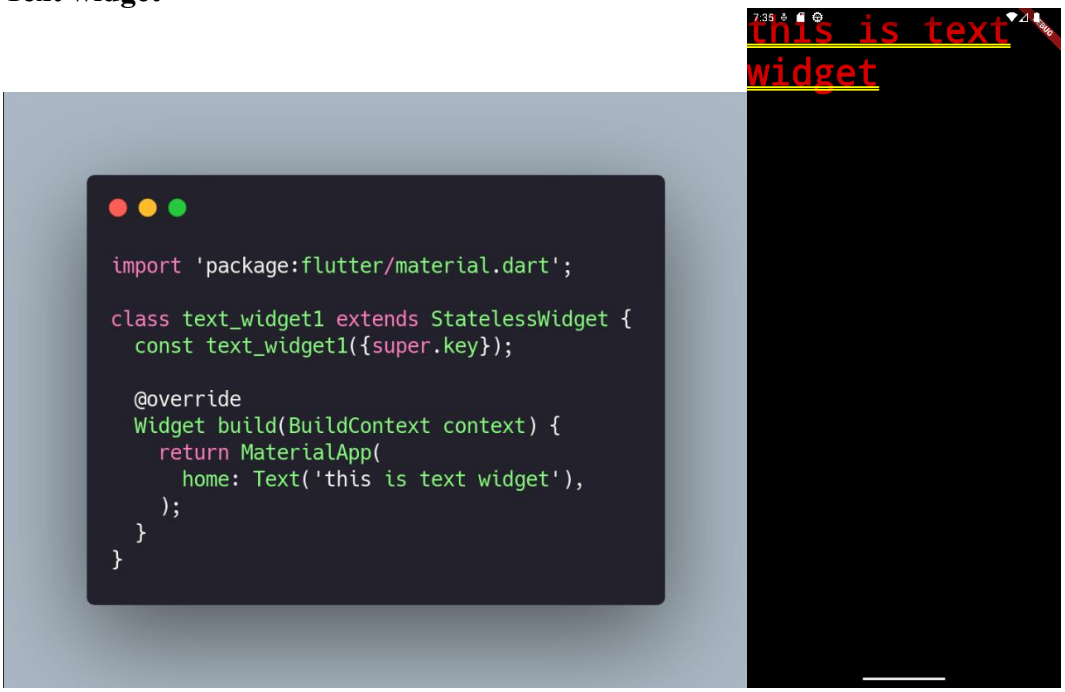
Nama : M.Najwan Hibatullah / XIR3

Presensi : 23

PENJELASAN WIDGET

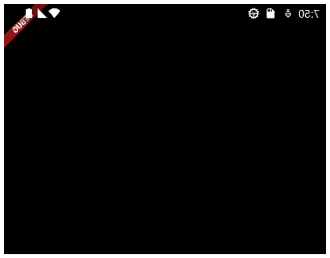
1. **Text widget**



Kode ini menjelaskan untuk mengeluarkan text di sebuah device android pada bagian ‘home’. Bagian yang berisi library dan widget untuk membuat sebuah ui dapat dilakukan dengan cara import package:flutter/material.dart. Disambung dengan membuat class dan membuat widget build yang digunakan untuk membuat sebuah interface.

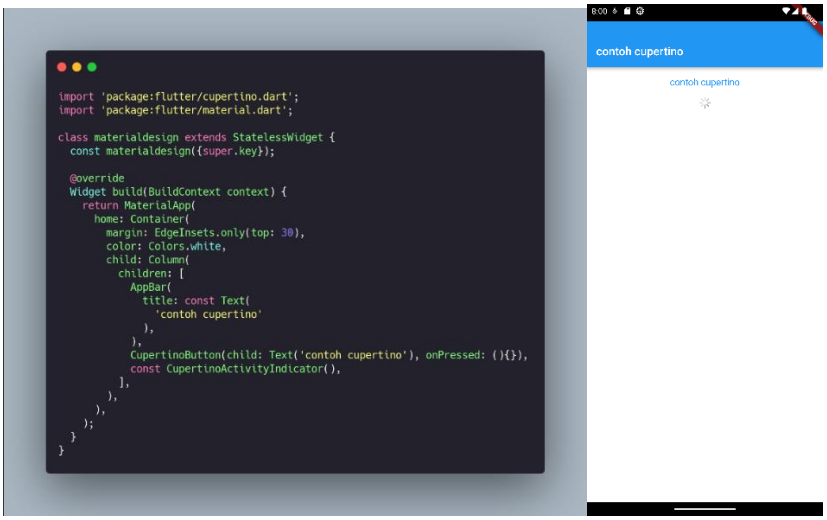
2. **Image Widget**

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

class image_1 extends StatelessWidget {
  const image_1({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp (
      home: Image(image:
      NetworkImage('https://images.fineartamerica.com/images-medium-large-5/schrodinger-wave-equation-
science-photo-library.jpg')
      ),
    );
  }
}
```



Baris kode ini menjelaskan untuk mengeluarkan sebuah gambar di laman android bagian ‘Home’ dengan menggunakan property home dan mengambil asset dari ‘image’ yang berasal dari sebuah site.

3. **Material app and Cupertino**



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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Container(
        margin: EdgeInsets.only(top: 30),
        color: Colors.white,
        child: Column(
          children: [
            AppBar(
              title: const Text(
                'contoh cupertino'
              ),
            ),
            CupertinoButton(child: Text('contoh cupertino'), onPressed: (){}),
            const CupertinoActivityIndicator(),
          ],
        ),
      ),
    );
  }
}
```

