

Nama : zaida
Kelas : SIB-3D
NIM : 2241760130
Mata Kuliah : Pemrograman Mobile
Github : https://github.com/arabgg/P_Mobile

Weather app using API integration in Flutter

Fetching the packages

| Langkah | Deskripsi |
|---------|---|
| 1 | <pre>PS C:\laragon\www\Mobile_2024\weatherappclima> flutter pub add http Resolving dependencies... Downloading packages... async 2.11.0 (2.12.0 available) boolean_selector 2.1.1 (2.1.2 available) characters 1.3.0 (1.3.1 available) clock 1.1.1 (1.1.2 available) collection 1.18.0 (1.19.1 available) fake_async 1.3.1 (1.3.2 available) flutter_lints 4.0.0 (5.0.0 available) + http 1.2.2 + http_parser 4.0.2 (4.1.1 available)</pre> <p>Install paket http</p> <p>Paket http di Flutter digunakan untuk melakukan HTTP requests, seperti:</p> <ul style="list-style-type: none">• Mengambil data dari server/API (GET request): Mengambil data JSON dari server, misalnya untuk mendapatkan data cuaca, berita, atau produk dari API. |

| | |
|---|--|
| | <ul style="list-style-type: none"> • Mengirim data ke server (POST request): Mengirim data ke server, misalnya mengisi formulir, mendaftarkan pengguna, atau mengunggah data. • Menghapus data di server (DELETE request): Menghapus data tertentu dari server, seperti menghapus catatan pengguna. • Mengubah data di server (PUT atau PATCH request): Memperbarui data tertentu di server, misalnya memperbarui informasi profil pengguna. |
| 2 | Install paket geolocator |

| | |
|--|---|
| | <pre>PS C:\laragon\www\Mobile_2024\weatherappclima> flutter pub add geolocator Resolving dependencies... Downloading packages... async 2.11.0 (2.12.0 available) boolean_selector 2.1.1 (2.1.2 available) characters 1.3.0 (1.3.1 available) clock 1.1.1 (1.1.2 available) collection 1.18.0 (1.19.1 available) + crypto 3.0.6 fake_async 1.3.1 (1.3.2 available) + fixnum 1.1.1 flutter_lints 4.0.0 (5.0.0 available)</pre> |
|--|---|

| | |
|---|--|
| 3 | <pre>PS C:\laragon\www\Mobile_2024\weatherappclima> flutter pub get Resolving dependencies... Downloading packages... async 2.11.0 (2.12.0 available) boolean_selector 2.1.1 (2.1.2 available) characters 1.3.0 (1.3.1 available) clock 1.1.1 (1.1.2 available) collection 1.18.0 (1.19.1 available) fake_async 1.3.1 (1.3.2 available) flutter_lints 4.0.0 (5.0.0 available) http_parser 4.0.2 (4.1.1 available) leak_tracker 10.0.5 (10.0.8 available)</pre> <p>menginstal dan mengelola dependensi</p> |
| 4 | <p>Tambahkan yang berikut ke file "gradle.properties"</p> <pre>2 android.useAndroidX=true 3 android.enableJetifier=true</pre> |
| 5 | <pre>8 android { 9 namespace = "com.example.weatherappclima" 10 compileSdk = 34 // Set compileSdkVersion ke 34 11 }</pre> <p>menyetel compileSdkVersionfile "android/app/build.gradle" ke 34</p> |

| | |
|---|---|
| 6 | <p>Di Android, Anda perlu menambahkan izin ACCESS_COARSE_LOCATION atau ACCESS_FINE_LOCATION ke Android Manifest Anda. Untuk melakukannya, buka file AndroidManifest.xml (terletak di bawah android/app/src/main) dan tambahkan salah satu dari dua baris berikut sebagai turunan langsung dari <manifest> tag (saat Anda mengonfigurasi kedua izin, ACCESS_FINE_LOCATION akan digunakan oleh plugin geolocator):</p> <pre> 3 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> 4 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /> </pre> <p>Dimulai dari Android 10, Anda perlu menambahkan ACCESS_BACKGROUND_LOCATION izin (di samping ACCESS_COARSE_LOCATION atau ACCESS_FINE_LOCATION) jika</p> |
| | <p>Anda ingin terus menerima pembaruan bahkan saat Aplikasi Anda berjalan di latar belakang:</p> <pre> 5 <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" /> </pre> |

