



MODUL

MOBILE Programming



Disusun Oleh:

Ade Hendini, M.Kom

Agung Sasongko, M.Kom

Muhammad Sony Maulana, ST., M.Kom

UNIVERSITAS BINA SARANA INFORMATIKA

2022

KATA PENGANTAR

Puja dan puji syukur selalu kami panjatkan kehadirat Allah Swt yang telah memberikan semua nikmatnya sehingga penulis berhasil menyelesaikan buku yang berjudul Mobile Programming ini tanpa adanya kendala yang berarti. Tujuan dari penyusunan modul ini adalah untuk memudahkan para mahasiswa Sistem Informasi Fakultas Teknik dan Informatika Universitas Bina Sarana Informatika dalam mengenal dan memahami mobile programming dan praktik pembuatan API. Modul ini disusun dalam tahapan agar pemula dapat lebih mudah mempelajari mobile programming dan mengembangkan aplikasi mobile yang dikhususkan kepada sistem operasi android.

Keberhasilan penyusunan modul ini tentunya bukan atas usaha penulis saja namun ada banyak pihak yang turut membantu dan memberikan dukungan untuk suksesnya penulisan modul ini. Untuk itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan baik secara moril ataupun material sehingga buku ini berhasil disusun.

Modul yang ada ini tentu tidak luput dari kekurangan. Selalu ada celah untuk perbaikan. Kritik, saran serta masukkan dari pembaca sangat kami harapkan untuk penyempurnaan kedepannya.

Pontianak, Oktober 2022

Tim Penulis

DAFTAR ISI

KATA PENGANTAR	2
DAFTAR ISI	3
PERTEMUAN 1	6
Penjelasan Tugas Project.....	6
Pengenalan Mobile Programming.....	6
Perbandingan Flutter dan React Native	7
Perangkat Hardware.....	8
Install Git	8
Install JDK	8
Install Android Studio	14
Install Flutter.....	16
PERTEMUAN 2	21
Konfigurasi Android Studio dengan Flutter	21
Membuat dan Menjalankan Projek Dengan Android Studio.....	22
TUGAS	28
PERTEMUAN 3	29
Membuat dan Menjalankan Projek Dengan VSCode dan Handphone Android.....	29
Install VSCode Sebagai Alternatif Editor	29
Membuat projek flutter dengan VSCode	29
Menjalankan aplikasi dengan Handphone Android.....	32
Membuat dan menjalankan Projek dengan flutlab.io dan Web Emulator	40
Membuat dan Menjalankan Projek di flutlab.io.....	42
PERTEMUAN 4	45
Struktur Folder Flutter	45
Membuat Hello World.....	46
Membuat Widget Column.....	52
Membuat Widget Row	53
StatelessWidget dan StatefullWidget	54
TUGAS	54
PERTEMUAN 5	55
Membuat Halaman Data Poli	55
Membuat Detail Poli.....	57

Menampilkan Detail Poli saat Data Poli diklik	59
TUGAS.....	62
PERTEMUAN 6	63
Memisahkan List Poli menjadi Poli Item.....	63
Membuat Form Tambah Poli.....	64
TUGAS.....	66
PERTEMUAN 7	67
Pemisahan Widget Kedalam Fungsi-Fungsi.....	67
Membuat Fungsi Tombol Simpan dan Menampilkan Data pada Detail Poli.....	68
Membuat Form Ubah Poli	70
TUGAS.....	72
PERTEMUAN 8	72
UJIAN TENGAH SEMESTER	72
PERTEMUAN 9	73
Menampilkan Pesan Konfirmasi Hapus Data	73
Membuat Halaman Beranda dan Login	75
Membuat Halaman Beranda.....	75
Membuat Halaman Login.....	75
Membuat Sidebar Menu (Drawer)	76
TUGAS.....	79
PERTEMUAN 10	80
Apa itu API.....	80
Arsitektur API	80
Install Postman.....	81
Membuat API Klinik dummy dengan mockapi.io	82
Create Poli dengan method POST	86
List Poli dengan method GET.....	86
Show dengan method GET	86
Update Poli dengan method PUT	86
Delete Poli dengan method DELETE	87
TUGAS.....	87
PERTEMUAN 11	88
Menambahkan Depedencies	88
Pada Editor Offline (VSCode).....	88

Pada Editor Online (flutlab.io)	88
Membuat Class Untuk Menyimpan Token	89
Mengarahkan Halaman Awal ke Login	90
Persiapan Koneksi ke API	90
Membuat Service Login.....	91
Membuat Service Poli.....	92
TUGAS.....	93
PERTEMUAN 12	94
Mengambil List Poli dari API	94
Menambah Data Poli ke API	96
TUGAS.....	97
PERTEMUAN 13	98
Mengambil Detail Poli dari API.....	98
Menghapus Data Poli dari API.....	100
Mengubah Data Poli dari API.....	103
TUGAS.....	104
PERTEMUAN 14	105
Presentasi Projek.....	105
PERTEMUAN 15	105
Presentasi Projek.....	105
PERTEMUAN 16	105
Ujian Akhir Semester.....	105

PERTEMUAN 1

Penjelasan Tugas Project

Tugas project diadakan untuk memperoleh nilai UTS dan UAS, dengan kata lain tugas projek ini sebagai pengganti UTS dan UAS. Tugas ini dikerjakan secara kelompok dengan maksimal 1 kelompok sebanyak 5 mahasiswa atau lebih disesuaikan dengan jumlah mahasiswa dalam satu kelas.

Bentuk Tugas Projek :

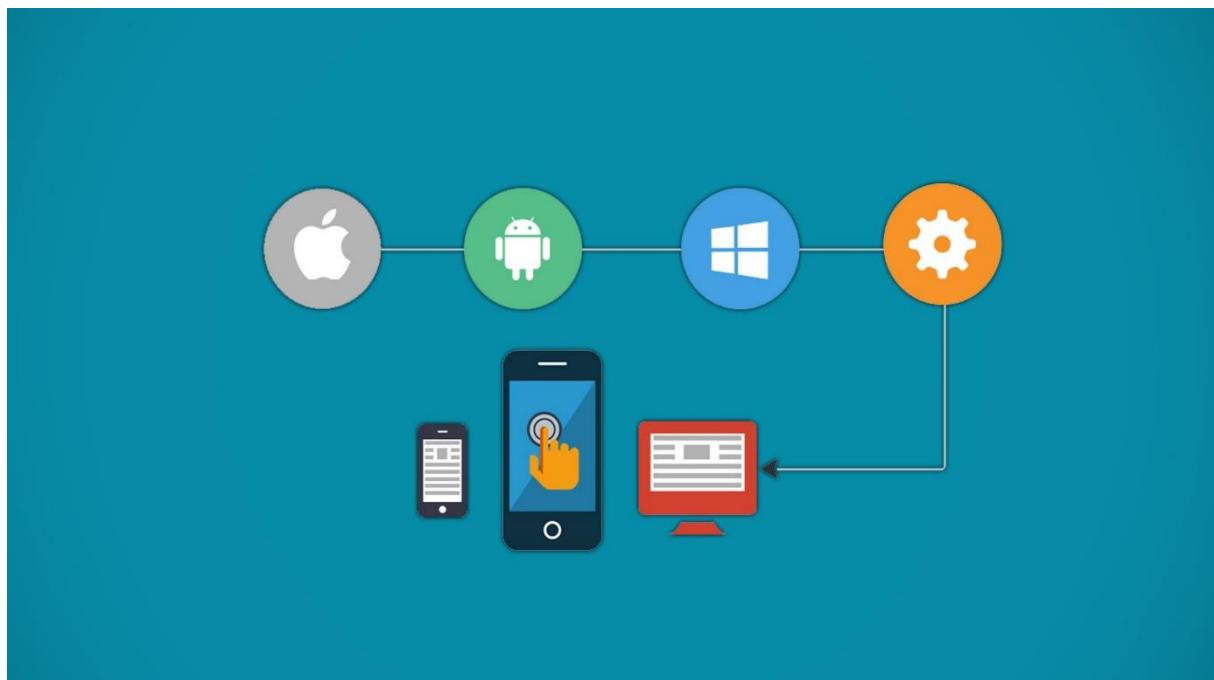
1. Projek merupakan program aplikasi (CRUD) menggunakan flutter dengan minimal 3 tabel.
2. Projek dipresentasikan pada pertemuan 14 dan 15

Pengenalan Mobile Programming

Mobile programming adalah suatu proses pembuatan aplikasi yang diperuntukkan bagi perangkat mobile di sistem operasi android maupun ios, yang bersifat daring maupun luring.

Mobile programming juga dapat diartikan suatu teknik pemrograman yang diterapkan dalam mengembangkan aplikasi di perangkat mobile, seperti smartphone dan tablet PC.

Pengembang aplikasi mobile disebut dengan mobile programmer. Sedangkan untuk pengujian aplikasi mobile dapat menggunakan handphone ataupun emulator, salah satu yang terkenal dalam pengujian aplikasi mobile adalah google android emulator.



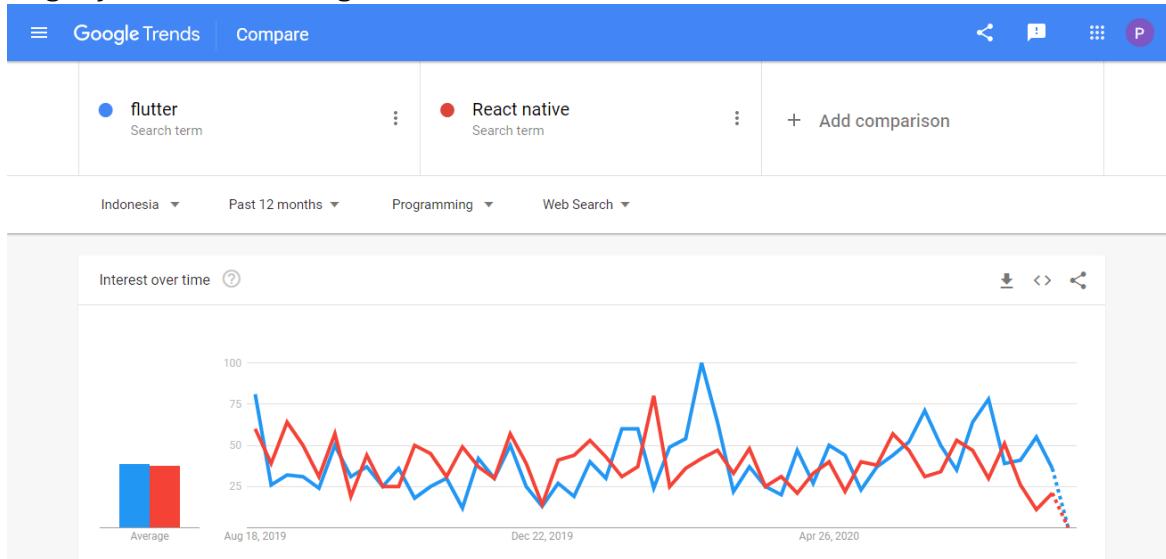
Sumber: <https://www.adwebstudio.com/>

Saat ini pengembangan aplikasi berbasis mobile lebih terfokus kepada tiga sistem operasi yaitu, sistem operasi Android, IoS, dan Microsoft. Meskipun diantara ketiganya yang paling terkenal adalah sistem operasi Android.

Dalam buku ini penekanan pembahasan mobile programming akan difokuskan kepada penggunaan flutter untuk pengembangan aplikasi mobile dan pengembangan API.

Perbandingan Flutter dan React Native

Alasan kenapa pada modul ini penekanan pembahasannya pada pengembangan aplikasi dengan *flutter* bukan dengan *react native* disebabkan:



Sumber: <https://definite.co.id/>

Berdasarkan grafik diatas, terlihat bahwa pencarian *Flutter* (biru) di google angkanya lebih tinggi dibandingkan dengan React Native (merah) antara tahun 2019 sampai 2020. Artinya, semakin banyak yang mencari dan mempelajari pembuatan aplikasi mobile dengan memanfaatkan *flutter*.

	Flutter	React Native
Creator	Google	facebook
Bahasa Pemrograman	 Dart	 JavaScript
UI	Aplikasi akan berperilaku sama dan terlihat serupa di kedua OS (iOS dan Android)	Tampilan dan perilaku aplikasi akan menyesuaikan dengan tiap-tiap OS (iOS dan Android)
Aplikasi	Google Ads App, Xianyu app by Alibaba, Hamilton	Facebook, Instagram, Tesla, Skype

Sumber: <https://definite.co.id/>

Flutter merupakan Toolkit UI portable/seperangkat SDK yang dimanfaatkan untuk membangun aplikasi dan di-compile secara native ke desktop, web, maupun mobile dari satu project saja. Sebaliknya React Native merupakan framework untuk mengembangkan aplikasi native yang menggunakan react. React sendiri adalah library Javascript terpopuler untuk membuat user interface (UI).

Lebih jelas perbandingan pengembangan aplikasi mobile antara flutter dan react native ditunjukkan pada gambar 1.3.

Perangkat Hardware

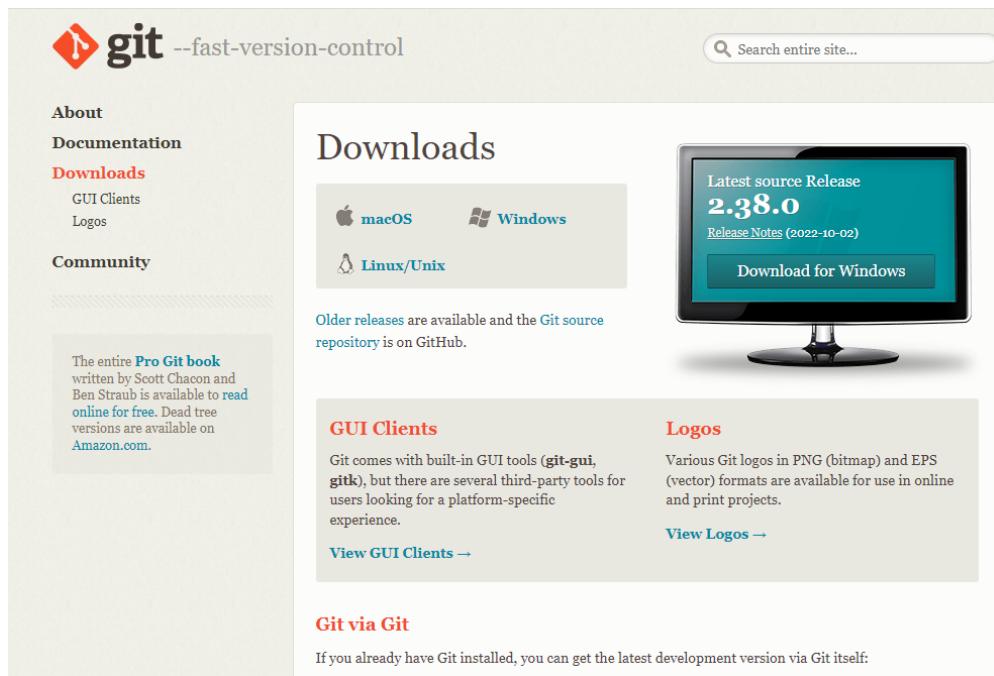
Sebelum memulai melakukan mobile programming ada beberapa hal yang harus kita persiapkan mulai dari penyiapan perangkat hardware maupun software.

Untuk kebutuhan spesifikasi minimum hardware yang perlu dipersiapkan adalah:

- 1) Prosesor Intel Core i5
- 2) RAM 8 GB
- 3) Hardisk Space 1,5 GB

Install Git

Buka laman <https://git-scm.com/downloads>, kemudian klik tombol download



Kemudian lakukan installasi git dari file yang telah diunduh.

Install JDK

JDK (Java Development Kit) adalah sebuah perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode java ke bytecode yang dapat dimengerti dan dapat dijalankan oleh JRE (Java Runtime Environment). JDK wajib terinstall pada komputer yang akan melakukan proses pembuatan aplikasi berbasis java, namun tidak wajib terinstall di komputer yang akan menjalankan aplikasi yang dibangun dengan java.

JDK dapat diunduh pada laman <https://jdk.java.net/>

jdk.java.net

Production and Early-Access OpenJDK Builds, from Oracle

Ready for use: JDK 19, JDK 18, JMC 8

Early access: JDK 20, Generational ZGC, Jextract, Loom, Metropolis, Panama, & Valhalla

Looking to learn more about Java? Visit [dev.java](#) for the latest Java developer news and resources.

Looking for Oracle JDK builds and information about Oracle's enterprise Java products and services? Visit the [Oracle JDK Download page](#).

Pilih **JDK 19** atau dapat juga memilih jdk terbaru, kemudian download file zip untuk windows, jika menggunakan windows

jdk.java.net

GA Releases

JDK 19
JDK 18
JMC 8

Early-Access Releases

JDK 20
Generational ZGC
Jextract
Loom
Metropolis
Panama
Valhalla

Reference Implementations

Java SE 19
Java SE 18
Java SE 17
Java SE 16
Java SE 15
Java SE 14
Java SE 13
Java SE 12
Java SE 11
Java SE 10
Java SE 9
Java SE 8
Java SE 7

OpenJDK JDK 19 General-Availability Release

This page provides production-ready open-source builds of the Java Development Kit, version 19, an implementation of the Java SE 19 Platform under the GNU General Public License, version 2, with the Classpath Exception.

Commercial builds of JDK 19 from Oracle, under a non-open-source license, can be found at the [Oracle Technology Network](#).

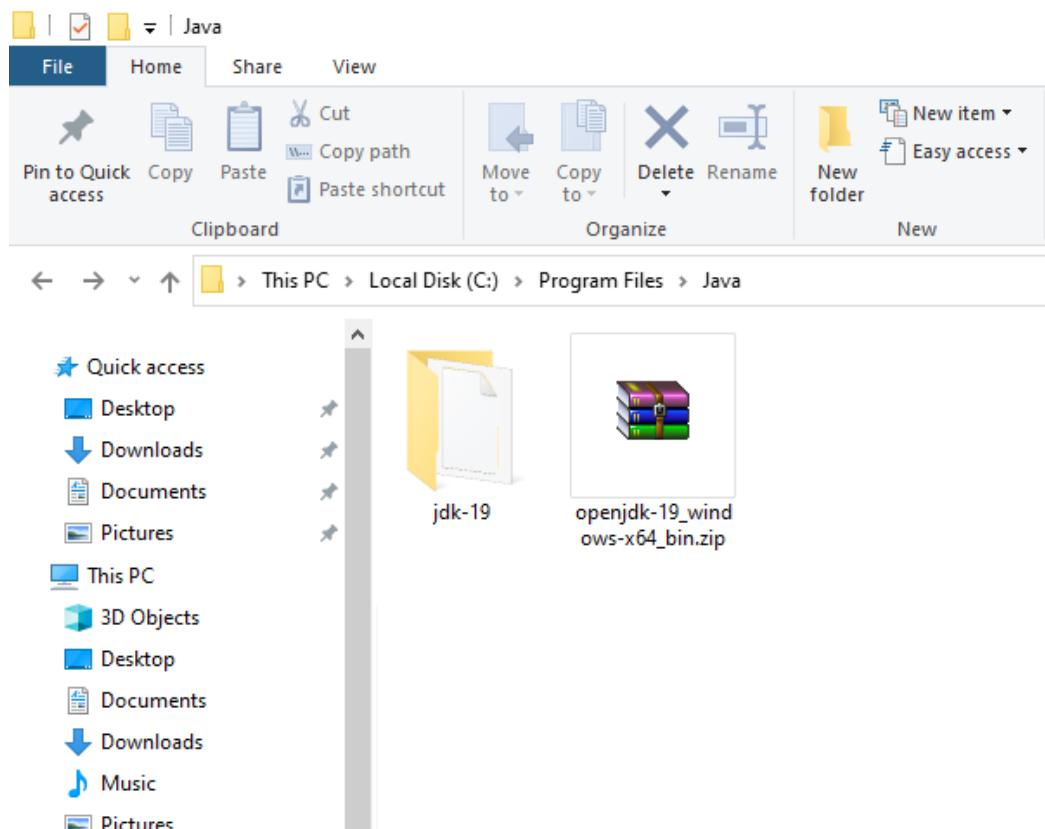
Documentation

- [Features](#)
- [Release notes](#)
- [API Javadoc](#)

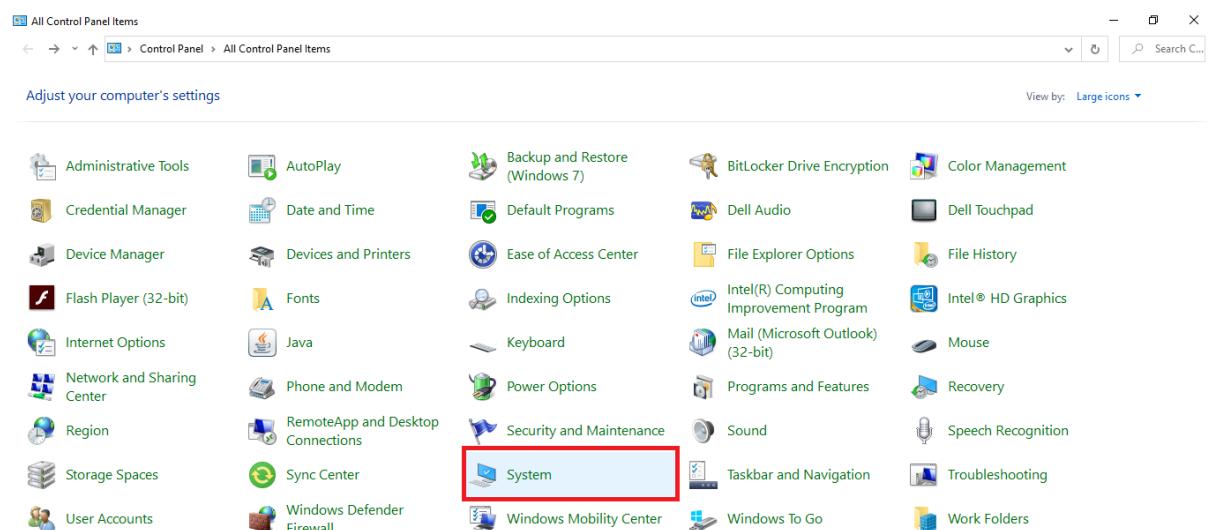
Builds

Linux/AArch64	tar.gz (sha256)	194401252 bytes
Linux/x64	tar.gz (sha256)	195673378 bytes
macOS/AArch64	tar.gz (sha256)	190416515 bytes
macOS/x64	tar.gz (sha256)	192323254 bytes
Windows/x64	zip (sha256)	194357214 bytes

Kemudian extrak berkas file tersebut pada laptop/komputer di alamat **C:\Program Files\Java**



Kemudian buka **Control Panel**, pilih "System"



Kemudian pilih "Advanced System Setting"

Related settings

BitLocker settings

Device Manager

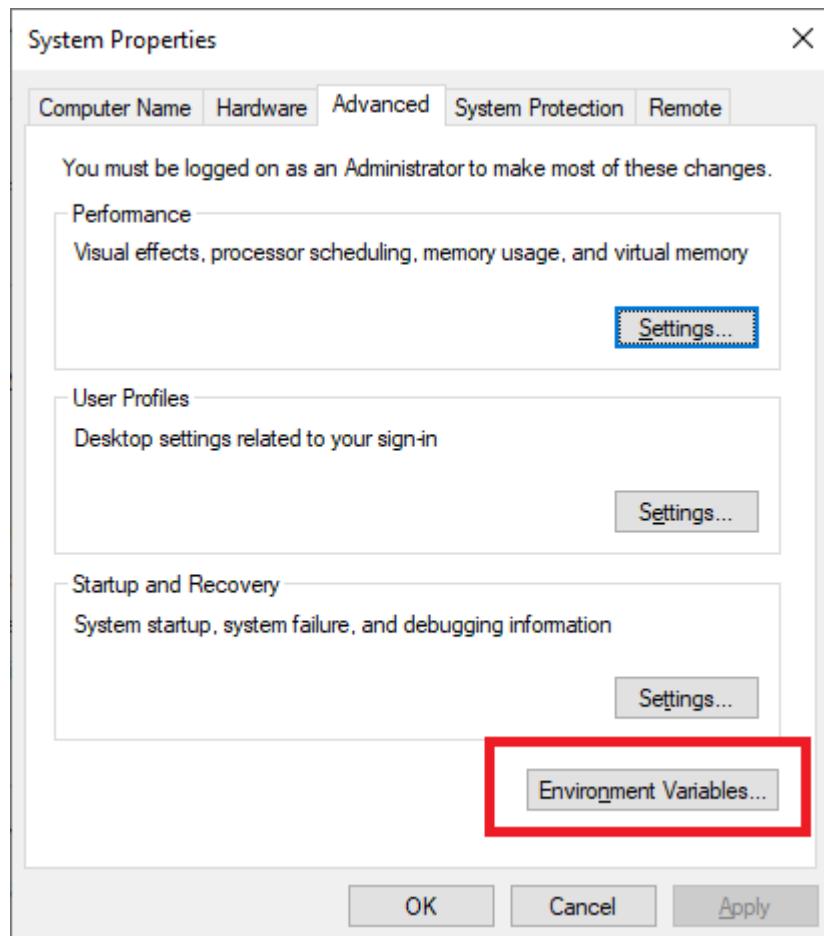
Remote desktop

System protection

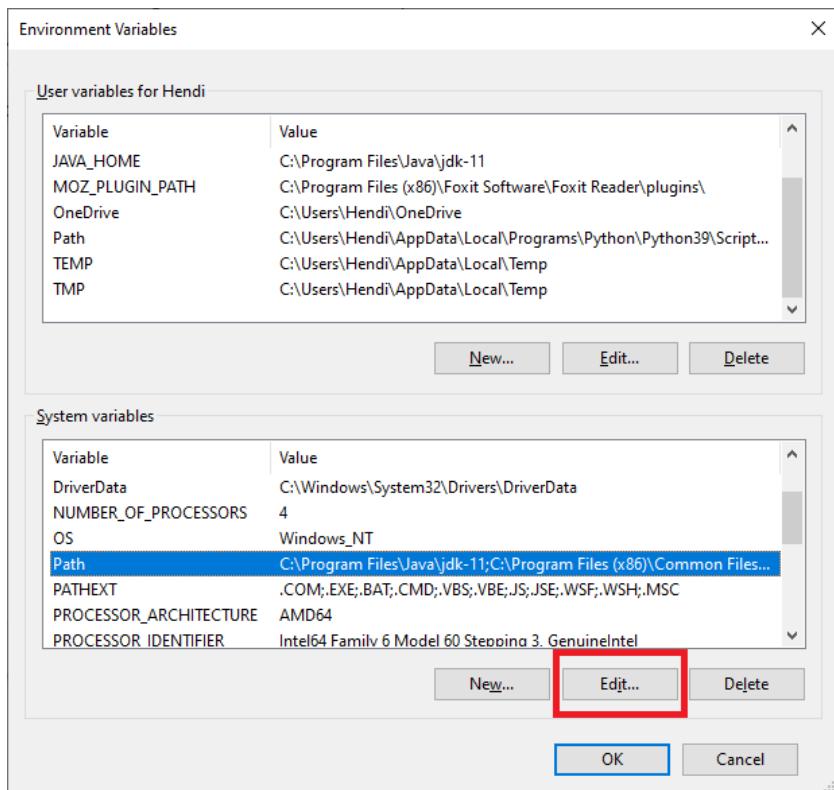
Advanced system settings

Rename this PC (advanced)

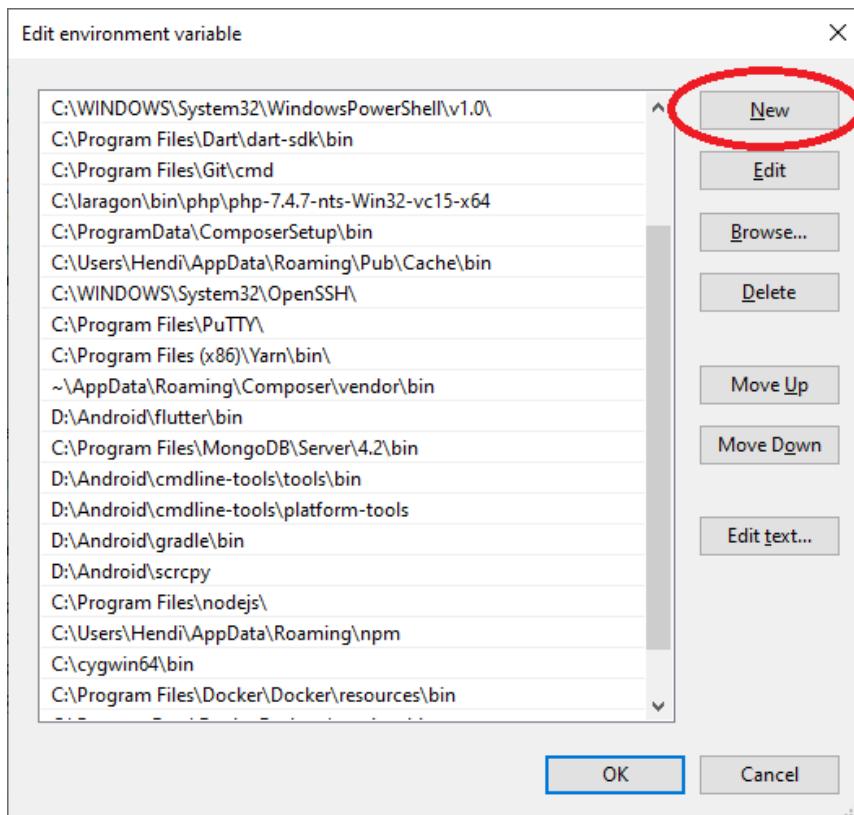
Kemudian klik tombol "Environtment Variables"



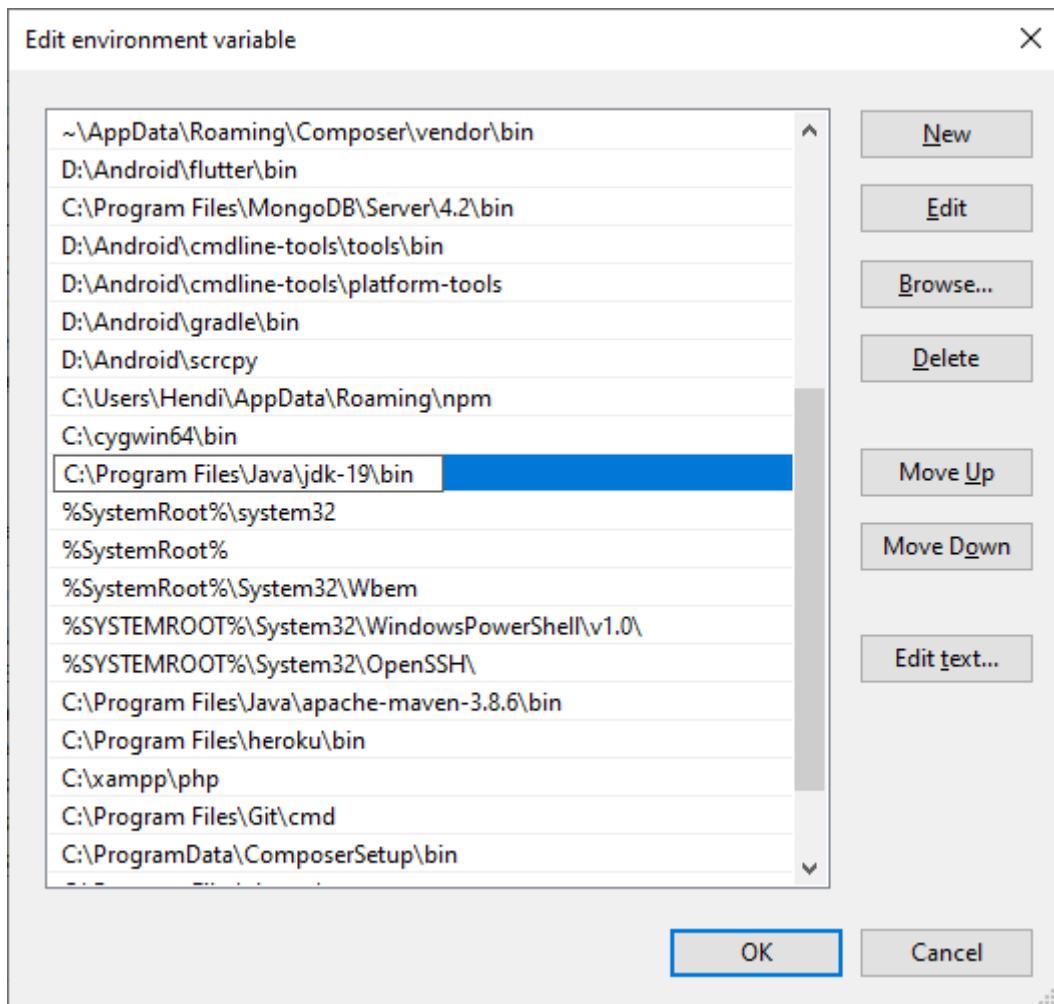
Pilih **Path** pada bagian System Variables kemudian Klik tombol "Edit"



Kemudian klik tombol “New”



Kemudian masukkan alamat folder bin pada jdk yang telah kita ekstrak dalam hal ini misalnya "C:\Program Files\Java\jdk-19\bin"



Biasanya agar JDK dapat berfungsi perlu dilakukan restart laptop/komputer. Untuk memeriksa apakah instalasi berhasil, buka command prompt kemudian ketikkan **java --version** atau **javac --version**

```
PS C:\> java --version
openjdk 19 2022-09-20
OpenJDK Runtime Environment (build 19+36-2238)
OpenJDK 64-Bit Server VM (build 19+36-2238, mixed mode, sharing)
PS C:\> javac --version
javac 19
PS C:\> |
```

A screenshot of a Windows PowerShell window titled 'Windows PowerShell'. The command 'java --version' was run, displaying the Java version as 'openjdk 19 2022-09-20' and the runtime environment as 'OpenJDK Runtime Environment (build 19+36-2238)'. The command 'javac --version' was also run, displaying the Java compiler version as 'javac 19'. The prompt 'PS C:\> |' is visible at the bottom.

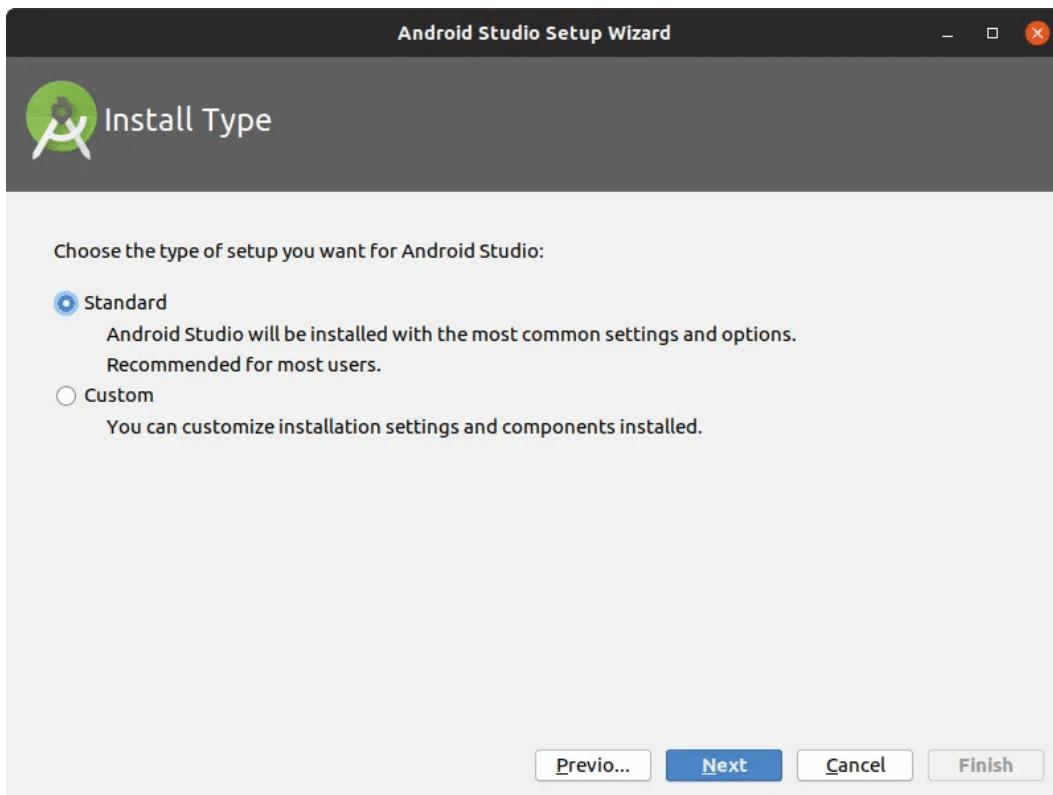
Install Android Studio

Android studio dapat diunduh pada laman <https://developer.android.com/studio> .

