MATERI

Hari-13-Flutter Styling

Flutter menggunakan *material design* sebagai *style default* untuk mengatur UI-nya, meskipun kita diberikan kebebasan untuk memodifikasi sesuai dengan keinginan. Maka pada bagian pertama ini, kita akan belajar bagaimana menggunakan *widget* **Scaffold**, serta *attribute* apa saja yang perlu kita ketahui.

Memulai dari awal untuk mengenal satu persatu *widget* Flutter, dimulai dari duet maut **MaterialApp** & **Scaffold**, Buka *file* `lib/main.dart`, lalu menjadi

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return null;
  }
}
```

Penjelasan: Semua *widget* yang biasanya digunakan beserta *style*-nya akan dirangkum oleh `material.dart` yang kita *import* pada *line* pertama. Ketika Flutter dijalankan, maka secara otomatis dia akan mengeksekusi *class* yang menjadi *value* dari `runApp()` yang berada di dalam *method* `main()`. *Class* `MyApp()` meng-*extend* **StatelessWidget**, dimana Flutter memiliki dua buah *widget* yang bernama **Stateless** dan **Stateful**. Perbedaan keduanya secara detail tidak akan kita bahas pada kesempatan kali ini, singkatnya *stateless* adalah *widget* yang tidak akan berubah setelah diinisiasi atau setelah dimuat. Sebaliknya berlaku untuk *stateful* adalah *widget* yang bisa berubah ketika mendapatkan perintah dari pengguna.

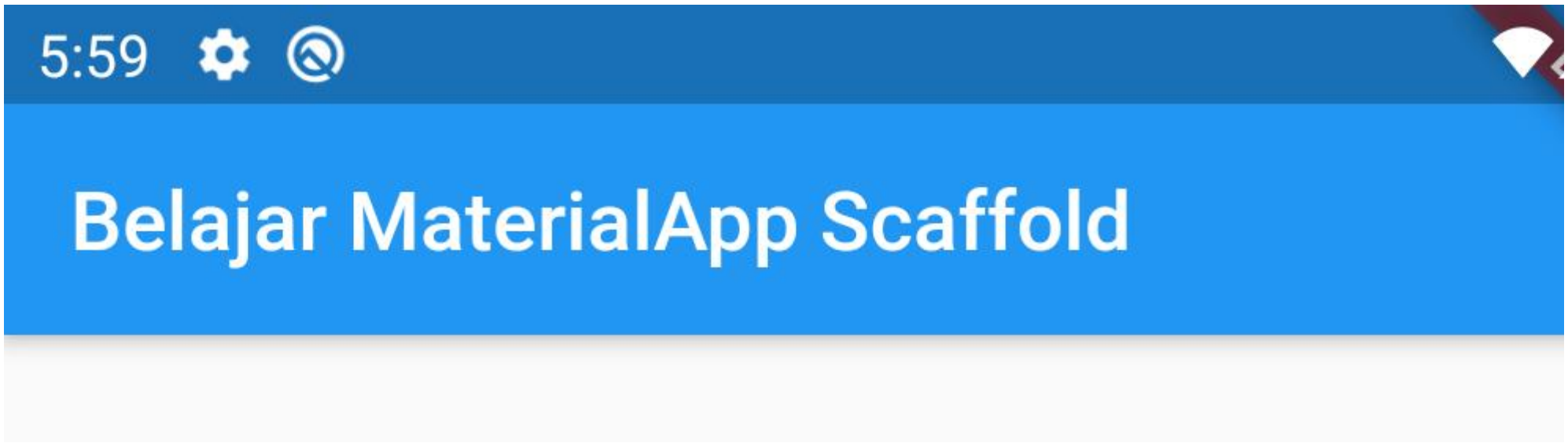
Tiba saatnya untuk menggunakan duet **MaterialApp** *widget* dan **Scaffold** *widget*, di dalam *file* yang sama, modifikasi menjadi

