

GUIDED
PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK
DATA STORAGE BAGIAN 3 (API)



Disusun Oleh :
Maria Nathasya Desfera Pangestu / 2211104008
SE0601

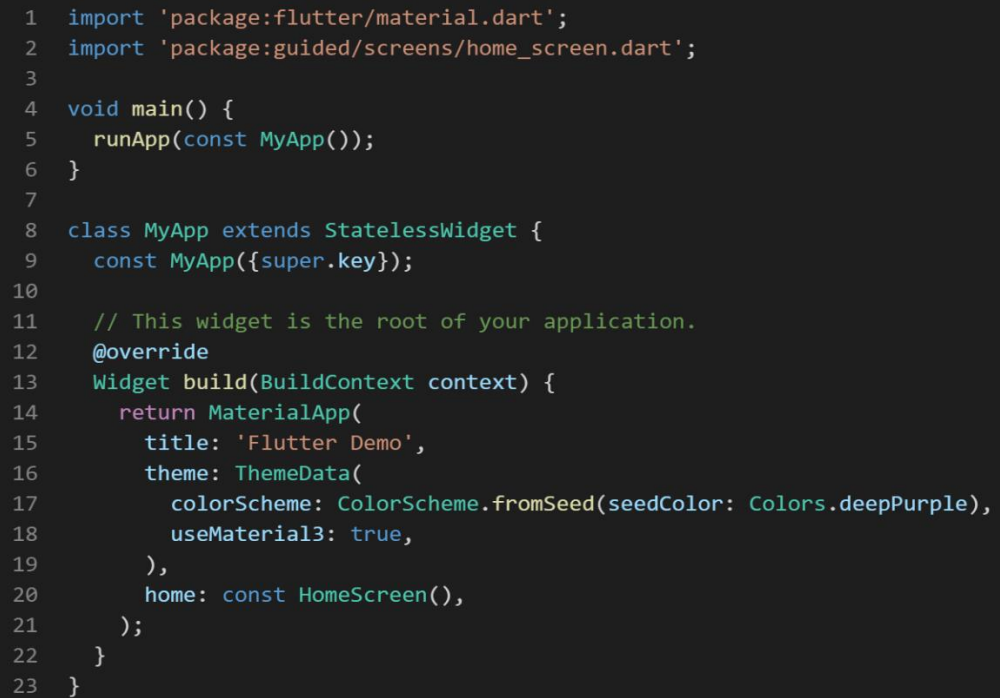
Asisten Praktikum :
Muhammad Faza Zulian Gesit Al Barru
Aisyah Hasna Aulia

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

GUIDED

1. Source code main.dart



```
1  import 'package:flutter/material.dart';
2  import 'package:guided/screens/home_screen.dart';
3
4  void main() {
5    runApp(const MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    const MyApp({super.key});
10
11    // This widget is the root of your application.
12    @override
13    Widget build(BuildContext context) {
14      return MaterialApp(
15        title: 'Flutter Demo',
16        theme: ThemeData(
17          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
18          useMaterial3: true,
19        ),
20        home: const HomeScreen(),
21      );
22    }
23  }
```

Source code home_screen



```
1  import 'package:flutter/material.dart';
2  import 'package:guided/services/api_service.dart';
3
4  class HomeScreen extends StatefulWidget {
5    const HomeScreen({super.key});
6
7    @override
8    State<HomeScreen> createState() => _HomeScreenState();
9  }
10
11 class _HomeScreenState extends State<HomeScreen> {
12   List<dynamic> _posts = []; // Menyimpan list posts
13   bool _isLoading = false; // Untuk indikator loading
14   final ApiService _apiService = ApiService(); // Instance ApiService
15
16   // Fungsi untuk menampilkan Snackbar
17   void _showSnackBar(String message) {
18     ScaffoldMessenger.of(context)
19       .showSnackBar(SnackBar(content: Text(message)));
20   }
21
22   // Fungsi untuk memanggil API dan menangani operasi
23   Future<void> _handleApiOperation(
24     Future<void> operation, String successMessage) async {
25     setState(() {
26       _isLoading = true;
27     });
28     try {
29       await operation; // Menjalankan operasi API
30       setState(() {
31         _posts = _apiService.posts;
32       });
33       _showSnackBar(successMessage);
34     } catch (e) {
35       _showSnackBar('Error: $e');
36     } finally {
37       setState(() {
38         _isLoading = false;
39       });
40     }
41   }
42 }
```

```

43  @override
44  Widget build(BuildContext context) {
45    return Scaffold(
46      appBar: AppBar(
47        title: const Text('HTTP Request Example'),
48        centerTitle: true,
49        backgroundColor: const Color.fromARGB(255, 222, 135, 240),
50      ),
51      body: Padding(
52        padding: const EdgeInsets.all(12.0),
53        child: Column(
54          crossAxisAlignment: CrossAxisAlignment.start,
55          children: [
56            if (_isLoading)
57              const Center(child: CircularProgressIndicator())
58            else if (_posts.isEmpty)
59              const Text(
60                "Tekan tombol GET untuk mengambil data",
61                style: TextStyle(fontSize: 16),
62              )
63            else
64              Expanded(
65                child: ListView.builder(
66                  itemCount: _posts.length,
67                  itemBuilder: (context, index) {
68                    return Padding(
69                      padding: const EdgeInsets.only(bottom: 12.0),
70                      child: Card(
71                        elevation: 4,
72                        child: ListTile(
73                          title: Text(
74                            _posts[index]['title'],
75                            style: const TextStyle(
76                              fontWeight: FontWeight.bold, fontSize: 16),
77                          ),
78                          subtitle: Text(
79                            _posts[index]['body'],
80                            style: const TextStyle(fontSize: 14),

