Flutter Sesi 3

Navigasi Tab pada flutter

Tab adalah bagian dari antarmuka pengguna yang digunakan untuk menavigasi pengguna melalui route yang berbeda (misalnya, halaman) ketika diklik. Penggunaan tab dalam aplikasi adalah praktik standar yang sudah biasa dilakukan dibanyak aplikasi. Flutter menyediakan cara sederhana untuk membuat tata letak tab menggunakan library material. Dalam artikel ini, kita akan menjelajahi hal yang sama secara detail.

Untuk memahami konsep tab dan fungsinya dalam aplikasi Flutter dengan lebih baik, mari kita bangun sebuah aplikasi sederhana dengan 5 tab dengan mengikuti langkah-langkah di bawah ini:

- Merancang TabController.
- Menambahkan Tab pada Aplikasi.
- Menambahkan konten pada setiap Tab.

Merancang TabController

TabController mengontrol fungsi dari setiap tab dengan menyinkronkan tab dan konten satu sama lain, sesuai dengan namanya. Salah satu cara termudah untuk membuat tab di Flutter adalah dengan menggunakan widget DefaultTabController. Berikut adalah contoh implementasinya:

```
DefaultTabController(
//ada 5 tab jadi tambahkan angka 5 ke length
   length: 5,
```

```
  child:
);
```

Menambahkan Tab

Untuk membuat sebuah tab di Flutter, dapat menggunakan widget TabBar seperti contoh di bawah ini:

```
home: DefaultTabController(
       length: 5,
       child: Scaffold(
         appBar: AppBar(
           bottom: const TabBar(
             tabs: [
               Tab(icon: Icon(Icons.people_alt)),
               Tab(icon: Icon(Icons.music_video)),
               Tab(icon: Icon(Icons.camera_alt)),
               Tab(icon: Icon(Icons.grade)),
               Tab(icon: Icon(Icons.email)),
             ],
             ),
     ),),)
```

Menambahkan Konten pada Setiap Tab

Widget TabBarView dapat digunakan untuk menentukan konten yang akan ditampilkan di setiap tab. Untuk mempermudah, kita akan menampilkan ikon di dalam setiap tab sebagai pengganti teks, seperti yang ditunjukkan di bawah ini:

```
          body: const TabBarView(
            children: [
              Icon(Icons.music_note),
              Icon(Icons.music_video),
```

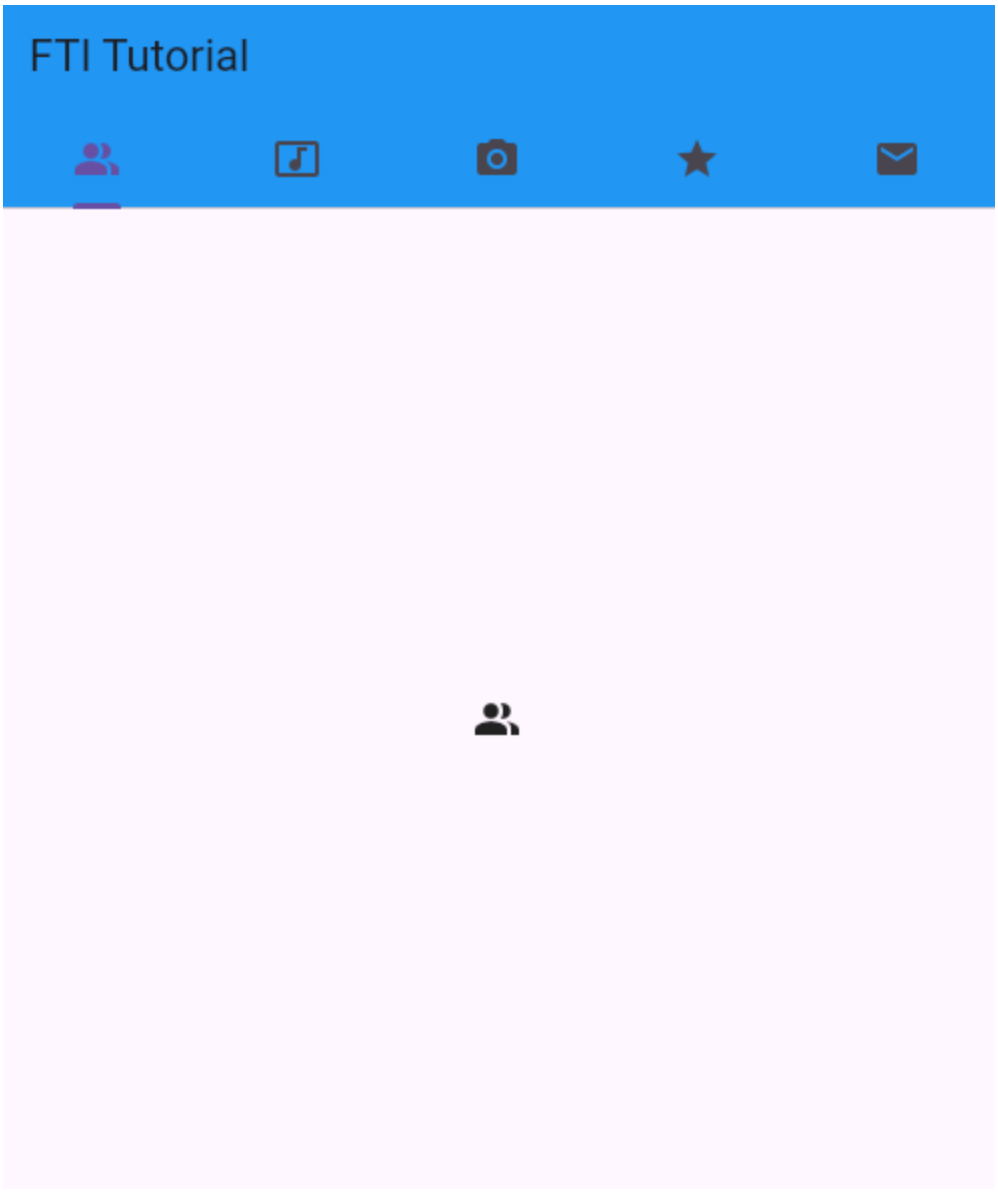
```
              Icon(Icons.camera_alt),
              Icon(Icons.grade),
              Icon(Icons.email),
            ],
          ),
```

