

**LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XIV
DATA STORAGE
'API'**



Disusun Oleh :

Dwi Candra Pratama / 2211104035

SE06-02

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

GUIDED

DATA STORAGE API

Tujuan Praktikum
<ol style="list-style-type: none">1. Memahami konsep dasar database web service menggunakan REST API.2. Mengetahui kegunaan dan penerapan metode HTTP (GET, POST, PUT, DELETE) dalam pengelolaan data.3. Membuat antarmuka pengguna (UI) yang menampilkan data dari API serta memungkinkan interaksi dengan server.

A. Apa itu REST API

REST API (Representational State Transfer Application Programming Interface) adalah antarmuka yang memungkinkan aplikasi klien untuk berinteraksi dengan database melalui protokol HTTP. REST API menyediakan cara untuk membaca, menambahkan, memperbarui, dan menghapus data dari database tanpa harus mengakses database langsung. Mendapatkan token unik dari setiap perangkat pengguna.

Kegunaan REST API :

- a. Interoperabilitas: REST API memungkinkan aplikasi berbasis web dan mobile untuk mengakses data yang sama.
- b. Efisiensi: Data yang dikirimkan biasanya ringan (format JSON atau XML), membuatnya cepat untuk dikirim dan diterima.
- c. Keamanan: API bisa membatasi akses menggunakan token autentikasi.

B. Apa itu HTTP

HTTP (Hypertext Transfer Protocol) adalah protokol komunikasi utama yang digunakan untuk mengirimkan data antara klien (misalnya browser atau aplikasi) dan server.

Metode HTTP Utama :

1. GET: Mengambil data dari server.
2. POST: Mengirim data baru ke server.
3. PUT/PATCH: Memperbarui data yang ada di server.
4. DELETE: Menghapus data dari server.

Komponen HTTP Request

1. URL: Alamat yang menunjukkan resource tertentu.
2. Method: Operasi yang akan dilakukan (GET, POST, dll.).
3. Headers: Informasi tambahan seperti format data atau token autentikasi.
4. Body: Data yang dikirimkan (digunakan dalam POST/PUT).

Komponen HTTP Response

1. Status Code: Menunjukkan hasil operasi (misalnya, 200 untuk berhasil, 404 untuk resource

tidak ditemukan).

2. Headers: Informasi tambahan dari server.

3. Body: Data yang dikembalikan server (biasanya dalam format JSON).

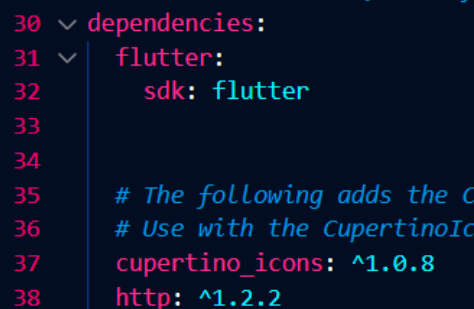
C. Praktikum

Langkah-langkah implementasi REST API di Flutter meliputi Implementasi HTTP GET, Implementasi HTTP POST, Implementasi HTTP PUT, & Implementasi HTTP DELETE.

a. Buat proyek Flutter baru

b. Tambahkan dependensi http ke file pubspec.yaml:

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.8  
  http: ^1.2.2
```



```
30 dependencies:  
31 flutter:  
32   sdk: flutter  
33  
34  
35 # The following adds the Cupertino Icons  
36 # Use with the CupertinoIcons class  
37 cupertino_icons: ^1.0.8  
38 http: ^1.2.2
```

c. Berikut implementasi Codenya

Folder **Screen** isi file **homepage.dart**

```
import 'package:flutter/material.dart';  
import 'package:pertemuan_14/services/api_service.dart';  
  
class HomepageScreen extends StatefulWidget {  
  const HomepageScreen({super.key});  
  
  @override  
  State<HomepageScreen> createState() => _HomepageScreenState();  
}  
  
class _HomepageScreenState extends State<HomepageScreen> {  
  List<dynamic> _posts = [];  
  bool _isLoading = false;  
  final ApiService _apiService = ApiService();  
  
  void _showSnackBar(String message) {  
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:  
Text(message)));
```

