LAPORAN GUIDED & UNGUIDED

PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIV

DATA STORAGE

Disusun Oleh :

Ahmad Junaidi / 2211104002

SE-06-01

Asisten Praktikum :

Ayu Susilowati

Noviana Rizki Anisa Putri

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

GUIDED

Souce Code main.dart:

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

void main() {
  runApp(MaterialApp(
    home: HomeScreen(),
    debugShowCheckedModeBanner: false,
  ));
}
```

Souce Code home_screen.dart:

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

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

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

  // Fungsi untuk menampilkan SnackBar
  void _showSnackBar(String message) {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text(message)));
  }

  // Fungsi untuk menangani operasi API
  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);
   }
  }

  Widget _buildButton(String label, Color color, Function onPressed) {
   return ElevatedButton(
    onPressed: () => onPressed(),
    style: ElevatedButton.styleFrom(
     backgroundColor: color,
     shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),
     padding: EdgeInsets.symmetric(vertical: 12, horizontal: 24),
    ),
    child: Text(
     label,
     style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
    ),
   );
  }

  Widget _buildCard(String title, String body) {
   return Card(
    margin: EdgeInsets.symmetric(vertical: 10, horizontal: 16),
    elevation: 4,
    shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12)),
    child: Padding(
     padding: const EdgeInsets.all(16.0),
     child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
       Text(
        title,
        style: TextStyle(
         fontSize: 18,
         fontWeight: FontWeight.bold,
         color: Colors.black87,
        ),
       ),
       SizedBox(height: 8),
       Text(
        body,
        style: TextStyle(fontSize: 14, color: Colors.black54),
```

```
          ),
        ],
      ),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(
        'HTTP Request Example',
        style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
      ),
      backgroundColor: Colors.blueAccent,
      centerTitle: true,
    ),
    body: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(height: 10),
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 16.0),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              _buildButton('GET', Colors.orange, () {
                _handleApiOperation(
                    _apiService.fetchPosts(), 'Data berhasil diambil!');
              }),
              _buildButton('POST', Colors.green, () {
                _handleApiOperation(
                    _apiService.createPost(), 'Data berhasil ditambahkan!');
              }),
              _buildButton('UPDATE', Colors.blue, () {
                _handleApiOperation(
                    _apiService.updatePost(), 'Data berhasil diperbarui!');
              }),
              _buildButton('DELETE', Colors.red, () {
                _handleApiOperation(
                    _apiService.deletePost(), 'Data berhasil dihapus!');
              }),
            ],
          ),
```

```
        ),
        SizedBox(height: 10),
        Divider(color: Colors.grey),
        Expanded(
          child: _isLoading
              ? Center(child: CircularProgressIndicator())
              : _posts.isEmpty
                  ? Center(
                      child: Text(
                        "Tekan tombol GET untuk mengambil data",
                        style: TextStyle(fontSize: 16),
                      ),
                    )
                  : ListView.builder(
                      itemCount: _posts.length,
                      itemBuilder: (context, index) => _buildCard(
                        _posts[index]['title'],
                        _posts[index]['body'],
                      ),
                    ),
        ),
      ],
    ),
  );
 }
}
```

Souce api_serfice main.dart:

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

class ApiService {
 final String baseUrl = "https://jsonplaceholder.typicode.com";
 List<dynamic> posts = [];

 // HTTP GET
 Future<void> fetchPosts() async {
  final response = await http.get(Uri.parse('$baseUrl/posts'));

