UIN Maulana
Malik Ibrahim
Malang

MODUL PRAKTIKUM

MOBILE PROGRAMMING

ALLIN JUNIKHAH, M.T.

DAFTAR ISI

Modul 1

Pemrograman Native dengan Java : Installasi Software

1. PERSIAPAN ANDROID

Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak, seperti *smartphone* dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc. dengan dukungan finansial dari Google yang kemudian membelinya pada tahun 2005. Pada perkembangan lebih lanjut, Google mengembangkan Android TV, Android Auto serta Wear OS (smartwatch).

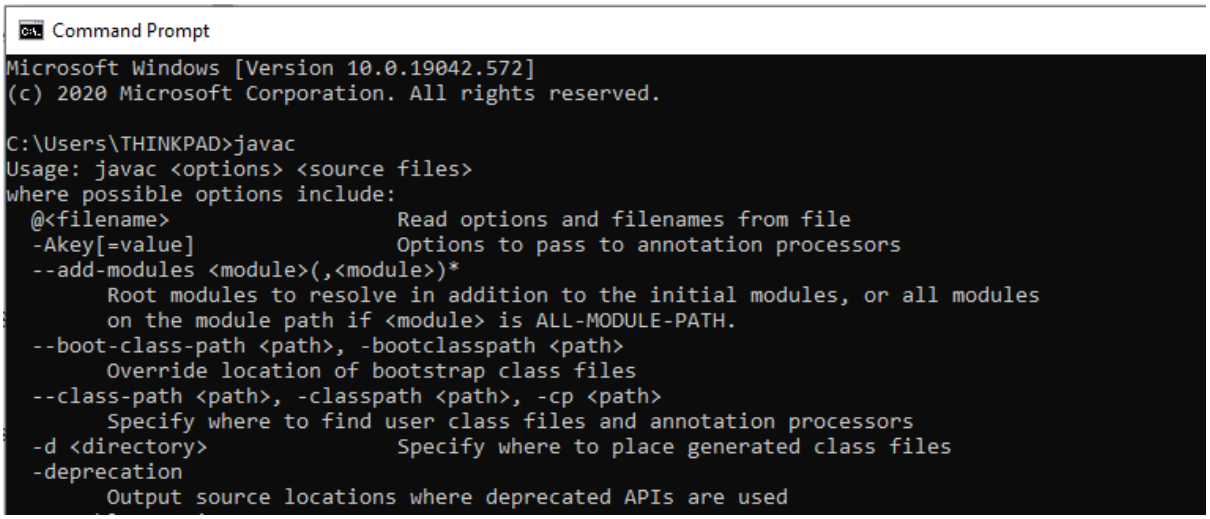
Android Codenames

Pada pengembangan rilis Android menggunakan codename secara alfabetis yang diinspirasi oleh nama makanan. API level menentukan teknis implementasi kode yang berbeda.

Codename	Version	API level/NDK release
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19

Instalasi JAVA SDK

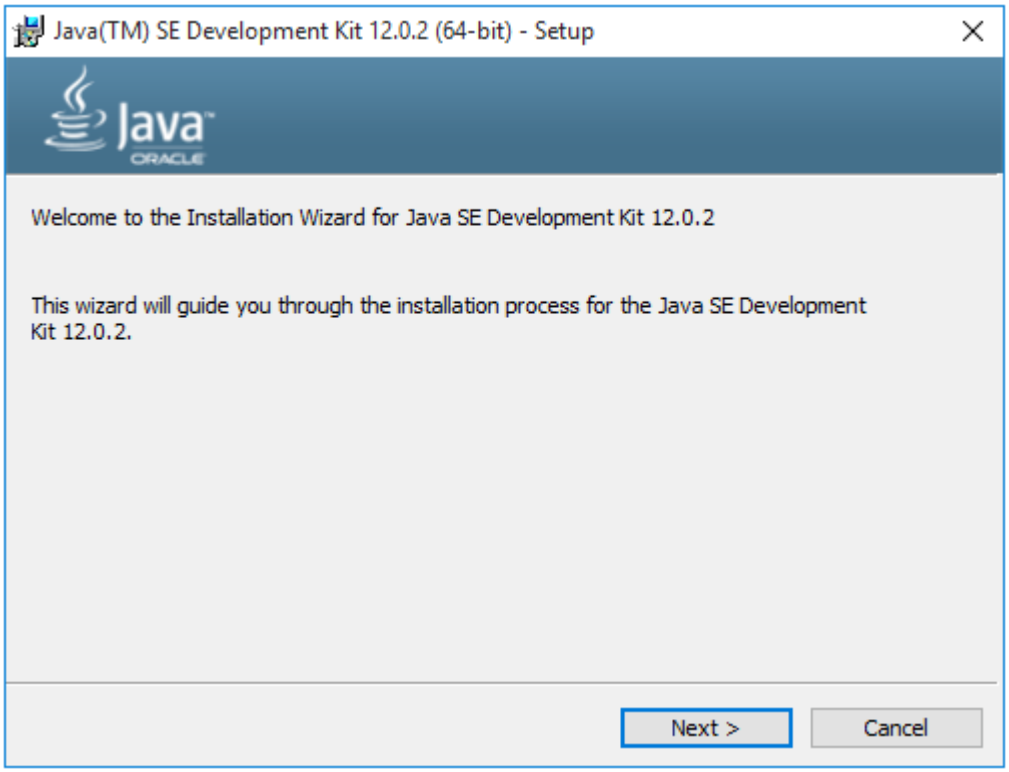
Lewati modul ini jika anda sudah memiliki Java SDK di komputer anda, untuk mengecek installasi Java pada komputer silahkan buka terminal (command prompt) dan ketikkan command `java` dan `javac` jika kedua perintah tersebut menampilkan hasil yang benar maka lewati bagian ini. Contoh hasil terminal jika JDK belum terinstall dengan benar.

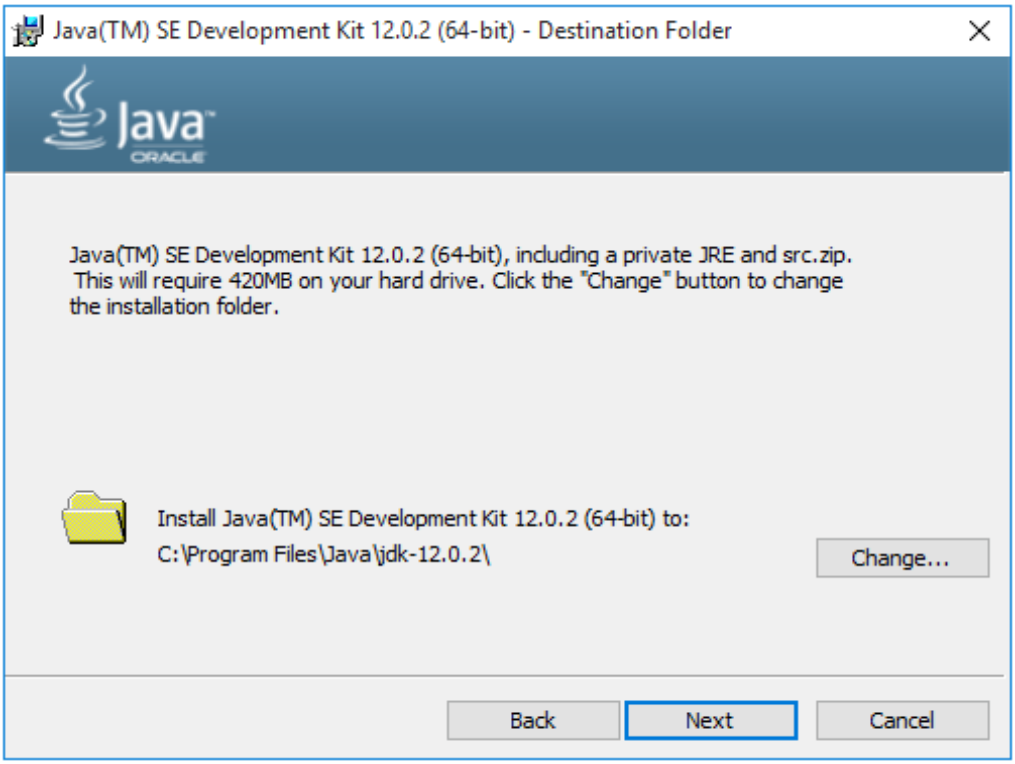


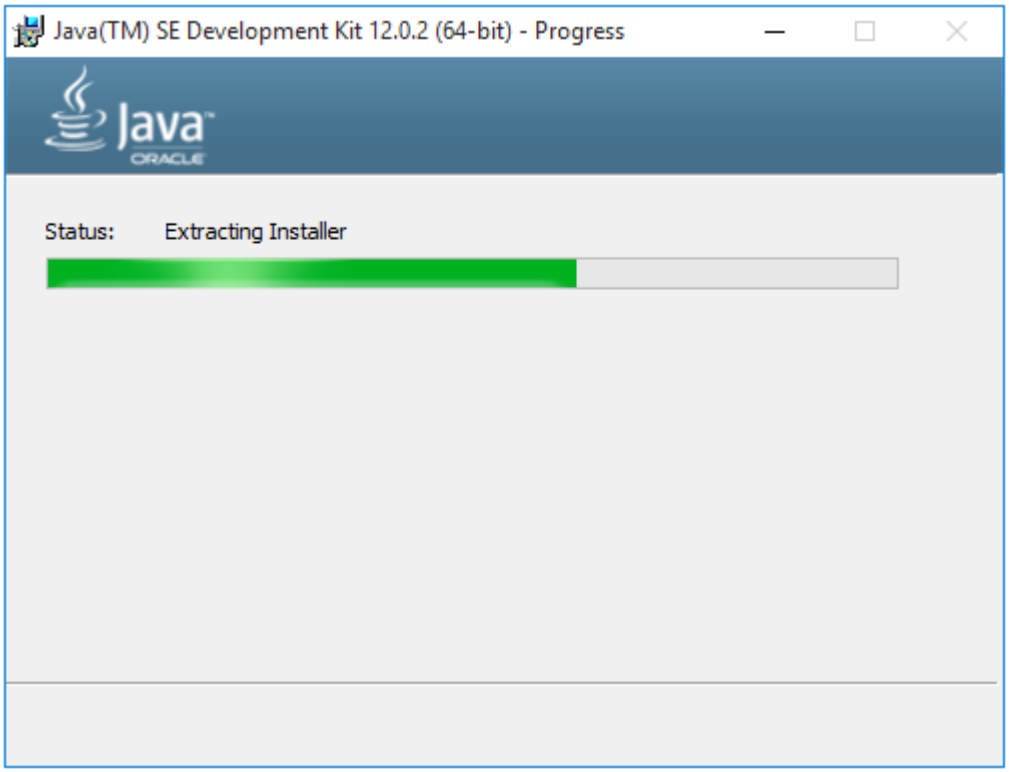
Jika kedua perintah tersebut gagal dilakukan berarti anda belum menginstall JRE dan JDK dengan benar.

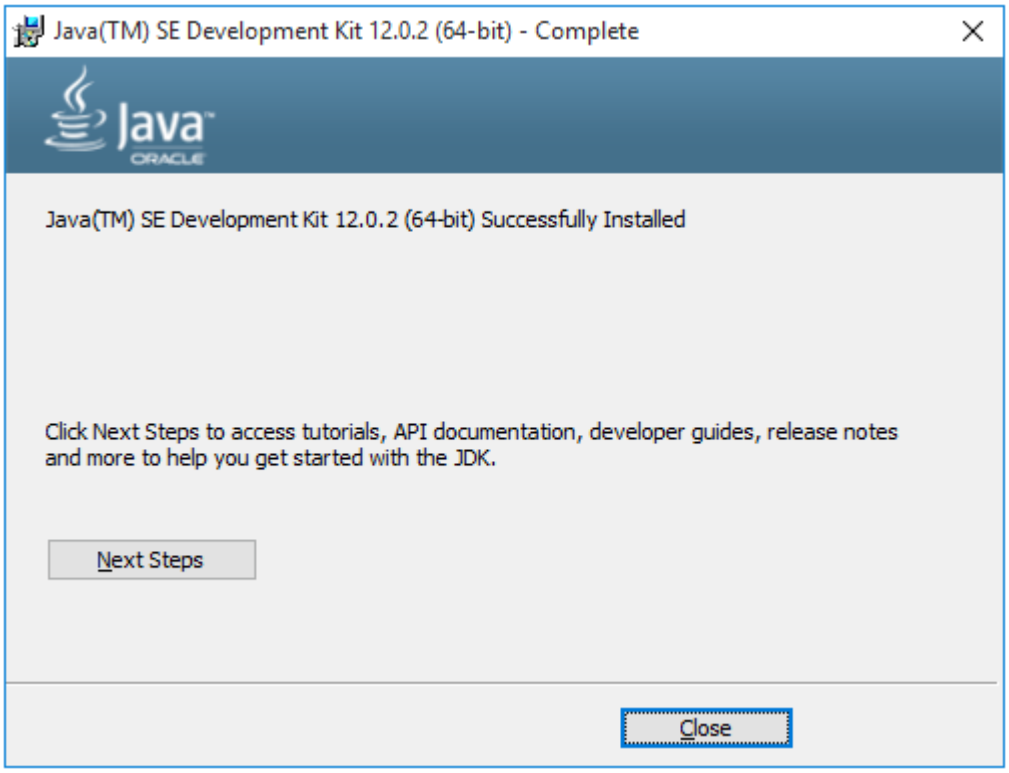
Langkah-langkah installasi

Klik Installer dan ikuti langkah langkahnya




Java(TM) SE Development Kit 12.0.2 (64-bit) - Destination Folder
Java
ORACLE
Java(TM) SE Development Kit 12.0.2 (64-bit), including a private JRE and src.zip.
This will require 420MB on your hard drive. Click the "Change" button to change
the installation folder.
Install Java(TM) SE Development Kit 12.0.2 (64-bit) to:
C:\Program Files\Java\jdk-12.0.2\
Change...
Back
Next
Cancel


Java(TM) SE Development Kit 12.0.2 (64-bit) - Progress
Java
ORACLE
Status:
Extracting Installer



Instalasi Android Studio

Kebutuhan

1. RAM minimal 4 GB, direkomendasikan 8 GB
2. Minimal penyimpanan 2 GB, direkomendasikan 4 GB
3. Minimum resolusi layar 1280 x 800

Langkah-langkah

Untuk melakukan instalasi Android Studio, silahkan mengikuti langkah-langkah pada tautan berikut:

https://developer.android.com/studio/install

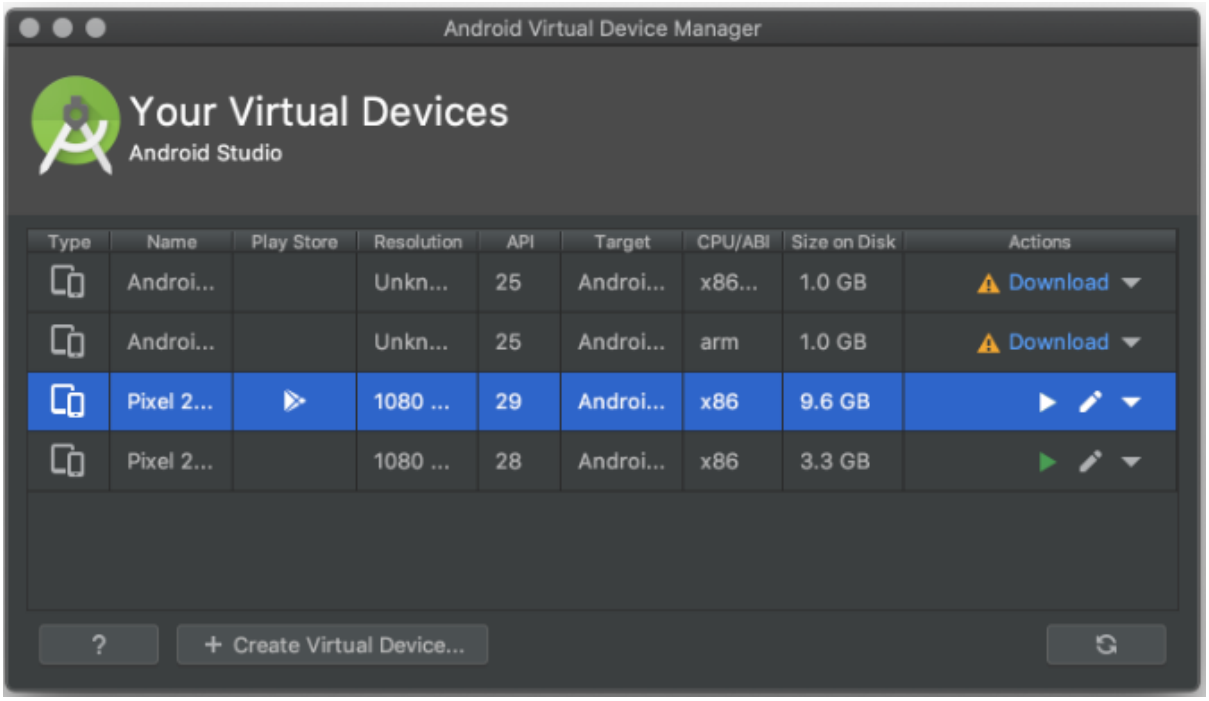
Sebelum melakukan instalasi, pastikan bahwa JDK (*Java Development Kit)* telah terinstall pada komputer anda.

Instalasi Emulator Android

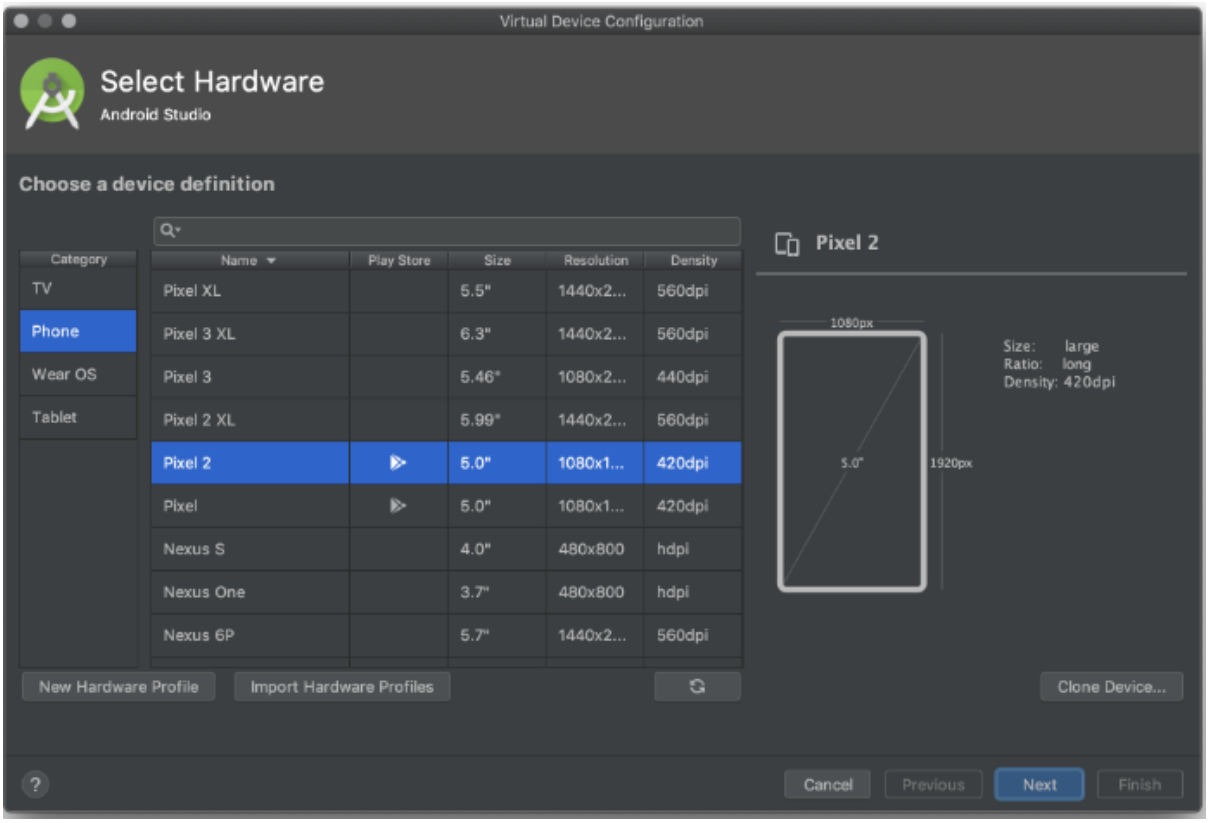
Langkah-langkah

Untuk memudahkan pengembangan aplikasi Android anda dapat menggunakan Emulator.

- Silahkan buka aplikasi **Android Studio**
- Pada tampilan Welcome Screen, carilah menu **Configure** yang ada pada pojok bawah.
- Pilih menu **AVD Manager**
- Pada tampilan AVD Manager, tekan tombol **Create Virtual Device**



- Pilih Device sesuai kebutuhan anda, kemudian tekan tombol **Next**

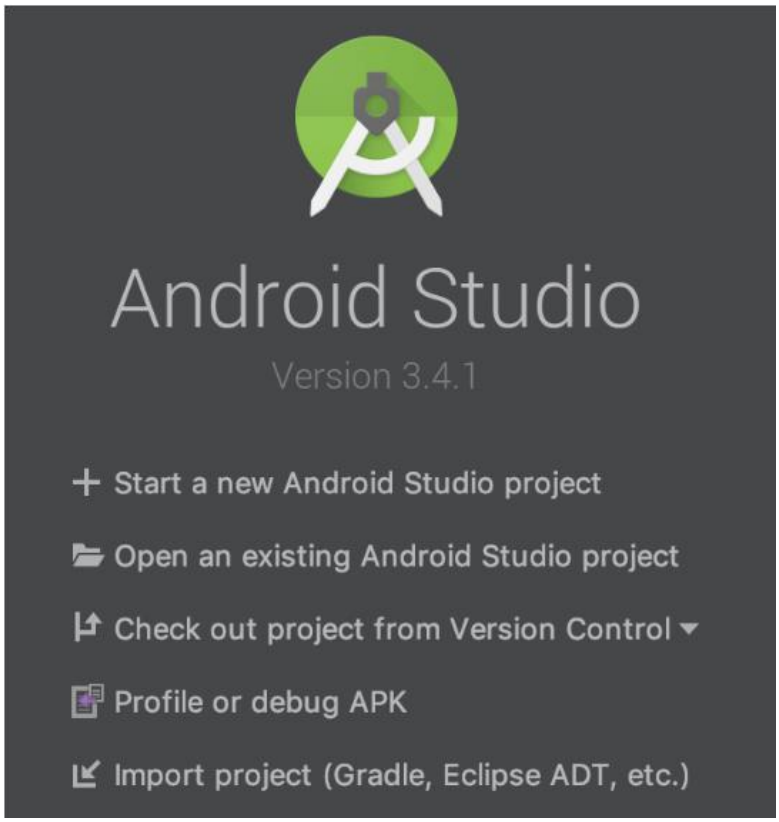


- Pilih Versi API, kemudian tekan tombol **Next**
- Beri nama **AVD Name** anda, kemudian tekan tombol **Finish**

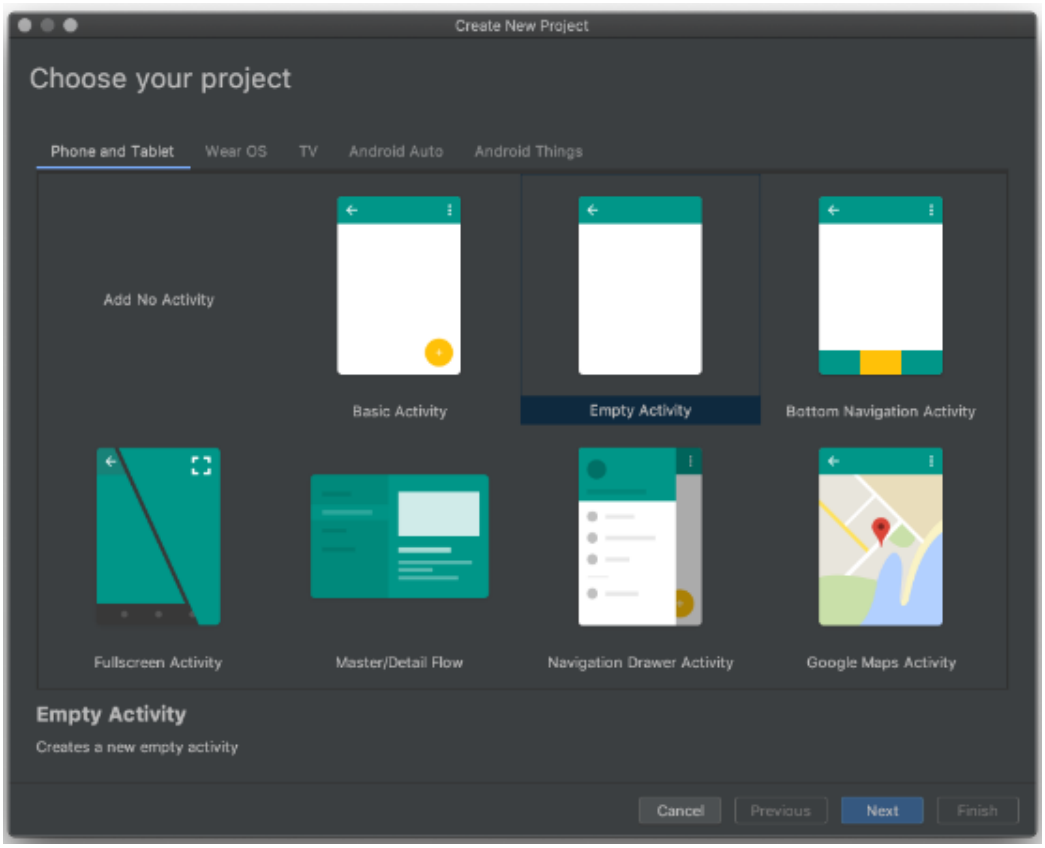
Hello Android

Pada bagian ini anda akan mempelajari langkah-langkah untuk membuat project Android. Silahkan ikuti langkah-langkah di bawah ini:

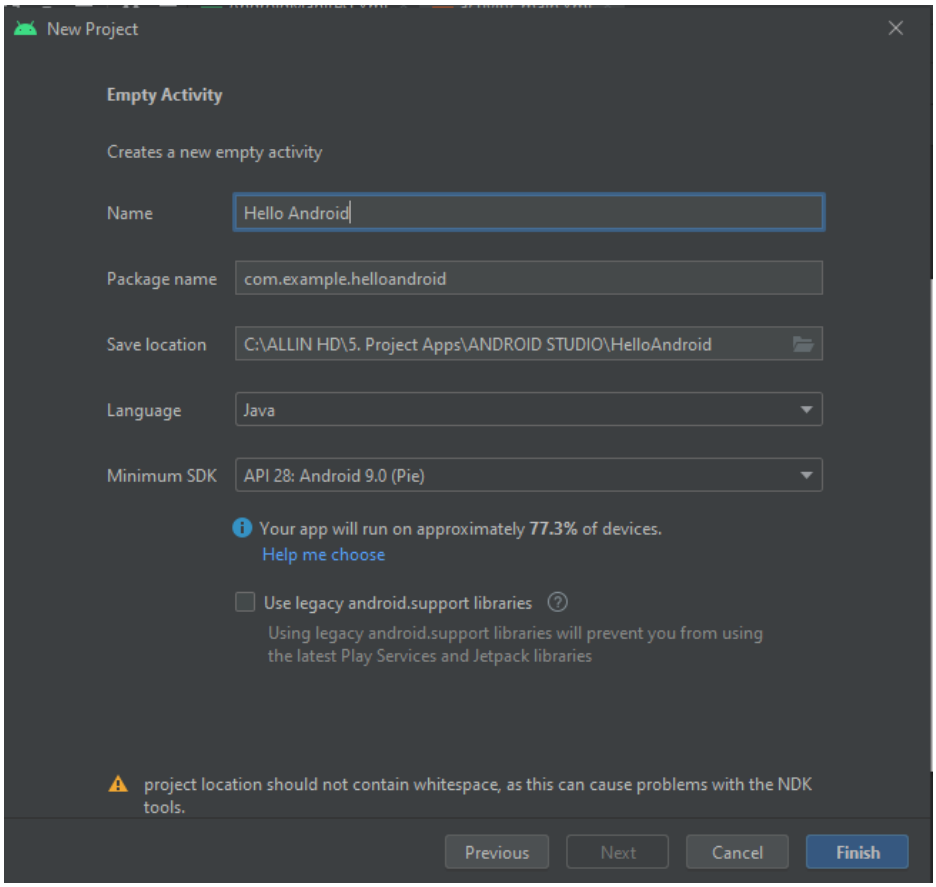
- Pada **Welcome Screen** pilih menu **Start a new Android Studio project**



- Pilih **Empty Activity**



- Beri nama project Android anda **Hello Android**



- Pilih bahasa yang digunakan yaitu: Java
- Tunggu proses build hingga selesai.
- Jalankan project anda dengan menekan tombol segitiga hijau.

- Deploy aplikasi anda dengan target device yang diinginkan.

2. PENGENALAN ANDROID STUDIO

Pendahuluan

Pada praktikum kali ini anda akan mempelajari lebih dalam mengenai Android Studio, Git, struktur project Android dan pada aplikasi Android.

Tujuan Pembelajaran

- Mahasiswa mampu menggunakan Android Studio dengan baik (menu, window, shortcut)
- Mahasiswa memahami struktur folder project android dan kegunaan file pada masing masing folder
- Mahasiswa mampu mengintegrasikan Git dengan Android Studio

Alat dan Bahan

1. Laptop atau PC
2. Android Studio

Tour Android Studio

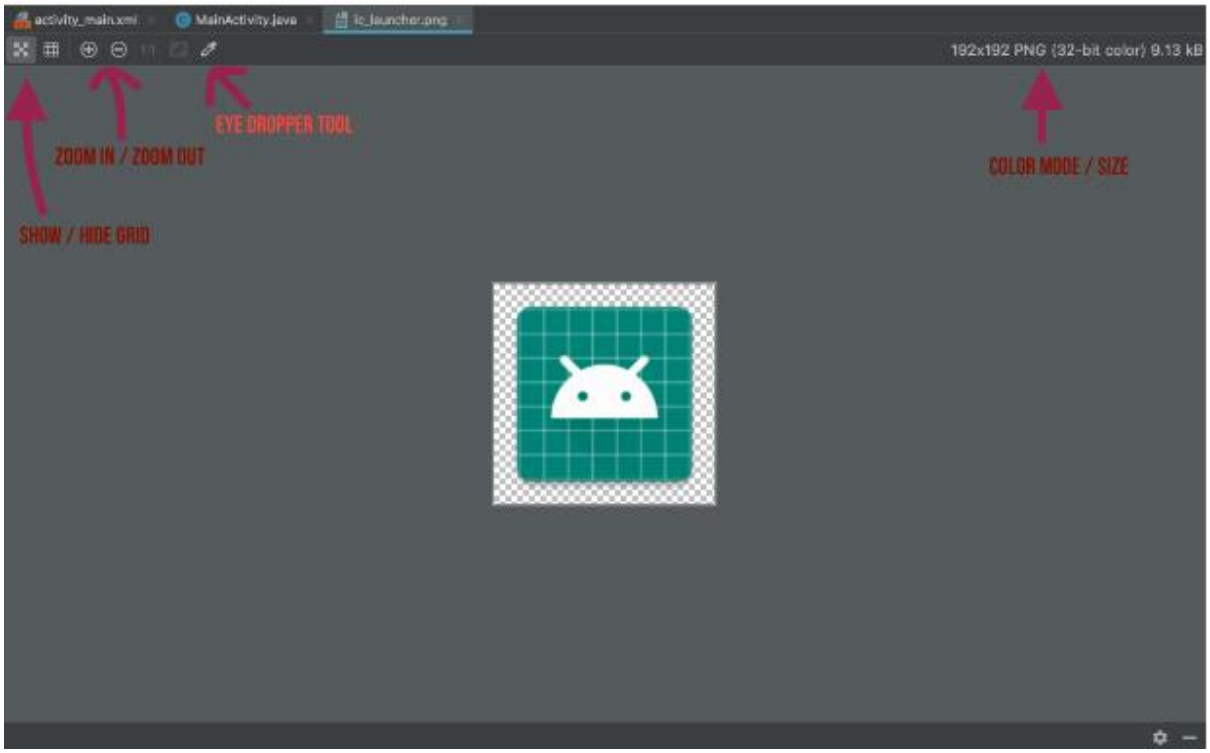
Pada sub bab ini anda akan mempelajari mengenai cara menggunakan android studio dengan baik, seperti shortcut, generator, dan layout layout penting pada Android Studio.

Layout

Tampilan layout android studio dapat berubah ubah sesuai dengan file yang sedang kita buka. Pada umumnya ada tiga jenis file yang akan sering dibuka ketika membuat sebuah aplikasi android yaitu gambar, java, dan xml.

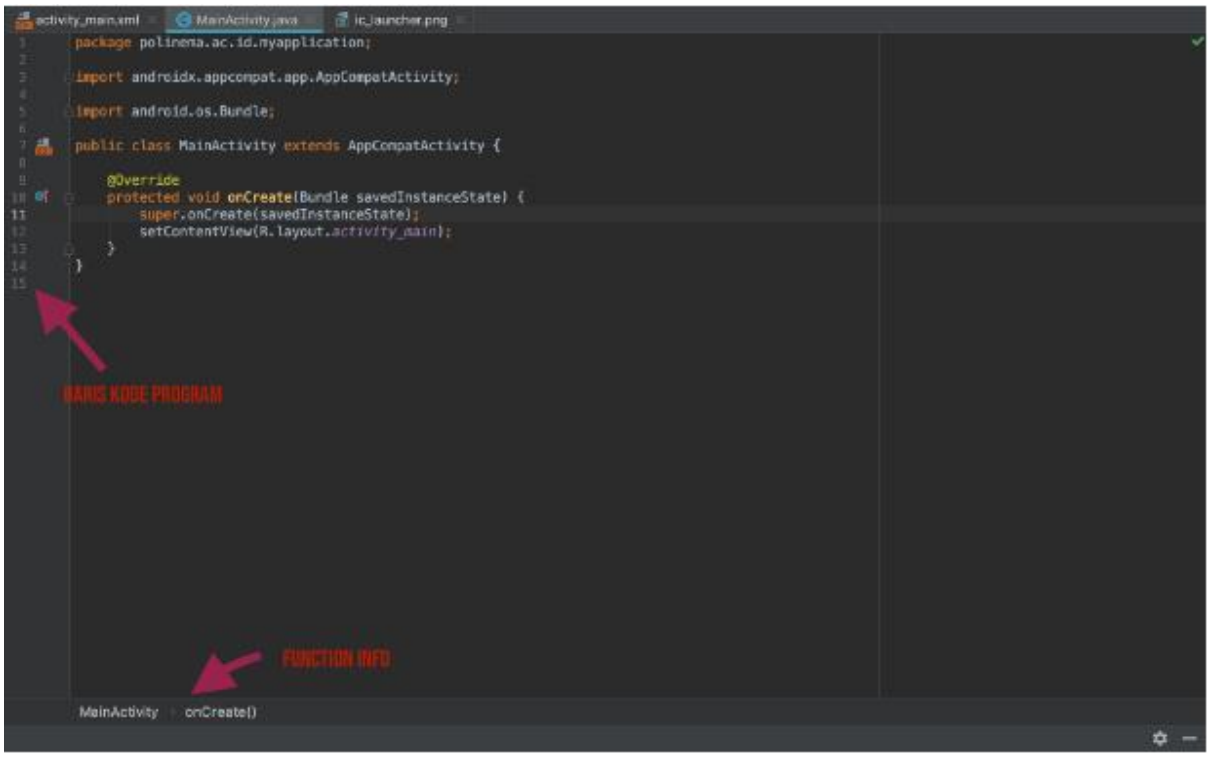
Layout file gambar

Berikut ini layout editor android ketika anda membuka sebuah file gambar.



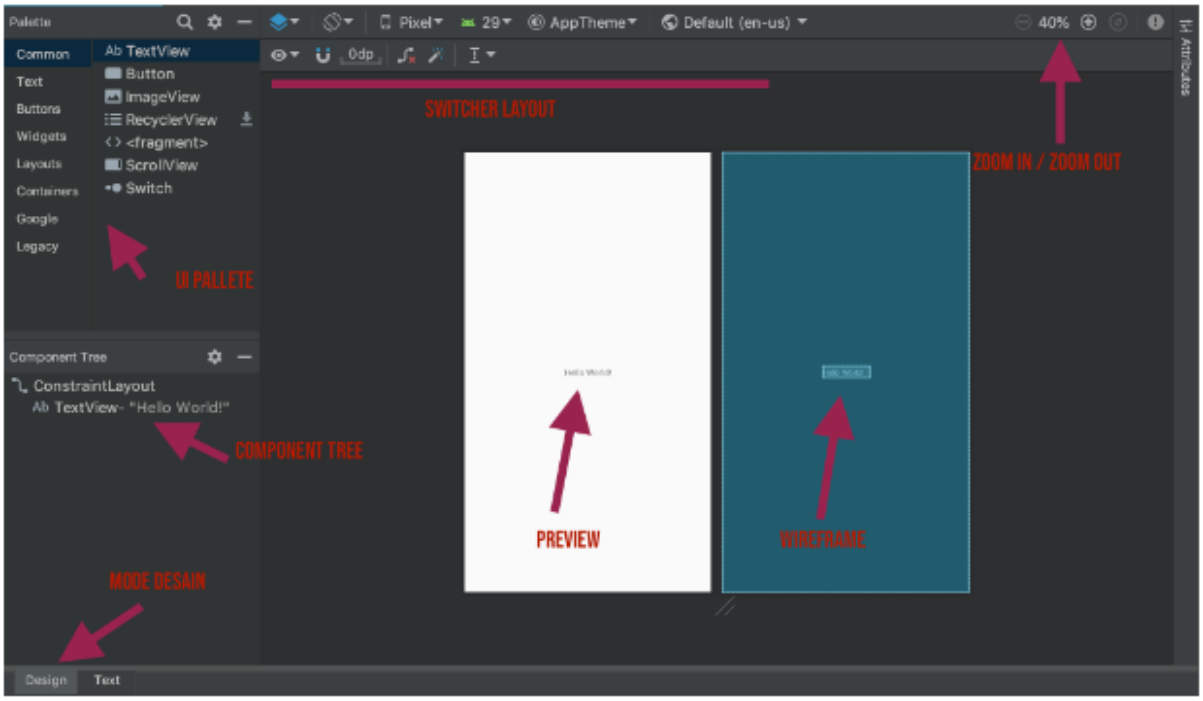
Layout file java

Berikut ini layout editor android ketika anda membuka sebuah file java.



Layout file xml

Berikut ini layout editor android ketika anda membuka sebuah file layout xml.



Shortcut

Berikut ini daftar shorcut yang sering digunakan pada saat membuat program android pada android studio.

Pencarian

- Shift + Shift - *Search Every Where*
- Ctrl + F - *Find Text with in a Single File*
- Ctrl + Shift + F - *Find Text in All Files*
- Ctrl + R - *Replace Selected Text in a Single File*
- Ctrl + Shift + R - *Replace Selected Text in all Files (Be Careful while Using This)*
- Ctrl + Shift + A - *Search for IDE Commands*

Navigasi

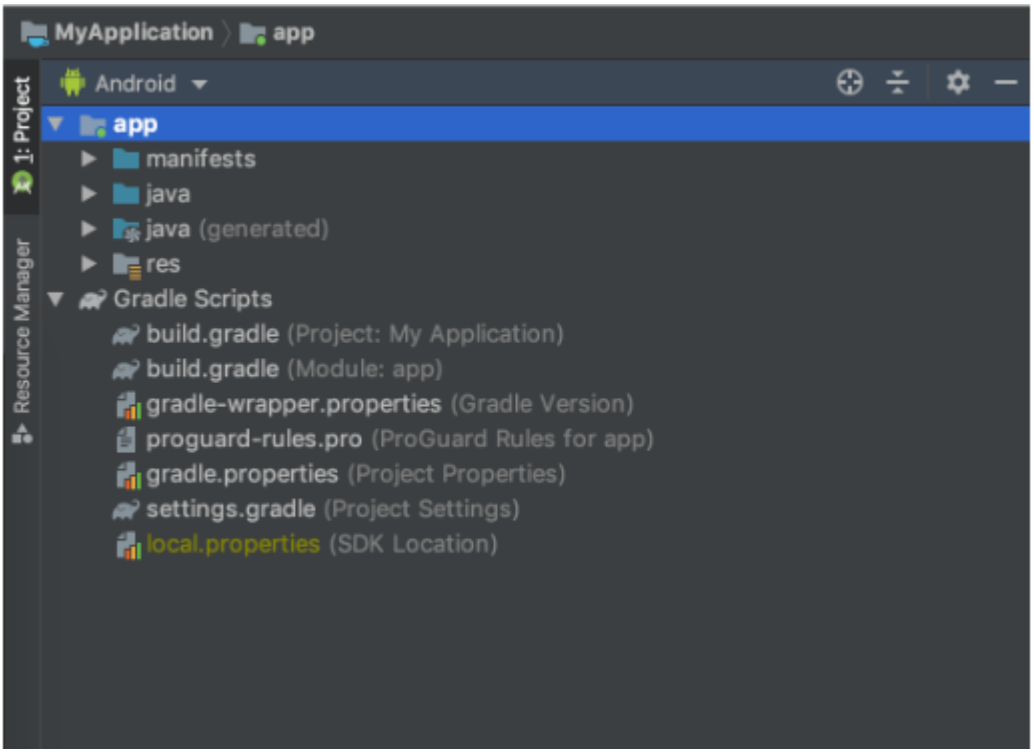
- Ctrl + N - Navigate to Class
- Ctrl + Shift + N - Navigate to a File
- Ctrl + B - Jump to Declareations
- Alt + ↑ - Jump to Previous Method
- Alt + ↓ - Jump to Next Method
- Ctrl + G - Jump to Line
- Ctrl + E - Recent Files
- Ctrl + Shift + Back Space - Jump to Last Edited Location
- Ctrl + B - Find Declarations
- Ctrl + Left Mouse (or) Ctrl + Alt + F7 - Show Usage
- Alt + F7 or Ctrl + F7 - Find usages /Find usages in file
- Ctrl + Shift + B - Find Implementations
- F3 - Find Next
- Shift + F3 - Find Previous

Editing

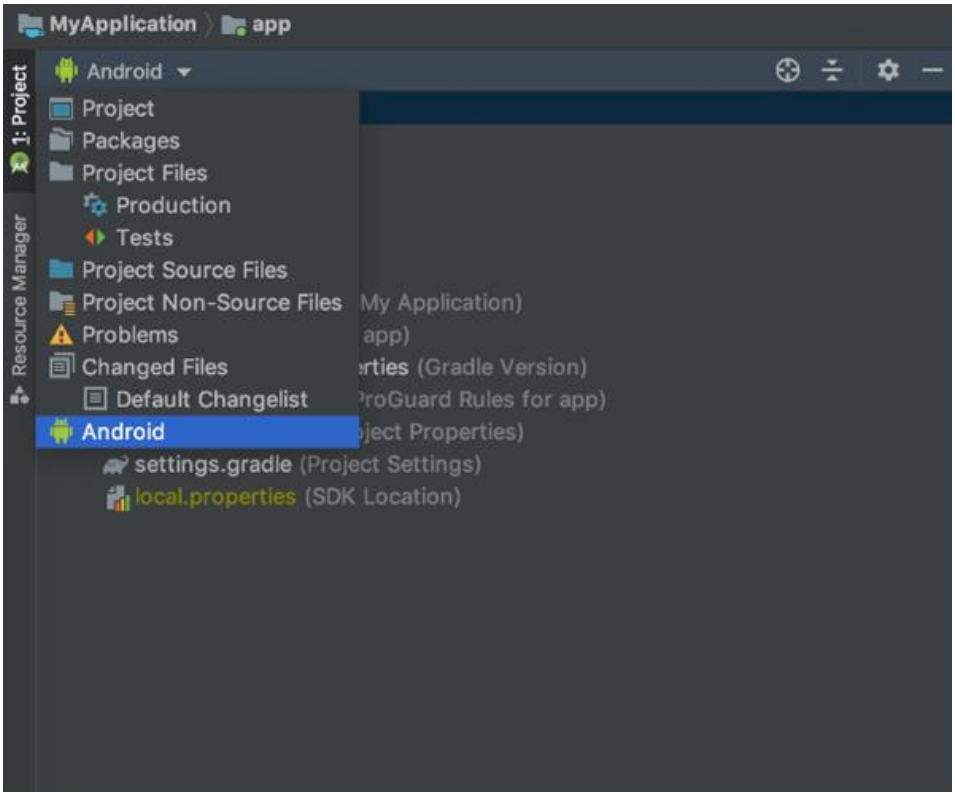
- Ctrl + F6 - Refractor Code
- Ctrl + D - Duplicate a Line/Selected part
- Ctrl + Y - Delete a Line/Selected part
- Ctrl + Q - Quick Documentation
- Ctrl + Space - Code completion
- Ctrl + Shift + Space - Smart code completion (by expected type removes unrelated suggestions)
- Alt + Insert - Generate Code
- Ctrl + J - Insert Live template
- Ctrl + O - Override methods
- Ctrl + I - Implement methods
- Ctrl + Alt + T - Surround with…
- Ctrl + / - Comment / uncomment with line comment
- Ctrl + Shift + / - Comment / uncomment with block comment
- Ctrl + Alt + L - Reformat code

Struktur Folder Project Android

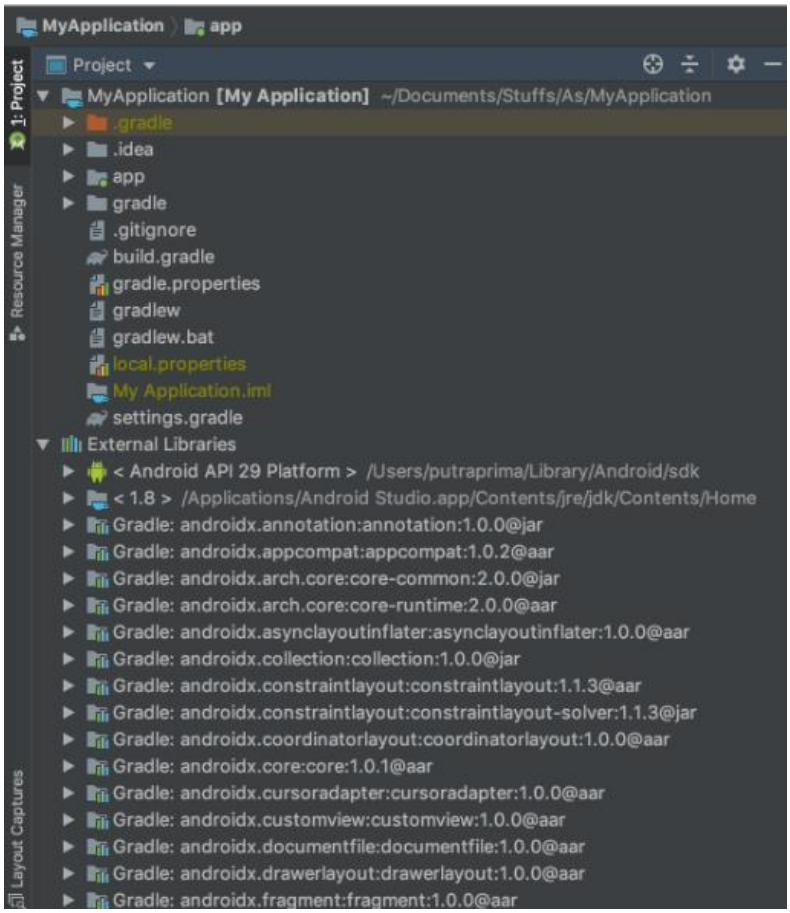
Pada sub bab ini anda akan mempelajari mengenai struktur folder pada sebuah project android. Berikut ini struktur folder default kode program android pada android studio.



Perhatikan sebuah dropdown di atas folder tree jika pada gambar sebelumnya digunakan "view" project android banyak view lain yang dapat digunakan daftar "view" dapat anda lihat dengan melakukan klik pada dropdown tersebut.



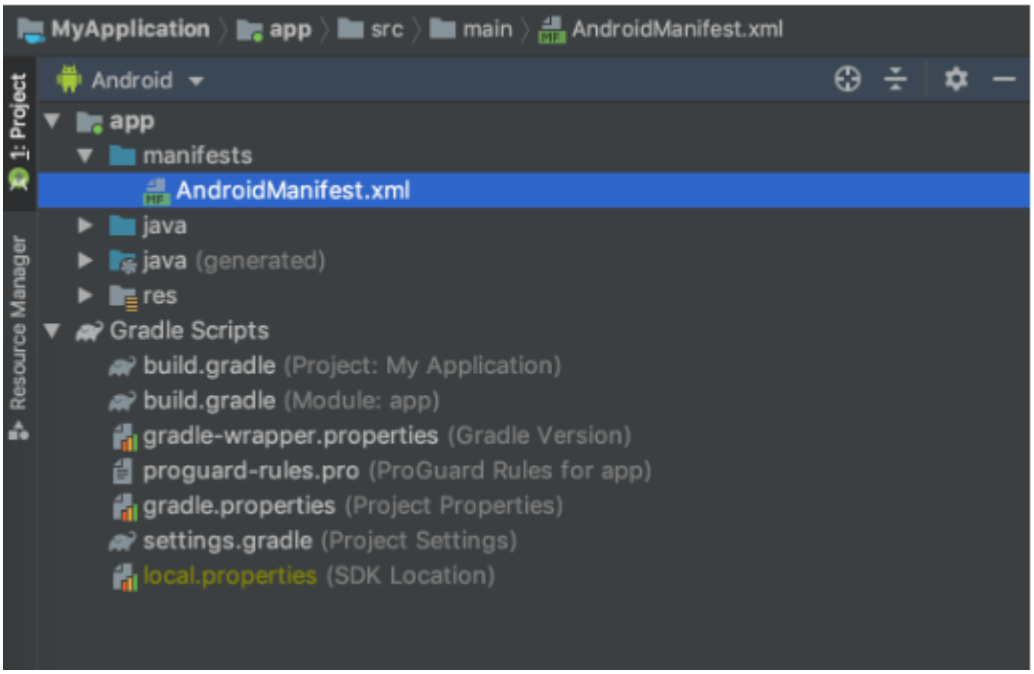
Berikut ini tampilan file tree jika view diganti ke "Project"



Karena view project yang paling umum digunakan adalah "android" kembalikanlah view kembali ke "android" sehingga file tree anda kembali pada gambar pertama.