```
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("Belajar MaterialApp Scaffold"),),
      ),
    );
  }
}
```

Penjelasan: Ada beberapa bagian yang perlu dijelasan, diantaranya

1. `MaterialApp` adalah sebuah *parent* dimana yang diapitnya akan menerapkan *style* **material design**. Selain itu, dia juga memiliki *control* untuk mengatur *route, theme, supported locales*, dan lain sebagainya. Akan tetapi pada seri kali ini kita hanya akan membahas tentang *theme*-nya saja, adapun fungsi lainnya akan dirangkaikan dengan materi lainnya.
2. `Scaffold` memiliki peran untuk mengatur struktur visual *layout* dengan mengimplementasikan *material design*, dimana dia juga memiliki kemampuan untuk membuat *app bars, drawers, snack bars, bottom sheets dan lain sebagainya*.
3. `appBar` adalah salah satu properti yang dimiliki oleh **Scaffold** *widget* untuk membuat sebuah bar pada aplikasi, salah satu contohnya adalah menampilkan teks **Belajar MaterialApp Scaffold**

Adapun hasil yang akan kita peroleh seperti gambar di bawah ini



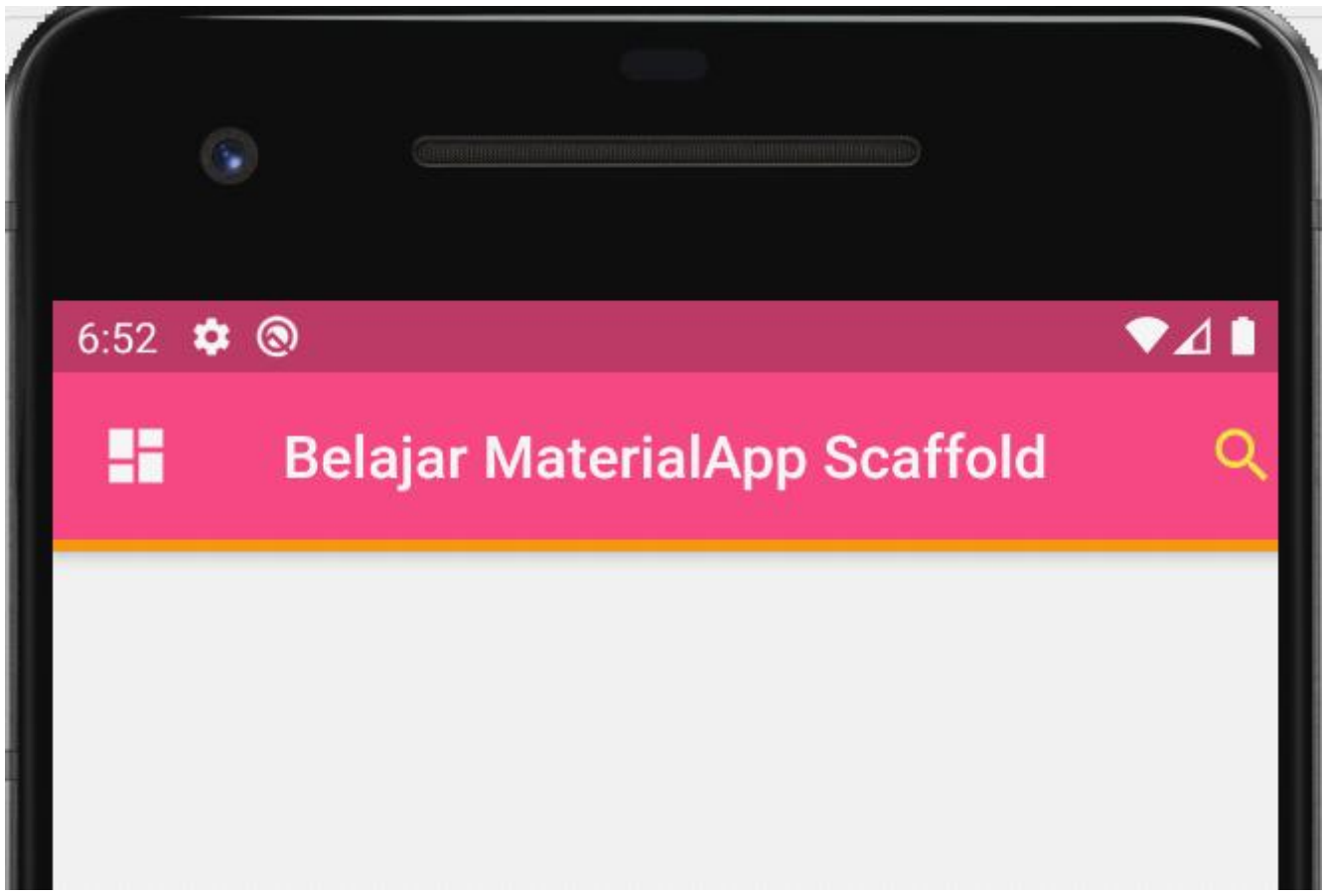
Menyelami lebih dalam lagi, kita akan menerapkan beberapa bagian dari *attribute* `appBar()`. Dengan *file* yang sama, modifikasi menjadi

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          leading: Icon(Icons.dashboard),
          title: Text("Belajar MaterialApp Scaffold"),
          actions: <Widget>[
            Icon(Icons.search),
            // Icon(Icons.find_in_page)
