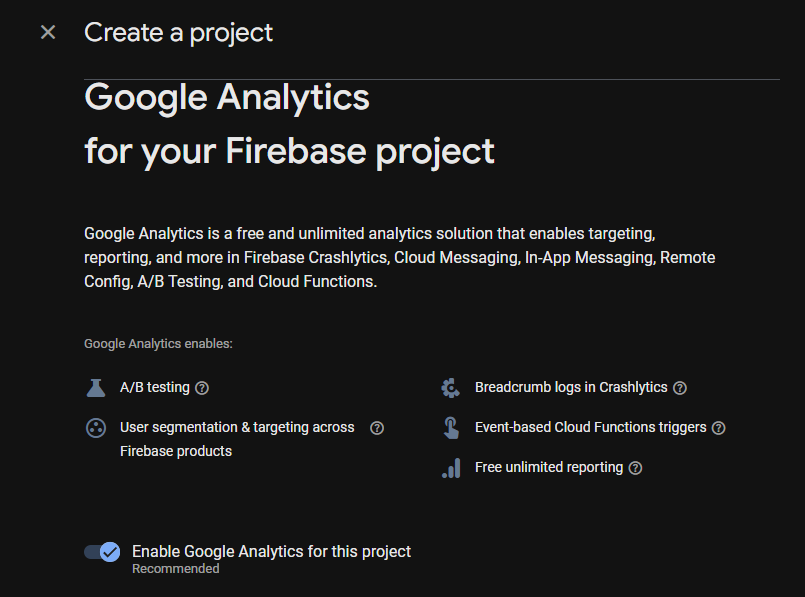
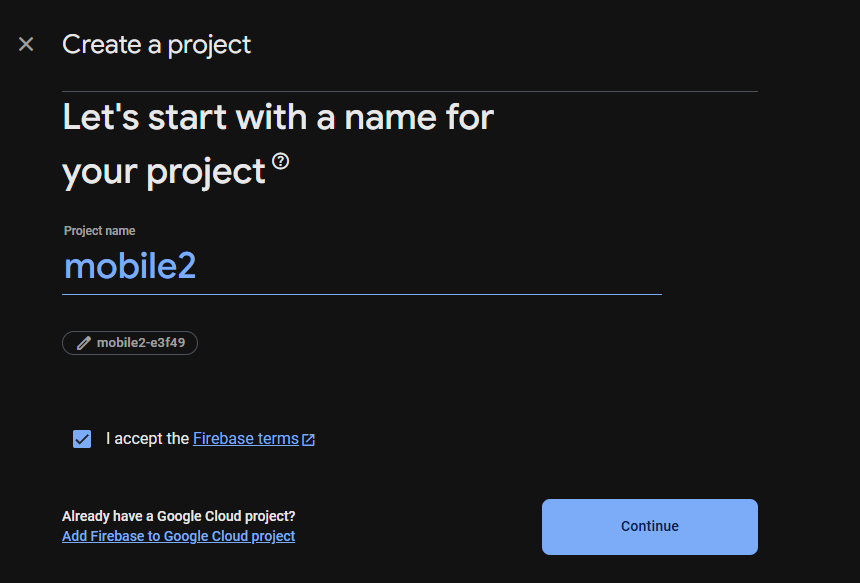
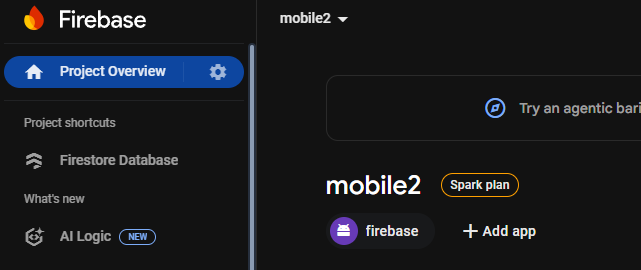
TUGAS 5

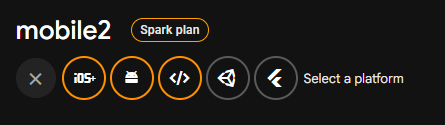
| NIM | : 22.230.0001 |
| --- | --- |
| NAMA | : Fadhli Hilman Saputra |
| KELAS | : 7P51 |

1. Membuat akun firebase pada <https://console.firebase.google.com/u/0/mfa>
2. Buat project baru dengan firebase



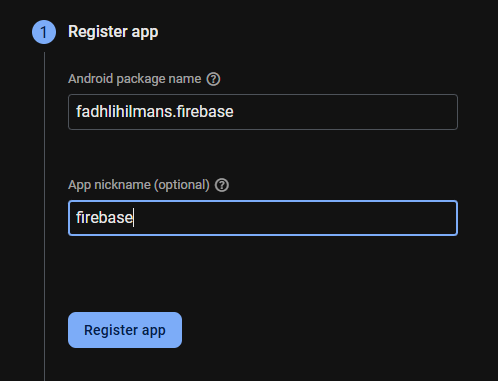
1. Pilih “+add app”, lalu pilih “icon android”



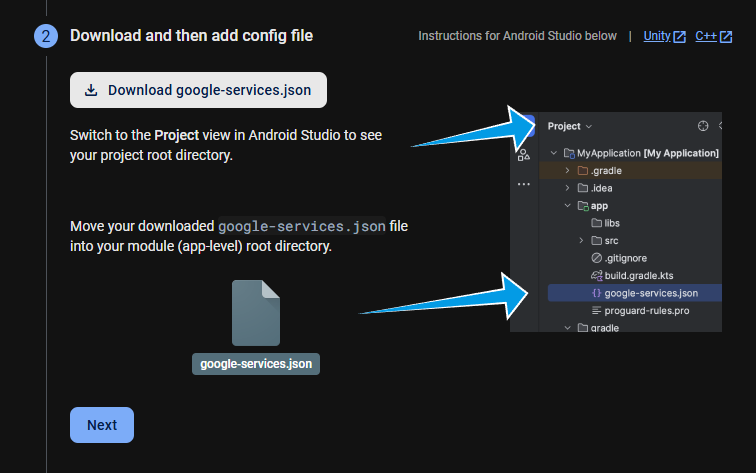


1. Ikuti step berikut:

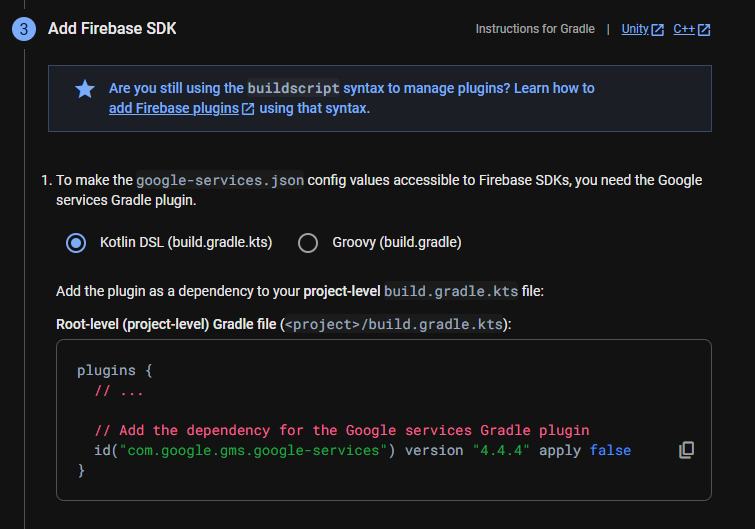
* Lengkapi package name,



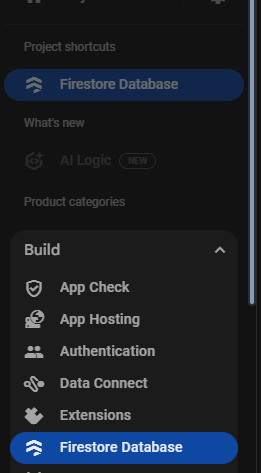
* download google-services.json



* selanjutnya pilih kotlin dsl (build.gradle.kts)