Manifest

Pada folder manifest terdapat file yang berisi informasi mengenai permission, fitur dan activity/fragment yang digunakan.



Berikut ini isi dari file manifest pada sebuah template "Hello World"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloandroid">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HelloAndroid">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Pada file ini dapat anda lihat bahwa terdapat beberapa informasi penting mengenai aplikasi yang dibuat mulai dari :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloandroid">
```

Potongan kode manifest di atas menunjukkan bahwa package name yang digunakan adalah " com.example.helloandroid " implikasinya adalah semua kode program pada folder java harus menggunakan nama package ini.

Potongan kode selanjutnya adalah :

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.HelloAndroid">
```

Pada kode xml ini di setting nama aplikasi, icon launcher, support rtl, dan tema aplikasi

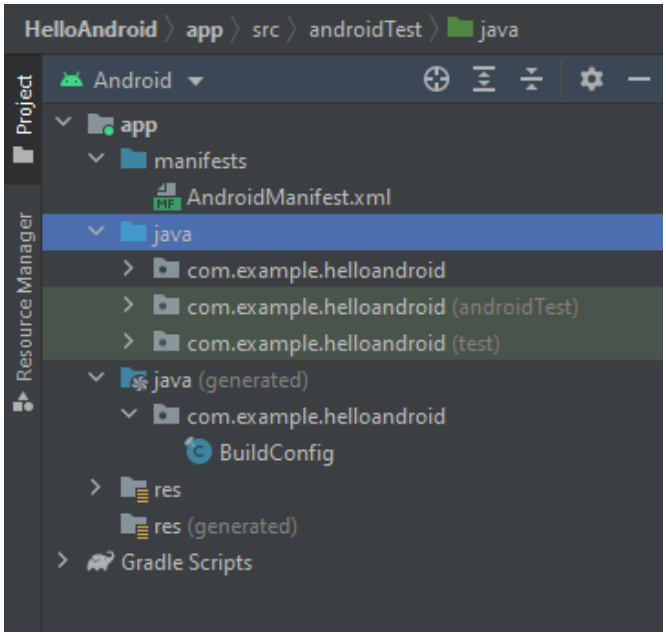
Selanjutnya pada tag `activity` berisi informasi mengenai activity yang dimiliki oleh aplikasi

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

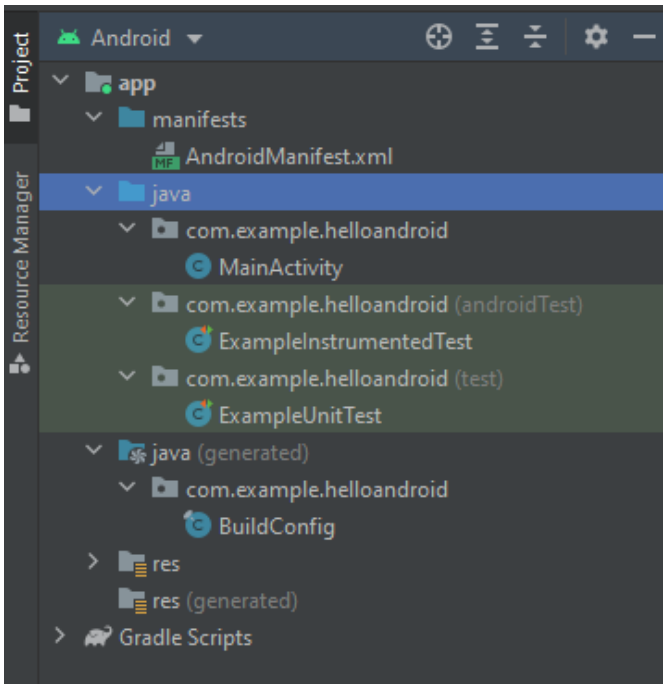
Perhatikan pada tag `intent-filter` terdapat tag child `category` dengan properties `android.intent.category.LAUNCHER` ini menandakan bahwa activity pada tag ini merupakan activity yang **pertama** kali dibuka

Java

Isi dari folder java dapat dilihat pada gambar dibawah ini :

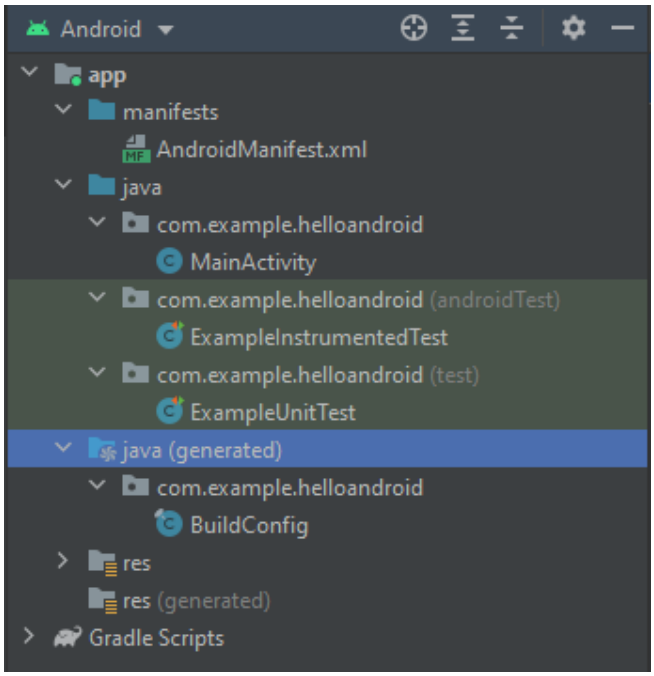


Perhatikan terdapat tiga folder dengan nama yang sama sesuai dengan package name pada file manifest, dimana pada manifest package name nya adalah **com.example.helloandroid** maka pada folder java terdapat folder dengan nama yang sesuai dengan potongan pertama pada package name yaitu **com.example.helloandroid**. Masing masing folder walaupun mempunyai nama yang sama memiliki fungsi yang berbeda, folder pertama berisi kode program, folder kedua dan ketiga berisi file testing untuk aplikasi.



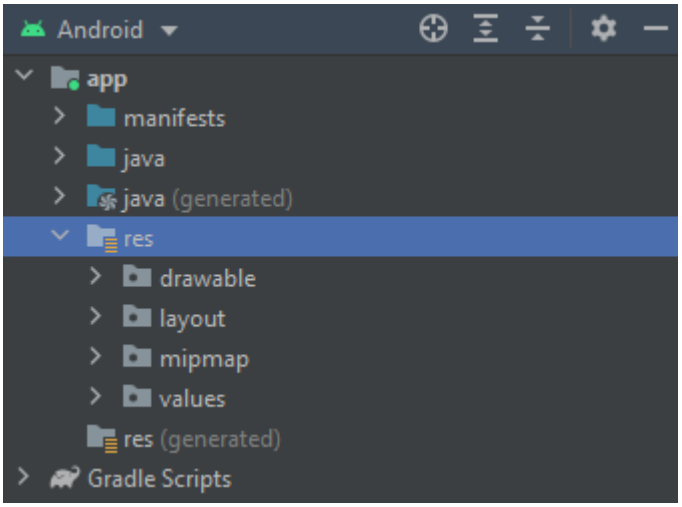
Java(Generated)

Pada Folder ini berisi kode program yang di generate oleh android studio, dalam kondisi apapun anda sebaiknya tidak merubah kode program pada folder ini.



Res

Folder selanjutnya adalah folder res, dimana pada folder ini berisi semua resource lokal yang dibutuhkan untuk pembuatan aplikasi selain kode program. Isi default folder ini adalah **drawable, layout, mimpap, dan values**



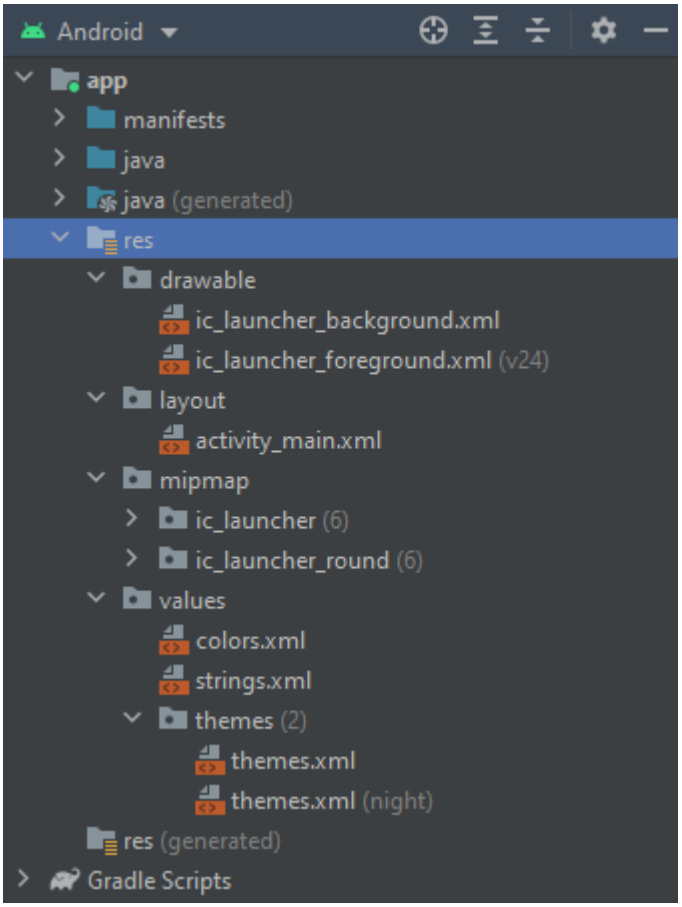
Isi pada folder drawable dapat berupa sebuah gambar atau xml drawable dengan syarat penamaan pada folder ini tidak boleh menggunakan angka, huruf kapital dan menggunakan underscore sebagai penghubung antar kata (jika ada spasi).

Isi pada folder layout adalah layout xml yang digunakan untuk membuat tampilan activity atau fragment pada android studio. Layout ini menggunakan tag xml dimana pembuatannya memiliki aturan aturan sendiri seperti pada pembuatan html untuk aplikasi web. Pembahasan mengenai layout akan diperdalam pada pertemuan pertemuan selanjutnya.

Isi pada folder mipmap adalah logo aplikasi perhatikan ada beberapa resolusi yang harus disediakan oleh sebuah gambar untuk dapat menjadi logo aplikasi yang baik hdpi, mdpi, xhdpi, xxhdipi,xxxhdpi.

Isi pada folder values berupa file xml yang mengatur warna, string ,dan style pada aplikasi.

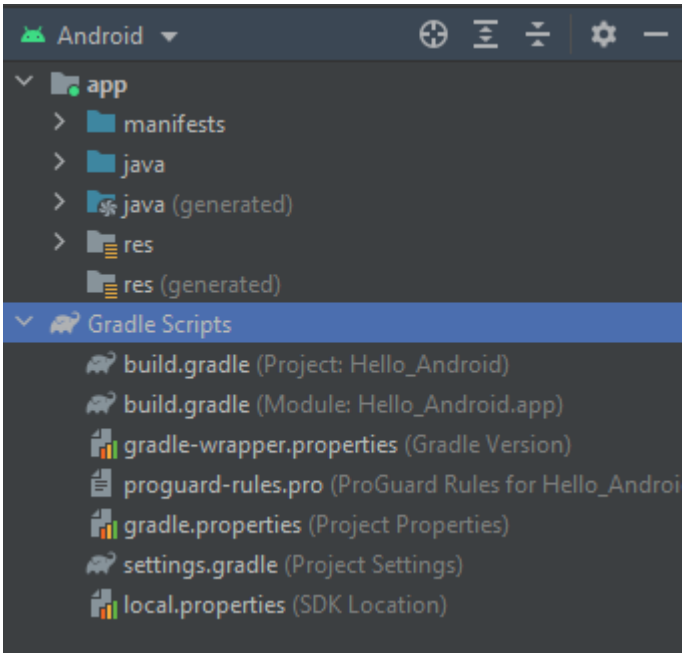
Isi folder res akan dibahas lebih dalam pada pertemuan mengenai desain layout dan fragment.



Gradle scripts

Gradle merupakan build system yang digunakan oleh android studio, pada file ini berisi konfigurasi konfigurasi penting mengenai proses build aplikasi. File gradle pada sebuah project android studio terdiri dari dua level yaitu level `project` dan level `module` paling tidak

sebuah project android memiliki 2 file gradle, jika project tersebut memiliki modul lebih dari satu maka file gradle nya akan bertambah sesuai dengan jumlah module yang dimiliki.



Berikut ini isi dari file gradle pada module app

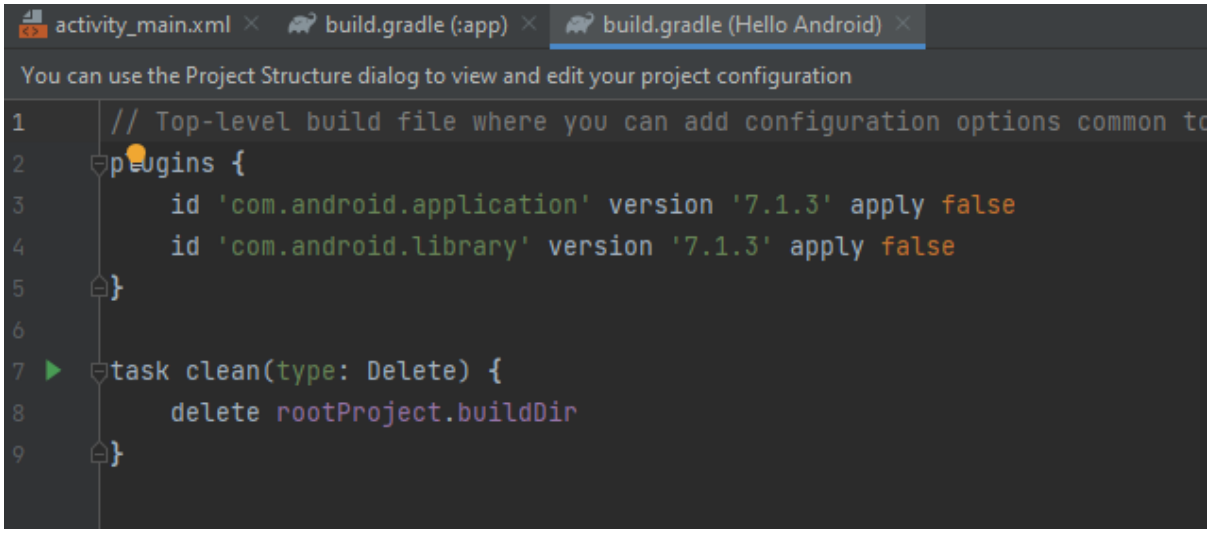
activity_main.xml | build.gradle (:app)

You can use the Project Structure dialog to view and edit your project configuration

```
plugins {
    id 'com.android.application'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.example.helloandroid"
        minSdk 28
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.
        }
```

Pada file gradle ini berisi semua library dan konfigurasi yang digunakan untuk module app, anda akan sering mengubah dan mempelajari isi file ini seiring dengan berjalannya perkuliahan.

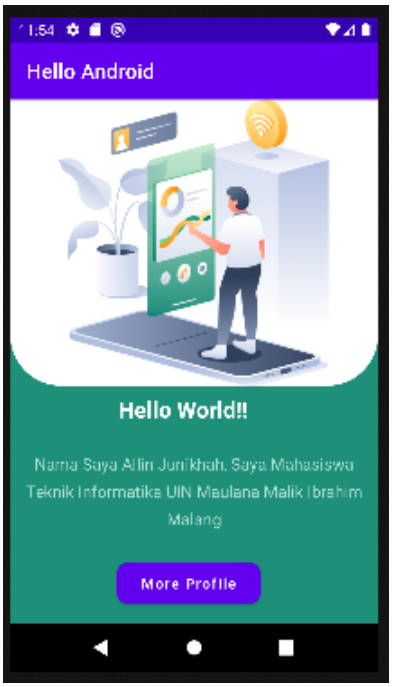
Sedangkan gradle pada level project berisi repository mana yang digunakan untuk memeriksa dependency aplikasi yang di isi pada file gradle module.



Tugas

1. Ubahlah teksview yang ada pada `activiy_main.xml` menjadi nama lengkap anda!
2. Pelajari dan buatlah layout pada video tutorial youtube berikut ini https://www.youtube.com/watch?v=6dtFTQExhaI , Tampilkan layout versi anda!

   Example :



Modul 2

Pemrograman Native dengan Java : Activity dan Intent pada Android

1. MATERI

Membuat aplikasi sederhana pindah Layout (Activity) pada Android Studio menggunakan Intent

gambar ini diambil dari https://www.tutorialspoint.com/android/android_intents_filters.htm

Intent

Intent adalah mekanisme untuk melakukan Action pada komponen dalam satu aplikasi yang sama maupun aplikasi lain. Pemanfaatannya sebagai berikut :

- Untuk berpindah halaman dari satu Activity ke Activity lain dengan atau tanpa data
- Untuk menjalankan background Service misal dibutuhkan sebuah proses service untuk mengambil lokasi pengguna, download file atau sikronisasi ke server
- Untuk menyampaikan sebuah objek dari komponen Broadcast misal jika ingin mengetahui jika device Android sudah selesai booting setelah diaktifkan

Explicit dan Implicit Intent

➢ **Explicit Intent**

Untuk mengaktifkan komponen-komponen dalam satu aplikasi yang sama misal : Berpindah Activity atau mengaktifkan service untuk mendownload file secara background

➢ **Implicit Intent**

Untuk mengaktikan komponen dari aplikasi lain. Misal : mengaktifkan dial phone pada aplikasi Telp, mengaktifkan driving direction pada Google Maps atau mengirimkan pesan via Gmail, sms atau aplikasi lainnya.

➢ Ilustrasi

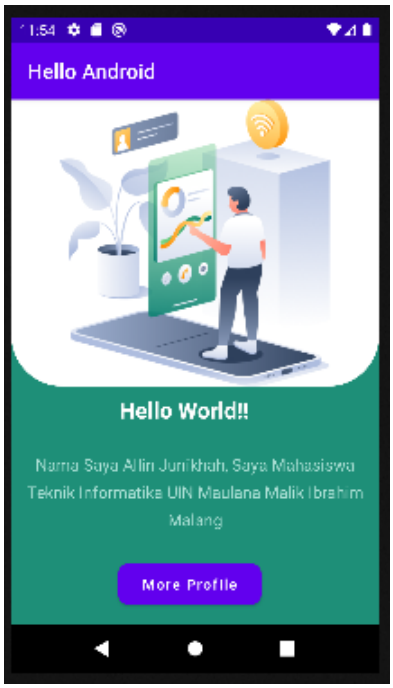
Ilustrasi bagaimana Intent mengaktifkan Activity pada aplikasi lain.

1. Activity A membuat Intent untuk melakukan sebuah Action dengan data object yang ada.
2. Sistem Android akan mencari aplikasi yang cocok Berdasarkan Intent filter
3. Activity pada aplikasi yang cocok akan ditampilkan.

2. PRAKTIKUM

a. Membuat Activity Baru

Pada praktikum sebelumnya kita telah membuat user interface sederhana dengan Button. Pada praktikum kali ini, kita akan melanjutkan dengan menambahkan aksi pada tombol **"More Profile"** yang telah kita buat agar ketika ia diketuk, akan menampilkan activity baru.



Daftar file xml layout yang digunakan dalam project ini:

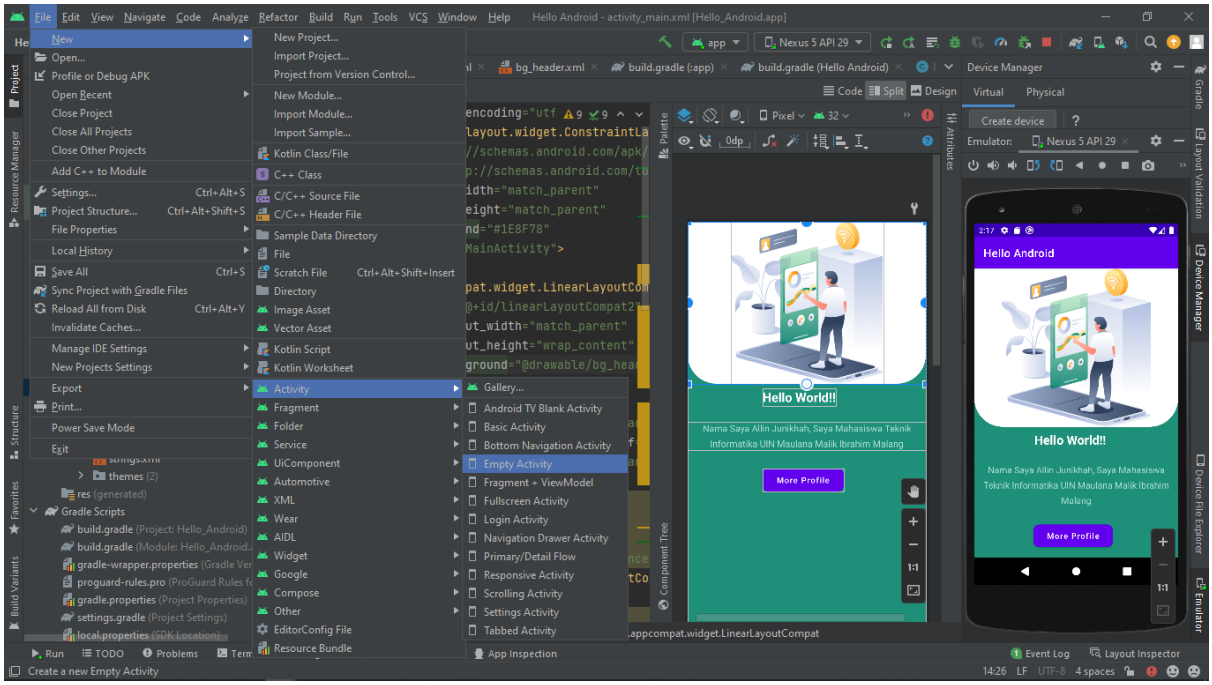
- activity_main.xml
- activity_profile.xml

Daftar file java activity yang digunakan dalam project ini:

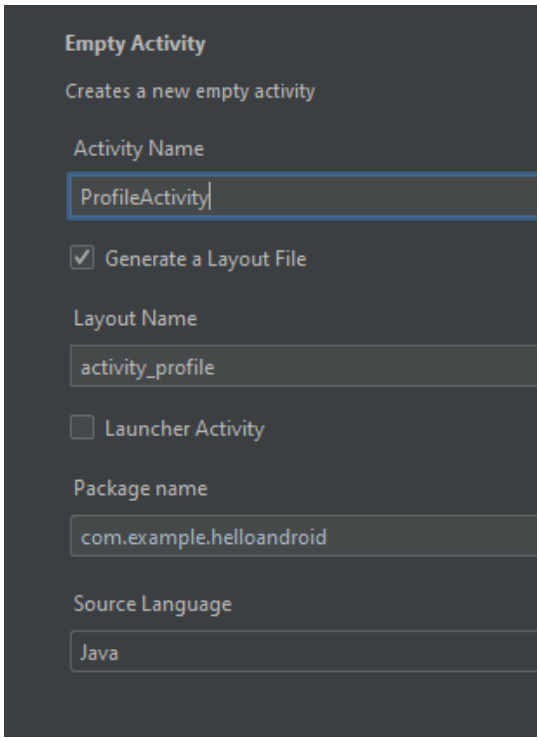
- MainActivity.java

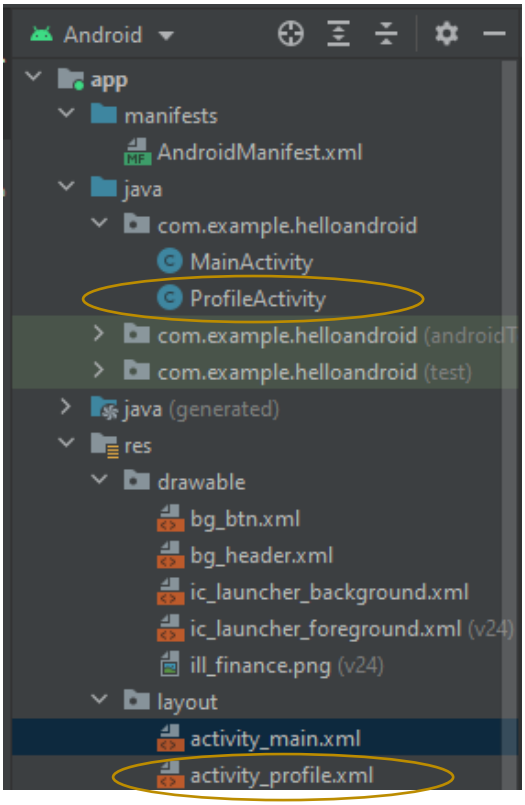
Berikut ini adalah langkah-langkah dalam pembuatan **Activity** Baru.

