BAB II
DINAMIS UI

Pembelajaran yang akan dilakukan pada bab ini merupakan pembelajaran yang ditingkatkan lagi setelah praktikan mengetahui hal-hal mendasar terkait Flutter. Pada pembelajaran ini akan diajarkan terkait cara membuat tampilan aplikasi yang dinamis dengan menggunakan ListView.builder. UI pada aplikasi akan dibuat secara dinamis berdasarkan data yang tersedia. Gambarannya adalah jika ListView() pada bab 1 berisi widget-widget yang dibuat secara manual maka di bab ini akan mengisikan widget-widget di dalam ListView() dibentuk secara otomatis sesuai dengan data yang tersedia.

2.1 Tujuan Praktikum

Tujuan	Penjelasan
Mengetahui dan memahami bagaimana caranya membuat aplikasi dapat digulir	Pada bab ini, praktikan akan diajarkan tentang widget baru pada Flutter yang dapat membuat halaman dapat digulir, yakni **ListView()** dan turunannya, yaitu **ListView.builder()**
Membuat dan memanggil data *dummy*	Dalam bab ini, praktikan akan diajarkan bagaimana caranya membuat sebuah *list* yang berisikan data *dummy* dan cara memanggil data *dummy* tersebut ke tampilan aplikasi

2.2 Persyaratan Praktikum

Disarankan praktikan menggunakan *hardware* dan *software* sesuai pada dokumentasi ini. Apabila terdapat versi yang lumayan lampau dari versi yang direkomendasikan atau *hardware* yang lawas maka sebaiknya bertanya kepada Asisten Mengajar Shift.

***HARDWARE* YANG DIBUTUHKAN PRAKTIKUM**	**JENIS**
PC / Laptop CPU	**≥ 4 Cores**
PC / Laptop RAM	**≥ 8 GB**
PC / Laptop *Storage*	**≥ 10 GB**

***SOFTWARE* YANG DIBUTUHKAN PRAKTIKUM**
Android *Studio* / *Visual Studio Code*

2.3 Materi Praktikum

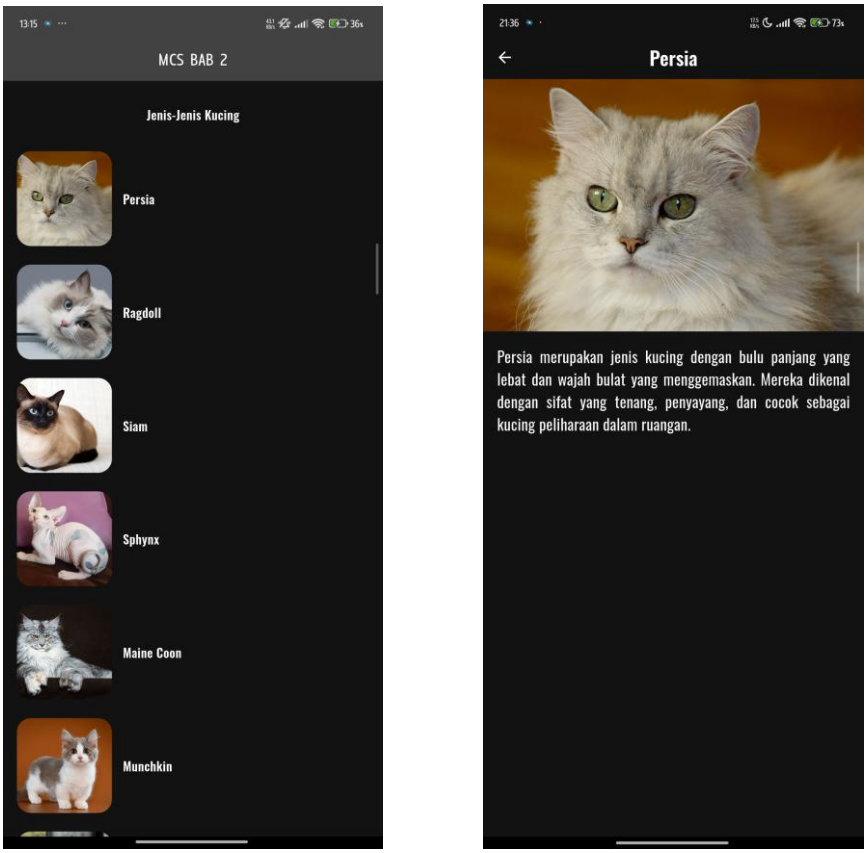
Pada praktikum MCS bab 2, praktikan akan diajarkan bagaimana caranya membuat sebuah aplikasi yang memiliki tampilan UI yang dinamis. Dinamis UI sendiri merupakan sebuah tampilan yang isi konten dari halaman tersebut dapat berubah-ubah secara otomatis, baik dari segi *content*, jumlah *content* yang ditampilkan, *light mode* atau *dark mode*, dan lain-lain. Pada praktikum kali ini, para praktikan akan membuat sebuah aplikasi daftar kucing, dimana jumlah data kucing yang ditampilkan bergantung pada data yang tersedia. Data tersebut dapat ditambahkan atau dikurangi sesuai dengan keinginan pengguna.

