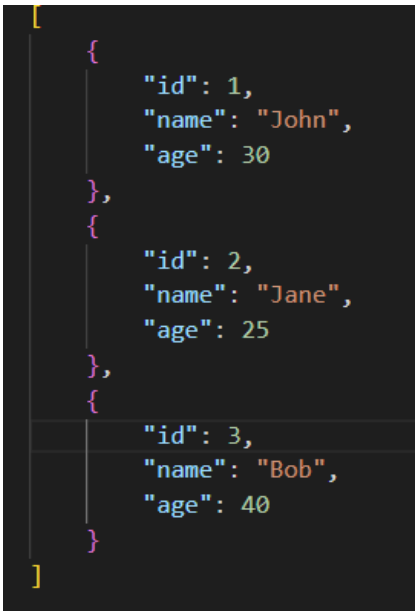
PRAKTIKUM MCS BAB 5
Thingspeak

PENDAHULUAN

Pada praktikum MCS bab 5 akan belajar bagaimana membangun aplikasi yang terhubung dengan thingspeak melalui Application Programming Interface (API). Thingspeak merupakan perangkat lunak bersifat terbuka/layanan yang biasa digunakan untuk project Internet Of Things (IOT) berguna untuk memantau dan mengumnpulkan data atau bahkan perubahan data dengan koneksi API. Aplikasi mengambil data dari API dalam bentuk JSON. JSON memiliki berbagai macam bentuk seperti array of objects, object with nested, object with array dan bentuk lainnya.



Array of objects

Array with nested

```json
{
    "name": "John",
    "age": 30,
    "hobbies": ["reading", "traveling", "hiking"]
}
```

Objects with array

Interaksi dengan JSON memiliki dua metode yaitu encode dan decode. Encode adalah proses mengubah object menjadi format JSON, contohnya adalah ketika aplikasi membaca data dari API. sedangkan decode adalah kebalikannya yaitu mengubah format JSON menjadi object, contohnya adalah ketika mengirim data (biasanya ke database sebelum diproses dengan query).

PRAKTIKUM BAB 5

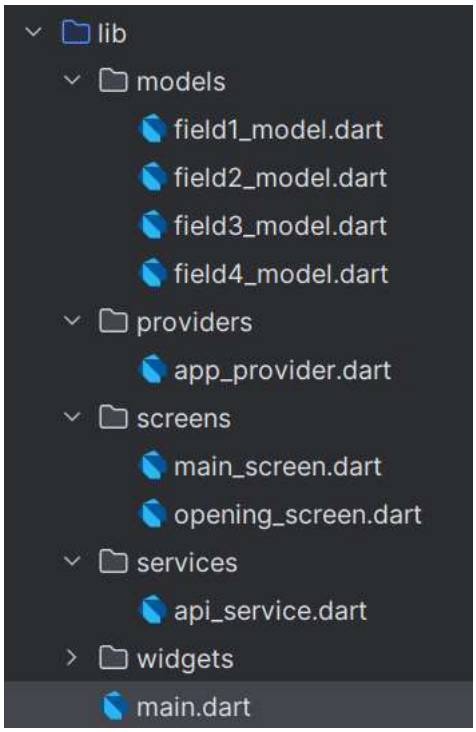
Tampilan aplikasi yang akan dibangun.

Tampilan pembukaan aplikasi

Tampilan membaca field

Penjelasan cara kerja aplikasi akan diterangkan oleh Penanggung Jawab (PJ)

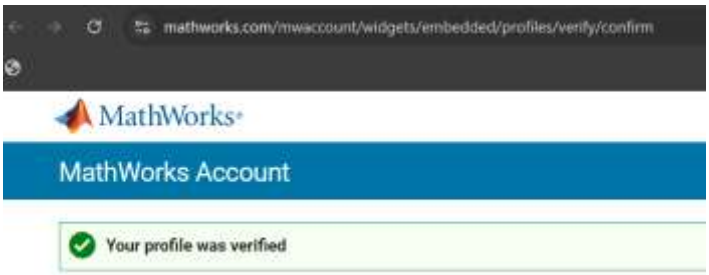
Buat terlebih dahulu folder dan file project msc bab 5 dan tambahkan juga package dio dan provider.



