LAPORAN PRAKTIKUM

PEMROGRAMAN PERANGKAT BERGERAK

MODUL VII

NAVIGASI DAN NOTIFIKASI

DISUSUN OLEH :
MUHAMMAD FARIZ NUR HIDAYAT / 2211104069
KELAS : SE-06-2

Asisten Praktikum :
Muhammad Faza Zulian Gesit Al Barru
Aisyah Hasna Aulia

Dosen Pengampu :
Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2024

NAVIGASI DAN NOTIFIKASI

1. Model

Pada umumnya, hampir seluruh aplikasi yang dibuat akan bekerja dengan data. Data dalam sebuah aplikasi memiliki sangat banyak bentuk, tergantung dari aplikasi yang dibuat. Setiap data yang diterima atau dikirimkan akan lebih baik apabila memiliki standar yang sama. Hampir mustahil melakukan peneliharaan *project* yang kompleks tanpa model.

- **Membuat Model Class**

  Untuk membuat model, buatlah direktori baru pada folder lib project flutter, kemudian buat sebuah file class dart dengan nama filenya adalah nama data yang ingin dijadikan model. Sebagai contoh, ketika mambuat model user dengan response json di bawah ini:

```
{
  "user_id": 1,
  "id": 13,
  "title": "Bedroom Pop"
}
```

  Maka buatlah file user.dart di dalam folder models, dengan code seperti di bawah ini:

```
class Album  {
  final int userId;
  final int id;
  final String title;

  const Album({
    required this.userId,
    required this.id,
```

```
    required this.title,
  });

  factory Album.fromJson(Map<String, dynamic> json) {
    return Album(
      userId: json['user_id'],
      id: json['id'],
      title: json['title'],
    );
  }
}
```

2. Navigation

Dalam Flutter, navigation merujuk pada cara berpindah antar halaman (atau tampilan) di aplikasi. Sistem navigasi di Flutter berbasis route dan navigator. Setiap halaman atau layar di Flutter disebut sebagai route, dan Navigator adalah widget yang mengelola stack dari route tersebut.

- **Navigation Pindah Halaman**

  Untuk melakukan navigasi ke halaman lain pada Flutter, dapat gunakan code seperti di bawah ini:

```
Navigator.push(
  Context, MaterialPageRoute(builder: (context) =>
SecondRoute()),
);
```

  Untuk melakukan navigasi kembali ke halaman sebelumnya, dapat gunakan code seperti di bawah ini:

```
Navigator.pop(context);
```

  Potongan code diatas harus diletakkan dalam function sebuah widget, misalnya pada *onPressed* milik ElevatedButton. SecondRoute pada contoh dapat diubah menjadi halaman bari yang dituju.

- **Navigation Mengirim Data**

  Untuk dapat melakukan navigasi dengan mengirimkan data ke halaman lain, perlu disiapkan 2 hal, yakni:

  1) Halaman baru memiliki parameter data yang diminta.
  2) Halaman awal mengirimlan data melalui parameter.

  Untuk dapat melakukan hal tersebut, kita dapat membuat sebuah halaman baru seperti di bawah ini:

```
class DetailScreen extends StatelessWidget {
  const DetailScreen({Key? key, required this.title}) :
super(key: key);

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(title),
      ),
    );
  }
}
```

  Pada code di atas, telah dibuat halaman baru yang memiliki parameter data yang diinginkan yaitu atribut title dengan tipe data string.

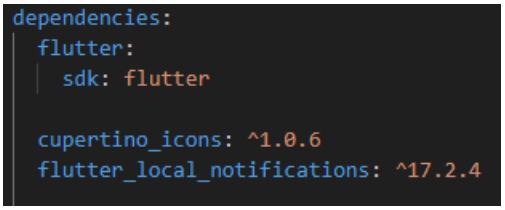
```
Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => DetailScreen(title:
“Detail User”)),
);
```

  Berbeda dengan contoh navigasi sebelumnya, pada navigasi di atas ditambahkan parameter title berisi string “Detail User” yang akan

dikirimkan ke halaman baru.

