

Version 1.0

Juli, 2024



# PEMROGRAMAN MOBILE

Modul 1 Materi - User Interface, Architecture,  
and State Management

**Tim Penyusun:**

Ali Sofyan, S.kom, M.kom

Ahmad Naufal

Marchanda Balqis

LABORATORIUM INFORMATIKA  
Universitas Muhammadiyah Malang

## PEMROGRAMAN MOBILE

### PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Flutter dan Dart
2. State Management dengan GetX
3. Architecture pada Flutter

### TUJUAN

1. Mampu memahami serta mengimplementasikan State Management dengan Framework Flutter
2. Mampu memahami Architecture Framework Flutter

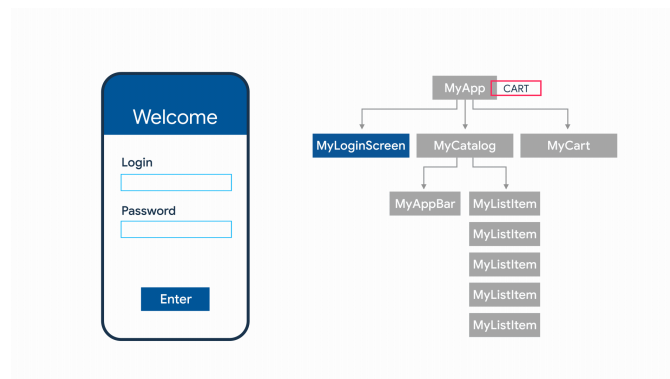
### PERSIAPAN HARDWARE & SOFTWARE

1. Laptop atau PC
2. IDE Android Studio atau Visual Studio Code
3. Flutter SDK: [Flutter 3.16.9](#)

### MATERI POKOK

#### State Management

[State management](#) adalah cara kita mengelola dan mengatur data (atau "state") aplikasi ini, sehingga perubahan data tersebut dapat terlihat di UI aplikasi dengan tepat dan efisien. State adalah data yang bisa berubah seiring waktu.



[Ilustrasi state management](#)

## GetX

GetX adalah salah satu dari beberapa [list state management](#) yang dimiliki Flutter. GetX dikenal sebagai state management sederhana dan powerful untuk mengkombinasikan injeksi dependensi, dan route management secara praktis dan efisien. GetX memiliki 3 prinsip dasar yang menjadi prioritas untuk semua resource di dalamnya, yaitu:

### 1. Performa

GetX fokus pada performa dan konsumsi resource minimum. GetX tidak menggunakan Stream atau ChangeNotifier.

### 2. Produktifitas

GetX menggunakan sintaks yang mudah dan nyaman sehingga banyak hal dapat dilakukan dengan lebih mudah, menghemat waktu development, dan memberikan performa maksimal pada aplikasi.

Umumnya, developer akan selalu berhubungan dengan penghapusan controller dari memori. Dengan GetX, ini tidak diperlukan, karena resource akan dihapus dari memori secara default ketika tidak digunakan. Jika anda ingin menyimpannya kedalam memori, cukup dengan mendeklarasikan secara eksplisit "permanent: true" pada dependensi. Sehingga dapat menghemat waktu dan mengurangi resiko penggunaan dependensi yang tidak diperlukan dalam memori. Pemuatan dependensi juga bersifat "lazy" secara default.

### 3. Organisasi

GetX memungkinkan pemisahan View, Presentation Logic, Business Logic, Dependency Injection, dan Navigasi. Developer tidak perlu lagi konteks untuk berpindah antar halaman sehingga dalam hal ini aplikasi tidak lagi bergantung pada widget tree (visualisasi).

Developer tidak perlu konteks untuk mengakses controller/bloc melalui InheritedWidget, sehingga presentation logic dan business logic benar benar terpisahkan dari lapisan visual. Kelas Controller/Model/Bloc tidak perlu menginjeksi ke dalam widget tree melalui

multiprovider, untuk hal ini GetX menggunakan fitur dependency injection nya sendiri yang terpisah dari View secara total.

Dengan GetX kode akan bersih secara default sehingga setiap fitur aplikasi dapat mudah dicari. Selain untuk memfasilitasi maintenance, hal ini dapat membuat pembagian modul pada Flutter menjadi sangat mungkin.



[GetX](#) adalah cara termudah, praktis, dan scalable untuk membangun aplikasi dengan performa tinggi menggunakan Flutter SDK, dengan ekosistem besar di sekelilingnya yang bekerjasama secara sempurna, mudah dipahami untuk pemula, dan akurat untuk ahli. Aman, stabil, up-to-date, dan menawarkan banyak cakupan build-in API yang tidak tersedia di dalam default Flutter SDK.

GetX tidak "bloated". Dirinya memiliki banyak fitur yang memungkinkan anda memulai programming tanpa mengkhawatirkan apapun, namun setiap fiturnya terletak di dalam kontainer terpisah, dan hanya dimulai setelah digunakan. Jika anda hanya menggunakan State Management, hanya State Management yang akan di-compile. Jika anda hanya menggunakan routes, state management tidak akan di-compile.

GetX memiliki ekosistem yang besar, komunitas yang juga besar, banyak kolaborator, dan akan di maintenance selama Flutter ada. GetX juga mampu berjalan dengan kode yang sama di Android, iOS, Web, Mac, Linux, Windows, dan server anda. Juga memungkinkan untuk me-reuse kode yang dibuat di front end ke backend dengan Get Server.

## Installation

Untuk menginstal GetX cukup dengan menambahkan GetX plugin pada file `pubspec.yaml`.

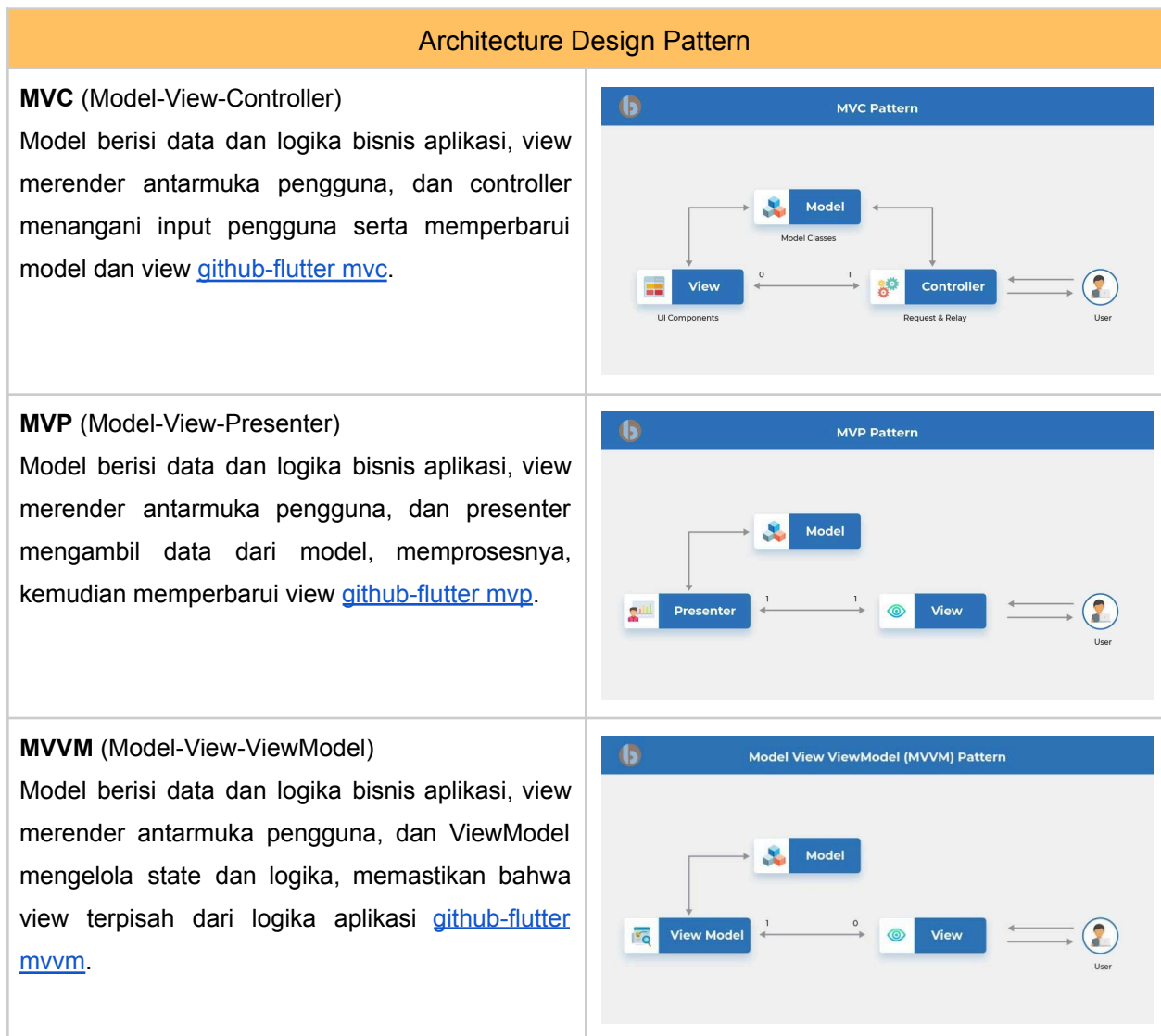
4. Buka file `pubspec.yaml` pada project
  -  nama\_project
  -  pubspec.yaml
5. Tambahkan dependensi GetX, lalu save

```
dependencies:  
  flutter:  
    sdk: flutter  
  
  get: 4.6.5
```

6. Buka terminal lalu jalankan perintah `flutter pub get` untuk menginstall library
7. Setelah terinstall, maka GetX sudah dapat digunakan

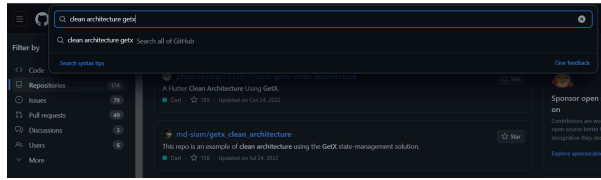
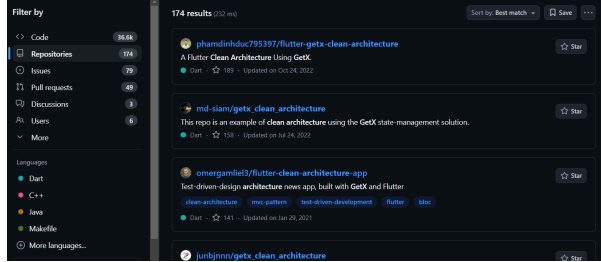
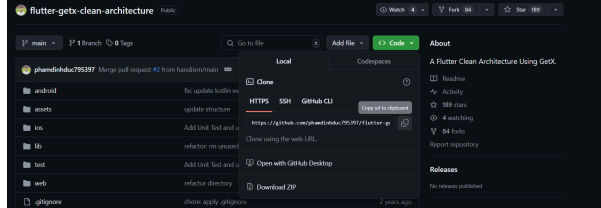
## Architecture

Sebelum menggunakan GetX, alangkah baiknya untuk memahami architecture terlebih dahulu. Architecture membantu untuk mengatur dan merancang kode serta komponen dalam aplikasi Flutter agar aplikasi menjadi lebih mudah dikelola, scalable, dan maintainable [bacancytechnology-architecture flutter](#) [geeksforgeeks-architecture flutter](#).



Saat ini sudah banyak architecture yang dapat diakses secara public. Terdapat beberapa cara untuk menggunakan sebuah architecture, pada modul ini akan mencontohkan 2 cara yaitu Pull Github dan Get CLI.

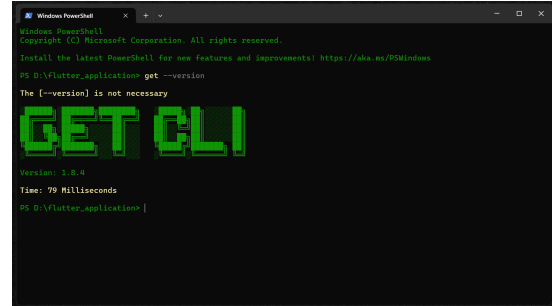
## Pull Github

<ol style="list-style-type: none"> <li>1. Buka <a href="https://github.com">github.com</a></li> <li>2. Cari pattern yang akan digunakan</li> </ol>	 <p>The screenshot shows the GitHub search interface with the query 'clean architecture getx'. The results list several repositories, including 'A Flutter Clean Architecture Using GetX' by phamdinhdac795397, 'mdl-siam/getx_clean_architecture', and 'omeramili3/flutter-clean-architecture-app'.</p>
<ol style="list-style-type: none"> <li>3. Gunakan filter bahasa, pilih Dart</li> <li>4. Pilih salah satu pattern</li> </ol>	 <p>The screenshot shows the search results filtered by the Dart language. The results list several repositories, including 'A Flutter Clean Architecture Using GetX' by phamdinhdac795397, 'mdl-siam/getx_clean_architecture', and 'omeramili3/flutter-clean-architecture-app'.</p>
<ol style="list-style-type: none"> <li>5. Copy url repository github dari pattern yang akan digunakan</li> </ol>	 <p>The screenshot shows the GitHub repository page for 'flutter-getx-clean-architecture'. The 'Clone' button is highlighted, and the URL 'https://github.com/PhamDinhDuc795397/flutter-getx-clean-architecture.git' is visible.</p>
<ol style="list-style-type: none"> <li>6. Buka terminal, arahkan ke direktori untuk menyimpan project tersebut. Lalu cloning url repository github yang sudah dicopy.</li> </ol>	<p>Contoh:</p> <pre><code>`git clone https://github.com/phamdinhdac795397/flutter-getx-clean-architecture.git`</code></pre>

## Get CLI

[Get CLI](#) adalah command line interface yang dapat digunakan untuk membuat seluruh infrastrukturGetX dengan menggunakan template. Get CLI juga dapat menjadi solusi untuk mengurangi waktu setup controller, view, dan bindings.

1. Pastikan Flutter sudah terinstall pada sistem operasi
2. Install Get CLI melalui terminal  
`Flutter pub global activate get\_cli`
3. Verifikasi instalasi Get CLI  
`get --version`



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Flutter_applications> get --version

The [--version] is not necessary

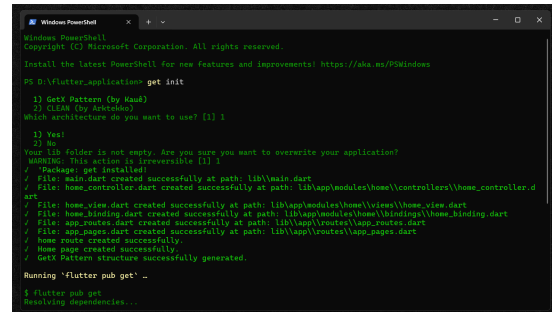
GET CLI

Version: 1.2.4
Time: 79 Milliseconds
PS D:\Flutter_applications>
  
```

## Overwrite Project

Perlu diperhatikan proses ini akan menimpa (**overwrite**) folder lib atau **mengganti keseluruhan** isi dari folder lib sebelumnya.

1. Gunakan command line untuk navigasi ke dalam folder project
2. Jalankan perintah  
`get init`
3. Pilih architecture **GetX Pattern**
4. Lalu overwrite
5. Tunggu hingga proses selesai
6. Buka project yang telah di-overwrite pada IDE atau Text Editor



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Flutter_applications> get init

[?] GetX Pattern (by Kaud)
[?] (Ctrl Q to quit)
Which architecture do you want to use? [?] 1
[?] Yes
[?] No
Your lib folder is not empty. Are you sure you want to overwrite your application?
Warning: This action is irreversible [?] 1
/ Package: get installed
/ File: main.dart created successfully at path: lib\main.dart
/ File: home_controller.dart created successfully at path: lib\app\modules\home\controllers\home_controller.d
art
/ File: home_view.dart created successfully at path: lib\app\modules\home\views\home_view.dart
/ File: home_binding.dart created successfully at path: lib\app\modules\home\bindings\home_binding.dart
/ File: app_router.dart created successfully at path: lib\app\routes\app_router.dart
/ File: app_pages.dart created successfully at path: lib\app\routes\app_pages.dart
/ Home page created successfully
/ Home page created successfully
/ GetX Pattern structure successfully generated.

Running "flutter pub get" ..
$ flutter pub get
Resolving dependencies...
  
```



## Create Project

Get CLI dapat digunakan untuk membuat project Flutter dengan architecture yang tersedia serta opsi lainnya.

<ol style="list-style-type: none"> <li>1. Jalankan perintah <code>`get create project`</code></li> <li>2. Pilih <b>Flutter Project</b> [1]</li> <li>3. Beri nama project</li> <li>4. Beri nama company's domain</li> <li>5. Pilih bahasa pemrograman untuk IOS</li> <li>6. Pilih bahasa untuk android</li> <li>7. Pilih support null safe</li> <li>8. Pilih linter ssebagai code analysis</li> </ol>	 <pre>PS D:\&gt; get create project  Flutter Project Get browser Select which type of project you want to create? [1] 1 What is the name of the project? trial What is your company's domain? Example: com.yourcompany com.trial  [1] Swift [2] Kotlin What language do you want to use on iOS? [1] 1 [2] Kotlin What language do you want to use on android? [1] 1 [2] Yes Do you want to use null safety? [1] 1 [2] No  Running 'flutter create D:\trial' ...  \$ flutter create --no-pub -i swift -s kotlin --org com.trial D:\trial Recreating project D:\trial\ios\Runner\Classes\MainViewController.m (created) Dart files All done! You can find general documentation for Flutter at: https://docs.flutter.dev/ Detailed API documentation is available at: https://api.flutter.dev/ If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev</pre>
<ol style="list-style-type: none"> <li>9. Pilih architecture yang akan digunakan.</li> <li>10. Pada tutorial ini menggunakan <b>GetX Pattern</b> (opsional)</li> </ol>	 <pre>flutter pub get Creating .pubspec.lock \$ flutter pub get To: flutter pub outdated for more information. Package: flutter_lints installed File: analysis_options.yaml created successfully at path: analysis_options.yaml  [1] GetX Pattern (by Hand) [2] BLOC (by Repository) Which architecture do you want to use? [1] 1 [2] No Your lib folder is not empty. Are you sure you want to overwrite your application? WARNING: This action is irreversible [1] 1 Package: get installed File: main.dart created successfully at path: lib/main.dart File: home_controller.dart created successfully at path: lib/app/modules/home/controllers/home_controller.d All done! You can find general documentation for Flutter at: https://docs.flutter.dev/ Detailed API documentation is available at: https://api.flutter.dev/ If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev</pre>
<ol style="list-style-type: none"> <li>11. Pilih overwrite application</li> <li>12. Tunggu hingga proses selesai</li> <li>13. Buka project yang telah di-overwrite pada IDE atau Text Editor</li> </ol>	 <pre>Which architecture do you want to use? [1] 1 [2] No Your lib folder is not empty. Are you sure you want to overwrite your application? WARNING: This action is irreversible [1] 1 Package: get installed File: main.dart created successfully at path: lib/main.dart File: home_controller.dart created successfully at path: lib/app/modules/home/controllers/home_controller.d File: home_view.dart created successfully at path: lib/app/modules/home/views/home_view.dart File: home_binding.dart created successfully at path: lib/app/modules/home/bindings/home_binding.dart File: app_routes.dart created successfully at path: lib/app/routes/app_routes.dart File: app_pages.dart created successfully at path: lib/app/routes/app_pages.dart Home routes created successfully Home pages created successfully GetX Pattern structure successfully generated.  Running 'flutter pub get' ...</pre>

## Custom

Struktur infrastruktur project juga dapat dikustomisasi sesuai kebutuhan masing-masing dengan menambah, mengurangi, atau memodifikasi struktur proyek sesuai keperluan.

## GetMaterialApp

[GetMaterialApp](#) adalah sebuah widget yang telah dikonfigurasi sebelumnya, dengan default MaterialApp sebagai child, sehingga tidak diperlukan lagi konfigurasi secara manual. GetMaterialApp menyediakan fitur seperti routing, injeksi dependensi, translasi/terjemahan, dan berbagai kebutuhan navigasi lainnya. GetMaterialApp diperlukan untuk menangani route, snackbar, internasionalisasi/terjemahan, bottomSheet, dialog, dan high-level API yang berhubungan dengan routing dan konteks GetX.

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

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

  @override
  Widget build(BuildContext context) {
    SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);
    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: "/",
      unknownRoute: GetPage(
        name: "/notfound",
        page: () => NotFoundPage(),
      ),
      getPages: [
        // root
        GetPage(
          name: "/",
          page: () => RootPage(),
        ),
        // other
        GetPage(
          name: "/second",
          page: () => SecondPage(),
        ),
        GetPage(
          name: "/third",
          page: () => ThirdPage(),
        ),
      ],
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      themeMode: ThemeMode.system,
      translationsKeys: const {
        'en_US': {
```

```

        'welcome': 'Welcome',
        'page_root': 'Root Page',
        'page_login': 'Login Page',
        'page_register': 'Register Page',
        'page_notfound': 'Not Found Page',
    },
    },
    locale: const Locale('en', 'US'),
);
}
}

```

debugShowCheckedModeBanner	bool	Menghilangkan banner "Debug" pada aplikasi
initialRoute	String?	Screen pertama yang akan ditampilkan saat aplikasi dibuka
unknownRoute	GetPage?	Screen yang akan ditampilkan saat route yang diminta tidak ditemukan (404)
getPages	List<GetPage>	Daftar screen atau routes yang ada pada aplikasi
theme	ThemeData?	Konfigurasi untuk light theme mode
darkTheme	ThemeData?	Konfigurasi untuk dark theme mode
themeMode	ThemeMode?	Mode tema yang digunakan aplikasi: light (terang), dark (gelap), atau system (mengikuti sistem)
translationsKeys	Map<String, Map<String, String>>?	Translation aplikasi atau sebagai kamus bahasa pada aplikasi
locale	Locale?	Lokasi bahasa yang dipakai

required nullable

## GetPage

Sebagian aplikasi memiliki beberapa screen dengan informasi yang berbeda-beda. Dalam Flutter, screens atau pages disebut juga dengan [routes](#). GetPage adalah class milik GetX untuk menginisialisasi screens dan routes pada aplikasi [navigate-GetX](#).

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

  @override
  Widget build(BuildContext context) {
    SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);
    return GetMaterialApp(
      // other code
    ),
    getPages: [
      GetPage(
        name: "/",
        page: () => const RootPage(),
      ),
      GetPage(
        name: "/second",
        page: () => const SecondPage(),
      ),
      GetPage(
        name: "/third",
        page: () => const ThirdPage(),
      ),
    ],
    // other code
  );
}
```

name	String	Nama dari route/page/screen/path URL (untuk web)
page	Widget Function()	Widget atau class widget yang akan ditampilkan
binding	Bindings?	Mengkoneksikan View dan Controller secara mudah
middlewares	List?	Memantau event pada route dan memberikan tindakan sesuai kebutuhan

**required** nullable

Setelah `GetMaterialApp` dan `GetPage` dikonfigurasi, fitur [navigasi GetX](#) sudah dapat digunakan.

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(SecondPage()),
                child: const Text("Get.to"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.toNamed('second'),
                child: const Text("Get.toNamed"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
```

```

children: [
  const Text("Second Page", style: TextStyle(fontSize: 30)),
  SizedBox(
    width: 120,
    child: ElevatedButton(
      onPressed: () => Get.off(ThirdPage()),
      child: const Text("Get.off"),
    ),
  ),
  SizedBox(
    width: 120,
    child: ElevatedButton(
      onPressed: () => Get.offAll(ThirdPage()),
      child: const Text("Get.offAll"),
    ),
  ),
  SizedBox(
    width: 120,
    child: ElevatedButton(
      onPressed: () => Get.back(),
      child: const Text("Get.back"),
    ),
  ),
],
),
);
}
}

```

```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Third Page", style: TextStyle(fontSize: 30)),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.back(),

```

```

        child: const Text("Get.back"),
      ),
    ),
  ],
),
),
);
}
}

```

#### Navigasi menggunakan GetX

<code>Get.to(OtherScreen())</code>	Berpindah halaman dengan memanggil widget class lain.
<code>Get.toNamed('other')</code>	Berpindah halaman dengan sebuah nama route.
<code>Get.back()</code>	Kembali ke halaman sebelumnya.
<code>Get.off(NextScreen())</code>	Berpindah halaman dengan mengganti halaman sebelumnya.
<code>Get.offAll(NextScreen())</code>	Berpindah halaman dan menghapus semua halaman sebelumnya.

## GetController

[GetController](#) adalah class pada GetX yang berfungsi untuk mengontrol informasi yang digunakan pada aplikasi. Penggunaan controller dapat dilakukan dengan beberapa cara tergantung kebutuhannya.

```
class CountController extends GetxController {
  final count = 0.obs;

  Future<CountController> init() async {
    const duration = Duration(seconds: 5);
    Future.delayed(duration, () => ++count.value);
    print("CountController:: Future init ${count.value}");
    return this;
  }

  @override
  void onInit() {
    super.onInit();
    ++count.value;
    print("CountController:: onInit ${count.value}");
  }

  @override
  void onReady() {
    super.onReady();
    ++count.value;
    print("CountController:: onReady ${count.value}");
  }

  @override
  void onClose() {
    super.onClose();
    --count.value;
    print("CountController:: onClose ${count.value}");
  }

  void increment() {
    count.value++;
    print("CountController:: increment ${count.value}");
  }
}
```

```
class LazyCountController extends GetxController {
  final count = 0.obs;

  @override
  void onInit() {
    super.onInit();
    count.value++;
    print("LazyCountController:: onInit ${count.value}");
  }
}
```



```

@Override
void onReady() {
    super.onReady();
    count.value++;
    print("LazyCountController:: onReady ${count.value}");
}

@Override
void onClose() {
    super.onClose();
    count.value--;
    print("LazyCountController:: onClose ${count.value}");
}

void increment() {
    count.value++;
    print("LazyCountController:: increment ${count.value}");
    update();
}
}

```

obs	Observable merupakan Rx Types yang memiliki beragam metode dan operator.
onInit	Function bawaan controller yang dipanggil segera setelah widget dialokasikan di memori.
onReady	Function bawaan controller yang dipanggil setelah <code>onInit()</code> . Bagus digunakan untuk navigasi, snackbar, dialogs, route baru, atau async request.
onClose	Function bawaan controller yang dipanggil sebelum <code>onDelete()</code> untuk melakukan dispose resources, seperti menutup events, streams, atau objek yang berpotensi menyebabkan memory leaks.
increment	Function pada umumnya yang berada dalam controller.

## Implementasi

```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            const Text("Root Page", style: TextStyle(fontSize: 30)),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(SecondPage()),
                child: const Text("Get.put"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(ThirdPage()),
                child: const Text("Get.lazyPut"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(FourthPage()),
                child: const Text("Get.putAsync"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

**Get.put**

```

class SecondPage extends StatelessWidget {
  SecondPage({super.key});
  // Menginisialisasi controller menggunakan Get.put
  final controller = Get.put(CountController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Get.put Page", style: TextStyle(fontSize: 30)),
            // Menggunakan Obx untuk mendengarkan perubahan pada controller.count
            Obx(
              () => Text(
                "${controller.count}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => controller.increment(),
                child: const Text("Increment"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.back(),
                child: const Text("Get.back"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

**Get.lazyPut**

```

class ThirdPage extends StatelessWidget {
  ThirdPage({super.key});
  // Menginisialisasi controller menggunakan Get.lazyPut
  // Get.lazyPut() digunakan untuk menginisialisasi controller hanya saat pertama kali
  dibutuhkan
  final lazyPut = Get.lazyPut(() => LazyCountController());
  // Mengambil instance dari controller yang telah diinisialisasi
  final controller = Get.find<LazyCountController>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Get.lazyPut Page", style: TextStyle(fontSize: 30)),
            // Menggunakan Obx untuk mendengarkan perubahan pada controller.count
            Obx(
              () => Text(
                "${controller.count}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => controller.increment(),
                child: const Text("Increment"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.back(),
                child: const Text("Get.back"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

**Get.putAsync**

```

class FourthPage extends StatelessWidget {
  FourthPage({super.key});



  // Mengambil instance dari controller yang telah diinisialisasi dengan Get.putAsync
  final controller = Get.find<CountController>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Get.putAsync Page", style: TextStyle(fontSize: 30)),
            Obx(
              () => Text(
                "${controller.count}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => controller.increment(),
                child: const Text("Increment"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.back(),
                child: const Text("Get.back"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

```

## GetService

[GetService](#) adalah class mirip dengan [GetXController](#), yaitu memiliki metode `onInit()`, `onReady()`, dan `onClose()`, tetapi tidak memiliki "logic" di dalamnya. [GetService](#) hanya memberi tahu Sistem Dependency Injection GetX bahwa subclass ini tidak bisa dihapus dari memori.

1. Buka file `pubspec.yaml` pada project
  -  nama\_project
  -  pubspec.yaml
2. Tambahkan dependensi `get_storage`, lalu save

```
dependencies:
  flutter:
    sdk: flutter

  get: 4.6.5
  get_storage: 2.1.1
```

3. Buka terminal lalu jalankan perintah `flutter pub get` untuk menginstall library
4. Setelah terinstall, maka GetX sudah dapat digunakan
5. Implementasi GetX Service

main.dart:

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Get.putAsync(() => StorageService().init());
  runApp(const Main());
}
```

```
class StorageService extends GetxService {
  final storage = GetStorage("GetService");
  final inputControl = TextEditingController();
  RxString message = ''.obs;

  Future<StorageService> init() async {
    await GetStorage.init("GetService");
    await writeStorage("GetService", "Welcome to GetService");
    message.value = await readStorage("GetService");
    storage.listenKey("GetService", (value) {
      message.value = value;
    });
  }
}
```

```

        print("StorageService:: ${message.value}");
    });
    return this;
}

readStorage(String key) {
    return storage.read(key);
}

writeStorage(String key, dynamic value) {
    return storage.write(key, value);
}
}

```

```

class RootPage extends StatelessWidget {
  RootPage({super.key});
  final service = Get.find<StorageService>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text("Root Page", style: TextStyle(fontSize: 30)),
            Obx(
              () => Text(
                service.message.value,
                style: const TextStyle(fontSize: 30),
              ),
            ),
            Container(
              margin: const EdgeInsets.only(top: 10),
              padding: const EdgeInsets.symmetric(horizontal: 20),
              child: TextField(
                controller: service.inputControl,
                onChanged: (value) {
                  service.writeStorage("GetService", value);
                },
                decoration: InputDecoration(
                  label: const Text('Realtime Storage'),
                  border: OutlineInputBorder(
                    borderSide: BorderSide(
                      width: 1.5,

```

```

        color: Theme.of(context).colorScheme.onBackground,
      ),
    ),
  ),
),
],
),
),
);
}
}

```

Get.find<StorageService>	S find<S>({String? tag})	Sebuah function dari Get yang dipakai untuk menggunakan GetService atau GetController yang sudah di inisialisasi
Obx	Widget	Sebuah class yang akan me return widget dengan metode Widget Function()
service.message.obs	RxString	Observable merupakan Rx Types yang memiliki beragam metode dan operator internal
storage.listenKey	void Function()	Untuk memantau perubahan pada storage yang memiliki key tertentu
message.value	void set value(String val)	Value adalah sebuah function setter yang dari tipe data Rx

**required** nullable



## Bindings

Bindings dalam GetX digunakan untuk menginisialisasi dan mengatur dependensi seperti controllers, repository, atau API secara otomatis tanpa harus melakukannya secara langsung di setiap halaman (page). Hal ini membantu menjaga kode lebih terstruktur dan mudah dikelola.

### Implementasi

#### 1. CountBinding

```
class CountBinding extends Bindings {
  @override
  void dependencies() {
    Get.put(CountController());
  }
}
```

#### 2. LazyCountBinding

```
class LazyCountBinding extends Bindings {
  @override
  void dependencies() {
    Get.lazyPut(() => LazyCountController());
  }
}
```

### Penggunaan pada GetX

```
GetPage(
  name: "/",
  page: () => RootPage(),
),
GetPage(
  name: "/second",
  page: () => SecondPage(),
  binding: CountBinding(),
),
GetPage(
  name: "/third",
  page: () => ThirdPage(),
  binding: LazyCountBinding(),
),
GetPage(
  name: "/fourth",
  page: () => FourthPage(),
),
```

```

class RootPage extends StatelessWidget {
  const RootPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text("Root Page", style: TextStyle(fontSize: 30)),
          ElevatedButton(
            onPressed: () => Get.toNamed("/second"),
            child: Text("Go to Get.put"),
          ),
          ElevatedButton(
            onPressed: () => Get.toNamed("/third"),
            child: Text("Go to Get.lazyPut"),
          ),
          ElevatedButton(
            onPressed: () => Get.toNamed("/fourth"),
            child: Text("Go to Get.putAsync"),
          ),
        ],
      ),
    );
  }
}

```

```

class SecondPage extends StatelessWidget {
  SecondPage({super.key});
  final controller = Get.find<CountController>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text("Get.put Page", style: TextStyle(fontSize: 30)),
          Obx(
            () => Text(
              "${controller.count}",
              style: TextStyle(fontSize: 30),
            ),
          ),
          ElevatedButton(
            onPressed: () => controller.increment(),
            child: Text("Increment"),
          ),
        ],
      ),
    );
  }
}

```

```

        ElevatedButton(
            onPressed: () => Get.back(),
            child: Text("Back"),
        ),
    ],
),
);
}
}

```

```

class ThirdPage extends StatelessWidget {
  ThirdPage({super.key});
  final controller = Get.find<LazyCountController>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text("Get.lazyPut Page", style: TextStyle(fontSize: 30)),
          Obx(
            () => Text(
              "${controller.count}",
              style: TextStyle(fontSize: 30),
            ),
          ),
          ElevatedButton(
            onPressed: () => controller.increment(),
            child: Text("Increment"),
          ),
          ElevatedButton(
            onPressed: () => Get.back(),
            child: Text("Back"),
          ),
        ],
      ),
    );
  }
}

```

## GetMiddleware

[GetMiddleware](#) digunakan untuk mengawasi dan mengontrol peristiwa routing dalam aplikasi. Middleware, dapat menambahkan fungsionalitas tambahan, seperti autentikasi, otorisasi, atau validasi, sebelum pengguna diarahkan ke halaman tertentu.

### Implementasi

```
class SessionMiddleware extends GetMiddleware {
  StorageService storageService = Get.find();

  @override
  RouteSettings? redirect(String? route) {
    storageService.message.refresh();
    if (!storageService.message.contains("user")) {
      return const RouteSettings(name: "/");
    }

    return null;
  }
}
```

### Penggunaan Middleware pada Route

```
GetPage(
  name: "/second",
  page: () => SecondPage(),
  binding: CountBinding(),
  middlewares: [SessionMiddleware()],
),
```

Pada contoh berikut, middleware `SessionMiddleware` memastikan bahwa pengguna hanya dapat mengakses `SecondPage` jika session valid (misalnya, sudah login). Jika tidak, pengguna akan diarahkan kembali ke halaman utama (`RootPage`).

```
class RootPage extends StatelessWidget {
  RootPage({super.key});
  final service = Get.find<StorageService>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
```

```

children: [
  Text("Root Page", style: TextStyle(fontSize: 30)),
  Obx(
    () => Text(
      "${service.message.obs}",
      style: TextStyle(fontSize: 30),
    ),
  ),
  Container(
    margin: const EdgeInsets.symmetric(horizontal: 20),
    child: TextField(
      controller: service.inputControl,
      onChanged: (value) {
        service.writeStorage("GetService", value);
      },
      decoration: InputDecoration(
        label: Text('Session Storage'),
        border: OutlineInputBorder(
          borderSide: BorderSide(
            width: 1.5,
            color: Theme.of(context).colorScheme.onBackground,
          ),
        ),
      ),
    ),
  ),
  SizedBox(
    width: 120,
    child: ElevatedButton(
      onPressed: () => Get.toNamed("/second"),
      child: const Text("login"),
    ),
  ),
],
);
}
}

```

## GetView

Keunggulan utama [GetView](#) adalah menyediakan akses mudah ke controller yang telah diinisialisasi, tanpa perlu memanggil `Get.find<Controller>()` secara manual.

### Implementasi

#### 1. Inisialisasi Service atau Controller

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Get.putAsync(() => StorageService().init());
  runApp(const Main());
}
```

#### 2. Buat Controller

```
class ViewController extends GetxController {
  final service = Get.find<StorageService>();
  final count = 0.obs;

  void increment() {
    count.value++;
    print("ViewController:: increment ${count.value}");
    update();
  }
}
```

#### 3. Buat Binding untuk Controller

```
class ViewBinding extends Bindings {
  @override
  void dependencies() {
    Get.put(ViewController());
  }
}
```

#### 4. Tentukan Route dan Bindings di GetPage

```
GetPage(
  name: "/",
  page: () => const RootPage(),
  binding: ViewBinding(),
),
```

## 5. Gunakan GetView di View

```
class RootPage extends GetView<ViewController> {
  const RootPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text(
              "Root Page",
              style: TextStyle(fontSize: 30),
            ),
            Obx(
              () => Text(
                "${controller.service.message.value.capitalize}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            Container(
              margin: const EdgeInsets.only(top: 10),
              padding: const EdgeInsets.symmetric(horizontal: 20),
              child: TextField(
                controller: controller.service.inputControl,
                onChanged: (value) {
                  controller.service.writeStorage("GetService", value);
                },
                decoration: InputDecoration(
                  label: const Text('Realtime Storage'),
                  border: OutlineInputBorder(
                    borderSide: BorderSide(
                      width: 1.5,
                      color: Theme.of(context).colorScheme.onBackground,
                    ),
                  ),
                ),
              ),
            ),
            Container(
              margin: const EdgeInsets.only(top: 30),
              child: Column(children: [
                Obx(
                  () => Text(
                    "${controller.count.obs}",

```

```
        style: const TextStyle(fontSize: 30),
      ),
    ),
    SizedBox(
      width: 120,
      child: ElevatedButton(
        onPressed: () => controller.increment(),
        child: const Text("increment"),
      ),
    ),
  ],
),
],
),
),
);
}
```



## GetBuilder

[GetBuilder](#) adalah salah satu cara untuk mengelola state di GetX. GetBuilder bekerja dengan cara mengontrol grup state yang ada dalam controller. Saat ada perubahan state, hanya widget yang menggunakan GetBuilder tersebut yang akan diupdate, tanpa harus melakukan perhitungan ulang atau perubahan pada widget lain yang tidak terkait.

### Penggunaan GetBuilder

```
GetBuilder<ViewController>(builder: (controller) {
  // UI yang ingin di-update berdasarkan state di ViewController
});
```

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

  @override
  Widget build(BuildContext context) {
    return GetBuilder<ViewController>(builder: (controller) {
      return Scaffold(
        body: SizedBox(
          width: context.width,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const Text(
                "Root Page",
                style: TextStyle(fontSize: 30),
              ),
              Text(
                "${controller.service.message.value.capitalize}",
                style: const TextStyle(fontSize: 30),
              ),
              Container(
                margin: const EdgeInsets.only(top: 10),
                padding: const EdgeInsets.symmetric(horizontal: 20),
                child: TextField(
                  controller: controller.service.inputControl,
                  onChanged: (value) {
                    controller.service.writeStorage("GetService", value);
                  },
                  decoration: InputDecoration(
                    label: const Text('Realtime Storage'),
                    border: OutlineInputBorder(
                      borderSide: BorderSide(
                        width: 1.5,
                        color: Theme.of(context).colorScheme.onBackground,

```

```

        ),
      ),
    ),
  ),
),
Container(
  margin: const EdgeInsets.only(top: 30),
  child: Column(
    children: [
      Text(
        "${controller.count.value}",
        style: const TextStyle(fontSize: 30),
      ),
      SizedBox(
        width: 120,
        child: ElevatedButton(
          onPressed: () => controller.increment(),
          child: const Text("increment"),
        ),
      ),
    ],
  ),
),
),
),
),
),
);
});
}
}

```

## Obx | Obs

**Obx** adalah sebuah widget dalam GetX yang digunakan untuk meng-update UI secara otomatis saat ada perubahan pada variabel yang bersifat observabel (**obs**). Tidak perlu lagi menggunakan **setState()** untuk memperbarui UI, sehingga lebih efisien dan sederhana. Sedangkan **Obs** Merupakan singkatan dari "Observable," yang dalam konteks GetX adalah variabel reaktif yang akan memicu pembaruan UI setiap kali nilainya berubah. Variabel ini digunakan dalam reaktif programming (Rx) untuk mengawasi dan bereaksi terhadap perubahan data.

### Controller variabel Obs

```
class CounterController extends GetxController {
  var count = 0.obs;

  void increment() {
    count++;
  }
}
```

### Penggunaan Obx

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

  @override
  Widget build(BuildContext context) {
    final CounterController controller = Get.put(CounterController());

    return Scaffold(
      appBar: AppBar(
        title: const Text('Obx & Obs Example'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Obx(() => Text(
              '${controller.count}',
              style: Theme.of(context).textTheme.headline4,
            )),
          ],
        ),
      ),
    );
  }
}
```

```

    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () => controller.increment(),
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ),
  );
}
}

```

## CODELAB

### TUGAS 1

1. Lakukan instalasi State Management GetX pada project kalian
2. Lakukan refactoring folder project dengan menerapkan architecture pattern
3. Lakukan refactoring ke codingan kalian agar codingan lebih rapi dan mudah untuk dibaca.

## KRITERIA & DETAIL PENILAIAN

### Bobot Penilaian Modul 1 Materi (15%)

Berhasil melakukan instalasi state management	40
Berhasil melakukan refactoring codingan dengan menerapkan architecture pattern	40
Codingan rapi dan mudah dibaca	20
<b>Total</b>	<b>100</b>