PRAKTIKUM MCS BAB 3

Sqflite

PENDAHULUAN

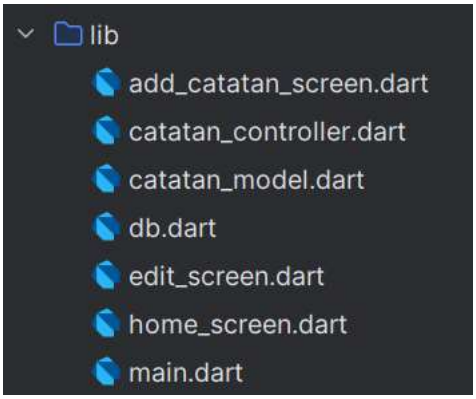
Pada praktikum MCS bab 3 akan belajar cara membangun database lokal dan bentuk penggunaannya dalam bentuk aplikasi android yang dibangun dengan framework flutter. Sqflite adalah package yang digunakan untuk membangun database lokal pada aplikasi yang dibangun menggunakan flutter. Aplikasi yang akan dibangun berbentuk mirip seperti aplikasi note pada umumnya namun memiliki tema yang berbeda. Isi note tersebut akan disimpan ke dalam database. Definisi database sesuai namanya yaitu basis data dimana data dikumpulkan dan dapat terelasi dengan sistematis yang membutuhkan memori.

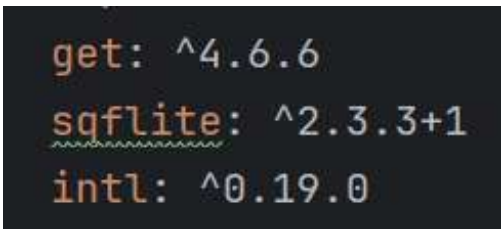
PRAKTIKUM BAB 1

Tampilan aplikasi yang akan dibangun

Penjelasan cara kerja aplikasi akan diterangkan oleh Penanggung Jawab (PJ)

Buat beberapa file .dart di dalam folder lib dan tambahkan beberapa package berikut e pubspec.yaml





Buka file catatan_model.dart dan isi dengan code berikut :

```
class CatatanModel{
  int? id;
  String? tanggal;
  int? warna;
  String? tugas1;
  String? tugas2;
```

```
  String? tugas3;

  CatatanModel({
    this.id,
    required this.tanggal,
    required this.warna,
    required this.tugas1,
    required this.tugas2,
    required this.tugas3,
  });

  //toJson
  Map<String, dynamic> toJson(){
    final Map<String, dynamic> data = <String, dynamic>{};
    data['id'] = id;
    data['tanggal'] = tanggal;
    data['warna'] = warna;
    data['tugas1'] = tugas1;
    data['tugas2'] = tugas2;
    data['tugas3'] = tugas3;
    return data;
  }

  CatatanModel.fromJson(Map<String,dynamic> json){
    id = json["id"];
    tanggal = json["tanggal"];
    warna = json["warna"];
    tugas1 = json["tugas1"];
    tugas2 = json["tugas2"];
    tugas3 = json["tugas3"];
  }
}
```

fromJson dan toJson digunakan saat berkerja dengan Java Script Object Notation (JSON) untuk melakukan encode dan decode. Encode adalah ketika kita ingin mengirim atau konversi object ke JSON sedangkan decode adalah penarikan atau konversi dari JSON ke object. Nantinya di dalam aplikasi saat digunakan oleh user akan memasukkan data ke database melalui proses toJson atau encode terlebih dahulu dan saat membaca data yang ada di dalam database akan melewati proses decode.