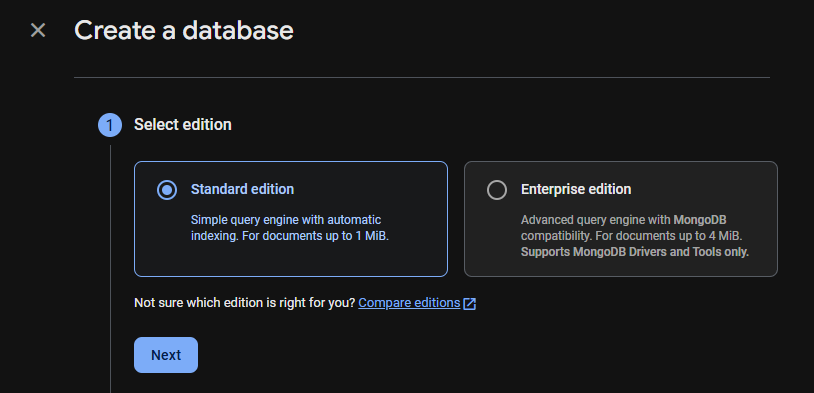


1. Buat database baru dengan cara pilih build 🡪 firebase database pada sidebar

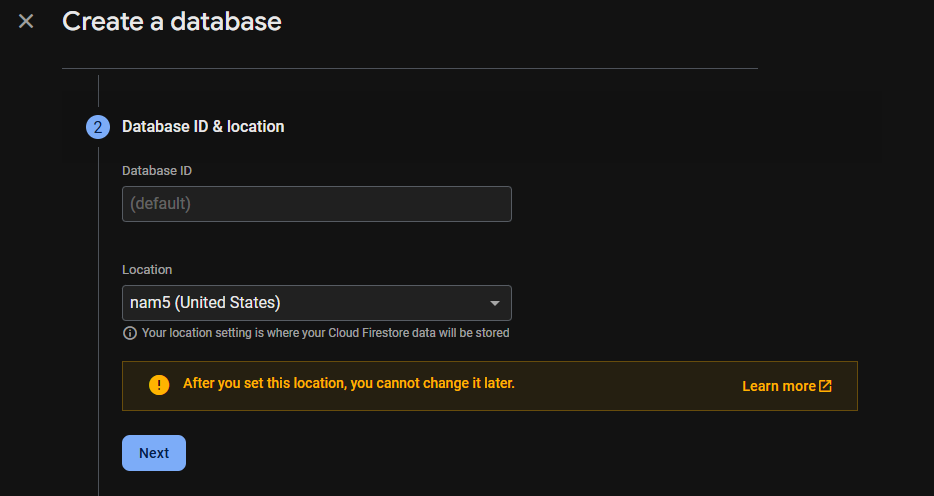


1. Selanjutnya create database dan ikuti step berikut:

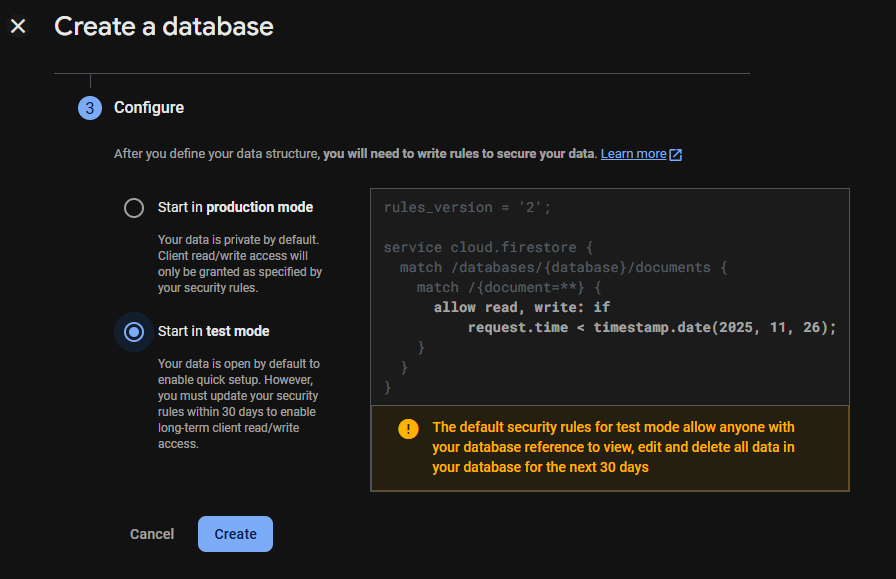
* pilih standart edition



* ID location biarkan default

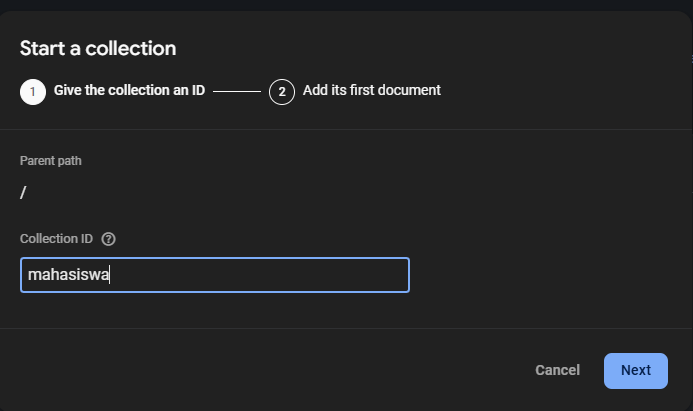


* Pilih Start in test mode

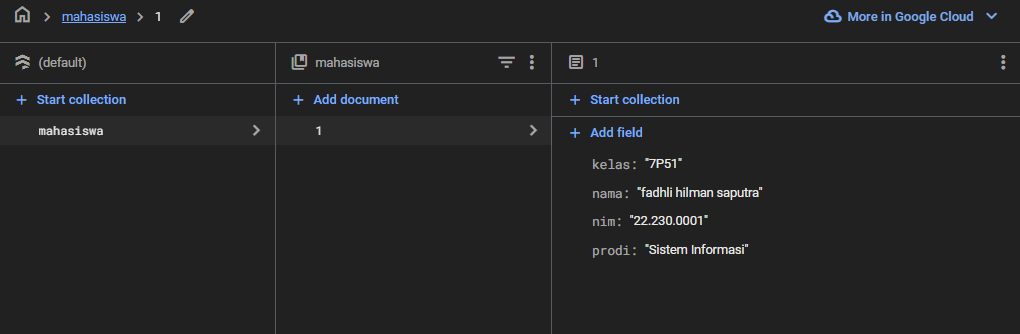
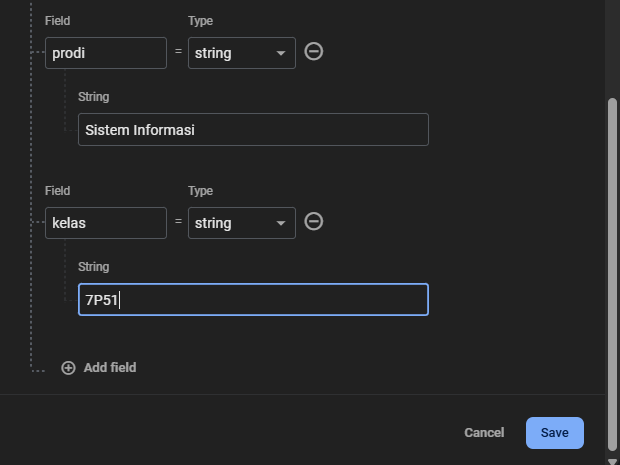
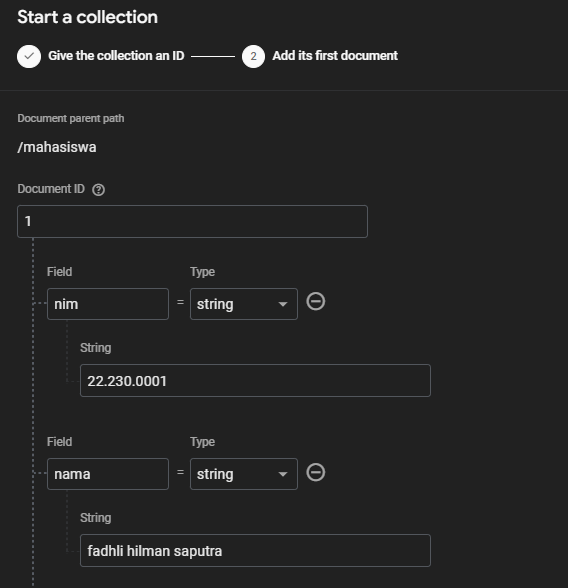


1. Selanjutnya pilih Start Collection

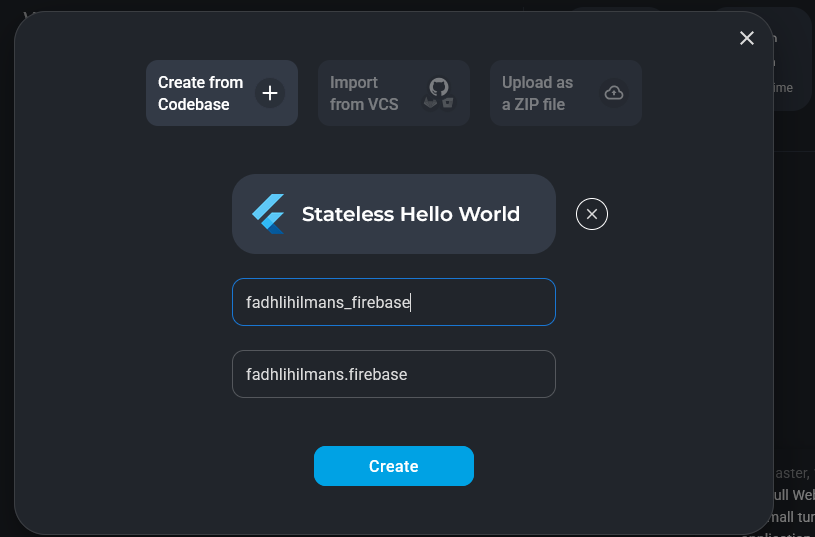
* Collection id isi “mahasiswa” sebagai nama table



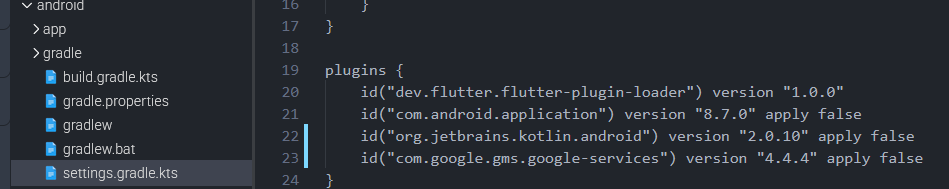
* Inputkan nim, nama, prodi, kelas serta tipe data dan value



1. Buat project pada <https://flutlab.io/>
2. Create project, sesuaikan packagename dengan penamaan android packagename pada firebase sebelumnya, disini saya menggunakan nama “fadhlihilmans.firebase”



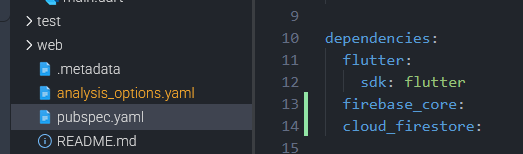
1. Pada file : android/setting.gradle.kts



Penjelasan :

* Ubah Versi KotlinMenjadi: 2.0.10 agar kompatible dengan versi terbaru