Pada flutter, untuk membuat aplikasi yang memiliki tampilan dinamis terdapat 1 widget yang dapat digunakan, yakni widget **ListView.builder().** Widget ini sama seperti widget ListView() yang digunakan pada bab praktikum sebelumnya. Namun, pada widget ini terdapat 1 properti tambahan, yakni properti **itemCount:** yang akan merender *content* aplikasi berdasarkan banyaknya nilai yang di*input* pada properti tersebut.

2.4 Prosedur Praktikum

2.4.1 Tampilan Aplikasi

Berikut merupakan tampilan dari aplikasi yang akan dibentuk pada praktikum bab 2.

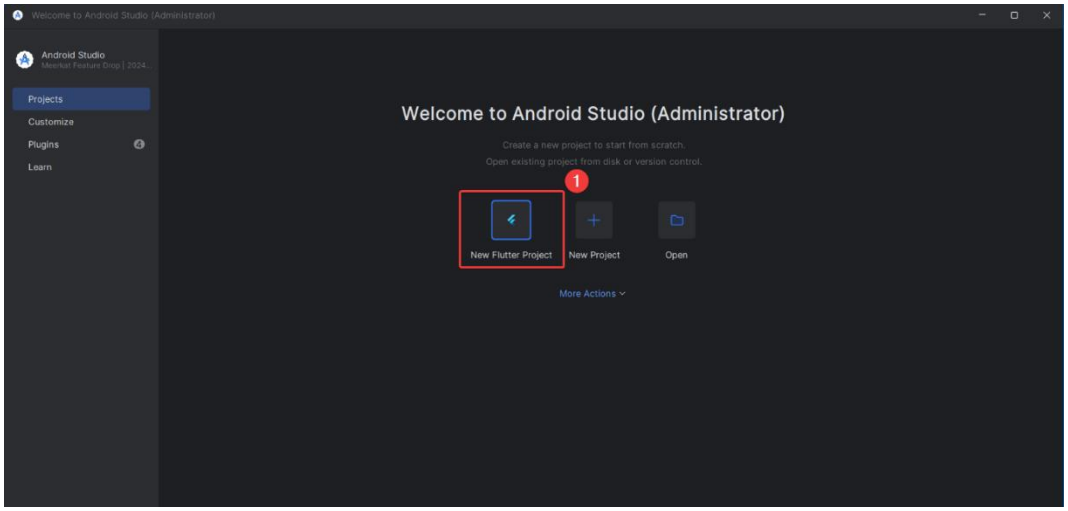


Gambar 2.1 Tampilan Halaman Aplikasi yang Akan diimplementasikan

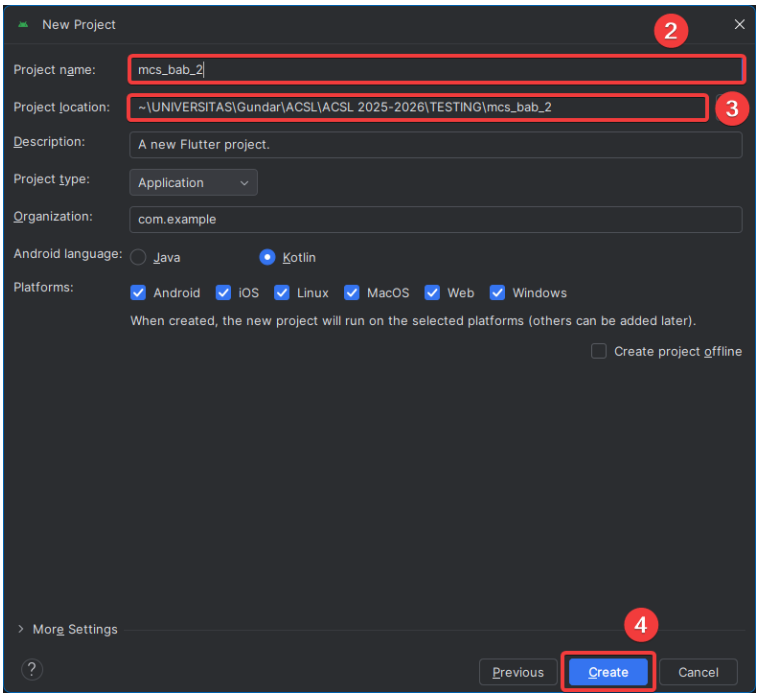
Penjelasan terkait bagaimana cara aplikasi bekerja akan diterangkan oleh asisten yang mengajar.

