LAPORAN PRAKTIKUM MINGGU 2 APLIKASI MOBILE

" Linier, Relative dan Constraint Layout "

Disusun oleh

Nama : Ahmad Catur Yulianto
NIM/Golongan : E31191894 / C

GOLONGAN C
PROGRAM STUDI MANAJEMEN INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI JEMBER
2021

BAB I
PENDAHULUAN

1.1 Latar Belakang

Layout Editor memungkinkan Anda membuat tata letak dengan cepat dengan menyeret elemen UI ke editor desain visual, alih-alih menulis XML tata letak secara manual. Editor desain ini dapat menampilkan pratinjau tata letak Anda pada beragam versi dan perangkat Android, dan Anda dapat secara dinamis mengubah ukuran tata letak untuk memastikannya berfungsi dengan baik pada berbagai ukuran layar.Ketika kita ingin membuat aplikasi haruslah memiliki tampilan yang nyaman di lihat bagi pengguna. Layout adalah suatu tampilan tata letak di android studio untuk mengatur penempatan text/gambar yang sudah terkonsep. Untuk itu layout di sini adalah bagian terpenting untuk memperindah tampilan pada aplikasi yang kita buat nyaman di lihat bagi pengguna.

1.2 Rumusan masalah

1) Apa pengertian Layout?
2) Apa pengertian Linier Layout?
3) Apa pengertian Relative Layout?
4) Apa pengertian Constraint Layout?

1.3 Tujuan

1) Mahasiswa dapat memahami konsep dasar dari linier layout.
2) Mahasiswa dapat mengimplementasikan linier pada pemrograman berbasis android

BAB II

TEORI

2.1 Pengertian Layout

Layout atau Tata letak menentukan struktur dari antarmuka pengguna di aplikasi Anda, seperti di dalam aktivitas. Semua elemen pada tata letak dibuat menggunakan hierarki objek View dan ViewGroup. View biasanya menggambar sesuatu yang terlihat, dan pengguna dapat berinteraksi dengannya. Sedangkan ViewGroup adalah container tak terlihat yang menentukan struktur tata letak bagi View dan objek ViewGroup lainnya, seperti yang ditampilkan pada gambar 1.

Objek View yang biasanya disebut "widget" dapat berupa salah satu dari banyak subclass, seperti Button atau TextView. Objek ViewGroup yang biasanya disebut "tata letak" dapat berupa salah satu dari banyak jenis yang menyediakan berbagai struktur tata letak, seperti LinearLayout atau ConstraintLayout .

Anda dapat mendeklarasikan tata letak dengan dua cara:

a. **Deklarasikan elemen UI dalam XML**. Android menyediakan kosakata XML sederhana yang sesuai dengan class dan subclass Tampilan, seperti halnya untuk widget dan tata letak.

   Anda juga dapat menggunakan Layout Editor di Android Studio untuk membuat tata letak XML menggunakan antarmuka tarik lalu lepas.

b. **Buat instance elemen tata letak pada saat runtime**. Aplikasi Anda dapat membuat objek Tampilan dan ViewGroup (serta memanipulasi propertinya) secara terprogram.

Dengan mendeklarasikan UI pada XML, Anda dapat memisahkan presentasi aplikasi dari kode yang mengontrol perilakunya. Menggunakan file XML juga mempermudah dalam menyediakan berbagai tata letak untuk orientasi dan ukuran layar yang berbeda (yang dibahas lebih jauh di Mendukung Berbagai Ukuran Layar).

Framework Android memberi Anda fleksibilitas untuk menggunakan salah satu atau kedua metode ini untuk membuat UI aplikasi Anda. Misalnya, Anda dapat mendeklarasikan tata letak default aplikasi pada XML, kemudian memodifikasinya saat runtime.

2.2 Pengertian Linier Layout

LinearLayout adalah sekelompok tampilan yang menyejajarkan semua anak dalam satu arah, secara vertikal atau horizontal. Anda bisa menetapkan arah layout dengan atribut android:orientation.Secara gampangnya menggunakan Linear Layout dapat mengatur komponen didalamnya agar rapi sesuai dengan orientasi. Penggunaan Linear Layout biasanya pada saat ingin membuat FORM atau ROW pada RECYLERVIEW. Orientasi untuk Linear Layout sendiri ada 2 yaitu Horizontal dan Vertical.