  if (response.statusCode == 200) {
   posts = json.decode(response.body);
  } else {
```

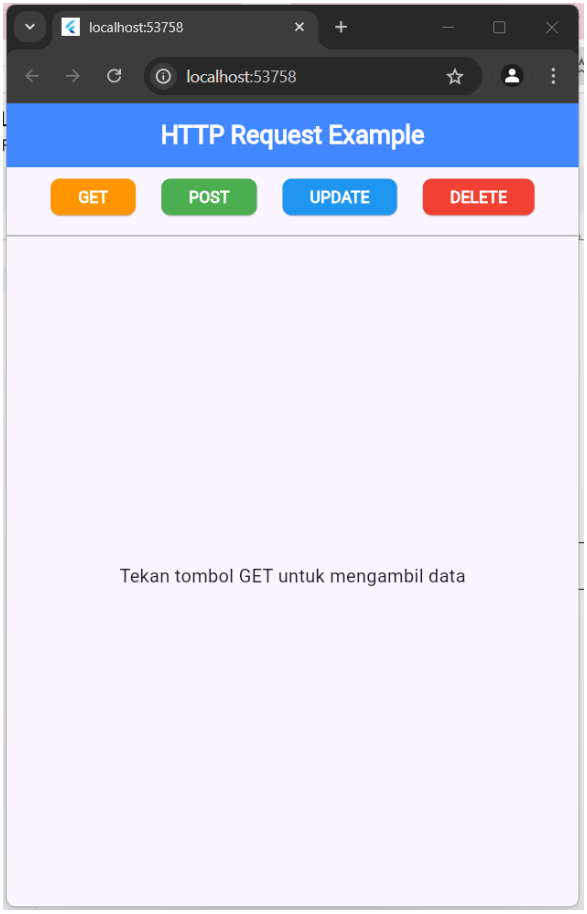
```
    throw Exception('Failed to load posts');
  }
}

// HTTP POST
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': 'Contoh POST', 'userId': 1}),
  );

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

// HTTP PUT
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');
  }
}

// HTTP DELETE
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');
    }
  }
}
```

Souce Code pubspec.yaml:

```
version: 1.0.0+1

environment:
  sdk: ^3.5.4

dependencies:
  flutter:
    sdk: flutter
 cupertino_icons: ^1.0.8
  http: ^1.2.2
  get: ^4.6.6

dev_dependencies:
  flutter_test:
    sdk: flutter


 flutter_lints: ^4.0.0
```

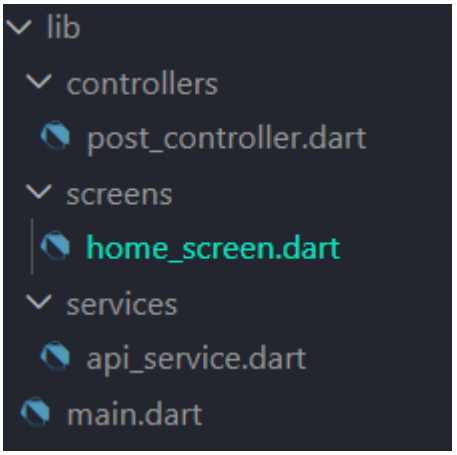
Output GUIDED:


localhost:53758
localhost:53758
HTTP Request Example
GET
POST
UPDATE
DELETE
Tekan tombol GET untuk mengambil data


localhost:64189
HTTP Request Example
GET
POST
UPDATE
DELETE
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 dihapus!

UNGUIDED

Struktur File:



Souce Code main.dart:

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul_14/screens/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 GetX Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const HomeScreen(),
    );
  }
}
```

Souce home_screen.dart:

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul_14/controllers/post_controller.dart';

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

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

    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example with GetX'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        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 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),
                          ),
                        ),
                      );
                    },
                  ),
                )),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: controller.fetchPosts,
            style: ElevatedButton.styleFrom(backgroundColor:
Colors.orange),
            child: const Text('GET'),
          ),
          ElevatedButton(
            onPressed: controller.createPost,
            style: ElevatedButton.styleFrom(backgroundColor:
Colors.green),
            child: const Text('POST'),
          ),
          ElevatedButton(
            onPressed: controller.updatePost,
            style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
            child: const Text('UPDATE'),
          ),
          ElevatedButton(
            onPressed: controller.deletePost,
            style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
            child: const Text('DELETE'),
          ),
        ],
      ),
    ),
  );
 }
}
```

Souce post_controller.dart:

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