1. **File → New → Activity → Empty Activity**



2. Kemudian isi seperti berikut ini



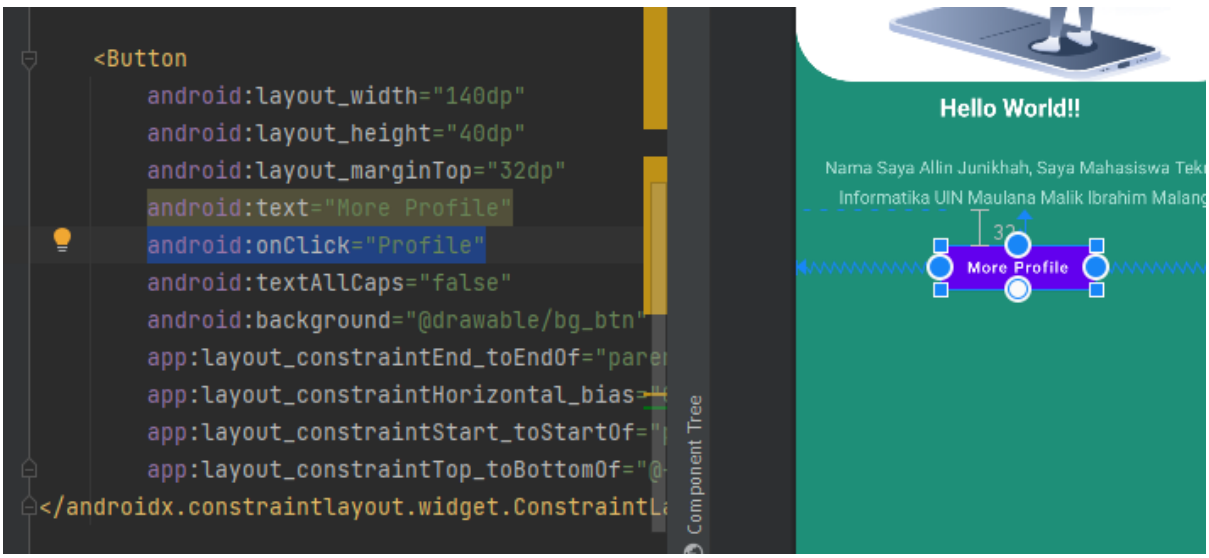


Setiap kita membuat activity baru maka secara bersamaan kita akan membuat file layout **xml baru.**

3. Setelah terdapat 2 Activity maka sekarang kita mengatur attribute Button pada **activity_main.xml** supaya ketika di klik maka secara otomatis akan berpindah ke activity_profile.xml (Activity kedua). Tambahkan terlebih dahulu script seperti berikut

Android:onClick=" Profile"

Pada file **activity_main.xml** tepatnya pada tombol button.



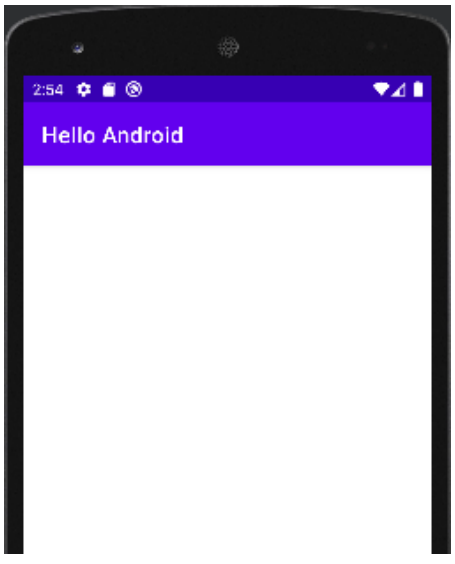
4. Tambahkan script berikut pada file **MainActivity.java**

```
public void Profile(View view)
{
Intent intent = new Intent(MainActivity.this, ProfileActivity.class);
startActivity(intent);
}
```

5. Tambahkan **import android.view.View;**
   Hingga menjadi kumpulan kode seperti di bawah ini.

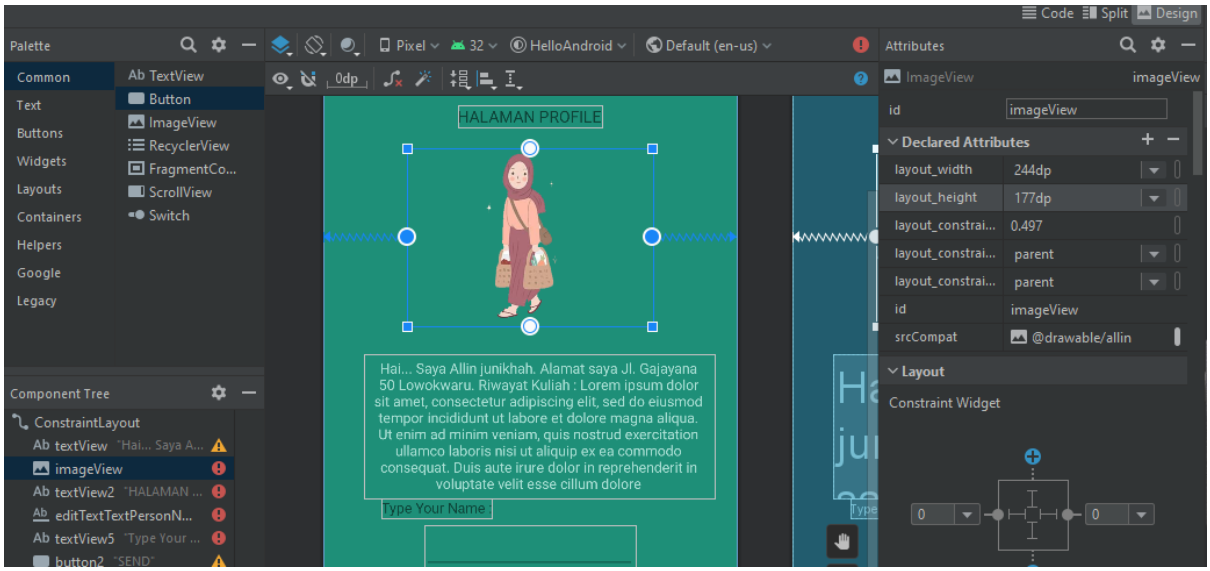
```
package com.example.helloandroid;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void Profile(View view)
    {
        Intent intent = new Intent( packageContext: MainActivity.this, ProfileActivity.class);
        startActivity(intent);
    }
}
```

6. **Run** aplikasi, tombol **More Profile** sudah dapat berpindah pada halaman activity baru **(activity_profile.xml).**
   Jika dianalogikan, kesimpulannya adalah Intent merupakan suatu alat yang berfungsi untuk menjembatani antara *FirstActivity* dengan *SecondActivity* sehinnga pada kondisi tertentu fungsi *Intent* akan dipanggil untuk menampilkan *SecondActivity.* Seperti pada contoh diatas saya menggunakan kondisi ketika diklik*(onClick)*.

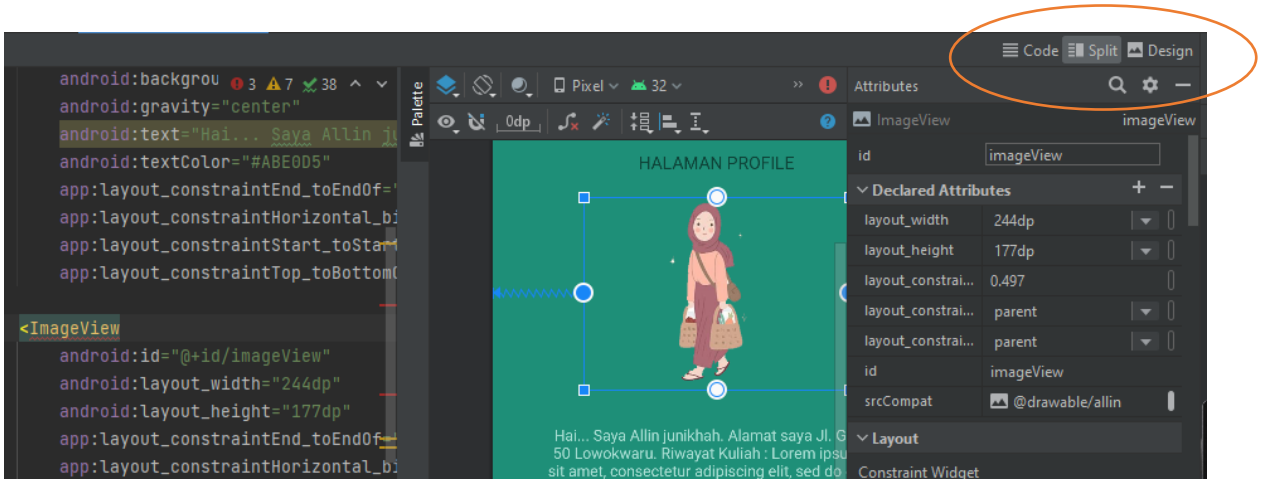


7. **Halaman** activity_profile.xml yang masih kosong menampilkan halaman kosong. Tambahkan beberapa tampilan pada activity tersebut. Seperti contoh berikut.

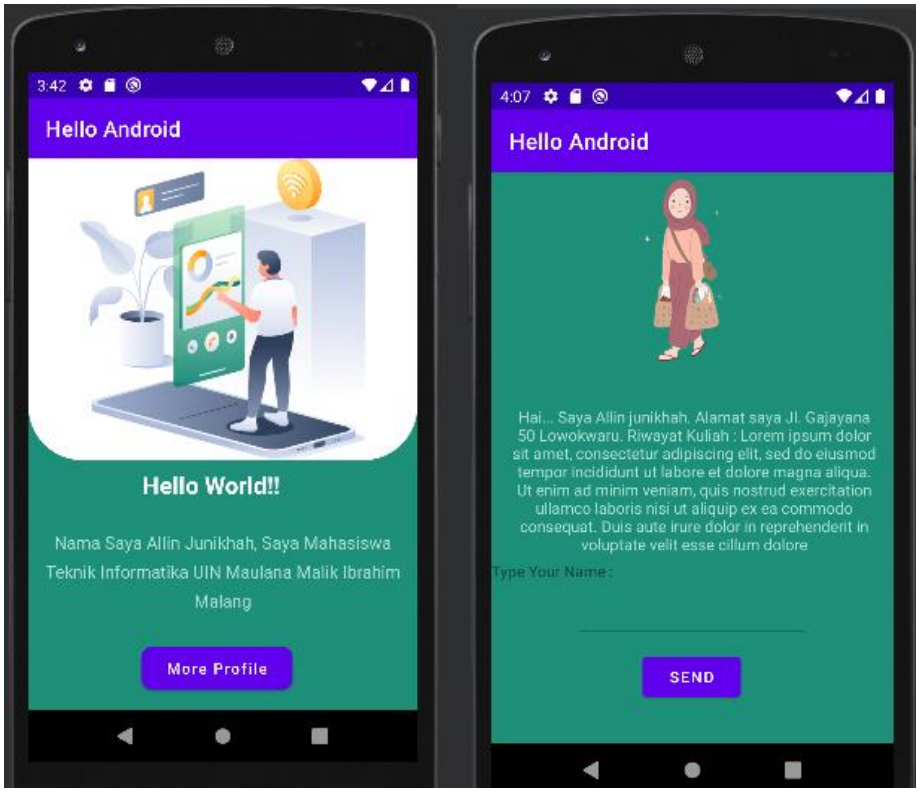
   Kita dapat menggunakan **drag and drop** dari palette yang ada maupun dapat menulis script source secara manual.



   Kita dapat berganti dari mode **design** maupun **code** sesuai keperluan kita.



8. Run sekali lagi aplikasi dengan activity kedua (activity Profile) yang telah dibuat.



activity_main activity_profile

3. TUGAS

1. Buat Activity Profile versi kalian. Tampilkan nama, nim, avatar/foto !
2. Buat 3 activity berbeda!

Modul 3

Pemrograman Hybrid dengan Dart dan Flutter Framework : Installasi Software

1. MATERI

a. Apa itu Flutter

Flutter adalah sebuah framework dari bahasa Dart yang diciptakan oleh Google. Flutter merupakan teknologi yang digunakan untuk membuat aplikasi antar platform (Android, iOS, Web, Desktop) dengan menggunakan satu bahasa pemrograman yaitu Dart.

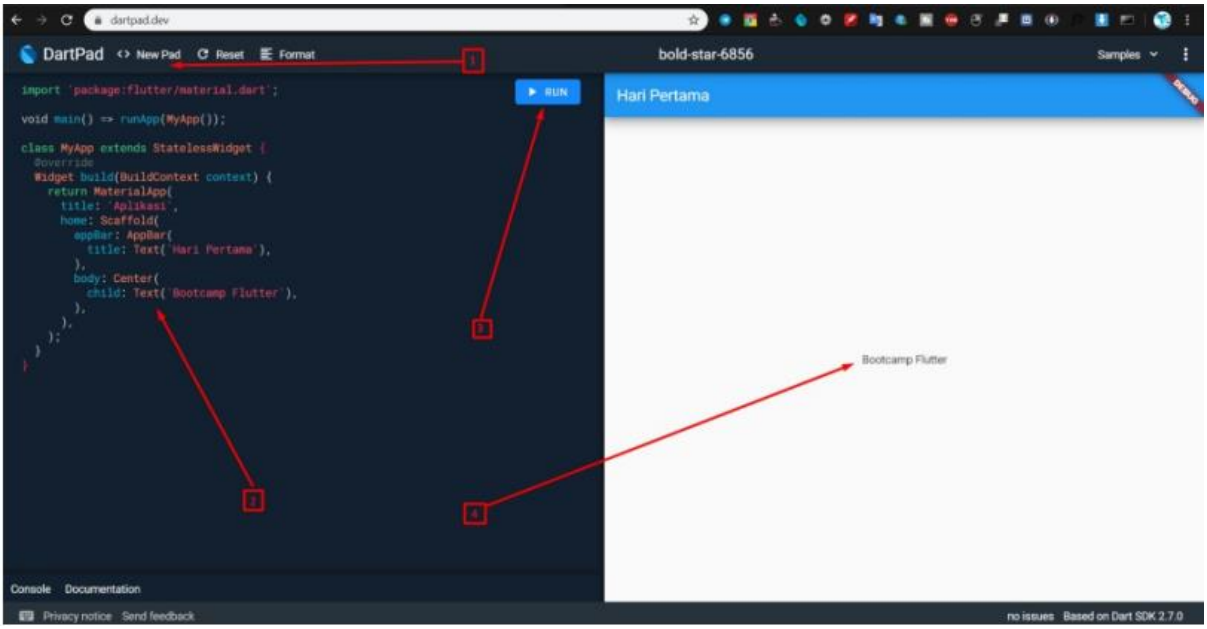
Framework Flutter dan Bahasa Dart

Dart merupakan bahasa pemrograman yang dikembangkan oleh Google untuk kebutuhan umum (*general-purpose programming language*). Dart diperuntukan untuk berbagai macam platform diantaranya web, aplikasi mobile, server dan perangkai IoT (*Internet of Things*).

Dart merupakan bahasa berorientasi objek dengan menggunakan syntax bahasa C. Dart diperkenalkan Google sebagai bahasa pengganti Javascript. Versi pertama Flutter dikenal sebagai “Sky” dan berjalan pada sistem operasi Android. Diresmikan pada perhelatan Dart Developer Summit 2015, dengan tujuan untuk mampu merender grafis secara konsisten pada 120 bingkai per detik (fps).

b. Mencoba Flutter

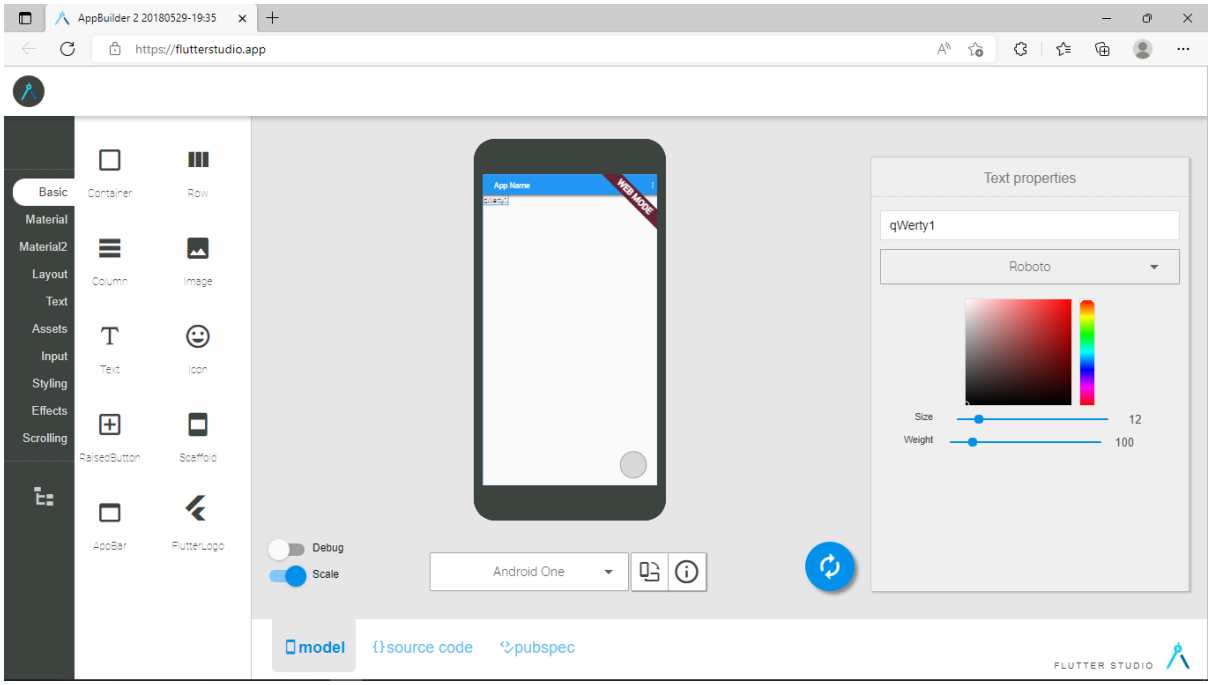
Sebelum menginstall Flutter, kita bisa mencoba ngoding Flutter secara online melalui web: https://dartpad.dev/

