#07 | Layout dan Navigasi

| **Summary** | Mampu melakukan penyusunan layout dan navigasi serta routing |
| --- | --- |
| **URL** | 07-layout-navigasi |
| **Category** | Mobile Flutter |
| **Environment** | web |
| **Status** | Published |
| **Feedback Link** | https://github.com/jti-polinema/flutter-codelab/issues |
| **Author** | Habibie Ed Dien |
| **Author LDAP** | hbb-polinema |
| **Analytics Account** | G-H7507PFEJB |

[Layout dan Navigasi](#_ok7k5uux6)

[Tujuan Praktikum](#_21yzqg98x7h6)

[Sumber Daya yang Dibutuhkan](#_dusjnkkj26ob)

[Pengetahuan yang Anda harus Miliki](#_319h1e1sjzdz)

[Konsep Layout di Flutter](#_t0tqoph2ohlh)

[Memahami stateful widget life cycle](#_g7nutupwvu6v)

[Key life cycle states](#_89gt0c2q7m8e)

[Creation of the state](#_5w4vw82pp9b0)

[Initializing the state](#_oqkju35smcmf)

[Build](#_gkwhsy46pizn)

[Disposing of the state](#_hqa3lu1pn7e0)

[Mounted](#_7uyvqx1t47)

[Input widgets and forms](#_sl1nr883oe3v)

[Getting input through a controller](#_ndqn33mzkwzq)

[FormField and TextField](#_y4w796pa4qp3)

[Accessing the FormField widget's state](#_mpg6x3enher)

[Form](#_s4fuwn64dog5)

[Accessing the Form widget's state](#_gx5hx3cpzhfw)

[Using a key](#_auyyiycryknw)

[Using InheritedWidget](#_cxyxs6lfwvl6)

[Validating user input](#_qx6uhmge0oss)

[Custom input and FormField widgets](#_el89ede9berg)

[Creating custom inputs](#_lc4ug7jk1eko)

[Custom input widget example](#_od6s7wl3f9jr)

[Praktikum 1: Menerapkan Gesture Detector](#_vmpz7ap0erhd)

[Langkah 1: Buat Project Baru](#_spr93e7x3yo2)

[Langkah 2: Menambahkan GestureDetector](#_g8csgggwssw)

[Langkah 3: Buat Dokumentasi atau Laporan Praktikum](#_2q85gi9dhaiq)

[Langkah 4: Eksperimen Jenis Gesture Lain](#_p0t5r1yzb5qp)

[Praktikum 2: Menerapkan Input Widget dan Forms](#_86j27ymyw6sc)

[Langkah 1: Buat class FormContoh](#_z4xb79s85ro)

[Langkah 2: Buat class \_FormContohState](#_x89yqkiqmitd)

[Langkah 3: Tambah Widget FormContoh di main.dart](#_uvblkrqosg3y)

[Praktikum 3: Menerapkan Custom Input dan FormField Widget](#_gz7tbatl2uh8)

[Langkah 1: Buat class VerificationCodeInput](#_h9m2druzjfj8)

[Langkah 2: Buat class \_VerificationCodeInputState](#_2nzf86f8qvki)

[Langkah 3: Buat class VerificationCodeFormField](#_gxsz8i7mkdo3)

[Langkah 4: Buat class \_VerificationCodeFormFieldState](#_qp3pp1metiux)

[Langkah 5: Tambah variabel \_controller](#_ybah5qq959kt)

[Langkah 6: Tambah Widget VerificationCodeFormField](#_e4m3gfagvjh)

[Tugas Praktikum](#_dz35nqtk0wjb)

[Selamat!](#_s22qmcj4zu61)

[Apa selanjutnya?](#_9mqifffit2ew)

[Referensi](#_an95hoqxrlhg)

# Layout dan Navigasi



**Terakhir diperbarui:** 09 Oktober 2023

**Penulis:** Habibie Ed Dien

Pada codelab ini, Anda akan mempelajari konsep dan praktik untuk dasar-dasar framework Flutter dengan menerapkan layout dan navigasi.

## Tujuan Praktikum

Setelah menyelesaikan codelab ini Anda akan mampu untuk:

* menerapkan jenis-jenis layout
* menerapkan navigasi
* menerapkan routing

| **Catatan:** Materi ini diadaptasi dari Buku Flutter for Beginners - second edition dan Buku Ajar Pemrograman Mobile Flutter JTI Polinema serta sumber referensi lainnya. |
| --- |

## Sumber Daya yang Dibutuhkan

Berikut merupakan sumber daya yang diperlukan untuk menyelesaikan praktikum ini:

* PC atau Laptop dengan spesifikasi minimum RAM 4GB Processor Core i3
* Koneksi internet
* Chrome Browser
* Perangkat mobile (Android/iOS)