```
class ApiController extends GetxController {
  final String baseUrl = "https://jsonplaceholder.typicode.com";

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

  // Snackbar helper
  void showSuccessSnackBar(String message) {
    Get.snackbar(
      'Success',
      message,
      backgroundColor: Colors.green,
      colorText: Colors.white,
      snackPosition: SnackPosition.BOTTOM,
      duration: const Duration(seconds: 2),
    );
  }

  void showErrorSnackBar(String message) {
    Get.snackbar(
      'Error',
      message,
      backgroundColor: Colors.red,
      colorText: Colors.white,
      snackPosition: SnackPosition.BOTTOM,
      duration: const Duration(seconds: 2),
    );
  }

  // GET Posts
  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      final response = await http.get(Uri.parse('$baseUrl/posts'));
      if (response.statusCode == 200) {
        posts.value = json.decode(response.body);
        showSuccessSnackBar('Data berhasil diambil!');
      } else {
        throw Exception('Failed to load posts');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
```

```
  }
}

// POST Data
Future<void> createPost() async {
  isLoading.value = true;
  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) {
      posts.add({
        'title': 'Flutter Post',
        'body': 'Ini contoh POST.',
        'id': posts.length + 1,
      });
      showSuccessSnackBar('Data berhasil ditambahkan!');
    } else {
      throw Exception('Failed to create post');
    }
  } catch (e) {
    showErrorSnackBar('Error: $e');
  } finally {
    isLoading.value = false;
  }
}

// UPDATE Data
Future<void> updatePost() async {
  isLoading.value = true;
  try {
    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) {
      var updatedPost = posts.firstWhere((post) => post['id'] == 1);
      updatedPost['title'] = 'Updated Title';
      updatedPost['body'] = 'Updated Body';
      showSuccessSnackBar('Data berhasil diperbarui!');
    } else {
      throw Exception('Failed to update post');
    }
  } catch (e) {
    showErrorSnackBar('Error: $e');
  } finally {
    isLoading.value = false;
  }
 }

 // DELETE Data
 Future<void> deletePost() async {
  isLoading.value = true;
  try {
    final response = await http.delete(Uri.parse('$baseUrl/posts/1'));
    if (response.statusCode == 200) {
      posts.removeWhere((post) => post['id'] == 1);
      showSuccessSnackBar('Data berhasil dihapus!');
    } else {
      throw Exception('Failed to delete post');
    }
  } catch (e) {
    showErrorSnackBar('Error: $e');
  } finally {
    isLoading.value = false;
  }
 }
}
```

Souce Code 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 = [];
```

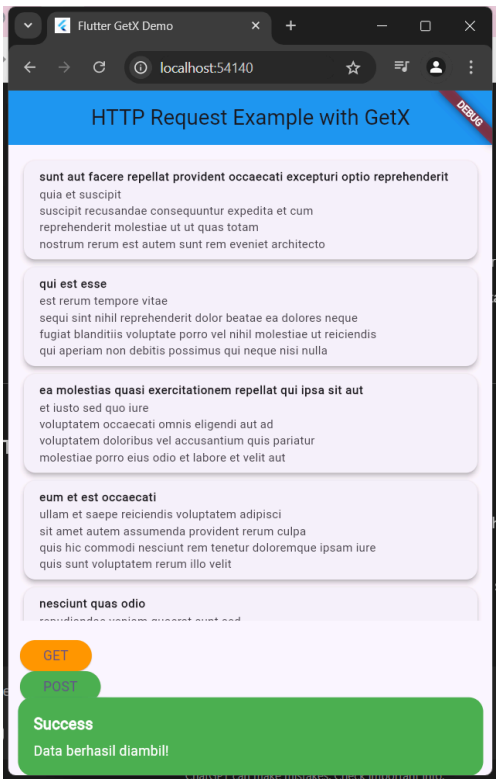
```
// HTTP GET
Future<void> fetchPosts() async {
  final response = await http.get(Uri.parse('$baseUrl/posts'));

  if (response.statusCode == 200) {
    posts = json.decode(response.body);
  } else {
    throw Exception('Failed to load posts');
  }
}

// HTTP POST
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': 'Contoh POST', 'userId': 1}),
  );

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

// HTTP PUT
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');
```

```
  }
 }

 // HTTP DELETE
 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');
  }
 }
}
```

Output Programs UNGUIDED

Tampilan Awal:

Fitur get



penjelasan get:

Fungsi:

- Tombol ini digunakan untuk mengambil data dari server melalui HTTP GET request.
- Menggunakan `ApiService.fetchPosts()` di dalam `PostController` untuk mendapatkan data dari API.

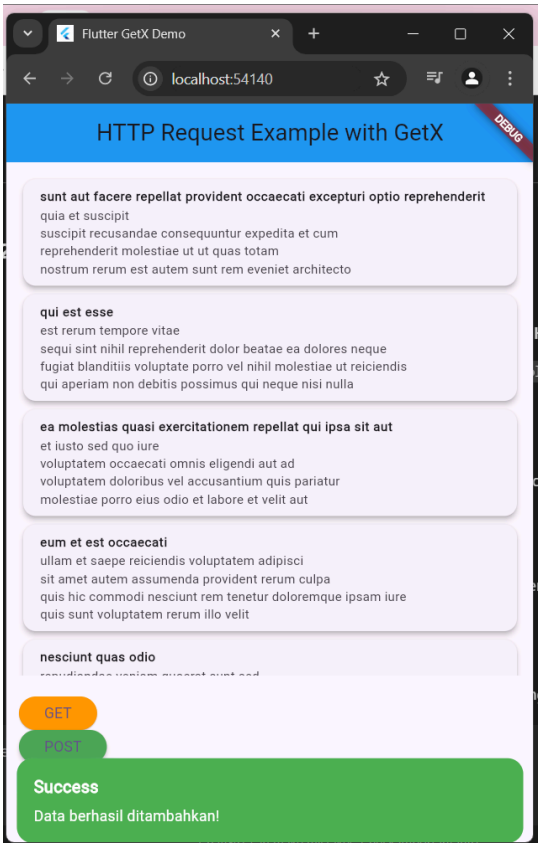
  Tampilan Hasil:

- Ketika tombol GET ditekan, data dari API akan ditampilkan dalam bentuk ListView.
- Jika data berhasil diambil, akan muncul Snackbar berisi pesan: "Berhasil: Data berhasil diambil".
- Jika gagal, muncul Snackbar dengan warna merah: "Error: Gagal mengambil data".

  Penggunaan State Management:

- `posts.assignAll()` digunakan untuk menyimpan data ke dalam state GetX, sehingga tampilan ListView akan diperbarui otomatis dengan Obx.

Fitur POST



Penjelasan POST:

Fungsi:

- Tombol ini digunakan untuk menambah data ke server melalui HTTP POST request.
- Menggunakan `ApiService.createPost()` di dalam `PostController` untuk menambahkan data baru.

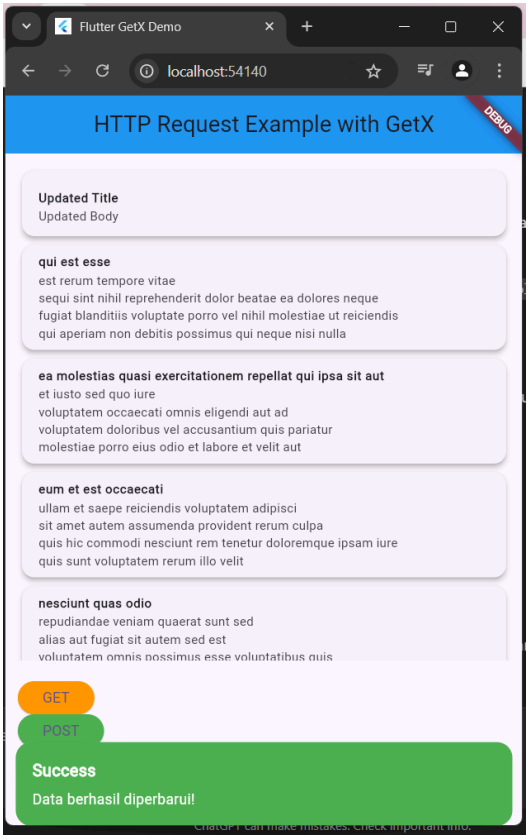
Tampilan Hasil:

- Setelah tombol POST ditekan, data baru akan ditambahkan ke daftar ListView.
- Snackbar sukses akan muncul dengan pesan:
  "Berhasil: Data berhasil ditambahkan".
- Jika terjadi kesalahan, akan muncul Snackbar dengan warna merah:
  "Error: Gagal menambah data".

Penjelasan Teknis:

- Setelah data ditambahkan, fungsi `fetchPosts()` dipanggil ulang agar tampilan diperbarui dengan data terbaru.
- State `posts` diperbarui otomatis berkat penggunaan GetX.

Fitur PUT/UPDATE



Penjelasan Update:

Fungsi:

- Tombol ini digunakan untuk memperbarui data yang sudah ada di server menggunakan HTTP PUT request.
- Menggunakan `ApiService.updatePost()` di dalam `PostController` untuk mengubah data.

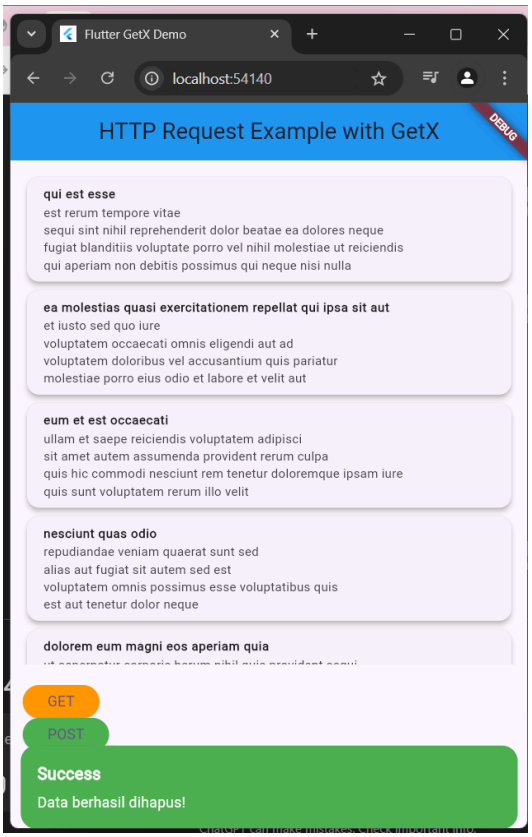
Tampilan Hasil:

- Setelah tombol PUT ditekan, data pada ListView akan diperbarui.
- Snackbar sukses akan muncul dengan pesan:
  "Berhasil: Data berhasil diperbarui".
- Jika ada kesalahan, muncul Snackbar dengan warna merah:
  "Error: Gagal memperbarui data".

Penjelasan Teknis:

- Data dengan ID tertentu akan diperbarui di server.
- Setelah berhasil, fetchPosts() dipanggil ulang untuk memperbarui state GetX.

Fitur DELETE



Penjelasan DELETE:

- Fungsi:
  - Tombol ini digunakan untuk menghapus data dari server melalui HTTP DELETE request.
  - Menggunakan `ApiService.deletePost()` di dalam `PostController` untuk menghapus data.
- Tampilan Hasil:
  - Setelah tombol DELETE ditekan, data dengan ID tertentu akan dihapus.
  - Data yang dihapus tidak akan lagi muncul di ListView.
  - Snackbar sukses akan muncul dengan pesan: "Berhasil: Data berhasil dihapus".
  - Jika ada kesalahan, muncul Snackbar dengan warna merah: "Error: Gagal menghapus data".
- Penjelasan Teknis:
  - Fungsi ini memanggil `deletePost()` yang menghapus data dari server.
  - Setelah sukses, data diperbarui melalui fetchPosts(), sehingga tampilan ListView ikut diperbarui otomatis.

