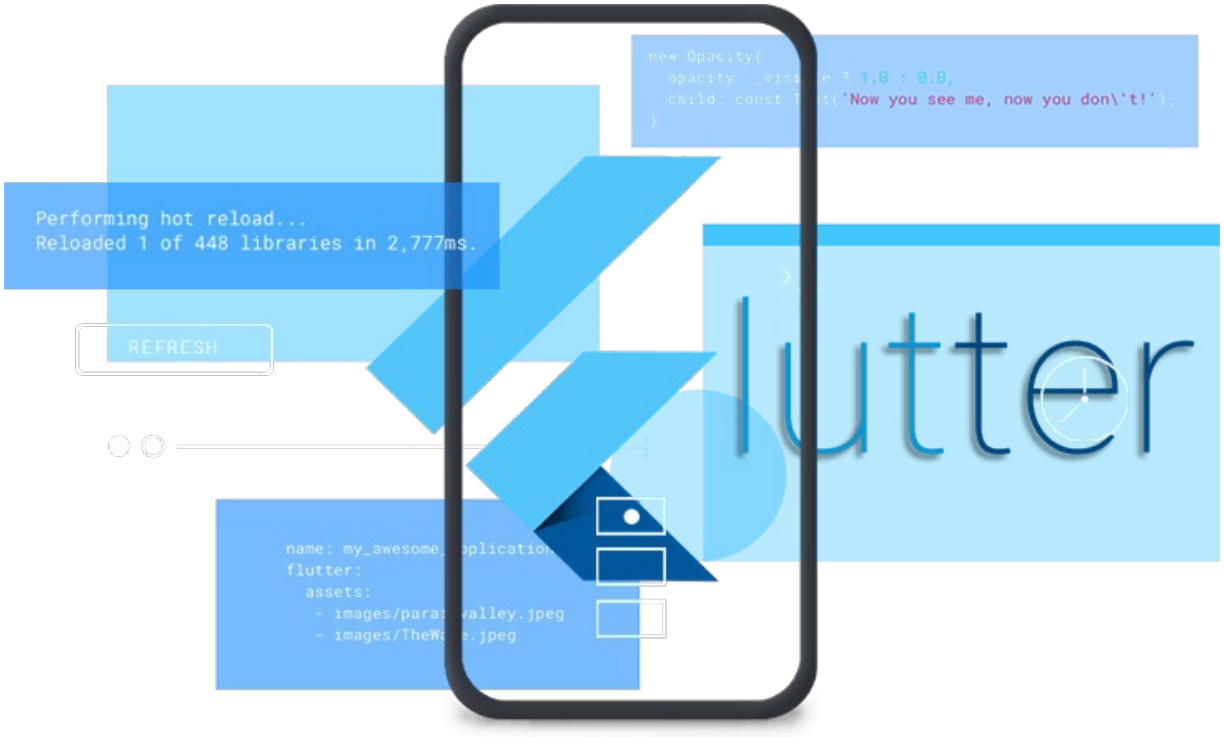
MODUL **MATAKULIAH PEMROGRAMAN MOBILE**



Oleh : Ahmad Subki, S.ST., M.Kom

FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI

UNIVERSITAS TEKNOLOGI MATARAM

2020

Kata Pengantar

Puji Syukur kehadirat Tuhan Yang Maha Esa karena atas limpahan rahmat-Nya sehingga kami dapat menyelesaikan modul Matakuliah Pemrograman Mobile untuk Program Studi (S1) Teknik Informatika. Modul ini juga dilengkapi dengan latihan soal untuk menguji pemahaman siswa terkait dengan materi yang terdapat pada modul. Dalam modul ini akan dibahas tentang cara membuat aplikasi mobile sederhana dengan menggunakan flutter.

Kami menyadari masih banyak kekurangan dalam penyusunan modul ini. Oleh karena itu, kami sangat mengharapkan kritik dan saran demi perbaikan dan kesempurnaan modul ini.

Kami mengucapkan terima kasih kepada berbagai pihak yang telah membantu proses penyelesain modul ini, terutama dosen-dosen pengampu mata kuliah Pemrograman Mobile, yang telah membantu dalam penyusuna modul ini.

Semoga modul ini dapat bermanfaat bagi kita semua, khususnya para peserta didik.

Mataram, Oktober 2020

ttd

Penyusun

Daftar Isi

1. Deskripsi Matakuliah.....iv
2. Modul 1. Arsitektur Perangkat Bergerak.....1
3. Modul 2. Pengenalan dan Installasi.....5
4. Modul 3. Membuat Aplikasi Android Pertama.....18
5. Modul 4. Tipe Data dan Variabel.....24
6. Modul 5. Mengenal Widget Flutter.....30
7. Modul 6. Statefull Widget dan Stateless Widget.....35
8. Modul 7. Membuat Aplikasi Sederhana dengan Flutter.....40
9. Modul 8. Membuat Desain Layout.....44
10. Modul 9. Membuat Drawer.....53
11. Modul 10. App Bar dan Tab Bar Widget.....64
12. Modul 11. Notification dan Alert.....72
13. Modul 12. Pengenalan JSON.....79
14. Modul 13. Implementasi JSON.....84
15. Modul 14. JSON Overview.....92
16. Daftar Pustaka.....103

DESKRIPSI MATAKULIAH

Judul Mata Kuliah : Pemrograman Mobile

Jumlah SKS : 2

A. Deskripsi Matakuliah

Pemrograman Mobile adalah matakuliah pemrograman yang ditujukan untuk pembuatan aplikasi diperangkat mobile, yang dapat dibuat salah satunya dengan menggunakan bahasa pemrograman Java, Kotlin, Swift dan C. Perkuliahan ini akan dilaksanakan sebanyak 16 kali pertemuan dengan ketentuan 1 kali pertemuan **Ujian Tengah Semester** dan 1 kali pertemuan **Ujian Akhir Semester.** Mahasiswa diwajibkan untuk berperan aktif dalam proses pembelajaran, baik dalam diskusi maupun dalam hal pengerjaan tugas. Pada matakuliah ini penilaian dilakukan dengan memperhatikan keaktifan mahasiswa dalam setiap kegiatan proses belejar. Semua proses belajar akan terekam secara otomatis dalam sistem, apabila dikemudian hari mahasiswa mendapatkan nilai yang tidak sesuai dengan apa yang diharapkan, maka dosen akan mengecek kembali Log aktifitas yang dilakukan oleh mahasiswa bersangkutan.

B. Capaian Pembelajaran Matakuliah

Pada matakuliah ini mahasiswa diharapkan memiliki kemampuan akhir berupa :

1. Mahasiswa mampu menjelaskan arsitektur perangkat mobile
2. Mahasiswa mampu menginstallasi pendukung aplikasi moblie pada komputer
3. Mahasiswa mampu mambuat aplikasi mobile pertama
4. Mahasiswa mampu menjelaskan method serta dapat menggunakan method dan widget
5. Mahasiswa mampu membuat Aplikasi Sederhana dengan Webview
6. Mahasiswa mampu membuat desain layout pada android yang user friendly menggunakan
7. Mahasiswa mampu menggunakan Drawer Widget & Routes dan JSON App Overview
8. Mampu membuat aplikasi sederhana dan meguplоad ke playstore

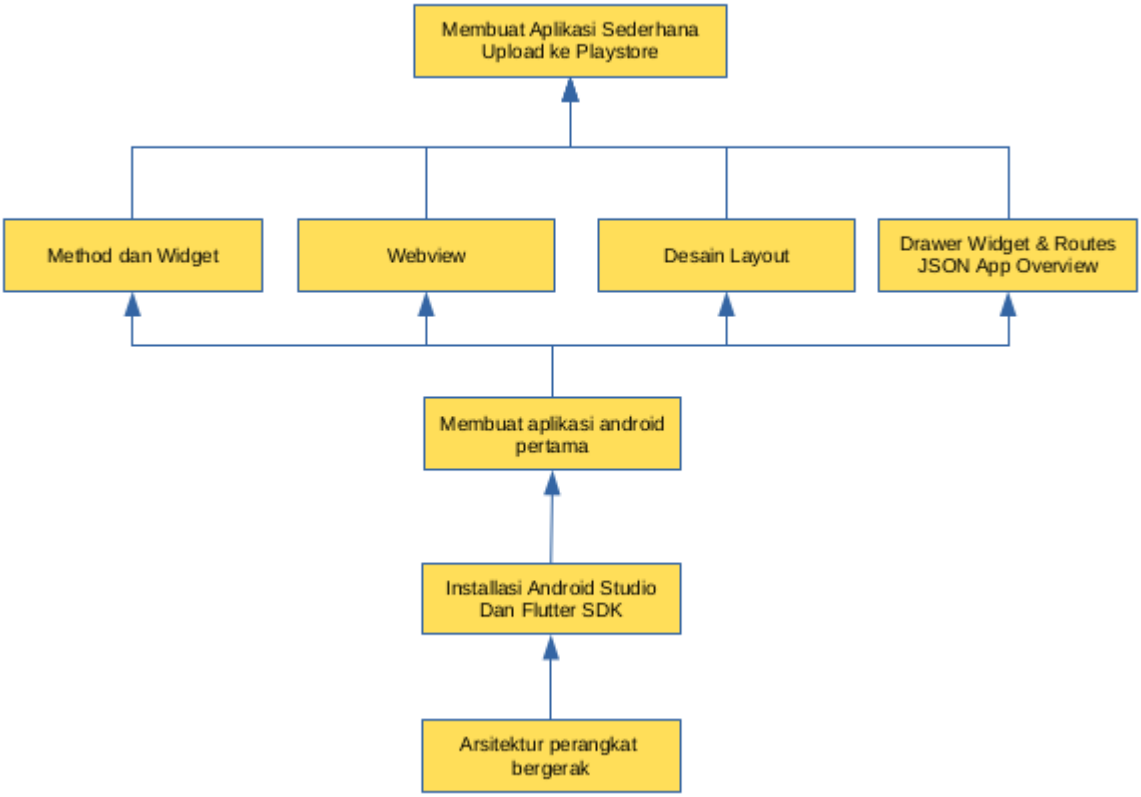
C. Bahan Kajian

Berikut adalah bahan kajian matakuliah pemrograman mobil :

1. Pengenalan android, sejarah android, keunggulan aplikasi android, arsitektur perangkat mobile, android, xml, gradle, persiapan & cara installing android dengan menggunakan sdk android studio
2. Install android studio, installasi android sdk, installasi pemrograman mobile

3. Pengenalan pemrograman mobile, Variabel , Tipe Data, Fungsi Input Dan Output
4. Operator, control flow, struktur data list, class, fungsi/method, stateful widget dan stateless widget pada flutter
5. Layout, App Bar & Tab Bar Widget, Custom Widget, Input, Selection
6. Drawer widget & routes, notification widgets, JSON app overview, create android apk
7.

D. Peta Kompetensi Matakuliah



E. Daftar Judul

1. **Bab 1. Arsitektur Perangkat Bergerak**
   - Pertemuan 1. Arsitektur Perangkat Bergerak
2. **Bab 2. Pengenalan dan Installasi**
   - Pertemuan 2. Pengenalan dan Installasi
3. **Bab 3. Membuat Aplikasi Android Pertama**
   - Pertemuan 3. Membuat Aplikasi Android Pertama
   - Pertemuan 4. Tipe Data dan Variabel

4. **Bab 4. Method dan Widget**
   - Pertemuan 5. Mengenal Widget Flutter
   - Pertemuan 6. Statefull Widget dan Stateless Widget
5. **Bab 5. Membuat Aplikasi Sederhana dengan Flutter**
   - Pertemuan 7. Membuat Aplikasi Sederhana dengan Webview
   - Pertemuan 8. Ujian Tengah Semester
6. **Bab 6. Layout**
   - Pertemuan 9. Membuat Desain Layout
   - Pertemuan 10. Membuat Drawer
   - Pertemuan 11. App Bar dan Tab Bar Widget
   - Pertemuan 12. Notification dan Alert
7. **Bab 7. JSON**
   - Pertemuan 13. Pengenalan JSON
   - Pertemuan 14. Implementasi JSON
   - Pertemuan 15. JSON Overview

F. Petunjuk Penggunaan

1. Sebelum membaca setiap modul, mahasiswa diharapkan membaca terlebih dahulu deskripsi mata kuliah, capaian pembelajaran, bahan kajian dari mata kuliah ini, serta memahami peta kompetensi sebagai petunjuk tahap-tahap belajar pemrograman mobile.
2. Pada bagian Pendahuluan di setiap modul diberikan deskripsi singkat untuk mengetahui secara garis isi dari modul tersebut, tujuan instruksional khusus yang harus dicapai mahasiswa setelah mempelajari modul dan peta kompetensi yang menggambarkan langkah-langkah pembelajaran untuk mencapai kompetensi yang diharapkan.
3. Pada bagian isi diberikan uraian materi yang berisi penjelasan yang harus dibaca yang dilanjutkan dengan contoh dan non contoh dalam berbagai sistem informasi berbasis web. Uraian tersebut dilanjutkan dengan source code program sebagai latihan yang langsung dikrjakan oleh mahasiswa pada komputer masing-masing dengan bimbingan dari dosen untuk memastikan langkah-langkah yang dikerjakan sudah benar.
4. Setelah program selesai dibuat, jalankan program tersebut di layar browser anda. Pastikan hasilnya sesuai dengan print screen di modul anda.
5. Pada akhir pertemuan, anda diminta mengerjakan soal tes untuk menguji pemahaman anda terhadap materi dalam modul. Harap selesaikan soal tersebut secara objektif sesuai waktu yang ditentukan oleh dosen.
6. Sebagai penutup, dosen akan memberikan umpan balik terhadap hasil tes atau terhadap kegiatan pembelajaran dalam pertemuan tersebut.

BAB 1
ARSITEKTUR PERANGKAT MOBILE

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang arsitektur perangkat bergerak, diharapkan mahasiswa mampu memahami dan mampu menjelaskan tentang aristektur perangkat bergerak

2 Capaian Pembelajaran

Mampu menjelaskan arsitektur perangkat bergerak (mobile programming)

3 Bahan Kajian

Pengenalan android, sejarah android, keunggulan aplikasi android, arsitektur perangkat mobile, android, xml, gradle, persiapan & cara installing android dengan menggunakan sdk android studio

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 1. Arsitektur Perangkat Mobile/Bergerak

Pada materi pertama ini kita akan membahas tentang arsitektur perangkat mobile, seperti yang kita ketahui bersama bahwa perangkat mobile kini menjadi salah satu kebutuhan yang wajib dimiliki oleh semua orang. Sehingga sampai saat ini perkembangannya kian semakin meningkat. Tak jarang kita melihat seseorang memiliki perangkat mobile lebih dari 1 bahkan mungkin lebih dari 2.

Untuk matakuliah kita di Pemrograman Mobile ini, kita akan membahas tentang :

1. Pengenalan Android
2. Sejarah Android
3. Keunggulan Aplikasi Android
4. Arsitektur Perangkat Mobile

1. Pengenalan dan Sejarah Android

Android sendiri adalah sebuah sistem operasi yang berbasis Linux dan dirancang khusus untuk perangkat layar sentuh (touch screen). Android kini menjadi sistem operasi yang paling banyak digunakan oleh berbagai perangkat/device. Android awalanya dikembangkan oleh Android.Inc, dan didukung dengan pembiayaan oleh perusahaan rakasasa yaitu Google. Melihat peluang yang sangat menjanjikan pada Sistem Operasi baru ini pada tahun 2005 kemudian Android dibeli secara permanen oleh Google. Google kemudian merlisis Sistem Operasi Andorid pertama kali pada tahun 2007 bersamaan dengan berdirinya *Open Handset Alliance.*

Android dikembangkan dengan konsep *Open Source* artinya setiap orang bebas mengembangkan kode yang ada di dalamnya. Namun tetap Google merilis kodenya dibawah lisensi Apache. Tentunya dengan konsep *Open Source* ini membuat Sistem Operasi ini sangat diminati oleh banyak pengembang perangkat lunak, operator nirkabel dan pengembang aplikasi lainnya.

2. Keunggulan Andorid

Sebagai sebuah sistem operasi yang sangat populer tentunya Android memiliki keunggulan dibandingkan dengan pesaing-pesaingnya semisal iOS. Berikut adalah keunggulan Android yang bisa sedikit kita bahas bersama.

a. Open Source

Seperti yang telah kita bahas di atas, Android merupakan sistem operasi yang bersifat terbuka, dan bisa dikembangkan tanpa takut melanggar Hak Cipta.

b. Mendukung Touch Screen

Tentunya Android diciptkan semata mata untuk device yang mendukung layar sentuh agar mendapatkan hasil maksimal dari kinerja Android.

c. Didukung banyak Aplikasi

Bila kita lihat Google Play terdapat banyak sekali aplikasi yang bisa digunakan untuk mendukung kinerja dari Android sendiri.

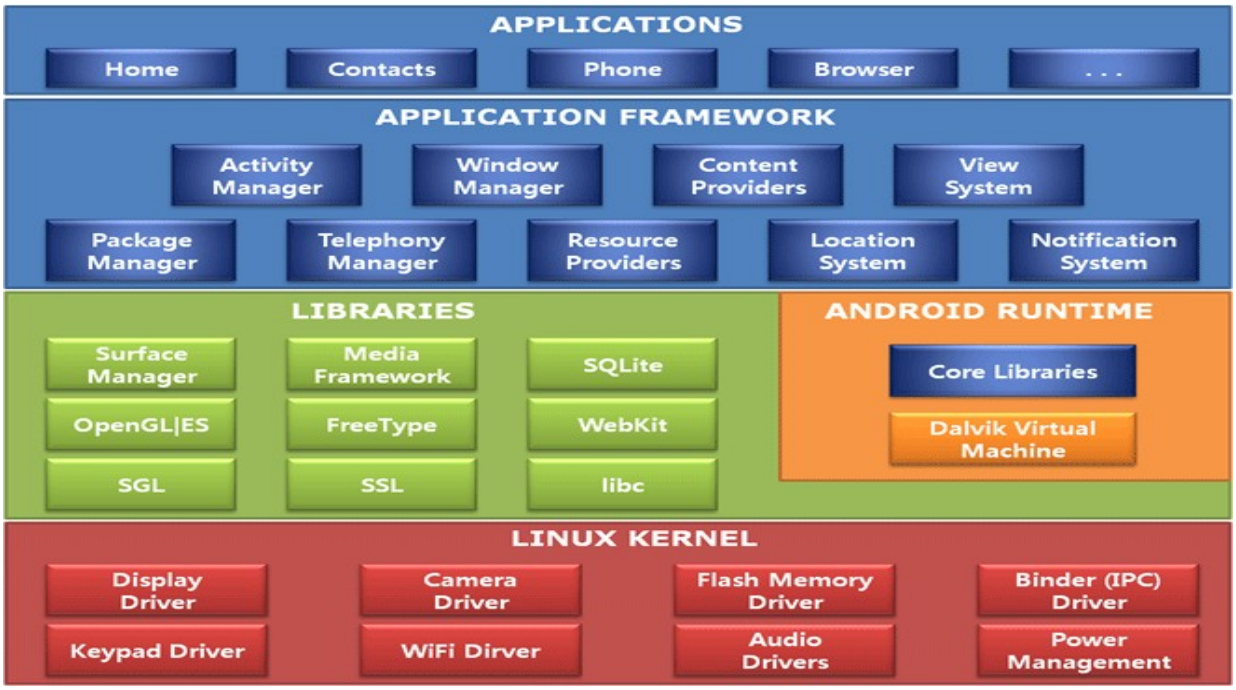
d. Dikembangkan oleh Perusahaan Besar

Android di backup oleh perusahaan berkelas dunia, semisal Google, Samsung, HTC, OPPO dan lainnya.

e. Selalu di Update

Salah satu yang menjadi nyawa sebuah sistem operasi yaitu dukungan Update dari Pengembang. Dan inilah yang membuat Android itu terus berkembang hingga sekarang.

3. Arsitektur Perangkat Mobile



Gambar diatas merupakan arsitektur dari Android yang mana terdiri dari 5 elemen yaitu :

a. Aplikkasi

Aplikasi ini merupakan lapisan dimana kita biasanya langsung melihat tampilannya seperti Kalender, SMS, Telephone dll.

b. Aplication Framework

Pada lapisan ini biasanya pengembang memanfaatkannya untuk dapat mengakses secara penuh ke sistem operasi android. Hal ini biasanya digunakan pengembang untuk dapat mengatur alaram, menambahkan notifikasi dll.

c. Library

Uniknya dalam library android sendiri menggunakan bahasa C/C++ yang dapat mendukung semua komppnen pada sistem operasi Android

d. Android Runtime

Satu set libraries inti yang menyediakan sebagai besar fungsi Yang tersedia di libraries inti dari bahasa pemrograman java. Setiap aplikasi akan berjalan sebagai proses sendiri pada *DalvikVirtual Machine.*

e. Linux Kernel

Pada lapisan ini mengatur tentang keamanan, manajemen memory, network, proses, driver, manajemen power dan sebagainya. Android saat ini telah merilisi versi terbaru yaitu Android 11.

C. Evaluasi

Setelah menyelesaikan modul ini, mahasiswa diharuskan mengerjakan untuk mengerjakan Tugas 1.

Latihan	Pelajari dan cobalah bahasa pemrograman Flutter lebih lanjut dari laman web flutter.dev
Tugas 1	Mahasiswa membuat makalah tentang sejarah, keunggulan dan arsitektur Android melalui internet, dan menyusunnya dalam bentuk makalah.
Refrensi	https://flutter.dev https://medium.com/community/flutter

MODUL 2
PENGENALAN DAN INSTALLASI

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang langkah-langkah untuk melakukan installasi pendukung pemrograman mobile pada komputer dengan software android studio dan framework flutter

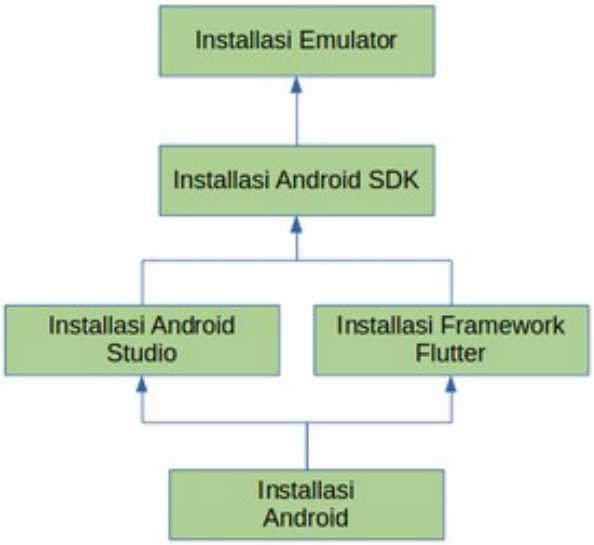
2 Capaian Pembelajaran

Mahasiswa mampu menginstallasi pendukung aplikasi moblie pada komputer

3 Bahan Kajian

Install android studio, installasi android sdk, installasi pemrograman mobile, installasi flutter SDK

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 2. Pengenalan dan Installasi

Materi kali ini kita akan membahas bagaimana melakukan installasi Android Studio untuk pertamakali, sebalum kita belajar membuat aplikasi Andorid Pertama kita. Adapun yang perlu kita persiapkan yaitu :

1. Spesifikasi Komputer

Dikutip dari situs developer.android.com, spesifikasi minimal laptop atau komputer yang bisa menjalankan Android Studio yaitu sebagai berikut :

OS Windows

- Menggunakan OS Microsoft® Windows® 7/8/10 (32- or 64-bit)
- Ram minimal 3 GB, yang direkomendasikan 8 GB RAM; tambahan 1 GB untuk menjalankan Emulator Android
- Ruang disk yang tersedia minimal 2 GB,
- Namun yang direkomendasikan yaitu 4 GB (500 MB untuk IDE + 1.5 GB untuk Android SDK and emulator system image)
- Resolusi minimal yang digunakan adalah 1280 x 800

Mac OS

- Mac® OS X® 10.10 (Yosemite) atau lebih tinggi sampai 10.13 (macOS High Sierra)
- Ram minimal 3 GB, yang direkomendasikan 8 GB RAM; tambahan 1 GB untuk menjalankan Emulator Android
- Ruang disk yang tersedia minimal 2 GB,
- Namun yang direkomendasikan yaitu 4 GB (500 MB untuk IDE + 1.5 GB untuk Android SDK and emulator system image)
- Resolusi minimal yang digunakan adalah 1280 x 800

OS Linux

- GNOME or KDE desktop
- Tested on Ubuntu® 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)
- Distribusi 64-bit yang mampu menjalankan aplikasi 32-bit
- GNU C Library (glibc) 2.19 atau yang lebih baru
- Ram minimal 3 GB, yang direkomendasikan 8 GB RAM; tambahan 1 GB untuk menjalankan Emulator Android
- Ruang disk yang tersedia minimal 2 GB,
- Namun yang direkomendasikan yaitu 4 GB (500 MB untuk IDE + 1.5 GB untuk Android SDK and emulator system image)
- Resolusi minimal yang digunakan adalah 1280 x 800

Chrome OS

- 8 GB RAM atau lebih sangat direkomendasikan
- 4 GB minimal
- Minimal resolusi layar 1280 x 800
- Menggunakan prosesor Intel i5 atau yang lebih tinggi sangat direkomendasikan

Rekomendasi device :

- Acer: Chromebook 13/Spin 13, Chromebox CXI3
- Lenovo: Yoga C630 Chromebook
- HP: Chromebook x360 14, Chromebox G2
- Dell: Inspiron Chromebook 14
- ASUS: Chromebox 3
- ViewSonic: NMP660 Chromebox
- CTL: Chromebox CBx1

Note : Disarankan menggunakan OS 64 bit

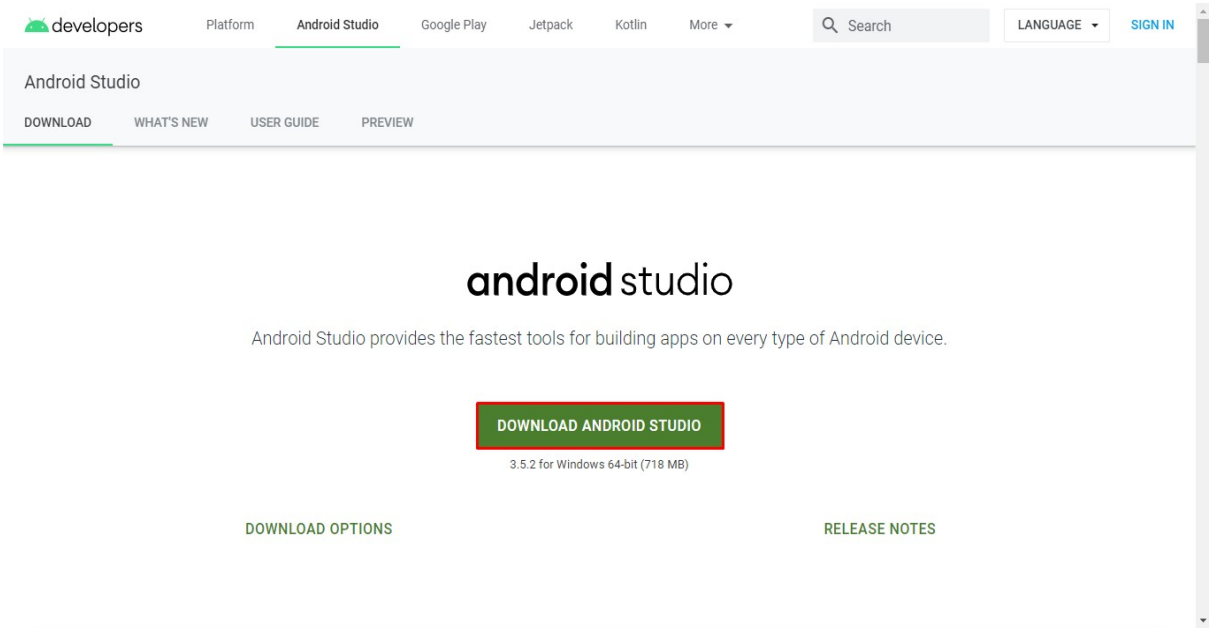
Apabila kita perhatikan, spesifikasi minimal untuk komputer apabila ingin bergelut di dunia mobile programing memang cukup besar.

