

LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

Android Fundamental 2 - WEEK 7



Disusun oleh:

Nama : Alfath Roziq Widhayaka

Nim : L0122012

Kelas : A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

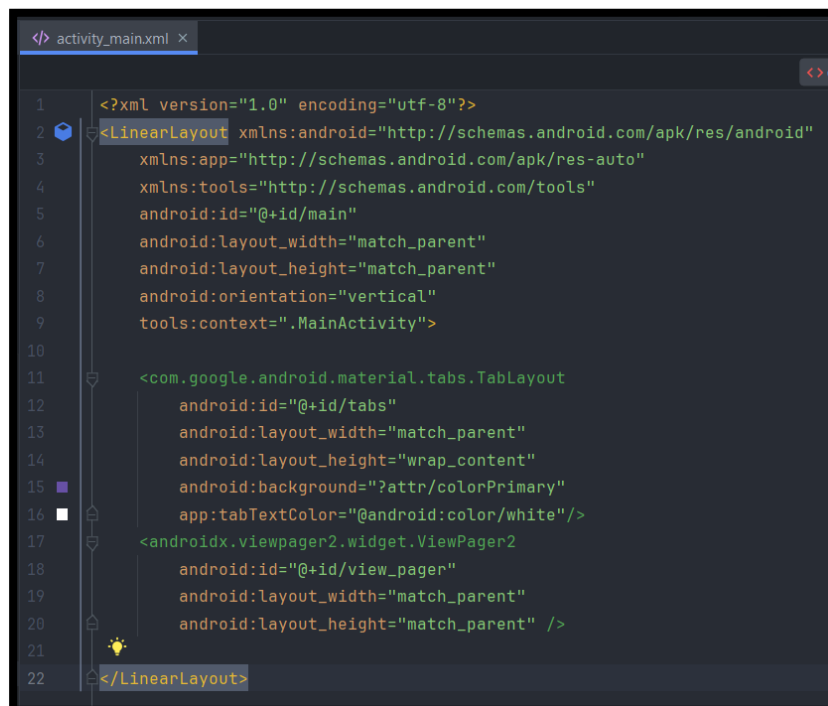
2024

1. Screenshot Source Code

Source code dibawah ini merupakan source code praktikum PAB ke-7 yang menggunakan **Tab Layout** yang berisi **RecyclerView** di Android Studio. **Tab Layout** disini hanya digunakan untuk memfasilitasi sebuah **RecyclerView** agar dapat berpindah fragment dengan cepat, sedangkan **RecyclerView** digunakan untuk menampilkan dan membuat objek sesuai dengan ukuran layar. Aplikasi yang menggunakan **RecyclerView** menjadi efisien dan lebih menghemat memori. Saya disini membuat list item layout dengan tema “**Nama Kucing dan Anjing**”.

A. Membuat Tab Layout

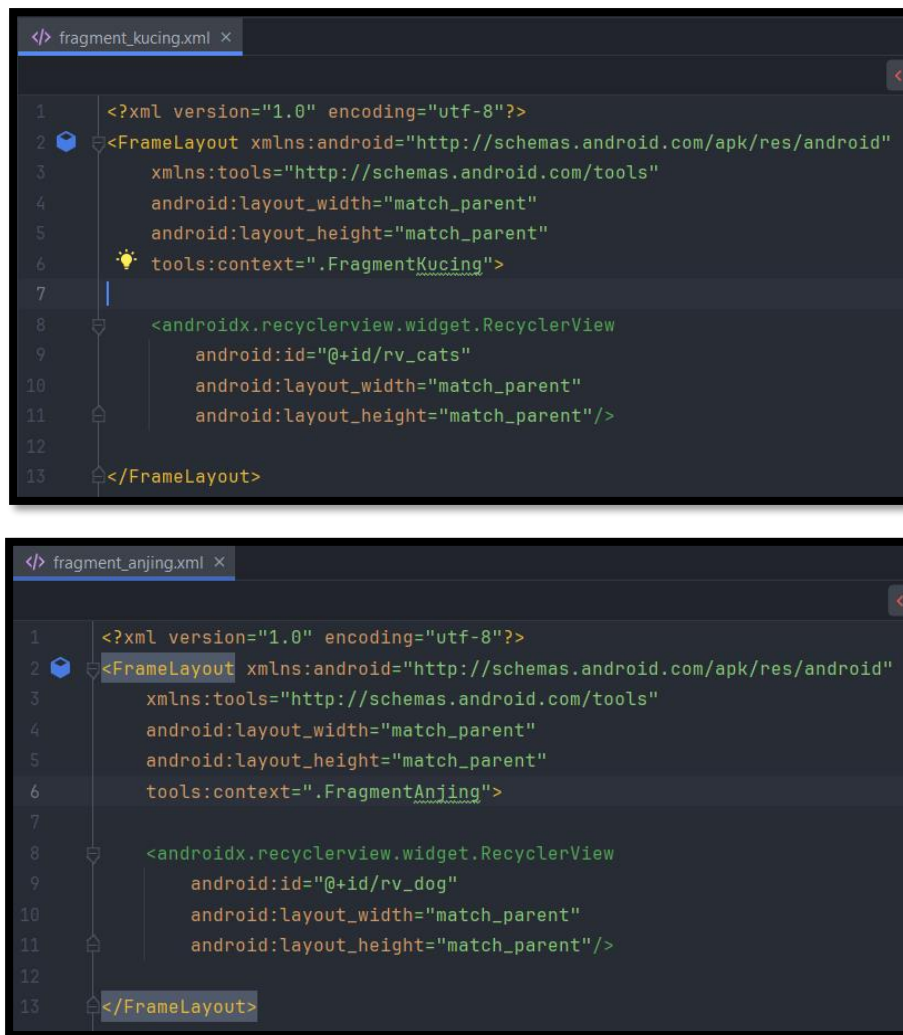
1. Menambahkan Tab Layout dan View Pager 2 pada activity_main.xml agar tab layout ini dapat digunakan di semua fragment.