```

```

81         ),
82     ),
83 ),
84 );
85 },
86 ),
87 ),
88 const SizedBox(height: 20),
89 ElevatedButton(
90   onPressed: () => _handleApiOperation(
91     _apiService.fetchPosts(), 'Data berhasil diambil!'),
92   style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
93   child: const Text('GET'),
94 ),
95 const SizedBox(height: 10),
96 ElevatedButton(
97   onPressed: () => _handleApiOperation(
98     _apiService.createPost(), 'Data berhasil ditambahkan!'),
99   style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
100   child: const Text('POST'),
101 ),
102 const SizedBox(height: 10),
103 ElevatedButton(
104   onPressed: () => _handleApiOperation(
105     _apiService.updatePost(1), 'Data berhasil diperbarui!'),
106   style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
107   child: const Text('UPDATE'),
108 ),
109 const SizedBox(height: 10),
110 ElevatedButton(
111   onPressed: () => _handleApiOperation(
112     _apiService.deletePost(1), 'Data berhasil dihapus!'),
113   style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
114   child: const Text('DELETE'),
115 ),
116 ],
117 ),
118 ),
119 );
120 }
121 }
122

```

Source code api_service

```
1 import 'dart:convert';
2 import 'package:http/http.dart' as http;
3
4 class ApiService {
5   final String baseUrl = "https://jsonplaceholder.typicode.com";
6   List<dynamic> posts = []; // Menyimpan data post yang diterima
7
8   // Fungsi untuk GET data
9   Future<void> fetchPosts() async {
10     try {
11       final response = await http.get(Uri.parse('$baseUrl/posts'));
12       if (response.statusCode == 200) {
13         posts = json.decode(response.body) as List<dynamic>;
14       } else {
15         throw Exception(
16           'Failed to load posts. Status code: ${response.statusCode}');
17       }
18     } catch (e) {
19       throw Exception('Failed to fetch posts. Error: $e');
20     }
21   }
22
23   // Fungsi untuk POST data
24   Future<void> createPost() async {
25     try {
26       final response = await http.post(
27         Uri.parse('$baseUrl/posts'),
28         headers: {'Content-Type': 'application/json'},
29         body: json.encode({
30           'title': 'Flutter Post',
31           'body': 'Ini contoh POST.',
32           'userId': 1,
33         })),
34     );
```

