BAB 4
RESPONSIVE & ADAPTIVE

4.1 Tujuan Pembelajaran :

1. Mahasiswa mengetahui konsep *Responsive*
2. Mahasiswa mengetahui jenis *Responsive*
3. Mahasiswa dapat membuat *Responsive*

4.2 *Software* yang dibutuhkan :

1. Flutter SDK
2. Emulator Android
3. Android Studio
4. Visual Studio Code

4.3 Responsive

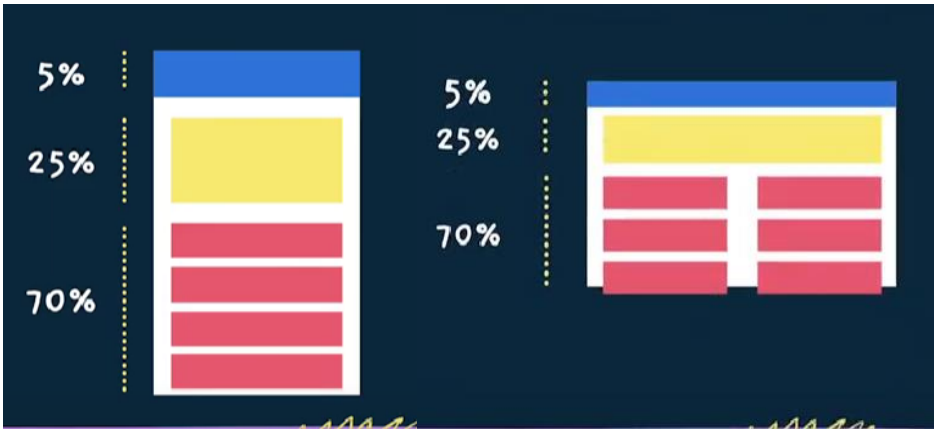
Salah satu tujuan utama flutter adalah membuat kerangka kerja yang memungkinkan untuk mengembangkan aplikasi dari satu basis kode yang terlihat dan terasa hebat di platform manapun. Artinya, aplikasi yang dibuat mungkin muncul di layar dengan berbagai ukuran, mulai dari jam tangan, ponsel lipat dengan dua layer, hingga monitor beresolusi tinggi.

Istilah tersebut menggambarkan konsep untuk skenario dari responsive. Responsive biasanya memiliki tata letak yang disesuaikan dengan ukuran layar yang tersedia. Misalnya menata ulang UI jika pengguna mengubah ukuran layar, atau mengubah orientasi perangkat. Ini sangat diperlukan ketika aplikasi yang sama dapat berjalan di berbagai perangkat.

A. Media Query

Media query merupakan sebuah Teknik yang dapat digunakan oleh desainer untuk mengatur sebuah tampilan dari layout berbeda untuk setiap device (perangkat). Dengan menggunakan media query ini, secara otomatis tampilan layar dapat dengan bebas menyeting tampilan di berbagai resolusi dan lebar, maupun tinggi.

Berikut merupakan contoh layout dari *Media Query*



Berikut contoh penerapan *Source Code* dari *Media Query*

```
import 'dart:math';
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: HomePage(),
    );
  }
}
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final mediaQueryHeight =
MediaQuery.of(context).size.height;
    final mediaQueryWidth =
MediaQuery.of(context).size.width;
    final myAppBar = AppBar(
      title: Text("Media Query"),
    );

    final bodyHeight = mediaQueryHeight -
        myAppBar.preferredSize.height -
        MediaQuery.of(context).padding.top;
```

```
    final bool isLandscape =
        MediaQuery.of(context).orientation ==
Orientation.landscape;
    return Scaffold(
      appBar: myAppBar,
      body: Center(
        child: (isLandscape)
            ? Column(
                children: [
                  Container(
                    height: bodyHeight * 0.5,
                    width: mediaQueryWidth,
                    color: Colors.blue,
                  ),
                  Container(
                    height: bodyHeight * 0.5,
                    color: Colors.blue,
                    child: GridView.builder(
                      gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(
                        crossAxisCount: 2,
                        mainAxisSpacing: 10,
                        crossAxisSpacing: 10,
                      ),
                      itemCount: 100,
                      itemBuilder: (context, index) {
                        return GridTile(
                          child: Container(
                            color: Color.fromARGB(
                              255,
                              Random().nextInt(156),
                              Random().nextInt(156),
                              Random().nextInt(156),
                            ),
                          ),
                        );
                      },
                    ),
                  ),
                ],
              )
            : Column(
                children: [
```

