MODUL DIKLAT

PENINGKATAN KOMPETENSI GURU TIK

PENGABDIAN KEPADA MASYARAKAT

UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA

Pendahuluan

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

Ada beberapa alasan kenapa menggunakan flutter dalam pengembangan aplikasi mobile pada modul ini.

1. Flutter menyediakan fitur hot reload

   Fitur ini menjadikan pemrograman mobile serasa pengembangan web karena setiap ada perubahan tidak perlu melakukan kompilasi yang berulang-ulang untuk melihat hasilnya.

   Sedangkan pada pemrograman mobile pada Android Studio khususnya yang native (Java), harus melakukan build APK disetiap debug dan melihat hasil pada emulator.

   Proses build atau kompilasi yang berulang inilah yang kadang membutuhkan waktu yang lama, apalagi spek komputer yang digunakan tidak sesuai dengan kebutuhan minimum untuk mengembangkan aplikasi mobile.

2. Flutter menggunakan bahasa pemrograman Dart, sedangkan Android Studio menggunakan java dan Kotlin
3. Aplikasi yang dihasilkan menggunakan Flutter dapat di kompilasi/build ke Android dan iOS, sedangkan Android Studio hanya untuk Android saja.

Contoh aplikasi yang menggunakan Flutter

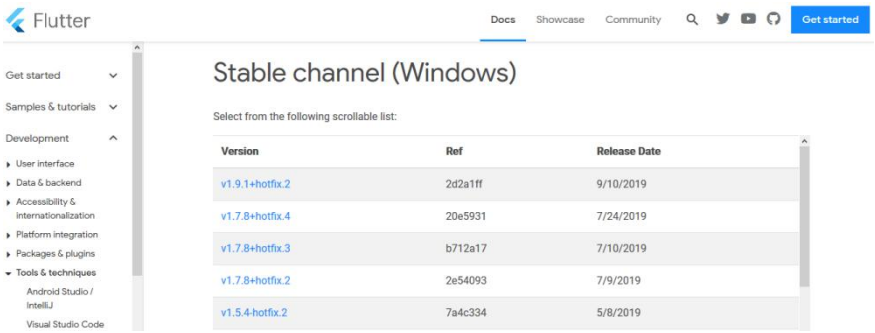
1. Alibaba (Android);
2. Google AdWords (Android);
3. App Tree (Android);
4. Topline (Android);
5. Hamilton (Android dan iOS);

Modul 1 : Android Studio dan Flutter

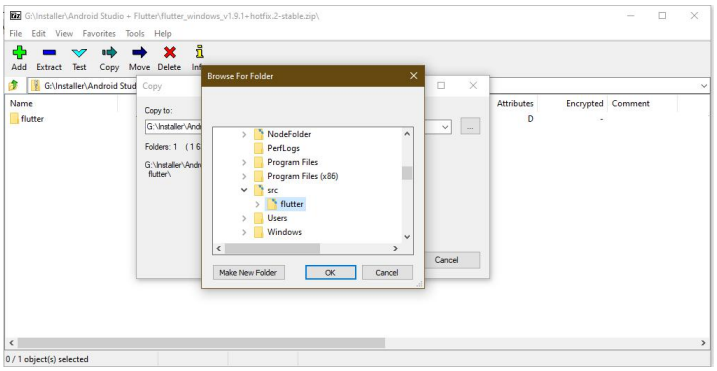
Persiapan dalam belajar pemrograman mobile menggunakan Flutter.

1. Unduh Flutter SDK dengan alamat berikut

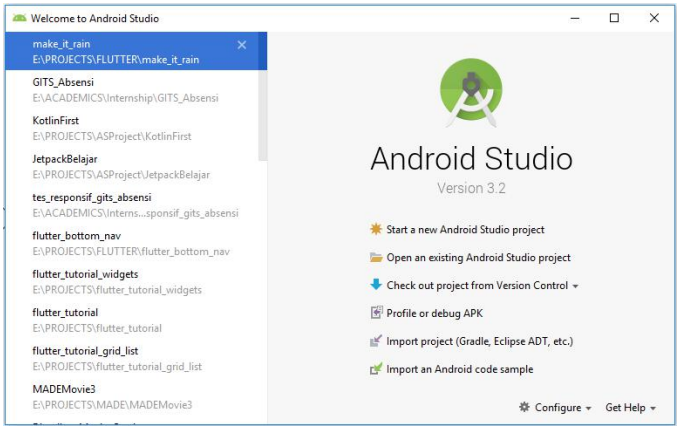
   https://flutter.io/sdk-archive/



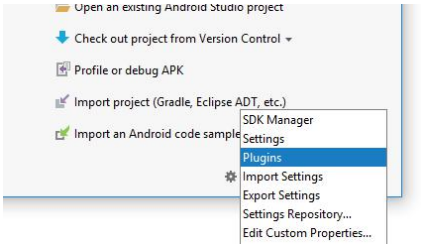
2. Ekstrak file unduhan ZIP/RAR Flutter SDK ke dalam folder yang diinginkan, penempatan folder sebisa mungkin yang mudah dijangkau. Misal di drive C:/src



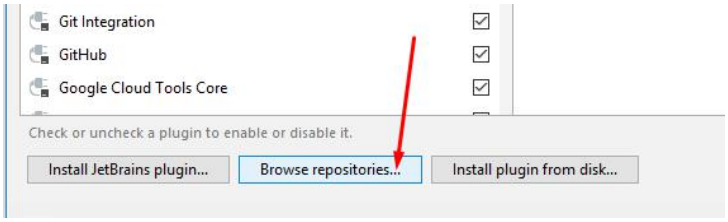
3. Buka Android Studio yang sudah terinstal, sampai muncul seperti berikut