2. Installasi Android Studio

Setelah kita memenuhi kebutuhan spesifikasi minimal komputer yang dibutuhkan untuk Android Studio, selanjutnya kita harus melakukan installasi Android Studio.

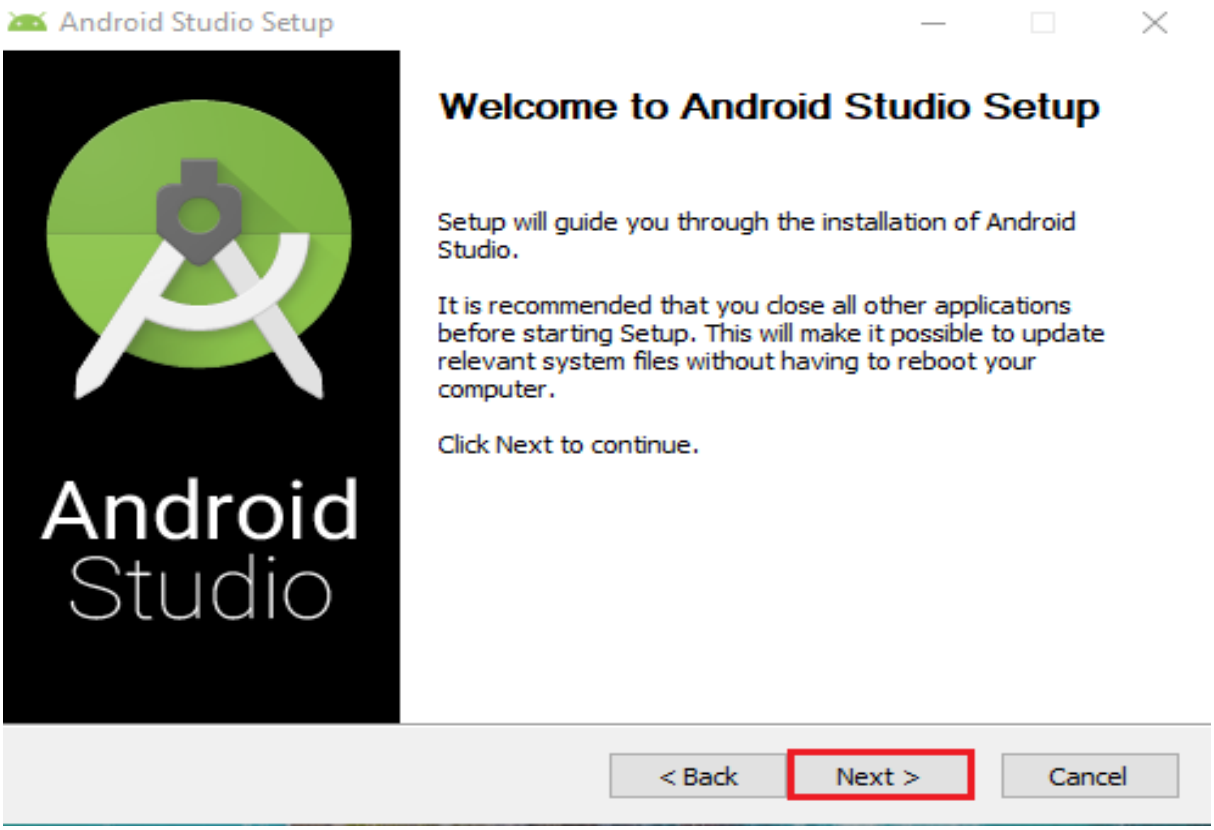
1. Download Android Studio

Berikut adalah link untuk mendownload aplikasi Android Studio https://developer.android.com/studio .



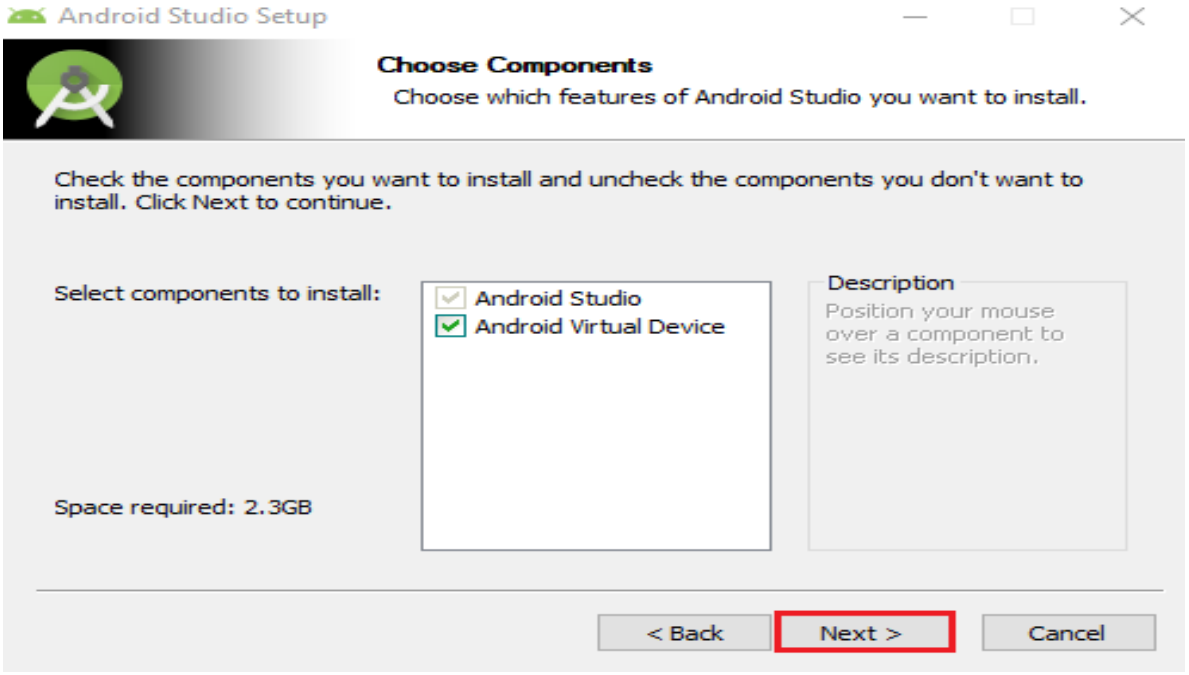
2. Install Android Studio

Cara menginstall Android Studio sama saja dengan melakukan installasi pada aplikasi lainnya. Ketika pertama kali dijalankan kita hanya perlu menekan tombol **Next.**



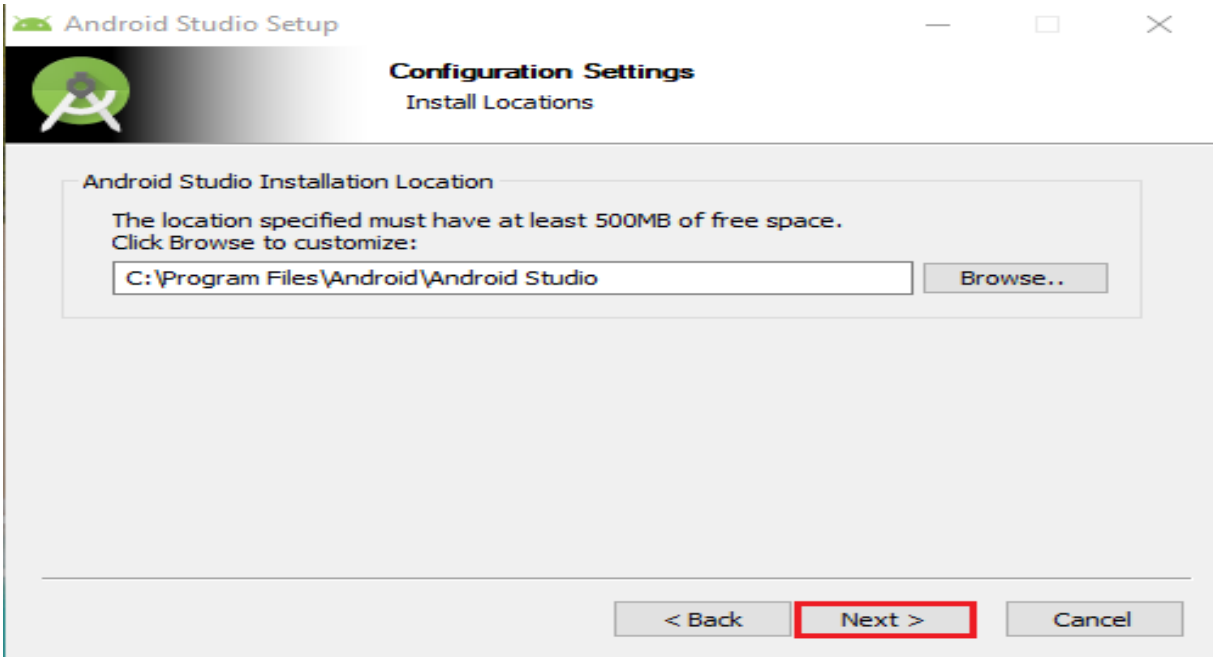
3. Memilih Komponen

Komponen yang kita install sesuikan dengan kebutuhan masing-masing, bila kita juga membutuhkan **Android Virtual Device** maka centang, namun AVD ini cukup besar ukurannya.



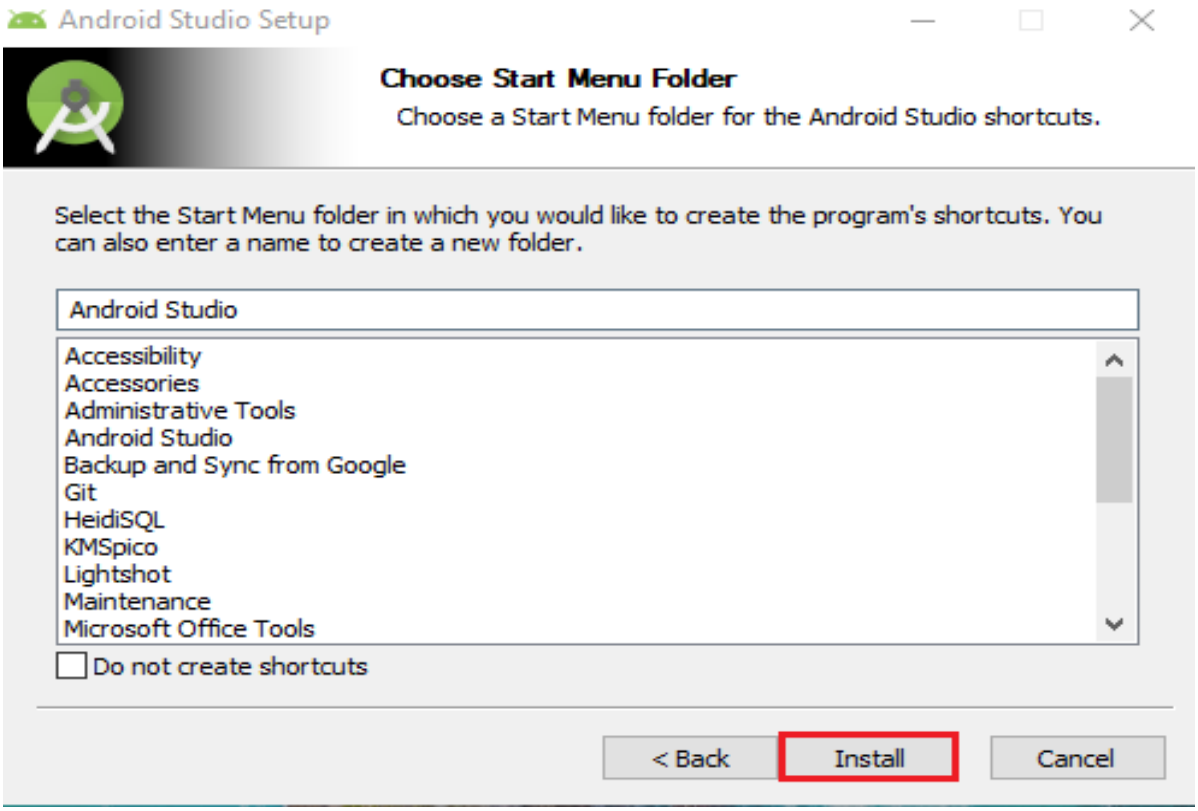
4. Lokasi Installasi

Untuk lokasi installasi bisa dimana saja, tapi biasanya ada di Program File kalo menggunakan Windows.

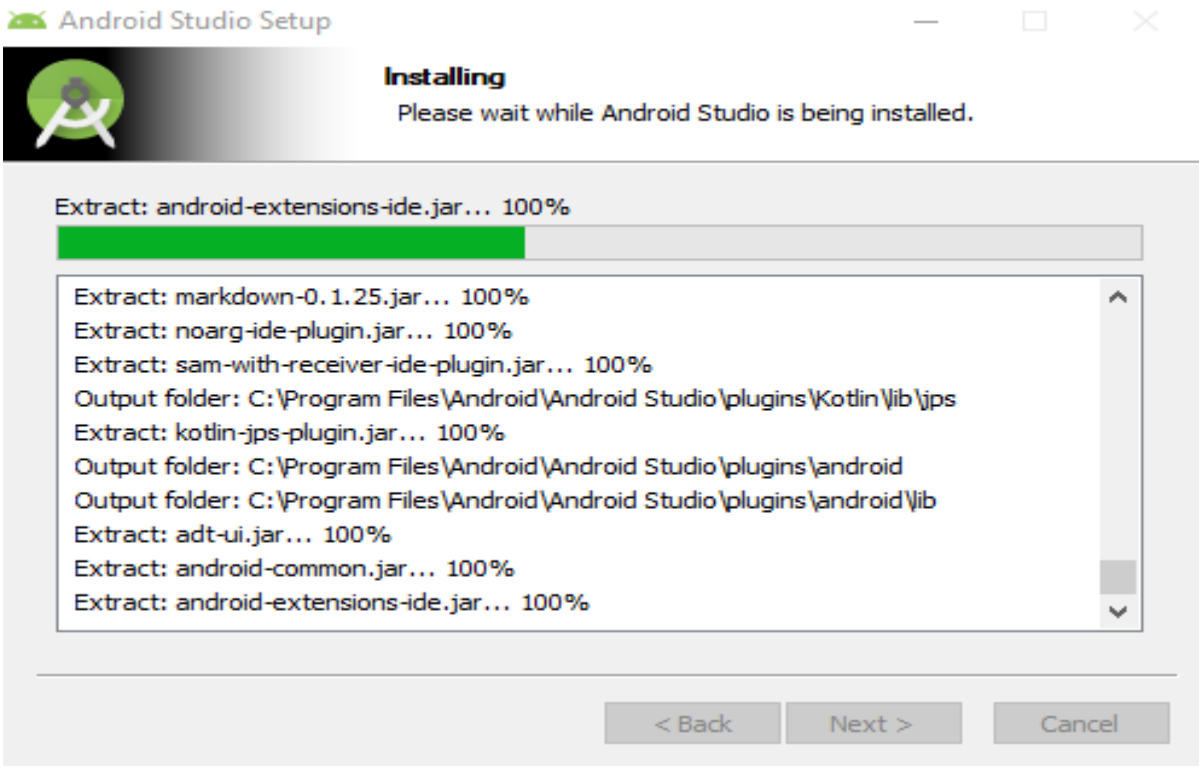


5. Proses Installasi

Setelah semua siap, tidak perlu menunggu lama tinggal kita langsung eksekusi, klik Install maka proses install pun akan berlangsung.



Tunggu hingga proses installasi selesai, dengan cara membuat secangkir kopi biar tidak bosen. karena mungkin prosesnya berlangsung sekitar beberapa menit.



Setelah kita menyelesaikan installasi, Android Studio tidak bisa langsung digunakan, namun perlu dilakukan beberapa pengaturan dan penambahan komponen lainnya.

Installasi Android SDK Android Studio

Android SDK ini sangat penting bahkan bisa dikatakan wajib bagi pengembang aplikasi Android. Namun pastikan ketika melakukan installasi komputer kita terhubung dengan Internet. Karena SDK akan secara otomatsi terdownload.

Installasi Selesai

Setelah melakukan installasi SDK maka Android Studio sudah bisa digunakan untuk membuat aplikasi Android.

3. Installasi Flutter SDK

Sebelum kita lanjut untuk melakukan installasi Flutter, sebelumnya kita terlebih dahulu setidaknya mengetahui apa itu flutter ?

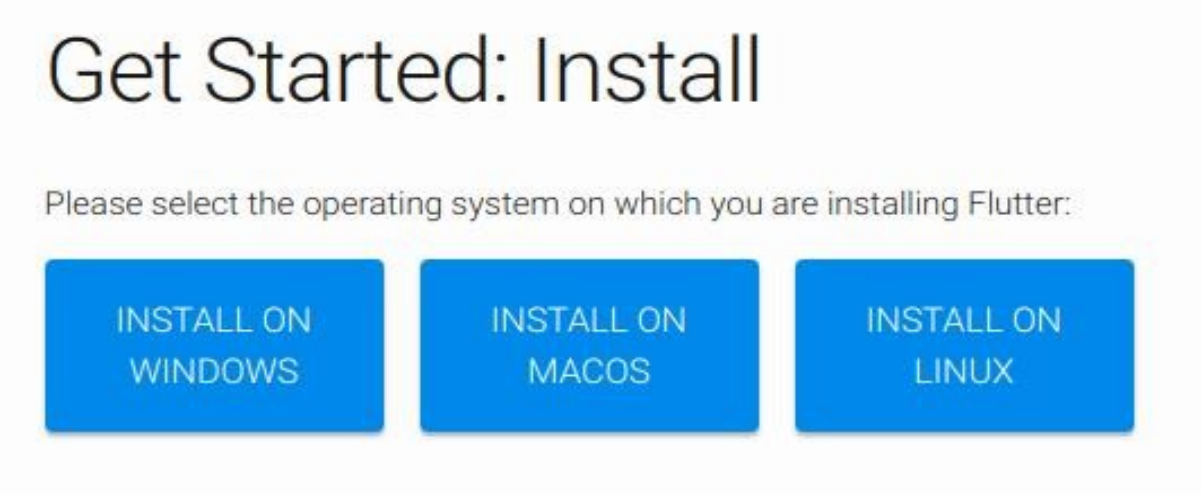
Flutter

Flutter merupakan sebuah framework untuk mengembangkan aplikasi Android, iOS dan Web dengan cukup sekali menulis kode pemrograman. Flutter sendiri

dikembangkan oleh Google pada tahun 2016. Saat ini Flutter telah banyak digunakan dan memiliki banyak komunitas. Menariknya Flutter ini mirip sekali dengan bahasa pemrograman Java dan C++.

1. Download Flutter SDK

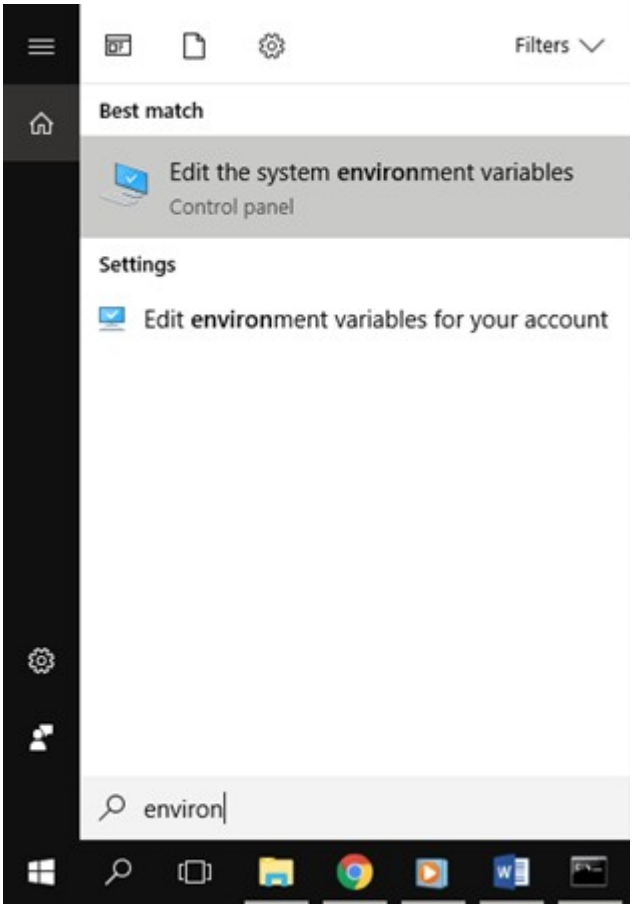
Untuk melakukan installasi Flutter, yang harus pertamakali kita lakukan yaitu mendownload SDK Flutter pada link berikut https://flutter.dev/docs/get-started/install



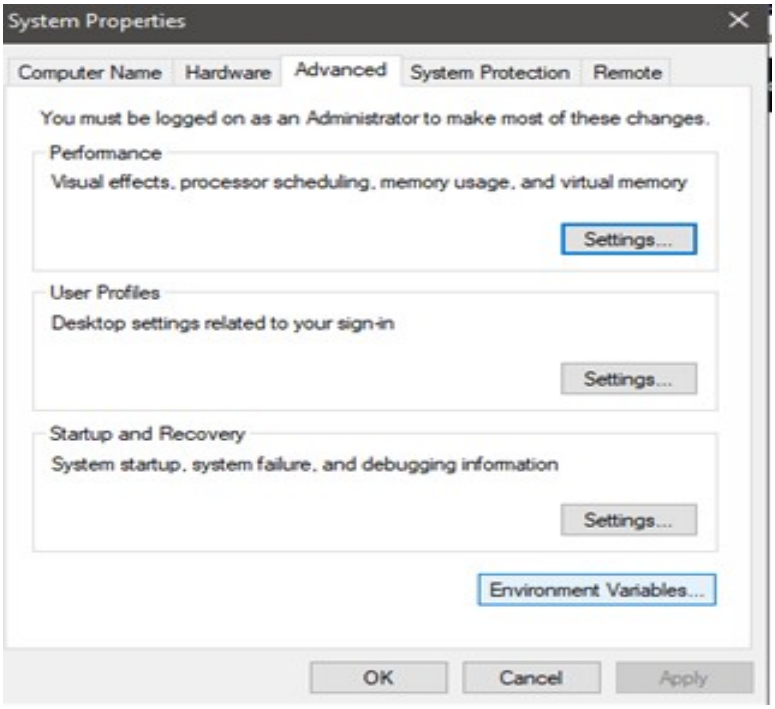
2. Ekstak Flutter SDK

File Flutter SDK dapat di ekstrak dimana saja, sesuai dengan keinginan kita. Namun jangan lupa setalah melakukan ekstrak Flutter SDK selanjutnya kita mengatur lokasi **PATH** sebagai berikut:

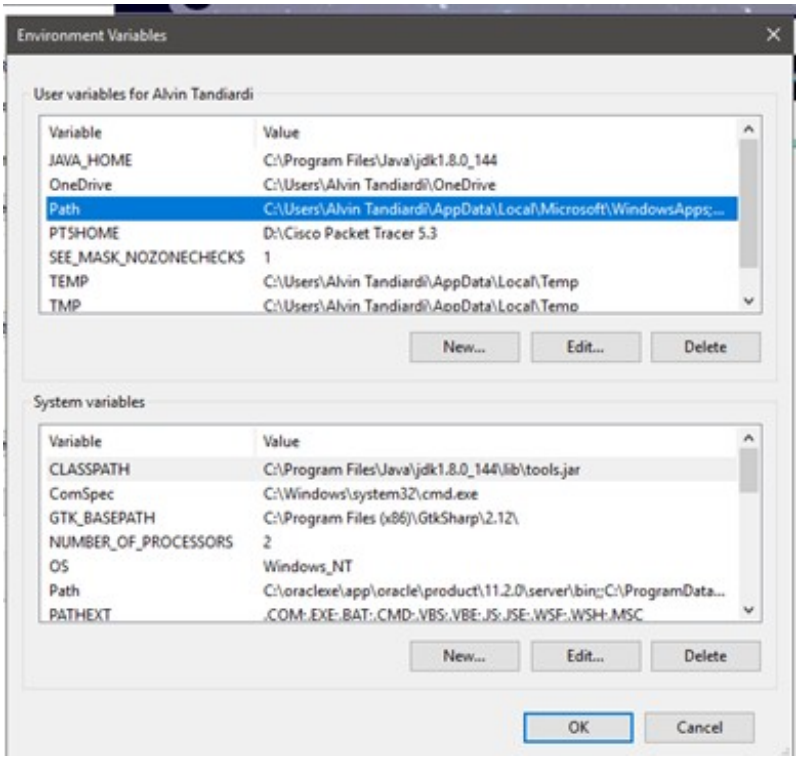
Buka tombol window pada keyboard -> Search Environtment -> Klik **Edit the system environment variables.**



Kemudian akan muncul menu sebagai berikut ini.



