

**TUGAS PENDAHULUAN  
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XIV  
DATA STORAGE 'API'**



**Disusun Oleh :  
SALMAN ALFA RIZZI  
2211104056**

**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**

# DATA STORAGE API

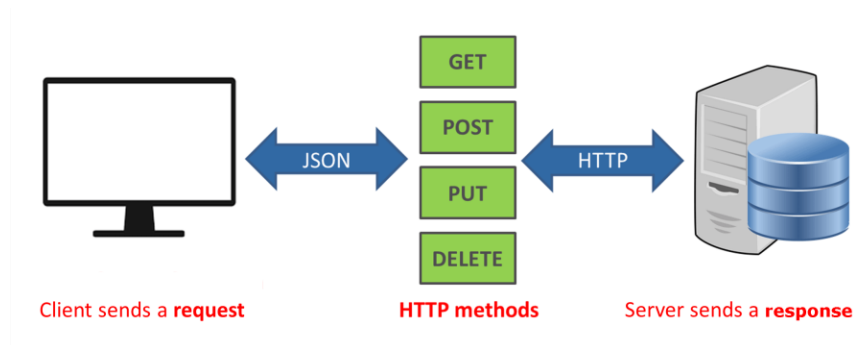
## Tujuan Praktikum

Memahami konsep dasar database web service menggunakan REST API.

Mengetahui kegunaan dan penerapan metode HTTP (GET, POST, PUT, DELETE) dalam pengelolaan data.

Membuat antarmuka pengguna (UI) yang menampilkan data dari API serta memungkinkan interaksi dengan server.

## Data Storage dengan Rest API



Gambar 1. Cara Kerja Restful API

### 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

Langkah-langkah implementasi REST API di Flutter

### 1. Persiapan Proyek Flutter

- a. Buat proyek Flutter baru
- b. Tambahkan dependensi http ke file pubspec.yaml:

```
dependencies:  
  flutter:  
    sdk: flutter  
  http: ^0.15.0
```

Jalankan perintah flutter pub get untuk menginstal dependensi:

### 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:

```

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 `screens`:

Fungsi untuk memanggil file api service

```

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

```

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),
                      ),

```

```

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

```

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'),
),

```

## 4. Implementasi HTTP POST

### a. Membuat Service POST

Tambahkan metode berikut ke `api_service.dart`:

```

// 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

Tambahkan tombol untuk POST di `home_screen.dart`:

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

## 5. Implementasi HTTP PUT

a. Membuat Service PUT

Tambahkan metode berikut ke `api_service.dart`:

```
// 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

Tambahkan logika untuk memperbarui postingan di `home_screen.dart`:

```
ElevatedButton(  
  onPressed: () => _handleApiOperation(  
    _apiService.updatePost(), 'Data berhasil diperbarui!'),
```

```
style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),  
child: const Text('UPDATE'),  
),
```

## 6. Implementasi HTTP DELETE

### a. Membuat Service DELETE

Tambahkan metode berikut ke `api_service.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');  
  }  
}
```

### b. Membuat tampilan UI untuk DELETE

Tambahkan tombol untuk DELETE di `home_screen.dart`:

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



## Tugas Mandiri (Unguided)

Modifikasi tampilan Guided dari praktikum di atas:

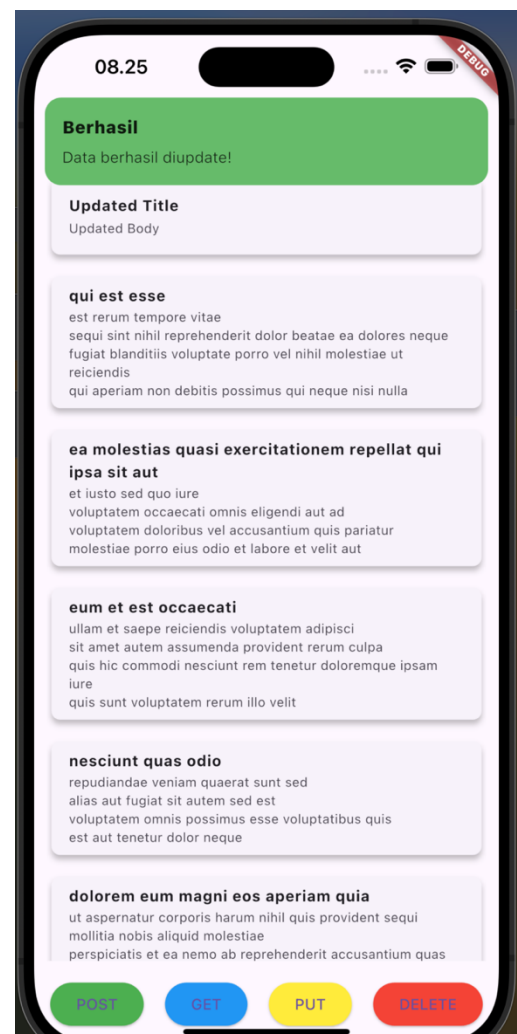
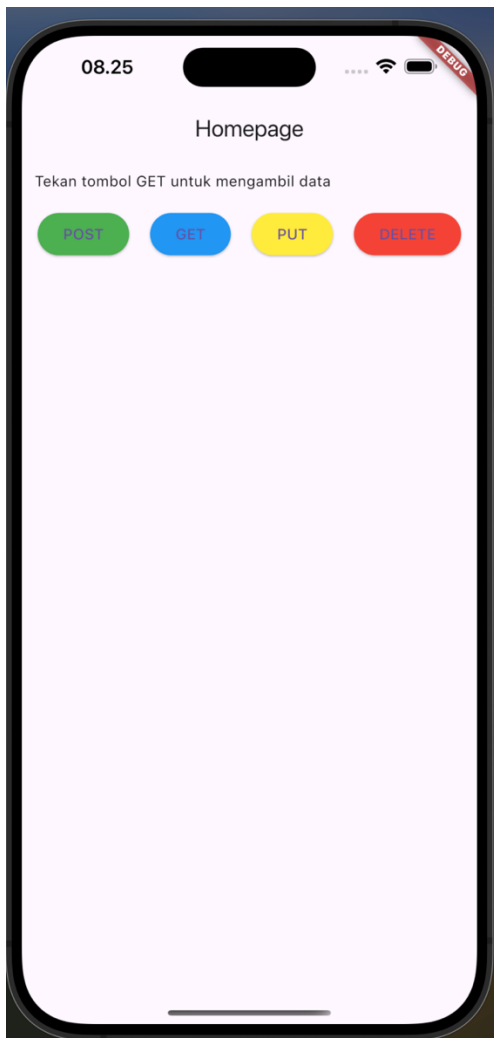
### a. Gunakan State Management dengan GetX:

- 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, screenshot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.*





SourceCode:

```
Post_controller.dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:praktikum_14/service/api_service.dart';

class PostController extends GetxController {
  final ApiService _apiService = ApiService();
  var posts = <dynamic>[].obs;
  var isLoading = false.obs;

  Future<void> fetchPosts() async {
    isLoading.value = true;
    try {
      await _apiService.fetchPosts();
      posts.assignAll(_apiService.posts);
      Get.snackbar('Berhasil', 'Data berhasil diambil!');
```

```

        backgroundColor: Colors.green[400]);
    } catch (e) {
        Get.snackbar('Error', e.toString());
    } finally {
        isLoading.value = false;
    }
}

Future<void> createPost() async {
    isLoading.value = true;
    try {
        await _apiService.createPost();
        posts.assignAll(_apiService.posts);
        Get.snackbar('Berhasil', 'Data berhasil ditambahkan!',
            backgroundColor: Colors.green[400]);
    } catch (e) {
        Get.snackbar('Error', e.toString());
    } finally {
        isLoading.value = false;
    }
}

Future<void> updatePost() async {
    isLoading.value = true;
    try {
        await _apiService.updatePost();
        posts.assignAll(_apiService.posts);
        Get.snackbar('Berhasil', 'Data berhasil diupdate!',
            backgroundColor: Colors.green[400]);
    } catch (e) {
        Get.snackbar('Error', e.toString());
    } finally {
        isLoading.value = false;
    }
}

Future<void> deletePost() async {
    isLoading.value = true;
    try {
        await _apiService.deletePost();
        posts.assignAll(_apiService.posts);
        Get.snackbar('Berhasil', 'Data berhasil dihapus!',
            backgroundColor: Colors.green[400]);
    } catch (e) {
        Get.snackbar('Error', e.toString());
    } finally {
        isLoading.value = false;
    }
}
}

