

Nama : Fikri Maulana
Kelas : IF4410
NIM : 1301200239

MODUL 13

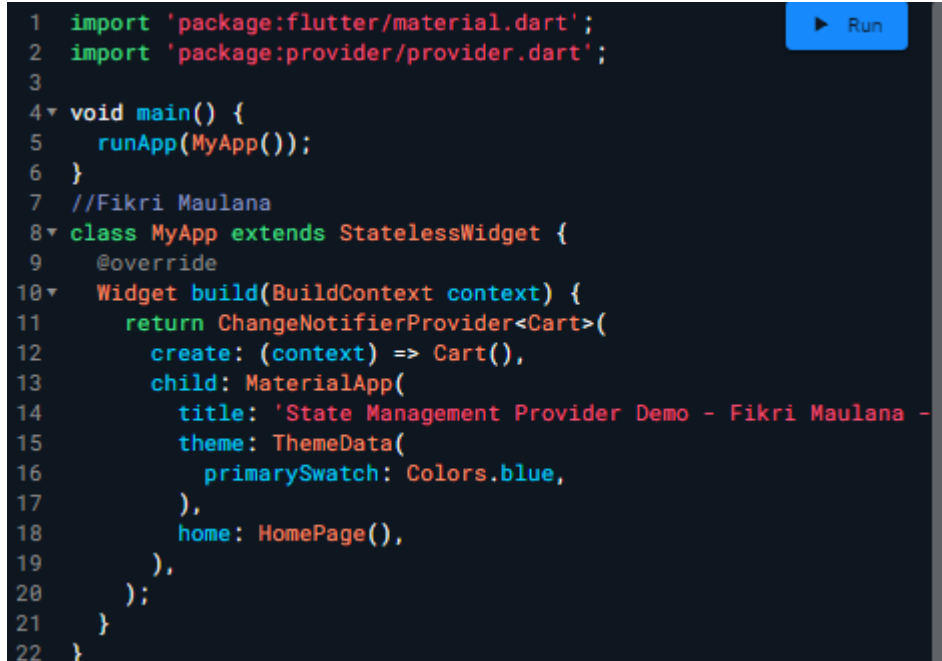
13.1 State Management

13.1.1. Pengenalan Model

Ketika aplikasi semakin kompleks dibuat, maka pasti akan ada saatnya dimana harus dibagikan state aplikasi ke berbagai halaman yang ada. Flutter adalah deklaratif, sehingga Flutter membangun user interface berdasarkan state saat ini. Dengan menggunakan state management, dapat dilakukan sentralisasi semua state dari berbagai macam UI Control untuk mengendalikan aliran data lintas aplikasi.

13.1.2 State Management Provider

Provider adalah salah satu state management yang tersedia di Flutter. Provider memiliki widget Consumer, yaitu sebuah widget yang listen terhadap perubahan value dari provider tersebut, yang nantinya akan melakukan rebuild widget di bawahnya ketika perubahan tersebut terjadi. Berikut ini adalah contoh penggunaan Provider dalam aplikasi pembelian sederhana.

A screenshot of a code editor showing Dart code for a Flutter application. The code imports 'package:flutter/material.dart' and 'package:provider/provider.dart'. It defines a 'main' function that calls 'runApp(MyApp())'. The 'MyApp' class extends 'StatelessWidget' and overrides the 'build' method. In the 'build' method, it returns a 'ChangeNotifierProvider<Cart>' widget. The 'create' property of the provider is set to 'Cart()', and the 'child' property is set to a 'MaterialApp' widget. The 'MaterialApp' has a title 'State Management Provider Demo - Fikri Maulana -', a blue primary swatch, and a 'HomePage()' as the home screen. A 'Run' button is visible in the top right corner of the code editor.

```
1 import 'package:flutter/material.dart';
2 import 'package:provider/provider.dart';
3
4 void main() {
5   runApp(MyApp());
6 }
7 //Fikri Maulana
8 class MyApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return ChangeNotifierProvider<Cart>({
12      create: (context) => Cart(),
13      child: MaterialApp(
14        title: 'State Management Provider Demo - Fikri Maulana -',
15        theme: ThemeData(
16          primarySwatch: Colors.blue,
17        ),
18        home: HomePage(),
19      ),
20    });
21  }
22 }
```

```

23 //Fikri Maulana
24 class HomePage extends StatelessWidget {
25   @override
26   Widget build(BuildContext context) {
27     return Scaffold(
28       appBar: AppBar(
29         title: Text('Shopping Cart - Fikri Maulana'),
30       ),
31       body: Column(
32         children: [
33           Expanded(
34             child: ItemList(),
35           ),
36           CartTotal(),
37           ClearCartButton(),
38         ],
39       ),
40     );
41   }
42 }

