TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIII
NETWORKING

Disusun Oleh :

Rizky Hanifa Afania / 2211104017

SE-06-01

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

SOAL

1. **Apa yang dimaksud dengan state management pada Flutter?**

   Jawab:

   State management dalam Flutter adalah cara untuk mengelola "state" atau data yang berubah-ubah dalam aplikasi, seperti teks pada layar, posisi tombol, atau daftar item yang ditampilkan. Selain itu, state management bisa digunakan untuk memisahkan antara logic dan view, yang mana logic tersebut juga bisa reusable. State adalah informasi yang mempengaruhi tampilan aplikasi dan dapat berubah seiring waktu, misalnya saat pengguna berinteraksi dengan aplikasi.

2. **Sebut dan jelaskan komponen-komponen yang ada di dalam GetX.**

   Jawab:

   GetX adalah salah satu jenis state management yang mempunyai performa tinggi, injection dependency yang cerdas dan manajemen route yang cepat dan praktis. Berikut ini adalah komponen-komponen yang ada di dalam GetX:

   a. **State Management (Pengelolaan State)**

      GetX mempunya dua cara untuk mengelola state, yaitu Reactive State dan Simple State.

      - **Reactive State Management**

        Menggunakan class Rx untuk membuat data reaktif, sehingga UI otomatis diperbarui setiap kali data berubah. Berikut komponen utamanya:

        1. Rx<Type>: menjadikan data reaktif (misalnya RxInt, RxString).
        2. .obs: adalah shortcut untuk membuat data reaktif.

      - **Simple State Management**

        Menggunakan GetBuilder atau GetX widget untuk mengupdate UI secara manual tanpa membuat data reaktif. Berikut komponen utamanya:

        1. GetBuilder: digunakan untuk mengupdate widget hanya jika ada perubahan state.

   b. **Route Management (Manajemen Navigasi)**

      GetX mempunyai cara untuk mengelola navigasi tanpa perlu Navigator bawaan Flutter. Berikut komponen utamanya:

      - Get.to(): berfungsi untuk berpindah ke halaman lain.
      - Get.off(): berfungsi untuk berpindah ke halaman lain dan menutup halaman sebelumnya.

- Get.back(): berfungsi untuk kembali ke halaman sebelumnya.
- Get.arguments: berfungsi untuk engirim data antar halaman.

c. Dependency Injection (DI)

GetX mempunyai built-in dependency injection yang digunakan untuk mengelola lifecycle object dan membuatnya tersedia di mana saja. Berikut komponen utamanya:

- Get.put(): berfungsi untuk menyimpan instance controller yang dapat diakses global.
- Get.lazyPut(): berfungsi untuk menyimpan instance tetapi hanya membuatnya saat dibutuhkan.
- Get.find(): berfungsi untuk mengambil instance yang telah disimpan.

d. Middleware

Middleware digunakan untuk mengontrol akses ke halaman. Misalnya untuk memeriksa apakah pengguna sudah login sebelum masuk ke halaman tertentu. Berikut komponen utamanya:

- GetMiddleware: class yang berfungsi untuk membuat middleware.
- redirect: berfungsi untuk mengarahkan pengguna ke halaman lain jika syarat tertentu tidak terpenuhi.

e. Snackbar, Dialog, dan BottomSheet

GetX mempunyai utilitas bawaan untuk menampilkan notifikasi, dialog, atau bottom sheet tanpa memerlukan BuildContext, dengan komponen utama yaitu:

