LAPORAN PRAKTIKUM

NIM : 2003073

Nama : Ica Natasya

Kelas : D3TI.2C

Mata Kuliah : **Pemrograman Perangkat Bergerak (TIU3403)**

Praktikum ke / Judul : 6 / Framework Flutter

Tanggal Praktikum : 21 Maret 2022

Dosen Pengampu : Fachrul Pralienka Bani Muhamad, S.ST., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
POLITEKNIK NEGERI INDRAMAYU

2022

PRAKTIKUM 6
FRAMEWORK FLUTTER

A. TUJUAN PRAKTIKUM

Tujuan Umum

Mahasiswa membuat tampilan antarmuka pengguna pada aplikasi perangkat bergerak menggunakan framework flutter

Tujuan Khusus

Mahasiswa dapat

1. Menjelaskan konsep dasar framework flutter
2. Menjelaskan keunggulan penggunaan framework flutter
3. Menggambarkan struktur project flutter
4. Menjelaskan struktur kode tampilan antarmuka framework flutter
5. Mengimplementasikan framework flutter pada pembuatan antarmuka pengguna pada perangkat bergerak

B. TEORI SINGKAT

Pengenalan framework flutter

Flutter adalah kit pengembangan perangkat lunak (SDK) (Flutter) bersumber terbuka untuk dikembangkan aplikasi lintas platform dan gratis untuk digunakan. Ini SDK Google untuk kerajinan aplikasi yang indah, cepat, dan dikompilasi secara asli untuk seluler, web, dan desktop dari satu basis kode.

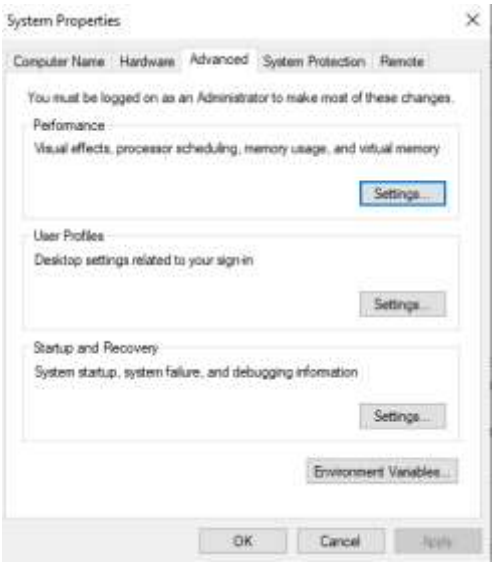
Aplikasi Flutter ditulis dalam bahasa Dart (Anak panah). Dart dioptimalkan untuk membangun antarmuka pengguna khusus (UI) dan cepat aplikasi lintas platform. Kode Dart dikompilasi ke kode mesin asli untuk aplikasi platform sedang berjalan. Misalnya, kode Dart web dikompilasi ke JavaScript.

Flutter adalah framework lengkap yang menyediakan rendering & widget UI, solusi manajemen status, navigasi, pengujian, dan aplikasi perangkat keras antarmuka pemrograman (API) untuk berinteraksi dengan fitur tingkat perangkat seperti sensor, Bluetooth, dll.

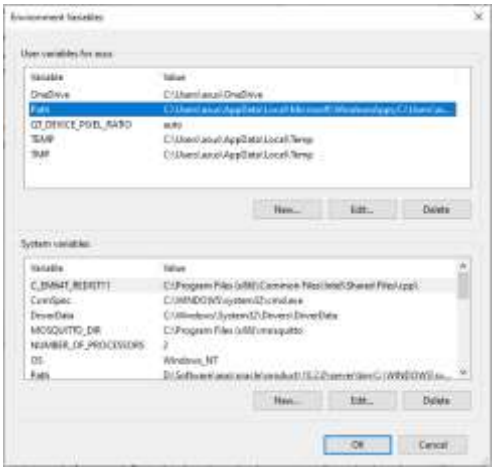
Pengaturan lingkungan pengembangan

Aplikasi Flutter dapat dibuat di macOS, Windows, Linux, dan Chrome OS.