Source code lengkapnya dapat dilihat dibawah ini:

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

// fungsi untuk mentrigger proses build aplikasi flutter
void main() {
  runApp(const TabBarDemo());
}

// class yang digunakan untuk build aplikasi
class TabBarDemo extends StatelessWidget {
  const TabBarDemo({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false, //Untuk menghilangkan banner
debug
      home: DefaultTabController(
        length: 5,
        child: Scaffold(
          appBar: AppBar(
            bottom: const TabBar(
              tabs: [
                Tab(icon: Icon(Icons.music_note)),
                Tab(icon: Icon(Icons.music_video)),
                Tab(icon: Icon(Icons.camera_alt)),
                Tab(icon: Icon(Icons.grade)),
                Tab(icon: Icon(Icons.email)),
              ],
            ),
            title: const Text('FTI Tutorial'),
            backgroundColor: Colors.blue,
```

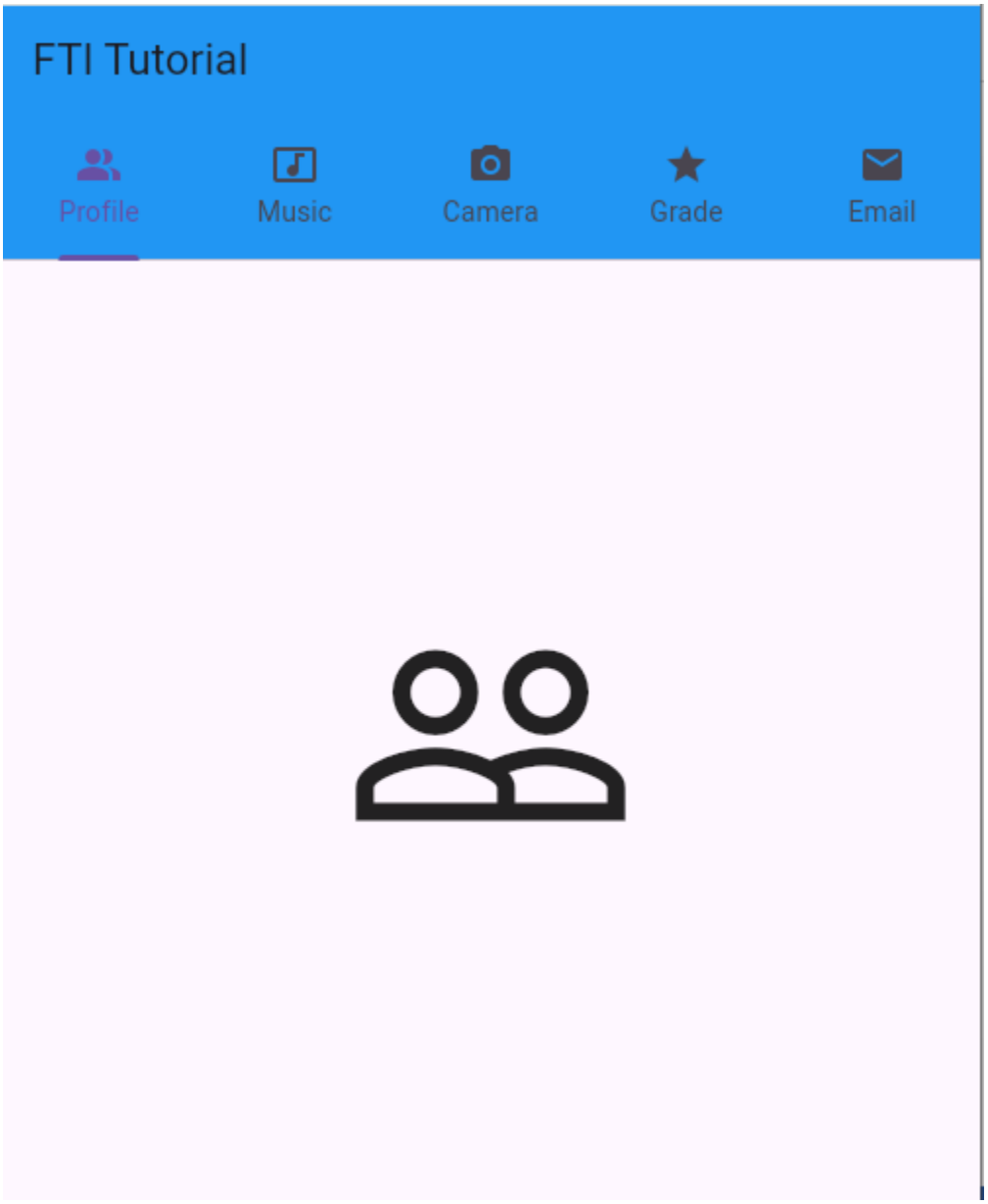
```
          ),
          body: const TabBarView(
            children: [
              Icon(Icons.music_note),
              Icon(Icons.music_video),
              Icon(Icons.camera_alt),
              Icon(Icons.grade),
              Icon(Icons.email),
            ],
          ),
        ),
      ),
    );
  }
}
```

Hasil :


FTI Tutorial

Latihan Kustom Tab

1. Buatlah project baru dan desain seperti tampilan 1 berikut:



2.

Membuat ListView pada flutter

Di Flutter, **ListView** adalah widget list yang dapat di-scroll dan diisi dengan widget-widget yang disusun secara linear. ListView menampilkan widget child-nya satu demi satu di arah scroll vertical atau horizontal.

Ada beberapa jenis ListView :

- ListView
- ListView.builder
- ListView.separated
- ListView.custom

Constructor ListView

