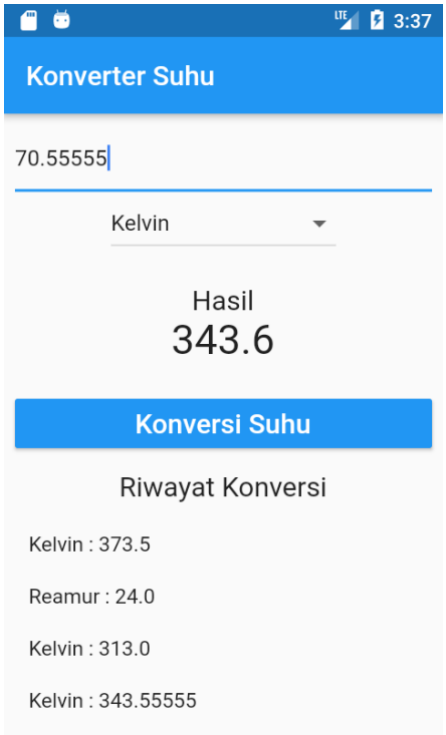
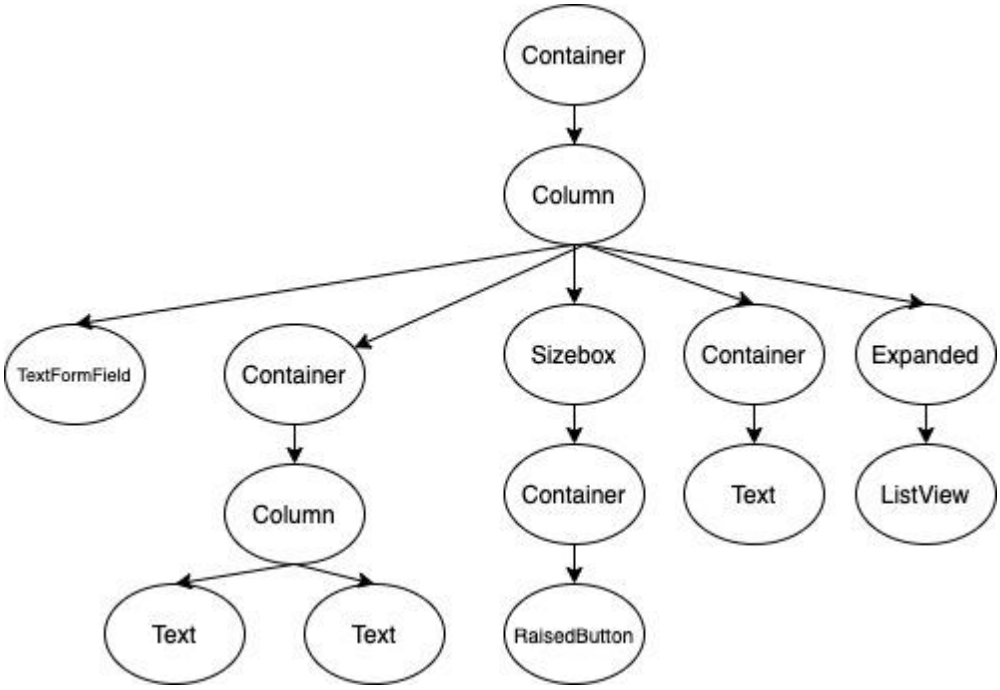
BAB 4. STATEFULL WIDGET DAN MAP

4.1 Desain Aplikasi

Pada praktikum bab 4 anda akan belajar mengenai cara menggunakan array atau list untuk membuat aplikasi dengan menggunakan flutter. Aplikasi yang dibuat akan mengembangkan aplikasi yang telah dibuat pada bab 3 yaitu untuk mengkonversi suhu. Berikut adalah desain mockup aplikasi yang akan dibuat. Pada aplikasi bab 4 akan ditambahkan DropdownButton dan ListView widget.



Widget tree yang digunakan penulis dapat diilustrasikan pada gambar di bawah ini. Widget tree yang dibuat tidak terbatas atau harus seperti pada gambar di bawah ini untuk membuat tampilan seperti mockup di atas.



4.2 Teori

4.2.1 List

Sebelum masuk pada praktikum akan dijelaskan pengantar tentang List pada Dart. List adalah collection yang seperti pada bahasa pemrograman yang lain. Pada bahasa pemrograman Dart, array adalah list of object sehingga disebut juga dengan list. Berikut ini adalah contoh list pada Dart

```
//contoh list of integer
var list = [1, 2, 3];

//contoh list of string
var list = [ 'Car', 'Boat', 'Plane', ];
```

List pada Dart mengguna index 0 sebagai nilai awal dan **list.length – 1** adalah index nilai terakhir dari sebuat list.