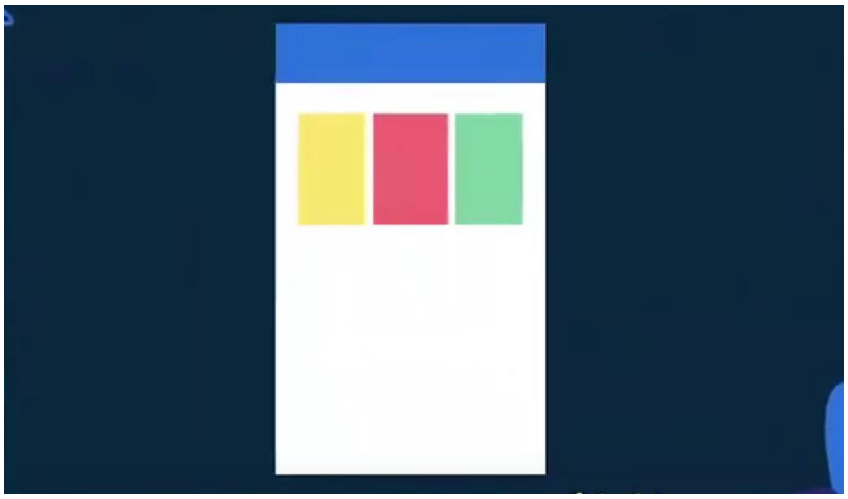
```
                Container(
                  height: bodyHeight * 0.3,
                  width: mediaQueryWidth,
                  color: Colors.blue,
                ),
                Container(
                  height: bodyHeight * 0.7,
                  color: Colors.red,
                  child: ListView.builder(
                    itemCount: 100,
                    itemBuilder: (context, index) {
                      return ListTile(
                        leading: CircleAvatar(),
                        title: Text("Ini Adalah list"),
                      );
                    },
                  ),
                ),
              ],
            ),
        ),
      );
  }
}
```

B. Flexible dan Expanded

Flexible Widget merupakan widget yang digunakan untuk membuat tampilan yang mudah diatur dan bersifat fleksibel. Untuk membuat suatu widget dengan ukuran perbandingan tertentu menggunakan flexible widget akan sangat membantu. Untuk menggunakan flexible widget harus berada didalam Row, Column, atau Flex.

Sedangkan Expanded Widget merupakan sebuah widget yang memperluas anak dari Row, Column, atau flex sehingga mengisi ruang yang tersedia. Dengan menggunakan expanded widget, membuat children dari row, column dan flex akan mengisi ruang yang tersedia.

Berikut merupakan contoh layout dari *Flexible* dan *Expanded*



1. Berikut contoh penerapan *Source Code* dari *Flexible*

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomePage(),
    );
  }
}
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Flexible dan Expanded"),
      ),
      body: Row(
        children: [
          Container(
            height: 100,
            color: Colors.red,
            child: Text("Text1Text1"),
          ),
          Container(
            height: 100,
```

```
            color: Colors.blue,
            child: Text("Text1"),
          ),
          Flexible(
            fit: FlexFit.tight,
            flex: 2,
            child: Container(
              height: 100,
              color: Colors.green,
              child: Text("Text2Text2"),
            ),
          ),
          Flexible(
            fit: FlexFit.loose,
            flex: 1,
            child: Container(
              height: 100,
              color: Colors.amber,
              child: Text("Text3Text3"),
            ),
          ),
        ],
      ),
    );
  }
}
```

2. Berikut contoh penerapan *Source Code* dari *Expanded*