```
ListView(
{Key key,
Axis scrollDirection: Axis.vertical,
bool reverse: false,
ScrollController controller,
bool primary,
ScrollPhysics physics,
bool shrinkWrap: false,
EdgeInsetsGeometry padding,
double itemExtent,
bool addAutomaticKeepAlives: true,
bool addRepaintBoundaries: true,
bool addSemanticIndexes: true,
double cacheExtent,
List children: const [],
int semanticChildCount,
DragStartBehavior dragStartBehavior: DragStartBehavior.start,
ScrollViewKeyboardDismissBehavior keyboardDismissBehavior:
ScrollViewKeyboardDismissBehavior.manual,
String restorationId,
```

```
Clip clipBehavior: Clip.hardEdge}
)
```

ListView.Buildder

```
ListView.builder(
{Key key,
Axis scrollDirection: Axis.vertical,
bool reverse: false,
ScrollController controller,
bool primary,
ScrollPhysics physics,
bool shrinkWrap: false,
EdgeInsetsGeometry padding,
double itemExtent,
@required IndexedWidgetBuilder itemBuilder,
int itemCount,
bool addAutomaticKeepAlives: true,
bool addRepaintBoundaries: true,
bool addSemanticIndexes: true,
double cacheExtent,
int semanticChildCount,
DragStartBehavior dragStartBehavior: DragStartBehavior.start,
ScrollViewKeyboardDismissBehavior keyboardDismissBehavior:
ScrollViewKeyboardDismissBehavior.manual,
String restorationId,
Clip clipBehavior: Clip.hardEdge}
)
```

ListView.custom

```
const ListView.custom(
{Key key,
Axis scrollDirection: Axis.vertical,
bool reverse: false,
ScrollController controller,
bool primary,
ScrollPhysics physics,
bool shrinkWrap: false,
```

```
EdgeInsetsGeometry padding,
double itemExtent,
@required SliverChildDelegate childrenDelegate,
double cacheExtent,
int semanticChildCount,
DragStartBehavior dragStartBehavior: DragStartBehavior.start,
ScrollViewKeyboardDismissBehavior keyboardDismissBehavior:
ScrollViewKeyboardDismissBehavior.manual,
String restorationId,
Clip clipBehavior: Clip.hardEdge}
)
```

ListView.separated

