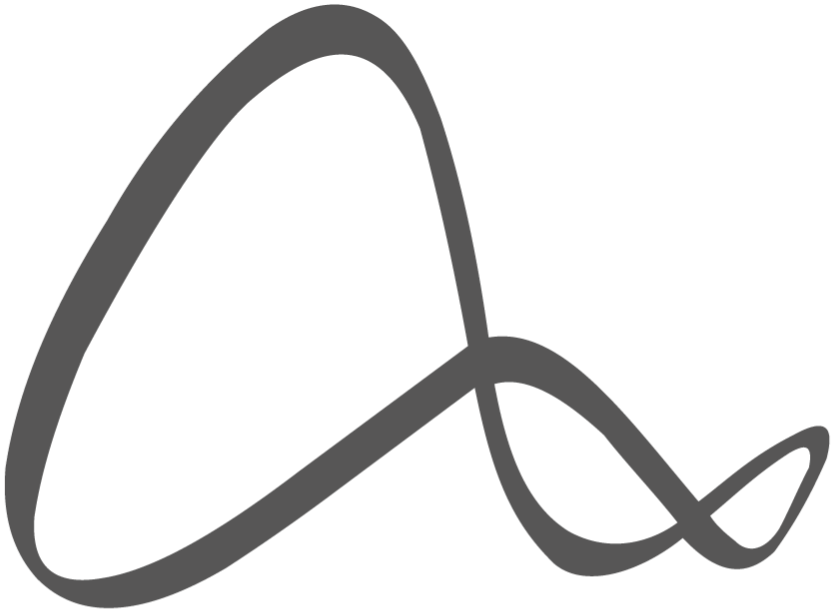


upiita-ipn

Propuesta de Proyecto Final | Resultados

Programación de Dispositivos Móviles

Cesáreo Javier Muñoz Rodríguez

Equipo 6

Castillo Martínez Leonel Jafet

15/01/2021

Contenido

1. Propuesta

El proyecto final que se propone realizar es una aplicación móvil de notas.

El usuario tendrá una pizarra central en la que podrá visualizar una vista previa de las notas que ha creado hasta el momento, y al acceder a cada una podrá editarla o hasta borrarla.

También tendrá un botón de acción en la pizarra central que le permita crear una nota de texto nueva, donde deberá guardar los cambios para que se suba a la pizarra central y así poder visualizarla junto con las otras notas.

En la figura 1 se muestra una pantalla estimada a la que se desea presentar hasta el final.

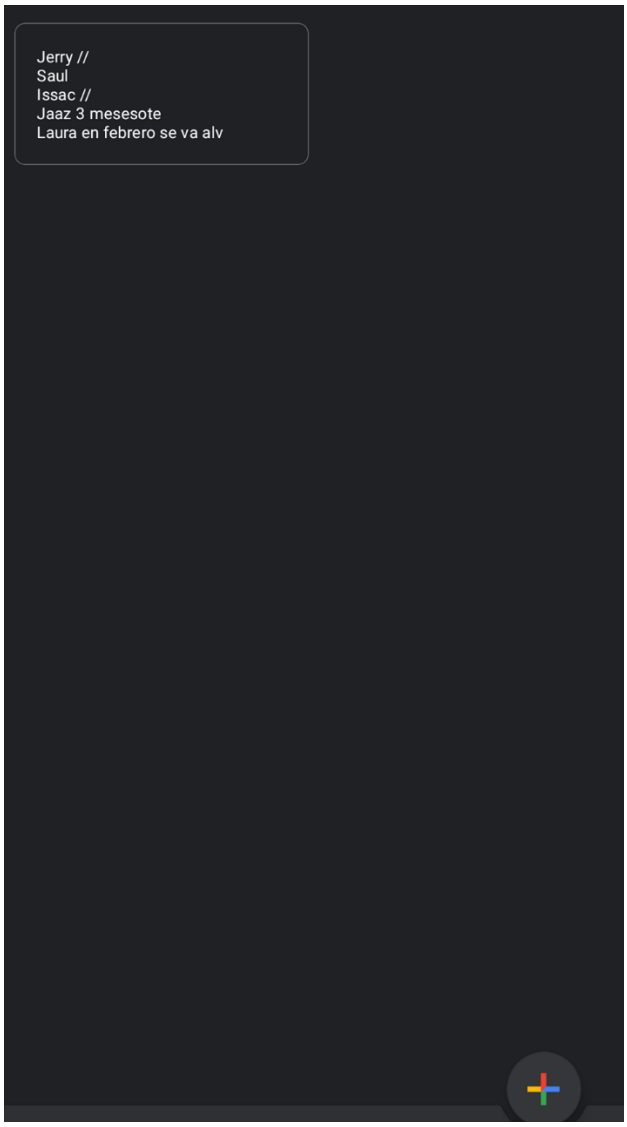


Figura 1. Visualización estimada de la aplicación.