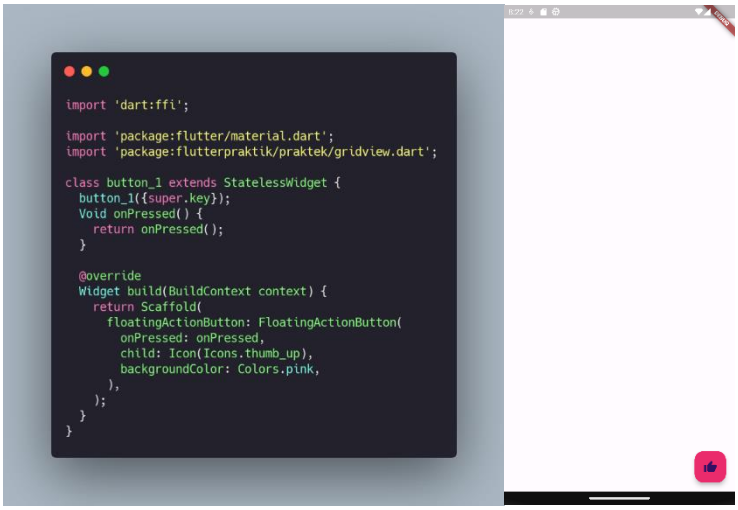

dalam baris ini kita harus import paket ‘flutter/cupertino’ dan ‘flutter/material’, lalu di bagian container memiliki background warna putih dan memiliki margin top sebesar 30.

Appbar digunakan untuk menampilkan “contoh Cupertino”, Appbar adalah bagian material design yang digunakan untuk menampilkan judul aplikasi di halaman header

Cupertino digunakan untuk menampilkan “contoh Cupertino” yang dibagian bawah appbar, Ketika ditekan maka tidak ada Tindakan yang terjadi

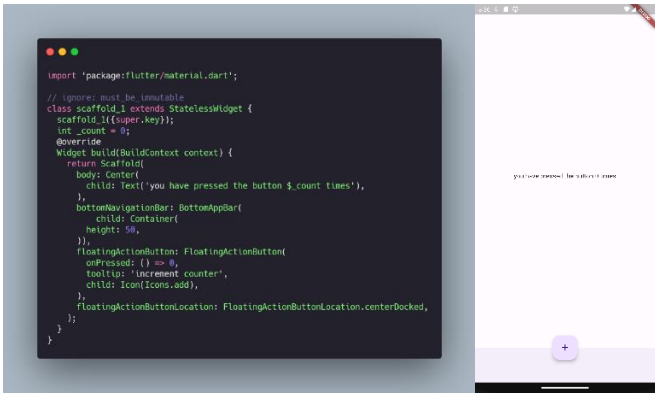
CupertinoActivityIndicator digunakan untuk menampilkan jika android sedang melakukan sebuah proses melakukan tugas dan automatis terjadi loading dan cupertinoActivityIndicator akan terus berputar sampai tugas nya selsesai.

4. **Button**



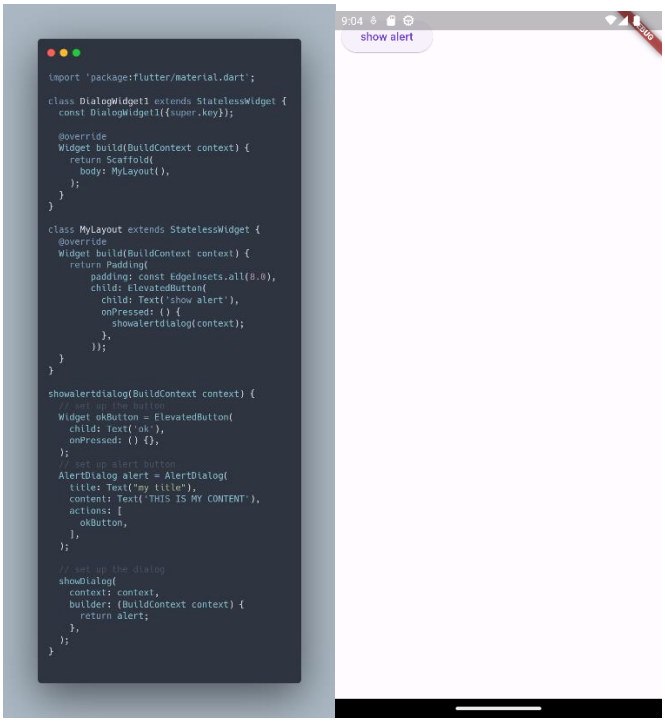
'floatingActionButton' ini digunakan untuk menampilakan sebuah tombol navigasi yang berada di laman dengan menggunakan icon thumb up dan jika di pencet tidak mengeksekusi apa pun. Lalu background color thumb up menggunakan warna pink

5. **Scaffold**



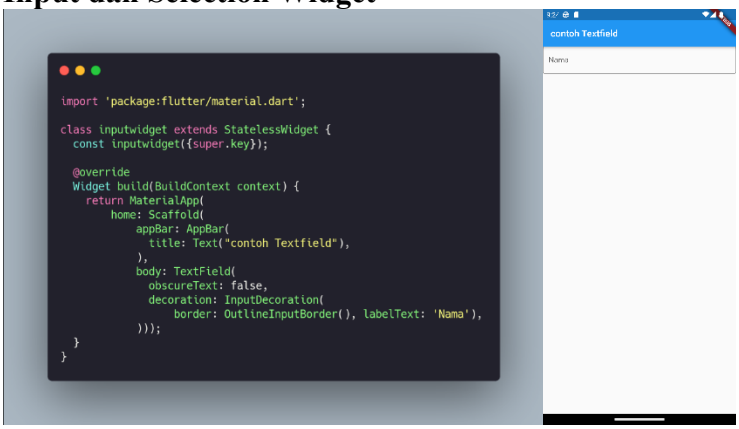
Scaffold digunakan untuk tata letak untuk material design, dan disini adalah sebuah floating action button menggunakan counter seberapa user menekan tombol action button tersebut.