Folder dan file project

Sebelum lanjut menulis code untuk membangun aplikasi, buat akun Thingspeak

Registrasi akun



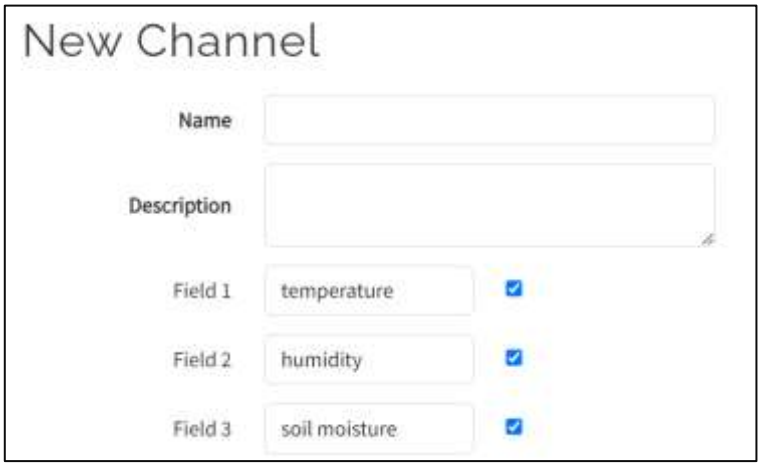
Verifikasi email

Menambahkan password akun thingspeak

Sign up successful

Buka website https://thingspeak.com/ dan klick Get Started For Free. Isi email address, location dan nama pengguna kemudian klick Continue. Untuk verifikasi email address Thingspeak akan mengirim instruksi melalui email. Terakhir berikan password untuk akun Thingspeak.

