LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE BAGIAN 3

Disusun Oleh :

Aditya prabu mukti / 2211104037

SE-06-02

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

2024

GUIDED

A. home_screen.dart

Sourcecode:

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

class HomeScreen extends StatefulWidget {
  const HomeScreen({Key? key}) : super(key: key);

  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final ApiService _apiService = ApiService();
  List<dynamic> _posts = [];
  bool _isLoading = false;

  // Fungsi untuk menampilkan SnackBar
  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;
      });
    }
  }

  Future<void> _fetchPosts() async {
    await _handleApiOperation(
      _apiService.fetchPosts(),
      'Data berhasil diambil!',
    );
  }

  Future<void> _createPost() async {
    await _handleApiOperation(
      _apiService.createPost(),
      'Data berhasil ditambahkan!',
    );
  }

  Future<void> _updatePost() async {
    await _handleApiOperation(
      _apiService.updatePost(),
      'Data berhasil diperbarui!',
    );
  }

  Future<void> _deletePost() async {
    await _handleApiOperation(
      _apiService.deletePost(),
      'Data berhasil dihapus!',
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('CRUD API Example'),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                ElevatedButton(
                  onPressed: _isLoading ? null : _fetchPosts,
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
                  child: const Text('GET'),
                ),
                ElevatedButton(
                  onPressed: _isLoading ? null : _createPost,
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
                  child: const Text('POST'),
                ),
                ElevatedButton(
                  onPressed: _isLoading ? null : _updatePost,
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
                  child: const Text('UPDATE'),
                ),
                ElevatedButton(
                  onPressed: _isLoading ? null : _deletePost,
                  style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
                  child: const Text('DELETE'),
                ),
              ],
            ),
            const SizedBox(height: 16),
            _isLoading
                ? const Center(child: CircularProgressIndicator())
                : _posts.isEmpty
                    ? const Text(
                        "Tekan tombol GET untuk mengambil data",
                        style: TextStyle(fontSize: 14),
                      )
                    : Expanded(
                        child: ListView.builder(
                          itemCount: _posts.length,
                          itemBuilder: (context, index) {
                            return Padding(
                              padding: const EdgeInsets.only(bottom: 12.0),
                              child: Card(
                                elevation: 4,
                                child: ListTile(
                                  title: Text(
                                    _posts[index]['title'],
                                    style: const TextStyle(
                                      fontWeight: FontWeight.bold,
                                      fontSize: 14,
                                    ),
                                  ),
                                  subtitle: Text(
                                    _posts[index]['body'],
                                    style: const TextStyle(fontSize: 12),
                                  ),
                                ),
                              ),
                            );
                          },
                        ),
                      ),
          ],
        ),
      ),
    );
  }
}
```

Penjelasan:

Kode di atas menampilkan bagian dari aplikasi Flutter yang mengelola interaksi CRUD (Create, Read, Update, Delete) dengan sebuah API menggunakan kelas ApiService. UI aplikasi ini dibangun dengan Flutter dan melibatkan HomeScreen yang menampilkan tombol untuk setiap operasi CRUD dan sebuah daftar postingan. Operasi CRUD diimplementasikan melalui fungsi yang menghandle state loading dan menampilkan pesan melalui Snackbar berdasarkan respons API. UI juga menunjukkan indikator loading saat data sedang diproses, memberikan interaksi yang responsif dan informatif kepada pengguna.

B. api_service.dart

Sourcecode:

```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 {
    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');
    }
  }

  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': 'Ini contoh POST.',
        'userId': 1,
      }),
    );
    if (response.statusCode == 201) {
      posts.add({
        'title': 'Flutter Post',
        'body': 'Ini contoh POST.',
        'id': posts.length + 1,
      });
    } else {
      throw Exception('Failed to create post');
    }
  }

  Future<void> updatePost() async {
    final response = await http.put(
      Uri.parse('$baseUrl/posts/1'),
      headers: {'Content-Type': 'application/json'},
      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');
    }
  }

