PRATIK APLIKASI MOBILE

"Sync dan Async Process & Sqflite"
Modul Praktikum 9

Disusun oleh :

Diah Munica Nawang
V3922015 / TI D

Dosen :
Trisna Ari Roshinta, S.S.T., M.T

PS D-III TEKNIK INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS SEBELAS MARET
2023

Langkah Praktikum

1. Implementasi Sync Process

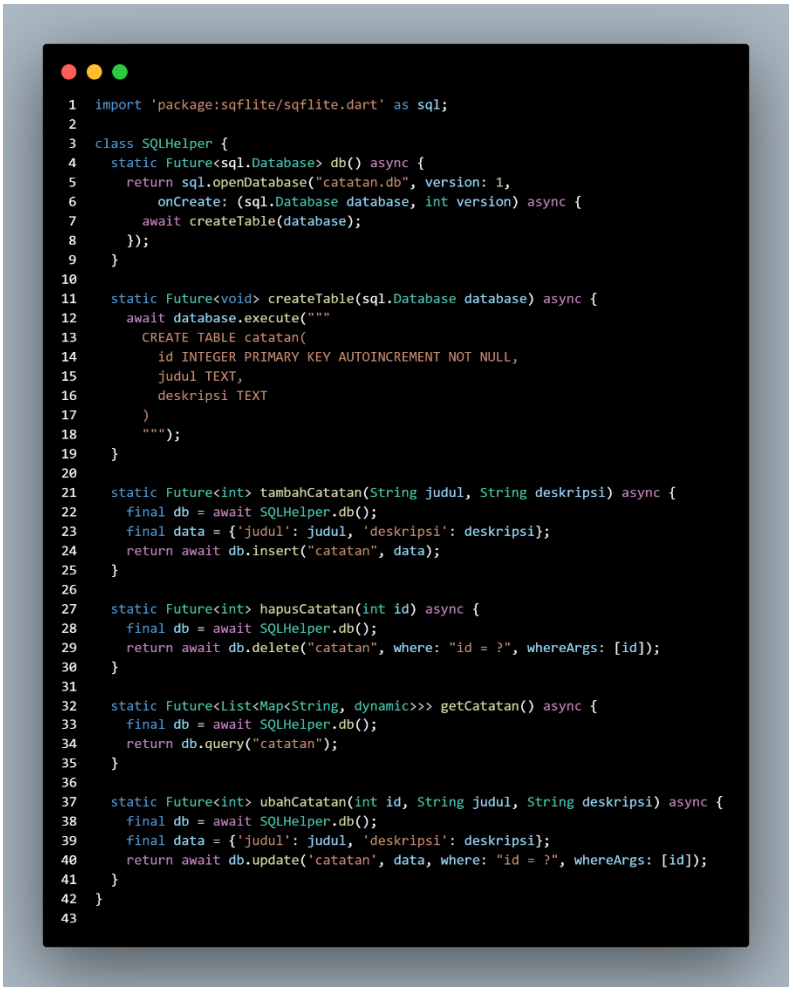
Deskripsi

Pada langkah ini, saya akan membuat fungsi untuk mengakses database SQLite secara sinkron dan melakukan operasi CRUD sederhana.

Langkah-langkah

1. Inisialisasi database SQLite.
2. Tambahkan data ke database.
3. Ambil data dari database.
4. Update data dalam database.
5. Hapus data dari database.

