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

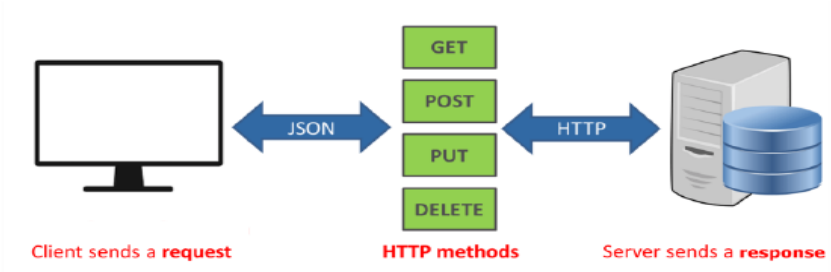
FAKULTAS INFORMATIKA

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 Projek 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.

```yaml
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:

```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) as List<dynamic>;
      } else {
        throw Exception(
            'Failed to load posts. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to fetch posts. Error: $e');
    }
  }
```

   b. Membuat tampilan UI untuk GET
   Buat file home_screen.dart di dalam folder screens:
   Fungsi untuk memanggil file api service

```dart
class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  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;
      });
      _showSnackBar(successMessage);
    } catch (e) {
      _showSnackBar('Error: $e');
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }
```

Menampilkan response API

```dart
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          if (_isLoading)
            const Center(child: CircularProgressIndicator())
          else if (_posts.isEmpty)
            const Text(
              "Tekan tombol GET untuk mengambil data",
              style: TextStyle(fontSize: 16),
            ) // Text
          else
            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: 16), // TextStyle
                        ), // Text
                        subtitle: Text(
                          _posts[index]['body'],
                          style: const TextStyle(fontSize: 14),
                        ), // Text
                      ), // ListTile
                    ), // Card
                  ); // Padding
                },
              ), // ListView.builder
            ), // 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            ElevatedButton(
97              onPressed: () => _handleApiOperation(
98                  _apiService.createPost(), 'Data berhasil ditambahkan!'),
99              style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
100             child: const Text('POST'),
101           ), // ElevatedButton
102           const SizedBox(height: 10),
103           ElevatedButton(
```

5. Implementasi HTTP UPDATE
   a. Membuat Service UPDATE

   Tambahkan metode berikut ke api_service.dart:

```dart
// Fungsi untuk UPDATE data
Future<void> updatePost(int postId) async {
  try {
    final response = await http.put(
      Uri.parse('$baseUrl/posts/$postId'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode({
        'title': 'Updated Title',
        'body': 'Updated Body',
        'userId': 1,
      }),
    );
    if (response.statusCode == 200) {
      final updatedPostIndex =
          posts.indexWhere((post) => post['id'] == postId);
      if (updatedPostIndex != -1) {
        posts[updatedPostIndex] = {
          'id': postId,
          'title': 'Updated Title',
          'body': 'Updated Body',
          'userId': 1,
        };
      } else {
        throw Exception('Post with id $postId not found in local data');
      }
    } else {
      throw Exception(
          'Failed to update post. Status code: ${response.statusCode}');
    }
  } catch (e) {
    throw Exception('Failed to update post. Error: $e');
  }
}
```

b. Membuat tampilan UI untuk UPDATE

Tambahkan logika untuk memperbarui postingan di home_screen.dart:

```dart
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.updatePost(1), 'Data berhasil diperbarui!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('UPDATE'),
            ), // ElevatedButton
```

6. Implementasi HTTP DELETE

a. Membuat Service DELETE

Tambahkan metode berikut ke api_service.dart:

```dart
  // 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) {
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception(
            'Failed to delete post. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to delete post. Error: $e');
    }
  }
}
```

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

