

Modul 13

Komponen dalam mendesain Aplikasi

Slider, ToolTips, dan Progress Indicator

Module Overview

Mengenal dan mengimplementasikan widget Slider, ToolTips dan Progress Indicator pada aplikasi flutter. Widget tooltips dan progress indicator berguna untuk memberikan informasi berupa tips atau indikator waktu respon kepada pengguna, sedangkan slider dapat menerima interaksi dari pengguna untuk mengubah value.

Module Objectives

Setelah mempelajari dan mempraktikkan modul ini, mahasiswa diharapkan dapat:

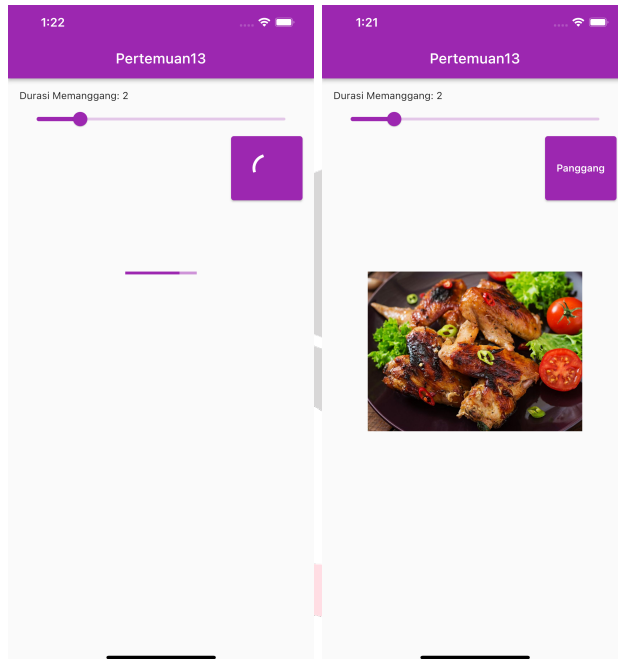
- Mengetahui kegunaan dari widget Slider, ToolTips dan Progress.
- Mampu menerapkan widget Slider, ToolTips dan Progress pada aplikasi flutter.

Baners, Dialog dan Snackbars

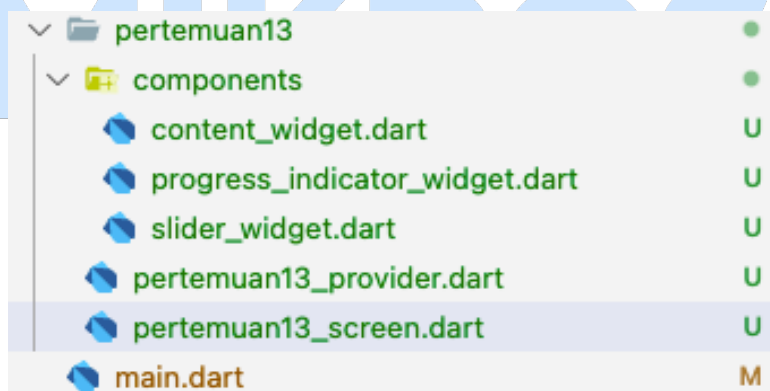
Menerapkan Card dan ListTile

Keterangan:

Aplikasi ini akan menampilkan data json menggunakan card dan listtile pada sebuah list item..



1. Berikut adalah struktur folder untuk pertemuan 12.



Keterangan:


- pertemuan13_screen.dart merupakan file untuk screen UI flutter (root home)

- pertemuan13_provider.dart digunakan provide logic pada aplikasi.
 - file di dalam folder **components**, yaitu:
 - content_widget.dart, adalah section yang berisi gambar.
 - progress_indicator_widget.dart, adalah section yang berisi progress indicator pada tombol "Panggang".
 - slider_widget.dart, adalah section yang berisi slider untuk mengatur durasi.
2. Buatlah screen baru dengan nama file **pertemuan13_screen.dart** pada folder pertemuan13. Import **material**, Gunakan **StatefulWidget**, namai class **statefulWidget** dengan **Pertemuan13Screen** dan gunakan **Scaffold** untuk memulai mendesain komponen.

```
1 import 'package:flutter/material.dart';
2
3 class Pertemuan13Screen extends StatefulWidget {
4   Pertemuan13Screen({Key? key}) : super(key: key);
5
6   @override
7   State<Pertemuan13Screen> createState() => _Pertemuan13ScreenState();
8 }
9
10 class _Pertemuan13ScreenState extends State<Pertemuan13Screen> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       appBar: AppBar(title: const Text('Pertemuan13')),
15       body: Text('Pertemuan13'),
16     );
17   }
18 }
19
```