3. Notification

Untuk mengirimkan notifikasi dalam aplikasi Flutter, dapat digunakan package bernama flutter_local_notifications. Tambahkan terlebih dahulu package tersebut ke dalam *pubspec.yaml*.



Konfigurasi Gradle

Pastikan minSdkVersion di *android/app/build.gradle* diatur ke 21 atau lebih tinggi:

```
defaultConfig {
    minSdkVersion 21
}
```

Konfigurasi iOS

Buka file *ios/Runner/Info.plist* dan tambahkan kode berikut untuk mendapatkan izin notifikasi pada iOS:

```
    <key>UIBackgroundModes</key>
    <array>
        <string>fetch</string>
        <string>remote-notification</string>
    </array>
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSAllowsArbitraryLoads</key>
        <true/>
    </dict>
```

```
      <key>NSUserNotificationUsageDescription</key>
<string>We need your permission to send notifications.</string>
```

Setelah menambahkan package, ubah file *AndroidManifest.xml* dengan menambahkan barisan code seperti di bawah ini:

```
<!-- Izin untuk menghidupkan kembali notifikasi setelah reboot -->
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<!-- Izin untuk mengaktifkan getaran pada notifikasi --> <uses-
permission android:name="android.permission.VIBRATE" />
```

Tambahkan potongan kode di bawah ini di luar build dari stateful widget tersebut:

```
FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin
=FlutterLocalNotificationsPlugin();
```

Dengan membuat sebuah object Flutter LocalNotificationsPlugin, operasi yang terdapaat dalam package flutter_notification dapat digunakan.

Override method initState dari widget tersebut dengan menambahkan code seperti di bawah ini:

```
void initState() {
    super.initState();
    var initializationSettingsAndroid =
    AndroidInitializationSettings('flutter_devs');
    var initializationSettingsIOs = IOSInitializationSettings();
    var initSetttings = InitializationSettings(
        initializationSettingsAndroid, initializationSettingsIOs);
    flutterLocalNotificationsPlugin.initialize(initSetttings,
        onSelectNotification: onSelectNotification);
}
```

Kemudian buat sebuah function yang akan mengendalikan ketika notifikasi

dipilih:

```
Future onSelectNotification(String payload) {
    Navigator.of(context).push(MaterialPageRoute(builder: (_) {
        return NewScreen(
            payload: payload,
        );
    }));
}
```

Buatlah sebuah widget baru yang akan menjadi halaman berikutnya setelah notifikasi dipilih:

```
class NewScreen extends StatelessWidget {
    String payload;
    NewScreen({
        @required this.payload,
    });
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
            title: Text(payload),
        ));
    }
}
```

Kemudian buat sebuah function yang fungsinya untuk menampilkan notifikasi sederhana untuk android:

```
showNotification() async {
    var android = new AndroidNotificationDetails(
        'id', 'channel ', 'description',
        priority: Priority.High, importance: Importance.Max);
    var iOS = new IOSNotificationDetails();
    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
```

```
        0, 'Flutter devs', 'Flutter Local Notification Demo',
platform,
        payload: 'Welcome to the Local Notification demo ');
}
```

- AndroidNotificationDetails akan berisi mengenai detail notifikasi pada android.
- IOSNotificationDetails akan berisi mengenai detail notifikasi pada iOS.
- Kunci untuk menampilkan notifikasi terletak pada pemanggilan function flutterLocalNotificationsPlugin yang berfungsi untuk menampilkan notifikasi sesuai dengan platform yang digunakan.