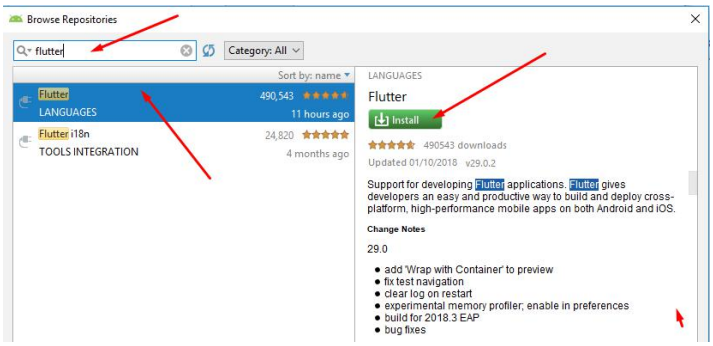
4. Buka *Configure -> Plugins*



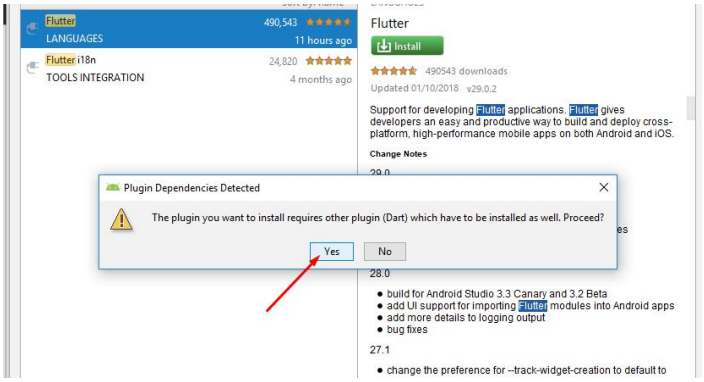
5. Setelah muncul jendela *Plugins*, maka pilihlah *"Browse Repositories..."*



6. Kemudian akan muncul jendela *"Browse Repositories"*. Kemudian lakukan pencarian dengan kata kunci: *"flutter"*.

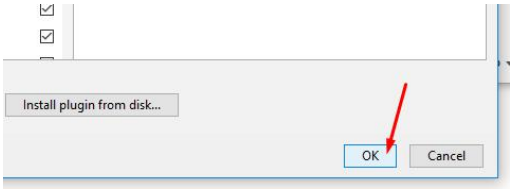


7. Jika anda mendapat *notification* berikut ini, pilih saja "Yes"

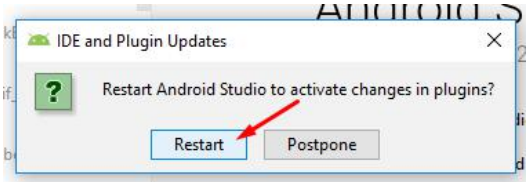


8. Tunggu proses pemasangan *plugin* hingga selesai.
9. Klik *Close*

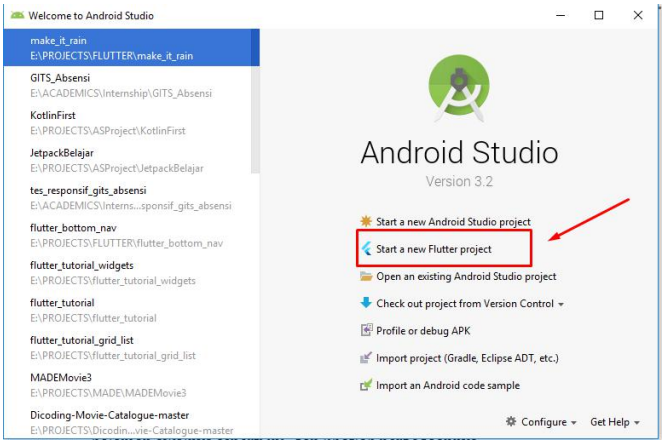
10. Klik *OK*



11. Kemudian anda akan mendapat notifikasi untuk mulai ulang *Android Studio*. Pilihlah tombol *"Restart"*.



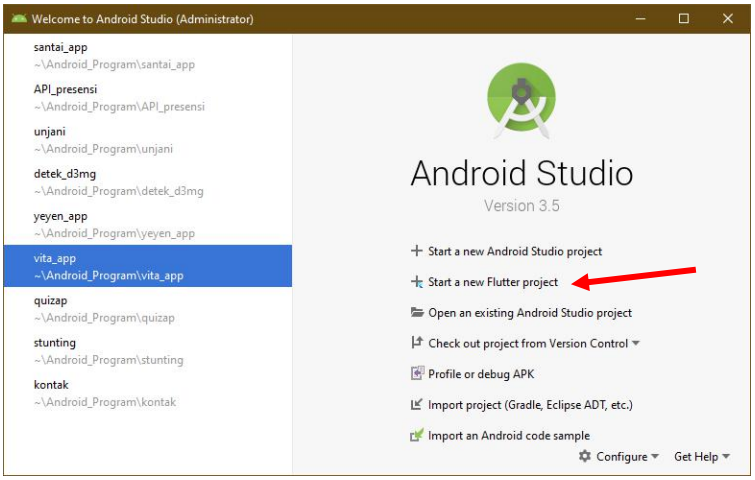
12. Tunggu proses mulai ulang *Android Studio,* hingga muncul kembali halaman awalnya seperti ini, dan lihatlah perbedaannya.



