

2025

1

IES San Vicente

Antonio Vieira Rubio

# [PROYECTO FINAL DE CICLO: SHELLTRACKER]

Memoria del proyecto final de ciclo

1.- Introducción.....	3
2.- Antecedentes, Contexto y Trabajos Futuros.....	6
2.1.- Aplicaciones de preparadores físicos o seguimiento nutricional.....	6
2.2.- Entendiendo el contexto, ¿Hacia dónde va ShellTracker y por qué cubre una necesidad?.....	8
2.3.- Trabajos futuros en ShellTracker.....	9
3.- Diseño.....	10
3.1.- Arquitectura general.....	10
3.2.- Tecnologías utilizadas.....	10
3.3.- Estructura funcional de la aplicación.....	11
3.4.- Bases de datos.....	14
3.5.- Flujos de trabajo.....	17
3.6.- Módulos de la aplicación.....	18
4.- Resultado.....	19
4.1.- Pantallas de la aplicación.....	19
4.2.- Implementación del sistema de consultas a la API pública.....	23
4.3.- Implementación del sistema de autenticación con Firebase.....	24
5.- Conclusiones.....	26
5.1.- Problemas encontrados.....	26
6.- Manual del usuario.....	26
6.1.- ¿Qué es ShellTracker?.....	26
7.- Bibliografía.....	33

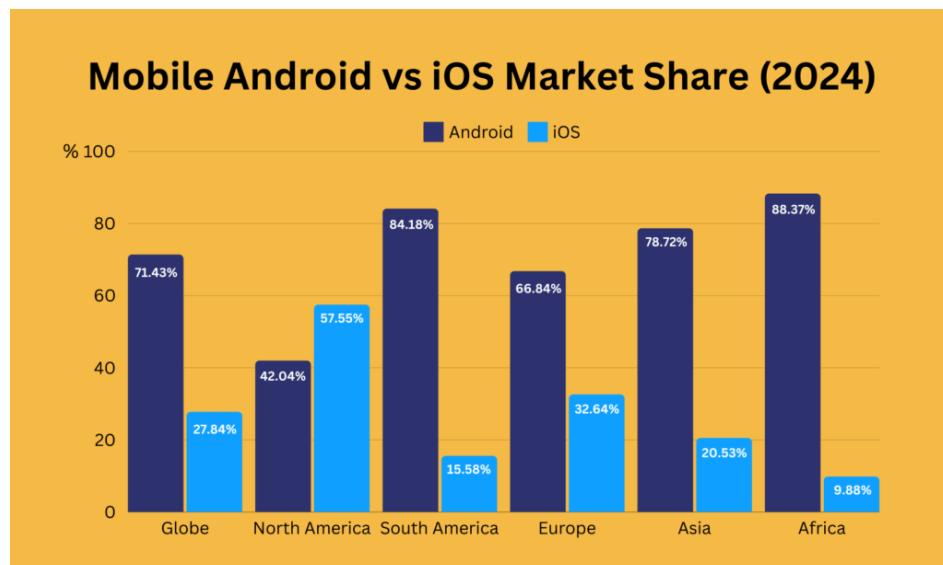
## 1.- Introducción.

Dado todo lo visto en el ciclo superior de Desarrollo de Aplicaciones Multiplataforma, he optado por desarrollar un aplicación multiplataforma, una aplicación disponible tanto para Android como para iOS, es cierto que durante estos dos años el lenguaje de programación que más he aprendido ha sido Java, pero he enfocado este proyecto a un reto, a un aprendizaje real y desde 0, pues uno de los principales objetivos es ganar independencia y autoridad con el código.

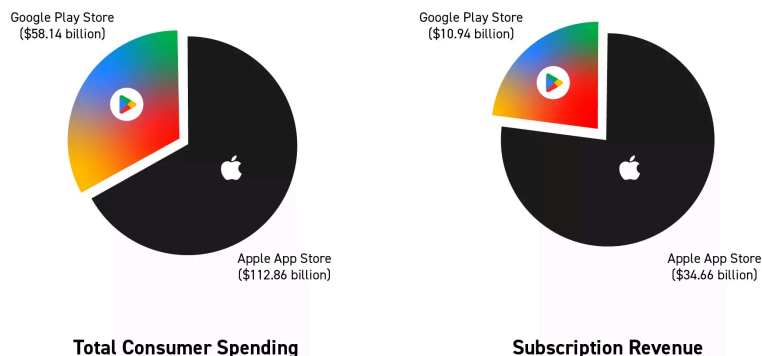
El imparable auge de la tecnología y la telefonía móvil, su consolidación y maduración convierte las aplicaciones multiplataforma en un nicho en el que poder centrarse sabiendo la importante perspectiva de negocio con la que cuentan gracias a su versatilidad.

ShellTracker está desarrollada en React-Native, framework de desarrollo de aplicaciones móviles de código abierto, creado por Facebook, que permite construir aplicaciones nativas para Android e iOS utilizando JavaScript y React. En lugar de crear aplicaciones híbridas que se ejecutan en un navegador, React Native genera aplicaciones nativas que se comportan como si fueran escritas en los lenguajes nativos de cada plataforma, esto nos genera un punto de partida realmente ventajoso ya que nos permite aparecer en ambos mercados, tanto Play Store como App Store.

Es cierto que la mayoría de usuarios de aplicaciones móviles los encontramos bajo el sistema operativo de Android, por lo que en un primer estudio del contexto podríamos pensar que es suficiente con desarrollar bajo ese modelo de negocio tal y como se muestra en la gráfica:



Pero no debemos olvidar que los usuarios de iOS son los que más están dispuestos a comprar servicios y/o suscripciones de pago, por lo que al desarrollador le puede derivar en un menor tráfico en la aplicación, pero más rentabilidad económica:



En cuanto a el almacenamiento de los datos, he optado por dos soluciones completamente diferentes, tendremos por un lado una base de datos PostgreSQL, encargada de almacenar todos los registros de los alimentos que cada uno de los usuarios haga por su cuenta a lo largo del día, ahora bien, esos alimentos no están almacenados en dicha base de datos, si no que los recuperamos de una API pública. La aplicación, al tener que relacionarse con otros sistemas, me ha obligado a implementar una solución para el intercambio de información en JSON, mediante GraphQL que es un lenguaje de consulta de APIs que permite pedir exactamente los datos que necesitamos; una vez tenemos esos datos StepZen nos permitirá crear la API GraphQL de forma más sencilla y declarativa.

Por otro lado, almacenamos los usuarios que se registren en la aplicación en una base de datos no relacional, Firebase. Como experiencia personal y tras enfrentarme a este tipo de base de datos durante la FCT, me genera una comodidad incomparable, más sencilla de consultar, más sencilla de desarrollar y también, más sencilla de escalar.

En lo relacionado a la interfaz de usuario, sobretodo me he centrado en un diseño limpio, algo minimalista y lo más sencillo posible en cuanto a lectura y usabilidad de cara al usuario, para ello optamos por Material UI (MUI) que nos ha permitido un diseño limpio, moderno y responsive; además también incorporé estilos propios mediante CSS e iconografía.

