Modul 1

Pengenalan Flutter

Instalasi, Flutter widget (statefull, stateless), scaffold, text, icon, image.

Module Overview

Mengenal flutter sebagai framework multiplatform, melakukan instalasi, memahami konsep dasar dari komponen-komponennya, dan mencoba menggunakan widget komponen material dasar flutter.

Module Objectives

Setelah mempelajari dan mempraktikkan modul ini, mahasiswa diharapkan dapat:

- Memahami flutter sebagai sebuah framework multiplatform
- Mengetahui cara setup dan instalasi framework flutter pada perangkat desktop
- Memahami struktur folder flutter
- Memahami konsep dasar dari komponen widget flutter
- Menjalankan aplikasi pertama flutter.

Pengantar Flutter

Secara umum, pengembangan aplikasi mobile dilakukan secara native yaitu menggunakan kerangka kerja (framework) berdasarkan bahasa java / kotlin untuk android dan IOS menyediakan kerangka kerja native berdasarkan bahasa Objective-C / Swift. Sehingga, untuk bisa mengembangkan aplikasi untuk ke dua platform (Android & IOS) tersebut maka perlu menulis code menjadi 2 (dua) bahasa berbeda dan menggunakan 2 (dua) kerangka kerja yang berbeda. Ini tentu akan menghabiskan banyak waktu dan membutuhkan upaya yang besar dalam proses development, maka solusi alternatif yaitu menggunakan framework yang mendukung pengembangan secara Hybrid.

Flutter menggunakan pendekatan Hybrid dengan kerangka kerja sederhana dan berkinerja tinggi berdasarkan bahasa Dart. Ini memungkinkan kinerja tinggi untuk merender UI langsung di sistem operasi (tidak melalui translasi framework), tidak seperti framework lain yang berbasis HTML dan javascript yang harus melakukan pekerjaan berat dalam mengkonversi kode ke kode asli yang berimbas ke peforma.

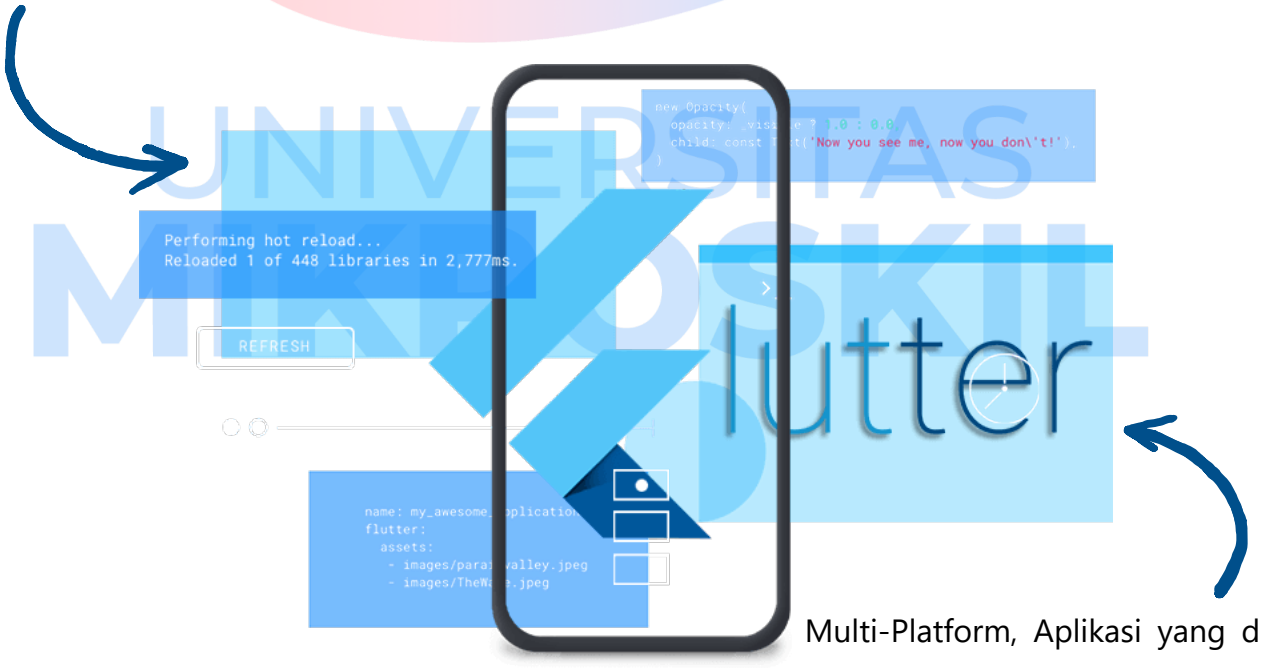
Flutter adalah framework aplikasi mobil yang diciptakan oleh Google pada tahun 2015 dan digunakan para developer untuk membuat aplikasi multiplatform hanya dengan satu basis coding (codebase).

Flutter menawarkan banyak widget (UI) atau component (umumnya menggunakan komponen google material) yang telah diuptimalkan untuk lingkungan seluler (contoh, tersedianya cupertino style untuk UI native IOS).

Fitur pada Flutter:

- Modern dan Reactive framework
- Menggunakan bahasa Dart Programming yang mudah untuk dipelajari
- Cepat dalam pengembangan aplikasi mobil
- User interface yang menarik dengan peforma tinggi (lancar)
- Memiliki banyak katalog widget
- Mendukung multi-platform dengan codebase yang sama.
- Dapat menghasilkan aplikasi dengan peforma tinggi.

Support **Hot reload**, Flutter akan otomatis melakukan injeksi pada aplikasi sehingga perubahan dapat dilihat langsung tanpa perlu compile dari awal.



Multi-Platform, Aplikasi yang dibuat pakai Flutter dapat dijalankan di Android dan iOS sekaligus.

Cari tahu:

- Apa yang dimaksud dengan reactive framework / programming?
- Bagaimana flutter bisa memiliki peforma tinggi mendekati native?
- Apa saja widget / komponen yang dimiliki oleh flutter?

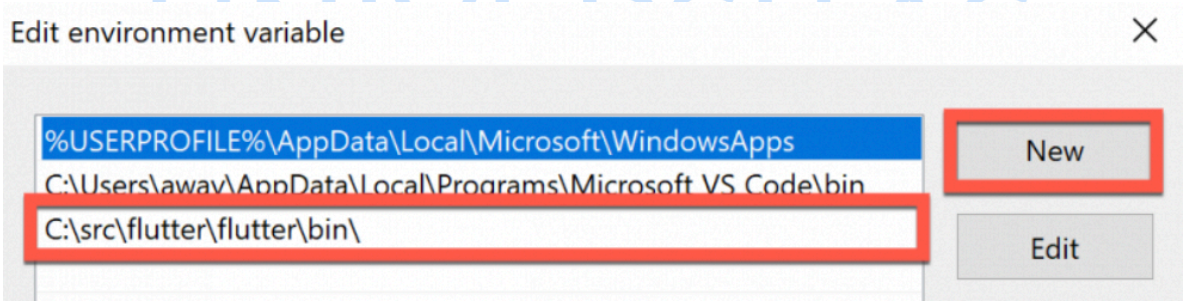
Setup dan Instalasi Flutter

Bagian ini akan menjelaskan tahapan instalasi Flutter SDK dan kebutuhan sistem pada sistem operasi Windows.

Tahap 1: Download Flutter SDK dengan mengakses link **https://docs.flutter.dev/get-started/install** Download versi stable 2.10.2 (update Maret 2022) dengan nama file flutter_windows_2.10.2-stable.zip

Tahap 2: Unzip file arsip dan letakkan pada folder C:\Flutter\ *(letak folder bebas)*

Tahap 3: Update system path windows untuk menambahkan direktori flutter bin.



Tahap 4: Flutter menyediakan tools untuk pengecekan kebutuhan dalam pengembangan aplikasi menggunakan flutter yaitu:

```
flutter doctor
```

Tahap 5: Jalankan perintah di atas melalui terminal (CMD pada windows) untuk melakukan analisis sistem dan menampilkan laporan seperti tampilan berikut:

```
sio@macbook-air-sio ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.10.2, on macOS 12.2.1 21D62 darwin-arm, locale
    en-ID)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 13.2.1)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2021.1)
[✓] VS Code (version 1.64.2)
[✓] Connected device (1 available)
[✓] HTTP Host Availability

• No issues found!
```

Tahap 6: Install Android SDK, if reported by flutter doctor.

Tahap 7: Install Android Studio, if reported by flutter doctor (Opsional).

Tahap 8: Install VS Code, if reported by flutter doctor.

Tahap 9: Install external emulator, atau jalankan android emulator maupun menggunakan perangkat android yang terhubung melalui koneksi kabel.

Tahap 10: Install Plugin Flutter dan Dart di VS Code atau Android Studio. Ini memudakan developer untuk menggunakan template aplikasi flutter, running dan debugging aplikasi, ect.

Tahap 10: Buat proyek baru pada VS code melalui command pallette (**CTRL + Shift + P)**, kemudian ketik: **Flutter: New Project.** Arahkan pada folder tujuan dan beri nama aplikasi.

Run Flutter project:
Pastikan telah menjalankan emulator atau terhubung pada perangkat mobile, kemudian run perintah berikut melalui terminal pada root project folder.

```
flutter run
```

r ← untuk **hot reload**

Struktur folder flutter

Sebelum mulai membangun aplikasi menggunakan flutter, developer harus mengenal struktur folder, codebase flutter dan menerapkan clean architecture. Secara umum, hirarki folder dan file pada flutter adalah sebagai berikut:

```
--project_name
   |--android
   |--ios
   |--lib
      |--main.dart
   |--test
   |--web
   |--windows
--.gitignore
--.metadata
--.gitignore
--.packages
--analysis_options.yaml
--project_name.iml
--pubspec.lock
--README.md
```

Penjelasan

Android: Folder ini digunakan untuk proses build menjadi aplikasi native ke platform android. Terdiri dari file ***build.gradle*** untuk menentukan versi SDK dari aplikasi, dan ***AndroidManifest.xml*** untuk mengatur permissions, seperti izin akses kamera, storage, lokasi, ect.

IOS: Memiliki fungsi yang sama seperti folder android yaitu diperlukan untuk menjalankan aplikasi ke platform IOS.

lib: Letak dari file dart aplikasi flutter. Saat instalasi awal, akan ada 1 file dart sebagai main / root file dari aplikasi flutter yaitu **main.dart**.

test: Dapat digunakan untuk kebutuhan testing dari kode yang telah ditulis.

assets: Secara default tidak tersedia. Ini bisa ditambahkan di dalam folder project yang nantinya dapat digunakan untuk menyimpan gambar, font dan file lainnya yang diperlukan dalam pengembangan aplikasi flutter.

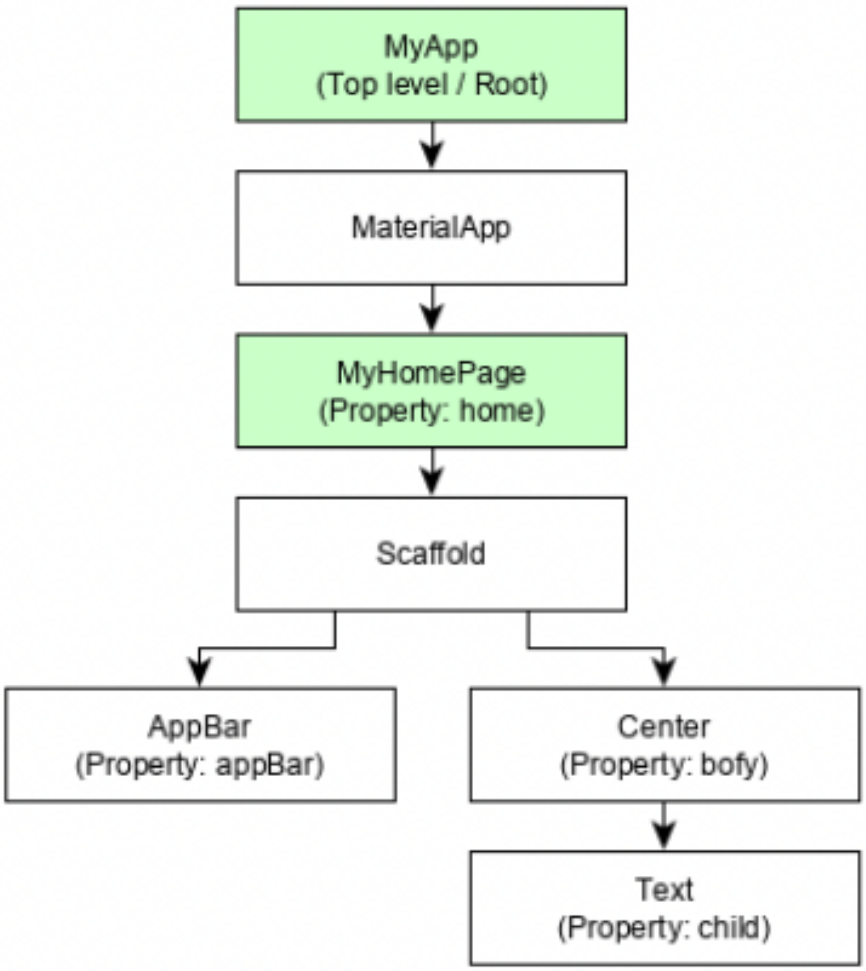
pubspec.yaml: Berisi konfigurasi dari proyek flutter yang terdiri dari nama, dekripsi dan versi flutter, depedencies yang digunakan dan inisialisasi assets yang digunakan.

Lain-lain: File yang biasa diciptakan saat membuat proyek flutter.

Komponen Widget flutter

Konsep utama dari framework flutter adalah "pada flutter, semua adalah widget". Widget adalah komponen User Interface (UI) yang digunakan untuk membuat user interface dari aplikasi. Seperti komponen Scaffold, Container, Button, Text, ect.

UI pada flutter terdiri dari 1 (satu) atau lebih children (widgets), yang kemudian bisa terdiri dari widget lainnya. Berikut adalah hirarki sederhana dari aplikasi flutter.



Gambar 1 contoh Hirarki widget flutter

Perhatikan file **main.dart**.

Bari awal pada file main.dart akan terdiri dari perintah import package yang digunakan, seperti package default yaiti **material.dart.** Package ini digunakan untuk membuat UI (user interface) berdasarkan pedoman material desain Android (akses: https://material.io/components/).

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

Ini adalah bagian awal untuk mengakses aplikasi flutter. Bagian ini memanggil fungsi runApp dan memberikan objek kelas MyApp. tujuannya adalah untuk menampilkan widget ke layar.

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

Class **MyApp** mewarisi property dari **StatelessWidget**. Konsep Stateless digunakan untuk widget yang statenya tidak berubah. Contohnya untuk menampilkan statik Text "Hello world", yang tidak akan berubah.

method **build** bertujuan untuk membuat bagian dari UI aplikasi yaitu MaterialApp. Ini adalah sebuah widget untuk konfigurasi root UI dari aplikasi. Terdapat 3 properti yaitu: **Title** untuk judul aplikasi, **Theme** untuk konfigurasi themeData yang digunakan oleh aplikasi, dan Home merupakan konfigurasi UI yang pertama dipanggil saat menjalankan aplikasi.

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

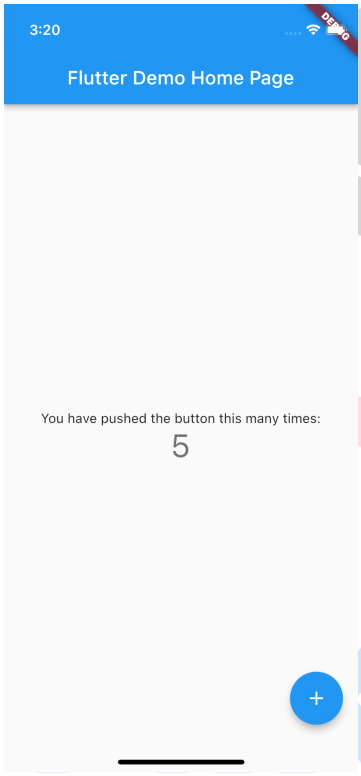
class MyHomePage adalah StatefullWidget yang mengembalikan Widget Scaffold pada layar. Konsep statefull adalah widget yang dapat berubah-ubah secara dinamis, contohnya fungsi counter (default fungsi saat membuat aplikasi flutter).

Pada lingkup `class MyHomePage extends StatefulWidget` kita dapat inisialisasi variabel yang akan menjadi properti dari Widget MyHomePage.

Pada lingkup `class _MyHomePageState extends State<MyHomePage>` kita dapat menggunakan properti dari MyHomepage, inisialisasi variabel, membuat fungsi, dan menambahkan component material seperti Scaffold.

```
class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title}) : super(key: key);
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
```

```
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),
    );
  }
}
```



Tampilan default aplikasi flutter.

Text, Icon, Image

Text

Learn more: *https://docs.flutter.dev/development/ui/widgets/text*
Widget text menampilkan string dengan *single style*.

Pada main.dart, silahkan ganti baris ini dan terapkan style

```
const Text('You have pushed the button this many times:',)
```

Sehingga menjadi:


```
const Text(
              'Total:',
              style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 16,
                  color: Colors.blueGrey,
                  letterSpacing: 2),
            ),
```

Widget text memiliki constructor (baca, subclass) yaitu Text.rich() yang dapat menampilkan paragraf dengan style TextSpans berbeda. Contoh:

```
  TextSpan(
                text: 'Aplikasi', // default text style
                children: <TextSpan>[
                  TextSpan(
                      text: ' belajar ',
                      style: TextStyle(fontStyle: FontStyle.italic)),
                  TextSpan(
                      text: 'berhitung',
                      style: TextStyle(fontWeight: FontWeight.bold)),
                ],
              ),
            )
```

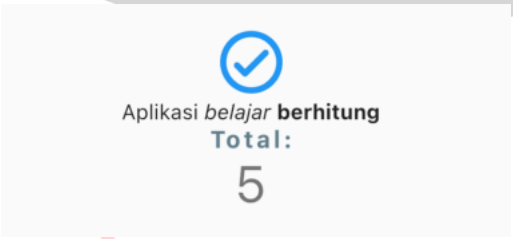
Hasil:

Icon

Secara default flutter menggunakan icon dari material desain yang liat assetnya bisa dilihat melalui link https://fonts.google.com/icons?selected=Material+Icons. Widget icon digunakan untuk menampilkan icon pada aplikasi flutter. Tambahkan widget icon di atas Widget text.

```
const Icon(
              Icons.check_circle_outline_rounded,
              color: Colors.blue,
              size: 50,
            ),
```

Hasil:



learn more: https://api.flutter.dev/flutter/material/Icons-class.html

Image

Image adalah widget yang dapat digunakan untuk menampilkan asset gambar atau file gambar dari internet. Agar aplikasi flutter dapat mengenali dan menggunakan gambar, maka perlu dildenfikasi lokasi assets pada pubspec.yaml

Langkah 1: Tambahkan folder 'assets' di dalam root folder.

Langkah 2: Tambahkan gambar di dalam folder assets.

Langkah 3: Identifikasi lokasi assets pada **pubspec.yaml**.

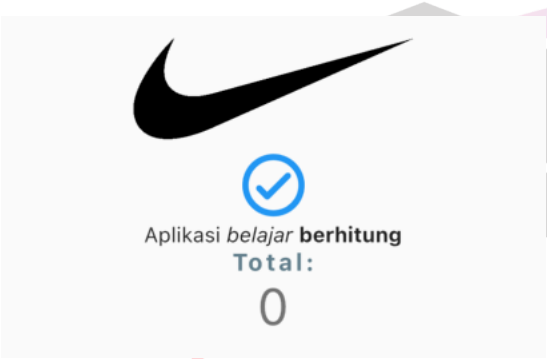
```
# To add assets to your application, add an assets section, like this:
  assets:
    - assets/
```

Langkah 4: Restart aplikasi agar aplikasi flutter mengenali assets yang baru ditambah.

Langkah 5: Tambahkan widget image dengan menuliskan path letak gambar yang telah ditambahkan.

```
Image.asset('assets/logo.png', width: 200),
```

Hasil:



Selain menambahkan gambar dari assets, Image punya subclass **Image.network()** untuk menampilkan gambar dari internet. Silahkan cari url gambar dari internet dan tambahkan pada properti url.

```
Image.network(
              'https://purepng.com/public/uploads/large/purepng.com-
mariomariofictional-charactervideo-gamefranchisenintendodesigner-
1701528634653vywuz.png',
              width: 200,
            ),
```

learn more: https://docs.flutter.dev/development/ui/assets-and-images