

VERSION 1.1

JUNE, 2024



PEMROGRAMAN MOBILE

Modul 2 – Network API & Web Interfaces

Tim Penyusun:

Ali Sofyan, S.kom, M.Kom

Ahmad Naufal

Marchanda Balqis

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PEMROGRAMAN MOBILE

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Flutter dan Dart
2. State Management dengan GetX
3. Project Architecture seperti MVC, MVP, dll.

TUJUAN

Mahasiswa diharapkan mampu untuk melakukan dan memahami:

1. Mampu memahami apa itu API dan konsep nya.
2. Mampu mengimplementasikan API pada aplikasi menggunakan Framework Flutter.
3. Mampu mengimplementasikan WebView pada aplikasi menggunakan Framework Flutter.

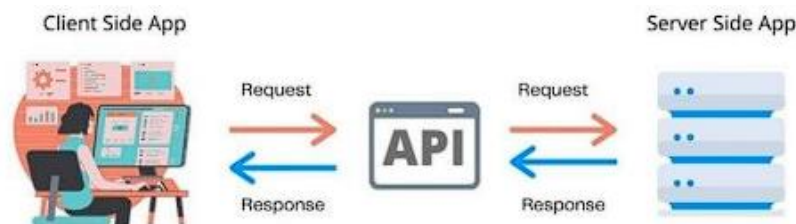
PERSIAPAN SOFTWARE/APLIKASI

1. Laptop/PC
2. IDE Android Studio / VS code
3. Postman / HTTPie
4. Flutter SDK : [Flutter 3.16.9](#)

MATERI POKOK

PENGERTIAN API

API atau Application Programming Interface adalah sekumpulan perintah, fungsi, serta protocol 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 di analogikan, seperti ketika kita melihat menu di sebuah restoran, kita bisa melihat jenis makanan yang ada beserta penjelasannya. Sebagai pelanggan, kita tidak perlu tahu bagaimana restoran menyiapkan pesanan, karena yang penting adalah mereka mengeluarkan apa yang kita pesan. [Apa Itu API](#)



Client Side App akan melakukan request terhadap server 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.

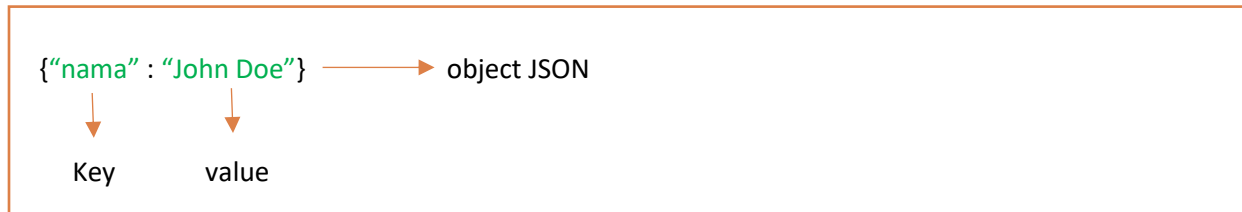
Pada Flutter kita dapat mengambil data di internet menggunakan aplikasi pihak ketiga yang dibuat oleh tim Flutter itu sendiri. Flutter sudah menyediakan cara sederhana untuk mengelola data yang bersumber dari internet menggunakan [http package](#).

JSON (JAVASCRIPT OBJECT NOTATION)

JSON 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

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:



