LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Putra Whyra Pratama S. **NIM. 2310817210029**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2024

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 3

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Putra Whyra Pratama S.
NIM : 2310817210029

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar NIM. 2210817210026	Muti`a Maulida S.Kom M.T.I NIP. 19881027 201903 20 13

DAFTAR ISI

DAFTAR GAMBAR

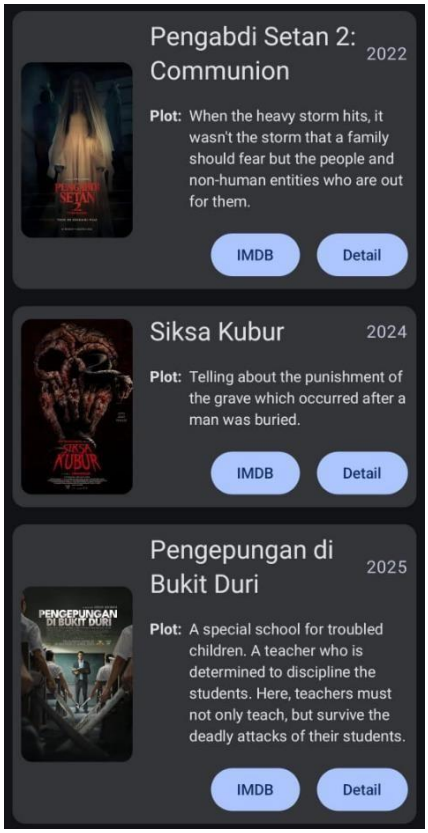
DAFTAR TABEL

SOAL 1

Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
   1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
   2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4.
      Terdapat 2 button dalam list, dengan fungsi berikut:
   a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
   b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
   5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
   6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi
      responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
   7. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
   8. Aplikasi berbasis XML harus menggunakan ViewBinding

UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 1. Contoh UI List

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:

Gambar 2. Contoh UI Detail

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code MainActivity.kt

```
1   package com.example.wuwalist
2
3   import android.os.Bundle
4   import android.util.Log
5   import androidx.activity.enableEdgeToEdge
6   import androidx.appcompat.app.AppCompatActivity
7   import androidx.core.view.ViewCompat
8   import androidx.core.view.WindowInsetsCompat
9
10  class MainActivity : AppCompatActivity() {
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14
15          val fragmentManager = supportFragmentManager
16          val homeFragment = HomeFragment()
17          val fragment =
    fragmentManager.findFragmentByTag(HomeFragment::class.java.s
    impleName)
18          if (fragment !is HomeFragment) {
19              Log.d("MyFlexibleFragment", "Fragment Name :" +
    HomeFragment::class.java.simpleName)
20              fragmentManager
21                  .beginTransaction()
22                  .add(R.id.frame_container, homeFragment,
    HomeFragment::class.java.simpleName)
23                  .commit()
24          }
25      }
26  }
```

2. Character.kt

Tabel 2. Source Code Character.kt

```
1   package com.example.wuwalist
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5
6
7   @Parcelize
8   data class Character(
9       val name: String,
10      val link: String,
11      val description: String,
12      val photo: Int
```

```
13  ):Parcelable
```

