



MODUL

MOBILE Programming



Tim Penyusun:

Ade Hendini, M.Kom

Agung Sasongko, M.Kom

Muhammad Sony Maulana, ST., M.Kom

UNIVERSITAS BINA SARANA INFORMATIKA

PROGRAM STUDI SISTEM INFORMASI

2022

KATA PENGANTAR

Puja dan puji syukur selalu kami panjatkan kehadiran Allah Swt yang telah memberikan semua nikmatnya sehingga penulis berhasil menyelesaikan modul ajar 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 mempelajarai 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, Maret 2022

Tim Penyusun

Daftar Isi

Aplikasi Mobile dengan Flutter	6
Menyiapkan perangkat	6
Install Git.....	6
Install JDK.....	6
Install Android Studio.....	11
Install Flutter.....	16
Konfigurasi Android Studio dengan Flutter	22
Membuat dan menjalankan projek.....	23
Membuat dan menjalankan projek dengan VSCode dan Handphone Android	30
Install VSCode sebagai alternatif editor	30
Membuat projek flutter dengan VSCode	31
Menjalankan aplikasi dengan Handphone Android	34
Struktur Folder Flutter	41
Membuat Hello World	42
Membuat Widget Column	48
Membuat Widget Row.....	50
StatelessWidget dan StatefulWidget	51
Membuat Form	52
Pemisahan Widget Kedalam fungsi-fungsi.....	57
Membuat Detail Produk.....	58
Membuat fungsi tombol simpan dan menampilkan data pada Detail Produk	59
Membuat ListView Produk.....	62
Membuat Route (Pindah Halaman)	64
Pemisahan Widget ke dalam Class StatelessWidget	66
Menampilkan Detail Produk saat ListView diklik	68
Membuat projek flutter yang terhubung dengan API.....	72
Apa itu API	72
Arsitektur API	73
Membuat projek Toko API (Restful API).....	73
Installasi Apache, MySql dan PHP (XAMPP)	73
Install Postman	78
API SPEC.....	79
Pembuatan Database.....	82
Installasi CodeIgniter 4 sebagai Restful API	84

Membuat hasil response.....	86
Registrasi.....	87
Membuat model Registrasi.....	87
Membuat controller Registrasi	88
Menambah route Registrasi.....	88
Login	90
Membuat model Member	90
Membuat model Login.....	90
Membuat controller Login	91
Menambahkan route Login.....	92
Mencoba Rest	92
CRUD Produk	92
Membuat model Produk	92
Membuat controller produk	92
Mencoba Rest	95
Membuat projek flutter tokokita	97
Membuat Model.....	97
Login	98
Registrasi.....	99
Produk.....	99
Membuat Halaman	99
Registrasi.....	100
Login	104
Form Produk	107
Detail Produk	109
Tampil List Produk.....	111
Membuat Helper Modul	115
Menambahkan depedencies.....	115
Membuat Class Token.....	115
Http request.....	117
Membuat Bloc	120
Registrasi.....	121
Login	122
Logout.....	122
Produk.....	122

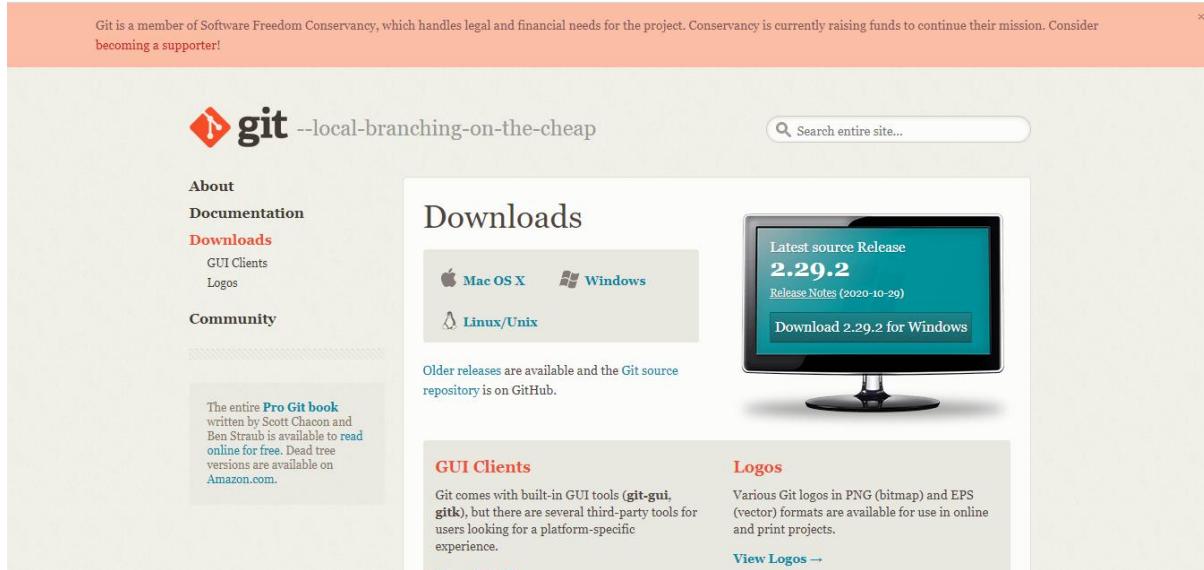
Menyatukan Fungsionalitas	124
Membuat Common Dialog Widget	124
Modifikasi main.dart	127
Modifikasi registrasi_page.dart.....	128
Modifikasi login_page.dart (fungsi login).....	132
Modifikasi produk_page.dart.....	136
Memodifikasi Form Produk (produk_form.dart).....	141

Aplikasi Mobile dengan Flutter

Menyiapkan perangkat

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

Java Development Kit builds, from Oracle

Ready for use: JDK 15, JMC 7

Early access: JDK 17, JDK 16, JMC 8, Lanai, Loom, Metropolis, Panama, & Valhalla

Reference implementations: Java SE 15, 14, 13, 12, 11, 10, 9, 8, & 7

ORACLE®

© 2021 Oracle Corporation and/or its affiliates
Terms of Use · Privacy · Trademarks

Pilih **JDK 15**, kemudian download file zip untuk windows, jika menggunakan windows

[jdk.java.net](https://jdk.java.net/15/) **JDK 15.0.2 General-Availability Release**

This page provides production-ready open-source builds of the Java Development Kit, version 15, an implementation of the Java SE 15 Platform under the GNU General Public License, version 2, with the Classpath Exception.

Commercial builds of JDK 15.0.2 from Oracle, under a non-open-source license, can be found at the Oracle Technology Network.

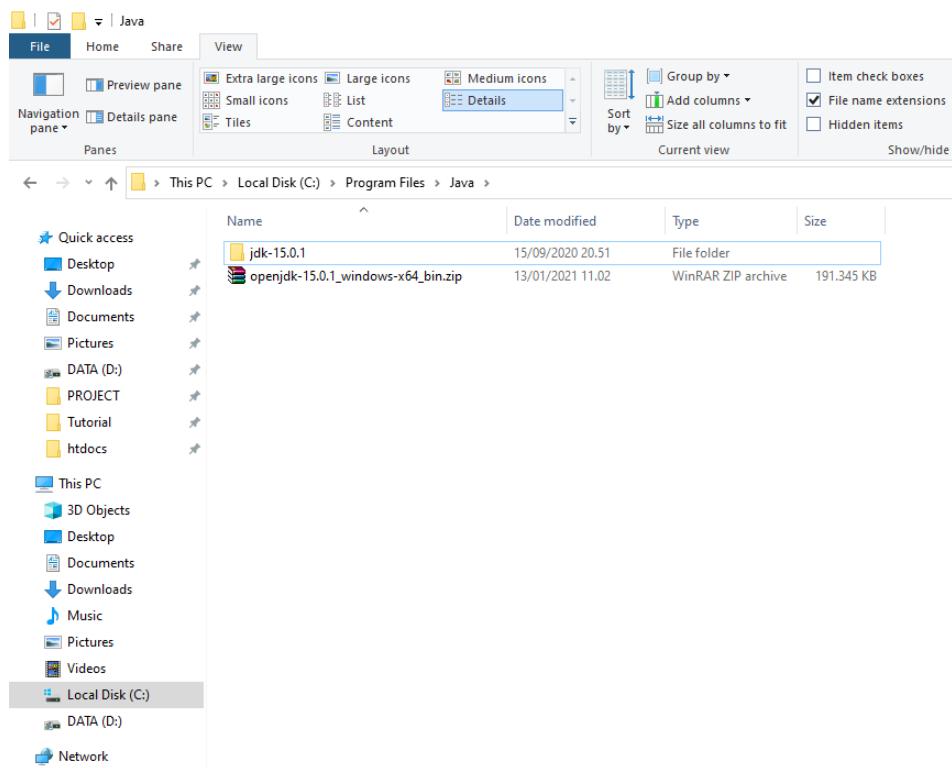
Documentation

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

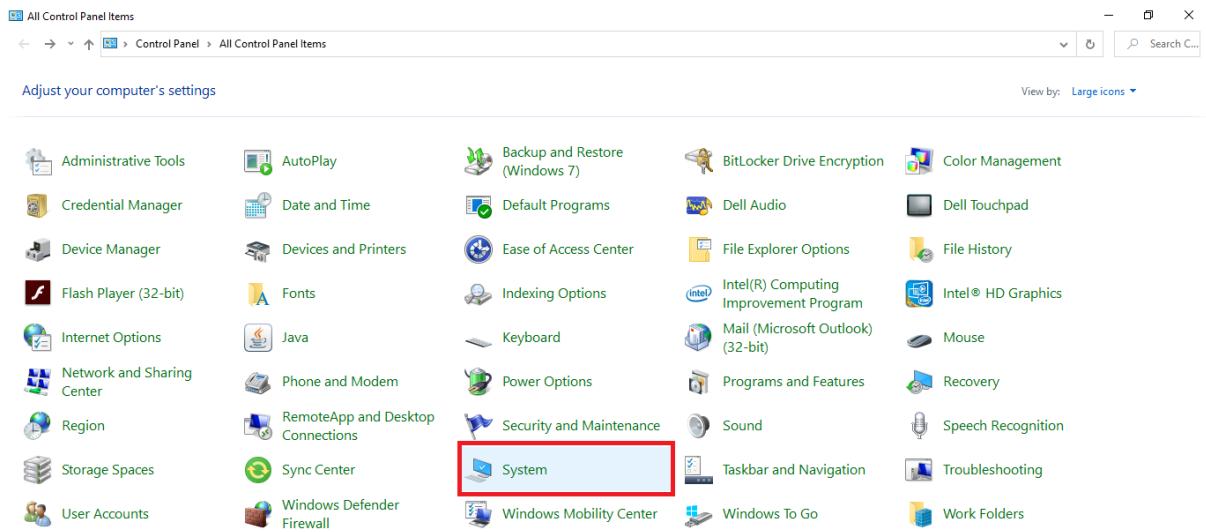
Builds

Linux/AArch64	tar.gz (sha256)	170507166 bytes
Linux/x64	tar.gz (sha256)	195340587
macOS/x64	tar.gz (sha256)	192067136
Windows/x64	zip (sha256)	195939486

Kemudian extrak berkas file tersebut pada laptop/komputer



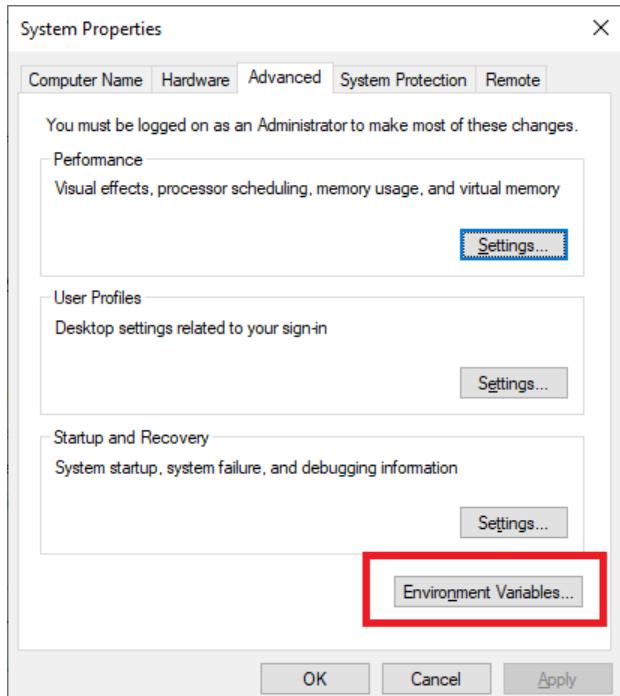
Kemudian buka **Control Panel**, pilih "System"



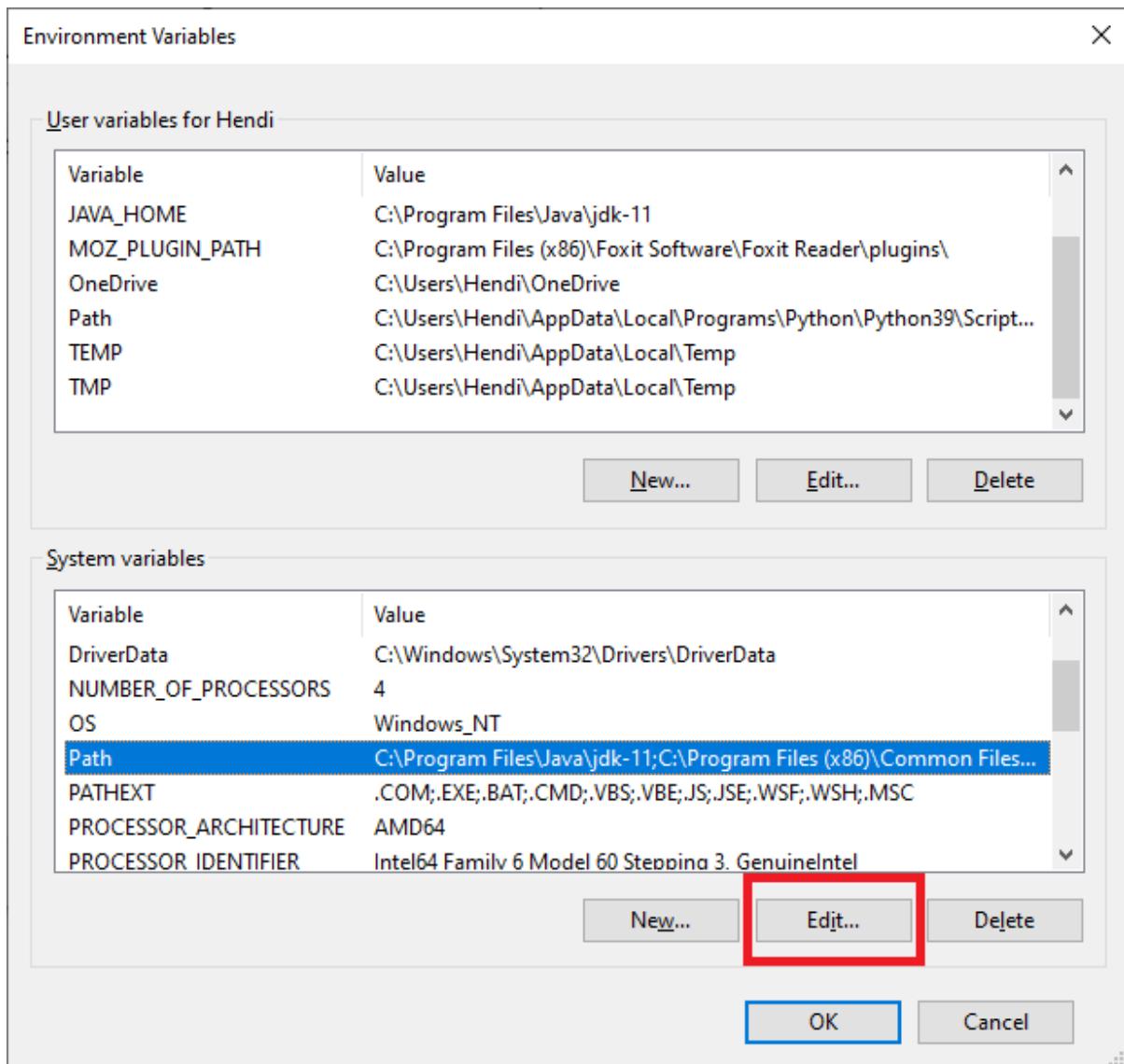
Kemudian pilih "Advanced System Setting"



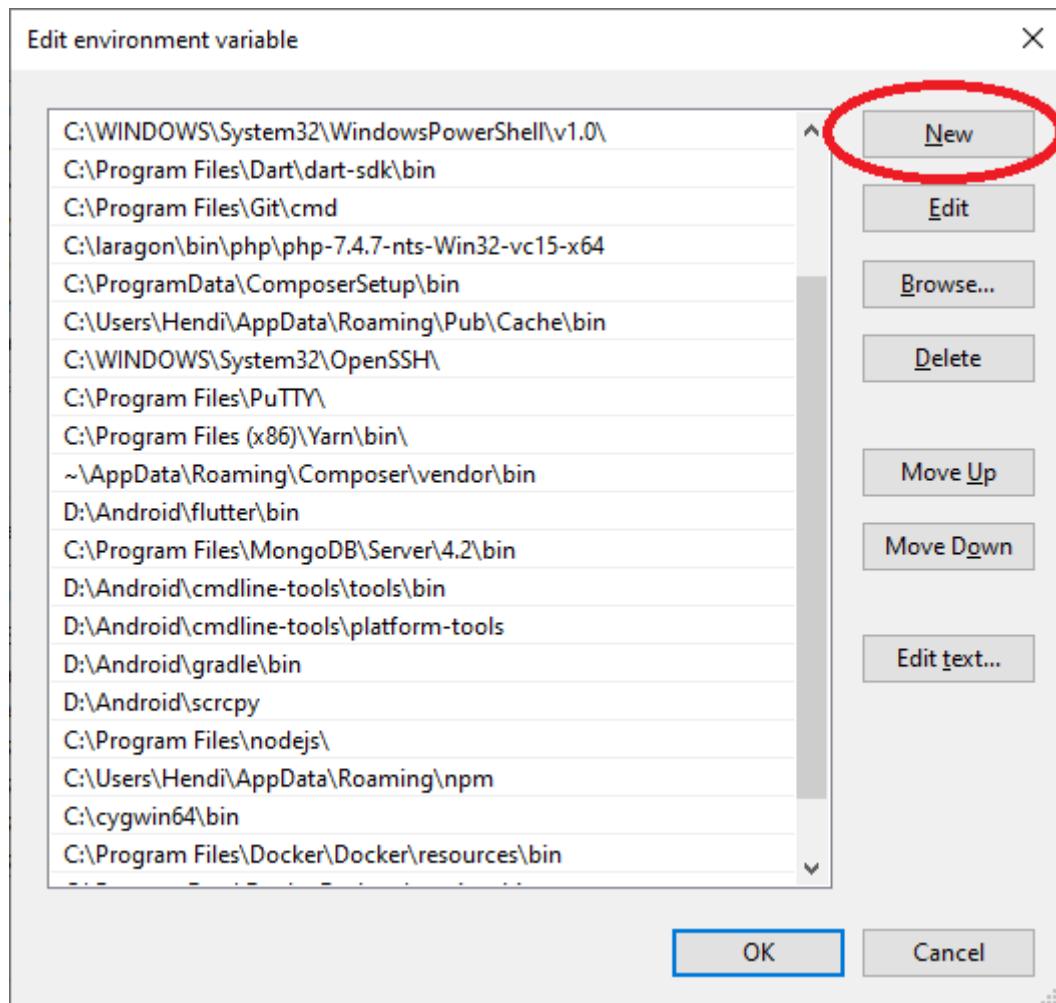
Kemudian klik tombol "Environtment Variables"



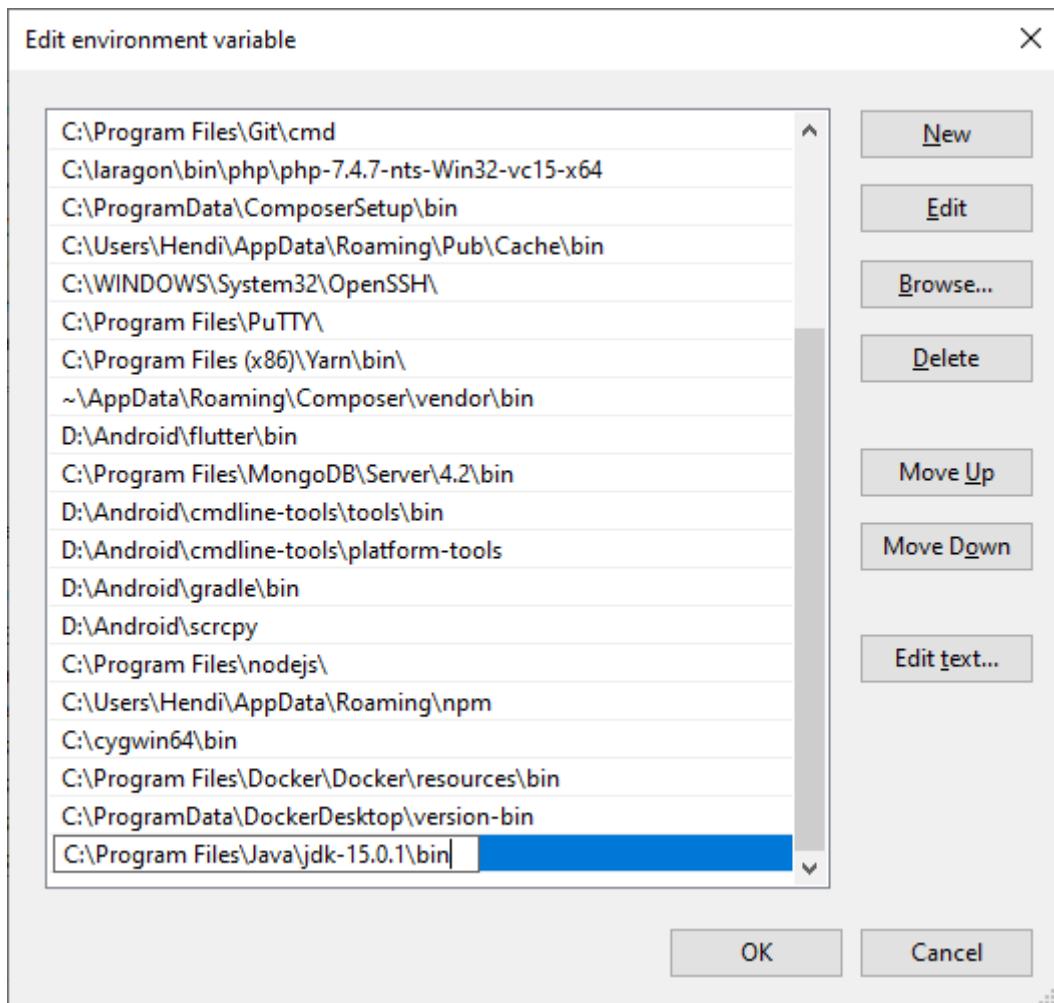
Pilih Path pada bagian System Variables kemudian Klik tombol "Edit"



Kemudian klik tombol “New”



Kemudian masukkan alamat folder bin pada jdk yang telah kita ekstrak dalam hal ini misalnya
“C:\Program Files\Java\jdk-15.0.1\bin”

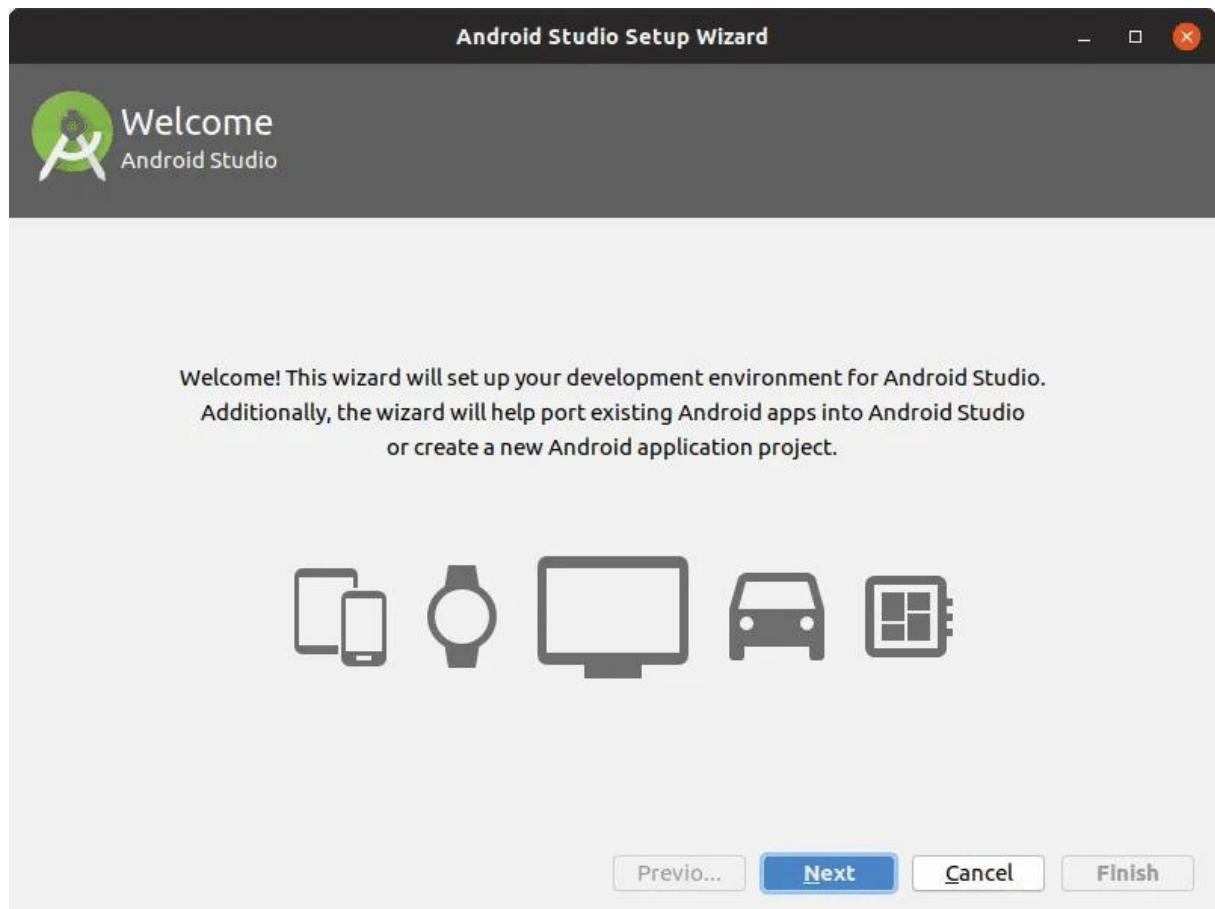


Biasanya agar JDK dapat berfungsi perlu dilakukan restart laptop/komputer. Untuk memeriksa apakah installasi berhasil, buka command prompt kemudian ketikkan **java -version** atau **javac -version**

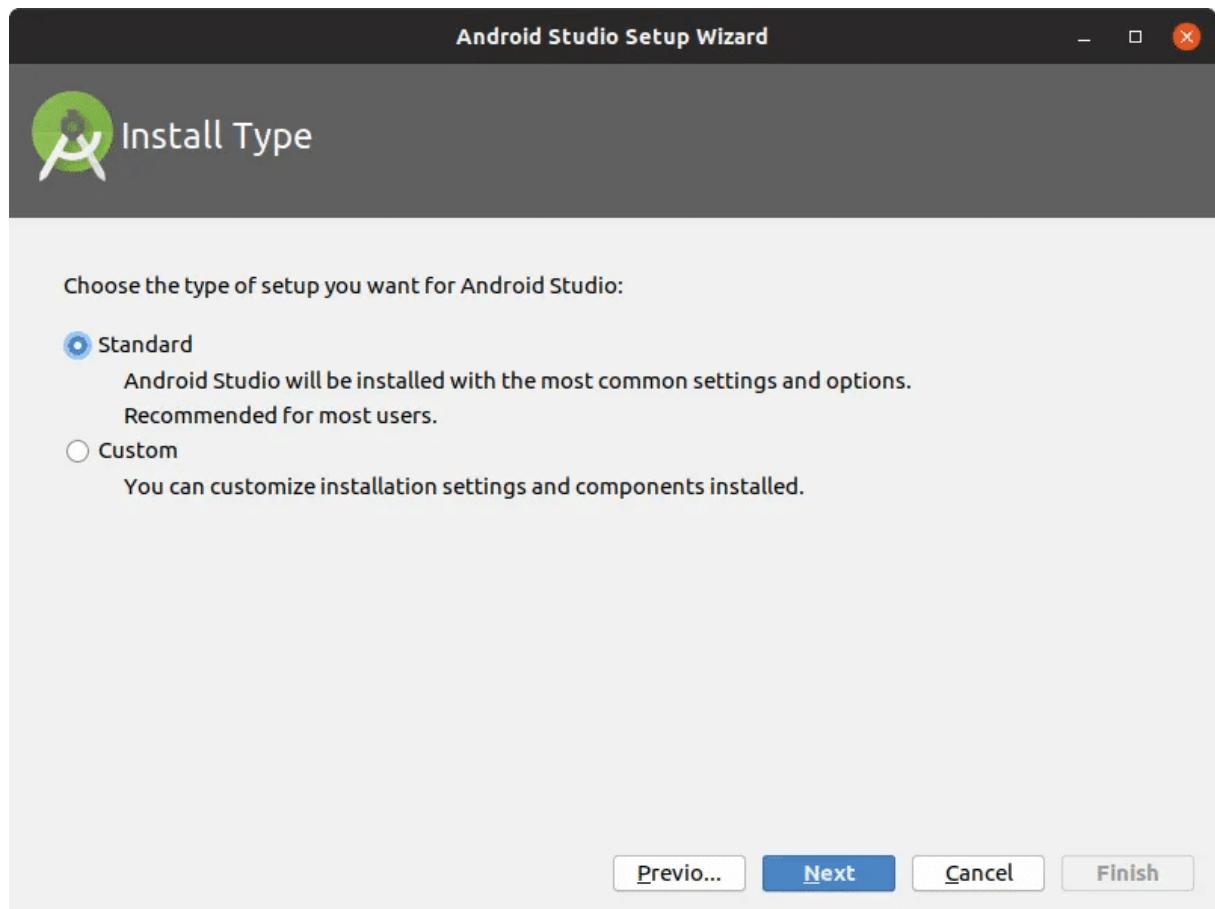
```
PS C:\> java -version
java version "1.8.0_281"
Java(TM) SE Runtime Environment (build 1.8.0_281-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode)
PS C:\> javac -version
javac 15.0.1
PS C:\> |
```

Install Android Studio

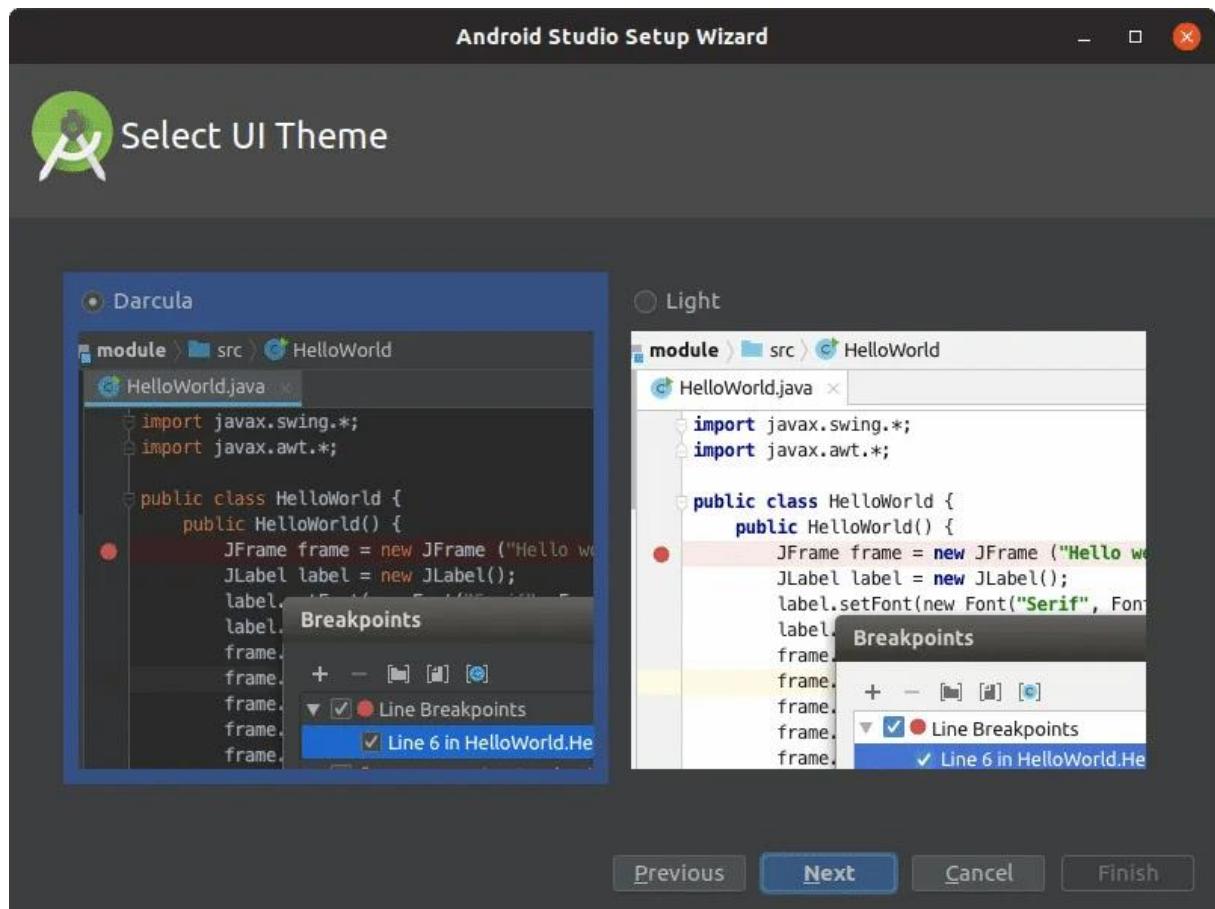
Android studio dapat diunduh pada laman <https://developer.android.com/studio>. Setelah diunduh klik dua kali pada file yang telah diunduh tersebut, kemudian lakukan installasi dengan mengikuti langkah-langkah yang telah disediakan



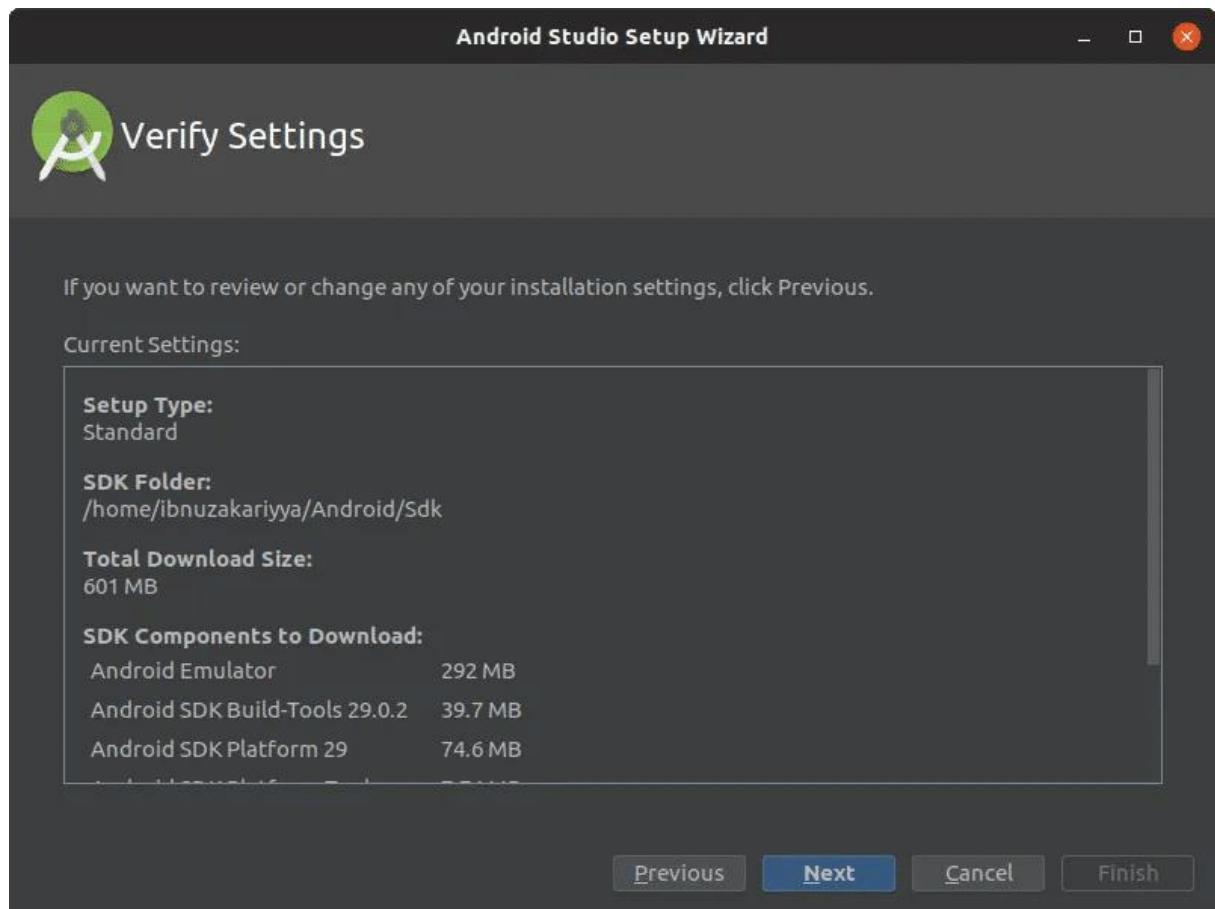
Kemudian pilih tipe standar dan klik next



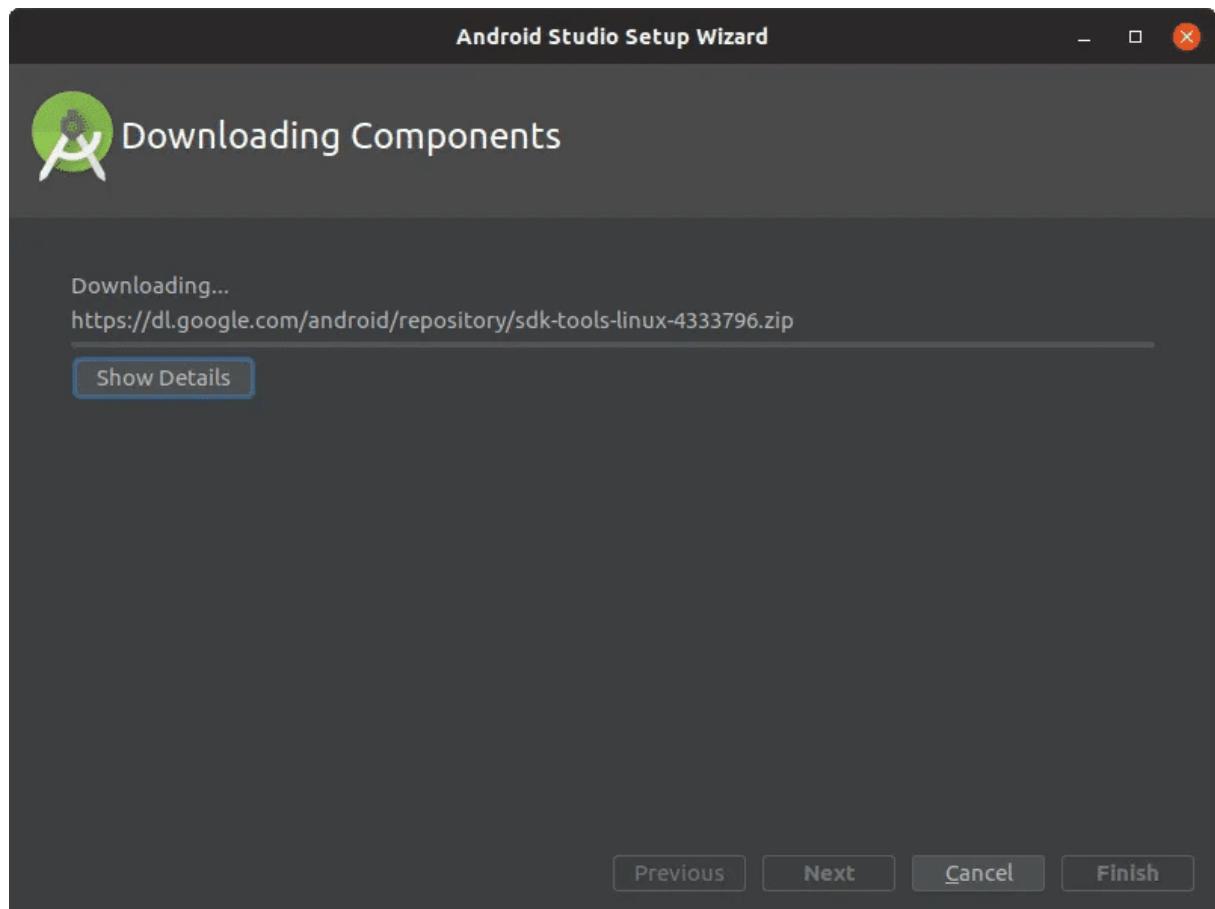
Pilih tema tampilan kemudian klik next



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"

Migrate your packages to null safety!

Get started

1. Install

2. Set up an editor

3. Test drive

4. Write your first app

5. Learn more

From another platform?

- Flutter for Android devs
- Flutter for iOS devs
- Flutter for React Native devs
- Flutter for web devs
- Flutter for Xamarin.Forms devs

<https://flutter.dev/docs/get-started/install/windows>

Docs Showcase Community

Get started

Install

Docs > Get started > Install

Select the operating system on which you are installing Flutter:

Windows macOS Linux Chrome OS

Important: If you're in China, first read [Using Flutter in China](#).

Kemudian pilih **flutter_windows_** untuk mengunduh file flutter

Get started

1. Install

2. Set up an editor

3. Test drive

4. Write your first app

5. Learn more

From another platform?

- Flutter for Android devs
- Flutter for iOS devs
- Flutter for React Native devs
- Flutter for web devs
- Flutter for Xamarin.Forms devs
- Introduction to declarative UI

Dart language overview

https://storage.googleapis.com/flutter_infra/releases/stable/windows/flutter_windows_1.22.5-stable.zip

Docs Showcase Community

Get the Flutter SDK

- Download the following installation bundle to get the latest stable release of the Flutter SDK:

[flutter_windows_1.22.5-stable.zip](#)

For other release channels, and older builds, see the [SDK releases](#) page.

- Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\src\flutter`).