Script :



```dart
import 'package:sqflite/sqflite.dart' as sql;

class SQLHelper {
  static Future<sql.Database> db() async {
    return sql.openDatabase("catatan.db", version: 1,
        onCreate: (sql.Database database, int version) async {
      await createTable(database);
    });
  }

  static Future<void> createTable(sql.Database database) async {
    await database.execute("""
      CREATE TABLE catatan(
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
        judul TEXT,
        deskripsi TEXT
      )
      """);
  }

  static Future<int> tambahCatatan(String judul, String deskripsi) async {
    final db = await SQLHelper.db();
    final data = {'judul': judul, 'deskripsi': deskripsi};
    return await db.insert("catatan", data);
  }

  static Future<int> hapusCatatan(int id) async {
    final db = await SQLHelper.db();
    return await db.delete("catatan", where: "id = ?", whereArgs: [id]);
  }

  static Future<List<Map<String, dynamic>>> getCatatan() async {
    final db = await SQLHelper.db();
    return db.query("catatan");
  }

  static Future<int> ubahCatatan(int id, String judul, String deskripsi) async {
    final db = await SQLHelper.db();
    final data = {'judul': judul, 'deskripsi': deskripsi};
    return await db.update('catatan', data, where: "id = ?", whereArgs: [id]);
  }
}

```

Penjelasan Script :

Import Library:

```
dart
• import 'package:sqflite/sqflite.dart' as sql;
```

Baris ini mengimpor library `sqflite` dan memberikannya alias `sql` untuk digunakan dalam script. Library ini digunakan untuk mengakses dan mengelola database SQLite.

- **Class SQLHelper**:

Ini adalah definisi kelas utama yang digunakan untuk mengelola akses ke database SQLite. Kelas ini memiliki beberapa metode yang memfasilitasi operasi CRUD (Create, Read, Update, Delete) pada tabel "catatan" dalam database SQLite.

- **Static Method db()**:

  - Metode ini digunakan untuk membuka atau membuat database SQLite.
  - Itu mengembalikan objek `Future<sql.Database>`, yang akan berisi koneksi ke database.
  - Jika database "catatan.db" belum ada, maka metode `onCreate` akan dipanggil untuk membuat tabel "catatan".

- **Static Method createTable()**:

  - Metode ini digunakan untuk membuat tabel "catatan" jika belum ada.
  - Ini menggunakan perintah SQL `CREATE TABLE` untuk mendefinisikan skema tabel.

- **Static Method tambahCatatan()**:

  - Metode ini digunakan untuk menambahkan catatan baru ke tabel "catatan".
  - Menggunakan objek `db` untuk melakukan operasi `insert` yang mengembalikan ID dari catatan yang ditambahkan.

- **Static Method hapusCatatan()**:

  - Metode ini digunakan untuk menghapus catatan dari tabel "catatan" berdasarkan ID.
  - Menggunakan objek `db` untuk melakukan operasi `delete` yang mengembalikan jumlah catatan yang dihapus.

- **Static Method getCatatan()**:

  - Metode ini digunakan untuk mengambil seluruh catatan dari tabel "catatan".

- Menggunakan objek `db` untuk melakukan operasi `query` yang mengembalikan daftar `Map<String, dynamic>` yang berisi catatan-catatan.

- **Static Method ubahCatatan()**:

  - Metode ini digunakan untuk mengubah catatan dalam tabel "catatan" berdasarkan ID.
  - Menggunakan objek `db` untuk melakukan operasi `update` yang mengembalikan jumlah catatan yang diubah.

2. Implementasi Async Process

Deskripsi

Pada langkah ini, saya membuat fungsi untuk mengakses database SQLite secara asinkron menggunakan `async` dan `await`.

Langkah-langkah

1. Inisialisasi database SQLite secara asinkron.
2. Tambahkan data ke database dengan operasi asinkron.
3. Ambil data dari database dengan operasi asinkron.
4. Update data dalam database dengan operasi asinkron.
5. Hapus data dari database dengan operasi asinkron.

