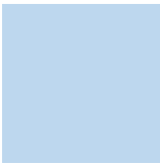


MODUL

PEMROGRAMAN BERBASIS PLATFORM



PERTEMUAN – 6

DATA 1 – SQFLITE

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA





Daftar Gambar

3	Gambar 1. Flutter dan SQLite.....	3
4	Gambar 2. Dependency pada pubspec.yaml.....	5
5	Gambar 3. Struktur file entity	6
6	Gambar 4. Class Employee Entity.....	7
7	Gambar 5. Class Employee Entity.....	7
8	Gambar 6. Import library sqflite pada sql_helper	8
9	Gambar 7. Pembuatan class SQLHelper pada file sql_helper.....	8
10	Gambar 8. Fungsi untuk membuat table employee dengan column id, name, dan email menggunakan syntax SQL.....	8
11	Gambar 9. Fungsi untuk memanggil table, memasukan data, dan membaca data object pada table yang sudah dibuat sebelumnya.....	9
12	Gambar 10. Fungsi untuk update, dan menghapus data pada table yang sudah ada	9
13	Gambar 11. Halaman inputan data.....	10
14	Gambar 12. Struktur file pada folder lib.....	10
15	Gambar 13. Input Page bagian 1.....	11
16	Gambar 14. Input Page bagian 2.....	12
17	Gambar 15. Input Page bagian 3, pemanggilan fungsi dari sql_helper yang akan digunakan dalam halaman ini.....	12
18	Gambar 16. Halaman Main(Kosong)	13
19	Gambar 17. Halaman Main(dengan data yang ditampilkan menggunakan list)	13
20	Gambar 18. Implementasi library slider agar tampilan lebih menarik(slide ke kiri pada data pada list)	14
21	Gambar 19. Main.dart bagian 1	15
22	Gambar 20. Main.dart bagian 2	16
23	Gambar 21. Main.dart bagian 3.....	16
24	Gambar 22. Main.dart bagian 4	17



TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Memahami apa itu SQFLITE.
2. Memahami cara kerja SQFLITE.
3. Mengimplementasikan CRUD dengan database SQLite pada aplikasi flutter menggunakan SQFLITE.

DASAR TEORI

A. SQFLITE

SQFLITE merupakan salah satu plugin atau library dari pub.dev yang dapat digunakan dalam flutter untuk mengimplementasikan database SQLite apabila aplikasi yang dibuat memerlukan “Data Persistence” atau data dalam jumlah besar di perangkat lokal dengan database, bukan local file atau dalam bentuk key-value. Dengan demikian, SQFLite memungkinkan aplikasi menyimpan data secara lokal dan melakukan CRUDS yang biasa ditemukan pada aplikasi-aplikasi umum lainnya dengan performa yang lebih baik daripada metode persistence lainnya.



Gambar 1. Flutter dan SQLite



***Note :**

1. Memahami penggunaan “async, await, then, future” dalam dart.

2. Mengingat kembali operasi CRUDS dan syntax SQL

referensi: <https://jelenaaa.medium.com/when-to-use-async-await-then-and-future-in-dart-5e00e64ab9b1#:~:text=Async%20means%20that%20this%20function,value%20from%20your%20asynchronous%20function>

