



ITB Stikom
Ambon

PENGEMBANGAN APLIKASI ANDROID

#5 Flutter Widget Design UI

By Chairil Ali, S.Kom

Setelah sebelumnya kita mempelajari bagaimana flutter bekerja bersama dart dalam membuat sebuah tampilan pada layar, yuk kita belajar tentang widget widget yang akan sering dipakai dalam proses development UI dalam pengembangan aplikasi.

Widget catalog

UI > Widgets

Create beautiful apps faster with Flutter's collection of visual, structural, platform, and interactive widgets. In addition to browsing widgets by category, you can also see all the widgets in the [widget index](#).

| | | |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Accessibility Make your app accessible. Visit | Animation and Motion Bring animations to your app. Visit | Assets, Images, and Icons Manage assets, display images, and show icons. Visit |
| Async Async patterns to your Flutter application. Visit | Basics Widgets you absolutely need to know before building your first Flutter app. Visit | Cupertino (iOS-style widgets) Beautiful and high-fidelity widgets for current iOS design language. Visit |
| Input Take user input in addition to input widgets in Material Components and Cupertino. Visit | Interaction Models Respond to touch events and route users to different views. Visit | Layout Arrange other widgets columns, rows, grids, and many other layouts. Visit |

Materi pada Modul Ini

1. Basic widget
2. Layout
3. Form
4. Navigator



**Dalam basic widget kali ini, kita akan
mengenal:**

- Scaffold
- AppBar
- Container
- Text
- Button
- Icon
- Images
- CircleAvatar

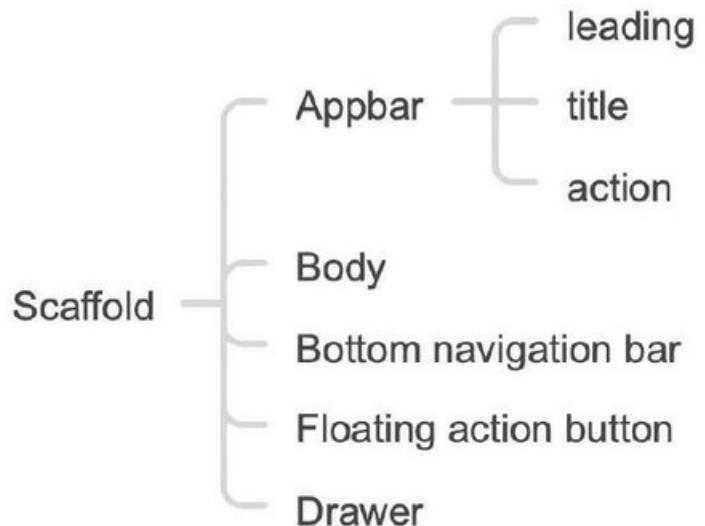
Scaffold

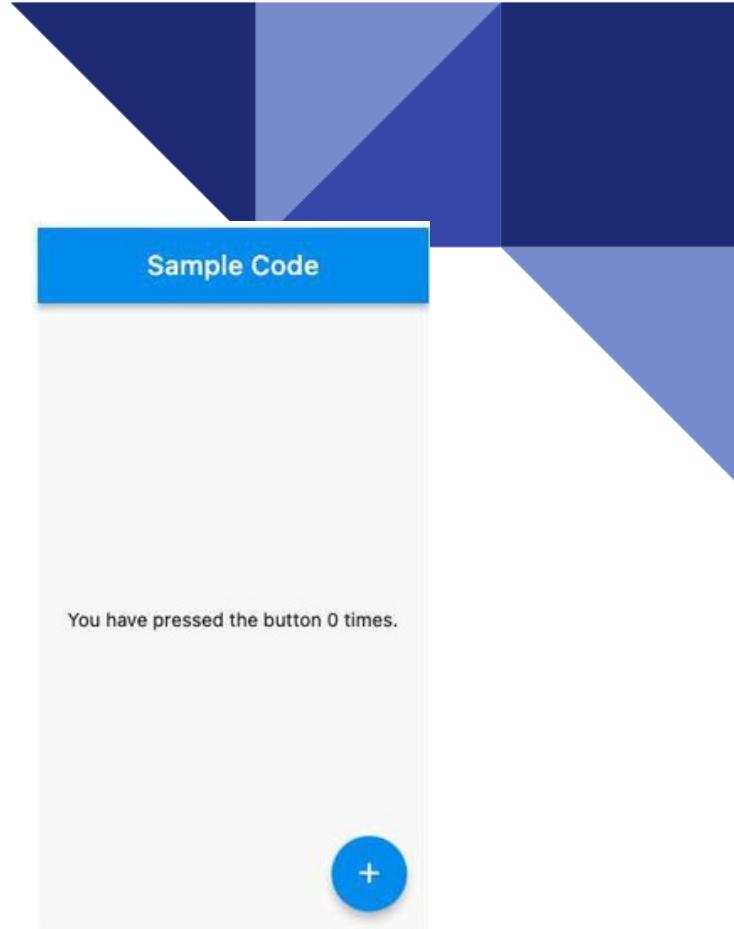
Scaffold adalah widget yang mengimplementasi struktur tata letak dari design material.

Lalu apa itu design material ?

Material Design adalah sebuah gaya desain terbaru dari Google yang memiliki prinsip bahwa desain itu harus seperti sebuah kertas pada kenyataannya.

Terus apa bedanya dengan kertas? Material design merupakan kertas ajaib yang bisa membesar, mengecil, membelah diri, berubah warna, dan bertransformasi menjadi berbagai bentuk.





Sample Code

Disamping ini adalah contoh tampilan layar dari widget scaffold, dimana dia mempunyai appbar di paling atas, yang berwarna biru dengan title Sample Code.

Lalu ada body yang berisi text dan berada di bagian tengah dengan wording ‘You have pressed the button 0 times.’

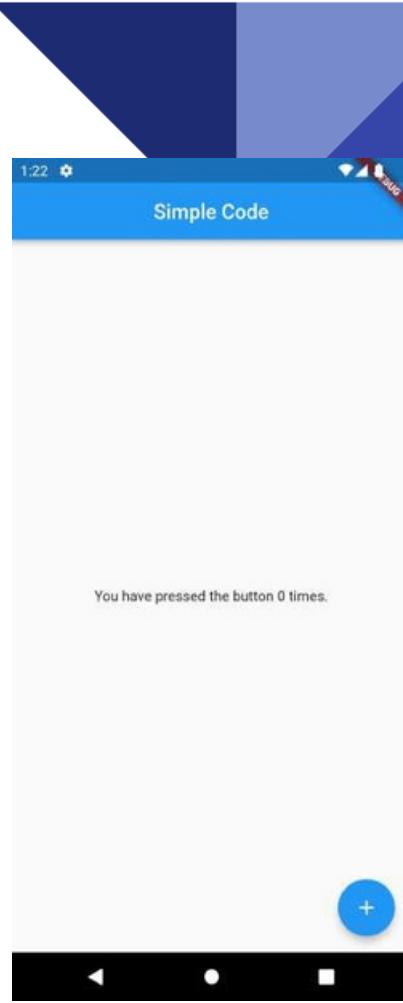
Ialu ada floatingActionButton di pojok kanan bawah yang berisi icon add.

You have pressed the button 0 times.



Scaffold -> appBar

Appbar selalu ditempatkan di bagian atas layar.
Jika scaffold mempunyai drawer, maka button icon hamburger akan ditambahkan secara otomatis untuk menangani open close drawer.



Dalam potongan code disamping, Appbar berada di baris ke 14 dengan saya set untuk named parameteranya yang pertama centerTitle ini supaya title nya ada dibagian tengah appbar, lalu yang kedua parameter title yang ini valuenya adalah widget, sehingga kita perlu memberinya paling simple adalah widget text.

Saya beri keywords const karena didalam text itu valuenya sudah fix menggunakan string text 'Simple Code'. Jadi widget ini tidak akan direbuild lagi ketika widget ini ditrigger untuk rebuild melalui mekanisme setState.

```
13   return Scaffold(  
14     appBar: AppBar(  
15       centerTitle: true,  
16       title: const Text("Simple Code"),  
17     ), // AppBar  
18     body: const Center(  
19       child: Text('You have pressed the button 0 times.'),  
20     ), // Center  
21     floatingActionButton: FloatingActionButton(  
22       onPressed: () {  
23         openForm();  
24       },  
25       child: const Icon(Icons.add),  
26     ), // FloatingActionButton  
27   ); // Scaffold  
28 }
```

Scaffold -> body

Di parameter body ini nantilah yang akan banyak kita tulis kodennya. Karena disini nantilah widget yang akan tampil hampir 99% di layar kita. Body ini butuh kembalian widget, widget disini bisa di isi oleh widget yang bisa menampung widget lain, seperti Column, ListView, Stack, Container, dll. Ini nanti akan kita bahas di layout widget. Untuk contoh case disamping yang ada di baris 18, body saya isi dengan widget Center yang isinya terdapat widget text dengan value 'You have pressed the button 0 times.'

```
13 |     return Scaffold(
14 |       appBar: AppBar(
15 |         centerTitle: true,
16 |         title: const Text("Simple Code"),
17 |       ), // AppBar
18 |       body: const Center(
19 |         child: Text('You have pressed the button 0 times.'),
20 |       ), // Center
21 |       floatingActionButton: FloatingActionButton(
22 |         onPressed: () {
23 |           openForm();
24 |         },
25 |         child: const Icon(Icons.add),
26 |       ), // FloatingActionButton
27 |     ); // Scaffold
28 | }
```

Scaffold -> floatingActionButton

Parameter floatActionButton juga sudah disediakan oleh flutter yang fungsinya membuat button yang floating di sisi bawah kanan, ini untuk posisi bisa kita geser2 dengan config yang dia sediakan. Seperti dibaris 21 ini, floatingactionbutton dia membutuhkan kembalian widget dan widgetnya pakai widget floatingactionbutton dengan child sebuah icon Icons.add. Nanti hasilnya seperti di tampilan yang sudah saya share di slide sebelumnya.

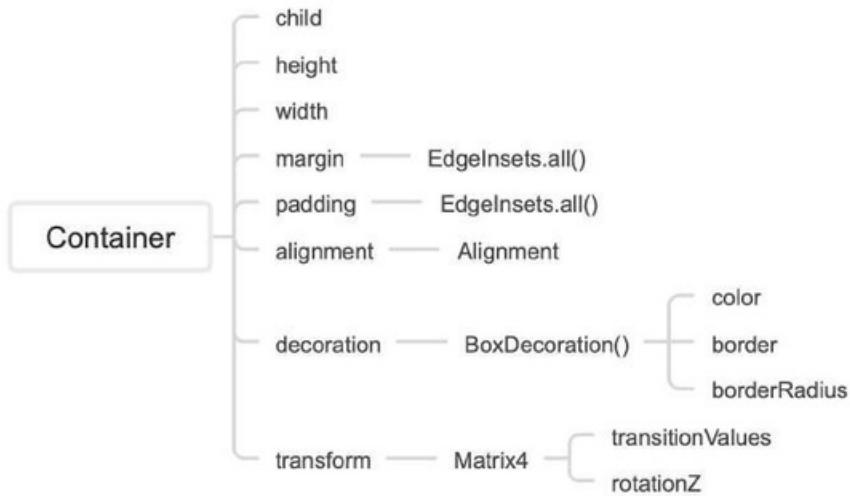
```
13 |     return Scaffold(
14 |       appBar: AppBar(
15 |         centerTitle: true,
16 |         title: const Text("Simple Code"),
17 |       ), // AppBar
18 |       body: const Center(
19 |         child: Text('You have pressed the button 0 times.'),
20 |       ), // Center
21 |       floatingActionButton: FloatingActionButton(
22 |         onPressed: () {
23 |           openForm();
24 |         },
25 |         child: const Icon(Icons.add),
26 |       ), // FloatingActionButton
27 |     ); // Scaffold
28 |   }
```

Container

Widget Container ini setara dengan tag <div></div> di HTML

ini adalah wadah yang dapat kita gunakan untuk menyesuaikan bentuk, posisi, ukuran, pewarnaan dan masih banyak lagi. Wadah ini sangat banyak digunakan karena menyelesaikan banyak kasus seperti membuat border melengkung atau bekerja dengan bentuk bentuk tertentu.

Di samping adalah mindmap properties yang dimiliki oleh widget container.



Seperti contoh disamping, widget container kita masukan ke dalam body punya widget scaffold.

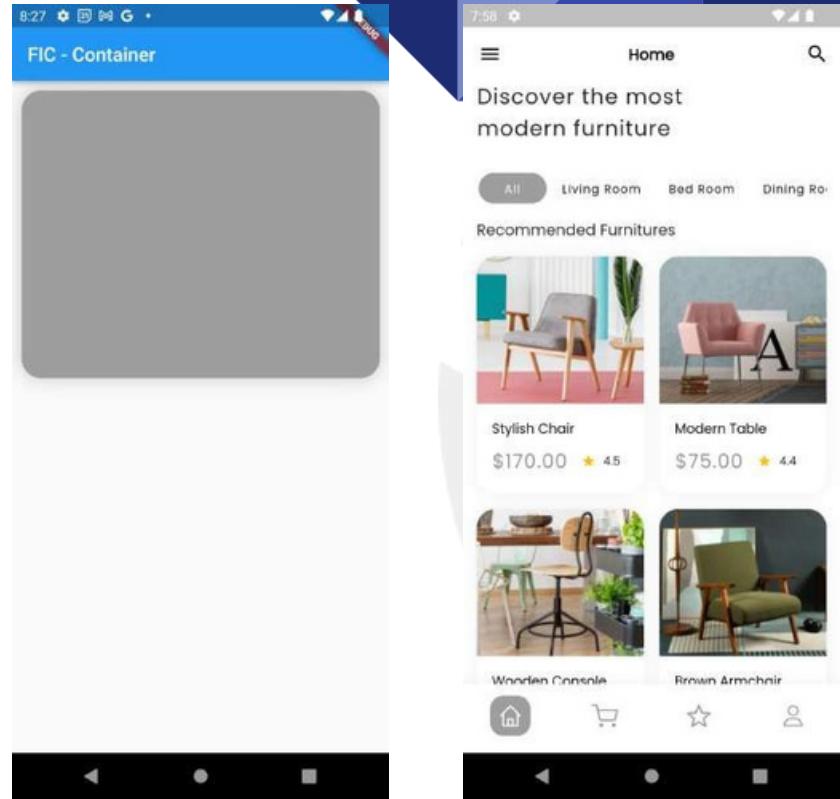
Di dalam nya kita dapat memberi nilai tinggi, lebar, jarak antara border, decoration dan dia juga punya child yang hanya bisa 1 anak widget. Namun tenang saja, anaknya ini bisa diisi dengan column atau row jadi tetap di dalamnya bisa berisi banyak widget. Contoh

disamping ceritanya kita akan membuat wadah atau container dengan tinggi 300px, lebar selebar layar, margin dengan tepi layar itu 10px, dan mempunya dekorasi warna grey, border radius yang melengkung dengan lengkungan 20px, lalu ada bayang2 juga.

Hasilnya bisa dilihat di slide selanjutnya

```
12 |   body: Container(
13 |     height: 300,
14 |     width: double.infinity,
15 |     margin: const EdgeInsets.all(10.0),
16 |     decoration: BoxDecoration(
17 |       color: Colors.grey,
18 |       borderRadius: BorderRadius.circular(20),
19 |       boxShadow: [
20 |         BoxShadow(
21 |           color: Colors.black.withOpacity(0.2),
22 |           offset: Offset.zero,
23 |           blurRadius: 15.0,
24 |         ) // BoxShadow
25 |       ],
26 |     ), // BoxDecoration
27 |   ), // Container
```

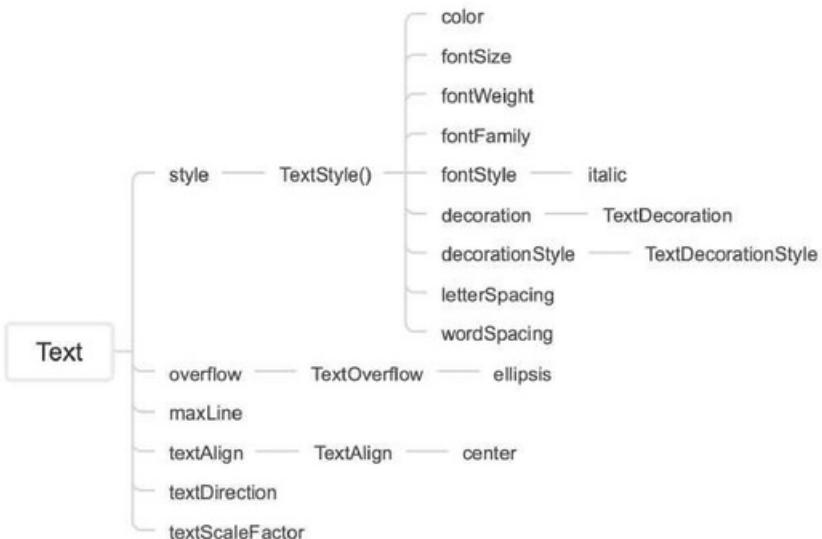
Hasilnya yang sebelah kiri itu nanti dapat kita implement kedalam latihan slicing dan bisa menjadi card yang menampung gambar dan keterangan title harga rating.



Text

Widget teks digunakan untuk menampilkan teks di layar. Ini sangat fleksibel buat kita dalam memanipulasi text seperti mengubah warna font, menggunakan asset font atau paket dari Google Font, menambah ukuran font, memberi garis bawah pada text dan masih banyak lainnya.

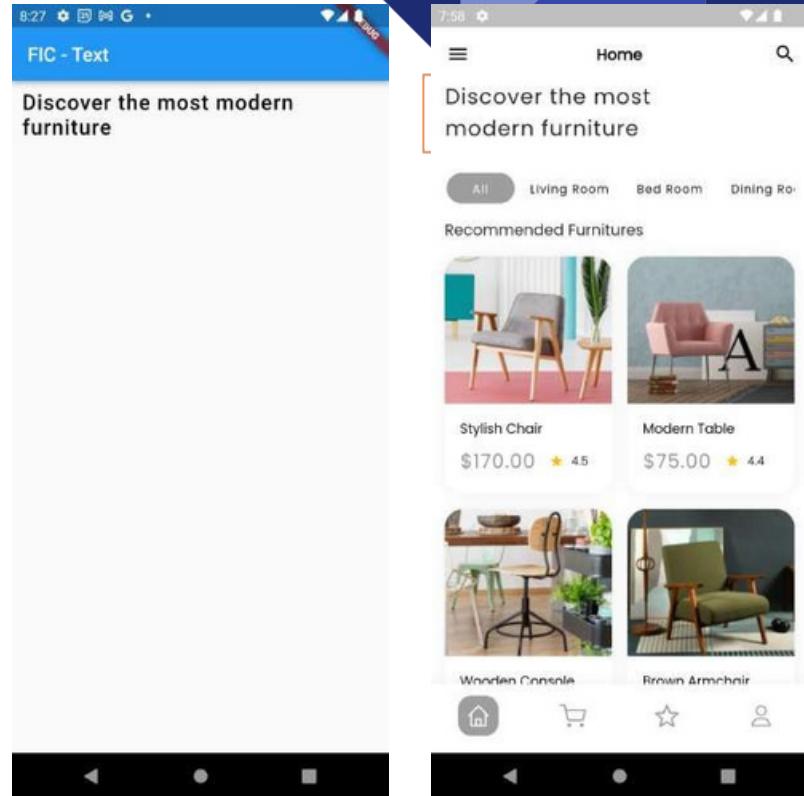
Di samping adalah mindmap yang dimiliki oleh widget text.