En la figura 2 se muestra una pantalla estimada de lo que deberá poder ver el usuario al presionar el botón de “crear nueva nota”.

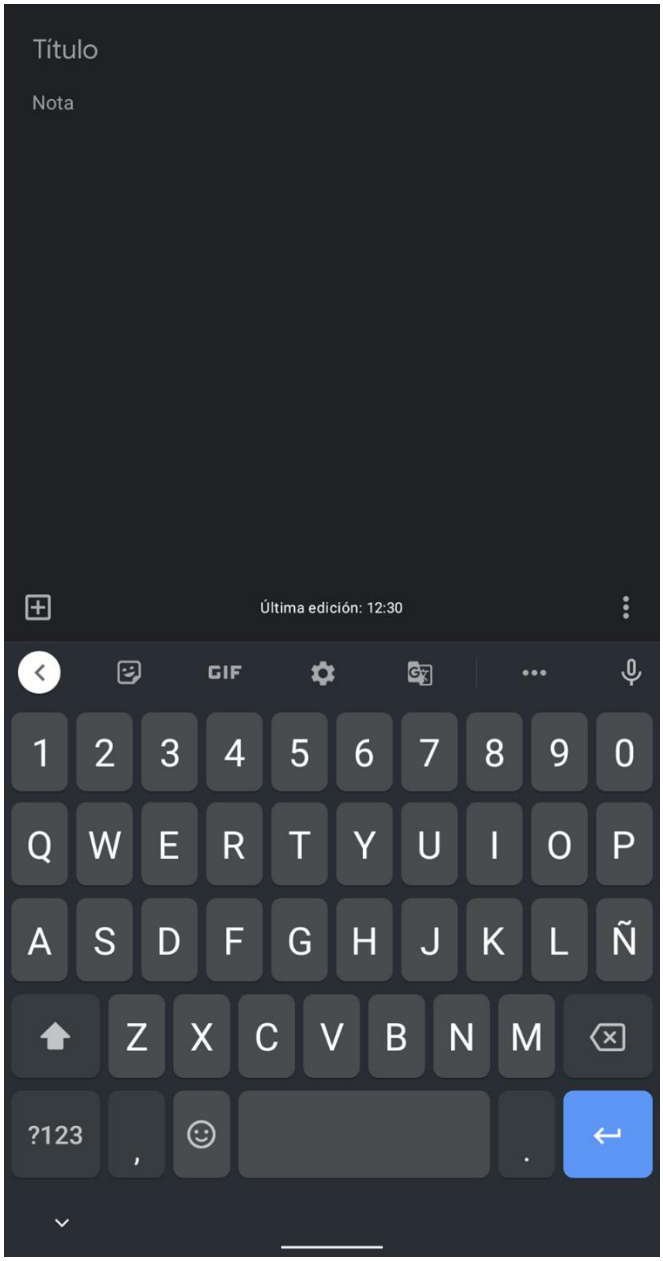
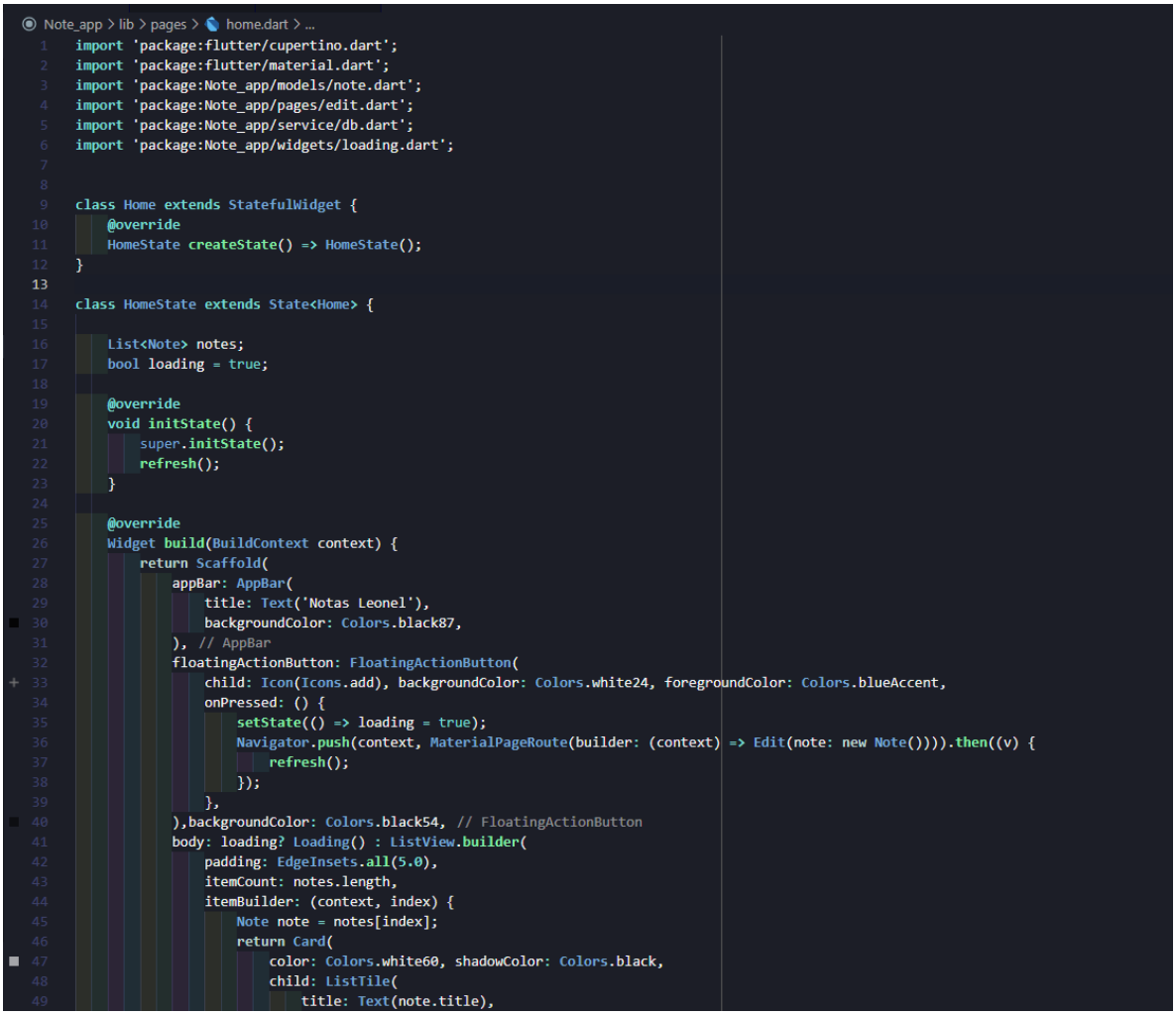


