

VERSION 1.0

JULI, 2023



PEMROGRAMAN MOBILE

PUBLIC API DAN WEBVIEW – MODUL 3 MATERI

TIM PENYUSUN:

- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM.
- MUHAMMAD ZULFIQOR LILHAQ
- RIYAN PUTRA FIRJATULLAH

PRESENTED BY: LAB. INFORMATIKA UNIVERSITAS MUHAMMADIYAH MALANG

CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat memahami penggunaan API menggunakan Framework Flutter.
 2. Mahasiswa dapat memahami penggunaan Webview menggunakan Framework Flutter.
-

SUB CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat mengimplementasikan API pada aplikasi menggunakan Framework Flutter.
 2. Mahasiswa dapat mengimplementasikan Webview pada aplikasi menggunakan Framework Flutter.
-

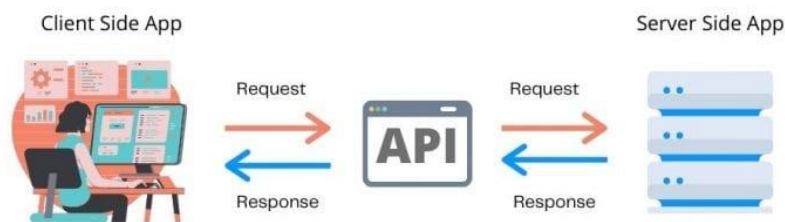
KEBUTUHAN HARDWARE & SOFTWARE

1. PC/Laptop
 2. IDE Android Studio/ Visual Studio Code
 3. Flutter SDK: <https://docs.flutter.dev/release/archive?tab=windows>
-

MATERI POKOK

Pengertian API

API atau Application Programming Interface adalah sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh programmer saat membangun perangkat lunak tertentu. Dengan kata lain, API adalah sebuah penghubung antara suatu aplikasi untuk dapat berinteraksi dengan aplikasi lainnya. Jika kita ibaratkan pada pengambilan data dari internet, API adalah sebuah jembatan antara server dengan klien (aplikasi yang akan kita bangun). Format response dari API biasanya berbentuk JSON.



Client Side App akan melakukan request terhadap server melalui dengan dijembatani oleh API. Kemudian dari sisi server akan memberikan respon ke client side app yang nanti akan ditampilkan pada platform seperti web atau android dan dapat diakses oleh user.

Pada Flutter kita dapat mengambil data di internet menggunakan aplikasi pihak ketiga yang dibuat oleh tim Flutter itu sendiri. Flutter sudah menyediakan cara sederhana untuk mengelola data yang bersumber dari internet menggunakan `http` package.

JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) adalah format data ringan yang digunakan untuk pertukaran data antar aplikasi atau server dan klien. Ini sangat umum digunakan dalam pengembangan web dan mobile untuk mengirim dan menerima data. JSON mudah dibaca oleh manusia dan mudah diurai (parsed) oleh mesin. Struktur dari JSON terdiri dari **key** dan **value**.

1. **Key** harus dalam bentuk string dan juga berisi urutan karakter yang diapit oleh tanda kutip.
2. **Value** adalah tipe data JSON yang valid dan dapat berbentuk array, object, string, boolean, angka, atau null.

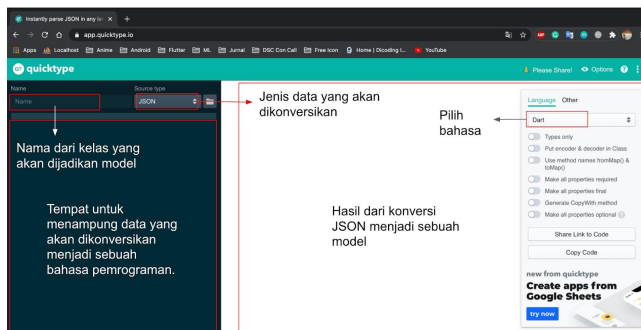
Objek (object) JSON diawali dan diakhiri dengan kurung kurawal `{}`. Di dalam kurung kurawal tersebut dapat berisi dua atau lebih **key/value** dengan tanda koma yang memisahkan keduanya. Sedangkan tiap **key** diikuti oleh simbol titik dua untuk membedakannya dengan **value**. Contoh format JSON:

```
{
  "nama": "John Doe",
  "umur": 30,
  "alamat": {
    "jalan": "Jl. Pahlawan",
    "kota": "Jakarta"
  },
  "hobi": ["berenang", "membaca"]
}
```

