PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIII

NETWORKING

Disusun Oleh:

Aorinka Anendya Chazanah / 2211104013

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

GUIDED

State management adalah proses mengelola state atau status aplikasi dalam Flutter, yang mencakup data atau informasi yang berubah sepanjang siklus hidup aplikasi. State ini memengaruhi tampilan antarmuka pengguna (UI), seperti input pengguna, data dari API, hingga status internal widget. Dalam aplikasi yang semakin kompleks, state sering kali perlu dibagikan atau digunakan di berbagai halaman untuk memastikan konsistensi data.

Karena Flutter bersifat deklaratif, antarmuka pengguna (UI) dibangun berdasarkan state saat ini. Dengan state management, semua state dari berbagai UI control dapat disentralisasi sehingga aliran data dalam aplikasi menjadi lebih terkendali. Hal ini penting, terutama pada aplikasi Flutter yang sering kali melibatkan banyak widget yang saling berhubungan.

Manfaat utama pengelolaan state yang baik meliputi:

- **Sinkronisasi UI dan Data**: Memastikan bahwa UI selalu mencerminkan data terkini.
- **Kode yang Terorganisasi**: Mempermudah pengembangan dan pemeliharaan aplikasi.
- **Pengurangan Bug**: Dengan pengelolaan state yang benar, kemungkinan bug akibat data yang tidak konsisten dapat diminimalkan.

Jenis State dalam Flutter

1. **Ephemeral State (State Lokal)**
   Ephemeral state adalah state yang hanya relevan untuk widget tertentu dan tidak digunakan oleh widget lain. Contohnya termasuk state untuk *TextField* atau *Checkbox*. Pengelolaan ephemeral state biasanya dilakukan menggunakan *StatefulWidget*. Pendekatan ini terbagi menjadi dua: *StatefulWidget* untuk state lokal dan *InheritedWidget* untuk berbagi state antar widget.
2. **App State (State Global)**
   App state mengacu pada state yang digunakan di berbagai widget dalam aplikasi, seperti informasi pengguna yang sedang login, data keranjang belanja, atau tema aplikasi. Untuk mengelola app state, diperlukan pendekatan state management yang lebih kompleks, biasanya dengan bantuan paket atau pustaka Flutter. Berikut adalah beberapa di antaranya:

- **Provider**
  Library resmi dari tim Flutter yang memanfaatkan *InheritedWidget* untuk mengelola state dengan cara yang lebih sederhana dan efisien.
- **BloC/Cubit**
  Pendekatan berbasis *stream* yang memisahkan *business logic* dari UI. BloC sangat cocok untuk aplikasi besar dan kompleks karena mendukung pengelolaan logika bisnis secara terpisah.
- **Riverpod**
  Framework modern yang dirancang sebagai alternatif dari Provider. Riverpod menawarkan fleksibilitas lebih dan mengatasi keterbatasan yang dimiliki Provider.
- **GetX**
  Framework serbaguna untuk Flutter yang menyediakan solusi lengkap untuk *state management*, *routing*, dan *dependency injection*. GetX dikenal karena kemampuannya untuk meminimalkan kode boilerplate, meningkatkan efisiensi, serta mendukung aplikasi dengan kebutuhan reaktivitas tinggi.

Berikut ini cara instalasi GetX:

1) **Tambahkan GetX ke dalam proyek Flutter melalui pubspec.yaml :**

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  get: ^4.6.6
```

2) **Konfigurasi dasar**

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view/detail_page.dart';
import 'package:modul13/view/my_home_page.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
```

3) State Management dengan GetX

a) Membuat Controller class controller untuk mengelola state. Misalnya, untuk counter sederhana:

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

class CounterController extends GetxController {
  var counter = 0.obs;

