LAPORAN RESMI

MODUL II

LAYOUT, WIDGET VIEW DAN RECYCLER VIEW

PEMROGRAMAN BERGERAK

NAMA	**:**	**SEVIN DIAS ANDIKA**
N.R.P	**:**	**210441100105**
DOSEN	**:**	**Ir. ACH. DAFID, S.T., M.T.**
ASISTEN	**:**	**DAVID NASRULLOH**
TGL PRAKTIKUM	**:**	**31 MARET 2023**

Disetujui : .. APRIL 2022
Asisten

DAVID NASRULLOH
190441100060

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pemrograman bergerak adalah salah satu jenis pemrograman yang berkaitan dengan pengembangan aplikasi dan perangkat lunak yang dapat dijalankan pada perangkat mobile seperti smartphone dan tablet. Seiring dengan perkembangan teknologi yang semakin pesat, penggunaan perangkat mobile semakin meningkat drastis, sehingga memunculkan kebutuhan yang besar untuk aplikasi mobile yang lebih banyak dan bervariasi. Oleh karena itu, pemrograman bergerak menjadi salah satu bidang yang sangat penting untuk dipelajari. Di dalam bidang pemrograman bergerak, kita akan mempelajari bahasa pemrograman Kotlin yang dirancang untuk membangun aplikasi mobile pada platform Android dan iOS.

Sebuah view adalah obyek yang menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung. Sedangkan viewgroup adalah sebuah obyek yang mewadahi obyek-obyek view dan viewgroup itu sendiri sehingga membentuk satu kesatuan tampilan aplikasi yang utuh. Layout ini akan menempatkan komponen-komponen di dalamnya secara horizontal atau vertikal. Linearlayout memiliki atribut weight untuk masing-masing child view yang berguna untuk menentukan porsi ukuran view dalam sebuah ruang (space) yang tersedia. RecyclerView adalah tampilan yang menggunakan arsitektur yang disederhanakan dengan UI controller, ViewModel, dan LiveData

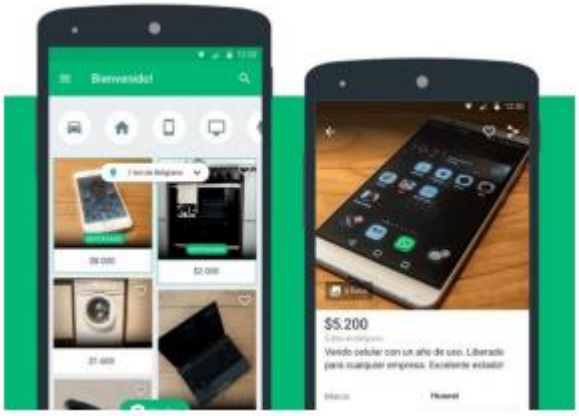
1.2 Tujuan

- ➢ Membuat Layout dengan Linear Layout dan Constraint Layout.
- ➢ Mampu menggunakan Widget View (masukan) untuk membuat aplikasi sederhana.
- ➢ Merepresentasikan data dengan menggunakan komponen recylerview.

BAB II
DASAR TEORI

2.1 Layout

Pada modul ini, kita akan mempelajari komponen View dan ViewGroup. Kedua komponen ini dapat berkolaborasi sehingga membentuk antar muka dengan contoh seperti pada gambar di bawah ini:



Pada dasarnya semua elemen antar pengguna di aplikasi Android dibangun menggunakan dua buah komponen inti, yaitu view dan viewgroup.

Sebuah view adalah obyek yang menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung.

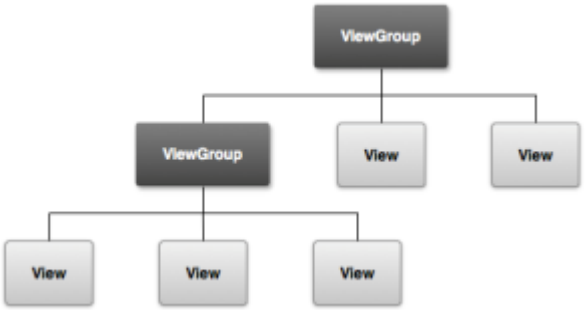
Contoh komponen turunan dari view seperti :

- TextView, komponen yang berguna untuk menampilkan teks ke layar.
- Button, komponen yang membuat pengguna dapat berinteraksi dengan cara ditekan untuk melakukan sesuatu.
- ImageView, Komponen untuk menampilkan gambar.
- ListView, komponen untuk menampilkan informasi dalam bentuk list.
- GridView, komponen untuk menampilkan informasi dalam bentuk grid.
- RadioButton, komponen yang memungkinkan pengguna dapat memilih satu pilihan dari berbagai pilihan yang disediakan.

