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

GUIDED

Manajemen state dalam Flutter adalah proses pengelolaan status atau kondisi aplikasi, yang mencakup data atau informasi yang dapat berubah selama aplikasi berjalan. State ini berpengaruh langsung pada tampilan antarmuka pengguna (UI), seperti respons terhadap input pengguna, pengambilan data dari API, atau kondisi internal widget. Dalam aplikasi yang lebih kompleks, state sering kali perlu dibagikan antar halaman untuk menjaga konsistensi data.

Karena Flutter menggunakan pendekatan deklaratif, antarmuka pengguna (UI) dibangun berdasarkan state terkini. Dengan menerapkan manajemen state, seluruh state dari berbagai elemen kontrol UI dapat dipusatkan, sehingga aliran data dalam aplikasi lebih mudah diatur. Hal ini sangat penting, terutama untuk aplikasi Flutter yang biasanya terdiri dari banyak widget yang saling berinteraksi.

Manfaat utama dari pengelolaan state yang efektif mencakup hal-hal berikut:

1. Sinkronisasi Antara UI dan Data: Memastikan antarmuka pengguna selalu menampilkan data terbaru..
2. Kode Lebih Tertata: Memudahkan proses pengembangan dan perawatan aplikasi.
3. Minimalkan Bug: Dengan manajemen state yang tepat, risiko terjadinya bug akibat data yang tidak konsisten dapat dikurangi.

Jenis State dalam Flutter

1. **Ephemeral State (State Lokal)**
   Ephemeral state merupakan jenis state yang hanya berlaku pada widget tertentu dan tidak dibutuhkan oleh widget lain. Contohnya adalah state yang mengatur TextField atau Checkbox. Pengelolaan ephemeral state biasanya dilakukan dengan menggunakan StatefulWidget. Terdapat dua pendekatan dalam mengelolanya: menggunakan StatefulWidget untuk state lokal dan InheritedWidget untuk berbagi state antar widget.
2. **App State (State Global)**
   App state merujuk pada state yang dibutuhkan oleh banyak widget di seluruh aplikasi, seperti data pengguna yang sedang login, informasi keranjang belanja, atau pengaturan tema aplikasi. Untuk mengelola app state, diperlukan metode manajemen state yang lebih kompleks, biasanya dengan menggunakan paket atau pustaka Flutter. Beberapa contoh yang umum digunakan adalah:

- **Provider**

  Sebuah pustaka resmi dari tim Flutter yang memanfaatkan **InheritedWidget** untuk menyederhanakan dan meningkatkan efisiensi dalam pengelolaan state.
- **BloC/Cubit**

  Pendekatan berbasis stream yang memisahkan logika bisnis dari UI. Cocok untuk aplikasi besar dan kompleks karena memungkinkan pengelolaan logika secara terpisah dengan struktur yang lebih terorganisasi.
- **Riverpod**

  Framework modern yang dirancang sebagai alternatif dari Provider. Riverpod menawarkan fleksibilitas yang lebih baik dan mampu mengatasi beberapa keterbatasan yang ada pada Provider.
- **GetX**

  Framework multifungsi untuk Flutter yang mencakup solusi lengkap untuk state management, routing, dan dependency injection. GetX terkenal karena kemampuannya mengurangi kode boilerplate, meningkatkan efisiensi, dan 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),
          ),
```

4) Routing dengan GetX

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