Buka file db.dart dan isi dengan code berikut :

```
import 'package:mcs_bab_3/catatan_model.dart';
import 'package:sqflite/sqflite.dart';

class DB{
  static Database? catatanDb;
  static const String catatanDbTable = "catatan";

  static Future<void> initCatatanDb() async{
    if(catatanDb != null){
      return;
    }
```

```
    try{
      String path = "${await getDatabasesPath()}/catatan.db";
      catatanDb = await openDatabase(
        path, version: 1, onCreate: (db, version){
          const sql = """CREATE TABLE $catatanDbTable(
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          tanggal STRING,
          warna INTEGER,
          tugas1 STRING,
          tugas2 STRING,
          tugas3 STRING)""";
          return db.execute(sql);
      },
      );
    } catch(e){
      print("error : $e");
    }
  }

  //insert
  static Future<int> insert(CatatanModel catatanModel) async{
    return await catatanDb?.insert(catatanDbTable, catatanModel.toJson())
?? 1;
  }

  //retrieve
  static Future<List<Map<String, dynamic>>> query() async{
    return await catatanDb!.query(catatanDbTable,);
  }

  //delete
  static delete(CatatanModel catatanModel)async {
    return catatanDb!.delete(catatanDbTable, where: "id = ?", whereArgs:
[catatanModel.id]);
  }

  //update warna jadi 0
  static updateWarna0(int id) async{
    return await catatanDb!.rawUpdate(
      """UPDATE $catatanDbTable SET warna = 0 WHERE id = ?""", [id]
    );
  }

  //update warna jadi 1
  static updateWarna1(int id) async{
    return await catatanDb!.rawUpdate(
        """UPDATE $catatanDbTable SET warna = 1 WHERE id = ?""", [id]
    );
  }

  //update warna jadi 2
  static updateWarna2(int id) async{
    return await catatanDb!.rawUpdate(
        """UPDATE $catatanDbTable SET warna = 2 WHERE id = ?""", [id]
    );
  }

  //update warna jadi 3
  static updateWarna3(int id) async{
    return await catatanDb!.rawUpdate(
        """UPDATE $catatanDbTable SET warna = 3 WHERE id = ?""", [id]
```

```
    );
  }
}
```

fungsi initCatatanDb digunakan untuk inisialisasi database. Di dalam fungsi initCatatanDb() akan dilakukan pengecekan. Jika database belum excist maka database akan dibuat, bisa dilihat di dalam blok try di atas. Jika database sudah excist maka database tidak akan dibuat dan keluar begitu saja, bisa dilihat di dalam blok if. Di dalam database kita juga melakukan pembuatan table beserta atributnya dan juga untuk pengisian, perubahan, penghapusan data di dalamnya. Pembuatan table dan beserta atributnya biasa disebut Data Definition Language (DDL) dan untuk pengisian, perubahan, penghapusan data di dalamnya biasa disebut Data Manipulate Language (DML).

Buka file catatan_controller.dart dan isi dengan code berikut :

```
import 'package:get/get.dart';
import 'package:mcs_bab_3/catatan_model.dart';

import 'db.dart';

class CatatanController extends GetxController{
  var catatanList = [].obs;

  //get all data from the table
  void getCatatanData() async{
    List<Map<String, dynamic>> catatanData = await DB.query();
    catatanList.assignAll(catatanData.map((e) =>
CatatanModel.fromJson(e)).toList());
  }

  //insert atau tambah data
  Future<int> insert({required CatatanModel catatanModel}) async{
    return await DB.insert(catatanModel);
  }

  //delete
  void delete({required CatatanModel catatanModel}) async{
    await DB.delete(catatanModel);
  }

  //update warna jadi 0
  void updateWarna0({required int id}) async{
    await DB.updateWarna0(id);
  }

  //update warna jadi 1
  void updateWarna1({required int id}) async{
    await DB.updateWarna1(id);
  }

  //update warna jadi 2
```

```
  void updateWarna2({required int id}) async{
    await DB.updateWarna2(id);
  }

  //update warna jadi 3
  void updateWarna3({required int id}) async{
    await DB.updateWarna3(id);
  }
}
```

Kita menggunakan GetX agar dapat memudahkan komunikasi dengan database. Kita membuat variabel catatanList yang isinya [].obs. variabel catatanList akan dijadikan sebagai variabel observasi dimana data ditampung di dalam list lalu diobservasi. Memungkinkan GetX memantau perubahan pada variabel secara otomatis. Fungsi getCatatanData() digunakan untuk mendapatkan data dari database (retrieve). Dan method lainnya untuk memasukkan data (insert) mengubah data di dalam database (update) dan menghapus data di dalam database (delete).