6. **Dialog**



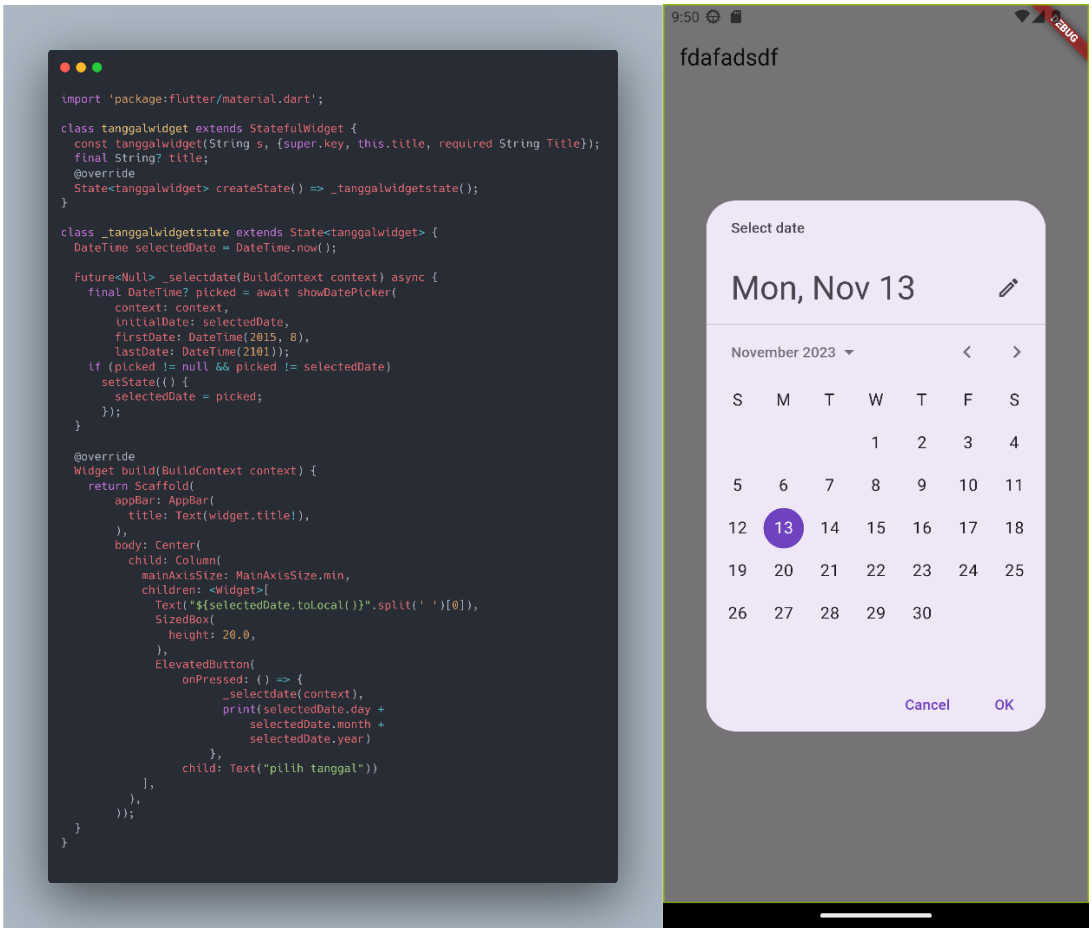
DI baris kode ini menjelaskan bahwa jika kita menekan tombol ‘Show alert’ maka akan keluar sebuah pop up yang menunjukkan dari property ‘showalertdialog’ berfungsi, property ini digunakan Ketika anda mau melakukan yang sangat krusial, sebagai contoh jika anda melakukan transaksi atau membeli seusatu yang bersangkut paut dengan sebuah yang sangat krusial kedepannya.

7. Input dan Selection Widget



Kode ini adalah implementasi widget Flutter sederhana untuk membuat antarmuka pengguna hanya dengan satu input teks. Kelas "inputwidget" adalah turunan dari "StatelessWidget", yang menampilkan aplikasi berjudul "Contoh Bidang Teks" di bilah aplikasi. Halaman utama berisi “TextField” untuk memasukkan teks berjudul “Nama”. Kode ini membuat UI Flutter dasar untuk input teks.