Gambar 1 sc activity_main.xml

Kode XML di atas mendefinisikan tata letak antarmuka pengguna untuk aktivitas dalam aplikasi Android. LinearLayout digunakan sebagai wadah utama dengan orientasi vertikal, di dalamnya terdapat TabLayout dan ViewPager2. TabLayout digunakan untuk menampilkan tab yang dapat digunakan pengguna untuk navigasi antara beberapa fragmen atau halaman dalam ViewPager2. ViewPager2 adalah tampilan yang memungkinkan pengguna untuk meluncurkan antarmuka pengguna dari satu tampilan ke tampilan lainnya, seperti meluncurkan antara berbagai fragmen dalam aplikasi. Ini memberikan tata letak yang responsif dan intuitif bagi pengguna untuk berinteraksi dengan konten aplikasi yang berbeda.

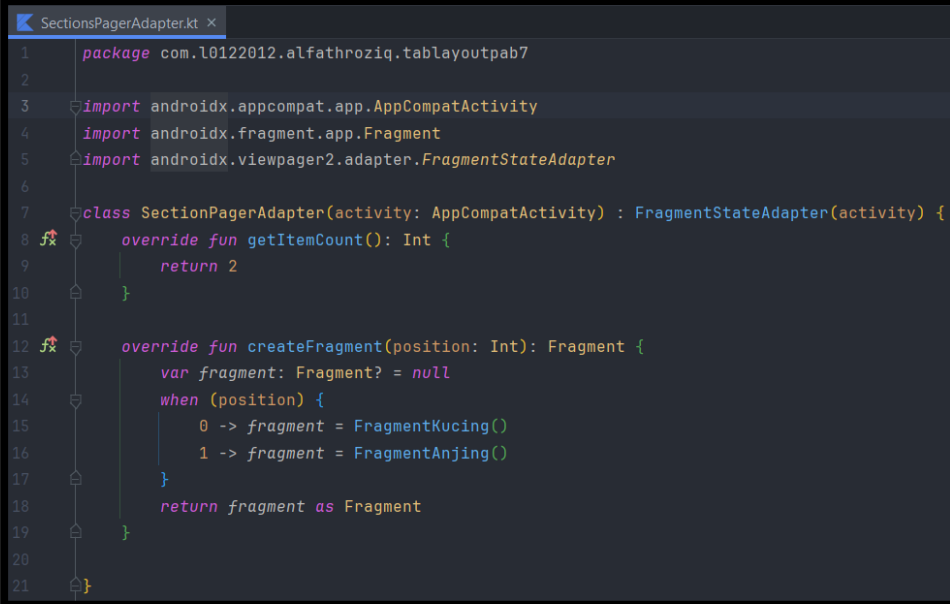
2. Tambahkan 2 blank fragment yang nantinya akan diisi untuk RecyclerView, untuk fragment tersebut diberi nama “**FragmentKucing**” dan “**FragmentAnjing**”.



Gambar 2 sc fragment_kucing dan anjing.xml

Kedua kode diatas merupakan isi dari 2 xml yang baru saja ditambahkan tadi, ketika ditambahkan 2 fragment blank baru maka akan otomatis membuat xml. Lalu mengisi RecyclerView pada kedua XML tersebut agar dapat digunakan pada kedua Fragment tersebut.