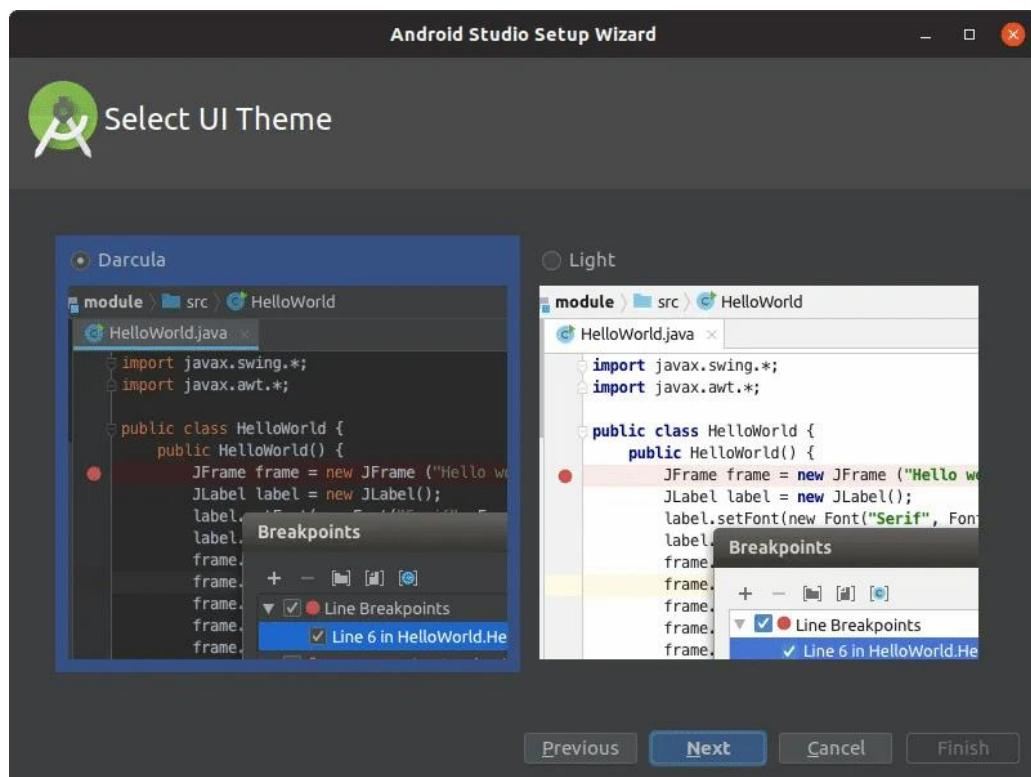
Setelah diunduh klik dua kali pada file yang telah diunduh tersebut, kemudian lakukan installasi dengan mengikuti langkah-langkah yang telah disediakan



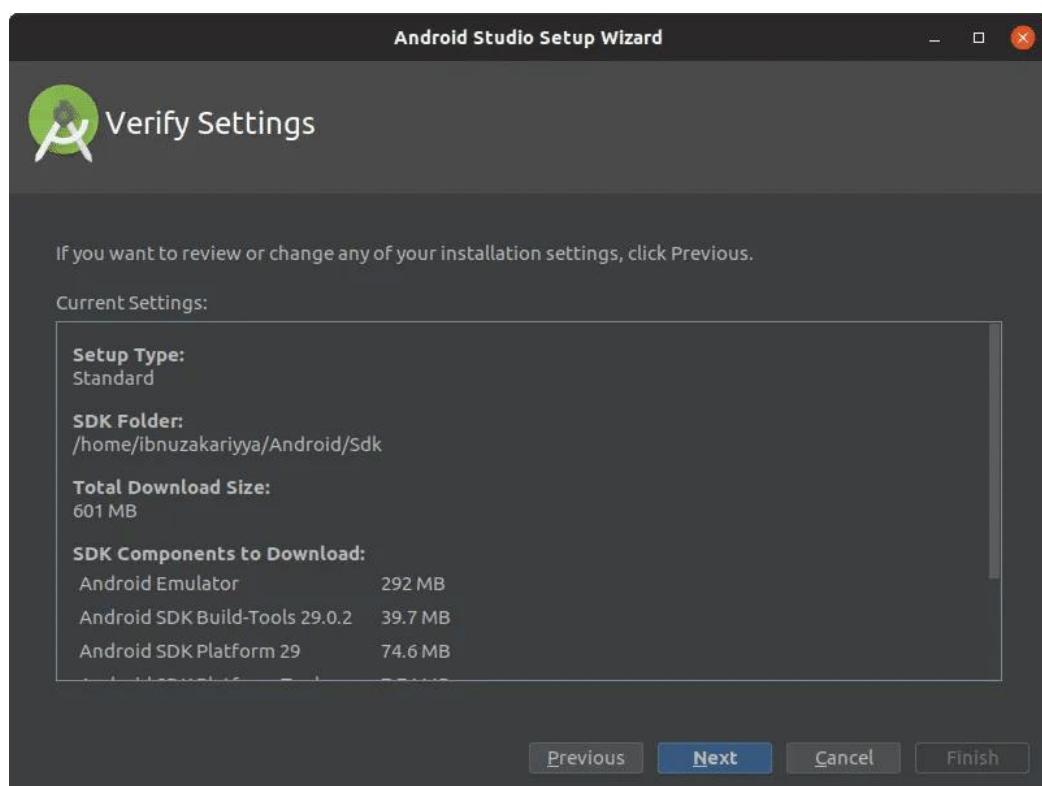
Kemudian pilih tipe standar dan klik next



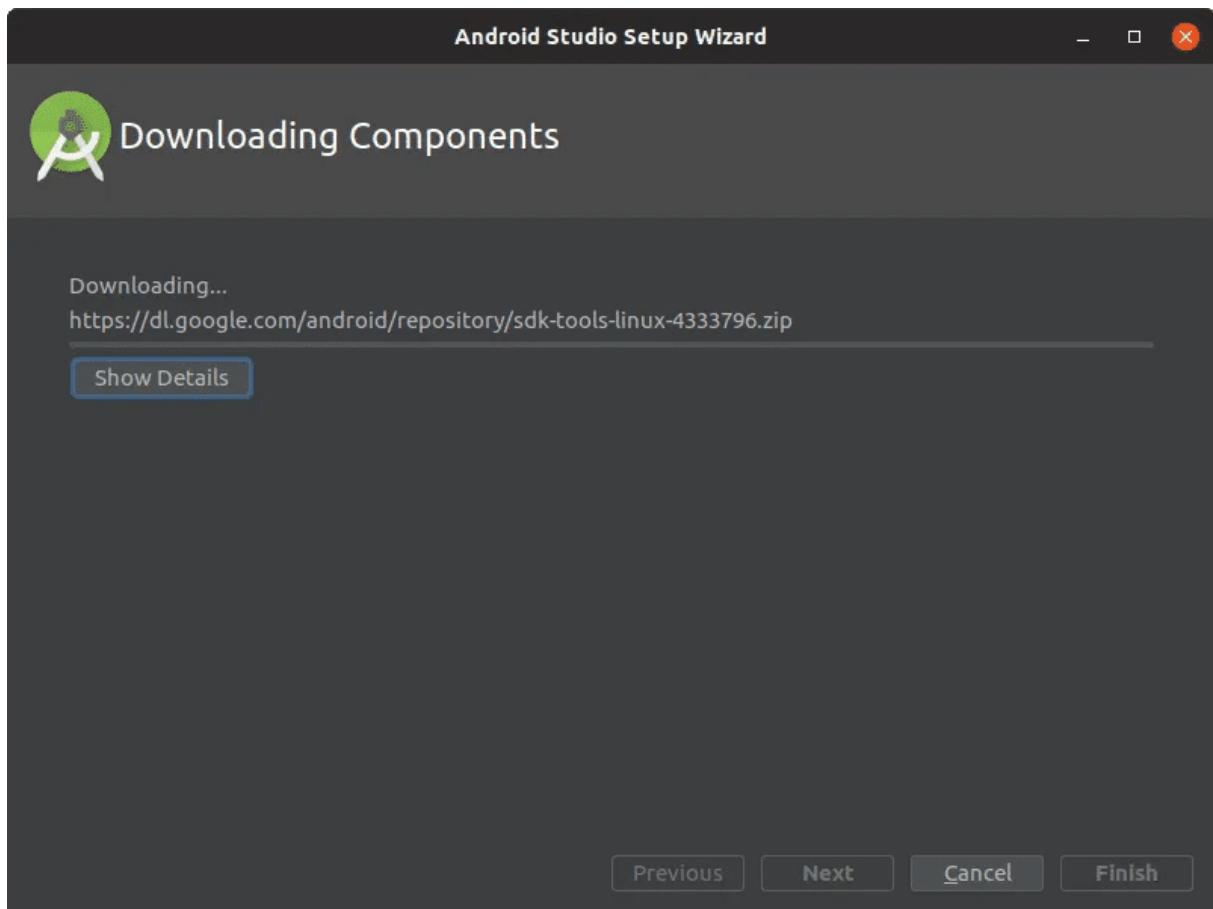
Pilih thema tampilan kemudian klik next



Kemudian klik tombol next



Pastikan komputer terhubung dengan internet yang stabil, karena android studio akan mengunduh komponen-komponen yang diperlukan



Setelah selesai klik finish

Install Flutter

Flutter adalah sebuah framework open-source yang dikembangkan oleh Google untuk membangun antarmuka (*user interface/UI*) aplikasi Android dan iOS.

Apa bedanya membuat aplikasi android menggunakan Java/Kotlin (*native*) dengan Flutter.

Dari bahasa pemrograman yang digunakan, Flutter menggunakan bahasa pemrograman Dart, sedangkan Android Native menggunakan bahasa pemrograman Java dan Kotlin. Aplikasi yang kita buat dengan Flutter dapat di-build ke Android dan iOS. Sedangkan Android Native hanya bisa di-build ke Android saja.

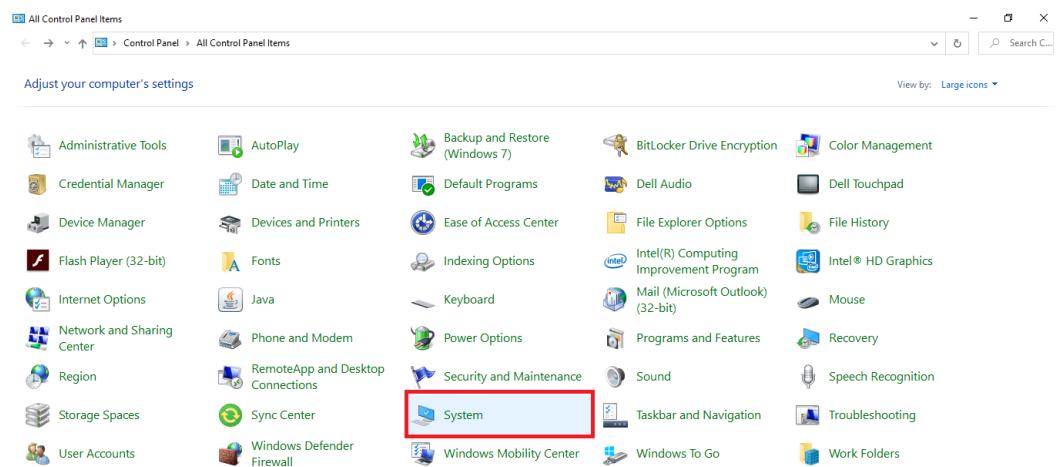
Untuk menginstall flutter, buka laman <https://flutter.dev/docs/get-started/install>

Kemudian pilih "Windows"

Kemudian pilih **flutter_windows_** untuk mengunduh file flutter

Kemudian ekstrak file zip flutter misalnya di "D:/Android"

Kemudian buka **Control Panel**, pilih “System”

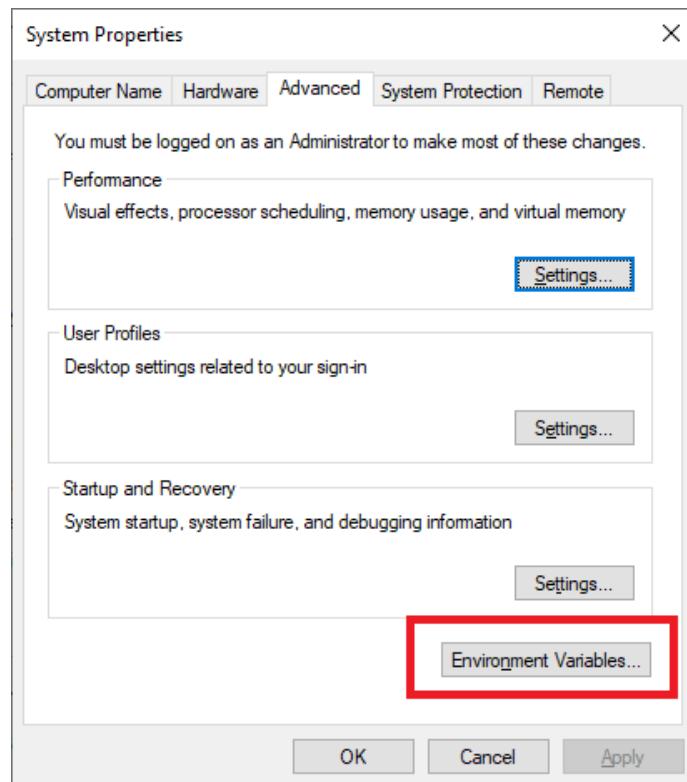


Kemudian

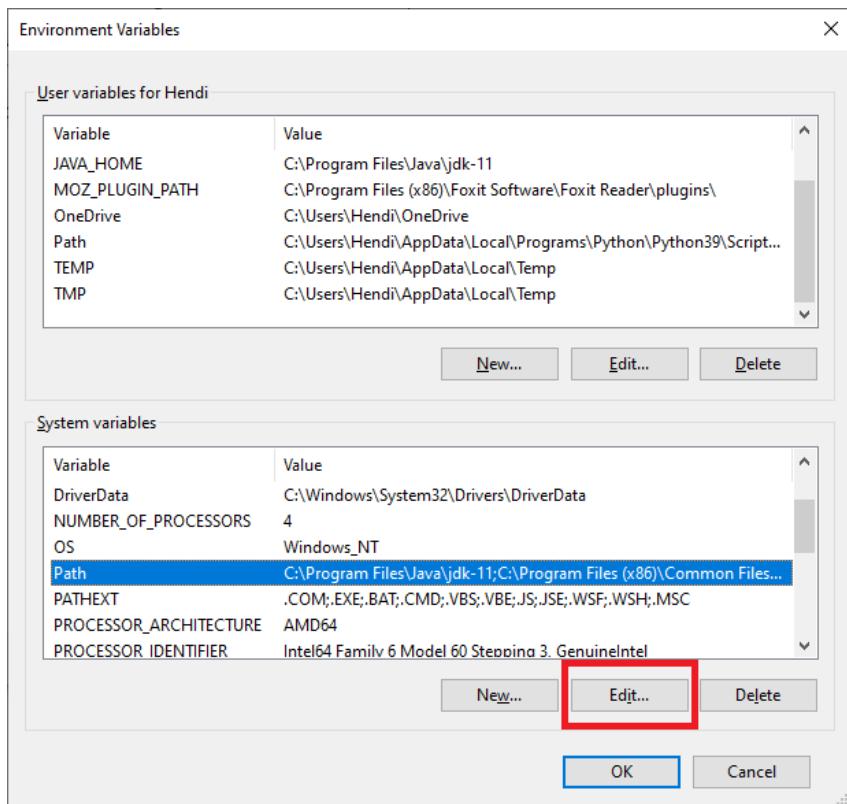
pilih “Advanced System Setting”



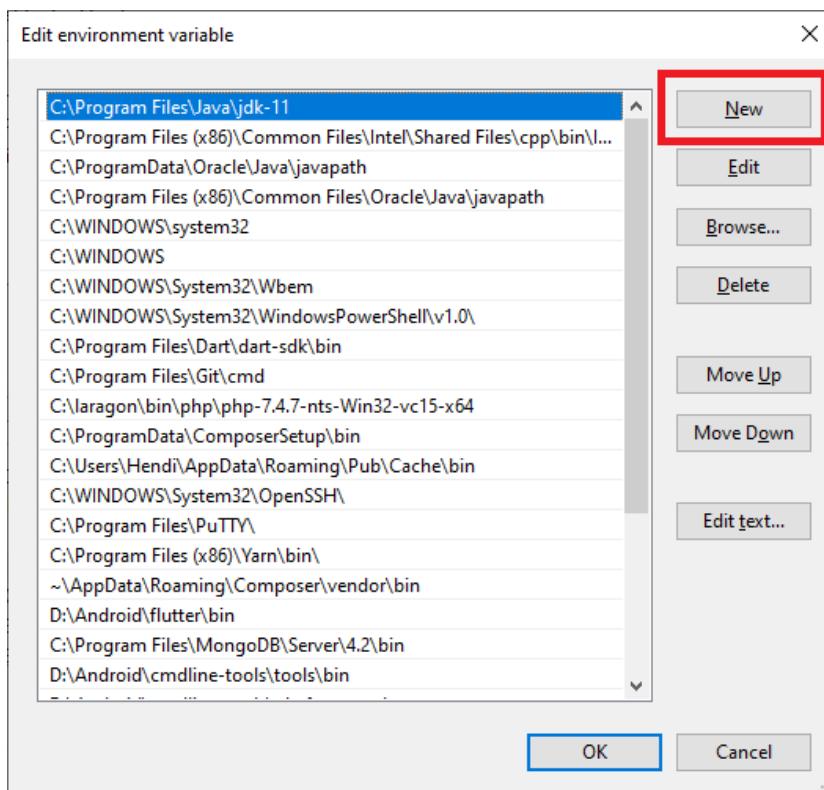
Kemudian klik tombol “Environtment Variables”



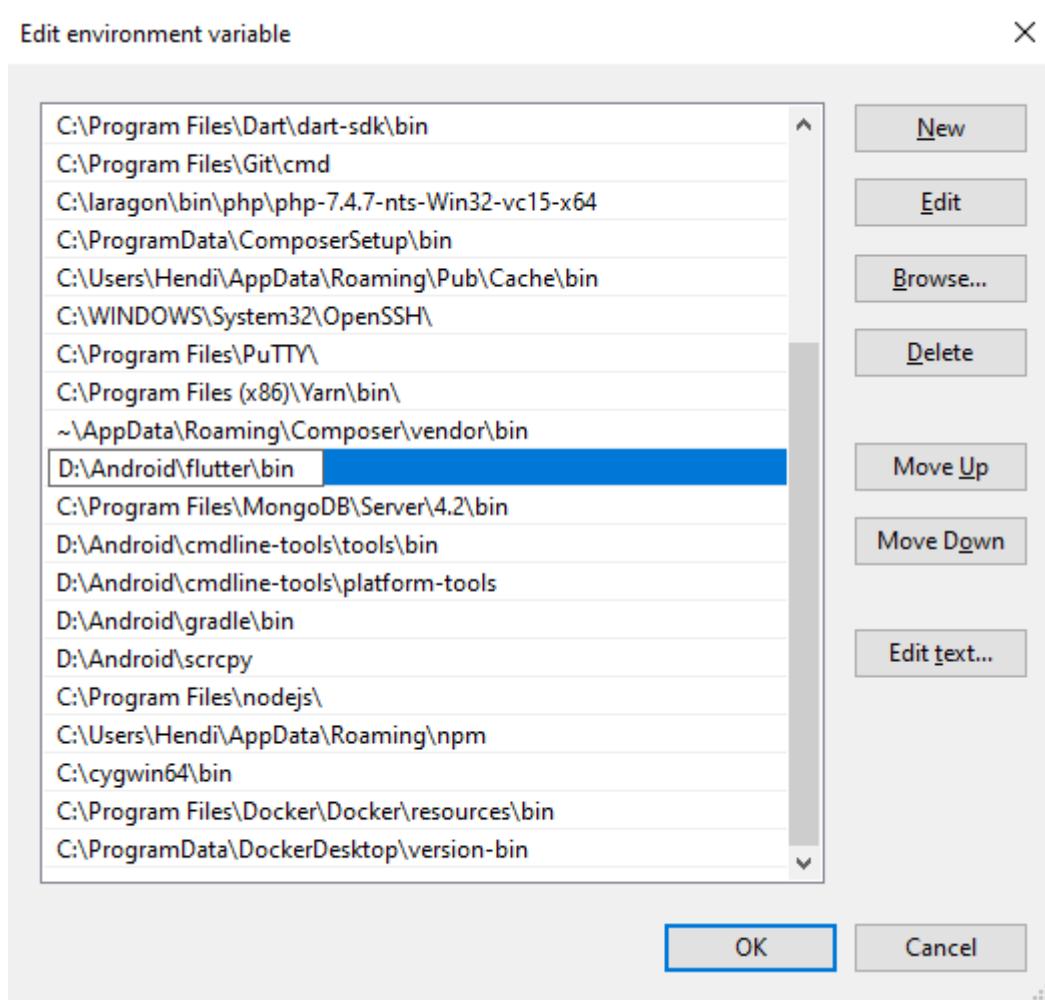
Kemudian pilih pada “System Variables” pilih “Path” dan klik tombol “Edit”



Kemudian klik tombol “New”



Kemudian masukkan alamat folder bin pada flutter yang telah kita ekstrak dalam hal ini misalnya "D:\Android\flutter\bin"



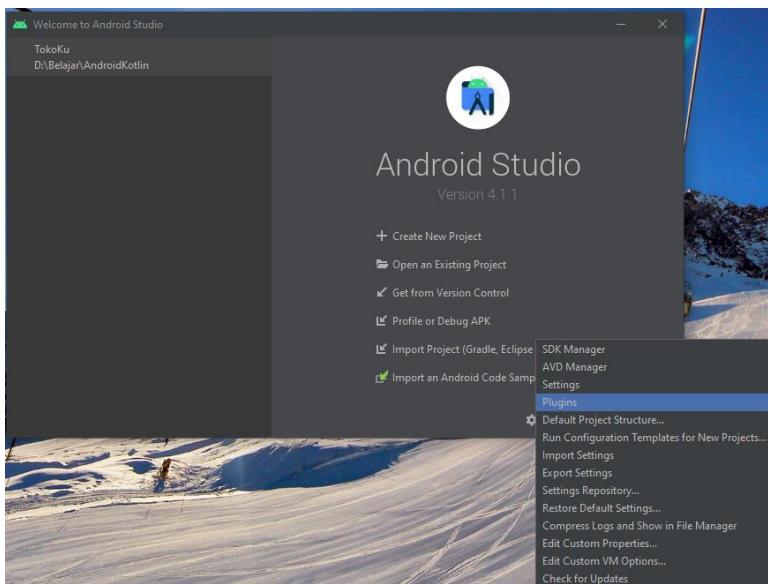
Biasanya agar flutter dapat berfungsi perlu dilakukan restart laptop/komputer. Untuk memeriksa apakah installasi berhasil, buka command prompt kemudian ketikkan **flutter --version**

```
PS C:\> flutter --version
Flutter 3.3.4 • channel stable • https://github.com/flutter/flutter.git
Framework • revision eb6d86ee27 (13 days ago) • 2022-10-04 22:31:45 -0700
Engine • revision c08d7d5efc
Tools • Dart 2.18.2 • DevTools 2.15.0
PS C:\> |
```

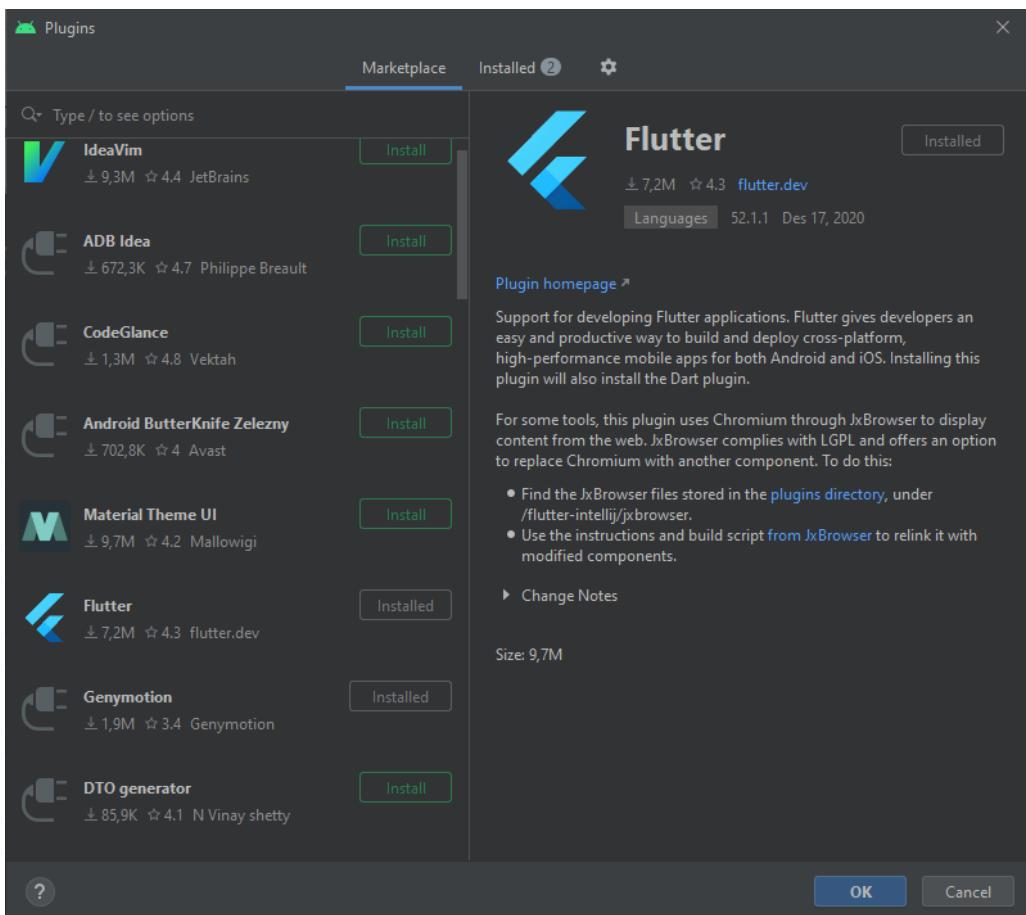
PERTEMUAN 2

Konfigurasi Android Studio dengan Flutter

Jalankan Android Studio kemudian pada menu “Configure” pilih “Plugins”



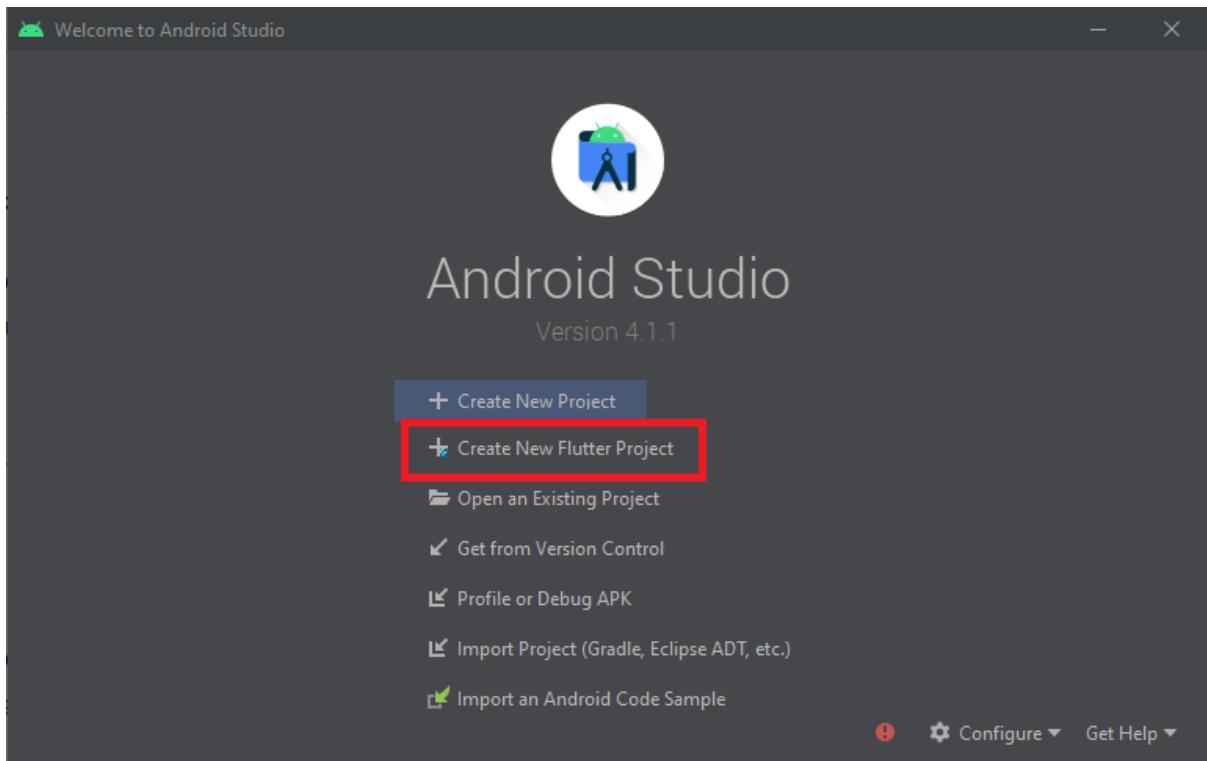
Pilih “Flutter” kemudian klik tombol “Install”



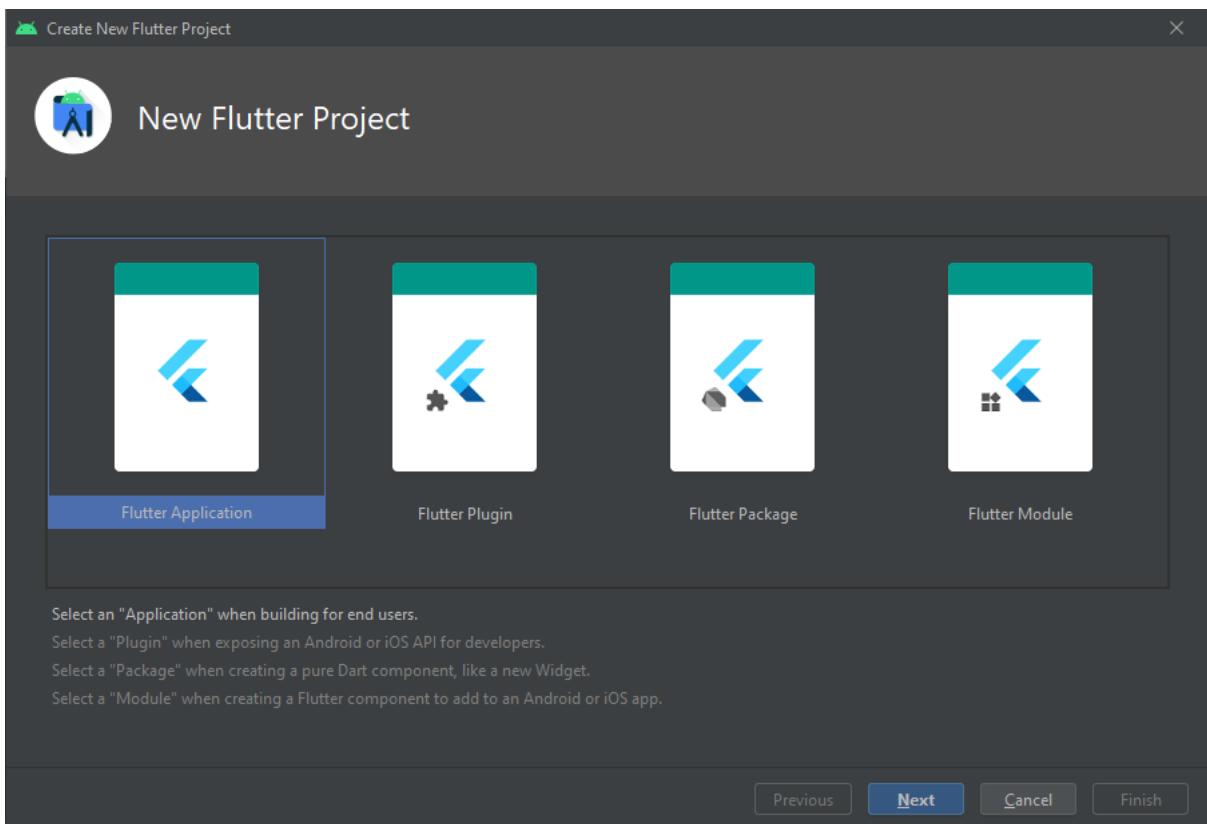
Setelah selesai, restart/tutup Android Studio

Membuat dan Menjalankan Projek Dengan Android Studio

Jalankan Android Studio kemudian pilih “Create New Flutter Project”



Kemudian pilih “Flutter Application” dan klik “Next”



Kemudian tentukan :

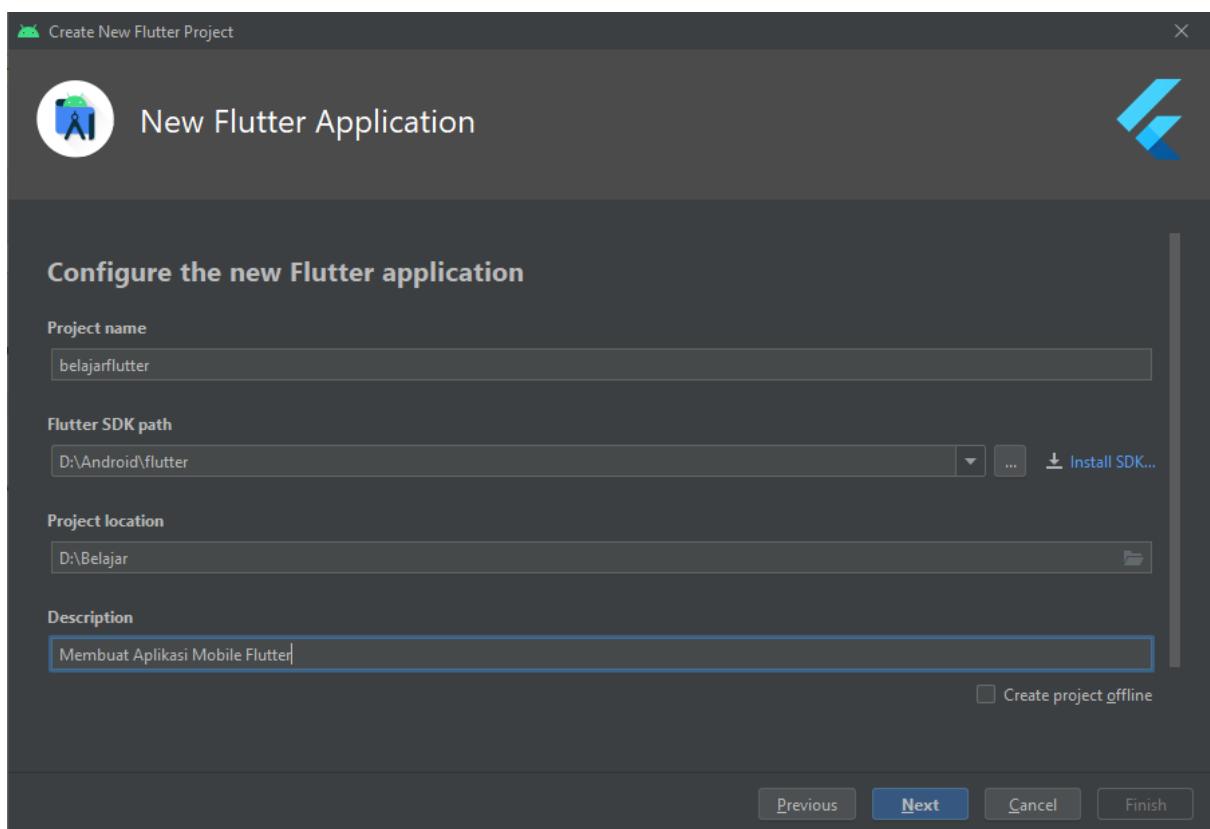
Project Name : belajarflutter

Flutter SDK path : D:\Android\flutter (masukkan sesuai alamat folder flutter diletakkan)