Dalam contoh kode disamping cerita nya kita ingin menampilkan text dengan style warnanya hitam, ukuran font nya 22px, ketebalan huruf nya medium atau tengah-tengah. Dan letter spasi nya 1px. Hasilnya dapat kita lihat di slide selanjutnya

```
13   |   body: Container(
14   |   |   padding: const EdgeInsets.all(10.0),
15   |   |   child: Column(
16   |   |   |   children: const [
17   |   |   |   |   Text(
18   |   |   |   |   |   'Discover the most modern furniture',
19   |   |   |   |   |   style: TextStyle(
20   |   |   |   |   |   |   color: Colors.black,
21   |   |   |   |   |   |   fontSize: 22.0,
22   |   |   |   |   |   |   fontWeight: FontWeight.w500,
23   |   |   |   |   |   |   letterSpacing: 1,
24   |   |   |   |   |   ), // TextStyle
25   |   |   |   |   |   ), // Text
26   |   |   |   |   ], // Column
27   |   |   |   ), // Container
```

Bila kita running hasilnya seperti gambar sebelah kiri.
Contoh ini nanti dapat kita implementasi kedalam sample latihan slicing untuk tulisan yang ada dipaling atas sebagai tagline product.

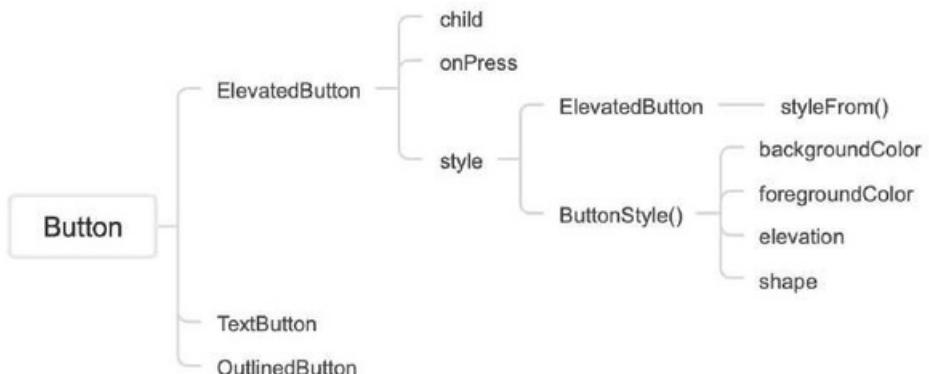


Button

Tombol adalah salah satu bagian dasar dalam aplikasi apapun karena tombol merupakan cara paling intuitif untuk memberi tahu pengguna bahwa saat ditekan, akan terjadi sesuatu.

Flutter memiliki beberapa jenis tombol yang sudah disediakan ya itu ElevatedButton, TextButton, OutlinedButton.

Disamping adalah mindmap untuk widget button untuk flutter versi terakhir. (3.3.x)



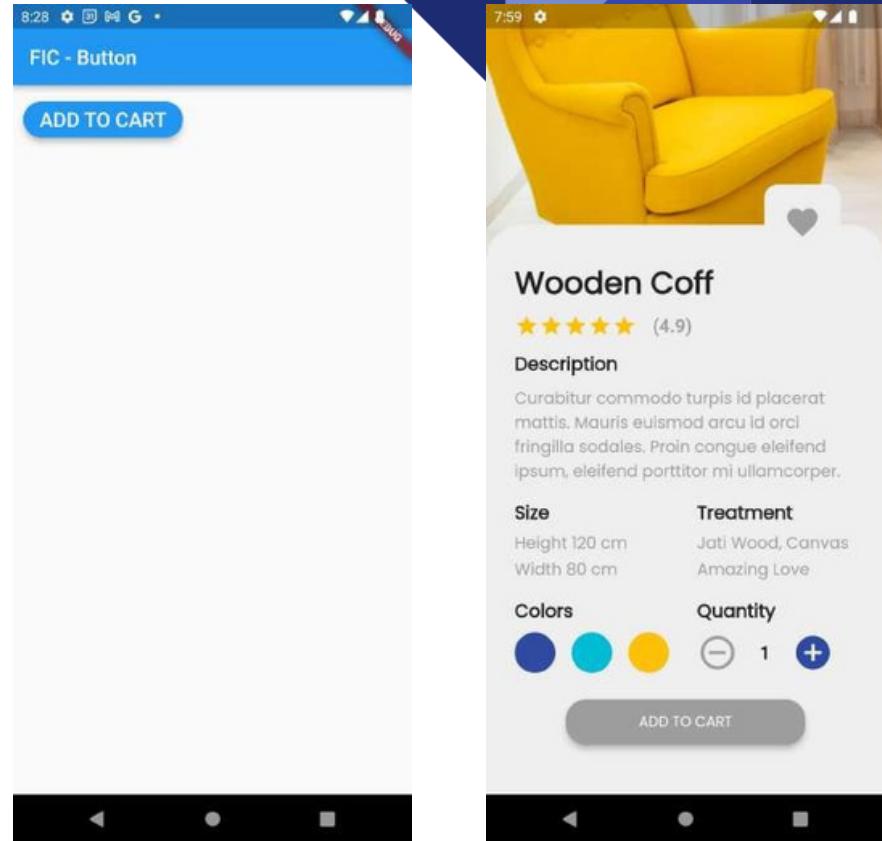
Dalam contoh disamping, ceritanya kita akan membuat sebuah button dengan style background biru, dan bentuknya itu akan ada lengkungan di tiap sisi sisinya, serta mempunyai sedikit bayangan. Lalu di dalamnya akan ada text dengan warna putih dan ukuran font 20px.

Hasilnya akan ada di slide berikutnya

```
13   |   body: Container(
14   |   |   padding: const EdgeInsets.all(10.0),
15   |   |   child: Column(
16   |   |   |   children: [
17   |   |   |   |   ElevatedButton(
18   |   |   |   |   |   onPressed: () {},
19   |   |   |   |   |   style: ElevatedButton.styleFrom(
20   |   |   |   |   |   |   backgroundColor: Colors.blue,
21   |   |   |   |   |   |   shape: RoundedRectangleBorder(
22   |   |   |   |   |   |   |   borderRadius: BorderRadius.circular(20),
23   |   |   |   |   |   |   ), // RoundedRectangleBorder
24   |   |   |   |   |   |   shadowColor:
25   |   |   |   |   |   |   |   Colors.grey[20], | You, 1 second ago *
26   |   |   |   |   |   |   |   elevation: 5.0,
27   |   |   |   |   |   |   ),
28   |   |   |   |   |   child: Text(
29   |   |   |   |   |   |   "Add To Cart".toUpperCase(),
30   |   |   |   |   |   |   style: const TextStyle(
31   |   |   |   |   |   |   |   fontSize: 20,
32   |   |   |   |   |   |   |   fontWeight: FontWeight.w500,
33   |   |   |   |   |   |   |   color: Colors.white,
34   |   |   |   |   |   |   |   ), // TextStyle
35   |   |   |   |   |   |   |   ), // Text
36   |   |   |   |   |   |   ), // ElevatedButton
37   |   |   |   |   ],
38   |   |   |   ),
39   |   |   ), // Column
39   |   ), // Container
```

Gambar sebelah kiri adalah hasil codenya ketika di running.

Ini nanti akan bermanfaat untuk membuat tombol seperti tombol tambah ke keranjang atau tombol hubungi penjual.



Icon

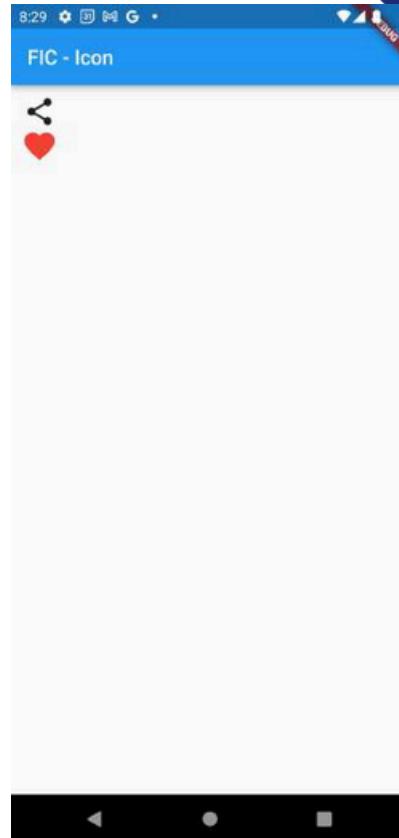
Icon sudah tersedia di flutter, jadi kita tidak perlu mengunduhnya.

Icon ini adalah gambar vektor sehingga ukurannya walau berubah-rubah ukuran, tidak akan kehilangan kualitas gambarnya.

Disamping contoh penggunaan widget icon, dimana ceritanya saya ingin membuat icon share dengan ukuran 32, dan juga icon favorite dengan ukuran 36 dan bisa langsung dikasih warna merah. Hasilnya ada di slide berikutnya

```
12 |   body: Container(
13 |     padding: const EdgeInsets.all(10.0),
14 |     child: Column(
15 |       children: const [
16 |         Icon(
17 |           Icons.share,
18 |           size: 32.0,
19 |         ), // Icon
20 |         Icon(
21 |           Icons.favorite,
22 |           size: 36.0,
23 |           color: Colors.red,
24 |         ), // Icon
25 |       ],
26 |     ), // Column
27 |   ), // Container
```

Hasilnya ada di gambar sebelah kiri dan penggunaan icon ini akan banyak sekali dipakai dalam aplikasi apapun yang akan kita buat. Untuk di sample gold material, icon ini untuk menandai suatu product telah di masukan ke product favorite user.



Image

Beberapa konstruktor disediakan untuk berbagai cara agar gambar dapat ditentukan:

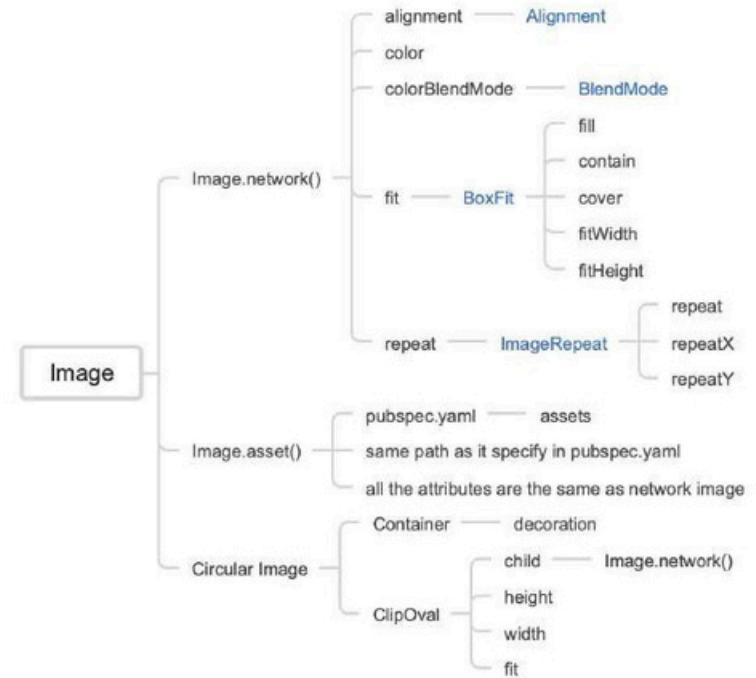
`Image.new`, untuk mendapatkan gambar dari `ImageProvider`.

`Image.asset`, untuk mendapatkan gambar dari `AssetBundle` menggunakan kunci.

`Image.network`, untuk mendapatkan gambar dari URL.

`Image.file`, untuk mendapatkan gambar dari File.

`Image.memory`, untuk mendapatkan gambar dari `Uint8List`.



Image

Code disamping, saya ingin mempraktekan image yang diambil dari asset. Pertama set dulu config di pubspec.yaml

assets:

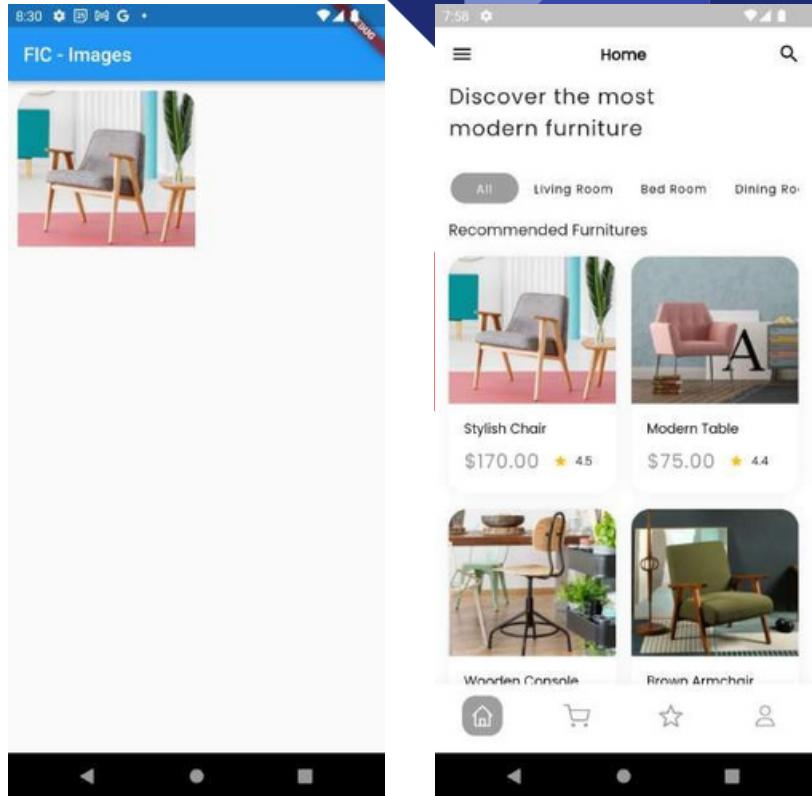
- assets/icons/
- assets/images/
- assets/images/furniture/

```
13   body: Container(
14     padding: const EdgeInsets.all(10.0),
15     child: Column(
16       children: [
17         ClipRRect(
18           borderRadius: const BorderRadius.only(
19             topLeft: Radius.circular(20),
20             topRight: Radius.circular(20),
21           ), // BorderRadius.only
22           child: Image.asset('assets/images/furniture/img_product_2.png'),
23         ), // ClipRRect
24       ],
25     ), // Column
26   ), // Container
```

Jangan lupa masukan gambar nya ke folder yang sudah didaftarkan ke config assets.

Image

Bila dirunning kode sebelumnya, maka tampilannya akan seperti gambar sebelah kiri. Dan nantinya dapat kita implementasi untuk membuat tampilan seperti di gambar sebelah kanan.

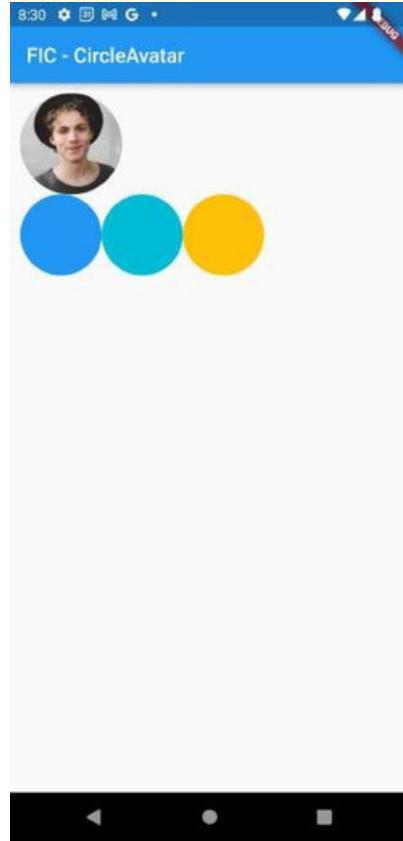


CircleAvatar

Ini adalah widget yang selalu berbentuk lingkaran yang mempunyai background warna atau gambar. Biasa digunakan untuk gambar profil user. Walau tidak menutup kemungkinan untuk urusan lain yang penting urusan bentuk lingkaran dengan latar belakang gambar dan warna. Seperti contoh disamping. Dibari 24 ini kita membuat profile picture menggunakan circleavatar dengan radius 50. Lalu diline 30 kita membuat row yang isinya widget lingkarang dengan latar belakang warna dengan radius 40. Hasilnya dapat kita lihat dislide berikutnya

```
19 |   body: Container(
20 |     padding: const EdgeInsets.all(10.0),
21 |   child: Column(
22 |     crossAxisAlignment: CrossAxisAlignment.start,
23 |     children: [
24 |       const CircleAvatar(
25 |         radius: 50,
26 |         backgroundImage: NetworkImage(
27 |           "https://i.ibb.co/PGv8ZzG/me.jpg",
28 |         ), // NetworkImage
29 |       ), // CircleAvatar
30 |       Row(
31 |         children: [
32 |           ...List.generate(
33 |             colors.length,
34 |             (index) => CircleAvatar(
35 |               radius: 40,
36 |               backgroundColor: colors[index],
37 |             ), // CircleAvatar
38 |           ), // List.generate
39 |         ],
40 |       ), // Row
41 |       ],
42 |     ), // Column
43 |   ), // Container
```

Gambar sebelah kiri adalah hasil dari kode kita sebelumnya, dan widget ini nanti nya dapat kita implementasikan ke dalam project kita bisa menjadi profil photo atau pilihan warna dengan bentuk lingkaran.





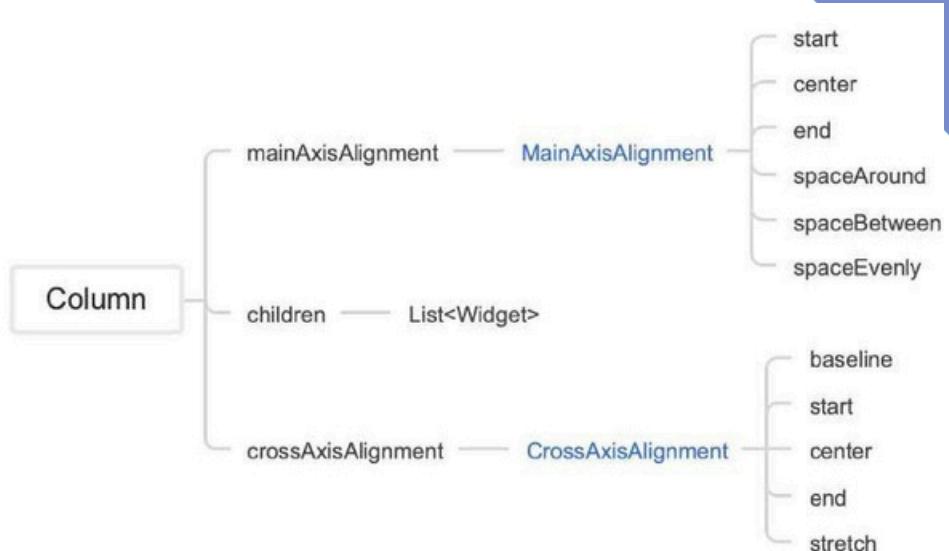
**Dalam basic Layout kali ini, kita akan
mengenal:**

- Column
- Row
- ListView
- GridView
- Padding
- AspectRatio
- Center
- Expanded
- SizedBox
- Wrap
- Stack

Column

Widget kolom ini menempatkan widget-widget anaknya pada sumbu vertikal dengan batasan ruang yang sudah ditentukan.