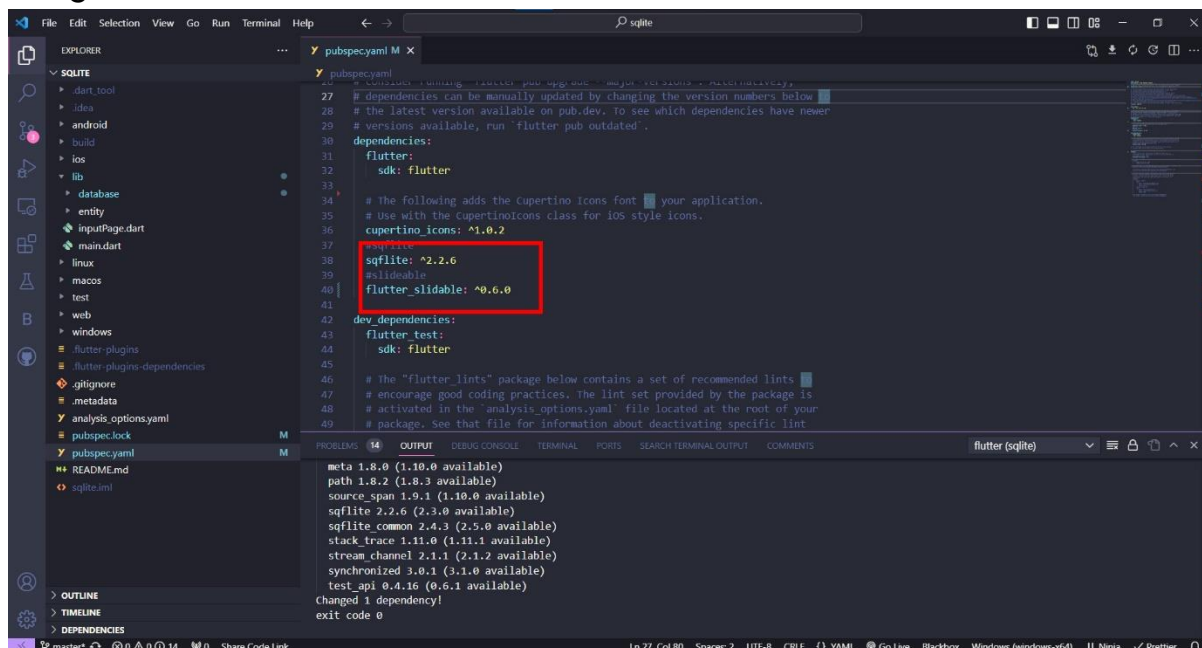
GUIDED – CRUD

Poin yang dipelajari dalam Guided 1 ini, yaitu :

1. Memahami cara membuat fungsi CRUD dengan SQLite.
2. Memahami struktur folder dan file dalam penggunaan SQLite.

Pada guided ini, kita akan mencoba memahami penggunaan SQLite sebagai database dengan membuat aplikasi daftar *Employee* sederhana menggunakan flutter. Silahkan ikuti langkah – langkah berikut ini :

1. Buat project flutter baru gd5_x_yyyy, x merupakan nama kelas, dan yyyy merupakan 4 digit npm terakhir praktikan. Buka dengan IDE (VSCode pada modul ini).
2. Tambahkan library sqflite dan flutter_slidable pada file pubspec.yaml di project. Library sqflite digunakan untuk mengimplementasikan database SQLite dan slidable untuk library UI. (gambar 2)

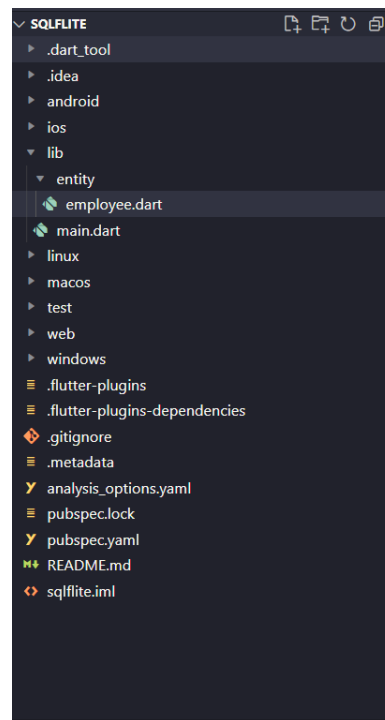


Gambar 2. Dependency pada pubspec.yaml

3. Kemudian buat folder entity yang akan digunakan untuk menyimpan object pada aplikasi agar lebih rapi. Pada project kali ini, kita akan membuat object employee sederhana.



*Pembiasaan pengelolaan file yang baik akan terasa sangat membantu jika aplikasi yang dibuat merupakan aplikasi skala besar dan terus bertumbuh.