Buat channel baru

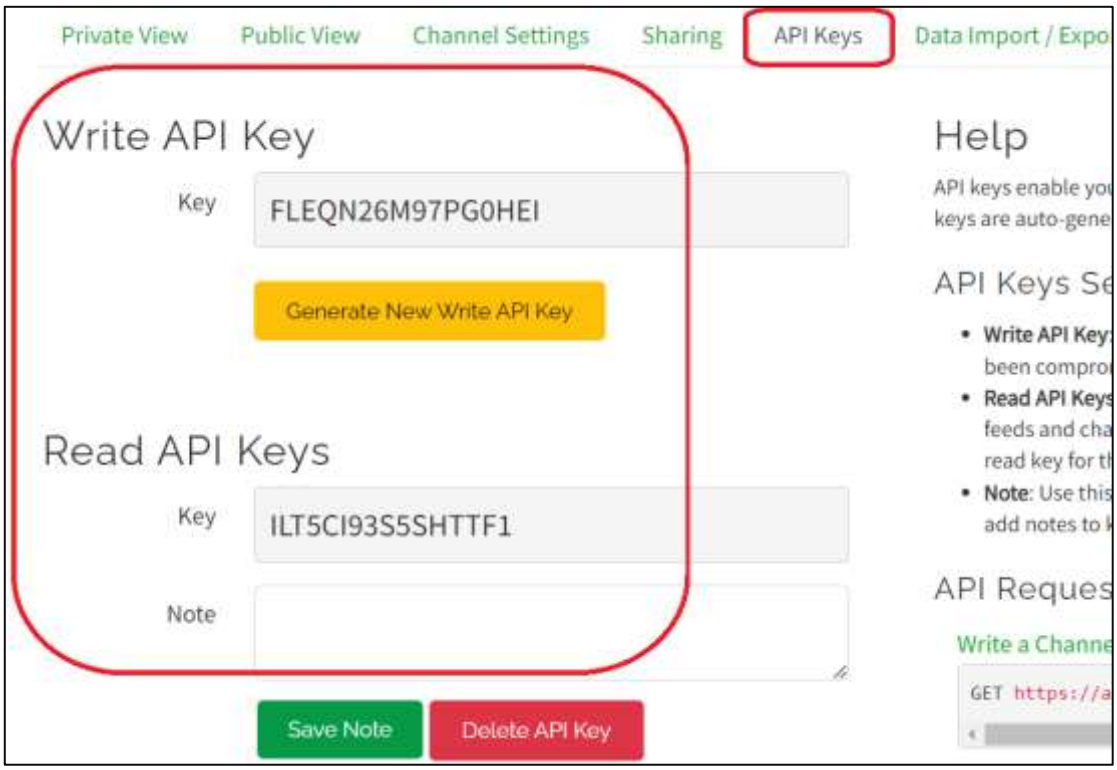


Channel fields

Sebuah akun dari Thingspeak untuk free member dapat membuat 4 channel dan setiap channel memiliki 8 fields. Setiap 1 field menyimpan 1 data, misalnya suhu ruangan yang dibaca oleh sensor dht dan masuk ke micro controller. Di praktikum ini kita akan membuat 1 channel 3 fields. 3 fields tersebut akan kita asumsikan sebagai data dari suhu ruangan, kelembaban ruangan dan kelembaban tanah. Berikan nama channel dan deskripsi bebas begitu juga nama dari setiap field namun untuk mempermemudah praktikum dan sebagai tanda maka field 1 diberi nama temperature, field 2 diberi nama humidity, field 3 diberi nama soil moisture. Jika sudah selesai dibuat akan muncul informasi channel.

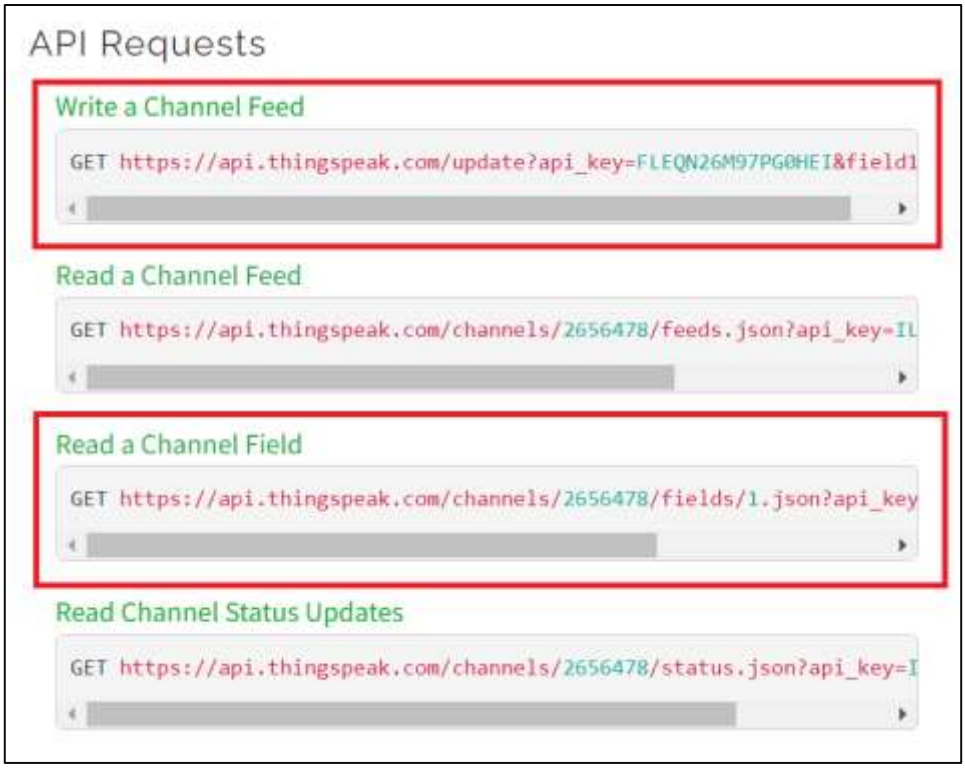
Informasi channel

Beralih ke menu Api keys dan perhatikan api requestnya



Menu api keys

Write api keys dan Read api keys adalah kunci yang berbentuk susunan angka dan huruf untuk digabungkan ke endpoint. Write api keys dan Read api keys digenerasikan dari channel yang dibuat pemilik akun.



Api request

Untuk mengubah nilai suatu field menggunakan endpoint Write metode GET dengan memasukan nilau baru ke dalam paramater di akhir url. Jika ingin mengubah nilai dari suatu field misalnya field pertama maka bentuk endpoint di akhir url adlah &filed1=<value>, jika field kedua maka bentuk endpoint di akhir url adlah &filed2=<value>.

Isikan 3 field yang sudah dibuat menggunakan endpoint Write, field pertama diberi nilai 23, field kedua diberi nilai 52 dan field ketiga diberi nilai 39.

https://api.thingspeak.com/update?api_key=<Write API Key> &field<nomor field>=<value>

Endpoint write ini tidak akan ada di dalam aplikasi dan hanya dijalankan untuk inisialisasi nilai field. Menjalankan endpoint ini bisa menggunakan web browser ataupun menggunakan aplikasi postman.