```

```

43 //Fikri Maulana
44 class ItemList extends StatelessWidget {
45   @override
46   Widget build(BuildContext context) {
47     final cart = Provider.of<Cart>(context);
48     return ListView.builder(
49       itemCount: cart.items.length,
50       itemBuilder: (context, index) {
51         final item = cart.items[index];
52         return ListTile(
53           title: Text(item.name),
54           subtitle: Text('\$${item.price.toStringAsFixed(2)}'),
55           trailing: IconButton(
56             icon: Icon(Icons.add),
57             onPressed: () => cart.addItem(item),
58           ),
59         );
60       },
61     );
62   }
63 }

```

```

64 //Fikri Maulana
65 class CartTotal extends StatelessWidget {
66   @override
67   Widget build(BuildContext context) {
68     final cart = Provider.of<Cart>(context);
69     return Padding(
70       padding: EdgeInsets.all(16.0),
71       child: Text(
72         'Total: \${cart.totalValue.toStringAsFixed(2)}',
73         style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.
74       ),
75     );
76   }
77 }
78 //Fikri Maulana
79 class ClearCartButton extends StatelessWidget {
80   @override
81   Widget build(BuildContext context) {
82     final cart = Provider.of<Cart>(context);
83     return ElevatedButton(
84       onPressed: () => cart.clearCart(),
85       child: Text('Clear Cart'),
86     );
87   }
88 }

```

```

89 //Fikri Maulana
90 class Cart extends ChangeNotifier {
91   List<Item> _items = [
92     Item(name: 'Item 1', price: 40.0),
93     Item(name: 'Item 2', price: 50.0),
94     Item(name: 'Item 3', price: 60.0),
95   ];
96
97   List<Item> get items => _items;
98
99   double get totalValue => _items.fold(0, (sum, item) => sum + i
100
101   void addItem(Item item) {
102     _items.add(item);
103     notifyListeners();
104   }
105
106   void clearCart() {
107     _items.clear();
108     notifyListeners();
109   }
110 }
111
112 class Item {
113   final String name;
114   final double price;
115
116   Item({required this.name, required this.price});
117 }
118