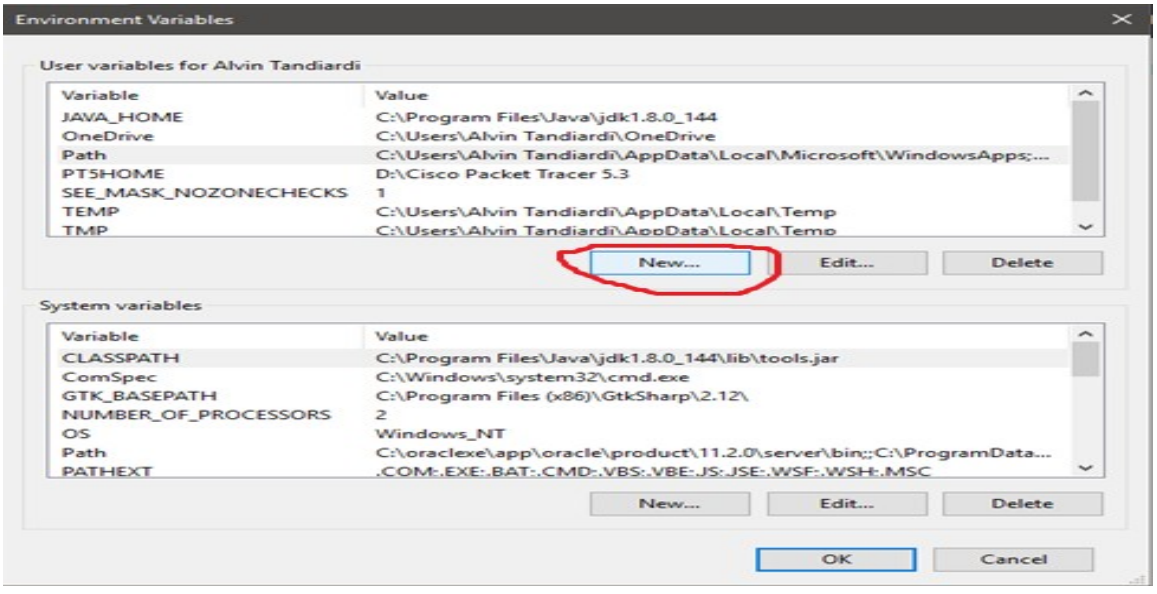
Selanjutnya Klik pada tombol **Environtment Variabel**



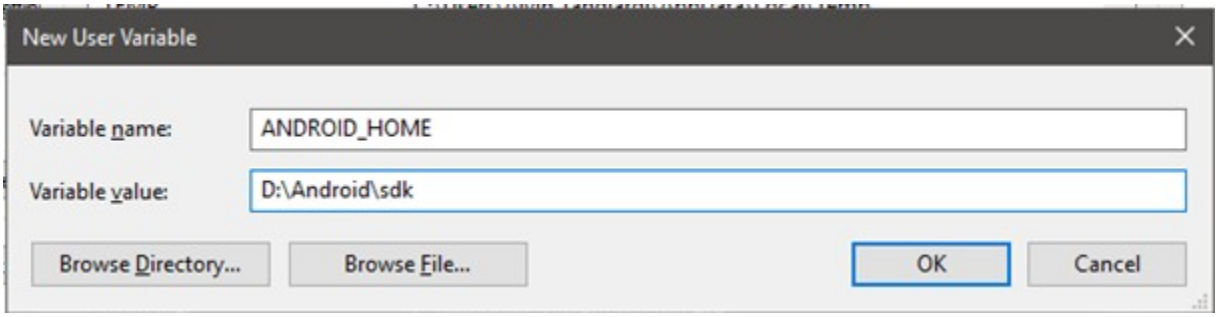
Pada Path klik tombol **Edit** dan ***Copy – Paste*** tuliskan lokasi dimana kamu menyimpan folder Flutter SDK tadi (flutter/bin) Tekan ok untuk menyimpan.

Disini kita menyimpan di **D:\flutter\bin** Ini berguna agar kamu dapat menjalankan fungsi flutter seperti flutter doctor langsung melalui cmd

Selanjutnya kita tambahkan **New Environtment Variabel**



Isi Variable name dengan **ANDROID_HOME** dan Variable value dengan lokasi sdk Android kamu, contoh direktori kita -> **D:\Android\sdk**

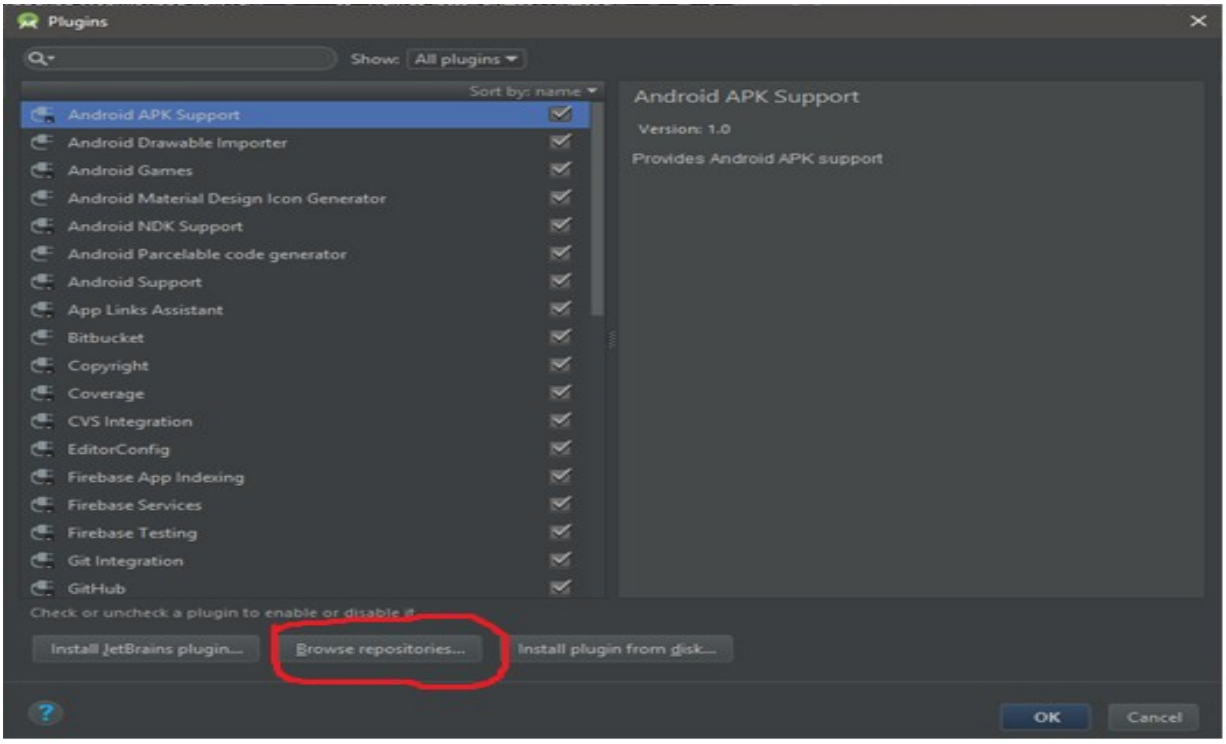


3. Setting Android Studio

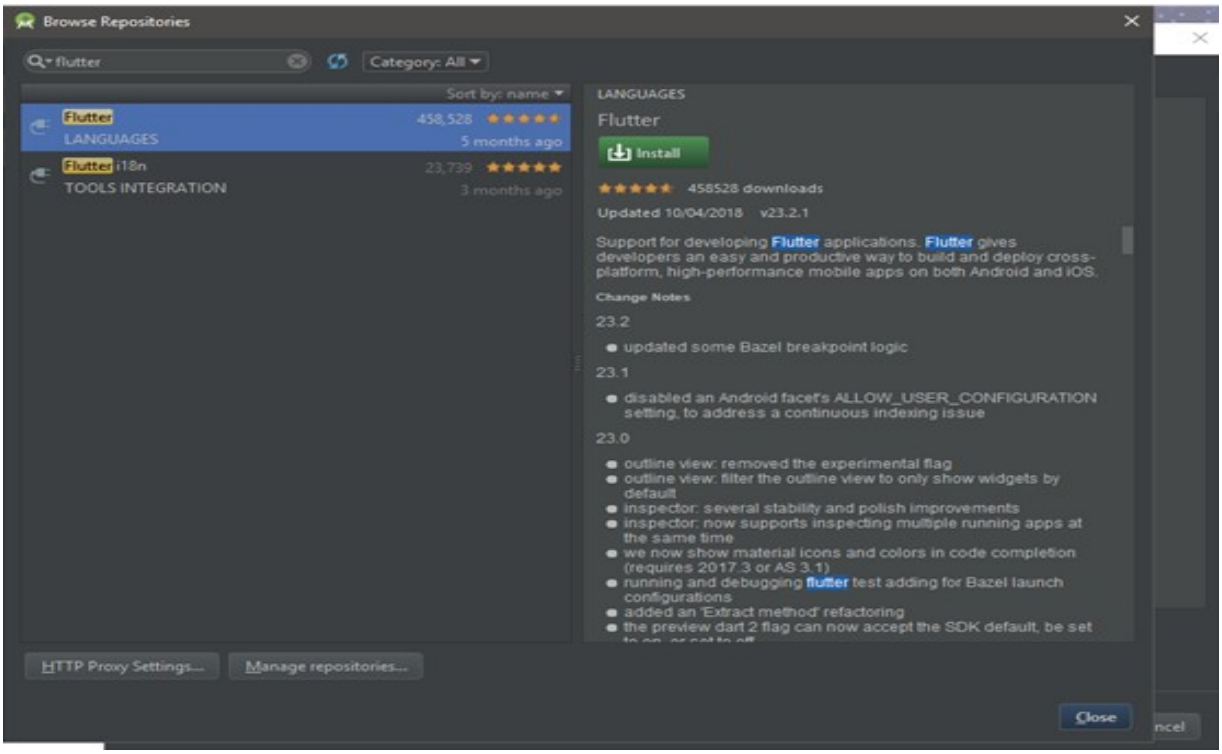
Mari buka Android Studio yang telah kita Install Sebelumnya, selanjutnya Klik **Configure** pada bagian kanan bawah.



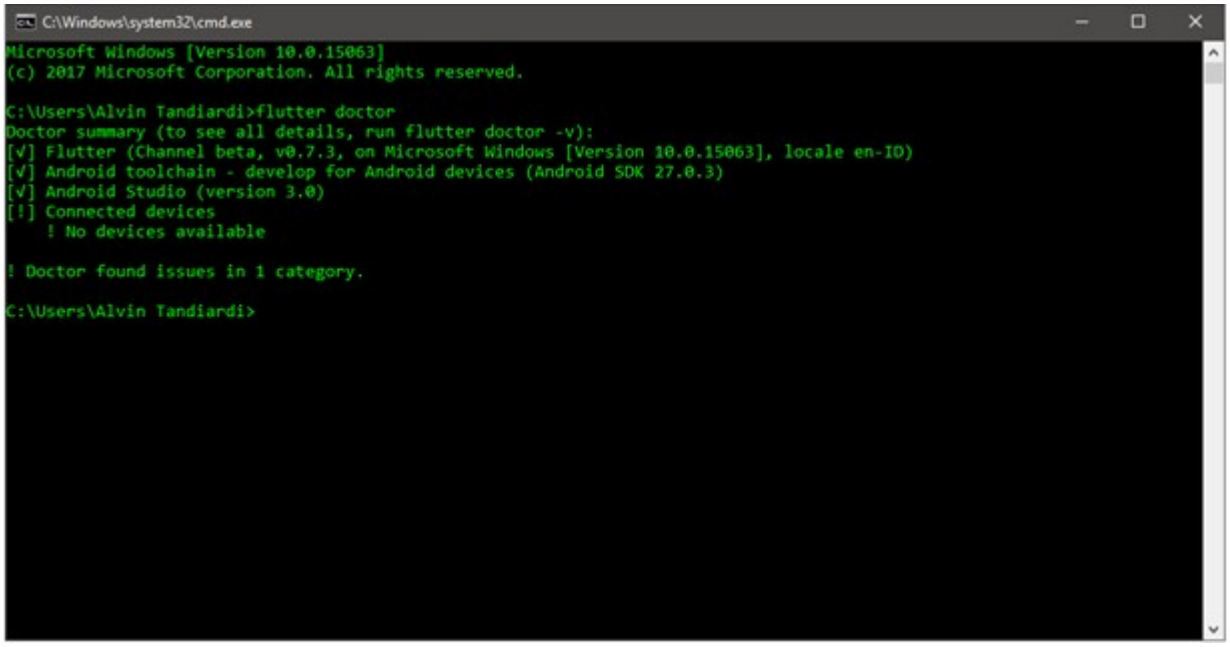
Selanjutnya install Plugin Flutter dengan cara mengetikkan kata "Flutter" pada menu Search yang ada di atas.



Kemudian klik Install.



Jika telah selesai. Mari kita coba dengan menjalankan Command Promt - Flutter Doctor.



Bila muncul seperti gambar di atas, maka Installasi telah selesai.

C. Evaluasi

Setelah mahasiswa selesai mempelajari modul ini, mahasiswa diminta mencoba secara mandiri bagaimana cara installasi Android Studio dan Flutter SDK.

Latihan	Install Android Studio dan Flutter SDK pada perangkat komputer anda masing-masing di rumah Silakan mempelajari bahasa pemrograman flutter lebih lanjut dari laman web flutter.dev
Tugas	-
Refrensi	https://flutter.dev https://medium.com/community/flutter

BAB 3
MEMBUAT APLIKASI PERTAMA DENGAN FLUTTER

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang dasar-dasar dari pemrograman android dengan flutter, dan mahasiswa akan mencoba untuk membuat aplikasi android pertamanya.

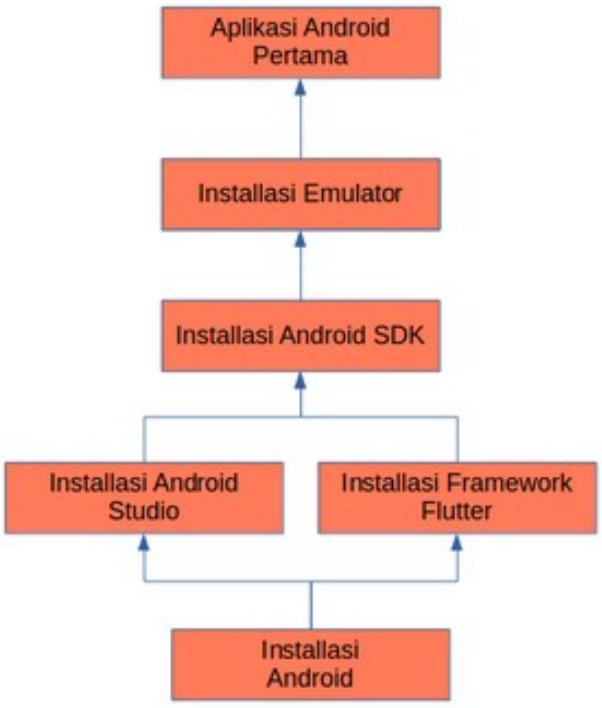
2 Capaian Pembelajaran

Mahasiswa mampu menginstallasi pendukung aplikasi moblie pada komputer

3 Bahan Kajian

Pengenalan pemrograman mobile, variabel, tipe data, fungsi input dan output

4 Peta Kompetensi Modul



B. Kegiatan Belajar

	Pertemuan 3. Membuat Aplikasi Android Pertama

Pada materi kali ini, bersama sama kita akan mempelajari bagaimana membuat sebuah aplikasi sederhana dengan Flutter. Pada pertemuan sebalumnya pernah kita singgung tentang apa itu Flutter.

Flutter sendiri adalah sebuah Framework yang memudahkan seorang developer mengembangkan aplikasi berbasis mobile. Uniknya Flutter ini bisa digunakan untuk mengembangkan aplikasi Androri, iOS dan bahkan Web sekaligus dengan sekali coding. Sangat mengesankan.

Flutter dikembangkan oleh perusahaan raksasa yaitu Google. Tidak tanggung tanggung saat ini Flutter menjadi salah satu Framework wajib yang harus dikuasai oleh programmer mobile aplication.

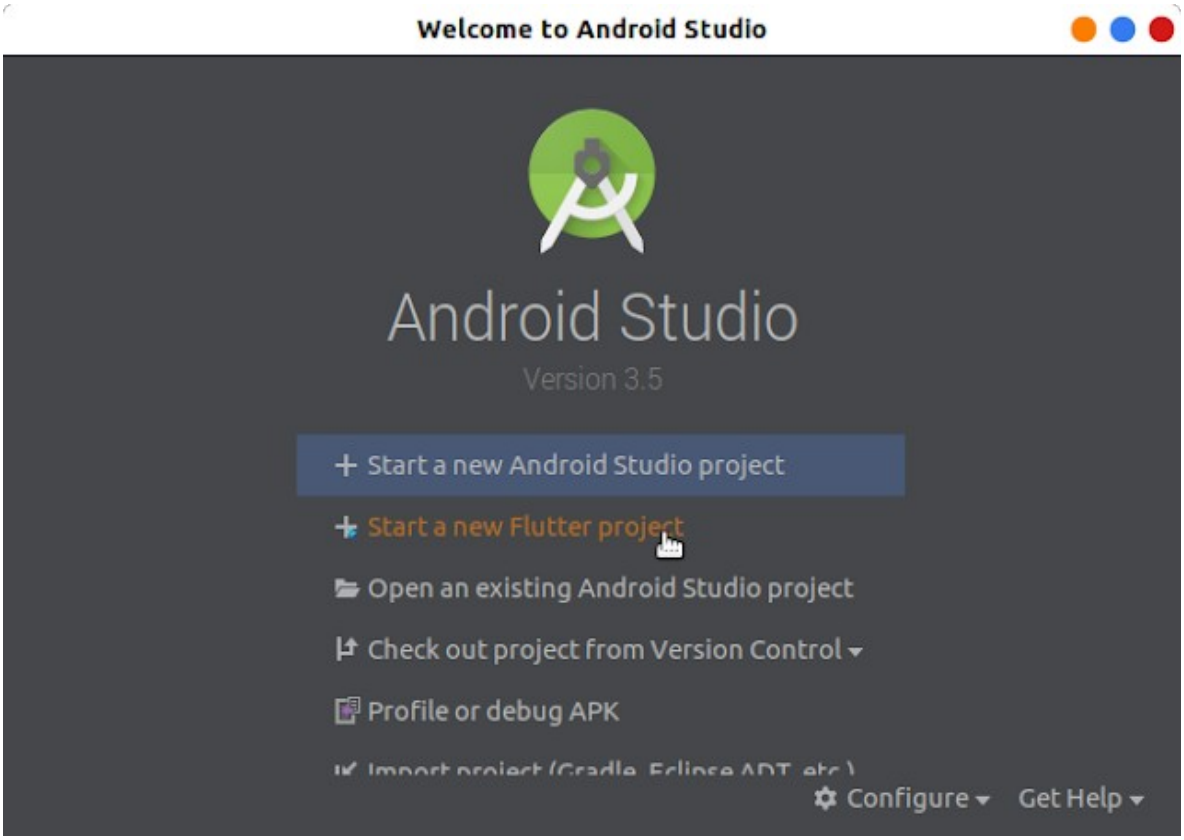
Aplikasi Android Pertama Menggunakan Flutter

Saat pertama kali kita membuka project baru pada Flutter, kita akan langsung disuguhkan sebuah kode pemrograman. Kode ini merupakan contoh aplikasi Flutter sederhana yang nantinya bisa kita kembangkan menjadi berbagai macam bentuk aplikasi. Langkah-langkah memulai pemrograman mobile dengan Flutter.

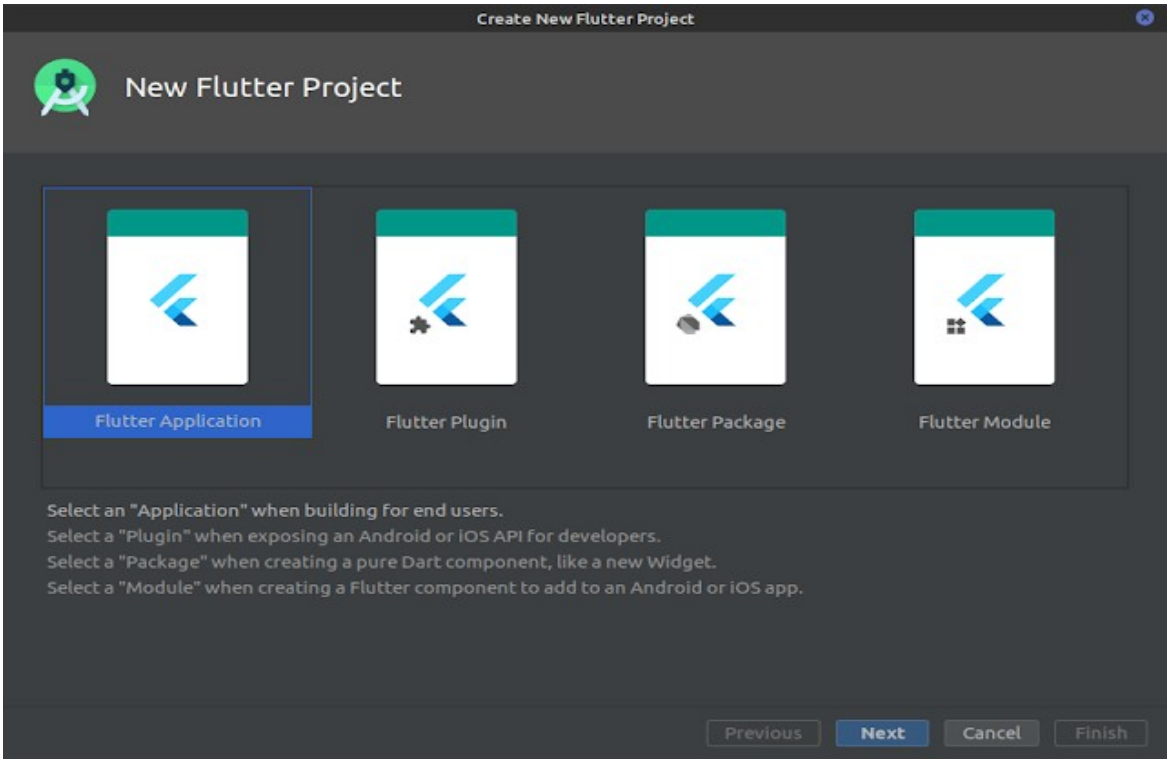
Buka Android Studio

Salah satu tools wajib yang harus dimiliki untuk membuat aplikasi Android yang pastinya adalah Android Studio. Begitu juga dengan Flutter ini. Kita bisa menggunakan Android studio untuk menjalankannya.

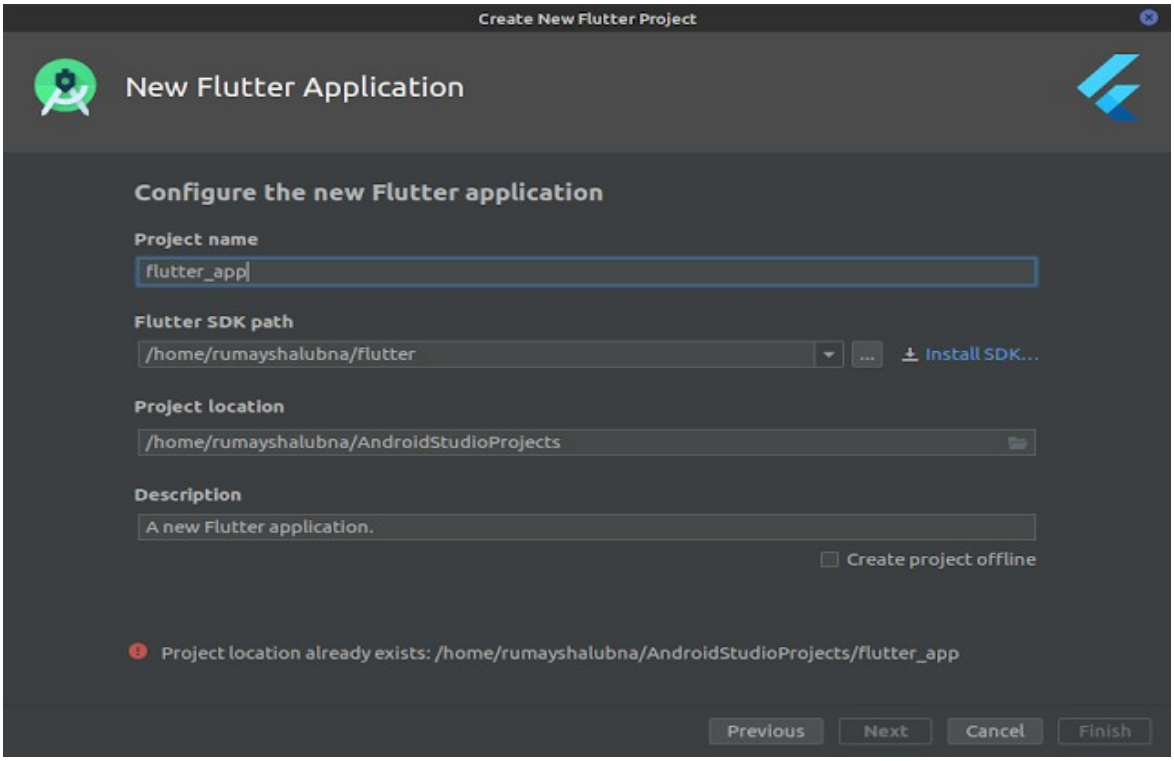
1. **Memulai Project Baru (Klik Start a new Flutter Project)**