Disamping ini mindmap dari widget column.
mainAxisAlignment adalah alignment yang mengikuti sumbu axis nya column artinya sumbu vertikal. Sedang crossAxisAlignment adalah sebaliknya yaitu alignment yang mengikuti sumbu horizontal.

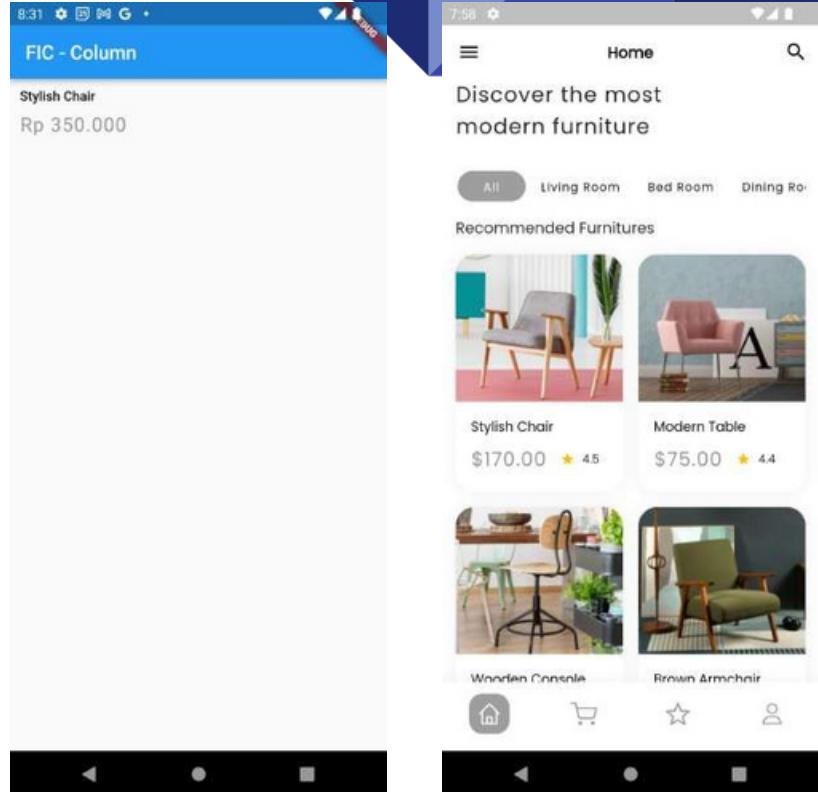


Dalam contoh code disamping ceritanya saya ingin membuat tampilan susun dari atas ke bawah. Baris 14 kita gunakan widget column lalu cross alignmentnya saya set start supaya rata kiri. Dan setelah itu childrennya pertama saya isi text, lalu jarak 10px, lalu bawahnya text lagi berupa harga.

Hasilnya seperti slide selanjutnya

```
12 |   body: Container(
13 |     padding: const EdgeInsets.all(10.0),
14 |     child: Column(
15 |       crossAxisAlignment: CrossAxisAlignment.start,
16 |       children: const [
17 |         Text(
18 |           'Stylish Chair',
19 |           style: TextStyle(
20 |             color: Colors.black,
21 |             fontSize: 14,
22 |             fontWeight: FontWeight.w500,
23 |           ), // TextStyle
24 |         ), // Text
25 |         SizedBox(
26 |           height: 10,
27 |         ), // SizedBox
28 |         Text(
29 |           'Rp 350.000',
30 |           style: TextStyle(
31 |             fontSize: 20,
32 |             color: Color(0xFF9A9390),
33 |             fontWeight: FontWeight.w400,
34 |             letterSpacing: 1,
35 |           ), // TextStyle
36 |         ), // Text
37 |       ],
38 |     ), // Column
39 |   ), // Container
```

Gambar sebelah kiri adalah hasilnya, widget ini akan banyak kita gunakan dalam pengembangan aplikasi karena sifatnya yang dapat membuat layout berbaris dari atas ke bawah. Nantinya dapat kita implementasi seperti contoh di sebelah kanan dalam menata tampilan title product dan harga serta rating.

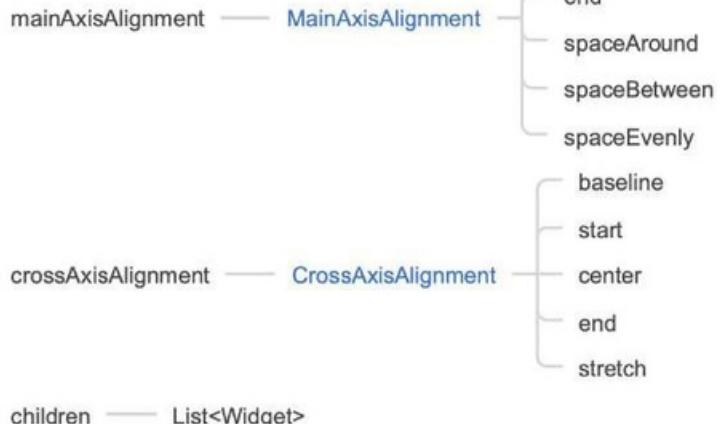


Row

Widget row ini menempatkan widget-widget anaknya pada sumbu horizontal dengan batasan ruang yang sudah ditentukan, kebalikan dari column.

Disamping ini mindmap dari widget row.

mainAxisAlignment adalah alignment yang mengikuti sumbu axis nya row artinya sumbu horizontal. Sedang crosAxisAlignment adalah sebaliknya yaitu aligment yang mengikuti sumbu vertikal..

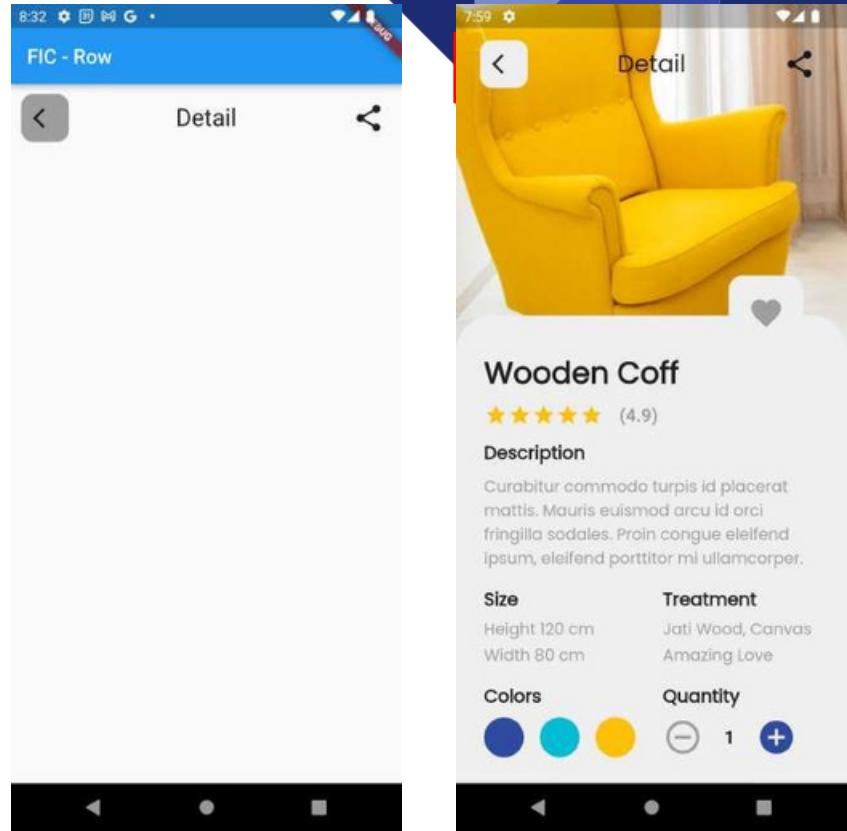


Contoh kode disamping cerita kita akan membuat tampilan header diaaman disebelah kiri ada icon back, ditengah ada title dan disebelah kanan ada icon share.

Hasilnya dapat kita lihat di slide berikutnya.

```
16 |   Row(
17 |     mainAxisAlignment: MainAxisAlignment.spaceBetween,
18 |     children: [
19 |       Container(
20 |         decoration: BoxDecoration(
21 |           color: Colors.grey,
22 |           borderRadius: BorderRadius.circular(12.0),
23 |         ), // BoxDecoration
24 |         child: IconButton(
25 |           onPressed: () {},
26 |           icon: const Icon(Icons.arrow_back_ios),
27 |         ), // IconButton
28 |       ), // Container
29 |       const Text(
30 |         "Detail",
31 |         style: TextStyle(
32 |           fontSize: 24.0,
33 |           fontWeight: FontWeight.normal,
34 |         ), // TextStyle
35 |       ), // Text
36 |       IconButton(
37 |         onPressed: () {},
38 |         icon: const Icon(
39 |           Icons.share,
40 |           size: 32.0,
41 |         ), // Icon
42 |       ), // IconButton
43 |     ],
44 |   ), // Row
```

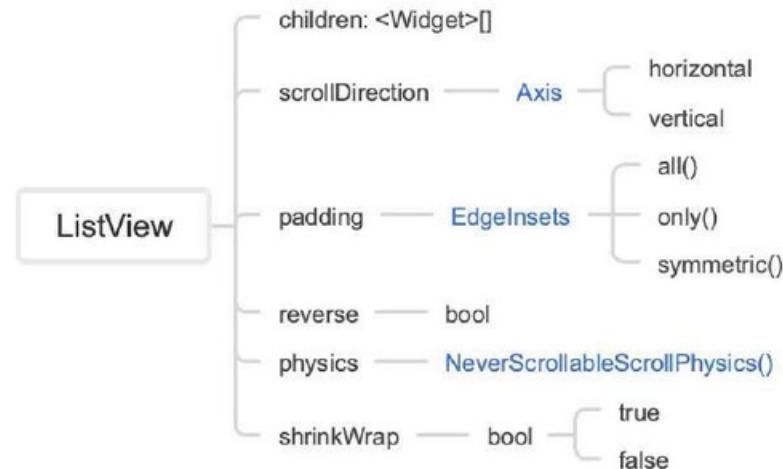
Gambar sebelah kiri adalah hasil kode yang kita buat sebelumnya. Dan nantinya dapat kita implementasi untuk membuat sample latihan slicings di detail product untuk headernya.



ListView

ListView pada dasarnya adalah sebuah Kolom dengan perilaku bergulir atau scroll karena menempatkan satu atau lebih banyak anak dalam sumbu vertikal, secara berurutan. Widget ini sangat banyak digunakan karena memberikan keunggulan untuk menggulirkan konten ketika jumlah content lebih besar dari ukuran layar.

Untuk membuat listview scroll ke samping, teman-teman bisa menggunakan properties scrollDirection Axis.horizontal seperti yang terlihat di mindmap.



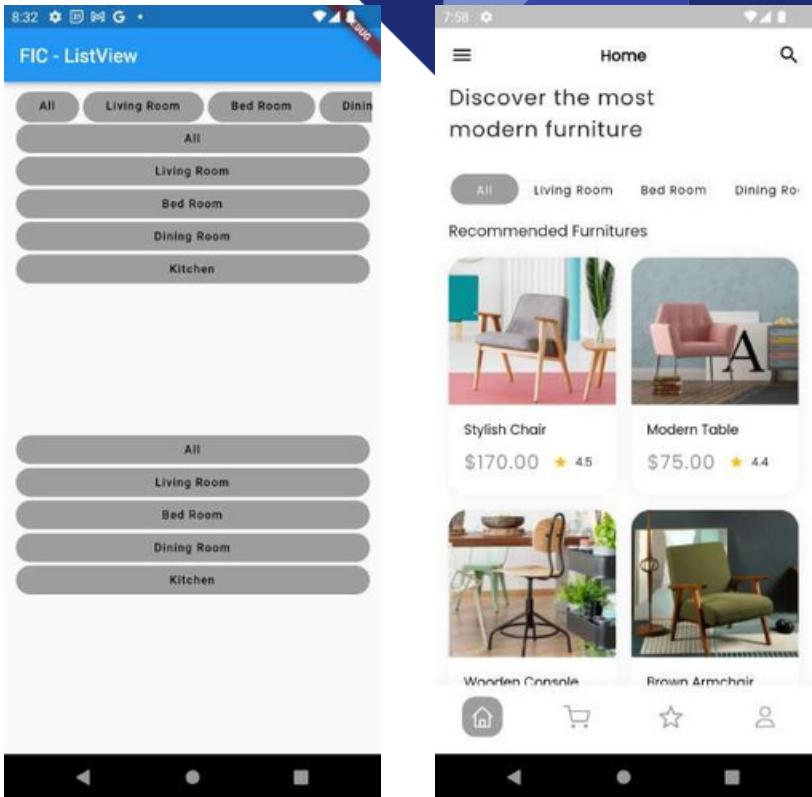
Dalam contoh code disamping ini, saya ingin memperlihatkan bagaimana kita dapat membuat tampilan list ke samping dan jumlah kontennya melebihi lebar layar dan itu tidak masalah, karena dia dapat di scroll ke samping. Kode lengkapnya dapat temen-temen akses di [github](#).

Pada baris 26 kita membuat widget listview dimana childnya adalah list dari categories.

Hasilnya dapat kita lihat di slide selanjutnya

```
26   |         child: ListView(
27   |           scrollDirection: Axis.horizontal,
28   |           children: List.generate(
29   |             categories.length,
30   |             (index) {
31   |               return GestureDetector(
32   |                 onTap: () {},
33   |                 child: AnimatedContainer(
34   |                   duration: const Duration(milliseconds: 150),
35   |                   decoration: BoxDecoration(
36   |                     borderRadius: BorderRadius.circular(20),
37   |                     color: Colors.grey,
38   |                   ), // BoxDecoration
39   |                   padding: const EdgeInsets.symmetric(
40   |                     vertical: 8,
41   |                     horizontal: 24,
42   |                   ), // EdgeInsets.symmetric
43   |                   margin: const EdgeInsets.symmetric(horizontal: 2),
44   |                   child: Text(
45   |                     categories[index],
46   |                     textAlign: TextAlign.center,
47   |                     style: const TextStyle(
48   |                       fontSize: 12,
49   |                       color: Colors.black,
50   |                       fontWeight: FontWeight.w500,
51   |                       letterSpacing: 1,
52   |                     ), // TextStyle
53   |                   ), // Text
54   |                   ), // AnimatedContainer
55   |                 ); // GestureDetector
56   |               ),
57   |             ), // List.generate
58   |           ), // ListView
```

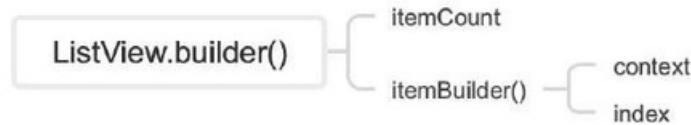
Gambar sebelah kiri dan bagian atas itu adalah hasil dari kode sebelumnya. Hasilnya berupa jejeran button dari kanan ke kiri. Case ini nanti dapat kita praktikan dalam pembuatan furniture apps untuk memilih kategori furnitures.



ListView.builder

`ListView.builder(...)` sangat berguna ketika list harus dibuat berdasarkan data yang lebih banyak.

Dokumentasi resmi Flutter menyarankan untuk menggunakan konstruktor bernama `builder(...)` ketika sumber data adalah data yang panjang karena secara efisien widget ini akan mengelola anak-anaknya ketika dipanggil dan hanya akan di build ketika terlihat dilayar.. Jadi, daripada secara manual mengisi `ListView` yang panjang dengan loop `for`, gunakan `builder()` yang lebih efisien.



Disamping adalah contoh tampilan yang menggunakan listview builder. Untuk data yang panjang seperti data dari database sangatlah cocok menggunakan widget ini karena data akan diproses build ketika data di tampilkan di layar akibat dari scroll kebawah atau keatas.

```
96    |         child: ListView.builder(
97    |         itemCount: categories.length,
98    |         itemBuilder: (context, index) {
99    |             return GestureDetector(
100                |             onTap: () {},
101                |             child: AnimatedContainer(
102                    |                     duration: const Duration(milliseconds: 150),
103                    |                     decoration: BoxDecoration(
104                        |                         borderRadius: BorderRadius.circular(20),
105                        |                         color: □Colors.grey,
106                    ), // BoxDecoration
107                    |                     padding: const EdgeInsets.symmetric(
108                        |                         vertical: 8,
109                        |                         horizontal: 24,
110                    ), // EdgeInsets.symmetric
111                    |                     margin: const EdgeInsets.symmetric(
112                        |                         horizontal: 2, vertical: 2), // EdgeInsets.symmetric
113                    |                     child: Text(
114                        |                         categories[index],
115                        |                         textAlign: TextAlign.center,
116                        |                         style: const TextStyle(
117                            |                             fontSize: 12,
118                            |                             color: □Colors.black,
119                            |                             fontWeight: FontWeight.w500,
120                            |                             letterSpacing: 1,
121                        ), // TextStyle
122                        |                     ), // Text
123                    ), // AnimatedContainer
124                    |                 ), // GestureDetector
125                ), // ListView.builder
```



Berikut hasil tampilan listview builder scrolldirection nya vertikal atau dari atas ke bawah. Untuk data sedikit mungkin tidak terlalu terlihat perbedaannya, namun untuk data yang banyak akan terlihat lebih efisien menggunakan listview builder.

Gridview.builder

Widget GridView secara otomatis menempatkan elemen dalam grid dan jumlah kolom ditentukan oleh nilai yang diteruskan ke crossAxisCount. Kode ini dikatakan responsif karena ketika lebar layar berubah, maka UI akan diatur ulang.

GridView.builder()

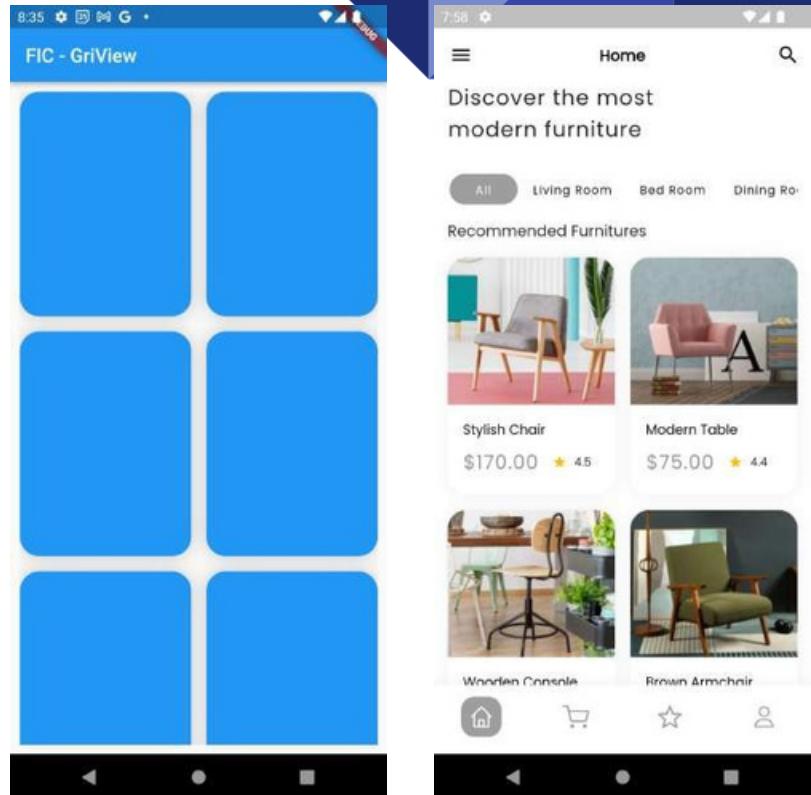


Kode disamping kita menggunakan gridView untuk membuat tampilan 2 column yang akan berisi container dengan bentuk yang siap menampung tampilan product.