```
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());
```

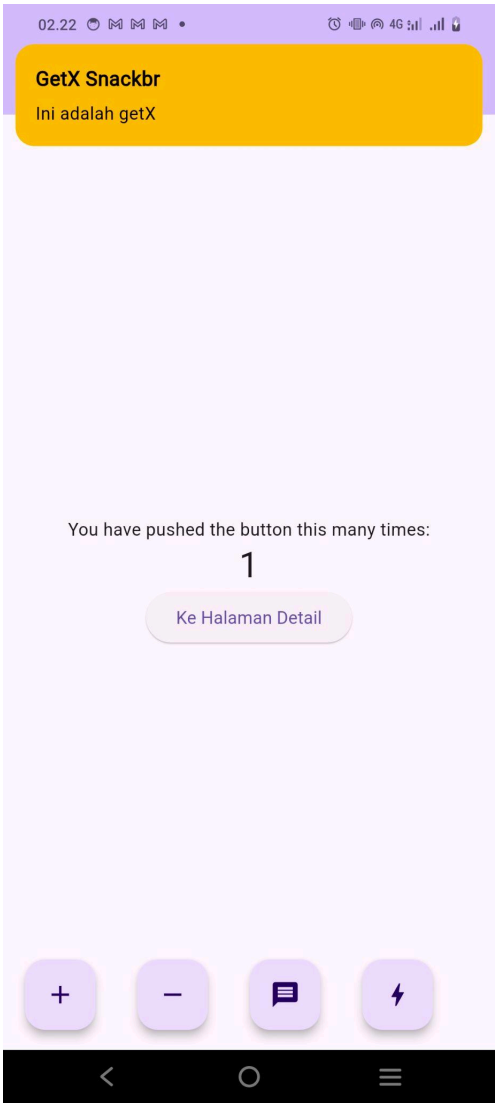
3. Mengambil Instance
   Ambil instance di mana saja dalam aplikasi:

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

6) Snackbar

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

Output program



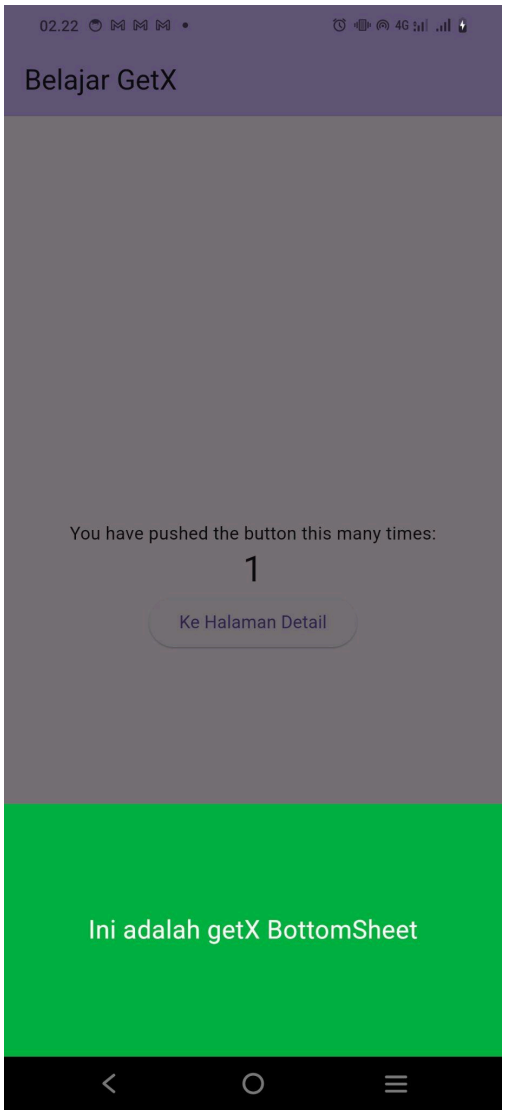
7) Dialog

```
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.

JAWAB

File main.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:

- main.dart

  File *main.dart* berfungsi sebagai titik awal aplikasi Flutter ini. Aplikasi diinisialisasi menggunakan GetMaterialApp untuk mendukung navigasi dan manajemen state dengan GetX. Rute-rute aplikasi didefinisikan melalui getPages, di mana halaman utama diarahkan ke HomePage, sedangkan halaman lainnya menuju ke AddNotePage. File ini bertugas mengatur navigasi dan kerangka dasar aplikasi.

- home_page.dart

  File ini mengimplementasikan halaman utama yang menampilkan daftar catatan. Dengan menggunakan NoteController dari GetX, data catatan dikelola secara sinkron melalui widget Obx. Halaman ini memungkinkan pengguna untuk menghapus catatan menggunakan tombol hapus dan menambahkan catatan baru dengan tombol "+", yang akan mengarahkan ke halaman tambah catatan. Seluruh elemen UI dirancang agar mudah digunakan dan responsif.

