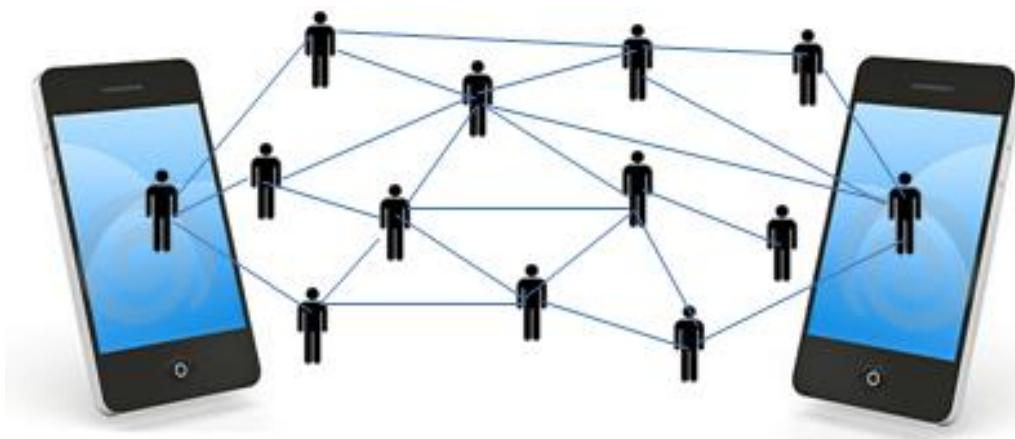




MODUL

MOBILE Programming



Disusun Oleh:

UNIT PENGEMBANGAN AKADEMIK

UNIVERSITAS BINA SARANA INFORMATIKA

FAKULTAS TEKNIK DAN INFORMATIKA

PROGRAM STUDI ILMU KOMPUTER

2021

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 mempelajari mobile programming dan mengembangkan aplikasi mobile yang dikhususkan kepada sistem operasi android.

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

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

Pontianak, Maret 2021

Tim Penulis

DAFTAR ISI

KATA PENGANTAR.....	2
PERTEMUAN I.....	7
1. Pengenalan Mobile Programming	7
a. Mobile Programming	7
b. Perbandingan Flutter dan React Native.....	8
2. Persiapan Aplikasi Mobile dengan Flutter	9
a. Menyiapkan perangkat Hardware	9
b. Menyiapkan Perangkat Software.....	9
1) Instalasi Git.....	9
2) Instalasi JDK.....	10
3) Instalasi Android Studio	15
4) Instalasi Flutter	18
5) Konfigurasi Android Studio dengan Flutter	24
3. Tugas Pertemuan 1	25
PERTEMUAN 2.....	26
1. Membuat dan menjalankan projek	26
2. Membuat dan menjalankan projek dengan VSCode dan Handphone Android.....	34
a. Instalasi VSCode sebagai alternatif editor	34
b. Membuat projek flutter dengan VSCode.....	34
c. Menjalankan aplikasi dengan Handphone Android.....	37
3. Struktur Folder Flutter	45
a. Membuat Hello World	46
b. Membuat Widget Column	51
c. Membuat Widget Row.....	52
d. Mengenal StatelessWidget dan StatefulWidget	54
4. Tugas Pertemuan 2	55
PERTEMUAN 3.....	59
1. Mengenal Widget dan Fungsinya	59
2. Pemisahan Widget Kedalam fungsi-fungsi.....	59
3. Membuat Detail Produk.....	60
4. Membuat fungsi tombol simpan dan menampilkan data pada Detail Produk.....	60
5. Membuat ListView Produk.....	64
6. Tugas Pertemuan 3	65

PERTEMUAN 4.....	66
1. Membuat Route (Pindah Halaman)	66
2. Pemisahan Widget ke dalam Class StatelessWidget	67
3. Menampilkan Detail Produk saat ListView diklik	69
4. Tugas Pertemuan 4	73
PERTEMUAN 5.....	74
1. Membuat projek flutter yang terhubung dengan API	74
a. Apa itu API.....	74
b. Arsitektur API.....	74
2. Membuat projek Toko API (Restful API)	75
a. Installasi Apache, MySql dan PHP (XAMPP).....	75
b. Install Composer	80
c. Install Postman.....	82
3. API SPEC	84
a. Registrasi.....	84
b. Login.....	84
c. Produk.....	85
4. Tugas Pertemuan 5	87
PERTEMUAN 6.....	88
1. Pembuatan Database.....	88
a. SQL membuat tabel member.....	88
b. SQL membuat tabel member_token	88
c. SQL membuat tabel produk.....	88
2. Installasi Lumen sebagai Restful API.....	89
3. Konfigurasi Projek	89
a. Koneksi ke database.....	89
b. Membuat hasil response.....	91
4. Registrasi.....	92
a. Membuat model Registrasi.....	92
b. Membuat controller Registrasi	93
c. Menambah route Registrasi	94
PERTEMUAN 7	98
1. Konsep MVC.....	98
2. Membuat Login	98
a. Membuat model Member	98

b.	Membuat model Login.....	99
c.	Membuat controller Login	99
d.	Menambahkan route Login.....	100
e.	Mencoba Rest	100
3.	CRUD Produk.....	101
f.	Membuat model Produk.....	101
g.	Membuat controller produk	101
c.	Mencoba Rest	105
PERTEMUAN 8		108
PERTEMUAN 9		109
1.	Membangun Projek Dengan Flutter	109
2.	Pengembangan Projek flutter tokokita.....	110
a.	Membuat Model	111
b.	Membuat Class Login Pada Model.....	111
c.	Membuat Class Registrasi Pada Model.....	112
d.	Membuat Class Produk Pada Model.....	112
3.	Membuat Halaman	113
a.	Membuat Form Registrasi.....	113
b.	Membuat Form Login	117
c.	Membuat Form Produk.....	120
d.	Membuat Form Detail Produk	122
e.	Membuat Tampilan List Produk.....	123
PERTEMUAN 10		129
1.	Membuat Helper Modul	129
a.	Menambahkan depencies.....	129
b.	Membuat Class Token.....	130
c.	Http request.....	132
2.	Membuat Bloc.....	135
a.	Registrasi.....	136
b.	Login.....	136
c.	Logout	136
d.	Produk.....	137
PERTEMUAN 11		139
1.	Menyatukan Fungsionalitas	139
a.	Membuat Common Dialog Widget	139

b.	Modifikasi main.dart.....	142
c.	Modifikasi registrasi_page.dart	143
d.	Modifikasi login_page.dart (fungsi login)	147
PERTEMUAN 12		152
1.	Modifikasi produk_page.dart.....	152
a.	Menambahkan fungsi logout pada drawer.....	152
b.	Menampilkan Data Produk dari Rest API.....	154
2.	Memodifikasi Form Produk (produk_form.dart)	157
a.	Membuat fungsi simpan	157
b.	Membuat fungsi ubah.....	160
PERTEMUAN 13-15		167

PERTEMUAN I

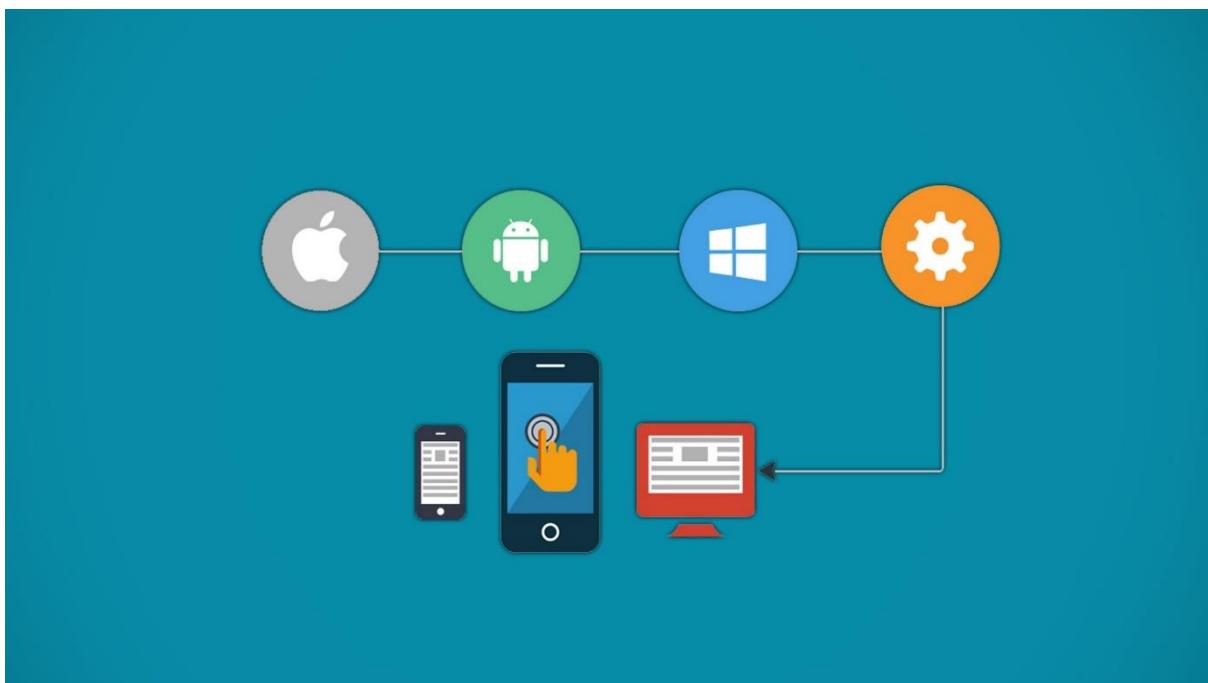
1. Pengenalan Mobile Programming

a. Mobile Programming

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

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

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



Gambar 1.1 Mobile Programming

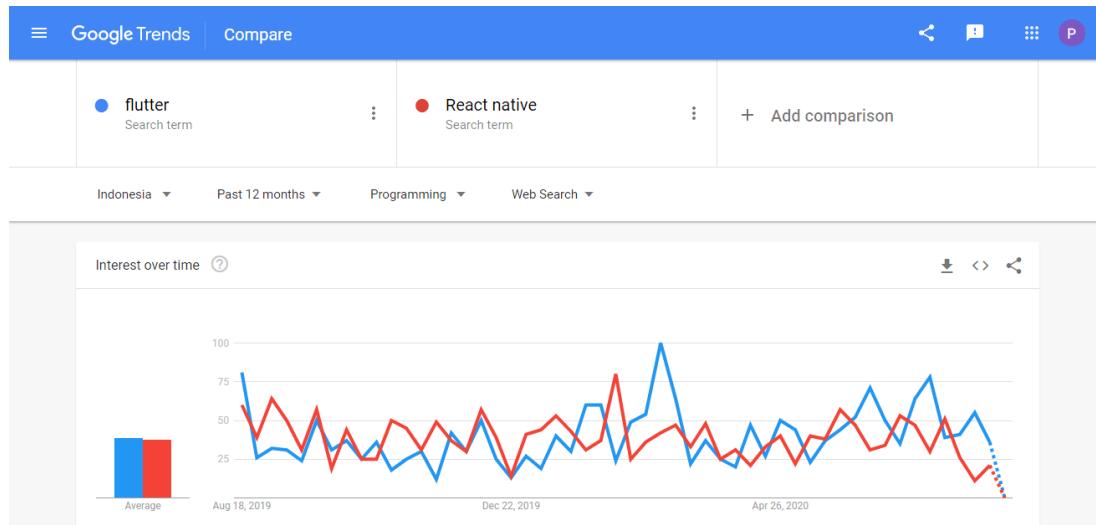
Sumber: <https://www.adwebstudio.com/>

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

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

b. Perbandingan Flutter dan React Native

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



Gambar 1.2 Perbandingan search term dari Flutter vs React Native di Indonesia, Agustus 2019 – Agustus 2020

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

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

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

Gambar 1.3 Perbandingan singkat antara Flutter vs React Native

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

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

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

2. Persiapan Aplikasi Mobile dengan Flutter

a. Menyiapkan perangkat Hardware

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

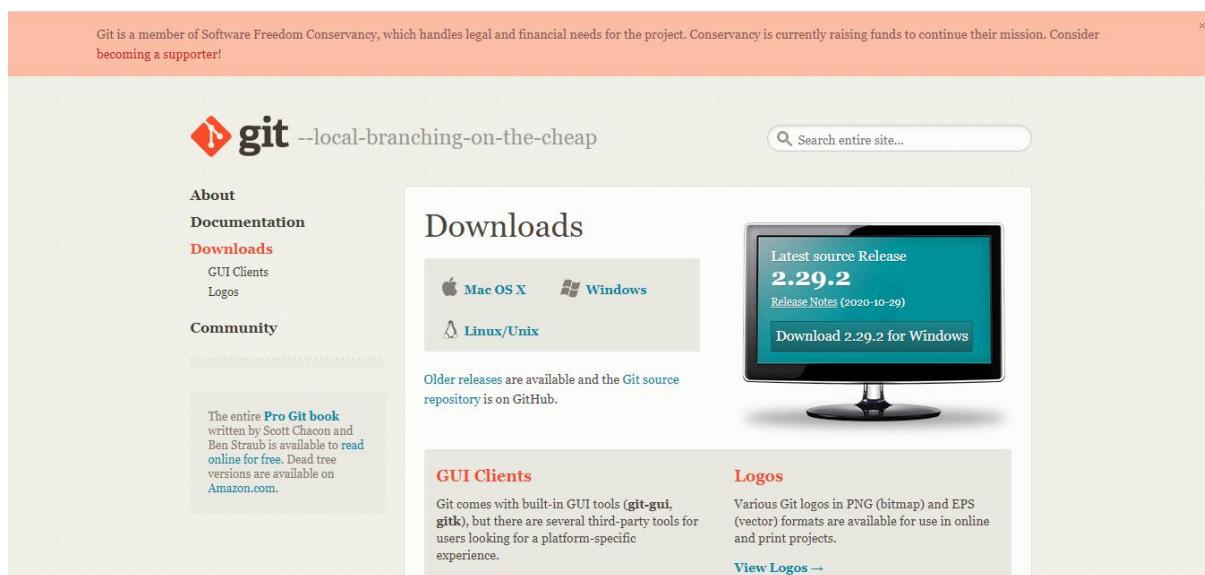
Untuk kebutuhan spesifikasi minimum hardware yang perlu dipersiapkan adalah:

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

b. Menyiapkan Perangkat Software

- 1) Instalasi Git

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



Gambar 1.2 Download Git

Sumber: <https://git-scm.com/downloads>

Kemudian lakukan installasi git dari file yang telah diunduh.

2) Instalasi 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/>



Gambar 1.3 Download JDK

Sumber : <https://jdk.java.net/>

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

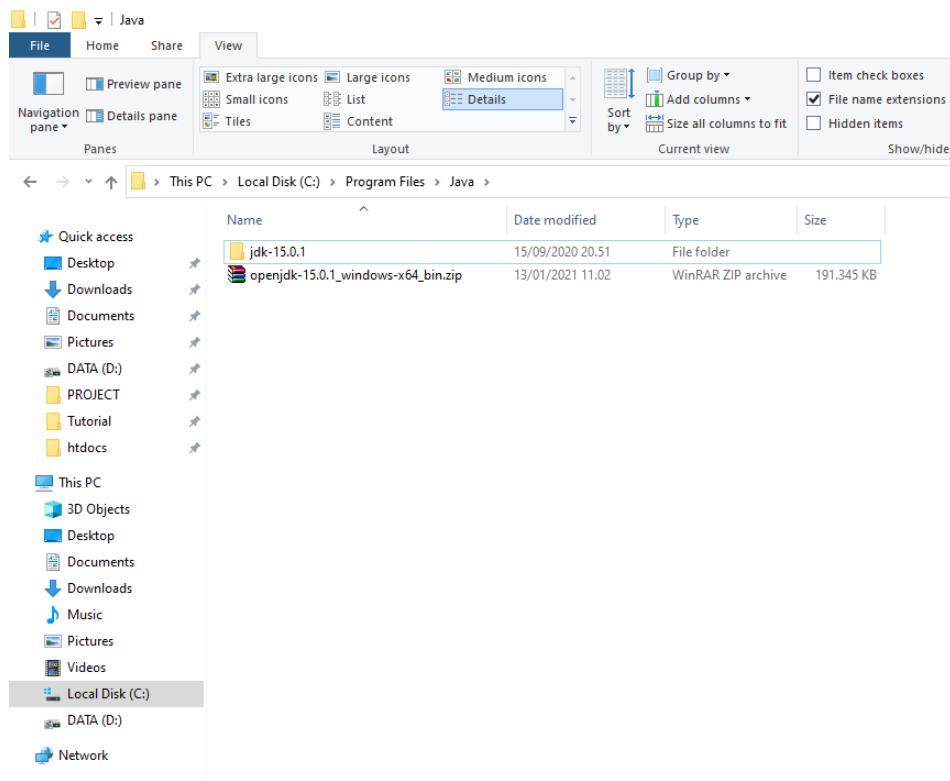
This screenshot shows the 'JDK 15.0.2 General-Availability Release' page. On the left, there's a sidebar with links for 'GA Releases' (JDK 15, JMC 7), 'Early-Access Releases' (JDK 17, 16, JMC 8, Lanai, Loom, Metropolis, Panama, Valhalla), 'Reference Implementations' (Java SE 15, 14, 13, 12, 11, 10, 9, 8, 7), and 'Feedback' (Report a bug). The main content area has a heading 'JDK 15.0.2 General-Availability Release'. It includes a paragraph about the release being production-ready open-source builds of the Java Development Kit, version 15, under the GNU General Public License, version 2, with the Classpath Exception. It also mentions commercial builds available on the Oracle Technology Network. Below this is a 'Documentation' section with links to 'Features', 'Release notes', and 'API Javadoc'. The 'Builds' section lists download links for different platforms: Linux/AArch64 (tar.gz), Linux/x64 (tar.gz), macOS/x64 (tar.gz), and Windows/x64 (zip).

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

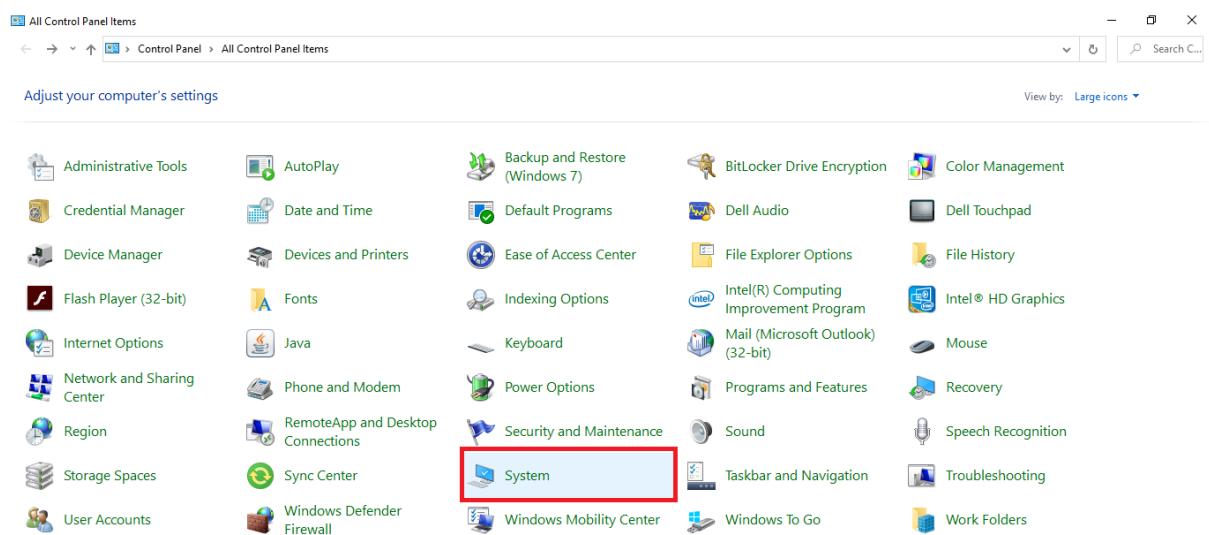
Gambar 1.4 Pilih JDK 15

Sumber : <https://jdk.java.net/>

Kemudian extrak berkas file tersebut pada laptop/komputer



Kemudian buka Control Panel, pilih "System"

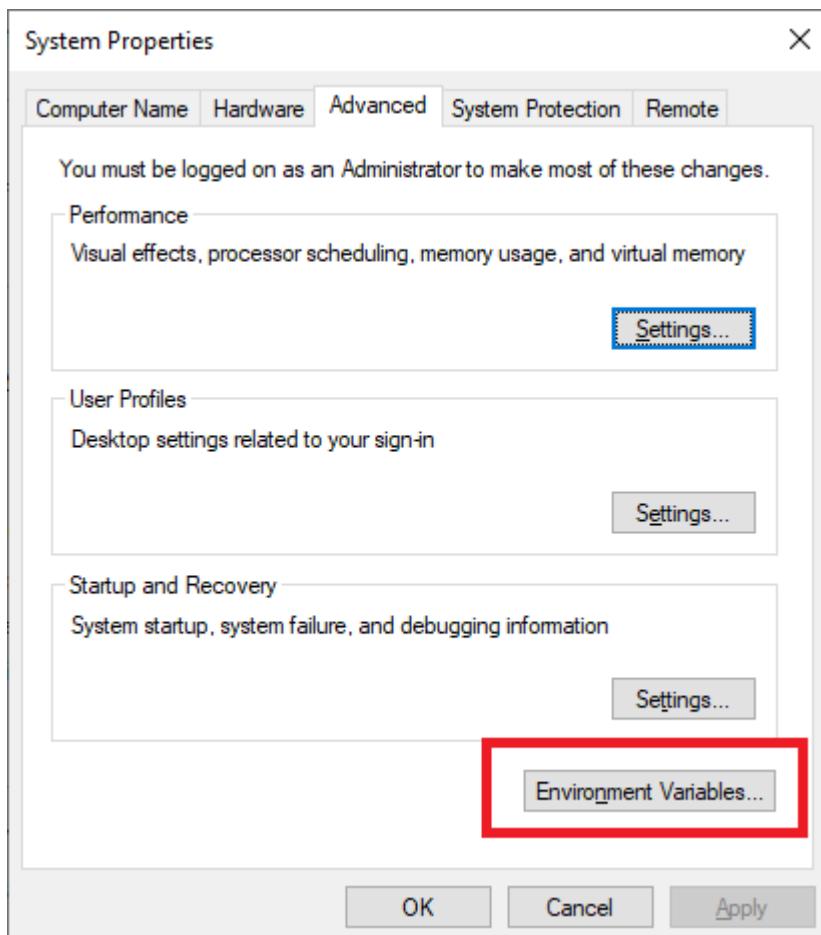


Kemudian pilih "Advanced System Setting"

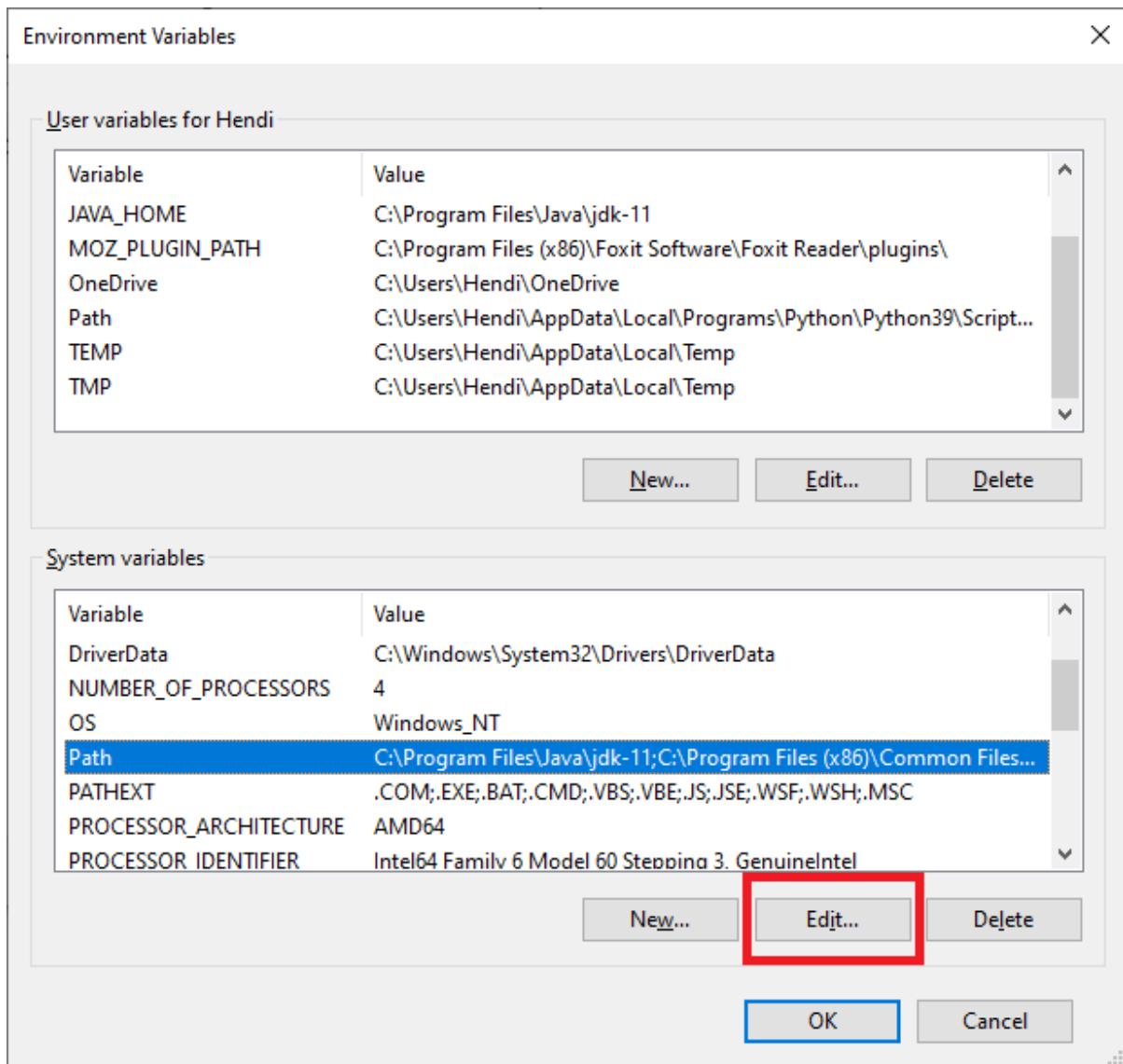
Control Panel Home

-  Device Manager
-  Remote settings
-  System protection
-  Advanced system settings

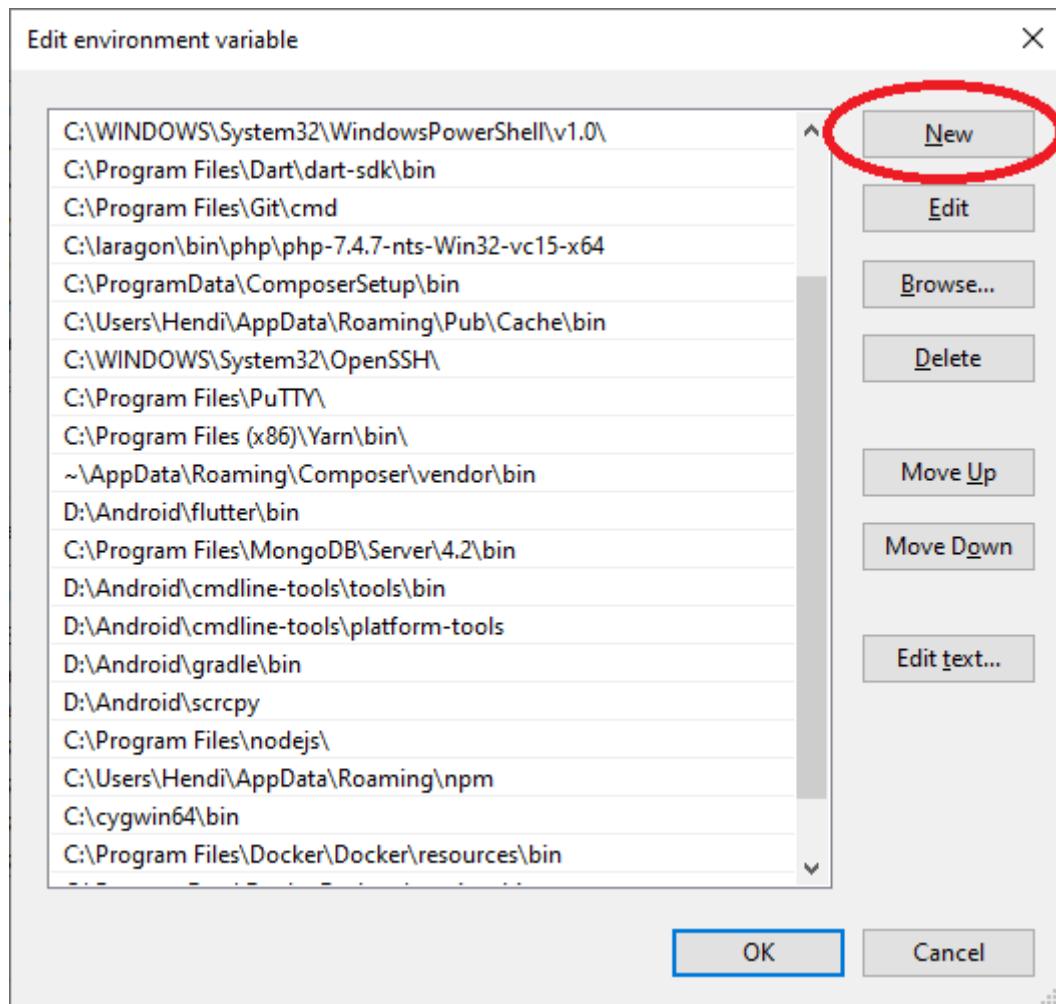
Kemudian klik tombol "Environtment Variables"



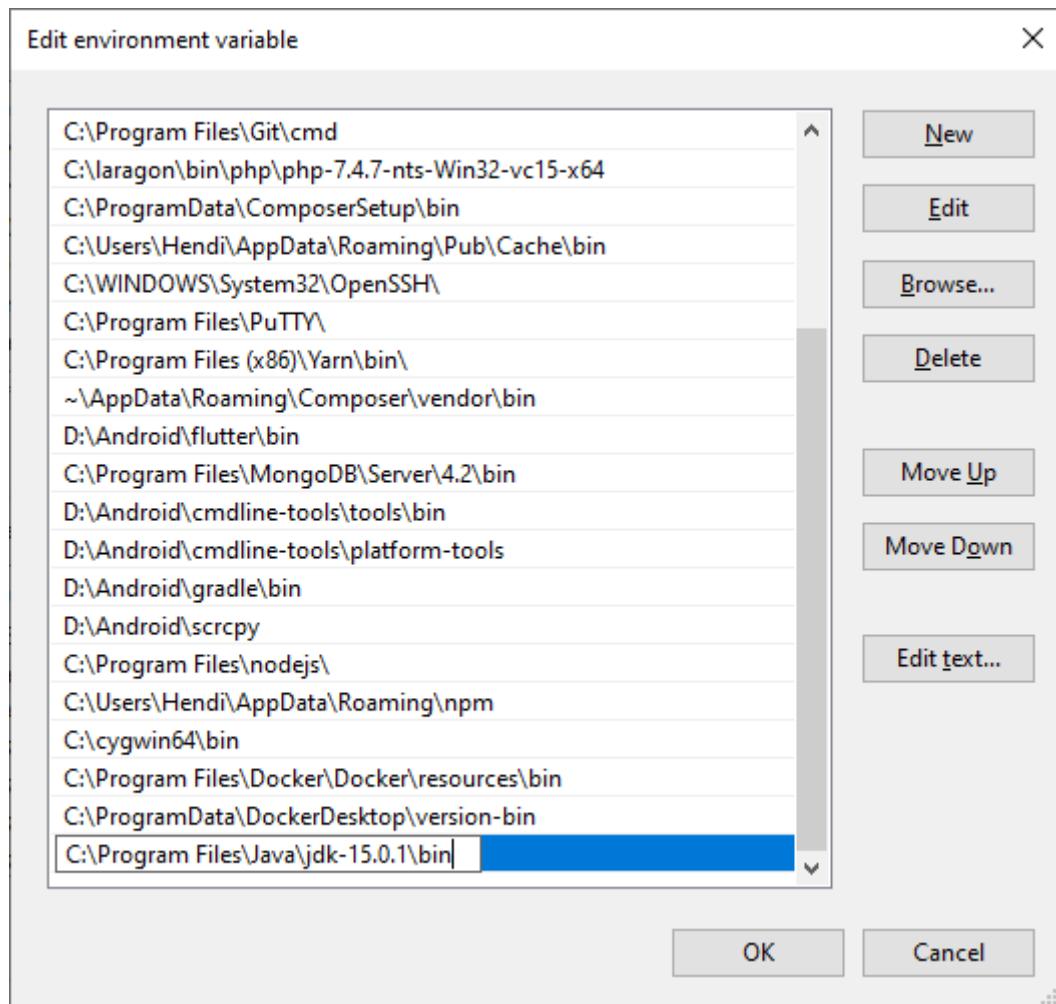
Pilih **Path** pada bagian System Variables kemudian Klik tombol "Edit"



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”



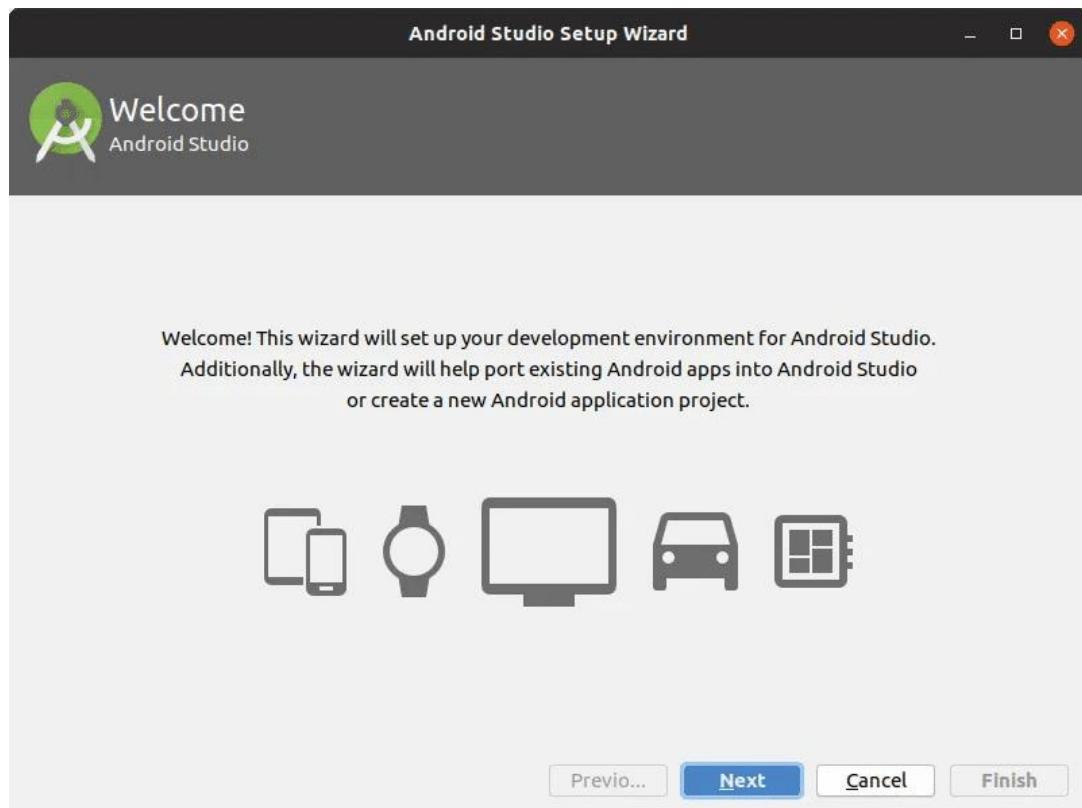
Biasanya agar JDK dapat berfungsi perlu dilakukan restart laptop/komputer. Untuk memeriksa apakah installasi berhasil, buka command prompt kemudian ketikkan **java -version** atau **javac -version**

```
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:\> |
```

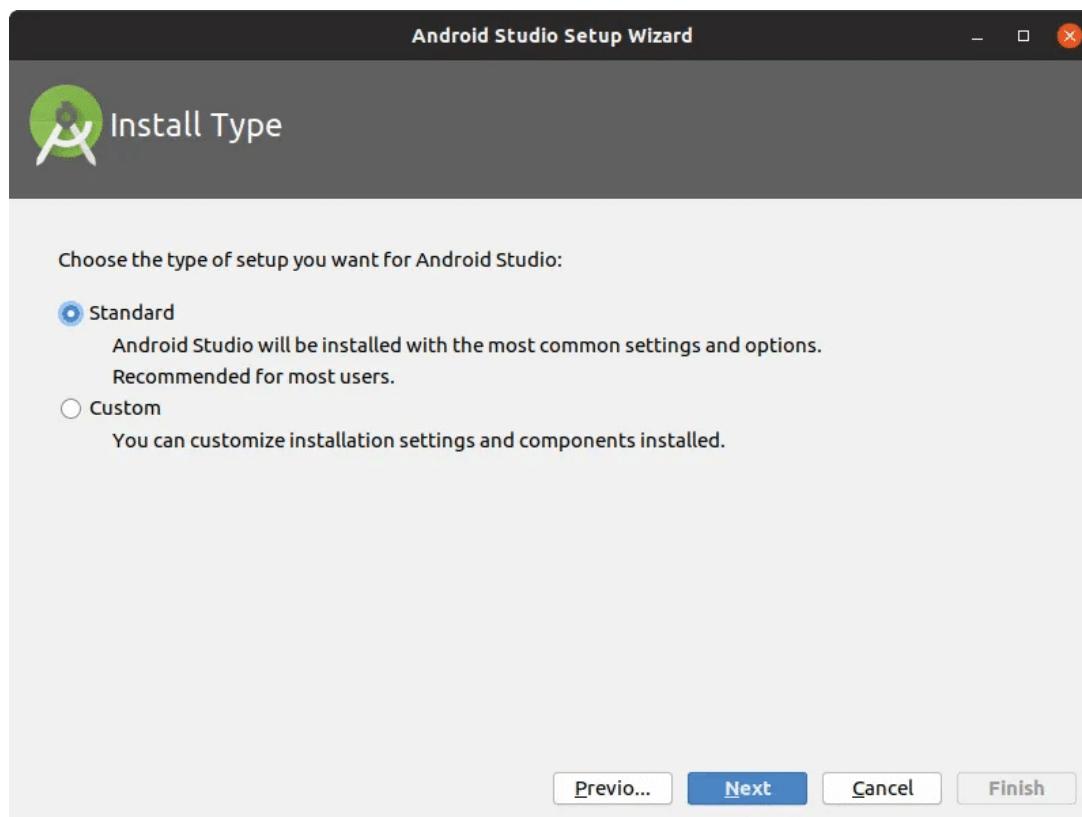
3) Instalasi Android Studio

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

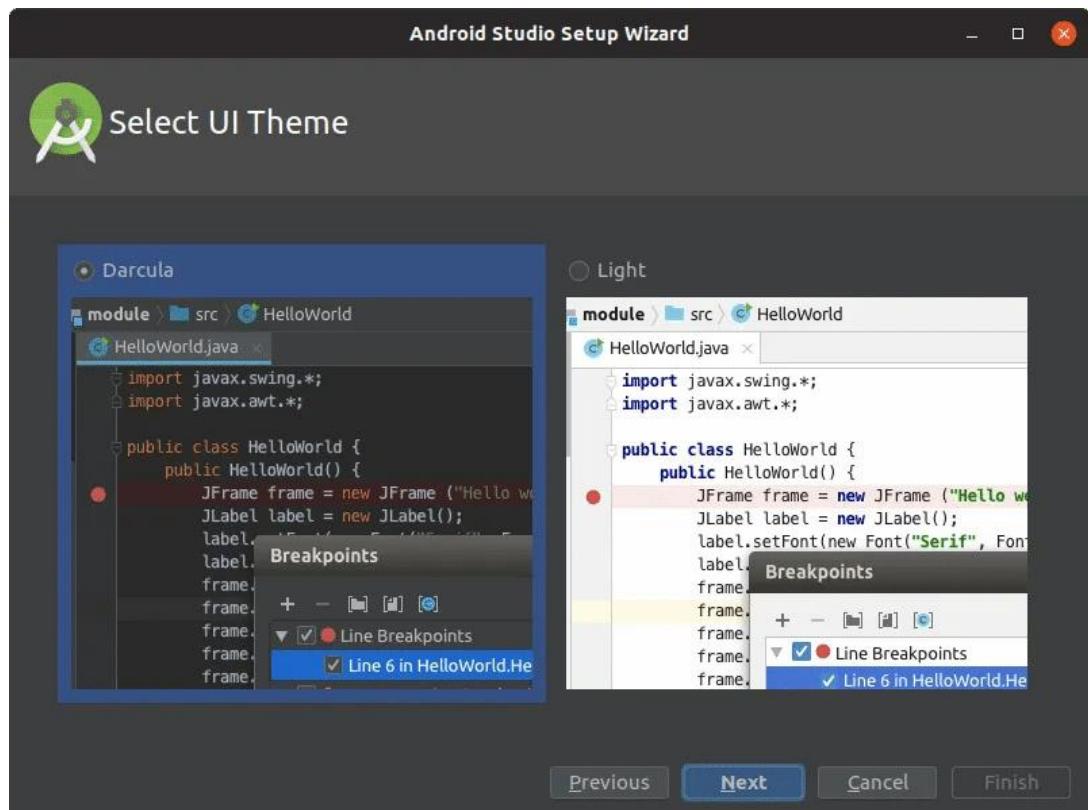
Setelah diunduh klik dua kali pada file yang telah diunduh tersebut, kemudian lakukan instalasi dengan mengikuti langkah-langkah yang telah disediakan



