LAPORAN PRAKTIKUM
Modul 14
“Data Storage (API)”

Disusun Oleh:
Ganesha Rahman Gibran -2211104058
Kelas S1SE-06-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

Tujuan

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.

Landasan Teori

1. REST API

REST API (Representational State Transfer Application Programming Interface) adalah suatu antarmuka yang memungkinkan aplikasi klien untuk berinteraksi dengan database melalui protokol HTTP. REST API menyediakan mekanisme untuk membaca, menambahkan, memperbarui, dan menghapus data dari database tanpa harus langsung mengakses database. Hal ini membuat REST API menjadi pilihan populer dalam pengembangan aplikasi berbasis web dan mobile.

Kegunaan REST API:

1. Interoperabilitas: Memungkinkan berbagai aplikasi, baik web maupun mobile, untuk mengakses data yang sama secara konsisten.
2. Efisiensi: Data yang dikirimkan melalui REST API biasanya berformat ringan seperti JSON atau XML, sehingga lebih cepat diproses.
3. Keamanan: REST API mendukung autentikasi menggunakan token, sehingga akses ke data dapat dibatasi dan lebih aman.

2. HTTP (Hypertext Transfer Protocol)

HTTP adalah protokol komunikasi utama yang digunakan untuk mengirimkan data antara klien (misalnya browser atau aplikasi) dan server. REST API menggunakan metode HTTP untuk melakukan operasi pada data.

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 pada server.
2. Method: Operasi yang ingin dilakukan, seperti GET atau POST.
3. Headers: Informasi tambahan seperti format data (JSON) atau token autentikasi.
4. Body: Data yang dikirimkan (biasanya digunakan dalam POST/PUT).

Komponen HTTP Response:

1. Status Code: Kode yang menunjukkan hasil operasi, seperti 200 OK untuk berhasil atau 404 Not Found untuk resource tidak ditemukan.

2. Headers: Informasi tambahan dari server.
3. Body: Data yang dikembalikan server, biasanya dalam format JSON.

3. REST API dalam Flutter

Flutter adalah framework pengembangan aplikasi mobile yang mendukung integrasi REST API melalui paket HTTP. Dalam Flutter, REST API digunakan untuk mengelola data aplikasi secara dinamis. Operasi seperti GET, POST, PUT, dan DELETE dapat diimplementasikan menggunakan library HTTP, yang diatur dalam file service untuk memisahkan logika backend dari antarmuka pengguna (UI).

Alur Implementasi REST API:

1. Persiapan Proyek Flutter: Membuat proyek baru dan menambahkan dependensi HTTP pada file pubspec.yaml.
2. Struktur Folder: Membuat folder services untuk file API dan screens untuk UI.
3. Operasi HTTP:
   - GET: Mengambil data dari server.
   - POST: Menambahkan data baru ke server.
   - PUT: Memperbarui data yang ada di server.
   - DELETE: Menghapus data dari server.

4. Prinsip Pemisahan Logika API dan UI

Dalam pengembangan aplikasi, pemisahan antara logika API (service layer) dan UI (presentation layer) adalah praktik yang direkomendasikan. Hal ini memudahkan pengelolaan kode, memungkinkan pengembangan yang modular, serta meningkatkan efisiensi debugging dan pengujian.

5. Format Data JSON

JSON (JavaScript Object Notation) adalah format data ringan yang sering digunakan dalam komunikasi REST API. JSON mudah dibaca oleh manusia dan mudah diproses oleh mesin, sehingga menjadi standar untuk pertukaran data antara klien dan server.

Guided

Input :

- main.dart

```
import 'package:api/screens/homepage_screen.dart';
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
```

```
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: HomrpageScreen(),
    );
  }
}
```

