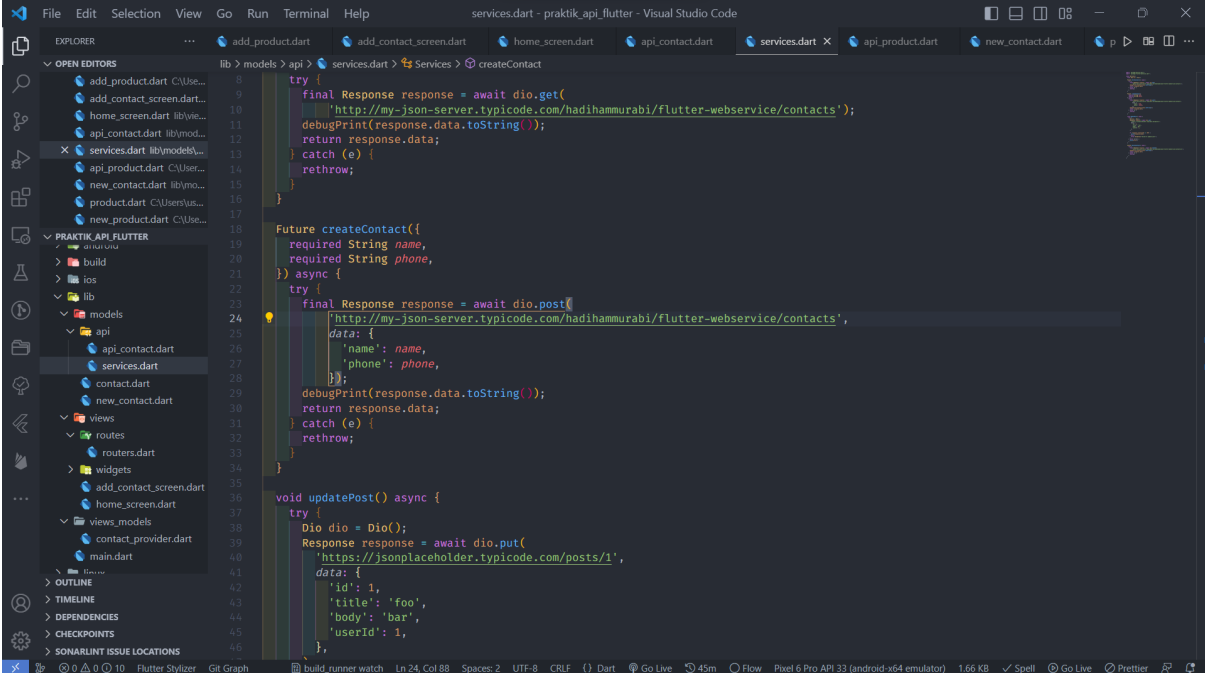


Nama : Mohd Ryan Obillah
Kelas : SIB 4 Flutter Kelas B

Soal Prioritas 1 (80)

