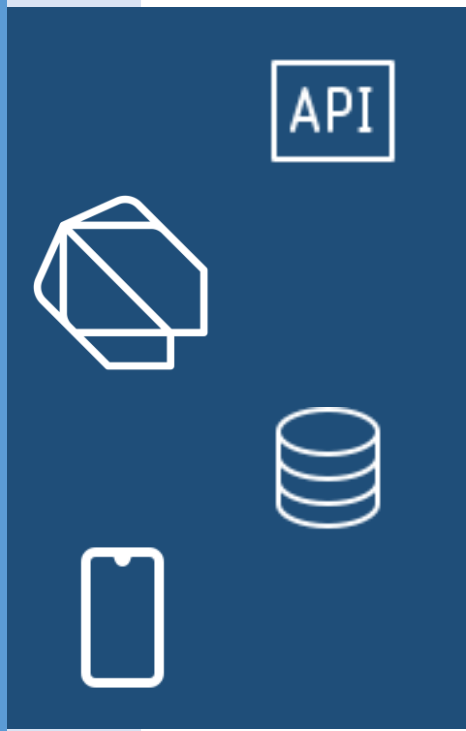
MODUL

PEMROGRAMAN BERBASIS PLATFORM

PERTEMUAN – 3

Widget dan Layout (UI) 2



PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

DAFTAR ISI

TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Membuat aplikasi sederhana yang terdiri dari Form Login, Form Register dan Homepage untuk mempelajari dasar pembuatan UI di Flutter.
2. Memahami kegunaan widget seperti Expanded, Card, ListTile, AlertDialog, dan ListView dalam pembuatan aplikasi Flutter.
3. Membuat tampilan UI untuk menampilkan list data dengan menggunakan widget seperti Expanded, Card, ListTile, dan ListView.
4. Membuat bottom navigation pada aplikasi.
5. Melakukan parsing data antar layout pada aplikasi Flutter
6. Memahami perbedaan antara Stateful dan Stateless Widget.

DASAR TEORI

A. PENGANTAR WIDGET

Karena pada Bahasa Dart, semuanya dapat didefinisikan sebagai object. Begitu juga pada widget di flutter. Widget dapat didefinisikan sebagai object, yang dimana sebagai object, widget dapat disimpan sebagai variable, direpresentasikan sebagai method, maupun sebagai kelas. Berikut contoh-contohnya :

```
static const text = Text("Widget Disimpan dalam variable");
static const List<Widget> _widgetOptions = <Widget>[
  HomePageMember(),
  BookingPage(),
]; // <Widget>[]
```

Gambar 1. Widget disimpan sebagai variabel

```
//* Widget namaMethod(typeParams1  namaParams ....)
IconButton iconNav(IconData icon, String tooltipMessage,BuildContext context , String routeLink) {
  return IconButton(
      iconSize: 40,
      onPressed: () => {
        Navigator.pushNamed(context, routeLink )
      },
      icon: Icon(
        icon,
      ), // Icon
      tooltip: tooltipMessage); // IconButton
}
```

Gambar 2. Widget direpresentasikan sebagai method

```dart
class NarrowLayout extends StatelessWidget {
  const NarrowLayout({super.key});
  @override
  Widget build(BuildContext context) {
    return PeopleList(
        onPersonTap: (person) => Navigator.of(context).push(MaterialPageRoute(
            builder: (context) => Scaffold(
              appBar: AppBar(),
              body: PersonDetail(person),

            ), // Scaffold
          ),),); // MaterialPageRoute // PeopleList
  }
}
```

Gambar 3. Widget direpresentasikan sebagai class

Kapan waktu yang tepat merepresentasikan widget sebagai variable, method ataupun kelas?

1. Variable ketika Anda menggunakan widget yang sama berulang.
2. Method ketika Anda membuat widget yang bergantung pada suatu kondisi atau parameter.
3. Class Ketika membuat widget yang kompleks.

Sebagai catatan, Anda tidak perlu selalu menyimpan widget sebagai variable, method ataupun class. Anda dapat mengevaluasi situasi khusus Anda dan memutuskan apakah Anda perlu menyimpan sebagai widget sebagai variable berdasarkan kebutuhan dan kejelasan code Anda (Pertimbangkan faktor readable, konteks dari variable, perfoma dan lain-lain).

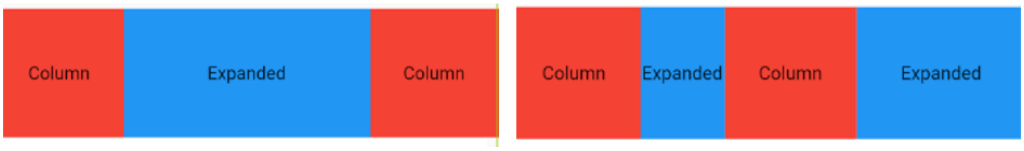
B. PENGENALAN WIDGET

1. Expanded

Expanded merupakan sebuah widget pembungkus yang dapat menampung satu widget child saja dan bertujuan untuk memperluas widget child tersebut untuk mengambil sisa ruang yang tersedia. Dengan menggunakan Expanded, widget child akan ditempatkan didalamnya akan diperbesar atau diperkecil sesuai dengan ukuran parent widget, sehingga mengisi seluruh ruang yang tersedia.

Kesalahan umum yang sering terjadi :

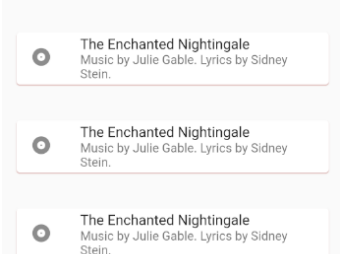
a. Expanded tidak memiliki parent widget yang memiliki konstrain(batasan). Misalnya Ketika Expanded diposisikan sebagai child dari container, namun container tersebut tidak didefinisikan width dan heightnya. Menyebabkan Expanded tidak tahu seberapa besar harus melebar atau memanjang dan akan menyebabkan error.
b. Expanded yang memiliki lebih dari satu parent widget yang memiliki konstrain bertentangan.
c. Expanded yang digunakan pada layout yang kompleks atau bersarang secara dalam (nested widget).



Gambar 4. Expanded

2. Card

Card merupakan sebuah widget untuk membuat sebuah panel/wadah konten yang telah memiliki style bawaan seperti bayangan, sudut membulat(rounded), margin default.



Gambar 5. Card

3. ListTile

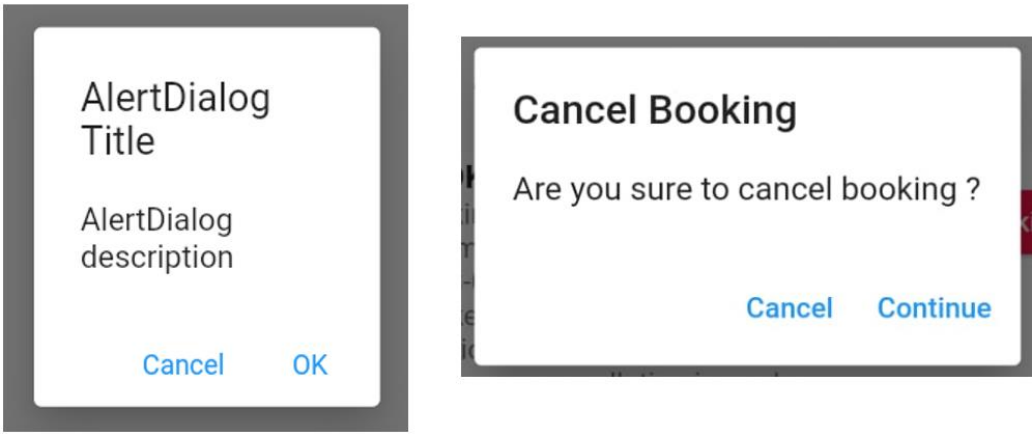
   ListTile adalah sebuah widget yang digunakan untuk menampilkan item dalam list dengan tata letak yang konsisten. ListTile memiliki struktur yang sudah ditentukan dengan elemen-elemen seperti ikon, judul, subjudul, dan elemen tambahan seperti trailing. Terkadang ListTile dibungkus didalam card.