Figura 2. Creación de nueva nota.

También se desplegarán avisos en la pantalla para preguntar al usuario si está seguro de querer borrar o guardar los cambios en alguna de las notas. Esto utilizando los conocimientos previamente adquiridos sobre notificaciones en Android.

2. Desarrollo

Para el desarrollo de esta aplicación se utilizó Android Studio, donde se utilizó el siguiente layout para la página principal a la que podrá acceder el usuario



```
Note_app > lib > pages > home.dart > ...
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:Note_app/models/note.dart';
import 'package:Note_app/pages/edit.dart';
import 'package:Note_app/service/db.dart';
import 'package:Note_app/widgets/loading.dart';


class Home extends StatefulWidget {
    @override
    HomeState createState() => HomeState();
}

class HomeState extends State<Home> {

    List<Note> notes;
    bool loading = true;

    @override
    void initState() {
        super.initState();
        refresh();
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Notas Leonel'),
                backgroundColor: Colors.black87,
            ), // AppBar
            floatingActionButton: FloatingActionButton(
                child: Icon(Icons.add), backgroundColor: Colors.white24, foregroundColor: Colors.blueAccent,
                onPressed: () {
                    setState(() => loading = true);
                    Navigator.push(context, MaterialPageRoute(builder: (context) => Edit(note: new Note()))).then((v) {
                        refresh();
                    });
                },
            ),backgroundColor: Colors.black54, // FloatingActionButton
            body: loading? Loading() : ListView.builder(
                padding: EdgeInsets.all(5.0),
                itemCount: notes.length,
                itemBuilder: (context, index) {
                    Note note = notes[index];
                    return Card(
                        color: Colors.white60, shadowColor: Colors.black,
                        child: ListTile(
                            title: Text(note.title),
```

Figura 3. Layout home de la aplicación.