Endpoint read akan ada di dalam aplikasi. Endpoint read pada umumnya memiliki url bentuk :

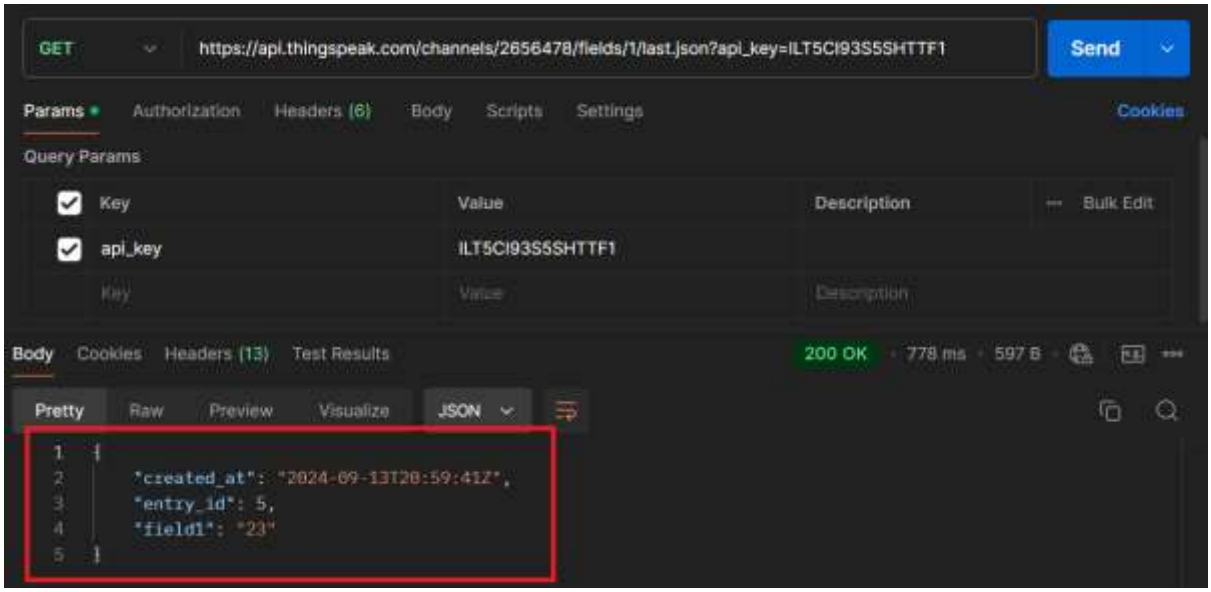
https://api.thingspeak.com/channels/(Channel id)/fields/<nomor field>.json?api_key=Read API Keys&results=2

dan akan sedikit dirubah menjadi

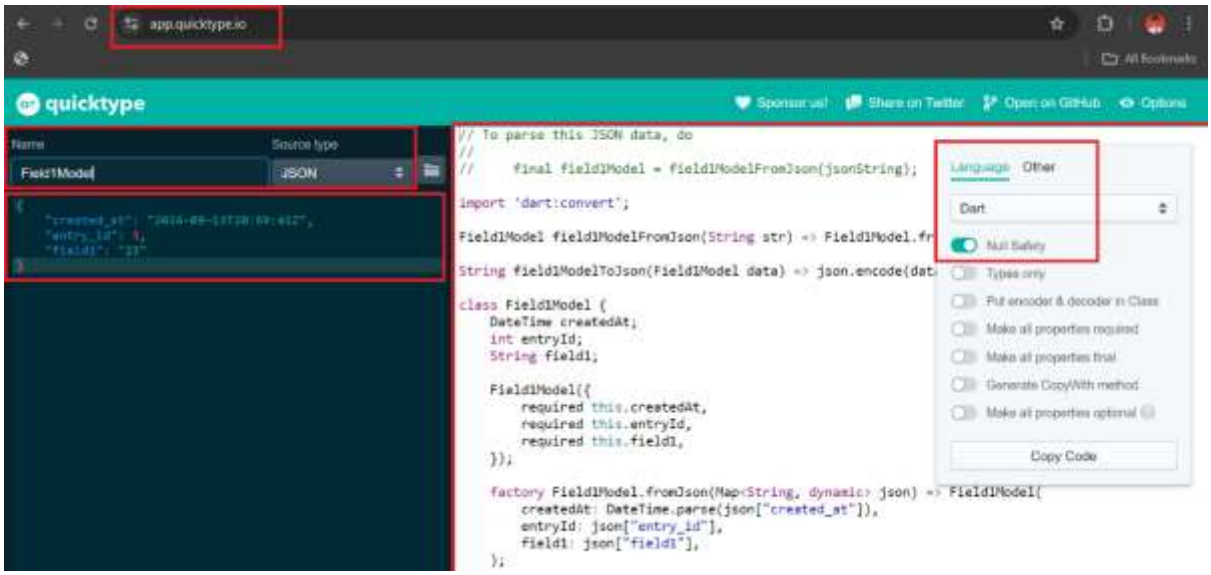
https://api.thingspeak.com/channels/(Channel id)/fields/<nomor field>/last.json?api_key=Read API Keys