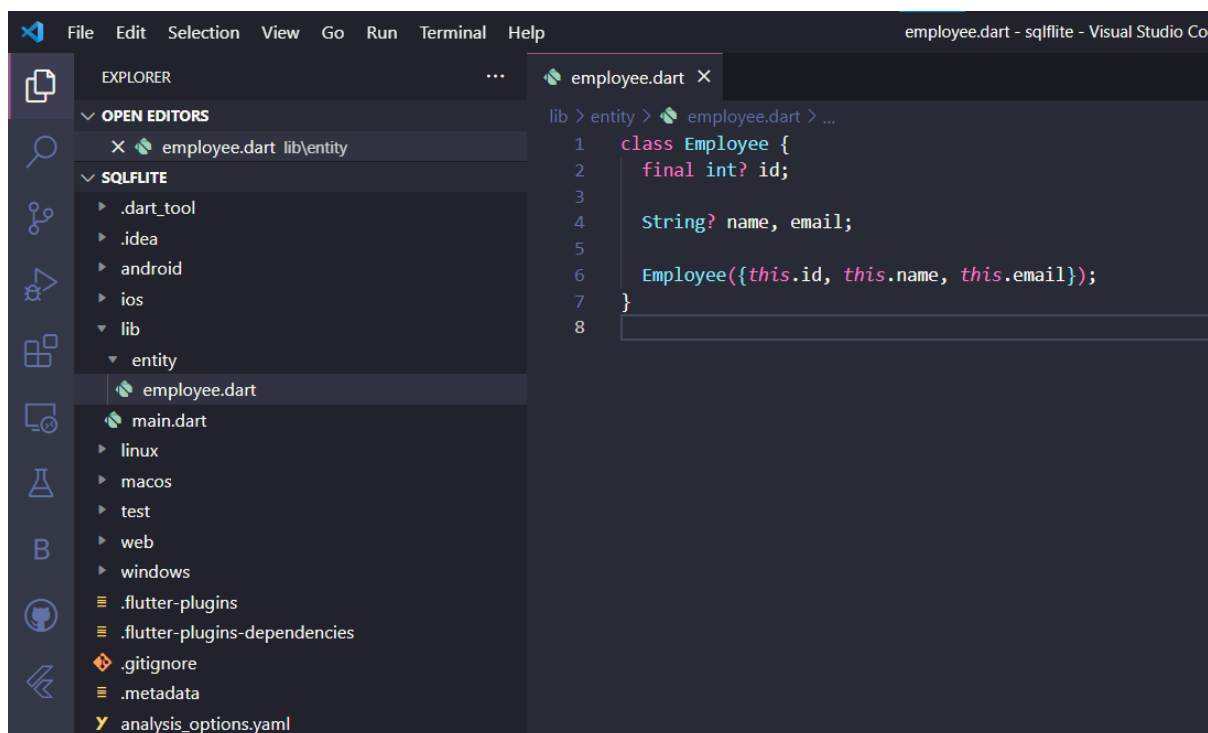


Gambar 3. Struktur file entity

4. Object pada folder entity merupakan class object yang berisi variabel-variabel yang terdapat pada object. Sebagai latihan, isikan class Employee dengan variabel **id** dengan tipe integer, **name** dengan tipe string, dan **email** dengan tipe string.

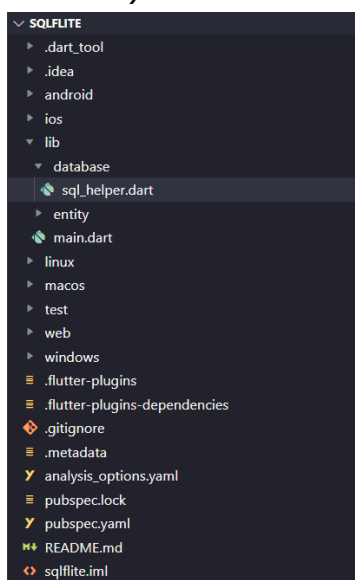
Selain itu, buat class constructor untuk object Employee agar nantinya class constructor dapat dipanggil untuk membuat object Employee. (gambar 4)

*Perhatikan nama variabel, nama kelas, dan nama class constructor.