1. Lakukan **POST** request dengan menggunakan DIO untuk mengirimkan data contact! gunakan url pada slide 12.

Jawab :



```
services.dart - praktik_api_flutter - Visual Studio Code

try {
  final Response response = await dio.get(
    'http://my-json-server.typicode.com/hadihammurabi/flutter-webservice/contacts');
  debugPrint(response.data.toString());
  return response.data;
} catch (e) {
  rethrow;
}

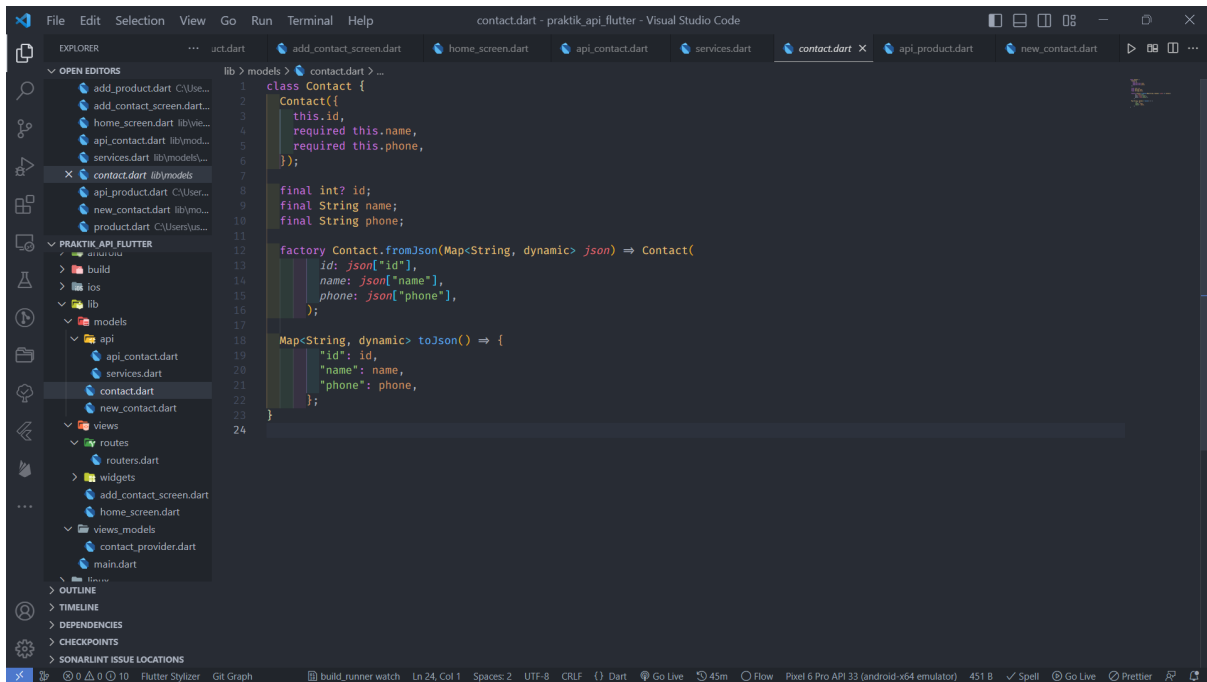
Future createContact({
  required String name,
  required String phone,
}) async {
  try {
    final Response response = await dio.post(
      'http://my-json-server.typicode.com/hadihammurabi/flutter-webservice/contacts',
      data: {
        'name': name,
        'phone': phone,
      });
    debugPrint(response.data.toString());
    return response.data;
  } catch (e) {
    rethrow;
  }
}

void updatePost() async {
  try {
    Dio dio = Dio();
    Response response = await dio.put(
      'https://jsonplaceholder.typicode.com/posts/1',
      data: {
        'id': 1,
        'title': 'foo',
        'body': 'bar',
        'userId': 1,
      },
    );
  } catch (e) {
    rethrow;
  }
}
```

2. Ubahlah bentuk JSON berikut kedalam bentuk Object dari suatu class! Gunakan data JSON pada URL berikut:

<https://my-json-server.typicode.com/hadihammurabi/flutter-webservice/contacts/2>