3. Membuat Class **SectionsPagerAdapter** untuk mengatur **ViewPager2**



```

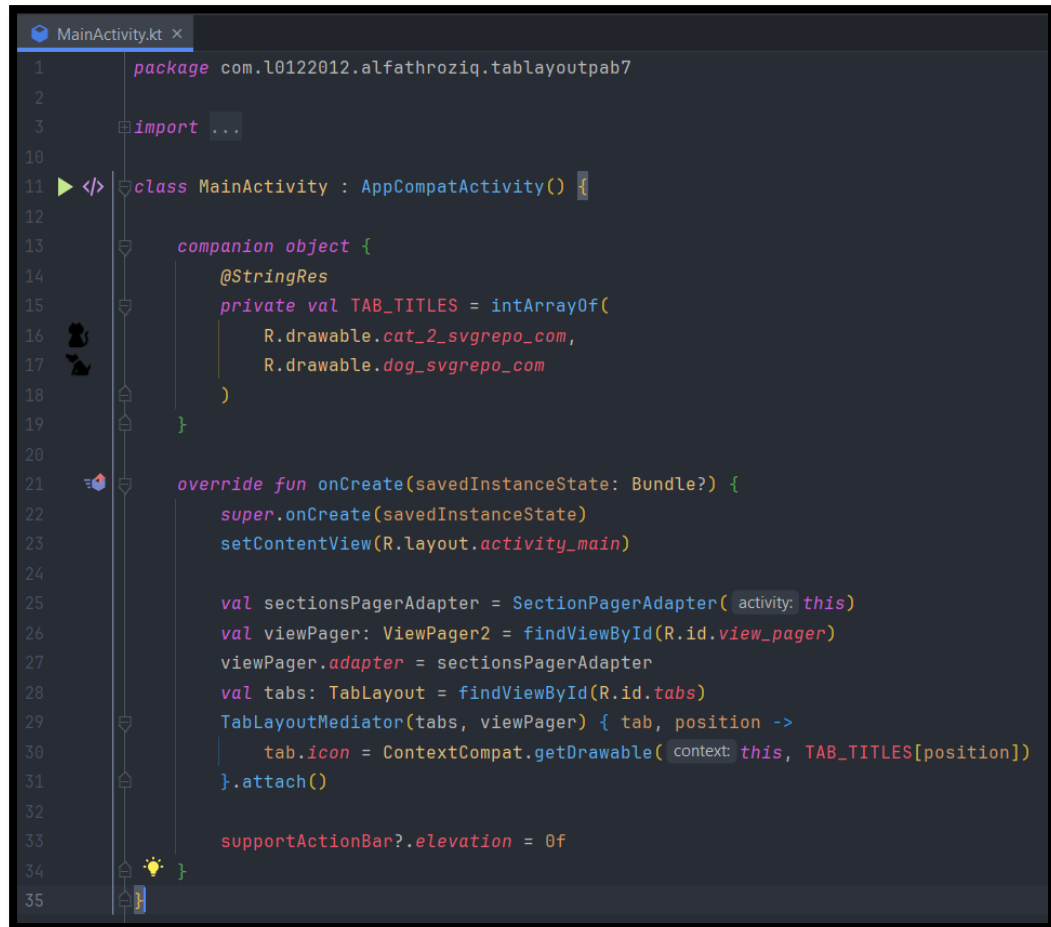
1  package com.l0122012.alfathroziq.tablayoutpab7
2
3  import androidx.appcompat.app.AppCompatActivity
4  import androidx.fragment.app.Fragment
5  import androidx.viewpager2.adapter.FragmentStateAdapter
6
7  class SectionPagerAdapter(activity: AppCompatActivity) : FragmentStateAdapter(activity) {
8      override fun getItemCount(): Int {
9          return 2
10     }
11
12     override fun createFragment(position: Int): Fragment {
13         var fragment: Fragment? = null
14         when (position) {
15             0 -> fragment = FragmentKucing()
16             1 -> fragment = FragmentAnjing()
17         }
18         return fragment as Fragment
19     }
20 }
21

```

Gambar 3 SectionPagerAdapter.kt

Kode Kotlin di atas menggunakan TabLayout dan ViewPager2 untuk menampilkan dua fragmen yang berbeda. Kelas **SectionPagerAdapter** adalah adapter yang digunakan untuk mengelola fragmen dalam ViewPager2. Dengan mewarisi dari **FragmentStateAdapter**, kelas ini digunakan sebagai pengelolaan fragmen secara dinamis saat pengguna berinteraksi dengan tata letak. Metode **getItemCount** menentukan jumlah total fragmen yang akan ditampilkan, dalam kasus ini adalah dua. Metode **createFragment** digunakan untuk membuat dan mengembalikan fragmen yang sesuai dengan posisi tertentu. Dalam contoh ini, terdapat dua fragmen: **FragmentKucing** untuk posisi 0 dan **FragmentAnjing** untuk posisi 1. Ini memungkinkan aplikasi untuk menampilkan konten yang relevan tergantung pada tab yang dipilih oleh pengguna.

4. Memanggil kelas baru tadi ke **MainActivity.kt**, selanjutnya menginisialisasi vektor SVG yang digunakan untuk mengganti nama tab layout yang diatas menjadi logo kucing dan anjing.



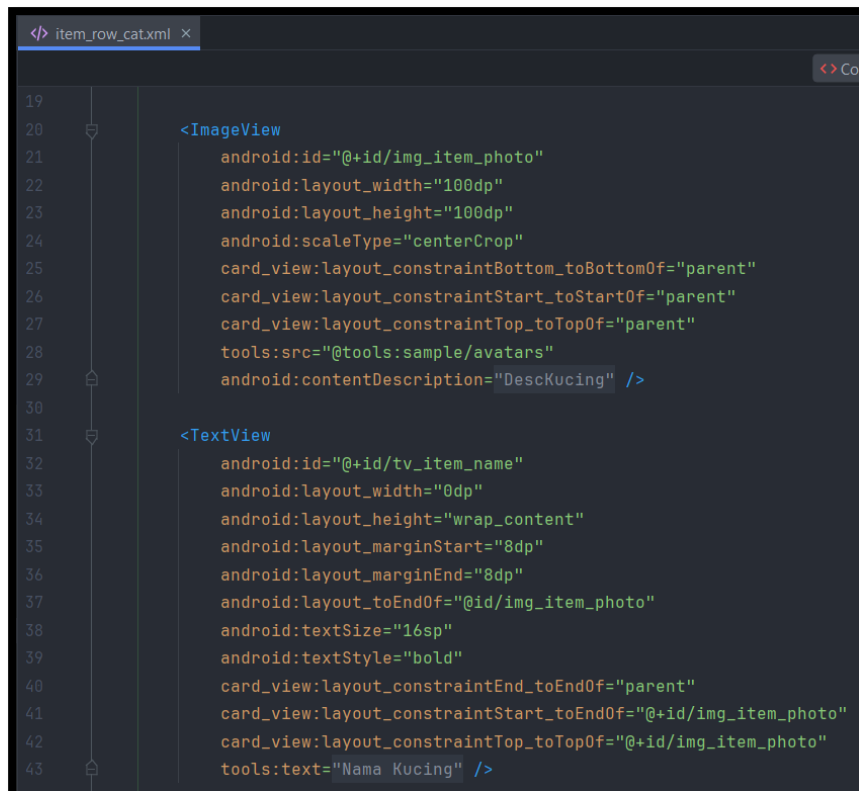
```
1 package com.l0122012.alfathroziq.tablayoutpab7
2
3 import ...
4
5
6
7
8
9
10
11 class MainActivity : AppCompatActivity() {
12
13     companion object {
14         @StringRes
15         private val TAB_TITLES = intArrayOf(
16             R.drawable.cat_2_svgrepo_com,
17             R.drawable.dog_svgrepo_com
18         )
19     }
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContentView(R.layout.activity_main)
24
25         val sectionsPagerAdapter = SectionsPagerAdapter( activity: this)
26         val viewPager: ViewPager2 = findViewById(R.id.view_pager)
27         viewPager.adapter = sectionsPagerAdapter
28         val tabs: TabLayout = findViewById(R.id.tabs)
29         TabLayoutMediator(tabs, viewPager) { tab, position ->
30             tab.icon = ContextCompat.getDrawable( context: this, TAB_TITLES[position])
31         }.attach()
32
33         supportActionBar?.elevation = 0f
34     }
35 }
```