8. **Date And Time**



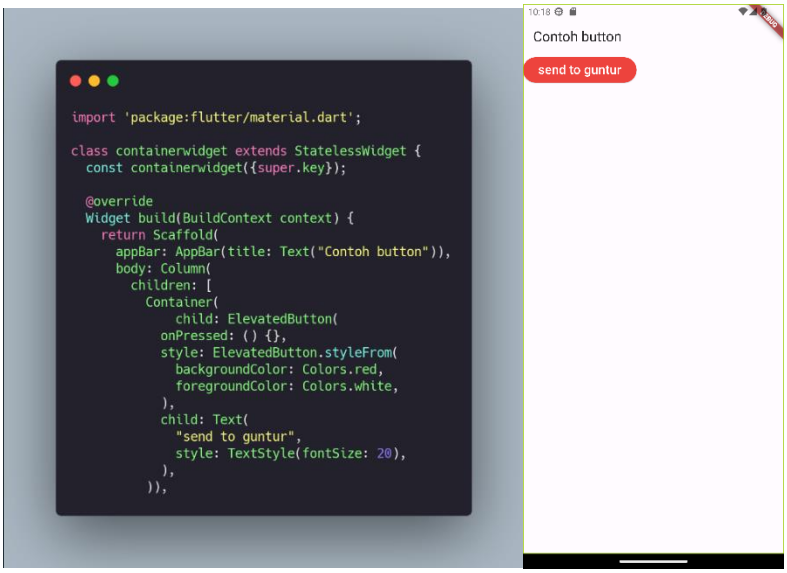
```dart
import 'package:flutter/material.dart';

class tanggalwidget extends StatefulWidget {
  const tanggalwidget(String s, {super.key, this.title, required String Title});
  final String? title;
  @override
  State<tanggalwidget> createState() => _tanggalwidgetstate();
}

class _tanggalwidgetstate extends State<tanggalwidget> {
  DateTime selectedDate = DateTime.now();

  Future<Null> _selectdate(BuildContext context) async {
    final DateTime? picked = await showDatePicker(
        context: context,
        initialDate: selectedDate,
        firstDate: DateTime(2015, 8),
        lastDate: DateTime(2101));
    if (picked != null && picked != selectedDate)
      setState(() {
        selectedDate = picked;
      });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text(widget.title!),
        ),
        body: Center(
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              Text("${selectedDate.toLocal()}".split(' ')[0]),
              SizedBox(
                height: 20.0,
              ),
              ElevatedButton(
                  onPressed: () => {
                        _selectdate(context),
                        print(selectedDate.day +
                            selectedDate.month +
                            selectedDate.year)
                      },
                  child: Text("pilih tanggal"))
            ],
          ),
        ));
  }
}
```


Kode ini merupakan implementasi widget Flutter untuk UI yang memungkinkan pengguna memilih tanggal.

Kelas "datewidget" adalah turunan dari "StatefulWidget" dengan konstruktor yang menerima beberapa parameter, termasuk parameter opsional dan wajib.
Kelas status '_datewidgetstate' mendefinisikan perilaku widget, dengan properti 'selectedDate' mewakili tanggal yang dipilih, awalnya ditetapkan sebagai tanggal saat ini.

Terdapat metode '_selectdate' untuk menampilkan dialog pemilihan tanggal dan memperbarui 'Tanggal yang Dipilih' berdasarkan pilihan pengguna. Antarmuka pengguna menampilkan tanggal yang dipilih dan tombol "pilih tanggal".
Jika tombol ini ditekan, akan muncul kotak dialog pemilihan tanggal dan tanggal yang dipilih akan ditampilkan dan dicetak dalam format hari, bulan, tahun. Secara keseluruhan, kode ini memberi pengguna kemampuan untuk memilih tanggal menggunakan fitur pemilih tanggal bawaan Flutter.

9. Container (Prop child)

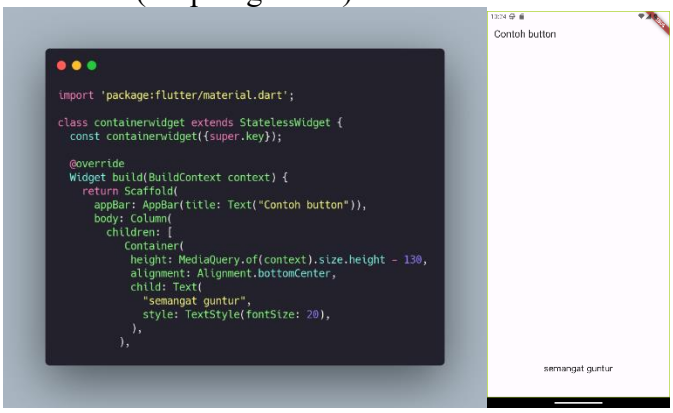


Kode tersebut memiliki keterhubungan antara properti `child` dan struktur keseluruhan. Dalam bagian yang menciptakan tombol dengan `ElevatedButton`, properti `child` digunakan untuk menentukan isi atau kontennya, yaitu teks "send to guntur" dengan ukuran font 20. Ini memberikan informasi tentang apa yang akan ditampilkan di dalam tombol.

Selanjutnya, tombol tersebut ditempatkan dalam sebuah widget `Container`. Meskipun properti `child` pada `Container` tidak memberikan pengaruh khusus dalam konteks ini, penggunaan `Container` bisa digunakan untuk mengatur tata letak atau gaya tertentu di sekitar tombol.

Secara keseluruhan, penggunaan properti `child` pada `ElevatedButton` memberikan kemampuan untuk menentukan konten tombol, dan penggunaan `Container` mengelola struktur dan gaya umum dari elemen-elemen di dalamnya, meskipun dalam contoh ini tidak digunakan secara khusus untuk memodifikasi properti `child`.

10. Container (Prop alignment)



Kode tersebut mengimplementasikan sebuah widget Flutter yang menampilkan antarmuka pengguna dengan menggunakan widget `Container`. Dalam struktur UI, terdapat sebuah `Scaffold` yang memiliki `AppBar` dengan judul "Contoh button". Bagian inti dari antarmuka berada dalam sebuah kolom (`Column`), di mana terdapat satu `Container`.

`Container` ini memiliki properti ketinggian (height) yang diatur dinamis sesuai dengan tinggi layar dikurangi 130. Selain itu, pengaturan `alignment` pada `Container`

diterapkan untuk menempatkan isi ke bagian bawah tengah layar. Di dalam `Container`, terdapat sebuah teks dengan isi "semangat guntur" dan ukuran font 20.

Penggunaan `Container` dalam konteks ini memberikan kontrol terhadap tata letak dan penataan elemen di dalamnya. Keseluruhan, kode ini menciptakan antarmuka pengguna dengan desain yang memanfaatkan properti `Container` untuk mengelola tampilan dan tata letak dengan lebih fleksibel.