```dart
import 'package:flutter/material.dart';
import 'package:guided/screens/home_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 HomeScreen(),
    );
  }
}
```

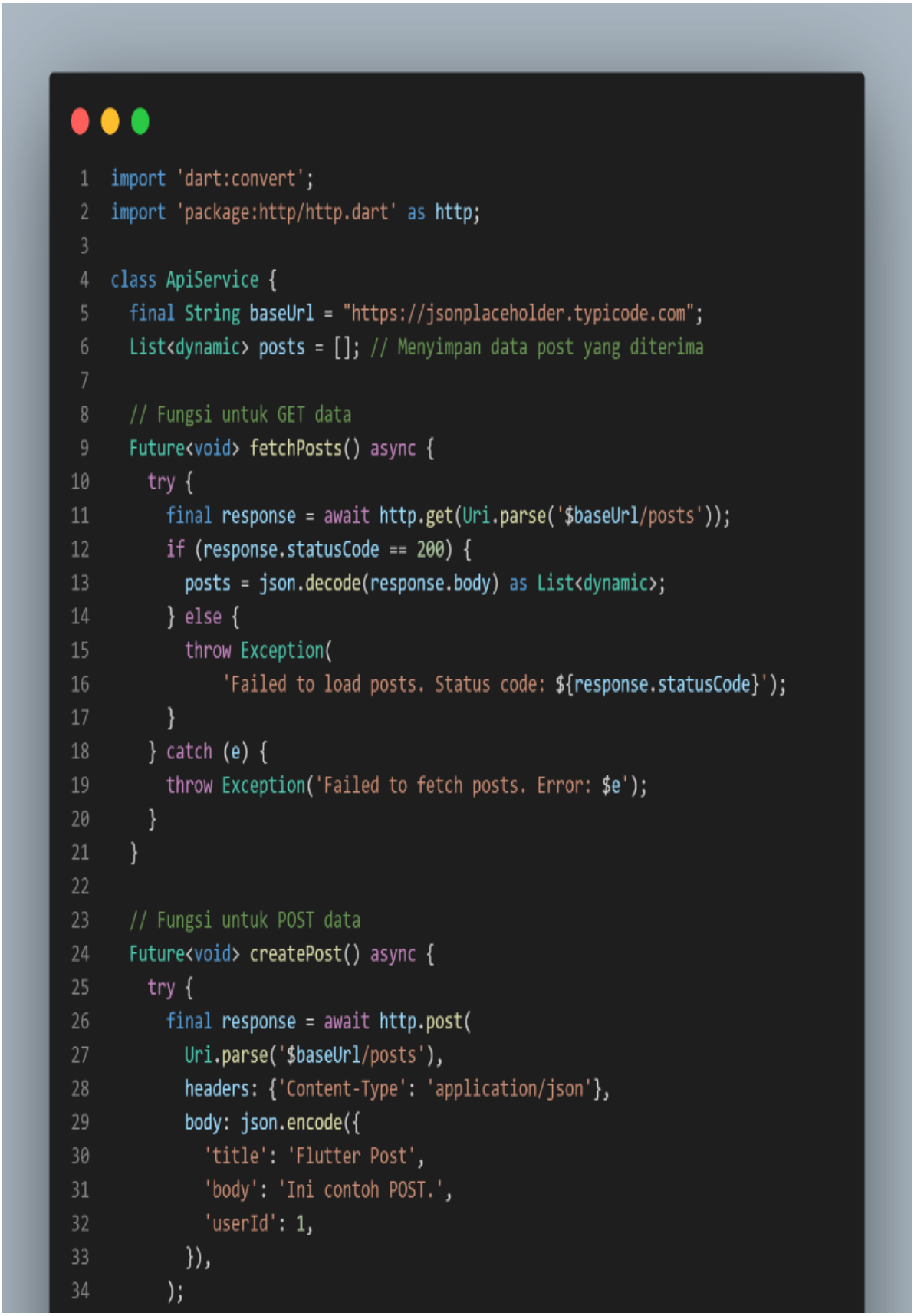
Source code home_screen

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

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

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  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;
      });
      _showSnackBar(successMessage);
    } catch (e) {
      _showSnackBar('Error: $e');
    } finally {
      setState(() {
        _isLoading = false;
      });
    }
  }

```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('HTTP Request Example'),
      centerTitle: true,
      backgroundColor: const Color.fromARGB(255, 222, 135, 240),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          if (_isLoading)
            const Center(child: CircularProgressIndicator())
          else if (_posts.isEmpty)
            const Text(
              "Tekan tombol GET untuk mengambil data",
              style: TextStyle(fontSize: 16),
            )
          else
            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: 16),
                        ),
                        subtitle: Text(
                          _posts[index]['body'],
                          style: const TextStyle(fontSize: 14),
```