Buka file main.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:mcs_bab_3/db.dart';
import 'package:mcs_bab_3/home_screen.dart';

void main() async{
  WidgetsFlutterBinding.ensureInitialized();
  await DB.initCatatanDb();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Catatan App',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.dark(),
        useMaterial3: true,
      ),
      home: const HomeScreen(),
    );
  }
}
```

fungsi main() kita buat agar menjadi asynchronous lalu di dlaamnya kita buat agar aplikasi mempersiapkan widget yang dibutuhkan dan mengisialisasi database. Di dalam class MyApp biasanya kita menggunakan MaterialApp() namun kali ini kita menggunakan GetMaterialApp() karena kita menggunakan state management GetX yang mana di dalam aplikasi ini berfungsi untuk dapat membantu berkomunikasi dengan database.

Buka file home_screen.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:intl/intl.dart';
import 'package:mcs_bab_3/add_catatan_screen.dart';
import 'package:mcs_bab_3/catatan_controller.dart';
import 'package:mcs_bab_3/catatan_model.dart';
import 'package:mcs_bab_3/db.dart';
import 'package:mcs_bab_3/edit_screen.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final CatatanController catatanController = Get.put(CatatanController());

  Future<void> ubahTanggal(BuildContext context, CatatanModel catatanModel)
async{
    DateTime? setDate = await showDatePicker(
      context: context,
      initialDate: DateTime.now(),
      firstDate: DateTime(2024),
      lastDate: DateTime(2025),
    );

    if(setDate != null && setDate != DateTime.now()){
      setState(() {
        DB.catatanDb!.rawUpdate(
          """UPDATE ${DB.catatanDbTable} SET tanggal =
"${DateFormat.yMd().format(setDate)}" WHERE id = ?""",
          [catatanModel.id],
        );
        catatanController.getCatatanData();
      });
    }
  }

  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    catatanController.getCatatanData();
  }
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text("Catatan App"),
      actions: [
        GestureDetector(
          child: const Icon(Icons.add),
          onTap: (){
            Get.to(const AddCatatanScreen());
          },
        ),
        const SizedBox(width: 10,),
      ],
    ),
    body: SingleChildScrollView(
      child: Column(
        children: [
          const SizedBox(height: 20,),

          Container(
            margin: const EdgeInsets.symmetric(horizontal: 20),
            padding: const EdgeInsets.all(14),
            width: double.infinity,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(14),
              border: Border.all(color: Colors.white),
              boxShadow: const [
                BoxShadow(
                  color: Colors.white,
                  offset: Offset(0.0, 0.1),
                  blurRadius: 10.0,
                ),
              ],
              color: const Color(0xff2E3840),
            ),
            child: const Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  'Create your notes in this App',
                  style: TextStyle(fontSize: 16),
                ),
                Text(
                  '- Write your plan',
                  style: TextStyle(fontWeight: FontWeight.w600),
                ),
                Text(
                  '- Write your needs',
                  style: TextStyle(fontWeight: FontWeight.w600),
                ),
                Text(
                  '- Write your memories',
                  style: TextStyle(fontWeight: FontWeight.w600),
                )
              ],
            ),
          ),

          const SizedBox(height: 30,),
```

```
            Container(
              width: double.infinity,
              child: Obx(() => ListView.builder(
                itemCount: catatanController.catatanList.length,
                shrinkWrap: true,
                itemBuilder: (_, index) {
                  CatatanModel catatanModel =
catatanController.catatanList[index];
                  return Container(
                    margin: const EdgeInsets.symmetric(vertical: 16,
horizontal: 18,),
                    padding: const EdgeInsets.all(12),
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(14),
                      color: catatanModel.warna == 0?
                      const Color(0xff4C4C6D) : catatanModel.warna == 1?
                      const Color(0xff6F61C0) : catatanModel.warna == 2?
                      Colors.blue[900] : Colors.pink[500],
                      border: Border.all(width: 3, color: Colors.white,)
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Row(
                          mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                          children: [
                            Row(
                              children: [
                                Text("${catatanModel.tanggal}", style:
const TextStyle(fontSize: 16, fontWeight: FontWeight.w600),),
                                IconButton(
                                  onPressed: (){
                                    setState(() {
                                      ubahTanggal(context,
catatanController.catatanList[index]);
                                    });
                                  },
                                  icon: const Icon(Icons.edit),
                                ),
                                IconButton(
                                  icon: const Icon(Icons.brush),
                                  onPressed: (){
                                    setState(() {
                                      if(catatanModel.warna == 0){
                                        catatanController.updateWarna1(id:
catatanModel.id!);
                                      } else if(catatanModel.warna == 1){
                                        catatanController.updateWarna2(id:
catatanModel.id!);
                                      } else if(catatanModel.warna == 2){
                                        catatanController.updateWarna3(id:
catatanModel.id!);
                                      } else{
                                        catatanController.updateWarna0(id:
catatanModel.id!);
                                      }
                                      catatanController.getCatatanData();
                                    });
                                  },
```

```
                                ),
                              ],
                            ),

                            Row(
                              children: [
                                IconButton(
                                  onPressed: (){
                                    setState(() {

catatanController.delete(catatanModel: catatanModel);
                                      catatanController.getCatatanData();
                                    });
                                  },
                                  icon: const Icon(Icons.delete),
                                ),
                                IconButton(
                                  onPressed: (){
                                    setState(() {
                                      Get.to(EditScreen(catatanModel:
catatanModel));

                                    });
                                  },
                                  icon: const Icon(Icons.edit),)
                              ],
                            )
                          ],
                        ),

                        const SizedBox(height: 10,),

                        Container(height: 3, color: Colors.white,),

                        const SizedBox(height: 10,),

                        const Text("Tugas1", style: TextStyle(fontSize: 16,
fontWeight: FontWeight.w600),),

                        Text(catatanModel.tugas1!),

                        const SizedBox(height: 10,),

                        const Text("Tugas2", style: TextStyle(fontSize: 16,
fontWeight: FontWeight.w600),),

                        Text(catatanModel.tugas2!),

                        const SizedBox(height: 10,),

                        const Text("Tugas3", style: TextStyle(fontSize: 16,
fontWeight: FontWeight.w600),),

                        Text(catatanModel.tugas3!),
                      ],
                    ),
                  );
                },

              ),),
            )
          ],
```

```
        ),
      ),
    );
  }
}
```

