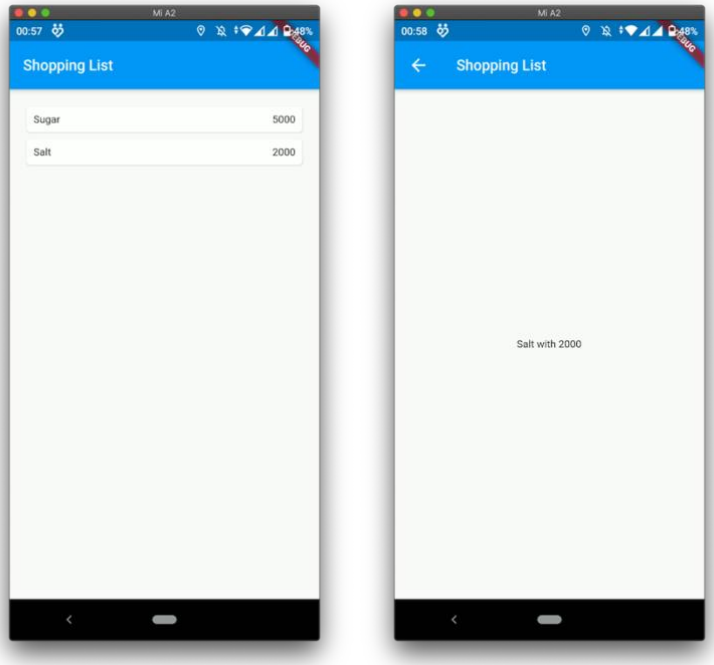
BAB 5. NAVIGASI DAN RUTE

5.1 Desain Aplikasi

Pada praktikum bab 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 berikut (mockup dibuat sesederhana mungkin, 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.



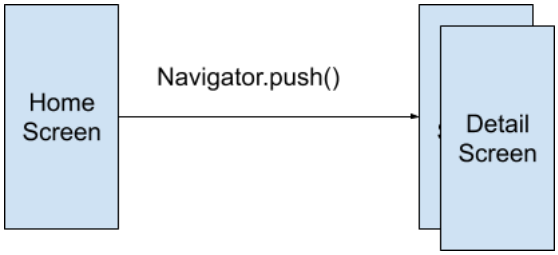
5.2 Teori

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

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

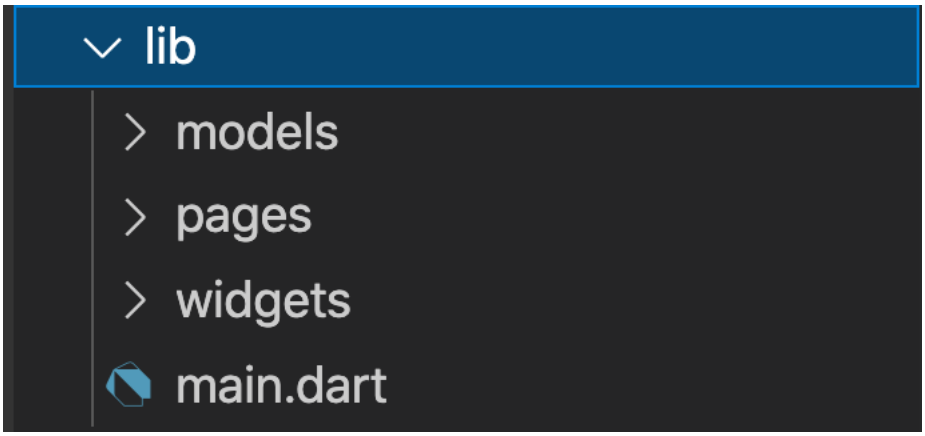
Sebuah Route umumnya dimasukkan (*push*) atau diambil (*pop*) dari dan ke tumpukan Navigator. Ketika sebuah halaman dilakukan operasi push, maka halaman tersebut akan

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.



5.3 Praktikum

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



5.3.1 Praktikum 1 - 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 pada potongan berikut.

```dart
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    throw UnimplementedError();
  }
}
```

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.

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

5.3.2 Praktikum 2 - Berpindah Halaman

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({this.name, this.price});
}
```

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)
  ];
```

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 juga widget Card. Kode yang telah umum, pada bagian ini tidak ditampilkan. Gambaran kode yang dibutuhkan dapat anda lihat sebagai berikut.

```
body: Container(
  margin: EdgeInsets.all(8),
  child: ListView.builder(
    padding: EdgeInsets.all(8),
    itemCount: items.length,
    itemBuilder: (context, index) {
      final item = items[index];
      return Card(
        child: Container(
          margin: EdgeInsets.all(8),
          child: Row(
            children: [
              Expanded(child: Text(item.name)),
              Expanded(
                child: Text(
                  item.price.toString(),
                  textAlign: TextAlign.end,
                ), // Text
              ) // Expanded
            ],
          ), // Row
        ), // Container
      ); // Card
    },
  ), // ListView.builder
), // Container
```

Jalankan aplikasi pada emulator atau pada device anda. Pastikan pada halaman awal telah berhasil menampilkan ListView. Jika ada kesalahan, segera perbaiki sebelum melanjutkan ke langkah berikutnya.

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 kali ini akan digunakan widget InkWell.

Untuk menambahkan sentuhan, letakkan cursor pada widget pembuka Card. Kemudian gunakan shortcut quick fix dari VSCode (`Ctrl + .` 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');
  },
```

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

5.3.3 Praktikum 3 - Pengiriman Data

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

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

5.3.4 Praktikum 4 - Pembacaan Data

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.

```
final Item itemArgs = ModalRoute.of(context).settings.arguments;
```

5.3.5 Tugas Praktikum

Pada hasil akhir dari aplikasi multi halaman yang telah anda selesaikan, tambahkan setidaknya satu atribut tambahan. Sesuaikan dan modifikasi tampilan sehingga menjadi aplikasi yang menarik. Selain itu, pecah widget menjadi kode yang lebih kecil.