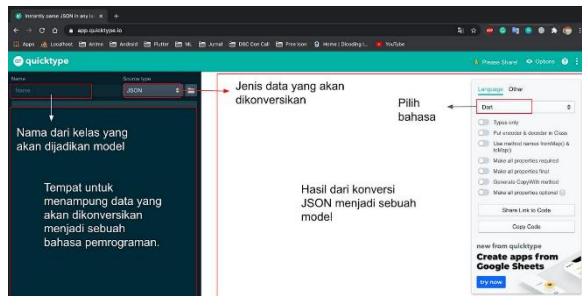
```

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

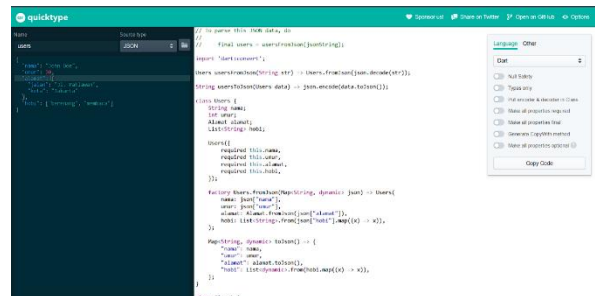
JSON Parsing adalah proses mengambil data dalam **format JSON** dan mengubahnya menjadi struktur data yang dapat digunakan dalam aplikasi. Dalam konteks pengembangan perangkat lunak, **JSON Parsing** adalah langkah penting untuk mengambil informasi yang dikirim oleh server dalam bentuk **JSON** dan mengkonversinya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

QUICKTYPE

[Quicktype](#) merupakan sebuah website untuk melakukan proses generate **class model** dan juga serialisasi yang bersumber dari **JSON**, **schema**, dan **GraphQL**. Dengan memanfaatkan website ini, kita dapat melakukan pekerjaan dengan 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



Tampilan Konversi JSON

PUBLIC API

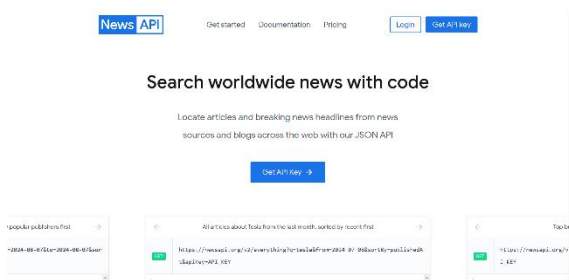
Public API (Application Programming Interface) adalah antarmuka yang disediakan oleh suatu perangkat lunak atau layanan untuk memungkinkan developer lain berinteraksi dengan perangkat lunak atau layanan tersebut. Public API memungkinkan developer luar untuk mengakses dan menggunakan fungsionalitas yang disediakan oleh 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. Terdapat beberapa tempat untuk mengakses API yang tersedia secara gratis dan terbuka yaitu antara lain [AnyAPI](#), [Todd motto](#), [n0shakePublic-APIs](#), [Marcelscruz](#).

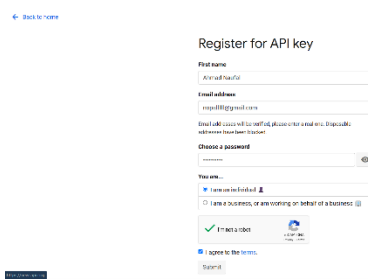
Penggunaan

Mari kita mencoba membuat **class model** untuk Public API kita. Kali ini kita akan menggunakan Public API dari [News API](#).

1. Pada halaman Home pilih **Get API Key** untuk melakukan registrasi akun.

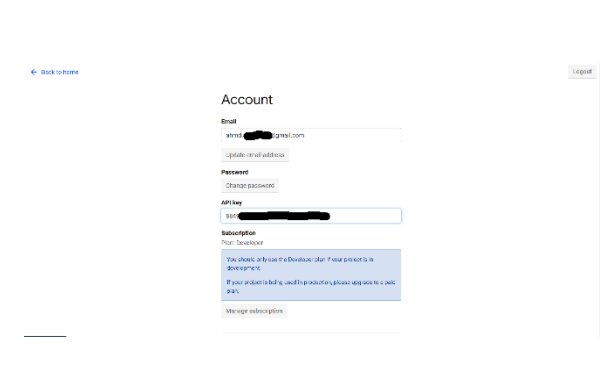


2. Pada halaman registrasi isi data diri dan email yang ingin digunakan.

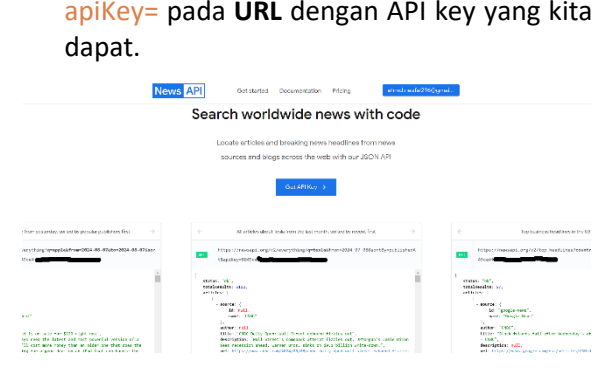


3. Setelah submit akan mendapatkan **API key** melalui **email** atau di **profile News API**.


4. Setelah mendapatkan **API Key**, tentukan URL tautan yang akan digunakan. Kalian bisa **melihat tautan** apa saja yang disediakan di **halaman Home** atau kita tinggal **mengganti**