Deklarasi catatanController yang menjadi instance dari blue print CatatanController dan gungsi ubahTanggal() untuk mengubah data tanggal yang ada di database nanti. Fungsi initState adalah method yang dikalankan pertama saat class HomeScreen() dipanggil atau berada di HomeScreen(). Di dalam AppBar bagian action terdapat icon add yang digunakan untuk menavigasikan pengguna ke halaman untuk memasukkan data. Lalu dibagian body bagian Container yang mempunyai child Obx(). Obx() adalah widget khusus untuk melakukan pengamatan (observe) perubahan pada variabel yang telah dideklarasikan sebagai observables. Ketika nilai dari variabel tersebut berubah maka widget Obx akan secara otomatis membangun ulang tampilan (rebuild) disesuaikan dengan perubahan tersebut.

Buka file add_catatan_screen.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:mcs_bab_3/catatan_controller.dart';
import 'package:get/get.dart';
import 'package:mcs_bab_3/catatan_model.dart';

class AddCatatanScreen extends StatefulWidget {
  const AddCatatanScreen({super.key});

  @override
  State<AddCatatanScreen> createState() => _AddCatatanScreenState();
}

class _AddCatatanScreenState extends State<AddCatatanScreen> {
  final CatatanController catatanController = Get.put(CatatanController());
  String pilihTanggal = "SelectDate";
  int pilihWarna = 1;
  TextEditingController tugas1Controller = TextEditingController();
  TextEditingController tugas2Controller = TextEditingController();
  TextEditingController tugas3Controller = TextEditingController();

  Future<void> selectDate() async{
    DateTime? setDate = await showDatePicker(
      context: context,
      initialDate: DateTime.now(),
      firstDate: DateTime(2024),
      lastDate: DateTime(2030),
    );

    if(setDate != null && setDate != DateTime.now()){
      setState(() {
        pilihTanggal = DateFormat.yMd().format(setDate).toString();
```