Hasilnya seperti gambar di slide selanjutnya.

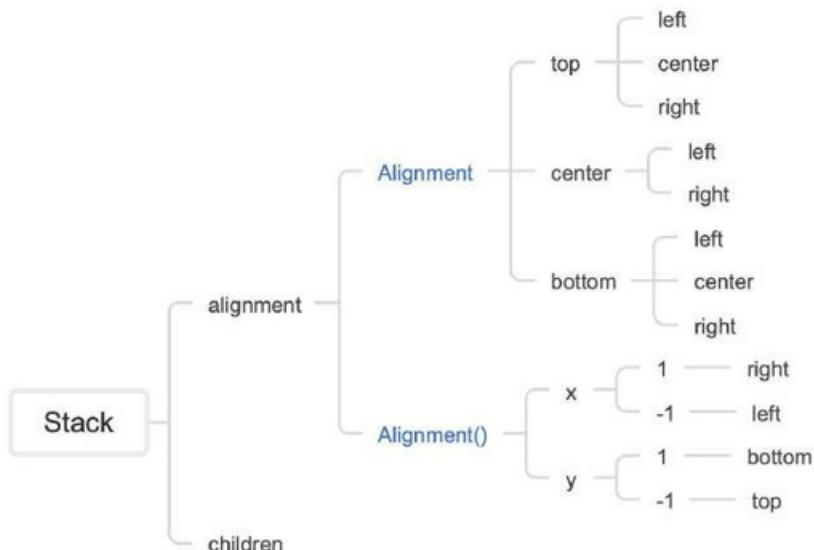
```
17 |           └── Expanded(
18 |             └─child: GridView.count(
19 |               crossAxisCount: 2,
20 |               childAspectRatio: 185 / 243,
21 |               mainAxisSpacing: 16,
22 |               crossAxisSpacing: 16,
23 |               children: [
24 |                 ...List.generate(
25 |                   6,
26 |                   (index) => Container(
27 |                     height: 300,
28 |                     width: double.infinity,
29 |                     decoration: BoxDecoration(
30 |                       color: Colors.blue,
31 |                       borderRadius: BorderRadius.circular(20),
32 |                       boxShadow: [
33 |                         BoxShadow(
34 |                           color: Colors.black.withOpacity(0.2),
35 |                           offset: Offset.zero,
36 |                           blurRadius: 15.0,
37 |                         ) // BoxShadow
38 |                       ],
39 |                     ), // BoxDecoration
40 |                   ), // Container
41 |                 ), // List.generate
42 |               ],
43 |             ), // GridView.count
44 |           ), // Expanded
```

Gambar sebelah kiri adalah hasil dari kode kita sebelumnya, dan kode ini nantinya dapat kita implementasi di kasus-kasus aplikasi yang membutuhkan tampilan grid view column lebih dari satu. Seperti gambar sebelah kanan. Itu tampilan menggunakan gridView dengan crossAxisCount nya berjumlah 2 .



Stack

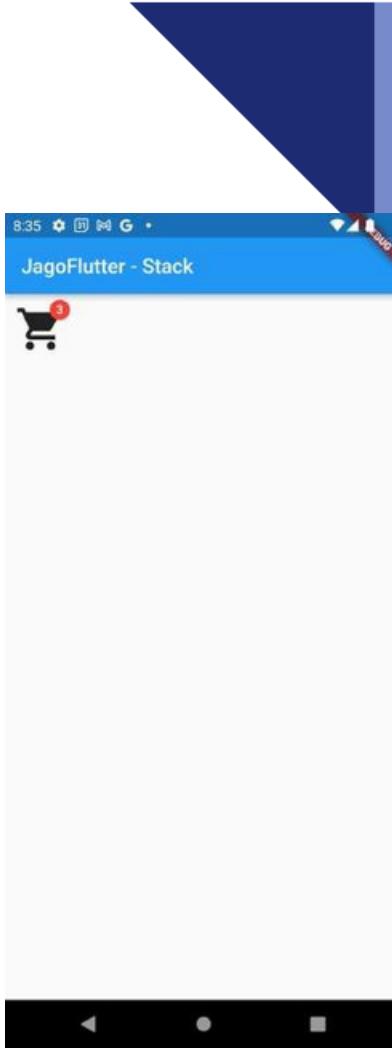
Dengan Widget Tumpukan, Anda dapat menumpuk widget dan memposisikannya dengan bebas di layar menggunakan widget Positioned. Bahkan jika childrennya ditempatkan di luar batas UI, tidak akan ada kesalahan sistem yang muncul, karena Stack tidak membatasi batas lebar dan tinggi.



Seperti contoh code disamping. Saya membuat tampilan menggunakan stack dimana dibagian bawah saya mempunyai icon shopping cart dengan size 50, lalu diatasnya saya tumpuk dengan widget lingkaran dengan radius 10 dan background warna merah lalu didalamnya terdapat text angka 3.

Hasilnya seperti gambar di slide selanjutnya

```
16 |   └── Stack(
17 |     clipBehavior: Clip.none,
18 |     children: const [
19 |       Icon(
20 |         Icons.shopping_cart,
21 |         size: 50,
22 |       ), // Icon
23 |       Positioned(
24 |         top: -4.0,
25 |         right: -4.0,
26 |         child: CircleAvatar(
27 |           radius: 10,
28 |           backgroundColor: Colors.red,
29 |           child: Text(
30 |             '3',
31 |             style: TextStyle(
32 |               fontSize: 12,
33 |               fontWeight: FontWeight.bold,
34 |               color: Colors.white,
35 |             ), // TextStyle
36 |             ), // Text
37 |           ), // CircleAvatar
38 |         ), // Positioned
39 |       ],
40 |     ) // Stack
```



Terlihat lingkaran merah berada diatas shopping cart dan menumpuk di bagian atas sebelah kanan.



Padding

Padding widget melakukan persis seperti namanya, menambahkan padding atau ruang kosong di sekitar widget atau sekelompok widget. Kita dapat menerapkan padding di sekitar widget apapun dengan menempatkannya sebagai child dari widget Padding.

Padding

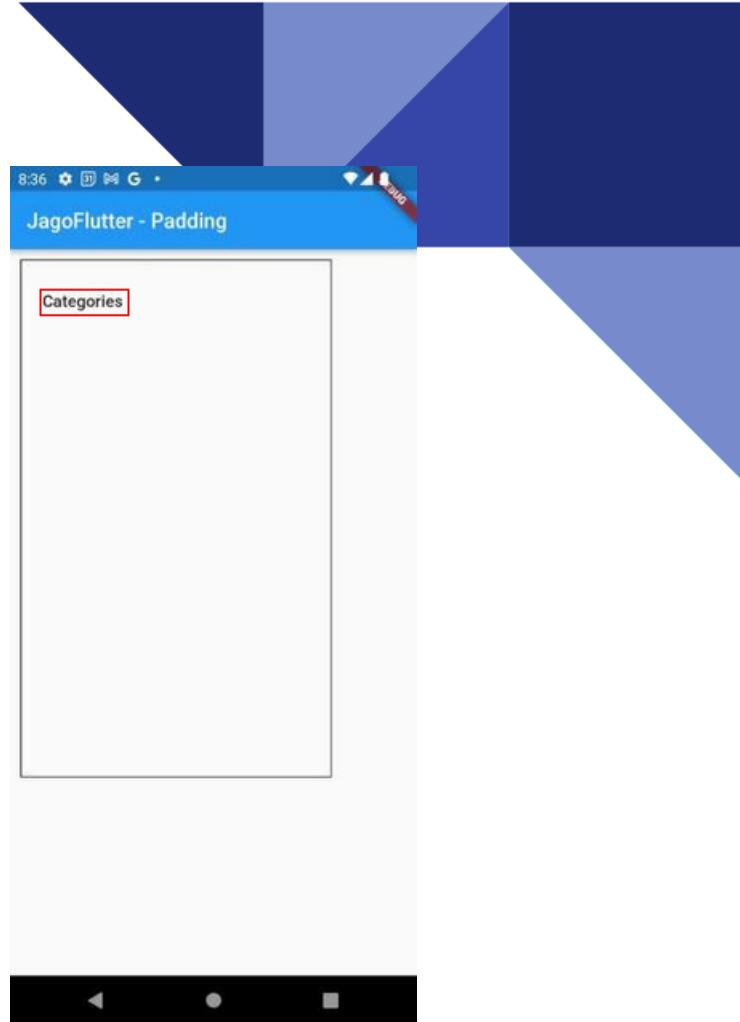
- `EdgeInsets.only()`
- `EdgeInsets.symmetric()`
- `EdgeInsets.all()`
- `EdgeInsets.fromLTRB()`

Seperti contoh disamping. Text categories saya bungkus dengan widget padding supaya mempunyai jarak dengan area luarnya. Disini saya set jarak dari text ke arah kiri sejauh 20px, keatas 30px dan ke bawah 40px.

Hasilnya bisa kita lihat di slide selanjutnya

```
13   |   body: Container(
14   |   |   height: 500,
15   |   |   width: 300,
16   |   |   margin: const EdgeInsets.all(10.0),
17   |   |   decoration: BoxDecoration(border: Border.all()),
18   |   |   child: Column(
19   |   |   |   crossAxisAlignment: CrossAxisAlignment.start,
20   |   |   |   children: const [
21   |   |   |   Padding(
22   |   |   |   |   padding: EdgeInsets.only(
23   |   |   |   |   left: 20,
24   |   |   |   |   top: 30,
25   |   |   |   |   bottom: 40,
26   |   |   |   |   ), // EdgeInsets.only
27   |   |   |   |   child: Text(
28   |   |   |   |   |   'Categories',
29   |   |   |   |   |   style: TextStyle(
30   |   |   |   |   |   |   fontSize: 16.0,
31   |   |   |   |   |   |   fontWeight: FontWeight.w500,
32   |   |   |   |   |   |   ), // TextStyle
33   |   |   |   |   |   ), // Text
34   |   |   |   |   ), // Padding
35   |   |   |   ],
36   |   |   |   ), // Column
37   |   |   ), // Container
```

Disini terlihat widget text categories yang saya kotakin itu mempunyai jarak dengan widget luarnya yang saya sudah kasih garis border hitam. Jarak ke samping kiri sejauh 20px, ke atas 30 px dan ke bawah 40 px, karena widget luarnya itu lebih luas, maka jarak ke bawahnya tidak begitu terlihat.



AspectRatio

Widget yang mencoba mengubah ukuran child ke rasio tinggi dan lebar tertentu.

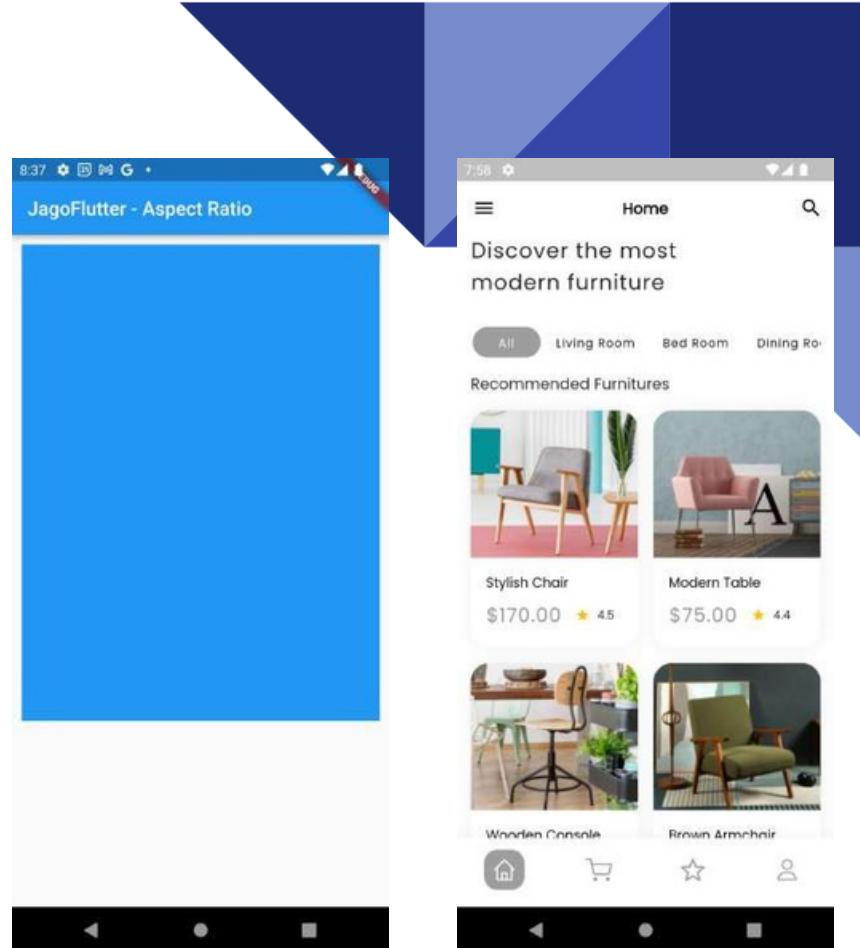
Widget terlebih dahulu mencoba rasio lebarnya terlebih dahulu sesuai yang sudah diatur. Ketinggian widget ditentukan dengan menerapkan rasio aspek yang diberikan ke lebar, dinyatakan sebagai rasio lebar terhadap tinggi.



Seperti contoh code disamping. Saya memasukan container dengan color blue kedalam widget aspectratio dengan rasio lebar 180 dan tingginya 240. Dan widget ini akan otomatis mengikuti lebar dan tinggi layar dengan rasio yang sama.

```
16   |           └─AspectRatio(  
17   |               aspectRatio: 180 / 240,  
18   |               child: Container(  
19   |                   color: Colors.blue,  
20   |               ), // Container  
21   |           ) // AspectRatio
```

Hasilnya seperti gambar disamping kiri. Mungkin tidak begitu terlihat hasil rasionalnya, namun ketika widget ini dimasukan ke dalam gridview, maka akan cocok sekali karena konten yg ada di dalam gridview nanti nya akan memiliki rasio yg sama dan fleksible mengikuti lebar dan tinggi layar device.



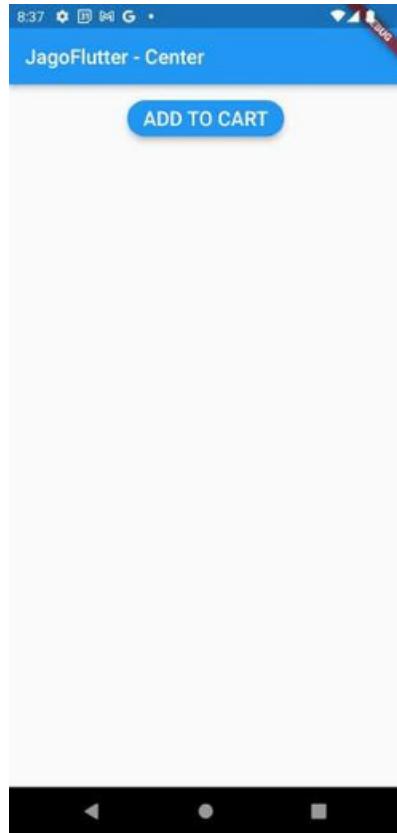
Center

Widget ini akan memusatkan anak nya kediri dia, yang itu artinya widget anaknya akan ada ditengah-tengah widget disekitarnya karena sifatnya yang dapat memenuhi ruang di sekitarnya.

Pada contoh di samping saya bungkus button add to cart dengan center sehingga tampilannya bisa berada ditengah layar.

```
16  ┌── Center(  
17  │   ┌── child: ElevatedButton(  
18  │   │   onPressed: () {},  
19  │   │   style: ElevatedButton.styleFrom(  
20  │   │       backgroundColor: Colors.blue,  
21  │   │       shape: RoundedRectangleBorder(  
22  │   │           borderRadius: BorderRadius.circular(20),  
23  │   │       ), // RoundedRectangleBorder  
24  │   │       shadowColor:  
25  │   │           Colors.grey[20], //specify the button's  
26  │   │       elevation: 5.0,  
27  │   │   ),  
28  │   ┌── child: Text(  
29  │   │   "Add To Cart".toUpperCase(),  
30  │   │   style: const TextStyle(  
31  │   │       fontSize: 20,  
32  │   │       fontWeight: FontWeight.w500,  
33  │   │       color: Colors.white,  
34  │   │   ), // TextStyle  
35  │   │   ), // Text  
36  │   ┌── ), // ElevatedButton  
37  ┌── ), // Center
```

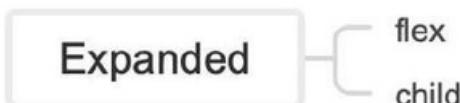
Hasil dari code sebelumnya adalah gambar sebelah kiri, dimana button nya posisi sekarang berada di tengah bagian atas. Kenapa bagian atas karena center ini mengikuti sifat widget sekitarnya yaitu column dengan mainAxisAlignment nya default start alias ada dibagian paling atas.



Expanded

Widget yang memperluas childnya dari Row, Column, atau Flex sehingga child tersebut mengisi ruang yang tersedia.

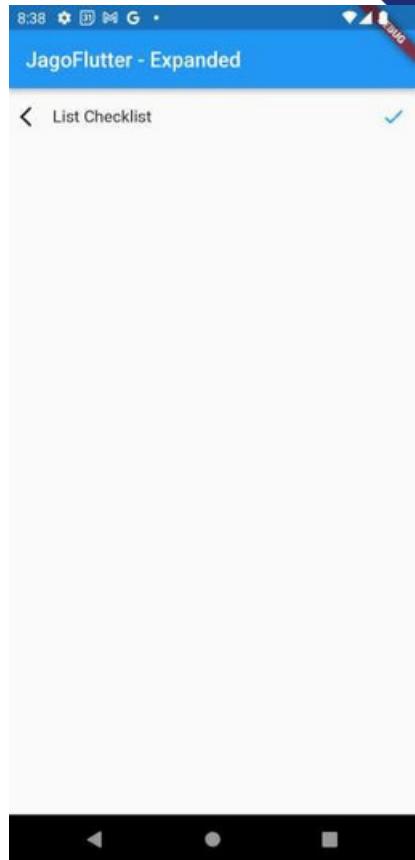
Dengan expanded widget child dari Row, Column, atau Flex dapat memperluas diri untuk mengisi ruang yang tersedia di sepanjang sumbu utamanya (mis., secara horizontal untuk Baris/row atau vertikal untuk Kolom). Jika beberapa child diperluas, ruang yang tersedia dibagi di antara mereka menurut faktor flex.