- screens/homepage_screen.dart

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

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

  @override
  State<HomrpageScreen> createState() => _HomrpageScreenState();
}

class _HomrpageScreenState extends State<HomrpageScreen> {
  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: Text('REST API-Praktikum 14'),
          centerTitle: true,
        ),
        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),
          ),
          ),
          ),
          );
          },
          ),
          ),
          ElevatedButton(
onPressed: () => _handleApiOperation(
_apiService.fetchPosts(), 'Data berhasil diambil!'),
style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
child: const Text('GET'),
),
SizedBox(height: 20),
ElevatedButton(
onPressed: () => _handleApiOperation(
_apiService.createPost(), 'Data berhasil ditambahkan!'),
style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
child: const Text('POST'),
),
SizedBox(height: 20),
```

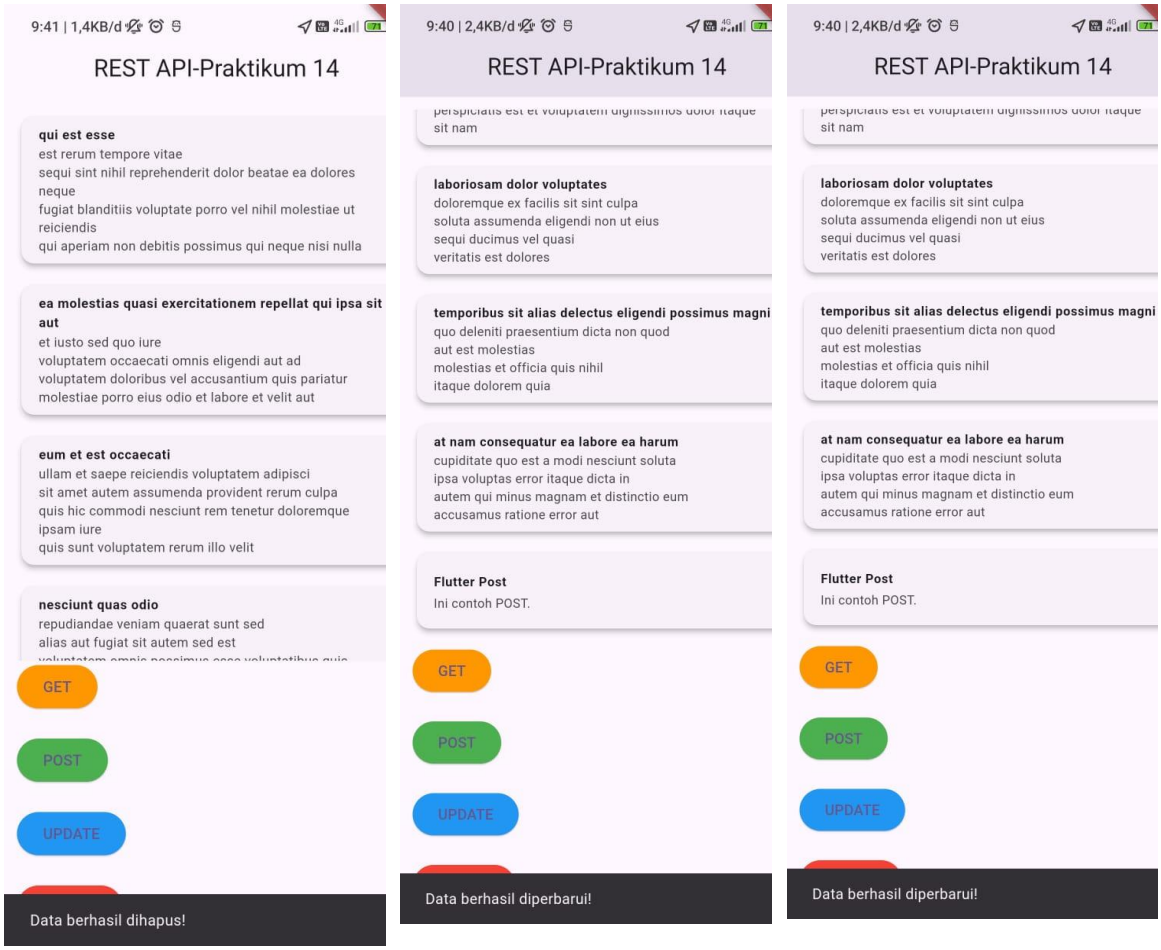
```
 ElevatedButton(
 onPressed: () => _handleApiOperation(
 _apiService.updatePost(), 'Data berhasil diperbarui!'),
 style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
 child: const Text('UPDATE'),
 ),
 SizedBox(height: 20),
 ElevatedButton(
 onPressed: () => _handleApiOperation(
 _apiService.deletePost(), 'Data berhasil dihapus!'),
 style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
 child: const Text('DELETE'),
 ),
         ],
         ),
       ),
     );
   }
}
```

- services/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'),
  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');
  }
  }

}
```

