UNGUIDED
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE API

Disusun Oleh :

Ilham Lii Assidaq

2311104068

Asisten Praktikum :

Yoga Eka Pratama

Zulfa Mustafa Akhyar Iswahyudi

Dosen Pengampu :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

UNGUIDED

1. SOAL

A. Modifikasi tampilan Guided dari praktikum di atas:

1. Gunakan State Management dengan GetX
   - Atur data menggunakan state management GetX agar lebih mudah dikelola.
   - Implementasi GetX meliputi pembuatan controller untuk mengelola data dan penggunaan widget Obx untuk menampilkan data secara otomatis setiap kali ada perubahan.
2. Tambahkan Snackbar untuk Memberikan Respon Berhasil:
   - Tampilkan snackbar setelah setiap operasi berhasil, seperti menambah atau memperbarui data
   - Gunakan Get.snackbar agar pesan sukses muncul di layar dan mudah dipahami oleh pengguna

Sourcecode

Controllers/post_controller.dart

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

class PostController extends GetxController {
  var posts = <PostModel>[].obs;
  var isLoading = false.obs;
  final ApiService _apiService = ApiService();


  void _showSuccess(String msg) {
    Get.snackbar("Sukses", msg, snackPosition: SnackPosition.BOTTOM);
  }

  Future<void> fetchPosts() async {
    try {
      isLoading(true);
      final result = await _apiService.fetchPosts();
      posts.assignAll(result);
      _showSuccess("Data berhasil diambil!");
    } catch (e) {
      Get.snackbar("Error", e.toString());
```

```
    } finally {
      isLoading(false);
    }
  }

  Future<void> createPost() async {
    try {
      isLoading(true);
      final newPost = PostModel(
        title: "New Post",
        body: "This is a new post created via API",
        userId: 1,
      );
      await _apiService.createPost(newPost);

      final result = await _apiService.fetchPosts();
      posts.assignAll(result);
      _showSuccess("Data berhasil ditambahkan!");
    } catch (e) {
      Get.snackbar("Error", e.toString());
    } finally {
      isLoading(false);
    }
  }

  Future<void> deletePost(int postId) async {
    try {
      isLoading(true);
      await _apiService.deletePost(postId);
      posts.removeWhere((post) => post.id == postId);
      _showSuccess("Data berhasil dihapus!");
    } catch (e) {
      Get.snackbar("Error", e.toString());
    } finally {
      isLoading(false);
    }
  }
}
```

Models/post_model.dart

```
class PostModel {
  final int? id;
  final String title;
  final String body;
  final int userId;
```

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

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


  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'title': title,
      'body': body,
      'userId': userId,
    };
  }
}
```

Services/api_service.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((item) => PostModel.fromJson(item)).toList();
    } else {
      throw Exception('Gagal memuat data');
    }
  }

  Future<PostModel> createPost(PostModel post) async {
```

```
    final response = await http.post(
      Uri.parse('$baseUrl/posts'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(post.toJson()),
    );
    if (response.statusCode == 201) {
      return PostModel.fromJson(json.decode(response.body));
    } else {
      throw Exception('Gagal menambah data');
    }
  }

  Future<void> deletePost(int postId) async {
    final response = await http.delete(Uri.parse('$baseUrl/posts/$postId'));
    if (response.statusCode != 200) {
      throw Exception('Gagal menghapus data');
    }
  }
}
```

Views/home_Screen.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../controllers/post_controller.dart';

class HomeScreen extends StatelessWidget {
  // Inisialisasi Controller
  final PostController controller = Get.put(PostController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("REST API GetX")),
      body: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton(
                onPressed: controller.fetchPosts,
                child: Text("GET"),
              ),
              ElevatedButton(
                onPressed: controller.createPost,
                child: Text("POST"),
              ),
```