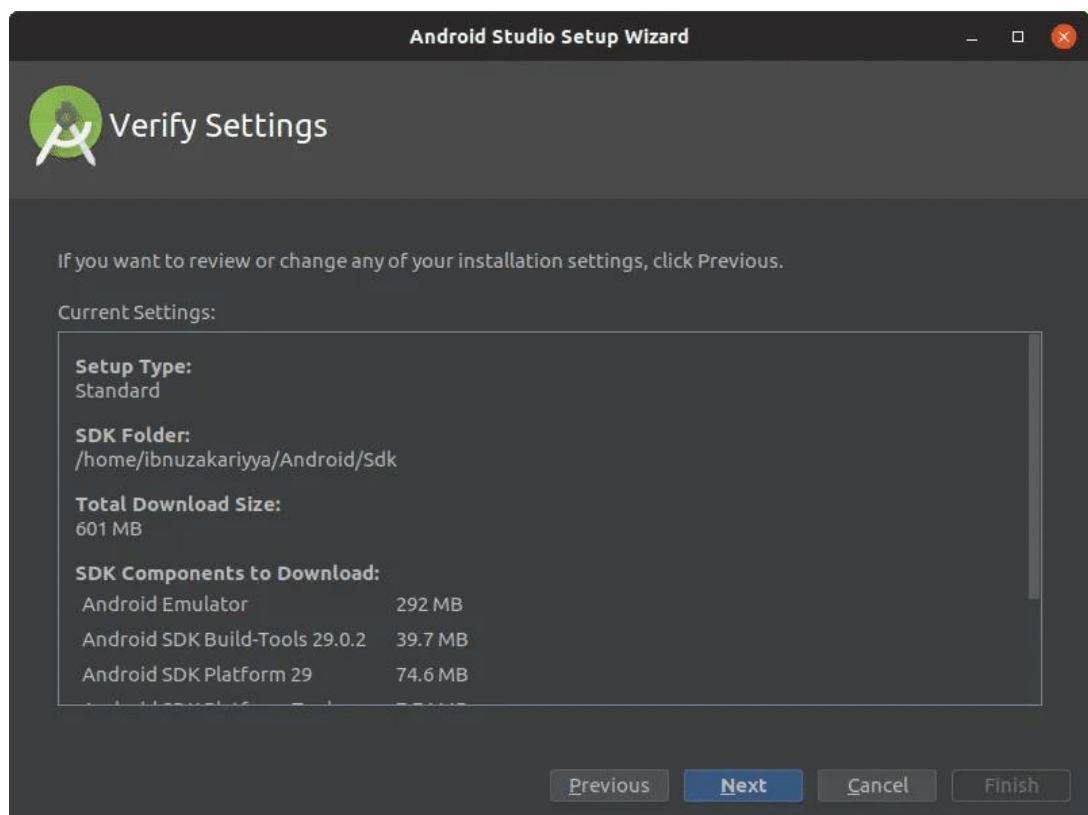
Kemudian pilih tipe standar dan klik next



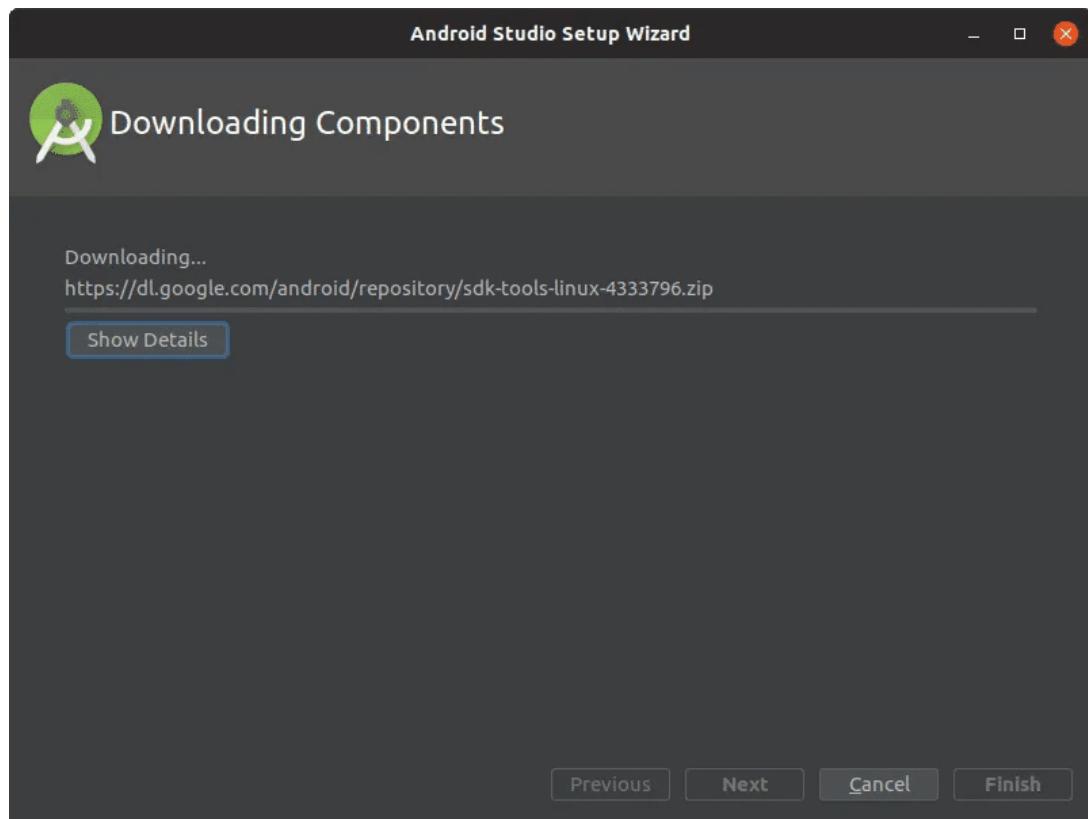
Pilih tema tampilan kemudian klik next



Kemudian klik tombol next



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



Setelah selesai klik finish

4) Instalasi 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 sistem operasinya, untuk "Windows" maka klik Windows dan akan muncul link download flutter untuk sistem operasi 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

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

Get the Flutter SDK

1. 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.

2. 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

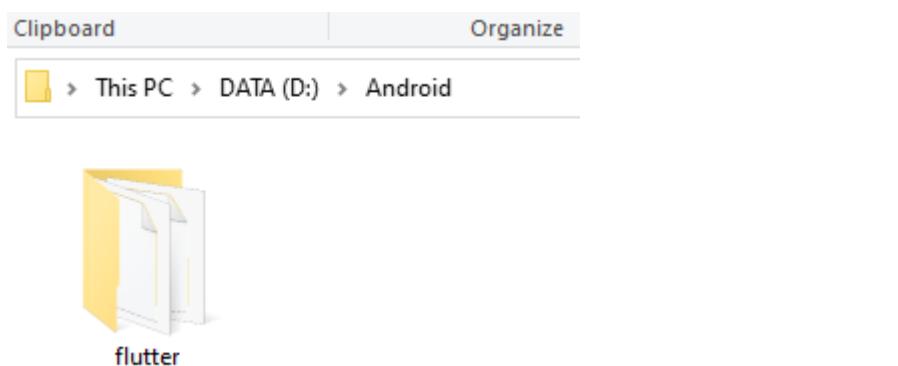
Set up your Android device

Set up the Android emulator

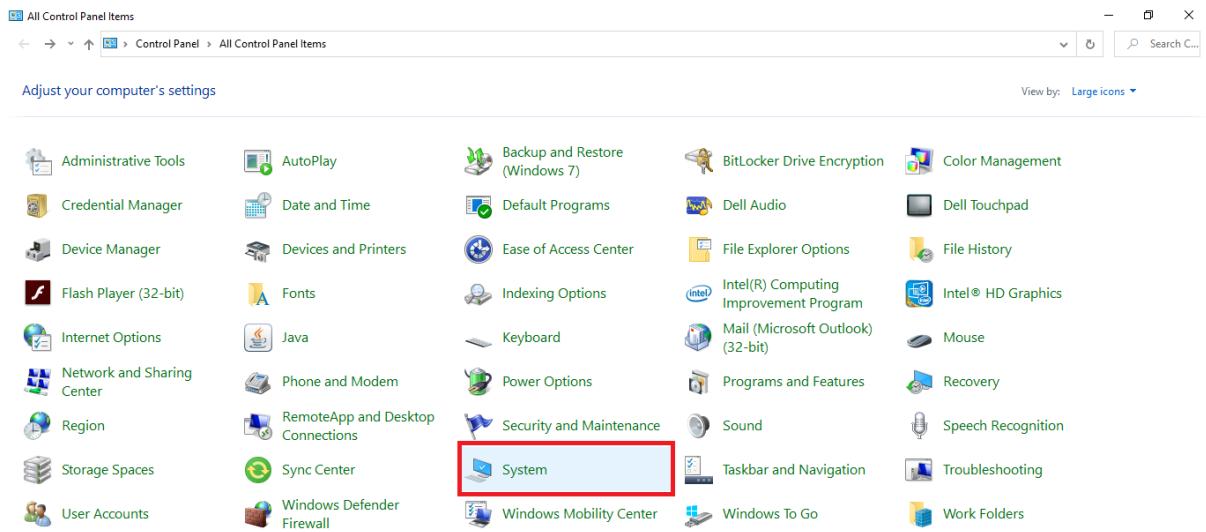
Web setup

Next step

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



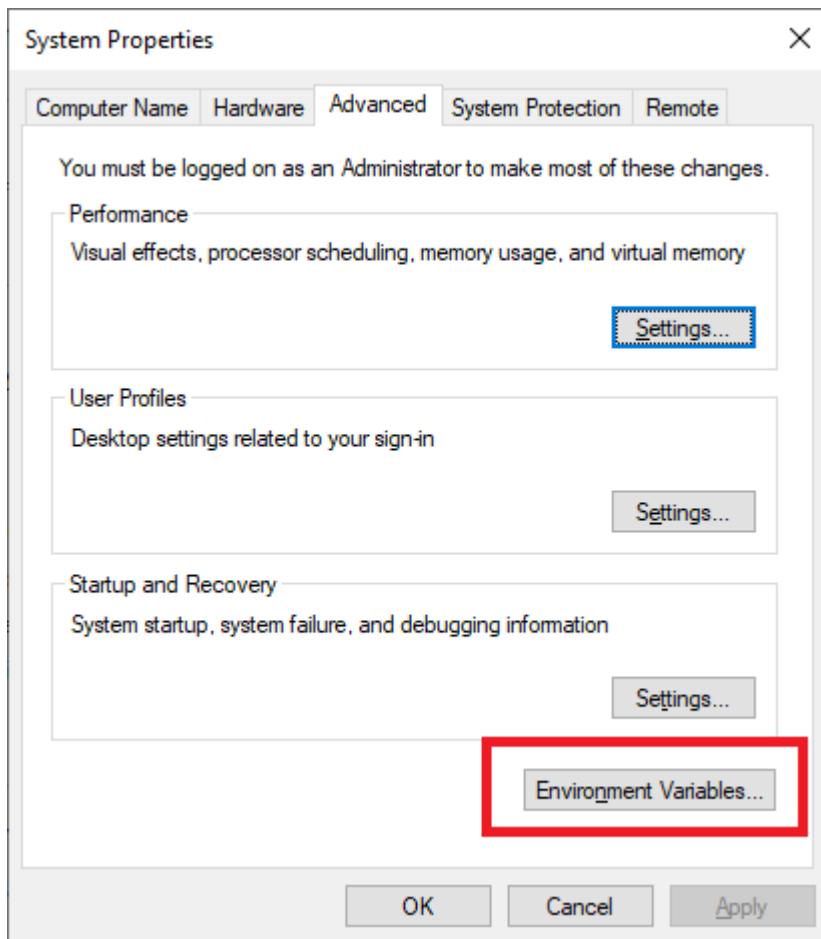
Kemudian buka **Control Panel**, pilih "System"



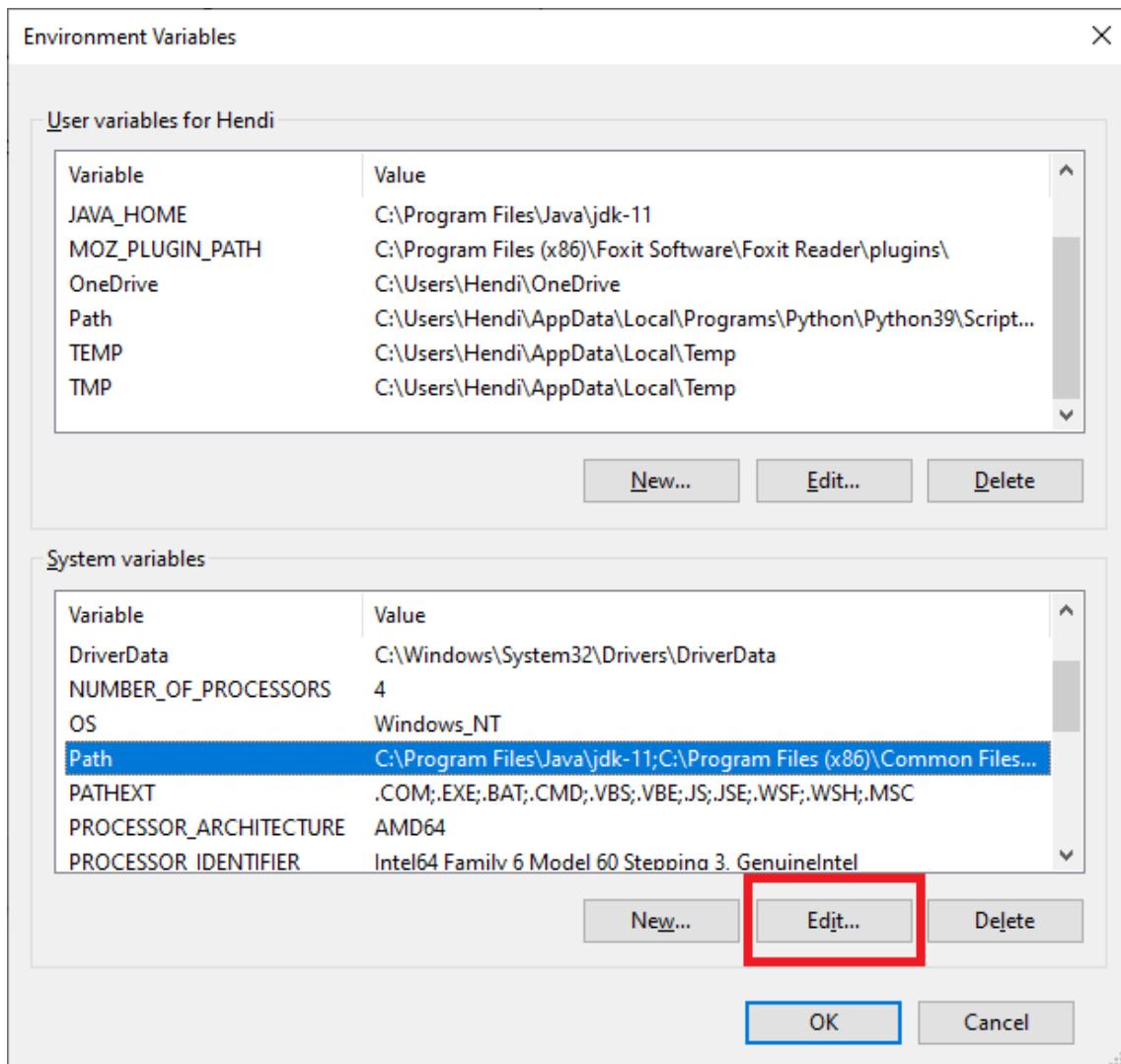
Kemudian pilih "Advanced System Setting"



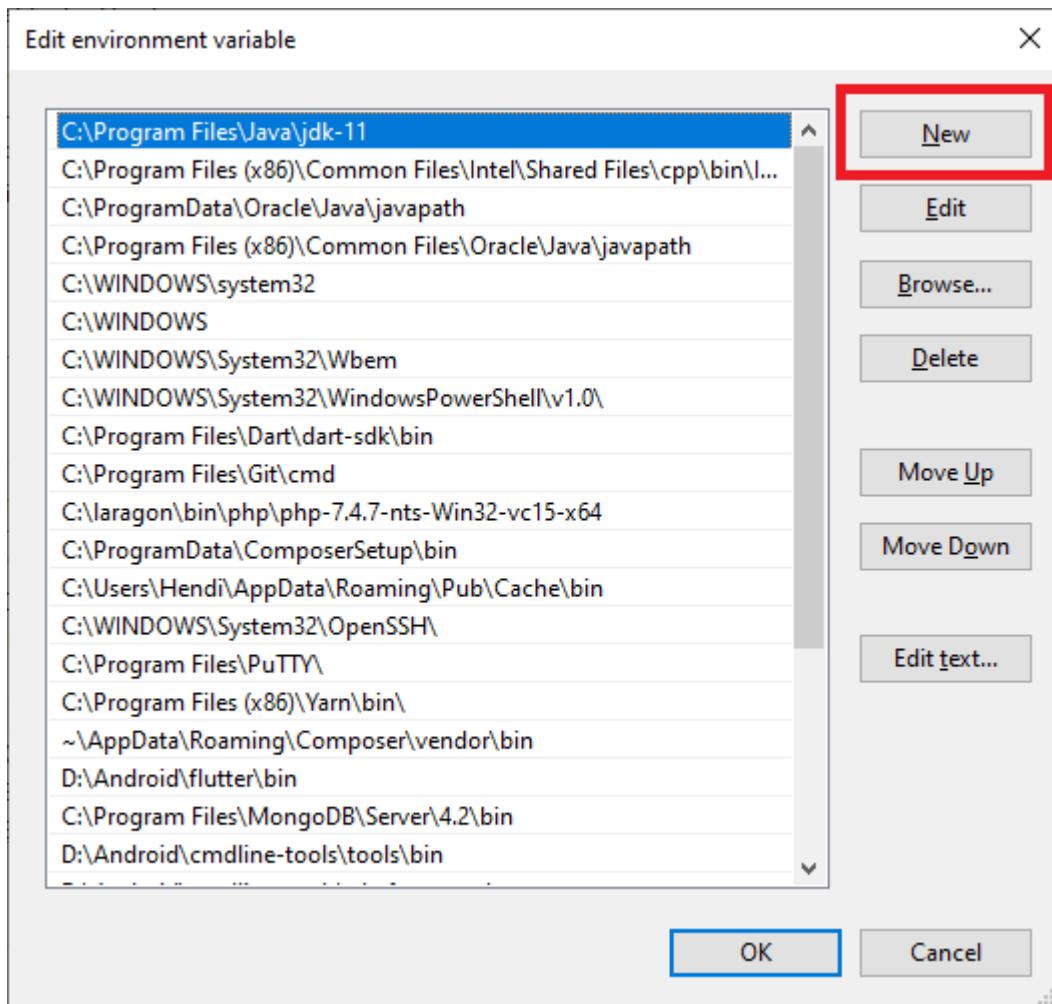
Kemudian klik tombol "Environtment Variables"



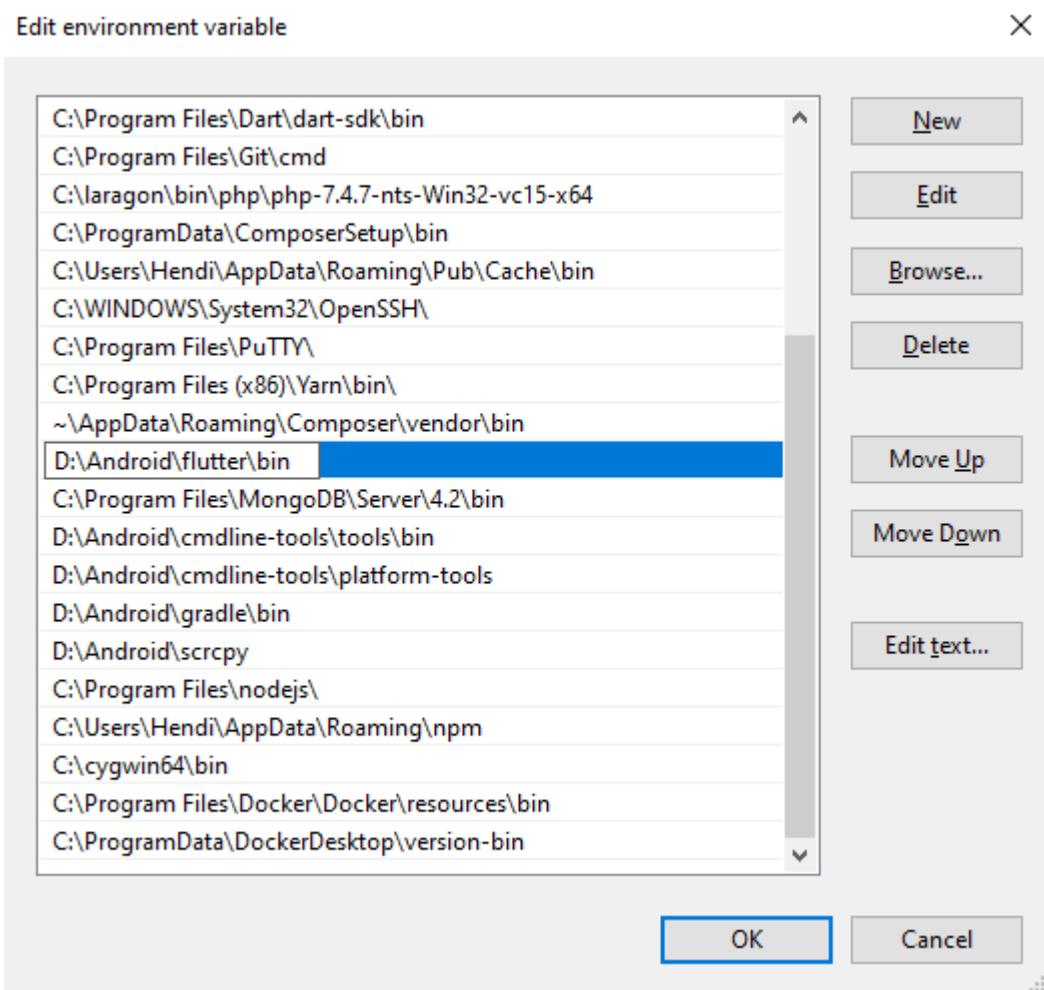
Kemudian pilih pada “System Variables” pilih “Path” dan klik tombol “Edit”



Kemudian klik tombol “New”

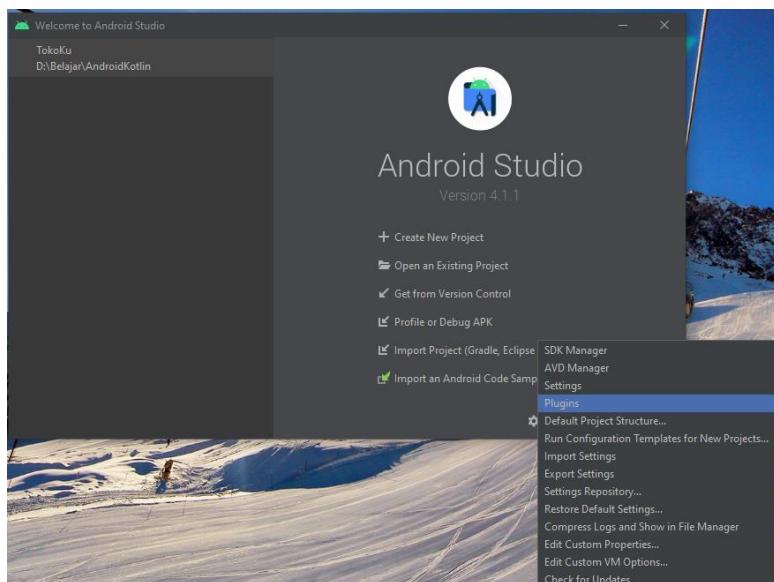


Kemudian masukkan alamat folder bin pada flutter yang telah kita ekstrak dalam hal ini misalnya "D:\Android\flutter\bin"

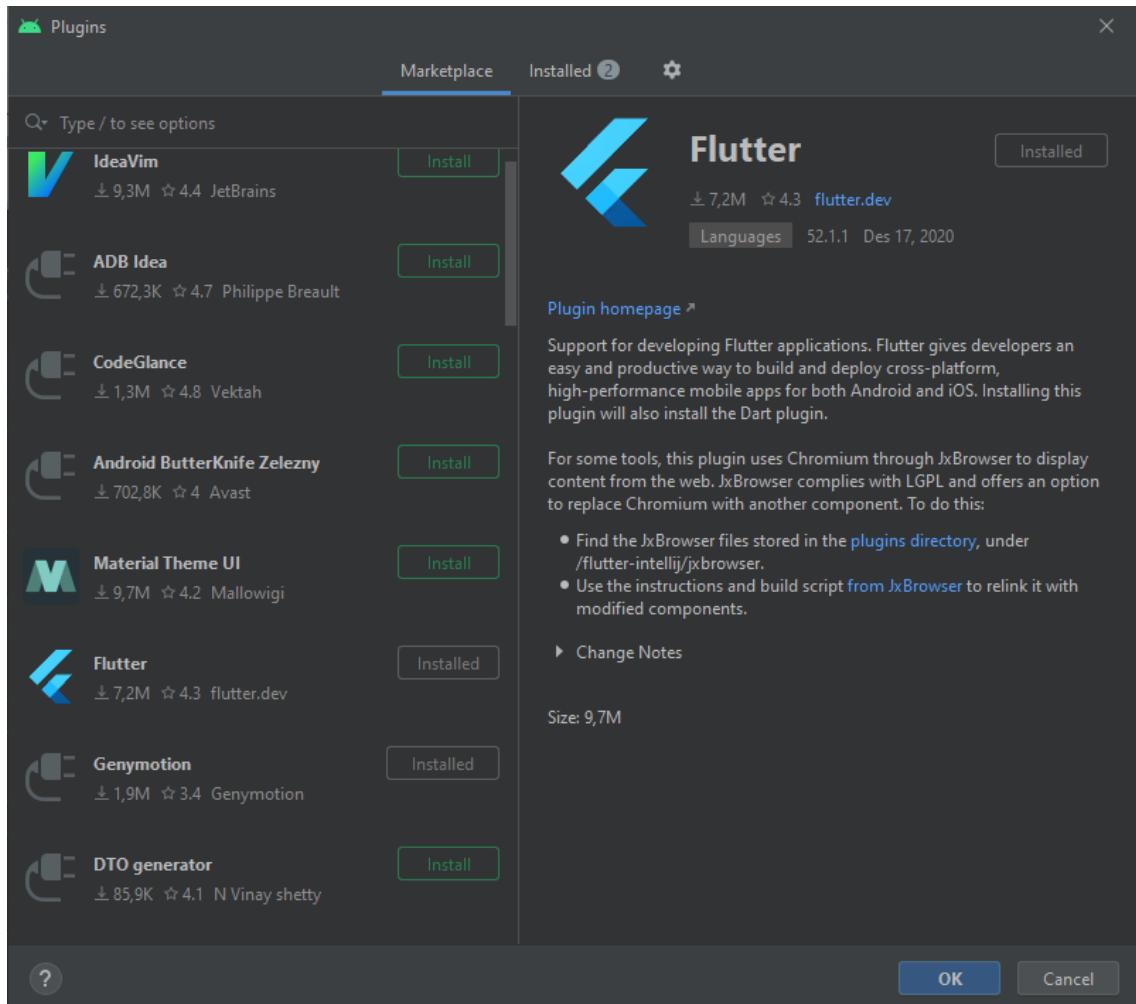


5) Konfigurasi Android Studio dengan Flutter

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



Pilih “Flutter” kemudian klik tombol “Install”



Setelah selesai, restart/tutup Android Studio

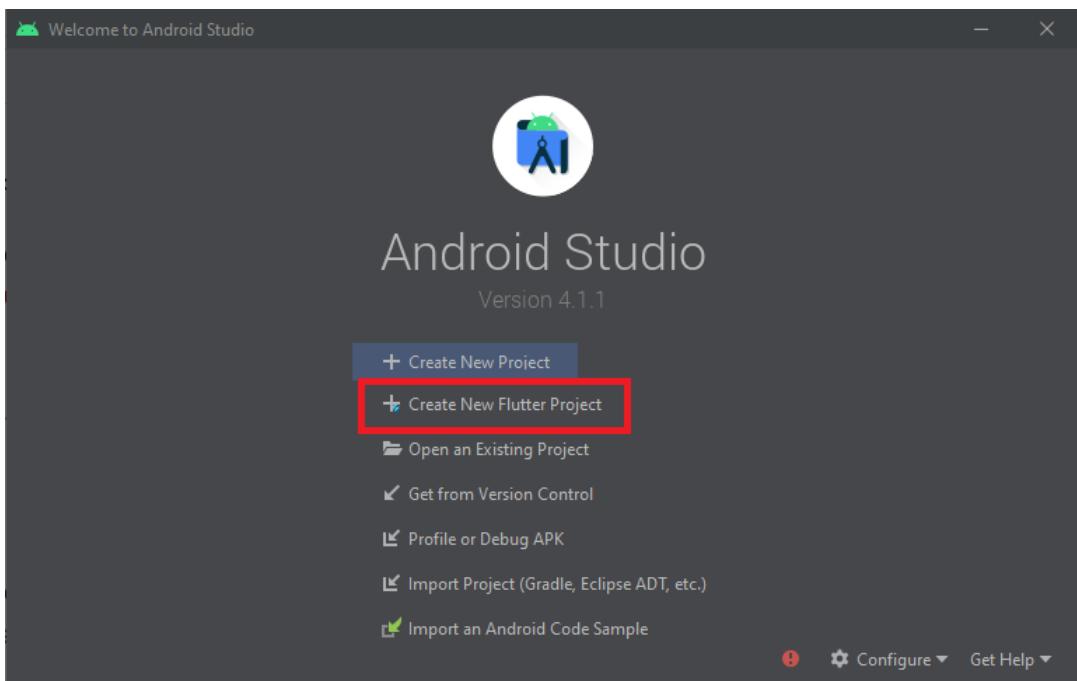
3. Tugas Pertemuan 1

1. Download VSCode pada laman <https://code.visualstudio.com/download>
2. Lakukan instalasi VSCode dengan mengikuti petunjuk pada Pertemuan 2 part 2 tentang instalasi VSCode sebagai alternatif editor.
3. Memastikan kebutuhan software VSCode sudah terinstal dengan baik.

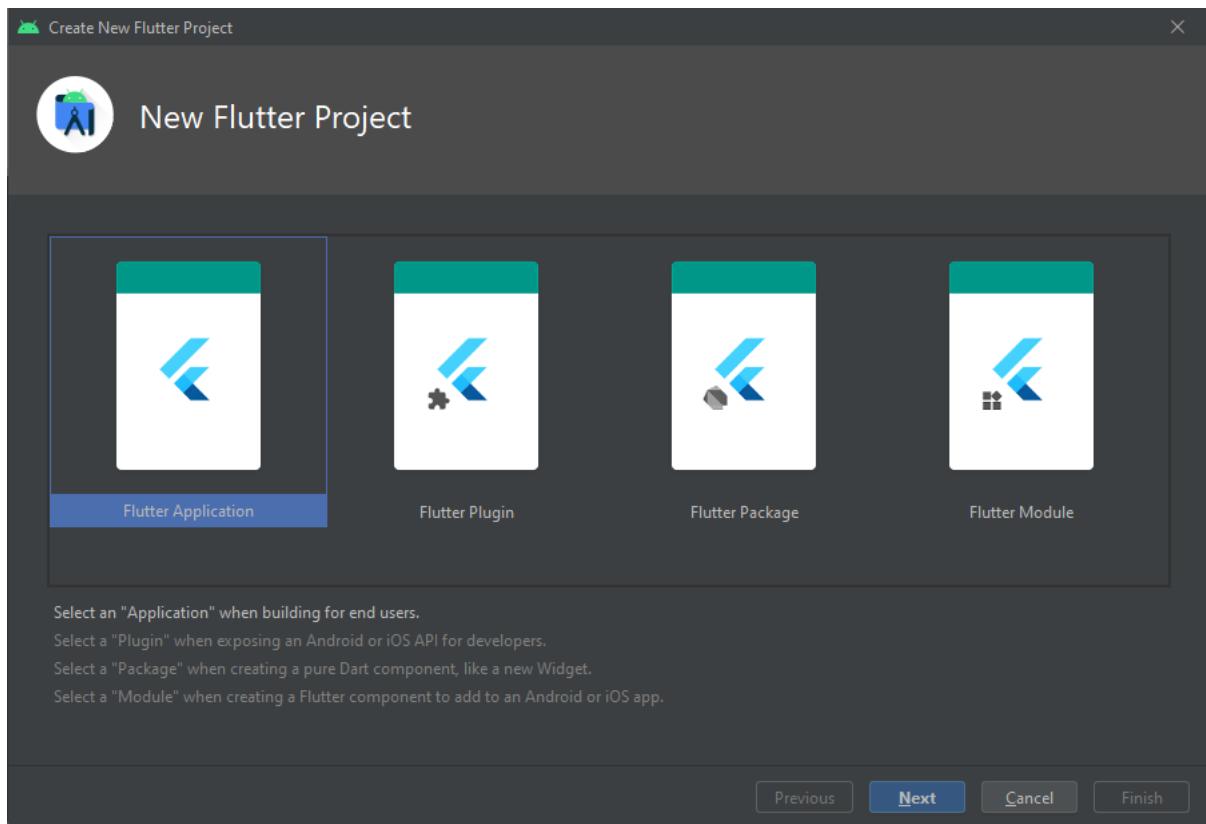
PERTEMUAN 2

1. Membuat dan menjalankan projek

Jalankan Android Studio kemudian pilih “Create New Flutter Project”



Kemudian pilih “Flutter Application” dan klik “Next”



Kemudian tentukan :

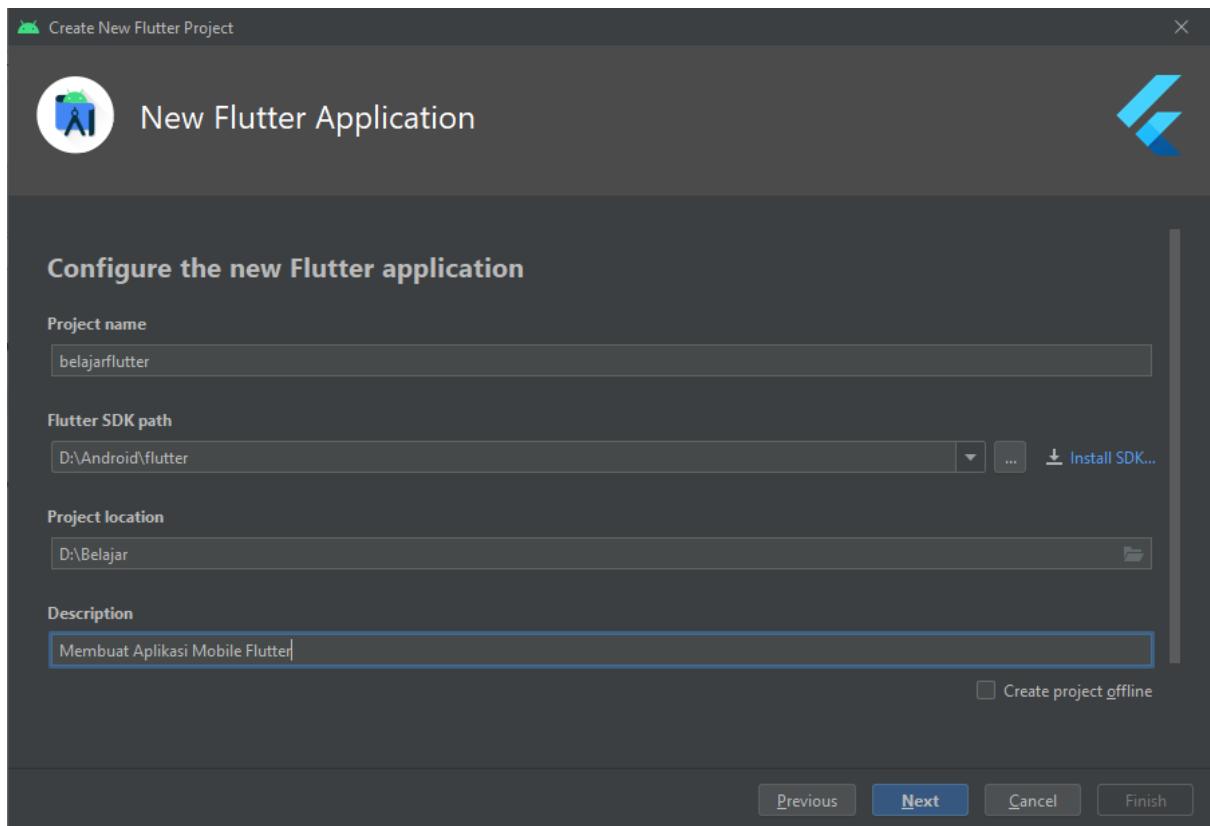
Project Name : belajarflutter

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

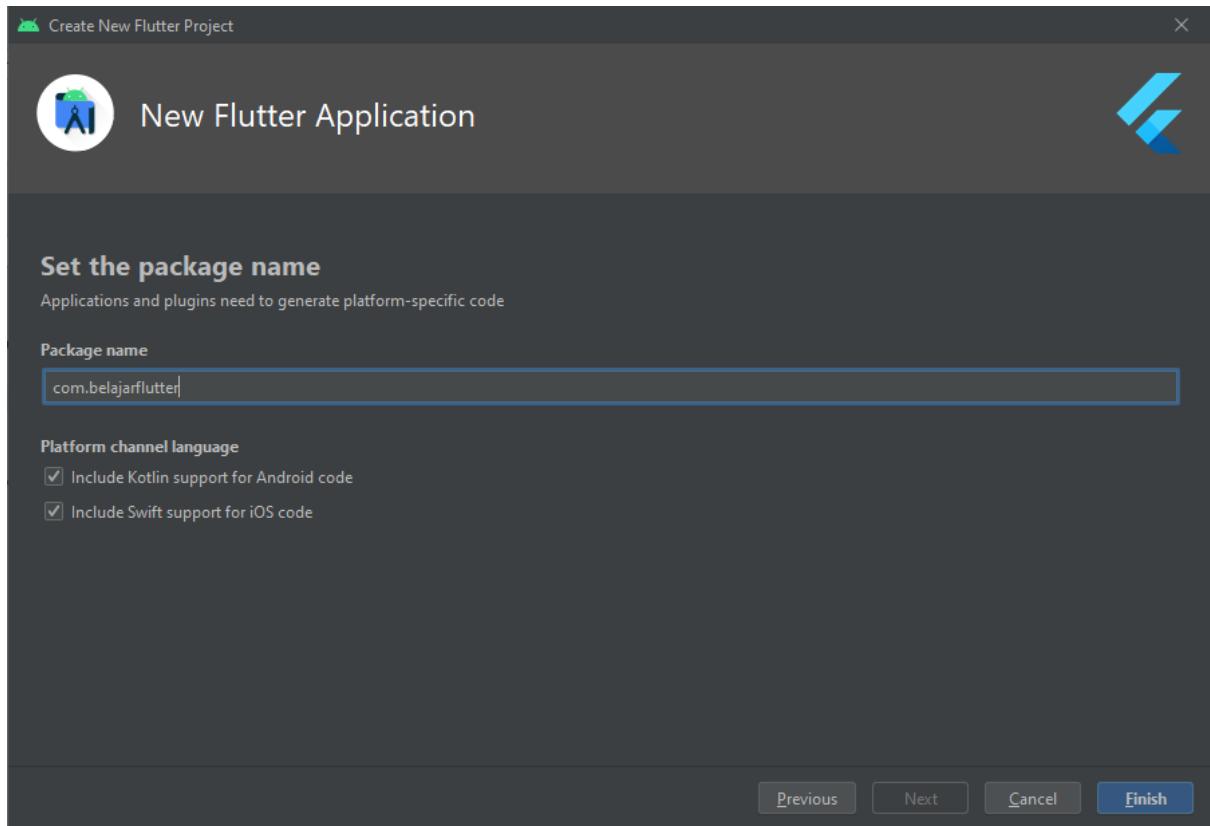
Project location : D:\Belajar (bebas memasukkan dimanapun)