* Tambahkan **id("com.google.gms.google-services") version "4.4.4"** apply false

Ini adalah plugin Google Services. Ia digunakan untuk memproses file konfigurasi Google (seperti google-services.json) dan secara otomatis menambahkan dependensi yang diperlukan untuk layanan seperti Firebase

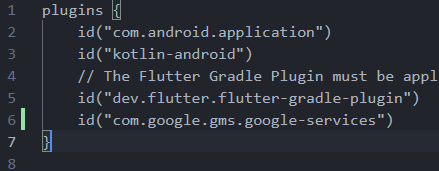
1. Buka pubspec.yaml lalu tambahkan library yang diperlukan:  
   

Penjelasan :

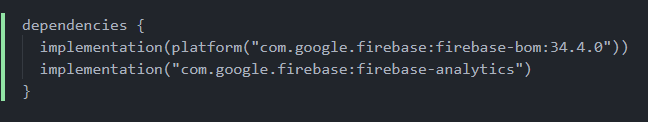
* Firebase\_core : Diperlukan sebagai inisialisasi layanan Firebase. wajib menyertakan package ini agar aplikasi Flutter dapat terhubung dengan firebase
* Cloud\_firestore : Diperlukan secara spesifik untuk menggunakan layanan basis data dari Google, yaitu Cloud Firestore

1. Buka file : android/app/build.gradle.kts

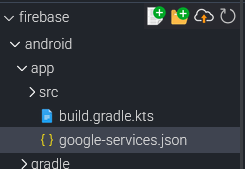
* Tambahkan id("com.google.gms.google-services")



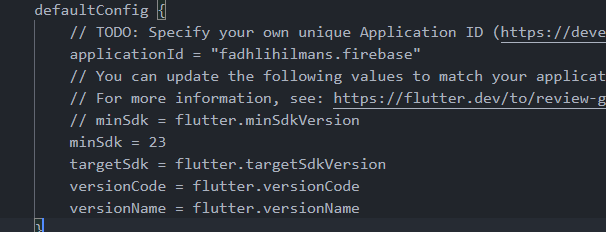
* Tambahkan dependensi berikut:



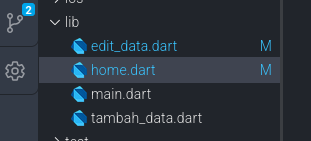
1. Letakkan (upload) file google-services.json yang sudah didownload pada android/app/