Warning: Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the [Flutter repo](#) on GitHub, and change branches or tags as needed. For example:

```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

Contents

System requirements

Get the Flutter SDK

Update your path

Run flutter doctor

Android setup

Install Android Studio

Set up your Android device

Set up the Android emulator

Web setup

Next step

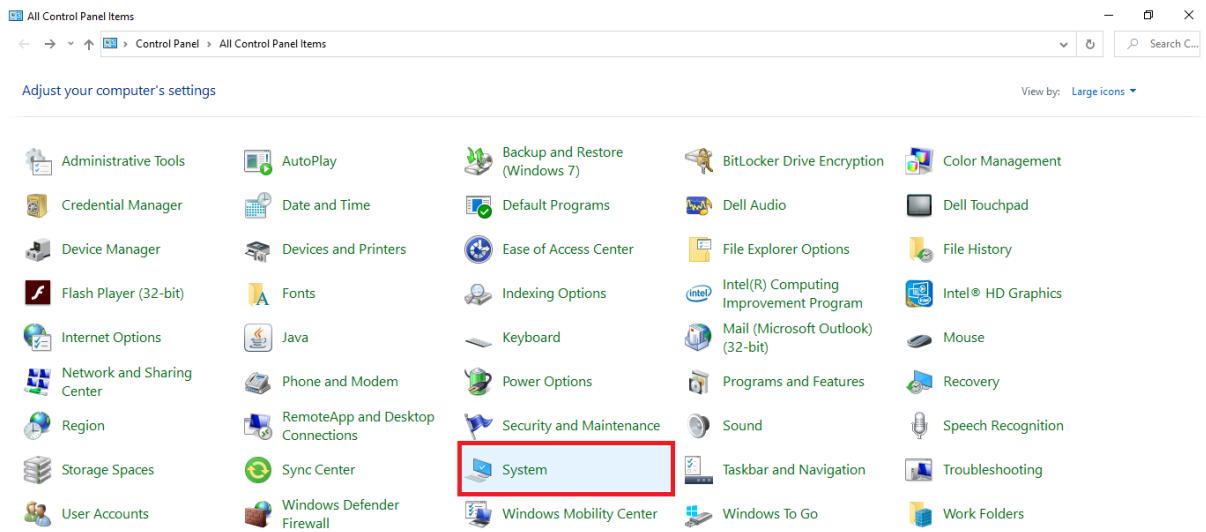
Kemudian ekstrak file zip flutter misalnya di "D:/Android"

Clipboard Organize

This PC > DATA (D:) > Android

flutter

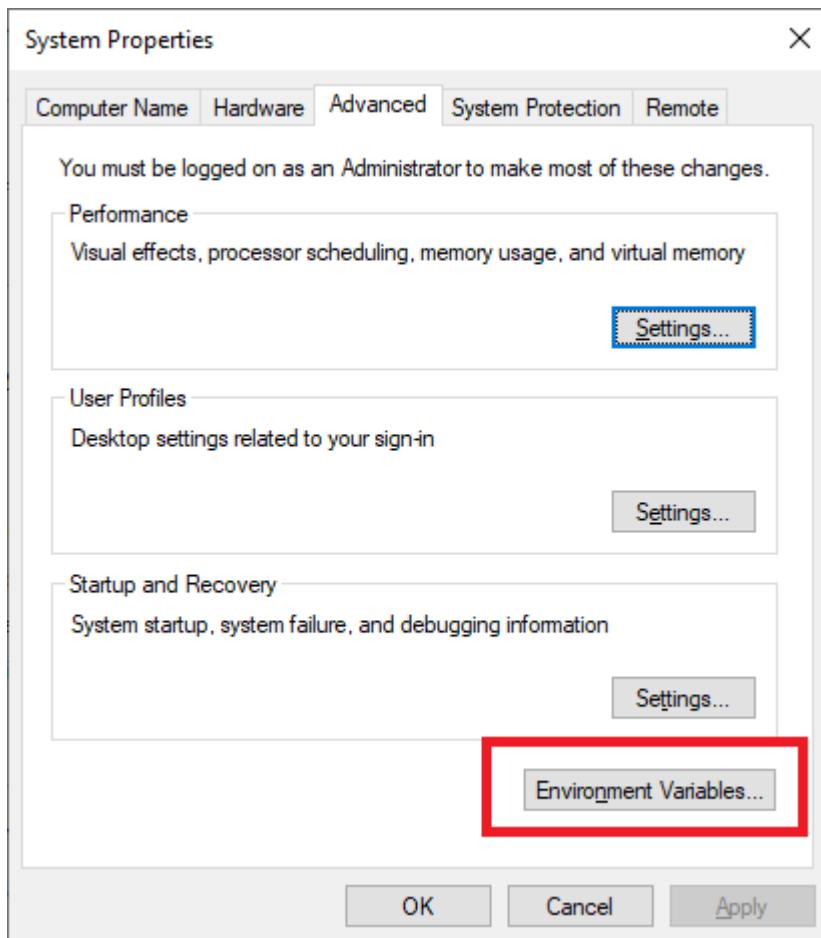
Kemudian buka **Control Panel**, pilih "System"



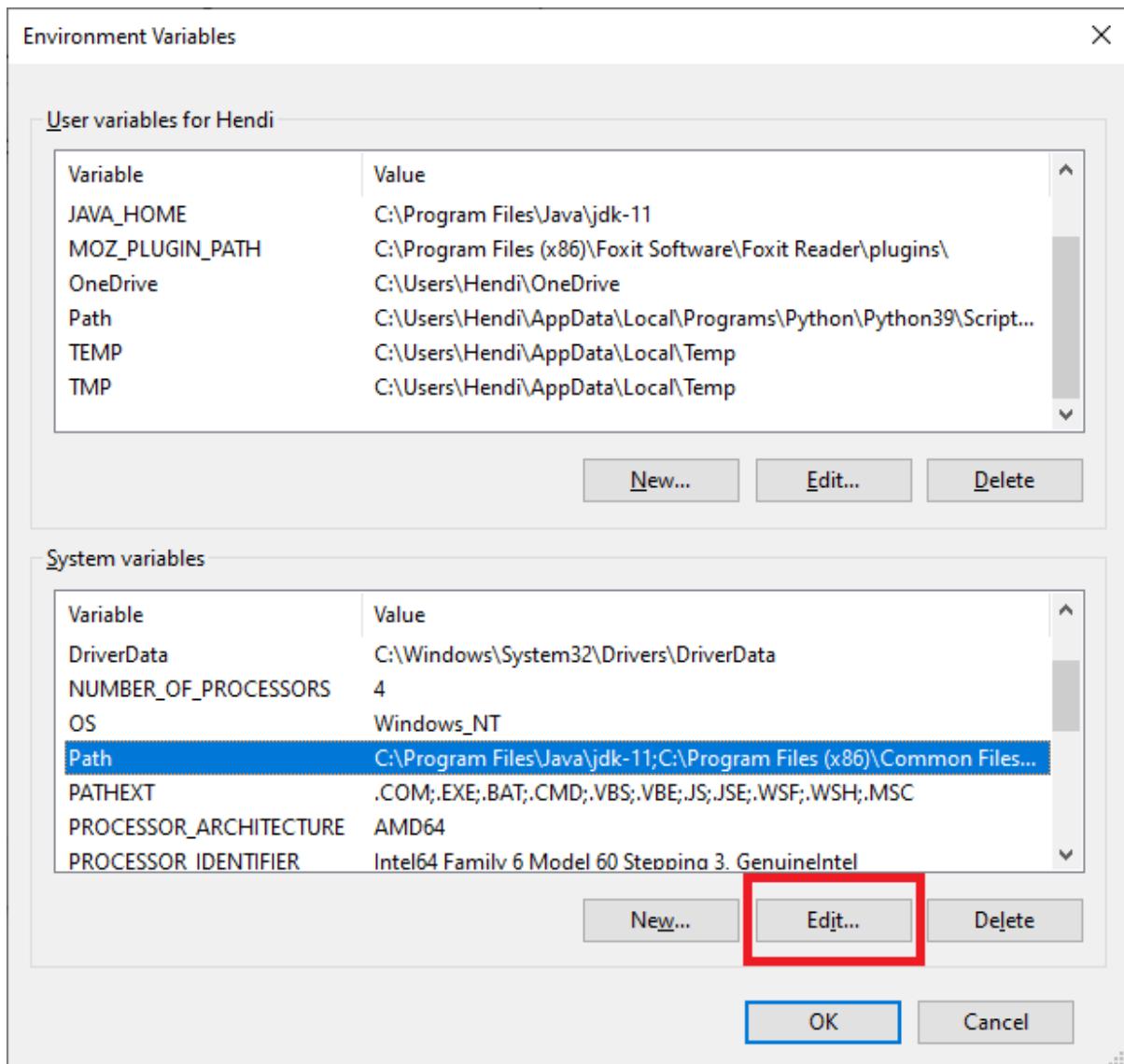
Kemudian pilih "Advanced System Setting"



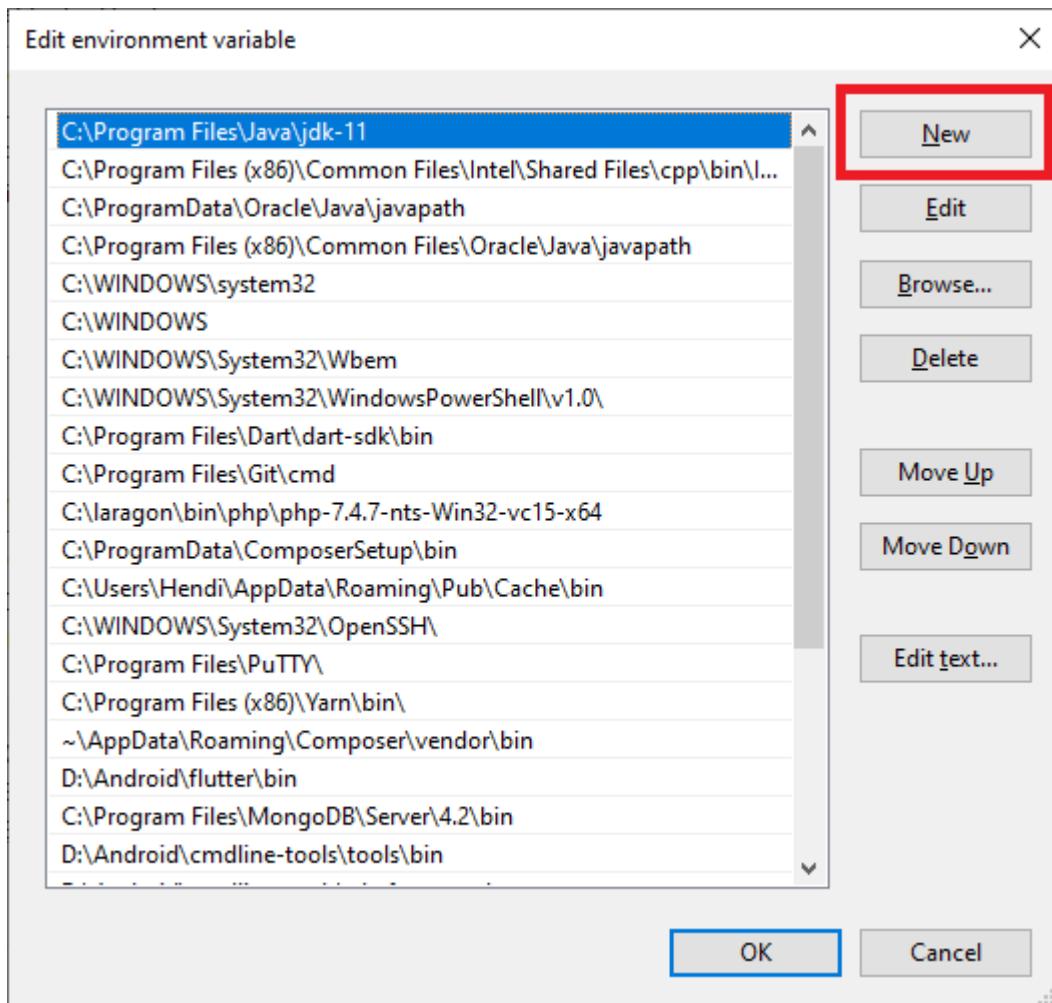
Kemudian klik tombol "Environtment Variables"



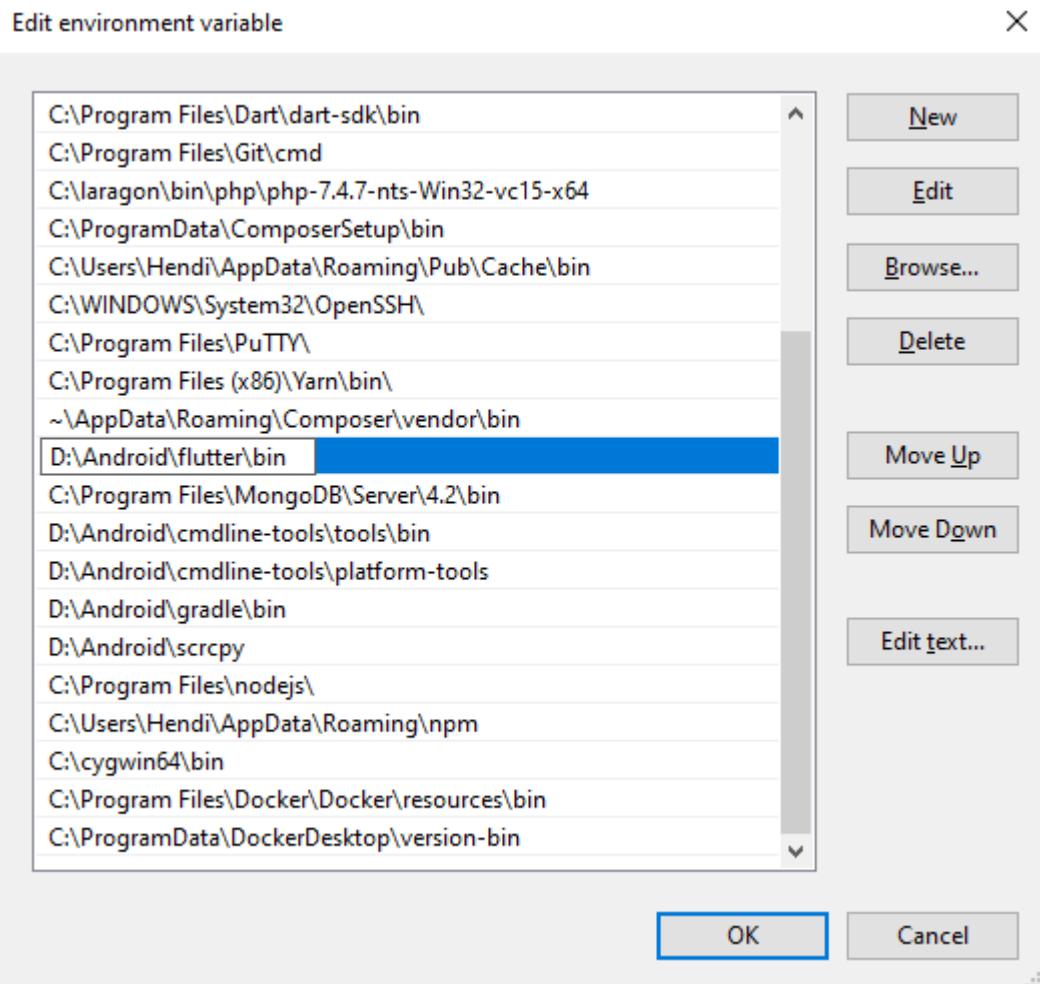
Kemudian pilih pada “System Variables” pilih “Path” dan klik tombol “Edit”



Kemudian klik tombol “New”

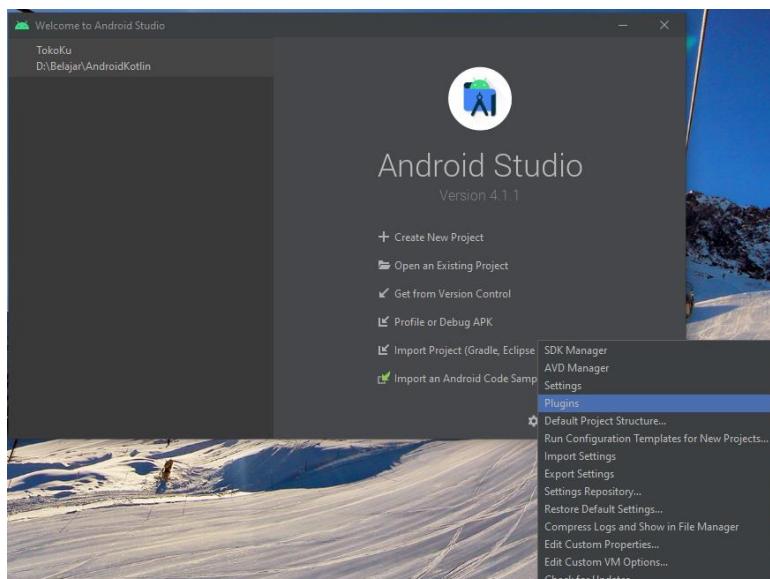


Kemudian masukkan alamat folder bin pada flutter yang telah kita ekstrak dalam hal ini misalnya "D:\Android\flutter\bin"

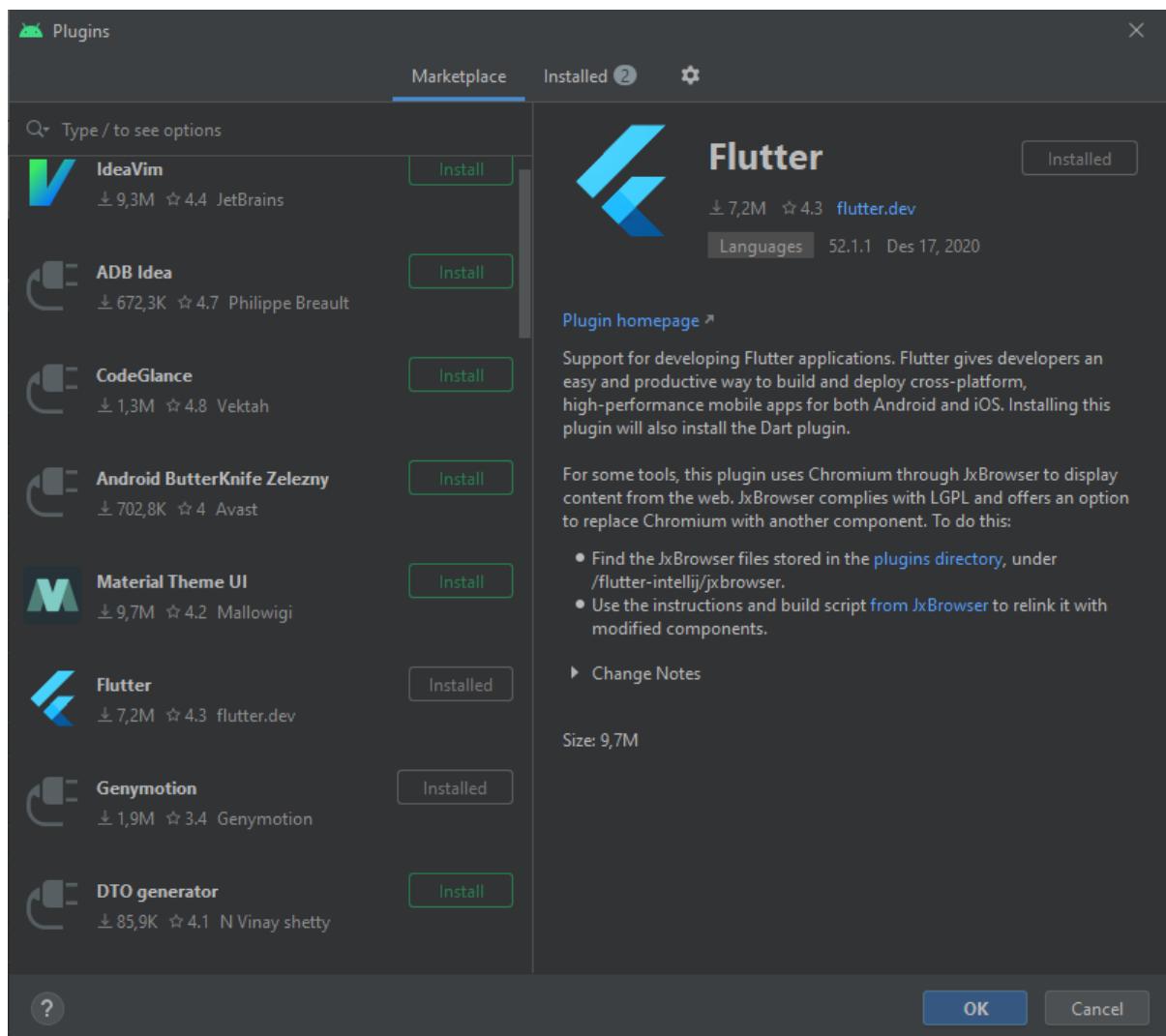


Konfigurasi Android Studio dengan Flutter

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



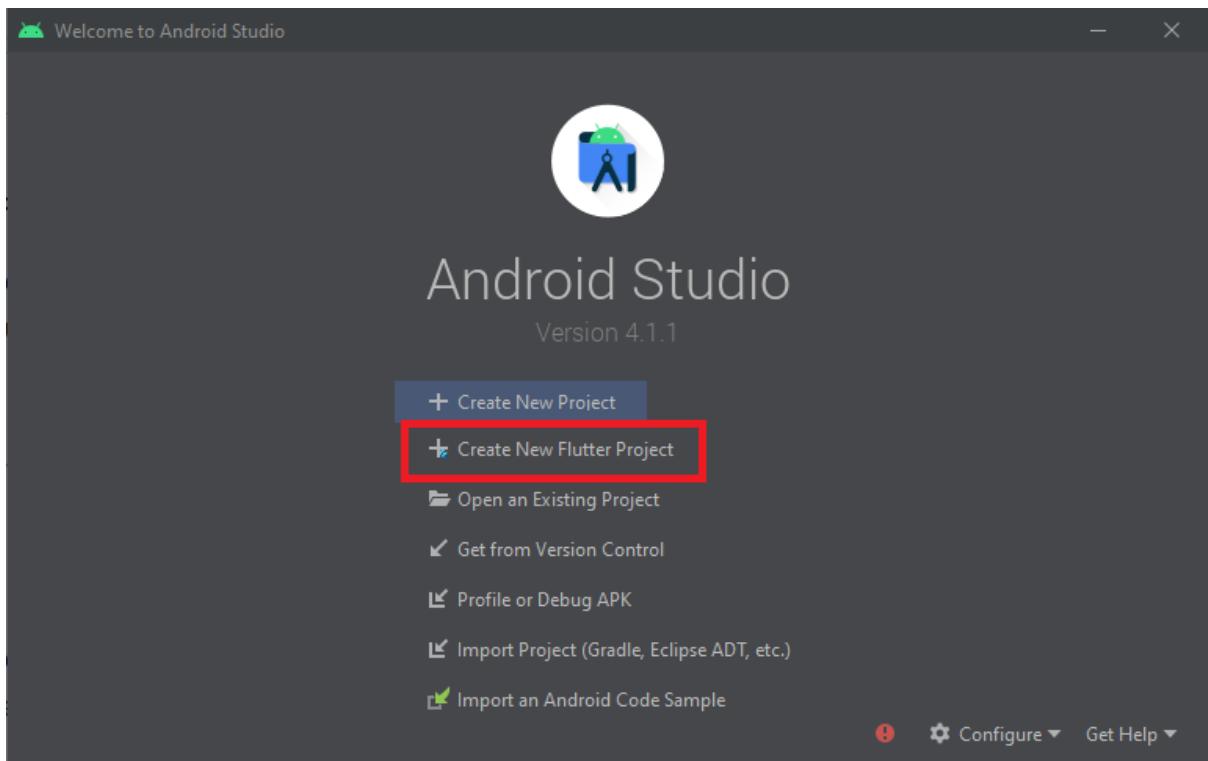
Pilih “Flutter” kemudian klik tombol “Install”



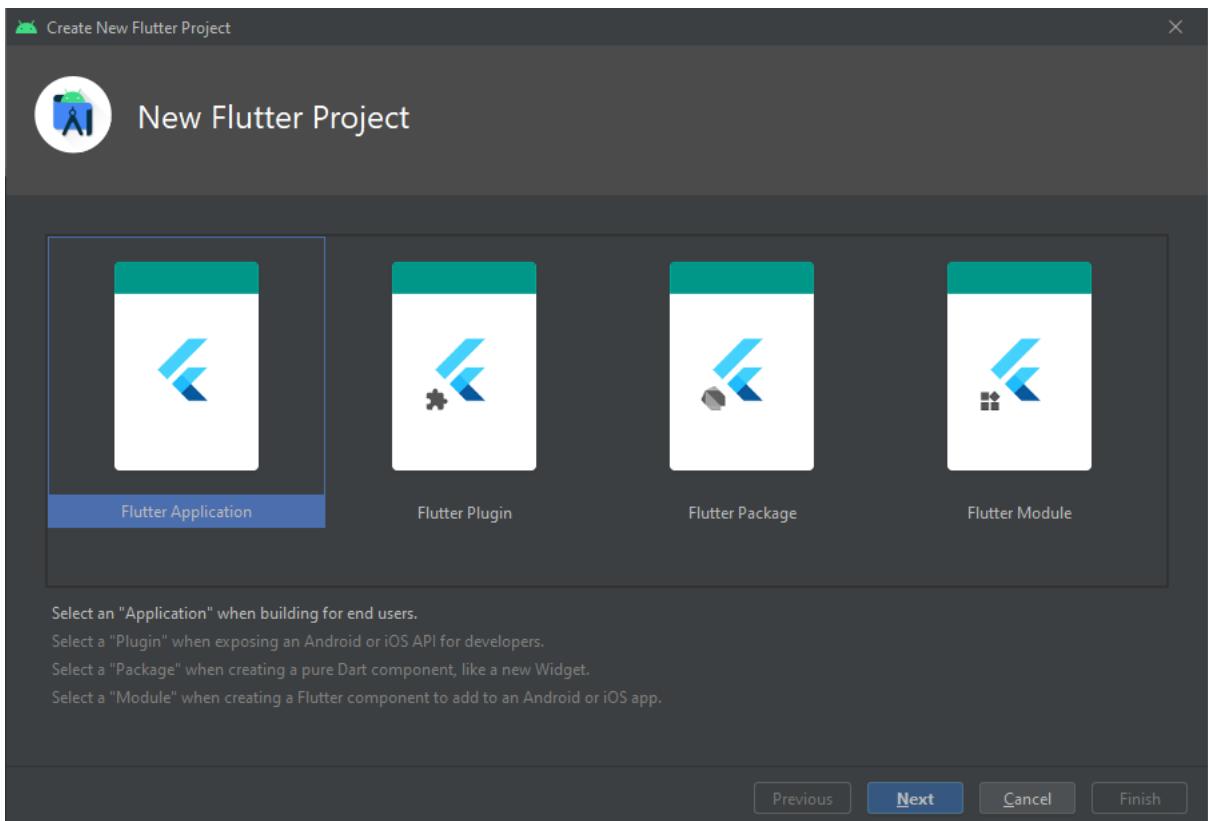
Setelah selesai, restart/tutup Android Studio

Membuat dan menjalankan projek

Jalankan Android Studio kemudian pilih “Create New Flutter Project”



Kemudian pilih “Flutter Application” dan klik “Next”



Kemudian tentukan :

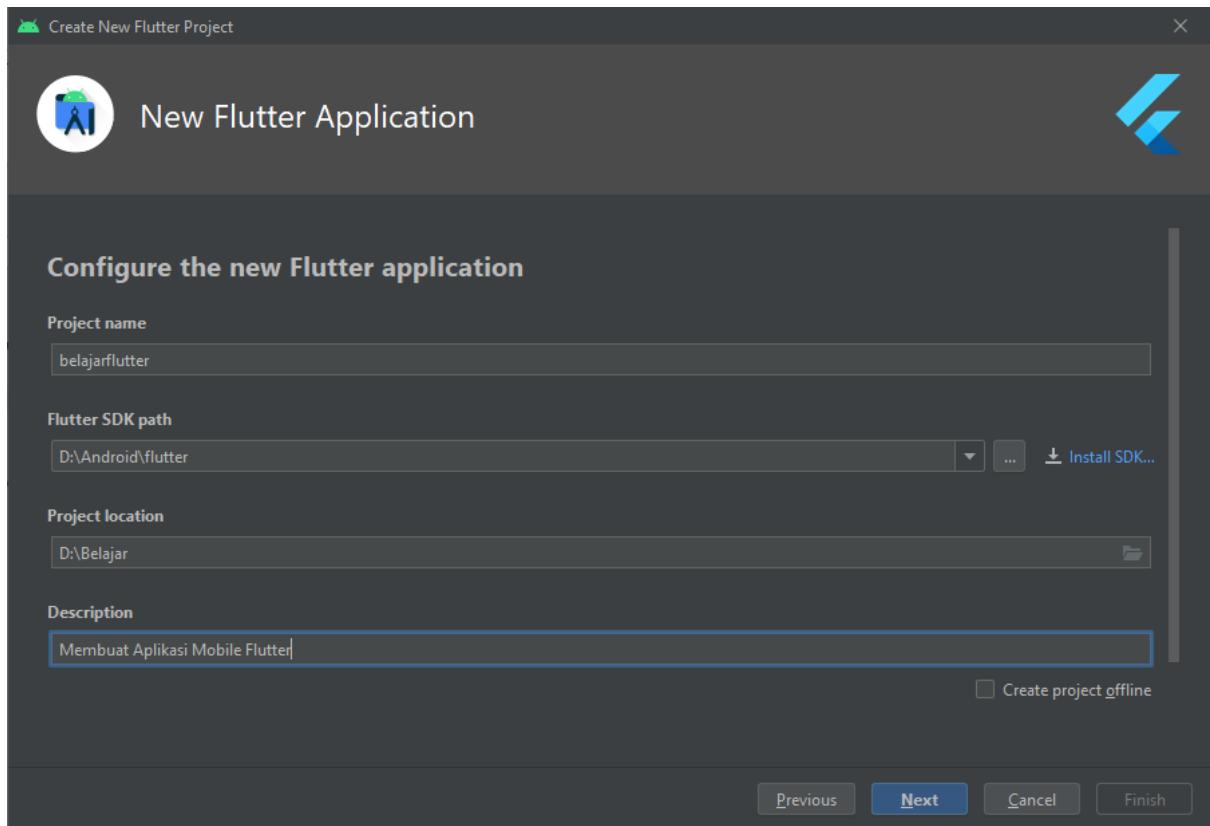
Project Name : belajarflutter

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

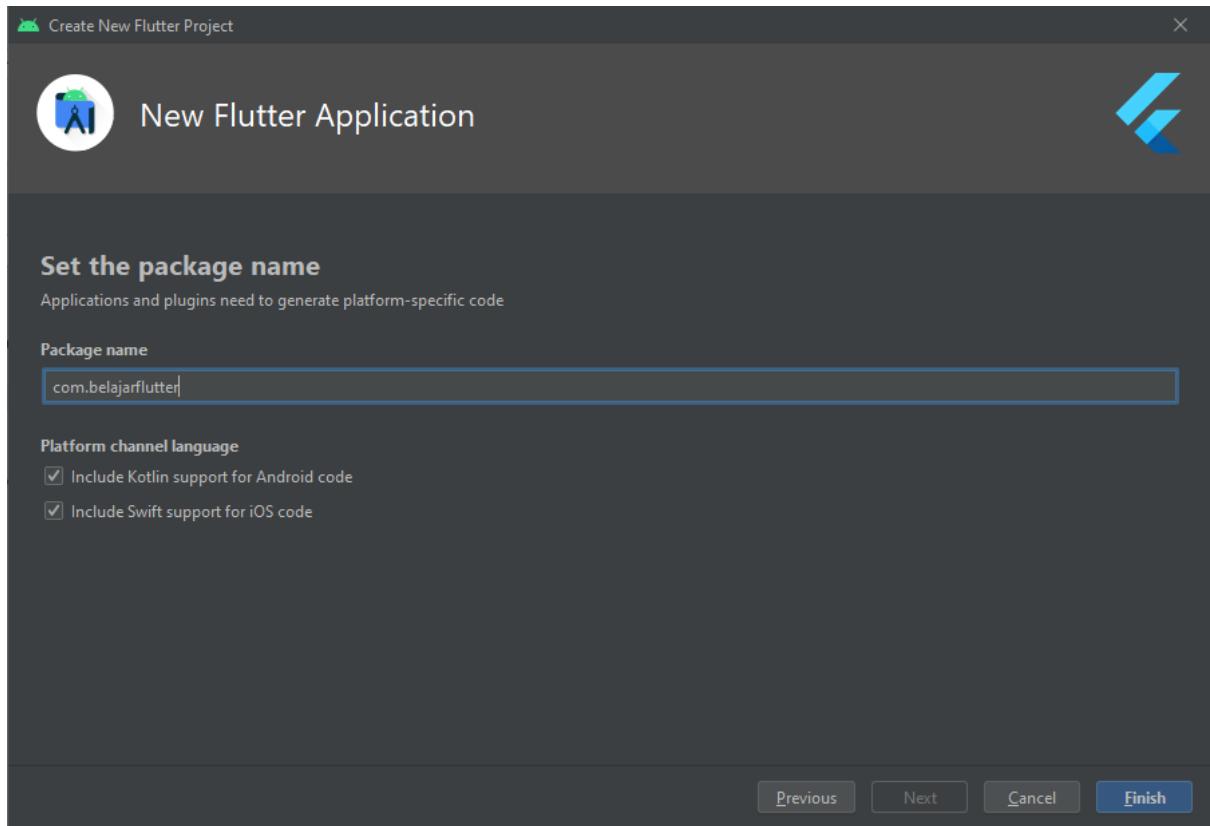
Project location : D:\Belajar (bebas memasukkan dimanapun)

Description : Membuat Aplikasi Mobile Flutter (bebas)

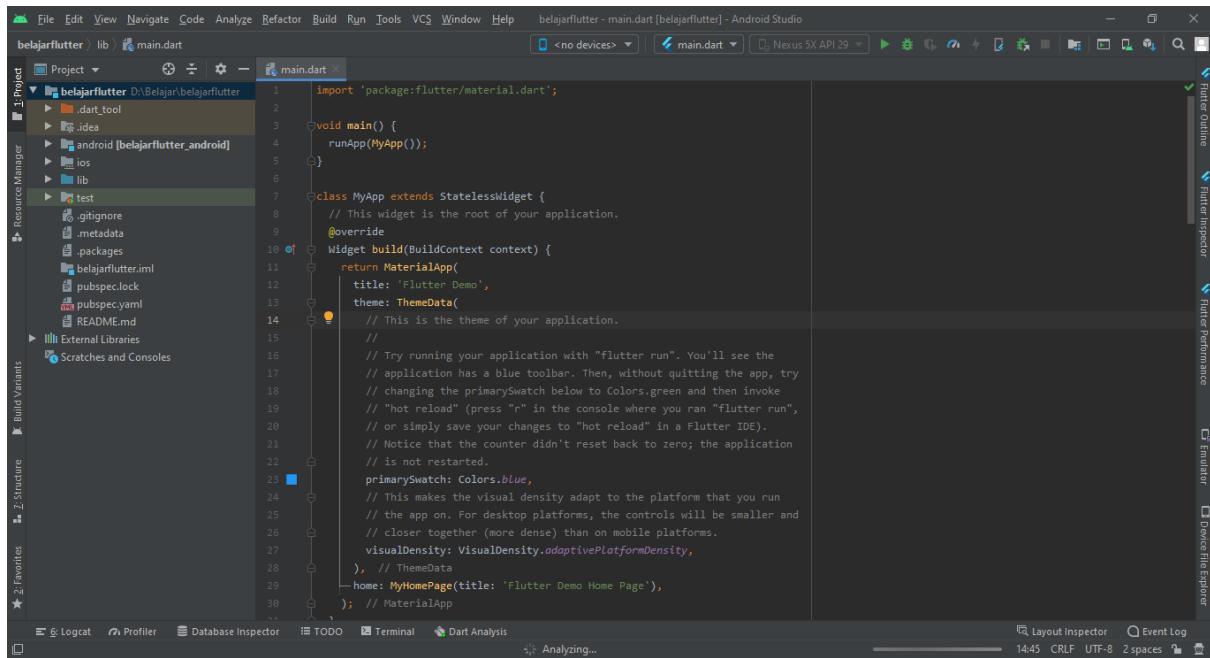
Kemudian klik tombol “Next”



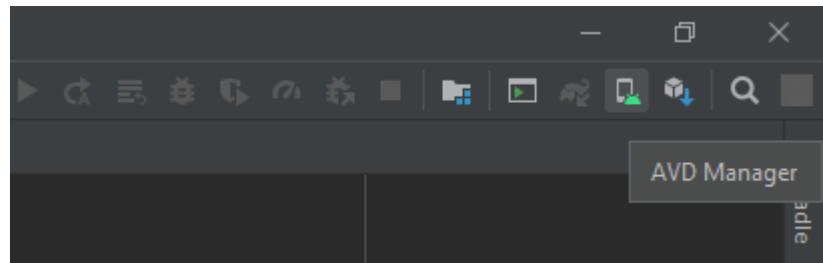
Kemudian klik tombol “Finish”. Pastikan perangkat terhubung ke internet, karena diperlukan untuk mengunduh projek yang dibuat



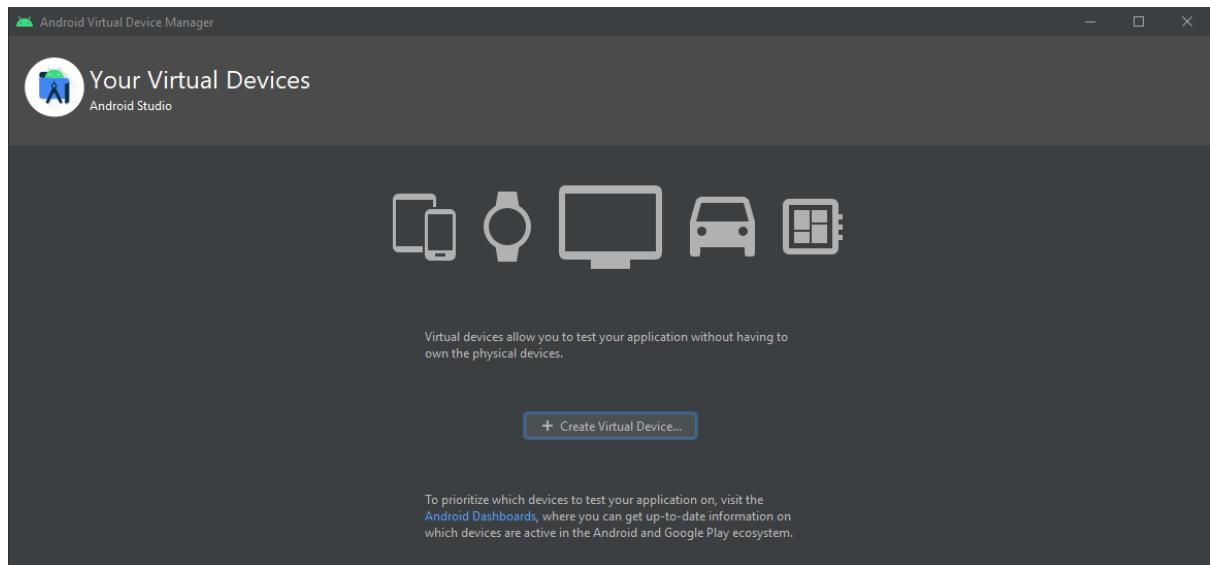
Setelah selesai akan muncul projek yang telah dibuat



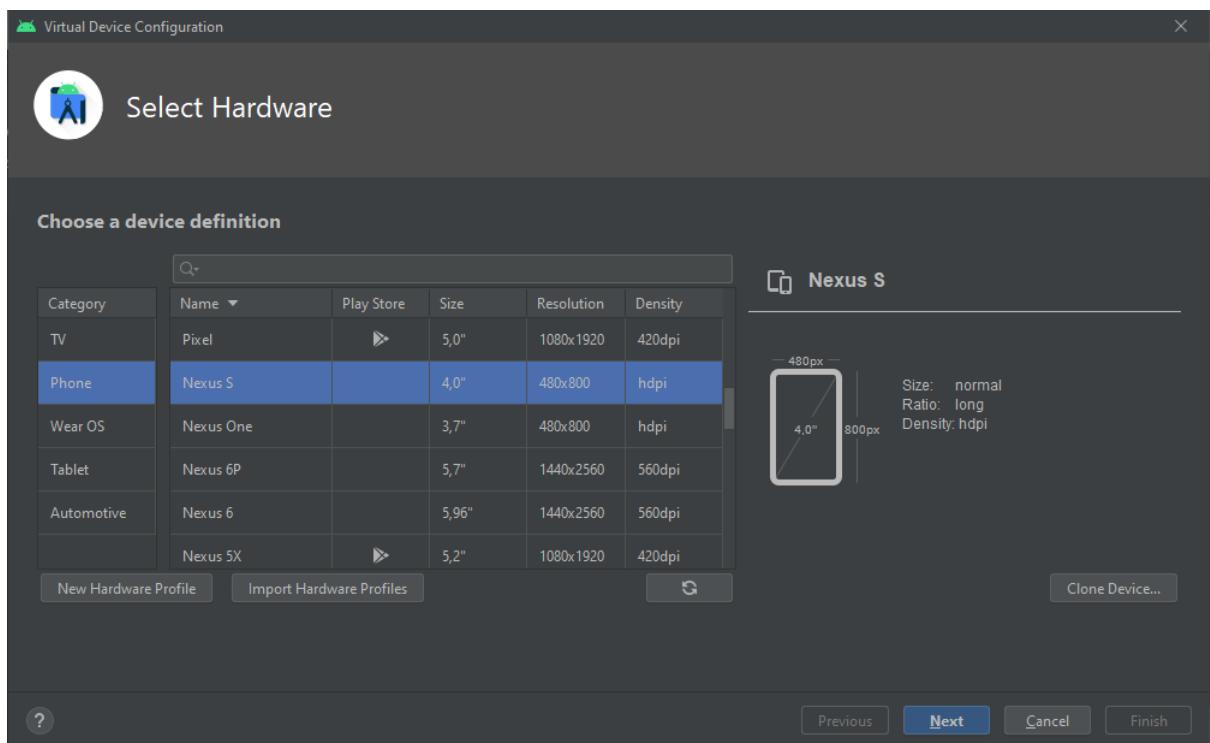
Untuk menjalankan projek, kita memerlukan emulator android. Pertama kita akan membuat emulator android dengan cara klik "AVD Manager" pada pojok kiri atas



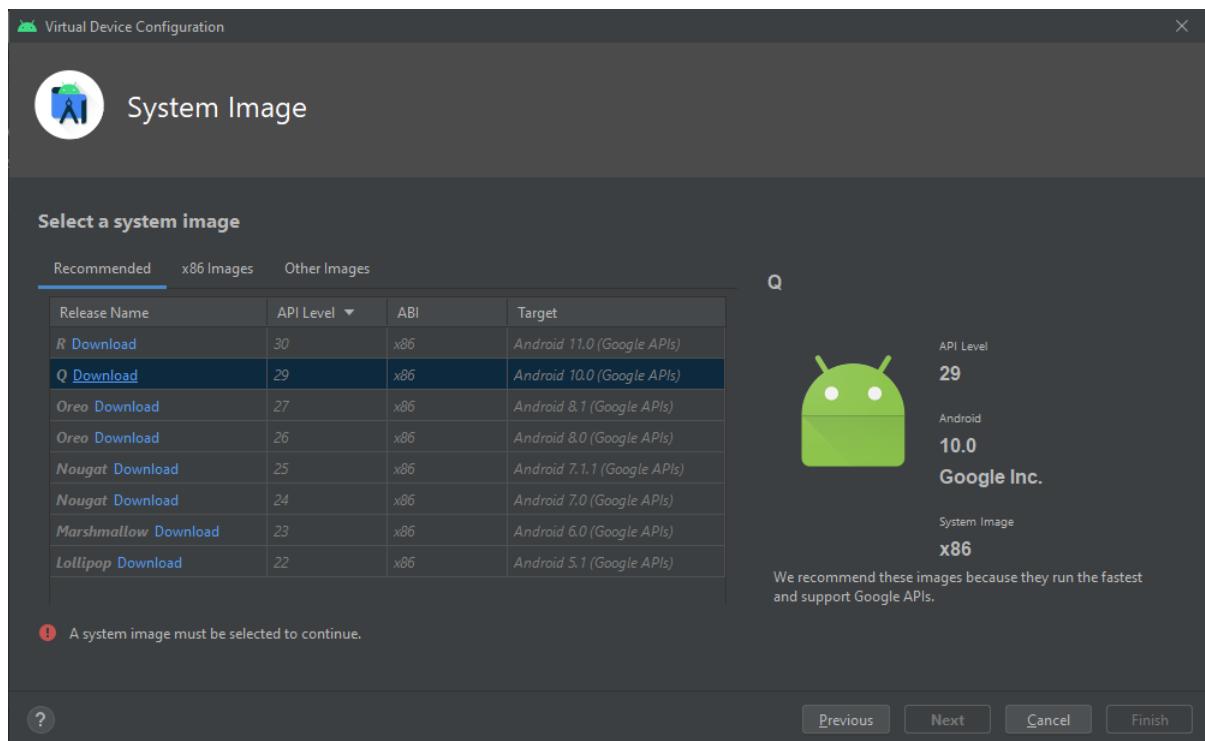
Kemudian klik tombol “Create Virtual Device”



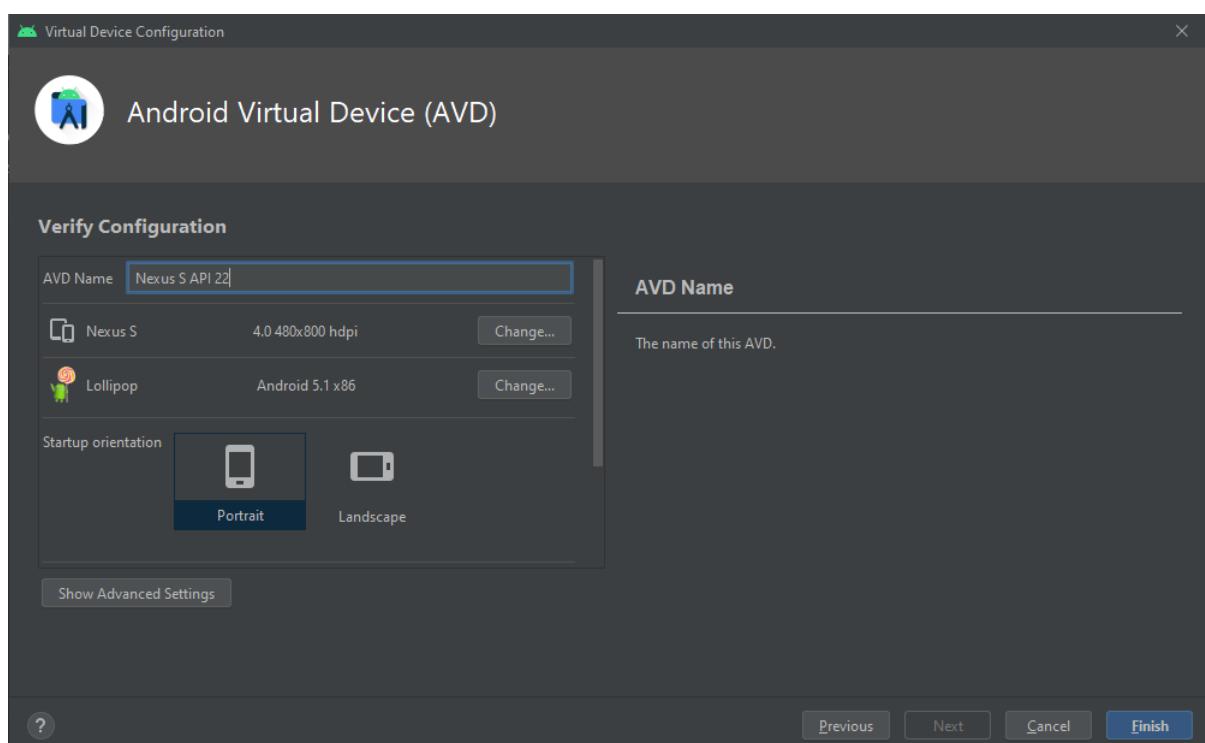
Pilih salah satu device yang dinginkan, misalnya “Nexus S” kemudian klik tombol “Next”



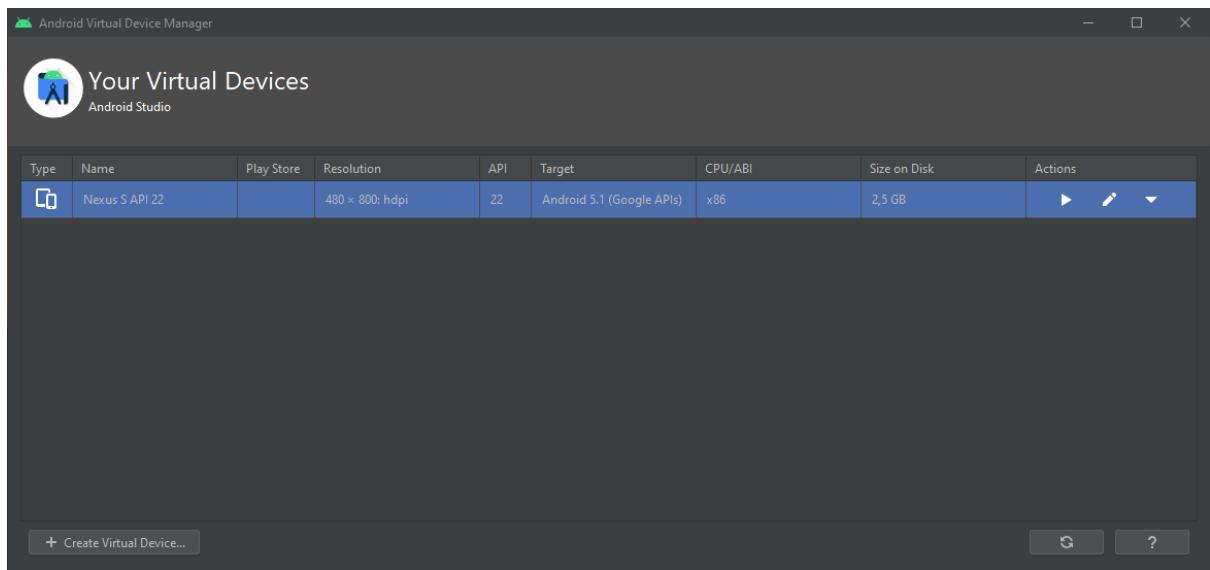
Kemudian kita akan memilih Versi Sistem Operasi Android, dalam hal ini misalnya "Q", jika belum ada maka kita akan diminta untuk mengunduh terlebih dahulu.



Setelah itu klik Finish



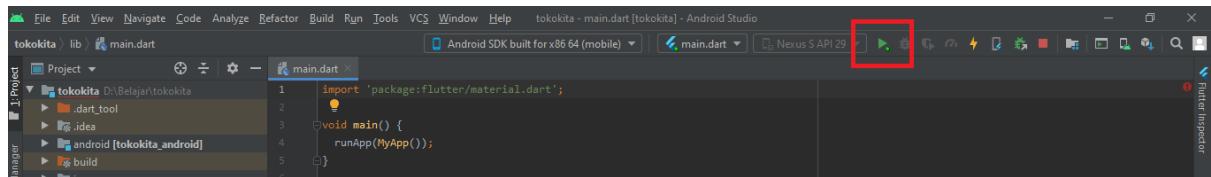
Untuk Menjalankan Emulator, klik logo play



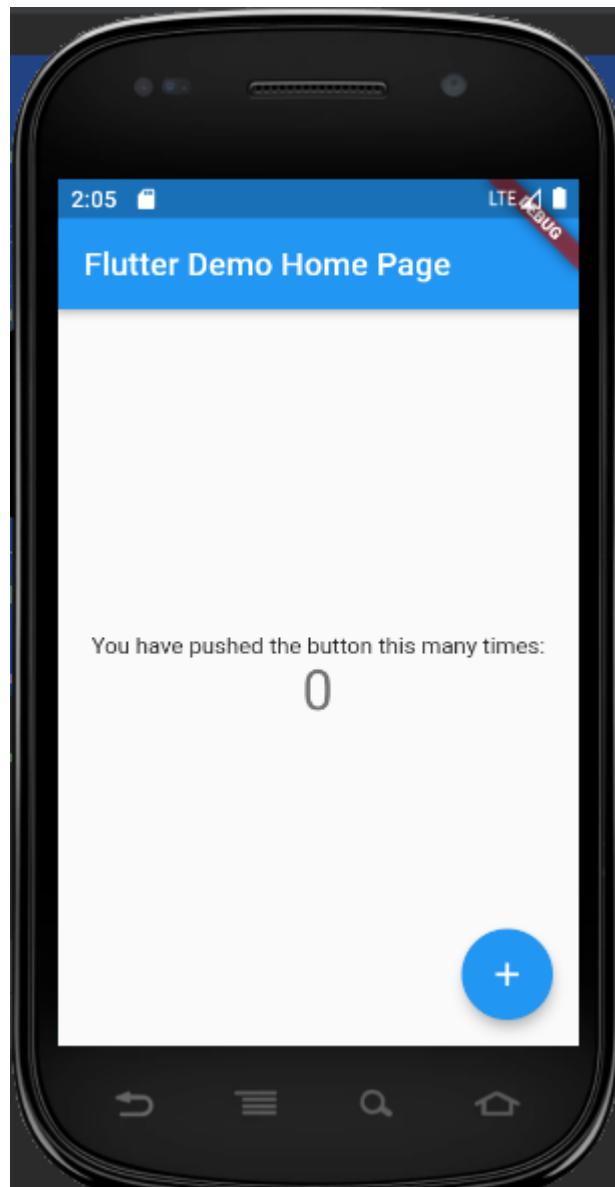
Kemudian akan muncul emulator android seperti berikut:



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



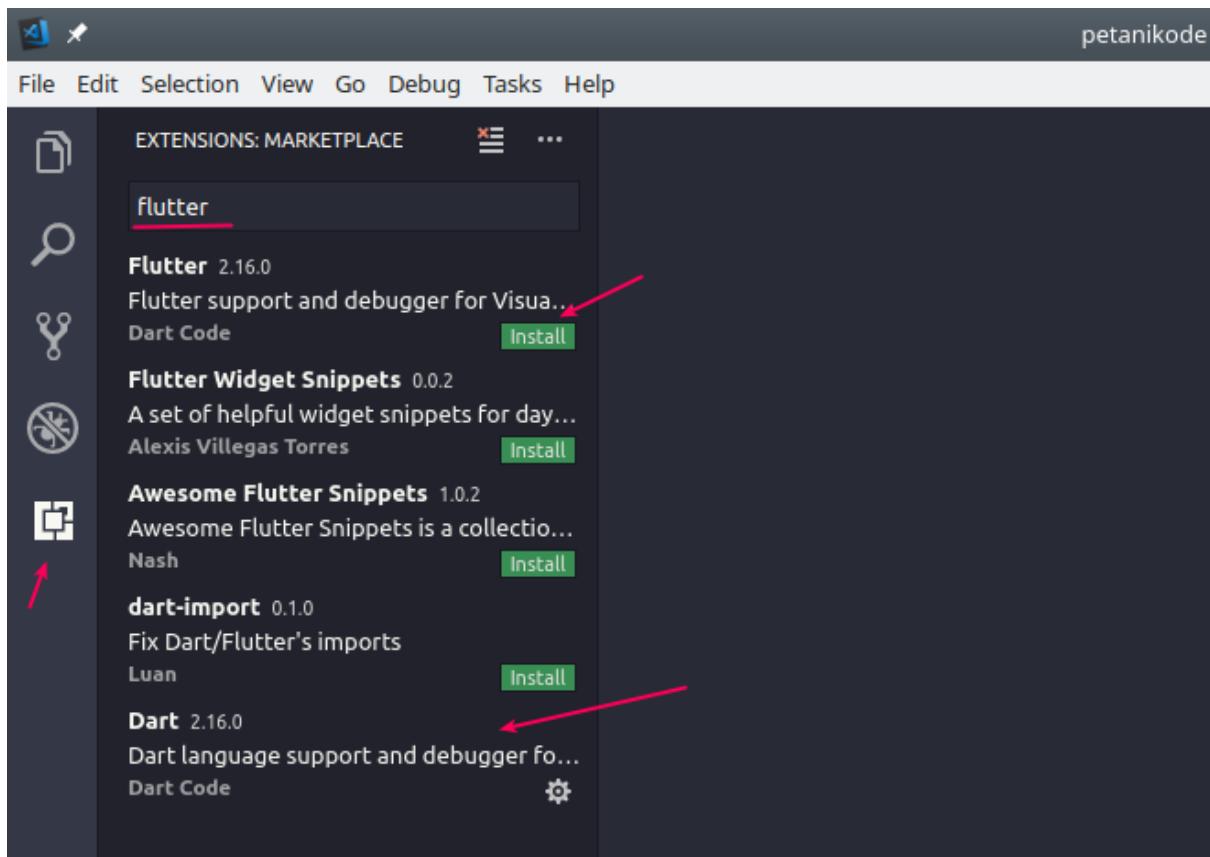
Pada emulator akan tampil hasil projek seperti berikut



Membuat dan menjalankan projek dengan VSCode dan Handphone Android

Install VSCode sebagai alternatif editor

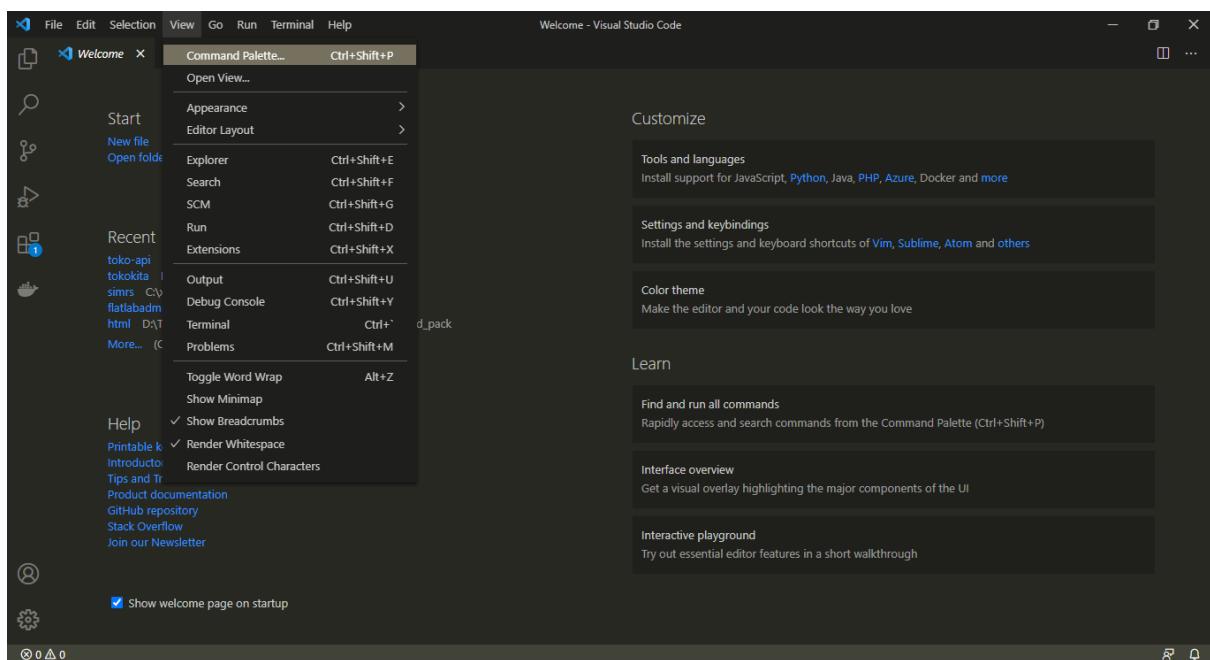
Unduh VSCode pada laman <https://code.visualstudio.com/download> kemudian install. Agar flutter dapat digunakan pada VSCode, perlu diinstall beberapa extension flutter yang dibutuhkan.



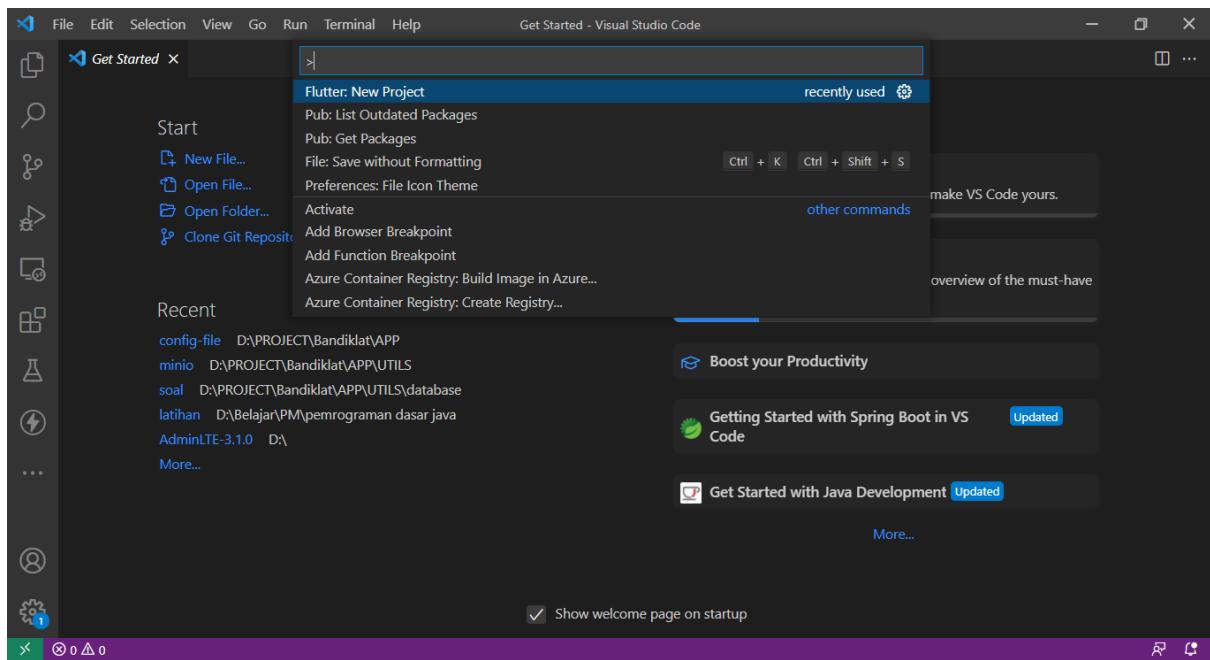
Setelah itu restart/tutup VSCode

Membuat projek flutter dengan VSCode

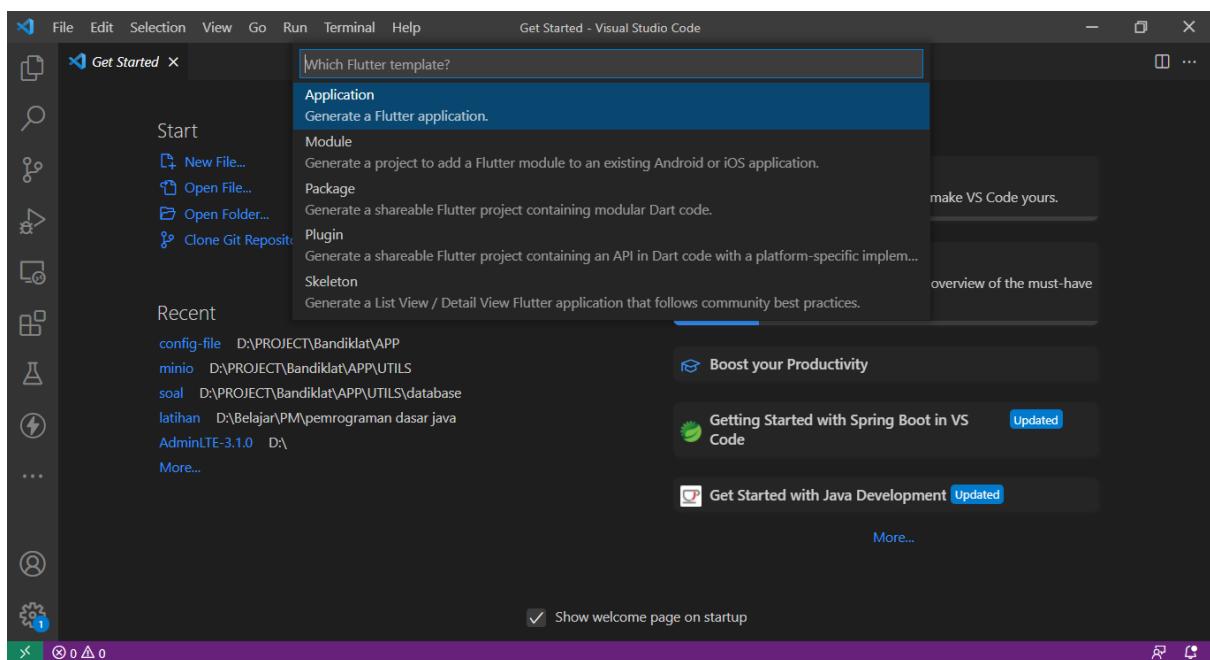
Jalankan VSCode, pada menubar pilih **view -> command Palette...** atau dapat juga dengan shortcut **Ctrl + Shift + P**



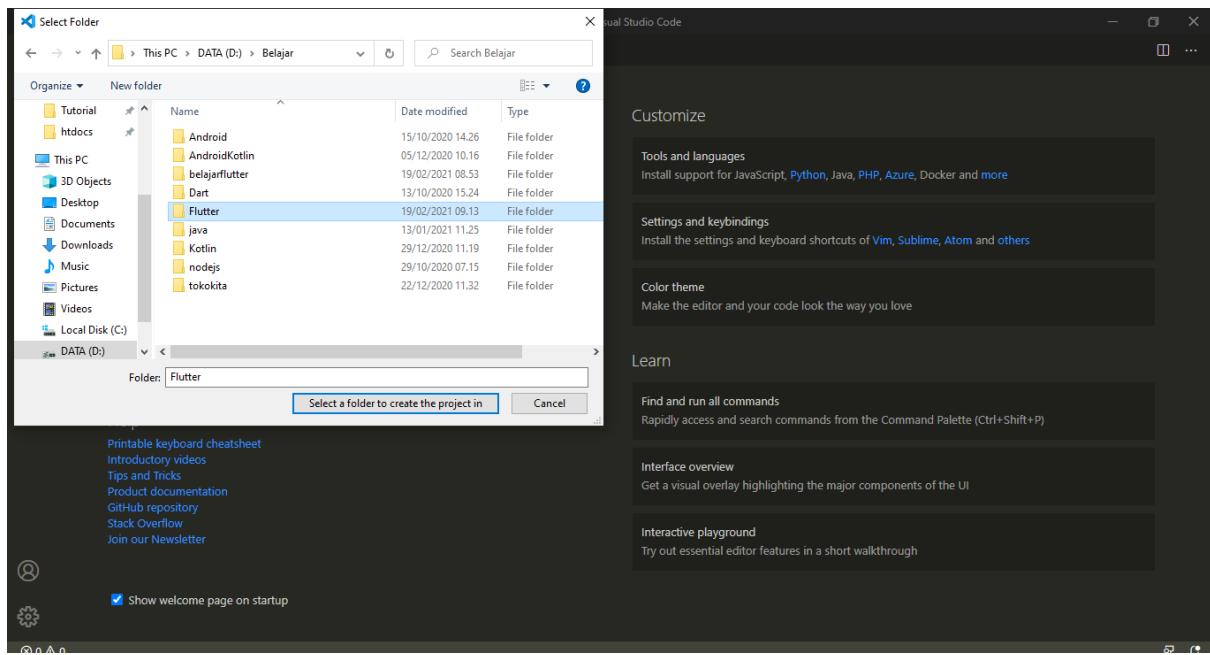
Kemudian ketikkan **flutter** dan pilih **Flutter: New Project**



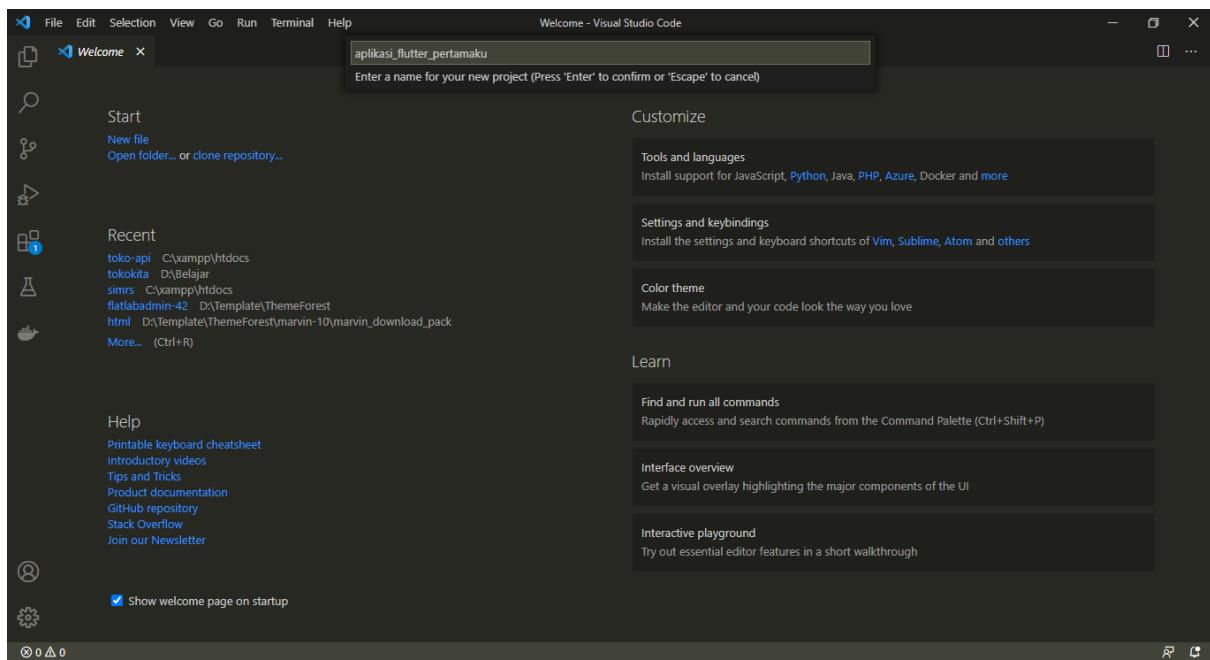
Kemudian pilih Application



Pilih folder tempat projek tersebut



Kemudian tentukan nama projek flutter yang ingin dibuat misalnya
aplikasi_flutter_pertamaku



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

The screenshot shows the Visual Studio Code interface with the following details:

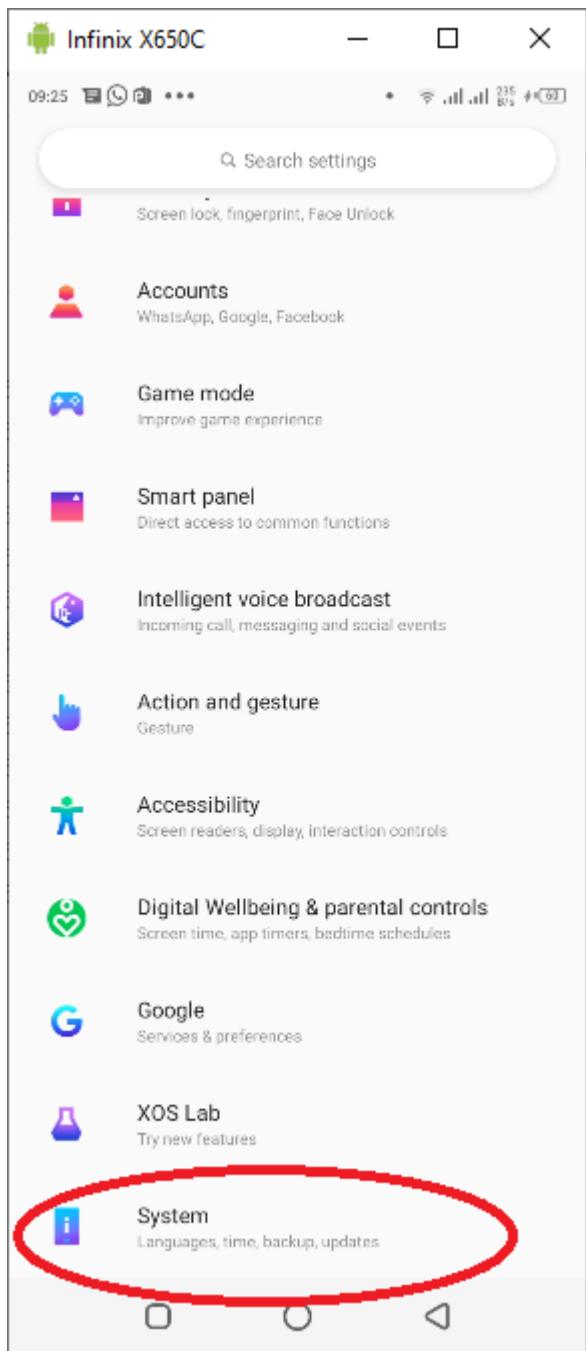
- File Explorer (Left):** Shows the project structure: `APLIKASI_FLUTTER_PERTAMAKU` (containing `.dart_tool`, `.idea`, `android`, `ios`, `lib` with `main.dart`, `test`, `.gitignore`, `.metadata`, `packages`, `aplikasi_flutter_pertamaku.iml`, `pubspec.lock`, `pubspec.yaml`, and `README.md`), `NPM SCRIPTS`, `DEPENDENCIES`, and `SONARLINT RULES`.
- Code Editor (Center):** Displays the `main.dart` file content in Dart syntax.