3. ListCharacterAdapter.kt

Tabel 3. Source Code ListCharacterAdapter.kt

```
1   package com.example.wuwalist
2
3   import android.view.LayoutInflater
4   import android.view.View
5   import android.view.ViewGroup
6   import android.widget.Button
7   import android.widget.ImageView
8   import android.widget.TextView
9   import androidx.recyclerview.widget.RecyclerView
10
11  class ListCharacterAdapter(
12      private val listCharacter: ArrayList<Character>,
13      private val onWikiClick: (String) -> Unit,
14      private val onDetailClick: (String, Int, String) ->
    Unit)
15      :
    RecyclerView.Adapter<ListCharacterAdapter.ListViewHolde
    r>() {
16
17      class ListViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
18          val imgPhoto: ImageView =
    itemView.findViewById(R.id.img_character)
19          val tvName: TextView =
    itemView.findViewById(R.id.tv_character)
20          val tvDeskripsi: TextView =
    itemView.findViewById(R.id.tv_deskripsi)
21          val tvProfile: TextView =
    itemView.findViewById(R.id.tv_deskripsi)
22          val btnWiki: Button =
    itemView.findViewById(R.id.btn_link)
23          val btnDetail: Button =
    itemView.findViewById(R.id.btn_detail)
24      }
25
26      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ListViewHolder {
27          val view: View =
    LayoutInflater.from(parent.context).inflate(R.layout.it
    em_character, parent, false)
28          return ListViewHolder(view)
```

```
29      }
30
31      override fun getItemCount(): Int =
        listCharacter.size
32
33      override fun onBindViewHolder(holder:
        ListViewHolder, position: Int) {
34          val (name, link, description, photo) =
        listCharacter[position]
35          holder.tvName.text = name
36          holder.imgPhoto.setImageResource(photo)
37          holder.tvDeskripsi.text = description
38          holder.btnWiki.setOnClickListener {
        onWikiClick(link) }
39          holder.btnDetail.setOnClickListener {
        onDetailClick(name, photo, description) }
40      }
41  }
```

4. HomeFragment.kt

Tabel 4. Source Code HomeFragment.kt

```
1   package com.example.wuwalist
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import androidx.fragment.app.Fragment
7   import androidx.recyclerview.widget.LinearLayoutManager
8   import android.view.LayoutInflater
9   import android.view.View
10  import android.view.ViewGroup
11  import android.widget.Button
12  import android.widget.ImageView
13  import android.widget.TextView
14  import androidx.recyclerview.widget.RecyclerView
15  import com.example.wuwalist.databinding.FragmentHomeBinding
16
17  class HomeFragment : Fragment() {
18
19      private var _binding: FragmentHomeBinding? = null
20      private val binding get() = _binding!!
21
22      private lateinit var characterAdapter:
        ListCharacterAdapter
23      private val list = ArrayList<Character>()
24
25
26      override fun onCreateView(
```

```
27          inflater: LayoutInflater, container: ViewGroup?,
28          savedInstanceState: Bundle?
29      ): View {
30          _binding = FragmentHomeBinding.inflate(inflater,
    container, false)
31
32          list.clear()
33          list.addAll(getListCharacter())
34          setupRecyclerView()
35
36          return binding.root
37
38      }
39
40      private fun setupRecyclerView() {
41          characterAdapter = ListCharacterAdapter(
42              list,
43              onWikiClick = { link ->
44                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(link))
45                  startActivity(intent)
46              },
47              onDetailClick = { name, photo, description ->
48                  val detailFragment = DetailFragment().apply
    {
49                      arguments = Bundle().apply {
50                          putString("EXTRA_NAME", name)
51                          putInt("EXTRA_PHOTO", photo)
52                          putString("EXTRA_DESCRIPTION",
    description)
53                      }
54                  }
55
56                  parentFragmentManager.beginTransaction()
57                      .replace(R.id.frame_container,
    detailFragment)
58                      .addToBackStack(null)
59                      .commit()
60              }
61          )
62
63          binding.rvCharacter.apply {
64              layoutManager = LinearLayoutManager(context)
65              adapter = characterAdapter
66              setHasFixedSize(true)
67          }
68      }
69
70      private fun getListCharacter(): ArrayList<Character> {
71          val dataName =
    resources.getStringArray(R.array.nama_character)
72
```

```
        val dataPhoto =
73  resources.obtainTypedArray(R.array.photo_character)
        val dataLink =
74  resources.getStringArray(R.array.link_character)
        val dataDescription =
75  resources.getStringArray(R.array.deskripsi_character)
76          val listCharacter = ArrayList<Character>()
77          for (i in dataName.indices) {
                val character = Character(dataName[i],
    dataLink[i], dataDescription[i], dataPhoto.getResourceId(i,
78  -1))
79              listCharacter.add(character)
80          }
81          dataPhoto.recycle()
82          return listCharacter
83      }
84
85      override fun onDestroyView() {
86          super.onDestroyView()
87          _binding = null
88      }
    }
```