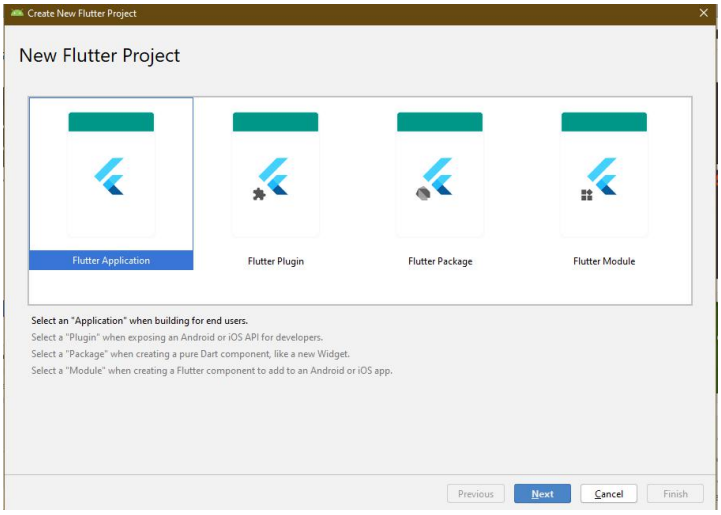
Modul 2: Flutter Fundamental

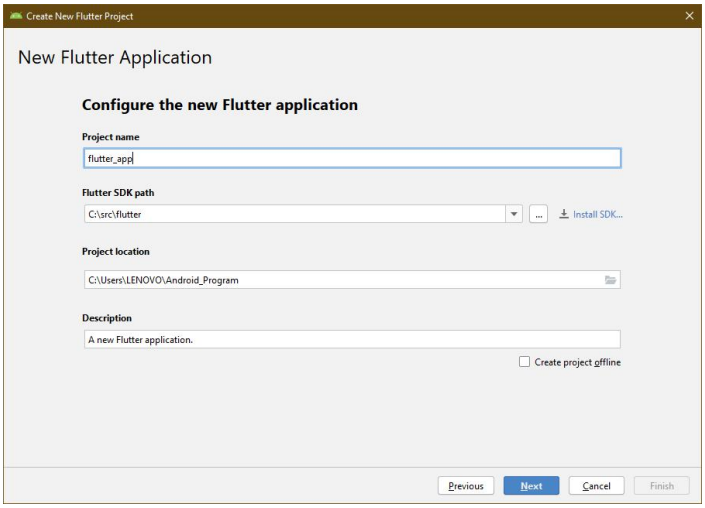
Membuat Project Flutter di Android Studio

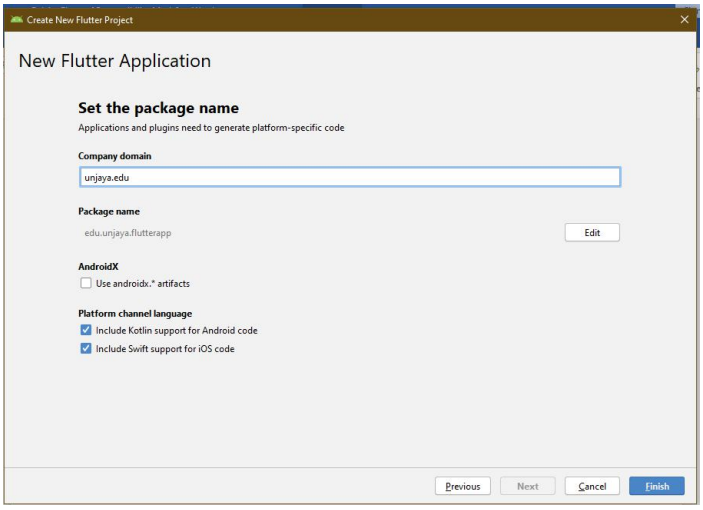
1. Buka Android Studio, kemudian pilih *Start a new Flutter project*



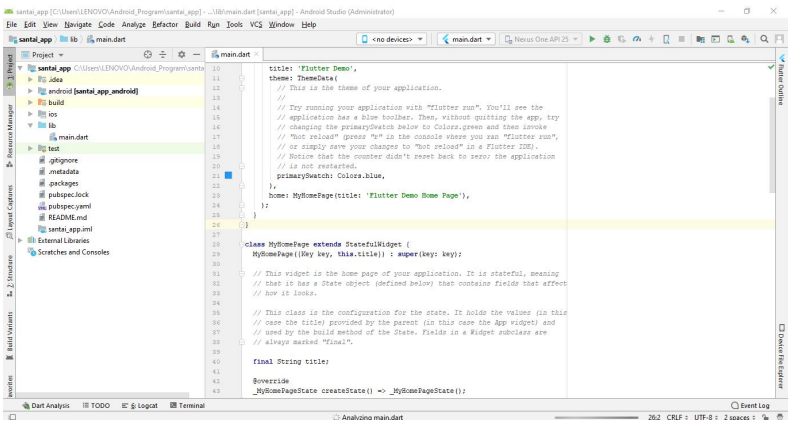
2. Muncul tampilan wizard untuk membuat Flutter project







3. Flutter Project siap di *build* atau *run*

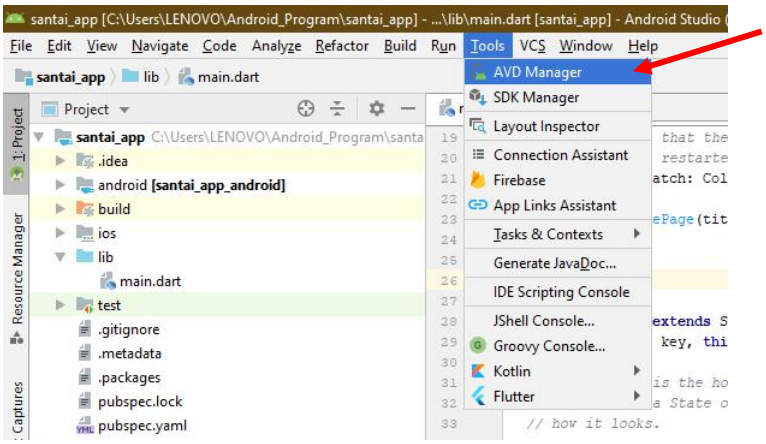


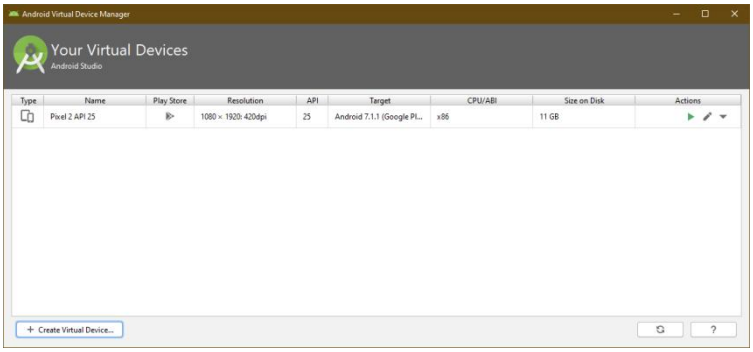
4. Build dan Run Flutter Project

   Cara menjalankan dan build project Flutter di Android Studio sama seperti project Android biasa.