Expanded

Expanded harus berada di bawahnya row atau column, dalam contoh kode disamping, expanded ada di dalam row dan membungkus text list checklist, yg artinya text ini akan mengisi sisa area yg ada. Yg akhirnya icon arrow back dan check akan ada di sebelah kiri dan kanan.

```
16 |   Row(           |
17 |     children: const [           |
18 |       Icon(Icons.arrow_back_ios),           |
19 |       Expanded(           |
20 |         child: Padding(           |
21 |           padding: EdgeInsets.all(8.0),           |
22 |           child: Text(           |
23 |             'List Checklist',           |
24 |             style: TextStyle(fontSize: 16),           |
25 |           ), // Text           |
26 |           ), // Padding           |
27 |           ), // Expanded           |
28 |           Icon(           |
29 |             Icons.check,           |
30 |             color: Colors.blue,           |
31 |           ) // Icon           |
32 |           ],           |
33 |         ), // Row           |
34 |       ],           |
35 |     ), // Column
```

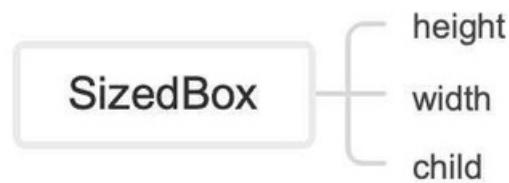


Hasil dari kode sebelumnya adalah seperti ini. Dimana list checklist memenuhi area yang ada sehingga icon check berada di paling ujung kanan.

SizedBox

Sebuah kotak dengan ukuran tertentu.

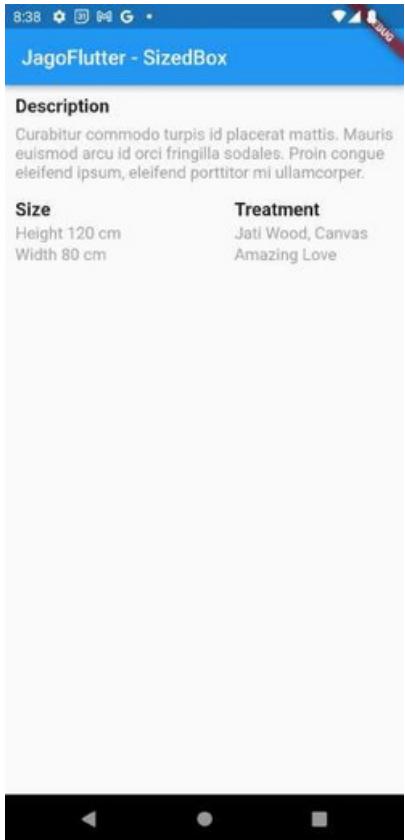
Jika diberi child, widget ini memaksanya child untuk memiliki lebar dan/atau tinggi tertentu. Nilai ini akan diabaikan jika induk widget ini mempunyai ukuran sendiri, maka sizedbox mengikuti ukuran parent nya. Hal ini dapat diatasi dengan membungkus child dari SizedBox ke dalam sebuah widget yang memungkinkan ukurannya sesuai dengan ukuran induknya, seperti Center atau Align.



Untuk contoh di samping sizedbox dipakai untuk membuat jarak antar text dengan membuat kotak kosong dengan tinggi 4 dan tinggi 2 .

```
37   |       Column(                                     , // Column
38   |         crossAxisAlignment: CrossAxisAlignment.start,
39   |         children: const [
40   |           Text(
41   |             "Size",
42   |             style: TextStyle(
43   |               fontSize: 18.0,
44   |               fontWeight: FontWeight.bold,
45   |             ), // TextStyle
46   |             ), // Text
47   |             SizedBox(height: 4.0),
48   |             Text(
49   |               "Height 120 cm",
50   |               style: TextStyle(
51   |                 fontSize: 16.0,
52   |                 color: Colors.grey,
53   |                 fontWeight: FontWeight.normal,
54   |               ), // TextStyle
55   |               ), // Text
56   |               SizedBox(height: 2.0),
57   |               Text(
58   |                 "Width 80 cm",
59   |                 style: TextStyle(
60   |                   fontSize: 16.0,
61   |                   color: Colors.grey,
62   |                   fontWeight: FontWeight.normal,
63   |                 ), // TextStyle
64   |                 ), // Text
65   |               ],
66   |             ), // Column
```

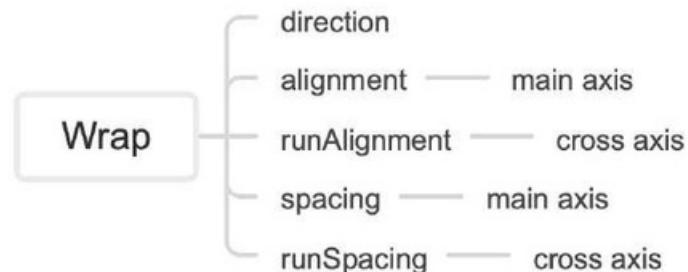
Hasilnya seperti gambar di samping kiri dimana antar widget atas bawah terdapat jarak yang dibuat oleh sizedbox.



Wrap

Widget yang menampilkan childnya dalam beberapa urutan horizontal atau vertikal.

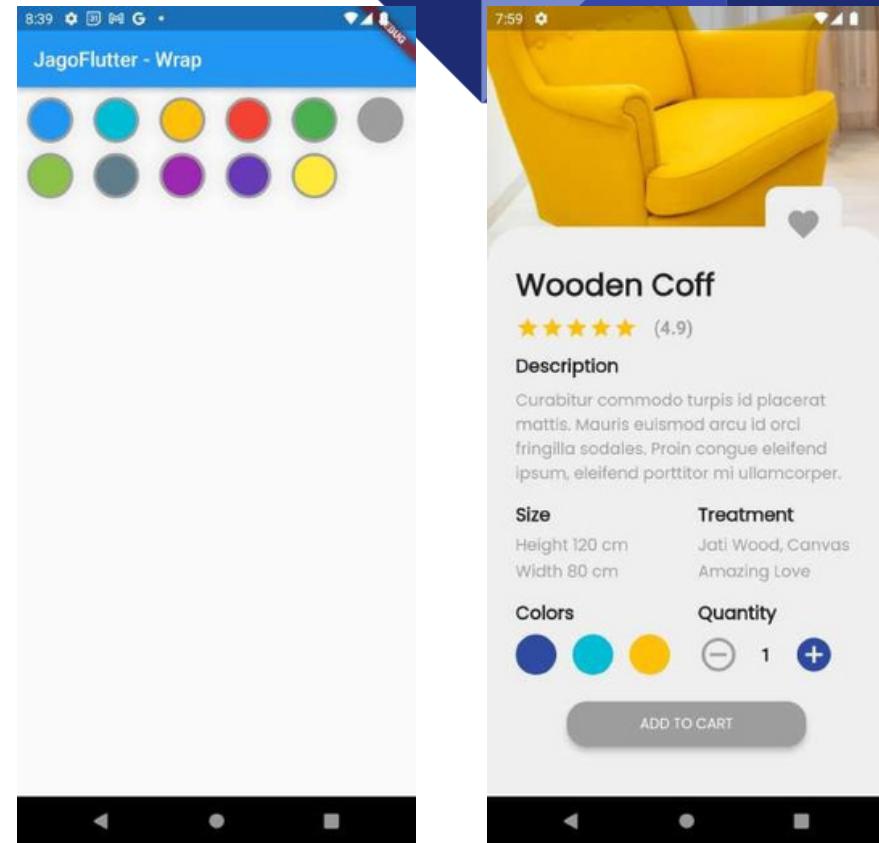
Sebuah wrap widget meletakkan setiap child dan mencoba untuk menempatkan childnya berdekatan dengan child sebelumnya di sumbu utama, diberikan arahnya, dan meninggalkan ruang jarak di antaranya. Jika tidak ada cukup ruang untuk memasukkan childnya, wrap membuat lintasan baru yang berdekatan dengan child yang sudah ada di sumbu silang.



Contoh kode disamping memperlihatkan list widget lingkaran dengan warna dibungkus dengan wrap dan hasilnya ketika kontennya tadi menabrak lebar layar, maka konten selanjutkan akan ditempat paling dekat dengan temannya namun disisi cross dari sumbu utama alias ketika ke samping maka next contentnya akan ada dibawahnya.

```
31     Wrap(
32       spacing: 20,
33       runSpacing: 10,
34       children: colors.map((color) {
35         return InkWell(
36           onTap: () {},
37           child: Container(
38             width: 45,
39             height: 45,
40             decoration: BoxDecoration(
41               border: Border.all(width: 3, color: Colors.grey),
42               color: color,
43               shape: BoxShape.circle,
44               boxShadow: [
45                 BoxShadow(
46                   color: Colors.black.withOpacity(0.1),
47                   offset: Offset.zero,
48                   blurRadius: 15,
49                 ) // BoxShadow
50               ], // BoxDecoration
51             ), // Container
52           ); // InkWell
53         }).toList(),
54       ), // Wrap
```

Kita bisa lihat hasilnya di gambar sebelah kiri dimana ketika widget circleavatar nya sudah penuh ke samping kanan, akan anaknya akan fleksible mencari ruang dibawahnya sehingga semua childrennya akan selalu terlihat.





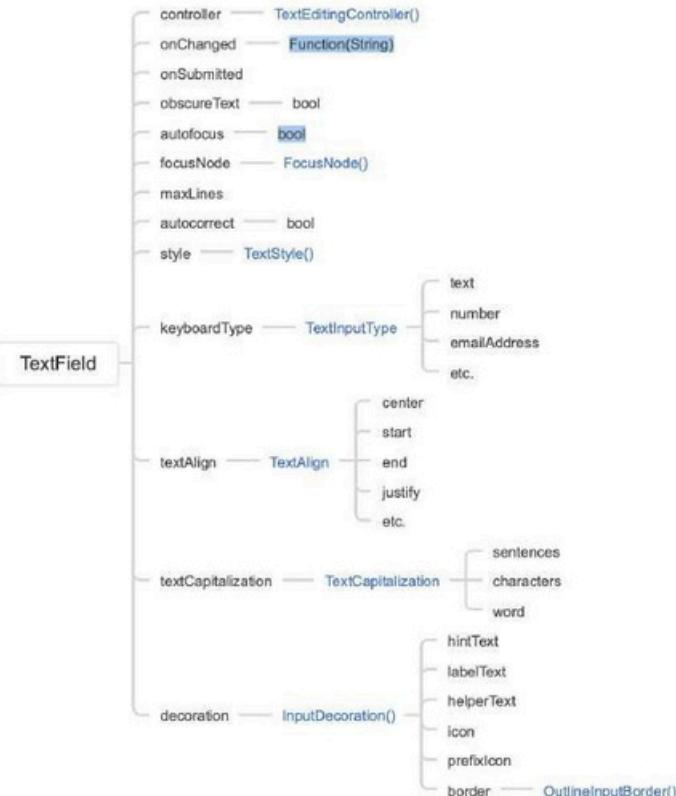
**Dalam basic Form kali ini, kita akan
mengenal:**

- TextField
- Dropdown
- Switch
- Radio
- Checkbox
- DatePicker
- Dialog
- Bottomsheet
- Snackbar

TextField

Bidang teks atau TextField memungkinkan user untuk memasukkan teks, baik dengan keyboard fisik maupun keyboard dilayar.

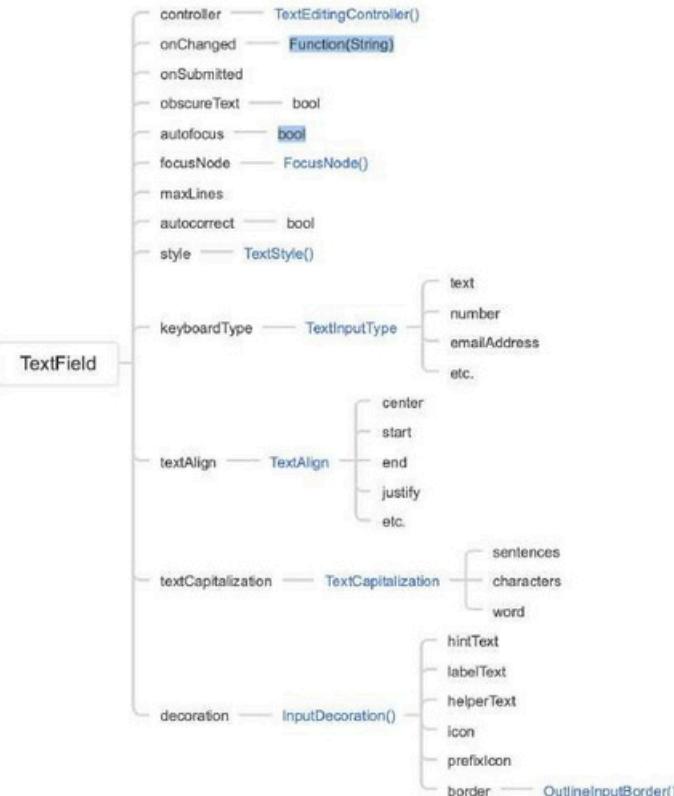
TextField memanggil callback onChanged setiap kali user mengubah teks di bidang. Jika pengguna menunjukkan bahwa mereka telah selesai mengetik di bidang (mis., dengan menekan tombol Enter di keyboard), TextField akan memanggil callback onSubmitted.



TextField

Untuk mengontrol teks yang ditampilkan di bidang teks, gunakan controller. Misalnya, untuk mengatur nilai awal bidang teks, gunakan controller yang sudah berisi beberapa teks. Controller juga dapat mengontrol wilayah pemilihan dan penulisan (dan untuk mengamati perubahan pada teks, pemilihan, dan wilayah penulisan).

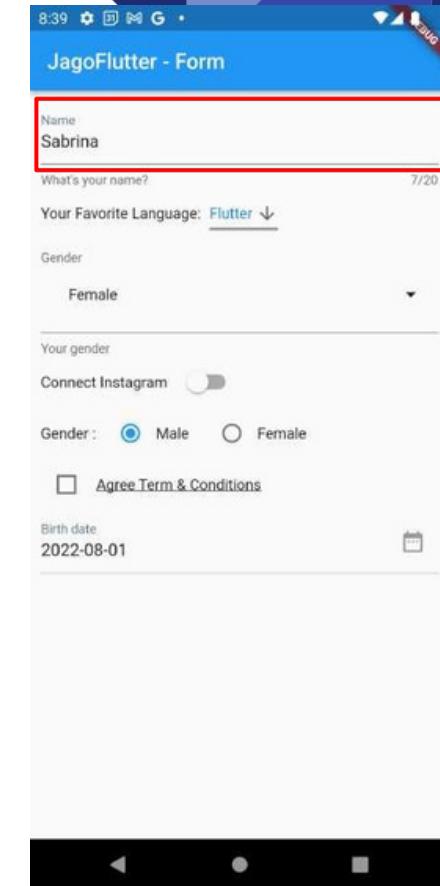
Secara default, bidang teks memiliki dekorasi yang menggambar pembatas di bawah bidang teks. Anda dapat menggunakan properti decoration untuk mengontrol dekorasi, misalnya dengan menambahkan label atau ikon. Jika Anda menyetel properti dekorasi ke null, dekorasi akan dihapus seluruhnya, termasuk padding tambahan yang dimasukkan oleh dekorasi untuk menghemat ruang bagi label.



Contoh code disamping ini ceritanya saya ingin membuat text field dengan panjang karakter input 20, terdapat controller untuk mengontrol textfield, ada decoration dengan label Name, lalu terdapat border underline berwarna bluegrey dan ada onchange juga yang bisa kita pakai untuk melihat perubahan tiap kali textfield di edit. Hasilnya seperti gambar di slide selanjutnya

```
32 |   TextField(          You, 16 hours ago • i
33 |     maxLength: 20,
34 |     controller: textController,
35 |     decoration: const InputDecoration(
36 |       labelText: 'Name',
37 |       labelStyle: TextStyle(
38 |         color: Colors.blueGrey,
39 |       ), // TextStyle
40 |       enabledBorder: UnderlineInputBorder(
41 |         borderSide: BorderSide(
42 |           color: Colors.blueGrey,
43 |         ), // BorderSide
44 |         ), // UnderlineInputBorder
45 |         helperText: "What's your name?",
46 |       ), // InputDecoration
47 |       onChanged: (value) {},
48 |     ), // TextField
```

Hasilnya seperti gambar disamping ini dimana terdapat textfield dengan label Name dan underscore blue gery serta tanda max karakter 20 dan helper noted what's your name?



Dropdown

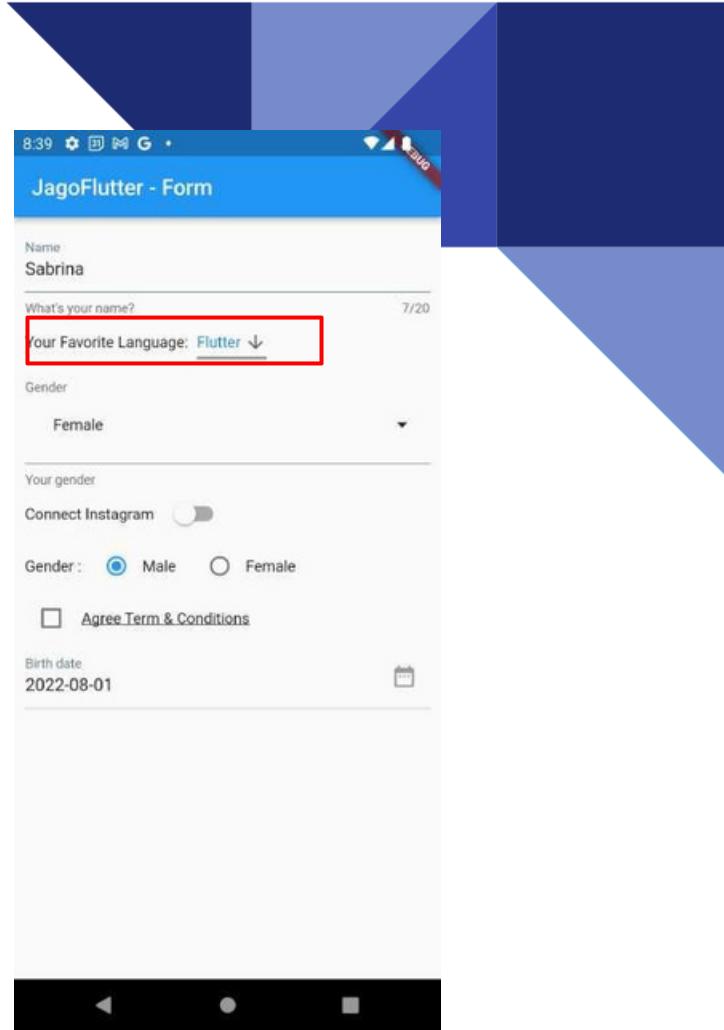
Tombol dropdown memungkinkan user memilih dari sejumlah item. Tombol menunjukkan item yang dipilih saat ini, serta panah yang membuka menu untuk memilih item lain.

Seperti kode disamping, saya membuat dropdownbutton disampingnya text your favorite language. Properties value wajib di isi dan wajib ada datanya di dalam item dengan value item wajib ada salah satunya itu yg terdapat di properties value.

```
45 |   Row(                                     You, 1 second ago • Uncommitted changes
46 |     mainAxisAlignment: MainAxisAlignment.start,
47 |     crossAxisAlignment: CrossAxisAlignment.center,
48 |     children: [
49 |       const Text("Your Favorite Language:"),
50 |       const SizedBox(
51 |         width: 8,
52 |       ), // SizedBox
53 |       DropdownButton(
54 |         value: selected,
55 |         icon: const Icon(Icons.arrow_downward),
56 |         iconSize: 20,
57 |         style: TextStyle(color: Colors.blue[600]),
58 |         underline: Container(
59 |           decoration: const BoxDecoration(
60 |             border: Border(
61 |               bottom: BorderSide(
62 |                 color: Colors.grey,
63 |                 width: 3,
64 |               ), // BorderSide
65 |             ), // Border
66 |           ), // BoxDecoration
67 |           ), // Container
68 |           items: dropdownList
69 |             .map((e) => DropdownMenuItem(value: e, child: Text(e)))
70 |             .toList(),
71 |             onChanged: (String? val) {
72 |               setState(() {
73 |                 if (val != null) selected = val;
74 |               });
75 |             },
76 |           ), // DropdownButton
77 |         ), // Row
```

Hasil kode nya ketika di running akan seperti ini.

