

Laporan Praktikum

Mobile Programming

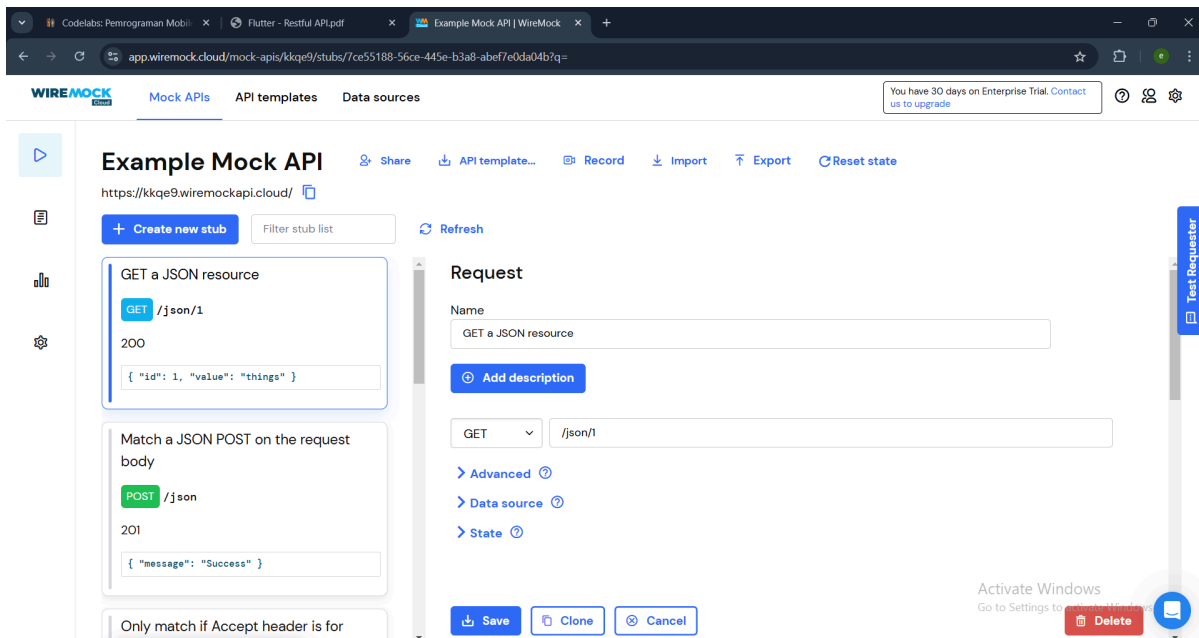
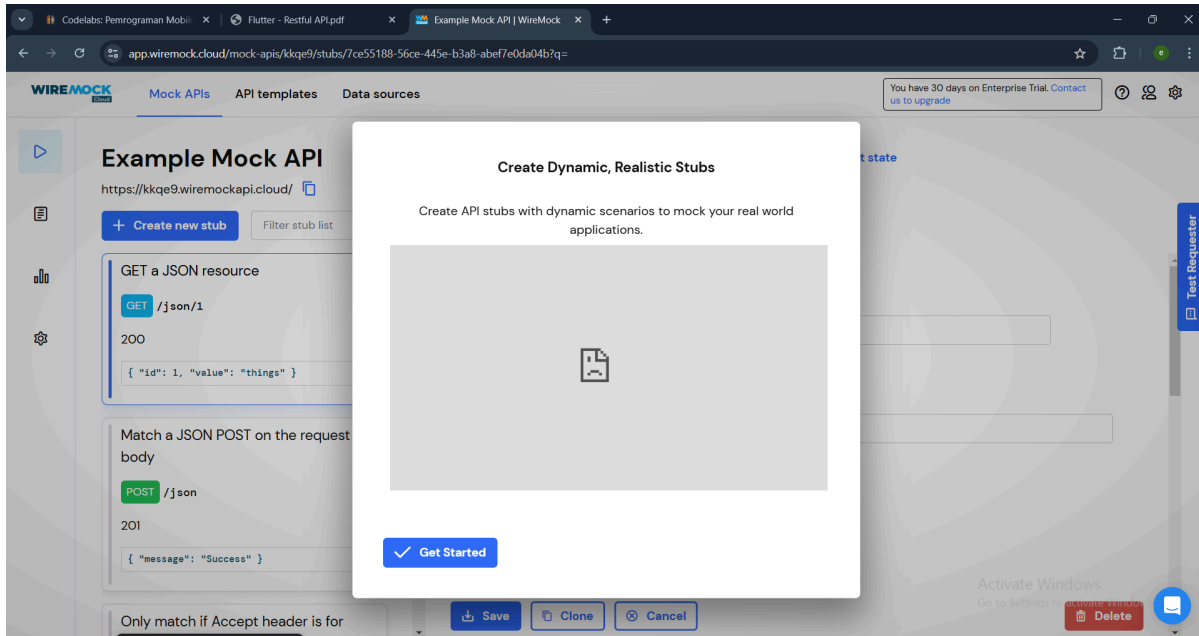
NAMA : Ersa Oktavian Ramadan

NIM : 2241720208

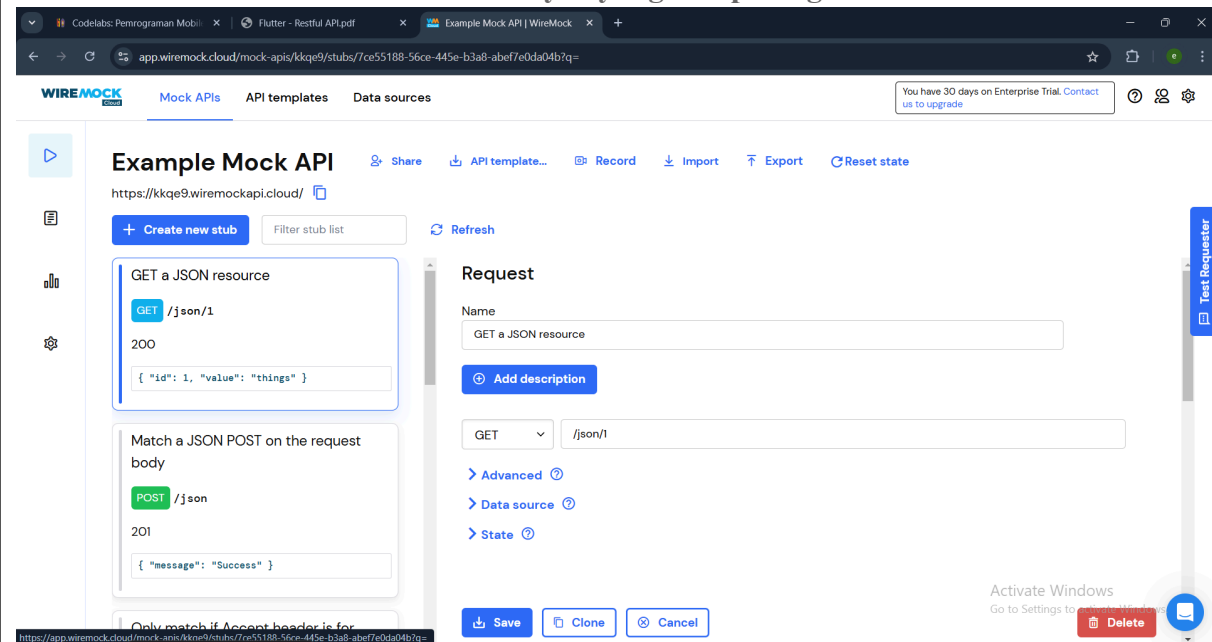
KELAS : TI-3C / 09

Praktikum 1, Designing an HTTP client and getting data

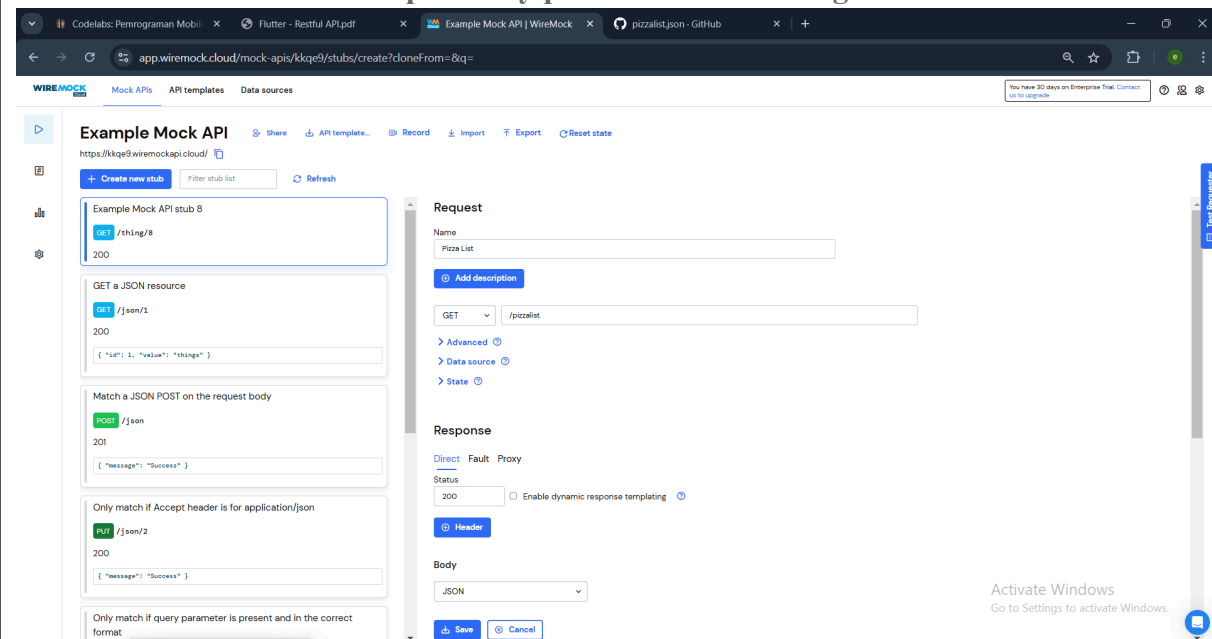
1. Mendaftarlah ke layanan Lab Mock di <https://app.wiremock.cloud/>. Bisa anda gunakan akun google untuk mendaftar. Jika berhasil mendaftar dan login, akan muncul seperti gambar berikut.