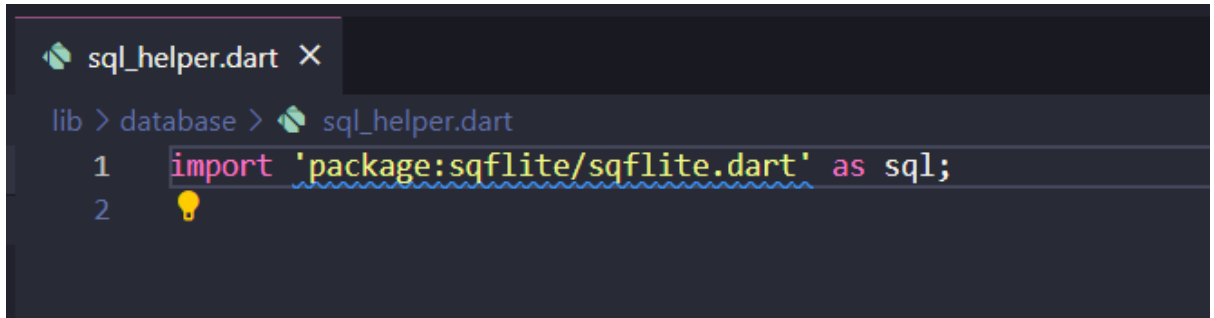
Gambar 4. Class Employee Entity

5. Buat folder database yang nantinya akan digunakan menampung file dart yang berisi syntax-syntax sql dan untuk mempermudah pengimplementasikan library sqflite sebelumnya untuk entity yang sudah dibuat sebelumnya pada aplikasi. Beri nama file tersebut dengan sql_helper. (gambar 5)



Gambar 5. Class Employee Entity

6. Import library sqflite pada file sql_helper dan aliasing package sqflite sebagai sql. (gambar 6)



```

sql_helper.dart X
lib > database > sql_helper.dart
1 import 'package:sqflite/sqflite.dart' as sql;
2

```

Gambar 6. Import library sqflite pada sql_helper

7. Buat class SQLHELPER yang nantinya akan berisi implementasi package sqflite yang sudah diimport sebelumnya untuk entity Employee.



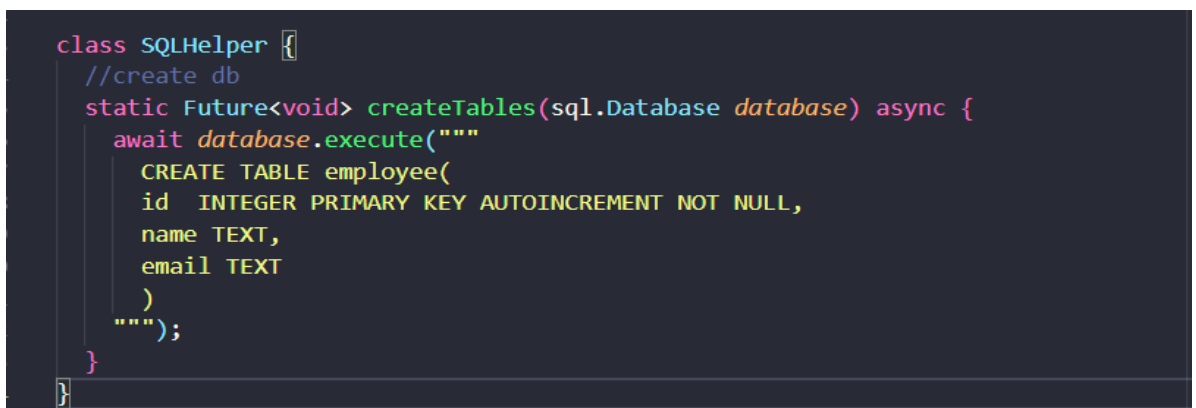
```

sql_helper.dart
lib > database > sql_helper.dart > SQLHelper
1 import 'package:sqflite/sqflite.dart' as sql;
2
3 class SQLHelper {
4
5 }
6