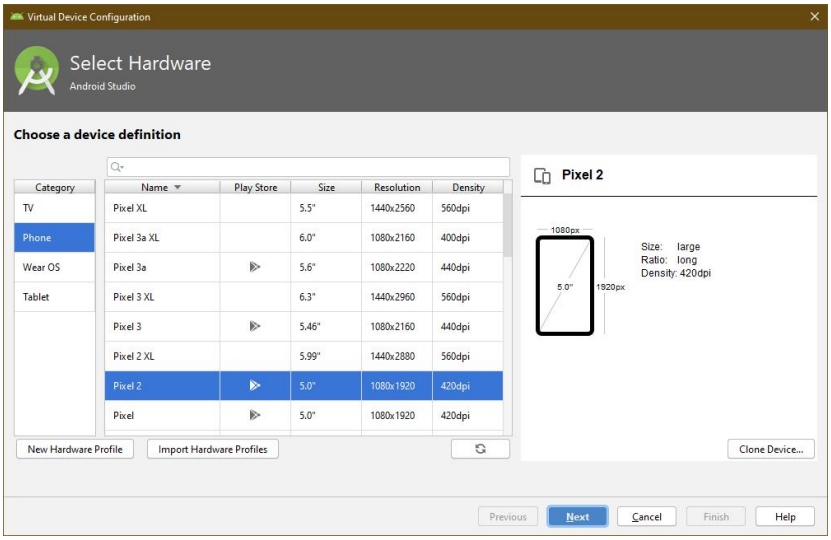
   Kita tinggal klik tombol **Run**…

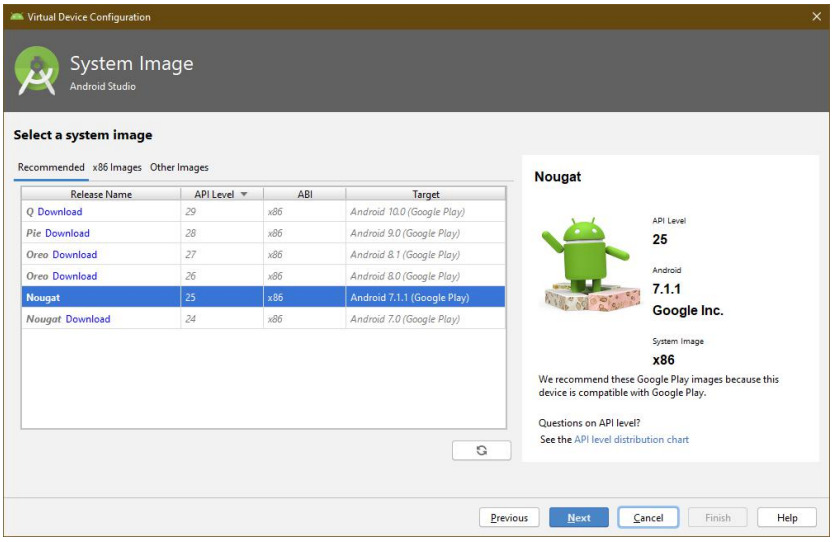
5. Emulator yang digunakan harus dipastikan sudah siap, jalan. Kalau belum arahkan kursor ke menu Tool → AVD Manager

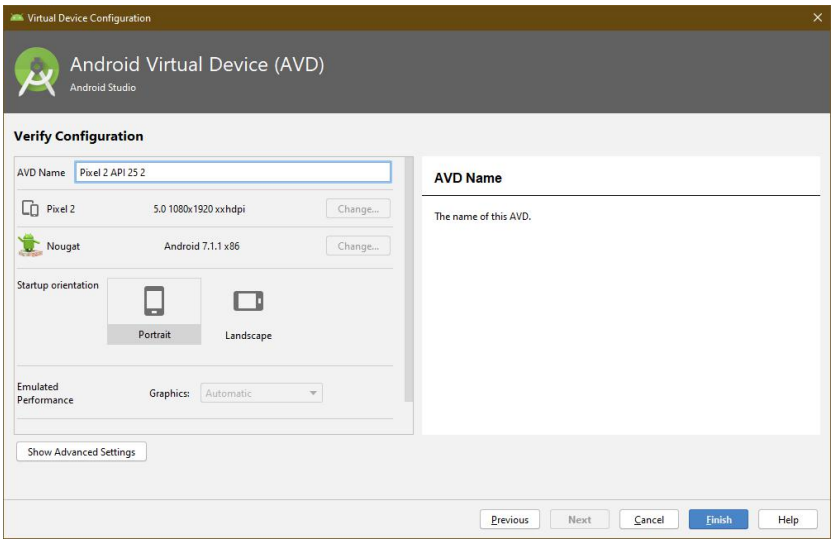




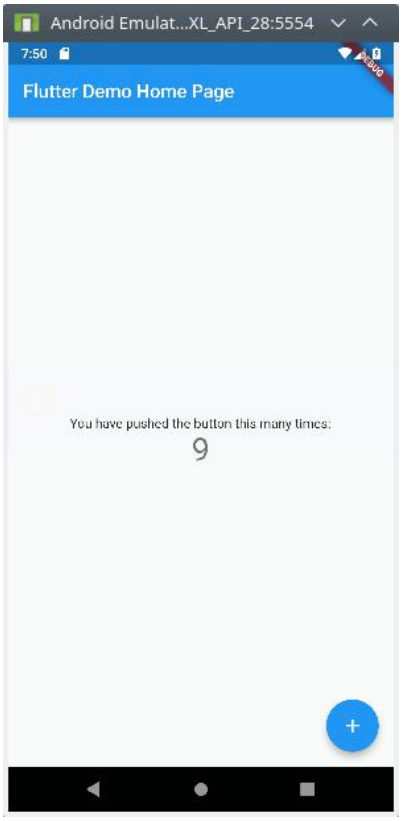
6. Daftar Virtual Device, jika tidak ditemukan maka lanjutkan dengan menekan tombol Create Virtual Device



