

**TUGAS PENDAHULUAN / TUGAS UNGUIDED
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XV
DESIGN PATTERN**



Disusun Oleh :

Farhan Kurniawan / 2311104073

SE-07-02

Asisten Praktikum :

**Zulfa Mustafa Akhyar Iswahyudi
Yoga Eka Pratama**

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

TUGAS PENDAHULUAN / TUGAS UNGUIDED

A. SOAL

Tugas Mandiri (Unguided)

Dari tugas unguided modul 14 yang telah dikerjakan, ubahlah kodenya sehingga sesuai dengan salah satu design pattern yang telah dipelajari.

Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program.

Kreatifitas menjadi nilai tambah.

B. JAWABAN

Source Code pada Main Dart

```
//models/post_models.dart
class PostModel {
  final int id;
  final String title;
  final String body;

  PostModel({required this.id, required this.title, required this.body});

  factory PostModel.fromJson(Map<String, dynamic> json) {
    return PostModel(id: json['id'], title: json['title'], body: json['body']);
  }
}

//services/api_services.dart
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/post_model.dart';

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";

  Future<List<PostModel>> fetchPosts() async {
    final response = await http.get(Uri.parse('$baseUrl/posts'));
  }
}
```

```
if (response.statusCode == 200) {  
    List data = json.decode(response.body);  
    return data.map((e) => PostModel.fromJson(e)).toList();  
} else {  
    throw Exception('Gagal mengambil data');  
}  
}
```

```
Future<void> createPost() async {  
    await http.post(  
        Uri.parse('$baseUrl/posts'),  
        headers: {'Content-Type': 'application/json'},  
        body: json.encode({  
            'title': 'Post Baru',  
            'body': 'Contoh MVVM',  
            'userId': 1,  
        })),  
    );  
}
```

```
Future<void> updatePost() async {  
    await http.put(  
        Uri.parse('$baseUrl/posts/1'),  
        headers: {'Content-Type': 'application/json'},  
        body: json.encode({  
            'title': 'Update MVVM',  
            'body': 'Data diperbarui',  
            'userId': 1,  
        })),  
    );  
}
```

```
Future<void> deletePost() async {  
    await http.delete(Uri.parse('$baseUrl/posts/1'));  
}
```

```
}

//viewmodels/post_viewmodel.dart
import 'package:get/get.dart';
import '../models/post_model.dart';
import '../services/api_service.dart';

class PostViewModel extends GetxController {
  final ApiService apiService = ApiService();

  var posts = <PostModel>[].obs;
  var isLoading = false.obs;

  void getPosts() async {
    try {
      isLoading(true);
      posts.value = await apiService.fetchPosts();
      Get.snackbar("Sukses", "Data berhasil diambil");
    } finally {
      isLoading(false);
    }
  }

  void addPost() async {
    await apiService.createPost();
    Get.snackbar("Sukses", "Data berhasil ditambahkan");
  }

  void updatePost() async {
    await apiService.updatePost();
    Get.snackbar("Sukses", "Data berhasil diperbarui");
  }

  void deletePost() async {
    await apiService.deletePost();
  }
}
```

```

    Get.snackbar("Sukses", "Data berhasil dihapus");
  }
}

//views/home_view.dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../viewmodels/post_viewmodel.dart';

class HomeView extends StatelessWidget {
  final PostViewModel viewModel = Get.put(PostViewModel());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("MVVM - REST API")),
      body: Column(
        children: [
          Wrap(
            spacing: 8,
            children: [
              ElevatedButton(onPressed: viewModel.getPosts, child: Text("GET")),
              ElevatedButton(onPressed: viewModel.addPost, child: Text("POST")),
              ElevatedButton(
                onPressed: viewModel.updatePost,
                child: Text("PUT"),
              ),
              ElevatedButton(
                onPressed: viewModel.deletePost,
                child: Text("DELETE"),
              ),
            ],
          ),
          Expanded(
            child: Obx() {

```

```

        if (viewModel.isLoading.value) {
            return Center(child: CircularProgressIndicator());
        }

        return ListView.builder(
            itemCount: viewModel.posts.length,
            itemBuilder: (context, index) {
                final post = viewModel.posts[index];
                return Card(
                    child: ListTile(
                        title: Text(post.title),
                        subtitle: Text(post.body),
                    ),
                );
            },
        );
    },
),
],
),
);
}
}