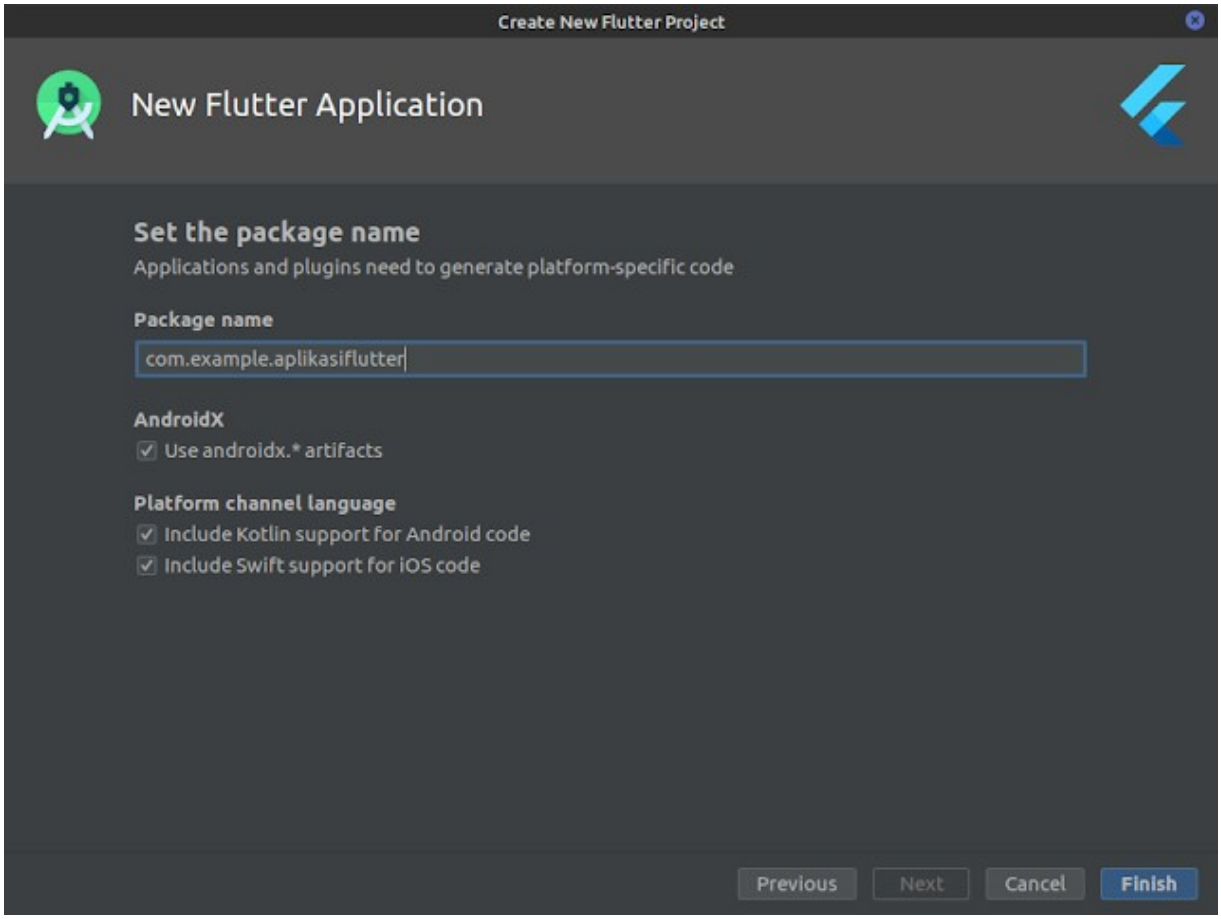
2. Pilih Flutter Aplication



3. Isi nama Project Name (harus menggunakan huruf kecil)



4. Package name dibairkan seperti aslinya saja kemudian Finish



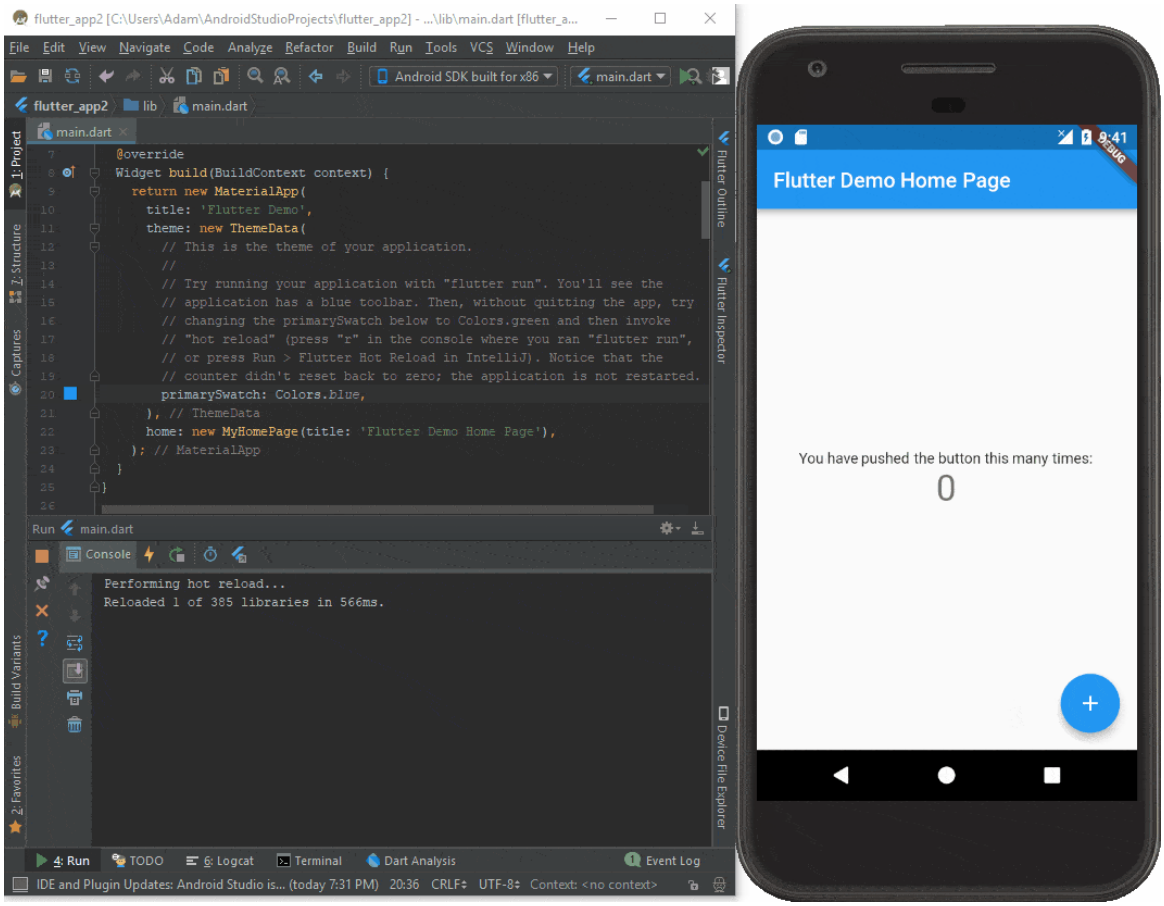
Tunggu sebentar untuk proses pembuatan project pada Flutter. Setelah selesai maka akan muncul kode **Project Flutter** yang merupakan bawaa dari Flutter sendiri.

Kode Program :

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       title: 'Flutter Demo',
10      debugShowCheckedModeBanner: false,
11      theme: ThemeData(
12        primarySwatch: Colors.blue,
13      ),
14      home: MyHomePage(title: 'Flutter Demo Home Page'),
15    );
```

```
16   }
17 }
18
19 class MyHomePage extends StatefulWidget {
20   MyHomePage({Key key, this.title}) : super(key: key);
21
22   final String title;
23
24   @override
25   _MyHomePageState createState() => _MyHomePageState();
26 }
27
28 class _MyHomePageState extends State<MyHomePage> {
29   int _counter = 0;
30
31   void _incrementCounter() {
32     setState(() {
33       _counter++;
34     });
35   }
36
37   @override
38   Widget build(BuildContext context) {
39     return Scaffold(
40       appBar: AppBar(
41         title: Text(widget.title),
42       ),
43       body: Center(
44         child: Column(
45           mainAxisAlignment: MainAxisAlignment.center,
46           children: <Widget>[
47             Text(
48               'You have pushed the button this many times:',
49             ),
50             Text(
51               '$_counter',
52               style: Theme.of(context).textTheme.headline4,
53             ),
54           ],
55         ),
56       ),
57       floatingActionButton: FloatingActionButton(
58         onPressed: _incrementCounter,
59         tooltip: 'Increment',
60         child: Icon(Icons.add),
61       ),
62     );
63   }
64 }
```

Apabila kode di atas di jalankan maka akan muncul tampilan sebagai berikut:



Gambar di atas merupakan aplikasi Flutter pertama kita yang secara default di generate oleh aplikasi Flutter. adapun kode nya bisa kita tambahkan dan kurangi sesuai dengan aplikasi yang akan kita buat.

C. Evaluasi

Setelah mempelajari modul ini, mahasiswa dapat mengetahui cara memulai untuk membuat sebuah aplikasi android.

Latihan	Sebagai bahan latihan, mahasiswa dapat mencoba menjalankan aplikasi android pertama secara mandiri.
Tugas	-
Refrensi	https://flutter.dev https://medium.com/community/flutter

BAB 3
MEMBUAT APLIKASI ANDROID PERTAMA

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang berbagai jenis tipe data yang dapat digunakan pada pemrograman dart, serta cara menggunakannya dalam kode program.

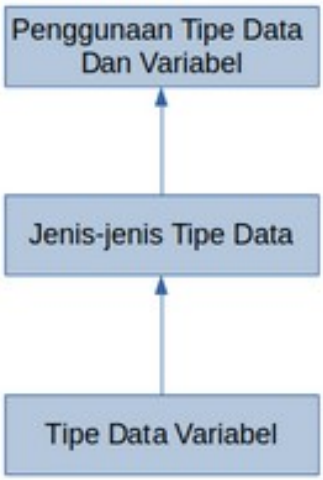
2 Capaian Pembelajaran

Mahasiswa diharapkan mampu mengenal jenis-jenis tipe data yang digunakan dalam pemrograman dart

3 Bahan Kajian

Pengenalan pemrograman mobile, variabel, tipe data, fungsi input dan output

4 Peta Kompetensi Modul



B. Kegiatan Belajar

	Pertemuan 4. Tipe Data dan Variabel

Pada materi kali ini kita akan mempelajari tentang tipe data dan variabel pada **Flutter**. Sebelumnya kita telah membahas bahwa **Flutter** dibuat menggunakan bahasa **Dart.** Maka begitu juga untuk tipe data dan variabel pada Flutter ini mengadopsi bahasa dasarnya.

1. Apa itu Dart ?

Dart merupakan bahasa pemrograman yang dikembangkan oleh Google, yang secara umum digunakan untuk membuat aplikasi *android, front-end web, IoT, backend (CLI),* dan *Game.* Itu artinya bahasa bahasa Dart ini merupakan bahasa pemrograman dengan tujuan yang umum.

2. Platform Dart

Palaftorm Dart digunakan pada tiga jenis *framework* seperti:



Pada kesempatan ini kita akan mempelajar Dart pada framework Flutter.

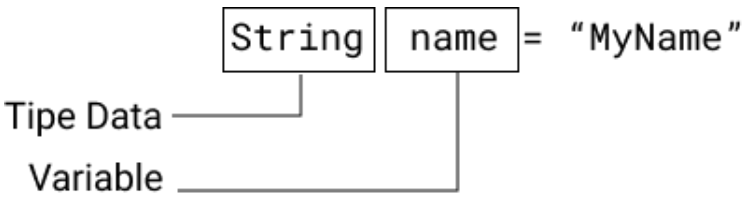
3. Variabel dan Tipe Data Flutter

Setiap bahasa pemrograman wajib hukumnya memiliki Variabel dan Tipe Data. Dan pada Flutter kita akan mengenal variabel dan tipe data dalam bahasa Dart. Dart sendiri adalah bahasa pemrograman basic yang berjalan di belakang Flutter.

Variabel merupakan sebuah nama bebas yang dapat digunakan untuk menyimpan sebuah nilai data-data.

Tipe Data merupakan nilai dari variabel.

Sebagai contoh.



pada gambar di atas dapat kita lihat bahwa String merupakan tipe data sedangkan Variabel -nya adalah name.

4. Variabel

Variabel menandakan bahwa terdapat penyimpanan suatu nilai. Variabel diperlukan untuk memberikan nama kepada nilai yang ada agar mudah untuk dikenali. Apabila tidak menggunakan variabel maka tentu nilai yang ada tidak tau akan disimpan dimana oleh program. Seperti misalnya, kita meminta Anda untuk mengambil suatu benda, sedangkan benda tersebut kita tidak tau namanya. Tentu Anda akan bingung sendiri. Nah itulah gunanya variabel agar memberikan nama pada sebuah nilai.

5. Aturan Penulisan Variabel pada Dart

Ada beberapa hal yang harus diperhatikan dalam penulisan variabel, agar tidak memiliki makna yang ambigu.

1. Variabel tidak boleh menggunakan nama semisal if, class, function, string atau integer.
2. Variabel tidak boleh menggunakan Angka semisal nama123.
3. Variabel tidak boleh menggunkan spasi tapi boleh menggunakan *camleCase* atau *underscore,* contoh namaMahasiswa atau nama_mahasiswa

6. Implementasi Variabel

Setelah kita mengetahui aturan dalam penulisan variabel kini saatnya kita praktikan bagaimana cara penulisan variabel yang benar.

```
var namaMahasiswa;
var nama_mahasiswa;
```

7. Tipe Data

Tipe data merupakan nilai yang tersimpan didalam variabel, banyak jenis tipe data yang ada dalam sebuah bahasa pemrograman. Contoh seperti:

(a) **Number**
(b) **Integer (Int)**
(c) **Double**
(d) **String**
(e) **Boolean**
(f) **List**
(g) **Sets**
(h) **Maps**

(i) **Runes**

(j) **Symbols**

8. Tipe Data dan Variabel Dart

Sebagai bahasa pemrograman dart memiliki tipe data yang sama seperti bahasa pemrograman lainnya, namun tipe data dasar pada dart sendiri terbagi menjadi tiga.

- Tipe data number (int, double)
- Tipe data boolean (bool)
- Tipe data teks (string)

Untuk membuat sebuah variabel pada Dart sendiri hampir sama cara penulisannya dengan Java, yaitu dengan menyebutkan tipe data nya kemudian variabelnya contoh:

```
1 var title = "Hello World !";
2 int angka = 2020;
3 double suhu = 37.8;
```

Pada kode di atas, kita bisa menuliskan tipe data dengan dua cara, yaitu menyebut langsung tipe datanya atau dengan menggunakan **var** yang nantinya bisa di isi dengan jenis tipe data apa saja.

9. Membuat Hello World pada Flutter

Pada pertemuan sebelumnya kita membuat aplikasi pertama flutter menggunakan kode default yang muncul ketika membuat project baru. Lalu bagaimana cara kita memanipulasi kode program yang ada tersebut. Pada materi ini kita akan bahas.

10. Hapus kode project

Langkah pertama yang perlu kita lakukan untuk memulai membuat kode aplikasi pada flutter yaitu dengan menghapus kode yang muncul secara default. Kemudian kita ganti dengan kode program yang ingin kita buat.

Pada kali ini kita akan membuat tampilan **Hello World !** menggunakan Widget Text, dengan kode sebagai berikut:

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(
4     Center(
5       child: Text(
6           "Hello World",
7           textDirection: TextDirection.ltr
```

```
 8        ),
 9      )
10    );
11 }
```

Ketika di jalankan pada layar *smartphone* akan muncul tulisan seperti gambar di bawah ini :



Pada kode :

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

merupakan library flutter yang berguna memanggil library material.dart.

```
1 Center(
2       child: Text(
3           "Hello World",
4           textDirection: TextDirection.ltr
5       ),
6     )
```

kode di atas akan secara otomatis membuat tulisa "Hello World" berda di posisi tengah dari layar *smartphone.*

C. Evaluasi

Setelah mempelajari modul ini, mahasiswa dapat memahami tentang Tipe Data dan Variabel yang digunakan dalam bahasa pemrograman dart, yang merupakan bahasa bawaan dari framework flutter.

Latihan	Mahasiswa diharapkan dapat mencoba secara mandiri tentang tipe data dan variabel.
Tugas	-
Refrensi	https://flutter.dev https://medium.com/community/flutter

BAB 5
METHOD DAN WIDGET

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Method dan Widget pada Flutter. Widget merupakan komponen penting dalam framework flutter, sehingga mahasiswa diharapkan mampu menjelaskan dan mempraktikan mengenai method dan widget.

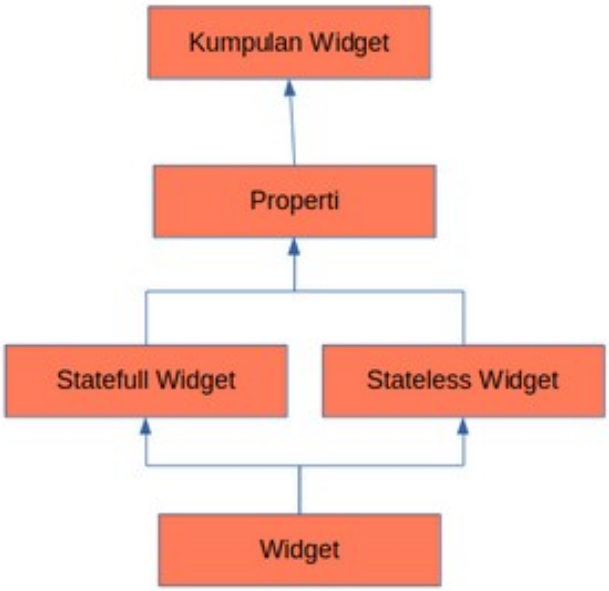
2 Capaian Pembelajaran

Mahasiswa mampu menjelaskan method serta dapat menggunakan method dan widget

3 Bahan Kajian

Operator, control flow, struktur data list, class, fungsi/method, stateful widget dan stateless widget pada flutter

4 Peta Kompetensi Modul



B. Kegiatan Belajar



Pertemuan 5. Pengenalan Method dan Widget pada Flutter

Pada materi kali ini kita bahas tentang fungsi Input dan Output, sebelum kita masuk lebih dalam terkait dengan Flutter, kita belajar sedikit tentang **Method dan Widget.** Dan pada Flutter kedua hal ini sangatlah penting. Method pada flutter sering disebut dengan Widget. Pada Flutter maka kita harus familiar dengan istliah widget ini karena memiliki berbagai macam fungsi yang dapat mendukung kita dalam membuat aplikasi android.

1. Apa itu Widget ?

Widget secara sederhana merupakan properti yang kita gunakan untuk membuat tampilan seperti tombol, list, icon, gambar, dll.

2. Properti Widget

Contoh misalnya kita akan membuat sebuah tombol yang memiliki background berwarna merah dan warna tulisannya berwarna putih.

Untuk membuat tombol tersebut maka kita perlu mengatur semuanya melalui properti. Seperti contoh kode berikut.

```
1 class MyHomePage extends SatelessWidget {
2 @overaid
3 Widget buld(BuildContext context) {
4    return Scaffold (
5         appBar: AppBar(
6            title: Text ("Tombol Pertama"),
7            backgroundColor: Colors.red,
8
9          ), //AppBar
10       ); // Scaffold
11   }
12 }
```

Dari kode di atas kita bisa mempelajari terkait ciri-ciri widget, widget selalu diawali dengan huruf kapital dan berupa class, ada beberapa widget yang ada pada kode di atas yaitu **Scaffold**, **AppBar** dan **Text**.

Dalam widget Scaffold memiliki properti berupa appBar , sedangkan pada widgat AppBar terdapat properti title dan pada widget Text terdapat properti **backgroundColor**.

Intinya bila diawali huruf kapital maka itu adalah widget, dan di dalam widget ada properti. Bisa saja di dalam widget ada widget lagi, namun pada intinya cara penulisannya sama saja.

3. Widget Umum

Widget yang paling umum digunakan ketika membuat aplikasi menggunakan Flutter yaitu :

a. Stateless

Merupakan widget yang memiliki fungsi untuk menampilkan sesuatu yang bersifat statis artinya tetap dan tidak akan berubah lagi.

Contoh :

```
1 class MyHomePage extends SatelessWidget {
2 @overaid
3 Widget buld(BuildContext context) {
4    return Scaffold ();
5   }
6 }
7
```

b. Statefull

Berbeda dengan stateless widget statefull digunakan untuk menampilkan data yang bersifat dinamis atau selalu mengalami perubahan.

Contoh :

```
 1 class MyWidget extends StatefulWidget{
 2 MyWidget({Key key}) : super(key:key);
 3     @override
 4     _MyWidget createState() => new _MyWidget();
 5 }
 6 class _MyWidget extends State<MyWidget>{
 7     Widget build(BuildContext context){
 8     return Widgets;
 9     }
10 }
```

c. Scaffold

Scaffold merupakan widget yang utama pada flutter, Scaffold memiliki beberapa parameter yang dapat digunakan untuk mengatur tampilan aplikasi android kita,

seperti AppBar serta body. Bahkan kita bisa merubah background body pada aplikasi melalui scaffold.

Contoh :

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Sample Code'),
    ),
    body: Center(
      child: Text('You have pressed the button $_count times.')
    ),
    backgroundColor: Colors.blueGrey.shade200,
    floatingActionButton: FloatingActionButton(
      onPressed: () => setState(() => _count++),
      tooltip: 'Increment Counter',
      child: const Icon(Icons.add),
    ),
  );
}
```

d. Layout

Layout berguna untuk mengatur letak atau posisi widget seperti **Row** atau **Column** Untuk menampilkan widget bersifat horizontal kita bisa menggunakan **Row.** Sedangkan untuk widget yang bersifat vertikal bisa kita gunakan **Column**.

Contoh :

```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text("Row"),
        ),
        body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Container(
              child: Text("Halo 1 !!!"),
              color: Colors.lime,
              padding: EdgeInsets.all(16.0),
            ),
            Container(
              child: Text("Halo 2 !!!"),
              color: Colors.purple,
              padding: EdgeInsets.all(16.0),
```

```
            ),
            Container(
              child: Text("Halo 3 !!!"),
              color: Colors.lightBlue,
              padding: EdgeInsets.all(16.0),
            ),
          ],
        ));
  }
}
```

e. Container

Container merupakan widget yang berguna sebagai pembungkus widget lain sehingga dapat diberikan margin, padding, warna bakground ataupun dekorasi.

Contoh :

```
...
body: Container(
          padding: EdgeInsets.all(32.0),
          margin: EdgeInsets.fromLTRB(20.0, 10.0, 20.0, 0),
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(20.0),
            color: Colors.purple),
          child: Text('Haiii', style:
TextStyle(color: Colors.white, fontSize: 20.0),)
        )
…
```

Sekian dulu pembahasan kita kali ini, semoga ada manfaatnya, untuk praktiknya silahkan dicoba masing-masing.

C. Evaluasi

Setelah mempelajari modul ini, mahasiswa dapat memahami tentang Widget yang digunakan dalam bahasa pemrograman dart, yang merupakan bahasa bawaan dari framework flutter.

Latihan	Mahasiswa diharapkan dapat mencoba secara mandiri tentang berbagai jenis widget yang digunakan pada flutter
Tugas	-
Refrensi	https://flutter.dev

BAB 4
METHOD DAN WIDGET

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Method dan Widget pada Flutter. Widget merupakan komponen penting dalam framework flutter, sehingga mahasiswa diharapkan mampu menjelaskan dan mempraktikan mengenai method dan widget.

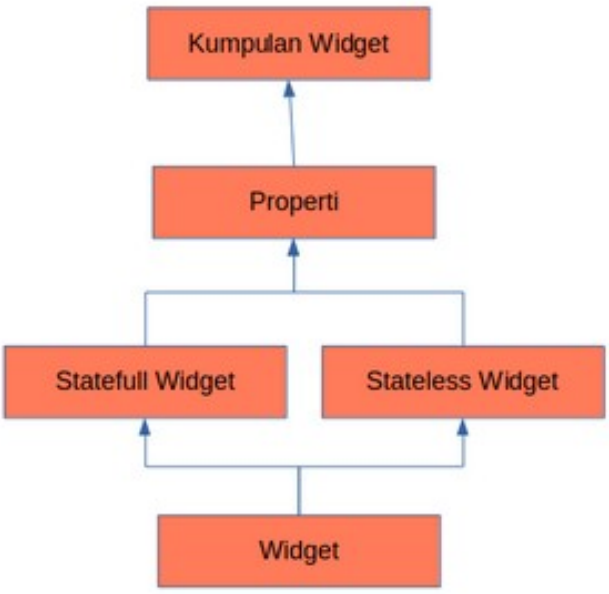
2 Capaian Pembelajaran

Mahasiswa mampu menjelaskan method serta dapat menggunakan method dan widget

3 Bahan Kajian

Operator, control flow, struktur data list, class, fungsi/method, stateful widget dan stateless widget pada flutter

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 6. Stateless Widget dan Statefull WIdget

Materi kali ini kita akan membahas sedikit terkait bagaimana implementasi statefull widget dan tateless widget dua hal ini memiliki fungsi yang berbeda, tergantung kebutuhan pembuat aplikasi. Namun karena kita masih sama sama belajar, mungkin agak sedikit susah menentukan kapan pakai statefull widget dan kapan harus memakai stateless widget untuk itu mari kita simak bersama sama.

1. Apa sih Stateless Widget

Seperti pertemuan sebelumnya, kita telah menjelaskan bahwa stateless widget merupakan widget yang bersifat statis, artinya widget tersebut tidak akan pernah berubah. Makanya stateless widget biasanya telah di rancang terlebih dahulu contohnya seperti **AppBar.**



Seperti gambar di atas, itu adalah contoh Appbar, AppBar yang dibuat statis artinya AppBar tidak akan pernah berubah, untuk mencobanya silahkan menggunakan kode berikut.