Keseluruhan code di atas akan membentuk sebuah file seperti di bawah ini:

BUKTI PRAKTIKUM

main.dart

Source Code

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan7/pages/mypage.dart';

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

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: Mypage(),
    );
  }
}

```

Deskripsi Program

Program di atas adalah sebuah aplikasi Flutter sederhana yang mendemonstrasikan penggunaan widget statis. Berikut adalah deskripsi singkatnya:

1. Tujuan Program: Aplikasi ini berfungsi sebagai aplikasi dasar Flutter yang menampilkan halaman utama (`Mypage`), yang diimpor dari berkas `mypage.dart`.

2. Main Function: Fungsi `main()` berfungsi sebagai titik awal aplikasi, memanggil `runApp(MyApp())` untuk menjalankan aplikasi.

3. MyApp Class:
- `MyApp` adalah kelas utama aplikasi yang diturunkan dari `StatelessWidget`.
- Aplikasi ini menggunakan `MaterialApp` sebagai pembungkus utama dengan judul "Flutter Demo".
- Tema aplikasi diatur dengan skema warna berbasis `deepPurple` menggunakan `ThemeData` dan `ColorScheme.fromSeed()`.

4. Homepage: Halaman utama aplikasi adalah `Mypage()`, yang dirujuk sebagai widget layar awal (`home`).

Program ini menggunakan gaya Material 3 dan dapat diperluas dengan menambahkan konten lebih lanjut ke dalam halaman `mypage.dart`.

MODELS
product.dart

Source Code

```dart
class product {
  final int id;
  final String nama;
  final double harga;
  final String gambarUrl;
  final String deskripsi;

  // constructor
  product({
    required this.id,
    required this.nama,
    required this.harga,
    required this.gambarUrl,
    required this.deskripsi,
  });

  // method untuk mengkonversi JSON => object Product
  factory product.fromJson(Map<String, dynamic> json) {
    return product(
      id: json['id'],
      nama: json['nama'],
      harga: json['harga'],
      gambarUrl: json['gambarUrl'],
      deskripsi: json['deskripsi'],
    );
  }

  // method untuk mengkonversu object product => JSON
  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'nama': nama,
      'harga': harga,
      'gambarUrl': gambarUrl,
      'deskripsi': deskripsi,
    };
  }
}

```

Deskripsi Program

1. Tujuan Program: Kode ini mendefinisikan sebuah kelas `product` yang digunakan untuk merepresentasikan sebuah produk dengan atribut seperti ID, nama, harga, URL gambar, dan deskripsi. Kelas ini juga menyediakan metode untuk konversi dari dan ke format JSON.

2. Kelas `product`:
- Memiliki 5 atribut: `id` (int), `nama` (String), `harga` (double), `gambarUrl` (String), dan `deskripsi` (String). Semua atribut ini bersifat final dan wajib diinisialisasi melalui constructor.

3. Constructor:
- Constructor dari kelas `product` menerima parameter yang di-`required` untuk mengisi atribut-atributnya, sehingga memastikan setiap objek produk memiliki nilai untuk setiap atribut.

4. Metode `fromJson`:
- Merupakan factory constructor yang digunakan untuk mengubah data dalam bentuk Map (JSON) menjadi sebuah objek `product`. Data JSON harus berisi kunci-kunci yang sesuai dengan atribut kelas.

5. Metode `toJson`:
- Metode ini digunakan untuk mengonversi objek `product` kembali ke dalam bentuk Map (JSON), dengan mengembalikan nilai dari atribut-atribut objek sebagai pasangan kunci-nilai JSON.

Program ini berguna untuk mempermudah serialisasi dan deserialisasi data produk ketika berkomunikasi dengan API atau penyimpanan data dalam format JSON.

PAGES
detailpage.dart

Source Code

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

class DetailPage extends StatelessWidget {
  const DetailPage({super.key, required this.data});

  final Widget data;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Detail Page'),
        centerTitle: true,
        backgroundColor: Colors.blue[400],
      ),
      body: Center(
        child: data,
      ),
    );
  }
}

```

Deskripsi Program

1. Tujuan Program: Kode ini mendefinisikan sebuah halaman bernama `DetailPage` dalam aplikasi Flutter. Halaman ini menampilkan sebuah widget yang diterima sebagai parameter dan digunakan untuk menampilkan detail konten.