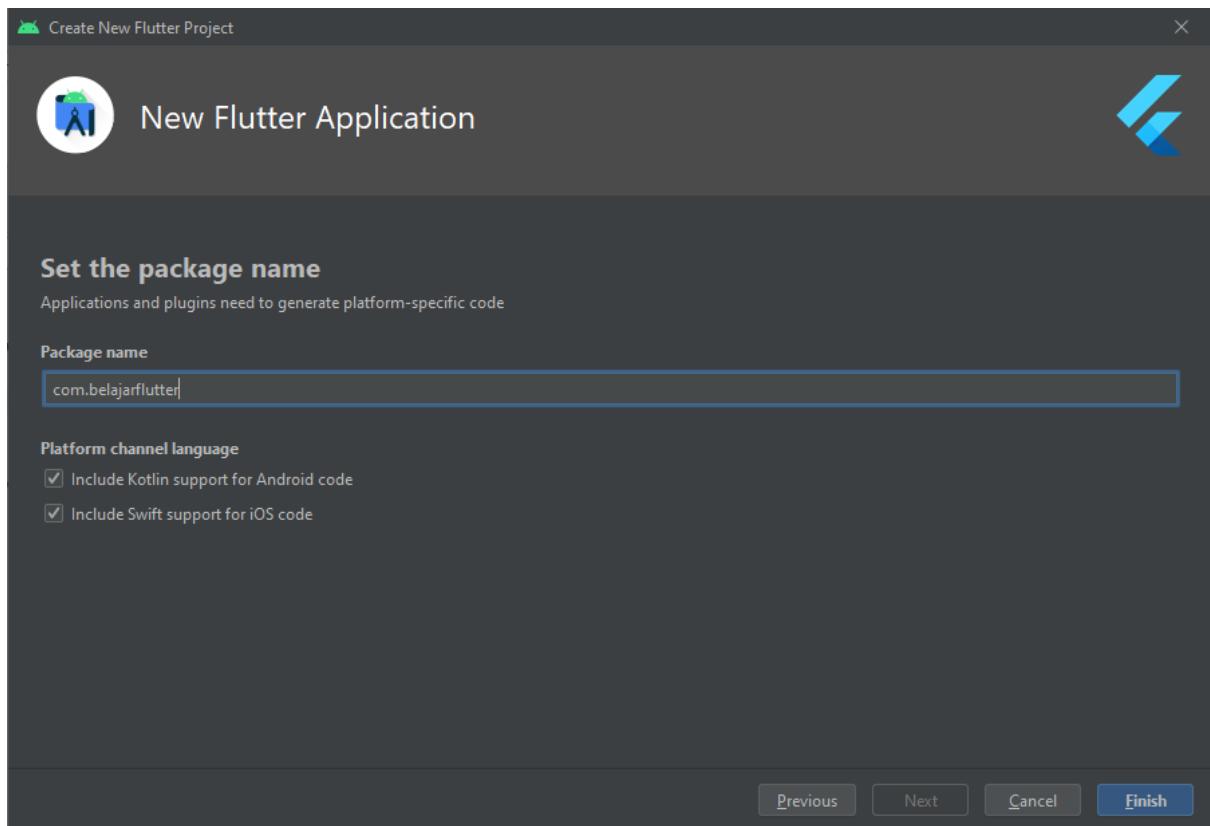
Project location : D:\Belajar (bebas memasukkan dimanapun)

Description : Membuat Aplikasi Mobile Flutter (bebas)

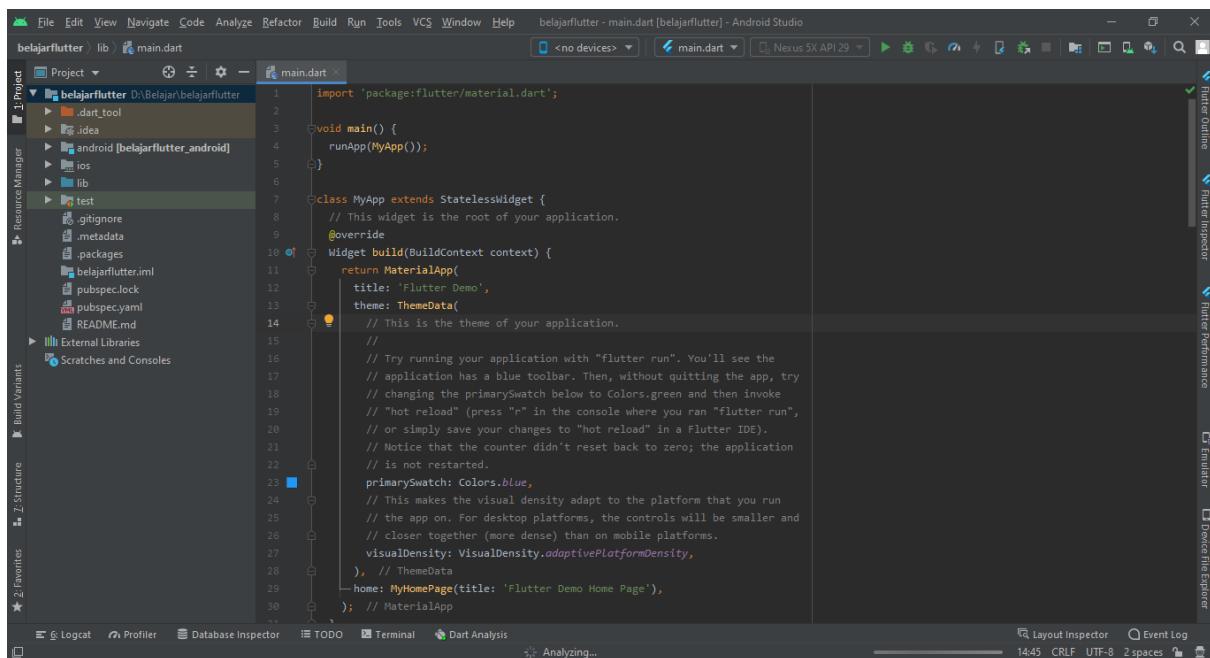
Kemudian klik tombol “Next”



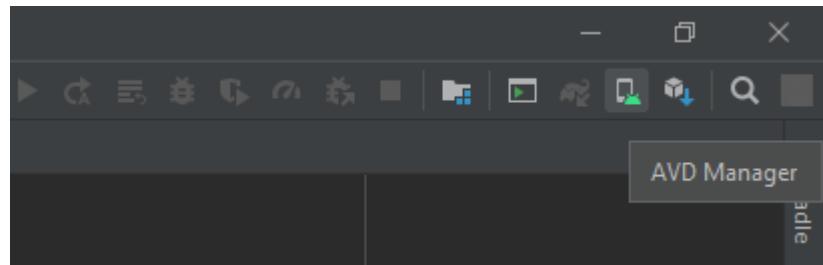
Kemudian klik tombol “Finish”. Pastikan perangkat terhubung ke internet, karena diperlukan untuk mengunduh projek yang dibuat



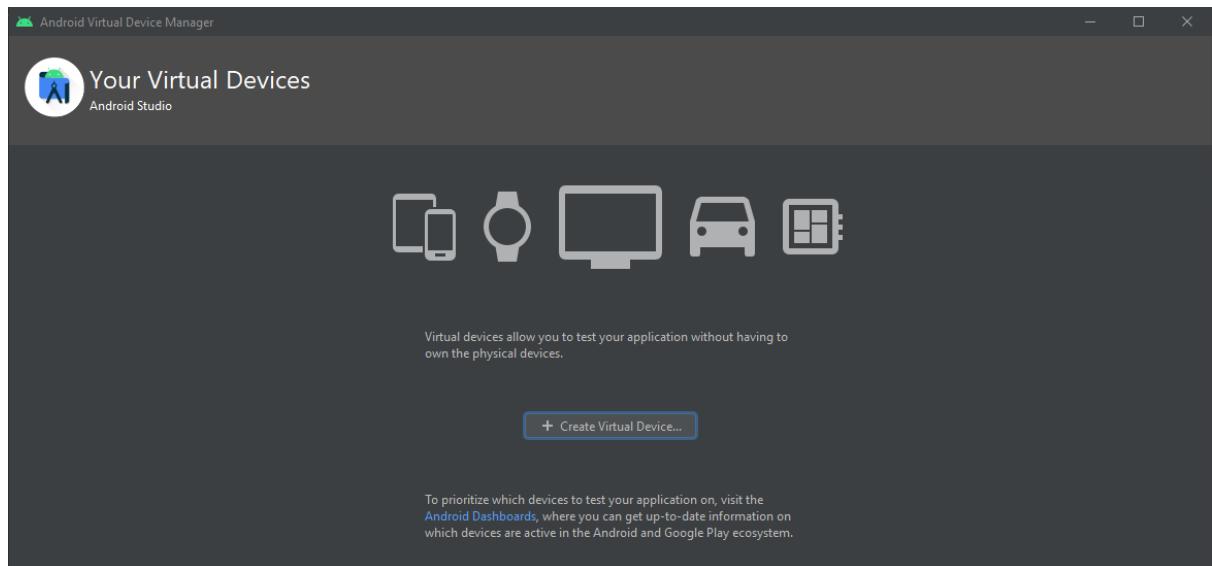
Setelah selesai akan muncul projek yang telah dibuat



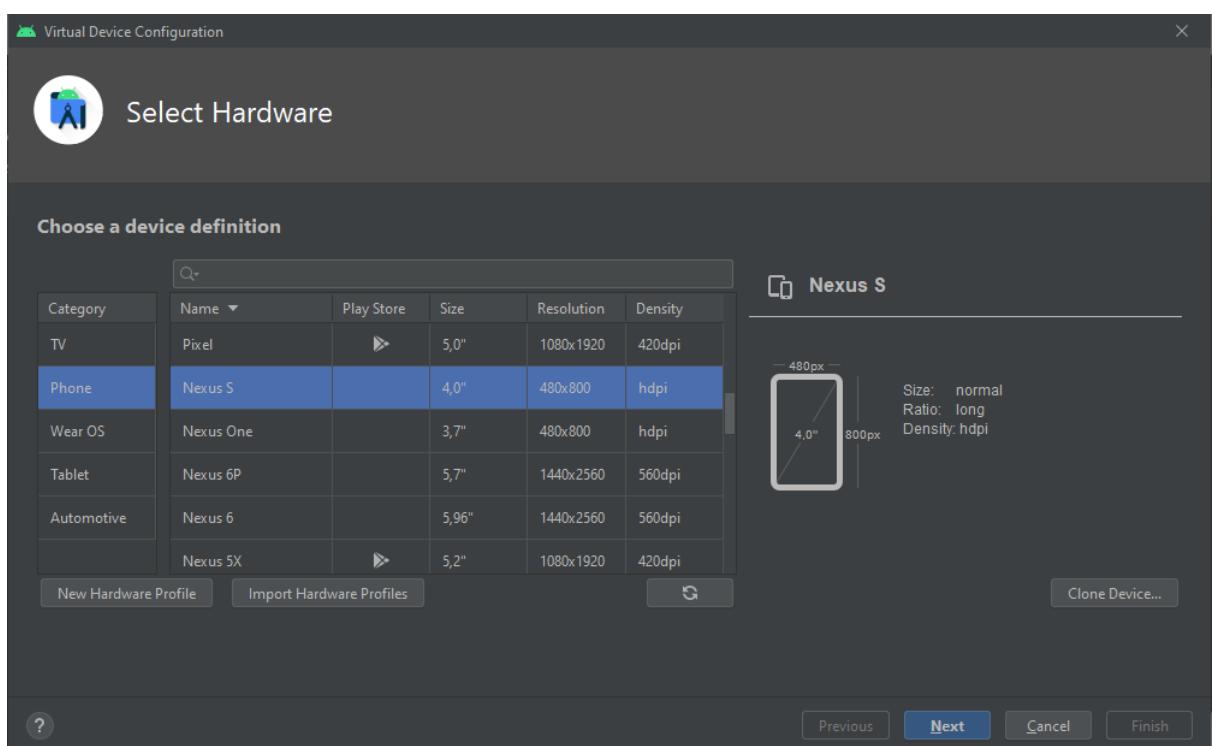
Untuk menjalankan projek, kita memerlukan emulator android. Pertama kita akan membuat emulator android dengan cara klik "AVD Manager" pada pojok kiri atas



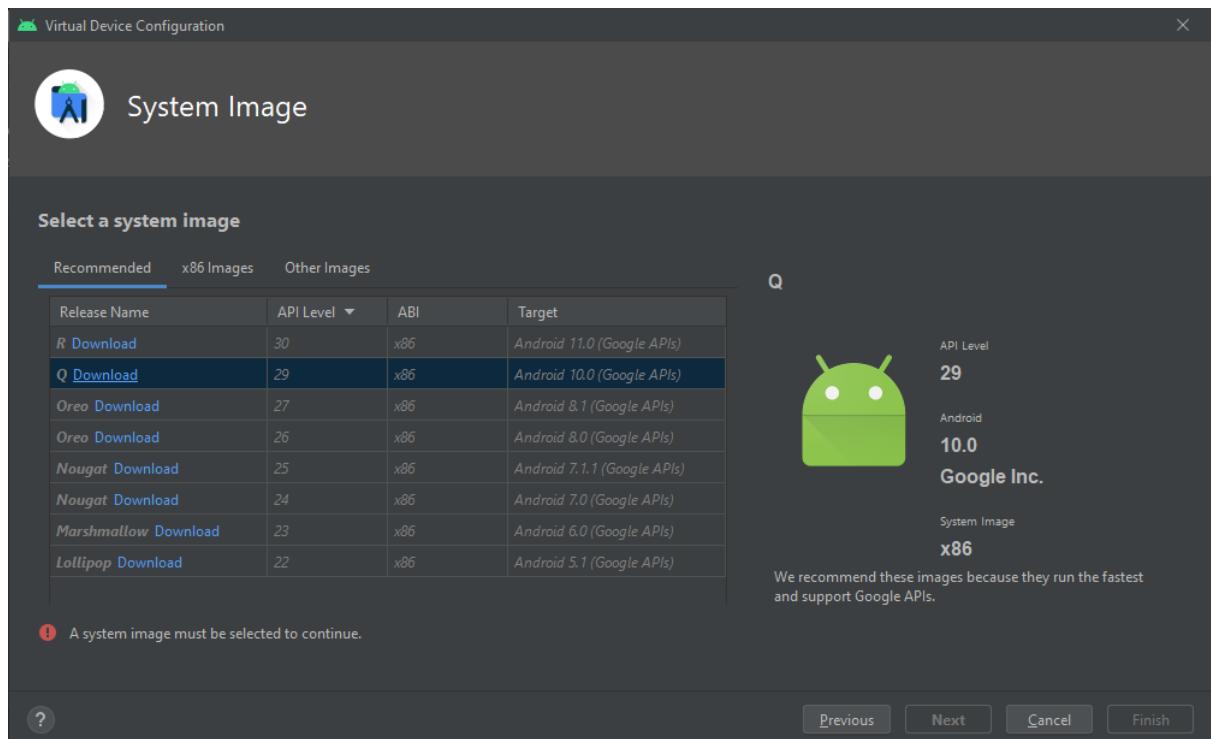
Kemudian klik tombol “Create Virtual Device”



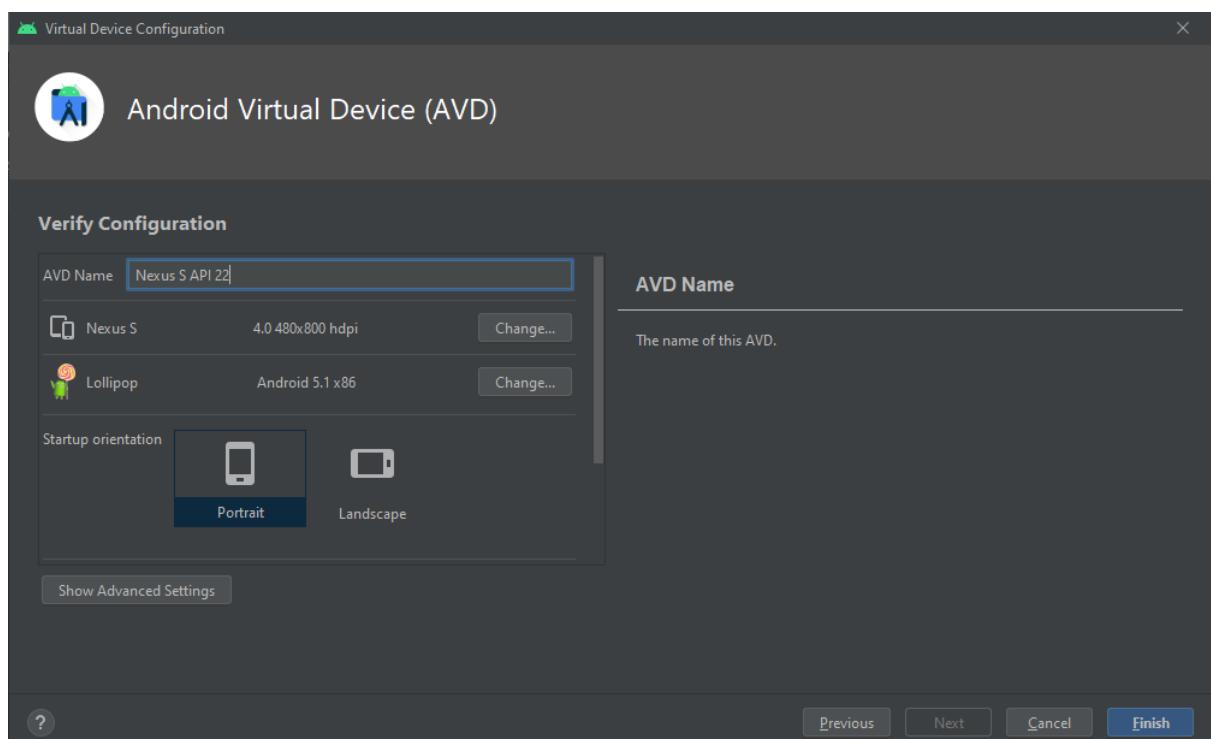
Pilih salah satu device yang dinginkan, misalnya “Nexus S” kemudian klik tombol “Next”



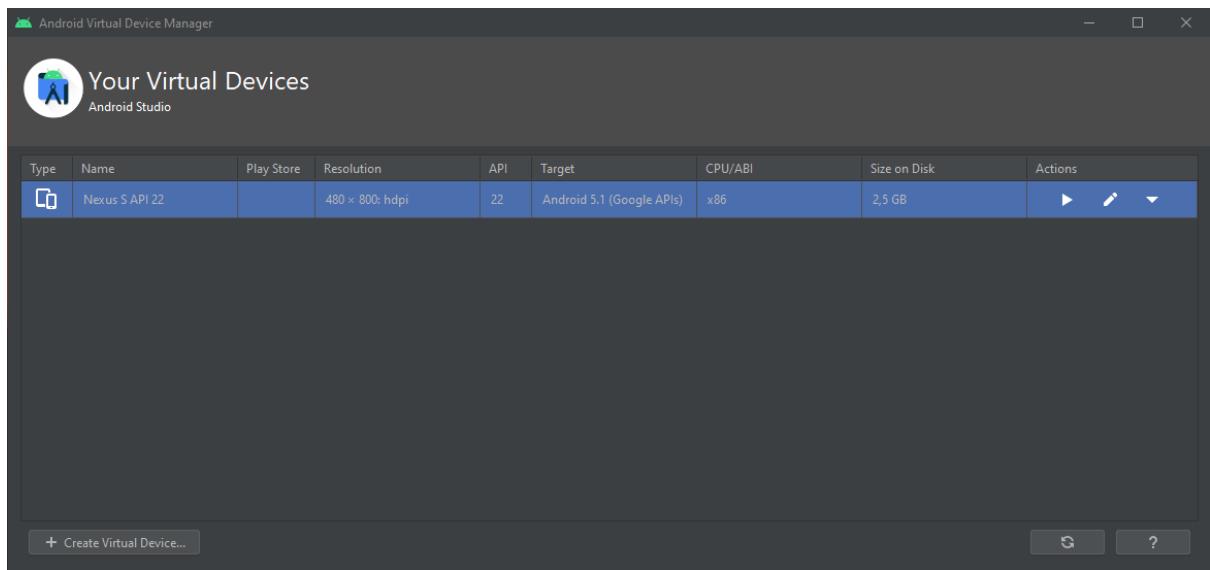
Kemudian kita akan memilih Versi Sistem Operasi Android, dalam hal ini misalnya "Q", jika belum ada maka kita akan diminta untuk mengunduh terlebih dahulu.



Setelah itu klik Finish



Untuk Menjalankan Emulator, klik logo play



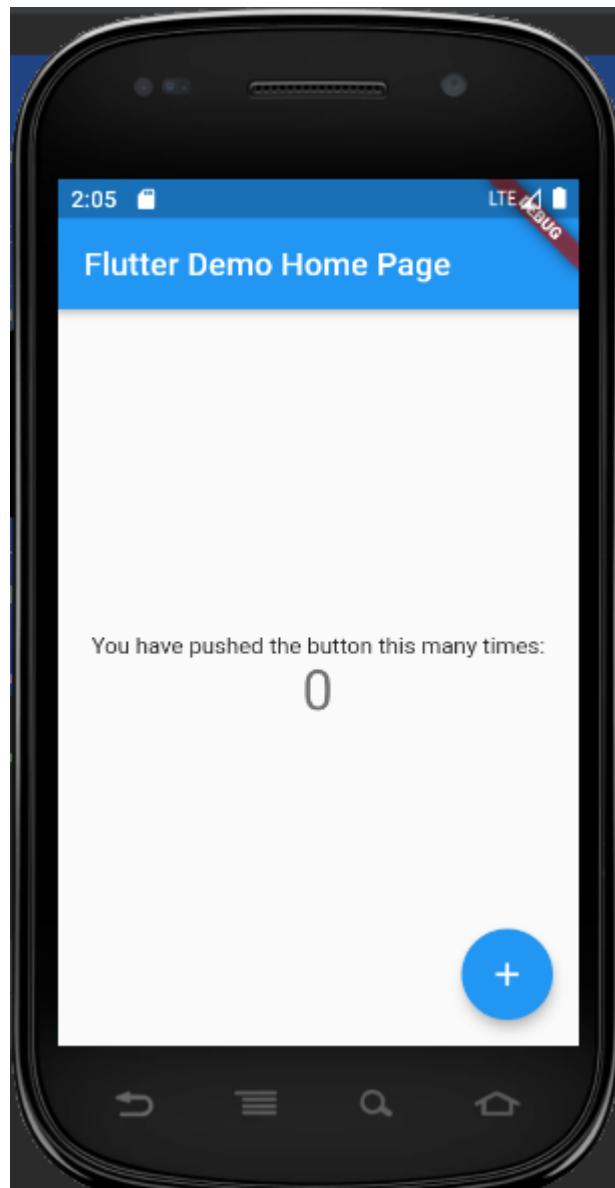
Kemudian akan muncul emulator android seperti berikut:



Jika emulator sudah berjalan, kita dapat mengeksekusi projek dengan cara klik tombol play (berwarna hijau) pada Android Studio



Pada emulator akan tampil hasil projek seperti berikut



TUGAS

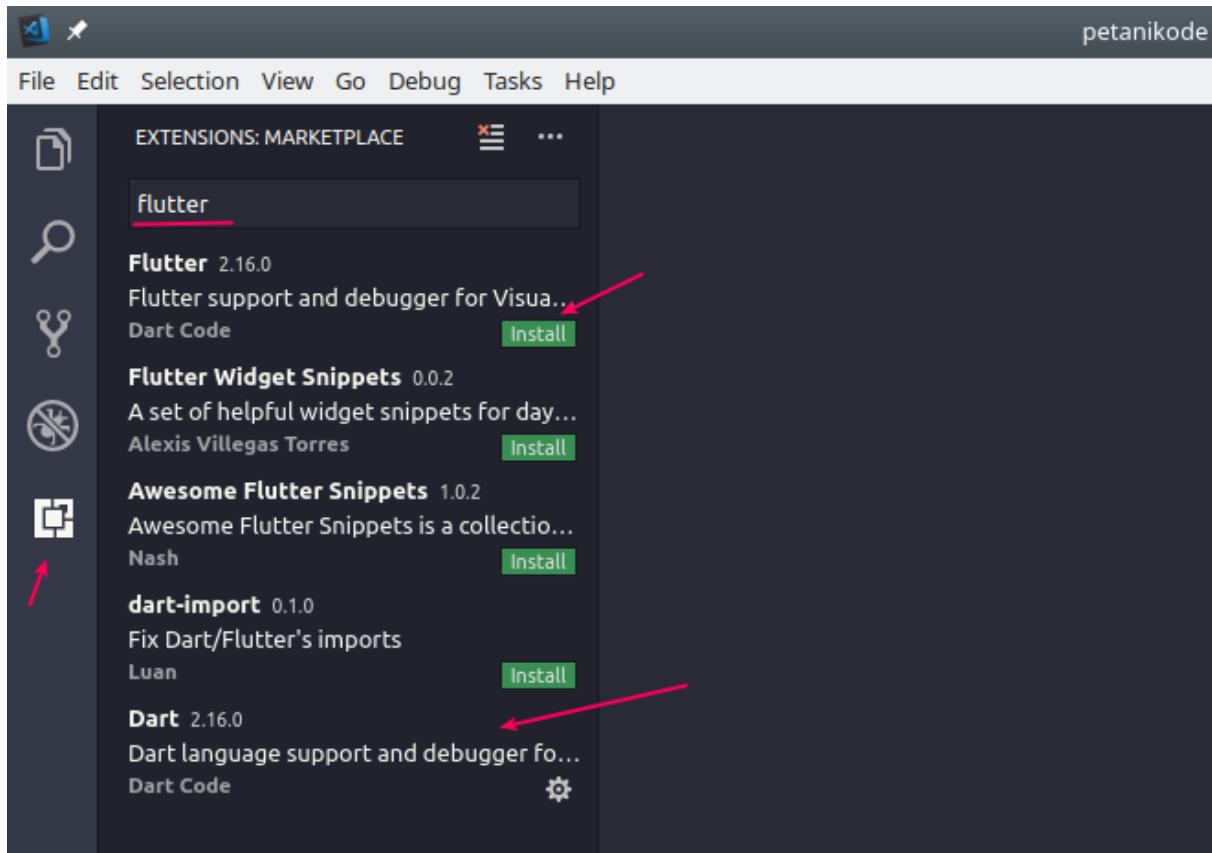
1. Download VSCode pada laman <https://code.visualstudio.com/download>
2. Lakukan installasi VSCode dengan mengikuti petunjuk pada pertemuan 3

PERTEMUAN 3

Membuat dan Menjalankan Projek Dengan VSCode dan Handphone Android

Install VSCode Sebagai Alternatif Editor

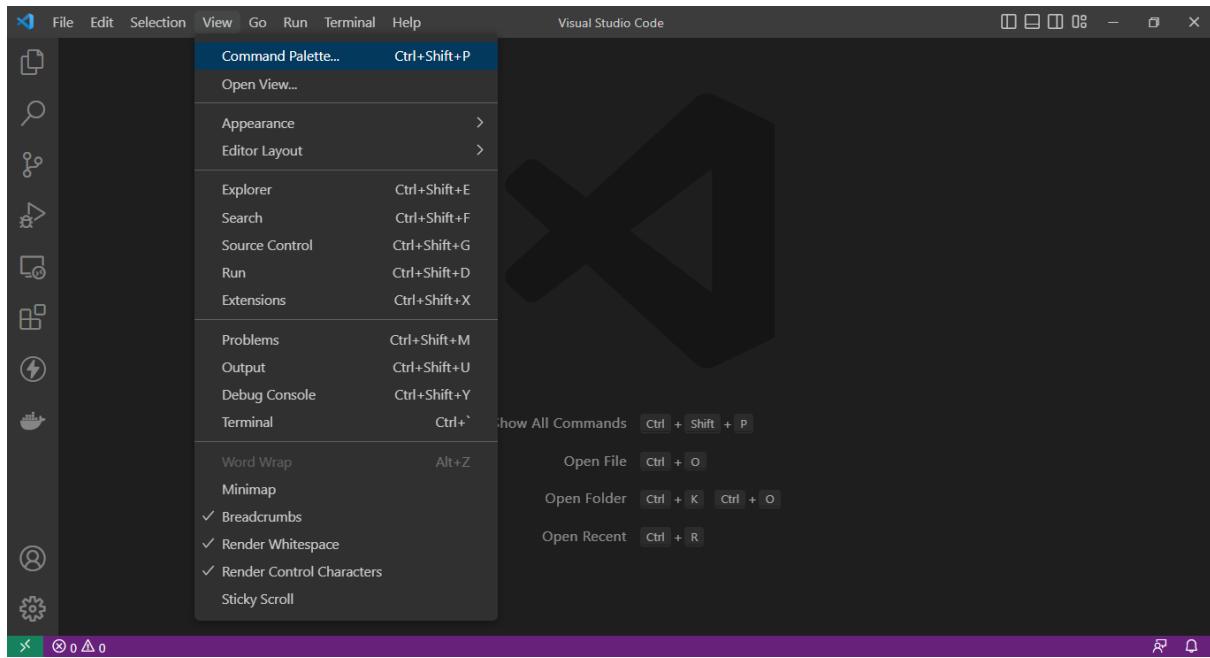
Unduh VSCode pada laman <https://code.visualstudio.com/download> kemudian install. Agar flutter dapat digunakan pada VSCode, perlu diinstall beberapa extension flutter yang dibutuhkan.



