# LAPORAN PRAKTIKUM
# PEMROGRAMAN PERANGKAT BERGERAK

## MODUL XIV

## DATA STORAGE API


**Disusun Oleh :**

**Devrin Anggun Saputri / 2211104001**

**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

## A. Apa itu 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. Apa itu 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

### 1. Persiapkan projek flutter

a. Buat proyek flutter baru

b. Tambahkan dependency http ke file pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  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

```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 {
    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');
    }
  }
```

b. Membuat tampilan UI untuk GET

buat file home_screen.dart di dalam folder screen

```dart
// 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;
      });
    }
  }
```

c. Menampilkan responses API

```
body: Padding(
  padding: const EdgeInsets.all(12.0),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      _isLoading
          ? const Center(child: CircularProgressIndicator())
          : _posts.isEmpty
              ? const Text(
                  "Tekan tombol GET untuk mengambil data",
                  style: TextStyle(fontSize: 12),
                ) // Text
              : 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: 12), // TextStyle
                            ), // Text
```

d. Tambahkan tombol untuk GET di home_screen.dart

```
ElevatedButton(
  onPressed: () => _handleApiOperation(
      _apiService.fetchPosts(), 'Data berhasil diambil!'),
  style: ElevatedButton.styleFrom(backgroundColor: Colors.orange),
  child: const Text('GET'),
), // ElevatedButton
```

**4. Implementasi HTTP POST**

a. Membuat service Post

```
// Fungsi untuk POST data
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');
  }
}
```

b. Membuat tampilan UI untuk POST

```
const SizedBox(height: 10),
ElevatedButton(
  onPressed: () => _handleApiOperation(
      _apiService.createPost(), 'Data berhasil ditambahkan!'),
  style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
  child: const Text('POST'),
), // ElevatedButton
```

**5. Implementasi HTTP PUT**

a. Membuat Service PUT

```
// Fungsi untuk UPDATE data
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');
  }
}
```

b. Membuat tampilan UI untuk PUT

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

**6. Implementasi HTTP DELETE**

a. Membuat service Delete

```
// Fungsi untuk DELETE data
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');
  }
}
}
```

b. Membuat tampilan UI untuk Delete

```
ElevatedButton(
  onPressed: () => _handleApiOperation(
      _apiService.deletePost(), 'Data berhasil dihapus!'),
  style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
  child: const Text('DELETE'),
), // ElevatedButton
```

**Source Code Praktikum:**

- **main.dart**

```dart
import 'package:flutter/material.dart';
import 'package:praktikum_14/screens/home_screen.dart';

Run | Debug | Profile
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,
      ), // ThemeData
      home: const HomeScreen(),
    ); // MaterialApp
  }
}
```

- **home_screen.dart**

```dart
import 'package:flutter/material.dart';
import 'package:praktikum_14/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;
      });
    }
  }

```

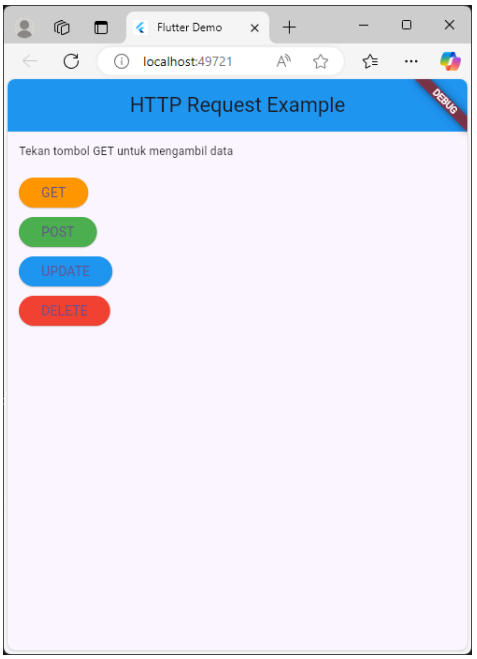
```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            _isLoading
                ? const Center(child: CircularProgressIndicator())
                : _posts.isEmpty
                    ? const Text(
                        "Tekan tombol GET untuk mengambil data",
                        style: TextStyle(fontSize: 12),
                      )
                    : 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: 12),
                                  ),
                                  subtitle: Text(
                                    _posts[index]['body'],
                                    style: const TextStyle(fontSize: 12),
                                  ),
                                ),
                              ),
                            );
                          },
                        ),
                      ),
            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(), 'Data berhasil diperbarui!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
              child: const Text('UPDATE'),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () => _handleApiOperation(
                  _apiService.deletePost(), 'Data berhasil dihapus!'),
              style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
              child: const Text('DELETE'),
            ),
          ],
        ),
      ),
    );
  }
}
```