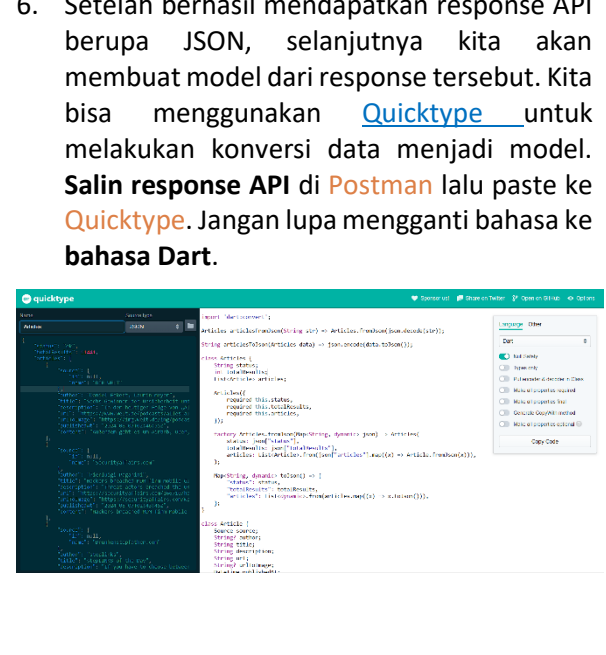
apiKey= pada **URL** dengan API key yang kita dapat.



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 mendapatkan response API berupa **JSON**, selanjutnya kita akan membuat model dari response tersebut. Kita bisa menggunakan **Quicktype** untuk melakukan konversi data menjadi model. **Salin response API** di **Postman** lalu paste ke **Quicktype**. Jangan lupa mengganti bahasa ke **bahasa Dart**.



7. Setelah itu masuk ke project kita lalu buat file **article.dart**. Bisa kalian buat melalui folder **lib/app/data/models/article.dart**, lalu paste model yang sudah dibuat. Untuk lebih detail nya kalian bisa mengakses project online yang sudah kami buat. [Modul 2 Public API Flutter Online Project](#)

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 protocol 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`.

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

<pre>dependencies: cupertino_icons: ^1.0.8 get: 4.6.6 flutter: sdk: flutter</pre>	<pre>dependencies: cupertino_icons: ^1.0.8 get: 4.6.6 # http package http: ^1.2.0 flutter: sdk: flutter</pre>
---	---

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

Penggunaan

Sebelum mengikuti langkah-langkah dibawah ini pastikan kalian sudah memiliki **API key** dari [News API](#).

1. Setelah model response tersedia, kita akan membuat **class ApiService** pada `lib/app/data/services/http_controller.dart`. 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). ***Catatan: Perlu diperhatikan bahwa pada source code yang diberikan menggunakan response dari URL**
https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=YOUR_API_KEY
jadi pastikan class model kalian sesuai dengan response dari URL diatas.

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

import '../models/article.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;
```

```

@override
void onInit() async {
  await fetchArticles();
  super.onInit();
}

Future<void> fetchArticles() async{
  try {
    isLoading.value = 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 = Articles.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;
  }
}
}