2. Kelas `DetailPage`:
- `DetailPage` adalah kelas yang diturunkan dari `StatelessWidget`, yang berarti halaman ini bersifat statis dan tidak berubah setelah dibangun.
- Halaman ini menerima sebuah widget melalui parameter `data` yang akan ditampilkan di tubuh halaman.

3. Atribut `data`:
- Atribut `data` bertipe `Widget` diterima sebagai parameter yang diperlukan dalam constructor. Widget ini nantinya akan ditampilkan di dalam halaman.

4. Tampilan UI:
- AppBar: Bagian atas halaman memiliki AppBar dengan judul "Detail Page", yang terletak di tengah (menggunakan `centerTitle`), dan memiliki latar belakang berwarna biru.

- Body: Di dalam `body`, widget `data` ditampilkan di tengah halaman menggunakan `Center` widget, yang menampilkan konten yang diberikan secara dinamis.

Program ini memungkinkan tampilan halaman detail yang dapat menampilkan berbagai konten dinamis yang disediakan melalui parameter `data`.

mypage.dart

Source Code

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan7/models/product.dart';
import 'package:pertemuan7/pages/detailpage.dart';

class Mypage extends StatelessWidget {
  Mypage({super.key});

  final List<product> products = [
    product(
        id: 1,
        nama: 'Mouse',
        harga: 300000.00,
        gambarUrl:
'https://resource.logitechg.com/w_386,ar_1.0,c_limit,f_auto,q_auto,dpr_2.0/d_transparent.gif/content/dam/gaming/en/products/g502x-plus/gallery/g502x-plus-gallery-1-black.png?v=1'
,
        deskripsi: 'lumayan'),
    product(
        id: 2,
        nama: 'keyboard Mechanical',
        harga: 500000.00,
        gambarUrl:
'https://resource.logitech.com/w_1600,c_limit,q_auto,f_auto,dpr_1.0/d_transparent.gif/content/dam/logitech/en/products/keyboards/mx-mechanical/gallery/mx-mechanical-keyboard-top-view-graphite-us.png?v=1&quot'
,
        deskripsi: 'Mechanical keyboard agak mahal'),
    product(
        id: 3,
        nama: 'Headphone Gaming',
        harga: 450000.00,
        gambarUrl:
'https://m.media-amazon.com/images/I/61CGHv6kmWL.AC_UF894,1000_QL80.jpg',
        deskripsi: 'bassnya mantep pol'),
  ];