Gambar 4 MainActivity.kt

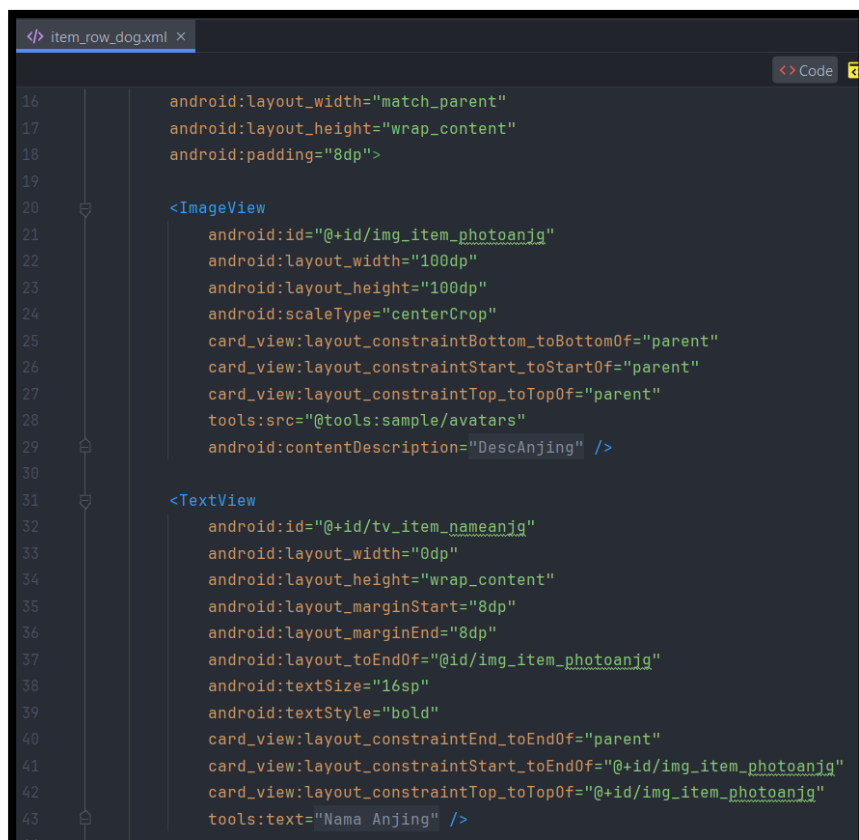
Kode Kotlin di atas adalah implementasi dari aktivitas utama dalam aplikasi Android yang menggunakan TabLayout dan ViewPager2 untuk menampilkan dua fragmen yang berbeda. Saat aktivitas dibuat **onCreate**, sebuah **SectionsPagerAdapter** dibuat untuk mengelola fragmen dalam ViewPager2. **TabLayoutMediator** digunakan untuk menghubungkan TabLayout dengan ViewPager2, sehingga setiap tab memiliki ikon yang sesuai dengan posisi fragmen yang terkait. Icon tab diambil dari array **TAB_TITLES**, yang berisi referensi ke gambar-gambar yang disimpan di drawable resources aplikasi. Dengan demikian, pengguna dapat berinteraksi dengan tab untuk berpindah antara fragmen yang berkorespondensi, memungkinkan navigasi yang intuitif dalam aplikasi. Elemen UI lainnya seperti ActionBar elevation juga disesuaikan untuk menciptakan tampilan yang konsisten dan menarik.