## Pengetahuan yang Anda harus Miliki

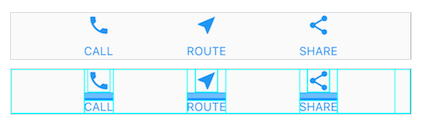
* Algoritma dan Pemrograman Dasar
* Pemrograman Berorientasi Objek
* Bahasa Pemrograman Dart

# Konsep Layout di Flutter

Duration: 10:00

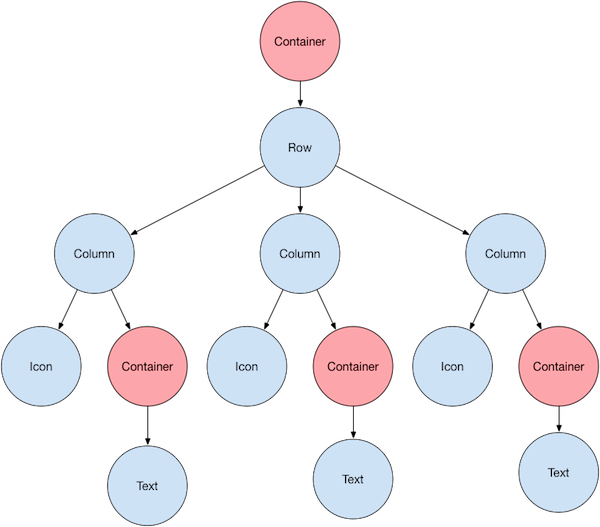
Inti dari mekanisme tata letak Flutter adalah widget. Di Flutter, hampir semuanya adalah widget—bahkan model tata letak pun merupakan widget. Gambar, ikon, dan teks yang Anda lihat di aplikasi Flutter semuanya merupakan widget. Namun hal-hal yang tidak Anda lihat juga merupakan widget, seperti baris, kolom, dan kisi yang mengatur, membatasi, dan menyelaraskan widget yang terlihat.

Anda membuat tata letak dengan menyusun widget untuk membuat widget yang lebih kompleks. Misalnya, tangkapan layar pertama di bawah ini menunjukkan 3 ikon dengan label di bawah masing-masing ikon:



Tangkapan layar kedua menampilkan tata letak visual, memperlihatkan deretan 3 kolom yang setiap kolomnya berisi ikon dan label.

Berikut diagram pohon widget untuk UI ini:



Sebagian besar tampilannya akan terlihat seperti yang Anda harapkan, namun Anda mungkin bertanya-tanya tentang wadahnya (ditampilkan dalam warna merah muda). Kontainer adalah kelas widget yang memungkinkan Anda menyesuaikan widget turunannya. Gunakan Kontainer ketika Anda ingin menambahkan padding, margin, batas, atau warna latar belakang, untuk menyebutkan beberapa kemampuannya.

Dalam contoh ini, setiap widget Teks ditempatkan dalam Wadah untuk menambahkan margin. Seluruh Baris juga ditempatkan dalam Wadah untuk menambahkan bantalan di sekitar baris.

UI lainnya dalam contoh ini dikendalikan oleh properti. Atur warna Ikon menggunakan properti warnanya. Gunakan properti Text.style untuk mengatur font, warna, berat, dan sebagainya. Kolom dan baris memiliki properti yang memungkinkan Anda menentukan bagaimana turunannya disejajarkan secara vertikal atau horizontal, dan berapa banyak ruang yang harus ditempati oleh turunannya.

# Praktikum 1: Membangun Layout di Flutter

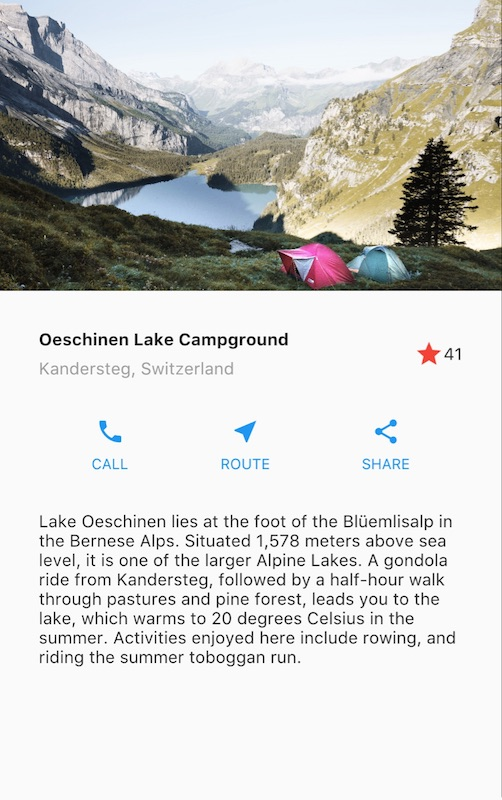
Duration: 15:00

| **Apa yang akan Anda pelajari**   * Cara kerja mekanisme tata letak Flutter. * Cara menata widget secara vertikal dan horizontal. * Cara membuat tata letak Flutter. |
| --- |

