PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIII

NETWORKING

Disusun Oleh:

Nadia Putri Rahmaniar / 2211104012

S1 SE-06-01

Asisten Praktikum:

MuhammadFazaZulian 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?**

   State management dalam Flutter adalah proses mengatur perubahan data atau status aplikasi serta memastikan data tersebut selaras dengan UI. Dengan state management, logika bisnis dapat dipisahkan dari UI, sehingga pengelolaan data menjadi lebih efisien, terutama ketika aplikasi semakin kompleks dan membutuhkan pembaruan tampilan berdasarkan status tertentu. Flutter menyediakan beberapa pendekatan untuk state management, seperti setState, Provider, Riverpod, BLoC, dan GetX.

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

   GetX adalah framework state management yang populer karena kinerjanya yang unggul, sistem dependency injection yang sederhana, serta pengelolaan route yang cepat dan mudah digunakan. Berikut adalah beberapa komponen utama dalam GetX:

   a. **Pengelolaan State**

      GetX menawarkan dua pendekatan dalam mengelola state: Reactive State Management dan Simple State Management.

      1. **Reactive State Management**

         Pendekatan ini menggunakan kelas Rx untuk membuat data bersifat reaktif, sehingga UI akan otomatis diperbarui ketika data mengalami perubahan. Komponen utamanya meliputi:

         - **Rx**: Digunakan untuk membuat data menjadi reaktif, seperti RxInt atau RxString.
         - **.obs**: Sintaks singkat untuk menjadikan data reaktif.

      2. **Simple State Management**

         Pendekatan ini memanfaatkan widget seperti GetBuilder atau GetX untuk memperbarui UI secara manual tanpa perlu menjadikan data reaktif. Komponen utamanya adalah:

         - **GetBuilder**: Digunakan untuk memperbarui widget hanya ketika terjadi perubahan pada state tertentu.

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

      GetX mempermudah navigasi antar halaman tanpa harus menggunakan Navigator bawaan Flutter. Beberapa fungsi utama dalam manajemen route di GetX meliputi:

      - Get.to(): Digunakan untuk berpindah ke halaman baru.
      - Get.off(): Memungkinkan berpindah ke halaman baru sekaligus menutup

halaman sebelumnya.

- Get.back(): Untuk kembali ke halaman sebelumnya.
- Get.arguments: Berfungsi untuk mengirimkan data antar halaman.

c. Dependency Injection (DI)

GetX memiliki sistem dependency injection bawaan yang memungkinkan pengelolaan lifecycle objek dan membuatnya dapat diakses di seluruh aplikasi. Beberapa komponen utamanya adalah:

- Get.put(): Menyimpan instance controller yang dapat diakses secara global.
- Get.lazyPut(): Menyimpan instance dan hanya menginisialisasinya saat dibutuhkan.
- Get.find(): Mengambil instance yang sudah disimpan sebelumnya.

d. Middleware

Middleware dalam GetX berfungsi untuk mengatur akses ke halaman tertentu, seperti memverifikasi status login pengguna sebelum mereka dapat mengakses halaman tertentu. Komponen utama middleware adalah:

- GetMiddleware: Kelas yang digunakan untuk mendefinisikan middleware.
- redirect: Fungsi untuk mengarahkan pengguna ke halaman lain jika kondisi tertentu tidak terpenuhi.

e. Snackbar, Dialog, dan BottomSheet

GetX menyediakan utilitas built-in untuk menampilkan notifikasi, dialog, atau bottom sheet tanpa memerlukan BuildContext. Komponen utama yang digunakan adalah:

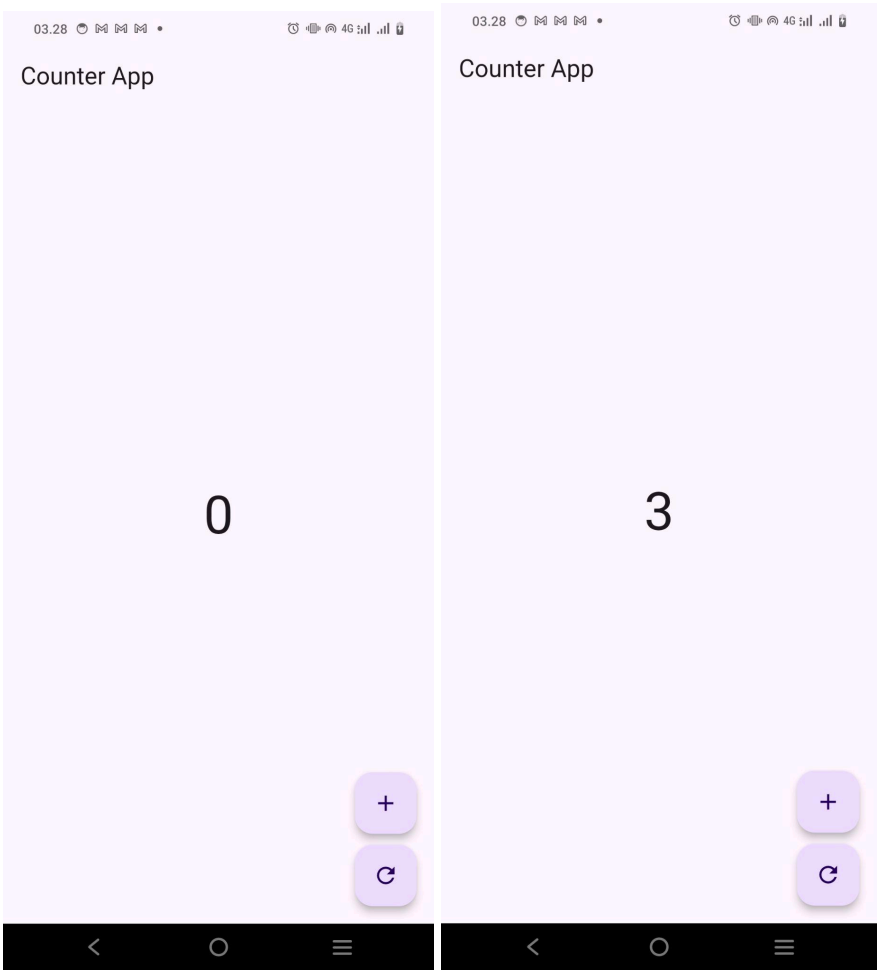
- Get.snackbar: Untuk menampilkan snackbar.
- Get.defaultDialog: Untuk menampilkan dialog.
- Get.bottomSheet: Untuk menampilkan bottom sheet.

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

```
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 counter = 0.obs; // Menggunakan .obs untuk membuat variabel
reaktif

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

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

class HomePage extends StatelessWidget {
  final CounterController controller = Get.put(CounterController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Counter App")),
      body: Center(
        child: Obx(() {
          // Menampilkan nilai counter yang sudah diperbarui
          return Text(
            "${controller.counter.value}", // Menggunakan controller
untuk mendapatkan nilai counter
            style: TextStyle(fontSize: 48),
          );
        }),
      ),
      floatingActionButton: Column(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          FloatingActionButton(
            onPressed: () {
              // Menambahkan nilai counter
              controller.increment();
```

```
            },
            child: Icon(Icons.add),
          ),
          SizedBox(height: 10),
          FloatingActionButton(
            onPressed: () {
              // Mereset nilai counter
              controller.reset();
            },
            child: Icon(Icons.refresh),
          ),
        ],
      ),
    );
  }
}

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

Output



Deskripsi Program

Aplikasi ini memiliki dua fitur utama: menambah dan mengatur ulang nilai counter. Fungsionalitasnya terdiri dari dua komponen utama, yaitu Controller dan UI.

Pada komponen pertama, **CounterController** adalah kelas yang berperan dalam mengelola nilai counter. Variabel counter dideklarasikan sebagai **RxInt** dengan menggunakan **.obs**, sehingga menjadi reaktif dan memungkinkan pembaruan UI secara otomatis setiap kali nilai counter berubah. Controller ini menyediakan dua fungsi utama: **increment()** untuk menambah nilai counter dan **reset()** untuk mengembalikan nilai counter ke nol.

Di sisi UI, terdapat kelas **HomePage** yang bertugas menampilkan antarmuka pengguna. Melalui widget **Scaffold**, sebuah **Text** digunakan untuk menampilkan nilai counter secara dinamis dengan bantuan **Obx()**, yang secara otomatis memperbarui tampilan saat nilai counter berubah. Selain itu, ada dua tombol aksi berupa **FloatingActionButton**, yang masing-masing digunakan untuk menambah nilai counter melalui fungsi **increment()** dan mengatur ulang nilai counter dengan fungsi **reset()**.

Aplikasi ini dijalankan melalui fungsi **main()**, yang memanfaatkan **MaterialApp** untuk memuat **HomePage** sebagai tampilan utamanya.