

GUIDED & UNGUIDED
PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK
MODUL XIV
DATA STORAGE BAGIAN 3 (API)



Disusun Oleh :
Maria Nathasya Desfera Pangestu / 2211104008
SE0601

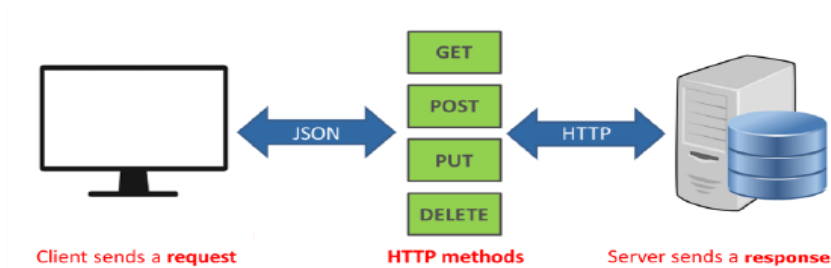
Asisten Praktikum :
Muhammad Faza Zulian Gesit Al Barru
Aisyah Hasna Aulia

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

GUIDED

Data Storage dengan Rest API



A. 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 :

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

B. 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 pada Flutter:

1. Persiapan Proyek Flutter
 - a. Pertama kita bisa membuat proyek Flutter baru.
 - b. Lalu menambahkan dependensi http ke file pubspec.yaml seperti gambar dibawah, lalu jalankan perintah flutter pub get pada terminal untuk menginstal dependensi.

```
dependencies:  
  flutter:  
    sdk: flutter  
  
  # The following adds the Cupertino Icons font to your application.  
  # Use with the CupertinoIcons class for iOS style icons.  
  cupertino_icons: ^1.0.8  
  http: ^1.2.2
```

2. Membuat Folder Struktur
Buat folder services untuk file API dan screens untuk file UI di dalam folder lib.

3. Implementasi HTTP GET

Kita akan menggunakan API dari <https://jsonplaceholder.typicode.com/>

- a. Membuat Service GET

Buat file api_service.dart di dalam folder services:

```
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   }  
}
```

- b. Membuat tampilan UI untuk GET

Buat file home_screen.dart di dalam folder screens:

Fungsi untuk memanggil file api service

```

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    }

```

Menampilkan response API

```

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         ) // Text
63     ],
64     else
65     Expanded(
66       child: ListView.builder(
67         itemCount: _posts.length,
68         itemBuilder: (context, index) {
69           return Padding(
70             padding: const EdgeInsets.only(bottom: 12.0),
71             child: Card(
72               elevation: 4,
73               child: ListTile(
74                 title: Text(
75                   _posts[index]['title'],
76                   style: const TextStyle(
77                     fontWeight: FontWeight.bold, fontSize: 16), // TextStyle
78                 ), // Text
79                 subtitle: Text(
80                   _posts[index]['body'],
81                   style: const TextStyle(fontSize: 14),
82                 ), // Text
83               ), // ListTile
84             ), // Card
85           ); // Padding
86         }, // ListView.builder
87       ), // Expanded

```

Tambahkan tombol untuk GET di home_screen.dart:

```
89         ElevatedButton(  
90             onPressed: () => _handleApiOperation(  
91                 _apiService.fetchPosts(), 'Data berhasil diambil!'),  
92             style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),  
93             child: const Text('GET'),  
94         ), // ElevatedButton
```

4. Implementasi HTTP POST

a. Membuat Service POST

Tambahkan metode berikut ke api_service.dart:

```
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 }
```

b. Membuat tampilan UI untuk POST

Tambahkan tombol untuk POST di home_screen.dart:

```
96         const SizedBox(height: 10),  
97         ElevatedButton(  
98             onPressed: () => _handleApiOperation(  
99                 _apiService.createPost(), 'Data berhasil ditambahkan!'),  
100             style: ElevatedButton.styleFrom(backgroundColor: Colors.green),  
101             child: const Text('POST'),  
102         ), // ElevatedButton  
103         const SizedBox(height: 10),  
104         ElevatedButton(  
105             onPressed: () => _handleApiOperation(  
106                 _apiService.updatePost(), 'Data berhasil diupdate!'),  
107             style: ElevatedButton.styleFrom(backgroundColor: Colors.purple),  
108             child: const Text('UPDATE'),  
109         ), // ElevatedButton
```