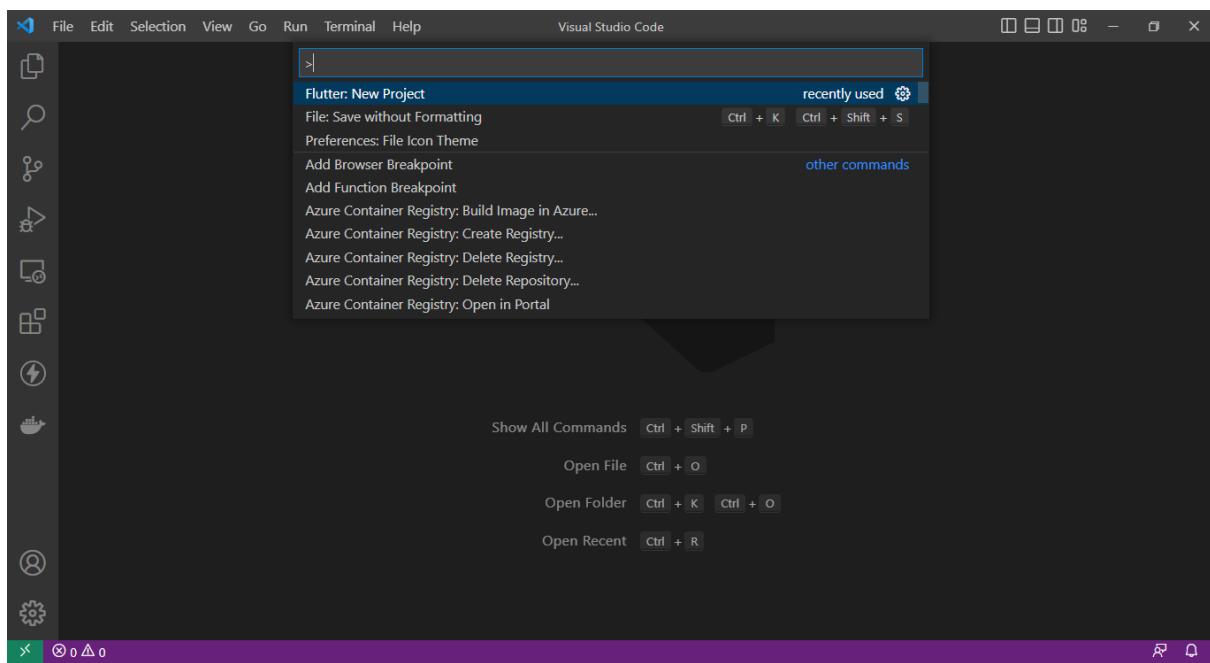
Setelah itu restart/tutup VSCode

Membuat projek flutter dengan VSCode

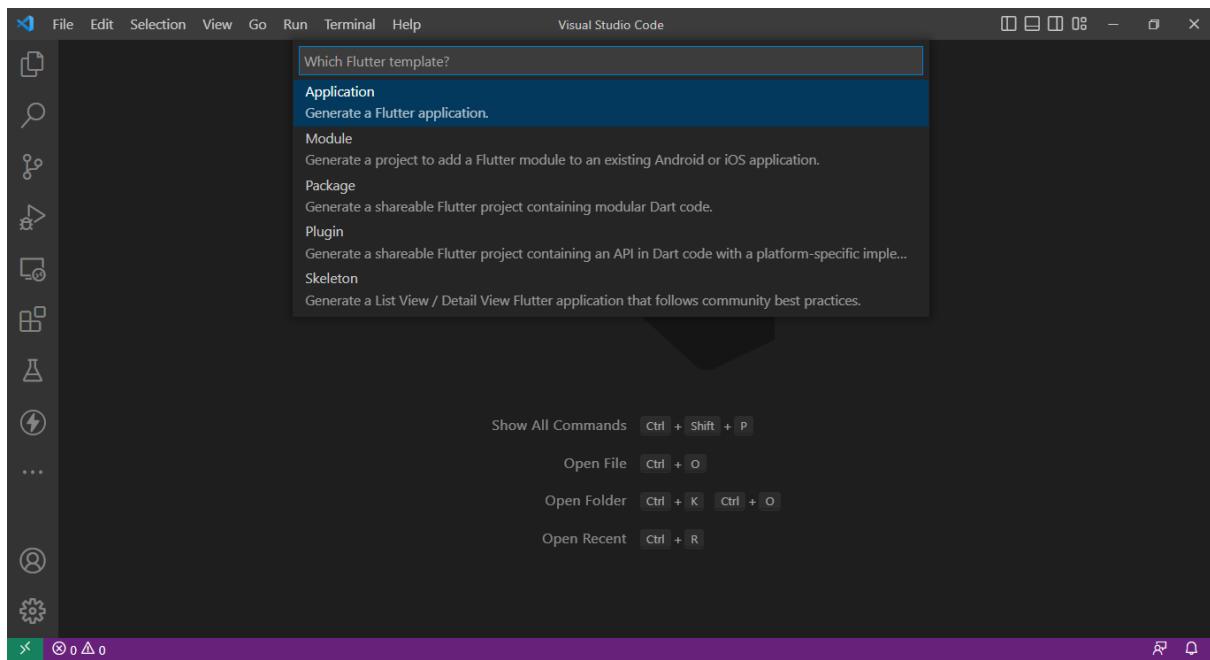
Jalankan VSCode, pada menubar pilih **view -> command Palette...** atau dapat juga dengan shortcut **Ctrl + Shift + P**



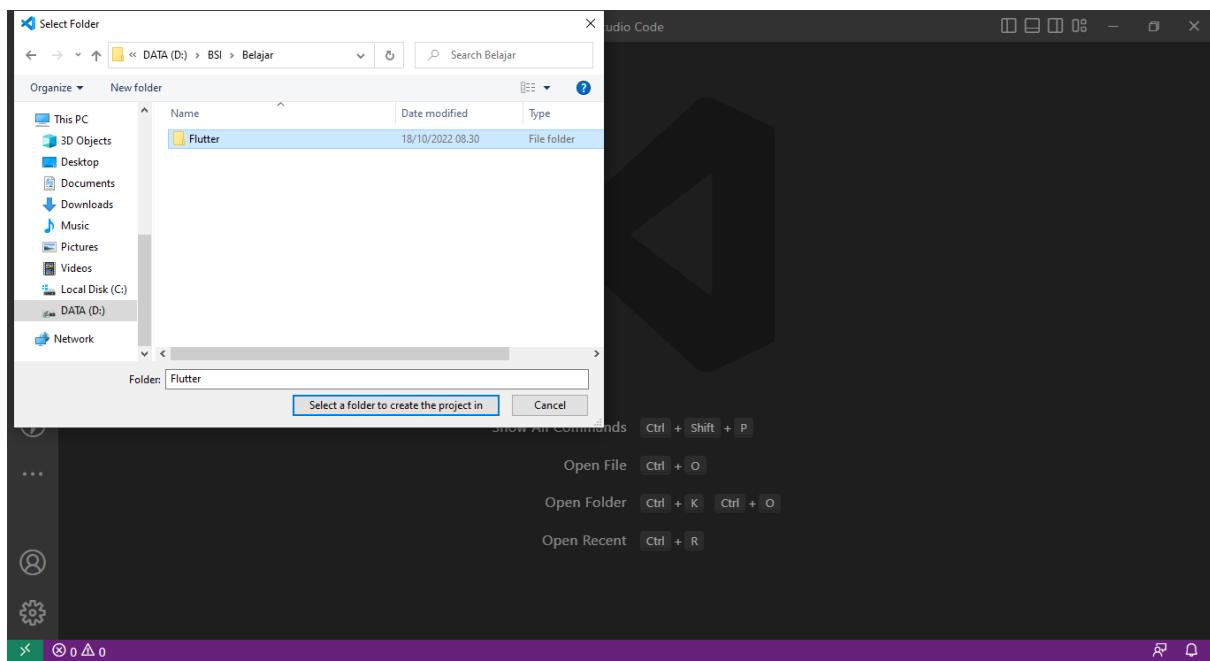
Kemudian ketikkan **flutter** dan pilih **Flutter: New Project**



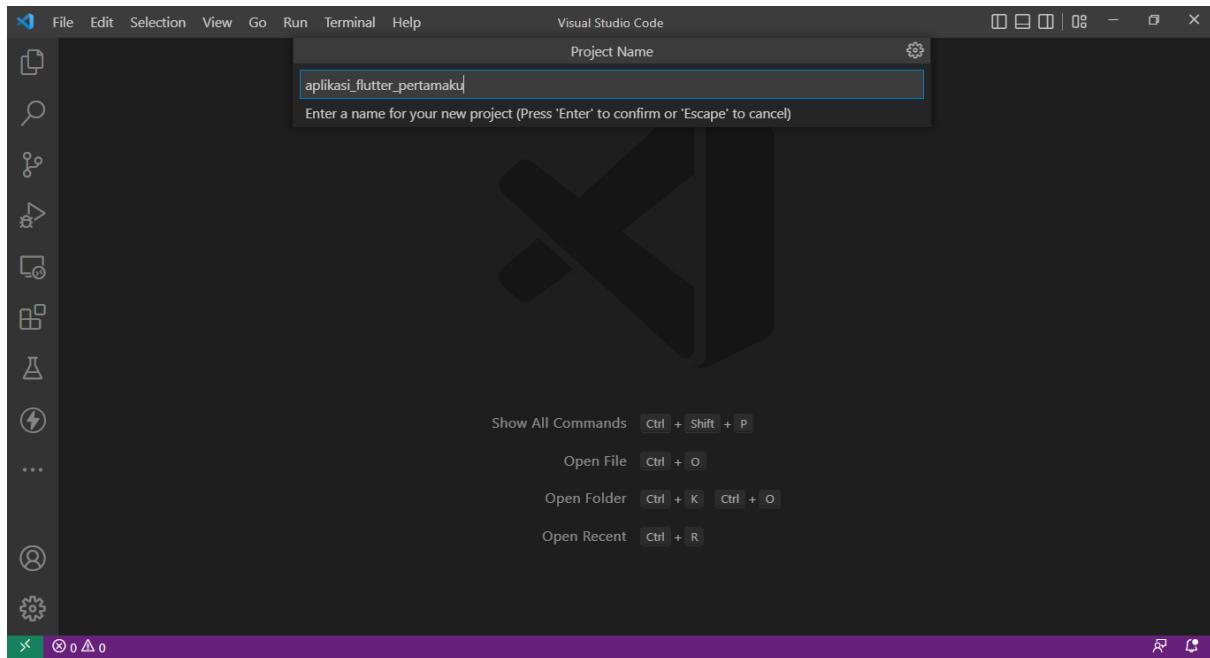
Kemudian pilih **Application**



Pilih folder tempat projek tersebut



Kemudian tentukan nama projek flutter yang ingin dibuat dengan nama **aplikasi_flutter_pertamaku**



Kemudian tekan **Enter** dan tunggu hingga proses unduhan selesai

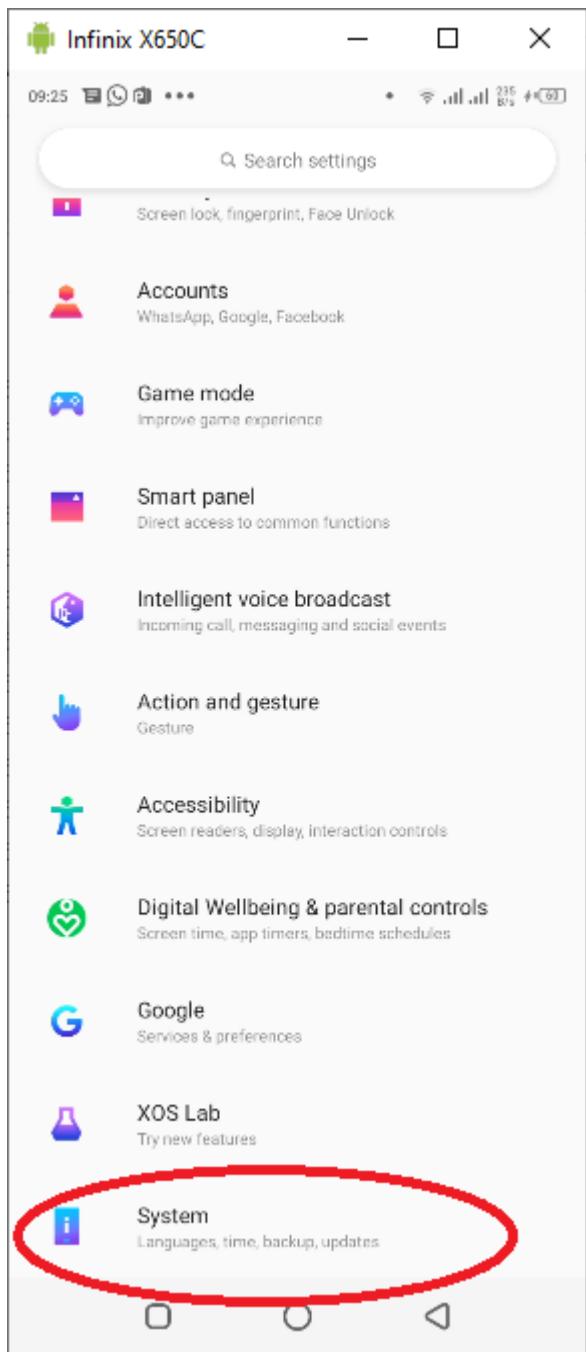
```
EXPLORER          main.dart ×
KLINIK_APP
  > .dart_tool
  > .idea
  > android
  > ios
  > lib
    > main.dart
      main.dart
        lib > main.dart
          1 import 'package:flutter/material.dart';
          2
          3 void main() {
          4   runApp(const MyApp());
          5 }
          ...
          7 class MyApp extends StatelessWidget {
          8   const MyApp({super.key});
          9
          10 // This widget is the root of your application.
          11 @override
          12 Widget build(BuildContext context) {
          13   return MaterialApp(
          14     title: 'Flutter Demo',
          15     theme: ThemeData(
          16       // This is the theme of your application.
          17       //
          18       // Try running your application with "flutter run". You'll see the
          19       // application has a blue toolbar. Then, without quitting the app, try
          20       // changing the primarySwatch below to Colors.green and then invoke
          21       // "hot reload" (press "r" in the console where you ran "flutter run",
          22       // or simply save your changes to "hot reload" in a Flutter IDE).
          23       // Notice that the counter didn't reset back to zero; the application
          24       // is not restarted.
          25       primarySwatch: Colors.blue,
          26     ),
          27     home: const MyHomePage(title: 'Flutter Demo Home Page'),
        
```

The screenshot shows the Visual Studio Code interface with the "main.dart" file open in the editor. The code is the standard Flutter "Hello World" application. The Explorer sidebar on the left shows the project structure, including the "lib" folder which contains the "main.dart" file. The status bar at the bottom indicates the file is being analyzed.

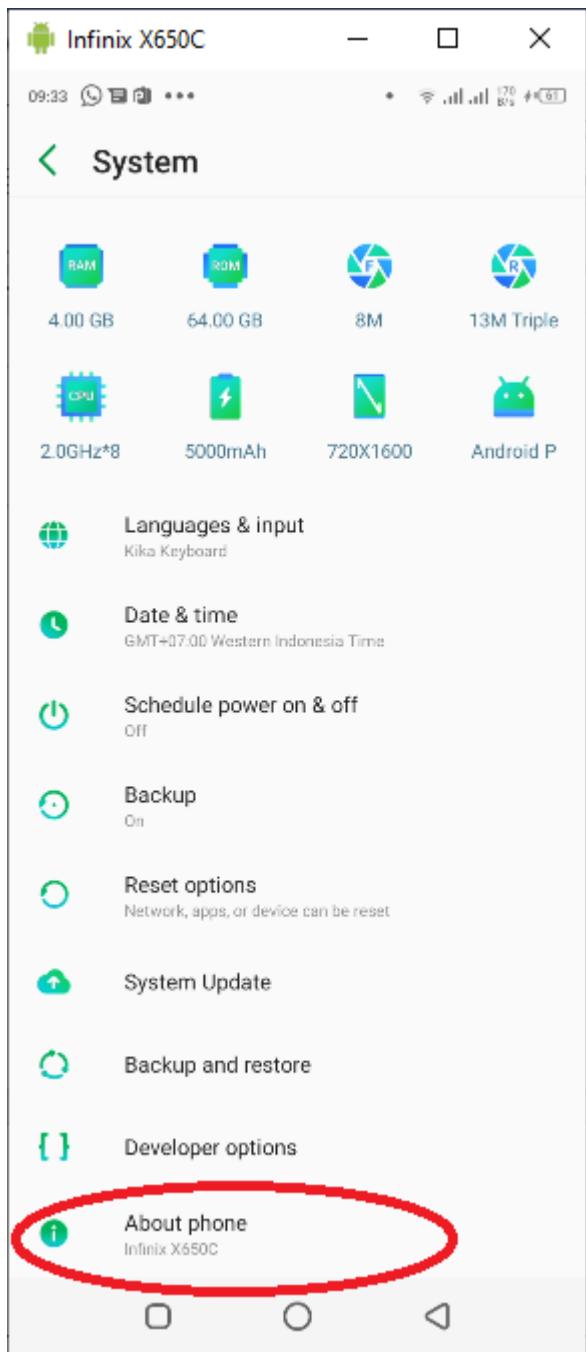
Menjalankan aplikasi dengan Handphone Android

Untuk menjalankan projek flutter dari VSCode dapat menggunakan Emulator AVD yang telah dibuat sebelumnya menggunakan Android Studio ataupun menggunakan Device Handphone Android langsung.

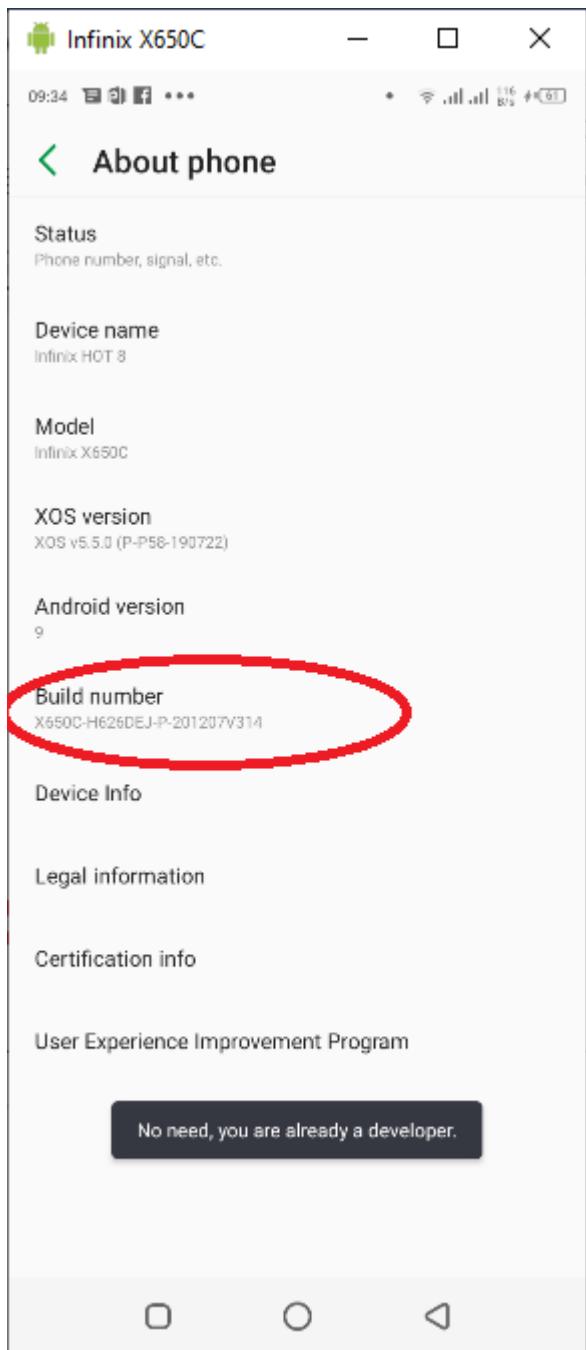
Untuk menggunakan android device secara langsung, pertama aktifkan dulu mode developer dengan cara buka **Setting** kemudian pilih **System** kemudian pilih **About Phone**, untuk masing-masing device mungkin terdapat perbedaan untuk lokasi **About Phone** ada pula yang berada pada **Additional Setting**



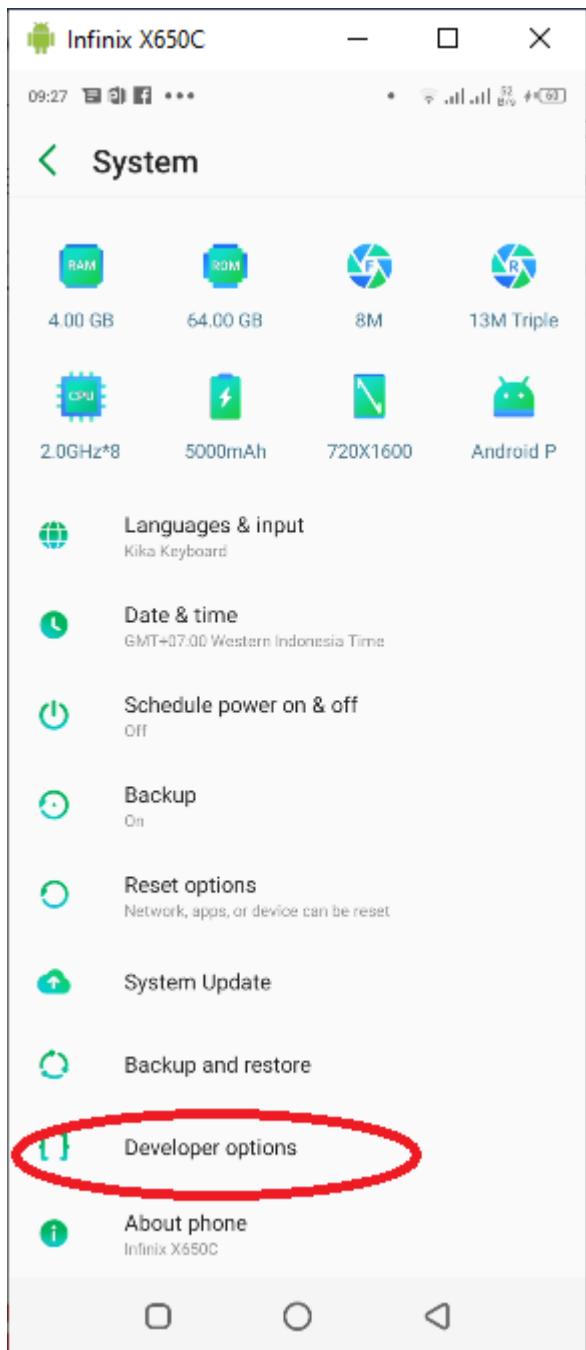
Kemudian pilih **About Phone**



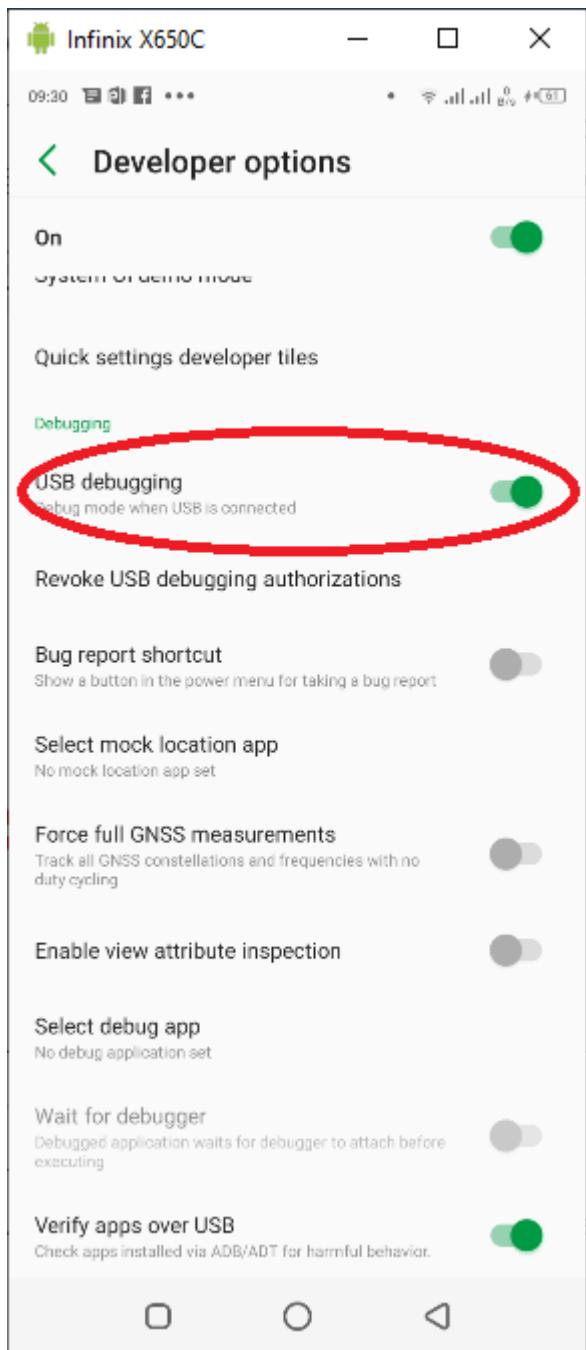
Kemudian ketuk **Build number** beberapa kali, namun ini juga berbeda untuk beberapa versi misalnya untuk Xiaomi dengan mengetuk **MIUI Version** beberapa kali



Selanjutnya mengaktifkan USB Debugger dengan cara pilih **Developer Option** pada **System**, **Developer Option** ini akan muncul setelah mode Developer diaktifkan dengan cara diatas



Kemudian aktifkan **USB Debugging**



Jika telah selesai, hubungkan Handphone android dengan laptop/komputer dengan kabel data, untuk memeriksa apakah sudah terhubung dengan Handphone, dapat dilihat pada VSCode bagian pojok kanan bawah akan tertera nama device yang terhubung

```

File Edit Selection View Go Run Terminal Help
APLIKASI_FLUTTER_PERTAMAKU
main.dart
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Flutter Demo',
13      theme: ThemeData(
14        // This is the theme of your application.
15        //
16        // Try running your application with "flutter run". You'll see the
17        // application has a blue toolbar. Then, without quitting the app, try
18        // changing the primarySwatch below to Colors.green and then invoke
19        // "hot reload" (press "r" in the console where you ran "flutter run",
20        // or simply save your changes to "hot reload" in a Flutter IDE).
21        // Notice that the counter didn't reset back to zero; the application
22        // is not restarted.
23        primarySwatch: Colors.blue,
24        // This makes the visual density adapt to the platform that you run
25        // the app on. For desktop platforms, the controls will be smaller and
26        // closer together (more dense) than on mobile platforms.
27        visualDensity: VisualDensity.adaptivePlatformDensity,
28      ), // ThemeData
29      home: MyHomePage(title: 'Flutter Demo Home Page'),
30    ); // MaterialApp
31  }
32}

```

Ln 12, Col 29 Spaces: 2 UTF-8 CRLF Dart Flutter: 1.22.6 Infinix X650C (android-arm64)

Atau jika pada Android Studio terletak pada toolbar bagian atas tengah

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help belajarflutter - main.dart [belajarflutter] - Android Studio
belajarflutter > lib > main.dart
Project Test
main.dart
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Flutter Demo',
13      theme: ThemeData(
14        // This is the theme of your application.
15        //
16        // Try running your application with "flutter run". You'll see the
17        // application has a blue toolbar. Then, without quitting the app, try
18        // changing the primarySwatch below to Colors.green and then invoke
19        // "hot reload" (press "r" in the console where you ran "flutter run",
20        // or simply save your changes to "hot reload" in a Flutter IDE).
21        // Notice that the counter didn't reset back to zero; the application
22        // is not restarted.
23        primarySwatch: Colors.blue,
24        // This makes the visual density adapt to the platform that you run
25        // the app on. For desktop platforms, the controls will be smaller and
26        // closer together (more dense) than on mobile platforms.
27        visualDensity: VisualDensity.adaptivePlatformDensity,
28      ), // ThemeData
29      home: MyHomePage(title: 'Flutter Demo Home Page'),
30    ); // MaterialApp
31  }
32}

```

Agar laptop bekerja lebih ringan dapat digunakan Text Editor VSCode dan menjalankan projek langsung menggunakan Handphone Android. Untuk menjalankan projek melalui VSCode dengan klik logo play pada bagian pojok kanan atas

```

File Edit Selection View Go Run Terminal Help
main.dart - aplikasi_flutter_pertamaku - Visual Studio Code
EXPLORER main.dart
APLIKASI_FLUTTER_PERTAMAKU
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Flutter Demo',
13      theme: ThemeData(
14        // This is the theme of your application.
15        //
16        // Try running your application with "flutter run". You'll see the
17        // application has a blue toolbar. Then, without quitting the app, try
18        // changing the primarySwatch below to Colors.green and then invoke
19        // "hot reload" (press "r" in the console where you ran "flutter run",
20        // or simply save your changes to "hot reload" in a Flutter IDE).
21        // Notice that the counter didn't reset back to zero; the application
22        // is not restarted.
23        primarySwatch: Colors.blue,
24        //
25        // This makes the visual density adapt to the platform that you run
26        // the app on. For desktop platforms, the controls will be smaller and
27        // closer together (more dense) than on mobile platforms.
28        visualDensity: VisualDensity.adaptivePlatformDensity,
29        ),
30        // ThemeData
31        home: MyHomePage(title: 'Flutter Demo Home Page'),
32      ); // MaterialApp
}

```

Ln 18, Col 76 Spaces:2 UTF-8 CRLF Dart Dart DevTools Flutter:1.22.6 Infinix X650C (android-arm64)

```

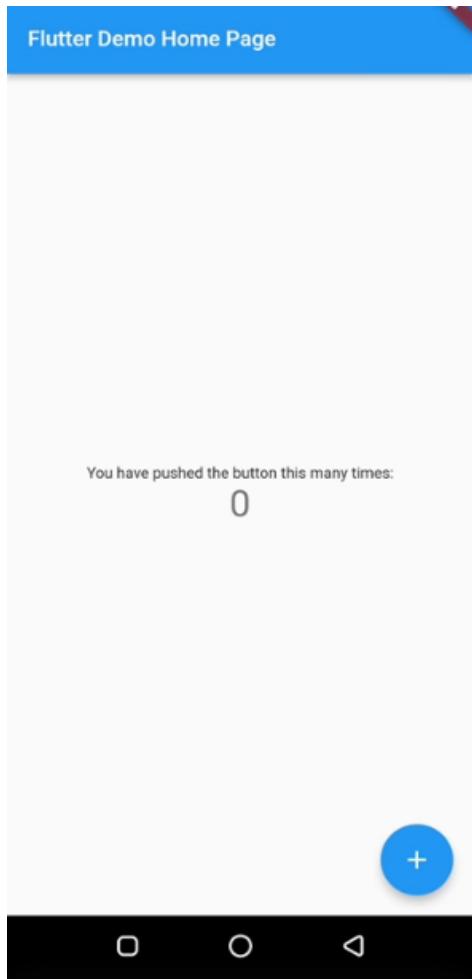
File Edit Selection View Go Run Terminal Help
main.dart - aplikasi_flutter_pertamaku - Visual Studio Code
EXPLORER main.dart
APLIKASI_FLUTTER_PERTAMAKU
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Flutter Demo',
13      theme: ThemeData(
14        // This is the theme of your application.
15        //
16        // Try running your application with "flutter run". You'll see the
17        // application has a blue toolbar. Then, without quitting the app, try
18        // changing the primarySwatch below to Colors.green and then invoke
19        // "hot reload" (press "r" in the console where you ran "flutter run",
20        // or simply save your changes to "hot reload" in a Flutter IDE).
21        // Notice that the counter didn't reset back to zero; the application
22        // is not restarted.
23        primarySwatch: Colors.blue,
24        //
25        // This makes the visual density adapt to the platform that you run
26        // the app on. For desktop platforms, the controls will be smaller and
27        // closer together (more dense) than on mobile platforms.
28        visualDensity: VisualDensity.adaptivePlatformDensity,
29        ),
30        // ThemeData
31        home: MyHomePage(title: 'Flutter Demo Home Page'),
32      ); // MaterialApp
}

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE
Launching lib/main.dart on Infinix X650C in debug mode...
Built build/app/outputs/flutter-apk/app-debug.apk.
Connecting to VM Service at ws://127.0.0.1:59054/EeZyoXtxGj4=/ws
lib/main.dart:1

```

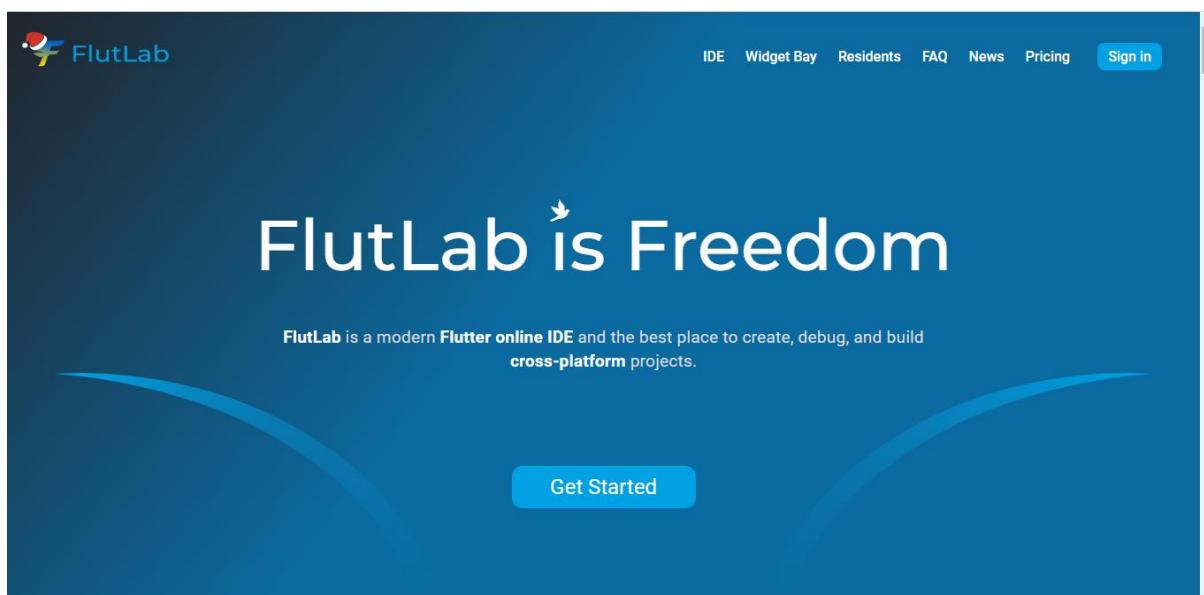
Ln 17, Col 60 Spaces:2 UTF-8 CRLF Dart Dart DevTools Flutter:1.22.6 Infinix X650C (android-arm64)

Adapun tampilannya adalah sebagai berikut

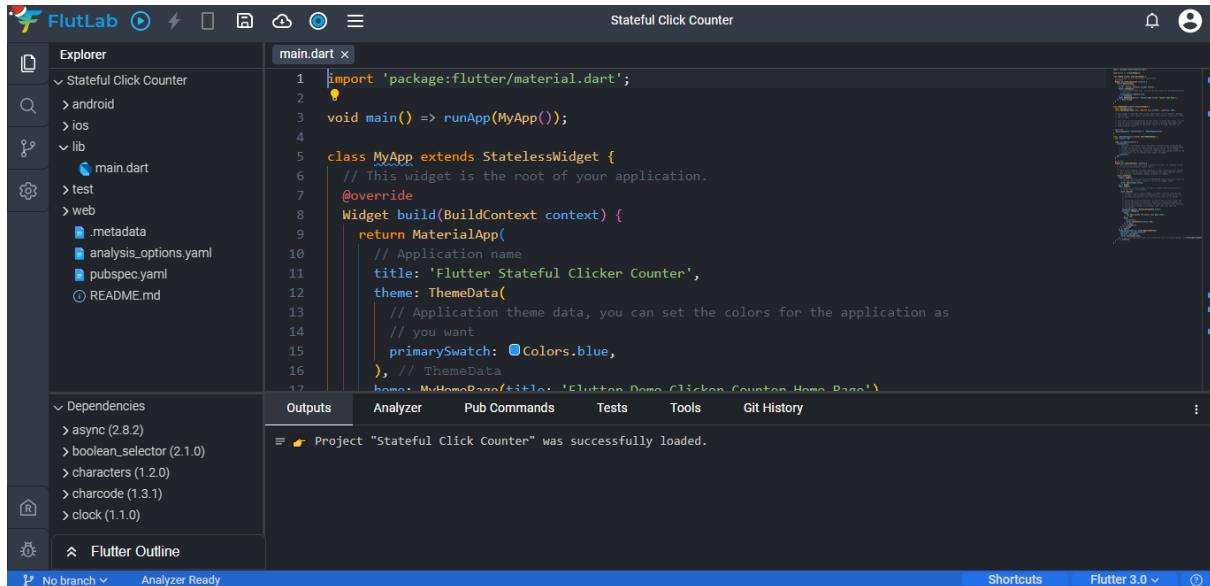


Membuat dan menjalankan Projek dengan flutlab.io dan Web Emulator

Selain menggunakan Android Studio dan VS Code, untuk melakukan pengkodingan flutter dapat pula menggunakan editor online yaitu **flutlab.io**. Namun untuk menggunakan flutlab.io ini memerlukan akses internet



Silahkan buka laman flutlab.io kemudian klik **Get Started** maka akan muncul laman dengan template awal code flutter



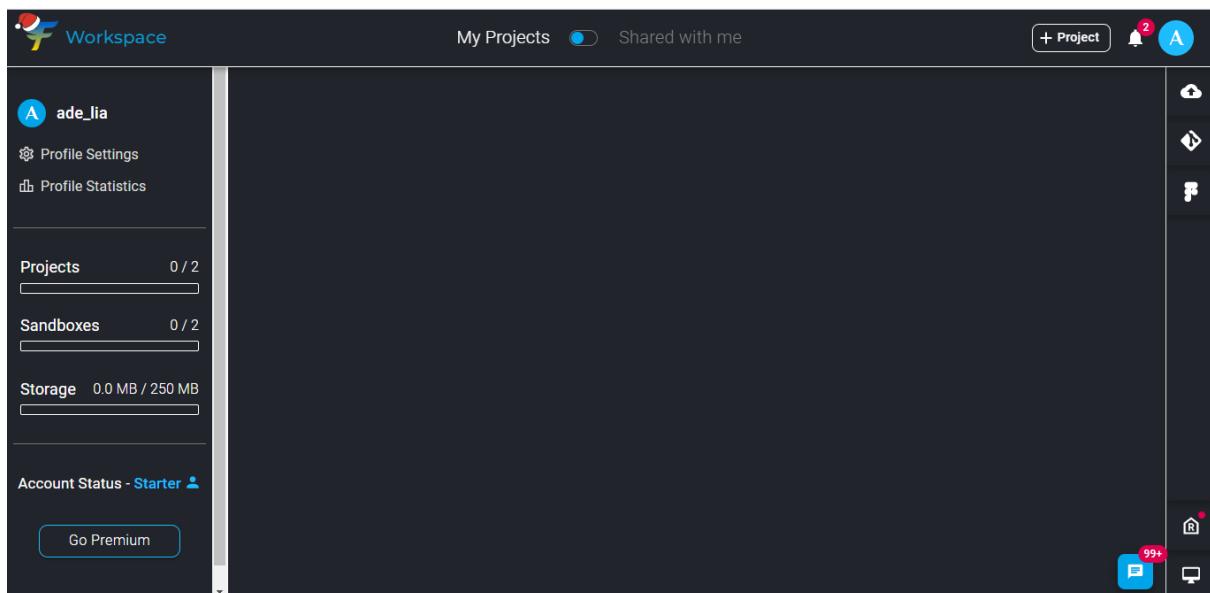
The screenshot shows the FlutLab interface with the project "Stateful Click Counter" open. The main.dart file is displayed in the code editor, containing the following code:

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   // This widget is the root of your application.
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       // Application name
11       title: 'Flutter Stateful Clicker Counter',
12       theme: ThemeData(
13         // Application theme data, you can set the colors for the application as
14         // you want
15         primarySwatch: Colors.blue,
16       ), // ThemeData
17 }
```

The code defines a StatelessWidget named MyApp that returns a MaterialApp with a title of 'Flutter Stateful Clicker Counter' and a blue primarySwatch. The project dependencies listed in pubspec.yaml include async, boolean_selector, characters, charcode, clock, and http.

