Hari-15-Flutter Navigation

Pada Mater kali ini kita akan membahas navigasi routing pada flutte. dalam sebuah alikasi mobile biasanya memiliki full-screen elemen yang disebut "screen" atau "page". di Flutter element tersebut disebut route dan di kelola oleh widget Navigator. Widget Navigator berfungsi untuk menampilakn konten ke halaman atau layar baru. jika pada native android, Navigator route di namakan activity dan di ios sebagai viewController. wodget Navigator bekerja seperti tumpukan layar(stack), ia menggunakan prinsip LIFO(Last-In, First Out). ada 2 mthod yang dapat digunakan untuk navigtor pada widget yaitu:

1. navigator.push() : Methode push digunakan untuk menambahkan rute lain keatas tumpukan screen(stack)saat ini. Halaman baru ditampilkan diatas halaman sebelumnya.
2. Navigator.pop(): Methode pop menghapus rute paling atas dari tumpukan. ini menampilkan halaman sebelumnya kepada pengguna. Simple Routing Flutter Penggunaan widget Navigator untuk routing di flutter yaitu seperti dibawah ini `Navigator.push(`

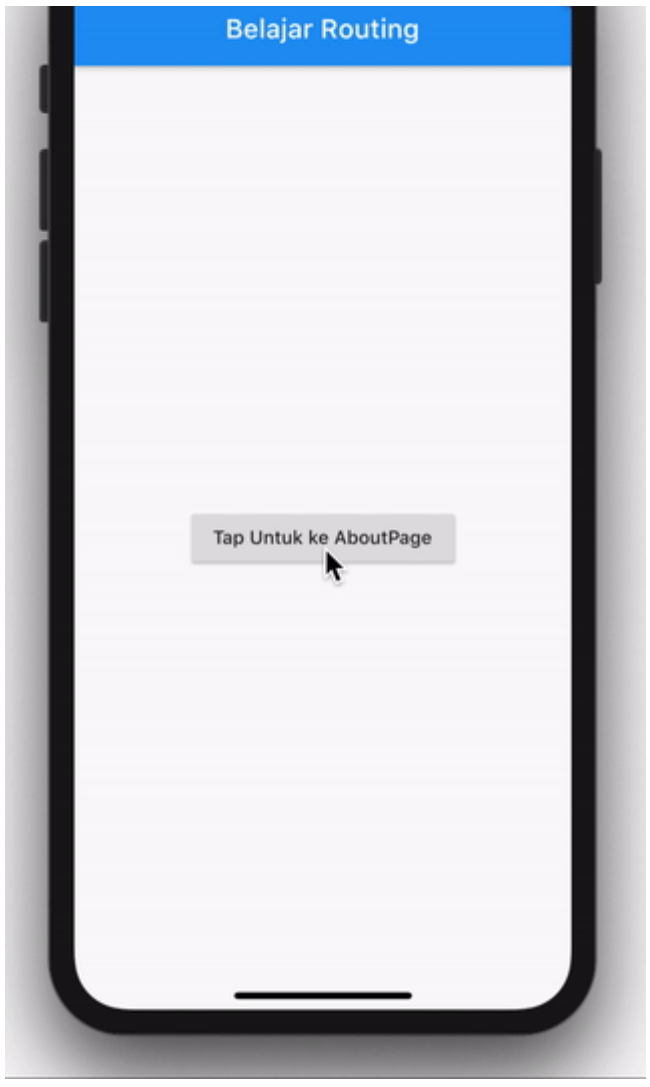
```
context,
   MaterialPageRoute(builder: (context) {
      return AboutPage()
  })
);
```

Dalam contoh penerapannya, kita akan membuat routing sederhana menggunakan metode Navigator.push untuk navigasi ke layar baru dan Navigator.pop untuk kembali ke layar sebelumnya.