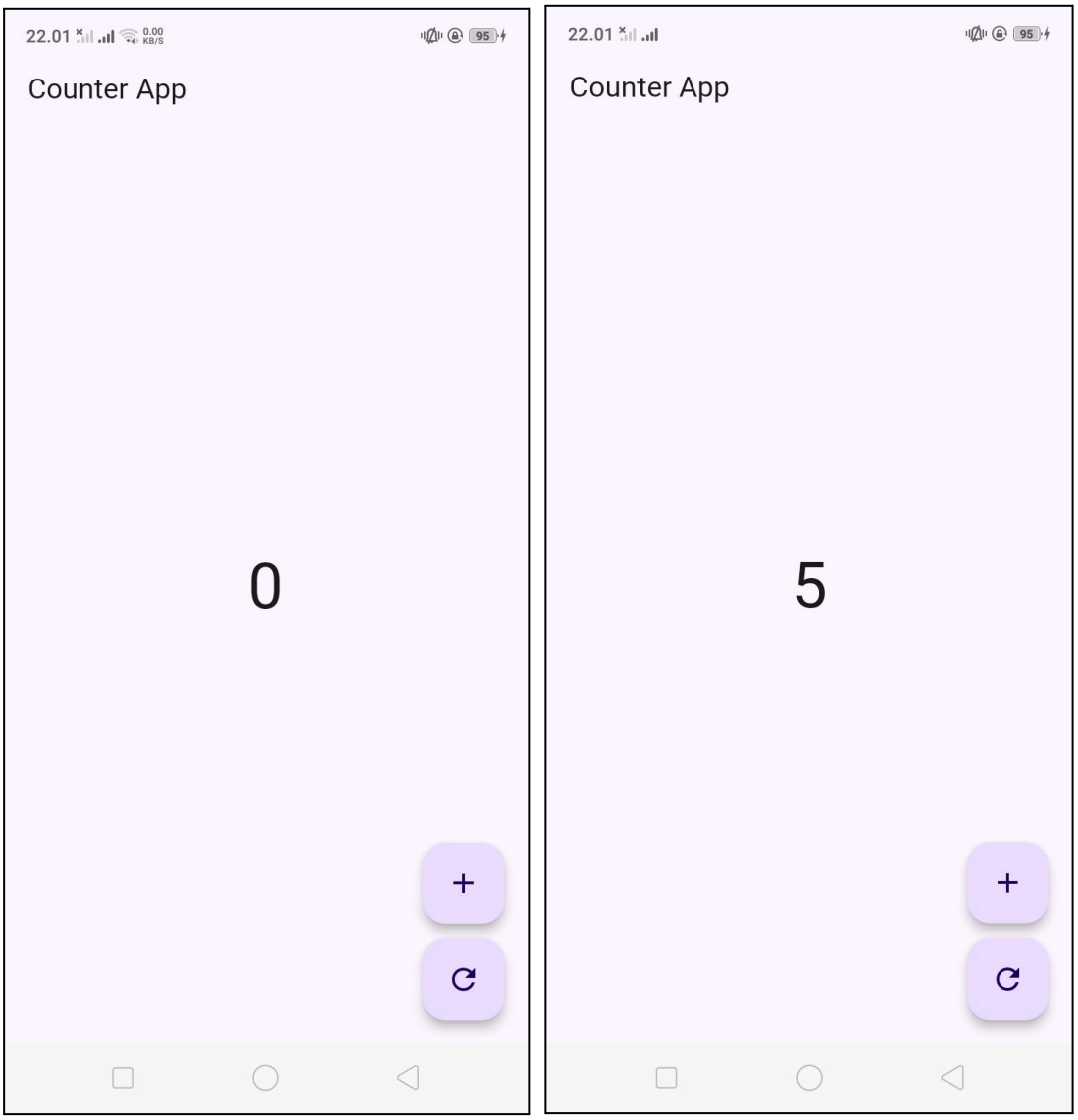
- Get.snackbar: berfungsi untuk menampilkan snackbar.
- Get.defaultDialog: berfungsi untuk menampilkan dialog.
- Get.bottomSheet: berfungsi untuk menampilkan bottom sheet.

3. **Lengkapilah code di bawah ini, dan tampilkan hasil outputnya serta jelaskan.**

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

/// Controller untuk mengelola state counter
class CounterController extends GetxController {
  // Variabel untuk menyimpan nilai counter
  var count = 0.obs;

  // Fungsi untuk menambah nilai counter
  void increment() {
    count++;
  }

  // Fungsi untuk mereset nilai counter
  void reset() {
    count.value = 0;
  }
}

/// Halaman utama aplikasi
class HomePage extends StatelessWidget {
  // Menginisialisasi controller
  final CounterController controller = Get.put(CounterController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Counter App")),
      body: Center(
        // Obx digunakan untuk membuat widget reaktif terhadap perubahan data
        child: Obx(() {
          return Text(
            "${controller.count}", // Menampilkan nilai counter
            style: TextStyle(fontSize: 48),
          );
        }),
      ),
      floatingActionButton: Column(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          FloatingActionButton(
            onPressed: () {
              controller.increment(); // Menambah nilai counter
            },
            child: Icon(Icons.add),
          ),
          SizedBox(height: 10),
          FloatingActionButton(
            onPressed: () {
              controller.reset(); // Mereset nilai counter
            },
            child: Icon(Icons.refresh),
          ),
        ],
      ),
    );
  }
}

void main() {
  runApp(GetMaterialApp(
    debugShowCheckedModeBanner: false,
    home: HomePage(),
  ));
}
```

Screenshoot Output

Tampilan awal program, angkanya masih 0. Ketika pengguna menekan tombol (+) maka angkanya akan bertambah. Di bawah ini adalah tampilan ketika pengguna menekan tombol sebanyak 5 kali sehingga angka di layar berubah menjadi 5. Ketika tombol reset di klik, maka angka berubah menjadi 0 lagi.



Deskripsi Program

Source diatas berguna untuk menghitung (counter app) yang menggunakan GetX. GetX adalah sebuah library Flutter yang digunakan untuk mempermudah mengelola state dan interaksi antar komponen. Aplikasi ini terdiri dari dua bagian utama yaitu, Controller dan View (UI). Controller adalah tempat logika aplikasi dikelola, seperti menambah nilai (increment) atau mereset nilai counter. View adalah tampilan antarmuka yang menunjukkan data (nilai counter) kepada pengguna dan merespons interaksi mereka.

Ketika pengguna menekan tombol Tambah (+) aplikasi akan memanggil fungsi increment() di controller untuk meningkatkan nilai counter dan secara otomatis

memperbarui tampilan karena variabel count bersifat reaktif (dideklarasikan dengan .obs). Lalu saat tombol Reset ditekan, fungsi reset() akan dijalankan untuk mengatur ulang nilai counter menjadi nol dan tampilan akan langsung disesuaikan tanpa perlu logika tambahan. Dengan cara ini, program menjaga alur kerja yang sederhana dan efisien: controller mengelola data, sedangkan UI hanya menampilkan hasil perubahan data secara otomatis.