```

}

Future<void> _handleApiOperation(Future<void> operation, String
successMessage) async {
  setState() {
    _isLoading = true;
  });
  try {
    await operation;
    setState() {
      _posts = _apiService.posts;
    });
    _showSnackBar(successMessage);
  } catch (e) {
    _showSnackBar('Error: $e');
  } finally {
    setState() {
      _isLoading = false;
    });
  }
}

Future<void> _fetchPosts() async {
  await _handleApiOperation(_apiService.fetchPosts(), 'Data berhasil
diambil!');
}

Future<void> _createPost() async {
  await _handleApiOperation(_apiService.createPost(), 'Data berhasil
ditambahkan!');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Homepage'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // Menampilkan indikator loading atau pesan kosong
          if (_isLoading)
            const Center(child: CircularProgressIndicator())

```

```

else if (_posts.isEmpty)
  const Text(
    "Tekan tombol GET untuk mengambil data",
    style: TextStyle(fontSize: 12),
  )
else
  Expanded(
    child: ListView.builder(
      itemCount: _posts.length,
      itemBuilder: (context, index) {
        return Padding(
          padding: const EdgeInsets.only(bottom: 12.0),
          child: Card(
            elevation: 4,
            child: ListTile(
              title: Text(
                _posts[index]['title'] ?? 'No Title',
                style: const TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 12,
                ),
              ),
              subtitle: Text(
                _posts[index]['body'] ?? 'No Content',
                style: const TextStyle(fontSize: 12),
              ),
            ),
          ),
        );
      },
    ),
  ),
  const SizedBox(height: 10),

  // Tombol GET, POST, UPDATE, DELETE dalam satu baris
  Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      ElevatedButton(
        onPressed: _fetchPosts,
        style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
        child: const Text('GET'),
      ),
      ElevatedButton(
        onPressed: _createPost,
        style: ElevatedButton.styleFrom(backgroundColor: Colors.green),

```

```

        child: const Text('POST'),
      ),
      ElevatedButton(
        onPressed: () => _handleApiOperation(
          _apiService.updatePost(), 'Data berhasil diperbarui!'),
        style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
        child: const Text('UPDATE'),
      ),
      ElevatedButton(
        onPressed: () => _handleApiOperation(
          _apiService.deletePost(), 'Data berhasil dihapus!'),
        style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
        child: const Text('DELETE'),
      ),
    ],
  ),
],
),
),
);
}
}

```

Folder **Services** isi file **api_services.dart**

```

import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
  List<dynamic> posts = []; // Menyimpan data post yang diterima

  // Fungsi untuk GET data
  Future<void> fetchPosts() async {
    final response = await http.get(Uri.parse('$baseUrl/posts'));

    if (response.statusCode == 200) {
      posts = json.decode(response.body);
    } else {
      throw Exception('Failed to load posts');
    }
  }

  // Fungsi untuk POST data
  Future<void> createPost() async {
    final response = await http.post(
      Uri.parse('$baseUrl/posts'),

```

```

headers: {'Content-Type': 'application/json'},
body: json.encode({
  'title': 'Flutter Post',
  'body': 'Ini contoh POST.',
  'userId': 1,
}),
);

if (response.statusCode == 201) {
  posts.add({
    'title': 'Flutter Post',
    'body': 'Ini contoh POST.',
    'id': posts.length + 1,
  });
} else {
  throw Exception('Failed to create post');
}
}

// Fungsi untuk UPDATE data
Future<void> updatePost() async {
  final response = await http.put(
    Uri.parse('$baseUrl/posts/1'),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'title': 'Updated Title',
      'body': 'Updated Body',
      'userId': 1,
    }),
  );

  if (response.statusCode == 200) {
    final updatedPost = posts.firstWhere((post) => post['id'] == 1, orElse: () =>
null);
    if (updatedPost != null) {
      updatedPost['title'] = 'Updated Title';
      updatedPost['body'] = 'Updated Body';
    }
  } else {
    throw Exception('Failed to update post');
  }
}

