**Marfol**









CICLO FORMATIVO DE GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Índice de contenidos

[Agradecimientos 3](#_Toc136614718)

[Resumen 4](#_Toc136614719)

[Licencia 5](#_Toc136614720)

[1.Introducción 6](#_Toc136614721)

[2.Necesidades del sector productivo 7](#_Toc136614722)

[2.1 Análisis de la situación actual 7](#_Toc136614723)

[2.2 Necesidades del cliente y oportunidades del negocio 7](#_Toc136614724)

[2.3 Marfol – Simplificación de la división de cuenta 7](#_Toc136614725)

[3.Diseño del proyecto 8](#_Toc136614726)

[3.1 Fases del proyecto 8](#_Toc136614727)

[3.1.1 Análisis 8](#_Toc136614728)

[3.1.2 Diseño 10](#_Toc136614729)

[Colores 10](#_Toc136614730)

[Base de datos 14](#_Toc136614731)

[Arquitectura firestore 15](#_Toc136614732)

[Arquitectura storage 19](#_Toc136614733)

[3.1.3 Implementación 20](#_Toc136614734)

[Inicio 20](#_Toc136614735)

[Comprobar si el usuario está logueado 20](#_Toc136614736)

[Index 21](#_Toc136614737)

[Autenticación 21](#_Toc136614738)

[Contraseña olvidada 23](#_Toc136614739)

[Registro 24](#_Toc136614740)

[Gestión de usuario 26](#_Toc136614741)

[Home 27](#_Toc136614742)

[Editar datos del usuario 28](#_Toc136614743)

[Gestión de datos 30](#_Toc136614744)

[Personas 31](#_Toc136614745)

[Restaurantes 38](#_Toc136614746)

[Platos 42](#_Toc136614747)

# Agradecimientos

**Kayler y Javier**

Presentamos a **Luis Daniel Casado Guimaraes (DAW)**, un talentoso artista digital que ha dejado su huella en nuestro proyecto de manera extraordinaria.

Daniel ha sido un pilar fundamental en la creación de imágenes y animaciones iniciales para nuestra plataforma. Su habilidad creativa y su enfoque meticuloso han llevado nuestras ideas a un nivel completamente nuevo y emocionante.

Las imágenes que Daniel ha creado son verdaderamente impresionantes, capturando la esencia de nuestra aplicación con maestría. Su atención al detalle y su capacidad para transmitir emociones a través de sus creaciones han dejado a nuestros usuarios asombrados y cautivados desde el primer momento en que interactúan con nuestra interfaz.

Pero eso no es todo, Daniel también ha dotado a nuestra plataforma de una animación inicial que es simplemente espectacular. Su talento para dar vida a elementos estáticos es sorprendente, y cada vez que los usuarios inician nuestra aplicación, son recibidos por una experiencia visualmente impactante gracias a su trabajo.

El aporte creativo de Daniel ha sido invaluable para el éxito de nuestro proyecto, añadiendo un toque de magia a cada elemento visual. Su dedicación y pasión por el arte digital han elevado nuestra plataforma a nuevas alturas y han dejado una impresión duradera en todos aquellos que la disfrutan.

Gracias danieh.

# Resumen

El proyecto Marfol es desarrollado por confugiradores, una organización no especificada que se dedica al diseño y desarrollo de aplicaciones móviles.

Marfol tiene como objetivo abordar las necesidades de simplificar y agilizar el proceso de dividir la cuenta después de una comida en un restaurante. Busca eliminar la confusión y complicaciones que surgen al compartir los gastos, brindando una herramienta intuitiva y eficiente para calcular los costos individuales de manera equitativa.

La posible demanda de Marfol son todos aquellos usuarios que frecuentan restaurantes y desean una solución práctica para dividir la cuenta de forma justa. Este servicio puede ser utilizado por individuos, grupos de amigos, familias u otros comensales que deseen una manera más fácil y precisa de realizar la división de gastos.

Marfol es una aplicación móvil que simplifica el proceso de dividir la cuenta después de una comida en un restaurante. La aplicación permite a los comensales ingresar fácilmente los platos y bebidas que han ordenado, asignando automáticamente los costos a cada individuo según los elementos seleccionados. Utiliza un algoritmo inteligente para calcular la división equitativa, teniendo en cuenta las preferencias de los usuarios. Además, ofrece características adicionales como la posibilidad de añadir elementos personalizados y compartir los resultados de la división de la cuenta a través de mensajes de texto o correo electrónico. Marfol también guarda un historial completo de comidas previas para facilitar el seguimiento y la documentación de gastos compartidos.

En conclusión, Marfol proporciona una solución innovadora y conveniente para simplificar la división de la cuenta en restaurantes, satisfaciendo las necesidades de los usuarios al eliminar la confusión y complicaciones asociadas con los gastos compartidos.

# Licencia

**Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by-sa/3.0/es/ o envie una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.**

# 1.Introducción

Este documento se refiere a la realización del módulo de Proyecto del CFGS en Desarrollo de Aplicaciones Multiplataforma (DAM). El módulo de Proyecto complementa la formación establecida para los demás módulos profesionales que forman parte del título, enfocándose en el análisis del entorno, diseño del proyecto y organización de la implementación.

Marfol es una aplicación móvil diseñada para simplificar la división de la cuenta después de comer en un restaurante. Con su interfaz intuitiva, los comensales pueden ingresar fácilmente los participantes, platos, bebidas, y la aplicación se encarga de calcular los costos individuales de manera equitativa. Además, Marfol ofrece ajustes personalizados, como considerar situaciones especiales como compartir, y muestra resultados precisos para que cada persona pague su parte exacta.

En resumen, Marfol es la solución definitiva para dividir la cuenta en restaurantes. Con su funcionalidad eficiente, eliminando las complicaciones matemáticas y las discusiones interminables, Marfol simplifica el proceso y garantiza una experiencia sin complicaciones para todos los comensales. Ya no es necesario preocuparse por los gastos compartidos, ya que Marfol se encarga de todo de manera rápida, justa y sencilla.



# 2.Necesidades del sector productivo

## 2.1 Análisis de la situación actual

En este contexto, se analiza la situación actual relacionada con la división de la cuenta después de una comida en un restaurante. Se identifican las dificultades y complicaciones que suelen surgir al compartir los gastos, como las discusiones sobre quién consumió qué y el cálculo manual de los costos individuales. Este análisis destaca la necesidad de simplificar y agilizar este proceso.

## 2.2 Necesidades del cliente y oportunidades del negocio

Se identifican las necesidades de los clientes, que incluyen eliminar la confusión y complicaciones al dividir la cuenta, así como simplificar el proceso y asegurar una división equitativa. Estas necesidades crean una oportunidad de negocio para desarrollar una aplicación móvil como Marfol, que brinde una solución intuitiva y eficiente para calcular los costos individuales de manera justa y sin complicaciones.

## 2.3 Marfol – Simplificación de la división de cuenta

El proyecto Marfol propone una solución innovadora para simplificar la división de la cuenta después de comer en un restaurante. La aplicación móvil Marfol ofrece una interfaz amigable que permite a los comensales ingresar los platos y bebidas que han ordenado, asignando automáticamente los costos a cada individuo. Utilizando un algoritmo inteligente, Marfol calcula una división equitativa, considerando las preferencias de los usuarios y permitiendo ajustes personalizados. Además, Marfol ofrece funciones de personalización, compartición de resultados y registro de comidas anteriores, brindando una experiencia completa y satisfactoria para los usuarios. Con Marfol, la división de la cuenta se convierte en una tarea rápida y sin complicaciones, abordando las necesidades y demandas de los clientes en este sector.

# 3.Diseño del proyecto

## 3.1 Fases del proyecto

### 3.1.1 Análisis

En esta fase se establecerán los requisitos del proyecto, definiendo tanto las funcionalidades que debe incluir como las características y cualidades que deben cumplirse. En el caso de Marfol, se trata de una aplicación móvil para la división de la cuenta en restaurantes, por lo que se identificarán los requisitos específicos para su desarrollo. Estos requisitos se clasificarán en "Requisitos Funcionales", que describen las funcionalidades necesarias, y "Requisitos No Funcionales", que se refieren a las características y propiedades deseadas de la aplicación.

**Requisitos funcionales**

Marfol debe permitir a los usuarios registrarse y gestionar los nombres de los comensales. Los usuarios deben poder ingresar fácilmente los elementos de la orden y la aplicación debe realizar cálculos automáticos para dividir la cuenta de manera equitativa. Además, se deben proporcionar ajustes personalizados, una visualización clara de los resultados y la opción de compartirlos. Marfol también debe almacenar un registro de comidas anteriores para facilitar el volver a utilizar los mismos datos a futuro.

**Requisitos no funcionales**

Los requisitos no funcionales para Marfol incluyen la necesidad de una interfaz de usuario intuitiva y fácil de usar, un rendimiento rápido y preciso en el cálculo de la división de la cuenta, la confiabilidad y disponibilidad constante de la aplicación, la seguridad de los datos, la compatibilidad con diferentes dispositivos, la capacidad de escalar para manejar un crecimiento en usuarios, la facilidad de mantenimiento y actualización, y proporcionar una experiencia de usuario agradable y atractiva.

Requisitos Mínimos de Android:

Para garantizar un rendimiento óptimo y una experiencia fluida en la aplicación Marfol, se recomienda que los dispositivos Android cumplan con los siguientes requisitos mínimos:

|  |  |
| --- | --- |
| Requisito | Especificación |
| Velocidad CPU | Procesador Quad Core de al menos 1.2 GHz o superior |
| RAM (GB) | Mínimo 2 GB de RAM recomendado |
| Cámara principal | Resolución mínima de la cámara principal de 3.0 MP (megapíxeles) |
| Almacenamiento Interno (GB) | Mínimo 1 GB de almacenamiento interno disponible |
| Almacenamiento Externo | Compatibilidad con tarjetas MicroSD para expandir la memoria (se recomienda capacidad de hasta 32 GB) |
| Localización | Dispositivo compatible con GPS (Sistema de Posicionamiento Global) para utilizar la geolocalización |
| Versión Android | Compatible con Android 7.X o versiones superiores (recomendado Android 7 o posterior) |
| Conexión | Requiere conexión Wi-Fi y 4G (servicio de datos) para acceder a los servicios utilizados por la aplicación |

Requisitos Generales:

Además de los requisitos específicos de Android, se recomienda tener en cuenta los siguientes requisitos generales para garantizar un óptimo rendimiento y funcionamiento de la aplicación Marfol:

* Conectividad a Internet: Se requiere una conexión estable y confiable para acceder a todas las funciones de Marfol.
* Permisos de la aplicación: Se recomienda otorgar los permisos solicitados para el correcto funcionamiento de Marfol.
* Actualizaciones de la aplicación: Se recomienda mantener Marfol actualizado para tener acceso a las últimas mejoras y correcciones.
* Espacio de almacenamiento: Es necesario disponer de suficiente espacio en el dispositivo para almacenar datos de Marfol.

Estos requisitos generales son importantes para garantizar un uso óptimo de la aplicación Marfol y asegurar una experiencia positiva para los usuarios. Asegúrate de cumplir con estos requisitos y recomendaciones para obtener el máximo rendimiento y funcionalidad de la aplicación.

### 3.1.2 Diseño

En esta sección, presentamos el diseño visual de la aplicación Marfol, incluyendo los colores utilizados y la intención detrás de su elección.