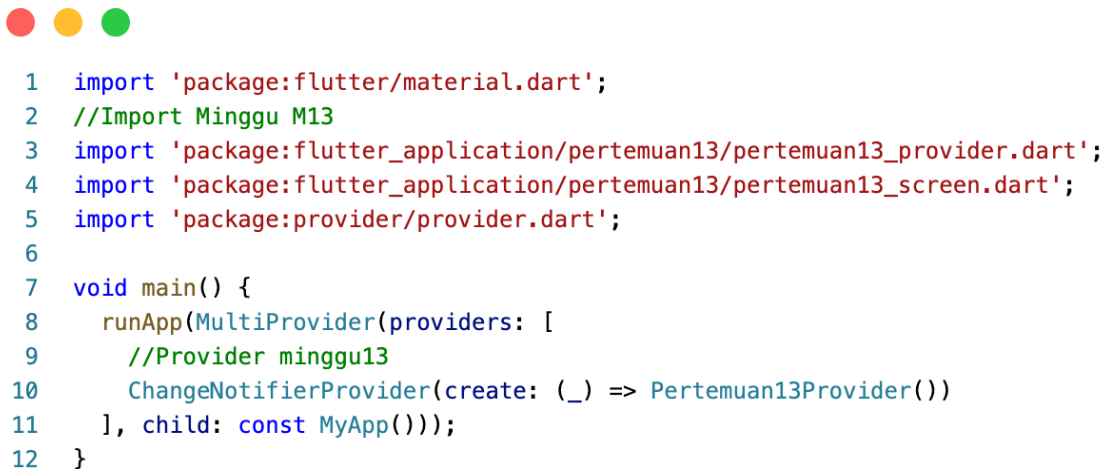
Keterangan:

- Update **main.dart** menjadi "home: Pertemuan13Screen()"
 - Pada terminal root folder ketik "flutter run" untuk menjalankan aplikasi.
3. Buatlah sebuah file dengan nama `pertemuan13_provider.dart` untuk menyediakan logic proses nantinya. Namai dengan class **Pertemuan13Provider** seperti gambar dibawah.



```
1 import 'package:flutter/material.dart';
2
3 class Pertemuan13Provider extends ChangeNotifier {
4
5 }
```

Kemudian, tambahkan pada `MultiProvider` di **main.dart** seperti gambar berikut:



```
1 import 'package:flutter/material.dart';
2 //Import Minggu M13
3 import 'package:flutter_application/pertemuan13/pertemuan13_provider.dart';
4 import 'package:flutter_application/pertemuan13/pertemuan13_screen.dart';
5 import 'package:provider/provider.dart';
6
7 void main() {
8   runApp(MultiProvider(providers: [
9     //Provider minggu13
10    ChangeNotifierProvider(create: (_) => Pertemuan13Provider())
11  ], child: const MyApp()));
12 }
```

****Pastikan import Pertemuan13Provider telah dilakukan.**

****Pastikan penulisan nama class sesuai.**

4. **Menggunakan Slider.** Pada minggu ini, setiap widget akan pisahkan menjadi beberapa component, sehingga **buatlah slider** pada file **slider_widget.dart**

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_application/pertemuan13/pertemuan13_provider.dart';
3 import 'package:provider/provider.dart';
4
5 class SlideWidget extends StatelessWidget {
6   const SlideWidget({Key? key}) : super(key: key);
7
8   @override
9   Widget build(BuildContext context) {
10     final prov = Provider.of<Pertemuan13Provider>(context);
11     return Slider(
12       value: prov.sliderValue,
13       min: 0,
14       max: 10,
15       label: prov.sliderValue.round().toString(),
16       onChanged: (value) {
17         prov.setSliderValue = value;
18       },
19     );
20   }
21 }
22
```

Keterangan:

- **Baris 10**, sebuah variable untuk menunjukkan provider yang digunakan.
- **Baris 12**, slider memerlukan value yang akan terus berubah-ubah ketika user mengubah nilai slidernya.
- **Baris 17**, adalah sebuah setter untuk mengganti "value" slider. **setSliderValue** dan **sliderValue** akan disediakan pada provider.

Atur **setSliderValue** dan **sliderValue** pada **Pertemuan13Provider**. Tambahkan variable, setter dan getter berikut di dalam class **Pertemuan13Provider**:



```

1 double _sliderValue = 0;
2 double get sliderValue => _sliderValue;
3 set setSliderValue(val) {
4   _sliderValue = val;
5   notifyListeners();
6 }

```

Keterangan:

- **Baris 1**, sebuah lokal variable untuk slidervalue.
- **Baris 2**, sebuah getter untuk mengirim nilai _sliderValue.
- Baris 3 - 5, sebuah setter untuk menerima dan mengubah nilai _sliderValue ketika user mengubah value dari slider. Ketika ada perubahan maka ChangeNotifier akan memberitahu ada perubahan state melalui fungsi **notifyListeners()**.

Menampilkan SlideWidget pada body Scaffold pertemuan13screen.dart.