// Fungsi untuk DELETE data
Future<void> deletePost() async {
  final response = await http.delete(

```

```
Uri.parse('$baseUrl/posts/1'),
);

if (response.statusCode == 200) {
  posts.removeWhere((post) => post['id'] == 1);
} else {
  throw Exception('Failed to delete post');
}
}
```

Main.dart

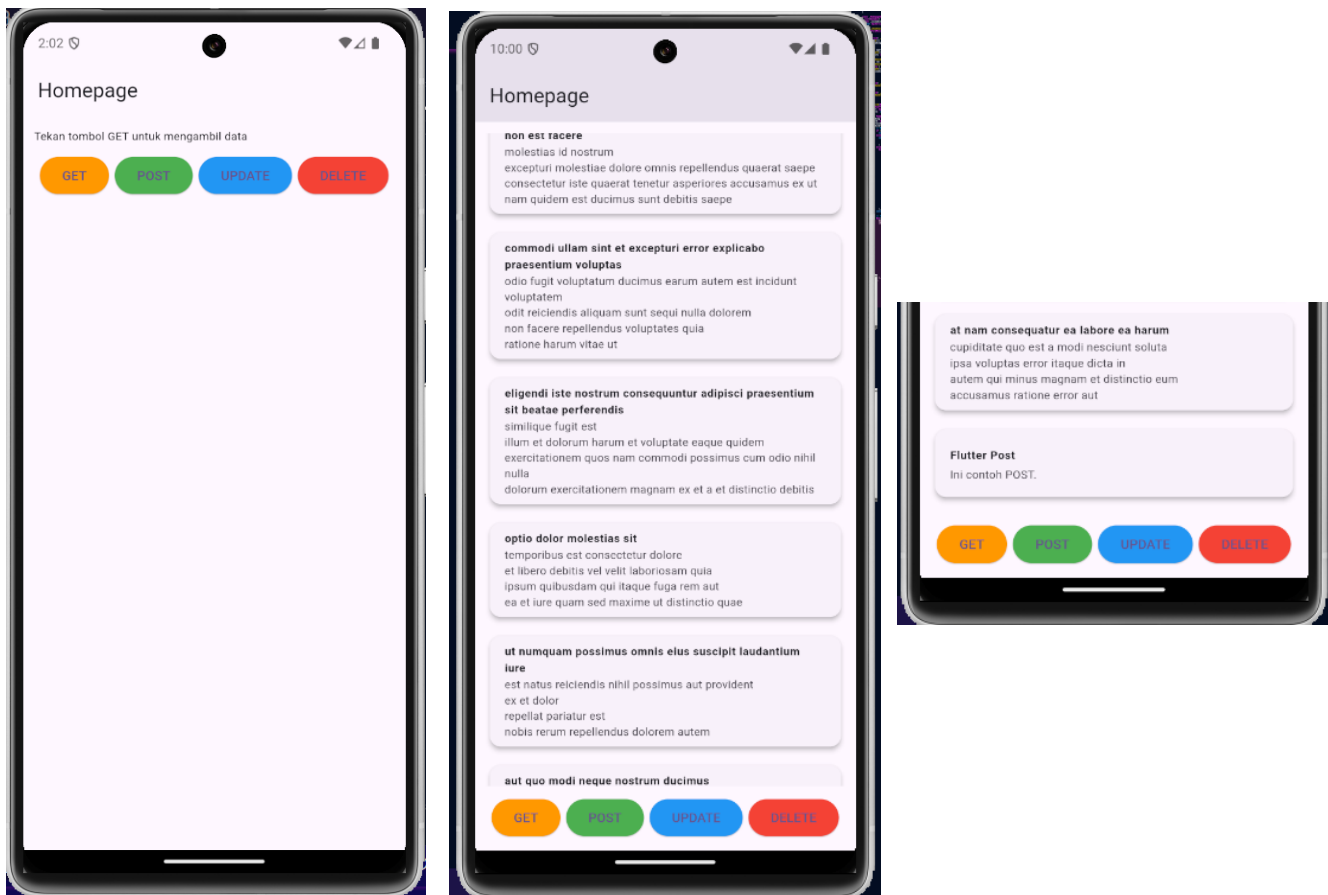
```
import 'package:flutter/material.dart';
import 'package:pertemuan_14/screen/homepage_screen.dart';