```
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 Run | Debug
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   // This widget is the root of your application.
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        // This is the theme of your application.
16        //
17        // Try running your application with "flutter run". You'll see the
18        // application has a blue toolbar. Then, without quitting the app, try
19        // changing the primarySwatch below to Colors.green and then invoke
20        // "hot reload" (press "r" in the console where you ran "flutter run",
21        // or simply save your changes to "hot reload" in a Flutter IDE).
22        // Notice that the counter didn't reset back to zero; the application
23        // is not restarted.
24        primarySwatch: Colors.blue,
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      ), // ThemeData
30      home: MyHomePage(title: 'Flutter Demo Home Page'),
31    ); // MaterialApp
32 }
```
- Terminal (Bottom):** Shows status information: Ln 18, Col 76, Spaces: 2, UTF-8, CRLF, Dart, Flutter: 1.22.6, No Device.

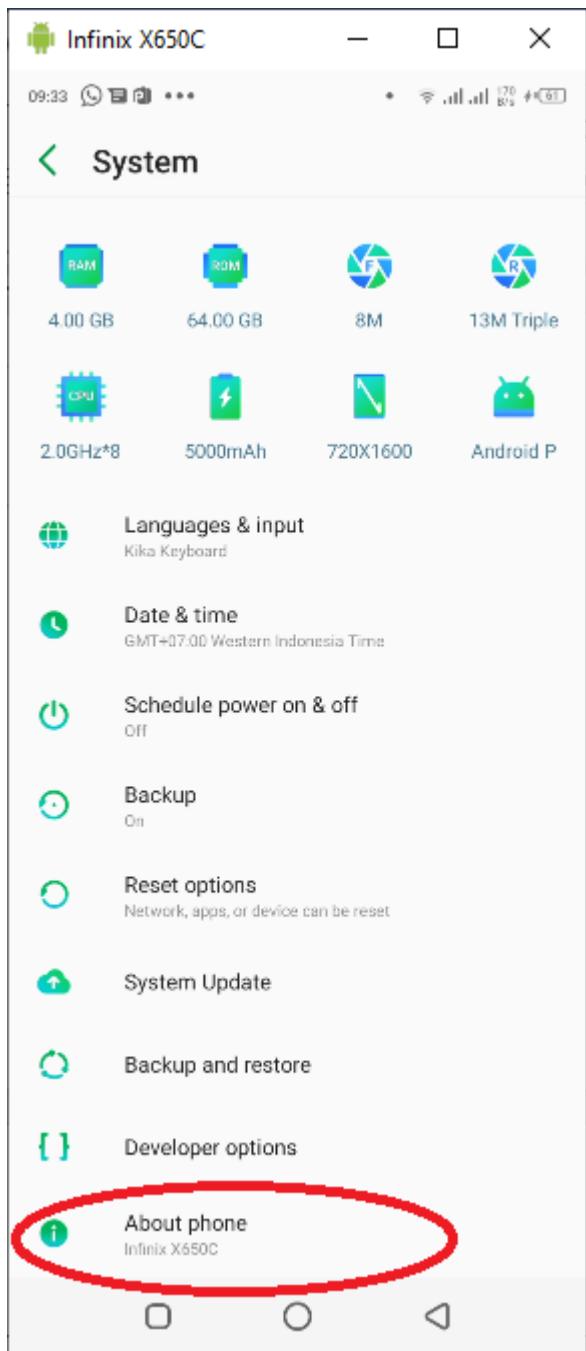
Menjalankan aplikasi dengan Handphone Android

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

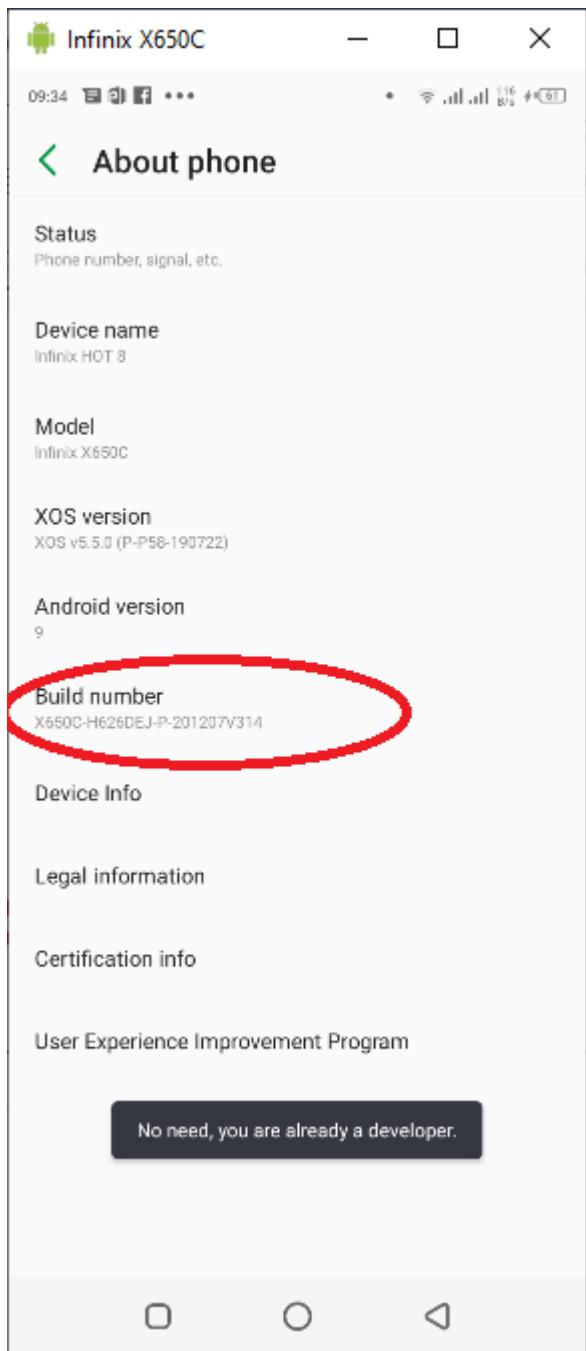
Untuk menggunakan android device secara langsung, pertama aktifkan dulu mode developer dengan cara buka **Setting** kemudian pilih **System** kemudian pilih **About Phone**, untuk masing-masing device mungkin terdapat perbedaan untuk lokasi **About Phone** ada pula yang berada pada **Additional Setting**



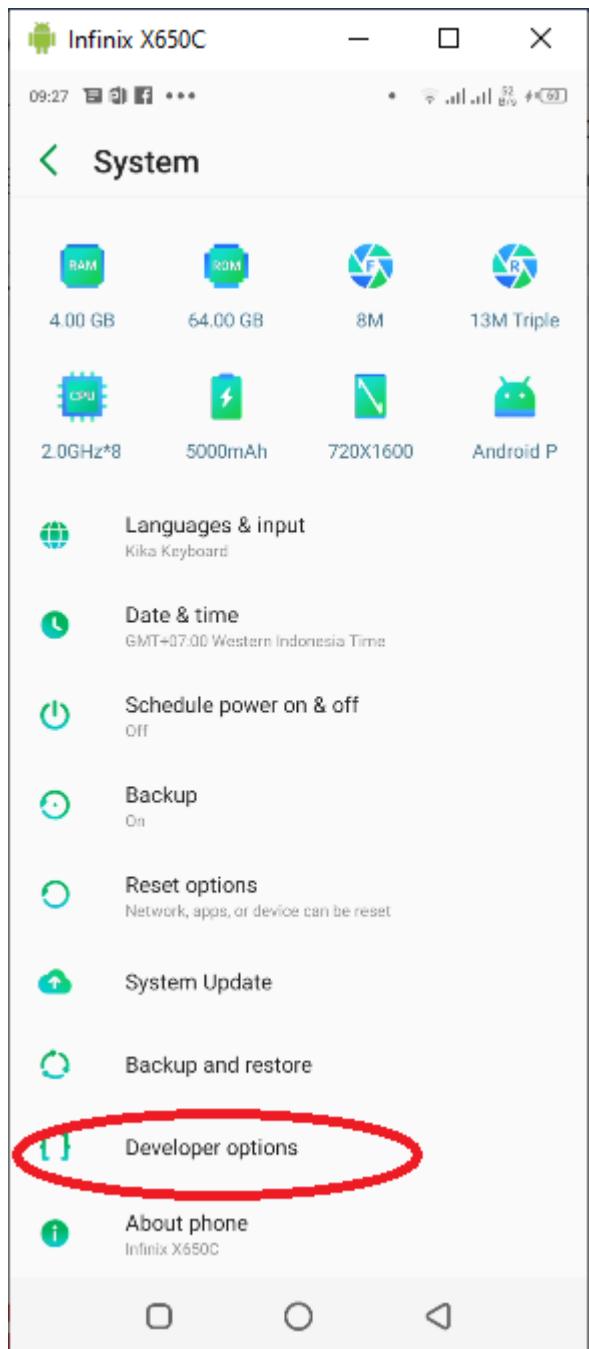
Kemudian pilih **About Phone**



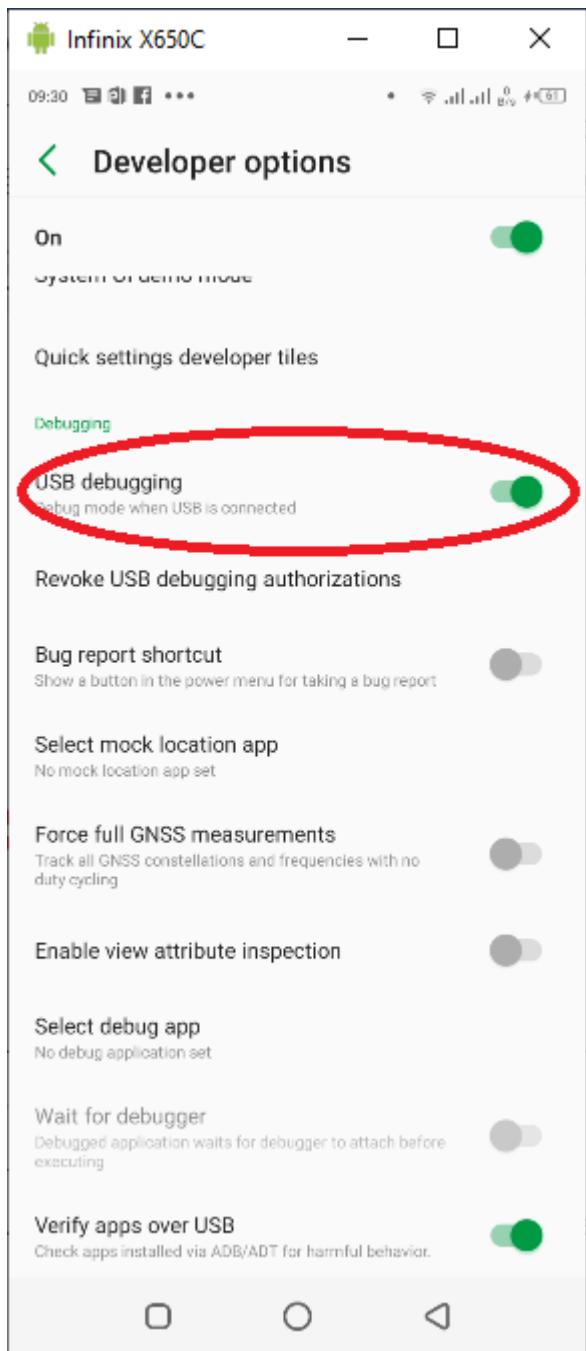
Kemudian ketuk **Build number** beberapa kali, namun ini juga berbeda untuk beberapa versi misalnya untuk Xiaomi dengan mengetuk **MIUI Version** beberapa kali



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
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        // 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

```

Line 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 D:\Belajar\belajarflutter
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
APLIKASI_FLUTTER_PERTAMAKU
main.dart
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 Run | Debug
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   // This widget is the root of your application.
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        // This is the theme of your application.
16        //
17        // Try running your application with "flutter run". You'll see the
18        // application has a blue toolbar. Then, without quitting the app, try
19        // changing the primarySwatch below to Colors.green and then invoke
20        // "hot reload" (press "r" in the console where you ran "flutter run",
21        // or simply save your changes to "hot reload" in a Flutter IDE).
22        // Notice that the counter didn't reset back to zero; the application
23        // is not restarted.
24        primarySwatch: Colors.blue,
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      ), // ThemeData
30      home: MyHomePage(title: 'Flutter Demo Home Page'),
31    ); // MaterialApp
32 }
```

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
APLIKASI_FLUTTER_PERTAMAKU
main.dart
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 Run | Debug
4 void main() {
5   runApp(MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   // This widget is the root of your application.
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        // This is the theme of your application.
16        //
17        // Try running your application with "flutter run". You'll see the
18        // application has a blue toolbar. Then, without quitting the app, try
19        // changing the primarySwatch below to Colors.green and then invoke
20        // "hot reload" (press "r" in the console where you ran "flutter run",
21        // or simply save your changes to "hot reload" in a Flutter IDE).
22        // Notice that the counter didn't reset back to zero; the application
23        // is not restarted.
24        primarySwatch: Colors.blue,
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      ), // ThemeData
30      home: MyHomePage(title: 'Flutter Demo Home Page'),
31    ); // MaterialApp
32 }
```

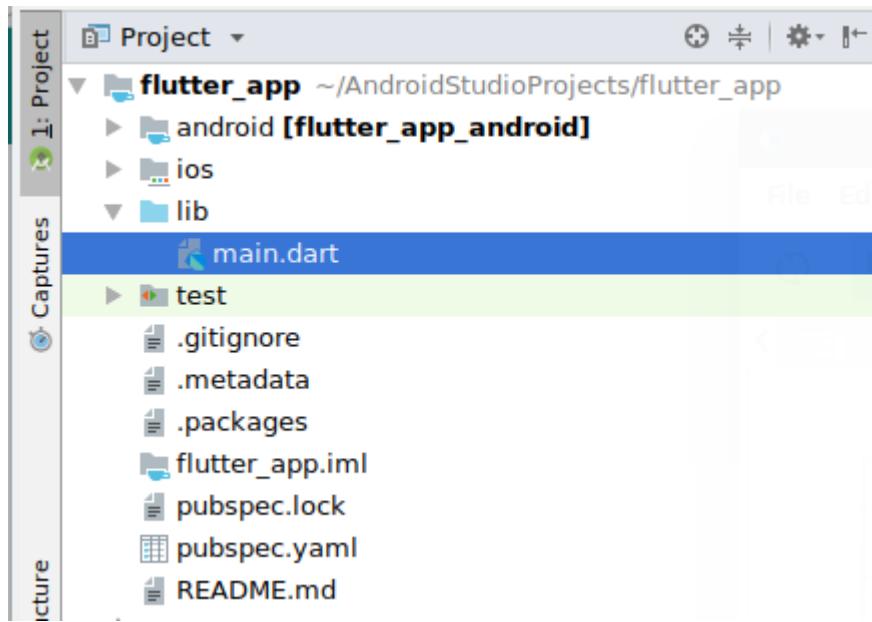
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE Filter (e.g. text, | exclude) lib\main.dart:1

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://127.0.0.1:59654/EeZyoXHx9j4=/ws

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

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 libraray yang dibutuhkan;
- ② **README.md** merupakan file markdown yang berisi penjelasan tentang source code.

