# Tugas 2 Mobile Application
# Flutter Dengan SQLite


Dibuat Oleh :

| | |
|---|---|
| Nama | : Muhamad Faisal Halim |
| NIM | : 19.240.0163 |
| Kelas | : 5P43 |
| Mata Kuliah | : Mobile Application |

Sekolah Tinggi Manajemen Informatika Dan Komputer
Widya Pratama Pekalongan
Jl. Patriot No. 25 Pekalongan

## A. TUJUAN

Pembelajaran pada sub modul ini bertujuan menambah wawasan kita bahwa flutter atau khususnya aplikasi android dapat berinteraksi dengan beberapa database, misal : MySQL dan SQLite.

Ada dua perbedaan dalam penggunaan database diatas, dimana jika kita menggunakan MySQL data kita akan tersimpan didalam database MySQL di suatu server tertentu. Beda halnya dengan SQLite, data kita akan tersmpan secara local pada perangkat android kita. dan tersimpan dalam bentuk file .db.

## B. ALAT DAN BAHAN

Dalam praktikum ini kita perlu menyiapkan beberpa software, yaitu:

1. VSCode
2. Extensi Dart Dan Flutter VSCode

VSCode dan extensinya diperlukan untuk membantu kita dalam pembuatan kode aplikasinya. dan untuk testing yang sering saya lakukan adalah lengsung menggunakan perangkat android. dan media yang sering digunakan bisa melalui kabel atau wireles.

## C. STUDI KASUS

Dari praktikum pada UAS semester 4 kemarin data yang diinput akan dimasukan kedalam database. untuk itu kita memerlukan beberapa package tambahan pada proyek flutter kita, yang di tuliskan dalam file pubspec.yaml.

1. sqflite
2. path_provider

Disini kita juga memerlukan 1 class yang khusus untuk menghandle data yang kita perlukan, atau bisa dibilang sebagai model. untuk kode dari class model ini bisa dilihat pada gambar dibawah ini.

```dart
class LaptopModel {
  final int? id;
  final String nama;
  final String seri;
  final String ukuran;
  final String processor;
  final String ram;
  final String storage;

  LaptopModel({
    this.id, required this.nama, required this.seri, required this.ukuran,
    required this.processor, required this.ram, required this.storage
  });

  factory LaptopModel.fromMap(Map<String, dynamic> json) => LaptopModel(
    id: json['id'],
    nama: json['nama'],
    seri: json['seri'],
    ukuran: json['ukuran'],
    processor: json['processor'],
    ram: json['ram'],
    storage: json['storage']
  );

  Map<String, dynamic> toMap() {
    return {
      'nama': nama,
      'seri': seri,
      'ukuran': ukuran,
      'processor': processor,
      'ram': ram,
      'storage': storage
    };
  }
}
```

kita juga akan memerlukan 1 class khusus untuk menghandle CRUD kedalam database di aplikasi kita. dan kodenya sebagai berikut.

```dart
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
import 'package:path_provider/path_provider.dart';
import 'package:tugas2_0163/config/config.dart';

import 'dart:io';

import 'package:tugas2_0163/models/laptop_model.dart';


class DatabaseHelper {
  DatabaseHelper._privateConstructor();
  static final DatabaseHelper instance = DatabaseHelper._privateConstructor();

  static Database? _database;
  Future<Database> get database async => _database = await _initDatabase();

  Future<Database> _initDatabase() async {
    return await openDatabase(MyConfig.instance.db_name, version: 1, onCreate: _onCreate);
  }

  Future _onCreate(Database db, int version) async{
    await db.execute('''
      CREATE TABLE '''+ MyConfig.instance.tb_name +'''(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nama TEXT,
        seri TEXT,
        ukuran TEXT,
        processor TEXT,
        ram TEXT,
        storage TEXT
      )
    ''');
  }
  Future<List<LaptopModel>> getAllItems() async {
    Database db = await instance.database;
    var tv = await db.query(MyConfig.instance.tb_name);

    List<LaptopModel> tvlist = tv.isNotEmpty
      ? tv.map((e) => LaptopModel.fromMap(e)).toList()
      : [];

    return tvlist;
  }

  Future<int> insert(LaptopModel laptop) async {
    Database db = await instance.database;
    return db.insert(MyConfig.instance.tb_name, laptop.toMap());
  }

  Future<int> remove(int id) async {
    Database db = await instance.database;
    return await db.delete(MyConfig.instance.tb_name, where: 'id = ?', whereArgs: [id]);
  }

  Future<int> update(LaptopModel laptop) async {
    Database db = await instance.database;
    return await db.update(MyConfig.instance.tb_name, laptop.toMap(),
      where:  "id=?", whereArgs: [laptop.id]
    );
  }

}
```

