TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIII

NETWORKING

Disusun Oleh :

GANES GEMI PUTRA / 2311104075

S1SE-07-02

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

TUGAS PENDAHULUAN

SOAL

1. Apa yang dimaksud dengan state management pada Flutter?
2. Sebut dan jelaskan komponen-komponen yang ada di dalam GetX.
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 {
  // TODO: Tambahkan variabel untuk menyimpan nilai counter
  // TODO: Buat fungsi untuk menambah nilai counter

  // TODO: Buat fungsi untuk mereset nilai counter
}
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(() {
          // TODO: Lengkapi logika untuk menampilkan nilai
counter
          return Text(
            "0", // Ganti ini dengan nilai counter
style: TextStyle(fontSize: 48),
          );
        }),
),
      floatingActionButton: Column(
```

```
        mainAxisAlignment: MainAxisAlignment.end,
children: [
          FloatingActionButton(
onPressed: () {
              // TODO: Tambahkan logika untuk menambah nilai
counter             },
            child: Icon(Icons.add),
          ),
          SizedBox(height: 10),
FloatingActionButton(
onPressed: () {
              // TODO: Tambahkan logika untuk mereset nilai
counter             },
            child: Icon(Icons.refresh),
          ),
        ],
      ),
    );
  }
}
void main() {
runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
home: HomePage(),
  ));
}
```

Screenshoot Output

(lampirkan bukti screenshoot output dari sourcecode)

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri) **minimal 5 kalimat.**

JAWABAN :

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

State management adalah cara mengelola dan menyimpan data (state) yang dipakai oleh widget sehingga ketika data berubah, UI dapat bereaksi/merender ulang sesuai perubahan itu. Dalam Flutter ada banyak pendekatan (setState, InheritedWidget/Provider, BLoC, Redux, GetX, Riverpod, dsb.). Tujuannya: memisahkan logika/penyimpanan data dari tampilan, menjaga konsistensi data antar widget, mempermudah testabilitas dan skala aplikasi.

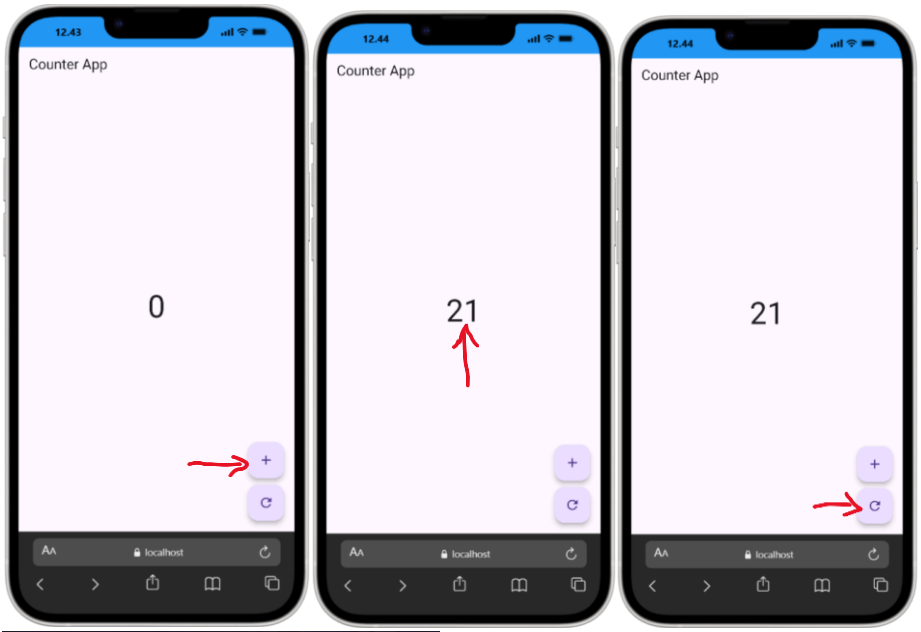
2) Komponen-komponen yang ada di dalam GetX (ringkas & jelas)

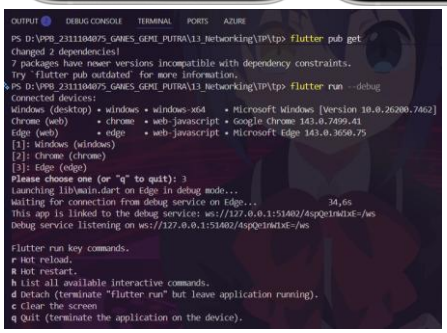
1. **GetxController**
   - Class untuk menaruh logika dan state. Biasanya dibuat turunannya untuk fitur tertentu (mis. CounterController).
2. **Reactive types (Rx)**
   - Tipe-tipe reaktif seperti RxInt, RxString, atau shorthand dengan .obs (contoh: 0.obs). Saat value berubah, widget yang mendengarkan otomatis rebuild.
3. **Obx / GetX widget**
   - Widget yang memantau nilai reaktif. Gunakan Obx(() => Text(...)) agar UI update otomatis saat Rx berubah.
4. **Dependency injection (Get.put, Get.lazyPut, Get.find)**
   - Cara mendaftarkan / mengambil controller atau service dari "container" GetX. Contoh Get.put(CounterController()).
5. **Routing (Get.to, Get.off, GetMaterialApp)**
   - Sistem routing ringan bawaan GetX (opsional). Untuk fitur ini biasanya gunakan GetMaterialApp.
6. **StateManager / Workers / Mixins lainnya**
   - Fitur tambahan untuk lifecycle, bindings, worker (ever, once, debounce) untuk observasi otomatis.

3) Lengkapi code — kode lengkap dan penjelasan :

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

/// Controller untuk mengelola state counter
class CounterController extends GetxController {
  RxInt counter = 0.obs;

  void increment() {
    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(() {
      return Text(
       "${controller.counter}",
       style: TextStyle(fontSize: 48),
      );
     }),
    ),
    floatingActionButton: Column(
     mainAxisAlignment: MainAxisAlignment.end,
     children: [
      FloatingActionButton(
       onPressed: () {