Building the App

| Langkah | Deskripsi |
|---------|--|
| 1 | <p>Dalam file main.dart default, kita akan menghapus kode yang tidak diperlukan dan menyimpan kode yang diperlukan saja. Di sini, kita akan bekerja dengan MaterialApp, bukan MyApp. Oleh karena itu, kita menghapus referensi MyApp di main.dart dan menghapus file pengujian di bagian proyek. Setelah referensi MyApp dihapus dan diganti dengan MaterialApp() sebagai widget root, main.dart akan terlihat seperti di bawah ini</p> <pre> lib > main.dart > main Run Debug Profile 1 import 'package:flutter/material.dart'; void main() 2 runApp(3 MaterialApp(), 4); 5 } </pre> |

| | |
|---|---|
| 2 | <p>Di dalam berkas dart ini, impor paket material.dart. Sekarang kita akan membuat widget Stateful untuk membangun aplikasi kita. Pintasan untuk membuatnya adalah dengan mengetikkan stful dan kita akan mendapatkan kode kerangka widget</p> <pre> lib > 📄 homescreen.dart > ... 1 import 'package:flutter/material.dart'; 2 3 class HomeScreen extends StatefulWidget { 4 @override 5 _YourWidgetNameState createState() => _YourWidgetNameState(); 6 } 7 class _YourWidgetNameState extends State<HomeScreen> { 8 @override 9 Widget build(BuildContext context) { 10 return Container(); 11 } 12 } 13 </pre> <p>Stateful. Ganti YourWidgetName dengan nama kustom Anda sendiri.</p> |
| 3 | <p>Sekarang setelah selesai, impor berkas dart ini di main.dart dan tetapkan properti home MaterialApp() ke nama yang diberikan untuk widget Stateful yang dibuat. Properti home digunakan untuk menampilkan layar awal dalam suatu aplikasi saat</p> |

aplikasi hanya melibatkan satu layar. Di sini saya telah memberi nama sebagai HomeScreen() dan nama berkas dart sebagai homescreen.dart

```
lib > main.dart > main
1  import 'package:flutter/material.dart';
2  import 'homescreen.dart';
   Run | Debug | Profile
3  void main() {
4    runApp(
5      MaterialApp(
6        debugShowCheckedModeBanner: false,
7        home: HomeScreen(),
8        theme: ThemeData(
9          primaryColor: Colors.white,
10         hintColor: Colors.white,
11       ), // ThemeData
12     ), // MaterialApp
13   );
14 }
```

- 4 Langkah 1: Mendapatkan koordinat lokasi saat ini
Di dalam homescreen.dart kita akan mendapatkan lintang dan bujur lokasi saat ini. Lintang dan bujur tidak ditampilkan dalam versi akhir aplikasi, jadi kita akan mencoba mencetak nilai-nilai di terminal.
Untuk ini, pertama-tama kita harus mengimpor paket geolocator di homescreen.dart.

```
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
class HomeScreen extends StatefulWidget {
  @override
  _YourWidgetName createState() => _YourWidgetName();
}
class _YourWidgetName extends State<
HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
  getCurrentLocation() async {
    var p = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.low,
      forceAndroidLocationManager: true,
    );
    if (p != null) {
      print('Lat:${p.latitude}, Long:${p.longitude}');
    } else {
      print('Data unavailable');
    }
  }
}
```

```
}  
}  
  
}
```

Ketika metode `getCurrentLocation()` dari paket `Geolocator` dipanggil, metode tersebut akan mengembalikan nilai Posisi (yang terdiri dari lintang, bujur, akurasi, ketinggian, arah, dll.). Karena waktu untuk mendapatkan nilai Posisi tidak diketahui dan dapat diperoleh kapan saja di masa mendatang, kita menggunakan kata kunci `await`. Ini berarti fungsi tersebut bekerja secara asinkron dari eksekusi yang tersisa. Akurasi posisi dapat diatur oleh properti `desirableAccuracy` dan pengelola lokasi android dapat diatur oleh properti `forceAndroidLocationManager`. Nilai Posisi yang dikembalikan terdiri dari detail lokasi perangkat saat ini. Untuk mendapatkan garis lintang dan garis bujur, parameter yang sesuai dari nilai Posisi dipanggil dan dicetak sesuai dengan itu. Fungsi ini kemudian dipanggil dalam metode `initState()` dari widget `Stateful()` seperti yang ditunjukkan

```

import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
class HomeScreen extends StatefulWidget
{
  @override
  _HomeScreenState createState() => _HomeScreenState();
}
class _HomeScreenState extends
State<HomeScreen> {
  @override
  void initState() {
    // TODO: implement initState
    super.initState();    getCurrentLocation();
  }

  Future<void> getCurrentLocation() async {
    try {
      var position = await Geolocator.getCurrentPosition(
desiredAccuracy: LocationAccuracy.low,
forceAndroidLocationManager: true,
      );
      if (position != null) {
        print('Lat: ${position.latitude}, Long:
${position.longitude}');
      } else {
        print('Data
unavailable');
      }
    } catch (e) {
      print('Error: $e');
    }
  }

  @override
  Widget build(BuildContext context)
  {
    return SafeArea(
      child:
Scaffold(
        )
      );
  }
}


```

```

    }
  } catch (e) {
    print('Error: $e');
  }
}

@override
Widget build(BuildContext context)
{
  return SafeArea(
    child:
Scaffold(
      )
    );
}
}

```


| | |
|---|--|
| | <p>Dengan ini jalankan aplikasinya dan kita akan mendapatkan lintang dan bujur</p> <pre>I/flutter (1969): Lat: -7.9450296, Long: 112.615551</pre> <p>lokasi kita saat ini.</p> |
| 5 | <p>LANGKAH 2: Mendapatkan kunci API dari OpenWeatherMap</p> <p>Untuk mendapatkan data cuaca terkini, kami akan menggunakan API OpenWeatherMap. Untuk mengakses data API, kami memerlukan kunci API.</p>  <p>Untuk melakukannya, pilih bagian Harga dari menu atas.</p> |

