Josep Ulldemonis, 1604197
Daniel García, 1600877
Albert Ceballos, 1604259
Gerard Asbert, 1603295

Informe de Desarrollo de la Aplicación Botanicapp

1. Introducción

1.1. Objetivo del Proyecto

El objetivo de este proyecto es desarrollar una aplicación móvil capaz de identificar plantas a partir de fotografías. La aplicación utiliza servicios en la nube y APIs para ofrecer una solución rápida y precisa, y así aprender a utilizar este tipo de tecnologías.

1.2. Descripción General

La aplicación desarrollada permite a los usuarios tomar una foto de una planta o cogerla de la galería, subirla a la aplicación y obtener la identificación de la planta en tiempo real. Para lograr esto, se han utilizado varias tecnologías y servicios, incluyendo Cloud Functions, Firebase y la API de Pl@ntnet.

2. Tecnologías Utilizadas

2.1. Frontend

- **Framework**: Flutter
- **Lenguaje**: Dart
- **Bibliotecas**: Provider, GoogleAuth, Firebase, GoogleCloud

2.2. Backend

- **Cloud Functions**: Utilizada para subir el backend que permite controlar el sistema de usuarios y un registro del historial de cada uno.
- **Firebase**: Utilizado como base de datos para almacenar la información de las plantas y los usuarios.

2.3. API de Terceros

- **Pl@ntnet API**: Utilizada para la identificación de las plantas a partir de las fotografías proporcionadas por los usuarios.