void main() {
  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 MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: HomepageScreen(),
    );
  }
}
```


OutPutnya:



Tugas Mandiri (Unguided)

Modifikasi tampilan Guided dari praktikum di atas:

- 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.
- 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.

Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program. Kreativitas menjadi nilai tambah.

Jawaban

SourcCodanya:

Folder **Controllers** isi File **post_controllers.dart**

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan_14/services/api_service.dart';

class PostController extends GetxController {
  final ApiService _apiService = ApiService();
  var posts = [].obs; // State untuk menyimpan data post
  var isLoading = false.obs; // State untuk indikator loading

  // Fungsi untuk GET data
  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      await _apiService.fetchPosts();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil diambil!',
        backgroundColor: Colors.orange,
        colorText: Colors.white,
        snackPosition: SnackPosition.TOP,
      );
    } catch (e) {
      Get.snackbar('Error', 'Gagal mengambil data: $e');
    } finally {
      isLoading.value = false;
    }
  }

  // Fungsi untuk POST data
  Future<void> createPost() async {
    try {
      await _apiService.createPost();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil ditambahkan!',
        backgroundColor: Colors.green,
        colorText: Colors.white,
        snackPosition: SnackPosition.BOTTOM,
      );
    } catch (e) {
      Get.snackbar('Error', 'Gagal menambahkan data: $e');
    }
  }
}
```

```

// Fungsi untuk UPDATE data
Future<void> updatePost() async {
  try {
    await _apiService.updatePost();
    posts.assignAll(_apiService.posts);
    Get.snackbar('Success', 'Data berhasil diperbarui!',
      backgroundColor: Colors.blue,
      colorText: Colors.white,
      snackPosition: SnackPosition.TOP,
    );
  } catch (e) {
    Get.snackbar('Error', 'Gagal memperbarui data: $e');
  }
}

// Fungsi untuk DELETE data
Future<void> deletePost() async {
  try {
    await _apiService.deletePost();
    posts.assignAll(_apiService.posts);
    Get.snackbar('Success', 'Data berhasil dihapus!',
      backgroundColor: Colors.red,
      colorText: Colors.white,
      snackPosition: SnackPosition.BOTTOM,
    );
  } catch (e) {
    Get.snackbar('Error', 'Gagal menghapus data: $e');
  }
}
}

```

Folder **Screen** isi File **homepage_screen.dart**

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan_14/controllers/post_controllers.dart';

class HomepageScreen extends StatelessWidget {
  final PostController postController = Get.put(PostController());

  HomepageScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Homepage'),

```

```

),
body: Padding(
  padding: const EdgeInsets.all(12.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      // Menampilkan indikator loading atau pesan kosong
      Obx(() {
        if (postController.isLoading.value) {
          return const Center(child: CircularProgressIndicator());
        } else if (postController.posts.isEmpty) {
          return const Text(
            "Tekan tombol GET untuk mengambil data",
            style: TextStyle(fontSize: 12),
          );
        } else {
          return Expanded(
            child: ListView.builder(
              itemCount: postController.posts.length,
              itemBuilder: (context, index) {
                final post = postController.posts[index];
                return Padding(
                  padding: const EdgeInsets.only(bottom: 12.0),
                  child: Card(
                    elevation: 4,
                    child: ListTile(
                      title: Text(
                        post['title'] ?? 'No Title',
                        style: const TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 12,
                        ),
                      ),
                      subtitle: Text(
                        post['body'] ?? 'No Content',
                        style: const TextStyle(fontSize: 12),
                      ),
                    ),
                  ),
                );
              },
            );
          );
        }
      })
    ],
  ),
),

```

