Modul Praktikum

Pemrograman Mobile

6

RESPONSIVE & THEME

TUJUAN PEMBELAJARAN

A. Mahasiswa memahami responsive pada flutter

B. Mahasiswa memahami penggunaan material

C. Mahasiswa memahami penggunaan animated container pada flutter

D. Mahasiswa memahami penggunaan Theme

E. Mahasiswa memahami penggunaan SnackBar

F. Mahasiswa memahami penggunaan AlertDialog

G. Mahasiswa memahami pembuatan Introduction Screen

DASAR TEORI

1. Media Query

Dari pembelajaran materi-materi sebelumnya kita mengetahui bahwa flutter merupakan salah satu framework untuk mengembangkan aplikasi di berbagai platform. Platform pada mobile sendiri memiliki berbagai macam ukuran layar. Oleh karena itu kita harus memahami bagaimana untuk menerapkan layout yang mampu menyesuaikan dengan berbagai ukuran layar yang berbeda, salah satu contoh nya kita dapat menggunakan **MediaQuery**. MediaQuery adalah salah satu kelas yang ada pada flutter yang dapat kita gunakan untuk mendapatkan ukuran dan juga orientasi layar.

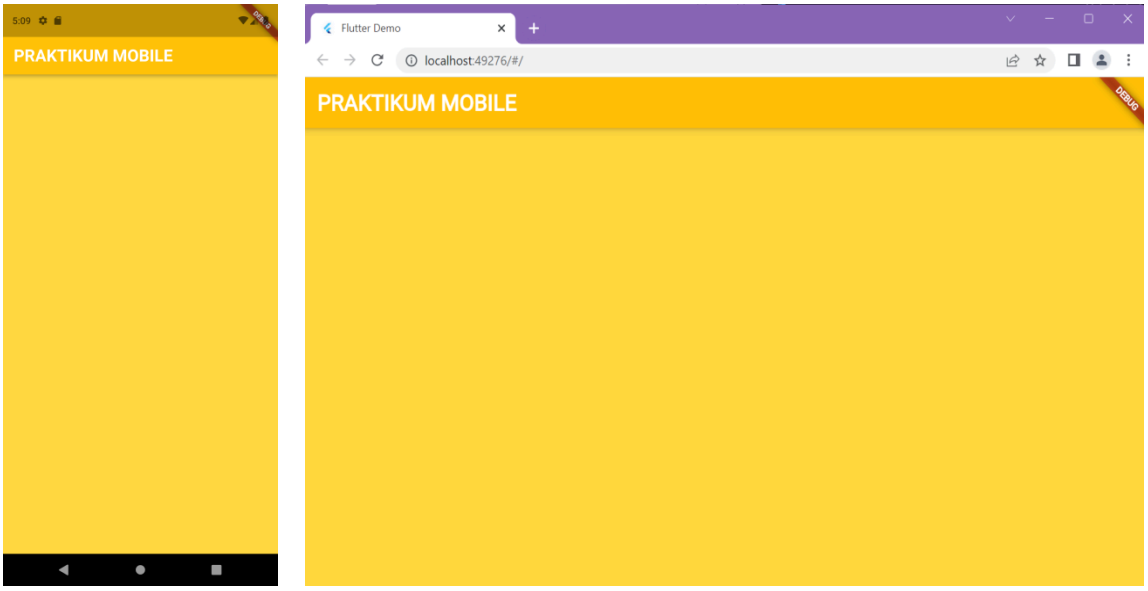
Berikut adalah contoh penerapan MediaQuery
Pertama kita buat variabel di dalam *widget build context* untuk tinggi dan lebar layout menggunakan *MediaQuery*.

```
var lebar = MediaQuery.of(context).size.width;
var tinggi = MediaQuery.of(context).size.height;
```

setelah itu kita menggunakan widget container yang didalam nya menggunakan parameter width dan height untuk memanggil variabel yang kita buat tadi.

```
  Container(
        width: lebar,
        height: tinggi,
        color: Colors.amberAccent,
      ),
```

Output:



Dari output di atas dapat di lihat bahwa dengan menggunakan MediaQuery maka panjang lebar mengikuti layout layar.