Untuk sebagai pembeda dari aplikasi buatan teman saya dikelas, saya menambahkan 1 file config, file ini digunakan untuk menyimpan semua config yang diperlukan dalam pembuatasn aplikasi ini. misal dalam kasus saya, untuk menyimpan hal berikut.

```dart
class MyConfig {

  MyConfig._privateConstructor();
  static final MyConfig instance = MyConfig._privateConstructor();

  // Config App
  String get app_name => "0163 CRUD SQLITE";
  String get insert_page => "0163 ADD NEW LAPTOP";
  String get edit_page => "0163 EDIT LAPTOP";

  // Database
  String get db_name => "Laptopku.db";
  String get tb_name => "Laptops";

}
```

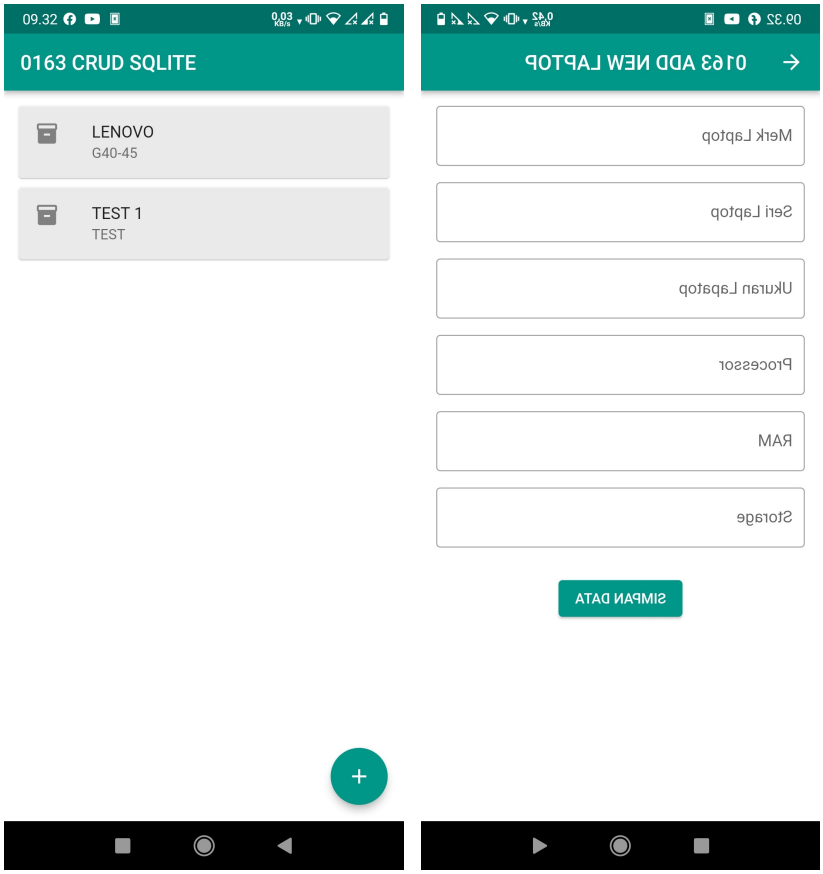
secara keseluruhan aplikasinya sama dengan apa yang dibuat pada saat UAS smt 4 kemarin, namun ada sedikit penambahan, dan perubahan.