Se utilizaron elementos widgets para poder mostrar las notas existentes, además de una barra superior de navegación y un botón que, al presionarlo, direcciona al usuario a la página correspondiente a “Crear nota”.

En la siguiente figura se puede observar el layout utilizado para el apartado de edición y creación de nota.

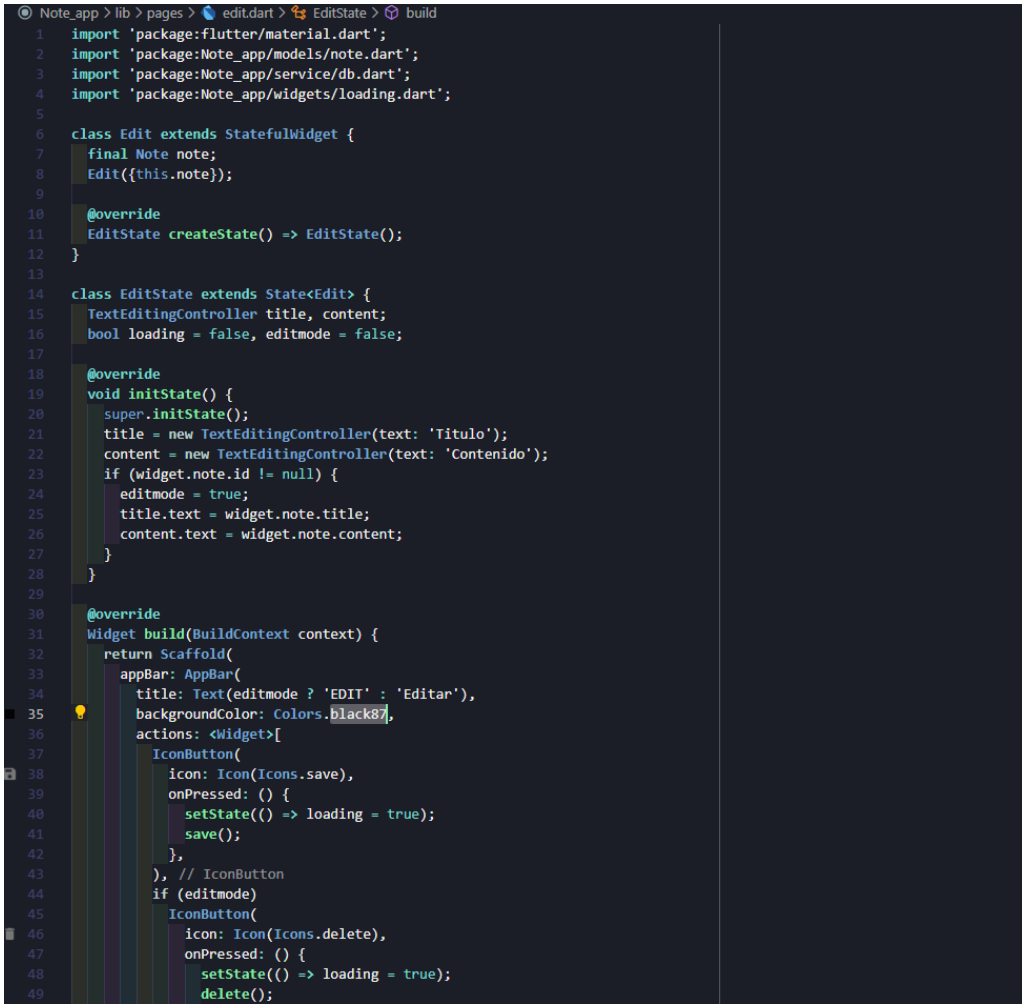


Figura 4. Layout edit.

Se creó una condicionante para que, al crear una nota, solo se accione un botón de guardar, mientras que, al entrar a la página desde la interacción con una nota existente, se acciona un botón de "eliminar" para borrar la nota seleccionada.

