MODUL PEMROGRAMAN MOBILE

CIM 430

MODUL PERTEMUAN 12

FLUTTER – REST API

(APPLICATION PROGRAMMING INTERFACE)

DISUSUN OLEH TIM DOSEN :

7174 – SAWALI WAHYU, S.KOM, M.KOM

8126 – JEFRY SUNUPURWA ASRI, S.KOM, M.KOM

7176 – IKSAN RAMADHAN, S.KOM, M.KOM

UNIVERSITAS ESA UNGGUL

FAKULTAS ILMU KOMPUTER

TAHUN 2021

FLUTTER – REST API
(APPLICATION PROGRAMMING INTERFACE)

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

1. Mahasiswa mampu membuat koneksi API Pada Aplikasi Mobile
2. Mahasiswa mampu memahami arsitektur model aplikasi untuk membuat Rest API
3. Mahasiswa mampu membuat aplikasi dengan kompleksitas koneksi antara API dan Aplikasi Mobile yang dapat memiliki output yang relevan.

Flutter - REST API

Flutter menyediakan paket http untuk menggunakan sumber daya HTTP. http adalah pustaka berbasis masa depan dan menggunakan fitur await dan async. Ini menyediakan banyak metode tingkat tinggi dan menyederhanakan pengembangan aplikasi seluler berbasis REST.

Konsep dasar

Paket http menyediakan kelas tingkat tinggi dan http untuk melakukan permintaan web.

- Kelas http menyediakan fungsionalitas untuk melakukan semua jenis permintaan HTTP.
- Metode http menerima url, dan informasi tambahan melalui Dart Map (data posting, header tambahan, dll.,). Ini meminta server dan mengumpulkan respons kembali dalam pola async / await. Misalnya, kode di bawah ini membaca data dari url yang ditentukan dan mencetaknya di konsol.

```
print(await http.read('https://flutter.dev/'));
```

Beberapa metode inti adalah sebagai berikut -

- **read** - Meminta url yang ditentukan melalui metode GET dan mengembalikan respons sebagai <String> Masa Depan
- **get** - Minta url yang ditentukan melalui metode GET dan kembalikan respons sebagai <Response> Masa Depan. Respon adalah kelas yang memegang informasi respon.
- **post** - Minta url yang ditentukan melalui metode POST dengan memposting data yang disediakan dan mengembalikan respons sebagai <Response> Masa Depan
- **put** - Minta url yang ditentukan melalui metode PUT dan kembalikan respons sebagai <Response> Masa Depan
- **head** - Minta url yang ditentukan melalui metode HEAD dan kembalikan respons sebagai <Response> Masa Depan
- **delete** - Minta url yang ditentukan melalui metode DELETE dan kembalikan respons sebagai <Response> Masa Depan

http juga menyediakan kelas klien HTTP yang lebih standar, klien. klien mendukung koneksi persisten. Ini akan berguna ketika banyak permintaan yang akan dibuat ke server tertentu. Itu harus ditutup dengan benar menggunakan metode tutup. Jika tidak, ini mirip dengan kelas http. Kode sampelnya adalah sebagai berikut −

```
var client = new http.Client();
try {
   print(await client.get('https://flutter.dev/'));
}
finally {
   client.close();
```

Mengakses API layanan Produk

Mari kita buat aplikasi sederhana untuk mendapatkan data produk dari server web dan kemudian menampilkan produk menggunakan *ListView* .

- Buat aplikasi *Flutter* baru di studio Android, *product_rest_app* .
- Ganti kode startup default (main.dart) dengan kode *product_nav_app* kami .
- Salin folder aset dari *product_nav_app* ke *product_rest_app* dan tambahkan aset di dalam file pubspec.yaml.

```
flutter:
   assets:
      - assets/appimages/floppy.png
      - assets/appimages/iphone.png
      - assets/appimages/laptop.png
      - assets/appimages/pendrive.png
      - assets/appimages/pixel.png
      - assets/appimages/tablet.png
```

- Konfigurasikan paket http di file pubspec.yaml seperti yang ditunjukkan di bawah ini -

```
dependencies:
   http: ^0.12.0+2
```

- Di sini, kami akan menggunakan versi terbaru dari paket http. Android studio akan mengirimkan peringatan paket bahwa pubspec.yaml telah diperbarui.