Selesaikan langkah-langkah praktikum berikut ini menggunakan editor Visual Studio Code (VS Code) atau Android Studio atau code editor lain kesukaan Anda.

| **Perhatian:** Diasumsikan Anda telah berhasil melakukan setup environment Flutter SDK, VS Code, Flutter Plugin, dan Android SDK pada pertemuan pertama. |
| --- |

**Tampilan akhir yang akan Anda buat**



## Langkah 1: Buat Project Baru

Buatlah sebuah project flutter baru dengan nama **layout\_flutter**. Atau sesuaikan style laporan praktikum yang Anda buat.

## Langkah 2: Buka file lib/main.dart

Buka file main.dart lalu ganti dengan kode berikut. Isi nama dan NIM Anda di text title.

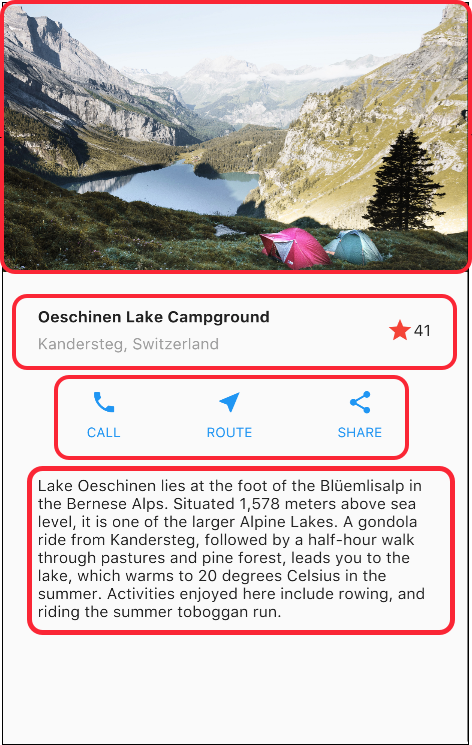
| import 'package:flutter/material.dart';  void main() => runApp(const MyApp());  class MyApp extends StatelessWidget {  const MyApp({super.key});  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Flutter layout: Nama dan NIM Anda',  home: Scaffold(  appBar: AppBar(  title: const Text('Flutter layout demo'),  ),  body: const Center(  child: Text('Hello World'),  ),  ),  );  }  } |
| --- |

## Langkah 3: Identifikasi layout diagram

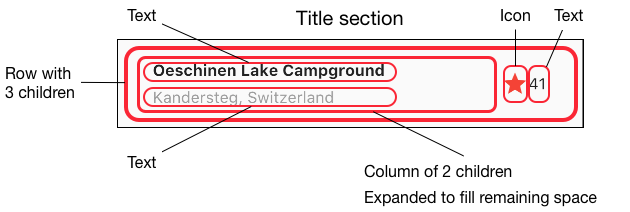
Langkah pertama adalah memecah tata letak menjadi elemen dasarnya:

* Identifikasi baris dan kolom.
* Apakah tata letaknya menyertakan kisi-kisi (grid)?
* Apakah ada elemen yang tumpang tindih?
* Apakah UI memerlukan tab?
* Perhatikan area yang memerlukan alignment, padding, atau borders.

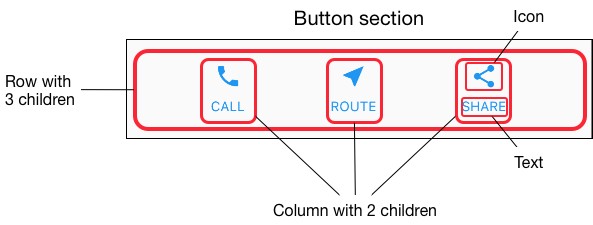
Pertama, identifikasi elemen yang lebih besar. Dalam contoh ini, empat elemen disusun menjadi sebuah kolom: sebuah gambar, dua baris, dan satu blok teks.



Selanjutnya, buat diagram setiap baris. Baris pertama, disebut bagian Judul, memiliki 3 anak: kolom teks, ikon bintang, dan angka. Anak pertamanya, kolom, berisi 2 baris teks. Kolom pertama itu memakan banyak ruang, sehingga harus dibungkus dengan widget yang Diperluas.



Baris kedua, disebut bagian Tombol, juga memiliki 3 anak: setiap anak merupakan kolom yang berisi ikon dan teks.