2. Material

Material pada flutter adalah suatu sistem design untuk mengatur tampilan atau UI. Material ini terdiri dari panduan, komponen, dan alat yang dapat disesuaikan untuk perancangan user interface di berbagai platform seperti Android, iOS, Web, dan Flutter. Komponen Material umumnya meliputi beberapa kategori yaitu display, navigation, actions, input, dan communication.

Flutter secara default juga telah menerapkan komponen-komponen *Material Design* yang telah disediakan melalui library material. Setiap kita membuat sebuah project ataupun class dalam flutter program akan secara default mengimport library material

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

Tanpa mengimpor library tersebut program akan error karena seperti yang dijelaskan sebelumnya dalam material terdapat beberapa kategori dan salah satu di antara kategori tersebut selalu ada di dalam project atau class, contoh nya saja seperti widget yang selalu di gunakan yaitu *MaterialApp*. Widget tersebut merupakan dasar aplikasi yang menggunakan material. Widget seperti AppBar, Button, Snackbar, dan beberapa widget lainnya yang kita gunakan juga termasuk dalam komponen *Material*. Kita dapat melihat daftar lengkap widget-widget yang termasuk komponen Material di https://flutter.dev/docs/development/ui/widgets/material.

3. Animated Container

Animated Container adalah widget container sederhana dengan animasi.. Animated Container ini dapat dianimasikan dengan mengubah nilai propertinya yang sama dengan widget Container.

Berikut contoh penerapan Animated Container
Pertama buatlah variabel tipe boolean

```
  bool selected = false;
```

setelah itu di body kita menggunakan widget *GestureDetector* dan *AnimatedContainer* untuk mengatur animasi pada container.

```
body: GestureDetector(
        onTap: () {
          setState(() {
            selected = !selected;
          });
        },
        child: Center(
          child: AnimatedContainer(
            width: selected ? 300.0 : 100.0,
            height: selected ? 300.0 : 100.0,
            color: selected ? Colors.red : Colors.blue,
            alignment:
                selected ? Alignment.center :
AlignmentDirectional.topCenter,
```

```
            duration: const Duration(seconds: 2),
            curve: Curves.fastOutSlowIn,

          ),
        ),
      ),
```

Ketika container di tap maka container otomatis akan berubah sesuai nilai property yang sudah di tetapkan.

4. Theme

Penggunaan Tema memungkinkan Anda untuk mendefinisikan atribut-atribut tampilan umum seperti warna, tipografi, button style, dan elemen UI lainnya secara terpusat. Dengan cara ini, Anda dapat memastikan bahwa seluruh aplikasi memiliki tampilan yang konsisten, sehingga pengguna merasa nyaman dan akrab dengan antarmuka pengguna. Sedangkan penggunaan Feedback membantu Anda memberikan informasi kepada pengguna dengan cara yang tidak mengganggu aliran kerja mereka, sehingga meningkatkan pengalaman pengguna.

Pada saat membangun sebuah aplikasi kita akan membuat banyak sekali tampilan berupa text dan button yang memiliki style tertentu. Pada Flutter kita akan menggunakan theme untuk mengatur nilai default aplikasi kita. Pada Flutter kita akan menggunakan sebuah class bernama Navigator.

Pertama tama buat file theme_screen.dart untuk memperlihatkan efek dari perubahan tema yang berisi kode berikut:

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

class ThemeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Nama Kalian'),
      ),
      body: Center(
```