11. Container (Prop color)

Bagian properti 'color' dalam kode tersebut digunakan untuk menentukan warna latar belakang dari 'Container'. Pada contoh ini, properti 'color' diatur sebagai 'Colors.yellow', sehingga latar belakang 'Container' akan memiliki warna kuning. Penetapan warna tertentu pada properti 'color' memberikan elemen antarmuka pengguna (UI), dalam hal ini 'Container', tampilan visual yang konsisten dengan desain yang diinginkan. Warna kuning pada latar belakang 'Container' menciptakan kontras dengan teks di dalamnya, membuatnya lebih mudah terbaca.

Dengan menggunakan properti 'color', kita dapat dengan mudah mengubah tampilan dan estetika dari elemen UI yang bersangkutan. Pemilihan warna tertentu dapat memberikan kesan yang diinginkan, tergantung pada desain aplikasi secara keseluruhan. Jadi, dalam konteks kode tersebut, atribut 'color' digunakan untuk memberikan warna latar belakang kuning pada 'Container', menambahkan elemen visual dan desain ke antarmuka pengguna secara keseluruhan.

12. Container (height n Width prop)

Bagian properti 'width' dan 'height' dalam kode tersebut digunakan untuk mengatur dimensi dari widget 'Container'. Pada contoh ini, properti 'width' diatur sebagai 200 dan 'height' diatur juga sebagai 200. Oleh karena itu, 'Container' akan memiliki lebar (width) dan tinggi (height) masing-masing sebesar 200 piksel. Pengaturan dimensi ini memungkinkan pengembang secara spesifik menentukan ukuran dari suatu elemen antarmuka pengguna (UI), dalam hal ini 'Container'.

Dengan menetapkan nilai tertentu untuk 'width' dan 'height', kita dapat mengendalikan seberapa besar atau kecil suatu elemen akan muncul di layar. Dalam kasus kode ini, 'Container' dengan lebar dan tinggi 200 piksel akan berukuran relatif kecil di dalam antarmuka pengguna. Properti ini dapat digunakan untuk mencapai tata letak dan desain yang diinginkan sesuai dengan kebutuhan aplikasi.

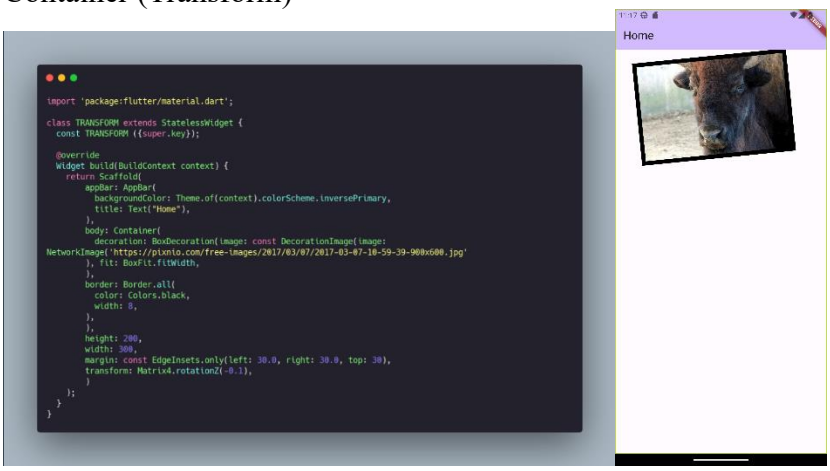
13. Container (Margin prop)

Bagian properti "margin" dari kode digunakan untuk menentukan batas (margin) di sekitar "Container". Dalam contoh ini, properti 'margin' disetel ke 'EdgeInsets.all(50)', artinya ada batas 50 piksel di semua tepinya ("atas", "kanan", "bawah", dan "kiri") dari 'Container'. Menggunakan properti 'margin' memungkinkan pengembang mengontrol ruang putih di sekitar elemen dalam tata letak antarmuka pengguna. Properti ini berguna dalam memberikan jarak visual antara elemen dengan elemen lain di sekitarnya. Dengan menetapkan nilai "margin", pengembang dapat mencapai tata letak yang sesuai dengan desain yang diinginkan, sekaligus memberikan elemen UI, dalam hal ini "Container", tampilan yang menarik secara visual. Jadi, properti “margin” dapat digunakan untuk mengontrol jarak antara elemen dan elemen sekitarnya di dalam antarmuka pengguna Flutter.

14. Container (Padding prop)

Bagian properti 'padding' dari kode tersebut digunakan untuk menentukan spasi (padding) di sekitar konten (child) dari widget 'Container'. Dalam contoh ini, properti 'padding' disetel ke 'EdgeInsets.only(left: 20)', yang berarti hanya ada 20 piksel padding di sisi kiri 'Container'. Menggunakan properti 'padding' memungkinkan pengembang mengontrol ruang kosong di sekitar konten elemen dalam widget. Dengan menetapkan nilai 'padding', pengembang dapat memberikan jarak visual antara konten 'Container' dan tepinya, menciptakan tata letak dan tampilan yang lebih terorganisir. Dalam hal ini, pengaturan 'padding' membuat teks 'pola pikir pembelajaran' di dalam 'Container' berjarak 20 piksel dari tepi kiri. Properti ini berguna untuk mencapai tata letak yang sesuai dengan desain yang diinginkan, serta memberikan tampilan visual yang lebih terorganisir pada elemen UI, dalam hal ini "Container".

15. Container (Transform)



Kode tersebut merupakan implementasi widget Flutter yang menggambarkan transformasi pada suatu kontainer (Container). Dalam strukturnya, terdapat kelas `TRANSFORM` yang merupakan turunan dari `StatelessWidget`. Dalam metode `build`, sebuah `Scaffold` dibuat sebagai kerangka kerja dasar aplikasi dengan `AppBar` berjudul "Home" dan latar belakang warna yang diambil dari tema aplikasi.