5. Implementasi HTTP UPDATE

a. Membuat Service UPDATE

Tambahkan metode berikut ke api_service.dart:

```

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 }

```

b. Membuat tampilan UI untuk UPDATE

Tambahkan logika untuk memperbarui postingan di home_screen.dart:

```

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 ), // ElevatedButton

```

6. Implementasi HTTP DELETE

a. Membuat Service DELETE

Tambahkan metode berikut ke api_service.dart:

```

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 }

```

- b. Membuat tampilan UI untuk DELETE

Tambahkan tombol untuk DELETE memperbarui postingan di home_screen.dart:

```
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 ), // ElevatedButton
```

D. Hasil saat praktikum

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     );
```

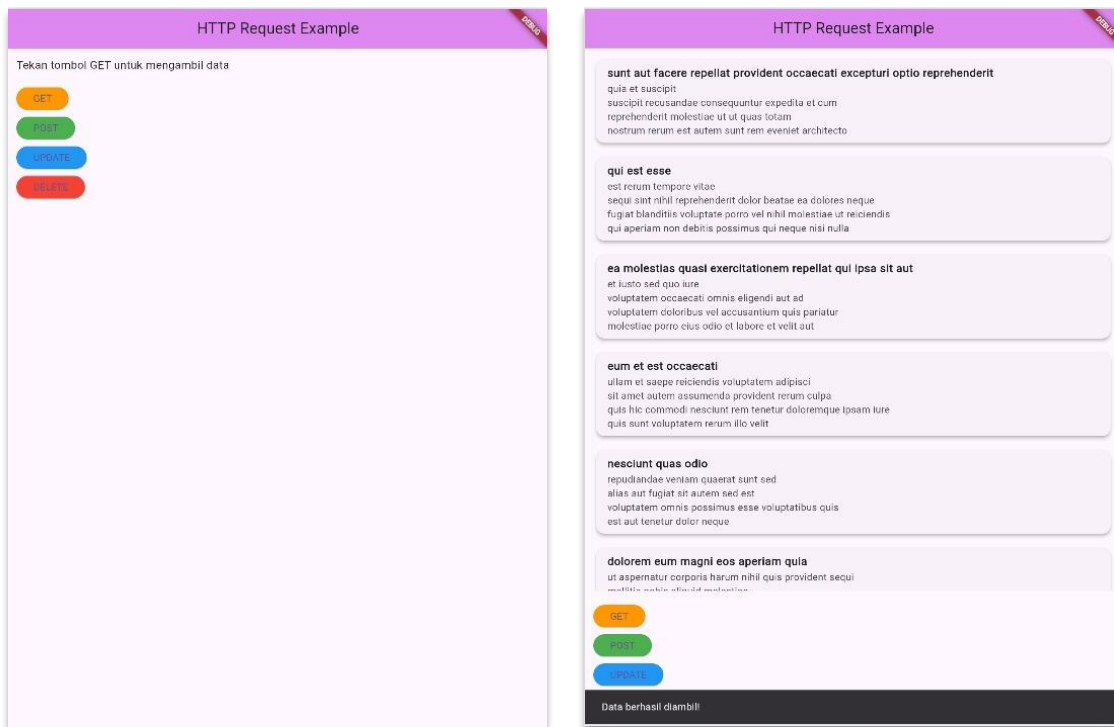
```