```
ListView.separated(
{Key key,
Axis scrollDirection: Axis.vertical,
bool reverse: false,
ScrollController controller,
bool primary,
ScrollPhysics physics,
bool shrinkWrap: false,
EdgeInsetsGeometry padding,
@required IndexedWidgetBuilder itemBuilder,
@required IndexedWidgetBuilder separatorBuilder,
@required int itemCount,
bool addAutomaticKeepAlives: true,
bool addRepaintBoundaries: true,
bool addSemanticIndexes: true,
double cacheExtent,
DragStartBehavior dragStartBehavior: DragStartBehavior.start,
ScrollViewKeyboardDismissBehavior keyboardDismissBehavior:
ScrollViewKeyboardDismissBehavior.manual,
String restorationId,
Clip clipBehavior: Clip.hardEdge}
)
```

Properties Dari ListView

- **childrenDelegate:** Properti ini menggunakan object SliverChildDelegate. Ini berfungsi sebagai delegasi yang memberikan children untuk ListView.
- **clipBehaviour:** Properti ini menggunakan object Clip enum (final) yang digunakan untuk mengontrol apakah konten dalam ListView akan dipotong atau tidak.
- **itemExtent:** itemExtent menggunakan nilai double sebagai object untuk mengontrol area yang dapat di scroll dalam ListView.
- **padding:** Properti ini menggunakan EdgeInsetsGeometry sebagai object untuk memberikan ruang antara Listview dan anak-anaknya.
- **scrollDirection:** Properti ini menggunakan enum Axis sebagai object untuk memutuskan arah gulir pada ListView..
- **shrinkWrap:** Properti ini menggunakan nilai boolean sebagai object untuk memutuskan apakah ukuran area yang dapat di scroll akan ditentukan oleh konten di dalam ListView.

ListView

Ini adalah constructor standar untuk class ListView. ListView secara sederhana menggunakan list widget dan membuatnya dapat di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh karena itu, menggunakan terlalu banyak widget dapat mengurangi efisiensi list.

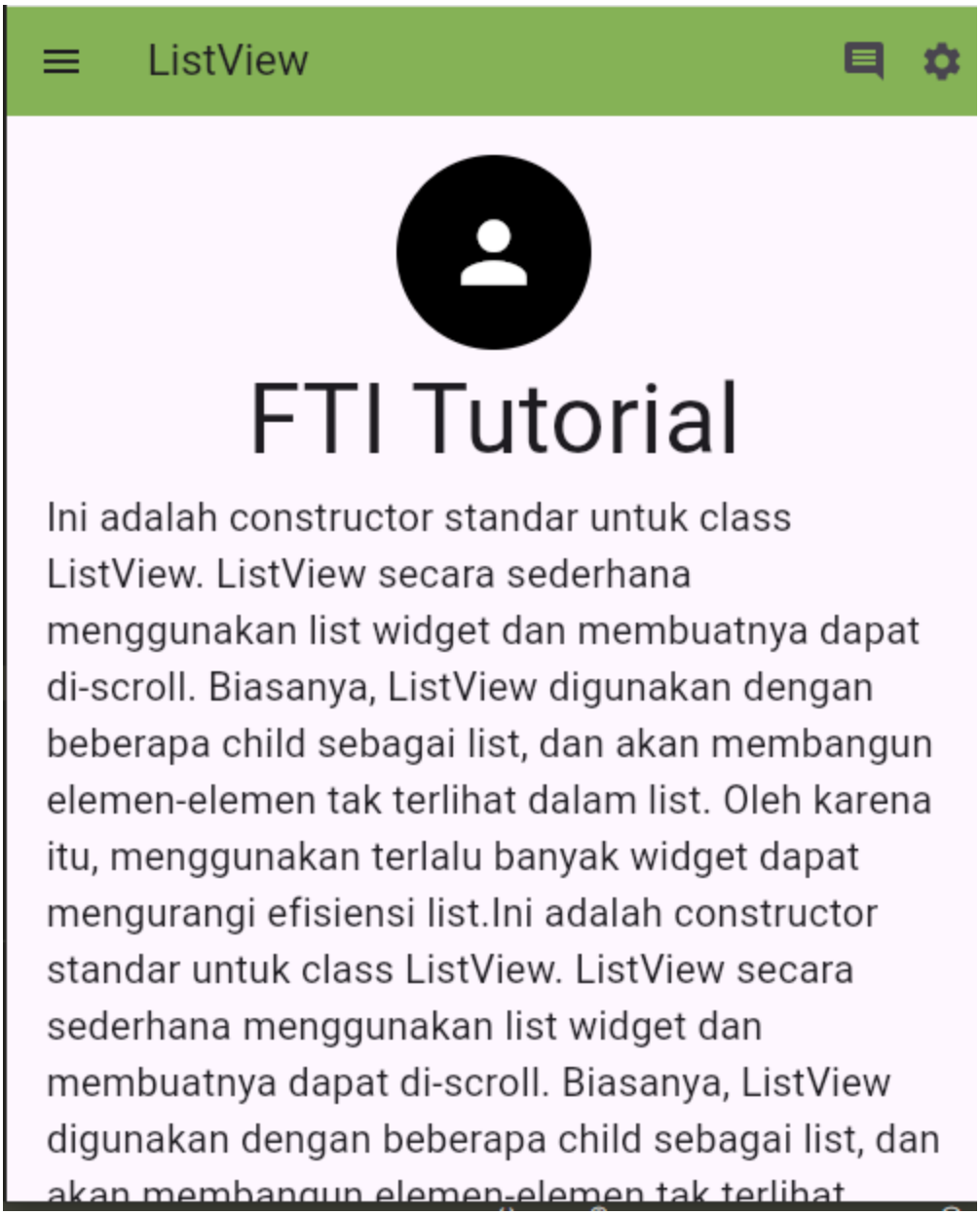
Latihan ListView Constructor :

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

// fungsi untuk mentrigger proses build aplikasi flutter
void main() {
  runApp(MyApp()); //MaterialApp
}

MaterialApp MyApp() {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text("ListView"),
        actions: [
          IconButton(
            icon: const Icon(Icons.comment),
            tooltip: 'Comment Icon',
            onPressed: () {},
          ), //IconButton
          IconButton(
            icon: const Icon(Icons.settings),
            tooltip: 'Setting Icon',
            onPressed: () {},
          ), //IconButton
        ], //[]
        backgroundColor: Colors.lightGreen,
        elevation: 50.0,
        leading: IconButton(
          icon: const Icon(Icons.menu),
          tooltip: 'Menu Icon',
          onPressed: () {},
        ),
        systemOverlayStyle: SystemUiOverlayStyle.light,
      ), //AppBar
      body: ListView(
        padding: const EdgeInsets.all(20),
        children: const [
```

