

**LAPORAN PRAKTIKUM**  
**MATA KULIAH PEMROGRAMAN MOBILE**

Dosen Pengampu : Ade Ismail, S.Kom., M.TI.

**JOBSHEET 5: LAYOUT DAN NAVIGASI**



Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**  
**POLITEKNIK NEGERI MALANG**

**2025**

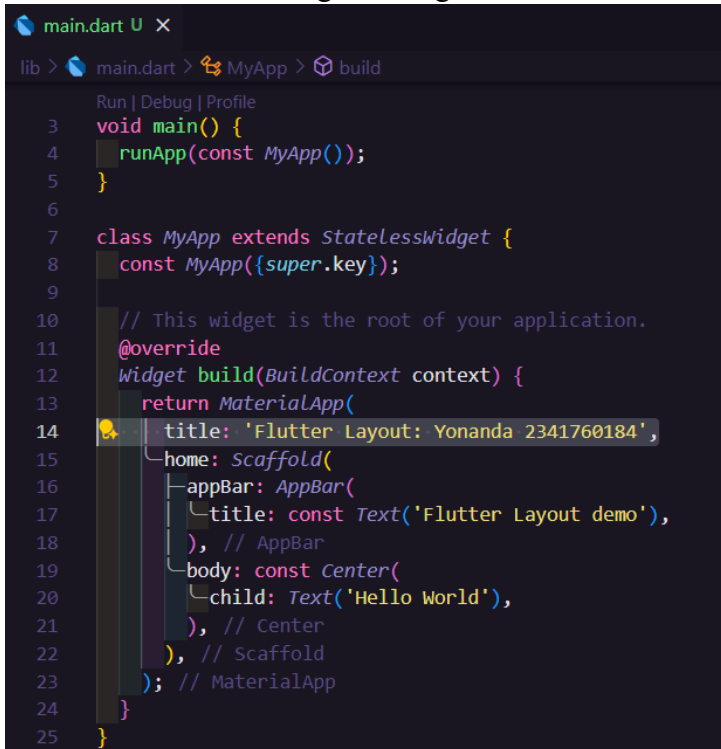
## PRAKTIKUM 1: MEMBANGUN LAYOUT DI FLUTTER

### 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.



```
main.dart U X
lib > main.dart > MyApp > build

Run | Debug | Profile
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Layout: Yonanda 2341760184',
15      home: Scaffold(
16        appBar: AppBar(
17          title: const Text('Flutter Layout demo'),
18        ), // AppBar
19        body: const Center(
20          child: Text('Hello World'),
21        ), // Center
22      ), // Scaffold
23    ); // MaterialApp
24  }
25 }
```

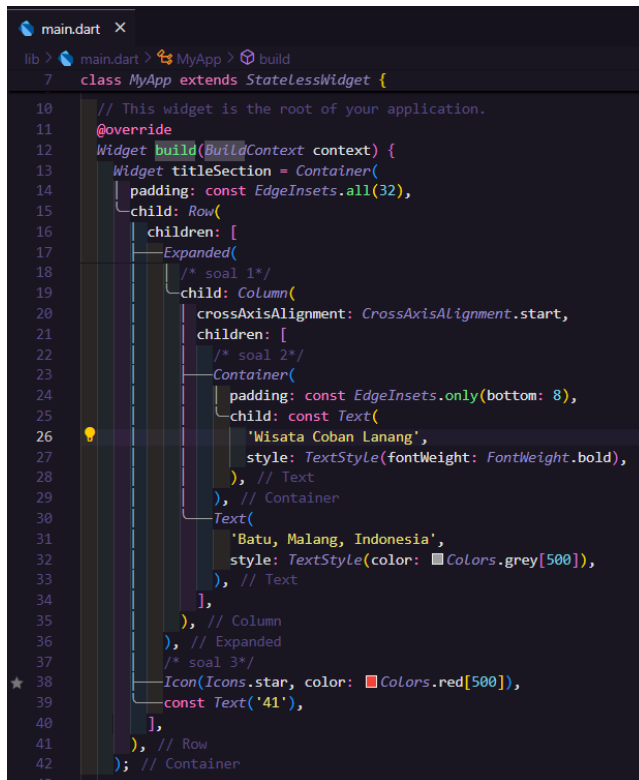
### Langkah 3: Identifikasi layout diagram