  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');
    }
  }
}
```

Penjelasan:

Kode dalam gambar menunjukkan implementasi kelas ApiService dalam Dart, yang mengelola interaksi dengan sebuah API web. Kelas ini memiliki fungsi untuk mengambil data postingan, menambahkan postingan baru, memperbarui postingan yang ada, dan menghapus postingan. Setiap fungsi melakukan permintaan ke API—menggunakan metode GET, POST, PUT, dan DELETE—dan memproses responsnya. Fungsi ini menangani respons berhasil dengan menyimpan atau memodifikasi data di variabel posts dan melemparkan pengecualian jika ada kesalahan.

C. main.dart

Sourcecode:

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan14/screen/home_screen.dart';

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

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

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

penjelasan:

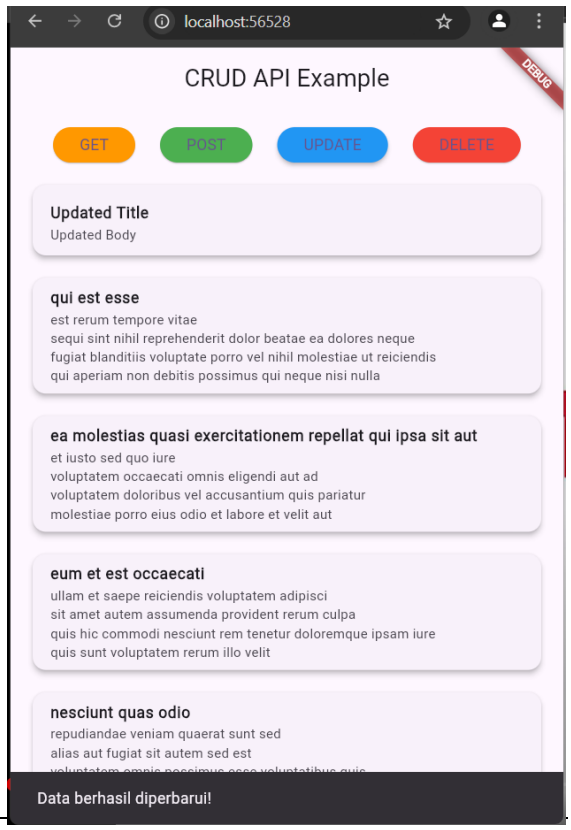
kode tersebut merupakan bagian dari aplikasi Flutter yang mendefinisikan struktur dasar aplikasi. Di dalam fungsi main(), aplikasi memulai dengan menjalankan MyApp, yang merupakan widget tanpa state (stateless). MyApp membangun sebuah MaterialApp yang menggunakan tema gelap dengan warna dasar ungu tua (deepPurple). Tema ini ditetapkan menggunakan ThemeData.fromSeed, yang menghasilkan skema warna secara otomatis berdasarkan warna ungu tua yang diberikan. Aplikasi ini menetapkan HomeScreen sebagai halaman utama, yang diimpor dari paket screen/home_screen.dart. Ini mengatur tampilan utama aplikasi saat dijalankan.

HASIL OUTPUT:


localhost:56528
CRUD API Example
DEBUG
GET
POST
UPDATE
DELETE
Tekan tombol GET untuk mengambil data


localhost:56528
CRUD API Example
DEBUG
GET
POST
UPDATE
DELETE
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
Data berhasil diambil!


localhost:56528
CRUD API Example
DEBUG
GET
POST
UPDATE
DELETE
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
Data berhasil ditambahkan!


localhost:56528
CRUD API Example
DEBUG
GET
POST
UPDATE
DELETE
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
Data berhasil diperbarui!


localhost:56528
CRUD API Example
DEBUG
GET
POST
UPDATE
DELETE
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
perspiciatis 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
Data berhasil dihapus!

UNGUIDED

Modifikasi tampilan Guided dari praktikum di atas:

1) **Gunakan State Management dengan GetX:**
   a) Atur data menggunakan state management GetX agar lebih mudah dikelola.
   b) Implementasi GetX meliputi pembuatan controller untuk mengelola data dan penggunaan widget Obx untuk menampilkan data secara otomatis setiap kali ada perubahan.
2) **Tambahkan Snackbar untuk Memberikan Respon Berhasil:**
   a) Tampilkan snackbar setelah setiap operasi berhasil, seperti menambah atau memperbarui data.
   b) Gunakan Get.snackbar agar pesan sukses muncul di layar dan mudah dipahami oleh pengguna.

Jawab:

1. main.dart

Sourcecode:

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:pertemuan14/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',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const HomepageScreen(),
    );
  }
}
```