```
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'Headline Large',
              style: Theme.of(context).textTheme.headlineLarge),
            SizedBox(height: 20),
            Text(
              'Body Large',
              style: Theme.of(context).textTheme.bodyLarge),
            Text(
              'Body Medium',
              style: Theme.of(context).textTheme.bodyMedium),
            Text(
              'Body Small',
              style: Theme.of(context).textTheme.bodySmall),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {},
              child: Text('Elevated Button'),
            ),
            SizedBox(height: 20),
            OutlinedButton(
              onPressed: () {},
              child: Text('Outlined Button'),
            ),
            SizedBox(height: 20),
            TextButton(
              onPressed: () {},
              child: Text('Text Button'),
            ),
          ],
        ),
      ),
    );
  }
}
```

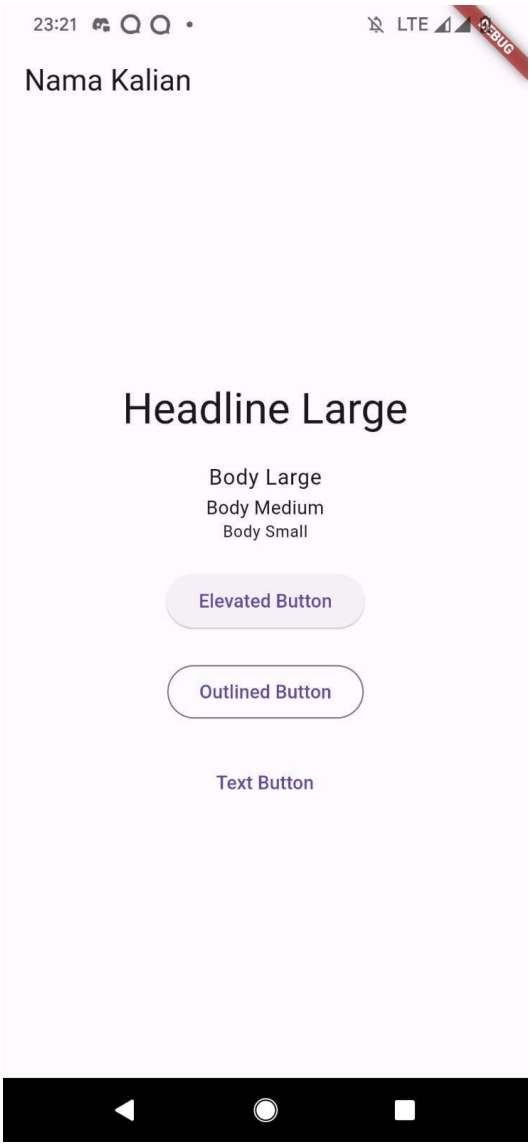
Selanjutnya, kita edit main.dart menjadi seperti ini:

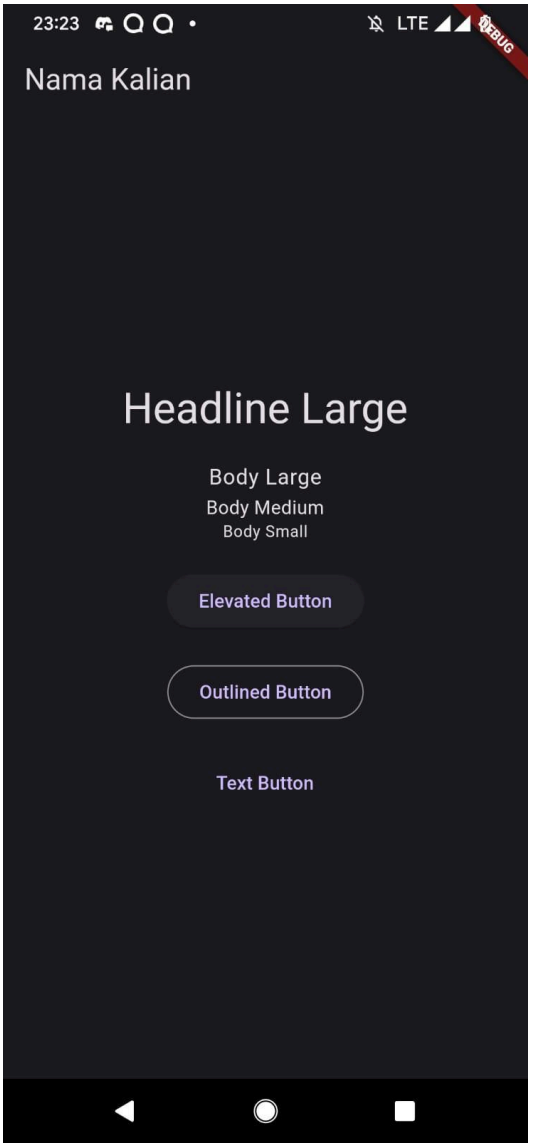
```
import 'package:flutter/material.dart';