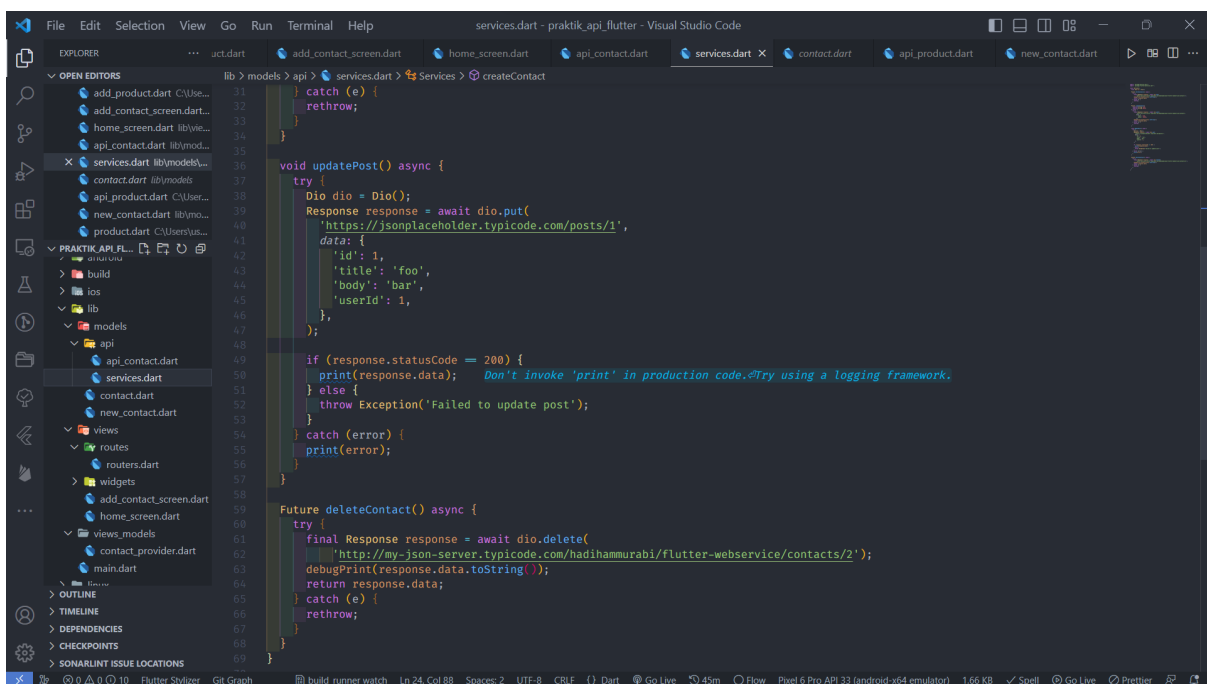
Jawab :



```
1 class Contact {
2   Contact({
3     this.id,
4     required this.name,
5     required this.phone,
6   });
7
8   final int? id;
9   final String name;
10  final String phone;
11
12  factory Contact.fromJson(Map<String, dynamic> json) => Contact(
13    id: json["id"],
14    name: json["name"],
15    phone: json["phone"],
16  );
17
18  Map<String, dynamic> toJson() => {
19    "id": id,
20    "name": name,
21    "phone": phone,
22  };
23
24 }
```

3. Lakukan **PUT** request dengan menggunakan DIO.

Jawab :

