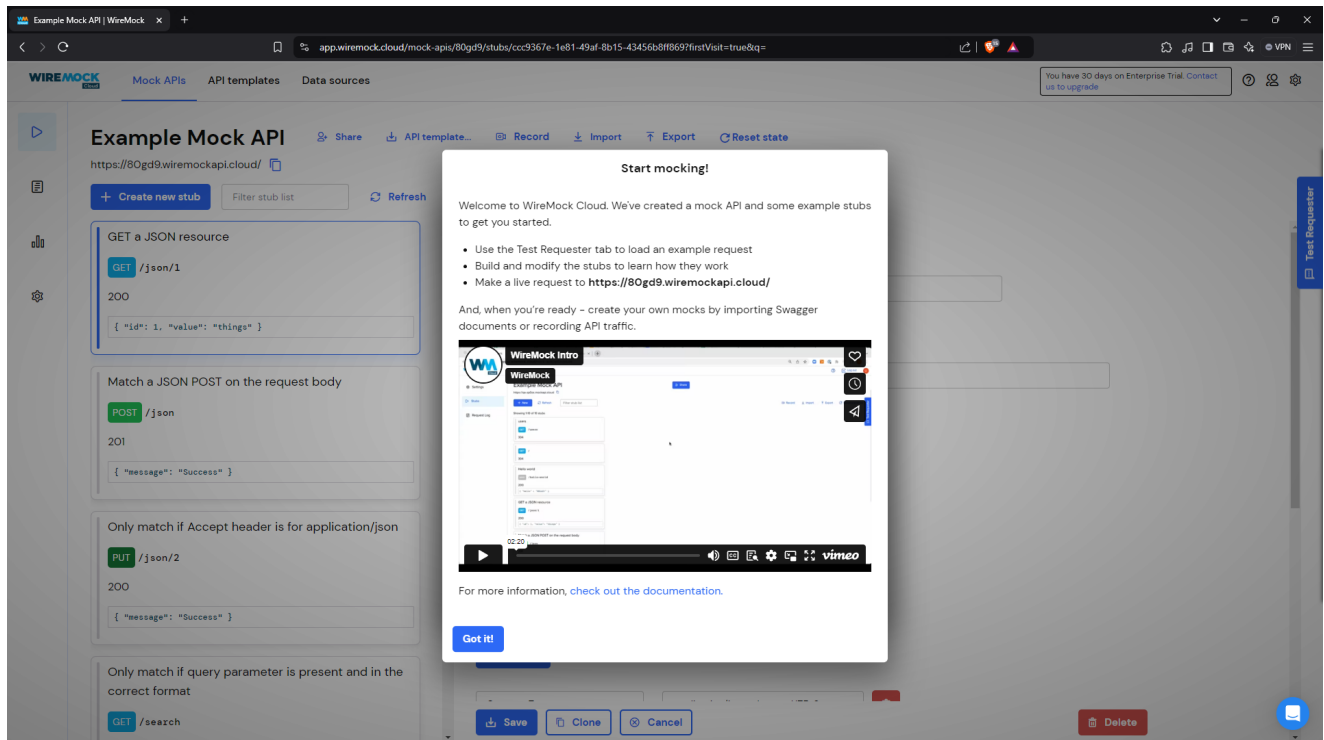


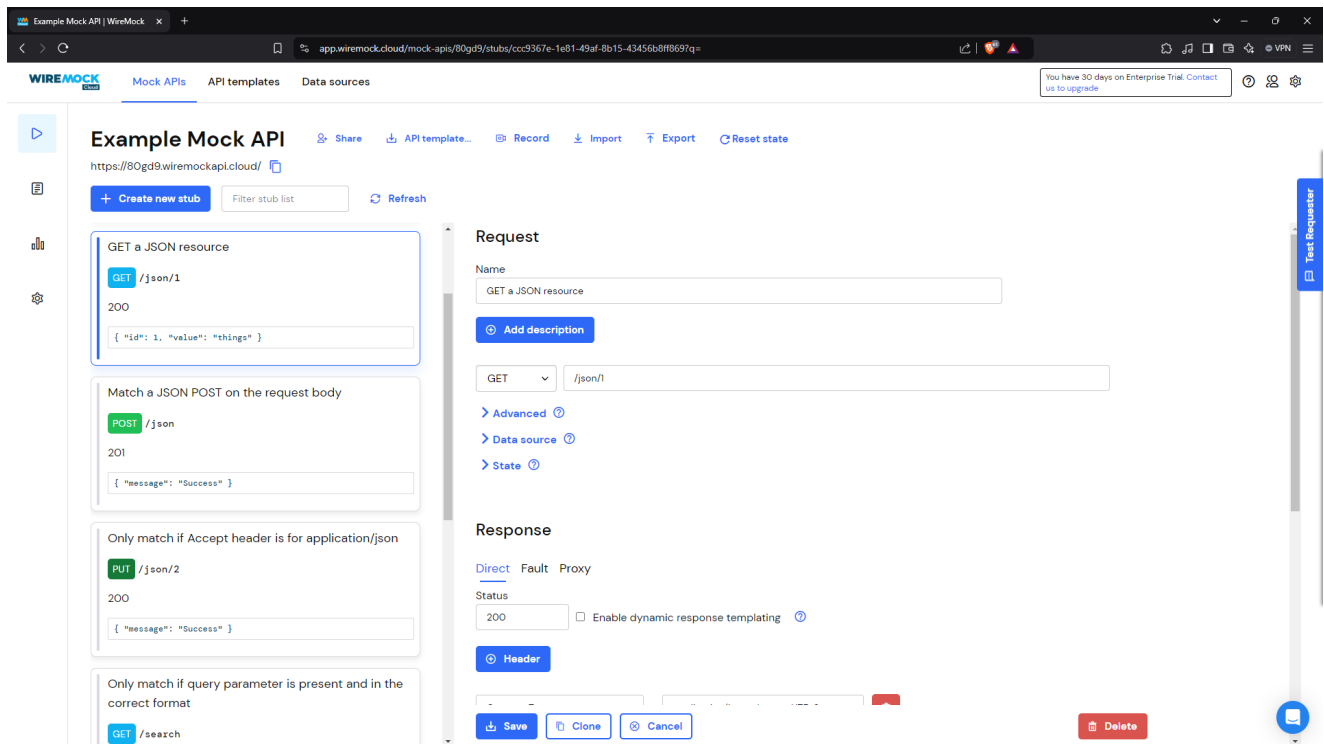
#14 | RESTFUL API

A. Praktikum 1, Designing an HTTP client and getting data

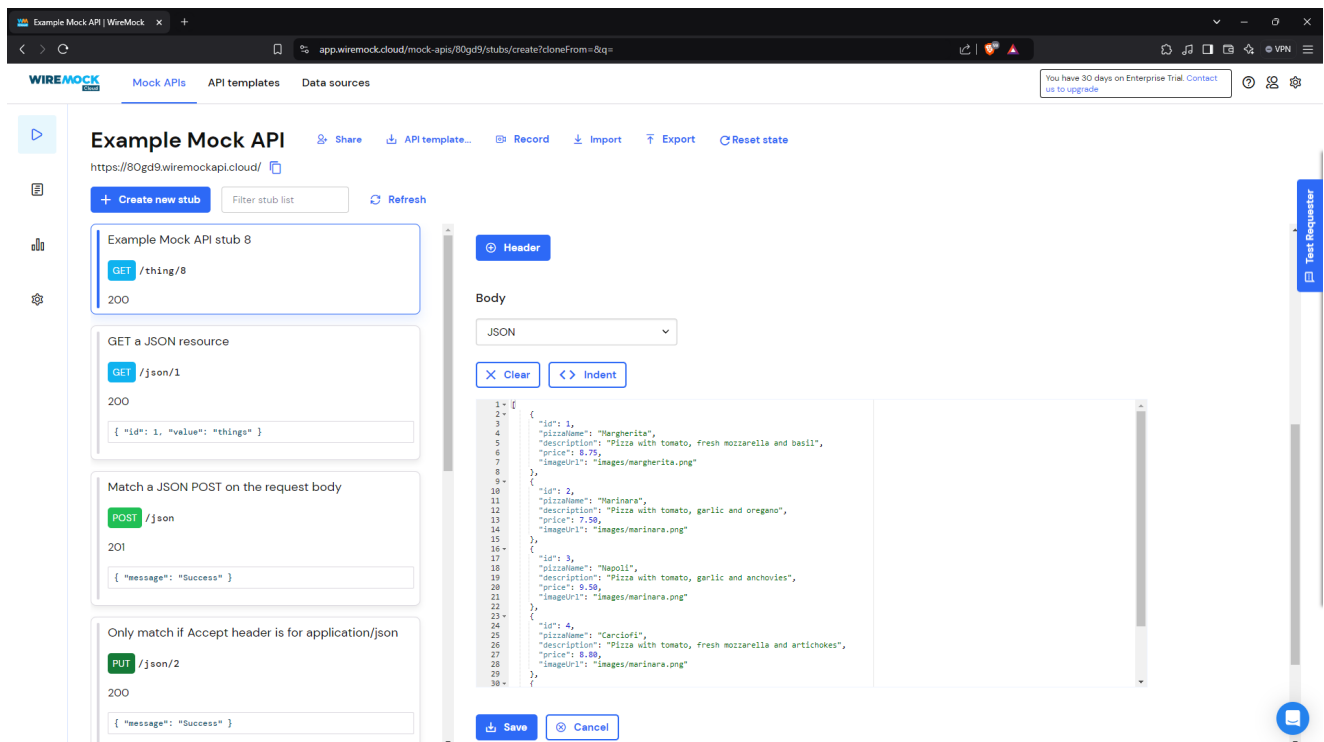
Mendaftarlah ke layanan Lab Mock di <https://app.wiremock.cloud/>. Bisa anda gunakan akun google untuk