Penjelasan:

Kode ini merupakan konfigurasi awal untuk aplikasi Flutter yang menggunakan GetMaterialApp dari paket get untuk memanfaatkan fitur manajemen state dan navigasi yang lebih efisien. Aplikasi diinisialisasi dengan tema yang berbasis warna ungu tua dan menggunakan Material Design 3. Halaman utama diatur ke HomepageScreen, yang mungkin menyediakan tampilan utama aplikasi

2. home_screen.dart

Sourcecode:

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

class ApiController extends GetxController {
  final ApiService _apiService = ApiService();

  var posts = <dynamic>[].obs;
  var isLoading = false.obs;

  Future<void> fetchPosts() async {
    isLoading(true);
    try {
      await _apiService.fetchPosts();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil diambil!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal mengambil data: $e');
    } finally {
      isLoading(false);
    }
  }

  Future<void> createPost() async {
    isLoading(true);
    try {
      await _apiService.createPost();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil ditambahkan!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal menambah data: $e');
    } finally {
      isLoading(false);
    }
  }

  Future<void> updatePost() async {
    isLoading(true);
    try {
      await _apiService.updatePost();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil diperbarui!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal memperbarui data: $e');
    } finally {
      isLoading(false);
    }
  }

  Future<void> deletePost() async {
    isLoading(true);
    try {
      await _apiService.deletePost();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Success', 'Data berhasil dihapus!');
    } catch (e) {
      Get.snackbar('Error', 'Gagal menghapus data: $e');
    } finally {
      isLoading(false);
    }
  }
}

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

  @override
  Widget build(BuildContext context) {
    final ApiController controller = Get.put(ApiController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('REST API with GetX'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(12),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Obx(() => controller.isLoading.value
                ? const Center(child: CircularProgressIndicator())
                : controller.posts.isEmpty
                    ? const Text(
                        "Tekan tombol GET untuk mengambil data",
                        style: TextStyle(fontSize: 12),
                      )
                    : Expanded(
                        child: ListView.builder(
                          itemCount: controller.posts.length,
                          itemBuilder: (context, index) {
                            return Padding(
                              padding: const EdgeInsets.only(bottom: 12.0),
                              child: Card(
                                elevation: 4,
                                child: ListTile(
                                  title: Text(
                                    controller.posts[index]['title'],
                                    style: const TextStyle(
                                        fontWeight: FontWeight.bold,
                                        fontSize: 12),
                                  ),
                                  subtitle: Text(
                                    controller.posts[index]['body'],
                                    style: const TextStyle(fontSize: 12),
                                  ),
                                ),
                              ),
                            );
                          },
                        ),
                      )),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Row(
                children: [
                  ElevatedButton(
                    onPressed: () => controller.createPost(),
                    style:
                        ElevatedButton.styleFrom(backgroundColor: Colors.green),
                    child: const Text('POST'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    onPressed: () => controller.fetchPosts(),
                    style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.orange),
                    child: const Text('GET'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    onPressed: () => controller.updatePost(),
                    style:
                        ElevatedButton.styleFrom(backgroundColor: Colors.blue),
                    child: const Text('UPDATE'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    onPressed: () => controller.deletePost(),
                    style:
                        ElevatedButton.styleFrom(backgroundColor: Colors.red),
                    child: const Text('DELETE'),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Penjelasan:

Kode ini mendeskripsikan implementasi aplikasi Flutter yang menggunakan Getx untuk manajemen state dan komunikasi API. Didefinisikan ApiController yang mengelola operasi CRUD—mengambil, menambah, memperbarui, dan menghapus postingan—dengan bantuan ApiService. Setiap operasi mengaktifkan indikator loading sebelum melakukan permintaan dan memperbarui daftar postingan jika berhasil, serta menampilkan pesan kesuksesan atau kegagalan menggunakan Get.snackbar. HomepageScreen adalah UI utama yang menampilkan postingan dalam ListView dan tombol untuk setiap aksi CRUD. UI ini menggambarkan status loading atau menampilkan postingan yang tersedia, memungkinkan pengguna untuk berinteraksi dengan data secara langsung dari antarmuka.