`main.dart`

```
import 'package:flutter/material.dart';
void main() {
  runApp(MaterialApp(
    home: HomePage(),
  ));
}
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Belajar Routing'),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            Route route = MaterialPageRoute(builder: (context) => AboutPage());
            Navigator.push(context, route);
          },
          child: Text('Tap Untuk ke AboutPage'),
        ),
      ),
    );
  }
}
class AboutPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tentang Aplikasi'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Kembali'),
        ),
      ),
    );
  }
}
```

Contoh kode di atas membuat dua stateless widget dimana halaman awal menggunakan HomePage dan berpindah ke AboutPage saat tombol di tap. Tampilannya seperti gambar di bawah ini



Named Routing

Sesuai namanya, Named Routing yaitu memberi nama pada routing dengan tujuan untuk mempermudah dalam membaca dan menentukan arah dari suatu navigasi. Sedikit berbeda dengan simple routing, disini kita akan menggunakan Navigator.pushNamed untuk menuju ke halaman baru namun tetap menggunakan Navigator.pop untuk kembali ke halaman sebelumnya.

Navigator.pushNamed membutuhkan dua properti wajib yaitu context dan string sebagai nama routenya. Kita juga dapat mengirim parameter object ke dalam route.

```
Navigator.pushNamed(context, String, {Object});
```

```
    initialRoute: '/',
    routes: <String, WidgetBuilder>{
      '/': (context) => HomePage(),
      '/about': (context) => AboutPage(),
    },
  ));
}
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Belajar Routing'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(context, '/about');
          },
          child: Text('Tap Untuk ke AboutPage'),
        ),
      ),
    );
  }
}
class AboutPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tentang Aplikasi'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Kembali'),
        ),
      ),
    );
  }
}
```

Contoh kode diatas akan sama dengan gambar 1.1.

Perhatikan potongan kode pada main function dibawah ini

```
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: <String, WidgetBuilder>{
      '/': (context) => HomePage(),
      '/about': (context) => AboutPage(),
    },
  ));
}
```

method initialRoute tidak wajib jika dalam routes kita memiliki “/”. Kita juga dapat menggunakan properti home sebagai initialRoute seperti di bawah ini.

```
void main() {
  runApp(MaterialApp(
    home: HomePage(),
    routes: <String, WidgetBuilder>{
      '/': (context) => HomePage(),
      '/about': (context) => AboutPage(),
    },
  ));
}
```

Route Generator

Meskipun named routing terlihat lebih baik dari simple routing, namun apabila ingin membuat aplikasi dengan skala yang cukup besar atau ingin mengatur animasi transisi dari tiap routing maka onGenerateRoute merupakan pilihan yang tepat. Dalam tutorial route generator ini kita akan coba memisahkan antara main, screen dan routing file

Main.dart

```
  runApp(MaterialApp(
    onGenerateRoute: RouteGenerator.generateRoute,
  ));
}
```

screen.dart

```
import 'package:flutter/material.dart';
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Belajar Routing'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/about');
              },
              child: Text('Tap Untuk ke AboutPage'),
            ),
            RaisedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/halaman-404');
              },
              child: Text('Tap Halaman lain'),
            ),
          ],
        ),
      ),
    );
  }
}
class AboutPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Tentang Aplikasi'),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Kembali'),
        ),
      ),
    );
  }
}
```

routes.dart

```
  static Route<dynamic> generateRoute(RouteSettings settings) {
    // jika ingin mengirim argument
    // final args = settings.arguments;
    switch (settings.name) {
      case '/':
        return MaterialPageRoute(builder: (_) => HomePage());
      case '/about':
        return MaterialPageRoute(builder: (_) => AboutPage());
        // return MaterialPageRoute(builder: (_) => AboutPage(args));
      default:
        return _errorRoute();
    }
  }
  static Route<dynamic> _errorRoute() {
    return MaterialPageRoute(builder: (_) {
      return Scaffold(
        appBar: AppBar(title: Text("Error")),
        body: Center(child: Text('Error page')),
      );
    });
  }
}
```

pada contoh diatas kita juga menambahkan error page dimana jika routing tidak ditemukan maka akan menampikan halaman error tersebut