- **api_service.dart**

```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 {
    final response = await http.get(Uri.parse('$baseUrl/posts'));
    if (response.statusCode == 200) {
      posts = json.decode(response.body);
    } else {
      throw Exception('Failed to load posts');
    }
  }
  // Fungsi untuk POST data
Future<void> createPost() async {
  final response = await http.post(
    Uri.parse('$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');
  }
}

// Fungsi untuk UPDATE data
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');
  }
}
```

```dart
// Fungsi untuk DELETE data
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');
  }
}
}
```

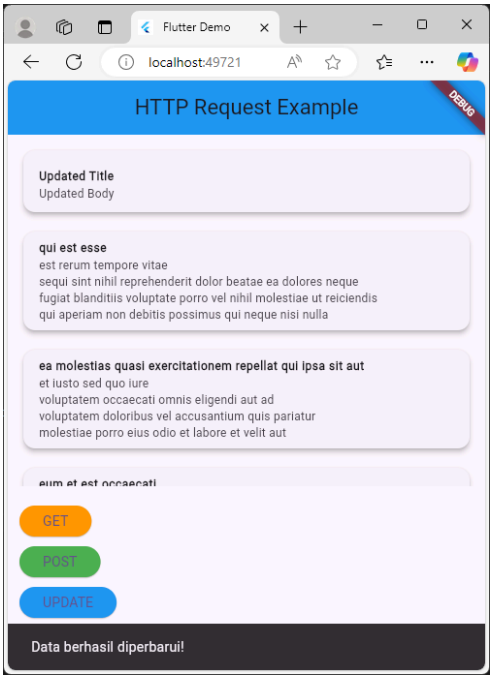
- **Output Hasil**
  1. **Halaman awal**




  2. **Get**

## 3. Post


## 4. Update



5. **Delete**


- **Deskripsi Program:**

  Program ini adalah aplikasi Flutter sederhana yang berfungsi untuk melakukan operasi CRUD (Create, Read, Update, Delete) melalui API menggunakan *package* HTTP. Aplikasi ini menampilkan empat tombol utama: **GET**, **POST**, **UPDATE**, dan **DELETE**, yang masing-masing mewakili operasi CRUD. Data diambil dari API publik "https://jsonplaceholder.typicode.com" dan ditampilkan dalam bentuk daftar pada layar utama. Setiap tombol akan memicu operasi API, seperti mengambil data (GET), menambah data baru (POST), memperbarui data yang sudah ada (UPDATE), atau menghapus data (DELETE). Program ini memiliki indikator *loading* yang muncul saat operasi sedang berlangsung dan akan menampilkan pesan sukses atau error melalui *SnackBar*. Aplikasi ini menggunakan *state management* sederhana dengan setState untuk memperbarui UI sesuai perubahan data yang terjadi.

