Pertemuan 9

Penyimpanan Data Client-Server pada Aplikasi Flutter

Yunia Ikawati

Workshop Pemrograman Perangkat Bergerak

Tujuan Pembelajaran

- Memahami arsitektur client-server

- Mengintegrasikan Flutter dengan REST API

- Menyimpan dan membaca data dari server

ARSITEKTUR CLIENT SERVER

Arsitektur **client-server pada Flutter** menggambarkan bagaimana aplikasi Flutter (client) berkomunikasi dengan server untuk mengakses, menyimpan, dan mengelola data.

✅ Client (Flutter App)

- Berada di sisi pengguna (mobile/web app).
- Menyediakan antarmuka pengguna (UI).
- Mengirim permintaan ke server (**HTTP Request**).
- Menerima data dari server (**HTTP Response**).

✅ Server (Backend/API)

- Menangani permintaan dari client.
- Server biasanya dibangun menggunakan framework backend seperti **Node.js, Django,** atau **Express.js.**
- Mengakses database.
- Mengirimkan data kembali ke client dalam format **JSON/XML** (biasanya JSON).

✅ Database

- Tempat penyimpanan permanen data.
- Bisa berupa **MySQL**, **PostgreSQL**, **MongoDB**, dll.

MENGAPA FLUTTER UNTUK CLIENT?

1. Multiplatform (Cross-Platform)

- Satu basis kode bisa digunakan untuk Android, iOS, web, dan desktop.
- Tidak perlu menulis kode berbeda untuk setiap platform.

2. UI yang Indah & Responsif

- Flutter punya sistem UI sendiri (bukan native wrapper).
- Banyak widget siap pakai (Material, Cupertino).
- Dukungan animasi yang halus dan mudah dibuat.

3. Integrasi API Mudah

- Paket seperti http dan dio sangat memudahkan komunikasi dengan REST API.
- Parsing JSON bisa dilakukan langsung dengan json.decode.

TOOLS YANG DIGUNAKAN

Komponen	Tools/Framework
Client	Flutter + HTTP/Dio package
API/Server	PHP, Node.js, Python Flask/Django, dll
Database	MySQL, PostgreSQL, MongoDB
Testing API	Postman / cURL