Setelah tata letak telah dibuat diagramnya, cara termudah adalah dengan menerapkan pendekatan bottom-up. Untuk meminimalkan kebingungan visual dari kode tata letak yang banyak bertumpuk, tempatkan beberapa implementasi dalam variabel dan fungsi.

## Langkah 4: Implementasi title row

Pertama, Anda akan membuat kolom bagian kiri pada judul. Tambahkan kode berikut di bagian atas metode build() di dalam kelas MyApp:

| Widget titleSection = Container(  padding: const EdgeInsets.all(...),  child: Row(  children: [  Expanded(  **/\* soal 1\*/**  child: Column(  crossAxisAlignment: …,  children: [  **/\* soal 2\*/**  Container(  padding: const EdgeInsets.only(bottom: …),  child: const Text(  'Wisata Gunung di Batu',  style: TextStyle(  fontWeight: FontWeight.bold,  ),  ),  ),  Text(  'Batu, Malang, Indonesia',  style: TextStyle(...),  ),  ],  ),  ),  **/\* soal 3\*/**  Icon(  …,  color: …,  ),  const Text(...),  ],  ),  ); |
| --- |

**/\* soal 1 \*/** Letakkan widget Column di dalam widget Expanded agar menyesuaikan ruang yang tersisa di dalam widget Row. Tambahkan properti crossAxisAlignment ke CrossAxisAlignment.start sehingga posisi kolom berada di awal baris.

**/\* soal 2 \*/** Letakkan baris pertama teks di dalam Container sehingga memungkinkan Anda untuk menambahkan padding = 8. Teks ‘Batu, Malang, Indonesia’ di dalam Column, set warna menjadi abu-abu.

**/\* soal 3 \*/** Dua item terakhir di baris judul adalah ikon bintang, set dengan warna merah, dan teks “41”. Seluruh baris ada di dalam Container dan beri padding di sepanjang setiap tepinya sebesar 32 piksel. Kemudian ganti isi body text ‘Hello World’ dengan variabel titleSection seperti berikut:



# Praktikum 2: Implementasi button row

Duration: 10:00

Selesaikan langkah-langkah praktikum berikut ini dengan melanjutkan dari praktikum sebelumnya.

## Langkah 1: Buat method Column \_buildButtonColumn

Bagian tombol berisi 3 kolom yang menggunakan tata letak yang sama—sebuah ikon di atas baris teks. Kolom pada baris ini diberi jarak yang sama, dan teks serta ikon diberi warna primer.

Karena kode untuk membangun setiap kolom hampir sama, buatlah metode pembantu pribadi bernama buildButtonColumn(), yang mempunyai parameter warna, Icon dan Text, sehingga dapat mengembalikan kolom dengan widgetnya sesuai dengan warna tertentu.

**lib/main.dart (\_buildButtonColumn)**

| class MyApp extends StatelessWidget {  const MyApp({super.key});  @override  Widget build(BuildContext context) {  // ···  }  Column \_buildButtonColumn(Color color, IconData icon, String label) {  return Column(  mainAxisSize: MainAxisSize.min,  mainAxisAlignment: MainAxisAlignment.center,  children: [  Icon(icon, color: color),  Container(  margin: const EdgeInsets.only(top: 8),  child: Text(  label,  style: TextStyle(  fontSize: 12,  fontWeight: FontWeight.w400,  color: color,  ),  ),  ),  ],  );  }  } |
| --- |

## Langkah 2: Buat widget buttonSection

Buat Fungsi untuk menambahkan ikon langsung ke kolom. Teks berada di dalam Container dengan margin hanya di bagian atas, yang memisahkan teks dari ikon.

Bangun baris yang berisi kolom-kolom ini dengan memanggil fungsi dan set warna, Icon, dan teks khusus melalui parameter ke kolom tersebut. Sejajarkan kolom di sepanjang sumbu utama menggunakan MainAxisAlignment.spaceEvenly untuk mengatur ruang kosong secara merata sebelum, di antara, dan setelah setiap kolom. Tambahkan kode berikut tepat di bawah deklarasi titleSection di dalam metode build():

**lib/main.dart (buttonSection)**

| Color color = Theme.of(context).primaryColor;  Widget buttonSection = Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  \_buildButtonColumn(color, Icons.call, 'CALL'),  \_buildButtonColumn(color, Icons.near\_me, 'ROUTE'),  \_buildButtonColumn(color, Icons.share, 'SHARE'),  ],  ); |
| --- |

## Langkah 3: Tambah button section ke body

Tambahkan variabel buttonSection ke dalam body seperti berikut:



# Praktikum 3: Implementasi text section

Duration: 10:00

Selesaikan langkah-langkah praktikum berikut ini dengan melanjutkan dari praktikum sebelumnya.