        controller.increment();
       },
       child: Icon(Icons.add),
      ),
      SizedBox(height: 10),
      FloatingActionButton(
       onPressed: () {
        controller.reset();
       },
       child: Icon(Icons.refresh),
      ),
     ],
    ),
   );
  }
}

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





Penjelasan Detail:

1. Import dan Dependencies

- import 'package:flutter/material.dart';: Mengimpor widget dasar Flutter.
- import 'package:get/get.dart';: Mengimpor GetX untuk state management.
- Ditambahkan dependency get: ^4.6.6 di pubspec.yaml untuk menggunakan GetX.

2. CounterController Class

- Extends GetxController: Kelas ini mengelola state aplikasi menggunakan GetX.
- RxInt counter = 0.obs;: Variabel reaktif yang menyimpan nilai counter. .obs membuatnya observable, sehingga UI akan update otomatis saat nilai berubah.
- void increment(): Fungsi untuk menambah nilai counter sebesar 1.
- void reset(): Fungsi untuk mereset nilai counter kembali ke 0.

3. HomePage Class

- StatelessWidget: Widget tanpa state internal, state dikelola oleh controller.
- final CounterController controller = Get.put(CounterController());: Membuat instance controller dan mendaftarkannya ke GetX dependency injection.
- **AppBar**: Menampilkan judul "Counter App".
- **Body**:
    - Center: Memposisikan konten di tengah layar.

    - Obx(() { ... }): Widget reaktif yang mendengarkan perubahan pada observable variables. Ketika counter berubah, UI akan rebuild otomatis.
    - Text("${controller.counter}", style: TextStyle(fontSize: 48)): Menampilkan nilai counter dalam teks besar.
- **FloatingActionButton**:
    - Dua tombol floating: satu untuk increment, satu untuk reset.
    - Tombol atas (add): Memanggil controller.increment() saat ditekan.
    - Tombol bawah (refresh): Memanggil controller.reset() saat ditekan.

4. main() Function

- runApp(MaterialApp(...)): Menjalankan aplikasi Flutter dengan Material Design.
- debugShowCheckedModeBanner: false: Menyembunyikan banner debug.
- home: HomePage(): Menetapkan HomePage sebagai halaman utama.

Output Aplikasi:

Aplikasi telah berhasil dijalankan dalam mode debug di browser Edge. Berikut adalah tampilan dan fungsionalitasnya:

- **Tampilan Awal**:
    - AppBar dengan judul "Counter App".
    - Teks besar "0" di tengah layar (nilai counter awal).
    - Dua floating action button di kanan bawah: ikon "+" dan ikon "refresh".
- **Interaksi**:
    - Tekan tombol "+" (add): Nilai counter bertambah 1 (0 → 1 → 2 → ...).
    - Tekan tombol "refresh": Nilai counter kembali ke 0.
    - UI update secara real-time tanpa perlu setState() karena menggunakan GetX reactive state management.
- **Debug Service**: Aplikasi terhubung ke Flutter debug service di ws://127.0.0.1:51402/... dan DevTools tersedia di http://127.0.0.1:9101/... untuk debugging dan profiling.

Aplikasi ini mendemonstrasikan penggunaan GetX untuk state management yang sederhana dan efisien dalam Flutter, di mana perubahan state secara otomatis memicu update UI.

➢ **Kesimpulan**

Pada tugas ini, telah berhasil diimplementasikan aplikasi **Counter App** menggunakan Flutter dengan **state management GetX**. GetX memungkinkan pengelolaan state yang lebih sederhana dan reaktif tanpa menggunakan setState(), sehingga perubahan nilai pada counter langsung memperbarui tampilan UI secara otomatis. Melalui pemanfaatan RxInt, Obx, dan Get.put(), aplikasi dapat bekerja secara efisien dengan memisahkan logika dari tampilan. Tugas ini juga menunjukkan bahwa penggunaan GetX membuat proses pengembangan lebih terstruktur, mudah dipahami, dan cocok untuk proyek dengan kebutuhan state yang dinamis. Secara keseluruhan, implementasi kode berhasil dijalankan dan fitur penambahan serta reset