```

const SizedBox(height: 10),

// Tombol GET, POST, UPDATE, DELETE dalam satu baris
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    ElevatedButton(
      onPressed: postController.fetchPosts,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
      child: const Text('GET'),
    ),
    ElevatedButton(
      onPressed: postController.createPost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
      child: const Text('POST'),
    ),
    ElevatedButton(
      onPressed: postController.updatePost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
      child: const Text('UPDATE'),
    ),
    ElevatedButton(
      onPressed: postController.deletePost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
      child: const Text('DELETE'),
    ),
  ],
),
],
),
),
);
}
}

```

Folder **services** isi **api_service.dart**

```

import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
  List<dynamic> posts = []; // Menyimpan data post yang diterima

  // Fungsi untuk GET data
  Future<void> fetchPosts() async {
    final response = await http.get(Uri.parse('$baseUrl/posts'));

```

```

    if (response.statusCode == 200) {
      posts = json.decode(response.body);
    } else {
      throw Exception('Failed to load posts');
    }
  }
}

// Fungsi untuk POST data
Future<void> createPost() async {
  final response = await http.post(
    Uri.parse('$baseUrl/posts'),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'title': 'Flutter Post',
      'body': 'Ini contoh POST.',
      'userId': 1,
    }),
  );

  if (response.statusCode == 201) {
    posts.add({
      'title': 'Flutter Post',
      'body': 'Ini contoh POST.',
      'id': posts.length + 1,
    });
  } else {
    throw Exception('Failed to create post');
  }
}

// Fungsi untuk UPDATE data
Future<void> updatePost() async {
  final response = await http.put(
    Uri.parse('$baseUrl/posts/1'),
    headers: {'Content-Type': 'application/json'},
    body: json.encode({
      'title': 'Updated Title',
      'body': 'Updated Body',
      'userId': 1,
    }),
  );

  if (response.statusCode == 200) {
    final updatedPost = posts.firstWhere((post) => post['id'] == 1, orElse: () => null);
    if (updatedPost != null) {

```

```

        updatedPost['title'] = 'Updated Title';
        updatedPost['body'] = 'Updated Body';
    }
    } else {
        throw Exception('Failed to update post');
    }
}

// Fungsi untuk DELETE data
Future<void> deletePost() async {
    final response = await http.delete(
        Uri.parse('$baseUrl/posts/1'),
    );