Pada bagian body, terdapat suatu `Container` yang memiliki dekorasi berupa gambar dari URL eksternal. Gambar tersebut diatur untuk mengisi lebar kontainer sesuai kebutuhan

dengan menggunakan `BoxFit.fitWidth`. Kontainer ini juga memiliki batas berwarna hitam dengan lebar 8 piksel.

Dimensi dari kontainer ditetapkan dengan tinggi 200 piksel dan lebar 300 piksel, sementara margin diberikan pada sisi kiri, kanan, dan atas sebesar 30 piksel. Paling menarik, terdapat pula transformasi yang diterapkan pada kontainer dengan menggunakan matriks rotasi sebesar -0.1 radian pada sumbu Z. Ini menghasilkan efek visual berupa rotasi kontainer.

Secara keseluruhan, kode ini menghasilkan tampilan antarmuka pengguna Flutter yang menarik dengan penggunaan transformasi pada suatu elemen, memberikan efek rotasi pada kontainer yang menampilkan gambar dengan dekorasi batas.

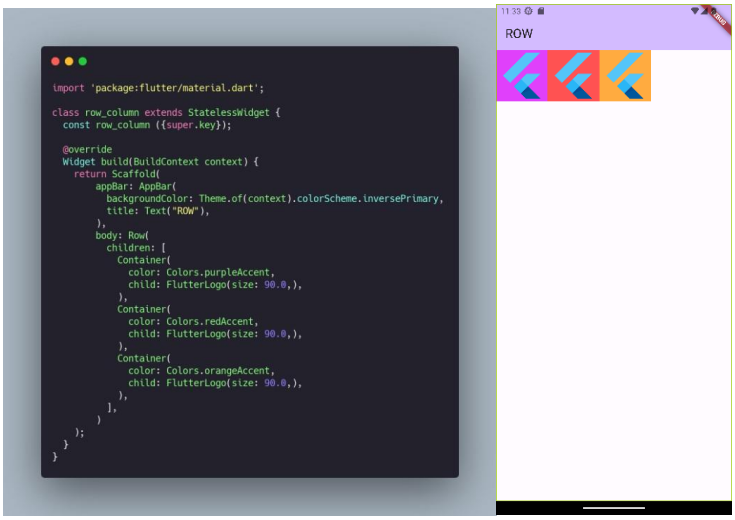
16. Container (Property decoration)

Kode tersebut merupakan implementasi widget Flutter yang menggunakan transformasi dan dekorasi pada suatu kontainer. Dalam struktur kodenya, terdapat kelas `transform_decoration` yang menjadi turunan dari `StatelessWidget`. Pada metode `build`, sebuah `Scaffold` dibuat dengan `AppBar` berjudul "Home" dan latar belakang warna yang diambil dari tema aplikasi.

Di dalam body, terdapat sebuah `Container` yang memiliki dekorasi berupa gambar dari URL eksternal. Gambar tersebut diatur untuk mengisi lebar kontainer sesuai kebutuhan dengan menggunakan `BoxFit.fitWidth`. Kontainer ini juga memiliki batas berwarna hitam dengan lebar 8 piksel dan sudut-sudut yang dibulatkan dengan `borderRadius` sebesar 12.

Dimensi dari kontainer ditetapkan dengan tinggi 200 piksel dan lebar 200 piksel, sementara margin diberikan sebesar 20 piksel. Keseluruhan struktur ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan transformasi dan dekorasi pada suatu elemen, yaitu kontainer dengan gambar, batas, dan sudut-sudut yang dibulatkan.

17. Row

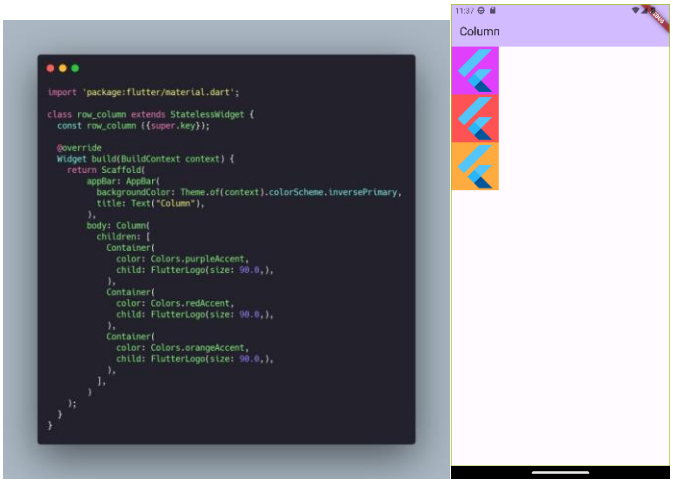


Kode di atas merupakan implementasi widget Flutter yang menggunakan tata letak `Row` untuk menyusun beberapa kontainer secara horizontal. Dalam kelas `row_column`, yang merupakan turunan dari `StatelessWidget`, terdapat metode `build` yang mengembalikan suatu widget `Scaffold`. `Scaffold` ini menyediakan kerangka kerja dasar untuk halaman aplikasi dengan `AppBar` berjudul "ROW" yang memiliki latar belakang warna yang diambil dari tema aplikasi.

Pada bagian body, terdapat widget `Row` yang mengandung beberapa `Container`. Setiap `Container` memiliki latar belakang warna yang berbeda (ungu, merah, dan oranye) dengan menggunakan properti `color`. Di dalam setiap `Container`, terdapat widget `FlutterLogo` yang menampilkan logo Flutter dengan ukuran sebesar 90.0 piksel.

Secara keseluruhan, kode ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan tata letak `Row` untuk menyusun tiga kontainer secara horizontal, masing-masing dengan latar belakang warna yang berbeda dan berisi logo Flutter.