```
1 class MyApp extends StatelessWidget {
2     @override
3     Widget build(BuildContext context) {
4         return Scaffold(
5             appBar: AppBar(
6                 title: Text('JuniorAcademy'),
7             ),
8             body: Center(
9                 child: Text(
10                    '10',
```

```
                    style: TextStyle(
                        fontSize: 30
                    ),
                ),
            )
        );
    }
}
```

Perhatikan kode di atas

```
class MyApp extends StatelessWidget { //kode }
```

Stateless widget ditandai dengan kode seperti di atas, StatelessWidget. yang akan me return nilai **Widget Scaffold** dimana **Scaffold memiliki** atrribut yaitu appBar, title, body, child dan seterusnya.

2. Statefull Widget

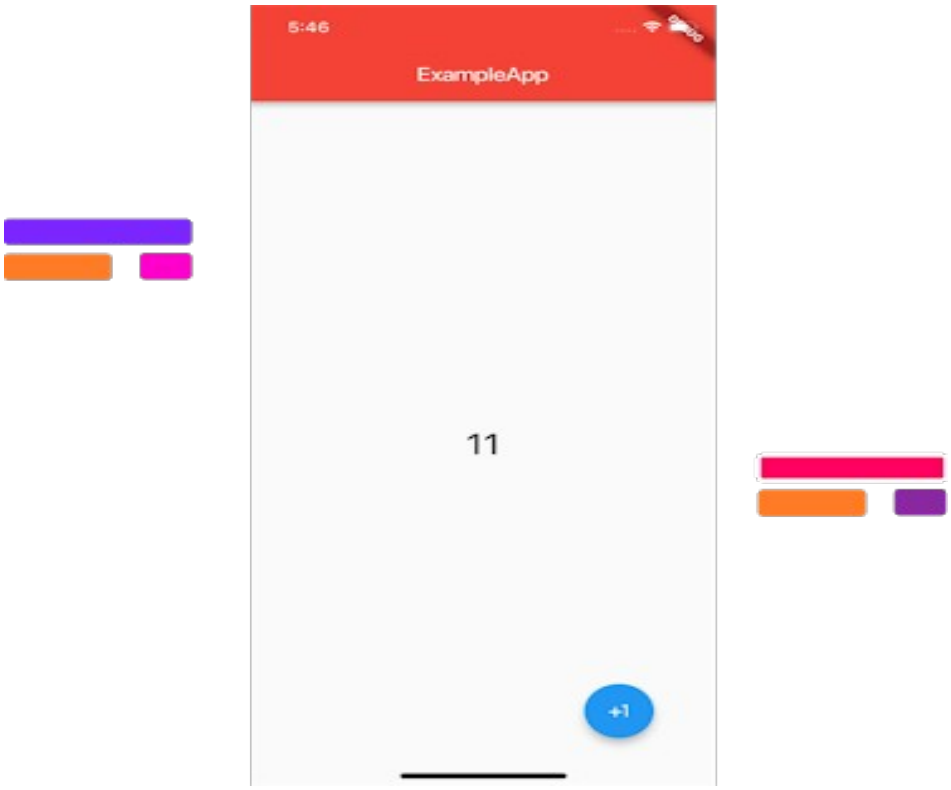
Hal yang paling menarik dari flutter yaitu adanya Statefull Widget, karena widget ini merupakan widget yang bersifat dinami, artinya widget ini dapat berubah ubah tergantung kebutuhan. Contohnya membuat form login, membuat counter atau membuat apa saja yang nilainya bisa berubah. Dalam Statefull Widget biasnaya ditandai dengan adapanya fungsi setState () yang berfungsi untuk memberitahu bahwa objek ada yang berubah pada State.

Contoh kode sebagai berikut :

```
class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('ExampleApp'),
            ),
            body: Container(
                child: NumberScreen(),
            )
        );
    }
}
class NumberScreen extends StatefulWidget {
  @override
  _NumberScreenState createState() => _NumberScreenState();
}
class _NumberScreenState extends State<NumberScreen> {
    int number = 10;
    @override
    Widget build(BuildContext context) {
        return Stack(
```

```
23              children: <Widget>[
24                  Center(
25                      child: Text(
26                          this.number.toString(),
27                          style: TextStyle(
28                              fontSize: 30
29                          ),
30                      ),
31                  ),
32                  Positioned(
33                      bottom: 50,
34                      right: 50,
35                      child: FloatingActionButton(
36                          child: Icon(Icons.plus_one),
37                          onPressed: () {
38                              setState(() {
39                                  this.number += 1;
40                              });
41                          },
42                      ),
43                  )
44              ],
45          );
46      }
47  }
```

Bila dijalankan akan mirip seperti tampilan aplikasi default yang ter-Create ketika memulai flutter pertama kali.



setState() pada Stateless Widget

Selanjutnya muncul pertanyaan apakah setState() dapat digunakan pada stateless widget ?, seperti yang telah kita jelaskan bahwa setState() merupakan ciri dari statefull widget.

Kesimpulan

Secara umum, bahwa kedua widget ini memiliki fungsi dan kegunaan masing masing, bahkan bisa digunakan secara bersandingan.

C. Evaluasi

Setelah mempelajari modul ini, mahasiswa dapat memahami tentang Widget yang digunakan dalam bahasa pemrograman dart, yang merupakan bahasa bawaan dari framework flutter.

Latihan	Mahasiswa diharapkan dapat mencoba secara mandiri tentang berbagai jenis widget yang digunakan pada flutter
Tugas	-
Refrensi	https://flutter.dev

BAB 5
MEMBUAT APLIKASI SEDERHANA DENGAN FLUTTER

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang membuat aplikasi sederhana dengan flutter, yaitu menampilkan halaman sebuah wibsite dengan memanfaatkan sebuah plugin webview.

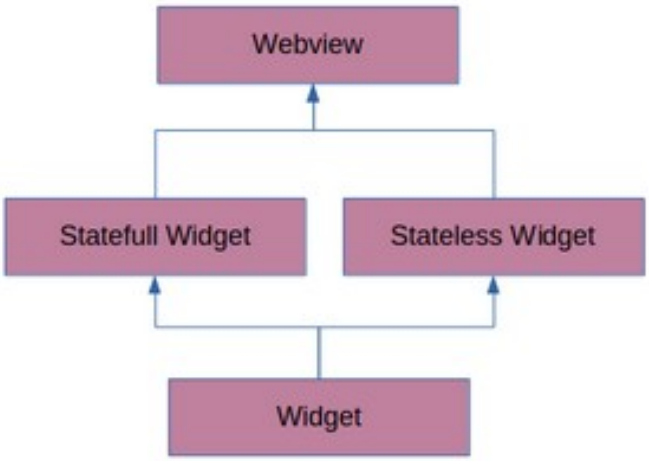
2 Capaian Pembelajaran

Mahasiswa mampu membuat webview android dengan flutter

3 Bahan Kajian

Penggunaan Webview

4 Peta Kompetensi Modul



B. Kegiatan Belajar

	Pertemuan 7. Membuat Aplikasi Sederhana dengan Webview

Pada materi kali ini kita akan belajar membuat sebuah aplikasi sederhana dengan menggunakan flutter, setelah beberapa pertemuan sebelumnya kita telah membahas terkait dengan teori flutter. Mari kita praktikan bersama ilmu yang telah kita pelajari sebelumnya.

Pada pertemuan ini kita akan membuat sebuah webview menggunakan flutter, kenapa kita belajar membuat sebuah webview ?

Karena membuat webview ini termasuk hal yang mudah, dan kita yakin ini akan sangat bermanfaat bagi semua, karena nanti webview bisa langsung di upload ke playstore. Mari kita mulai.

1. Membuat AppBar

Sebuah aplikasi android sejatinya selalu memilik yang namnya AppBar. AppBar ini nanti berfungsi sebagai header yang menunjukkan nama aplikasi atau judul halaman yang sedang tampil di layar user.

```
1 import 'package:flutter/material.dart';
2 void main() => runApp(MyWebView());
3 class MyWebView extends StatelessWidget {
4   @override
5   Widget build(BuildContext context) {
6     return Scaffold(
7       appBar: AppBar(title: Text("Webview Aplikasi")),
8     );
9   }
10 }
```

Baik coba perhatikan kode di atas.

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

Kod ini berfungsi memanggil material.dart yang berisi berbagai fungsi dasar seperti widget - widget yang telah kita pelajari sebelumnya.

```
void main() => runApp(MyWebView());
```

Kode ini berfungsi untuk melakukan perintah kepada aplikasi agar menjalankan kode di atas untuk pertama kali. Jadi apabila aplikasi di jalankan, kode yang pertama kali di eksekusi oleh aplikasi adalah widget dengan nama **MyWebView**. Semoga bisa dipahami.

```
1 class MyWebView extends StatelessWidget {
2   @override
3   Widget build(BuildContext context) {
4     return Scaffold(
5       appBar: AppBar(title: Text("JuniorAcademy Webview")),
6
7     );
8   }
9 }
```

Selanjutnya kode ini, berisi widget dengan nama JuniorAcademy Webview dimana widget digunakan yaitu StatelessWidget. seperti yang sudah kita pelajari fungsi dari widget tersebut yaitu untuk menampilkan sesuatu tampilan yang bersifat statis, artinya bersifat tetap, contohnya seperti AppBar yang kita buat.

```
1 return Scaffold(
2
3       appBar: AppBar(title: Text("Webview Aplikasi")),
4
5     );
```

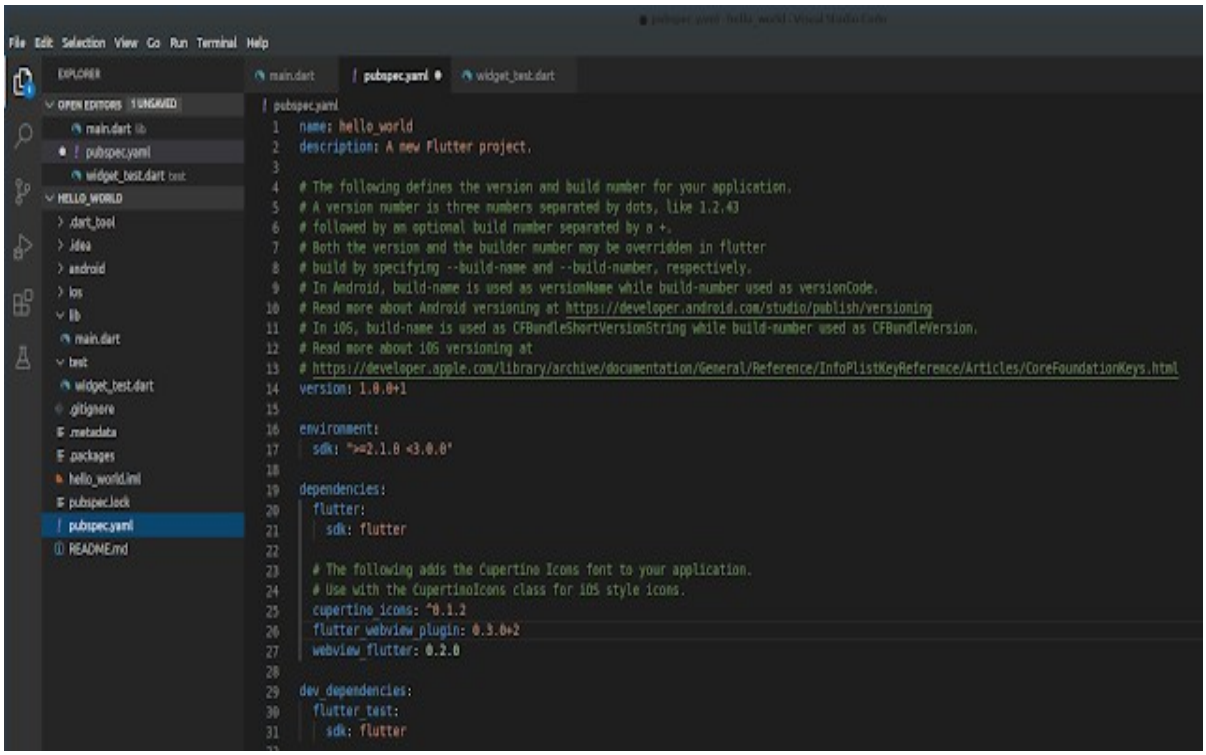
Dari **statelesswidget** tersebut terdapat kembalian berupa Scaffold dimana Scaffold tersebut menampilkan widget AppBar yang diberi judul **Webview** Aplikasi.

2. Memanggil Webview

Baik langkah selanjutnya, kita harus menambahkan plugin webview pada pubspec.yaml. plugin tersebut berguna memanggil fungsi webview.

```
flutter_webview_plugin: 0.3.0+2
webview_flutter: 0.2.0
```

Untuk lokasinya bisa cek gambar di bawah ini.



Setelah kita mendapatkan plugin dari webview, kita tinggal menambahkan sedikit kode pada kode sebelumnya yang telah kita buat.

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_webview_plugin/flutter_webview_plugin.dart';
3 void main() => runApp(MyWebView());
4
5 class MyWebView extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return WebviewScaffold(
9       url: 'UTM Mataram',
10      hidden: true,
11      appBar: AppBar(title: Text("JuniorAcademy Webview")),
12
13    );
14  }
15 }
```

```
import 'package:flutter_webview_plugin/flutter_webview_plugin.dart';
```

Pada kode di atas, kita menambahkan perintah memanggil plugin webview.

```
1 return WebviewScaffold(
2       url: 'https://junioracademy.id',
3       hidden: true,
4       appBar: AppBar(title: Text("UTM Mataram")),
5
6     );
```

sedangkan untuk kode di atas ini, untuk memanggil halaman website https://utmmataram.ac.id.id

c. Evaluasi

Setelah memepelajari materi pada modul ini, mahasiswa dapat membuat sebuah webview untuk website

Latihan	Mahasiswa dapat mencoba secara mandiri untuk membuat sebuah webview
Tugas	Mahasiswa ditugaskan untuk membuat sebuah aplikasi webview dengan menggunakan flutter.
Refrensi	https://flutter.dev

BAB 6
LAYOUT

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang desain layout pada flutter. Layout merupakan suatu teknik untuk mengatur tata letak dari komponen-komponen pada aplikasi android agar terlihat menarik.

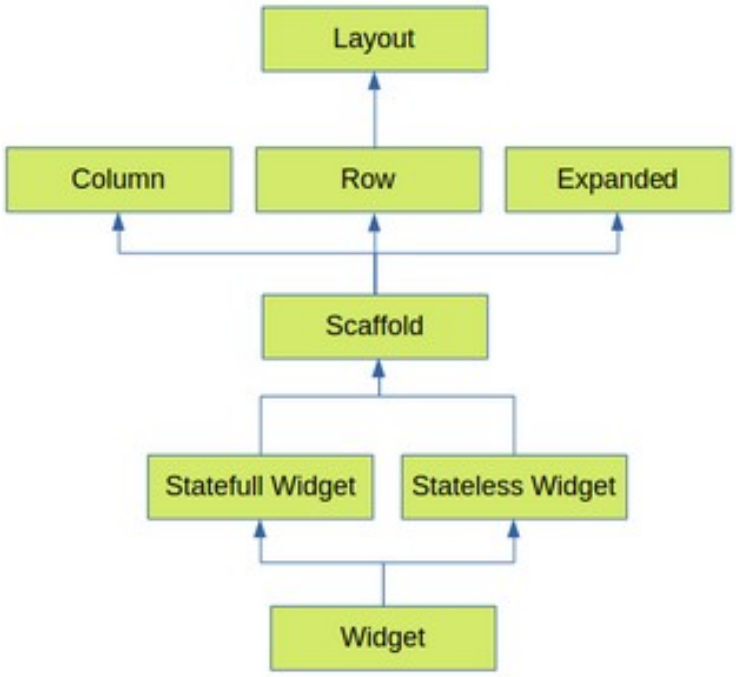
2 Capaian Pembelajaran

Mahasiswa mampu membuat desain layout pada android yang user friendly menggunakan

3 Bahan Kajian

Layout, App Bar & Tab Bar Widget, Custom Widget, Input, Selection

4 Peta Kompetensi Modul



B. Kegiatan Belajar

	Pertemuan 9. Membuat Desain Layout

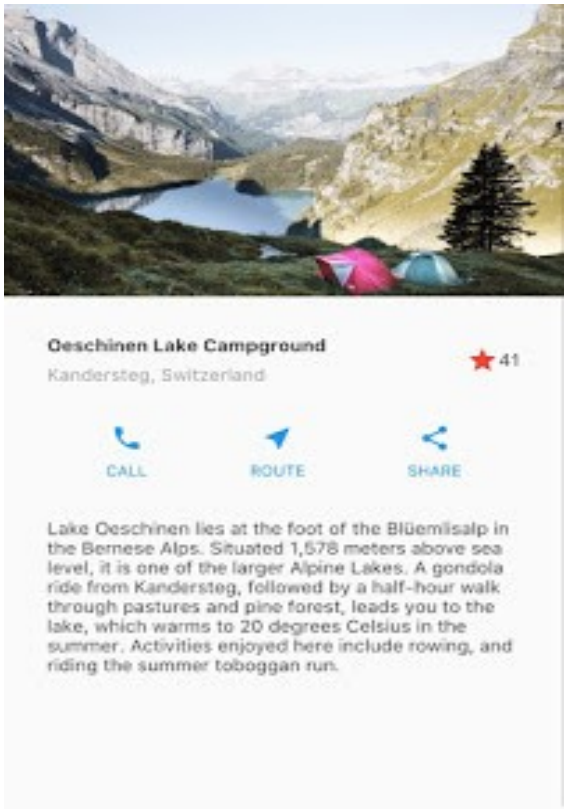
Pertemuan selanjutnya, kita akan membahas tentang bagaimana membuat layout dalam Flutter.

1. Layout

Apasih itu layout ?
Layout merupakan sebuah teknik untuk menagatur tata letak dari suatu desain yang telah dibuat dan ditempatkan dalam sebuah bidang dan menggunakan media yang telah di konsep terlebih dahulu. Lantas bagaimana kita membuat sebuah layout pada Flutter ?

2. Membuat Layout Flutter

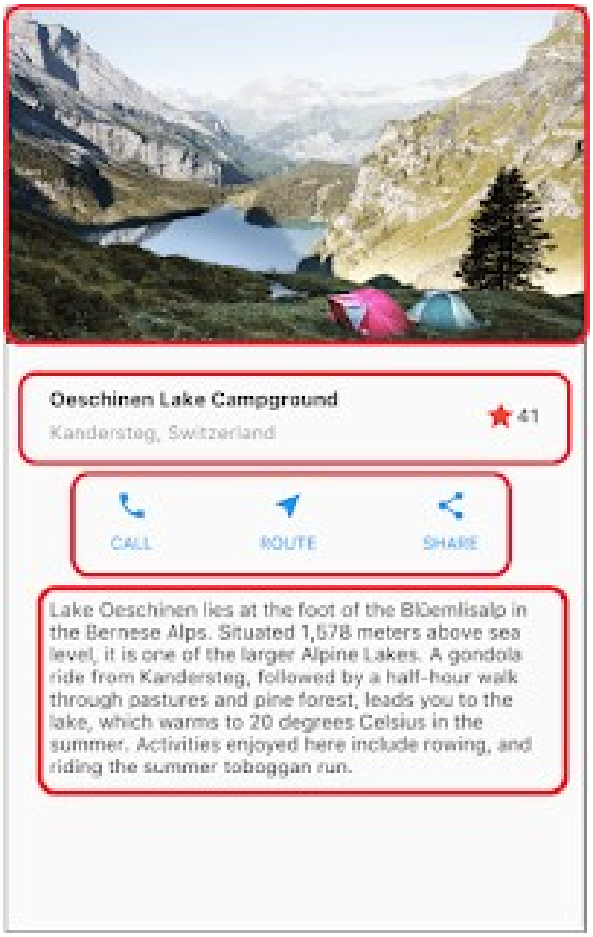
Pertama kita harus memahami, konsep dalam membuat layout terlebih dahulu, terutama dalam Flutter. Untuk lebih jelasnya kita akan membahas sedikit tentang Layout pada flutter pada gambar di bawah ini.



Perhatikan gambar, kita akan membuat layout seperti gambar di atas misalnya. Hal pertama yang kita lakukan yaitu membagi gambar menjadi beberapa bagian, agar memudahkan kita ketika menerjemahkannya dalam sebuah koda pemrograman.

- Identifikasi baris dan kolom.
- Apakah layout termasuk ada grid?
- Apakah ada elemen yang tumpang tindih?
- Apakah UI memerlukan tab?

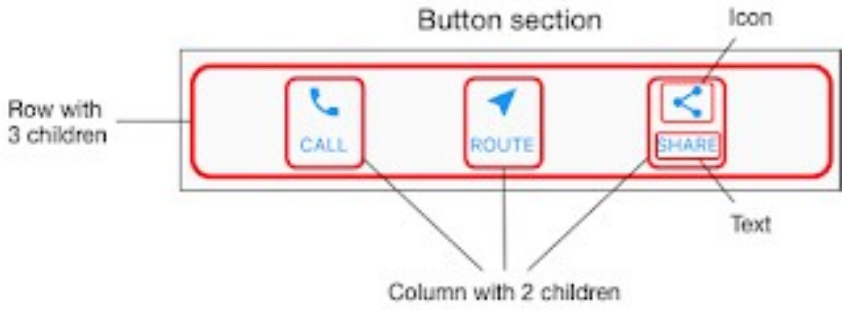
- Perhatikan area yang membutuhkan padding, margin, atau batas.



Sehingga kita dapat menghasilkan pembagian seperti gambar di atas, namun pembagian tersebut bisa kita lebih rincikan seperti berikut.



Pada gambar di atas, kita bisa membagi menjadi beberapa bagian yant terdiri dari, title, icon, text dan row.



Begitupun dengan button section, kita bisa membagi menjadi bagian seperti terdiri dari row, colum dan icon.

3. Menerjemahkan dalam kode pemrograman

- **Membuat project baru pada flutter**

Pada langkah ini, kita membuat project baru yang kemudian pada file main.dart, kita sisakan kode seperti berikut :

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   // debugPaintSizeEnabled = true;
5   runApp(MyApp());
6 }
```

- **Menambahkan Judul**

Pertama, kita akan membangun kolom kiri di bagian judul. Tambahkan Stateless Widget pada kode berikut di bagian atas metode build () dari kelas **MyApp**.

```
 1 class MyApp extends StatelessWidget {
 2   @override
 3   Widget build(BuildContext context) {
 4     Widget titleSection = Container(
 5       padding: const EdgeInsets.all(32),
 6       child: Row(
 7         children: [
 8           Expanded(
 9             /*1*/
10             child: Column(
11               crossAxisAlignment: CrossAxisAlignment.start,
12               children: [
13                 /*2*/
14                 Container(
15                   padding: const EdgeInsets.only(bottom: 8),
16                   child: Text(
```

```
                    'Oeschinen Lake Campground',
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
                Text(
                  'Kandersteg, Switzerland',
                  style: TextStyle(
                    color: Colors.grey[500],
                  ),
                ),
              ],
            ),
          ),
          Icon(
            Icons.star,
            color: Colors.red[500],
          ),
          Text('41'),
        ],
      ),
    );
```

- **Membuat Button Section**

Bagian tombol berisi 3 kolom yang menggunakan layoutyang sama dan ikon di atas baris teks.

```
Color color = Theme.of(context).primaryColor;
   Widget buttonSection = Container(
     child: Row(
       mainAxisAlignment: MainAxisAlignment.spaceEvenly,
       children: [
         _buildButtonColumn(color, Icons.call, 'CALL'),
         _buildButtonColumn(color, Icons.near_me, 'ROUTE'),
         _buildButtonColumn(color, Icons.share, 'SHARE'),
       ],
     ),
   );
```

- **Membuat Text**

Tentukan bagian teks sebagai variabel. Letakkan teks dalam Wadah dan tambahkan padding di sepanjang setiap tepi. Tambahkan kode berikut tepat di bawah deklarasi buttonSection:

```
Widget textSection = Container(
     padding: const EdgeInsets.all(32),
     child: Text(
       'Lake Oeschinen lies at the foot of the Blüemlisalp in the Bernese '
```

```
 5         'Alps. Situated 1,578 meters above sea level, it is one of the '
 6         'larger Alpine Lakes. A gondola ride from Kandersteg, followed by a '
 7         'half-hour walk through pastures and pine forest, leads you to the '
 8         'lake, which warms to 20 degrees Celsius in the summer. Activities '
 9         'enjoyed here include rowing, and riding the summer toboggan run.',
10         softWrap: true,
11       ),
12     );
```

- **Menampilkan Gambar**

Pada bagian ini, kita akan menambahkan sebuah gambar, gambar bebas. Sebagai contoh kita bisa gunakan gambar di bawah ini.



Selanjutnya gambar ini, kita simpan dalam project dengan folder baru bernama **asset**. Agar gambar bisa terbaca maka, pada file **pubspec.yaml** kita tambahkan perintah berikut.

```
1 name: layout
2 description: >
3   Sample app from "Building Layouts",
4 https://flutter.io/docs/development/ui/layout.
5 version: 1.0.0
6 environment:
7   sdk: '>=2.0.0-dev.68.0 <3.0.0'
8
```

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^0.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter

flutter:
  uses-material-design: true
  assets:
    - images/lake.jpg
```

Setelah mengatur pubspect.yaml ini, kita lanjutkan kode pada bagian main.dart dengan kode berikut.

```
return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter layout demo'),
        ),
        body: ListView(
          children: [
            Image.asset(
              'images/lake.jpg',
              width: 600,
              height: 240,
              fit: BoxFit.cover,
            ),
            titleSection,
            buttonSection,
            textSection,
          ],
        ),
      ),
    );
  }
```

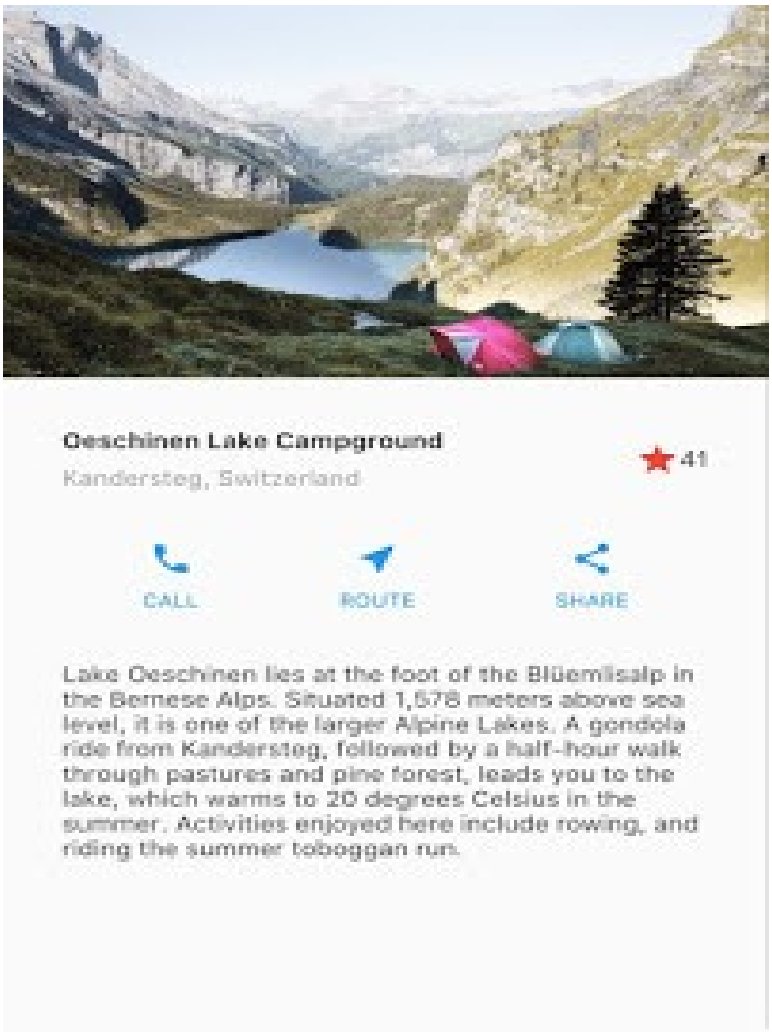
- **Sentuhan terakhir**

Pada langkah terakhir ini, atur semua elemen dalam ListView, bukan Kolom, karena **ListView** mendukung scroll pada aplikasi ketika aplikasi dijalankan pada perangkat kecil.

```
Column _buildButtonColumn(Color color, IconData icon, String label) {
   return Column(
     mainAxisSize: MainAxisSize.min,
     mainAxisAlignment: MainAxisAlignment.center,
     children: [
       Icon(icon, color: color),
```

```
 7         Container(
 8           margin: const EdgeInsets.only(top: 8),
 9           child: Text(
10             label,
11             style: TextStyle(
12               fontSize: 12,
13               fontWeight: FontWeight.w400,
14               color: color,
15             ),
16           ),
17         ),
18       ],
19     );
20   }
21 }
```

Bila dijalnkan maka akan menampilkan gambar sebagai berikut.