          ],
          actionsIconTheme: IconThemeData(color: Colors.yellow),
          backgroundColor: Colors.pinkAccent,
          bottom: PreferredSize(
            child: Container(
              color: Colors.orange,
              height: 4.0,
            ),
            preferredSize: Size.fromHeight(4.0)
          ),
          centerTitle: true,
        ),
      ),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Penjelasan: Berikut beberapa penjelasan dari penggunaan *attribute* di atas, diantaranya

1. **leading, title dan actions** adalah kesatuan yang membentuk urutan penempatan sesuai dengan urutan nama yang saya *bold* diawal. Ketiganya bisa menampilkan apa saja yang diinginkan, adapun `leading` dan `title` hanya bisa memuat satu *widget*, sedangkan `actions` bisa menampung lebih dari satu *widget* sesuai kebutuhan.
2. `actionsIconTheme` akan mengatur *theme* yang ingin diterapkan pada `actions` dengan catatan jika yang dimuat adalah *widget* Icon.
3. `bottom` sesuai namanya akan ditempatkan dibawah *appbar*, misalnya saja kita ingin memberikan efek garis pada bottom *appbar*.
4. `debugShowCheckedModeBanner` secara *default* bernilai **true** yang akan menampilkan *debug banner* pada pojok kanan atas, set *value*-nya jadi **false** untuk menghilangkan *debug banner* tersebut.

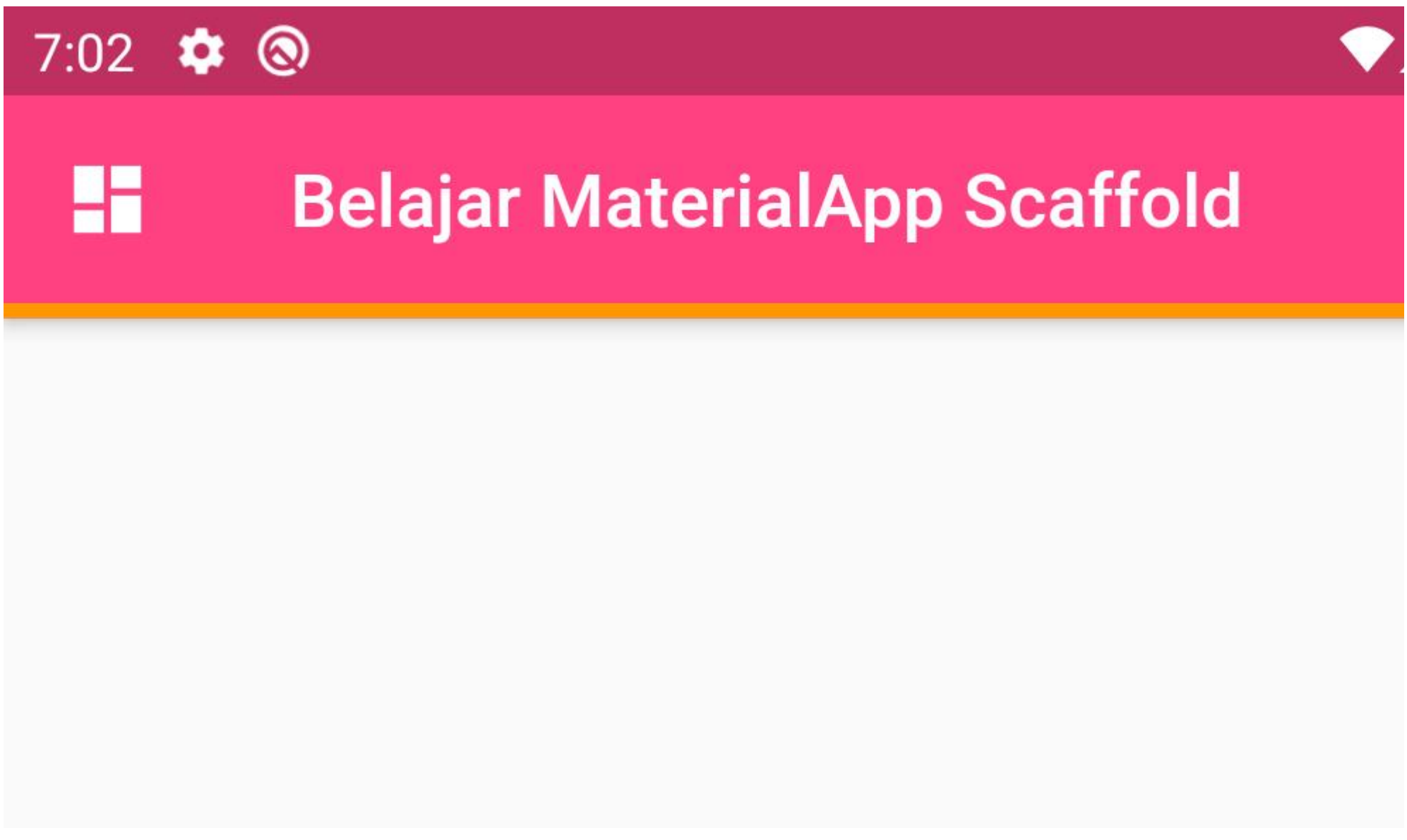
Adapun hasil yang akan kita peroleh seperti gambar dibawah ini

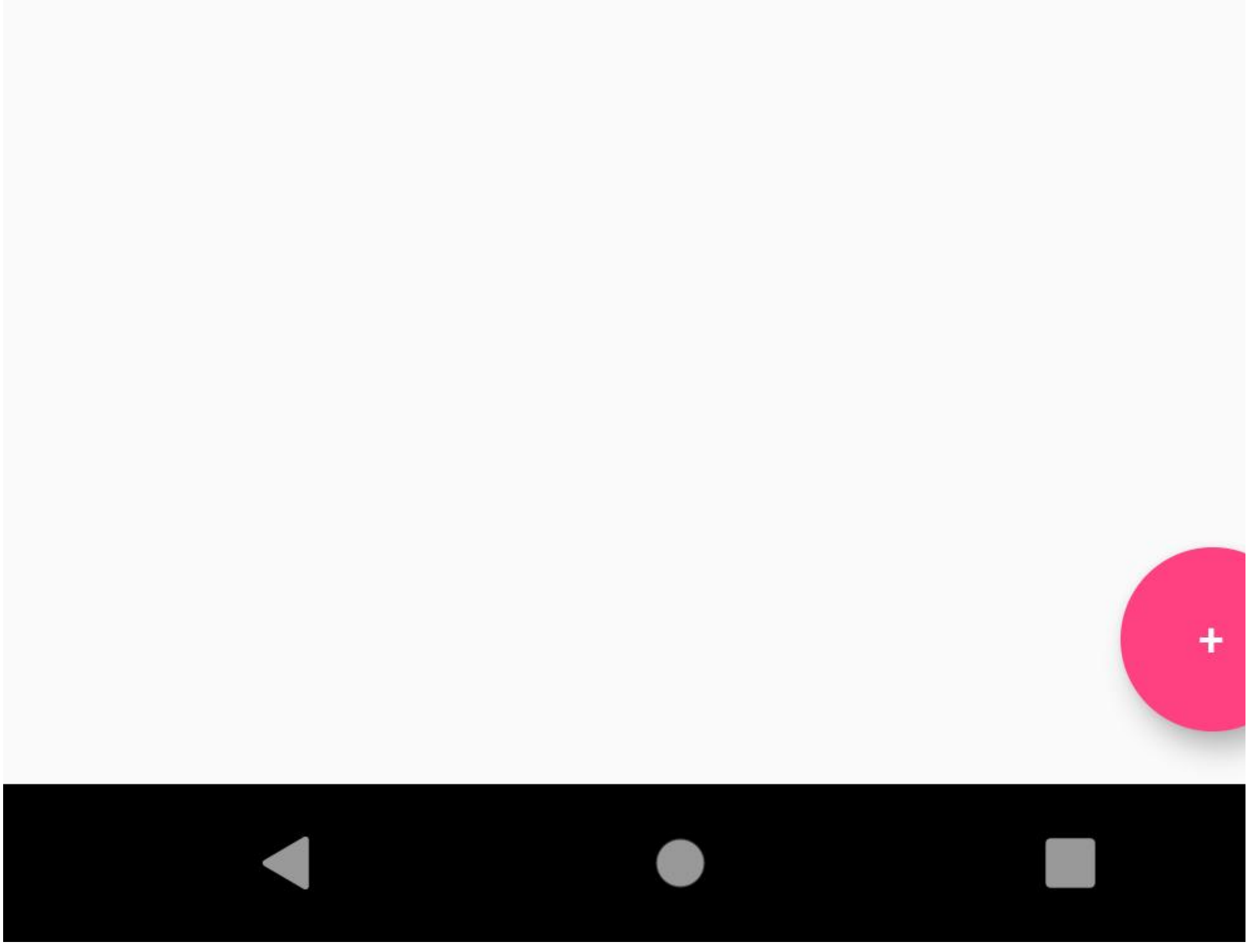