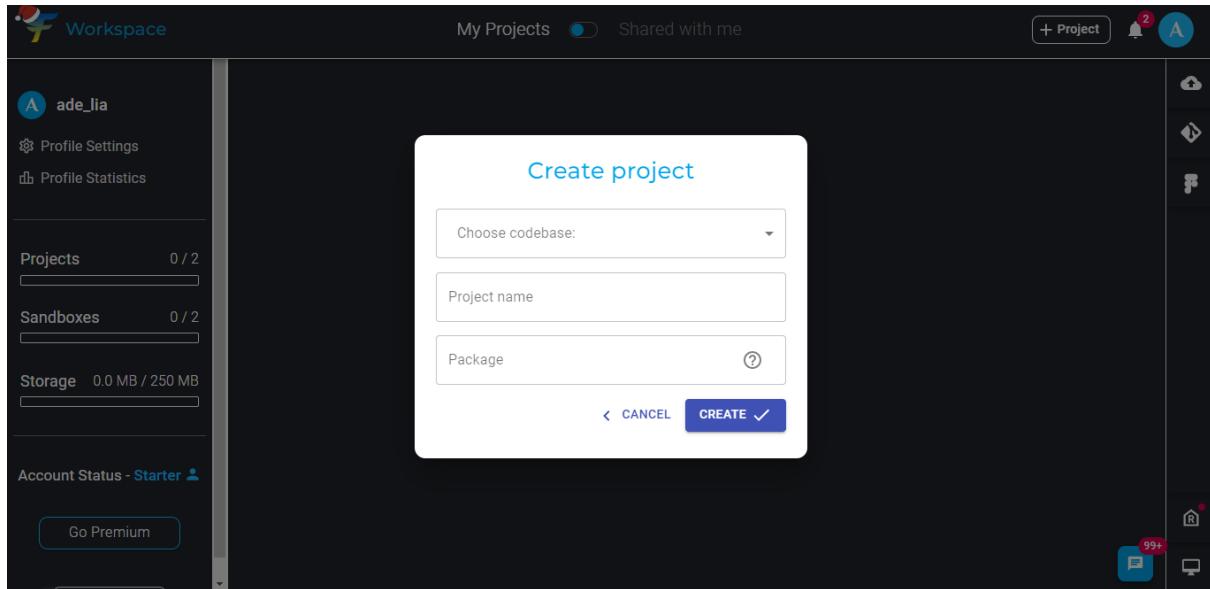
Agar aplikasi yang dibuat dapat tersimpan, lakukan login dengan menekan tombol dengan logo user pada pojok kanan atas, kemudian kita dapat login dengan dua cara, yaitu login dengan google ataupun membuat akun tersendiri pada flutlab.io

Setelah login pada flutlab.io maka akan tampil halaman workspace



Membuat dan Menjalankan Projek di flutlab.io

Klik tombol **+ Project** yang terdapat pada pojok kanan atas

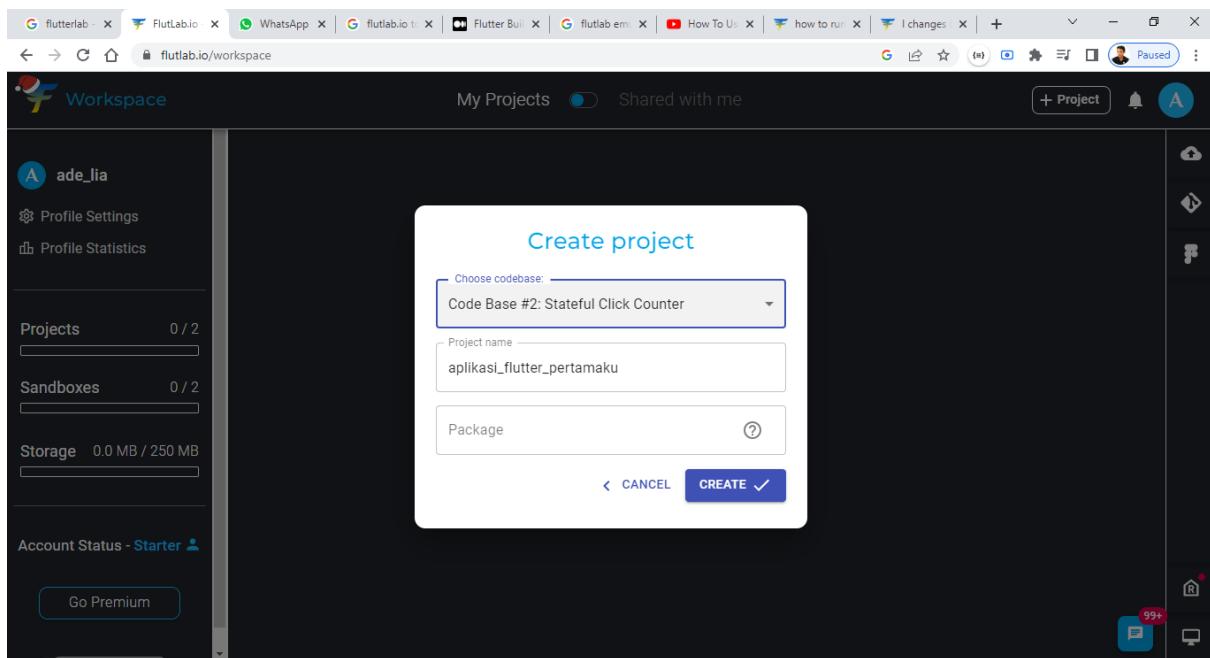


Pada form tersebut isikan dengan data berikut:

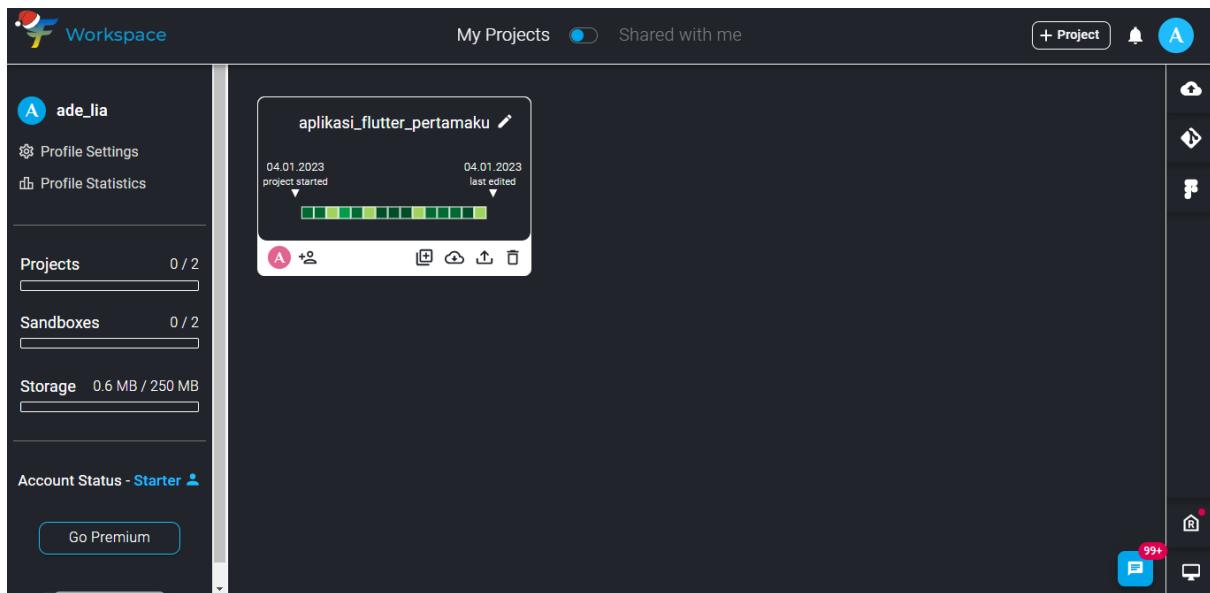
Codebase : Code Base #2: Stateful Click Counter

Projek Name : aplikasi_flutter_pertamaku

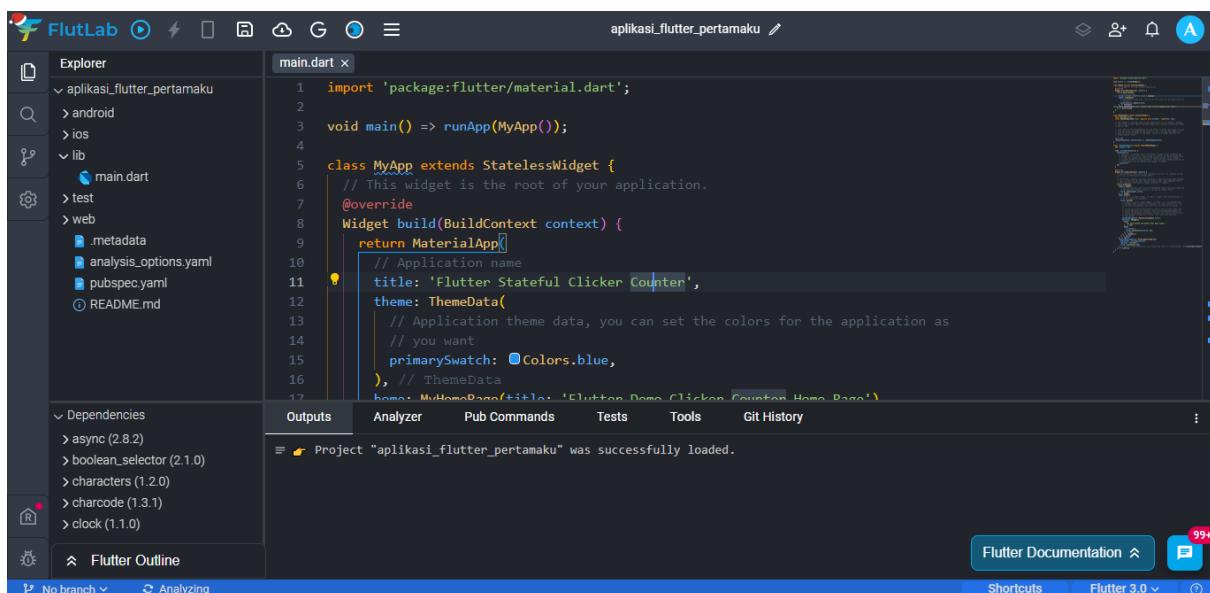
Untuk bagian **Package** dapat diabaikan dan akan mengambil nama package default yaitu **com.codegemz.flutlab**



Kemudian klik tombol **CREATE**



Kemudian kita akan diarahkan ke halaman workspace dengan projek yang telah dibuat sebelumnya. Untuk membuka projek, klik projek yang telah dibuat, maka akan muncul code projek tersebut



Untuk menjalankan projek, klik tombol (pastikan opsi yang terpilih adalah **web**) yang terdapat di pojok kiri atas, tunggu hingga proses build selesai.

The screenshot shows the FlutLab interface for a Flutter web project named "aplikasi_flutter_pertamaku". The main code editor displays the `main.dart` file with the following code:

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   // This widget is the root of your application.
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       // Application name
11       title: 'Flutter Stateful Clicker Counter',
12       theme: ThemeData(
13         // Application theme data, you can set the colors for the application
14         // you want
15         primarySwatch: Colors.blue,
16       ), // ThemeData
17     );
18   }
19 }
```

A yellow banner at the top right says "WORKING ON IT". Below it, another message says "It's your turn to build" and "Building the web archive in progress". A progress bar is shown below the message. A note at the bottom of the code editor says "Sit back and think about changes you introduced. You have a minute for meditation. What can be improved?".

The left sidebar shows the project structure with files like `main.dart`, `pubspec.yaml`, and `README.md`. The bottom status bar indicates "Build started by ade_jia. Target: web".

Setelah selesai, akan muncul tampilan seperti berikut

The screenshot shows the FlutLab interface with the same project structure and code as the previous screenshot. The code editor now shows the updated `main.dart` file with the addition of a button and counter logic:

```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   // This widget is the root of your application.
7   @override
8   Widget build(BuildContext context) {
9     return MaterialApp(
10       // Application name
11       title: 'Flutter Stateful Clicker Counter',
12       theme: ThemeData(
13         // Application theme data, you can set the colors for the application
14         // you want
15         primarySwatch: Colors.blue,
16       ), // ThemeData
17       home: MyHomePage(),
18     );
19   }
20 }
```

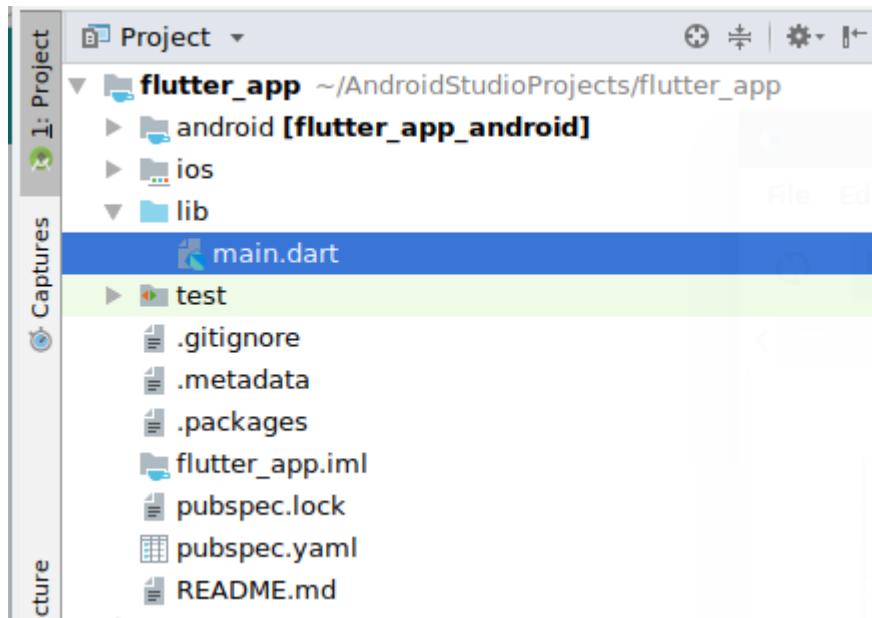
To the right of the code editor, a mobile phone simulator displays the running application. The screen shows a button labeled "+". Below the button, a text field displays "0". Above the text field, a message says "You have pushed the button this many times:". The bottom status bar shows "Flutter Demo Clicker Counter Home Page".

Setelah melakukan perubahan pada kodingan, kita dapat menyimpan pembaruan dengan shortcut **CTRL + S**

PERTEMUAN 4

Struktur Folder Flutter

Adapun struktur folder Projek flutter adalah sebagai berikut:



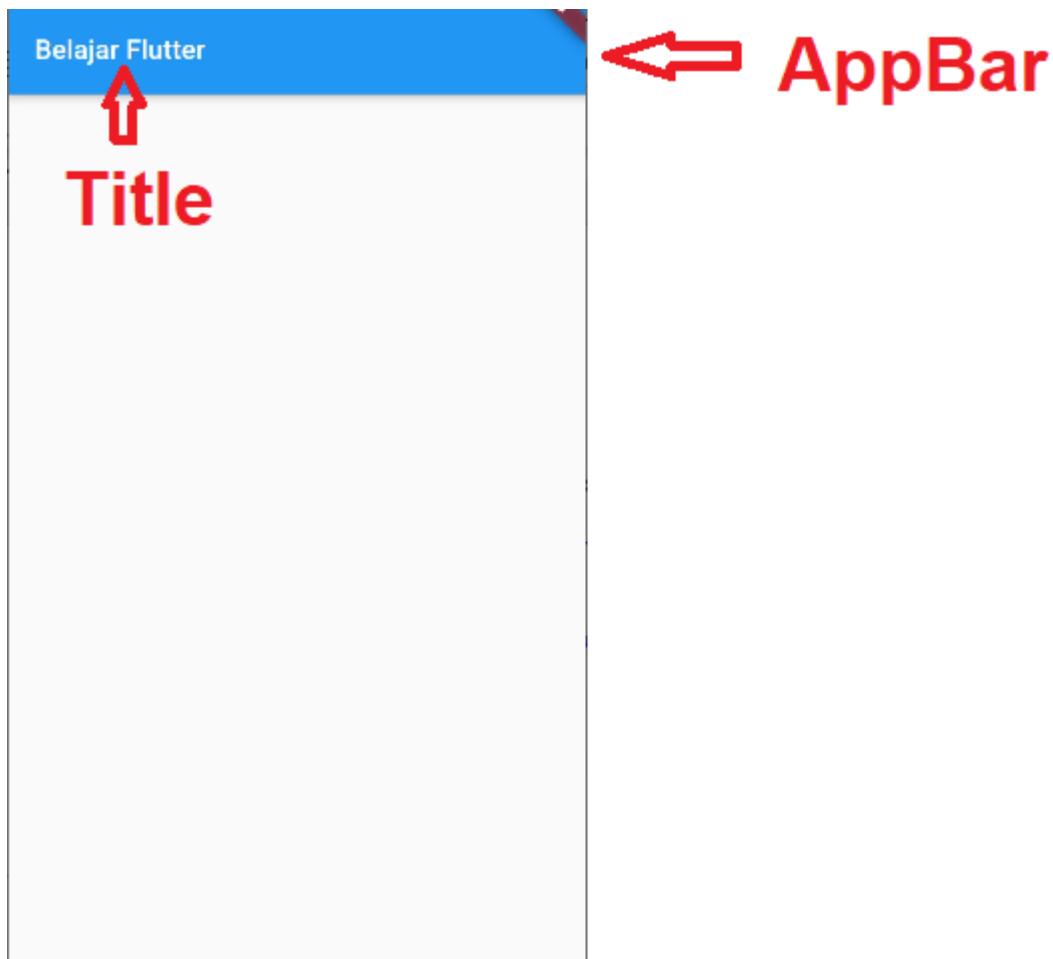
- ② **android** berisi source code untuk aplikasi android;
- ② **ios** berisi source code untuk aplikasi iOS;
- ② **lib** berisi source code Dart, di sini kita akan menulis kode aplikasi;
- ② **test** berisi source code Dart untuk testing aplikasi;
- ② **.gitignore** adalah file [Git](#);
- ② **.metadata** merupakan file yang berisi metadata project yang di-generate otomatis;
- ② **.packages** merupakan file yang berisi alamat path package yang dibuat oleh pub;
- ② **flutter_app.iml** merupakan file XML yang berisi keterangan project;
- ② **pubspec.lock** merupakan file yang berisi versi-versi library atau package. File ini dibuat oleh pub. Fungsinya untuk mengunci versi package.
- ② **pubspec.yaml** merupakan file yang berisi informasi tentang project dan library yang dibutuhkan;
- ② **README.md** merupakan file markdown yang berisi penjelasan tentang source code.

Membuat Hello World

Buka projek aplikasi_flutter_pertamaku menggunakan VSCode atau dapat pula menggunakan **flutlab.io**. Buka file main.dart yang terletak pada folder lib kemudian ubah menjadi

```
1. import 'package:flutter/material.dart';
2.
3. void main() {
4.   runApp(const MyApp());
5. }
6.
7. class MyApp extends StatelessWidget {
8.   const MyApp({Key? key}) : super(key: key);
9.
10. @override
11. Widget build(BuildContext context) {
12.   return MaterialApp(
13.     title: 'Klinik',
14.     home: Scaffold(
15.       appBar: AppBar(
16.         title: const Text("Belajar Flutter"),
17.       ),
18.     ),
19.   );
20. }
21. }
```

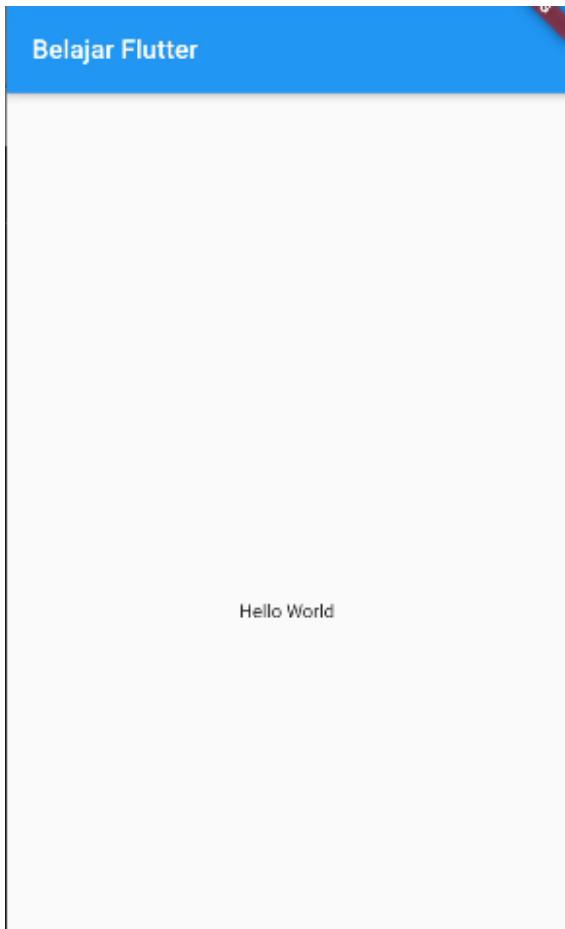
Ketika dijalankan akan menghasilkan



Kemudian untuk menambahkan tampilan dibagian dalam (body), pada fungsi Scaffold terdapat parameter **body**. Silahkan modifikasi **main.dart** menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2.
3. void main() {
4.   runApp(const MyApp());
5. }
6.
7. class MyApp extends StatelessWidget {
8.   const MyApp({Key? key}) : super(key: key);
9.
10. @override
11. Widget build(BuildContext context) {
12.   return MaterialApp(
13.     title: 'Klinik',
14.     home: Scaffold(
15.       appBar: AppBar(
16.         title: const Text("Belajar Flutter"),
17.       ),
18.       body: const Center(
19.         child: Text("Hello World"),
20.       ),
21.     ),
22.   );
23. }
24. }
```

Sehingga akan menghasilkan



Pada aplikasi di atas, kita membuat **StatelessWidget** yang berisi widget **MaterialApp()**. Kemudian di dalam **MaterialApp()** berisi widget lagi: **Scaffold**, **AppBar**, **Center**, dan **Text**.

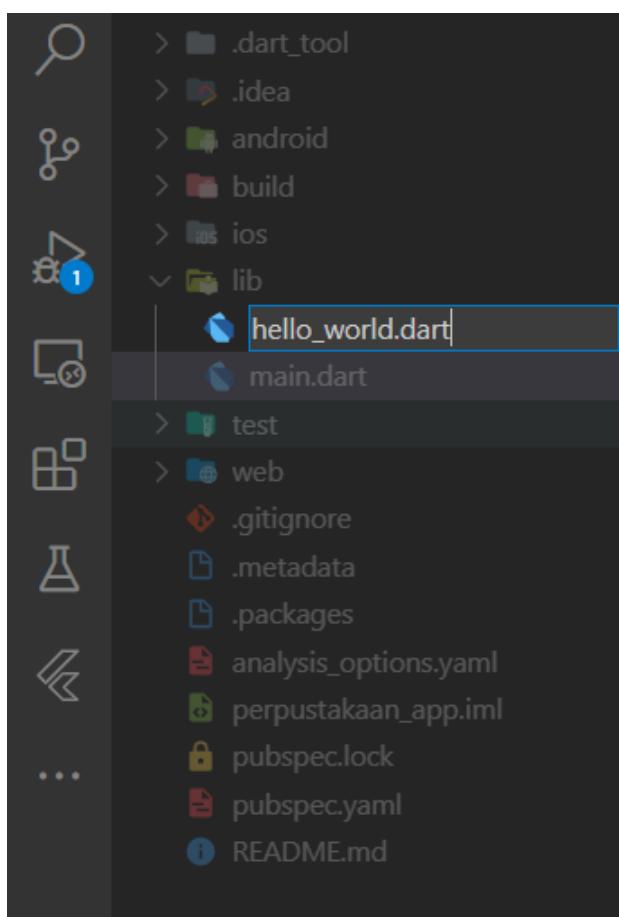
Ini adalah widget dasar.

Penjelasan:

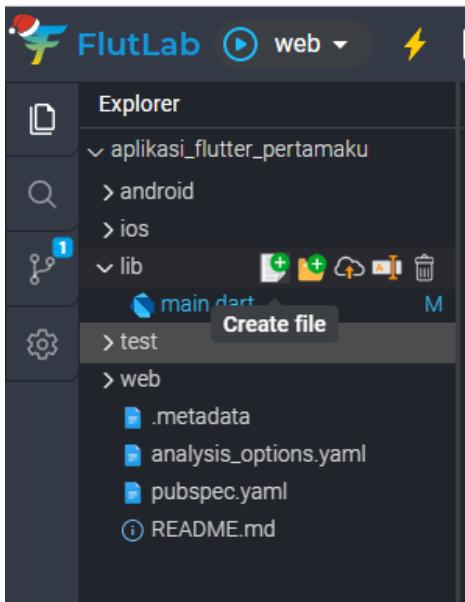
- MyApp adalah StatelessWidget, merupakan widget induk;
- MaterialApp adalah widget yang membungkus beberapa widget yang menggunakan tema material design
- Scaffold adalah widget untuk struktur dasar material design;
- AppBar adalah widget untuk membuat AppBar;
- Center adalah Widget layout untuk membuat widget ke tengah;
- Text adalah widget untuk membuat teks.

Untuk mempermudah dalam pembacaan kode dan maintenance dapat dilakukan dengan memisahkan **MyApp** dengan halaman yang ingin ditampilkan.

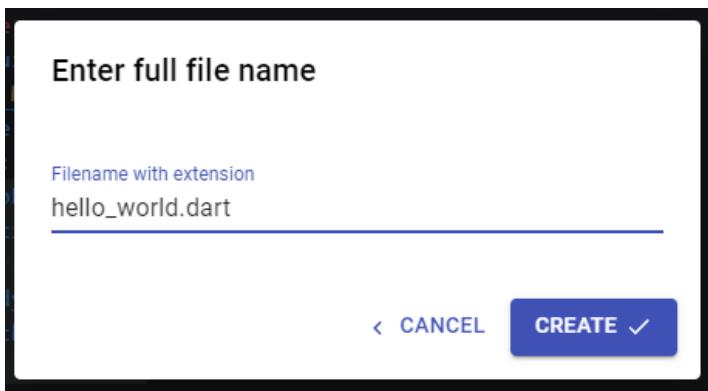
Silahkan buat sebuah file dengan nama **hello_world.dart** di dalam folder **lib**



Jika menggunakan flutlab.io arahkan cursor pada folder lib, kemudian pilih **Create File**



Kemudian ketikkan nama file yang ingin dibuat yaitu hello_world.dart



Kemudian bagian **Scaffold** pada **main.dart** yang telah dibuat tadi akan kita masukkan ke dalam **hello_world.dart**, sehingga pada **hello_world.dart** akan menjadi

```
1. import 'package:flutter/material.dart';
2.
3. class HelloWorld extends StatelessWidget {
4.   const HelloWorld({Key? key}) : super(key: key);
5.
6.   @override
7.   Widget build(BuildContext context) {
8.     return Scaffold(
9.       appBar: AppBar(
10.         title: const Text("Belajar Flutter"),
11.       ),
12.       body: const Center(
13.         child: Text("Hello World"),
14.       ),
15.     );
16.   }
17. }
```

Pada file **main.dart** kita modifikasi kembali pada bagian **home** menjadi

```
1. import 'package:flutter/material.dart';
2. import 'hello_world.dart';
3.
4. void main() {
5.   runApp(const MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   const MyApp({Key? key}) : super(key: key);
10.
11. @override
12. Widget build(BuildContext context) {
13.   return const MaterialApp(
14.     title: 'Klinik',
15.     home: HelloWorld(),
16.   );
17. }
18. }
```

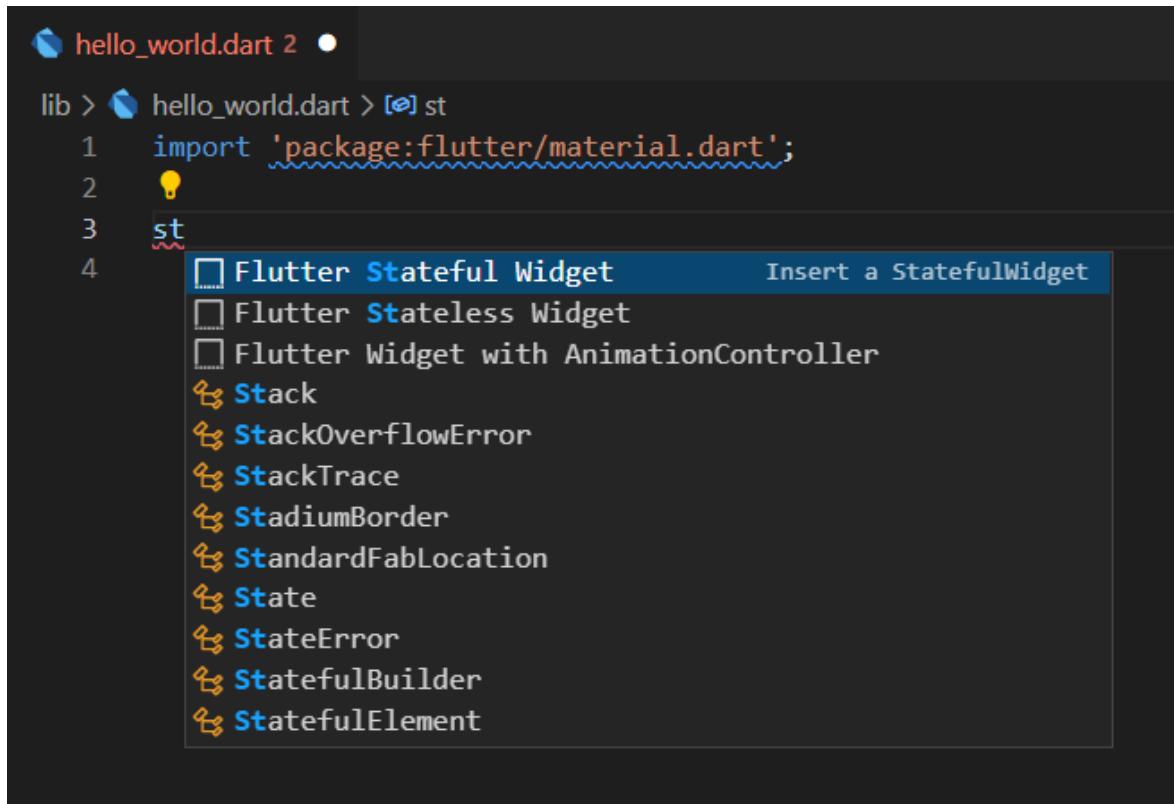
Pada bagian **home**, kita memanggil class **HelloWorld** yang telah kita buat sebelumnya pada file **hello_world.dart**

Jika kita perhatikan pada bagian body, terdapat Widget Center kemudian didalam Widget Center tersebut terdapat parameter child untuk meletakkan Widget lain didalam widget tersebut, dalam hal ini adalah Widget Text

```
Center(
  child: Text('Hello World'),
),
```

Catatan : dalam Widget selain **child**, terdapat pula **children** dengan type data array yang dimana kita dapat menempatkan beberapa Widget didalamnya contohnya pada Widget Column dan Row

Untuk mempercepat dalam pembuatan class pada VSCode dapat dilakukan dengan mengetik **st** kemudian memilih **stateless widget** ataupun **stateful widget** kemudian ketikkan nama class yang diinginkan

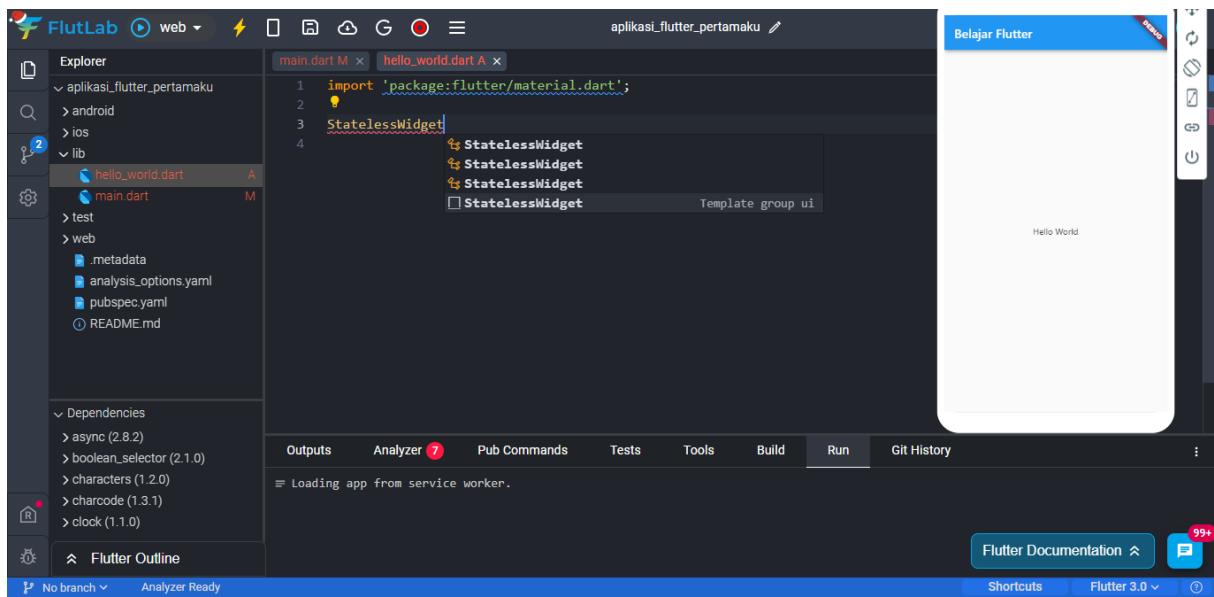


A screenshot of a code editor showing a dropdown menu. The menu is triggered by the letter 'st' in the code. The dropdown title is 'Flutter StatefulWidget' with the subtitle 'Insert a StatefulWidget'. Other items in the dropdown include: Flutter StatelessWidget, Flutter Widget with AnimationController, Stack, StackOverflowError, StackTrace, StadiumBorder, StandardFabLocation, State, StateError, StatefulWidgetBuilder, and StatefulWidgetElement.

```
lib > hello_world.dart > st
1 import 'package:flutter/material.dart';
2
3 st
4
```

- Flutter StatefulWidget Insert a StatefulWidget
- Flutter StatelessWidget
- Flutter Widget with AnimationController
- Stack
- StackOverflowError
- StackTrace
- StadiumBorder
- StandardFabLocation
- State
- StateError
- StatefulWidgetBuilder
- StatefulWidgetElement

Atau jika menggunakan flutlab.io dengan mengetikkan **State** kemudian memilih stateless widget ataupun stateful widget dengan logo persegi dibagian kiri, kemudian ketikkan nama class yang diinginkan



Membuat Widget Column

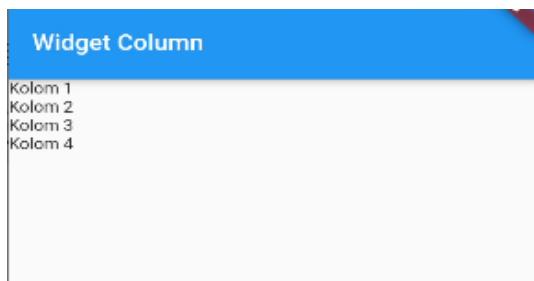
Buat sebuah file dengan nama **column_widget.dart** didalam folder lib, kemudian ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class ColumnWidget extends StatelessWidget {
4.   const ColumnWidget({Key? key}) : super(key: key);
5.
6.   @override
7.   Widget build(BuildContext context) {
8.     return Scaffold(
9.       appBar: AppBar(
10.         title: const Text("Widget Column"),
11.       ),
12.       body: Column(
13.         children: const [
14.           Text("Kolom 1"),
15.           Text("Kolom 2"),
16.           Text("Kolom 3"),
17.           Text("Kolom 4")
18.         ],
19.       ),
20.     );
21.   }
22. }
```

Kemudian pada file **main.dart** ubah kodennya menjadi

```
1. import 'package:flutter/material.dart';
2. import 'column_widget.dart';
3.
4. void main() {
5.   runApp(const MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   const MyApp({Key? key}) : super(key: key);
10.
11.   @override
12.   Widget build(BuildContext context) {
13.     return const MaterialApp(
14.       title: 'Klinik',
15.       home: ColumnWidget(),
16.     );
17.   }
18. }
```

Dan hasilnya akan menjadi seperti berikut



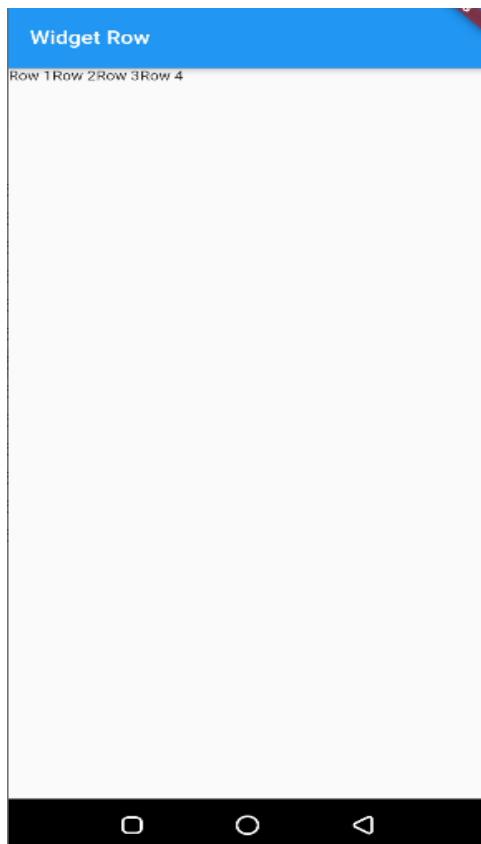
Column biasanya digunakan untuk membuat Form

Membuat Widget Row

Untuk menampilkan Widget dalam posisi horizontal dapat menggunakan Widget Row. Buat sebuah file didalam folder **lib** dengan nama **row_widget.dart**, kemudian ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class RowWidget extends StatelessWidget {
4.   const RowWidget({Key? key}) : super(key: key);
5.
6.   @override
7.   Widget build(BuildContext context) {
8.     return Scaffold(
9.       appBar: AppBar(
10.         title: const Text("Widget Row"),
11.       ),
12.       body: Row(
13.         children: const [
14.           Text("Row 1"),
15.           Text("Row 2"),
16.           Text("Row 3"),
17.           Text("Row 4")
18.         ],
19.       ),
20.     );
21.   }
22. }
```

Kemudian seperti sebelumnya masukkan class **RowWidget** tersebut kedalam home pada **main.dart**, seperti pada column widget sebelumnya dan hasilnya akan menjadi



StatelessWidget dan StatefulWidget

StatelessWidget adalah class widget yang propertinya *immutable*, artinya nilainya tidak bisa diubah, sedangkan **StatefulWidget** nilainya dapat berubah-ubah.