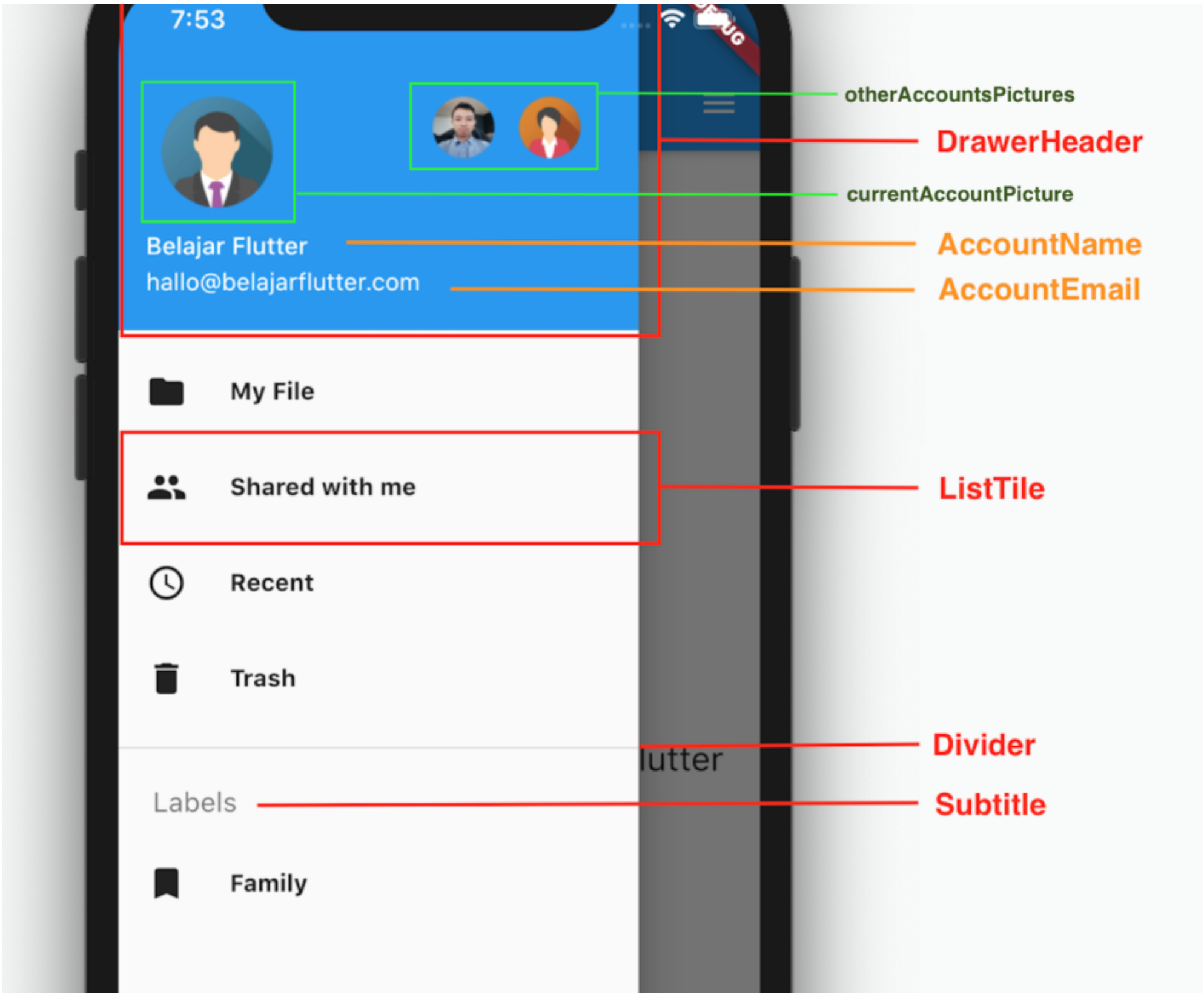
Mengenal Drawer Widget di Flutter

Sebelum ke tahap cara membuat navigation drawer, ada baiknya kita mengerti tentang Drawer itu sendiri. Drawer widget merupakan single child widget yang artinya hanya dapat memiliki satu child widget di dalamnya. Karena Drawer hanya memiliki properti child dan bukan children, maka untuk menampatkan item-item lain bisa menggunakan ListView widget. Sebenarnya tidak ada