```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          leading: Icon(Icons.dashboard),
          title: Text("Belajar MaterialApp Scaffold"),
          actions: <Widget>[
            Icon(Icons.search),
            // Icon(Icons.find_in_page)
          ],
          actionsIconTheme: IconThemeData(color: Colors.yellow),
          backgroundColor: Colors.pinkAccent,
          bottom: PreferredSize(
            child: Container(
              color: Colors.orange,
              height: 4.0,
            ),
            preferredSize: Size.fromHeight(4.0)
          ),
          centerTitle: true,
        ),

        //PERUBAHAN BARU
        floatingActionButton: FloatingActionButton(
          backgroundColor: Colors.pinkAccent,
          child: Text('+'),
          onPressed: () {},
        ),
        body: null,
        //PERUBAHAN BARU
      ),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Penjelasan: `floatingActionButton` digunakan untuk membuat tombol melayang pada pojok kanan bawah, sedangkan `body` memuat *content* yang akan ditampilkan pada *frame* putih yang sedang kosong saat ini.

Adapun tampilan akhir yang akan kita peroleh terlihat seperti gambar di bawah ini.





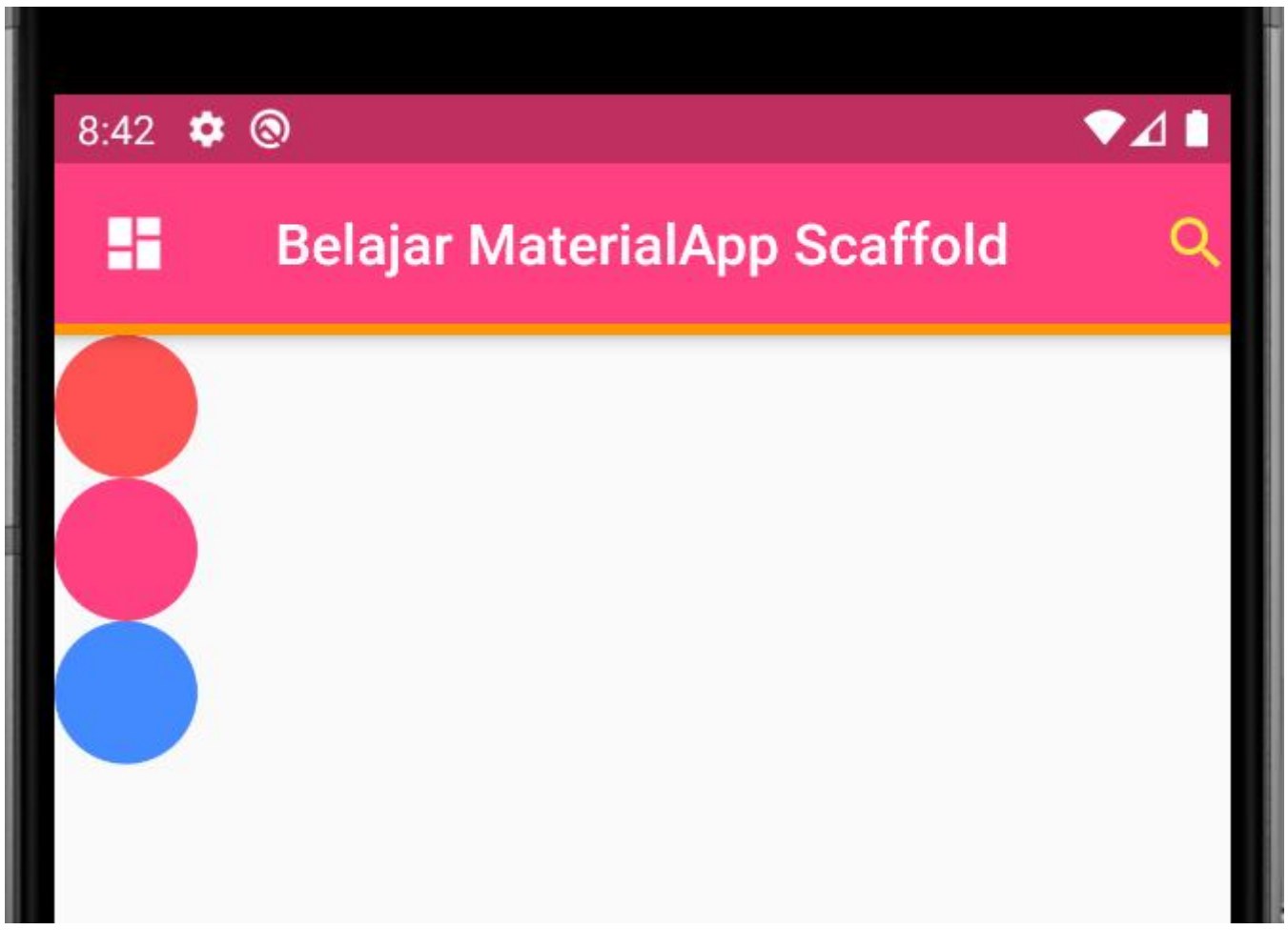
Static User Interface (View-Group-Widget)

a. Pasangan Row-Column

Jika anda berlatar belakang dari *classic web developer*, maka menggambar **row-column** bisa diasumsikan sebagai *table* yang memiliki *row* dan *column*. Sebab dahulu kala, mengatur tata letak bisa memanfaatkan *tag table* pada *html*. Jadi **Row** untuk mengatur *widget* agar tersusun secara *horizontal* (***red***: dari kiri ke kanan), sedangkan **Column** berlaku sebaliknya, yakni secara *vertical* (***red***: dari atas ke bawah).

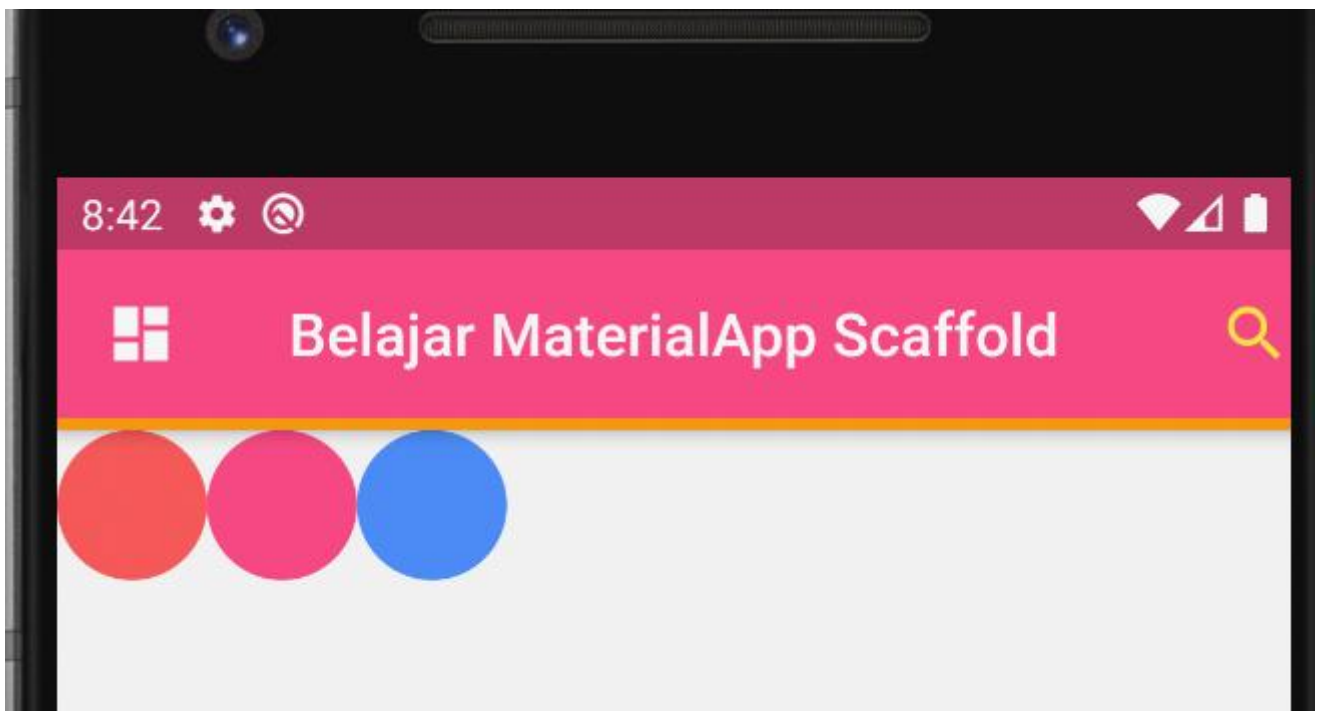
```
body: Column(children: <Widget>[
  Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
  Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
  Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.blueAccent, shape: BoxShape.circle),),
],),
```