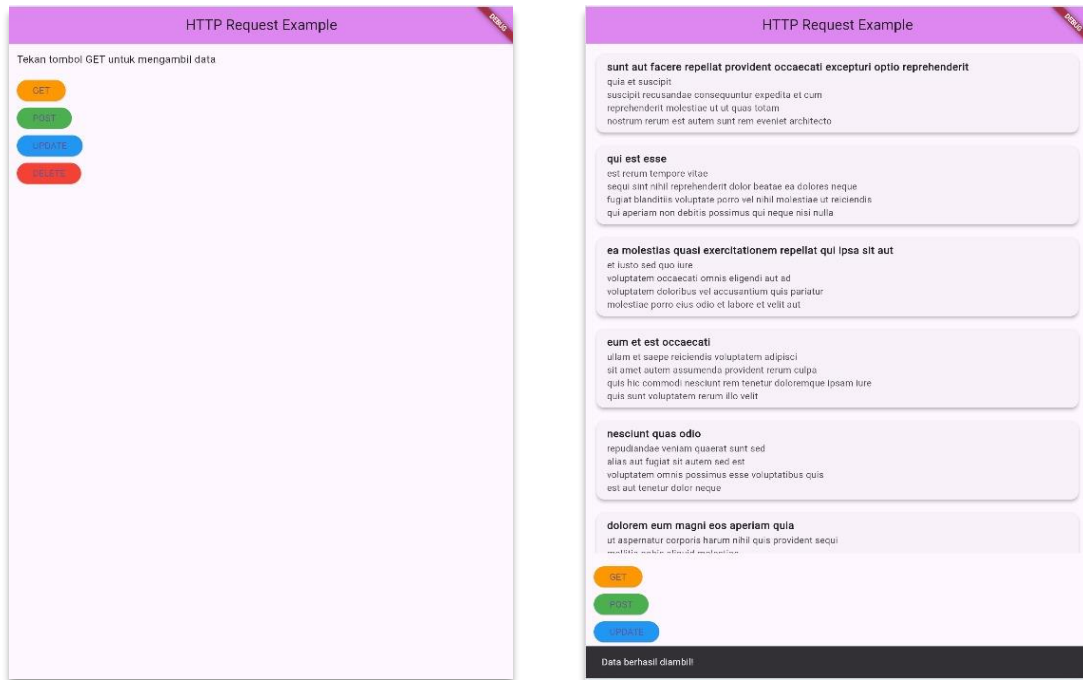
```

35     if (response.statusCode == 201) {
36         final newPost = json.decode(response.body);
37         posts.add(newPost); // Menambahkan data baru ke daftar posts
38     } else {
39         throw Exception(
40             'Failed to create post. Status code: ${response.statusCode}');
41     }
42 } catch (e) {
43     throw Exception('Failed to create post. Error: $e');
44 }
45 }
46
47 // Fungsi untuk UPDATE data
48 Future<void> updatePost(int postId) async {
49     try {
50         final response = await http.put(
51             Uri.parse('$baseUrl/posts/$postId'),
52             headers: {'Content-Type': 'application/json'},
53             body: json.encode({
54                 'title': 'Updated Title',
55                 'body': 'Updated Body',
56                 'userId': 1,
57             })),
58         );
59         if (response.statusCode == 200) {
60             final updatedPostIndex =
61                 posts.indexWhere((post) => post['id'] == postId);
62             if (updatedPostIndex != -1) {
63                 posts[updatedPostIndex] = {
64                     'id': postId,
65                     'title': 'Updated Title',
66                     'body': 'Updated Body',
67                     'userId': 1,
68                 };
69             } else {
70                 throw Exception('Post with id $postId not found in local data');
71             }
72         } else {
73             throw Exception(
74                 'Failed to update post. Status code: ${response.statusCode}');
75         }
76     } catch (e) {
77         throw Exception('Failed to update post. Error: $e');
78     }
79 }
80
81 // Fungsi untuk DELETE data
82 Future<void> deletePost(int postId) async {
83     try {
84         final response = await http.delete(Uri.parse('$baseUrl/posts/$postId'));
85         if (response.statusCode == 200) {
86             posts.removeWhere((post) => post['id'] == postId);
87         } else {
88             throw Exception(
89                 'Failed to delete post. Status code: ${response.statusCode}');
90         }
91     } catch (e) {
92         throw Exception('Failed to delete post. Error: $e');
93     }
94 }
95 }

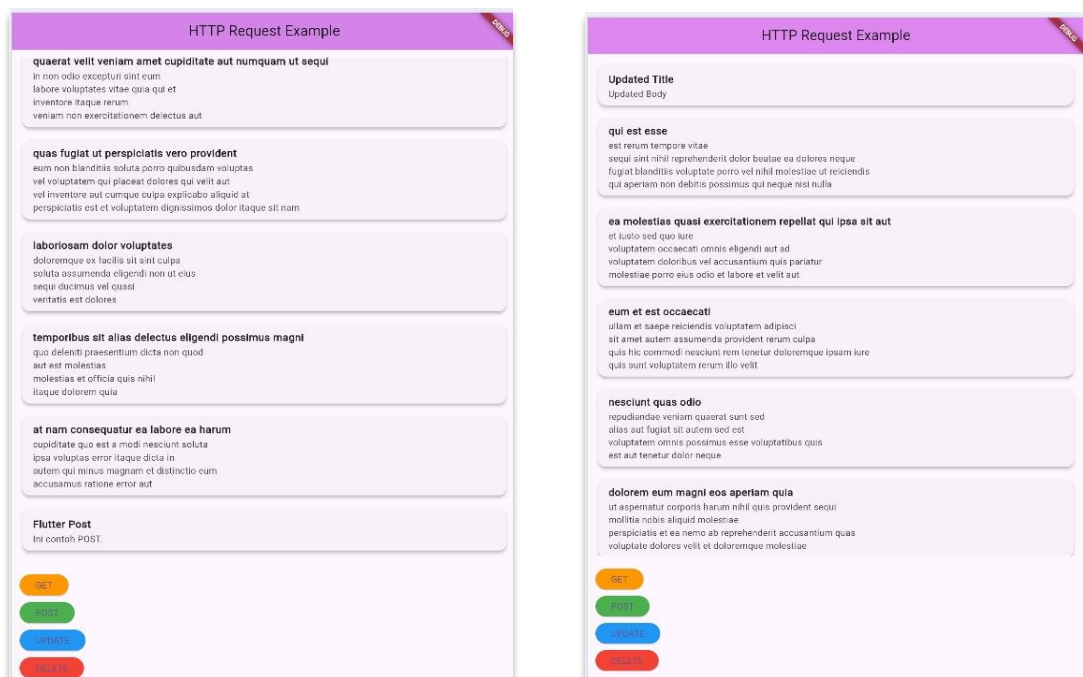
```