- Checkbox, komponen yang memungkinkan pengguna dapat memilih lebih dari satu dari pilihan yang ada.

Sedangkan viewgroup adalah sebuah obyek yang mewadahi obyek-obyek view dan viewgroup itu sendiri sehingga membentuk satu kesatuan tampilan aplikasi yang utuh. Contoh komponen viewgroup adalah:

- LinearLayout
- FrameLayout
- RelativeLayout
- TableLayout



Jika diterjemahkan di dalam sebuah viewgroup akan ditampung dua buah komponen view dan satu komponen viewgroup yang terdiri dari 3 buah komponen view. Salah satu contoh dari tampilan dalam file layout xml untuk merepresentasikan kolaborasi view dan viewgroup seperti ini :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am a TextView" />
    <Button android:id="@+id/button"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am a Button" />
</LinearLayout>
```

Obyek turunan viewgroup LinearLayout menjadi kontainer untuk obyek turunan view, button, dan textview. Beberapa komponen viewgroup seperti linearlayout, relativelayout, framelayout, dan tablelayout merupakan komponen yang paling banyak digunakan untuk menjadi parent/root dari komponen-komponen view.

Berikut adalah definisi singkat dan inti dari komponen-komponen di atas terhadap penempatan komponen view (child) di dalamnya. Kita akan membahas Linear Layout dan Constrain Layout.

LinearLayout

Layout ini akan menempatkan komponen-komponen di dalamnya secara horizontal atau vertikal. Linearlayout memiliki atribut weight untuk masing-masing child view yang berguna untuk menentukan porsi ukuran view dalam sebuah ruang (space) yang tersedia.

Constrain Layout. Apa itu ConstraintLayout? (https://blog.dicoding.com/kenallebihdekat-dengan-constraintlayout/)

ConstraintLayout merupakan salah satu komponen ViewGroup yang dapat kita gunakan untuk menyusun tampilan aplikasi yang kompleks tanpa adanya nested

layout. ConstraintLayout tersedia dengan dukungan kompatibilitas mulai dari Android 2.3 (API Level 9) sampai dengan yang terbaru.

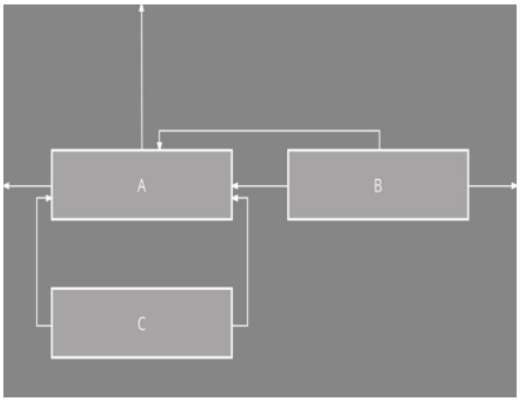
ConstraintLayout memiliki kesamaan dengan RelativeLayout. Dalam penggunaan semua view yang berada di dalamnya disusun berhubungan antara parent dan view lainnya. Tapi ConstraintLayout lebih fleksibel dari RelativeLayout dan mudah digunakan dengan dukungan Layout Editor pada Android Studio.

Let’s say kita menambah view baru ke dalam ConstraintLayout. Kita gunakan drag and drop di Layout Editor yang berada pada tab Design atau dengan menambahnya secara manual melalui tab Text. Kita perlu menentukan posisi dari view atau bagaimana agar view tersebut terhubung dengan parent layout atau view lainnya.

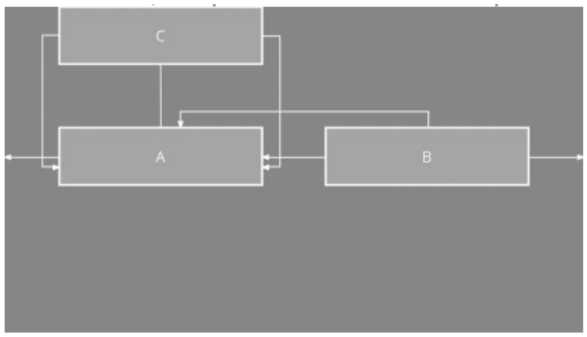
Kenapa gerangan? Karena setelah ditambahkan, view tersebut tidak memiliki constraint yang menghubungkannya dengan parent layout atau view lainnya. Sehingga ketika dijalankan, posisi dari view tersebut akan berada di bagian atas sebelah kiri.

Berbeda ceritanya dengan RelativeLayout. Saat kita ingin menentukan posisi atau menghubungkan dua buah view, kita bisa menggunakan attribute seperti layout_below atau layout_above. Nah untuk ConstraintLayout kita akan menggunakan constraint sebagai dasar dalam menentukan posisi agar sebuah view dapat terhubung dengan view lainnya sesuai harapan kita.