4. ListView

   ListView merupakan sebuah widget yang digunakan untuk menampilkan list data secara dinamis pada aplikasi Flutter. Dengan menggunakan ListView, pengguna dapat mengontrol scrolling pada tampilan. Berbeda dengan Column yang akan terjadi overflow jika melebihi batas tampilan.

5. Alert Dialog

   Alert dialog merupakan sebuah jendela dialog/modal yang menampilkan suatu konten. Alert dialog dapat menampilkan konten sederhana hingga kompleks. Alert dialog dapat digunakan untuk konfirmasi, informasi detail, notifikasi, pesan kesalahan, dan lain-lain.



Gambar 6 Alert Dialog

6. Bottom Navigasi

   Komponen navigasi yang biasanya terletak pada bagian bawah layar. Bottom navigasi pada aplikasi mobile yang biasanya terdapat pada halaman home suatu aplikasi. Dengan

menggunakan bottom navigasi pengguna dapat pindah layar/halaman/fitur.

7. Key Pada Widget

   Key adalah sebuah objek yang digunakan dalam Flutter untuk memberikan identitas unik kepada widget. Penggunaan key di Flutter adalah opsional, Key biasanya digunakan Ketika aplikasi mengalami perubahan pada pohon widget (widget tree).

C. Stateful dan Stateless Widget

Pada flutter, terdapat 2 jenis widget, yaitu :

a. Stateless wdiget adalah widget yang tidak memiliki / tidak menyimpan state. Digunakan ketika tampilan yang dihasilkan selalu sama, karena data yang ditampilkan tidak pernah berubah. Contoh penggunaan : icon, logo, deskripsi statis.
b. Stateful Widget adalah widget yang menyimpan state, sehingga tampilan yang dihasilkan dapat berbeda, tergantung pada perubahan state . Contoh penggunaan stateful widget: form, animasi (cth : loading),widget yang memerlukan akses ke database.

D. Variable Scope dan Passing data antar Layout

Variable scope adalah suatu konsep fundamental dalam bahasa pemrograman. Konsep variable scope dapat berbeda antara Bahasa pemrograman yang berbeda. Setiap Bahasa pemrograman memiliki aturan dan tingkatan ruang lingkup variable yang berbeda sesuai dengan desain Bahasa tersebut. Beberapa contoh konsep variable scope yang umum dijumpai dalam bahasa pemrograman :

1. Function Scope
2. Block Scope
3. Lexical Scope : Konsep dalam pemrograman dimana akses ke variable ditentukan oleh struktur fisik kode . Artinya, Ketika Anda mendeklarasikan variable tersebut hanya dapat diakses

didalam fungsi tersebut dan fungsi-fungsi yang tertanam didalamnya.

4. Dan lainnya seperti Global Scope, Local Scope, Class Scope.

```dart
class MyWidget extends StatelessWidget {
  const MyWidget({super.key});

  //* Sebuah fungsi luar / method
  @override
  Widget build(BuildContext context) {
    String variableOnBuild = '';
    void innerFunction(){
      //* akses ke variable di fungsi/method build
      debugPrint(variableOnBuild);
    }
  innerFunction();
  return const Placeholder();
  }
}
```

Gambar 7. Inner Function dan Lexical Scope

Penempatan variable perlu diperhatikan dan dipertimbangkan dengan baik untuk meningkatkan efisiensi code, menghindari konflik nama, kinerja dan meminimalisir parsing data.

Beberapa cara untuk melakukan parsing data antar layout atau widget, yaitu :

a. Menggunakan constructor
b. Menggunakan route arguments
c. Menggunakan state management

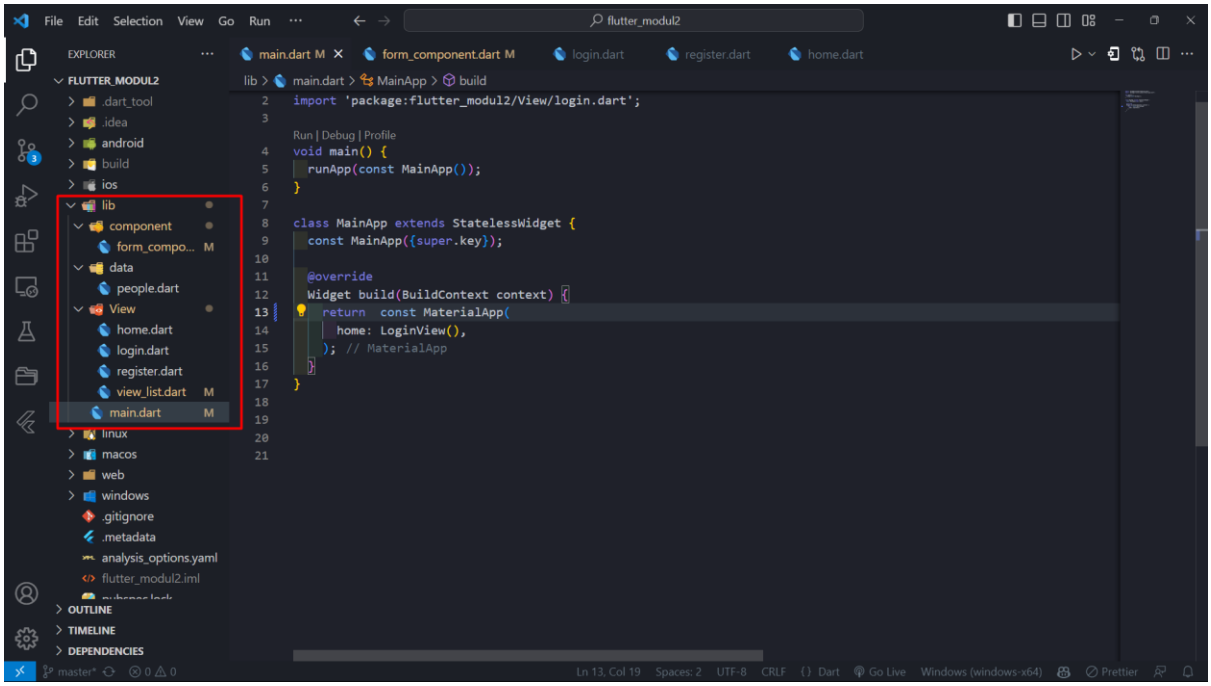
GUIDED – LAYOUT LOGIN, REGISTER, HOMEPAGE

Poin yang dipelajari dalam Guided 1 ini, yaitu :

1. Memahami cara membuat UI sederhana.
2. Memahami cara navigasi halaman di flutter.

Pada guided ini, kita akan membuat aplikasi sederhana yang terdiri dari halaman Login, Register, dan HomePage . Silahkan ikuti langkah – langkah berikut ini :

1. Sebelum memulai mengerjakan Guided, Anda dapat menonton video demo aplikasi di : Demo Widget & Layout 2.mp4 agar mendapatkan gambaran dari UI yang ingin dicapai.
2. Buka aplikasi Visual Studio Code, tekan **CTRL+SHIFT+P (Command Pallete)** pilih command Flutter: new project. Kemudian, pilih **Application (Empty)**.
3. Pilih Folder tempat Anda akan menyimpan project flutter. Kemudian, berikan nama pada project sesuai aturan pengerjaan guided (guidedlayout2_XXXX).
4. Setelah berhasil membuat project, kita akan bekerja di folder lib. Secara default, hanya akan Ada sebuah file main.dart. Anda dapat menyalin code main.dart (gambar 8) dan membuat struktur folder yang terdiri dari folder component, data, dan view. (Awalnya akan error karena LoginView, belum dibuat diabaikan saja)