Terdapat dropdown dan ada icon arrow downward dengan default value nya adalah Flutter.



Switch

Switch atau saklar adalah widget yang digunakan untuk memilih di antara dua opsi, AKTIF atau NONAKTIF. Dia tidak mempunya state sendiri. Untuk mempertahankan state nya, kita perlu memanggil properti onChanged. Jika nilai yang dikembalikan oleh properti ini adalah true, maka switch akan ON dan false saat OFF.

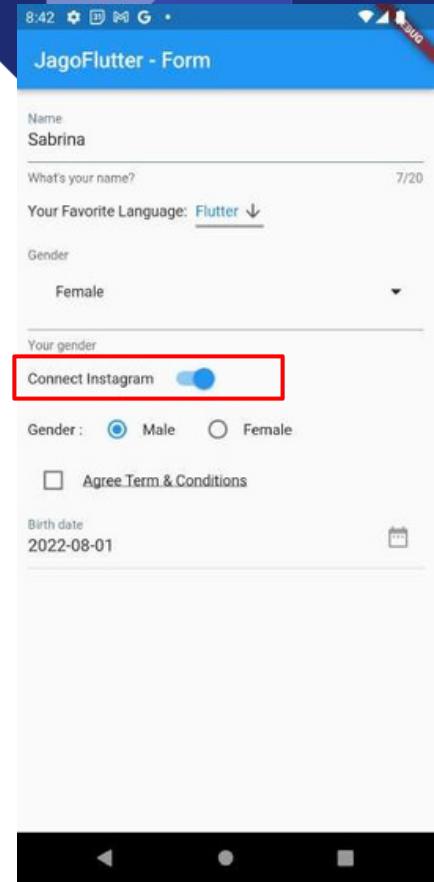
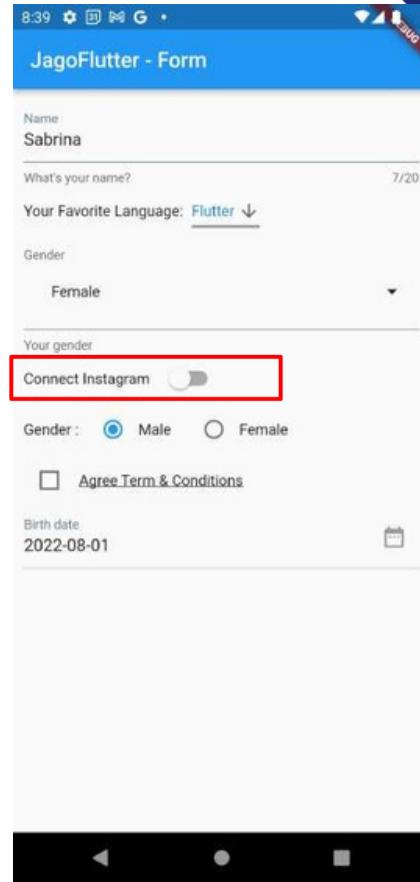


Seperti code disamping ini, state nya akan diubah dengan bantuan onchanged lalu dibantu dengan setState((){}) untuk mengupdate value switch dengan value yang baru untuk mempertahankan nilai nya.

```
148 Row(  
149   children: [  
150     const Text('Connect Instagram'),  
151     const SizedBox(  
152       width: 8,  
153     ), // SizedBox  
154     Switch(  
155       value: isOn,  
156       onChanged: (bool? val) {  
157         if (val != null) {  
158           setState(() {  
159             isOn = val;  
160           });  
161         }  
162       },  
163     ), // Switch  
164   ],  
165 ), // Row
```

Hasilnya nanti akan seperti gambar disamping ini,

Untuk gambar sebelah kiri Ini switch dalam kondisi off,
dan gambar sebelah kanan, switch dalam kondisi on.



Radio

Digunakan untuk memilih di antara sejumlah nilai yang sudah ditentukan. Ketika satu tombol radio dalam grup dipilih, tombol radio lain dalam grup tidak lagi dipilih. Nilainya bertipe tipe yang sama, parameter tipe dari kelas Radio. Enum biasanya digunakan untuk tujuan ini.

Tombol radio itu sendiri tidak mempertahankan state apa pun. Sebagai gantinya, radio akan memanggil callback onChanged, meneruskan nilai sebagai parameter. Jika groupValue dan nilai cocok, radio ini akan dipilih. Sebagian besar widget akan merespons onChanged dengan memanggil setState untuk memperbarui groupValue pada tombol radio.



Seperti contoh kode di samping, saya membuat groupValue dengan group param sex dan default valuenya male.

Lalu group valuenya akan diganti melalui properties onChanged nya ketika di klik akan mentriger properties ini dan mengupdate value nya.

```
168 |     children: [
169 |     const Text('Gender :'),
170 |     const SizedBox(),
171 |     width: 8,
172 |   ], // SizedBox
173 |   now + uncommitted changes
174 | ],
175 | ],
176 | ],
177 | ],
178 | ],
179 | ],
180 | ],
181 | ],
182 | ],
183 | ],
184 | ],
185 | ],
186 | ],
187 | ],
188 | ],
189 | ],
190 | ],
191 | ],
192 | ],
193 | ],
194 | ],
195 | ],
196 | ],
197 | ],
198 | ],
199 | ],
200 | ],
201 | ],
202 | ],
203 | ],
204 | ],
205 | ],
206 | ],
207 | ],
208 | ],
209 | ],
210 | ],
211 | ],
212 | ],
```

Hasil kodnya seperti disamping ini. Jika di klik male, maka female akan otomatis tidak terpilih dan sebaliknya.

The image displays two side-by-side screenshots of a mobile application interface titled "JagoFlutter - Form".

Screenshot 1 (Left): Shows a gender selection field where "Male" is selected (radio button is blue). A red box highlights this selection. Below the field is a "Connect Instagram" toggle switch, which is turned on (blue).

| | | |
|---------|---------------------------------------|------------------------------|
| Gender: | <input checked="" type="radio"/> Male | <input type="radio"/> Female |
|---------|---------------------------------------|------------------------------|

Screenshot 2 (Right): Shows the same gender selection field, but now "Female" is selected (radio button is blue). A red box highlights this selection. The "Connect Instagram" toggle switch is also turned on (blue).

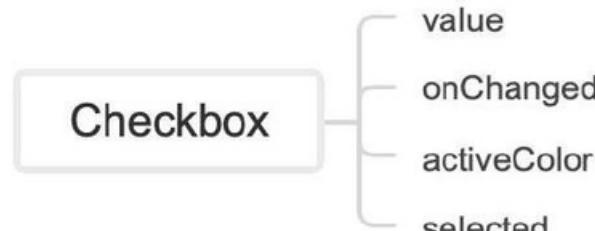
| | | |
|---------|----------------------------|-----------------------------------------|
| Gender: | <input type="radio"/> Male | <input checked="" type="radio"/> Female |
|---------|----------------------------|-----------------------------------------|

Both screenshots show other fields like Name (Sabrina), Favorite Language (Flutter), Gender (Female dropdown), and Birth date (2022-08-01). There is also a "Connect Instagram" toggle switch and a checkbox for "Agree Term & Conditions".

Checkbox

Kotak centang atau checkbox, tidak mempertahankan statusnya sendiri.. Sebagai gantinya, saat nilai checkbox berubah, widget akan memanggil callback onChanged untuk mengupdate nilai param yang dipakai di valuenya. Sebagian besar widget yang menggunakan checkbox akan mendengarkan callback onChanged dan membuat ulang checkbox dengan nilai baru untuk memperbarui tampilan visual checkbox.

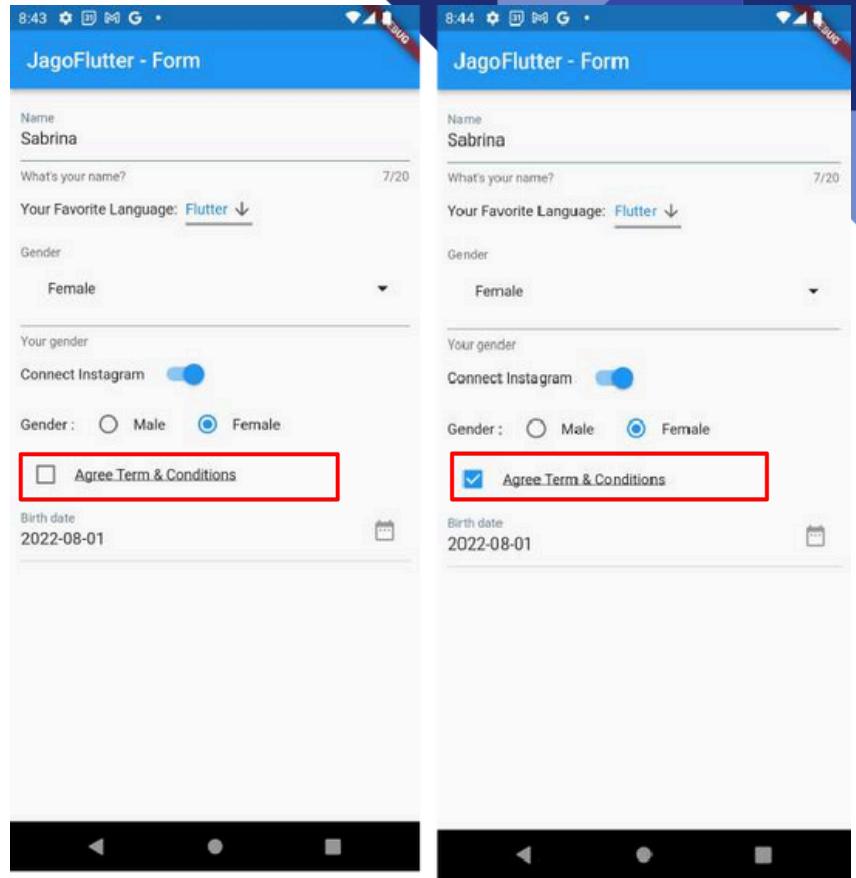
Kotak centang secara opsional dapat menampilkan tiga nilai - true, false, dan null. Ketika nilainya null, tanda hubung ditampilkan. Secara default mempunyai value false dan value checkbox harus true atau false.



Seperti contoh code disamping ini, saya membuat satu checkbox dengan value default false. Lalu ketika di klik akan mentriger onChanged dan value isCheckednya akan terupdate dengan data dari checkboxnya dan di rebuild ulang dengan bantuan setState.

```
214 |           Row(          |
215 |             children: [      |
216 |               Checkbox(       |
217 |                 value: isChecked,   |
218 |                 activeColor: Colors.blue,   |
219 |                 onChanged: (val) {           |
220 |                   setState(() {           |
221 |                     if (val != null) {        |
222 |                       isChecked = val;    |
223 |                     }                   |
224 |                   });                |
225 |                 },                  |
226 |               ), // Checkbox        |
227 |               const SizedBox(       |
228 |                 width: 4,            |
229 |               ), // SizedBox          |
230 |               const Text(          |
231 |                 'Agree Term & Conditions', |
232 |                 style: TextStyle(     |
233 |                   decoration: TextDecoration.underline, |
234 |                 ), // TextStyle      |
235 |               ), // Text            |
236 |             ], // Row           |
237 |           ), // Row
```

Hasilnya seperti gambar disamping, ketika di klik dan klik lagi.



DatePicker

showDatePicker akan menampilkan dialog yang berisi pemilihan tanggal.

Future await akan mengembalikan tanggal yang dipilih oleh user saat user mengkonfirmasi dialog. Jika pengguna membatalkan dialog, akan mengembalikan null.

Saat pemilihan tanggal pertama kali ditampilkan, ini akan menampilkan bulan tanggal awal, dengan tanggal awal dipilih.

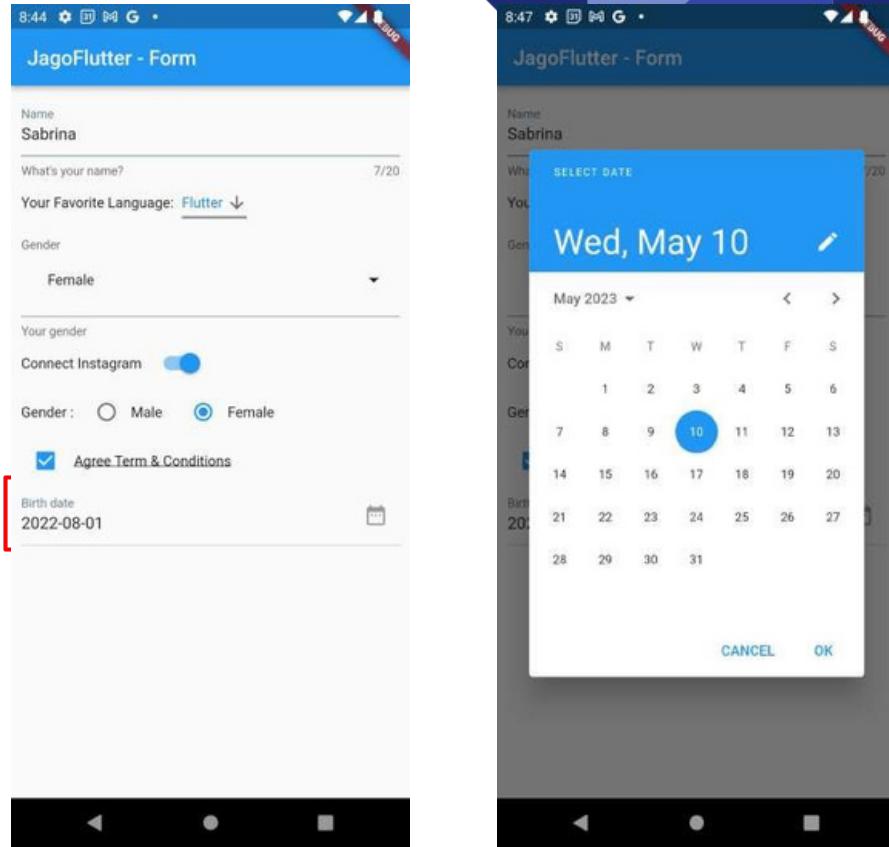
firstDate adalah tanggal paling awal yang diperbolehkan. lastDate adalah tanggal terbaru yang diperbolehkan. tanggal awal harus jatuh di antara tanggal-tanggal ini, atau sama dengan salah satunya..

```
238 |           InkWell(
239 |             onTap: () async {
240 |               DateTime? pickedDate = await showDatePicker(
241 |                 context: context,
242 |                 initialDate: DateTime.now(),
243 |                 firstDate: DateTime(2000),
244 |                 lastDate: DateTime(2100),
245 |               );
246 |               debugPrint("pickedDate: $pickedDate");
247 |             },
248 |             child: TextFormField(
249 |               initialValue: '2022-08-01',
250 |               maxLength: 20,
251 |               enabled: false,
252 |               decoration: const InputDecoration(
253 |                 labelText: 'Birth date',
254 |                 labelStyle: TextStyle(
255 |                   color: Colors.blueGrey,
256 |                 ), // TextStyle
257 |                 enabledBorder: UnderlineInputBorder(
258 |                   borderSide: BorderSide(
259 |                     color: Colors.blueGrey,
260 |                   ), // BorderSide
261 |                 ), // UnderlineInputBorder
262 |                 suffixIcon: Icon(Icons.date_range),
263 |                 helperText: "What's your birth date?",
264 |               ), // InputDecoration
265 |               onChanged: (value) {},
266 |             ), // TextFormField
267 |           ), // InkWell
```

Contoh kode disamping ini artinya ketika textformfield di klik, maka akan memanggil function onTap() lalu memanggil showDatePicker yang akan menampilkan dialog pemilihan tanggal berupa calendar.

```
238 |           InkWell(
239 |             onTap: () async {
240 |               DateTime? pickedDate = await showDatePicker(
241 |                 context: context,
242 |                 initialDate: DateTime.now(),
243 |                 firstDate: DateTime(2000),
244 |                 lastDate: DateTime(2100),
245 |               );
246 |               debugPrint("pickedDate: $pickedDate");
247 |             },
248 |             child: TextFormField(
249 |               initialValue: '2022-08-01',
250 |               maxLength: 20,
251 |               enabled: false,
252 |               decoration: const InputDecoration(
253 |                 labelText: 'Birth date',
254 |                 labelStyle: TextStyle(
255 |                   color: Colors.blueGrey,
256 |                 ), // TextStyle
257 |                 enabledBorder: UnderlineInputBorder(
258 |                   borderSide: BorderSide(
259 |                     color: Colors.blueGrey,
260 |                   ), // BorderSide
261 |                 ), // UnderlineInputBorder
262 |                 suffixIcon: Icon(Icons.date_range),
263 |                 helperText: "What's your birth date?",
264 |               ), // InputDecoration
265 |               onChanged: (value) {},
266 |             ), // TextFormField
267 |           ), // InkWell
```

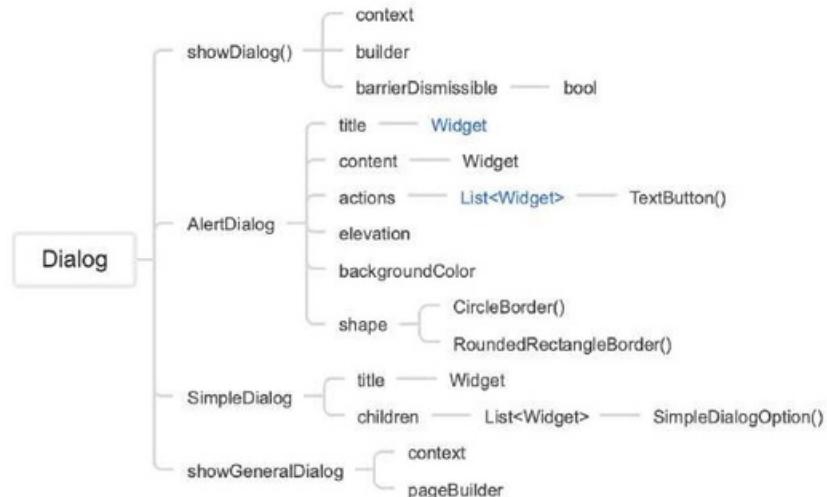
Code tersebut bila kita running hasilnya akan seperti gambar disamping ini. Pertama tampilan berupa textfield dan ketika diklik maka dialog calendar untuk memilih tanggal akan muncul dan kita dapat meng select salah satunya lalu kita ada pilihan ok atau cancel.



Dialog

Dialog adalah jenis jendela modal yang muncul di depan konten aplikasi untuk memberikan informasi penting atau meminta keputusan. Dialog menonaktifkan semua fungsionalitas aplikasi saat muncul, dan tetap berada di layar hingga dikonfirmasi, ditutup, atau tindakan yang diperlukan telah dilakukan.

Dialog sengaja mengganggu, sehingga harus digunakan secara cermat.