Contoh StatelessWidget :

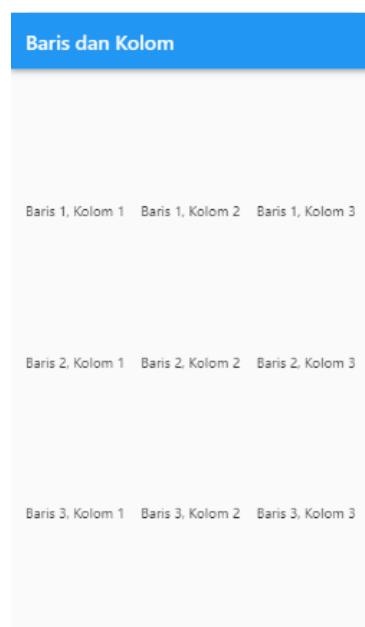
```
1. class HelloWorld extends StatelessWidget {  
2.   const HelloWorld({ Key? key }) : super(key: key);  
3.  
4.   @override  
5.   Widget build(BuildContext context) {  
6.     return Container(  
7.       );  
8.   }  
9. }  
10. }
```

Contoh StatefulWidget

```
1. class HelloWorld extends StatefulWidget {  
2.   const HelloWorld({ Key? key }) : super(key: key);  
3.  
4.   @override  
5.   _HelloWorldState createState() => _HelloWorldState();  
6. }  
7.  
8. class _HelloWorldState extends State<HelloWorld> {  
9.   @override  
10.  Widget build(BuildContext context) {  
11.    return Container(  
12.      );  
13.  }  
14. }  
15. }
```

TUGAS

Buat sebuah file dengan nama **baris_kolom.dart**, kemudian buat tampilan seperti berikut

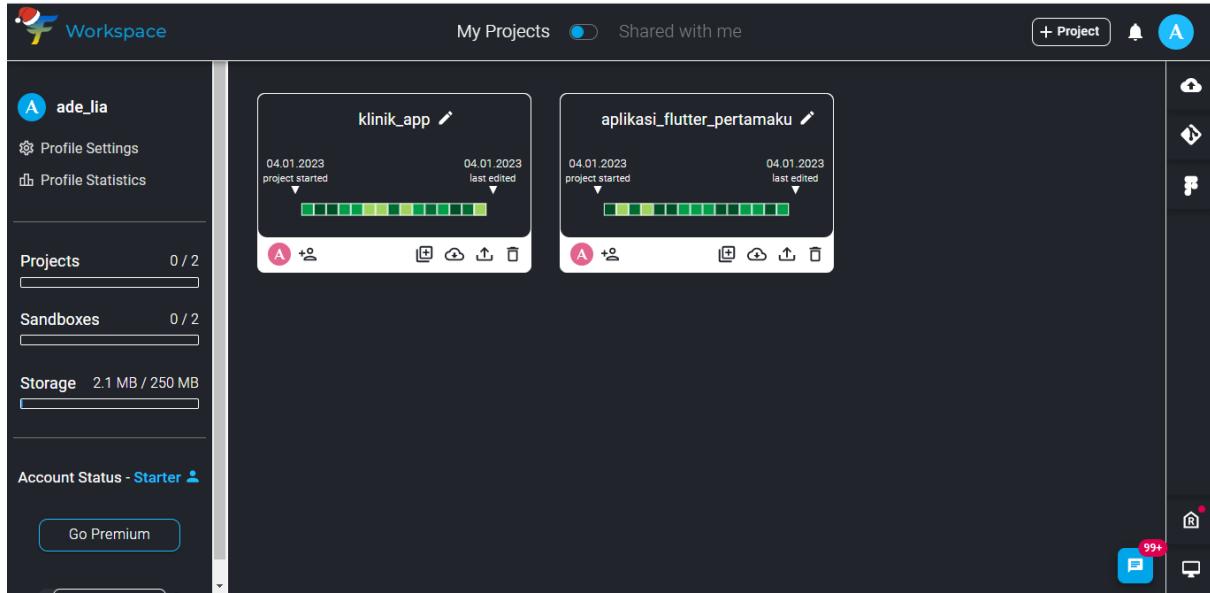


PERTEMUAN 5

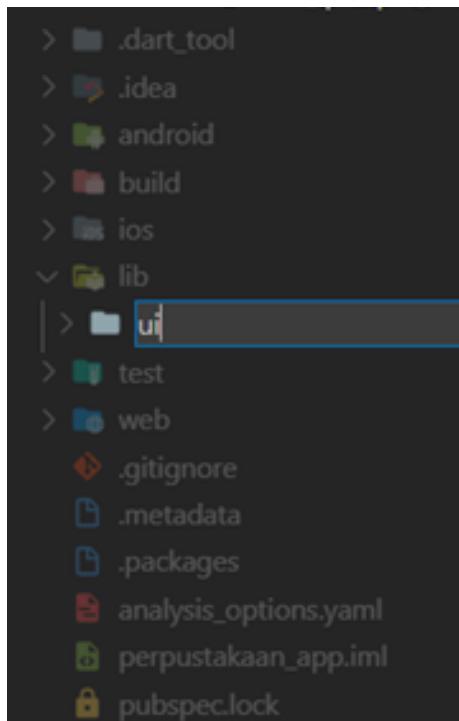
Membuat Halaman Data Poli

Buat sebuah projek dengan nama **klinik_app**, boleh menggunakan VSCode maupun **flutlab.io**

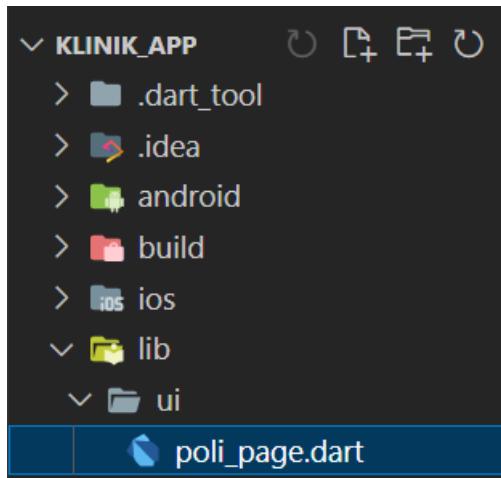
Berikut tampilan jika menggunakan flutlab.io



Selanjutnya kita akan belajar membuat form pada flutter, agar lebih rapi untuk tampilan halaman akan kita kelompokkan dalam sebuah folder tersendiri, dalam hal ini kita membuat folder dengan nama **ui** didalam folder **lib**.



Kemudian didalam folder **ui** tersebut kita buat sebuah file dengan nama **poli_page.dart**



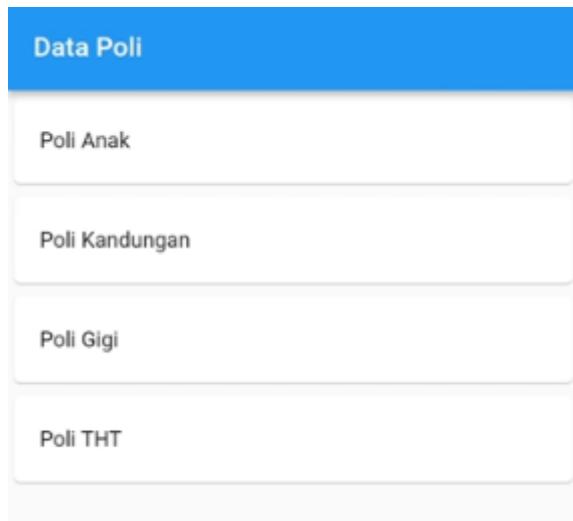
Kemudian Ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class PoliPage extends StatefulWidget {
4.   const PoliPage({super.key});
5.
6.   @override
7.   State<PoliPage> createState() => _PoliPageState();
8. }
9.
10. class _PoliPageState extends State<PoliPage> {
11.   @override
12.   Widget build(BuildContext context) {
13.     return Scaffold(
14.       appBar: AppBar(title: const Text("Data Poli")),
15.       body: ListView(
16.         children: const [
17.           Card(
18.             child: ListTile(
19.               title: const Text("Poli Anak"),
20.             ),
21.           ),
22.           Card(
23.             child: ListTile(
24.               title: const Text("Poli Kandungan"),
25.             ),
26.           ),
27.           Card(
28.             child: ListTile(
29.               title: const Text("Poli Gigi"),
30.             ),
31.           ),
32.           Card(
33.             child: ListTile(
34.               title: const Text("Poli THT"),
35.             ),
36.           ),
37.         ],
38.       ),
39.     );
40.   }
41. }
```

Ubah pada **main.dart** dengan memanggil class **PoliPage**

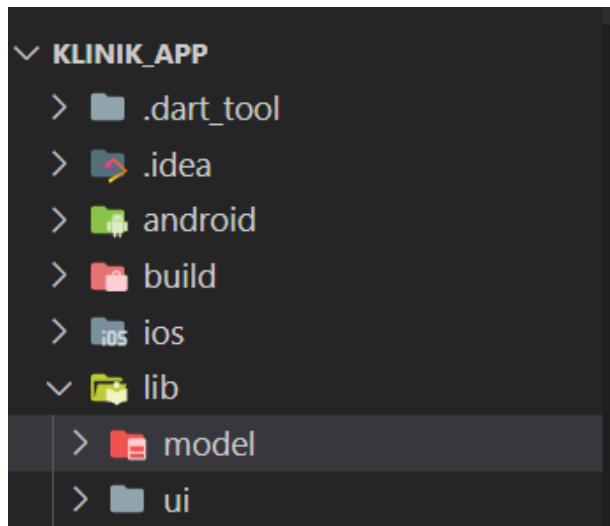
```
1. import 'package:flutter/material.dart';
2. import '/ui/poli_page.dart';
3.
4. void main() => runApp(MyApp());
5.
6. class MyApp extends StatelessWidget {
7.   @override
8.   Widget build(BuildContext context) {
9.     return MaterialApp(
10.       title: 'Klinik APP',
11.       debugShowCheckedModeBanner: false,
12.       home: PoliPage(),
13.     );
14.   }
15. }
```

Sehingga hasilnya akan menjadi

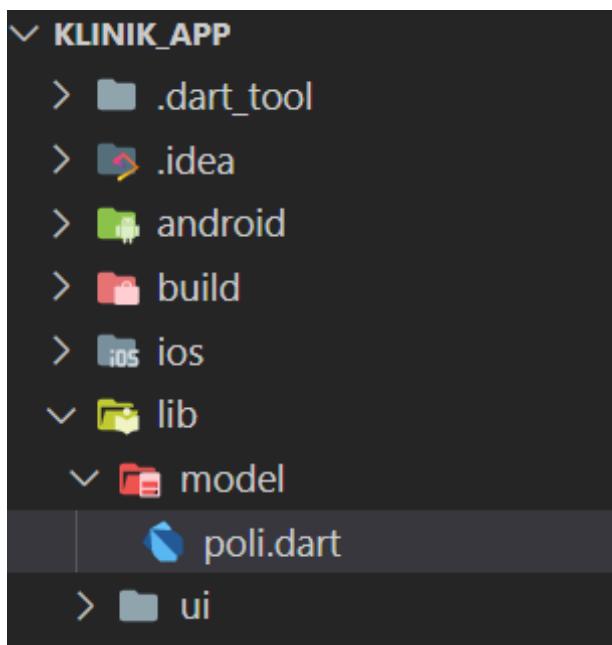


Membuat Detail Poli

Di dalam folder **lib** buat sebuah folder dengan nama **model**



Kemudian di dalam folder **model** tersebut, buat sebuah file dengan nama **poli.dart**



Di dalam file **poli.dart** tersebut ketikkan kode berikut

```
1. class Poli {  
2.   String? id;  
3.   String namaPoli;  
4.  
5.   Poli({this.id, required this.namaPoli});  
6. }
```

Buat sebuah file dengan nama **poli_detail.dart** di dalam folder **ui**, kemudian ketikkan kode berikut

```
1. import 'package:flutter/material.dart';  
2. import '../model/poli.dart';  
3.  
4. class PoliDetail extends StatefulWidget {  
5.   final Poli poli;  
6.  
7.   const PoliDetail({super.key, required this.poli});  
8.  
9.   @override  
10.  State<PoliDetail> createState() => _PoliDetailState();  
11. }  
12.  
13. class _PoliDetailState extends State<PoliDetail> {  
14.   @override  
15.   Widget build(BuildContext context) {  
16.     return Scaffold(  
17.       appBar: AppBar(title: Text("Detail Poli")),  
18.       body: Column(  
19.         children: [  
20.           SizedBox(height: 20),  
21.           Text(  
22.             "Nama Poli : ${widget.poli.namaPoli}",  
23.             style: TextStyle(fontSize: 20),  
24.           ),
```

```

25.         SizedBox(height: 20),
26.         Row(
27.             mainAxisAlignment: MainAxisAlignment.spaceEvenly,
28.             children: [
29.                 ElevatedButton(
30.                     onPressed: () {},
31.                     style:
32.                         ElevatedButton.styleFrom(backgroundColor: Colors.green),
33.                     child: const Text("Ubah")),
34.                 ElevatedButton(
35.                     onPressed: () {},
36.                     style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
37.                     child: const Text("Hapus")),
38.             ],
39.         ),
40.     ],
41. ),
42. );
43. }
44. }

```

Catatan : Jika menggunakan flutlab.io mungkin terjadi error pada bagian ElevatedButton.styleFrom(backgroundColor: Colors.green)

Untuk mengatasinya backgroundColor diubah menjadi primary

ElevatedButton.styleFrom(primary: Colors.green)

Menampilkan Detail Poli saat Data Poli diklik

Pada bagian ini kita akan menambahkan sebuah fungsi dimana saat salah satu Data Poli di klik, maka akan membuka halaman Detail Poli yang telah dibuat sebelumnya

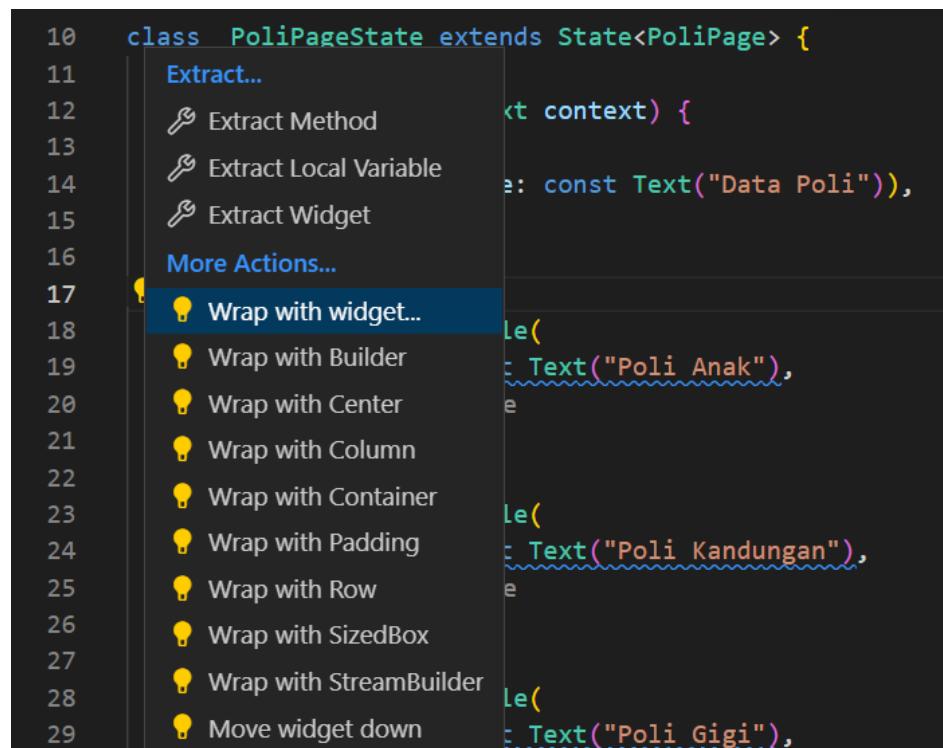
Kita akan memodifikasi Class PoliPage pada file **poli_page.dart**. Untuk menambahkan widget diatas widget lain yang telah dibuat dapat dilakukan dengan cara, arahkan kursor pada widget, misalnya dalam hal ini adalah widget **Card**

```

10  class _PoliPageState extends State<PoliPage> {
11    @override
12    Widget build(BuildContext context) {
13      return Scaffold(
14        appBar: AppBar(title: const Text("Data Poli")),
15        body: ListView(
16          children: const [
17            Card(
18              child: ListTile(
19                title: const Text("Poli Anak"),
20              ), // ListTile
21            ), // Card
22            Card(
23              child: ListTile(
24                title: const Text("Poli Kandungan"),
25              ), // ListTile
26            ), // Card
27            Card(
28              child: ListTile(
29                title: const Text("Poli Gigi"),

```

Pada bagian kiri akan muncul logo lampu, kemudian klik lampu tersebut dan pilih widget yang ingin ditambahkan atau dalam hal ini kita akan memilih **Wrap with widget..**



Kemudian akan menjadi

```

10 class _PoliPageState extends State<PoliPage> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       appBar: AppBar(title: const Text("Data Poli")),
15       body: ListView(
16         children: const [
17           widget(
18             child: Card(
19               child: ListTile(
20                 title: const Text("Poli Anak"),
21               ),
22             ),
23           ),
24           Card(
25             child: ListTile(
26               title: const Text("Poli Kandungan"),
27             ),
28           ),
29           Card(

```

Setelah itu ubah **widget** menjadi **GestureDetector** dan kita juga mengimport **poli_detail.dart** serta menambahkan **onTap** yang kemudian akan membuka halaman Detail Poli, sehingga kode untuk Class PoliPage menjadi

```

1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import 'poli_detail.dart';
4.
5. class PoliPage extends StatefulWidget {
6.   const PoliPage({super.key});
7.
8.   @override
9.   State<PoliPage> createState() => _PoliPageState();
10 }
11.
12 class _PoliPageState extends State<PoliPage> {
13   @override
14   Widget build(BuildContext context) {
15     return Scaffold(
16       appBar: AppBar(title: const Text("Data Poli")),
17       body: ListView(
18         children: [
19           GestureDetector(
20             child: Card(
21               child: ListTile(
22                 title: const Text("Poli Anak"),
23               ),
24             ),
25             onTap: () {
26               Poli poliAnak = new Poli(namaPoli: "Poli Anak");
27               Navigator.push(
28                 context,
29                 MaterialPageRoute(
30                   builder: (context) => PoliDetail(poli: poliAnak)));
31             },
32           ),
33           Card(
34             child: ListTile(

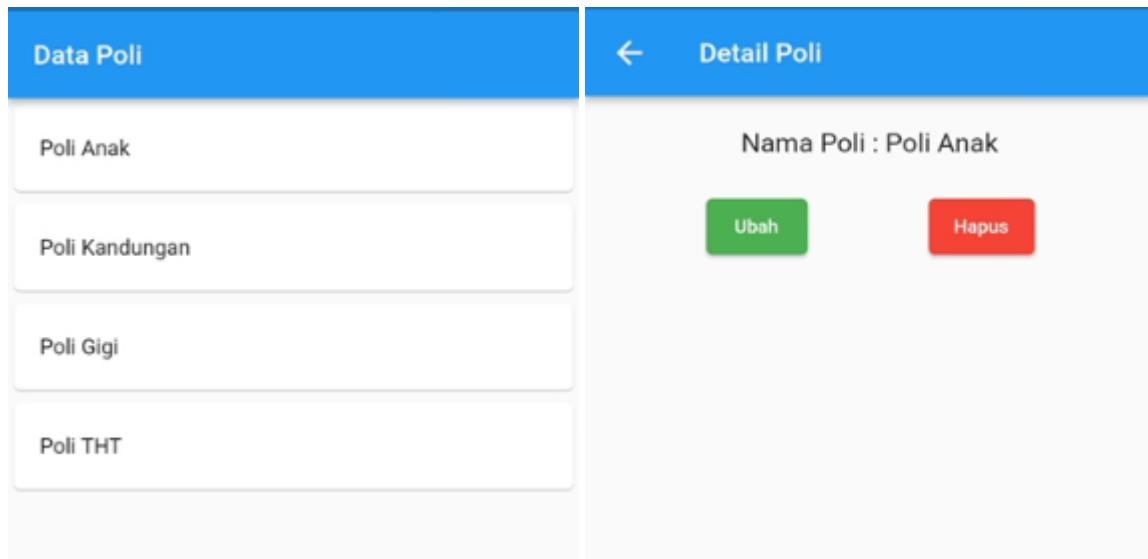
```

```

35.           title: const Text("Poli Kandungan"),
36.           ),
37.           ),
38.           Card(
39.             child: ListTile(
40.               title: const Text("Poli Gigi"),
41.             ),
42.             ),
43.             Card(
44.               child: ListTile(
45.                 title: const Text("Poli THT"),
46.               ),
47.             )
48.           ],
49.         ),
50.       );
51.     }
52.   }

```

Jika dijalankan, maka pada list "Poli Anak" dapat kita klik dan akan menuju ke halaman Detail Poli



TUGAS

Buat sebagaimana halnya pertemuan 5 (Membuat Halaman menampilkan list data dan Membuat Halaman Detail Data) dengan skema seperti gambar dibawah

Pegawai	Pasien
<ul style="list-style-type: none"> - id: int - nip: String - nama: String - tanggal_lahir: String - nomor_telepon: String - email: String - password: String 	<ul style="list-style-type: none"> - id: int - nomor_rm: String - nama: String - tanggal_lahir: String - nomor_telepon: String - alamat: String

PERTEMUAN 6

Memisahkan List Poli menjadi Poli Item

Jika dilihat pada list lainnya maka fungsi pindah ke halaman Detail Poli tidak akan berfungsi, karena kita harus membuat hal yang sama seperti pada list “Poli Anak”. Namun cara seperti ini tidak akan efektif, karena akan banyak kode dan menjadi tidak rapi. Solusinya adalah kita akan membuat class PoliItem yang akan menampilkan data Poli.

Buat file dengan nama **poli_item.dart** pada folder **ui** dan ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import 'poli_detail.dart';
4.
5. class PoliItem extends StatelessWidget {
6.   final Poli poli;
7.
8.   const PoliItem({super.key, required this.poli});
9.
10.  @override
11.  Widget build(BuildContext context) {
12.    return GestureDetector(
13.      child: Card(
14.        child: ListTile(
15.          title: Text("${poli.namaPoli}"),
16.        ),
17.        onTap: () {
18.          Navigator.push(context,
19.            MaterialPageRoute(builder: (context) => PoliDetail(poli: poli)));
20.
21.        },
22.      );
23.    }
24. }
```

GestureDetector adalah widget yang digunakan untuk mendeteksi gesture pada widget seperti gesture ontap, doubletab dan lain-lain.

Kemudian buka file **poli_page.dart** dan ubah menjadi

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import 'poli_detail.dart';
4. import 'poli_item.dart';
5.
6. class PoliPage extends StatefulWidget {
7.   const PoliPage({super.key});
8.
9.   @override
10.  State<PoliPage> createState() => _PoliPageState();
11. }
12.
13. class _PoliPageState extends State<PoliPage> {
14.   @override
15.   Widget build(BuildContext context) {
16.     return Scaffold(
17.       appBar: AppBar(title: const Text("Data Poli")),
18.       body: ListView(
```

```

19.         children: [
20.             PoliItem(poli: Poli(namaPoli: "Poli Anak")),
21.             PoliItem(poli: Poli(namaPoli: "Poli Kandungan")),
22.             PoliItem(poli: Poli(namaPoli: "Poli Gigi")),
23.             PoliItem(poli: Poli(namaPoli: "Poli THT")),
24.         ],
25.     ),
26. );
27. }
28. }
```

Membuat Form Tambah Poli

Selanjutnya kita akan belajar membuat form pada flutter, buat sebuah file dengan nama **poli_form.dart** didalam folder **ui**, kemudian ketikkan kode berikut.

```

1. import 'package:flutter/material.dart';
2.
3. class PoliForm extends StatefulWidget {
4.   const PoliForm({Key? key}) : super(key: key);
5.   _PoliFormState createState() => _PoliFormState();
6. }
7.
8. class _PoliFormState extends State<PoliForm> {
9.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
10.
11. @override
12. Widget build(BuildContext context) {
13.   return Scaffold(
14.     appBar: AppBar(title: const Text("Tambah Poli")),
15.     body: SingleChildScrollView(
16.       child: Form(
17.         key: _formKey,
18.         child: Column(
19.           children: [
20.             TextField(
21.               decoration: const InputDecoration(labelText: "Nama Poli")),
22.             SizedBox(height: 20),
23.             ElevatedButton(onPressed: () {}, child: const Text("Simpan"))
24.           ],
25.         ),
26.       ),
27.     ),
28.   );
29. }
30. }
```

Selanjutnya Form Poli diatas akan dibuka melalui halaman Data Poli. Buka kembali **poli_page.dart** kemudian pada bagian **AppBar** tambahkan kode berikut

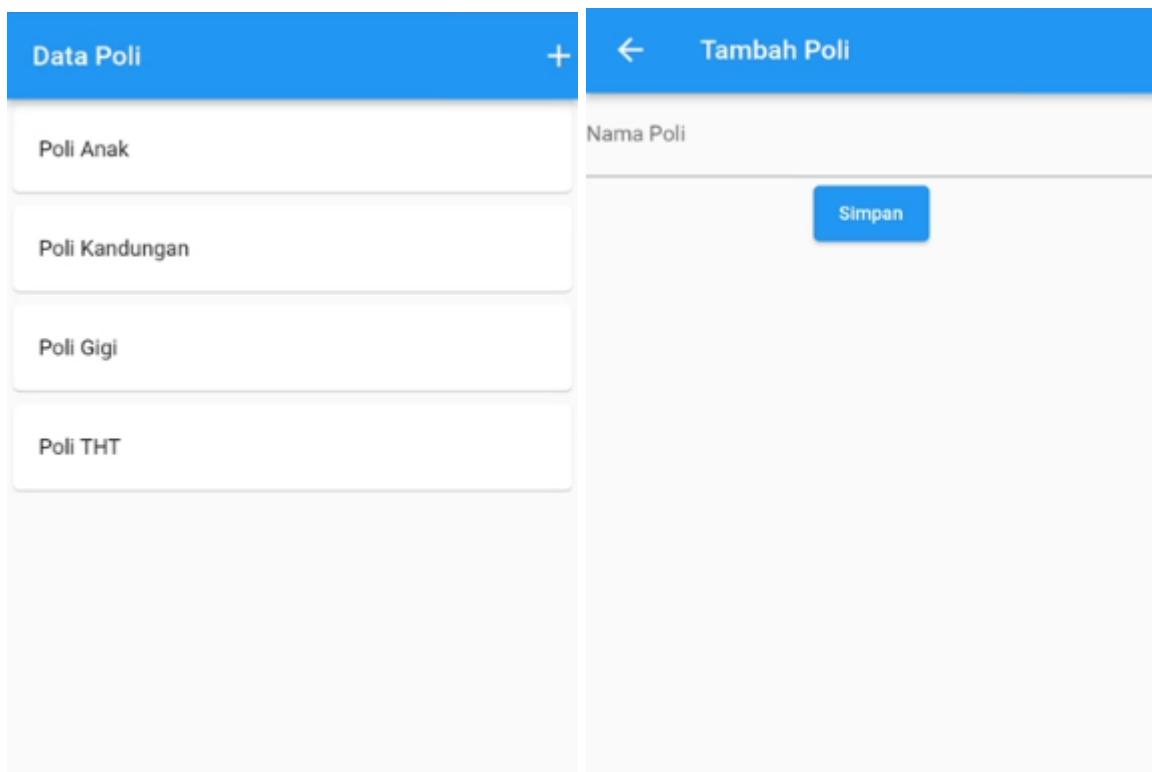
```

AppBar(
  title: const Text("Data Poli"),
  actions: [
    GestureDetector(
      child: const Icon(Icons.add),
      onTap: () {
        Navigator.push(
          context, MaterialPageRoute(builder: (context) => PoliForm()));
      },
    ),
  ],
)
```

Sehingga secara keseluruhan kode tersebut menjadi

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import 'poli_detail.dart';
4. import 'poli_item.dart';
5. import 'poli_form.dart';
6.
7. class PoliPage extends StatefulWidget {
8.   const PoliPage({super.key});
9.
10.  @override
11.  State<PoliPage> createState() => _PoliPageState();
12. }
13.
14. class _PoliPageState extends State<PoliPage> {
15.   @override
16.   Widget build(BuildContext context) {
17.     return Scaffold(
18.       appBar: AppBar(
19.         title: const Text("Data Poli"),
20.         actions: [
21.           GestureDetector(
22.             child: const Icon(Icons.add),
23.             onTap: () {
24.               Navigator.push(
25.                 context, MaterialPageRoute(builder: (context) => PoliForm()));
26.             },
27.           )
28.         ],
29.       ),
30.       body: ListView(
31.         children: [
32.           PoliItem(poli: Poli(namaPoli: "Poli Anak")),
33.           PoliItem(poli: Poli(namaPoli: "Poli Kandungan")),
34.           PoliItem(poli: Poli(namaPoli: "Poli Gigi")),
35.           PoliItem(poli: Poli(namaPoli: "Poli THT")),
36.         ],
37.       ),
38.     );
39.   }
40. }
```

Hasilnya akan muncul icon + pada bagian kanan AppBar, jika diklik akan membuka PoliForm



TUGAS

Buat sebagaimana halnya pertemuan 6 (Memisahkan class item dan Membuat Form tambah) dengan skema seperti gambar dibawah



PERTEMUAN 7

Pemisahan Widget Kedalam Fungsi-Fungsi

Agar kode mudah dibaca dan dikembangkan, akan lebih baik jika widget-widget yang digunakan dipisahkan kedalam method/function tertentu, misalnya pada **poli_form.dart** terdapat widget seperti **TextField** dan **ElevatedButton**, pada widget tersebut akan kita pisahkan kedalam method tersendiri didalam class, sehingga menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2.
3. class PoliForm extends StatefulWidget {
4.   const PoliForm({Key? key}) : super(key: key);
5.   _PoliFormState createState() => _PoliFormState();
6. }
7.
8. class _PoliFormState extends State<PoliForm> {
9.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
10.
11. @override
12. Widget build(BuildContext context) {
13.   return Scaffold(
14.     appBar: AppBar(title: const Text("Tambah Poli")),
15.     body: SingleChildScrollView(
16.       child: Form(
17.         key: _formKey,
18.         child: Column(
19.           children: [_fieldNamaPoli(), SizedBox(height: 20), _tombolSimpan()],
20.         ),
21.       ),
22.     ),
23.   );
24. }
25.
26. _fieldNamaPoli() {
27.   return TextField(
28.     decoration: const InputDecoration(labelText: "Nama Poli"),
29.   );
30. }
31.
32. _tombolSimpan() {
33.   return ElevatedButton(
34.     onPressed: () {},
35.     child: const Text("Simpan"));
36. }
37. }
```

Membuat Fungsi Tombol Simpan dan Menampilkan Data pada Detail Poli

Buka kembali file **poli_form.dart** import halaman **poli_detail.dart** kemudian tambahkan attribute

```
final _namaPoliCtrl = TextEditingController();
```

Pada **poli_form.dart** sehingga menjadi

```
1. import 'package:flutter/material.dart';
2. import 'package:klinikapp/model/poli.dart';
3. import 'package:klinikapp/ui/poli_detail.dart';
4.
5. class PoliForm extends StatefulWidget {
6.   const PoliForm({Key? key}) : super(key: key);
7.   _PoliFormState createState() => _PoliFormState();
8. }
9.
10. class _PoliFormState extends State<PoliForm> {
11.   final _formKey = GlobalKey<FormState>();
12.   final _namaPoliCtrl = TextEditingController();
13.
14.   @override
15.   Widget build(BuildContext context) {
16.     return Scaffold(
17.       appBar: AppBar(title: const Text("Tambah Poli")),
18.       body: SingleChildScrollView(
19.         child: Form(
20.           key: _formKey,
21.           child: Column(
22.             children: [_fieldNamaPoli(), SizedBox(height: 20), _tombolSimpan()],
23.           ),
24.         ),
25.       ),
26.     );
27.   }
28.
29.   _fieldNamaPoli() {
30.     return TextField(
31.       decoration: const InputDecoration(labelText: "Nama Poli"),
32.       controller: _namaPoliCtrl,
33.     );
34.   }
35.
36.   _tombolSimpan() {
37.     return ElevatedButton(
38.       onPressed: () {
39.         Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);
40.         Navigator.pushReplacement(context,
41.             MaterialPageRoute(builder: (context) => PoliDetail(poli: poli)));
42.       },
43.       child: const Text("Simpan"));
44.   }
45. }
```

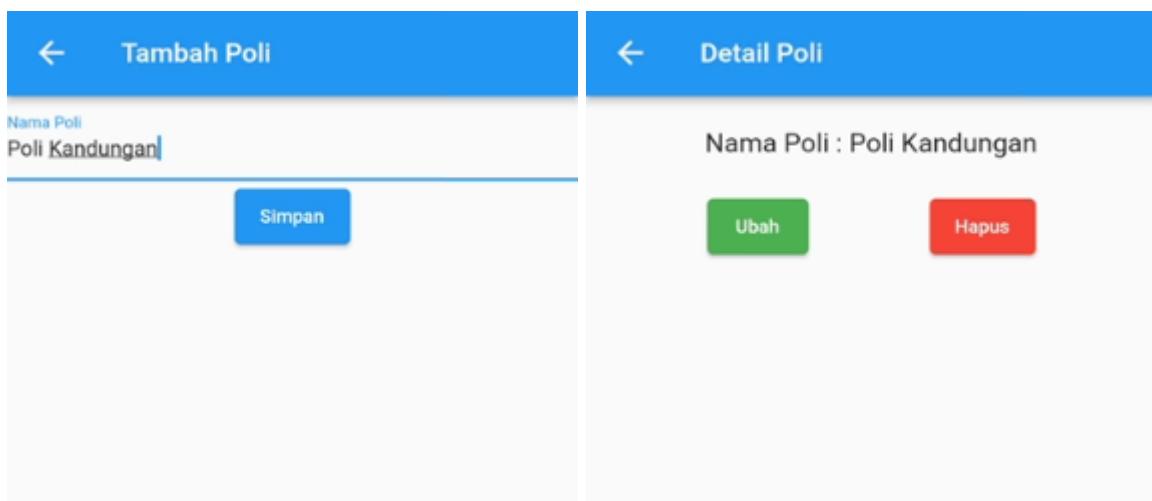
Pada setiap **TextField** yang telah dibuat, data yang diinput dikirim ke attribute **TextEditingController()** yang telah kita buat sebelumnya

Pada fungsi `_fieldNamaPoli()` menjadi