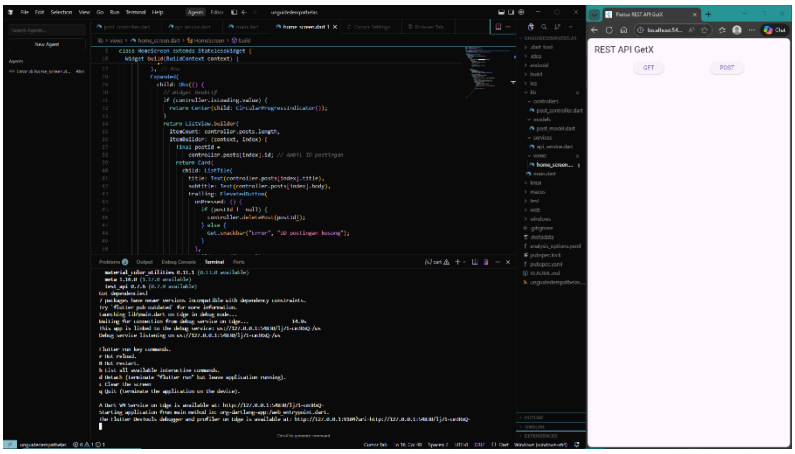
```
            ],
          ),
          Expanded(
            child: Obx(() {
              // Widget Reaktif
              if (controller.isLoading.value) {
                return Center(child: CircularProgressIndicator());
              }
              return ListView.builder(
                itemCount: controller.posts.length,
                itemBuilder: (context, index) {
                  final postId =
                      controller.posts[index].id; // Ambil ID postingan
                  return Card(
                    child: ListTile(
                      title: Text(controller.posts[index].title),
                      subtitle: Text(controller.posts[index].body),
                      trailing: ElevatedButton(
                        onPressed: () {
                          if (postId != null) {
                            controller.deletePost(postId!);
                          } else {
                            Get.snackbar("Error", "ID postingan kosong");
                          }
                        },
                        child: Text("DELETE"),
                      ),
                    ),
                  );
                },
              );
            }),
          ),
        ],
      ),
    );
  }
}
```

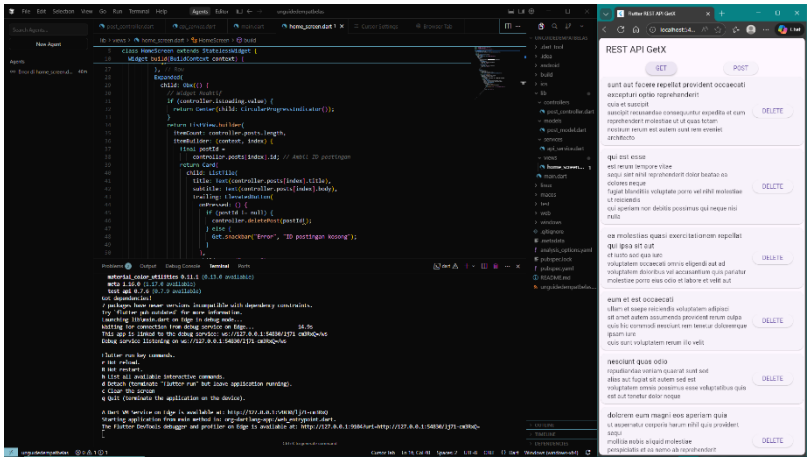
Lib/main.dart

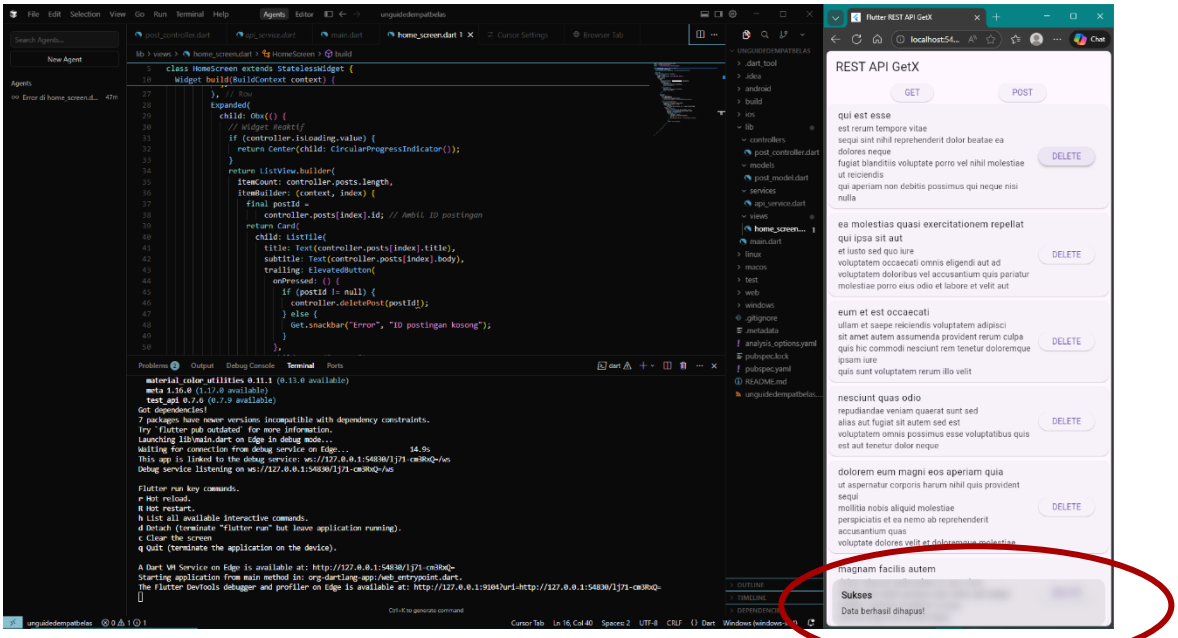
```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'views/home_screen.dart';

void main() {
```