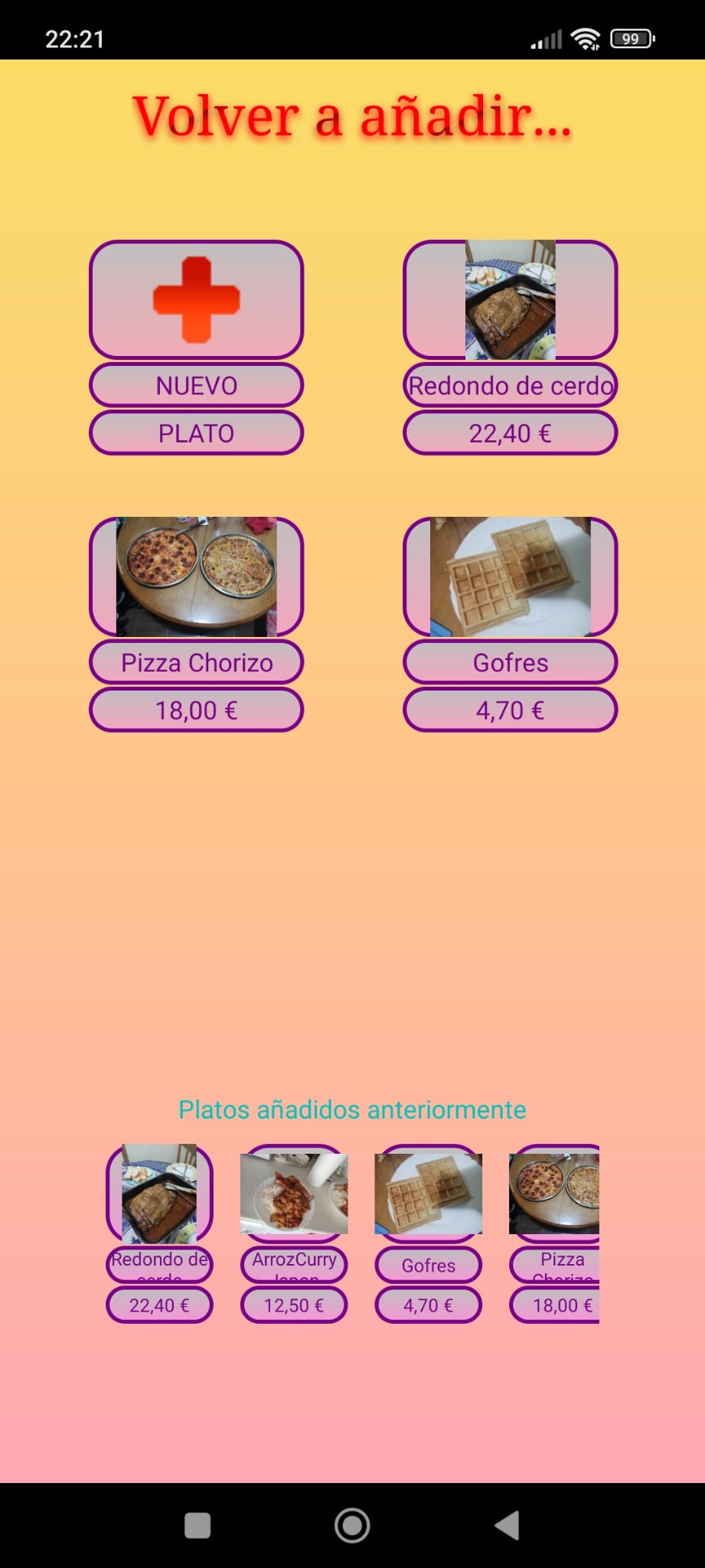
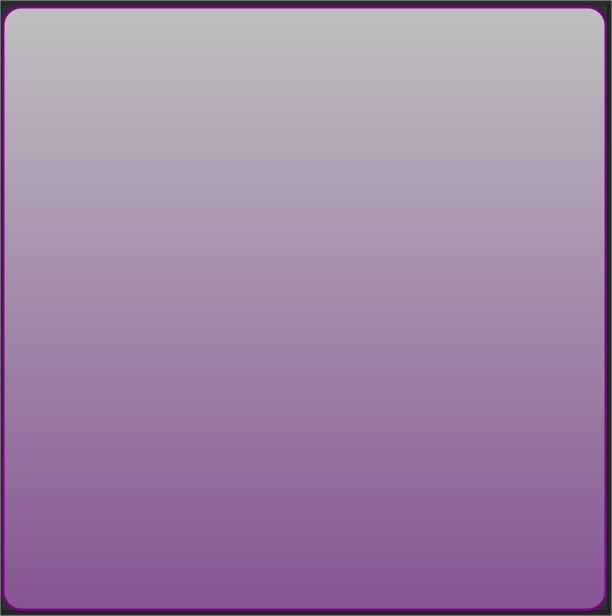
#### Colores

Base:

****

* #FFA8B4
* #FFC68C
* #FBDD65

Estos colores base se caracterizan por ser alegres, vivos y cálidos. Han sido seleccionados para transmitir una sensación de alegría y felicidad, ya que la aplicación Marfol se utiliza en momentos de compartir comidas y disfrutar de momentos especiales.

****Platos:

* #7AE77EFF
* #bfbfbf

Los colores seleccionados para los platos se encuentran en una gama de tonos vibrantes y representan la diversidad y la variedad de opciones gastronómicas que ofrece la aplicación.

El inicio del color más claro se desvanece hacia un tono más intenso, lo que aporta un aspecto visual atractivo y moderno a los elementos relacionados con los platos.

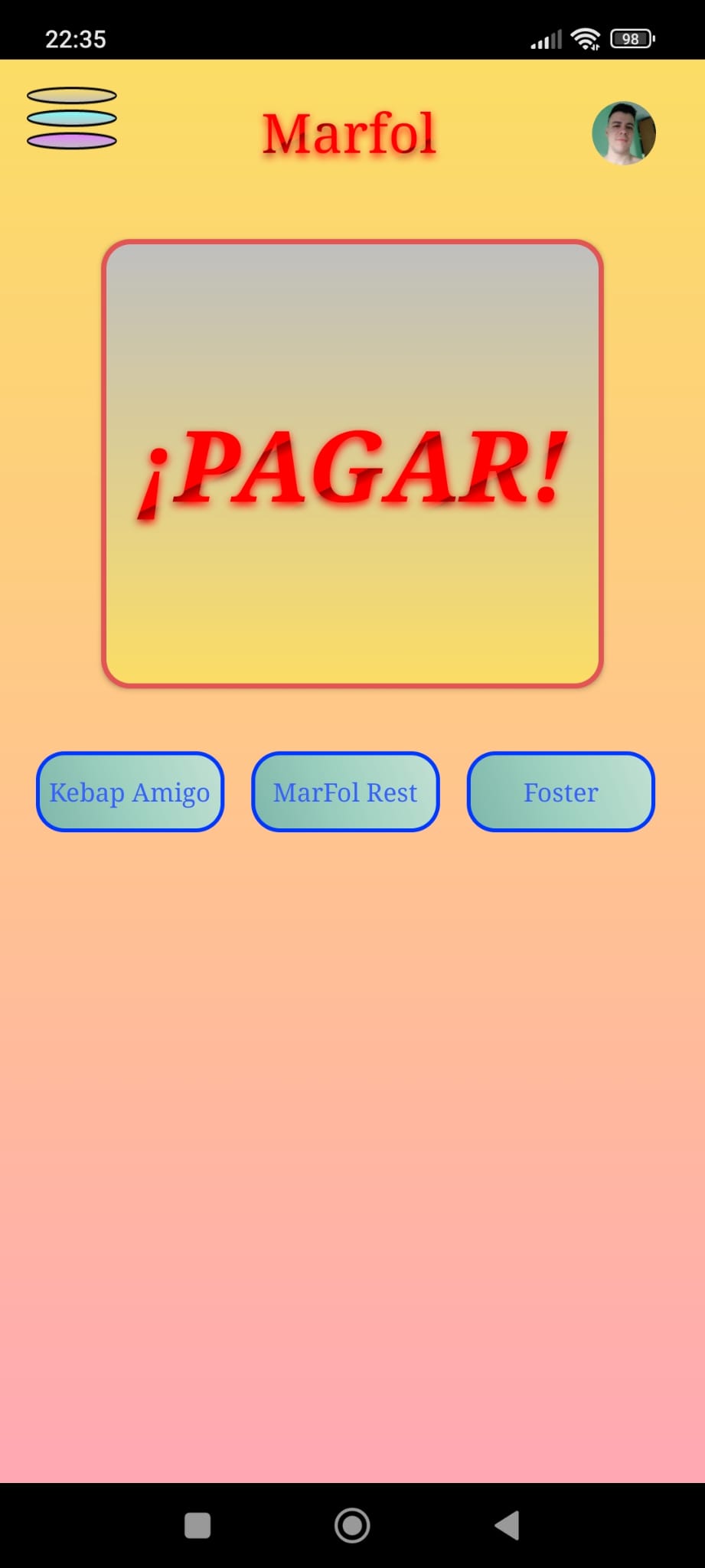
Personas:

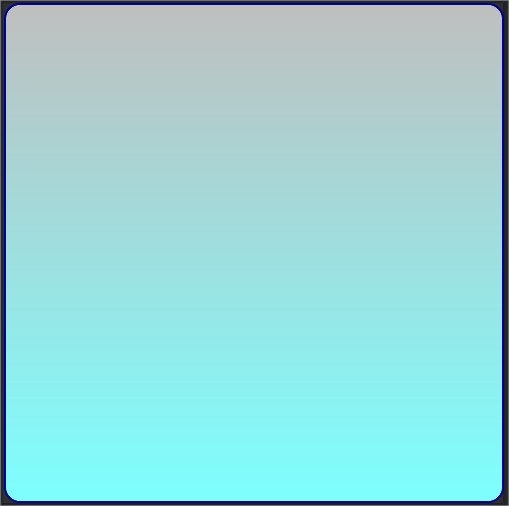
****

* + #FBDD65
  + #bfbfbf

Los colores utilizados para representar a las personas en la aplicación están basados en tonos cálidos y amigables. El color de inicio, un tono de amarillo dorado, simboliza la conexión y la interacción entre los comensales.

El desvanecimiento hacia un tono más claro en el final del color crea un efecto suave y armonioso.

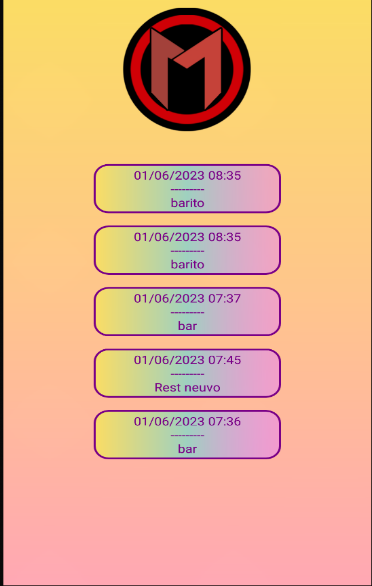
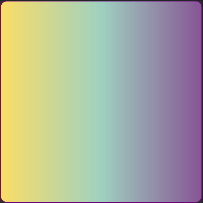
****Restaurantes:

****

* + #7EFFFF
  + #bfbfbf

Los colores elegidos para los restaurantes se basan en tonos azules que evocan una sensación de frescura, tranquilidad y profesionalidad.

El degradado utilizado proporciona un aspecto moderno y agradable a la vista.

****Historial:

* #FBDD65
* #9FD1C0
* #7AE77EFF

Los tres colores principales de la aplicación se usan para el historial, la fusión de colores representa todo el proceso de utilizar Marfol, desde añadir una persona hasta el restaurante con sus platos. Con esta paleta de colores, buscamos transmitir una sensación de armonía y conexión.