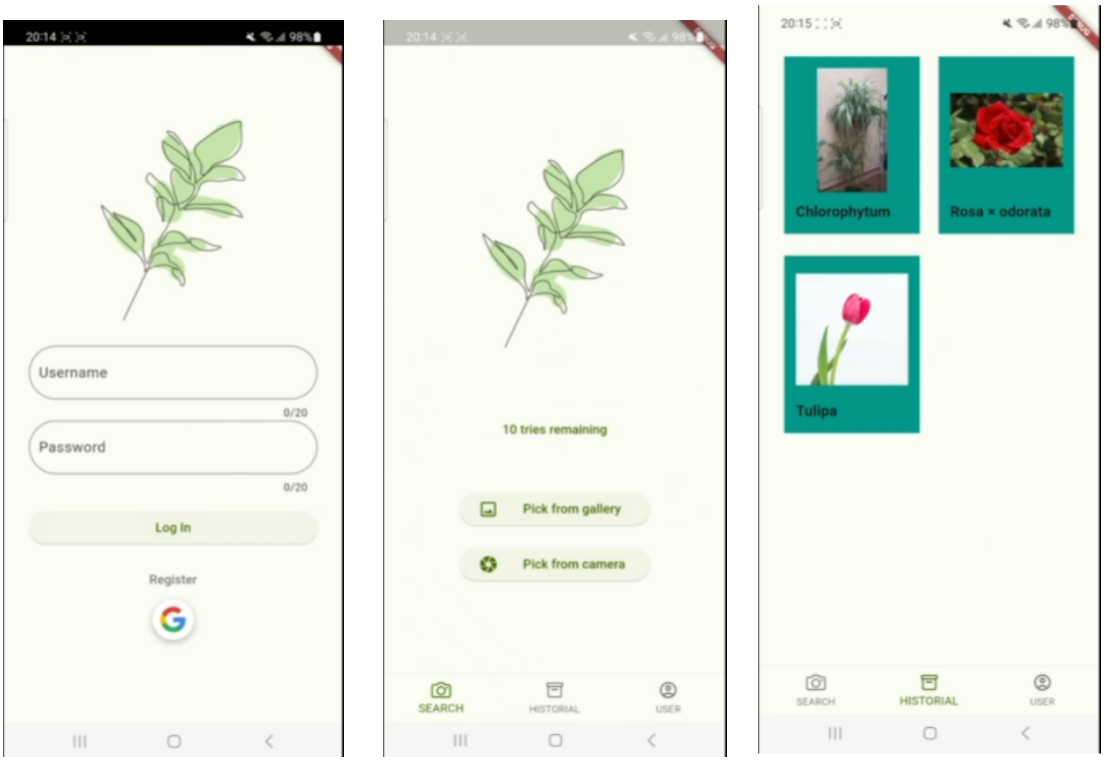
3. Descripción del Desarrollo

3.1. Arquitectura de la Aplicación

La arquitectura de la aplicación consiste de un frontend y un backend. El frontend se ocupa de los inicios de sesión y registros de usuario conectándose con la api de Firebase. También recoge las imágenes y se las pasa a la base de datos (para guardarlas para el mantenimiento del historial) y al backend, el cual se ocupa de mandarlas a la API Pl@ntnet para que haga la identificación. Cuando el backend acaba con la identificación, le pasa los resultados al frontend para que se los muestre a los usuarios, y tambien lo guarda en la base de datos de Firestore para mantener el historial del usuario.

3.2. Implementación del Frontend

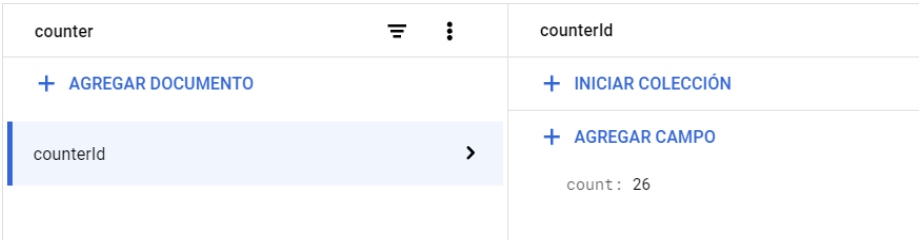
- **Interfaz de Usuario**: Hemos desarrollado una interfaz simple, en la cual hay una pantalla de inicio de sesión, donde el usuario puede registrarse e iniciar sesión mediante una cuenta de Google. Seguidamente se redirige al usuario a la pantalla principal, donde se le da al usuario la opción de escoger una imagen de la galería o tomar una foto con la cámara para que sea identificada. Desde esta pantalla principal el usuario también puede acceder a su historial de identificaciones y a su perfil de usuario.



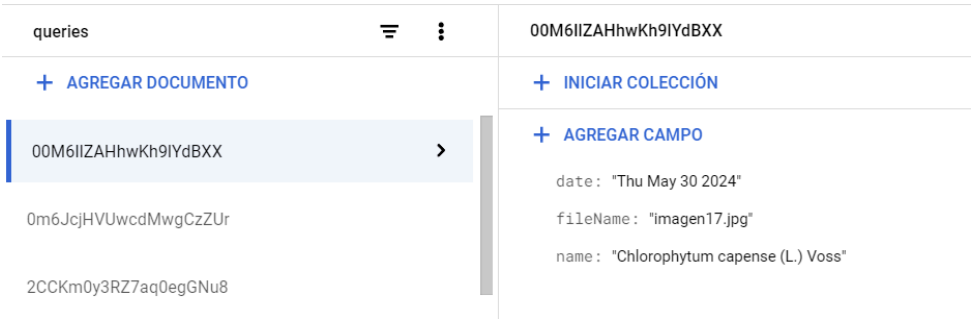
- **Funcionalidades Clave**: Iniciar sesión y registrar-se, recogida de imágenes (Galería o Cámara), Acceso a historial.

3.3. Implementación del Backend

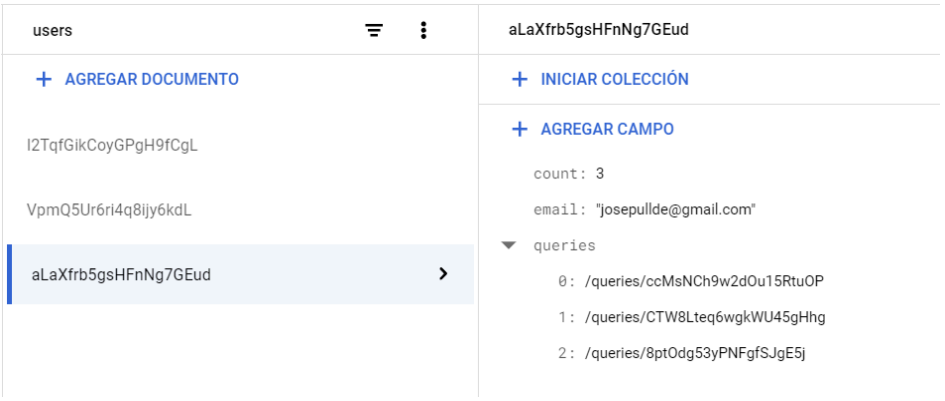
- **Cloud Functions**: Todo el backend está subido a una cloud function que premitirá interactuar con la base de datos para el control de los usuarios y de las queries, también de un pequeño historial.
- **Base de Datos**: Hemos utilizado la base de datos de Firestore, y la hemos organizado en tres sectores:
    - El primer sector lo hemos utilizado para llevar un counter de las queries, para vigilar que no nos pasemos del límite gratuïto