El tema principal de la aplicación es el seguimiento alimenticio, principalmente de las calorías diarias que ingerimos. Esto lo conseguimos gracias a las consultas de los alimentos de la API externa y a los registros de estos alimentos que cada usuario realiza. A través de un formulario y un estudio sencillo de su estilo de vida, podemos averiguar cuál es la ingesta calórica recomendada para cada uno de los usuarios por lo que, mediante los registros de los alimentos, iremos restando esas calorías a las recomendadas de cada usuario. Si los valores del formulario del usuario cambian a lo largo del tiempo, ya sea el peso, el estilo de vida, o incluso la altura, cada uno puede ir adaptando esos datos a su progreso y por resultante necesidad, las calorías recomendadas se irán modificando automáticamente y en tiempo real, sin necesidad de actualizar la aplicación, adaptándose así al usuario. Esto es gracias al sistema de estados que ofrece React.

Los usuarios deberán darse de alta en el servicio de la aplicación para poder utilizar sus funcionalidades. Será de vital importancia completar el formulario de registro con honestidad para poder sacarle el máximo partido a la aplicación. En este caso, no recae ninguna responsabilidad en el usuario de cara a la integridad de la base de datos, ninguno de los campos del formulario que rellene perjudicarán ya

que todos están gestionados mediante validación y comprobación de los datos. El campo o clave que utilizaremos para relacionar los alimentos y los usuarios será el “uid” que en ningún momento el usuario sabe de la existencia de ese campo.

Una vez registrados, deberán iniciar sesión contra la base de datos, y si las credenciales de autorización son correctas, podrán acceder al menú principal. Una vez se inicie sesión se guardará dicha sesión de usuario, para que cada vez que accedan a la aplicación no tengan que volver a pasar por el formulario de login a no ser que cierre la sesión de forma voluntaria. Pero aunque se cierre la sesión se guardarán los alimentos y el seguimiento calórico que se ha ido realizando mediante ese día.

Como ya se ha mencionado, la aplicación permite realizar un seguimiento de las calorías de los alimentos que el usuario ingiere, es principalmente eso lo que el usuario encontrará en la pantalla principal; además tendrá otra pantalla de búsqueda de alimentos y otra relacionada con el perfil del usuario donde podrá tanto modificar algunos datos del perfil y cerrar sesión.

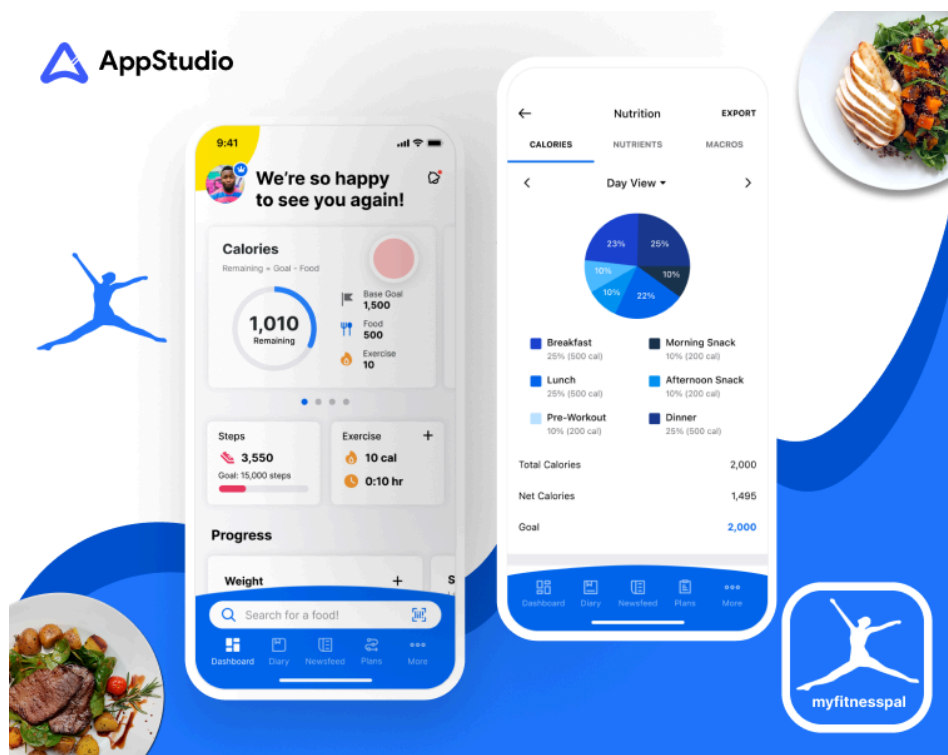
## 2.- Antecedentes, Contexto y Trabajos Futuros.

### 2.1.- Aplicaciones de preparadores físicos o seguimiento nutricional.

Todas estas aplicaciones guardan un factor parecido al nuestro, aunque en el caso de ShellTracker me limitaré a exponer su funcionalidad directa al seguimiento calórico, sin realizar seguimientos macro-nutricionales o seguimiento deportivo. Con esto, podré destacar una interfaz de usuario limpia, una funcionalidad intuitiva y menos inversión de tiempo por parte del usuario dentro de la aplicación.

#### ·MyFitnessPal

Es la más conocida dentro del nicho del seguimiento deportivo y/o nutricional por cuenta propia, se trata de una aplicación muy completa, con versión gratuita pero realmente muy limitada, y versión de pago. Es otra aplicación para ayudar al usuario a lograr sus objetivos alimenticios y de actividad física.



#### ·Harbiz



Es también la más conocida, pero en este caso de un nicho diferente, el nicho del seguimiento de actividad física y nutricional por parte de un profesional, es la que más utilizan los preparadores físicos. Muy completa, a la vez que compleja, sin lugar a dudas, el usuario promedio tarda en acostumbrarse a su uso, pero una vez lo hacen, no encuentran otra alternativa en la competencia. Permite generar dietas según los requisitos exactos del cliente, teniendo en cuenta patologías médicas, gustos, alergias... Permite seguimiento del ejercicio físico y estudio del progreso. Me sirve como inspiración pero no es un punto de referencia que quiera alcanzar con ShellTracker.

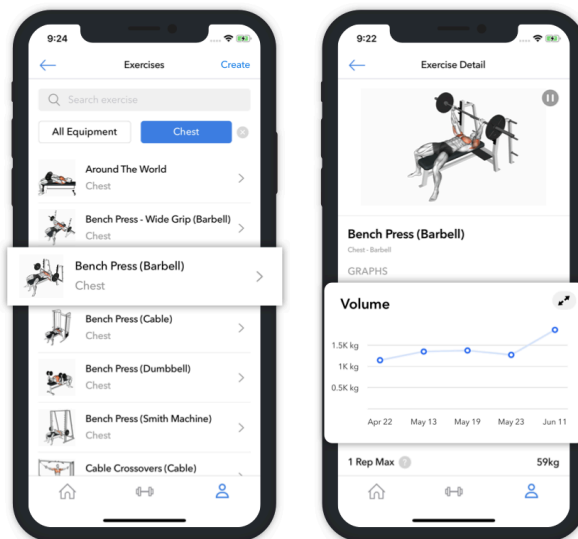


·Hevy

Es otra aplicación para el seguimiento de la actividad física con modalidad de pago y gratuita, pero solo para eso, no tiene ninguna otra funcionalidad salvo la de agregar amigos, interfaz sencilla, intuitiva, minimalista, necesita de poca inversión de tiempo en la aplicación para su uso, sin lugar a dudas, me ha servido de inspiración y en un futuro, una de las funciones que me gustaría incorporar en ShellTacker es el seguimiento de la actividad física de la misma manera o similar que lo hacen aquí.

