MODUL PEMROGRAMAN MOBILE

CIM 430

MODUL PRAKTIKUM 6

DEVELOPMENT DESIGN NAVIGATION UI MOBILE APPS ANDROID WITH GETX MICRO-FRAMEWORK

DISUSUN OLEH

7174 – Sawali Wahyu, S.Kom, M.Kom

8126 – Jefry Sunupurwa Asri, S.Kom, M.Kom

7176 – Iksan Ramadhan, S.Kom, M.Kom

UNIVERSITAS ESA UNGGUL

FAKULTAS ILMU KOMPUTER

TAHUN 2021

DEVELOPMENT DESIGN NAVIGATION UI MOBILE APPS ANDROID WITH GETX MICRO-FRAMEWORK

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

1. Mahasiswa Mampu memahami konsep micro-framework dalam membuat konsep Navigation UI.
2. Mahasiswa mampu mengimplementasi konsep dasar route management dan state management dalam merancang Navigation UI.

B. Content of Material

PRAKTIKUM 6 – KONSEP DASAR MEMODELKAN MICRO-FRAMEWORK

GetX adalah micro-framework yang bertujuan untuk meminimalkan boilerplate yang dikombinasikan dengan sintaks yang rapi dan pendekatan yang sederhana. Saat mengembangkan aplikasi flutter dengan GetX, percaya semuanya akan terasa lebih praktis.

Url Package : https://pub.dev/packages/get

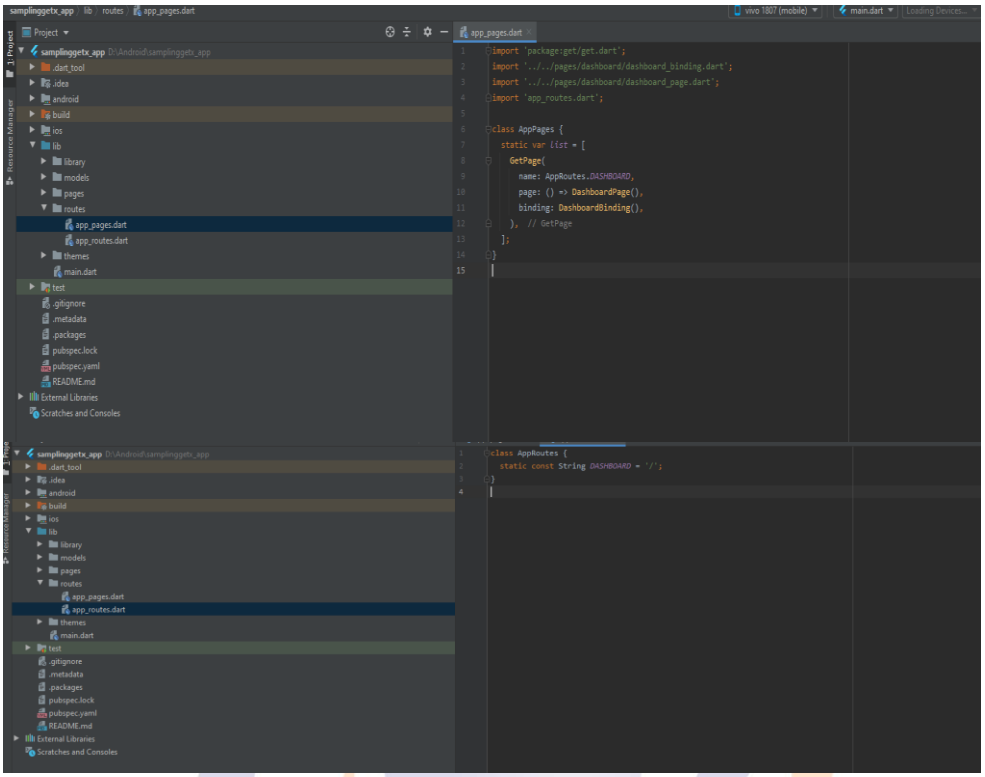
Package GetX begitu spesial karena fiture yang ditawarkan sangat menarik. Memiliki tiga (3) pilar utama yaitu sebagai state management, route management dan Dependency management.

