MODUL 15

Disusun Oleh :
Nama : Ganes Gemi Putra
Kelas : SE-07-02
NIM : (2311104075)

Dosen : YUDHA ISLAMI SULISTYA S.Kom M.CS

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2025

Pendahuluan

Dalam pengembangan perangkat lunak, penerapan design pattern sangat penting untuk menghasilkan kode yang terstruktur, mudah dipelihara, dan mudah dikembangkan. Pada modul 14 ini, mahasiswa diminta untuk mengimplementasikan salah satu design pattern yang telah dipelajari dengan cara mengubah kode dari tugas unguided sebelumnya.

Oleh karena itu, pada tugas ini dilakukan refactoring kode aplikasi Flutter dengan menerapkan design pattern **Model–View–ViewModel (MVVM)** guna memisahkan antara tampilan antarmuka, data, dan logika bisnis aplikasi.

Design Pattern MVVM

Model–View–ViewModel (MVVM) adalah design pattern yang memisahkan aplikasi menjadi tiga bagian utama, yaitu:

1. **Model**
   Berfungsi untuk merepresentasikan struktur data yang digunakan dalam aplikasi.
2. **View**
   Bertugas untuk menampilkan antarmuka pengguna serta menerima input dari pengguna.
3. **ViewModel**
   Berfungsi sebagai penghubung antara Model dan View yang mengelola state serta business logic aplikasi.

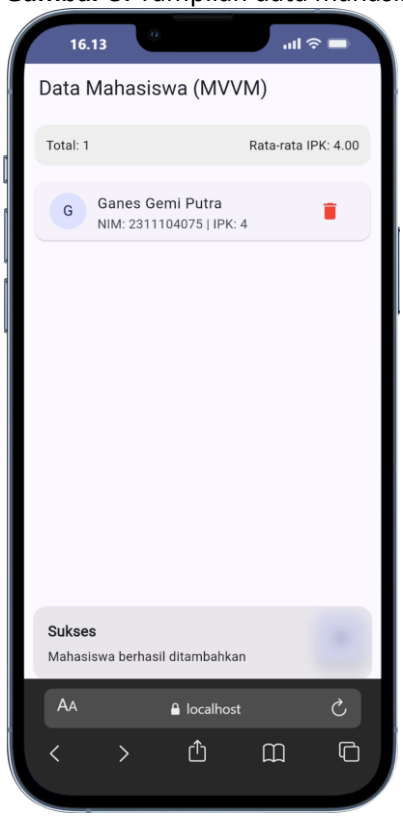
Penerapan MVVM bertujuan agar kode lebih modular, mudah diuji, dan memisahkan logika aplikasi dari tampilan.

Screenshot Output

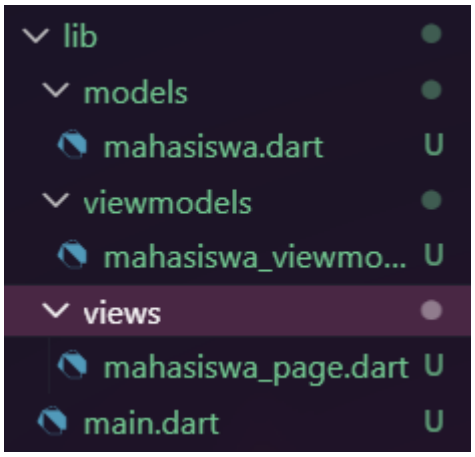
Gambar 1. Tampilan halaman utama aplikasi Data Mahasiswa

Gambar 2. Dialog tambah data mahasiswa

Gambar 3. Tampilan data mahasiswa setelah ditambahkan



SOURCECODE :



mahasiswa.dart :

```
class Mahasiswa {
  final String nama;
  final String nim;
  final double ipk;

  Mahasiswa({
    required this.nama,
    required this.nim,
    required this.ipk,
  });
}
```

mahasiswa_viewmodel.dart :

```
import 'package:get/get.dart';
import '../models/mahasiswa.dart';

/// ViewModel: tempat business logic + state aplikasi (MVVM).
class MahasiswaViewModel extends GetxController {
  final RxList<Mahasiswa> daftarMahasiswa = <Mahasiswa>[].obs;

  /// (Kreatifitas) Validasi sederhana + pesan error.
  String? validateInput(String nama, String nim, String ipkText) {
    if (nama.trim().isEmpty) return "Nama tidak boleh kosong.";
    if (nim.trim().isEmpty) return "NIM tidak boleh kosong.";
    final ipk = double.tryParse(ipkText.replaceAll(',', '.'));
    if (ipk == null) return "IPK harus angka.";
    if (ipk < 0 || ipk > 4) return "IPK harus 0.0 - 4.0.";
    return null;
  }

  void tambahMahasiswa(String nama, String nim, double ipk) {
    daftarMahasiswa.add(Mahasiswa(nama: nama.trim(), nim: nim.trim(), ipk: ipk));
  }

  void hapusMahasiswa(int index) {
```

```
    if (index >= 0 && index < daftarMahasiswa.length) {
      daftarMahasiswa.removeAt(index);
    }
  }

  int get totalMahasiswa => daftarMahasiswa.length;

  double get rataRataIpk {
    if (daftarMahasiswa.isEmpty) return 0;
    final total = daftarMahasiswa.fold<double>(0, (sum, mhs) => sum + mhs.ipk);
    return total / daftarMahasiswa.length;
  }
}
```

mahasiswa_page.dart :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../viewmodels/mahasiswa_viewmodel.dart';