## Advanced tracking

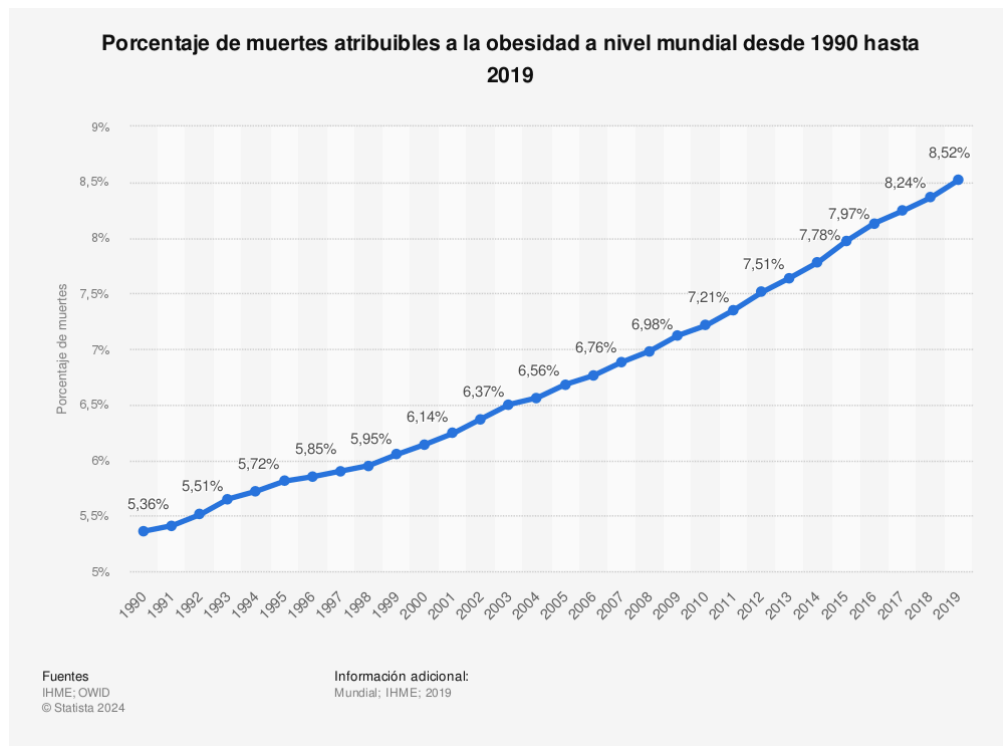
-  **Calculate One Rep Max**
- 1RM** **High quality exercise videos**
- +200** **Create custom exercises**
-  **Advanced charts**



## 2.2.- Entendiendo el contexto, ¿Hacia dónde va ShellTracker y por qué cubre una necesidad?

Una vez puesto en contexto y como ejemplo las anteriores tres aplicaciones me facilita explicar y detallar cuál es el propósito de ShellTracker y hacia donde me gustaría apuntar o qué nicho me gustaría cubrir.

Con dicha aplicación me gustaría cubrir el nicho del seguimiento deportivo y/o nutricional por cuenta propia, toda autonomía e independencia por parte del usuario siempre va a ser bienvenida, pero de una manera diferente a MyFitnessPal, no está la necesidad de que el usuario pague por funcionalidad de la app ni que esté obligado a ver publicidad para usar la aplicación, quiero que desde un primer momento el usuario pueda realizar el seguimiento nutricional de manera completa al igual que podemos hacer con Hevy el seguimiento de actividad física. De manera sencilla, intuitiva y minimalista, como sello de identidad de ShellTracker quiero mantener una interfaz en la que el usuario invierta el menor tiempo posible. En todas las aplicaciones de la competencia, se nos obliga a los usuarios a invertir un tiempo que a veces, no tenemos e incluso a realizar una curva de aprendizaje de la interfaz de la aplicación que no tenemos porqué hacerla.



Tal y como se muestra en la gráfica, la obesidad es a día de hoy un problema creciente en la sociedad actual por lo que si pensamos en una solución, al alcance todo el mundo, a través de cualquier dispositivo móvil, sin necesidad de inversión económica y mediante una inversión mínima de tiempo, podemos ayudar a todas esas personas que no tienen conocimientos en consumo calórico y/o nutrición a través de ShellTracker, donde podrán ser más conscientes de sus requisitos calóricos y de la ingesta de alimentos y calorías diarias que realizan.

La aplicación, en esta primera versión, deberá permitir a los usuarios añadir alimentos a su registro diario y buscarlos según sus ingestas o curiosidades, además de realizar el conteo o resta de las calorías recomendadas de cada usuario.



Las principales funciones de la aplicación son:

- Alta de usuarios nuevos y modificación de usuarios existentes
- Inicio y finalización de la sesión de los usuarios.
- Los usuarios podrán tener imagen de perfil si lo desean. Dicha imagen podrá ser capturada usando la cámara o accediendo a la galería del dispositivo móvil.
- Datos de los usuarios almacenados en la aplicación para evitar que deba introducir sus datos de inicio de sesión cada vez que acceda a la aplicación.
- Alta o registro de alimentos en el perfil del usuario para poder llevar a cabo el conteo calórico.

Para ello se realizará la búsqueda de los alimentos en una API externa y se insertarán los datos en nuestra base de datos cuando el usuario agregue dicho alimento.

- Estudio de la actividad física, peso, altura y género mediante una fórmula para aconsejar el consumo calórico.
- Resta calórica de las calorías recomendadas y suma de las calorías ingeridas.
- Actualización de los datos de los usuarios y los registros de los alimentos en tiempo real tanto en la aplicación como en la base de datos.

### 2.3.- Trabajos futuros en ShellTracker.

Una de las principales recomendaciones por los profesionales para combatir la obesidad es la dieta sana acompañada de ejercicio, por lo que una de las incorporaciones que tendrá ShellTracker será el seguimiento de la actividad física centrada en la fuerza. La idea es incorporar una interfaz de usuario similar a la que ya tenemos o a la que encontramos en Hevy, que nos permita también la búsqueda de ejercicios, al igual que en la web de [ShellFitness](#), también desarrollada completamente por mí con React y Material UI.

Además de un seguimiento de la actividad física, podríamos mejorar el seguimiento nutricional mediante el seguimiento macro-nutritivo, incorporaré una ampliación de la interfaz de usuario donde además de ver el consumo recomendado e ingesta de calorías, el cliente pueda ver los macronutrientes recomendados e ingeridos; de esta manera, podrá ser mucho más consciente de sus necesidad de grasas saludables, carbohidratos y proteínas.

Una vez tengamos ambas funcionalidades, seguramente el cliente o usuario de la aplicación ya tenga la suficiente adherencia para interesarse por su progreso más allá de los resultados obvios, por lo que incorporaré una sección de “Progreso”, donde se podrán consultar gráficas de las ingestas anteriores, estadísticas en progreso del peso según sus objetivos y comparación de fotos personales del usuario si desea subirlas.

### 3.- Diseño.

Tras la fase de análisis y definición de requerimientos, se ha llevado a cabo el diseño técnico de ShellTracker. El objetivo principal de esta fase ha sido establecer la arquitectura interna del sistema, seleccionar las tecnologías más adecuadas y definir cómo interactúan entre sí todos los componentes de la aplicación para cumplir con los objetivos funcionales y no funcionales del proyecto.

Tras analizar diversas alternativas tecnológicas, y considerando tanto la naturaleza del problema como los conocimientos adquiridos en el ciclo formativo, se ha optado por desarrollar ShellTracker utilizando React Native con Expo, GraphQL, Firebase, StepZen y Neon(PostgreSQL) como conjunto de tecnologías base.