## Langkah 1: Buat widget textSection

Tentukan bagian teks sebagai variabel. Masukkan teks ke dalam Container dan tambahkan padding di sepanjang setiap tepinya. Tambahkan kode berikut tepat di bawah deklarasi buttonSection:

| Widget textSection = Container(  padding: const EdgeInsets.all(32),  child: const Text(  'Carilah teks di internet yang sesuai '  'dengan foto atau tempat wisata yang ingin '  'Anda tampilkan. '  'Tambahkan nama dan NIM Anda sebagai '  'identitas hasil pekerjaan Anda. '  'Selamat mengerjakan 🙂.',  softWrap: true,  ),  ); |
| --- |

Dengan memberi nilai softWrap = true, baris teks akan memenuhi lebar kolom sebelum membungkusnya pada batas kata.

## Langkah 2: Tambahkan variabel text section ke body

Tambahkan widget variabel textSection ke dalam body seperti berikut:



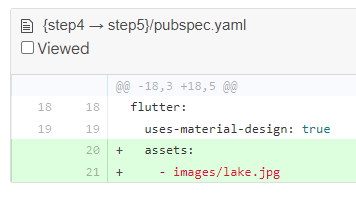
# Praktikum 4: Implementasi image section

Duration: 10:00

Selesaikan langkah-langkah praktikum berikut ini dengan melanjutkan dari praktikum sebelumnya.

## Langkah 1: Siapkan aset gambar

Anda dapat mencari gambar di internet yang ingin ditampilkan. Buatlah folder images di root project **layout\_flutter**. Masukkan file gambar tersebut ke folder images, lalu set nama file tersebut ke file pubspec.yaml seperti berikut:

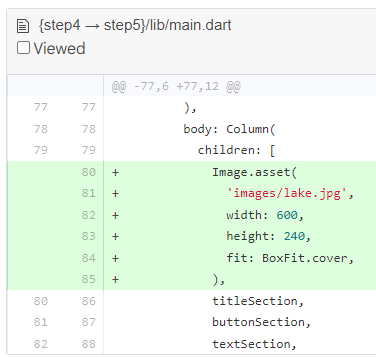


Contoh nama file gambar di atas adalah lake.jpg

| **Tips**   * Perhatikan bahwa pubspec.yaml sensitif terhadap huruf besar-kecil, jadi tulis assets: dan URL gambar seperti yang ditunjukkan di atas. * File pubspec juga sensitif terhadap spasi, jadi gunakan indentasi yang tepat. * Anda mungkin perlu memulai ulang program yang sedang berjalan (baik di simulator atau perangkat yang terhubung) agar perubahan pubspec dapat diterapkan. |
| --- |

## Langkah 2: Tambahkan gambar ke body

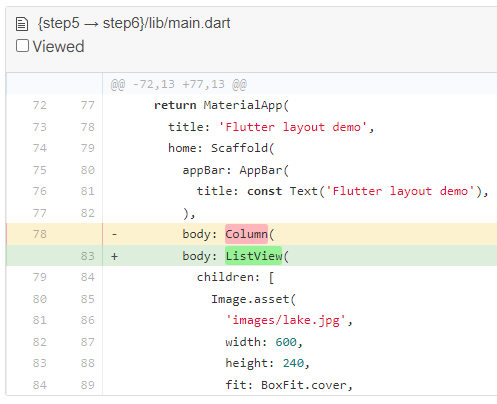
Tambahkan aset gambar ke dalam body seperti berikut:



BoxFit.cover memberi tahu kerangka kerja bahwa gambar harus sekecil mungkin tetapi menutupi seluruh kotak rendernya.

## Langkah 3: Terakhir, ubah menjadi ListView

Pada langkah terakhir ini, atur semua elemen dalam ListView, bukan Column, karena ListView mendukung scroll yang dinamis saat aplikasi dijalankan pada perangkat yang resolusinya lebih kecil.



# Tugas Praktikum 1

Duration: 120:00

1. Selesaikan Praktikum 1 sampai 4, lalu dokumentasikan dan push ke repository Anda berupa screenshot setiap hasil pekerjaan beserta penjelasannya di file README.md!
2. Silakan implementasikan di project baru "**basic\_layout\_flutter**" dengan mengakses sumber ini: <https://docs.flutter.dev/codelabs/layout-basics>
3. Kumpulkan link commit repository GitHub Anda ke spreadsheet yang telah disediakan!

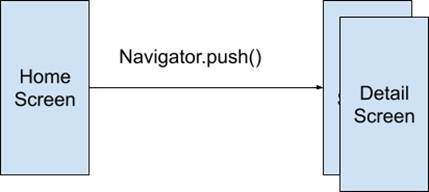
# Konsep Navigasi dan Rute

Duration: 5:00

Perpindahan halaman di Flutter, ditangani oleh Navigator dengan melibatkan konsep sebagai berikut:

* **Navigator**: sebuah widget yang mengatur tumpukan (struktur data *stack*) dari objek rute.
* **Route**: sebuah objek yang merepresentasikan tampilan, umumnya diimplementasikan oleh class seperti MaterialPageRoute.