1) Ronte Management

Jika sebelumnya kita manage route dengan source code seperti dibawah ini

```
Navigator.push(
      context, MaterialPageRoute(builder: (context) {
            return HomePage()
            })
);
```

Maka dengan getx kita dapat mempersingkatnya menjadi :



2) State Management

GetX menawarkan tiga tipe state management yaitu

a) **Reactive** : selalu mengikuti perubahan apapun yang terjadi di state
b) **On update** : konsepnya seperti provider dan listener dimana saat ada aksi update maka akan merebuild widget
c) gabungan reactive dan on update

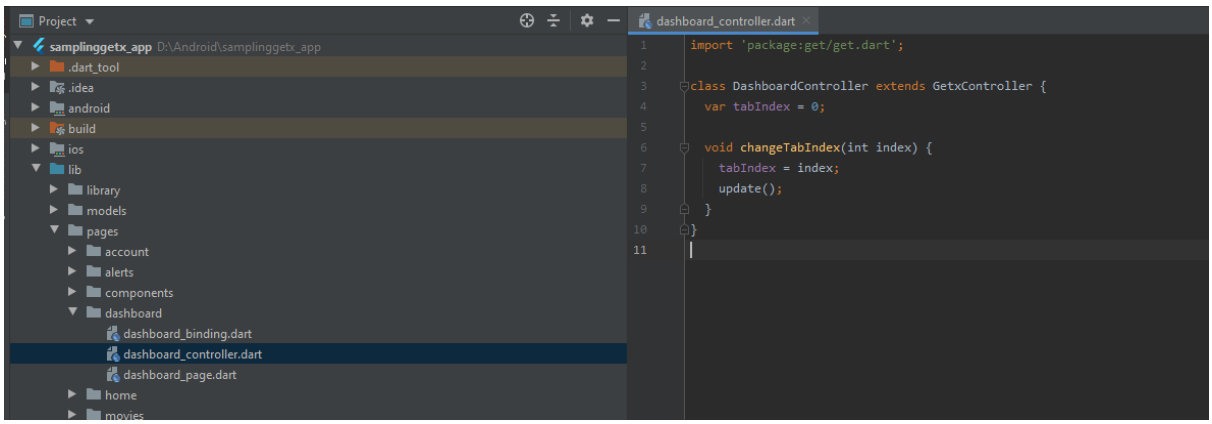
dashboard_page.dart

```dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../../pages/alerts/alerts_page.dart';
import '../../pages/account/account_page.dart';
import '../../pages/home/home_page.dart';
import '../../pages/movies/movies_page.dart';

import 'dashboard_controller.dart';

class DashboardPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetBuilder<DashboardController>(
      builder: (controller) {
        return Scaffold(
          body: SafeArea(
            child: IndexedStack(
              index: controller.tabIndex,
              children: [
                HomePage(),
                MoviesPage(),
                AlertsPage(),
                AccountPage(),
              ],
            ),  // IndexedStack
          ),  // SafeArea
          bottomNavigationBar: BottomNavigationBar(...),  // BottomNavigationBar
        );  // Scaffold
      },
    );  // GetBuilder
  }

  _bottomNavigationBarItem({IconData icon, String label}) {
    return BottomNavigationBarItem(
      icon: Icon(icon),
      label: label,
    );  // BottomNavigationBarItem
  }
}
```



3) Dependency Management

GetX memiliki pengelola dependensi yang sederhana yang memungkinkan kita mengambil class yang sama dengan Bloc atau Controller Anda hanya dengan 1 baris kode, tanpa provider context, tanpa inheritedWidget

Pendekatan GetX cukup sederhana. Alih-alih membuat sebuah instance secara langsung, kita membungkusnya dengan sebuah instance Get (class), seperti ini:

```
Get.put(Controller());
```