Kemudian gulir ke bawah hingga Anda melihat berbagai rencana untuk cuaca

Current weather and forecasts collection

| Free | Startup | Developer | Professional | Enterprise |
|---|---|---|---|---|
| | 30 GBP/ month | 140 GBP/ month | 370 GBP/ month | from 1500 GBP/ month |
| Get API key | Subscribe | Subscribe | Subscribe | Subscribe |
| 60 calls/minute 1,000,000 calls/month | 600 calls/minute 10,000,000 calls/month | 3,000 calls/minute 100,000,000 calls/month | 30,000 calls/minute 1,000,000,000 calls/month | 200,000 calls/minute 5,000,000,000 calls/month |
| Current Weather 3-hour Forecast 5 days Hourly Forecast 4 days Daily Forecast 16 days Climatic Forecast 30 days Bulk Download | Current Weather 3-hour Forecast 5 days Hourly Forecast 4 days Daily Forecast 16 days Climatic Forecast 30 days Bulk Download | Current Weather 3-hour Forecast 5 days Hourly Forecast 4 days Daily Forecast 16 days Climatic Forecast 30 days Bulk Download | Current Weather 3-hour Forecast 5 days Hourly Forecast 4 days Daily Forecast 16 days Climatic Forecast 30 days Bulk Download | Current Weather 3-hour Forecast 5 days Hourly Forecast 4 days Daily Forecast 16 days Climatic Forecast 30 days Bulk Download |
| Basic weather maps Historical maps | Basic weather maps Historical maps | Advanced weather maps Historical maps | Advanced weather maps Historical maps | Advanced weather maps Historical maps |

terkini. Pilih paket Gratis dan klik opsi Dapatkan kunci API.

Setelah selesai, akan muncul jendela baru yang mengharuskan Anda membuat akun. Berikan informasi yang diperlukan dan buat akun Anda.

Create New Account

Setelah akun dibuat, akan muncul pop-up yang menanyakan tujuan penggunaan kunci API kita

How and where will you use our API?

X

Hi! We are doing some housekeeping around thousands of our customers. Your impact will be much appreciated. All you need to do is to choose in which exact area you use our services.

Company

* Purpose

Choose answer



Cancel

Save

Di sini, menentukan perusahaan bersifat opsional, sedangkan menentukan tujuan bersifat wajib. Jika Anda tidak tahu tujuan mana yang harus dipilih, Anda dapat memilih Pendidikan/Sains. Lalu, klik simpan.

Sekarang, di dasbor Anda, buka bagian kunci API. Di sini, Anda akan menemukan semua kunci API di bawah bagian Kunci. Harap dicatat bahwa kunci API di sini bersifat unik bagi Anda dan Anda tidak boleh membagikannya kepada orang lain.

New Products Services **API keys** Billing plans Payments Block logs My orders My profile Ask a question

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

| Key | Name | Status | Actions | Create key |
|---|---------|--------|---|---|
|  | Default | Active |   | <input type="text" value="API key name"/> <button>Generate</button> |

Setelah Anda mendapatkan kunci API, Anda dapat melanjutkan untuk memanggil API guna mendapatkan data cuaca. Ada berbagai format pemanggilan API dan bagaimana responsnya, yang semuanya dapat dirujuk di sini. Untuk tujuan kita, kita akan memanggil API melalui format lintang bujur dan format nama kota seperti yang diberikan di bawah ini **API call by latitude longitude format**

`https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}`

API call by city name format

https://api.openweathermap.org/data/2.5/weather?q={ city name }&appid={ API

GET

https://api.openweathermap.org/data/2.5/weather?lat=44.34&lon=10.99&appid=2635cea49537c43390794b07ee1ca8b4

Params

Authorization

Headers (7)

Body

Scripts

Settings

Query Params

| <input checked="" type="checkbox"/> | Key | Value | Description |
|-------------------------------------|-------|----------------------------------|-------------|
| <input checked="" type="checkbox"/> | lat | 44.34 | |
| <input checked="" type="checkbox"/> | lon | 10.99 | |
| <input checked="" type="checkbox"/> | appid | 2635cea49537c43390794b07ee1ca8b4 | |
| | Key | Value | Description |

Body

Cookies

Headers (9)

Test Results

200 OK

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"coord": {

3

"lon": 10.99,

4

"lat": 44.34

5

},

6

"weather": [

7

{

8

"id": 800,

9

"main": "Clear",

10

"description": "clear sky",

11

"icon": "01d"

12

}

]