Sebelum dimasukkann dan dijalankan di dalam aplikasi jalankan terlebih dahulu di web browser ataupun postman untuk mengambil response dari setiap field. Copy response yang

didapatkan, buka link website https://app.quicktype.io/ atur nama model, bahasa yang dipilih bahasa pemrograman Dart gunakan Null Safety dan paste ke dalamnya.



Response from API



Quicktype

Lakukan konversi untuk semua field kemudian letakkan hasil dari conversi JSON ke dalam file yang berada di folder models, di dalam folder models terdapat 3 file .dart, masing-masing file .dart berisi hasil konversi JSON dari field yang berbeda.

Isi dari file field1_model.dart :

```
// To parse this JSON data, do
//
//     final field1Model = field1ModelFromJson(jsonString);

import 'dart:convert';

Field1Model field1ModelFromJson(String str) =>
```

```
Field1Model.fromJson(json.decode(str));

String field1ModelToJson(Field1Model data) => json.encode(data.toJson());

class Field1Model {
  DateTime createdAt;
  int entryId;
  String field1;

  Field1Model({
    required this.createdAt,
    required this.entryId,
    required this.field1,
  });

  factory Field1Model.fromJson(Map<String, dynamic> json) => Field1Model(
    createdAt: DateTime.parse(json["created_at"]),
    entryId: json["entry_id"],
    field1: json["field1"],
  );

  Map<String, dynamic> toJson() => {
    "created_at": createdAt.toIso8601String(),
    "entry_id": entryId,
    "field1": field1,
  };
}
```

Isi dari file field2_model.dart :

```
// To parse this JSON data, do
//
//     final field2Model = field2ModelFromJson(jsonString);

import 'dart:convert';

Field2Model field2ModelFromJson(String str) =>
Field2Model.fromJson(json.decode(str));

String field2ModelToJson(Field2Model data) => json.encode(data.toJson());

class Field2Model {
  DateTime createdAt;
  int entryId;
  String field2;

  Field2Model({
    required this.createdAt,
    required this.entryId,
    required this.field2,
  });

  factory Field2Model.fromJson(Map<String, dynamic> json) => Field2Model(
    createdAt: DateTime.parse(json["created_at"]),
    entryId: json["entry_id"],
    field2: json["field2"],
  );

  Map<String, dynamic> toJson() => {
    "created_at": createdAt.toIso8601String(),
    "entry_id": entryId,
```

```
    "field2": field2,
  };
}
```

Isi dari file field3_model.dart :

```
// To parse this JSON data, do
//
//     final field3Model = field3ModelFromJson(jsonString);

import 'dart:convert';

Field3Model field3ModelFromJson(String str) =>
Field3Model.fromJson(json.decode(str));

String field3ModelToJson(Field3Model data) => json.encode(data.toJson());

class Field3Model {
  DateTime createdAt;
  int entryId;
  String field3;

  Field3Model({
    required this.createdAt,
    required this.entryId,
    required this.field3,
  });

  factory Field3Model.fromJson(Map<String, dynamic> json) => Field3Model(
    createdAt: DateTime.parse(json["created_at"]),
    entryId: json["entry_id"],
    field3: json["field3"],
  );

  Map<String, dynamic> toJson() => {
    "created_at": createdAt.toIso8601String(),
    "entry_id": entryId,
    "field3": field3,
  };
}
```

Dengan adanya model ini dapat membantu aplikasi membaca data saat berkomunikasi dengan API.