2.3 Pengertian Relative Layout

Relative Layout merupakan layout yang berfungsi untuk mengatur tata letak komponen atau widget aplikasi android dengan cara relative (secara bebas) tidak hanya vertikal atau horisontal saja seperti pada Linear Layout. Relative Layout memungkinkan kita untuk memposisikan komponen mana saja yang kita inginkan, sehingga dianggap sebagai layout yang paling fleksibel. Karena alasan tersebut Relative layout menjadi layout yang paling banyak digunakan setelah Linear Layout di Android.

2.4 Pengertian Constraint Layout

Sebenarnya ConstraintLayout mirip dengan RelativeLayout, karena letak View bergantung pada View lain dalam satu Layout ataupun dengan Parent Layoutnya. Contohnya di RelativeLayout kita bisa meletakkan sebuah View di atas, bawah, atau samping View lain. Kita juga dapat 34 mengatur posisinya berdasarkan Parent Layout menggunakan tag seperti centerVertical, alignParentBottom, dll. tapi ConstraintLayout jauh lebih Fleksibel dan lebih mudah digunakan pada Layout Editor. Bayangkan seperti ini, pada ConstraintLayout, setiap View memiliki tali (Constraint) yang menarik tiap sisinya, yang mana tali tersebut bisa kita atur Elastisitas, Margin, dsb. Tali tersebut wajib kita “ikatkan” kepada anchor point atau suatu titik yang dapat berupa sisi dari Parent Layout, View lain, ataupun titik bantu (helper) yang bisa kita buat sendiri. Tanpa constraint tersebut, sebuah Layout secara default akan terletak di posisi [0,0] (sebelah kiri-atas), sehingga sebuah View paling tidak membutuhkan 2 Constraint untuk sisi X dan sisi Y.

BAB III

HASIL DAN PEMBAHASAN

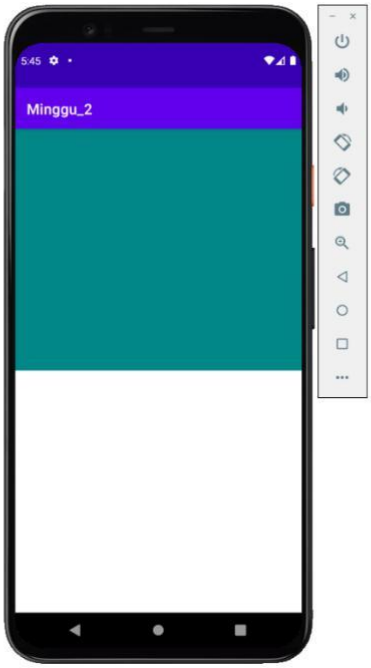
1. Program hasil praktikum 2

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:weightSum="2"
    android:background="@color/cardview_dark_background"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/teal_700"
        >
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#FFFFFF"
        >
    </LinearLayout>

</LinearLayout>
```

Pembahasan :

Pada praktikum ini belajar mengatur layout dan menambahkan child layout. Layout yang digunakan adalah Linear Layout dengan design orientation vertical atau membuat tampilan child layout yang berjejer secara vertical.

Hasil :



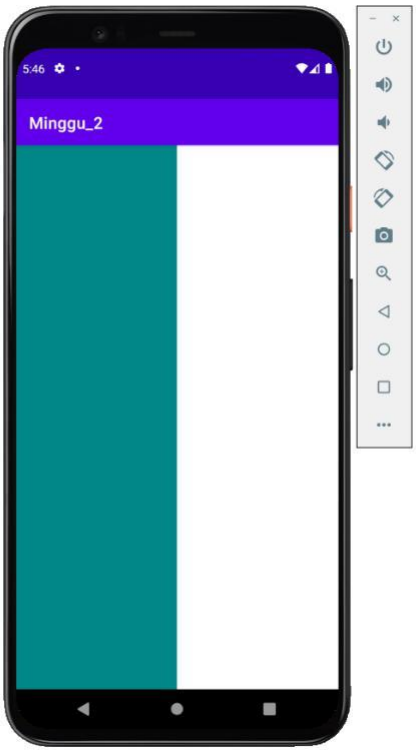
2. Mengubah orientation menjadi horizontal

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="horizontal"
    android:weightSum="2"
    android:background="@color/cardview_dark_background"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@color/teal_700"
        >
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#FFFFFF"
        >
    </LinearLayout>

</LinearLayout>
```

