LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE BAGIAN 3

Disusun Oleh :

Zivana Afra Yulianto / 2211104039

SE-06-02

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

BAB I
PENDAHULUAN

A. DASAR TEORI

REST API memungkinkan aplikasi berkomunikasi dengan server melalui HTTP (GET, POST, PUT, DELETE) untuk mengelola data. Flutter mendukung integrasi API dan GetX digunakan untuk state management yang efisien, sedangkan **Snackbar** memberikan notifikasi singkat kepada pengguna.

B. MAKSUD DAN TUJUAN

- Mempelajari implementasi REST API dengan Flutter.
- Membuat aplikasi dengan fitur CRUD (Create, Read, Update, Delete).
- Menggunakan **GetX** untuk manajemen data dan **Snackbar** sebagai feedback pengguna.

BAB II
IMPLEMENTASI

A. GUIDED

Code main.dart

```
import 'package:flutter/material.dart';
import 'package:praktikum/screen/home_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',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: HomeScreen(),
    );
  }
}
```

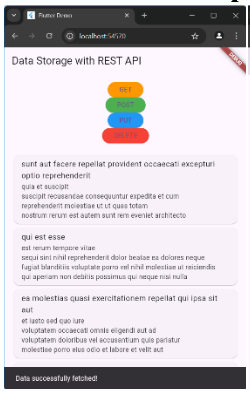
Code service/api_service.dart :

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

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

  // Fetch posts (GET)
  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');
    }
  }

  // Create a new post (POST)
  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': 'This is a sample post.',
        'userId': 1,
      }),
    );

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

  // Update a post (PUT)
  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');
    }
  }

  // Delete a post (DELETE)
  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');
    }
  }
}
```

Code screen/home_screen.dart :

```
import 'package:flutter/material.dart';
import 'package:praktikum/service/api_service.dart';

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  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;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Data Storage with REST API'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.fetchPosts(), 'Data successfully fetched!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
              child: const Text('GET'),
            ),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.createPost(), 'Post successfully created!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
              child: const Text('POST'),
            ),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.updatePost(), 'Post successfully updated!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('PUT'),
            ),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.deletePost(), 'Post successfully deleted!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
              child: const Text('DELETE'),
            ),
            const SizedBox(height: 20),
            _isLoading
                ? const CircularProgressIndicator()
                : _posts.isEmpty
                    ? const Text("Press the GET button to load data.")
                    : Expanded(
                        child: ListView.builder(
                          itemCount: _posts.length,
                          itemBuilder: (context, index) {
                            return Card(
                              child: ListTile(
                                title: Text(_posts[index]['title']),
                                subtitle: Text(_posts[index]['body']),
                              ),
                            );
                          },
                        ),
                      ),
          ],
        ),
      ),
    );
  }
}
```

Screenshoot Output :



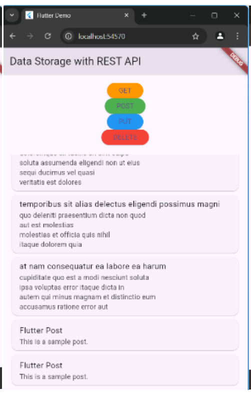
Data Storage with REST API

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

Data successfully fetched!



Data Storage with REST API

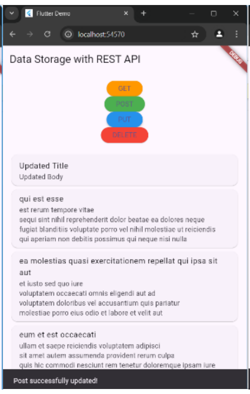
soluta assumenda eligendi non ut eius
sequi ducimus vel quasi
veritatis est dolores

temporibus sit alias delectus eligendi possimus magni
quo deleniti praesentium dicta non quod
aut est molestias
molestias et officia quis nihil
itaque dolorem quia

at nam consequatur ea labore ea harum
cupiditate quo est a modi nesciunt soluta
ipsa voluptas error itaque dicta in
autem qui minus magnam et distinctio eum
accusamus ratione error aut

Flutter Post
This is a sample post.

Flutter Post
This is a sample post.



Data Storage with REST API

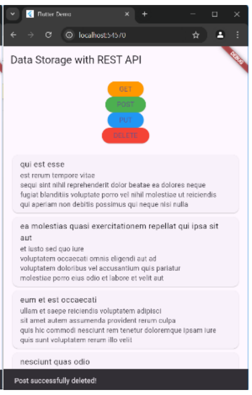
Updated Title
Updated Body

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

Post successfully updated!



Data Storage with REST API

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

Post successfully deleted!

Get **Post** **Put** **Delete**

Deskipsi :

1. Service GET
   Mengambil data dari API (`/posts`) menggunakan HTTP GET, hasilnya disimpan dalam list `posts`.
2. Tampilan GET
   Menampilkan data dari list `posts` dalam bentuk daftar menggunakan `ListView.builder`. Terdapat indikator loading dan pesan jika data kosong.
3. Service POST
   Menambahkan data baru ke server menggunakan HTTP POST. Data berupa JSON berisi `title`, `body`, dan `userId`.
4. Tampilan POST
   Membuat tombol untuk menambahkan data ke server. Data baru langsung ditampilkan di daftar.
5. Service PUT
   Memperbarui data di server untuk ID tertentu (ID = 1) menggunakan HTTP PUT. Data yang diperbarui adalah `title` dan `body`.
6. Tampilan PUT
   Membuat tombol untuk memperbarui data. Setelah diperbarui, daftar data diperbarui otomatis.
7. Service DELETE
   Menghapus data dari server untuk ID tertentu (ID = 1) menggunakan HTTP DELETE.
8. Tampilan DELETE
   Membuat tombol untuk menghapus data. Daftar data diperbarui setelah penghapusan.

B. UNGUIDED

Code main.dart :