Output :



Unguided

1. **Modifikasi tampilan Guided dari praktikum di atas:**
   a. Gunakan State Management dengan GetX:
   b. Atur data menggunakan state management GetX agar lebih mudah dikelola.
   c. 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:**
   a. Tampilkan snackbar setelah setiap operasi berhasil, seperti menambah atau memperbarui data.
   b. Gunakan Get.snackbar agar pesan sukses muncul di layar dan mudah dipahami oleh pengguna.

Input

- Main.dart

```
import 'package:flutter/material.dart';
import 'package:unguided_api/screens/homepage_screen.dart';

void main() {
```

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: HomepageScreen(),
    );
  }
}
```

- services/api_service.dart

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

import '../services/api_services.dart';

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

  @override
  Widget build(BuildContext context) {
    final apiController = Get.put(ApiController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API - GetX'),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Obx(
              () => apiController.isLoading.value
                  ? const Center(child: CircularProgressIndicator())
                  : apiController.posts.isEmpty
                      ? const Text(
                          "Tekan tombol GET untuk mengambil data",
                          style: TextStyle(fontSize: 12),
                        )
                      : Expanded(
                          child: ListView.builder(
                            itemCount: apiController.posts.length,
                            itemBuilder: (context, index) {
```

```
                                return Padding(
                                  padding: const EdgeInsets.only(bottom: 12.0),
                                  child: Card(
                                    elevation: 4,
                                    child: ListTile(
                                      title: Text(
                                        apiController.posts[index]['title'],
                                        style: const TextStyle(
                                            fontWeight: FontWeight.bold,
                                            fontSize: 12),
                                      ),
                                      subtitle: Text(
                                        apiController.posts[index]['body'],
                                        style: const TextStyle(fontSize: 12),
                                      ),
                                    ),
                                  ),
                                );
                              },
                            ),
                          ),
                  ),
                  ElevatedButton(
                    onPressed: apiController.fetchPosts,
                    style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
                    child: const Text('GET'),
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: apiController.createPost,
                    style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
                    child: const Text('POST'),
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: apiController.updatePost,
                    style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
                    child: const Text('UPDATE'),
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: apiController.deletePost,
                    style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
                    child: const Text('DELETE'),
                  ),
                ],
              ),
            ),
          );
        }
}
```

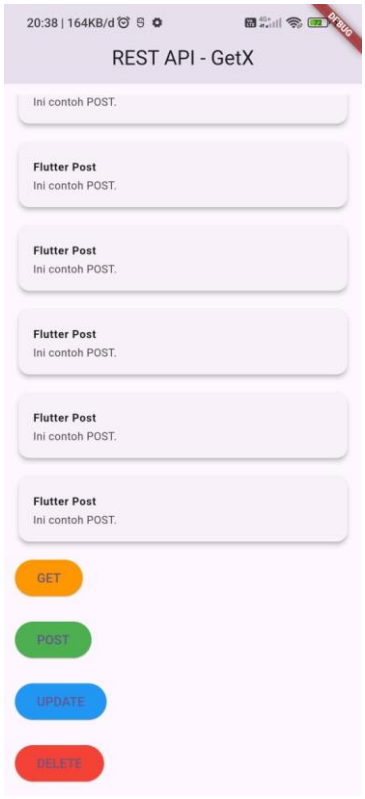
- screens/homepage_screen.dart

```
import 'package:get/get.dart';
import 'package:http/http.dart' as http;
```

```
import 'dart:convert';

class ApiController extends GetxController {
  final String baseUrl = "https://jsonplaceholder.typicode.com";

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

  // Fungsi untuk GET data
  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      final response = await http.get(Uri.parse('$baseUrl/posts'));
      if (response.statusCode == 200) {
        posts.value = json.decode(response.body);
        Get.snackbar('Sukses', 'Data berhasil diambil!');
      } else {
        throw Exception('Gagal mengambil data');
      }
    } catch (e) {
      Get.snackbar('Error', e.toString());
    } finally {
      isLoading.value = false;
    }
  }

  // Fungsi untuk POST data
  Future<void> createPost() async {
    try {
      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,
        });
        Get.snackbar('Sukses', 'Data berhasil ditambahkan!');
      } else {
        throw Exception('Gagal melakukan Post');
      }
    } catch (e) {
      Get.snackbar('Error', e.toString());
    }
  }

  // Fungsi untuk UPDATE data
  Future<void> updatePost() async {
```

```
      try {
        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 index = posts.indexWhere((post) => post['id'] == 1);
          if (index != -1) {
            posts[index]['title'] = 'Updated Title';
            posts[index]['body'] = 'Updated Body';
          }
          Get.snackbar('Sukses', 'Data berhasil diperbarui!');
        } else {
          throw Exception('Gagal melakukan Update data');
        }
      } catch (e) {
        Get.snackbar('Error', e.toString());
      }
    }

    // Fungsi untuk DELETE data
    Future<void> deletePost() async {
      try {
        final response = await http.delete(Uri.parse('$baseUrl/posts/1'));
        if (response.statusCode == 200) {
          posts.removeWhere((post) => post['id'] == 1);
          Get.snackbar('Sukses', 'Data berhasil dihapus!');
        } else {
          throw Exception('Gagal menghapus data');
        }
      } catch (e) {
        Get.snackbar('Error', e.toString());
      }
    }
}
```