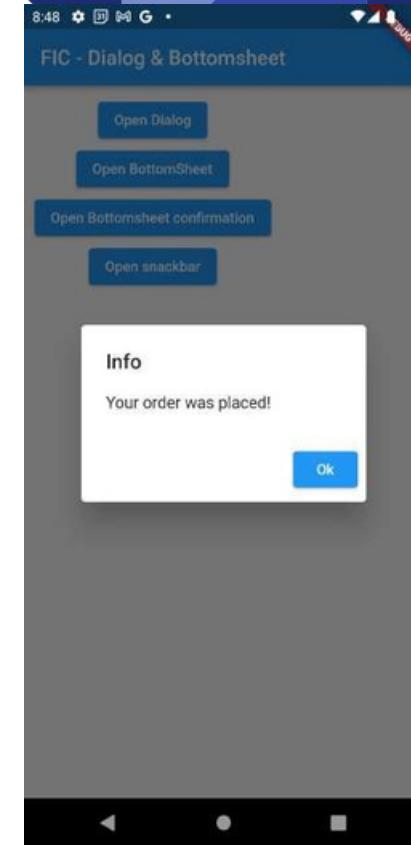
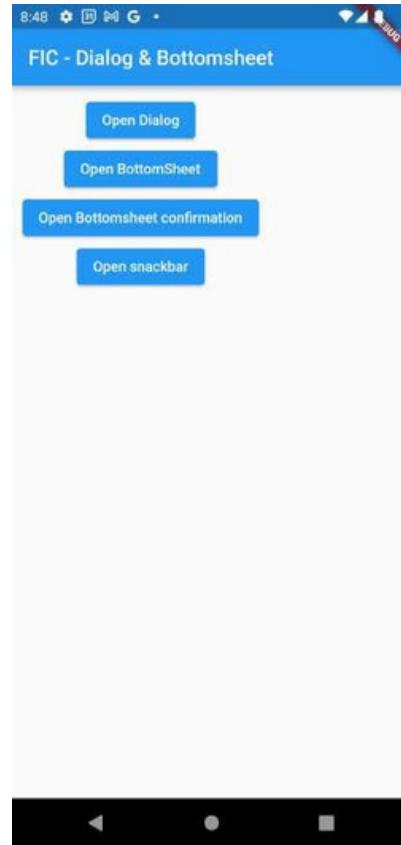
Dialog

Contoh code disamping ini saya mempunya button dengan text Open Dialog, dan ketika di klik maka akan menjalankan onPressed dan menjalankan showDialog, disini saya mengembalikan AlertDialog dengan berisi title, content dan action.

```
21   ElevatedButton(          ↴
22     onPressed: () async {    ↴
23       await showDialog<void>(    ↴
24         context: context,    ↴
25         barrierDismissible: true,    ↴
26         builder: (BuildContext context) {    ↴
27           return AlertDialog(    ↴
28             title: const Text('Info'),    ↴
29             content: SingleChildScrollView(    ↴
30               child: ListBody(    ↴
31                 children: const [    ↴
32                   Text('Your order was placed!'),    ↴
33                 ],    ↴
34               ), // ListBody    ↴
35             ), // SingleChildScrollView    ↴
36             actions: [    ↴
37               ElevatedButton(    ↴
38                 style: ElevatedButton.styleFrom(    ↴
39                   backgroundColor: Colors.blue,    ↴
40                 ),    ↴
41                 onPressed: () {    ↴
42                   Navigator.pop(context);    ↴
43                 },    ↴
44                 child: const Text("Ok"),    ↴
45               ), // ElevatedButton    ↴
46             ],    ↴
47           );    ↴
48         },    ↴
49       );    ↴
50     },    ↴
51     child: const Text('Open Dialog'),    ↴
52   ), // ElevatedButton
```

Dialog

Hasil dari kode tersebut jika kita running akan seperti pada gambar disamping. Dimana button Open Dialog ketika di klik, maka akan muncul modal dialog yang menutupi layar belakangnya sampai kita klik button ok untuk kembali ke aplikasi.



BottomSheet

Bottom sheet adalah tampilan tambahan yang muncul dibagian bawah yang umum digunakan pada ponsel. Ada tiga penggunaan yang cocok untuk tiap kasusnya:

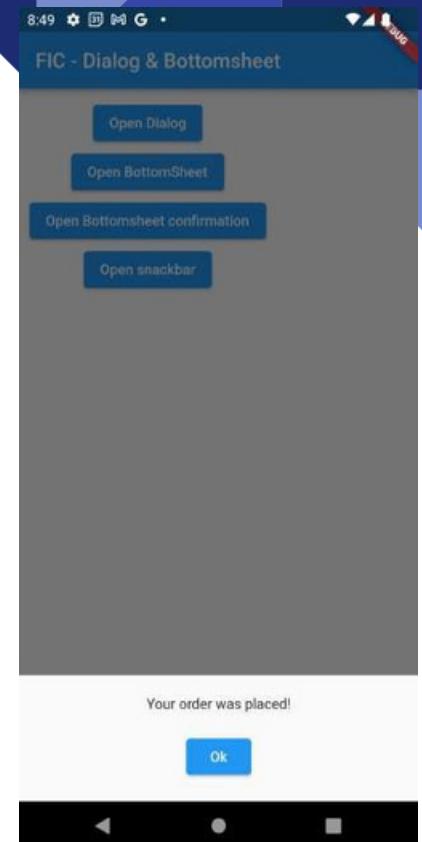
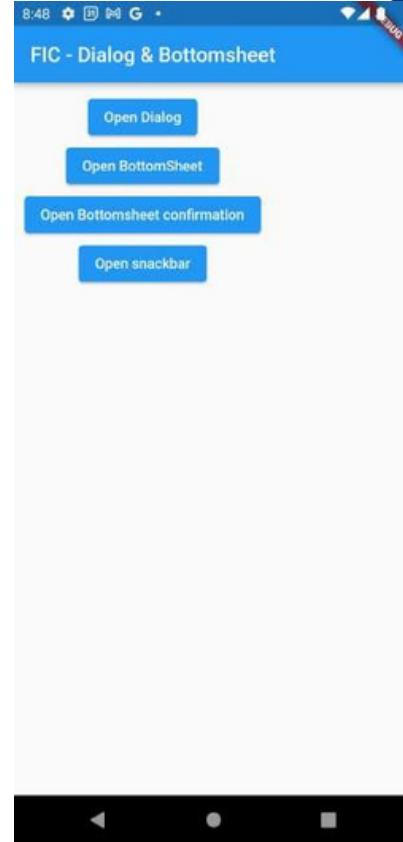
- Bottom sheet standar: menampilkan konten yang melengkapi konten utama layar. Bottom sheet tetap terlihat saat user berinteraksi dengan konten utama.
- Bottom sheet modal adalah alternatif untuk dialog sederhana di seluler dan menyediakan ruang untuk item tambahan, deskripsi yang lebih panjang, dan ikonografi. Mereka harus ditutup untuk berinteraksi dengan konten aplikasi utamanya..
- Bottom sheet yang menyediakan permukaan kecil yang diciutkan yang dapat diperluas oleh user untuk mengakses fitur atau tugas utama. Mereka menawarkan akses tetap bottom sheet standar dengan ruang dan fokus bottom sheet modal.



BottomSheet

Contoh kode di samping, saya mempunya button open bottomsheet dan ketika saya klik maka akan menjalankan onPressed function dan mentriger showModalBottomSheet sehingga widget yang di return oleh nya akan muncul di bagian bawah seperti gambar di slide selanjutnya.

```
56     |   ElevatedButton(
57     |   | onPressed: () async {
58     |   |   await showModalBottomSheet<void>(
59     |   |   context: context,
60     |   |   builder: (BuildContext context) {
61     |   |       return Padding(
62     |   |       | padding: const EdgeInsets.all(20.0),
63     |   |       | child: SizedBox(
64     |   |       |   width: MediaQuery.of(context).size.width,
65     |   |       |   child: Column(
66     |   |       |       mainAxisSize: MainAxisSize.center,
67     |   |       |       mainAxisAlignment: MainAxisAlignment.end,
68     |   |       |       children: [
69     |   |       |           const Text('Your order was placed!'),
70     |   |       |           const SizedBox(
71     |   |       |               height: 20.0,
72     |   |       |           ), // SizedBox
73     |   |       |           ElevatedButton(
74     |   |       |               style: ElevatedButton.styleFrom(
75     |   |       |                   backgroundColor: Colors.blue,
76     |   |       |               ),
77     |   |       |               onPressed: () {
78     |   |       |                   Navigator.pop(context);
79     |   |       |               },
80     |   |       |               child: const Text("Ok"),
81     |   |       |           ), // ElevatedButton
82     |   |       ],
83     |   |   ), // Column
84     |   |   ), // SizedBox
85     |   | ); // Padding
86     |   },
87     |   );
88     |   ),
89     |   child: const Text('Open BottomSheet'),
90     | ), // ElevatedButton
```



BottomSheet

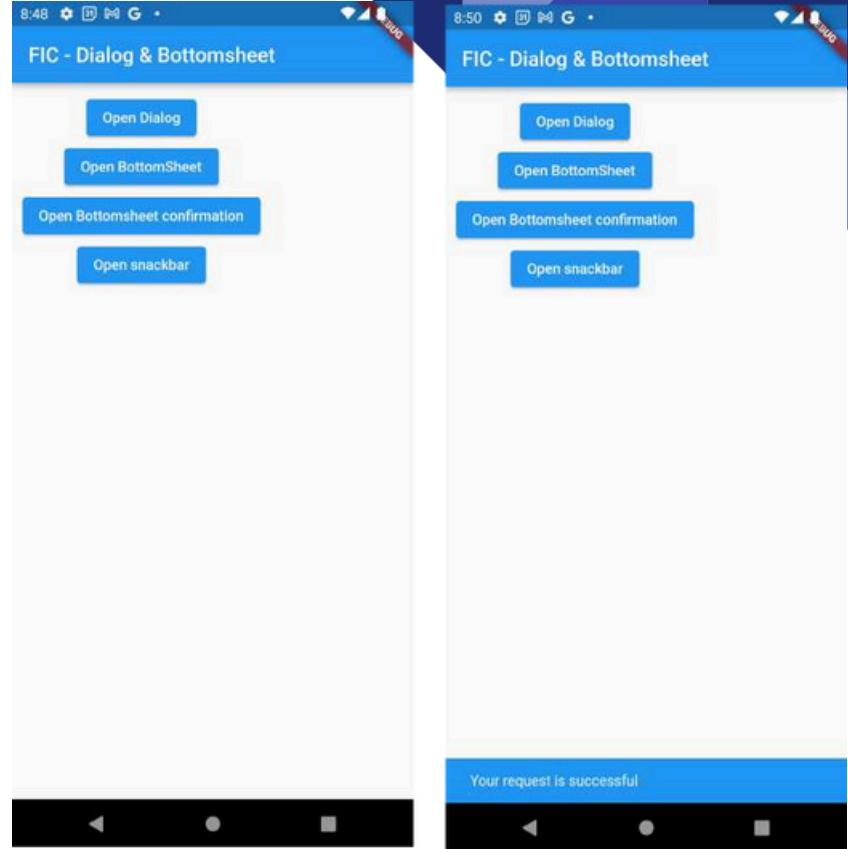
Ketika button open butomsheet diklik akan muncul bottomsheet dibagian bawah.

Snackbar

Widget ini bermanfaat untuk memberi tahu user secara singkat saat tindakan tertentu terjadi. Misalnya, saat user menghapus pesan dalam daftar, Anda mungkin ingin memberitahu mereka bahwa pesan tersebut telah dihapus. Anda bahkan mungkin ingin memberi mereka opsi untuk membatalkan tindakan.

Seperti contoh kode di samping, snackbar saya trigger ketika saya klik button Open Snackbar.

```
163           ↴ ElevatedButton(
164             onPressed: () {
165               ScaffoldMessenger.of(context).showSnackBar(
166                 const SnackBar(
167                   backgroundColor: Colors.blue,
168                   content: Text('Your request is successful'),
169                 ), // SnackBar
170               );
171             },
172           child: const Text('Open snackbar'),
173         ), // ElevatedButton
```



Snackbar

Hasil kode sebelumnya ketika di running akan seperti ini. Ketika kita klik button open snackbar maka flutter akan menjalankan snackbar seperti di gambar sebelah kanan. Ada snackbar di bagian bawah layar dengan latar belakang biru dan text berwarna putih, seperti kode flutter yang kita tulis sebelumnya.



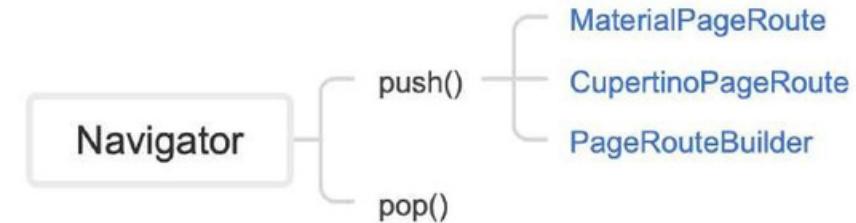
**Dalam basic Navigation kali ini, kita akan
mengenal:**

- Navigator push
- Navigator pop
- Bottom Navigation Bar
- TabBar
- Drawer
- SliverAppBar

Navigation push

Sebagian besar app berisi beberapa layar untuk menampilkan berbagai jenis informasi. Misalnya, aplikasi mungkin memiliki layar yang menampilkan produk. Saat pengguna mengetuk gambar suatu produk, layar baru menampilkan detail tentang produk tersebut.

Terminologi: Di Flutter, layar dan halaman disebut rute.



Navigation push

Di Android, rute setara dengan activity. Di iOS, rute setara dengan ViewController. Di Flutter, rute hanyalah sebuah widget.

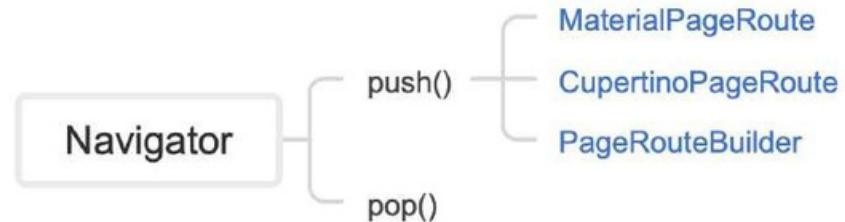
Aturan ini menggunakan Navigator untuk menavigasi ke rute baru.

Beberapa bagian selanjutnya menunjukkan cara bernavigasi di antara dua rute, menggunakan langkah-langkah berikut:

Buat dua rute:

Arahkan ke rute kedua : Navigator.push().

Kembali ke rute pertama : Navigator.pop().



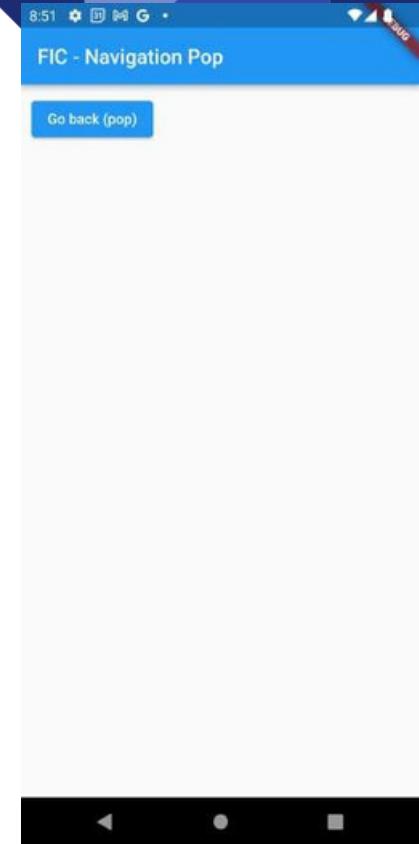
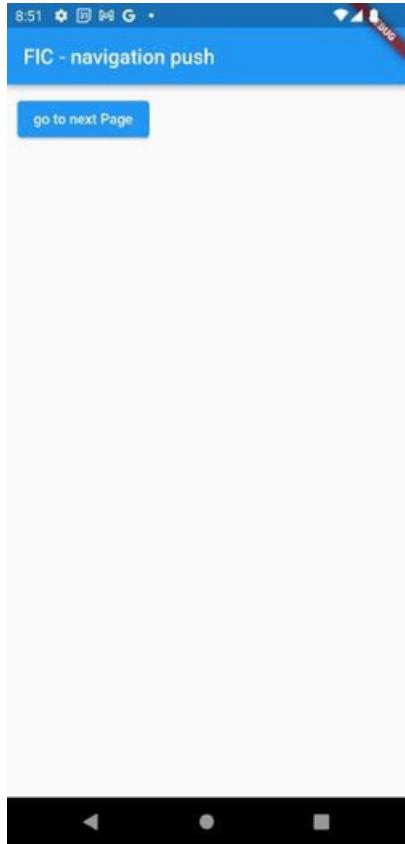
Navigation push

Contoh code disamping ini kita akan praktikan push ke route page selanjutnya dengan cara klik button go to next page.

```
20 |   body: Container(
21 |     padding: const EdgeInsets.all(10.0),
22 |   child: Column(
23 |     children: [
24 |       ElevatedButton(
25 |         onPressed: () {
26 |           Navigator.of(context)
27 |             .push(MaterialPageRoute(builder: (context) {
28 |               return const NavigationPop();
29 |             })); // MaterialPageRoute
30 |           },
31 |           child: const Text('go to next Page'),
32 |         ), // ElevatedButton
33 |         ],
34 |       ), // Column
35 |     ), // Container
```

Navigation push

Tampilannya akan seperti ini, ketika button go to next page di klik, maka akan pindah ke page sebelah kanan.



Navigation pop

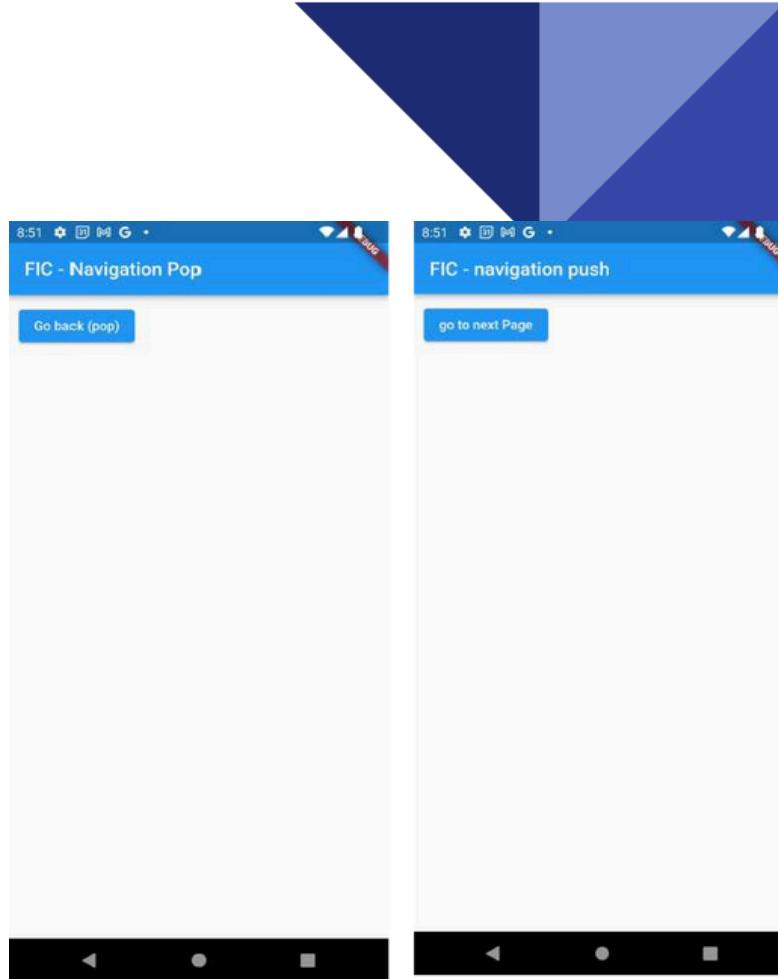
Navigation pop ini berfungsi untuk kembali ke page sebelumnya setelah tampil karena navigator.pop.