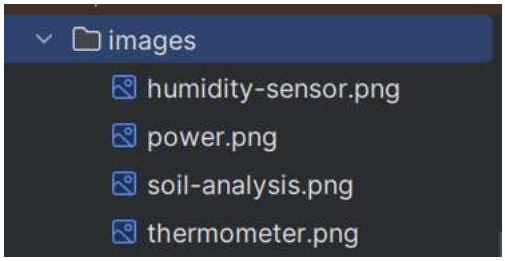
Kemudian buka file api_service.dart dan isi dengan kode berikut :

```
import 'package:dio/dio.dart';
import 'package:mcs_bab_5/models/field1_model.dart';
import '../models/field2_model.dart';
import '../models/field3_model.dart';

class ApiService{
  Dio dio = Dio();

  String readKey = "ILT5CI93S5SHTTF1";

  String field1Url =
"https://api.thingspeak.com/channels/2656478/fields/1/last.json?api_key=";
  String field2Url =
"https://api.thingspeak.com/channels/2656478/fields/2/last.json?api_key=";
  String field3Url =
```

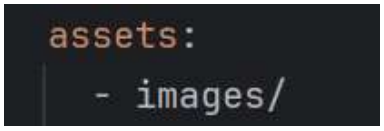
```
"https://api.thingspeak.com/channels/2656478/fields/3/last.json?api_key=";

  Future<Field1Model> getField1() async{
      try{
        final response = await dio.get("$field1Url$readKey");
        return Field1Model.fromJson(response.data);
      } catch(e){
        rethrow;
      }
  }

  Future<Field2Model> getField2() async{
    try{
      final response = await dio.get("$field2Url$readKey");
      return Field2Model.fromJson(response.data);
    } catch(e){
      rethrow;
    }
  }

  Future<Field3Model> getField3() async{
    try{
      final response = await dio.get("$field3Url$readKey");
      return Field3Model.fromJson(response.data);
    } catch(e){
      rethrow;
    }
  }
}
```

Kita menggunakan Dio untuk berkomunikasi dengan API dan mendapatkan responsenya di dalam function getField1() getField2() getField3(). Masing-masing method berguna untuk mendapatkan response dari url yang memiliki data field berbeda-beda. Response yang berbeda-beda ditangkap oleh property dari 3 class model yang sudah dibuat. Karena data yang berasal dari endpoint tersebut berbentuk JSON sehingga untuk dapat membacanya harus melalui proses konversi menggunakan property yang ada di class model yaitu fromJson.

Aplikasi pada praktikum ini diperlukan memasukkan gambar, cara untuk menambahkan gambar ke aplikasi yaitu dengan membuat folder images pada root project, letakkan gambar yang dibutuhkan ke dalam folfer tersebut dan atur dibagian pubspec.yaml dengan menghilangkan comment pada bagian assets.



Folder berisi gambar



Configure images in pubspec

Kemudian buka file app_provider.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:mcs_bab_5/models/field1_model.dart';
import 'package:mcs_bab_5/screens/main_screen.dart';
import 'package:mcs_bab_5/services/api_service.dart';
import '../models/field2_model.dart';
import '../models/field3_model.dart';

class AppProvider extends ChangeNotifier {
  TextStyle roboto14Italic = GoogleFonts.roboto(fontSize: 14, fontWeight:
FontWeight.w400,);
  TextStyle roboto14 = GoogleFonts.roboto(fontSize: 14, fontWeight:
FontWeight.w500,);
  TextStyle roboto14SemiBold = GoogleFonts.roboto(fontSize: 14, fontWeight:
FontWeight.w600,);
  TextStyle roboto14Bold = GoogleFonts.roboto(fontSize: 14, fontWeight:
FontWeight.w700,);
  TextStyle roboto16Italic = GoogleFonts.roboto(fontSize: 16, fontWeight:
FontWeight.w400,);
  TextStyle roboto16 = GoogleFonts.roboto(fontSize: 16, fontWeight:
FontWeight.w500,);
  TextStyle roboto16SemiBold = GoogleFonts.roboto(fontSize: 16, fontWeight:
FontWeight.w600,);
  TextStyle roboto16Bold = GoogleFonts.roboto(fontSize: 16, fontWeight:
FontWeight.w700,);

  TextStyle whiteRoboto14Bold = GoogleFonts.roboto(fontSize: 14,
fontWeight: FontWeight.w700, color: Colors.white);

