

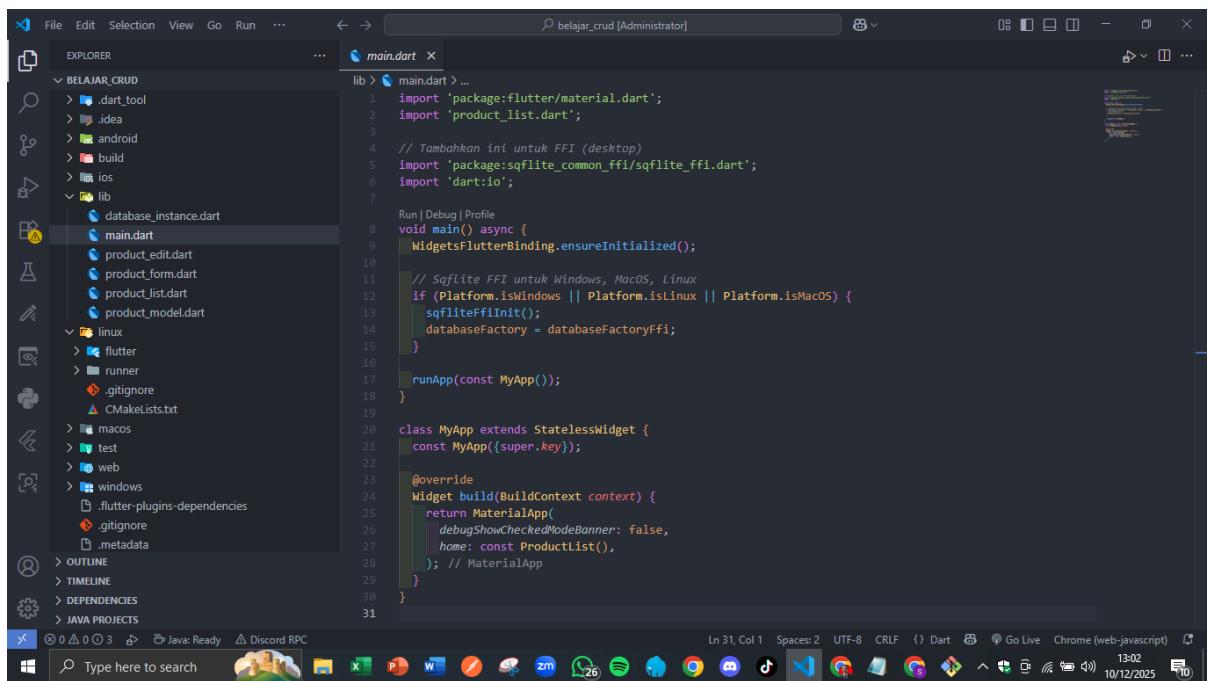
Nama : Stevanus Andika Galih Setiawan.

Kelas : RK231.

NIM : 202303110008.

Tugas : Laporan Pemrograman mobile

Link github: [https://github.com/StevanusAndika/belajar\\_crud](https://github.com/StevanusAndika/belajar_crud)



The screenshot shows a Windows desktop environment with a Visual Studio Code (VS Code) window open. The window title is "belajar\_crud [Administrator]". The left sidebar (EXPLORER) shows the project structure for a Flutter application named "BELAJAR\_CRUD". The "lib" folder contains several Dart files: database\_instance.dart, main.dart, product\_edit.dart, product\_form.dart, product\_list.dart, and product\_model.dart. It also includes platform-specific folders: android, ios, linux, macos, test, web, windows, and flutter-plugins-dependencies. The "lib/main.dart" file is selected and displayed in the main code editor area. The code implements a main() asynchronous function that initializes the Flutter binding and checks for the presence of the sqflite\_common\_ffi package. If found, it performs FFI initialization and sets the database factory. Finally, it runs the MyApp widget. The MyApp class extends StatelessWidget and returns a MaterialApp with a ProductList home screen. The status bar at the bottom of the VS Code window shows "Ln 31, Col 1" and other standard status indicators. The taskbar at the bottom of the screen displays various pinned icons for applications like Discord, Excel, Word, and Google Chrome.

```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2 import 'product_list.dart';
3
4 // Tambahkan ini untuk FFI (desktop)
5 import 'package:sqflite_common_ffi/sqflite_ffi.dart';
6 import 'dart:io';
7
8 Run | Debug | Profile
9 void main() async {
10   WidgetsFlutterBinding.ensureInitialized();
11
12   // Sqflite FFI untuk Windows, MacOs, Linux
13   if (Platform.isWindows || Platform.isLinux || Platform.isMacOS) {
14     sqfliteFFIInit();
15     databaseFactory = databaseFactoryFFI;
16   }
17
18   runApp(const MyApp());
19
20 class MyApp extends StatelessWidget {
21   const MyApp({super.key});
22
23   @override
24   Widget build(BuildContext context) {
25     return MaterialApp(
26       debugShowCheckedModeBanner: false,
27       home: const ProductList(),
28     ); // MaterialApp
29   }
30 }
```