5. DetailFragment.kt

Tabel 5. Source Code DetailFragment.kt

```
1   package com.example.wuwalist
2
3   import android.os.Bundle
4   import androidx.fragment.app.Fragment
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import
    com.example.wuwalist.databinding.FragmentDetailBinding
9
10  class DetailFragment : Fragment() {
11
12      private var _binding: FragmentDetailBinding? = null
13      private val binding get() = _binding!!
14
15      override fun onCreateView(
16          inflater: LayoutInflater, container: ViewGroup?,
17          savedInstanceState: Bundle?
18      ): View {
19          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
20
21          val name = arguments?.getString("EXTRA_NAME")
22          val photo = arguments?.getInt("EXTRA_PHOTO")
```

```
23          val description =
    arguments?.getString("EXTRA_DESCRIPTION")
24
25
26          binding.tvCharacter.text = name
27          binding.tvDeskripsi.text = description
28          photo?.let {
29              binding.imgCharacter.setImageResource(it)
30          }
31
32          return binding.root
33      }
34
35      override fun onDestroyView() {
36          super.onDestroyView()
37          _binding = null
38      }
39  }
```

6. activity_main.xml

Tabel 6. Source Code activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MainActivity"
8       android:id="@+id/frame_container">
9   </FrameLayout>
```

7. fragment_home.xml

Tabel 7. Source Code fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="wrap_content"
5       android:layout_height="wrap_content"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:orientation="horizontal"
8       tools:context=".HomeFragment">
9
10      <androidx.recyclerview.widget.RecyclerView
11          android:id="@+id/rv_character"
12          android:layout_width="match_parent"
```

```
13          android:layout_height="match_parent"
14          app:layout_constraintBottom_toBottomOf="parent"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18  </androidx.constraintlayout.widget.ConstraintLayout>
```

8. fragment_detail.xml

Tabel 8. Source Code fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".DetailFragment">
8
9       <ImageView
10          android:id="@+id/img_character"
11          android:layout_width="190dp"
12          android:layout_height="261dp"
13          android:layout_marginTop="16dp"
14          android:src="@drawable/card_carlotta"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18
19      <TextView
20          android:id="@+id/tv_character"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:layout_marginTop="19dp"
24          android:text="TextView"
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27
    app:layout_constraintTop_toBottomOf="@+id/img_character" />
28
29      <TextView
30          android:id="@+id/tv_profile"
31          android:layout_width="wrap_content"
32          android:layout_height="wrap_content"
33          android:layout_marginStart="10dp"
34          android:gravity="start"
35          android:text="Profile"
36          android:textStyle="bold"
37          app:layout_constraintStart_toStartOf="parent"
38
```

```
app:layout_constraintTop_toBottomOf="@+id/tv_character" />

    <TextView
        android:id="@+id/tv_deskripsi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="20dp"
        android:text="TextView"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tv_profile" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