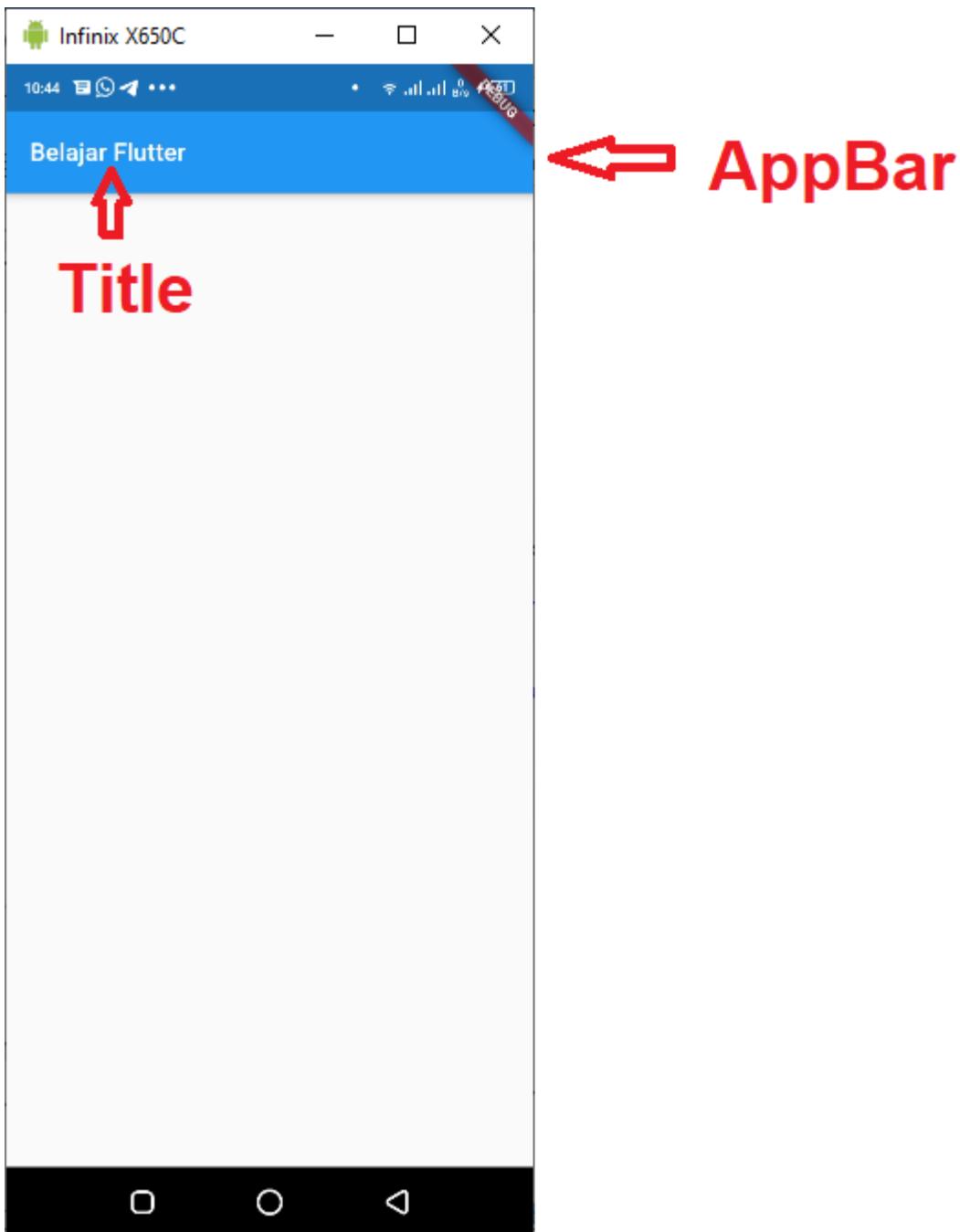
Membuat Hello World

Buka projek **aplikasi_flutter_pertamaku** menggunakan VSCode agar lebih ringan. 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: "Aplikasi Flutter Pertama",  
14.      home: Scaffold(  
15.        appBar: AppBar(  
16.          title: const Text('Belajar Flutter'),  
17.        ),  
18.      ),  
19.    );  
20.  }  
21. }
```

Ketika dijalankan akan menghasilkan

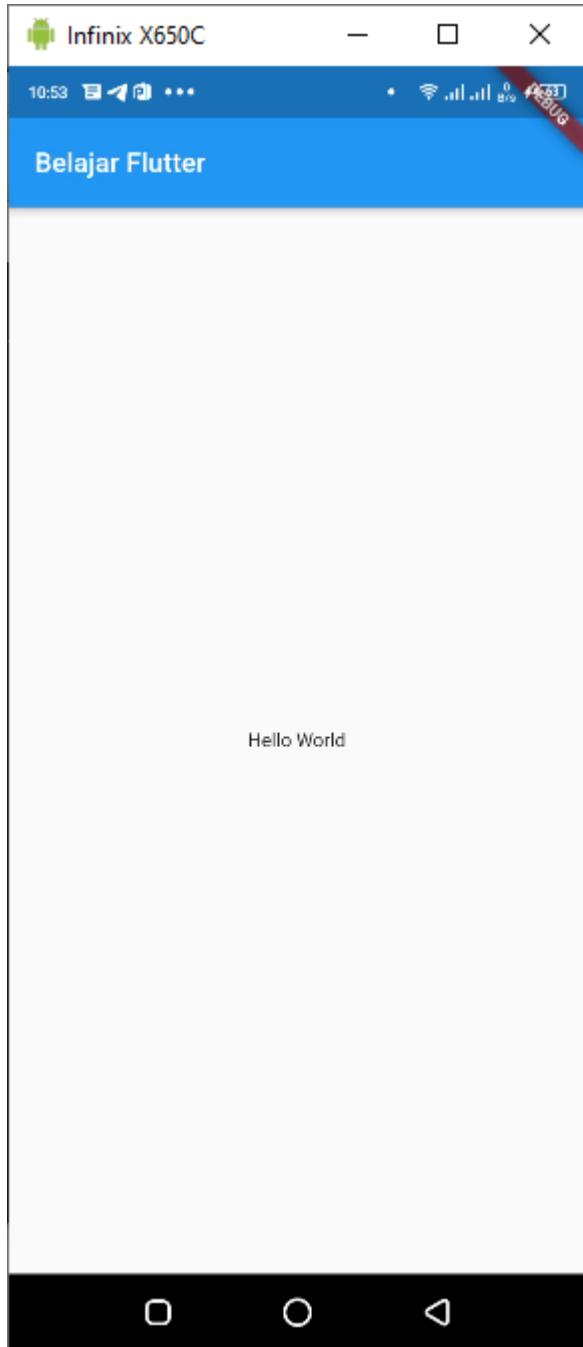


Kemudian untuk menambahkan tampilan dibagian dalam (body), pada fungsi Scaffold terdapat parameter **body**. Silahkan modifikasi **main.dart** menjadi seperti berikut

```
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: "Aplikasi Flutter Pertama",
```

```
14.     home: Scaffold(
15.         appBar: AppBar(
16.             title: Text('Belajar Flutter'),
17.         ),
18.         body: const Center(
19.             child: Text("Hello World"),
20.         ),
21.     ),
22. );
23. }
24. }
```

Sehingga akan menghasilkan



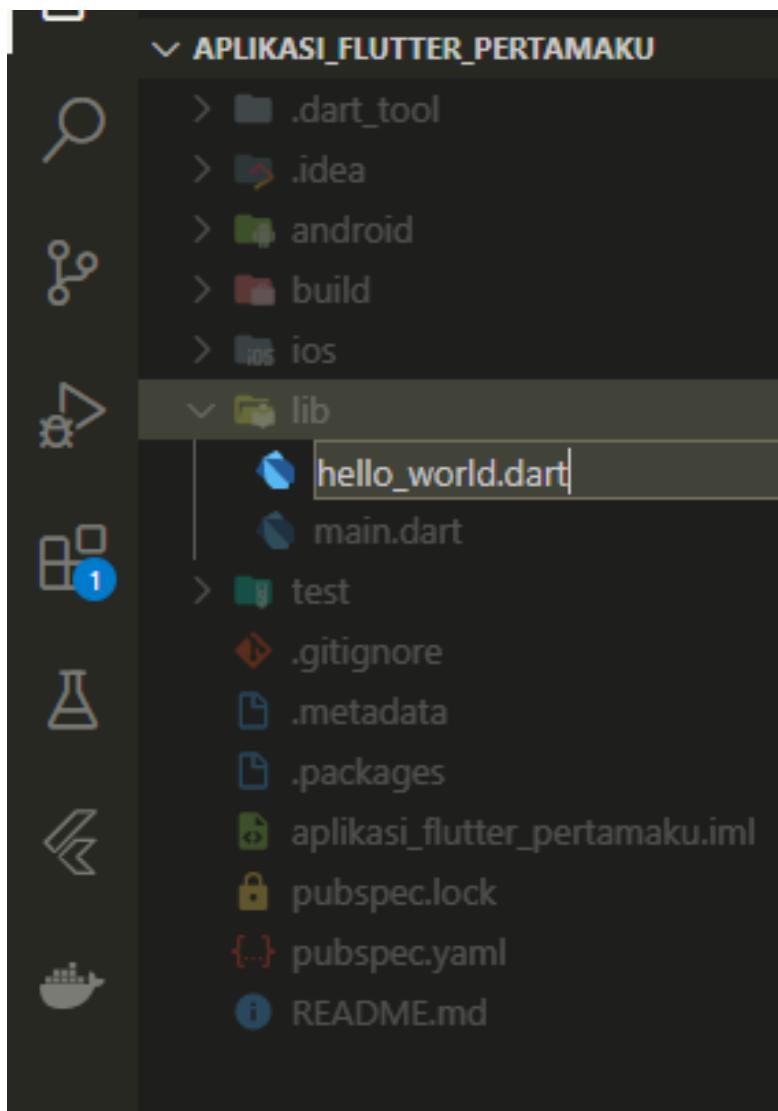
Pada aplikasi di atas, kita membuat **StatelessWidget** yang berisi widget **MaterialApp()**. Kemudian di dalam **MaterialApp()** berisi widget lagi: **Scaffold**, **AppBar**, **Center**, dan **Text**. Ini adalah widget dasar.

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**



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.   @override
5.   Widget build(BuildContext context) {
6.     return Scaffold(
7.       appBar: AppBar(
8.         title: const Text('Belajar Flutter'),
9.       ),
10.      body: const Center(
11.        child: Text('Hello World'),
12.      ),
13.    );
14.  }
15. }
```

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

```
1. import 'package:aplikasi_flutter_pertamaku/hello_world.dart';
2. import 'package:flutter/material.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: "Aplikasi Flutter Pertama",
15.     home: HelloWorld(),
16.   );
17. }
18. }
```

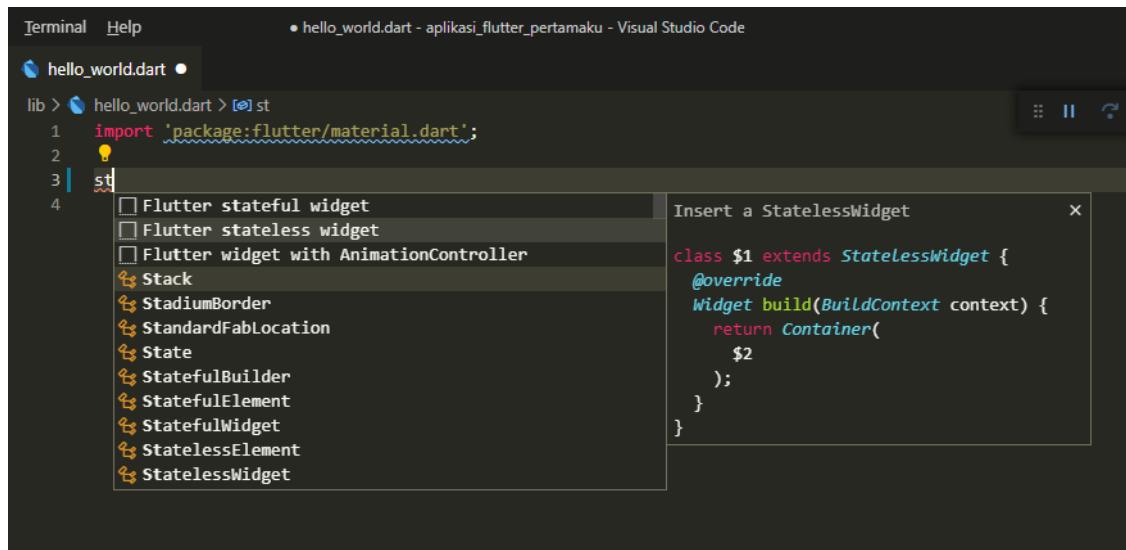
Pada bagian **home**, kita memanggil class **HelloWorld** yang telah kita buat sebelumnya pada file **hello_world.dart**

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



Membuat Widget Column

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

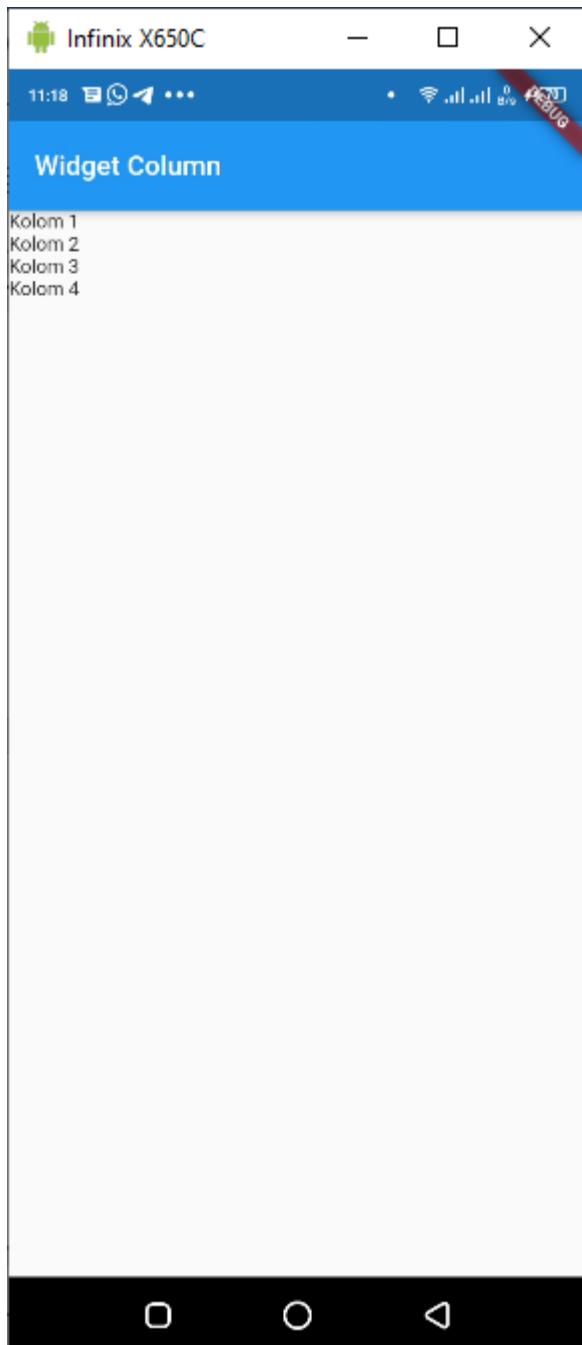
```
1. import 'package:flutter/material.dart';  
2.  
3. class ColumnWidget extends StatelessWidget {  
4.   const ColumnWidget({Key? key}) : super(key: key);  
5.  
6.   @override  
7.   Widget build(BuildContext context) {  
8.     return Scaffold(  
9.       appBar: AppBar(  
10.         title: const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

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

```
1. import 'package:aplikasi_flutter_pertamaku/column_widget.dart';  
2. import 'package:flutter/material.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: "Aplikasi Flutter Pertama",  
15.     home: ColumnWidget(),  
16.   );  
17. }  
18. }
```

Dan hasilnya akan menjadi seperti berikut



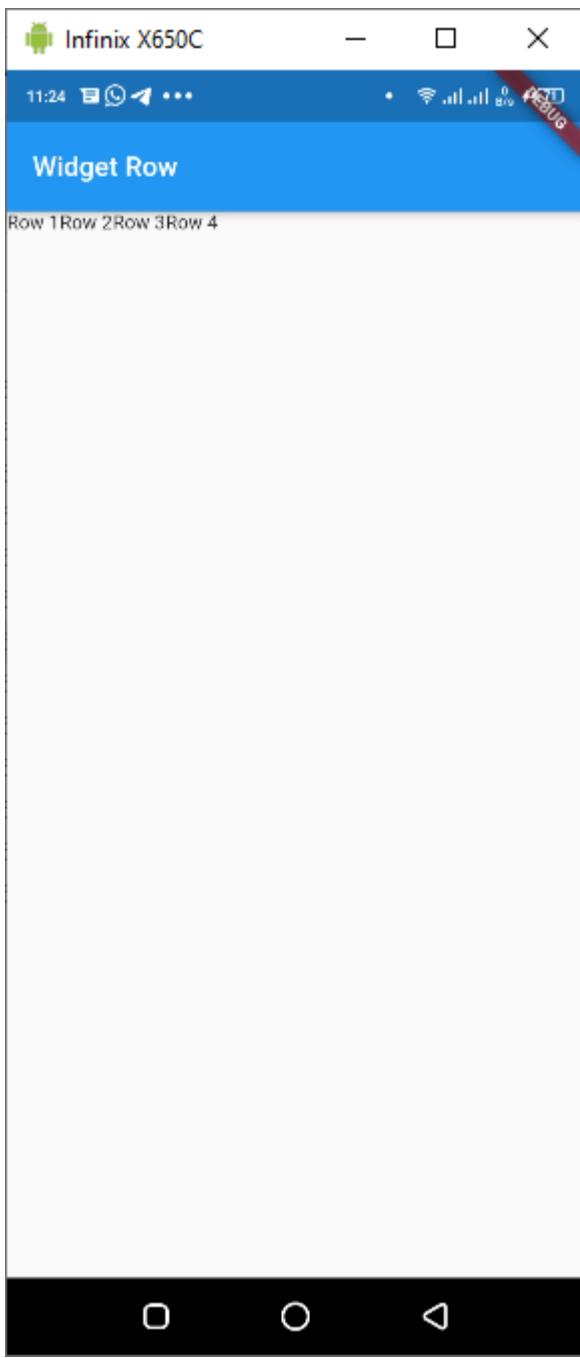
Column biasanya digunakan untuk membuat Form

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**, dan hasilnya akan menjadi



StatelessWidget dan StatefullWidget

StatelessWidget adalah class widget yang propertinya *immutable*, artinya nilainya tidak bisa diubah, sedangkan **StatefullWidget** 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. }
```

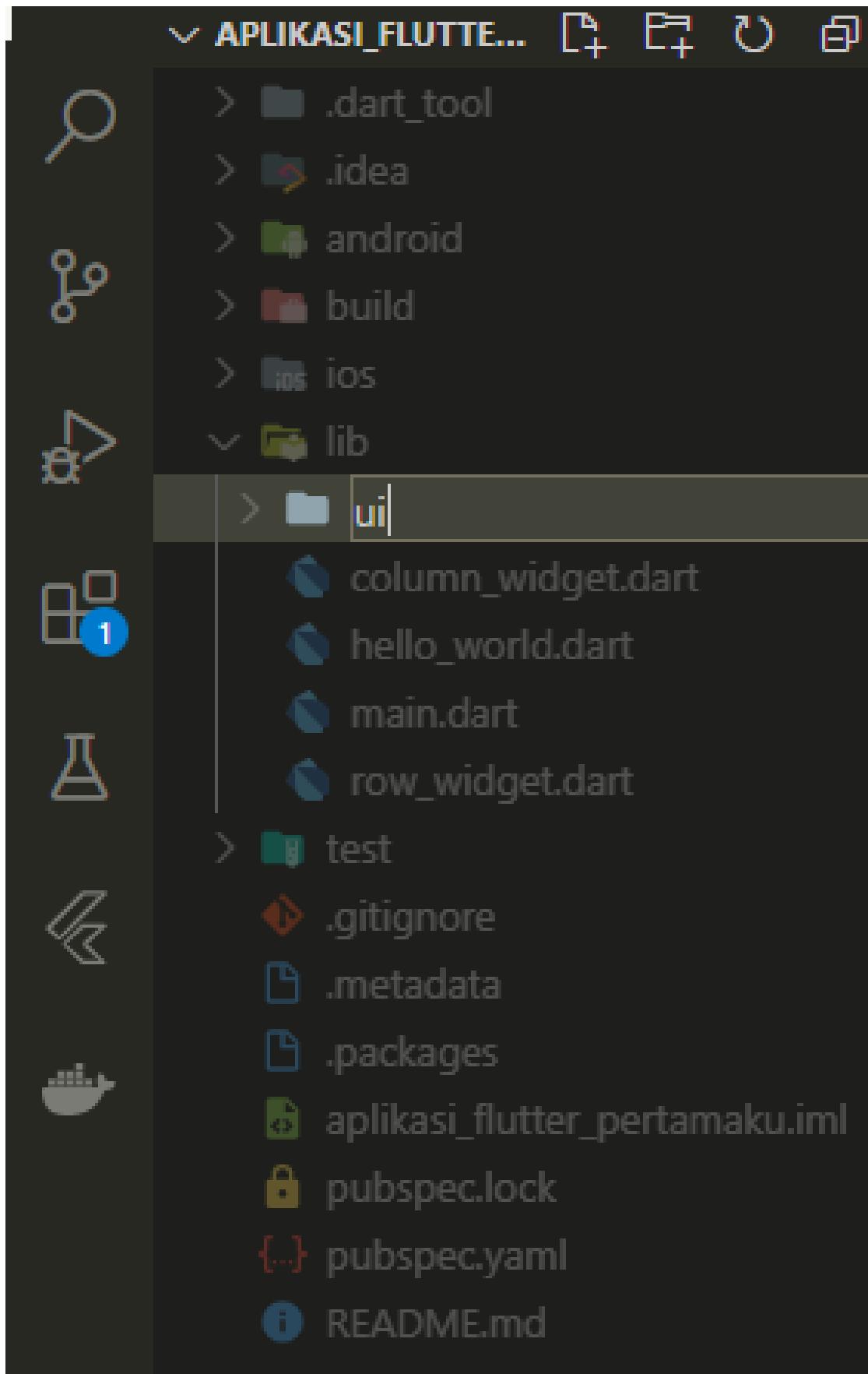
```
9.    }
10. }
```

Contoh StatefullWidget

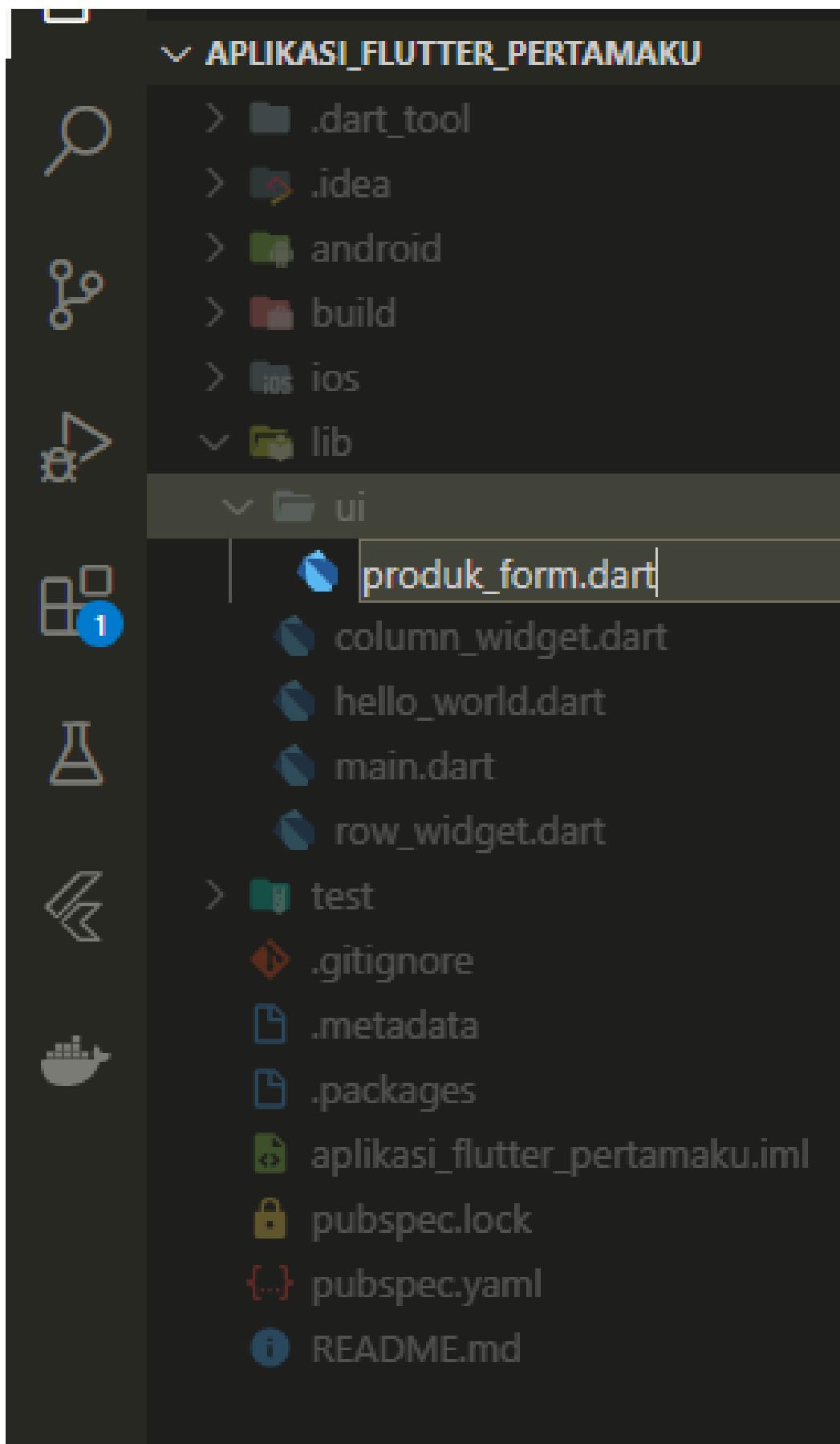
```
1. class HelloWorld extends StatefulWidget {
2.   const HelloWorld({ Key? key }) : super(key: key);
3.
4.   @override
5.   _HelloWorldState createState() => _HelloWorldState();
6. }
7.
8. class _HelloWorldState extends State<HelloWorld> {
9.   @override
10.  Widget build(BuildContext context) {
11.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

Membuat Form

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 produk_form.dart



Kemudian Ketikkan kode berikut

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

Ubah pada **main.dart** dengan memanggil class **ProdukForm**, sehingga hasilnya akan menjadi



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 **produk_form.dart** terdapat widget seperti **TextField** dan **Button**, pada widget tersebut akan kita pisahkan kedalam method tersendiri didalam class, sehingga menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2.
3. class ProdukForm extends StatefulWidget {
4.   const ProdukForm({Key? key}) : super(key: key);
5.
6.   @override
7.   _ProdukFormState createState() => _ProdukFormState();
```

```

8. }
9.
10. class _ProdukFormState extends State<ProdukForm> {
11.   @override
12.   Widget build(BuildContext context) {
13.     return Scaffold(
14.       appBar: AppBar(
15.         title: const Text('Form Produk'),
16.       ),
17.       body: SingleChildScrollView(
18.         child: Column(
19.           children: [
20.             _textboxKodeProduk(),
21.             _textboxNamaProduk(),
22.             _textboxHargaProduk(),
23.             _tombolSimpan()
24.           ],
25.         ),
26.       );
27.     );
28.   }
29.
30.   _textboxKodeProduk() {
31.     return TextField(decoration: const InputDecoration(labelText: "Kode Produk"));
32.   }
33.
34.   _textboxNamaProduk() {
35.     return TextField(decoration: const InputDecoration(labelText: "Nama Produk"));
36.   }
37.
38.   _textboxHargaProduk() {
39.     return TextField(decoration: const InputDecoration(labelText: "Harga"));
40.   }
41.
42.   _tombolSimpan() {
43.     return ElevatedButton(onPressed: () {}, child: const Text('Simpan'));
44.   }
45. }

```

Membuat Detail Produk

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

```

1. import 'package:flutter/material.dart';
2.
3. class ProdukDetail extends StatefulWidget {
4.   final String? kodeProduk;
5.   final String? namaProduk;
6.   final int? harga;
7.
8.   const ProdukDetail({Key? key, this.kodeProduk, this.namaProduk, this.harga})
9.     : super(key: key);
10.
11.   @override
12.   _ProdukDetailState createState() => _ProdukDetailState();
13. }
14.
15. class _ProdukDetailState extends State<ProdukDetail> {
16.   @override
17.   Widget build(BuildContext context) {
18.     return Scaffold(

```

```

19.     appBar: AppBar(
20.         title: const Text('Detail Produk'),
21.     ),
22.     body: Column(
23.         children: [
24.             Text("Kode Produk : " + widget.kodeProduk.toString()),
25.             Text("Nama Produk : ${widget.namaProduk}"),
26.             Text("Harga : ${widget.harga}")
27.         ],
28.     ),
29. );
30. }
31. 
```

Membuat fungsi tombol simpan dan menampilkan data pada Detail Produk

Buka kembali file `produk_form.dart` tambahkan attribute

```

final _kodeProdukTextboxController = TextEditingController();
final _namaProdukTextboxController = TextEditingController();
final _hargaProdukTextboxController = TextEditingController(); 
```

pada Class `ProdukFormState` sehingga menjadi

```

1. import 'package:aplikasi_flutter_pertamaku/ui/produk_detail.dart';
2. import 'package:flutter/material.dart';
3.
4. class ProdukForm extends StatefulWidget {
5.     const ProdukForm({Key? key}) : super(key: key);
6.
7.     @override
8.     _ProdukFormState createState() => _ProdukFormState();
9. }
10.
11. class _ProdukFormState extends State<ProdukForm> {
12.     final _kodeProdukTextboxController = TextEditingController();
13.     final _namaProdukTextboxController = TextEditingController();
14.     final _hargaProdukTextboxController = TextEditingController();
15.
16.     @override
17.     Widget build(BuildContext context) {
18.         return Scaffold(
19.             appBar: AppBar(
20.                 title: const Text('Form Produk'),
21.             ),
22.             body: SingleChildScrollView(
23.                 child: Column(
24.                     children: [
25.                         _textboxKodeProduk(),
26.                         _textboxNamaProduk(),
27.                         _textboxHargaProduk(),
28.                         _tombolSimpan()
29.                     ],
30.                 ),
31.             ),
32.         );
33.     }
34.
35.     _textboxKodeProduk() {
36.         return TextField(
37.             decoration: const InputDecoration(labelText: "Kode Produk"));
38.     }
39.
40.     _textboxNamaProduk() { 
```

```

41.     return TextField(
42.         decoration: const InputDecoration(labelText: "Nama Produk"));
43.     }
44.
45.     _textboxHargaProduk() {
46.         return TextField(decoration: const InputDecoration(labelText: "Harga"));
47.     }
48.
49.     _tombolSimpan() {
50.         return ElevatedButton(onPressed: () {}, child: const Text('Simpan'));
51.     }
52. }
```

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

Pada fungsi `_textboxKodeProduk()` menjadi

```

_textboxKodeProduk() {
    return TextField(
        decoration: const InputDecoration(labelText: "Kode Produk"),
        controller: _kodeProdukTextboxController,
    );
}
```

Pada fungsi `_textboxNamaProduk()` menjadi

```

_textboxNamaProduk() {
    return TextField(
        decoration: const InputDecoration(labelText: "Kode Produk"),
        controller: _namaProdukTextboxController,
    );
}
```

Pada fungsi `_textboxhargaProduk()` menjadi

```

_textboxHargaProduk() {
    return TextField(
        decoration: const InputDecoration(labelText: "Kode Produk"),
        controller: _hargaProdukTextboxController,
    );
}
```

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

```

_tombolSimpan() {
    return ElevatedButton(
        onPressed: () {
            String kodeProduk = _kodeProdukTextboxController.text;
            String namaProduk = _namaProdukTextboxController.text;
            int harga = int.parse(
                _hargaProdukTextboxController.text); //parsing dari String ke int
            // pindah ke halaman Produk Detail dan mengirim data
            Navigator.of(context).push(MaterialPageRoute(
                builder: (context) => ProdukDetail(
                    kodeProduk: kodeProduk,
```

```

        namaProduk: namaProduk,
        harga: harga,
    )));
},
child: const Text('Simpan')));
}

```

Sehingga keseluruhan kode menjadi

```

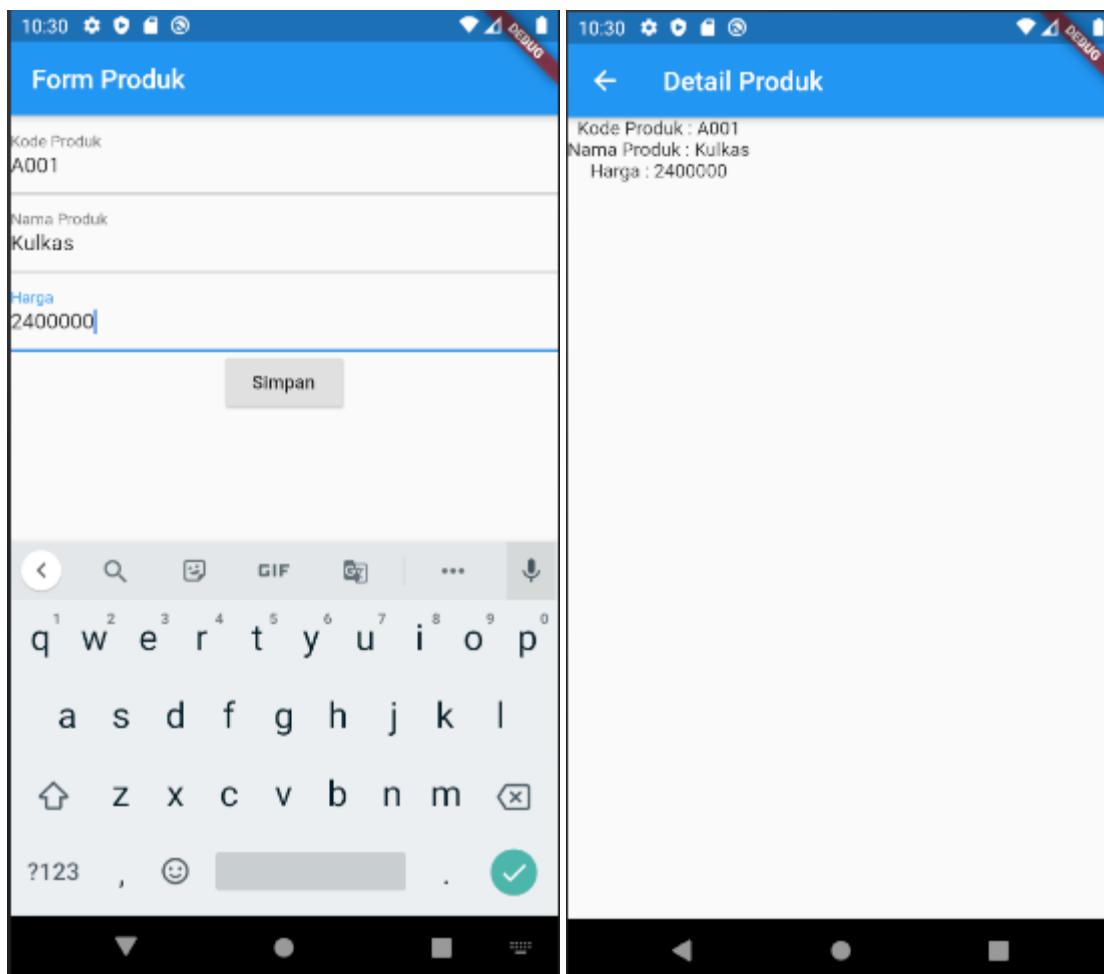
1. import 'package:aplikasi_flutter_pertamaku/ui/produk_detail.dart';
2. import 'package:flutter/material.dart';
3.
4. class ProdukForm extends StatefulWidget {
5.   const ProdukForm({Key? key}) : super(key: key);
6.
7.   @override
8.   _ProdukFormState createState() => _ProdukFormState();
9. }
10.
11. class _ProdukFormState extends State<ProdukForm> {
12.   final _kodeProdukTextboxController = TextEditingController();
13.   final _namaProdukTextboxController = TextEditingController();
14.   final _hargaProdukTextboxController = TextEditingController();
15.
16.   @override
17.   Widget build(BuildContext context) {
18.     return Scaffold(
19.       appBar: AppBar(
20.         title: const Text('Form Produk'),
21.       ),
22.       body: SingleChildScrollView(
23.         child: Column(
24.           children: [
25.             _textboxKodeProduk(),
26.             _textboxNamaProduk(),
27.             _textboxHargaProduk(),
28.             _tombolSimpan()
29.           ],
30.         ),
31.       ),
32.     );
33.   }
34.
35.   _textboxKodeProduk() {
36.     return TextField(
37.       decoration: const InputDecoration(labelText: "Kode Produk"),
38.       controller: _kodeProdukTextboxController,
39.     );
40.   }
41.
42.   _textboxNamaProduk() {
43.     return TextField(
44.       decoration: const InputDecoration(labelText: "Nama Produk"),
45.       controller: _namaProdukTextboxController,
46.     );
47.   }
48.
49.   _textboxHargaProduk() {
50.     return TextField(
51.       decoration: const InputDecoration(labelText: "Harga"),
52.       controller: _hargaProdukTextboxController,
53.     );
54.   }
55.

```

```

56.     _tombolSimpan() {
57.       return ElevatedButton(
58.         onPressed: () {
59.           String kodeProduk = _kodeProdukTextboxController.text;
60.           String namaProduk = _namaProdukTextboxController.text;
61.           int harga = int.parse(
62.             _hargaProdukTextboxController.text); //parsing dari String ke int
63.           // pindah ke halaman Produk Detail dan mengirim data
64.           Navigator.of(context).push(MaterialPageRoute(
65.             builder: (context) => ProdukDetail(
66.               kodeProduk: kodeProduk,
67.               namaProduk: namaProduk,
68.               harga: harga,
69.             )));
70.         },
71.         child: const Text('Simpan'));
72.     }
73.   }

```



Membuat ListView Produk

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

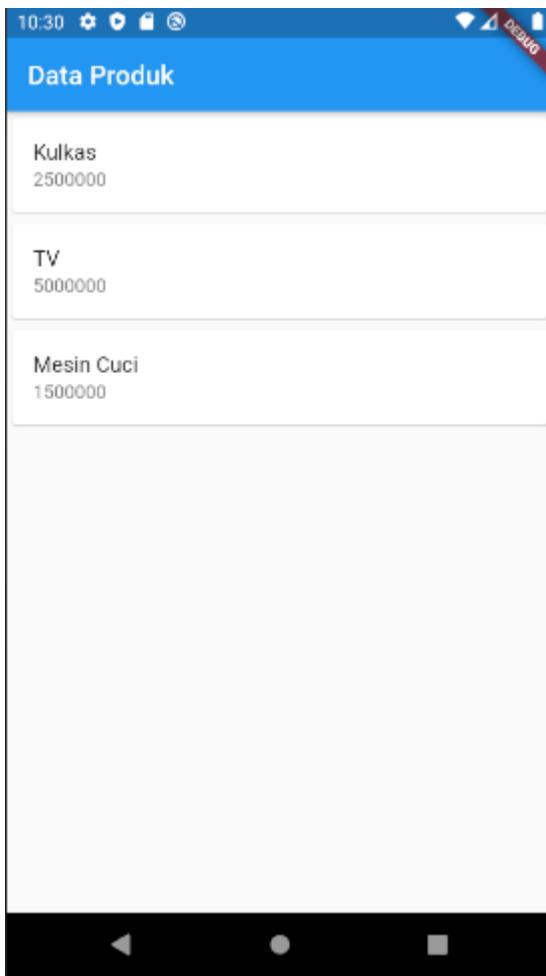
```

1. import 'package:flutter/material.dart';
2.
3. class ProdukPage extends StatefulWidget {
4.   const ProdukPage({Key? key}) : super(key: key);
5.