- Klik Dapatkan opsi ketergantungan. Android studio akan mendapatkan paket dari Internet dan mengkonfigurasinya dengan benar untuk aplikasi.
- Impor paket http di file main.dart -

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

- Buat file JSON baru, products.json dengan informasi produk seperti yang ditunjukkan di bawah ini -

```
[
    {
        "name": "iPhone",
        "description": "iPhone is the stylist phone ever",
        "price": 1000,
        "image": "iphone.png"
    },
    {
        "name": "Pixel",
        "description": "Pixel is the most feature phone ever",
        "price": 800,
        "image": "pixel.png"
```

```
    },
    {
        "name": "Laptop",
        "description": "Laptop is most productive development
tool",
        "price": 2000,
        "image": "laptop.png"
    },
    {
        "name": "Tablet",
        "description": "Tablet is the most useful device ever for
meeting",
        "price": 1500,
        "image": "tablet.png"
    },
    {
        "name": "Pendrive",
        "description": "Pendrive is useful storage medium",
        "price": 100,
        "image": "pendrive.png"
    },
    {
        "name": "Floppy Drive",
        "description": "Floppy drive is useful rescue storage
medium",
        "price": 20,
        "image": "floppy.png"
    }
]
```

- Buat folder baru, JSONWebServer dan tempatkan file JSON, products.json.
- Jalankan server web apa pun dengan JSONWebServer sebagai direktori akarnya dan dapatkan jalur webnya. Misalnya, http://192.168.184.1:8000/products.json. Kami dapat menggunakan server web apa pun seperti apache, nginx dll.,
- Cara termudah adalah dengan menginstal aplikasi http-server berbasis node. Ikuti langkah-langkah yang diberikan di bawah ini untuk menginstal dan menjalankan aplikasi http- server
    - Instal aplikasi Nodejs (nodejs.org)
    - Buka folder JSONWebServer.

```
cd /path/to/JSONWebServer
```

- Instal paket http-server menggunakan npm.

```
npm install -g http-server
```

- Sekarang, jalankan server.

```
http-server . -p 8000

Starting up http-server, serving .
Available on:
```

```
http://192.168.99.1:8000
http://127.0.0.1:8000
Hit CTRL-C to stop the server
```

- Buat file baru, Product.dart di folder lib dan pindahkan kelas Produk ke dalamnya.
- Tulis konstruktor pabrik di kelas Produk, Product.fromMap untuk mengubah peta data yang dipetakan menjadi objek Produk. Biasanya file JSON akan diubah menjadi objek Dart Map dan kemudian diubah menjadi objek yang relevan (Product).

```
factory Product.fromJson(Map<String, dynamic> data) {
   return Product(
      data['name'],
      data['description'],
      data['price'],
      data['image'],
   );
}
```

- Kode lengkap Product.dart adalah sebagai berikut -

```
class Product {
   final String name;
   final String description;
   final int price;
   final String image;

   Product(this.name, this.description, this.price, this.image);
   factory Product.fromMap(Map<String, dynamic> json) {
      return Product(
         json['name'],
         json['description'],
         json['price'],
         json['image'],
      );
   }
}
```

- Tulis dua metode - parseProducts dan fetchProducts - di kelas utama untuk mengambil dan memuat informasi produk dari server web ke objek List <Product>.

```
List<Product> parseProducts(String responseBody) {
   final parsed = json.decode(responseBody).cast<Map<String,
dynamic>>();
   return parsed.map<Product>((json)
=>Product.fromJson(json)).toList();
}
Future<List<Product>> fetchProducts() async {
   final response = await
http.get('http://192.168.1.2:8000/products.json');
   if (response.statusCode == 200) {
      return parseProducts(response.body);
```

```
    } else {
      throw Exception('Unable to fetch products from the REST
API');
    }
}
```

- Perhatikan poin-poin berikut di sini -
    - Future digunakan untuk memuat informasi produk secara lambat. Lazy loading adalah konsep untuk menunda eksekusi kode sampai dibutuhkan.
    - http.get digunakan untuk mengambil data dari Internet.
    - json.decode digunakan untuk memecahkan kode data JSON ke dalam objek Peta Dart. Setelah data JSON didekodekan, itu akan diubah menjadi Daftar <Produk> menggunakan fromMap dari kelas Produk.
    - Di kelas MyApp, tambahkan variabel anggota baru, produk berjenis Future <Product> dan sertakan dalam konstruktor.

```
class MyApp extends StatelessWidget {
   final Future<List<Product>> products;
   MyApp({Key key, this.products}) : super(key: key);
   ...
```