#### 3.1.- Arquitectura general

ShellTracker se ha diseñado siguiendo una arquitectura de tipo cliente-servidor desacoplada, donde el frontend y el backend se comunican a través de GraphQL, aprovechando las ventajas de este lenguaje de consulta moderno para reducir el consumo de datos, simplificar las operaciones y mejorar la experiencia del usuario.

El flujo principal de funcionamiento de la aplicación es el siguiente:

- El usuario se autentica mediante Firebase (registro o inicio de sesión).
- Se obtiene un token de usuario (uid), que se utiliza para autenticar peticiones posteriores.
- El cliente, construido con React Native, realiza consultas o mutaciones mediante Apollo Client a una API GraphQL alojada en StepZen.
- StepZen actúa como puente entre el frontend y la base de datos Neon, resolviendo las operaciones y enviando la información de vuelta al cliente.
- Esta separación de responsabilidades permite escalar la aplicación fácilmente y realizar cambios en cualquiera de los extremos sin afectar directamente al otro.

#### 3.2.- Tecnologías utilizadas

A continuación, se describen de manera más detallada los principales componentes tecnológicos del proyecto:

Componente	Función
<b>React Native (con Expo)</b>	Framework para el desarrollo de aplicaciones móviles multiplataforma. Permite escribir una única base de código en TypeScript y compilar tanto para Android como iOS.
<b>Firebase Authentication</b>	Servicio de autenticación de usuarios mediante email y contraseña. Ofrece un sistema seguro, gratuito y fácil de integrar con el frontend. Se utiliza exclusivamente para gestionar sesiones.

<b>Apollo Client</b>	Cliente GraphQL utilizado en el frontend. Se encarga de gestionar las peticiones y mutaciones al servidor GraphQL, así como del manejo de caché, estado de red y errores.
<b>GraphQL</b>	Lenguaje de consulta de datos que permite solicitar únicamente los campos necesarios, reduciendo el tamaño de las respuestas y mejorando la eficiencia.
<b>StepZen</b>	Plataforma que expone una API GraphQL a partir de bases de datos SQL. Permite definir esquemas y resolvers para conectar con Neon sin necesidad de un backend personalizado.
<b>Neon (PostgreSQL)</b>	Base de datos relacional sin servidor, escalable y administrada en la nube. Almacena los datos introducidos por los usuarios.
<b>React Navigation</b>	Librería de navegación para React Native que permite gestionar pantallas, rutas y stacks de forma optimizada para móviles.

### 3.3.- Estructura funcional de la aplicación

A continuación, se detallan las principales funcionalidades de la aplicación ShellTracker, mostrando cómo se comunican con el servidor a través de operaciones GraphQL (mutaciones o consultas) utilizando Apollo Client.

Descripción	Registra un nuevo usuario en Firebase.
Método (Firebase)	POST
Parámetros	email, password, returnSecureToken
Método (GraphQL)	POST (mutación GraphQL con Apollo Client)
Headers	Authorization: Bearer {idToken}Content-Type: application/json
Mutación GraphQL	insertUserProfile(uid, name, email, gender, age, height, weight, activity)
Campos guardados	uid, name, email, gender, age, height, weight, activity
Ejemplo de mutación	<pre> mutation {   insertUserProfile(     uid: "CW2rzQGiL1UcCnk1aWOiiXRB3Ea2",     name: "Carlos",     email: "usuario@ejemplo.com",     gender: "Masculino",     age: 28,     height: 175,     weight: 70,     activity: "moderada"   ) {     uid     name     email     gender     age </pre>

	<pre> height weight activity } } </pre>
--	---

Descripción	Permite a un usuario iniciar sesión mediante Firebase Authentication.
Método	POST
Parámetros (JSON)	email, password, returnSecureToken
Uso posterior	Se utiliza idToken para realizar peticiones autenticadas al backend (StepZen).

Descripción	Permite a los usuarios registrar alimentos consumidos en la base de datos.
Método	POST (mutación GraphQL con Apollo Client)
Mutación GraphQL	insertFoodEntry(uid, label, calories, brand, date)
Campos guardados	uid, label (alimento), calories, brand, date (timestamp)

Descripción	Permite modificar los datos del perfil del usuario ya registrado.
Método	POST (mutación GraphQL con Apollo Client)
Mutación GraphQL	updateUserProfile(uid, name, gender, age, height, weight, activity)
Campos actualizables	name, age, height, weight, activity
Requiere autenticación	Sí (mediante idToken del usuario)
Ejemplo de respuesta del servidor	<pre>{   "data": {     "updateUserProfile": {       "name": "Carlos Actualizado",       "age": 29,       "height": 180,       "weight": 72,       "activity": "ligera"     }   } }</pre>

### 3.4.- Bases de datos

La base de datos de FireBase encargada de almacenar los usuarios tendrá una única tabla:

Campo	Tipo de dato	Descripción
uid	string	Identificador único de Firebase Auth
name	string	Nombre del usuario
email	string	Correo electrónico
gender	string	Género del usuario
age	number	Edad en años
height	number	Altura en centímetros
weight	number	Peso en kilogramos
activity	string	Nivel de actividad física (ej: "muy_activa")

Y la base de datos encargada de gestionar los alimentos registrados por los usuarios tendrá otra única tabla:

Campo	Tipo de dato	Descripción
id	SERIAL PRIMARY KEY	Identificador único (auto incremental)
user_id	VARCHAR(255)	Identificador del usuario (Firebase uid)
food_id	VARCHAR(255)	Identificador del alimento
label	VARCHAR(255)	Nombre o etiqueta del alimento
kcal	INTEGER	Calorías del alimento
created_at	TIMESTAMP	Fecha y hora de registro