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.
- 

### 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:



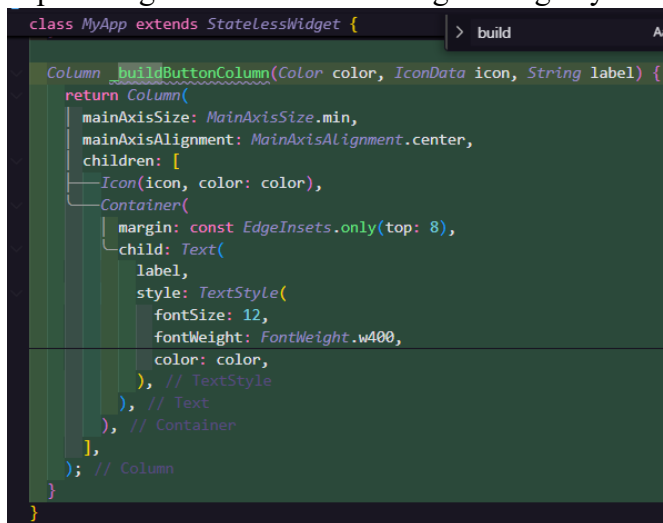
```
main.dart X
lib > main.dart > MyApp > build
7 class MyApp extends StatelessWidget {
10   // This widget is the root of your application.
11   @override
12   Widget build(BuildContext context) {
13     Widget titleSection = Container(
14       padding: const EdgeInsets.all(32),
15       child: Row(
16         children: [
17           Expanded(
18             /* soal 1*/
19             child: Column(
20               crossAxisAlignment: CrossAxisAlignment.start,
21               children: [
22                 /* soal 2*/
23                 Container(
24                   padding: const EdgeInsets.only(bottom: 8),
25                   child: const Text(
26                     'Wisata Coban Lanang',
27                     style: TextStyle(fontWeight: FontWeight.bold),
28                   ), // Text
29                 ), // Container
30                 Text(
31                   'Batu, Malang, Indonesia',
32                   style: TextStyle(color: Colors.grey[500]),
33                 ), // Text
34               ],
35             ), // Column
36           ), // Expanded
37           /* soal 3*/
38           Icon(Icons.star, color: Colors.red[500]),
39           const Text('41'),
40         ],
41       ), // Row
42     ); // Container
43   }
```

## PRAKTIKUM 2: IMPLEMENTASI BUTTON ROW

### 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.

A screenshot of an IDE showing the implementation of the `buildButtonColumn` method. The code is written in Dart and is part of a class `MyApp` that extends `StatelessWidget`. The method `buildButtonColumn` takes three parameters: `Color color`, `IconData icon`, and `String label`. It returns a `Column` widget with the following properties: `mainAxisSize: MainAxisSize.min`, `mainAxisAlignment: MainAxisAlignment.center`, and a list of children. The children list contains an `Icon` widget with the provided `icon` and `color`, followed by a `Container` widget. The `Container` has a `margin` of `const EdgeInsets.only(top: 8)` and a single child, a `Text` widget. The `Text` widget has the `label` as its text, a `TextStyle` with `fontSize: 12`, `fontWeight: FontWeight.w400`, and the same `color` as the icon. The code is color-coded and includes comments like `// Text` and `// Column`.

```
class MyApp extends StatelessWidget {  
  Column buildButtonColumn(Color color, IconData icon, String label) {  
    return Column(  
      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,  
            ), // TextStyle  
          ), // Text  
        ), // Container  
      ],  
    ); // Column  
  }  
}
```

### 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()`:

```
class MyApp extends StatelessWidget {
  Widget build(BuildContext context) {

    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:

```
return MaterialApp(
  title: 'Flutter Layout: Yonanda 2341760184',
  home: Scaffold(
    appBar: AppBar(title: const Text('Flutter Layout Demo')),
    body: Column(children: [titleSection, buttonSection]),
  ), // Scaffold
); // MaterialApp
}
```

## PRAKTIKUM 3: IMPLEMENTASI TEXT SECTION

### 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(  
    'Nama: Yonanda Mayla Rusdiaty\n'  
    'NIM: 2341760184\n\n'  
    'Coban Lanang adalah destinasi wisata alam yang terletak di Dusun Ngajung, Desa Pandanrejo, Kecamatan  
    softWrap: true,  
  ), // Text  
); // Container
```