Description : Membuat Aplikasi Mobile Flutter (bebas)

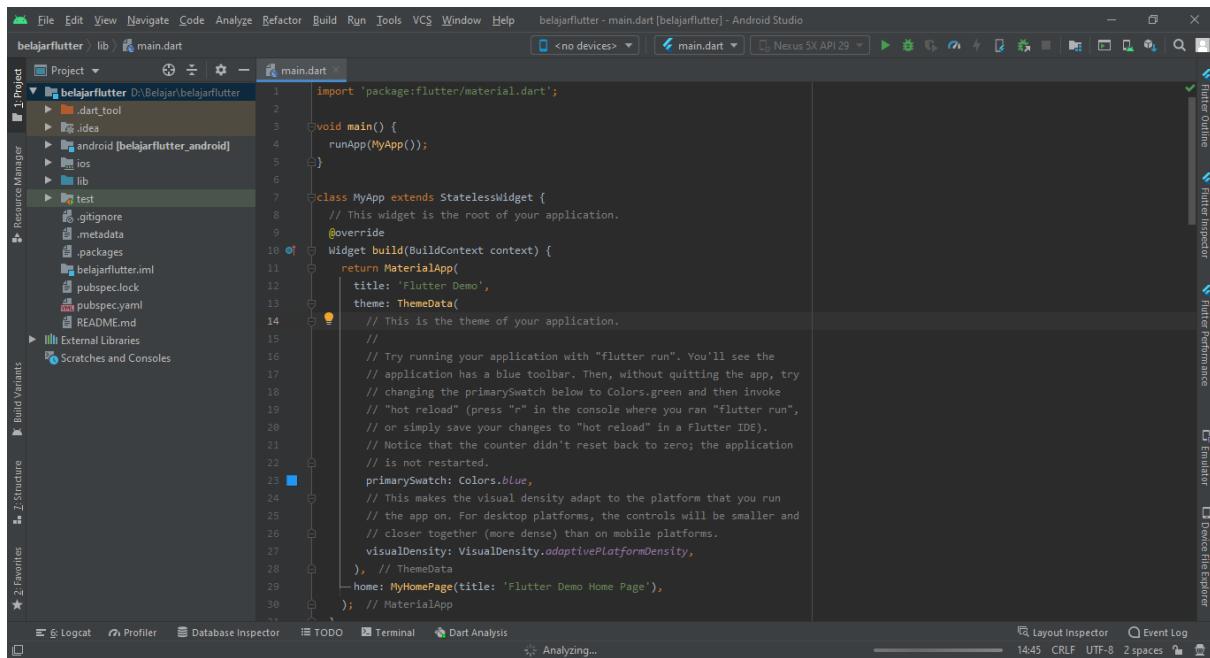
Kemudian klik tombol "Next"



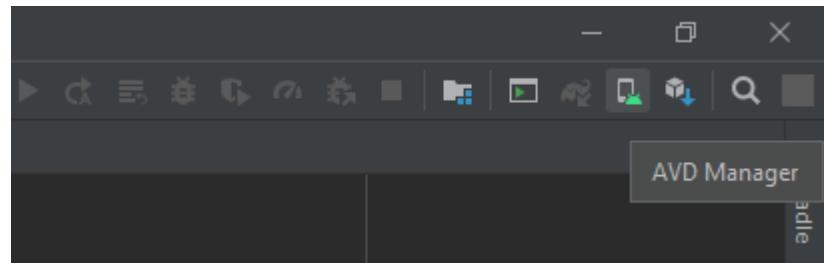
Kemudian klik tombol “Finish”. Pastikan perangkat terhubung ke internet, karena diperlukan untuk mengunduh projek yang dibuat



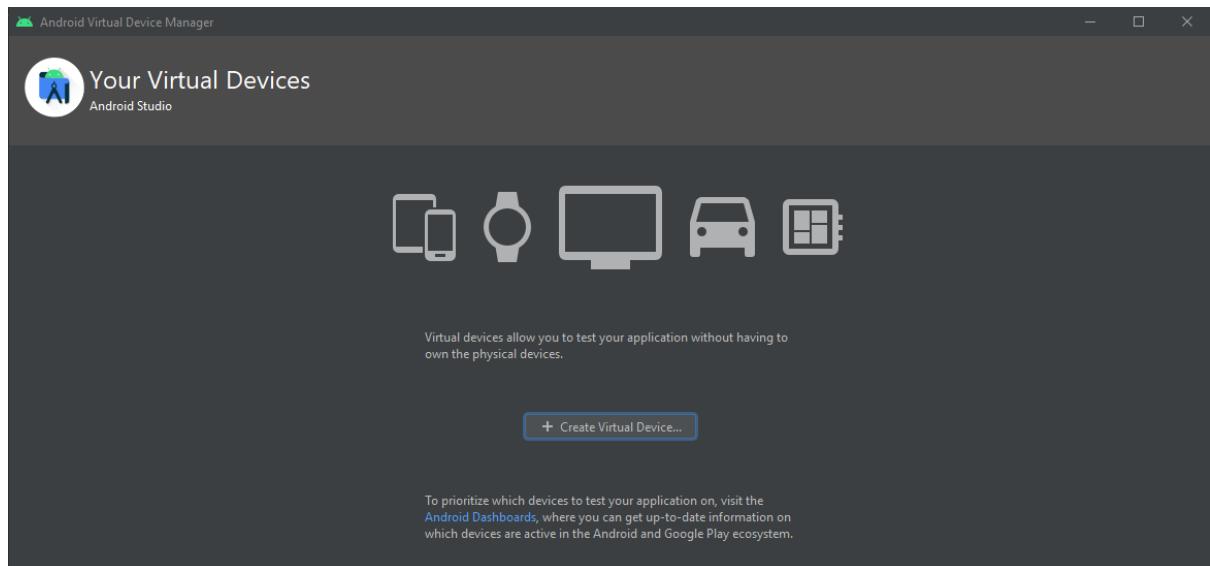
Setelah selesai akan muncul projek yang telah dibuat



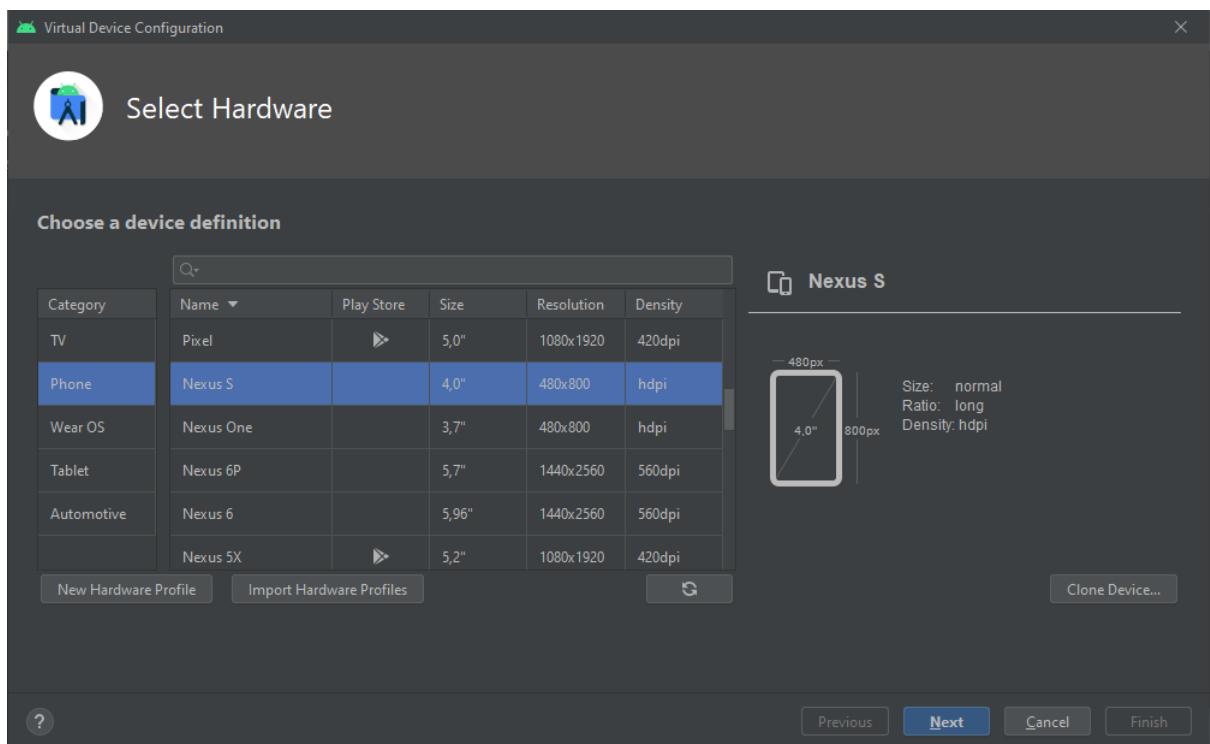
Untuk menjalankan projek, kita memerlukan emulator android. Pertama kita akan membuat emulator android dengan cara klik "AVD Manager" pada pojok kiri atas



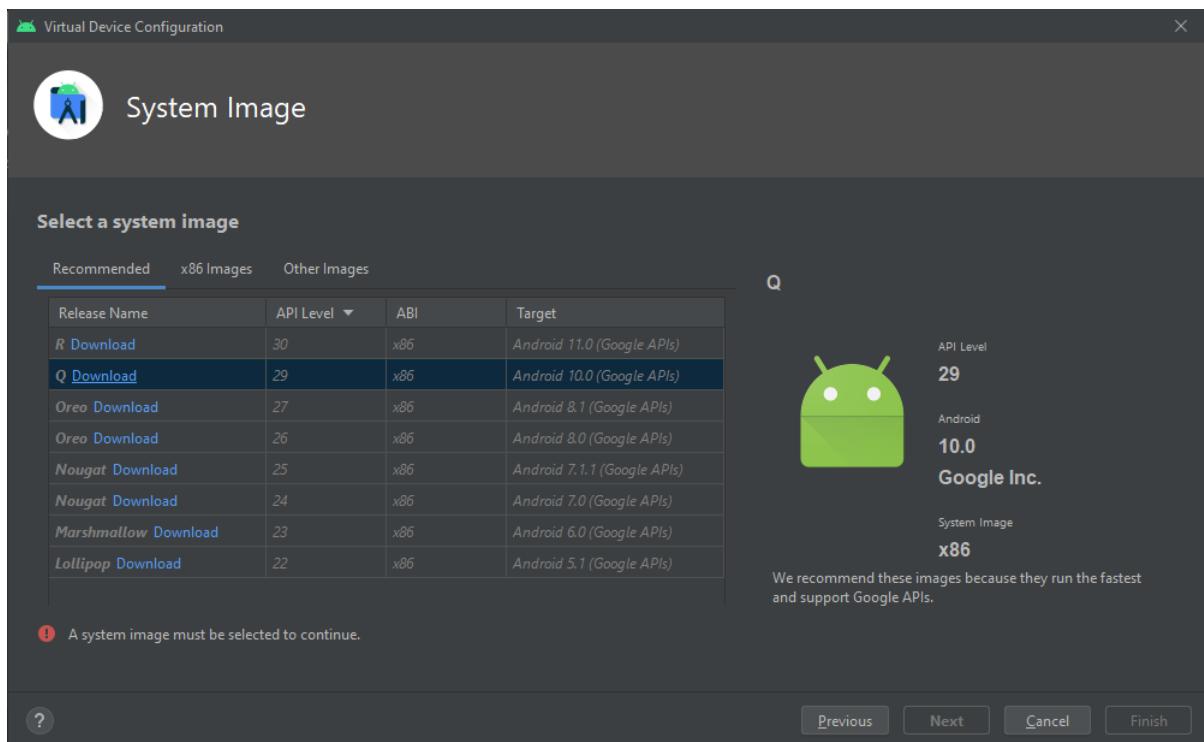
Kemudian klik tombol “Create Virtual Device”



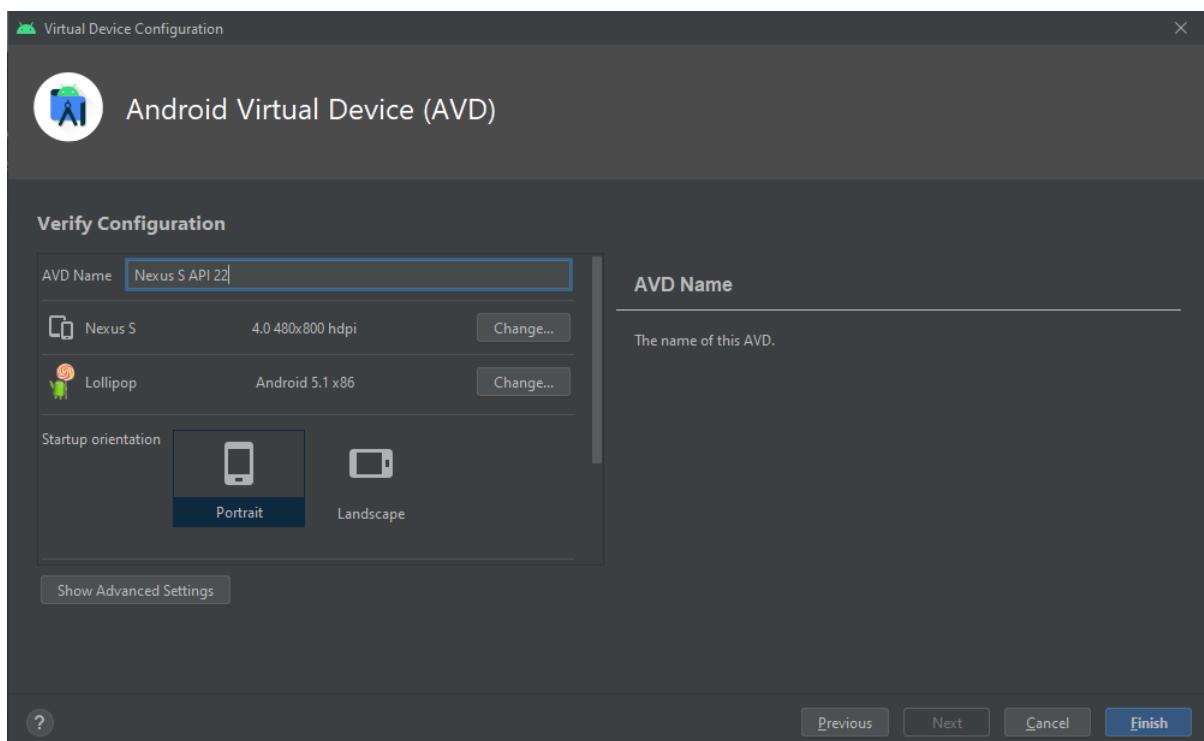
Pilih salah satu device yang dinginkan, misalnya “Nexus S” kemudian klik tombol “Next”



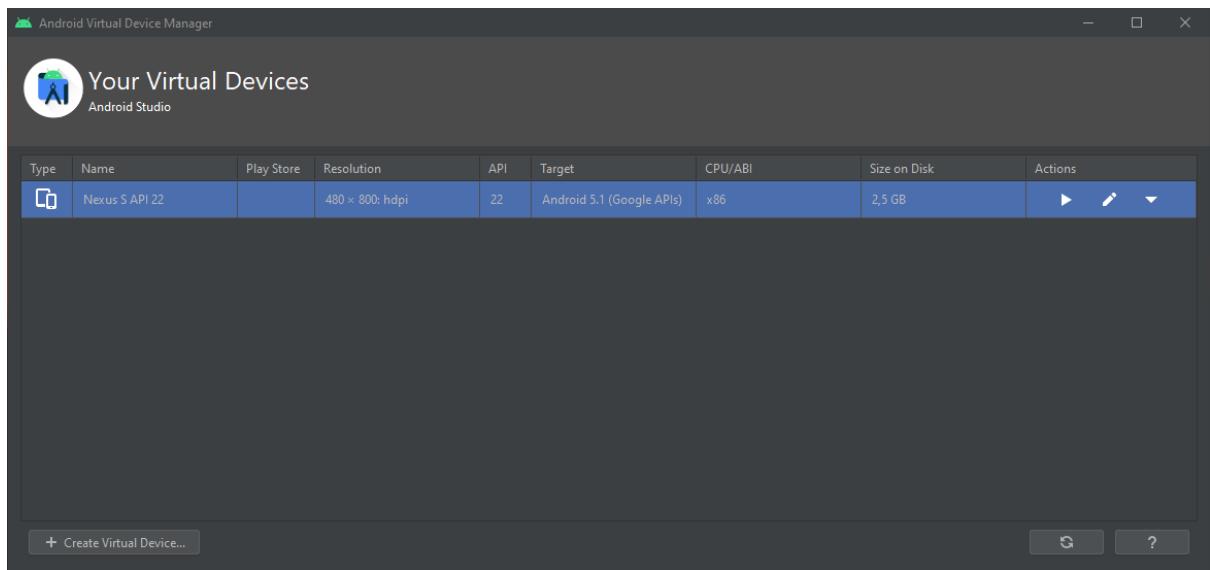
Kemudian kita akan memilih Versi Sistem Operasi Android, dalam hal ini misalnya "Q", jika belum ada maka kita akan diminta untuk mengunduh terlebih dahulu.



Setelah itu klik Finish



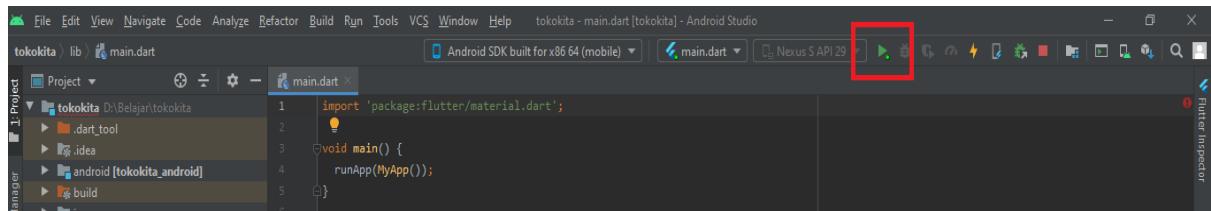
Untuk Menjalankan Emulator, klik logo play



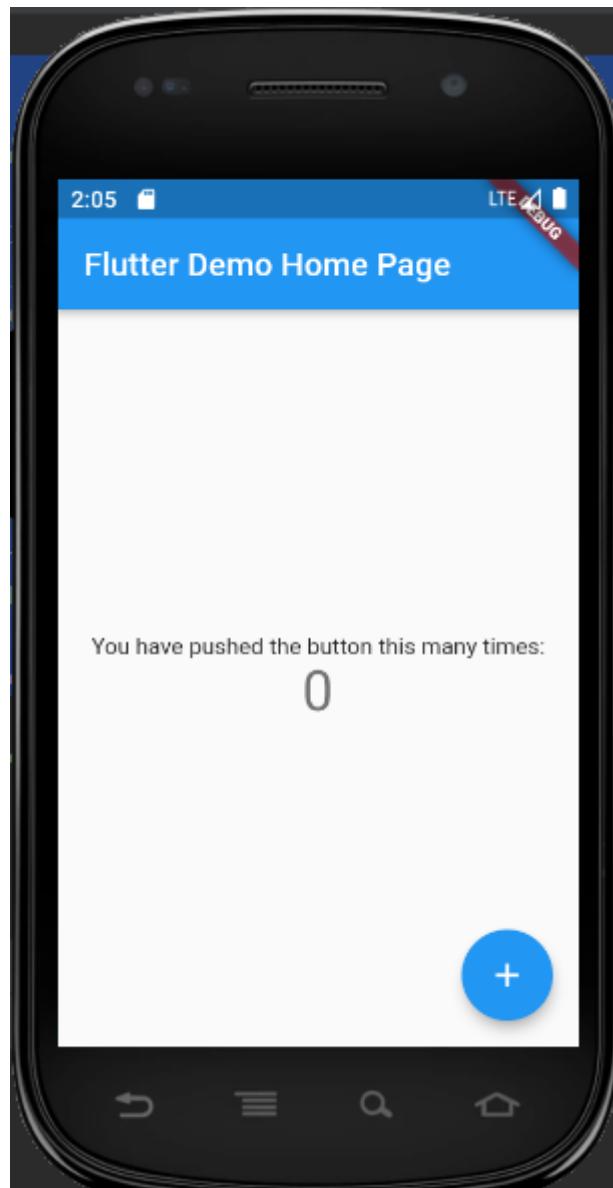
Kemudian akan muncul emulator android seperti berikut:



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



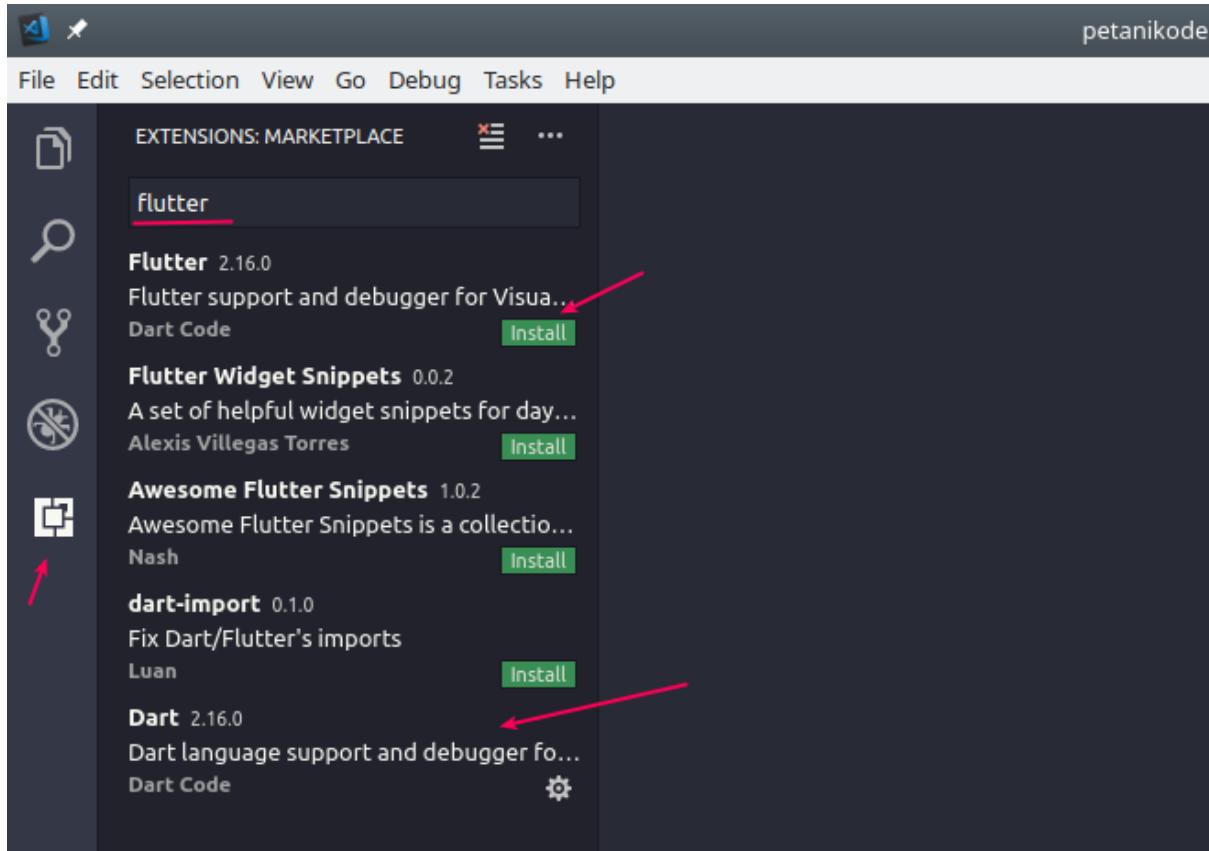
Pada emulator akan tampil hasil projek seperti berikut



2. Membuat dan menjalankan projek dengan VSCode dan Handphone Android

a. Instalasi VSCode sebagai alternatif editor

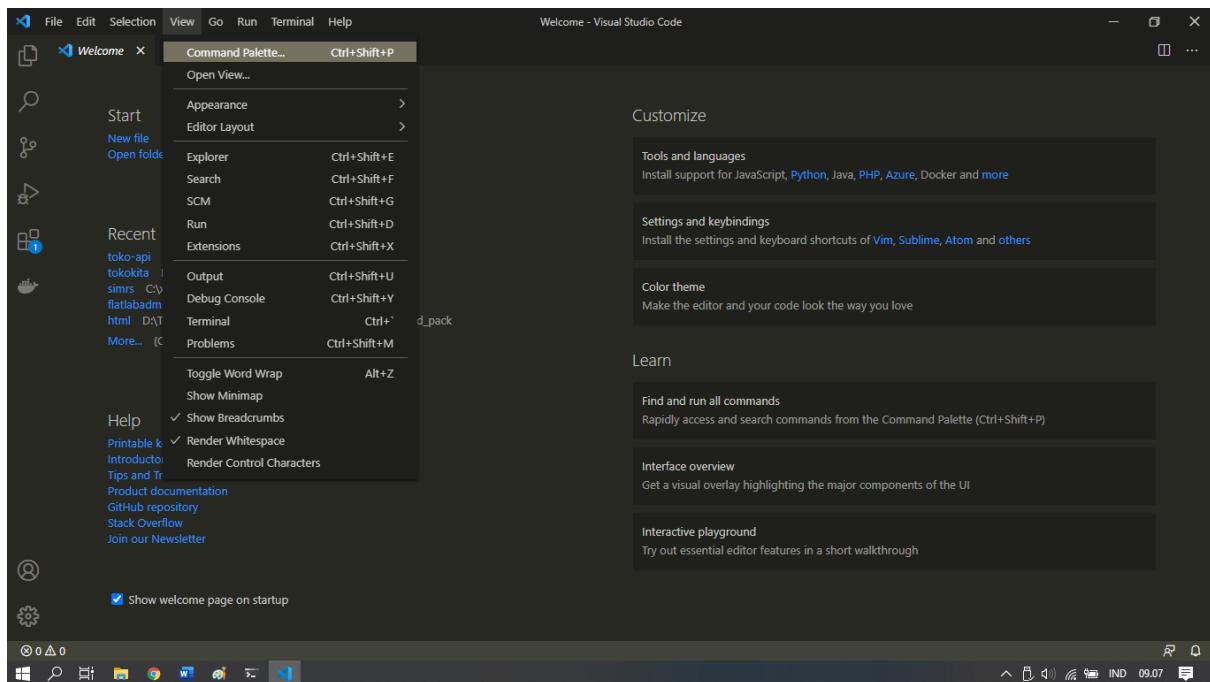
Unduh VSCode pada laman <https://code.visualstudio.com/download> kemudian install. Agar flutter dapat digunakan pada VSCode, perlu diinstall beberapa extension flutter yang dibutuhkan.



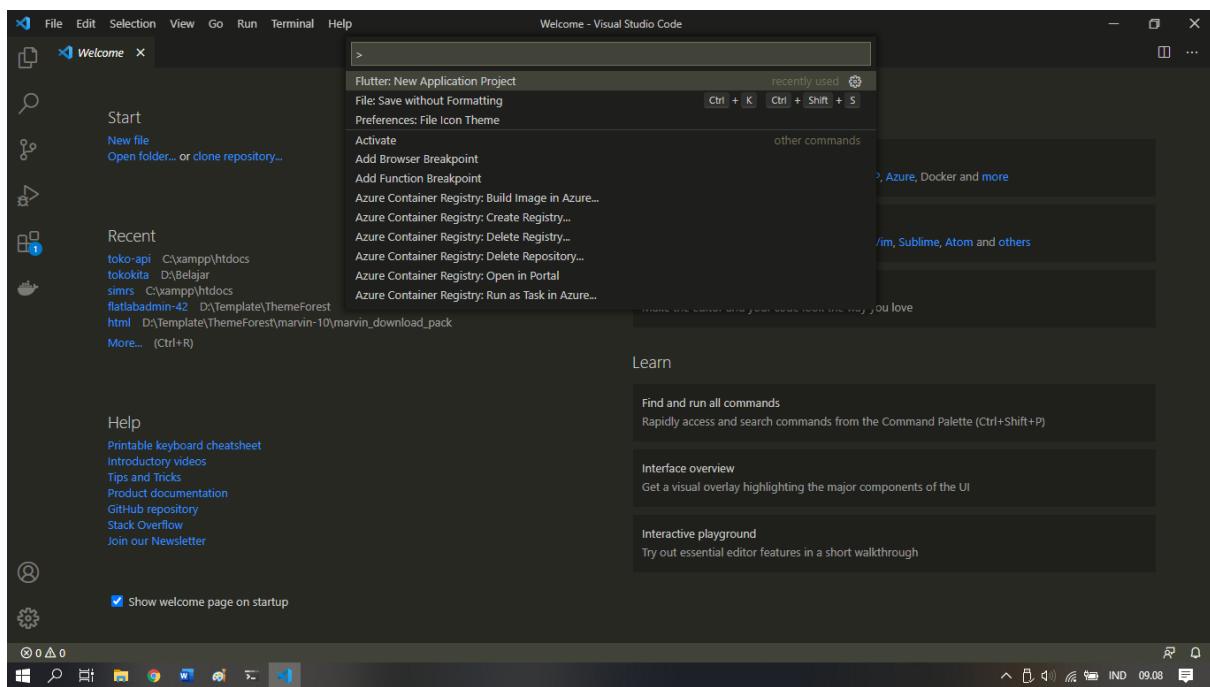
Setelah itu restart/tutup VSCode

b. Membuat projek flutter dengan VSCode

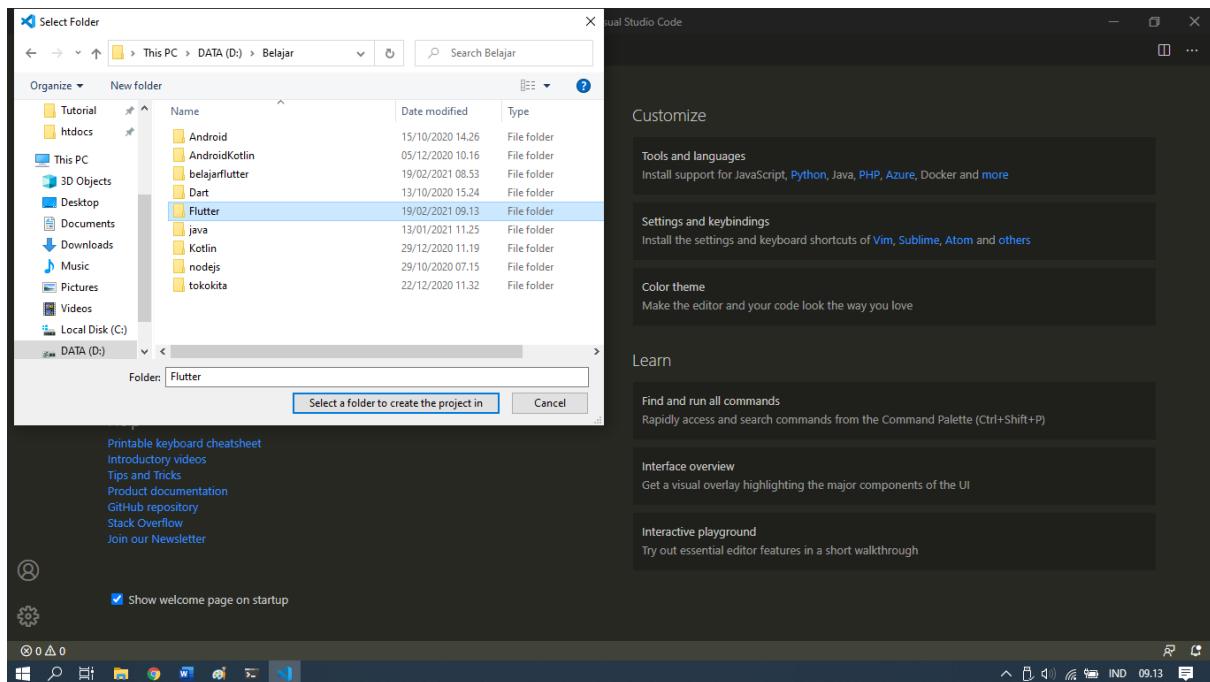
Jalankan VSCode, pada menubar pilih **view -> command Palette...** atau dapat juga dengan shortcut **Ctrl + Shift + P**



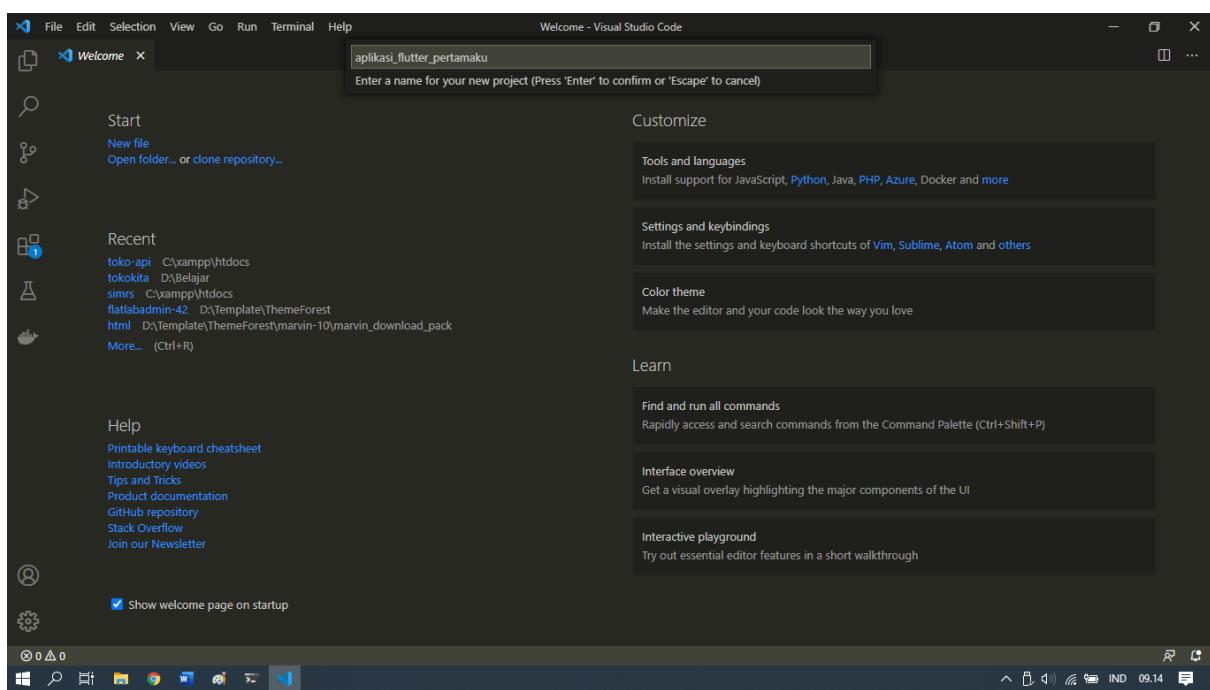
Kemudian ketikkan **flutter** dan pilih **Flutter: New Application Project**



Pilih folder tempat projek tersebut



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



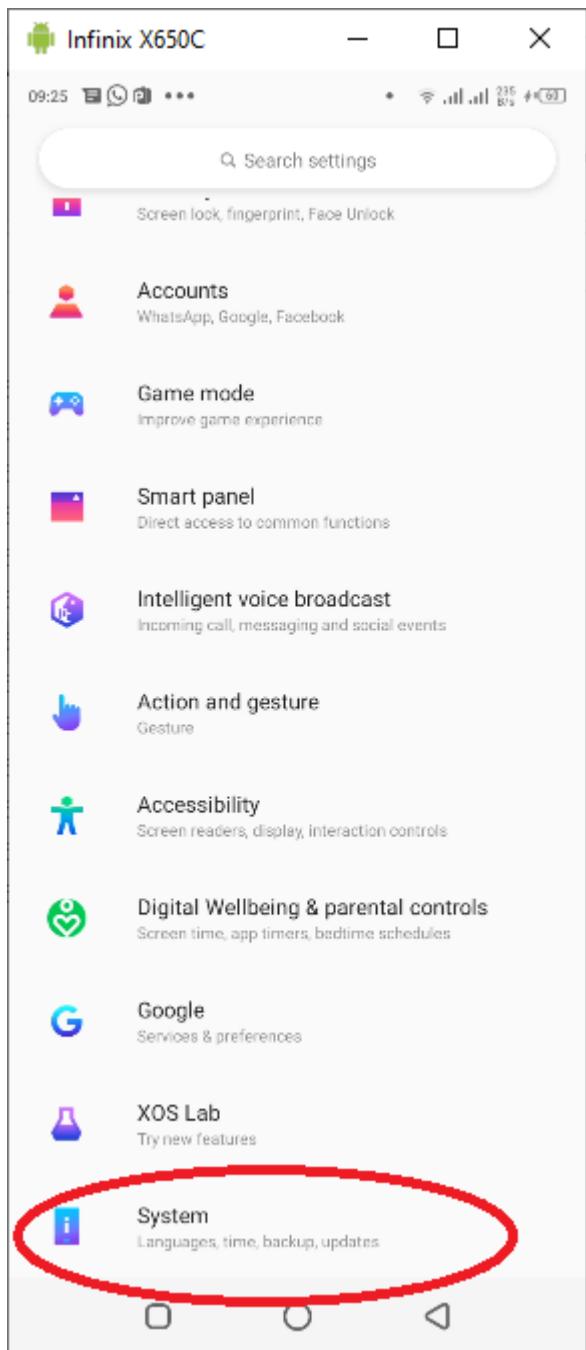
Kemudian tekan **Enter** dan tunggu hingga proses unduhan selesai

```
main.dart - aplikasi_flutter_pertamaku - Visual Studio Code
File Edit Selection View Go Run Terminal Help
main.dart x
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 }
```

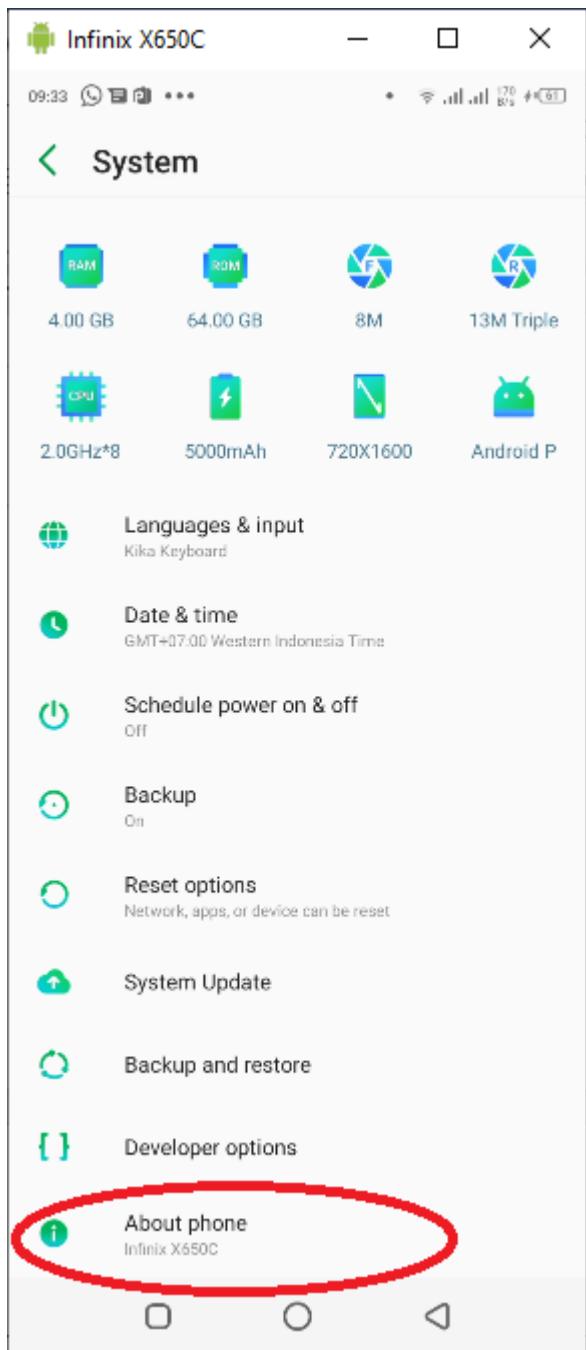
c. Menjalankan aplikasi dengan Handphone Android