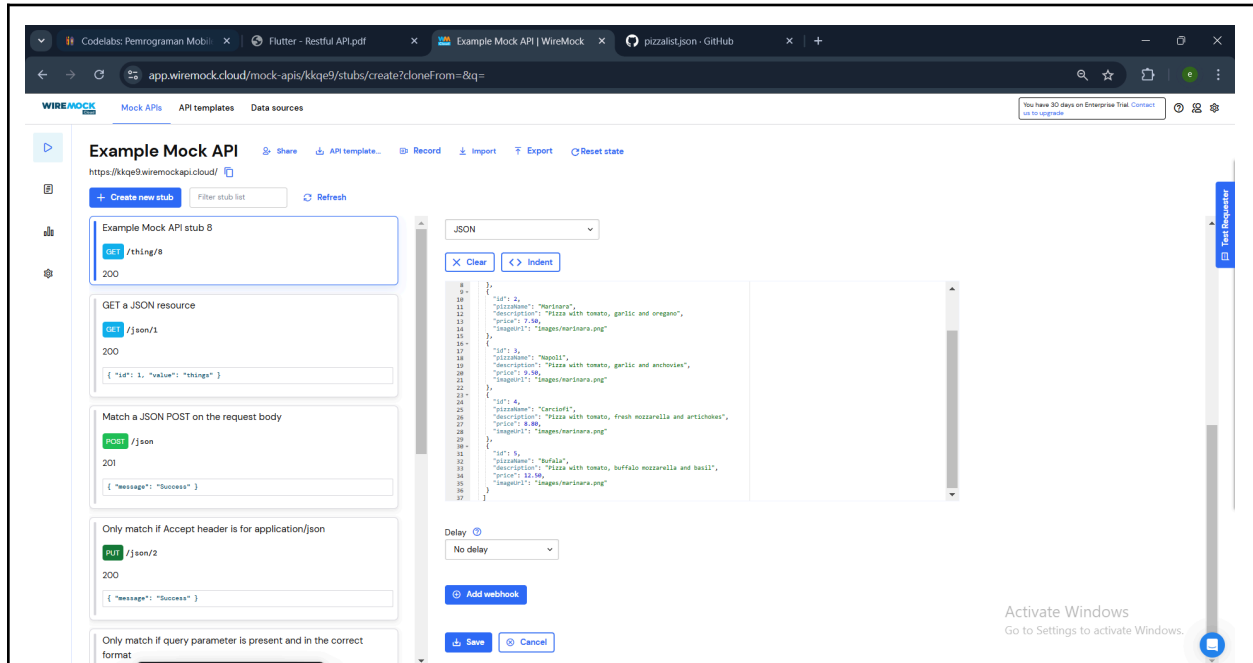


2. Di halaman dashboard, klik menu Stubs, kemudian klik entri pertama yaitu “GET a JSON resource”. Anda akan melihat layar yang mirip dengan berikut.

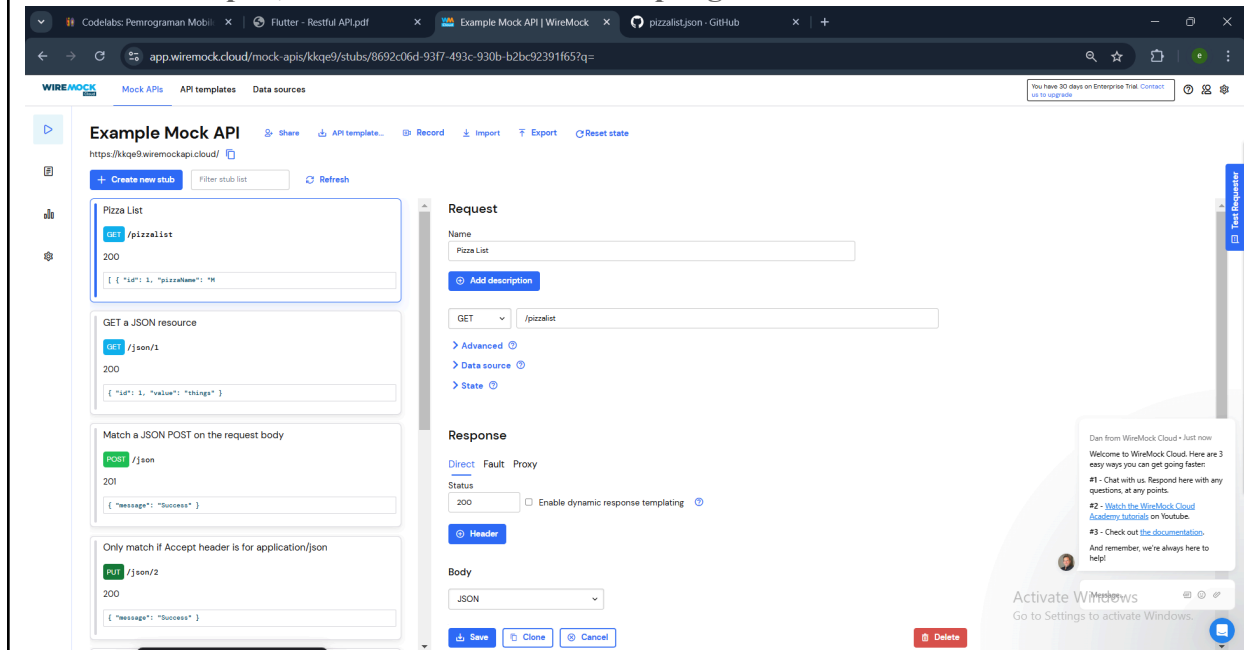


3. Klik “Create new stub”. Di kolom sebelah kanan, lengkapi data berikut. Namanya adalah “Pizza List”, kemudian pilih GET dan isi dengan “/pizzalist”. Kemudian, pada bagian Response, untuk status 200, kemudian pada Body pilih JSON sebagai formatnya dan isi konten JSON dari https://bit.ly/pizzalist. Perhatikan gambar berikut.

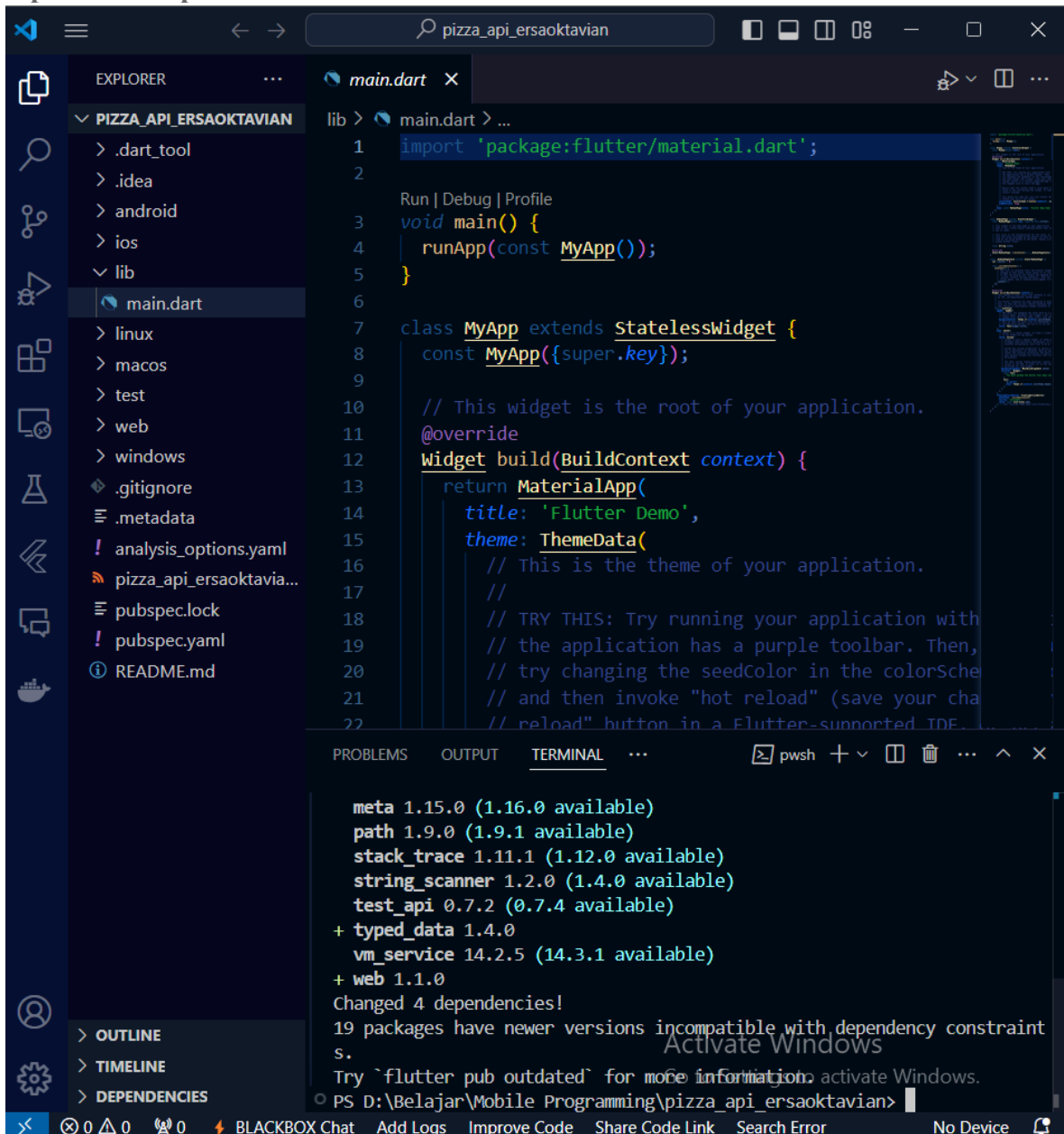




4. Tekan tombol SAVE di bagian bawah halaman untuk menyimpan Mock ini. Jika berhasil tersimpan, maka Mock API sudah siap digunakan.



