

Laporan Praktikum

Mobile Programming

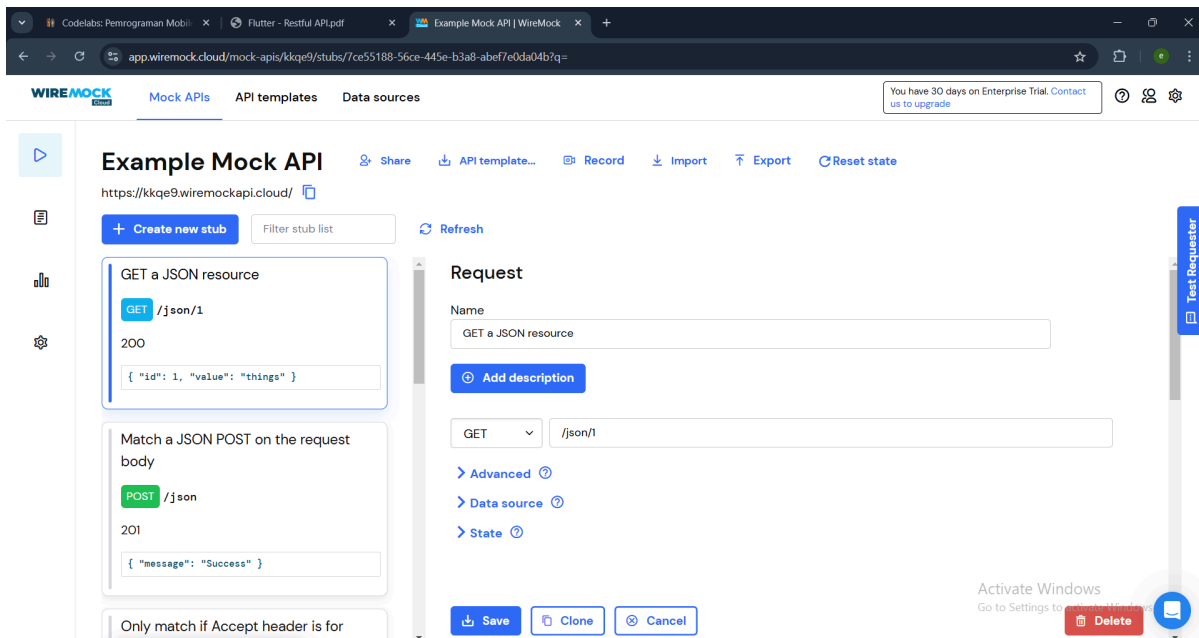
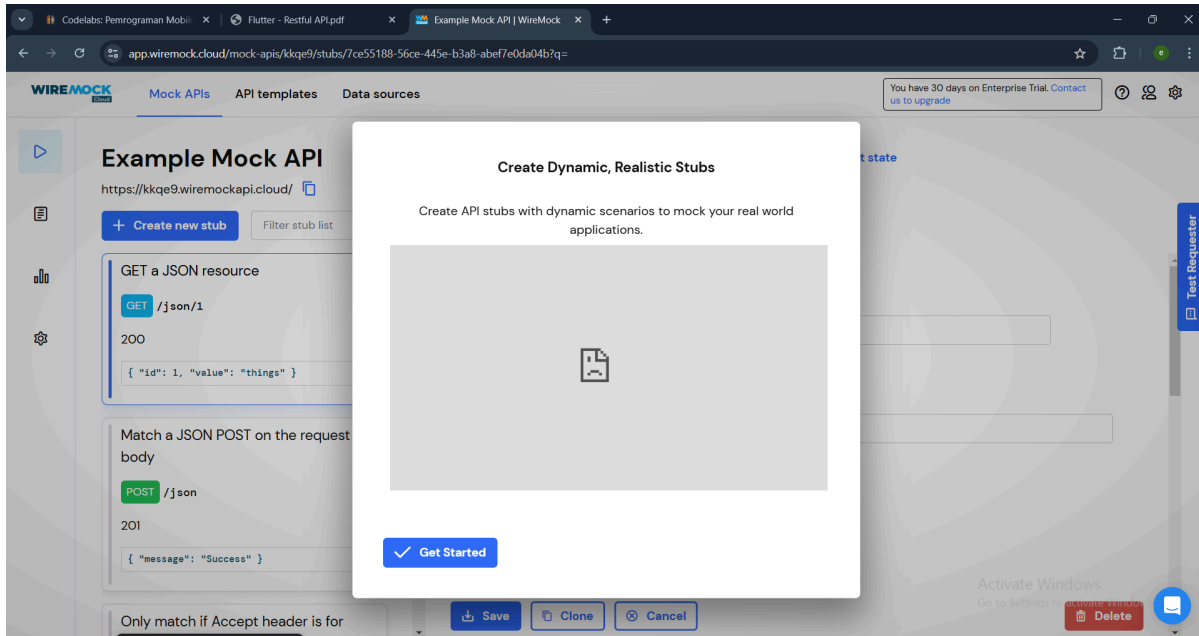
NAMA : Ersa Oktavian Ramadan

NIM : 2241720208

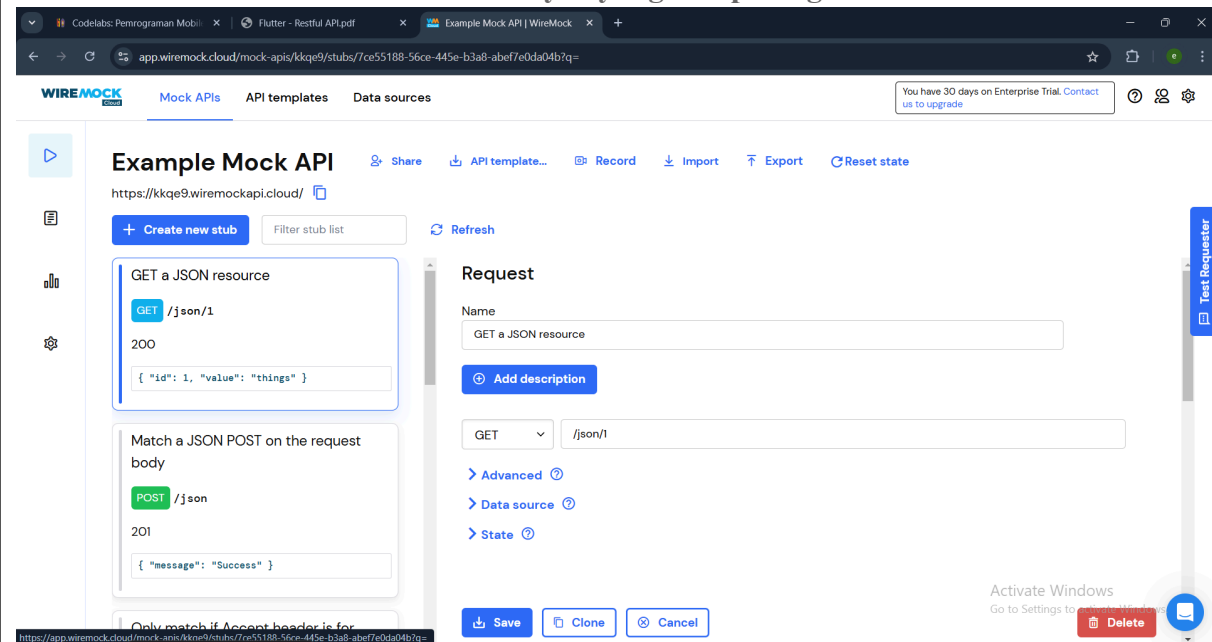
KELAS : TI-3C / 09

Praktikum 1, Designing an HTTP client and getting data

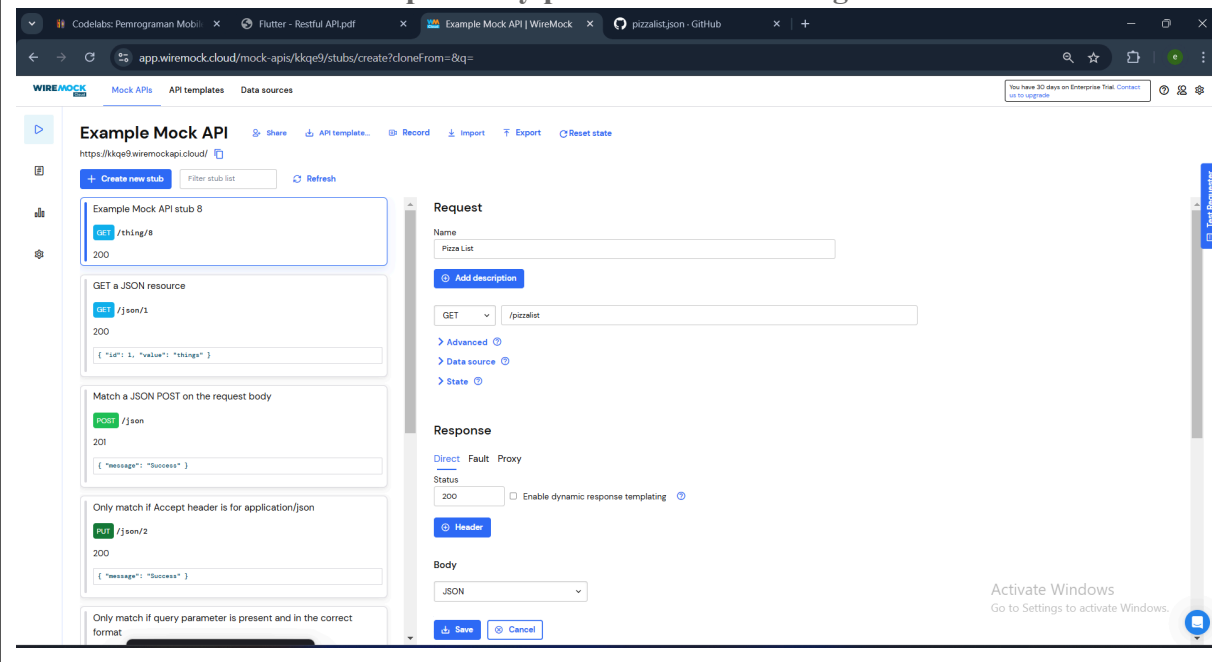
1. Mendaftarlah ke layanan Lab Mock di <https://app.wiremock.cloud/>. Bisa anda gunakan akun google untuk mendaftar. Jika berhasil mendaftar dan login, akan muncul seperti gambar berikut.

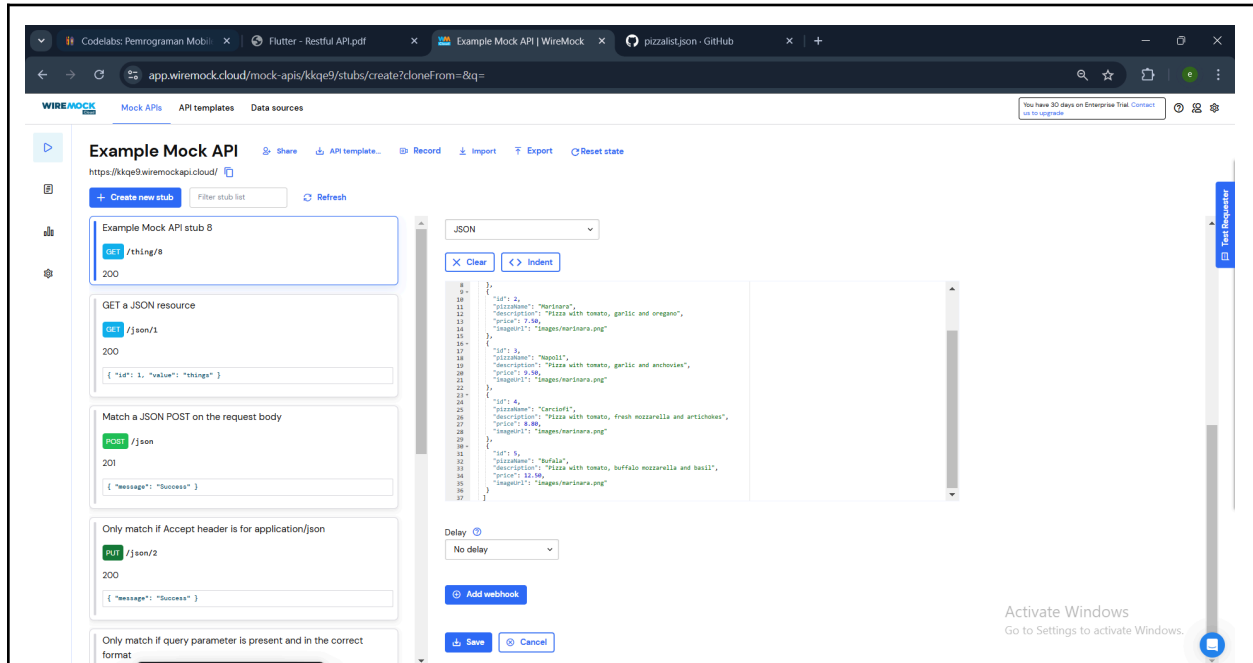


2. Di halaman dashboard, klik menu Stubs, kemudian klik entri pertama yaitu “GET a JSON resource”. Anda akan melihat layar yang mirip dengan berikut.

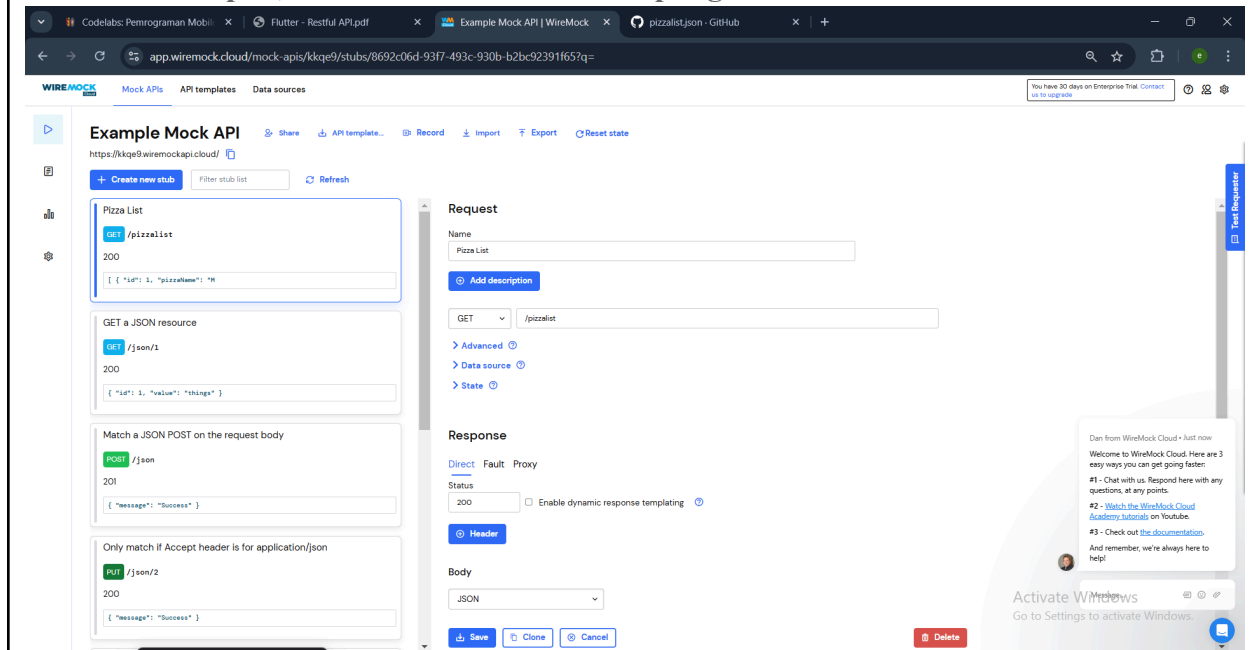


3. Klik “Create new stub”. Di kolom sebelah kanan, lengkapi data berikut. Namanya adalah “Pizza List”, kemudian pilih GET dan isi dengan “/pizzalist”. Kemudian, pada bagian Response, untuk status 200, kemudian pada Body pilih JSON sebagai formatnya dan isi konten JSON dari https://bit.ly/pizzalist. Perhatikan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

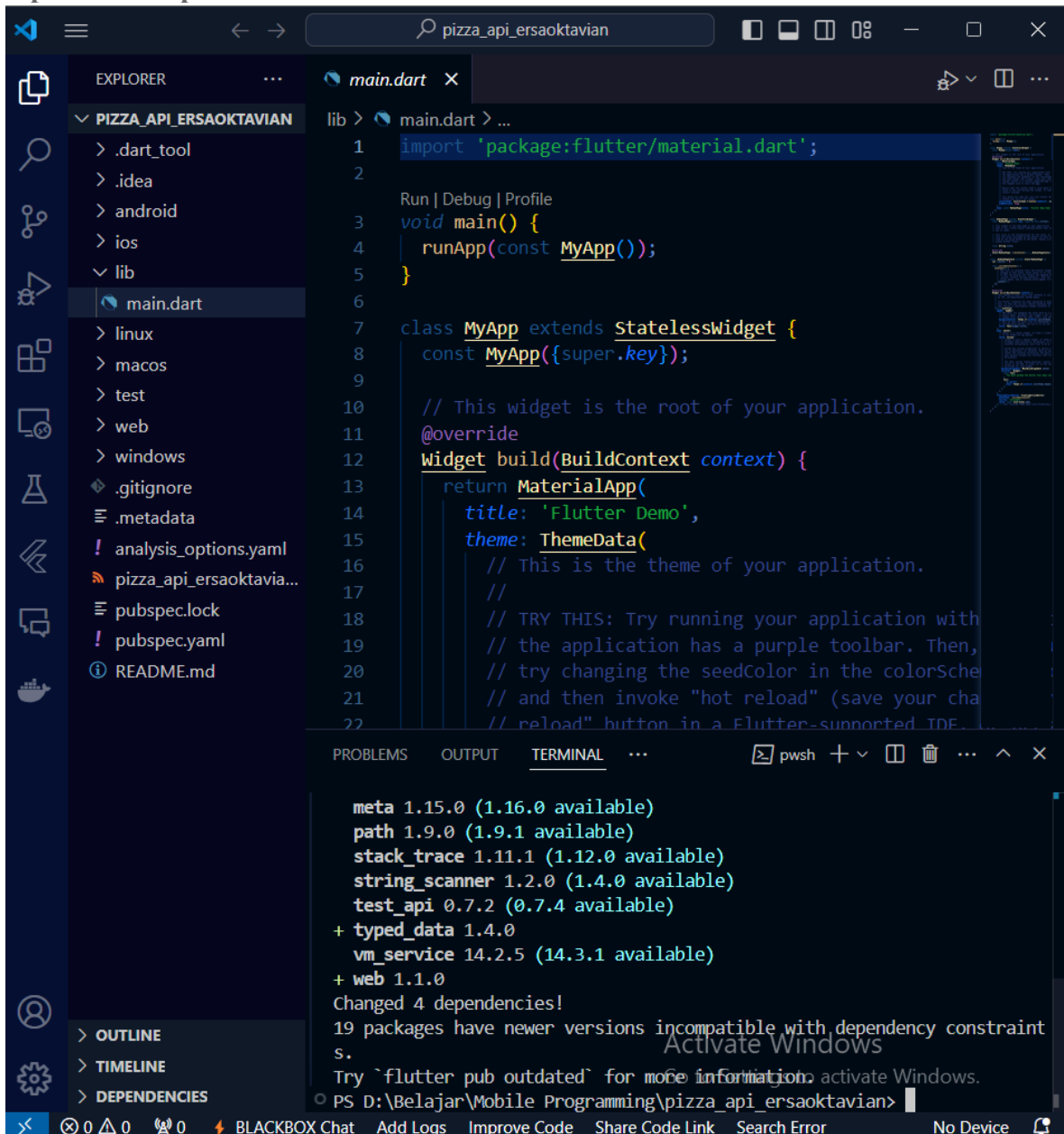




4. Tekan tombol SAVE di bagian bawah halaman untuk menyimpan Mock ini. Jika berhasil tersimpan, maka Mock API sudah siap digunakan.



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



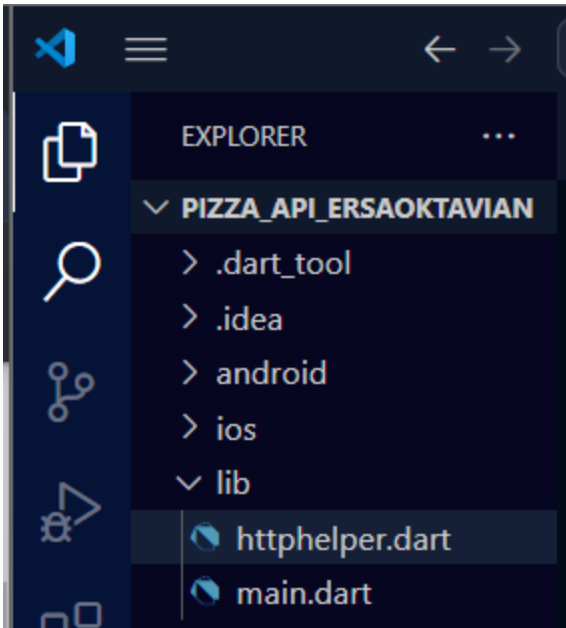
The screenshot shows an IDE with a project named `pizza_api_ersaoktavian`. The `main.dart` file is open, showing the following code:

```
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 }
```

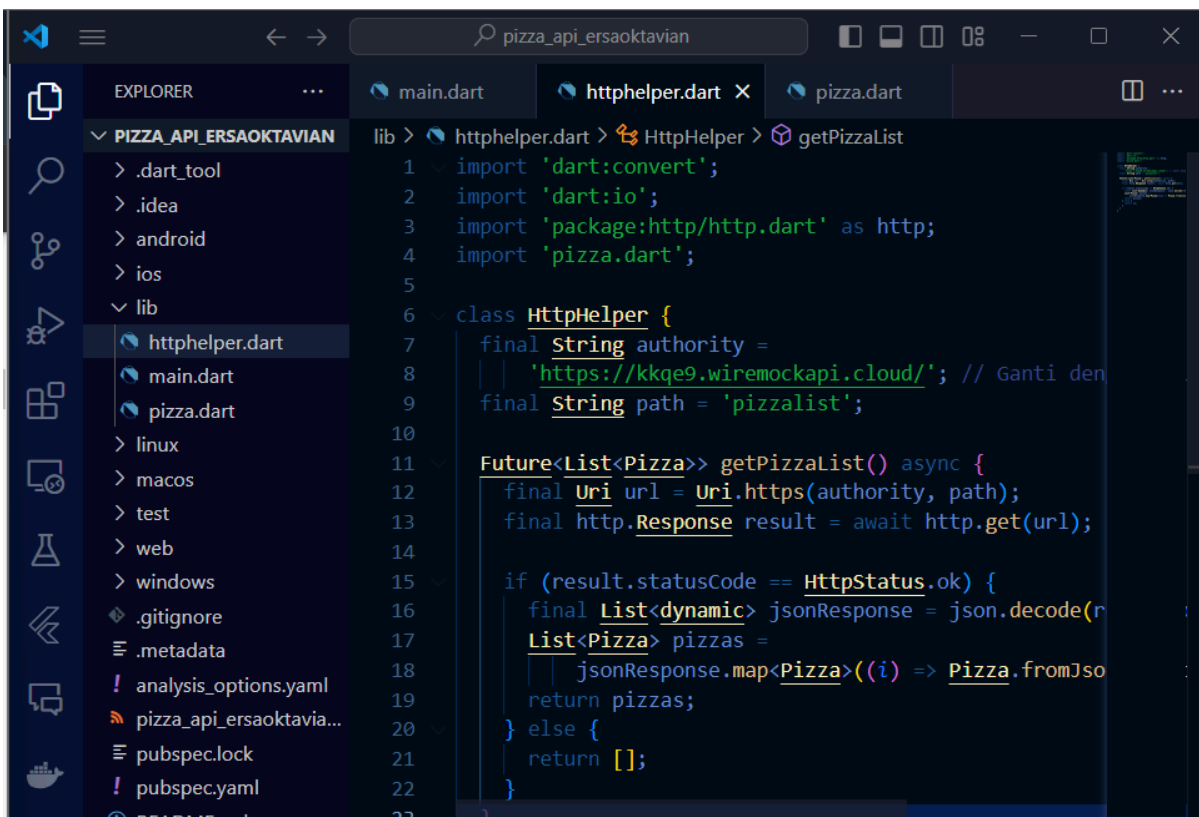
The terminal window at the bottom shows the output of a Flutter command, listing dependencies and their versions:

```
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>
```

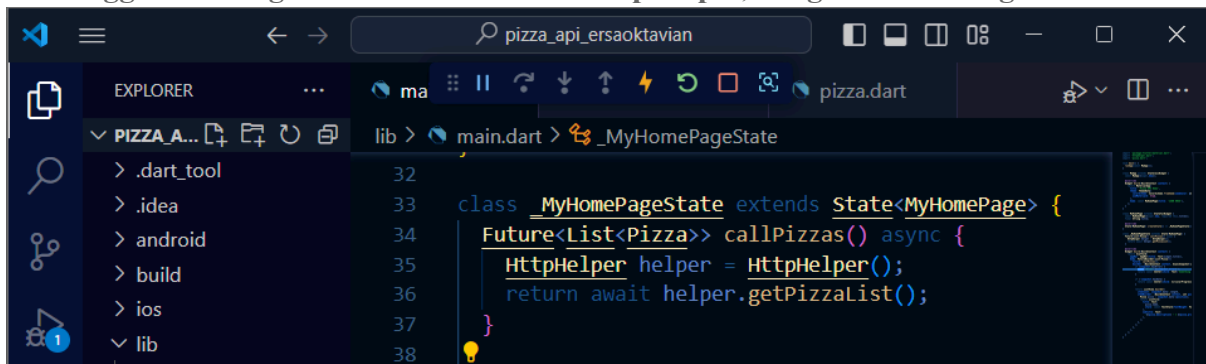
6. DI folder “lib” project anda, tambahkan file dengan nama “httphelper.dart”.



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



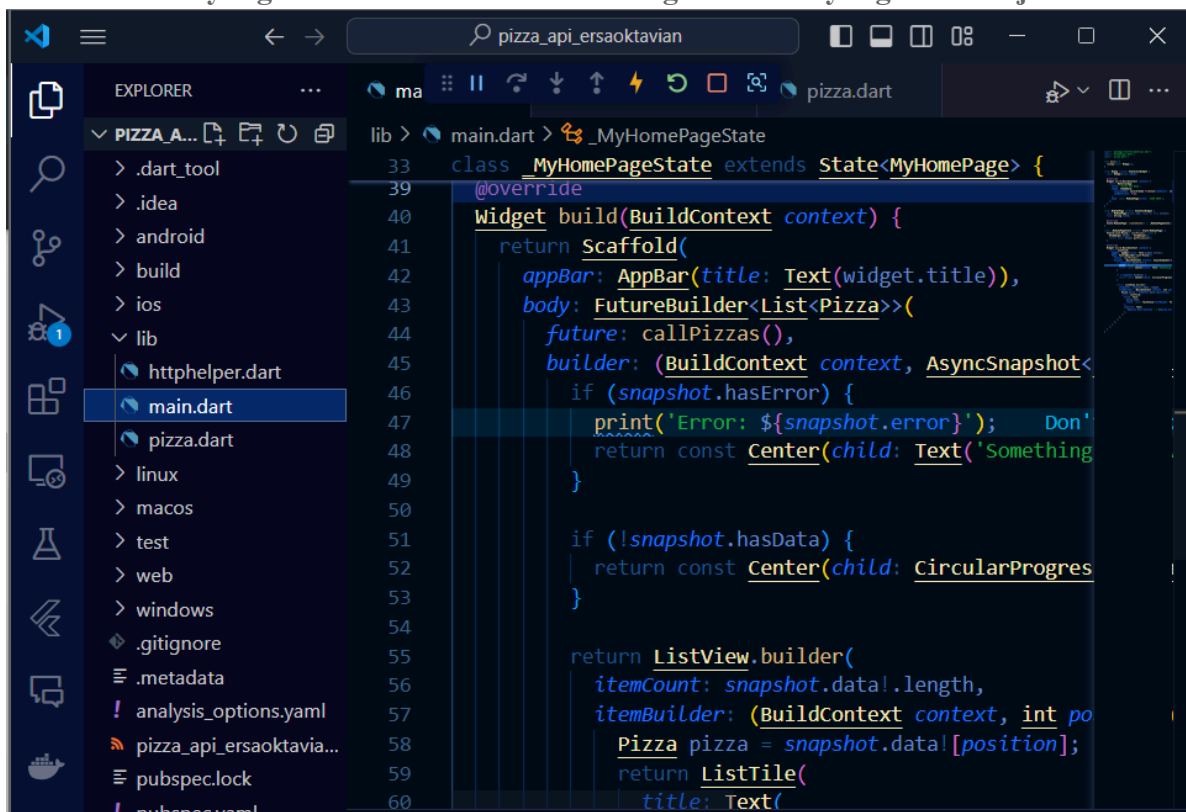
8. Di file “main.dart”, di class `_MyHomePageState`, tambahkan metode bernama “callPizzas”. Metode ini mengembalikan sebuah Future dari daftar objek Pizza dengan memanggil metode `getPizzaList` dari kelas `HttpHelper`, dengan kode sebagai berikut:



The screenshot shows the Visual Studio Code editor with the file `main.dart` open. The Explorer panel on the left shows the project structure with folders like `.dart_tool`, `.idea`, `android`, `build`, `ios`, and `lib`. The `lib` folder is expanded, showing `httphelper.dart`, `main.dart`, and `pizza.dart`. The `main.dart` file is selected, and the editor shows the `_MyHomePageState` class. The `callPizzas` method is being added to the class.

```
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`:



The screenshot shows the Visual Studio Code editor with the file `main.dart` open. The Explorer panel on the left shows the project structure with folders like `.dart_tool`, `.idea`, `android`, `build`, `ios`, and `lib`. The `lib` folder is expanded, showing `httphelper.dart`, `main.dart`, and `pizza.dart`. The `main.dart` file is selected, and the editor shows the `_MyHomePageState` class. The `build` method is being added to the class.

```
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           if (snapshot.hasError) {
47             print('Error: ${snapshot.error}'); Don't
48             return const Center(child: Text('Something
49           })
50         if (!snapshot.hasData) {
51           return const Center(child: CircularProgressIndicator());
52         }
53         return ListView.builder(
54           itemCount: snapshot.data!.length,
55           itemBuilder: (BuildContext context, int position) {
56             Pizza pizza = snapshot.data![position];
57             return ListTile(
58               title: Text(
59               )
60             )
61           )
62         )
63       )
64     );
65   }
66 }
```

```
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



—