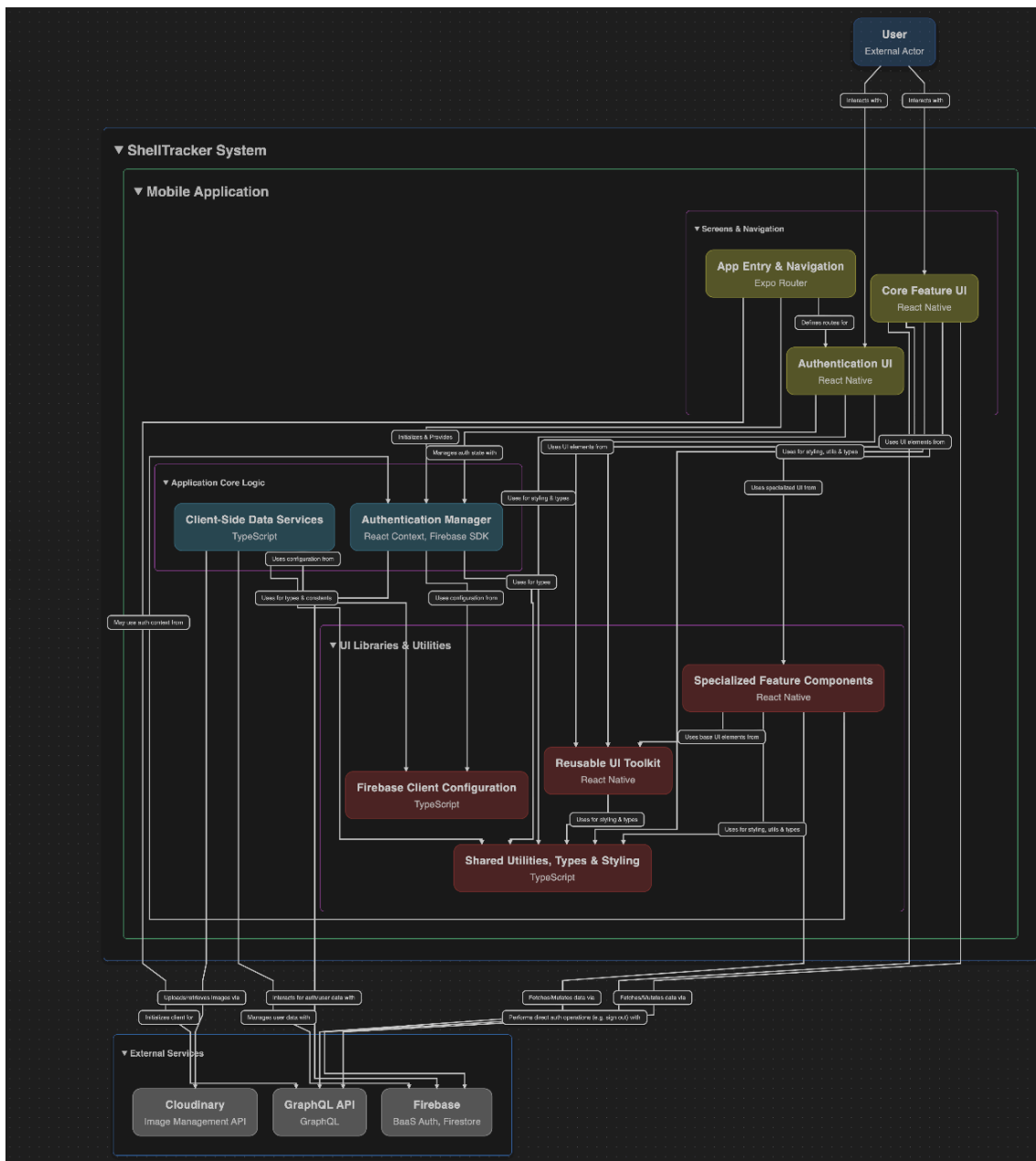
El script necesario para generar la tabla lo observamos a continuación:

```
CREATE TABLE food_log (  
    id SERIAL PRIMARY KEY,  
    user_id VARCHAR(255) NOT NULL,  
    food_id VARCHAR(255) NOT NULL,  
    label VARCHAR(255) NOT NULL,  
    kcal INTEGER NOT NULL,  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```



### 3.5.- Flujos de trabajo.

A continuación se representa la navegabilidad entre pantallas, así como el flujo de datos entre ellas:



Para observarlo con más detalle, dejo anclado un enlace a un archivo [drawio](#) donde podremos verlo con mejor calidad.

### 3.6.- Módulos de la aplicación.

En esta sección se describen los módulos de la aplicación, con una descripción por pantallas:

- El acceso a la aplicación se puede realizar de dos formas distintas: registro e inicio de sesión.
- Registro de usuarios: Esta pantalla validará y recogerá los datos introducidos por el usuario en el formulario, y en caso de introducir los datos correctamente, creará el usuario en la base de datos y le dará acceso a la aplicación, de manera contrario si los introduce incorrectamente o no los introduce todos los necesario, no se creará el usuario. Los campos que introducirá ya se han detallado previamente.
- Inicio de sesión de usuarios existentes: Los usuarios existentes podrán introducir sus datos directamente en esta pantalla para iniciar sesión, aunque este paso no será necesario si ya se encuentran logueados porque entonces serán redirigidos a la pantalla principal.
- Búsqueda de alimentos: En esta pantalla el usuario podrá buscar alimentos, comidas o bebidas por su nombre o marca, se desplegarán en forma de lista donde el usuario podrá elegir una opción tocando el símbolo de “+” que se mostrará en cada tarjeta correspondiente a un alimento.
- Home Screen: en la pantalla principal de la aplicación al principio de cada día podremos ver una tarjeta con las calorías recomendadas y consumidas dependiendo de cada usuario y los registros que haya realizado, además conforma vaya registrando los alimentos, se irán mostrando en forma de lista.
- Perfil: En esta pantalla se observará el perfil del usuario con dos opciones diferentes para cerrar sesión y para modificar su perfil, si pulsamos sobre el botón de cerrar sesión podremos observar como sale un warning confirmando el cierre de la sesión; si por el contrario damos a editar perfil, tendremos la opción de cambiar la imagen y algunos datos básicos del usuario críticos para calcular su consumos calórico, es aquí donde podremos gestionar el progreso.

#### 4.- Resultado.

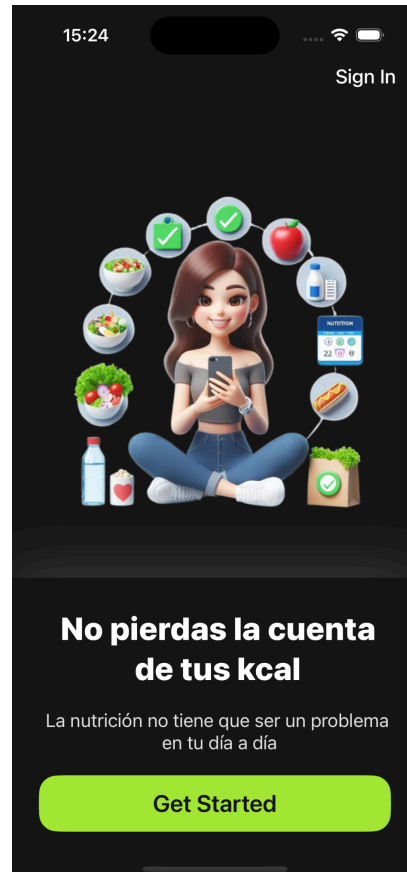
Una vez se ha realizado la implementación y desarrollo de la aplicación, se debe evaluar el proceso y los resultados.

En cuanto a las funcionalidades definidas y deseadas inicialmente en la fase de diseño técnico se han conseguido implementar todas ellas, aunque durante el proceso he percibido la necesidad de la propia aplicación en incorporar o mejorar algunos componentes o funcionalidades como eliminar alimentos que el usuario no ha ingerido pero que ha agregado sin querer o como ajustar las cantidades que el usuario ha ingerido y no solo basarnos en las que nos da API pública.

Como ya he mencionado anteriormente se ha intentado realizar una interfaz intuitiva, que exija la menor inversión en cantidad de tiempo posible, minimalista, usando iconos conocidos por todos en las diversas áreas de la aplicación.

##### 4.1.- Pantallas de la aplicación

A continuación se muestran las diferentes pantallas que posee la aplicación:



15:24

**¡Hola!** 😊

Crea una cuenta para continuar

Nombre

Correo electrónico

Contraseña

Edad

Selecciona género

Altura (cm)

Peso (kg)

Nivel de actividad

**Registrarse**

15:24

**¡Hola de nuevo!** 😊

Inicia sesión para continuar

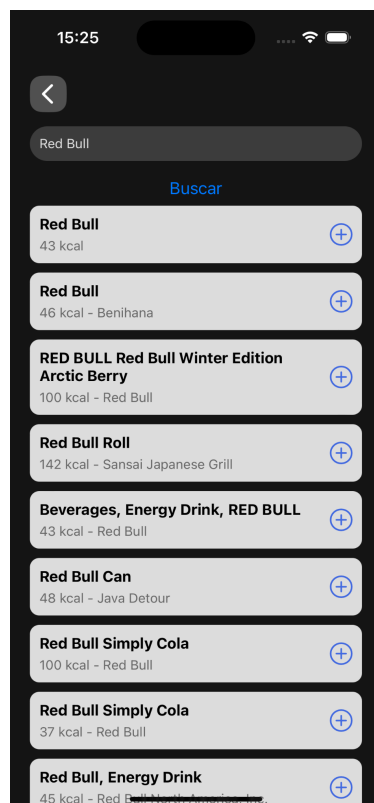
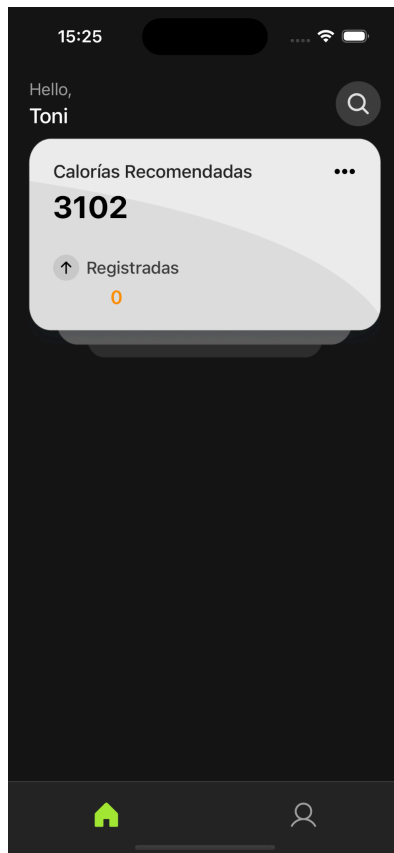
Correo electrónico

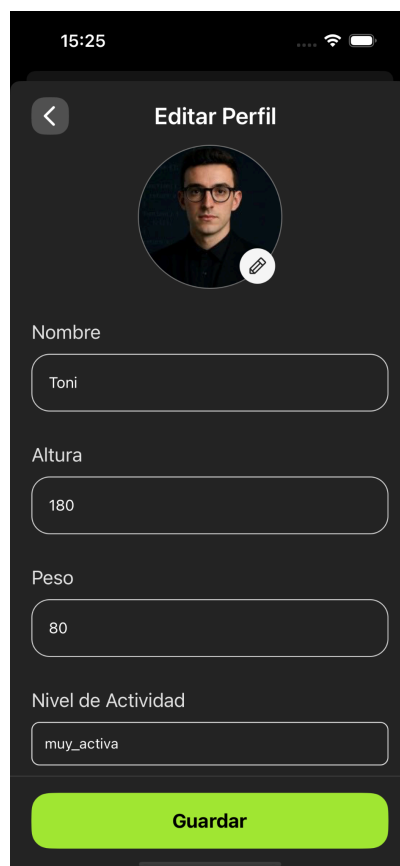
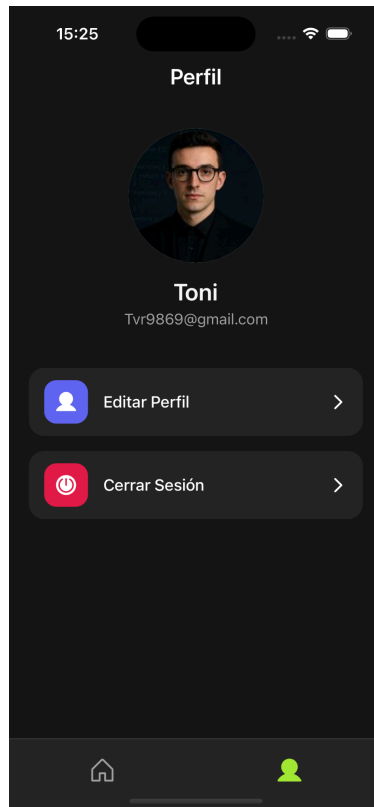
Contraseña

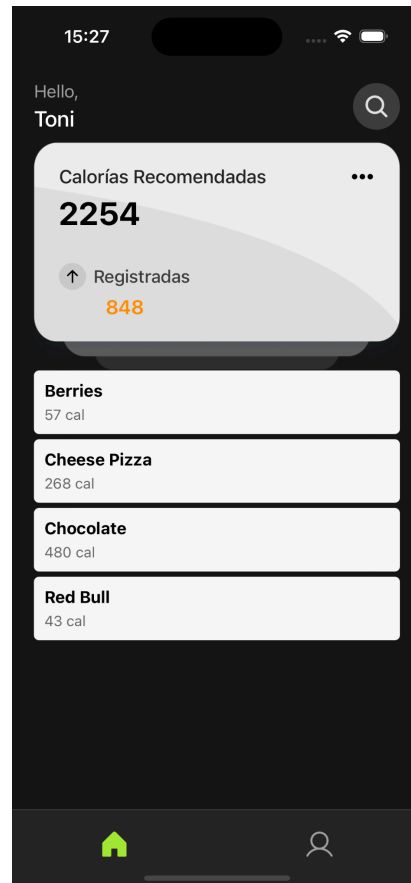