1. Penjelasan aplikasi:

Aplikasi ini adalah aplikasi Flutter sederhana untuk mengelola data produk dengan operasi CRUD (Create, Read, Update, Delete) menggunakan SQLite sebagai database lokal. Aplikasi mendukung platform desktop melalui FFI (Foreign Function Interface).

2. Penjelasan kode:

A. Main.dart:

- File entry point aplikasi.
- Menginisialisasi binding Flutter
- Konfigurasi khusus untuk desktop dengan menggunakan sqlite\_common\_ffi agar SQLite dapat berjalan di Windows, Linux, dan macOS.
- Menjalankan aplikasi dengan MyApp sebagai root widget.
- Masalah yang mungkin terjadi: Jika FFI tidak diinisialisasi dengan benar di platform desktop, database tidak akan berfungsi.

B. Database\_instance.dart:

- Singleton class untuk mengelola koneksi database.
- Membuat/membuka database SQLite di lokasi penyimpanan aplikasi.
- Membuat tabel product dengan kolom: id, name, category, created\_at, updated\_at.
- Pattern singleton memastikan hanya satu instance database yang aktif.

C. product\_model.dart:

- Model data untuk produk.
- Konversi dari atau ke Map untuk interaksi dengan database.
- Menggunakan null safety untuk field yang opsional.

D. product\_form.dart:

- Form untuk menambah produk baru.
- Menggunakan TextEditingController untuk menangkap input.
- Membuat objek ProductModel dan menyimpan ke database.

E. product\_edit.dart:

Fungsi:

- Form untuk mengedit produk yang sudah ada.
- Mengisi form dengan data produk yang dipilih.
- Melakukan update data berdasarkan ID produk.

F. product\_list.dart:

Fungsi:

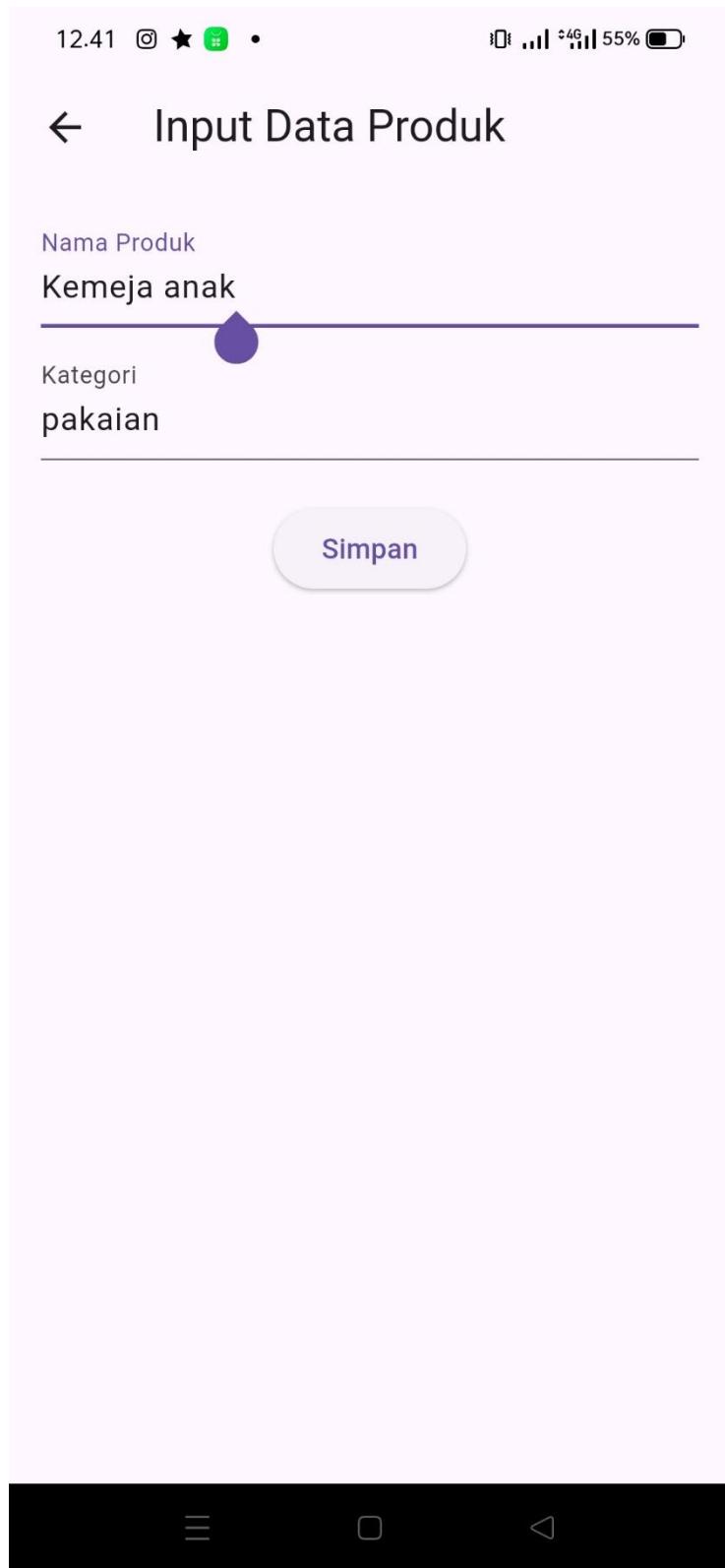
- Menampilkan daftar produk dalam ListView.
- Tombol tambah (FloatingActionButton) untuk menambah produk.
- Tombol edit dan delete pada setiap item produk.
- Auto reload data setelah CRUD