1. Ganti minSdkVersion di file android/app/build.gradle.kts menjadi 23 agar dapat menggunakan fitur Firebase



1. pada folder “lib/” siapkan halaman “tambah\_data.dart”, “edit\_data.dart”, “home.dart”, “main.dart”



1. Menyiapkan halaman tambah\_data.dart

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'home.dart';

class TambahData extends StatefulWidget {

@override

\_TambahDataState createState() => \_TambahDataState();

}

class \_TambahDataState extends State<TambahData> {

TextEditingController nimController = TextEditingController();

TextEditingController namaController = TextEditingController();

TextEditingController prodiController = TextEditingController();

TextEditingController kelasController = TextEditingController();

void addData() {

DocumentReference documentReference = FirebaseFirestore.instance

.collection('mahasiswa')

.doc(nimController.text);

Map<String, dynamic> mhs = ({

"nim": nimController.text,

"nama": namaController.text,

"prodi": prodiController.text,

"kelas": kelasController.text,

});

documentReference

.set(mhs)

.whenComplete(() => print('${nimController.text} created'));

Navigator.pop(context);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ADD DATA"),

),

body: Padding(

padding: EdgeInsets.all(10.0),

child: ListView(

children: <Widget>[

Text(

"Input Data Mahasiswa",

style: TextStyle(

color: Colors.red,

fontStyle: FontStyle.italic,

fontWeight: FontWeight.bold,

fontSize: 25,

),

),

SizedBox(

height: 40,

),

TextFormField(

controller: nimController,

decoration: InputDecoration(

labelText: "NIM",

),

),

TextFormField(

controller: namaController,

decoration: InputDecoration(labelText: "Nama"),

),

TextFormField(

controller: prodiController,

decoration: InputDecoration(labelText: "Prodi"),

),

TextFormField(

controller: kelasController,

decoration: InputDecoration(labelText: "Kelas"),

),

SizedBox(

height: 50,

),

ElevatedButton(

onPressed: () {

addData();

Navigator.pushAndRemoveUntil(

context,

MaterialPageRoute(builder: (context) => Home()),

(Route<dynamic> route) => false,

);

},

child: Text("Simpan"),

),

],

),

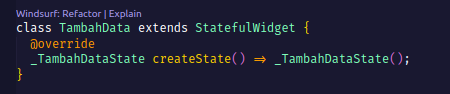
),

);

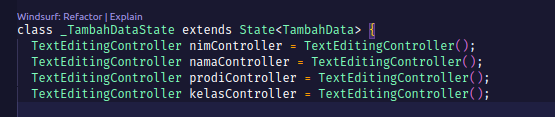
}

}

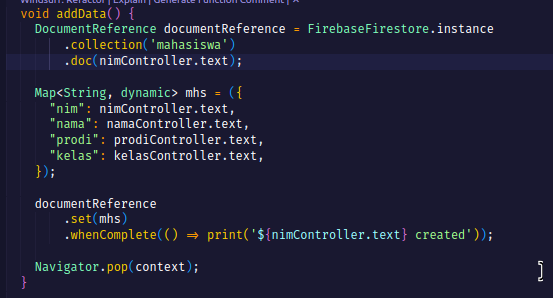
Penjelasan :



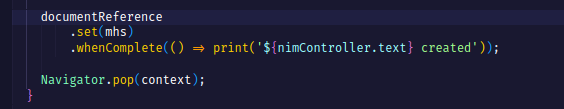
* Mendeklarasikan kelas TambahData sebagai StatefulWidget,



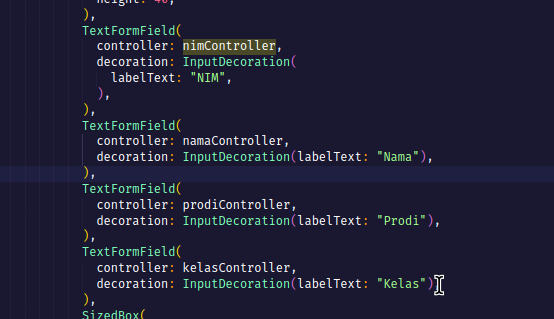
* Membuat sebuah objek controller untuk TextFormField, yaitu kolom input untuk nim, nama, prodi, kelas. Objek ini berfungsi sebagai jembatan untuk teks yang diinput oleh pengguna pada kolom tersebut.



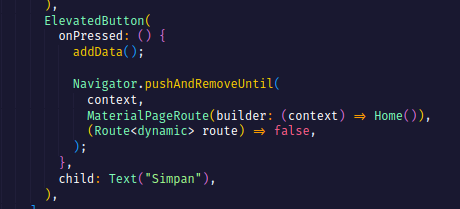
* “void addData()” Mendefinisikan sebuah metode (fungsi) bernama addData. Metode ini berisi logika utama untuk mengambil data dari semua controller dan memproses penyimpanannya ke database Cloud Firestore.
* "DocumentReference documentReference = FirebaseFirestore.instance.collection('mahasiswa').doc(nimController.text);" membuat sebuah deklarasi spesifik di dalam database Cloud Firestore. Ini memberitahu sistem untuk Mengakses instance (layanan) FirebaseFirestore, Melihat ke dalam collection (nama table) bernama mahasiswa ,Menargetkan sebuah dokumen yang ID-nya diambil dari teks yang ada di nimController.
* "Map<String, dynamic> mhs = ({ ... });" Membuat sebuah objek Map (struktur data dengan format key-value) yang berisi data mahasiswa. "nim", "nama", "prodi" , "kelas" akan menjadi nama field di dalam Firestore.



* perintah eksekusi untuk mengirim data. Metode .set(mhs) akan mengirim data yang ada di dalam Map mhs ke lokasi yang sudah ditentukan oleh documentReference. Jika data dengan ID (NIM) tersebut belum ada, data baru akan dibuat. Jika sudah ada, datanya akan ditimpa/update.



* Widget antarmuka tampilan yang menampilkan kolom input teks. Properti controller pada setiap TextFormField (seperti controller: nimController, controller: namaController, controller: prodiController, dan controller: kelasController) merupakan bagian yang menghubungkan dengan variabel controller-nya masing-masing, sehingga data yang diinput pada setiap kolom (NIM, Nama, Prodi, dan Kelas) dapat dibaca dan dikelola

.

* onPressed: () { addData(); ... }, fungsi yang mengeksekusi ketika ElevatedButton ("Simpan") diklik. Di dalam fungsi ini, memanggil metode addData() untuk memulai proses penyimpanan data ke Firebase.
* “Navigator.pushAndRemoveUntil(context, ... (Route<dynamic> route) => false);” Perintah yang dieksekusi setelah memanggil addData(). Perintah ini akan mengarahkan ke halaman Home.