¿Has olvidado la contraseña?

**Iniciar sesión**

¿No tienes cuenta? [Regístrate](#)







#### 4.2.- Implementación del sistema de consultas a la API pública

Sin lugar a dudas merece una sección aparte en la memoria de ShellTracker, ha sido una de las soluciones que he incorporado más innovadoras y que más pueden generar curiosidad, pues al ser algo relativamente innovar, no demasiada gente conoce cómo funciona.

En el desarrollo de ShellTracker, una parte esencial del sistema fue la integración con una API pública para ofrecer a los usuarios información nutricional detallada sobre los alimentos. Para ello, se utilizó la API de Edamam, especializada en datos nutricionales y composición de ingredientes, que permite acceder a información precisa como calorías, macronutrientes, índice glucémico, y más.

La API de Edamam funciona originalmente sobre REST, pero para facilitar su integración con nuestro backend y optimizar las consultas desde el frontend, se utilizó StepZen, una herramienta que permite convertir APIs REST en endpoints GraphQL de manera sencilla. Esto permitió definir esquemas personalizados y realizar consultas más específicas, reduciendo la sobrecarga de datos innecesarios.

Mediante StepZen, se configuró un esquema GraphQL que consumía los endpoints de Edamam, con autenticación mediante claves API. Esto nos permitió exponer un único endpoint GraphQL que centraliza la lógica de consulta y la transforma en una respuesta estructurada y adaptada a las necesidades del frontend.

Aunque los datos provenientes de Edamam se consumen de forma dinámica, se diseñó un sistema de almacenamiento en PostgreSQL para guardar ciertas consultas frecuentes o personalizadas realizadas por los usuarios, como sus registros nutricionales recientes. Esto no solo mejora el rendimiento,

evitando llamadas innecesarias a la API externa, sino que también permite ofrecer una experiencia más personalizada.

Para gestionar esta integración, se utilizó un backend que se comunicaba tanto con StepZen como con la base de datos PostgreSQL. Las consultas se gestionaban de forma asincrónica, y se añadieron controles de errores y límites de uso para cumplir con las restricciones de la API gratuita de Edamam.

Gracias a esta arquitectura, ShellTracker permite al usuario buscar alimentos por nombre, visualizar información nutricional precisa, y guardar los resultados en su perfil. La integración con Edamam mediante StepZen y GraphQL supuso una solución eficiente y escalable, que mejoró tanto el rendimiento como la experiencia de usuario final.

#### **4.3.- Implementación del sistema de autenticación con Firebase.**

Para gestionar de forma segura el acceso de los usuarios a ShellTracker, se implementó un sistema de autenticación utilizando Firebase Authentication, una solución de Google que ofrece servicios de autenticación sencillos, escalables y seguros, ideal para proyectos que requieren login sin desarrollar un sistema de autenticación personalizado desde cero.

El objetivo principal fue permitir el registro y acceso mediante correo electrónico y contraseña, manteniendo una experiencia fluida y segura para el usuario.

Entre las ventajas claras que podemos encontrar están:

- Sistema gratuito y fácil de integrar.
- Amplio soporte para distintos métodos de autenticación (email, Google, Facebook, etc.).
- Gestión automática de tokens y sesiones.
- Buenas prácticas de seguridad implementadas por defecto.

Pasos para la implementación:

1. Crear un proyecto en Firebase
2. Acceder a <https://console.firebase.google.com>
3. Crear un nuevo proyecto y configurarlo con el nombre "ShellTracker".
4. Registrar la app
5. Desde la consola de Firebase, registrar la app web (Web App) para obtener las credenciales de conexión (apiKey, authDomain, etc.).
6. 3. Instalar el SDK de Firebase
7. Instalamos firebase en el proyecto con "npm install firebase"
8. Configuramos firebase en el archivo correspondiente
9. Y gestionamos a través de contextos en el código los estados de la sesión del usuario.