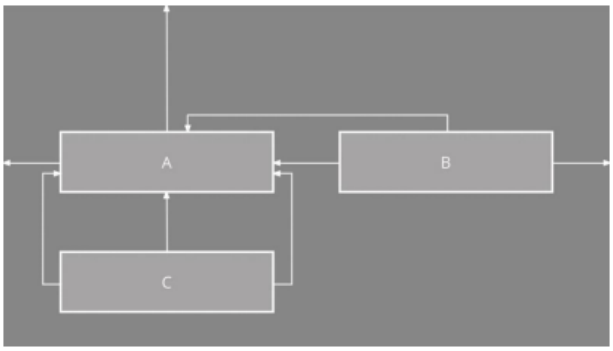
Setiap view setidaknya memiliki satu vertikal dan horizontal constraint. Misal kita memiliki sebuah layout dengan tampilan pada Layout Editor seperti berikut:



Susunan tampilan di atas akan terlihat normal. Tidak ada yang salah di Layout Editor. Tapi jika kita perhatikan seksama, view C diatas hanya memiliki horizontal constraint yang diatur sejajar dengan view A. Sehingga ketika jika kita coba menjalankannya, sama seperti yang disebutkan diatas, maka posisi dari view C akan berada di posisi atas seperti berikut:



Berbeda jika kita menambahkan vertikal constraint pada view C yang diatur terikat dengan view A seperti berikut:



Ketika dijalankan, apa yang terjadi? Yang tampil akan sesuai dengan apa yang terlihat di Layout Editor.

2.2 Komponen Widget View

Paket widget pada dasarnya merupakan visualisasi dari elemen user interface (UI) yang digunakan pada layar aplikasi Android di mana kita dapat merancang sendiri sesuai kebutuhan.

Widget di dalam Android ditampilkan dengan konsep View. Di mana aplikasi Android pada umumnya menggunakan widget sebagai Layout XML. Untuk mengimplementasikan widget, selain file kotlin kita juga membutuhkan tambahan

dua file. Berikut ini adalah file-file yang umumnya kita butuhkan apabila kita membuat widget:

1. File Kotlin. Berupa file yang mengimplementasikan aksi dari widget. Jika kita mendefinisikan suatu widget beserta posisinya di layar yang didefinisikan dari file XML, kita harus melakukan coding di file kotlin yang dapat mengambil semua nilai atribut dari file layout XML yang didefinisikan.
2. File XML. Sebuah file yang mendefinisikan komponen elemen-elemen XML yang digunakan untuk inisialisasi widget serta atribut yang mendukungnya.
3. Layout XML. File XML menggambarkan atau penambahan keterangan pada layout widget kita.

Komponen widget TextView dan Button sudah kita bahas pada modul sebelumnya. Beberapa komponen widget akan kita bahas saat ini. Widget EditText untuk menuliskan teks ke aplikasi dan akan ditangkap oleh aplikasi untuk diolah. Widget Image Button untuk membuat button yang diberi gambar. Widget Image View untuk membuat tampilan gambar. Sedangkan widget RadioButton/ RadioGroup biasanya digunakan bersama-sama.

Di dalam satu RadioGroup terdapat beberapa RadioButton. Dan di dalam satu RadioGroup user hanya dapat melakukan satu check/pemilihan RadioButton. Dan yang terakhir widget akan kita bahas CheckBox, pilihan yang dapat dipilih lebih dari satu item.

Event Handling.

Android dapat menangani event dari interaksi dengan pengguna. Saat mempertimbangkan event dalam user interface, pendekatannya adalah menangkap event dari objek View tertentu yang digunakan pengguna untuk berinteraksi. Kelas View menyediakan sarana untuk melakukannya.

Dalam berbagai kelas View yang akan digunakan untuk menyusun layout, mungkin dapat dilihat beberapa method callback publik yang tampak berguna untuk kejadian UI. Method ini dipanggil oleh framework Android ketika masing-masing