```



2. API yang sudah di panggil kemudian akan ditampilkan pada [screen widget](#) atau kalian bisa mengakses [online editor ini](#) untuk lebih detail nya.

DIO PACKAGE

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

Installation

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

1. Buka file [pubspec.yaml](#) pada project.
 -  nama_project
 -  pubspec.yaml

2. Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

<pre>dependencies: cupertino_icons: ^1.0.8 get: 4.6.6 flutter: sdk: flutter</pre>	<pre>dependencies: cupertino_icons: ^1.0.8 get: 4.6.6 # dio package dio: ^5.5.0+1 flutter: sdk: flutter</pre>
--	--

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

Penggunaan

Sebelum mengikuti langkah-langkah dibawah ini pastikan kalian sudah memiliki **API key** dari [News API](https://newsapi.org/).

1. Setelah model response tersedia, kita akan membuat **class ApiService** pada **lib/app/data/services/dio_controller.dart**. 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). ***Catatan: Perlu diperhatikan bahwa pada source code yang diberikan menggunakan response dari URL**

https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=YOUR_API_KEY
jadi pastikan class model kalian sesuai dengan response dari **URL** diatas.

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

import '../models/article.dart';

class DioController extends GetxController {
  static const String _baseUrl = 'https://newsapi.org/v2/';
  static const String _apiKey = '8849ce4f79484316bb3d4e00adfd54ef'; //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;

  final dio = Dio(); //membuat object dio

  @override
  void onInit() async {
    await fetchArticles();
    super.onInit();
  }
}
```



```

}

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 = Articles.fromJson(jsonData);
      articles.value = articlesResult.articles;
    }else{
      print("Request is failed with status ${response.statusCode}");
    }
  } catch (e) {
    print('An error occured: $e');
  }finally{
    isLoading.value = false; //set status loading to false when it done
  }
}
}

```



2. API yang sudah di panggil kemudian akan ditampilkan pada [screen widget](#) atau kalian bisa mengakses [online editor ini](#) untuk lebih detail nya.

GETCONNECT PACKAGE

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 [getconnect-plugin](#) pada file [pubspec.yaml](#).

1. Buka file [pubspec.yaml](#) pada project.
 -  nama_project
 -  [pubspec.yaml](#)

- Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**, jika kalian membuat project Flutter menggunakan GetX command dari awal maka tidak perlu merubah apapun di filenya.

<pre>dependencies: cupertino_icons: ^1.0.8 flutter: sdk: flutter</pre>	<pre>dependencies: cupertino_icons: ^1.0.8 get: 4.6.6 flutter: sdk: flutter</pre>
---	--

- Selanjutnya **GetConnect** sudah dapat dipakai oleh project.

Penggunaan

Sebelum mengikuti langkah-langkah dibawah ini pastikan kalian sudah memiliki **API key** dari [News API](#).

- Setelah model response tersedia, kita akan membuat **class ApiService** pada **lib/app/data/services/dio_controller.dart**. 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). ***Catatan: Perlu diperhatikan bahwa pada source code yang diberikan menggunakan response dari URL**

https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=YOUR_API_KEY
jadi pastikan class model kalian sesuai dengan response dari **URL** diatas.

```
import 'package:get/get.dart';
import '../models/article.dart';

class GetConnectController extends GetConnect {
  static const String _baseUrl = 'https://newsapi.org/v2/';
  static const String _apiKey =
    '8849ce4f79484316bb3d4e00adfd54ef'; // 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;

  @override
  void onInit() {
    fetchArticles();
    super.onInit();
  }

  Future<void> fetchArticles() async {
    try {
      isLoading.value = true; // set loading state to true
```

```

final response = await get(
  '${_baseUrl}top-
  headlines?country=$_country&category=$_category&apiKey=$_apiKey');

if (response.statusCode == 200) {
  final articlesResult = Articles.fromJson(response.body);
  articles.value = articlesResult.articles;
} else {
  print("Request failed with status ${response.statusCode}");
}
} catch (e) {
  print('An error occurred: $e');
} finally {
  isLoading.value = false; // set status loading to false when it done
}
}
}