mendaftar. Jika berhasil mendaftar dan login, akan muncul seperti gambar berikut.



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.

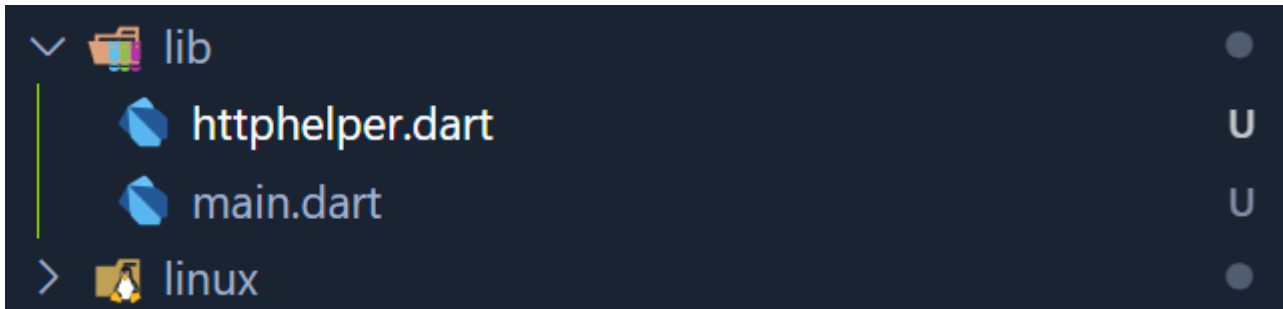


4. Tekan tombol SAVE di bagian bawah halaman untuk menyimpan Mock ini. Jika berhasil tersimpan, maka Mock API sudah siap digunakan.