B. Langkah-langkah menampilkan data menggunakan RecyclerView

1. Membuat layout XML untuk layout item pada `item_row_cat.xml` dan `item_row_dog.xml`.



```
<?xml version="1.0" encoding="utf-8" android:namespace="android.support.design.widget" android:prefix="card_view">
<include layout="@layout/item_cat_card" />
<ImageView
    android:id="@+id/img_item_photo"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:scaleType="centerCrop"
    card_view:layout_constraintBottom_toBottomOf="parent"
    card_view:layout_constraintStart_toStartOf="parent"
    card_view:layout_constraintTop_toTopOf="parent"
    tools:src="@tools:sample/avatars"
    android:contentDescription="DescKucing" />
<TextView
    android:id="@+id/tv_item_name"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_toEndOf="@id/img_item_photo"
    android:textSize="16sp"
    android:textStyle="bold"
    card_view:layout_constraintEnd_toEndOf="parent"
    card_view:layout_constraintStart_toEndOf="@id/img_item_photo"
    card_view:layout_constraintTop_toTopOf="@id/img_item_photo"
    tools:text="Nama Kucing" />
```

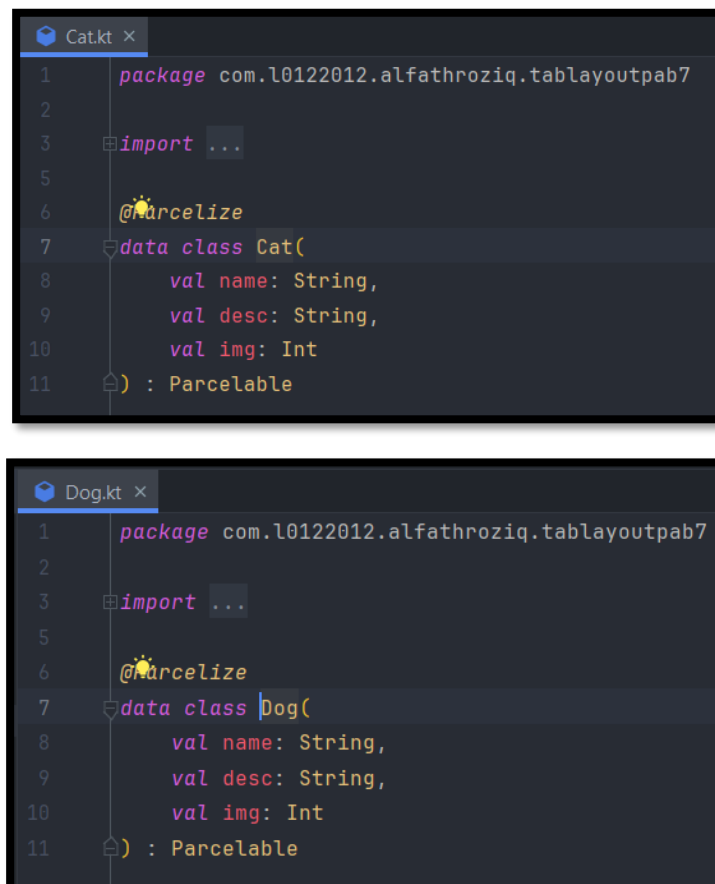


```
<?xml version="1.0" encoding="utf-8" android:namespace="android.support.design.widget" android:prefix="card_view">
<include layout="@layout/item_dog_card" />
<ImageView
    android:id="@+id/img_item_photoanjg"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:scaleType="centerCrop"
    card_view:layout_constraintBottom_toBottomOf="parent"
    card_view:layout_constraintStart_toStartOf="parent"
    card_view:layout_constraintTop_toTopOf="parent"
    tools:src="@tools:sample/avatars"
    android:contentDescription="DescAnjing" />
<TextView
    android:id="@+id/tv_item_nameanjg"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_toEndOf="@id/img_item_photoanjg"
    android:textSize="16sp"
    android:textStyle="bold"
    card_view:layout_constraintEnd_toEndOf="parent"
    card_view:layout_constraintStart_toEndOf="@id/img_item_photoanjg"
    card_view:layout_constraintTop_toTopOf="@id/img_item_photoanjg"
    tools:text="Nama Anjing" />
```

Gambar 6, 7 sc layout `item_row_logocat` dan `anjing.xml`

Source code XML milik **item_row_cat** dan **dog.xml** di atas mendefinisikan tata letak untuk sebuah tampilan kartu (CardView) dalam sebuah aplikasi Android. Kartu tersebut berisi **gambar kucing** dan **anjing** yang diletakkan di sebelah kiri atas dengan lebar 100dp dan tinggi 100dp, diikuti oleh **nama kucing** yang diletakkan di sebelah kanan gambar kucing dengan huruf ukuran 16sp, dan **deskripsi kucing** yang diletakkan di bawah nama kucing dan anjing dengan maksimal 5 baris teks. Tampilan tersebut diatur secara responsif dengan lebar yang cocok dengan parent dan tinggi yang mengikuti konten, serta memiliki margin yang ditentukan untuk tata letak yang lebih baik.

2. Menambah class baru yaitu **Cat.kt** dan **Dog.kt** dan menambahkan pewarisan **RecyclerView.Adapter**.



```
Cat.kt
1 package com.l0122012.alfathroziq.tablayoutpab7
2
3 import ...
4
5
6 @Parcelize
7 data class Cat(
8     val name: String,
9     val desc: String,
10    val img: Int
11 ) : Parcelable

Dog.kt
1 package com.l0122012.alfathroziq.tablayoutpab7
2
3 import ...
4
5
6 @Parcelize
7 data class Dog(
8     val name: String,
9     val desc: String,
10    val img: Int
11 ) : Parcelable
```