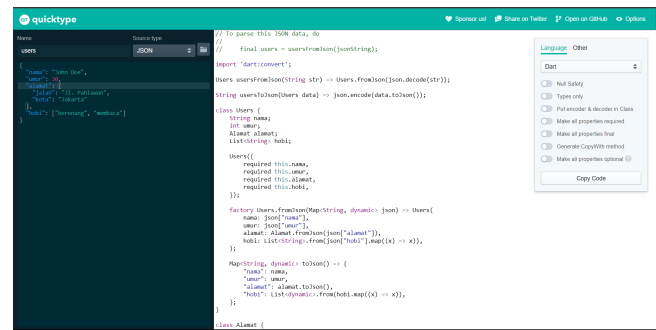
JSON Parsing adalah proses mengambil data dalam format JSON dan mengubahnya menjadi struktur data yang dapat digunakan dalam aplikasi. Dalam konteks pengembangan perangkat lunak, JSON Parsing adalah langkah penting untuk mengambil informasi yang dikirim oleh server dalam bentuk JSON dan mengkonversinya menjadi objek atau tipe data yang dapat dimanfaatkan oleh program kita. Dalam melakukan konversi data dari JSON menjadi sebuah kelas model terdapat cara cepat untuk melakukannya yaitu dengan menggunakan aplikasi **Quicktype**.

Quicktype

[Quicktype](#) merupakan sebuah website untuk melakukan proses generate kelas model dan juga serialisasi yang bersumber dari JSON, schema, dan GraphQL. Dengan memanfaatkan website ini, kita dapat melakukan pekerjaan dengan data sangat cepat tanpa harus melakukan coding manual. Untuk melakukan konversi website ini juga sudah mendukung beberapa bahasa pemrograman seperti Dart, Kotlin, Java, dsb. Kita cukup melakukan copy dan paste data dari JSON dan nanti akan secara otomatis di-generate kelas modelnya sendiri.



Tampilan Quicktype



Public API

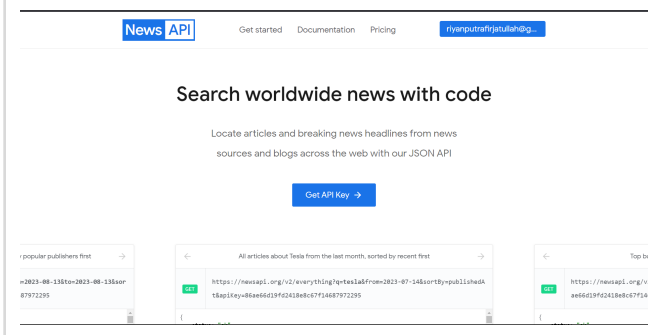
Public API (Application Programming Interface) adalah antarmuka yang disediakan oleh suatu perangkat lunak atau layanan untuk memungkinkan pengembang lain berinteraksi dengan perangkat lunak atau layanan tersebut. Public API memungkinkan pengembang eksternal untuk mengakses dan menggunakan fungsionalitas yang disediakan oleh perangkat lunak atau layanan tersebut, seperti mengambil atau mengirim data, menjalankan tindakan tertentu, atau berkomunikasi dengan sumber daya eksternal.

Public API umumnya disediakan oleh perusahaan, platform, atau layanan untuk memungkinkan pengembang pihak ketiga membangun aplikasi, layanan, atau integrasi dengan perangkat lunak atau layanan tersebut. Public API memainkan peran penting dalam memungkinkan pengembangan aplikasi lintas platform, integrasi sistem, dan ekosistem yang lebih luas. Terdapat beberapa tempat untuk mengakses API yang tersedia secara gratis dan terbuka yaitu antara lain [AnyAPI](#), [Toddmotto](#), dan [Public APIs](#).

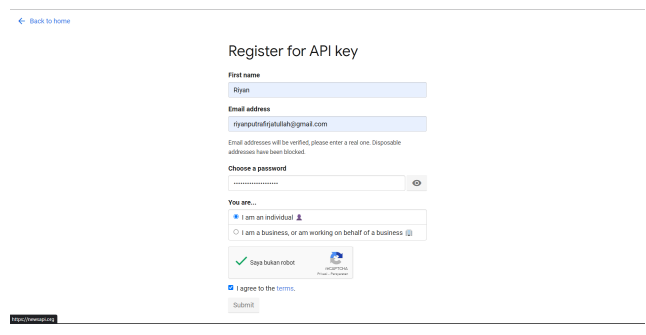
Penggunaan

Pada percobaan kali ini kita akan menggunakan Public API dari [News API](#).