}

key}

| | |
|--|---|
| | <p>Karena kedua panggilan API memiliki kunci dan domain API yang sama, kami akan menyimpan nilai-nilai tersebut dalam file dart baru bernama constants di bawah folder lib. Untuk membuat file, lihat bagian Membangun Aplikasi di atas. Dalam file</p> |
|--|---|

constants.dart yang dibuat, tambahkan baris kode berikut. `const String domain = "https://api.openweathermap.org/data/2.5/weather?";` `const String apiKey = "PASTE YOUR API KEY HERE";`
Sekarang kami akan mencoba memperoleh data cuaca yang akan menjadi respons JSON.

```
lib > constants.dart > ...  
1  const String domain = "https://api.openweathermap.org/data/2.5/weather?";  
2  const String apiKey = "PASTE_YOUR_API_KEY_HERE";  
3
```

LANGKAH 3: Mendapatkan data cuaca dari lokasi saat ini

Setelah berhasil menyelesaikan langkah-langkah di atas, kita akan mencoba mendapatkan data cuaca. Untuk ini di dalam `homescreen.dart` kita akan menulis fungsi untuk menghubungkan aplikasi kita ke internet terlebih dahulu, lalu memperoleh data cuaca. Di sinilah kita akan mengimplementasikan paket `http`. Jadi pertama-tama kita mengimpor paket sebagai `http` sehingga menjadi lebih mudah untuk mengakses berbagai bidang dalam paket

```
73  import 'package:flutter/material.dart';  
74  import 'package:geolocator/geolocator.dart';  
75  import 'package:http/http.dart' as http;  
76  import 'constants.dart' as k;  
77  import 'dart:convert';
```

Demikian pula kita juga mengimpor `constants.dart` tempat kita menyalin kunci API dan tautan domain. Bersamaan dengan itu, saat kita berurusan dengan mendapatkan satu respons JSON untuk setiap panggilan, kita menggunakan pustaka konversi dart untuk mendekode respons JSON yang kita peroleh.

Panggilan API dengan format lintang bujur

<https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}>

Selanjutnya kita akan menulis fungsi untuk menghubungkan aplikasi kita ke internet. Fungsi ini juga akan asynchronous karena mengembalikan masa depan. di sini kita pertama-tama membuat objek Klien sehingga kita tidak perlu membuka dan menutup port setiap kali kita memanggil metode `get`. Kemudian kita menyediakan alamat URI (URI adalah urutan karakter yang membantu mengidentifikasi sumber daya logis atau fisik yang terhubung ke internet) yang merupakan format panggilan API yang menyediakan bidang yang diperlukan.

```
import 'package:flutter/material.dart';  
import 'package:geolocator/geolocator.dart';  
import 'package:http/http.dart' as http;  
import 'constants.dart' as k; import  
'dart:convert';
```



```

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
} class _HomeScreenState extends
State<HomeScreen> {
  @override void
initState() {
super.initState();
getCurrentLocation();
  }

  Future<void> getCurrentLocation() async { try {
var position = await Geolocator.getCurrentPosition(
desiredAccuracy: LocationAccuracy.low,
forceAndroidLocationManager: true,
); if (position != null) {
print('Lat: ${position.latitude}, Long:
${position.longitude}');
fetchWeather(position.latitude, position.longitude);
} else { print('Data
unavailable');
}
} catch (e) {
print('Error: $e');
}
}

Future<void> fetchWeather(double lat, double lon) async {
final url =
'${k.domain}lat=$lat&lon=$lon&appid=${k.apiKey}&units=metric';
try { final response = await http.get(Uri.parse(url));
if (response.statusCode == 200) { final data =
json.decode(response.body); // Menampilkan data JSON
lengkap di konsol
print(jsonEncode(data)); // Untuk melihat JSON
lengkap seperti di gambar } else {
print('Error: ${response.statusCode}'); }
} catch (e) {
print('Error: $e');
}
}

```

```

    }
  }

  @override
  Widget build(BuildContext context)
  {
    return SafeArea(
      child: Scaffold(
        appBar: AppBar(
          title: Text('Weather App'),
        ),
        body: Center(
          child: Text('Mengambil Cuaca...'),
        ),
      ),
    );
  }
}

```

```

Restarted application in 1,706ms.
I/flutter ( 7672): Lat: -7.94503, Long: 112.6155355
I/flutter ( 7672): {"coord":{"lon":112.6155,"lat":-7.945},"weather":[{"id":501,"main":"Rain","description":"moderate rain","icon":"10n"}],"base":"stations","main":{"temp":26.3,"feels_like":26.3,"temp_min":26.3,"temp_max":26.3,"pressure":1009,"humidity":90,"sea_level":1009,"grnd_level":953},"visibility":10000,"wind":{"speed":0.67,"deg":353,"gust":0.99},"rain":{"1h":2.49},"clouds":{"all":100},"dt":1731754348,"sys":{"type":2,"id":2096469,"country":"ID","sunrise":1731708001,"sunset":1731752931,"timezone":25200,"id":1636722,"name":"Malang","cod":200}
D/DecorView[]( 7672): onWindowFocusChanged hasWindowFocus false
W/MiuiMagicPointerUtilsStubHeadImpl( 7672): MiuiMagicPointerUtilsStubHeadImpl has been initialized !!