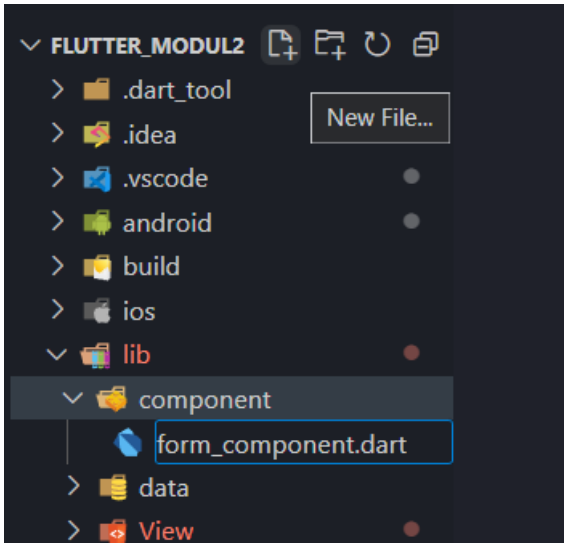


Gambar 8. Struktur Folder & code main.dart

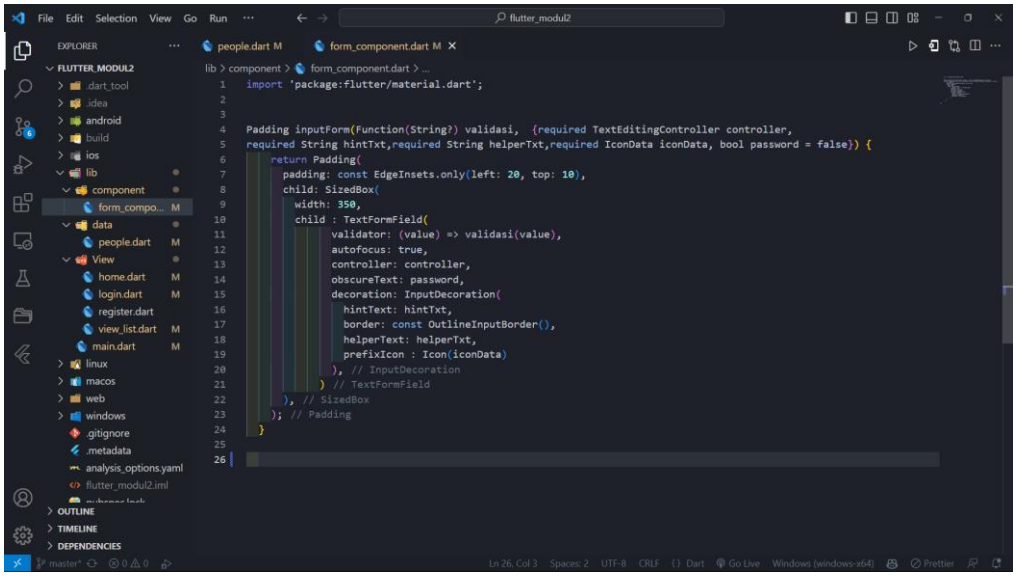
NOTE :

a. Pada project flutter, best practice / konvensi penulisan nama project dan file di dalam project flutter, menggunakan snakecase (contoh nama project : snake_case , nama file : login_page.dart).
b. Pada project flutter, sebenarnya tidak ada aturan baku mengenai struktur folder, namun sebaiknya kita mengikuti dan menyesuaikan dengan kebiasaan yang dilakukan oleh developer flutter. Agar code, kita dapat juga dipahami oleh pengembang/developer lainnya.
c. Teman-teman, disarankan mempelajari berbagai shortcut dan fitur yang dapat digunakan di visual studio code dan di project flutter. Berikut beberapa shortcut :
    a) Ctrl + shift + r atau ctrl + . untuk melakukan refactor
    b) Ketik STL/STF + tekan ctrl+spasi pilih Flutter Stateful Widget atau Flutter Stateless Widget untuk generate template.
    c) Seleksi + CTRL+SHIFT+D
    d) Dan lain-lain.

5. Sebelum memulai membuat halaman login / register, kita perlu membuat component terlebih dahulu yang nantinya digunakan pada halaman login dan register . erhatikan gambar 9 untuk cara pembuatan file dan gambar 10 untuk pengkodean form_component.dart

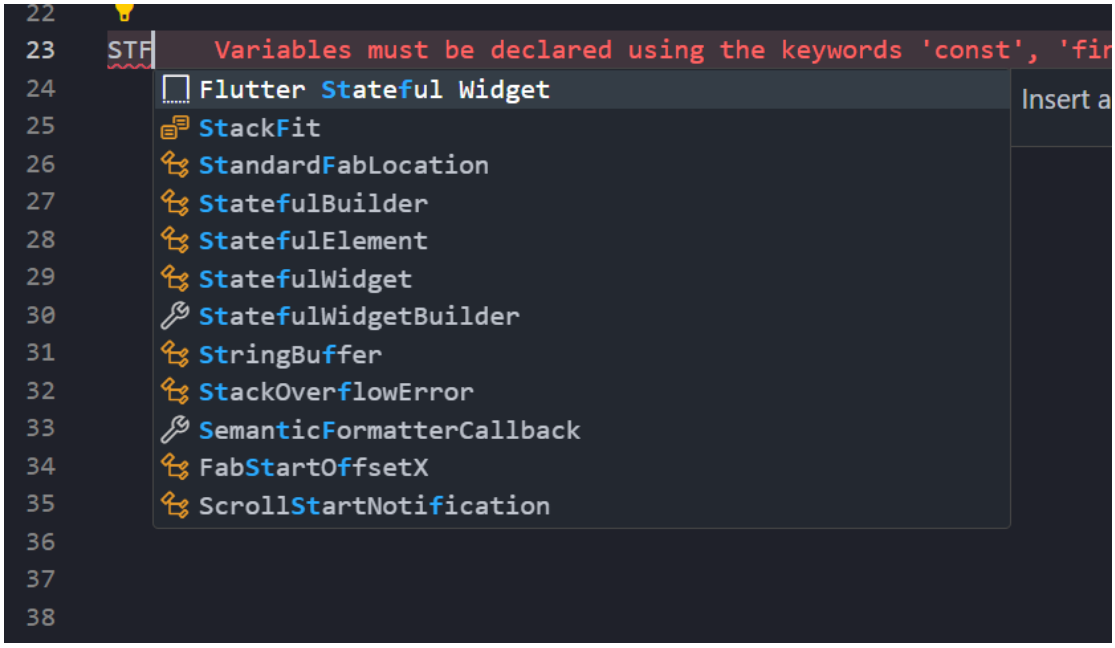


Gambar 9 Menambahkan file

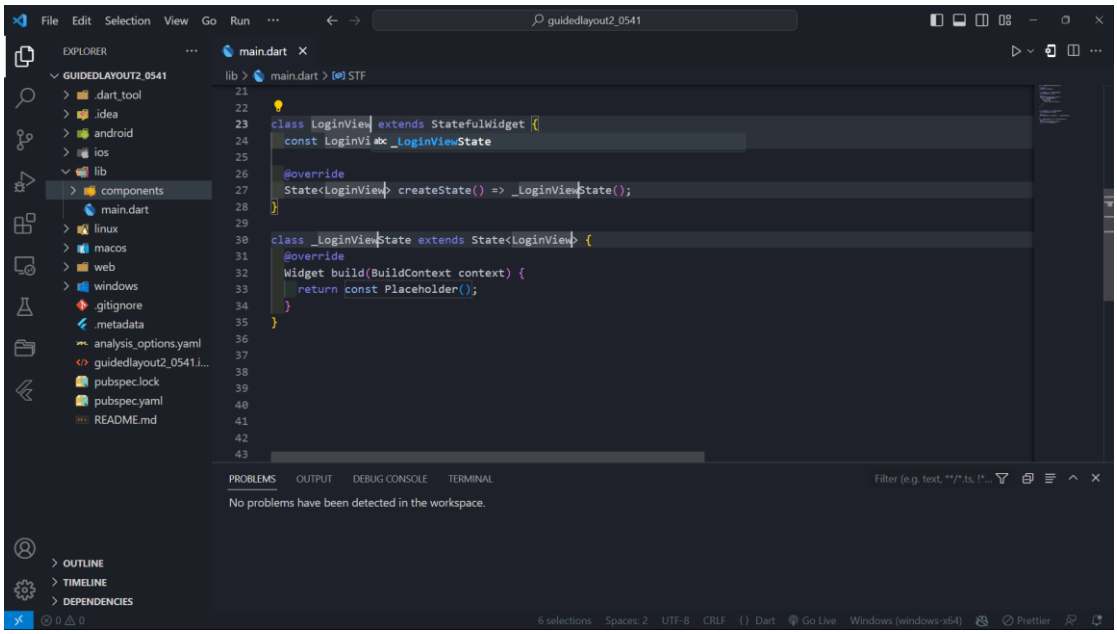


Gambar 10.Form Component

6. Kemudian, kita lanjut pada pengkodean untuk view login.dart. Anda dapat membuat template halaman terlebih dahulu dengan ketik STF kemudian tekan ctrl+spasi pilih Flutter Stateful Widget sesuai gambar 11.