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

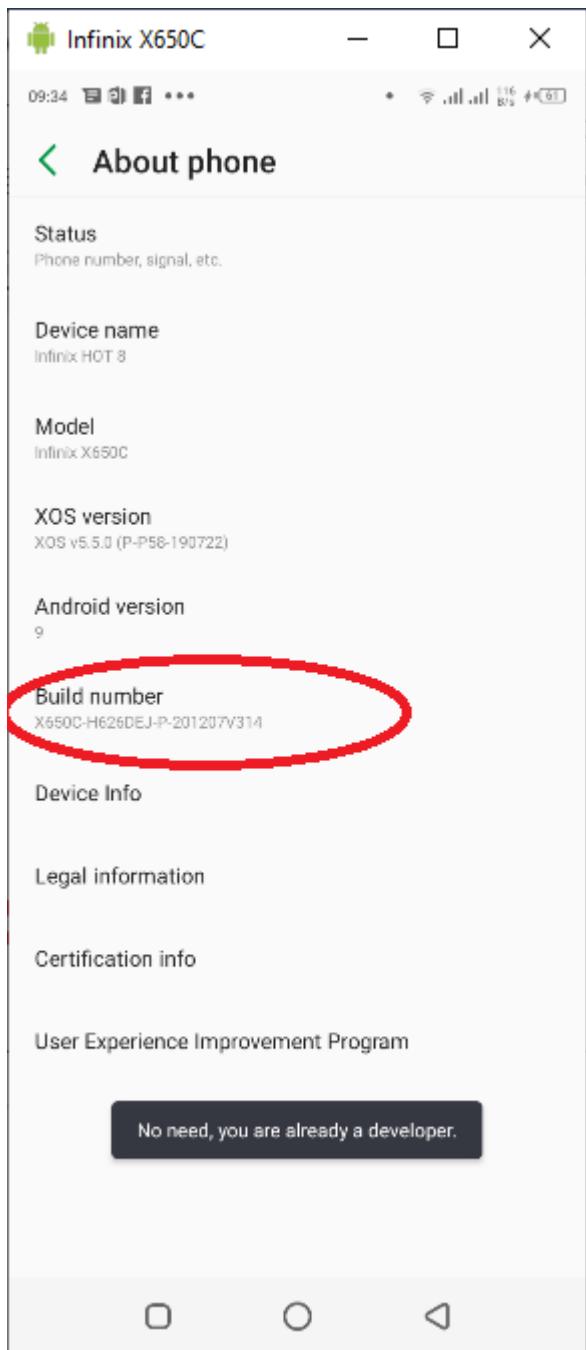
Untuk menggunakan android device secara langsung, pertama aktifkan dulu mode developer dengan cara buka **Setting** kemudian pilih **System** kemudian pilih **About Phone**, untuk masing-masing device mungkin terdapat perbedaan untuk lokasi **About Phone** ada pula yang berada pada **Additional Setting**



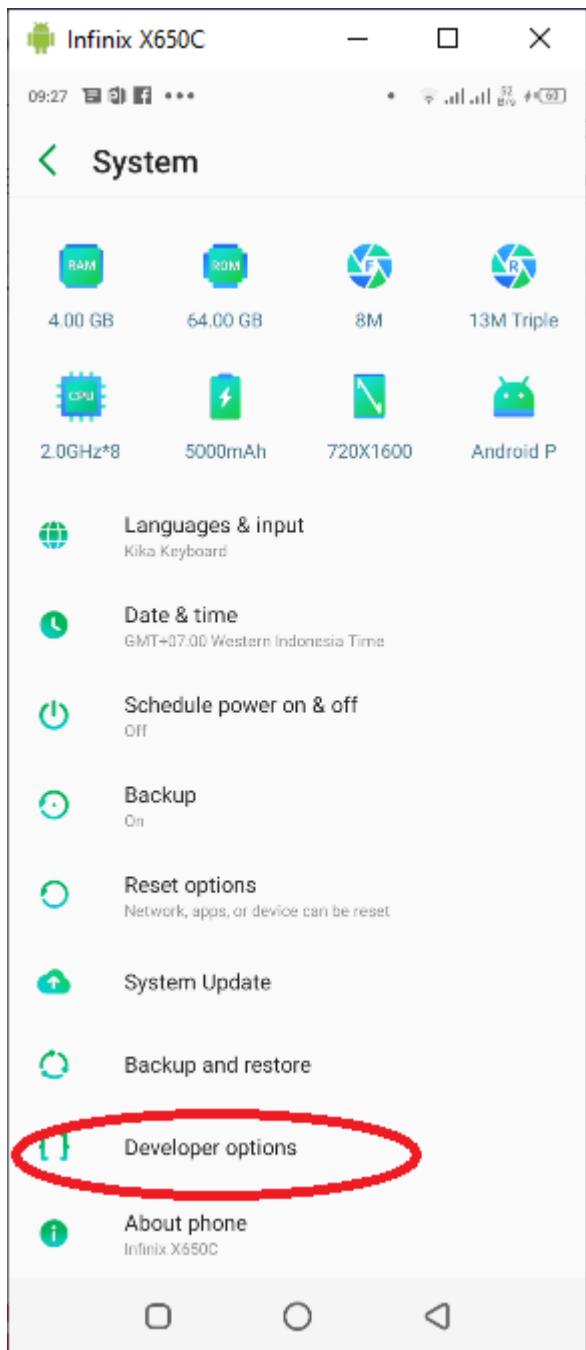
Kemudian pilih **About Phone**



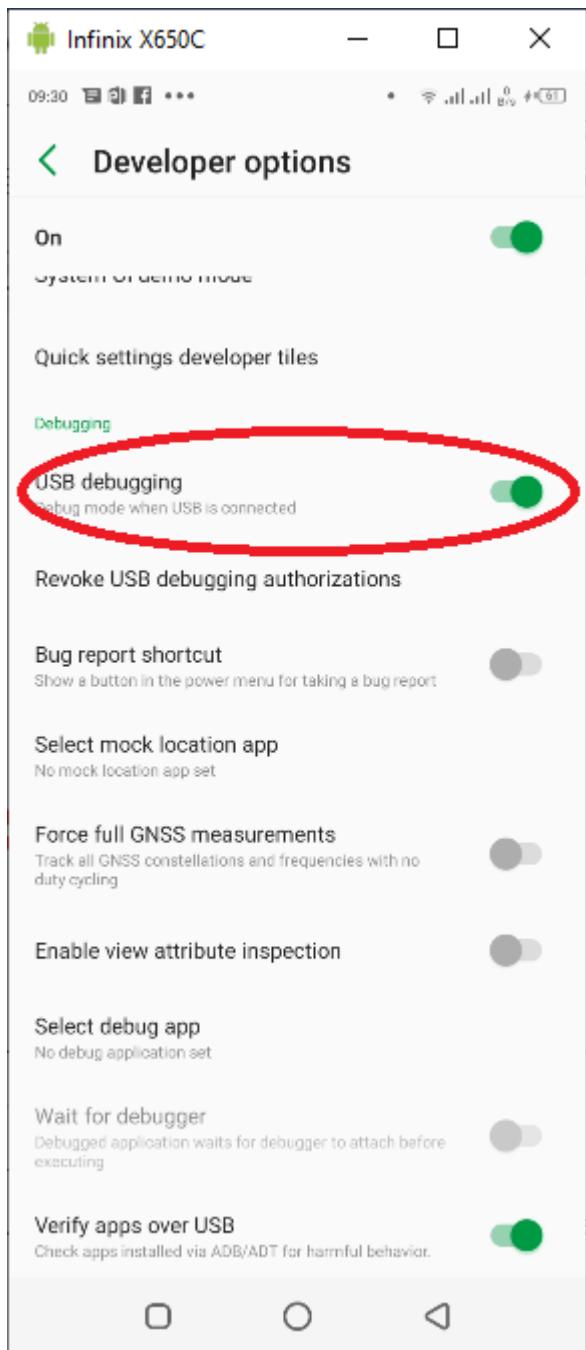
Kemudian ketuk **Build number** beberapa kali, namun ini juga berbeda untuk beberapa versi misalnya untuk Xiaomi dengan mengetuk **MIUI Version** beberapa kali



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



Kemudian aktifkan USB Debugging



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

```

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

```

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 1-Project belajarflutter D:\Belajar\belajarflutter
  - dart_tool
  - .idea
  - android [belajarflutter_android]
  - ios
  - lib
  - test
    - gitignore
    - .metadata
    - packages
    - belajarflutter.iml
    - pubspec.lock
    - pubspec.yaml
    - README.md
External Libraries Scratches and Consoles
Build Variants Z Structure
  ★ 2 Favorites
Resource Manager
Event Log Layout Inspector
1229 CRLF UTF-8 2 spaces

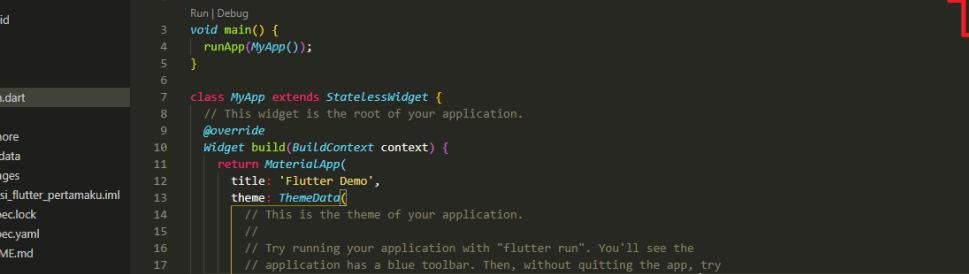
```

```

import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
        //
        // This makes the visual density adapt to the platform that you run
        // the app on. For desktop platforms, the controls will be smaller and
        // closer together (more dense) than on mobile platforms.
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ), // ThemeData
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
}

```

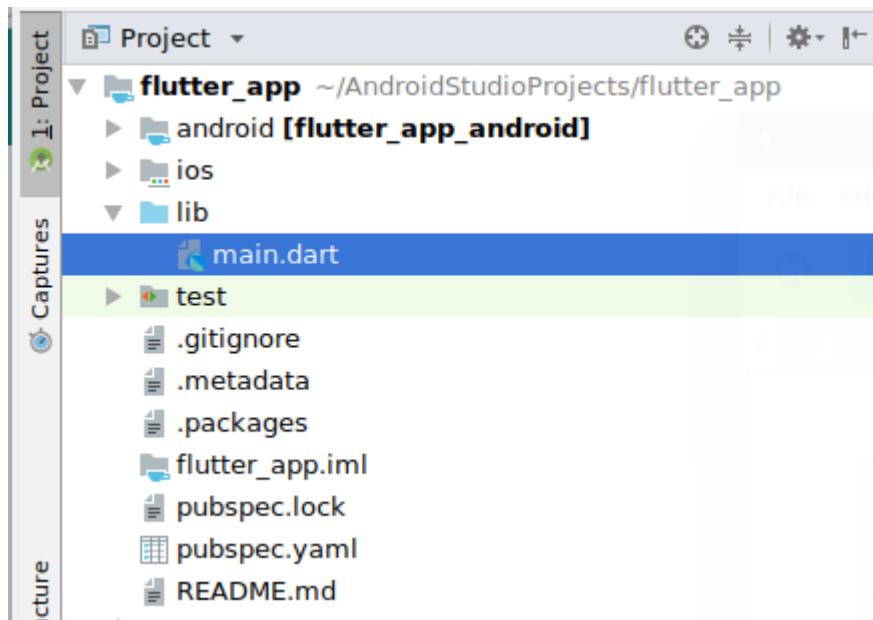
Agar laptop bekerja lebih ringan dapat digunakan Text Editor VSCode dan menjalankan projek langsung menggunakan Handphone Android. Untuk menjalankan projek melalui VSCode dengan klik logo play pada bagian pojok kanan atas



```
lib > lib/main.dart > MyApp > main.dart
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 }
```

3. Struktur Folder Flutter

Adapun struktur folder Projek flutter adalah sebagai berikut:



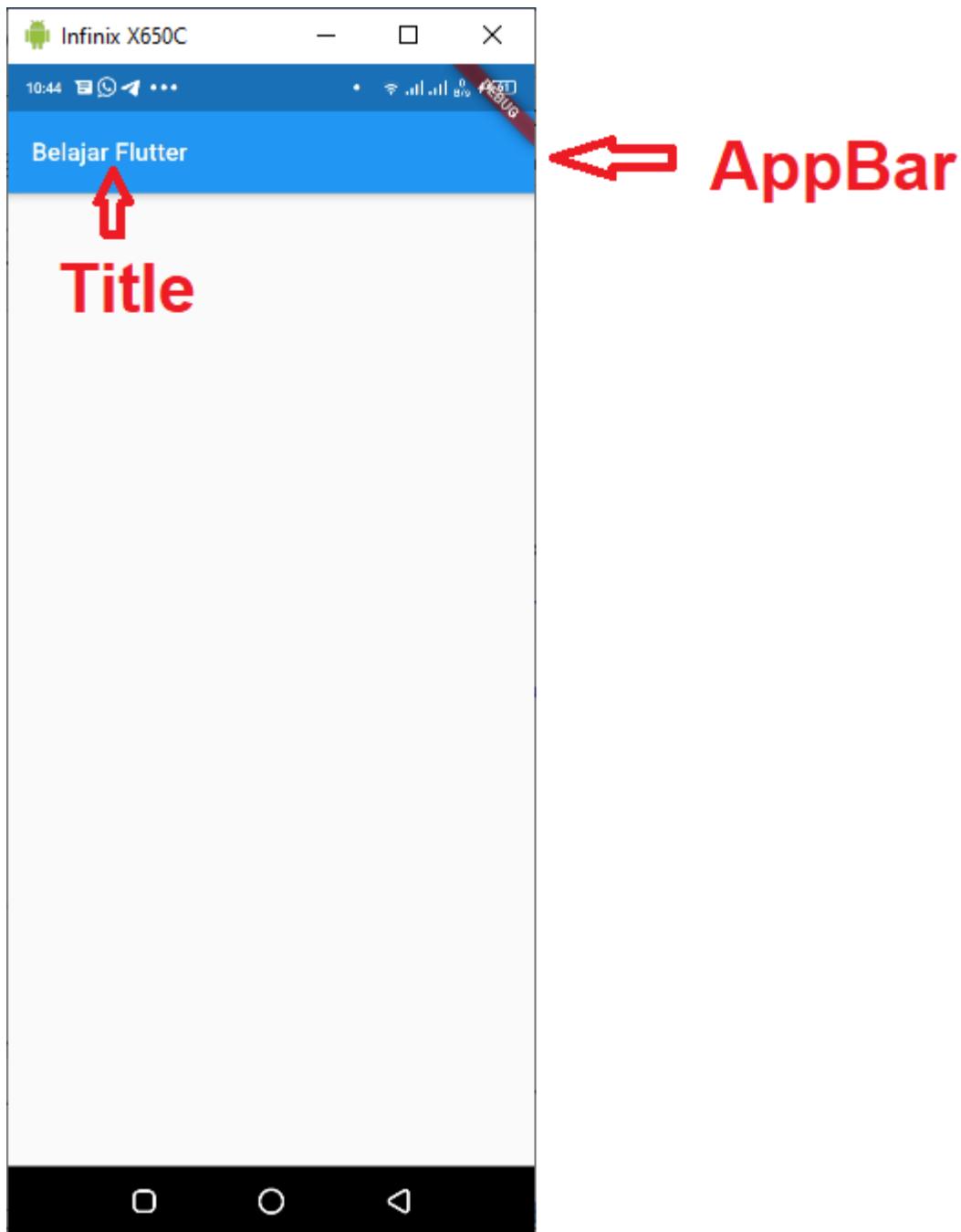
- ② **android** berisi source code untuk aplikasi android;
- ② **ios** berisi source code untuk aplikasi iOS;
- ② **lib** berisi source code Dart, di sini kita akan menulis kode aplikasi;
- ② **test** berisi source code Dart untuk testing aplikasi;
- ② **.gitignore** adalah file [Git](#);
- ② **.metadata** 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.

a. 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(MyApp());
5. }
6.
7. class MyApp extends StatelessWidget {
8.   @override
9.   Widget build(BuildContext context) {
10.     return MaterialApp(
11.       title: 'Aplikasi Flutter Pertama',
12.       home: Scaffold(
13.         appBar: AppBar(
14.           title: Text('Belajar Flutter'),
15.         ),
16.       ),
17.     );
18.   }
19. }
```

Ketika dijalankan akan menghasilkan

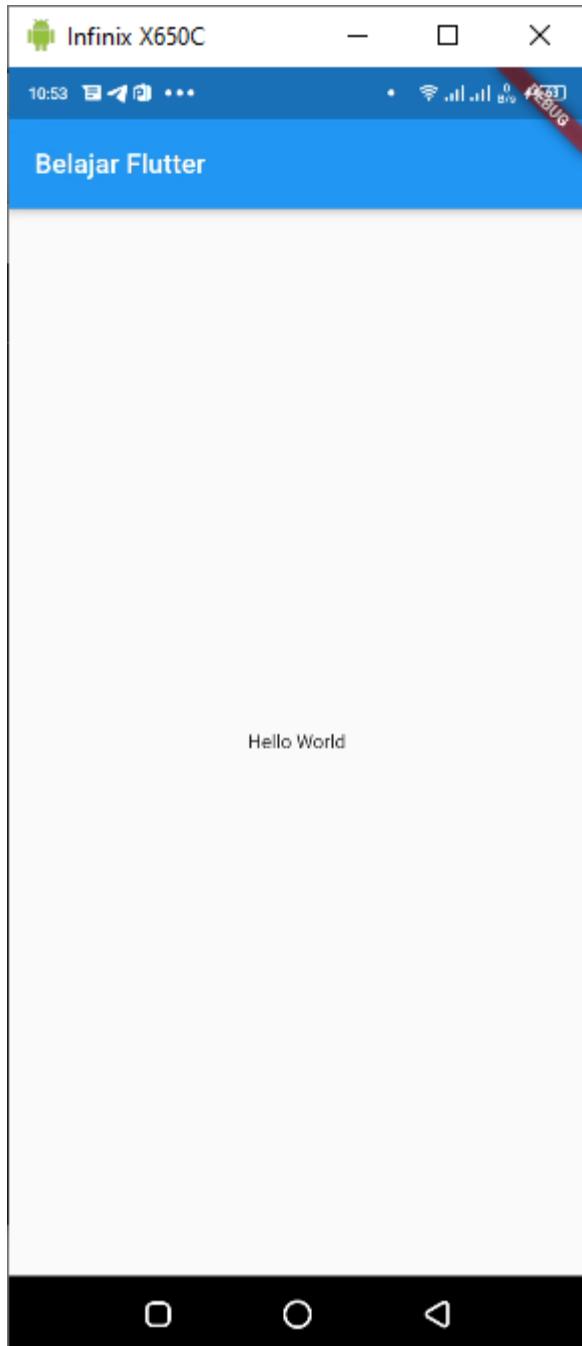


Kemudian untuk menambahkan tampilan dibagian dalam (body), pada fungsi Scaffold terdapat parameter **body**. Silahkan modifikasi **main.dart** menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2.
3. void main() {
4.   runApp(MyApp());
5. }
6.
7. class MyApp extends StatelessWidget {
8.   @override
9.   Widget build(BuildContext context) {
10.    return MaterialApp(
11.      title: 'Aplikasi Flutter Pertama',
12.      home: Scaffold(
```

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

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**.

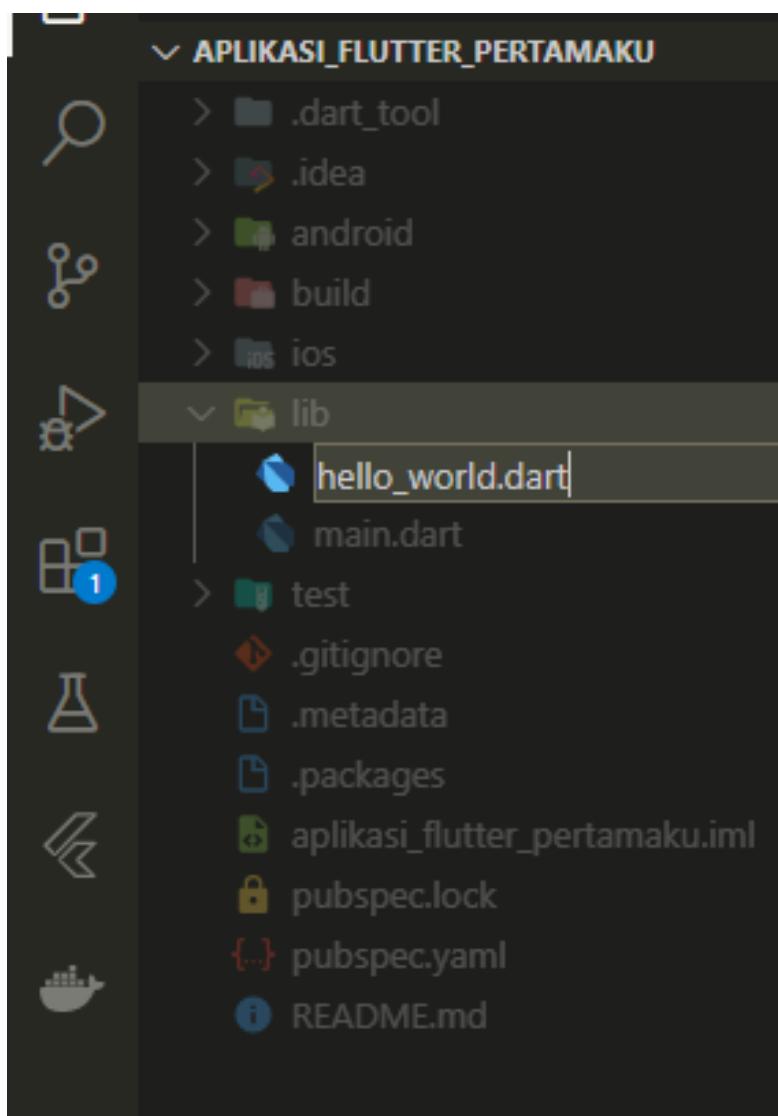
Ini adalah widget dasar.

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: Text('Belajar Flutter'),
9.       ),
10.      body: 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(MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   @override
10.  Widget build(BuildContext context) {
11.    return MaterialApp(
12.      title: 'Aplikasi Flutter Pertama',
13.      home: HelloWorld(),
14.    );
15. }
16. 
```

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

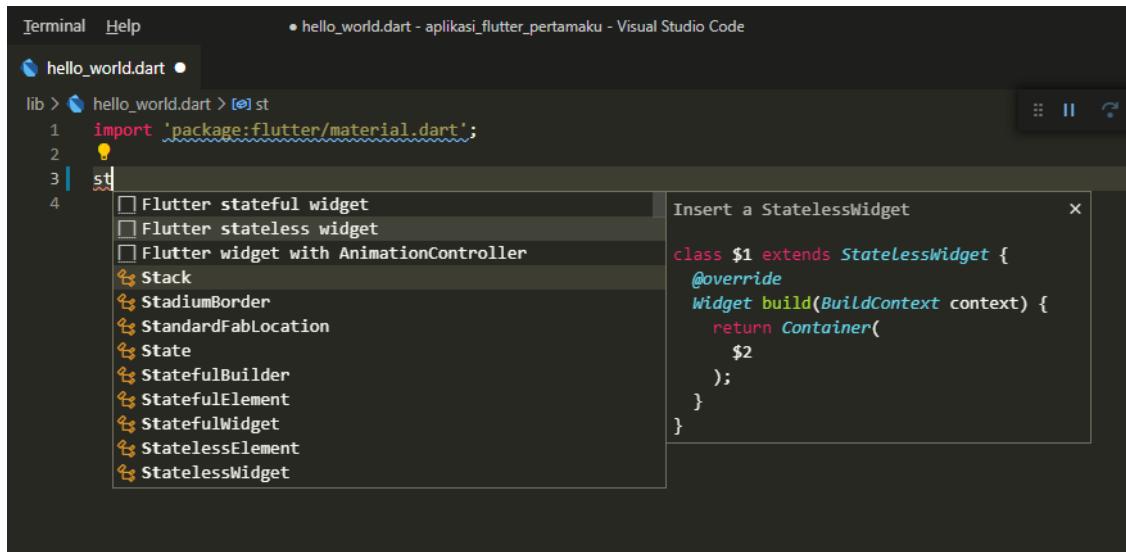
Jika kita perhatikan pada bagian **body**, terdapat Widget **Center** kemudian didalam Widget **Center** tersebut terdapat parameter **child** untuk meletakkan Widget lain didalam widget tersebut, dalam hal ini adalah Widget **Text**

```

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

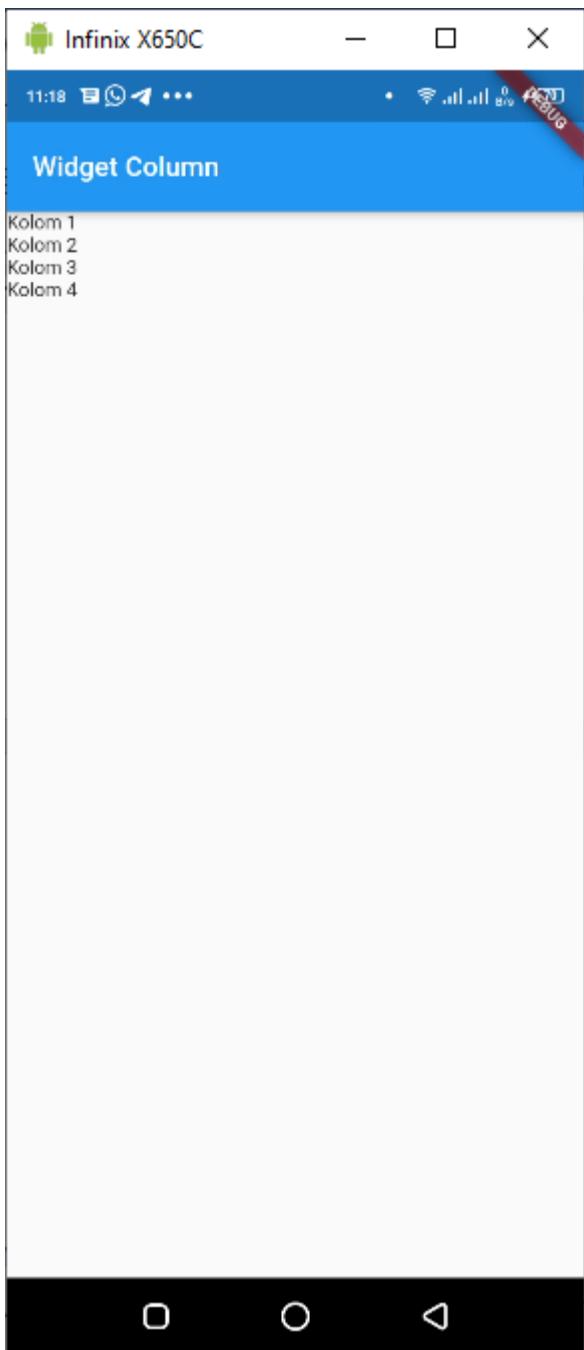


b. Membuat Widget Column

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.   @override
5.   Widget build(BuildContext context) {
6.     return Scaffold(
7.       appBar: AppBar(
8.         title: Text('Widget Column'),
9.       ),
10.      body: Column(
11.        children: [
12.          Text('Kolom 1'),
13.          Text('Kolom 2'),
14.          Text('Kolom 3'),
15.          Text('Kolom 4'),
16.        ],
17.      ),
18.    );
19. }
20. }
```

Dan hasilnya akan menjadi seperti berikut



Column biasanya digunakan untuk membuat Form

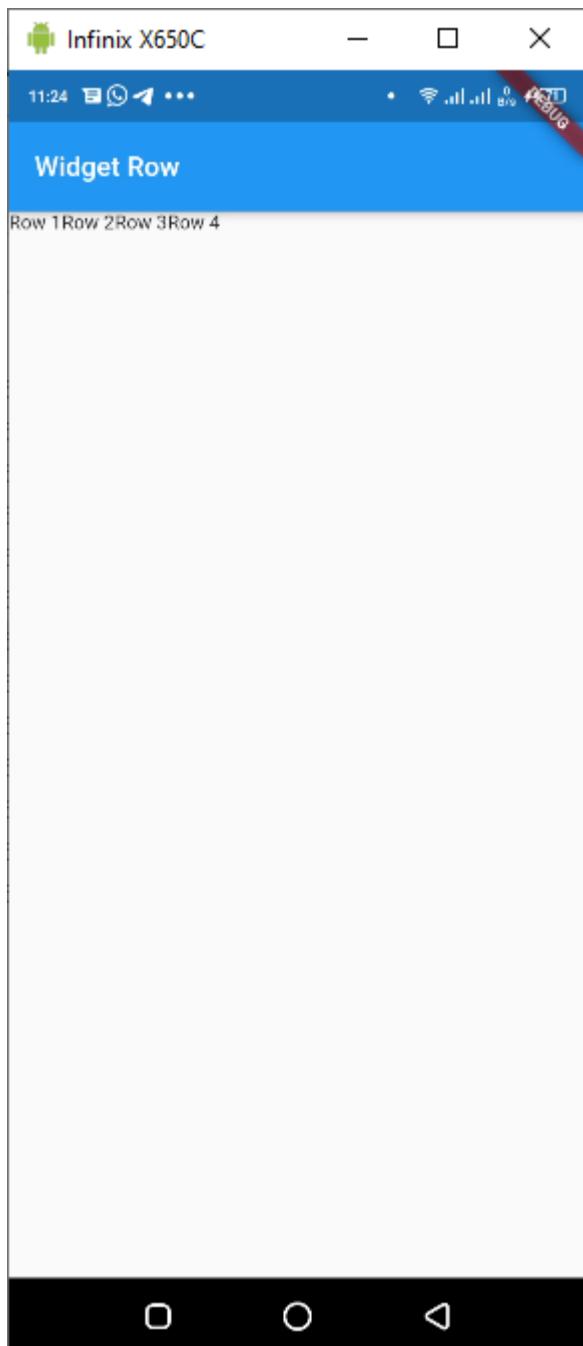
c. Membuat Widget Row

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.   @override
5.   Widget build(BuildContext context) {
6.     return Scaffold(
7.       appBar: AppBar(
8.         title: Text('Widget Row'),
```

```
9.      ),
10.     body: Row(
11.       children: [
12.         Text('Row 1'),
13.         Text('Row 2'),
14.         Text('Row 3'),
15.         Text('Row 4'),
16.       ],
17.     ),
18.   );
19. }
20. }
```

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



d. Mengenal StatelessWidget dan StatefulWidget

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

Contoh StatelessWidget :

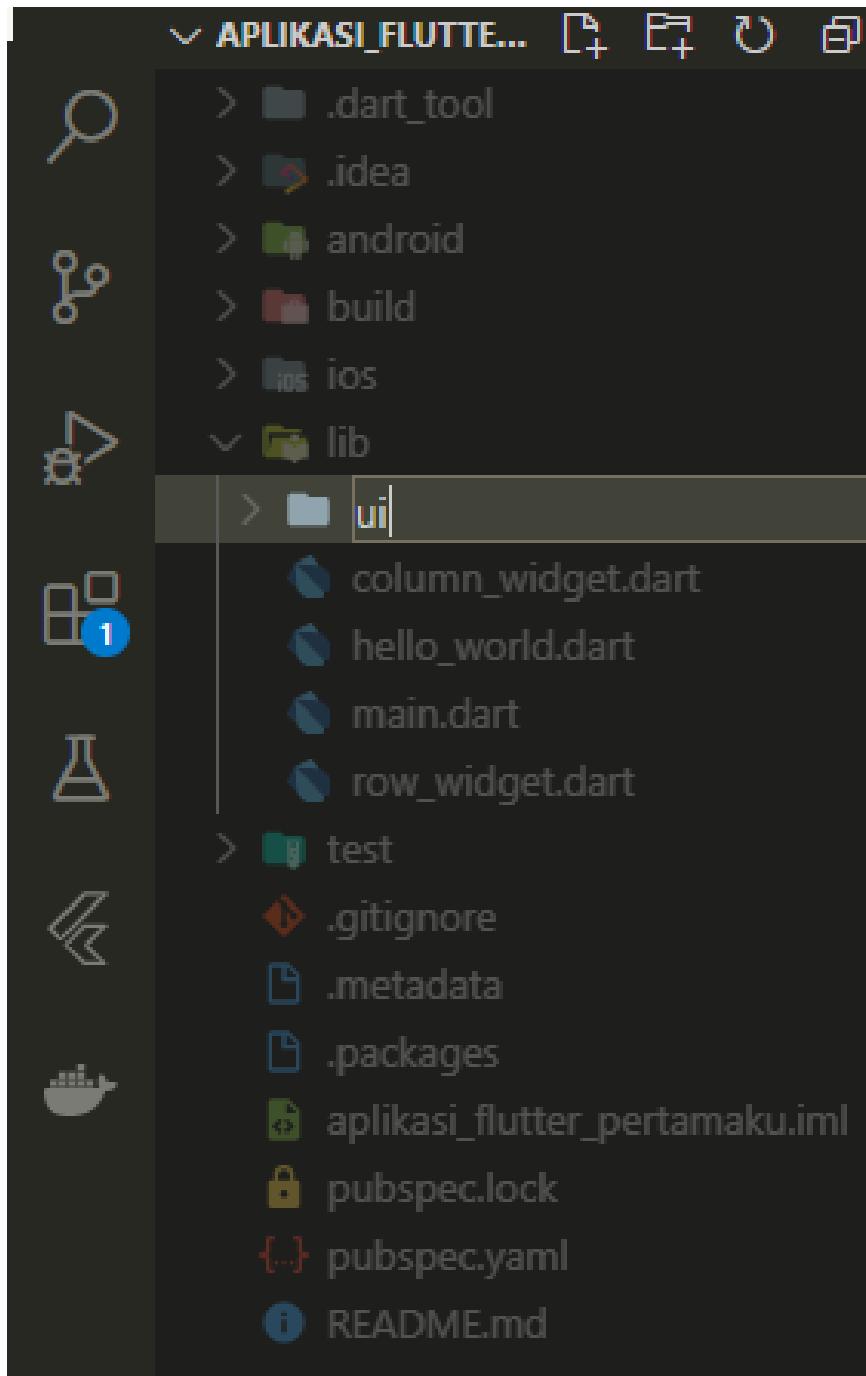
```
1. class HelloWorld extends StatelessWidget {  
2.   @override  
3.   Widget build(BuildContext context) {  
4.     return Container(  
5.     //  
6.   );  
7. }  
8. }
```

Contoh StatefulWidget

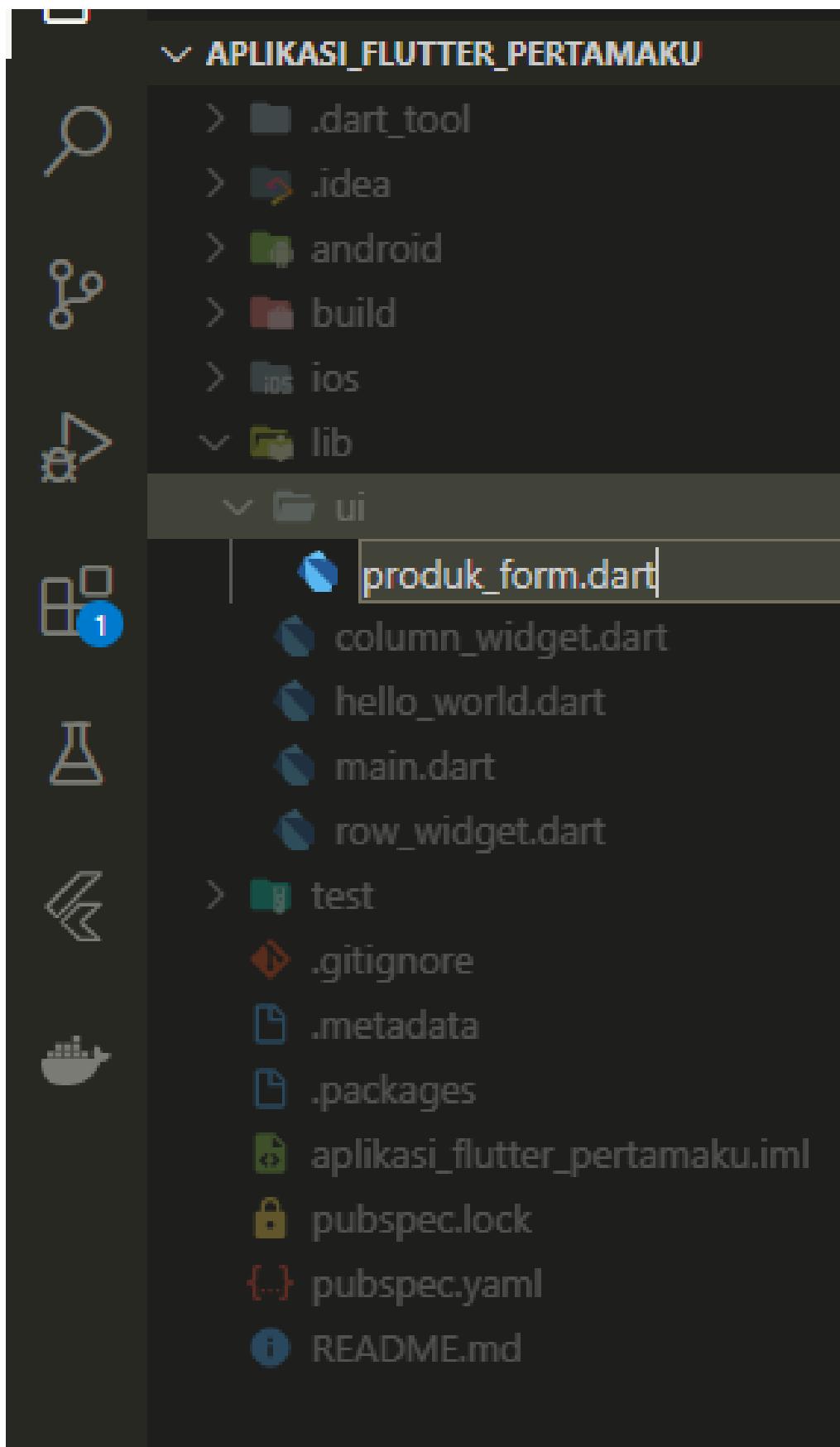
```
1. class HelloWorld extends StatefulWidget {  
2.   @override  
3.   _HelloWorldState createState() => _HelloWorldState();  
4. }  
5.  
6. class _HelloWorldState extends State<HelloWorld> {  
7.   @override  
8.   Widget build(BuildContext context) {  
9.     return Container(  
10.    //  
11.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

4. Tugas Pertemuan 2

1. Membuat Form dengan flutter dengan mengikuti Langkah-langkah seperti berikut ini:
 - a. Untuk membuat form dengan 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.   @override
5.   _ProdukFormState createState() => _ProdukFormState();
6. }
7.
8. class _ProdukFormState extends State<ProdukForm> {
9.   @override
10.  Widget build(BuildContext context) {
11.    return Scaffold(
12.      appBar: AppBar(
13.        title: Text('Form Produk'),
14.      ),
15.      body: SingleChildScrollView(
16.        child: Column(
17.          children: [
18.            TextField(
19.              decoration: InputDecoration(labelText: "Kode Produk"),
20.            ),
21.            TextField(
22.              decoration: InputDecoration(labelText: "Nama Produk"),
23.            ),
24.            TextField(
25.              decoration: InputDecoration(labelText: "Harga"),
26.            ),
27.            RaisedButton(
28.              child: Text('Simpan'),
29.              onPressed: () {},
30.            )
31.          ],
32.        ),
33.      ),
34.    );
35.  }
36. }
```

- b. Ubah pada `main.dart` dengan memanggil class `ProdukForm`, sehingga hasilnya akan menjadi



PERTEMUAN 3

1. Mengenal Widget dan Fungsinya
2. 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.   @override
5.   _ProdukFormState createState() => _ProdukFormState();
6. }
7.
8. class _ProdukFormState extends State<ProdukForm> {
9.   @override
10.  Widget build(BuildContext context) {
11.    return Scaffold(
12.      appBar: AppBar(
13.        title: Text('Form Produk'),
14.      ),
15.      body: SingleChildScrollView(
16.        child: Column(
17.          children: [
18.            _textboxKodeProduk(),
19.            _textboxNamaProduk(),
20.            _textboxHargaProduk(),
21.            _tombolSimpan()
22.          ],
23.        ),
24.      ),
25.    );
26.  }
27.
28. _textboxKodeProduk() {
29.   return TextField(
30.     decoration: InputDecoration(labelText: "Kode Produk"),
31.   );
32. }
33.
34. _textboxNamaProduk() {
35.   return TextField(
36.     decoration: InputDecoration(labelText: "Nama Produk"),
37.   );
38. }
39.
40. _textboxHargaProduk() {
41.   return TextField(
42.     decoration: InputDecoration(labelText: "Harga"),
43.   );
44. }
45.
46. _tombolSimpan() {
47.   return RaisedButton(
48.     child: Text('Simpan'),
49.     onPressed: () {},
50.   );
51. }
```

```
51. }
52. }
```