Deskripsi

Aplikasi ini adalah CRUD sederhana (Create, Read, Update, Delete) yang menggunakan State Management GetX untuk mengelola data secara efisien. Aplikasi ini berinteraksi dengan API eksternal untuk mengambil, menambah, memperbarui, dan menghapus data. Dengan dukungan Snackbar, pengguna mendapatkan respon instan setelah setiap operasi berhasil atau gagal, sehingga memberikan pengalaman pengguna yang lebih baik dan informatif.

Aplikasi ini cocok untuk memahami konsep dasar integrasi API dengan Flutter, penerapan GetX sebagai state management, dan feedback notifikasi menggunakan Snackbar.

Fitur Utama Aplkasi

GET Data (Read)

- Mengambil data dari API dan menampilkannya dalam bentuk ListView.
- Data diperbarui secara otomatis berkat penggunaan GetX State Management.

POST Data (Create)

- Menambahkan data baru ke server menggunakan HTTP POST request.
- Data baru akan langsung tampil di ListView setelah berhasil ditambahkan.
- Menampilkan Snackbar sukses sebagai notifikasi umpan balik.

PUT Data (Update)

- Memperbarui data yang ada di server menggunakan HTTP PUT request.
- Data yang diperbarui akan ditampilkan di ListView secara real-time.
- Menampilkan Snackbar sukses atau error jika operasi gagal.

DELETE Data (Delete)

- Menghapus data dari server menggunakan HTTP DELETE request.
- Data yang dihapus akan langsung hilang dari ListView.
- Menampilkan Snackbar sukses atau error untuk memberikan notifikasi hasil operasi.

Snackbar Notifikasi

- Menggunakan `Get.snackbar` untuk memberikan notifikasi instan.
- Snackbar akan muncul saat operasi berhasil (warna hijau) atau gagal (warna merah).

Cara Kerja Aplikasi

1. **State Management dengan GetX**

- **PostController**:
    - Mengelola data dari API menggunakan `ApiService` (GET, POST, PUT, DELETE).
    - Menyimpan data di dalam state **`RxList`** (`posts`) agar perubahan data langsung mempengaruhi tampilan.
    - State dikelola menggunakan **`Obx`**, sehingga tampilan diperbarui otomatis

2. Operasi CRUD

   GET:

   - Memanggil `ApiService.fetchPosts()` untuk mengambil data dari API.
   - Data ditampilkan dalam ListView.

   POST:

   - Memanggil `ApiService.createPost()` untuk menambahkan data baru.
   - Data akan langsung ditambahkan ke dalam state `posts` dan ditampilkan.

   PUT:

   - Memanggil `ApiService.updatePost()` untuk memperbarui data berdasarkan ID.
   - State diperbarui agar tampilan otomatis merefleksikan perubahan.

   DELETE:

   - Memanggil `ApiService.deletePost()` untuk menghapus data berdasarkan ID.
   - Data dihapus dari state `posts` dan ListView diperbarui.

3. snackbar Notifikasi

   Setelah setiap operasi (GET, POST, PUT, DELETE), Snackbar akan ditampilkan menggunakan `Get.snackbar`:

   - Sukses: Menampilkan pesan positif.
   - Error: Menampilkan pesan dengan background merah.

4. interaksi Pengguna:

   Pengguna menekan tombol GET, POST, PUT, atau DELETE pada UI.

   Tombol memanggil fungsi di PostController, yang berinteraksi dengan API melalui `ApiService`.

   Data akan ditampilkan atau diperbarui di layar berkat GetX State Management.

   Respon sukses atau error ditampilkan melalui Snackbar untuk memberikan feedback langsung kepada pengguna.