```

```
6.     @override
7.     _ProdukPageState createState() => _ProdukPageState();
8. }
9.
10. class _ProdukPageState extends State<ProdukPage> {
11.   @override
12.   Widget build(BuildContext context) {
13.     return Scaffold(
14.       appBar: AppBar(
15.         title: const Text('Data Produk'),
16.       ),
17.       body: ListView(
18.         children: const [
19.           // List 1
20.           Card(
21.             child: ListTile(
22.               title: Text('Kulkas'),
23.               subtitle: Text('2500000'),
24.             ),
25.           ),
26.           // List 2
27.           Card(
28.             child: ListTile(
29.               title: Text('TV'),
30.               subtitle: Text('5000000'),
31.             ),
32.           ),
33.           // List 3
34.           Card(
35.             child: ListTile(
36.               title: Text('Mesin Cuci'),
37.               subtitle: Text('1500000'),
38.             ),
39.           ),
40.         ],
41.       ),
42.     );
43.   }
44. }
```

Kemudian daftarkan **ProdukPage** pada **main.dart**, dan hasilnya akan menjadi seperti berikut



Membuat Route (Pindah Halaman)

Buka file **produk_page.dart**, kemudian modifikasi pada bagian **AppBar** menjadi seperti berikut

```
AppBar(  
    title: const Text("Data Produk"),  
    actions: [  
        GestureDetector(  
            // menampilkan icon +  
            child: const Icon(Icons.add),  
            //pada saat icon + di tap  
            onTap: () async {  
                //berpindah ke halaman ProdukForm  
                Navigator.push(  
                    context,  
                    MaterialPageRoute(  
                        builder: (context) => const ProdukForm()));  
            }  
    ],  
)
```

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

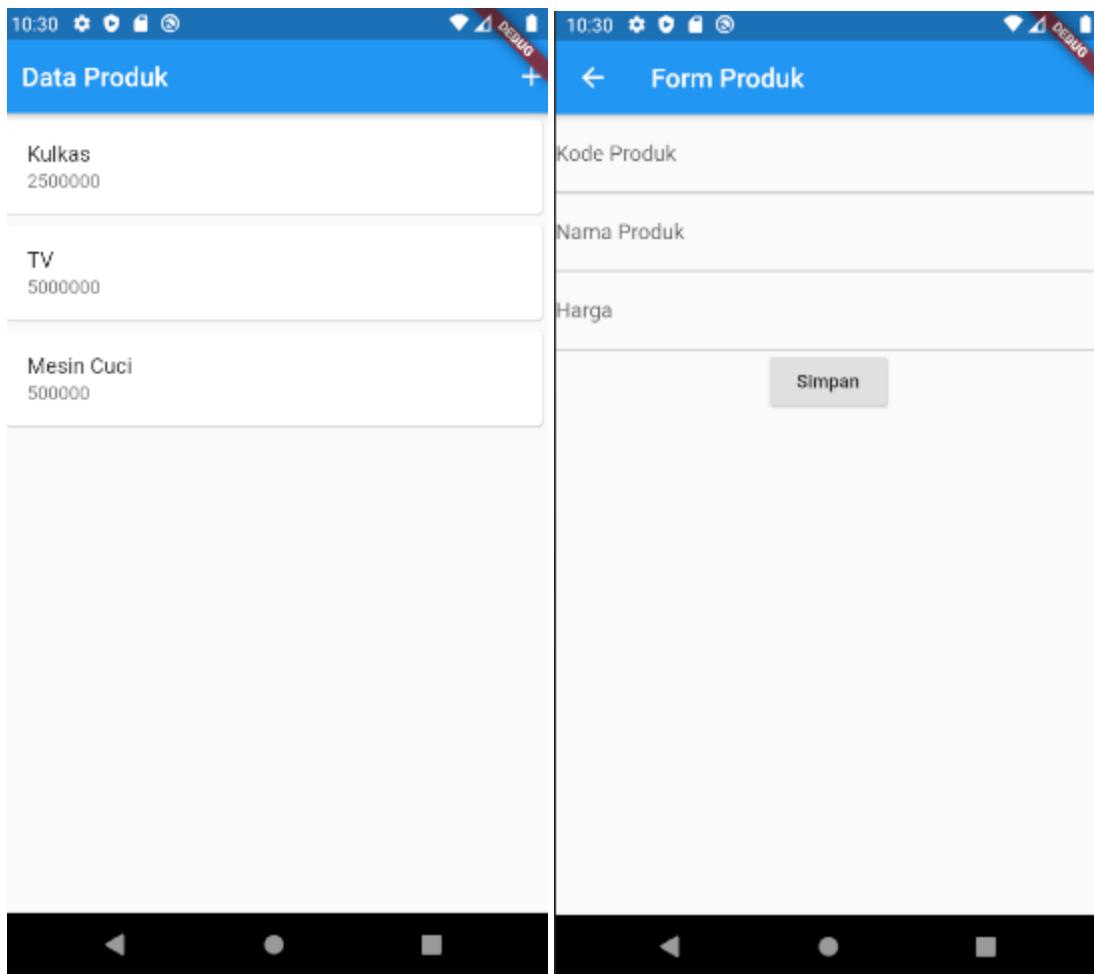
Secara Keseluruhan kode tersebut akan menjadi

```

1. import 'package:aplikasi_flutter_pertamaku/ui/produk_form.dart';
2. import 'package:flutter/material.dart';
3.
4. class ProdukPage extends StatefulWidget {
5.   const ProdukPage({Key? key}) : super(key: key);
6.
7.   @override
8.   _ProdukPageState createState() => _ProdukPageState();
9. }
10.
11. class _ProdukPageState extends State<ProdukPage> {
12.   @override
13.   Widget build(BuildContext context) {
14.     return Scaffold(
15.       appBar: AppBar(
16.         title: const Text('Data Produk'),
17.         actions: [
18.           GestureDetector(
19.             // menampilkan icon +
20.             child: const Icon(Icons.add),
21.             onTap: () async {
22.               // berpindah ke halaman ProdukForm
23.               Navigator.push(
24.                 context,
25.                 MaterialPageRoute(
26.                   builder: (context) => const ProdukForm()));
27.             })
28.           ],
29.         ),
30.       body: ListView(
31.         children: const [
32.           // List 1
33.           Card(
34.             child: ListTile(
35.               title: Text('Kulkas'),
36.               subtitle: Text('2500000'),
37.             ),
38.           ),
39.           // List 2
40.           Card(
41.             child: ListTile(
42.               title: Text('TV'),
43.               subtitle: Text('5000000'),
44.             ),
45.           ),
46.           // List 3
47.           Card(
48.             child: ListTile(
49.               title: Text('Mesin Cuci'),
50.               subtitle: Text('1500000'),
51.             ),
52.           )
53.         ],
54.       ),
55.     );
56.   }
57. }

```

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



Pemisahan Widget ke dalam Class StatelessWidget

Selain pemisahan widget ke dalam suatu function/method, pemisahan juga dapat dilakukan menggunakan class StatelessWidget, pada contoh kali ini kita akan memisahkan **Card** dengan membuat class tersendiri. Buka file `produk_page.dart`, kemudian buat sebuah class **ItemProduk** diluar class **ProdukPage**

```
class ItemProduk extends StatelessWidget {
    final String? kodeProduk;
    final String? namaProduk;
    final int? harga;

    const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})
        : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Card(
            child: ListTile(
                title: Text(namaProduk.toString()),
                subtitle: Text(harga.toString()),
            ),
        );
    }
}
```

Sehingga kode pada produk_page.dart menjadi

```
1. import 'package:aplikasi_flutter_pertamaku/ui/produk_form.dart';
2. import 'package:flutter/material.dart';
3.
4. class ProdukPage extends StatefulWidget {
5.   const ProdukPage({Key? key}) : super(key: key);
6.
7.   @override
8.   _ProdukPageState createState() => _ProdukPageState();
9. }
10.
11. class _ProdukPageState extends State<ProdukPage> {
12.   @override
13.   Widget build(BuildContext context) {
14.     return Scaffold(
15.       appBar: AppBar(
16.         title: const Text('Data Produk'),
17.         actions: [
18.           GestureDetector(
19.             // menampilkan icon +
20.             child: const Icon(Icons.add),
21.             onTap: () async {
22.               // berpindah ke halaman ProdukForm
23.               Navigator.push(
24.                 context,
25.                 MaterialPageRoute(
26.                   builder: (context) => const ProdukForm()));
27.             })
28.           ],
29.         ),
30.       body: ListView(
31.         children: const [
32.           // List 1
33.           ItemProduk(
34.             kodeProduk: "A001",
35.             namaProduk: "Kulkas",
36.             harga: 2500000,
37.           ),
38.           // List 2
39.           ItemProduk(
40.             kodeProduk: "A002",
41.             namaProduk: "TV",
42.             harga: 5000000,
43.           ),
44.           // List 3
45.           ItemProduk(
46.             kodeProduk: "A003",
47.             namaProduk: "Mesin Cuci",
48.             harga: 1500000,
49.           ),
50.         ],
51.       ),
52.     );
53.   }
54. }
55.
56. class ItemProduk extends StatelessWidget {
57.   final String? kodeProduk;
58.   final String? namaProduk;
59.   final int? harga;
60.
```

```

61. const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})
62.   : super(key: key);
63.
64. @override
65. Widget build(BuildContext context) {
66.   return Card(
67.     child: ListTile(
68.       title: Text(nombre.toString()),
69.       subtitle: Text(precio.toString()),
70.     ),
71.   );
72. }
73. 
```

Menampilkan Detail Produk saat ListView diklik

Pada bagian ini kita akan menambahkan sebuah fungsi dimana saat salah satu ListView Produk di klik, maka akan membuka halaman Detail Produk yang telah kita buat sebelumnya

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

```

56 class ItemProduk extends StatelessWidget {
57   final String? kodeProduk;
58   final String? nombreProduk;
59   final int? precio;
60
61   const ItemProduk({Key? key, this.kodeProduk, this.nombreProduk, this.precio})
62   |   : super(key: key);
63
64   @override
65   Widget build(BuildContext context) {
66     return Card(
67       child: ListTile(
68         title: Text(nombre.toString()),
69         subtitle: Text(precio.toString()),
70       ), // ListTile
71     ); // Card
72   }
73 } 
```

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

```
56 class ItemProduk extends StatelessWidget {  
57     final String? kodeProduk;  
58     final String? namaProduk;  
59     final int? harga;  
60  
61     const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})  
62         : super(key: key);  
63  
64     Remove this widget  
65     Wrap with Builder  
66     Wrap with Center  
67     Wrap with Column  
68     Wrap with Container  
69     Wrap with Padding  
70     Wrap with Row  
71     Wrap with SizedBox  
72     Wrap with StreamBuilder  
73 } Wrap with widget...  
74 Extract Method  
Extract Local Variable  
Extract Widget
```

Ln 66, Col 12 Spaces: 2 UTF-8 CRLF Dart

Kemudian akan menjadi

```
56 class ItemProduk extends StatelessWidget {  
57     final String? kodeProduk;  
58     final String? namaProduk;  
59     final int? harga;  
60  
61     const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})  
62         : super(key: key);  
63  
64     @override  
65     Widget build(BuildContext context) {  
66         return widget(  
67             child: Card(  
68                 child: ListTile(  
69                     title: Text(namaProduk.toString()),  
70                     subtitle: Text(harga.toString()),  
71                     ), // ListTile  
72                 ), // Card  
73             );  
74     }  
75 }  
76
```

Setelah itu ubah **widget** menjadi GestureDetector dan kita juga menambahkan onTap yang kemudian akan membuka halaman Detail Produk, sehingga kode untuk Class ItemProduk menjadi

```
class ItemProduk extends StatelessWidget {
    final String? kodeProduk;
    final String? namaProduk;
    final int? harga;

    const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})
        : super(key: key);

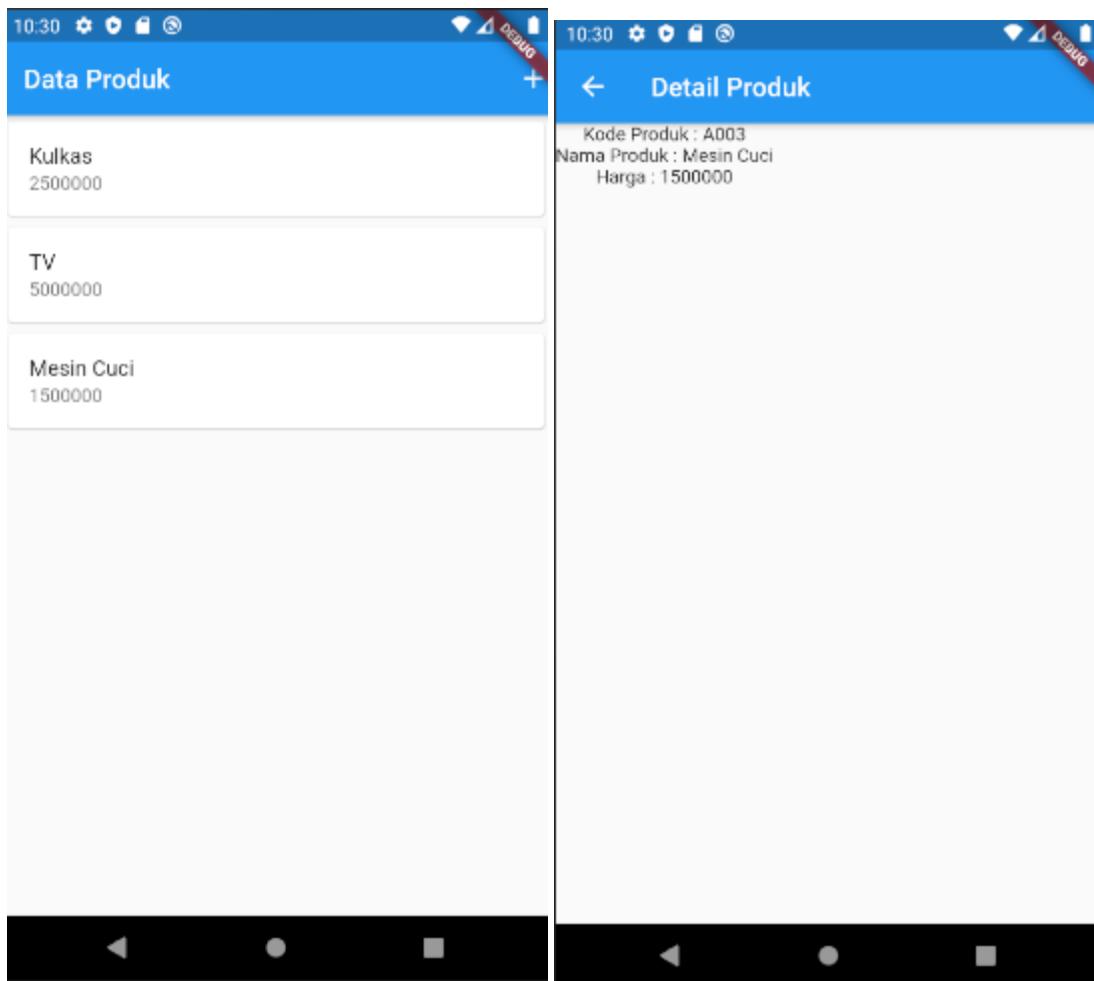
    @override
    Widget build(BuildContext context) {
        return GestureDetector(
            child: Card(
                child: ListTile(
                    title: Text(namaProduk.toString()),
                    subtitle: Text(harga.toString()),
                ),
            ),
            onTap: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => ProdukDetail(
                            kodeProduk: kodeProduk,
                            namaProduk: namaProduk,
                            harga: harga,
                        )));
            },
        );
    }
}
```

Adapun untuk keseluruhan kode pada **produk_page.dart** sebagai berikut

```
1. import 'package:aplikasi_flutter_pertamaku/ui/produk_detail.dart';
2. import 'package:aplikasi_flutter_pertamaku/ui/produk_form.dart';
3. import 'package:flutter/material.dart';
4.
5. class ProdukPage extends StatefulWidget {
6.     const ProdukPage({Key? key}) : super(key: key);
7.
8.     @override
9.     _ProdukPageState createState() => _ProdukPageState();
10. }
11.
12. class _ProdukPageState extends State<ProdukPage> {
13.     @override
14.     Widget build(BuildContext context) {
15.         return Scaffold(
16.             appBar: AppBar(
17.                 title: const Text('Data Produk'),
18.                 actions: [
19.                     GestureDetector(
20.                         // menampilkan icon +
21.                         child: const Icon(Icons.add),
22.                         onTap: () async {
23.                             // berpindah ke halaman ProdukForm
```

```

24.             Navigator.push(
25.                 context,
26.                 MaterialPageRoute(
27.                     builder: (context) => const ProdukForm())));
28.             })
29.         ],
30.     ),
31.     body: ListView(
32.         children: const [
33.             // List 1
34.             ItemProduk(
35.                 kodeProduk: "A001",
36.                 namaProduk: "Kulkas",
37.                 harga: 2500000,
38.             ),
39.             // List 2
40.             ItemProduk(
41.                 kodeProduk: "A002",
42.                 namaProduk: "TV",
43.                 harga: 5000000,
44.             ),
45.             // List 3
46.             ItemProduk(
47.                 kodeProduk: "A003",
48.                 namaProduk: "Mesin Cuci",
49.                 harga: 1500000,
50.             ),
51.         ],
52.     ),
53. );
54. }
55.
56.
57. class ItemProduk extends StatelessWidget {
58.     final String? kodeProduk;
59.     final String? namaProduk;
60.     final int? harga;
61.
62.     const ItemProduk({Key? key, this.kodeProduk, this.namaProduk, this.harga})
63.         : super(key: key);
64.
65.     @override
66.     Widget build(BuildContext context) {
67.         return GestureDetector(
68.             child: Card(
69.                 child: ListTile(
70.                     title: Text(namaProduk.toString()),
71.                     subtitle: Text(harga.toString()),
72.                 ),
73.             ),
74.             onTap: () {
75.                 Navigator.push(
76.                     context,
77.                     MaterialPageRoute(
78.                         builder: (context) => ProdukDetail(
79.                             kodeProduk: kodeProduk,
80.                             namaProduk: namaProduk,
81.                             harga: harga,
82.                         )));
83.             },
84.         );
85.     }
86. }
```



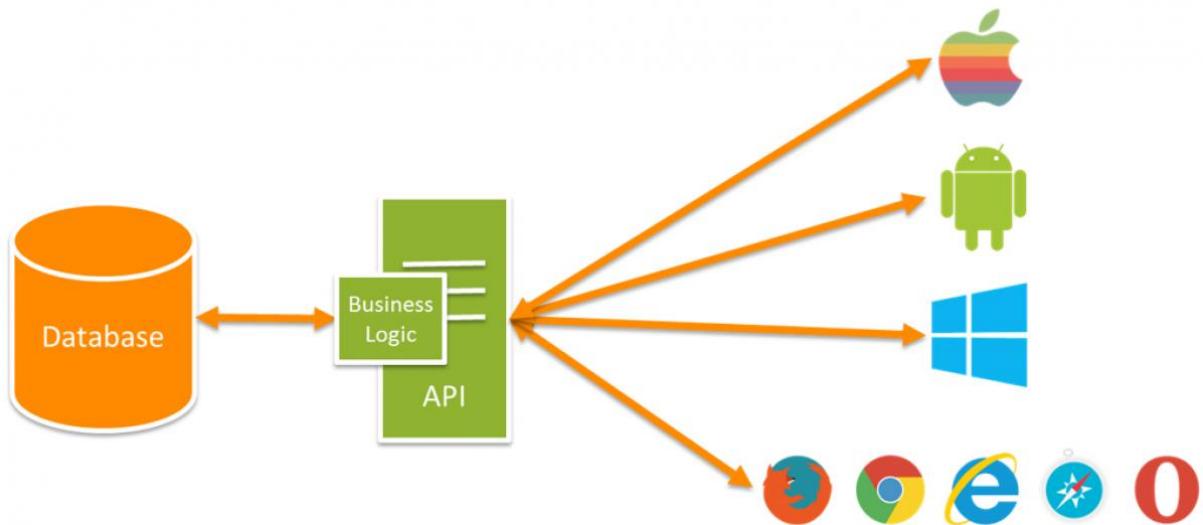
Membuat projek flutter yang terhubung dengan API

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.

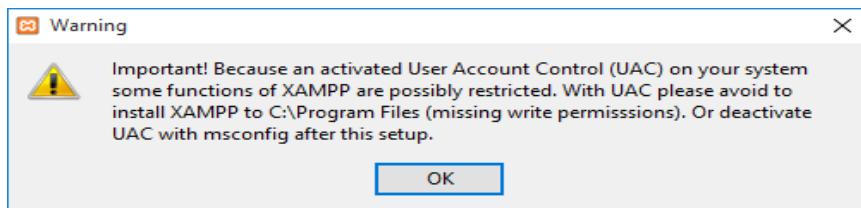
Membuat projek Toko API (Restful API)

Installasi Apache, MySQL dan PHP (XAMPP)

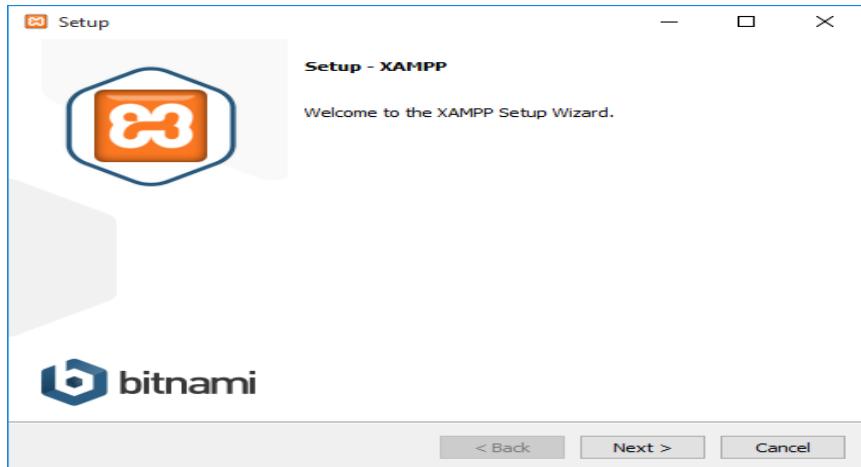
XAMPP adalah perangkat lunak untuk membuat komputer menjadi web server. XAMPP dapat di download melalui link url:

[1.](#)

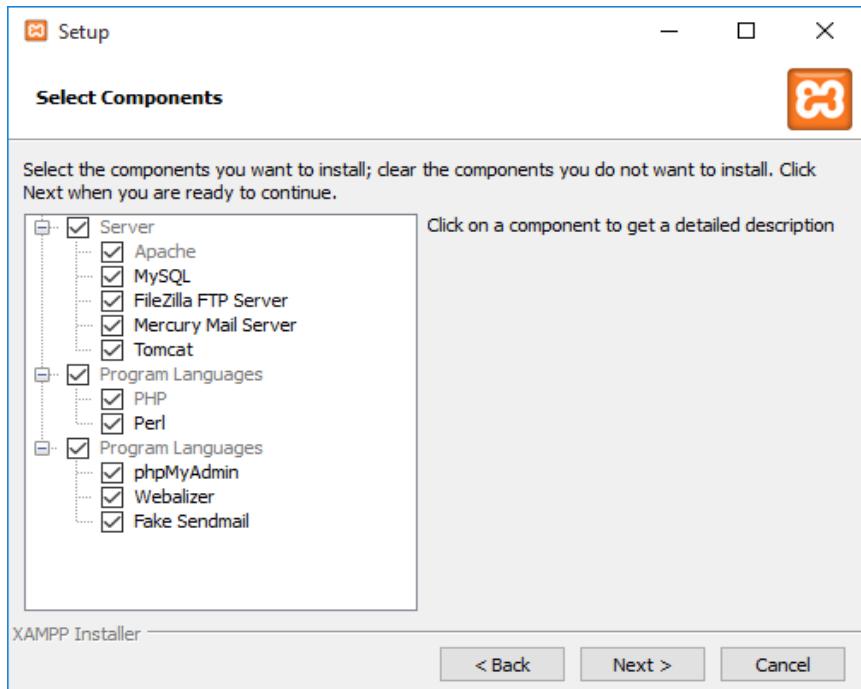
Setelah itu, double klik file xampp yang baru di download. Jika muncul pesan error seperti gambar dibawah, abaikan saja dan lanjutkan klik tombol OK.



Berikutnya akan muncul jendela **Setup-XAMPP**. Klik tombol **Next** untuk melanjutkan proses berikutnya.

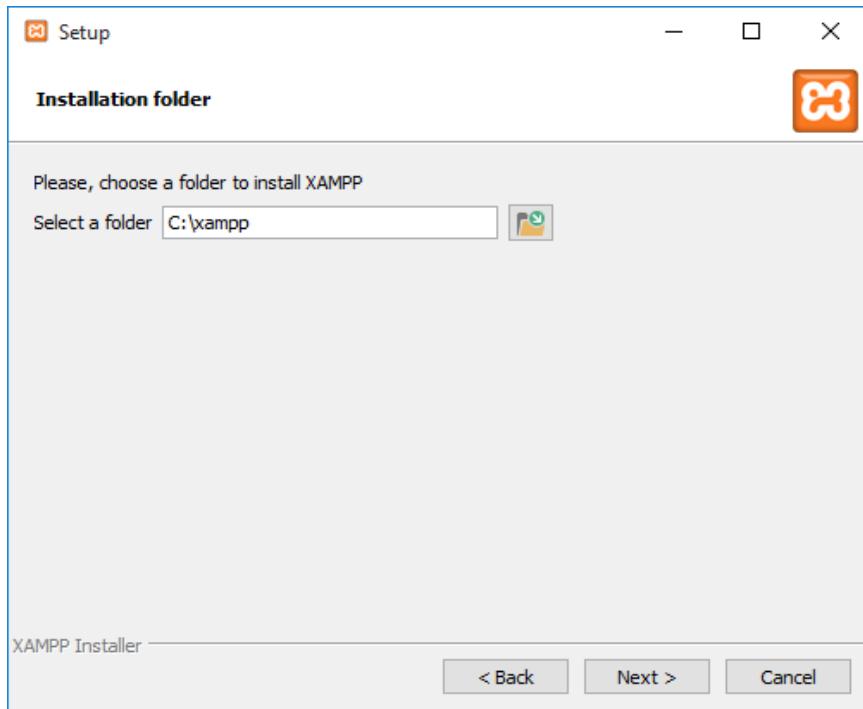


Selanjutnya akan muncul jendela **Select Components**, yang meminta untuk memilih aplikasi yang akan diinstall. Centang saja semua kemudian klik tombol **Next**.

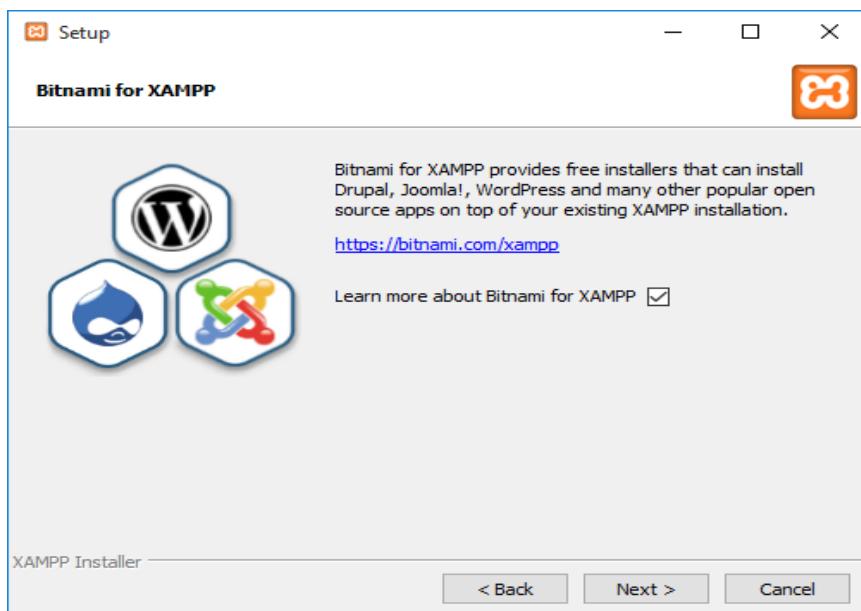


Kemudian akan muncul jendela **Installation Folder**, dimana anda diminta untuk menentukan lokasi penyimpanan folder xampp, secara bawaan akan diarahkan ke lokasi **c:\xampp**. Jika anda

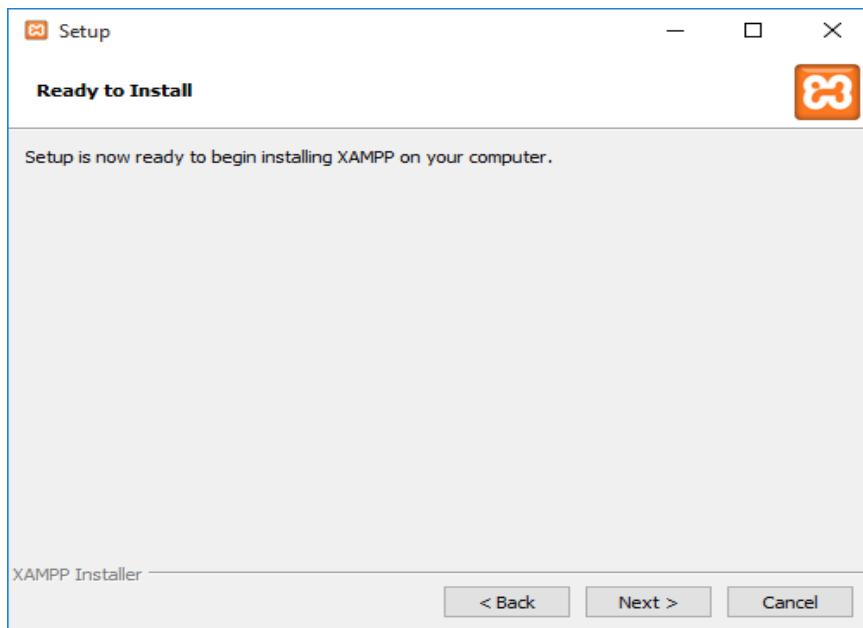
ingin menyimpannya di folder lain, anda dapat menekan tombol bergambar folder (**Browse**), kemudian klik tombol **Next**.



Kita akan menjumpai jendela tawaran untuk mempelajari lebih lanjut tentang Bitnami. Silahkan checklist jika ingin mempelajari lebih lanjut. Bitnami adalah pustaka dari aplikasi client server yang populer seperti misalnya CMS WordPress atau Drupal, dengan penawaran kemudahan dalam installasi hanya dengan satu klik. Kemudian klik tombol **Next**.



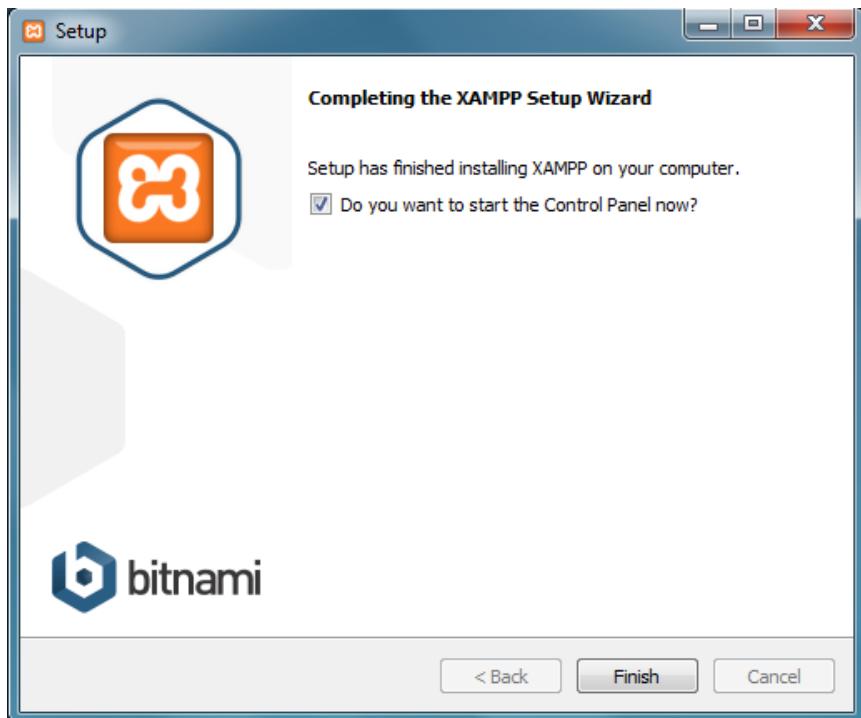
Kemudian muncul jendela **Ready To Install** yang menunjukkan xampp sudah siap di install. Kemudian klik tombol **Next**.



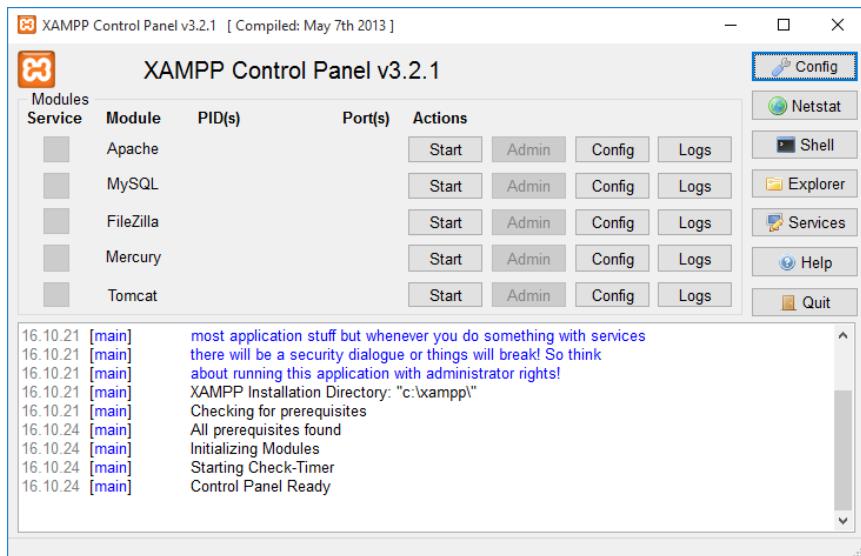
Dan proses installasi pun berjalan.



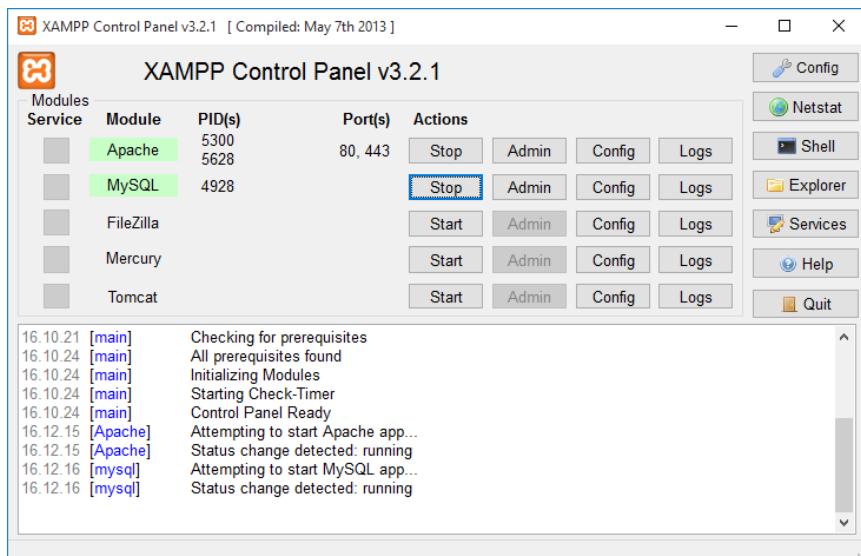
Tunggu hingga proses install selesai dan muncul jendela sebagai berikut.



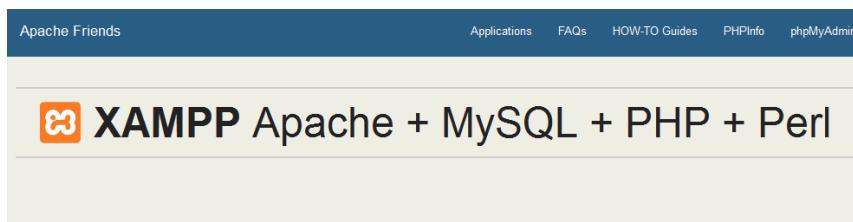
Klik tombol **Finish** setelah itu akan muncul jendela **Xampp Control Panel** yang berguna untuk menjalankan server.



Klik tombol **Start** pada kolom Actions untuk module **Apache** dan **MySQL**.



Untuk mengetahui apakah installasi telah berhasil atau tidak, ketikkan ‘localhost/’ pada browser, jika berhasil akan muncul halaman seperti gambar dibawah.



Welcome to XAMPP for Windows 5.6.12

You have successfully installed XAMPP on this system! Now you can start using Apache, MySQL, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

Start the XAMPP Control Panel to check the server status.

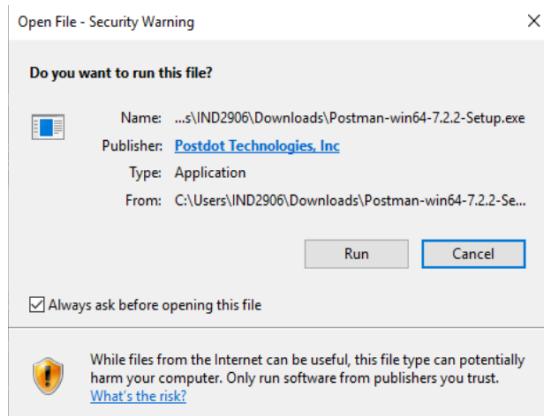
Install Postman

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).

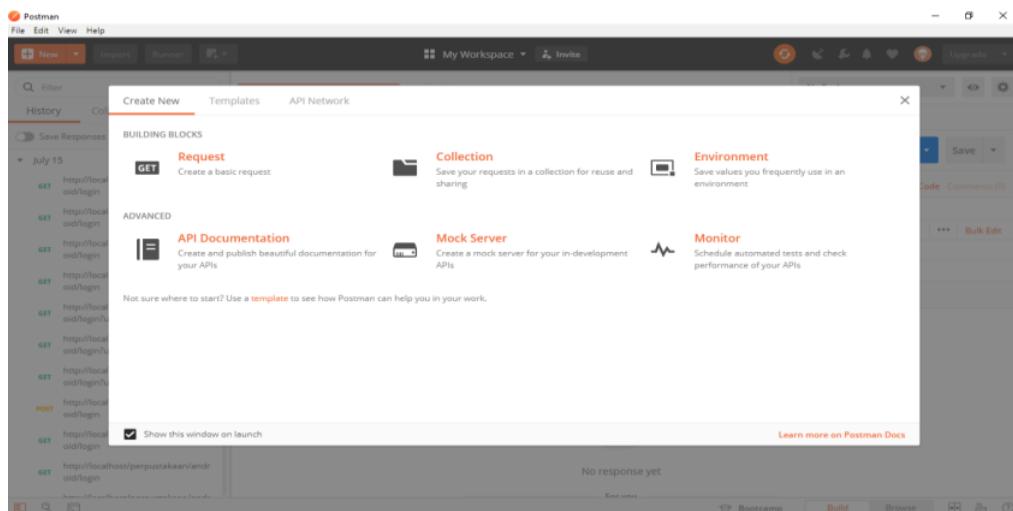
Postman tersedia sebagai aplikasi asli untuk sistem operasi macOS, Windows (32-bit dan 64-bit), dan Linux (32-bit dan 64-bit). Untuk mendapatkan aplikasi Postman, dapat diunduh

pada website resminya yaitu [getpostman.com](https://www.postman.com/downloads/) 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



API SPEC

API SPEC ini bermaksud untuk membuat standar API sebagai dokumentasi kepada pengembang baik itu frontend maupun backend

A. Registrasi

EndPoint	/registrasi
Method	POST

Header	<ul style="list-style-type: none"> Content-Type: application/json
Body	{ "nama" : "string", "email" : "string, unique", "password" : "string" }
Response	{ "code" : "integer", "status" : "boolean", "data" : "string" }

B. Login

EndPoint	/login
Method	POST
Header	<ul style="list-style-type: none"> Content-Type: application/json
Body	{ "email" : "string", "password" : "string" }
Response	{ "code" : "integer", "status" : "boolean", "data" : { "token" : "string", "user" : { "id" : "integer", "email" : "string", } } }

C. Produk

1. List Produk

EndPoint	/produk
Method	GET
Header	<ul style="list-style-type: none"> Content-Type: application/json
Response	{ "code" : "integer", "status" : "boolean", "data" : [}

```

{
  "id" : "integer",
  "kode_produk" : "string",
  "nama_produk" : "string",
  "harga" : "integer",
},
{
  "id" : "integer",
  "kode_produk" : "string",
  "nama_produk" : "string",
  "harga" : "integer",
}
]
}

```

2. Create Produk

EndPoint	/produk
Method	POST
Header	<ul style="list-style-type: none"> Content-Type: application/json
Body	{ "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer" }
Response	{ "code" : "integer", "status" : "boolean", "data" : { "id" : "integer", "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer", } }

3. Update Produk

EndPoint	/produk/{id}/update
Method	POST
Header	<ul style="list-style-type: none"> Content-Type: application/json
Body	{ "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer" }

Response	<pre>{ "code" : "integer", "status" : "boolean", "data" : "boolean" }</pre>
----------	--

4. Show Produk

EndPoint	/produk/{id}
Method	GET
Header	<ul style="list-style-type: none"> Content-Type: application/json
Response	<pre>{ "code" : "integer", "status" : "boolean", "data" : { "id" : "integer", "kode_produk" : "string", "nama_produk" : "string", "harga" : "integer", } }</pre>

5. Delete Produk

EndPoint	/produk/{id}
Method	DELETE
Header	<ul style="list-style-type: none"> Content-Type: application/json
Response	<pre>{ "code" : "integer", "status" : "boolean", "data" : "boolean" }</pre>

Pembuatan Database

Buat database dengan nama : **toko_api**

Kemudian buat tabel-tabel dengan perintah sebagai berikut :

- SQL membuat tabel member

```
CREATE table member (
    id INT NOT NULL AUTO_INCREMENT,
```

```

    nama VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    PRIMARY KEY(id)
);

```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(11)			No	<i>None</i>		AUTO_INCREMENT
2	nama	varchar(255)	utf8mb4_general_ci		No	<i>None</i>		
3	email	varchar(255)	utf8mb4_general_ci		No	<i>None</i>		
4	password	varchar(255)	utf8mb4_general_ci		No	<i>None</i>		

2. SQL membuat tabel member_token

```

CREATE table member_token (
    id INT NOT NULL AUTO_INCREMENT,
    member_id INT NOT NULL,
    auth_key VARCHAR(255) NOT NULL,
    FOREIGN KEY (member_id) REFERENCES member(id) on update cascade on delete no action,
    PRIMARY KEY(id)
);

```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(11)			No	<i>None</i>		AUTO_INCREMENT
2	member_id 	int(11)			No	<i>None</i>		
3	auth_key	varchar(255)	utf8mb4_general_ci		No	<i>None</i>		

3. SQL membuat tabel produk

```

CREATE table produk (
    id INT NOT NULL AUTO_INCREMENT,
    kode_produk VARCHAR(255) NOT NULL,
    nama_produk VARCHAR(255) NOT NULL,
    harga INT NOT NULL,
    PRIMARY KEY(id)
);

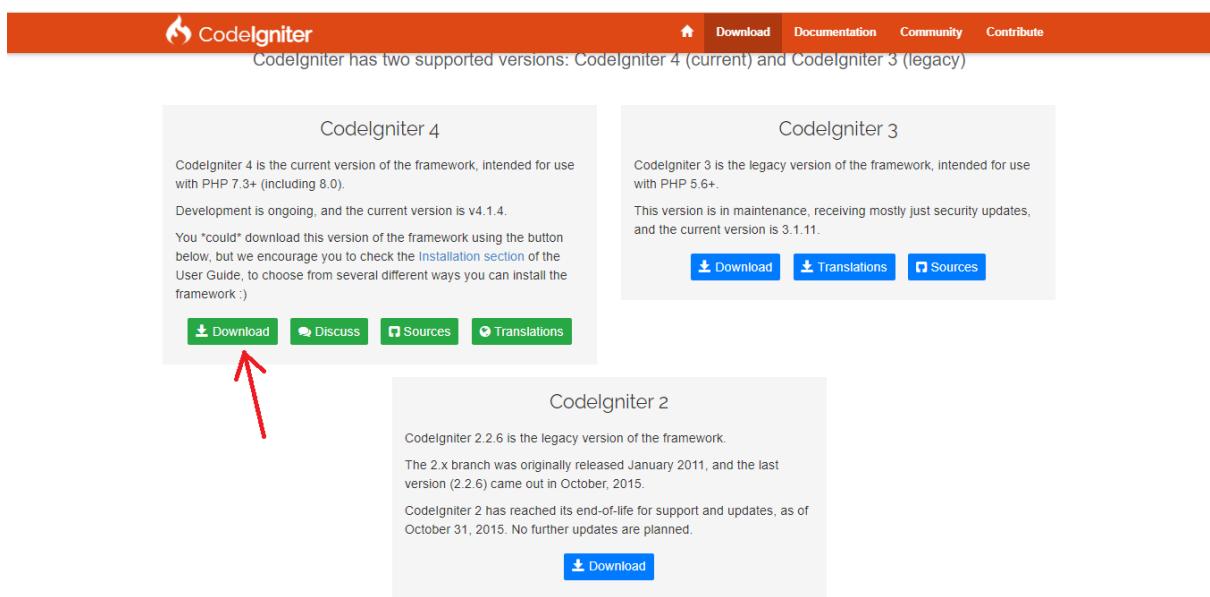
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(11)			No	None		AUTO_INCREMENT
2	kode_produk	varchar(255)	utf8mb4_general_ci		No	None		
3	nama_produk	varchar(255)	utf8mb4_general_ci		No	None		
4	harga	int(11)			No	None		

Installasi CodeIgniter 4 sebagai Restful API

Selanjutnya install projek CodeIgniter pada web server (pada folder C:\xampp\htdocs)

Framework CodeIgniter 4 dapat di unduh di <https://codeigniter.com/download>

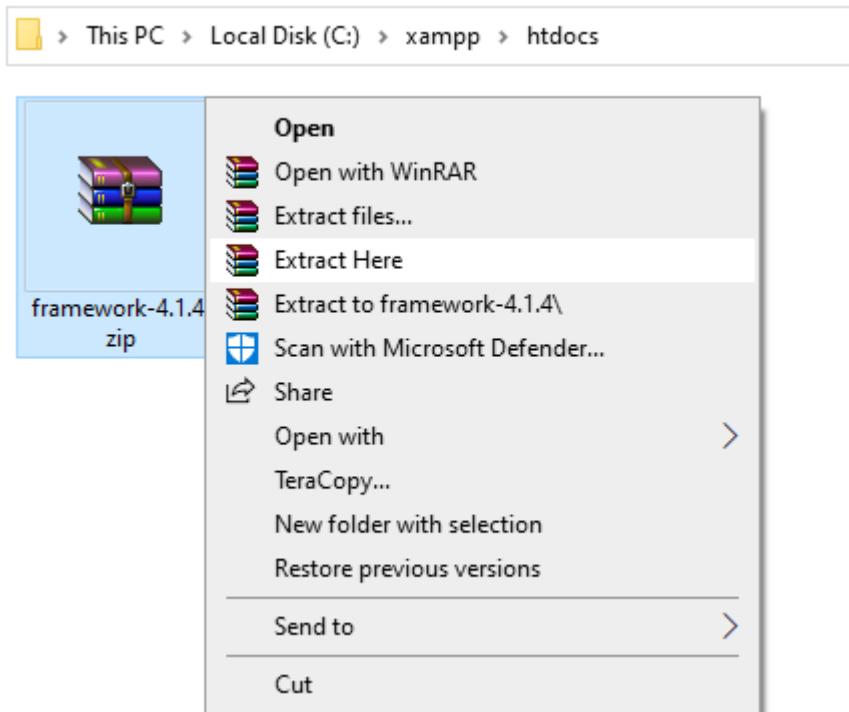


The screenshot shows the official CodeIgniter download page. At the top, there's a navigation bar with links for Home, Download, Documentation, Community, and Contribute. Below the navigation, it says "CodeIgniter has two supported versions: CodeIgniter 4 (current) and CodeIgniter 3 (legacy)".

There are three main sections:

- CodeIgniter 4**: Described as the current version for PHP 7.3+ (including 8.0). It mentions ongoing development and the current version v4.1.4. It includes a note about "could" download the legacy version. Below this, there are four buttons: Download (highlighted with a red arrow), Discuss, Sources, and Translations.
- CodeIgniter 3**: Described as the legacy version for PHP 5.6+. It mentions maintenance and the current version 3.1.11. Below this, there are three buttons: Download, Translations, and Sources.
- CodeIgniter 2**: Described as the legacy version released in January 2011. It mentions end-of-life support ended on October 31, 2015. Below this, there is one button: Download.

Setelah di unduh ekstrak file CodeIgniter pada folder **C:\xampp\htdocs**



Kemudian ubah nama folder menjadi **toko-api**. Pada projek buka file **Database.php** yang terletak pada folder **app\config**, kemudian cari kode berikut