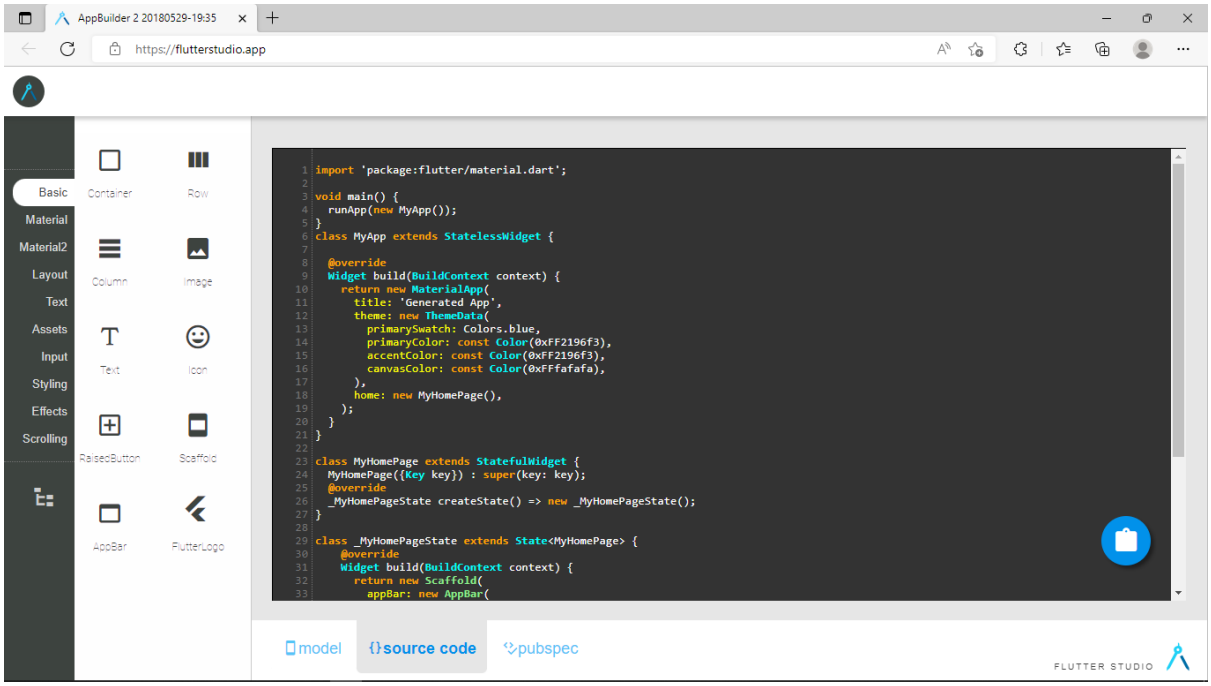


Klik New Pad, tuliskan code, dan run

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Aplikasi',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Hari Pertama'),
        ),
        body: Center(
          child: Text('Bootcamp Flutter'),
        ),
      ),
    );
  }
}
```

Ada satu lagi aplikasi online untuk membuat aplikasi Flutter dengan cara *drag and drop* yaitu https://Flutterstudio.app/


AppBuilder 2 20180529-19:35
https://flutterstudio.app
Basic
Material
Material2
Layout
Text
Assets
Input
Styling
Effects
Scrolling
Container
Row
Column
Image
Text
Icon
RaisedButton
Scaffold
AppBar
FlutterLogo
App Name
WEB MODE
Text properties
qWerty1
Roboto
Size
12
Weight
100
Debug
Scale
Android One
model
source code
pubspec
FLUTTER STUDIO


AppBuilder 2 20180529-19:35
https://flutterstudio.app
Basic
Material
Material2
Layout
Text
Assets
Input
Styling
Effects
Scrolling
Container
Row
Column
Image
Text
Icon
RaisedButton
Scaffold
AppBar
FlutterLogo
import 'package:flutter/material.dart';
void main() {
runApp(new MyApp());
}
class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
return new MaterialApp(
title: 'Generated App',
theme: new ThemeData(
primarySwatch: Colors.blue,
primaryColor: const Color(0xFF2196f3),
accentColor: const Color(0xFF2196f3),
canvasColor: const Color(0xFFfafafa),
),
home: new MyHomePage(),
);
}
}
class MyHomePage extends StatefulWidget {
MyHomePage({Key key}) : super(key: key);
@override
_MyHomePageState createState() => new _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
@override
Widget build(BuildContext context) {
return new Scaffold(
appBar: new AppBar(
model
source code
pubspec
FLUTTER STUDIO

2. PRAKTIKUM

a. Install Flutter

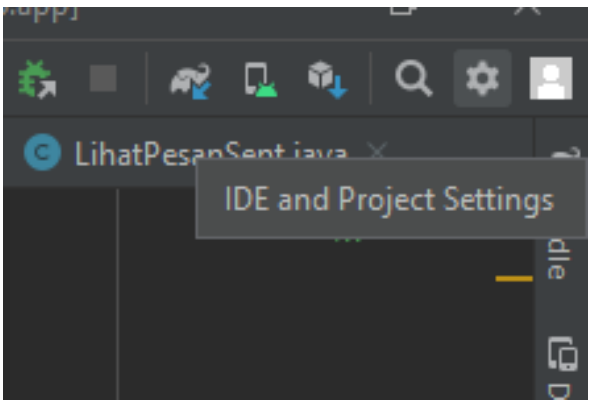
Karena kemudahan Flutter untuk membuat aplikasi Android dan iOS dalam satu single codebase membuat para developer merasa terbantu untuk hal deployment, dan kita bisa memakai Flutter tidak hanya pada satu sistem operasi saja, melainkan Flutter SDK dapat dijalankan di *multi platform* seperti **Windows, Linux dan MacOS**.

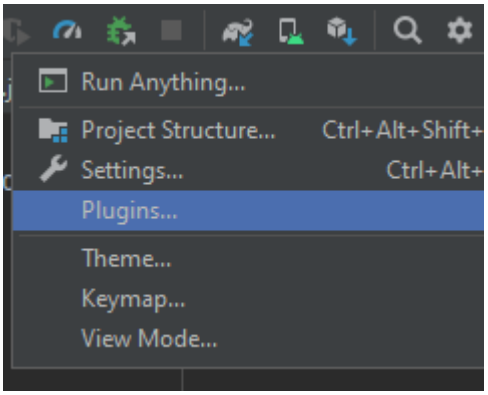
Komponen yang akan kita pakai adalah sebagai berikut:

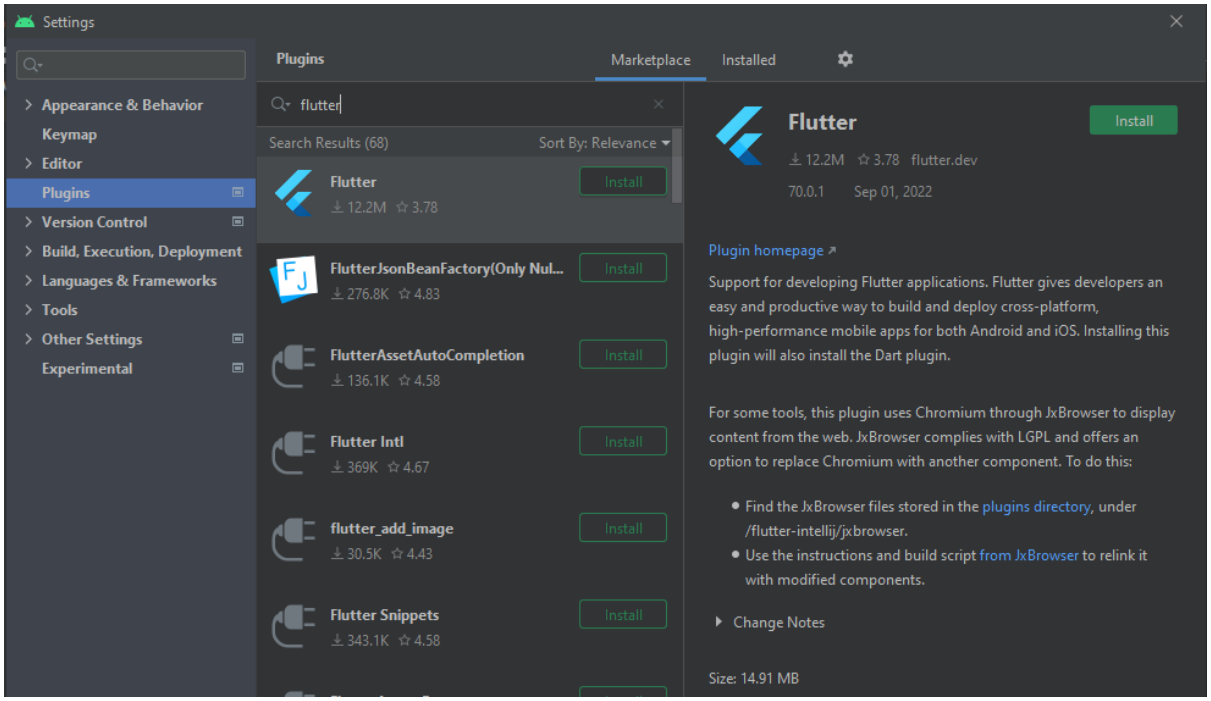
1. Android Studio
2. Flutter SDK
3. Visual Studio Code

Step 1: Installing Android Studio (Done)

- Jika sudah selesai pilih menu **Configure -> Plugins->Browse Repositories**, kemudian cari dengan kata kunci **Flutter**. Maka pencarian Flutter akan muncul pada urutan nomor 1. Tekan tombol install untuk memasang Flutter di Android Studio.
- Atau jika pernah mengembangkan project dengan Android Studio sebelumnya, cari **IDE and Project Setting → Plugins,** kemudian cari dengan kata kunci **Flutter**. Maka pencarian Flutter akan muncul pada urutan nomor 1. Tekan tombol install untuk memasang Flutter di Android Studio.

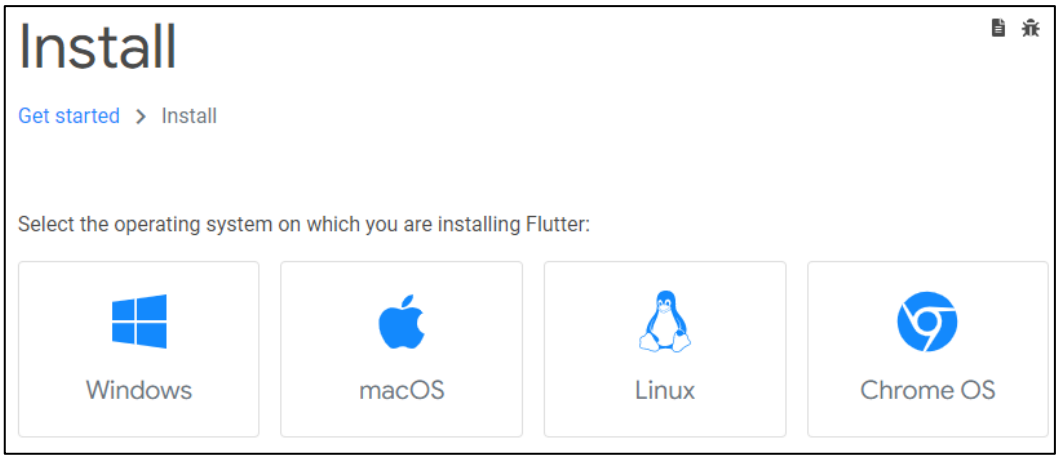




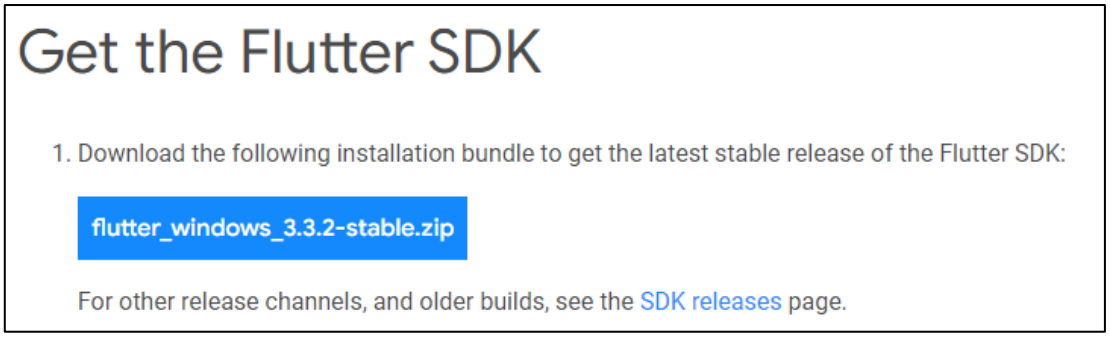


Step 2: Installing Flutter

- Pertama kita harus mengunduh Flutter SDK dari situs resminya pada link berikut https://Flutter.dev/docs/get-started/install
- Kita akan disuguhkan 3 tipe pilihan sistem operasi untuk mengunduh SDK, pilih sesuai dengan sistem operasi yang dipakai

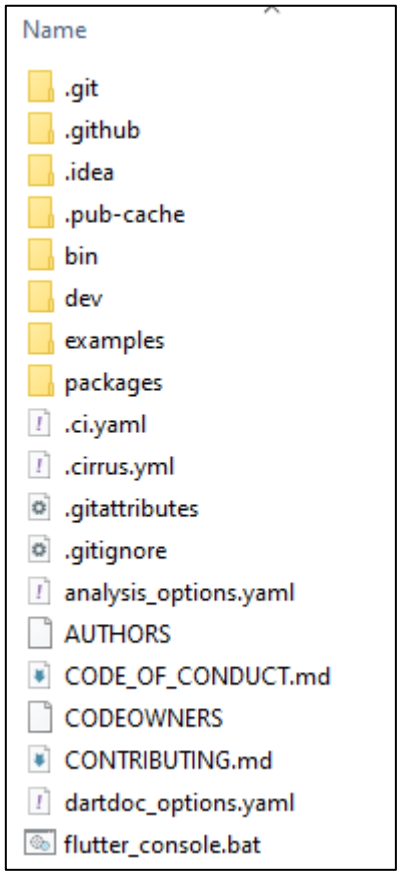


Unduh file Flutter SDK versi stable dengan menekan tombol biru pada step pertama di situs resminya



Silahkan extract file tersebut di lokasi folder yang diinginkan. Perlu diperhatikan karena lokasi tersebut akan kita masukkan ke dalam Environment Variable, jadi perludiingat kembali dimana kita menaruhnya.

Kira-kira isi foldernya adalah sebagai berikut:

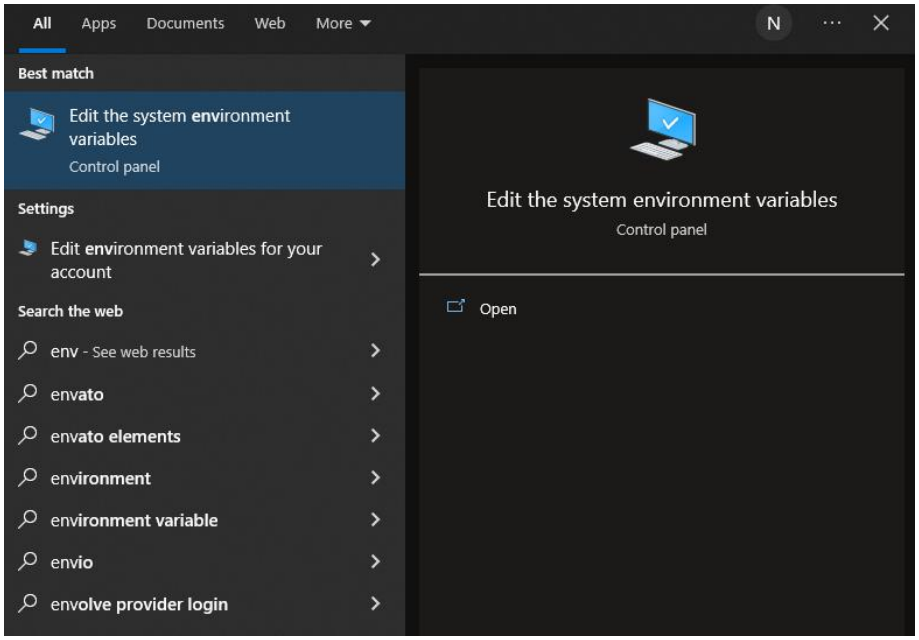


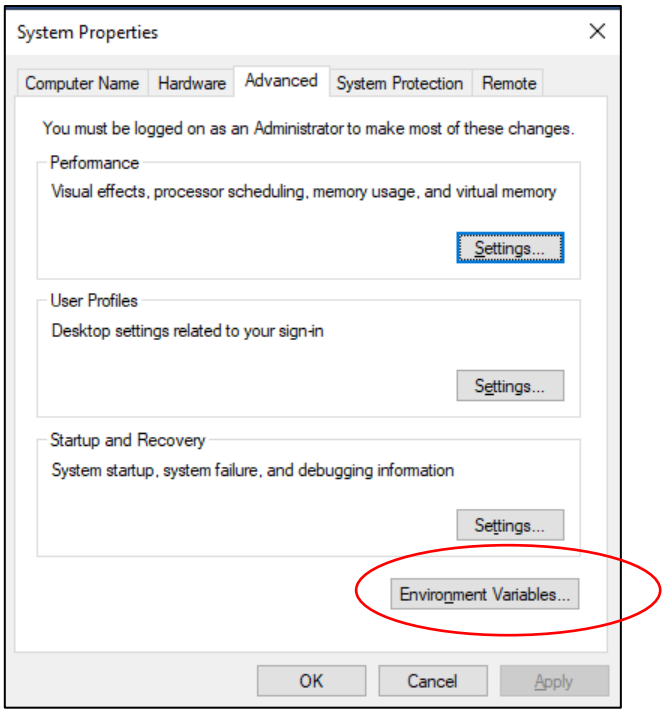
Step 3: Setting Environment Variable

Sebelum Flutter dapat digunakan melalui terminal kita perlu melakukan setting environment variable terlebih dahulu, karena proses ini berfungsi sebagai *shortcut path* agar bisa menggunakan perintah Flutter pada terminal. Untuk cara penyetingannya mungkin berbeda pada tiap sistem operasi.

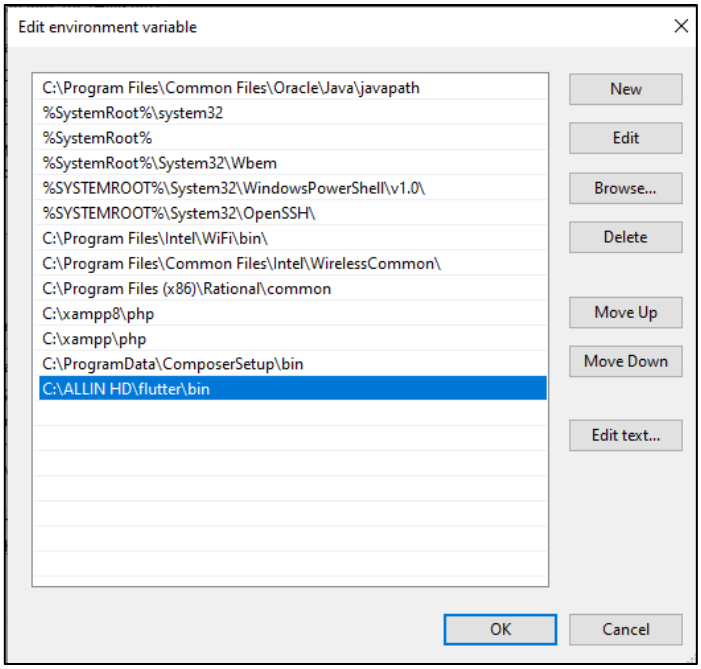
Windows

- Pada Start Search di windows cari dengan kata kunci “env” maka akan muncul pilihan Setting Environment Variable -> Environment Variable.

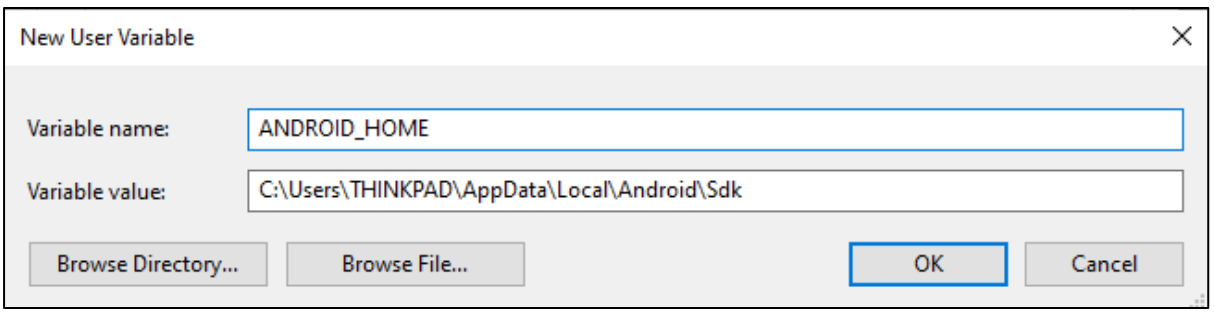




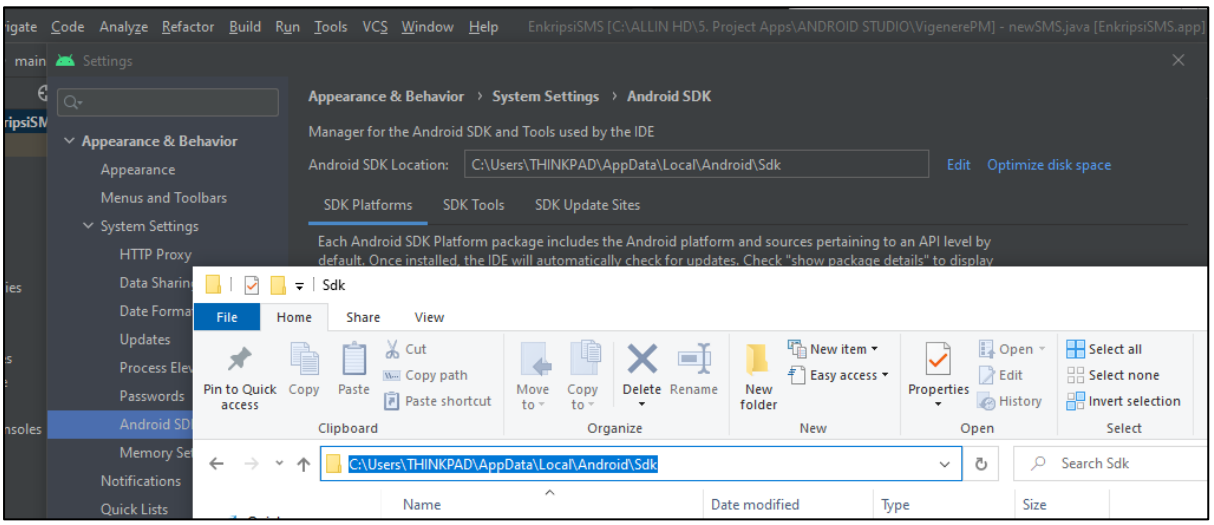
- Dibagian Users Variable dan System Variables klik 2x item PATH, maka akan muncul form edit environment variable seperti berikut.
- Silahkan klik New dan masukkan PATH lokasi dimana kita menginstall Flutter.



- Selanjutnya pada User Variable tekan tombol New dan tambahkan PATH untuk ANDROID_HOME sebagai berikut:

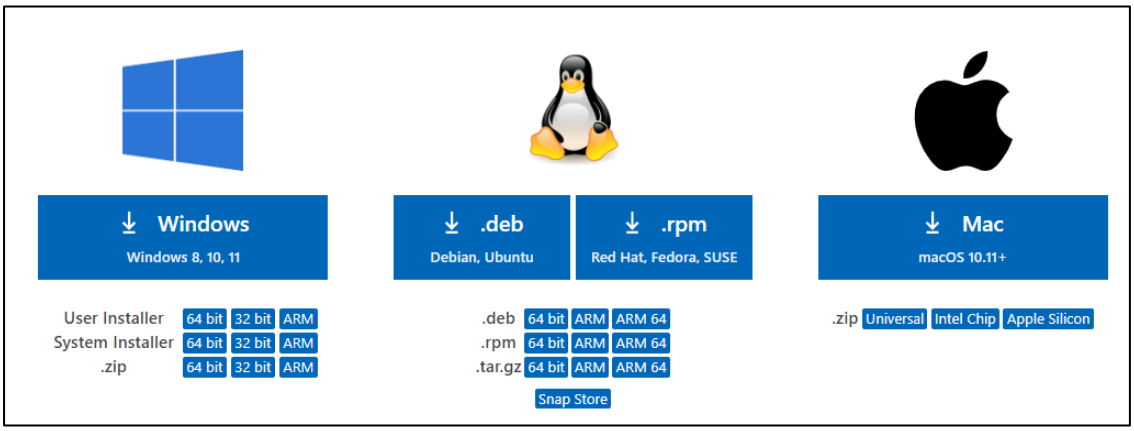


Jika kita lupa meletakkan Android SDK kita, kita dapat melihat pada SDK Manager pada Android Studio.

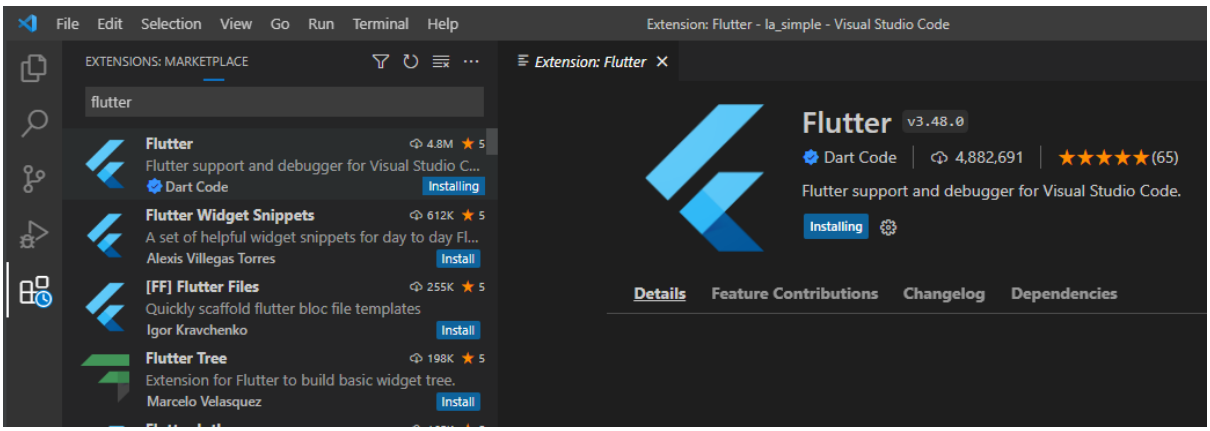


Step 3: Install Visual Studio Code

- Kita dapat mengunduh Visual Studio Code pada link berikut https://code.visualstudio.com/download (Pilih sesuai sistem operasi yang dipakai)



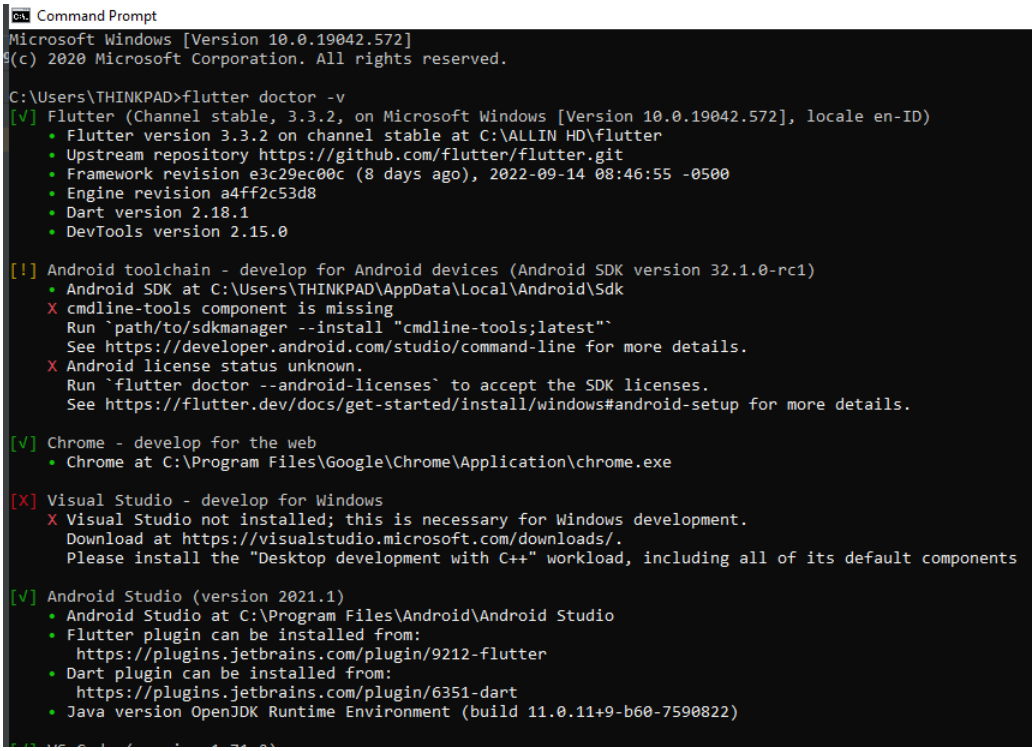
- Jika proses instalasi sudah berhasil sekarang waktunya menginstall **extension Flutter** di Visual Studio Code. Klik pada menu extension di sebelah kiri, kemudian cari dengan kata kunci **"Flutter"**, maka kita akan mendapatkannya di urutan pertama. Tekan install dan restart jika sudah selesai.



Step 4: Flutter Doctor

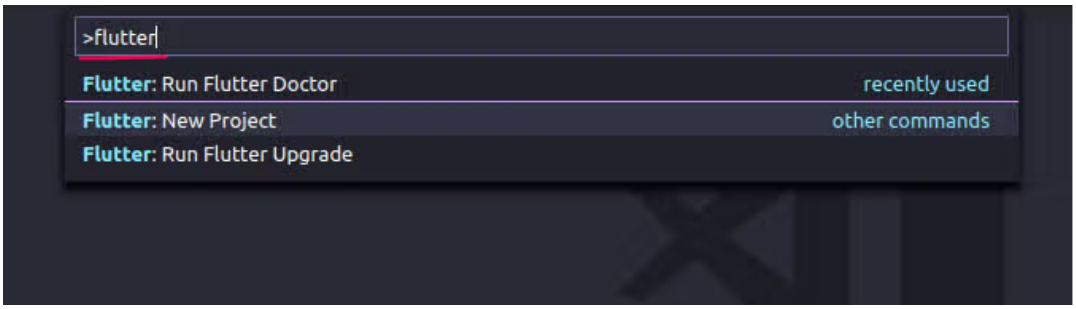
- Setelah semua proses di atas berhasil dilakukan sekarang kita uji apakah proses instalasi kita telah berjalan dengan lancar. Flutter menyediakan sebuah tools bernama Flutter Doctor yang berfungsi sebagai utilitas yang memeriksa environment Flutter kita terhadap masalah-masalah yang mungkinterjadi.
- Buka terminal kemudian masukkan perintah **Flutter doctor**.

Maka proses diagnose akan berjalan. Jika proses ini tidak terjadi masalah berarti kita sudah bisa mulai develop **Flutter** menggunakan **Visual Studio Code**. Note: Jika kalian menemukan error mengenai android-license yang belum terpasang, silahkan masukkan perintah berikut untuk memperbaikinya **Flutter doctor –android-licenses.** Ikuti yang disarankan dari flutter doctor, seperti pada beberapa error dibawah ini. Lakukan setting cmdline-tool CLI pada Android Studio, mungkin settingan belum tercentang. Dan lakukan installasi Dekstop development C++ pada Visual Studio installer. Gunakan google untuk troubleshooting jika dirasa error belum terselesaikan.



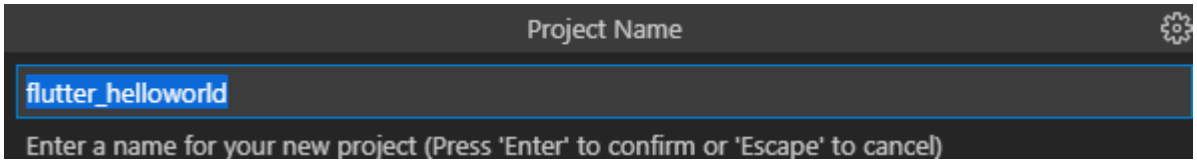
Step 5: Membuat Aplikasi Pertama

Cara membuat project Flutter di VS Code: Tekan tombol Ctrl+Shift+P, lalu pilih New Project.

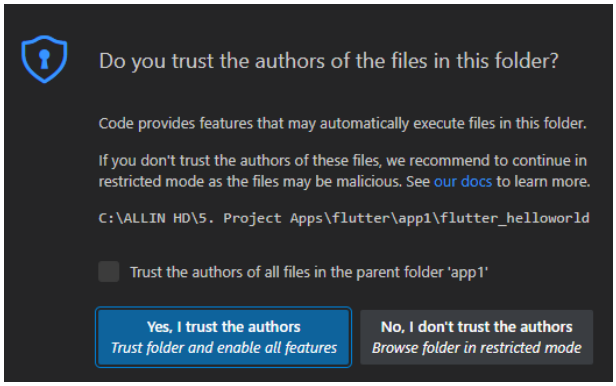


Kemudian kita akan diminta untuk menentukan lokasi penyimpanan project. Kita bisa menyimpan di tempat yang diinginkan. Pilih folder yang digunakan menyimpan project flutter.

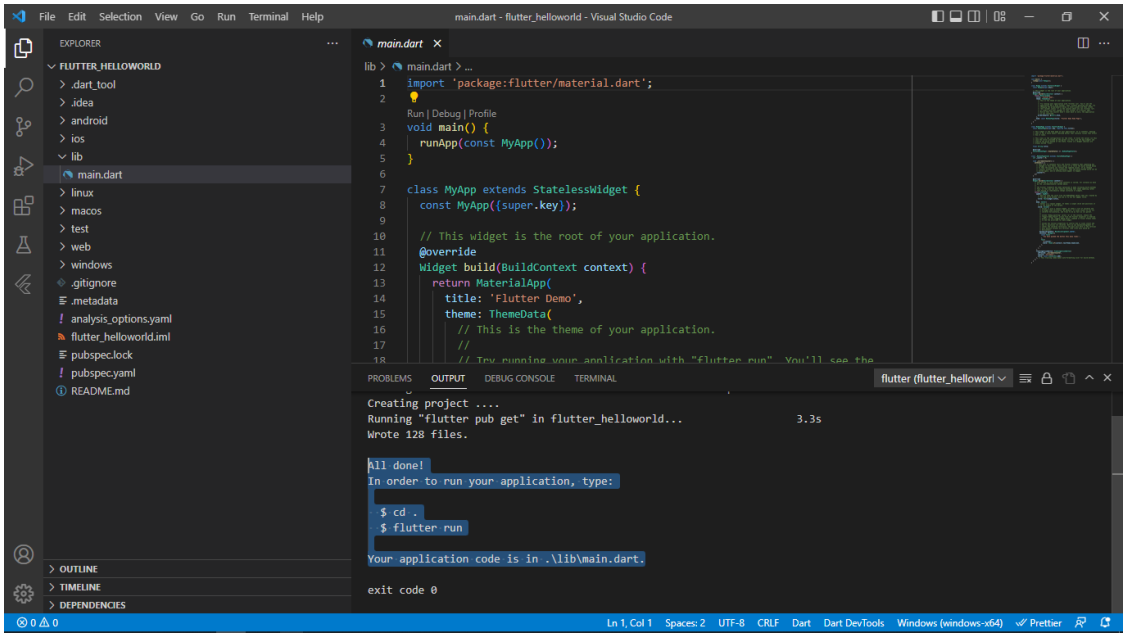
Kemudian kita akan diminta menentukan nama projectnya.



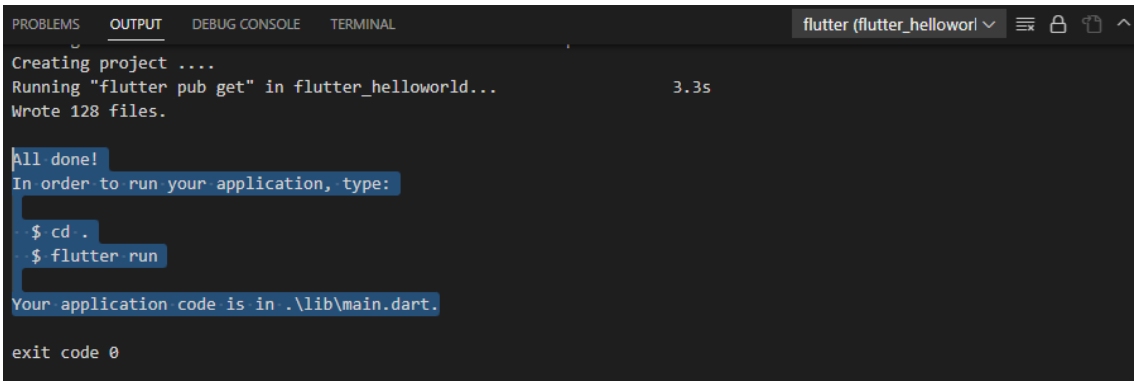
Kemudian pilih yes, I trust the author seperti di bawah.



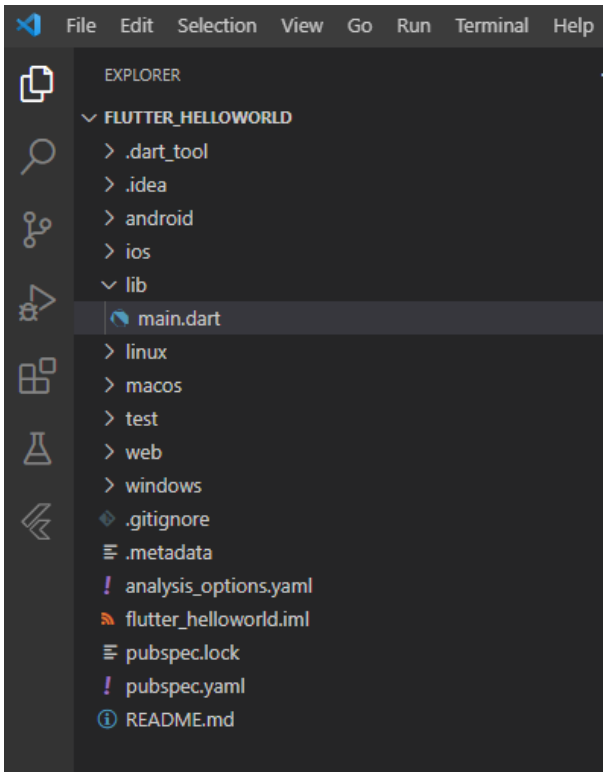
Project flutter_helloworld



Tampilan terminal

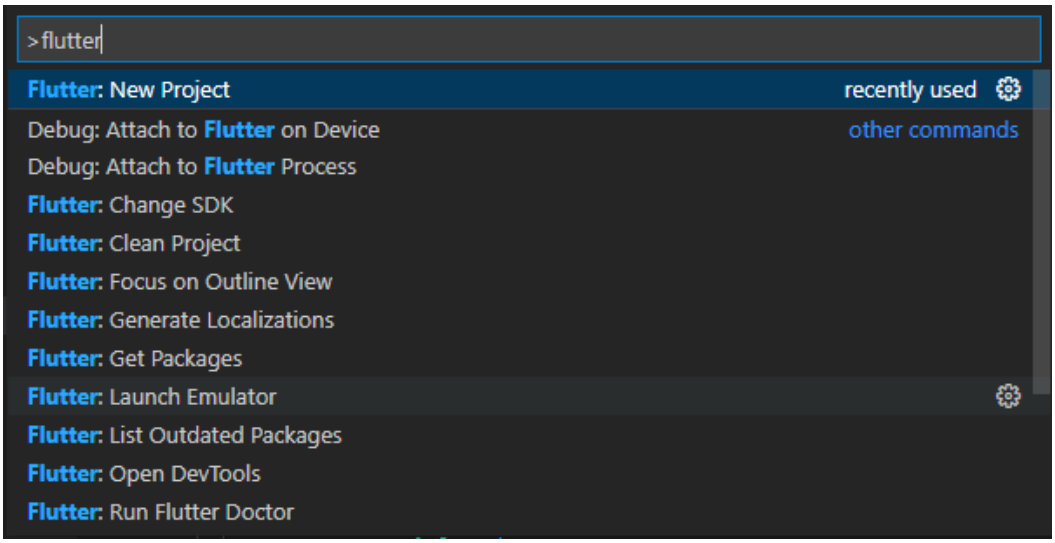


Struktur Flutter

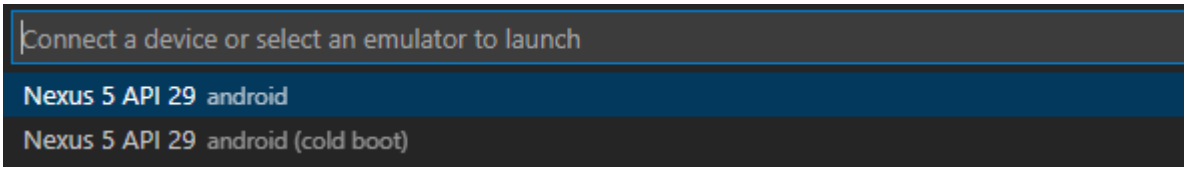


Step 6: Menjalankan Emulator

Tekan : ctrl + Shift + P . Ketik flutter Launch Emulator



Pilih emulatormu yang telah sebelumnya dibuat pada Android Studio



3. TUGAS

- Menyelesaikan Installasi.
- Membuat aplikasi pertama Hello World!

Modul 4

Flutter Framework : Widget

1. MATERI

a. Widgets

Semua yang berhubungan dengan komponen Flutter disebut dengan Widget, dimulai dari text input, button, radio button, label text, dan sejenisnya. Widget ini disusun secara hierarki dan ditampilkan pada layar. Pada flutter terdapat dua widget utama yang berperan sebagai container dari seluruh layar kita, yaitu antara lain **Stateful Widget** dan **Stateless Widget.**

Stateful Widget

Stateful Widget adalah widget yang dinamis atau dapat berubah-ubah. Dinamis yang dimaksud disini adalah setiap isi komponen dalam Stateful widget dapat berubah secara dinamis pada saat pengguna menggunakan aplikasi.

Code dasar dari Stateful Widget:

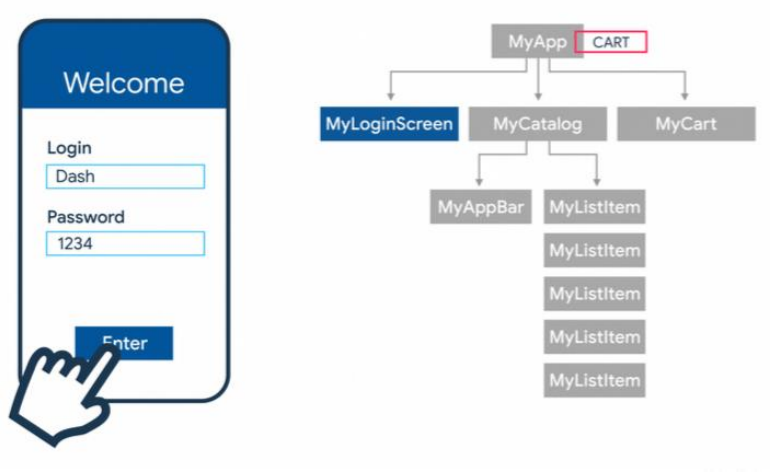
```
class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Container(