```
import 'package:flutter/material.dart';
import 'package:praktikum/screens/home_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',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: HomeScreen(),
    );
  }
}
```

Code controllers/notes_controller.dart :

```
import 'package:get/get.dart';
import '../service/api_service.dart';

class PostController extends GetxController {
  var posts = [].obs;
  var isLoading = false.obs;

  final ApiService _apiService = ApiService();

  void fetchPosts() async {
    try {
      isLoading.value = true;
      await _apiService.fetchPosts();
      posts.value = _apiService.posts;
    } finally {
      isLoading.value = false;
    }
  }

  void createPost() async {
    try {
      isLoading.value = true;
      await _apiService.createPost();
      fetchPosts(); // Refresh data
    } finally {
      isLoading.value = false;
    }
  }

  void updatePost() async {
    try {
      isLoading.value = true;
      await _apiService.updatePost();
      fetchPosts();
    } finally {
      isLoading.value = false;
    }
  }
  void deletePost() async {
    try {
      isLoading.value = true;
      await _apiService.deletePost();
      fetchPosts();
    } finally {
      isLoading.value = false;
    }
  }
}
```

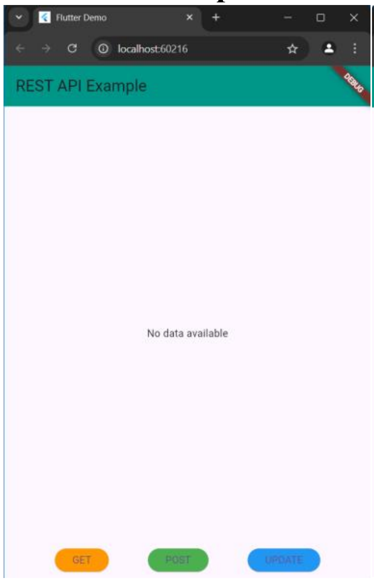
Code service/api_service.dart :

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

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

  // Fetch posts (GET)
  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');
    }
  }

  // Create a new post (POST)
  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': 'This is a sample post.',
        'userId': 1,
      }),
    );

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

  // Update a post (PUT)
  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');
    }
  }
  // Delete a post (DELETE)
  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');
    }
  }
}
```

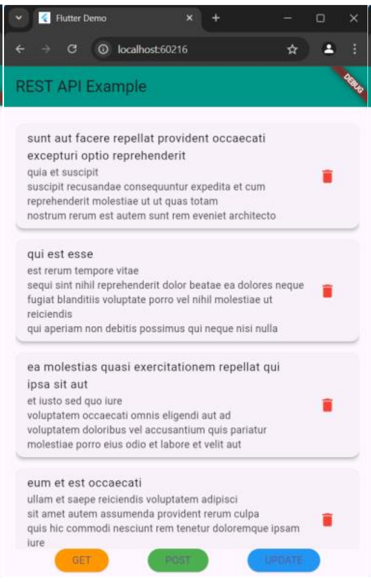
Code screen/home_screen.dart :