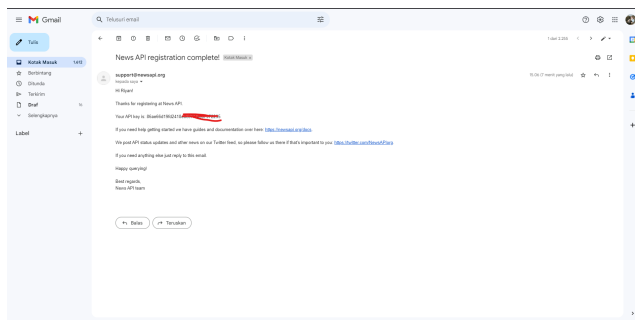
1. Pada halaman Home pilih **Get API Key**



2. Pada halaman register isi data diri dan email yang ingin digunakan.



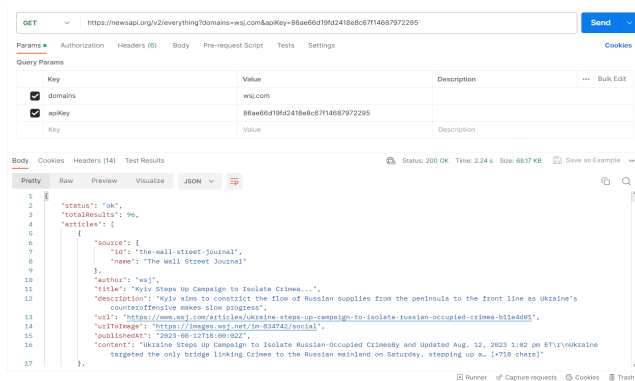
3. Setelah submit akan mendapatkan **API Key** melalui **email**.



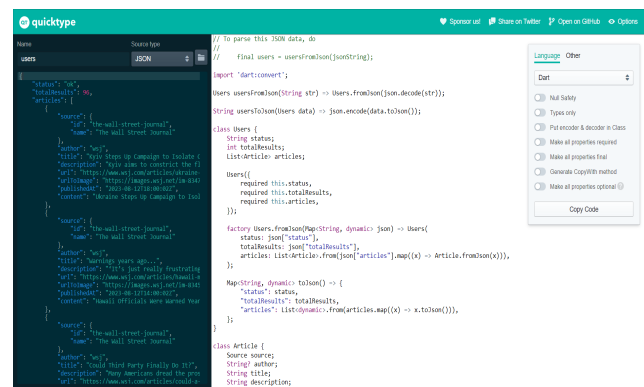
4. Setelah mendapatkan **API Key**, tentukan **URL** tautan yang akan digunakan. Kita tinggal mengganti setelah **apiKey=** pada URL dengan **API Key** yang kita dapat.



5. Gunakan Aplikasi **Postman** untuk mengetes response API caranya yaitu dengan **salin URL** tadi ke dalam Postman pilih **metode GET** dan klik **Send**.



6. Setelah berhasil mendapat response API berupa **JSON**, selanjutnya kita akan membuat model dari response tersebut. Kita bisa menggunakan aplikasi **Quicktype** untuk melakukan konversi data menjadi sebuah model. Salin data response API di Postman lalu Paste ke aplikasi Quicktype. Setelah selesai, salin hasil konversi data ke model.