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 }

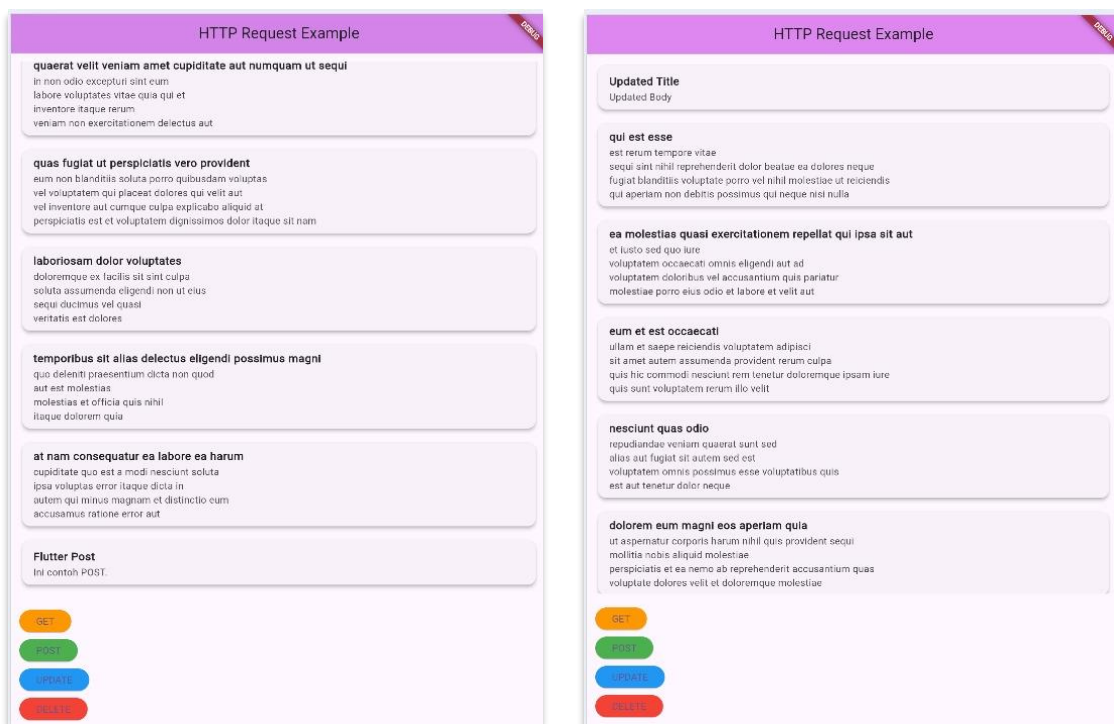
```

2. Screenshot output

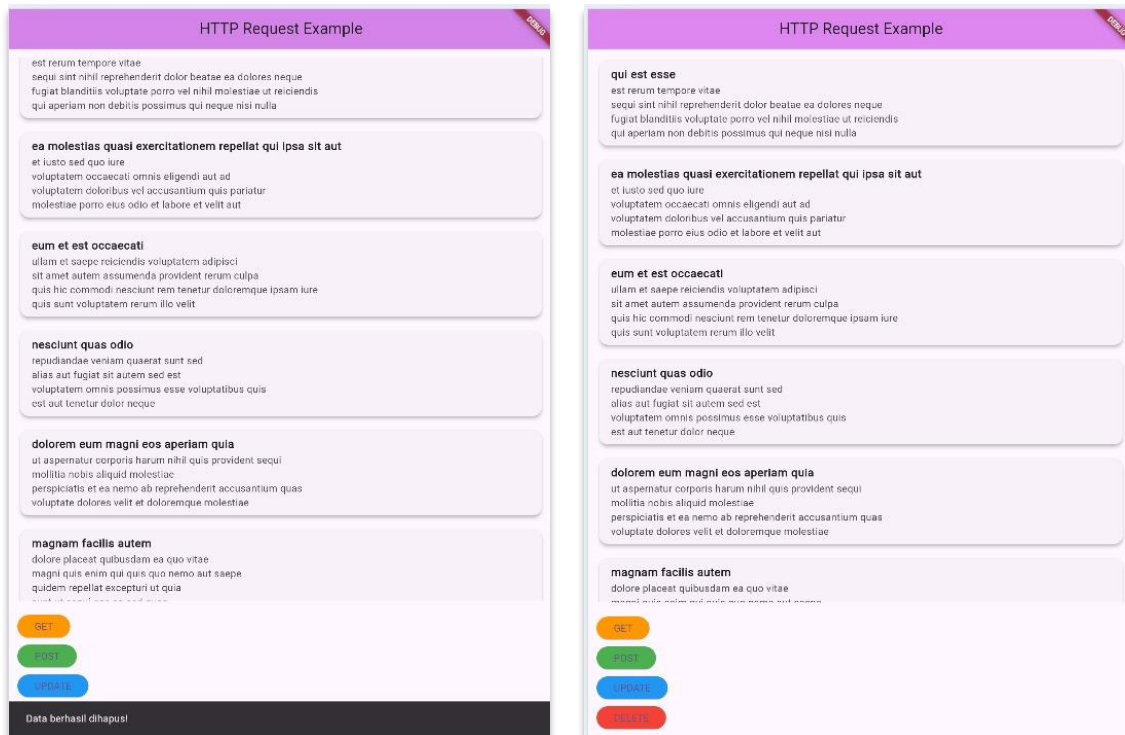
Tampilan awal saat aplikasi di run dan saat mengklik tombol get



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.

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.

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

Jawab

1. Source code
 - Source code main.dart