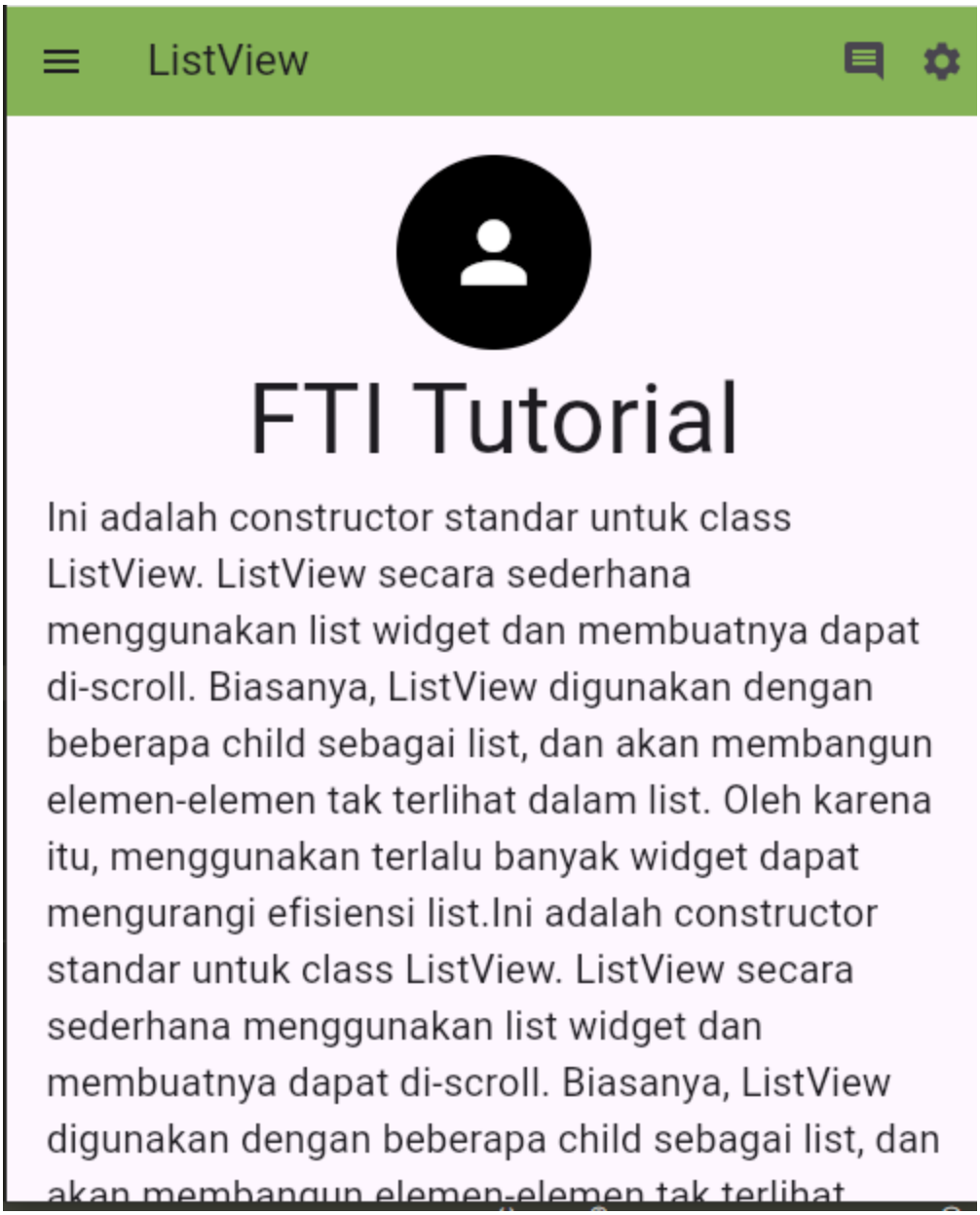
```
          CircleAvatar(
            maxRadius: 50,
            backgroundColor: Colors.black,
            child: Icon(Icons.person, color: Colors.white, size: 50),
          ),
          Center(
            child: Text(
              'FTI Tutorial',
              style: TextStyle(
                fontSize: 50,
              ),
            ),
          ),
          Text(
            "Ini adalah constructor standar untuk class ListView.
ListView secara sederhana menggunakan list widget dan membuatnya dapat
di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai
list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh
karena itu, menggunakan terlalu banyak widget dapat mengurangi
efisiensi list.Ini adalah constructor standar untuk class ListView.
ListView secara sederhana menggunakan list widget dan membuatnya dapat
di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai
list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh
karena itu, menggunakan terlalu banyak widget dapat mengurangi
efisiensi list.Ini adalah constructor standar untuk class ListView.
ListView secara sederhana menggunakan list widget dan membuatnya dapat
di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai
list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh
karena itu, menggunakan terlalu banyak widget dapat mengurangi
efisiensi list.Ini adalah constructor standar untuk class ListView.
ListView secara sederhana menggunakan list widget dan membuatnya dapat
di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai
list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh
karena itu, menggunakan terlalu banyak widget dapat mengurangi
efisiensi list.",
            style: TextStyle(
              fontSize: 20,
            ),
          ),
```