```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomePage(),
    );
  }
}
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        title: Text("Flexible dan Expanded"),
      ),
      body: Row(
        children: [
          Container(
            height: 100,
            color: Colors.red,
            child: Text("Text1Text1"),
          ),
          Container(
            height: 100,
            color: Colors.blue,
            child: Text("Text1"),
          ),
          Expanded(
            flex: 2,
            child: Container(
              height: 100,
              color: Colors.green,
              child: Text("Text2Text2"),
            ),
          ),
          Expanded(
            flex: 1,
            child: Container(
              height: 100,
              color: Colors.amber,
              child: Text("Text3Text3"),
            ),
          ),
        ],
      ),
    );
  }
}
```

C. Fitted Box

Widget FittedBox adalah single child layout widget, yang artinya widget ini hanya dapat memiliki satu child yang ditampilkan. Dalam kasus ini, widget Row ditambahkan sebagai turunan dari widget FittedBox. Widget

Row memiliki dua Container sebagai turunannya. Biasanya, child kedua dari widget Column akan melebar ke satu sisi saat merender semua child. Namun dengan FittedBox, masalah melebar widget ini dapat diatasi. Jadi fungsi dari FittedBox yaitu mensklakan dan memposisikan child didalam widget parent. Berikut merupakan contoh layout dari *Fitted Box*

Berikut contoh penerapan *Source Code* dari *Fitted Box*

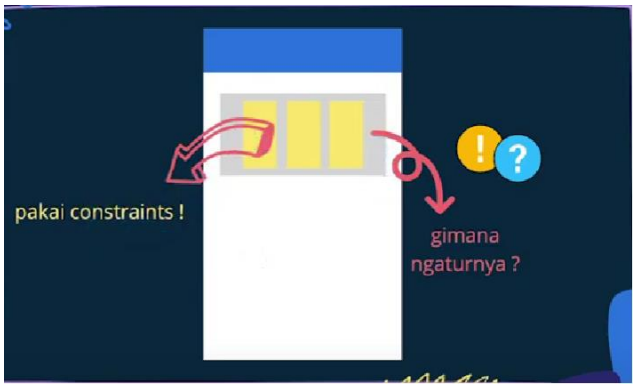
```
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text("Fitted Box"),
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Center(
              child: Container(
                color: Colors.blue,
                width: 300,
```

```
                child: Container(
                  color: Colors.blue,
                  width: 300,
                  height: 110,
                  child: FittedBox(
                    alignment: Alignment.centerRight,
                    fit: BoxFit.fill,
                    child:
Image.network("https://picsum.photos/500/500"),
                  ),
                ),
              ),
            ),
            SizedBox(height: 20),
            Center(
              child: Container(
                color: Colors.blue,
                width: 300,
                child: Container(
                  color: Colors.blue,
                  width: 300,
                  height: 110,
                  child: FittedBox(
                    alignment: Alignment.centerRight,
                    fit: BoxFit.fill,
                    child: Text(
                      "HALOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO",
                      style: TextStyle(
                        fontSize: 35,
                        color: Colors.white,
                      ),
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

D. Layout Builder

LayoutBuilder membantu widget tree di dalam widget flutter yang dapat bergantung pada ukuran widget asli. Di dalam Flutter dapat menggunakan widget layout sebagai parameter. Layout Builder memiliki dua parameter yaitu konteks dan Boxconstrant yang mana BuildContext mengacu pada widget, tetapi box constraint lebih penting untuk memberikan lebar pada parent widget yang digunakan untuk mengelola child sesuai dengan ukuran parent-nya.

Perbedaan utama antara Media Query dengan LayoutBuilder adalah Media Query menggunakan konteks ukuran layer, bukan ukuran widget khusus. Sementara Layout Builder dapat menentukkan lebar dan tinggi maksimum widget apa pun. Berikut contoh layout dari *Layout Builder*



Berikut contoh penerapan *source code* dari *Layout Builder*

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomePage(),
    );
  }
}
class HomePage extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    final widthApp = MediaQuery.of(context).size.width;
    final heightApp = MediaQuery.of(context).size.height;
    final paddingTop = MediaQuery.of(context).padding.top;
    final myAppBar = AppBar(
      title: Text("Layout Builder"),
    );
    final heightBody = heightApp - paddingTop -
myAppBar.preferredSize.height;
    return Scaffold(
      appBar: myAppBar,
      body: Container(
        width: widthApp,
        height: heightBody * 0.3,
        color: Colors.grey,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            myContainer(widthApp),
            myContainer(widthApp),
            myContainer(widthApp),
          ],
        ),
      ),
    );
  }
}
class myContainer extends StatelessWidget {
  final double widthApp;
  myContainer(this.widthApp);
  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(builder: (context, Constraints) {
      return Container(
        width: widthApp * 0.25,
        height: Constraints.maxHeight * 0.5,
        color: Colors.amber,
      );
    });
  }
}
```