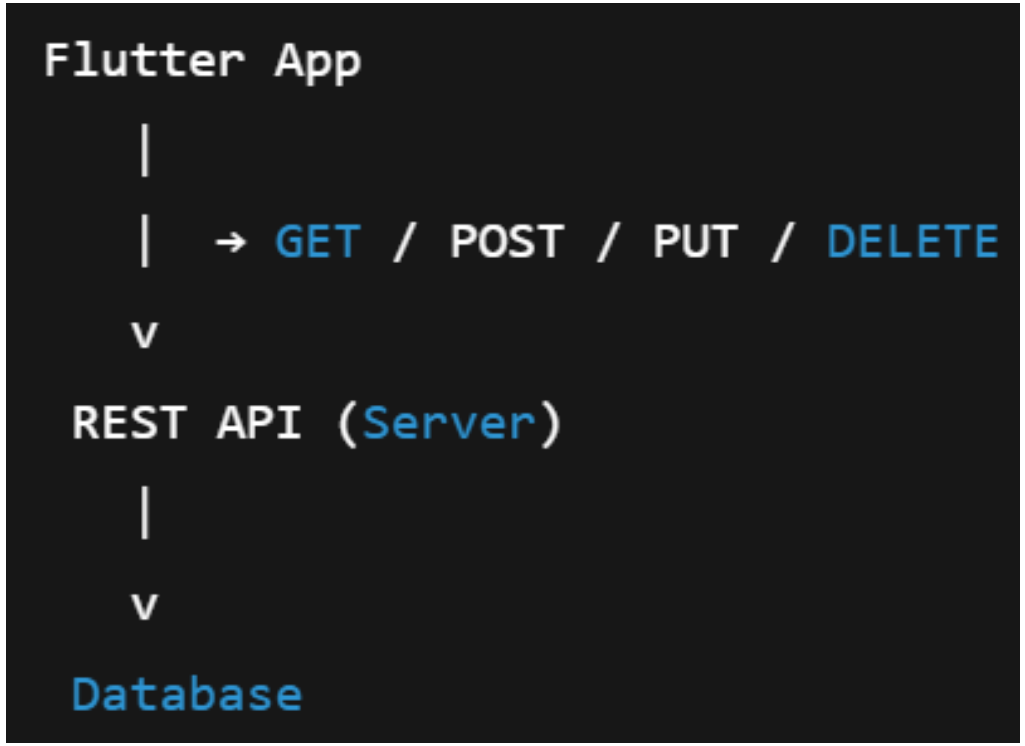
API PADA FLUTTER

API (**Application Programming Interface**) adalah jembatan komunikasi antara aplikasi (client) dan server (backend). Di Flutter, API biasa digunakan dalam bentuk **REST API** (menggunakan HTTP request).

Manfaat Flutter

1. Mengambil Data dari Server (GET)
2. Mengirim Data ke Server (POST)
3. Mengupdate Data (PUT / PATCH)
4. Menghapus Data (DELETE)
5. Autentikasi dan Login
6. Sinkronisasi Data

DIAGRAM API



Mengintegrasikan Flutter dengan REST API

- Mengintegrasikan Flutter dengan REST API adalah langkah penting agar aplikasi bisa berinteraksi dengan server untuk menyimpan, menampilkan, atau mengelola data. Proses ini umum digunakan dalam berbagai aplikasi nyata seperti login, to-do list, e-commerce, dll.

Langkah-Langkah Mengintegrasikan Flutter dengan REST API

1. Tambahkan Package HTTP

Tambahkan package http di file pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.14.0
```

2. Lalu jalankan:

```
flutter pub get
```

Langkah-Langkah Mengintegrasikan Flutter dengan REST API

2. Buat Fungsi Koneksi API

Misalnya, kita ingin mengambil data dari API:

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

Future<List<dynamic>> fetchData() async {
  final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

  if (response.statusCode == 200) {
    return jsonDecode(response.body); // Mengubah JSON ke List
  } else {
    throw Exception('Gagal mengambil data');
  }
}
```

Langkah-Langkah Mengintegrasikan Flutter dengan REST API

3. Tampilkan Data di UI dengan FutureBuilder

```dart
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Data dari API')),
      body: FutureBuilder(
        future: fetchData(),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            final data = snapshot.data as List;
            return ListView.builder(
              itemCount: data.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(data[index]['title']),
                );
              },
            );
          } else if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          return Center(child: CircularProgressIndicator());
        },
      ),
    );
  }
}
```

Langkah-Langkah Mengintegrasikan Flutter dengan REST API

4. Mengirim Data (POST) ke API

```dart
Future<void> postData(String title) async {
  final response = await http.post(
    Uri.parse('https://jsonplaceholder.typicode.com/posts'),
    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({'title': title}),
  );

  if (response.statusCode == 201) {
    print('Data berhasil dikirim');
  } else {
    throw Exception('Gagal mengirim data');
  }
}
```

Langkah-Langkah Mengintegrasikan Flutter dengan REST API

5. Praktik CRUD Lengkap

- GET → Ambil data
- POST → Tambah data
- PUT / PATCH → Edit data
- DELETE → Hapus data

PERCOBAAN
Contoh Penerapan Aplikasi Login menggunakan Flutter + API

- Struktur Client (Flutter)

```
lib/
├── main.dart
├── login_page.dart
├── services/
│   └── auth_service.dart
```

Struktur Server (PHP API)

```
api/
├── login.php
└── users.json (sebagai database sederhana)
```

Backend – API Login (PHP)