1. Menyiapkan halaman edit\_data,dart

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'home.dart';

class EditData extends StatefulWidget {

*final* String? nim;

*final* String? nama;

*final* String? prodi;

*final* String? kelas;

EditData({this.nim, this.nama, this.prodi, this.kelas});

@override

\_EditDataState createState() => \_EditDataState();

}

class \_EditDataState extends State<EditData> {

TextEditingController nimController = new TextEditingController();

TextEditingController namaController = new TextEditingController();

TextEditingController prodiController = new TextEditingController();

TextEditingController kelasController = new TextEditingController();

void editData() {

DocumentReference documentReference =

FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

Map<String, dynamic> mhs = ({

"nim": widget.nim,

"nama": namaController.text,

"prodi": prodiController.text,

"kelas": kelasController.text

});

*// update data to Firebase*

documentReference

.update(mhs)

.whenComplete(() => print('${widget.nim} updated'));

}

void deleteData() {

DocumentReference documentReference =

FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);

*// delete data from Firebase*

documentReference

.delete()

.whenComplete(() => print('${widget.nim} deleted'));

}

void konfirmasi() {

AlertDialog alertDialog = AlertDialog(

content: Text("Apakah anda yakin akan menghapus data '${widget.nama}'"),

actions: <Widget>[

ElevatedButton(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.red,

),

child: Text(

"OK DELETE!",

style: TextStyle(color: Colors.black),

),

onPressed: () {

deleteData();

Navigator.pushAndRemoveUntil(

context,

MaterialPageRoute(builder: (context) => Home()),

(Route<dynamic> route) => false,

);

},

),

ElevatedButton(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.green,

),

child: Text("CANCEL", style: TextStyle(color: Colors.black)),

onPressed: () => Navigator.pop(context),

),

],

);

showDialog(

context: context,

builder: (BuildContext context) {

return alertDialog;

},

);

}

@override

void initState() {

nimController = TextEditingController(text: widget.nim);

namaController = TextEditingController(text: widget.nama);

prodiController = TextEditingController(text: widget.prodi);

kelasController = TextEditingController(text: widget.kelas);

super.initState();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("EDIT DATA"),

),

body: Padding(

padding: EdgeInsets.all(10.0),

child: ListView(

children: <Widget>[

Text(

"Ubah Data Mahasiswa",

style: TextStyle(

color: Colors.red,

fontStyle: FontStyle.italic,

fontWeight: FontWeight.bold,

fontSize: 25,

),

),

SizedBox(

height: 40,

),

TextFormField(

controller: nimController,

decoration: InputDecoration(

labelText: "NIM",

),

),

TextFormField(

controller: namaController,

decoration: InputDecoration(labelText: "Nama"),

),

TextFormField(

controller: prodiController,

decoration: InputDecoration(labelText: "Prodi"),

),

TextFormField(

controller: kelasController,

decoration: InputDecoration(labelText: "Kelas"),

),

SizedBox(

height: 50,

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: [

ElevatedButton(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.orange,

),

onPressed: () {

editData();

Navigator.pushAndRemoveUntil(

context,

MaterialPageRoute(builder: (context) => Home()),

(Route<dynamic> route) => false,

);

},

child: Text("Ubah"),

),

ElevatedButton(

style: ElevatedButton.styleFrom(

foregroundColor: Colors.red,

),

onPressed: () {

konfirmasi();

},

child: Text("Hapus"),

),

],

),

],

),

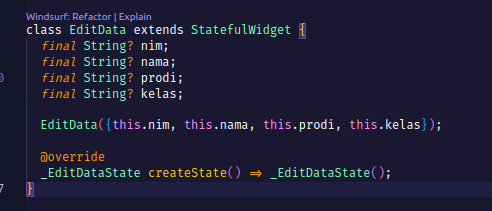
),

);

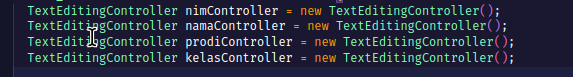
}

}

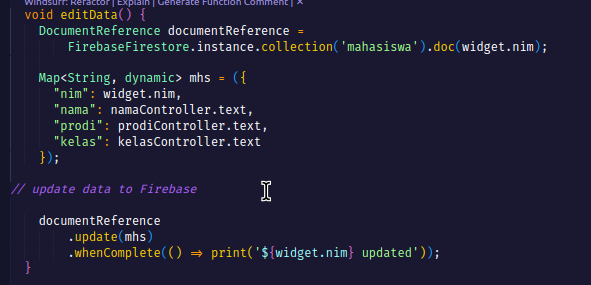
Penjelasan :



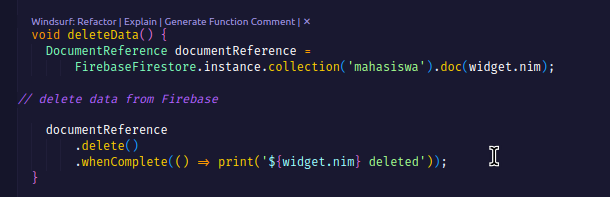
* Mendeklarasikan kelas EditData sebagai StatefulWidget
* "final String? nim; ... EditData({this.nim, this.nama, this.prodi, this.kelas});" kode yang berfungsi menerima data mahasiswa (NIM, Nama, Prodi, Kelas) dari halaman sebelumnya. Saat memilih salah satu data di halaman daftar Home, kode inilah yang "menangkap" data tersebut agar bisa ditampilkan di halaman EditData.



* Kode ini memasukkan data lama (seperti widget.nama) ke dalam kotak input "Nama". Hal yang sama berlaku untuk NIM, Prodi, dan Kelas. Jadi, saat halaman "Edit" terbuka, semua kolom sudah terisi.



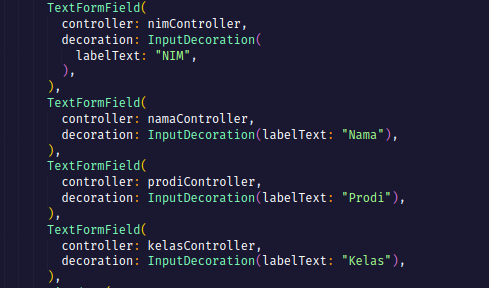
* "void editData()" Ini adalah metode / fungsi yang sebagai logika utama untuk memperbarui (meng-update) data di database Firebase.
* "DocumentReference documentReference = FirebaseFirestore.instance.collection('mahasiswa').doc(widget.nim);" Kode ini mencari data yang ingin diubah. Ia masuk ke collection mahasiswa dan mencari data yang NIM-nya (widget.nim) sesuai dengan yang sedang diedit.
* “documentReference.update(mhs)”, untuk menyimpan perubahan ke Firebase



* "void deleteData()" Ini adalah fungsi yang berisi perintah untuk menghapus data dari database.
* "documentReference.delete()", Ini adalah perintah untuk menghapus permanen dokumen yang alamatnya sudah ditemukan sebelumnya (berdasarkan widget.nim).



* "void konfirmasi()", Ini adalah fungsi yang berisi untuk memunculkan modal/dialog peringatan (pop-up) "Apakah Anda yakin?"
* "AlertDialog alertDialog = AlertDialog(...)", tombol pilihan ("OK DELETE!" dan "CANCEL").
* "onPressed: () { deleteData(); ... }", kode yang dijalankan kalau tombol "OK DELETE!" ditekan. Fungsinya memanggil deleteData() (untuk menghapus data) lalu pindah kembali ke halaman Home.
* "showDialog(context: context, ...)", kode tersebut berfungsi menampilkan modal dialog.



* Widget antarmuka tampilan yang menampilkan kolom input teks. Properti controller pada setiap TextFormField (seperti controller: nimController, controller: namaController, controller: prodiController, dan controller: kelasController)



* "ElevatedButton(onPressed: () { editData(); ... })", Ini adalah tombol "Ubah". Saat ditekan, ia memanggil fungsi editData() (untuk menyimpan perubahan) lalu kembali ke halaman Home.
* "ElevatedButton(onPressed: () { konfirmasi(); })", Ini adalah tombol "Hapus". Saat ditekan, ia tidak langsung menghapus, tapi memanggil fungsi konfirmasi() untuk memunculkan kotak peringatan dulu.