Penjelasan: ***Column*** bisa menampung lebih dari 1 *widget*, sehingga kita bisa memasukkan *widget* lainnya di dalam *children*-nya sesuka hati. Seperti yang disinggung di awal bahwa *column* akan menyusun *widget* yang ada di dalamnya dari atas ke bawah.



Penerapa **Row** sama saja dengan *Column*, bisa menampung lebih dari 1 *widget* di dalamnya. Hanya saja berbeda orientasi penggunaannya saja. Buka *file* yang sama dan modifikasi menjadi.

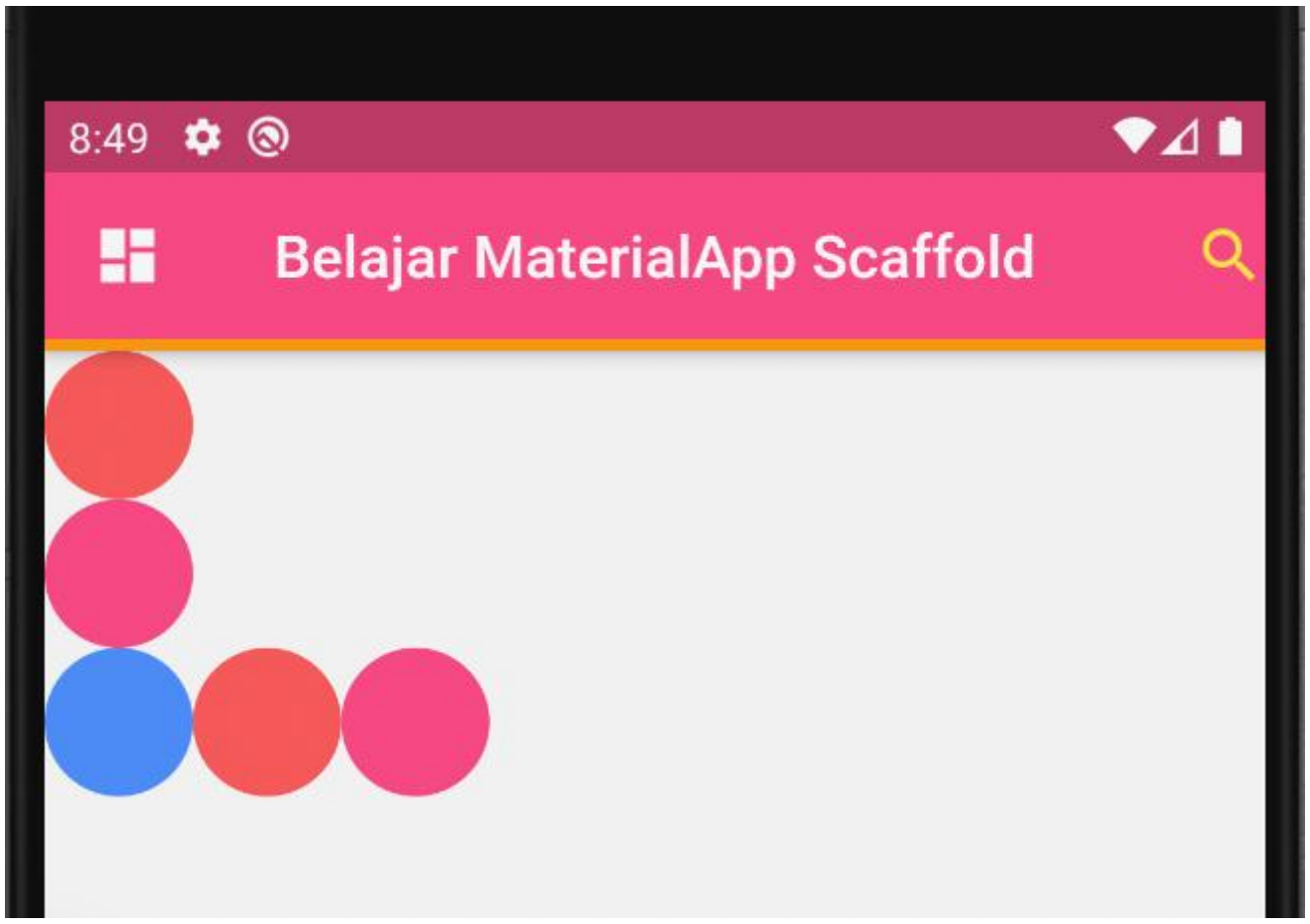
```
body: Row(
  children: <Widget>[
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.blueAccent, shape: BoxShape.circle),),
  ],
),
```



Selain bekerja secara individu, kita juga bisa memadukan antara *row* dan *column* dalam satu keadaan. Misalnya saja *case* yang diinginkan adalah 3 lingkaran membentuk urutan kebawah dan pada lingkaran terakhir urutannya kesamping.

```
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
    Row(
      children: <Widget>[
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.blueAccent, shape: BoxShape.circle),),
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
      ],
    )
  ],
),
```

Penjelasan: Perpaduan antara **Row & Column** di Flutter, dimana Row bisa dimasukkan ke dalam Column, begitpun sebaliknya. Ada hal baru yang kita pelajari yakni penggunaan `crossAxisAligment`, dimana *attribute* ini berfungsi untuk mengatur *aligment* dari Row maupun Column. Adapun sifatnya selalu berlawanan dengan orientasi dari Row & Column tersebut. Misalnya saja, Column memiliki sifat penerapan secara *vertical*, maka `crossAxisAligment` akan mengaturnya secara *horizontal*, begitupun dengan row.