tindakan terjadi pada objek itu. Misalnya, jika View (seperti Button) disentuh, method onTouchEvent() akan dipanggil pada objek itu. Kelas View salah satunya berisi sekumpulan interface bertumpuk dengan callback yang mudah didefinisikan. Antarmuka ini, yang disebut event listener, digunakan untuk melakukan interaksi pengguna dengan UI.

Event listener

Event listener merupakan antarmuka di kelas View yang berisi method callback tunggal. Method ini akan dipanggil oleh framework Android jika View yang telah didaftarkan dengan listener dipicu oleh interaksi pengguna dengan item dalam UI.

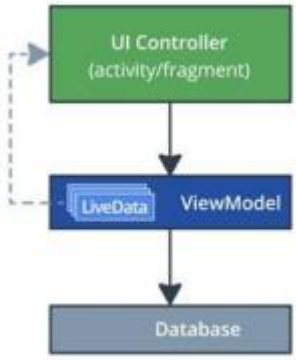
Yang juga disertakan dalam antarmuka event listener adalah method callback berikut ini:

1. Method onClick() dari View.OnClickListener. Ini dipanggil baik saat pengguna menyentuh item (jika dalam mode sentuh), maupun memfokuskan pada item dengan tombol navigasi atau trackball dan menekan tombol "enter" yang sesuai atau menekan trackball.
2. Method onLongClick() dari View.OnLongClickListener. Ini dipanggil baik saat pengguna menyentuh dan menahan item (jika dalam mode sentuh), maupun memfokuskan pada item dengan tombol navigasi atau trackball dan menekan serta menahan tombol "enter" yang sesuai atau menekan dan menahan trackball (selama satu detik).
3. Method onFocusChange() dari View.OnFocusChangeListener. Ini dipanggil saat pengguna menyusuri ke atau dari item, dengan menggunakan tombol navigasi atau trackball.
4. Method onKey() dari View.OnKeyListener. Ini dipanggil saat pengguna memfokuskan pada item dan menekan atau melepas tombol perangkat keras pada perangkat.
5. Method onTouch() dari View.OnTouchListener. Ini dipanggil saat pengguna melakukan tindakan yang digolongkan sebagai peristiwa sentuh, termasuk penekanan, pelepasan, atau isyarat perpindahan pada layar (dalam batasan item itu).

6. Method onCreateContextMenu() dari View.OnCreateContextMenuListener. Ini dipanggil saat Menu Konteks sedang dibuat (akibat "klik lama" terus-menerus).

2.3 Recycler View

RecyclerView adalah tampilan yang menggunakan arsitektur yang disederhanakan dengan UI controller, ViewModel, dan LiveData.



Menampilkan list atau grid data adalah salah satu tugas UI paling umum di Android. Daftar bervariasi dari yang sederhana hingga yang sangat kompleks. Daftar tampilan teks mungkin menampilkan data sederhana, seperti daftar belanja. Daftar yang kompleks, seperti daftar tujuan liburan yang beranotasi, dapat menunjukkan kepada pengguna banyak detail di dalam scrolling grid dengan header. Untuk mendukung semua kasus penggunaan ini, Android menyediakan widget RecyclerView.

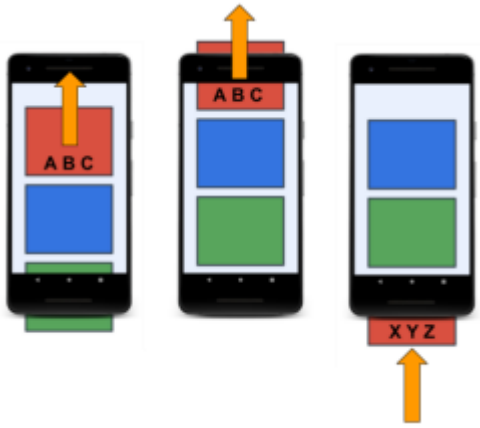
Manfaat terbesar dari RecyclerView adalah sangat efisien untuk daftar besar:

- Secara default, RecyclerView hanya berfungsi untuk memproses atau menggambar item yang saat ini terlihat di layar. Misalnya, jika list memiliki seribu elemen tetapi hanya 10 elemen yang terlihat, RecyclerView hanya berfungsi untuk menggambar 10 item di layar.

Ketika pengguna melakukan scroll, RecyclerView mengetahui item baru apa yang seharusnya ada di layar dan tidak cukup berfungsi untuk menampilkan item itu.