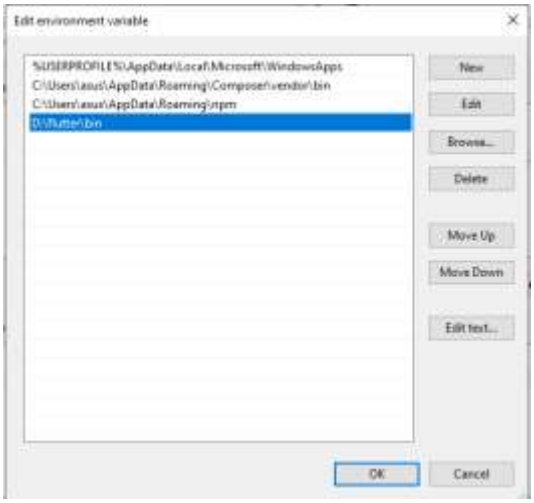
1. Ketik env pada pencarian di bagian bawah desktop, kemudian pilih Advanced.



2. Edit path pada User variables for asus.

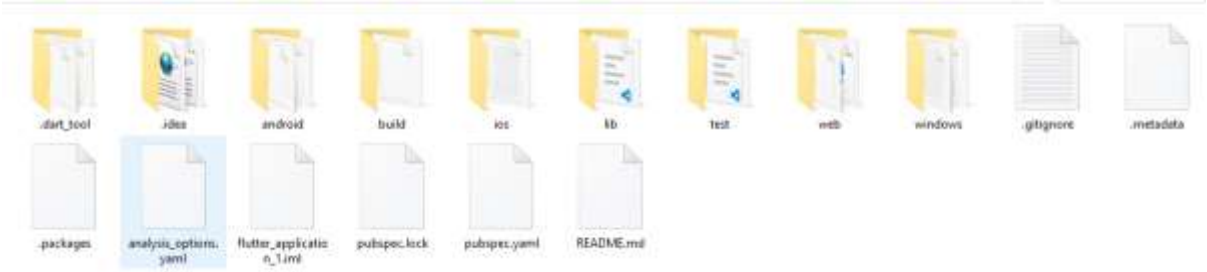


3. Kemudian klik New dan paste kan alamat penyimpanan flutter sampai folder bin.



4. Ketik D:\{YOUR_USER_NAME}\flutter\bin, lalu pilih OK.
5. Restart system.
6. Ketikkan flutter di baris perintah. Anda akan melihat pesan dengan beberapa Flutter instruksi antarmuka baris perintah (CLI). Flutter mungkin mengunduh secara opsional lebih banyak alat khusus Windows saat ini.

Struktur project framework flutter



- android/ and ios/: Folder-folder ini berisi kode khusus platform untuk setiap OS dan secara otomatis dikelola oleh IDE dan compiler. Strukturnya persis sama dengan yang Anda dapatkan dengan proyek Android baru di Android Studio atau proyek iOS di XCode.
- lib/: Folder ini penting: folder ini berisi kode sumber Dart dari aplikasi Flutter Anda.
- test/: Tes unit, tes widget, dan tes integrasi semuanya ada di folder ini.
- pubspec.yaml: File ini sangat penting karena mendefinisikan paket Dart dan mencantumkan dependensi aplikasi Flutter Anda.
- README.md: : Ini adalah file penurunan harga biasa yang dapat Anda temukan di repositori git mana pun. Ini digunakan untuk repositori git Anda dan pada saat yang sama sebagai "halaman beranda" di https://pub.dev jika Anda ingin menerbitkan sebuah paket.

Struktur kode program antarmuka pengguna pada framework flutter

Struktur kode antarmuka pengguna terdiri dari :

Class yang merupakan child dari StatefullWidget ataupun StatelessWidgets yang mempunyai method build yang mengembalikan class MaterialApp yang dialamnya terdapat property home diisi dengan class Scaffold ,pada class Scaffold terdapat property body yang dapat diisi dengan Widgets lainnya.

Widget dan state

Di bagian selanjutnya kita akan melihat bahwa kelas di Flutter menjadi widget ketika mewarisi dari StatelessWidget atau StatefulWidget.

StatelessWidget (kelas StatelessWidget) tidak dapat diubah. Setelah dibuat, itu tidak dapat diubah. Itu tidak mempertahankan statusnya. Widget tanpa kewarganegaraan dibuat hanya sekali, dan tidak dapat dibangun kembali di lain waktu. 'build()' widget stateless metode dipanggil hanya sekali.