```

1 class _Pertemuan13ScreenState extends State<Pertemuan13Screen> {
2   @override
3   Widget build(BuildContext context) {
4     final prov = Provider.of<Pertemuan13Provider>(context);
5     return Scaffold(
6       appBar: AppBar(title: const Text('Pertemuan13')),
7       body: Padding(
8         padding: const EdgeInsets.all(16.0),
9         child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
10          Text('Durasi Memanggang: ${prov.sliderValue.round().toString()}'),
11          const SlideWidget(),
12        ]),
13     ),
14   );
15 }
16 }

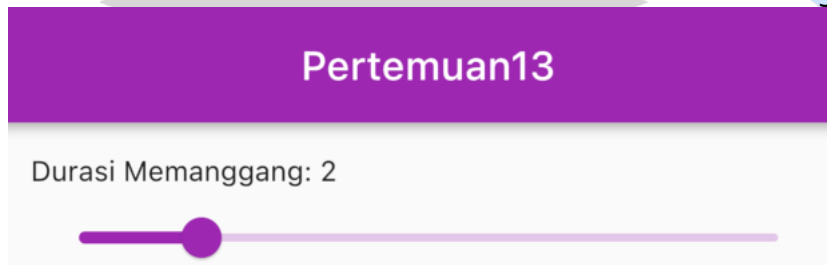
```

Keterangan:

- Pada body tambahkan Padding dan Column. Pada Children Column, tambahkan Text dan Widget **SliderWidget()** yang telah dibuat pada tahap sebelumnya.
- **Baris 10**, akan menampilkan sliderValue yang dibulatkan menggunakan fungsi **round()**.
- **Baris 11**, untuk memanggil **SliderWidget()**.

Hasil:

Reload flutter untuk melihat hasil. Geser slider untuk mengubah nilai slider.



5. **Menggunakan CircularProgressIndicator indeterminate.** Pada tahapan ini, circular indicator akan ditampilkan button selama durasi yang telah diatur pada slider. Tambahkan kode berikut pada file **progress_indicator_widget.dart**.

Keterangan:

- Menggunakan ElevatedButton, pada fungsi onPressed() akan memicu fungsi **mulaiMemanggang()** dengan mengirim nilai lama durasi memanggang.
- **Baris 16 - 20**, adalah sebuah conditional statements untuk mengecek kondisi. Jika sedangMemanggang maka tampilkan circular indicator, selain itu tampilkan text.



```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_application/pertemuan13/pertemuan13_provider.dart';
3 import 'package:provider/provider.dart';
4
5 class ProgressIndicatorWidget extends StatelessWidget {
6   const ProgressIndicatorWidget({Key? key}) : super(key: key);
7
8   @override
9   Widget build(BuildContext context) {
10     final prov = Provider.of<Pertemuan13Provider>(context);
11     return ElevatedButton(
12       style: ElevatedButton.styleFrom(minimumSize: Size(100, 90)),
13       onPressed: () {
14         prov.mulaiMemanggang(prov.sliderValue.round());
15       },
16       child: prov.sedangMemanggang == true
17         ? const CircularProgressIndicator(
18             color: Colors.white,
19           )
20         : const Text('Panggang'));
21   }
22 }
23
```



UNIVERSITAS
MIKROSKIL

Pada **pertemuan13_provider.dart**, tambahkan **mulaiMemanggang**, **sedangMemanggang** dan **selesaiMasak** di dalam class **Pertemuan13Provider** seperti tampilan berikut:



```
1  bool _sedangMemanggang = false;
2  bool get sedangMemanggang => _sedangMemanggang;
3  set setSedangMemanggang(val) {
4    _sedangMemanggang = val;
5    notifyListeners();
6  }
7
8  bool _selesaiMasak = false;
9  bool get selesaiMasak => _selesaiMasak;
10 set setSelesaiMasak(val) {
11   _selesaiMasak = val;
12   notifyListeners();
13 }
14
15 mulaiMemanggang(int val) async {
16   setSedangMemanggang = true;
17   Future.delayed(Duration(seconds: val), () {
18     setSedangMemanggang = false;
19     setSelesaiMasak = true;
20   });
21 }
```

Keterangan:

- ketika user klik tombol "Panggang", maka akan memicu fungsi "mulaiMemanggang" sesuai durasi yang diset dari Slider. Mula-Mula, **setSedangMemanggang = true** agar di muncul progress indicator.
- Future.delayed adalah sebuah fungsi delayed sesuai durasi. ***Pada dunia mobile programming, fungsi ini bisa berupa request ke server atau***

logika kompleks yang memerlukan waktu. Ketika telah selesai delay maka `setSedangMemasak = false` dan `setMasakSelesai = true`. Ini berfungsi untuk memicu tampilan / state ketika durasi selesai.

Tampilkan `ProgressIndicatorWidget` pada pertemuan13_screen seperti tampilan berikut:



```

1  body: Padding(
2    padding: const EdgeInsets.all(16.0),
3    child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
4      Text('Durasi Memanggang: ${prov.sliderValue.round().toString()}'),
5      const SlideWidget(),
6      const Align(
7        child: ProgressIndicatorWidget(),
8        alignment: Alignment.bottomRight,
9      ),
10   ]),
11 ),

```

Save all dan refresh flutter untuk melihat perubahan.



UNIVERSITAS
MIKROSKIL

6. **Menggunakan Progress Indicator Determinate dan ToolTips.** Bedanya pada tahap ini digunakan sebuah Linear Indicator yang menampilkan lama waktu proses. Pada `content_widget.dart` buatlah seperti tampilan berikut:Ket



```

1  import 'package:flutter/material.dart';
2  import 'package:flutter_application/pertemuan13/pertemuan13_provider.dart';
3  import 'package:provider/provider.dart';
4
5  class ContentWidget extends StatelessWidget {
6    const ContentWidget({Key? key}) : super(key: key);
7
8    @override
9    Widget build(BuildContext context) {
10      final prov = Provider.of<Pertemuan13Provider>(context);
11
12      return Center(
13        child: Container(
14          child: prov.sedangMemanggang == true
15            ? SizedBox(
16              width: 100,
17              child: TweenAnimationBuilder(
18                tween: Tween<double>(begin: 0, end: 1),
19                duration: Duration(seconds: prov.sliderValue.round()),
20                builder: (context, double value, _) =>
21                  LinearProgressIndicator(
22                    value: value,
23                  ),
24            )
25            : prov.selesaiMasak == true
26              ? Tooltip(
27                message: 'Ayam Panggang',
28                child: Image.network(
29                  'https://bit.ly/ayampanggang22',
30                  width: 300,
31                ),
32              )
33            : Container(),
34        ),
35      );
36    }
37  }
38

```

Keterangan:

- Widget ini akan menampilkan `linearIndicator` jika tombol "Panggang" diklik yaitu `sedangMemanggang = true`. Jika `selesaiMasak` maka akan tampil gambar, selain itu tampilkan sebuah container kosong.
- **Baris 17 - 23**, Digunakan untuk menampilkan animasi `linearIndicator` yang menampilkan lama proses sesuai durasi yang diset. Digunakan widget `TweenAnimationBuilder` untuk memberikan animation builder pada `linearIndicator`.

Tampilkan `ContentWidget()` pada `pertemuan13_screen`, seperti gambar berikut:



```

1  Widget build(BuildContext context) {
2    final prov = Provider.of<Pertemuan13Provider>(context);
3    return Scaffold(
4      appBar: AppBar(title: const Text('Pertemuan13')),
5      body: Padding(
6        padding: const EdgeInsets.all(16.0),
7        child: Column(crossAxisAlignment: CrossAxisAlignment.start, children: [
8          Text('Durasi Memanggang: ${prov.sliderValue.round().toString()}'),
9          const SlideWidget(),
10         const Align(
11           child: ProgressIndicatorWidget(),
12           alignment: Alignment.bottomRight,
13         ),
14         const SizedBox(height: 100),
15         const ContentWidget()
16       ]),
17    ),
18  );
19 }

```

Save dan reload untuk melihat perubahan.

Latihan Praktek

1. Tambahkan toolTips pada slider, sehingga dapat memberi tahu value dari slider.
2. Ubah circularProgressIndicator menjadi tipe **determine indicator** sehingga menampilkan lama proses.
3. Lakukan pengecekan kondisi jika *durasi memanggang* = 0, maka tampilkan sebuah alertDialog untuk menginformasikan bahwa durasi tidak boleh 0. Selain itu, lakukan proses **mulaiMemanggang()**.



UNIVERSITAS
MIKROSKIL