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API Example'),
        backgroundColor: Colors.teal,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            Expanded(
              child: Obx(() {
                if (postController.isLoading.value) {
                  return const Center(child: CircularProgressIndicator());
                }
                if (postController.posts.isEmpty) {
                  return const Center(child: Text('No data available'));
                }
                return ListView.builder(
                  itemCount: postController.posts.length,
                  itemBuilder: (context, index) {
                    final post = postController.posts[index];
                    return Card(
                      margin: const EdgeInsets.symmetric(vertical: 8.0),
                      elevation: 4,
                      child: ListTile(
                        title: Text(post['title']),
                        subtitle: Text(post['body']),
                        trailing: IconButton(
                          icon: const Icon(Icons.delete, color: Colors.red),
                          onPressed: () => postController.deletePost(),
                        ),
                      ),
                    );
                  },
                );
              }),
            ),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                ElevatedButton(
                  onPressed: () => postController.fetchPosts(),
                  child: const Text('GET'),
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
                ),
                ElevatedButton(
                  onPressed: () => postController.createPost(),
                  child: const Text('POST'),
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
                ),
                ElevatedButton(
                  onPressed: () => postController.updatePost(),
                  child: const Text('UPDATE'),
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

Sreenshoot Output :



No data available

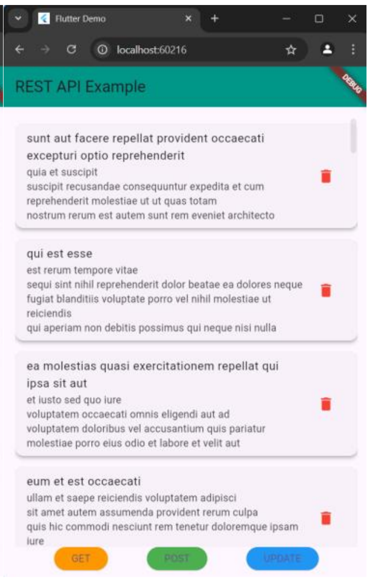


sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

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
quis hic commodi nesciunt rem tenetur doloremque ipsam iure



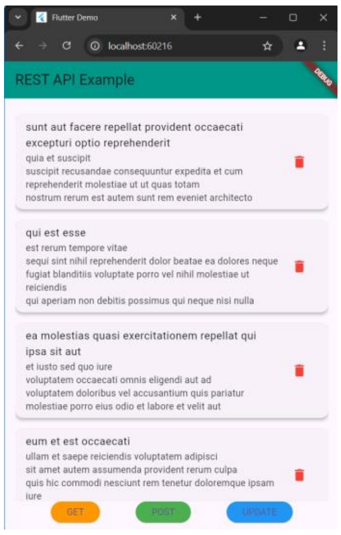
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

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
quis hic commodi nesciunt rem tenetur doloremque ipsam iure

Awal **GET** **POST**

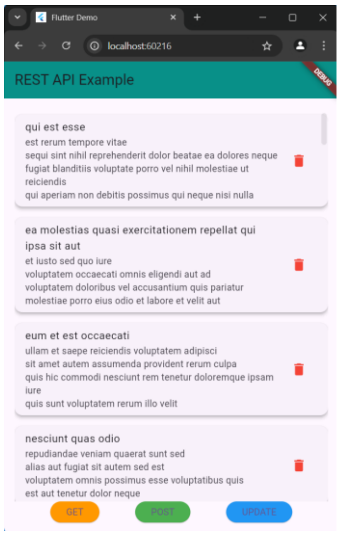


sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

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
quis hic commodi nesciunt rem tenetur doloremque ipsam iure



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
voluptatem omnis possimus esse voluptatibus quis
est aut tenetur dolor neque

UPDATE **DELETE**

Deskripsi Program

Berikut deskripsi singkat untuk bagian Unguided:

1. State Management dengan GetX
   1. Data diatur menggunakan GetX untuk pengelolaan state yang lebih efisien.
   2. `GetBuilder` atau `Obx` digunakan untuk memantau perubahan data secara real-time di UI tanpa perlu setState.
2. Penggunaan Snackbar dengan GetX
   1. Menambahkan notifikasi berbentuk Snackbar menggunakan `Get.snackbar` setelah setiap operasi (GET, POST, PUT, DELETE) berhasil dijalankan.
   2. Snackbar memberikan feedback langsung kepada pengguna tentang status keberhasilan atau kegagalan operasi.
3. Modifikasi UI
   1. Tampilan antarmuka diperbaiki agar lebih menarik dan user-friendly.
   2. Elemen-elemen UI seperti tombol dan daftar data diberi warna dan gaya modern untuk meningkatkan pengalaman pengguna.

BAB III
KESIMPULAN

KESIMPULAN

REST API dan Flutter memungkinkan pengelolaan data yang dinamis. **GetX** mempermudah sinkronisasi data dan UI, sementara **Snackbar** meningkatkan pengalaman pengguna dengan memberikan notifikasi atas keberhasilan operasi CRUD.