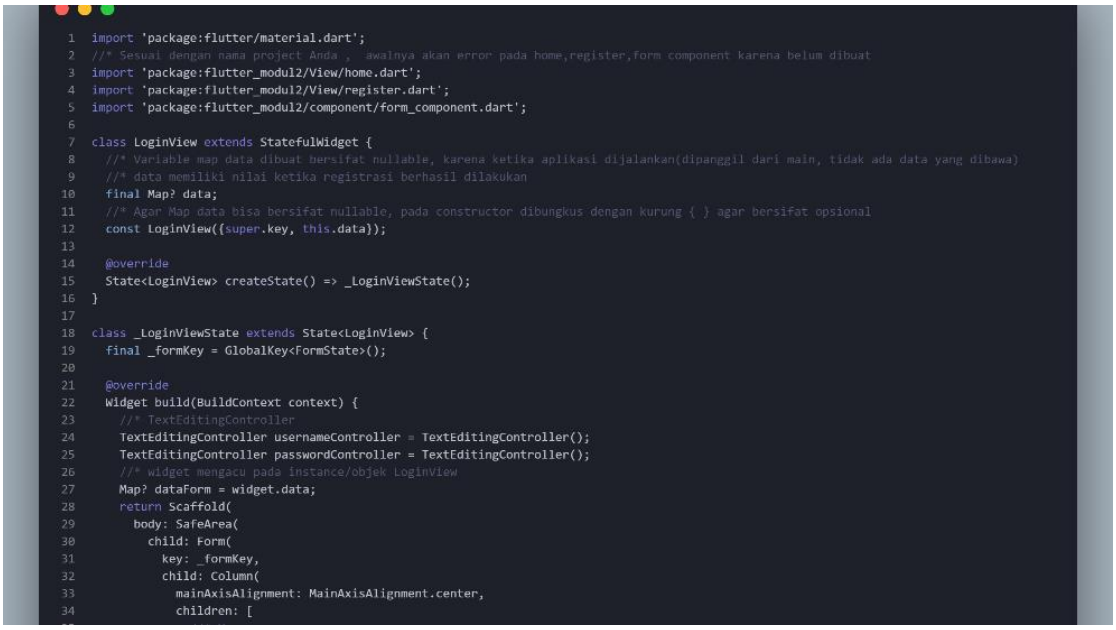


Gambar 11. Shortcut STF



Gambar 12 Multiple Cursor

7. Kemudian langsung ubah nama kelasnya (secara default setelah mengetikkan STL/STF kita akan memiliki beberapa kursor untuk mengetikkan nama classnya pada posisi tertentu pada template agar sesuai).
8. Kemudian, Anda dapat menyalin code dibawah (gambar 13 – gambar 15) :

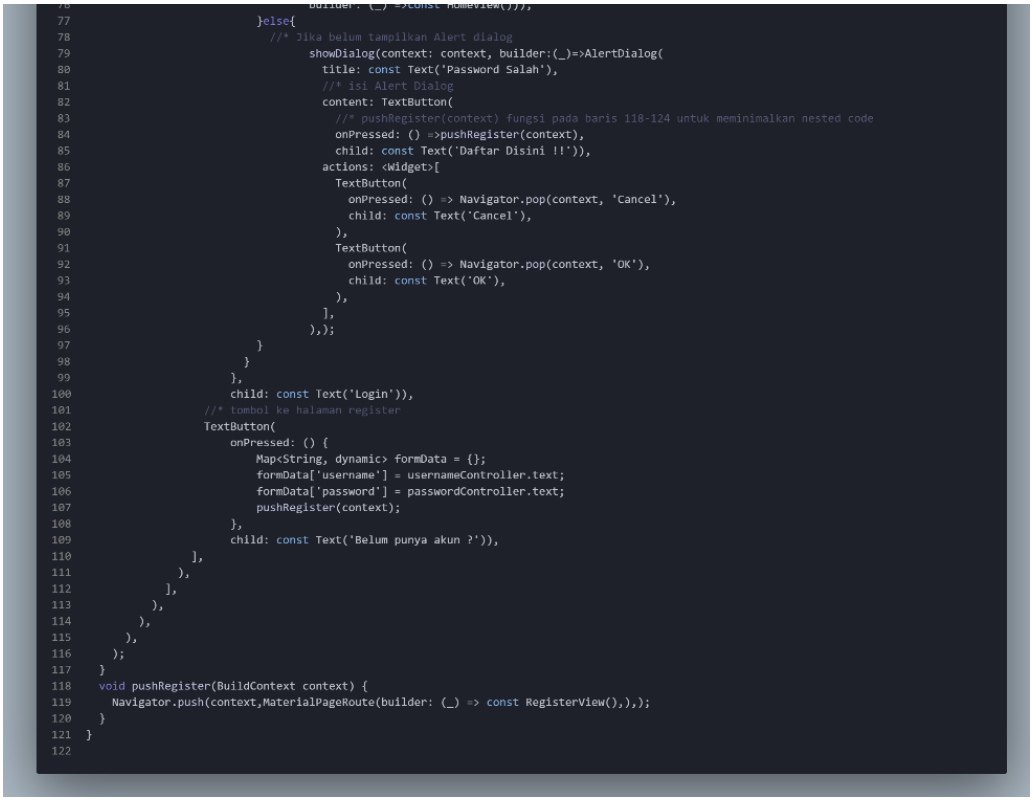


```dart
import 'package:flutter/material.dart';
//* Sesuai dengan nama project Anda ,  awalnya akan error pada home,register,form component karena belum dibuat
import 'package:flutter_modul2/View/home.dart';
import 'package:flutter_modul2/View/register.dart';
import 'package:flutter_modul2/component/form_component.dart';

class LoginView extends StatefulWidget {
  //* Variable map data dibuat bersifat nullable, karena ketika aplikasi dijalankan(dipanggil dari main, tidak ada data yang dibawa)
  //* data memiliki nilai ketika registrasi berhasil dilakukan
  final Map? data;
  //* Agar Map data bisa bersifat nullable, pada constructor dibungkus dengan kurung { } agar bersifat opsional
  const LoginView({super.key, this.data});

  @override
  State<LoginView> createState() => _LoginViewState();
}

class _LoginViewState extends State<LoginView> {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    //* TextEditingController
    TextEditingController usernameController = TextEditingController();
    TextEditingController passwordController = TextEditingController();
    //* widget mengacu pada instance/objek LoginView
    Map? dataForm = widget.data;
    return Scaffold(
      body: SafeArea(
        child: Form(
          key: _formKey,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
```