    );
  }
}
```

Stateless Widget

Stateless Widget adalah widget yang statis atau tidak dapat berubah-ubah. Widget ini biasanya digunakan sebagai komponen sekunder didalam stateful widget yang dapat diisi sebuah komponen kustomisasi yang dibuat oleh pengguna.

Code dasar dari Stateless Widget:

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(

    );
  }
}
```



MyLoginScreen, MyCatalog, dan MyCart merupakan Stateful Widget, hal ini dibuktikan karena mereka merupakan komponen utama yang dapat berubah isinya.

Sedangkan MyAppBar, dan MyListItem adalah Stateless Widget yang merupakan komponen statis dan telah dikustomisasi secara kebutuhan oleh developer. Kedua widget ini berada didalam Stateful Widget karena komponen ini tidak akan berubah dan bersifat statis, sedangkan MyListItem bisa saja berubah menjadi komponen lain dan hal ini diizinkan oleh Stateful Widget yang bersifat dinamis.

2. PRAKTIKUM

- A. Icon

Seperti yang kita tau, icon adalah gambar kecil untuk “mewakili” penjelasan sebuah object. pada aplikasi mobile, pada bagian atas aplikasi yang disebut dengan App Bar, kita akan melihat ada gambar kecil. Kita juga dapat menggunakan Icon untuk membuat gambar tersebut. Tapi kita hanya memanggil Icon — icon yang hanya tersedia di flutter.

Contoh syntax :

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home:Icon(
          Icons.access_alarm,
          size: 64.0,
          color: Theme.of(context).primaryColor,
        ),  // Icon
    );  // MaterialApp
  }
}
```

➢ B. Text

Berikutnya yaitu kita akan menampilkan sebuah teks. Tanpa teks di aplikasi kita, tentu saja semua terasa jadi aneh bukan? Text merupakan salah satu bagian penting dalam aplikasi.

Code Syntax:

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home:Text('Simple text demo.',
      style: TextStyle(
                  color: Colors.blue,
                  fontSize: 32.0,
                  fontStyle: FontStyle.italic,
                  decoration: TextDecoration.underline),
      ),
    );
  }
}
```

➢ C. Text Field

Jika dalam sebuah aplikasi kita suka melihat sebuah tempat untuk memasukkan teks (input), maka ini adalah widget tersebut di Flutter. Jelas sekali kegunaannya, yaitu untuk membuat sebuah text input.

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

void main() => runApp(MyApp());

//textfield sebuah widget yang digunakan user untuk memasukkan kalimat
class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  TextEditingController controller = TextEditingController();
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("text field"),
        ),
        body: Column(
          children: <Widget>[
            TextField(
              onChanged: (value) {
                setState(() {});
              },
              controller: controller,
            ),
            Text(controller.text)
          ],
        ),
      ),
    );
  }
}
```

➢ D. Button

Sebuah aplikasi harus memiliki sebuah tombol untuk menjalankan perintah lanjutan dari pengguna. Contohnya adalah ketika pengguna sudah menuliskan sesuatu yang dia cari pada textfield, maka dibutuhkan tombol untuk mencari apa yang sudah pengguna tulis. Oleh karena itu kita membutuhkan widget button.

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

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            body: ListView(
      children: <Widget>[
        Container(
          margin: EdgeInsets.symmetric(horizontal: 100),
          child: new ConstrainedBox(
            constraints: const BoxConstraints(minWidth:
double.infinity),
            child: new RaisedButton(
                color: Colors.blue,
                textColor: Colors.white,
                disabledColor: Colors.grey,
                disabledTextColor: Colors.black,
                splashColor: Colors.blueAccent,
                onPressed: () {},
                child: Text(
                  "Apply this buttom",
                  style: TextStyle(fontSize: 15),
                )
              ),
            ),
          )
        ],
      )
    ));
  }
}
```

➢ E. Layout

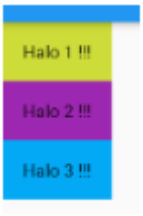
Untuk mengatur tata letak atau posisi widget, maka kita akan mengenal dua buah widget yaitu Row dan Column

Row akan menampilkan widget-widget secara horizontal atau sebaris dari kiri ke kanan, widget Row menggunakan property children artinya widget ini bisa diisi oleh banyak widget.


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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
    home: Scaffold(
```

```
        appBar: AppBar(
          title: Text("Row"),
        ),
        body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: <Widget>[
            Container(
              child: Text("Halo 1 !!!"),
              color: Colors.lime,
              padding: EdgeInsets.all(16.0),
            ),
            Container(
              child: Text("Halo 2 !!!"),
              color: Colors.purple,
              padding: EdgeInsets.all(16.0),
            ),
            Container(
              child: Text("Halo 3 !!!"),
              color: Colors.lightBlue,
              padding: EdgeInsets.all(16.0),
            ),
          ],
        ))
    );
  }
}
```

Column berlaku sebaliknya widget akan mengarah secara vertikal atau dari atas kebawah, widget ini juga menggunakan property children artinya widget ini bisa diisi oleh banyak widget. Coba ganti code syntax **Row** dengan **Column** seperti code di bawah ini.



column

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

3. TUGAS

- Membuat tampilan aplikasi dengan perpaduan min. 4 widgets dalam 1 halaman.

Modul 5

Flutter Framework : Navigation

1. MATERI

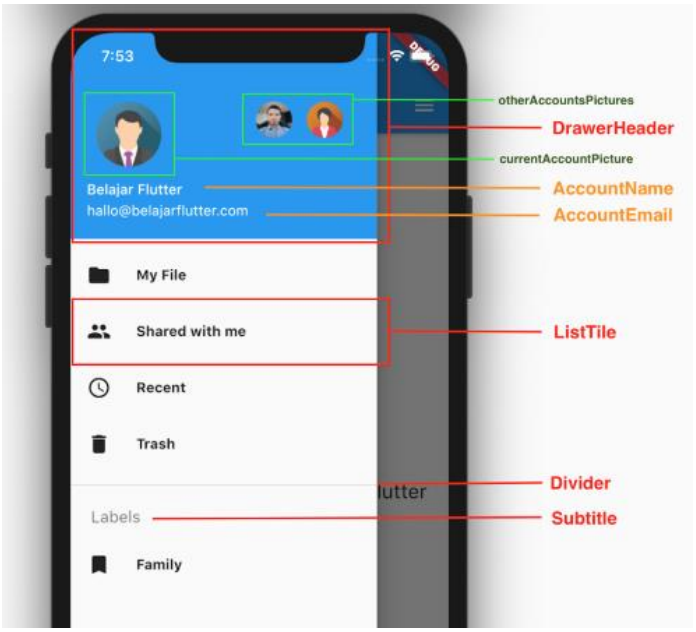
A. Navigation Drawer

Navigation Drawer merupakan menu navigasi yang tampil penuh pada sisi kanan atau kiri sebuah aplikasi. Untuk memunculkan Navigation Drawer bisa dengan cara di mengeser / swipe layar atau dengan menekan icon menu pada appBar. Pada Flutter, cara membuat navigation drawer sangat mudah yaitu hanya menggunakan widget Drawer yang dapat kita tempatkan pada properti drawer atau endDrawer di Scaffold Widget.

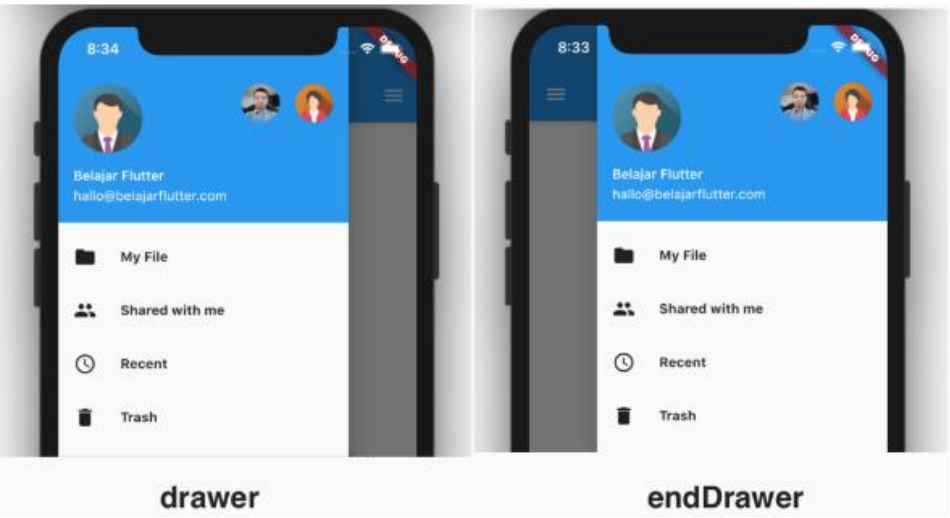
Sebelum ke tahap cara membuat navigation drawer, ada baiknya kita mengerti tentang Drawer itu sendiri. Drawer widget merupakan single child widget yang artinya hanya dapat memiliki satu child widget di dalamnya. Karena Drawer hanya memiliki properti child dan bukan children, maka untuk menampatkan item-item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

B. Anatomi Drawer

a. Header,
b. List Item,
c. Divider dan
d. Subtitle



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

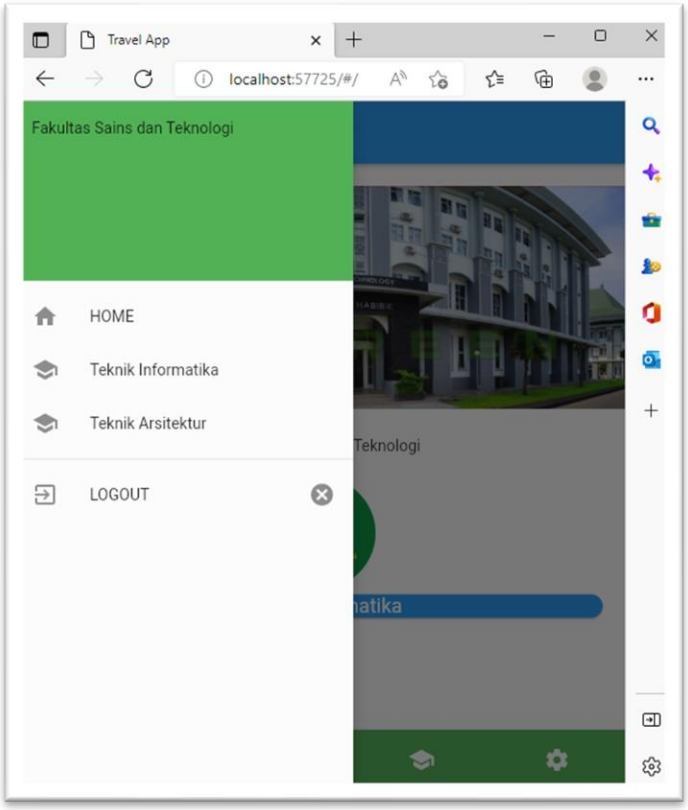


Posisi icon masing-masing drawer berbeda sesuai dengan properti yang digunakan

2. PRAKTIKUM

```
59      return Scaffold(
60        appBar: AppBar(
61          title: const Text('UIN Maliki Apps'),
62        ),  // AppBar
63        body: SizedBox(...),  // SizedBox
122
123       /*...*/
136       drawer: Drawer(
137         child: ListView(
138           padding: EdgeInsets.zero,
139           children: [
140             const DrawerHeader(
141               decoration: BoxDecoration(
142                 color: Colors.green,
143
144               ),  // BoxDecoration
145               child: Text('Fakultas Sains dan Teknologi'),
146             ),  // DrawerHeader
147             ListTile(
148               leading: Icon(Icons.home),
149               title: const Text('HOME'),
150               onTap: () => print('This is Home Menu')),  // ListTile
```

```
            ListTile(
              leading: Icon(Icons.school),
              title: const Text('Teknik Informatika'),
              onTap: () {
                Navigator.pop(context);
                Navigator.push(context,
                    MaterialPageRoute(builder: (context) => InformatikaPage(),
                    ));  // MaterialPageRoute
              },
            ),  // ListTile
            new Divider(),
            ListTile(
              leading: Icon(Icons.exit_to_app),
              trailing: new Icon(Icons.cancel),
              title: const Text('LOGOUT'),
              onTap: () {
                Navigator.of(context).pop();
              },
            ),  // ListTile
          ],
        ),  // ListView
      ),  // Drawer
```



3. TUGAS

1. Buatlah 3 Menu berbeda pada Drawer yang menautkan pada 3 halaman berbeda.
2. Gantilah posisi drawer yang sudah dibuat menggunakan endDrawer

Modul 6

Flutter Framework : Fetch Data From Internet

1. MATERI

A. Koneksi Restful API dengan http Flutter

Restful api merupakan media website yang digunakan untuk memberikan akses untuk mengambil resource dan data pada server masing – masing. Dengan menggunakan Restful API setiap program yang berjalan pada setiap platform akan dapat berkomunikasi satu dengan yang lain. Data yang dihasilkan oleh Restful API berupa json, dengan json tersebut data akan di kirim ke perangkat yang membutuhkan data tersebut. Selanjutnya data tersebut akan di olah oleh perangkat masing – masing dengan bahasa permrograman masing – masing dimana perangkat tersebut koneksi dengan Restful API.

Berlaku juga dengan Flutter, Flutter juga dapat memanfaatkan hal ini untuk mengambil data dari suatu server dengan menggunakan Restful Api. Bila Flutter telah dapat terhubung dengan Restful Api dari suatu server, programmer dapat melakukan aktifitas mengambil data dengan GET, POST untuk mengirim data, PUT untuk metode perubahan suatu data diserver, dan DELETE untuk menghapus data.

B. Persiapan untuk menghubungan ke Restful API

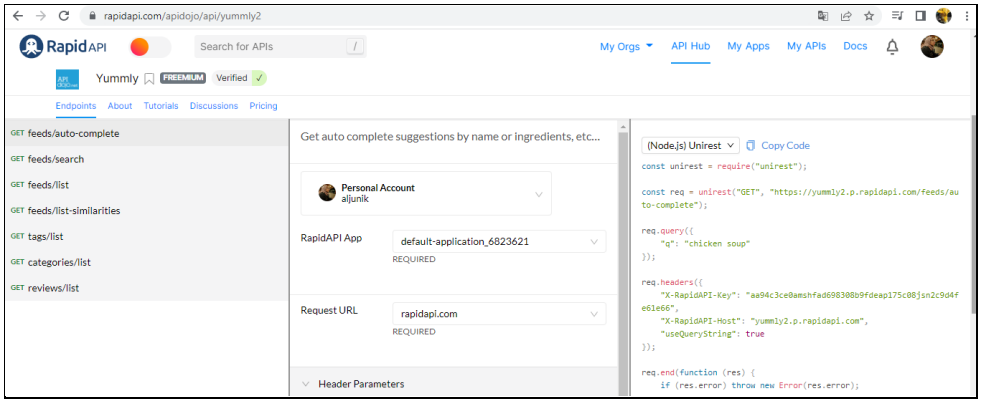
Sebelum melangkah cara menghubungkan ke Restful API, diperlukan untuk persiapan dari flutter dengan dependencies yang akan di gunakan pada Flutter. Dependencies yang diperlukan untuk Flutter dapat di lihat pada potongan syntaq pada di bawah ini :

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
  http: ^0.13.5
```

Potongan dari dependencies itu di letakan pada pubspec.yml dari flutter.