```
1  import 'package:flutter/material.dart';
2  import 'package:get/get.dart';
3  import 'package:unguided/screen/homescreen.dart';
4
5  void main() {
6    runApp(const MyApp());
7  }
8
9  class MyApp extends StatelessWidget {
10    const MyApp({super.key});
11
12    @override
13    Widget build(BuildContext context) {
14      return GetMaterialApp(
15        title: 'Flutter GetX Demo',
16        theme: ThemeData(
17          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
18          useMaterial3: true,
19        ),
20        home: const HomeScreen(),
21      );
22    }
23  }
24
25
```

- Source code `homescreen.dart`

[illegible]


```

46         ),
47     ),
48 ),
49 );
50 },
51 ),
52 )),
53 const SizedBox(height: 20),
54 Row(
55   mainAxisAlignment: MainAxisAlignment.spaceEvenly,
56   children: [
57     ElevatedButton(
58       onPressed: controller.fetchPosts,
59       style: ElevatedButton.styleFrom(
60         backgroundColor: const Color.fromARGB(255, 230, 188, 255),
61         foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
62       ),
63       child: const Text('GET'),
64     ),
65     ElevatedButton(
66       onPressed: controller.createPost,
67       style: ElevatedButton.styleFrom(
68         backgroundColor: const Color.fromARGB(255, 255, 232, 163),
69         foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
70       ),
71       child: const Text('POST'),
72     ),
73     ElevatedButton(
74       onPressed: controller.updatePost,
75       style: ElevatedButton.styleFrom(
76         backgroundColor: const Color.fromARGB(255, 209, 255, 148),
77         foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
78       ),
79       child: const Text('UPDATE'),
80     ),
81     ElevatedButton(
82       onPressed: controller.deletePost,
83       style: ElevatedButton.styleFrom(
84         backgroundColor: const Color.fromARGB(255, 255, 188, 214),
85         foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
86       ),
87       child: const Text('DELETE'),
88     ),
89   ],
90 ),
91 ],
92 ),
93 ),
94 );
95 }
96 }
97

```

- Source code controller.dart

```
1 import 'dart:convert';
2 import 'package:flutter/material.dart';
3 import 'package:get/get.dart';
4 import 'package:http/http.dart' as http;
5
6 class ApiController extends GetxController {
7   final String baseUrl = "https://jsonplaceholder.typicode.com";
8
9   var posts = <dynamic>[].obs;
10  var isLoading = false.obs;
11
12  // Snackbar helper
13  void showSuccessSnackBar(String message) {
14    Get.snackbar(
15      'Success',
16      message,
17      backgroundColor: const Color.fromARGB(255, 247, 255, 140),
18      colorText: const Color.fromARGB(255, 20, 20, 20),
19      snackPosition: SnackPosition.TOP, // Changed to TOP
20      duration: const Duration(seconds: 2),
21    );
22  }
23
24  void showErrorSnackBar(String message) {
25    Get.snackbar(
26      'Error',
27      message,
28      backgroundColor: const Color.fromARGB(255, 255, 104, 74),
29      colorText: const Color.fromARGB(255, 20, 20, 20),
30      snackPosition: SnackPosition.TOP, // Changed to TOP
31      duration: const Duration(seconds: 2),
32    );
33  }
34
35  // GET Posts
36  Future<void> fetchPosts() async {
37    isLoading.value = true;
38    try {
39      final response = await http.get(Uri.parse('$baseUrl/posts'));
40      if (response.statusCode == 200) {
41        posts.value = json.decode(response.body);
42        showSuccessSnackBar('Berhasil mengambil data!');
43      } else {
44        throw Exception('Failed to load posts');
45      }
46    } catch (e) {
47      showErrorSnackBar('Error: $e');
48    } finally {
49      isLoading.value = false;
50    }
51  }
```