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('class Model'),
        centerTitle: true,
        backgroundColor: Colors.blue[400],
      ),
      body: ListView.builder(
        itemCount: products.length,
        itemBuilder: (context, index) {
          final product = products[index];
          return ListTile(
            leading: Image.network(
              product.gambarUrl,
              width: 70,
              height: 70,
            ),
            title: Text(product.nama),
            subtitle: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text('Rp${product.harga}'),
                Text('Deskripsi: ${product.deskripsi}'),
              ],
            ),
            onTap: () {
              print('tap layar berhasil');
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (_) => DetailPage(
                    data: Icon(Icons.notifications_outlined),
                  ),
                ),
              );
            },
          );
        },
      ),
    );
  }
}
```

Deskripsi Program

1. Tujuan Program: Kode ini merupakan bagian dari aplikasi Flutter yang menampilkan daftar produk pada halaman utama (`Mypage`). Setiap produk memiliki gambar, nama, harga, dan deskripsi yang diambil dari sebuah daftar. Saat pengguna mengetuk salah satu produk, mereka akan diarahkan ke halaman detail (`DetailPage`).

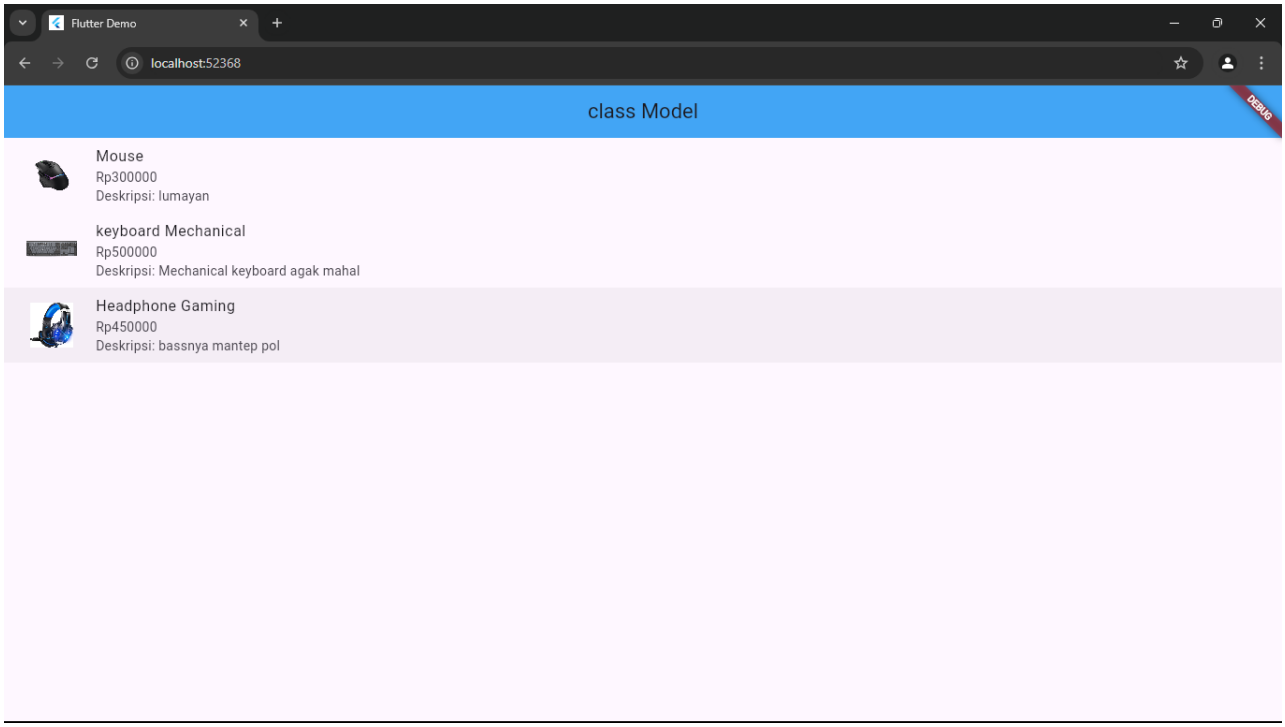
2. Kelas `Mypage`:
- `Mypage` adalah kelas yang menampilkan halaman utama dan mewarisi `StatelessWidget`, yang berarti kontennya statis dan tidak berubah setelah dibangun.
- Terdapat daftar produk (`products`) yang berisi tiga item produk, masing-masing dengan atribut `id`, `nama`, `harga`, `gambarUrl`, dan `deskripsi`.

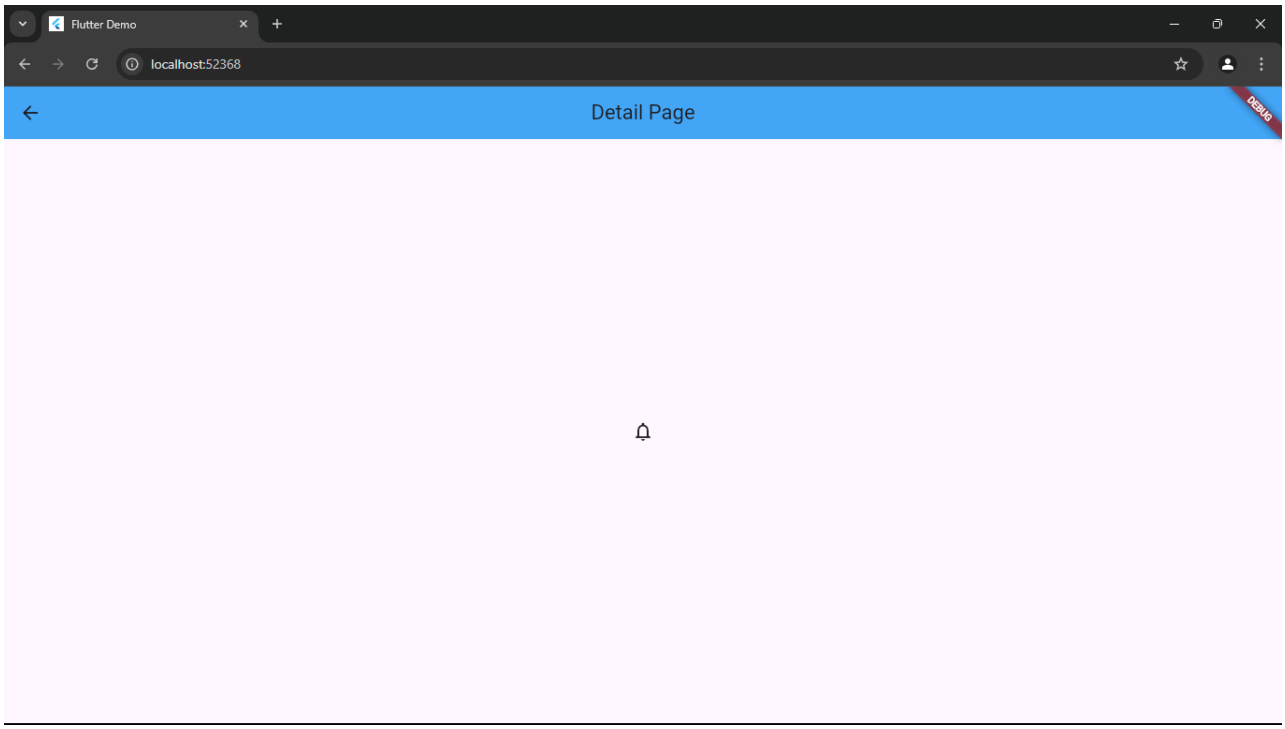
3. Tampilan UI:
- AppBar: Halaman memiliki AppBar dengan judul "class Model", yang terletak di tengah dan memiliki latar belakang berwarna biru.
- Body: Menggunakan `ListView.builder` untuk membangun daftar produk secara dinamis. Setiap item dalam daftar adalah `ListTile`, yang menampilkan gambar produk (`leading`), nama (`title`), harga dan deskripsi (`subtitle`).

4. Navigasi ke DetailPage:
- Saat pengguna mengetuk item dalam daftar (dengan `onTap`), aplikasi menampilkan pesan di console (menggunakan `print`), dan menavigasikan pengguna ke halaman DetailPage. Di halaman detail, sebuah ikon notifikasi ditampilkan melalui parameter `data`.

Program ini memberikan daftar produk yang dinamis dan menyediakan navigasi ke halaman detail saat pengguna memilih produk tertentu.

Output Aplikasi Keseluruhan





Tugas Mandiri (Unguided)

1. (Soal) Buatlah satu project untuk menampilkan beberapa produk dan halaman *e-commerce* dengan menerapkan class model serta navigasi halaman.

   Note: Jangan lupa sertakan source code, screenshoot output, dan deskripsi program. Kreatifitas menjadi nilai tambah.