Gambar 13 login.dart (Bagian 1)

```dart
              //* Username
              inputForm((p0) {
                if (p0 == null || p0.isEmpty) {
                  return "username tidak boleh kosong";
                }
                return null;
              },
                  controller: usernameController,
                  hintTxt: "Username",
                  helperTxt: "Inputkan User yang telah didaftar",
                  iconData: Icons.person),
              //* Password
              inputForm((p0) {
                if(p0 == null || p0.isEmpty){
                  return "password kosong";
                }
                return null;
              },
                  password: true,
                  controller: passwordController,
                  hintTxt: "Password",
                  helperTxt: "Inputkan Password",
                  iconData: Icons.password),
              //* Baris yang berisi tombol login dan tombol mengarah kehalaman register
              Row(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: [
                  //*tombol login
                  ElevatedButton(
                      //* Fungsi yang dijalankan saat tombol ditekan.
                      onPressed: () {
                        //* Cek statenya sudah valid atau belum valid
                        if (_formKey.currentState!.validate()) {
                          //* jika sudah valid, cek username dan password yang diinputkan pada form telah sesuai dengan data yang dibawah
                          //* dari halaman register atau belum
                          if(dataForm!['username'] == usernameController.text && dataForm['password'] == passwordController.text )
                          {
                            //* Jika sesuai navigasi ke halaman Home
                          Navigator.push(
                              context,
                              MaterialPageRoute(
                                  builder: (_) =>const HomeView()));
```

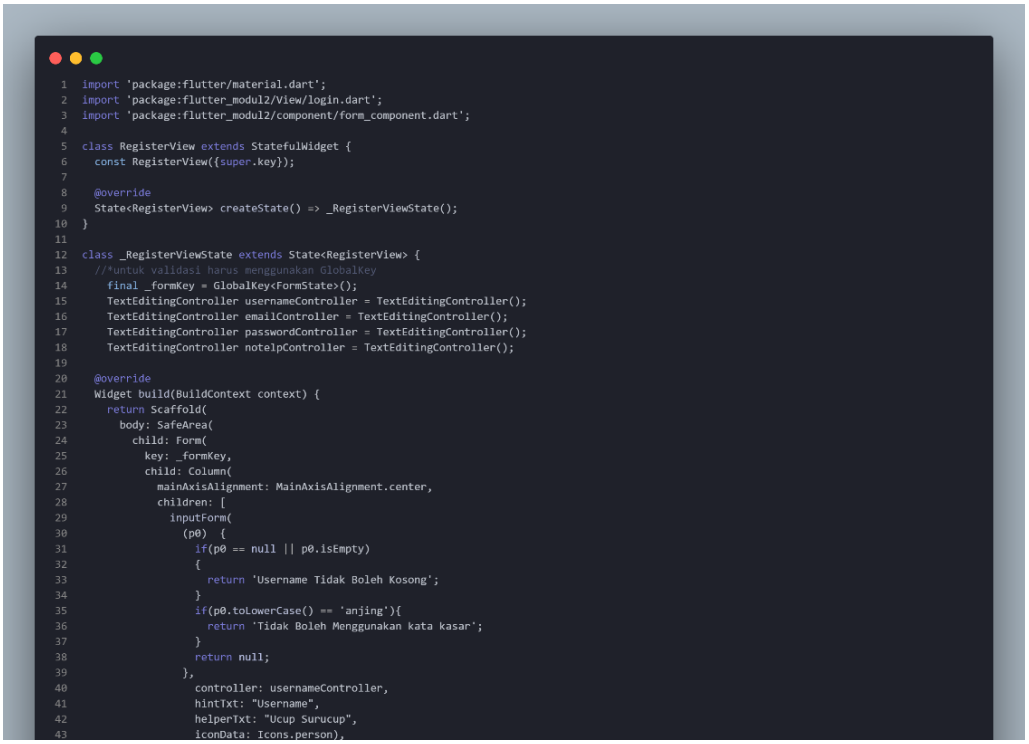
Gambar 14. login.dart (Bagian 2)



```
                        builder: (_) =>const HomeView())),
                    }else{
                      //* Jika belum tampilkan Alert dialog
                        showDialog(context: context, builder:(_)=>AlertDialog(
                          title: const Text('Password Salah'),
                          //* isi Alert Dialog
                          content: TextButton(
                            //* pushRegister(context) fungsi pada baris 118-124 untuk meminimalkan nested code
                            onPressed: () =>pushRegister(context),
                            child: const Text('Daftar Disini !!')),
                          actions: <Widget>[
                            TextButton(
                              onPressed: () => Navigator.pop(context, 'Cancel'),
                              child: const Text('Cancel'),
                            ),
                            TextButton(
                              onPressed: () => Navigator.pop(context, 'OK'),
                              child: const Text('OK'),
                            ),
                          ],
                        ),);
                    }
                  }
                },
                child: const Text('Login')),
              //* tombol ke halaman register
              TextButton(
                  onPressed: () {
                      Map<String, dynamic> formData = {};
                      formData['username'] = usernameController.text;
                      formData['password'] = passwordController.text;
                      pushRegister(context);
                  },
                  child: const Text('Belum punya akun ?')),
            ],
          ),
        ],
      ),
    ),
  ),
);
}
void pushRegister(BuildContext context) {
  Navigator.push(context,MaterialPageRoute(builder: (_) => const RegisterView(),),);
}
}
```

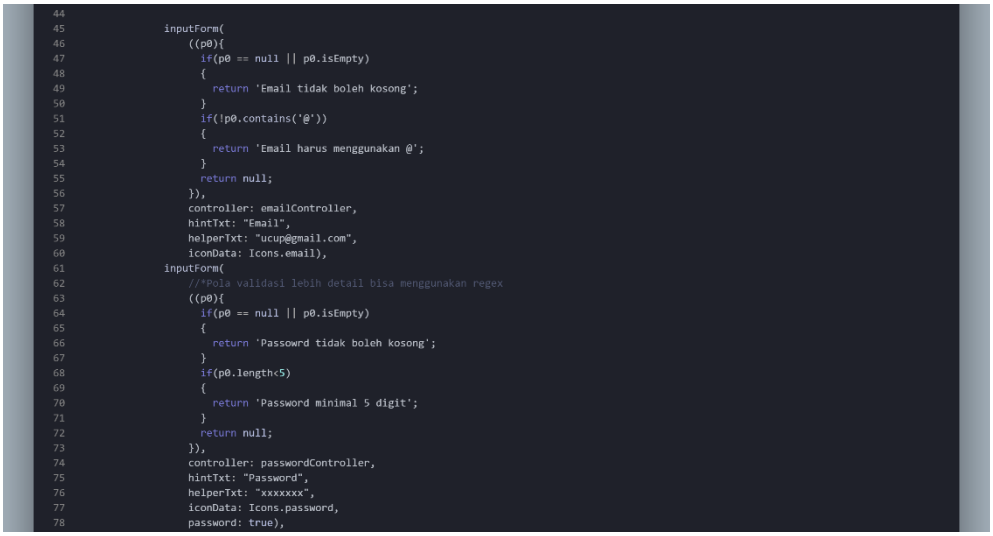
Gambar 15 login.dart (bagian 3)

9. Selanjutnya, kita pindah kehalaman RegisterView, teman-teman dapat menyalin code pada gambar 16 – gambar 18 (Gunakan shortcut STF) :



```dart
import 'package:flutter/material.dart';
import 'package:flutter_modul2/View/login.dart';
import 'package:flutter_modul2/component/form_component.dart';

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

  @override
  State<RegisterView> createState() => _RegisterViewState();
}

class _RegisterViewState extends State<RegisterView> {
  //*untuk validasi harus menggunakan GlobalKey
    final _formKey = GlobalKey<FormState>();
    TextEditingController usernameController = TextEditingController();
    TextEditingController emailController = TextEditingController();
    TextEditingController passwordController = TextEditingController();
    TextEditingController notelpController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Form(
          key: _formKey,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              inputForm(
                (p0)  {
                  if(p0 == null || p0.isEmpty)
                  {
                    return 'Username Tidak Boleh Kosong';
                  }
                  if(p0.toLowerCase() == 'anjing'){
                    return 'Tidak Boleh Menggunakan kata kasar';
                  }
                  return null;
                },
                  controller: usernameController,
                  hintTxt: "Username",
                  helperTxt: "Ucup Surucup",
                  iconData: Icons.person),
```