9. item_character.xml

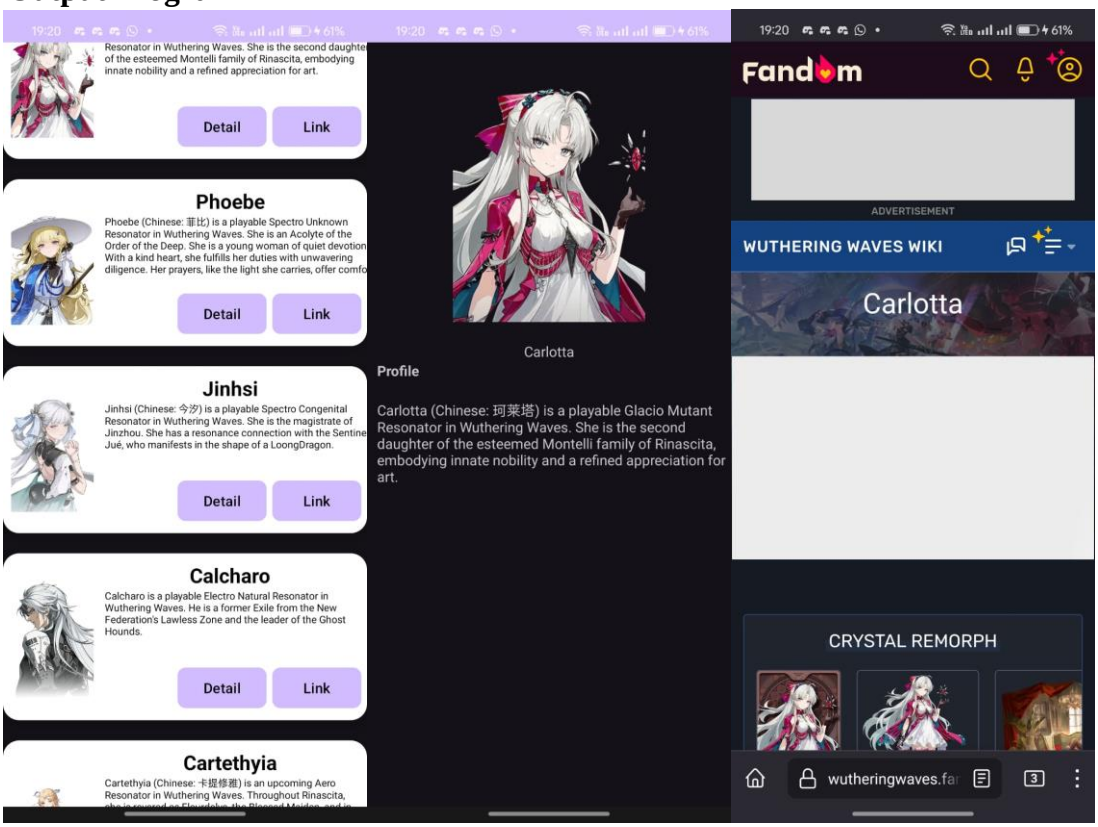
Tabel 9. Source Code item_character.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="10dp"
        app:cardBackgroundColor="#FFFFFF"
        app:cardCornerRadius="16dp"
        app:cardElevation="16dp"
        app:cardPreventCornerOverlap="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp">

```

```
32              <TextView
33                  android:id="@+id/tv_character"
34                  android:layout_width="wrap_content"
35                  android:layout_height="wrap_content"
36                  android:text="Carlotta"
37                  android:textColor="#000000"
38                  android:textSize="20sp"
39                  android:textStyle="bold"
40                  app:layout_constraintEnd_toEndOf="parent"
41
    app:layout_constraintStart_toEndOf="@+id/img_character"
42                  app:layout_constraintTop_toTopOf="parent" />
43
44              <ImageView
45                  android:id="@+id/img_character"
46                  android:layout_width="90dp"
47                  android:layout_height="143dp"
48
    app:layout_constraintBottom_toBottomOf="parent"
49
    app:layout_constraintEnd_toStartOf="@+id/tv_deskripsi"
50                  app:layout_constraintHorizontal_bias="0.4"
51
    app:layout_constraintStart_toStartOf="parent"
52                  app:layout_constraintTop_toTopOf="parent"
53                  app:layout_constraintVertical_bias="0.526"
54                  app:srcCompat="@drawable/card_carlotta" />
55
56              <TextView
57                  android:id="@+id/tv_deskripsi"
58                  android:layout_width="268dp"
59                  android:layout_height="62dp"
60                  android:layout_marginStart="11dp"
61                  android:text="Carlotta (Chinese: 珂莱塔) is a
    playable Glacio Mutant Resonator in Wuthering Waves. She is
    the second daughter of the esteemed Montelli family of
    Rinascita, embodying innate nobility and a refined
    appreciation for art."
62                  android:textColor="#000000"
63                  android:textSize="10sp"
64                  app:layout_constraintEnd_toEndOf="parent"
65
    app:layout_constraintStart_toEndOf="@+id/img_character"
66
    app:layout_constraintTop_toBottomOf="@+id/tv_character" />
67
68              <Button
69                  android:id="@+id/btn_detail"
70                  android:layout_width="wrap_content"
71                  android:layout_height="wrap_content"
72                  android:layout_marginStart="83dp"
73
```