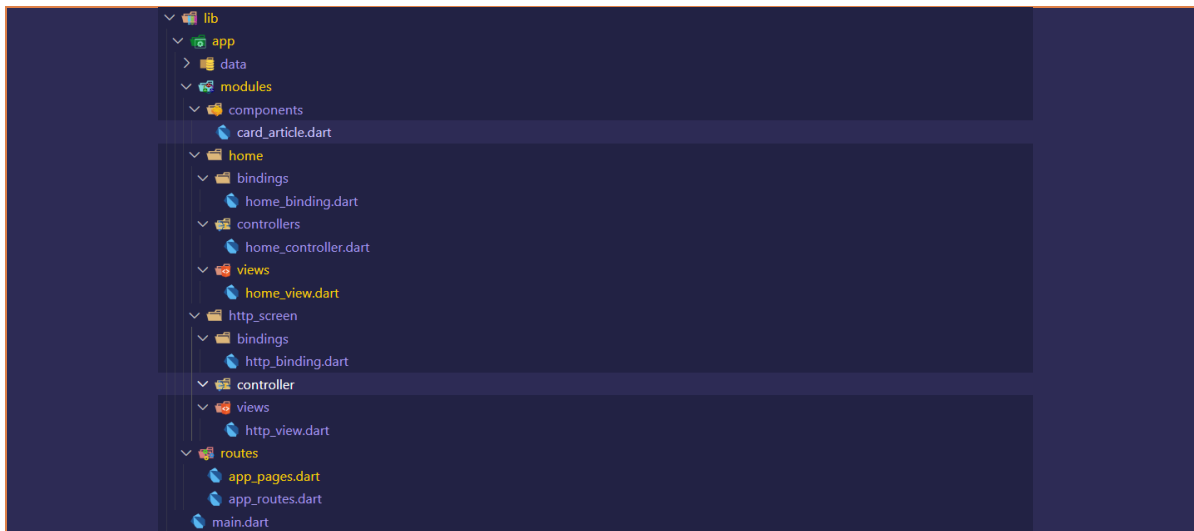
```

2. API yang sudah di panggil kemudian akan ditampilkan pada [screen widget](#) atau kalian bisa mengakses [online editor ini](#) untuk lebih detail nya.

MENAMPILKAN API

Setelah berhasil memanggil API, selanjutnya response akan ditampilkan pada screen widget. Pada contoh source code dibawah ini, menampilkan khusus HTTP package. Jika kalian ingin melihat implementasi package lain, kalian bisa mengakses [online editor ini](#).

1. Sebelum menulis kode, hal yang perlu kita lakukan adalah membuat beberapa **folder** dan **file**. Silahkan ikuti struktur **folder** dan **file** di bawah ini.



2. Setelah membuat **folder** dan **file** yang dibutuhkan. Selanjutnya kita akan membuat sebuah **custom widget** untuk menampilkan **List Card Article** yang ada dengan menulis source code dibawah ini ke dalam file **card_article.dart**.

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul_2_public_api/app/routes/app_pages.dart';
import '../data/models/article.dart';

class CardArticle extends StatelessWidget {
  final Article article;
  const CardArticle({Key? key, required this.article}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.only(bottom: 15),
      child: ListTile(
        contentPadding:
          const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
        leading: Hero(
          tag: article.urlToImage ?? article.title, // Use a fallback tag
          child: SizedBox(
            width: 100,
            height: 100,
            child: article.urlToImage != null
              ? Image.network(
                  article.urlToImage!,
                  fit: BoxFit.cover,
                )
              : Container(
                  color: Colors.grey,
                  child: const Center(
                    child: Text(
                      'No Image',
                      textAlign: TextAlign.center,
                    ),
                  ),
                ),
            ),
          ),
        title: Text(
          article.title,
        ),
        subtitle: Text(article.author ?? 'Unknown'),
        onTap: () {
```

```

        Get.toNamed(Routes.ARTICLE_DETAILS, arguments: article);
    },
),
);
}
}

```

3. Selanjutnya kita akan membuat halaman yang akan menampilkan **list article** menggunakan **HTTP Package**. Untuk source code dibawah ini bisa kalian tambahkan ke dalam file **http_view.dart** dan pastikan untuk **sudah membuat CardArticle**.

```

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import '../data/services/http_controller.dart';
import '../components/card_article.dart';

class HttpView extends GetView<HttpController> {
  const HttpView({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP'),
      ),
      body: Obx(() {
        if (controller.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 ListView.builder(
            shrinkWrap: true,
            itemCount: controller.articles.length,
            itemBuilder: (context, index) {
              var article = controller.articles[index];
              return CardArticle(article: article);
            },
          );
        }
      })
    );
  }
}

```

```

    }));
  }
}

```

4. Setelah menambahkan kode ke dalam file `http_view.dart` jangan lupa untuk menambahkan **Binding**, **AppRoute**, dan **AppPage**.
 - Menambah **Binding** pada file `http_binding.dart`.

```

import 'package:get/get.dart';

import '../data/services/http_controller.dart';

class HttpBinding extends Bindings {
  @override
  void dependencies() {
    Get.lazyPut<HttpController>(
      () => HttpController(),
    );
  }
}