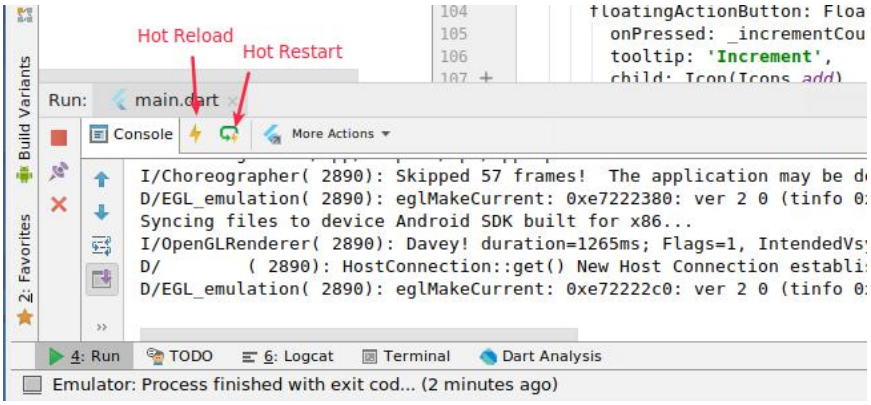




7. Setelah Flutter Project Running, berikut hasil nya

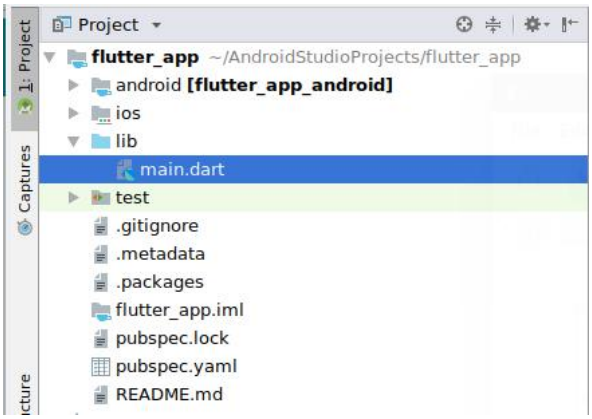


8. Fitur hot reload dapat langsung digunakan dengan menekan tombol yang ditunjukkan panah di bawah atau cukup melakukan penyimpanan setelah perubahan pada script, secara otomatis Android Studio melakukan hot reload.



Modul 3: Flutter Fundamental

Memahami Struktur Direktori Project Flutter

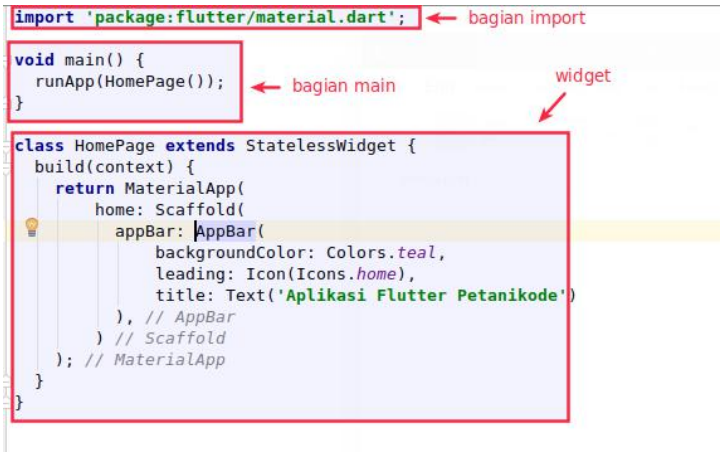


Penjelasan struktur direktori :

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

Struktur Kode Aplikasi

Kode program awal yang didapatkan berada pada file main.dart



1. **Bagian Import**

   Bagian import adalah tempat kita mendeklarasikan atau mengimpor library yang dibutuhkan pada aplikasi.

2. **Bagian Main**

   Bagian main adalah fungsi utama dari aplikasi yang akan menjadi *entri point*. Fungsi ini akan dieksekusi pertama kali saat aplikasi dibuka.

3. **Bagian Widget**

   Bagian widget adalah tempat kita membuat widget. Aplikasi Flutter sebenarnya terdiri dari susunan widget. Widget bisa disebut juga elemen-elemen seperti Tombol, Teks, Layout, Image, dan sebagainya.

Modul 4: Widget dalam Flutter

Prinsip dasar pada Flutter adalah semuanya berupa widget. Terdapat dua jenis widget yaitu sebagai berikut :

1. Stateless widget: class widget yang propertinya immutable, artinya nilainya tidak bisa diubah, keadaannya tidak berubah dari waktu ke waktu.
2. Stateful widget: dimana keadaannya bisa berubah dari waktu ke waktu.

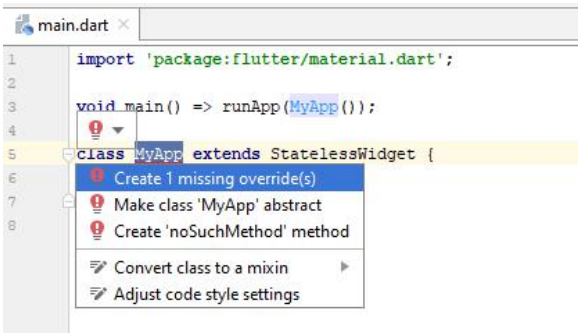
1. Membuat StateLessWidget

Cara membuat StatelessWidget adalah dengan membuat class turunan (extends) dari class StatelessWidget.

```
class MyApp extends StatelessWidget {}
```

Baris import yang pertama untuk memasukkan package utama Flutter yang berisi widget-widget standard (tombol, text, image, list, dll) yang paling sering dipakai. Sedangkan baris void main yang kedua untuk memanggil class utama dari aplikasi kita yang bernama MyApp. Perhatikan ada garis merah di bawahnya mendandakan class tersebut belum ada. Nah sekarang mari kita buat class MyApp dengan mengetik kode di bawah:

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

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

class MyApp extends StatelessWidget {

}
```



Kita membuat class MyApp yang merupakan keturunan dari StatelessWidget (yang berisi komponen UI yang tetap). Perhatikan ada garis merah di bawah nama class MyApp. Ini terjadi karena class tersebut belum menyertakan metode wajib bawaannya. Anda tidak perlu repot-repot mengetiknya, cukup arahkan kursor di posisi garis merah tadi, lalu tekan Alt-Enter. Maka akan muncul menu pilihan seperti di atas. Silahkan klik pilihan paling atas: “Create 1 missing override(s)”.

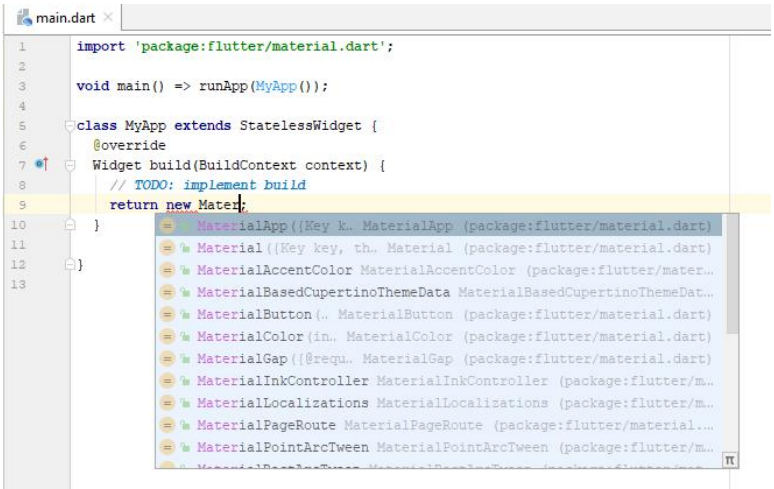
Hasilnya seperti gambar di bawah:

Jangan lupa selalu pakai fitur auto complete ini, penting bagi programmer. Jadi kita tidak perlu capek-capek ketik kode dan menghapal syntax-nya. Auto complete ini juga meminimalisir kesalahan penulisan kode.

Syntax override maksudnya dia menimpa (replace) fungsi “Widget build” bawaan class parent-nya (StatelessWidget). Perhatikan setiap class turunan dari StatelessWidget dan StatefulWidget harus menyertakan fungsi “Widget build” sebelumnya.

Yang namanya fungsi lazimnya ada output-nya. Demikian pula dengan fungsi “Widget build” ini juga ada keluaran atau output-nya. Apa yang jadi output-nya didefinisikan pada syntax “return”. Perhatikan di atas tertulis “return null” menandakan dia belum mengeluarkan output apa pun. Jika kita ingin fungsi tersebut mengeluarkan tulisan “Hello World” maka return-nya harus kita ubah sebagai berikut (jangan lupa hapus TODO dan null ya).

Pertama. Ganti “null” dengan “MaterialApp”, tapi titik koma setelah “null” jangan dihapus. Untuk diperhatikan, sebisa mungkin tidak ketik sendiri, cukup panggil dari auto complete, lalu pilih yang paling atas:

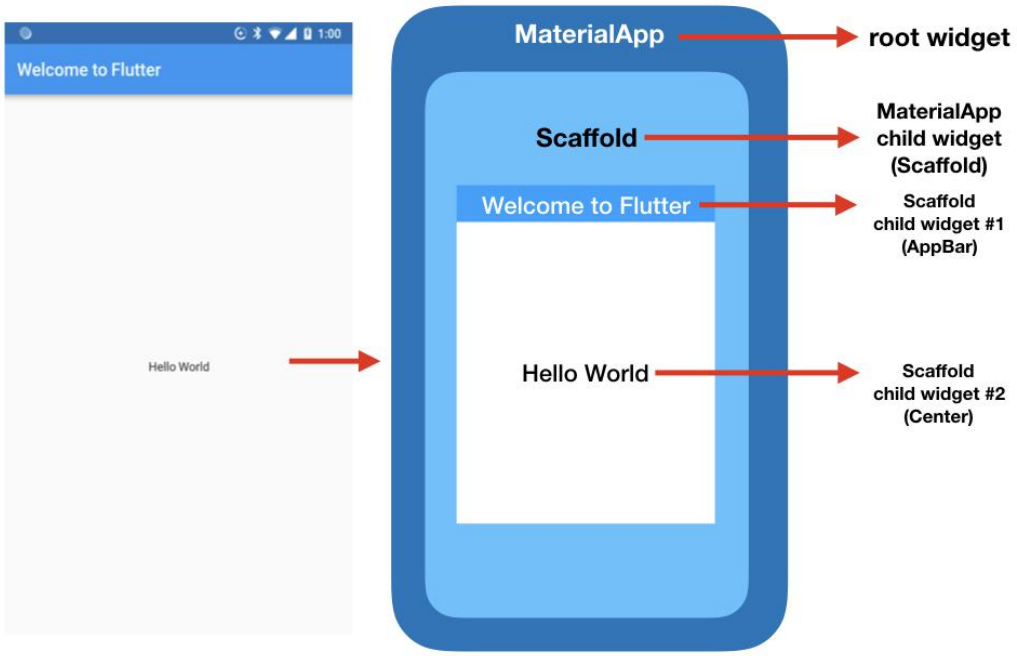


main.dart

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new MaterialApp();
  }

}

```

MaterialApp adalah widget yang menjadi puncak tertinggi (istilahnya root widget) dari aplikasi kita. Setiap aplikasi pasti ada root widget yang berfungsi sebagai container (penampung) widget-widget di bawahnya.

MaterialApp() adalah root widget. Artinya di dalamnya akan ada widget lagi, yaitu Scaffold(). Di dalam widget Scaffold ini akan ada widget lagi,yaitu AppBar() dan Center(). Skemanya kira-kira seperti ini:



Kode program seluruhnya

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new MaterialApp(
      title: "Welcome to Flutter",
      home: new Scaffold(
        appBar: new AppBar(
          title: const Text("Welcome to Flutter"),
        ), // AppBar
        body: const Center(
          child: const Text("Hello World"),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }

}
```

2. Membuat Stateful Widget

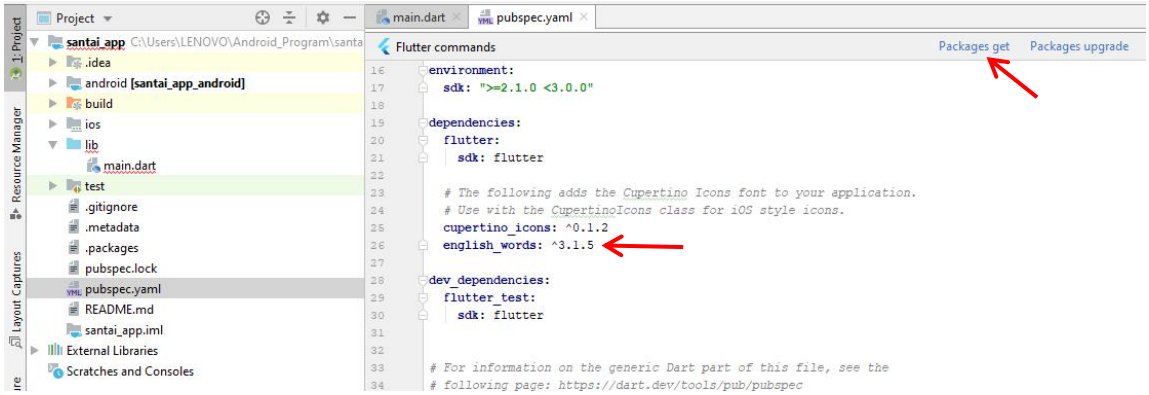
Adapun langkah-langkah membuat stateful widget adalah sebagai berikut:

a. Membuat class Stateful Widget baru (misal: RandomWords) yang merupakan keturunan (extends) dari class StatefulWidget.
b. Membuat class state baru (misal: RandomWordsState) yang merupakan keturunan (extends) dari class State<Nama Class Point satu di atas> atau State<RandomWords>
c. Isi class stateful widget pada point satu (RandomWords) isinya adalah instansiasi dari class pada point dua (RandomWordsState)
d. Isi class state pada point dua (RandomWordsState) isinya adalah alur logika yang akan dijalankan ketika class stateful widget pada point satu (RandomWords) dipanggil.

Menambah Paket Eksternal

Paket adalah sekumpulan modul dan fungsi yang bisa menambah kemampuan aplikasi kita. Dalam project ini kita akan menambah paket “english_words”. Salah satu fiturnya adalah bisa men-generate random words berbahasa Inggris.

Expand project nama_startup dan double click pubspec.yaml, lalu tambahkan english_words: ^3.1.5 di bawah dependencies:



Kemudian klik tombol Packages get, agar paket yang baru ditambahkan terinstal ke Android Studio dan bisa dipanggil kapan saja.

Berikutnya kita buat sebuah contoh Stateful Widget dengan mengambil paket eksternal, yaitu english_word versi 3.1.5.

Langkah pertama adalah dengan melakukan pemanggilan paket english_word menggunakan import

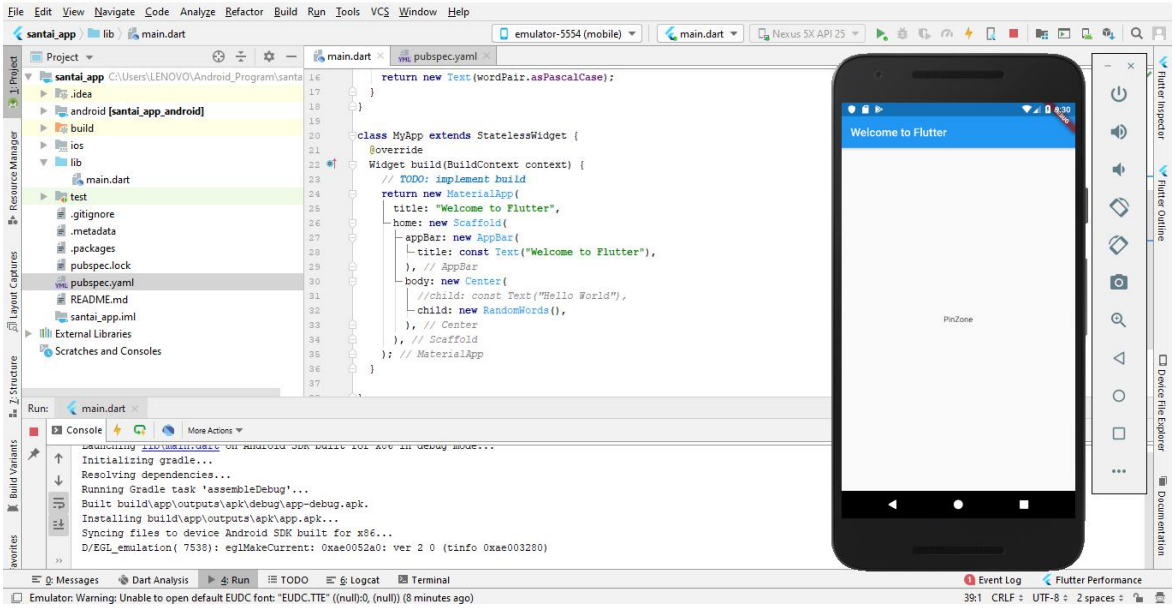
```
import 'package:flutter/material.dart';
import 'package:english_words/english_words.dart';
```

Berikutnya membuat class baru untuk menampilkan teks secara random

```
class RandomWords extends StatefulWidget {
  @override
  RandomWordsState createState() => RandomWordsState();
}

class RandomWordsState extends State<RandomWords> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    final WordPair wordPair = new WordPair.random();
    return new Text(wordPair.asPascalCase);
  }
}
```

Kemudian di dalam class MyApp dilakukan pemanggilan class RandomWords

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return new MaterialApp(
      title: "Welcome to Flutter",
      home: new Scaffold(
        appBar: new AppBar(
          title: const Text("Welcome to Flutter"),
        ), // AppBar
        body: new Center(
          child: new RandomWords(), // Pemanggilan class RandomWords
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }

}
```



Modul 5: Widget Layout

Mengatur tata letak atau layout pada User Interface sangatlah penting untuk kenyamanan pengguna saat menggunakan aplikasi yang kita buat, Mempunyai user interface yang bagus dan rapi pastinya menjadi kebanggaan tersendiri untuk developernya dan juga penggunanya, salah satu widget yang paling penting dalam flutter untuk menyusun user interface, yaitu ***Row*** dan ***Column***.

Row dan Column mempunyai peranan seperti LinearLayout, kalo di Android Studio atau aplikasi semacamnya. Row digunakan untuk menyusun widget secara Horizontal, sedangkan Column digunakan untuk menyusun widget secara Vertical.