3. 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.   ProdukDetail({this.kodeProduk, this.namaProduk, this.harga});
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: Text("Detail Produk"),
20.       ),
21.       body: Column(
22.         children: [
23.           Text("Kode Produk : " + widget.kodeProduk),
24.           Text("Nama Produk : ${widget.namaProduk}"), // jika didalam String
25.           Text("Harga : ${widget.harga.toString()}"), // jika didalam String
26.         ],
27.       ),
28.     );
29.   }
30. }
```

4. 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:flutter/material.dart';
2.
3. class ProdukForm extends StatefulWidget {
4.   @override
5.   _ProdukFormState createState() => _ProdukFormState();
6. }
7.
8. class _ProdukFormState extends State<ProdukForm> {
9.
10.  final _kodeProdukTextboxController = TextEditingController();
11.  final _namaProdukTextboxController = TextEditingController();
12.  final _hargaProdukTextboxController = TextEditingController();
13.
14.  @override
```

```

15.     Widget build(BuildContext context) {
16.       return Scaffold(
17.         appBar: AppBar(
18.           title: Text('Form Produk'),
19.         ),
20.         body: SingleChildScrollView(
21.           child: Column(
22.             children: [
23.               _textboxKodeProduk(),
24.               _textboxNamaProduk(),
25.               _textboxHargaProduk(),
26.               _tombolSimpan()
27.             ],
28.           ),
29.         );
30.     }
31.   }
32.
33.   _textboxKodeProduk() {
34.     return TextField(
35.       decoration: InputDecoration(labelText: "Kode Produk"),
36.     );
37.   }
38.
39.   _textboxNamaProduk() {
40.     return TextField(
41.       decoration: InputDecoration(labelText: "Nama Produk"),
42.     );
43.   }
44.
45.   _textboxHargaProduk() {
46.     return TextField(
47.       decoration: InputDecoration(labelText: "Harga"),
48.     );
49.   }
50.
51.   _tombolSimpan() {
52.     return RaisedButton(
53.       child: Text('Simpan'),
54.       onPressed: () {},
55.     );
56.   }
57. }

```

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: InputDecoration(labelText: "Kode Produk"),
    controller: _kodeProdukTextboxController,
  );
}

```

Pada fungsi `_textboxNamaProduk()` menjadi

```

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

```

```
        );
}
```

Pada fungsi `_textboxhargaProduk()` menjadi

```
_textboxHargaProduk() {
    return TextField(
        decoration: 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 RaisedButton(
        child: Text('Simpan'),
        onPressed: () {
            String kode_produk = _kodeProdukTextboxController.text;
            String nama_produk = _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(new MaterialPageRoute(builder: (context)=>ProdukDetail(kodeProduk: kode_produk, namaProduk: nama_produk, harga: harga)));
        },
    );
}
```

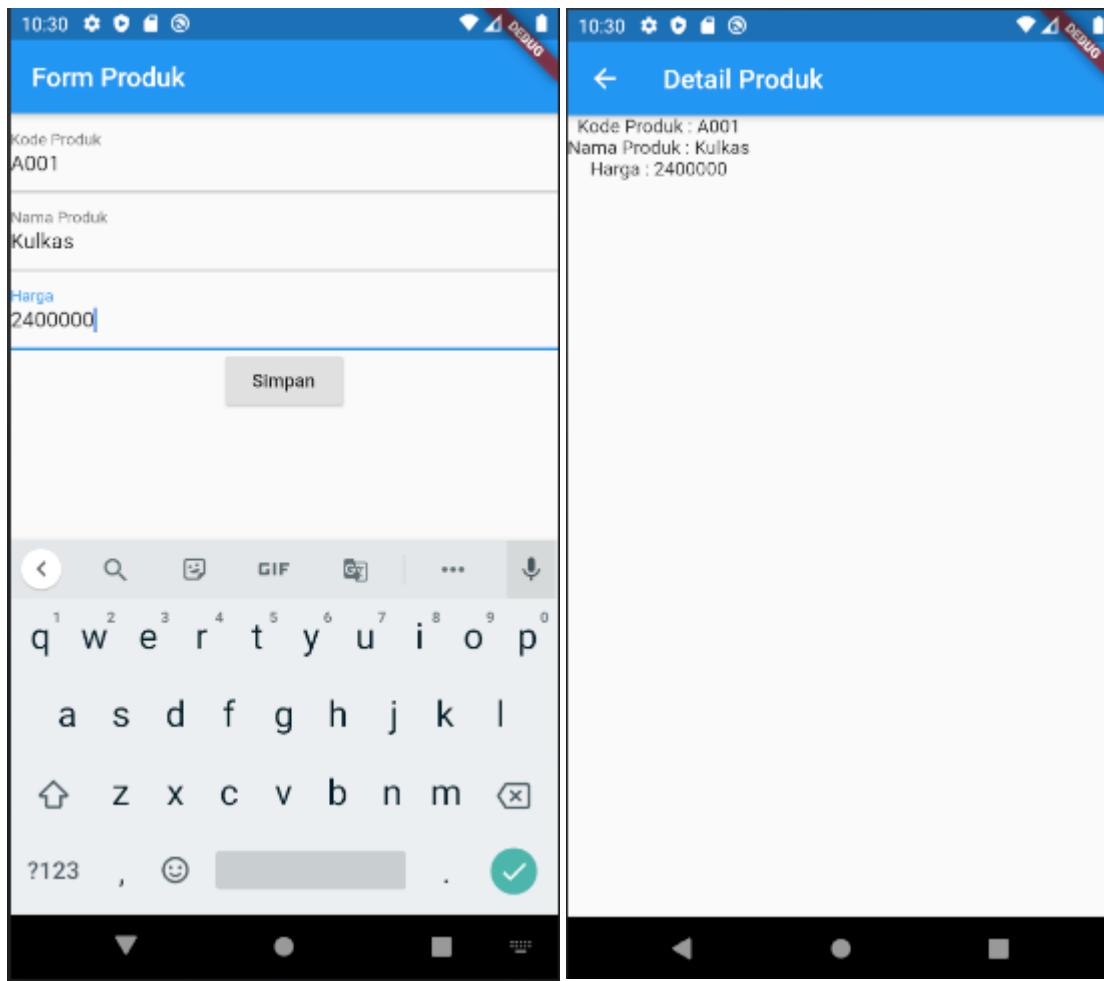
Sehingga keseluruhan kode menjadi:

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

```

27.         ],
28.     ),
29.   ),
30. );
31. }
32.
33. _textboxKodeProduk() {
34.   return TextField(
35.     decoration: InputDecoration(labelText: "Kode Produk"),
36.     controller: _kodeProdukTextboxController,
37.   );
38. }
39.
40. _textboxNamaProduk() {
41.   return TextField(
42.     decoration: InputDecoration(labelText: "Nama Produk"),
43.     controller: _namaProdukTextboxController,
44.   );
45. }
46.
47. _textboxHargaProduk() {
48.   return TextField(
49.     decoration: InputDecoration(labelText: "Harga"),
50.     controller: _hargaProdukTextboxController,
51.   );
52. }
53.
54. _tombolSimpan() {
55.   return RaisedButton(
56.     child: Text('Simpan'),
57.     onPressed: () {
58.       String kode_produk = _kodeProdukTextboxController.text;
59.       String nama_produk = _namaProdukTextboxController.text;
60.       int harga = int.parse(_hargaProdukTextboxController.text); //parsing dari String ke int
61.       // pindah ke halaman Produk Detail dan mengirim data
62.       Navigator.of(context).push(new MaterialPageRoute(builder: (context)=>ProdukDetail(kodeProduk: kode_produk, namaProduk: nama_produk, harga: harga,)));
63.     },
64.   );
65. }
66. }

```



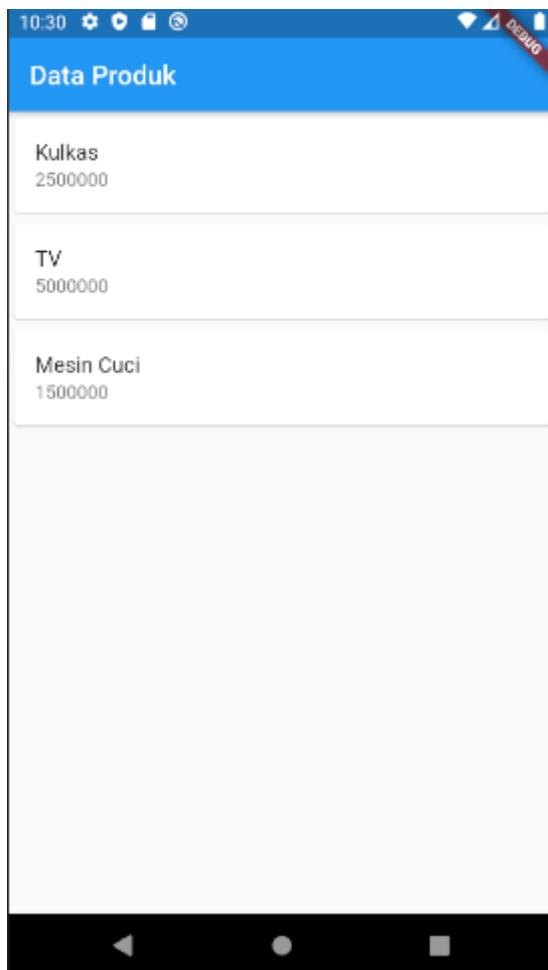
5. Membuat ListView Produk

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.   @override
5.   _ProdukPageState createState() => _ProdukPageState();
6. }
7.
8. class _ProdukPageState extends State<ProdukPage> {
9.   @override
10.  Widget build(BuildContext context) {
11.    return Scaffold(
12.      appBar: AppBar(
13.        title: Text("Data Produk"),
14.      ),
15.      body: ListView(
16.        children: [
17.          //list 1
18.          Card(
19.            child: ListTile(
20.              title: Text("Kulkas"),
21.              subtitle: Text("2500000"),
22.            ),
23.          ),
24.          //list 2
```

```
25.          Card(
26.            child: ListTile(
27.              title: Text("TV"),
28.              subtitle: Text("5000000"),
29.            ),
30.          ),
31.          //list 3
32.          Card(
33.            child: ListTile(
34.              title: Text("Mesin Cuci"),
35.              subtitle: Text("1500000"),
36.            ),
37.          ),
38.        ],
39.      ),
40.    );
41.  }
42. }
```

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



6. Tugas Pertemuan 3

PERTEMUAN 4

1. Membuat Route (Pindah Halaman)

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

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

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

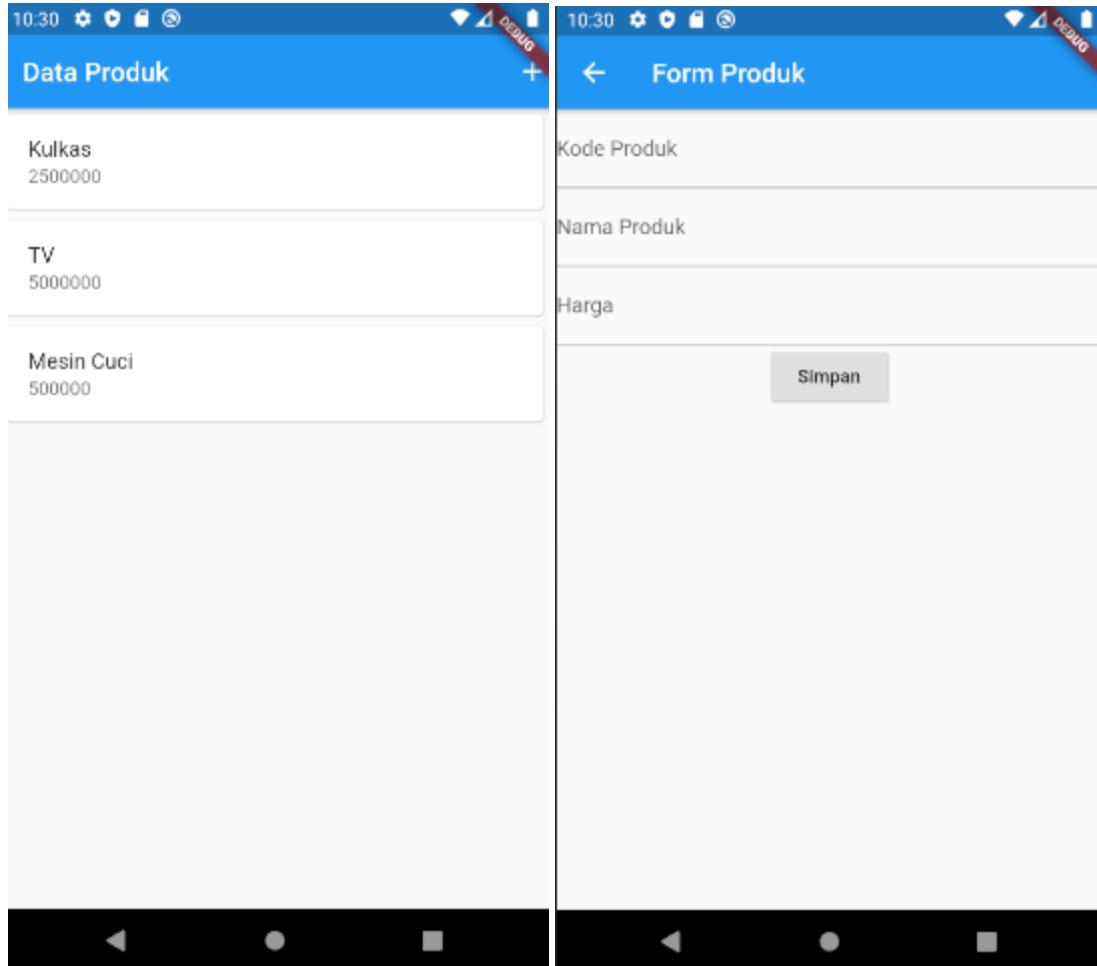
Secara Keseluruhan kode tersebut akan menjadi

```
1. import 'package:aplikasi_flutter_pertamaku/ui/produk_form.dart';  
2. import 'package:flutter/material.dart';  
3.  
4. class ProdukPage extends StatefulWidget {  
5.     @override  
6.     _ProdukPageState createState() => _ProdukPageState();  
7. }  
8.  
9. class _ProdukPageState extends State<ProdukPage> {  
10.    @override  
11.    Widget build(BuildContext context) {  
12.        return Scaffold(  
13.            appBar: AppBar(  
14.                title: Text("Data Produk"),  
15.                actions: [  
16.                    GestureDetector(  
17.                        // menampilkan icon +  
18.                        child: Icon(Icons.add),  
19.                        //pada saat icon + di tap  
20.                        onTap: () async {  
21.                            //berpindah ke halaman ProdukForm  
22.                            Navigator.push(context,  
23.                                new MaterialPageRoute(builder: (context) => ProdukForm()));  
24.                            },  
25.                        ),  
26.                    ],  
27.                ),  
28.                body: ListView(  
29.                    children: [  
30.                        //list 1  
31.                        Card(  
32.                            child: ListTile(  
33.                                title: Text("Kulkas"),  
34.                                subtitle: Text("2500000"),  
35.                            ),  
36.                        ),  
37.                    ],  
38.                ),  
39.            ),  
40.        );  
41.    }  
42. }
```

```

36.        ),
37.        //list 2
38.        Card(
39.            child: ListTile(
40.                title: Text("TV"),
41.                subtitle: Text("5000000"),
42.            ),
43.        ),
44.        //list 3
45.        Card(
46.            child: ListTile(
47.                title: Text("Mesin Cuci"),
48.                subtitle: Text("500000"),
49.            ),
50.        ),
51.    ],
52. ),
53. );
54. }
55. 
```

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



2. 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;
    //membuat constructor
    ItemProduk({this.kodeProduk, this.namaProduk, this.harga});
    @override
    Widget build(BuildContext context) {
        return Card(
            child: ListTile(
                title: Text(namaProduk),
                subtitle: Text(harga.toString()), //parsing dari int ke string
            ),
        );
    }
}
```

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.     @override
6.     _ProdukPageState createState() => _ProdukPageState();
7. }
8.
9. class _ProdukPageState extends State<ProdukPage> {
10.    @override
11.    Widget build(BuildContext context) {
12.        return Scaffold(
13.            appBar: AppBar(
14.                title: Text("Data Produk"),
15.                actions: [
16.                    GestureDetector(
17.                        // menampilkan icon +
18.                        child: Icon(Icons.add),
19.                        //pada saat icon + di tap
20.                        onTap: () async {
21.                            //berpindah ke halaman ProdukForm
22.                            Navigator.push(context,
23.                                new MaterialPageRoute(builder: (context) => ProdukForm()));
24.                        },
25.                    ),
26.                ],
27.            ),
28.            body: ListView(
29.                children: [
30.                    //list 1
31.                    ItemProduk(
32.                        kodeProduk: "A001",
33.                        namaProduk: "Kulkas",
34.                        harga: 2500000,
35.                    ),
36.                    //list 2
37.                    ItemProduk(
38.                        kodeProduk: "A002",
39.                        namaProduk: "TV",
40.                        harga: 5000000,
```

```

41.        ),
42.        //list 3
43.        ItemProduk(
44.            kodeProduk: "A003",
45.            namaProduk: "Mesin Cuci",
46.            harga: 1500000,
47.        ),
48.    ],
49.),
50.);
51.}
52.}
53.
54. class ItemProduk extends StatelessWidget {
55.     final String kodeProduk;
56.     final String namaProduk;
57.     final int harga;
58.
59.     //membuat constructor
60.     ItemProduk({this.kodeProduk, this.namaProduk, this.harga});
61.
62.     @override
63.     Widget build(BuildContext context) {
64.         return Card(
65.             child: ListTile(
66.                 title: Text(namaProduk),
67.                 subtitle: Text(harga.toString()), //parsing dari int ke string
68.             ),
69.         );
70.     }
71. }

```

3. 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

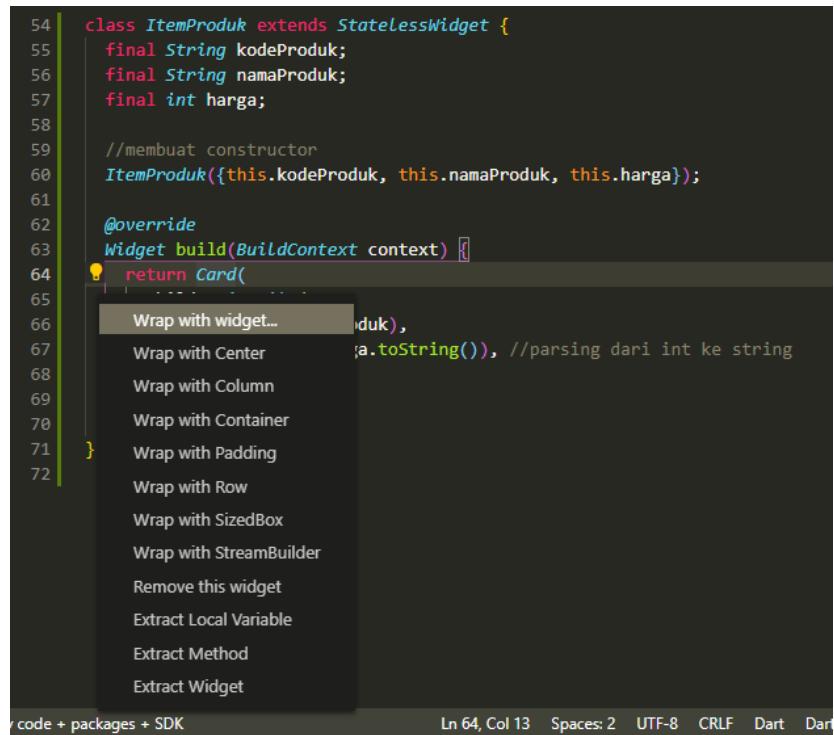
Kita akan memodifikasi Class ItemProduk pada file **produk_page.dart**. Untuk menambahkan widget diatas widget yang telah dibuat dapat dilakukan dengan cara, arahkan kursor pada widget, misalnya dalam hal ini adalah widget **Card**

```

54. class ItemProduk extends StatelessWidget {
55.     final String kodeProduk;
56.     final String namaProduk;
57.     final int harga;
58.
59.     //membuat constructor
60.     ItemProduk({this.kodeProduk, this.namaProduk, this.harga});
61.
62.     @override
63.     Widget build(BuildContext context) {
64.         return Card(
65.             child: ListTile(
66.                 title: Text(namaProduk),
67.                 subtitle: Text(harga.toString()), //parsing dari int ke string
68.             ), // ListTile
69.         ); // Card
70.     }
71. }

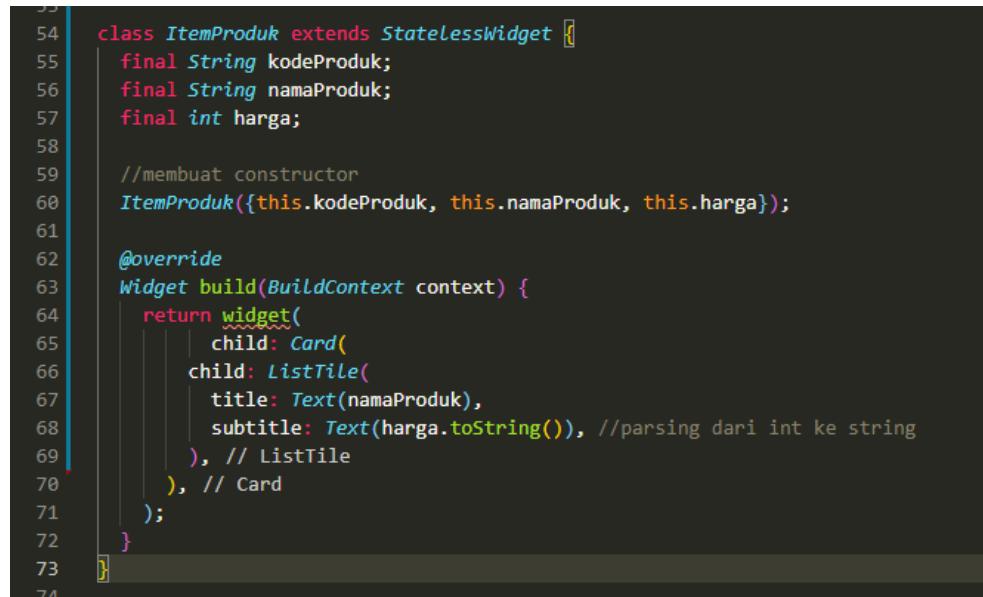
```

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



```
54 class ItemProduk extends StatelessWidget {  
55     final String kodeProduk;  
56     final String namaProduk;  
57     final int harga;  
58  
59     //membuat constructor  
60     ItemProduk({this.kodeProduk, this.namaProduk, this.harga});  
61  
62     @override  
63     Widget build(BuildContext context) {  
64         return Card(  
65             child:  
66                 Wrap with widget...  
67                 Wrap with Center  
68                 Wrap with Column  
69                 Wrap with Container  
70                 Wrap with Padding  
71                 Wrap with Row  
72                 Wrap with SizedBox  
73                 Wrap with StreamBuilder  
74                 Remove this widget  
75                 Extract Local Variable  
76                 Extract Method  
77                 Extract Widget  
78         );  
79     }  
80 }
```

Kemudian akan menjadi



```
54 class ItemProduk extends StatelessWidget {  
55     final String kodeProduk;  
56     final String namaProduk;  
57     final int harga;  
58  
59     //membuat constructor  
60     ItemProduk({this.kodeProduk, this.namaProduk, this.harga});  
61  
62     @override  
63     Widget build(BuildContext context) {  
64         return widget(  
65             child: Card(  
66                 child: ListTile(  
67                     title: Text(namaProduk),  
68                     subtitle: Text(harga.toString()), //parsing dari int ke string  
69                 ), // ListTile  
70             ), // Card  
71         );  
72     }  
73 }  
74 }
```

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;
```

```

//membuat constructor
ItemProduk({this.kodeProduk, this.namaProduk, this.harga});

@Override
Widget build(BuildContext context) {
    return GestureDetector(
        child: Card(
            child: ListTile(
                title: Text(nombre),
                subtitle: Text(harga.toString()), //parsing dari int ke string
            ),
        ),
        onTap: () {
            // pindah ke halaman Produk Detail dan mengirim data
            Navigator.of(context).push(new MaterialPageRoute(
                builder: (context) => ProdukDetail(
                    nombre: nombre,
                    precio: precio,
                    cantidad: cantidad,
                )));
        },
    );
}
}

```

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

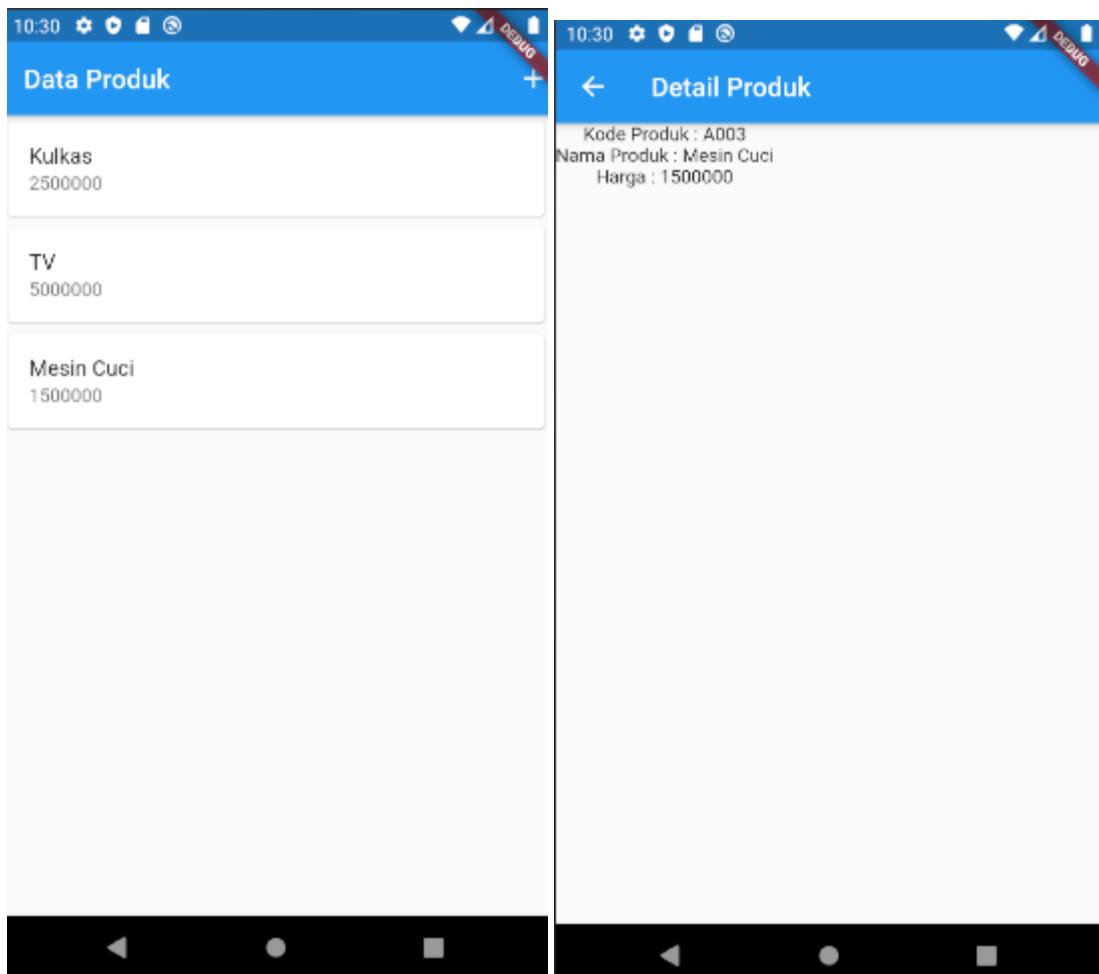
```

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.   @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: Text("Data Produk"),
16.         actions: [
17.           GestureDetector(
18.             // menampilkan icon +
19.             child: Icon(Icons.add),
20.             //pada saat icon + di tap
21.             onTap: () async {
22.               //berpindah ke halaman ProdukForm
23.               Navigator.push(context,
24.                 new MaterialPageRoute(builder: (context) => ProdukForm()));
25.             },
26.           ),
27.         ],
28.       ),
29.       body: ListView(
30.         children: [
31.           //list 1
32.           ItemProduk(
33.             nombre: "A001",
34.             precio: "Kulkas",

```