    if (response.statusCode == 200) {
        posts.removeWhere((post) => post['id'] == 1);
    } else {
        throw Exception('Failed to delete post');
    }
}
}
}

```

Main.dart

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan_14/screen/homepage_screen.dart';

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

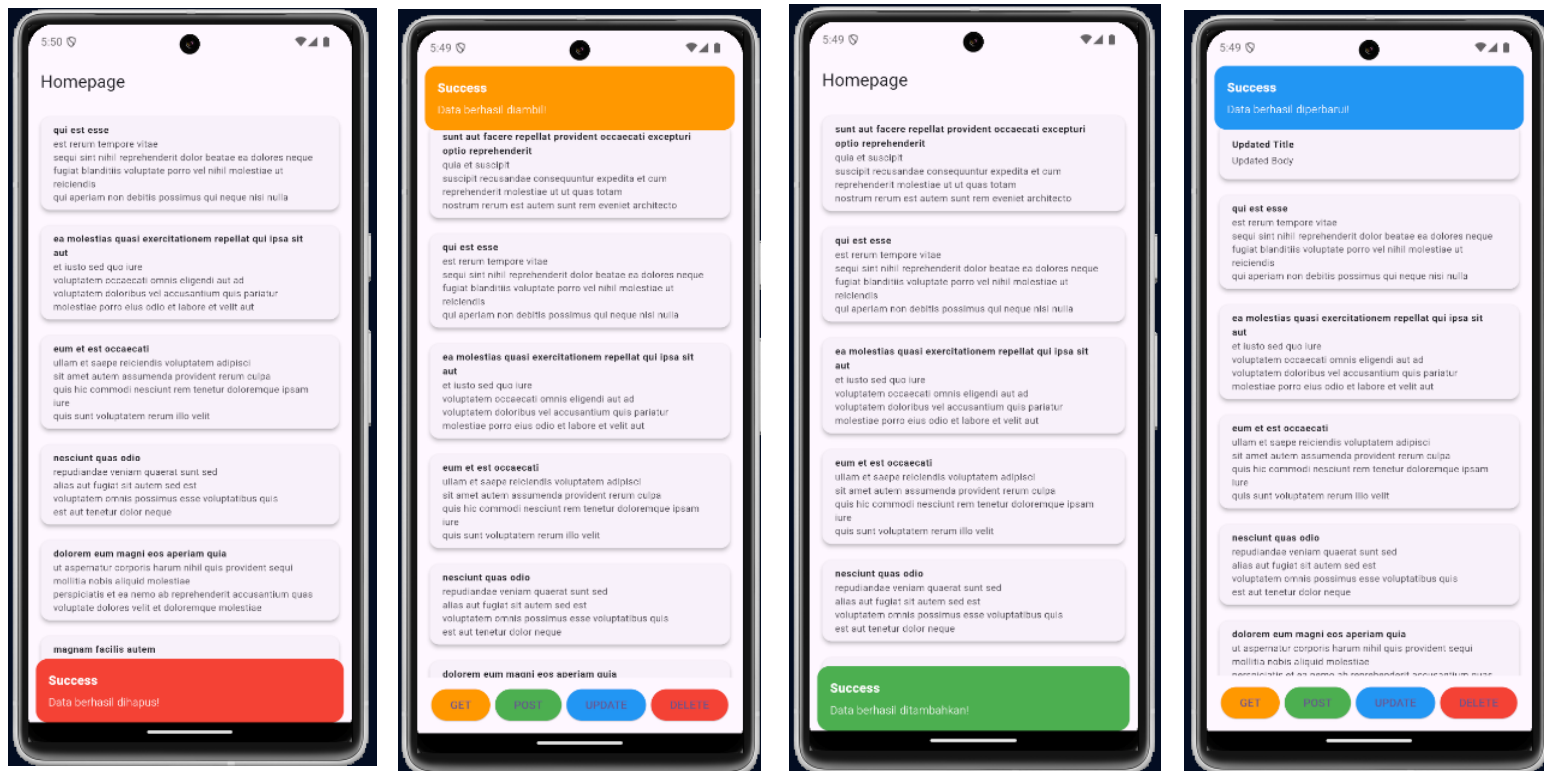
class MyApp extends StatelessWidget {
    const MyApp({super.key});

    @override
    Widget build(BuildContext context) {
        return GetMaterialApp( // Menggunakan GetMaterialApp
            title: 'Flutter GetX Demo',
            debugShowCheckedModeBanner: false,
            theme: ThemeData(
                colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
                useMaterial3: true,
            ),
            home: HomepageScreen(),
        );
    }
}

```

}

OutPutnya:



Deskripsi Program:

Program ini merupakan aplikasi berbasis Flutter yang menggunakan JSONPlaceholder API untuk mengelola data postingan dengan metode GET, POST, PUT, dan DELETE. Aplikasi ini memanfaatkan state management GetX untuk mempermudah pengelolaan data dan memberikan notifikasi interaktif melalui Snackbar setelah setiap operasi berhasil dilakukan. Antarmuka aplikasi terdiri dari daftar postingan yang ditampilkan menggunakan widget ListView serta tombol aksi untuk mengambil, menambahkan, memperbarui, dan menghapus data.

Setiap fungsi API diimplementasikan melalui ApiService untuk memisahkan logika bisnis dari antarmuka pengguna. Notifikasi Snackbar ditampilkan di bagian atas atau bawah layar dengan warna berbeda sesuai jenis aksi, seperti hijau untuk POST dan merah untuk DELETE. Struktur kode yang modular memudahkan pengembang dalam memahami, mengelola, dan memperluas aplikasi ini untuk kebutuhan proyek lainnya. Aplikasi ini cocok untuk belajar integrasi API, manajemen state dengan GetX, dan pembuatan antarmuka responsif di Flutter.