```

Gambar 7. Pembuatan class SQLHelper pada file sql_helper

8. Buat fungsi static future sesuai fungsi yang dibutuhkan terhadap object yang telah dibuat sebelumnya.
*Perhatikan nama setiap fungsi dan parameter yang diperlukan karena fungsi-fungsi pada sql_helper yang akan digunakan dalam CRUD.



```

class SQLHelper {
  //create db
  static Future<void> createTables(sql.Database database) async {
    await database.execute("""
      CREATE TABLE employee(
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
        name TEXT,
        email TEXT
      )
    """);
  }
}

```

Gambar 8. Fungsi untuk membuat table employee dengan column id, name, dan email menggunakan syntax SQL


```
//call db
static Future<sql.Database> db() async {
  return sql.openDatabase('employee.db', version: 1,
    onCreate: (sql.Database database, int version) async {
      await createTables(database);
    });
}

//insert Employee
static Future<int> addEmployee(String name, String email) async {
  final db = await SQLHelper.db();
  final data = {'name': name, 'email': email};
  return await db.insert('employee', data);
}

//read Employee
static Future<List<Map<String, dynamic>>> getEmployee() async {
  final db = await SQLHelper.db();
  return db.query('employee');
}
```

145

146 *Gambar 9. Fungsi untuk memanggil table, memasukan data, dan membaca data object*
147 *pada table yang sudah dibuat sebelumnya*

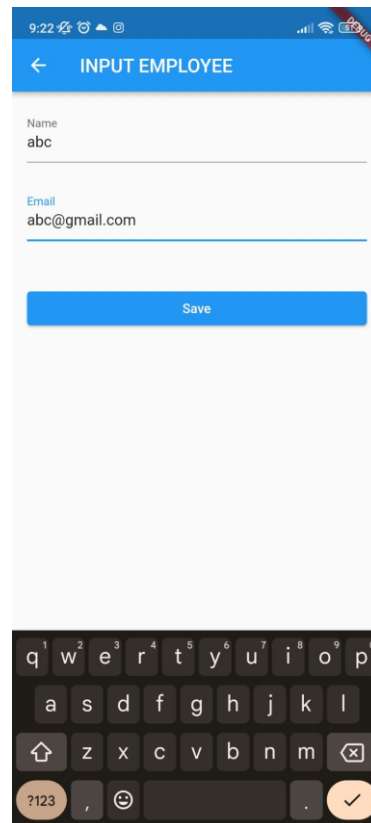
```
//update Employee
static Future<int> editEmployee(int id, String name, String email) async {
  final db = await SQLHelper.db();
  final data = {'name': name, 'email': email};
  return await db.update('employee', data, where: "id = $id");
}

//delete Employee
static Future<int> deleteEmployee(int id) async {
  final db = await SQLHelper.db();
  return await db.delete('employee', where: "id = $id");
}
```

148

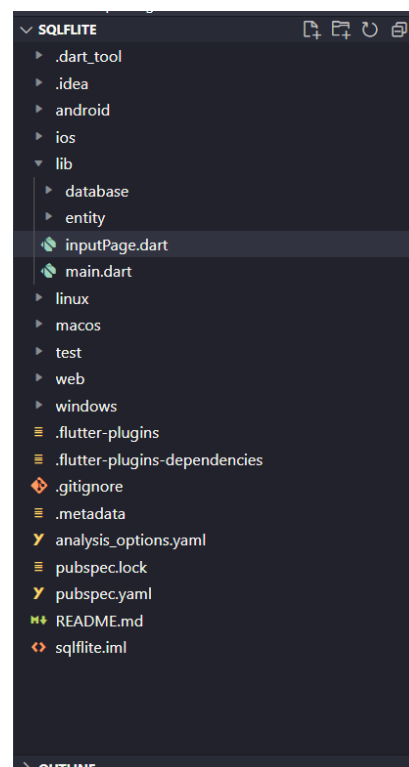
149 *Gambar 10. Fungsi untuk update, dan menghapus data pada table yang sudah ada*