3. api_service.dart

Sourcecode:

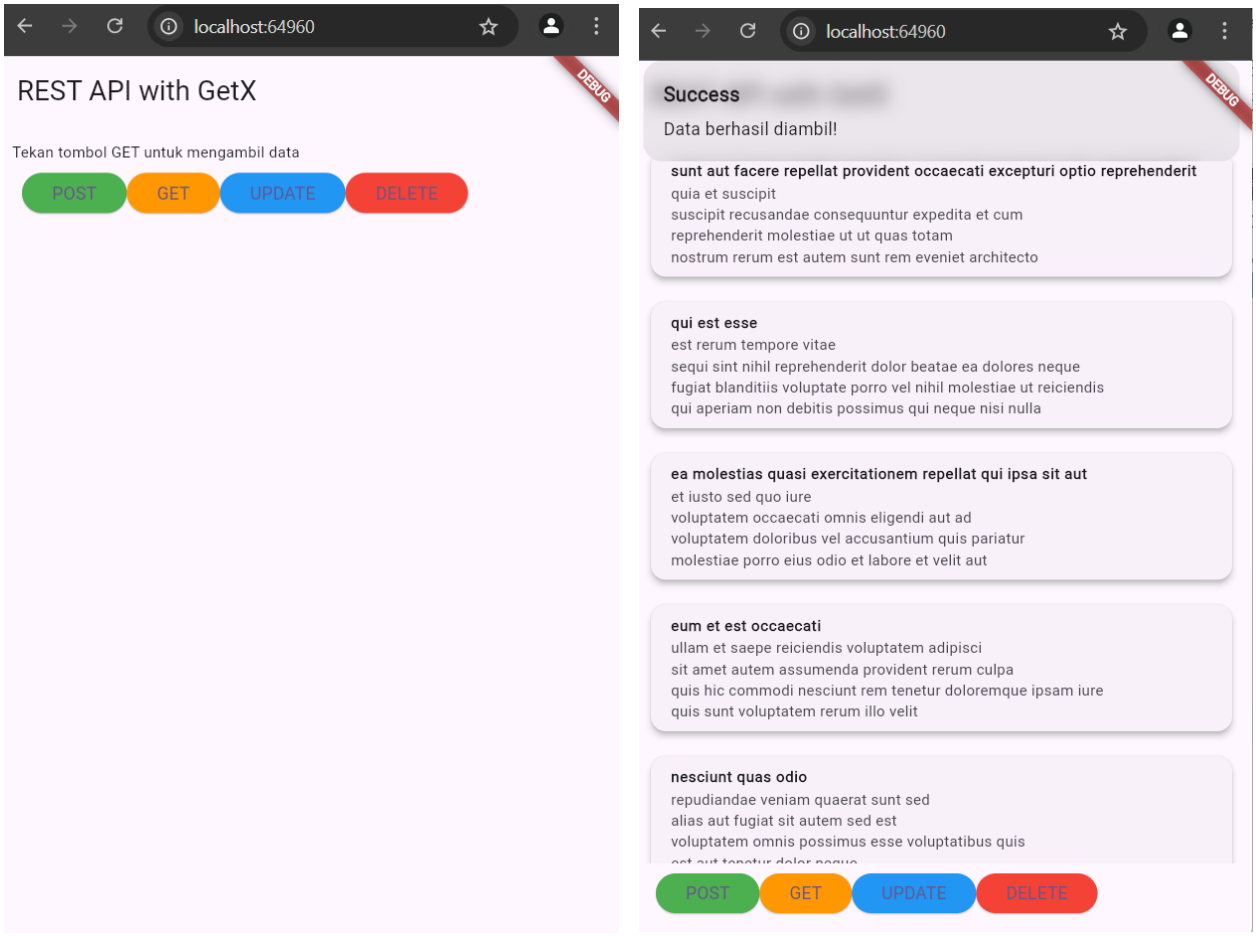
```dart
import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
  List<dynamic> posts = [];
  Future<void> fetchPosts() async {
    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');
    }
  }

  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': 'Ini contoh POST.',
        'userId': 1,
      }),
    );
    if (response.statusCode == 201) {
      posts.add({
        'title': 'Flutter Post',
        'body': 'Ini contoh POST.',
        'id': posts.length + 1,
      });
    } else {
      throw Exception('Failed to create post');
    }
  }

  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');
    }
  }

  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');
    }
  }
}
```

