

**LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XIV
DATA STORAGE 'API'**



Disusun Oleh :
Muhammad Abdul Aziz / 2211104026
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

File main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:prak_14/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 }
```

File screens/home_screen.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:prak_14/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   // Fungsi untuk menampilkan Snackbar
16   void _showSnackBar(String message) {
17     ScaffoldMessenger.of(context)
18       .showSnackBar(SnackBar(content: Text(message)));
19   }
20
21   // Fungsi untuk memanggil API dan menangani operasi
22   Future<void> _handleApiOperation(
23     Future<void> operation, String successMessage) async {
24     setState(() {
25       _isLoading = true;
26     });
27     try {
28       await operation; // Menjalankan operasi API
29       setState(() {
30         _posts = _apiService.posts;
31       });
32       _showSnackBar(successMessage);
33     } catch (e) {
34       _showSnackBar('Error: $e');
35     } finally {
36       setState(() {
37         _isLoading = false;
38       });
39     }
40   }
```

```

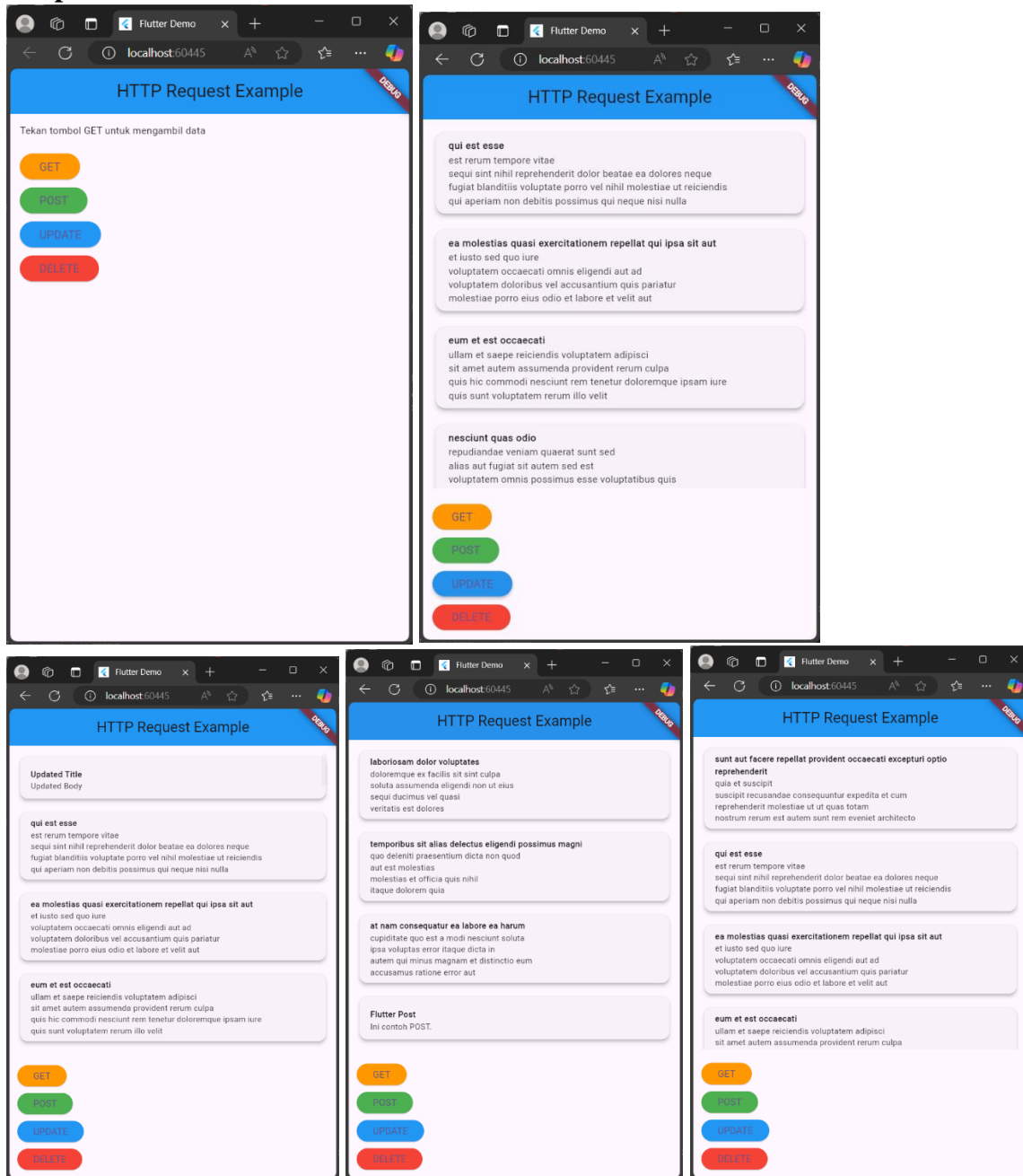
1  @override
2  Widget build(BuildContext context) {
3    return Scaffold(
4      appBar: AppBar(
5        title: const Text('HTTP Request Example'),
6        centerTitle: true,
7        backgroundColor: Colors.blue,
8      ),
9      body: Padding(
10       padding: const EdgeInsets.all(12.0),
11       child: Column(
12         crossAxisAlignment: CrossAxisAlignment.start,
13         children: [
14           _isLoading
15             ? const Center(child: CircularProgressIndicator())
16             : _posts.isEmpty
17               ? const Text(
18                 "Tekan tombol GET untuk mengambil data",
19                 style: TextStyle(fontSize: 12),
20               )
21               : Expanded(
22                 child: ListView.builder(
23                   itemCount: _posts.length,
24                   itemBuilder: (context, index) {
25                     return Padding(
26                       padding: const EdgeInsets.only(bottom: 12.0),
27                       child: Card(
28                         elevation: 4,
29                         child: ListTile(
30                           title: Text(
31                             _posts[index]['title'],
32                             style: const TextStyle(
33                               fontWeight: FontWeight.bold,
34                               fontSize: 12),
35                           ),
36                           subtitle: Text(
37                             _posts[index]['body'],
38                             style: const TextStyle(fontSize: 12),
39                           ),
40                         ),
41                       ),
42                     );
43                   },
44                 ),
45               ),
46           const SizedBox(height: 20),
47           ElevatedButton(
48             onPressed: () => _handleApiOperation(
49               _apiService.fetchPosts(), 'Data berhasil diambil!'),
50             style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
51             child: const Text('GET'),
52           ),
53           const SizedBox(height: 10),
54           ElevatedButton(
55             onPressed: () => _handleApiOperation(
56               _apiService.createPost(), 'Data berhasil ditambahkan!'),
57             style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
58             child: const Text('POST'),
59           ),
60           const SizedBox(height: 10),
61           ElevatedButton(
62             onPressed: () => _handleApiOperation(
63               _apiService.updatePost(), 'Data berhasil diperbarui!'),
64             style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
65             child: const Text('UPDATE'),
66           ),
67           const SizedBox(height: 10),
68           ElevatedButton(
69             onPressed: () => _handleApiOperation(
70               _apiService.deletePost(), 'Data berhasil dihapus!'),
71             style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
72             child: const Text('DELETE'),
73           ),
74         ],
75       ),
76     ),
77   );
78 }
79 }
80