7. Setelah itu masuk ke project kita lalu buat file **article.dart** contohnya bisa dibuat melalui folder **data/models/article.dart**, lalu paste model yang sudah dibuat tadi.

```
class ArticlesResult {
  final String status;
  final int totalResults;
  final List<Article> articles;
```

```
ArticlesResult({
  required this.status,
  required this.totalResults,
  required this.articles,
```

```

});

factory ArticlesResult.fromJson(Map<String, dynamic> json) => ArticlesResult(
  status: json["status"],
  totalResults: json["totalResults"],
  articles: List<Article>.from((json["articles"] as List)
    .map((x) => Article.fromJson(x))
    .where((article) =>
      article.author != null &&
      article.description != null &&
      article.urlToImage != null &&
      article.publishedAt != null &&
      article.content != null)),
  );
}

class Article {
  String? author;
  String title;
  String? description;
  String url;
  String? urlToImage;
  DateTime? publishedAt;
  String? content;

  Article({
    required this.author,
    required this.title,
    required this.description,
    required this.url,
    required this.urlToImage,
    required this.publishedAt,
    required this.content,
  });

  factory Article.fromJson(Map<String, dynamic> json) => Article(
    author: json["author"],
    title: json["title"],
    description: json["description"],
    url: json["url"],
    urlToImage: json["urlToImage"],
    publishedAt: DateTime.parse(json["publishedAt"]),
    content: json["content"],
  );
}

```

```
);  
}
```

8. Setelah menyiapkan model `response`. Selanjutnya `API` dapat dipanggil menggunakan salah satu dari beberapa `package` berikut [http-package](#), [dio-package](#), [getconnect-package](#).



HTTP Package

HTTP package adalah salah satu paket (library) yang disediakan dalam bahasa Dart dan digunakan secara luas dalam pengembangan aplikasi Flutter. Paket ini memberikan dukungan untuk berkomunikasi dengan server menggunakan protokol HTTP untuk mengambil atau mengirim data melalui jaringan.

Dengan HTTP package, Anda dapat melakukan berbagai operasi seperti melakukan permintaan (request) ke server untuk mendapatkan data **GET request**, mengirim data ke server **POST request**, mengirim data terbaru ke server **PUT request**, menghapus data dari server **DELETE request**, dan lainnya.

Installation

Instalasi dilakukan cukup dengan menambahkan http plugin pada file **pubspec.yaml** [http](#).

- 1 Buka file **pubspec.yaml** pada project
 -  nama_project
 -  **pubspec.yaml**
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

```
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: ^1.0.2
```

```
dependencies:  
flutter:  
  sdk: flutter  
  
# http  
http: ^1.1.0  
  
cupertino_icons: ^1.0.2
```

- 3 Selanjutnya **http** sudah dapat dipakai oleh project

Penggunaan

Pada percobaan kali ini kita akan menggunakan Public API dari [News API](#).

1. Setelah model response tersedia, kita akan membuat class **ApiService**. Di dalam kelas tersebut kita akan membuat fungsi Future yang bersifat async untuk menghubungkan antara aplikasi Flutter (klien) dengan data yang bersumber dari API (server). Buatlah file **http.controller.dart**. Pada modul 2 kita telah mempelajari **State Management GetX** maka penggunaan **State Management GetX** dapat digunakan pada bagian ini.

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

import 'package:module_app/internal/models/models.dart';

class HttpController extends GetxController {
  static const String _baseUrl = 'https://newsapi.org/v2/';
  static const String _apiKey = 'YOUR_API_KEY'; //Ganti ke API KEY yang sudah didapat
  static const String _category = 'business';
  static const String _country = 'us'; //us maksudnya United States ya

  RxList<Article> articles = RxList<Article>([]);
  RxBool isLoading = false.obs; // Observable boolean for loading state

  @override
  onInit() async {
    await fetchArticles();
  }

  Future<void> fetchArticles() async {
    try {
      isLoading.value = true; // Set loading state to true
      final response =
        await
      http.get(Uri.parse('${_baseUrl}top-headlines?country=$_country&category=$_category&apiKey=$_apiKey'));
      if (response.statusCode == 200) {
        final jsonData = response.body;
        final articlesResult = ArticlesResult.fromJson(json.decode(jsonData));
        articles.value = articlesResult.articles;
      } else {
        print('Request failed with status: ${response.statusCode}');
      }
    }
  }
}
```