```
public $default = [
    'DSN'      => '',
    'hostname' => 'localhost',
    'username' => '',
    'password' => '',
    'database' => '',
    'DBDriver'  => 'MySQLi',
    'DBPrefix'  => '',
    'pConnect'  => false,
    'DBDebug'   => (ENVIRONMENT !== 'production'),
    'charset'   => 'utf8',
    'DBCollat'  => 'utf8_general_ci',
    'swapPre'   => '',
    'encrypt'   => false,
    'compress'  => false,
    'strictOn'  => false,
    'failover'  => [],
    'port'      => 3306,
];
```

Untuk menghubungkan projek dengan database kita akan mengisikan **hostname**, **username**, **password** dan **database** menjadi

```
public $default = [
    'DSN'      => '',
    'hostname' => 'localhost',
```

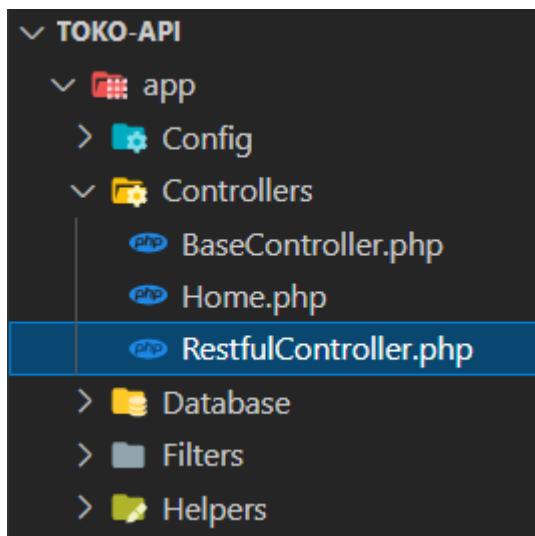
```

        'username' => 'root',
        'password' => '',
        'database' => 'toko_api',
        'DBDriver' => 'MySQLI',
        'DBPrefix' => '',
        'pConnect' => false,
        'DBDebug' => (ENVIRONMENT !== 'production'),
        'charset' => 'utf8',
        'DBCollat' => 'utf8_general_ci',
        'swapPre' => '',
        'encrypt' => false,
        'compress' => false,
        'strictOn' => false,
        'failover' => [],
        'port' => 3306,
    ];

```

Membuat hasil response

Buat sebuah file dengan nama **RestfulController.php** pada folder **app\Controllers**



Kemudian tambahkan kode pada file tersebut sehingga menjadi

```

1. <?php
2.
3. namespace App\Controllers;
4.
5. use CodeIgniter\RESTful\ResourceController;
6.
7. class RestfulController extends ResourceController
8. {
9.
10.     protected $format = 'json';
11.
12.     protected function responseHasil($code, $status, $data)
13.     {
14.         return $this->respond([

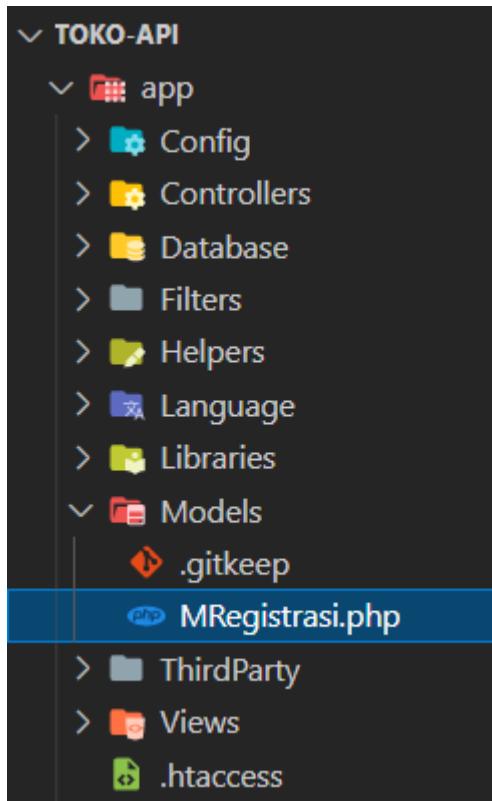
```

```
15.          'code' => $code,
16.          'status' => $status,
17.          'data' => $data
18.      ]);
19.  }
20. }
```

Registrasi

Membuat model Registrasi

Buat sebuah file dengan nama **MRegistrasi.php** pada folder **app/Models**

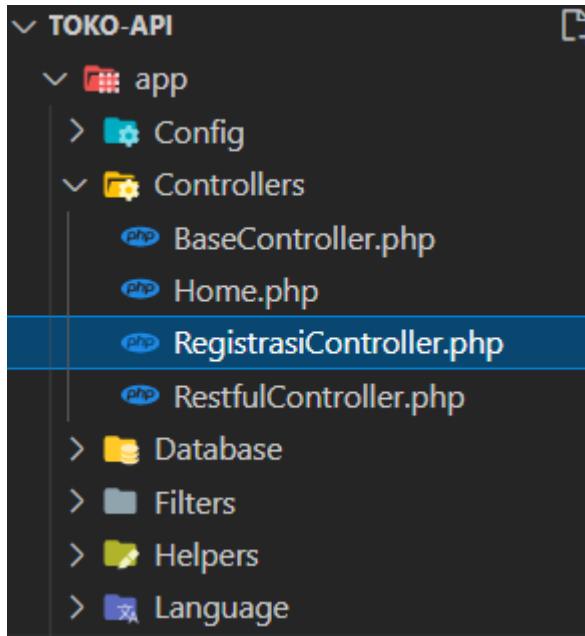


Kemudian pada file **MRegistrasi.php** ketikkan kode berikut

```
1. <?php
2.
3. namespace App\Models;
4.
5. use CodeIgniter\Model;
6.
7. class MRegistrasi extends Model
8. {
9.     protected $table = 'member';
10.    protected $allowedFields = ['nama', 'email', 'password'];
11. }
```

Membuat controller Registrasi

Buat sebuah file dengan nama **RegistrasiController.php** pada folder **app\Controllers**



Kemudian pada file **RegistrasiController.php** tersebut masukkan kode berikut

```
1. <?php
2.
3. namespace App\Controllers;
4.
5. use App\Models\MRegistrasi;
6.
7. class RegistrasiController extends RestfulController
8. {
9.
10.     public function registrasi()
11.     {
12.         $data = [
13.             'nama' => $this->request->getVar('nama'),
14.             'email' => $this->request->getVar('email'),
15.             'password' => password_hash($this->request->getVar('password'),
16.             PASSWORD_DEFAULT)
17.         ];
18.         $model = new MRegistrasi();
19.         $model->save($data);
20.         return $this->responseHasil(200, true, "Registrasi Berhasil");
21.     }
22. }
```

Menambah route Registrasi

Buka file **Routes.php** pada folder **app\Config**

Kemudian lihat baris ke 34, telah terdapat routing dengan method get, kita akan menambahkan routing untuk registrasi sehingga kode menjadi seperti berikut

```

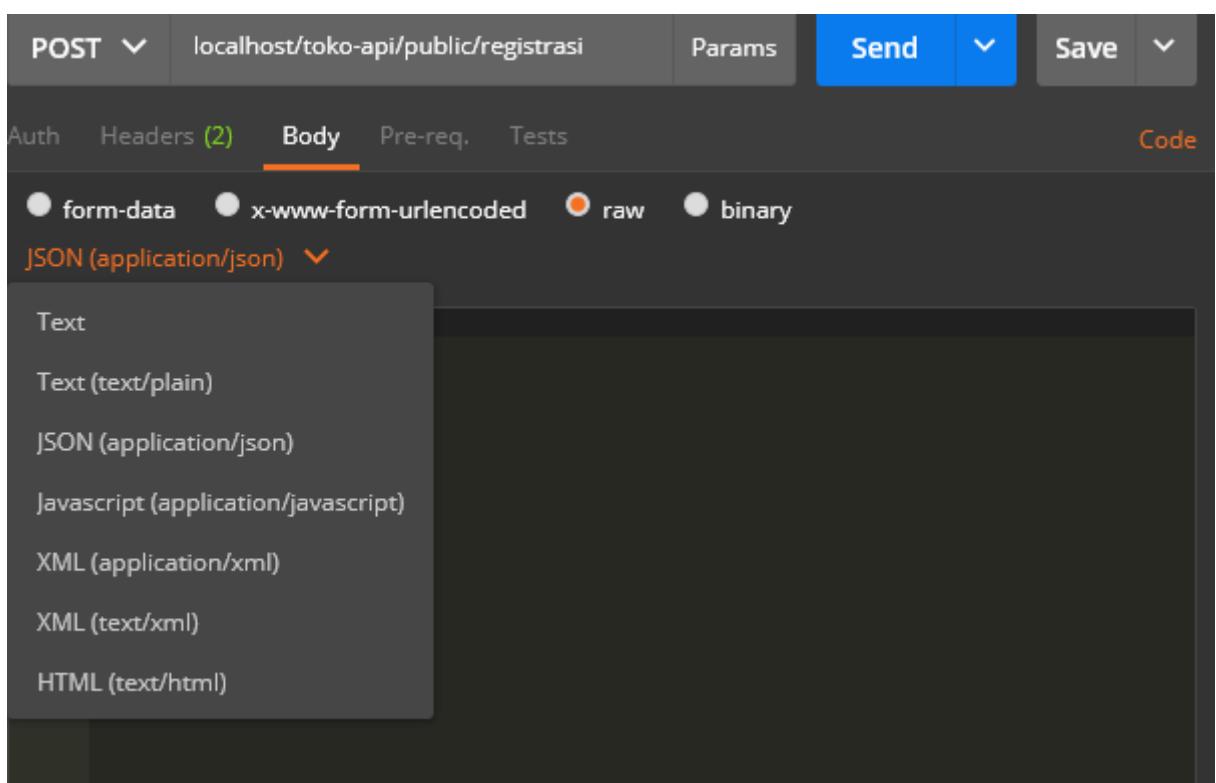
/*
* -----
* Route Definitions
* -----
*/
// We get a performance increase by specifying the default
// route since we don't have to scan directories.
$routes->get('/', 'Home::index');
$routes->post('/registrasi', 'RegistrasiController::registrasi');

```

Pada baris terakhir kita menambahkan routing registrasi agar dapat diakses dengan method POST. Untuk mengakses registrasi, kita gunakan Postman dengan alamat url **localhost/toko-api/public/registrasi** dengan method POST

Adapun langkah menggunakan postman untuk menguji Rest API yang telah dibuat adalah sebagai berikut:

1. Buka aplikasi Postman yang telah terinstall
2. Masukkan alamat url untuk melakukan registrasi toko-api yang telah kita buat yaitu <http://localhost/toko-api/public/registrasi>
3. Selanjutnya pilih pengujian dengan type POST
4. Kemudian klik **Body** yang berada pada bagian bawah inputan url, kemudian pilih **raw**
5. Kemudian dibagian bawah pilihan **raw**, pilih **JSON (application/json)**



6. Kemudian isi dengan format JSON sesuai dengan field request pada **RegistrasiController** pada fungsi registrasi yaitu *nama*, *email*, *password* kemudian klik tombol **Send**

```
$data = [
    'nama' => $this->request->getPost('nama'),
    'email' => $this->request->getPost('email'),
    'password' => password_hash($this->request-
>getPost('password'), PASSWORD_DEFAULT)
];
```

Contoh isian data

```
{
    "nama": "Administrator",
    "email": "admin@admin.com",
    "password": "admin"
}
```

Login

Membuat model Member

Buat sebuah file dengan nama **MMember.php** pada folder **app\Models** dan ketikkan kode berikut

```
1. <?php
2.
3. namespace App\Models;
4.
5. use CodeIgniter\Model;
6.
7. class MMember extends Model
8. {
9.     protected $table = 'member';
10. }
```

Membuat model Login

Buat sebuah file dengan nama **MLogin.php** pada folder **app\Models** dan ketikkan kode berikut

```

1. <?php
2.
3. namespace App\Models;
4.
5. use CodeIgniter\Model;
6.
7. class MLogin extends Model
8. {
9.     protected $table = 'member_token';
10.    protected $allowedFields = ['member_id', 'auth_key'];
11. }

```

Membuat controller Login

Buat sebuah file dengan nama **LoginController.php** pada folder **app/Controllers** dan ketikkan kode berikut

```

1. <?php
2.
3. namespace App\Controllers;
4.
5. use App\Models\MLogin;
6. use App\Models\MMember;
7.
8. class LoginController extends RestfulController
9. {
10.
11.     public function login()
12.     {
13.         $email = $this->request->getVar('email');
14.         $password = $this->request->getVar('password');
15.
16.         $model = new MMember();
17.         $member = $model->where(['email' => $email])->first();
18.         if (!$member) {
19.             return $this->responseHasil(400, false, "Email tidak ditemukan");
20.         }
21.         if (!password_verify($password, $member['password'])) {
22.             return $this->responseHasil(400, false, "Password tidak valid");
23.         }
24.
25.         $login = new MLogin();
26.         $auth_key = $this->RandomString();
27.         $login->save([
28.             'member_id' => $member['id'],
29.             'auth_key' => $auth_key
30.         ]);
31.         $data = [
32.             'token' => $auth_key,
33.             'user' => [
34.                 'id' => $member['id'],
35.                 'email' => $member['email'],
36.             ]
37.         ];
38.         return $this->responseHasil(200, true, $data);
39.     }
40.
41.     private function RandomString($length = 100)
42.     {
43.         $karakter =
44.             '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
45.         $panjang_karakter = strlen($karakter);
46.         $str = '';

```

```

46.         for ($i = 0; $i < $length; $i++) {
47.             $str .= $karakter[rand(0, $panjang_karakter - 1)];
48.         }
49.     return $str;
50. }
51. 
```

Menambahkan route Login

Pada route tambahkan kode untuk mengakses login sehingga menjadi sebagai berikut

```
$routes->post('/registrasi', 'RegistrasiController::registrasi');
$routes->post('/login', 'LoginController::login');
```

Mencoba Rest

Silahkan coba Rest API dengan memasukkan url <http://localhost/toko-api/public/login> dengan method POST

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost/toko-api/public/login
- Headers:** (2 items)
- Body:** (JSON application/json)


```
{
1: "email": "admin@admin.com",
2: "password": "admin"
3: }
```
- Code:** Status: 200 OK Time: 211 ms
- Body (Pretty):**

```
1: {
2:   "code": 200,
3:   "status": true,
4:   "data": {
5:     "token": "Mdj8yPsp8cNbckNe1dmMikggebcYgrU2duFIztyIGJIdFdN3Ak
YNgLzvds9Km9mffrQ1sKeQDKOzbHKElVbbNswofPiEuN91Cy",
6:     "user": {
7:       "id": "1",
8:       "email": "admin@admin.com"
9:     }
10:   }
11: }
```

CRUD Produk

Membuat model Produk

Buat sebuah file dengan nama **MProduk.php** pada folder **app/Models** dan ketikkan kode berikut

```

1. <?php
2.
3. namespace App\Models;
4.
5. use CodeIgniter\Model;
6.
7. class MProduk extends Model
8. {
9.     protected $table = 'produk';
10.    protected $primaryKey = 'id';
11.    protected $allowedFields = ['kode_produk', 'nama_produk', 'harga'];
12. } 
```

Membuat controller produk

Buat sebuah file dengan nama **ProdukController** pada folder **app/Controllers** dan ketikkan kode berikut

```

1. <?php
2.
3. namespace App\Controllers;
4.
5. use App\Models\MProduk;
6.
7. class ProdukController extends RestfulController
8. {
9.
10. }
```

Selanjutnya pada class **ProdukController** kita akan menambahkan fungsi (function) CRUD produk, yaitu:

Membuat fungsi create produk

```

public function create()
{
    $data = [
        'kode_produk' => $this->request->getVar('kode_produk'),
        'nama_produk' => $this->request->getVar('nama_produk'),
        'harga' => $this->request->getVar('harga')
    ];

    $model = new MProduk();
    $model->insert($data);
    $produk = $model->find($model->getInsertID());
    return $this->responseHasil(200, true, $produk);
}
```

Membuat fungsi list produk

```

public function list()
{
    $model = new MProduk();
    $produk = $model->findAll();
    return $this->responseHasil(200, true, $produk);
}
```

Membuat fungsi tampil produk

```

public function detail($id)
{
    $model = new MProduk();
    $produk = $model->find($id);
    return $this->responseHasil(200, true, $produk);
}
```

Membuat fungsi update produk

```

public function ubah($id)
{
    $data = [
        'kode_produk' => $this->request->getVar('kode_produk'),
        'nama_produk' => $this->request->getVar('nama_produk'),
        'harga' => $this->request->getVar('harga')
    ];

    $model = new MProduk();
    $model->update($id, $data);
    $produk = $model->find($id);

    return $this->responseHasil(200, true, $produk);
}
```

Membuat fungsi delete produk

```
public function hapus($id)
{
    $model = new MProduk();
    $produk = $model->delete($id);

    return $this->responseHasil(200, true, $produk);
}
```

Sehingga adapun keseluruhan kode **ProdukController.php** adalah sebagai berikut

```
1. <?php
2.
3. namespace App\Controllers;
4.
5. use App\Models\MProduk;
6.
7. class ProdukController extends RestfulController
8. {
9.     public function create()
10.    {
11.        $data = [
12.            'kode_produk' => $this->request->getVar('kode_produk'),
13.            'nama_produk' => $this->request->getVar('nama_produk'),
14.            'harga' => $this->request->getVar('harga')
15.        ];
16.
17.        $model = new MProduk();
18.        $model->insert($data);
19.        $produk = $model->find($model->getInsertID());
20.        return $this->responseHasil(200, true, $produk);
21.    }
22.
23.    public function list()
24.    {
25.        $model = new MProduk();
26.        $produk = $model->findAll();
27.        return $this->responseHasil(200, true, $produk);
28.    }
29.
30.    public function detail($id)
31.    {
32.        $model = new MProduk();
33.        $produk = $model->find($id);
34.        return $this->responseHasil(200, true, $produk);
35.    }
36.
37.    public function ubah($id)
38.    {
39.        $data = [
40.            'kode_produk' => $this->request->getVar('kode_produk'),
41.            'nama_produk' => $this->request->getVar('nama_produk'),
42.            'harga' => $this->request->getVar('harga')
43.        ];
44.
45.        $model = new MProduk();
46.        $model->update($id, $data);
47.        $produk = $model->find($id);
48.
49.        return $this->responseHasil(200, true, $produk);
50.    }
}
```

```

51.
52.     public function hapus($id)
53.     {
54.         $model = new MProduk();
55.         $produk = $model->delete($id);
56.
57.         return $this->responseHasil(200, true, $produk);
58.     }
59.

```

Menambahkan route produk

Agar ProdukController dapat diakses, selanjutnya tambah route untuk mengakses produk pada file **app/Config/Routes.php**

```

$routes->group('produk', function ($routes) {
    $routes->post('/', 'ProdukController::create');
    $routes->get('/', 'ProdukController::list');
    $routes->get('(:segment)', 'ProdukController::detail/$1');
    $routes->put('(:segment)', 'ProdukController::ubah/$1');
    $routes->delete('(:segment)', 'ProdukController::hapus/$1');
}) ;

```

Adapun keseluruhan kode pada **app/Config/Routes.php** adalah sebagai berikut

```

$routes->get('/', 'Home::index');
$routes->post('/registrasi', 'RegistrasiController::registrasi');
$routes->post('/login', 'LoginController::login');

$routes->group('produk', function ($routes) {
    $routes->post('/', 'ProdukController::create');
    $routes->get('/', 'ProdukController::list');
    $routes->get('(:segment)', 'ProdukController::detail/$1');
    $routes->put('(:segment)', 'ProdukController::ubah/$1');
    $routes->delete('(:segment)', 'ProdukController::hapus/$1');
}) ;

```

Mencoba Rest

Create Produk (localhost/toko-api/public/produk) dengan method POST

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost/toko-api/public/produk
- Headers:** (2)
- Body:** (Green dot indicating JSON)
- Auth:** (None)
- Params:** (None)
- Send** button
- Save** dropdown
- Status:** 200 OK
- Time:** 148 ms
- Code:** (Grey tab)
- Body:** (Orange tab)
- Cookies:** (Grey tab)
- Headers:** (8)
- Test Results:** (Grey tab)
- Pretty:** (Grey tab)
- Raw:** (Grey tab)
- Preview:** (Grey tab)
- JSON:** (Grey tab)
- Save Response** button

Body (Pretty):

```

1  {
2      "kode_produk": "A001",
3      "nama_produk": "Realme Smart TV 50 Inch",
4      "harga": 5700000
5  }

```

Body (Raw):

```

{
  "kode_produk": "A001",
  "nama_produk": "Realme Smart TV 50 Inch",
  "harga": 5700000
}

```

List Produk (localhost/toko-api/public/produk) dengan method GET

The screenshot shows the Postman interface with a successful GET request to the '/produk' endpoint. The response status is 200 OK and the time taken is 120 ms. The response body is a JSON object:

```
1 v {  
2   "code": 200,  
3   "status": true,  
4   "data": [  
5     {  
6       "id": "1",  
7       "kode_produk": "A001",  
8       "nama_produk": "Realme Smart TV 50 Inch",  
9       "harga": "5700000"  
10    }  
11  ]  
12 }
```

Show Produk (localhost/toko-api/public/produk/{id}) dengan method GET

The screenshot shows the Postman interface with a successful GET request to the '/produk/1' endpoint. The response status is 200 OK and the time taken is 195 ms. The response body is a JSON object:

```
1 v {  
2   "code": 200,  
3   "status": true,  
4   "data": {  
5     "id": "1",  
6     "kode_produk": "A001",  
7     "nama_produk": "Realme Smart TV 50 Inch",  
8     "harga": "5700000"  
9   }  
10 }
```

Update Produk (localhost/toko-api/public/produk/{id}) dengan method PUT

The screenshot shows the Postman interface with a successful PUT request to the '/produk/1' endpoint. The response status is 200 OK and the time taken is 117 ms. The response body is a JSON object:

```
1 v {  
2   "code": 200,  
3   "status": true,  
4   "data": {  
5     "id": "1",  
6     "kode_produk": "A001",  
7     "nama_produk": "Samsung TV 60 Inch",  
8     "harga": "7800000"  
9   }  
10 }
```

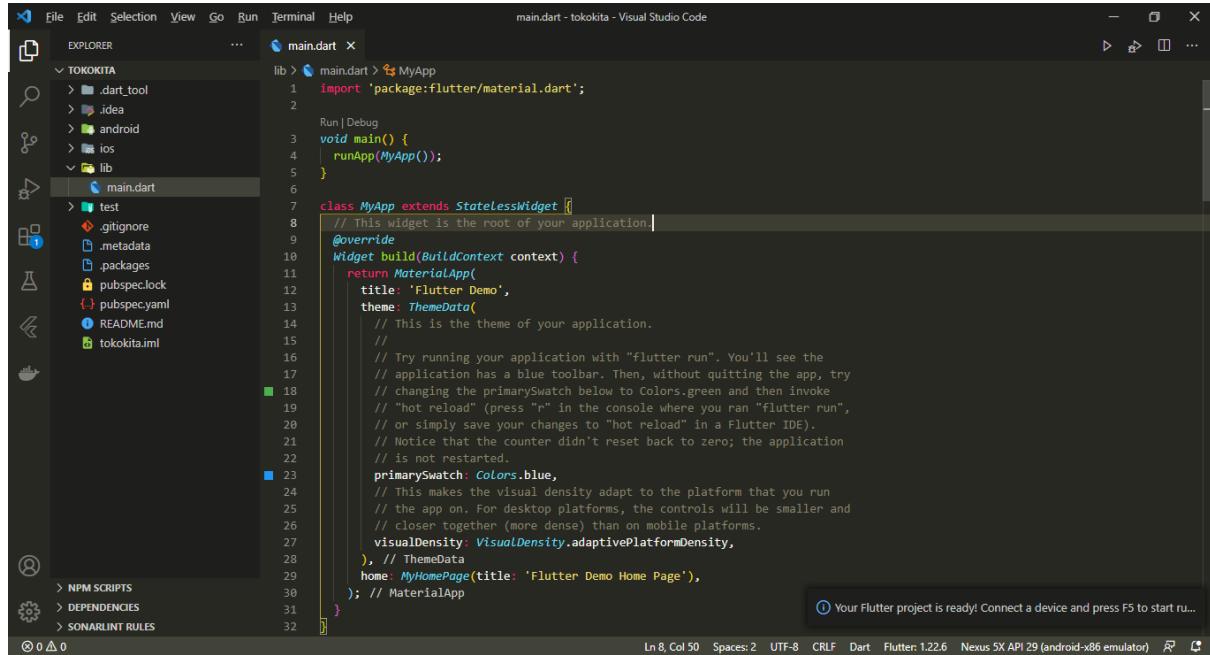
Delete Produk (localhost/toko-api/public/produk) dengan method DELETE

The screenshot shows the Postman interface with a successful DELETE request to the '/produk/3' endpoint. The response status is 200 OK and the time taken is 184 ms. The response body is a JSON object:

```
1 v {  
2   "code": 200,  
3   "status": true,  
4   "data": true  
5 }
```

Membuat projek flutter tokokita

Buat sebuah projek flutter dengan nama **tokokita**



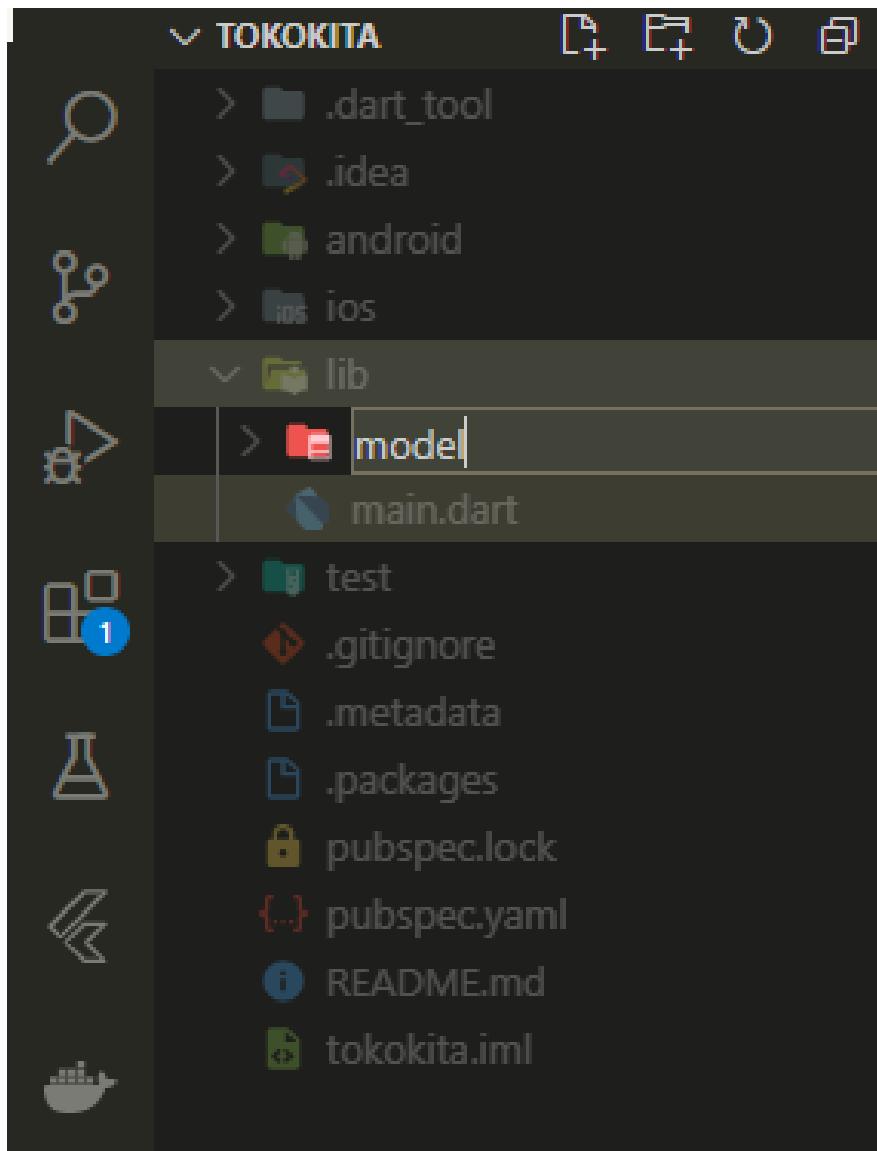
```
lib > main.dart > MyApp
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: '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 8, Col 50 Spaces: 2 UTF-8 CRLF Dart Flutter: 1.22.6 Nexus 5X API 29 (android-x86 emulator)

Your Flutter project is ready! Connect a device and press F5 to start ru...

Membut Model

Buat folder dengan nama **model** pada folder **lib**



Login

Buat sebuah file dengan nama **login.dart** pada folder **model**. Kemudian masukkan kode berikut

```
1. class Login {  
2.   int? code;  
3.   bool? status;  
4.   String? token;  
5.   int? userID;  
6.   String? userEmail;  
7.  
8.   Login({this.code, this.status, this.token, this.userID, this.userEmail});  
9.  
10.  factory Login.fromJson(Map<String, dynamic> obj) {  
11.    return Login(  
12.      code: obj['code'],  
13.      status: obj['status'],  
14.      token: obj['data']['token'],  
15.      userID: obj['data']['user']['id'],
```

```
16.     userEmail: obj['data']['user']['email']);
17. }
18. }
```

Registrasi

Buat sebuah file dengan nama **registrasi.dart** pada folder **model**. Kemudian masukkan kode berikut

```
1. class Registrasi {
2.   int? code;
3.   bool? status;
4.   String? data;
5.
6.   Registrasi({this.code, this.status, this.data});
7.
8.   factory Registrasi.fromJson(Map<String, dynamic> obj) {
9.     return Registrasi(
10.       code: obj['code'],
11.       status: obj['status'],
12.       data: obj['data']);
13.   }
14. }
```

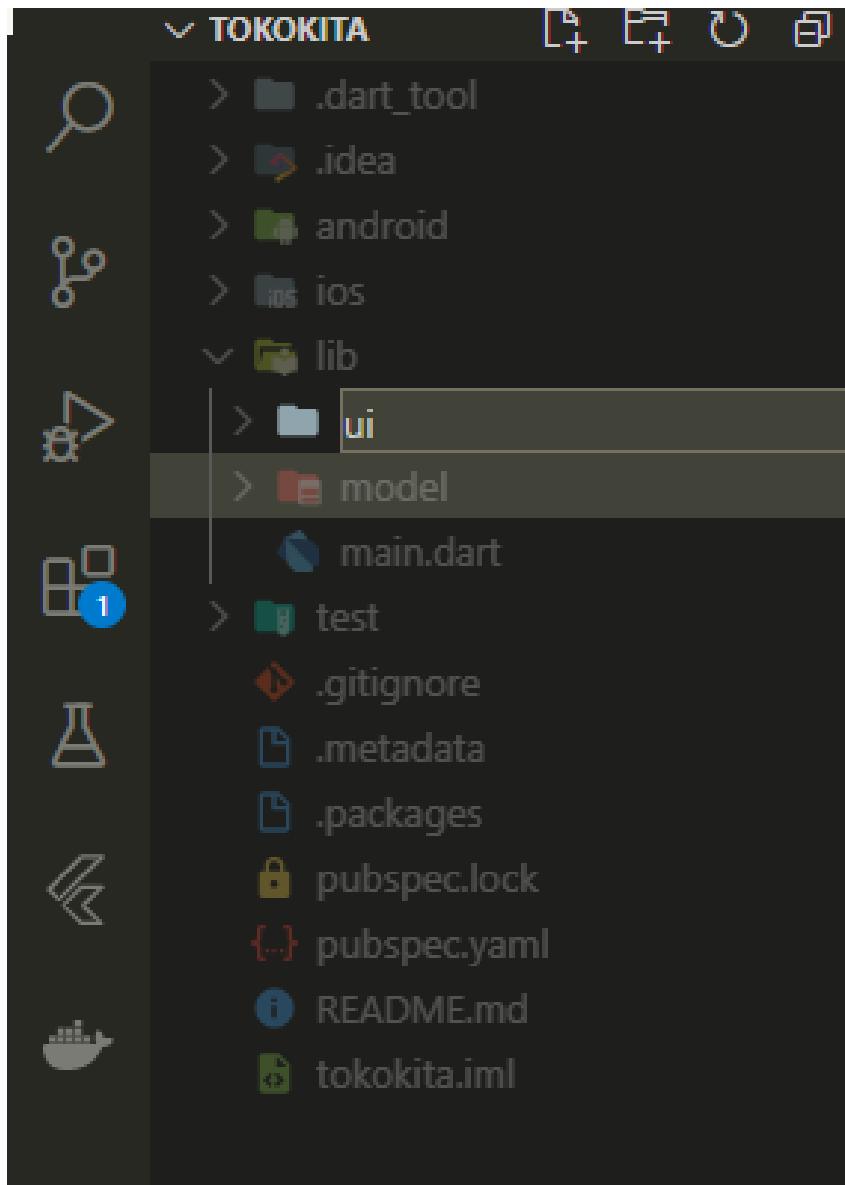
Produk

Buat sebuah file dengan nama **produk.dart** pada folder **model**. Kemudian ketikkan kode berikut

```
1. class Produk {
2.   int? id;
3.   String? kodeProduk;
4.   String? namaProduk;
5.   int? hargaProduk;
6.
7.   Produk({this.id, this.kodeProduk, this.namaProduk, this.hargaProduk});
8.
9.   factory Produk.fromJson(Map<String, dynamic> obj) {
10.     return Produk(
11.       code: obj['id'],
12.       kodeProduk: obj['kode_produk'],
13.       namaProduk: obj['nama_produk'],
14.       hargaProduk: obj['harga']
15.     );
16.   }
17. }
```

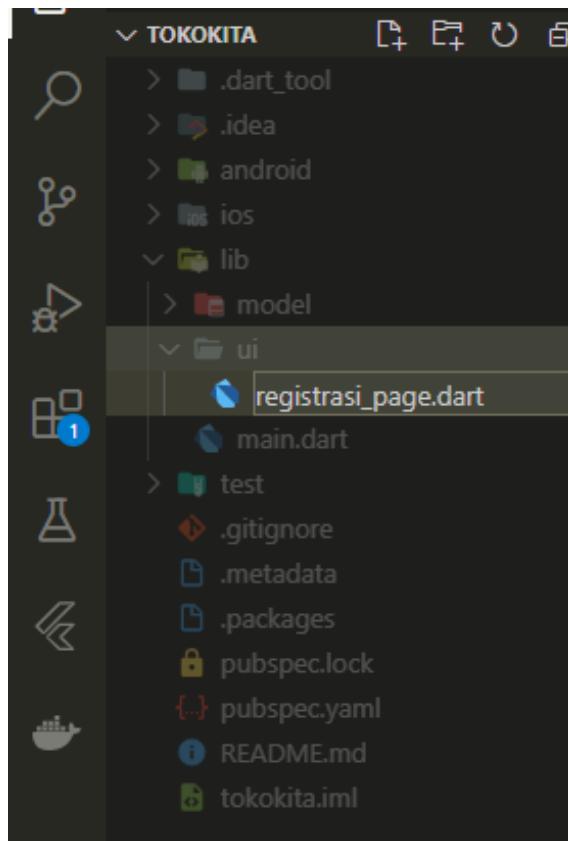
Membuat Halaman

Pertama kita akan memecah bagian-bagian kode menjadi beberapa bagian, adapun untuk tampilan, dikelompokkan kedalam folder **ui**.



Registrasi

Buat sebuah file dengan nama **registrasi_page.dart** pada folder ui.



Pada file **registrasi_page.dart** ketikkan kode berikut

```
1. import 'package:flutter/material.dart';
2.
3. class RegistrasiPage extends StatefulWidget {
4.   const RegistrasiPage({Key? key}) : super(key: key);
5.
6.   @override
7.   _RegistrasiPageState createState() => _RegistrasiPageState();
8. }
9.
10. class _RegistrasiPageState extends State<RegistrasiPage> {
11.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
12.   bool _isLoading = false;
13.
14.   final TextEditingController _namaTextboxController = TextEditingController();
15.   final TextEditingController _emailTextboxController = TextEditingController();
16.   final TextEditingController _passwordTextboxController = TextEditingController();
17.
18.   @override
19.   Widget build(BuildContext context) {
20.     return Scaffold(
21.       appBar: AppBar(
22.         title: const Text("Registrasi"),
23.       ),
24.       body: SingleChildScrollView(
25.         child: Padding(
26.           padding: const EdgeInsets.all(8.0),
27.           child: Form(
28.             key: _formKey,
29.             child: Column(
```

```
30.     mainAxisAlignment: MainAxisAlignment.center,
31.     children: [
32.         _namaTextField(),
33.         _emailTextField(),
34.         _passwordTextField(),
35.         _passwordKonfirmasiTextField(),
36.         _buttonRegistrasi()
37.     ],
38. ),
39. ),
40. ),
41. );
42. );
43. }
44.
45. //Membuat Textbox Nama
46. Widget _namaTextField() {
47.     return TextFormField(
48.         decoration: const InputDecoration(labelText: "Nama"),
49.         keyboardType: TextInputType.text,
50.         controller: _namaTextboxController,
51.         validator: (value) {
52.             if (value!.length < 3) {
53.                 return "Nama harus diisi minimal 3 karakter";
54.             }
55.             return null;
56.         },
57.     );
58. }
59.
60. //Membuat Textbox email
61. Widget _emailTextField() {
62.     return TextFormField(
63.         decoration: const InputDecoration(labelText: "Email"),
64.         keyboardType: TextInputType.emailAddress,
65.         controller: _emailTextboxController,
66.         validator: (value) {
67.             //validasi harus diisi
68.             if (value!.isEmpty) {
69.                 return 'Email harus diisi';
70.             }
71.             //validasi email
72.             Pattern pattern =
73.                 r'^(([<>()[]\\.,;:\\s@"]+([^.][<>()[]\\.,;:\\s@"]+)*|(".+"))@(([0-9]{1,3}\\.\\{0-9}{1,3}\\.{0-9}{1,3}\\.{0-9}{1,3}\\])|(([a-zA-Z\\-0-9]+\\.)+[a-zA-Z]{2,}))$';
74.             RegExp regex = RegExp(pattern.toString());
75.             if (!regex.hasMatch(value)) {
76.                 return "Email tidak valid";
77.             }
78.             return null;
79.         },
80.     );
81. }
82.
83. //Membuat Textbox password
84. Widget _passwordTextField() {
85.     return TextFormField(
86.         decoration: const InputDecoration(labelText: "Password"),
87.         keyboardType: TextInputType.text,
88.         obscureText: true,
89.         controller: _passwordTextboxController,
90.         validator: (value) {
91.             //jika karakter yang dimasukkan kurang dari 6 karakter
92.             if (value!.length < 6) {
```

```

93.         return "Password harus diisi minimal 6 karakter";
94.     }
95.     return null;
96.   },
97. );
98. }
99.
100.    //membuat textbox Konfirmasi Password
101.    Widget _passwordKonfirmasiTextField() {
102.      return TextFormField(
103.        decoration: const InputDecoration(labelText: "Konfirmasi Password"),
104.        keyboardType: TextInputType.text,
105.        obscureText: true,
106.        validator: (value) {
107.          //jika inputan tidak sama dengan password
108.          if (value != _passwordTextboxController.text) {
109.            return "Konfirmasi Password tidak sama";
110.          }
111.          return null;
112.        },
113.      );
114.    }
115.
116.    //Membuat Tombol Registrasi
117.    Widget _buttonRegistrasi() {
118.      return ElevatedButton(
119.        child: const Text("Registrasi"),
120.        onPressed: () {
121.          var validate = _formKey.currentState!.validate();
122.        });
123.    }
124.  }

```

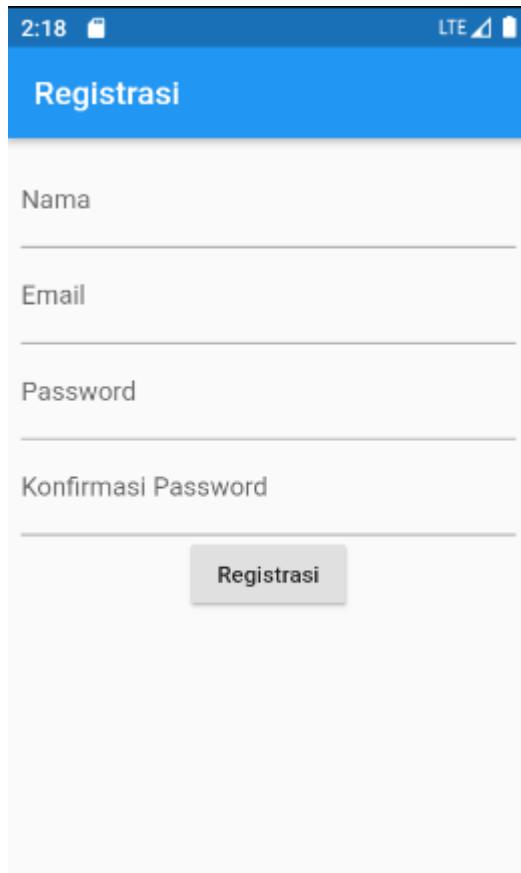
Untuk mencoba halaman `registrasi_page`, buka file `main.dart` kemudian ubah kode menjadi seperti berikut ini

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/registrasi_page.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: 'Toko Kita',
15.     debugShowCheckedModeBanner: false,
16.     home: RegistrasiPage(),
17.   );
18. }
19. }

```

Dan hasilnya seperti berikut



Login

Buat sebuah file dengan nama `login_page.dart` pada folder `ui` dengan kode berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/registrasi_page.dart';
3.
4. class LoginPage extends StatefulWidget {
5.   const LoginPage({Key? key}) : super(key: key);
6.
7.   @override
8.   _LoginPageState createState() => _LoginPageState();
9. }
10.
11. class _LoginPageState extends State<LoginPage> {
12.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
13.   bool _isLoading = false;
14.
15.   final TextEditingController _emailTextboxController = TextEditingController();
16.   final TextEditingController _passwordTextboxController = TextEditingController();
17.
18.   @override
19.   Widget build(BuildContext context) {
20.     return Scaffold(
21.       appBar: AppBar(
22.         title: const Text('Login'),
23.       ),
24.       body: SingleChildScrollView(
25.         child: Padding(
26.           padding: const EdgeInsets.all(8.0),
27.           child: Form(
28.             key: _formKey,
```

```

29.         child: Column(
30.             children: [
31.                 _emailTextField(),
32.                 _passwordTextField(),
33.                 _buttonLogin(),
34.                 const SizedBox(
35.                     height: 30,
36.                 ),
37.                 _menuRegistrasi()
38.             ],
39.         ),
40.     ),
41. ),
42. ),
43. );
44. }
45.
46. //Membuat Textbox email
47. Widget _emailTextField() {
48.     return TextFormField(
49.         decoration: const InputDecoration(labelText: "Email"),
50.         keyboardType: TextInputType.emailAddress,
51.         controller: _emailTextboxController,
52.         validator: (value) {
53.             //validasi harus diisi
54.             if (value!.isEmpty) {
55.                 return 'Email harus diisi';
56.             }
57.             return null;
58.         },
59.     );
60. }
61.
62. //Membuat Textbox password
63. Widget _passwordTextField() {
64.     return TextFormField(
65.         decoration: const InputDecoration(labelText: "Password"),
66.         keyboardType: TextInputType.text,
67.         obscureText: true,
68.         controller: _passwordTextboxController,
69.         validator: (value) {
70.             //jika karakter yang dimasukkan kurang dari 6 karakter
71.             if (value!.isEmpty) {
72.                 return "Password harus diisi";
73.             }
74.             return null;
75.         },
76.     );
77. }
78.
79. //Membuat Tombol Login
80. Widget _buttonLogin() {
81.     return ElevatedButton(
82.         child: const Text("Login"),
83.         onPressed: () {
84.             var validate = _formKey.currentState!.validate();
85.         });
86. }
87.
88. // Membuat menu untuk membuka halaman registrasi
89. Widget _menuRegistrasi() {
90.     return Center(
91.         child: InkWell(
92.             child: const Text(
93.                 "Registrasi",

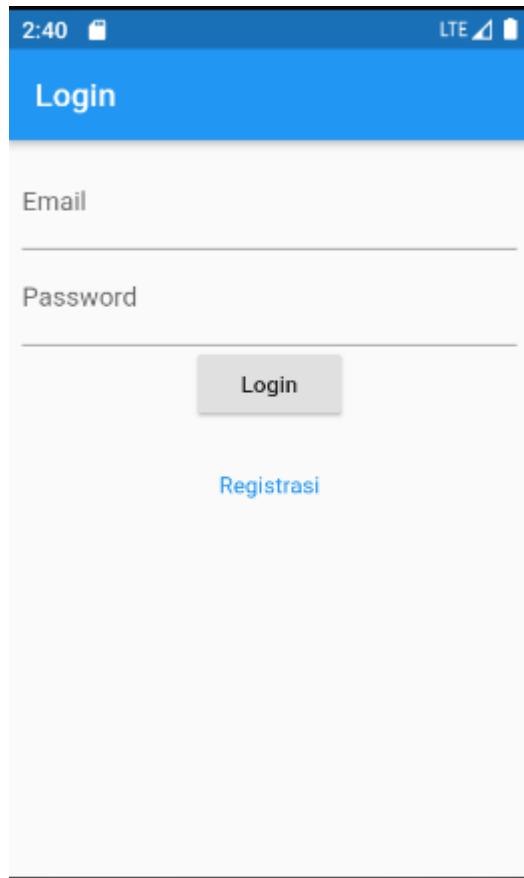
```

```
94.        style: TextStyle(color: Colors.blue),
95.        ),
96.        onTap: () {
97.            Navigator.push(context,
98.                MaterialPageRoute(builder: (context) => const RegistrasiPage()));
99.        },
100.       ),
101.      );
102.     }
103. }
```

