Modul 3

Using Sq(F)Lite To Store Data in Local Database

Using Sq(F)Lite to CRUD app data.

Module Overview

Mengenal metode penyimpanan data lokal menggunakan Sq(F)Lite pada aplikasi flutter dan bagaimana cara menerapkannya.

Module Objectives

Setelah mempelajari dan mempraktikkan modul ini, mahasiswa diharapkan dapat:

- Mengetahui apa yang dimaksud penyimpanan data lokal
- Menerapkan Sq(F)Lite sesuai dengan penggunaannya

Pada pembelajaran sebelumnya Anda telah belajar cara menyimpan data ke perangkat mobile Anda dengan menggunakan SharedPreferences yang merupakan metode penyimpanan data dengan menggunakan pasangan key dan value dari data yang akan disimpan. Anda juga telah mengetahui bahwa penggunaan

SharedPreferences hanya disarankan untuk menyimpan data-data sederhana serta data-data yang tidak memiliki struktur data. Lalu bagaimana jika Anda ingin menyimpan data dalam jumlah yang banyak serta memiliki struktur data seperti misalnya No. ID, Nama, Alamat, No. Telepon, dan lain sebagainya? Di sinilah Anda akan belajar menggunakan metode penyimpanan data pada database lokal dengan menggunakan Sq(F)Lite.

Sq(F)Lite merupakan salah satu pustaka yang dapat Anda gunakan untuk menyimpan data pada perangkat mobile Anda dengan menggunakan database lokal. Terdapat beberapa fitur utama dari pustaka ini di antaranya:

- Ringan dan cepat
- Dapat berdiri sendiri
- Handal dalam menyimpan data
- Menyediakan fitur yang lengkap

Jika Anda sebelumnya sudah pernah menggunakan database SQL, maka Anda hanya perlu melakukan sedikit penyesuaian untuk mulai menggunakan pustaka Sq(F)Lite ini. Sq(F)Lite adalah pilihan yang sangat baik untuk menyimpan data di perangkat mobile Anda karena mudah diterapkan, aman, lintas platform, dan ringkas.

Instalasi Sq(F)Lite

Pada contoh ini, kita akan mencoba menyertakan Sq(F)Lite dalam proyek kita. Jadi, di file pubspec.yaml / pubspec assist Anda, tambahkan dependensi, sebagai berikut:

```
sqflite:
```

Selain itu kita juga akan melakukan manipulasi path untuk mendapatkan lokasi path dari database yang akan dipergunakan. Untuk itu tambahkan juga dependensi berikut di pubspec.yaml Anda:

```
path:
```

Untuk memudahkan anda dalam menetukan versi dari Sqflite dan path, anda dapat tidak mendeklarasikan versi yang digunakan. pubspec.yaml akan mentukan versi terbaik untuk kedua library tersebut dapat anda gunakan.

Untuk memudahkan anda dalam membaca data terbaru, anda juga dapat menggunakan provider dalam memudahkan anda membaca data terbaru. Untuk itu tambahkan juga dependensi berikut di pubspec.yaml Anda:

```
  provider: ^6.0.3
```

Setelah itu, anda dapat membangun tampilan front-end sebagai berikut, dimulai dari class. Untuk memodelkan database

- **ShoppingList.dart**

```dart
class ShoppingList {
  int id;
  String name;
  int sum;

  ShoppingList(this.id, this.name, this.sum);

  Map<String, dynamic> toMap() {
    return {'id': id, 'name': name, 'sum': sum};
  }

  @override
  String toString() {
    return "id : $id\nname : $name\nsum : $sum";
  }
}

```

kita akan memerlukan metode toMap() yang akan mengembalikan tipe data Map dengan key String, dan value dinamis. Map untuk key dengan tipe data string ini akan digunakan untuk menentukan kolom pada sebuah tabel, sedangkan value akan digunakan sebagai data yang akan diisi ke dalam tabel. Untuk tipe data dalam table

dibuat menjadi dinamis, karena di dalam table, nilai tersebut dapat memiliki tipe yang berbeda.