```
52
53 // POST Data
54 Future<void> createPost() async {
55   isLoading.value = true;
56   try {
57     final response = await http.post(
58       Uri.parse('$baseUrl/posts'),
59       headers: {'Content-Type': 'application/json'},
60       body: json.encode({
61         'title': 'Flutter Post',
62         'body': 'Ini contoh POST.',
63         'userId': 1,
64       }),
65     );
66     if (response.statusCode == 201) {
67       posts.add({
68         'title': 'Flutter Post',
69         'body': 'Ini contoh POST.',
70         'id': posts.length + 1,
71       });
72       showSuccessSnackBar('Yeay data kamu berhasil ditambahkan!');
73     } else {
74       throw Exception('Failed to create post');
75     }
76   } catch (e) {
77     showErrorSnackBar('Error: $e');
78   } finally {
79     isLoading.value = false;
80   }
81 }
82
83 // UPDATE Data
84 Future<void> updatePost() async {
85   isLoading.value = true;
86   try {
```

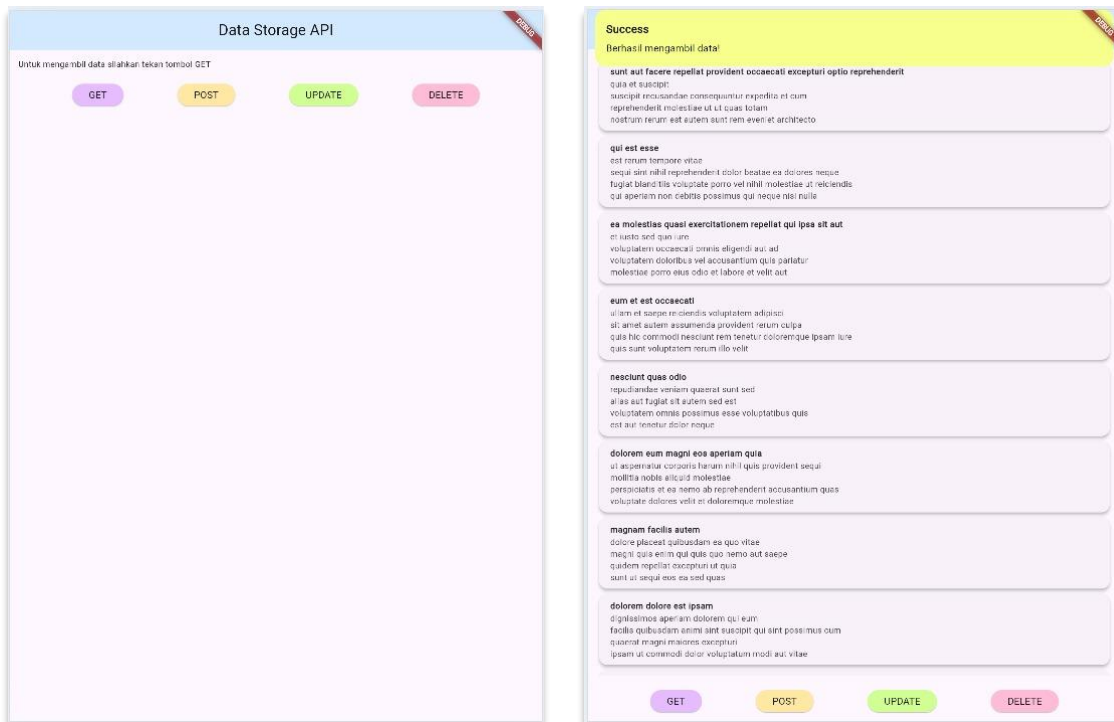
```