150 9. Selanjutnya buat halaman untuk menginputkan data seperti berikut :



Gambar 11. Halaman inputan data

10. Pertama, buat file inputPage.dart sebagai halaman untuk melakukan fungsi insert dan update data.



Gambar 12. Struktur file pada folder lib

11. Import package material dari library flutter, sqflite yang sudah diimplementasikan pada file sql_helper sebelumnya, dan entity employee. Halaman ini membutuhkan variable-variable object employee.

```

inputPage.dart X
lib > inputPage.dart > ...
1  import 'package:flutter/material.dart';
2  import 'package:sqflite/database/sql_helper.dart';
3  import 'package:sqflite/entity/employee.dart';
4
5  class InputPage extends StatefulWidget {
6    const InputPage(
7      {super.key,
8        required this.title,
9        required this.id,
10       required this.name,
11       required this.email});
12
13    final String? title, name, email;
14    final int? id;
15
16    @override
17    State<InputPage> createState() => _InputPageState();
18  }
19
20  class _InputPageState extends State<InputPage> {
21    TextEditingController controllerName = TextEditingController();
22    TextEditingController controllerEmail = TextEditingController();
23
24    @override
25    Widget build(BuildContext context) {
26      if (widget.id != null) {
27        controllerName.text = widget.name!;
28        controllerEmail.text = widget.email!;
29      }
30      return Scaffold(
31        appBar: AppBar(
32          title: Text("INPUT EMPLOYEE"),
33        ), // AppBar
34        body: ListView(
35          padding: EdgeInsets.all(16),
36          children: <Widget>[
37            TextField(
38              controller: controllerName

```

Controller ini akan digunakan sebagai penampung inputan dari user

Jika id yang diterima halaman tidak null, maka inputan akan diisi nama dan email yang diterima juga oleh halaman

Gambar 13. Input Page bagian 1

```

inputPage.dart X
lib > inputPage.dart > ...
31 appBar: AppBar(
32   title: Text("INPUT EMPLOYEE"),
33 ), // AppBar
34 body: ListView(
35   padding: EdgeInsets.all(16),
36   children: <Widget>[
37     TextField(
38       controller: controllerName,
39       decoration: const InputDecoration(
40         border: UnderlineInputBorder(),
41         labelText: 'Name',
42       ), // InputDecoration
43     ), // TextField
44     SizedBox(height: 24),
45     TextField(
46       controller: controllerEmail,
47       decoration: const InputDecoration(
48         border: UnderlineInputBorder(),
49         labelText: 'Email',
50       ), // InputDecoration
51     ), // TextField
52     SizedBox(height: 48),
53     ElevatedButton(
54       child: Text('Save'),
55       onPressed: () async {
56         if (widget.id == null) {
57           await addEmployee();
58         } else {
59           await editEmployee(widget.id!);
60         }
61         Navigator.pop(context);
62       },
63     ), // ElevatedButton
64   ], // <Widget>[]
65 ); // ListView // Scaffold
66 }

```

Dengan button yang sama, kita dapat melakukan 2 fungsi sekaligus dimana aplikasi akan melakukan insert data jika id yang diterima inputPage adalah null dan akan melakukan update jika id tidak null

163

164

Gambar 14. Input Page bagian 2

```

67
68 Future<void> addEmployee() async {
69   await SQLHelper.addEmployee(controllerName.text, controllerEmail.text);
70 }
71
72 Future<void> editEmployee(int id) async {
73   await SQLHelper.editEmployee(id, controllerName.text, controllerEmail.text);
74 }
75 }
76