- Widget tanpa kewarganegaraan. Gunakan widget semacam ini saat Anda perlu membuat UI yang tidak akan berubah seiring waktu. Ini adalah blok "mandiri" yang tidak bergantung pada peristiwa atau sumber eksternal; itu hanya bergantung pada konstruktor dan data internalnya.

StatefulWidget (StatefulWidget) bisa berubah. Itu melacak negara.Ini membangun kembali beberapa kali selama masa pakainya. Metode `build()` widget Stateful adalah dipanggil berkali-kali.

- Widget stateful. Gunakan widget semacam ini saat Anda perlu membuat UI yang akan berubah seiring waktu. Dalam hal ini UI akan berubah secara dinamis karena peristiwa eksternal seperti respons yang diterima dari permintaan HTTP atau panggilan balik yang dipicu dengan menekan tombol.

STATE

Komunitas Flutter telah berjuang dengan masalah ini. Karena ini, mereka telah menemukan beberapa solusi untuk menangani manajemen negara. Semua solusi ini memiliki satu aspek yang mereka bagikan: pemisahan model dan tampilan. Kami akan mengimplementasikan aplikasi menggunakan 3 strategi pengelolaan negara yang berbeda:

1. Memperbarui UI menggunakan setState,
2. Melewati status di sekitar pohon widget, dengan bantuan widget Penyedia,
3. Alternatif untuk setState(), diimplementasikan dengan bantuan widget BlocBuilder.

Interaksi routes dan navigasi

Flutter memiliki mekanisme perutean imperatif, widget Navigator, dan mekanisme perutean deklaratif yang lebih idiomatis (yang mirip dengan metode build seperti yang digunakan dengan widget), widget Router.

Kedua sistem tersebut dapat digunakan bersama-sama (memang sistem deklaratif dibangun menggunakan sistem imperatif).

Aplikasi yang lebih rumit biasanya lebih baik dilayani oleh Router API, melalui konstruktor MaterialApp.router. Ini memerlukan beberapa pekerjaan awal untuk menjelaskan cara mengurai tautan dalam untuk aplikasi Anda dan cara memetakan status aplikasi ke kumpulan halaman aktif, tetapi lebih ekspresif dalam jangka panjang.

- **Dasar-dasar navigasi dan routes**

Mengatur informasi di beberapa layar adalah salah satu blok bangunan terpenting dari arsitektur apapun. Contoh yang sangat umum adalah di mana halaman pertama aplikasi Anda adalah login, jika pengguna memberikan kombinasi nama pengguna dan kata sandi yang benar, halaman selamat datang muncul.

Di dunia Flutter, halaman aplikasi Anda disebut rute, atau layar, dan mereka adalah setara dengan Aktivitas di Android atau ViewControllers di iOS.

Cara paling umum untuk membuat halaman melibatkan penggunaan widget stateless atau stateful dengan Scaffold (atau CupertinoPageScaffold). kami ingin menyarankan Anda kemungkinan struktur folder untuk aplikasi Anda:

- lib/
- rute/
- widget/
- main.dart
- route.dart

Sangat intuitif, route/ akan berisi semua widget UI yang mewakili rute aplikasi. Di dalam widget/ kami sarankan untuk meletakkan semua widget yang dapat digunakan kembali yang mendukung pembuatan halaman aplikasi Anda.

- **Membuat routes**

Kami akan membuat aplikasi sederhana dengan dua layar: HomePage, rute pertama yang muncul saat aplikasi dimulai, dan RandomPage, rute yang menampilkan nomor acak di tengah layar. Tata letak halaman akan dibuat dengan Scaffold yang nyaman dari perpustakaan bahan.

```dart
// Located in routes/home_page.dart
class HomePage extends StatelessWidget {
    const HomePage();

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: RaisedButton(
                    onPressed: () {}
                    child: const Text("Random"),
                ),
            ),
        );
    }
}

// Located in routes/random_page.dart
class RandomPage extends StatelessWidget {
    const RandomPage();

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Center(
                child: Text("${Random().nextInt(20)}"),
            ),
        );
    }
}
```

- **File main.dart**

Struktur umum file main.dart mungkin terlihat seperti berikut ini. Penggunaan material alih-alih cupertino tidak relevan, itu hanya berarti kami menggunakan MaterialApp() sebagai ganti CupertinoApp().