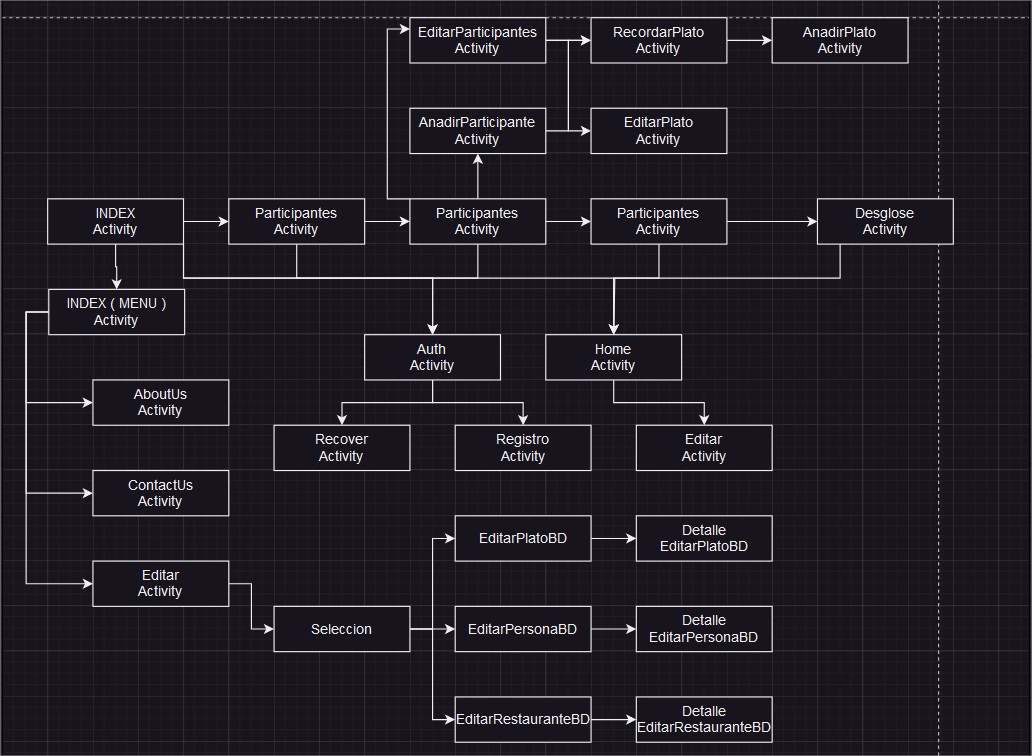


Los colores utilizados fueron seleccionados pensando en la naturaleza de la aplicación y su objetivo de proporcionar una experiencia agradable y positiva a los usuarios. Estos colores ayudan a crear una interfaz vibrante, alegre y fácil de utilizar, lo que resulta en una experiencia visualmente atractiva.

Además, mejoran la usabilidad de la aplicación al permitir que los usuarios identifiquen fácilmente los elementos en función de los colores elegidos.

Cliclo de vida:

//cambiar esta imagen



#### Base de datos

En este proyecto, se ha decidido utilizar Firebase como la plataforma principal para el almacenamiento de datos y la autenticación de usuarios en la nube. Firebase ofrece una amplia gama de servicios, y en este caso, se han utilizado Authentication, Firestore y Storage para satisfacer las necesidades del proyecto.

Firestore es una base de datos NoSQL en tiempo real alojada en la nube que permite almacenar y sincronizar datos de manera eficiente. En el proyecto, Firestore se ha utilizado para gestionar la autenticación de usuarios y almacenar información relacionada con ellos.

* Firebase Authentication proporciona un sistema seguro y confiable para autenticar usuarios en la aplicación. A través de Authentication, se ha implementado un sistema de registro e inicio de sesión que permite a los usuarios crear una cuenta utilizando su dirección de correo electrónico y una contraseña segura.
* Firebase Firestore se utiliza para almacenar información relevante del usuario, como su nombre, dirección de correo electrónico y otros detalles personalizados. Cada usuario tiene su propio documento en Firestore, donde se guardan estos datos. Esto permite un acceso rápido y eficiente a la información del usuario y facilita su gestión y personalización.
* Firebase Storage es un servicio en la nube que permite almacenar y recuperar archivos, como imágenes o cualquier otro tipo de archivo, de manera segura y eficiente. En el proyecto, Storage se ha utilizado para gestionar el almacenamiento de imágenes relacionadas con los usuarios

Al registrarse o actualizar su perfil, los usuarios pueden cargar su imagen y esta se almacenará de forma segura en Storage. La URL de la imagen se guarda en Firestore, lo que permite recuperar y mostrar la imagen de perfil en la interfaz de usuario correspondiente, lo mismo se aplicará para platos y personas.

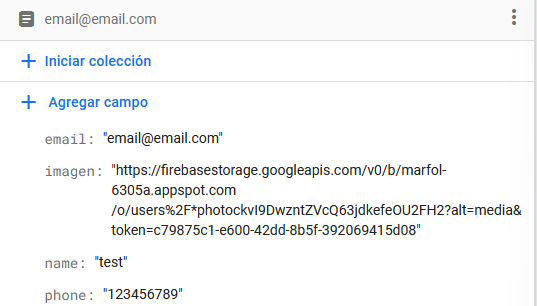
La elección de Firebase en este proyecto ha permitido un manejo eficiente de los datos y el almacenamiento de archivos. Gracias a las características y funcionalidades proporcionadas, los usuarios pueden registrarse y autenticarse de forma segura en la aplicación. Además, pueden almacenar y recuperar imágenes de perfil y otros archivos relevantes para su uso en la aplicación.

#### Arquitectura firestore

En este proyecto, se ha diseñado una estructura de base de datos utilizando Firestore para satisfacer las necesidades específicas. Se han creado cinco colecciones principales: "users", "personas", "restaurantes", "platos" e “historial”.



**Users**: Almacena la información de los usuarios registrados en la aplicación.



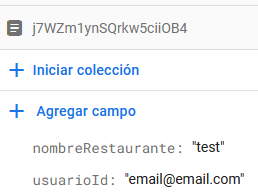
* Email (obligatorio): Almacena la dirección de correo electrónico del usuario.
* Imagen (opcional): Permite al usuario almacenar una imagen de perfil.
* Name(opcional): Guarda el nombre del usuario.
* Phone(opcional): Almacena el número de teléfono del usuario (escala a futuro).

**Personas**: Se utiliza para almacenar los contactos de cada usuario. Cada documento



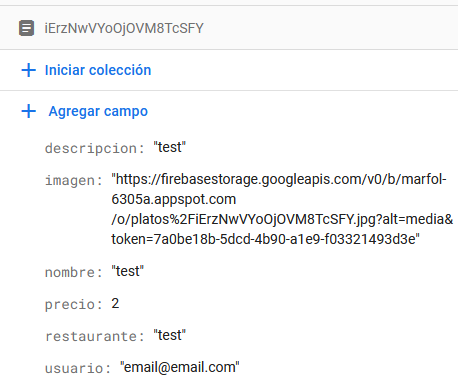
* Descripción (opcional): Permite agregar una descripción adicional de la persona.
* Imagen (opcional): Permite almacenar una imagen de perfil para la persona.
* Nombre (obligatorio): Almacena el nombre de la persona.
* UsuarioId(automático): Hace referencia al email del usuario al que pertenece esta persona.

**Restaurantes**: Almacena la información de los restaurantes asociados a cada usuario.



* NombreRestaurante (obligatorio): Almacena el nombre del restaurante.
* UsuarioId (automático): Hace referencia al email del usuario al que pertenece este restaurante.

**Platos**: Se utiliza para almacenar información sobre los platos asociados a cada usuario y restaurante.



* Descripción (opcional): Permite agregar una descripción adicional del plato.
* Imagen (opcional): Permite almacenar una imagen para un plato.
* Nombre (obligatorio): Almacena el nombre del plato.
* Precio (obligatorio): Almacena el precio del plato.
* Restaurante (automático): Hace referencia al restaurante al que pertenece el plato.
* Usuario (automático): Hace referencia al email del usuario.

Los usuarios pueden almacenar varios platos con el mismo nombre mientras sean de un restaurante distinto.

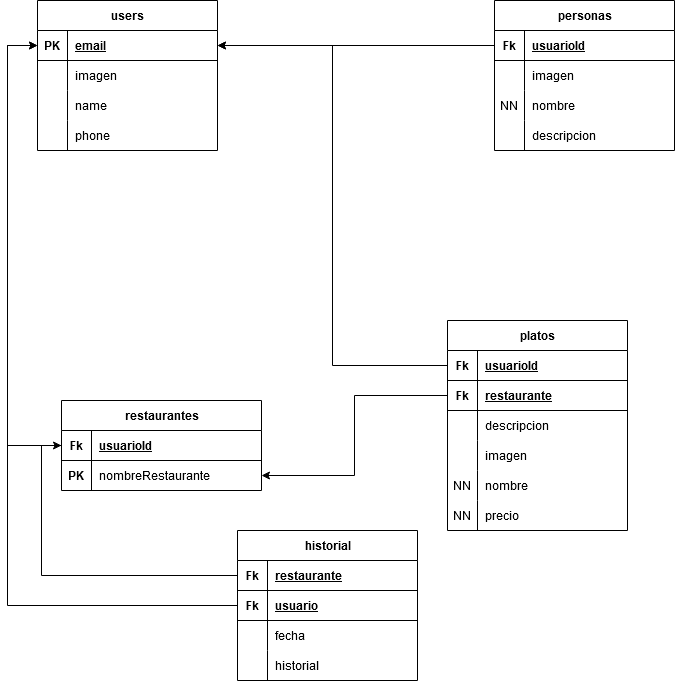
**Historial**: Se utiliza registrar las instancias en las que el usuario ha utilizado los servicios de Marfol.



* Fecha (automática): se guarda la hora y el día.
* Historial (automático): se guarda el nombre y el precio final de cada persona
* Restaurante (automático): se guarda el nombre del restaurante
* Usuario (automático): Hace referencia al email del usuario.

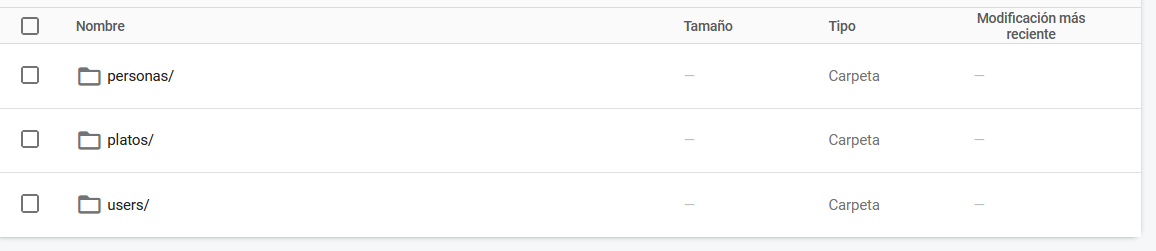
La base de datos ha sido diseñada de manera eficiente y coherente para satisfacer las necesidades del proyecto. Utilizamos el correo electrónico de cada usuario como identificador principal para filtrar la información de forma precisa. Los usuarios pueden almacenar varias personas con distintos nombres, y en caso de duplicados, se actualizan automáticamente. Lo mismo sucede con los platos y restaurantes, donde varios usuarios pueden tener registros con los mismos nombres debido a los identificadores únicos asignados. Esta estructura nos permite gestionar de manera adecuada y segura los datos, asegurando una experiencia fluida para los usuarios.

A pesar de que Firebase no es una base de datos relacional, hemos desarrollado un enfoque pseudorelacional utilizando el correo electrónico como elemento clave.