```

//main.dart

```

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

```

```

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

```

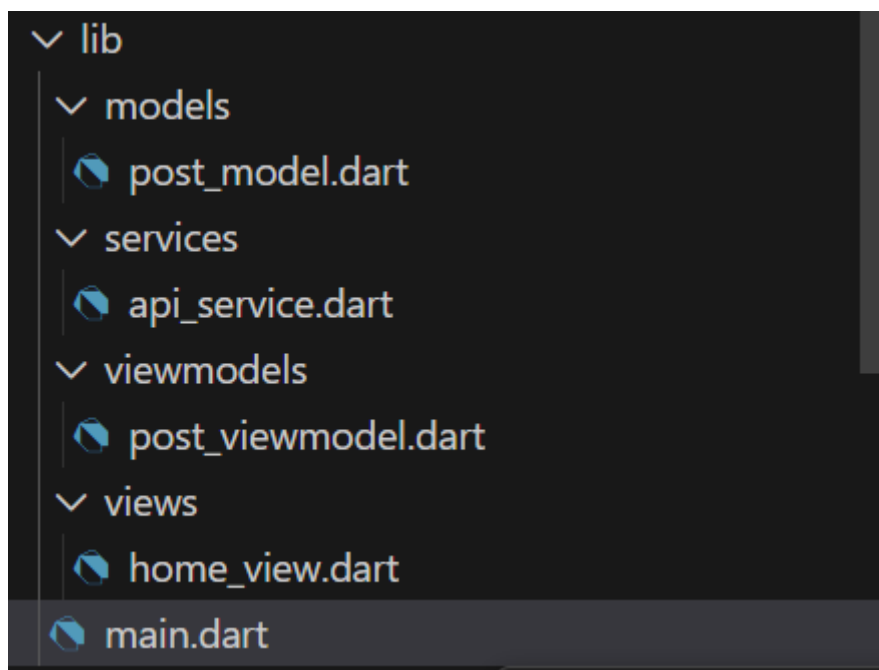
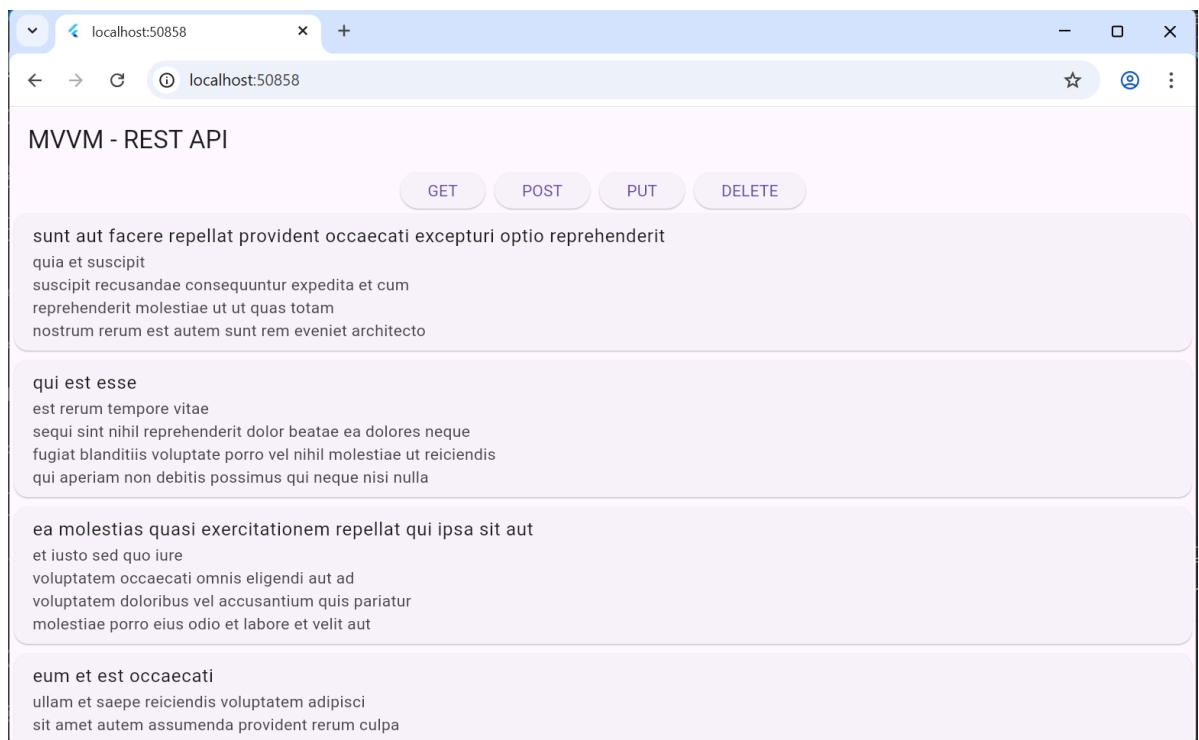
```

class MyApp extends StatelessWidget {
    @override

```

```
Widget build(BuildContext context) {  
    return GetMaterialApp(debugShowCheckedModeBanner: false, home:  
    HomeView());  
}  
}
```

Screenshot Output:



Deskripsi Program

Program ini merupakan aplikasi mobile berbasis **Flutter** yang dikembangkan dengan menerapkan **Design Pattern MVVM (Model–View–ViewModel)** sebagai arsitektur utama aplikasi. Aplikasi ini merupakan pengembangan dari Unguided Modul 14 yang sebelumnya masih menggunakan pendekatan controller, kemudian direstrukturisasi agar sesuai dengan konsep pemisahan tanggung jawab pada pola MVVM.

Pada aplikasi ini, **Model** digunakan untuk merepresentasikan struktur data Post yang diperoleh dari REST API, **View** bertanggung jawab untuk menampilkan antarmuka pengguna serta menerima interaksi dari pengguna, sedangkan **ViewModel** berfungsi sebagai penghubung antara View dan Model yang mengelola logika aplikasi serta proses pengambilan dan pengelolaan data dari API. Proses komunikasi data dilakukan menggunakan metode HTTP **GET, POST, PUT, dan DELETE** dengan memanfaatkan layanan API publik.

Dengan penerapan MVVM, aplikasi menjadi lebih terstruktur, mudah dipelihara, serta memudahkan pengembangan lanjutan. Selain itu, penggunaan **GetX** membantu pengelolaan state secara reaktif dan menampilkan notifikasi berupa snackbar setiap kali operasi data berhasil dilakukan, sehingga meningkatkan pengalaman pengguna dalam menggunakan aplikasi.