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

void main() => runApp(const RandomApp());

// 1.
class RandomApp extends StatelessWidget {
    const RandomApp();

    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            onGenerateTitle: (context) => "Random App",

            initialRoute: RouteGenerator.homePage,          // 2.
            onGenerateRoute:  RouteGenerator.generateRoute, // 3.

              // Hides the debug stripe on the top-right corner
              // which might be annoying to see!
              debugShowCheckedModeBanner: false,
          );
      }
  }
```

- **File routes.dart**

Memiliki satu kelas yang menangani seluruh perutean aplikasi Anda sangat bagus karena mencakup sepenuhnya prinsip tanggung jawab tunggal. Akibatnya, ketika Anda harus berurusan dengan rute, Anda akan selalu membuka route.dart karena setiap logika terkait rute hanya ada di sana. Ini adalah kontrol "terpusat" dari rute yang menjaga kode tetap bersih.
Berilah nama apa pun yang Anda inginkan tetapi kami sarankan itu harus mengandung kata "rute" sehingga Anda akan segera mengenali isinya. Contohnya adalah menggunakan bahan perpustakaan, jadi kami menggunakan pengembalian MaterialPageRoute().

```dart
// 1.
class RouteGenerator {
    // 2.
    static const String homePage = '/';
    static const String randomPage = '/random';

    // 3.
    RouteGenerator._() {}

    // 3.
    static Route<dynamic> generateRoute(RouteSettings settings) {
        //4.
        switch (settings.name) {
            case homePage:
                // .5
                return MaterialPageRoute(
                    builder: (_) => const HomePage(),
                );

            case randomPage:
                return MaterialPageRoute(
                    builder: (_) => const RandomPage(),
                );