```
      });
    }
  }

  addCatatan() async{
    await catatanController.insert(
        catatanModel: CatatanModel(
          tanggal: pilihTanggal,
          warna: pilihWarna,
          tugas1: tugas1Controller.text,
          tugas2: tugas2Controller.text,
          tugas3: tugas3Controller.text,
    ));
    catatanController.getCatatanData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ListView(
        children: [
          const SizedBox(height: 20,),

          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            padding: EdgeInsets.all(14),
            width: double.infinity,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(14),
              boxShadow: const [
                BoxShadow(
                  color: Colors.white,
                  offset: Offset(0.0, 0.1),
                  blurRadius: 10.0,
                )
              ],
              color: const Color(0xff2E3840),
            ),
            child: const Text("add your note", style: TextStyle(fontSize:
16, fontWeight: FontWeight.w600),),
          ),

          const SizedBox(height: 30,),

          //pilih tanggal
          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            width: double.infinity,
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Text(pilihTanggal, style: const TextStyle(fontSize: 16,
fontWeight: FontWeight.w600),),
                GestureDetector(
                  child: const Icon(Icons.edit_calendar),
                  onTap: (){selectDate();},
                )
              ],
            ),
          ),
```

```
          const SizedBox(height: 20,),

          //pilih warna
          Row(
            mainAxisAlignment: MainAxisAlignment.end,
            children: [
              Wrap(
                children: List<Widget>.generate(4, (index){
                  return GestureDetector(
                    child: Container(
                      margin: EdgeInsets.symmetric(horizontal: 18),
                      height: MediaQuery.of(context).size.width/13,
                      width: MediaQuery.of(context).size.width/13,
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(50),
                        color: index == 0? Color(0xff4C4C6D) :
                        index == 1 ? Color(0xff6F61C0) :
                        index == 2 ? Colors.blue[900] :
                        Colors.pink[500]
                      ),
                      child: pilihWarna == index? Icon(Icons.done_all,) :
Container()
                    ),
                    onTap: (){
                      setState(() {
                        pilihWarna = index;
                      });
                    },
                  );
                }),
              ),
            ],
          ),

          const SizedBox(height: 20,),

          //isi tugas1
          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas 1"),
                TextFormField(
                  controller: tugas1Controller,
                  maxLines: 5,
                  textAlign: TextAlign.justify,
                  decoration: const InputDecoration(
                      focusedBorder: UnderlineInputBorder(
                        borderSide: BorderSide(color:
Colors.purpleAccent,),
                      )
                  ),
                ),
              ],
            ),
          ),

          const SizedBox(height: 30,),
```

```
          //isi tugas2
          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas 2"),
                TextFormField(
                  controller: tugas2Controller,
                  maxLines: 5,
                  textAlign: TextAlign.justify,
                  decoration: const InputDecoration(
                      focusedBorder: UnderlineInputBorder(
                        borderSide: BorderSide(color:
Colors.purpleAccent,),
                      )
                  ),
                ),
              ],
            ),
          ),

          const SizedBox(height: 30,),

          //isi tugas3
          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas 3"),
                TextFormField(
                  controller: tugas3Controller,
                  maxLines: 5,
                  textAlign: TextAlign.justify,
                  decoration: const InputDecoration(
                      focusedBorder: UnderlineInputBorder(
                          borderSide: BorderSide(color:
Colors.purpleAccent,)
                      )
                  ),
                ),
              ],
            ),
          ),

          const SizedBox(height: 30,),

          Container(
            margin: EdgeInsets.symmetric(horizontal: 18),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.end,
              children: [
                IconButton(
                  onPressed: () async{
                    await addCatatan();
                    Get.back();
                  },
                  icon: Icon(Icons.save_outlined),
                )
              ],
```

```
            ),
          )
        ],
      ),
    );
  }
}
```

Persiapkan semua variabel dan fungsi yang diperlukan dan lanjutkan pembuatan UI nya. Fungsi addCatatan() berguna untuk memasukkan data dari semua form input ke database dan tambahkan getCatatanModel agar obx dapat langsung membaca kondisi yang ada di database.

Buka file edit_screen.dart dan isi code berikut :