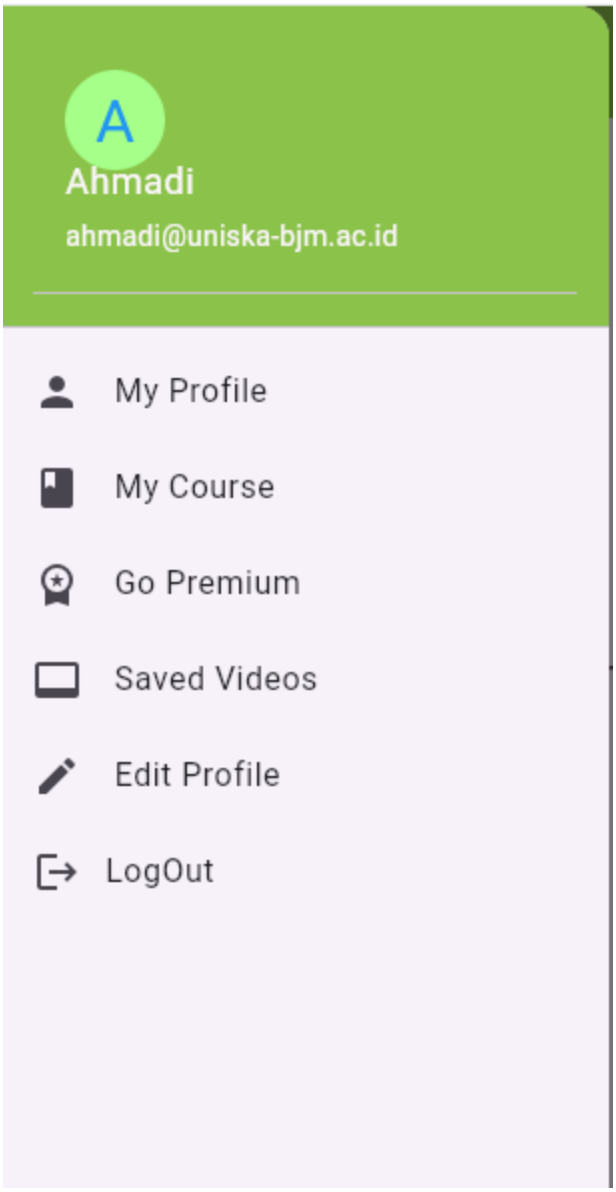
```
        ],
      ),
      //Center
    ), //Scaffold
    debugShowCheckedModeBanner: false, //Digunakan untuk menghapus
Debug Banner
  );
}
```

Hasil ListView Constructor:



Latihan dan Tugas list + Drawer


ListView
FTI Tutorial
Ini adalah constructor standar untuk class ListView. ListView secara sederhana menggunakan list widget dan membuatnya dapat di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai list, dan akan membangun elemen-elemen tak terlihat dalam list. Oleh karena itu, menggunakan terlalu banyak widget dapat mengurangi efisiensi list.Ini adalah constructor standar untuk class ListView. ListView secara sederhana menggunakan list widget dan membuatnya dapat di-scroll. Biasanya, ListView digunakan dengan beberapa child sebagai list, dan akan membangun elemen-elemen tak terlihat



ListView.builder

ListView.builder adalah constructor yang digunakan untuk membangun list widget yang diulang. Constructor ini memiliki dua parameter utama:

1. itemCount: Parameter ini menentukan jumlah pengulangan widget yang akan dibangun dan bersifat opsional. Jika tidak ditentukan, maka widget akan dibangun secara tak terbatas.

2. itemBuilder: Parameter ini harus diisi dan digunakan untuk membangun widget yang akan dihasilkan sebanyak nilai itemCount.

Dengan demikian, Anda dapat menggunakan constructor ListView.builder untuk membuat daftar widget yang diulangi dengan mudah dan efektif.