List pada Dart mengguna index 0 sebagai nilai awal dan **list.length – 1** adalah index nilai terakhir dari sebuat list.

```
var list = [1, 2, 3];
print(list.length == 3);
```

```
print (list[1] == 2);
```

Untuk membuat list pada saat compile time dapat menggunakan const

```
var constantList = const [1, 2, 3];
```

Mulai Dart 2.3 dikenalkan **spread operator** (…) dan **null-aware spread operator** (…?) yang digunakan insert banyak nilai ke collection. Contohnya pada source code di bawah ini, digunakan untuk memasukkan nilai dari list ke list2.

```
var list = [1, 2, 3];
var list2 = [0, ...list];
print(list2.length == 4);
```

Anda juga dapat menggunakan **null-aware spread operator** untuk menghindari exception ketika list bernilai null.

```
var list;
var list2 = [0, ...?list];
print(list2.length == 1);
```

Dart juga menyediakan **collection if** dan **collection for** yang digunakan untuk membuat collection dengan kondisi dan repetisi / perulangan. Berikut adalah contoh **collection if** untuk membuat list yang terdiri dari 3 atau 4 item sesuai dengan kondisi.

```
var nav = [
  'Home',
  'Furniture',
  'Plants',
  if (promoActive) 'Outlet'
];
```

Berikut adalah contoh **collection for** untuk memanipulasi list item sebelum ditambahkan pada list yang lain.

```
var listOfInts = [1, 2, 3];
var listOfStrings = [
    '#0', for (var i in listOfInts) '#$i'
];
print (listOfStrings[1] == '#1');
```

4.3 Praktikum 1 Menambahkan Dropdown Widget

4.3.1 Tambahkan code berikut dibawah Input

Pada bab 3, anda telah membuat source code untuk mengkonvert dari kelvin dan reamur. Cuplikan source code anda adalah sebagai berikut :

```

```

Berdasarkan ilustrasi mockup, pada praktikum ini kita akan menambahkan dropdown dan listview. Pada praktikum pertama akan ditambahkan dropdown dibawah Input. Tambahkan code berikut.

```
DropdownButton(
                items: [
                  DropdownMenuItem(
                      value: "Kelvin", child: Container(child: Text("Kelvin"))),
                  DropdownMenuItem(
                      value: "Reamur", child: Container(child: Text("Reamur"))),
                ],
                value: null,
                onChanged: (String changeValue) {},
              ),
```

4.3.2 Menggunakan Map pada DropDownButton

Dapat diamati pada source code dropdown items merupakan list yang terdiri dari DropdownMenuItem Widget. Hal ini akan tidak efektif karena code kita akan panjang jika

terdapat banyak item yang harus dimasukkan ke items. Hal ini dapat ditangani dengan menggunakan fungsi map pada list. Pertama buat list seperti berikut:

```
var listItem = [
    "Kelvin",
    "Reamur"
  ];
```

Selanjutnya ubah DropdownButton menjadi seperti berikut. Maksut dari code dibawah ini adalah :

`DropdownButton<String>` = String digunakan untuk memberi tipe data value dari Dropdown adalah bertipe String.

`listItem.map((String value) =` Digunakan untuk melakukan iterasi untuk setiap item dari listItem sesuai dengan parameter bertipe String.

```
DropdownButton<String>(
                items:
                  listItem.map((String value){
                    return DropdownMenuItem<String>(
                      value: value,
                      child: Text(value),);
                  }).toList(),
                value: null,
                onChanged: (String changeValue) {},
              ),
```

4.3.3 Mengeset Parameter value pada DropdownButton

Parameter value pada DropdownButton digunakan untuk mengeset nilai pada DropdownButton. Agar nilai defaultnya pada saat pertama muncul Text “Pilih Konversi ke Suhu” maka langkah yang dapat dilakukan adalah membuat variabel bertipe String seperti berikut pada class yang mengextend State.

```
class MyAppState extends State<MyApp> {


  double _inputUser = 0;
  double _kelvin = 0;
  double _reamur = 0;
  final inputController = TextEditingController();
  String _newValue = "Kelvin";
  double _result = 0;
.
.
.
.
```

Selanjutnya isi value dengan variabel `_newValue`.

```
DropdownButton<String>(
                items:
                  listItem.map((String value){
                    return DropdownMenuItem<String>(
                      value: value,
                      child: Text(value),);
                  }).toList(),
                value: _newValue,
                onChanged: (String changeValue) {},
              ),
```

4.3.4 Mengisi Fungsi pada onChanged

Pada saat ini nilai dari _newValue tidak diupdate ketika user mengubah pilihan pada dropdown. Hal tersebut dapat ditangani pada funsgi onChanged seperti berikut.

```
onChanged: (String changeValue) {
                  setState(() {
                    _newValue = changeValue;
                  });
                },
```