Sebuah Route umumnya dimasukkan (*push*) atau diambil (*pop*) dari dan ke tumpukan Navigator. Ketika sebuah halaman dilakukan operasi push, maka halaman tersebut akan diletakkan di atas halaman yang memanggilnya. Ilustrasi tersebut dapat anda lihat pada gambar berikut. Dan jika pop dipanggil (tombol back ditekan) maka aplikasi akan menampilkan halaman sebelumnya. Selain itu Flutter juga mendukung adanya penamaan Route yang didefinisikan di awal.



# Praktikum 5: Membangun Navigasi di Flutter

Duration: 90:00

| **Apa yang akan Anda pelajari**   * Cara kerja mekanisme navigation dan route di Flutter. * Cara membuat navigation dan route di Flutter. |
| --- |

Selesaikan langkah-langkah praktikum berikut ini menggunakan editor Visual Studio Code (VS Code) atau Android Studio atau code editor lain kesukaan Anda. Materi ini dapat dimasukkan ke Laporan Praktikum folder **Week** atau **Pertemuan 08**.

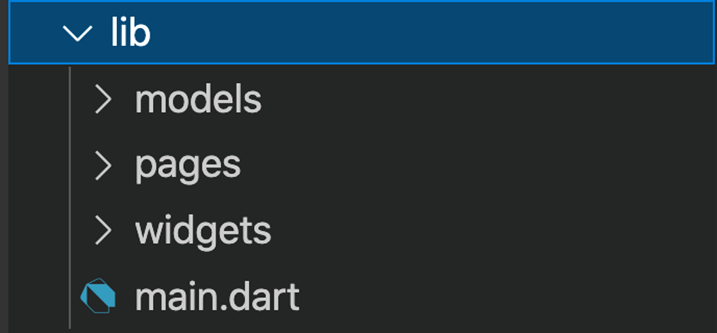
**Tampilan akhir yang akan Anda buat**

****

Pada praktikum 5 ini anda akan belajar mengenai pembangunan aplikasi bergerak multi halaman. Aplikasi yang dikembangkan berupa kasus daftar barang belanja. Pada aplikasi ini anda akan belajar untuk berpindah halaman dan mengirimkan data ke halaman lainnya. Gambaran mockup hasil akhir aplikasi dapat anda lihat pada gambar di atas (mockup dibuat sederhana, sehingga Anda mempunyai banyak ruang untuk berkreasi). Desain aplikasi menampilkan sebuah ListView widget yang datanya bersumber dari List. Ketika item ditekan, data akan dikirimkan ke halaman berikutnya.

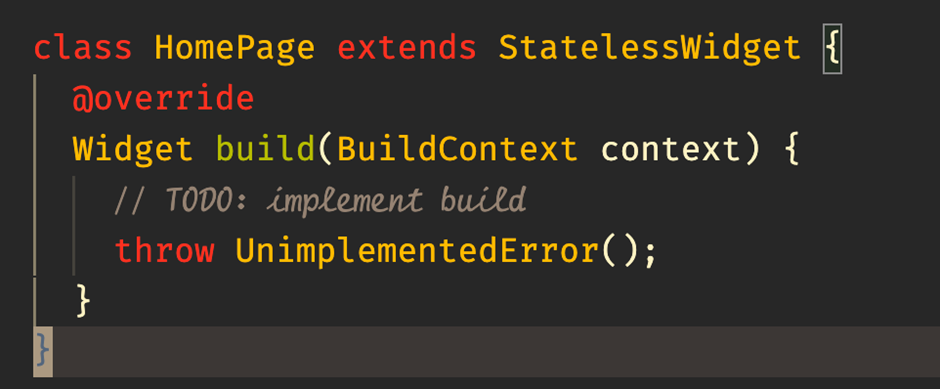
## Langkah 1: Siapkan project baru

Sebelum melanjutkan praktikum, buatlah sebuah project baru Flutter dengan nama **belanja** dan susunan folder seperti pada gambar berikut. Penyusunan ini dimaksudkan untuk mengorganisasi kode dan widget yang lebih mudah.