Gambar 16. register.dart (bagian 1)



```dart
              inputForm(
                  ((p0){
                    if(p0 == null || p0.isEmpty)
                    {
                      return 'Email tidak boleh kosong';
                    }
                    if(!p0.contains('@'))
                    {
                      return 'Email harus menggunakan @';
                    }
                    return null;
                  }),
                  controller: emailController,
                  hintTxt: "Email",
                  helperTxt: "ucup@gmail.com",
                  iconData: Icons.email),
              inputForm(
                  //*Pola validasi lebih detail bisa menggunakan regex
                  ((p0){
                    if(p0 == null || p0.isEmpty)
                    {
                      return 'Passowrd tidak boleh kosong';
                    }
                    if(p0.length<5)
                    {
                      return 'Password minimal 5 digit';
                    }
                    return null;
                  }),
                  controller: passwordController,
                  hintTxt: "Password",
                  helperTxt: "xxxxxxx",
                  iconData: Icons.password,
                  password: true),
```

Gambar 17. register.dart (bagian 2)

```
79              inputForm(
80                  ((p0){
81                    //* untuk menglihat contoh penggunaan regex, uncomment baris dibawah yang dicomment
82                    // final RegExp regex = RegExp(r'^\0?[1-9]\d{1,14}$');
83                    if(p0 == null || p0.isEmpty)
84                    {
85                      return 'Nomor Telepon tidak boleh kosong';
86                    }
87                    // if(!regex.hasMatch(p0))
88                    // {
89                      // return 'Nomor Telepon tidak valid';
90                    // }
91                    return null;
92                  }),
93                  controller: notelpController,
94                  hintTxt: "No Telp",
95                  helperTxt: "082123456789",
96                  iconData: Icons.phone_android),
97              ElevatedButton(
98                  onPressed: () {
99                    if (_formKey.currentState!.validate()) {
100                     // ScaffoldMessenger.of(context).showSnackBar(
101                     // const SnackBar(content: Text('Processing Data')));
102                       Map<String,dynamic> formData = {};
103                       formData['username']  = usernameController.text;
104                       formData['password']  = passwordController.text;
105                       //* Navigator.push(context, MaterialPageRoute(builder: (BuildContext buildContext) =>  LoginView(data: formData ,)) );
106                       Navigator.push(context, MaterialPageRoute(builder: (_) =>  LoginView(data: formData ,)) );
107                     }
108                 },
109                 child: const Text('Register'))
110           ],
111         ),
112       ),
113     ),
114   );
115 }
116 }
117
```

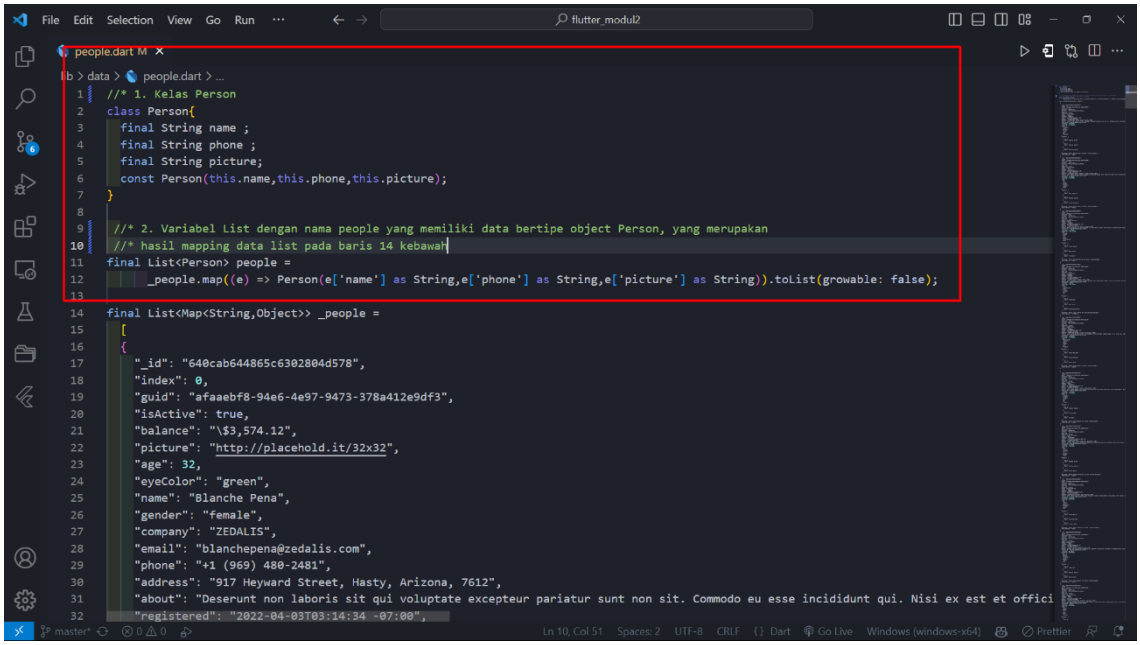
Gambar 18. register.dart (bagian 3)

10. Selanjutnya kita pindah kehalaman home (halaman setelah user berhasil login) , Anda dapat menyalin code berikut (gambar 19) :

```dart
import 'package:flutter/material.dart';
import 'package:flutter_modul2/View/view_list.dart';

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

  @override
  State<HomeView> createState() => _HomeViewState();
}

class _HomeViewState extends State<HomeView> {

  //* selectedIndex berkaitan dengan index halaman pada bottomNavigasi
  int _selectedIndex = 0;
  //* fungsi yang nantinya akan dijalankan setiap menekan menu pada navbar
  void _onItemTapped(int index) {
    //* setState berkaitan dengan fungsi untuk menampilkan perubahan kondisi & dalam banyak kasus akan menggunakan ini
    setState(() {
      _selectedIndex = index;
    });
  }

  //* Menampung List Widget yang akan ditampilkan sesuai index yang dipilih.
  static const List<Widget> _widgetOptions = <Widget>[

    //*index 0
    Center(
      child: Image(image: NetworkImage('https://picsum.photos/200/300'))
      ),
    //* index 1
    ListNamaView(), //* Jika Error di comment dulu aja
    //* index 2
    Center(
      child: Text(
        'Index 3: Profile',
      ),
    ),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      //* setting navigasi bar
      bottomNavigationBar: BottomNavigationBar(
        items: const [
        BottomNavigationBarItem(icon: Icon(Icons.home,),label: 'Home'),
        BottomNavigationBarItem(icon: Icon(Icons.list,),label: 'List'),
        BottomNavigationBarItem(icon: Icon(Icons.person,),label: 'Profile'),
        ],
        currentIndex: _selectedIndex, //* parameter : yang menampung index dari menu bottomNav
        onTap: _onItemTapped,  //* menjalankan fungsi _onItemTapped, yang dimana fungsi ini akan mengubah nilai index dan melakukan setstate sesuai index
      ),
      //* bagian body dari home berdasarkan List _widgetOption berdasarkan indeks selectedIndex
      body: _widgetOptions.elementAt(_selectedIndex), //* Mengubah tampilan widget sesuai nilai selectedIndex
    );
  }
}
```