```
    }  
  } catch (e) {  
    print('An error occurred: $e');  
  } finally {  
    isLoading.value = false; // Set loading state to false when done  
  }  
}  
}
```



2. API yang sudah berhasil dipanggil kemudian akan ditampilkan pada [screen widget](#)

Dio Package

Adalah library jaringan HTTP yang powerful untuk Dart/Flutter, yang mendukung konfigurasi Global, Pencegat, FormData, Pembatalan permintaan, Pengunggahan/pengunduhan file, Timeout, Adaptor khusus, Transformers, dll.

Installation

Instalasi dilakukan cukup dengan menambahkan http plugin pada file `pubspec.yaml` [dio](#).

- 1 Buka file `pubspec.yaml` pada project
 -  nama_project
 -  `pubspec.yaml`
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

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: ^1.0.2

dependencies:

flutter:

 sdk: flutter

dio

dio: ^5.3.2

cupertino_icons: ^1.0.2

- 3 Selanjutnya **dio** sudah dapat dipakai oleh project

Penggunaan

Pada percobaan kali ini kita akan menggunakan Public API dari [News API](#).

1. Setelah model response tersedia, kita akan membuat class **ApiService**. Di dalam kelas tersebut kita akan membuat fungsi Future yang bersifat async untuk menghubungkan antara aplikasi Flutter (klien) dengan data yang bersumber dari API (server). Buatlah file **api_service.dart** pada folder **data/api/api_service.dart**. Pada modul 2 kita telah mempelajari **State Management GetX** maka penggunaan **State Management GetX** dapat digunakan pada bagian ini.

```
import 'dart:convert';
import 'package:dio/dio.dart';
import 'package:get/get.dart';

import 'package:module_app/internal/models/models.dart';

class DioController extends GetxController {
  static const String _baseUrl = 'https://newsapi.org/v2/';
  static const String _apiKey = 'YOUR_API_KEY'; //Ganti ke API KEY yang sudah didapat
  static const String _category = 'business';
  static const String _country = 'us'; //us maksudnya United States ya

  RxList<Article> articles = RxList<Article>([]);
  RxBool isLoading = false.obs; // Observable boolean for loading state

  final dio = Dio();

  @override
  onInit() async {
    await fetchArticles();
  }

  Future<void> fetchArticles() async {
    try {
      isLoading.value = true; // Set loading state to true
      final response = await
dio.get('${_baseUrl}top-headlines?country=$_country&category=$_category&apiKey=$_apiKey');
      if (response.statusCode == 200) {
        final jsonData = response.data;
        final articlesResult = ArticlesResult.fromJson(json.decode(jsonData));
        articles.value = articlesResult.articles;
      } else {
        print('Request failed with status: ${response.statusCode}');
      }
    }
  }
}
```

```
    }  
  } catch (e) {  
    print('An error occurred: $e');  
  } finally {  
    isLoading.value = false; // Set loading state to false when done  
  }  
}  
}
```



2. API yang sudah berhasil dipanggil kemudian akan ditampilkan pada [screen widget](#)

GetConnect Package

Adalah salah satu fitur dari package Getx yang memudahkan berkomunikasi dari backend ke frontend menggunakan http atau websocket [get-connect](#).

Installation

Instalasi dilakukan cukup dengan menambahkan http plugin pada file `pubspec.yaml` [http](#).

- 1 Buka file `pubspec.yaml` pada project
 -  nama_project
 -  `pubspec.yaml`
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

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: ^1.0.2

dependencies:

flutter:

 sdk: flutter

state management

get: 4.6.5

cupertino_icons: ^1.0.2

- 3 Selanjutnya **http** sudah dapat dipakai oleh project

Penggunaan

Pada percobaan kali ini kita akan menggunakan Public API dari [News API](#).

1. Setelah model response tersedia, kita akan membuat class `ApiService`. Di dalam kelas tersebut kita akan membuat fungsi Future yang bersifat async untuk menghubungkan antara aplikasi Flutter (klien) dengan data yang bersumber dari API (server). Buatlah file `api_service.dart` pada folder `data/api/api_service.dart`. Pada modul 2 kita telah mempelajari `State Management GetX` maka penggunaan `State Management GetX` dapat digunakan pada bagian ini.

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

import 'package:module_app/internal/models/models.dart';

class GetConnectController extends GetxController {
  static const String _baseUrl = 'https://newsapi.org/v2/';
  static const String _apiKey = 'YOUR_API_KEY'; //Ganti ke API KEY yang sudah didapat
  static const String _category = 'business';
  static const String _country = 'us'; //us maksudnya United States ya

  RxList<Article> articles = RxList<Article>([]);
  RxBool isLoading = false.obs; // Observable boolean for loading state

  @override
  onInit() async {
    await fetchArticles();
  }

  Future<void> fetchArticles() async {
    try {
      isLoading.value = true; // Set loading state to true
      final response =
        await
        get(Uri.parse('${_baseUrl}top-headlines?country=$_country&category=$_category&apiKey=$_apiKey'));
      if (response.statusCode == 200) {
        final jsonData = response.body;
        final articlesResult = ArticlesResult.fromJson(json.decode(jsonData));
        articles.value = articlesResult.articles;
      } else {
        print('Request failed with status: ${response.statusCode}');
      }
    }
  }
}
```

```
    }  
  } catch (e) {  
    print('An error occurred: $e');  
  } finally {  
    isLoading.value = false; // Set loading state to false when done  
  }  
}  
}
```