En la figura 5 se muestra dicha diferencia:

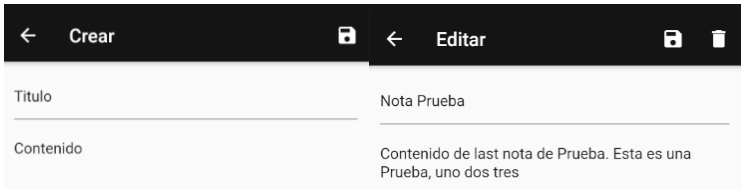


Figura 5. Diferencia en botones página crear y editar.

Como se puede observar, en la pantalla "Editar" se puede interactuar con un botón de eliminar, cosa con la que no cuenta la pantalla "Crear".

Para el diseño de las notas se utilizó el siguiente layout:

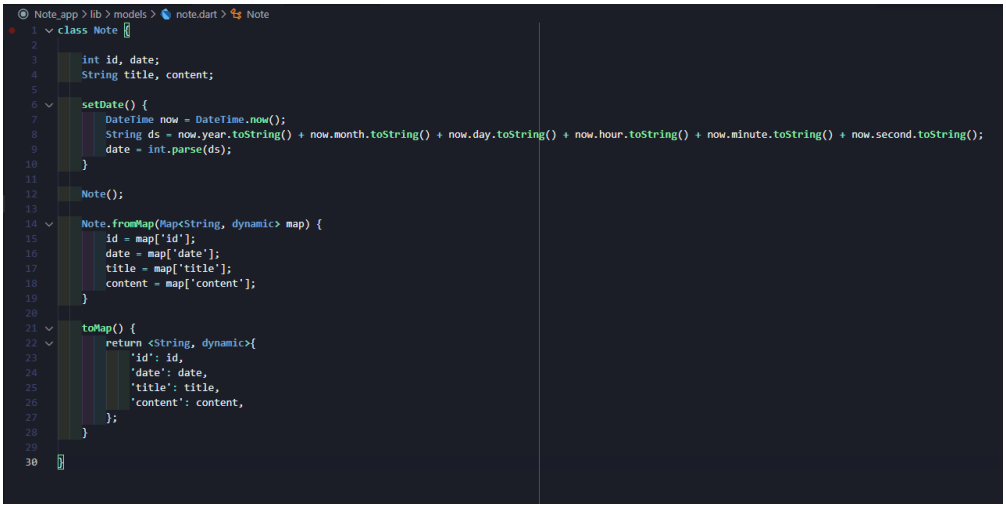


Figura 6. Layout diseño de notas.

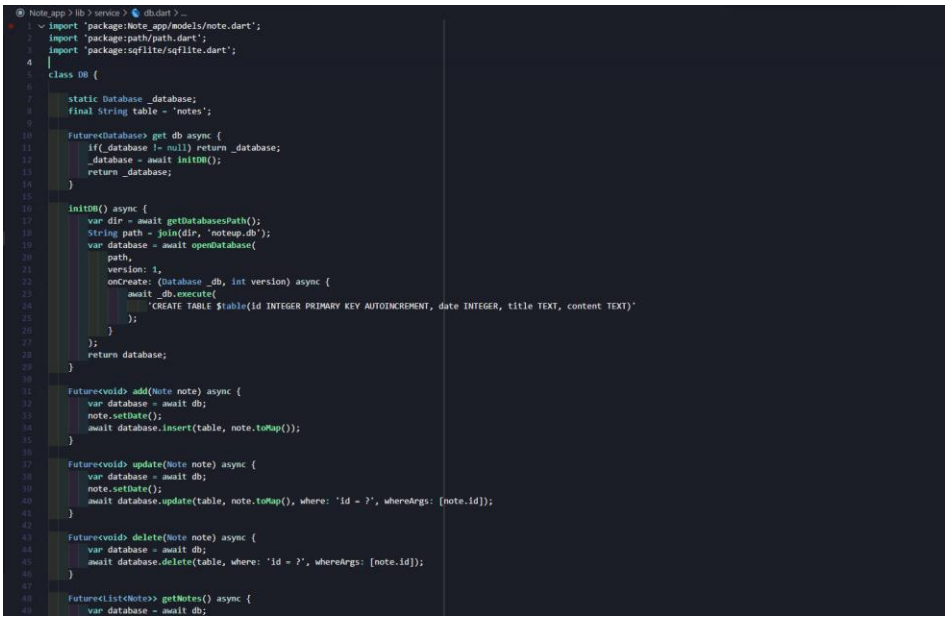
También se añadió un widget indicador de carga, el cual se muestra durante un par de segundos y para su implementación se utilizó la siguiente plantilla.