```

- Menambah **Route** pada file `app_routes.dart`.

```

part of 'app_pages.dart';
// DO NOT EDIT. This is code generated via package:get_cli/get_cli.dart

abstract class Routes {
  Routes._();
  static const HOME = _Paths.HOME;
  static const HTTP = _Paths.HTTP;
  static const ARTICLE_DETAILS = _Paths.ARTICLE_DETAILS;
}

abstract class _Paths {
  _Paths._();
  static const HOME = '/home';
  static const HTTP = '/http_view';
  static const ARTICLE_DETAILS = '/article_details';
}

```

- Menambah **Page** pada file `app_pages.dart`.

```
import 'package:get/get.dart';
import
'package:modul_2_public_api/app/modules/http_screen/bindings/http_binding.d
art';
import
'package:modul_2_public_api/app/modules/http_screen/views/http_view.dart';

import '../modules/home/bindings/home_binding.dart';
import '../modules/home/views/home_view.dart';

part 'app_routes.dart';

class AppPages {
  AppPages._();

  static const INITIAL = Routes.HOME;

  static final routes = [
    GetPage(
      name: _Paths.HOME,
      page: () => HomeView(),
      binding: HomeBinding(),
    ),
    GetPage(
      name: _Paths.HTTP,
      page: () => const HttpView(),
      binding: HttpBinding(),
    ),
  ];
}
```

5. Selanjutnya kita akan membuat **Home page** pada file `home_view.dart` yang nantinya kita jadikan sebagai halaman untuk berpindah ke package selain `http`.

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

import 'package:get/get.dart';

// import '../components/appbar_view.dart';
import '../routes/app_pages.dart';
import '../controllers/home_controller.dart';

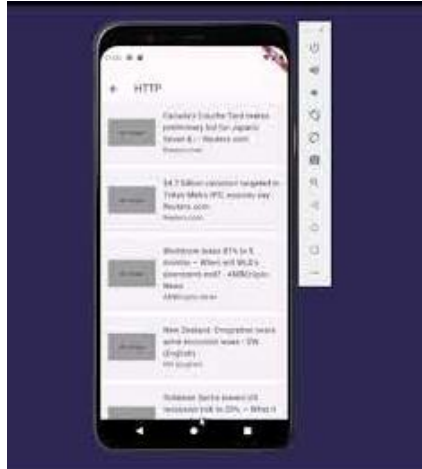
class HomeView extends GetView<HomeController> {
  const HomeView({Key? key}) : super(key: key);
```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("HomePage"),
      actions: [
        PopupMenuButton(
          onSelected: (value) {
            switch (value) {
              case 0:
                Get.toNamed(Routes.HTTP);
                break;
            }
          },
          itemBuilder: (context) => [
            const PopupMenuItem(
              value: 0,
              child: TextButton(
                onPressed: null,
                child: Text("HTTP Page"),
              ),
            ),
          ],
        ),
      ],
    body: const Column(
      mainAxisAlignment: MainAxisAlignment.start,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Text("HOW TO USE THIS APPLICATION"),
        Text("1. Buka icon titik 3 di pojok kanan atas"),
        Text("2. Pilih Package yang ingin kalian liat sebagai
outputnya"),
      ],
    ),
  );
}

```

6. Untuk hasil dari source codenya bisa Anda liat pada video di bawah ini.



WEBVIEW

WebView adalah widget dalam Flutter yang memungkinkan Anda untuk **menampilkan konten web** di dalam aplikasi Flutter Anda. **Widget WebView** sangat berguna ketika Anda ingin meintegrasikan konten web seperti halaman web, halaman login, atau konten lain yang diberikan oleh server ke dalam aplikasi Flutter Anda. Dengan **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](#).

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

<pre>dependencies: cupertino_icons: ^1.0.8 flutter: sdk: flutter</pre>	<pre>dependencies: cupertino_icons: ^1.0.2 get: 4.6.6 #WebView webview_flutter: ^4.8.0 flutter: sdk: flutter</pre>
---	--

3. Selanjutnya **WebView** sudah dapat dipakai oleh project.

Penggunaan

Berikut adalah contoh penggunaan WebView pada source code [http-package](#) sebelumnya.

1. Kita akan membuat **WebViewWidget** terlebih dahulu di file `lib/app/modules/article_detail/views/article_detail_web_view.dart`. Perlu diperhatikan karena kita akan mempassing data dari halaman `article_detail_view.dart` jadi kita harus membuat **constructor**.