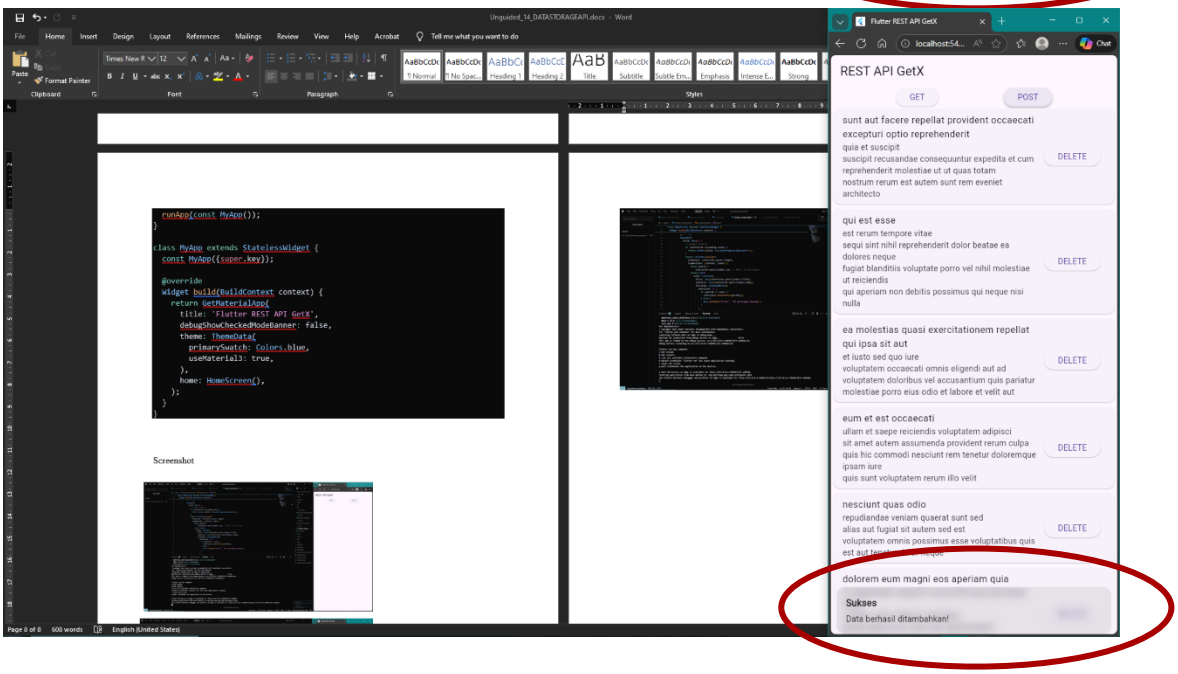
```
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Flutter REST API GetX',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        useMaterial3: true,
      ),
      home: HomeScreen(),
    );
  }
}
```

Screenshot









Penjelasan:

Aplikasi Flutter ini merupakan implementasi modern dari integrasi REST API yang memanfaatkan GetX sebagai solusi state management reaktif. Dengan struktur proyek yang terorganisir ke dalam folder controllers, models, services, dan views, aplikasi ini berhasil memisahkan logika bisnis dari antarmuka pengguna secara efisien. Fitur utama yang ditawarkan mencakup operasi CRUD lengkap, mulai dari pengambilan daftar postingan melalui metode GET, penambahan data baru dengan POST, hingga penghapusan data menggunakan metode DELETE. Penggunaan teknologi pendukung seperti paket HTTP dan API mock dari JSONPlaceholder memungkinkan aplikasi untuk mensimulasikan interaksi server secara real-time dalam lingkungan pengembangan yang stabil.

Dalam operasionalnya, aplikasi ini mengandalkan konsep reaktivitas GetX di mana variabel yang ditandai dengan .obs di dalam PostController akan secara otomatis memicu pembaruan pada widget Obx di HomeScreen setiap kali terjadi perubahan data. Proses komunikasi data dikelola oleh ApiService yang bertugas melakukan parsing JSON menjadi objek PostModel, sementara pengguna mendapatkan umpan balik visual yang intuitif melalui indikator loading dan notifikasi snackbar yang muncul setelah setiap aksi berhasil dilakukan. Secara keseluruhan, proyek ini menunjukkan penerapan arsitektur perangkat lunak yang bersih, memadukan kemudahan navigasi GetX dengan keandalan Flutter SDK untuk menciptakan aplikasi mobile yang responsif, mudah dipelihara, dan siap untuk dikembangkan lebih lanjut ke skala yang lebih kompleks.