C. Menentukan Server Restful API

Untuk contoh penggunaan flutter untuk mengambil data dari server disini akan menggunakan Restful API dengan server di https://rapidapi.com/apidojo/api/yummly2



D. Model untuk menampung JSON dari Restful API

Hasil dari json akan di ubah dalam bentuk model yang membuat programmer akan mudah dalam pengambilan data tanpa harus mengentahui struktur data dari json sendiri. Untuk class model sendiri digunakan seperti.

```dart
class Album {
  final int userId;
  final int id;
  final String title;

  const Album({
    required this.userId,
    required this.id,
    required this.title,
  });

  factory Album.fromJson(Map<String, dynamic> json) {
    return Album(
      userId: json['userId'],
      id: json['id'],
      title: json['title'],
    );
  }
}
```

E. Mempersiapkan koneksi data ke server Restful API dengan http

Persiapan yang kita lakukan sebelum kita mengambil data dari suatu server dari Restful API. Persiapan yang dibutuhkan merupakan persiapan untuk membuat koneksi ke Restful API. Untuk membuat koneksi ini dapat digunakan dependencies dari Flutter berupa http.

```dart
Future<Album> fetchAlbum() async {
  final response = await http
      .get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    return Album.fromJson(jsonDecode(response.body));
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load album');
  }
}
```

F. Membuat suatu Widget untuk koneksi Restful API Flutter

Koneksi dan data telah di dapatkan dari server SWAPI sekarang cara untuk menampilkan data dari server ke Widget. Untuk menampilkan data tersebut dapat memanfaatkan widget dari FutureWidget, widget ini dipakai dikarenakan result dari suatu server menggunakan Future yang memberikan suatu delay pengambilan data. Hal ini disebabkan oleh pengambilan data dari server yang membutuhkan suatu waktu untuk data itu dapat di konsumsi oleh Aplikasi. Tampilan dari aplikasi hanya berupa list dari menu dari hasil yang didapatkan dari Restful API server.

```dart
class _MyAppState extends State<MyApp> {
  late Future<Album> futureAlbum;

  @override
  void initState() {
    super.initState();
    futureAlbum = fetchAlbum();
  }
  // ···
}
```

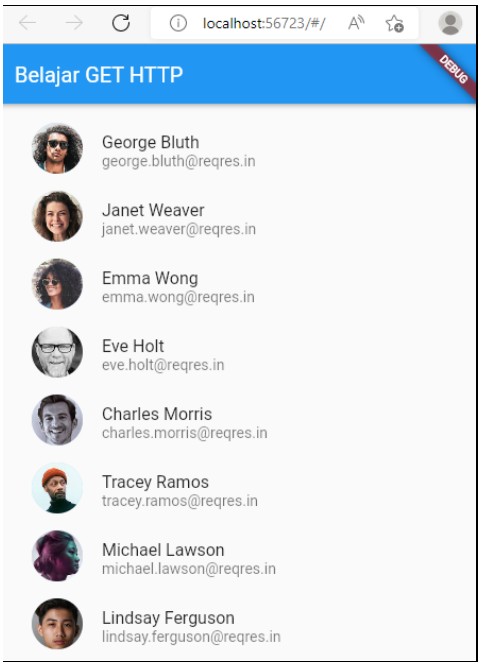
```dart
FutureBuilder<Album>(
  future: futureAlbum,
  builder: (context, snapshot) {
    if (snapshot.hasData) {
      return Text(snapshot.data!.title);
    } else if (snapshot.hasError) {
      return Text('${snapshot.error}');
    }

    // By default, show a loading spinner.
    return const CircularProgressIndicator();
  },
)
```

1. PRAKTIKUM

A. PRAKTIKUM 1 [Project 1 – fetch data from internet]

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

class BelajarGetData extends StatelessWidget {
  final String apiUrl = 'https://reqres.in/api/users?per_page=15';

  Future<List<dynamic>> _fecthDataUsers() async {
    final Uri url = Uri.parse(apiUrl);
    var result = await http.get(url);
    return json.decode(result.body)['data'];
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Belajar GET HTTP'),
      ),
      body: Container(
        child: FutureBuilder<List<dynamic>>(
          future: _fecthDataUsers(),
          builder: (BuildContext context, AsyncSnapshot snapshot) {
            if (snapshot.hasData) {
              return ListView.builder(
                  padding: EdgeInsets.all(10),
                  itemCount: snapshot.data.length,
                  itemBuilder: (BuildContext context, int index) {
                    return ListTile(
                      leading: CircleAvatar(
                        radius: 30,
                        backgroundImage:
                            NetworkImage(snapshot.data[index]['avatar']),
                      ),
                      title: Text(snapshot.data[index]['first_name'] +
                          " " +
                          snapshot.data[index]['last_name']),
                      subtitle: Text(snapshot.data[index]['email']),
                    );
                  });
            } else {
              return Center(child: CircularProgressIndicator());
            }
          },
```

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



B. PRAKTIKUM 2 [project 2 – static data]

main.dart

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Cooking Recipe - API Demo',
      theme: ThemeData(
          primarySwatch: Colors.blue,
          primaryColor: Colors.white,
          textTheme: TextTheme(bodyText2: TextStyle(color: Colors.white))),
```

```
      home: HomePage(),
    );
  }
}
```

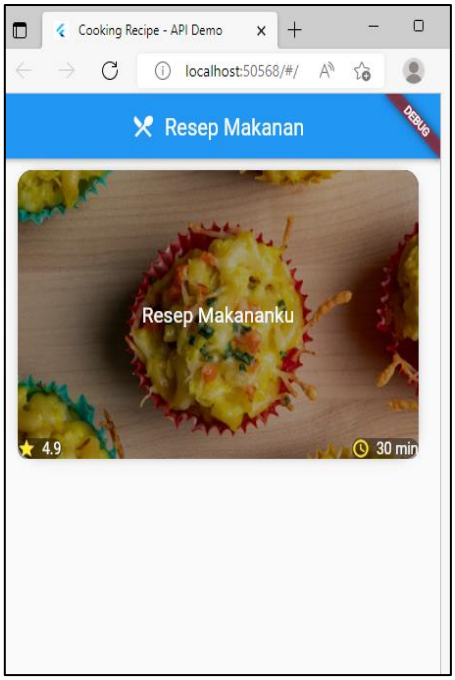
home.dart

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

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(Icons.restaurant_menu),
              SizedBox(
                width: 10,
              ),
              Text('Resep Makanan')
            ],
          ),
        ),
        body: ResepCard(
          title: 'Resep Makananku',
          rating: '4.9',
          cookTime: '30 min',
          thumbnailUrl:
              'https://lh3.googleusercontent.com/d7r4wwVc-
ZBiSUl_r5X7n5Ip7fD_DteFAgwf7bfLHRaOC6TVY623pLSHv5-
FipLQ3JBQwhjCQ1aVvFAfqlx1YWZ3XcA76kPFa6k=s360',
        ));
  }
}
```

resep_card.dart

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

class ResepCard extends StatelessWidget {
  final String title;
  final String rating;
  final String cookTime;
  final String thumbnailUrl;

  const ResepCard({
    required this.title,
    required this.rating,
    required this.cookTime,
    required this.thumbnailUrl,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
        margin: EdgeInsets.symmetric(horizontal: 22, vertical: 10),
        width: MediaQuery.of(context).size.width,
        height: 250,
        decoration: BoxDecoration(
            color: Colors.black,
            borderRadius: BorderRadius.circular(15),
            boxShadow: [
              BoxShadow(
                  color: Colors.black.withOpacity(0.6),
                  offset: Offset(0.0, 0.10),
                  blurRadius: 10.0,
                  spreadRadius: -6.0),
            ],
            image: DecorationImage(
                colorFilter: ColorFilter.mode(
                    Colors.black.withOpacity(0.35), BlendMode.multiply),
                image: NetworkImage(thumbnailUrl),
                fit: BoxFit.cover)),
        child: Stack(
          children: [
            Align(
              child: Padding(
                padding: EdgeInsets.symmetric(horizontal: 5.0),
                child: Text(title,
                    style: TextStyle(fontSize: 19),
                    overflow: TextOverflow.ellipsis,
                    maxLines: 2,
                    textAlign: TextAlign.center),
              ),
```

```
            alignment: Alignment.center,
          ),
          Align(
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Container(
                  decoration: BoxDecoration(
                      color: Colors.black.withOpacity(0.4),
                      borderRadius: BorderRadius.circular(15)),
                  child: Row(
                    children: [
                      Icon(
                        Icons.star,
                        color: Colors.yellow,
                        size: 18,
                      ),
                      SizedBox(width: 7),
                      Text(rating)
                    ],
                  ),
                ),
                Container(
                  decoration: BoxDecoration(
                      color: Colors.black.withOpacity(0.4),
                      borderRadius: BorderRadius.circular(15)),
                  child: Row(
                    children: [
                      Icon(
                        Icons.schedule,
                        color: Colors.yellow,
                        size: 18,
                      ),
                      SizedBox(width: 7),
                      Text(cookTime)
                    ],
                  ),
                ),
              ],
            ),
            alignment: Alignment.bottomLeft,
          )
        ],
      ));
  }
}
```



TUGAS : Belajar Flutter API

Modul 7

Flutter Framework : Restful API

1. MATERI

A. JavaScript Object Notation (JSON)

JavaScript object notation atau JSON adalah format yang digunakan untuk menyimpan dan mentransfer data. Ada dua hal yang perlu Anda pelajari agar dapat menggunakan JSON, yaitu syntax (cara penulisan) dan jenis value-nya. JSON selalu dibuka dan ditutup dengan tanda {} atau kurung kurawal. Syntax-nya terdiri dari dua elemen, yaitu key dan value. Keduanya dipisahkan oleh titik dua agar jelas.

```
{"city":"New York", "country":"United States "}
```

Anda dapat membuat nested object dan nested array dalam kode JSON. Untuk memahami penggunaannya, amati kedua contoh yang dibahas pada bagian ini.

Pertama, mari perhatikan contoh nested object berikut:

```
“karyawan”: {
        “nama”:“Anton”,
        “asal”:“Bandung”.
        “hobi”: {
                “hobi1”:“berenang”,
                “hobi2”:“melukis”,
                “hobi3”:“jogging”
        }
}
```

B. Membuat REST API Tiruan

Apa itu REST?

REST merupakan singkatan dari ***Representational State Transfer.*** Yang merupakan gaya arsitektur untuk merancang aplikasi yang saling terhubung. Dengan menggunakan HTTP sederhana untuk memungkinkan komunikasi antar mesin. Jadi, alih-alih menggunakan URL untuk memanipulasi beberapa informasi pengguna, REST mengirimkan permintaan HTTP seperti GET, POST, DELETE, dll ke URL untuk memanipulasi data.

Sebagai contoh, sebagai ganti membuat sebuah permintaan GET ke URL seperti **/deleteuser?id=10**, permintaannya akan menjadi seperti **DELETE /user/10**.

Mengapa kita membutuhkan sebuah REST API Tiruan?

REST API membentuk back-end untuk aplikasi mobile dan web. Ketika mengembangkan aplikasi, kadang-kadang Anda tidak memiliki REST API yang siap digunakan untuk tujuan pengembangan. Untuk

melihat mobile atau web app beraksi, kita memerlukan server yang melempar beberapa data JSON dummy.

Saat itulah REST API tiruan berfungsi. json-server menyediakan fungsi untuk mendirikan sebuah server REST API tiruan dengan praktis.

Referensi Membuat REST API Tiruan dengan MOCKAPI.IO :

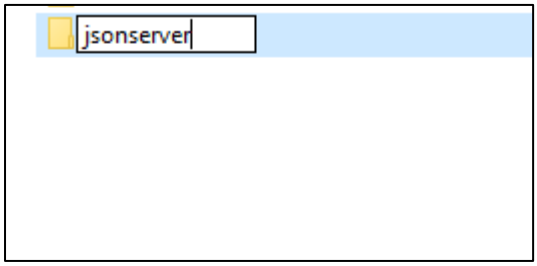
https://www.youtube.com/watch?v=VhDDnJs6CM4

2. PRAKTIKUM

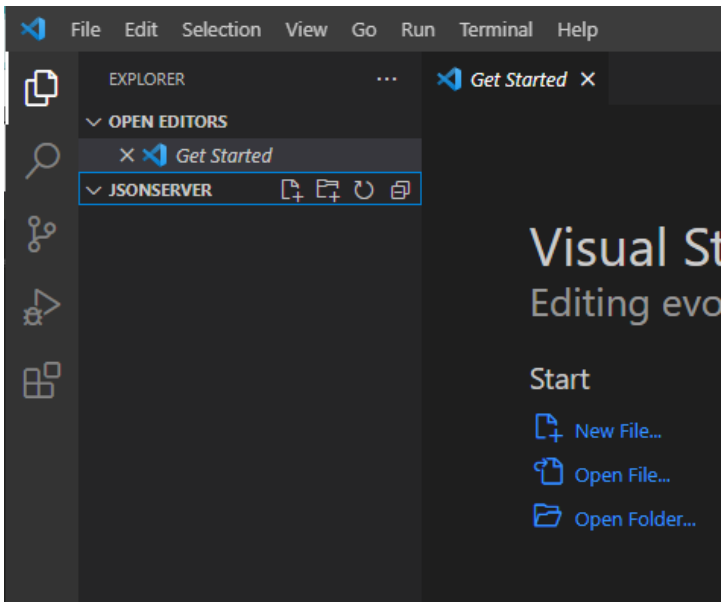
PRAKTIKUM 1.

Membuat REST API Tiruan dengan JSON Server

Sebelum memulai terlebih dahulu kita buat folder dengan nama **jsonserver**



Buka folder tersebut dengan IDE. Kita akan memulai menginstall JSON Server pada folder menggunakan **Node Package Manager** (npm). Pastikan npm telah terinstall sebelumnya.

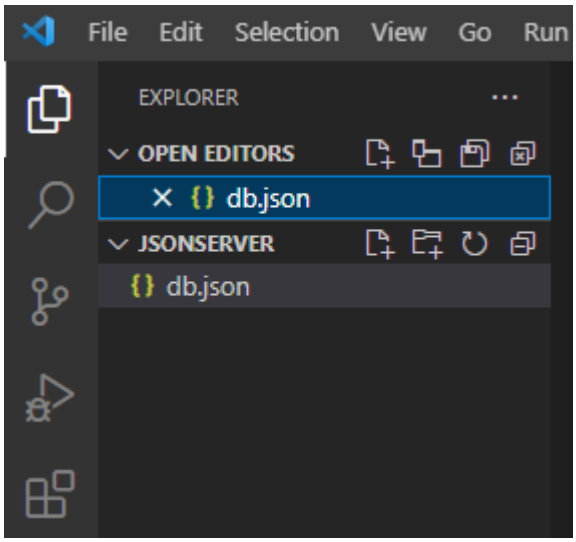


Selanjutnya lakukan installasi JSON Server dengan code seperti berikut

```
npm install -g json-server
```

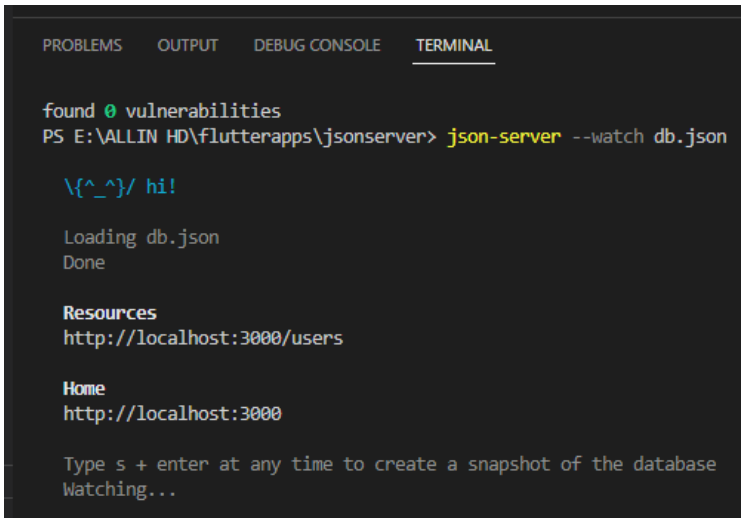
```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\ALLIN HD\flutterapps\jsonserver> npm install -g json-server

added 109 packages, and audited 110 packages in 2m

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS E:\ALLIN HD\flutterapps\jsonserver>
```

Selanjutnya buat **db.json** untuk menampung data yang akan digunakan. Buat file JSON dummy dengan beberapa data sesuai kebutuhan Anda.

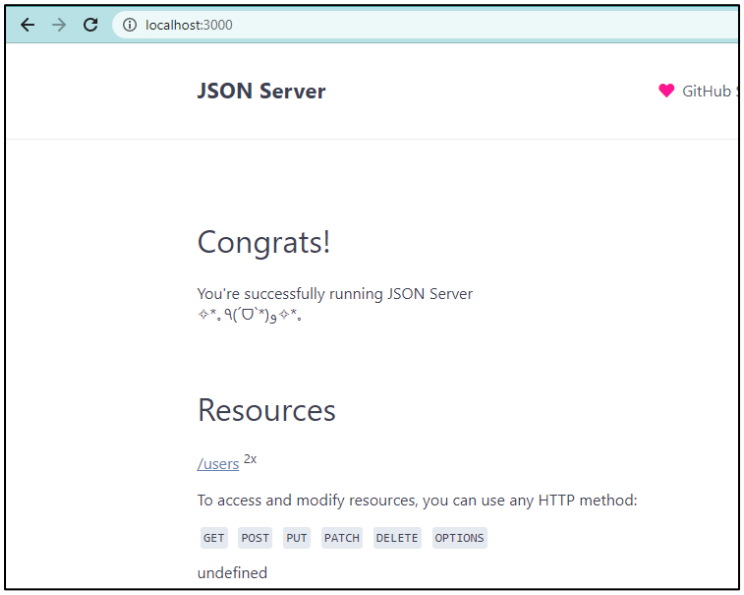



```json
{
    "users": [{
        "id": 1,
        "name": "roy",
        "email": "roy21@gmail.com",
        "age" : 21
    }, {
        "id": 2,
        "name": "anita",
        "email": "anita23@gmail.com",
        "age" : 23
    }]
}
```

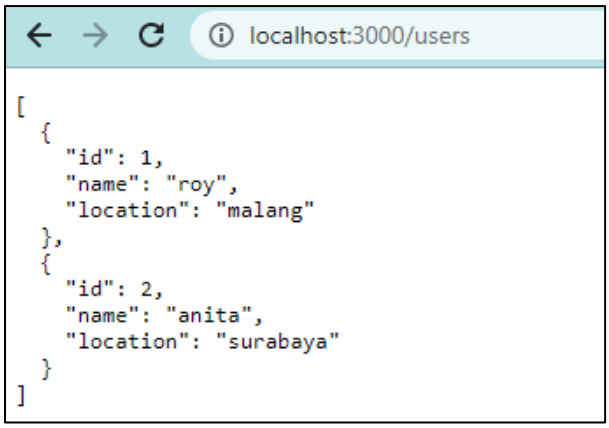
Jalankan JSON Server dengan kode sebagai berikut

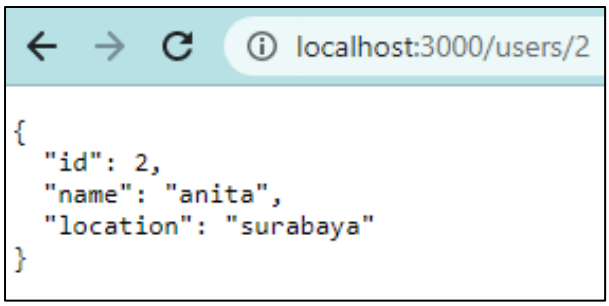


Buka browser dan ketikkan URL Home http://localhost:3000/

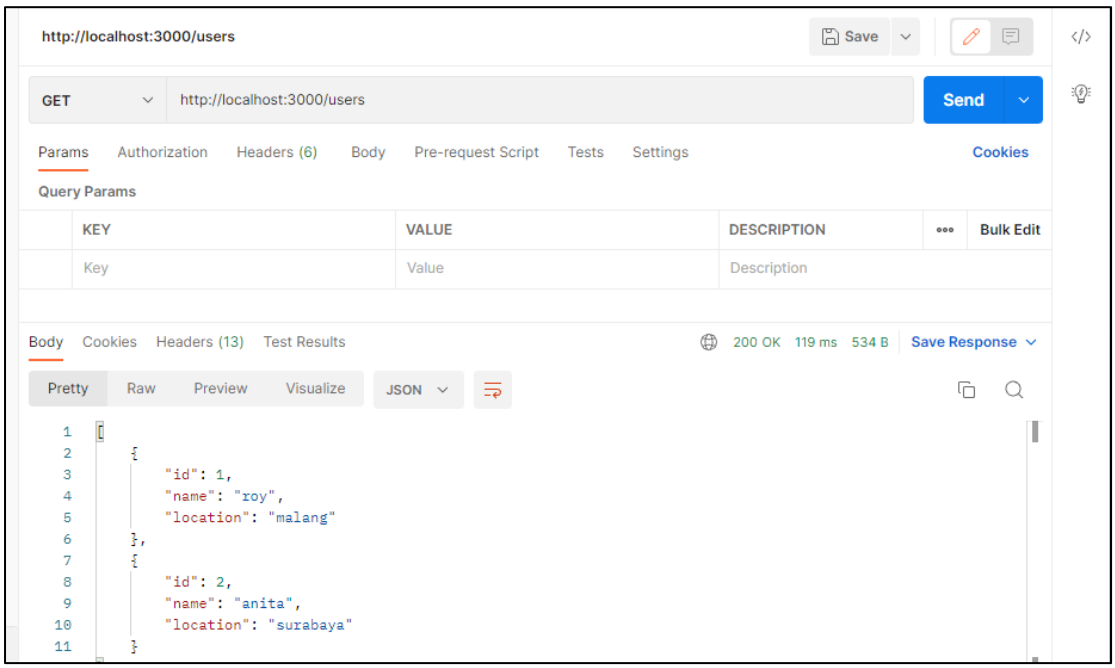


Untuk melihat data yang telah kita masukkan dalam db.json dapat mengetikkan URL http://localhost:3000/users





Kita juga dapat menggunakan aplikasi pembantu seperti **POSTMAN** untuk menguji response API



Apabila sudah sering berkutat dengan ***API (Application Programming Interface),*** maka Anda pasti familiar dengan aplikasi Postman. Ini adalah alat yang tergolong populer untuk melakukan testing API atau menyederhanakan alur kerja dan pengembangannya.