5. Buatlah project flutter baru dengan nama pizza_api_nama_anda, tambahkan dependensi "http" melalui terminal.

```
flutter pub add http
```

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.

```
import 'dart:io';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'pizza.dart';

class HttpHelper {
  final String authority = '80gd9.wiremockapi.cloud';
  final String path = 'pizzalist';

  Future<List<Pizza>> getPizzaList() async {
    final Uri uri = Uri.https(authority, path);
    final http.Response result = await http.get(uri);

    if (result.statusCode == HttpStatus.ok) {
      final jsonResponse = json.decode(result.body);
      List<Pizza> pizzas =
        jsonResponse.map<Pizza>((i) => Pizza.fromJson(i)).toList();
      return pizzas;
    } else {
      return [];
    }
  }
}
```

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:

```
Future<List<Pizza>> callPizzas() async {
  HttpHelper helper = HttpHelper();
```

```

    List<Pizza> pizzas = await helper.getPizzaList();
    return pizzas;
}

```

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

```

class _MyHomePageState extends State<MyHomePage> {
  Future<List<Pizza>> callPizzas() async {
    HttpHelper helper = HttpHelper();
    List<Pizza> pizzas = await helper.getPizzaList();
    return pizzas;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('JSON'),
      ),
      body: FutureBuilder(
        future: callPizzas(),
        builder: (BuildContext context, AsyncSnapshot<List<Pizza>> snapshot) {
          if (snapshot.hasError) {
            return const Text('Something went wrong');
          }
          if (!snapshot.hasData) {
            return const CircularProgressIndicator();
          }
          return ListView.builder(
            itemCount: (snapshot.data == null) ? 0 : snapshot.data!.length,
            itemBuilder: (BuildContext context, int index) {
              return ListTile(
                title: Text(snapshot.data![index].pizzaName),
                subtitle: Text('${snapshot.data![index].description} - \${
                snapshot.data![index].price}'),
              );
            },
          );
        },
      ),
    );
  }
}

```

10. Jalankan aplikasi. Anda akan melihat layar yang mirip dengan berikut ini:



JSON

Margherita

Pizza with tomato, fresh mozzarella and basil - \$ 8.75

Marinara

Pizza with tomato, garlic and oregano - \$ 7.5

Napoli

Pizza with tomato, garlic and anchovies - \$ 9.5

Carciofi

Pizza with tomato, fresh mozzarella and artichokes - \$
8.8

Bufala

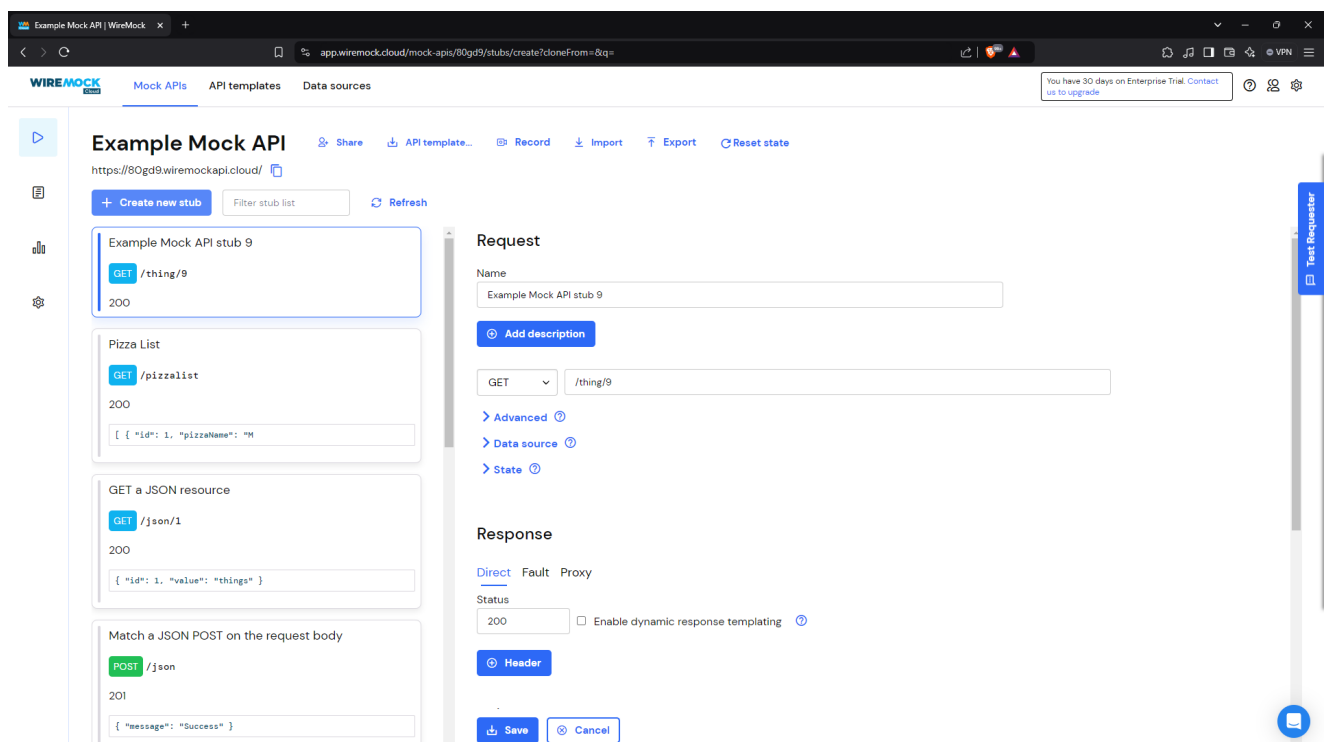
Pizza with tomato, buffalo mozzarella and basil - \$ 12.5