    - El segundo sector lo hemos utilizado para organizar las queries. Cada query la guardamos con su ID como nombre, y guardamos la fecha en que se hizo, la nombre de la imagen enviada, y finalmente el nombre de la planta identificada en dicha imagen.



- El tercer y ultimo sector lo utilizamos para guardar a los usuarios. Para cada usuario guardamos su correo electrónico, el contador de queries que ha hecho para mantener un límite, y las mismas queries que ha hecho dicho usuario para poder mantener un historial.



3.4. Integración con Pl@ntnet API

- **Llamadas a la API**: Las llamadas a la API se hacen a partir de una petición MultipartRequest Post desde el FrontEnd, en la que con la api key y la imagen jpeg adjunta en el body. Limitamos el número máximo de llamadas a la API a 10 diarias por cada usuario teniendo un control en la base de datos del número de queries ejecutadas, se comprueba este número antes de realizar cualquier interacción a la API, así nos aseguramos que no sobrepase el límite diario gratuito que ofrece la API.

- **Procesamiento de Datos**: Los datos recibidos de la API se procesan a partir de un json que incluye la información sobre la planta enviada, hemos cogido la información más relevante para nuestras utilidades.

4. Pruebas y Validación

4.1. Estrategia de Pruebas

Para realizar pruebas hemos usado un mobil Android para ejecutar la aplicación y la pagina web de nuestro proyecto en Google Cloud para visualizar los cambios en nuestras bases de datos. También hemos usado dos ordenadores, uno para el frontend y el otro para el backend, para visualizar las conexiones frontend-backend y sus resultados.

5. Desafíos y Soluciones

5.1. Desafíos Técnicos

· **Procesamiento de Imágenes**: La carga y procesamiento de imágenes grandes presentaron muchos problemas, sobre todo en el momento de descargar estas del bucket. Se solucionó generando una url firmada cada vez que se requeria de dicha imagen

· **Integración de API**: Garantizar que las llamadas a la API de Pl@ntnet y a las Cloud Functions sean eficientes.

6. Conclusión

6.1. Resumen del Proyecto

En resumen, el proyecto ha sido instructivo en múltiples sectores, desde el desarrollo de una aplicación, la creación de un servidor y el manejo de las conexiones frontend-backend, hasta el uso de cloud functions y la organización y uso de APIs y sus riesgos.

No hemos tenido muchos problemas a la hora de crear las diferentes partes por separado, pero el problema ha sido juntarlas de forma que todo funcionara correctamente. También ha resultado una dificultad adicional el tener que aplicar medidas de seguridad, debido a que el uso indebido de la cloud puede resultar muy peligroso económicamente.

Finalmente, ha estado bien aprender sobre la nube, aunque creemos que lanzar un proyecto como estudiante es inviable debido a los costos asociados al uso de la cloud.

6.2. Futuras Mejoras

En un futuro podríamos implementar más medidas de seguridad, ya que como hemos dicho, sería muy peligroso que un usuario abusase de nuestro uso de la cloud. También podríamos pulir el aspecto de la aplicación, añadir alguna funcionalidad más (por ejemplo, que a parte de devolver el nombre de la planta también devolviese información importante sobre dicha planta), y finalmente, podríamos pensar en alguna forma de monetizar la aplicación por si decidiésemos lanzarla en un futuro.

7. Anexos

7.1. Código Fuente

https://github.com/albertceballos0/UAB-SISTEMES-MULTIMEDIA

https://console.cloud.google.com/functions/list?hl=es&project=sistemes-multimedia

7.2. Documentación Adicional

https://my.plantnet.org/doc/openapi