Penjelasan:

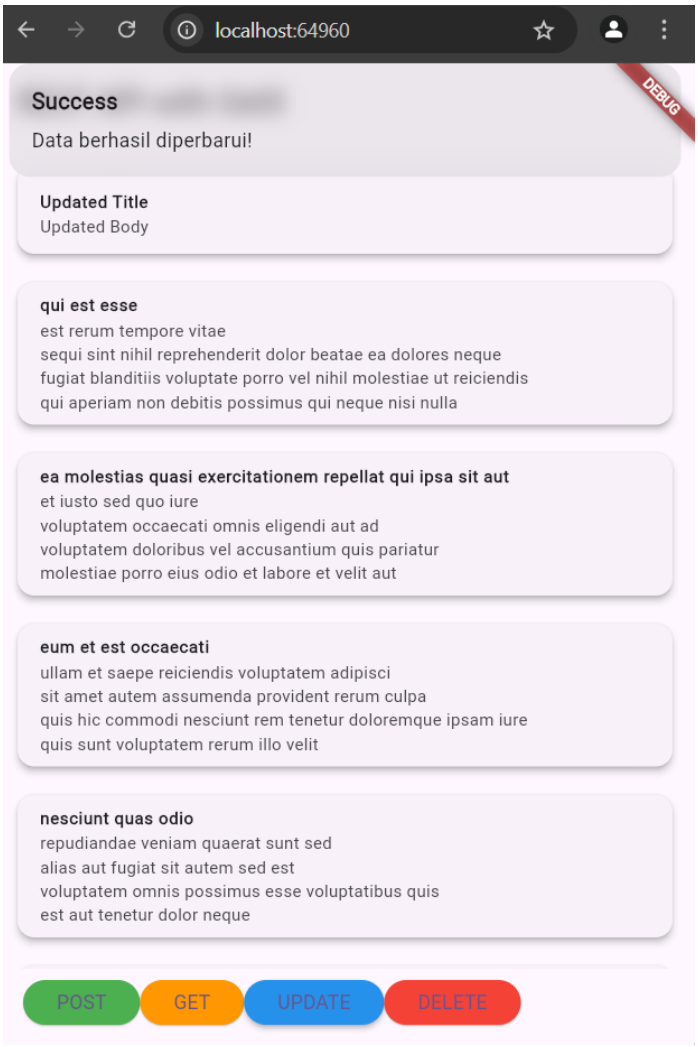
Kode ini mendefinisikan kelas ApiService dalam Dart yang menggunakan library http untuk berinteraksi dengan API JSON Placeholder. Kelas ini menyediakan fungsi untuk operasi CRUD (Create, Read, Update, Delete) pada postingan. Fungsi fetchPosts mengambil data dari endpoint /posts dan memuatnya ke dalam list posts jika berhasil. Fungsi createPost mengirimkan data baru ke API dan menambahkannya ke list jika berhasil. Fungsi updatePost memperbarui postingan dengan ID tertentu dan mengubah data di list jika berhasil. Akhirnya, fungsi deletePost

menghapus postingan berdasarkan ID dan mengeluarkannya dari list jika berhasil. Setiap fungsi mengelola status respons dan melempar kesalahan jika operasi gagal.

Hasil output:




localhost:64960
DEBUG
Success
Data berhasil ditambahkan!
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
voluptatem omnis possimus esse voluptatibus quis
POST
GET
UPDATE
DELETE


localhost:64960
DEBUG
Success
Data berhasil diperbarui!
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
POST
GET
UPDATE
DELETE


localhost:64960
DEBUG
Success
Data berhasil dihapus!
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
dolorem eum magni eos aperiam quia
ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae
perspiciatis 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
POST
GET
UPDATE
DELETE