```
_fieldNamaPoli() {  
    return TextField(  
        decoration: const InputDecoration(labelText: "Nama Poli"),  
        controller: _namaPoliCtrl,  
    );  
}
```

Kemudian pada fungsi `_tombolSimpan()` pada saat diklik akan mengirim data inputan dan menampilkan data tersebut pada **PoliDetail** yang telah kita buat sebelumnya

```
_tombolSimpan() {  
    return ElevatedButton(  
        onPressed: () {  
            Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);  
            Navigator.pushReplacement(context  
                MaterialPageRoute(builder: (context) => PoliDetail(poli: poli)));  
        },  
        child: const Text("Simpan"));  
}
```



Membuat Form Ubah Poli

Pada bagian ini kita akan membuat Form untuk mengubah data poli, dimana pada halaman Detail Poli terdapat tombol ubah dan pada saat tombol ubah tersebut diklik, maka halaman akan beralih ke Form Ubah Poli dengan menampilkan data poli yang akan diubah.

Buat sebuah file dengan nama **poli_update_form.dart** di dalam folder **ui**, kemudian ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import '/ui/poli_detail.dart';
4.
5. class PoliUpdateForm extends StatefulWidget {
6.   final Poli poli;
7.
8.   const PoliUpdateForm({Key? key, required this.poli}) : super(key: key);
9.   _PoliUpdateFormState createState() => _PoliUpdateFormState();
10. }
11.
12. class _PoliUpdateFormState extends State<PoliUpdateForm> {
13.   final _formKey = GlobalKey<FormState>();
14.   final _namaPoliCtrl = TextEditingController();
15.
16. @override
17. void initState() {
18.   super.initState();
19.   setState(() {
20.     _namaPoliCtrl.text = widget.poli.namaPoli;
21.   });
22. }
23.
24. @override
25. Widget build(BuildContext context) {
26.   return Scaffold(
27.     appBar: AppBar(title: const Text("Ubah Poli")),
28.     body: SingleChildScrollView(
29.       child: Form(
30.         key: _formKey,
31.         child: Column(
32.           children: [_fieldNamaPoli(), SizedBox(height: 20), _tombolSimpan()],
33.         ),
34.       ),
35.     ),
36.   );
37. }
38.
39. _fieldNamaPoli() {
40.   return TextField(
41.     decoration: const InputDecoration(labelText: "Nama Poli"),
42.     controller: _namaPoliCtrl,
43.   );
44. }
45.
46. _tombolSimpan() {
47.   return ElevatedButton(
48.     onPressed: () {
49.       Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);
50.       Navigator.pop(context);
51.       Navigator.pushReplacement(context,
52.           MaterialPageRoute(builder: (context) => PoliDetail(poli: poli)));
53.     },
54.     child: const Text("Simpan Perubahan"));
55. }
```

```
55. }
56. }
```

Kemudian buka file **poli_detail.dart** , kita import **poli_update_form.dart** kemudian untuk memudahkan dalam pengkodean, lakukan pemisahan ke dalam fungsi untuk tombol ubah dan hapus. Pada bagian tombol ubah kita lakukan penambahan kode pada bagian **onPressed** seperti menjadi berikut

```
1. import 'package:flutter/material.dart';
2. import 'poli_update_form.dart';
3. import '../model/poli.dart';
4.
5. class PoliDetail extends StatefulWidget {
6.   final Poli poli;
7.
8.   const PoliDetail({super.key, required this.poli});
9.
10.  @override
11.  State<PoliDetail> createState() => _PoliDetailState();
12. }
13.
14. class _PoliDetailState extends State<PoliDetail> {
15.   @override
16.   Widget build(BuildContext context) {
17.     return Scaffold(
18.       appBar: AppBar(title: Text("Detail Poli")),
19.       body: Column(
20.         children: [
21.           SizedBox(height: 20),
22.           Text(
23.             "Nama Poli : ${widget.poli.namaPoli}",
24.             style: TextStyle(fontSize: 20),
25.           ),
26.           SizedBox(height: 20),
27.           Row(
28.             mainAxisAlignment: MainAxisAlignment.spaceEvenly,
29.             children: [
30.               _tombolUbah(),
31.               _tombolHapus(),
32.             ],
33.           ),
34.         ],
35.       ),
36.     );
37.   }
38.
39.   _tombolUbah() {
40.     return ElevatedButton(
41.       onPressed: () {
42.         Navigator.push(
43.           context,
44.           MaterialPageRoute(
45.             builder: (context) => PoliUpdateForm(poli: widget.poli)));
46.       },
47.       style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
48.       child: const Text("Ubah"));
49.   }
50.
51.   _tombolHapus() {
52.     return ElevatedButton(
53.       onPressed: () {},
54.       style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
55.       child: const Text("Hapus"));
56.   }
}
```

TUGAS

Buat sebagaimana halnya pertemuan 7 (Membuat fungsi tombol simpan, menampilkan data pada halaman detail dan membuat form ubah) dengan skema seperti gambar dibawah

Pegawai	Pasien
- id: int - nip: String - nama: String - tanggal_lahir: String - nomor_telepon: String - email: String - password: String	- id: int - nomor_rm: String - nama: String - tanggal_lahir: String - nomor_telepon: String - alamat: String

PERTEMUAN 8**UJIAN TENGAH SEMESTER**

PERTEMUAN 9

Menampilkan Pesan Konfirmasi Hapus Data

Masih pada file **poli_detail.dart** kita akan mengimport **poli_page.dart** dan juga memodifikasi fungsi **_tombolHapus** dimana saat tombol hapus diklik, akan memunculkan pesan konfirmasi penghapusan data, jika yakin ingin menghapus, halaman akan beralih ke halaman data mahasiswa. Namun untuk saat ini, karena tidak menggunakan database, maka data tidak akan terhapus.

```
_tombolHapus() {
    return ElevatedButton(
        onPressed: () {
            AlertDialog alertDialog = AlertDialog(
                content: const Text("Yakin ingin menghapus data ini?"),
                actions: [
                    // tombol ya
                    ElevatedButton(
                        onPressed: () {
                            Navigator.pop(context);
                            Navigator.pushReplacement(context,
                                MaterialPageRoute(builder: (context) => PoliPage()));
                        },
                        child: const Text("YA"),
                        style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
                    ),
                    // tombol batal
                    ElevatedButton(
                        onPressed: () {
                            Navigator.pop(context);
                        },
                        child: Text("Tidak"),
                        style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
                    )
                ],
            );
            showDialog(context: context, builder: (context) => alertDialog);
        },
        style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
        child: const Text("Hapus"));
}
```

Adapun keseluruhan kode **poli_detail.dart** adalah

```
1. import 'package:flutter/material.dart';
2. import 'poli_page.dart';
3. import 'poli_update_form.dart';
4. import '../model/poli.dart';
5.
6. class PoliDetail extends StatefulWidget {
7.   final Poli poli;
8.
9.   const PoliDetail({super.key, required this.poli});
10.
11. @override
12. State<PoliDetail> createState() => _PoliDetailState();
13. }
14.
15. class _PoliDetailState extends State<PoliDetail> {
16.   @override
17.   Widget build(BuildContext context) {
18.     return Scaffold(
19.       appBar: AppBar(title: Text("Detail Poli")),
```

```

20.     body: Column(
21.       children: [
22.         SizedBox(height: 20),
23.         Text(
24.           "Nama Poli : ${widget.poli.namaPoli}",
25.           style: TextStyle(fontSize: 20),
26.         ),
27.         SizedBox(height: 20),
28.         Row(
29.           mainAxisAlignment: MainAxisAlignment.spaceEvenly,
30.           children: [
31.             _tombolUbah(),
32.             _tombolHapus(),
33.           ],
34.         ),
35.       ],
36.     ),
37.   );
38. }
39.
40. _tombolUbah() {
41.   return ElevatedButton(
42.     onPressed: () {
43.       Navigator.push(
44.         context,
45.         MaterialPageRoute(
46.           builder: (context) => PoliUpdateForm(poli: widget.poli)));
47.     },
48.     style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
49.     child: const Text("Ubah"));
50. }
51.
52. _tombolHapus() {
53.   return ElevatedButton(
54.     onPressed: () {
55.       AlertDialog alertDialog = AlertDialog(
56.         content: const Text("Yakin ingin menghapus data ini?"),
57.         actions: [
58.           // tombol ya
59.           ElevatedButton(
60.             onPressed: () {
61.               Navigator.pop(context);
62.               Navigator.pushReplacement(context,
63.                 MaterialPageRoute(builder: (context) => PoliPage()));
64.             },
65.             child: const Text("YA"),
66.             style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
67.           ),
68.           // tombol batal
69.           ElevatedButton(
70.             onPressed: () {
71.               Navigator.pop(context);
72.             },
73.             child: Text("Tidak"),
74.             style: ElevatedButton.styleFrom(backgroundColor: Colors.green),
75.           ),
76.         ],
77.       );
78.       showDialog(context: context, builder: (context) => alertDialog);
79.     },
80.     style: ElevatedButton.styleFrom(backgroundColor: Colors.red),
81.     child: const Text("Hapus"));
82. }
83. }

```

Membuat Halaman Beranda dan Login

Membuat Halaman Beranda

Buat sebuah file dengan nama **beranda.dart** pada folder **ui** dan ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class Beranda extends StatelessWidget {
4.   const Beranda({super.key});
5.
6.   @override
7.   Widget build(BuildContext context) {
8.     return Scaffold(
9.       appBar: AppBar(title: Text("Beranda")),
10.      body: Center(
11.        child: Text("Selamat Datang"),
12.      ),
13.    );
14.  }
15. }
```

Membuat Halaman Login

Buat sebuah file dengan nama **login.dart** pada folder **ui** dan ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class Login extends StatefulWidget {
4.   const Login({Key? key}) : super(key: key);
5.   _LoginState createState() => _LoginState();
6. }
7.
8. class _LoginState extends State<Login> {
9.   final _formKey = GlobalKey<FormState>();
10.  final _usernameCtrl = TextEditingController();
11.  final _passwordCtrl = TextEditingController();
12.
13.  @override
14.  Widget build(BuildContext context) {
15.    return Scaffold(
16.      body: SingleChildScrollView(
17.        child: SafeArea(
18.          child: Container(
19.            height: MediaQuery.of(context).size.height,
20.            child: Column(
21.              mainAxisAlignment: MainAxisAlignment.center,
22.              children: [
23.                Text("Login Admin",
24.                  style: TextStyle(fontSize: 22, fontWeight: FontWeight.w500)),
25.                SizedBox(height: 50),
26.                Center(
27.                  child: Form(
28.                    key: _formKey,
29.                    child: Container(
30.                      width: MediaQuery.of(context).size.width / 1.3,
31.                      child: Column(
32.                        children: [
33.                          _usernameTextField(),
34.                          SizedBox(height: 20),
35.                          _passwordTextField(),
36.                          SizedBox(height: 40),
37.                          _tombolLogin(),
38.                        ],
39.                      ),
40.                    ),
41.                  ),
42.                ),
43.              ],
44.            ),
45.          ),
46.        ),
47.      ),
48.    );
49.  }
50. }
```

```

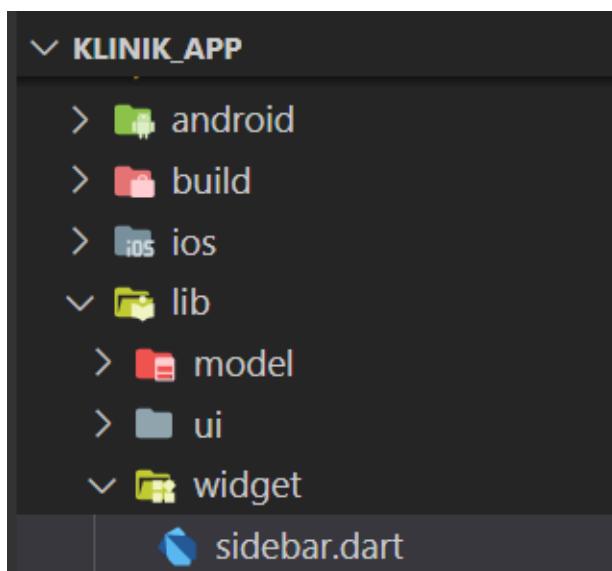
38.             ],
39.             )),
40.             )),
41.             )
42.             ],
43.             )),
44.             )),
45.             ),
46.         );
47.     }
48.
49.     Widget _usernameTextField() {
50.         return TextFormField(
51.             decoration: InputDecoration(labelText: "Username"),
52.             controller: _usernameCtrl,
53.         );
54.     }
55.
56.     Widget _passwordTextField() {
57.         return TextFormField(
58.             decoration: InputDecoration(labelText: "Password"),
59.             obscureText: true,
60.             controller: _passwordCtrl,
61.         );
62.     }
63.
64.     Widget _tombolLogin() {
65.         return Container(
66.             width: MediaQuery.of(context).size.width,
67.             child: ElevatedButton(child: Text("Login"), onPressed: () {}));
68.     }
69. }

```

Membuat Sidebar Menu (Drawer)

Drawer adalah menu yang berada di sebelah kiri layar android. Biasanya drawer akan muncul saat kita menekan icon menu yang berada di kiri atas appbar. Sedangkan di website, drawer bisa kita sebut juga dengan sidebar menu.

Buat sebuah folder di dalam **lib** dengan nama **widget** kemudian di dalam folder **widget** tersebut buat sebuah file dengan nama **sidebar.dart**



Buka file **sidebar.dart** dan ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2. import '../ui/beranda.dart';
3. import '../ui/login.dart';
4. import '../ui/poli_page.dart';
5.
6. class Sidebar extends StatelessWidget {
7.   const Sidebar({Key? key}) : super(key: key);
8.   @override
9.   Widget build(BuildContext context) {
10.     return Drawer(
11.       child: ListView(
12.         padding: EdgeInsets.zero,
13.         children: [
14.           UserAccountsDrawerHeader(
15.             accountName: Text("Admin"),
16.             accountEmail: Text("admin@admin.com")),
17.           ListTile(
18.             leading: Icon(Icons.home),
19.             title: Text("Beranda"),
20.             onTap: () {
21.               Navigator.push(
22.                 context, MaterialPageRoute(builder: (context) => Beranda()));
23.             },
24.           ),
25.           ListTile(
26.             leading: Icon(Icons.accessible),
27.             title: Text("Poli"),
28.             onTap: () {
29.               Navigator.push(
30.                 context, MaterialPageRoute(builder: (context) => PoliPage()));
31.             },
32.           ),
33.           ListTile(
34.             leading: Icon(Icons.people),
35.             title: Text("Pegawai"),
36.             onTap: () {},
37.           ),
38.           ListTile(
39.             leading: Icon(Icons.account_box_sharp),
40.             title: Text("Pasien"),
41.             onTap: () {},
42.           ),
43.           ListTile(
44.             leading: Icon(Icons.logout_rounded),
45.             title: Text("Keluar"),
46.             onTap: () {
47.               Navigator.pushAndRemoveUntil(
48.                 context,
49.                 MaterialPageRoute(builder: (context) => Login()),
50.                 (Route<dynamic> route) => false);
51.             },
52.           ),
53.         ],
54.       ),
55.     );
56.   }
57. }
```

Kemudian buka file **beranda.dart** pada bagian Scaffold tambahkan parameter **drawer** dan masukkan class **Sidebar** pada parameter drawer tersebut (baris 10).

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

```

2. import '../widget/sidebar.dart';
3.
4. class Beranda extends StatelessWidget {
5.   const Beranda({super.key});
6.
7.   @override
8.   Widget build(BuildContext context) {
9.     return Scaffold(
10.       drawer: Sidebar(),
11.       appBar: AppBar(title: Text("beranda")),
12.       body: Center(
13.         child: Text("Selamat Datang"),
14.       ),
15.     );
16.   }
17. }

```

Kemudian buka file **poli_page.dart** pada bagian Scaffold lakukan hal yang sama seperti pada halaman beranda yaitu tambahkan parameter **drawer** dan masukkan class **Sidebar** pada parameter drawer tersebut (baris 19).

```

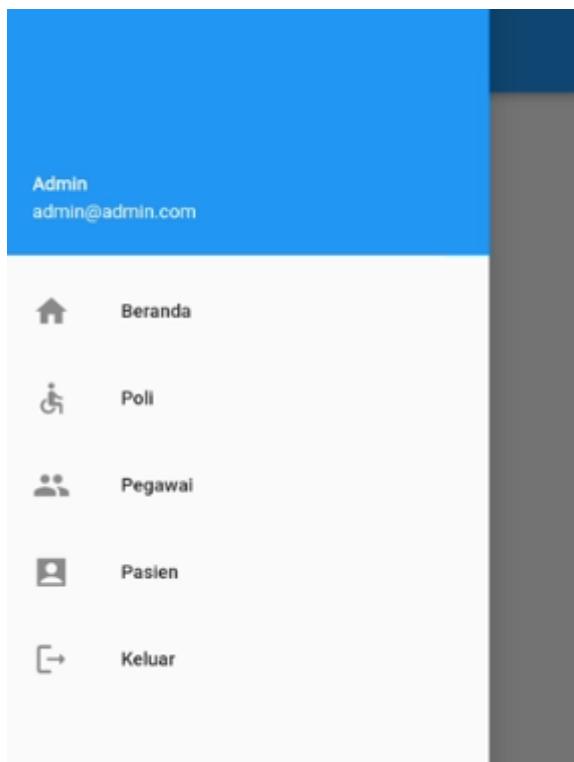
1. import 'package:flutter/material.dart';
2. import '../widget/sidebar.dart';
3. import '../model/poli.dart';
4. import 'poli_detail.dart';
5. import 'poli_item.dart';
6. import 'poli_form.dart';
7.
8. class PoliPage extends StatefulWidget {
9.   const PoliPage({super.key});
10.
11.  @override
12.  State<PoliPage> createState() => _PoliPageState();
13. }
14.
15. class _PoliPageState extends State<PoliPage> {
16.  @override
17.  Widget build(BuildContext context) {
18.    return Scaffold(
19.      drawer: Sidebar(),
20.      appBar: AppBar(
21.        title: const Text("Data Poli"),
22.        actions: [
23.          GestureDetector(
24.            child: const Icon(Icons.add),
25.            onTap: () {
26.              Navigator.push(
27.                context,
28.                MaterialPageRoute(builder: (context) => PoliForm()),
29.              ),
30.            ],
31.          ),
32.          body: ListView(
33.            children: [
34.              PoliItem(poli: Poli(namaPoli: "Poli Anak")),
35.              PoliItem(poli: Poli(namaPoli: "Poli Kandungan")),
36.              PoliItem(poli: Poli(namaPoli: "Poli Gigi")),
37.              PoliItem(poli: Poli(namaPoli: "Poli THT")),
38.            ],
39.          ),
40.        );
41.      }
42.    }

```

Buka file **main.dart** arahkan halaman awal pada beranda

```
1. import 'package:flutter/material.dart';
2. import 'ui/beranda.dart';
3.
4. void main() {
5.   runApp(const MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   const MyApp({super.key});
10.
11. @override
12. Widget build(BuildContext context) {
13.   return const MaterialApp(
14.     debugShowCheckedModeBanner: false,
15.     title: 'Klinik',
16.     home: Beranda(),
17.   );
18. }
19. }
```

Sehingga hasilnya akan seperti berikut



TUGAS

- Buat Pesan Konfirmasi hapus data untuk bagian Pegawai dan Pasien
- Jika kita lihat pada drawer, terdapat menu **Beranda, Poli, Pegawai, Pasien** dan Keluar. Tugas Anda menyatukan halaman Manajemen (CRUD) Pegawai dan Pasien yang telah anda buat sebelumnya untuk tampil saat menu pegawai atau pasien dipilih

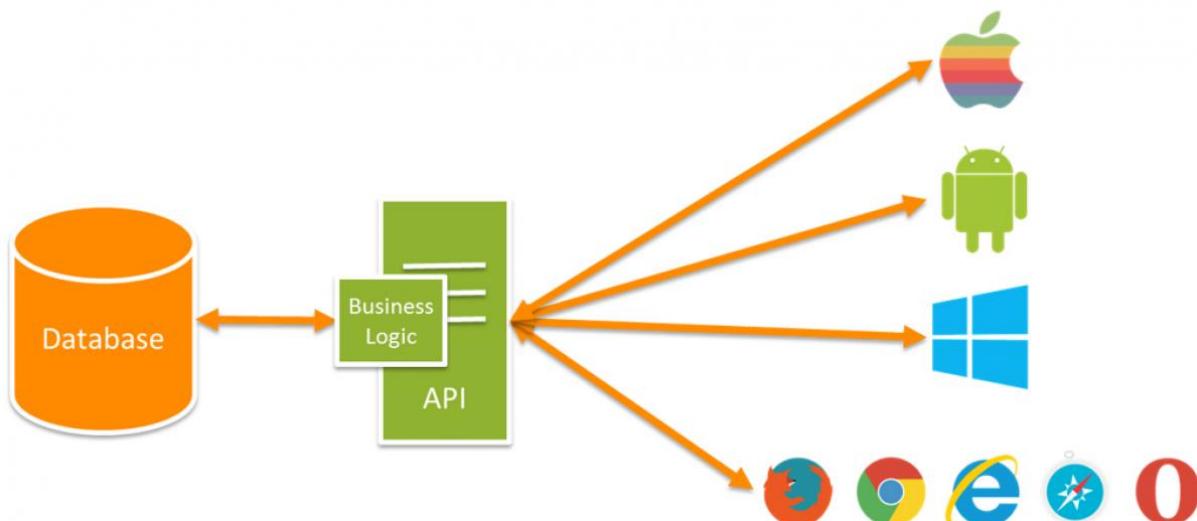
PERTEMUAN 10

Apa itu API

API atau Application Programming Interface adalah sebuah interface yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. Jadi, API berperan sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau lintas platform.

Perumpamaan yang bisa digunakan untuk menjelaskan API adalah seorang pelayan di restoran. Tugas pelayan tersebut adalah menghubungkan tamu restoran dengan juru masak. Tamu cukup memesan makanan sesuai daftar menu yang ada dan pelayan memberitahukannya ke juru masak. Nantinya, pelayan akan kembali ke tamu tadi dengan masakan yang sudah siap sesuai pesanan.

Itulah gambaran tugas dari API dalam pengembangan aplikasi.



Sumber : codepolitan.com

Arsitektur API

Ada tiga arsitektur API yang sering digunakan oleh developer dalam pembangunan aplikasi. Arsitektur ini berkaitan pada bentuk data yang dikirim. Adapun Arsitektur API yang sering digunakan adalah

1. RPC

RPC merupakan teknologi untuk membuat komunikasi antara client side dan server side bisa dilakukan dengan konsep sederhana.

RPC memiliki dua jenis, yaitu XML-RPC dan JSON-RPC. Sesuai namanya, XML-RPC menggunakan format XML sebagai media perpindahan data, sedangkan JSON-RPC menggunakan JSON untuk perpindahan data.

2. SOAP

Arsitektur API lainnya adalah SOAP (Simple Object Access Protocol). Arsitektur ini menggunakan XML (Extensible Markup Language) yang memungkinkan semua data disimpan dalam dokumen.

3. REST

REST atau Representational State Transfer adalah arsitektur API yang cukup populer karena kemudahan penggunaannya. Tak perlu coding yang panjang untuk menggunakannya.

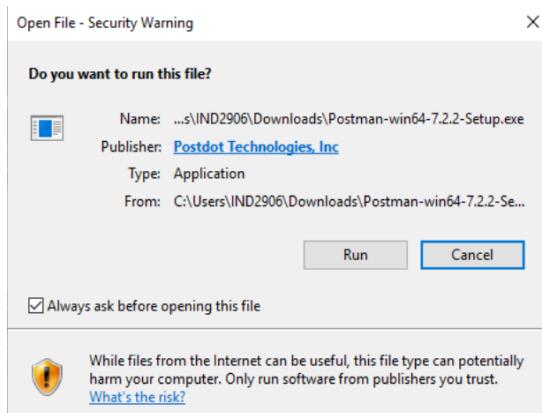
REST menggunakan JSON sebagai bentuk datanya sehingga lebih ringan. Performa aplikasi pun menjadi lebih baik.

Install Postman

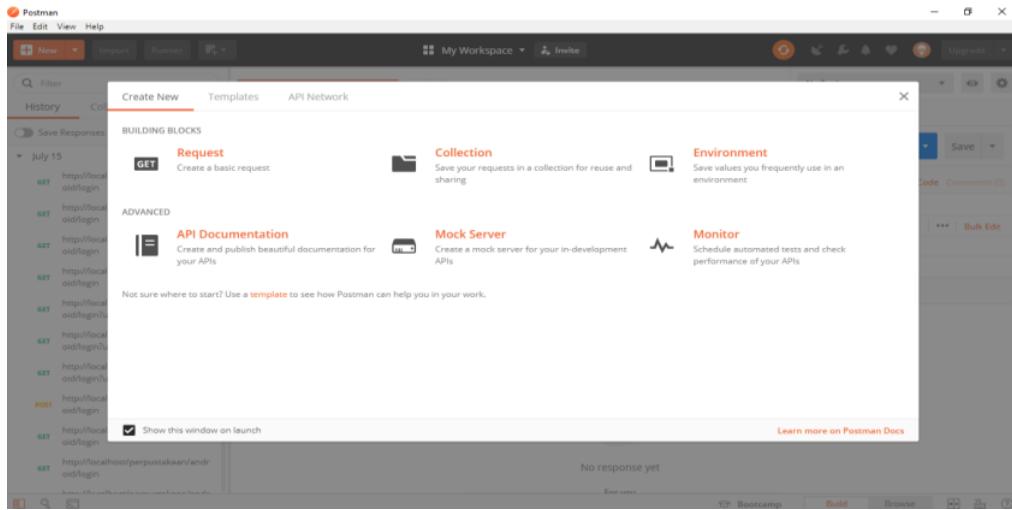
Postman adalah sebuah aplikasi fungsinya adalah sebagai REST Client atau istilahnya adalah aplikasi yang digunakan untuk melakukan uji coba REST API yang telah kita buat. Postman ini merupakan tools wajib bagi para developer yang bergerak pada pembuatan API, fungsi utama postman ini adalah sebagai GUI API Caller Pemanggil. namun sekarang postman juga menyediakan fitur lain yaitu Sharing Collection API for Documentation (free), Testing API (free), Realtime Collaboration Team (paid), Monitoring API (paid), Integration (paid).

Postman tersedia sebagai aplikasi asli untuk sistem operasi macOS, Windows (32-bit dan 64-bit), dan Linux (32-bit dan 64-bit). Untuk mendapatkan aplikasi Postman, dapat diunduh pada website resminya yaitu getpostman.com atau dapat diunduh pada halaman <https://www.postman.com/downloads/>

Setelah berhasil mengunduh paket instalasi postman, kemudian jalankan dengan cara klik dua kali. Pilih run jika muncul pop up seperti berikut :



Kemudian tunggu hingga proses instalasi selesai dan muncul seperti gambar berikut



Membuat API Klinik dummy dengan mockapi.io

Silahkan kunjungi laman <https://mockapi.io/> kemudian daftar/buat akun mockapi.io

The easiest way to mock REST APIs! (Check out [docs](#))

[GET STARTED](#)

DEMO PROJECT

API endpoint
https://SECRET.mockapi.io/:endpoint

[NEW RESOURCE](#) [GENERATE ALL](#) [RESET ALL](#)

```

graph TD
    users[users] --> posts[posts]
    posts --> likes[likes]
    posts --> comments[comments]
  
```

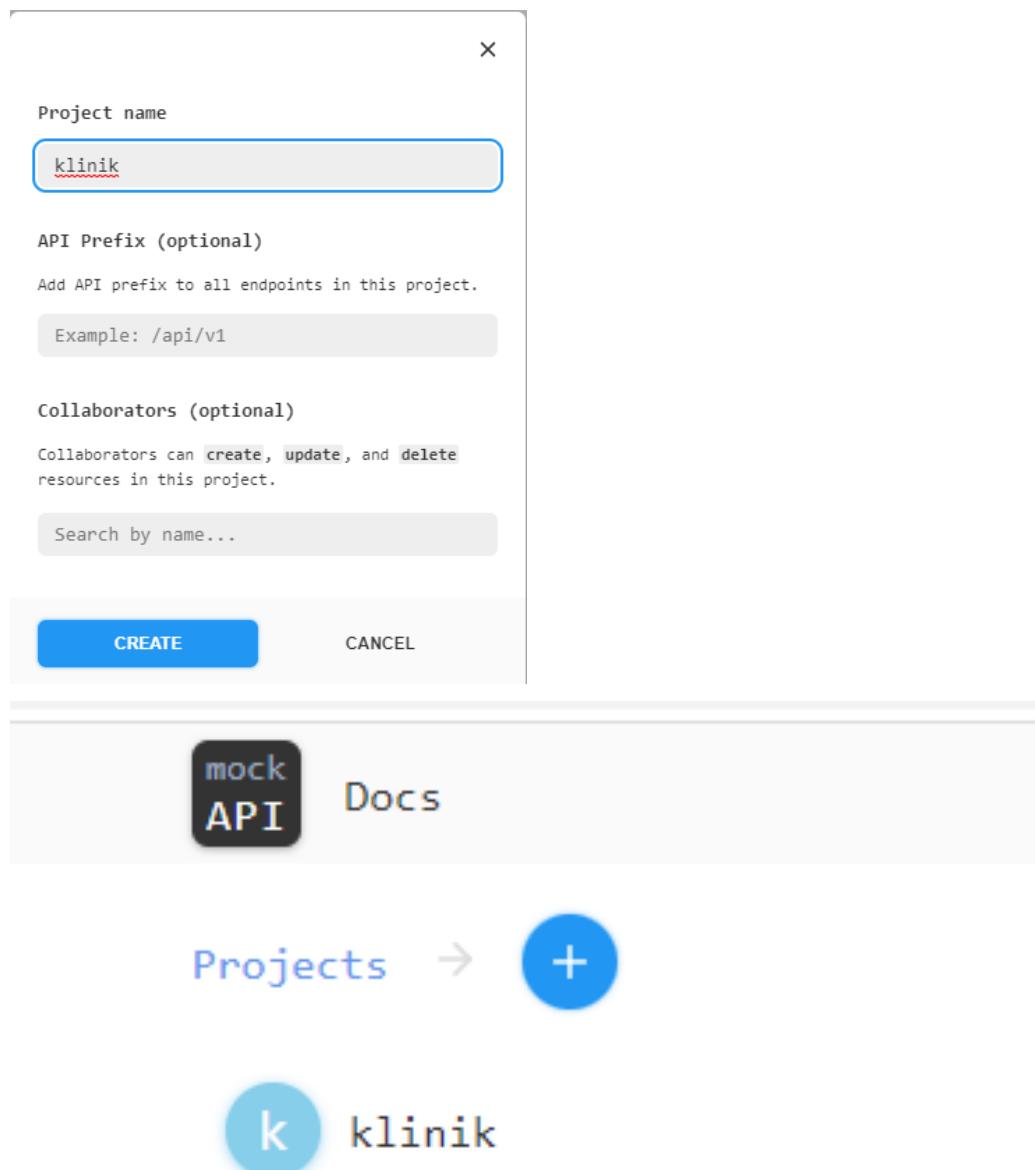
Contact us mockapi.io@gmail.com

Jika sudah login ke dalam mockapi.io kemudian buat projek dengan mengklik tombol tambah

Projects → +

No projects yet...

Buat projek dengan nama **klinik** kemudian klik tombol **create**



Kemudian klik projek klinik yang telah dibuat tadi sehingga muncul seperti tampilan berikut

The screenshot shows the Mock API interface. At the top left is a dark button labeled "mock API". To its right is a "Docs" link. Below this, a navigation bar includes "Projects" (with a dropdown arrow), a blue circular icon containing a white letter "k", the project name "klinik", a user profile icon, and a three-dot menu icon. The main content area is titled "API endpoint" and displays the URL "https://63588b60c27556d2893f7a12.mockapi.io/:endpoint". Below this is a blue "NEW RESOURCE" button.

Klik tombol **New Resource** sehingga muncul tampilan berikut

The screenshot shows the "New Resource" creation dialog. At the top right is a close button ("X"). The first section is "Resource name", which contains a placeholder "Enter meaningful resource name, it will be used to generate API endpoints." and a text input field with the placeholder "Example: users, comments, articles...". The second section is "Schema (optional)", with the sub-instruction "Define Resource schema, it will be used to generate mock data.". It features four input fields for "id" (set to "Object ID"), "createdAt" (set to "Faker.js" and "Recent"), "name" (set to "Faker.js" and "Find name"), and "avatar" (set to "Faker.js" and "Avatar"). A plus sign button ("+") is located below these fields. The third section is "Object template (optional)", with the sub-instruction "To define more complex structure for your data use JSON template. You can reference https://www.jsoneditoronline.org/". At the bottom are two buttons: a blue "CREATE" button on the left and a "CANCEL" button on the right.

Isi **Resource Name** dengan **poli** kemudian pada bagian **Schema** ubah sehingga menjadi seperti gambar berikut

The screenshot shows the configuration screen for creating a new API resource. At the top, there is a field labeled "Resource name" with the value "poli". Below it, under "Schema (optional)", there are two fields: "id" (Object ID) and "nama_poli" (String). A plus sign button is available to add more fields. At the bottom, there is a "CREATE" button and a "CANCEL" button.

Resource name

Enter meaningful resource name, it will be used to generate API endpoints.

poli

Schema (optional)

Define Resource schema, it will be used to generate mock data.

id Object ID

nama_poli String

+

Object template (optional)

To define more complex structure for your data use JSON template. You can reference Faker.js methods using `'\$`.