2. Screenshot output

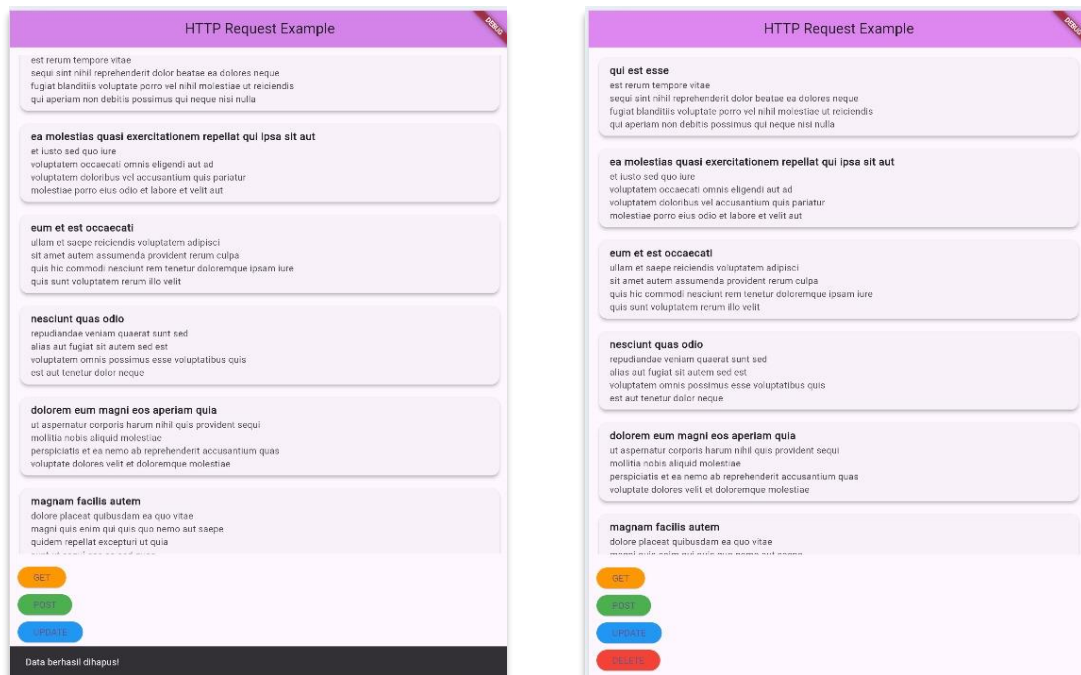
Tampilan awal saat aplikasi di run dan saat mengklik tombol get



Tampilan saat mengklik tombol post dan saat mengklik tombol update



Tampilan saat mengklik tombol delete dan saat data sudah dihapus



3. Penjelasan

Aplikasi tersebut menggunakan operasi HTTP (GET, POST, UPDATE, DELETE) untuk berinteraksi dengan API menggunakan kelas ApiService. Menggunakan widget StatefulWidget yang memungkinkan pengelolaan state secara dinamis. Di kelas _HomeScreenState terdapat _posts untuk menyimpan data dari API, _isLoading untuk indikator loading saat operasi berlangsung, dan _apiService sebagai instance dari kelas ApiService yang mengelola panggilan API. Lalu ada _showSnackBar untuk menampilkan pesan kepada pengguna melalui komponen SnackBar. Kemudian ada _handleApiOperation untuk menangani setiap operasi API, termasuk menampilkan indikator loading, memanggil operasi yang diteruskan, memperbarui state dengan data terbaru, serta menampilkan pesan sukses atau error melalui SnackBar. Aplikasi ini memakai widget Scaffold dengan AppBar berjudul "HTTP Request Example". Pada bagian body data ditampilkan dalam bentuk Card berisi judul dan isi dari properti title dan body pada data API. Tombol untuk melakukan operasi API:

- Tombol GET untuk memanggil metode fetchPosts gunanya mengambil data.
- Tombol POST untuk memanggil metode createPost fungsinya menambahkan data baru.
- Tombol UPDATE memanggil metode updatePost gunanya memperbarui data tertentu.
- Tombol DELETE untuk memanggil metode deletePost fungsinya menghapus data tertentu.