Gambar 8,9 sc class cat dan dog.kt

Source code di atas adalah definisi dari data class Kotlin bernama **Cat.kt** dan **Dog.kt** yang diimplementasikan dengan parcelable menggunakan anotasi **@Parcelize**. Kelas Cat dan Dog memiliki tiga properti yaitu **name** untuk nama kucing dan anjing, **desc** untuk deskripsi kucing dan anjing, dan **img** untuk menyimpan referensi gambar kucing dan anjing dalam bentuk Integer. Menggunakan **Parcelable** memungkinkan objek kelas Cat dan Dog dapat dikirimkan antar komponen aplikasi Android dengan mudah dan efisien.

```

1 package com.l0122012.alfathroziq.tablayoutpab7
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import android.widget.ImageView
7 import android.widget.TextView
8 import androidx.recyclerview.widget.RecyclerView
9
10 class ListCatAdapter(private val listCat: ArrayList<Cat>) : RecyclerView.Adapter<ListCatAdapter.ListViewHolder>() {
11     class ListViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
12         val imgPhoto: ImageView = itemView.findViewById(R.id.img_item_photo)
13         val tvName: TextView = itemView.findViewById(R.id.tv_item_name)
14         val tvDescription: TextView = itemView.findViewById(R.id.tv_item_description)
15     }
16
17     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ListViewHolder {
18         val view: View = LayoutInflater.from(parent.context).inflate(R.layout.item_row_cat, parent, attachToRoot: false)
19         return ListViewHolder(view)
20     }
21
22     override fun getItemCount(): Int = listCat.size
23
24     override fun onBindViewHolder(holder: ListViewHolder, position: Int) {
25         val (name, desc, img) = listCat[position]
26         holder.imgPhoto.setImageResource(img)
27         holder.tvName.text = name
28         holder.tvDescription.text = desc
29     }
30 }

```

Gambar 10 sc class ListCatAdapter.kt

Source code di atas adalah adapter untuk **RecyclerView** yang bertugas menghubungkan data **listCat** dengan tampilan item pada RecyclerView. Kelas **ListCatAdapter.kt** menerima **ArrayList<Cat>** sebagai parameter konstruktor untuk menampilkan daftar kucing dan anjing. Setiap item pada RecyclerView ditampilkan menggunakan layout **item_row_cat** tadi. Untuk pembuatan **ListDogAdapter.kt** juga ditambahkan sebagai penghubung class **Dog**.

- Memasukkan data string masing-masing data kucing dan data anjing pada **strings.xml**.

```
9      <string-array name="data_kucing">
10          <item>Kucing Anggora</item>
11          <item>Kucing Bengal</item>
12          <item>Kucing Kampung</item>
13          <item>Kucing Maine Coon</item>
14          <item>Kucing Munchkin</item>
15          <item>Kucing Persia</item>
16          <item>Kucing Ragdoll</item>
17          <item>Kucing Russian Blue</item>
18          <item>Kucing Siam</item>
19          <item>Kucing Sphynx</item>
20      </string-array>

48     <string-array name="data_anjing">
49         <item>Anjing Beagle</item>
50         <item>Anjing Bulldog</item>
51         <item>Anjing Kampung</item>
52         <item>Anjing Labrador</item>
53         <item>Anjing Pudel</item>
54         <item>Anjing Rotweiler</item>
55         <item>Anjing Sepherd</item>
56         <item>Anjing Shorthead Pointer</item>
57         <item>Anjing Dachsun</item>
58         <item>Anjing Golder</item>
59     </string-array>
```

Gambar 11,12 sc strings.xml

Source code diatas merupakan strings yang digunakan untuk sumber daya dalam aplikasi ini seperti nama kucing dan anjing, deskripsi kucing dan anjing, dan gambar kucing dan anjing. Terdapat dua set array string yang berisi data kucing dan anjing dari dua wilayah berbeda, yaitu **data_kucing** dan **data_anjing**. Setiap array berisi nama-nama kucing dan anjing, deskripsi singkat tentang kucing dan anjing tersebut. Terdapat dua set array integer yang berisi referensi gambar untuk masing-masing data kucing dan anjing dalam setiap wilayah yaitu **img_kucing** dan **img_anjing**.

- Menambahkan RecyclerView dengan adapter dan layoutManager pada **FragmentKucing.kt** dan **FragmentAnjing.kt**.

```
32     @SuppressWarnings("Recycle")
33     private fun getListCats() : ArrayList<Cat> {
34         val dataName = resources.getStringArray(R.array.data_kucing)
35         val dataDesc = resources.getStringArray(R.array.desc_kucing)
36         val dataImg = resources.obtainTypedArray(R.array.img_kucing)
37         val listHero = ArrayList<Cat>()
38         for (i in dataName.indices) {
39             val hero = Cat(dataName[i], dataDesc[i], dataImg.getResourceId(i, -1))
40             listHero.add(hero)
41         }
42         return listHero
43     }
44
45     private fun showRecyclerList(rvCats: RecyclerView) {
46         rvCats.layoutManager = LinearLayoutManager(requireContext())
47         val listCatAdapter = ListCatAdapter(list)
48         rvCats.adapter = listCatAdapter
49     }
50 }
```