#### Arquitectura storage

Para el almacenamiento de imágenes se ha utilizado firebase storage.



Dentro del código se generan las carpetas para personas, platos y users.

Cada carpeta almacena sus respectivas imágenes

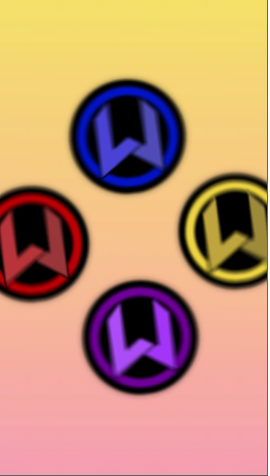
### 3.1.3 Implementación

#### Inicio

Marfol incluye una animación sencilla al abrir la aplicación, lo cual aporta un elemento estético y atractivo visualmente.

Cada vez que se inicia la app, se muestra una actividad con esta animación, que tiene una duración de 3 segundos.

Al finalizar la animación, se redirige automáticamente al index. Esta implementación brinda una experiencia más dinámica y agradable al usuario al interactuar con la aplicación.

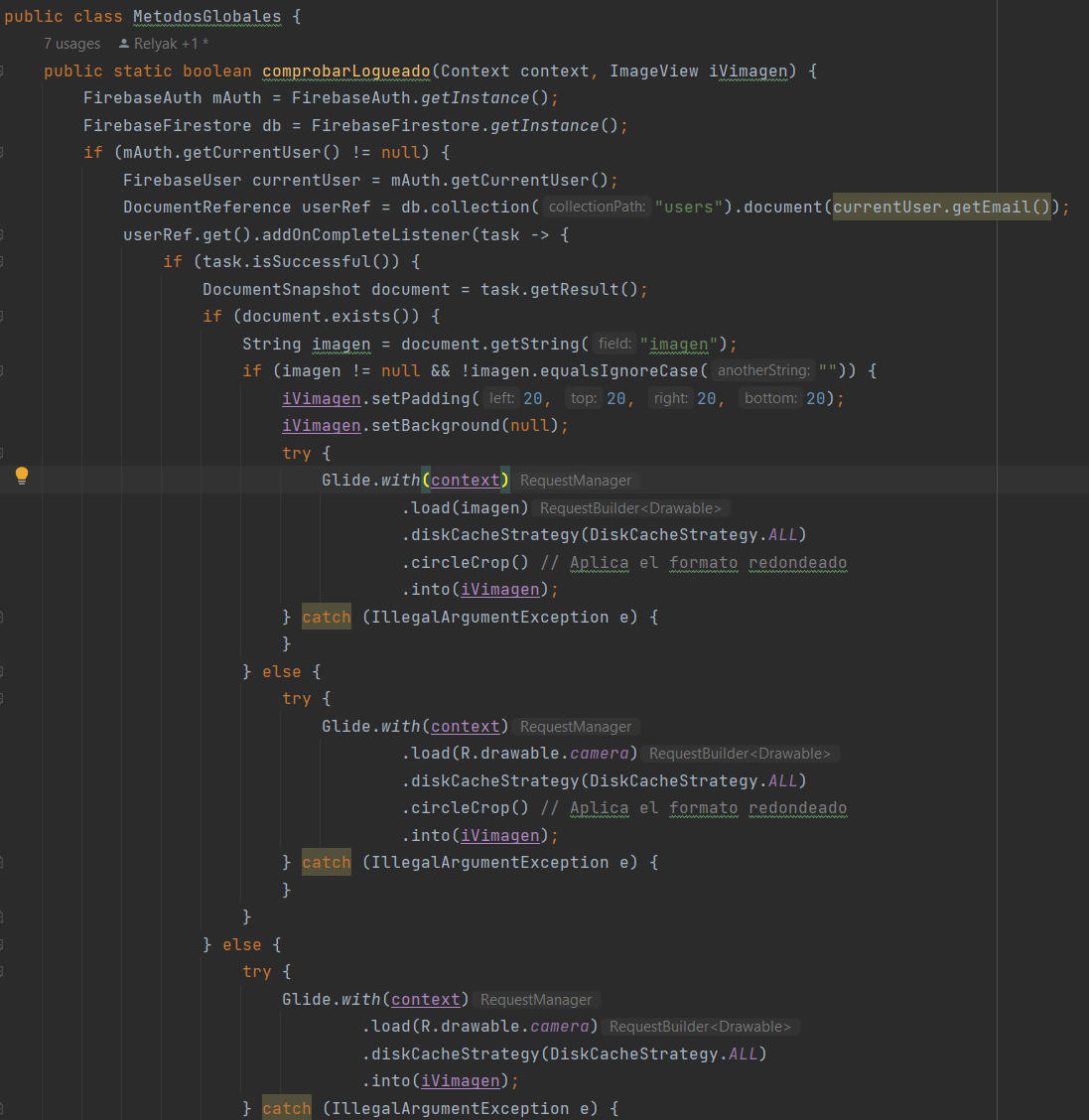
 

#### Comprobar si el usuario está logueado

Antes de comenzar con el funcionamiento, es importante mencionar el método "comprobarLogueado" que se utiliza para verificar la existencia de una instancia de usuario. Esto permite cambiar la imagen del usuario.

Se emplea la librería "Glide" para cargar las imágenes. El método busca si la instancia del usuario está vacía y, en caso afirmativo, busca (en la colección “users”) y actualiza la imagen del usuario. Este método recibe como parámetros el contexto (para su uso en varias actividades) y una imagen correspondiente a cada actividad en particular.

Se hacen las comprobaciones necesarias para evitar los fallos con la imagen.



Este método resume muy bien la funcionalidad de firebase, comprueba si existe la instancia del usuario, apuntará a una colección con esa instancia y realizará una consulta.

#### Index

-----------------------------aquí explicar index----------------------

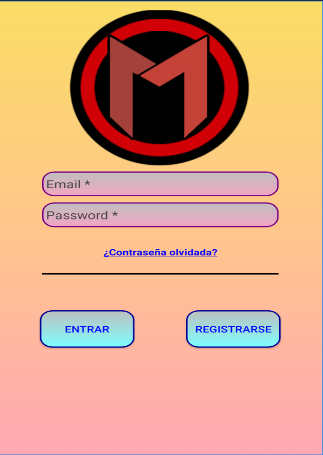
#### Autenticación

Marfol es una aplicación que ofrece la opción de crear una cuenta para almacenar datos, pero no es un requisito obligatorio. Los usuarios tienen la posibilidad de utilizar la aplicación sin necesidad de tener una cuenta registrada.

En Marfol, el usuario puede autenticarse en las actividades que contengan este icono.



Al entrar en esta actividad te lleva al apartado de autenticación que contiene esta interfaz.

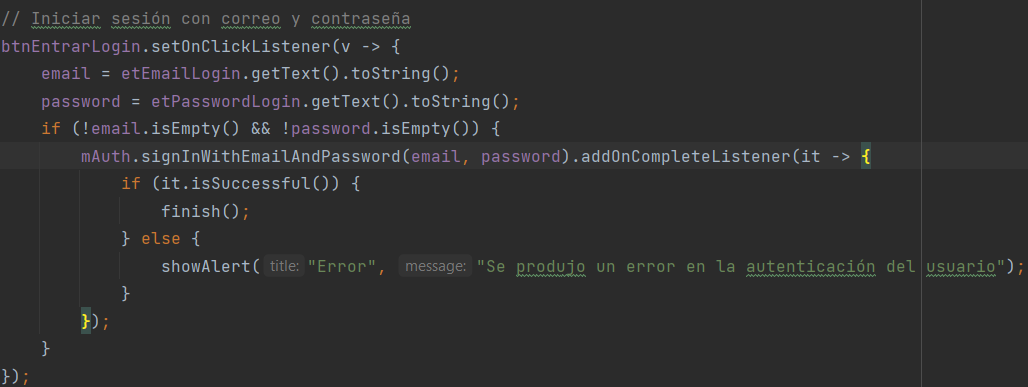


Esta actividad es una puerta de entrada a la aplicación para aquellos usuarios que ya tienen una cuenta creada. Su propósito principal es permitir un acceso rápido y sencillo.

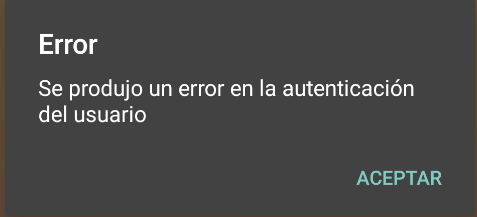
Se incluye la funcionalidad *“contraseña olvidada”* para aquellos usuarios que han olvidado su contraseña.

También se incluye la funcionalidad de registro para los nuevos usuarios.

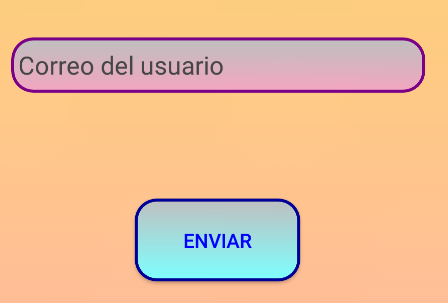
Para iniciar sesión, se realiza una consulta en Firebase utilizando los campos de correo electrónico y contraseña. Si el usuario existe y las credenciales son correctas, finaliza la actividad volviendo al index.



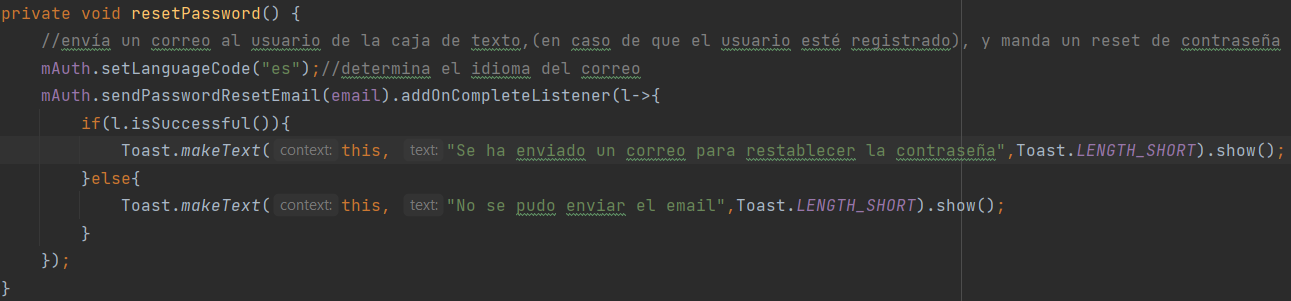
En caso contrario, se muestra un mensaje de error.