- add_note_page.dart

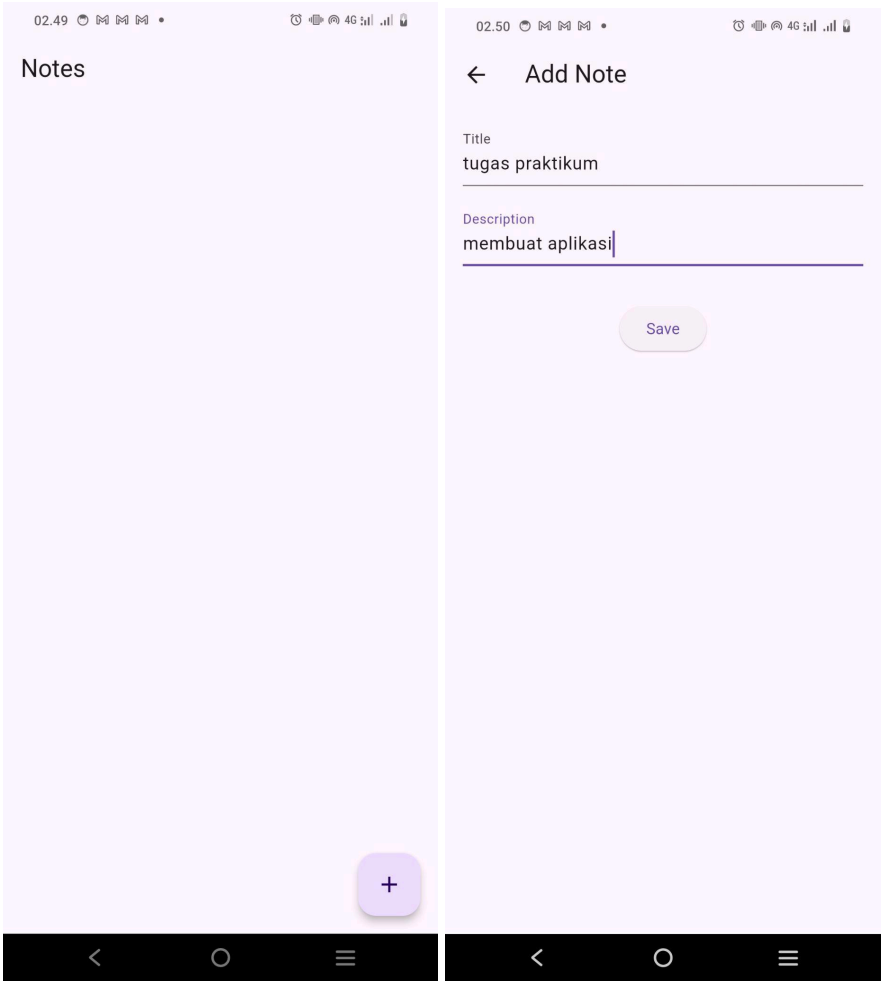
  Halaman ini digunakan untuk menambahkan catatan baru. Terdapat form input untuk memasukkan judul dan deskripsi catatan, di mana data tersebut dikelola melalui controller. Tombol simpan akan menambahkan catatan baru ke daftar menggunakan fungsi di dalam controller, kemudian pengguna akan diarahkan kembali ke HomePage. Dengan bantuan navigasi berbasis GetX, perpindahan antarhalaman menjadi lebih mudah dan efisien.
- note_controller.dart

  File ini adalah *controller* yang mengelola state daftar catatan menggunakan GetX. State catatan disimpan dalam variabel reaktif, dan fungsi-fungsi seperti menambahkan serta menghapus catatan disediakan di sini. File ini menjadi pusat logika aplikasi, memastikan data dapat dikelola secara efisien dan diakses oleh berbagai widget.