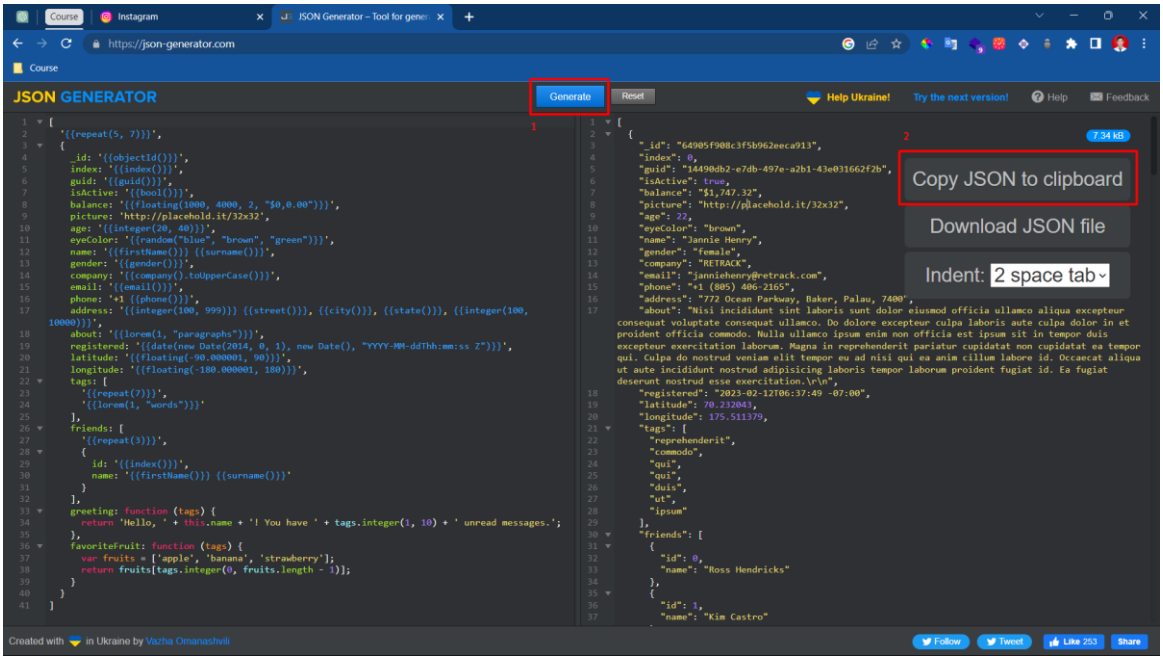
Gambar 19. home.dart

11. Sebelum kita masuk, ke halaman yang menampilkan list nama, kita perlu membuat modelnya terlebih dahulu, teman-teman fokus buat bagian didalam kotak merah pada gambar 20 terlebih dahulu :
    a. Kelas Person, sebuah blueprint untuk menyimpan data yang awalnya bertipe Map<String,Object> menjadi Object baru. (Representasi objek dari data di JSON)
    b. List<Person> people, merupakan sebuah list dari object Person yang dikonversi dari variable List<Map<String,Object>> _people.



Gambar 20 people.dart

12. Selanjutnya untuk membuat variable _people, teman-teman dapat mengambil datanya dari https://json-generator.com/, copy dan definisikan dalam variable _people . sehingga :
final List<Map<String,Object>> _people = *hasil_copy_paste*



Gambar 21. JSON GENERATOR

13. Selanjutnya kita membuat layout view_list (menampilkan list nama dengan ListTile dan Card) , teman-teman dapat menyalin code pada gambar 22 - gambar 25:

```dart
import 'package:flutter/material.dart';
import 'package:flutter_modul2/data/people.dart';

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Daftar Nama"),
      ),
      body: LayoutBuilder(builder: (context, constraints) {
        if(constraints.maxWidth > 600){
          //* Landscape
          return const WideLayout();
        }else{
          //* Portrait
          return const NarrowLayout();
        }
      }),
    );
  }
}

class NarrowLayout extends StatelessWidget {
  const NarrowLayout({super.key});
  @override
  Widget build(BuildContext context) {
    return PeopleList(
        onPersonTap: (person) => Navigator.of(context).push(MaterialPageRoute(
              builder: (context) => Scaffold(
                appBar: AppBar(),
                body: PersonDetail(person),
              ),
            ),),);
  }
}


```

Gambar 22. view_list.dart bagian 1

```dart
class WideLayout extends StatefulWidget {
  const WideLayout({super.key});

  @override
  State<WideLayout> createState() => _WideLayoutState();
}


class _WideLayoutState extends State<WideLayout> {
  Person? _person;
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        SizedBox(
          width: 300,
          child: Padding(
            padding: const EdgeInsets.all(12.0),
            child: PeopleList(
              onPersonTap: (person) => setState(() => _person = person),
            ),
          ),
        ),
        Expanded(
          flex: 3,
          child: _person == null ? const Placeholder() : PersonDetail(_person!),
        ),
      ],
    );
  }
}

```

Gambar 23 view_list.dart bagian 2

```dart

class PeopleList extends StatelessWidget {
  final void Function(Person) onPersonTap;
  const PeopleList({super.key, required this.onPersonTap});

  @override
  Widget build(BuildContext context) {
    return ListView(children: [
      for (var person in people)
        ListTile(

            leading: Image.network(person.picture)  ,
            // leading: const HtmlElementView(viewType: "<img>")  ,
            title: Text(person.name),
            onTap: () => onPersonTap(person)),
    ]);
  }
}

class PersonDetail extends StatelessWidget {
  final Person person;
  const PersonDetail(this.person, {super.key}); // masalah penyusunan

  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(
      // ignore: avoid_types_as_parameter_names
      builder: (buildContext, boxConstraints) {
          return Center(
            child: boxConstraints.maxHeight > 200 ? Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                MouseRegion(
                  // ignore: avoid_print
                  onHover: (_) => {print("Hello World")},
                  child: Text(person.name),
                ),
```

Gambar 24 view list.dart bagian 3

```
                ),
                Text(person.phone),
                ElevatedButton(
                  onPressed: () {},
                  child: const Text("Contact Me"),
                ),
              ],
            ) :
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: [
                MouseRegion(
                  // ignore: avoid_print
                  onHover: (_) => {print("Hello World")},
                  child: Text(person.name),
                ),
                Text(person.phone),
                ElevatedButton(
                  onPressed: () {},
                  child: const Text("Contact Me"),
                ),
              ],
          ),);
        }
    );
  }
}

```

Gambar 25 view_list.dart bagian 4

Selamat, Guided Anda telah selesai !!!!

PEMBAHASAN GUIDED

1. Pada project ini, kita memiliki file :
   a. main.dart

      Pada Bagian ini merupakan Bagian yang pertama dieksekusi Ketika aplikasi dijalankan, karena terdapat fungsi main yang menjalankan kelas/object MainApp();

```
Run | Debug | Profile
void main() {
  runApp(const MainApp());
}
```

Gambar 26 Fungsi runApp

      Kemudian pada bagian mainApp menjadikan LoginView() menjadi halaman pertama ditampilkan saat di run karena didefinisikan pada attribute home.

```
@override
Widget build(BuildContext context) {
  return   const MaterialApp(
    home:  LoginView(),
  ); // MaterialApp
}
```

Gambar 27 value dari home : LoginView()

   b. form_component.dart

      Pada file ini, kita memiliki sebuah fungsi inputForm yang merupakan fungsi yang membungkus TextFormField , menjadi widget bentukan yang nantinya akan kita gunakan pada halaman login dan register. Terdapat beberapa parameter :

      1) validasi : callback function
      2) controller
      3) hintText , helperText, dan iconData
      4) password(berkaitan dengan menyembunyikan password menjadi ****), secara default dinonaktifkan, attribute aslinya Bernama obscure text.

```
Padding inputForm(Function(String?) validasi,  {required TextEditingController controller,
required String hintTxt,required String helperTxt,required IconData iconData, bool password = false}) {
    return Padding(
```

Gambar 28 Parameter pada fungsi inputForm

c. login.dart dan register.dart

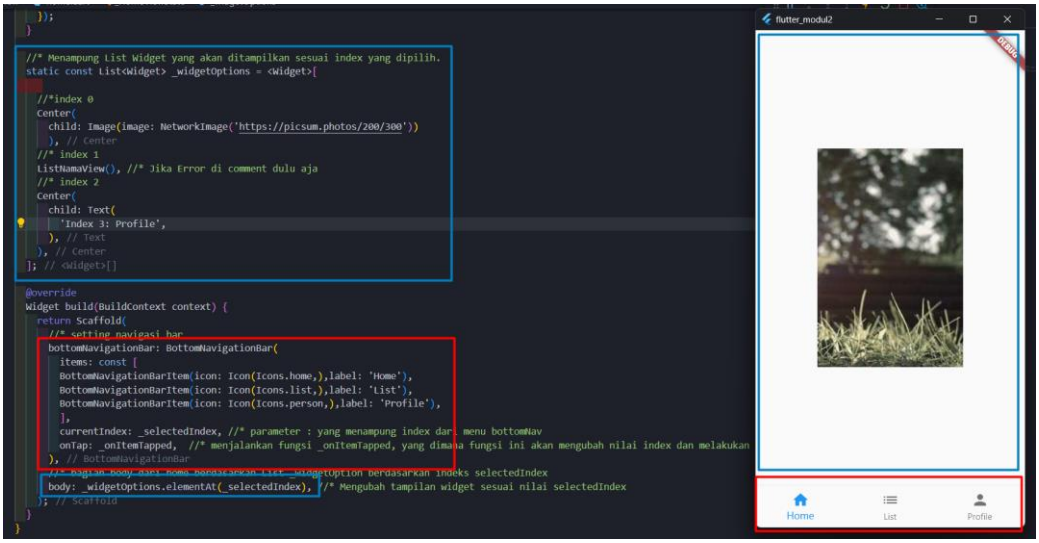
Poin yang perlu dipahami adalah :