```

35.             harga: 2500000,
36.         ),
37.         //list 2
38.         ItemProduk(
39.             kodeProduk: "A002",
40.             namaProduk: "TV",
41.             harga: 5000000,
42.         ),
43.         //list 3
44.         ItemProduk(
45.             kodeProduk: "A003",
46.             namaProduk: "Mesin Cuci",
47.             harga: 1500000,
48.         ),
49.     ],
50. ),
51. );
52. }
53. }
54.
55. class ItemProduk extends StatelessWidget {
56.   final String kodeProduk;
57.   final String namaProduk;
58.   final int harga;
59.
60.   //membuat constructor
61.   ItemProduk({this.kodeProduk, this.namaProduk, this.harga});
62.
63.   @override
64.   Widget build(BuildContext context) {
65.     return GestureDetector(
66.       child: Card(
67.         child: ListTile(
68.           title: Text(namaProduk),
69.           subtitle: Text(harga.toString()), //parsing dari int ke string
70.         ),
71.       ),
72.       onTap: () {
73.         // pindah ke halaman Produk Detail dan mengirim data
74.         Navigator.of(context).push(new MaterialPageRoute(
75.             builder: (context) => ProdukDetail(
76.                 kodeProduk: kodeProduk,
77.                 namaProduk: namaProduk,
78.                 harga: harga,
79.             )));
80.       },
81.     );
82.   }
83. }
```



4. Tugas Pertemuan 4

- a. Mendownload XAMPP, Composer dan Postman untuk kebutuhan pertemuan 5! (Link download bisa diakses pada modul pertemuan 5)
- b. Melakukan instalasi XAMPP, Composer dan Postman sesuai dengan Langkah-langkah yang ada pada pertemuan 5!
- c. Memastikan keberhasilan tahapan poin a dan b sebelum memasuki pertemuan 5!

PERTEMUAN 5

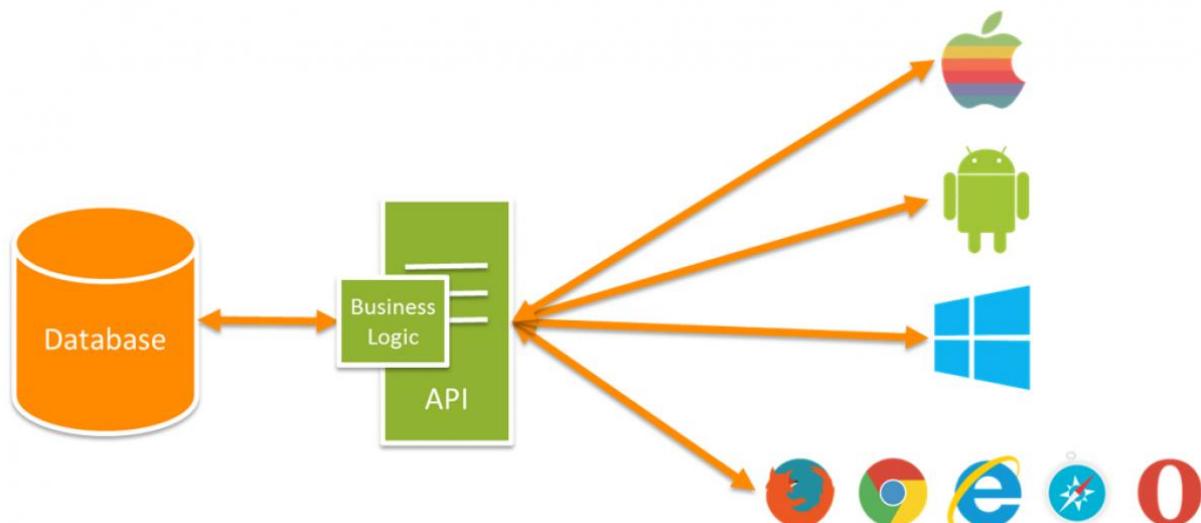
1. Membuat projek flutter yang terhubung dengan API

a. 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

b. 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.

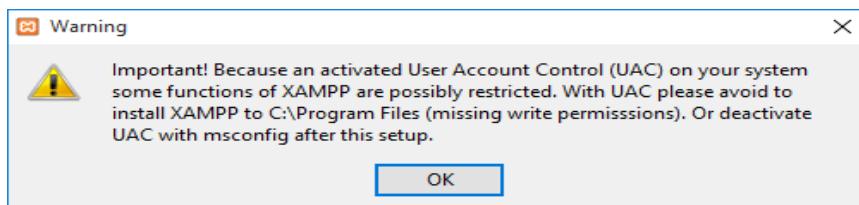
2. Membuat projek Toko API (Restful API)

a. Installasi Apache, MySql dan PHP (XAMPP)

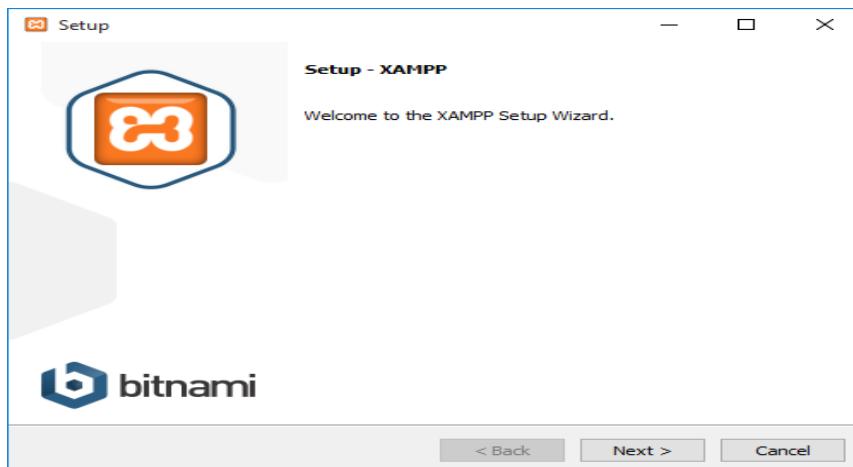
XAMPP adalah perangkat lunak untuk membuat komputer menjadi web server. XAMPP dapat di download melalui link url:

<https://www.apachefriends.org/download.html>.

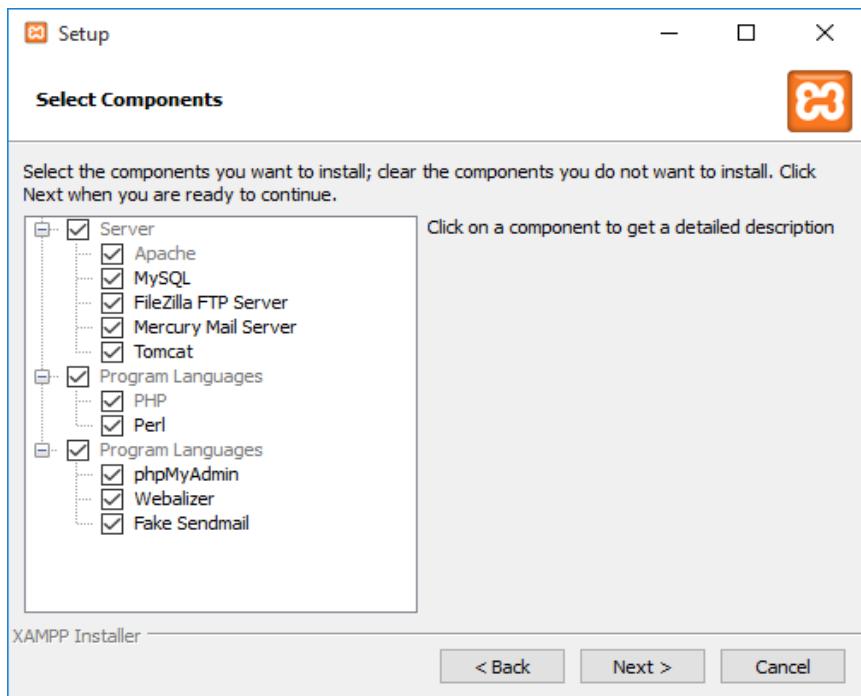
Setelah itu, double klik file xampp yang baru di download. Jika muncul pesan error seperti gambar dibawah, abaikan saja dan lanjutkan klik tombol OK.



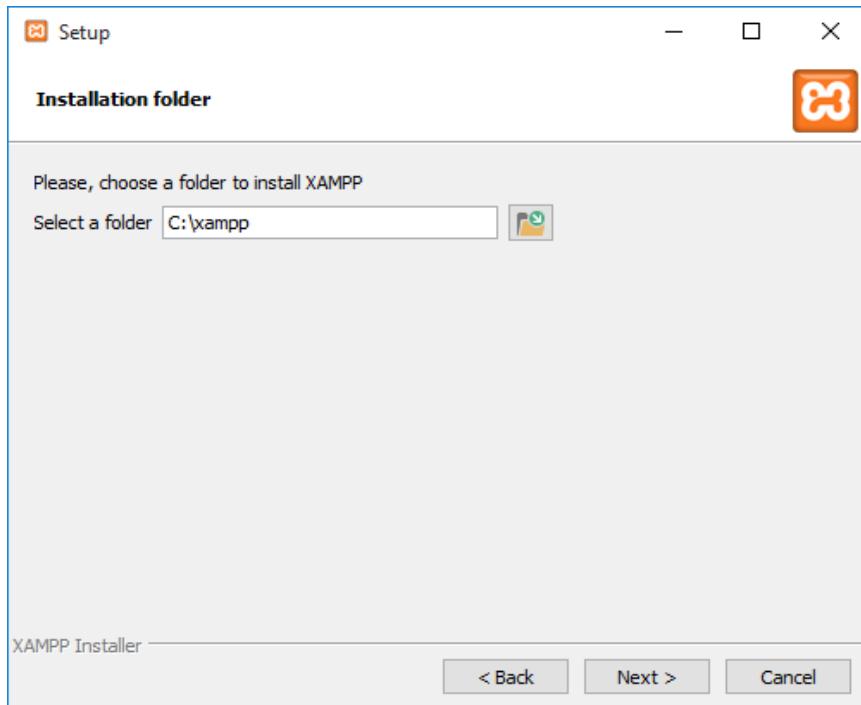
Berikutnya akan muncul jendela **Setup-XAMPP**. Klik tombol **Next** untuk melanjutkan proses berikutnya.



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**.



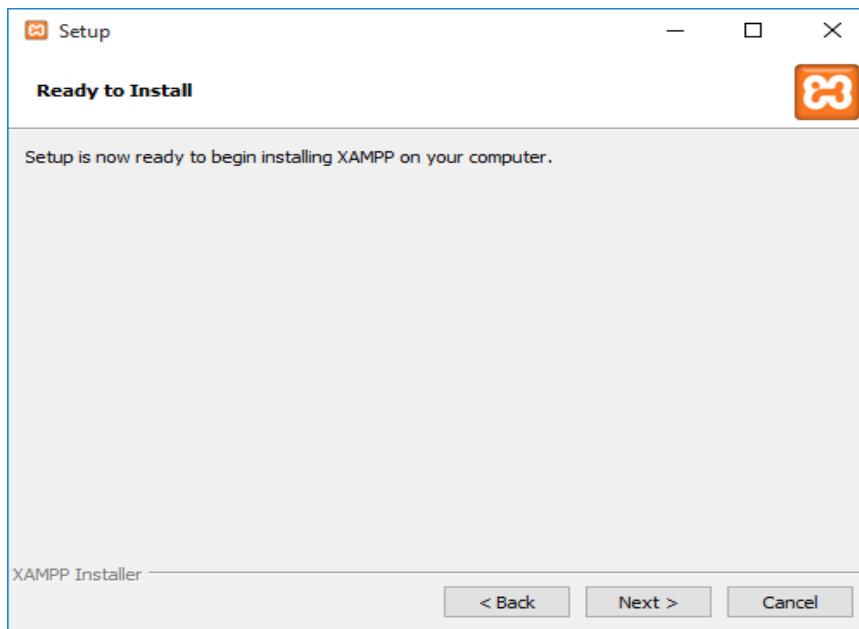
Kita akan menjumpai jendela tawaran untuk mempelajari lebih lanjut tentang Bitnami. Silahkan checklist jika ingin mempelajari lebih lanjut. Bitnami adalah pustaka dari aplikasi

client server yang populer seperti misalnya CMS WordPress atau Drupal, dengan penawaran kemudahan dalam instalasi hanya dengan satu klik. Kemudian klik tombol **Next**.



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

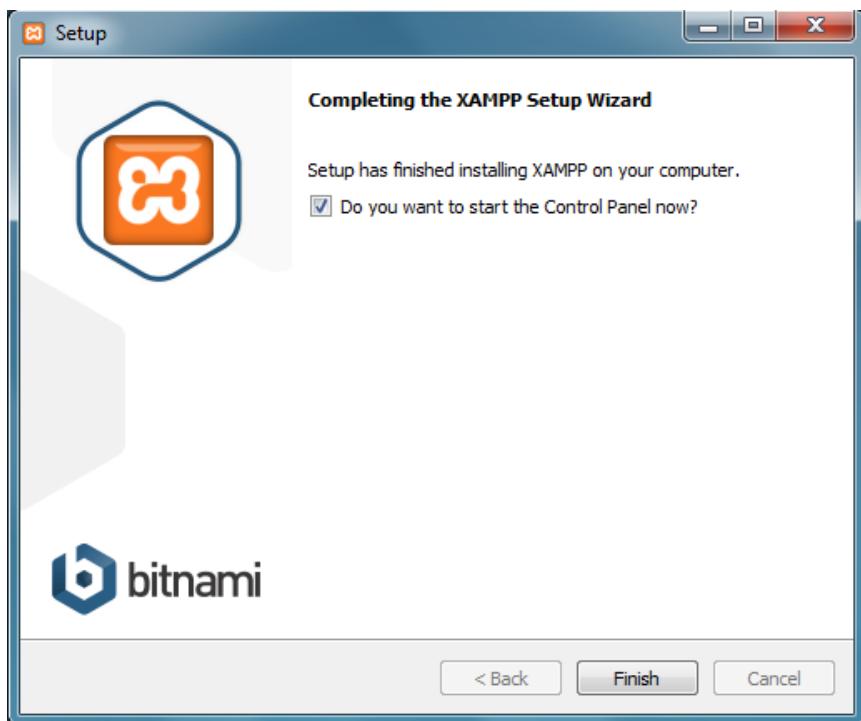
Kemudian klik tombol **Next**.



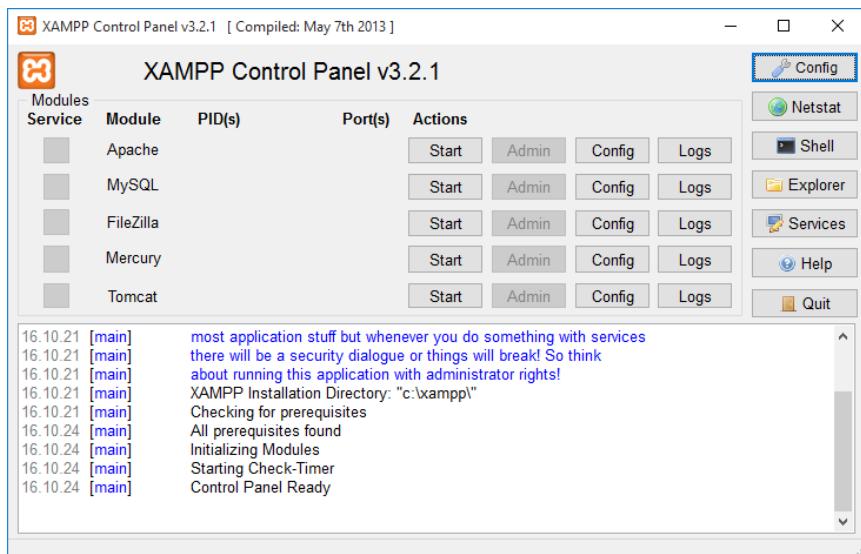
Dan proses instalasi pun berjalan.



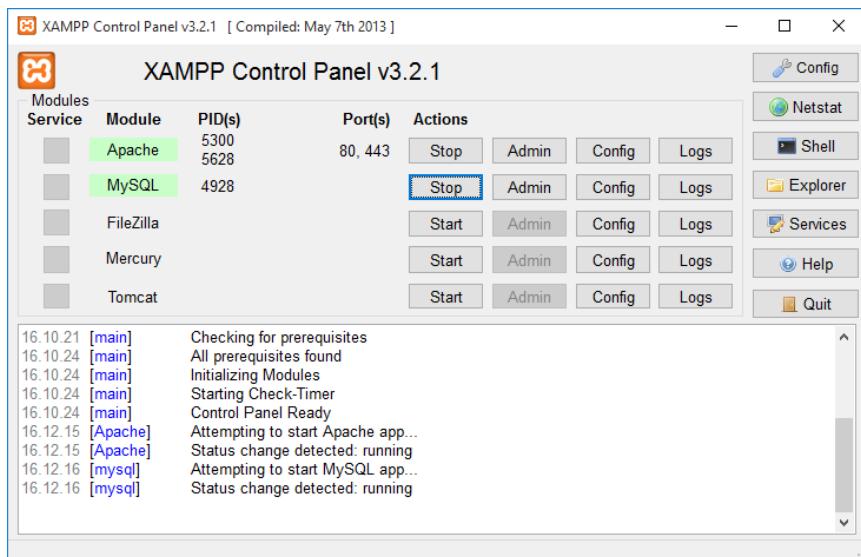
Tunggu hingga proses install selesai dan muncul jendela sebagai berikut.



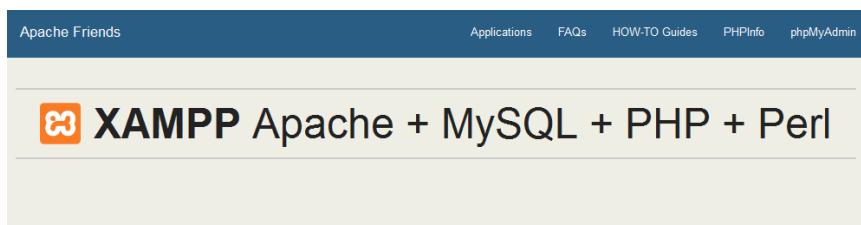
Klik tombol **Finish** setelah itu akan muncul jendela **Xampp Control Panel** yang berguna untuk menjalankan server.



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.

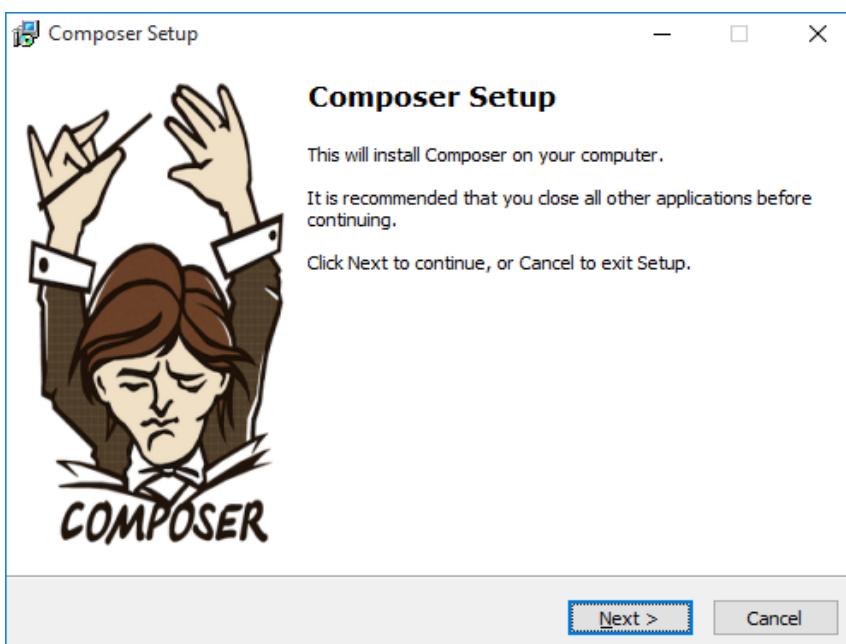
Start the XAMPP Control Panel to check the server status.

b. Install Composer

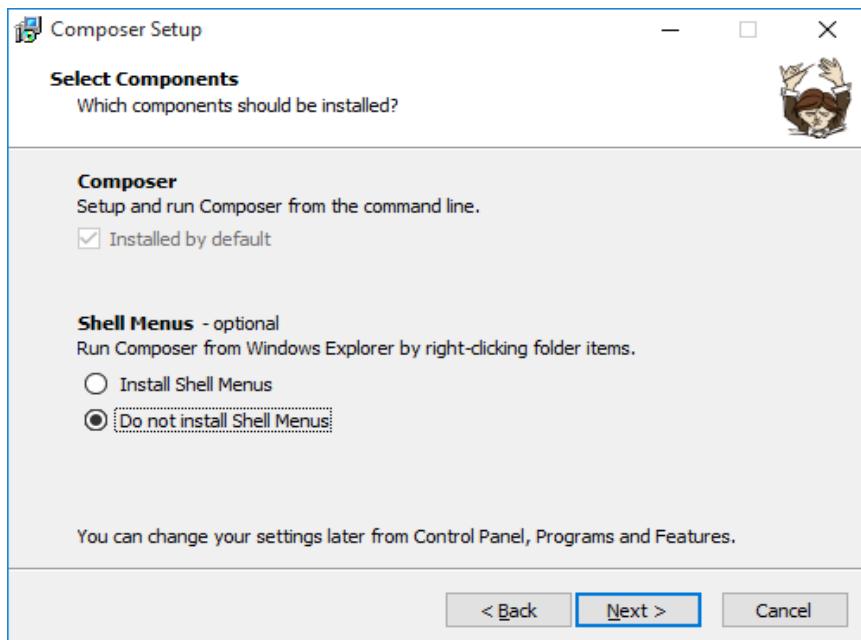
Composer adalah Dependency Management Tools untuk PHP. Sederhananya Composer akan membantu kita mencari dan mendownload file-file yang diperlukan oleh sebuah aplikasi yang akan install yang akan tersimpan pada folder **vendor**. Composer untuk windows dapat didownload di <https://getcomposer.org/Composer-Setup.exe>. Install composer sangat mudah, cukup double klik file composer yang telah di download.



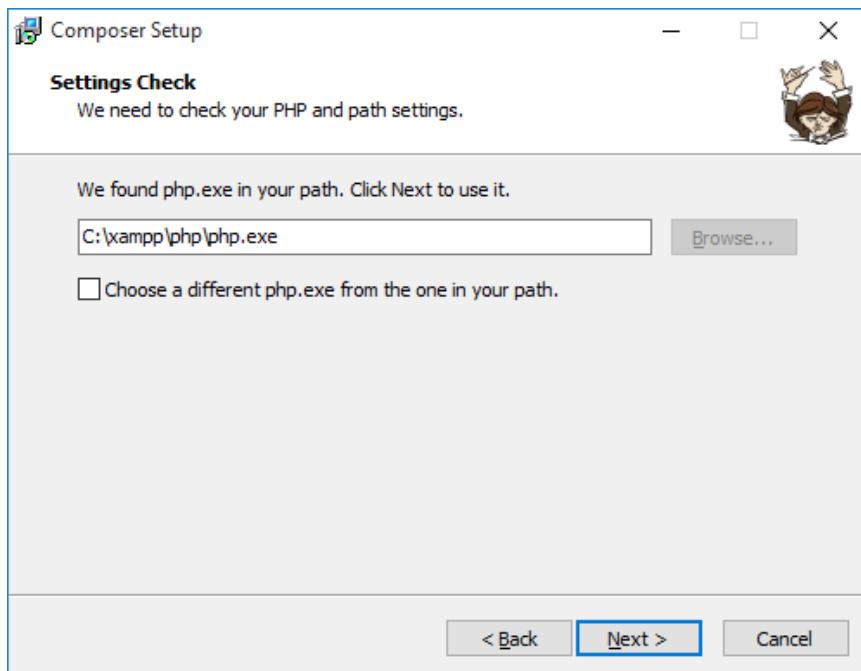
Kemudian klik tombol **Next**.

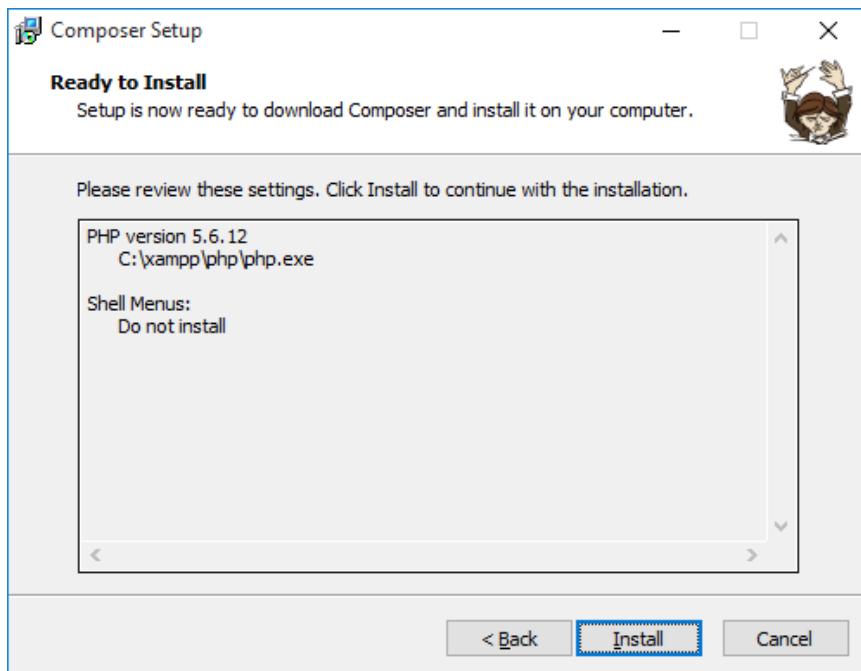


Pada jendela **Selects Components** kita dapat lanjutkan dengan menekan tombol **Next**.



Pada umumnya secara otomatis composer akan mendapatkan file php.exe yang telah kita install sebelumnya (Installasi XAMPP). Lanjutkan dengan menekan tombol **Next** dan **Install**.



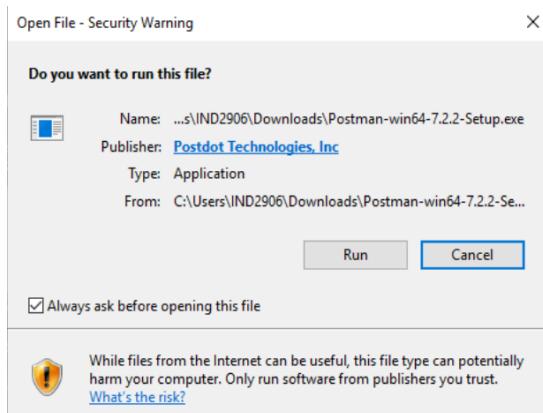


c. Install Postman

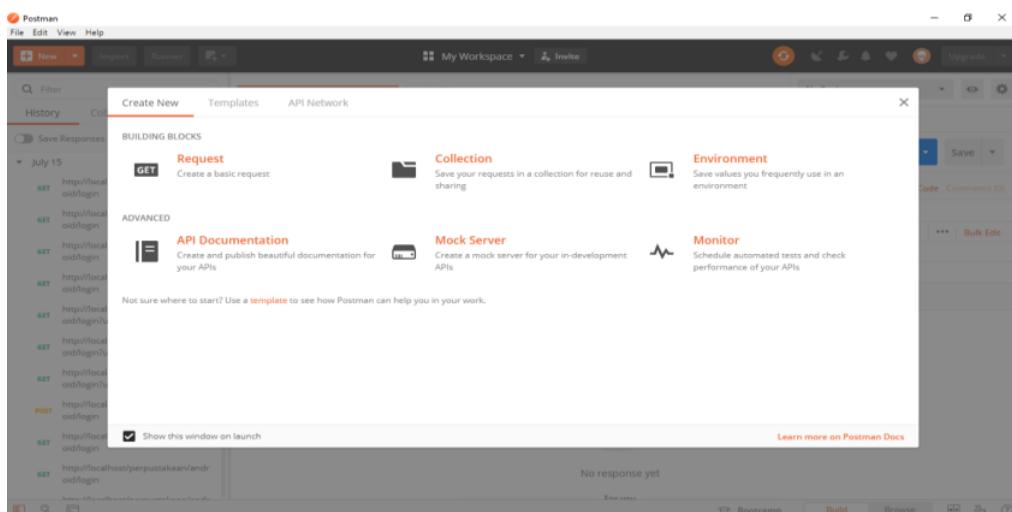
Postman adalah sebuah aplikasi fungsinya adalah sebagai REST Client atau istilahnya adalah aplikasi yang digunakan untuk melakukan uji coba REST API yang telah kita buat. Postman ini merupakan tools wajib bagi para developer yang bergerak pada pembuatan API, fungsi utama postman ini adalah sebagai GUI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

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

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



Kemudian tunggu hingga proses instalasi selesai dan muncul seperti gambar berikut



3. 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	<pre>{ "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", }] }</pre>

2) Create Produk

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

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	{ "code" : "integer", "status" : "boolean", "data" : "boolean" }

4) Show Produk

EndPoint	/produk/{id}
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", } }

5) Delete Produk

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

4. Tugas Pertemuan 5

- d. Download Lumen untuk persiapan pertemuan 6 dan lakukan instalasi Lumen sesuai petunjuk yang ada pada pertemuan 6
- e. Memastikan Lumen sudah terinstalasi sebelum pertemuan 6 dimulai.

PERTEMUAN 6

1. Pembuatan Database

Buat database dengan nama : **toko_api**

Kemudian buat tabel-tabel dengan perintah sebagai berikut :

a. 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	None			AUTO_INCREMENT
2	nama	varchar(255)	utf8mb4_general_ci	No	None			
3	email	varchar(255)	utf8mb4_general_ci	No	None			
4	password	varchar(255)	utf8mb4_general_ci	No	None			

b. 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	None			AUTO_INCREMENT
2	member_id 	int(11)		No	None			
3	auth_key	varchar(255)	utf8mb4_general_ci	No	None			

c. 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		

2. Installasi Lumen sebagai Restful API

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

Untuk membuat projek lumen dapat menggunakan composer pada command prompt dengan menjalankan perintah

```
composer create-project --prefer-dist laravel/lumen <nama_projek>
```

Sekarang kita akan membuat projek Restful API dengan nama **toko-api**. Silahkan buka Command Prompt kemudian ketikkan perintah

```
C:\Users\BSI> cd c:\xampp\htdocs
C:\xampp\htdocs> composer create-project --prefer-dist laravel/lumen toko-api
```

3. Konfigurasi Projek

a. Koneksi ke database

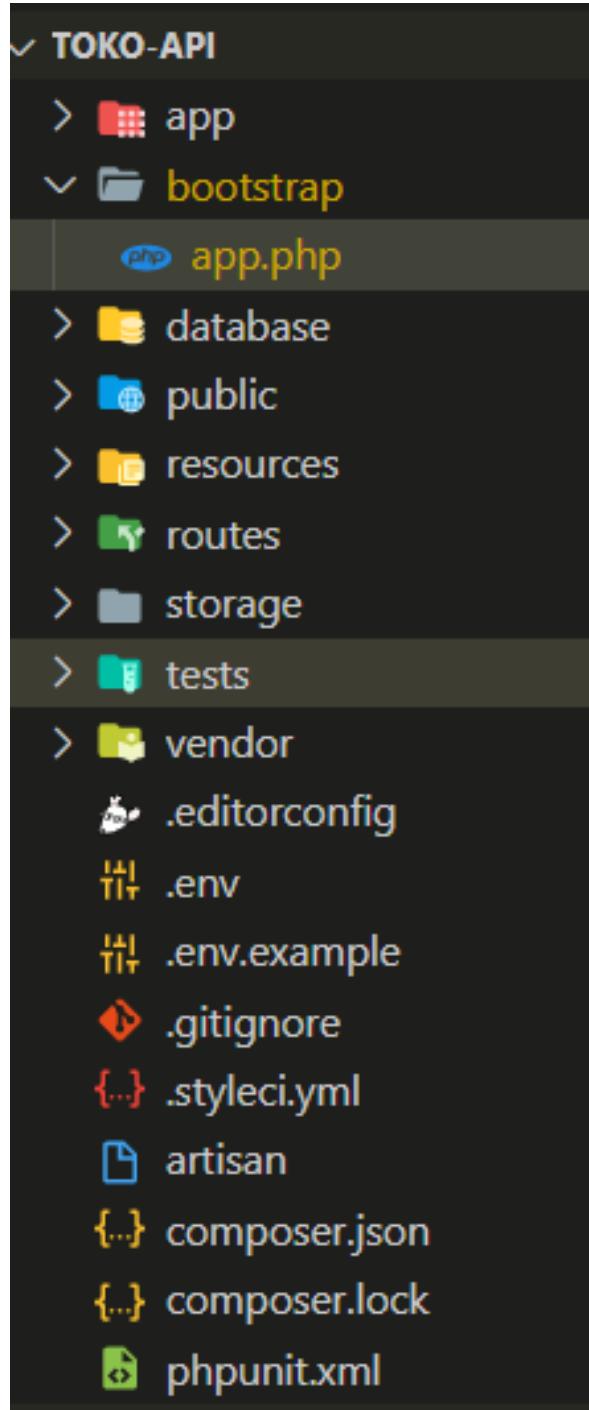
Buka file **.env** kemudian ubah kode berikut

```
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Menjadi sebagai berikut

```
DB_DATABASE=toko_api  
DB_USERNAME=root  
DB_PASSWORD=
```

Kemudian buka file `app.php` pada folder “bootstrap”



Kemudian hilangkan komentar pada kode berikut (pada baris 26 dan 28)

```
// $app->withFacades();
```

```
// $app->withEloquent();
```

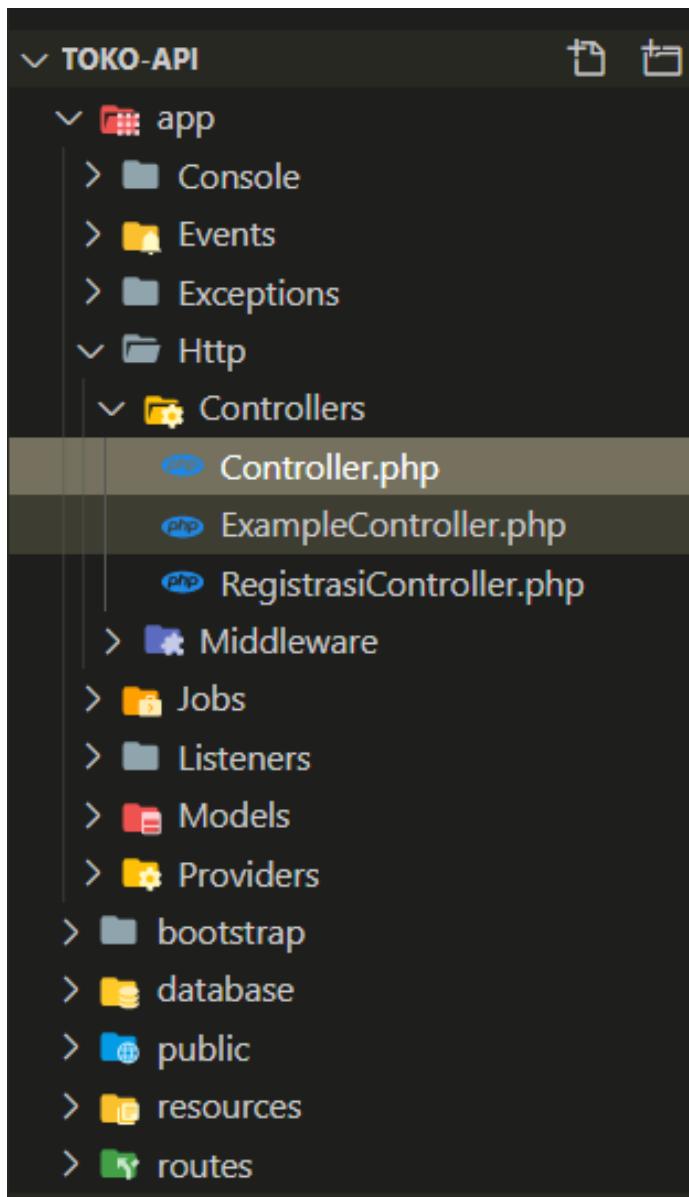
Menjadi

```
$app->withFacades();
```

```
$app->withEloquent();
```

b. Membuat hasil response

Buka file **Controller.php** pada folder “app\Http\Controllers”



Kemudian tambahkan kode pada file tersebut sehingga menjadi

```

<?php

namespace App\Http\Controllers;

use Laravel\Lumen\Routing\Controller as BaseController;

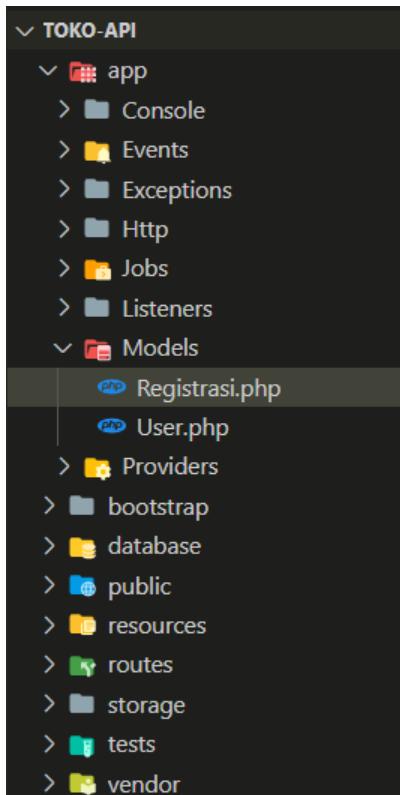
class Controller extends BaseController
{
    protected function responseHasil($code, $status, $data)
    {
        return \response()->json([
            'code' => $code,
            'status' => $status,
            'data' => $data
        ]);
    }
}

```

4. Registrasi

a. Membuat model Registrasi

Buat sebuah file dengan nama **Registrasi.php** pada folder “app/Models”



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

```

<?php

namespace App\Models;

```

```
use Illuminate\Database\Eloquent\Model;

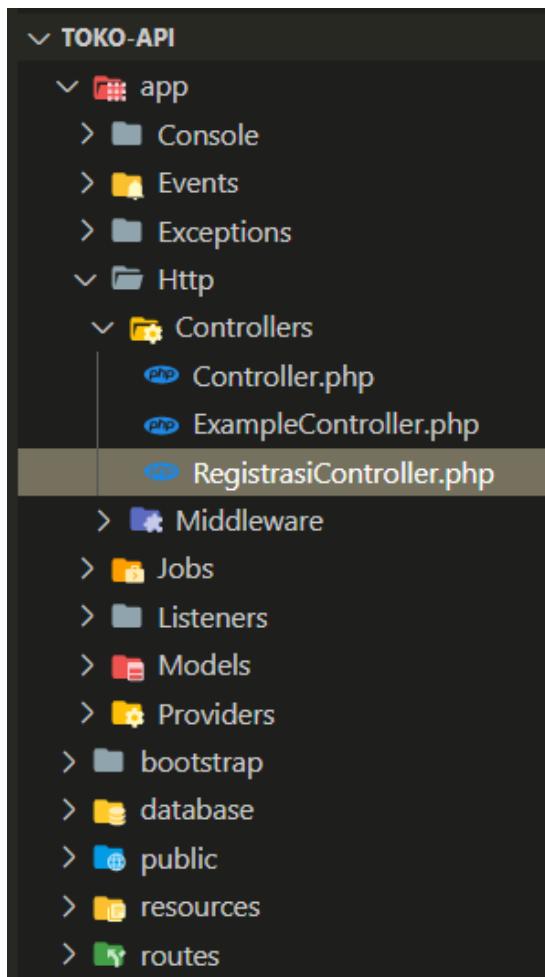
class Registrasi extends Model
{
    protected $table = 'member';

    protected $fillable = ['nama', 'email', 'password'];

    public $timestamps = false;
}
```

b. Membuat controller Registrasi

Buat sebuah file dengan nama **RegistrasiController.php** pada folder “app\Http\Controller”



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

```
<?php

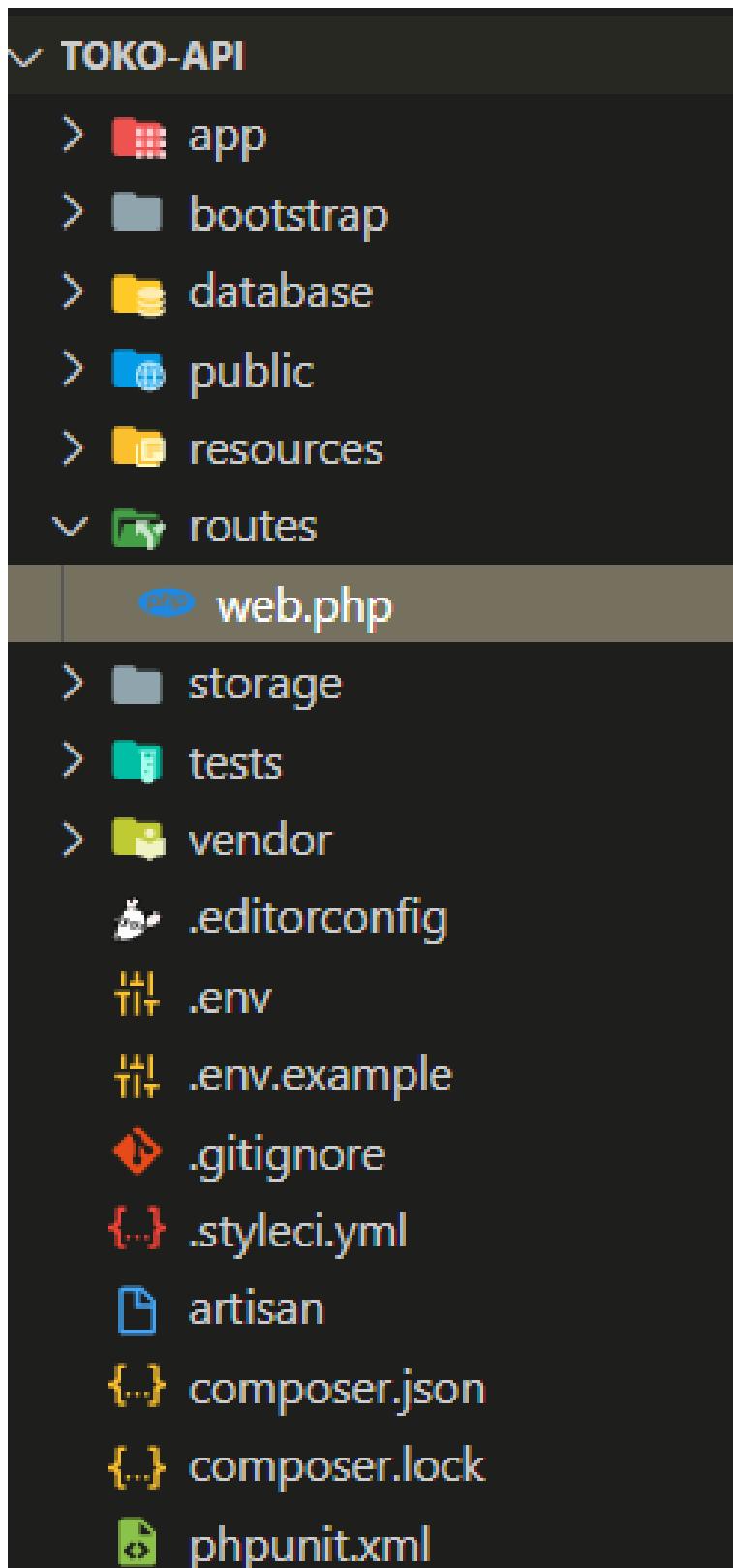
namespace App\Http\Controllers;

use App\Models\Registrasi;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class RegistrasiController extends Controller
{
    public function registrasi(Request $request)
    {
        $nama = $request->input('nama');
        $email = $request->input('email');
        $password = Hash::make($request->input('password'));

        Registrasi::create([
            'nama' => $nama,
            'email' => $email,
            'password' => $password,
        ]);
        return $this->responseHasil(200, true, "Registrasi Berhasil");
    }
}
```

c. Menambah route Registrasi
Buka file **web.php** pada folder “routes”



Kemudian tambahkan kode berikut:

```

<?php

/** @var \Laravel\Lumen\Routing\Router $router */

/*
-----
/ Application Routes
-----
/
/ Here is where you can register all of the routes for an application.
/ It is a breeze. Simply tell Lumen the URIs it should respond to
/ and give it the Closure to call when that URI is requested.
/
*/



$router->get('/', function () use ($router) {
    return $router->app->version();
});

$router->post('/registrasi', ['uses' => 'RegistrasiController@registerasi']);

```

Pada baris terakhir kita menambahkan routing registrasi agar dapat diakses. 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 **x-www-form-urlencoded**
5. Kemudian isi key sesuai dengan field request pada **RegistrasiController** pada fungsi registrasi yaitu *nama, email, password* kemudian klik tombol **Send**

```

$nama = $request->input('nama');
$email = $request->input('email');
$password = Hash::make($request->input('password'));

```

The screenshot shows a Postman interface with a POST request to `localhost/toko-api/public/registrasi`. The request body is in form-data format, containing four fields: `nama` (Administrator), `email` (`admin@admin.com`), `password` (admin), and a key-value pair of `Key` and `Value`. The response body is displayed in JSON format, showing a success message.

Body

form-data

	KEY	VALUE	...	Bulk Edit
<input checked="" type="checkbox"/>	nama	Administrator		
<input checked="" type="checkbox"/>	email	admin@admin.com		
<input checked="" type="checkbox"/>	password	admin		
	Key	Value		

Body

Pretty Raw Preview Visualize JSON

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

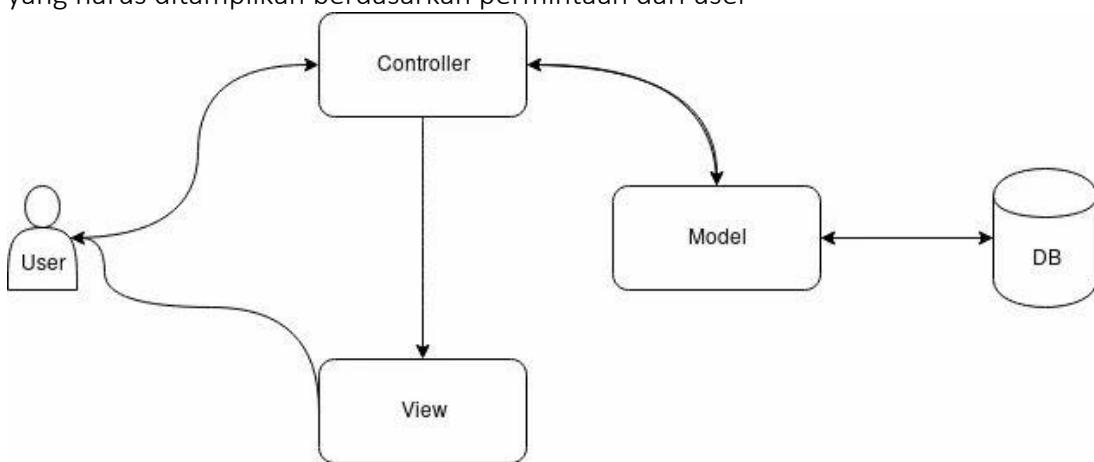
PERTEMUAN 7

1. Konsep MVC

Sama seperti Framework PHP pada umumnya, Lumen juga mengusung konsep MVC dalam Design Pattern.

MVC adalah konsep arsitektur dalam pembangunan aplikasi yang membagi aplikasi menjadi 3 bagian besar. Yang mana setiap bagian memiliki tugas-tugas serta tanggung jawab masing-masing. Tiga bagian tersebut adalah: model, view dan controller.

- **Model:** Bertugas untuk mengatur, menyiapkan, memanipulasi dan mengorganisasikan data (dari database) sesuai dengan instruksi dari controller.
- **View:** Bertugas untuk menyajikan informasi (yang mudah dimengerti) kepada user sesuai dengan instruksi dari controller.
- **Controller:** Bertugas untuk mengatur apa yang harus dilakukan model, dan view mana yang harus ditampilkan berdasarkan permintaan dari user



2. Membuat Login

a. Membuat model Member

Buat sebuah file dengan nama **Member.php** pada folder *Models* dan ketikkan kode berikut

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Member extends Model  
{  
    protected $table = 'member';  
  
    public $timestamps = false;  
}
```

b. Membuat model Login

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

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Login extends Model  
{  
    protected $table = 'member_token';  
  
    protected $fillable = ['member_id', 'auth_key'];  
  
    public $timestamps = false;  
}
```

c. Membuat controller Login

Buat sebuah file dengan nama **LoginController** pada folder “app/Http/Controllers” dan ketikkan kode berikut

```
<?php  
  
namespace App\Http\Controllers;  
  
use App\Models\Login;  
use App\Models\Member;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Hash;  
  
class LoginController extends Controller  
{  
    public function login(Request $request)  
    {  
        $email = $request->input('email');  
        $password = $request->input('password');  
  
        $member = Member::query()->firstWhere(['email' => $email]);  
        if ($member == null) {  
            return $this->responseHasil(400, false, 'Email tidak ditemukan');  
        }  
        if (!Hash::check($password, $member->password)) {  
            return $this->responseHasil(400, false, 'Password tidak valid');  
        }  
    }  
}
```

```

$login = Login::create([
    'member_id' => $member->id,
    'auth_key' => $this->RandomString(),
]);
if (!$login) {
    return $this->responseHasil(401, false, 'Unauthorized');
}
$data = [
    'token' => $login->auth_key,
    'user' => [
        'id' => $member->id,
        'email' => $member->email,
    ]
];
return $this->responseHasil(200, true, $data);
}

private function RandomString($length = 100)
{
    $karakter =
'012345678dssd9abcdefghijklmnoprstuvwxyzABCDEFGHIJKLMNPQRSTUVWXYZ';
    $panjang_karakter = strlen($karakter);
    $str = '';
    for ($i = 0; $i < $length; $i++) {
        $str .= $karakter[rand(0, $panjang_karakter - 1)];
    }
    return $str;
}

```

d. Menambahkan route Login

Pada route tambahkan kode untuk mengakses login sehingga menjadi sebagai berikut

```

$route->post('/registrasi', ['uses' => 'RegistrasiController@registerasi']);
$route->post('/login', ['uses' => 'LoginController@login']);

```

e. Mencoba Rest

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

3. CRUD Produk

f. Membuat model Produk

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

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Produk extends Model
{
    protected $table = 'produk';

    protected $fillable = ['kode_produk', 'nama_produk', 'harga'];

    public $timestamps = false;
}
```

g. Membuat controller produk

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

```
<?php

namespace App\Http\Controllers;

use App\Models\Produk;
use Illuminate\Http\Request;

class ProdukController extends Controller
```

```
{  
}  
}
```

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

1. Membuat fungsi create produk

```
public function create(Request $request)  
{  
    $kodeProduk = $request->input('kode_produk');  
    $namaProduk = $request->input('nama_produk');  
    $harga = $request->input('harga');  
  
    $produk = Produk::create([  
        'kode_produk' => $kodeProduk,  
        'nama_produk' => $namaProduk,  
        'harga' => $harga,  
    ]);  
    return $this->responseHasil(200, true, $produk);  
}
```

2. Membuat fungsi list produk

```
public function list()  
{  
    $produk = Produk::all();  
    return $this->responseHasil(200, true, $produk);  
}
```

3. Membuat fungsi tampil produk

```
public function show($id)  
{  
    $produk = Produk::findOrFail($id);  
    return $this->responseHasil(200, true, $produk);  
}
```

4. Membuat fungsi update produk

```
public function update(Request $request, $id)  
{  
    $kodeProduk = $request->input('kode_produk');  
    $namaProduk = $request->input('nama_produk');  
    $harga = $request->input('harga');  
  
    $produk = Produk::findOrFail($id);  
    $result = $produk->update([  
        'kode_produk' => $kodeProduk,
```

```

        'nama_produk' => $namaProduk,
        'harga' => $harga,
    ]);
    return $this->responseHasil(200, true, $result);
}

```

5. Membuat fungsi delete produk

```

public function delete($id)
{
    $produk = Produk::findOrFail($id);
    $delete = $produk->delete();
    return $this->responseHasil(200, true, $delete);
}

```

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

```

<?php

namespace App\Http\Controllers;

use App\Models\Produk;
use Illuminate\Http\Request;

class ProdukController extends Controller
{
    public function create(Request $request)
    {
        $kodeProduk = $request->input('kode_produk');
        $namaProduk = $request->input('nama_produk');
        $harga = $request->input('harga');

        $produk = Produk::create([
            'kode_produk' => $kodeProduk,
            'nama_produk' => $namaProduk,
            'harga' => $harga,
        ]);
        return $this->responseHasil(200, true, $produk);
    }

    public function list()
    {
        $produk = Produk::all();
        return $this->responseHasil(200, true, $produk);
    }

    public function show($id)
    {
        $produk = Produk::findOrFail($id);
        return $this->responseHasil(200, true, $produk);
    }
}

```

```

    }

    public function update(Request $request, $id)
    {
        $kodeProduk = $request->input('kode_produk');
        $namaProduk = $request->input('nama_produk');
        $harga = $request->input('harga');

        $produk = Produk::findOrFail($id);
        $result = $produk->update([
            'kode_produk' => $kodeProduk,
            'nama_produk' => $namaProduk,
            'harga' => $harga,
        ]);
        return $this->responseHasil(200, true, $result);
    }

    public function delete($id)
    {
        $produk = Produk::findOrFail($id);
        $delete = $produk->delete();
        return $this->responseHasil(200, true, $delete);
    }
}

```

6. Menambahkan route produk

Agar ProdukController dapat diakses, selanjutnya tambah route untuk mengakses produk pada file “routes/web.php”

```

$router->group(['prefix' => 'produk'], function ($router) {
    $router->post('/', ['uses' => 'ProdukController@create']);
    $router->get('/', ['uses' => 'ProdukController@list']);
    $router->get('/{id}', ['uses' => 'ProdukController@show']);
    $router->post('/{id}/update', ['uses' => 'ProdukController@update']);
    $router->delete('/{id}', ['uses' => 'ProdukController@delete']);
});

```

Adapun keseluruhan kode pada “routes/web.php” adalah sebagai berikut