```

165

166

167

Gambar 15. Input Page bagian 3, pemanggilan fungsi dari sql_helper yang akan digunakan dalam halaman ini

168

169

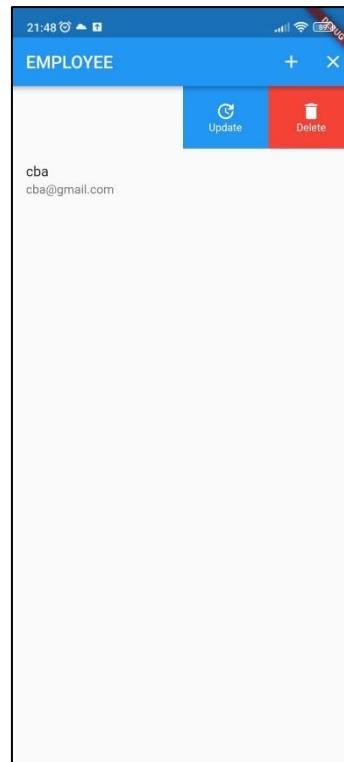
12. Selanjutnya buat halaman utama untuk menampilkan data employee menggunakan list sebagai berikut :



Gambar 16. Halaman Main(Kosong)



Gambar 17. Halaman Main(dengan data yang ditampilkan menggunakan list)



Gambar 18. Implementasi library slider agar tampilan lebih menarik (slide ke kiri pada data pada list)

13. Import beberapa package pada main.dart yang nantinya akan menjadi halaman default aplikasi.



```
main.dart X
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2 import 'package:flutter_slidable/flutter_slidable.dart';
3 import 'package:sqlite/database/sql_helper.dart';
4 import 'package:sqlite/entity/employee.dart';
5 import 'package:sqlite/inputPage.dart';
6
Run | Debug | Profile
7 void main() {
8   runApp(const MyApp());
9 }
10
11 class MyApp extends StatelessWidget {
12   const MyApp({super.key});
13
14   // This widget is the root of your application.
15   @override
16   Widget build(BuildContext context) {
17     return MaterialApp(
18       title: 'SQFLITE',
19       theme: ThemeData(
20         primarySwatch: Colors.blue,
21       ), // ThemeData
22       home: const HomePage(
23         title: 'SQFLITE',
24       ), // HomePage
25     ); // MaterialApp
26   }
27 }
```

Gambar 19. Main.dart bagian 1

```

main.dart X
lib > main.dart > _HomePageState > build
29 class HomePage extends StatefulWidget {
30   const HomePage({super.key, required this.title});
31
32   final String title;
33
34   @override
35   State<HomePage> createState() => _HomePageState();
36 }
37
38 class _HomePageState extends State<HomePage> {
39   //getData
40   List<Map<String, dynamic>> employee = [];
41   void refresh() async {
42     final data = await SQLHelper.getEmployee();
43     setState(() {
44       employee = data;
45     });
46   }
47
48   @override
49   void initState() {
50     refresh();
51     super.initState();
52   }

```

Pembuatan fungsi refresh untuk merefresh halaman agar data yang ditampilkan merupakan data paling baru. Data yang didapatkan dari fungsi getEmployee pada sql_helper ditampilkan pada variable employee

