LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

Data Storage (API)

Disusun Oleh :

Satria Ariq Adelard

Dompas/2211104033 SE 06 2

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. GUIDED

Source Code

- main.dart

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan14/screen/homepage_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: const HomepageScreen(),
    );
  }
}
```

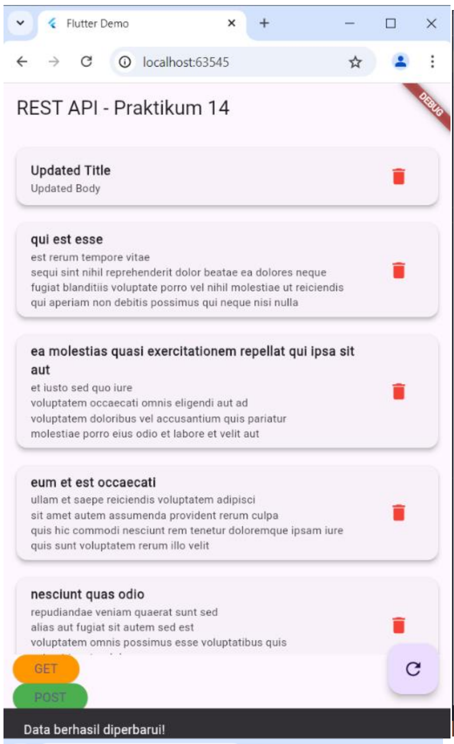
- homepage_screen.dart

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan14/services/api_service.dart';

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

  @override
  State<HomepageScreen> createState() => _HomepageScreenState();
}

class _HomepageScreenState extends State<HomepageScreen> {
  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; // Mengupdate posts setelah operasi
berhasil
      });
      _showSnackBar(successMessage); // Menampilkan SnackBar sukses
    } catch (e) {
      _showSnackBar('Error: $e'); // Menampilkan SnackBar error
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API - Praktikum 14'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(12),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Indikator loading
            if (_isLoading)
              const Center(child: CircularProgressIndicator())
            // Pesan jika data kosong
            else if (_posts.isEmpty)
              const Text(
                "Tekan tombol GET untuk mengambil data",
                style: TextStyle(fontSize: 14),
              )
            // Menampilkan daftar data
            else
              Expanded(
                child: ListView.builder(
                  itemCount: _posts.length,
                  itemBuilder: (context, index) {
                    final post = _posts[index];
                    return Padding(
                      padding: const EdgeInsets.only(bottom: 12.0),
                      child: Card(
                        elevation: 4,
                        child: ListTile(
                          title: Text(
              post['title'] ?? 'No Title',
              style: const TextStyle(
                fontWeight: FontWeight.bold, fontSize: 14),
             ),
             subtitle: Text(
              post['body'] ?? 'No Body',
              style: const TextStyle(fontSize: 12),
             ),
             trailing: IconButton(
              icon: const Icon(Icons.delete, color:
Colors.red),
              onPressed: () => _handleApiOperation(
               _apiService.deletePost(post['id']),
               'Data berhasil dihapus!',
              ),
             ),
            ),
           ),
          );
         },
        ),
       ),
      // Tombol GET
      ElevatedButton(
       onPressed: () => _handleApiOperation(
         _apiService.fetchPosts(), 'Data berhasil diambil!'),
       style: ElevatedButton.styleFrom(backgroundColor:
Colors.orange),
       child: const Text('GET'),
      ),
      // Tombol POST
      ElevatedButton(
       onPressed: () => _handleApiOperation(
         _apiService.createPost(), 'Data berhasil ditambahkan!'),
       style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
       child: const Text('POST'),
      ),
      // Tombol UPDATE
      ElevatedButton(
       onPressed: () => _handleApiOperation(
        _apiService.updatePost(1, 'Updated Title', 'Updated Body'),
        'Data berhasil diperbarui!',
       ),
       style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
       child: const Text('UPDATE'),
      ),
     ],
    ),
   ),
   floatingActionButton: FloatingActionButton(
    onPressed: () => _handleApiOperation(
     _apiService.fetchPosts(), // Contoh pemanggilan API
     'Posts fetched successfully!',
        ),
        child: const Icon(Icons.refresh),
      ),
    );
  }
```