2.4.2 Implementasi Aplikasi

Dalam mengimplementasikan tampilan dari desain aplikasi di atas, terdapat beberapa langkah yang harus dilewati terlebih dahulu agar proses praktikum dapat berjalan dengan lancar dan terselesaikan sesuai dengan apa yang dituju. Langkah-langlah dalam pembuatan *project* baru sama seperti yang telah dilakukan pada praktikum pertemuan sebelumnya.



Gambar 2.2 Tampilan Awal *Software* Android *Studio*



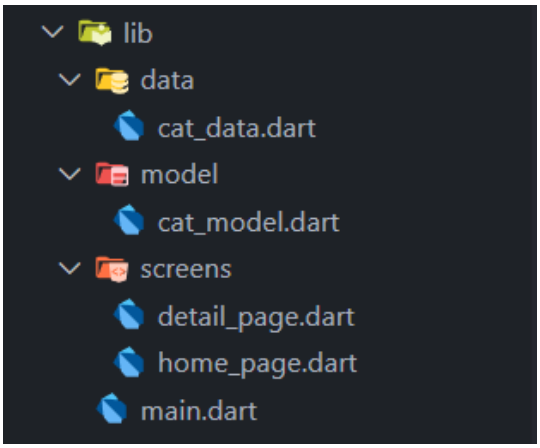
Gambar 2.3 Proses Pembuatan *Project* Baru

Jika file *project* sudah berhasil terbentuk, pastikan kembali bahwa tampilan yang diberikan oleh android studio sudah berada pada menu ***project***. Jika tampilan menu sudah berada pada bagian *project*, bukalah file **pubspec.yaml** dan *scroll* file tersebut hingga mendapatkan bagian **depedencies**. Pada file ini, kita akan menambahkan *package* baru yang diambil melalui halaman *website* **https://pub.dev/**, yakni *package* **google_fonts**. *Package* ini digunakan untuk

memberikan *styling* terhadap widget text yang digunakan pada aplikasi. Dengan menggunakan *package* ini, kita dapat memperindah teks aplikasi dengan tema font yang disediakan oleh google.

Gambar 2.4 *Package* yang digunakan

Setelah memasukkan *package* google_fonts ke dalam *project* flutter, langkah selanjutnya adalah membuat beberapa file dan folder yang dibentuk di dalam folder **lib,** sehingga *tree project* pada folder lib akan terlihat, seperti pada gambar berikut:



Gambar 2.5 Struktur *Tree Project*

Berikutnya masuklah ke dalam file **cat_model.dart** yang ada pada folder model dan ketikan kode program berikut:

```
class CatModel {
  String urlImage;
  String name;
```

```
  String desc;

  CatModel({
    required this.urlImage,
    required this.name,
    required this.desc});
}
```

