Flutter Sesi 4

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.

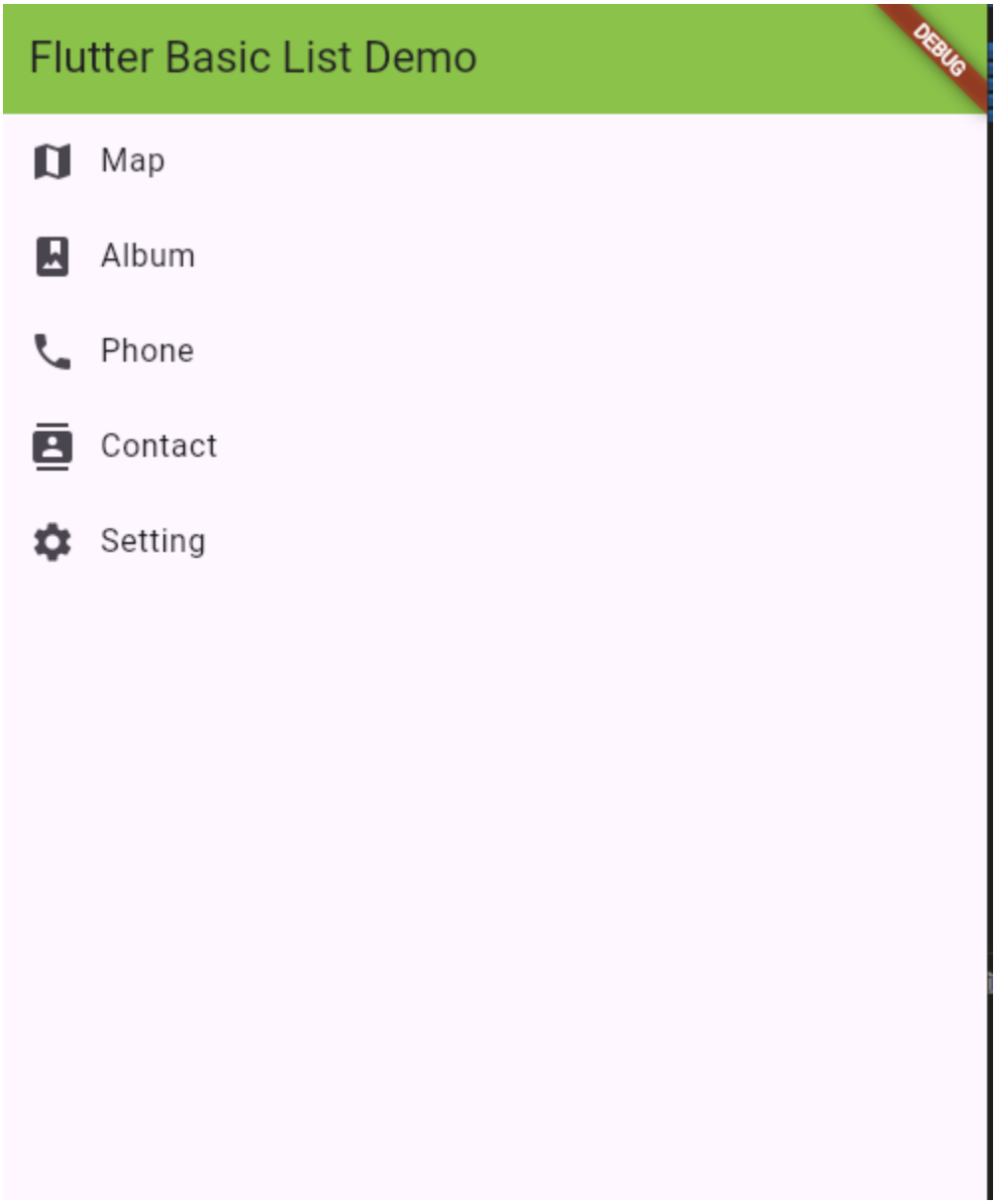
Contoh berikut menampilkan daftar dasar dalam aplikasi Flutter.

Latihan :

```dart
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 :


Flutter Basic List Demo
DEBUG
Map
Album
Phone
Contact
Setting

Orientasi UI di Flutter

Semua aplikasi harus dapat menyesuaikan antarmuka pengguna (UI) mereka berdasarkan orientasi ponsel. Terdapat dua mode orientasi yaitu portrait dan landscape pada smartphone. Dalam Flutter, hal ini dilakukan dengan menggunakan **OrientationBuilder**, yang menentukan orientasi saat ini dari aplikasi. Kita akan melihatnya dengan membangun sebuah aplikasi sederhana dan mengubah orientasinya serta menampilkan perubahan UI.

Berikut adalah langkah-langkah untuk membangun aplikasi yang mengubah UI berdasarkan orientasi ponsel:

- Buatlah sebuah GridView dengan 3 kolom.
- Gunakan OrientationBuilder untuk mengubah jumlah kolom.

Mari kita bahas kedua langkah tersebut dengan lebih detail.

Membuat GridView

Untuk membuat sebuah GridView dengan 3 kolom, gunakan kode seperti yang ditunjukkan di bawah ini:

```
GridView.count(
  crossAxisCount: 3,
);
```

Menggunakan OrientationBuilder

Seperti yang telah dibahas sebelumnya, orientationBuilder menentukan orientasi aplikasi saat ini. Kita akan mendesain orientationBuilder sedemikian rupa sehingga akan menampilkan 3 kolom dalam mode portrait

dan 4 kolom dalam mode landscape. Untuk melakukannya, ikuti kode di bawah ini:

```
OrientationBuilder(
  builder: (context, orientation) {
    return GridView.count(
      crossAxisCount: orientation == Orientation.portrait ? 3 : 4,
    );
  },
);
```

Source code lengkap dapat dilihat di bawah ini:

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

void main() {
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    const appTitle = 'FTI Tutorial';

    return const MaterialApp(
      title: appTitle,
      home: OrientationList(
        title: appTitle,
      ),
    );
  }
}

class OrientationList extends StatelessWidget {
  final String title;
```

```
  const OrientationList({Key? key, required this.title}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(title), backgroundColor:
Colors.blue),
      body: OrientationBuilder(
        builder: (context, orientation) {
          return GridView.count(
            //Grid dengan 3 kolom untuk mode portrait dan 4 kolom
untuk mode landscape
          crossAxisCount: orientation == Orientation.portrait ? 3 : 4,
            // random item generator
            children: List.generate(100, (index) {
              return Center(
                child: Text(
                  'A $index',
                  style: Theme.of(context).textTheme.headlineLarge,
                ),
              );
            }),
          );
        },
      ),
    );
  }
}
```

Hasil :

Gestures di Flutter

Gestures atau gerakan tangan digunakan untuk berinteraksi dengan aplikasi. Biasanya digunakan pada perangkat yang menggunakan layar sentuh untuk berinteraksi secara fisik dengan aplikasi. Gerakan tersebut dapat berupa sekali ketukan pada layar hingga interaksi fisik yang lebih kompleks seperti

menggeser ke arah tertentu hingga menggulirkan aplikasi. Gerakan tersebut sangat digunakan dalam permainan dan hampir setiap aplikasi membutuhkannya untuk berfungsi karena perangkat semakin banyak yang menggunakan layar sentuh. Dalam artikel ini, kami akan membahasnya secara rinci.

Beberapa gerakan yang sering digunakan antara lain:

- ✓ **Tap:** Menyentuh permukaan perangkat dengan ujung jari selama beberapa saat dan akhirnya melepaskan ujung jari.
- ✓ **Double Tap:** Mengetuk dua kali dalam waktu singkat.
- ✓ **Drag:** Menyentuh permukaan perangkat dengan ujung jari dan kemudian menggerakkan ujung jari secara perlahan dan akhirnya melepaskan ujung jari.
- ✓ **Flick:** Sama seperti drag, namun dilakukan dengan kecepatan yang lebih tinggi.
- ✓ **Pinch:** Memiringkan permukaan perangkat menggunakan dua jari atau melakukan gerakan mencubit pada layar perangkat.
- ✓ **Zoom:** Kebalikan dari pinch.
- ✓ **Panning:** Menyentuh permukaan perangkat dengan ujung jari dan menggerakkannya ke arah yang diinginkan tanpa melepaskan ujung jari.

Widget GestureDetector pada Flutter digunakan untuk mendeteksi interaksi fisik dengan aplikasi pada UI. Jika sebuah widget harus mengalami gerakan,

maka widget tersebut ditempatkan di dalam widget GestureDetector. Widget yang sama menangkap gerakan tersebut dan mengembalikan tindakan atau respons yang tepat.

Berikut adalah daftar gerakan dan event yang sesuai dengan gerakan tersebut:

Tap

- ✓ onTapDown
- ✓ onTapUp
- ✓ onTap
- ✓ onTapCancel

Double tap

- ✓ onDoubleTap

Long press

- ✓ onLongPress

Vertical drag

- ✓ onVerticalDragStart
- ✓ onVerticalDragUpdate
- ✓ onVerticalDragEnd

Horizontal drag

- ✓ onHorizontalDragStart
- ✓ onHorizontalDragUpdate
- ✓ onHorizontalDragEnd

Pan

- ✓ onPanStart
- ✓ onPanUpdate
- ✓ onPanEnd

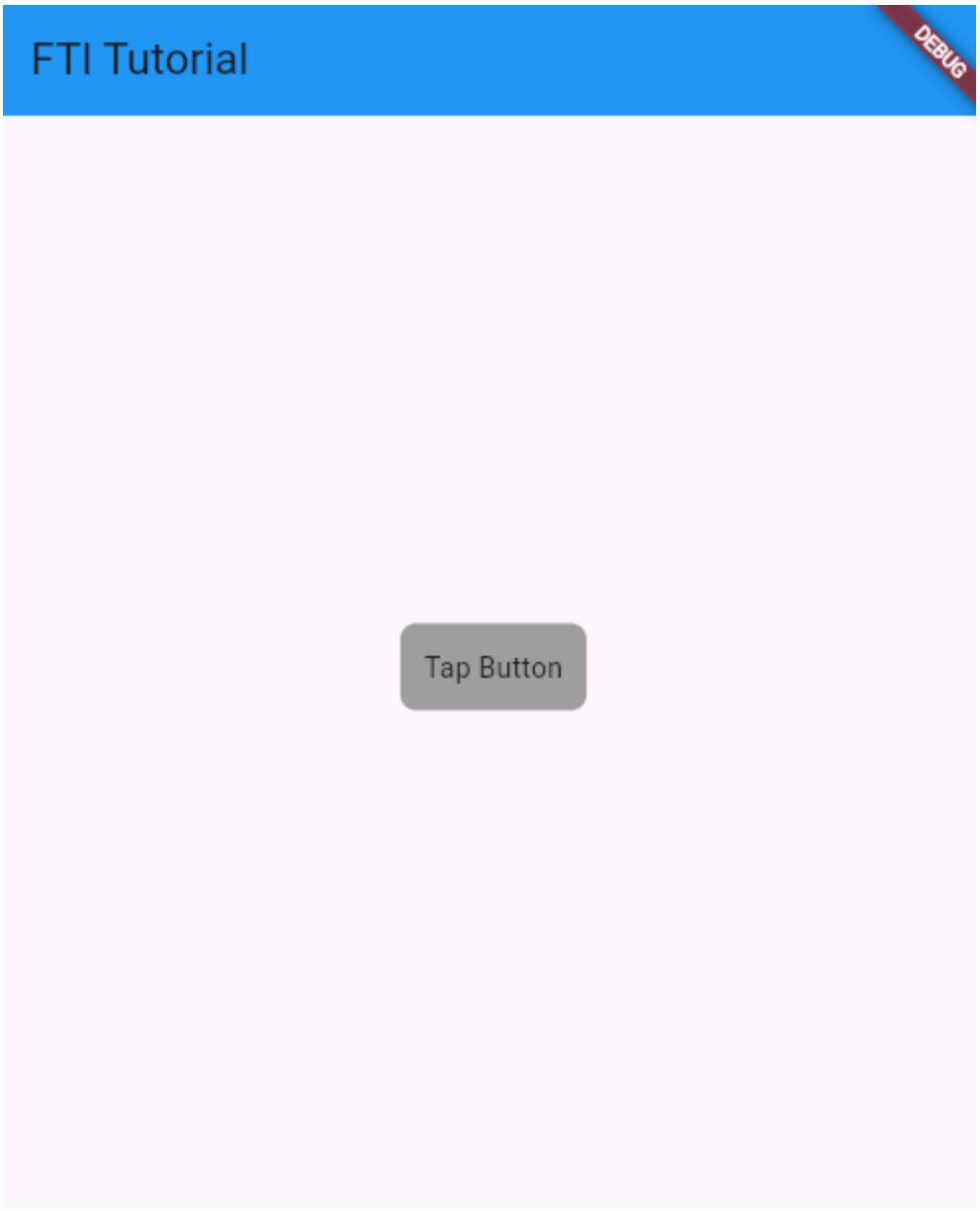
Contoh Latihan:

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    const title = 'Gestures';

    return const MaterialApp(
      title: title,
      home: MyHomePage(title: title),
    );
  }
}

class MyHomePage extends StatelessWidget {
  final String title;

  const MyHomePage({Key? key, required this.title}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FTI Tutorial'),
        backgroundColor: Colors.blue,
      ),
```

```
      body: const Center(child: MyButton()),
    );
  }
}

class MyButton extends StatelessWidget {
  const MyButton({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    showSnackbar(Color color, String message) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          backgroundColor: color,
          content: Text(message),
        ),
      );
    }

    return GestureDetector(
      // untuk menampilkan snackbar ketika melakukan gesture
      onTap: () {
        showSnackbar(Colors.black, 'Kamu telah melakukan Tapped pada
Button');
      },
      onDoubleTap: () {
        showSnackbar(
            Colors.purple, 'Kamu telah melakukan Double Tapped pada
Button');
      },
      onLongPress: () {
        showSnackbar(
            Colors.green, 'Kamu telah melakukan Long Press pada
Button');
      },
      // Button Tap
      child: Container(
        padding: const EdgeInsets.all(12.0),
        decoration: BoxDecoration(
```

```
          color: Colors.grey,
          borderRadius: BorderRadius.circular(8.0),
        ),
        child: const Text('Tap Button'),
      ),
    );
  }
}
```

Hasil :


FTI Tutorial
DEBUG
Tap Button