



**BINAR**  
A C A D E M Y

# Design UI pada Android

Silver - Chapter 2 - Topic 4

---

**Selamat datang di Chapter 2 Topik 4 online  
course Android dari Binar Academy!**





# Hi Teman-Teman 🙌

Masih di Chapter 2 niih ...

Di topik sebelumnya, kita sudah bahas tentang aturan penamaan, atau code convention di Android Studio

Pada **Topik 4** ini, Kita akan belajar **Cara Menerapkan Android UI Layouting**. Yuk, lanjut!





**Detailnya, kita bakal bahas hal-hal berikut ini :**

- Layouting pada Android
- Menulis dan membuat XML
- Komponen-komponen View.
- Komponen-komponen Viewgroup





Ini adalah gambaran sekilas dari apa yang akan kamu pelajari hari ini

## LAYOUT

Layout pada Android merupakan suatu tampilan tata letak di Android yang berguna untuk mengatur tata letak setiap komponen tampilan.

### XML

Menulis / Memuat

#### View

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageView
- ImageButton
- ProgressBar
- Spinner

#### ViewGroup

- LinearLayout
- RelativeLayout
- ConstraintLayout
- TableLayout
- FrameLayout
- WebView
- ListView
- GridView



**Yupp, kita bakal ngomongin Layout  
beserta cara pembuatan Layout khusus  
dengan XML.**

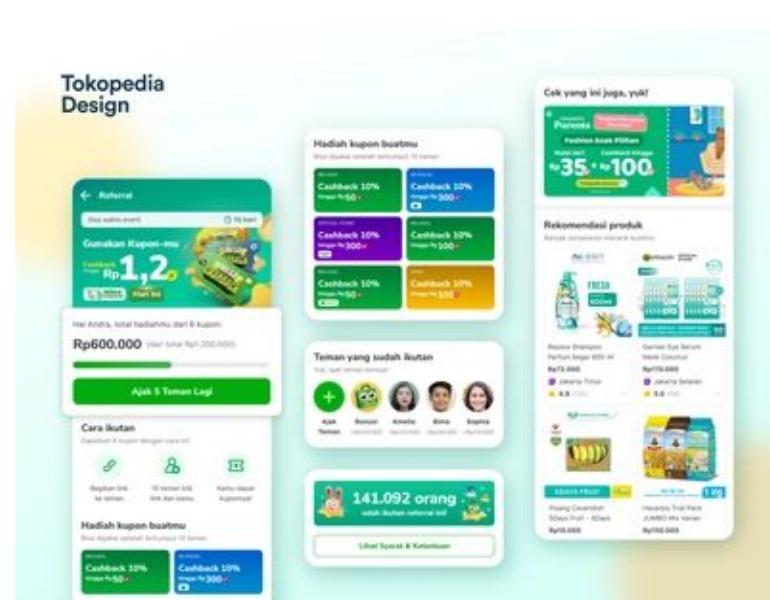
**Pertama, apa sih Layout itu?**



# Pastinya, kamu pernah pakai aplikasi dong?

Sekarang banyak banget aplikasi yang tersedia di playstore dengan ragam pelayanan yang ditawarkan.

Masing-masing aplikasi yang kamu lihat di playstore itu menawarkan fitur, warna dan tampilan yang pastinya berbeda juga dari yang lain~