//Fungsi untuk menambah
  void incrementCounter() {
    counter++;
  }
//Fungsi untuk mengurangi
  void decrementCounter() {
    counter--;
  }
```

b) Menggunakan Controller di UI

- Tambahkan controller ke dalam widget menggunakan Get.put() untuk dependency injection.
- Gunakan Obx untuk memantau perubahan state.

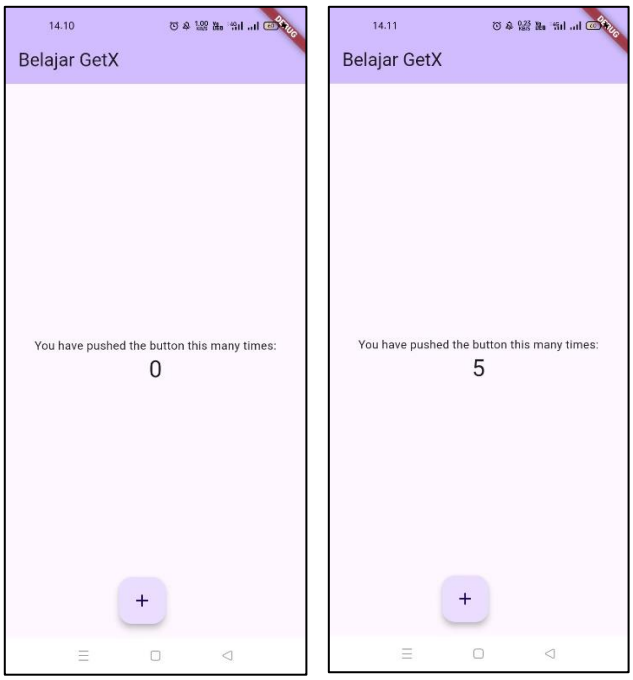
```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import
'package:modul13/view_model/controller.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ),
      body: Center(
        child: Obx( Text (
          'Counter: $ {controller.count}',
```

```
        style: TextStyle(fontSize: 25),
       )),
      ),
      floatingActionButton: Row(
        mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
        children: [
          FloatingActionButton(
            onPressed:
controller.incrementCounter,
            tooltip: 'Increment',
            child: const Icon(Icons.add),
          ),
FloatingActionButton(
            onPressed:
controller.decrementCounter,
            tooltip: 'Decrement',
            child: const Icon(Icons.remove),
          ),
```

Output program icrement



Output program decrement

4) Routing dengan GetX

a. Definisikan Route Gunakan GetPage pada main.dart untuk mendefinisikan rute aplikasi:

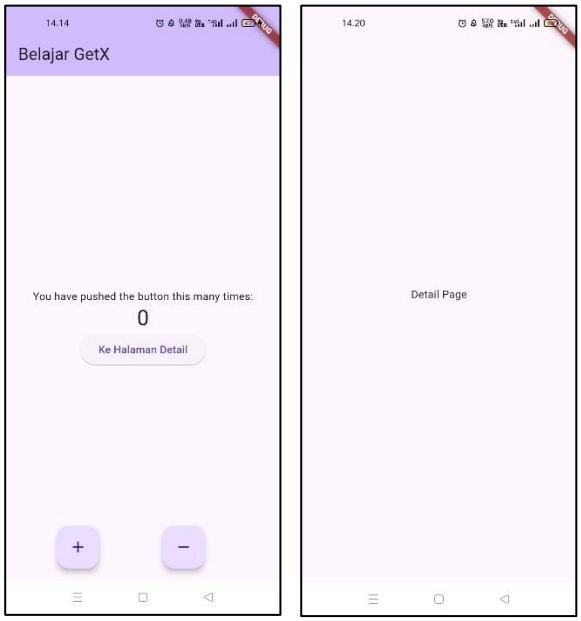
```dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view/detail_page.dart';
import 'package:modul13/view/my_home_page.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      initialRoute: '/',
      getPages: [
        GetPage(name: '/', page: () =>
MyHomePage(title: 'Belajar GetX')),
        GetPage(name: '/detail', page: () =>
DetailPage()),
      ],
    );
  }
}
```

Output program



b. Navigasi
   - Get.to(): Navigasi ke halaman baru.
   - Get.back(): Kembali ke halaman sebelumnya.
   - Get.off(): Menghapus semua halaman sebelumnya.
   - Get.offAll() : Menghapus semua halaman dalam stack.

5) Dependency Injection dengan GetX

1. Injeksi Sederhana
   Gunakan Get.put() untuk membuat instance controller yang tersedia di mana saja:

```
final CounterController controller =
Get.put(CounterController());
```

2. Lazy Loading
   Gunakan Get.lazyPut() jika ingin membuat instance hanya saat dibutuhkan:

```
Get.lazyPut(() => CounterController());
```

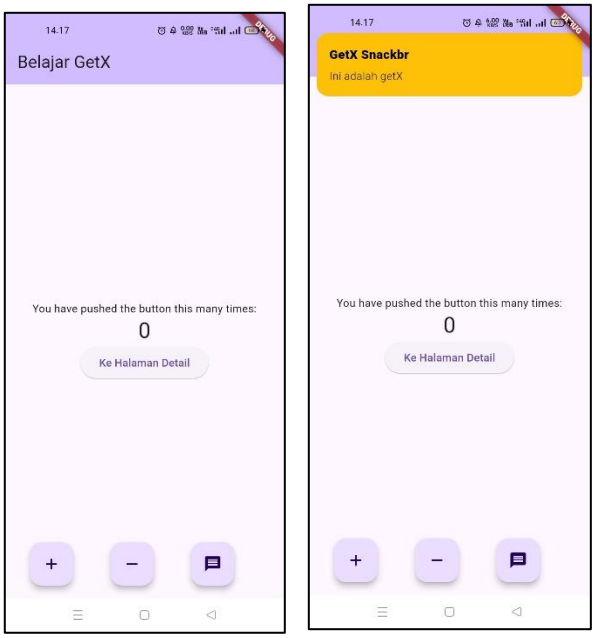
3. Mengambil Instance
   Ambil instance di mana saja dalam aplikasi:

```
final CounterController controller = Get.find();
```

6) Snackbar

```
void getSnackBar() {
    Get.snackbar(
      'GetX Snackbr',
      'Ini adalah getX',
      backgroundColor: Colors.amber,
      colorText: Colors.black,
    );
```

Output program



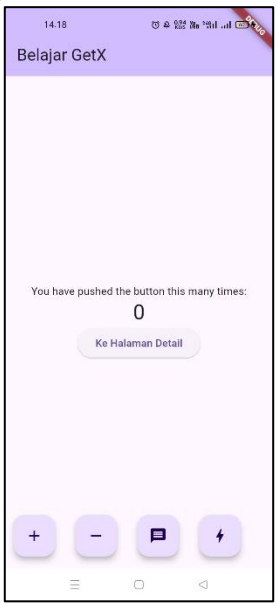
7) Dialog

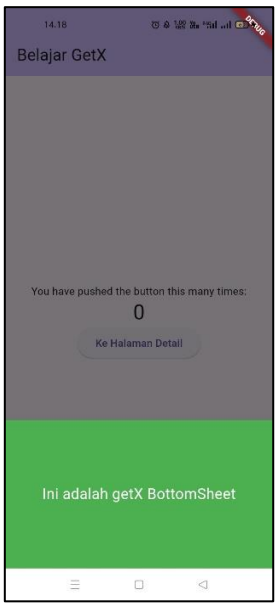
```
Get.defaultDialog(
   title: 'Dialog Title',
   middleText: 'This is a dialog',
);
```

8) BottomSheet

```
void getbottomshet() {
    Get.bottomSheet(
      Container(
        height: 200,
        color: Colors.green,
        child: const Center(
          child: Text(
            'Ini adalah getX BottomSheet',
            style: TextStyle(
              color: Colors.white,
              fontSize: 20,
            ),
          ),
        ),
      ),
    );
  }
```

Output program





UNGUIDED

SOAL

Buatlah Aplikasi Catatan Sederhana menggunakan GetX, dengan ketentuan sebagai berikut:

1. Halaman utama atau Homepage untuk menampilkan daftar catatan yang telah ditambahkan. Setiap catatan terdiri dari judul dan deskripsi singkat, serta terdapat tombol untuk menghapus catatan dari daftar.
2. Halaman kedua untuk menambah catatan baru, berisi : form untuk memasukkan judul dan deskripsi catatan, serta tombol untuk menyimpan catatan ke daftar (Homepage).
3. Menggunakan getx controller.
4. Menggunakan getx routing untuk navigasi halaman.

JAWABAN

File main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      title: 'Simple Notes App',
      initialRoute: '/',
      getPages: [
        GetPage(name: '/', page: () => const HomePage()),
        GetPage(name: '/add', page: () => const AddNotePage()),
      ],
    );
  }
```

```
}
```