18. Column



Kode di atas merupakan implementasi widget Flutter yang menggunakan tata letak `Column` untuk menyusun beberapa kontainer secara vertikal. Dalam kelas `row_column`, yang merupakan turunan dari `StatelessWidget`, terdapat metode `build` yang mengembalikan suatu widget `Scaffold`. `Scaffold` ini menyediakan

kerangka kerja dasar untuk halaman aplikasi dengan `AppBar` berjudul "Column" yang memiliki latar belakang warna yang diambil dari tema aplikasi.

Di dalam bagian body, terdapat widget `Column` yang berisi tiga `Container`. Setiap `Container` memiliki latar belakang warna yang berbeda (ungu, merah, dan oranye) menggunakan properti `color`. Di dalam setiap `Container`, terdapat widget `FlutterLogo` yang menampilkan logo Flutter dengan ukuran sebesar 90.0 piksel.

Secara keseluruhan, kode ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan tata letak `Column` untuk menyusun tiga kontainer secara vertikal, masing-masing dengan latar belakang warna yang berbeda dan berisi logo Flutter.

19. Stack

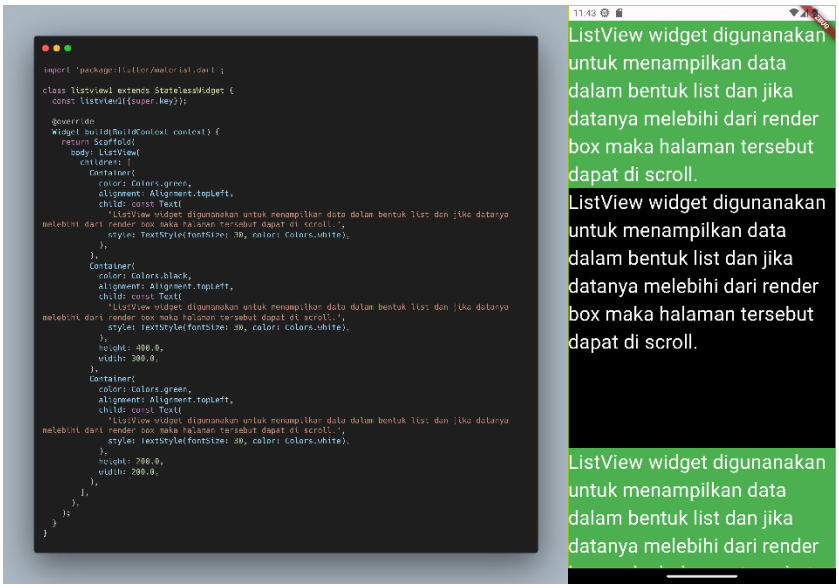
Kode di atas merupakan implementasi widget Flutter yang menggunakan tata letak `Stack` untuk menumpuk beberapa kontainer di atas satu sama lain. Dalam kelas `stack1`, yang merupakan turunan dari `StatelessWidget`, terdapat metode `build` yang mengembalikan suatu widget `Scaffold`. `Scaffold` ini memberikan kerangka kerja dasar untuk halaman aplikasi.

Di dalam bagian body, terdapat widget `Stack` yang berisi tiga `Container`. Setiap `Container` memiliki latar belakang warna yang berbeda (hijau, merah, dan ungu) menggunakan properti `color`. Masing-masing juga memiliki teks yang ditampilkan di bagian bawahnya dengan ukuran dan warna tertentu.

Penting untuk dicatat bahwa kontainer-kontainer tersebut ditempatkan di posisi yang sama di bagian bawah layar karena memiliki properti `alignment: Alignment.bottomCenter`. Selain itu, dua kontainer yang terakhir memiliki dimensi yang berbeda, dengan yang kedua memiliki tinggi 400.0 dan lebar 300.0, sedangkan yang ketiga memiliki tinggi 200.0 dan lebar 200.0.

Secara keseluruhan, kode ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan tata letak `Stack` untuk menumpuk tiga kontainer, masing-masing dengan warna dan ukuran yang berbeda, di bagian bawah layar.