Figura 7. Plantilla widget Circular Indicador de Proceso

Figura 8. Indicador Circular de Progreso.

Para el almacenamiento de las notas y se hizo uso de una base de datos, en la cual se almacenó la información de las notas, dales como un id único, el nombre de la tabla, el

contenido, la fecha y hora de creación. En la figura 9 se puede observar el código utilizado para esta implementación.



```dart
import 'package:Note_app/models/note.dart';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';

class DB {

    static Database _database;
    final String table = 'notes';

    Future<Database> get db async {
        if(_database != null) return _database;
        _database = await initDB();
        return _database;
    }

    initDB() async {
        var dir = await getDatabasesPath();
        String path = join(dir, 'noteup.db');
        var database = await openDatabase(
            path,
            version: 1,
            onCreate: (Database _db, int version) async {
                await _db.execute(
                    'CREATE TABLE $table(id INTEGER PRIMARY KEY AUTOINCREMENT, date INTEGER, title TEXT, content TEXT)'
                );
            }
        );
        return database;
    }

    Future<void> add(Note note) async {
        var database = await db;
        note.setDate();
        await database.insert(table, note.toMap());
    }

    Future<void> update(Note note) async {
        var database = await db;
        note.setDate();
        await database.update(table, note.toMap(), where: 'id = ?', whereArgs: [note.id]);
    }

    Future<void> delete(Note note) async {
        var database = await db;
        await database.delete(table, where: 'id = ?', whereArgs: [note.id]);
    }

    Future<List<Note>> getNotes() async {
        var database = await db;
```

Figura 9. Código de implementación de la BD.

3. Resultados

A continuación, se muestran los resultados obtenidos en la aplicación, la cual se ejecutó en un simulador de Android Studio, el equipo simulado fue un Google Pixel 4, el cual cuenta con Android 10.0.

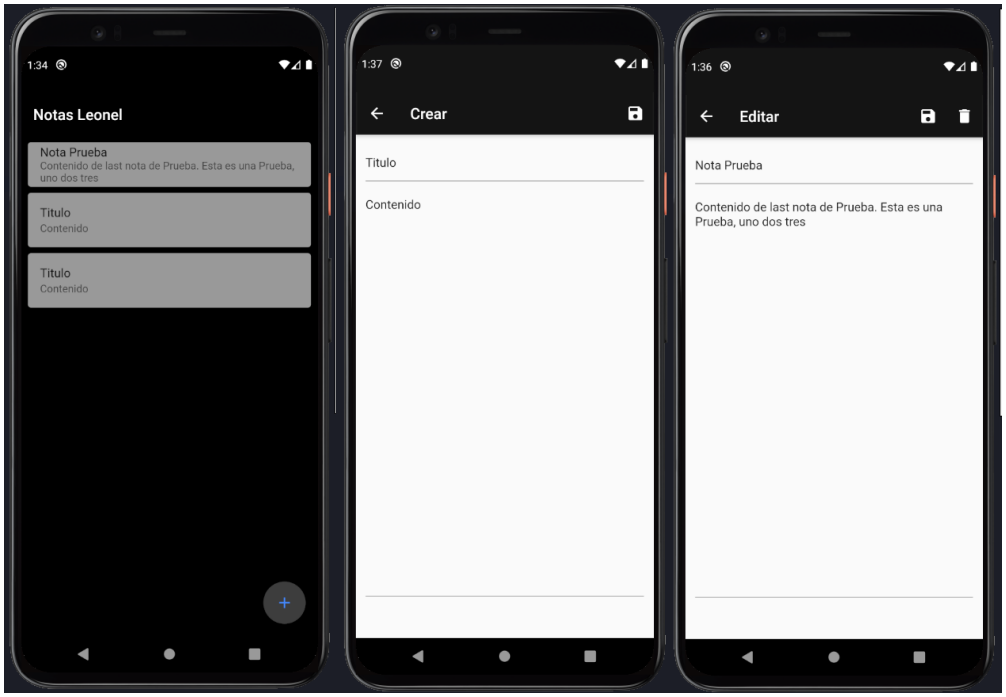


Figura 10. Funcionamiento de la aplicación de notas.

En la última figura se puede observar la pantalla inicial limpia, después de haber borrado todas las notas existentes.

Figura 11. Pantalla inicial sin notas guardadas.