Output


20:38 | 164KB/d
REST API - GetX
Ini contoh POST.
Flutter Post
Ini contoh POST.
Flutter Post
Ini contoh POST.
Flutter Post
Ini contoh POST.
Flutter Post
Ini contoh POST.
Flutter Post
Ini contoh POST.
GET
POST
UPDATE
DELETE


20:38 | 159KB/d
REST API - GetX
Updated Title
Updated Body
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque
ipsam iure
quis sunt voluptatem rerum illo velit
GET
POST
UPDATE
DELETE


20:39 | 190KB/d
REST API - GetX
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque
ipsam iure
quis sunt voluptatem rerum illo velit
nesciunt quas odio
repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
GET
POST
UPDATE
DELETE


20:37 | 149KB/d
REST API - GetX
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
GET
POST
UPDATE
DELETE

Deskripsi kode

Implementasi aplikasi Flutter dengan pengelolaan REST API menggunakan GetX sebagai state management. Aplikasi terdiri dari dua file utama: api_controller.dart, yang berisi logika untuk operasi CRUD (Create, Read, Update, Delete) terhadap data dari API eksternal, dan homepage_screen.dart, yang merupakan antarmuka pengguna utama untuk menampilkan data dan tombol-tombol operasi API. Data yang diambil dari API dikelola secara reaktif menggunakan properti observabel (obs), memungkinkan pembaruan UI secara otomatis saat data berubah. Pengguna dapat melakukan GET untuk mengambil data, POST untuk menambahkan data baru, PUT untuk memperbarui data, dan DELETE untuk menghapus data. Notifikasi aksi seperti keberhasilan atau kegagalan ditampilkan menggunakan Get.snackbar. Antarmuka aplikasi dirancang dengan elemen visual seperti daftar data (list), tombol aksi, dan indikator pemuatan (loading indicator).

Kesimpulan

REST API merupakan antarmuka penting dalam pengembangan aplikasi modern, khususnya untuk menghubungkan aplikasi klien dengan server menggunakan protokol HTTP. Dengan metode HTTP seperti GET, POST, PUT, dan DELETE, data dapat dikelola secara efisien, mulai dari pengambilan, penambahan, pembaruan, hingga penghapusan data. Implementasi REST API pada Flutter memanfaatkan pustaka http untuk menangani permintaan dan respons HTTP, serta melibatkan pembuatan antarmuka pengguna yang responsif. Praktikum ini memperlihatkan bahwa REST API tidak hanya meningkatkan interoperabilitas antar sistem tetapi juga mendukung efisiensi, keamanan, dan pengelolaan data secara terstruktur dalam pengembangan aplikasi berbasis Flutter.