87     final response = await http.put(
88         Uri.parse('$baseUrl/posts/1'),
89         body: json.encode({
90             'title': 'Updated Title',
91             'body': 'Updated Body',
92             'userId': 1,
93         })),
94     );
95     if (response.statusCode == 200) {
96         var updatedPost = posts.firstWhere((post) => post['id'] == 1);
97         updatedPost['title'] = 'Updated Title';
98         updatedPost['body'] = 'Updated Body';
99         showSuccessSnackBar('Kamu berhasil memperbarui data!');
100     } else {
101         throw Exception('Failed to update post');
102     }
103 } catch (e) {
104     showErrorSnackBar('Error: $e');
105 } finally {
106     isLoading.value = false;
107 }
108 }
109
110 // DELETE Data
111 Future<void> deletePost() async {
112     isLoading.value = true;
113     try {
114         final response = await http.delete(Uri.parse('$baseUrl/posts/1'));
115         if (response.statusCode == 200) {
116             posts.removeWhere((post) => post['id'] == 1);
117             showSuccessSnackBar('Data berhasil dihapus!');
118         } else {
119             throw Exception('Failed to delete post');
120         }
121     } catch (e) {
122         showErrorSnackBar('Error: $e');
123     } finally {
124         isLoading.value = false;
125     }
126 }
127 }
128