```

LANGKAH 4: Mendapatkan data cuaca dari berbagai kota

Mirip dengan LANGKAH 3, kami akan menerapkan pengambilan data cuaca dari kota tertentu berdasarkan nama kota. Di sini, satu-satunya perbedaan adalah alihalih memberikan garis lintang dan garis bujur dalam panggilan API, kami memberikan nama kota.

API call by city name format

<https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}>

```

I/flutter ( 7034): {"coord":{"lon":106.8451,"lat":-6.2146},"weather":[{"id":701,"main":"Mist","description":"mist","icon":"50n"}],"id":501,"main":"Rain","description":"moderate rain","icon":"10n"}],"base":"stations","main":{"temp":299.59,"feels_like":299.59,"temp_min":299.11,"temp_max":304.77,"pressure":1007,"humidity":94,"sea_level":1007,"grnd_level":1005},"visibility":2000,"wind":{"speed":1.54,"deg":300},"clouds":{"all":40},"dt":1731757654,"sys":{"type":1,"id":9383,"country":"ID","sunrise":1731709531,"sunset":1731754172,"timezone":25200,"id":1642911,"name":"Jakarta","cod":200}

```

```
import 'package:flutter/material.dart'; import
'package:http/http.dart' as http;
import 'constants.dart'; // Import file constants.dart
import 'dart:convert';
class HomeScreen extends StatefulWidget
{
  @override
```

```

    _HomeScreenState createState() => _HomeScreenState();
  }
  class _HomeScreenState extends State<HomeScreen> {
    String weatherInfo = "Mengambil data cuaca...";

    Future<void> getCityWeather() async {      final url =
Uri.parse(weatherUrl); // Mengambil URL dari constants.dart      try
{      final response = await http.get(url);      if
(response.statusCode == 200) {      final data =
json.decode(response.body);      print(jsonEncode(data)); //
Menampilkan data JSON lengkap di konsol
      } else {      print('Error:
${response.statusCode}');      }
    } catch (e) {
      setState(() {      weatherInfo
= "Error: $e";
      });
      print('Error: $e');
    }
  }

  @override void initState() {      super.initState();
getCityWeather(); // Panggil fungsi untuk mendapatkan data cuaca
}

  @override
  Widget build(BuildContext context)
  {      return SafeArea(      child:
Scaffold(      appBar: AppBar(
        title: Text('Weather App'),
      ),      body: Center(
child: Padding(      padding: const
EdgeInsets.all(16.0),      child: Text(
weatherInfo,

```

```

        style: TextStyle(fontSize: 18),
        textAlign: TextAlign.center,
      ),
    ),
  ),
);
}
}

```

```

const String cityName = "Jakarta"; const String apiKey =
"2635cea49537c43390794b07ee1ca8b4"; // API Key const String
weatherUrl =
"https://api.openweathermap.org/data/2.5/weather?q=$cityName&appid=$a
piKey";

```

LANGKAH 5: Menyelesaikan Pembuatan

Sekarang setelah kita memperoleh data cuaca dan data lokasi, kita akan melanjutkan untuk membuat UI aplikasi. Jadi di sini kita akan menggunakan kartu pada latar belakang gradien untuk menampilkan kondisi cuaca. Untuk itu, pertama-tama kita menyediakan badan Scaffold() yang merupakan Container() yang menutupi seluruh layar dan yang memiliki isian gradien. Di sini saya telah menggunakan gradien dari Gradient Backgrounds yang memiliki koleksi besar gradien latar belakang beserta kode warna yang sesuai. Berikut ini adalah gradien yang saya pilih.

8:39

0.89
KB/d



Mengambil data cuaca...

Kode untuk memperoleh gradien latar belakang lengkap seperti yang ditunjukkan di bawah ini. Di sini kita akan menyetel properti `resizeToAvoidBottomInset` dari widget `Scaffold()` ke `false` sehingga perubahan ukuran widget saat keyboard muncul dapat dihindari.

```
Widget build(BuildContext context)
{
  return SafeArea(
    child: Scaffold(
      body: Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              Color(0xFF8FF43F),
              Color(0xFF16A085),
            ],
          ),
        child: Center(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Text("Mengambil data cuaca...",
              style: TextStyle(
                fontSize: 18,
                color: Colors.white,
              ),
            ),
            textAlign: TextAlign.center,
          ),
        ),
      ),
    ),
  );
};
```

Kemudian kami akan menyediakan beberapa variabel untuk menyimpan nilai suhu, tekanan, kelembaban, tutupan awan, nama kota dan status data, yaitu apakah data tersebut diambil dan siap digunakan oleh aplikasi. Variabel-variabel ini dideklarasikan di dalam widget `Stateful` sebelum metode `initState`.

```
bool isLoading = false;
num temp = 0;  num
press = 0;  num hum =
0;  num cover = 0;
String cityname = '';
```



```
temp = data['main']['temp'];
press = data['main']['pressure'];
hum = data['main']['humidity'];
cover = data['clouds']['all'];
cityname = data['name'];
isLoading = true;
```