/// View: hanya tampilan + input user. Logic ada di ViewModel (MVVM).
class MahasiswaPage extends StatelessWidget {
  MahasiswaPage({super.key});

  final MahasiswaViewModel vm = Get.put(MahasiswaViewModel());

  void _showTambahDialog(BuildContext context) {
    final namaC = TextEditingController();
    final nimC = TextEditingController();
    final ipkC = TextEditingController();

    Get.dialog(
      AlertDialog(
        title: const Text("Tambah Mahasiswa"),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: namaC,
              decoration: const InputDecoration(labelText: "Nama"),
            ),
            TextField(
              controller: nimC,
              decoration: const InputDecoration(labelText: "NIM"),
            ),
            TextField(
              controller: ipkC,
              keyboardType: TextInputType.number,
              decoration: const InputDecoration(labelText: "IPK (0 - 4)"),
            ),
```

```
        ],
      ),
      actions: [
       TextButton(
        onPressed: () => Get.back(),
        child: const Text("Batal"),
       ),
       ElevatedButton(
        onPressed: () {
         final err = vm.validateInput(namaC.text, nimC.text, ipkC.text);
         if (err != null) {
          Get.snackbar("Validasi Gagal", err, snackPosition: SnackPosition.BOTTOM);
          return;
         }
         final ipk = double.parse(ipkC.text.replaceAll(',', '.'));
         vm.tambahMahasiswa(namaC.text, nimC.text, ipk);
         Get.back();
         Get.snackbar("Sukses", "Mahasiswa berhasil ditambahkan",
           snackPosition: SnackPosition.BOTTOM);
        },
        child: const Text("Simpan"),
       ),
      ],
    ),
   );
}

@override
Widget build(BuildContext context) {
 return Scaffold(
   appBar: AppBar(
    title: const Text("Data Mahasiswa (MVVM)"),
   ),
   floatingActionButton: FloatingActionButton(
    onPressed: () => _showTambahDialog(context),
    child: const Icon(Icons.add),
   ),
   body: Column(
    children: [
     // Ringkasan (kreatifitas): card ringkasan.
     Obx(() {
      return Container(
       width: double.infinity,
       margin: const EdgeInsets.all(12),
       padding: const EdgeInsets.all(12),
       decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(12),
        color: Colors.grey.shade200,
       ),
       child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
```

```
          children: [
            Text("Total: ${vm.totalMahasiswa}"),
            Text("Rata-rata IPK: ${vm.rataRataIpk.toStringAsFixed(2)}"),
          ],
        ),
      );
    }),

    Expanded(
      child: Obx(() {
        if (vm.daftarMahasiswa.isEmpty) {
          return const Center(child: Text("Belum ada data mahasiswa."));
        }

        return ListView.builder(
          itemCount: vm.daftarMahasiswa.length,
          itemBuilder: (context, index) {
            final m = vm.daftarMahasiswa[index];
            return Card(
              margin: const EdgeInsets.symmetric(horizontal: 12, vertical: 6),
              child: ListTile(
                leading: CircleAvatar(
                  child: Text(m.nama.isNotEmpty ? m.nama[0].toUpperCase() : "?"),
                ),
                title: Text(m.nama),
                subtitle: Text("NIM: ${m.nim} | IPK: ${m.ipk}"),
                trailing: IconButton(
                  icon: const Icon(Icons.delete, color: Colors.red),
                  onPressed: () {
                    Get.defaultDialog(
                      title: "Hapus Data",
                      middleText: "Yakin hapus ${m.nama}?",
                      textCancel: "Batal",
                      textConfirm: "Hapus",
                      confirmTextColor: Colors.white,
                      onConfirm: () {
                        vm.hapusMahasiswa(index);
                        Get.back();
                        Get.snackbar("Terhapus", "Data berhasil dihapus",
                            snackPosition: SnackPosition.BOTTOM);
                      },
                    );
                  },
                ),
              ),
            );
          },
        );
      }),
    ),
  ],
```

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

main.dart :

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'views/mahasiswa_page.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Modul 14 -> MVVM',
      theme: ThemeData(
        useMaterial3: true,
        colorSchemeSeed: Colors.indigo,
      ),
      home: MahasiswaPage(),
    );
  }
}
```

Kesimpulan

Berdasarkan hasil implementasi yang telah dilakukan, penerapan design pattern MVVM pada aplikasi Flutter dapat meningkatkan kerapihan struktur kode serta memisahkan antara tampilan dan logika bisnis aplikasi.

Dengan adanya pemisahan antara Model, View, dan ViewModel, pengembangan aplikasi menjadi lebih terorganisir, mudah dipelihara, serta lebih fleksibel untuk dikembangkan di masa mendatang.