- 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 {
    try {
      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. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error fetching posts: $e');
    }
  }

  // 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) {
        final newPost =
            json.decode(response.body); // Mengambil respons baru dari server
        posts.add({
          'id': newPost['id'] ?? (posts.length + 1),
          'title': newPost['title'] ?? 'Flutter Post',
          'body': newPost['body'] ?? 'Ini contoh POST.',
        });
      } else {
        throw Exception(
            'Failed to create post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error creating post: $e');
    }
  }

  // Fungsi untuk UPDATE data
  Future<void> updatePost(int postId, String newTitle, String newBody) async
{
    try {
      final response = await http.put(
        Uri.parse('$baseUrl/posts/$postId'),
        headers: {'Content-Type': 'application/json'},
        body: json.encode({
          'title': newTitle,
          'body': newBody,
          'userId': 1,
        }),
      );

      if (response.statusCode == 200) {
        // Update post dalam list lokal
        final index = posts.indexWhere((post) => post['id'] == postId);
        if (index != -1) {
          posts[index]['title'] = newTitle;
          posts[index]['body'] = newBody;
        }
      } else {
        throw Exception(
            'Failed to update post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error updating post: $e');
    }
  }

  // Fungsi untuk DELETE data
  Future<void> deletePost(int postId) async {
    try {
      final response = await
http.delete(Uri.parse('$baseUrl/posts/$postId'));
      if (response.statusCode == 200) {
        // Menghapus post dari list lokal
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception(
            'Failed to delete post. Status Code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error deleting post: $e');
    }
  }
}
```

Output



REST API - Praktikum 14

Updated Title
Updated Body

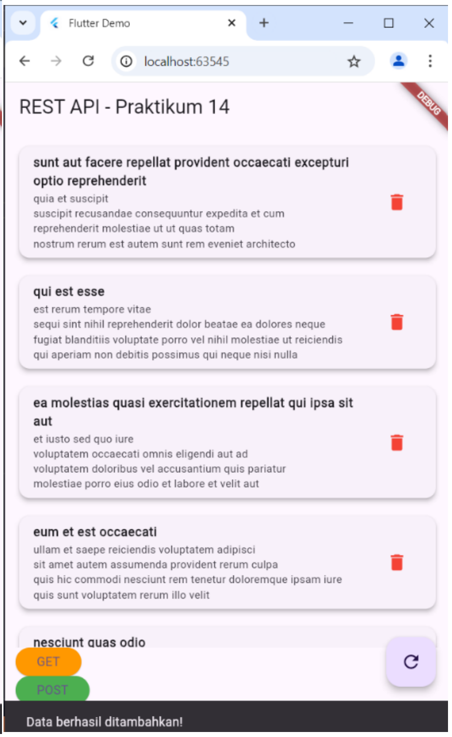
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla

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

nesciunt quas odio
repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
voluptatem omnis possimus esse voluptatibus quis

Data berhasil diperbarui!



REST API - Praktikum 14

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

