LAPORAN PRAKTIKUM
Modul 14
“API”

Disusun Oleh:
Muhammad Ralfi - 2211104054
Kelas SE-06-2

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

1. **Tujuan**
    a. Mahasiswa mampu memahami API dalam Flutter
    b. Mahasiswa mampu mengimplementasikan penggunaan API dalam Flutter
    c. Mengetahui kegunaan dan penerapan metode HTTP (GET, POST, PUT, DELETE) dalam pengelolaan data
    d. Membuat antarmuka yang menampilkan data dari API state
2. **Landasan Teori**

    REST API adalah protokol antarmuka yang memfasilitasi komunikasi antara aplikasi klien dan database melalui HTTP, memungkinkan operasi data seperti membaca, menambah, memperbarui, dan menghapus tanpa akses database langsung. Keunggulannya meliputi kemampuan interoperabilitas antar platform, pengiriman data yang efisien dalam format JSON atau XML, serta fitur keamanan melalui token autentikasi untuk membatasi akses pengguna.

    HTTP merupakan protokol dasar untuk pertukaran data antara klien dan server, dengan empat metode utama: GET untuk mengambil data, POST untuk mengirim data baru, PUT/PATCH untuk memperbarui data, dan DELETE untuk menghapus data. Setiap permintaan HTTP terdiri dari URL sebagai alamat sumber daya, metode operasi, headers untuk informasi tambahan, dan body untuk data yang dikirim. Sedangkan respons HTTP mencakup status code yang mengindikasikan hasil operasi, headers dari server, dan body yang berisi data dalam format tertentu.
3. **Guided**
    a. File main.dart

```
import 'package:api/screen/hompegae_screen.dart';
import 'package:flutter/material.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(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
   Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const HompegaeScreen(),
    );
  }
}
```

    b. File services/api_serices.dart

```
import 'dart:convert';
```

```dart
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'),
      body: json.encode({
        'title': 'Updated Title',
        'body': 'Updated Body',
        'userId': 1,
      }),
    );
    if (response.statusCode == 200) {
      final updatedPost = posts.firstWhere((post) => post['id'] == 1);
      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');
    }
  }

}
```