Atau anda juga dapat menggunakan fungsi terpisah kemudian gunakan fungsi dropdownOnChanged pada parameter value

```
void dropdownOnChanged(String changeValue){
      setState(() {
          _newValue = changeValue;
      });
  }
```

4.3.5 Mengubah Result.dart

Langkah selanjutnya adalah mengubah Result.dart sehingga memiliki tampilan dan hanya menerima 1 variabel.

```
class Result extends StatelessWidget {
  const Result({
    Key key,
    @required this.result,
  }) : super(key: key);

  final double result;

  @override
  Widget build(BuildContext context) {
    return
        Container(
          margin: EdgeInsets.only(top: 20,bottom: 20),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text("Hasil",style: TextStyle(fontSize: 20),),
              Text(
                result.toStringAsFixed(1),
                style: TextStyle(fontSize: 30),
              )
            ],
          ),
        );
    // );
  }
}
```

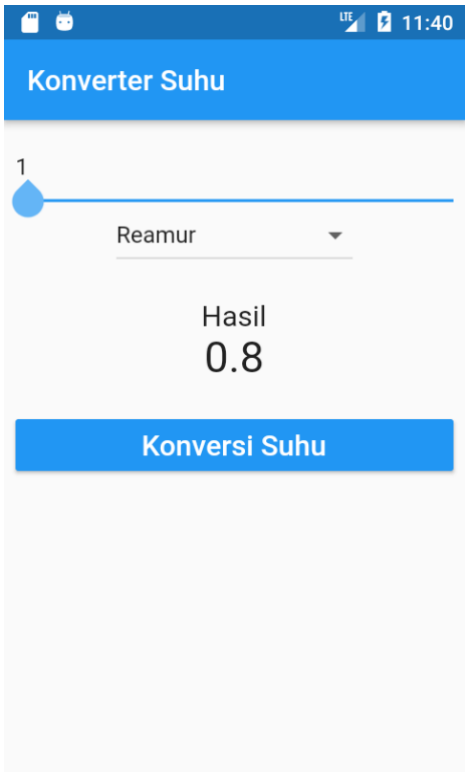
4.3.6 Mengubah Fungsi Perhitungan Suhu

Fungsi perhitungan suhu perlu untuk diubah sehingga hanya memproses konversi sesuai dengan pilihan pengguna.

```
void perhitunganSuhu() {
    setState(() {
      _inputUser = double.parse(inputController.text);


      if (_newValue == "Kelvin")
        _result = _inputUser + 273;
      else
        _result = (4 / 5) * _inputUser;
    });
  }
```

Sampai Pada tahap ini anda sudah belajar cara untuk menggunakan list dan fungsi map yang ada di Dart. Dan hasil aplikasi yang ada buat adalah sebagai berikut.



4.3.7 Tugas Praktikum 1

Kalian ubah proses pada aplikasi sehingga dapat memproses ketika ada perubahan dropdown tanpa di klik konversi suhu.

4.4 Tugas Praktikum 2 Menambahkan ListView Widget

Pada praktikum kedua sama dengan praktikum pertama yaitu dengan memanfaatkan map pada list dapat digunakan untuk mengisi history dari ListView. Langkah-langkah untuk menambah history adalah sebagai berikut.

1. Membuat variable bertipe List<String> dengan menggunakan code berikut

```
List<String> listViewItem =  List<String>();
```

2. Mengisi listViewItem setiap kali terjadi proses konversi
3. Menampilkan hasil listViewItem menggunakan map, menggunakan code berikut.

```
listViewItem.map((String value) {
                    return Container(
                        margin: EdgeInsets.all(10),
                        child: Text(
                          value,
                          style: TextStyle(fontSize: 15),
                        ));
                }).toList()
```

4. Menjadikan widget yang lebih kecil dengan menggunakan extract widget. Hasil akhir setelah diextract widget adalah seperti berikut.

```
body: Container(
          margin: EdgeInsets.all(8),
          child: Column(
            children: [
              Input(inputController: inputController),
              DropdownKonversi(listItem: listItem, newValue: _newValue,
dropdownOnChanged : dropdownOnChanged),
              Result(result: _result),
              Convert(konvertHandler: perhitunganSuhu),
              Container(
```

```
              margin: EdgeInsets.only(top: 10, bottom: 10),
              child: Text(
                "Riwayat Konversi",
                style: TextStyle(fontSize: 20),
              ),
            ),
            Expanded(
              child: RiwayatKonversi(listViewItem: listViewItem),
            ),
          ],
        ),
      ),
```

Lengkapilah source code yang kurang sehingga mendapatkan hasil akhir praktikum seperti berikut.