- Buatlah file **users.json** diletakkan di folder flutter_api didalam folder htdoc

```
htdocs/
└── flutter_api/
    ├── login.php
    └── users.json
```

users.json

```
1  [
2    { "username": "admin", "password": "123456" },
3    { "username": "user", "password": "password" }
4  ]
```

Backend – API Login (PHP)

- Buatlah file **login.php** diletakkan di folder flutter_api didalam folder htdoc Bersama file users.json

login.php

```php
<?php
header("Content-Type: application/json");

$data = json_decode(file_get_contents("php://input"), true);
$username = $data['username'];
$password = $data['password'];

$users = json_decode(file_get_contents("users.json"), true);

foreach ($users as $user) {
    if ($user['username'] == $username && $user['password'] == $password) {
        echo json_encode(["status" => "success", "message" => "Login successful"]);
        exit;
    }
}

echo json_encode(["status" => "error", "message" => "Invalid credentials"]);
```

Client - Flutter

- Pada bagian file pubspec.yaml edit atau tambahkan seperti berikut: File ini ada di root folder project Flutter kamu.


```
dependencies:
  flutter:
    sdk: flutter
  http:^1.3.0
```

Client - Flutter

- Buatlah file **auth_service.dart** seperti berikut:

```
lib/
├── main.dart
├── login_page.dart
├── services/
│   └── auth_service.dart
```

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

class AuthService {
  static Future<Map<String, dynamic>> login(
    String username,
    String password,
  ) async {
    final response = await http.post(
      Uri.parse("http://localhost/flutter_api/login.php"), // sesuaikan path
      headers: {"Content-Type": "application/json"},
      body: jsonEncode({"username": username, "password": password}),
    );

    return jsonDecode(response.body);
  }
}
```

Client - Flutter

- Buatlah file **login_page.dart** seperti berikut:

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

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final userController = TextEditingController();
  final passController = TextEditingController();
  String result = "";

  void handleLogin() async {
    var response = await AuthService.login(
      userController.text,
      passController.text,
    );
    setState(() {
      result = response['message'];
    });
  }
```

```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Login API")),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: userController,
              decoration: InputDecoration(labelText: "Username"),
            ),
            TextField(
              controller: passController,
              decoration: InputDecoration(labelText: "Password"),
              obscureText: true,
            ),
            SizedBox(height: 20),
            ElevatedButton(onPressed: handleLogin, child: Text("Login")),
            SizedBox(height: 20),
            Text(result),
          ],
        ),
      ),
    );
  }
}
```

Client - Flutter

- Buatlah file **main.dart** seperti berikut:

```
lib/
├── main.dart
├── login_page.dart
├── services/
│   └── auth_service.dart
```

```dart
import 'package:flutter/material.dart';
import 'login_page.dart';

Run | Debug | Profile
void main() {
  runApp(MaterialApp(home: LoginPage()));
}
```

Cara Menjalankan Aplikasi Login

1. Jalankan server lokal **(XAMPP/Laragon)** dan tempatkan file login.php dan users.json di dalam folder htdocs/flutter_api.
2. Jalankan aplikasi Flutter dengan emulator atau device dengan menjalankan file **main.dart**.
3. Login menggunakan username **admin** dan password **123456** sesuai yang ada pada file users.json.

Output Pada Emulator

LATIHAN

- Silahkan praktekkan Langkah-Langkah membuat aplikasi login dan logout dengan menggunakan Flutter dan Rest API

https://sobatcoding.com/articles/flutter-membuat-form-login-dan-logout-menggunakan-rest-api

LAPORAN

- Kerjakan Percobaan dan Latihan lalu buat laporan.

Referensi

- https://docs.flutter.dev/data-and-backend/networking
- https://www.youtube.com/watch?v=rzTpk849ktA&list=PL0-7Xi0GB3teRqkuBusUEcVrP6OlYpD9w
- https://sobatcoding.com/articles/flutter-membuat-form-login-dan-logout-menggunakan-rest-api