20. Listview



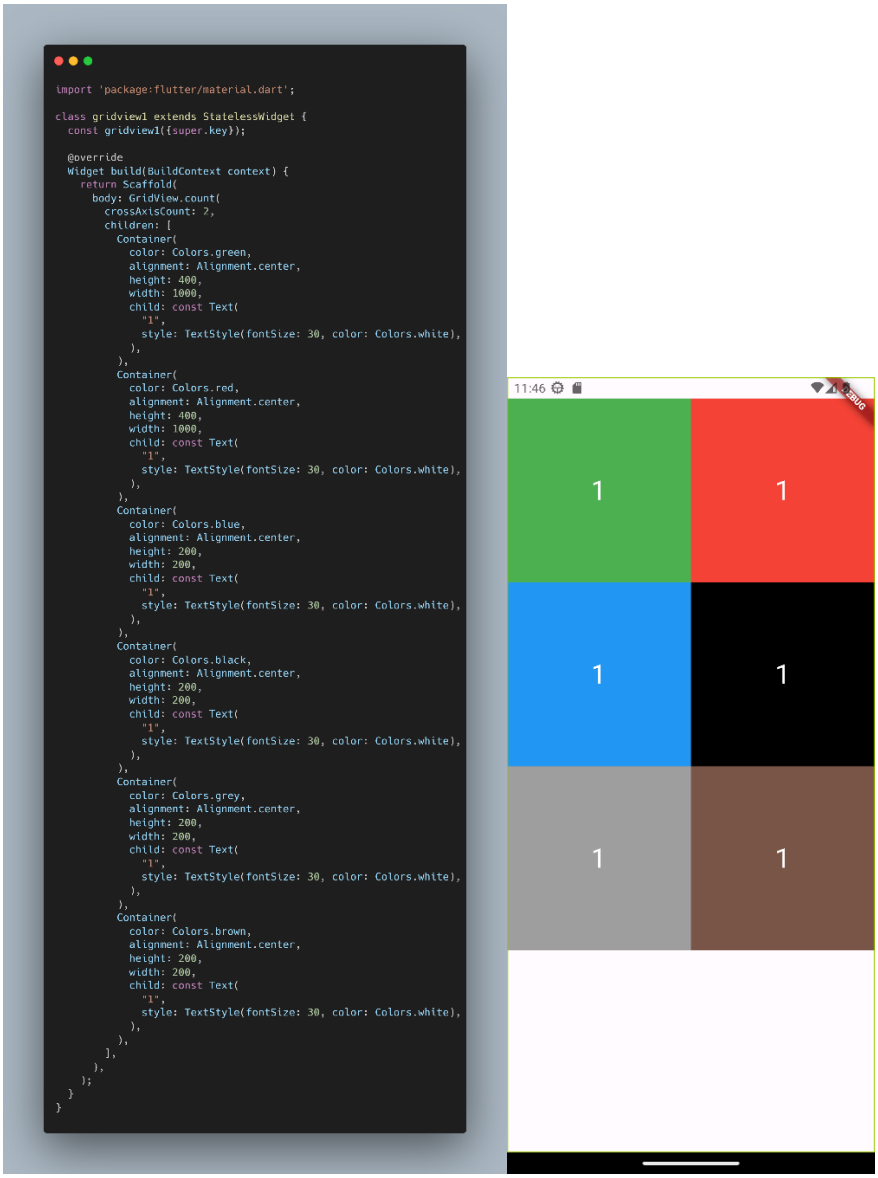
Kode di atas adalah implementasi dari widget Flutter yang menggunakan `ListView` untuk menampilkan daftar kontainer dengan teks di dalamnya. Dalam kelas `listview1`, yang merupakan turunan dari `StatelessWidget`, terdapat metode `build` yang mengembalikan suatu widget `Scaffold`. `Scaffold` ini memberikan kerangka kerja dasar untuk halaman aplikasi.

Di dalam body, terdapat widget `ListView` yang berisi tiga `Container`. Setiap `Container` memiliki latar belakang warna yang berbeda (hijau, hitam, dan hijau kembali) menggunakan properti `color`. Di dalam setiap `Container`, terdapat teks yang menjelaskan penggunaan `ListView` dalam Flutter.

Penting untuk dicatat bahwa kontainer-kontainer tersebut memiliki properti `alignment: Alignment.topLeft`, sehingga teksnya diletakkan di bagian kiri atas setiap kontainer. Selain itu, dua kontainer yang terakhir memiliki dimensi yang berbeda, dengan yang kedua memiliki tinggi 400.0 dan lebar 300.0, sedangkan yang ketiga memiliki tinggi 200.0 dan lebar 200.0.

Secara keseluruhan, kode ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan `ListView` untuk menampilkan tiga kontainer, masing-masing dengan warna dan dimensi yang berbeda, serta teks yang menjelaskan fungsi dari `ListView`. Jika konten melebihi render box, halaman dapat di-scroll untuk melihat lebih banyak isi.

21. Grid view



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

class gridview1 extends StatelessWidget {
  const gridview1({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: GridView.count(
        crossAxisCount: 2,
        children: [
          Container(
            color: Colors.green,
            alignment: Alignment.center,
            height: 400,
            width: 1000,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
          Container(
            color: Colors.red,
            alignment: Alignment.center,
            height: 400,
            width: 1000,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
          Container(
            color: Colors.blue,
            alignment: Alignment.center,
            height: 200,
            width: 200,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
          Container(
            color: Colors.black,
            alignment: Alignment.center,
            height: 200,
            width: 200,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
          Container(
            color: Colors.grey,
            alignment: Alignment.center,
            height: 200,
            width: 200,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
          Container(
            color: Colors.brown,
            alignment: Alignment.center,
            height: 200,
            width: 200,
            child: const Text(
              "1",
              style: TextStyle(fontSize: 30, color: Colors.white),
            ),
          ),
        ],
      ),
    );
  }
}
```


Kode di atas merupakan implementasi widget Flutter yang menggunakan `GridView` untuk menyusun kontainer dalam bentuk grid. Dalam kelas `gridview1`, yang merupakan turunan dari `StatelessWidget`, terdapat metode `build` yang mengembalikan suatu widget `Scaffold`. `Scaffold` ini memberikan kerangka kerja dasar untuk halaman aplikasi.

Di dalam body, terdapat widget `GridView.count` yang mendefinisikan grid dengan jumlah kolom sebanyak 2 (`crossAxisCount: 2`). Grid ini berisi enam `Container` yang masing-masing memiliki latar belakang warna yang berbeda (hijau, merah, biru, hitam, abu-abu, dan cokelat). Setiap `Container` memiliki teks "1" di tengahnya dengan warna putih dan ukuran font 30.

Penting untuk dicatat bahwa dimensi setiap `Container` ditentukan menggunakan properti `height` dan `width`. Kontainer pertama dan kedua memiliki lebar 1000 dan tinggi 400, sementara kontainer sisanya memiliki tinggi dan lebar sebesar 200. Semua kontainer diatur dengan `alignment: Alignment.center` untuk menempatkan teks di tengah.

Secara keseluruhan, kode ini menciptakan tampilan antarmuka pengguna Flutter yang menggunakan `GridView` untuk menyusun enam kontainer dalam bentuk grid dengan warna

dan ukuran yang berbeda. Setiap kontainer berisi teks "1" di tengahnya, menciptakan tampilan yang teratur dan simetris.