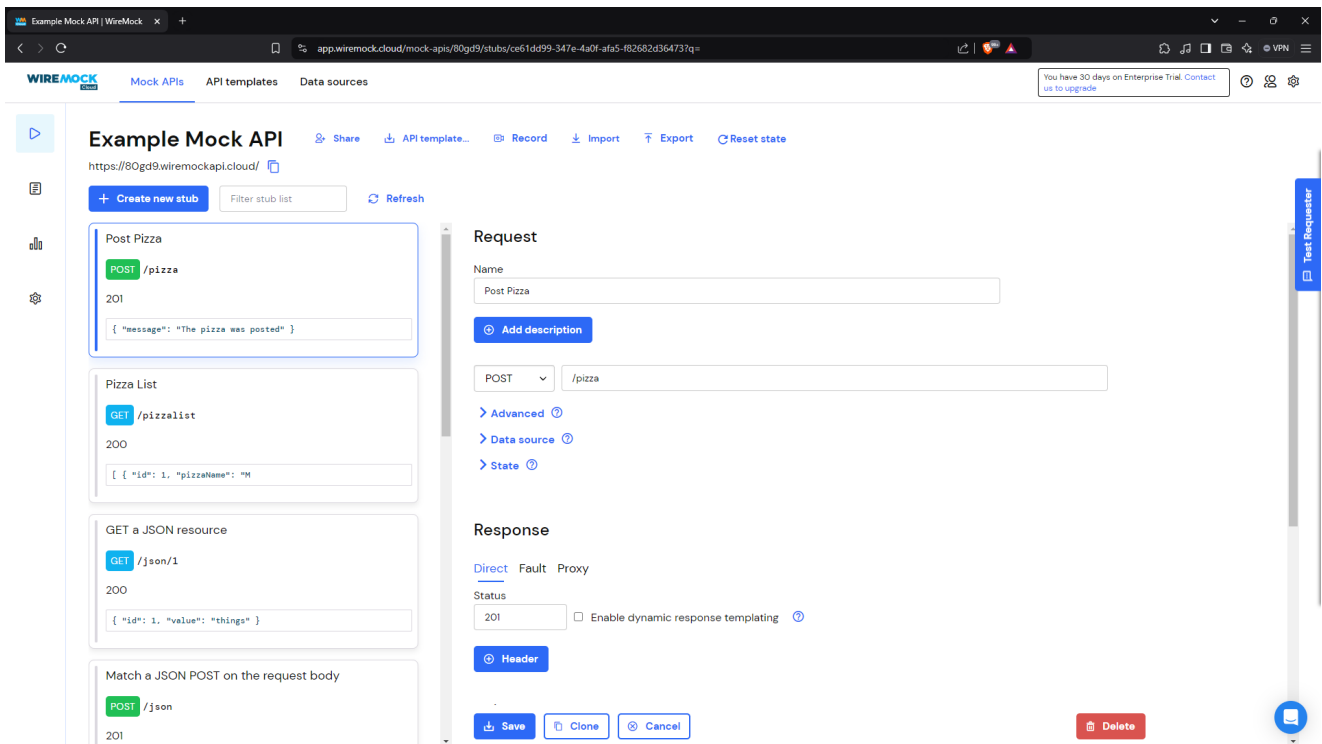
Praktikum 2. POST-ing data

Dalam praktikum ini, Anda akan mempelajari cara melakukan tindakan POST pada layanan web. Hal ini berguna ketika Anda menyambungkan ke layanan web yang tidak hanya menyediakan data, tetapi juga memungkinkan Anda untuk mengubah informasi yang tersimpan di sisi server. Biasanya, Anda harus menyediakan beberapa bentuk autentikasi ke layanan, tetapi untuk praktikum ini, karena kita menggunakan layanan Mock API, hal ini tidak diperlukan.

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:

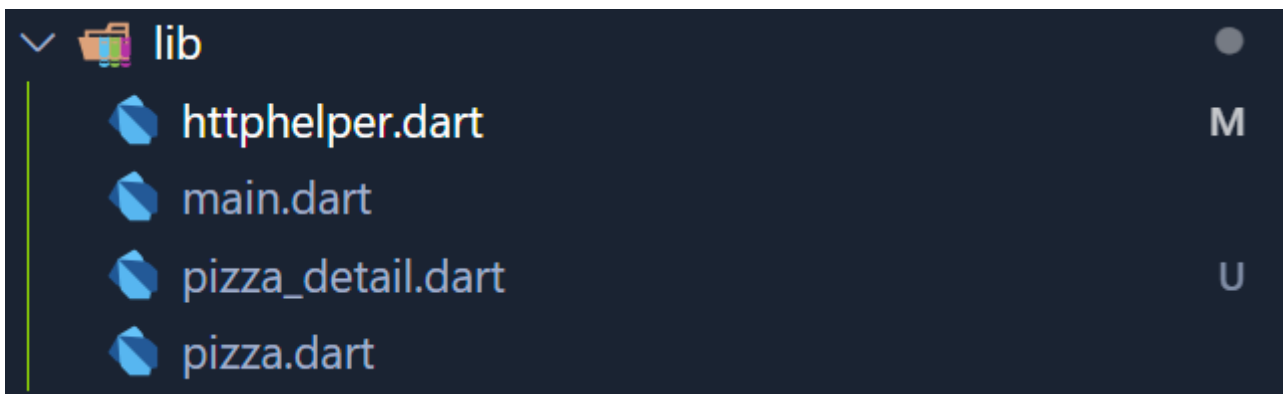


3. Simpan.

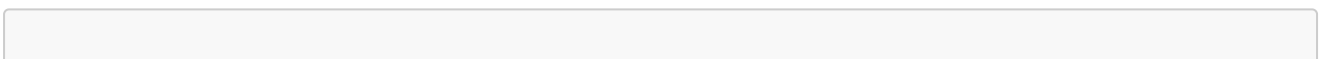
4. Di proyek Flutter, di file `httpHelper.dart`, di kelas `HttpHelper`, buat metode baru bernama `postPizza`, lengkapi kode sebagai berikut.

```
Future<String> postPizza(Pizza pizza) async {
  const postPath = '/pizza';
  String post = json.encode(pizza.toJson());
  Uri uri = Uri.https(authority, postPath);
  http.Response r = await http.post(uri, body: post);
  return r.body;
}
```