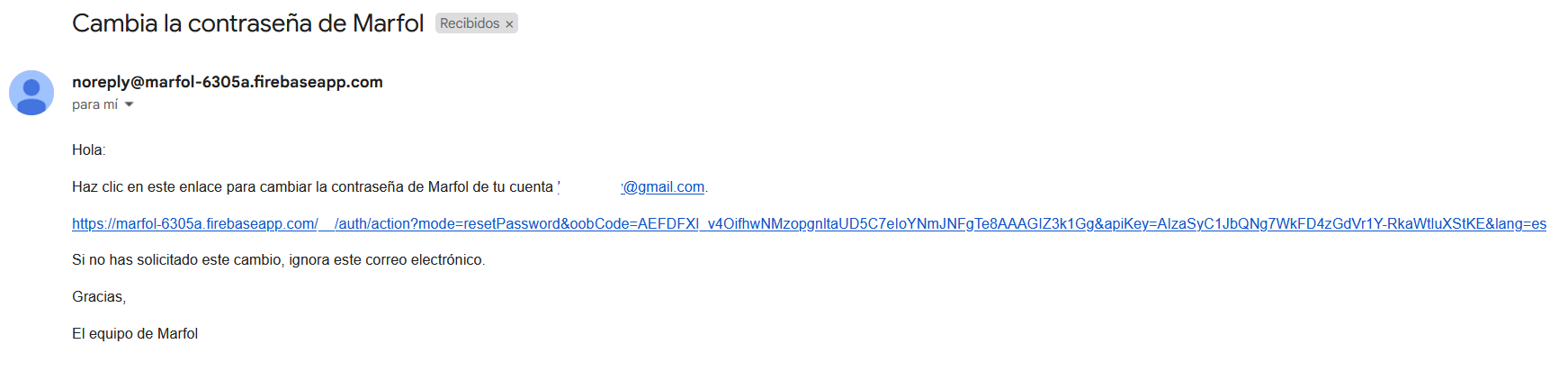
##### Contraseña olvidada



Para el proceso de recuperación de contraseña, se solicita al usuario que proporcione su correo electrónico. Si el correo electrónico proporcionado coincide con el de algún usuario registrado, se envía automáticamente un correo de recuperación desde la aplicación Marfol.

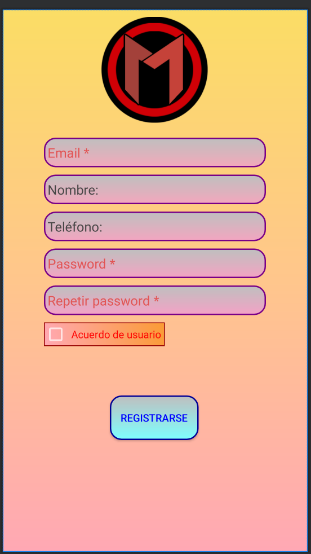


Este correo contiene las instrucciones necesarias para que el usuario pueda restablecer su contraseña y acceder nuevamente a su cuenta.



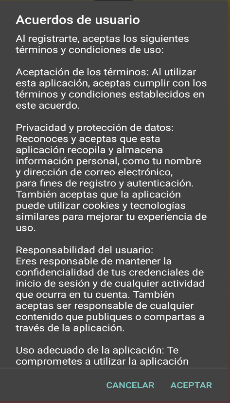
##### Registro

La actividad de registro en la aplicación tiene como objetivo permitir a los usuarios crear una nueva cuenta.



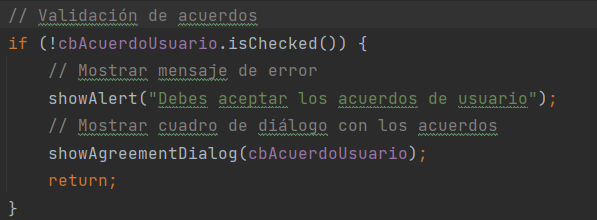
Al iniciar esta actividad, se presenta una interfaz de usuario intuitiva que solicita al usuario que proporcione su dirección de correo electrónico y una contraseña. Estos datos son esenciales para crear una cuenta personalizada y segura.

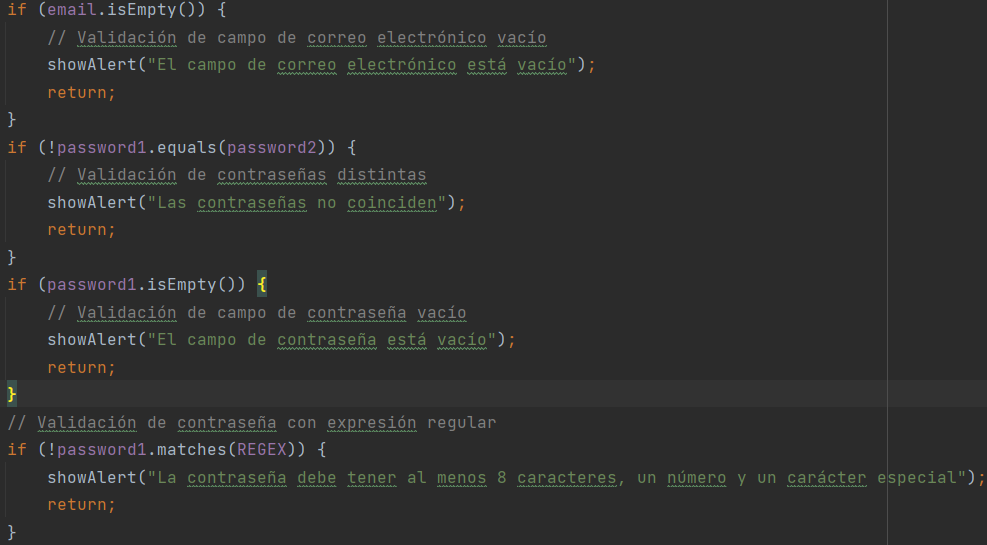
Los campos obligatorios en esta actividad son el email, contraseña (que tiene que cumplir unos requisitos mínimos de seguridad) y los acuerdos de usuario.



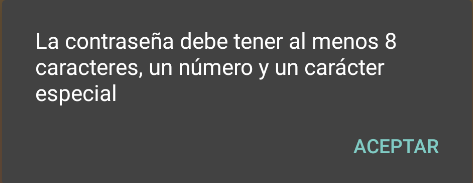
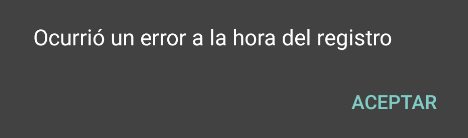
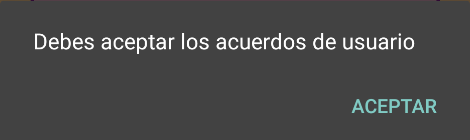
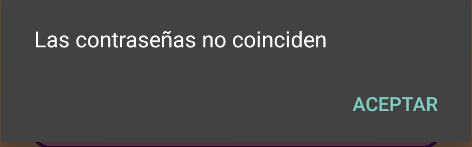
Al registrarte en la aplicación, el usuario acepta los términos y condiciones de uso que incluyen la aceptación de los términos establecidos, el manejo de privacidad y protección de datos, la responsabilidad del usuario, el uso adecuado de la aplicación y el reconocimiento de la propiedad intelectual.

Se han incorporado varias validaciones para evitar fallos.





En caso que ocurra algún tipo de error, se mostrará un mensaje personalizado para cada caso.

Una vez que el usuario ha ingresado los campos obligatorios y es válido, se realiza el registro con el botón “REGISTRARSE”.

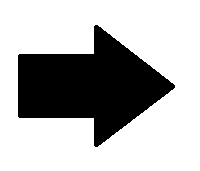


A su vez, se crea dentro de Firestore un documento con los datos del nuevo usuario.

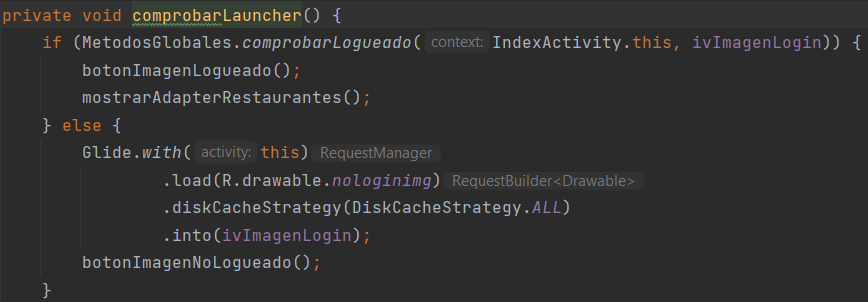
Después del registro el usuario es redirigido al index.

#### Gestión de usuario

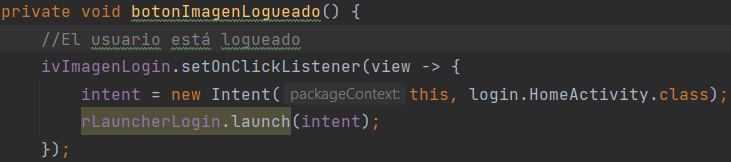
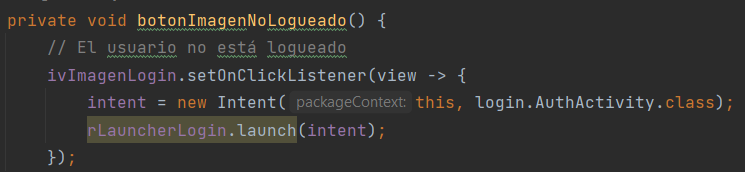
Una vez logueado, se modificará la funcionalidad de la imagen, cargando la imagen del usuario y si no la tiene cargará una por defecto.



Para cambiar la funcionalidad de la imagen rimero se comprueba si el usuario está logueado.



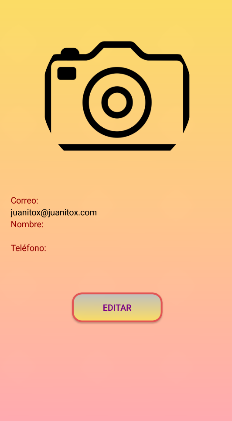
Después cambia la funcionalidad con el método *botonImagenLogueado* que simplemente enviará al login o a la gestión del usuario. En el caso contrario enviará al usuario a la autenticación.

##### Home

Lo primero que hará dentro de esta actividad es comprobar si el usuario está logueado, *comprobarLogueado*, y traerá los datos de firebase.

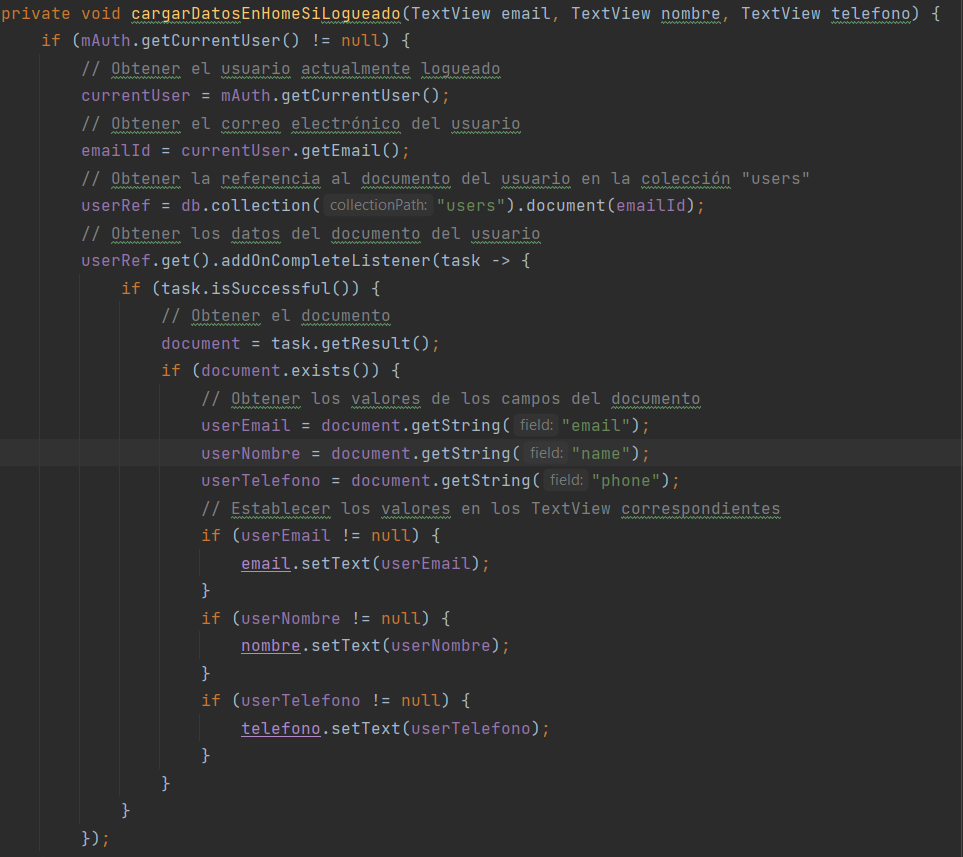
Home sin datos



El usuario podrá cambiar la foto o sus datos con el botón editar que llevará a otra actividad.