C. Evaluasi

Setelah mempelajari modul ini, mahasiswa dapat memahami penggunaan layout pada pemrograman mobile.

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang cara penggunaan layout pada flutter
Tugas	Mahasiswa ditugaskan untuk membuat contoh layout berupa column, row dan expanded.
Refrensi	https://flutter.dev https://medium.com/community/flutter

BAB 6
LAYOUT

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang desain layout pada flutter. Layout merupakan suatu teknik untuk mengatur tata letak dari komponen-komponen pada aplikasi android agar terlihat menarik.

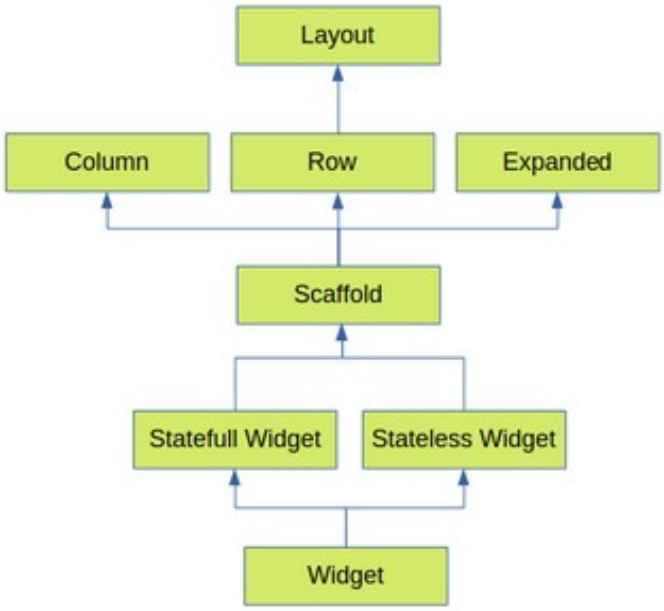
2 Capaian Pembelajaran

Mahasiswa mampu membuat desain layout pada android yang user friendly menggunakan

3 Bahan Kajian

Layout, App Bar & Tab Bar Widget, Custom Widget, Input, Selection

4 Peta Kompetensi Modul



B. Kegiatan Belajar

	Pertemuan 10. Membuat Drawer

Navigation Drawer merupakan menu navigasi yang tampil penuh pada sisi kanan atau kiri sebuah aplikasi. Untuk memunculkan Navigation Drawer bisa dengan cara di

mengeser / swipe layar atau dengan menekan icon menu pada appBar. Pada Flutter, cara membuat navigation drawer sangat mudah yaitu hanya menggunakan widget **Drawer** yang dapat kita tempatkan pada properti drawer atau endDrawer di **Scaffold** Widget.

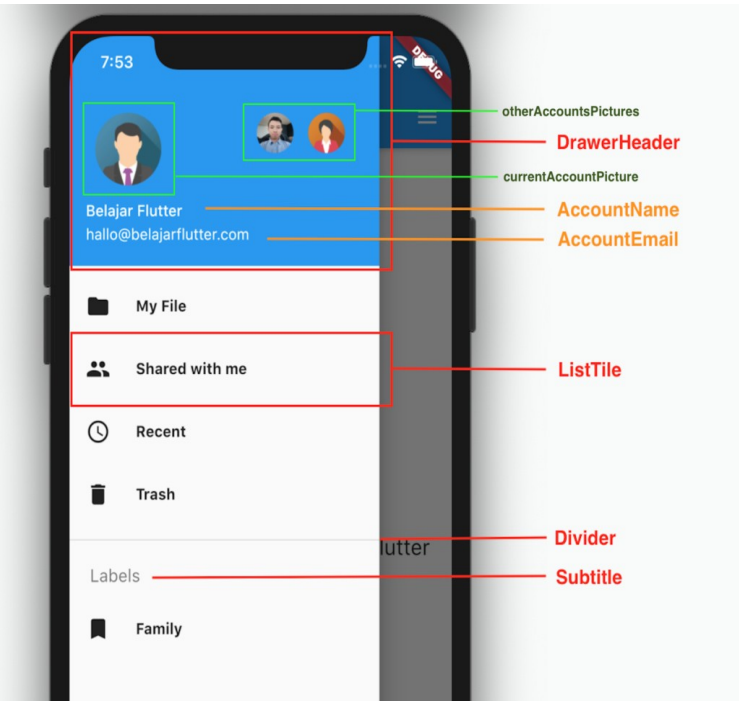
1. Mengenal Drawer Widget di Flutter

Sebelum ke tahap cara membuat navigation drawer, ada baiknya kita mengerti tentang Drawer itu sendiri. Drawer widget merupakan single child widget yang artinya hanya dapat memiliki satu child widget di dalamnya. Karena Drawer hanya memiliki properti child dan bukan children, maka untuk menampatkan item-item lain bisa menggunakan ListView widget. Sebenarnya tidak ada keharusan menggunakan ListView namun keuntungan menggunakan ListView widget dibandingkan dengan column atau widget lainnya yaitu untuk memudahkan dalam mengatur list Item dan vertical scrolling apabila item menu melebihi tinggi layar.

2. Anatomi Drawer

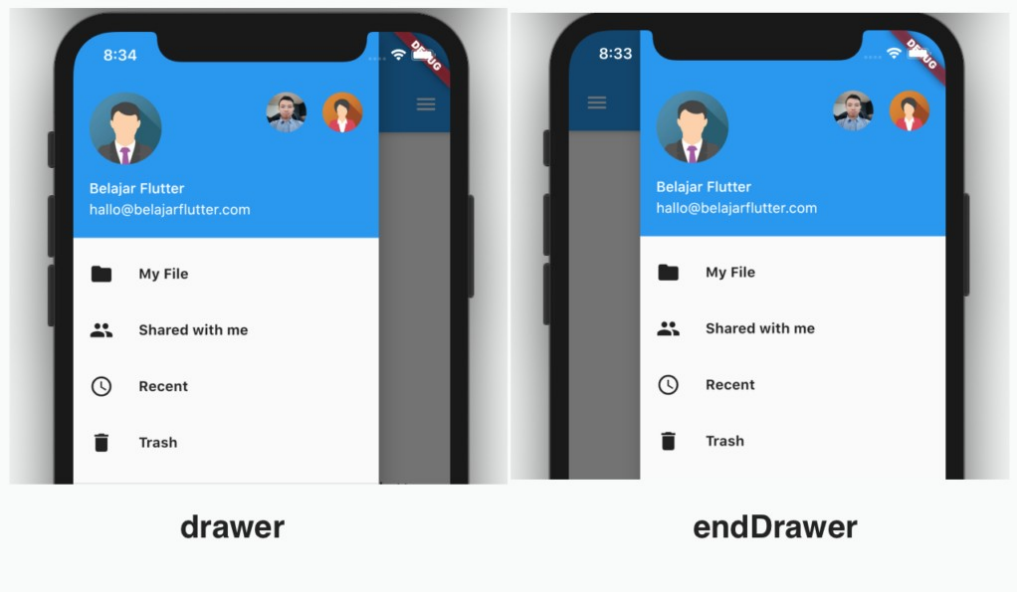
Anatomi dari sebuah navigation drawer menurut material design itu ada delapan (8) point, namun disini kita sederhanakan saja menjadi 4 bagian yaitu

1. **Header,**
2. **List Item,**
3. **Divider dan**
4. **Subtitle**

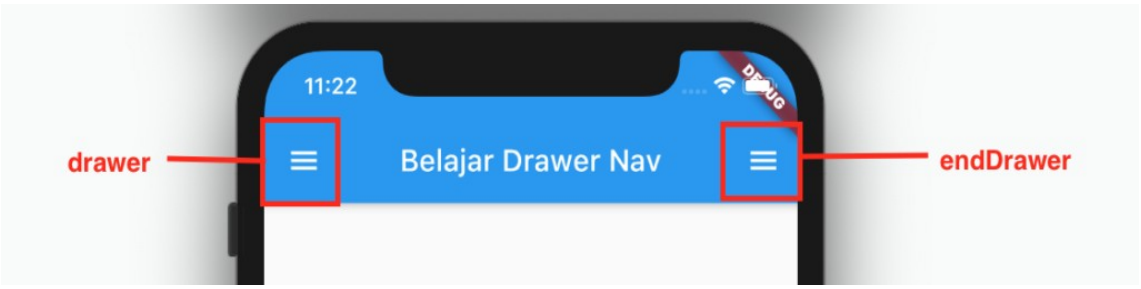


3. Jenis Drawer

Jenis Drawer navigasi pada flutter dibagi menjadi dua sesuai dengan letak posisi drawer itu sendiri yaitu drawer dan endDrawer. Drawer yang muncul di sebelah kiri dinamakan drawer, sedangkan apabila muncul dari sebelah kanan dinamakan endDrawer karena menggunakan properti endDrawer



Posisi icon masing-masing drawer berbeda sesuai dengan properti yang digunakan



4. Cara membuat navigation drawer pada flutter :

(1) Buat projek flutter baru dengan nama **belajar_flutter** atau gunakan projek flutter yang ada. Dalam folder lib, buat file baru bernama **nav-drawer.dart**

```
lib
  main.dart
  nav-drawer.dartCopy
```

(2) Buka file **nav-drawer.dart** dan ketikan kode dibawah ini

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

class DrawerWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: <Widget>[
          _drawerHeader(),
          _drawerItem(
              icon: Icons.folder,
              text: 'My Files',
              onTap: () => print('Tap My Files')),
          _drawerItem(
              icon: Icons.group,
              text: 'Shared with me',
              onTap: () => print('Tap Shared menu')),
          _drawerItem(
              icon: Icons.access_time,
              text: 'Recent',
              onTap: () => print('Tap Recent menu')),
          _drawerItem(
              icon: Icons.delete,
              text: 'Trash',
              onTap: () => print('Tap Trash menu')),
          Divider(height: 25, thickness: 1),
          Padding(
            padding: const EdgeInsets.only(left: 20.0, top: 10, bottom: 10),
            child: Text("Labels",
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black54,
                )),
          ),
          _drawerItem(
              icon: Icons.bookmark,
              text: 'Family',
              onTap: () => print('Tap Family menu')),
        ],
      ),
    );
  }
}
```

Perhatikan fungsi **_drawerHeader** dan **_drawerItem** pada kode diatas.

1. **_drawerHeader** : Sebuah local function yang akan me-return DrawerHeader widget
2. **_drawerItem** : Lokal Function untuk me-return DrawerItem widget

3. **Divider** : untuk membuat garis pemisah antar menu

4. Padding dan Text digunakan untuk membuat **subtitle** navigation bar. Kita tidak pisahkan ke function terpisah seperti drawerHeader dan drawerItem karena bentuknya yang sederhanan dan hanya menggunakannya satu kali.

Underscore (_) pada nama function untuk menandakan bahwa variable atau function tersebut merupakan lokal function atau hanya digunakan pada file ini saja.

(3) Masih pada file **nav-drawer.dart**, tambahkan function **_drawerHeader**() dibawah

```
Widget _drawerHeader() {
  return UserAccountsDrawerHeader(
    currentAccountPicture: ClipOval(
      child: Image(
          image: AssetImage('assets/images/orang1.jpeg'), fit:
BoxFit.cover),
    ),
    otherAccountsPictures: [
      ClipOval(
        child: Image(
            image: AssetImage('assets/images/orang2.jpg'), fit:
BoxFit.cover),
      ),
      ClipOval(
        child: Image(
            image: AssetImage('assets/images/orang3.jpeg'), fit:
BoxFit.cover),
      )
    ],
    accountName: Text('Belajar Flutter'),
    accountEmail: Text('test@gmail.com'),
  );
```

Header yang kita gunakan yaitu UserAccountsDrawerHeader widget dimana memiliki empat properti utama yaitu : accountName, accountEmail, currentAccountPicture, otherAccountsPictures. current dan other picture properties digunakan untuk menampilkan foto.

(4) Setelah header maka selanjutnya kita buat **_drawerItem** widget kode nya seperti di bawah ini

```
Widget _drawerItem({IconData icon, String text, GestureTapCallback onTap}) {
  return ListTile(
    title: Row(
      children: <Widget>[
        Icon(icon),
        Padding(
          padding: EdgeInsets.only(left: 25.0),
```

```
          child: Text(
            text,
            style: TextStyle(
              fontWeight: FontWeight.bold,
            ),
          ),
        ),
      ],
    ),
    onTap: onTap,
  );
}
```

Hasil akhir kode dari file **nav-drawer.dart** menjadi seperti ini

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

class DrawerWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: <Widget>[
          _drawerHeader(),
          _drawerItem(
              icon: Icons.folder,
              text: 'My Files',
              onTap: () => print('Tap My Files')),
          _drawerItem(
              icon: Icons.group,
              text: 'Shared with me',
              onTap: () => print('Tap Shared menu')),
          _drawerItem(
              icon: Icons.access_time,
              text: 'Recent',
              onTap: () => print('Tap Recent menu')),
          _drawerItem(
              icon: Icons.delete,
              text: 'Trash',
              onTap: () => print('Tap Trash menu')),
          Divider(height: 25, thickness: 1),
          Padding(
            padding: const EdgeInsets.only(left: 20.0, top: 10, bottom: 10),
            child: Text("Labels",
                style: TextStyle(
                  fontSize: 16,
                  color: Colors.black54,
                )),
          ),
          _drawerItem(
              icon: Icons.bookmark,
              text: 'Family',
```

```
39                onTap: () => print('Tap Family menu')),
40          ],
41        ),
42      );
43    }
44  }
45
46  Widget _drawerHeader() {
47    return UserAccountsDrawerHeader(
48      currentAccountPicture: ClipOval(
49        child: Image(
50            image: AssetImage('assets/images/orang2.jpeg'), fit: BoxFit.cover),
51      ),
52      otherAccountsPictures: [
53        ClipOval(
54          child: Image(
55              image: AssetImage('assets/images/orang1.jpg'), fit: BoxFit.cover),
56        ),
57        ClipOval(
58          child: Image(
59              image: AssetImage('assets/images/orang3.jpeg'), fit: BoxFit.cover),
60        )
61      ],
62      accountName: Text('Belajar Flutter'),
63      accountEmail: Text('hallo@belajarflutter.com'),
64    );
65  }
66
67  Widget _drawerItem({IconData icon, String text, GestureTapCallback onTap}) {
68    return ListTile(
69      title: Row(
70        children: <Widget>[
71          Icon(icon),
72          Padding(
73            padding: EdgeInsets.only(left: 25.0),
74            child: Text(
75              text,
76              style: TextStyle(
77                fontWeight: FontWeight.bold,
78              ),
79            ),
80          ),
81        ],
82      ),
83      onTap: onTap,
84    );
85  }
```

(5) Sekarang kita beralih ke **main.dart** dan import nav-drawer menjadi seperti ini

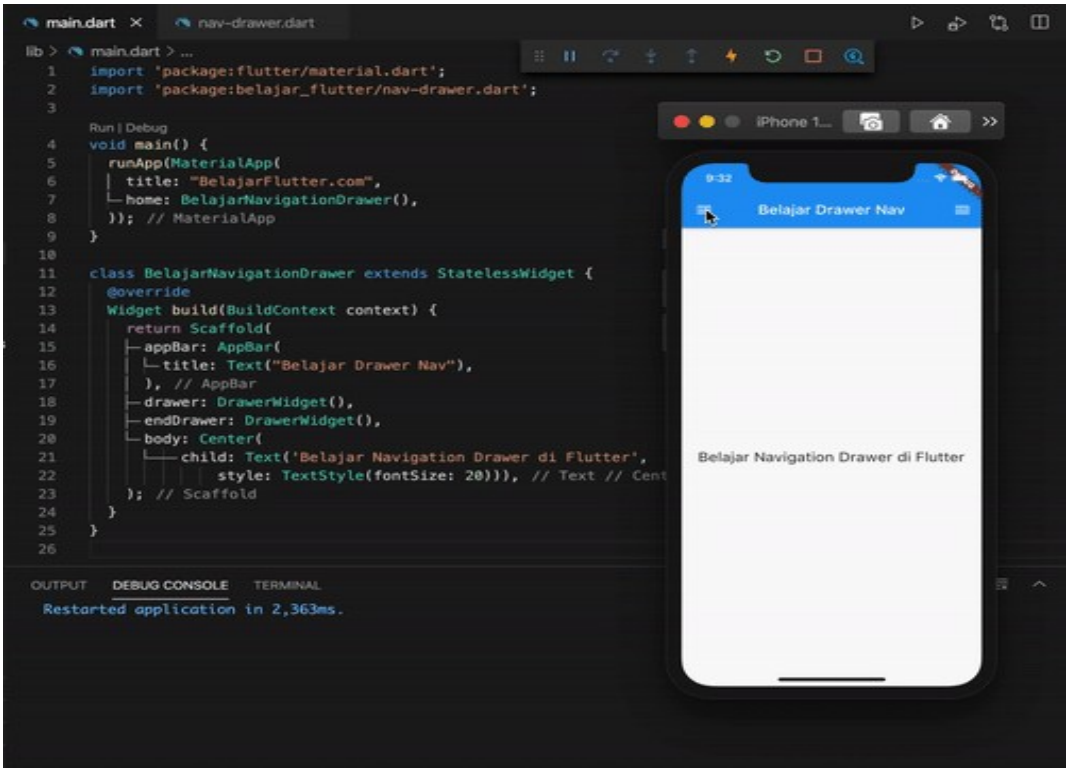
```
1  import 'package:flutter/material.dart';
2  import 'package:belajar_flutter/nav-drawer.dart';
3
```

```dart
void main() {
  runApp(MaterialApp(
    title: "BelajarFlutter.com",
    home: BelajarNavigationDrawer(),
  ));
}

class BelajarNavigationDrawer extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Belajar Drawer Nav"),
      ),
      drawer: DrawerWidget(),
      endDrawer: DrawerWidget(),
      body: Center(
          child: Text('Belajar Navigation Drawer di Flutter',
              style: TextStyle(fontSize: 20))),
    );
  }
}
```

Perhatikan pada baris kedua, import 'package:**belajar_flutter**/nav-drawer.dart'; tulisan belajar_flutter merupakan nama projek yang kami buat pada tahap pertama, apabila nama projek anda berbeda maka harap sesuaikan dengan nama projek anda masing-masing.

(6) Selesai, silahkan coba jalankan atau reload flutter anda

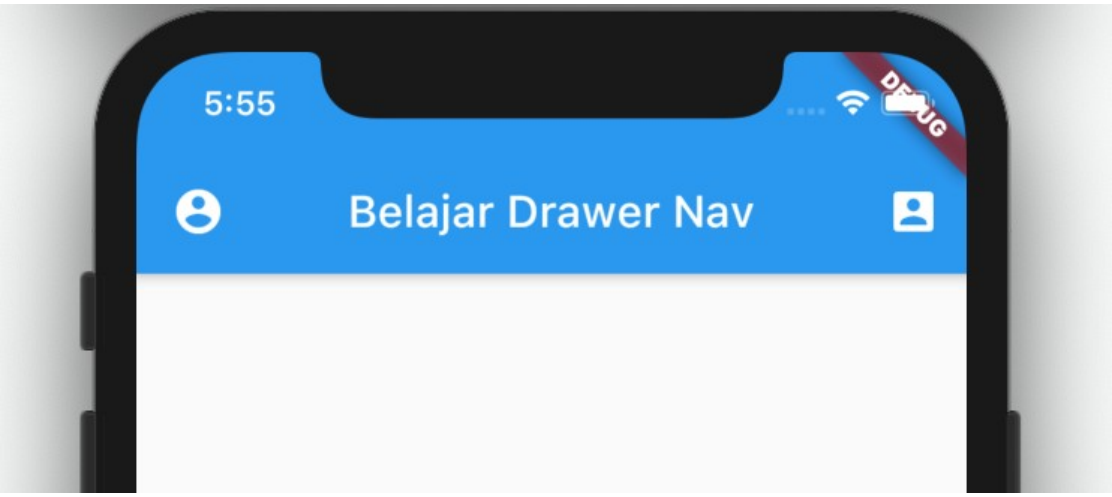


5. Mengganti Icon Drawer

Untuk mengganti default icon, kita bisa menggunakan properti **leading** atau **actions** properti pada appBar. Untuk memanggil drawer menggunakan leading kita perlu menambahkan **key** pada Scaffold. contoh kode nya seperti ini …

```
class Belajar extends StatelessWidget {
  final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        key: _scaffoldKey,
        appBar: AppBar(
            leading: new IconButton(
          icon: new Icon(Icons.account_circle),
          onPressed: () => _scaffoldKey.currentState.openDrawer(),
        )));
  }
}
```

Fungsi openDrawer pada kode () => **_scaffoldKey.currentState.openDrawer** ini bertujuan untuk memanggil drawer. Sedangkan apabila ingin menampilkan endDrawer bisa dengan fungsi **.openEndDrawer()**



Hasil akhir **main.dart** menjadi seperti ini

```
import 'package:flutter/material.dart';
import 'package:belajar_flutter/nav-drawer.dart';

void main() {
  runApp(MaterialApp(
    title: "BelajarFlutter.com",
    home: BelajarNavigationDrawer(),
  ));
}
```

```
10
11 class BelajarNavigationDrawer extends StatefulWidget {
12   @override
13   _BelajarNavigationDrawerState createState() =>
14       _BelajarNavigationDrawerState();
15 }
16
17 class _BelajarNavigationDrawerState extends State<BelajarNavigationDrawer> {
18   final GlobalKey<ScaffoldState> _scaffoldKey = new GlobalKey<ScaffoldState>();
19   @override
20   Widget build(BuildContext context) {
21     return Scaffold(
22       key: _scaffoldKey,
23       appBar: AppBar(
24         leading: new IconButton(
25           icon: new Icon(Icons.account_circle),
26           onPressed: () => _scaffoldKey.currentState.openDrawer(),
27         ),
28         title: Text("Belajar Drawer Nav"),
29         actions: <Widget>[
30           IconButton(
31               icon: new Icon(Icons.account_box, color: Colors.white),
32               onPressed: () => _scaffoldKey.currentState.openEndDrawer()),
33         ],
34       ),
35       drawer: DrawerWidget(),
36       endDrawer: DrawerWidget(),
37       body: Center(
38         child: Text(
39           'Belajar Navigation Drawer di Flutter',
40           style: TextStyle(fontSize: 20),
41         ),
42       ),
43     );
44   }
45 }
```

Penutup

Tutorial diatas merupakan dasar cara membuat navigation drawer di flutter. Masih banyak yang dapat kita explore seperti membuat kostum drawer header dengan DrawerHeader(), atau routing navigasi, dan masih banyak lagi. Kita akan coba bahas pada kesempatan selanjutnya.

C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat membuat Drawer pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang cara pembuatan drawer pada flutter

Tugas	-
Refrensi	https://flutter.dev/docs/cookbook/design/drawer https://material.io/components/navigation-drawer/

BAB 6
LAYOUT

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Tabbar pada flutter. Tab Bar merupakan suatu Widget yang difungsikun sebgai kontroller atau navigation.

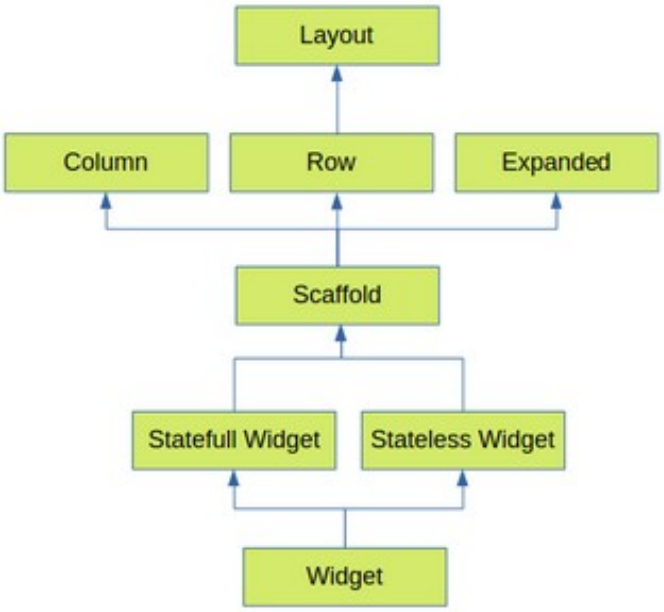
2 Capaian Pembelajaran

Mahasiswa mampu membuat desain layout pada android yang user friendly menggunakan

3 Bahan Kajian

Layout, App Bar & Tab Bar Widget, Custom Widget, Input, Selection

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 11. App Bar dan Tab Bar Widget

TabBar merupakan komponen penting, biasanya berisi menu yang memudahkan kita untuk berpindah halaman dengan cara melakukan slide kekanan ataupun kekiri pada layar hp. Tabs sendiri dapat kita tempatkan pada top position ataupun bottom possition dan terkadang ada juga yang menggunakan keduanya sekaligus.

1. Membuat Tabs Widget

Pertama kita import ‘package: flutter/material.dart ’ dan buat fungsi main() terlebih dahulu. Kita buat juga widget home.

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(new MaterialApp(
5     title: "Tab Bar",
6     home: new Home(),
7   ));
8 }
```

Kemudian dibawah main() kita buat kelas Home. Disini kelas Home menggunakan statefulWidget yang bersifat dinamis sehingga widget dapat dirender kembali ketika terdapat event atau user action.

Didalam kelas **_HomeState** kita buat nilai kembalian/ return scaffold atau di android studio sama dengan activity. Di dalam scaffold tambahkan appbar. Pada appbar kita gunakan background dengan warna red, dan buat title dengan “Catalogue Movie”, dan bottom dengan widget **TabBar** untuk tab yang akan kita tampilkan.

```
 1 class Home extends StatefulWidget {
 2   @override
 3   _HomeState createState() => _HomeState();
 4 }
 5
 6 class _HomeState extends State<Home> with SingleTickerProviderStateMixin {
 7   @override
 8   Widget build(BuildContext context) {
 9     return new Scaffold(
10       appBar: new AppBar(
11         backgroundColor: Colors.red,
12         title: new Text("Catalogue Movie"),
13         bottom: new TabBar(),
14       ),
15     );
16   }
17 }
```

Pada Appbar kita akan buat ada 4 tab, jadi kita harus buat controler terlebih dahulu. TabController berfungsi untuk mengkoordinasikan antara Tabbar dan TabbarView. Tambahkan kode berikut didalam kelas_HomeState TabController controller;

```
1 @override
2 void initState() {
3 controller = new TabController(vsync: this, length: 4);
```

```
 4 super.initState();
 5 }
 6
 7 @override
 8 void dispose() {
 9 controller.dispose();
10 super.dispose();
11 }
```

Pada Appbar bagian bottom kita masukan controller yang telah dibuat tadi.