```
{  
  "username": "$internet.userName",  
  "knownIps": ["$internet.ip", "$internet.ipv6"],  
}
```

EXAMPLE

CREATE CANCEL

Kemudian klik tombol **create** dan hasilnya akan menjadi seperti berikut

The screenshot shows the main dashboard of the Mock API tool. At the top, there is a navigation bar with "mock API" and "Docs" buttons. Below the navigation, there is a "Projects" section with a "klinik" project selected. In the center, there is a "API endpoint" field containing the URL "https://63588b60c27556d2893f7a12.mockapi.io/:endpoint". Below the URL, there are "NEW RESOURCE" and "RESET ALL" buttons. A "GENERATE ALL" button is also visible. At the bottom, there is a table for managing the "poli" resource, which currently has 0 entries. The table includes "Data", "Edit", and "Delete" buttons.

mock API Docs

Projects → klinik

API endpoint
https://63588b60c27556d2893f7a12.mockapi.io/:endpoint

NEW RESOURCE GENERATE ALL RESET ALL

poli	0	Data	Edit	Delete
------	---	------	------	--------

Pada bagian bawah **API endpoint** (yang digaris bawah merah) merupakan alamat utama url yang dapat kita kunjungi untuk mengakses API.

Untuk mengakses data mahasiswa yang telah kita buat sebelumnya, kita dapat mengakses dengan alamat <https://63588b60c27556d2893f7a12.mockapi.io/poli> (sesuaikan dengan url yang ada pada mockapi anda)

Create Poli dengan method POST

The screenshot shows a Postman request for a POST method to the URL <https://63588b60c27556d2893f7a12.mockapi.io/poli>. The response status is 201 Created, with a response time of 19 s and 34 B. The JSON body sent was:

```
1 + {  
2   "nama_poli": "Poli Umum"  
3 }
```

The response preview shows the created poli entry:

```
1 + {  
2   "nama_poli": "Poli Umum",  
3   "id": "1"  
4 }
```

List Poli dengan method GET

The screenshot shows a Postman request for a GET method to the URL <https://63588b60c27556d2893f7a12.mockapi.io/poli>. The response status is 200 OK, with a response time of 683 ms and 71 B. The response preview shows two poli entries:

```
1 + [  
2   {  
3     "nama_poli": "Poli Umum",  
4     "id": "1"  
5   },  
6   {  
7     "nama_poli": "Poli Anak",  
8     "id": "2"  
9   }  
10 ]
```

Show dengan method GET

The screenshot shows a Postman request for a GET method to the URL <https://63588b60c27556d2893f7a12.mockapi.io/poli/1>. The response status is 200 OK, with a response time of 1.29 s and 34 B. The response preview shows one poli entry:

```
1 + {  
2   "nama_poli": "Poli Umum",  
3   "id": "1"  
4 }
```

Update Poli dengan method PUT

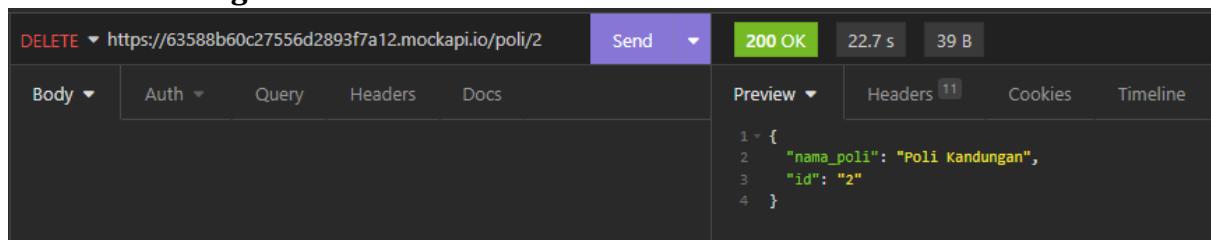
The screenshot shows a Postman request for a PUT method to the URL <https://63588b60c27556d2893f7a12.mockapi.io/poli/2>. The response status is 200 OK, with a response time of 3.76 s and 39 B. The JSON body sent was:

```
1 + {  
2   "nama_poli": "Poli Kandungan"  
3 }
```

The response preview shows the updated poli entry:

```
1 + {  
2   "nama_poli": "Poli Kandungan",  
3   "id": "2"  
4 }
```

Delete Poli dengan method DELETE

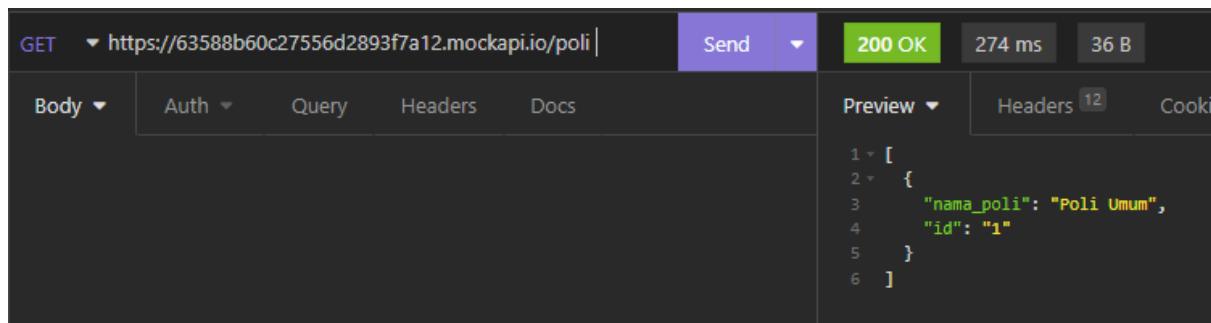


DELETE https://63588b60c27556d2893f7a12.mockapi.io/poli/2

Send 200 OK 22.7 s 39 B

Body Auth Query Headers Docs Preview Headers Cookies Timeline

```
1 < {  
2   "nama_poli": "Poli Kandungan",  
3   "id": "2"  
4 }
```



GET https://63588b60c27556d2893f7a12.mockapi.io/poli

Send 200 OK 274 ms 36 B

Body Auth Query Headers Docs Preview Headers Cookies

```
1 < [  
2   {  
3     "nama_poli": "Poli Umum",  
4     "id": "1"  
5   }  
6 ]
```

TUGAS

Buat tabel (Resource) dengan skema seperti berikut

Pegawai
- id: int
- nip: String
- nama: String
- tanggal_lahir: Date
- nomor_telepon: String
- email: String
- password: String

Pasien
- id: int
- nomor_rm: String
- nama: String
- tanggal_lahir: Date
- nomor_telepon: String
- alamat: String

PERTEMUAN 11

Menambahkan Depedencies

Pada Editor Offline (VSCode)

Dalam pembuatan aplikasi ini dibutuhkan depedensi untuk menerima dan mengirim request ke Rest API (**dio**). Pastikan komputer atau laptop terhubung ke internet, buka file **pubspec.yaml** kemudian pada bagian **dependencies** tambahkan kode berikut

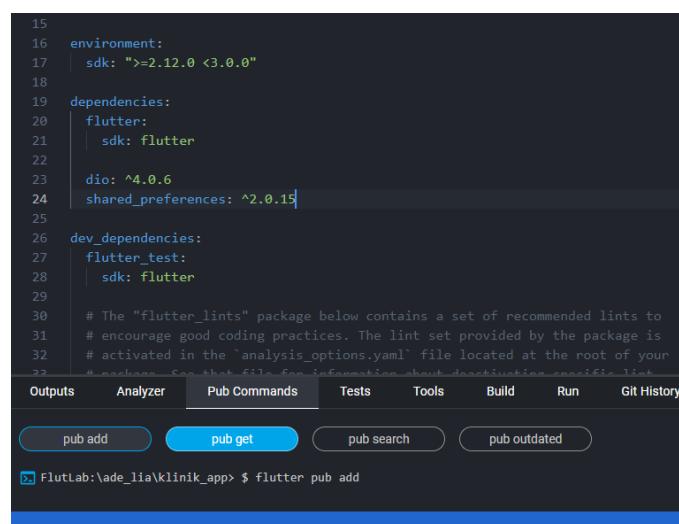
```
29. dependencies:
30.   flutter:
31.     sdk: flutter
32.
33.
34.   # The following adds the Cupertino Icons font to your application.
35.   # Use with the CupertinoIcons class for iOS style icons.
36.   cupertino_icons: ^1.0.2
37.   dio: ^4.0.6
38.   shared_preferences: ^2.0.15
39.
40. dev_dependencies:
41.   flutter_test:
42.     sdk: flutter
```

Simpan file tersebut, dan secara otomatis VS Code akan mengunduh depedencies tersebut, jika tidak berhasil, untuk mengunduh depedencies atau package yang telah ditambahkan, dapat dilakukan dengan membuka **CommandPrompt** kemudian masuk ke folder projek dan ketikkan **flutter pub get** kemudian tekan Enter

```
[perpustakaan_app] flutter pub get
Running "flutter pub get" in perpustakaan_app...
exit code 0
```

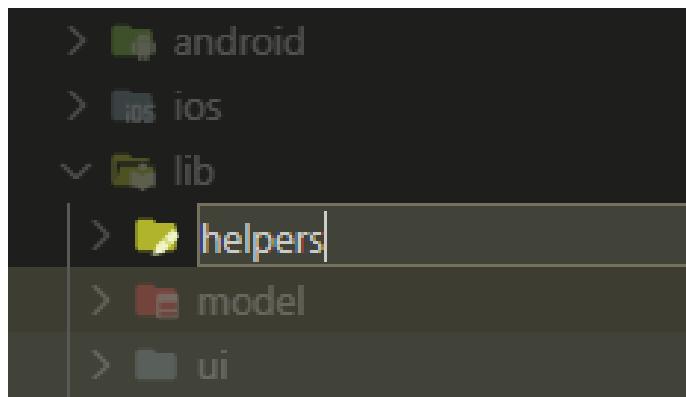
Pada Editor Online (flutlab.io)

Buka file **pubspec.yaml** dibagian bawah **dependencies**: tambahkan depedensi **dio** kemudian simpan, dibagian bawah editor terdapat tab **Pub Commands**, pilih tab tersebut kemudian klik **pub get**



Membuat Class Untuk Menyimpan Token

Pada bagian ini kita akan membuat class yang akan menyimpan token dan data pengguna saat berhasil login. Buat sebuah folder dengan nama helpers



Kemudian pada folder helpers buat sebuah file dengan nama **user_info.dart** dan masukkan kode berikut

```
1. import 'package:shared_preferences/shared_preferences.dart';
2.
3. const String TOKEN = "token";
4. const String USER_ID = "userID";
5. const String USERNAME = "username";
6.
7. class UserInfo {
8.   Future setToken(String value) async {
9.     final SharedPreferences pref = await SharedPreferences.getInstance();
10.    return pref.setString(TOKEN, value);
11.  }
12.
13. Future<String?> getToken() async {
14.   final SharedPreferences pref = await SharedPreferences.getInstance();
15.   return pref.getString(TOKEN);
16. }
17.
18. Future setUserID(String value) async {
19.   final SharedPreferences pref = await SharedPreferences.getInstance();
20.   return pref.setString(USER_ID, value);
21. }
22.
23. Future<String> getUserID() async {
24.   final SharedPreferences pref = await SharedPreferences.getInstance();
25.   return pref.getString(USER_ID).toString();
26. }
27.
28. Future setUsername(String value) async {
29.   final SharedPreferences pref = await SharedPreferences.getInstance();
30.   return pref.setString(USERNAME, value);
31. }
32.
33. Future<String> getUsername() async {
34.   final SharedPreferences pref = await SharedPreferences.getInstance();
35.   return pref.getString(USERNAME).toString();
36. }
37.
38. Future<void> logout() async {
39.   final SharedPreferences pref = await SharedPreferences.getInstance();
40.   pref.clear();
41. }
42. }
```

Mengarahkan Halaman Awal ke Login

Buka kembali file **main.dart** kita akan memodifikasi file tersebut dengan kondisi jika belum login maka akan membuka halaman login, namun jika sudah login maka akan membuka halaman beranda

```
1. import 'package:flutter/material.dart';
2. import '/helpers/user_info.dart';
3. import '/ui/beranda.dart';
4. import '/ui/login.dart';
5.
6. Future<void> main() async {
7.   WidgetsFlutterBinding.ensureInitialized();
8.   var token = await UserInfo().getToken();
9.   print(token);
10.  runApp(MaterialApp(
11.    title: "Klinik APP",
12.    debugShowCheckedModeBanner: false,
13.    home: token == null ? Login() : Beranda(),
14.  ));
15. }
```

Persiapan Koneksi ke API

Buat sebuah fie dengan nama **api_client.dart** pada folder **helpers** kemudian ketikkan kode berikut

```
1. import 'package:dio/dio.dart';
2.
3. final Dio dio = Dio(BaseOptions(
4.   baseUrl: 'https://63588b60c27556d2893f7a12.mockapi.io/',
5.   connectTimeout: 5000,
6.   receiveTimeout: 3000));
7.
8. class ApiClient {
9.   Future<Response> get(String path) async {
10.     try {
11.       final response = await dio.get(Uri.encodeFull(path));
12.       return response;
13.     } on DioError catch (e) {
14.       throw Exception(e.message);
15.     }
16.   }
17.
18.   Future<Response> post(String path, dynamic data) async {
19.     try {
20.       final response = await dio.post(Uri.encodeFull(path), data: data);
21.       return response;
22.     } on DioError catch (e) {
23.       throw Exception(e.message);
24.     }
25.   }
26.
27.   Future<Response> put(String path, dynamic data) async {
28.     try {
29.       final response = await dio.put(Uri.encodeFull(path), data: data);
30.       return response;
31.     } on DioError catch (e) {
32.       throw Exception(e.message);
33.     }
34. }
```

```

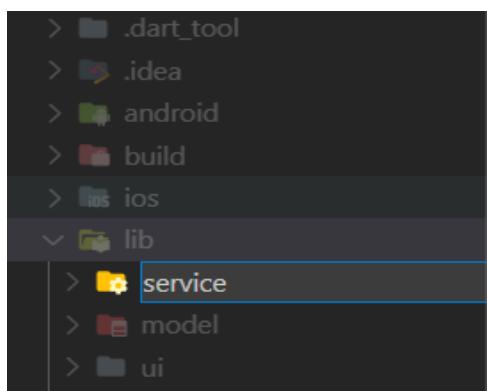
34. }
35.
36. Future<Response> delete(String path) async {
37.   try {
38.     final response = await dio.delete(Uri.encodeFull(path));
39.     return response;
40.   } on DioError catch (e) {
41.     throw Exception(e.message);
42.   }
43. }
44. }

```

Pada baris 4 (`baseUrl: 'https://63588b60c27556d2893f7a12.mockapi.io/'`) Sesuaikan dengan API Endpoint pada mockapi.io yang telah dibuat sebelumnya

Membuat Service Login

Buat sebuah folder dengan nama **service** pada folder **lib**



Pada bagian ini kita akan membuat dummy login. Buat sebuah file pada folder **service** dengan nama **login_service.dart** kemudian ketikkan kode berikut

```

1. import '../helpers/user_info.dart';
2.
3. class LoginService {
4.   Future<bool> login(String username, String password) async {
5.     bool isLogin = false;
6.     if (username == 'admin' && password == 'admin') {
7.       await UserInfo().setToken("admin");
8.       await UserInfo().setUserID("1");
9.       await UserInfo().setUsername("admin");
10.      isLogin = true;
11.    }
12.    return isLogin;
13.  }
14. }

```

Kemudian buka file **login.dart** yang ada pada folder **ui**. Pada bagian fungsi tombol login kita ubah menjadi seperti berikut

```

Widget _tombolLogin() {
  return Container(
    width: MediaQuery.of(context).size.width,
    child: ElevatedButton(
      child: Text("Login"),
      onPressed: () async {

```

```

String username = _usernameCtrl.text;
String password = _passwordCtrl.text;
await LoginService().login(username, password).then((value) {
    if (value == true) {
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (context) => Beranda()));
    } else {
        AlertDialog alertDialog = AlertDialog(
            content: const Text("Username atau Password Tidak Valid"),
            actions: [
                ElevatedButton(
                    onPressed: () {
                        Navigator.pop(context);
                    },
                    child: const Text("OK"),
                    style: ElevatedButton.styleFrom(primary: Colors.green),
                ),
            ],
        );
        showDialog(
            context: context, builder: (context) => alertDialog);
    }
});
)));
}
}

```

Membuat Service Poli

Buka file **poli.dart** pada folder **model**, kemudian kita akan menambahkan dua buah method yaitu

```
factory Poli.fromJson(Map<String, dynamic> json) =>
    Poli(id: json["id"], namaPoli: json["nama_poli"]);
```

Dan

```
Map<String, dynamic> toJson() => {"nama_poli": namaPoli};
```

Sehingga secara keseluruhan kode menjadi

```

1. class Poli {
2.     String? id;
3.     String namaPoli;
4.
5.     Poli({this.id, required this.namaPoli});
6.
7.     factory Poli.fromJson(Map<String, dynamic> json) =>
8.         Poli(id: json["id"], namaPoli: json["nama_poli"]);
9.
10.    Map<String, dynamic> toJson() => {"nama_poli": namaPoli};
11. }
```

Kemudian buat sebuah file pada folder **service** dengan nama **poli_service.dart** dan ketikkan kode berikut

```
1. import 'package:dio/dio.dart';
2. import '../helpers/api_client.dart';
3. import '../model/poli.dart';
4.
5. class PoliService {
6.   Future<List<Poli>> listData() async {
7.     final Response response = await ApiClient().get('poli');
8.     final List data = response.data as List;
9.     List<Poli> result = data.map((json) => Poli.fromJson(json)).toList();
10.    return result;
11.  }
12.
13. Future<Poli> simpan(Poli poli) async {
14.   var data = poli.toJson();
15.   final Response response = await ApiClient().post('poli', data);
16.   Poli result = Poli.fromJson(response.data);
17.   return result;
18. }
19.
20. Future<Poli> ubah(Poli poli, String id) async {
21.   var data = poli.toJson();
22.   final Response response = await ApiClient().put('poli/${id}', data);
23.   print(response);
24.   Poli result = Poli.fromJson(response.data);
25.   return result;
26. }
27.
28. Future<Poli> getById(String id) async {
29.   final Response response = await ApiClient().get('poli/${id}');
30.   Poli result = Poli.fromJson(response.data);
31.   return result;
32. }
33.
34. Future<Poli> hapus(Poli poli) async {
35.   final Response response = await ApiClient().delete('poli/${poli.id}');
36.   Poli result = Poli.fromJson(response.data);
37.   return result;
38. }
39. }
```

TUGAS

Buat kode untuk Service Pegawai dan Pasien

PERTEMUAN 12

Mengambil List Poli dari API

Buka kembali file **poli_page.dart** yang ada pada folder **ui**, kita akan menambahkan fungsi

```
Stream<List<Poli>> getList() async* {
    List<Poli> data = await PoliService().listData();
    yield data;
}
```

Dan melakukan beberapa perubahan pada bagian **body** menjadi

```
body: StreamBuilder(
    stream: getList(),
    builder: (context, AsyncSnapshot snapshot) {
        if (snapshot.hasError) {
            return Text(snapshot.error.toString());
        }
        if (snapshot.connectionState != ConnectionState.done) {
            return Center(
                child: CircularProgressIndicator(),
            );
        }
        if (!snapshot.hasData &&
            snapshot.connectionState == ConnectionState.done) {
            return Text('Data Kosong');
        }

        return ListView.builder(
            itemCount: snapshot.data.length,
            itemBuilder: (context, index) {
                return PoliItem(poli: snapshot.data[index]);
            },
        );
    },
),
```

Adapun modifikasi keseluruhan menjadi seperti kode berikut

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import '../service/poli_service.dart';
4. import 'poli_detail.dart';
5. import 'poli_form.dart';
6. import 'poli_item.dart';
7. import '../widget/sidebar.dart';
8.
9. class PoliPage extends StatefulWidget {
10.   const PoliPage({Key? key}) : super(key: key);
11.   _PoliPageState createState() => _PoliPageState();
12. }
13.
14. class _PoliPageState extends State<PoliPage> {
15.   Stream<List<Poli>> getList() async* {
16.     List<Poli> data = await PoliService().listData();
17.     yield data;
18.   }
19.
20.   @override
21.   Widget build(BuildContext context) {
22.     return Scaffold(
```

```
23.      drawer: Sidebar(),
24.      appBar: AppBar(
25.        title: const Text("Data Poli"),
26.        actions: [
27.          GestureDetector(
28.            child: const Icon(Icons.add),
29.            onTap: () {
30.              Navigator.push(
31.                context,
32.                MaterialPageRoute(builder: (context) =>
33.                  PoliForm()));
34.            },
35.          ),
36.        ],
37.        body: StreamBuilder(
38.          stream: getList(),
39.          builder: (context, AsyncSnapshot snapshot) {
40.            if (snapshot.hasError) {
41.              return Text(snapshot.error.toString());
42.            }
43.            if (snapshot.connectionState != ConnectionState.done) {
44.              return Center(
45.                child: CircularProgressIndicator(),
46.              );
47.            }
48.            if (!snapshot.hasData &&
49.                snapshot.connectionState == ConnectionState.done) {
50.              return Text('Data Kosong');
51.            }
52.            return ListView.builder(
53.              itemCount: snapshot.data.length,
54.              itemBuilder: (context, index) {
55.                return PoliItem(poli: snapshot.data[index]);
56.              },
57.            );
58.          },
59.        ),
60.      );
61.    }
62. }
```

Menambah Data Poli ke API

Buka file **poli_form.dart** pada folder **ui** pada bagian fungsi `_tombolSimpan()` kita ubah menjadi seperti berikut

```
_tombolSimpan() {
    return ElevatedButton(
        onPressed: () async {
            Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);
            await PoliService().simpan(poli).then((value) {
                Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                        builder: (context) => PoliDetail(poli: value)));
            });
        },
        child: const Text("Simpan"));
}
```

Adapun secara keseluruhan kodennya seperti berikut

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import '../service/poli_service.dart';
4. import 'poli_detail.dart';
5.
6. class PoliForm extends StatefulWidget {
7.   const PoliForm({Key? key}) : super(key: key);
8.   _PoliFormState createState() => _PoliFormState();
9. }
10.
11. class _PoliFormState extends State<PoliForm> {
12.   final _formKey = GlobalKey<FormState>();
13.   final _namaPoliCtrl = TextEditingController();
14.
15.   @override
16.   Widget build(BuildContext context) {
17.     return Scaffold(
18.       appBar: AppBar(title: const Text("Tambah Poli")),
19.       body: SingleChildScrollView(
20.         child: Form(
21.           key: _formKey,
22.           child: Column(
23.             children:      [_fieldNamaPoli()],
24.             _tombolSimpan(),
25.           ),
26.         ),
27.       );
28.   }
29.
30.   _fieldNamaPoli() {
31.     return TextField(
32.       decoration: const InputDecoration(labelText: "Nama Poli"),
33.       controller: _namaPoliCtrl,
34.     );
35. }
```

```
36.  
37. _tombolSimpan() {  
38.   return ElevatedButton(  
39.     onPressed: () async {  
40.       Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);  
41.       await PoliService().simpan(poli).then((value) {  
42.         Navigator.pushReplacement(  
43.           context,  
44.           MaterialPageRoute(  
45.             builder: (context) => PoliDetail(poli: value)));  
46.       });  
47.     },  
48.     child: const Text("Simpan"));  
49.   }  
50. }
```

TUGAS

Buat pula mengambil List serta menyimpan data ke API untuk tabel pegawai dan pasien

PERTEMUAN 13

Mengambil Detail Poli dari API

Buka file **poli_detail.dart** pada folder **ui**, kita akan menambahkan fungsi

```
Stream<Poli> getData() async* {
    Poli data = await PoliService().getById(widget.poli.id.toString());
    yield data;
}
```

Dan melakukan perubahan pada bagian body menjadi

```
body: StreamBuilder(
    stream: getData(),
    builder: (context, AsyncSnapshot snapshot) {
        if (snapshot.hasError) {
            return Text(snapshot.error.toString());
        }
        if (snapshot.connectionState != ConnectionState.done) {
            return Center(
                child: CircularProgressIndicator(),
            );
        }
        if (!snapshot.hasData &&
            snapshot.connectionState == ConnectionState.done) {
            return Text('Data Tidak Ditemukan');
        }
        return Column(
            children: [
                SizedBox(height: 20),
                Text(
                    "Nama Poli : ${snapshot.data.namaPoli}",
                    style: TextStyle(fontSize: 20),
                ),
                SizedBox(height: 20),
                Row(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: [_tombolUbah(), _tombolHapus()],
                )
            ],
        );
    },
),
```

Pada fungsi `_tombolUbah()` menjadi

```
_tombolUbah() {
    return StreamBuilder(
        stream: getData(),
        builder: (context, AsyncSnapshot snapshot) => ElevatedButton(
            onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) =>
                            PoliUpdateForm(poli: snapshot.data)));
            },
            style: ElevatedButton.styleFrom(primary: Colors.green),
            child: const Text("Ubah")));
}
```

Sehingga secara keseluruhan kodanya menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2. import '../service/poli_service.dart';
3. import 'poli_page.dart';
4. import 'poli_update_form.dart';
5. import '../model/poli.dart';
6.
7. class PoliDetail extends StatefulWidget {
8.   final Poli poli;
9.
10.  const PoliDetail({Key? key, required this.poli}) : super(key: key);
11.  _PoliDetailState createState() => _PoliDetailState();
12. }
13.
14. class _PoliDetailState extends State<PoliDetail> {
15.   Stream<Poli> getData() async* {
16.     Poli data = await PoliService().getById(widget.poli.id.toString());
17.     yield data;
18.   }
19.
20.   @override
21.   Widget build(BuildContext context) {
22.     return Scaffold(
23.       appBar: AppBar(title: Text("Detail Poli")),
24.       body: StreamBuilder(
25.         stream: getData(),
26.         builder: (context, AsyncSnapshot snapshot) {
27.           if (snapshot.hasError) {
28.             return Text(snapshot.error.toString());
29.           }
30.           if (snapshot.connectionState != ConnectionState.done) {
31.             return Center(
32.               child: CircularProgressIndicator(),
33.             );
34.           }
35.           if (!snapshot.hasData &&
36.               snapshot.connectionState == ConnectionState.done) {
37.             return Text('Data Tidak Ditemukan');
38.           }
39.           return Column(
40.             children: [
41.               SizedBox(height: 20),
42.               Text(
43.                 "Nama Poli : ${snapshot.data.namaPoli}",
44.                 style: TextStyle(fontSize: 20),
45.               ),
46.               SizedBox(height: 20),
47.               Row(
48.                 mainAxisAlignment: MainAxisAlignment.spaceEvenly,
49.                 children: [_tombolUbah(), _tombolHapus()],
50.               )
51.             ],
52.           );
53.         },
54.       ),
55.     );
56.   }
57.
58.   _tombolUbah() {
59.     return StreamBuilder(
60.       stream: getData(),
61.       builder: (context, AsyncSnapshot snapshot) => ElevatedButton(
62.         onPressed: () {
63.           Navigator.push(
```

```

64.        context,
65.        MaterialPageRoute(
66.            builder: (context) =>
67.                PoliUpdateForm(poli: snapshot.data)));
68.        },
69.        style: ElevatedButton.styleFrom(primary: Colors.green),
70.        child: const Text("Ubah")));
71.    }
72.
73.    _tombolHapus() {
74.        return ElevatedButton(
75.            onPressed: () {
76.                AlertDialog alertDialog = AlertDialog(
77.                    content: const Text("Yakin ingin menghapus data ini?"),
78.                    actions: [
79.                        ElevatedButton(
80.                            onPressed: () {
81.                                Navigator.pop(context);
82.                                Navigator.pushReplacement(context, MaterialPageRoute(builder:
83.                                    (context) => PoliPage())));
84.                            },
85.                            child: const Text("YA"),
86.                            style: ElevatedButton.styleFrom(primary: Colors.red),
87.                            ),
88.                        ElevatedButton(
89.                            onPressed: () {
90.                                Navigator.pop(context);
91.                            },
92.                            child: const Text("Tidak"),
93.                            style: ElevatedButton.styleFrom(primary: Colors.green),
94.                            )
95.                        ],
96.                    );
97.                showDialog(context: context, builder: (context) => alertDialog);
98.            },
99.            style: ElevatedButton.styleFrom(primary: Colors.red),
100.           child: const Text("Hapus"));
101.    }

```

Menghapus Data Poli dari API

Masih pada file **poli_detail.dart** pada folder **ui**, kita akan mengubah fungsi pada `_tombolHapus` menjadi

```

_tombolHapus() {
    return ElevatedButton(
        onPressed: () {
            AlertDialog alertDialog = AlertDialog(
                content: const Text("Yakin ingin menghapus data ini?"),
                actions: [
                    StreamBuilder(
                        stream: getData(),
                        builder: (context, AsyncSnapshot snapshot) => ElevatedButton(
                            onPressed: () async {
                                await PoliService()
                                    .hapus(snapshot.data)
                                    .then((value) {
                                Navigator.pop(context);
                                Navigator.pushReplacement(
                                    context,
                                    MaterialPageRoute(
                                        builder: (context) => PoliPage())));
                            });

```

```

        },
        child: const Text("YA"),
        style: ElevatedButton.styleFrom(primary: Colors.red),
    )),  

    ElevatedButton(  

        onPressed: () {  

            Navigator.pop(context);  

        },  

        child: const Text("Tidak"),
        style: ElevatedButton.styleFrom(primary: Colors.green),
    )  

],
);
showDialog(context: context, builder: (context) => alertDialog);
},  

style: ElevatedButton.styleFrom(primary: Colors.red),
child: const Text("Hapus"));
}

```

Sehingga secara keseluruhan kodennya menjadi seperti berikut

```

1. import 'package:flutter/material.dart';
2. import '../service/poli_service.dart';
3. import 'poli_page.dart';
4. import 'poli_update_form.dart';
5. import '../model/poli.dart';
6.
7. class PoliDetail extends StatefulWidget {
8.     final Poli poli;
9.
10.    const PoliDetail({Key? key, required this.poli}) : super(key: key);
11.    _PoliDetailState createState() => _PoliDetailState();
12. }
13.
14. class _PoliDetailState extends State<PoliDetail> {
15.     Stream<Poli> getData() async* {
16.         Poli data = await PoliService().getById(widget.poli.id.toString());
17.         yield data;
18.     }
19.
20.     @override
21.     Widget build(BuildContext context) {
22.         return Scaffold(
23.             appBar: AppBar(title: Text("Detail Poli")),
24.             body: StreamBuilder(
25.                 stream: getData(),
26.                 builder: (context, AsyncSnapshot snapshot) {
27.                     if (snapshot.hasError) {
28.                         return Text(snapshot.error.toString());
29.                     }
30.                     if (snapshot.connectionState != ConnectionState.done) {
31.                         return Center(
32.                             child: CircularProgressIndicator(),
33.                         );
34.                     }
35.                     if (!snapshot.hasData &&
36.                         snapshot.connectionState == ConnectionState.done) {
37.                         return Text('Data Tidak Ditemukan');
38.                     }
39.                     return Column(
40.                         children: [
41.                             SizedBox(height: 20),

```

```

42.             Text(
43.                 "Nama Poli : ${snapshot.data.namaPoli}",
44.                 style: TextStyle(fontSize: 20),
45.             ),
46.             SizedBox(height: 20),
47.             Row(
48.                 mainAxisAlignment: MainAxisAlignment.spaceEvenly,
49.                 children: [_tombolUbah(), _tombolHapus()],
50.             )
51.         ],
52.     );
53. },
54. ),
55. );
56. }
57.
58. _tombolUbah() {
59.     return StreamBuilder(
60.         stream: getData(),
61.         builder: (context, AsyncSnapshot snapshot) => ElevatedButton(
62.             onPressed: () {
63.                 Navigator.push(
64.                     context,
65.                     MaterialPageRoute(
66.                         builder: (context) =>
67.                             PoliUpdateForm(poli: snapshot.data)));
68.             },
69.             style: ElevatedButton.styleFrom(primary: Colors.green),
70.             child: const Text("Ubah")));
71. }
72.
73. _tombolHapus() {
74.     return ElevatedButton(
75.         onPressed: () {
76.             AlertDialog alertDialog = AlertDialog(
77.                 content: const Text("Yakin ingin menghapus data ini?"),
78.                 actions: [
79.                     StreamBuilder(
80.                         stream: getData(),
81.                         builder: (context, AsyncSnapshot snapshot) => ElevatedButton(
82.                             onPressed: () async {
83.                                 await PoliService()
84.                                     .hapus(snapshot.data)
85.                                     .then((value) {
86.                                         Navigator.pop(context);
87.                                         Navigator.pushReplacement(
88.                                             context,
89.                                             MaterialPageRoute(
90.                                                 builder: (context) => PoliPage()));
91.                                         });
92.                             },
93.                             child: const Text("YA"),
94.                             style: ElevatedButton.styleFrom(primary: Colors.red),
95.                         )),
96.                     ElevatedButton(
97.                         onPressed: () {
98.                             Navigator.pop(context);
99.                         },
100.                            child: const Text("Tidak"),
101.                            style: ElevatedButton.styleFrom(primary: Colors.green),
102.                        ),
103.                    ],
104.                );
105.                showDialog(context: context, builder: (context) => alertDialog);
106.            },
107.            style: ElevatedButton.styleFrom(primary: Colors.red),

```

```
108.           child: const Text("Hapus"));
109.       }
110.   }
```

Mengubah Data Poli dari API

Buka file `poli_update_form.dart` pada folder `ui`, kita akan menambahkan sebuah fungsi berikut

```
Future<Poli> getData() async {
    Poli data = await PoliService().getById(widget.poli.id.toString());
    setState(() {
        _namaPoliCtrl.text = data.namaPoli;
    });
    return data;
}
```

Kemudian pada bagian `initState` diubah menjadi

```
@override
void initState() {
    super.initState();
    getData();
}
```

Kemudian pada bagian `_tombolSimpan()` diubah menjadi

```
_tombolSimpan() {
    return ElevatedButton(
        onPressed: () async {
            Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);
            String id = widget.poli.id.toString();
            await PoliService().ubah(poli, id).then((value) {
                Navigator.pop(context);
                Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                        builder: (context) => PoliDetail(poli: value)));
            });
        },
        child: const Text("Simpan Perubahan"));
}
```

Adapun keseluruhan kode sebagai berikut

```
1. import 'package:flutter/material.dart';
2. import '../model/poli.dart';
3. import '../service/poli_service.dart';
4. import 'poli_detail.dart';
5.
6. class PoliUpdateForm extends StatefulWidget {
7.     final Poli poli;
8.
9.     const PoliUpdateForm({Key? key, required this.poli}) : super(key: key);
10.    _PoliUpdateFormState createState() => _PoliUpdateFormState();
11. }
12.
13. class _PoliUpdateFormState extends State<PoliUpdateForm> {
14.     final _formKey = GlobalKey<FormState>();
15.     final _namaPoliCtrl = TextEditingController();
16.
```

```

17. Future<Poli> getData() async {
18.   Poli data = await PoliService().getById(widget.poli.id.toString());
19.   setState(() {
20.     _namaPoliCtrl.text = data.namaPoli;
21.   });
22.   return data;
23. }
24.
25. @override
26. void initState() {
27.   super.initState();
28.   getData();
29. }
30.
31. @override
32. Widget build(BuildContext context) {
33.   return Scaffold(
34.     appBar: AppBar(title: const Text("Ubah Poli")),
35.     body: SingleChildScrollView(
36.       child: Form(
37.         key: _formKey,
38.         child: Column(
39.           children: [_fieldNamaPoli(), SizedBox(height: 20), _tombolSimpan()],
40.         ),
41.       ),
42.     ),
43.   );
44. }
45.
46. _fieldNamaPoli() {
47.   return TextField(
48.     decoration: const InputDecoration(labelText: "Nama Poli"),
49.     controller: _namaPoliCtrl,
50.   );
51. }
52.
53. _tombolSimpan() {
54.   return ElevatedButton(
55.     onPressed: () async {
56.       Poli poli = new Poli(namaPoli: _namaPoliCtrl.text);
57.       String id = widget.poli.id.toString();
58.       await PoliService().ubah(poli, id).then((value) {
59.         Navigator.pop(context);
60.         Navigator.pushReplacement(
61.           context,
62.           MaterialPageRoute(
63.             builder: (context) => PoliDetail(poli: value)));
64.       });
65.     },
66.     child: const Text("Simpan Perubahan"));
67.   }
68. }

```

TUGAS

Buat pula Detail, Menghapus data dan mengubah data dari API untuk tabel pegawai dan pasien

PERTEMUAN 14

Presentasi Projek

PERTEMUAN 15

Presentasi Projek

PERTEMUAN 16

Ujian Akhir Semester