  Color mainColor = const Color(0xff36725D);
  String loremIpsum =
  "Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever
since the 1500s, when an unknown printer took a galley of type and
scrambled it to make a type specimen book. It has survived not only five
centuries, but also the leap into electronic typesetting, remaining
essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with
desktop publishing software like Aldus PageMaker including versions of
Lorem Ipsum.";
  String thermoMeterImage = "images/thermometer.png";
  String humiditySensorImage = "images/humidity-sensor.png";
  String soilAnalysisImage = "images/soil-analysis.png";
  Field1Model? field1model;
  Field2Model? field2model;
  Field3Model? field3model;

  gotoMainScreen({required BuildContext context}){
    Navigator.push(
      context, MaterialPageRoute(builder: (context) => const
MainScreen(),),
    );
    notifyListeners();
```

```
  }

  Future getTemperature() async{
    notifyListeners();
    return field1model = await ApiService().getField1();
  }

  Future getHumidity() async{
    notifyListeners();
    return field2model = await ApiService().getField2();
  }

  Future getSoilMoisture() async{
    notifyListeners();
    return field3model = await ApiService().getField3();
  }
}
```

Dapat dilihat di dalam provider mendeklarasikan beberapa variabel dan juga function. Variabel dengan tipe String yang bernama thermoMeterImage dan humiditySensorImage, soilAnalysisImage adalah variabel untuk menyimpan path dimana gambar disimpan. Variabel dengan nama field1model, field2model, field3model adalah variabel yang nantinya menampung return saat function yang berada di class ApiService() bernama getField1(), getField2() dan getField3() dipanggil. getField1(), getField2() dan getField3() berjalan saat getTemperature(), getHumidty() dan getSoilMoisture() dipanggil.

Kemudian buka file main.dart dan isi dengan kode berikut :

```
import 'package:flutter/material.dart';
import 'package:mcs_bab_5/providers/app_provider.dart';
import 'package:mcs_bab_5/screens/opening_screen.dart';
import 'package:provider/provider.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider<AppProvider>(
          create: (context) => AppProvider()
            ..getTemperature()
            ..getHumidity()
            ..getSoilMoisture(),
        )
      ],
      child: MaterialApp(
        title: 'Mcs Bab 5',
        debugShowCheckedModeBanner: false,
        theme: ThemeData(
          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
          useMaterial3: true,
        ),
```

```
        home: const OpeningScreen()
      ),
    );
  }
}
```

Dapat dilihat ada code ..getTemperature, ..getHumidty() dan ..getSoilMoisture(). Bentuk penulisan tersebut memiliki tujuan agar tiga function tersebut langsug berjalan saat aplikasi dibuka.

Kemudian buka file opening_screen.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:mcs_bab_5/providers/app_provider.dart';
import 'package:provider/provider.dart';

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

  @override
  State<OpeningScreen> createState() => _OpeningScreenState();
}

class _OpeningScreenState extends State<OpeningScreen> {

  @override
  void initState() {
    Provider.of<AppProvider>(context, listen: false,).getTemperature();
    Provider.of<AppProvider>(context, listen: false,).getHumidity();
    Provider.of<AppProvider>(context, listen: false,).getSoilMoisture();
    // TODO: implement initState
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Consumer<AppProvider>(
      builder: (context, appProvider, child) {
        return Scaffold(
          appBar: AppBar(
          title: Text("Agro Tech" , style: appProvider.whiteRoboto14Bold,),
            centerTitle: true,
            backgroundColor: appProvider.mainColor,
          ),
          body: Center(
            child: ListView(
              shrinkWrap: true,
              physics: const NeverScrollableScrollPhysics(),
              children: [
                Container(
                  margin: const EdgeInsets.symmetric(horizontal: 20),
                  width: double.infinity,
                  child: Text(
                    appProvider.loremIpsum,
                    style: appProvider.roboto14Bold,
                    textAlign: TextAlign.justify,),
                ),

                const SizedBox(height: 30,),
```

```
                Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  children: [
                    GestureDetector(
                      child: Container(
                        margin: const EdgeInsets.symmetric(horizontal: 20),
                        padding: const EdgeInsets.symmetric(vertical: 12,
horizontal: 16),
                        decoration: BoxDecoration(
                          borderRadius: BorderRadius.circular(24),
                          color: appProvider.mainColor,
                        ),
                        child: Text("Continue", style:
appProvider.whiteRoboto14Bold,),
                      ),
                      onTap: () {
                        appProvider.gotoMainScreen(context: context);
                      },
                    )
                  ],
                )
              ],
            ),
          ),
        );
      },
    );
  }
}
```

Di bagian initState() function getTemperature(), getHumidty() dan getSoilMoisture() dipanggil lagi karena walaupun saat inisialisasi provider tiga function tersebut sudah dipanggil, hal tersebut tidak dapat menjadi jaminan bahwa function yang dipanggil dapat berjalan.