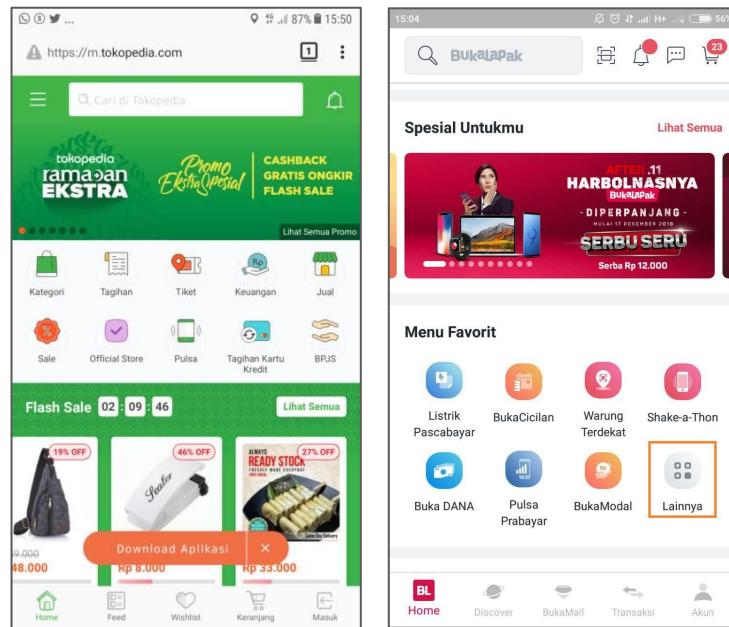


# Aplikasi sejenis aja, bisa punya variasi tampilan

Coba lihat dua tampilan aplikasi di samping.

Walaupun sama-sama bergerak di e-commerce, tapi beda banget kan?

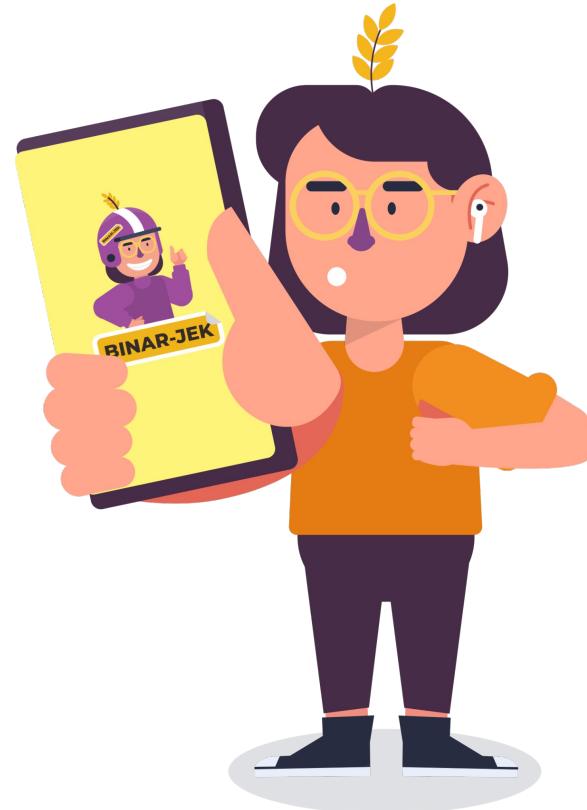
Nah, kita akan perdalam nih dasar-dasar membuat tampilan kayak gambar itu atau kita sebut dengan **layout**.



## Jadi, Apa itu layout?

Layout merupakan **tata letak dari suatu elemen desain**.

Ita mencakup pengaturan penempatan teks, gambar, atau komponen lainnya pada aplikasi sehingga tampilan yang dibuat terlihat rapi dan nyaman untuk pengguna.



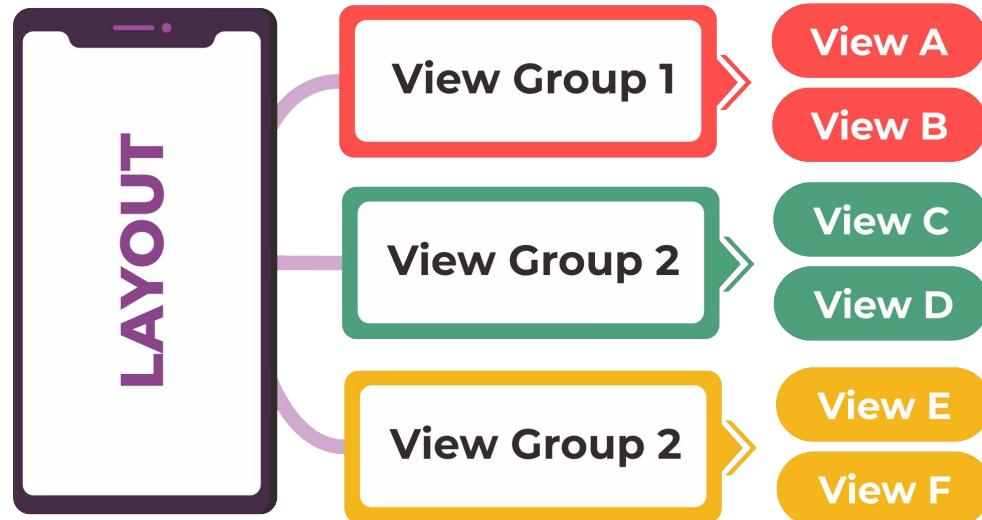


Dasar dari semua komponen tampilan di Android adalah **object view yang dibuat dari view class** dan menempati area yang terdapat di layar. Ia bertanggung jawab untuk menggambar dan menangani event yang terjadi.