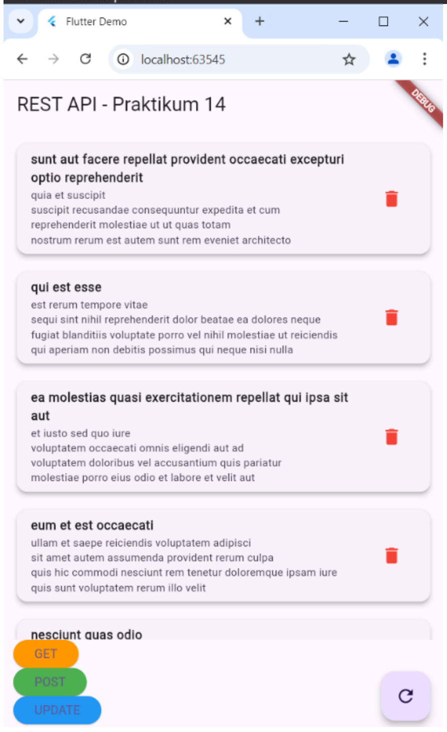
qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla

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
quis sunt voluptatem rerum illo velit

nesciunt quas odio

Data berhasil ditambahkan!



REST API - Praktikum 14

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
quis sunt voluptatem rerum illo velit

nesciunt quas odio

Deskripsi

Aplikasi ini merupakan program berbasis Flutter yang memanfaatkan widget stateful untuk menangani operasi CRUD (Create, Read, Update, Delete) pada data post menggunakan API publik JSONPlaceholder. Dalam antarmuka yang sederhana, aplikasi ini memungkinkan pengguna untuk melihat daftar post, serta melakukan pengambilan, penambahan, pembaruan, dan penghapusan data. Manajemen state, seperti penyimpanan data post dan indikator proses loading, diintegrasikan langsung ke dalam kelas State dari widget layar utama. Selain itu, SnackBar digunakan sebagai media pemberitahuan kepada pengguna untuk menginformasikan hasil dari setiap operasi, baik sukses maupun gagal. Walaupun berfungsi dengan baik, pendekatan ini dapat menyulitkan pengelolaan kode karena logika bisnis dicampur langsung dengan komponen UI.

2. UNGUIDED

Source Code

- main.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:praktikum14/screen/home_screen.dart';

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

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

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

- home_screen.dart

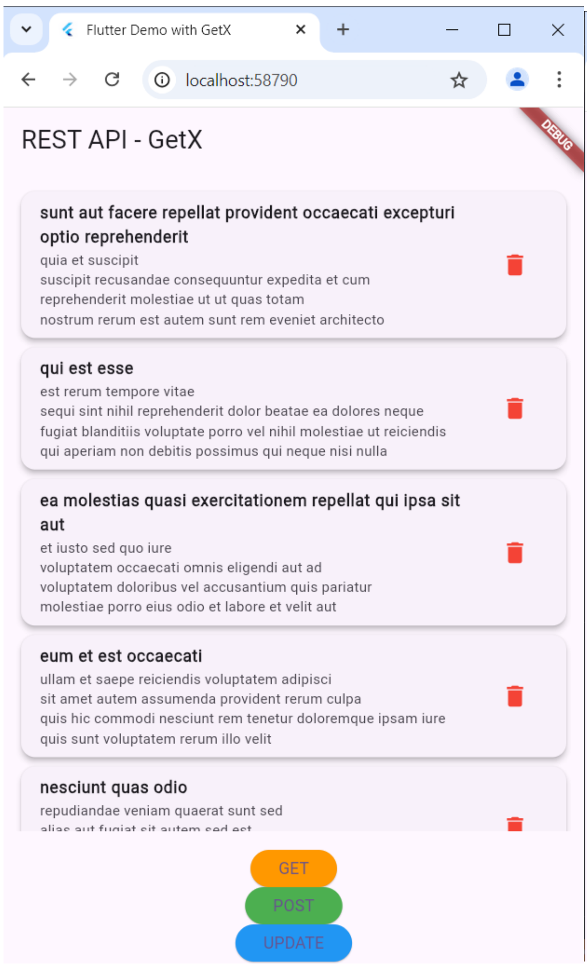
```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:praktikum14/services/api_services.dart';

```

```
class HomepageScreen extends StatelessWidget {
  HomepageScreen({super.key});
  final _apiService = ApiService();
  final posts = <dynamic>[].obs;
  final isLoading = false.obs;

  // Logika untuk fetch data
  void fetchPosts() async {
    isLoading(true);
    try {
      await _apiService.fetchPosts();
      posts.value = _apiService.posts;
      Get.snackbar('Success', 'Data successfully fetched');
    } catch (e) {
      Get.snackbar('Error', e.toString());
    } finally {
      isLoading(false);
    }
  }

  // Logika untuk create post
  void createPost() async {
    isLoading(true);
    try {
      await _apiService.createPost();
      posts.value = _apiService.posts;
      Get.snackbar('Success', 'Data successfully added');
    } catch (e) {
      Get.snackbar('Error', e.toString());
    } finally {
      isLoading(false);
    }
  }