En caso de entrar a esta actividad sin estar logueado (error) la actividad finalizará, evitando posibles fallos.

El método *cargarDatosEnHome* hará una consulta en la base de datos para cargar los campos nombre,teléfono y correo de cada usuario.



##### Editar datos del usuario

Dentro de esta actividad el usuario podrá modificar el nombre, teléfono y foto.

Se cargan los datos del usuario para facilitar la modificación, si no tiene datos se mostrarán los campos en blanco.

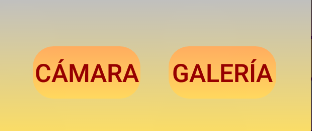
Editar sin datos



Editar con datos

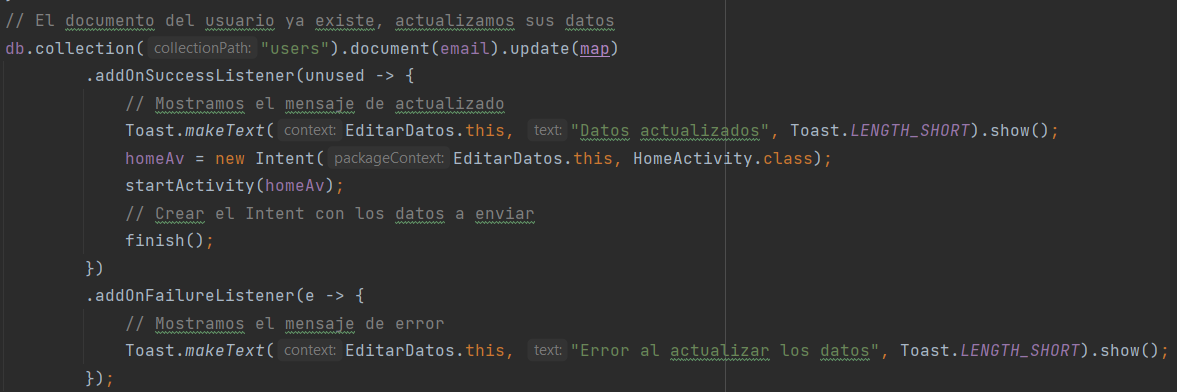


Al seleccionar la imagen dentro de Editar, se mostrará un pop-up para elegir cámara o galería.

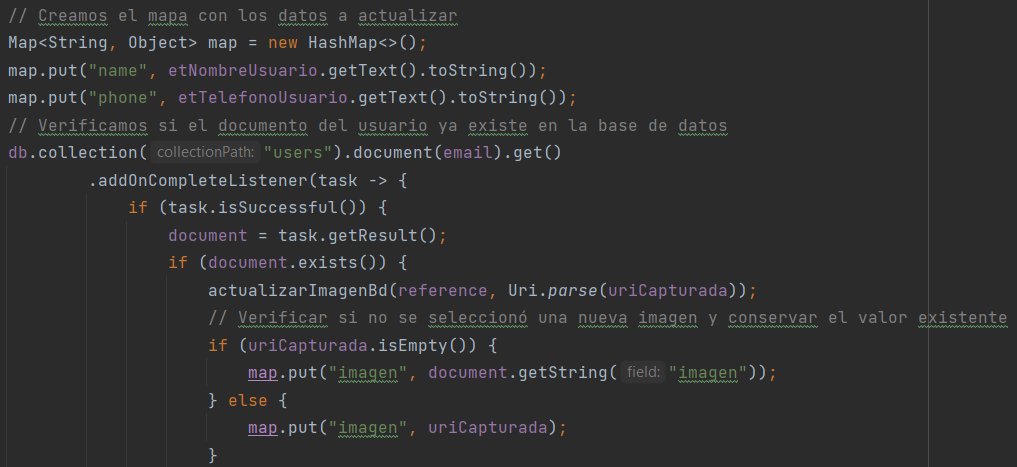


El funcionamiento principal de la actividad home y editar es hacer una consulta en firebase para traer los datos del usuario filtrando por el correo.

Además, en Editar se utiliza el método update, donde se manda la colección, el documento (con el correo del usuario) y un hashMap con los datos.



El mapa se llenará de los nuevos datos.



En caso de que el usuario no desee realizar modificaciones, se procede a sobrescribir los datos antiguos (ya cargados) con los nuevos.

Home con datos



#### Gestión de datos

Dentro del menú, si el usuario está logueado, se encuentra la opción Editar.



Al entrar se encuentran los botones PERSONAS, RESTAURANTES y PLATOS.

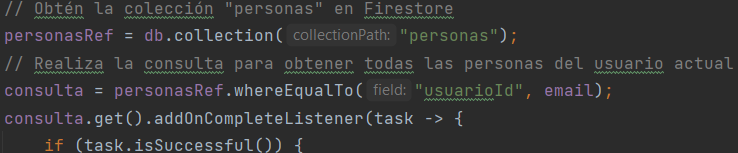
Cada opción permite ver, editar o borrar los datos relacionados al usuario.



##### Personas

Dentro de esta actividad se hace una consulta dentro de la base de datos, en la colección personas, filtrando por el usuario (whereEqualTo), con esos datos se llena un RecyclerView personalizado donde cada fila es la imagen y el nombre de la persona.

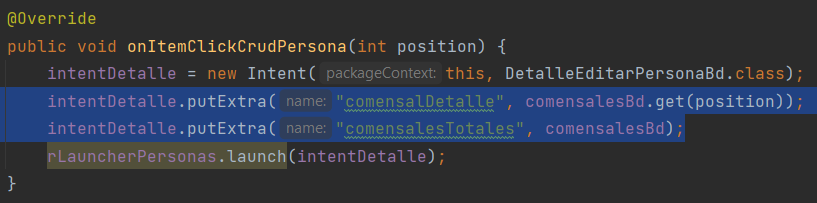
Se muestra una lista de todas las personas que alguna vez el usuario ha añadido.





El color amarillo representa esta colección, las filas tienen el estilo amarillo grisáceo. Las personas sin imagen, por defecto, son asignados con el logo amarillo.

Para el mejor funcionamiento y para evitar la redundancia, al seleccionar una persona se envían a través del intent, un objeto persona y la lista de todas las personas.



Finalmente, se abrirá otra actividad con el detalle de la persona seleccionada y sus datos se podrán modificar desde la nueva actividad.

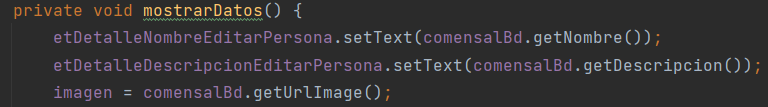


Dentro de esta actividad se recoge el intent con el objeto persona y la lista de personas para utilizarlos más adelante.



En esta actividad no es necesario hacer una consulta para rellenar datos debido a que la actividad *EditarPersona* envía el objeto persona con todos los datos del usuario.

Para llenar los datos, se crea un método *mostrarDatos* el cual asigna a los campos nombre, imagen y descripción los datos del objeto recibido.



Para la imagen se ha generado un método adicional que muestra una barra de progreso en el caso de que la imagen tarde en cargarse.



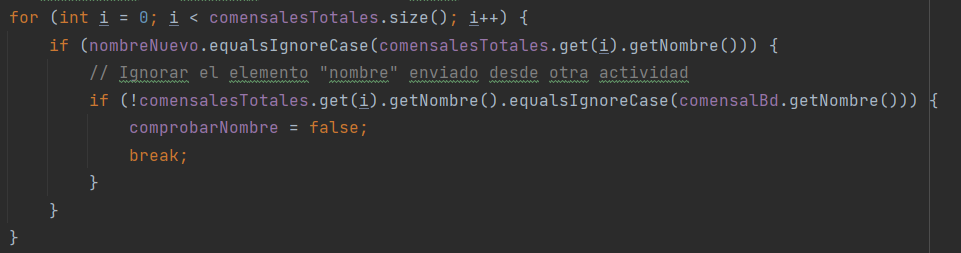
Se mostrará una barra de progreso mientras la imagen se carga.



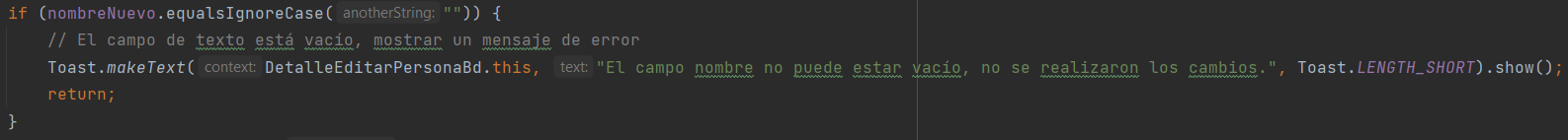
Cuando termine, se ocultará la barra de progreso y se mostrará la imagen.

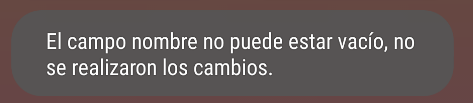


En esta actividad se comprueba que el nombre nuevo no exista en la base de datos del usuario, para ello se utiliza la lista de comensales recibida anteriormente y se valida con una variable boolean (siempre estará verdadera), se pondrá en falsa si encuentra un nombre igual dentro de la lista, además si el nombre encontrado es el mismo nombre del objeto recibido, lo ignorará.

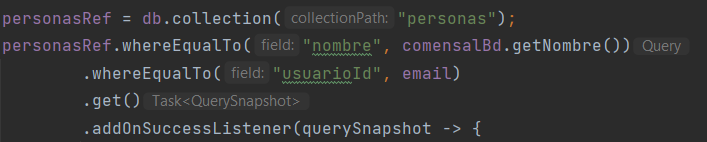


También se validará que el nombre nuevo no esté vacío.

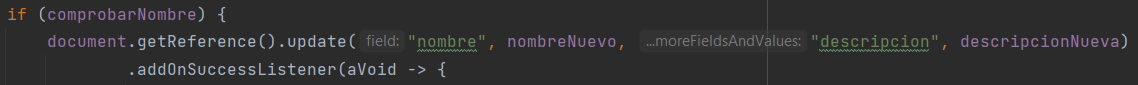




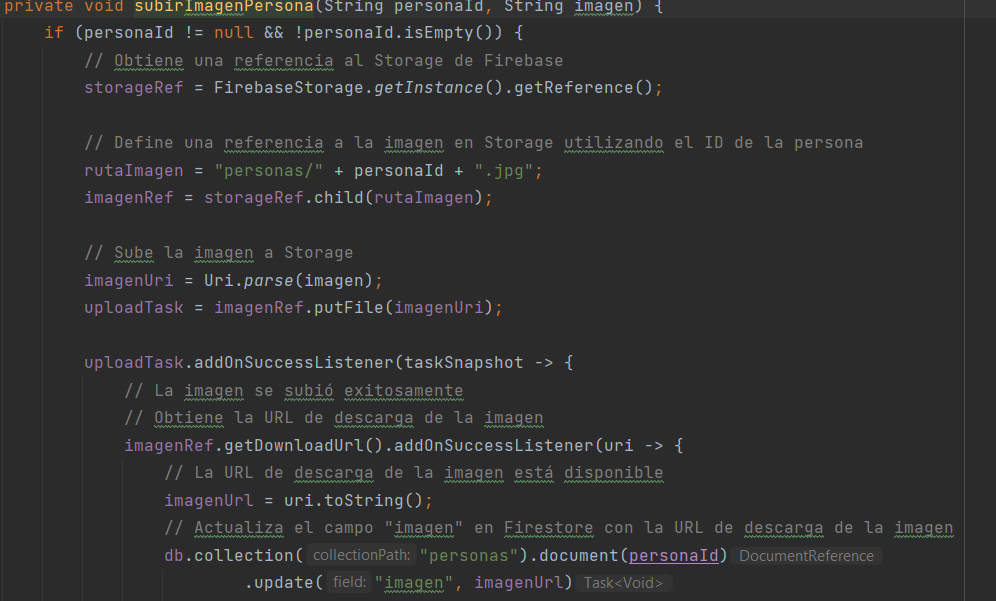
En el caso de que no se encuentren errores se realizará una consulta a la colección “personas” filtrando por nombre y el correo del usuario.