- Ketika suatu item scroll dari layar, tampilan item tersebut didaur ulang. Itu berarti item diisi dengan konten baru yang scroll ke layar. Perilaku RecyclerView ini menghemat banyak waktu pemrosesan dan membantu scroll list dengan lancar.
- Ketika suatu item berubah, alih-alih menggambar ulang seluruh daftar, RecyclerView dapat memperbarui satu item itu. Ini adalah keuntungan efisiensi yang sangat besar ketika menampilkan daftar item kompleks!

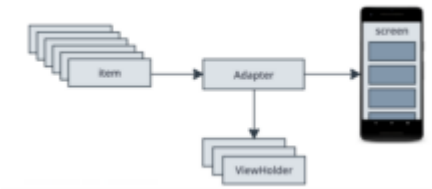
Dalam urutan yang ditunjukkan di bawah ini, kita dapat melihat bahwa satu tampilan telah diisi dengan data, ABC. Setelah itu tampilan bergulir dari layar, RecyclerView menggunakan kembali tampilan untuk data baru, XYZ.



Adapter pattern

Jika kita pernah bepergian antar negara yang menggunakan soket listrik yang berbeda, kita mungkin tahu bagaimana kita bisa mencolokkan perangkat kita ke outlet dengan menggunakan adaptor. Adaptor memungkinkan kita mengonversi satu jenis steker ke yang lain, yang benar-benar mengubah satu antarmuka menjadi yang lain. Pola adaptor dalam rekayasa perangkat lunak membantu objek bekerja dengan API lain. RecyclerView menggunakan adaptor untuk mengubah data aplikasi menjadi sesuatu yang dapat ditampilkan RecyclerView, tanpa mengubah cara aplikasi menyimpan dan memproses data. Untuk aplikasi pelacak tidur, kita membuat adaptor yang mengadaptasi data menjadi sesuatu yang RecyclerView tahu

cara menampilkannya, tanpa mengubah ViewModel. Mengimplementasikan sebuah RecyclerView



Untuk menampilkan data dalam RecyclerView, memerlukan bagian-bagian berikut:

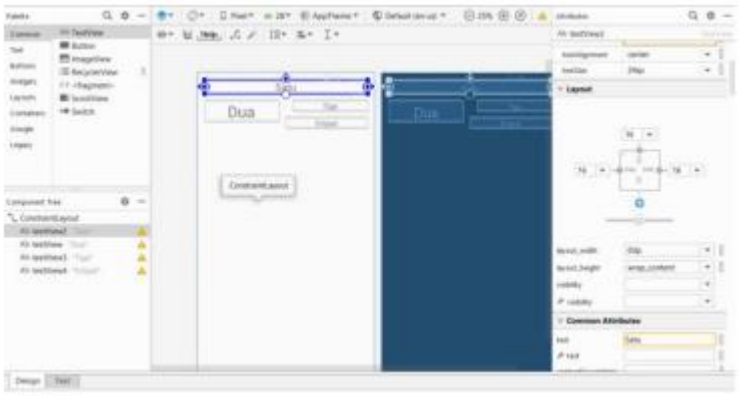
- Data untuk ditampilkan.
  Mesin virtual RecyclerView didefinisikan dalam file layout, untuk bertindak sebagai wadah untuk tampilan.
- Layout untuk satu item data.
  Jika semua item list terlihat sama, kita dapat menggunakan layout yang sama untuk semuanya, tetapi itu tidak wajib. Layout item harus dibuat secara terpisah dari layout fragmen, sehingga tampilan satu item pada satu waktu dapat dibuat dan diisi dengan data.
- Layout Manager.
  Layout Manager menangani organisasi (layout) komponen UI dalam tampilan.
- View holder.
  view holder extends kelas ViewHolder. Ini berisi informasi tampilan untuk menampilkan satu item dari layout item. Penampil tampilan juga menambahkan informasi yang digunakan RecyclerView untuk memindahkan tampilan di layar secara efisien.
- Adaptor.
  Adaptor menghubungkan data kita ke RecyclerView. Ini menyesuaikan data sehingga dapat ditampilkan di ViewHolder. RecyclerView menggunakan adaptor untuk mengetahui cara menampilkan data di layar.

BAB III

TUGAS PENDAHULUAN

3.1 SOAL

1. Buat project baru dengan desain sebagai berikut.



2. Buat project baru dengan menggunakan Linear Layout dengan minimal 3 komponen (TextView/Button) yang ditambahkan dan eksplorasilah atribut-atribut yang ada.

3.2 JAWABAN

1.
```
<?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="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="24dp"
        android:layout_marginStart="21dp"
        android:layout_marginTop="16dp"
```

