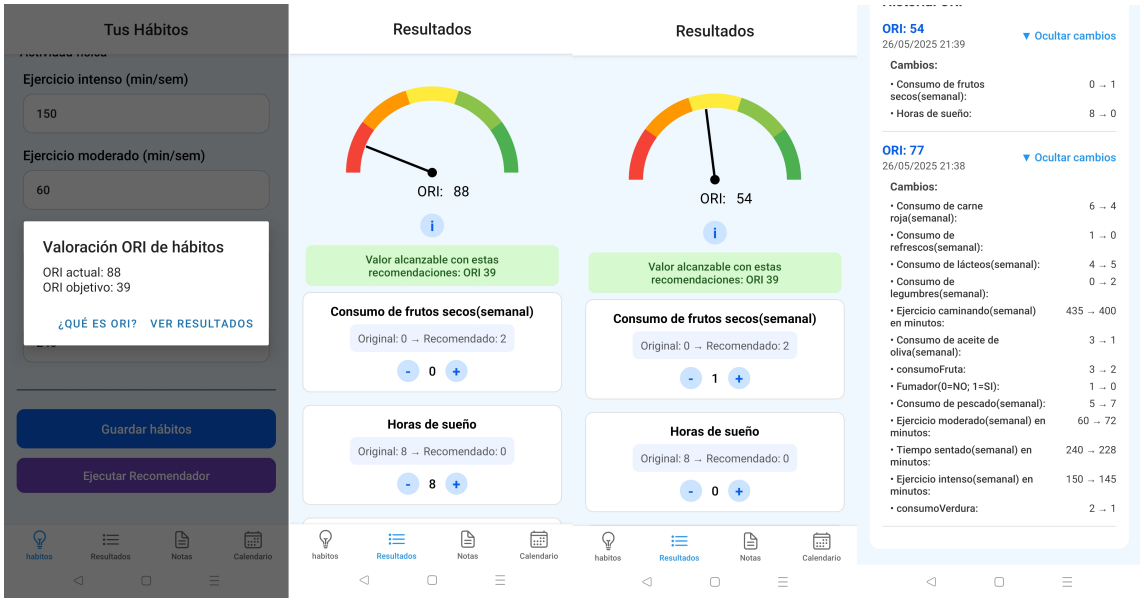

Figura 3.7: Caso de uso valor ORI, desde la ejecución del recomendador hasta la visualización del historial de los cambios de ORI

3.7. 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 a sus librerias que facilitan, en gran medida las conexiones backend, como en el caso de una API o Base de Datos. (web official React Native (2025))

- **Expo**: Entorno de desarrollo utilizado para aplicaciones móviles basadas en React Native. Expo proporciona una experiencia completa en la creación prueba y despliegue de una aplicación en diferentes dispositivos móviles android y iOS, gracias 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. Librerías como Expo router que implementan de forma sencilla la navegación entre pantallas basándose en la estructura de archivos y haciendo el proyecto más escalable. Entre otras.(EXPO (2025b))

- **NodeJS**: Es un entorno de ejecución para JavaScript. Se utiliza en el proyecto como base para el entorno de desarrollo de React native, siendo 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 creación de API-Rest.(web officialNodeJS (2025))

- **NPM (Node Package Manager**: Es un gestor de paquetes asociado al ecosistema 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 interfaz 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 4

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, además de la integración del algoritmo de recomendación de hábitos y un correcto flujo de funcionamiento entre todos los elementos.

Queda, como resultado, una aplicación que es de gran utilidad para muchas personas y que demuestra una vez más, el increíble potencial que tiene la inteligencia artificial aplicada a la salud. Además demuestra cómo 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.

4.0.1. Trabajo Futuro

Independientemente del resultado, siempre hay posibilidad de mejora y avances. De esta forma, hay algunas 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, calorías 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 estándares 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 ciertas 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 4.

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

El que lee mucho y anda mucho, ve mucho y sabe mucho.

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.

GenObia. An evolutionary habit recommender system to reduce the risk of overweight and obesity. *None*, páginas 3–6, 2025.

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.