1. Menyiapkan halaman home:

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'tambah\_data.dart';

import 'edit\_data.dart';

class Home extends StatefulWidget {

@override

HomeState createState() => HomeState();

}

class HomeState extends State<Home> {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Database Firebase'),

),

floatingActionButton: FloatingActionButton(

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => TambahData(),

),

);

},

child: Icon(Icons.add),

),

body: StreamBuilder(

stream: FirebaseFirestore.instance.collection('mahasiswa').snapshots(),

builder: (context, snapshot) {

if (snapshot.hasData) {

return Container(

padding: EdgeInsets.fromLTRB(5, 10, 5, 5),

child: Card(

child: ListView.builder(

shrinkWrap: true,

itemCount: snapshot.data!.docs.length,

itemBuilder: (context, index) {

DocumentSnapshot documentSnapshot =

snapshot.data!.docs[index];

return Column(

children: [

GestureDetector(

child: ListTile(

leading: Icon(Icons.person),

title: Text(documentSnapshot["nim"]),

*// subtitle: Text(documentSnapshot["nama"]),*

subtitle: Column(

crossAxisAlignment: CrossAxisAlignment

.start, *// Agar teks rata kiri*

children: [

Text(documentSnapshot["nama"]),

Text(

'Prodi: ${documentSnapshot["prodi"]}',

style: TextStyle(fontSize: 12),

),

Text(

'Kelas: ${documentSnapshot["kelas"]}',

style: TextStyle(fontSize: 12),

),

],

),

trailing: Icon(Icons.navigate\_next\_rounded),

),

onTap: () => Navigator.push(

context,

MaterialPageRoute(

builder: (context) => EditData(

nim: documentSnapshot["nim"],

nama: documentSnapshot["nama"],

prodi: documentSnapshot["prodi"],

kelas: documentSnapshot["kelas"],

),

),

),

),

Divider(

color: Colors.black,

indent: 10,

endIndent: 10,

)

],

);

},

),

),

);

} else {

return Center(

child: CircularProgressIndicator(

backgroundColor: Colors.black,

),

);

}

},

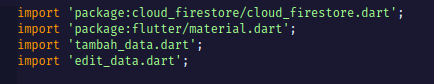
),

);

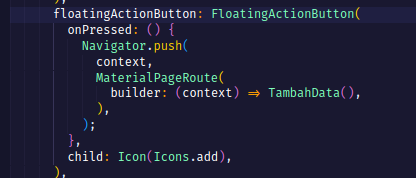
}

}

Penjelasan singkat :



* Berfungsi untuk menghubungkan halaman Home dengan dua halaman lain, yaitu TambahData dan EditData. Ini diperlukan agar aplikasi bisa berpindah halaman (navigasi) ke halaman tambah atau edit.



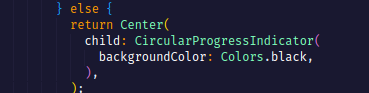
* widget yang membuat tombol dengan ikon tambah (Icons.add).



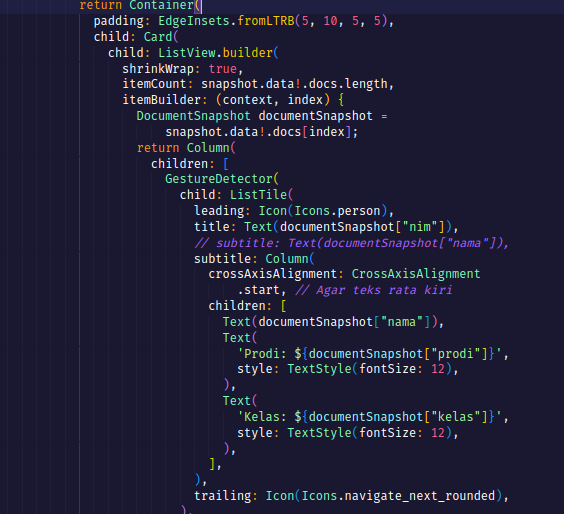
* StreamBuilder dan "stream: FirebaseFirestore.instance.collection('mahasiswa').snapshots()," berfungsi untuk mengelola data dari Firebase secara real-time. Ia akan otomatis memperbarui tampilan (ListView) setiap kali ada perubahan data (tambah, edit, atau hapus) di database Firebase pada collection 'mahasiswa'.



* Ini adalah pengecekan kondisi di dalam StreamBuilder. Kode ini memeriksa apakah snapshot (kiriman data dari Firebase) sudah berhasil diterima dan berisi data. Jika YA, maka tampilkan datanya.



* Jika snapshot.hasData adalah TIDAK (artinya data belum siap atau sedang dimuat), maka tampilkan ikon indikator loading di tengah layar.





* "ListView.builder(...)", Ini adalah widget yang membangun tampilan daftar (list) secara otomatis. Ia akan membuat satu baris item untuk setiap data mahasiswa yang diterima dari snapshot.
* "itemCount: snapshot.data!.docs.length", Kode ini memberitahu ListView.builder jumlah total item yang harus dibuat, berdasarkan berapa banyak dokumen (data) yang ada di dalam snapshot.
* "DocumentSnapshot documentSnapshot = snapshot.data!.docs[index];", Di dalam ListView.builder, kode ini berfungsi untuk mengambil satu per satu data mahasiswa dari daftar dan menyimpannya ke dalam variabel documentSnapshot.
* "child: ListTile(...)", widget yang mengatur tampilan untuk satu baris data mahasiswa, lengkap dengan ikon (leading), judul (title: Text"(documentSnapshot["nim"]))", dan subjudul (yang berisi Nama, Prodi, dan Kelas).
* "onTap: () => Navigator.push( context, MaterialPageRoute( builder: (context) => EditData( ... ) ), );", Ini adalah aksi yang terjadi saat salah satu item di daftar ditekan. Fungsinya adalah untuk membuka halaman EditData.

1. Menyiapkan halaman main.dart

import 'package:flutter/material.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'home.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(MyApp());

}

class MyApp extends StatelessWidget {

*// This widget is the root of your application.*

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Home(),

debugShowCheckedModeBanner: false,

);

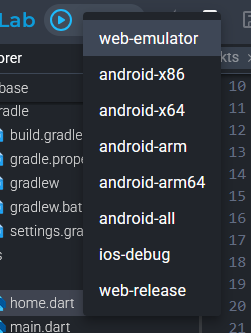
}

}

Penjelasan singkat :