```

<?php

/** @var \Laravel\Lumen\Routing\Router $router */

/*
|--------------------------------------------------------------------------
| Application Routes
|--------------------------------------------------------------------------
|
|

```

```

| Here is where you can register all of the routes for an application.
| It is a breeze. Simply tell Lumen the URIs it should respond to
| and give it the Closure to call when that URI is requested.
|
*/

```

```

$router->get('/', function () use ($router) {
    return $router->app->version();
});

$router->post('/registrasi', ['uses' => 'RegistrasiController@register']);
$router->post('/login', ['uses' => 'LoginController@login']);

$router->group(['prefix' => 'produk'], function ($router) {
    $router->post('/', ['uses' => 'ProdukController@create']);
    $router->get('/', ['uses' => 'ProdukController@list']);
    $router->get('/{id}', ['uses' => 'ProdukController@show']);
    $router->post('/{id}/update', ['uses' => 'ProdukController@update']);
    $router->delete('/{id}', ['uses' => 'ProdukController@delete']);
});

```

c. Mencoba Rest

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

The screenshot shows the Postman interface with a POST request to `localhost/toko-api/public/produk`. The request body is set to `form-data` and contains three fields: `kode_produk` (value: A001), `nama_produk` (value: Kulkas Sharp), and `harga` (value: 2500000). The response body is displayed in JSON format, showing a success status with code 200, status true, and a data object containing the product details.

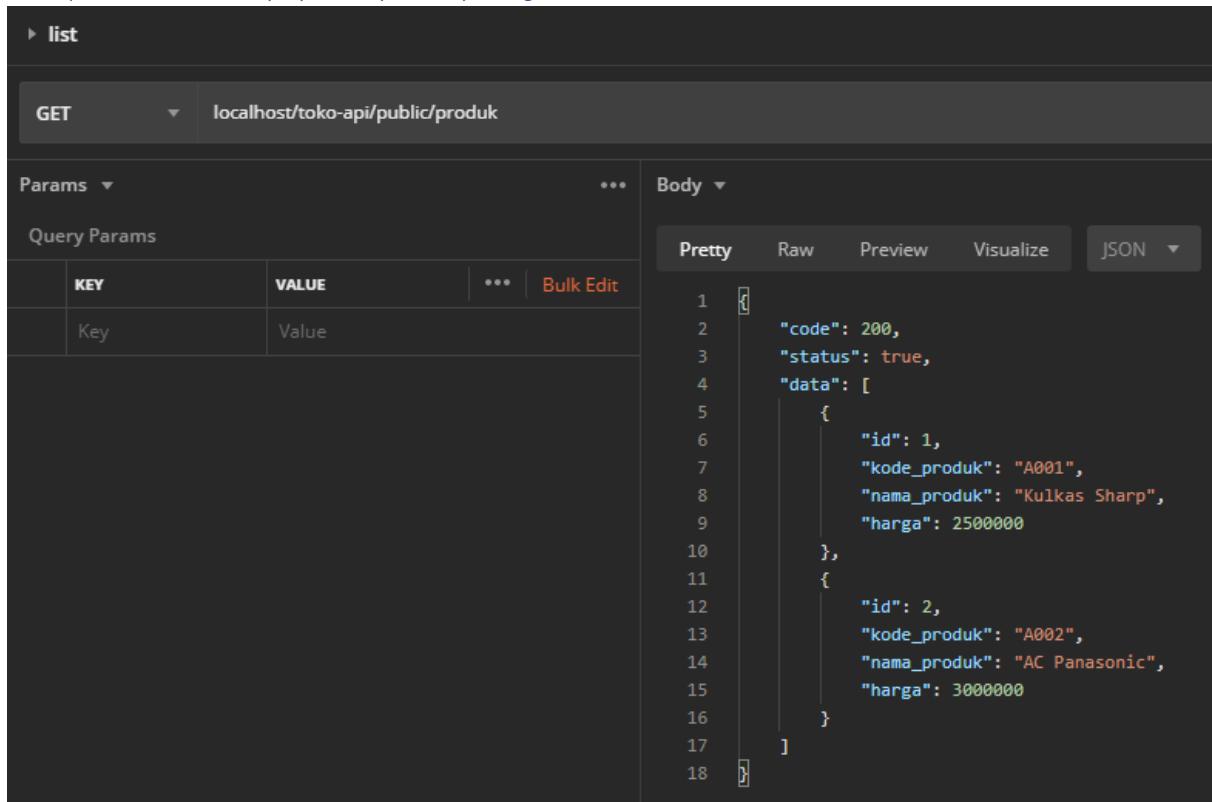
KEY	VALUE
<input checked="" type="checkbox"/> kode_produk	A001
<input checked="" type="checkbox"/> nama_produk	Kulkas Sharp
<input checked="" type="checkbox"/> harga	2500000
Key	Value

```

1   [
2     "code": 200,
3     "status": true,
4     "data": {
5       "kode_produk": "A001",
6       "nama_produk": "Kulkas Sharp",
7       "harga": "2500000",
8       "id": 1
9     }
10    ]

```

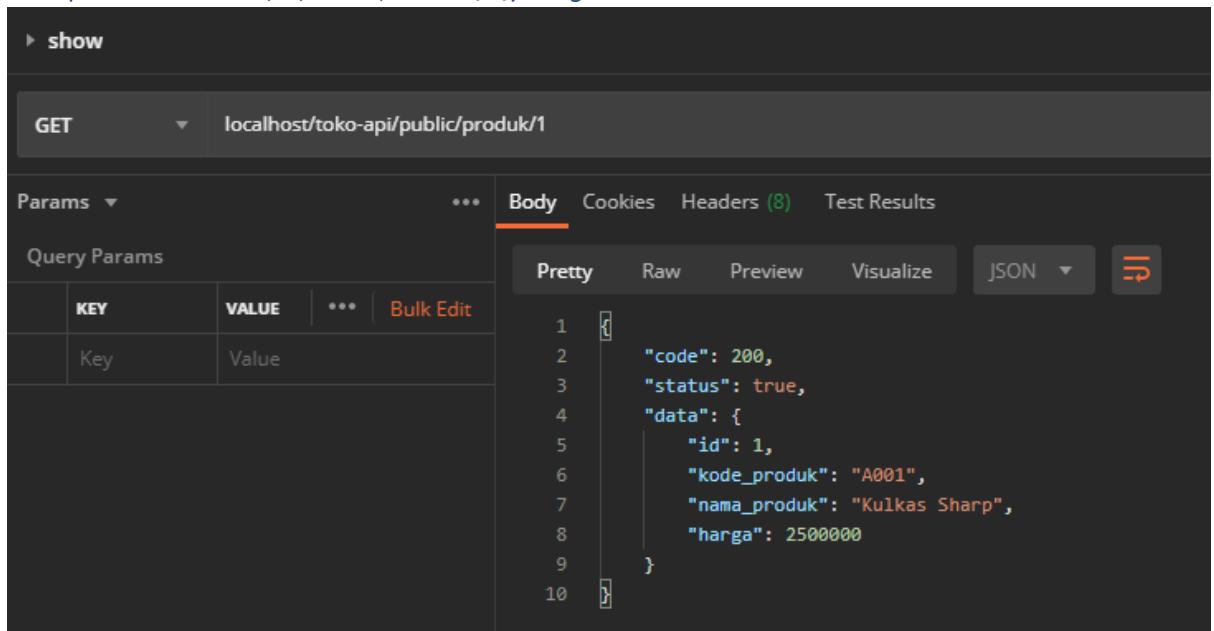
List Produk (`localhost/toko-api/public/produk`) dengan method GET



POSTMAN Screenshot showing the response for the GET request to `localhost/toko-api/public/produk`. The response body is:

```
1 {  
2   "code": 200,  
3   "status": true,  
4   "data": [  
5     {  
6       "id": 1,  
7       "kode_produk": "A001",  
8       "nama_produk": "Kulkas Sharp",  
9       "harga": 2500000  
10    },  
11    {  
12      "id": 2,  
13      "kode_produk": "A002",  
14      "nama_produk": "AC Panasonic",  
15      "harga": 3000000  
16    }  
17  ]  
18 }
```

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



POSTMAN Screenshot showing the response for the GET request to `localhost/toko-api/public/produk/1`. The response body is:

```
1 {  
2   "code": 200,  
3   "status": true,  
4   "data": {  
5     "id": 1,  
6     "kode_produk": "A001",  
7     "nama_produk": "Kulkas Sharp",  
8     "harga": 2500000  
9   }  
10 }
```

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

The screenshot shows a POST request to `localhost/toko-api/public/produk/1/update`. The request body is in form-data format, containing three fields: `kode_produk` (A001), `nama_produk` (Kulkas Polytron), and `harga` (2000000). The response tab shows a JSON object with code 200, status true, and data true.

KEY	VALUE
<input checked="" type="checkbox"/> kode_produk	A001
<input checked="" type="checkbox"/> nama_produk	Kulkas Polytron
<input checked="" type="checkbox"/> harga	2000000
Key	Value

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

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

The screenshot shows a DELETE request to `localhost/toko-api/public/produk/4`. There are no parameters or query params present. The response tab shows a JSON object with code 200, status true, and data true.

KEY	VALUE
Key	Value

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

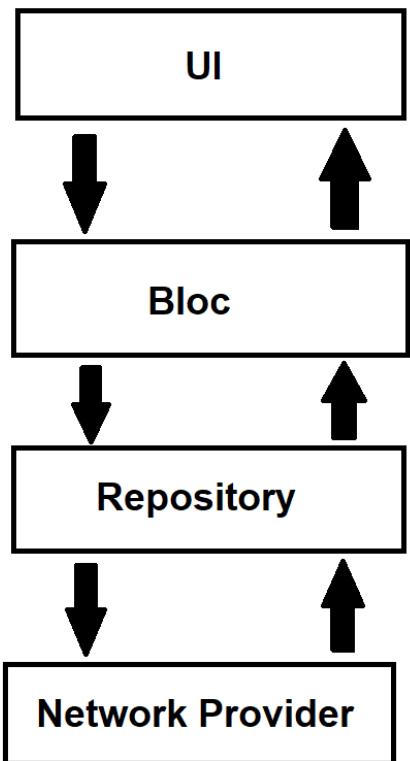
PERTEMUAN 8

Pelaksanaan UTS mata kuliah mobile programming.

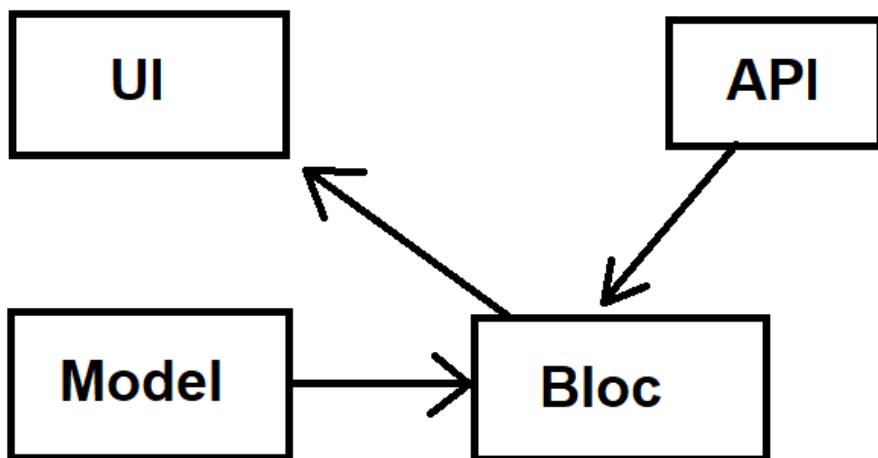
PERTEMUAN 9

1. Membangun Projek Dengan Flutter

Dalam membangun projek aplikasi mobile dengan flutter kali ini, kita menggunakan konsep kerja Design Pattern Bloc sederhana, yang ditampilkan pada gambar dibawah ini.



Untuk proses mengambil data dapat digambarkan sebagai berikut



Bloc akan mengambil data dari API kemudian menterjemahkan kedalam Objek Model yang kemudian ditampilkan ke UI

2. Pengembangan Projek flutter tokokita

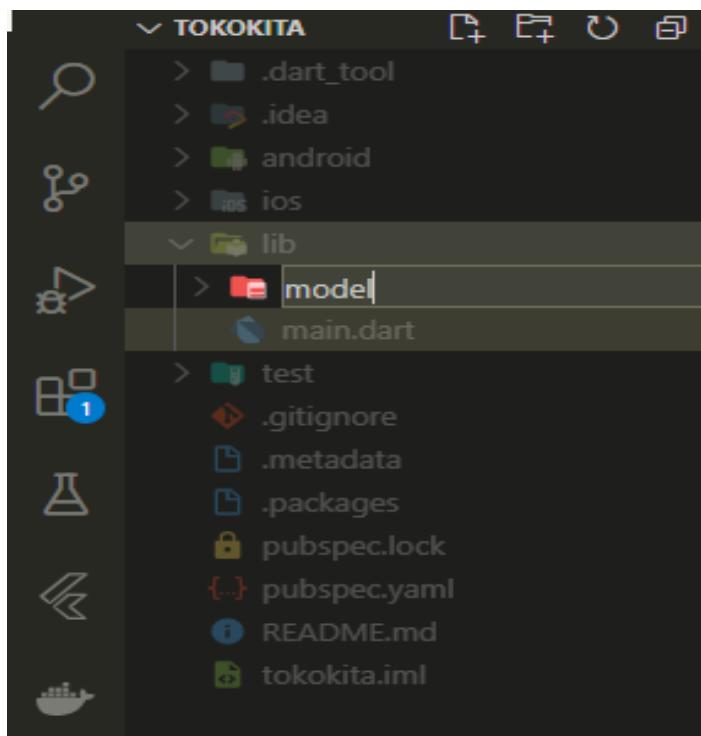
Buat sebuah projek flutter dengan nama **tokokita**

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

- File Explorer:** Shows the project structure under the root folder "TOKOKITA". The "lib" folder is expanded, showing "main.dart" and other files.
- Code Editor:** Displays the content of "main.dart". The code defines the root widget of the application as `MyApp`, which extends `StatelessWidget`. It includes comments explaining the theme and visual density.
- Status Bar:** Shows the current file is "main.dart - tokokita - Visual Studio Code". It also displays build information: Line 8, Col 50, Spaces: 2, UTF-8, CRLF, Dart, Flutter: 1.22.6, Nexus 5X API 29 (android-x86 emulator).

a. Membuat Model

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



b. Membuat Class Login Pada Model

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.  }
19. }
```

c. Membuat Class Registrasi Pada Model

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.   }
15. }
```

d. Membuat Class Produk Pada Model

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

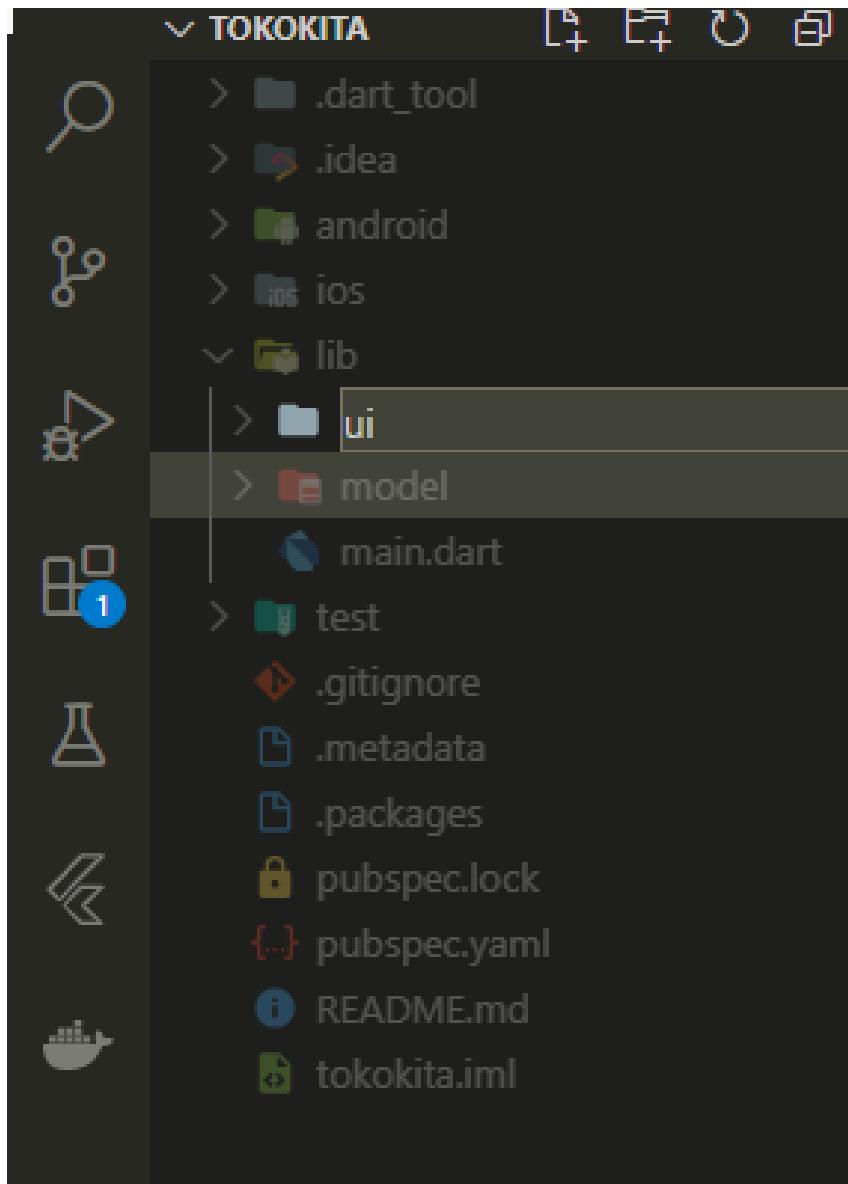
```

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.      id: obj['id'],
12.      kodeProduk: obj['kode_produkt'],
```

```
13.         namaProduk: obj[ 'nama_produk' ],
14.         hargaProduk: obj[ 'harga' ]
15.     );
16. }
17. }
```

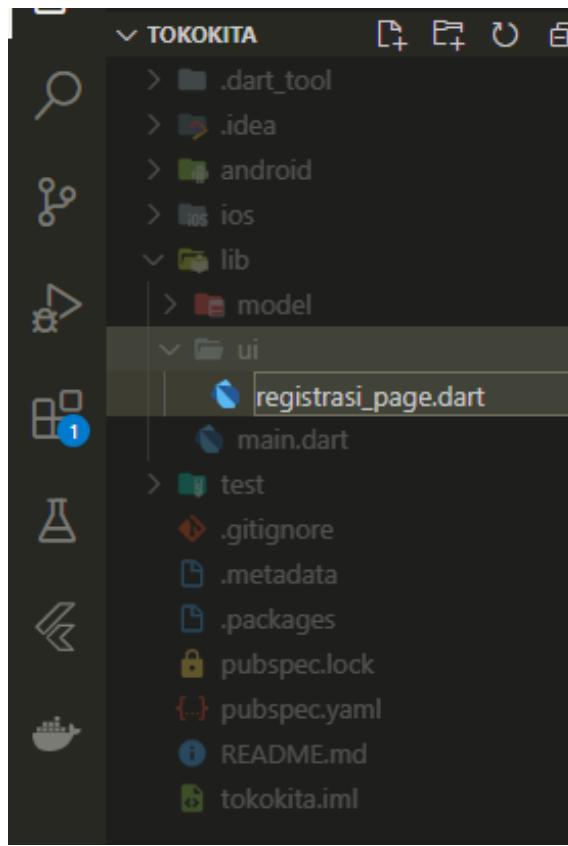
3. Membuat Halaman

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



a. Membuat Form 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.   @override
5.   _RegistrasiPageState createState() => _RegistrasiPageState();
6. }
7.
8. class _RegistrasiPageState extends State<RegistrasiPage> {
9.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
10.  bool _isLoading = false;
11.
12.  final TextEditingController _namaTextboxController = TextEditingController();
13.  final TextEditingController _emailTextboxController = TextEditingController();
14.  final TextEditingController _passwordTextboxController = TextEditingController();
15.
16.  @override
17.  Widget build(BuildContext context) {
18.    return Scaffold(
19.      appBar: AppBar(title: Text("Registrasi"),),
20.      body: SingleChildScrollView(
21.        child: Container(
22.          child: Padding(
23.            padding: const EdgeInsets.all(8.0),
24.            child: Form(
25.              key: _formKey,
26.              child: Column(
27.                mainAxisAlignment: MainAxisAlignment.center,
28.                children: [
29.                  _namaTextField(),
```

```

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

```

```

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

```

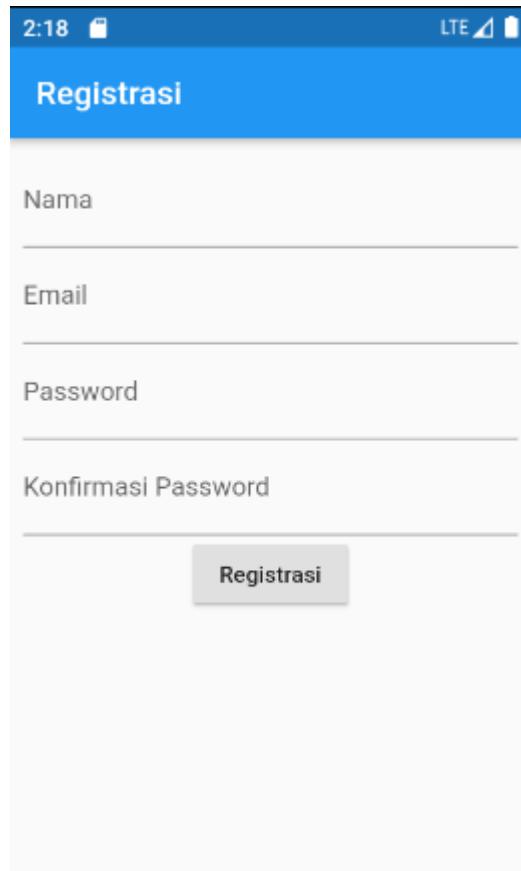
Untuk mencoba halaman `registrasi_page`, buka file `main.dart` kemudian ubah kode menjadi seperti berikut ini

```

1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/registrasi_page.dart';
3.
4. void main() {
5.   runApp(MyApp());
6. }
7.
8. class MyApp extends StatefulWidget {
9.   @override
10.  _MyAppState createState() => _MyAppState();
11. }
12.
13. class _MyAppState extends State<MyApp> {
14.
15.   @override
16.   Widget build(BuildContext context) {
17.     return MaterialApp(
18.       title: 'Toko Kita',
19.       debugShowCheckedModeBanner: false,
20.       home: RegistrasiPage(),
21.     );
22.   }
23. }

```

Dan hasilnya seperti berikut



b. Membuat Form 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.   @override
6.   _LoginPageState createState() => _LoginPageState();
7. }
8.
9. class _LoginPageState extends State<LoginPage> {
10.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
11.   bool _isLoading = false;
12.
13.   final TextEditingController _emailTextboxController = TextEditingController();
14.   final TextEditingController _passwordTextboxController = TextEditingController();
15.
16.   @override
17.   Widget build(BuildContext context) {
18.     return Scaffold(
19.       appBar: AppBar(title: Text("Login")),
20.       body: SingleChildScrollView(
21.         child: Container(
22.           child: Padding(
23.             padding: const EdgeInsets.all(8.0),
24.             child: Form(
25.               key: _formKey,
26.               child: Column(
27.                 children: [
28.                   _emailTextField(),
```

```

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

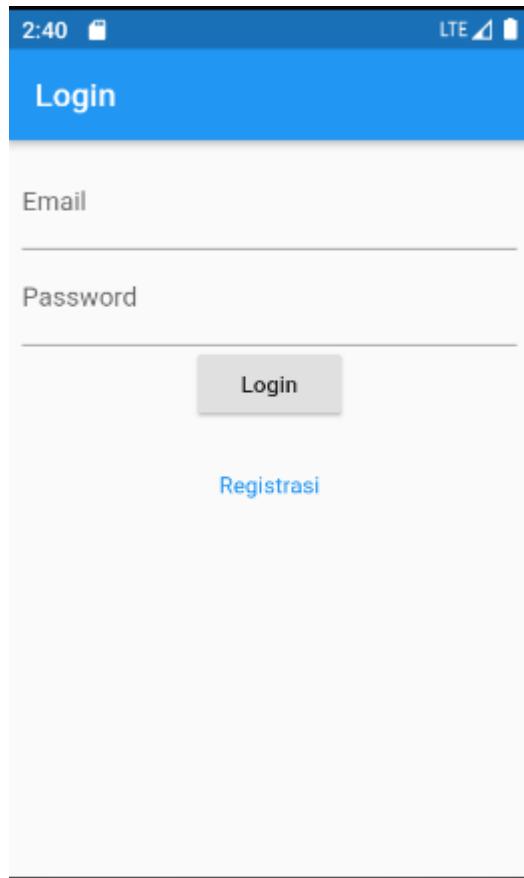
```

```
93.         ),
94.         ),
95.     );
96. }
97. }
```

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(MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   @override
10.  _MyAppState createState() => _MyAppState();
11. }
12.
13. class _MyAppState extends State<MyApp> {
14.
15.   @override
16.   Widget build(BuildContext context) {
17.     return MaterialApp(
18.       title: 'Toko Kita',
19.       debugShowCheckedModeBanner: false,
20.       home: LoginPage(),
21.     );
22.   }
23. }
```

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



c. Membuat 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.   ProdukForm({this.produk});
7.   @override
8.   _ProdukFormState createState() => _ProdukFormState();
9. }
10.
11. class _ProdukFormState extends State<ProdukForm> {
12.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
13.   bool _isLoading = false;
14.   String judul = "TAMBAH PRODUK";
15.   String tombolSubmit = "SIMPAN";
16.
17.   final TextEditingController _kodeProdukTextboxController = TextEditingController();
18.   final TextEditingController _namaProdukTextboxController = TextEditingController();
19.   final TextEditingController _hargaProdukTextboxController = TextEditingController();
20.
21.   @override
22.   void initState() {
23.     // TODO: implement initState
24.     super.initState();
```

```

25.     isUpdate();
26.   }
27.
28.   isUpdate(){
29.     if(widget.produk!=null){
30.       setState(() {
31.         judul = "UBAH PRODUK";
32.         tombolSubmit = "UBAH";
33.         _kodeProdukTextboxController.text = widget.produk.kodeProduk;
34.         _namaProdukTextboxController.text = widget.produk.namaProduk;
35.         _hargaProdukTextboxController.text = widget.produk.hargaProduk.toString();
36.       });
37.     }else{
38.       judul = "TAMBAH PRODUK";
39.       tombolSubmit = "SIMPAN";
40.     }
41.   }
42.
43.   @override
44.   Widget build(BuildContext context) {
45.     return Scaffold(
46.       appBar: AppBar(title: Text(judul)),
47.       body: SingleChildScrollView(
48.         child: Container(
49.           child: Padding(
50.             padding: const EdgeInsets.all(8.0),
51.             child: Form(
52.               key: _formKey,
53.               child: Column(
54.                 children: [
55.                   _kodeProdukTextField(),
56.                   _namaProdukTextField(),
57.                   _hargaProdukTextField(),
58.                   _buttonSubmit()
59.                 ],
60.               ),
61.             ),
62.           ),
63.         ),
64.       ),
65.     );
66.   }
67.
68.   //Membuat Textbox Kode Produk
69.   Widget _kodeProdukTextField() {
70.     return TextFormField(
71.       decoration: 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: 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: 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 RaisedButton(
116.         child: Text(tombolSubmit),
117.         onPressed: () {
118.             var validate = _formKey.currentState.validate();
119.         });
120. }
121. }

```

d. Membuat Form 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.     ProdukDetail({this.produk});
8.     @override
9.     _ProdukDetailState createState() => _ProdukDetailState();
10. }
11.
12. class _ProdukDetailState extends State<ProdukDetail> {
13.     @override
14.     Widget build(BuildContext context) {
15.         return Scaffold(
16.             appBar: AppBar(
17.                 title: Text('Detail Produk'),
18.             ),
19.             body: Center(
20.                 child: Column(
21.                     children: [
22.                         Text(
23.                             "Kode : ${widget.produk.kodeProduk}",
24.                             style: TextStyle(fontSize: 20.0),

```

```

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

```

e. Membuat Tampilan List Produk

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

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

```

2. import 'package:flutter/material.dart';
3. import 'package:tokokita/model/produk.dart';
4. import 'package:tokokita/ui/produk_detail.dart';
5. import 'package:tokokita/ui/produk_form.dart';
6.
7. class ProdukPage extends StatefulWidget {
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: Text('List Produk'),
18.         actions: [
19.           Padding(
20.             padding: EdgeInsets.only(right: 20.0),
21.             child: GestureDetector(
22.               child: Icon(Icons.add, size: 26.0),
23.               onTap: () async {
24.                 Navigator.push(
25.                   context,
26.                   MaterialPageRoute(
27.                     builder: (context) => ProdukForm()));
28.               },
29.             )),
30.           ],
31.         ),
32.       drawer: Drawer(
33.         child: ListView(
34.           children: [
35.             ListTile(
36.               title: Text('Logout'),
37.               trailing: Icon(Icons.logout),
38.               onTap: () async {},
39.             ),
40.           ],
41.         ),
42.       ),
43.       body: ListView(
44.         children: [
45.           ItemProduk(produk: Produk(id: 1, kodeProduk: 'A001', namaProduk: 'Kamera',
46.             hargaProduk: 5000000)),
47.           ItemProduk(produk: Produk(id: 2, kodeProduk: 'A002', namaProduk: 'Kulkas',
48.             hargaProduk: 2500000)),
49.           ItemProduk(produk: Produk(id: 3, kodeProduk: 'A003', namaProduk: 'Mesin Cuci',
50.             hargaProduk: 2000000)),
51.         ],
52.       );
53.     }
54.     class ItemProduk extends StatelessWidget {
55.       final Produk produk;
56.
57.       ItemProduk({this.produk});
58.
59.       @override
60.       Widget build(BuildContext context) {
61.         return Container(
62.           child: GestureDetector(
63.             onTap: () {

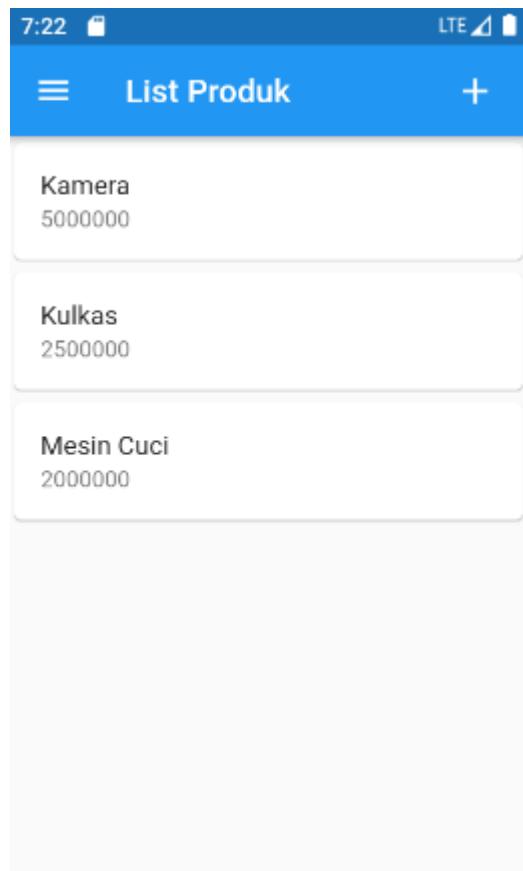
```

```
64.         Navigator.push(
65.             context,
66.             new MaterialPageRoute(
67.                 builder: (context) => ProdukDetail(produk,)));
68.     },
69.     child: Card(
70.         child: ListTile(
71.             title: Text(produk.namaProduk),
72.             subtitle: Text(produk.hargaProduk.toString()),
73.         ),
74.     ),
75. ),
76. );
77. }
78. }
```

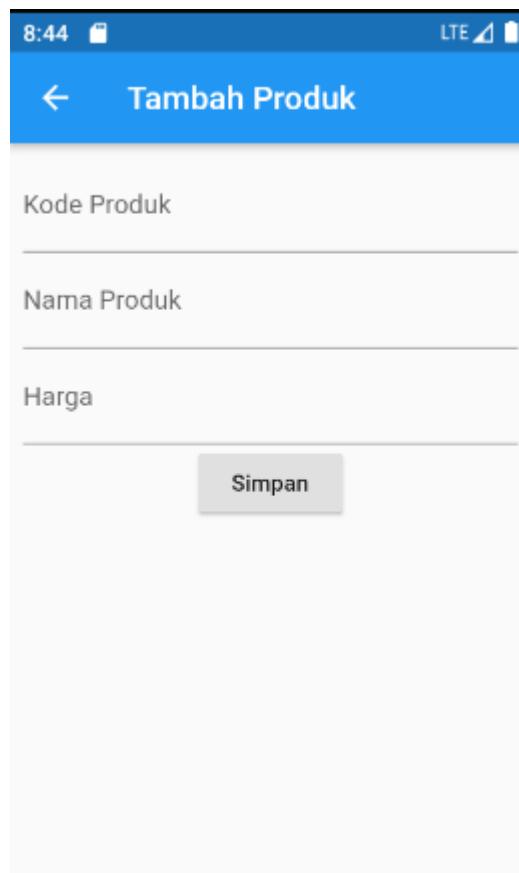
Untuk mencobanya modifikasi file **main.dart** menjadi seperti berikut

```
1. import 'package:flutter/material.dart';
2. import 'package:tokokita/ui/produk_page.dart';
3.
4. void main() {
5.   runApp(MyApp());
6. }
7.
8. class MyApp extends StatefulWidget {
9.   @override
10.  _MyAppState createState() => _MyAppState();
11. }
12.
13. class _MyAppState extends State<MyApp> {
14.
15.   @override
16.   Widget build(BuildContext context) {
17.     return MaterialApp(
18.       title: 'Toko Kita',
19.       debugShowCheckedModeBanner: false,
20.       home: ProdukPage(),
21.     );
22.   }
23. }
```

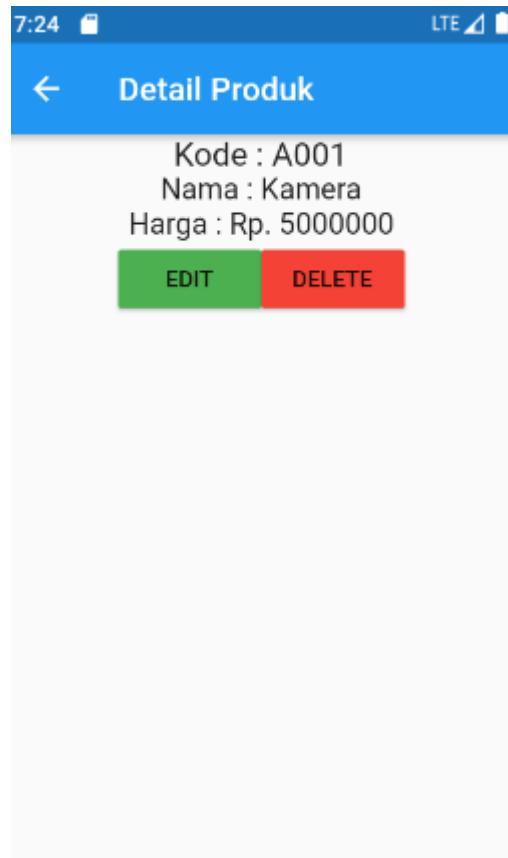
Maka tampilannya akan menjadi seperti berikut



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.