```
                        ),
                      ),
                    ),
                  );
                },
              ),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.fetchPosts(), 'Data berhasil diambil!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
              child: const Text('GET'),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.createPost(), 'Data berhasil ditambahkan!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
              child: const Text('POST'),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.updatePost(1), 'Data berhasil diperbarui!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('UPDATE'),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.deletePost(1), 'Data berhasil dihapus!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
              child: const Text('DELETE'),
            ),
          ],
        ),
      ),
    );
  }
}

```

Source code api_service

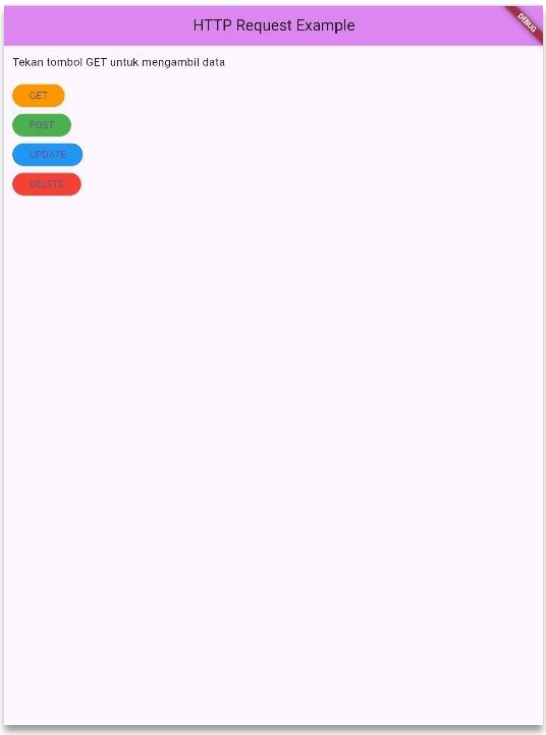


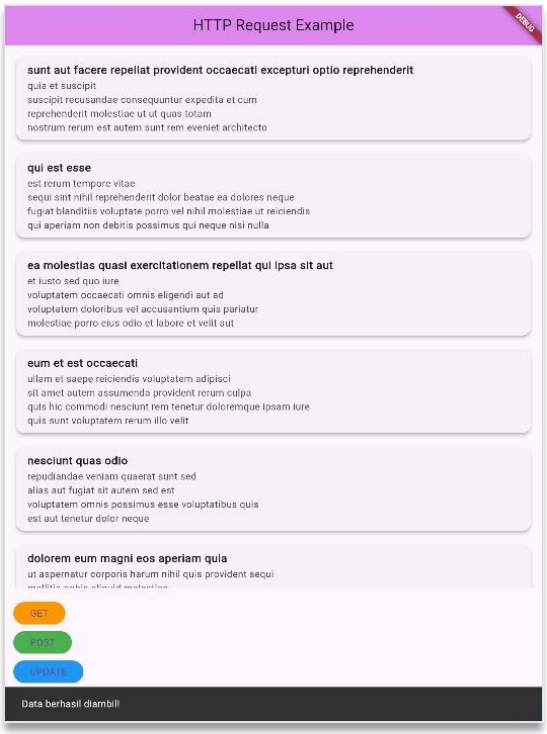
```dart
import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";
  List<dynamic> posts = []; // 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) as List<dynamic>;
      } else {
        throw Exception(
            'Failed to load posts. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to fetch posts. Error: $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,
        }),
      );
```

```dart
      if (response.statusCode == 201) {
        final newPost = json.decode(response.body);
        posts.add(newPost); // Menambahkan data baru ke daftar posts
      } else {
        throw Exception(
            'Failed to create post. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to create post. Error: $e');
    }
  }