keharusan menggunakan ListView namun keuntungan menggunakan ListView widget dibandingkan dengan column atau widget lainnya yaitu untuk memudahkan dalam mengatur list Item dan vertical scrolling apabila item menu melebihi tinggi layar.

Anatomi Drawer

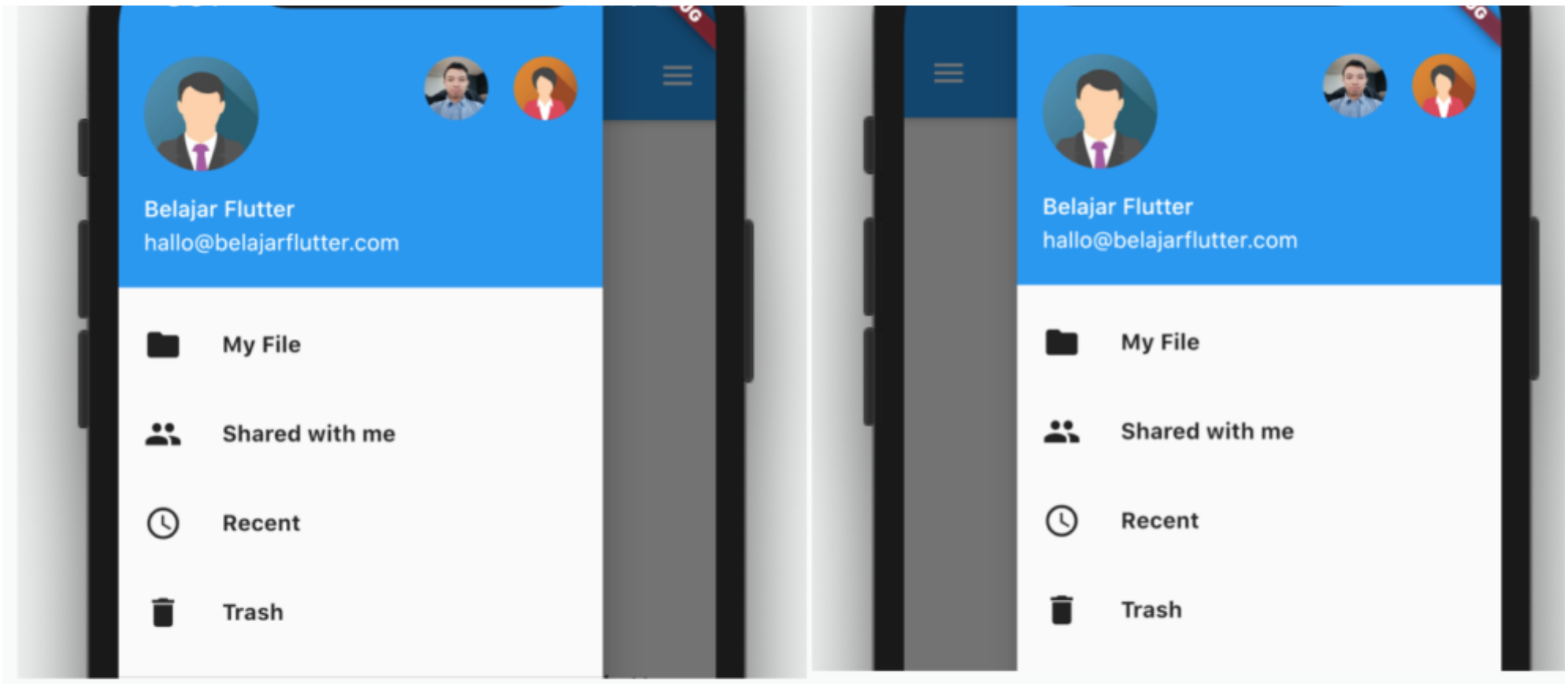
Anatomi dari sebuah navigation drawer menurut material design itu ada delapan (8) point, namun disini kita sederhanakan saja menjadi 4 bagian yaitu