import 'theme_screen.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Nama Kalian',
      theme: ThemeData(
        useMaterial3: true,
        brightness: Brightness.light,
      ),
      darkTheme: ThemeData(
        useMaterial3: true,
        brightness: Brightness.dark,
      ),
      themeMode: ThemeMode.system,
      home: ThemeScreen(),
    );
  }
}
```

Kita gunakan ThemeData untuk mendeklarasi Tema yang kita gunakan. Parameter useMaterial3 digunakan untuk mendeklarasi tema yang kita gunakan akan mengikuti guidelines dari Material 3 (Material You) yang merupakan guidelines terbaru untuk aplikasi android dari google. Saat mendeklarasi useMaterial3 sebagai benar, maka tema tersebut akan mengoverride tampilan komponen seperti button, switch dan lain lain yang semulanya Material 2 menjadi tampilan yang sesuai dengan Material 3.

Parameter brightness digunakan untuk menspesifikkan apakah ThemeData yang kita deklarasikan digunakan untuk light theme atau dark theme.

Lalu, jalankan aplikasi anda. Maka, akan terlihat halaman seperti ini dalam light mode atau dark mode:





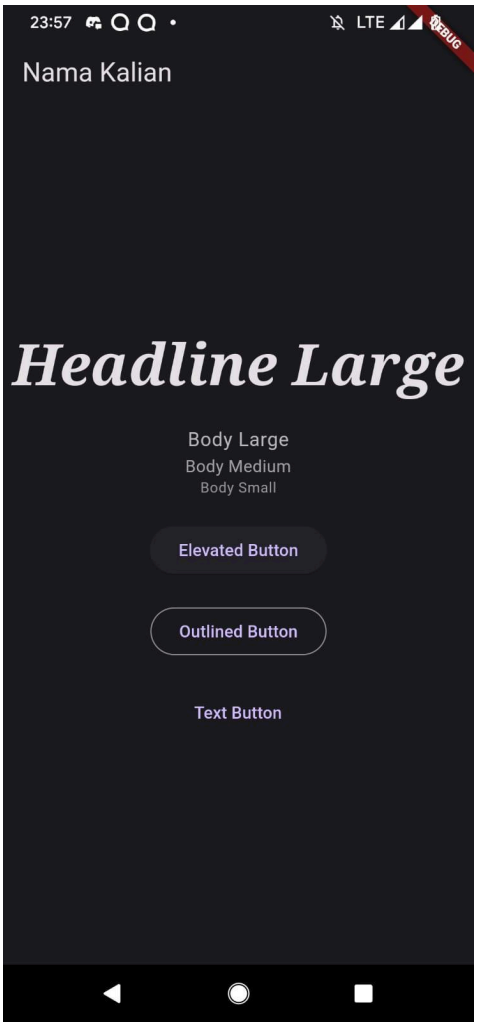
Pertama-tama kita ingin mengubah style dari widget text, maka isilah parameter textTheme di dalam ThemeData seperti dibawah:

```
    theme: ThemeData(
      useMaterial3: true,
      brightness: Brightness.light,
      textTheme: TextTheme(
        headlineLarge: TextStyle(
          fontSize: 48,
          fontWeight: FontWeight.w600,
          fontStyle: FontStyle.italic,
          fontFamily: 'Serif'
        ),
        bodyLarge: TextStyle(color: Colors.black87),
        bodyMedium: TextStyle(color: Colors.black54),
        bodySmall: TextStyle(color: Colors.black45),
      ),
    ),
    darkTheme: ThemeData(
      useMaterial3: true,
      brightness: Brightness.dark,
      textTheme: TextTheme(
        headlineLarge: TextStyle(
            fontSize: 48,
            fontWeight: FontWeight.w600,
            fontStyle: FontStyle.italic,
            fontFamily: 'Serif'
        ),
        bodyLarge: TextStyle(color: Colors.white70),
        bodyMedium: TextStyle(color: Colors.white60),
        bodySmall: TextStyle(color: Colors.white54),
      ),
    ),
```

TextTheme adalah koleksi TextStyle yang digunakan pada aplikasi flutter, dimana kita bisa mengubah salah satu textstyle sesuai dengan kita inginkan dan dapat kita gunakan di seluruh aplikasi. Parameter textTheme ada banyak meliputi guidelines Material 2 yang deprecated dan guidelines Material 3 yang terbaru. Untuk melihat semua style yang ada di TextTheme bisa kalian lihat di link berikut: https://m3.material.io/styles/typography/type-scale-tokens

fontSize digunakan untuk nilai besar font, fontWeight digunakan untuk menentukan ketebalan font, fontStyle digunakan untuk menentukan apakah italic atau tidak, dan fontFamily digunakan untuk menentukan font apa yang digunakan. Berikut cara untuk menggunakan custom font pada aplikasi Anda: https://docs.flutter.dev/cookbook/design/fonts

Lalu, jalankan aplikasi anda. Maka, akan terlihat perubahan gaya text pada halaman seperti ini dalam light mode atau dark mode:



5. SnackBar

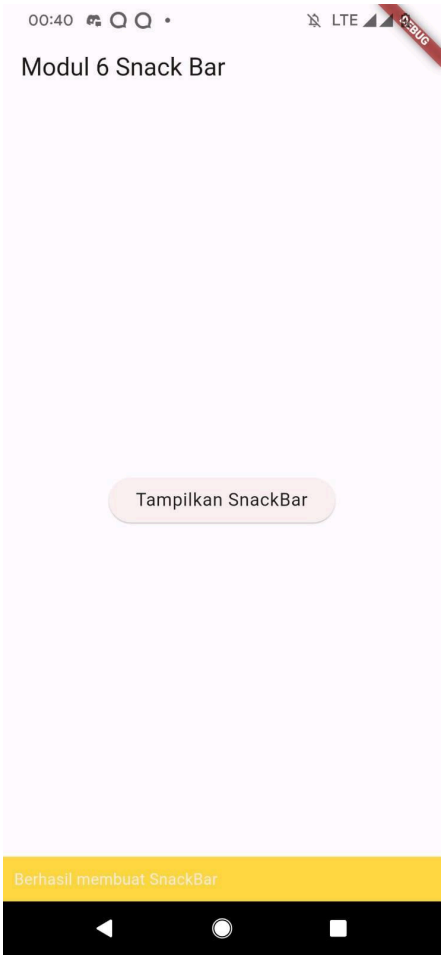
SnackBar adalah sebuah pesan singkat yang muncul di layar bawah hp kita. Pertama, buatlah StatelessWidget yang memiliki ElevatedButton yang akan digunakan sebagai trigger SnackBar.

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Modul 6 Snack Bar"),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {},
          child: Text(
            "Tampilkan SnackBar",
            style: Theme
                .of(context)
                .textTheme
                .bodyLarge,
          ),
        ),
      ),
    );
  }
}
```

Lalu pada property onPressed pada button buatlah variable yang menampung sebuah SnackBar, kemudian gunakanlah ScaffoldMessenger untuk menampilkan SnackBar kalian.

```
child: ElevatedButton(
  onPressed: () {
    final mySnackBar = SnackBar(
      content: Text("Berhasil membuat SnackBar"),
      duration: Duration(seconds: 3),
      padding: EdgeInsets.all(10),
      backgroundColor: Colors.amberAccent,
    );
    ScaffoldMessenger.of(context).showSnackBar(mySnackBar);
  },
```

Lalu, jalankan aplikasi anda. Maka, akan terlihat seperti ini:



6. Alert Dialog

Ketika kita membuat sebuah real project application, pastinya kita akan berhubungan denga API dan Server, terkadang server tidak selalu mulus seperti yang kita inginkan, untuk itu kita perlu memberi tahu pada user bahwa ada sesuatu kesalahan pada aplikasi kita. Kita dapat memberikan Warning, Error, Pesan dan Info kepada pengguna menggunakan Alert Dialog di Flutter.

Selain itu alert dialog di Flutter juga dapat memberikan informasi bahwa proses upload gambar berhasil, password atau email salah ketika login dan berbagai case lainnya di aplikasi

kita, oleh karena itu pada tutorial Konsep Koding kali ini kita akan mempelajari mengenai Alert Dialog pada Flutter karena itu salah satu hal yang penting utuk dipelajari.

Pertama kali kita buat stateless widget Bernama DialogPage di file baru Bernama dialog_page.dart yang isinya seperti ini:

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Nama Kalian"),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {},
          child: Text("Tampilkan Alert Dialog"),
        ),
      ),
    );
  }
}
```

Lalu di dalam DialogPage kita buat sebuah fungsi bernama showAlertDialog dengan parameter BuildContext, String judul, dan String konten dengan nilai kembalian Future<dynamic> yang berisi seperti ini:

```
  Future<dynamic> showAlertDialog(BuildContext context, String judul, String
konten) {
    return showDialog(
      context: context,
      builder: (context) {},
    );
  }
```

showDialog adalah sebuah fungsi yang akan menampilkan dialog yang disediakan oleh builder ke dalam halaman yang berjalan. Widget yang kita bisa gunakan pada builder ada 3: Dialog, SimpleDialog, AlertDialog. Kali ini kita akan coba menggunakan AlertDialog.

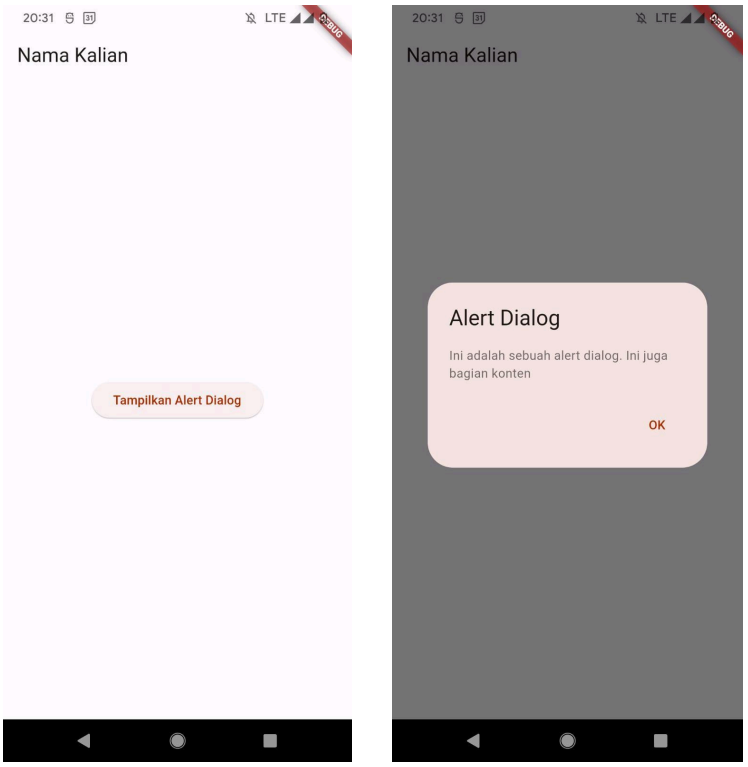
```
      builder: (context) {
        return AlertDialog(
          title: Text(judul),
          content: Text(konten),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text("OK"),
            ),
          ],
        );
      },
```

AlertDialog membutuhkan 3 parameter: title untuk menampilkan bagian judul, content untuk menampilkan bagian konten, actions adalah kumpulan aksi dibawah konten yang biasanya diisi oleh button ok dan cancel.

Lalu kita panggil fungsi showAlertDialog di dalam onPressed ElevatedButton:

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Nama Kalian"),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            showAlertDialog(
              context,
              "Alert Dialog",
              "Ini adalah sebuah alert dialog. Ini juga bagian konten",
            );
          },
          child: Text("Tampilkan Alert Dialog"),
        ),
      ),
    );
  }
```