```

File services/api_service.dart

```
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   // Fungsi untuk GET data
8   Future<void> fetchPosts() async {
9     final response = await http.get(Uri.parse('$baseUrl/posts'));
10    if (response.statusCode == 200) {
11      posts = json.decode(response.body);
12    } else {
13      throw Exception('Failed to load posts');
14    }
15  }
16
17   // Fungsi untuk POST data
18   Future<void> createPost() async {
19     final response = await http.post(
20       Uri.parse('$baseUrl/posts'),
21       headers: {'Content-Type': 'application/json'},
22       body: json.encode({
23         'title': 'Flutter Post',
24         'body': 'Ini contoh POST.',
25         'userId': 1,
26       })),
27     );
28     if (response.statusCode == 201) {
29       posts.add({
30         'title': 'Flutter Post',
31         'body': 'Ini contoh POST.',
32         'id': posts.length + 1,
33       });
34     } else {
35       throw Exception('Failed to create post');
36     }
37   }
38
39   // Fungsi untuk UPDATE data
40   Future<void> updatePost() async {
41     final response = await http.put(
42       Uri.parse('$baseUrl/posts/1'),
43       body: json.encode({
44         'title': 'Updated Title',
45         'body': 'Updated Body',
46         'userId': 1,
47       })),
48     );
49     if (response.statusCode == 200) {
50       final updatedPost = posts.firstWhere((post) => post['id'] == 1);
51       updatedPost['title'] = 'Updated Title';
52       updatedPost['body'] = 'Updated Body';
53     } else {
54       throw Exception('Failed to update post');
55     }
56   }
57
58   // Fungsi untuk DELETE data
59   Future<void> deletePost() async {
60     final response = await http.delete(
61       Uri.parse('$baseUrl/posts/1'),
62     );
63     if (response.statusCode == 200) {
64       posts.removeWhere((post) => post['id'] == 1);
65     } else {
66       throw Exception('Failed to delete post');
67     }
68   }
69 }
70
```

Output



Penjelasan

Kode ini adalah implementasi HTTP Request di Flutter menggunakan REST API dari *JsonPlaceholder* dengan struktur terpisah antara logika dan tampilan. File utama `main.dart` menjalankan aplikasi dengan halaman utama `HomeScreen`. Kelas `ApiService` menangani logika HTTP Request yang terdiri dari empat operasi, yaitu `fetchPosts()` untuk mengambil data, `createPost()` untuk menambahkan data, `updatePost()` untuk memperbarui data, dan `deletePost()` untuk menghapus data. Data yang diperoleh dari API disimpan dalam variabel `posts` dan digunakan untuk ditampilkan di antarmuka pengguna. Pada `HomeScreen`, komponen UI dibangun menggunakan `StatefulWidget`, di mana status loading dikelola melalui variabel `_isLoading` dan data disimpan dalam `_posts`. Fungsi `_handleApiOperation()` digunakan untuk memanggil operasi API secara dinamis, menampilkan `SnackBar` sebagai notifikasi keberhasilan atau kegagalan operasi. Aplikasi ini memiliki empat tombol utama (GET, POST, UPDATE, DELETE) yang memicu fungsi terkait di `ApiService` dan menampilkan data dalam bentuk `ListView` berupa kartu dengan judul dan isi dari setiap post. Dengan pendekatan ini, logika pemrosesan data API dipisahkan dari tampilan, sehingga kode menjadi lebih rapi, modular, dan mudah dikelola.

UNGUIDED

File main.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:get/get.dart';
3 import 'package:prak_14/screens/home_screen.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 }
```

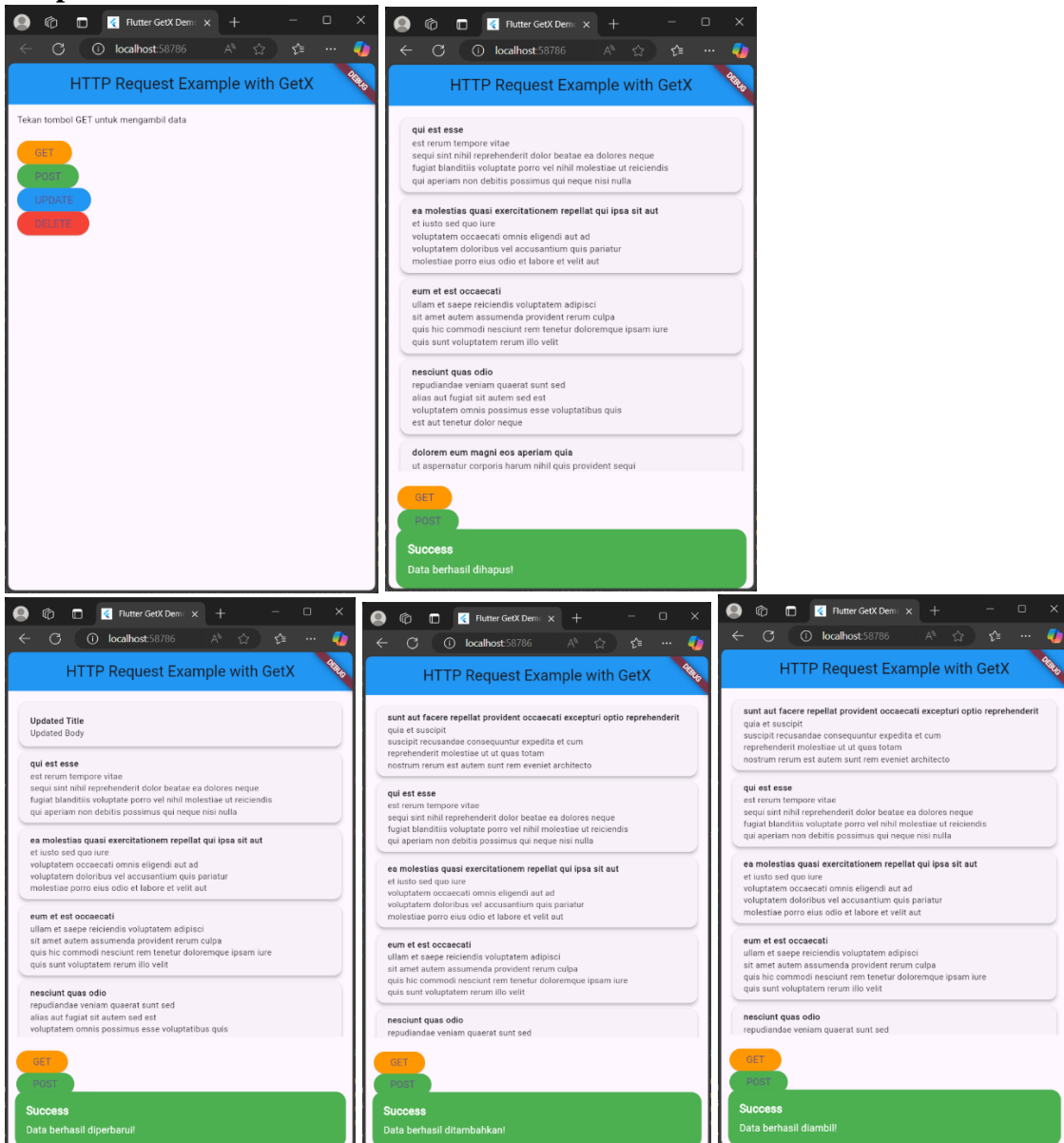
File screens/home_screen.dart

```
1 import 'package:flutter/material.dart';
2 import 'package:get/get.dart';
3 import 'package:prak_14/controllers/post_controller.dart';
4
5 class HomeScreen extends StatelessWidget {
6   const HomeScreen({super.key});
7
8   @override
9   Widget build(BuildContext context) {
10     final ApiController controller = Get.put(ApiController());
11
12     return Scaffold(
13       appBar: AppBar(
14         title: const Text('HTTP Request Example with GetX'),
15         centerTitle: true,
16         backgroundColor: Colors.blue,
17       ),
18       body: Padding(
19         padding: const EdgeInsets.all(12.0),
20         child: Column(
21           crossAxisAlignment: CrossAxisAlignment.start,
22           children: [
23             Obx(() => controller.isLoading.value
24               ? const Center(child: CircularProgressIndicator())
25               : controller.posts.isEmpty
26                 ? const Text(
27                   "Tekan tombol GET untuk mengambil data",
28                   style: TextStyle(fontSize: 12),
29                 )
30               : Expanded(
31                 child: ListView.builder(
32                   itemCount: controller.posts.length,
33                   itemBuilder: (context, index) {
34                     return Card(
35                       elevation: 4,
36                       child: ListTile(
37                         title: Text(
38                           controller.posts[index]['title'],
39                           style: const TextStyle(
40                             fontWeight: FontWeight.bold,
41                             fontSize: 12),
42                         ),
43                         subtitle: Text(
44                           controller.posts[index]['body'],
45                           style: const TextStyle(fontSize: 12),
46                         ),
47                       ),
48                     );
49                   },
50                 ),
51             ),
52           ],
53         ),
54       ),
55     );
56   }
57 }
```

File controllers/post_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: Colors.green,
18      colorText: Colors.white,
19      snackPosition: SnackPosition.BOTTOM,
20      duration: const Duration(seconds: 2),
21    );
22  }
23
24  void showErrorSnackBar(String message) {
25    Get.snackbar(
26      'Error',
27      message,
28      backgroundColor: Colors.red,
29      colorText: Colors.white,
30      snackPosition: SnackPosition.BOTTOM,
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('Data berhasil diambil!');
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('Data 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('Data berhasil diperbarui!');
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 }
```


Output



Penjelasan

Kode ini adalah implementasi HTTP Request di Flutter menggunakan REST API dari *JsonPlaceholder* dengan struktur terpisah antara logika dan tampilan. File utama `main.dart` menjalankan aplikasi dengan halaman utama `HomeScreen`. Kelas `ApiService` menangani logika HTTP Request yang terdiri dari empat operasi, yaitu `fetchPosts()` untuk mengambil data, `createPost()` untuk menambahkan data, `updatePost()` untuk memperbarui data, dan `deletePost()` untuk menghapus data. Data yang diperoleh dari API disimpan dalam variabel `posts` dan digunakan untuk ditampilkan di antarmuka pengguna. Pada `HomeScreen`, komponen UI dibangun menggunakan `StatefulWidget`, di mana status loading dikelola melalui variabel `_isLoading` dan data disimpan dalam `_posts`. Fungsi `_handleApiOperation()` digunakan untuk memanggil operasi API secara dinamis, menampilkan `SnackBar` sebagai notifikasi keberhasilan atau kegagalan operasi. Aplikasi ini memiliki empat tombol utama (GET, POST, UPDATE, DELETE) yang memicu fungsi terkait di `ApiService` dan menampilkan data dalam bentuk `ListView` berupa kartu dengan judul dan isi dari setiap post. Dengan pendekatan ini, logika pemrosesan data API dipisahkan dari tampilan, sehingga kode menjadi lebih rapi, modular, dan mudah dikelola.