Screenshot laporan:

1. Tampilan beranda



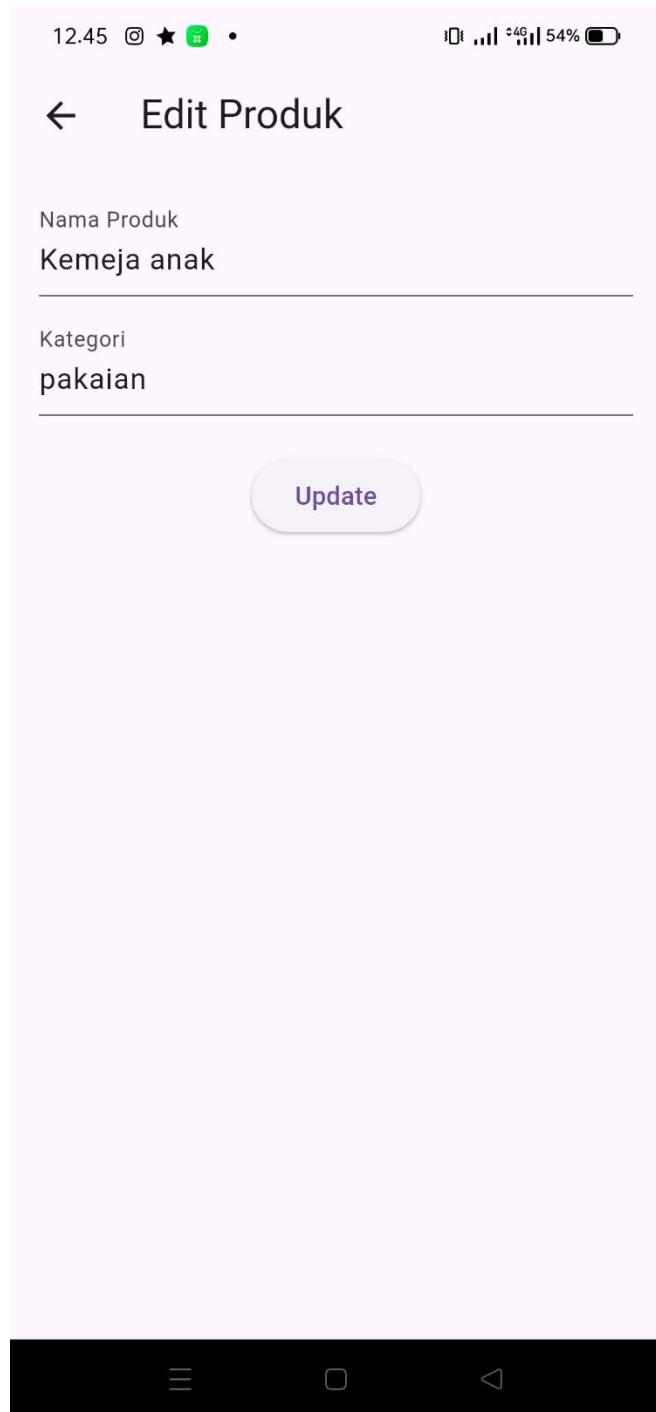
2. Tampilan form tambah produk ketika floating button (+) diklik:



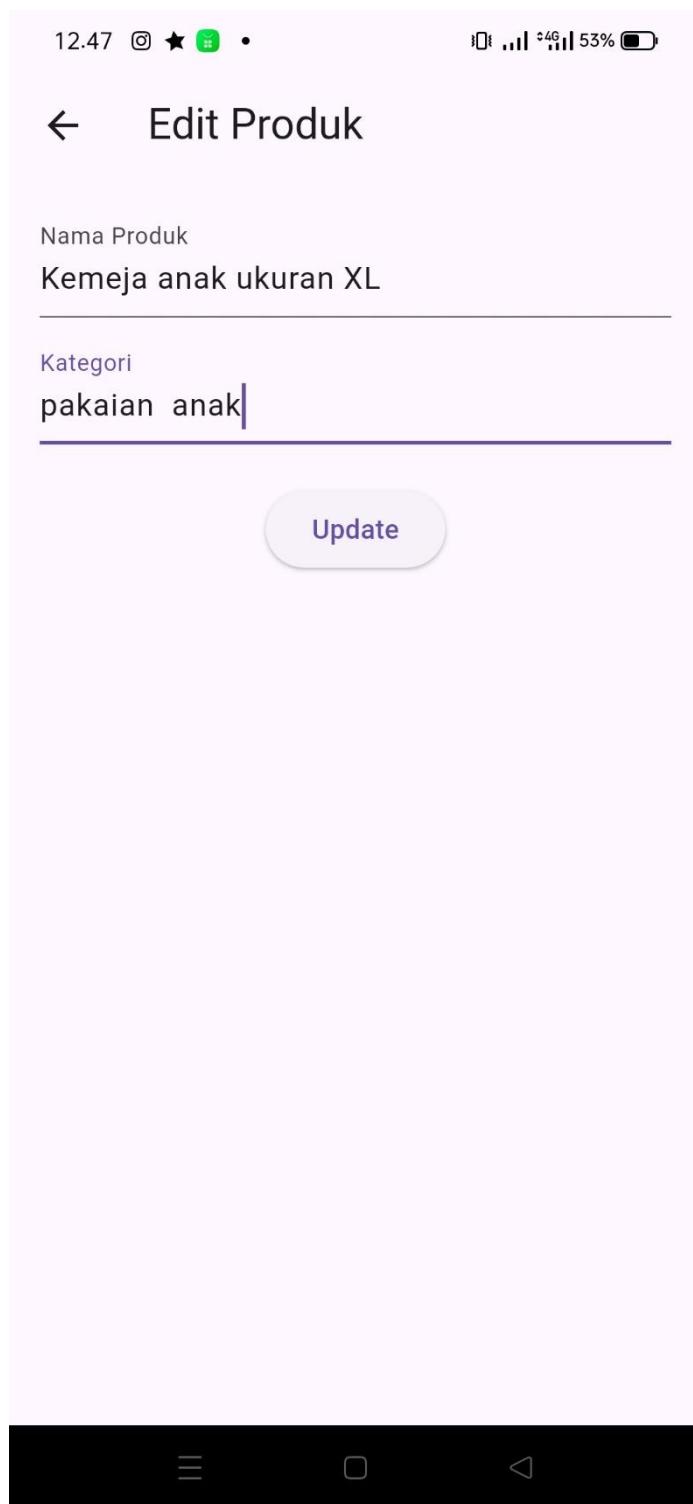
3. Tampilan ketika data produk dan kategori berhasil disimpan



4. Tampilan edit kategori dan produk



5. Tampilan edit kategori dan produk



6. Tampilan ketika berhasil data diedit dan update:



7. Tampilan Ketika icon trash ditekan dan data berhasil dihapus:



Kesimpulan:

- Operasi CRUD bisa dilakukan dengan baik,tanpa adanya error.
- Untuk SQLITE ketika dirun/debug menggunakan browser maka hanya blank putih saja,hal ini SQLITE hanya support di perangkat mobile(IOS dan Android saja).
- Architecture dan struktur data menggunakan pattern singleton dan pemisahan model dan views.
- Agar dapat program berjalan di website ,maka package hive (database nonSQL) dapat digunakan.

Referensi:

- <https://stackoverflow.com/questions/60150997/can-i-use-sqlite-flutter-for-web-apps>
- <https://ms3byoussef.medium.com/hive-in-flutter-a-detailed-guide-with-injectable-freezed-and-cubit-in-clean-architecture-c5c12ce8e00c>
- [https://pub.dev/packages/hive\\_flutter/versions](https://pub.dev/packages/hive_flutter/versions)
- [https://github.com/hivedb/hive\\_flutter](https://github.com/hivedb/hive_flutter)