Finalmente, se realiza un update con los nuevos datos.



Para las imágenes se ha realizado un método adicional debido a que las imágenes se almacenan en storage, firestore almacena la ruta de la imagen del storage dentro de la colección “personas”.



En este método se hace referencia dentro de storage a la carpeta “personas” y después se realiza un update en firestore, donde se envía por parámetro la nueva imagen.

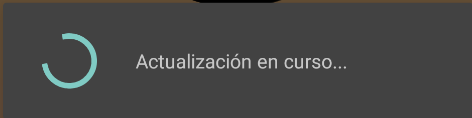
Al seleccionar “*EDITAR”* se realizan todas las comprobaciones anteriores y se hace una llamada al método de editar donde se realiza dentro un update con los nuevos datos.

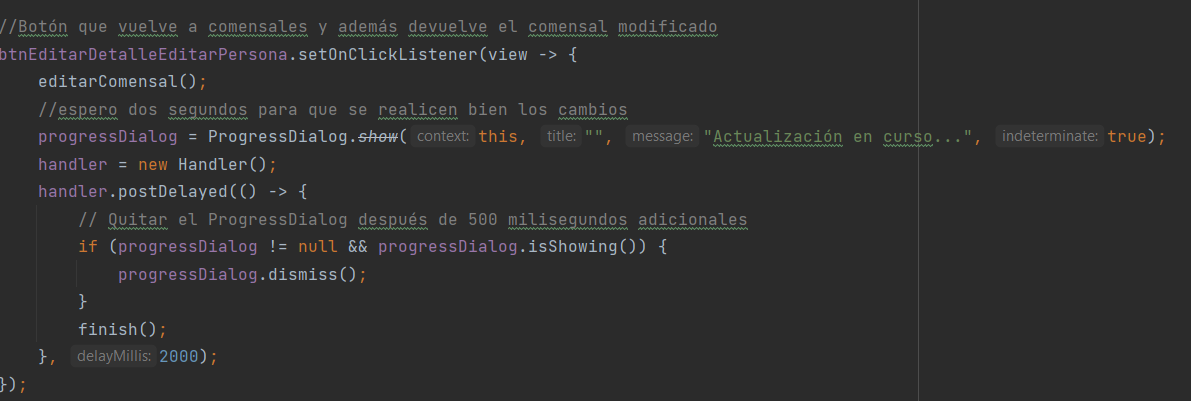


Por último, se añade un cuadro de texto de espera antes de finalizar la actividad.

Si no se retarda este proceso, debido a la velocidad a la que se produce todo el cambio, se actualiza dentro de la base de datos la nueva persona, pero no se verán aplicados los nuevos cambios en la lista.

Para evitar ese problema se ha añadido el cuadro de texto y dos segundos de espera.





Al terminar, se finaliza la actividad actual, volviendo atrás con la nueva lista.

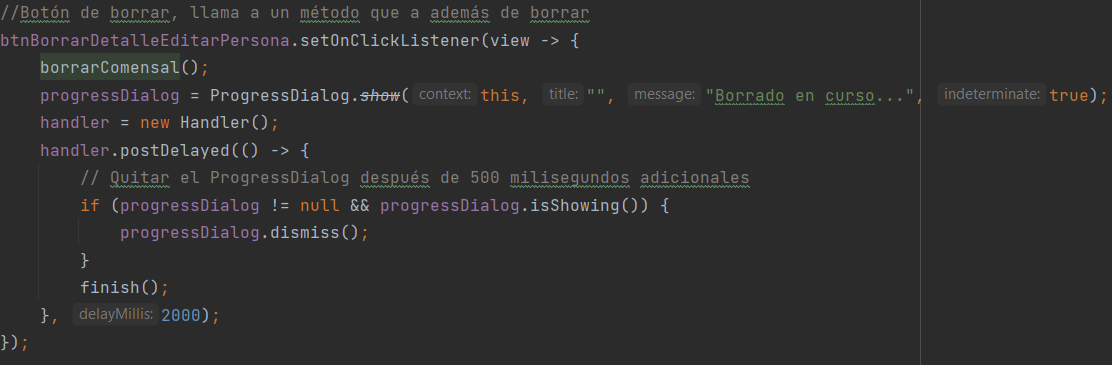
Dentro de la actividad *Detalle* también se encuentra el botón de borrar.



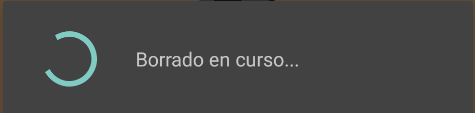
Al seleccionar este botón, se realiza una consulta en firestore dentro de la colección personas filtrando por nombre y correo. Si encuentra algún campo que coincida con el nombre y correo proporcionados en el detalle dentro de la base de datos se procede al borrado.



Se produce el mismo proceso que en editar, el programa realiza los cambios, pero aparece un cuadro de texto con 2 segundos de espera para así poderse actualizar también la lista.



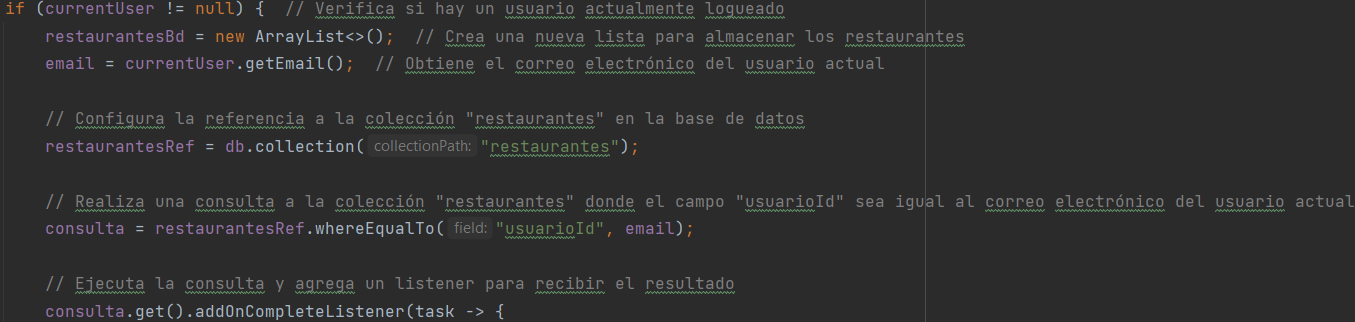
Al terminar, se finaliza la actividad actual, volviendo atrás con la nueva lista.



##### Restaurantes

Dentro de esta actividad se hace una consulta dentro de la base de datos, en la colección restaurantes, filtrando por el usuario (whereEqualTo), con esos datos se llena un RecyclerView personalizado donde cada fila es el nombre del restaurante.

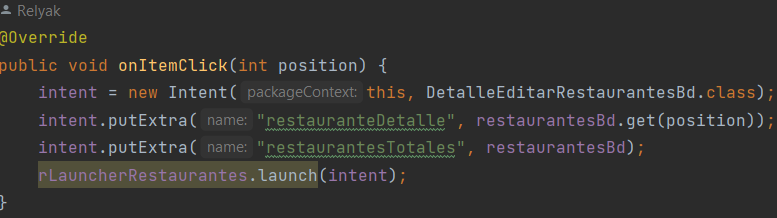
Se muestra una lista de todos los restaurantes que alguna vez el usuario ha guardado.





El color azul representa esta colección y cada fila tiene el color azul grisáceo.

Para el mejor funcionamiento y para evitar la redundancia, al seleccionar un restaurante se envían a través del intent, un objeto restaurante y la lista de todos los restaurantes.



Finalmente, se abrirá otra actividad con el detalle del restaurante seleccionado y sus datos se podrán modificar desde la nueva actividad.



Con el intent recogido se carga el nombre del restaurante.

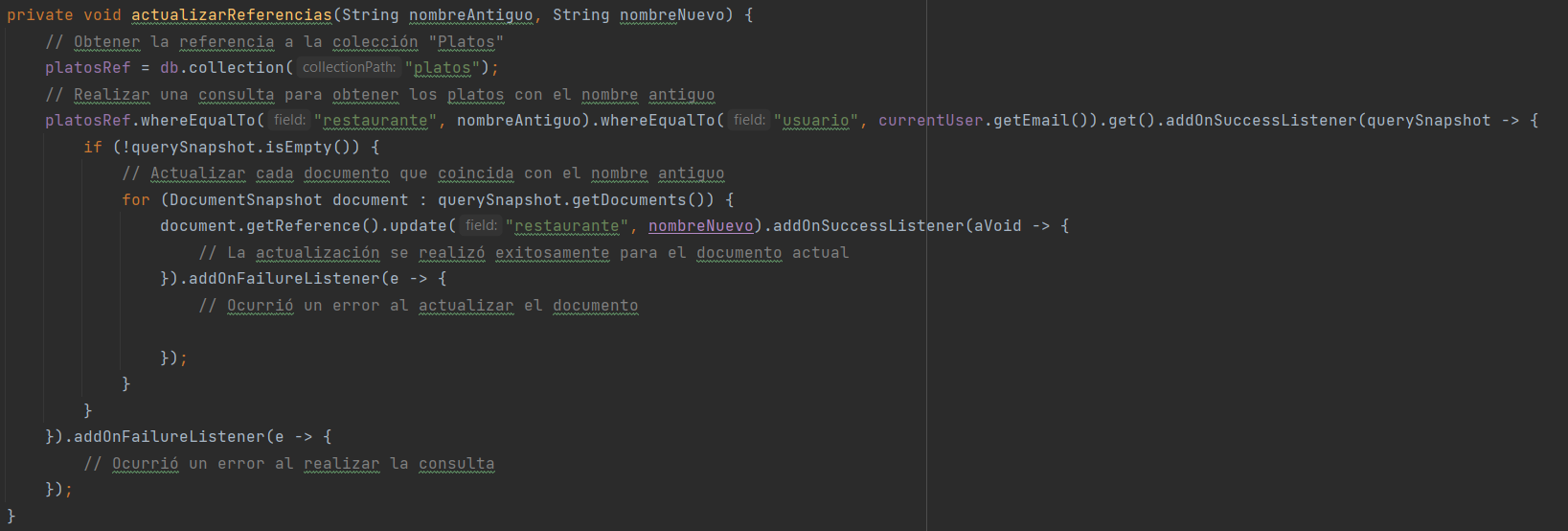
En esta clase se comprueba que el nuevo nombre no esté vacío ni tampoco repetido que el nombre ya exista en la lista.



Si se cumplen las comprobaciones, se procederá a actualizar los restaurantes utilizando el método update.