            default:
                throw FormatException("Route not found");
        }

    }
}

// 5.
class RouteException implements Exception {
    final String message;
    const RouteException( this.message);
}
```

- **Pages diantara navigation**

Pada titik ini, kami telah mengatur rute dengan benar di dalam route.dart dan siap untuk berpindah dari satu halaman ke halaman lainnya. Rekap singkat dari pekerjaan yang telah kami lakukan sejauh ini:

- Pembuatan dua rute yang disebut DemoPage dan RandomPage;
- Pengaturan rute di MaterialApp() dari main.dart;
- Pembuatan kelas RouteGenerator yang memetakan URI ke widget sehingga Kelas Navigator dapat membuka rute.

- **Navigator**

Kelas Navigator sudah ada sejak awal Flutter dan kami menjelaskan cara menggunakannya. Tim Flutter akan memberikan pembaruan pada sistem perutean dengan:
perubahan (beberapa di antaranya rusak):

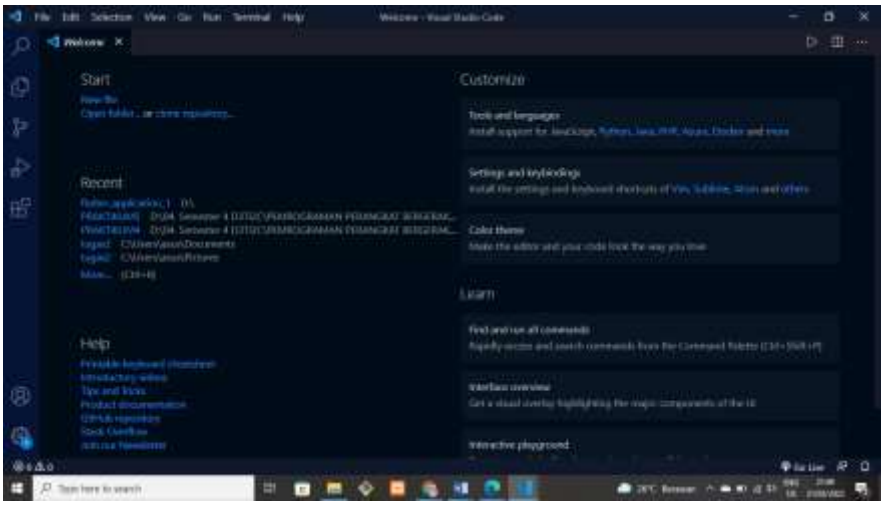
- pop() tidak mengembalikan nilai lagi;
- properti isInitialRoute dari RouteSetting tidak digunakan lagi;
- adanya widget baru bernama Router.

Router adalah widget baru yang dapat Anda gunakan untuk membuka dan menutup halaman aplikasi. Itu membungkus Navigator dan mengonfigurasi daftar halaman saat ini berdasarkan status aplikasi saat ini. Mari kita lihat apa yang akan Anda miliki yang harus dilakukan untuk memigrasi kode ke sistem