1) Kita menggunakan Stateful Widget, karena pada kasus login dan register terdapat perubahan state / kondisi
2) Kita menggunakan "widget.data" untuk melakukan parsing data yang telah dibuat pada halaman register ke halaman login. (Mengsimulasikan /menyederhanakan proses register dan login)
3) Setiap inputan kita memerlukan controller untuk mengontrol value dari inputan dan kita dapat mengakses value inputan kita dengan namaController.text.
4) Kita menggunakan Form agar proses validasi dapat dilakukan, dan untuk menggunakan Form diperlukan key.
5) Kita menggunakan Navigator.pop, Navigator.push untuk navigasi antar halaman

d. home.dart

Poin yang perlu dipahami adalah :

1) Kita menggunakan stateful widget karena terdapat perubahan state pada UI.
2) Halaman Home berfungsi sebagai template yang menampung halaman konten dan bottom navigasi . Ilustrasi :
3) Index 0, index 2 nantinya dapat digantikan dengan sebuah halaman (seperti pada index 1)



Gambar 29 Penjelasan code navbar

e. view_list.dart

Poin yang perlu diperhatikan :

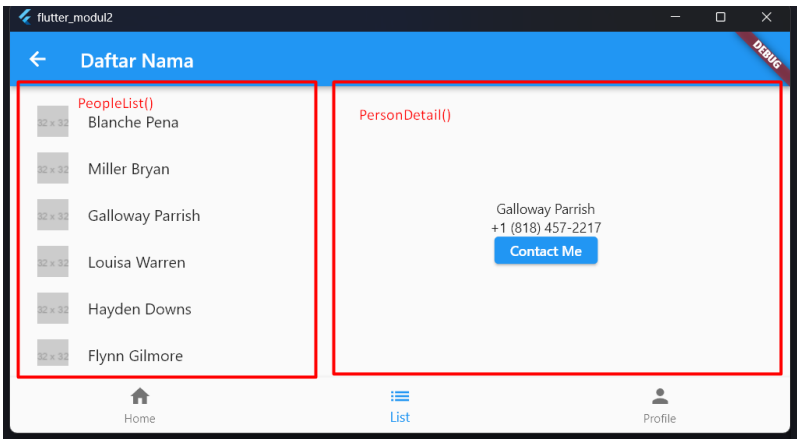
1) View_list.dart berfungsi untuk menampilkan list nama pada tab List. Terdapat 2 tampilan yaitu disaat kondisi perangkat landscape dan portrait. Hal ini dilakukan menggunakan pengecekan menggunakan LayoutBuilder.

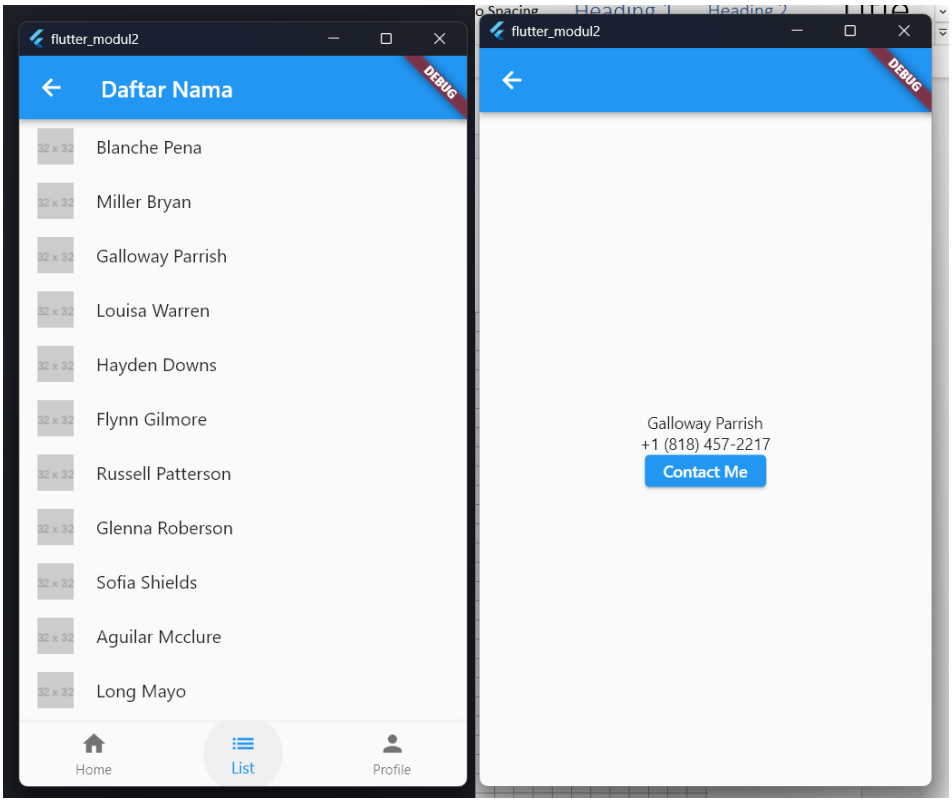
```
body: LayoutBuilder(builder: (context, constraints) {
  if(constraints.maxWidth > 600){
    //* Landscape
    return const WideLayout();
  }else{
    //* Portrait
    return const NarrowLayout();
  }
}), // LayoutBuilder
```

Gambar 30 code untuk membuat tampilan portrait dan landscape

2) Pada file view_list.dart, terdapat beberapa kelas :
   a) ListNamaView(), Bagian yang menghubungkan Home dan tampilan List. Pada Bagian inilah terdapat pengecekan kondisi layout.
   b) NarrowLayout(), layout disaat kondisi perangkat portrait.
   c) WideLayout(), layout disaat kondisi perangkat landscape

d) PeopleList(), layout yang menampilkan list nama menggunakan ListView() dan ListTile().
e) PersonDetail(), layout detail dari person, ditampilkan saat saat salah satu nama diselect/dipilih.





Gambar 31 Hasil App

f. People.dart

Poin yang perlu diperhatikan :

1) Pada file ini, kita membuat kelas Person,variabel List<Person> people, variabel List<Map<Person,Object>> _people. Variable _people, merupakan data mentah dari json, kemudian dikonversi ke list dari objek yang ditampung pada variable people.
2) JSON adalah format file berbasis teks yang biasa digunakan untuk pertukaran data antara server-klien melalui API. Materi ini nanti akan kalian pelajarin lebih lanjut di modul API, dan dimatakuliah pemrograman web. Pada project ini kita menggunakan data JSON sebagai data dummy. Kedepannya data tersebut tidak kita definisikan secara langsung seperti pada modul ini, namun didapatkan melalui API.

ATURAN PENGERJAAN GUIDED :

- **Guided dikerjakan selama waktu perkuliahan berlangsung.**
- **Penamaan projek guided harus sesuai dengan yang sudah dicontohkan.**
- **Guided dikumpulkan melalui github dengan penamaan setiap file & repository pada github adalah : *guidedlayout2_XXXX* (*contoh : guidedlayout2_9999*) dan pastikan sudah dalam mode PRIVATE.**
  - **XXXX → 4 DIGIT TERAKHIR NPM**
- **Setelah diupload melalui github, jangan lupa untuk mengumpulkan keseluruhan link file github melalui situs kuliah.**
- **Cara upload ke github, silahkan melihat pada modul "UPLOAD GITHUB".**