Postman adalah developing tools yang membantu penggunanya untuk membangun, menguji, dan memodifikasi API. Ketika menjalankan pengujian, Postman mengirim request API ke server web dan kemudian menerima segala jenis respons. Sesuai dengan namanya, ia berfungsi layaknya tukang pos.

Postman juga menawarkan banyak metode endpoint. Berikut ini yang paling populer beserta fungsinya:

GET: Dapatkan informasi
POST: Tambahkan informasi
PUT: Ganti informasi
PATCH: Perbarui informasi tertentu
DELETE: Hapus informasi

PRAKTIKUM 2.

Membuat Aplikasi Mobile dengan Implementasi Manipulasi Data API pada JSON Server dan HTTP Request.

main.dart

```
import 'package:api6flutter/home_page.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomeScreen(),
    );
  }
}
```

koneksi.dart

```
import 'package:api6flutter/model.dart';
import 'package:http/http.dart' as http;
```

```
class ApiService {
  var client = http.Client();

  Future<List<Profile>?> getProfiles() async {
    var uri = Uri.parse('http://localhost:3000/users');
    var response = await client.get(uri);

    if (response.statusCode == 200) {
      return profileFromJson(response.body);
    } else {
      return null;
    }
  }
}
```

model.dart

```
import 'dart:convert';

class Profile {
  int id;
  String name;
  String email;
  int age;

  Profile(
      {required this.id,
      required this.name,
      required this.email,
      required this.age});

  factory Profile.fromJson(Map<String, dynamic> map) {
    return Profile(
        id: map["id"], name: map["name"], email: map["email"], age:
map["age"]);
  }

  Map<String, dynamic> toJson() {
    return {"id": id, "name": name, "email": email, "age": age};
  }

  @override
  String toString() {
    return 'Profile{id: $id, name: $name, email: $email, age: $age}';
  }
}
```

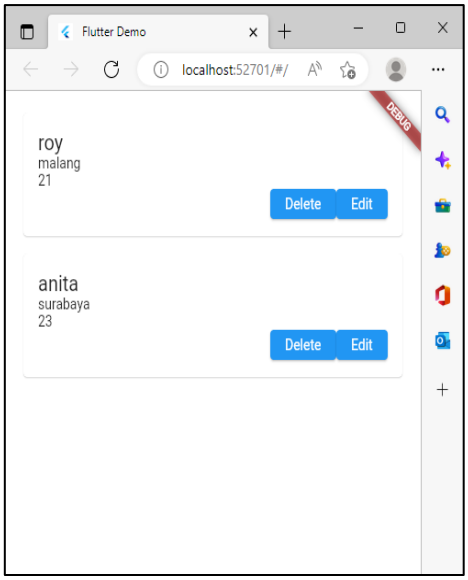
```
List<Profile> profileFromJson(String jsonData) {
  final data = json.decode(jsonData);
  return List<Profile>.from(data.map((item) => Profile.fromJson(item)));
}

String profileToJson(Profile data) {
  final jsonData = data.toJson();
  return json.encode(jsonData);
}
```

home_page.dart

```
import 'package:api6flutter/koneksi.dart';
import 'package:api6flutter/model.dart';
import 'package:flutter/material.dart';

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<Profile>? posts;
  var isLoaded = false;

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

  getData() async {
    //post=await
    posts = await ApiService().getProfiles();
    if (posts != null) {
      setState(() {
        isLoaded = true;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: FutureBuilder(
        future: ApiService().getProfiles(),
        builder:
```

```
            (BuildContext context, AsyncSnapshot<List<Profile>?> snapshot) {
          if (snapshot.hasError) {
            return Center(
              child: Text(
                  "Something wrong with message:
${snapshot.error.toString()}"),
            );
          } else if (snapshot.connectionState == ConnectionState.done) {
            List<Profile>? profiles = snapshot.data;
            return _buildListView(profiles!);
          } else {
            return Center(
              child: CircularProgressIndicator(),
            );
          }
        },
      ),
    );
  }

  Widget _buildListView(List<Profile> profiles) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
      child: ListView.builder(
        itemBuilder: (context, index) {
          Profile profile = profiles[index];
          return Padding(
            padding: const EdgeInsets.only(top: 8.0),
            child: Card(
              child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    Text(
                      profile.name,
                      style: Theme.of(context).textTheme.titleLarge,
                    ),
                    Text(profile.email),
                    Text(profile.age.toString()),
                    Row(
                      mainAxisAlignment: MainAxisAlignment.end,
                      children: <Widget>[
                        ElevatedButton(
                          onPressed: () {
                            // TODO: do something in here
                          },
                          child: Text("Delete"),
```

```
                        ),
                        ElevatedButton(
                          onPressed: () {
                            // TODO: do something in here
                          },
                          child: Text("Edit"),
                        ),
                      ],
                    ),
                  ],
                ),
              ),
            ),
          );
        },
        itemCount: profiles.length,
      ),
    );
  }
}
```



3. TUGAS

- **Melakukan POST Data pada API (Add/Save)**
- **Referensi untuk melakukan manipulasi data melalui Postman**
  https://www.indeveloper.id/2019/12/tutorial-melakukan-testing-api.html

Modul 8

Flutter Framework : Operasi CRUD data JSON

1. MATERI

A. JavaScript Object Notation (JSON)

JavaScript object notation atau JSON adalah format yang digunakan untuk menyimpan dan mentransfer data. Ada dua hal yang perlu Anda pelajari agar dapat menggunakan JSON, yaitu syntax (cara penulisan) dan jenis value-nya. JSON selalu dibuka dan ditutup dengan tanda {} atau kurung kurawal. Syntax-nya terdiri dari dua elemen, yaitu key dan value. Keduanya dipisahkan oleh titik dua agar jelas.

```
{"city":"New York", "country":"United States "}
```

Anda dapat membuat nested object dan nested array dalam kode JSON. Untuk memahami penggunaannya, amati kedua contoh yang dibahas pada bagian ini.

Pertama, mari perhatikan contoh nested object berikut:

```
“karyawan”: {
        “nama”:“Anton”,
        “asal”:“Bandung”.
        “hobi”: {
                “hobi1”:“berenang”,
                “hobi2”:“melukis”,
                “hobi3”:“jogging”
        }
}
```

B. Membuat REST API Tiruan

Apa itu REST?

REST merupakan singkatan dari ***Representational State Transfer.*** Yang merupakan gaya arsitektur untuk merancang aplikasi yang saling terhubung. Dengan menggunakan HTTP sederhana untuk memungkinkan komunikasi antar mesin. Jadi, alih-alih menggunakan URL untuk memanipulasi beberapa informasi pengguna, REST mengirimkan permintaan HTTP seperti GET, POST, DELETE, dll ke URL untuk memanipulasi data.

Sebagai contoh, sebagai ganti membuat sebuah permintaan GET ke URL seperti **/deleteuser?id=10**, permintaannya akan menjadi seperti **DELETE /user/10**.

Mengapa kita membutuhkan sebuah REST API Tiruan?

REST API membentuk back-end untuk aplikasi mobile dan web. Ketika mengembangkan aplikasi, kadang-kadang Anda tidak memiliki REST API yang siap digunakan untuk tujuan pengembangan. Untuk

melihat mobile atau web app beraksi, kita memerlukan server yang melempar beberapa data JSON dummy.

Saat itulah REST API tiruan berfungsi. json-server menyediakan fungsi untuk mendirikan sebuah server REST API tiruan dengan praktis.

Referensi Membuat REST API Tiruan dengan MOCKAPI.IO :

https://www.youtube.com/watch?v=VhDDnJs6CM4

C. Fungsi Create Read Update Delete Pada Flutter (CRUD)

Anda mungkin sudah sering mendengar istilah CRUD. Pada umumnya penggunaan JSON digunakan untuk melakukan komunikasi data dari client ke server maupun dari server ke client. Menampilkan data dari server, Tambah data ke server, Edit data ke server, Hapus data dari server

Saat kita membangun API, hal yang terpenting dari API adalah kemampuannya dalam menyediakan empat tipe fungsi dasar tersebut. Ilmuwan dan ahli komputer sering menyebut fungsi-fungsi ini dengan akronim CRUD. Sebuah model harus memiliki kemampuan untuk melakukan paling banyak empat fungsi ini.

Referensi pertama untuk operasi CRUD datang dari Haim Kilov pada tahun 1990 dalam sebuah artikel berjudul, "Dari semantik ke pemodelan data berorientasi objek." Namun, istilah ini pertama kali dipopulerkan oleh buku James Martin tahun 1983, *Managing the Data-base Environment*. Berikut rinciannya:

- **CREATE:** prosedur yang digunakan untuk melakukan pernyataan INSERT untuk membuat record baru.
- **READ:** prosedur yang digunakan untuk membaca catatan tabel berdasarkan keynoted utama dalam parameter input.
- **UPDATE:** prosedur yang dipakai untuk engeksekusi pernyataan UPDATE pada tabel berdasarkan kunci utama yang ditentukan untuk catatan dalam klausa WHERE dari pernyataan tersebut.
- **DELETE:** prosedur yang dipakai untuk menghapus baris tertentu dalam klausa WHERE.

2. PRAKTIKUM

main.dart

```
import 'package:api6flutter/home_page.dart';
import 'package:api6flutter/update.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomeScreen(),
      routes: {
        'home': (context) => HomeScreen(),
        'update': (context) => updateUser()
      },
    );
  }
}
```

koneksi.dart

```
import 'dart:convert';

import 'package:api6flutter/home_page.dart';
import 'package:api6flutter/model.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class ApiService {
  var client = http.Client();

  Future<List<Profile>?> getProfiles() async {
    var uri = Uri.parse('http://localhost:3000/users');
    var response = await client.get(uri);
    if (response.statusCode == 200) {
      return profileFromJson(response.body);
    } else {
      return null;
    }
  }

  Future<bool> saveProfile(
      String id, String name, String email, String age) async {
    var uri = Uri.parse('http://localhost:3000/users/?');
    final response = await client.post(uri,
        // headers: {"content-type": "application/json"},
        // body: profileToJson(data),
```

```
        body: ({'id': id, 'name': name, 'email': email, 'age': age}));

    if (response.statusCode == 201) {
      print('berhasil simpan');
      return true;
    } else {
      print('Gagal simpan');
      return false;
    }
  }

  Future deleteProfiles(String id) async {
    var uri = Uri.parse('http://localhost:3000/users/' + id);
    final response = await client.delete(
      uri,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
    );
    if (response.statusCode == 200) {
      return true;
    } else {
      return false;
    }
  }

  Future updateProfiles(
      String id, String name, String email, String age) async {
    var uri = Uri.parse('http://localhost:3000/users/' + id);
    final response = await client.put(uri,
        body: ({'id': id, 'name': name, 'email': email, 'age': age}));

    if (response.statusCode == 200) {
      print('berhasil diupdate');
      return true;
    } else {
      print('Gagal diupdate');
      return false;
    }
  }
}
```

model.dart

```
import 'dart:convert';

class Profile {
```

```
  dynamic id;
  String name;
  String email;
  dynamic age;

  Profile(
      {required this.id,
      required this.name,
      required this.email,
      required this.age});

  factory Profile.fromJson(Map<String, dynamic> map) {
    return Profile(
        id: map["id"], name: map["name"], email: map["email"], age:
map["age"]);
  }

  Map<String, dynamic> toJson() {
    return {"id": id, "name": name, "email": email, "age": age};
  }

  @override
  String toString() {
    return 'Profile{id: $id, name: $name, email: $email, age: $age}';
  }
}

List<Profile> profileFromJson(String jsonData) {
  final data = json.decode(jsonData);
  return List<Profile>.from(data.map((item) => Profile.fromJson(item)));
}

String profileToJson(Profile data) {
  final jsonData = data.toJson();
  return json.encode(jsonData);
}
```

home_page.dart

```
import 'dart:async';
import 'package:api6flutter/add.dart';
import 'package:api6flutter/koneksi.dart';
import 'package:api6flutter/model.dart';
import 'package:api6flutter/update.dart';
import 'package:flutter/material.dart';

class HomeScreen extends StatefulWidget {
```

```
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  List<Profile>? posts;
  var isLoaded = false;

  getData() async {
    posts = await ApiService().getProfiles();
    if (posts != null) {
      setState(() {
        isLoaded = true;
      });
    }
  }

  FutureOr onGoBack() {
    getData();
  }

  void navigateAddProfile() {
    Route route = MaterialPageRoute(builder: (context) => addUser());
    Navigator.push(context, route).then((value) => onGoBack());
  }

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text('Users Data'),
        ),
        body: Visibility(
          visible: isLoaded,
          child: FutureBuilder(
            future: ApiService().getProfiles(),
            builder:
                (BuildContext context, AsyncSnapshot<List<Profile>?> snapshot)
{
              if (snapshot.hasError) {
                return Center(
                  child: Text(
```

```
                    "Something wrong with message:
${snapshot.error.toString()}"),
                );
              } else if (snapshot.connectionState == ConnectionState.done) {
                List<Profile>? profiles = snapshot.data;
                return _buildListView(profiles!);
              } else {
                return Center(
                  child: CircularProgressIndicator(),
                );
              }
            },
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            Navigator.of(context)
                .push(MaterialPageRoute(builder: (context) => addUser()));
            navigateAddProfile();
          },
          tooltip: 'Tambah Data',
          child: Icon(Icons.add),
          backgroundColor: Colors.blue,
        ));
  }

  Widget _buildListView(List<Profile> profiles) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
      child: ListView.builder(
        itemBuilder: (context, index) {
          Profile profile = profiles[index];
          return Padding(
            padding: const EdgeInsets.only(top: 8.0),
            child: Card(
              child: Padding(
                padding: const EdgeInsets.all(16.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    Text(
                      profile.name,
                      style: Theme.of(context).textTheme.titleLarge,
                    ),
                    Text(profile.email),
                    Text(profile.id.toString()),
                    Text(profile.age.toString()),
                    Row(
```

```
                    mainAxisAlignment: MainAxisAlignment.end,
                    children: <Widget>[
                      IconButton(
                        icon: Icon(Icons.delete),
                        onPressed: () async {
                          bool respon =
                              await ApiService().deleteProfiles(profile.id);
                          if (respon) {
                            print('Delete Data Success');
                          } else {
                            print('Delete Data Failed');
                          }
                          getData();
                        },
                      ),
                      ElevatedButton(
                        onPressed: () {
                          Navigator.popAndPushNamed(context, 'update',
                              arguments: [
                                profile.id,
                                profile.name,
                                profile.email,
                                profile.age
                              ]);
                        },
                        child: Text("Edit"),
                      ),
                    ],
                  ),
                ],
              ),
            ),
          ),
        );
      },
      itemCount: profiles.length,
    ),
  );
 }
}
```

add.dart

```
import 'package:flutter/material.dart';
import 'package:api6flutter/koneksi.dart';
import 'package:rflutter_alert/rflutter_alert.dart';
```

```
class addUser extends StatefulWidget {
  const addUser({super.key});

  @override
  State<addUser> createState() => _addUserState();
}

class _addUserState extends State<addUser> {
  TextEditingController idfield = TextEditingController();
  TextEditingController namefield = TextEditingController();
  TextEditingController emailfield = TextEditingController();
  TextEditingController agefield = TextEditingController();
  int count = 0;

  void createProfile() {
    ApiService()
        .saveProfile(
            idfield.text, namefield.text, emailfield.text, agefield.text)
        .then((value) {
      setState(() {
        if (value) {
          Alert(
              context: context,
              title: 'Success',
              desc: 'Data Berhasil disimpan',
              type: AlertType.success,
              buttons: [
                DialogButton(
                    child: Text('ok',
                        style: TextStyle(color: Colors.white, fontSize: 18)),
                    onPressed: () {
                      Navigator.of(context).popUntil((_) => count++ >= 2);
                    })
              ]).show();
        } else {
          Alert(
              context: context,
              title: 'Failed',
              desc: 'Data Gagal disimpan',
              type: AlertType.error,
              buttons: [
                DialogButton(
                    child: Text('ok',
                        style: TextStyle(color: Colors.white, fontSize: 18)),
                    onPressed: () {
                      Navigator.pop(context);
                    })
              ]).show();
```

```
        }
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text('Add Data',
              style: TextStyle(color: Colors.black, fontSize: 26))),
      body: SingleChildScrollView(
        child: Column(
          children: [
            TextField(
              controller: idfield,
              decoration: InputDecoration(
                hintText: 'Masukkan ID',
                labelText: 'ID User',
                icon: Icon(Icons.assignment_outlined),
              ),
            ),
            TextField(
              controller: namefield,
              decoration: InputDecoration(
                hintText: 'Masukkan Nama',
                labelText: 'Name Users',
                icon: Icon(Icons.people),
              ),
            ),
            TextField(
              controller: emailfield,
              decoration: InputDecoration(
                hintText: 'Masukkan Email',
                labelText: 'Email User',
                icon: Icon(Icons.email),
              ),
              keyboardType: TextInputType.emailAddress,
            ),
            TextField(
              controller: agefield,
              decoration: InputDecoration(
                hintText: 'Masukkan Usia',
                labelText: 'Age User',
                icon: Icon(Icons.calendar_month_sharp),
              ),
              keyboardType: TextInputType.number,
            ),
```

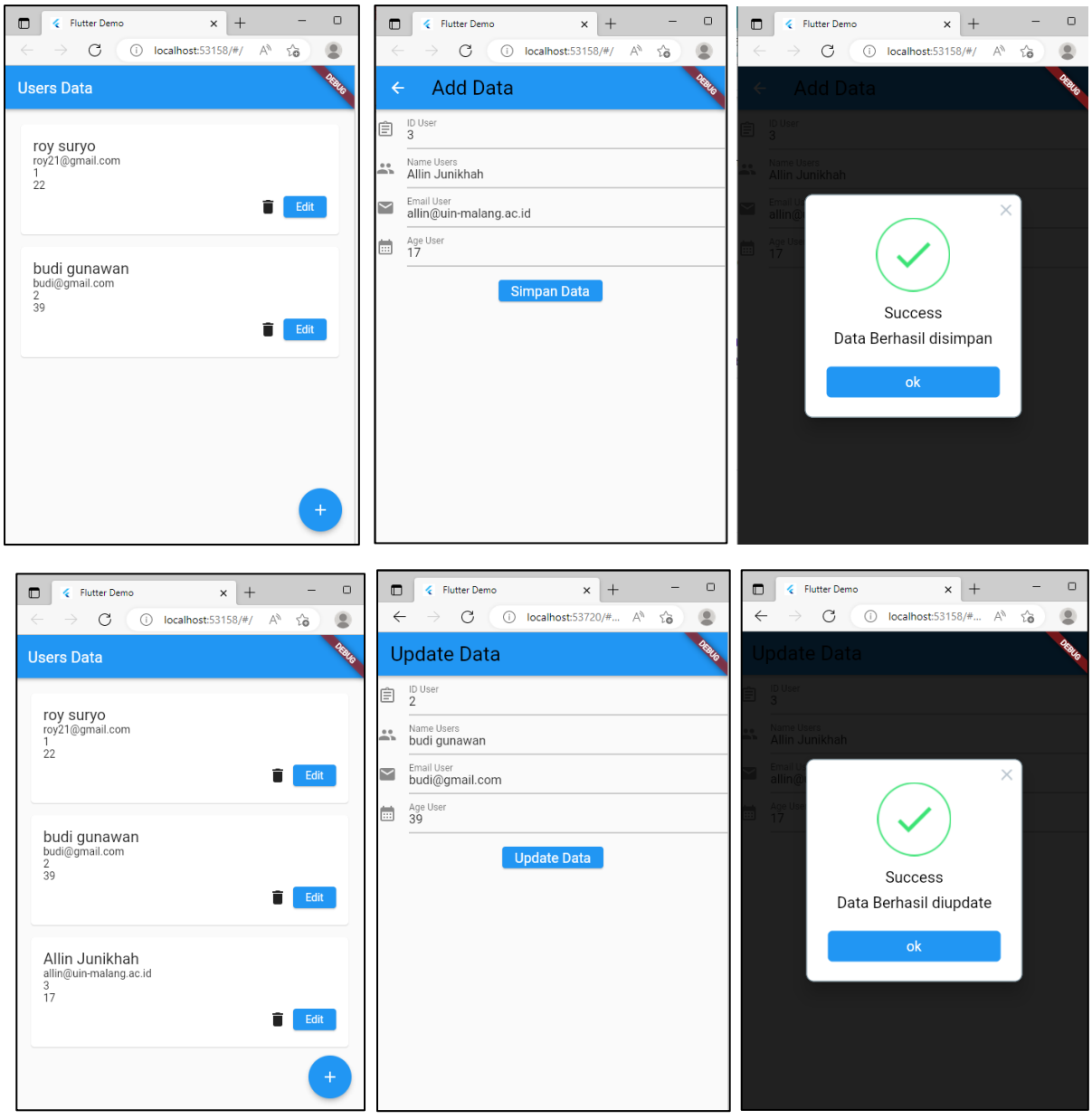
```
            SizedBox(
              height: 18,
            ),
            ElevatedButton(
                onPressed: () {
                  createProfile();
                },
                child: Text('Simpan Data', style: TextStyle(fontSize: 18))),
          ],
        ),
      ),
    );
  }
}
```

update.dart

```
import 'package:api6flutter/model.dart';
import 'package:flutter/material.dart';
import 'package:api6flutter/koneksi.dart';
import 'package:rflutter_alert/rflutter_alert.dart';

class updateUser extends StatefulWidget {
  const updateUser({super.key});

  @override
  State<updateUser> createState() => _updateUserState();
}

class _updateUserState extends State<updateUser> {
  TextEditingController idfield = TextEditingController();
  TextEditingController namefield = TextEditingController();
  TextEditingController emailfield = TextEditingController();
  TextEditingController agefield = TextEditingController();
  int count = 0;

  void updateProfile() {
    ApiService()
        .updateProfiles(
            idfield.text, namefield.text, emailfield.text, agefield.text)
        .then((value) {
      setState(() {
        if (value) {
          Alert(
              context: context,
              title: 'Success',
              desc: 'Data Berhasil diupdate',
```

```
            type: AlertType.success,
            buttons: [
              DialogButton(
                  child: Text('ok',
                      style: TextStyle(color: Colors.white, fontSize: 18)),
                  onPressed: () {
                    Navigator.of(context).popUntil((_) => count++ >= 2);
                  })
            ]).show();
      } else {
        Alert(
            context: context,
            title: 'Failed',
            desc: 'Data Gagal diupdate',
            type: AlertType.error,
            buttons: [
              DialogButton(
                  child: Text('ok',
                      style: TextStyle(color: Colors.white, fontSize: 18)),
                  onPressed: () {
                    Navigator.pop(context);
                  })
            ]).show();
      }
    });
  });
}

@override
Widget build(BuildContext context) {
  final args = ModalRoute.of(context)?.settings.arguments as List<dynamic>;
  idfield.text = args[0].toString();
  namefield.text = args[1];
  emailfield.text = args[2];
  agefield.text = args[3].toString();

  return Scaffold(
    appBar: AppBar(
        title: Text('Update Data',
            style: TextStyle(color: Colors.black, fontSize: 26))),
    body: SingleChildScrollView(
      child: Column(
        children: [
          TextField(
            controller: idfield,
            decoration: InputDecoration(
              hintText: 'Masukkan ID',
              labelText: 'ID User',
```

```
              icon: Icon(Icons.assignment_outlined),
            ),
          ),
          TextField(
            controller: namefield,
            decoration: InputDecoration(
              hintText: 'Masukkan Nama',
              labelText: 'Name Users',
              icon: Icon(Icons.people),
            ),
          ),
          TextField(
            controller: emailfield,
            decoration: InputDecoration(
              hintText: 'Masukkan Email',
              labelText: 'Email User',
              icon: Icon(Icons.email),
            ),
            keyboardType: TextInputType.emailAddress,
          ),
          TextField(
            controller: agefield,
            decoration: InputDecoration(
              hintText: 'Masukkan Usia',
              labelText: 'Age User',
              icon: Icon(Icons.calendar_month_sharp),
            ),
            keyboardType: TextInputType.number,
          ),
          SizedBox(
            height: 18,
          ),
          ElevatedButton(
              onPressed: () {
                updateProfile();
              },
              child: Text('Update Data', style: TextStyle(fontSize: 18))),
        ],
      ),
    ),
  );
 }
}
```



3. TUGAS

- **Progress ke-1 Final Project UAS Mata Kuliah Praktikum Pemrograman Mobile**