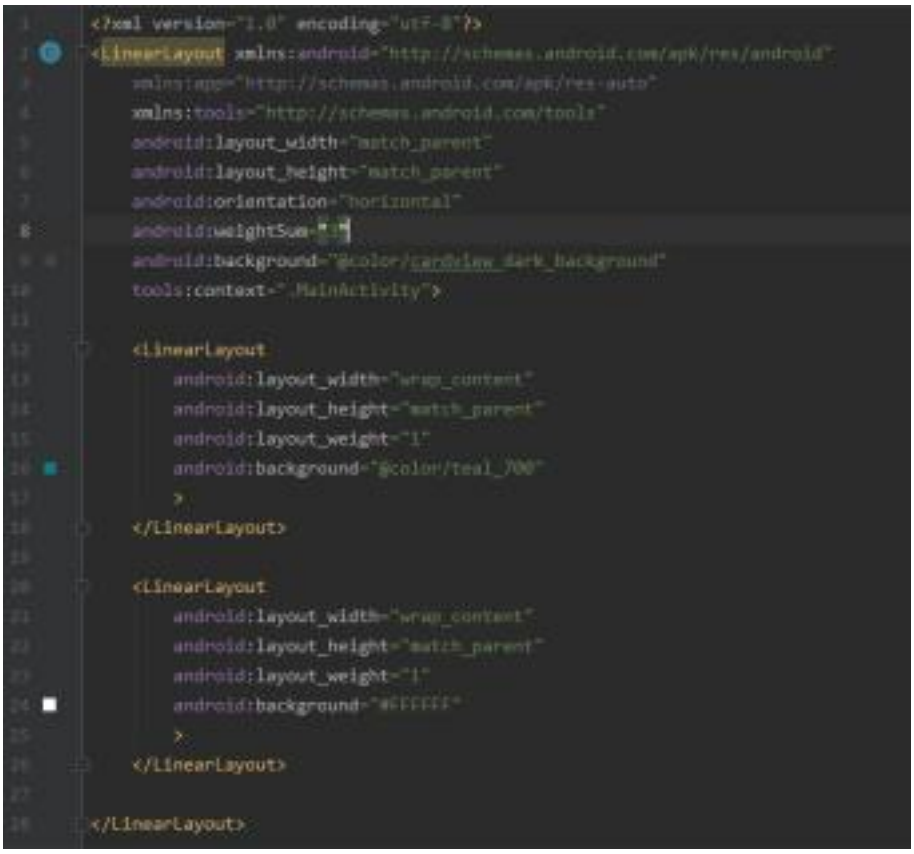
Pembahasan :

a. Mengubah orientation menjadi horizontal dengan cara mengubah kode *android:orientation="vertical"* menjadi *android:orientation="horizontal"* yang ada di dalam tag kode parent layout supaya tampilan child layout yang berjejer secara horizontal.
b. merubah properti yang ada di dalam tag kode child layout *android:layout_width="wrap_content"* bertujuan agar lebar digunakan agar lebar dari komponen, dapat mengikuti layar atau screen pada perangkat Android.
c. *android:layout_height="match_parent"* bertujuan agar tinggi, dapat mengikuti object atau ukuran penuh dari layar atau screen pada perangkan Android.
d. Pada gambar output tampilan child layout akan terbagi menjadi 2 karena pada parent layout memiliki properti *weightSum* bernilai **2**, dan pada child layout masing-masing meiliki properti *layout_weight* bernilai **1** yang akan membuat child layout terbagi dengan ukuran lebar yang sama.

Hasil :



3. Mengubah weight sum menjadi angka integer yang lain



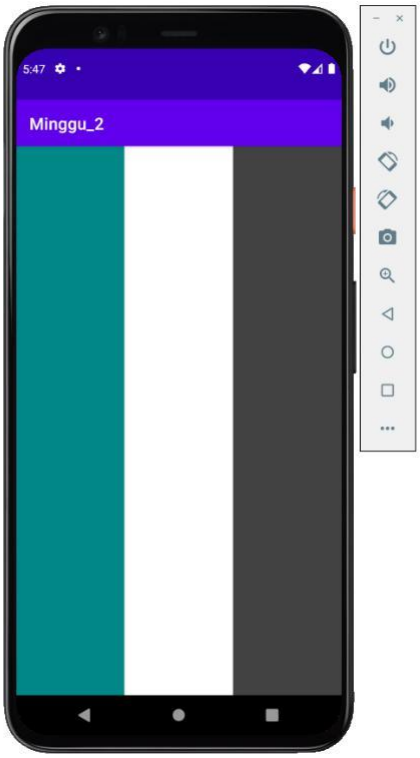
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="3"
    android:background="@color/cardview_dark_background"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@color/teal_700"
        >
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="#FFFFFF"
        >
    </LinearLayout>