1. Widget Row

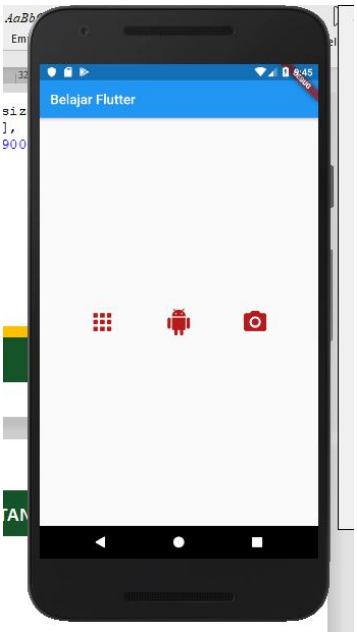
   a. Buka Aplikasi Visual Studio Code kalian, buat project baru, berinama project tersebut, lalu simpan pada lokasi yang diinginkan.
   b. Setiap widget yang kita buat didalam Row, akan tersusun secara Horizontal, untuk itu kita menggunakan atribut children, yaitu atribut turunan dari widget Row, berbeda dengan atribut child yang hanya dapat membuat satu widget saja, atribut children dapat membuat beberapa widget. Contohnya seperti berikut ini.

```
//Import Package yang diperlukan
import 'package:flutter/material.dart';

//Method utama untuk menjalankan program
void main() => runApp(new MainActivity());

//Class utama
class MainActivity extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Belajar Flutter',
      home: new Scaffold(
        //Membuat Widget AppBar
        appBar: new AppBar(
          //Menambahkan TitleBar
          title: new Text('Belajar Flutter'),
        ),

        body: new Center(
          //Menambahkan widget Row sebagai anak dari widget Center
          child: new Row(
            //Digunakan agar widget mengisi ruang kosong pada layar
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: <Widget>[
              new Icon(Icons.apps, color: Colors.red[900], size: 40.0),
              new Icon(Icons.android, color: Colors.red[900], size: 40.0),
              new Icon(Icons.camera_alt, color: Colors.red[900], size: 40.0),
            ],
          ),
        ),
      ),
    );
  }
}
```

Pada contoh program yang kita buat, pertama kita menempatkan Row kedalam widget Center, supaya tampil ditengah layar, selanjutnya kita tambahkan atribut ***mainAxisAlignment*** lalu kita set menjadi ***speceEvenly***, agar setiap widget didalam Row tampil tanpa menghamburkan ruang kosong pada layar, artinya jarak antara widget satu dengan widget lainnya akan tersusun dengan rapi, menyesuaikan ukuran layar.

Didalam Row kita coba menambahkan 3 buah widget Icon, lalu kita ubah artibutnya seperti warna menjadi merah, dan ukurannya kita set menjadi 40.0.

Selain speceEvenly terdapat beberapa atribut lainnya yang dapat kalian coba, seperti speceBeetwen, speceAround, dll.



2. Widget Column

Widget Column digunakan untuk menyusun widget turunannya secara Vertical, pada contoh berikut ini misalnya kita membuat beberapa widget Text kedalam Column.

Pada contoh di atas, pada bagian body, cukup mengganti baris `child: new Row`

Menjadi `child: new Column`

```
body: new Center(
  //Menambahkan widget Row sebagai anak dari widget Center
  child: new Column(
    //Digunakan agar widget mengisi ruang kosong pada layar
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: <Widget>[
      new Icon(Icons.apps, color: Colors.red[900], size: 40.0),
      new Icon(Icons.android, color: Colors.red[900], size: 40.0),
      new Icon(Icons.camera_alt, color: Colors.red[900], size: 40.0),
    ],
  ),
),
```

Sehingga output yang dihasilkan seperti berikut :

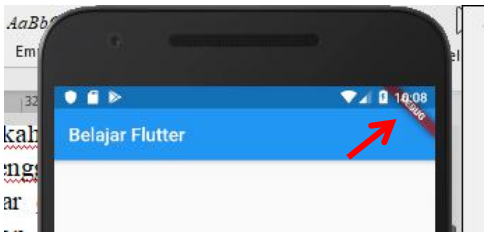
Modul 6: Persiapan Rilis Aplikasi Android

Persiapan rilis aplikasi android merupakan sebuah hal yang wajib diketahui oleh seorang pengembang aplikasi android, karena ini adalah langkah yang paling akhir agar aplikasi yang dikembangkan dapat digunakan oleh pengguna yaitu dengan melakukan instalasi aplikasi pada smartphone pengguna. Agar dapat diinstal pada smartphone android khususnya, harus tercipta file Application Package File, atau sering dikenal file APK.

Secara default ketika kita melakukan running aplikasi pada Android Studio akan tercipta sebuah file bernama app.apk yang terletak di folder berikut

```
Nama_folder_user\Android_Program\nama_aplikasi\build\app\outputs\apk
```

File APK yang tercipta saat running/debuging terus menampilkan tulisan DEBUG di sebelah kanan atas, seperti berikut



Agar tulisan DEBUG hilang maka, kita harus mempersiapkan beberapa langkah, yaitu

1. Android Manifest

- Buka android/app/src/AndroidManifest.xml
- Ubah nama package, android:label

2. Buat Launcher

- pada folder `<app dir>/android/app/src/main/res/` , tempatkan file ikon aplikasi. Secara default ditempatkan pada folder `mipmap-` .

- pada file `AndroidManifest.xml`, ubah referensi ikon pada atribut `android:icon` `<application android:icon="@mipmap/ic_launcher" ...`).

3. Buat Keystore

Di dalam cmd folder project tulis perintah :

```
keytool -genkey -v -keystore keystore.jks -keyalg RSA
-keysize 2048 -validity 10000 -alias key
```

4. Reference Keystore

- Buat file android/key.proprerties :

```
storePassword=<password from previous step>
keyPassword=<password from previous step>
keyAlias=key
storeFile=<location of the key store file, e.g. /Users/<user
name>/key.jks>
```

5. Ubah Gradle

- Buka android/app/build.gradle
- Ubah defaultConfig:

```
defaultConfig {
   applicationId "id.athalon.hello" //nama package
   minSdkVersion 16
   targetSdkVersion 27
   versionCode 1 //version code
   versionName "1.0" //version code
   testInstrumentationRunner       "android.support.test.runne
r.AndroidJUnitRunner" }
```

- Ubah bagian ini :

```
android {
```

Menjadi :

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
   keystoreProperties.load(new
FileInputStream(keystorePropertiesFile))
}
android {
```

- Ubah bagian ini :

```
buildTypes {
   release {
      signingConfig signingConfigs.debug
   }
}
```

Menjadi :

```
signingConfigs {
   release {
      keyAlias keystoreProperties['keyAlias']
      keyPassword keystoreProperties['keyPassword']
      storeFile file(keystoreProperties['storeFile'])
      storePassword keystoreProperties['storePassword']
   }
}

buildTypes {
   release {
      signingConfig signingConfigs.release
   }
}
```

6. Jalankan di cmd project

- flutter clean
- flutter build apk