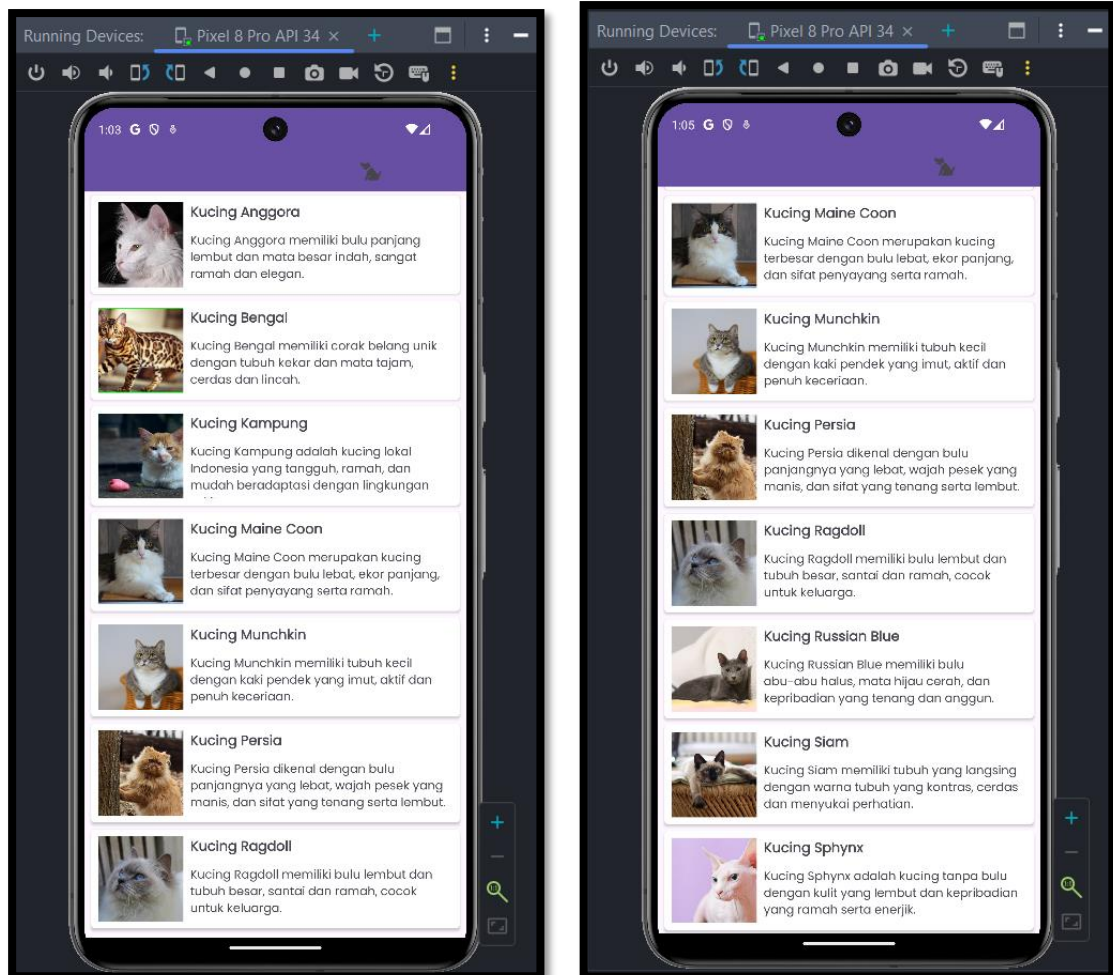
Gambar 13,14 sc class MainActivity.kt recyclerview

Source code di atas adalah kelas **FragmentKucing.kt** dan **FragmentAnjing.kt**. **RecyclerView** ditambahkan pada kelas ini untuk menampilkan daftar kucing dan anjing yang diinisialisasi dan diatur dengan kelas **ListCatAdapter.kt** dan **ListDogAdapter.kt**. Daftar kucing dan anjing diambil dari string array (data_kucing, desc_kucing) dan array of image resources (img_kucing) menggunakan metode **getListCats()** dan **getListDogs** dari **string.xml**. Metode **showRecyclerView()** ditambahkan dan digunakan untuk menampilkan daftar data kucing dan anjingmo dalam bentuk daftar, dengan setiap item memiliki fungsi klik.

2. Screenshot Terminal

Berikut merupakan hasil dari praktikum PAB ke-7 yang menerapkan materi Tab Layout yang berisi RecyclerView Berikut merupakan hasil pada pekerjaan milik saya dengan tema ”Nama Kucing dan Anjing”