  // Fungsi untuk UPDATE data
  Future<void> updatePost(int postId) async {
    try {
      final response = await http.put(
        Uri.parse('$baseUrl/posts/$postId'),
        headers: {'Content-Type': 'application/json'},
        body: json.encode({
          'title': 'Updated Title',
          'body': 'Updated Body',
          'userId': 1,
        }),
      );
      if (response.statusCode == 200) {
        final updatedPostIndex =
            posts.indexWhere((post) => post['id'] == postId);
        if (updatedPostIndex != -1) {
          posts[updatedPostIndex] = {
            'id': postId,
            'title': 'Updated Title',
            'body': 'Updated Body',
            'userId': 1,
          };
        } else {
          throw Exception('Post with id $postId not found in local data');
        }
      } else {
        throw Exception(
            'Failed to update post. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to update post. Error: $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) {
        posts.removeWhere((post) => post['id'] == postId);
      } else {
        throw Exception(
            'Failed to delete post. Status code: ${response.statusCode}');
      }
    } catch (e) {
      throw Exception('Failed to delete post. Error: $e');
    }
  }
}
```

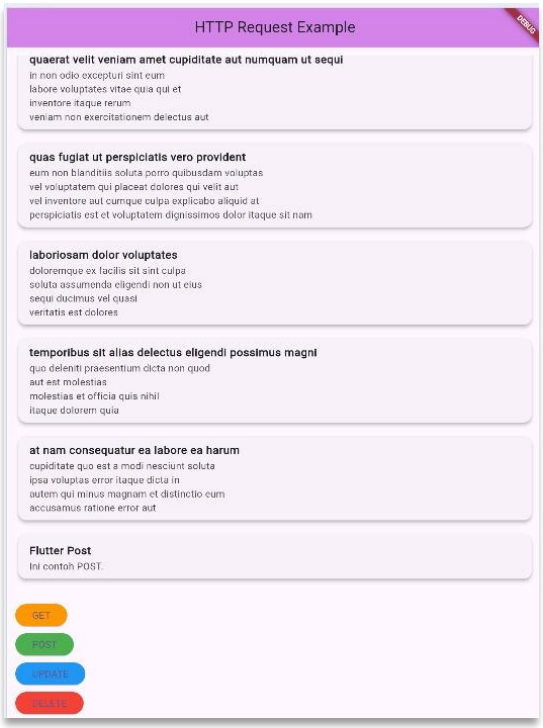
2. Screenshot output

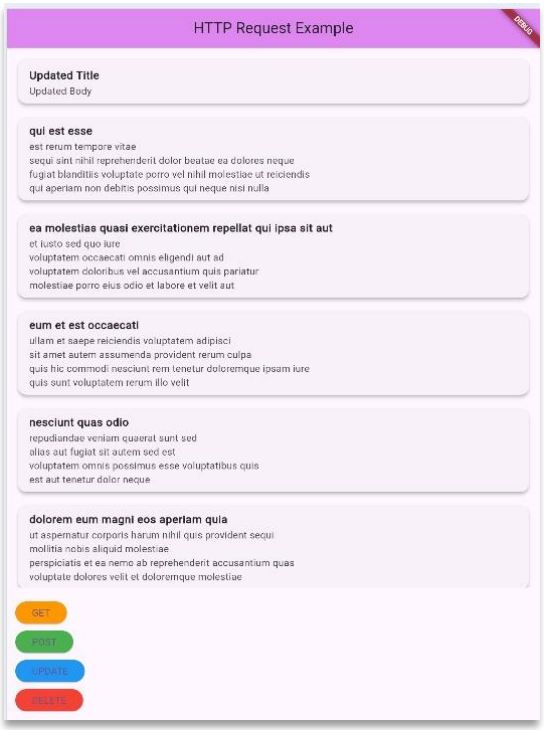
Tampilan awal saat aplikasi di run dan saat mengklik tombol get



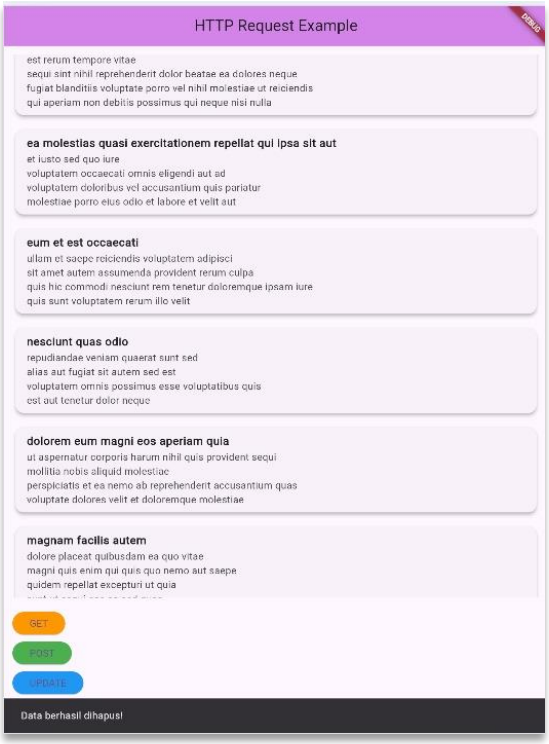


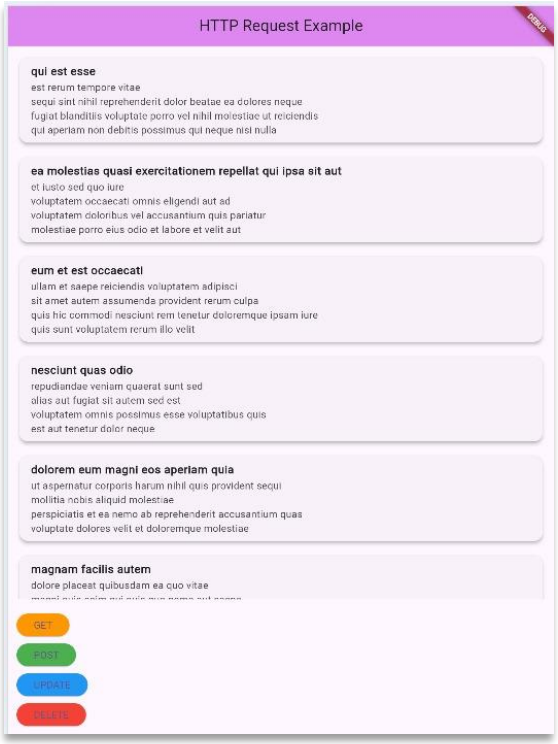
Tampilan saat mengklik tombol post dan saat mengklik tombol update





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, screenshoot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.

Jawab

1. Source code

- Source code main.dart

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:unguided/screen/homescreen.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(),
    );
  }
}
```