- Di kelas MyHomePage, tambahkan produk variabel anggota baru dari tipe Future <Product> dan sertakan dalam konstruktor. Selain itu, hapus variabel item dan metode yang relevan, panggilan metode getProducts. Menempatkan variabel produk dalam konstruktor. Ini akan memungkinkan untuk mengambil produk dari Internet hanya sekali saat aplikasi pertama kali dimulai.

```
class MyHomePage extends StatelessWidget {
   final String title;
   final Future<ListList<Product>> products;
   MyHomePage({Key key, this.title, this.products}) : super(key:
key);
   ...
```

- Ubah opsi beranda (MyHomePage) dalam metode build widget MyApp untuk mengakomodasi perubahan di atas -

```
home: MyHomePage(title: 'Product Navigation demo home page',
products: products),
```

- Ubah fungsi utama untuk memasukkan argumen <Product> Future -

```
void main() => runApp(MyApp(fetchProduct()));
```

- Buat widget baru, ProductBoxList untuk membangun daftar produk di halaman beranda.

```
class ProductBoxList extends StatelessWidget {
   final List<Product> items;
   ProductBoxList({Key key, this.items});

   @override
```

```
   Widget build(BuildContext context) {
      return ListView.builder(
         itemCount: items.length,
         itemBuilder: (context, index) {
            return GestureDetector(
               child: ProductBox(item: items[index]),
               onTap: () {
                  Navigator.push(
                     context, MaterialPageRoute(
                        builder: (context) =gt; ProductPage(item:
items[index]),
                     ),
                  );
               },
            );
         },
      );
   }
}
```

Perhatikan bahwa kami menggunakan konsep yang sama yang digunakan dalam aplikasi Navigasi untuk membuat daftar produk kecuali itu dirancang sebagai widget terpisah dengan meneruskan produk (objek) berjenis List <Product>.

- Terakhir, ubah metode *build* widget *MyHomePage* untuk mendapatkan informasi produk menggunakan opsi Future, bukan panggilan metode normal.

```
Widget build(BuildContext context) {
   return Scaffold(
      appBar: AppBar(title: Text("Product Navigation")),
      body: Center(
         child: FutureBuilder<List<Product>>(
            future: products, builder: (context, snapshot) {
               if (snapshot.hasError) print(snapshot.error);
               return snapshot.hasData ? ProductBoxList(items:
snapshot.data)

               // return the ListView widget :
               Center(child: CircularProgressIndicator());
            },
         ),
      )
   );
}
```