Layout melingkupi **View** dan **ViewGroup**.

View mewakili **komponen yang dapat kita lihat** seperti teks, tombol, dan gambar. Sedangkan, ViewGroup adalah yang **mengatur semua komponen itu tertata rapi**.

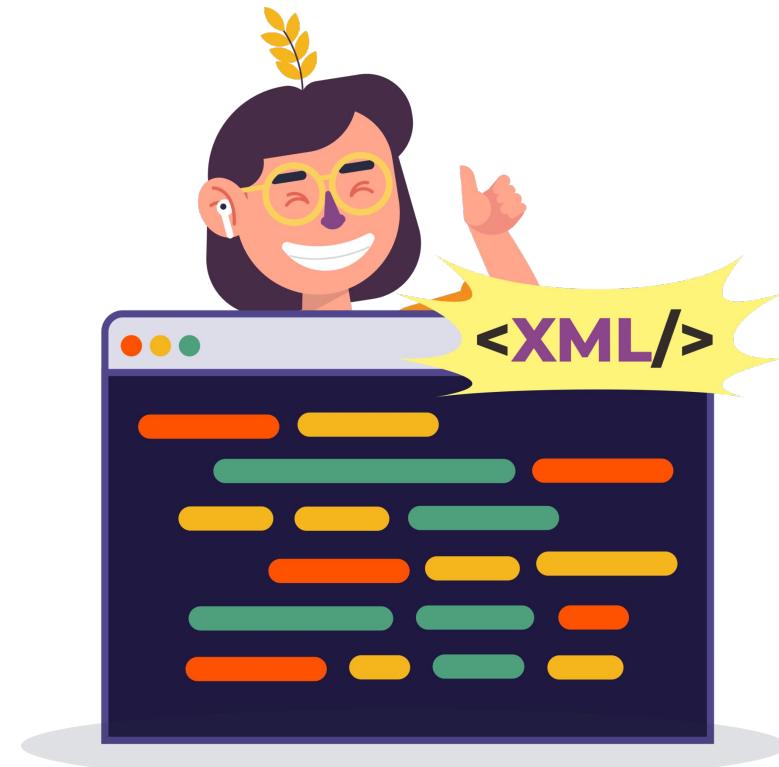


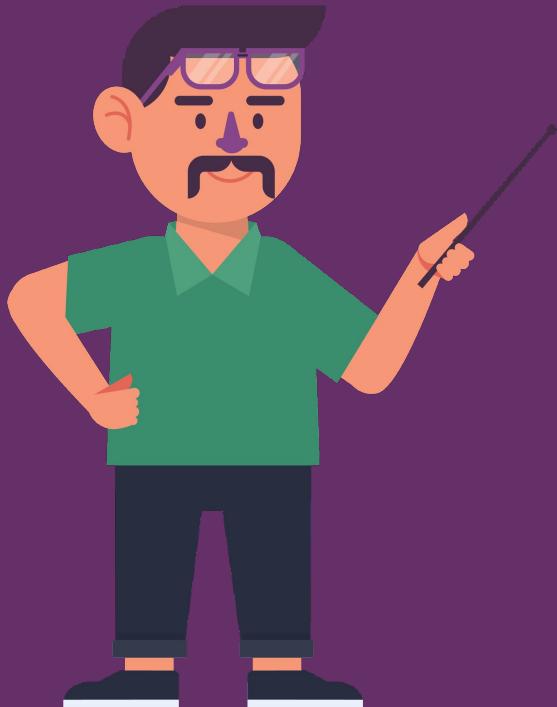


Baik View maupun ViewGroup, keduanya saling membutuhkan satu sama lain. Sehingga kita bisa membuat tampilan yang enak dipandang.

Kita juga bisa membuat *layout* khusus dengan membuat kelas yang diturunkan dari ViewGroup.

Nah disinilah skill koding kita yang sudah dipersiapkan sebelumnya akan bersinar. Semua yang kita lakukan untuk membuat layout khusus nggak akan jauh jauh dari **XML**.