C. PELAKSANAAN PRAKTIKUM

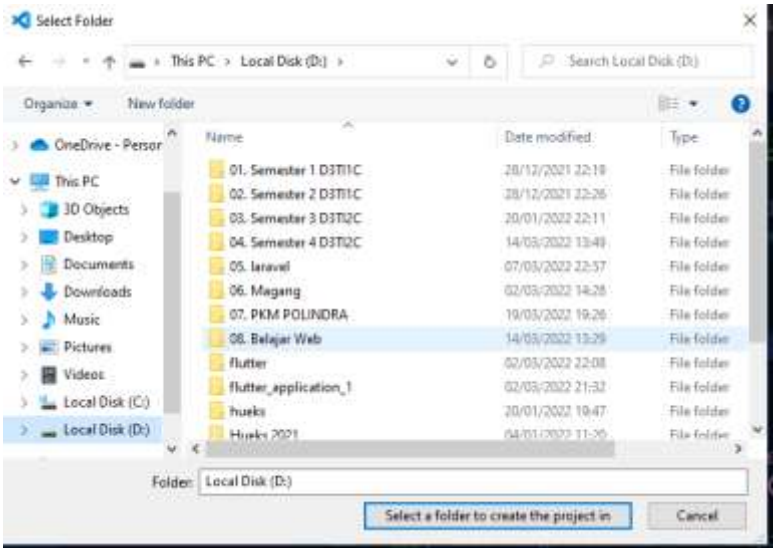
Langkah-langkah praktikum Membuat Project Flutter
https://www.youtube.com/watch?v=DzzFF-0U2Lw

1. Buka Visual Code



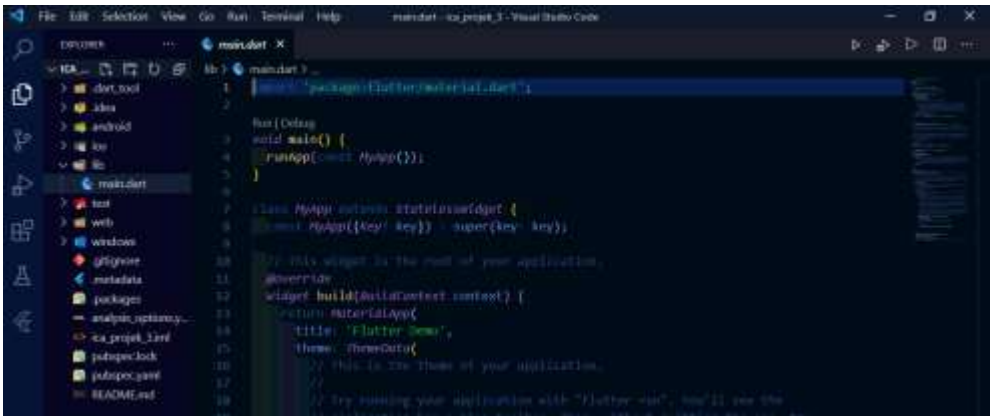
2. Masukkan perintah CTRL+SHIFT+P kemudian pilij Flutter New Project

3. Selanjutnya pilih penyimpanan project yang akan kita buat, kita simpan di D:



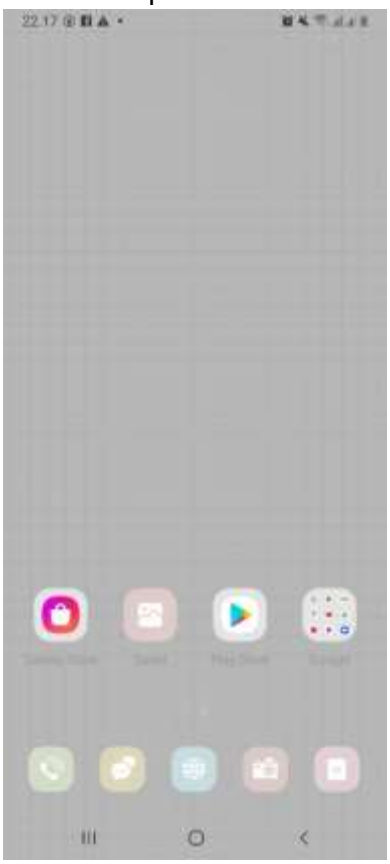
4. Kemudian rename nama project sesuai yang kita inginkan

5. Jika berhasil membuat project, maka akan muncul tampilan seperti gambar dibawah ini.

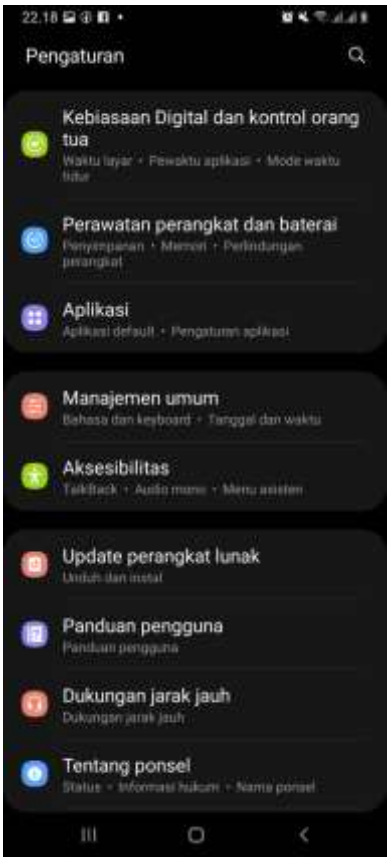


Langkah-langkah praktikum Running Flutter via Physical Device
https://www.youtube.com/watch?v=aohkII1C4JY Membuka aplikasi android studio.

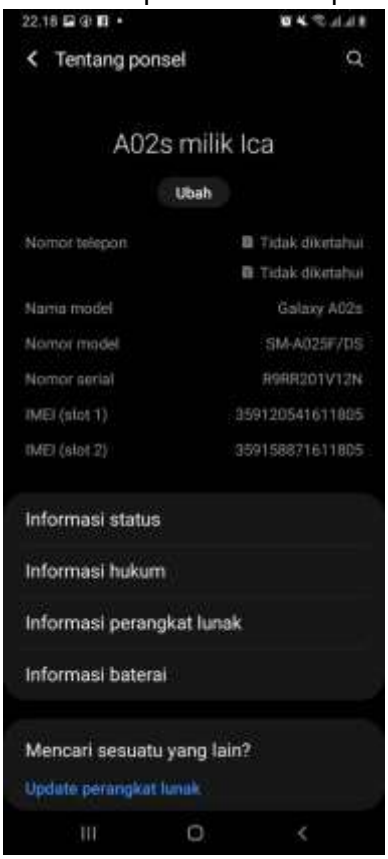
1. Buka Handphone.



2. Kemudian pilih pengaturan, Lalu pilih tentang ponsel.

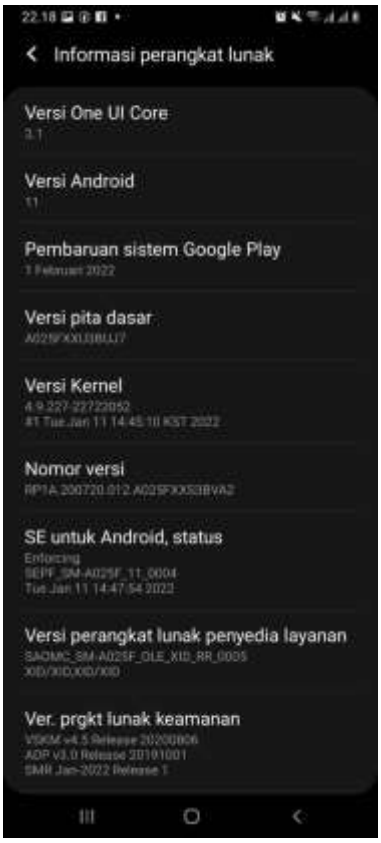


3. Kemudian pilih informasi perangkat lunak.

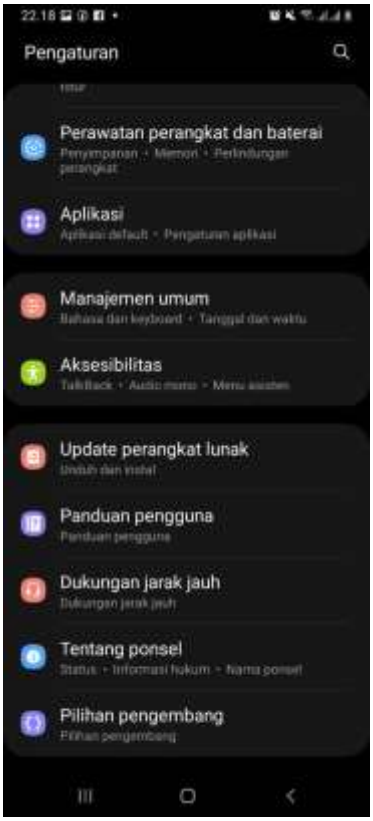


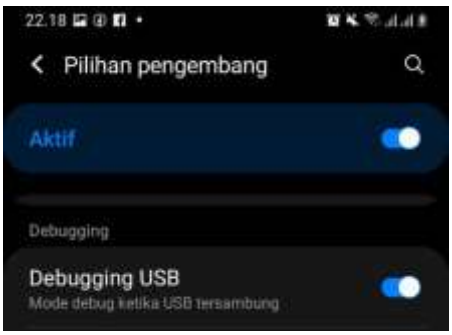
4. Kemudian setelah itu akan muncul tampilan pilihan pengembang telah aktif.

5. Setelah aktif maka akan muncul tampilan seperti digambar bawah ini.

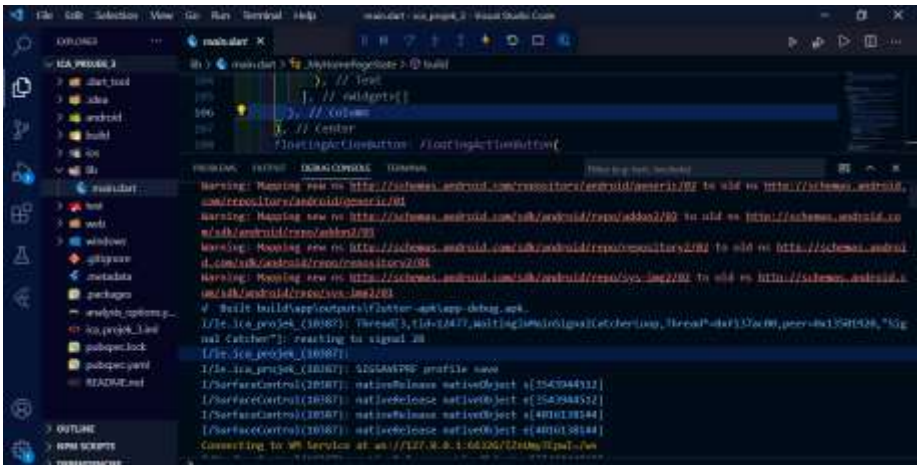