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:webview_flutter/webview_flutter.dart';

import '../data/models/article.dart';
import '../controllers/article_detail_controller.dart';

// ignore: must_be_immutable
class ArticleDetailWebView extends GetView<ArticleDetailController> {
  final Article article;
  const ArticleDetailWebView({
    super.key,
    required this.article,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("WebView"),
      ),
      body: WebViewWidget(
        controller: controller.webViewController(article.url),
      ));
  }
}
```

2. Setelah View Screen untuk **Web View** telah di buat, selanjut nya kita akan membuat function **WebViewController** pada file `article_detail_controller.dart` dan membuat **Binding** pada file `article_detail_bindings.dart`.

- **WebView Controller**

```
import 'package:get/get.dart';
import 'package:webview_flutter/webview_flutter.dart';

class ArticleDetailController extends GetxController {
  @override
  void onInit() {
    // TODO: implement onInit
    super.onInit();
  }

  @override
```

```

void onReady() {
  // TODO: implement onReady
  super.onReady();
}

@override
void onClose() {
  // TODO: implement onClose
  super.onClose();
}

WebViewController webViewController(String uri) {
  return
  WebViewController()..setJavaScriptMode(JavascriptMode.unrestricted)..loadRequest(Uri.parse(uri));
}
}

```

- **WebView Binding**

```

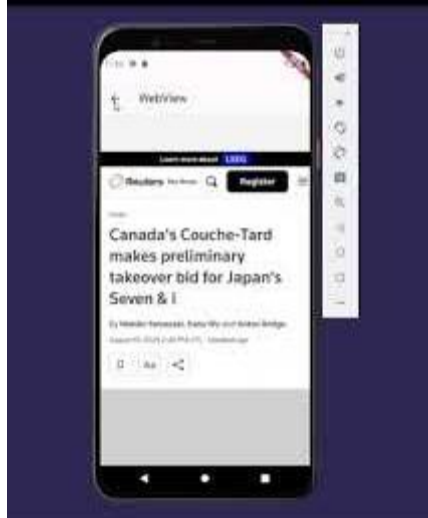
import 'package:get/get.dart';

import '../controllers/article_detail_controller.dart';

class ArticleDetailBinding extends Bindings {
  @override
  void dependencies() {
    Get.lazyPut(() => ArticleDetailController());
  }
}

```

3. Terakhir kita perlu menambahkan **AppRoutes** dan **AppPages** untuk **WebView Page**. Caranya sama seperti kita menambah **Routes** dan **Page** pada [Implementasi API](#).
4. Untuk hasil dari source codenya bisa Anda liat pada video di bawah ini.



Untuk memahami lebih detail atau ingin membandingkan kode yang telah kalian buat, bisa mengakses [online editor ini](#).

LINK GITHUB

Berikut ini merupakan link github yang akan digunakan sebagai contoh latihan pada modul 2 pemrograman mobile.

[Pemrograman-Mobile-2024-GitHub](#)

CODE LAB

TUGAS 1 – INSTALASI PACKAGE DAN WEBVIEW

1. Lakukan **instalasi package** (http, Dio, GetConnect) bebas sesuai keinginan kalian, selain itu juga install **WebView** dengan menambahkan dependencies di file `pubspec.yaml`

TUGAS 2 – LATIHAN IMPLEMENTASI HTTP, DIO, GETCONNECT DAN WEBVIEW

1. Lakukan latihan dari salah satu dari 3 package [http-package](#), [dio-package](#), [getconnect-package](#). API yang dipakai yaitu melalui <https://my-json-server.typicode.com/Fallid/codelab-api/db>.
2. Lakukan konversi pada data JSON menjadi sebuah model menggunakan aplikasi [QuickType](#).
3. Implementasikan penggunaan salah satu dari 3 package tersebut.
4. Buatlah **User Interface** untuk menampilkan data dari API yang sudah di ambil.
5. Implementasikan penggunaan [WebView](#) sederhana. Untuk URL website yang ingin ditampilkan bebas.
6. Untuk mempermudah pengerjaan Latihan CodeLab, kalian bisa melihat referensi langsung dari [online editor ini](#)

KRITERIA & DETAIL PENILAIAN

Bobot Penilaian Modul 2 Materi (15%)

Berhasil melakukan instalasi package dan webview.	10
---	----

Berhasil melakukan Latihan http dan WebView.	70
Menggunakan state management pada Latihan.	20
Total	100