Latihan ListView Builder :

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

// fungsi untuk mentrigger proses build aplikasi flutter
void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text("ListView.builder"),
          actions: [
            IconButton(
              icon: const Icon(Icons.comment),
              tooltip: 'Comment Icon',
              onPressed: () {},
            ), //IconButton
            IconButton(
              icon: const Icon(Icons.settings),
              tooltip: 'Setting Icon',
              onPressed: () {},
            ), //IconButton
          ], //[]
```

```
          backgroundColor: Colors.lightGreen,
          elevation: 50.0,
          leading: IconButton(
            icon: const Icon(Icons.menu),
            tooltip: 'Menu Icon',
            onPressed: () {},
          ),
          systemOverlayStyle: SystemUiOverlayStyle.light,
        ), //AppBar
        body: ListView.builder(
          itemCount: 31,
          itemBuilder: (context, position) {
            return Card(
              child: Padding(
                padding: const EdgeInsets.all(20.0),
                child: Text(
                  position.toString(),
                  style: const TextStyle(fontSize: 22.0),
                ),
              ),
            );
          },
        ),

        //Center
      ), //Scaffold
      debugShowCheckedModeBanner:
          false, //Digunakan untuk menghapus Debug Banner
    );
  }
}
```

Hasil ListView Builder:

ListView.separated

Constructor ListView.separated digunakan untuk membuat list widget yang dapat terdiri dari dua bagian yaitu list utama dan list pemisah. List pemisah digunakan untuk memisahkan widget pada list utama. Berbeda dengan

constructor ListView.builder, constructor ListView.separated harus memiliki parameter itemCount.

Latihan ListView.separated :

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

// fungsi untuk mentrigger proses build aplikasi flutter
void main() {
  runApp(const MyApp()); //MaterialApp
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text("ListView.builder"),
          actions: [
            IconButton(
              icon: const Icon(Icons.comment),
              tooltip: 'Comment Icon',
              onPressed: () {},
            ),
            IconButton(
              icon: const Icon(Icons.settings),
              tooltip: 'Setting Icon',
              onPressed: () {},
            ),
          ],
          backgroundColor: Colors.lightGreen,
          elevation: 50.0,
          leading: IconButton(
```

```
            icon: const Icon(Icons.menu),
            tooltip: 'Menu Icon',
            onPressed: () {},
          ),
          systemOverlayStyle: SystemUiOverlayStyle.light,
        ),
        body: ListView.separated(
          itemBuilder: (context, position) {
            return Card(
              child: Padding(
                padding: const EdgeInsets.all(15.0),
                child: Text(
                  'List Item $position',
                ),
              ),
            );
          },
          separatorBuilder: (context, position) {
            return Card(
              color: Colors.grey,
              child: Padding(
                padding: const EdgeInsets.all(5.0),
                child: Text(
                  'Separator $position',
                  style: const TextStyle(color: Colors.white),
                ),
              ),
            );
          },
          itemCount: 31,
        ),
      ),
      debugShowCheckedModeBanner:
          false, //Digunakan untuk menghapus Debug Banner
    );
  }
}
```

Hasil ListView.separated:

ListView.Custom

Untuk membangun ListView dengan fungsi khusus, kita dapat menggunakan constructor ListView.custom. Constructor ini memiliki parameter utama berupa SliverChildDelegate yang bertanggung jawab untuk membangun item dalam list.

Ada dua jenis SliverChildDelegate yang dapat digunakan:

- **SliverChildListDelegate**, yang menggunakan list widget children
- **SliverChildBuilderDelegate**, yang menggunakan IndexedWidgetBuilder, yaitu fungsi builder.

Secara lebih detail, kita dapat menyimpulkan bahwa ListView.builder dibuat menggunakan ListView.custom dengan SliverChildBuilderDelegate. Selain itu, constructor default ListView sebenarnya menggunakan ListView.custom dengan SliverChildListDelegate.

Membuat List di flutter

Daftar adalah elemen paling populer dari setiap web atau aplikasi seluler. Mereka terdiri dari beberapa baris item, yang meliputi teks, tombol, sakelar, ikon, thumbnail, dan banyak lagi. Kita dapat menggunakannya untuk menampilkan berbagai informasi seperti menu, tab, atau untuk memecahkan monoton file teks murni.

Di bagian ini, kita akan belajar bagaimana kita dapat bekerja dengan Daftar di Flutter. Fluttermemungkinkan Anda untuk bekerja dengan Daftar dengan cara yang berbeda, yang diberikan di bawah ini:

- Basic Lists
- Long Lists

- Grid Lists
- Horizontal Lists

Flutter termasuk sebuah widget **ListView** untuk bekerja dengan List, yang merupakan konsep dasar menampilkan data dalam aplikasi seluler. ListView adalah standar sempurna untuk menampilkan daftar yang hanya berisi beberapa item. ListView juga termasuk **Daftar Judul** widget, yang memberikan lebih banyak properti untuk struktur visual ke daftar data.