Selanjutnya kita akan menambahkan widget Visibilitas sebagai anak dari widget Kontainer. Jadi hanya saat ada data data cuaca akan ditampilkan, selain itu akan ditampilkan indikator pemuatan. Untuk ini nilai isLoading diubah secara dinamis dalam fungsi menggunakan metode setState. Kodenya seperti yang diberikan di bawah ini.

```
child: Visibility(
  visible: isLoading,
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
```

```

        children: [
          Text(
            "Kota: $cityname",
            style: TextStyle(
              fontSize: 18,
              color: Colors.white,
            ),
          ),
          SizedBox(height: 8),
          Text(
            "Suhu:
            ${temp.toStringAsFixed(1)}°C",
            style: TextStyle(
              color: Colors.white,
              fontSize: 18,
            ),
          ),
          SizedBox(height: 8),
          Text(
            "Tekanan: ${press}
            hPa",
            style: TextStyle(
              color:
              Colors.white,
            ),
          ),
          SizedBox(height: 8),
          Text(
            "Kelembapan:
            ${hum}%",
            style:
            TextStyle(
              fontSize: 18,
              color: Colors.white,
            ),
          ),
          SizedBox(height: 8),
          Text(
            "Tutupan Awan:
            ${cover}%",
            style:
            TextStyle(
              fontSize:
              18,
              color:
              Colors.white,
            ),
          ),
        ],
      ),
    ),
  ],
  replacement: Center(
    child:
    CircularProgressIndicator(

```

| | |
|--|--|
| | |
|--|--|

Sekarang, untuk menyimpan data cuaca ke dalam variabel, kita akan menyediakan fungsi lain untuk memperbarui variabel. Fungsi ini akan memiliki data JSON yang didekode sebagai parameter dan berdasarkan nilainya, nilai parameter ditetapkan.

```
void updateUI(var decodedData) {  
  setState(() {  
    if  
    (decodedData == null) {  
      temp = 0;          press = 0;  
      hum = 0;          cover = 0;  
      cityname = 'Not available';  
      isLoading = false;  
    } else {  
      temp =  
      decodedData['main']['temp'] - 273;  
      press = decodedData['main']['pressure'];  
      hum = decodedData['main']['humidity'];  
      cover = decodedData['clouds']['all'];  
      cityname = decodedData['name'];  
      isLoading  
      = true;  
    }  
  })  
}
```

Fungsi ini akan dipanggil di kedua fungsi pemanggil API jika kode status data yang diterima adalah 200

```
Future<void> getCurrentCityWeather() async {  
  final url = Uri.parse(weatherUrl);  
  try {  
    final response = await http.get(url);  
    if  
    (response.statusCode == 200) {  
      final  
      data = json.decode(response.body);  
      updateUI(data);  
      setState(() {  
        isLoading = true;  
      });  
    } else {  
      print("Error: ${response.statusCode}");  
    }  
  }  
}
```

Kode yang disorot dengan huruf tebal ditambahkan ke fungsi. Pada dasarnya, kode tersebut mendekode data JSON dan memanggil metode updateUI yang menetapkan nilai variabel ke data cuaca terkait. Bersamaan dengan itu, variabel yang mewakili status data yang diperoleh juga diperbarui di seluruh proses dengan bantuan metode setState. Demikian pula untuk fungsi yang menggunakan panggilan API dengan nama kota, kode yang sama ditambahkan.

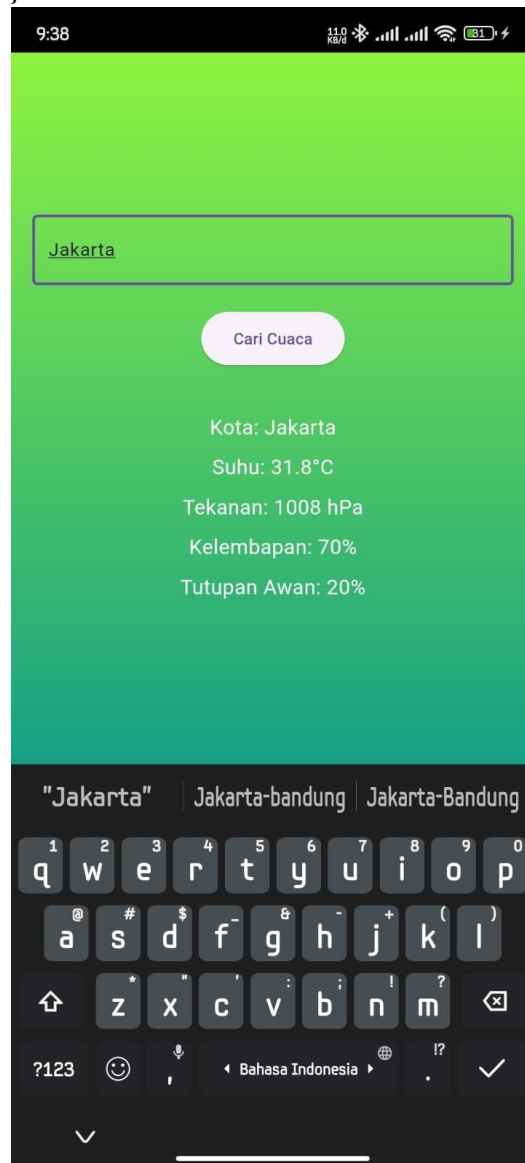
```
getCityWeather(String cityname) async {  
  //Networking code
```



```

if (response.statusCode == 200) {
  var data = response.body; var
  decodeData = json.decode(data);
  updateUI(decodeData);
  setState(() {
    isLoading = true;
  }); }
else {
  print(response.statusCode);
}
}