2. API yang sudah berhasil dipanggil kemudian akan ditampilkan pada [screen widget](#)

Menampilkan API

Setelah berhasil memanggil API, selanjutnya response akan ditampilkan pada screen widget

1. Kalian bisa mengkreasikan tampilan UI yang diinginkan. Contohnya disini kita akan membuat sebuah custom widget untuk menampilkan List Card Article yang ada dengan membuat file `card_article.dart`, berikut contohnya:

```
class CardArticle extends StatelessWidget {
  final Article article;

  const CardArticle({Key? key, required this.article}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Material(
      color: Theme.of(context).primaryColor,
      child: ListTile(
        contentPadding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
        leading: Hero(
          tag: article.urlToImage!,
          child: SizedBox(
            width: 150,
            height: 150,
            child: Image.network(
              article.urlToImage!,
            ),
          ),
        ),
        title: Text(
          article.title,
        ),
        subtitle: Text(article.author ?? ''),
        onTap: () => Navigator.pushNamed(
          context,
          ArticleDetailPage.routeName,
          arguments: article,
        ),
      ),
    );
  }
}
```

2. Selanjutnya kita akan membuat `ArticleListPage` untuk tempat list article yang akan ditampilkan dan tempat `CardArticle` digunakan. Jadi kita buat file `article_listpage.dart` pada project kita.

```
class ArticlePage extends StatelessWidget {
  final dioController = Get.find<DioController>();
  final httpController = Get.find<HttpController>();
  final getConnectController = Get.find<GetConnectController>();

  ArticlePage({super.key});

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Obx(() {
          if (getConnectController.isLoading.value) {
            // Display a progress indicator while loading
            return Center(
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(Theme.of(context).colorScheme.secondary),
              ),
            );
          } else {
            // Display the list of articles
            return Expanded(
              child: ListView.builder(
                shrinkWrap: true,
                itemCount: getConnectController.articles.length,
                itemBuilder: (context, index) {
                  var article = getConnectController.articles[index];
                  return CardArticle(article: article);
                },
              ),
            );
          }
        })),
      ],
    );
  }
}
```

3. Lalu kita bisa membuat halaman detail ketika salah satu list article di klik. Jadi kita buat file `article_detail_page.dart` pada project kita.

```
class ArticleDetailPage extends StatelessWidget {
  static const routeName = '/article_detail';

  final Article article;

  const ArticleDetailPage({Key? key, required this.article}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('News App'),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            Hero(
              tag: article.urlToImage!,
              child: Image.network(article.urlToImage!),
            ),
            Padding(
              padding: const EdgeInsets.all(10),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    article.description ?? "-",
                    style: Theme.of(context).textTheme.bodyMedium,
                  ),
                  const Divider(color: Colors.grey),
                  Text(
                    article.title,
                    style: Theme.of(context).textTheme.titleLarge,
                  ),
                  const Divider(color: Colors.grey),
                  Text(
                    'Date: ${article.publishedAt}',
                    style: Theme.of(context).textTheme.bodySmall,
                  ),
                  const SizedBox(height: 10),
                ]
              )
            )
          ]
        )
      )
    );
  }
}
```

```
Text(  
  'Author: ${article.author}',  
  style: Theme.of(context).textTheme.bodySmall,  
)  
const Divider(color: Colors.grey),  
Text(  
  article.content ?? "-",  
  style: Theme.of(context).textTheme.bodyLarge,  
)  
const SizedBox(height: 10),  
ElevatedButton(  
  child: const Text('Read more'),  
  onPressed: () {  
    //comingsoon  
  },  
)  
],  
)  
,  
],  
)  
,  
],  
)  
,  
];  
}  
}
```



Webview

WebView adalah sebuah widget dalam Flutter yang memungkinkan Anda untuk menampilkan konten web di dalam aplikasi Flutter Anda. Widget ini menggunakan WebView yang ada di platform masing-masing untuk menampilkan halaman web di dalam aplikasi Flutter.

