



PEMROGRAMAN PERANGKAT BERGERAK

Tahun Ajaran 2024/2025

Pengembang Modul :

Yudha Islami Sulistya, S.Kom., M.Cs.

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

RICKY REVENANDO

2211104047

SE06B

DATA STORAGE (BAGIAN I)

Tujuan Praktikum
Mahasiswa mampu memahami konsep layout pada Flutter
Mahasiswa dapat mengimplementasikan desain user interface pada Flutter

1. Pengenalan SQLite

SQLite adalah database relasional yang merupakan penyimpanan data secara offline untuk sebuah mobile app (pada *local storage*, lebih tepatnya pada *cache memory* aplikasi). SQLite memiliki CRUD (***create, read, update dan delete***). Empat operasi tersebut penting dalam sebuah penyimpanan.

Untuk struktur database pada SQLite, sama seperti SQL pada umumnya, variabel dan tipe data yang dimiliki tidak jauh berbeda dengan SQL. Untuk informasi terkait basic SQL ada pada [link](#) berikut.

2. SQL Helper Dasar

Dalam Flutter, SQL Helper biasanya merujuk pada penggunaan paket seperti *sqflite* untuk mengelola database SQLite. SQL Helper merupakan class untuk membuat beberapa method yang berkaitan dengan perubahan data. *sqflite* adalah plugin Flutter yang memungkinkan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada database SQLite.

Berikut adalah langkah-langkah dasar untuk menggunakan sqflite sebagai SQL Helper di Flutter :

1. Tambahkan plugin **sqflite** dan **path** ke file pubspec.yaml. Plugin bisa didapat [disini](#).
2. Buat class baru bernama DatabaseHelper untuk mengelola database dan import package sqflite dan path di file db_helper.dart.

```
import 'package:sqflite/sqflite.dart'; import  
'package:path/path.dart';
```

```
class DatabaseHelper {                                _instance =  
  static      final      DatabaseHelper  
  DatabaseHelper._internal();  
  static Database? _database;  
}
```

3. Buat factory constructor untuk mengembalikan instance singleton dan private singleton.

```

import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

// kelas DatabaseHelper untuk mengelola database
class DatabaseHelper {
  static final DatabaseHelper _instance = DatabaseHelper._internal();
  static Database? _database;

  factory DatabaseHelper() {
    return _instance;
  }

  // Private constructor
  DatabaseHelper._internal();
}

```

4. Buat Getter untuk database.

```

Future<Database> get database async {
  if (_database != null) {
    return _database!;
  } else {
    _database = await _initDatabase();
    return _database!;
  }
}

```

5. Inisialisasi database dengan nama database yang kita mau.

```

Future<Database> _initDatabase() async {
  // mendapatkan path untuk database
}

```

```
String path = join(await getDatabasesPath(),
'my_prakdatabase.db'); //
membuka database return
await openDatabase( path,
version: 1, onCreate:
_onCreate,
);
}
```

6. Kemudian buat tabel untuk database-nya dengan *record* atau *value id, title, dan description*.

```
Future<void> _onCreate(Database db, int version) async
{
    await db.execute('' CREATE TABLE my_table(
id INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL, title TEXT, description TEXT,
createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP)
'');
}
```

7. Buat metode untuk memasukkan data ke dalam tabel.

```
Future<int> insert(Map<String, dynamic> row) async {
Database db = await database; return await
db.insert('my_table', row);
}
```

8. Lalu, metode untuk mengambil semua data dari tabel.

```
Future<List<Map<String, dynamic>>> queryAllRows() async {
Database db = await database; return await
db.query('my_table');
}
```

9. Buat metode untuk memperbarui data dalam tabel.

```
Future<int> update(Map<String, dynamic> row) async {
```

```

        Database db = await database;
        int id = row['id'];
        return await db.update('my_table', row, where: 'id = ?',
        whereArgs: [id]);
    }