Untuk memudahkan anda dalam mengupdate tampilan ui, anda dapat menggunakan provider sebagai berikut:

- **MyProvider.dart**

```dart
import 'package:flutter/material.dart';
import 'package:my_sql_db/Praktek/model/ShoppingList.dart';

class ListProductProvider extends ChangeNotifier {
  List<ShoppingList> _shoppingList = [];
  List<ShoppingList> get getShoppingList => _shoppingList;
  set setShoppingList(value) {
    _shoppingList = value;
    notifyListeners();
  }

  void deleteById(ShoppingList) {
    _shoppingList.remove(ShoppingList);
    notifyListeners();
  }
}

```

Jangan lupa untuk mendaftarkan provider anda pada kelas main anda

- **MyProvider.dart**

```dart
import 'package:flutter/material.dart';
import 'package:my_sql_db/Praktek/Provider/myProvider.dart';
import 'package:my_sql_db/Praktek/Screen.dart';
import 'package:provider/provider.dart';

void main() {
  runApp(const MyApp());
}

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => ListProductProvider(),
      child: MaterialApp(
        title: 'Flutter Demo',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: Screen(),
```

```
      ),
    );
  }
}

```

Rancanglah Front-End dari aplikasi seperti berikut ini

- **Screen.dart**

```dart
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:my_sql_db/Praktek/ItemsScreen.dart';
import 'package:my_sql_db/Praktek/model/ShoppingList.dart';
import 'package:my_sql_db/Praktek/ui/shopping_list_dialog.dart';
import 'package:provider/provider.dart';
import 'Provider/myProvider.dart';

class Screen extends StatefulWidget {
  const Screen({Key? key}) : super(key: key);

  @override
  State<Screen> createState() => _ScreenState();
}

class _ScreenState extends State<Screen> {
  int id = 0;
  @override
  Widget build(BuildContext context) {
    var tmp = Provider.of<ListProductProvider>(context, listen: true);
    return Scaffold(
      appBar: AppBar(
        title: const Text("Shopping List"),
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.delete),
            tooltip: 'Delete All',
            onPressed: () {},
          ),
        ],
      ),
      body: ListView.builder(
          itemCount:
              tmp.getShoppingList != null ? tmp.getShoppingList.length : 0,
          itemBuilder: (BuildContext context, int index) {
            return Dismissible(
                key: Key(tmp.getShoppingList[index].id.toString()),
                onDismissed: (direction) {
                  String tmpName = tmp.getShoppingList[index].name;
                  int tmpId = tmp.getShoppingList[index].id;
                  setState(() {
                    tmp.deleteById(tmp.getShoppingList[index]);
                  });
```

```
                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                    content: Text("$tmpName deleted"),
                  ));
                },
                child: ListTile(
                  title: Text(tmp.getShoppingList[index].name),
                  leading: CircleAvatar(
                    child: Text("${tmp.getShoppingList[index].sum}"),
                  ),
                  onTap: () {
                    Navigator.push(context,
                        MaterialPageRoute(builder: (context) {
                      return ItemsScreen(tmp.getShoppingList[index]);
                    }));
                  },
                  trailing: IconButton(
                      icon: Icon(Icons.edit),
                      onPressed: () {
                        showDialog(
                            context: context,
                            builder: (context) {
                              return ShoppingListDialog().buildDialog(
                                  context, tmp.getShoppingList[index], false);
                            });
                      }),
                ));
          }),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          await showDialog(
              context: context,
              builder: (context) {
                return ShoppingListDialog()
                    .buildDialog(context, ShoppingList(++id, "", 0), true);
              });
        },
        child: Icon(Icons.add),
      ),
    );
  }

  @override
  void dispose() {
    super.dispose();
  }
}

```

- **ItemScreen.dart**

```
import 'package:flutter/material.dart';
import 'package:my_sql_db/Praktek/model/ShoppingList.dart';

class ItemsScreen extends StatefulWidget {
  final ShoppingList shoppingList;
  ItemsScreen(this.shoppingList);

  @override
  _ItemsScreenState createState() => _ItemsScreenState(this.shoppingList);
}
```

```
11
12 class _ItemsScreenState extends State<ItemsScreen> {
13   final ShoppingList shoppingList;
14   _ItemsScreenState(this.shoppingList);
15
16   @override
17   Widget build(BuildContext context) {
18     return Scaffold(appBar: AppBar(title: Text(shoppingList.name)));
19   }
20 }
21
```