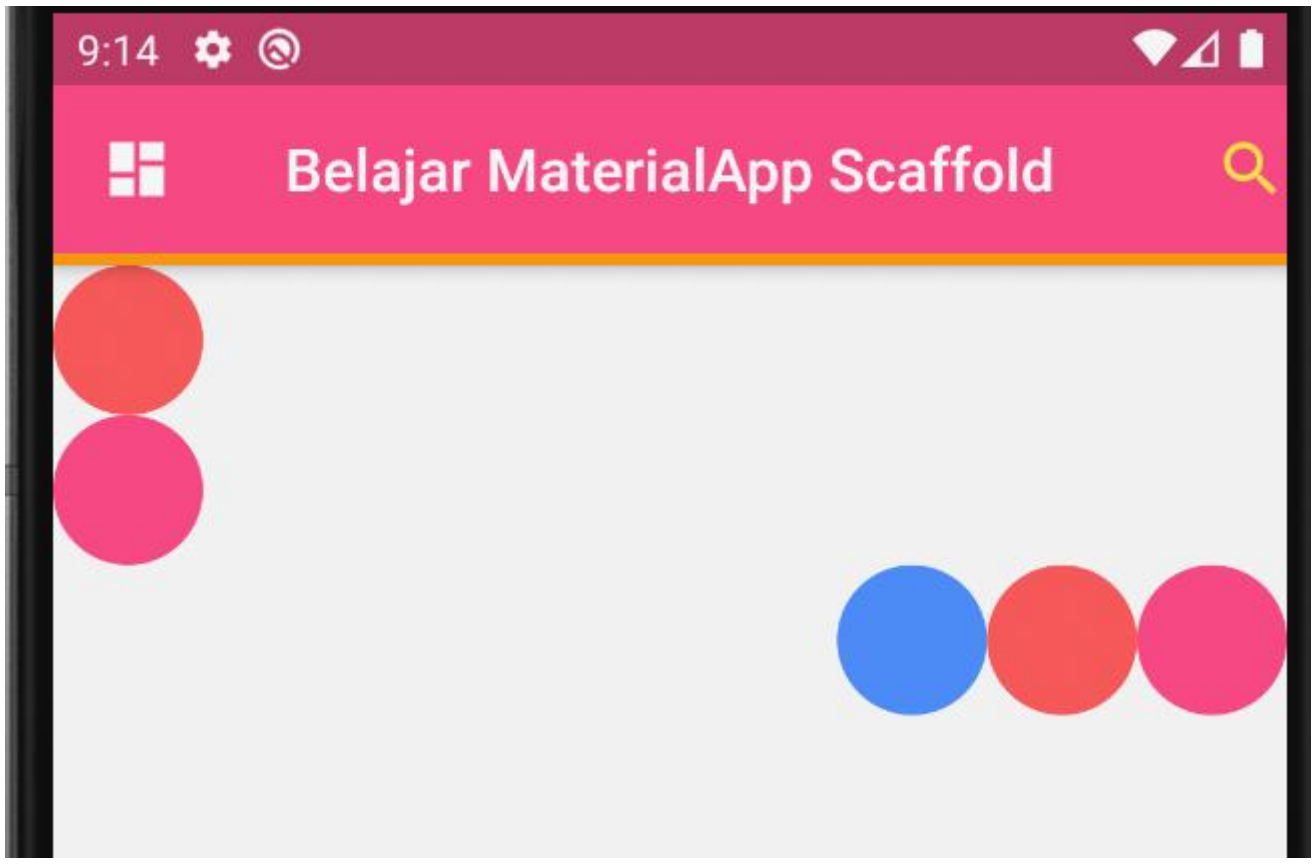


Selain `crossAxisAligment`, ada *attribute* lain yang sama pentingnya untuk memaksimalkan fungsi Row & Column. Misalnya saja *schema* yang diinginkan adalah kita ingin mengatur 3 lingkaran paling bawah akan disimpan pada sudut kanan *screen*, sedangkan dua lingkaran lainnya tetap pada posisinya.

Pada *schema* diatas, kita akan memanfaatkan peran `mainAxisAligment`, tambahkan *code* berikut sebagai attribute *row*

```
body: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: <Widget>[
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
    Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
    Row(

      //TAMBAHKAN CODE INI
      mainAxisAlignment: MainAxisAlignment.end,
      //TAMBAHKAN CODE INI

      children: <Widget>[
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.blueAccent, shape: BoxShape.circle),),
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.redAccent, shape: BoxShape.circle),),
        Container(width: 50, height: 50, decoration: BoxDecoration(color: Colors.pinkAccent, shape: BoxShape.circle),),
      ],
    )
  ],
),
```

Penjelasan: `mainAxisAligment` memiliki peran yang sama dengan `crossAxisAligment`, yakni untuk mengatur *aligment* dari *widget* yang ada di dalamnya. Hanya saja `mainAxisAligment` memiliki sifat yang sejalan dengan *widget*-nya. Misalnya **mainAxisAligment** pada Row akan mengatur *aligment* secara *horizontal*, sehingga jika *value*-nya adalah `.end` maka *widget*-nya akan ditempatkan diakhir *screen* secara *horizontal*.



Mengatur Padding

Padding adalah jarak atau ruang yang berada di antara border dan konten. Sedangkan margin adalah jarak atau ruang yang berada atau dihitung dari border ke arah luar atau jarak antara border dan konten yang berada di luar border.

Cara pertama kita bisa menggunakan Widget Padding, dimana widget ini akan mengatur padding untuk widget yang ada di dalamnya.

```
Padding(
  padding: EdgeInsets.all(8.0),
  child: const Card(child: Text('Hello World!')),
)
```

Pada contoh di atas akan menghasilkan padding di semua sisi yaitu atas, kanan, bawah dan kiri. Contoh lain bisa seperti berikut.

```
Padding(
  padding: EdgeInsets.fromLTRB(8.0, 5.0, 5.0, 4.0),
  child: const Card(child: Text('Hello World!')),
)
```

Pada contoh di atas akan menghasilkan padding di semua sisi dengan urutan kiri (L), atas (T), kanan (R), bawah (B).