Untuk mencobanya modifikasi file `main.dart` dimana pada bagian `home` akan memanggil `LoginPage()`

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/login_page.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: 'Toko Kita',
15.      debugShowCheckedModeBanner: false,
16.      home: LoginPage(),
17.    );
18.  }
19. }
```

Pada saat dijalankan akan terdapat link untuk membuka halaman registrasi pada bagian bawah form



Form Produk

Form produk yang akan kita buat berikut ini memiliki 2 fungsi yaitu untuk menambah data produk dan mengubah data produk

Buat sebuah file dengan nama **produk_form.dart** pada folder **ui** dengan kode berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/model/produk.dart';
3.
4. class ProdukForm extends StatefulWidget {
5.   Produk? produk;
6.
7.   ProdukForm({Key? key, this.produk}) : super(key: key);
8.
9.   @override
10.  _ProdukFormState createState() => _ProdukFormState();
11. }
12.
13. class _ProdukFormState extends State<ProdukForm> {
14.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
15.   bool _isLoading = false;
16.   String judul = "TAMBAH PRODUK";
17.   String tombolSubmit = "SIMPAN";
18.
19.   final TextEditingController _kodeProdukTextboxController = TextEditingController();
20.   final TextEditingController _namaProdukTextboxController = TextEditingController();
21.   final TextEditingController _hargaProdukTextboxController = TextEditingController();
22.
23.   @override
24.   void initState() {
```

```

25.     super.initState();
26.     isUpdate();
27.   }
28.
29.   isUpdate() {
30.     if (widget.produk != null) {
31.       setState(() {
32.         judul = "UBAH PRODUK";
33.         tombolSubmit = "UBAH";
34.         _kodeProdukTextboxController.text = widget.produk!.kodeProduk!;
35.         _namaProdukTextboxController.text = widget.produk!.namaProduk!;
36.         _hargaProdukTextboxController.text =
37.           widget.produk!.hargaProduk.toString();
38.       });
39.     } else {
40.       judul = "TAMBAH PRODUK";
41.       tombolSubmit = "SIMPAN";
42.     }
43.   }
44.
45.   @override
46.   Widget build(BuildContext context) {
47.     return Scaffold(
48.       appBar: AppBar(title: Text(judul)),
49.       body: SingleChildScrollView(
50.         child: Padding(
51.           padding: const EdgeInsets.all(8.0),
52.           child: Form(
53.             key: _formKey,
54.             child: Column(
55.               children: [
56.                 _kodeProdukTextField(),
57.                 _namaProdukTextField(),
58.                 _hargaProdukTextField(),
59.                 _buttonSubmit()
60.               ],
61.             ),
62.           ),
63.         ),
64.       ),
65.     );
66.   }
67.
68.   //Membuat Textbox Kode Produk
69.   Widget _kodeProdukTextField() {
70.     return TextFormField(
71.       decoration: const InputDecoration(labelText: "Kode Produk"),
72.       keyboardType: TextInputType.text,
73.       controller: _kodeProdukTextboxController,
74.       validator: (value) {
75.         if (value!.isEmpty) {
76.           return "Kode Produk harus diisi";
77.         }
78.         return null;
79.       },
80.     );
81.   }
82.
83.   //Membuat Textbox Nama Produk
84.   Widget _namaProdukTextField() {
85.     return TextFormField(
86.       decoration: const InputDecoration(labelText: "Nama Produk"),
87.       keyboardType: TextInputType.text,
88.       controller: _namaProdukTextboxController,
89.       validator: (value) {

```

```

90.         if (value!.isEmpty) {
91.             return "Nama Produk harus diisi";
92.         }
93.         return null;
94.     },
95. );
96. }
97.
98. //Membuat Textbox Harga Produk
99. Widget _hargaProdukTextField() {
100.     return TextFormField(
101.         decoration: const InputDecoration(labelText: "Harga"),
102.         keyboardType: TextInputType.number,
103.         controller: _hargaProdukTextboxController,
104.         validator: (value) {
105.             if (value!.isEmpty) {
106.                 return "Harga harus diisi";
107.             }
108.             return null;
109.         },
110.     );
111. }
112.
113. //Membuat Tombol Simpan/Ubah
114. Widget _buttonSubmit() {
115.     return OutlinedButton(
116.         child: Text(tombolSubmit),
117.         onPressed: () {
118.             var validate = _formKey.currentState!.validate();
119.         });
120.     }
121. }

```

Detail Produk

Buat sebuah file dengan nama `produk_detail.dart` pada folder `ui` dengan kode berikut

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/model/produk.dart';
3. import 'package:tokokita/ui/produk_form.dart';
4.
5. class ProdukDetail extends StatefulWidget {
6.     Produk? produk;
7.
8.     ProdukDetail({Key? key, this.produk}) : super(key: key);
9.
10.    @override
11.    _ProdukDetailState createState() => _ProdukDetailState();
12. }
13.
14. class _ProdukDetailState extends State<ProdukDetail> {
15.     @override
16.     Widget build(BuildContext context) {
17.         return Scaffold(
18.             appBar: AppBar(
19.                 title: const Text('Detail Produk'),
20.             ),
21.             body: Center(
22.                 child: Column(
23.                     children: [
24.                         Text(

```

```

25.             "Kode : ${widget.produk!.kodeProduk}",
26.             style: const TextStyle(fontSize: 20.0),
27.         ),
28.         Text(
29.             "Nama : ${widget.produk!.namaProduk}",
30.             style: const TextStyle(fontSize: 18.0),
31.         ),
32.         Text(
33.             "Harga : Rp. ${widget.produk!.hargaProduk.toString()}",
34.             style: const TextStyle(fontSize: 18.0),
35.         ),
36.         _tombolHapusEdit()
37.     ],
38. ),
39. ),
40. );
41. }
42.
43. Widget _tombolHapusEdit() {
44.     return Row(
45.         mainAxisAlignment: MainAxisAlignment.min,
46.         children: [
47.             //Tombol Edit
48.             OutlinedButton(
49.                 child: const Text("EDIT"),
50.                 onPressed: () {
51.                     Navigator.push(
52.                         context,
53.                         MaterialPageRoute(
54.                             builder: (context) => ProdukForm(
55.                                 produk: widget.produk!,
56.                             )));
57.                 },
58.             //Tombol Hapus
59.             OutlinedButton(
60.                 child: const Text("DELETE"), onPressed: () => confirmHapus(),
61.             ],
62.         );
63.     }
64.
65.     void confirmHapus() {
66.         AlertDialog alertDialog = AlertDialog(
67.             content: const Text("Yakin ingin menghapus data ini?"),
68.             actions: [
69.                 //tombol hapus
70.                 OutlinedButton(
71.                     child: const Text("Ya"),
72.                     onPressed: () {}),
73.                 ),
74.                 //tombol batal
75.                 OutlinedButton(
76.                     child: const Text("Batal"),
77.                     onPressed: () => Navigator.pop(context),
78.                 )
79.             ],
80.         );
81.
82.         showDialog(builder: (context) => alertDialog, context: context);
83.     }
84. }
```

Tampil List Produk

Buat sebuah file dengan nama **produk_page.dart** pada folder **ui** dengan kode berikut

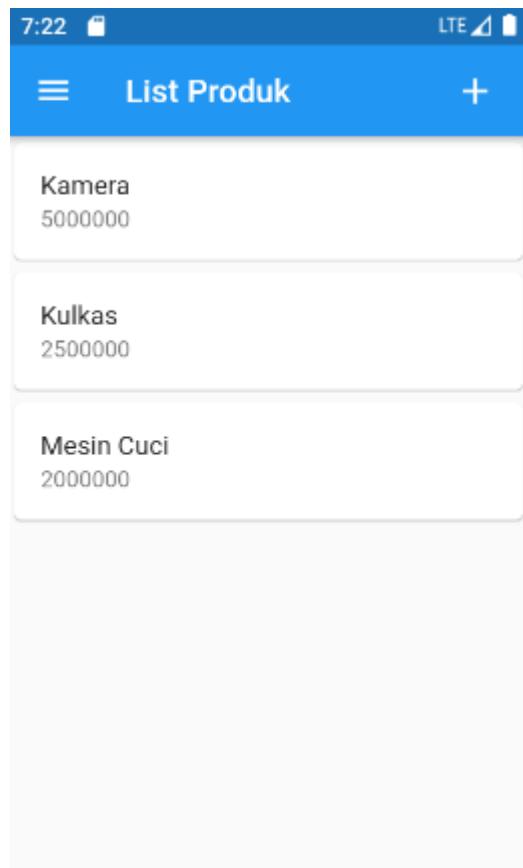
```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/model/produk.dart';
3. import 'package:tokokita/ui/produk_detail.dart';
4. import 'package:tokokita/ui/produk_form.dart';
5.
6. class ProdukPage extends StatefulWidget {
7.   const ProdukPage({Key? key}) : super(key: key);
8.
9.   @override
10.  _ProdukPageState createState() => _ProdukPageState();
11. }
12.
13. class _ProdukPageState extends State<ProdukPage> {
14.   @override
15.   Widget build(BuildContext context) {
16.     return Scaffold(
17.       appBar: AppBar(
18.         title: const Text('List Produk'),
19.         actions: [
20.           Padding(
21.             padding: const EdgeInsets.only(right: 20.0),
22.             child: GestureDetector(
23.               child: const Icon(Icons.add, size: 26.0),
24.               onTap: () async {
25.                 Navigator.push(context,
26.                   MaterialPageRoute(builder: (context) => ProdukForm()));
27.               },
28.             )),
29.           ],
30.         ),
31.       drawer: Drawer(
32.         child: ListView(
33.           children: [
34.             ListTile(
35.               title: const Text('Logout'),
36.               trailing: const Icon(Icons.logout),
37.               onTap: () async {},
38.             )
39.           ],
40.         ),
41.       ),
42.       body: ListView(
43.         children: [
44.           ItemProduk(
45.             produk: Produk(
46.               id: 1,
47.               kodeProduk: 'A001',
48.               namaProduk: 'Kamera',
49.               hargaProduk: 5000000),
50.           ItemProduk(
51.             produk: Produk(
52.               id: 2,
53.               kodeProduk: 'A002',
54.               namaProduk: 'Kulkas',
55.               hargaProduk: 2500000),
56.           ItemProduk(
57.             produk: Produk(
58.               id: 3,
59.               kodeProduk: 'A003',
60.               namaProduk: 'Mesin Cuci',
```

```
61.                 hargaProduk: 2000000)),
62.             ],
63.         )));
64.     }
65. }
66.
67. class ItemProduk extends StatelessWidget {
68.     final Produk produk;
69.
70.     const ItemProduk({Key? key, required this.produk}) : super(key: key);
71.
72.     @override
73.     Widget build(BuildContext context) {
74.         return GestureDetector(
75.             onTap: () {
76.                 Navigator.push(
77.                     context,
78.                     MaterialPageRoute(
79.                         builder: (context) => ProdukDetail(
80.                             produk: produk,
81.                         )));
82.             },
83.             child: Card(
84.                 child: ListTile(
85.                     title: Text(produk.namaProduk!),
86.                     subtitle: Text(produk.hargaProduk.toString()),
87.                     ),
88.                 ),
89.             );
90.     }
91. }
```

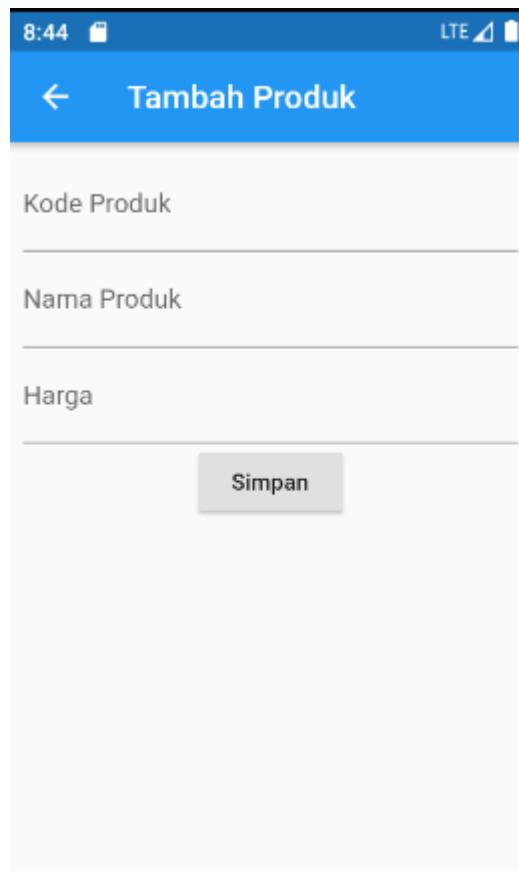
Untuk mencobanya modifikasi file **main.dart** menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/produk_page.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: 'Toko Kita',
15.            debugShowCheckedModeBanner: false,
16.            home: ProdukPage(),
17.        );
18.    }
19. }
```

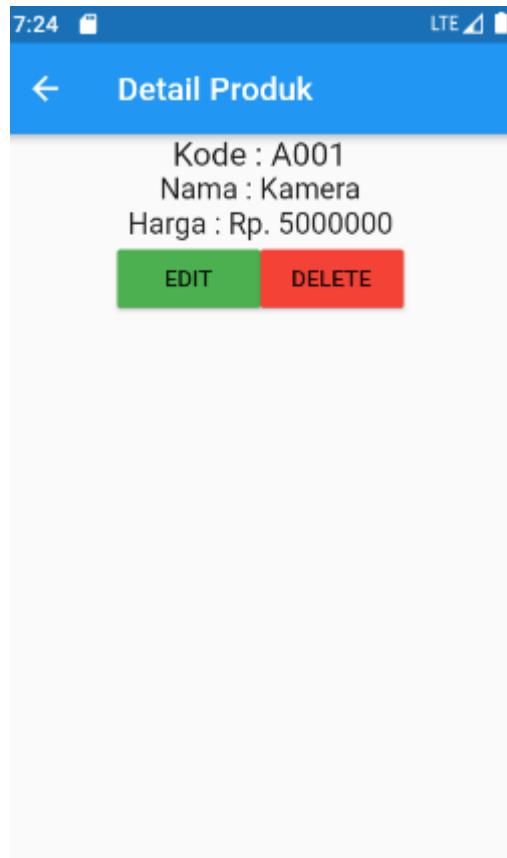
Maka tampilannya akan menjadi seperti berikut



Pada saat tombol tambah diklik maka akan muncul form produk seperti berikut



Jika salah satu data produk diklik maka akan muncul detail produk



Ketika tombol EDIT diklik maka akan muncul form produk untuk mengubah data produk



Pada materi selanjutnya akan ada modifikasi pada tampil produk agar dapat menampilkan data dari Rest API serta modifikasi pada Form Produk sehingga dapat berfungsi untuk menyimpan ataupun mengubah data pada Rest API

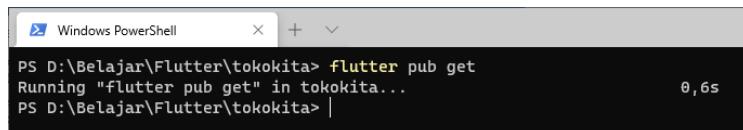
Membuat Helper Modul

Menambahkan dependencies

Dalam pembuatan aplikasi ini dibutuhkan depedensi untuk menerima dan mengirim request ke Rest API (**http**) dan depedensi untuk menyimpan token (**shared_preferences**). 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.   http: ^0.13.4  
38.   shared_preferences: ^2.0.11
```

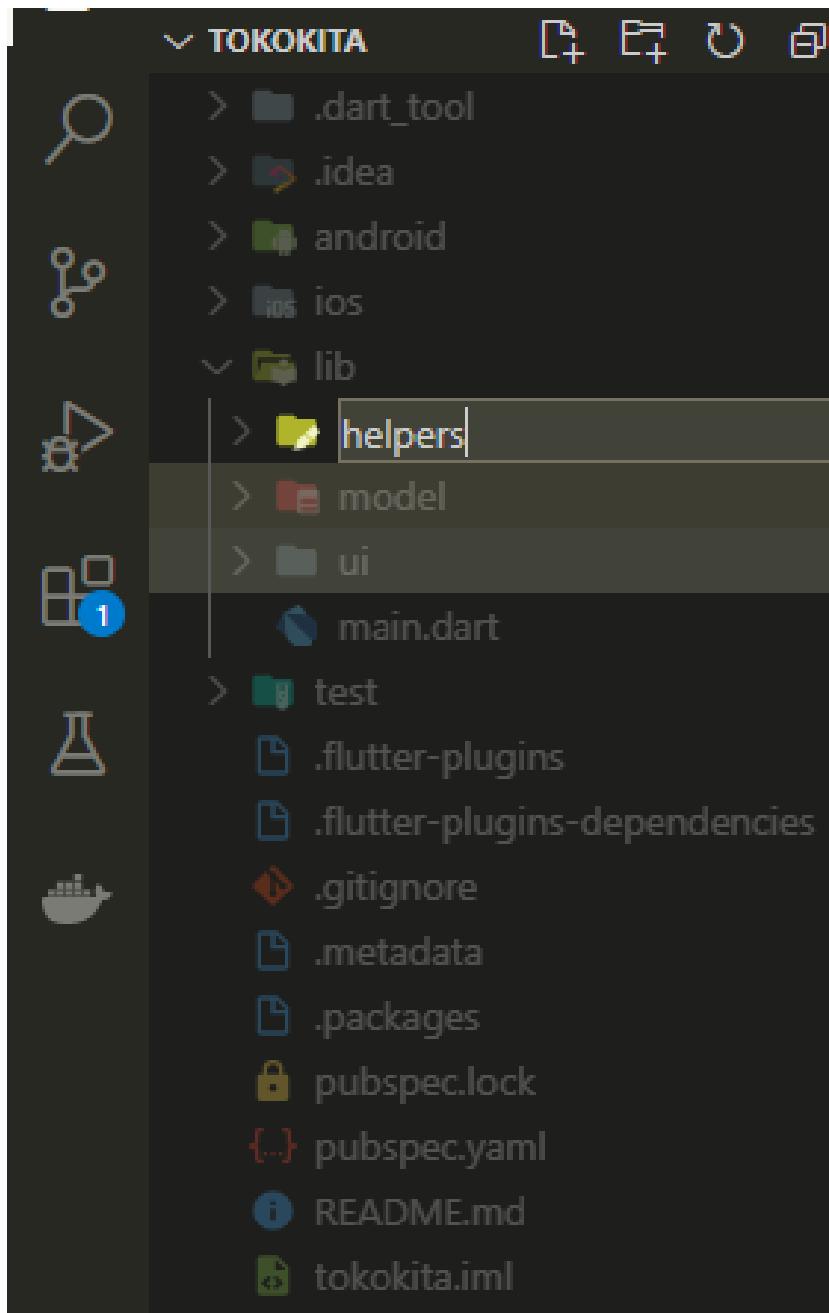
Untuk mengunduh depedencies atau package yang telah ditambahkan, buka **CommandPrompt** kemudian masuk ke folder projek dan ketikkan **flutter pub get** kemudian tekan Enter



```
PS D:\Belajar\Flutter\tokokita> flutter pub get  
Running "flutter pub get" in tokokita...  
PS D:\Belajar\Flutter\tokokita> | 0,6s
```

Membuat Class Token

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. class UserInfo {
4.   Future setToken(String value) async {
5.     final SharedPreferences pref = await SharedPreferences.getInstance();
6.     return pref.setString("token", value);
7.   }
8.
9.   Future<String?> getToken() async {
10.   final SharedPreferences pref = await SharedPreferences.getInstance();
```

```

11.     return pref.getString("token");
12.   }
13.
14.   Future setUserID(int value) async {
15.     final SharedPreferences pref = await SharedPreferences.getInstance();
16.     return pref.setInt("userID", value);
17.   }
18.
19.   Future<int?> getUserID() async {
20.     final SharedPreferences pref = await SharedPreferences.getInstance();
21.     return pref.getInt("userID");
22.   }
23.
24.   Future logout() async {
25.     final SharedPreferences pref = await SharedPreferences.getInstance();
26.     pref.clear();
27.   }
28. }
```

Http request

Membuat Modul Error Handling

Buat sebuah file pada folder `helpers` dengan nama `app_exception.dart` kemudian ketikkan kode berikut

```

1. class AppException implements Exception {
2.   final _message;
3.   final _prefix;
4.
5.   AppException([this._message, this._prefix]);
6.
7.   @override
8.   String toString() {
9.     return "${_prefix}${_message}";
10.  }
11. }
12.
13. class FetchDataException extends AppException {
14.   FetchDataException([String? message])
15.     : super(message, "Error During Communication: ");
16. }
17.
18. class BadRequestException extends AppException {
19.   BadRequestException([message]) : super(message, "Invalid Request: ");
20. }
21.
22. class UnauthorisedException extends AppException {
23.   UnauthorisedException([message]) : super(message, "Unauthorised: ");
24. }
25.
26. class UnprocessableEntityException extends AppException {
27.   UnprocessableEntityException([message])
28.     : super(message, "Unprocessable Entity: ");
29. }
30.
31. class InvalidInputException extends AppException {
32.   InvalidInputException([String? message]) : super(message, "Invalid Input: ");
33. }
```

Pada file ini berfungsi sebagai penanganan jika terjadi error saat melakukan permintaan atau pengiriman ke Rest API

Membuat Modul Http Request

Agar dapat mengirim atau menerima permintaan ke Rest API, dibuat sebuah fungsi baik itu method POST, GET dan DELETE.

Buat sebuah file di dalam folder **helpers** dengan nama **api.dart** dan masukkan kode berikut

```
1. import 'dart:io';
2.
3. import 'package:http/http.dart' as http;
4. import 'package:tokokita/helpers/user_info.dart';
5. import 'app_exception.dart';
6.
7. class Api {
8.   Future<dynamic> post(dynamic url, dynamic data) async {
9.     var token = await UserInfo().getToken();
10.    var responseJson;
11.    try {
12.      final response = await http.post(Uri.parse(url),
13.          body: data,
14.          headers: {HttpHeaders.authorizationHeader: "Bearer $token"});
15.      responseJson = _returnResponse(response);
16.    } on SocketException {
17.      throw FetchDataException('No Internet connection');
18.    }
19.    return responseJson;
20.  }
21.
22.   Future<dynamic> get(dynamic url) async {
23.     var token = await UserInfo().getToken();
24.     var responseJson;
25.     try {
26.       final response = await http.get(url,
27.           headers: {HttpHeaders.authorizationHeader: "Bearer $token"});
28.       responseJson = _returnResponse(response);
29.     } on SocketException {
30.       throw FetchDataException('No Internet connection');
31.     }
32.     return responseJson;
33.   }
34.
35.   Future<dynamic> delete(dynamic url) async {
36.     var token = await UserInfo().getToken();
37.     var responseJson;
38.     try {
39.       final response = await http.delete(url,
40.           headers: {HttpHeaders.authorizationHeader: "Bearer $token"});
41.       responseJson = _returnResponse(response);
42.     } on SocketException {
43.       throw FetchDataException('No Internet connection');
44.     }
45.     return responseJson;
46.   }
47.
48.   dynamic _returnResponse(http.Response response) {
49.     switch (response.statusCode) {
```

```

50.    case 200:
51.        return response;
52.    case 400:
53.        throw BadRequestException(response.body.toString());
54.    case 401:
55.        case 403:
56.            throw UnauthorisedException(response.body.toString());
57.    case 422:
58.        throw InvalidInputException(response.body.toString());
59.    case 500:
60.    default:
61.        throw FetchDataException(
62.            'Error occured while Communication with Server with StatusCode :
63.             ${response.statusCode}');
64.    }
65. }

```

Membuat List API url

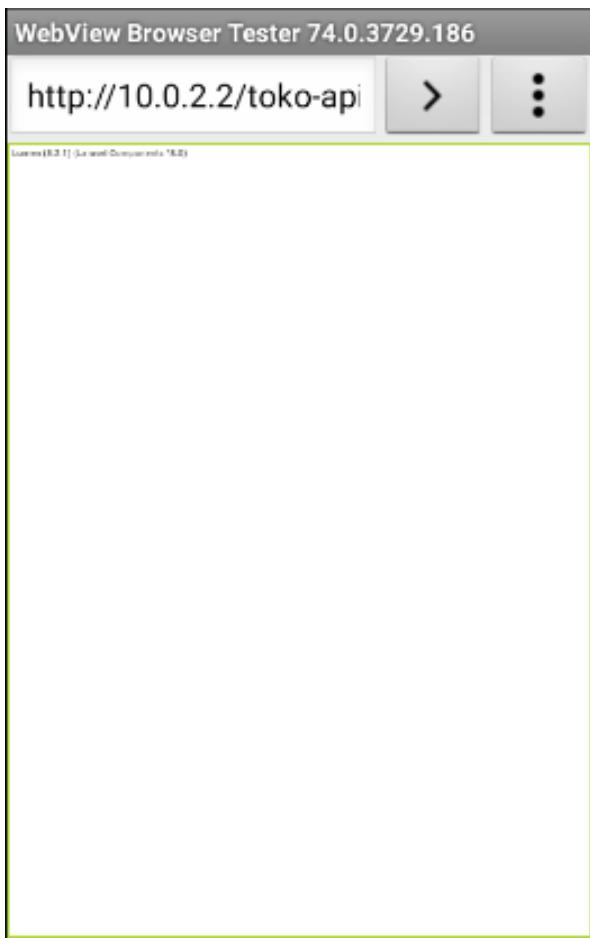
Buat sebuah file di dalam folder **helpers** dengan nama **api_url.dart**, dimana fungsi dari file ini adalah menyimpan alamat-alamat API yang telah dibuat sebelumnya (Endpoint dari API Spec)

```

1. class ApiUrl {
2.     static const String baseUrl = 'http://10.0.2.2/toko-api/public';
3.
4.     static const String registrasi = baseUrl + '/registrasi';
5.     static const String login = baseUrl + '/login';
6.     static const String listProduk = baseUrl + '/produk';
7.     static const String createProduk = baseUrl + '/produk';
8.
9.     static String updateProduk(int id) {
10.         return baseUrl + '/produk/' + id.toString() + '/update';
11.     }
12.
13.     static String showProduk(int id) {
14.         return baseUrl + '/produk/' + id.toString();
15.     }
16.
17.     static String deleteProduk(int id) {
18.         return baseUrl + '/produk/' + id.toString();
19.     }
20. }

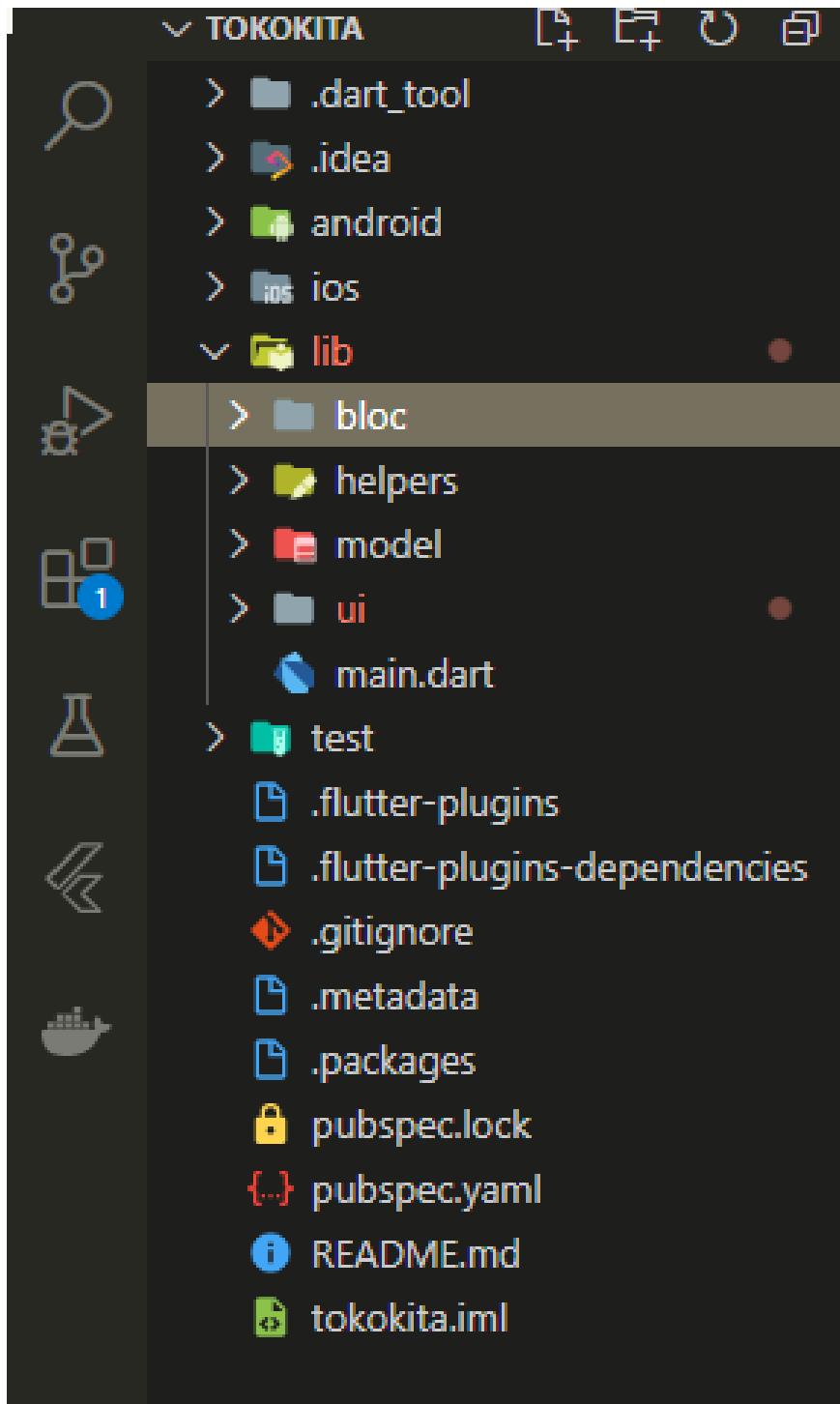
```

Pada baris kedua (baseUrl) merupakan alamat IP dari Rest API, untuk memeriksa apakah terhubung dengan emulator atau tidak, dapat dilakukan dengan memasukkan alamat tersebut pada browser yang ada pada emulator atau handphone android yang terkoneksi dengan laptop



Membuat Bloc

Buat sebuah folder bernama **bloc**. Di dalam folder ini berisis file-file yang berfungsi sebagai controller baik itu untuk melakukan proses login, registrasi dan lain-lain.



Registrasi

Buat sebuah file dengan nama **registrasi_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'dart:convert';
2.
3. import 'package:tokokita/helpers/api.dart';
4. import 'package:tokokita/helpers/api_url.dart';
5. import 'package:tokokita/model/registrasi.dart';
```

```
6.
7. class RegistrasiBloc {
8.     static Future<Registrasi> registrasi(
9.         {String? nama, String? email, String? password}) async {
10.    String apiUrl = ApiUrl.registrasi;
11.
12.    var body = {"nama": nama, "email": email, "password": password};
13.
14.    var response = await Api().post(apiUrl, body);
15.    var jsonObj = json.decode(response.body);
16.    return Registrasi.fromJson(jsonObj);
17. }
18. }
```

Login

Buat sebuah file dengan nama **login_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'dart:convert';
2.
3. import 'package:tokokita/helpers/api.dart';
4. import 'package:tokokita/helpers/api_url.dart';
5. import 'package:tokokita/model/login.dart';
6.
7. class LoginBloc {
8.     static Future<Login> login({String? email, String? password}) async {
9.        String apiUrl = ApiUrl.login;
10.       var body = {"email": email, "password": password};
11.       var response = await Api().post(apiUrl, body);
12.       var jsonObj = json.decode(response.body);
13.       return Login.fromJson(jsonObj);
14.    }
15. }
```

Logout

Buat sebuah file dengan nama **logout_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'package:tokokita/helpers/user_info.dart';
2.
3. class LogoutBloc{
4.     static Future logout() async {
5.        await UserInfo().logout();
6.    }
7. }
```

Produk

Pada bagian ini akan dibuat beberapa fungsi untuk mengambil, mengubah dan menghapus data produk dari Rest API. Buat sebuah file dengan nama **produk_bloc.dart** pada folder **bloc** kemudian masukkan kode berikut

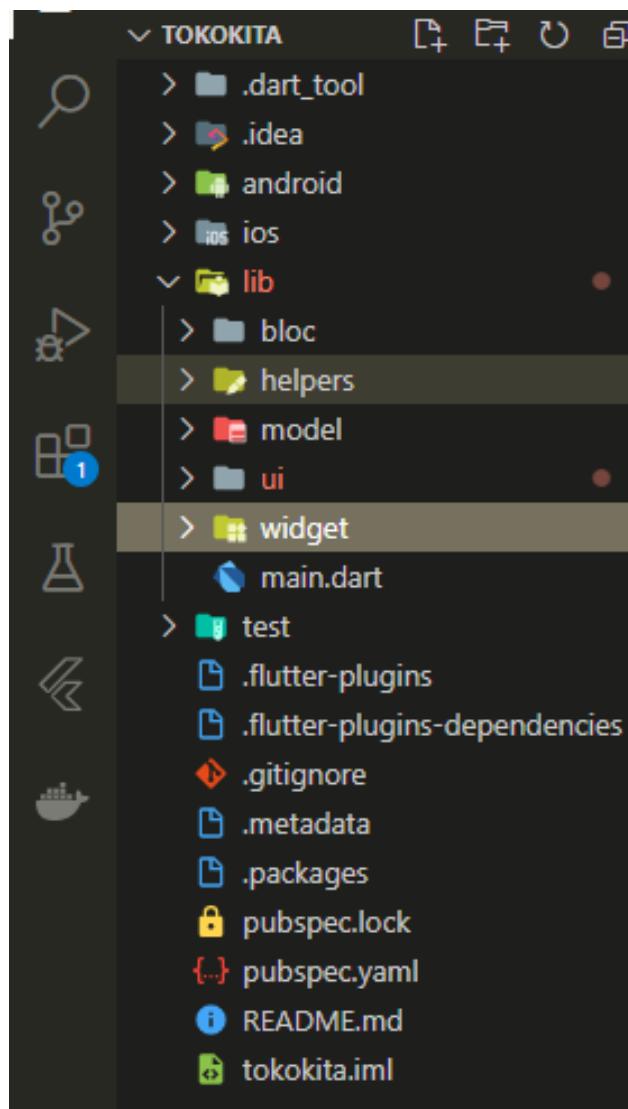
```

1. import 'dart:convert';
2.
3. import 'package:tokokita/helpers/api.dart';
4. import 'package:tokokita/helpers/api_url.dart';
5. import 'package:tokokita/model/produk.dart';
6.
7. class ProdukBloc {
8.   static Future<List<Produk>> getProduks() async {
9.     String apiUrl = ApiUrl.listProduk;
10.    var response = await Api().get(apiUrl);
11.    var jsonObj = json.decode(response.body);
12.    List<dynamic> listProduk = (jsonObj as Map<String, dynamic>)['data'];
13.    List<Produk> produks = [];
14.    for (int i = 0; i < listProduk.length; i++) {
15.      produks.add(Produk.fromJson(listProduk[i]));
16.    }
17.    return produks;
18.  }
19.
20. static Future addProduk({Produk? produk}) async {
21.   String apiUrl = ApiUrl.createProduk;
22.
23.   var body = {
24.     "kode_produk": produk!.kodeProduk,
25.     "nama_produk": produk.namaProduk,
26.     "harga": produk.hargaProduk.toString()
27.   };
28.
29.   var response = await Api().post(apiUrl, body);
30.   var jsonObj = json.decode(response.body);
31.   return jsonObj['status'];
32. }
33.
34. static Future<bool> updateProduk({required Produk produk}) async {
35.   String apiUrl = ApiUrl.updateProduk(produk.id!);
36.
37.   var body = {
38.     "kode_produk": produk.kodeProduk,
39.     "nama_produk": produk.namaProduk,
40.     "harga": produk.hargaProduk.toString()
41.   };
42.   print("Body : $body");
43.   var response = await Api().post(apiUrl, body);
44.   var jsonObj = json.decode(response.body);
45.   return jsonObj['data'];
46. }
47.
48. static Future<bool> deleteProduk({int? id}) async {
49.   String apiUrl = ApiUrl.deleteProduk(id!);
50.
51.   var response = await Api().delete(apiUrl);
52.   var jsonObj = json.decode(response.body);
53.   return (jsonObj as Map<String, dynamic>)['data'];
54. }
55. }
```

Menyatukan Fungsionalitas

Membuat Common Dialog Widget

Pada bagian ini akan dibuat dua buah dialog yang nantinya akan digunakan pada tampilan. Buat sebuah folder dengan nama **widget**



Kemudian buat sebuah file dengan nama **success_dialog.dart** dengan kode

```
1. import 'package:flutter/material.dart';
2.
3. class Consts {
4.   Consts._();
5.
6.   static const double padding = 16.0;
7.   static const double avatarRadius = 66.0;
8. }
9.
10. class SuccessDialog extends StatelessWidget {
11.   final String? description;
12.   final VoidCallback? onClick;
13.
14.   const SuccessDialog({Key? key, this.description, this.onClick})
```

```

15.      : super(key: key);
16.
17.  @override
18.  Widget build(BuildContext context) {
19.    return Dialog(
20.      shape: RoundedRectangleBorder(
21.          borderRadius: BorderRadius.circular(Consts.padding)),
22.      elevation: 0.0,
23.      backgroundColor: Colors.transparent,
24.      child: dialogContent(context),
25.    );
26.  }
27.
28.  dialogContent(BuildContext context) {
29.    return Container(
30.      padding: const EdgeInsets.only(
31.        top: Consts.padding,
32.        bottom: Consts.padding,
33.        left: Consts.padding,
34.        right: Consts.padding,
35.      ),
36.      margin: const EdgeInsets.only(top: Consts.avatarRadius),
37.      decoration: BoxDecoration(
38.        color: Colors.white,
39.        shape: BoxShape.rectangle,
40.        borderRadius: BorderRadius.circular(Consts.padding),
41.        boxShadow: const [
42.          BoxShadow(
43.            color: Colors.black26,
44.            blurRadius: 10.0,
45.            offset: Offset(0.0, 10.0),
46.          ),
47.        ],
48.      ),
49.      child: Column(
50.        mainAxisSize: MainAxisSize.min,
51.        children: [
52.          const Text(
53.            "SUKSES",
54.            style: TextStyle(
55.              fontSize: 24.0,
56.              fontWeight: FontWeight.w700,
57.              color: Colors.green),
58.          ),
59.          const SizedBox(height: 16.0),
60.          Text(
61.            description!,
62.            textAlign: TextAlign.center,
63.            style: const TextStyle(
64.              fontSize: 16.0,
65.            ),
66.          ),
67.          const SizedBox(height: 24.0),
68.          Align(
69.            alignment: Alignment.bottomRight,
70.            child: OutlinedButton(
71.              onPressed: () {
72.                Navigator.of(context).pop(); // To close the dialog
73.                okClick!();
74.              },
75.              child: const Text("OK"),
76.            ),
77.          ),
78.        ],
79.      ),

```

```
80.    );
81. }
82. }
```

Kemudian buat file dengan nama **warning_dialog.dart** dengan kode

```
1. import 'package:flutter/material.dart';
2.
3. class Consts {
4.   Consts._();
5.
6.   static const double padding = 16.0;
7.   static const double avatarRadius = 66.0;
8. }
9.
10. class WarningDialog extends StatelessWidget {
11.   final String? description;
12.   final VoidCallback? okClick;
13.
14.   const WarningDialog({Key? key, this.description, this.okClick})
15.     : super(key: key);
16.
17.   @override
18.   Widget build(BuildContext context) {
19.     return Dialog(
20.       shape: RoundedRectangleBorder(
21.           borderRadius: BorderRadius.circular(Consts.padding)),
22.       elevation: 0.0,
23.       backgroundColor: Colors.transparent,
24.       child: dialogContent(context),
25.     );
26.   }
27.
28.   dialogContent(BuildContext context) {
29.     return Container(
30.       padding: const EdgeInsets.only(
31.         top: Consts.padding,
32.         bottom: Consts.padding,
33.         left: Consts.padding,
34.         right: Consts.padding,
35.       ),
36.       margin: const EdgeInsets.only(top: Consts.avatarRadius),
37.       decoration: BoxDecoration(
38.         color: Colors.white,
39.         shape: BoxShape.rectangle,
40.         borderRadius: BorderRadius.circular(Consts.padding),
41.         boxShadow: const [
42.           BoxShadow(
43.             color: Colors.black26,
44.             blurRadius: 10.0,
45.             offset: Offset(0.0, 10.0),
46.           ),
47.         ],
48.       ),
49.       child: Column(
50.         mainAxisSize: MainAxisSize.min,
51.         children: [
52.           const Text(
53.             "GAGAL",
54.             style: TextStyle(
55.               fontSize: 24.0, fontWeight: FontWeight.w700, color: Colors.red),
56.           ),
57.           const SizedBox(height: 16.0),
58.           Text(

```

```

59.           description!,
60.           textAlign: TextAlign.center,
61.           style: const TextStyle(
62.               fontSize: 16.0,
63.           ),
64.           ),
65.           const SizedBox(height: 24.0),
66.           Align(
67.               alignment: Alignment.bottomRight,
68.               child: ElevatedButton(
69.                   onPressed: () {
70.                       Navigator.of(context).pop(); // To close the dialog
71.                       okClick!();
72.                   },
73.                   child: const Text("OK"),
74.               ),
75.           )
76.       ],
77.   ),
78. );
79. }
80. }
```

Modifikasi main.dart

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 list produk

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/helpers/user_info.dart';
3. import 'package:tokokita/ui/login_page.dart';
4. import 'package:tokokita/ui/produk_page.dart';
5.
6. void main() {
7.   runApp(const MyApp());
8. }
9.
10. class MyApp extends StatefulWidget {
11.   const MyApp({Key? key}) : super(key: key);
12.
13.   @override
14.   _MyAppState createState() => _MyAppState();
15. }
16.
17. class _MyAppState extends State<MyApp> {
18.   Widget page = const CircularProgressIndicator();
19.
20.   @override
21.   void initState() {
22.     super.initState();
23.     isLoggedIn();
24.   }
25.
26.   void isLoggedIn() async {
27.     var token = await UserInfo().getToken();
28.     if (token != null) {
29.       setState(() {
30.         page = const ProdukPage();
31.       });
32.     } else {
33.       setState(() {
```

```

34.         page = const LoginPage();
35.     });
36.   }
37. }
38.
39. @override
40. Widget build(BuildContext context) {
41.   return MaterialApp(
42.     title: 'Toko Kita',
43.     debugShowCheckedModeBanner: false,
44.     home: page,
45.   );
46. }
47.