1. pada tombol tampil data akan kita arahkan ke fungsi input data. dan setelah input sukses akan diarahkan ke halaman home dimana disana terdapat list data yang kita inputkan.

```dart
ElevatedButton(
  child: Text("SIMPAN DATA"),
  onPressed: () async {
    await DatabaseHelper.instance.insert(
      LaptopModel(
        nama: nama.text,
        seri: seri.text,
        ukuran: ukuran.text,
        processor: processor.text,
        ram: ram.text,
        storage: storage.text
      )
    );
    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => AppKu()),
      (Route<dynamic> route) => false,
    );
  },
),
```

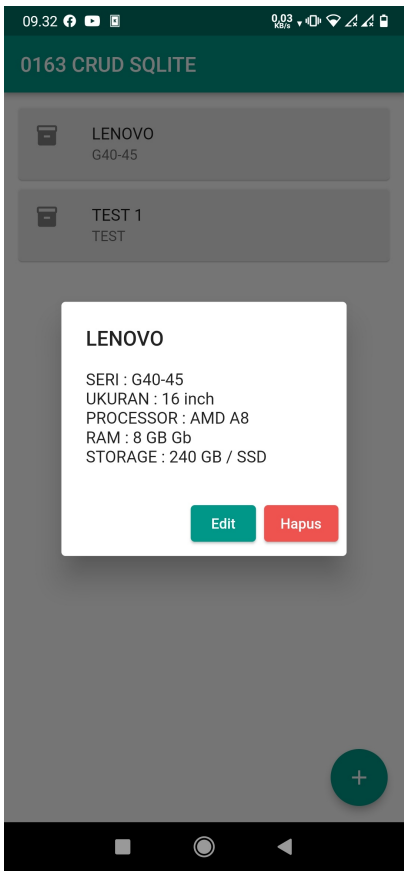
2. penambahan file dan halaman home, update.
3. perubahan route aplikasi.

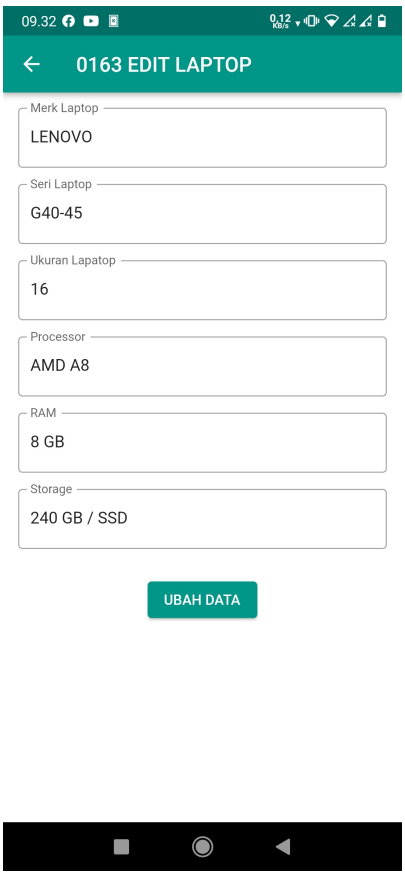
Tampilan aplikasi yang dibuat terlihat seperti berikut.









## D. ANALISIS DATA

Hanya ada 4 tipe data yang dapat disimpan dalam database SQLite ini, seperti yang dikatakan microsoft pada halaman websitenya. *SQLite hanya memiliki empat tipe data primitif: INTEGER, REAL, TEXT, dan BLOB. API yang mengembalikan nilai database sebagai objek hanya akan mengembalikan salah satu dari empat tipe ini.*

SQlite jika dibanding dengan MySQL tentu sangat jauh berbeda, MySQL memiliki banyak tipe data yang dapat disimpan olehnya, dan memerlukan koneksi khusus untuk dapat mengakses database. SQLite tidak memerlukan koneksi atau user khusus untuk dapat mengakses dabasenya. kita hanya perlu membuka file .db yang kita miliki dengan software lain. misal dilinux kita bisa menggunakan software beekeeper-studio.

## E. KESIMPULAN

Dengan adanya sqlite kita dapat menyimpan data kita secara local diaplikasi yang kita buat, dalam penerapan kasus lain. kita bisa menyimpan secara local ketika tidak ada internet, dan disimpan ke cloud ketika sudah memiliki internet, dengan aksi sync misalnya.

sekian kurang lebihnya laporan ini saya mohon maaf, kode dari aplikasi ini akan saya simpan ke https://github.com/halimkun/kuliahku

Terimakasih atas perhatianya.