**Ketentuan;**

Simpanlah file lembar jawaban ini dengan format; **Kelas\_Nama Lengkap**

Contoh; **TI 3C\_Rifki Fakhrudin**

**Upload tugas jobsheet ini dengan batas maksimum sesuai jadwal perkuliahan masing masing kelas**

Upload file tugas jobsheet di website **lms.polinema**

|  |  |  |
| --- | --- | --- |
| **Nama** | : | Rifki Fakhrudin |
| **Nim** | : | 2241720218 |
| **Kelas** | : | 3C |

Tulislah Jawaban Pada Kolom Yang tersedia di bawah ini;

**LEMBAR JAWABAN JOBSHEET-14**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
|  | **Praktikum 1, Designing an HTTP client and getting data** Mendaftarlah ke layanan Lab Mock di <https://app.wiremock.cloud/>. |
|  | Di halaman dahsboard, klik menu Stubs, kemudian klik entri pertama yaitu “GET a JSON resource”. Anda akan melihat layar yang mirip dengan berikut. |
|  | Klik “Create new stub”. |
|  | Tekan tombol SAVE di bagian bawah halaman untuk menyimpan Mock ini. Jika berhasil tersimpan, maka Mock API sudah siap digunakan. |
|  | Buatlah project flutter baru dengan nama pizza\_api\_nama\_anda, tambahkan depedensi “http” melalui terminal. |
|  | DI folder “lib” project anda, tambahkan file dengan nama “httphelper.dart”.  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 = 'dmr6j.wiremockapi.cloud';    final String path = 'pizzalist';    Future<List<Pizza>> getPizzaList() async {      final Uri url = Uri.https(authority, path);      final http.Response result = await http.get(url);      if (result.statusCode == HttpStatus.ok) {        final jsonResponse = json.decode(result.body);          // Provide a type argument to the map method to avoid type error        List<Pizza> pizzas = jsonResponse.map<Pizza>(          (i) => Pizza.fromJson(i),        ).toList();          return pizzas;      } else {        return [];      }    }  } |
|  | 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: |
|  | Pada metode build di class \_MyHomePageState, di dalam body Scaffold, tambahkan FutureBuilder yang membuat ListView dari widget ListTile yang berisi objek Pizza:  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 position) {                return ListTile(                  title: Text(snapshot.data![position].pizzaName),                  subtitle: Text(                    '${snapshot.data![position].description} - €${snapshot.data![position].price.toString()}',                  ),                );              },            );          },        ),      );    }  } |
|  | Jalankan aplikasi. Anda akan melihat layar yang mirip dengan berikut ini: |
|  | **Praktikum 2. POST-ing data**  Masuk ke layanan Lab Mock di https://app.wiremock.cloud/ dan klik bagian Stubs, kemudian, buatlah stub baru. |
|  | Simpan. |
|  | Di proyek Flutter, di file httpHelper.dart, di kelas HttpHelper, buat metode baru bernama postPizza, lengkapi kode sebagai berikut |
|  | Di dalam proyek, buat sebuah file baru bernama pizza\_detail.dart. |
|  | Buat StatefulWidget bernama PizzaDetailScreen |
|  | 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 |
|  | Override metode dispose() untuk membuang controllers |
|  | 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. |
|  | 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.      children: [                // Menampilkan hasil posting                Text(                  operationResult,                  style: TextStyle(                    backgroundColor: Colors.green[200],                    color: Colors.black,                  ),                ),                const SizedBox(height: 24),                // Input ID                TextField(                  controller: txtId,                  decoration: const InputDecoration(hintText: 'Insert ID'),                ),                const SizedBox(height: 24),                // Input Nama Pizza                TextField(                  controller: txtName,                  decoration:                      const InputDecoration(hintText: 'Insert Pizza Name'),                ),                const SizedBox(height: 24),                // Input Deskripsi                TextField(                  controller: txtDescription,                  decoration:                      const InputDecoration(hintText: 'Insert Description'),                ),                const SizedBox(height: 24),                // Input Harga                TextField(                  controller: txtPrice,                  decoration: const InputDecoration(hintText: 'Insert Price'),                ),                const SizedBox(height: 24),                // Input URL Gambar                TextField(                  controller: txtImageUrl,                  decoration: const InputDecoration(hintText: 'Insert Image Url'),                ),                const SizedBox(height: 48),                // Tombol Kirim POST                ElevatedButton(                  child: const Text('Send Post'),                  onPressed: () {                    postPizza();                  },                ),              ],            ),          ),        ),      );    } |
|  | Di bagian bawah kelas \_PizzaDetailState, tambahkan metode postPizza |
|  | Di file main.dart, impor file pizza\_detail.dart. |
|  | Di perancah metode build() dari kelas \_MyHomePageState, tambahkan FloatingActionButton yang akan menavigasi ke rute PizzaDetail. |
|  | Jalankan aplikasi. Pada layar utama, tekan FloatingActionButton untuk menavigasi ke rute PizzaDetail |
|  | 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** Masuk ke layanan Lab Mock di https://app.wiremock.cloud/ dan klik bagian Stubs, kemudian, buatlah stub baru. |
|  | Lengkapi isian seperti gambar berikut: |
|  | Simpan |
|  | Di proyek Flutter, tambahkan metode putPizza ke kelas HttpHelper di file http\_helper.dart |
|  | Di kelas PizzaDetailScreen di file pizza\_detail.dart, tambahkan dua properti, Pizza dan boolean, dan di konstruktor, atur dua properti tersebut |
|  | 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 |
|  | Edit metode savePizza sehingga memanggil metode helper.postPizza ketika properti isNew bernilai benar, dan helper.putPizza ketika bernilai salah |
|  | 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 |
|  | Di floatingActionButton, passing data Pizza baru dan menjadikan true untuk parameter isNew ke rute PizzaDetail |
|  | Jalankan aplikasi. Pada layar utama, ketuk Pizza mana pun untuk menavigasi ke rute PizzaDetail |
|  | Edit detail pizza di kolom teks dan tekan tombol Simpan. Anda akan melihat pesan yang menunjukkan bahwa detail pizza telah diperbarui |
|  | **Praktikum 4, DELETE-ing data**  Masuk ke layanan Wiremock di https://app.wiremock.cloud dan klik pada bagian Stubs pada contoh API. Kemudian, buatlah sebuah stub baru |
|  | Lengkapi isian, dengan data berikut: • Name: Delete Pizza • Verb: DELETE • Address: /pizza • Status: 200 • Body Type: json • Body: {"message": "Pizza was deleted"} |
|  | Save the new stub |
|  | Di proyek Flutter, tambahkan metode deletePizza ke kelas HttpHelper di file http\_helper.dart |
|  | Pada file main.dart, di metode build kelas \_MyHomePageState, refaktor itemBuilder dari ListView.builder agar ListTile terdapat dalam widget Dismissible, seperti berikut |
|  | Jalankan aplikasi. Saat Anda menggeser elemen apa pun dari daftar pizza, ListTile akan menghilang |