```



Di dalam kolom widget Visibilitas, kita akan menambahkan TextFormField sebagai anak pertama. Untuk pengontrol TextFormField ini, pengontrol juga

| | |
|--|---|
| | <p>disediakan, yang dideklarasikan bersama dengan variabel. <code>TextEditingController</code> <code>controller = TextEditingController();</code></p> |
|--|---|

| | |
|--|---|
| | <p>Kode TextFormField seperti yang diberikan di bawah ini. Perhatikan bahwa kami telah menetapkan nilai isLoading menjadi false setelah nama kota dimasukkan. Ini memberikan indikator pemuatan kepada pengguna saat mengambil data, sehingga pengguna akan tahu ada beberapa proses yang sedang berlangsung.</p> |
|--|---|

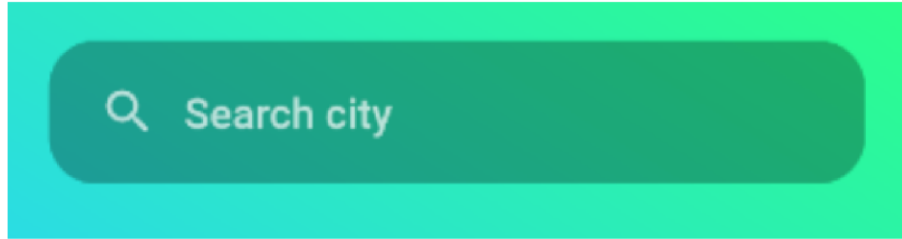

```

child: Column(
    mainAxisAlignment:
MainAxisAlignment.center,
    children: [
Container(
    width:
MediaQuery.of(context).size.width * 0.85,
height: MediaQuery.of(context).size.height *
0.09,
padding:
EdgeInsets.symmetric(horizontal: 10),
decoration: BoxDecoration(
    color:
Colors.black.withOpacity(0.3),
borderRadius: BorderRadius.all(
    Radius.circular(20),
),
),
child: Center(
    child:
TextFormField(
onFieldSubmitted: (String s) {
    setState(() {
        cityname =
s;
        getCityWeather(s);
isLoaded = false;
controller.clear();
    });
},
    controller:
controller,
    cursorColor:
Colors.white,
    style:
TextStyle(
        fontSize:
20,
        fontWeight: FontWeight.w600,
color: Colors.white),
    decoration:
InputDecoration(
        hintText:
'Search city',
        hintStyle:
TextStyle(
            fontSize: 18,
            color: Colors.white.withOpacity(0.7),
fontWeight: FontWeight.w600,
),
        prefixIcon: Icon(
Icons.search_rounded,
size: 25,
color: Colors.white.withOpacity(0.7),

```

```
        ),
        border: InputBorder.none,
      ),
    ),
  ),
),
```

Karena lebar tidak dapat ditentukan dalam TextFormField, lebarnya dibungkus di dalam wadah.



Setelah itu, untuk penggunaan memori minimum, kami akan membuang pengontrol dalam metode pembuangan widget Stateful seperti yang ditunjukkan di bawah ini.

```
@override
void dispose() {
    controller.dispose();
    super.dispose();
}
```

Berikutnya kita akan menambahkan anak berikutnya dari `Column()` yang merupakan widget `SizedBox` untuk menambahkan ruang yang diperlukan di antara komponen-komponen.

```
SizedBox(height: 30),
```

Hal ini diikuti oleh data nama Kota.



Hal ini diimplementasikan dengan menggunakan widget Row() yang dibungkus dengan widget Padding. Widget ini terdiri dari Ikon dan Teks dan kodenya seperti yang diberikan di bawah ini

```
Padding(  
    padding: const EdgeInsets.all(8.0),  
    child: Row(  
        crossAxisAlignment:  
  
CrossAxisAlignment.end,  
        children: [  
            Icon(  
                Icons.pin_drop,  
                color: Colors.red,  
                size: 40,  
            ),  
        ],  
    ),  
);
```

```
        SizedBox(width: 8),  
        Text(  
          cityname,  
          overflow: TextOverflow.ellipsis,  
          style: TextStyle(  
            fontSize: 28,  
            fontWeight: FontWeight.bold,
```

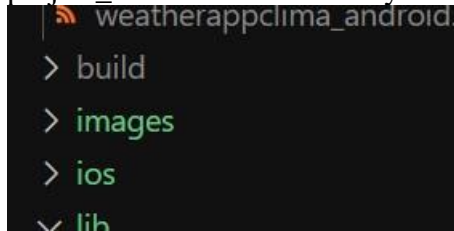
Hal ini diikuti lagi oleh widget SizedBox

```
SizedBox(height: 20)
```

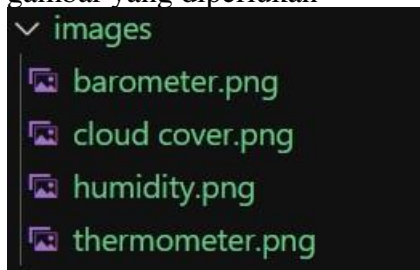


Berikutnya adalah bagian data cuaca yang ditampilkan sebagai kartu. Di sini saya menggunakan Container untuk membuat kartu kustom. Untuk data, saya menggunakan widget Gambar dan Teks.