```
                android:text="Detail"
                android:textColor="#000000"
                app:cornerRadius="8dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintStart_toEndOf="@+id/img_character"

app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
                app:layout_constraintVertical_bias="1.0" />

            <Button
                android:id="@+id/btn_link"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="20dp"
                android:layout_marginTop="12dp"
                android:layout_marginEnd="20dp"
                android:text="Link"
                android:textColor="#000000"
                app:cornerRadius="8dp"

app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@+id/btn_detail"

app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
                app:layout_constraintVertical_bias="0.0" />
        </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program



Gambar 3. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt:

- **Pada baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan, termasuk komponen dasar Android seperti Bundle, AppCompatActivity, dan utilitas untuk mengelola fragment. Impor ini memungkinkan aplikasi untuk menggunakan fitur-fitur Android modern seperti Fragment Manager dan logging.
- **Pada baris 11**, didefinisikan kelas MainActivity yang mewarisi AppCompatActivity, yang merupakan kelas dasar untuk aktivitas yang mendukung fitur-fitur kompatibilitas Android.
- **Pada baris 12 hingga 14**, terdapat method onCreate() yang merupakan entry point utama aktivitas. Di sini, layout activity_main.xml diinisialisasi menggunakan setContentView() untuk menampilkan antarmuka pengguna.
- **Pada baris 16 hingga 17**, dilakukan inisialisasi FragmentManager dan pembuatan instance HomeFragment. FragmentManager digunakan untuk mengelola fragment-fragment dalam aplikasi.
- **Pada baris 18**, dilakukan pengecekan apakah HomeFragment sudah ada dalam container menggunakan findFragmentByTag(). Ini mencegah duplikasi fragment saat aktivitas di-recreate, misalnya saat rotasi layar.
- **Pada baris 19 hingga 24**, terdapat blok kondisional yang hanya dijalankan jika HomeFragment belum ada. Di dalamnya, log debugging ditambahkan dan transaction fragment dilakukan untuk menambahkan HomeFragment ke container dengan ID frame_container.

2. Character.kt:

- **Pada baris 1 hingga 4**, dilakukan impor library yang diperlukan untuk membuat kelas data yang dapat di-parcel, yaitu Parcelable dan anotasi Parcelize.
- **Pada baris 6 hingga 12**, didefinisikan data class Character dengan anotasi @Parcelize yang mengimplementasikan interface Parcelable. Kelas ini memiliki empat properti: name (nama karakter), link (URL wiki), description (deskripsi karakter), dan photo (ID resource gambar).
- Penggunaan anotasi @Parcelize **pada baris 6** memungkinkan Kotlin untuk secara otomatis menghasilkan implementasi Parcelable tanpa perlu menulis kode boilerplate, sehingga memudahkan pengiriman objek Character antar komponen Android.

3. **ListCharacterAdapter.kt:**

- **Pada baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan untuk membuat adapter RecyclerView, termasuk komponen View, ViewGroup, dan widget-widget Android seperti Button, ImageView, dan TextView.
- **Pada baris 11 hingga 15**, didefinisikan kelas ListCharacterAdapter yang mewarisi RecyclerView.Adapter dengan tipe parameter ListViewHolder. Adapter ini menerima tiga parameter: listCharacter (daftar karakter), onWikiClick (callback untuk klik tombol wiki), dan onDetailClick (callback untuk klik tombol detail).
- **Pada baris 17 hingga 24**, didefinisikan inner class ListViewHolder yang mewarisi RecyclerView.ViewHolder. Kelas ini menginisialisasi referensi ke semua view dalam layout item_character menggunakan findViewById().
- **Pada baris 26 hingga 29**, diimplementasikan method onCreateViewHolder() yang meng-inflate layout item_character.xml dan mengembalikan instance ListViewHolder baru.
- **Pada baris 31**, diimplementasikan method getItemCount() yang mengembalikan jumlah item dalam listCharacter.
- **Pada baris 33 hingga 40**, diimplementasikan method onBindViewHolder() yang mengisi data ke view-view dalam ViewHolder. Pada baris 34, digunakan destructuring declaration untuk mengekstrak properti dari objek Character. **Pada baris 35-37**, data diset ke view yang sesuai. **Pada baris 38-39**, listener dikonfigurasi untuk tombol Wiki dan Detail.