```
    android:layout_marginEnd="20dp"
    android:text="Satu"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView"
    android:layout_width="124dp"
    android:layout_height="49dp"
    android:layout_marginStart="20dp"
    android:layout_marginTop="72dp"
    android:layout_marginEnd="267dp"
    android:text="Dua"
    android:textAlignment="center"
    android:textSize="34sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="131dp"
    android:layout_height="22dp"
    android:layout_marginStart="281dp"
    android:layout_marginTop="72dp"
    android:layout_marginEnd="72dp"
    android:text="Tiga"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
```

```
    tools:textAlignment="center" />

  <TextView
    android:id="@+id/textView4"
    android:layout_width="149dp"
    android:layout_height="20dp"
    android:layout_marginStart="282dp"
    android:layout_marginTop="120dp"
    android:layout_marginEnd="71dp"
    android:text="Empat"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

2.
```
<?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="match_parent"
    tools:context=".MainActivity">

    <ImageView
      android:id="@+id/imageView2"
      android:layout_width="411dp"
      android:layout_height="227dp"
      android:layout_marginTop="4dp"
      app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/_209321" />

  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="252dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.05"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/imageView2" />

  <Switch
    android:id="@+id/switch1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="264dp"
    android:layout_marginEnd="8dp"
    android:text="Switch"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.87"
    app:layout_constraintStart_toEndOf="@+id/button"
    app:layout_constraintTop_toTopOf="@+id/imageView2" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

BAB IV
IMPLEMENTASI

4.1 Source Code

1. Activity_main.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="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/recyclerview"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. List_item.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"
    android:layout_margin="8dp">
    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/title_image"
        android:layout_width="80dp"
        android:layout_height="80dp"
```

```
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:src="@drawable/a"
    app:shapeAppearanceOverlay="@style/RoundCorner"/>

<TextView
    android:id="@+id/tvHeading"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textColor="@color/black"
    android:text="jskdjlkjkjkklsakdlskalskdlaskdlsadklaskdlksdlak"
    android:textSize="16dp"
    android:textStyle="bold"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="32dp"
    android:layout_marginTop="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toEndOf="@+id/title_image"
    tools:layout_editor_absoluteY="0dp"
    app:layout_constraintTop_toTopOf="parent"/>

<View
    android:id="@+id/view"
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:layout_margin="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/underline"
    app:layout_constraintBaseline_toBottomOf="@id/title_image"
```

```
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="80dp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. News.kt