```

## Homepage\_screen.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:praktikum_14/controller/post_controller.dart';

class HomepageScreen extends StatelessWidget {
  final PostController postController = Get.put(PostController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Homepage'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Bagian untuk menampilkan loading atau data
            Obx(() {
              if (postController.isLoading.value) {
                return const Center(child: CircularProgressIndicator());
              } else if (postController.posts.isEmpty) {
                return const Text(
                  "Tekan tombol GET untuk mengambil data",
                  style: TextStyle(fontSize: 14),
                );
              } else {
                return Expanded(
                  child: ListView.builder(
                    itemCount: postController.posts.length,
                    itemBuilder: (context, index) {
                      return Padding(
                        padding: const EdgeInsets.only(bottom: 12.0),
                        child: Card(
                          elevation: 4,
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(8),
                          ),
                          child: ListTile(
                            title: Text(
                              postController.posts[index]['title'] ?? 'No Title',
                              style: const TextStyle(
                                fontWeight: FontWeight.bold,
                                fontSize: 14,
                              ),
                            ),
                            subtitle: Text(
                              postController.posts[index]['body'] ?? 'No Content',
                              style: const TextStyle(fontSize: 12),
                            ),
                          ),
                        ),
                      );
                    },
                  ),
                );
              }
            })
          ],
        ),
      ),
    );
  }
}
```

```

        ),
      ),
    );
  },
),
);
}
}),
const SizedBox(height: 16),
// Tombol API
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    ElevatedButton(
      onPressed: postController.createPost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
      child: const Text('POST'),
    ),
    const SizedBox(width: 8),
    ElevatedButton(
      onPressed: postController.fetchPosts,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.blue),
      child: const Text('GET'),
    ),
    const SizedBox(width: 8),
    ElevatedButton(
      onPressed: postController.updatePost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.yellow),
      child: const Text('PUT'),
    ),
    const SizedBox(width: 8),
    ElevatedButton(
      onPressed: postController.deletePost,
      style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
      child: const Text('DELETE'),
    ),
  ],
),
),
),
),
);
}
}

```

Api\_service.dart

```

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

class ApiService {
  final String baseUrl = "https://jsonplaceholder.typicode.com";

```

```
List<dynamic> posts = [];

Future<void> fetchPosts() async {
  try {
    final response = await http.get(Uri.parse('$baseUrl/posts'));
    if (response.statusCode == 200) {
      posts = json.decode(response.body);
    } else {
      throw Exception('Failed to load posts');
    }
  } catch (e) {
    throw Exception('Error: $e');
  }
}

Future<void> createPost() async {
  try {
    final response = await http.post(
      Uri.parse('$baseUrl/posts'),
      headers: {'Content-Type': 'application/json'},
      body: json.encode({
        'title': 'Flutter Post',
        'body': 'Ini contoh POST.',
        'userId': 1,
      }),
    );
    if (response.statusCode == 201) {
      posts.add(json.decode(response.body));
    } else {
      throw Exception('Failed to create post');
    }
  } catch (e) {
    throw Exception('Error: $e');
  }
}

Future<void> updatePost() async {
  try {
    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');
    }
  }
}
```

```

    }
  } catch (e) {
    throw Exception('Error: $e');
  }
}

Future<void> deletePost() async {
  try {
    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');
    }
  } catch (e) {
    throw Exception('Error: $e');
  }
}
}

```

#### Main.dart

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:praktikum_14/screen/homepage_screen.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      initialRoute: '/home',
      getPages: [
        GetPage(name: '/home',
          page: ()=> HomepageScreen()),
      ],
    );
  }
}

```