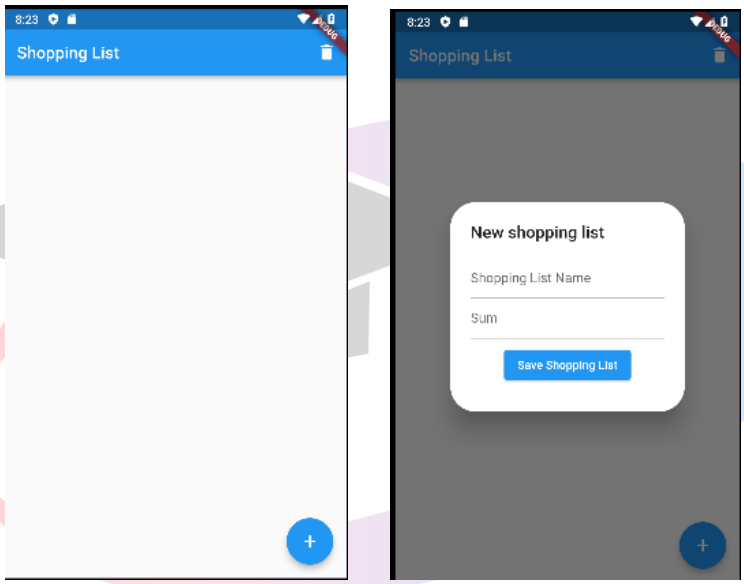
- **shopping_list_dialog.dart**

```
1  import 'package:flutter/material.dart';
2  import 'package:my_sql_db/Praktek/model/ShoppingList.dart';
3
4  class ShoppingListDialog {
5    ShoppingListDialog();
6
7    final txtName = TextEditingController();
8    final txtSum = TextEditingController();
9
10   Widget buildDialog(BuildContext context, ShoppingList list, bool isNew) {
11     if (!isNew) {
12       txtName.text = list.name;
13       txtSum.text = list.sum.toString();
14     } else {
15       txtName.text = "";
16       txtSum.text = "";
17     }
18     return AlertDialog(
19         title: Text((isNew) ? 'New shopping list' : 'Edit shopping list'),
20         shape:
21             RoundedRectangleBorder(borderRadius: BorderRadius.circular(30.0)),
22         content: SingleChildScrollView(
23           child: Column(children: <Widget>[
24             TextField(
25                 controller: txtName,
26                 decoration: InputDecoration(hintText: 'Shopping List Name')),
27             TextField(
28               controller: txtSum,
29               keyboardType: TextInputType.number,
30               decoration: InputDecoration(hintText: 'Sum'),
31             ),
32             Padding(
33               padding: const EdgeInsets.all(8.0),
34               child: ElevatedButton(
35                 child: Text('Save Shopping List'),
36                 onPressed: () {
37                   list.name = txtName.text != "" ? txtName.text : "Empty";
38                   list.sum = txtSum.text != "" ? int.parse(txtSum.text) : 0;
39                   Navigator.pop(context);
40                 },
41               ),
42             ),
```

```
43            ]),
44          ));
45    }
46 }
47
```

shopping_list_dialog.dart merupakan custom widget yang anda dapat bangun untuk memudahkan anda dalam melakukan pengimputan data melalui dialog.

Hasil :



Hasilnya, data belum dapat kita tampilkan pada tampilan front-end. Untuk itu kita akan mencoba menggunakan Sq(F)Lite untuk memproses data sehingga data dapat tersimpan secara permanen dan dapat diakses kembali meskipun aplikasi telah ditutup.