</LinearLayout>
```

Pembahasan :

a. Mengubah properti *weightSum* yang berguna untuk menentukan jumlah total bagian ruang dari keseluruhan lebar atau tinggi parent layout pada linier layout dan digunakan pada child layout dengan *android:layout_weight* di masing-masing view di dalam LinearLayout untuk menentukan jatah ruang yang didapatkan.

b. Ketika merubah value dari *android:weightSum* menjadi **3**, maka karena pada child layout properti *android:layout_weight* memiliki value **1,** Akan menyebabkan masing-masing child layout akan memiliki lebar 1/3 dari ukuran parent layou. pada output bagian yang berwarna gelap adalah background dari parent layout yang terlihat 1/3 dari ukuran lebar layar, background dari parent layout terlihat karena child layout secara total hanya mengisi 2/3 dari ukuran lebar parent layout.

Hasil :



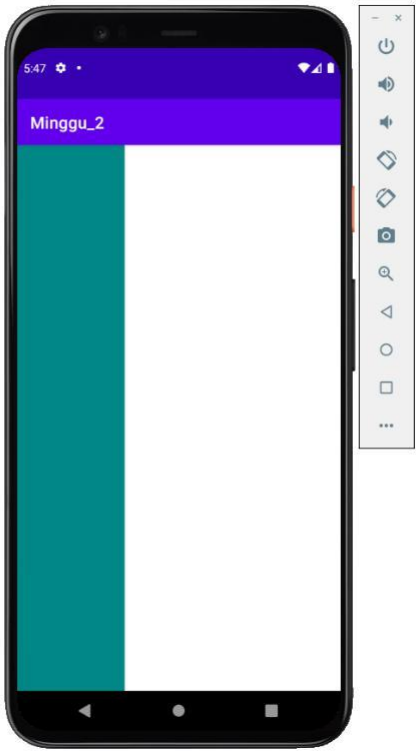
4. Mengubah layout_weight dari salah satu LinearLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="horizontal"
    android:weightSum="3"
    android:background="@color/cardview_dark_background"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:background="@color/teal_700"
        >
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="2"
        android:background="#FFFFFF"
        >
    </LinearLayout>

</LinearLayout>
```

Pembahasan :

Praktikum ini mengubah layout_weight dari salah satu child layout LinearLayout. Atribut *android:layout_weight* merupakan atribut untuk menentukan berapa lebar atau tinggi ruang ekstra untuk diberikan pada suatu view. ketika Atribut *android:layout_weight* dari child layout LinearLayout yang bawah atau pada tampilan output layout sebelah kanan dirubah valuenya menjadi **2**, maka ukuran lebar dari layout tersebut akan mengisi 2/3 bagian dari layar .

Hasil :



5. Kesimpulan

penataan layout dan penggunaan jenis layout yang tepat merupakan hal yang sangat dasar dan penting untuk membuat tampilan aplikasi yang bakal kita buat. Setelah melakukan praktikum ini saya menyimpulkan bahwa :

a. Beberapa layout yang bisa digunakan adalah Linier, Relative dan Constraint Layout.
b. Atribut *android:orientation="horizontal"* membuat tampilan layout yang berjejer secara horizontal.
c. Atribut *android:layout_width="wrap_content"* berfungsi mengikuti layar atau screen pada perangkat Android.

d. Atribut *android:layout_height="match_parent"* berfungsi supaya tinggi dapat mengikuti ukuran penuh dari layar atau screen pada perangkan Android.
e. Atribut *android:layout_weight* menentukan jumlah berat maksimum, dan dihitung sebagai jumlah *layout_weight* dari semua anak jika tidak ditentukan secara eksplisit.
f. Atribut *android:layout_weight* berfungsi untuk menentukan berapa lebar atau tinggi ruang ekstra untuk diberikan pada suatu view.

BAB IV

KESIMPULAN

Layout Editor memungkinkan Anda membuat tata letak dengan cepat dengan menyeret elemen UI ke editor desain visual, alih-alih menulis XML tata letak secara manual. Editor desain ini dapat menampilkan pratinjau tata letak Anda pada beragam versi dan perangkat Android, dan Anda dapat secara dinamis mengubah ukuran tata letak untuk memastikannya berfungsi dengan baik pada berbagai ukuran layar. Jadi layout di sini adalah bagian terpenting untuk memperindah tampilan pada aplikasi yang kita buat nyaman di lihat bagi pengguna.

DAFTAR PUSTAKA

BKPM Aplikasi Mobile.2021.” Linier, Relative dan Constraint Layout “ (Diakses 16 Maret 2021).

Membuat UI dengan Layout Editor.” https://developer.android.com/studio/write/layout-editor?hl=id “ (Diakses 16 Maret 2021).