Script :

```dart
import 'package:flutter/material.dart';
import 'package:mobile9/sql_helper.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Your Todo List',
      home: const MyHomePage(title: 'Your Todo List'),
    );
  }
}
```

```dart

class _MyHomePageState extends State<MyHomePage> {
  TextEditingController judulController = TextEditingController();
  TextEditingController deskripsiController = TextEditingController();

  List<Map<String, dynamic>> catatan = [];

  void refreshCatatan() async {
    final data = await SQLHelper.getCatatan();
    setState(() {
      catatan = data;
    });
  }

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

  Future<void> tambahCatatan() async {
    await SQLHelper.tambahCatatan(judulController.text, deskripsiController.text);
    refreshCatatan();
  }

  Future<void> hapusCatatan(int id) async {
    await SQLHelper.hapusCatatan(id);
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
      content: Text("Berhasil Dihapus"),
    ));
    refreshCatatan();
  }

  Future<void> ubahCatatan(int id) async {
    await SQLHelper.ubahCatatan(id, judulController.text, deskripsiController.text);
    refreshCatatan();
  }

  void modalForm(id) async {
    if (id != null) {
      final dataCatatan = catatan.firstWhere((item) => item['id'] == id);

      judulController.text = dataCatatan['judul'];
      deskripsiController.text = dataCatatan['deskripsi'];
    }
    showModalBottomSheet(
      context: context,
      builder: (context) => Container(
        padding: const EdgeInsets.all(15),
        width: double.infinity,
        height: 800,
        child: SingleChildScrollView(
          child: Column(
            children: [
              TextField(
                controller: judulController,
                decoration: const InputDecoration(hintText: "Judul"),
              ),
              const SizedBox(
                height: 10,
              ),
              TextField(
                controller: deskripsiController,
                decoration: const InputDecoration(hintText: "Deskripsi"),
              ),
              const SizedBox(
                height: 20,
              ),
              ElevatedButton(
                onPressed: () async {
                  if (id == null) {
                    await tambahCatatan();
                    print("Add");
                  } else {
                    print("Update");
                    await ubahCatatan(id);
                  }
                  judulController.text = '';
                  deskripsiController.text = '';
                  Navigator.pop(context);
                },
                child: Text(id == null ? 'Add' : 'Ubah'),
              ),
            ],
          ),
        ),
      ),
    );
  }