- Di sini perhatikan bahwa kami menggunakan widget FutureBuilder untuk merender widget. FutureBuilder akan mencoba mengambil data dari properti masa depan (jenis Future <List <Product>>). Jika properti masa depan mengembalikan data, itu akan membuat widget menggunakan ProductBoxList, jika tidak memunculkan kesalahan.
- Kode lengkap main.dart adalah sebagai berikut -

```
import 'package:flutter/material.dart';
import 'dart:async';
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'Product.dart';
void main() => runApp(MyApp(products: fetchProducts()));
List<Product> parseProducts(String responseBody) {
   final parsed = json.decode(responseBody).cast<Map<String,
dynamic>>();
   return parsed.map<Product>((json) =>
Product.fromMap(json)).toList();
}
Future<List<Product>> fetchProducts() async {
   final response = await
http.get('http://192.168.1.2:8000/products.json');
   if (response.statusCode == 200) {
      return parseProducts(response.body);
   } else {
      throw Exception('Unable to fetch products from the REST
API');
   } }
class MyApp extends StatelessWidget {
   final Future<List<Product>> products;
   MyApp({Key key, this.products}) : super(key: key);
   // This widget is the root of your application.
   @override
   Widget build(BuildContext context) {
      return MaterialApp(
         title: 'Flutter Demo',
         theme: ThemeData(
           primarySwatch: Colors.blue, ),
         home: MyHomePage(title: 'Product Navigation demo home
page', products: products),
      ); } }
class MyHomePage extends StatelessWidget {
   final String title;
   final Future<List<Product>> products;
   MyHomePage({Key key, this.title, this.products}) : super(key:
key);
   // final items = Product.getProducts();
   @override
   Widget build(BuildContext context) {
      return Scaffold(
         appBar: AppBar(title: Text("Product Navigation")),
         body: Center(
```

```
            child: FutureBuilder<List<Product>>(
               future: products, builder: (context, snapshot) {
                  if (snapshot.hasError) print(snapshot.error);
                  return snapshot.hasData ? ProductBoxList(items:
snapshot.data)
                  // return the ListView widget :
                  Center(child: CircularProgressIndicator());
               },
            ),
         )
      );
   }
}
class ProductBoxList extends StatelessWidget {
   final List<Product> items;
   ProductBoxList({Key key, this.items});

   @override
   Widget build(BuildContext context) {
      return ListView.builder(
         itemCount: items.length,
         itemBuilder: (context, index) {
            return GestureDetector(
               child: ProductBox(item: items[index]),
               onTap: () {
                  Navigator.push(
                     context, MaterialPageRoute(
                        builder: (context) => ProductPage(item:
items[index]),
                     ), );
               }, ); }, ); } }
class ProductPage extends StatelessWidget {
   ProductPage({Key key, this.item}) : super(key: key);
   final Product item;
   @override
   Widget build(BuildContext context) {
      return Scaffold(
         appBar: AppBar(title: Text(this.item.name),),
         body: Center(
            child: Container(
               padding: EdgeInsets.all(0),
               child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
```

```
                        Image.asset("assets/appimages/" +
this.item.image),
                        Expanded(
                           child: Container(
                              padding: EdgeInsets.all(5),
                              child: Column(
                                 mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                                 children: <Widget>[
                                    Text(this.item.name, style:
                                       TextStyle(fontWeight:
FontWeight.bold)),
this.item.price.toString()),      Text(this.item.description),
                                 )      Text("Price: " +
                           )
                        )              RatingBox(),
                     ]                 ],
                  ),
               ),
            ),
         ); } }
class RatingBox extends StatefulWidget {
   @override
   _RatingBoxState createState() =>_RatingBoxState();
}
class _RatingBoxState extends State<RatingBox> {
   int _rating = 0;
   void _setRatingAsOne() {
      setState(() {
         _rating = 1;
      });
   }
   void _setRatingAsTwo() {
      setState(() {
         _rating = 2;
      }); }
   void _setRatingAsThree() {
      setState(() {
         _rating = 3;
      }); }
   Widget build(BuildContext context) {
      double _size = 20;
      print(_rating);
      return Row(
         mainAxisAlignment: MainAxisAlignment.end,
```

```
         crossAxisAlignment: CrossAxisAlignment.end,
         mainAxisSize: MainAxisSize.max,

         children: <Widget>[
            Container(
               padding: EdgeInsets.all(0),
               child: IconButton(
                  icon: (
                     _rating >= 1
                     ? Icon(Icons.star, ize: _size,)
                     : Icon(Icons.star_border, size: _size,)
                  ),
                  color: Colors.red[500], onPressed:
_setRatingAsOne, iconSize: _size,
               ),
            ),
            Container(
               padding: EdgeInsets.all(0),
               child: IconButton(
                  icon: (
                     _rating >= 2
                     ? Icon(Icons.star, size: _size,)
                     : Icon(Icons.star_border, size: _size, )
                  ),
                  color: Colors.red[500],
                  onPressed: _setRatingAsTwo,
                  iconSize: _size, ), ),
            Container(
               padding: EdgeInsets.all(0),
               child: IconButton(
                  icon: (
                     _rating >= 3 ?
                     Icon(Icons.star, size: _size,)
                     : Icon(Icons.star_border, size: _size,)
                  ),
                  color: Colors.red[500],
                  onPressed: _setRatingAsThree,
                  iconSize: _size,
               ), ), ], ); } }
class ProductBox extends StatelessWidget {
   ProductBox({Key key, this.item}) : super(key: key);
   final Product item;
   Widget build(BuildContext context) {
      return Container(
   padding: EdgeInsets.all(2), height:
      140,
```

```
          child: Card(
             child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: <Widget>[
                   Image.asset("assets/appimages/" +
this.item.image),
                   Expanded(
                      child: Container(
                         padding: EdgeInsets.all(5),
                         child: Column(
                            mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                            children: <Widget>[
                               Text(this.item.name,
style:TextStyle(fontWeight: FontWeight.bold)),
                               Text(this.item.description),
                               Text("Price: " +
this.item.price.toString()),
                               RatingBox(),
                            ],
                         )
                      )
                   )
                ]
             ),
          )
       );
    }
}
```

Terakhir jalankan aplikasi untuk melihat hasilnya. Ini akan sama dengan contoh *Navigasi* kami kecuali datanya dari Internet, bukan lokal, data statis dimasukkan saat mengkode aplikasi.