Pada file tersebut, dibuatlah *class* baru bernama **CatModel()** yang di dalamnya terdapat beberapa *constructor*. *Class* ini digunakan sebagai kerangka dalam pembentukan sebuah data. Dalam *class* ini terdapat 3 variabel yang didefinisikan, yakni **urlImage, name,** dan **desc** dimana masing-masing variabel tersebut bersifat **required** pada saat pembuatan *constructor*. Hal tersebut menandakan bahwa ketika pengguna memanggil *class* tersebut, maka secara otomatis sistem meminta data untuk urlImage, name, dan desc dari data yang ingin ditambahkan. Kemudian bukalah file **cat_data.dart** yang tersimpan di dalam folder data dan masukkan kode program berikut:

```
List<CatModel> cats = [
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/persia.jpg",
    name: "Persia",
    desc:
        '''Persia merupakan jenis kucing dengan bulu panjang yang
lebat dan wajah bulat yang menggemaskan. Mereka dikenal dengan
sifat yang tenang, penyayang, dan cocok sebagai kucing peliharaan
dalam ruangan.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/ragdoll.png",
```

```
    name: "Ragdoll",
    desc:
        '''Ragdoll merupakan kucing besar dengan bulu panjang
yang halus dan mata biru memikat. Mereka terkenal dengan
kecenderungan mereka untuk rileks dan melonggar saat diangkat,
mirip dengan boneka ragdoll, dan sangat penyayang.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/siam.jpeg",
    name: "Siam",
    desc:
        '''Siam merupakan kucing dengan bulu pendek, mata biru
tajam, dan tubuh yang ramping. Mereka dikenal sebagai kucing vokal
yang suka berbicara dan memiliki kepribadian yang aktif serta
ramah.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/sphynx.jpg",
    name: "Sphynx",
    desc:
        '''Sphynx merupakan jenis kucing tanpa bulu yang memiliki
kulit lembut seperti kulit jeruk. Mereka sering menjadi perhatian
dengan penampilan yang unik dan ramah serta cerdas dalam perilaku
mereka.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/maine%20coon.jp
g",
```

```
    name: "Maine Coon",
    desc:
        '''Maine Coon merupakan salah satu kucing terbesar dengan
bulu panjang dan ekor berbulu tebal. Mereka memiliki sifat yang
ramah, lembut, dan cenderung energik, serta memiliki fisik yang
kuat.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/munchkin.jpg",
    name: "Munchkin",
    desc:
        '''Munchkin merupakan jenis kucing dengan tubuh pendek
dan kaki yang lebih pendek dari kucing biasa. Mereka memiliki
penampilan unik yang lucu dan aktif dalam bermain.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/bengal.png",
    name: "Bengal",
    desc:
        '''Bengal memiliki bulu yang berkilau dengan motif belang
yang mirip macan tutul. Mereka aktif, cerdas, dan suka bermain,
sering kali memiliki energi yang tinggi.''',
  ),
  CatModel(
    urlImage:
        "https://raw.githubusercontent.com/Fahmisbas/acsl-
mcs/master/Bab%203%20-
%20RecyclerView%20%26%20OnItemClickListener/cats/britain%20short
hair.jpg",
    name: "Britain Shorthair",
    desc:
```

```
        '''British Shorthair memiliki penampilan gemuk dengan
wajah yang bulat dan mata besar. Mereka cenderung tenang, santai,
dan mudah diurus, membuat mereka menjadi kucing peliharaan yang
populer.''',
  ),
];
```

File ini berisikan kumpulan data *dummy* yang tersimpan dalam variabel **cats** yang bertipe list. Seluruh data *dummy* tersebut dibentuk dengan menggunakan kerangka yang sebelumnya telah dibuat pada file cat_model.dart. Untuk menggunakan model kerangka yang telah dibuat sebelumnya, diperlukan pemanggilan terhadap *class* CatModel() pada saat mendefinisikan tipe data list untuk variabel cats. *Class* CatModel() ini perlu dipanggil jika kita ingin menggunakan kerangka yang telah dibuat, karena di dalam *class* CatModel() memiliki *constructor* yang berisikan kerangka untuk pengisian data.

Kemudian bukalah file **main.dart** untuk mendefinisikan beberapa konfigurasi yang akan digunakan pada aplikasi tersebut dan ketikan kode program berikut:

```
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MCS BAB 2',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: const ColorScheme.dark(),
        useMaterial3: true,
      ),
      home: const HomePage(),
    );}}
```

File ini hanya berisikan konfigurasi dasar yang diperlukan agar seluruh kode program dapat dijalankan dan memberikan tema terhadap aplikasi. File ini berisikan fungsi **main()** yang di dalamnya terdapat fungsi runApp() yang akan memanggil *class* **MyApp()** yang meng*extends* StatelessWidget. *Class* ini akan mengembalikan widget **MaterialApp()** yang di dalamnya terdapat beberapa konfigurasi sederhana untuk aplikasi. Konfigurasi ini merupakan konfigurasi yang digunakan pada pertemuan bab sebelumnya, hanya saja terdapat konfigurasi tambahan yang dilakukan pada properti **theme:** yang memanggil widget ThemeData() dengan nilai pada properti **colorScheme: const ColorScheme.dark()** yang akan memberikan tema gelap pada aplikasi.