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

### Langkah 2: Tambahkan variabel text section ke body

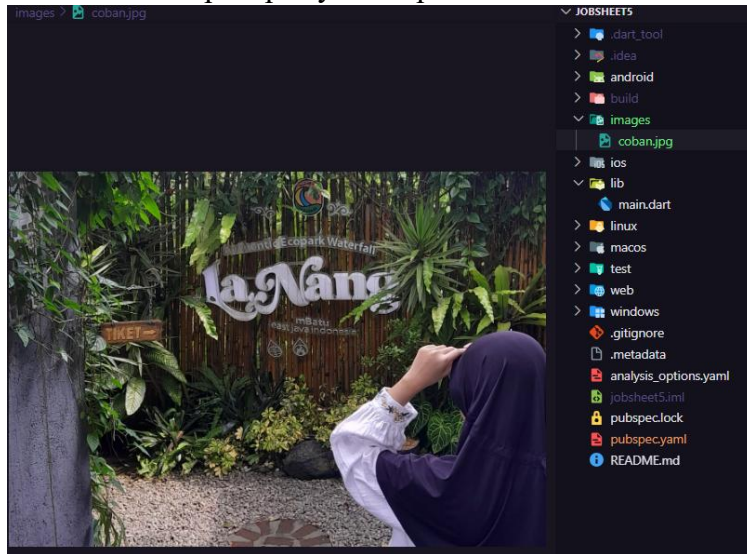
Tambahkan widget variabel textSection ke dalam body seperti berikut:

```
return MaterialApp(  
  title: 'Flutter Layout: Yonanda 2341760184',  
  home: Scaffold(  
    appBar: AppBar(title: const Text('Flutter Layout Demo')),  
    body: Column(children: [titleSection, buttonSection, textSection]),  
  ), // Scaffold  
); // MaterialApp  
}
```

## PRAKTIKUM 4: IMPLEMENTASI IMAGE SECTION

### 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:



```
flutter:  
  # The following line ensures that the package  
  # included with your application is the material Icons class.  
  uses-material-design: true  
  
  # To add assets to your application, add an assets  
  assets:  
    - images/coban.jpg
```

### Langkah 2: Tambahkan gambar ke body

Tambahkan aset gambar ke dalam body seperti berikut:

```

Widget imageSection = Image.asset(
  'images/coban.jpg',
  width: 600,
  height: 240,
  fit: BoxFit.cover,
);

return MaterialApp(
  title: 'Flutter Layout: Yonanda 2341760184',
  home: Scaffold(
    appBar: AppBar(title: const Text('Flutter Layout Demo')),
    body: Column(children: [titleSection, buttonSection, textSection]),
    // body: Column(children: [imageSection, titleSection, buttonSection, textSection]),
  ), // Scaffold
); // MaterialApp
}

```

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.

```

return MaterialApp(
  title: 'Flutter Layout: Yonanda 2341760184',
  home: Scaffold(
    appBar: AppBar(title: const Text('Flutter Layout Demo')),
    body: Column(
      body: ListView(
        children: [imageSection, titleSection, buttonSection, textSection],
      ), // ListView
    ), // Scaffold
  ); // MaterialApp
}

```



Berikut merupakan tampilan hasil implementasi:

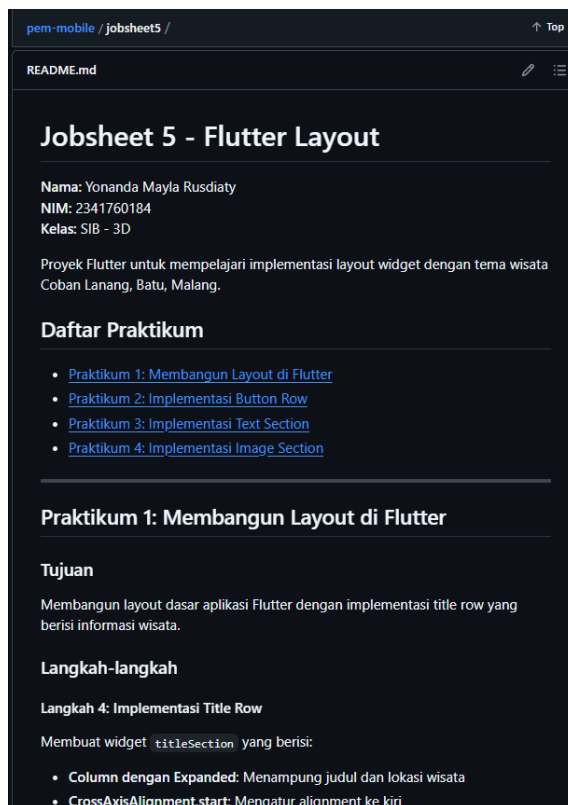


## Tugas Praktikum 1

1. Selesaikan Praktikum 1 sampai 4, lalu dokumentasikan dan push ke repository Anda berupa screenshot setiap hasil pekerjaan beserta penjelasannya di file README.md!

**Jawab:**

Sudah, seperti berikut:



2. Kumpulkan link commit repository GitHub Anda kepada dosen yang telah disepakati!

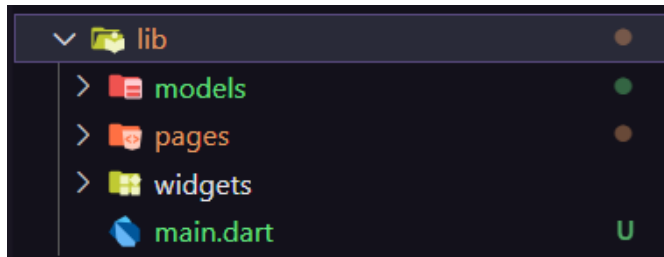
**Jawab:**

<https://github.com/yonandamayla/pem-mobile/>

## PRAKTIKUM 5: MEMBANGUN NAVIGASI DI FLUTTER

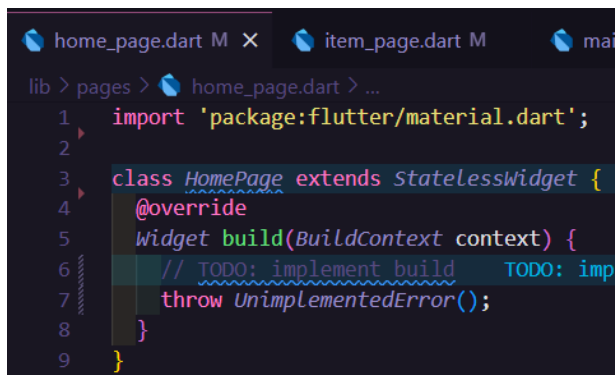
### Langkah 1: Siapkan project baru

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.



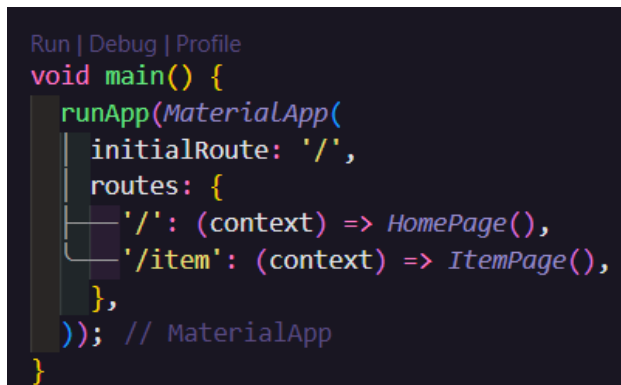
### Langkah 2: Mendefinisikan Route

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.



```
Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => HomePage(),
      '/item': (context) => ItemPage(),
    },
  )); // MaterialApp
}
```

### 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.

```
class Item {  
    String name;  
    int price;  
  
    Item({required this.name, required this.price});  
}
```

### 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.

```
class HomePage extends StatelessWidget {  
    final List<Item> items = [  
        Item(name: 'Sugar', price: 5000),  
        Item(name: 'Salt', price: 2000)  
    ];  
}
```

### 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.

```
home_page.dart M X item_page.dart main.dart
lib > pages > home_page.dart > ...
4 class HomePage extends StatelessWidget {
10   @override
11   Widget build(BuildContext context) {
12     return Scaffold(
13       appBar: AppBar(
14         title: Text('Shopping List'),
15       ), // AppBar
16       body: Container(
17         margin: EdgeInsets.all(8),
18         child: ListView.builder(
19           padding: EdgeInsets.all(8),
20           itemCount: items.length,
21           itemBuilder: (context, index) {
22             final item = items[index];
23             return Card(
24               child: Container(
25                 margin: EdgeInsets.all(8),
26                 child: Row(
27                   children: [
28                     Expanded(
29                       child: Text(item.name),
30                     ), // Expanded
31                     Expanded(
32                       child: Text(
33                         item.price.toString(),
34                         textAlign: TextAlign.end,
35                       ), // Text
36                     ), // Expanded
37                   ],
38                 ), // Row
39               ), // Container
40             ); // Card
41           }, // ListView.builder
42         ), // Container
43       ); // Scaffold
44     );
45   }
```



## 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.

```
return InkWell(  
  onTap: () {  
    Navigator.pushNamed(context, '/item');  
  },  
  child: Card(  
    title: Text('Item'),  
    child: Text('Item'),  
  ),  
);
```

## TUGAS PRAKTIKUM 2

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

```
final item = items[index];  
return ItemCard(  
  item: item,  
  onTap: () =>  
    Navigator.pushNamed(context, '/item', arguments: item),  
); // ItemCard
```

2. 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>)

```
@override  
Widget build(BuildContext context) {  
  final itemArgs = ModalRoute.of(context)!.settings.arguments as Item;
```

3. 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.
4. Silakan implementasikan Hero widget pada aplikasi **belanja** Anda dengan mempelajari dari sumber ini: <https://docs.flutter.dev/cookbook/navigation/hero-animations>
5. 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.



```
    }, // Expanded
    Footer(name: 'Yonanda Mayla Rusdiaty', nim: '23417660184')
  ],
```

6. Selesaikan Praktikum 5: Navigasi dan Rute tersebut. Cobalah modifikasi menggunakan plugin [go\\_router](#), lalu dokumentasikan dan push ke repository Anda berupa screenshot setiap hasil pekerjaan beserta penjelasannya di file README.md. Kumpulkan link commit repository GitHub Anda kepada dosen yang telah disepakati!