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



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



```
import 'package:flutter/material.dart';
import 'pizza.dart';
import 'httphelper.dart';
```

7. Buat StatefulWidget bernama PizzaDetailScreen.

```
import 'package:flutter/material.dart';
import 'pizza.dart';
import 'httphelper.dart';

class PizzaDetailScreen extends StatefulWidget {
  const PizzaDetailScreen({super.key});

  @override
  State<PizzaDetailScreen> createState() => _PizzaDetailScreenState();
}

class _PizzaDetailScreenState extends State<PizzaDetailScreen> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

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.

```
class _PizzaDetailScreenState extends State<PizzaDetailScreen> {
  TextEditingController txtId = TextEditingController();
  TextEditingController txtName = TextEditingController();
  TextEditingController txtDescription = TextEditingController();
  TextEditingController txtPrice = TextEditingController();
  TextEditingController txtImageUrl = TextEditingController();

  String operationalResult = '';

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

9. Override metode dispose() untuk membuang controllers


```

class _PizzaDetailScreenState extends State<PizzaDetailScreen> {
  TextEditingController txtId = TextEditingController();
  TextEditingController txtName = TextEditingController();
  TextEditingController txtDescription = TextEditingController();
  TextEditingController txtPrice = TextEditingController();
  TextEditingController txtImageUrl = TextEditingController();

  String operationalResult = '';

  @override
  void dispose() {
    txtId.dispose();
    txtName.dispose();
    txtDescription.dispose();
    txtPrice.dispose();
    txtImageUrl.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}

```

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.

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Pizza Detail'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12),
      child: SingleChildScrollView(
        child: Column(
          children: [],
        ),
      ),
    ),
  );
}

```

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.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Pizza Detail'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12),
      child: SingleChildScrollView(
        child: Column(
          children: [
            Text(
              operationalResult,
              style: TextStyle(
                backgroundColor: Colors.green[200], color: Colors.black),
            ),
            const SizedBox(
              height: 24,
            ),
            TextField(
              controller: txtId,
              decoration: const InputDecoration(hintText: 'Insert ID'),
            ),
            const SizedBox(
              height: 24,
            ),
            TextField(
              controller: txtName,
              decoration: const InputDecoration(hintText: 'Insert Pizza
Name'),
            ),
            const SizedBox(
              height: 24,
            ),
            TextField(
              controller: txtDescription,
              decoration: const InputDecoration(hintText: 'Insert
Description'),
            ),
            const SizedBox(
              height: 24,
            ),
            TextField(
              controller: txtPrice,
              decoration: const InputDecoration(hintText: 'Insert Price'),
            ),
            const SizedBox(
              height: 24,
```

```

        ),
        TextField(
          controller: txtImageUrl,
          decoration: const InputDecoration(hintText: 'Insert Image
Url'),
        ),
        ElevatedButton(
          onPressed: () => postPizza(),
          child: const Text('Send Post')
        )
      ],
    ),
  ),
);
}

```

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

```

Future postPizza() async {
  HttpHelper helper = HttpHelper();
  Pizza pizza = Pizza();
  pizza.id = int.tryParse(txtId.text) ?? 0;
  pizza.pizzaName = txtName.text;
  pizza.description = txtDescription.text;
  pizza.price = double.tryParse(txtPrice.text) ?? 0;
  pizza.imageUrl = txtImageUrl.text;

  String result = await helper.postPizza(pizza);
  setState(() {
    operationalResult = result;
  });
}

```

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

```

import 'package:flutter/material.dart';
import 'package:pizza_api_naufal/httphelper.dart';
import 'pizza.dart';
import 'pizza_detail.dart';