```
package com.example.recycelview

data class news(var titleImage: Int, var heading: String)
```

4. Adaptor.kt

```
package com.example.recycelview

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.imageview.ShapeableImageView

class adaptor(private val newsList: ArrayList<news>) :
    RecyclerView.Adapter<adaptor.MyViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
MyViewHolder {

        val itemView =
LayoutInflater.from(parent.context).inflate(R.layout.list_item,
            parent,false)
        return MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
```

```
        val currentItem = newsList[position]
        holder.titleImage.setImageResource(currentItem.titleImage)
        holder.tvHeading.text = currentItem.heading
    }

    override fun getItemCount(): Int {

        return newsList.size
    }

    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView){

        val titleImage : ShapeableImageView = itemView.findViewById(R.id.title_image)
        val tvHeading : TextView = itemView.findViewById(R.id.tvHeading)

    }
}
```

5. MainActivity.kt

```
package com.example.recycelview

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {

    private lateinit var newRecyclerView: RecyclerView
```

```
    private lateinit var newArrayList: ArrayList<news>
    lateinit var imageId : Array<Int>
    lateinit var heading : Array<String>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        imageId = arrayOf(
            R.drawable.a,
            R.drawable.b,
            R.drawable.c,
            R.drawable.d,
            R.drawable.e,
            R.drawable.f,
            R.drawable.g,
            R.drawable.h,
            R.drawable.i,
            R.drawable.j,
            R.drawable.c,
            R.drawable.d
        )

        heading = arrayOf(
            "Biden aims to expand vaccines for adults and children",
            "Just go my first shot, helping the world to be a safer placfe",
            "Local trains to be suspended in begal from tomorrow in view of Covid-19",
            "MHA asks states, UTs to ensure there are no fires in hospital",
            "Biden aims to expand vaccines for adults and children",
            "Just go my first shot, helping the world to be a safer placfe",
```

```
        "Local trains to be suspended in begal from tomorrow in view of Covid-19",
        "MHA asks states, UTs to ensure there are no fires in hospital",
        "Biden aims to expand vaccines for adults and children",
        "Just go my first shot, helping the world to be a safer placfe",
        "Local trains to be suspended in begal from tomorrow in view of Covid-19",
        "MHA asks states, UTs to ensure there are no fires in hospital"

    )

    newRecyclerView = findViewById(R.id.recyclerview)
    newRecyclerView.layoutManager = LinearLayoutManager(this)
    newRecyclerView.setHasFixedSize(true)

    newArrayList = arrayListOf<news>()
    getUserdata()
  }

  private fun getUserdata() {
    for(i in imageId.indices){
      val news = news(imageId[i],heading[i])
      newArrayList.add(news)
    }

    newRecyclerView.adapter = adaptor(newArrayList)
  }

}
```

4.2 Hasil


SM A115F
14.58
recycelview
Biden aims to expand vaccines for adults and children
Just go my first shot, helping the world to be a safer placfe
Local trains to be suspended in begal from tomorrow in view of Covid-19
MHA asks states, UTs to ensure there are no fires in hospital
Biden aims to expand vaccines for adults and children
Just go my first shot, helping the world to be a safer placfe
Local trains to be suspended in begal from tomorrow in view of Covid-19
MHA asks states, UTs to ensure there are no fires in hospital

BAB V
PENUTUP

5.1 Analisa

Dari hasil praktikum, Sebuah view adalah obyek yang menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung. Sedangkan viewgroup adalah sebuah obyek yang mewadahi obyek-obyek view dan viewgroup itu sendiri sehingga membentuk satu kesatuan tampilan aplikasi yang utuh. Layout ini akan menempatkan komponen-komponen di dalamnya secara horizontal atau vertikal. Linearlayout memiliki atribut weight untuk masing-masing child view yang berguna untuk menentukan porsi ukuran view dalam sebuah ruang (space) yang tersedia. RecyclerView adalah tampilan yang menggunakan arsitektur yang disederhanakan dengan UI controller, ViewModel, dan LiveData.

5.2 Kesimpulan

1. Sebuah view adalah obyek yang menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung.
2. viewgroup adalah sebuah obyek yang mewadahi obyek-obyek view dan viewgroup itu sendiri sehingga membentuk satu kesatuan tampilan aplikasi yang utuh.
3. RecyclerView adalah tampilan yang menggunakan arsitektur yang disederhanakan dengan UI controller, ViewModel, dan LiveData.