Widget WebView sangat berguna ketika Anda ingin **mengintegrasikan konten web** seperti halaman web, halaman login, atau konten lain yang diberikan oleh server ke dalam aplikasi Flutter Anda. Dengan menggunakan WebView, Anda dapat menjaga pengalaman pengguna tetap konsisten dengan tampilan aplikasi, sambil memanfaatkan fungsionalitas web yang mungkin diperlukan.

Installation

Instalasi dilakukan cukup dengan menambahkan webview plugin pada file **pubspec.yaml** [webview](#).

- 1 Buka file **pubspec.yaml** pada project
 -  nama_project
 -  pubspec.yaml
- 2 Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

```
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: ^1.0.2
```

```
dependencies:
flutter:
  sdk: flutter

# webview
webview_flutter: ^4.2.2

cupertino_icons: ^1.0.2
```

- 3 Selanjutnya **webview** sudah dapat dipakai oleh project

Penggunaan

Berikut adalah contoh penggunaan sederhana pada webview:

```
class WebViewExample extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('WebView Example'),  
      ),  
      body: WebView(  
        initialUrl: 'https://www.example.com', // Ganti dengan URL yang sesuai  
        javascriptMode: JavascriptMode.unrestricted, // Aktifkan JavaScript  
      ),  
    );  
  }  
}
```

Link Github

Berikut merupakan link github yang akan digunakan pada modul pemrograman mobile:

[Link Github](#)

KEGIATAN PRAKTIKUM

A. Instalasi HTTP dan Webview

1. Lakukan instalasi HTTP dan Webview pada project kalian dengan menambahkan dependencies di file `pubspec.yaml`
2. Kemudian lakukan `save/pub get`
3. Jalankan project seperti biasa

B. Latihan HTTP

1. Lakukan latihan untuk mengambil data di internet menggunakan `http` plugin. API yang dipakai yaitu melalui <https://jsonplaceholder.typicode.com/todos/5>.
2. Lakukan konversi pada data JSON menjadi sebuah model menggunakan aplikasi `QuickType`.
3. Buatlah sebuah function atau bisa sebuah class tersendiri untuk melakukan `fetchdata` (`ApiService`).
4. Buatlah `User Interface` untuk menampilkan data dari API yang sudah diambil.
5. Berikut adalah codingan bantuan jika kalian merasa kebingungan, tinggal melengkapi bagian yang kurang:

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:latihan_http/model/todos.dart';

class TodosPage extends StatefulWidget {
  const TodosPage({Key? key}) : super(key: key);

  @override
  State<TodosPage> createState() => _TodosPageState();
}

class _TodosPageState extends State<TodosPage> {
  late Future<Todos> _futureTodos;

  @override
  void initState() {
    super.initState();
    _futureTodos = fetchTodos();
  }

  Future<Todos> fetchTodos() async {
```

```

final response = await http
    .get(Uri.parse('URL_API'));
if (response.statusCode == 200) {
    return Todos.fromJson(json.decode(response.body));
} else {
    throw Exception('Failed to load album');
}
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text('Fetch Data Example'),
        ),
        body: Center(
            child: FutureBuilder<Todos>(
                future: _futureTodos,
                builder: (context, snapshot) {
                    var state = snapshot.connectionState;
                    if (state != ConnectionState.done) {
                        return const Center(child: CircularProgressIndicator());
                    } else {
                        if (snapshot.hasData) {
                            return Text(snapshot.data!.title);
                        } else if (snapshot.hasError) {
                            return Center(child: Text("${snapshot.error}"));
                        } else {
                            return const Text("");
                        }
                    }
                },
            ),
        ),
    );
}

```

6. Perlu diperhatikan bahwa codingan bantuan di atas masih belum menggunakan **state management** dan **Api Service** untuk fetch data masih dalam satu file dengan page presentation.
7. Maka opsi tambahan **gunakan state management Getx** dan **lakukan refactoring** pada codingan di atas, jika kalian menggunakan codingan bantuan.

RUBRIK PENILAIAN MODUL 3 MATERI**Bobot Penilaian Modul 3 Materi (20%)**

Berhasil melakukan instalasi Http dan Webview	35
Berhasil melakukan latihan http	45
Menggunakan state management pada latihan http	20
Total	100