```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: ListView.builder(
      itemCount: catatan.length,
      itemBuilder: (context, index) => Card(
        child: ListTile(
          title: Text(catatan[index]['judul']),
          subtitle: Text(catatan[index]['deskripsi']),
          trailing: SizedBox(
            width: 100,
            child: Row(
              children: [
                IconButton(
                  onPressed: () => modalForm(catatan[index]['id']),
                  icon: const Icon(Icons.edit),
                ),
                IconButton(
                  onPressed: () {
                    hapusCatatan(catatan[index]['id']);
                  },
                  icon: const Icon(Icons.delete),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        modalForm(null);
      },
      tooltip: 'Tambah Catatan',
      child: const Icon(Icons.add),
    ),
  );
}
}
```

Penjelasan Script :

1. **Import Libraries**:
   - Script ini mengimpor dua library utama, yaitu `flutter/material.dart` dan `mobile9/sql_helper.dart`. Library `flutter/material.dart` digunakan untuk membuat tampilan antarmuka pengguna (UI), sementara `mobile9/sql_helper.dart` adalah library yang berisi kode untuk berinteraksi dengan database SQLite.
2. **Main Function**:
   - Fungsi `main` merupakan fungsi utama dalam aplikasi Flutter.
   - Fungsi ini memanggil `MyApp`, yang merupakan widget utama aplikasi, dan memulai aplikasi.
3. **MyApp Class**:
   - Kelas `MyApp` adalah widget utama yang mewakili seluruh aplikasi.
   - Kelas ini mengonfigurasi dan menginisialisasi aplikasi Flutter.

- `MyApp` menggunakan `MaterialApp` sebagai root widget untuk menentukan judul aplikasi dan halaman awalnya.

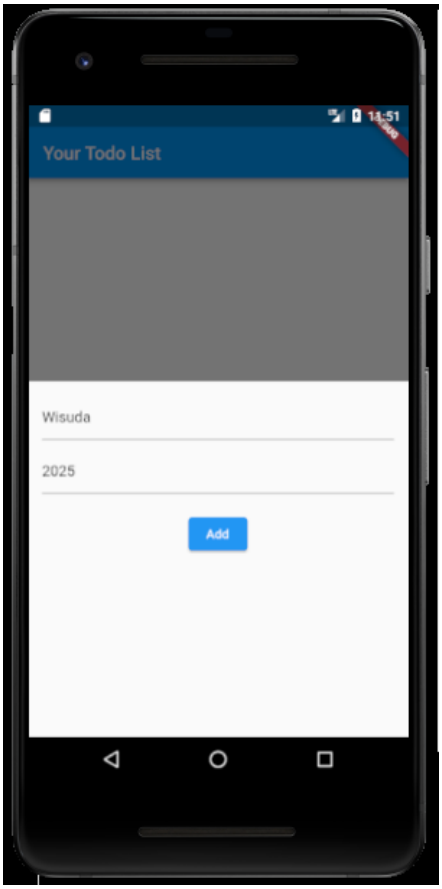
4. **MyHomePage Class**:
   - Kelas `MyHomePage` adalah widget yang mewakili halaman utama aplikasi.
   - Kelas ini menerima properti `title` yang digunakan untuk menampilkan judul halaman.
5. **_MyHomePageState Class**:
   - Kelas `_MyHomePageState` adalah state dari `MyHomePage`.
   - Ini mendefinisikan elemen-elemen UI seperti `TextField`, `ListView.builder`, dan `FloatingActionButton`.
6. **Controller Text Fields**:
   - `judulController` dan `deskripsiController` digunakan untuk mengontrol input teks pada judul dan deskripsi catatan.
7. **List catatan**:
   - Variabel `catatan` adalah daftar catatan yang akan ditampilkan dalam aplikasi.
   - Ini diperbarui setiap kali ada perubahan data dalam database SQLite.
8. **refreshCatatan() Method**:
   - Metode ini digunakan untuk mengambil data catatan dari database SQLite dan memperbarui tampilan aplikasi.
   - Ini dipanggil ketika aplikasi dimulai dan setelah operasi tambah, ubah, atau hapus catatan.
9. **initState() Method**:
   - Metode ini digunakan untuk menginisialisasi `catatan` ketika aplikasi pertama kali dimulai.
10. **tambahCatatan() Method**:
    - Metode ini digunakan untuk menambahkan catatan baru ke database SQLite.
    - Setelah penambahan, data di-refresh dan field input dikosongkan.
11. **hapusCatatan() Method**:
    - Metode ini digunakan untuk menghapus catatan dari database SQLite berdasarkan ID.
    - Ini juga menampilkan pesan notifikasi menggunakan `ScaffoldMessenger`.
12. **ubahCatatan() Method**:
    - Metode ini digunakan untuk mengubah catatan yang ada dalam database SQLite berdasarkan ID.
    - Setelah perubahan, data di-refresh.
13. **modalForm() Method**:
    - Metode ini menampilkan modal bottom sheet yang memungkinkan pengguna untuk menambah atau mengubah catatan.
    - Jika `id` tidak null, itu berarti operasi ubah, dan data catatan diisi ke dalam input fields.
    - Modal bottom sheet berisi input fields dan tombol "Add" atau "Ubah".
14. **build() Method**:
    - Metode ini digunakan untuk membangun tampilan utama aplikasi.

- Ini berisi `AppBar` dengan judul, daftar catatan yang ditampilkan dalam `ListView.builder`, dan `FloatingActionButton` untuk menambah catatan baru.

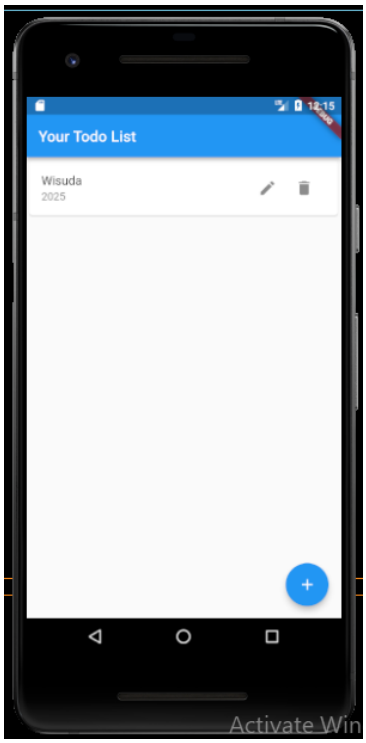
3. Screenshots

Tampilan Awal :

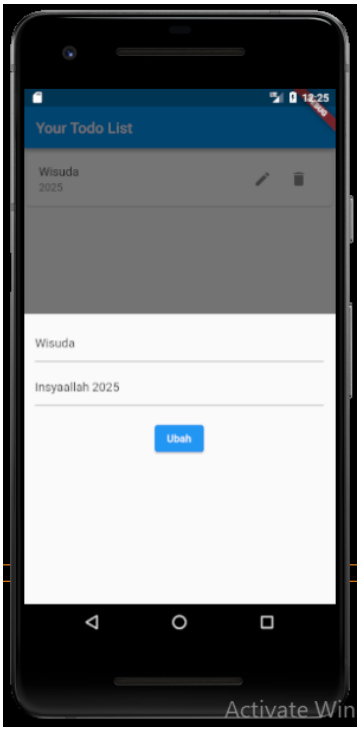
Create ToDo :



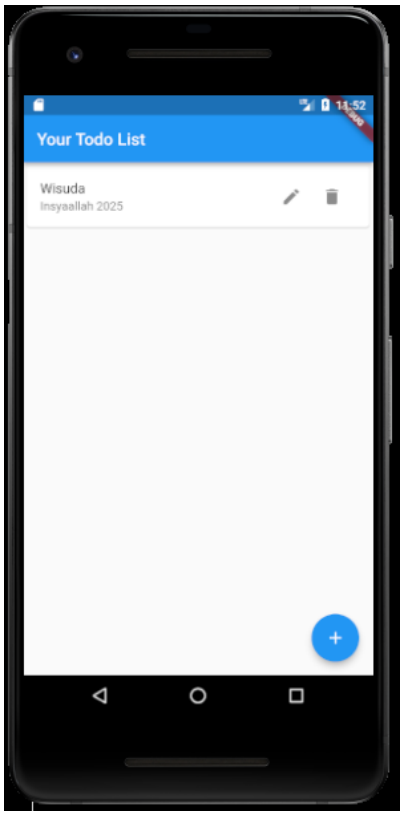
ToDo Berhasil Dibuat :



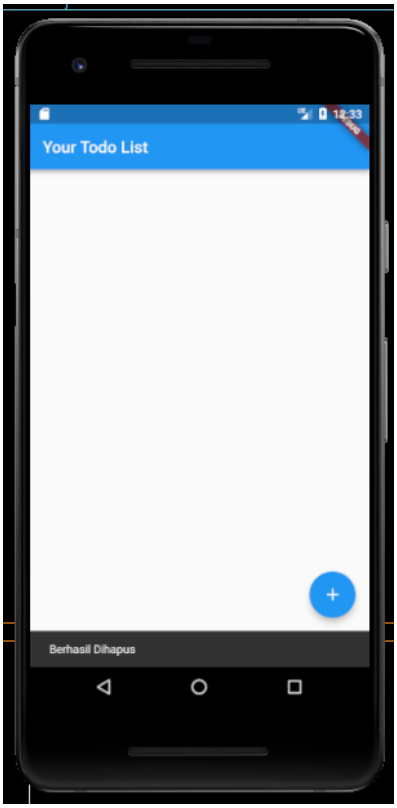
Edit ToDo :



ToDo Berhasil diedit :



Hapus ToDo Berhasil :



Hasil dan Kesimpulan

Dengan demikian, praktikum ini memberikan dasar yang kuat untuk memahami konsep sinkronisasi, asinkronisasi, dan interaksi dengan database dalam pengembangan aplikasi mobile menggunakan Flutter.