```
import 'package:flutter/material.dart';
import 'package:mcs_bab_3/catatan_controller.dart';
import 'package:mcs_bab_3/catatan_model.dart';
import 'package:mcs_bab_3/db.dart';
import 'package:get/get.dart';

class EditScreen extends StatefulWidget {
  CatatanModel catatanModel;

  EditScreen({
    super.key,
    required this.catatanModel,
  });

  @override
  State<EditScreen> createState() => _EditScreenState();
}

class _EditScreenState extends State<EditScreen> {
  final CatatanController catatanController = Get.put(CatatanController());
  TextEditingController tugas1Controller = TextEditingController();
  TextEditingController tugas2Controller = TextEditingController();
  TextEditingController tugas3Controller = TextEditingController();

  @override
  void initState() {
    tugas1Controller.text = widget.catatanModel.tugas1 ?? "";
    tugas2Controller.text = widget.catatanModel.tugas2 ?? "";
    tugas3Controller.text = widget.catatanModel.tugas3 ?? "";
    // TODO: implement initState
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ListView(
        children:[
          const SizedBox(height: 22,),
```

```
          Container(
            margin: const EdgeInsets.symmetric(horizontal: 18),
            padding: const EdgeInsets.all(14),
            width: double.infinity,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(14),
              border: Border.all(width: 3, ),
              boxShadow:  const [
                BoxShadow(
                  color: Colors.white,
                  offset: Offset(0.0, 0.1),
                  blurRadius: 10,
                ),
              ],
              color: const Color(0xff2E3840),
            ),
            child: const Text("Edit your note", style: TextStyle(fontSize:
16, fontWeight: FontWeight.w600),),
          ),

          const SizedBox(height: 30,),

          Container(
            margin: const EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas1"),
                TextFormField(
                  controller: tugas1Controller,
                  maxLines: null,
                ),
              ],
            ),
          ),

          const SizedBox(height: 20,),
          Container(
            margin: const EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas2"),
                TextFormField(
                  controller: tugas2Controller,
                  maxLines: null,
                ),
              ],
            ),
          ),

          const SizedBox(height: 20,),

          Container(
            margin: const EdgeInsets.symmetric(horizontal: 18),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                const Text("Tugas3"),
                TextFormField(
                  controller: tugas3Controller,
```

```
                  maxLines: null,
                ),
              ],
            ),
          ),

          const SizedBox(height: 20,),
        ],
      ),
      floatingActionButton: FloatingActionButton(
        shape: const CircleBorder(),
        backgroundColor: Colors.white,
        onPressed: () async{
          await DB.catatanDb!.rawUpdate(
            """UPDATE ${DB.catatanDbTable} SET tugas1 =
'${tugas1Controller.text}' WHERE id = ?""",
            [widget.catatanModel.id],
          );
          await DB.catatanDb!.rawUpdate(
            """UPDATE ${DB.catatanDbTable} SET tugas2 =
'${tugas2Controller.text}' WHERE id = ?""",
            [widget.catatanModel.id],
          );
          await DB.catatanDb!.rawUpdate(
            """UPDATE ${DB.catatanDbTable} SET tugas3 =
'${tugas3Controller.text}' WHERE id = ?""",
            [widget.catatanModel.id],
          );
          catatanController.getCatatanData();
          Get.back();
        },
        child: const Icon(Icons.save,),
      ),
    );
  }
}
```

Di bagian ini adalah tempat bagi pengguna untuk melakukan perubahan pada isi, yang mana setiap controller pada textformfield akan diisi sesuai dengan yang dipilih oleh pengguna. FloatingActionButton berguna untuk melakukan update ke database dengan isi dari controller tiap textformfield juga tambahkan getCatatanModel agar obx dapat langsung membaca kondisi yang ada di database.

LAPORAN PENDAHULUAN (LP)

1. Jelaskan apa otu database SQL!
2. Jelaskan apa itu DDL dan DML!
3. Jelaskan perbedaan basis data dengan system file tradisional!
4. Apa peran database pada suatu system informasi dan apa dampak jika suatu system informasi tidak memiliki database?

LAPORAN AKHIR (LA)

1. Berikan 3 point kesimpulan pada bab 3!