```
config > TS firebase.ts > [⌕] firebaseConfig > 🔗 appId
1
2 import { initializeApp } from "firebase/app";
3 import { initializeAuth, getReactNativePersistence } from "firebase/auth";
4 import AsyncStorage from '@react-native-async-storage/async-storage';
5 import { getFirestore } from "firebase/firestore";
6
7
8 const firebaseConfig = {
9   apiKey: [REDACTED]
10  authDomain: [REDACTED]
11  projectId: [REDACTED]
12  storageBucket: [REDACTED]
13  messagingSenderId: [REDACTED]
14  appId: [REDACTED]
15 };
16
17
18 const app = initializeApp(firebaseConfig);
19
20
21 export const auth = initializeAuth(app, {
22   persistence: getReactNativePersistence(AsyncStorage),
23 });
24
25
26 export const firestore = getFirestore(app);
```

## **5.- Conclusiones.**

La fase de implementación es la que más tiempo ha conllevado del proyecto.

### **5.1.- Problemas encontrados.**

Sin lugar a dudas durante el desarrollo de ShellTracker se han tenido que resolver distintos problemas, react-native ha supuesto una experiencia enriquecedora para el usuario final de la aplicación pero quizá no tanto para el desarrollo, puesto que la documentación sobre la última versión es realmente escasa y nada compatible con versiones anteriores.

Uno de los problemas iniciales a la hora de implementar la aplicación fue encontrar una API que me permitiese realizar una búsqueda de los alimentos y que dentro de esos alimentos, detallase los elementos que necesitaba para la aplicación.

Otro de los grandes retos de esta aplicación era la incorporación de los registros de los alimentos por parte de los usuarios y que solo durasen 24 horas, ya que consiste en un seguimiento nutritivo diario, las fechas y su relación con la caché de la aplicación han supuesto un gran inconveniente a la hora de registrar estos alimentos.

Finalmente, y con gran importancia, lo que para mí ha sido lo más complicado ha sido la organización del tiempo, pues después de una jornada laboral diaria de 10 horas también en el sector de la programación, pocas energías quedan para continuar con el desarrollo del proyecto, pero finalmente, lo he conseguido.

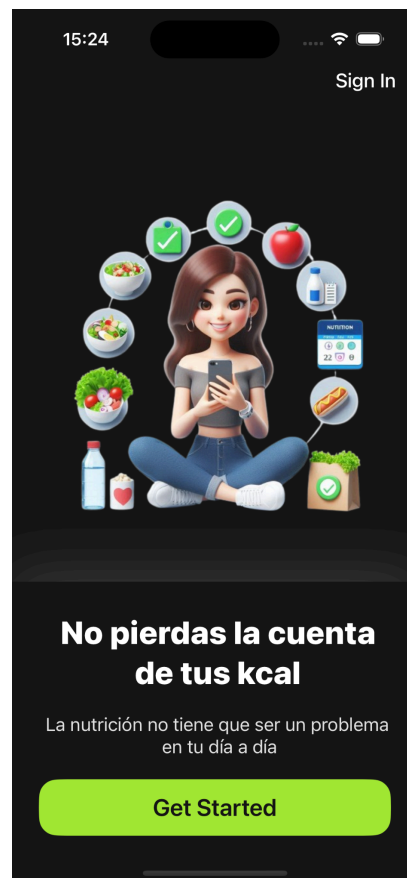
## **6.- Manual del usuario.**

### **6.1.- ¿Qué es ShellTracker?**

ShellTracker es una aplicación que nos permite de manera muy intuitiva, minimalista, fácil y rápida llevar un seguimiento calórico sobre los alimentos que ingerimos según nuestras necesidades u objetivos. De esta manera, ofrece la posibilidad de combatir un problema general y creciente en la sociedad actual.

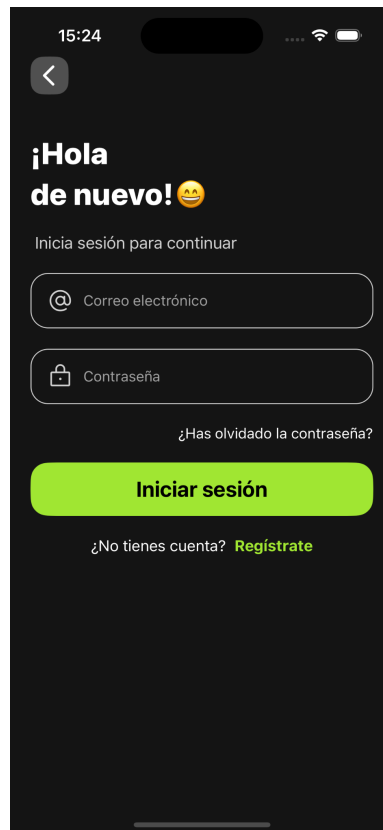
### **6.2.- Acceder a la aplicación**

ShellTracker permite que te conectes si ya eres un usuario registrado o que crees una cuenta si eres nuevo. Nada más acceder a la aplicación, tanto si tienes usuario como si no, se mostrará la pantalla principal:

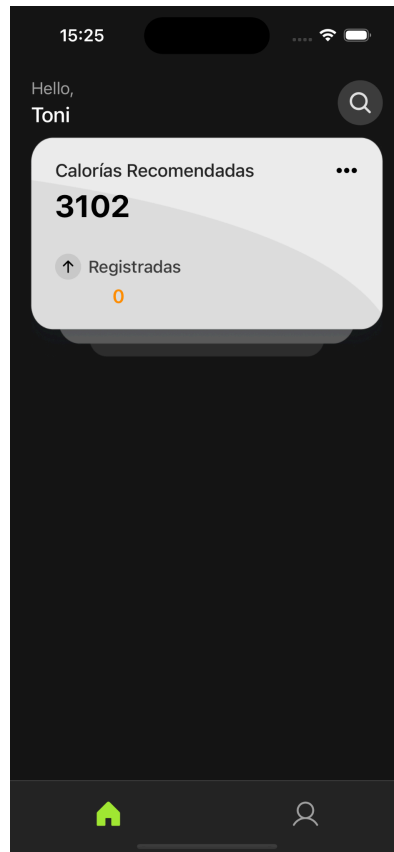


En esta pantalla observamos dos botones claramente identificativos, para registrarnos en la aplicación y para iniciar sesión.