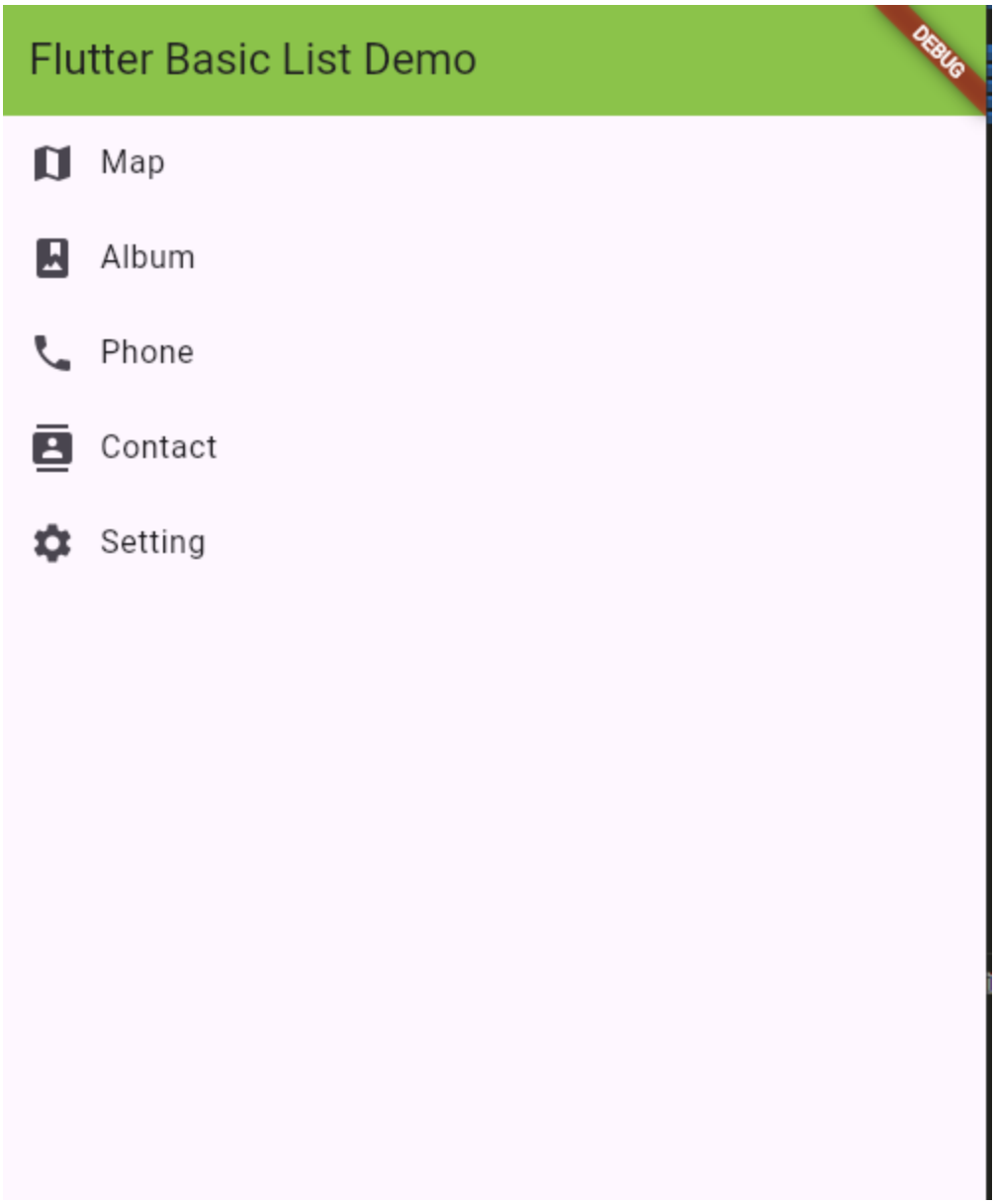
Contoh berikut menampilkan daftar dasar dalam aplikasi Flutter.

Latihan :

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final appTitle = 'Flutter Basic List Demo';

    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.gr,
          title: Text(appTitle),
        ),
        body: ListView(
          children: <Widget>[
            ListTile(
              leading: Icon(Icons.map),
              title: Text('Map'),
```

```
            ),
            ListTile(
              leading: Icon(Icons.photo_album),
              title: Text('Album'),
            ),
            ListTile(
              leading: Icon(Icons.phone),
              title: Text('Phone'),
            ),
            ListTile(
              leading: Icon(Icons.contacts),
              title: Text('Contact'),
            ),
            ListTile(
              leading: Icon(Icons.settings),
              title: Text('Setting'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Hasil :



Membuat daftar dengan list panjang

Terkadang Anda ingin menampilkan daftar yang sangat panjang dalam satu layar aplikasi Anda, maka, dalam hal ini, metode di atas untuk menampilkan daftar tidak sempurna. Untuk bekerja dengan daftar yang berisi sejumlah besar item, kita perlu menggunakan sebuah **ListView.builder()** konstruktor.

Perbedaan utama antara ListView dan ListView.builder adalah bahwa ListView membuat semua item sekaligus, sedangkan konstruktor ListView.builder () membuat item ketika mereka digulir ke layar.

Mari kita lihat contoh berikut. Buka **main.dart** file dan ganti kode berikut.