Kemudian buka file read_field.dart dan isi dengan code berikut :

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

class ReadField extends StatelessWidget {
  String result;
  Color color;
  String image;

  ReadField({
    super.key,
    required this.result,
    required this.color,
    required this.image,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      margin: const EdgeInsets.symmetric(horizontal: 24),
      padding: const EdgeInsets.symmetric(vertical: 18),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(24),
        border: Border.all(color: color, width: 4),
      ),
```

```
      child:
      Column(
        children: [
          SizedBox(
            width: MediaQuery.of(context).size.width / 5,
            height: MediaQuery.of(context).size.width / 5,
            child: Image.asset(image, fit: BoxFit.fill,),
          ),
          const SizedBox(height: 14,),
          Center(child: Text(result),),
        ],
      ),
    );
  }
}
```

Di class ReafField() adalah widget yang nantinya akan digunakan di halaman aplikasi saat membaca field dari channel thingspeak. Dengan adanya class ReadField() ketika ingin menggunakan widget hanya perlu memanggil class ReafField() saja, nanti constructor yang ada di class ReafField() hanya perlu diisi saat dipanggil. Terdapat 3 bagian di dalam constructor yang harus diisi, result adalah untuk menampilkan data dari field channel thingspeak, color adalah warna untuk border widget yang dibangun dan image untuk menampilkan gambar pada widget tersebut.

Kemudian buka file main_screen.dart dan isi dengan code berikut :

```
import 'package:flutter/material.dart';
import 'package:mcs_bab_5/providers/app_provider.dart';
import 'package:mcs_bab_5/widgets/read_field.dart';
import 'package:provider/provider.dart';

class MainScreen extends StatelessWidget {
  const MainScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Consumer<AppProvider>(
      builder: (context, appProvider, child) {
        return Scaffold(
          appBar: AppBar(
            title: Text("Agro Tech" , style:
appProvider.whiteRoboto14Bold,),
            centerTitle: true,
            automaticallyImplyLeading: false,
            backgroundColor: appProvider.mainColor,
          ),
          body: Center(
            child: ListView(
              shrinkWrap: true,
              physics: const NeverScrollableScrollPhysics(),
              children: [
                // temperature
                ReadField(
                  result: appProvider.field1model!.field1,
                  color: appProvider.mainColor,
                  image: appProvider.thermoMeterImage,
                ),

                const SizedBox(height: 20,),
```

```
                //humidity
                ReadField(
                  result: appProvider.field2model!.field2,
                  color: appProvider.mainColor,
                  image: appProvider.humiditySensorImage,
                ),

                const SizedBox(height: 20,),

                //soil moisture
                ReadField(
                  result: appProvider.field3model!.field3,
                  color: appProvider.mainColor,
                  image: appProvider.soilAnalysisImage,
                ),
              ],
            ),
          ),
        );
      },
    );
  }
}
```

Dari class ini dibuat suatu halaman untuk menampilkan data yang ada di Thingspeak melalui JSON. Widget yang digunakan adalah ReadField dan penggunaannya hanya perlu memanggil classnya saja. Ketika class ReadField() dipanggil, class ReadField() memiliki constructor yang di dalamnya terdapat 3 bagian yang harus diisi, result diisi dengan nilai field yang didapat dari channel Thingspeak, color diisi dengan warna, image diisi dengan gambar yang sudah disiapkan.

LAPORAN PENDAHULUAN (LP)

1. Berikan penjelasan apa itu Thingspeak!
2. Berikan penjelasan apa itu JSON!
3. Berikan penjelasan apa itu serialisasi dan deserialisasi pada JSON!
4. Berikan macam-macam cara consume API dan berikan penjelasannya!

LAPORAN AKHIR (LA)

1. Berikan kesimpulan pada bab 5!