```

Shopping Cart - Fikri Maulana

DEBUG

Item 1
\$40.00

+

Item 2
\$50.00

+

Item 3
\$60.00

+

Total: \$150.00

Clear Cart

13.2 Networking

Networking dalam Flutter adalah kemampuan aplikasi untuk berkomunikasi dengan server eksternal dan bertukar data. Flutter menyediakan berbagai metode dan pustaka untuk melakukan operasi jaringan, termasuk pengambilan dan penghapusan data, pengiriman dan pembaruan data, serta parsing JSON. Dengan dukungan pustaka HTTP seperti "http" atau "dio", pengembang dapat mengambil data dari server menggunakan metode GET, mengirim dan memperbarui data menggunakan metode POST, PUT, dan PATCH, serta menghapus data menggunakan metode DELETE. Selain itu, Flutter juga menyediakan dukungan yang kuat untuk parsing JSON melalui pustaka seperti "dart:convert". Dengan fitur-fitur ini, pengembang dapat dengan mudah mengintegrasikan aplikasi Flutter dengan server eksternal dan melakukan pertukaran data secara efisien.

```
1 import 'dart:convert';
2 import 'package:flutter/material.dart';
3 import 'package:http/http.dart' as http;
4
5 void main() {
6   runApp(MyApp());
7 }
8 //Fikri Maulana - 1301200239
9 class MyApp extends StatelessWidget {
10   @override
11   Widget build(BuildContext context) {
12     return MaterialApp(
13       title: 'Networking Demo - Fikri Maulana',
14       theme: ThemeData(
15         primarySwatch: Colors.blue,
16       ),
17       home: MyHomePage(),
18     );
19   }
20 }
```

```

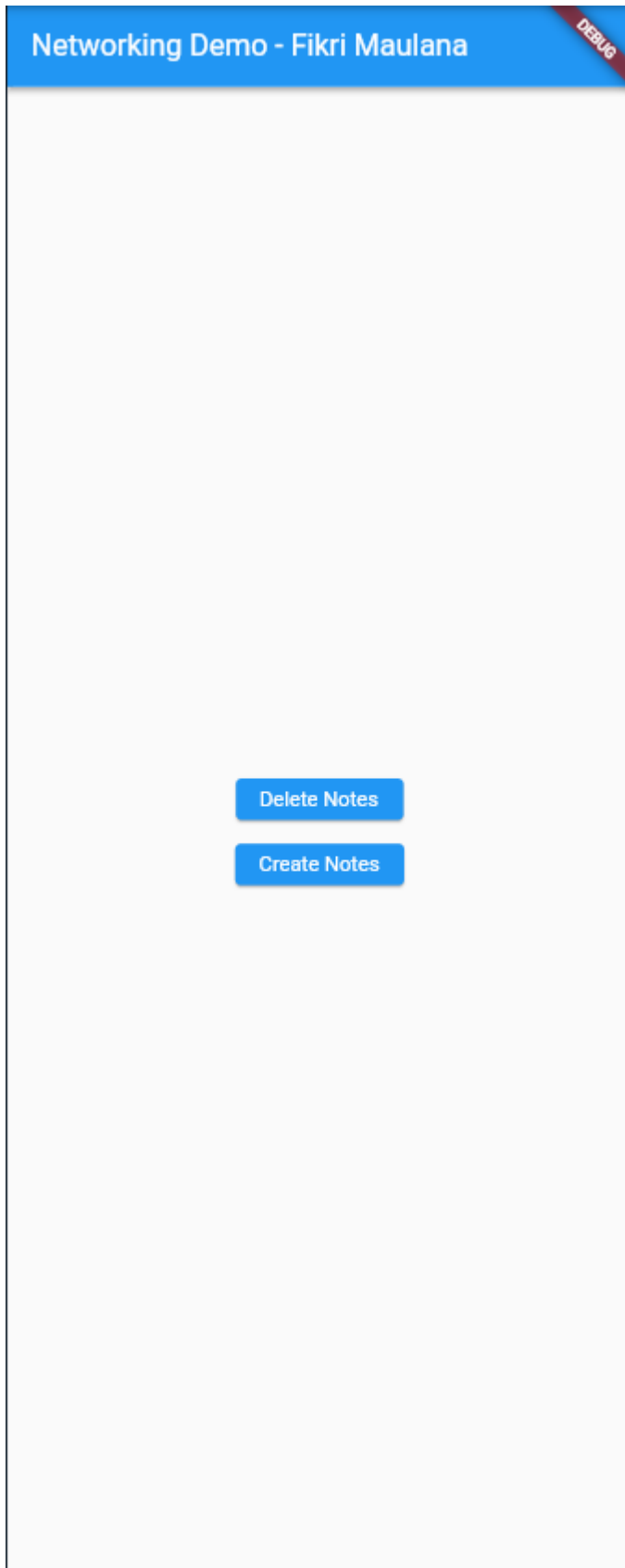
21 //Fikri Maulana - 1301200239
22 class MyHomePage extends StatelessWidget {
23   @override
24   Widget build(BuildContext context) {
25     return Scaffold(
26       appBar: AppBar(
27         title: Text('Networking Demo - Fikri Maulana'),
28       ),
29       body: Center(
30         child: Column(
31           mainAxisAlignment: MainAxisAlignment.center,
32           children: [
33             ElevatedButton(
34               onPressed: () {
35                 deleteNotes('1');
36               },
37               child: Text('Delete Notes'),
38             ),
39             SizedBox(height: 16),
40             ElevatedButton(
41               onPressed: () {
42                 createNotes('New Notes');
43               },
44               child: Text('Create Notes'),
45             ),
46           ],
47         ),
48       ),
49     );
50   }

```

```

51 //Fikri Maulana - 1301200239
52 Future<void> deleteNotes(String id) async {
53     final http.Response response = await http.delete(
54         Uri.parse('https://jsonplaceholder.typicode.com/albums/$id'),
55         headers: <String, String>{
56             'Content-Type': 'application/json; charset=UTF-8',
57         },
58     );
59     if (response.statusCode == 200) {
60         print('Notes $id deleted successfully');
61     } else {
62         print('Failed to delete album. Error: ${response.statusCode}');
63     }
64 }
65 //Fikri Maulana - 1301200239
66 Future<void> createNotes(String title) async {
67     final http.Response response = await http.post(
68         Uri.parse('https://jsonplaceholder.typicode.com/albums'),
69         headers: <String, String>{
70             'Content-Type': 'application/json; charset=UTF-8',
71         },
72         body: jsonEncode(<String, String>{
73             'title': title,
74         }),
75     );
76     if (response.statusCode == 201) {
77         print('Notes created successfully');
78         final Map<String, dynamic> album = jsonDecode(response.body);
79         print('New Notes ID: ${album['id']}');
80         print('New Notes Title: ${album['title']}');
81     } else {
82         print('Failed to create album. Error: ${response.statusCode}');
83     }
84 }
85 }
86

```



Ketika Create dan Delete notes ditekan

Console Documentation

```
Notes created successfully  
New Notes ID: 101  
New Notes Title: New Notes  
Notes 1 deleted successfully
```