  // Logika untuk update post
  void updatePost(int id, String title, String body) async {
    isLoading(true);
    try {
      await _apiService.updatePost(id, title, body);
      posts.value = _apiService.posts;
      Get.snackbar('Success', 'Data successfully updated');
    } catch (e) {
      Get.snackbar('Error', e.toString());
    } finally {
      isLoading(false);
    }
  }

  // Logika untuk delete post
  void deletePost(int id) async {
    isLoading(true);
    try {
      await _apiService.deletePost(id);
      posts.value = _apiService.posts;
      Get.snackbar('Success', 'Data successfully deleted');
    } catch (e) {
      Get.snackbar('Error', e.toString());
    } finally {
      isLoading(false);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API - GetX'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(12),
        child: Column(
          children: [
            Obx(() {
              if (isLoading.value) {
                return const Center(child: CircularProgressIndicator());
              }
              if (posts.isEmpty) {
                return const Text(
                  "Tekan tombol GET untuk mengambil data",
                  style: TextStyle(fontSize: 14),
                );
              }
              return Expanded(
                child: ListView.builder(
                  itemCount: posts.length,
                  itemBuilder: (context, index) {
                    final post = posts[index];
                    return Card(
                      elevation: 4,
                      child: ListTile(
                        title: Text(
                          post['title'] ?? 'No Title',
                          style: const TextStyle(
                              fontWeight: FontWeight.bold, fontSize: 14),
                        ),
                        subtitle: Text(
                          post['body'] ?? 'No Body',
                          style: const TextStyle(fontSize: 12),
                        ),
                        trailing: IconButton(
                          icon: const Icon(Icons.delete, color: Colors.red),
                          onPressed: () => deletePost(post['id']),
                        ),
                      ),
                    );
                  },
                ),
              );
            }),
            const SizedBox(height: 16),
            ElevatedButton(
              onPressed: fetchPosts,
              style: ElevatedButton.styleFrom(backgroundColor:
Colors.orange),
              child: const Text('GET'),
            ),
            ElevatedButton(
              onPressed: createPost,
              style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
              child: const Text('POST'),
            ),
            ElevatedButton(
              onPressed: () => updatePost(1, 'Updated Title', 'Updated
Body'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('UPDATE'),
            ),
          ],
        ),
      ),
    );
  }
}
```

- 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 = [];

  Future<void> fetchPosts() async {
    try {
      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. Status Code:
${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error fetching posts: $e');
    }
  }

  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) {
        final newPost = json.decode(response.body);
        posts.add({
          'id': newPost['id'] ?? (posts.length + 1),
          'title': newPost['title'] ?? 'Flutter Post',
          'body': newPost['body'] ?? 'Ini contoh POST.',
        });
      } else {
        throw Exception('Failed to create post. Status Code:
${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error creating post: $e');
    }
  }

  Future<void> updatePost(int postId, String newTitle, String newBody) async
{
    try {
      final response = await http.put(
        Uri.parse('$baseUrl/posts/$postId'),
        headers: {'Content-Type': 'application/json'},
        body: json.encode({
          'title': newTitle,
          'body': newBody,
          'userId': 1,
        }),
      );

      if (response.statusCode == 200) {
        final index = posts.indexWhere((post) => post['id'] == postId);
        if (index != -1) {
          posts[index]['title'] = newTitle;
          posts[index]['body'] = newBody;
        }
      } else {
        throw Exception('Failed to update post. Status Code:
${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error updating post: $e');
    }
  }
  Future<void> deletePost(int postId) async {
    try {
      final response = await
http.delete(Uri.parse('$baseUrl/posts/$postId'));
      if (response.statusCode == 200) {
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception('Failed to delete post. Status Code:
${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Error deleting post: $e');
    }
  }
}
```