```

Modifikasi registrasi_page.dart

Buka file **registrasi_page.dart** pada folder **ui** kemudian modifikasi fungsi `_buttonRegistrasi` dan tambahkan fungsi dengan nama `_submit` seperti dibawah

```

119. //Membuat Tombol Registrasi
120. Widget _buttonRegistrasi() {
121.   return ElevatedButton(
122.     child: const Text("Registrasi"),
123.     onPressed: () {
124.       var validate = _formKey.currentState!.validate();
125.       if (validate) {
126.         if (!isLoading) _submit();
127.       }
128.     });
129. }
130.
131. void _submit() {
132.   _formKey.currentState!.save();
133.   setState(() {
134.     isLoading = true;
135.   });
136.   RegistrasiBloc.registrasi(
137.     nama: _namaTextboxController.text,
138.     email: _emailTextboxController.text,
139.     password: _passwordTextboxController.text)
140.     .then((value) {
141.       showDialog(
142.         context: context,
143.         barrierDismissible: false,
144.         builder: (BuildContext context) => SuccessDialog(
145.             description: "Registrasi berhasil, silahkan login",
146.             onClick: () {
147.               Navigator.pop(context);
148.             },
149.           ));
150.     }, onError: (error) {
151.       showDialog(
152.         context: context,
153.         barrierDismissible: false,
154.         builder: (BuildContext context) => const WarningDialog(
155.             description: "Registrasi gagal, silahkan coba lagi",
156.           ));
157.     });
158.     setState(() {
159.       isLoading = false;
160.     });
161.   }

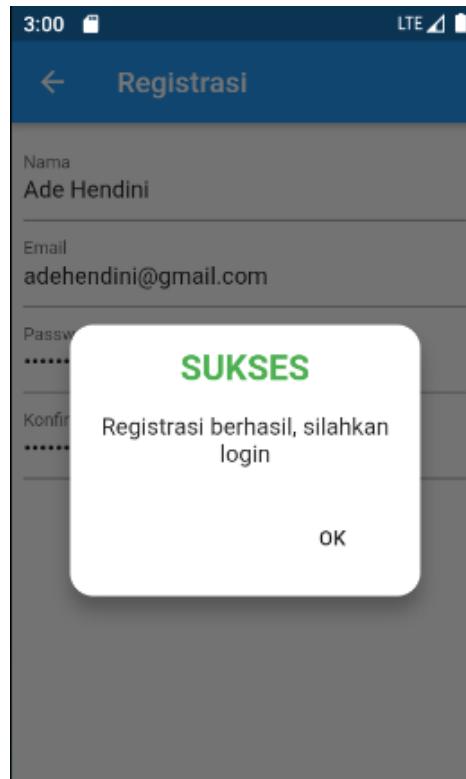
```

Sehingga keseluruhan kode pada `registrasi_page.dart` menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/registrasi_bloc.dart';
3. import 'package:tokokita/widget/success_dialog.dart';
4. import 'package:tokokita/widget/warning_dialog.dart';
5.
6. class RegistrasiPage extends StatefulWidget {
7.   const RegistrasiPage({Key? key}) : super(key: key);
8.
9.   @override
10.  _RegistrasiPageState createState() => _RegistrasiPageState();
11. }
12.
13. class _RegistrasiPageState extends State<RegistrasiPage> {
14.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
15.   bool _isLoading = false;
16.
17.   final TextEditingController _namaTextboxController = TextEditingController();
18.   final TextEditingController _emailTextboxController = TextEditingController();
19.   final TextEditingController _passwordTextboxController = TextEditingController();
20.
21.   @override
22.   Widget build(BuildContext context) {
23.     return Scaffold(
24.       appBar: AppBar(
25.         title: const Text("Registrasi"),
26.       ),
27.       body: SingleChildScrollView(
28.         child: Padding(
29.           padding: const EdgeInsets.all(8.0),
30.           child: Form(
31.             key: _formKey,
32.             child: Column(
33.               mainAxisAlignment: MainAxisAlignment.center,
34.               children: [
35.                 _namaTextField(),
36.                 _emailTextField(),
37.                 _passwordTextField(),
38.                 _passwordKonfirmasiTextField(),
39.                 _buttonRegistrasi()
40.               ],
41.             ),
42.           ),
43.         ),
44.       ),
45.     );
46.   }
47.
48. //Membuat Textbox Nama
49. Widget _namaTextField() {
50.   return TextFormField(
51.     decoration: const InputDecoration(labelText: "Nama"),
52.     keyboardType: TextInputType.text,
53.     controller: _namaTextboxController,
54.     validator: (value) {
55.       if (value!.length < 3) {
56.         return "Nama harus diisi minimal 3 karakter";
57.       }
58.       return null;
59.     },
60.   );
}
```

```
61. }
62.
63. //Membuat Textbox email
64. Widget _emailTextField() {
65.     return TextFormField(
66.         decoration: const InputDecoration(labelText: "Email"),
67.         keyboardType: TextInputType.emailAddress,
68.         controller: _emailTextboxController,
69.         validator: (value) {
70.             //validasi harus diisi
71.             if (value!.isEmpty) {
72.                 return 'Email harus diisi';
73.             }
74.             //validasi email
75.             Pattern pattern =
76.                 r'^(([a-zA-Z]+([a-zA-Z\d]+){1,3})\.[a-zA-Z\d]{1,3}\.([a-zA-Z\d]{1,3})\.[a-zA-Z\d]{1,3})$';
77.             RegExp regex = RegExp(pattern.toString());
78.             if (!regex.hasMatch(value)) {
79.                 return "Email tidak valid";
80.             }
81.             return null;
82.         },
83.     );
84. }
85.
86. //Membuat Textbox password
87. Widget _passwordTextField() {
88.     return TextFormField(
89.         decoration: const InputDecoration(labelText: "Password"),
90.         keyboardType: TextInputType.text,
91.         obscureText: true,
92.         controller: _passwordTextboxController,
93.         validator: (value) {
94.             //jika karakter yang dimasukkan kurang dari 6 karakter
95.             if (value!.length < 6) {
96.                 return "Password harus diisi minimal 6 karakter";
97.             }
98.             return null;
99.         },
100.    );
101.   }
102.
103. //membuat textbox Konfirmasi Password
104. Widget _passwordKonfirmasiTextField() {
105.     return TextFormField(
106.         decoration: const InputDecoration(labelText: "Konfirmasi Password"),
107.         keyboardType: TextInputType.text,
108.         obscureText: true,
109.         validator: (value) {
110.             //jika inputan tidak sama dengan password
111.             if (value != _passwordTextboxController.text) {
112.                 return "Konfirmasi Password tidak sama";
113.             }
114.             return null;
115.         },
116.     );
117.   }
118.
119. //Membuat Tombol Registrasi
120. Widget _buttonRegistrasi() {
121.     return ElevatedButton(
122.         child: const Text("Registrasi"),
123.         onPressed: () {
```

```
124.         var validate = _formKey.currentState!.validate();
125.         if (validate) {
126.             if (!isLoading) _submit();
127.         }
128.     });
129. }
130.
131. void _submit() {
132.     _formKey.currentState!.save();
133.     setState(() {
134.         isLoading = true;
135.     });
136.     RegistrasiBloc.registrasi(
137.         nama: _namaTextboxController.text,
138.         email: _emailTextboxController.text,
139.         password: _passwordTextboxController.text)
140.     .then((value) {
141.         showDialog(
142.             context: context,
143.             barrierDismissible: false,
144.             builder: (BuildContext context) => SuccessDialog(
145.                 description: "Registrasi berhasil, silahkan login",
146.                 onClick: () {
147.                     Navigator.pop(context);
148.                 },
149.             )));
150.     }, onError: (error) {
151.         showDialog(
152.             context: context,
153.             barrierDismissible: false,
154.             builder: (BuildContext context) => const WarningDialog(
155.                 description: "Registrasi gagal, silahkan coba lagi",
156.             )));
157.     });
158.     setState(() {
159.         isLoading = false;
160.     });
161. }
162. }
```



Modifikasi `login_page.dart` (fungsi `login`)

Buka file `login_page.dart` pada folder `ui` kemudian modifikasi fungsi `_buttonLogin` dan tambahkan fungsi dengan nama `_submit` seperti dibawah

```
83. //Membuat Tombol Login
84. Widget _buttonLogin() {
85.   return ElevatedButton(
86.     child: const Text("Login"),
87.     onPressed: () {
88.       var validate = _formKey.currentState!.validate();
89.       if (validate) {
90.         if (!isLoading) _submit();
91.       }
92.     });
93. }
94.
95. void _submit() {
96.   _formKey.currentState!.save();
97.   setState(() {
98.     isLoading = true;
99.   });
100.  LoginBloc.login(
101.    email: _emailTextboxController.text,
102.    password: _passwordTextboxController.text)
103.    .then((value) async {
104.      await UserInfo().setToken(value.token.toString());
105.      await UserInfo().setUserID(int.parse(value.userID.toString()));
106.      Navigator.pushReplacement(
107.        context, MaterialPageRoute(builder: (context) => const
108.          ProdukPage())));
109.    }, onError: (error) {
110.      print(error);
111.      showDialog(
```

```

111.         context: context,
112.         barrierDismissible: false,
113.         builder: (BuildContext context) => const WarningDialog(
114.             description: "Login gagal, silahkan coba lagi",
115.         )));
116.     });
117.     setState(() {
118.         _isLoading = false;
119.     });
120. }

```

Adapun kode secara keseluruhan menjadi seperti berikut

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/login_bloc.dart';
3. import 'package:tokokita/helpers/user_info.dart';
4. import 'package:tokokita/ui/produk_page.dart';
5. import 'package:tokokita/ui/registrasi_page.dart';
6. import 'package:tokokita/widget/warning_dialog.dart';
7.
8. class LoginPage extends StatefulWidget {
9.     const LoginPage({Key? key}) : super(key: key);
10.
11.    @override
12.    _LoginPageState createState() => _LoginPageState();
13. }
14.
15. class _LoginPageState extends State<LoginPage> {
16.     final _formKey = GlobalKey<FormState>();
17.     bool _isLoading = false;
18.
19.     final _emailTextboxController = TextEditingController();
20.     final _passwordTextboxController = TextEditingController();
21.
22.    @override
23.    Widget build(BuildContext context) {
24.        return Scaffold(
25.            appBar: AppBar(
26.                title: const Text('Login'),
27.            ),
28.            body: SingleChildScrollView(
29.                child: Padding(
30.                    padding: const EdgeInsets.all(8.0),
31.                    child: Form(
32.                        key: _formKey,
33.                        child: Column(
34.                            children: [
35.                                _emailTextField(),
36.                                _passwordTextField(),
37.                                _buttonLogin(),
38.                                const SizedBox(
39.                                    height: 30,
40.                                ),
41.                                _menuRegistrasi()
42.                            ],
43.                        ),
44.                    ),
45.                ),
46.            ),
47.        );
48.    }
49.
50. //Membuat Textbox email
51. Widget _emailTextField() {

```

```

52.     return TextFormField(
53.       decoration: const InputDecoration(labelText: "Email"),
54.       keyboardType: TextInputType.emailAddress,
55.       controller: _emailTextboxController,
56.       validator: (value) {
57.         //validasi harus diisi
58.         if (value!.isEmpty) {
59.           return 'Email harus diisi';
60.         }
61.         return null;
62.       },
63.     );
64.   }
65.
66.   //Membuat Textbox password
67.   Widget _passwordTextField() {
68.     return TextFormField(
69.       decoration: const InputDecoration(labelText: "Password"),
70.       keyboardType: TextInputType.text,
71.       obscureText: true,
72.       controller: _passwordTextboxController,
73.       validator: (value) {
74.         //jika karakter yang dimasukkan kurang dari 6 karakter
75.         if (value!.isEmpty) {
76.           return "Password harus diisi";
77.         }
78.         return null;
79.       },
80.     );
81.   }
82.
83.   //Membuat Tombol Login
84.   Widget _buttonLogin() {
85.     return ElevatedButton(
86.       child: const Text("Login"),
87.       onPressed: () {
88.         var validate = _formKey.currentState!.validate();
89.         if (validate) {
90.           if (!isLoading) _submit();
91.         }
92.       });
93.   }
94.
95.   void _submit() {
96.     _formKey.currentState!.save();
97.     setState(() {
98.       isLoading = true;
99.     });
100.    LoginBloc.login(
101.      email: _emailTextboxController.text,
102.      password: _passwordTextboxController.text)
103.      .then((value) async {
104.        await UserInfo().setToken(value.token.toString());
105.        await UserInfo().setUserID(int.parse(value.userID.toString()));
106.        Navigator.pushReplacement(
107.          context, MaterialPageRoute(builder: (context) => const
ProdukPage())));
108.      }, onError: (error) {
109.        print(error);
110.        showDialog(
111.          context: context,
112.          barrierDismissible: false,
113.          builder: (BuildContext context) => const WarningDialog(
114.            description: "Login gagal, silahkan coba lagi",
115.          )));

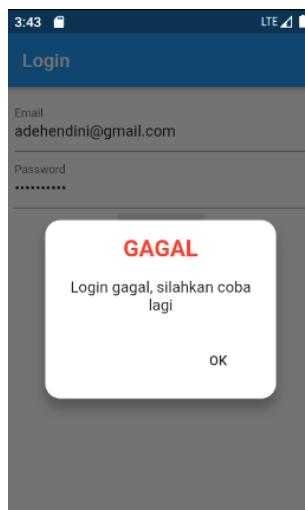
```

```

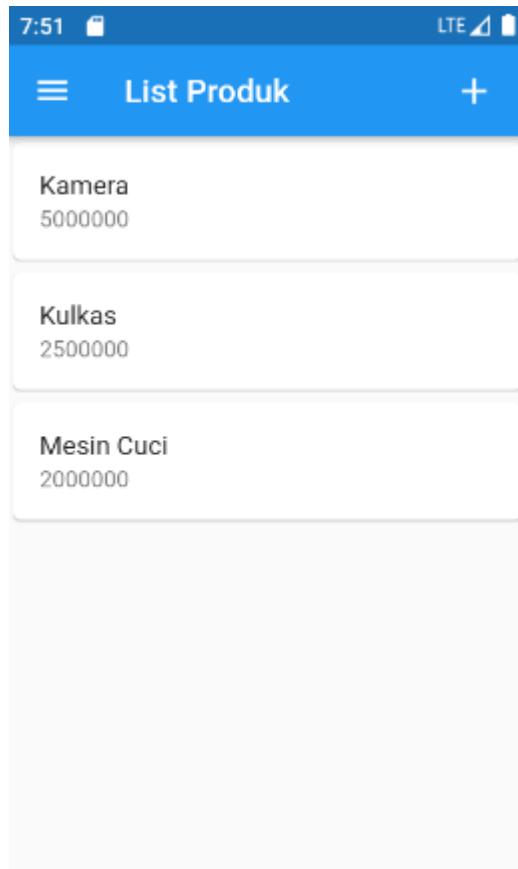
116.      });
117.      setState(() {
118.        _isLoading = false;
119.      });
120.    }
121.
122.    // Membuat menu untuk membuka halaman registrasi
123.    Widget _menuRegistrasi() {
124.      return Center(
125.        child: InkWell(
126.          child: const Text(
127.            "Registrasi",
128.            style: TextStyle(color: Colors.blue),
129.          ),
130.          onTap: () {
131.            Navigator.push(context,
132.              MaterialPageRoute(builder: (context) => const
133. RegistrasiPage()));
134.          },
135.        );
136.      }
137.    }

```

Jika Gagal akan muncul pesan seperti berikut



Jika berhasil akan menuju ke halaman `produk_page.dart`



Modifikasi produk_page.dart

Menambahkan fungsi logout pada drawer

Agar link logout dapat berfungsi, akan ditambahkan kode pada drawer logout, seperti berikut

```
33. drawer: Drawer(
34.           child: ListView(
35.             children: [
36.               ListTile(
37.                 title: const Text('Logout'),
38.                 trailing: const Icon(Icons.logout),
39.                 onTap: () async {
40.                   await LogoutBloc.logout().then((value) => {
41.                     Navigator.pushReplacement(
42.                       context,
43.                       MaterialPageRoute(
44.                         builder: (context) => LoginPage())))
45.                 });
46.               },
47.             )
48.           ],
49.         ),
50.       ),
51.     body: ListView(
```

Secara keseluruhan kode pada **produk_page.dart** menjadi seperti berikut

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

```

2. import 'package:tokokita/bloc/logout_bloc.dart';
3. import 'package:tokokita/model/produk.dart';
4. import 'package:tokokita/ui/login_page.dart';
5. import 'package:tokokita/ui/produk_detail.dart';
6. import 'package:tokokita/ui/produk_form.dart';
7.
8. class ProdukPage extends StatefulWidget {
9.   const ProdukPage({Key? key}) : super(key: key);
10.
11.  @override
12.  _ProdukPageState createState() => _ProdukPageState();
13. }
14.
15. class _ProdukPageState extends State<ProdukPage> {
16.  @override
17.  Widget build(BuildContext context) {
18.    return Scaffold(
19.      appBar: AppBar(
20.        title: const Text('List Produk'),
21.        actions: [
22.          Padding(
23.            padding: const EdgeInsets.only(right: 20.0),
24.            child: GestureDetector(
25.              child: const Icon(Icons.add, size: 26.0),
26.              onTap: () async {
27.                Navigator.push(context,
28.                  MaterialPageRoute(builder: (context) => ProdukForm()));
29.              },
30.            )));
31.      ],
32.    ),
33.    drawer: Drawer(
34.      child: ListView(
35.        children: [
36.          ListTile(
37.            title: const Text('Logout'),
38.            trailing: const Icon(Icons.logout),
39.            onTap: () async {
40.              await LogoutBloc.logout().then((value) => {
41.                Navigator.pushReplacement(
42.                  context,
43.                  MaterialPageRoute(
44.                    builder: (context) => LoginPage())));
45.              });
46.            },
47.          ),
48.        ],
49.      ),
50.    ),
51.    body: ListView(
52.      children: [
53.        ItemProduk(
54.          produk: Produk(
55.            id: 1,
56.            kodeProduk: 'A001',
57.            namaProduk: 'Kamera',
58.            hargaProduk: 5000000)),
59.        ItemProduk(
60.          produk: Produk(
61.            id: 2,
62.            kodeProduk: 'A002',
63.            namaProduk: 'Kulkas',
64.            hargaProduk: 2500000)),
65.        ItemProduk(
66.          produk: Produk(

```

```

67.             id: 3,
68.             kodeProduk: 'A003',
69.             namaProduk: 'Mesin Cuci',
70.             hargaProduk: 2000000)),
71.         ],
72.     )));
73. }
74. }
75.
76. class ItemProduk extends StatelessWidget {
77.   final Produk produk;
78.
79.   const ItemProduk({Key? key, required this.produk}) : super(key: key);
80.
81.   @override
82.   Widget build(BuildContext context) {
83.     return GestureDetector(
84.       onTap: () {
85.         Navigator.push(
86.           context,
87.           MaterialPageRoute(
88.             builder: (context) => ProdukDetail(
89.               produk: produk,
90.             )));
91.       },
92.       child: Card(
93.         child: ListTile(
94.           title: Text(produk.namaProduk!),
95.           subtitle: Text(produk.hargaProduk.toString()),
96.         ),
97.       ),
98.     );
99.   }
100. }

```

Menampilkan Data Produk dari Rest API

Pada bagian ini akan dimodifikasi file **produk_page.dart** sehingga dapat menampilkan data dari Rest API. Berikut kode keseluruhan

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/logout_bloc.dart';
3. import 'package:tokokita/bloc/produk_bloc.dart';
4. import 'package:tokokita/model/produk.dart';
5. import 'package:tokokita/ui/login_page.dart';
6. import 'package:tokokita/ui/produk_detail.dart';
7. import 'package:tokokita/ui/produk_form.dart';
8.
9. class ProdukPage extends StatefulWidget {
10.   const ProdukPage({Key? key}) : super(key: key);
11.
12.   @override
13.   _ProdukPageState createState() => _ProdukPageState();
14. }
15.
16. class _ProdukPageState extends State<ProdukPage> {
17.   @override
18.   Widget build(BuildContext context) {
19.     return Scaffold(
20.       appBar: AppBar(

```

```

21.      title: const Text('List Produk'),
22.      actions: [
23.        Padding(
24.          padding: const EdgeInsets.only(right: 20.0),
25.          child: GestureDetector(
26.            child: const Icon(Icons.add, size: 26.0),
27.            onTap: () async {
28.              Navigator.push(context,
29.                MaterialPageRoute(builder: (context) => ProdukForm()));
30.            },
31.          )),
32.        ],
33.      ),
34.      drawer: Drawer(
35.        child: ListView(
36.          children: [
37.            ListTile(
38.              title: const Text('Logout'),
39.              trailing: const Icon(Icons.logout),
40.              onTap: () async {
41.                await LogoutBloc.logout().then((value) => {
42.                  Navigator.pushReplacement(context,
43.                    MaterialPageRoute(builder: (context) => LoginPage()))
44.                );
45.              },
46.            ),
47.          ],
48.        ),
49.      ),
50.      body: FutureBuilder<List>(
51.        future: ProdukBloc.getProduks(),
52.        builder: (context, snapshot) {
53.          if (snapshot.hasError) print(snapshot.error);
54.          return snapshot.hasData
55.            ? ListProduk(
56.              list: snapshot.data,
57.            )
58.            : const Center(
59.              child: CircularProgressIndicator(),
60.            );
61.        },
62.      ),
63.    );
64.  }
65. }
66.
67. class ListProduk extends StatelessWidget {
68.   final List? list;
69.
70.   const ListProduk({Key? key, this.list}) : super(key: key);
71.
72.   @override
73.   Widget build(BuildContext context) {
74.     return ListView.builder(
75.       itemCount: list == null ? 0 : list!.length,
76.       itemBuilder: (context, i) {
77.         return ItemProduk(
78.           produk: list![i],
79.         );
80.       });
81.   }
82. }
83.
84. class ItemProduk extends StatelessWidget {
85.   final Produk produk;

```

```

86.
87.   const ItemProduk({Key? key, required this.produk}) : super(key: key);
88.
89.   @override
90.   Widget build(BuildContext context) {
91.     return GestureDetector(
92.       onTap: () {
93.         Navigator.push(
94.           context,
95.           MaterialPageRoute(
96.             builder: (context) => ProdukDetail(
97.               produk: produk,
98.             ))));
99.   },
100.   child: Card(
101.     child: ListTile(
102.       title: Text(produk.namaProduk!),
103.       subtitle: Text(produk.hargaProduk.toString()),
104.     ),
105.   ),
106. );
107. }
108. }
```

Adapun perubahan yang dilakukan adalah penambahan sebuah class bernama **ListProduk** dengan kode

```

67. class ListProduk extends StatelessWidget {
68.   final List? list;
69.
70.   const ListProduk({Key? key, this.list}) : super(key: key);
71.
72.   @override
73.   Widget build(BuildContext context) {
74.     return ListView.builder(
75.       itemCount: list == null ? 0 : list!.length,
76.       itemBuilder: (context, i) {
77.         return ItemProduk(
78.           produk: list![i],
79.         );
80.       });
81.   }
82. }
```

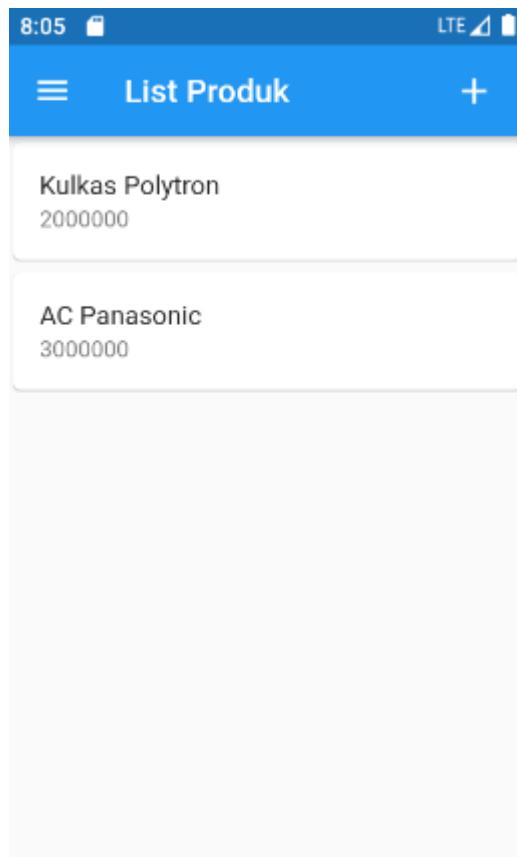
Kemudian perubahan pada bagian **body** menjadi

```

50. body: FutureBuilder<List>(
51.   future: ProdukBloc.getProduks(),
52.   builder: (context, snapshot) {
53.     if (snapshot.hasError) print(snapshot.error);
54.     return snapshot.hasData
55.       ? ListProduk(
56.         list: snapshot.data,
57.       )
58.       : const Center(
59.         child: CircularProgressIndicator(),
60.       );
61.   },
62. ),
```

Serta memasukkan beberapa package yang diperlukan

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/logout_bloc.dart';
3. import 'package:tokokita/bloc/produk_bloc.dart';
4. import 'package:tokokita/model/produk.dart';
5. import 'package:tokokita/ui/login_page.dart';
6. import 'package:tokokita/ui/produk_detail.dart';
7. import 'package:tokokita/ui/produk_form.dart';
```



Memodifikasi Form Produk (produk_form.dart)

Membuat fungsi simpan

Agar tombol simpan dapat berfungsi diperlukan kode fungsi untuk menyimpan data dengan memanggil bloc `produk_bloc` yang telah dibuat sebelumnya, kita akan menambahkan sebuah fungsi dengan nama `simpan` dan memodifikasi fungsi `_buttonSubmit`

```
116.      //Membuat Tombol Simpan/Ubah
117.      Widget _buttonSubmit() {
118.        return OutlinedButton(
119.          child: Text(tombolSubmit),
120.          onPressed: () {
121.            var validate = _formKey.currentState!.validate();
122.            if (validate) {
123.              if (!isLoading) {
124.                if (widget.produk != null) {
```

```

125.          //kondisi update produk
126.
127.          } else {
128.              //kondisi tambah produk
129.              simpan();
130.          }
131.      }
132.  });
133. });
134. }
135.
136. simpan() {
137.     setState(() {
138.         _isLoading = true;
139.     });
140.     Produk createProduk = Produk(id: null);
141.     createProduk.kodeProduk = _kodeProdukTextboxController.text;
142.     createProduk.namaProduk = _namaProdukTextboxController.text;
143.     createProduk.hargaProduk =
144.         int.parse(_hargaProdukTextboxController.text);
145.     ProdukBloc.addProduk(produk: createProduk).then((value) {
146.         Navigator.of(context).push(MaterialPageRoute(
147.             builder: (BuildContext context) => const ProdukPage()));
148.     }, onError: (error) {
149.         showDialog(
150.             context: context,
151.             builder: (BuildContext context) => const WarningDialog(
152.                 description: "Simpan gagal, silahkan coba lagi",
153.             ));
154.     });
155.     setState(() {
156.         _isLoading = false;
157.     });

```

Dengan keseluruhan kode menjadi

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/produk_bloc.dart';
3. import 'package:tokokita/model/produk.dart';
4. import 'package:tokokita/ui/produk_page.dart';
5. import 'package:tokokita/widget/warning_dialog.dart';
6.
7. class ProdukForm extends StatefulWidget {
8.     Produk? produk;
9.
10.    ProdukForm({Key? key, this.produk}) : super(key: key);
11.
12.    @override
13.    _ProdukFormState createState() => _ProdukFormState();
14. }
15.
16. class _ProdukFormState extends State<ProdukForm> {
17.     final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
18.     bool _isLoading = false;
19.     String judul = "TAMBAH PRODUK";
20.     String tombolSubmit = "SIMPAN";
21.
22.     final TextEditingController _kodeProdukTextboxController =
23.         TextEditingController();
24.     final TextEditingController _namaProdukTextboxController =
25.         TextEditingController();
26.     final TextEditingController _hargaProdukTextboxController =
27.         TextEditingController();

```

```

28.     super.initState();
29.     isUpdate();
30.   }
31.
32.   isUpdate() {
33.     if (widget.produk != null) {
34.       setState(() {
35.         judul = "UBAH PRODUK";
36.         tombolSubmit = "UBAH";
37.         _kodeProdukTextboxController.text = widget.produk!.kodeProduk!;
38.         _namaProdukTextboxController.text = widget.produk!.namaProduk!;
39.         _hargaProdukTextboxController.text =
40.           widget.produk!.hargaProduk.toString();
41.       });
42.     } else {
43.       judul = "TAMBAH PRODUK";
44.       tombolSubmit = "SIMPAN";
45.     }
46.   }
47.
48.   @override
49.   Widget build(BuildContext context) {
50.     return Scaffold(
51.       appBar: AppBar(title: Text(judul)),
52.       body: SingleChildScrollView(
53.         child: Padding(
54.           padding: const EdgeInsets.all(8.0),
55.           child: Form(
56.             key: _formKey,
57.             child: Column(
58.               children: [
59.                 _kodeProdukTextField(),
60.                 _namaProdukTextField(),
61.                 _hargaProdukTextField(),
62.                 _buttonSubmit()
63.               ],
64.             ),
65.           ),
66.         ),
67.       ),
68.     );
69.   }
70.
71.   //Membuat Textbox Kode Produk
72.   Widget _kodeProdukTextField() {
73.     return TextFormField(
74.       decoration: const InputDecoration(labelText: "Kode Produk"),
75.       keyboardType: TextInputType.text,
76.       controller: _kodeProdukTextboxController,
77.       validator: (value) {
78.         if (value!.isEmpty) {
79.           return "Kode Produk harus diisi";
80.         }
81.         return null;
82.       },
83.     );
84.   }
85.
86.   //Membuat Textbox Nama Produk
87.   Widget _namaProdukTextField() {
88.     return TextFormField(
89.       decoration: const InputDecoration(labelText: "Nama Produk"),
90.       keyboardType: TextInputType.text,
91.       controller: _namaProdukTextboxController,
92.       validator: (value) {

```

```

93.         if (value!.isEmpty) {
94.             return "Nama Produk harus diisi";
95.         }
96.         return null;
97.     },
98. );
99. }
100.
101. //Membuat Textbox Harga Produk
102. Widget _hargaProdukTextField() {
103.     return TextFormField(
104.         decoration: const InputDecoration(labelText: "Harga"),
105.         keyboardType: TextInputType.number,
106.         controller: _hargaProdukTextboxController,
107.         validator: (value) {
108.             if (value!.isEmpty) {
109.                 return "Harga harus diisi";
110.             }
111.             return null;
112.         },
113.     );
114. }
115.
116. //Membuat Tombol Simpan/Ubah
117. Widget _buttonSubmit() {
118.     return OutlinedButton(
119.         child: Text(tombolSubmit),
120.         onPressed: () {
121.             var validate = _formKey.currentState!.validate();
122.             if (validate) {
123.                 if (!isLoading) {
124.                     if (widget.produk != null) {
125.                         //kondisi update produk
126.
127.                         } else {
128.                             //kondisi tambah produk
129.                             simpan();
130.                         }
131.                     }
132.                 }
133.             });
134. }
135.
136. simpan() {
137.     setState(() {
138.         _isLoading = true;
139.     });
140.     Produk createProduk = Produk(id: null);
141.     createProduk.kodeProduk = _kodeProdukTextboxController.text;
142.     createProduk.namaProduk = _namaProdukTextboxController.text;
143.     createProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text);
144.     ProdukBloc.addProduk(produk: createProduk).then((value) {
145.         Navigator.of(context).push(MaterialPageRoute(
146.             builder: (BuildContext context) => const ProdukPage()));
147.     }, onError: (error) {
148.         showDialog(
149.             context: context,
150.             builder: (BuildContext context) => const WarningDialog(
151.                 description: "Simpan gagal, silahkan coba lagi",
152.             ));
153.     });
154.     setState(() {
155.         _isLoading = false;
156.     });
157. }

```

```
158. }
```

Membuat fungsi ubah

Sama halnya dengan simpan, kita buat sebuah fungsi **ubah** kemudian kita sertakan pada fungsi **_buttonSubmit**

Dengan fungsi ubah sebagai berikut

```
159.     ubah() {
160.         setState(() {
161.             _isLoading = true;
162.         });
163.         Produk updateProduk = Produk(id: null);
164.         updateProduk.id = widget.produk!.id;
165.         updateProduk.kodeProduk = _kodeProdukTextboxController.text;
166.         updateProduk.namaProduk = _namaProdukTextboxController.text;
167.         updateProduk.hargaProduk =
168.             int.parse(_hargaProdukTextboxController.text);
169.         ProdukBloc.updateProduk(produk: updateProduk).then((value) {
170.             Navigator.of(context).push(MaterialPageRoute(
171.                 builder: (BuildContext context) => const ProdukPage()));
172.         }, onError: (error) {
173.             showDialog(
174.                 context: context,
175.                 builder: (BuildContext context) => const WarningDialog(
176.                     description: "Permintaan ubah data gagal, silahkan coba
177. lagi",
178.                 ));
179.             setState(() {
180.                 _isLoading = false;
181.             });
182.         });
183.     }
```

Kemudian kita tambahkan pada fungsi **_buttonSubmit**

```
116. //Membuat Tombol Simpan/Ubah
117. Widget _buttonSubmit() {
118.     return OutlinedButton(
119.         child: Text(tombolSubmit),
120.         onPressed: () {
121.             var validate = _formKey.currentState!.validate();
122.             if (validate) {
123.                 if (!isLoading) {
124.                     if (widget.produk != null) {
125.                         //kondisi update produk
126.                         ubah();
127.                     } else {
128.                         //kondisi tambah produk
129.                         simpan();
130.                     }
131.                 }
132.             }
133.         });
134. }
```

Dengan kode keseluruhan menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/bloc/produk_bloc.dart';
3. import 'package:tokokita/model/produk.dart';
4. import 'package:tokokita/ui/produk_page.dart';
5. import 'package:tokokita/widget/warning_dialog.dart';
6.
7. class ProdukForm extends StatefulWidget {
8.   Produk? produk;
9.
10.  ProdukForm({Key? key, this.produk}) : super(key: key);
11.
12.  @override
13.  _ProdukFormState createState() => _ProdukFormState();
14. }
15.
16. class _ProdukFormState extends State<ProdukForm> {
17.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
18.   bool _isLoading = false;
19.   String judul = "TAMBAH PRODUK";
20.   String tombolSubmit = "SIMPAN";
21.
22.   final _kodeProdukTextboxController = TextEditingController();
23.   final _namaProdukTextboxController = TextEditingController();
24.   final _hargaProdukTextboxController = TextEditingController();
25.
26.   @override
27.   void initState() {
28.     super.initState();
29.     isUpdate();
30.   }
31.
32.   isUpdate() {
33.     if (widget.produk != null) {
34.       setState(() {
35.         judul = "UBAH PRODUK";
36.         tombolSubmit = "UBAH";
37.         _kodeProdukTextboxController.text = widget.produk!.kodeProduk!;
38.         _namaProdukTextboxController.text = widget.produk!.namaProduk!;
39.         _hargaProdukTextboxController.text =
40.           widget.produk!.hargaProduk.toString();
41.       });
42.     } else {
43.       judul = "TAMBAH PRODUK";
44.       tombolSubmit = "SIMPAN";
45.     }
46.   }
47.
48.   @override
49.   Widget build(BuildContext context) {
50.     return Scaffold(
51.       appBar: AppBar(title: Text(judul)),
52.       body: SingleChildScrollView(
53.         child: Padding(
54.           padding: const EdgeInsets.all(8.0),
55.           child: Form(
56.             key: _formKey,
57.             child: Column(
58.               children: [
59.                 _kodeProdukTextField(),
60.                 _namaProdukTextField(),
61.                 _hargaProdukTextField(),
62.                 _buttonSubmit()
63.               ],
64.             ),
65.           ),
66.         ),
67.       ),
68.     );
69.   }
70.
```

```

63.           ],
64.           ),
65.           ),
66.           ),
67.           );
68.       );
69.   }
70.
71.   //Membuat Textbox Kode Produk
72.   Widget _kodeProdukTextField() {
73.       return TextFormField(
74.           decoration: const InputDecoration(labelText: "Kode Produk"),
75.           keyboardType: TextInputType.text,
76.           controller: _kodeProdukTextboxController,
77.           validator: (value) {
78.               if (value!.isEmpty) {
79.                   return "Kode Produk harus diisi";
80.               }
81.               return null;
82.           },
83.       );
84.   }
85.
86.   //Membuat Textbox Nama Produk
87.   Widget _namaProdukTextField() {
88.       return TextFormField(
89.           decoration: const InputDecoration(labelText: "Nama Produk"),
90.           keyboardType: TextInputType.text,
91.           controller: _namaProdukTextboxController,
92.           validator: (value) {
93.               if (value!.isEmpty) {
94.                   return "Nama Produk harus diisi";
95.               }
96.               return null;
97.           },
98.       );
99.   }
100.
101.      //Membuat Textbox Harga Produk
102.      Widget _hargaProdukTextField() {
103.          return TextFormField(
104.              decoration: const InputDecoration(labelText: "Harga"),
105.              keyboardType: TextInputType.number,
106.              controller: _hargaProdukTextboxController,
107.              validator: (value) {
108.                  if (value!.isEmpty) {
109.                      return "Harga harus diisi";
110.                  }
111.                  return null;
112.              },
113.          );
114.      }
115.
116.      //Membuat Tombol Simpan/Ubah
117.      Widget _buttonSubmit() {
118.          return OutlinedButton(
119.              child: Text(tombolSubmit),
120.              onPressed: () {
121.                  var validate = _formKey.currentState!.validate();
122.                  if (validate) {
123.                      if (!isLoading) {
124.                          if (widget.produk != null) {
125.                              //kondisi update produk
126.                              ubah();
127.                          } else {

```

```

128.          //kondisi tambah produk
129.          simpan();
130.      }
131.    }
132.  });
133. });
134. }
135.
136. simpan() {
137.   setState(() {
138.     _isLoading = true;
139.   });
140.   Produk createProduk = Produk(id: null);
141.   createProduk.kodeProduk = _kodeProdukTextboxController.text;
142.   createProduk.namaProduk = _namaProdukTextboxController.text;
143.   createProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text);
144.   ProdukBloc.addProduk(produk: createProduk).then((value) {
145.     Navigator.of(context).push(MaterialPageRoute(
146.       builder: (BuildContext context) => const ProdukPage()));
147.   }, onError: (error) {
148.     showDialog(
149.       context: context,
150.       builder: (BuildContext context) => const WarningDialog(
151.         description: "Simpan gagal, silahkan coba lagi",
152.       ));
153.   });
154.   setState(() {
155.     _isLoading = false;
156.   });
157. }
158.
159. ubah() {
160.   setState(() {
161.     _isLoading = true;
162.   });
163.   Produk updateProduk = Produk(id: null);
164.   updateProduk.id = widget.produk!.id;
165.   updateProduk.kodeProduk = _kodeProdukTextboxController.text;
166.   updateProduk.namaProduk = _namaProdukTextboxController.text;
167.   updateProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text);
168.   ProdukBloc.updateProduk(produk: updateProduk).then((value) {
169.     Navigator.of(context).push(MaterialPageRoute(
170.       builder: (BuildContext context) => const ProdukPage()));
171.   }, onError: (error) {
172.     showDialog(
173.       context: context,
174.       builder: (BuildContext context) => const WarningDialog(
175.         description: "Permintaan ubah data gagal, silahkan coba lagi",
176.       ));
177.   });
178.   setState(() {
179.     _isLoading = false;
180.   });
181. }
182. }

```

Menambahkan fungsi hapus pada Detail Produk (produk_detail.dart)

Buka file `produk_detail.dart` pada folder `ui`, kemudian kita modifikasi pada fungsi `confirmHapus` menjadi seperti berikut

```

65. void confirmHapus() {
66.   AlertDialog alertDialog = AlertDialog(

```

```

67.     content: const Text("Yakin ingin menghapus data ini?"),
68.     actions: [
69.       //tombol hapus
70.       OutlinedButton(
71.         child: const Text("Ya"),
72.         onPressed: () {
73.           Navigator.push(
74.             context,
75.             MaterialPageRoute(
76.               builder: (context) => ProdukForm(
77.                 produk: widget.produk!,
78.               )));
79.         },
80.       ),
81.       //tombol batal
82.       OutlinedButton(
83.         child: const Text("Batal"),
84.         onPressed: () => Navigator.pop(context),
85.       )
86.     ],
87.   );
88.
89.   showDialog(builder: (context) => alertDialog, context: context);
90. }

```

Dengan kode keseluruhan menjadi

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/model/produk.dart';
3. import 'package:tokokita/ui/produk_form.dart';
4.
5. class ProdukDetail extends StatefulWidget {
6.   Produk? produk;
7.
8.   ProdukDetail({Key? key, this.produk}) : super(key: key);
9.
10.  @override
11.  _ProdukDetailState createState() => _ProdukDetailState();
12. }
13.
14. class _ProdukDetailState extends State<ProdukDetail> {
15.   @override
16.   Widget build(BuildContext context) {
17.     return Scaffold(
18.       appBar: AppBar(
19.         title: const Text('Detail Produk'),
20.       ),
21.       body: Center(
22.         child: Column(
23.           children: [
24.             Text(
25.               "Kode : ${widget.produk!.kodeProduk}",
26.               style: const TextStyle(fontSize: 20.0),
27.             ),
28.             Text(
29.               "Nama : ${widget.produk!.namaProduk}",
30.               style: const TextStyle(fontSize: 18.0),
31.             ),
32.             Text(
33.               "Harga : Rp. ${widget.produk!.hargaProduk.toString()}",
34.               style: const TextStyle(fontSize: 18.0),
35.             ),
36.             _tombolHapusEdit()
37.           ],

```

```

38.        ),
39.        ),
40.    );
41. }
42.
43. Widget _tombolHapusEdit() {
44.     return Row(
45.         mainAxisAlignment: MainAxisAlignment.min,
46.         children: [
47.             //Tombol Edit
48.             OutlinedButton(
49.                 child: const Text("EDIT"),
50.                 onPressed: () {
51.                     Navigator.push(
52.                         context,
53.                         MaterialPageRoute(
54.                             builder: (context) => ProdukForm(
55.                                 produk: widget.produk!,
56.                             )));
57.                 },
58.             //Tombol Hapus
59.             OutlinedButton(
60.                 child: const Text("DELETE"), onPressed: () => confirmHapus(),
61.             ],
62.         );
63.     }
64.
65.     void confirmHapus() {
66.         AlertDialog alertDialog = AlertDialog(
67.             content: const Text("Yakin ingin menghapus data ini?"),
68.             actions: [
69.                 //tombol hapus
70.                 OutlinedButton(
71.                     child: const Text("Ya"),
72.                     onPressed: () {
73.                         Navigator.push(
74.                             context,
75.                             MaterialPageRoute(
76.                                 builder: (context) => ProdukForm(
77.                                     produk: widget.produk!,
78.                                     )));
79.                     },
80.                 ),
81.                 //tombol batal
82.                 OutlinedButton(
83.                     child: const Text("Batal"),
84.                     onPressed: () => Navigator.pop(context),
85.                 )
86.             ],
87.         );
88.     }
89.     showDialog(builder: (context) => alertDialog, context: context);
90. }
91. }

```

Daftar Pustaka

- [1]. A. Sasongko, M.S. Maulana & W. Nugraha. "Web Programming; Membangun Aplikasi Mobile Kolaborasi Antara Flutter dan Codeigniter". Graha Ilmu. 2019.
- [2]. <https://flutter.dev/multi-platform/mobile>
- [3]. <https://developer.android.com/studio>
- [4]. A. Sasongko, M.S. Maulana, & Latifah. "Presensi Karyawan Berbasis Aplikasi Mobile Dengan Filter Jaringan Intranet Dan Imei". Jurnal Sistemasi, vol. 9, no. 1, pp. 92–102, 2020.