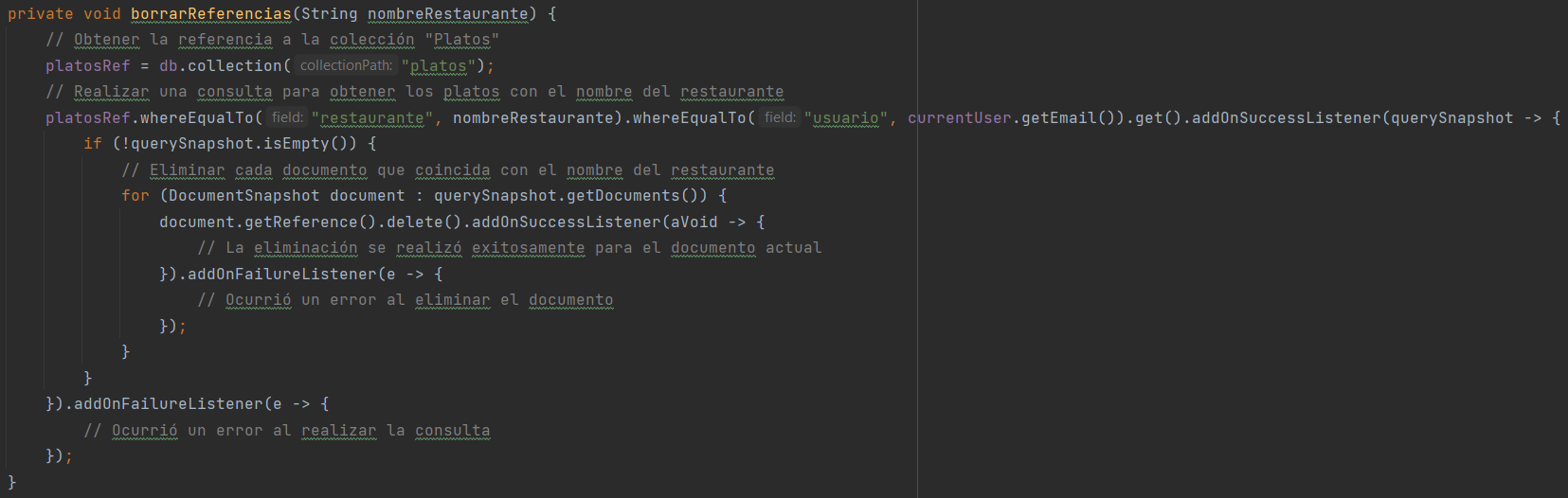
Esta clase tiene un funcionamiento distinto debido a que los platos tienen un restaurante de referencia, firestore no es relacional y no actualizará los campos restaurantes dentro de los platos

Para solventar ese problema al actualizar el restaurante se hace una consulta dentro de la colección platos filtrando por nombre antiguo y por email del usuario, a posteriori se actualizan los campos con el nuevo nombre.

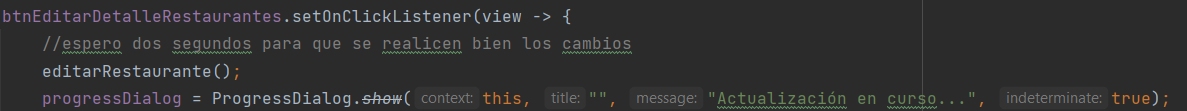


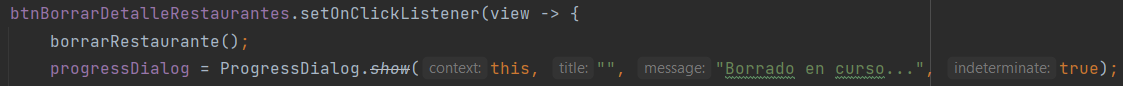
Para el borrado funciona exactamente igual, si el restaurante no existe y no se borran esas referencias es imposible acceder a esos platos.

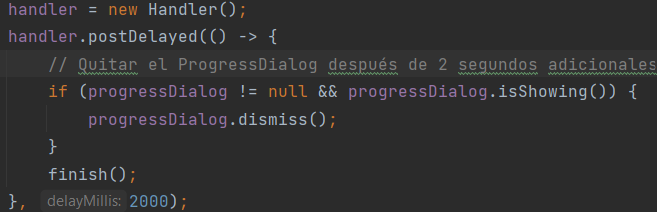
Se ha implementado un método que,al borrar el restaurante, busque los platos con ese nombre y los borre junto con el restaurante.



Ambos métodos, al terminar, esperarán 2 segundos.





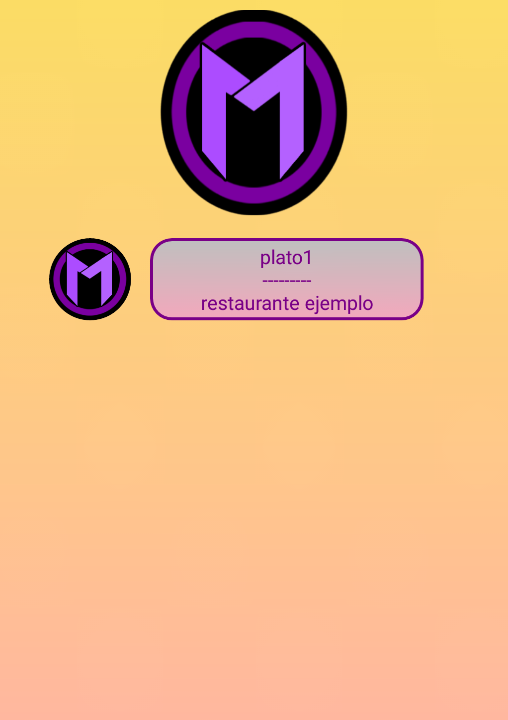


Al terminar, se finaliza la actividad actual, volviendo atrás con la nueva lista.

##### Platos

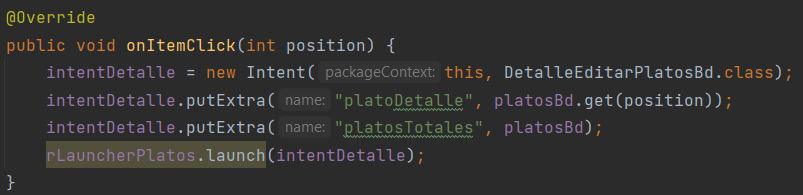
Dentro de esta actividad se hace una consulta dentro de la base de datos, en la colección platos, filtrando por el usuario (whereEqualTo), con esos datos se llena un RecyclerView personalizado donde cada fila es el nombre del plato y su imagen.

Se muestra una lista de todos los platos que alguna vez el usuario ha añadido.



El color morado representa esta colección, las filas tienen el estilo morado grisáceo. Los platos sin imagen, por defecto, son asignados con el logo morado.

Para el mejor funcionamiento y para evitar la redundancia, al seleccionar un plato, se envían a través del intent, un objeto plato y la lista de todos los platos.



Finalmente, se abrirá otra actividad con el detalle del plato seleccionado y sus datos se podrán modificar desde la nueva actividad.



Con el intent recogido se asignan los datos al plato, el nombre del restaurante no se podrá modificar.

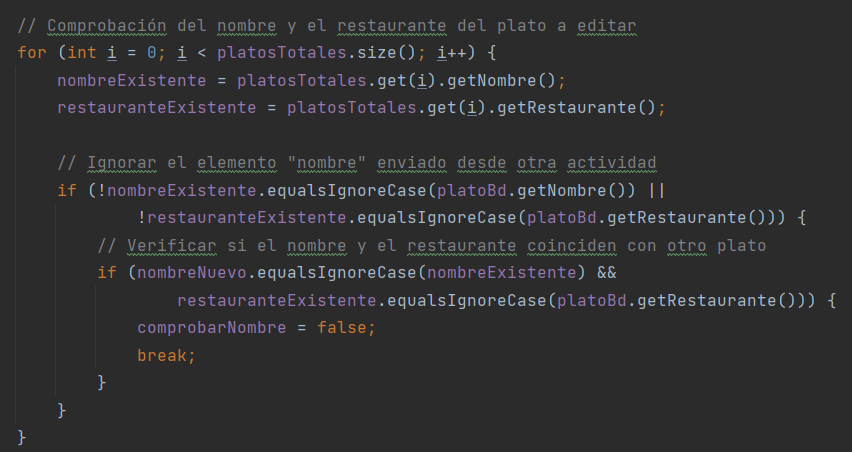
Para la imagen se ha generado un método adicional que muestra una barra de progreso en el caso de que la imagen tarde en cargarse.

Sin cargar Cargado

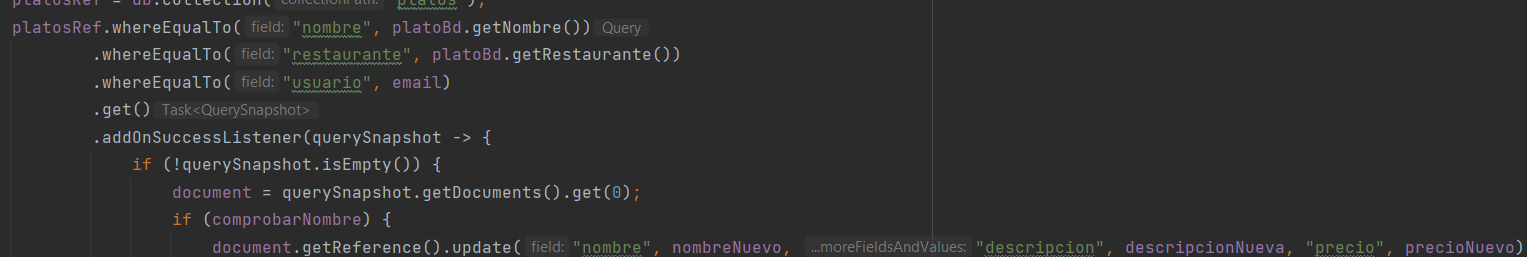
 

Como en las otras actividades, se comprobará que el nombre del nuevo plato no coincida con el usuario, plato y restaurante de la base de datos.

El usuario podrá tener varios platos con el mismo nombre mientras sean de distintos restaurantes.

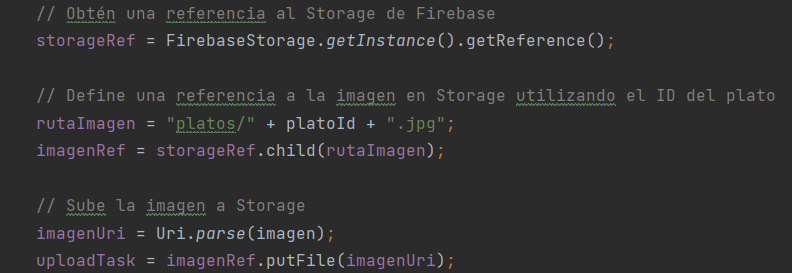


Se comprobará que el nombre no esté vacío y se hará una consulta a la base de datos filtrando por usuario, restaurante y nombre.



A continuación, se realizará el update en caso de seleccionar la opción editar.

Para la imagen se abrirá la cámara o galería, y llamará al método para insertar y actualizar la imagen dentro de storage.



Si se cumplen todas las validaciones se procederá a actualizar el plato con los nuevos datos. Se esperará dos segundos para terminar la actividad y evitar errores.

El método de borrado funciona igual que en las clases anteriores, filtra por nombre, usuario y restaurante y borra el plato seleccionado.



Al igual que actualizar, se esperará dos segundos antes de terminar la actividad.

El proyecto que el alumno presenta constará, al menos, de:

* Introducción,
* Esquemas y gráficos.
* Programación del proyecto.
* Justificación de la viabilidad.
* Prevención de riesgos.