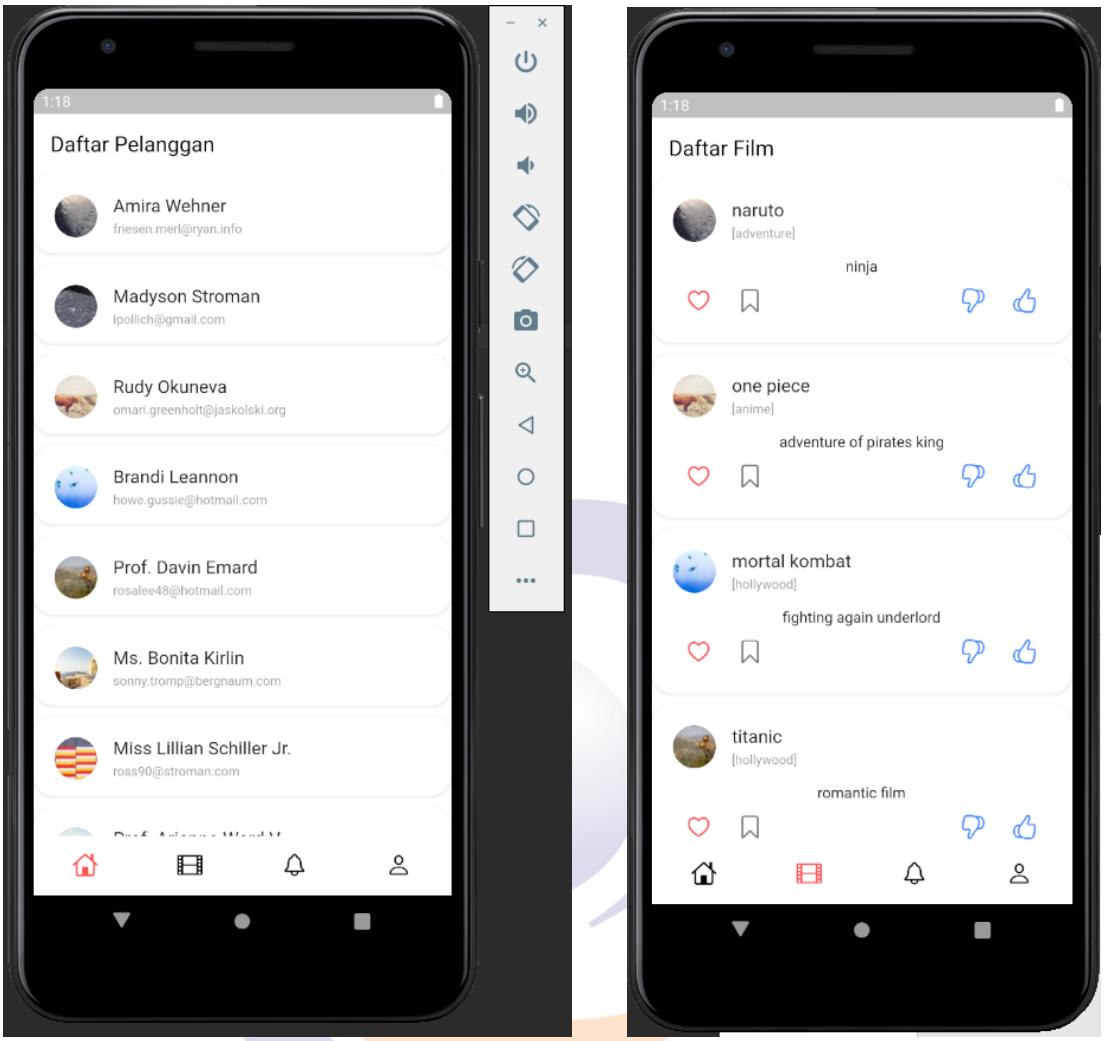
Get.put membuat dependensi tersedia untuk semua child route. Jadi, jika kita perlu mengakses instance yang sama di beberapa class lain, maka kita dapat melakukannya dengan menggunakan Get.find.