# 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.*

- **Source Code:**

**main.dart**

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:unguided_14/screens/home_screen.dart';

Run | Debug | Profile
void main() {
  runApp(app: 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,
      ), // ThemeData
      home: const HomeScreen(),
    ); // GetMaterialApp
  }
}

```

**home_screen.dart**

```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:unguided_14/controllers/post_controllers.dart';

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

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

```

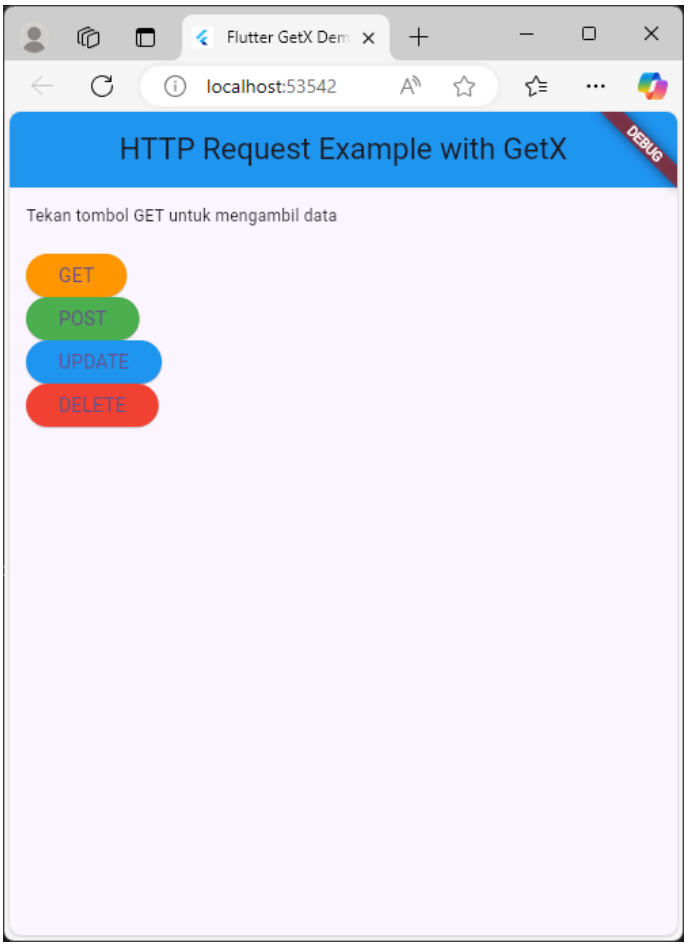
```
    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example with GetX'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: 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'),
            ),
          ],
        ),
      ),
    );
  }
}
```

**post_controller.dart**

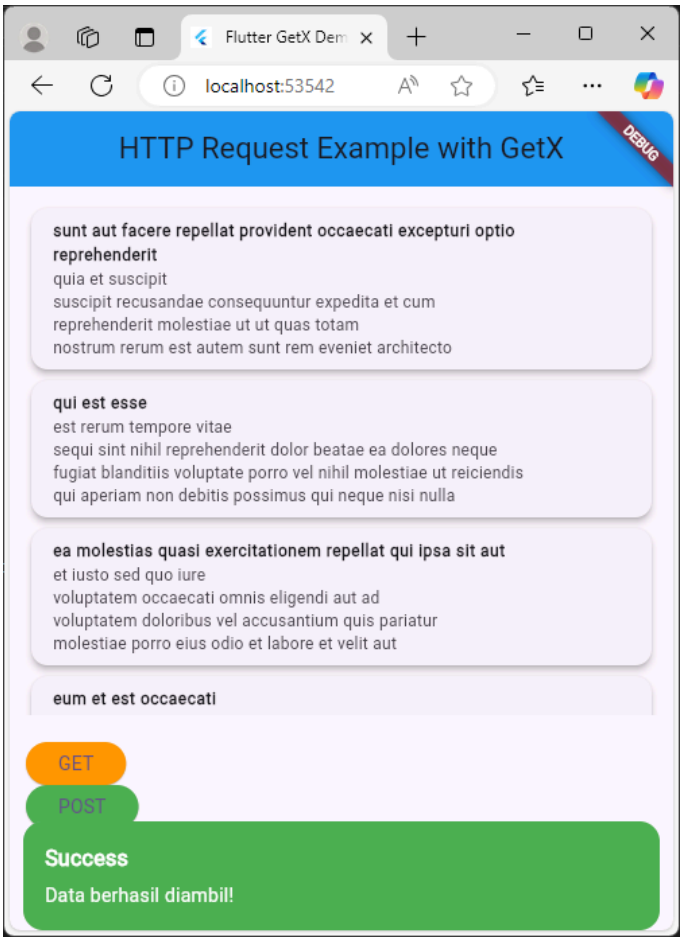
```dart
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:http/http.dart' as http;