```

10. Diakhiri dengan metode untuk menghapus data dari tabel.

```

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

```

a) Read

Di package sqflite, kita akan menggunakan query() untuk membaca data yang ada pada database. Sqflite di Flutter kita dapat membuat query menggunakan bermacam-macam perintah, seperti **where**, **groupBy**, **orderBy**, dan **having**. Selain itu, kita juga bisa membaca satu data atau banyak data sekaligus. Berikut barisan kode read pada sqflite :

Membaca semua data

```

Future<List<Map<String, dynamic>>> queryAllRows() async {
    Database db = await database;    return await
    db.query('my_table');
}

```

Membaca satu data melalui id

```

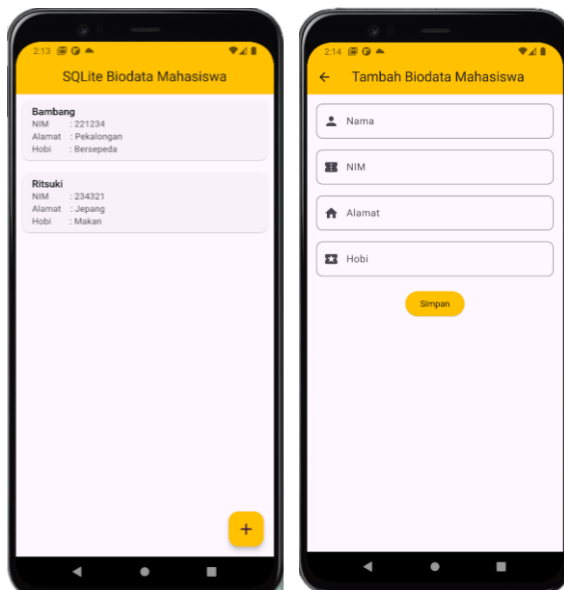
Future<List<Map<String, dynamic>>> getItem(int id) async {
    Database db = await database;
    Return await db.query('my_table', row, where: "id = ?", whereArgs:
    [id], limit: 1);
}

```

Tugas Mandiri (Unguided)

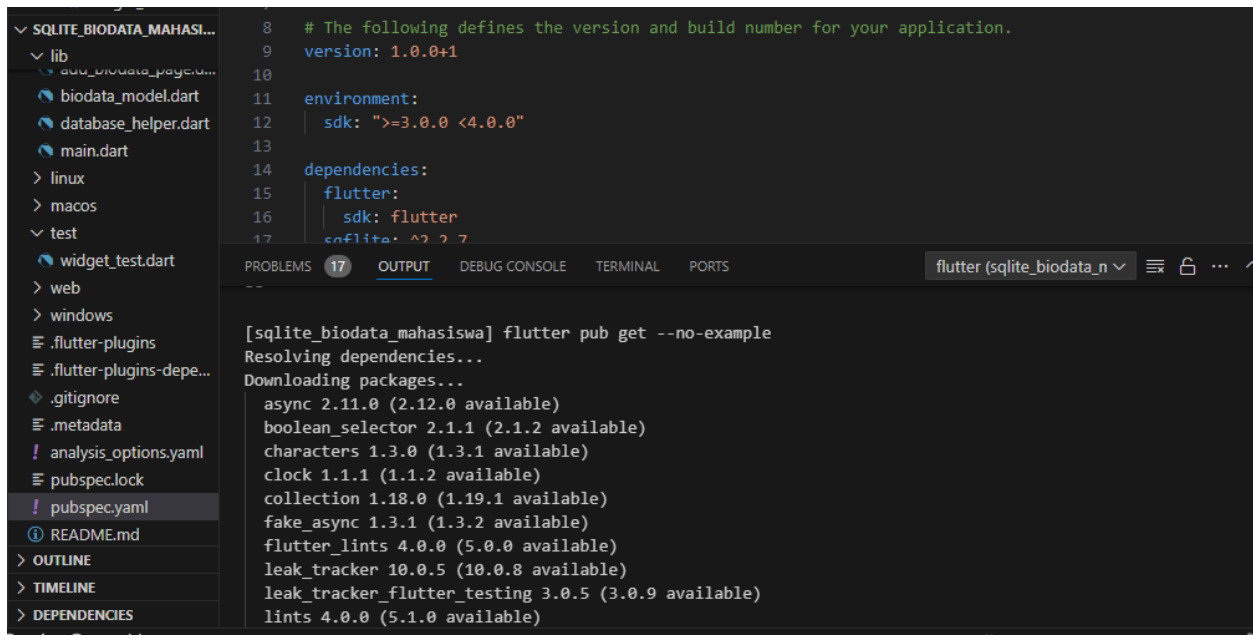
1. (Soal) Buatlah sebuah project aplikasi Flutter dengan SQLite untuk menyimpan data biodata mahasiswa yang terdiri dari nama, NIM, domisili, dan hobi. Data yang dimasukkan melalui form akan ditampilkan dalam daftar di halaman utama. Alur Aplikasi:

- a) Form Input: Buat form input untuk menambahkan biodata mahasiswa, dengan kolom:
 - Nama
 - Nim
 - Alamat
 - Hobi
- b) Tampilkan Daftar Mahasiswa: Setelah data berhasil ditambahkan, tampilkan daftar semua data mahasiswa yang sudah disimpan di halaman utama.
- c) Implementasikan fitur Create (untuk menyimpan data mahasiswa) dan Read (untuk menampilkan daftar mahasiswa yang sudah disimpan).
- d) Contoh output:



Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

OUTPUT



```
8 # The following defines the version and build number for your application.
9 version: 1.0.0+1
10
11 environment:
12   sdk: ">=3.0.0 <4.0.0"
13
14 dependencies:
15   flutter:
16     sdk: flutter
17   sqflite: ^2.2.7
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL PORTS flutter (sqlite_biodata_n)

[sqlite_biodata_mahasiswa] flutter pub get --no-example
Resolving dependencies...
Downloading packages...
async 2.11.0 (2.12.0 available)
boolean_selector 2.1.1 (2.1.2 available)
characters 1.3.0 (1.3.1 available)
clock 1.1.1 (1.1.2 available)
collection 1.18.0 (1.19.1 available)
fake_async 1.3.1 (1.3.2 available)
flutter_lints 4.0.0 (5.0.0 available)
leak_tracker 10.0.5 (10.0.8 available)
leak_tracker_flutter_testing 3.0.5 (3.0.9 available)
lints 4.0.0 (5.1.0 available)

←

Tambah Biodata Mahasiswa

DEBUG

Nama

ric

NIM

2211104047

Alamat

BLABLABLA

Hobi

BLABLABLA

Simpan

1. Struktur Aplikasi

Aplikasi menggunakan **SQLite** sebagai database lokal untuk menyimpan data. Flutter berfungsi sebagai kerangka kerja utama untuk membangun antarmuka pengguna.

Komponen Utama:

- **Database SQLite:** Untuk menyimpan data mahasiswa secara permanen di perangkat.
- **Form Input:** Untuk menambahkan data mahasiswa (Nama, NIM, Alamat, Hobi).
- **Daftar Data:** Menampilkan data mahasiswa yang telah disimpan.
- **CRUD (Create dan Read):** Implementasi untuk menambah dan membaca data dari database.

2. Fungsi Utama

a) Menyimpan Data (Create)

Data yang dimasukkan pengguna di **form input** (Nama, NIM, Alamat, dan Hobi) akan disimpan ke SQLite menggunakan fungsi insert. Berikut alurnya:

1. Pengguna mengisi semua kolom di form.
2. Saat tombol **Simpan** ditekan:
 - Data akan disimpan ke tabel di database SQLite.
 - Aplikasi akan kembali ke halaman utama dan memperbarui daftar data.

b) Membaca Data (Read)

Data yang sudah disimpan di database akan dibaca menggunakan perintah query SQLite, lalu ditampilkan di halaman utama. Setiap data ditampilkan dalam bentuk **card** atau list tile dengan informasi:

- Nama
- NIM
- Alamat
- Hobi

3. Struktur Database SQLite

Database terdiri dari satu tabel yang disebut biodata_mahasiswa. Kolom-kolom di tabel ini adalah:

- id (integer, primary key, auto-increment)
- nama (text)
- nim (text)
- alamat (text)
- hobi (text)