Setelah melakukan konfigurasi dasar pada file main.dart, langkah selanjutnya adalah membangun tampilan dari halaman home dengan menggunakan kode program berikut:

```
class HomePage extends StatefulWidget {
  const HomePage({super.key});
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("MCS BAB 2", style: TextStyle(color:
Colors.white)),
        centerTitle: true,
        backgroundColor: Colors.grey[800],
      ),
      body: ListView(
        children: [
          const SizedBox(height: 32),

          Center(
```

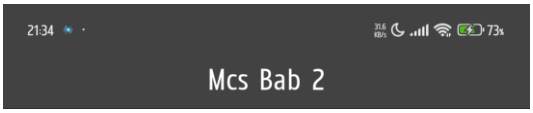
```
              child: Text(
                "Jenis-jenis Kucing",
                style: GoogleFonts.oswald(
                  fontSize: 16,
                  fontWeight: FontWeight.w600,
                ),
              ),
            ),

            const SizedBox(height: 24),

            ListView.builder(
              itemCount: cats.length,
              shrinkWrap: true,
              physics: const NeverScrollableScrollPhysics(),
              itemBuilder: (_, index) {
                CatModel cat = cats[index];
                return Container(
                  margin: const EdgeInsets.symmetric(
                    vertical: 12,
                    horizontal: 18,
                  ),
                  child: GestureDetector(
                    child: Row(
                      children: [
                        SizedBox(
                          height:
MediaQuery.of(context).size.width / 4,
                          width: MediaQuery.of(context).size.width
/ 4,
                          child: ClipRRect(
                            borderRadius:
BorderRadius.circular(18),
                            child:     Image.network(cat.urlImage,
fit: BoxFit.cover),
                          ),
                        ),
```

```
                        const SizedBox(width: 14),
                        Text(
                          cat.name,
                          style: GoogleFonts.oswald(
                            fontSize: 16,
                            fontWeight: FontWeight.w600,
                          ),
                        ),
                      ],
                    ),
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) {
                            return DetailPage(catModel: cat);
                          },
                        ),
                      );
                    },
                  ),
                );
              },
            ),
          ],
        ),
      );
  }
}
```

Class **HomePage** merupakan *class* pertama yang akan ditampilkan pada saat aplikasi dijalankan. *Class* ini akan meng*extends* StatefulWidget dan mengembalikan widget **Scaffold()** untuk membangun halamannya. Untuk membentuk halaman *home* yang berisikan daftar jenis-jenis kucing, penulisan kode dibagi menjadi ke dalam beberapa bagian, seperti **app bar** dan **body**.



Gambar 2.6 Tampilan AppBar Aplikasi

```
return Scaffold(
  appBar: AppBar(
    title: const Text("Mcs Bab 2", style: TextStyle(color:
Colors.white)),
    centerTitle: true,
    backgroundColor: Colors.grey[800],
  ),

  // ...

);
```

App bar pada aplikasi dibentuk dengan menggunakan widget **AppBar()** yang dikonfigurasi pada properti appBar: yang disediakan oleh widget Scaffold(). Pada widget AppBar() terdapat beberapa properti yang digunakan, seperti **title:** yang digunakan untuk memberikan judul pada AppBar dari aplikasi dengan mengembalikan widget Text(). Kemudian terdapat properti **centerTitle: true** untuk memposisikan title app bar berada di bagian dan properti **backgroundColor:** yang digunakan untuk memberikan warna *background* pada app bar aplikasi.

Kemudian pada properti **body:** digunakan widget ListView() untuk membangun fondasi awal dari bagian badan aplikasi. Widget ini memungkinkan halaman aplikasi dapat di*scroll*, sehingga seluruh *content* yang ingin ditampilkan dapat terlihat secara keseluruhan dengan cara menggeser layar perangkat yang digunakan.