```

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

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('JSON'),
    ),
    body: FutureBuilder(
      future: callPizzas(),
      builder: (BuildContext context, AsyncSnapshot<List<Pizza>>
snapshot) {
        if (snapshot.hasError) {
          return const Text('Something went wrong');
        }
        if (!snapshot.hasData) {
          return const CircularProgressIndicator();
        }
        return ListView.builder(
          itemCount: (snapshot.data == null) ? 0 : snapshot.data!.length,
          itemBuilder: (BuildContext context, int index) {
            return ListTile(
              title: Text(snapshot.data![index].pizzaName),
              subtitle: Text(
                '${snapshot.data![index].description} - \$
${snapshot.data![index].price}'),
            );
          },
        );
      },
    floatingActionButton: FloatingActionButton(
      child: const Icon(Icons.add),
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => const PizzaDetailScreen(),
          )
        );
      },
    ),
  );
}

```

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



JSON

Margherita

Pizza with tomato, fresh mozzarella and basil - \$ 8.75

Marinara

Pizza with tomato, garlic and oregano - \$ 7.5

Napoli

Pizza with tomato, garlic and anchovies - \$ 9.5

Carciofi

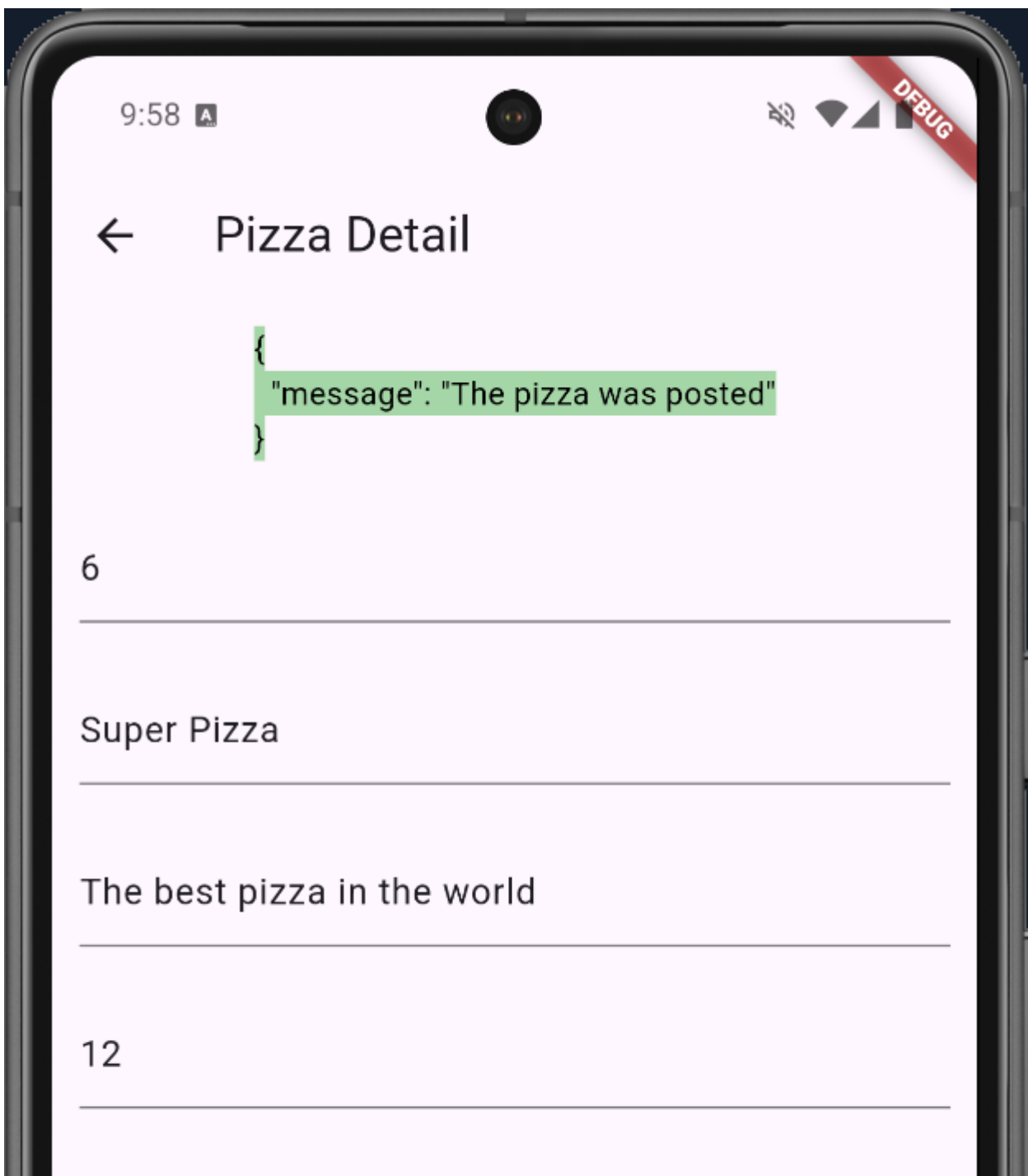
Pizza with tomato, fresh mozzarella and artichokes - \$ 8.8

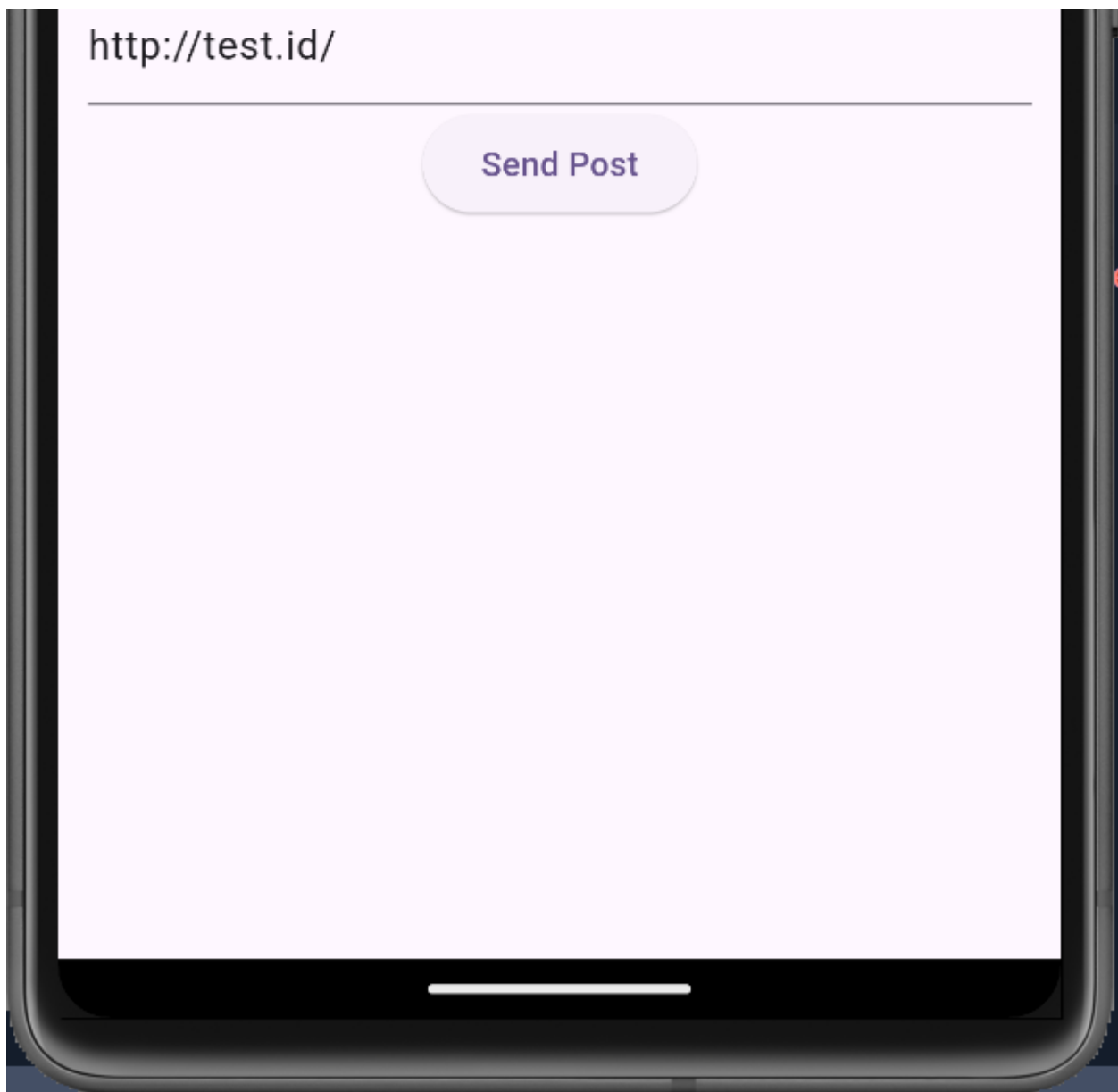
Bufala

Pizza with tomato, buffalo mozzarella and basil - \$ 12.5



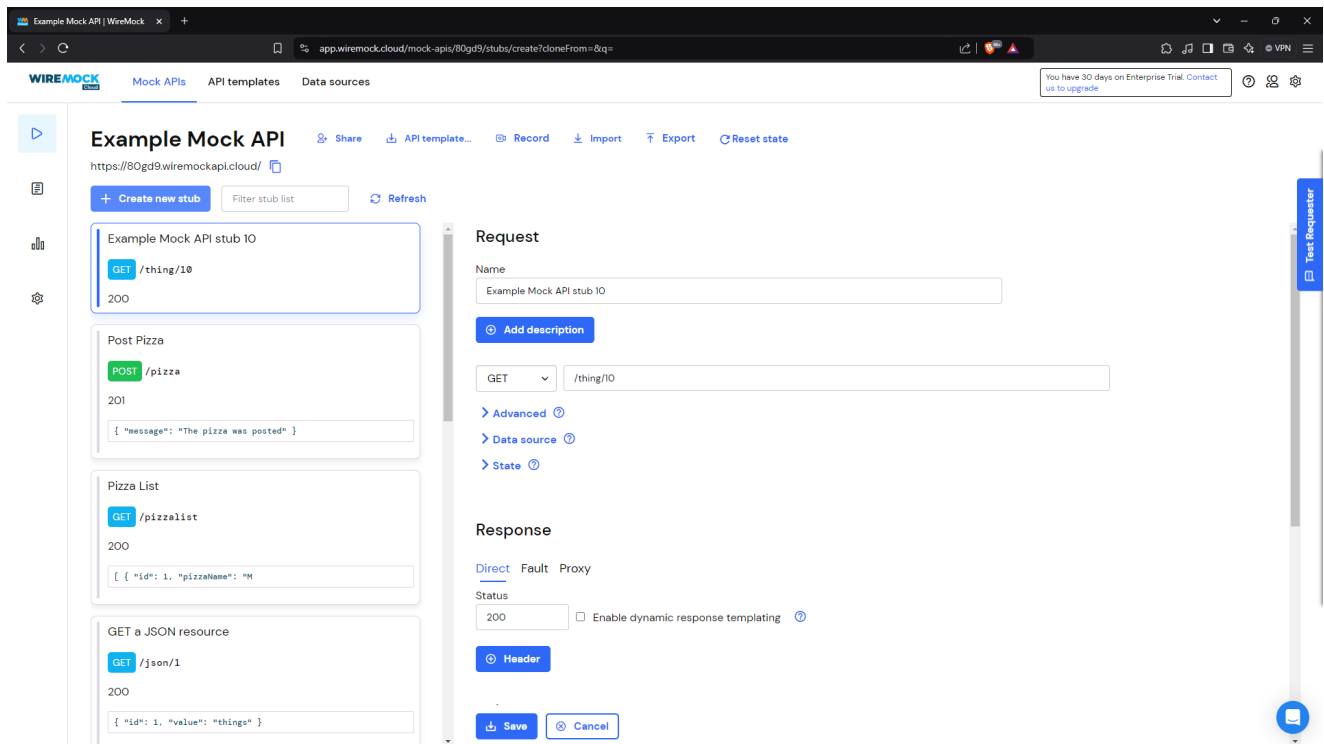
16. Tambahkan detail pizza di kolom teks dan tekan tombol Kirim Postingan. Anda akan melihat hasil yang berhasil, seperti yang ditunjukkan pada gambar berikut.



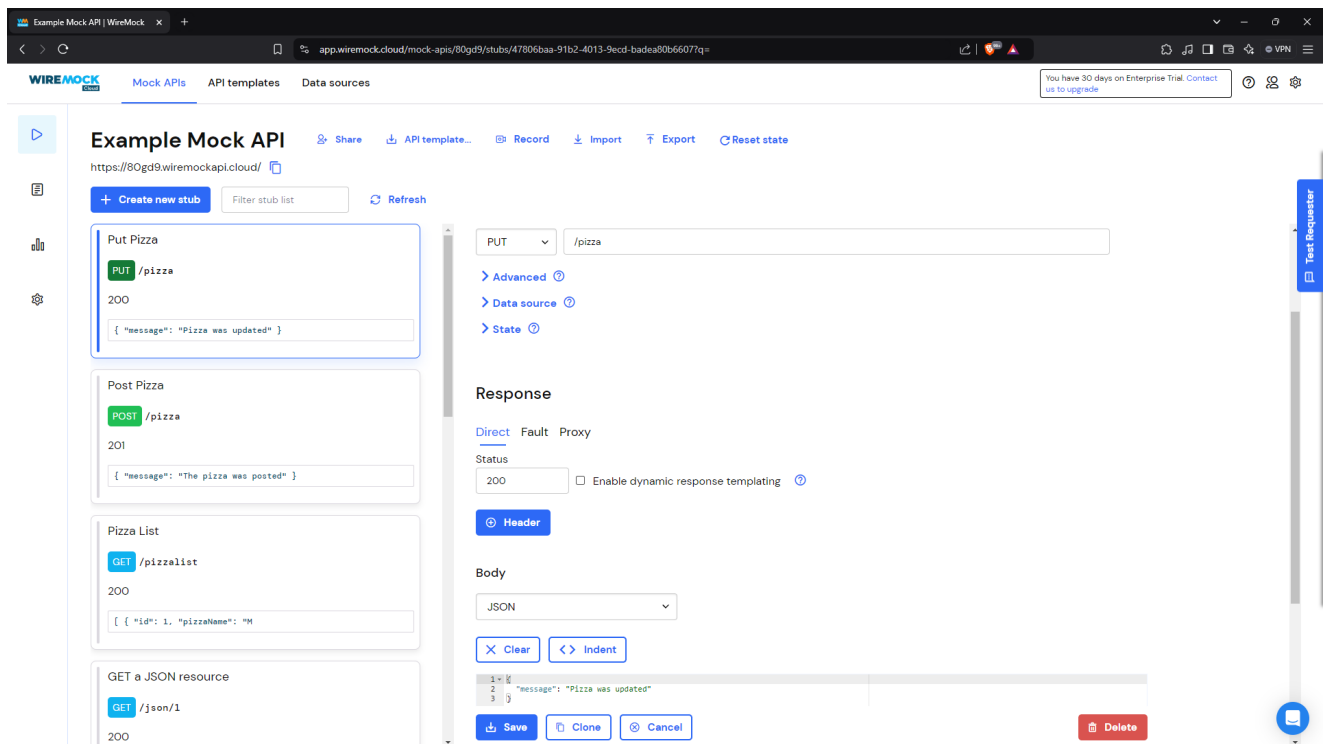


Praktikum 3, PUT-ting data

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, tambahkan metode putPizza ke kelas HttpHelper di file http_helper.dart