- Source code homescreen.dart

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:unguided/controller/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('Data Storage API'),
        centerTitle: true,
        backgroundColor: const Color.fromARGB(255, 210, 233, 255),
        foregroundColor: const Color.fromARGB(255, 20, 20, 20),
      ),
      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(
                        "Untuk mengambil data silahkan tekan tombol GET",
                        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),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                ElevatedButton(
                  onPressed: controller.fetchPosts,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 230, 188, 255),
                    foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
                  ),
                  child: const Text('GET'),
                ),
                ElevatedButton(
                  onPressed: controller.createPost,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 255, 232, 163),
                    foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
                  ),
                  child: const Text('POST'),
                ),
                ElevatedButton(
                  onPressed: controller.updatePost,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 209, 255, 148),
                    foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
                  ),
                  child: const Text('UPDATE'),
                ),
                ElevatedButton(
                  onPressed: controller.deletePost,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color.fromARGB(255, 255, 188, 214),
                    foregroundColor: const Color.fromARGB(255, 20, 20, 20), // Warna teks hitam
                  ),
                  child: const Text('DELETE'),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}

```

- Source code controller.dart

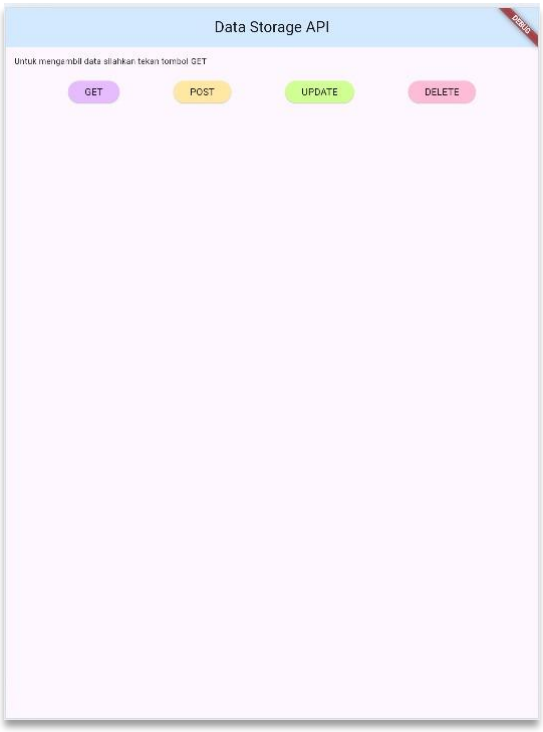
```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:http/http.dart' as http;