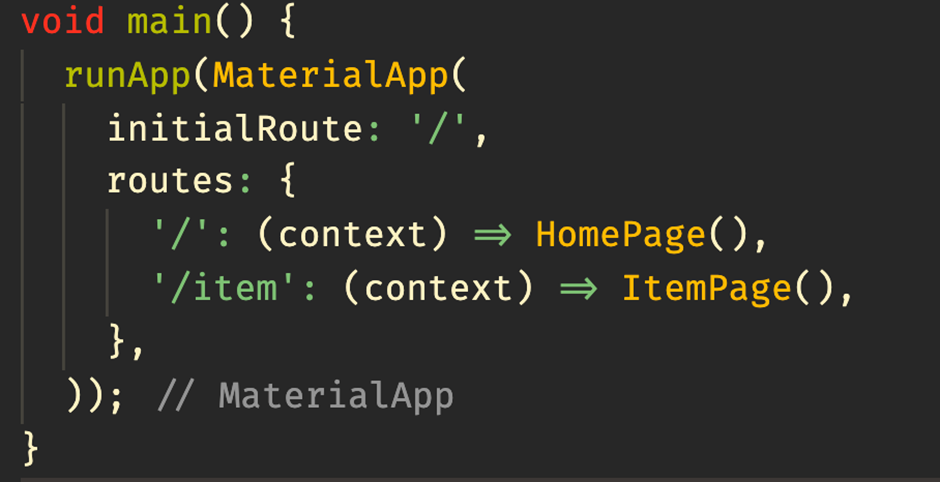
## Langkah 2: Mendefinisikan Route

Buatlah dua buah file dart dengan nama home\_page.dart dan item\_page.dart pada folder **pages**. Untuk masing-masing file, deklarasikan class HomePage pada file home\_page.dart dan ItemPage pada item\_page.dart. Turunkan class dari StatelessWidget. Gambaran potongan kode dapat anda lihat sebagai berikut.



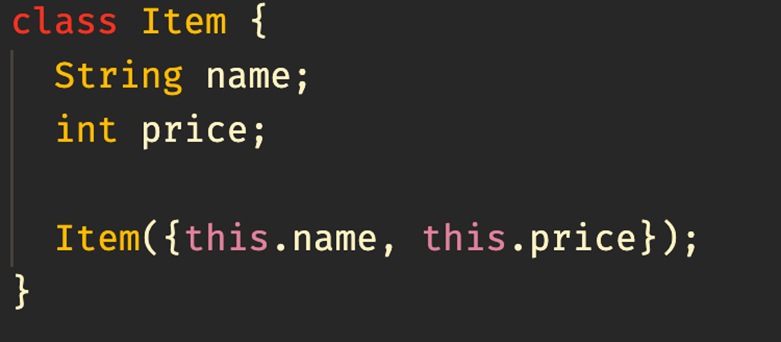
## Langkah 3: Lengkapi Kode di main.dart

Setelah kedua halaman telah dibuat dan didefinisikan, bukalah file main.dart. Pada langkah ini anda akan mendefinisikan **Route** untuk kedua halaman tersebut. Definisi penamaan **route** harus bersifat **unique**. Halaman **HomePage** didefinisikan sebagai **/**. Dan halaman **ItemPage** didefinisikan sebagai **/item**. Untuk mendefinisikan halaman awal, anda dapat menggunakan named argument initialRoute. Gambaran tahapan ini, dapat anda lihat pada potongan kode berikut.



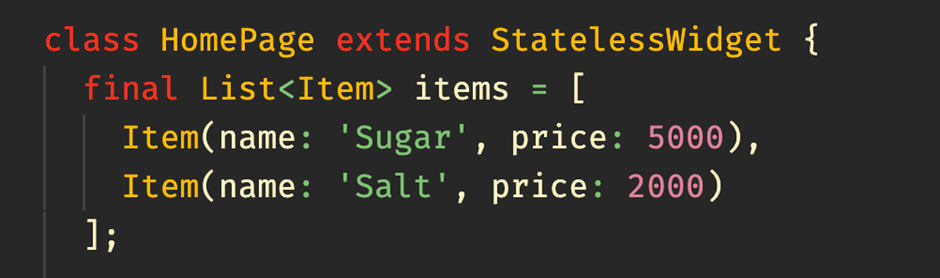
## Langkah 4: Membuat data model

Sebelum melakukan perpindahan halaman dari HomePage ke ItemPage, dibutuhkan proses pemodelan data. Pada desain mockup, dibutuhkan dua informasi yaitu nama dan harga. Untuk menangani hal ini, buatlah sebuah file dengan nama item.dart dan letakkan pada folder **models**. Pada file ini didefinisikan pemodelan data yang dibutuhkan. Ilustrasi kode yang dibutuhkan, dapat anda lihat pada potongan kode berikut.



## Langkah 5: Lengkapi kode di class HomePage

Pada halaman HomePage terdapat ListView widget. Sumber data ListView diambil dari model List dari object Item. Gambaran kode yang dibutuhkan untuk melakukan definisi model dapat anda lihat sebagai berikut.