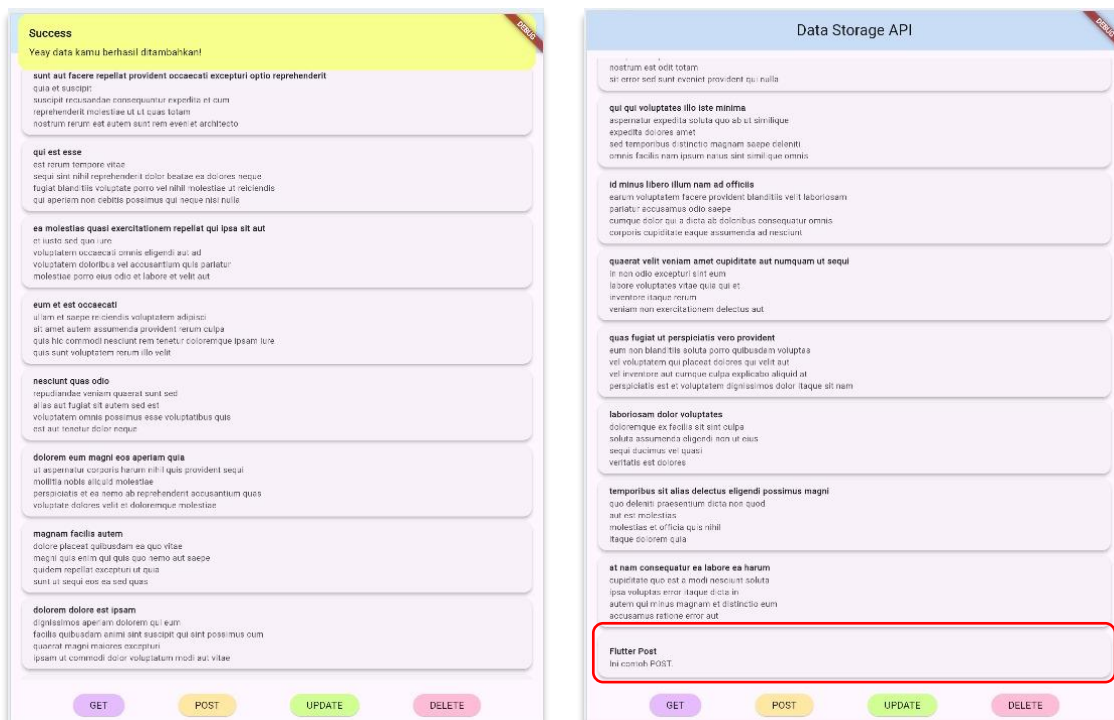
```

2. Hasil screenshot output

- Screenshot output tampilan awal dan ketika memilih tombol “GET”.



- Screenshot output tampilan ketika memilih tombol "POST" dan tampilan data yang berhasil ditambahkan.



- Screenshot output tampilan ketika memilih tombol "UPDATE" dan tampilan data yang berhasil diperbarui.

Success

Kamu berhasil memperbarui data!

Updated Title

Updated Body

qui est esse

est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditia 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

ut enim et saepe reiciendis voluptatem adipisci
atque autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur laboreque ipsa mollit
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

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae
persiciatis et ea nemo ab reprehenderit accusantium quas
voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae
magni quis enim qui quis quo nemo aut saepe
quidem repellat excepturi ut quia
sunt ut sequi eos ea sed quas

dolorem dolore est ipsam

dignissimos aperiam dolorem qui eum
facilis quibusdam animi sint suscipit qui sint possimus cum
quaerat magni maiores excepturi
ipsam ut commodi dolor voluptatum modi aut vitae

nesciunt iure omnis dolorem tempora et accusantium

consectetur animi nesciunt iure dolore

GET

POST

UPDATE

DELETE

Data Storage API

Updated Title

Updated Body

qui est esse

est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditia 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

ut enim et saepe reiciendis voluptatem adipisci
atque autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur laboreque ipsa mollit
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

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae
persiciatis et ea nemo ab reprehenderit accusantium quas
voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae
magni quis enim qui quis quo nemo aut saepe
quidem repellat excepturi ut quia
sunt ut sequi eos ea sed quas

dolorem dolore est ipsam

dignissimos aperiam dolorem qui eum
facilis quibusdam animi sint suscipit qui sint possimus cum
quaerat magni maiores excepturi
ipsam ut commodi dolor voluptatum modi aut vitae

nesciunt iure omnis dolorem tempora et accusantium

consectetur animi nesciunt iure dolore

GET

POST

UPDATE

DELETE

- Screenshot output tampilan ketika memilih tombol "DELETE".

Success

Data berhasil dihapus!

qui est esse

est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditia 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

ut enim et saepe reiciendis voluptatem adipisci
atque autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur laboreque ipsa mollit
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

dolorem eum magni eos aperiam quia

ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae
persiciatis et ea nemo ab reprehenderit accusantium quas
voluptate dolores velit et doloremque molestiae

magnam facilis autem

dolore placeat quibusdam ea quo vitae
magni quis enim qui quis quo nemo aut saepe
quidem repellat excepturi ut quia
sunt ut sequi eos ea sed quas

dolorem dolore est ipsam

dignissimos aperiam dolorem qui eum
facilis quibusdam animi sint suscipit qui sint possimus cum
quaerat magni maiores excepturi
ipsam ut commodi dolor voluptatum modi aut vitae

nesciunt iure omnis dolorem tempora et accusantium

consectetur animi nesciunt iure dolore
enim quia ad
veniam autem ut quam aut nobis
et est aut quod aut provident voluptas autem voluptas

GET

POST

UPDATE

DELETE

3. Penjelasan

Program tersebut terdiri dari HomeScreen dan logika bisnis di ApiController. HomeScreen adalah sebuah widget stateless yang menampilkan antarmuka untuk mengelola data menggunakan API dengan metode GET, POST, UPDATE, dan DELETE menggunakan library GetX. AppBar berfungsi sebagai area utama untuk menampilkan data dan memiliki tombol operasi CRUD. Data ditampilkan dalam bentuk daftar kartu menggunakan ListView.builder, di mana setiap elemen daftar diambil dari controller.posts. Tombol operasi GET, POST, UPDATE, DELETE dirancang dengan warna yang menarik dan memanggil fungsi terkait dari ApiController. Menggunakan HTTP requests, ApiController menangani logika bisnis aplikasi. Kontroler ini memiliki properti baseUrl untuk URL API, posts untuk menyimpan data dari server, dan isLoading untuk menunjukkan status pemrosesan. Operasi CRUD meliputi pengambilan data (GET), penambahan data baru (POST), pembaruan data (UPDATE), dan penghapusan data (DELETE). Setiap operasi dilengkapi dengan snackbar, yang menampilkan pesan sukses atau kesalahan di bagian atas layar menggunakan fitur Get.snackbar. Mekanisme reaktif Obx memastikan bahwa antarmuka pengguna dan logika bekerja secara sinkron, yang berarti bahwa antarmuka pengguna diperbarui secara instan dengan setiap perubahan data.