Gambar 2.7 Tampilan Halaman Utama

```
return Scaffold(

  // ...

  body: ListView(
    children: [
      const SizedBox(height: 32),

      Center(
        child: Text(
          "Jenis-jenis Kucing",
          style: GoogleFonts.oswald(
            fontSize: 16,
            fontWeight: FontWeight.w600,
          ),
        ),
      ),

      const SizedBox(height: 24),
```

```
      ListView.builder(
        itemCount: cats.length,
        shrinkWrap: true,
        physics: const NeverScrollableScrollPhysics(),
        itemBuilder: (_, index) {
          CatModel cat = cats[index];
          return Container(
            margin: const EdgeInsets.symmetric(
              vertical: 12,
              horizontal: 18,
            ),
            child: GestureDetector(
              child: Row(
                children: [
                  SizedBox(
                    height:  MediaQuery.of(context).size.width  /
4,
                    width:  MediaQuery.of(context).size.width  /
4,
                    child: ClipRRect(
                      borderRadius: BorderRadius.circular(18),
                      child:   Image.network(cat.urlImage,   fit:
BoxFit.cover),
                    ),
                  ),
                  const SizedBox(width: 14),
                  Text(
                    cat.name,
                    style: GoogleFonts.oswald(
                      fontSize: 16,
                      fontWeight: FontWeight.w600,
                    ),
                  ),
                ],
              ),
              onTap: () {
```

```
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) {
                        return DetailPage(catModel: cat);
                      },
                    ),
                  );
                },
              ),
            );
          },
        ),
      ],
    ),
);
```

Widget ListView() memiliki properti children: yang dapat menampung banyak widget di dalamnya. Pada bagian awal, terdapat pemanggilan terhadap widget **Text()** yang dibungkus dengan menggunakan widget Center() agar teks yang dihasilkan berada pada bagian tengah layar aplikasi. Widget **Text()** tersebut akan menampilkan *string* bertuliskan "Jenis-jenis Kucing" dengan pemberian *styling* terhadap jenis font, ukuran, dan ketebalan dari huruf tersebut. *Styling* tersebut menggunakan *package* yang telah ditambahkan pada saat awal praktikum, yakni *package* google_fonts. *Package* tersebut dapat dikonfigurasi pada properti **style:** dengan melakukan pemanggilan terhadap **GoogleFonts.** dan diikuti dengan jenis font yang akan digunakan. Kemudian untuk mengatur ukuran dan ketebalan font, dapat dilakukan pada properti **fontSize:** dan **fontWeight:**.

Kemudian di bawah widget Text() diberikan jarak sebesar 24 pixel dengan memanggil widget **SizedBox()**. Berikutnya untuk memanggil daftar kucing yang jumlahnya dapat berubah-ubah sesuai dengan keinginan pengguna, widget **ListView.builder()** dapat menjadi opsi pilihannya. Widget ini memiliki fungsi yang sama seperti widget ListView(), tetapi dalam widget ini terdapat properti

itemCount: yang berfungsi untuk menetapkan seberapa banyak data yang akan ditampilkan pada daftar tersebut.

```
return Scaffold(

  // ...

  body: ListView(
    children: [

      // ...

      ListView.builder(
        itemCount: cats.length,
        shrinkWrap: true,
        physics: const NeverScrollableScrollPhysics(),
        itemBuilder: (_, index) {
          CatModel cat = cats[index];
          return Container(
            margin: const EdgeInsets.symmetric(
              vertical: 12,
              horizontal: 18,
            ),
            child: GestureDetector(
              child: Row(
                children: [
                  SizedBox(
                    height:  MediaQuery.of(context).size.width  /
4,
                    width:  MediaQuery.of(context).size.width  /
4,
                    child: ClipRRect(
                      borderRadius: BorderRadius.circular(18),
                      child:   Image.network(cat.urlImage,   fit:
BoxFit.cover),
                    ),
                  ),
                  const SizedBox(width: 14),
```

```
                    Text(
                      cat.name,
                      style: GoogleFonts.oswald(
                        fontSize: 16,
                        fontWeight: FontWeight.w600,
                      ),
                    ),
                  ],
                ),
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) {
                        return DetailPage(catModel: cat);
                      },
                    ),
                  );
                },
              ),
            );
          },
        ),
      ],
    ),
);
```

Selain menggunakan properti itemCount: untuk menentukan jumlah *content* yang ingin ditampilkan, terdapat penggunaan properti **shrinkWrap:** yang akan menyesuaikan tinggi berdasarkan jumlah *content* yang ditampilkan dan properti **physics: NeverScrollableScrollPhysics()** yang akan membuat item yang berada di dalam ListView.builder() tidak dapat dilakukan *scroll.* Hal ini sangat diperlukan agar tidak terdapat *double scrolling* pada satu halaman aplikasi. Kemudian terdapat properti **itemBuilder:** yang digunakan untuk membangun widget lain di dalam ListView.builder().