## Langkah 6: Membuat ListView dan itemBuilder

Untuk menampilkan ListView pada praktikum ini digunakan itemBuilder. Data diambil dari definisi model yang telah dibuat sebelumnya. Untuk menunjukkan batas data satu dan berikutnya digunakan widget Card. Kode yang telah umum pada bagian ini tidak ditampilkan. Gambaran kode yang dibutuhkan dapat anda lihat sebagai berikut.



Jalankan aplikasi pada emulator atau pada device anda.

| **Perhatian:** Pastikan pada halaman awal telah berhasil menampilkan ListView. Jika ada kesalahan, segera perbaiki sebelum melanjutkan ke langkah berikutnya. |
| --- |

## Langkah 7: Menambahkan aksi pada ListView

Item pada ListView saat ini ketika ditekan masih belum memberikan aksi tertentu. Untuk menambahkan aksi pada ListView dapat digunakan widget InkWell atau GestureDetector. Perbedaan utamanya InkWell merupakan material widget yang memberikan efek ketika ditekan. Sedangkan GestureDetector bersifat umum dan bisa juga digunakan untuk gesture lain selain sentuhan. Pada praktikum ini akan digunakan widget InkWell.

Untuk menambahkan sentuhan, letakkan cursor pada widget pembuka Card. Kemudian gunakan shortcut quick fix dari VSCode (**Ctrl + .** pada Windows atau **Cmd + .** pada MacOS). Sorot menu wrap with widget… Ubah nilai widget menjadi InkWell serta tambahkan named argument onTap yang berisi fungsi untuk berpindah ke halaman ItemPage. Ilustrasi potongan kode dapat anda lihat pada potongan berikut.



Jalankan aplikasi kembali dan pastikan ListView dapat disentuh dan berpindah ke halaman berikutnya. Periksa kembali jika terdapat kesalahan.

# Tugas Praktikum 2

Duration: 120:00

1. Untuk melakukan pengiriman data ke halaman berikutnya, cukup menambahkan informasi arguments pada penggunaan Navigator. Perbarui kode pada bagian Navigator menjadi seperti berikut.

| Navigator.pushNamed(context, '/item', arguments: item); |
| --- |

1. Pembacaan nilai yang dikirimkan pada halaman sebelumnya dapat dilakukan menggunakan ModalRoute. Tambahkan kode berikut pada blok fungsi build dalam halaman ItemPage. Setelah nilai didapatkan, anda dapat menggunakannya seperti penggunaan variabel pada umumnya. (<https://docs.flutter.dev/cookbook/navigation/navigate-with-arguments>)

| final itemArgs = ModalRoute.of(context)!.settings.arguments as Item; |
| --- |

1. Pada hasil akhir dari aplikasi **belanja** yang telah anda selesaikan, tambahkan atribut foto produk, stok, dan rating. Ubahlah tampilan menjadi GridView seperti di aplikasi marketplace pada umumnya.
2. Silakan implementasikan Hero widget pada aplikasi **belanja** Anda dengan mempelajari dari sumber ini: <https://docs.flutter.dev/cookbook/navigation/hero-animations>
3. Sesuaikan dan modifikasi tampilan sehingga menjadi aplikasi yang menarik. Selain itu, pecah widget menjadi kode yang lebih kecil. Tambahkan **Nama** dan **NIM** di footer aplikasi **belanja** Anda.
4. Selesaikan Praktikum Navigasi dan Rute tersebut, lalu dokumentasikan dan push ke repository Anda berupa screenshot setiap hasil pekerjaan beserta penjelasannya di file README.md. Kumpulkan link commit repository GitHub Anda ke spreadsheet yang telah disediakan!

# Selamat!

Duration: 0:00

Selamat Anda telah menyelesaikan Codelab ini. Anda telah mempelajari Layout, Navigasi, dan rute di Flutter.

Pada codelab berikutnya, Anda akan mempelajari tentang manajemen plugin di Flutter.

## Apa selanjutnya?

Silakan cek beberapa sumber belajar lainnya…

* <https://docs.flutter.dev/cookbook/navigation/passing-data>
* [Flutter System Architecture](https://docs.google.com/presentation/d/1cw7A4HbvM_Abv320rVgPVGiUP2msVs7tfGbkgdrTy0I/edit#slide=id.p)
* <https://docs.flutter.dev/codelabs/layout-basics>
* <https://docs.flutter.dev/ui/interactivity>
* <https://gallery.flutter.dev/>

## Referensi

* <https://docs.flutter.dev/ui/layout>
* <https://docs.flutter.dev/ui/layout/tutorial>
* [Flutter for Beginners Second Edition](https://drive.google.com/file/d/1hJbctx-_O3B6kUeG70bJD5xfbnsWwBXY/view?usp=sharing)
* <https://www.youtube.com/flutterdev>
* <https://dart.dev/guides/language/language-tour>
* <https://github.com/PacktPublishing/Flutter-for-Beginners-Second-Edition>