4. **HomeFragment.kt:**

- **Pada baris 1 hingga 14**, dilakukan impor berbagai library yang diperlukan, termasuk Intent, Uri, Fragment, RecyclerView, dan komponen View Binding.
- **Pada baris 16 hingga 23**, didefinisikan kelas HomeFragment yang mewarisi Fragment. Di dalamnya, terdapat deklarasi variabel binding dengan pattern yang aman untuk memory leak, serta variabel adapter dan list untuk menyimpan data karakter.
- **Pada baris 25 hingga 36**, diimplementasikan method onCreateView() yang meng-inflate layout fragment_home.xml menggunakan View Binding. **Pada baris 31-32**, list karakter diisi dan RecyclerView diatur.
- **Pada baris 38 hingga 65**, didefinisikan method setupRecyclerView() yang menginisialisasi adapter dengan dua callback:
    - **Pada baris 41-44**, callback onWikiClick membuka browser dengan URL karakter.
    - **Pada baris 45-59**, callback onDetailClick membuat instance DetailFragment, mengisi arguments-nya dengan data karakter, dan melakukan transaction fragment untuk menampilkan detail.
    - **Pada baris 61-64**, RecyclerView dikonfigurasi dengan LinearLayoutManager dan adapter yang telah dibuat.
- **Pada baris 67 hingga 80**, didefinisikan method getListCharacter() yang mengambil data karakter dari resources. **Pada baris 68-71**, array data diambil dari resources. **Pada baris 72-76**, objek Character dibuat untuk setiap set data dan ditambahkan ke list. **Pada baris 77**, TypedArray di-recycle untuk membebaskan resources.

- **Pada baris 82 hingga 85**, diimplementasikan method onDestroyView() yang membersihkan referensi binding untuk mencegah memory leak.

5. **DetailFragment.kt:**

- **Pada baris 1 hingga 7**, dilakukan impor library yang diperlukan untuk membuat fragment dan menggunakan View Binding.
- **Pada baris 9 hingga 13**, didefinisikan kelas DetailFragment yang mewarisi Fragment. Di dalamnya, terdapat deklarasi variabel binding dengan pattern yang aman untuk memory leak.
- **Pada baris 15 hingga 32**, diimplementasikan method onCreateView() yang meng-inflate layout fragment_detail.xml menggunakan View Binding. **Pada baris 20-22**, data karakter diambil dari arguments. **Pada baris 24-28**, data tersebut ditampilkan pada view-view yang sesuai.
- **Pada baris 34 hingga 37**, diimplementasikan method onDestroyView() yang membersihkan referensi binding untuk mencegah memory leak.

6. **activity_main.xml:**

- **Pada baris 1 hingga 9**, didefinisikan layout utama aplikasi menggunakan FrameLayout. Layout ini diberi ID frame_container **pada baris 8**, yang digunakan sebagai container untuk fragment-fragment dalam aplikasi.

7. **fragment_home.xml:**

- **Pada baris 1 hingga 8**, didefinisikan layout untuk HomeFragment menggunakan ConstraintLayout. Layout ini dikonfigurasi dengan lebar dan tinggi wrap_content, yang mungkin perlu diubah menjadi match_parent untuk mengisi seluruh layar.
- **Pada baris 10 hingga 16**, didefinisikan RecyclerView dengan ID rv_character yang akan menampilkan daftar karakter. RecyclerView ini dikonfigurasi untuk mengisi seluruh parent layout menggunakan constraint.