6. Setelah itu klik pilihan pengembang, lalu aktifkan Debugging USB.





7. Lalu ranning main dari proyek yang telah kita buat.



8. Setelah selesai merunning maka akan muncul nama projek yang telah kita buat di handphone kita.

9. Kemudian buka flutter yang telah kita buat, maka akan muncul seperti gambar dibawah ini.

D. Langkah dan Hasil Latihan

Diisi jika ada instruksi latihan, dan cantumkan juga hasilnya

E. Hasil dan Penjelasan Tugas

Isi/konten pada bagian ini disesuaikan dengan instruksi pada tugas

F. Kesimpulan

Berikan kesimpulan dari rangkuman teori singkat, pelaksanaan praktikum, latihan, dan tugas. Kesimpulan dapat dibuat dengan format paragraf maupun susunan list.

G. Referensi

- Alberto Miola. “Flutter Complete Reference Create Beautiful, Fast and Native Apps for Any Device”.Independently Published. 2020.
- Simone Alessandria, Brian Kayfirz. “Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart”. Packt Publishing. Birmingham - Mumbai. 2021.

- Dieter Meiller. “Modern App Development with Dart and Flutter 2: A Comprehensive Introduction to Flutter”. Walter de Gruyter GmbH. Berlin - Boston. 2021.
- Priyanka Tyagi. “Pragmatic Flutter: Building Cross-Platform Mobile Apps for Android, iOS, Web & Desktop”. CRC Press Taylor & Francis Group, LLC. London - New York. 2022.