dashboard_binding.dart

```dart
import 'package:get/get.dart';
import '../../pages/account/account_controller.dart';
import '../../pages/home/home_controller.dart';
import '../../pages/movies/movies_controller.dart';
import 'dashboard_controller.dart';

class DashboardBinding extends Bindings {
  @override
  void dependencies() {
    Get.lazyPut<DashboardController>(() => DashboardController());
    Get.lazyPut<HomeController>(() => HomeController());
    Get.lazyPut<AccountController>(() => AccountController());
    Get.lazyPut<MoviesController>(() => MoviesController());
  }
}
```

OUTPUT PROGRAM APLIKASI :

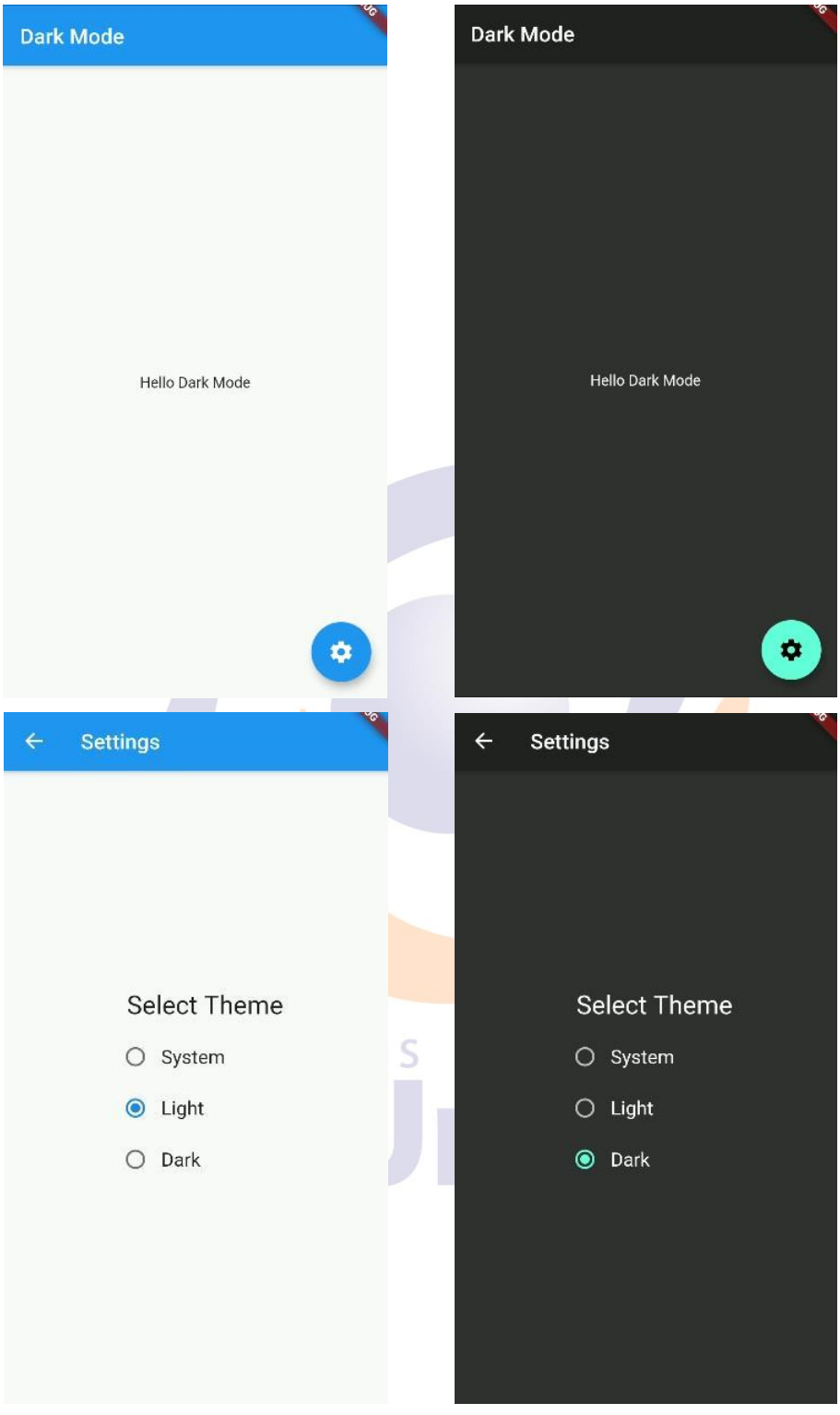


PRAKTIKUM 2 – DESAIN THEME ON MOBILE APPS SCREEN

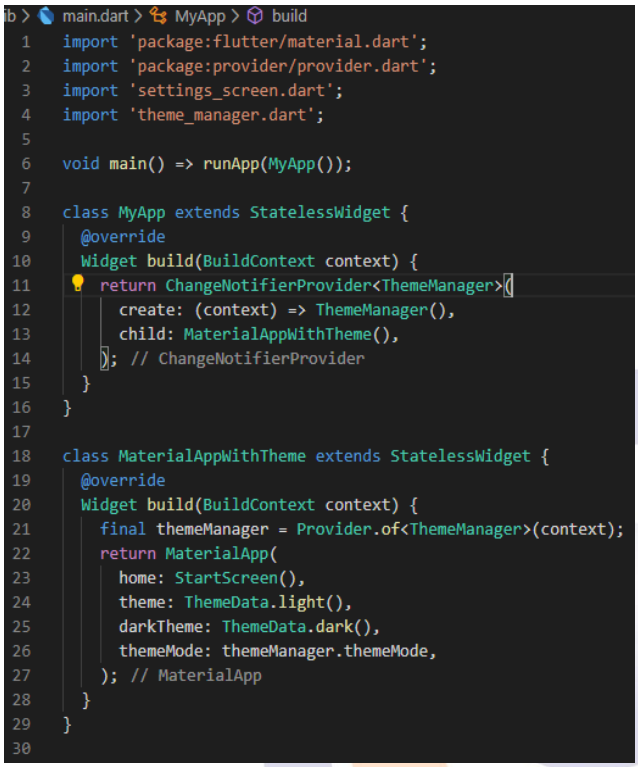
State Management adalah sebuah desain dalam coding di mana kita dapat memisahkan antara logic dan view, Tujuannya adalah agar logic dapat kembali digunakan. Pada kotlin atau java biasanya kita mengenal istilah **MVVM** atau **MVP**, pada web **MVC**, maka di Flutter ada **BLoC**, **Provider**, **Redux**, atau **MobX**.

Pada bagian ini akan menggunakan salah satunya yakni **Provider**. Contohnya adalah untuk mengubah tema warna tampilan aplikasi seperti contoh di bawah :

Light Theme **Dark Theme**



Untuk menggunakan package **Provider**, pastikan Anda memasukkannya dalam **pubspec.yaml**



```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'settings_screen.dart';
import 'theme_manager.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider<ThemeManager>(
      create: (context) => ThemeManager(),
      child: MaterialAppWithTheme(),
    ); // ChangeNotifierProvider
  }
}

class MaterialAppWithTheme extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final themeManager = Provider.of<ThemeManager>(context);
    return MaterialApp(
      home: StartScreen(),
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      themeMode: themeManager.themeMode,
    ); // MaterialApp
  }
}
```

```dart
class StartScreen extends StatelessWidget {
  StartScreen({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dark Mode"),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Hello Dark Mode"),
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => SettingsScreen(),
            ), // MaterialPageRoute
          );
        },
        tooltip: 'Settings',
        child: Icon(Icons.settings),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

```dart
      darkTheme: ThemeData.dark(),
      themeMode: themeManager.themeMode,
    ); // MaterialApp
  }
}

class StartScreen extends StatelessWidget {
  StartScreen({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dark Mode"),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Hello Dark Mode"),
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => SettingsScreen(),
            ), // MaterialPageRoute
          );
        },
        tooltip: 'Settings',
        child: Icon(Icons.settings),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```