8. **fragment_detail.xml:**

- **Pada baris 1 hingga 7**, didefinisikan layout untuk DetailFragment menggunakan ConstraintLayout dengan lebar dan tinggi match_parent.
- **Pada baris 9 hingga 16**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di tengah atas layout.
- **Pada baris 18 hingga 25**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini diposisikan di bawah gambar dan di tengah horizontal.
- **Pada baris 27 hingga 35**, didefinisikan TextView dengan ID tv_profile sebagai label untuk bagian profil. TextView ini diposisikan di bawah nama dengan alignment kiri dan style bold.
- **Pada baris 37 hingga 44**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini diposisikan di bawah label profil dengan alignment kiri.

9. **item_character.xml:**

- **Pada baris 1 hingga 6**, didefinisikan layout root untuk item karakter menggunakan ConstraintLayout dengan lebar match_parent dan tinggi wrap_content.
- **Pada baris 8 hingga 24**, didefinisikan CardView dengan ID card_view yang memberikan efek card dengan sudut melengkung dan bayangan. CardView ini dikonfigurasi dengan berbagai properti seperti margin, warna latar, radius sudut, dan elevasi.
- **Pada baris 26 hingga 30**, didefinisikan ConstraintLayout di dalam CardView yang akan menampung semua elemen UI item karakter.
- **Pada baris 32 hingga 42**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini dikonfigurasi dengan style bold, ukuran 20sp, dan warna teks hitam.
- **Pada baris 44 hingga 55**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di sebelah kiri layout.
- **Pada baris 57 hingga 69**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini dikonfigurasi dengan ukuran 10sp dan warna teks hitam.
- **Pada baris 71 hingga 82**, didefinisikan Button dengan ID btn_detail untuk melihat detail karakter. Button ini dikonfigurasi dengan sudut melengkung dan warna teks hitam.
- **Pada baris 84 hingga 96**, didefinisikan Button dengan ID btn_link untuk membuka link karakter di browser. Button ini juga dikonfigurasi dengan sudut melengkung dan warna teks hitam.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/PutraWhyra789/praktikum_pemrograman_mobile/tree/92e8d5649215e4c234002dec7fadbbacf53afd8d/Module%202

SOAL 2

2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

A. Jawaban

RecyclerView memang sering dianggap memiliki kode yang panjang dan cenderung repetitif atau boilerplate, apalagi jika dibandingkan dengan komponen seperti LazyColumn di Jetpack Compose yang jauh lebih ringkas dan modern. Namun, dalam konteks aplikasi berbasis XML seperti proyek saya, penggunaan RecyclerView masih menjadi pilihan yang relevan dan masuk akal.

Pertama, RecyclerView sudah lama menjadi standar dalam pengembangan UI berbasis daftar di Android. Komponen ini sudah sangat stabil, banyak didukung oleh pustaka eksternal, dan terdokumentasi dengan sangat baik. Hal ini membuatnya tetap menjadi pilihan utama, terutama bagi pengembang yang masih menggunakan pendekatan View System tradisional. Dalam proyek saya, misalnya, seluruh tampilan masih berbasis XML, sehingga lebih masuk akal untuk menggunakan RecyclerView dibanding harus mengadopsi Jetpack Compose hanya untuk mengganti daftar tampilan.

Selain itu, RecyclerView juga memberi kontrol yang lebih lengkap terhadap tampilan item dan bagaimana daftar tersebut ditampilkan atau diatur. Fitur-fitur seperti animasi, pengaturan cache, serta berbagai jenis layout manager menjadi nilai lebih yang membuatnya fleksibel untuk berbagai kebutuhan UI. Meskipun lebih verbose, struktur RecyclerView memberikan kejelasan dalam pemisahan logika tampilan dan data, khususnya dalam skala proyek yang lebih besar.

Jadi, meskipun secara sintaksis LazyColumn terlihat lebih sederhana dan modern, RecyclerView masih memiliki tempat penting, terutama ketika kita bekerja dalam lingkungan XML atau saat memerlukan fleksibilitas dan kontrol penuh terhadap tampilan daftar.