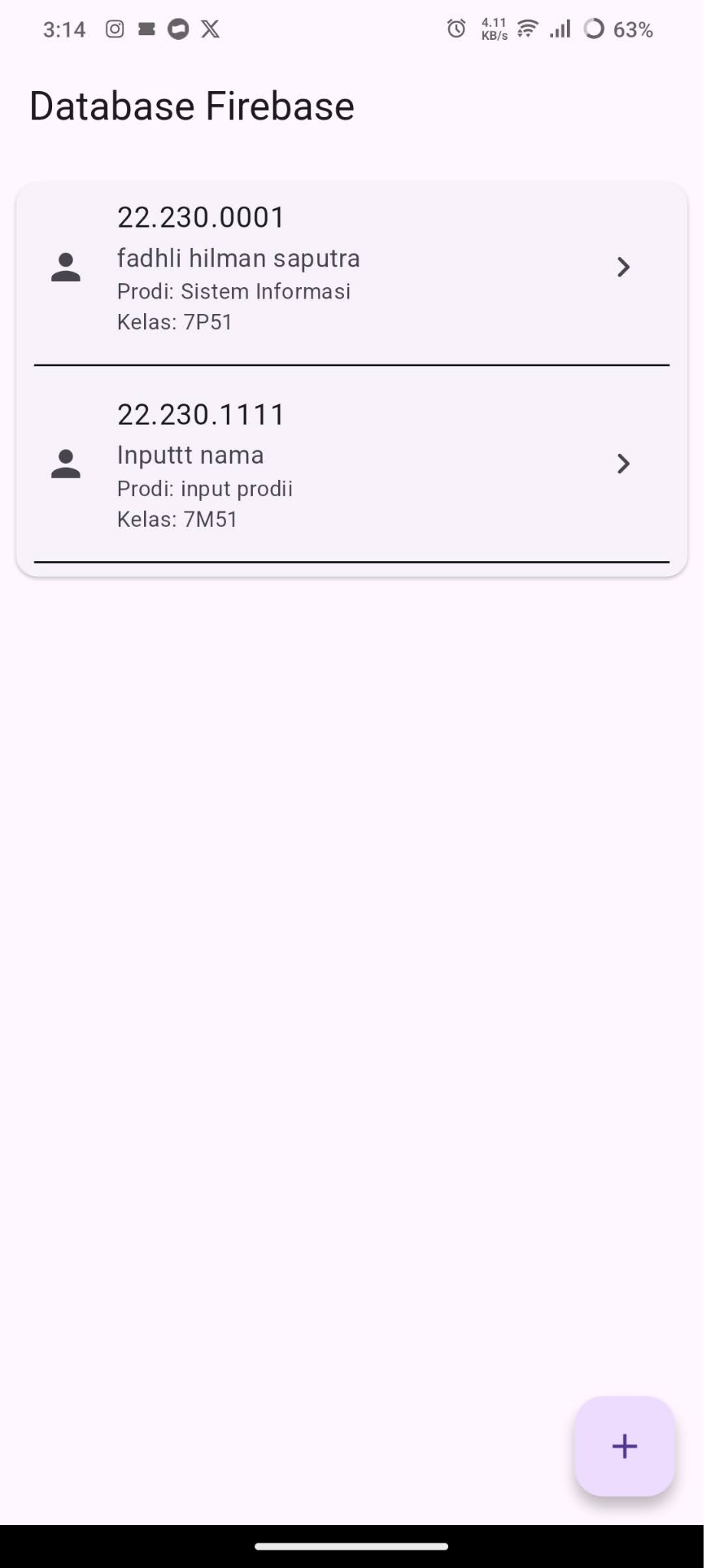
* import 'package:flutter/material.dart'; Mengimpor semua widget dan tema dasar Material Design Flutter
* import 'package:firebase\_core/firebase\_core.dart'; Mengimpor package Firebase Core yang diperlukan untuk menginisialisasi (menghubungkan) aplikasi dengan proyek Firebase Anda.
* import 'home.dart'; Mengimpor file home.dart yang berisi kelas Home, yaitu halaman utama (tampilan daftar data) yang akan ditampilkan pertama kali.
* Mengatur widget Home() (yang berisi tampilan data Firestore) sebagai halaman awal aplikasi.

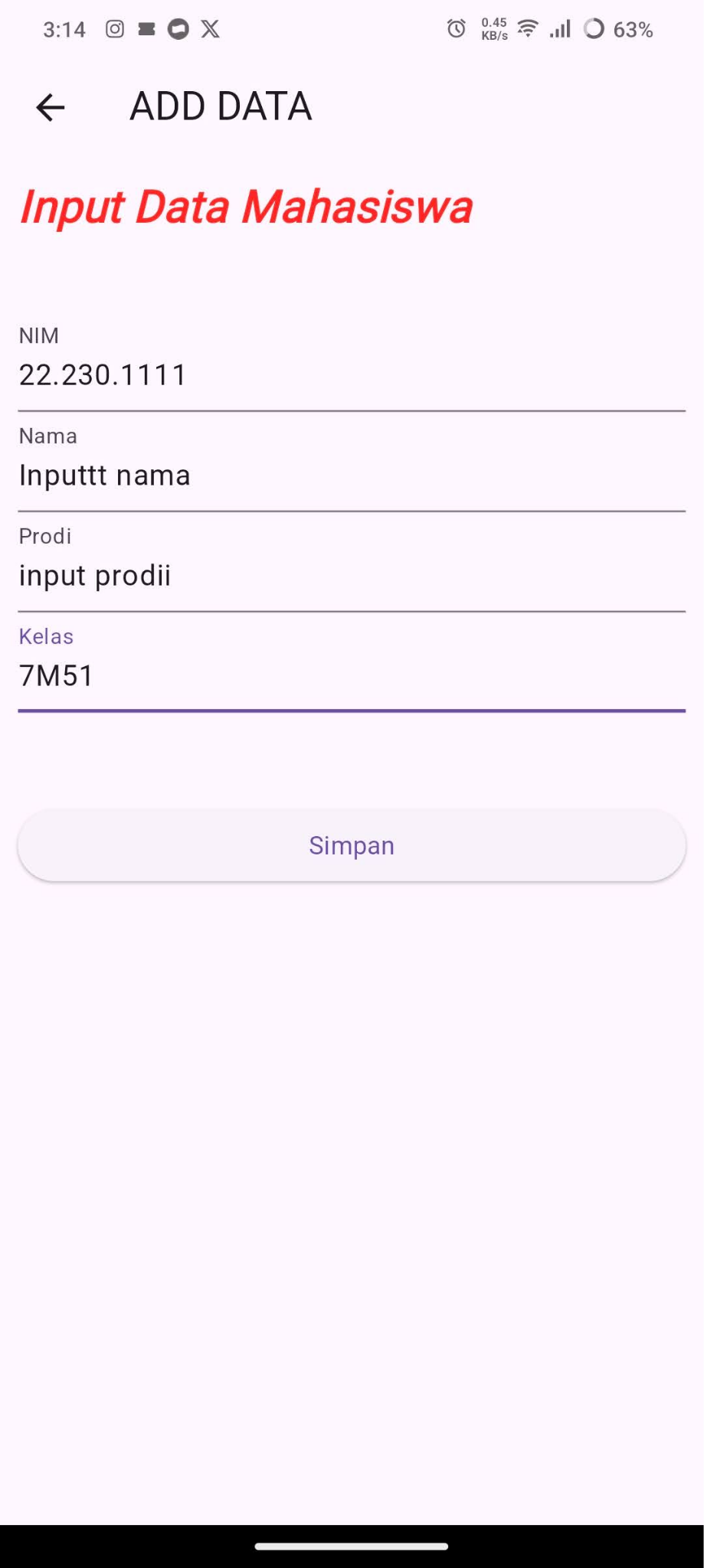
1. Jalankan projek :