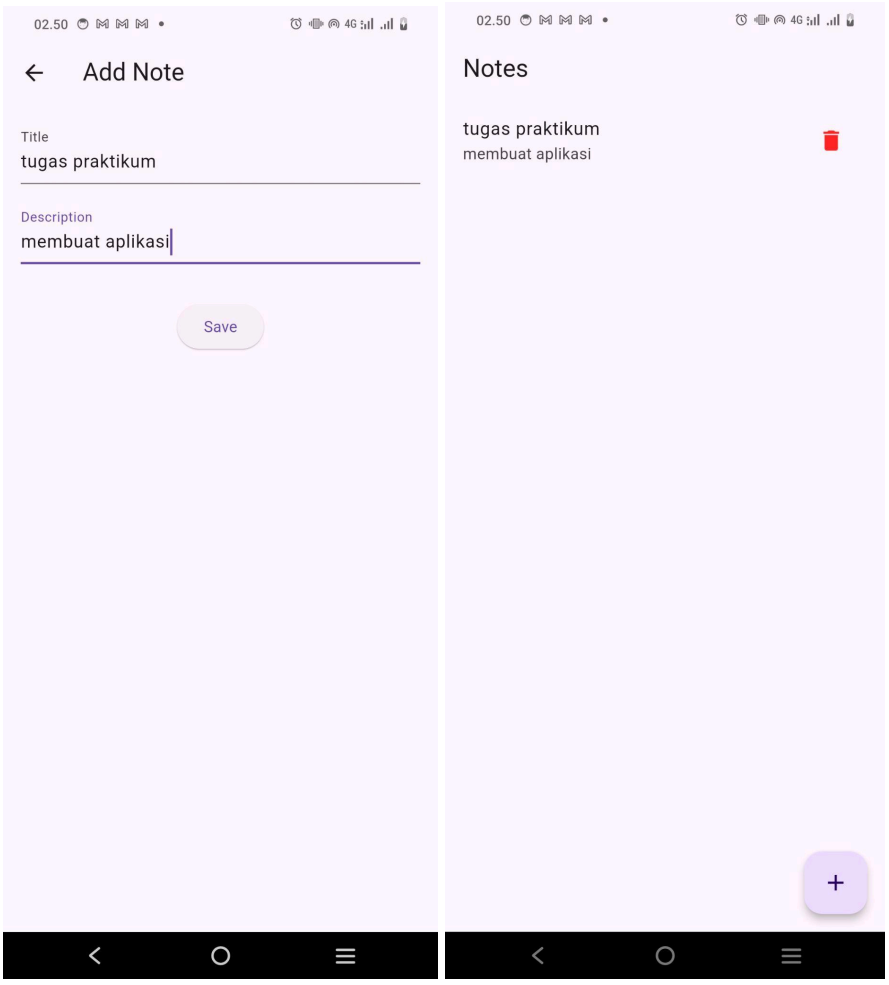
Program ini secara keseluruhan memanfaatkan GetX untuk manajemen state, *dependency injection*, dan navigasi, sehingga menciptakan aplikasi yang terstruktur dan mudah dikembangkan.

Output program

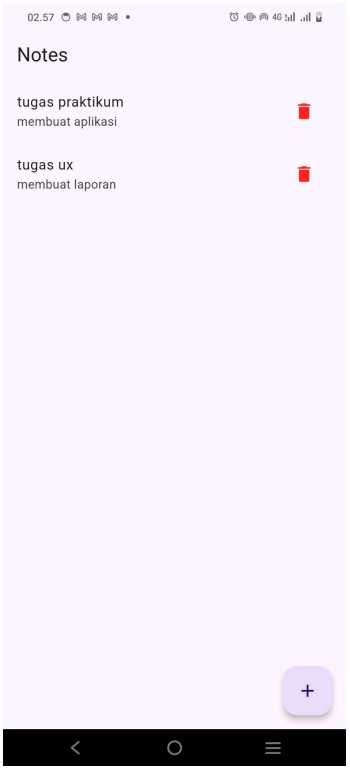
Ketika aplikasi pertama kali dibuka, terdapat tombol melayang (+) di sudut kanan bawah. Saat tombol tersebut diklik, akan muncul form untuk menambahkan data catatan yang terdiri dari kolom "title" dan "description," serta tombol "Save" untuk menyimpan data.



Kemudian, kita akan mencoba memasukkan data pertama. Apabila data berhasil disimpan, halaman utama akan menampilkan perubahan seperti berikut.



Lalu menambah data ke dua.



Setelah terdapat dua data, kita akan menghapus data pertama dengan menekan tombol "Hapus". Berikut adalah tampilan setelah data berhasil dihapus, di mana hanya tersisa satu data di layar.