Contoh lain untuk mengatur pada sisi tertentu saja bisa seperti berikut.

```
Padding(
  padding: EdgeInsets.only(left:8.0, right:8.0),
  child: const Card(child: Text('Hello World!')),
)
```

Pada contoh di atas akan menghasilkan padding hanya pada sisi yang kita sebutkan saja. Seperti pada contoh saya hanya menyebut left dan right. Selain menggunakan widget Padding kita juga bisa menggunakan Container sebagai berikut.

```
Container(
  padding: EdgeInsets.only(left:8.0, right:8.0),
  child: const Card(child: Text('Hello World!')),
)
```

Mengatur Margin

Untuk margin kita bisa menggunakannya pada widget container dikarenakan tidak ada widget khusus untuk margin seperti yang dimiliki padding. Sehingga penggunaannya akan seperti berikut.

```
Container(
  margin: EdgeInsets.all(8.0),
  child: const Card(child: Text('Hello World!')),
)
```

Dynamic User Interface (ListView, GridView)

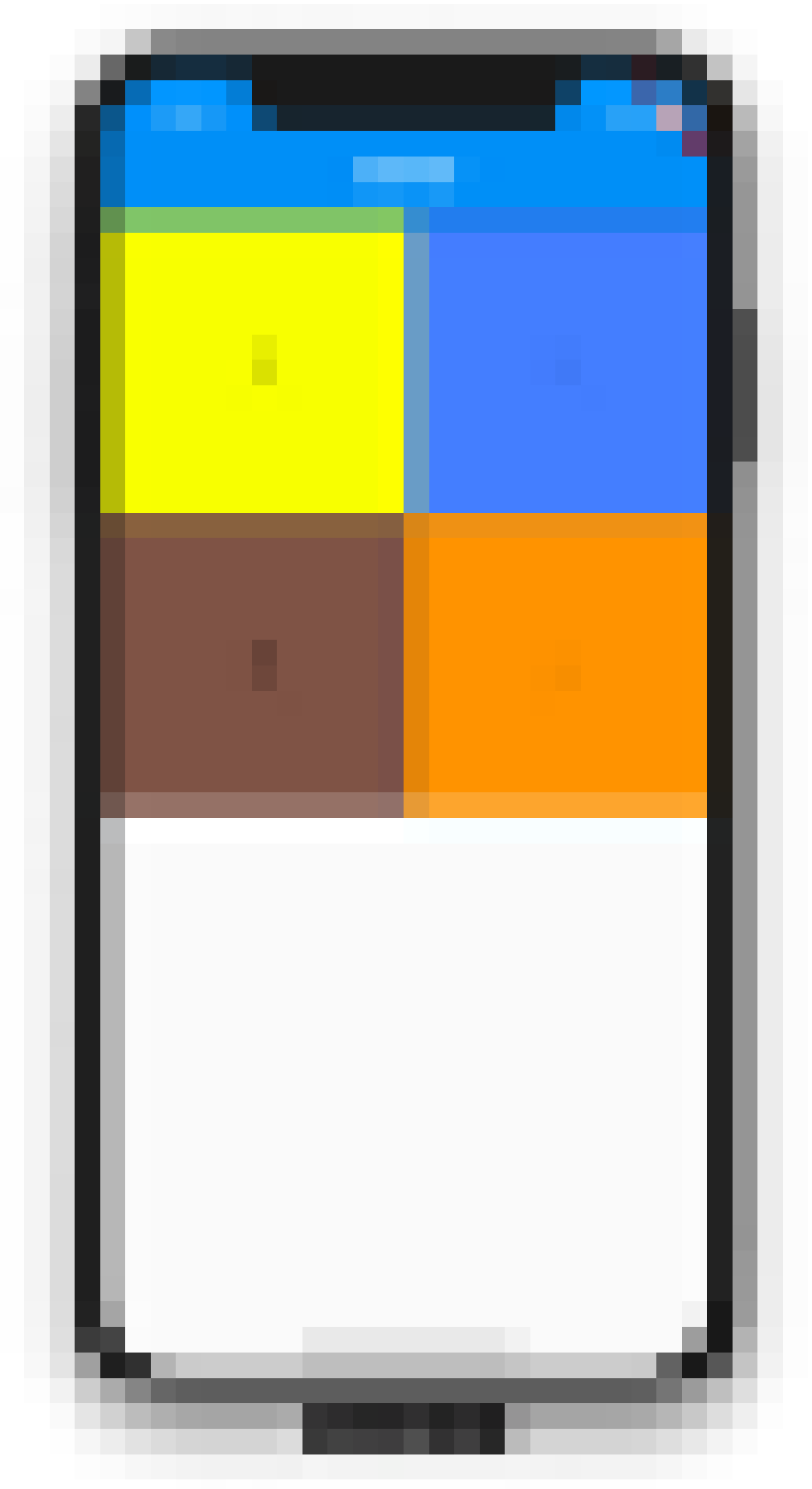
GridView

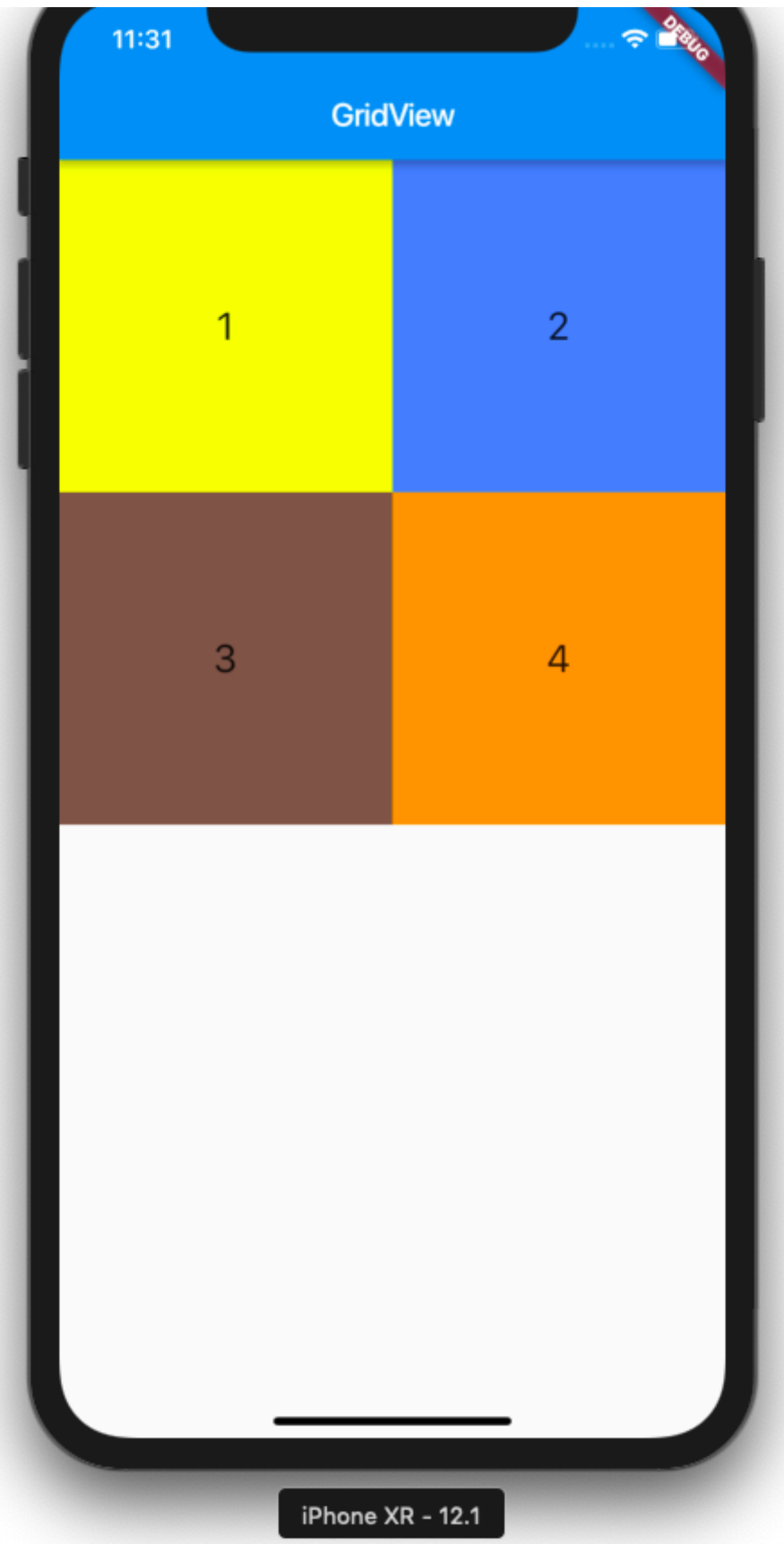
Di Flutter untuk membuat layout secara GridView kita bisa menggunakan widget `GridView`. Berdasarkan dokumentasinya dituliskan `GridView` adalah sebagai berikut.

Jadi, intinya adalah hampir sama seperti pada Android juga dimana, widget `GridView` memiliki konsep yang sama yaitu layout yang disusun secara 2 dimensi baik secara horizontal maupun vertikal.

Penggunaan Dasar

Sebagai penggunaan dasarnya kita akan coba buat tampilan seperti berikut.





Penggunaan dasar GridView

Berikut source code-nya.

```
    return Scaffold(
      appBar: AppBar(
        title: Text(
          "GridView",
          style: TextStyle(color: Colors.white),
        ),
      ),
      body: GridView.count(
        crossAxisCount: 2,
        children: <Widget>[
          Container(
            color: Colors.yellowAccent,
            child: Center(
              child: Text("1", style: TextStyle(fontSize: 24.0),),
            ),
          ),
          Container(
            color: Colors.blueAccent,
            child: Center(
              child: Text("2", style: TextStyle(fontSize: 24.0),),
            ),
          ),
          Container(
            color: Colors.brown,
            child: Center(
              child: Text("3", style: TextStyle(fontSize: 24.0),),
            ),
          ),
          Container(
            color: Colors.orange,
            child: Center(
              child: Text("4", style: TextStyle(fontSize: 24.0),),
            ),
          ),
        ],
      ),
    );
  }
}
```