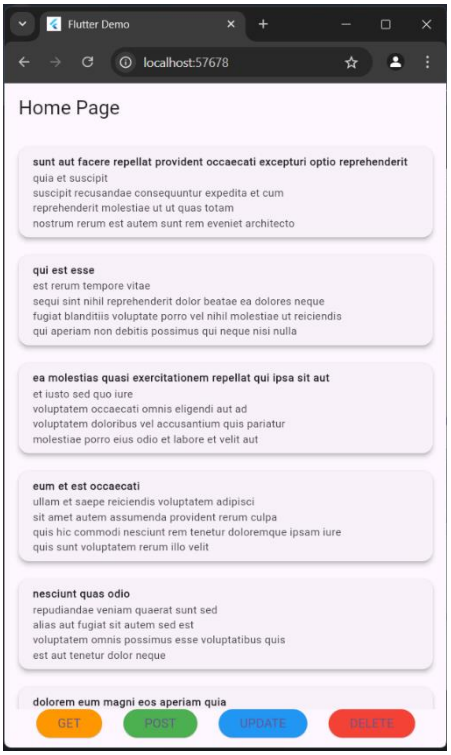
c. File screen/homepagescreen.dart

```
import 'package:api/services/api_servise.dart';
import 'package:flutter/material.dart';

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

  @override
  State<HompegaeScreen> createState() => _HompegaeScreenState();
}

class _HompegaeScreenState extends State<HompegaeScreen> {
  List<dynamic> _posts = []; // Menyimpan list posts
  bool _isLoading = false; // Untuk indikator loading
  final ApiService _apiService = ApiService(); // Instance ApiService
  // Fungsi untuk menampilkan SnackBar
  void _showSnackBar(String message) {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text(message)));
  }

  // Fungsi untuk memanggil API dan menangani operasi
  Future<void> _handleApiOperation(
      Future<void> operation, String successMessage) async {
    setState(() {
      _isLoading = true;
    });
    try {
      await operation; // Menjalankan operasi API
      setState(() {
        _posts = _apiService.posts;
      });
      _showSnackBar(successMessage);
    } catch (e) {
      _showSnackBar('Error: $e');
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

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

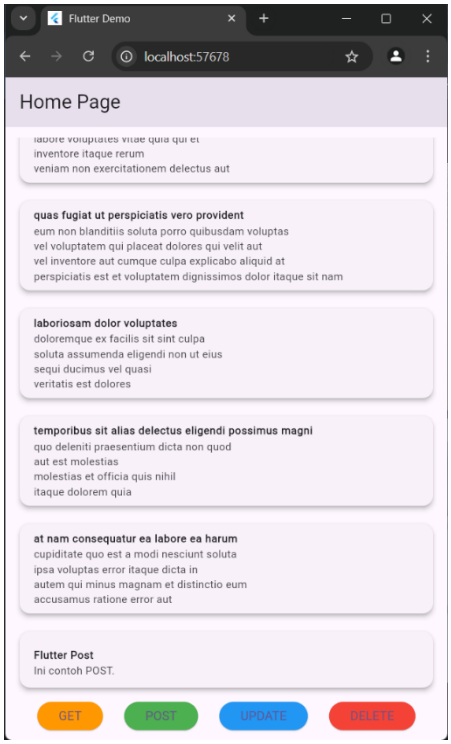
```
        ),
        body: Padding(
            padding: const EdgeInsets.all(12),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                _isLoading
                    ? const Center(child: CircularProgressIndicator())
                    : _posts.isEmpty
                        ? const Text(
                            "Tekan tombol GET untuk mengambil data",
                            style: TextStyle(fontSize: 12),
                          )
                        : 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'],
                                        style: const TextStyle(
                                            fontWeight:
FontWeight.bold,
                                            fontSize: 12),
                                      ),
                                      subtitle: Text(
                                        _posts[index]['body'],
                                        style: const
TextStyle(fontSize: 12),
                                      ),
                                    ),
                                  ),
                                );
                              },
                            ),
                          ),
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [
                    ElevatedButton(
                      onPressed: () => _handleApiOperation(
                          _apiService.fetchPosts(), 'Data berhasil
diambil!'),
                      style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.orange),
                      child: const Text('GET'),
                    ),
                    ElevatedButton(
                      onPressed: () => _handleApiOperation(
                          _apiService.createPost(),
                          'Data berhasil ditambahkan'),
                      style: ElevatedButton.styleFrom(
```

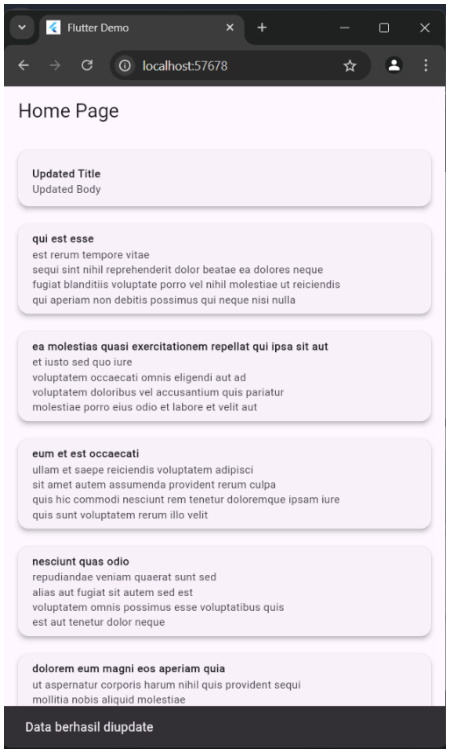
```
                        backgroundColor: Colors.green),
                    child: const Text('POST'),
                  ),
                  ElevatedButton(
                    onPressed: () => _handleApiOperation(
                        _apiService.updatePost(),
                        'Data berhasil diupdate'),
                    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'),
                  ),
                ],
              ),
            ],
          )));
  }
}
```

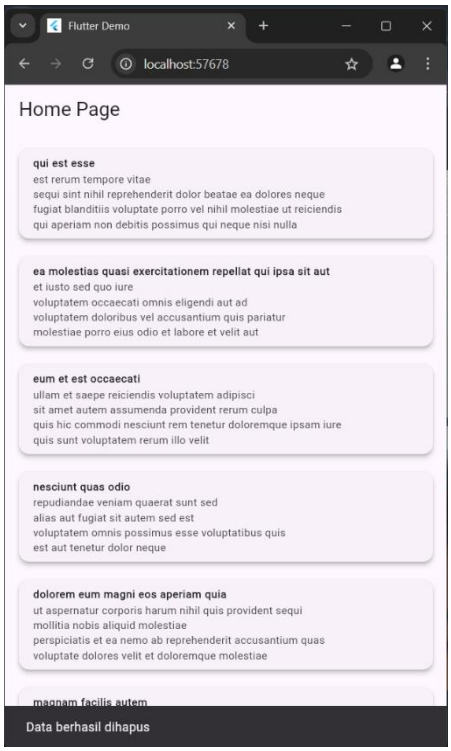
Output:



Telkom University Purwokerto







4. Unguided

Tugas Mandiri (Unguided)

Modifikasi tampilan Guided dari praktikum di atas:

a. **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.

b. **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.

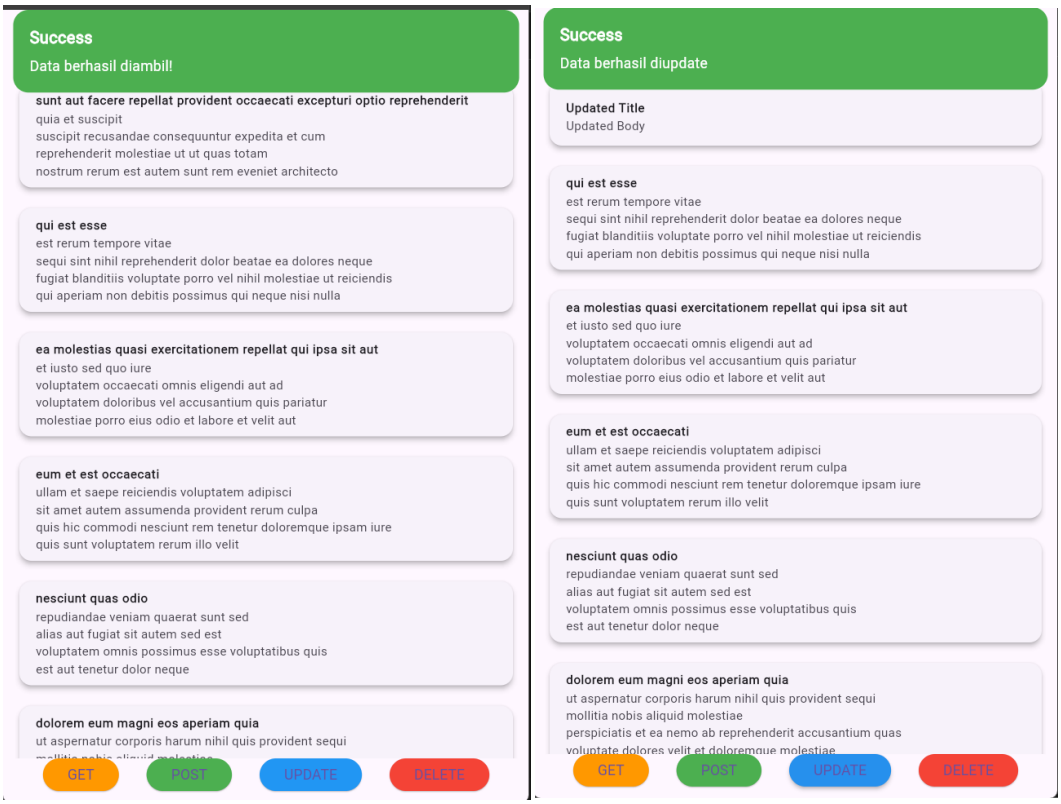
Buat File Controller