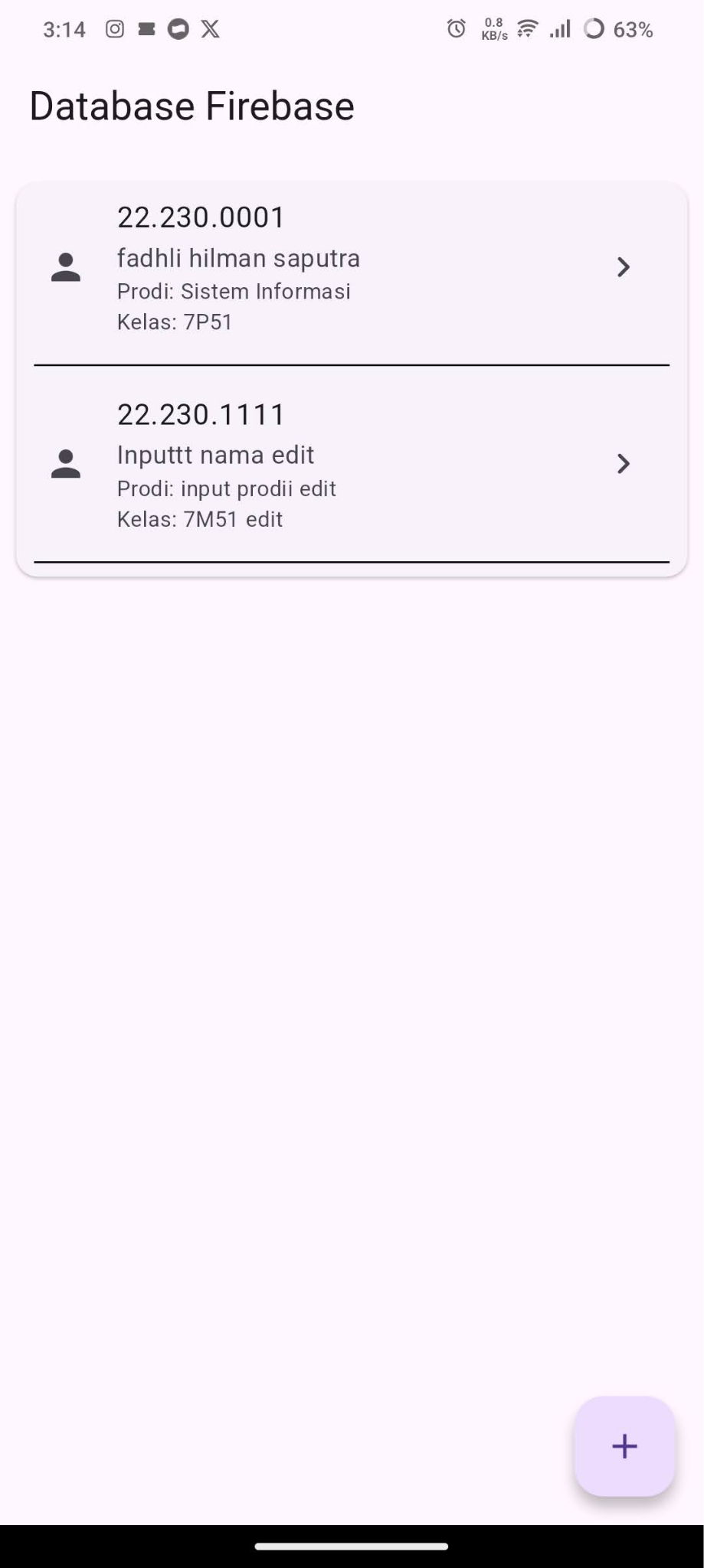
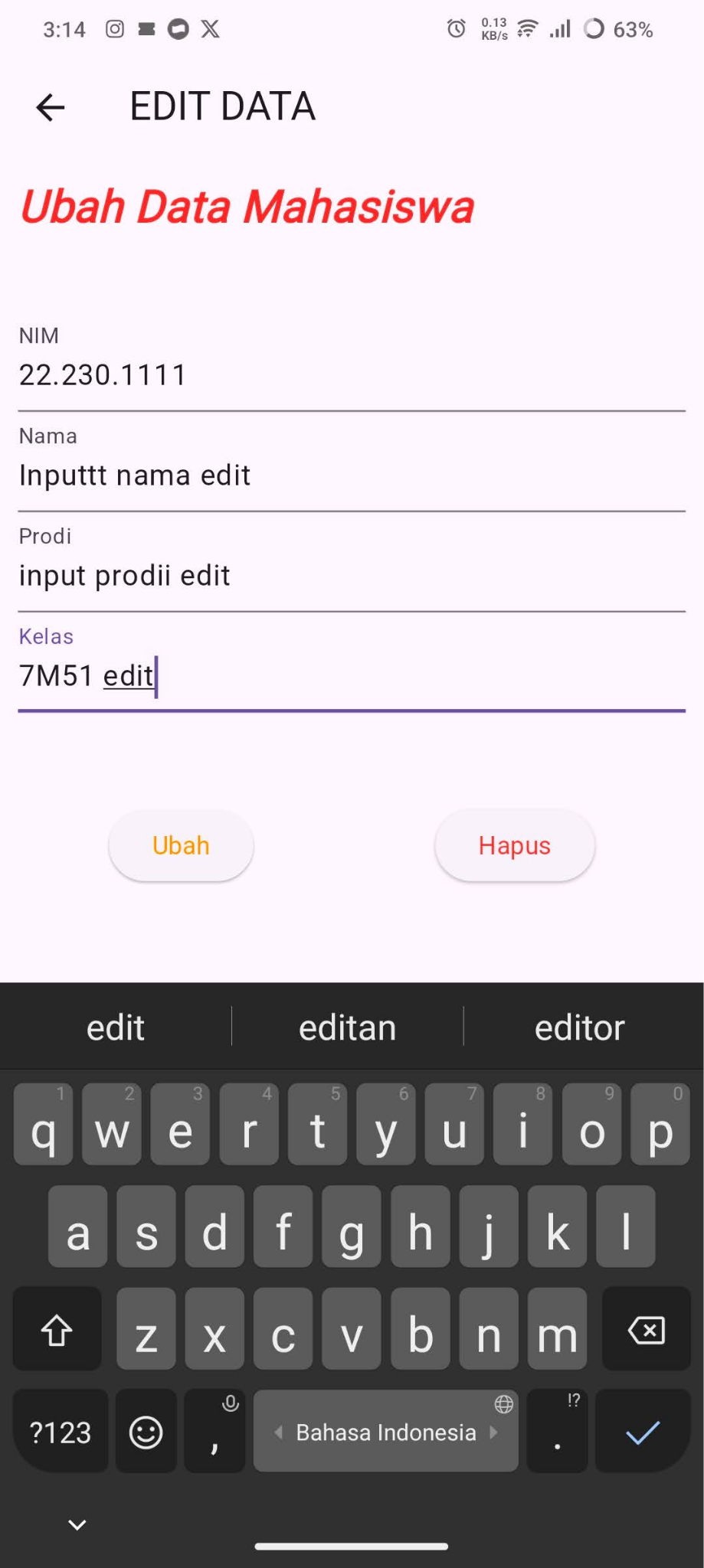
* Pilih android-arm64
* Lalu download pada hp
* Buka aplikasi hasil download tersebut
* Test Tampilan:



* Test Tambah Data



* Test Edit Data



* Test hapus data