5. Buatlah project flutter baru dengan nama `pizza_api_nama_anda`, tambahkan depedensi “http” melalui terminal.



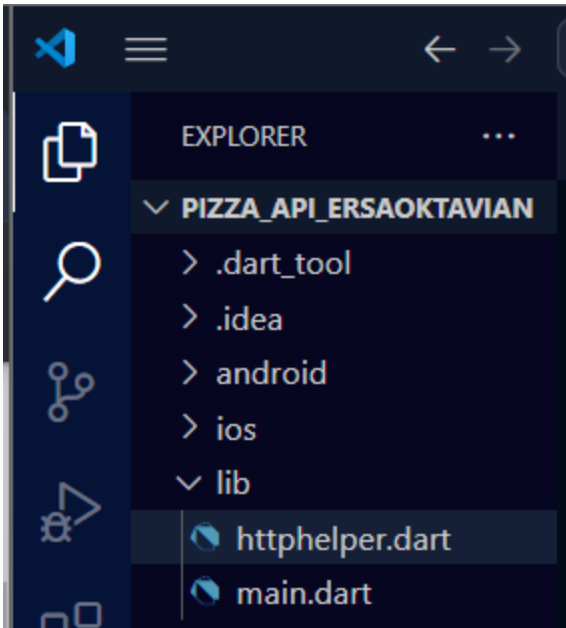
The screenshot shows an IDE with a project named `pizza_api_ersaoktavian`. The `main.dart` file is open, showing the following code:

```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15      theme: ThemeData(
16        // This is the theme of your application.
17        //
18        // TRY THIS: Try running your application with
19        // the application has a purple toolbar. Then,
20        // try changing the seedColor in the colorScheme
21        // and then invoke "hot reload" (save your changes
22        // and press ctrl+R) to see the effect.
23      ),
24    );
25  }
26 }
```

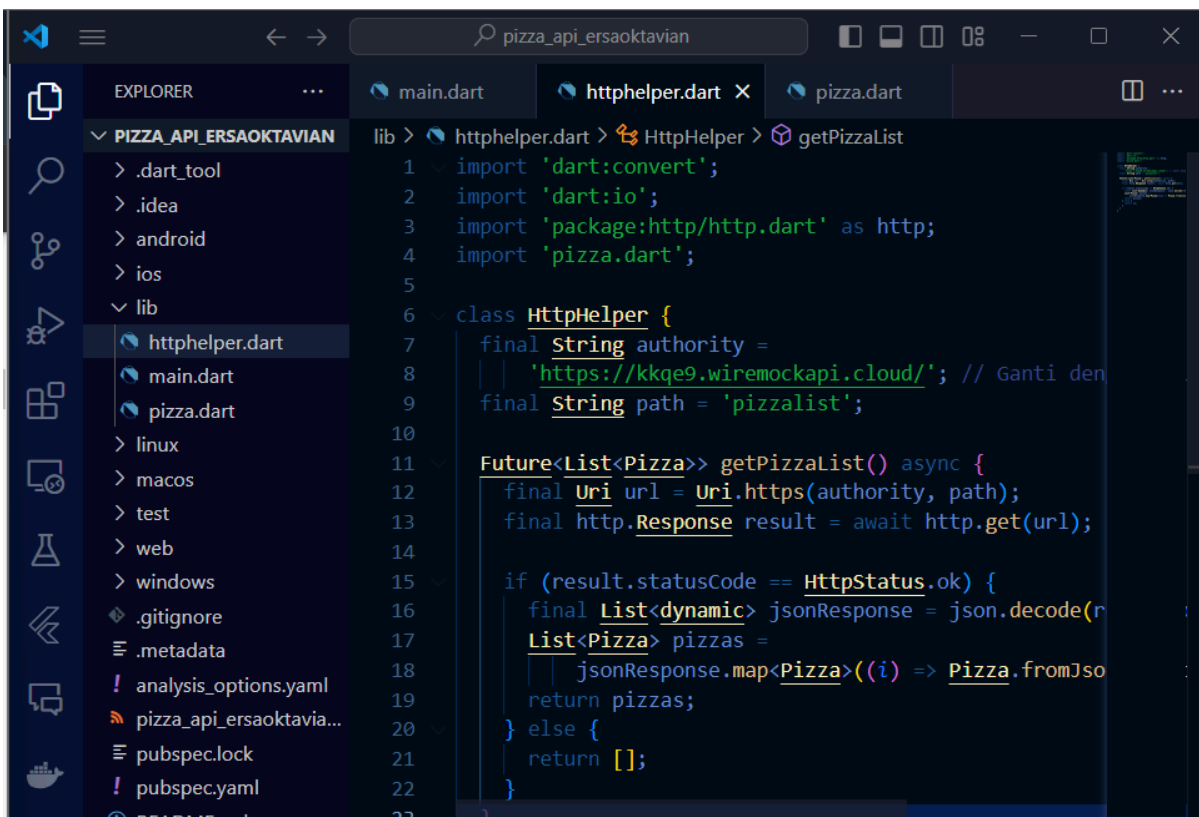
The terminal window shows the output of a Flutter command, listing dependencies and their versions:

```
meta 1.15.0 (1.16.0 available)
path 1.9.0 (1.9.1 available)
stack_trace 1.11.1 (1.12.0 available)
string_scanner 1.2.0 (1.4.0 available)
test_api 0.7.2 (0.7.4 available)
+ typed_data 1.4.0
  vm_service 14.2.5 (14.3.1 available)
+ web 1.1.0
Changed 4 dependencies!
19 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
PS D:\Belajar\Mobile Programming\pizza_api_ersaoktavian>
```

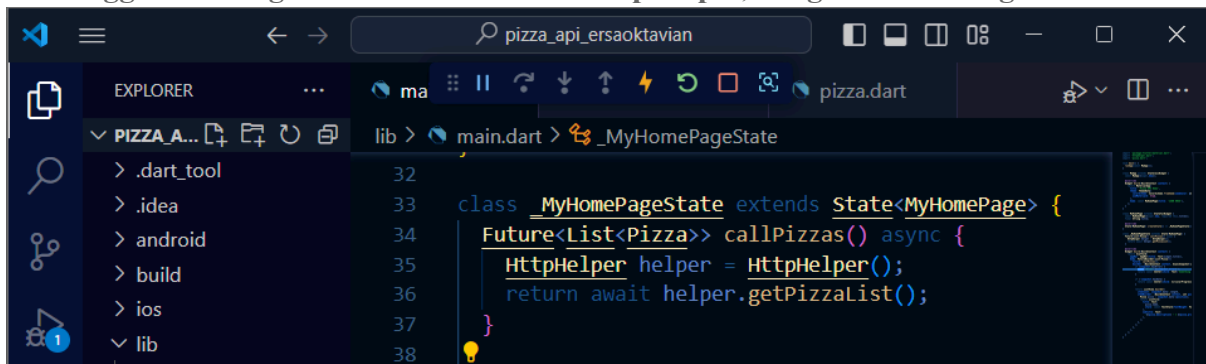
6. DI folder “lib” project anda, tambahkan file dengan nama “httphelper.dart”.



7. Isi httphelper.dart dengan kode berikut. Ubah “02z2g.mocklab.io” dengan URL Mock API anda.

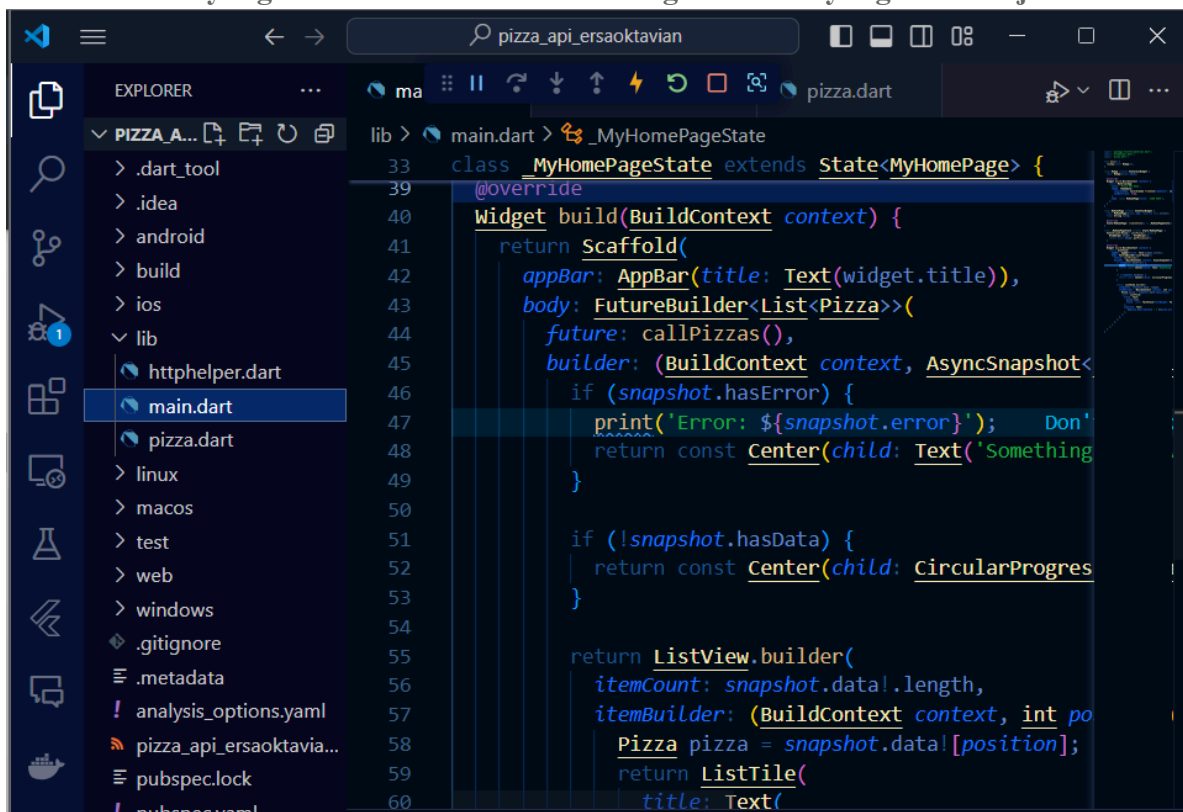


8. Di file “main.dart”, di class `_MyHomePageState`, tambahkan metode bernama “callPizzas”. Metode ini mengembalikan sebuah Future dari daftar objek Pizza dengan memanggil metode `getPizzaList` dari kelas `HttpHelper`, dengan kode sebagai berikut:



```
32
33 class _MyHomePageState extends State<MyHomePage> {
34   Future<List<Pizza>> callPizzas() async {
35     HttpHelper helper = HttpHelper();
36     return await helper.getPizzaList();
37   }
38 }
```

9. Pada metode `build` di class `_MyHomePageState`, di dalam body Scaffold, tambahkan `FutureBuilder` yang membuat `ListView` dari widget `ListTile` yang berisi objek `Pizza`:



```
33 class _MyHomePageState extends State<MyHomePage> {
39   @override
40   Widget build(BuildContext context) {
41     return Scaffold(
42       appBar: AppBar(title: Text(widget.title)),
43       body: FutureBuilder<List<Pizza>>(
44         future: callPizzas(),
45         builder: (BuildContext context, AsyncSnapshot<
46           List<Pizza>> snapshot) {
47           if (snapshot.hasError) {
48             print('Error: ${snapshot.error}');
49             return const Center(child: Text('Something
50               went wrong'));
51           }
52           if (!snapshot.hasData) {
53             return const Center(child: CircularProgressIndicator());
54           }
55           return ListView.builder(
56             itemCount: snapshot.data!.length,
57             itemBuilder: (BuildContext context, int position) {
58               Pizza pizza = snapshot.data![position];
59               return ListTile(
60                 title: Text(
```

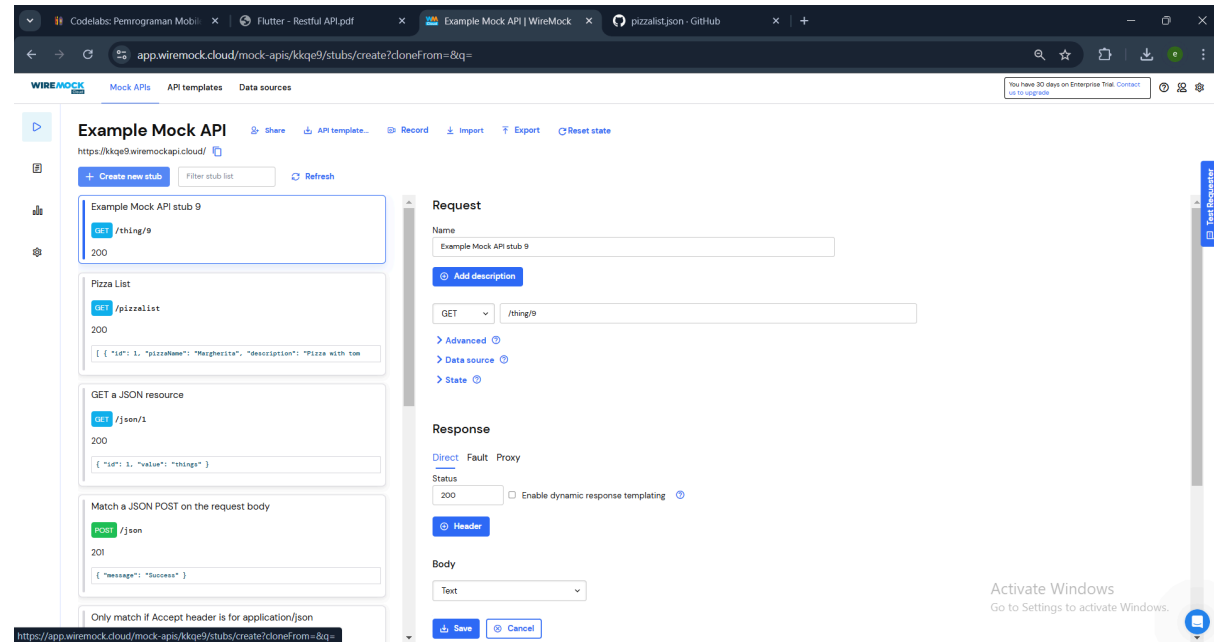
```
lib > main.dart > _MyHomePageState
33 class _MyHomePageState extends State<MyHomePage> {
40   Widget build(BuildContext context) {
53
54
55
56   return ListView.builder(
57     itemCount: snapshot.data!.length,
58     itemBuilder: (BuildContext context, int position) {
59       Pizza pizza = snapshot.data![position];
60       return ListTile(
61         title: Text(
62           pizza.name,
63           style: const TextStyle(fontWeight: FontWeight.bold),
64         ), // Text
65         subtitle: Text(
66           '${pizza.description} - € ${pizza.price}',
67         ), // Text
68       ); // ListTile
69     ); // ListView.builder
70   },
71 ); // FutureBuilder
72 ); // Scaffold
73 }
```


10. Output

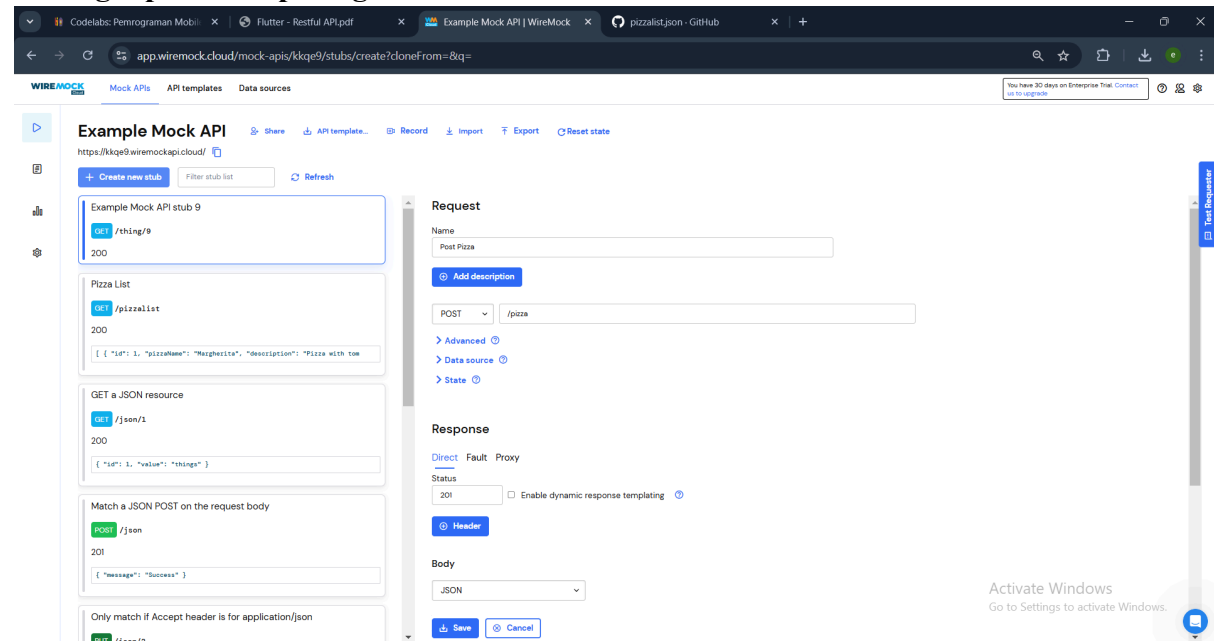


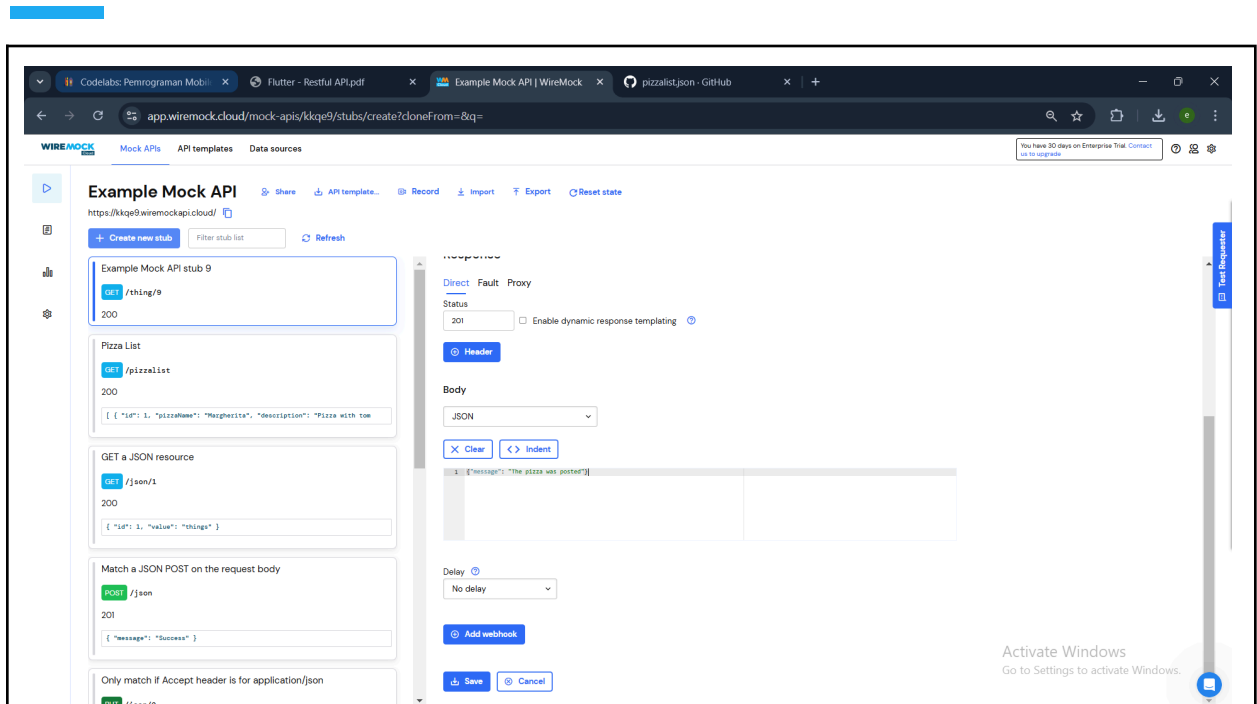
Praktikum 2. POST-ing data

1. Masuk ke layanan Lab Mock di <https://app.wiremock.cloud/> dan klik bagian Stubs, kemudian, buatlah stub baru.

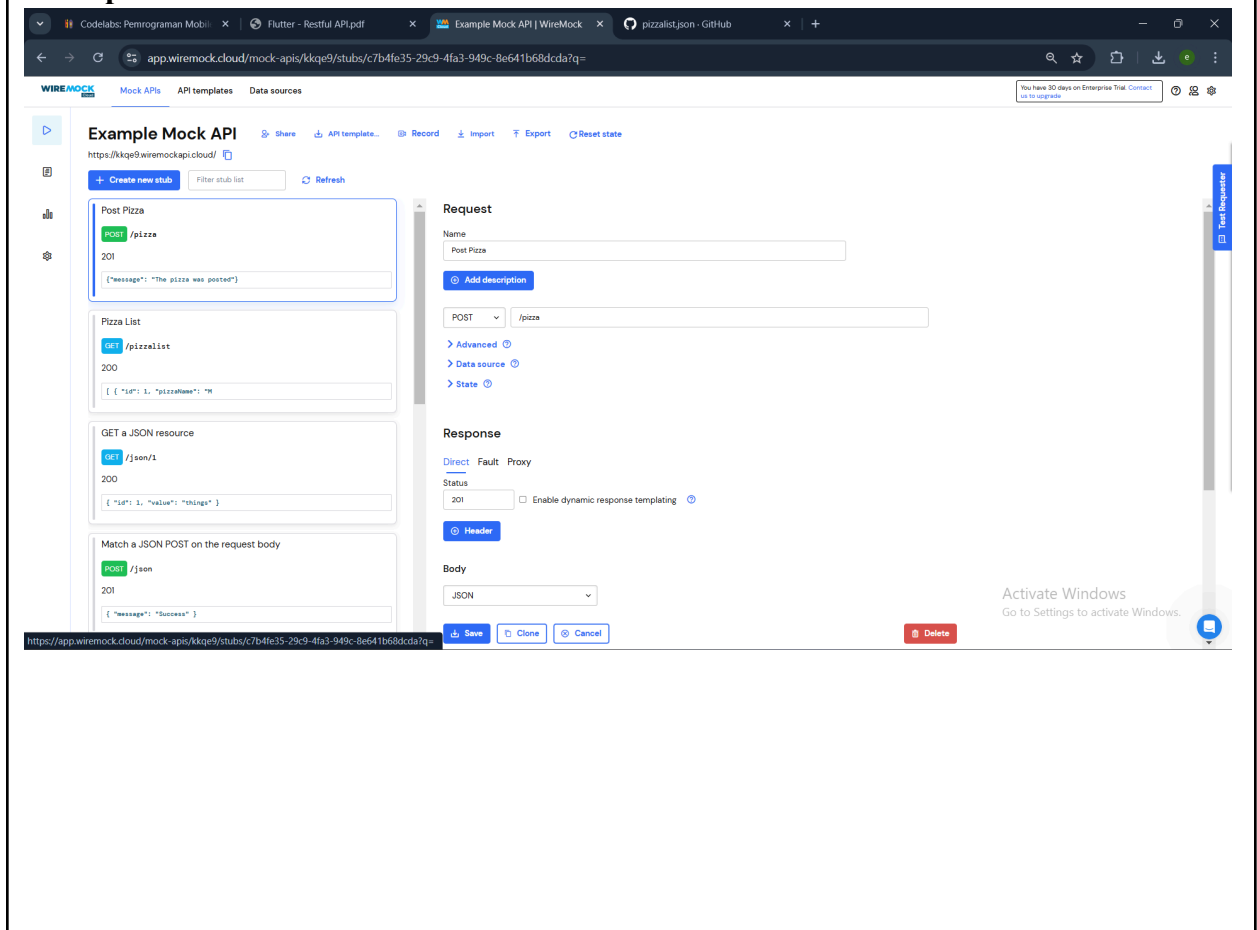


2. Lengkapi isian seperti gambar berikut:

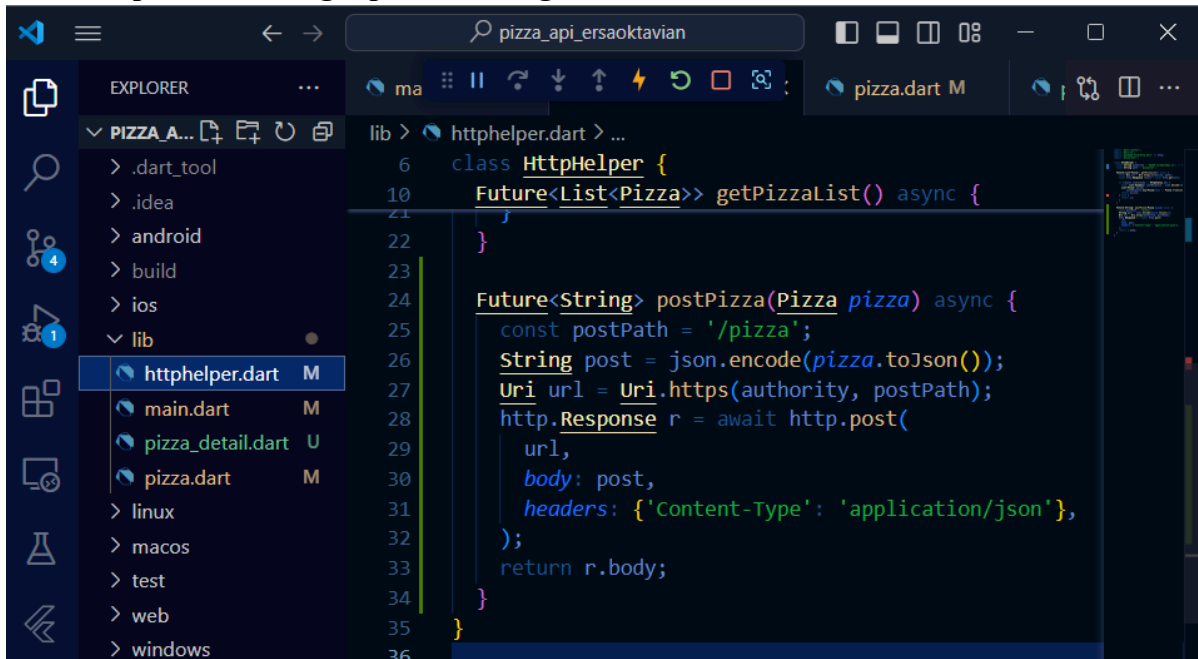




3. Simpan.

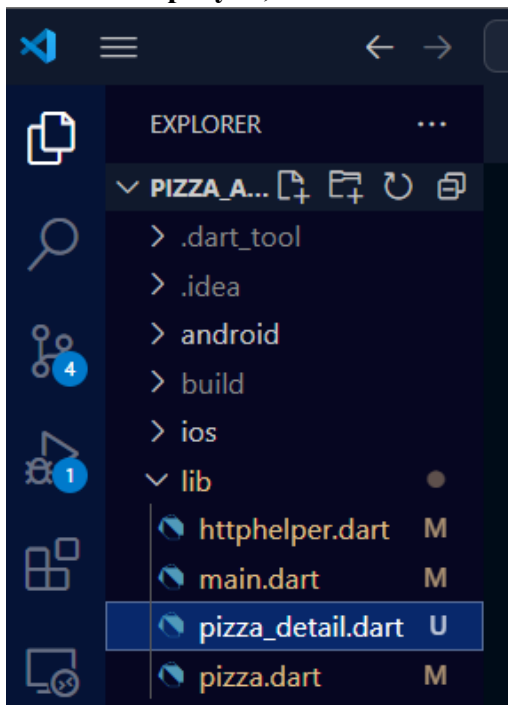


4. Di proyek Flutter, di file `httpHelper.dart`, di kelas `HttpHelper`, buat metode baru bernama `postPizza`, lengkapi kode sebagai berikut.

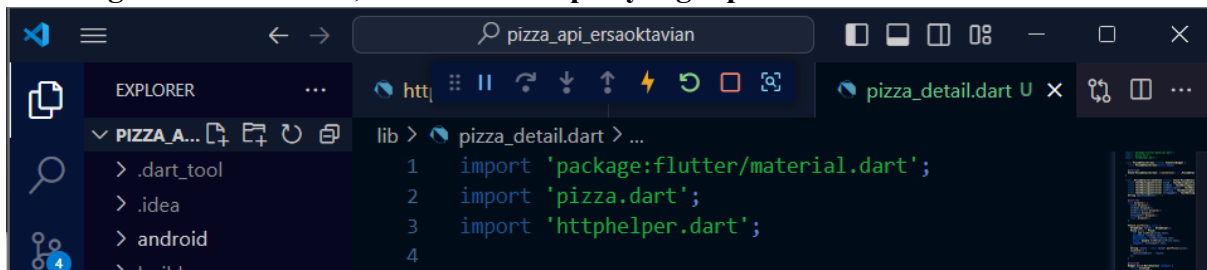


```
lib > httpHelper.dart > ...
6  class HttpHelper {
10  Future<List<Pizza>> getPizzaList() async {
21  }
22
23
24
25  Future<String> postPizza(Pizza pizza) async {
26    const postPath = '/pizza';
27    String post = json.encode(pizza.toJson());
28    Uri url = Uri.https(authority, postPath);
29    http.Response r = await http.post(
30      url,
31      body: post,
32      headers: {'Content-Type': 'application/json'},
33    );
34    return r.body;
35  }
36}
```

5. Di dalam proyek, buat sebuah file baru bernama `pizza_detail.dart`.

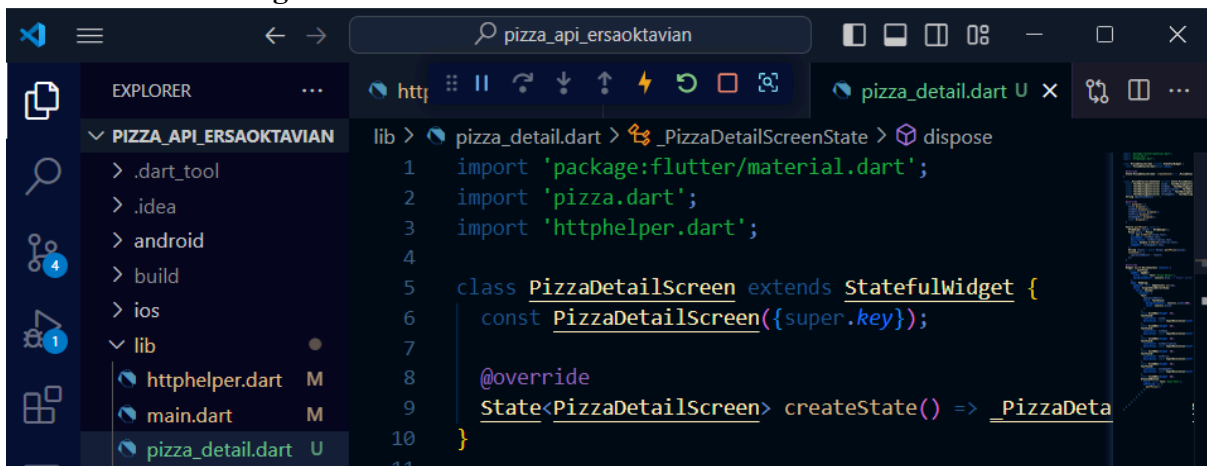


6. Di bagian atas file baru, tambahkan impor yang diperlukan.



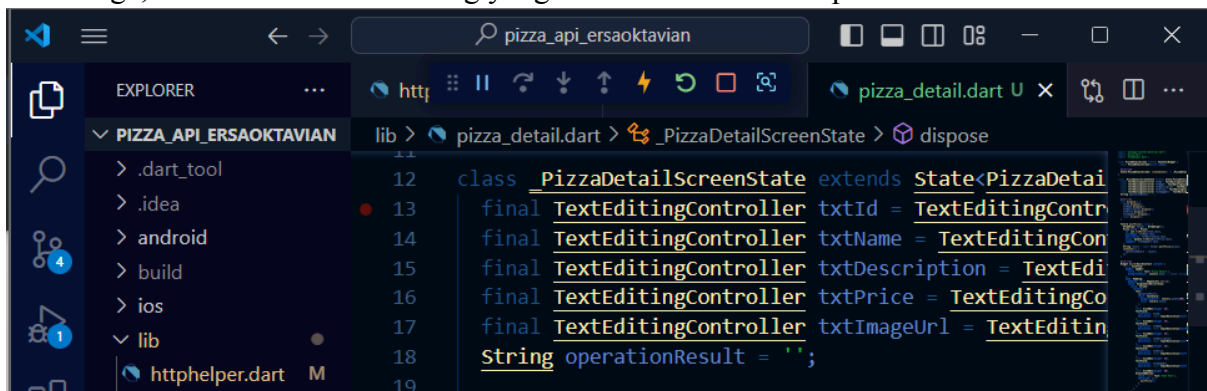
```
lib > pizza_detail.dart > ...
1  import 'package:flutter/material.dart';
2  import 'pizza.dart';
3  import 'http_helper.dart';
4
```

7. Buat StatefulWidget bernama PizzaDetailScreen



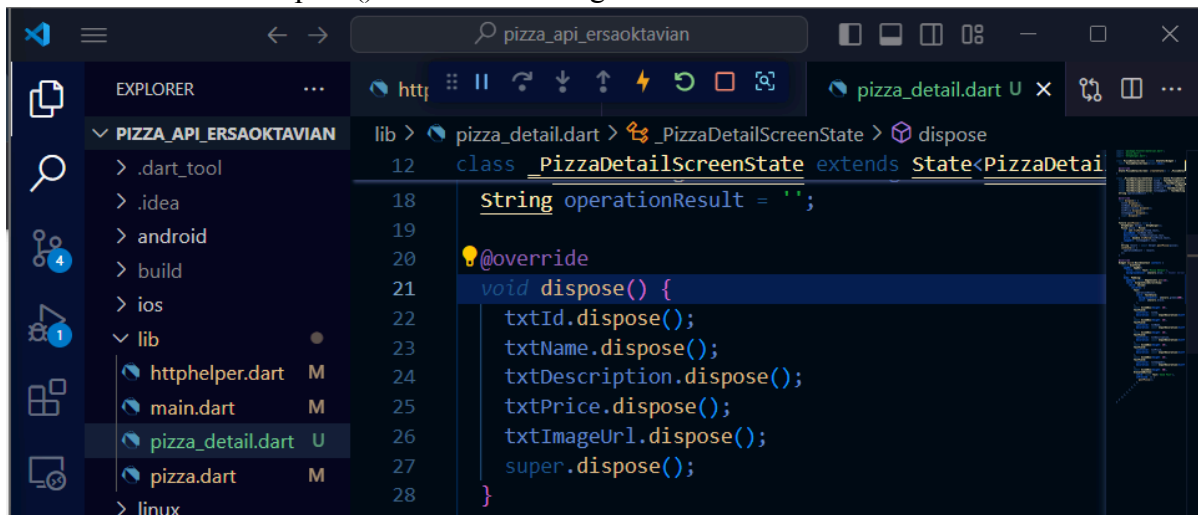
```
lib > pizza_detail.dart > _PizzaDetailScreenState > dispose
1  import 'package:flutter/material.dart';
2  import 'pizza.dart';
3  import 'http_helper.dart';
4
5  class PizzaDetailScreen extends StatefulWidget {
6    const PizzaDetailScreen({super.key});
7
8    @override
9    State<PizzaDetailScreen> createState() => _PizzaDetailScreenState;
10 }
11
```

8. Di bagian atas kelas _PizzaDetailScreenState, tambahkan lima widget TextEditingController. Widget ini akan berisi data untuk objek Pizza yang akan diposting nanti. Juga, tambahkan sebuah String yang akan berisi hasil dari permintaan POST.

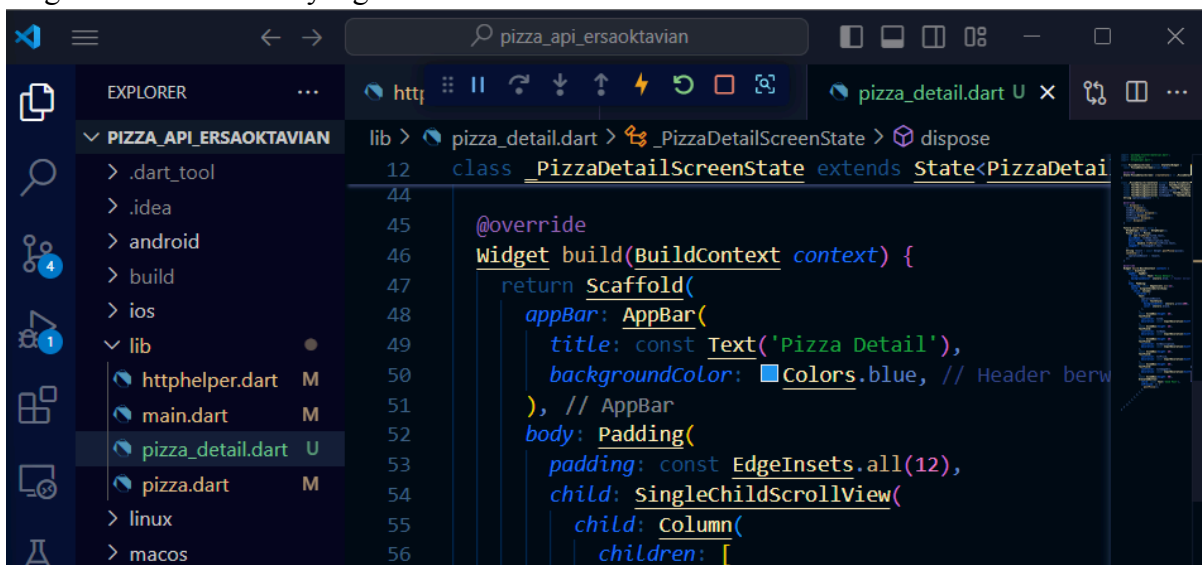


```
lib > pizza_detail.dart > _PizzaDetailScreenState > dispose
11
12 class _PizzaDetailScreenState extends State<PizzaDetailScreen> {
13   final TextEditingController txtId = TextEditingController();
14   final TextEditingController txtName = TextEditingController();
15   final TextEditingController txtDescription = TextEditingController();
16   final TextEditingController txtPrice = TextEditingController();
17   final TextEditingController txtImageUrl = TextEditingController();
18   String operationResult = '';
19 }
```

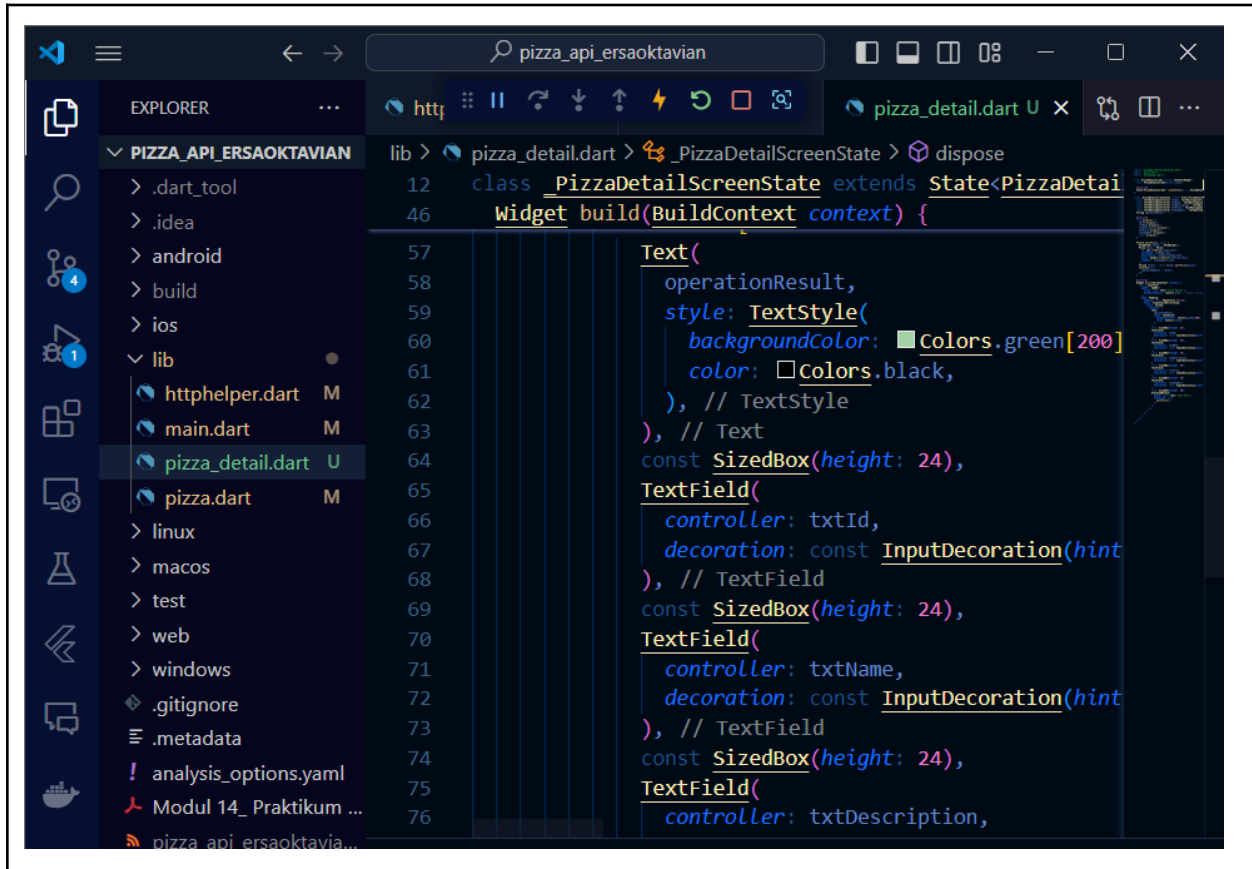
9. Override metode dispose() untuk membuang controllers

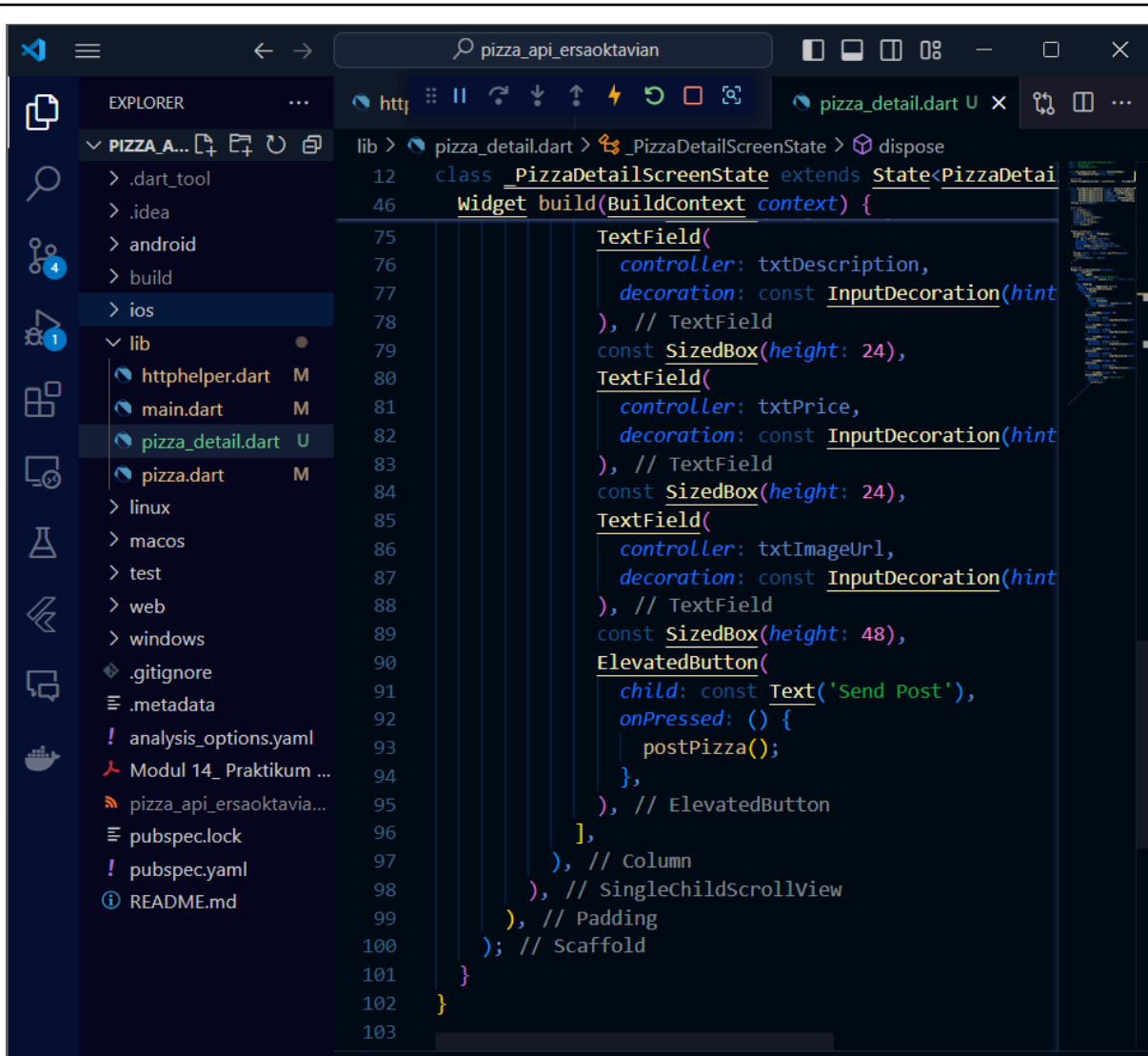


10. Dalam metode build() pada kelas, kita return sebuah Scaffold, yang AppBar-nya berisi Teks yang menyatakan “Detail Pizza” dan Body-nya berisi Padding dan SingleChildScrollView yang berisi Column.



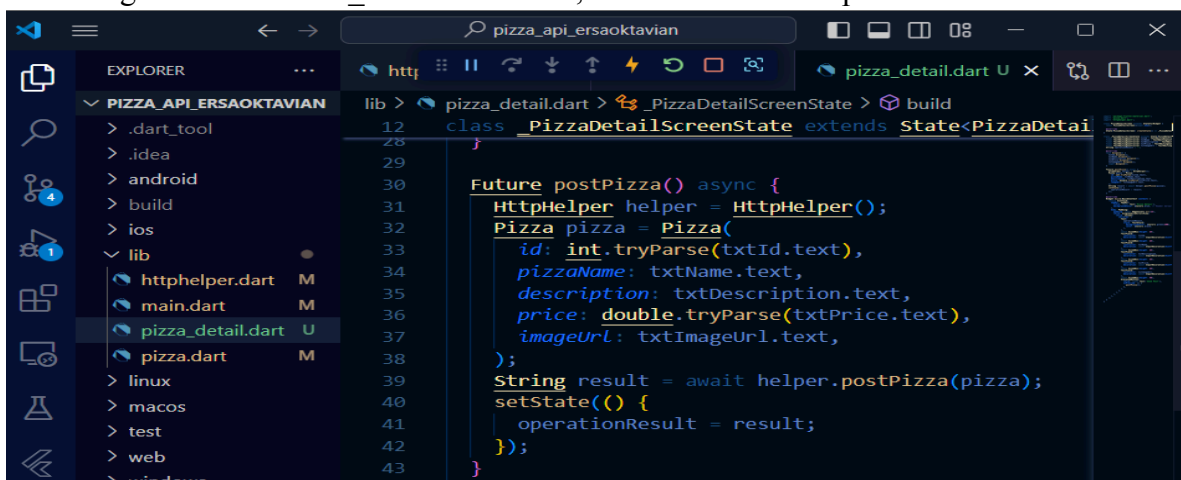
11. Untuk properti anak dari Column, tambahkan beberapa Text yang akan berisi hasil posting, lima TextFields, masing-masing terikat pada TextEditingController, dan sebuah ElevatedButton untuk menyelesaikan aksi POST (metode postPizza akan dibuat berikutnya). Juga, tambahkan SizedBox untuk memberi jarak pada widget di layar.





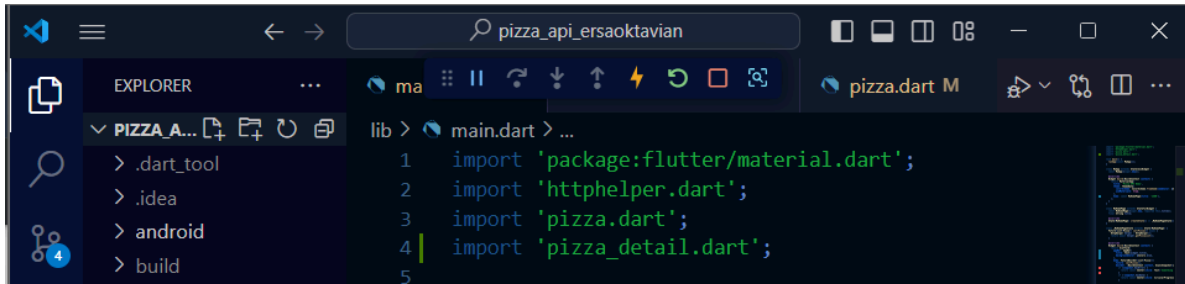
```
lib > pizza_detail.dart > _PizzaDetailScreenState > dispose
12 class _PizzaDetailScreenState extends State<PizzaDetail>
46   Widget build(BuildContext context) {
75     TextField(
76       controller: txtDescription,
77       decoration: const InputDecoration(hintText: 'Description'), // TextField
78     ), // TextField
79     const SizedBox(height: 24),
80     TextField(
81       controller: txtPrice,
82       decoration: const InputDecoration(hintText: 'Price'), // TextField
83     ), // TextField
84     const SizedBox(height: 24),
85     TextField(
86       controller: txtImageUrl,
87       decoration: const InputDecoration(hintText: 'Image URL'), // TextField
88     ), // TextField
89     const SizedBox(height: 48),
90     ElevatedButton(
91       child: const Text('Send Post'),
92       onPressed: () {
93         postPizza();
94       },
95     ), // ElevatedButton
96   ],
97 ), // Column
98 ), // SingleChildScrollView
99 ), // Padding
100 ); // Scaffold
101 }
102 }
103 }
```

12. Di bagian bawah kelas `_PizzaDetailState`, tambahkan metode `postPizza`.



```
lib > pizza_detail.dart > _PizzaDetailScreenState > build
12 class _PizzaDetailScreenState extends State<PizzaDetail>
28 {
29   // ...
30   Future postPizza() async {
31     HttpHelper helper = HttpHelper();
32     Pizza pizza = Pizza(
33       id: int.tryParse(txtId.text),
34       pizzaName: txtName.text,
35       description: txtDescription.text,
36       price: double.tryParse(txtPrice.text),
37       imageUrl: txtImageUrl.text,
38     );
39     String result = await helper.postPizza(pizza);
40     setState(() {
41       operationResult = result;
42     });
43   }
44 }
```

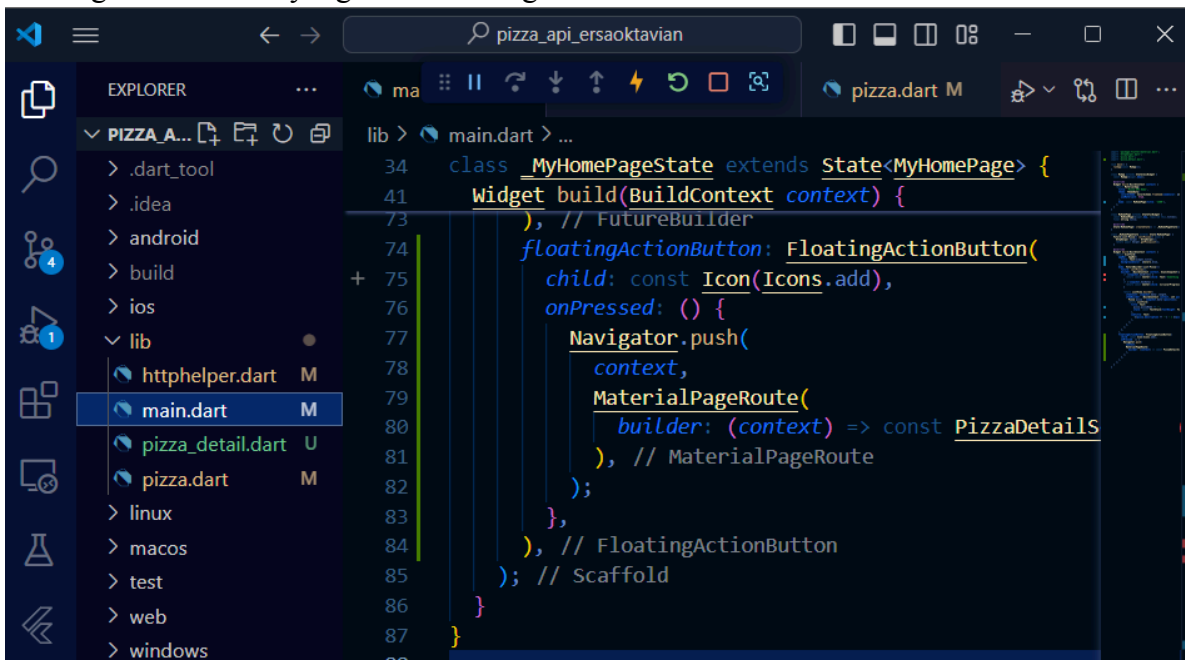

13. Di file main.dart, impor file pizza_detail.dart.



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure with folders like .dart_tool, .idea, android, and build. The main editor window shows the file lib/main.dart with the following code:

```
1 import 'package:flutter/material.dart';
2 import 'http_helper.dart';
3 import 'pizza.dart';
4 import 'pizza_detail.dart';
5
```

14. Di perancah metode build() dari kelas _MyHomePageState, tambahkan FloatingActionButton yang akan menavigasi ke rute PizzaDetail.

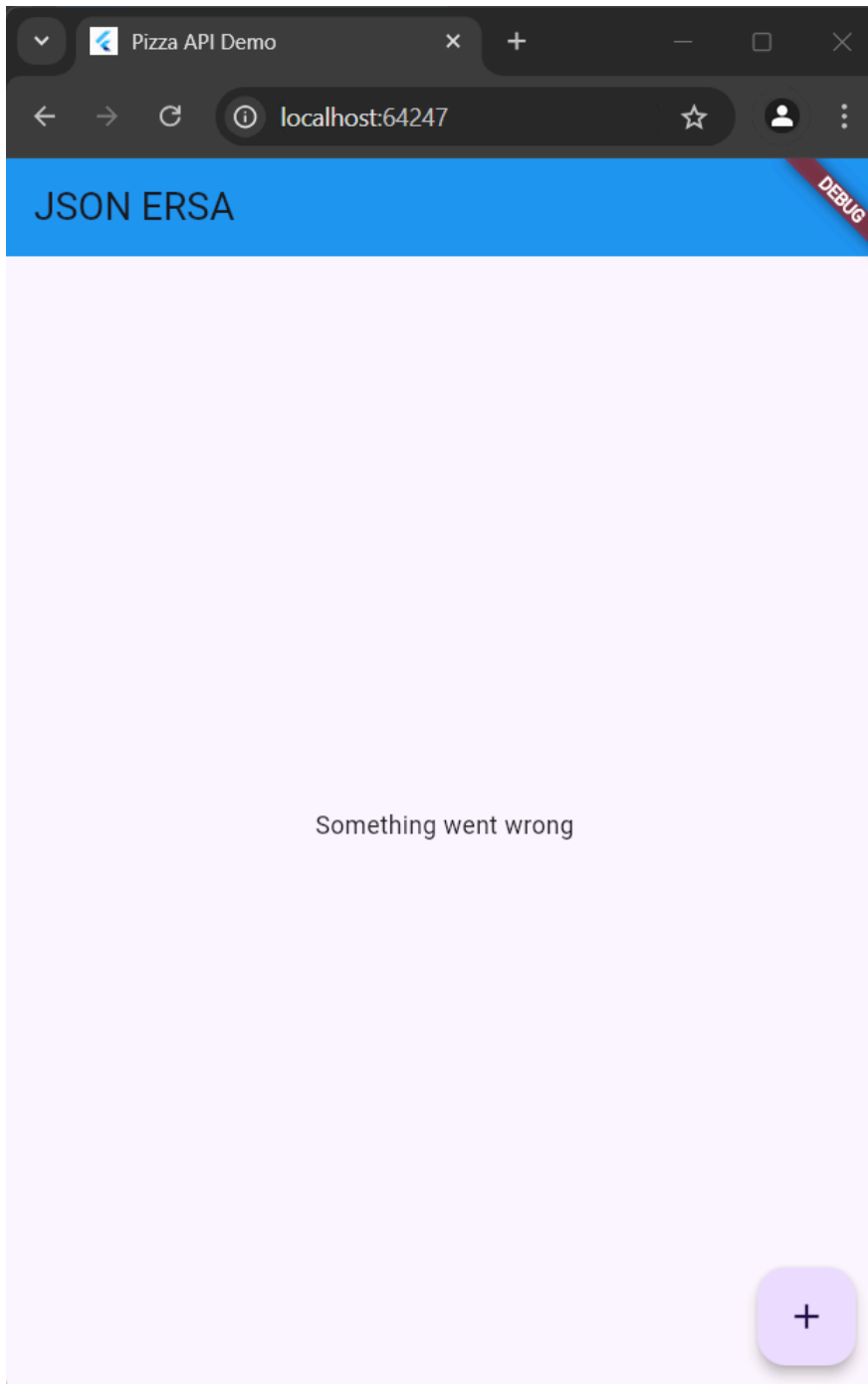


The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure with folders like .dart_tool, .idea, android, build, ios, lib, linux, macos, test, web, and windows. The main editor window shows the file lib/main.dart with the following code:

```
34 class _MyHomePageState extends State<MyHomePage> {
41   Widget build(BuildContext context) {
73     ), // FutureBuilder
74   },
75   floatingActionButton: FloatingActionButton(
76     child: const Icon(Icons.add),
77     onPressed: () {
78       Navigator.push(
79         context,
80         MaterialPageRoute(
81           builder: (context) => const PizzaDetails
82         ), // MaterialPageRoute
83       );
84     }, // FloatingActionButton
85   ); // Scaffold
86 }
87
```

15. Jalankan aplikasi. Pada layar utama, tekan FloatingActionButton untuk menavigasi ke rute PizzaDetail

Output Awal Lalu ditekan Floating Button maka akan diarahkan ke API dan insert Pizza



16. Tambahkan detail pizza di kolom teks dan tekan tombol Kirim Postingan. Anda akan melihat hasil yang berhasil, seperti yang ditunjukkan pada gambar berikut.
Output POST API dan Pizza Detail