```
import 'package:api/services/api_servise.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class PostController extends GetxController {
  final ApiService _apiService = Get.find<ApiService>();
  RxList<dynamic> get posts => _apiService.posts;
  final RxBool isLoading = false.obs;

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

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

  Future<void> updatePost() async {
    _handleApiOperation(_apiService.updatePost(), 'Data berhasil
   diupdate');
  }

  Future<void> deletePost() async {
    _handleApiOperation(_apiService.deletePost(), 'Data berhasil
   dihapus');
  }

  Future<void> _handleApiOperation(Future<void> operation, String
   successMessage) async {
    isLoading.value = true;
    try {
      await operation;
```

```
        Get.snackbar('Success', successMessage, backgroundColor:
    Colors.green, colorText: Colors.white);
     } catch (e) {
       Get.snackbar('Error', 'Error: $e', backgroundColor: Colors.red,
    colorText: Colors.white);
     } finally {
       isLoading.value = false;
     }
   }
 }
```

Output:

Untuk output masih sama hanya saja penggunaan GetX jadi lebih rapi dalam pengkodean, dan adanya penambahan snack bar saat data diambil, ditambahkan, diupdate atau dihapus



5. Kesimpulan

REST API dan HTTP merupakan komponen fundamental dalam komunikasi data modern. REST API menyediakan antarmuka untuk interaksi aplikasi dengan database melalui HTTP, menawarkan fleksibilitas dan keamanan dalam pengelolaan data. HTTP, sebagai protokol komunikasi utama, mendukung berbagai metode seperti GET, POST, PUT/PATCH, dan DELETE untuk manipulasi data, dengan struktur request-response yang terstandarisasi untuk pertukaran informasi yang efektif antara klien dan server.