Code disamping memperlihatkan kondisi ketika page 2 dengan button go back diklik, maka akan kembali ke page pertama.

```
17 |   body: Container(
18 |     padding: const EdgeInsets.all(10.0),
19 |     child: Column(
20 |       children: [
21 |         ElevatedButton(
22 |           onPressed: () {
23 |             Navigator.of(context).pop();
24 |           },
25 |           child: const Text('Go back (pop)'),
26 |         ), // ElevatedButton
27 |       ],
28 |     ), // Column
29 |   ), // Container
```

Navigation pop

Gambar sebelah kiri ketika button go back diklik maka akan kembali ke page 1 seperti gambar sebelah kanan.

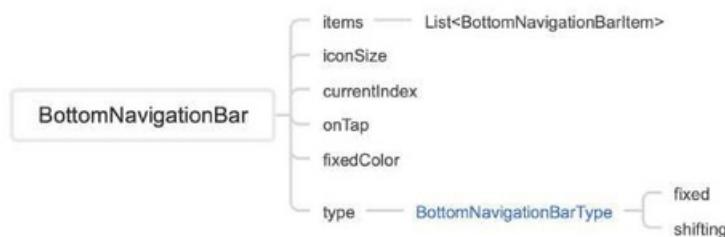


Bottom Navigation Bar

Ini adalah widget yang ditampilkan di bagian bawah aplikasi untuk memilih diantara beberapa tampilan, biasanya antara tiga dan lima.

Bottom navigation bar terdiri dari beberapa item dalam bentuk label teks, ikon, atau keduanya, yang diletakkan di atas material. Ini menyediakan navigasi cepat antara tampilan tingkat atas aplikasi. Untuk layar yang lebih besar, navigasi samping mungkin lebih cocok.

Bottom navigation bar biasanya digunakan bersamaan dengan Scaffold, yang disediakan sebagai argumen Scaffold.bottomNavigationBar.



Bottom Navigation Bar

Disamping ini contoh code bottom navigation bar dimana terdapat item sebanyak menuItems yang saya sudah didefinisikan terlebih dahulu ke dalam variable list.

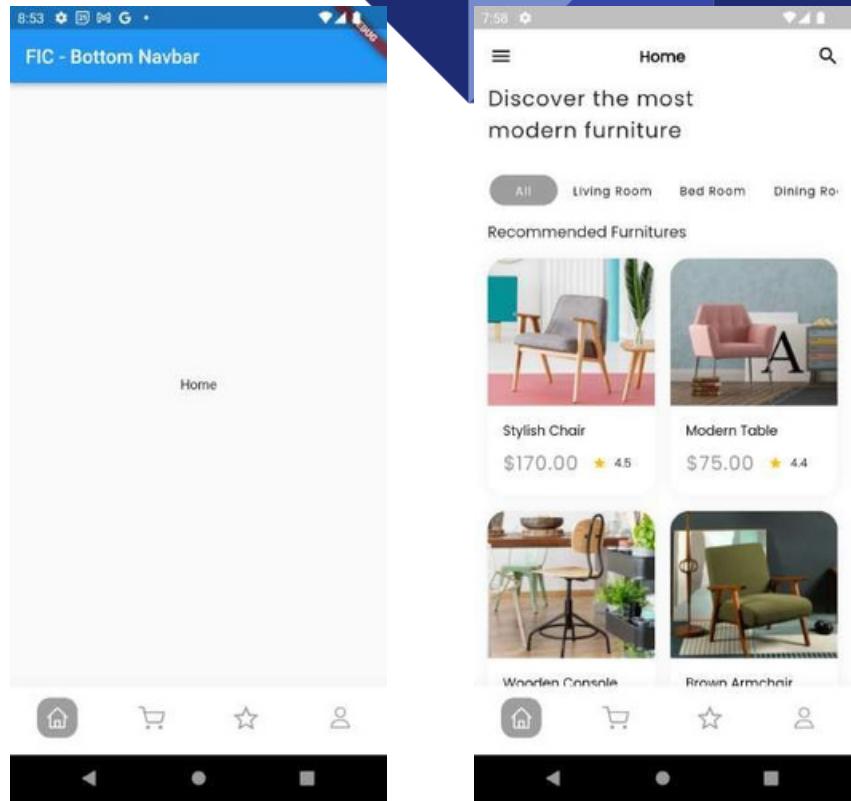
Properties yang sering dipakai adalah type, items, currentIndex, onTap, selectedItemColor.

Hasil dari code ini ada di slide selanjutnya

```
55   |--bottomNavigationBar: BottomNavigationBar()
56   |  backgroundColor: Colors.white,
57   |  showUnselectedLabels: false,
58   |  showSelectedLabels: false,
59   |  unselectedItemColor: Colors.black87,
60   |  elevation: 32.0,
61   |  type: BottomNavigationBarType.fixed,
62   |  selectedLabelStyle: const TextStyle(
63   |    height: 1.5,
64   |    fontSize: 12,
65   |  ), // TextStyle
66   |  unselectedLabelStyle: const TextStyle(
67   |    height: 1.5,
68   |    fontSize: 12,
69   |  ), // TextStyle
70   |  items: menuItems.map((i) {
71   |    return BottomNavigationBarItem(
72   |      activeIcon: Container(
73   |        padding: const EdgeInsets.all(10),
74   |        decoration: const BoxDecoration(
75   |          color: Colors.grey,
76   |          borderRadius: BorderRadius.all(Radius.circular(14)),
77   |        ), // BoxDecoration
78   |        child: SvgPicture.asset(
79   |          if('icon'),
80   |            color: Colors.white,
81   |          ), // SvgPicture.asset
82   |        ), // Container
83   |      icon: SvgPicture.asset(
84   |        if('icon'),
85   |          color: Colors.grey,
86   |        ), // SvgPicture.asset
87   |        label: if('label'),
88   |      ); // BottomNavigationBarItem
89   |    }).toList(),
90   |    currentIndex: _selectedIndex,
91   |    selectedItemColor: Colors.blue,
92   |    onTap: _onItemTapped,
93   |  ), // BottomNavigationBar
```

Bottom Navigation Bar

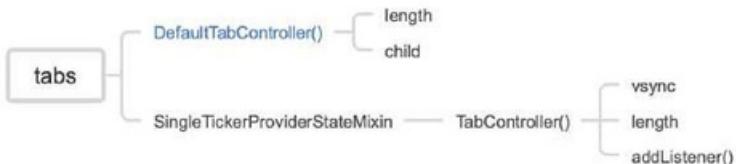
Hasilnya ada di gambar sebelah kiri dan code ini nanti bisa dikembangkan untuk digabungkan ke dalam sample latihan slicings aplikasi furniture.



TabBar

Widget ini menampilkan deretan tab horizontal.

Biasanya dibuat sebagai bagian AppBar.bottom dari AppBar dan bersamaan dengan TabBarView.



TabBar

Disamping adalah sample code untuk membuat tabbar dengan tabbarview dibawahnya.

```
29   return Scaffold(  
30     appBar: AppBar(  
31       title: const Text("Binar - Tabbar"),  
32     bottom: TabBar(  
33       controller: _tabController,  
34       tabs: const [  
35         Tab(  
36           icon: Icon(Icons.directions_boat),  
37         ), // Tab  
38         Tab(  
39           icon: Icon(Icons.directions_bus),  
40         ), // Tab  
41         Tab(  
42           icon: Icon(Icons.directions_ferry),  
43         ), // Tab  
44         ],  
45       ), // TabBar  
46     ), // AppBar  
47     body: TabBarView(  
48       controller: _tabController,  
49       children: const [  
50         Center(  
51           child: Text('Tab 1'),  
52         ), // Center  
53         Center(  
54           child: Text('Tab 2'),  
55         ), // Center  
56         Center(  
57           child: Text('Tab 3'),  
58         ), // Center  
59       ],  
60     ), // TabBarView
```



TabBar

Hasil kode nya bila kita running akan seperti ini.

Drawer

Di app yang menggunakan Desain Material, ada dua opsi utama untuk navigasi: tab dan drawer. Jika tidak ada cukup ruang untuk mendukung tab, drawer memberikan alternatif yang praktis.

Di Flutter, gunakan widget Drawer yang dikombinasikan dengan Scaffold untuk membuat tata letak dengan panel di samping Desain Material. Aturan ini menggunakan langkah-langkah berikut:

Buat scaffold.

Tambahkan drawer.

Isi drawer dengan item.

Tutup drawer secara terprogram.



Drawer

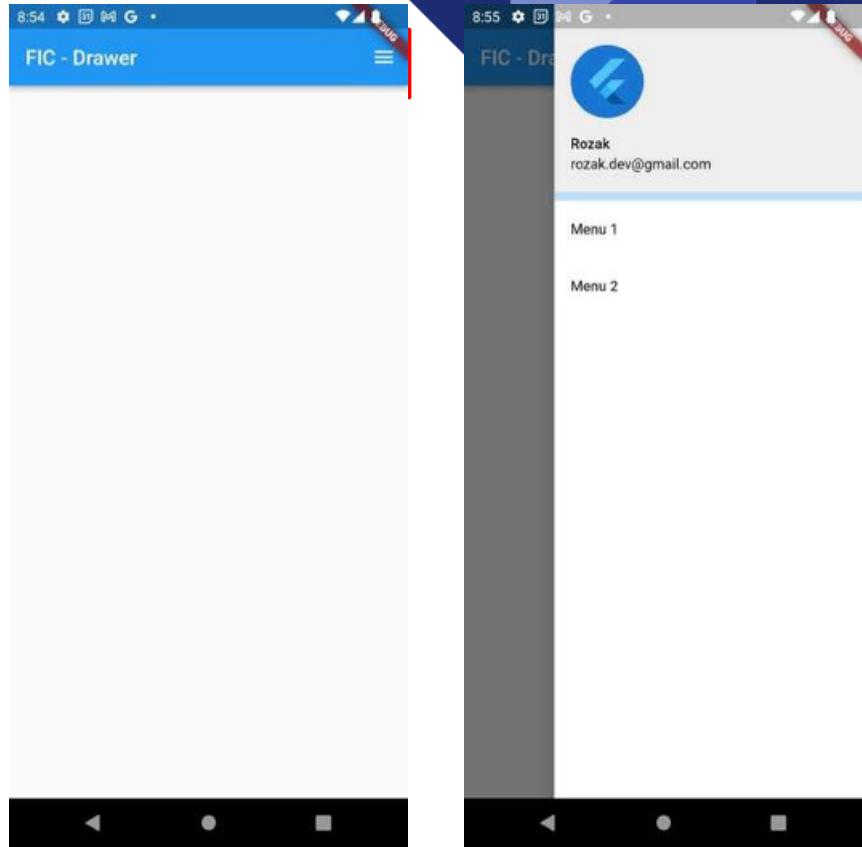
Disamping ini adalah contoh code dalam penggunaan drawer, disini saya menggunakan endDrawer yang artinya nanti munculnya dari sebelah kanan. Berisikan UserAccountDrawerHeader, dan juga listTile.

```
18  endDrawer: Drawer(
19    child: Container(
20      color: Colors.white,
21      child: ListView(
22        padding: const EdgeInsets.all(0),
23        children: [
24          Container(
25            color: Colors.blue[100],
26            child: UserAccountsDrawerHeader(
27              currentAccountPicture: const CircleAvatar(
28                child: FlutterLogo(size: 50),
29              ), // CircleAvatar
30              decoration: BoxDecoration(color: Colors.grey[200]),
31              accountName: const Text(
32                'Sabrina',
33                style: TextStyle(
34                  color: Colors.black,
35                ), // TextStyle
36                ), // Text
37                accountEmail: const Text(
38                  'sabrina.dev@gmail.com',
39                  style: TextStyle(
40                    color: Colors.black,
41                  ), // TextStyle
42                  ), // Text
43                ), // UserAccountsDrawerHeader
44                ), // Container
45                ListTile(
46                  title: const Text('Menu 1'),
47                  onTap: () {
48                    Navigator.of(context).pop();
49                  },
50                ), // ListTile
51                ListTile(
52                  title: const Text('Menu 2'),
53                  onTap: () {},
54                ), // ListTile
55                ], // You, 8 hours ago • init binar academy flutter app widget
56              ), // ListView
57            ), // Container
58          ), // Drawer
```

Drawer

Hasil code nya bila dirunning akan seperti ini.

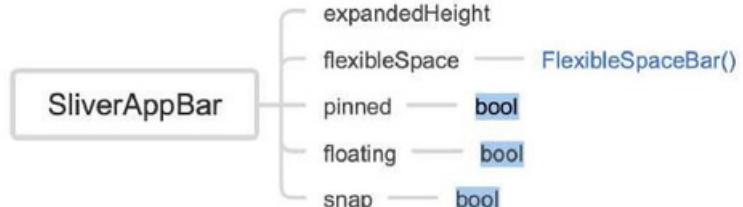
Nanti akan otomatis ada button hamburger yang menandakan scaffold ini memberikan nilai pada parameter drawer atau endDrawer. Ketika diklik maka akan muncul drawer dari samping yang berisi widget yang kita buat secara manual. Dalam contoh ini saya buat ada header yang berisi avatar name, dan email. Lalu bagian bawahnya berupa list menu yang dapat dipilih dan bisa dipakai untuk navigasi berpindah page.



SliverAppBar

Widget ini terintegrasi dengan CustomScrollView. AppBar terdiri dari toolbar dan kemungkinan widget lainnya, seperti TabBar dan FlexibleSpaceBar. AppBar biasanya memaparkan satu atau beberapa tindakan umum dengan IconButton yang secara opsional diikuti oleh PopupMenuItem untuk operasi yang kurang umum.

Sliver app bar biasanya digunakan sebagai anak pertama dari CustomScrollView, yang memungkinkan AppBar berintegrasi dengan tampilan scroll sehingga tingginya dapat bervariasi sesuai dengan offset atau melayang di atas konten lain dalam tampilan scroll. Untuk AppBar dengan ketinggian tetap di bagian atas layar, lihatlah AppBar, yang menggunakan slot dari Scaffold.appBar.



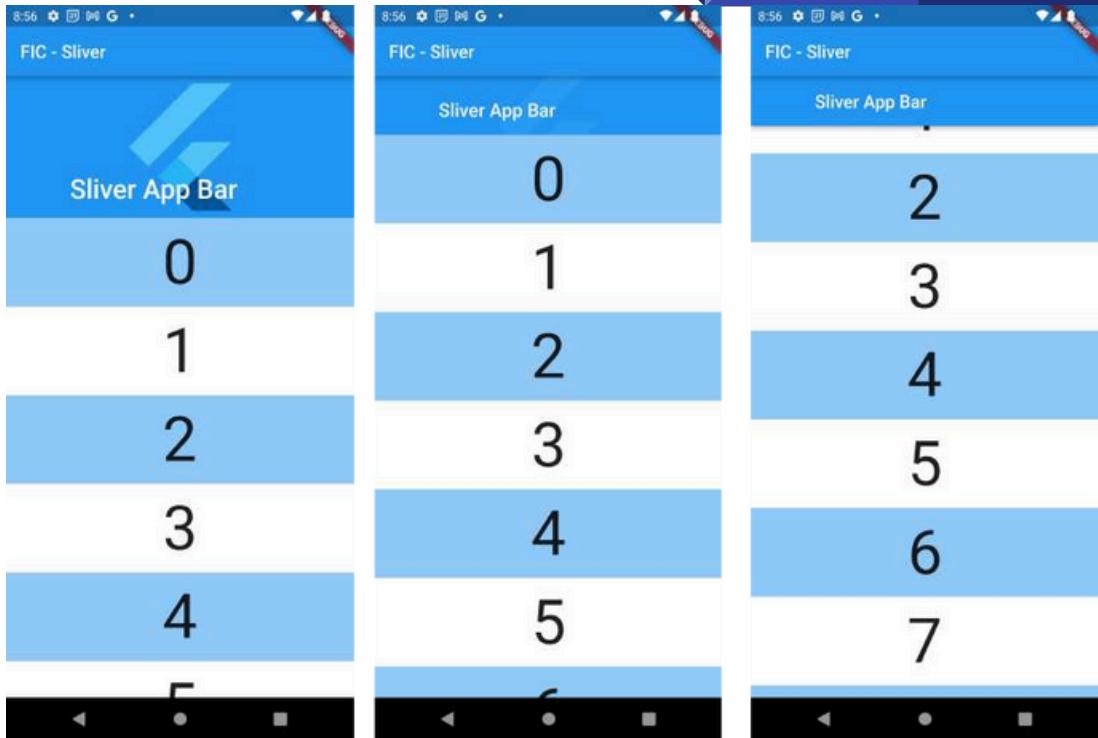
SliverAppBar

Disamping adalah contoh code penggunaan sliverappbar yang dimulai dulu dari baris ke 21 yaitu widget CustomScrolView yang didalamnya berisi list sliver dan anak pertama adalah sliverappbar dimana dia punya properties pinned, snap, floating yang bernilai true atau false. Terdapat properties expandedheight juga untuk set tinggi dari sliverappbar, dan terdapat flexiblespace jika scroll terjadi. Lalu di ikuti oleh sliverlist bagian dari customscrolview.

```
21   |   body: CustomScrollView(
22   |   |   slivers: [
23   |   |   |   SliverAppBar(
24   |   |   |   |   pinned: pinned,
25   |   |   |   |   snap: snap,
26   |   |   |   |   floating: floating,
27   |   |   |   |   expandedHeight: 160,
28   |   |   |   |   flexibleSpace: const FlexibleSpaceBar(
29   |   |   |   |   |   title: Text(
30   |   |   |   |   |   |   'Sliver App Bar',
31   |   |   |   |   |   |   ), // Text
32   |   |   |   |   |   |   background: FlutterLogo(),
33   |   |   |   |   |   |   ), // FlexibleSpaceBar
34   |   |   |   |   |   ), // SliverAppBar
35   |   |   |   |   SliverList(
36   |   |   |   |   |   delegate: SliverChildBuilderDelegate((context, index) {
37   |   |   |   |   |   |   return Container(
38   |   |   |   |   |   |   |   color: index.isOdd ? Colors.white : Colors.blue[200],
39   |   |   |   |   |   |   |   height: 100,
40   |   |   |   |   |   |   |   child: Center(
41   |   |   |   |   |   |   |   |   child: Text(
42   |   |   |   |   |   |   |   |   |   '$index',
43   |   |   |   |   |   |   |   |   |   textScaleFactor: 5,
44   |   |   |   |   |   |   |   |   ), // Text
45   |   |   |   |   |   |   |   |   ), // Center
46   |   |   |   |   |   |   |   ), // Container
47   |   |   |   |   |   |   |   ), childCount: 20), // SliverChildBuilderDelegate
48   |   |   |   |   |   ), // SliverList
49   |   |   ],
50   |   |   ), // CustomScrollView
```

SliverAppBar

Bila kita running hasilnya akan seperti ini. Dimana pertama sliver appbar akan menempati bagiannya dan ketika kita scroll ke atas dia akan fleksible mengikuti scroll sampai tinggal setinggi appbar.





ITB Stikom
Ambon

Terima Kasih