```
1 bottom: new TabBar(
2 controller: controller,
3 tabs: <Widget>[
4 new Tab()
5 ],
6 ),
```

Nah pada bagian ini sebelum mengisi tab, kita buat 4 halaman untuk tab nya. pilih folder lib> klik icon new file > isikan nama file dengan extention .dart contoh hal_home.dart. Pada hal_home.dart tambahkan kode berikut. buat juga halaman lain nowplaying.dart, upcoming.dartm dan favorite.dart dan copy paste kode tadi kemudian sesuaikan nama Kelas, Text dan Icon nya.

```
 1 class Home extends StatelessWidget {
 2   @override
 3   Widget build(BuildContext contect) {
 4     return new Container(
 5       child: new Center(
 6         child: new Column(
 7           children: <Widget>[
 8             new Padding(
 9               padding: new EdgeInsets.all(20.0),
10             ),
11             new Text(
12               "HOME",
13               style: new TextStyle(fontSize: 30.0),
14             ),
15             new Padding(
16               padding: new EdgeInsets.all(20.0),
17             ),
18             new Icon(
19               Icons.home,
20               size: 90.0,
21             )
22           ],
23         ),
24       ),
25     );
26   }
27 }
```

Kemudian pada main.dart pada Appbar Widget bagian bottom, isi tabs dengan Widget Tab.

```
1 bottom: new TabBar(
2 controller: controller,
3 tabs: <Widget>[
4 new Tab(icon: new Icon(Icons.home), text: "Home",),
5 new Tab(icon: new Icon(Icons.play_arrow), text: "NowPlaying"),
6 new Tab(icon: new Icon(Icons.movie), text: "UpComing"),
7 new Tab(icon: new Icon(Icons.star), text: "Favorite")
8 ],
9 ),
```

Kemudian Run aplikasi dengan menekan tombol F5 atau save file. Maka tampilan akan seperti ini



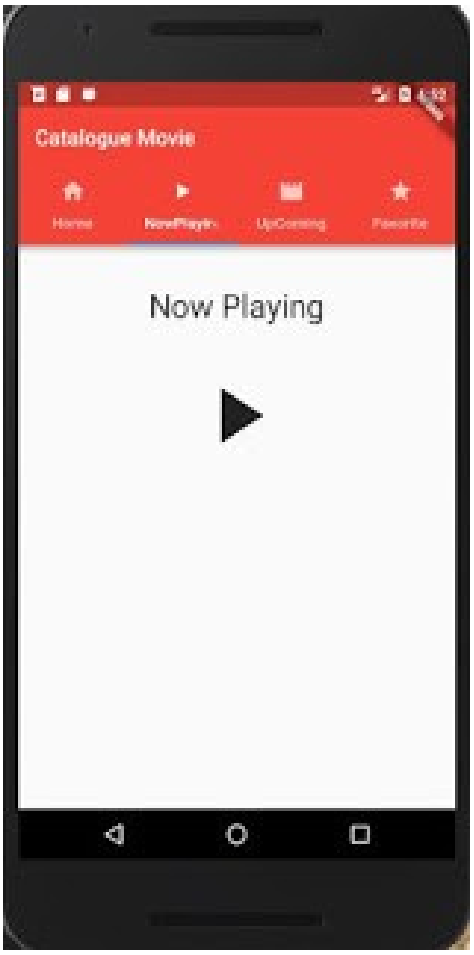
Sekarang kita akan menambah halaman pada body setiap tabnya menggunakan TabView. Untuk dapat memanggil halaman yang telah kita buat tadi, kita import dulu halaman pada main dart.

```
1 import './hal_home.dart' as home;
2 import './hal_nowplaying.dart' as nowplaying;
3 import './hal_upcoming.dart' as upcoming;
4 import './hal_favorite.dart' as favorite;
```

Kemudian buat body didalam Scaffold. Pada children kita panggil hamalan sesuai dengan tabs Widgetnya. Bagian body akan menampilkan isi halaman.

```
1 body: new TabBarView (
2 controller: controller,
3 children: <Widget>[
4 new home.Home(),
5 new nowplaying.NowPlaying(),
6 new upcoming.Upcoming(),
7 new favorite.Favorite(),
8 ],
9 ),
```

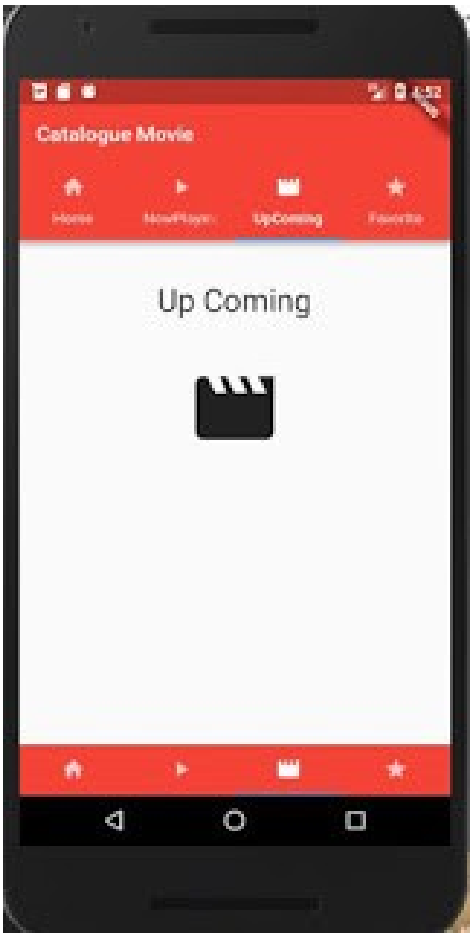
Kemudian kisa save maka hasilnya akan seperti berikut :



Kita juga bisa membuat tabbar dibagian bawah menggunakan bottomNavigationBar didalam scaffold.

```
bottomNavigationBar: new Material(
color: Colors.red,
child: new TabBar(
controller: controller,
tabs: <Widget>[
new Tab(icon: new Icon(Icons.home)),
new Tab(icon: new Icon(Icons.play_arrow)),
new Tab(icon: new Icon(Icons.movie)),
new Tab(icon: new Icon(Icons.star)),
],
),
)
```

Ketika di run, maka hasilnya akan jadi seperti ini



Kita dapat berpindah halaman dengan menggeser layar hp atau melakukan perpindahan halaman slide kekanan dan ke kiri dengan animasi yang telah tersedia pada Flutter.

Berikut Source Code Lengkap pada file main.dart

```
import 'package:flutter/material.dart';
import './hal_home.dart' as home;
```

```
 3 import './hal_nowplaying.dart' as nowplaying;
 4 import './hal_upcoming.dart' as upcoming;
 5 import './hal_favorite.dart' as favorite;
 6
 7 void main() {
 8   runApp(new MaterialApp(
 9     title: "Tab Bar",
10     home: new Home(),
11   ));
12 }
13
14 class Home extends StatefulWidget {
15   @override
16   _HomeState createState() => _HomeState();
17 }
18
19 class _HomeState extends State<Home> with SingleTickerProviderStateMixin {
20   TabController controller;
21
22   @override
23   void initState() {
24     controller = new TabController(vsync: this, length: 4);
25     super.initState();
26   }
27
28   @override
29   void dispose() {
30     controller.dispose();
31     super.dispose();
32   }
33
34   @override
35   Widget build(BuildContext context) {
36     return new Scaffold(
37         appBar: new AppBar(
38           backgroundColor: Colors.red,
39           title: new Text("Catalogue Movie"),
40           bottom: new TabBar(
41             controller: controller,
42             tabs: <Widget>[
43               new Tab(
44                 icon: new Icon(Icons.home),
45                 text: "Home",
46               ),
47               new Tab(icon: new Icon(Icons.play_arrow), text: "NowPlaying"),
48               new Tab(icon: new Icon(Icons.movie), text: "UpComing"),
49               new Tab(icon: new Icon(Icons.star), text: "Favorite")
50             ],
51           ),
52         ),
53         body: new TabBarView(
54           controller: controller,
55           children: <Widget>[
56             new home.Home(),
57             new nowplaying.NowPlaying(),
```

```
58              new upcoming.Upcoming(),
59              new favorite.Favorite(),
60            ],
61          ),
62          bottomNavigationBar: new Material(
63            color: Colors.red,
64            child: new TabBar(
65              controller: controller,
66              tabs: <Widget>[
67                new Tab(icon: new Icon(Icons.home)),
68                new Tab(icon: new Icon(Icons.play_arrow)),
69                new Tab(icon: new Icon(Icons.movie)),
70                new Tab(icon: new Icon(Icons.star)),
71              ],
72            ),
73          ));
74    }
75 }
```

C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat membuat Tab Bar pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang cara pembuatan Tab Bar pada flutter
Tugas	-
Refrensi	https://www.codepolitan.com/membuat-tab-bar-di-flutter https://flutter.dev/docs/cookbook/design/tabs

BAB 6
LAYOUT

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang notification dan alert pada flutter. Notification dan Alert merupakan suatu bentuk widget yang digunakan untuk memberikan warning / pemberitahuan.

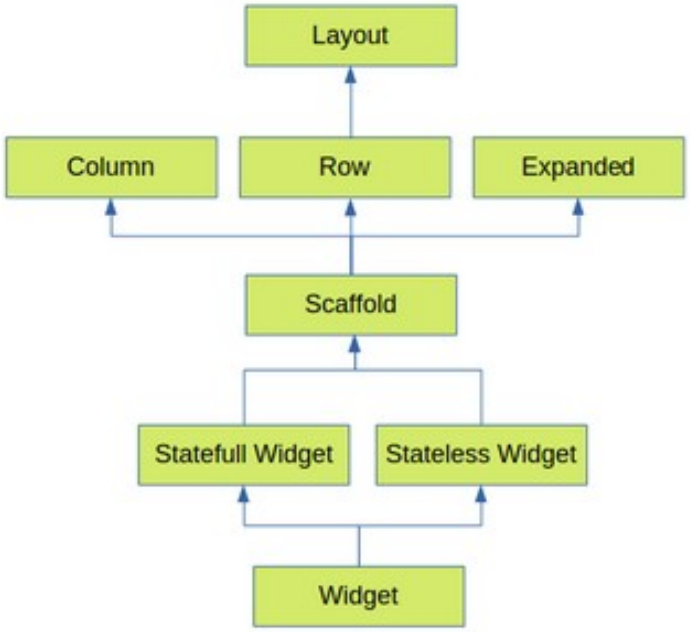
2 Capaian Pembelajaran

Mahasiswa mampu membuat desain layout pada android yang user friendly menggunakan

3 Bahan Kajian

Layout, App Bar & Tab Bar Widget, Custom Widget, Input, Selection

4 Peta Kompetensi Modul

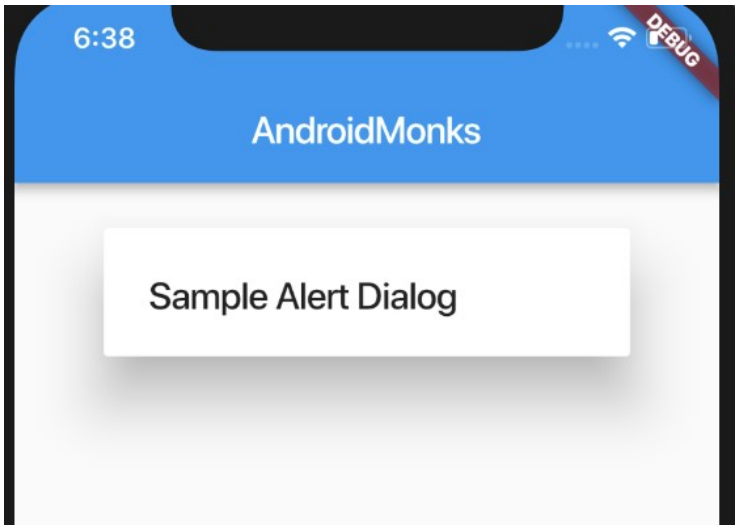


B. Kegiatan Belajar

Pertemuan 12. Notification dan Alert

SimpleDialog adalah salah satu jenis dari function **showDialog** yaitu **AlertDialog**. Perbedaannya hanya ada pada action bar nya dimana SimpleDialog tidak memiliki

action button. Jadi bisa dibilang SimpleDialog hanya kotak popup saja yang bisa kita isi dengan konten dan tombol. SimpleDialog biasanya digunakan untuk membuat popup yang berisi form atau opsi-opsi lainnya.



Contoh bentuknya sebagai berikut:

```
void _showSimpleDialog(context) {
    showDialog(
      context: context,
      builder: (context) {
        return SimpleDialog(
          children: <Widget>[
 //widget di sini//
          ],
        );
      }
    )
}
```

SimpleDialog memiliki children yang dapat diisi dengan lebih dari satu widget. jadi kita bebas membuat layout dalam SimpleDialog. Berikut contoh yang kita buat (SimpleDialog yang berisi form).

```
void _showSimpleDialog(context) {
    showDialog(
      context: context,
      builder: (context) {
        return SimpleDialog(
          children: <Widget>[
            Padding(
```

```
 8                padding: EdgeInsets.only(left: 10.0, right: 10.0),
 9                child: Row(
10                  children: <Widget>[
11                    Expanded(
12                      child: Text(
13                        "Tambah Pengguna",
14                        style: TextStyle(fontSize: 20.0),
15                      ),
16                    ),
17                    IconButton(
18                      icon: Icon(Icons.close),
19                      onPressed: () {
20                        Navigator.pop(context);
21                      },
22                    )
23                  ],
24                ),
25              ),
26              Divider(),
27              Padding(
28                padding: EdgeInsets.all(10.0),
29                child: Text(
30                  "Nama Lengkap",
31                ),
32              ),
33              Padding(
34                padding: EdgeInsets.only(left: 10.0, right: 10.0),
35                child: TextField(
36                  style: new TextStyle(
37                      fontSize: 14.0, height: 1.0, color: Colors.black),
38                  decoration: new InputDecoration(
39                    hintText: "Input nama lengkap",
40                    border: OutlineInputBorder(
41                        borderRadius: new BorderRadius.circular(3.0)),
42                    contentPadding: const EdgeInsets.symmetric(
43                        vertical: 14.0, horizontal: 15.0),
44                  ),
45                ),
46              ),
47              Padding(
48                padding: EdgeInsets.all(10.0),
49                child: Text(
50                  "Kontak",
51                ),
52              ),
53              Padding(
54                padding: EdgeInsets.only(left: 10.0, right: 10.0),
55                child: TextField(
56                  style: new TextStyle(
57                      fontSize: 14.0, height: 1.0, color: Colors.black),
58                  decoration: new InputDecoration(
59                    hintText: "Input kontak",
60                    border: OutlineInputBorder(
61                        borderRadius: new BorderRadius.circular(3.0)),
62                    contentPadding: const EdgeInsets.symmetric(
```

```
                      vertical: 14.0, horizontal: 15.0),
                ),
              ),
            ),

            Padding(
              padding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 0),
              child: RaisedButton(
                color: Colors.blue,
                child: Text(
                  "SIMPAN",
                  style: TextStyle(color: Colors.white),
                ),
                onPressed: () {

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

Kemudian tinggal kita panggil dengan action dari button sebagai berikut.

```
import 'package:flutter/material.dart';
class Homepage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Show Dialog"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: (){
            _showSimpleDialog(context);
          },
          child: Text("Click Here")
        )
      )
    );
  }
}
```

Full Code

```
import 'package:flutter/material.dart';
void main() {
```

```
 3   runApp(new MaterialApp(
 4     title: "Halaman",
 5     home: new Halaman(),
 6   ));
 7 }
 8 class Halaman extends StatelessWidget {
 9   @override
10   Widget build(BuildContext context) {
11     return Scaffold(
12       appBar: AppBar(
13         title: Text("Show Dialog"),
14       ),
15       body: Center(
16         child: RaisedButton(
17           onPressed: (){
18             _showSimpleDialog(context);
19           },
20           child: Text("Click Here")
21         )
22       )
23     );
24   }
25   void _showSimpleDialog(context) {
26     showDialog(
27       context: context,
28       builder: (context) {
29         return SimpleDialog(
30           children: <Widget>[
31             Padding(
32               padding: EdgeInsets.only(left: 10.0, right: 10.0),
33               child: Row(
34                 children: <Widget>[
35                   Expanded(
36                     child: Text(
37                       "Tambah Pengguna",
38                       style: TextStyle(fontSize: 20.0),
39                     ),
40                   ),
41                   IconButton(
42                     icon: Icon(Icons.close),
43                     onPressed: () {
44                       Navigator.pop(context);
45                     },
46                   )
47                 ],
48               ),
49             ),
50             Divider(),
51             Padding(
52               padding: EdgeInsets.all(10.0),
53               child: Text(
54                 "Nama Lengkap",
55               ),
56             ),
57             Padding(
```

```
58                padding: EdgeInsets.only(left: 10.0, right: 10.0),
59                child: TextField(
60                  style: new TextStyle(
61                      fontSize: 14.0, height: 1.0, color: Colors.black),
62                  decoration: new InputDecoration(
63                    hintText: "Input nama lengkap",
64                    border: OutlineInputBorder(
65                        borderRadius: new BorderRadius.circular(3.0)),
66                    contentPadding: const EdgeInsets.symmetric(
67                        vertical: 14.0, horizontal: 15.0),
68                  ),
69                ),
70              ),
71              Padding(
72                padding: EdgeInsets.all(10.0),
73                child: Text(
74                  "Kontak",
75                ),
76              ),
77              Padding(
78                padding: EdgeInsets.only(left: 10.0, right: 10.0),
79                child: TextField(
80                  style: new TextStyle(
81                      fontSize: 14.0, height: 1.0, color: Colors.black),
82                  decoration: new InputDecoration(
83                    hintText: "Input kontak",
84                    border: OutlineInputBorder(
85                        borderRadius: new BorderRadius.circular(3.0)),
86                    contentPadding: const EdgeInsets.symmetric(
87                        vertical: 14.0, horizontal: 15.0),
88                  ),
89                ),
90              ),
91
92              Padding(
93                padding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 0),
94                child: RaisedButton(
95                  color: Colors.blue,
96                  child: Text(
97                    "SIMPAN",
98                    style: TextStyle(color: Colors.white),
99                  ),
100                 onPressed: () {
101
102                 },
103               ),
104             )
105           ],
106         );
107       },
108     );
109   }
110 }
```

C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat membuat Notification/Alert pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang cara pembuatan alert pada flutter
Tugas	-
Refrensi	https://flutter.dev/

BAB 7
JAVASCRIPT OBJECT NOTATION (JSON)

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Drawer Routes dan JSON App Overview pada flutter. JSON dalah sebuah format data yang digunakan untuk pertukaran dan penyimpanan data

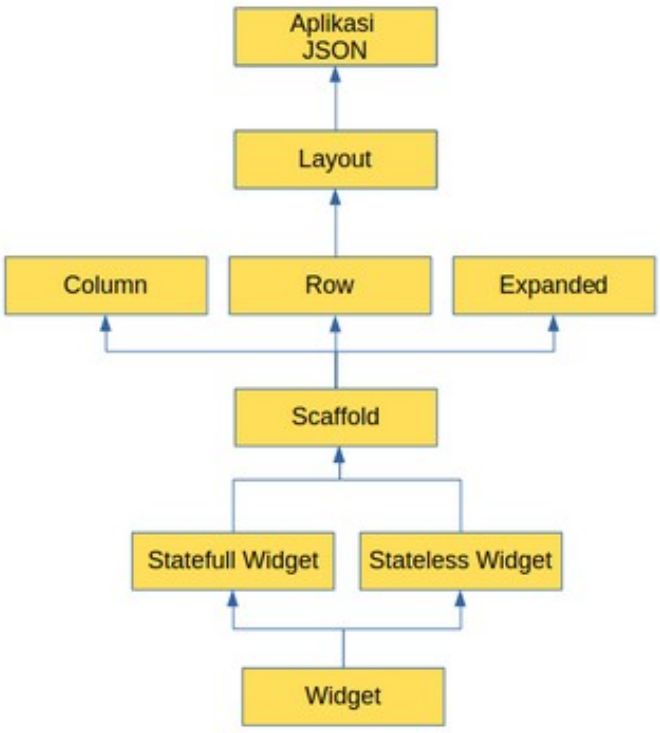
2 Capaian Pembelajaran

Mahasiswa mampu menggunakan Drawer Widget & Routes dan JSON App Overview

3 Bahan Kajian

Drawer widget & routes, notification widgets, JSON app overview, create android apk

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 12. Pengenalan JSON

Pada kali ini kita mencoba membahas mengenai bagaimana cara sederhana dalam menampilkan data dari JSON ke dalam aplikasi berbasis Flutter. Seperti yang kita ketahui saat ini aplikasi-aplikasi sudah dapat menampilkan data secara real-time dari server/website atau platform lainnya. Hal tersebut salah satunya karena adanya API yang menghubungkan data ke masing-masing platform tersebut. Pada contoh kali ini kita akan menggunakan sampel data json dari **https://jsonplaceholder.typicode.com/users/.** Jika dilihat struktur datanya kira-kira seperti di bawah ini. Total data ada 10.


```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
        "lng": "-34.4618"
```

```
      }
    },
    "phone": "010-692-6593 x09125",
    "website": "anastasia.net",
    "company": {
      "name": "Deckow-Crist",
      "catchPhrase": "Proactive didactic contingency",
      "bs": "synergize scalable supply-chains"
    }
  },
```

Tahap Pertama kita akan membuat file halamanJson.dart dalam project kita. Kemudian isi dengan kode di bawah ini.

```
import 'package:flutter/material.dart';
void main() {
  runApp(new MaterialApp(
    title: "My Apps",
    home: new HalamanJson(),
  ));
}
class HalamanJson extends StatefulWidget {
  @override
  _HalamanJsonState createState() => _HalamanJsonState();
}
class _HalamanJsonState extends State {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Data JSON"),
      ),
      drawer: DrawerApp(),
      body: Center(
          child: Text("Data JSON")
      ),
    );
  }
}
```

Selanjutnya kita membutuhkan beberapa package diantaranya sebagai berikut. Tambahkan http dependencies dalam file **pubspecs.yaml**

```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.12.0+1
```

Lalu import dalam **file .dart**

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

```
import 'dart:async';
Selanjutnya kita akan menggunakan Future untuk menjalankan http.get.
List datadariJSON;
Future ambildata() async {
    http.Response hasil = await http.get(
        Uri.encodeFull("https://jsonplaceholder.typicode.com/users"),
        headers: {"Accept": "application/json"});
    this.setState(() {
      datadariJSON = json.decode(hasil.body);
    });
}
```

Pada contoh **Future** di atas, sebelumnya kita telah membuat List terlebih dahulu yang bernama **datadariJSON** yang dimaksudkan akan menampung data yang akan diparsing dari url. Jadi sebelum Future **ambildata** dijalankan maka List **datadariJSON** masih bernilai **null**. setState di sini berfungsi untuk merubah state dari **datadariJSON** yang tadinya null menjadi berisi data dari hasil parsing. Lalu untuk menjalankan Future ambil data kita menggunakan **initState.**

```
@override
void initState() {
    this.ambildata();
}
```

Singkatnya Full Code-nya akan menjadi seperti di bawah ini. Di sini kita menampilkan List **datadariJSON** menggunakan **ListView.builder** dimana setiap ListTile nya dapat diubah secara custom sesuai keinginan masing-masing. Untuk melihat contoh Listview custom **di sini.**

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:async';
void main() {
  runApp(new MaterialApp(
    title: "My Apps",
    home: new HalamanJson(),
  ));
}
class HalamanJson extends StatefulWidget {
  @override
  _HalamanJsonState createState() => _HalamanJsonState();
}
class _HalamanJsonState extends State {
  List datadariJSON;
  Future ambildata() async {
    http.Response hasil = await http.get(
        Uri.encodeFull("https://jsonplaceholder.typicode.com/users"),
        headers: {"Accept": "application/json"});
    this.setState(() {
      datadariJSON = json.decode(hasil.body);
```