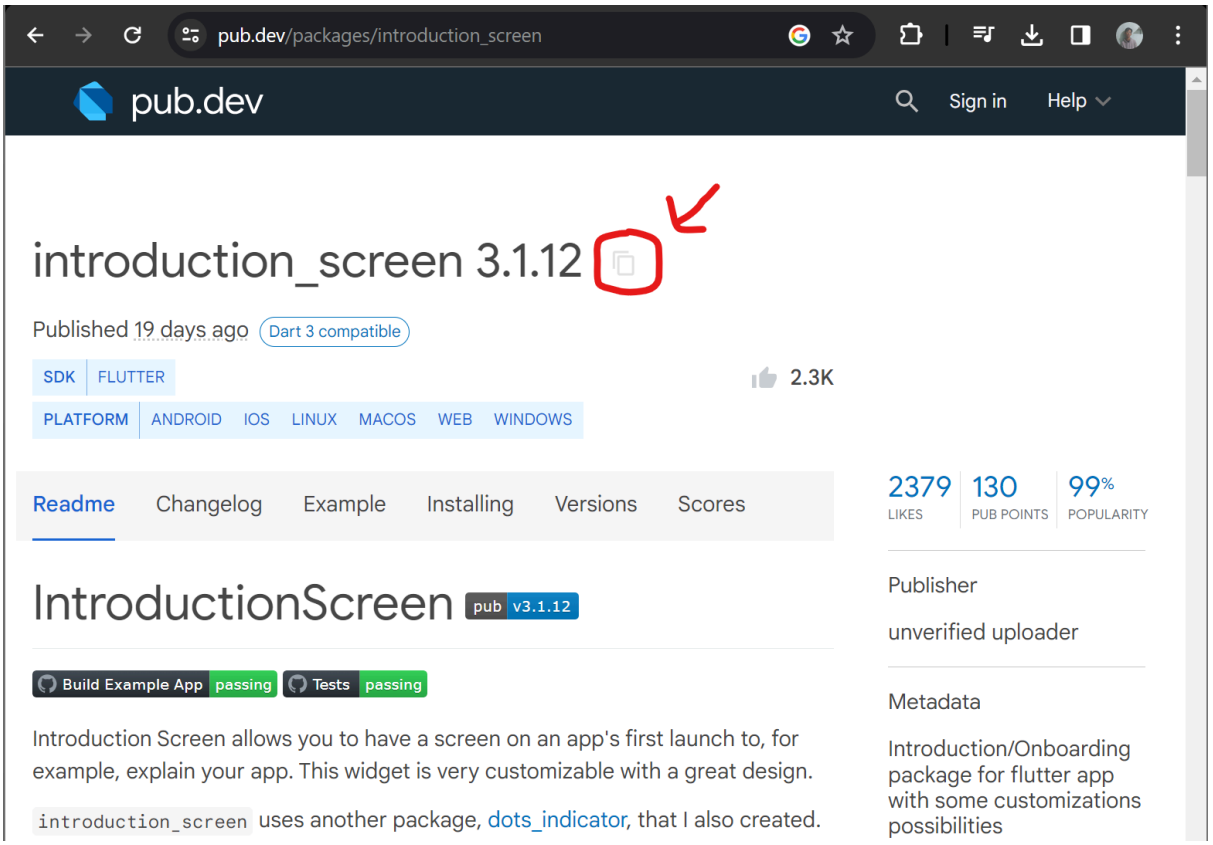
Lalu, jalankan aplikasi anda. Maka, akan terlihat seperti ini:



7. Introduction Screen

Introduction Screen memungkinkan Anda memiliki layar pada peluncuran pertama aplikasi untuk, misalnya, menjelaskan aplikasi Anda. Widget ini sangat dapat disesuaikan dengan desain yang bagus.

Pertama tama kita menuju ke halaman pub.dev dari packages introduction_screen. Lalu klik tombol copy. Berikut link ke halaman tersebut: https://pub.dev/packages/introduction_screen



Lalu paste ke pubspec.yaml dibawah cupertino_icons:

```
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your applicati
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  introduction_screen: ^3.1.12
```

Lalu buatlah sebuah stateless widget Bernama IntroductionPage didalam file introduction_page.dart yang berisi kode berikut:

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

import 'snackbar_page.dart';

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

  @override
  Widget build(BuildContext context) {
    return IntroductionScreen(
      next: Text("Selanjutnya"),
      done: Text("Selesai"),
      onDone: () {
        Navigator.of(context).pop();
        Navigator.of(context).push(
          MaterialPageRoute(
            builder: (context) {
              return SnackBarPage();
            },
          ),
        );
      },
      pages: [
        PageViewModel(
          title: "The First Page",
          body: "Number 1",
          image: Image.network("https://picsum.photos/800/500"),
        ),
        PageViewModel(
          title: "The Second Page",
          body: "Number 2",
          image: Image.network("https://picsum.photos/800/600"),
        ),
        PageViewModel(
          title: "The Third Page",
          body: "Number 3",
          image: Image.network("https://picsum.photos/900/700"),
        ),
      ],
    );
  }
}
```

Parameter next digunakan untuk menentukan apa yang ditampilkan pada button next, done digunakan untuk menentukan apa yang ditampilkan pada button done, onDone adalah fungsi

lambda yang akan dipanggil pada saat tombol done ditekan. Dan terakhir parameter pages adalah sebuah list yang berisi halaman dari introduction screen. Di dalam pages harus berisi widget PageViewModel yang berasal dari packages itu sendiri. PageViewModel memerlukan parameter title untuk judul, body untuk konten, dan image untuk gambar yang akan ditampilkan pada halaman.

Lalu, jalankan aplikasi anda. Maka, akan terlihat seperti ini: