

Tugas Mandiri (Unguided)

Modifikasi tampilan Guided dari praktikum di atas:

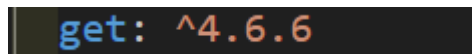
- a. Gunakan State Management dengan GetX:
 - Atur data menggunakan state management GetX agar lebih mudah dikelola.
 - Implementasi GetX meliputi pembuatan controller untuk mengelola data dan penggunaan widget Obx untuk menampilkan data secara otomatis setiap kali ada perubahan.
- b. Tambahkan Snackbar untuk Memberikan Respon Berhasil:
 - Tampilkan snackbar setelah setiap operasi berhasil, seperti menambah atau memperbarui data.
 - Gunakan Get.snackbar agar pesan sukses muncul di layar dan mudah dipahami oleh pengguna.

Penjelasan :

1. Menambahkan Get X untuk state manajemen

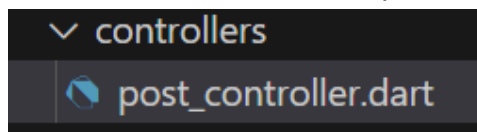
Sebelumnya, pengelolaan data dilakukan di dalam widget secara langsung dengan setState. Namun, dengan GetX, kita menggunakan controller untuk memisahkan logika dan pengelolaan data. Hal ini membuat kode lebih terstruktur dan mudah untuk dikelola, serta membuat UI lebih reaktif.

- a. Langkah pertama yaitu kita tambahkan packages bernama get pada pubsec.yaml pada dependencies . Pastikan gunakan versi yang terbaru, agar meminimalisir terjadinya eror.



```
get: ^4.6.6
```

- b. Buat folder controllers dan post_controllers.dart.



```
controllers  
  post_controller.dart
```

Controller ini bertanggung jawab untuk mengelola data dan status aplikasi.

```
import 'package:get/get.dart';  
import  
'package:unguided_pertemuan14/services/api_service.dart';  
  
class PostController extends GetxController {  
  var posts = <dynamic>[].obs;  
  var isLoading = false.obs;  
  final ApiService _apiService = ApiService();
```

```

// Fetch posts
Future<void> fetchPosts() async {
  isLoading.value = true;
  try {
    await _apiService.fetchPosts();
    posts.assignAll(_apiService.posts);
    Get.snackbar('Success', 'Posts berhasil diambil!',
      snackPosition: SnackPosition.BOTTOM);
  } catch (e) {
    Get.snackbar('Error', 'Gagal mengambil posts',
      snackPosition: SnackPosition.BOTTOM);
  } finally {
    isLoading.value = false;
  }
}

// Create post
Future<void> createPost() async {
  try {
    await _apiService.createPost();
    posts.add({
      'title': 'Flutter Post',
      'body': 'Ini contoh POST.',
      'id': posts.length + 1,
    });
    Get.snackbar('Success', 'Post berhasil ditambahkan!',
      snackPosition: SnackPosition.BOTTOM);
  } catch (e) {
    Get.snackbar('Error', 'Gagal menambahkan post',
      snackPosition: SnackPosition.BOTTOM);
  }
}

// Update post
Future<void> updatePost() async {
  try {
    await _apiService.updatePost();
    var post = posts.firstWhere((post) => post['id'] == 1);
    post['title'] = 'Updated Title';
    post['body'] = 'Updated Body';
    posts.refresh();
    Get.snackbar('Success', 'Post berhasil diperbarui!',
      snackPosition: SnackPosition.BOTTOM);
  } catch (e) {
    Get.snackbar('Error', 'Gagal memperbarui post',
      snackPosition: SnackPosition.BOTTOM);
  }
}

```

```

    }
  }

  // Delete post
  Future<void> deletePost() async {
    try {
      await _apiService.deletePost();
      posts.removeWhere((post) => post['id'] == 1);
      Get.snackbar('Success', 'Post berhasil dihapus!',
        snackPosition: SnackPosition.BOTTOM);
    } catch (e) {
      Get.snackbar('Error', 'Gagal menghapus post',
        snackPosition: SnackPosition.BOTTOM);
    }
  }
}

```

serta di dalam PostController memiliki properti yang dikelola dengan GetX, seperti posts dan isLoading. Kedua properti ini menggunakan tipe Rx, yang membuat mereka reaktif terhadap perubahan data.

- c. Langkah selanjutnya yaitu kita akan memodifikasi pada home_screen.dart. Di home_screen.dart, kita menggantikan penggunaan setState dengan controller dari GetX dan widget Obx untuk membuat UI yang reaktif.

```

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

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

    return Scaffold(
      appBar: AppBar(
        title: const Text('HTTP Request Example'),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(12),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [

```

```

// Obx digunakan untuk mendengarkan perubahan pada
posts dan isLoading
Obx(() => controller.isLoading.value
? const Center(child: CircularProgressIndicator())
: controller.posts.isEmpty
? Center(
  child: const Text(
    "Tekan tombol GET untuk mengambil data",
    style: TextStyle(fontSize: 12),
  ),
)
: Expanded(
  child: ListView.builder(
    itemCount: controller.posts.length,
    itemBuilder: (context, index) {
      return Padding(
        padding: const EdgeInsets.only(bottom: 12.0),
        child: Card(
          elevation: 4,
          child: ListTile(
            title: Text(
              controller.posts[index]['title'],
              style: const TextStyle(
                fontWeight: FontWeight.bold,
                fontSize: 12),
            ),
            subtitle: Text(
              controller.posts[index]['body'],
              style: const TextStyle(fontSize: 12),
            ),
          ),
        ),
      );
    },
  ),
)),
Center(
  child: ElevatedButton(
    onPressed: controller.fetchPosts, // Memanggil fungsi
fetchPosts dari controller
    style: ElevatedButton.styleFrom(backgroundColor:
Colors.orange),
    child: const Text('GET'),
  ),
),
const SizedBox(height: 12),
Center(

```

```

        child: ElevatedButton(
          onPressed: controller.createPost, // Memanggil fungsi
createPost dari controller
          style: ElevatedButton.styleFrom(backgroundColor:
Colors.green),
          child: const Text('POST'),
        ),
      ),
      const SizedBox(height: 12),
      Center(
        child: ElevatedButton(
          onPressed: controller.updatePost, // Memanggil fungsi
updatePost dari controller
          style: ElevatedButton.styleFrom(backgroundColor:
Colors.blue),
          child: const Text('UPDATE'),
        ),
      ),
      const SizedBox(height: 12),
      Center(
        child: ElevatedButton(
          onPressed: controller.deletePost, // Memanggil fungsi
deletePost dari controller
          style: ElevatedButton.styleFrom(backgroundColor:
Colors.red),
          child: const Text('DELETE'),
        ),
      ),
    ],
  ),
);
}

```

GetX Controller: Menggunakan `Get.put(PostController())` untuk mengakses controller yang berisi logika API dan data yang dikelola.

Obx: Widget Obx digunakan untuk memantau perubahan pada properti yang dideklarasikan dengan Rx (seperti `isLoading` dan `posts`). Ketika data berubah, widget akan otomatis di-rebuild.

- d. Kemudian kita akan memodifikasi `main.dart` pada `main.dart` kita cukup mengubah dari `material app` menjadi `getmaterial app`. Karena biasanya untuk memanfaatkan fitur-fitur dari `GetX` seperti `get.snackbar` dan `get.put` untuk mengelola state.

Output :