Pada praktikum ini, banyaknya *content* yang akan ditampilkan ditentukan berdasarkan banyaknya jumlah data yang tersimpan dalam variabel cats pada file cat_data.dart. Total jumlah data yang tersimpan pada variabel tersebut diambil dengan menggunakan argumen **cats.length.** Kemudian pada properti itemBuilder: terdapat sebuah deklarasi **CatModel cat = cats[index]** untuk memudahkan pengaksesan data yang tersimpan pada variabel cats berdasarkan indexnya.

Widget ListView.builder() akan mengembalikan sebuah **Container()** yang di dalamnya terdapat pemanggilan terhadap widget **Row()**. Widget Row() tersebut akan membuat widget di dalamnya tersusun secara *horizontal*. Adapun *content* yang berada di dalam widget tersebut adalah gambar dan nama kucing. Gambar kucing tersebut dipanggil dengan pemanggilan terhadap variabel **cat** yang diikuti dengan pemanggilan terhadap model **urlImage** (cat.urlImage). Untuk memanggil gambar tersebut diperlukan penggunaan dari widget **Image.network().** Gambar tersebut dibungkus dengan menggunakan widget **SizedBox()** dengan ukuran tinggi dan lebarnya diambil berdasarkan ukuran lebar layar perangkat dibagi dengan 4 dan di ujung dari setiap SizedBox() tersebut di*styling* dengan menggunakan BorderRadius.Circular() sebesar 18 agar bentuk gambar tidak terlalu kotak. Kemudian terdapat widget **Text()** yang akan mengembalikan nama dari setiap kucingnya dengan menggunakan perintah **cat.name**. Tulisan tersebut kemudian di*styling* dengan menggunakan *package* google_fonts untuk memperindah tampilan.

Untuk membuat daftar tersebut dapat disentuh dan menampilkan detail dari setiap data kucing, maka widget Row() yang membungkus gambar dan nama kucing harus dibungkus dengan menggunakan widget **GestureDetector()**. Widget ini memiliki properti **onTap:** yang dapat digunakan untuk memberikan suatu tindakan pada saat pengguna menekan salah satu daftar kucing tersebut. Tindakan yang diberikan oleh sistem pada saat pengguna menekan salah satu daftar kucing tersebut adalah melakukan *direct* ke halaman detail kucing dengan menggunakan perintah **Navigator.push().** Data yang dibawa ke halaman detail merupakan data yang dikirimkan oleh *class* HomePage berdasarkan index yang diakses.

Setelah melakukan penulisan kode pada halaman utama, langkah selanjutnya adalah menuliskan kode untuk halaman detail. Penulisan kode tersebut dilakukan pada file **detail_page.dart**. Berikut merupakan kode yang digunakan pada file tersebut:

```
class DetailPage extends StatefulWidget {
  CatModel catModel;
  DetailPage({super.key, required this.catModel});

  @override
  State<DetailPage> createState() => _DetailPageState();
}

class _DetailPageState extends State<DetailPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          widget.catModel.name,
          style: GoogleFonts.oswald(fontSize: 20, fontWeight:
FontWeight.w600),
        ),
        centerTitle: true,
      ),
      body: ListView(
        children: [
          Image.network(widget.catModel.urlImage),
          const SizedBox(height: 18),
          Container(
            margin: const EdgeInsets.symmetric(horizontal: 18),
            child: Text(
              widget.catModel.desc,
              style: GoogleFonts.oswald(fontSize: 16),
              textAlign: TextAlign.justify,
            ),
          ),
```

```
          ],
        ),);
    }
}
```

Halaman detail kucing dibentuk dari *class* **DetailPage** yang meng*extends* StatefulWidget. *Class* tersebut memiliki parameter catModel yang bertipe data CatModel. *Class* ini akan mengembalikan widget **Scaffold()** untuk membentuk halaman aplikasi dimana terdapat 2 bagian, yakni app bar dan body. Bagian app bar pada halaman detail ini akan menampilkan nama kucing yang dipilih pada halaman sebelunya. Nama kucing tersebut nantinya akan berada pada bagian tengah dan app bar tersebut di*styling* dengan menggunakan *package* google_fonts. Berikutnya pada properti **body:** digunakan widget ListView() yang di dalamnya akan mengembalikan widget **Image.network()** untuk mengembalikan gambar kucing dan widget **Text()** yang akan menampilkan deskripsi dari kucing yang telah dipilih pada halaman sebelumnya.