Si presionamos el botón verde de abajo para registrarnos observaremos el formulario que tenemos en la imagen, indispensable para saber las características y datos del usuario para aconsejar un consumo calórico, una vez presionemos el botón de Registrarse y se realicen las validaciones del formulario el usuario se habrá registrado en la base de datos y nos dirigirá a la pantalla principal.

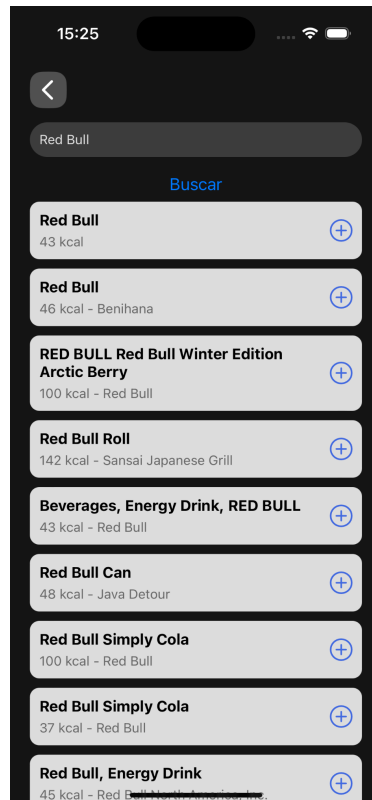


Si por el contrario, el usuario ya tiene una cuenta y lo que quiere es iniciar sesión, bastará con introducir su correo electrónico y su contraseña, tras las validaciones también se le redirigirá a la pantalla principal de la aplicación.

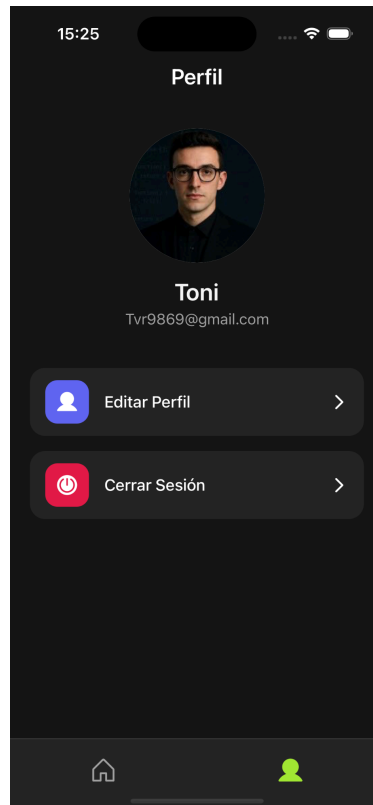


En la pantalla principal de la aplicación podemos observar un texto en la parte superior izquierda donde se recupera el nombre del usuario y un breve saludo, en la parte superior contraria al saludo tenemos el botón con forma del icono reconocido de la lupa para navegar a la pantalla de búsqueda de alimentos.

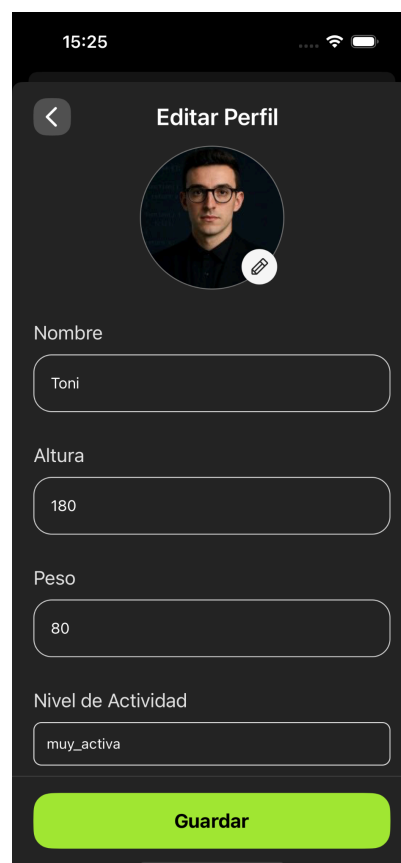
Justo abajo observamos la tarjeta con las calorías recomendadas y registradas por parte del usuario, y abajo del todo tenemos una pequeña barra de navegación donde encontramos tanto el icono de la pantalla principal como el del perfil.



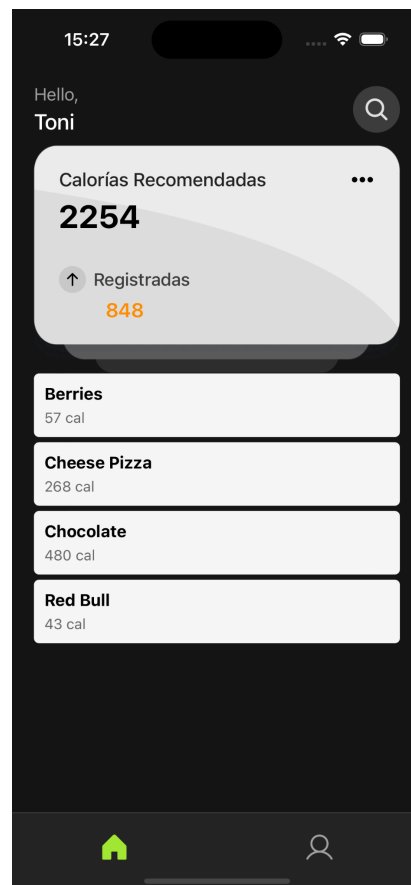
En la pantalla de búsqueda de alimentos observamos una barra donde podremos introducir un texto para buscar el alimento que deseamos registrar, cada uno de los alimentos que coincida con la request que mandamos a la API de Edamam, aparecerán en un componente en forma de tarjeta, con el nombre del alimento, las calorías de este y la marca si está registrada en la API. Además, observamos un icono con forma de “+” donde si lo pulsamos, registrará el alimento en el usuario, es el botón que genera el registro en la base de datos PostgreSQL.



Por otro lado la pantalla del perfil se compone de una imagen de usuario personalizable y dos botones para modificar el perfil o cerrar la sesión del usuario.



Si presionamos sobre el botón de editar perfil aparece una subpantalla donde nos permite modificar la imagen del usuario a través de la cámara o la galería del dispositivo, justo debajo tenemos un pequeño formulario donde nos permite cambiar el nombre del usuario, la altura, el peso o el nivel de actividad física; si modificamos estos datos, se actualizarán en tiempo real en la base de datos y en la pantalla principal.



Conforme se van agregando alimentos los podemos observar en la pantalla principal bajo el mismo componente con el que se muestran en la pantalla de búsqueda de alimentos.



## 7.- Bibliografía.

- <https://stackoverflow.com/questions>
- <https://developer.edamam.com/login>
- <https://chatgpt.com/>
- <https://firebase.google.com>
- <https://console.neon.tech/app>
- <https://reactnative.dev/>
- <https://www.npmjs.com/>