```
Future<String> putPizza(Pizza pizza) async {
  const putPath = '/pizza';
  String put = json.encode(pizza.toJson());
```



```

    Uri url = Uri.https(authority, putPath);
    http.Response r = await http.put(url, body: put);
    return r.body;
}

```

5. Di kelas PizzaDetailScreen di file pizza_detail.dart, tambahkan dua properti, Pizza dan boolean, dan di konstruktor, atur dua properti tersebut

```

final Pizza pizza;
final bool isNew;

const PizzaDetailScreen({super.key, required this.pizza, required this.isNew});

```

6. Di kelas PizzaDetailScreenState, override metode initState. Bila properti isNew dari kelas PizzaDetail tidak baru, properti ini akan menetapkan konten TextFields dengan nilai objek Pizza yang dilewatkan

```

@override
void initState() {
  if (!widget.isNew) {
    txtId.text = widget.pizza.id.toString();
    txtName.text = widget.pizza.pizzaName;
    txtDescription.text = widget.pizza.description;
    txtPrice.text = widget.pizza.price.toString();
    txtImageUrl.text = widget.pizza.imageUrl;
  }
  super.initState();
}

```

7. Edit metode savePizza sehingga memanggil metode helper.postPizza ketika properti isNew bernilai benar, dan helper.putPizza ketika bernilai salah

```

Future savePizza() async {
  HttpHelper helper = HttpHelper();
  Pizza pizza = Pizza();
  pizza.id = int.tryParse(txtId.text) ?? 0;
  pizza.pizzaName = txtName.text;
  pizza.description = txtDescription.text;
  pizza.price = double.tryParse(txtPrice.text) ?? 0;
  pizza.imageUrl = txtImageUrl.text;

  String result = await (widget.isNew ? helper.postPizza(pizza) :
  helper.putPizza(pizza));
  setState(() {
    operationalResult = result;
  });
}

```

```
});
}
```

8. Di file main.dart, di metode build _MyHomePageState, tambahkan properti onTap ke ListTile sehingga saat pengguna mengetuknya, aplikasi akan mengubah rute dan menampilkan layar PizzaDetail, dengan menampilkan data pizza yang ada saat ini dan menjadikan false untuk parameter isNew

```
return ListView.builder(
  itemCount: (snapshot.data == null) ? 0 : snapshot.data!.length,
  itemBuilder: (BuildContext context, int index) {
    return ListTile(
      title: Text(snapshot.data![index].pizzaName),
      subtitle: Text(
        '${snapshot.data![index].description} - \$ ${snapshot.data!
[index].price}'),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => PizzaDetailScreen(
              pizza: snapshot.data![index], isNew: false),
          ));
      },
    );
  },
);
```

9. Di floatingActionButton, passing data Pizza baru dan menjadikan true untuk parameter isNew ke rute PizzaDetail

```
floatingActionButton: FloatingActionButton(
  child: const Icon(Icons.add),
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => PizzaDetailScreen(pizza: Pizza(), isNew:
true),
      ));
  },
),
```

10. Jalankan aplikasi. Pada layar utama, ketuk Pizza mana pun untuk menavigasi ke rute PizzaDetail

11. Edit detail pizza di kolom teks dan tekan tombol Simpan. Anda akan melihat pesan yang menunjukkan bahwa detail pizza telah diperbarui