```
    });
  }
  @override
  void initState() {
    this.ambildata();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Data JSON"),
      ),
      body: Container(
        child: ListView.builder(
          itemCount: datadariJSON == null ? 0 : datadariJSON.length,
          itemBuilder: (context, i){
            return ListTile(
              title: Text(datadariJSON[i]['name']),
            );
          }
        ),
      ),
    );
  }
```

Saat menampilkan data dari List kita seperti menampilkan array pada umumnya dan disesuaikan dengan struktur dari JSON yang tersedia. Contohnya di sini menampilkan **name** dengan cara **datadariJSON[i]['name'].** Contoh lain jika ingin menampilkan nama jalan maka kita gunakan cara **datadariJSON[i]['address']['street'].**

C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat memahami struktru dari file JSON pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang file JSON dan cara penggunaannya pada flutter
Tugas	-
Refrensi	https://flutter.dev/

BAB 7
JAVASCRIPT OBJECT NOTATION (JSON)

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Drawer Routes dan JSON App Overview pada flutter. JSON dalah sebuah format data yang digunakan untuk pertukaran dan penyimpanan data

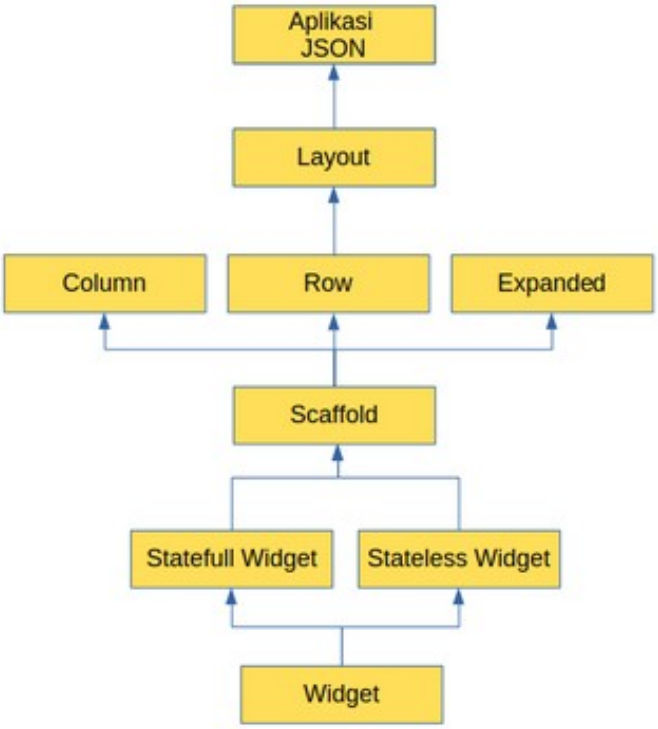
2 Capaian Pembelajaran

Mahasiswa mampu menggunakan Drawer Widget & Routes dan JSON App Overview

3 Bahan Kajian

Drawer widget & routes, notification widgets, JSON app overview, create android apk

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 14. Implementasi JSON

Pada materi kali ini, kita akan melanjutkan pembahasan tentang JSON, yaitu bagaimana cara menggunakannya dan implementasinya pada flutter.

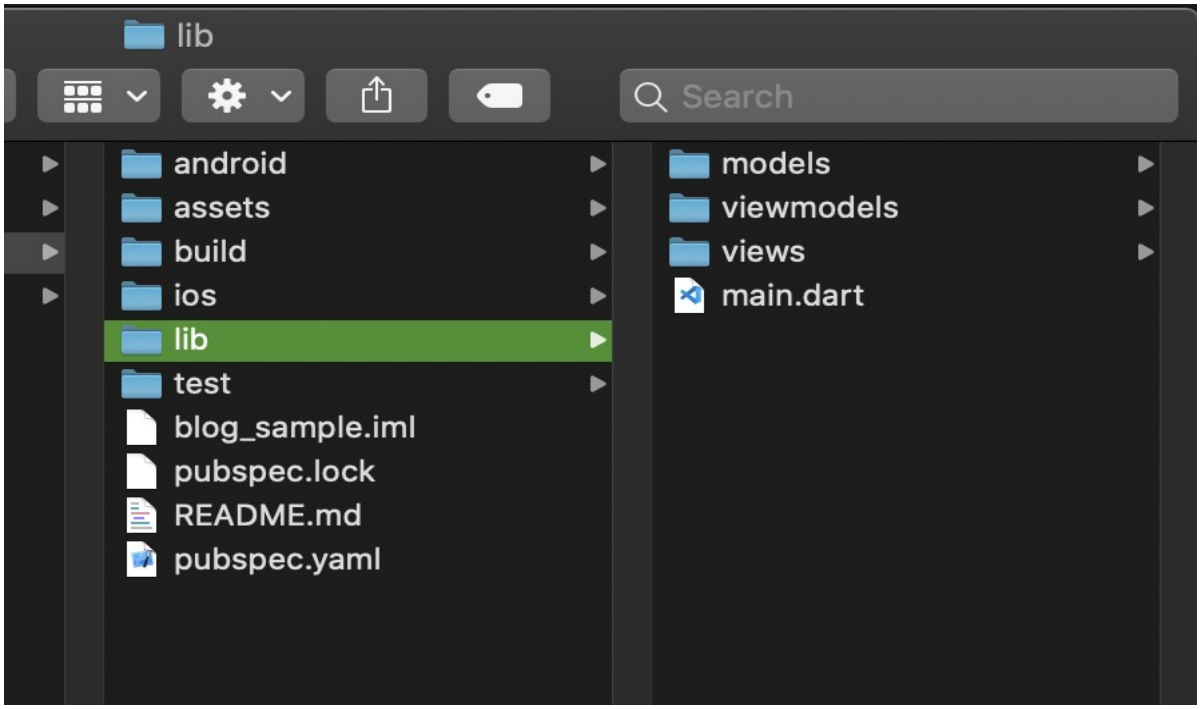
1. Model
Model / entity yang biasanya menggambarkan bentuk dari data yang digunakan atau ada yang bilang dengan istilah **PODO (Plain Old Dart Objects).**

2. View
Merepresentasikan UI/Layout dari sebuah aplikasi

3. ViewModel

Bagian ini berfungsi untuk melakukan pemrosesan data **(POST/GET dll)** dihubungkan dengan Model dan mengembalikan data kepada View.

Mungkin sekilas saja kira-kira seperti itu. Kalau ada penjelasan yang lebih bagus mungkin bisa dicari dari website tutorial lain. Oke langsung saja kita buat struktur foldernya seperti berikut.



Selanjutnya kita isi file **main.dart** dengan code berikut.

```
import 'views/halamanutama.dart';
import 'package:flutter/material.dart';
void main() {
  runApp(new MaterialApp(
    title: "My App",
    home: new MyApp(),
  ));
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MY App',
      home: HalamanUtama()
    );
  }
}
```

Kemudian kita buat halaman utamanya (**HalamanUtama**) pada folder views seperti berikut.

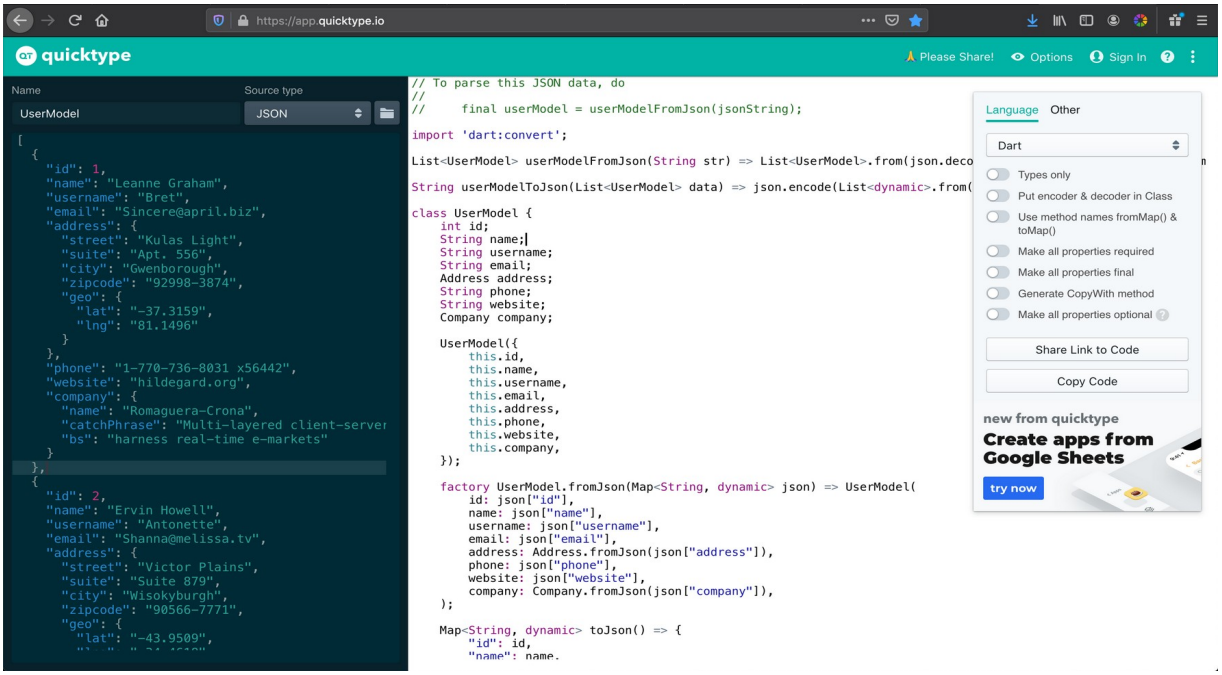
```
import 'package:flutter/material.dart';
class HalamanUtama extends StatefulWidget {
  @override
  _HalamanUtamaState createState() => _HalamanUtamaState();
}
class _HalamanUtamaState extends State<HalamanUtama> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Halaman Utama"),
      ),
      body: Text("data"),
    );
  }
}
```

Selanjutnya kita siapkan API dan Modelnya. Untuk API kita pakai contoh dari **https://jsonplaceholder.typicode.com/users.**

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
```

```
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
        "lng": "-34.4618"
      }
    },
    "phone": "010-692-6593 x09125",
    "website": "anastasia.net",
    "company": {
      "name": "Deckow-Crist",
      "catchPhrase": "Proactive didactic contingency",
      "bs": "synergize scalable supply-chains"
    }
  },]
```

Untuk membuat model biasanya kita pakai yang instan menggunakan https://app.quicktype.io/. Caranya tinggal kita paste saja json kita ke form yang sudah disediakan dan kita pilih bahasa pemrograma **dart.**



Kemudian tinggal kita copy paste saja hasilnya ke dalam file model kita yaitu di sini kita beri nama **usermodel.dart**.

```
// To parse this JSON data, do
//
//     final userModel = userModelFromJson(jsonString);
import 'dart:convert';
List<UserModel> userModelFromJson(String str) =>
List<UserModel>.from(json.decode(str).map((x) => UserModel.fromJson(x)));
String userModelToJson(List<UserModel> data) => json.encode(
List<dynamic>.from(data.map((x) => x.toJson())));
class UserModel {
    int id;
    String name;
    String username;
    String email;
    Address address;
    String phone;
    String website;
    Company company;
    UserModel({
        this.id,
        this.name,
        this.username,
        this.email,
        this.address,
        this.phone,
        this.website,
        this.company,
    });
    factory UserModel.fromJson(Map<String, dynamic> json) => UserModel(
        id: json["id"],
```

```
            name: json["name"],
            username: json["username"],
            email: json["email"],
            address: Address.fromJson(json["address"]),
            phone: json["phone"],
            website: json["website"],
            company: Company.fromJson(json["company"]),
        );
        Map<String, dynamic> toJson() => {
            "id": id,
            "name": name,
            "username": username,
            "email": email,
            "address": address.toJson(),
            "phone": phone,
            "website": website,
            "company": company.toJson(),
        };
}
class Address {
    String street;
    String suite;
    String city;
    String zipcode;
    Geo geo;
    Address({
        this.street,
        this.suite,
        this.city,
        this.zipcode,
        this.geo,
    });
    factory Address.fromJson(Map<String, dynamic> json) => Address(
        street: json["street"],
        suite: json["suite"],
        city: json["city"],
        zipcode: json["zipcode"],
        geo: Geo.fromJson(json["geo"]),
    );
    Map<String, dynamic> toJson() => {
        "street": street,
        "suite": suite,
        "city": city,
        "zipcode": zipcode,
        "geo": geo.toJson(),
    };
}
class Geo {
    String lat;
    String lng;
    Geo({
        this.lat,
        this.lng,
    });
    factory Geo.fromJson(Map<String, dynamic> json) => Geo(
```

```
        lat: json["lat"],
        lng: json["lng"],
    );
    Map<String, dynamic> toJson() => {
        "lat": lat,
        "lng": lng,
    };
}
class Company {
    String name;
    String catchPhrase;
    String bs;
    Company({
        this.name,
        this.catchPhrase,
        this.bs,
    });
    factory Company.fromJson(Map<String, dynamic> json) => Company(
        name: json["name"],
        catchPhrase: json["catchPhrase"],
        bs: json["bs"],
    );
    Map<String, dynamic> toJson() => {
        "name": name,
        "catchPhrase": catchPhrase,
        "bs": bs,
    };
}
```

Nah sekarang kita harus membuat fungsi untuk memanggil api tersebut yaitu dengan Future dengan nama file **userviewmodel.dart**. Contohnya seperti berikut.

```
 1 import 'dart:convert';
 2 import 'package:blog_sample/models/usermodel.dart';
 3 import 'package:http/http.dart' as http;
 4 class UserViewModel {
 5   Future<List<UserModel>> getUsers() async {
 6     try {
 7       http.Response hasil = await http.get(
 8           Uri.encodeFull("https://jsonplaceholder.typicode.com/users"),
 9           headers: {"Accept": "application/json"});
10       if (hasil.statusCode == 200) {
11         print("data category success");
12         final data = userModelFromJson(hasil.body);
13         return data;
14       } else {
15         print("error status " + hasil.statusCode.toString());
16         return null;
17       }
18     } catch (e) {
19       print("error catch $e");
20       return null;
21     }
22   }
```

```
23 }
```

Karena kita pakai package http maka jangan lupa menambahkan dependency pada pubspec.yaml sepert berikut.

```
1 dependencies:
2   http: ^0.12.1
```

Langkah terakhir tinggal kita panggil Future dari UserViewModel untuk ditampilkan ke dalam view dengan cara menambahkan kode berikut pada file **halamanutama.dart.**

```
 1 import 'package:blog_sample/models/usermodel.dart';
 2 import 'package:blog_sample/viewmodels/userviewmodel.dart';
 3 import 'package:flutter/material.dart';
 4 class HalamanUtama extends StatefulWidget {
 5   @override
 6   _HalamanUtamaState createState() => _HalamanUtamaState();
 7 }
 8 class _HalamanUtamaState extends State<HalamanUtama> {
 9   List<UserModel> dataUser = new List();
10   void getDataUser() {
11     UserViewModel().getUsers().then((value) {
12       setState(() {
13         dataUser = value;
14       });
15     });
16   }
17   @override
18   void initState() {
19     // TODO: implement initState
20     super.initState();
21     getDataUser();
22   }
23   @override
24   Widget build(BuildContext context) {
25     return Scaffold(
26       appBar: AppBar(
27         title: Text("Halaman Utama"),
28       ),
29       body: dataUser == null ? Center(
30 child: CircularProgressIndicator(),) : ListView.builder(
31         itemCount: dataUser.length,
32         itemBuilder: (context, i) {
33           return ListTile(
34             title: Text(dataUser[i].name),
35           );
36         },
37       ),
38     );
39   }
40 }
```

C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat memahami cara implementasi JSON pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang file implementasi JSON dan cara penggunaannya pada flutter
Tugas	-
Refrensi	https://www.byriza.com/flutter-28-parsing-json-pada-flutter https://medium.com/@randy.arba/load-file-json-dalam-flutter-9c721f6a5000

BAB 7
JAVASCRIPT OBJECT NOTATION (JSON)

A. Pendahuluan

1 Deskripsi Singkat

Pada materi ini akan dijelaskan tentang Drawer Routes dan JSON App Overview pada flutter. JSON dalah sebuah format data yang digunakan untuk pertukaran dan penyimpanan data

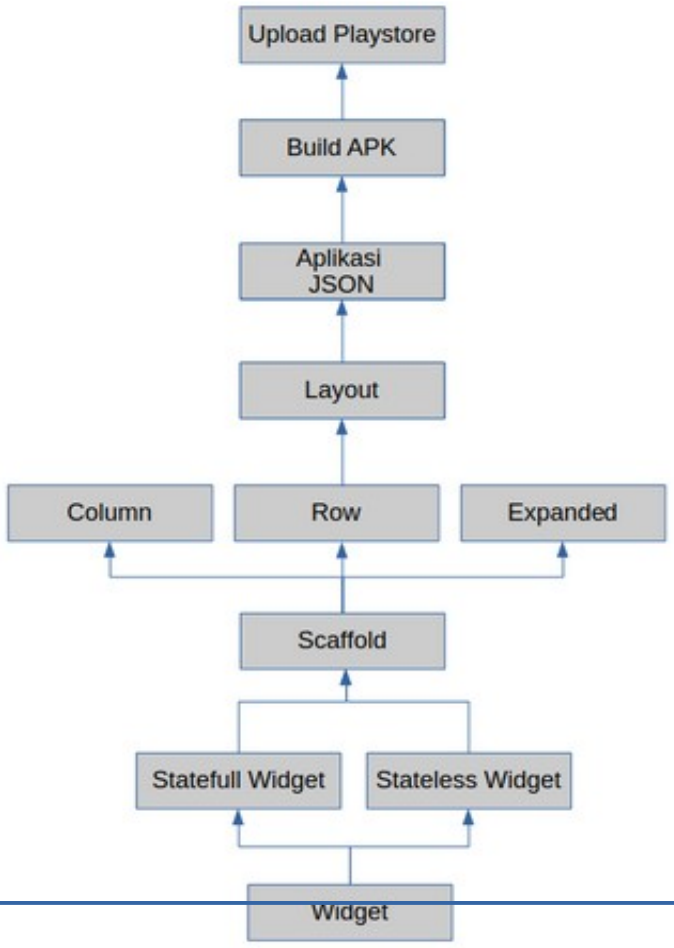
2 Capaian Pembelajaran

Mahasiswa mampu menggunakan Drawer Widget & Routes dan JSON App Overview.

3 Bahan Kajian

Drawer widget & routes, notification widgets, JSON app overview, create android apk, upload playstore

4 Peta Kompetensi Modul



B. Kegiatan Belajar

Pertemuan 15. JSON Overview

Sama seperti bahasa pemrograman pada umumnya pasti ada yang namanya pengolahan data yang berbentuk json termasuklah itu di Flutter. Sebelum kita mempelajari bagaimana melakukan Rest API Call di Flutter alangkah lebih baiknya kita memahami penggunaan dasar json dimana, pada artikel ini kita fokus tentang bagaimana kita melakukan parsing json secara manual.

1. Persiapan

Sebelum kita masuk ke pengolahan json silakan kita buat terlebih dahulu UI **main.dart**-nya seperti berikut.

```
import 'package:flutter/material.dart';void main() => runApp(MaterialApp(
      theme: ThemeData(
        primaryColor: Colors.red,
        accentColor: Colors.orangeAccent,
        primarySwatch: Colors.red,
      ),
      home: MainApp(),
    ));class MainApp extends StatefulWidget {
  @override
  _MainAppState createState() => _MainAppState();
}class _MainAppState extends State<MainApp> {
  String _jsonContent = "";  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Flutter JSON", style: TextStyle(color: Colors.white,),),
      ),
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            mainAxisSize: MainAxisSize.max,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  RaisedButton(
                    onPressed: () {
                      // TODO: do something in here
                    },
                    child: Text("Read JSON File"),
                  ),
                ],
              ),
```

```
37                Padding(
38                  padding: const EdgeInsets.only(top: 12.0),
39                  child: Text(_jsonContent, textAlign: TextAlign.center,),
40                ),
41              ],
42            ),
43          ),
44        ),
45      );
46    }
47 }
```

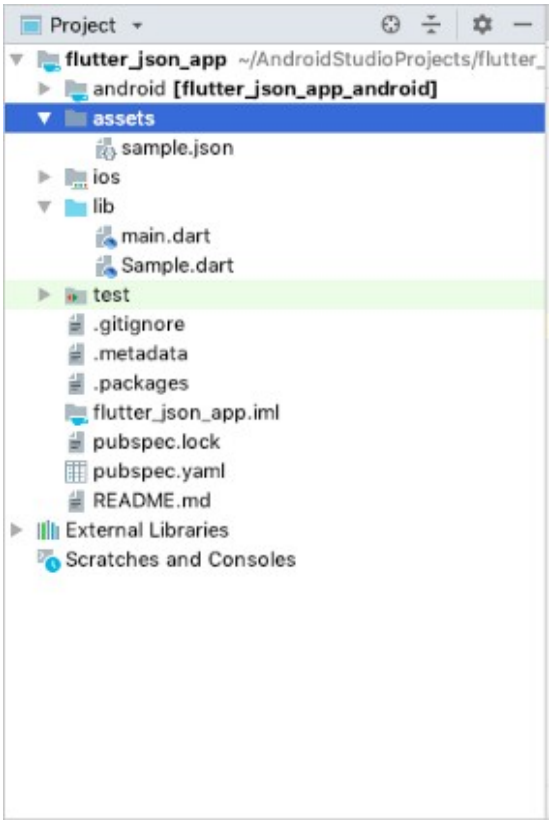
Dimana, output dari kode diatas adalah sebagai berikut.



Jadi, pada UI tersebut kita ada menyediakan satu widget RaisedButton dan Text yang mana nantinya kita beri listener onPressed pada RaisedButton tersebut untuk membaca data json-nya dan hasil dari bacaan tersebut kita tampilkan ke dalam widget Text.

2. JSON Sederhana

Untuk contoh sederhananya, silakan buat file json didalam direktori **assets** seperti berikut.



Buat file dengan nama **sample.json** kemudian ketikan kode dibawah ini :

```
1 {
2   "name": "Yudi Setiawan",
3   "age": 25
4 }
```

Lalu, kita buat class model-nya seperti berikut.

```
1 class Sample {
2   String name;
3   int age;  Sample({this.name, this.age});  @override
4   String toString() {
5     return 'Sample{name: $name, age: $age}';
6   }  factory Sample.fromJson(Map<String, dynamic> json) {
7     return Sample(
8       name: json["name"],
```

```
      age: json["age"]
    );
  }
```

Pada class model tersebut, bisa kita lihat terdapat 2 variable yaitu variable name dan age. Lalu, kita juga ada buat constructor-nya dan method toString()-nya. Selain itu, ada satu lagi method yang mungkin asing kita lihat untuk pertama kalinya yaitu method fromJson(). Sebenarnya method ini bernama Named Constructor di Dart. Named Constructor tersebut kita buat untuk mengubah data json kedalam bentuk class model. Lalu, kenapa kita pakai type dynamic untuk value di Map-nya? Karena, kita tidak pernah tahu didalam json tersebut kira-kira type apa saja yang bakalan tersedia. Contohnya seperti json yang diatas dimana, disitu ada 2 type yang tersedia yaitu type string dan number.

Sekarang kita buat kode untuk membaca file json-nya.

```
Future _loadSampleJson() async {
  String jsonString = await rootBundle.loadString("assets/sample.json");
  final jsonData = json.decode(jsonString);
  Sample sample = Sample.fromJson(jsonData);
  setState(() {
    _jsonContent = sample.toString();
    // sample.name => you can access field from class model
  });
}
```

Pada kode diatas, terdapat 2 keyword yang mungkin belum kita kenal yaitu, async dan await. Lalu, apa fungsinya kedua keyword tersebut. Berikut penjelasannya.

1. **async** berfungsi untuk memberikan eksekusi bahwa kode tersebut dijalankan secara asynchronous.
2. **await** berfungsi untuk memberikan eksekusi didalam asynchronous bahwa eksekusi await harus selesai terlebih dahulu (suspending function) lalu, setelah selesai await melakukan prosesnya dilanjutkan mengeksekusi kode berikutnya didalam asynchronous. Jika kita pernah belajar di Kotlin Coroutine itu ada istilahnya async dan await kira-kira fungsinya hampir sama.

Sekarang, kita panggil fungsi tersebut didalam listener onPressed si RaisedButton.

```
RaisedButton(
  onPressed: () {
    _loadSampleJson();
  },
  child: Text("Read JSON File"),
)
```

Lalu, jangan lupa ya tambahkan assets file sample.json kita didalam file **pubspec.yaml** seperti berikut.

```
name: flutter_json_app
description: Sample JSON in Flutter# The following defines the version and
build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# Read more about versioning at semver.org.
version: 1.0.0+1environment:
  sdk: ">=2.0.0-dev.68.0 <3.0.0"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: ^0.1.2dev_dependencies:
  flutter_test:
    sdk: flutter
# For information on the generic Dart part of this file, see the
# following page: https://www.dartlang.org/tools/pub/pubspec# The following
section is specific to Flutter.
flutter:  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true  # To add assets to your application, add an
assets section, like this:
  # assets:
  #  - images/a_dot_burr.jpeg
  #  - images/a_dot_ham.jpeg
  assets:
    - sample.json  # An image asset can refer to one or more resolution-
specific "variants", see
  # https://flutter.io/assets-and-images/#resolution-aware.  # For details
regarding adding assets from package dependencies, see
  # https://flutter.io/assets-and-images/#from-packages  # To add custom
fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.io/custom-fonts/#from-packages
```

Jika sudah lalu sync packages get-nya. Dan sekarang coba test jalankan app-nya lalu tap button Read JSON File. Dan lihat hasilnya :



2. JSON Array sederhana didalam JSON Object

Sekarang silakan kita ubah lagi file **sample.json** seperti berikut.

```
{
  "name": "Yudi Setiawan",
  "age": 25,
  "hobi": [
    "Coding",
    "Main Game",
    "Menulis"
  ]
}
```

Dikarenakan property hobi merupakan JSON Array maka, kita bisa buat class model-nya seperti berikut.

```
class Sample {
  String name;
  int age;
  List<String> hobi;  Sample({this.name, this.age, this.hobi});
  @override
  String toString() {
    return 'Sample{name: $name, age: $age, hobi: $hobi}';
  }  factory Sample.fromJson(Map<String, dynamic> json) {
    return Sample(
      name: json["name"],
      age: json["age"],
      hobi: List<String>.from(json["hobi"])
    );
  }
}
```

Untuk mem-parsing JSON Array seperti pada contoh json diatas maka, kita bisa mem-parsing-nya dengan cara **List<String>.from(…)** . Yang perlu diperhatikan pada contoh json diatas adalah kita lihat bahwa JSON Array-nya itu isinya String semua jadi kita tidak perlu membuat class model si item-nya. Sekarang coba jalankan lagi programnya.



3. JSON Object didalam JSON Object

Untuk contoh berikut kita akan buat json-nya seperti berikut.

```
{
  "name": "Yudi Setiawan",
  "age": 25,
  "hobi": [
    "Coding",
    "Main Game",
    "Menulis"
  ],
  "github": {
    "username": "CoderJava",
    "repository": 278
  }
}
```

Pada contoh json diatas kita ada menambahkan property github dimana, property tersebut merupakan JSON Object dan didalamnya terdapat property username dan repository. Untuk mem-parsing property github maka, kita perlu membuat class model-nya sendiri.

```
class Github {
  String username;
  int repository;  Github({this.username, this.repository});  @override
  String toString() {
    return 'Github{username: $username, repository: $repository}';
  }  factory Github.fromJson(Map<String, dynamic> json) {
    return Github(
      username: json["username"],
      repository: json["repository"]
    );
  }
}
```

Lalu, pada class model utamanya kita ubah menjadi seperti berikut.

```
class Sample {
  String name;
  int age;
  List<String> hobi;
  Github github;  Sample({this.name, this.age, this.hobi, this.github});
@override
  String toString() {
    return 'Sample{name: $name, age: $age, hobi: $hobi, github: $github}';
  }  factory Sample.fromJson(Map<String, dynamic> json) {
    return Sample(
      name: json["name"],
      age: json["age"],
      hobi: List<String>.from(json["hobi"]),
      github: Github.fromJson(json["github"]),
```

```
15        );
16    }
17
```

Sekarang coba kita jalankan lagi programnya.



C. Evaluasi

Setelah mempelajari materi ini, mahasiswa dapat memahami tentang penggunaan JSON pada pemrograman mobile menggunakan flutter

Latihan	Mahasiswa dapat mempraktikan secara mandiri tentang file JSON dan cara penggunaannya pada flutter
Tugas	-
Refrensi	https://www.byriza.com/flutter-28-parsing-json-pada-flutter https://medium.com/@randy.arba/load-file-json-dalam-flutter-9c721f6a5000

Daftar Pustaka

1. https://junioracademy.id
2. https://petanikode.com
3. https://www.byriza.com/
4. http://indonesiakucemerlang.com/webikc/index.html
5. https://flutter.dev/
6. https://medium.com/
7. https://belajarflutter.com