1. Header,
2. List Item,
3. Divider dan
4. Subtitle



Jenis Drawer

Jenis Drawer navigasi pada flutter dibagi menjadi dua sesuai dengan letak posisi drawer itu sendiri yaitu drawer dan endDrawer. Drawer yang muncul di sebelah kiri dinamakan drawer, sedangkan apabila muncul dari sebelah kanan dinamakan endDrawer karena menggunakan properti endDrawer



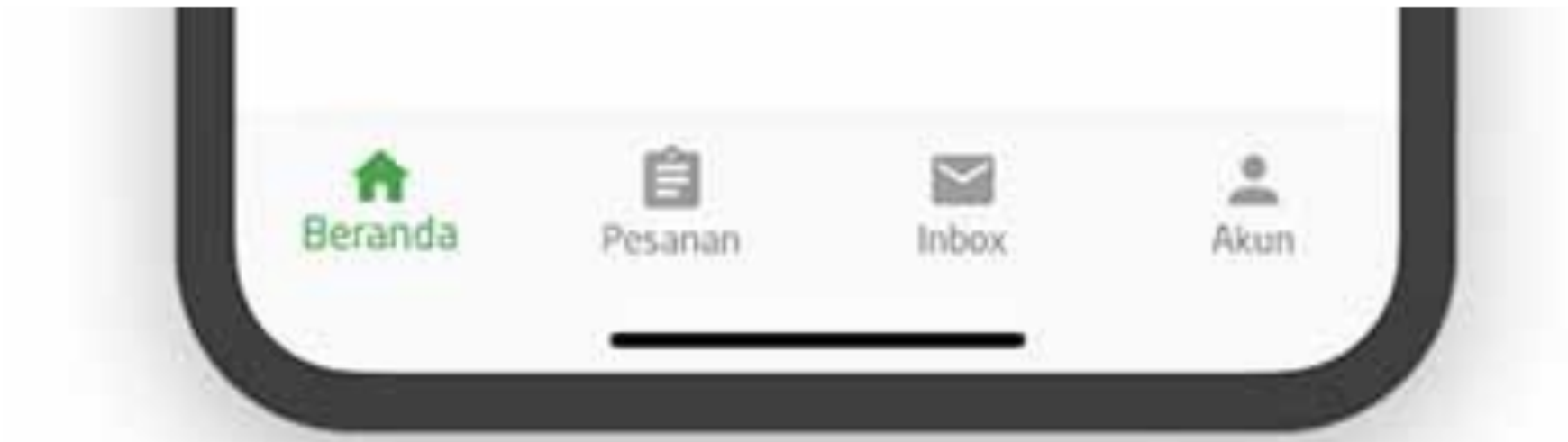
drawer **endDrawer**

Cara membuat navigation drawer pada flutter :

https://www.youtube.com/watch?v=Rp3RzZL5VaQ

Bottom Tab Navigator

Bottom Tab



Berikut cara pembuatannya

https://www.youtube.com/watch?v=1y-gfskpMIM

Rating - Feedback

Berikan Rating pada posting ini:

Berikan kritik dan saran..

Submit