File notes_controller.dart

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

class NotesController extends GetxController {
  var notes = <Map<String, String>>[].obs;

  void addNote(String title, String description) {
    notes.add({"title": title, "description": description});
  }

  void deleteNoteAt(int index) {
    notes.removeAt(index);
  }
}
```

File homepage.dart

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

class HomePage extends StatelessWidget {
  const HomePage({super.key});

  @override
  Widget build(BuildContext context) {
    final NotesController notesController =
Get.put(NotesController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('Notes'),
      ),
      body: Obx(() {
        return ListView.builder(
          itemCount: notesController.notes.length,
          itemBuilder: (context, index) {
            final note = notesController.notes[index];
            return ListTile(
              title: Text(note['title']!),
              subtitle: Text(note['description']!),
```

```
              trailing: IconButton(
                icon: const Icon(Icons.delete, color: Colors.red),
                onPressed: () {
                  notesController.deleteNoteAt(index);
                },
              ),
            );
          },
        );
      }),
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.add),
        onPressed: () {
          Get.toNamed('/add');
        },
      ),
    );
  }
}
```

File add_note_page.dart

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

class AddNotePage extends StatelessWidget {
  const AddNotePage({super.key});

  @override
  Widget build(BuildContext context) {
    final NotesController notesController = Get.find();
    final titleController = TextEditingController();
    final descriptionController = TextEditingController();

    return Scaffold(
      appBar: AppBar(
        title: const Text('Add Note'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: titleController,
```

```
              decoration: const InputDecoration(labelText:
'Title'),
            ),
            const SizedBox(height: 16),
            TextField(
              controller: descriptionController,
              decoration: const InputDecoration(labelText:
'Description'),
            ),
            const SizedBox(height: 32),
            ElevatedButton(
              onPressed: () {
                notesController.addNote(
                  titleController.text,
                  descriptionController.text,
                );
                Get.back();
              },
              child: const Text('Save'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Penjelasan program:

1. **main.dart**

   File main.dart adalah entry point aplikasi Flutter ini. File ini menginisialisasi aplikasi menggunakan GetMaterialApp untuk mendukung navigasi dan state management dari GetX. Di sini, rute-rute aplikasi didefinisikan menggunakan getPages, dengan halaman utama diarahkan ke HomePage dan halaman kedua diarahkan ke AddNotePage. File ini bertanggung jawab untuk mengatur navigasi dan struktur dasar aplikasi.

2. **home_page.dart**

   File ini berisi implementasi halaman utama yang menampilkan daftar catatan. Menggunakan controller NoteController yang disediakan oleh GetX, data catatan disinkronkan dengan UI melalui widget Obx. Halaman

ini memiliki daftar catatan yang dapat dihapus menggunakan tombol delete, serta tombol "+" untuk navigasi ke halaman tambah catatan. Semua elemen UI diatur untuk memberikan pengalaman pengguna yang intuitif dan responsif.

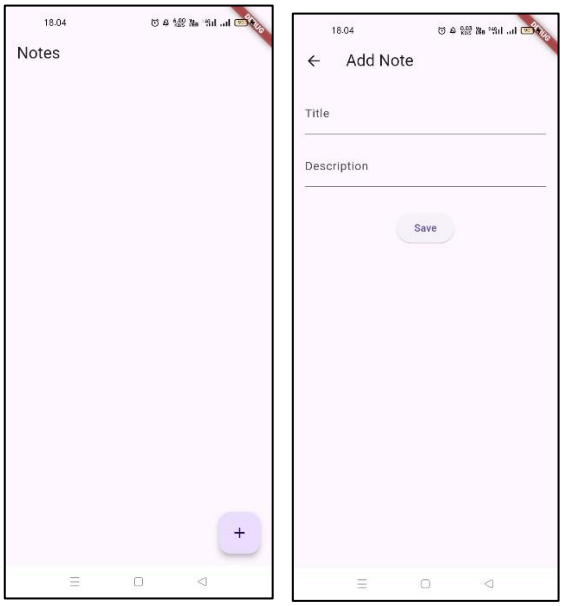
3. **add_note_page.dart**
   File ini adalah halaman untuk menambahkan catatan baru. Halaman ini terdiri dari form input untuk judul dan deskripsi catatan, yang datanya dikelola melalui controller. Tombol simpan akan menambahkan catatan baru ke daftar dengan memanfaatkan fungsi dalam controller, lalu pengguna diarahkan kembali ke HomePage. Dengan navigasi berbasis GetX, transisi antarhalaman menjadi sederhana dan efisien.