181

182

Gambar 20. Main.dart bagian 2

```

main.dart X
lib > main.dart > _HomePageState > build
54
55 @override
56 Widget build(BuildContext context) {
57   return Scaffold(
58     appBar: AppBar(
59       title: Text("EMPLOYEE"),
60       actions: [
61         IconButton(
62           icon: Icon(Icons.add),
63           onPressed: () async {
64             Navigator.push(
65               context,
66               MaterialPageRoute(
67                 builder: (context) => const InputPage(
68                   title: 'INPUT EMPLOYEE',
69                   id: null,
70                   name: null,
71                   email: null)), // InputPage // MaterialPageRoute
72             ).then((_) => refresh());
73           }, // IconButton
74         ),
75       ], // AppBar
76     body: ListView.builder(
77       itemCount: employee.length,
78       itemBuilder: (context, index) {
79         return Slidable(
80           child: ListTile(
81             title: Text(employee[index]['name']),
82             subtitle: Text(employee[index]['email']),
83           ), // ListTile
84           actionPane: SlidableDrawerActionPane(),
85           secondaryActions: [
86             IconSlideAction(
87               caption: 'Update',
88               color: Colors.blue,
89               icon: Icons.update,
90               onTap: () async {

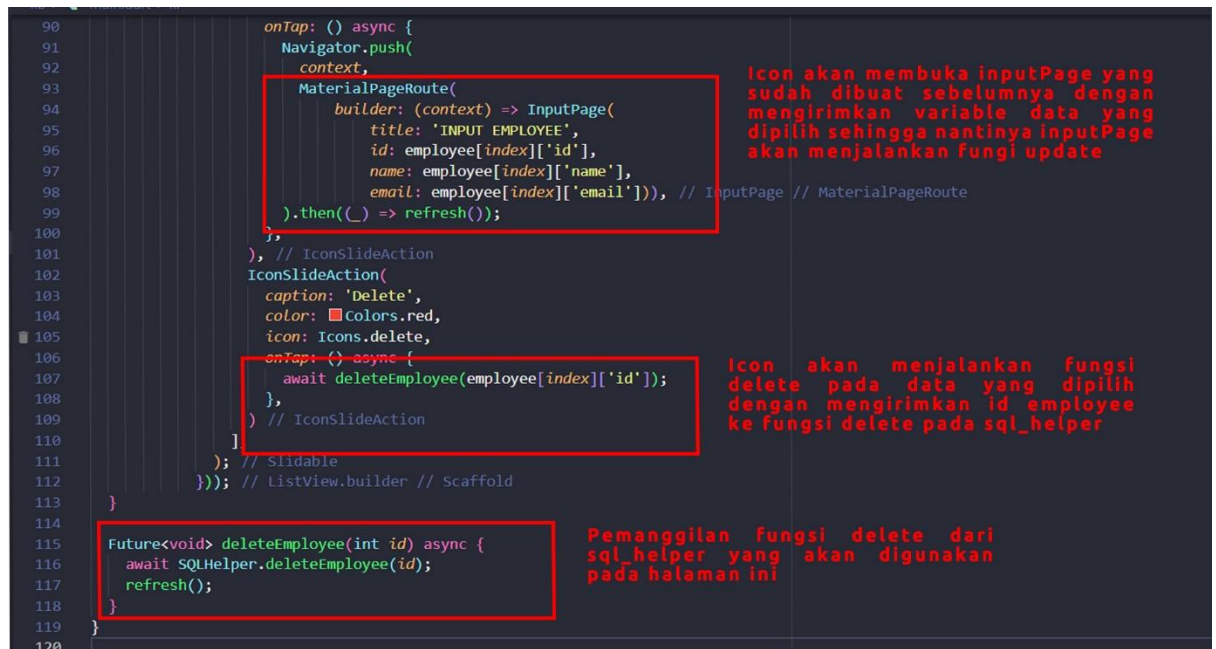
```

Icon akan membuka Input-Page yang sudah dibuat sebelumnya dengan mengirimkan variable null sehingga nantinya InputPage akan menjalankan fungsi insert

183

184

Gambar 21. Main.dart bagian 3



Gambar 22. Main.dart bagian 4



204 **ATURAN Pengerjaan Guided :**

- 205 - **Guided dikerjakan selama waktu perkuliahan berlangsung.**
- 206 - **Penamaan projek guided harus sesuai dengan yang sudah**
- 207 **dicontohkan.**
- 208 - **Guided wajib diupload ke github dengan status private dengan**
- 209 **penamaan: `gd5_x_yyyy`**
- 210 ○ **x → kelas**
- 211 ○ **yyyy → 4 DIGIT TERAKHIR NPM**
- 212 - **Setelah diupload melalui github, jangan lupa untuk mengundang**
- 213 **asisten ke github.**