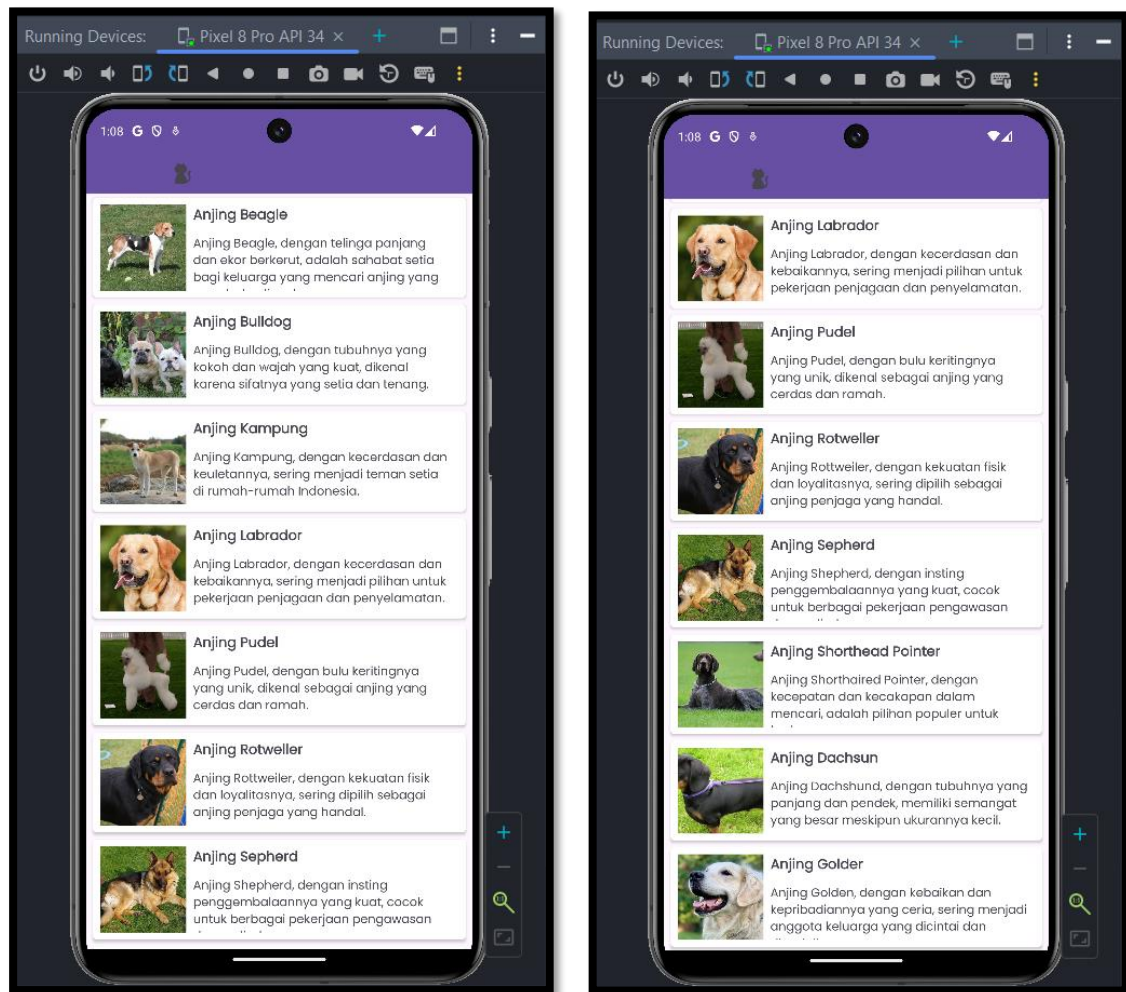
A. Tab Layout RecyclerView “Kucing”



Gambar 1 hasil Tab Layout dan RecyclerView pada halaman daftar kucing

Gambar di atas menampilkan hasil dari emulator setelah aplikasi dijalankan. Untuk item kucing terdapat 10 item pada aplikasi, sehingga pengguna dapat melakukan scrolling ke bawah karena menggunakan fungsi **RecyclerView**. Tata letak setiap item tersusun rapi dan teratur, tidak ada yang terlihat rusak atau terganggu dalam penampilannya. Gambar diatas merupakan susunan menggunakan sistem **List**. Gambar kiri merupakan tampilan awal, gambar kanan merupakan tampilan setelah scroll. Terlihat ketika pengguna menuju ke tab ”Kucing” maka untuk logo kucing menjadi transparan, namun untuk logo anjing menjadi ada logonya seperti di gambar. Begitu juga sebaliknya.

B. Tab Layout RecyclerView “Anjing”



Gambar 2 hasil Tab Layout dan RecyclerView pada halaman daftar anjing

Gambar di atas menampilkan hasil dari emulator setelah aplikasi dijalankan. Untuk item anjing terdapat 10 item pada aplikasi, sehingga pengguna dapat melakukan scrolling ke bawah karena menggunakan fungsi **RecyclerView**. Tata letak setiap item tersusun rapi dan teratur, tidak ada yang terlihat rusak atau terganggu dalam penampilannya. Gambar diatas merupakan susunan menggunakan sistem **List**. Gambar kiri merupakan tampilan awal, gambar kanan merupakan tampilan setelah scroll. Terlihat ketika pengguna menuju ke tab ”Anjing” maka untuk logo anjing menjadi transparan, namun untuk logo anjing menjadi ada logonya seperti di gambar. Begitu juga sebaliknya. Emulator ini sudah menerapkan untuk menambahkan logo di setiap tab.

KESIMPULAN

Dalam praktikum PAB ke-7 ini, saya berhasil mengimplementasikan **Tab Layout** yang berisi **RecyclerView** dalam aplikasi Android Studio. Dengan menggunakan Tab Layout, saya dapat dengan mudah berpindah antara fragmen yang berbeda, sementara RecyclerView memungkinkan tampilan yang efisien dan hemat memori untuk menampilkan daftar nama kucing dan anjing. Melalui langkah-langkah yang terperinci, mulai dari pembuatan layout XML, class baru, hingga menampilkan data pada RecyclerView, saya berhasil menghasilkan aplikasi yang responsif dan terstruktur dengan baik. Kesimpulannya, praktikum ini memberikan pemahaman yang baik tentang penggunaan Tab Layout dan RecyclerView dalam pengembangan aplikasi Android, serta kemampuan untuk mengatur fragmen dan menampilkan data dengan efisien. Selain itu, Saya juga berhasil menambahkan elemen visual yang menarik dengan mengganti logo pada setiap tab sesuai dengan konten yang ditampilkan, menunjukkan pemahaman yang baik tentang pengaturan UI dalam aplikasi Android.