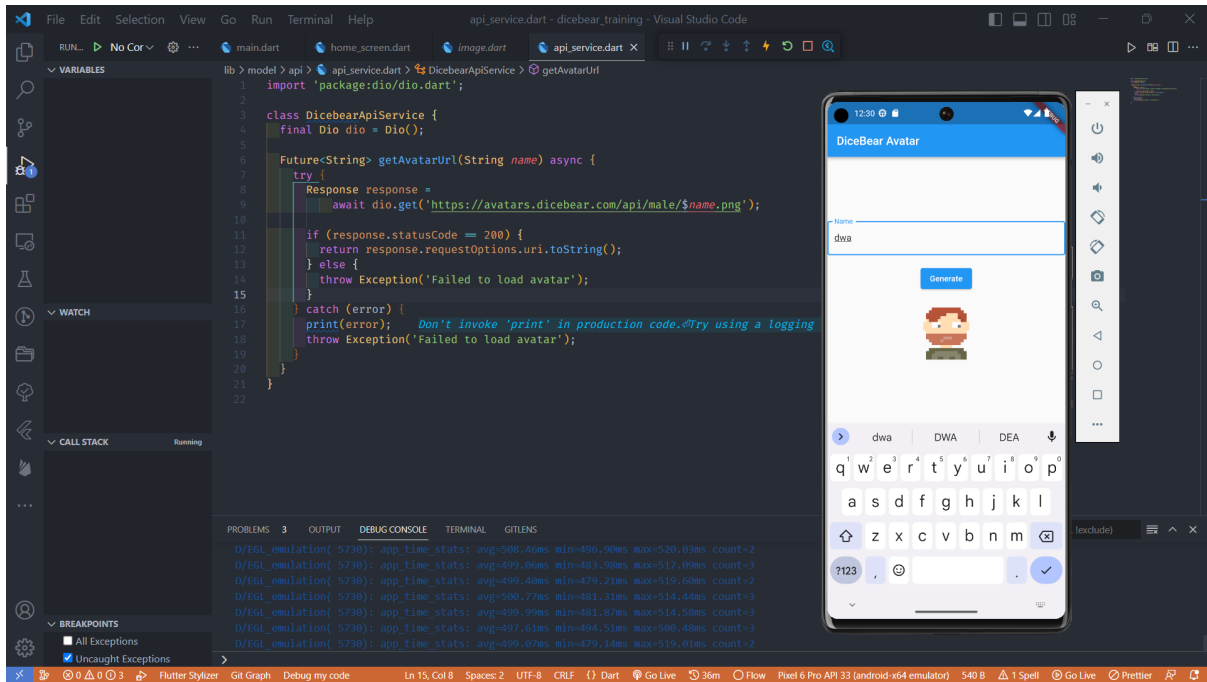


```
31 } catch (e) {
32   rethrow;
33 }
34
35 void updatePost() async {
36   try {
37     Dio dio = Dio();
38     Response response = await dio.put(
39       'https://jsonplaceholder.typicode.com/posts/1',
40       data: {
41         'id': 1,
42         'title': 'foo',
43         'body': 'bar',
44         'userId': 1,
45       },
46     );
47
48     if (response.statusCode == 200) {
49       print(response.data); // Don't invoke 'print' in production code. Try using a logging framework.
50     } else {
51       throw Exception('Failed to update post');
52     }
53   } catch (error) {
54     print(error);
55   }
56 }
57
58 Future deleteContact() async {
59   try {
60     final Response response = await dio.delete(
61       'https://my-json-server.typicode.com/hadihamurabi/flutter-webservice/contacts/2');
62     debugPrint(response.data.toString());
63     return response.data;
64   } catch (e) {
65     rethrow;
66   }
67 }
68
69 }
```

Soal Prioritas 2 (20)

1. Buatlah sebuah aplikasi sederhana untuk menampilkan gambar yang diambil melalui DiceBear API. Gambar harus diambil melalui mekanisme data fetching (tidak boleh hard code secara langsung).

Jawab :



Soal Eksplorasi (20)

1. Buatlah sebuah aplikasi untuk menampilkan gambar berdasarkan teks yang dimasukkan oleh user. Kriteria dari aplikasi yang dikembangkan adalah sebagai berikut:
 1. Gambar diambil melalui mekanisme data fetching ke DiceBear API.
 2. Ketika tombol generate ditekan. Maka gambar ditampilkan.

Jawab :

Visual Studio Code interface showing a Flutter application running on an Android emulator. The application is titled "DiceBear Avatar" and displays a pixelated avatar for the name "Ryan".

The code in the editor is as follows:

```
lib > model > api > api_service.dart > DicebearApiService > getAvatarUrl
1 import 'package:dio/dio.dart';
2
3
4
5
6 class DicebearApiService {
7   final Dio dio = Dio();
8
9   Future<String> getAvatarUrl(String name) async {
10     try {
11       Response response =
12         await dio.get('https://avatars.dicebear.com/api/male/$name.png');
13
14       if (response.statusCode == 200) {
15         return response.requestOptions.uri.toString();
16       } else {
17         throw Exception('Failed to load avatar');
18       }
19     } catch (error) {
20       print(error);
21       throw Exception('Failed to load avatar');
22     }
23   }
24 }
```

The emulator screen shows the app's UI, which includes a text input field labeled "Name" containing the text "Ryan", a "Generate" button, and the resulting pixelated avatar.

The bottom status bar indicates the app is running on a Pixel 6 Pro API 33 (android-x64 emulator) with a 540 B resolution. The status bar also shows various icons for debugging and development tools.