4. **note_controller.dart**
   File ini berisi controller untuk mengelola state daftar catatan menggunakan GetX. Controller ini menyimpan daftar catatan dalam variabel reaktif dan menyediakan fungsi untuk menambahkan dan menghapus catatan. File ini menjadi pusat logika bisnis aplikasi, memastikan data dikelola secara efisien dan dapat digunakan oleh berbagai widget di aplikasi.

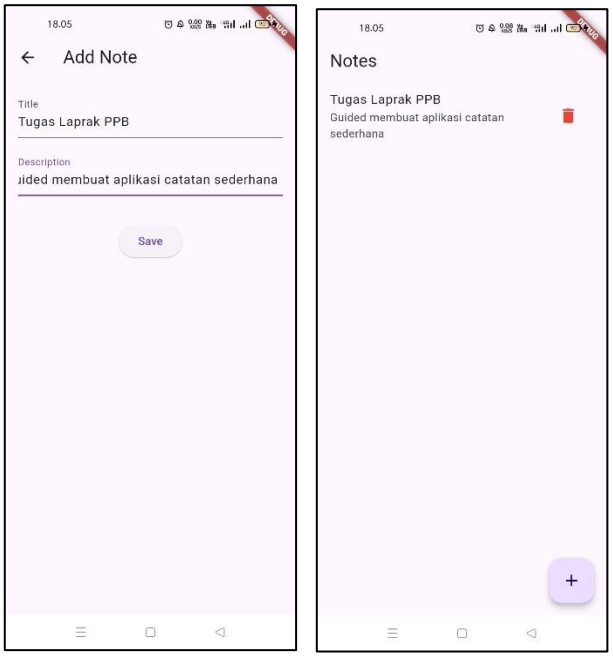
Secara keseluruhan, program ini memanfaatkan GetX untuk state management, dependency injection, dan navigasi.

Output program

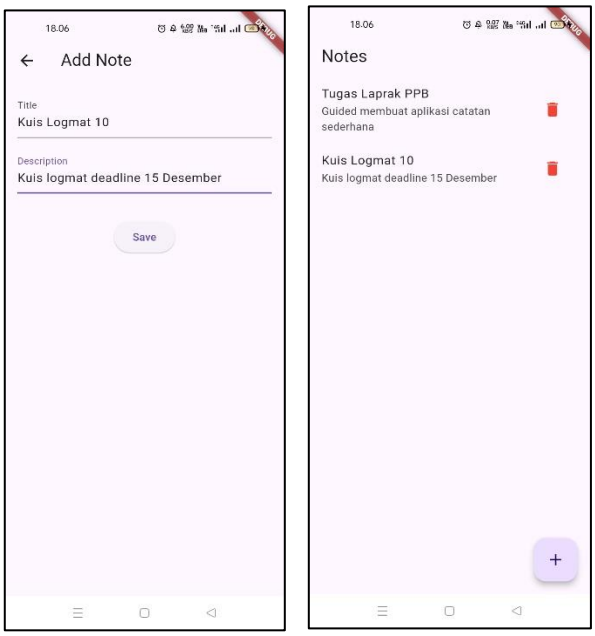
Tampilan awal ketika aplikasi dibuka terdapat floating button (+) di pojok kanan bawah. Apabila button di-klik maka muncul tampilan kolom untuk menambahkan data catatan berupa kolom “title” dan “description” serta button “Save” untuk menyimpan data.



Setelah itu kita akan mencoba menambah data pertama. Jika data berhasil disimpan maka akan muncul tampilan pada homepage seperti berikut.



Kemudian kita akan mencoba menambah data ke dua.



Setelah memiliki dua data, kita akan mencoba menghapus data yang pertama dengan memilih button “Hapus”. Berikut adalah tampilan ketika data telah terhapus, pada tampilan di bawah hanya tersisa 1 data saja.