Pada kode diatas, kita ada deklarasikan property nilai `crossAxisCount` sebanyak 2 dimana ini artinya kita mendeklarasikan berapa jumlah kolom pada GridView tersebut.

Video Documentasi Referensi

https://www.youtube.com/watch?v=bLOtZDTm4H8

ListView :

ini adalah bentuk default dari sebuah ListView class. Dengan ListView maka children atau widget yang ada di dalamnya akan menjadi scrollable (bisa di scroll). Penggunaan default ListView ini hanya untuk widget yang bersifat statis. Statis yang dimaksud bukan untuk isi kontennya melainkan lebih kepada jumlah widget di dalamnya.

Penggunaan default ListView :

Contoh kasus misalkan kita ingin membuat sebuah halaman detail aplikasi baca berita. Dimana biasanya untuk halaman tesebut umumnya memiliki item judul dan deskripsi. Untuk panjang deskripsi pada sebuah berita beragam dan bisa sangat panjang.

Dari contoh kasus di atas kita dapat simpulkan bahwa untuk jumlah item widgetnya sudah pasti (fix) yaitu hanya dua widget saja (title dan deskripsi). sedangkan untuk ukuran dari tinggi widget deskripsi bersifat dinamis sehingga ada kemungkinan melebihi ukuran tinggi layar. Pada kasus seperti ini maka ListWidget adalah yang paling aman. contoh kodenya seperti dibawah ini

```
      padding: EdgeInsets.all(15),
      child: Text('Flutter Widget: Penggunaan ListView Class',
          style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold)
      ),
    ),
    Container(
        padding: EdgeInsets.all(15),
        child: Text(
            '''Lorem Ipsum adalah contoh teks atau ...''',
            style: TextStyle(fontSize: 16)
        ),
    ),
  ]
)
```

Penjelasan : widget ListView memiliki 2 buah container dengan tiap containernya memiliki text widget. Untuk text sengaja tidak disertakan secara penuh agar kode mudah dibaca. untuk contoh text nya dapat dilihat di https://id.lipsum.com/

Seperti yang dapat kita lihat pada gambar diatas bahwa apabila text melewati ukuran layar maka akan otomatis menjadi scrollable.

video referensi documentasi :

https://www.youtube.com/watch?v=KJpkjHGiI5A

atau materi video dapat di rangkum di video ini

https://www.youtube.com/watch?v=PFkCSLnKKx0

Dan untuk lebih memahami cara kerja dari row, column, margin, padding sobat sanber dapat langsung mengerjakan soal tugas yang ada pada tugas-13-styling

sumber :

https://medium.com/nusanet/flutter-gridview-bad48c1f216c

https://belajarflutter.com/3-cara-benar-menggunakan-listview-pada-flutter/

Rating - Feedback

Berikan Rating pada posting ini:

Berikan kritik dan saran..

Submit