Output



Flutter Demo with GetX

localhost:58790

REST API - GetX

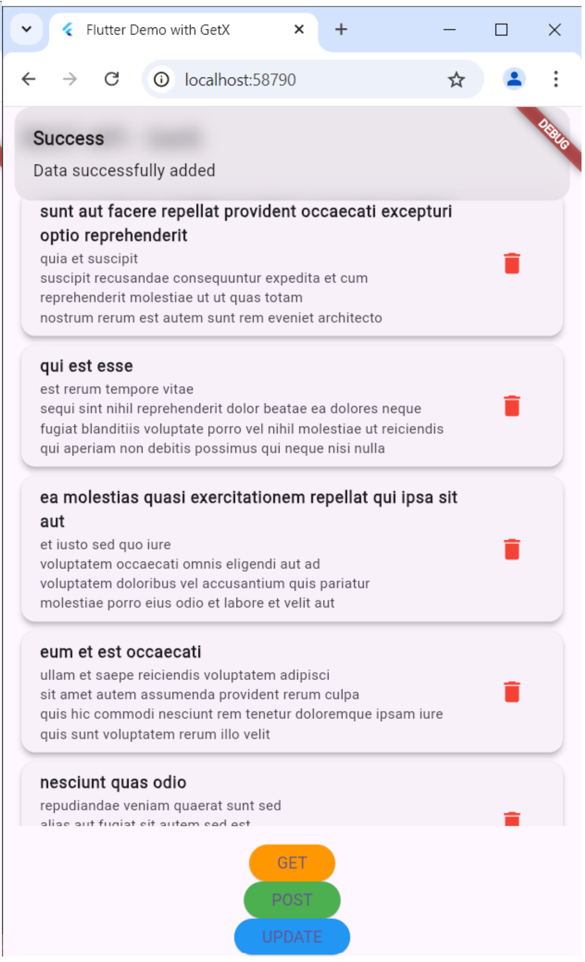
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

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
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



Flutter Demo with GetX

localhost:58790

Success
Data successfully added

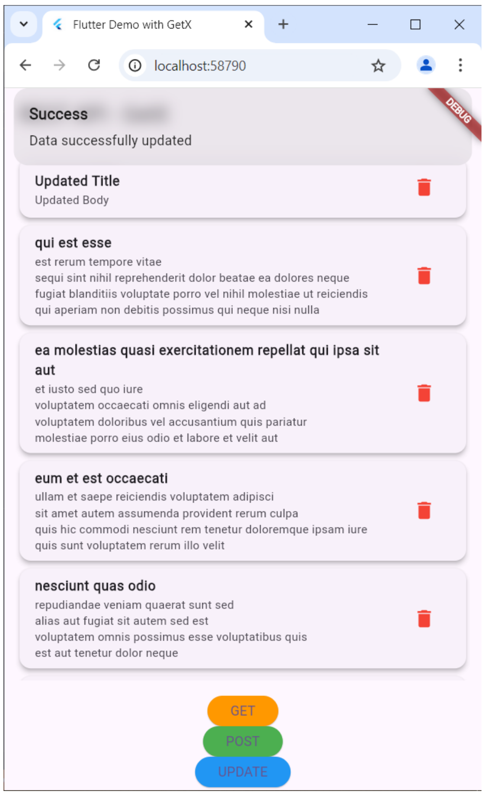
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

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
quis sunt voluptatem rerum illo velit

nesciunt quas odio
repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est



Success
Data successfully updated

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

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

Deskripsi

Aplikasi ini merupakan program berbasis Flutter yang memanfaatkan GetX untuk pengelolaan state dalam menjalankan operasi CRUD (Create, Read, Update, Delete) pada data post yang diambil dari API publik JSONPlaceholder. Aplikasi ini menghadirkan antarmuka sederhana yang memungkinkan pengguna untuk melihat daftar post, mengambil data, menambah, memperbarui, dan menghapus post. Dengan dukungan GetX, tampilan aplikasi secara otomatis diperbarui setiap kali data berubah, disertai dengan notifikasi berupa snackbar sebagai umpan balik langsung setelah setiap operasi berhasil dilakukan. Program ini dirancang menggunakan arsitektur bersih, dengan pemisahan yang jelas antara logika API dan pengelolaan state, sehingga memastikan efisiensi serta kemudahan dalam proses pengembangan dan pemeliharaan kode.