PERTEMUAN 10

1. Membuat Helper Modul

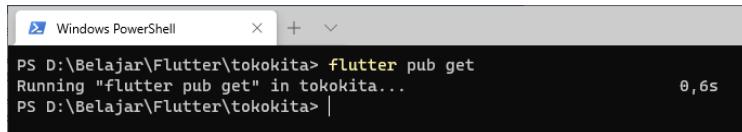
a. Menambahkan dependencies

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 tambahkan kode berikut

```
18. version: 1.0.0+1
19.
20. environment:
21.   sdk: ">=2.7.0 <3.0.0"
22.
23. dependencies:
24.   flutter:
25.     sdk: flutter
26.
27.
28.   # The following adds the Cupertino Icons font to your application.
29.   # Use with the CupertinoIcons class for iOS style icons.
30.   cupertino_icons: ^1.0.0
31.   http: ^0.12.2
32.   shared_preferences: ^0.5.12+2
33.
```

```
34. dev_dependencies:  
35.   flutter_test:  
36.     sdk: flutter  
37.
```

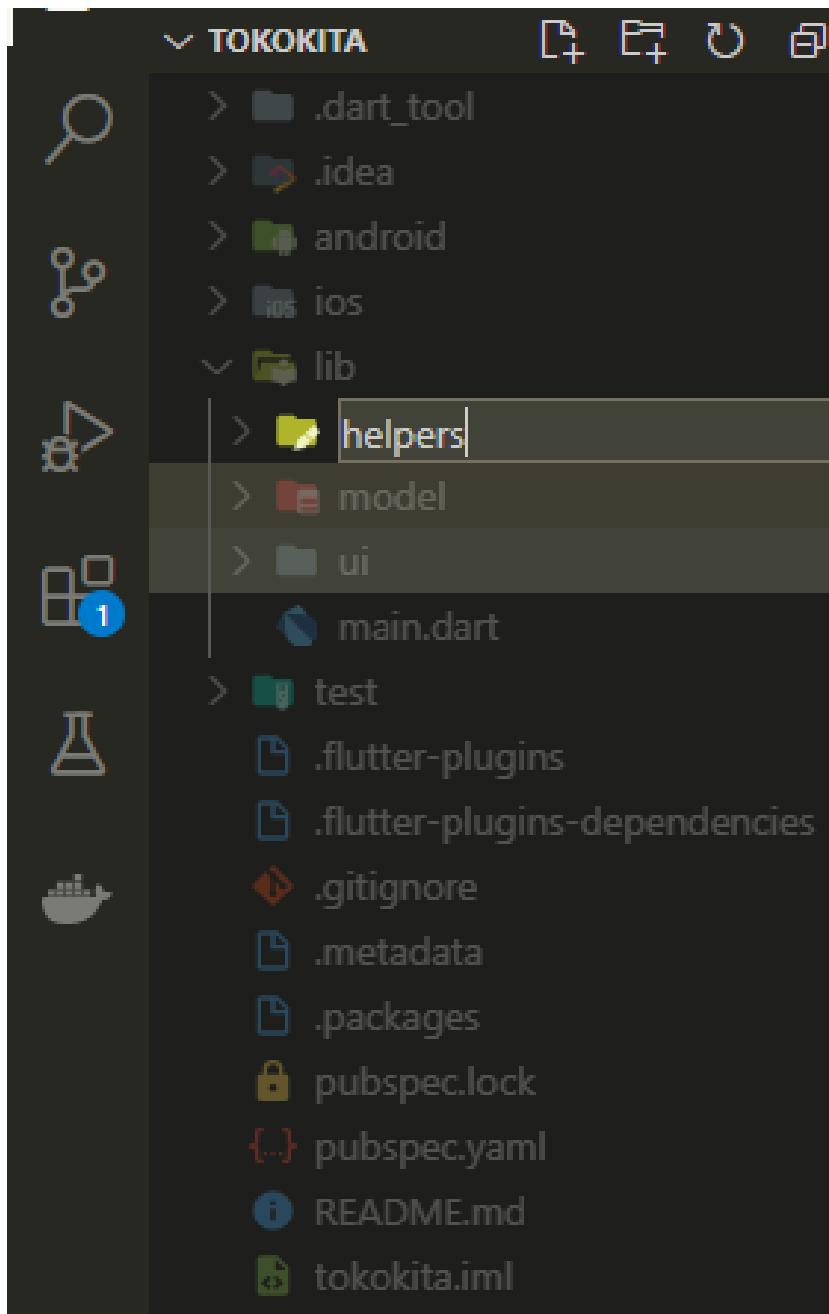
Untuk mengunduh depedencies atau package yang telah ditambahkan, buka **CommandPrompt** kemudian masuk ke folder projek dan ketikkan **flutter pub get** kemudian tekan Enter



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

b. 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. }
```

c. 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.   String toString() {
8.     return "${_prefix}${_message}";
9.   }
10. }
11.
12. class FetchDataException extends AppException {
13.   FetchDataException([String message])
14.     : super(message, "Error During Communication: ");
15. }
16.
17. class BadRequestException extends AppException {
18.   BadRequestException([message]) : super(message, "Invalid Request: ");
19. }
20.
21. class UnauthorisedException extends AppException {
22.   UnauthorisedException([message]) : super(message, "Unauthorised: ");
23. }
24.
25. class UnprocessableEntityException extends AppException {
26.   UnprocessableEntityException([message])
27.     : super(message, "Unprocessable Entity: ");
28. }
29.
30. class InvalidInputException extends AppException {
31.   InvalidInputException([String message]) : super(message, "Invalid Input: ");
32. }
```

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(String url, dynamic data) async {
9.     var token = await UserInfo().getToken();
10.    var responseJson;
11.    try {
12.      final response = await http.post(url, body: data, headers: {
13.        HttpHeaders.authorizationHeader: "Bearer $token"
14.      });
15.      responseJson = _returnResponse(response);
16.    } on SocketException {
17.      throw FetchDataException('No Internet connection');
18.    }
19.    return responseJson;
20.  }
21.
22. Future<dynamic> get(String url) async {
23.   var token = await UserInfo().getToken();
24.   var responseJson;
25.   try {
26.     final response = await http.get(url, headers: {
27.       HttpHeaders.authorizationHeader: "Bearer $token"
28.     });
29.     responseJson = _returnResponse(response);
30.   } on SocketException {
31.     throw FetchDataException('No Internet connection');
32.   }
33.   return responseJson;
34. }
35.
36. Future<dynamic> delete(String url) async {
37.   var token = await UserInfo().getToken();
38.   var responseJson;
39.   try {
40.     final response = await http.delete(url, headers: {
41.       HttpHeaders.authorizationHeader: "Bearer $token"
42.     });
43.     responseJson = _returnResponse(response);
44.   } on SocketException {
45.     throw FetchDataException('No Internet connection');
46.   }
47.   return responseJson;
48. }
49.
```

```

50.    dynamic _returnResponse(http.Response response) {
51.      switch (response.statusCode) {
52.        case 200:
53.          return response;
54.        case 400:
55.          throw BadRequestException(response.body.toString());
56.        case 401:
57.        case 403:
58.          throw UnauthorisedException(response.body.toString());
59.        case 422:
60.          throw InvalidInputException(response.body.toString());
61.        case 500:
62.        default:
63.          throw FetchDataException(
64.              'Error occured while Communication with Server with StatusCode : ${response.statusCode}');
65.      }
66.    }
67.  }

```

Membuat List API url

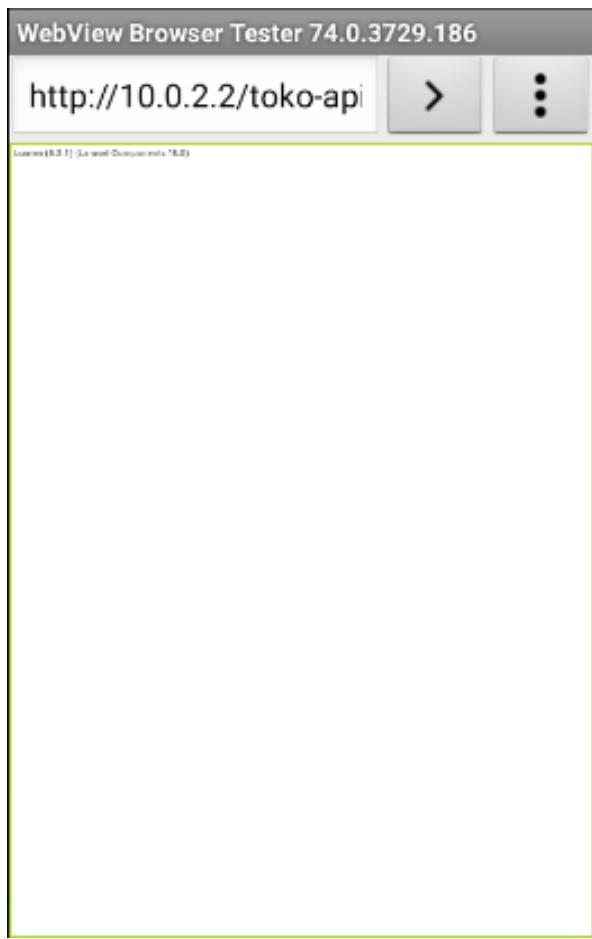
Buat sebuah file di dalam folder **helpers** dengan nama **api_url.dart**, dimana fungsi dari file ini adalah menyimpan alamat-alamat API yang telah dibuat sebelumnya (Endpoint dari API Spec)

```

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. }

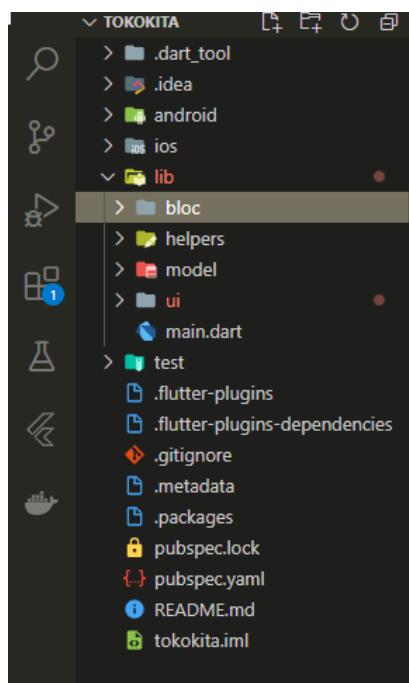
```

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



2. 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.



a. 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({String nama, String email, String password})
9.     async {
10.         String apiUrl = ApiUrl.registrasi;
11.
12.         var body = {
13.             "nama": nama,
14.             "email":email,
15.             "password":password
16.         };
17.
18.         var response = await Api().post(apiUrl, body);
19.         var jsonObj = json.decode(response.body);
20.         return Registrasi.fromJson(jsonObj);
21.     }
21. }
```

b. 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. }
```

c. 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. }
```

d. 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({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. }
56.

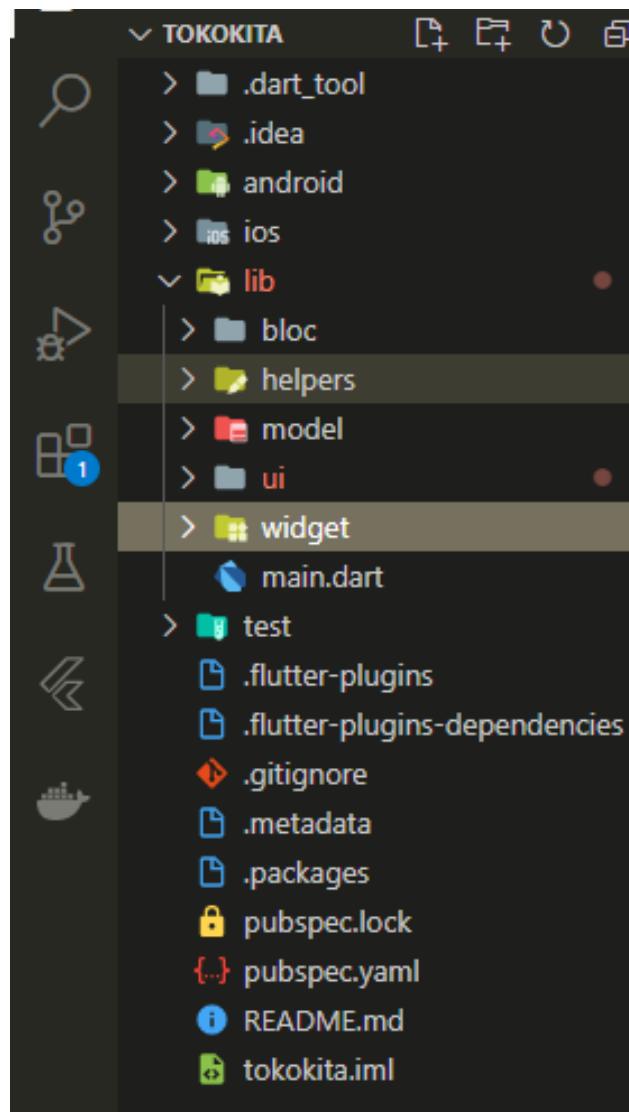
PERTEMUAN 11

1. Menyatukan Fungsionalitas

a. 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 okClick;
13.
14.     SuccessDialog({this.description, this.okClick});
15.
16.     @override
17.     Widget build(BuildContext context) {
18.         return Dialog(
19.             shape: RoundedRectangleBorder(
20.                 borderRadius: BorderRadius.circular(Consts.padding)),
21.             elevation: 0.0,
22.             backgroundColor: Colors.transparent,
23.             child: dialogContent(context),
24.         );
25.     }
26.
27.     dialogContent(BuildContext context) {
28.         return Container(
29.             padding: EdgeInsets.only(
30.                 top: Consts.padding,
31.                 bottom: Consts.padding,
32.                 left: Consts.padding,
33.                 right: Consts.padding,
34.             ),
35.             margin: EdgeInsets.only(top: Consts.avatarRadius),
36.             decoration: new BoxDecoration(
37.                 color: Colors.white,
38.                 shape: BoxShape.rectangle,
39.                 borderRadius: BorderRadius.circular(Consts.padding),
40.                 boxShadow: [
41.                     BoxShadow(
42.                         color: Colors.black26,
43.                         blurRadius: 10.0,
44.                         offset: const Offset(0.0, 10.0),
45.                     ),
46.                 ],
47.             ),
48.             child: Column(
49.                 mainAxisSize: MainAxisSize.min,
50.                 children: [
51.                     Text(
52.                         "SUKSES",
53.                         style: TextStyle(
54.                             fontSize: 24.0,
55.                             fontWeight: FontWeight.w700,
56.                             color: Colors.green),
57.                     ),
58.                     SizedBox(height: 16.0),
59.                     Text(
60.                         description,
61.                         textAlign: TextAlign.center,
62.                         style: TextStyle(
63.                             fontSize: 16.0,
64.                         ),
65.                     ),
66.                     SizedBox(height: 24.0),
67.                     Align(
68.                         alignment: Alignment.bottomRight,
69.                         child: FlatButton(
70.                             onPressed: () {
71.                                 Navigator.of(context).pop(); // To close the dialog
72.                                 okClick();
73.                             },
74.                             child: Text("OK"),
75.                         ),

```

```
76.        )
77.        ],
78.        ),
79.        );
80.    }
81. }
```

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 onClick;
13.
14.   WarningDialog({this.description, this.onClick});
15.
16.   @override
17.   Widget build(BuildContext context) {
18.     return Dialog(
19.       shape: RoundedRectangleBorder(
20.         borderRadius: BorderRadius.circular(Consts.padding)),
21.       elevation: 0.0,
22.       backgroundColor: Colors.transparent,
23.       child: dialogContent(context),
24.     );
25.   }
26.
27.   dialogContent(BuildContext context) {
28.     return Container(
29.       padding: EdgeInsets.only(
30.         top: Consts.padding,
31.         bottom: Consts.padding,
32.         left: Consts.padding,
33.         right: Consts.padding,
34.       ),
35.       margin: EdgeInsets.only(top: Consts.avatarRadius),
36.       decoration: new BoxDecoration(
37.         color: Colors.white,
38.         shape: BoxShape.rectangle,
39.         borderRadius: BorderRadius.circular(Consts.padding),
40.         boxShadow: [
41.           BoxShadow(
42.             color: Colors.black26,
43.             blurRadius: 10.0,
44.             offset: const Offset(0.0, 10.0),
45.           ),
46.         ],
47.       ),
48.       child: Column(
49.         mainAxisSize: MainAxisSize.min,
50.         children: [
51.           Text(
52.             "GAGAL",
53.             style: TextStyle(
54.               fontSize: 24.0, fontWeight: FontWeight.w700, color: Colors.red),
55.           ),
```

```

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

b. 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(MyApp());
8. }
9.
10. class MyApp extends StatefulWidget {
11.   @override
12.   _MyAppState createState() => _MyAppState();
13. }
14.
15. class _MyAppState extends State<MyApp> {
16.   Widget page = CircularProgressIndicator();
17.
18.   @override
19.   void initState() {
20.     super.initState();
21.     isLoggedIn();
22.   }
23.
24.   void isLoggedIn() async {
25.     var token = await UserInfo().getToken();
26.     if(token!=null){
27.       setState(() {
28.         page = ProdukPage(); 
```

```

29.      });
30.    }else{
31.      setState(() {
32.        page = LoginPage();
33.      });
34.    }
35.  }
36.
37. @override
38. Widget build(BuildContext context) {
39.   return MaterialApp(
40.     title: 'Toko Kita',
41.     debugShowCheckedModeBanner: false,
42.     home: page,
43.   );
44. }
45. 
```

c. Modifikasi registrasi_page.dart

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

```

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

```
156.         });
157.         setState(() {
158.             _isLoading = false;
159.         });
160.     }
```

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.     @override
8.     _RegistrasiPageState createState() => _RegistrasiPageState();
9. }
10.
11. class _RegistrasiPageState extends State<RegistrasiPage> {
12.     final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
13.     bool _isLoading = false;
14.
15.     final TextEditingController _namaTextboxController = TextEditingController();
16.     final TextEditingController _emailTextboxController = TextEditingController();
17.     final TextEditingController _passwordTextboxController = TextEditingController();
18.
19.     @override
20.     Widget build(BuildContext context) {
21.         return Scaffold(
22.             appBar: AppBar(title: Text("Registrasi"),),
23.             body: SingleChildScrollView(
24.                 child: Container(
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.
46. //Membuat Textbox Nama
47. Widget _namaTextField() {
48.     return TextFormField(
49.         decoration: InputDecoration(labelText: "Nama"),
50.         keyboardType: TextInputType.text,
51.         controller: _namaTextboxController,
52.         validator: (value){
53.             if(value.length < 3){
```

```

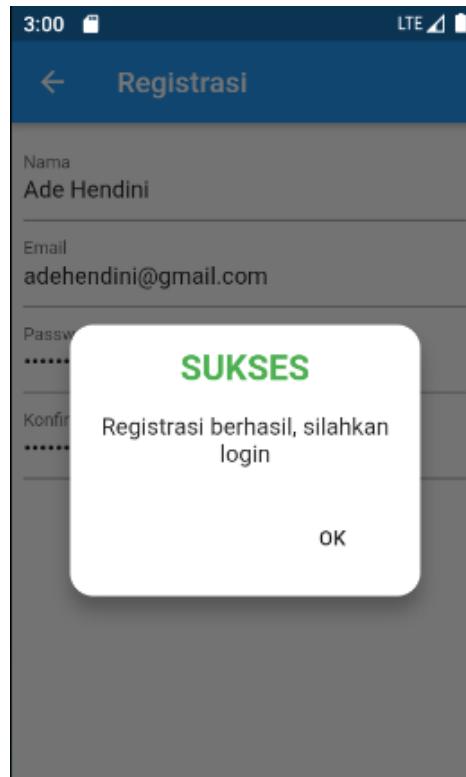
54.         return "Nama harus diisi minimal 3 karakter";
55.     }
56.     return null;
57.   },
58. );
59. }
60.
61. //Membuat Textbox email
62. Widget _emailTextField() {
63.   return TextFormField(
64.     decoration: InputDecoration(labelText: "Email"),
65.     keyboardType: TextInputType.emailAddress,
66.     controller: _emailTextboxController,
67.     validator: (value){
68.       //validasi harus diisi
69.       if(value.isEmpty){
70.         return 'Email harus diisi';
71.       }
72.       //validasi email
73.       Pattern pattern = r'^(([^\<()[]\.,;:\s@"]+(\.[^\<()[]\.,;:\s@"]+)*|(\.
74.       ".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-\0-
75.       9]+\.)+[a-zA-Z]{2,}))$';
76.       RegExp regex = new RegExp(pattern);
77.       if (!regex.hasMatch(value)) {
78.         return "Email tidak valid";
79.       }
80.     },
81.   );
82.
83. //Membuat Textbox password
84. Widget _passwordTextField() {
85.   return TextFormField(
86.     decoration: 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.     },
96.   );
97. }
98. }
99.
100. //membuat textbox Konfirmasi Password
101. Widget _passwordKonfirmasiTextField() {
102.   return TextFormField(
103.     decoration: 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.     },
112.   );
113. }
114. }
115.
116. //Membuat Tombol Registrasi

```

```

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

```



d. 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

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

```

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

```

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.     @override
10.    _LoginPageState createState() => _LoginPageState();
11. }
12.
13. class _LoginPageState extends State<LoginPage> {
14.     final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
15.     bool _isLoading = false;
16.
17.     final TextEditingController _emailTextboxController = TextEditingController();
18.     final TextEditingController _passwordTextboxController = TextEditingController();
19.
20.     @override
21.     Widget build(BuildContext context) {
22.         return Scaffold(
23.             appBar: AppBar(title: Text("Login")),
24.             body: SingleChildScrollView(
25.                 child: Container(
26.                     child: Padding(
27.                         padding: const EdgeInsets.all(8.0),
28.                         child: Form(
29.                             key: _formKey,
30.                             child: Column(
31.                                 children: [
32.                                     _emailTextField(),
33.                                     _passwordTextField(),
34.                                     _buttonLogin(),
35.                                     SizedBox(height: 30,),
36.                                     _menuRegistrasi()
37.                                 ],
38.                             ),
39.                         ),
40.                     ),
41.                 ),
42.             ),
43.         );
44.     }
45.
46. //Membuat Textbox email
47. Widget _emailTextField() {
48.     return TextFormField(
49.         decoration: InputDecoration(labelText: "Email"),
50.         keyboardType: TextInputType.emailAddress,
51.         controller: _emailTextboxController,

```

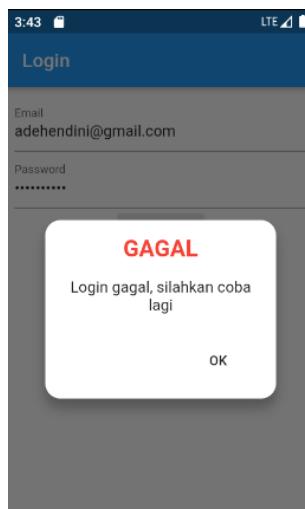
```

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: 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 RaisedButton(
82.         child: Text("Login"),
83.         onPressed: (){
84.             var validate = _formKey.currentState.validate();
85.             if(validate) {
86.                 if(!_isLoading) _submit();
87.             }
88.         });
89. }
90.
91. void _submit() {
92.     _formKey.currentState.save();
93.     setState(() {
94.         _isLoading = true;
95.     });
96.     LoginBloc.login(
97.         email: _emailTextboxController.text,
98.         password: _passwordTextboxController.text
99.     ).then((value) async{
100.         await UserInfo().setToken(value.token);
101.         await UserInfo().setUserID(value.userID);
102.         Navigator.pushReplacement(
103.             context, new MaterialPageRoute(builder: (context) => ProdukPage())
104.         );
105.     }, onError: (error){
106.         print(error);
107.         showDialog(
108.             context: context,
109.             barrierDismissible: false,
110.             builder: (BuildContext context) => WarningDialog(
111.                 description: "Login gagal, silahkan coba lagi",
112.             )
113.         );
114.     });
115.     setState(() {
116.         _isLoading = false;

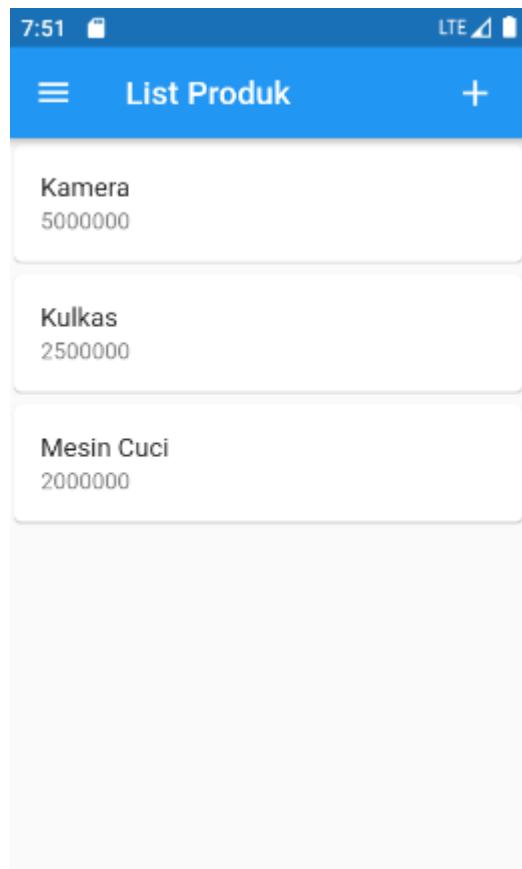
```

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

Jika Gagal akan muncul pesan seperti berikut



Jika berhasil akan menuju ke halaman produk_page.dart



PERTEMUAN 12

1. Modifikasi produk_page.dart

a. Menambahkan fungsi logout pada drawer

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

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

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

```
1. import 'package:flutter/cupertino.dart';
2. import 'package:flutter/material.dart';
3. import 'package:tokokita/bloc/logout_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.   @override
11.   _ProdukPageState createState() => _ProdukPageState();
12. }
13.
14. class _ProdukPageState extends State<ProdukPage> {
15.   @override
16.   Widget build(BuildContext context) {
17.     return Scaffold(
18.       appBar: AppBar(
19.           title: Text('List Produk'),
20.           actions: [
21.               Padding(
22.                   padding: EdgeInsets.only(right: 20.0),
23.                   child: GestureDetector(
24.                       child: Icon(Icons.add, size: 26.0),
25.                       onTap: () async {
26.                           Navigator.push(
27.                               context,
28.                               new MaterialPageRoute(
```

```

29.                 builder: (context) => ProdukForm())));
30.             },
31.         ))
32.     ],
33.   ),
34.   drawer: Drawer(
35.     child: ListView(
36.       children: [
37.         ListTile(
38.           title: Text('Logout'),
39.           trailing: Icon(Icons.logout),
40.           onTap: () async {
41.             await LogoutBloc.logout().then((value) {
42.               Navigator.pushReplacement(context, new MaterialPageRoute(builder:
43.                 (context) => LoginPage()));
44.             });
45.           },
46.         ],
47.       ),
48.     ),
49.     body: ListView(
50.       children: [
51.         ItemProduk(produk: Produk(id: 1, kodeProduk: 'A001', namaProduk: 'Kamera',
52.           hargaProduk: 5000000)),
53.         ItemProduk(produk: Produk(id: 2, kodeProduk: 'A002', namaProduk: 'Kulkas',
54.           hargaProduk: 2500000)),
55.         ItemProduk(produk: Produk(id: 3, kodeProduk: 'A003', namaProduk: 'Mesin Cu-
56. ci', hargaProduk: 2000000)),
57.       ],
58.     );
59.   }
60. class ItemProduk extends StatelessWidget {
61.   final Produk produk;
62.
63.   ItemProduk({this.produk});
64.
65.   @override
66.   Widget build(BuildContext context) {
67.     return Container(
68.       child: GestureDetector(
69.         onTap: () {
70.           Navigator.push(
71.             context,
72.             new MaterialPageRoute(
73.               builder: (context) => ProdukDetail(produk: produk,)));
74.         },
75.         child: Card(
76.           child: ListTile(
77.             title: Text(produk.namaProduk),
78.             subtitle: Text(produk.hargaProduk.toString()),
79.           ),
80.         ),
81.       ),
82.     );
83.   }
84. }
```

b. 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/cupertino.dart';
2. import 'package:flutter/material.dart';
3. import 'package:tokokita/bloc/logout_bloc.dart';
4. import 'package:tokokita/bloc/produk_bloc.dart';
5. import 'package:tokokita/model/produk.dart';
6. import 'package:tokokita/ui/login_page.dart';
7. import 'package:tokokita/ui/produk_detail.dart';
8. import 'package:tokokita/ui/produk_form.dart';
9.
10. class ProdukPage extends StatefulWidget {
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: Text('List Produk'),
21.         actions: [
22.           Padding(
23.             padding: EdgeInsets.only(right: 20.0),
24.             child: GestureDetector(
25.               child: Icon(Icons.add, size: 26.0),
26.               onTap: () async {
27.                 Navigator.push(
28.                   context,
29.                   new MaterialPageRoute(
30.                     builder: (context) => ProdukForm()));
31.               },
32.             )),
33.           ],
34.         ),
35.       drawer: Drawer(
36.         child: ListView(
37.           children: [
38.             ListTile(
39.               title: Text('Logout'),
40.               trailing: Icon(Icons.logout),
41.               onTap: () async {
42.                 await LogoutBloc.logout().then((value) {
43.                   Navigator.pushReplacement(context, new MaterialPageRoute(builder:
44.                     (context) => LoginPage()));
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 ? ListProduk(list: snapshot.data,) : Center(child:
55.             CircularProgressIndicator(),);
56.         },
57.       ),
```

```

57.      );
58.    }
59.  }
60.
61. class ListProduk extends StatelessWidget {
62.   final List list;
63.   ListProduk({this.list});
64.
65.   @override
66.   Widget build(BuildContext context) {
67.     return ListView.builder(
68.       itemCount: list==null ? 0:list.length,
69.       itemBuilder: (context, i){
70.         return ItemProduk(produk: list[i],);
71.       });
72.   }
73. }
74.
75.
76. class ItemProduk extends StatelessWidget {
77.   final Produk produk;
78.
79.   ItemProduk({this.produk});
80.
81.   @override
82.   Widget build(BuildContext context) {
83.     return Container(
84.       child: GestureDetector(
85.         onTap: () {
86.           Navigator.push(
87.             context,
88.             new MaterialPageRoute(
89.               builder: (context) => ProdukDetail(produk: produk,)));
90.         },
91.         child: Card(
92.           child: ListTile(
93.             title: Text(produk.namaProduk),
94.             subtitle: Text(produk.hargaProduk.toString()),
95.           ),
96.         ),
97.       ),
98.     );
99.   }
100. }

```

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

```

61. class ListProduk extends StatelessWidget {
62.   final List list;
63.   ListProduk({this.list});
64.
65.   @override
66.   Widget build(BuildContext context) {
67.     return ListView.builder(
68.       itemCount: list==null ? 0:list.length,
69.       itemBuilder: (context, i){
70.         return ItemProduk(produk: list[i],);
71.       });
72.   }

```

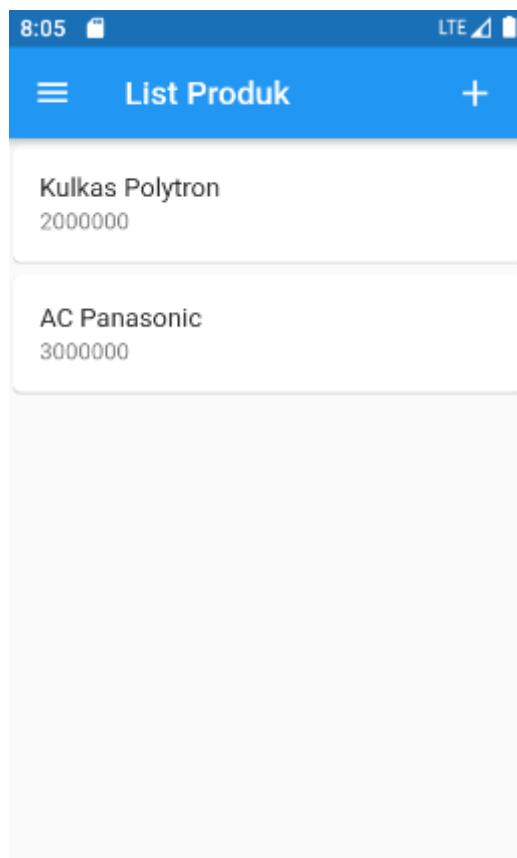
```
|73. }
```

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 ? ListProduk(list: snapshot.data,) : Center(child:
55.                 CircularProgressIndicator(),);
56.             },
56.         ),
```

Serta memasukkan beberapa package yang diperlukan

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



2. Memodifikasi Form Produk (produk_form.dart)

a. 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 RaisedButton(
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 = new Produk();
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(new MaterialPageRoute(builder: (BuildContext context)
=> ProdukPage())));
146.     },onError: (error){
147.         showDialog(
148.             context: context,
149.             builder: (BuildContext context) => WarningDialog(
150.                 description: "Simpan gagal, silahkan coba lagi",
151.             )
152.         );
153.     });
154.     setState(() {
155.         _isLoading = false;
156.     });
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.   ProdukForm({this.produk});
10.  @override
11.  _ProdukFormState createState() => _ProdukFormState();
12. }
13.
14. class _ProdukFormState extends State<ProdukForm> {
15.   final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
16.   bool _isLoading = false;
17.   String judul = "TAMBAH PRODUK";
18.   String tombolSubmit = "SIMPAN";
19.
20.   final TextEditingController _kodeProdukTextboxController = TextEditingController();
21.   final TextEditingController _namaProdukTextboxController = TextEditingController();
22.   final TextEditingController _hargaProdukTextboxController = TextEditingController();
23.
24.   @override
25.   void initState() {
26.     // TODO: implement initState
27.     super.initState();
28.     isUpdate();
29.   }
30.
31.   isUpdate(){
32.     if(widget.produk!=null){
33.       setState(() {
34.         judul = "UBAH PRODUK";
35.         tombolSubmit = "UBAH";
36.         _kodeProdukTextboxController.text = widget.produk.kodeProduk;
37.         _namaProdukTextboxController.text = widget.produk.namaProduk;
38.         _hargaProdukTextboxController.text = widget.produk.hargaProduk.toString();
39.       });
40.     }else{
41.       judul = "TAMBAH PRODUK";
42.       tombolSubmit = "SIMPAN";
43.     }
44.   }
45.
46.   @override
47.   Widget build(BuildContext context) {
48.     return Scaffold(
49.       appBar: AppBar(title: Text(judul)),
50.       body: SingleChildScrollView(
51.         child: Container(
52.           child: Padding(
53.             padding: const EdgeInsets.all(8.0),
54.             child: Form(
55.               key: _formKey,
56.               child: Column(
57.                 children: [
58.                   _kodeProdukTextField(),
59.                   _namaProdukTextField(),
60.                   _hargaProdukTextField(),
61.                   _buttonSubmit()
62.                 ],
63.               ),
64.             ),
65.           ),
66.         ),
67.       ),

```

```

68.    );
69. }
70.
71. //Membuat Textbox Kode Produk
72. Widget _kodeProdukTextField() {
73.     return TextFormField(
74.         decoration: 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: 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: 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 RaisedButton(
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 = new Produk();
141.         createProduk.kodeProduk = _kodeProdukTextboxController.text;
142.         createProduk.namaProduk = _namaProdukTextboxController.text;
143.         createProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text)
144. ;
145.         ProdukBloc.addProduk(produk: createProduk).then((value) {
146.             Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext
147. t context) => ProdukPage())));
148.         },onError: (error){
149.             showDialog(
150.                 context: context,
151.                 builder: (BuildContext context) => WarningDialog(
152.                     description: "Simpan gagal, silahkan coba lagi",
153.                 )
154.             );
155.         });
156.         setState(() {
157.             _isLoading = false;
158.         })
159.     }

```

b. 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 = new Produk();
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(new MaterialPageRoute(builder: (BuildContext context
=> ProdukPage())));
170.     },onError: (error){
171.         showDialog(
172.             context: context,
173.             builder: (BuildContext context) => WarningDialog(
174.                 description: "Permintaan ubah data gagal, silahkan coba lagi",
175.             )
176.         );
177.     });
178.     setState(() {
179.         _isLoading = false;

```

```
180.    });
181. }
```

Kemudian kita tambahkan pada fungsi `_buttonSubmit`

```
116. //Membuat Tombol Simpan/Ubah
117. Widget _buttonSubmit() {
118.     return RaisedButton(
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

```
135. import 'package:flutter/material.dart';
136. import 'package:tokokita/bloc/produk_bloc.dart';
137. import 'package:tokokita/model/produk.dart';
138. import 'package:tokokita/ui/produk_page.dart';
139. import 'package:tokokita/widget/warning_dialog.dart';
140.
141. class ProdukForm extends StatefulWidget {
142.     Produk produk;
143.     ProdukForm({this.produk});
144.     @override
145.     _ProdukFormState createState() => _ProdukFormState();
146. }
147.
148. class _ProdukFormState extends State<ProdukForm> {
149.     final _formKey = GlobalKey<FormState>();
150.     bool _isLoading = false;
151.     String judul = "TAMBAH PRODUK";
152.     String tombolSubmit = "SIMPAN";
153.
154.     final _kodeProdukTextboxController = TextEditingController();
155.     final _namaProdukTextboxController = TextEditingController();
156.     final _hargaProdukTextboxController = TextEditingController();
157.
158.     @override
159.     void initState() {
160.         // TODO: implement initState
161.         super.initState();
162.         isUpdate();
163.     }
164.
165.     isUpdate(){
```

```

166.     if(widget.produk!=null){
167.         setState(() {
168.             judul = "UBAH PRODUK";
169.             tombolSubmit = "UBAH";
170.             _kodeProdukTextboxController.text = widget.produk.kodeProduk;
171.             _namaProdukTextboxController.text = widget.produk.namaProduk;
172.             _hargaProdukTextboxController.text = widget.produk.hargaProduk.toString();
173.         });
174.     }else{
175.         judul = "TAMBAH PRODUK";
176.         tombolSubmit = "SIMPAN";
177.     }
178. }
179.
180. @override
181. Widget build(BuildContext context) {
182.     return Scaffold(
183.         appBar: AppBar(title: Text(judul)),
184.         body: SingleChildScrollView(
185.             child: Container(
186.                 child: Padding(
187.                     padding: const EdgeInsets.all(8.0),
188.                     child: Form(
189.                         key: _formKey,
190.                         child: Column(
191.                             children: [
192.                                 _kodeProdukTextField(),
193.                                 _namaProdukTextField(),
194.                                 _hargaProdukTextField(),
195.                                 _buttonSubmit()
196.                             ],
197.                         ),
198.                     ),
199.                 ),
200.             ),
201.         ),
202.     );
203. }
204.
205. //Membuat Textbox Kode Produk
206. Widget _kodeProdukTextField() {
207.     return TextFormField(
208.         decoration: InputDecoration(labelText: "Kode Produk"),
209.         keyboardType: TextInputType.text,
210.         controller: _kodeProdukTextboxController,
211.         validator: (value) {
212.             if (value.isEmpty) {
213.                 return "Kode Produk harus diisi";
214.             }
215.             return null;
216.         },
217.     );
218. }
219.
220. //Membuat Textbox Nama Produk
221. Widget _namaProdukTextField() {
222.     return TextFormField(
223.         decoration: InputDecoration(labelText: "Nama Produk"),
224.         keyboardType: TextInputType.text,
225.         controller: _namaProdukTextboxController,
226.         validator: (value) {
227.             if (value.isEmpty) {
228.                 return "Nama Produk harus diisi";
229.             }
230.             return null;

```

```

231.         },
232.     );
233. }
234.
235. //Membuat Textbox Harga Produk
236. Widget _hargaProdukTextField() {
237.     return TextFormField(
238.         decoration: InputDecoration(labelText: "Harga"),
239.         keyboardType: TextInputType.number,
240.         controller: _hargaProdukTextboxController,
241.         validator: (value) {
242.             if (value.isEmpty) {
243.                 return "Harga harus diisi";
244.             }
245.             return null;
246.         },
247.     );
248. }
249.
250. //Membuat Tombol Simpan/Ubah
251. Widget _buttonSubmit() {
252.     return RaisedButton(
253.         child: Text(tombolSubmit),
254.         onPressed: () {
255.             var validate = _formKey.currentState.validate();
256.             if(validate) {
257.                 if(!_isLoading){
258.                     if(widget.produk!=null){
259.                         //kondisi update produk
260.                         ubah();
261.                     }else{
262.                         //kondisi tambah produk
263.                         simpan();
264.                     }
265.                 }
266.             }
267.         });
268. }
269.
270. simpan() {
271.     setState(() {
272.         _isLoading = true;
273.     });
274.     Produk createProduk = new Produk();
275.     createProduk.kodeProduk = _kodeProdukTextboxController.text;
276.     createProduk.namaProduk = _namaProdukTextboxController.text;
277.     createProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text);
278.     ProdukBloc.addProduk(produk: createProduk).then((value) {
279.         Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext context
280.         t) => ProdukPage())));
281.     },onError: (error){
282.         showDialog(
283.             context: context,
284.             builder: (BuildContext context) => WarningDialog(
285.                 description: "Simpan gagal, silahkan coba lagi",
286.             )
287.         );
288.         setState(() {
289.             _isLoading = false;
290.         });
291.     }
292.
293.     ubah() {
294.         setState(() {

```

```

295.         _isLoading = true;
296.     });
297.     Produk updateProduk = new Produk();
298.     updateProduk.id = widget.produk.id;
299.     updateProduk.kodeProduk = _kodeProdukTextboxController.text;
300.     updateProduk.namaProduk = _namaProdukTextboxController.text;
301.     updateProduk.hargaProduk = int.parse(_hargaProdukTextboxController.text);
302.     ProdukBloc.updateProduk(produk: updateProduk).then((value) {
303.         Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext context
304.             t) => ProdukPage())));
305.     },onError: (error){
306.         showDialog(
307.             context: context,
308.             builder: (BuildContext context) => WarningDialog(
309.                 description: "Permintaan ubah data gagal, silahkan coba lagi",
310.             )
311.         );
312.     setState(() {
313.         _isLoading = false;
314.     });
315. }
316. }

```

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

```

63. void confirmHapus() {
64.     AlertDialog alertDialog = new AlertDialog(
65.         content: Text("Yakin ingin menghapus data ini?"),
66.         actions: [
67.             //tombol hapus
68.             RaisedButton(
69.                 child: Text("Ya"),
70.                 color: Colors.green,
71.                 onPressed: (){
72.                     ProdukBloc.deleteProduk(id: widget.produk.id).then((value){
73.                         Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext
74.                             t context) => ProdukPage())));
75.                     },onError: (error){
76.                         showDialog(
77.                             context: context,
78.                             builder: (BuildContext context) => WarningDialog(
79.                                 description: "Hapus data gagal, silahkan coba lagi",
80.                             )
81.                         );
82.                     });
83.                 },
84.                 //tombol batal
85.                 RaisedButton(
86.                     child: Text("Batal"),
87.                     color: Colors.red,
88.                     onPressed: ()=>Navigator.pop(context),
89.                 )
90.             ],
91.         );
92. }

```

```
93.     showDialog(context: context, child: alertDialog);
94. }
```

Dengan kode keseluruhan 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_form.dart';
5. import 'package:tokokita/ui/produk_page.dart';
6. import 'package:tokokita/widget/warning_dialog.dart';
7.
8. class ProdukDetail extends StatefulWidget {
9.   Produk produk;
10.  ProdukDetail({this.produk});
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: Text('Detail Produk'),
21.       ),
22.       body: Center(
23.         child: Column(
24.           children: [
25.             Text(
26.               "Kode : ${widget.produk.kodeProduk}",
27.               style: TextStyle(fontSize: 20.0),
28.             ),
29.             Text(
30.               "Nama : ${widget.produk.namaProduk}",
31.               style: TextStyle(fontSize: 18.0),
32.             ),
33.             Text(
34.               "Harga : Rp. ${widget.produk.hargaProduk.toString()}",
35.               style: TextStyle(fontSize: 18.0),
36.             ),
37.             _tombolHapusEdit()
38.           ],
39.         ),
40.       ),
41.     );
42.   }
43.
44. Widget _tombolHapusEdit() {
45.   return Row(
46.     mainAxisAlignment: MainAxisAlignment.end,
47.     children: [
48.       //Tombol Edit
49.       RaisedButton(
50.         child: Text("EDIT"), color: Colors.green, onPressed: () {
51.           Navigator.push(
52.             context,
53.             new MaterialPageRoute(
54.               builder: (context) => ProdukForm(produk: widget.produk,)));
55.         }
56.       )
57.     ],
58.   );
59. }
```

```
55.        }),
56.        //Tombol Hapus
57.        RaisedButton(
58.            child: Text("DELETE"), color: Colors.red, onPressed: ()=>confirmHapus(),
59.        ],
60.    );
61. }
62.
63. void confirmHapus() {
64.     AlertDialog alertDialog = new AlertDialog(
65.         content: Text("Yakin ingin menghapus data ini?"),
66.         actions: [
67.             //tombol hapus
68.             RaisedButton(
69.                 child: Text("Ya"),
70.                 color: Colors.green,
71.                 onPressed: (){
72.                     ProdukBloc.deleteProduk(id: widget.produk.id).then((value){
73.                         Navigator.of(context).push(new MaterialPageRoute(builder: (BuildContext context) => ProdukPage()));
74.                     },onError: (error){
75.                         showDialog(
76.                             context: context,
77.                             builder: (BuildContext context) => WarningDialog(
78.                                 description: "Hapus data gagal, silahkan coba lagi",
79.                             )
80.                         );
81.                     });
82.                 },
83.             ),
84.             //tombol batal
85.             RaisedButton(
86.                 child: Text("Batal"),
87.                 color: Colors.red,
88.                 onPressed: ()=>Navigator.pop(context),
89.             )
90.         ],
91.     );
92.
93.     showDialog(context: context, child: alertDialog);
94. }
95. }
```

PERTEMUAN 13-15

Pada pertemuan ini mahasiswa melakukan presentasi terhadap projek yang telah dihasilkan dan dikembangkan dari modul sebagai hasil dari nilai UAS.