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: const Color.fromARGB(255, 247, 255, 140),
      colorText: const Color.fromARGB(255, 20, 20, 20),
      snackPosition: SnackPosition.TOP, // Changed to TOP
      duration: const Duration(seconds: 2),
    );
  }

  void showErrorSnackBar(String message) {
    Get.snackbar(
      'Error',
      message,
      backgroundColor: const Color.fromARGB(255, 255, 104, 74),
      colorText: const Color.fromARGB(255, 20, 20, 20),
      snackPosition: SnackPosition.TOP, // Changed to TOP
      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('Berhasil mengambil data!');
      } else {
        throw Exception('Failed to load posts');
      }
    } catch (e) {
      showErrorSnackBar('Error: $e');
    } finally {
      isLoading.value = false;
    }
  }
```

```dart

  // 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('Yeay data kamu 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 {
```

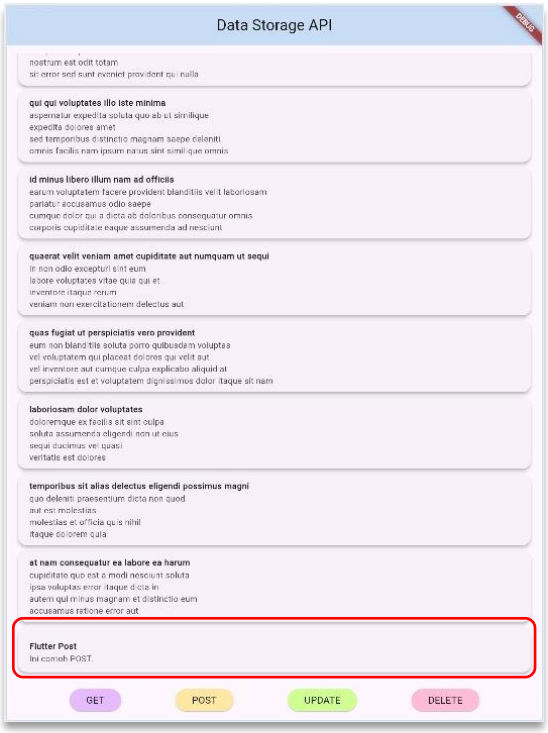
```dart
      final response = await http.put(
        Uri.parse('$baseUrl/posts/1'),
        body: json.encode({
          'title': 'Updated Title',
          'body': 'Updated Body',
          'userId': 1,
        }),
      );
      if (response.statusCode == 200) {
        var updatedPost = posts.firstWhere((post) => post['id'] == 1);
        updatedPost['title'] = 'Updated Title';
        updatedPost['body'] = 'Updated Body';
        showSuccessSnackBar('Kamu berhasil memperbarui data!');
      } 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;
    }
  }
}

```

2. Hasil screenshot output

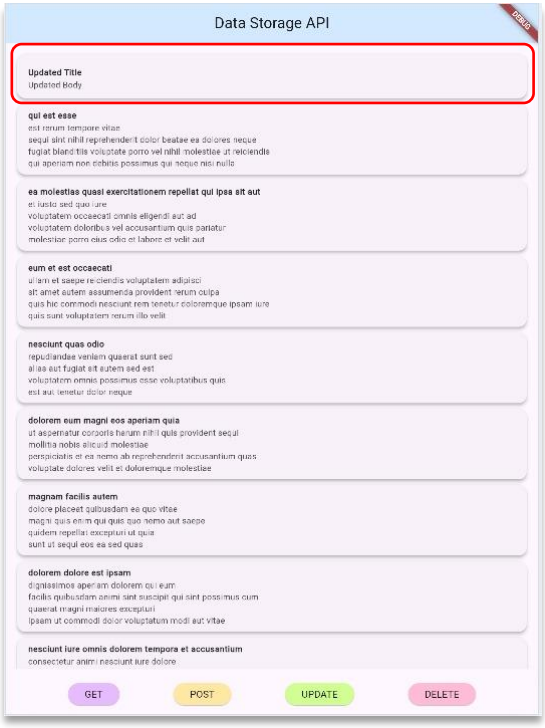
- Screenshot output tampilan awal dan ketika memilih tombol “GET”.



- 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.



- Screenshot output tampilan ketika memilih tombol "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.