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: 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;
```

```dart
    }
  }

  // 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;
    }
  }
}
```

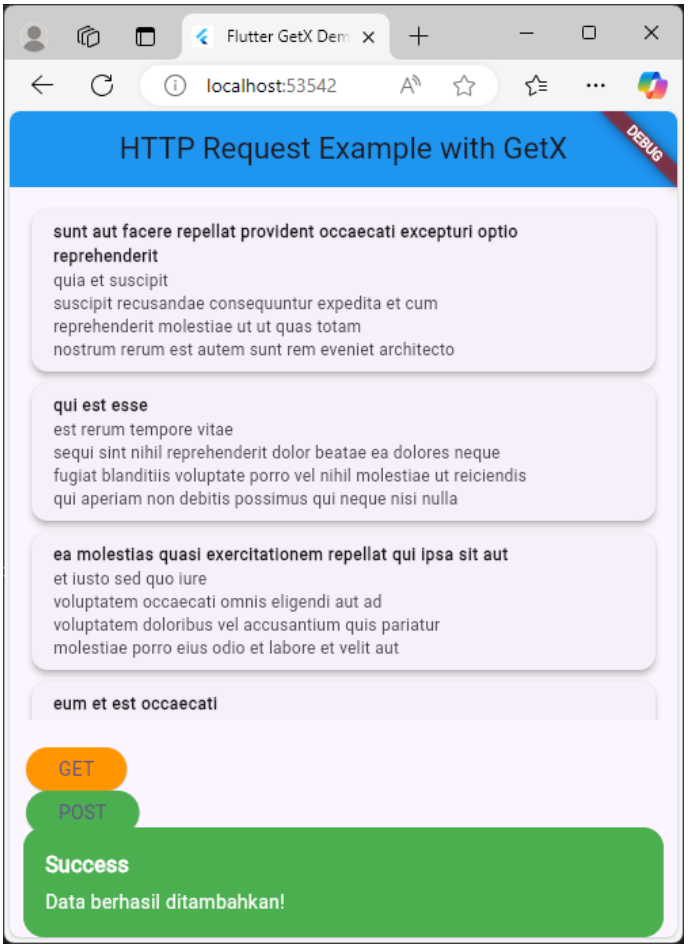
- **Output Hasil:**
  1. **Halaman awal**




  2. **Get**



## 3. Post




## 4. Update


5. **Delete**


- **Deskripsi Program:**

  Program ini adalah aplikasi Flutter sederhana yang memanfaatkan *state management* **GetX** untuk melakukan operasi CRUD (Create, Read, Update, Delete) melalui API. Aplikasi ini menggunakan *API* publik "https://jsonplaceholder.typicode.com" sebagai sumber data. Dengan bantuan **GetX**, data disimpan secara reaktif menggunakan variabel obs sehingga setiap perubahan langsung memperbarui UI. Terdapat empat fungsi utama dalam ApiController: **fetchPosts** untuk mengambil data, **createPost** untuk menambahkan data, **updatePost** untuk memperbarui data, dan **deletePost** untuk menghapus data. Setiap operasi menampilkan notifikasi keberhasilan atau kegagalan menggunakan Get.snackbar. Pengguna dapat berinteraksi dengan aplikasi melalui tombol GET, POST, UPDATE, dan DELETE yang disediakan di layar utama. Proses *loading* ditampilkan menggunakan *CircularProgressIndicator* selama operasi berlangsung, memberikan pengalaman pengguna yang lebih baik.