Creating Database

1. Buat sebuah file dart baru bernama dbhelper.dart. File ini akan kita gunakan untuk menyimpan setiap perintah yang berkaitan dengan database. Di dalam file ini buat sebuah class baru bernama DbHelper. Pada proses pertama dalam operasi database Sqlite, anda diwajibkan untuk melakukan inisialisasi dan membuka database. Adapun inisialisasi dan membuka database dapat

menggunakan fungsi **openDatabase()**. Pada saat proses openDatabase() anda dapat mentriger sebuat event **onCreate()** untuk membentuk table yang terhubung dengan database anda. Atau anda dapat mencreate tabel secara terpisah.

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

class DBHelper {
  Database? _database;
  final String _table_name = "shopping_list";
  final String _db_name = "shopinglist_database.db";
  final int _db_version = 1;

  DBHelper() {
    _openDB();
  }

  Future<void> _openDB() async {
    // penghapusan database digunakan, ketika anda sudah membuat database,
    // dan ternyata terjadi perubah pada table
    // await deleteDatabase(
    //     join(await getDatabasesPath(), 'shopinglist_database.db'));
    _database ??= await openDatabase(
      join(await getDatabasesPath(), _db_name),
      onCreate: (db, version) {
        return db.execute(
          'CREATE TABLE $_table_name (id INTEGER PRIMARY KEY, name TEXT, sum
INTEGER)',
        );
      },
      version: _db_version,
    );
  }
}

```

2. Sampai tahap ini, anda belum membuat database. Untuk memerintahkan program anda membentuk database, maka pada screen.dart inisiasikan sebuah variabel helper untuk mengakses class DbHelper. Ketika inisialisasi dilakukan, maka anda telah mengaktifkan sebuah database yang dapat anda gunakan. Pastikan setiap database yang tebuka harus ditutup pada **akhir program**.

```
class _ScreenState extends State<Screen> {
  int id = 0;
  DBHelper _dbHelper = DBHelper();
  @override
  Widget build(BuildContext context) {
```

Insert Data

1. Pada dbhelper.dart, tambahkan sebuah fungsi untuk melakukan penambahan list data ke dalam database.

```
31  Future<void> insertShoppingList(ShoppingList tmp) async {
32    await _database?.insert(
33      'shopping_list',
34      tmp.toMap(),
35      conflictAlgorithm: ConflictAlgorithm.replace,
36    );
37  }
```

2. Pada file shopping_list_dialog.dart, tambahkan perintah untuk menambahkan data ke dalam database pada widget Elevated Button dengan menggunakan helper database. Pastikan anda memberikan akses database pada kelas shopping_list_dialog, melalui constructor sehingga shopping_list_dialog.dart ditulisakan sebagai berikut.

```
1  import 'package:flutter/material.dart';
2  import 'package:my_sql_db/Praktek/DB/dbHelper.dart';
3  import 'package:my_sql_db/Praktek/model/ShoppingList.dart';
4
5  class ShoppingListDialog {
6    DBHelper _dbHelper;
7    ShoppingListDialog(this._dbHelper);
8
9    final txtName = TextEditingController();
10   final txtSum = TextEditingController();
11
12   Widget buildDialog(BuildContext context, ShoppingList list, bool isNew) {
13     if (!isNew) {
14       txtName.text = list.name;
15       txtSum.text = list.sum.toString();
16     } else {
17       txtName.text = "";
18       txtSum.text = "";
19     }
20     return AlertDialog(
21         title: Text((isNew) ? 'New shopping list' : 'Edit shopping list'),
22         shape:
23             RoundedRectangleBorder(borderRadius: BorderRadius.circular(30.0)),
24         content: SingleChildScrollView(
25           child: Column(children: <Widget>[
26             TextField(
27                 controller: txtName,
28                 decoration: InputDecoration(hintText: 'Shopping List Name')),
29             TextField(
```

```
              controller: txtSum,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(hintText: 'Sum'),
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: ElevatedButton(
                child: Text('Save Shopping List'),
                onPressed: () {
                  list.name = txtName.text != "" ? txtName.text : "Empty";
                  list.sum = txtSum.text != "" ? int.parse(txtSum.text) : 0;
                  _dbHelper.insertShoppingList(list);
                  Navigator.pop(context);
                },
              ),
            ),
          ]),
        ));
  }
}
```

3. Sebelum dapat menjalankan perintah insert ini, pastikan anda mengubah pemanggilan fungsi ShoppingListDialog() pada sreen.dart menjadi ShoppingListDialog(_dbHelper).

```
                trailing: IconButton(
                    icon: Icon(Icons.edit),
                    onPressed: () {
                      showDialog(
                          context: context,
                          builder: (context) {
                            return ShoppingListDialog(_dbHelper).buildDialog(
                                context, tmp.getShoppingList[index], false);
                          });
                    }),
              ));
        }),
    floatingActionButton: FloatingActionButton(
      onPressed: () async {
        await showDialog(
            context: context,
            builder: (context) {
              return ShoppingListDialog(_dbHelper)
                  .buildDialog(context, ShoppingList(++id, "", 0), true);
            });
      },
      child: Icon(Icons.add),
    ),
  );
}
```

Select/Read Data

1. Setelah berhasil memasukkan data, anda pasti ingin melihat hasilnya untuk muncul pada screen anda. Oleh karena itu, kita akan menggunakan bantuan provider dan dbhelper, untuk menampilkan data yang telah berhasil di tambahkan. Tambahkanlah sebuah fungsi untuk melakukan pembacaan data dari database.

```
39  Future<List<ShoppingList>> getmyShopingList() async {
40    if (_database != null) {
41      final List<Map<String, dynamic>> maps =
42          await _database!.query('shopping_list');
43      print("Isi DB" + maps.toString());
44      return List.generate(maps.length, (i) {
45        return ShoppingList(maps[i]['id'], maps[i]['name'], maps[i]['sum']);
46      });
47    }
48    return [];
49  }
```

2. Pada screen.dart, anda dapat menghubungkan provider dengan fungsi getMyShopingList() database, sehingga setiap kali terjadi peruban data, provider dapat langsung menampilkan hasilnya pada screen anda.

```
62                  trailing: IconButton(
63                      icon: Icon(Icons.edit),
64                      onPressed: () {
65                        showDialog(
66                            context: context,
67                            builder: (context) {
68                              return ShoppingListDialog(_dbHelper).buildDialog(
69                                  context, tmp.getShoppingList[index], false);
70                            });
71                        _dbHelper
72                            .getmyShopingList()
73                            .then((value) => tmp.setShoppingList = value);
74                      }),
75                ));
76          }),
77      floatingActionButton: FloatingActionButton(
78        onPressed: () async {
79          await showDialog(
80              context: context,
81              builder: (context) {
82                return ShoppingListDialog(_dbHelper)
83                    .buildDialog(context, ShoppingList(++id, "", 0), true);
84              });
```

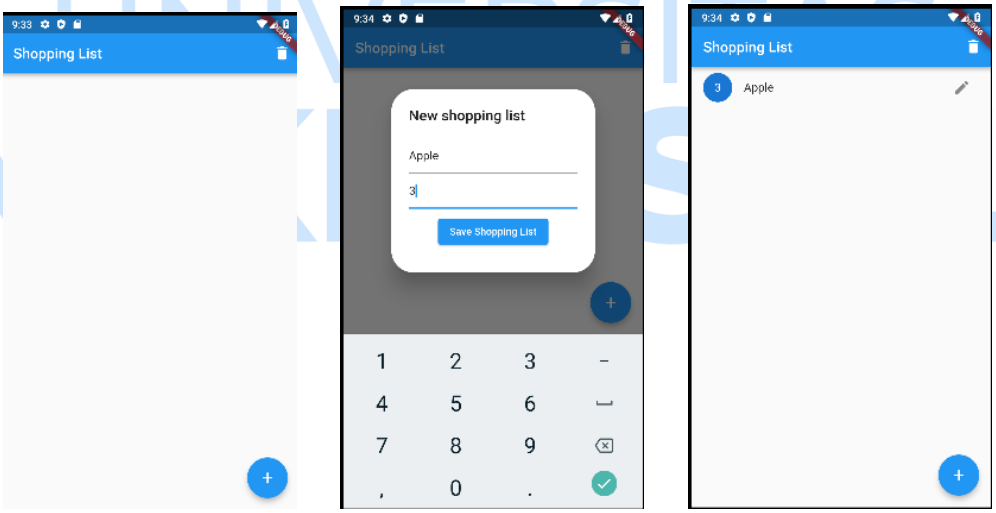
```
85            _dbHelper
86                .getmyShopingList()
87                .then((value) => tmp.setShoppingList = value);
88            },
89            child: Icon(Icons.add),
90          ),
91        );
92      }
```

3. Tambahkan pula pemanggilan fungsi ini pada saat pertama kali widget di build, sehingga data akan langung ditampikan kepada anda, setiap kali apikasi dibuka.

```
18 class _ScreenState extends State<Screen> {
19   int id = 0;
20   DBHelper _dbHelper = DBHelper();
21   @override
22   Widget build(BuildContext context) {
23     var tmp = Provider.of<ListProductProvider>(context, listen: true);
24     _dbHelper.getmyShopingList().then((value) => tmp.setShoppingList = value);
25     return Scaffold(
```

Hasil :

Saat ini, anda telah berhasil menjalankan perintah untuk melakukan operasi di dalam database baca dan tulis. Anda dapat melihat data anda akan tersimpan secara permanen di dalam aplikasi anda, walaupn anda telah menutup aplikasi dan membukanya kembali.



Delete Data

1. Pada dbhelper.dart, tambahkan sebuah fungsi untuk melakukan penghapusan data dari database.

```
51  Future<void> deleteShoppingList(int id) async {
52    await _database?.delete(
53      'shopping_list',
54      where: 'id = ?',
55      whereArgs: [id],
56    );
57  }
```

2. Data akan dihapus ketika disapu ke kanan atau kiri dari layar aplikasi. Pada screen.dart di bagian onDissmissed tambahkan perintah berikut untuk menghapus data dari database ketika aksi ini dilakukan.

```
44                onDismissed: (direction) {
45                  String tmpName = tmp.getShoppingList[index].name;
46                  int tmpId = tmp.getShoppingList[index].id;
47                  setState(() {
48                    tmp.deleteById(tmp.getShoppingList[index]);
49                  });
50                  _dbHelper.deleteShoppingList(tmpId);
51                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
52                    content: Text("$tmpName deleted"),
53                  ));
54                },
```

Close Database

3. Pada dbhelper.dart, tambahkan sebuah fungsi untuk menutup koneksi ke dalam database.

```
59  Future<void> closeDB() async {
60    await _database?.close();
61  }
```

4. Pastikan anda menambahkan proses untuk menutup database Ketika aplikasi selesi dilakukan dengan cara menambahkan fungsi closeDB() pada lifecycle dispose() aplikasi anda.

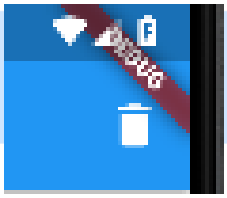
```
99  @override
100 void dispose() {
101   _dbHelper.closeDB();
102   super.dispose();
103 }
```

Latihan

1. Pada contoh diatas, masih terdapat kelemahan, dimana setiap aplikasi dibuka kembali, pengimputan data akan dimulai Kembali dari urutan pertama. Ini disebabkan oleh id tidak tersimpan secara permanen. Lakukan penyimpanan sharedPreference atau pembacaan id terakhir untuk menagani masalah tersebut

```
18  class _ScreenState extends State<Screen> {
19    int id = 0;
20    DBHelper _dbHelper = DBHelper();
21    @override
22    Widget build(BuildContext context) {
23      var tmp = Provider.of<ListProductProvider>(context,
24      _dbHelper.getmyShopingList().then((value) => tmp.se
25      return Scaffold(
26        appBar: AppBar(
27          title: const Text("Shopping List"),
```

2. Tambahkan proses untuk menghapus seluruh isi table database, ketika action pada appBar di klik.



3. Tambahkan sebuah aktifitas berupa history, dimana setiapkali penghapusan di lakukan, akan tersimpan dalam aktifitas history, termasuk dengan tanggal dan jam dilakukannya penghapusan data.

UNIVERSITAS
MIKROSKIL