Perhatikan bahwa gambar yang digunakan di sini berasal dari berkas proyek itu sendiri. Jadi untuk melakukannya, kita harus mengonfigurasi gambar tersebut. Untuk ini, pertama-tama buat direktori baru di folder proyek di bawah `project_name>New>Directory`



Beri nama direktori baru sebagai image. Setelah direktori dibuat, tambahkan gambar yang diperlukan



Setelah itu, buka file `pubspec.yaml` Anda. Gulir ke bawah ke bagian aset tempat gambar dimasukkan. Hapus komentar pada baris dan pastikan spasi sudah benar seperti yang ditunjukkan di bawah ini. Kode di bawah ini menyertakan semua file di bawah direktori gambar.

```
60   uses-material-design: true
61   assets:
62     - images/
```

Lalu jalankan perintah `pub get` di sudut kanan atas.

```
PS C:\laragon\www\Mobile_2024\weatherappclima> flutter pub get
Resolving dependencies...
Downloading packages...
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.18.0 (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
  flutter_lints 4.0.0 (5.0.0 available)
  http_parser 4.0.2 (4.1.1 available)
```

Setelah selesai, gambar akan dikonfigurasi di aplikasi. Sekarang di `homescreen.dart` kita akan menambahkan kode untuk membuat tampilan kartu.

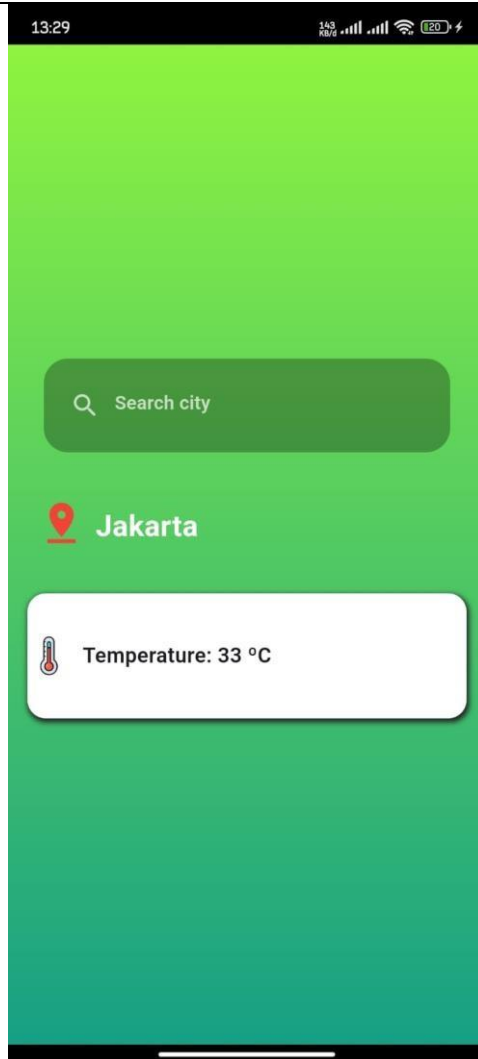
```
Container(
  width: double.infinity,
  height: MediaQuery.of(context).size.height
* 0.12,
```

```

margin: EdgeInsets.symmetric(vertical:
10),
decoration: BoxDecoration(
borderRadius: BorderRadius.all(
Radius.circular(15),
),
color: Colors.white,
boxShadow: [
BoxShadow(
color: Colors.grey.shade900,
offset: Offset(1, 2),
blurRadius: 3,
spreadRadius: 1,
),
],
),
child:
Row(
children: [
Image(
image:
AssetImage('images/thermometer.png'),
width:
MediaQuery.of(context).size.width * 0.09,
),
SizedBox(width: 10),
Text(
'Temperature: ${temp.toInt()}
°C',
style: TextStyle(
fontSize: 20,
fontWeight:
FontWeight.w600,

```

Setelah kode di atas ditambahkan, hentikan dan mulai aplikasi secara otomatis agar perubahan konfigurasi dapat disertakan. Saat aplikasi dimuat, kita akan mendapatkan tampilan kartu seperti yang ditunjukkan di bawah ini



Sekali lagi 3 kartu ditambahkan dengan mengganti gambar dan teks untuk parameter cuaca masing-masing. Untuk tampilan 3 kartu berikutnya, gambar dibungkus dengan bantalan di semua sisi.