# <XML/>

Untuk membuat **layout khusus** dengan **Android Studio** kita harus kenalan dulu sama **XML**.



### Jadi, XML itu apa sih?

Android XML adalah format pemrograman dalam bentuk teks yang bisa memberi keleluasaan pada kita dalam **mendesain UI layout dan elemen layar lain yang dikandungnya dengan cepat.**

Kenapa cepat? Karena caranya sama dengan membuat halaman web dalam HTML, yaitu menggunakan serangkaian elemen yang bersarang.





## View

### Text View

Hai Jimi Jamet

### Image View



### Button View

Home

Promo

Setiap file *layout* harus berisi **satu elemen root**, yang harus berupa object **View** atau **ViewGroup**.

Setelah kita mendefinisikan elemen root, kita dapat menambahkan layout object atau widget tambahan sebagai elemen turunan untuk secara bertahap membangun hirarki View yang menentukan layout kita.



## Kalau disimpulkan layout bisa dibuat dengan dua cara :

### 1. Mendeklarasikan UI dalam bentuk Code XML

Android menyediakan XML yang langsung sesuai dengan kelas View dan subclassnya, seperti yang digunakan widget dan layout.

Jika kita menggunakan versi Android Studio terbaru, maka layout XML akan otomatis menggunakan **ConstraintLayout**.

Kita juga dapat menggunakan Android Studio *layout editor* untuk membangun layout XML dengan **drag-and-drop**.



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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

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

Gambaran Deklarasi UI dalam XML



### 2. Instantiate layout elements saat runtime

Aplikasi kita dapat membuat object View dan ViewGroup (dan memanipulasi *properties*-nya) secara terprogram.

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

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

Gambaran Deklarasi UI dalam XML



**Mendeklarasikan UI dalam bentuk XML** membuat kita dapat memisahkan tampilan aplikasi dengan kode untuk mengontrol logic.

Opsi ini juga mempermudah kita dalam menyediakan layout yang berbeda untuk ukuran dan orientasi layar yang berbeda.

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

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

Gambaran Deklarasi UI dalam XML



Framework Android memberi kita fleksibilitas untuk menggunakan salah satu atau kedua method ini untuk membangun UI aplikasi.

Sebagai contoh, kita dapat mendeklarasikan layout default aplikasi kita dalam XML, dan kemudian memodifikasi layout tersebut saat runtime.

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment baru" />

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

Gambaran Deklarasi UI dalam XML



Contoh disamping merupakan layout XML yang menggunakan **LinearLayout** vertikal untuk menata TextView dan Button.

Setelah kita mendeklarasikan layout dalam XML, simpan file dengan ekstensi .xml, di folder res/layout/ pada project Android kita, jadi itu akan dikompilasi dengan benar.

Informasi lebih lanjut tentang syntax untuk file layout XML tersedia di dokumen Layout Resources.

```
<?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="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Hello, I am a TextView

HELLO, I AM A BUTTON



### Memuat Ulang (*Loading*) XML

Saat *compiling* aplikasi, setiap *file layout* XML dikompilasi ke resource View. Kita harus memuat *resource layout* dari kode aplikasi dalam method **onCreate()** pada Activity.

Method **onCreate()** dalam activity akan dipanggil oleh Android Framework **saat activity kita diluncurkan**.



```
fun onCreate(savedInstanceState: Bundle) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.main_layout)  
}
```



Cara memuat ulang XML, lakukan dengan memanggil **setContentView()**. Lalu, berikan referensi ke resource layout kita dalam bentuk: **R.layout.layout\_file\_name**.

Sebagai contoh, jika layout XML kita disimpan sebagai **main\_layout.xml**, kita akan memuatnya untuk *activity* kita seperti berikut:



```
fun onCreate(savedInstanceState: Bundle) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.main_layout)  
}
```



## <View/>

Kamu masih inget nggak di bagian slide introduction, XML bercabang menjadi dua? View dan GroupView.

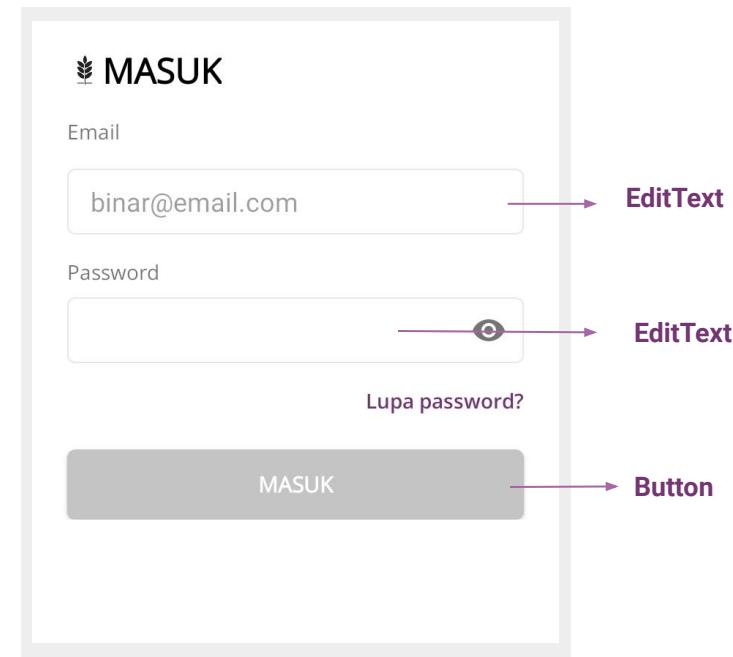
Kita mulai masuk ke situ nih.  
Selanjutnya, **View!**

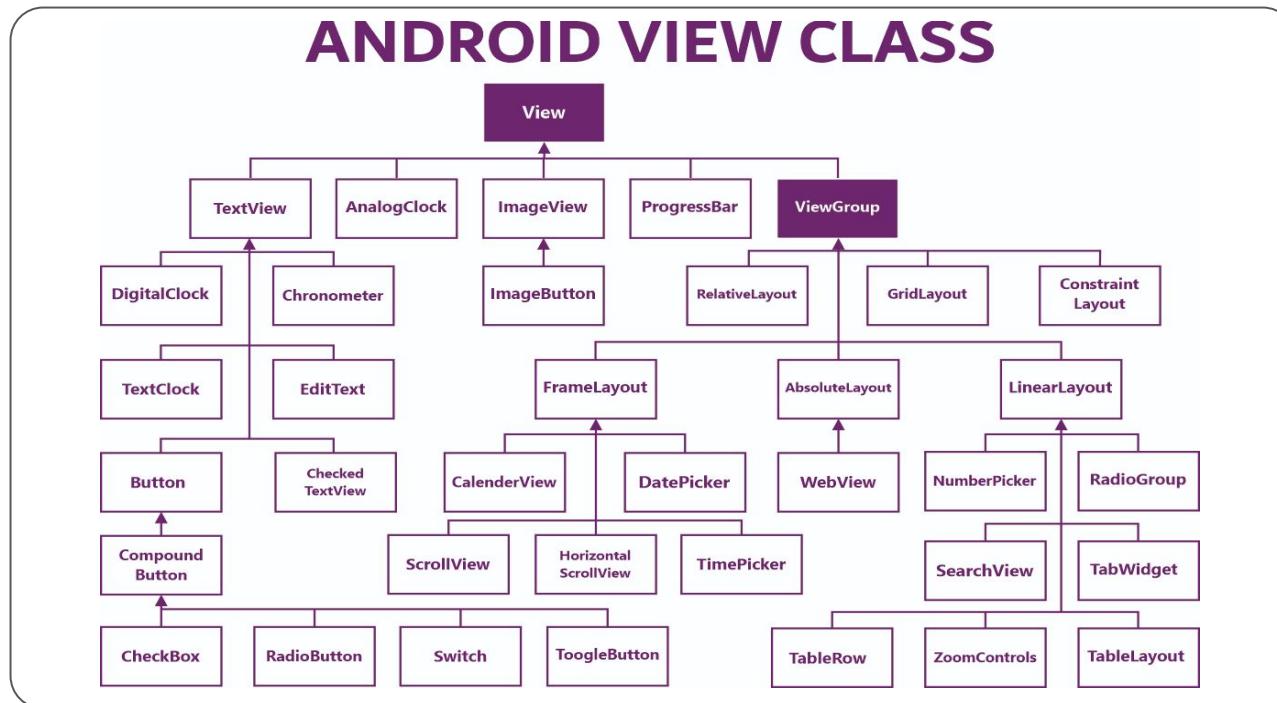


## Apa itu Android View

Android View adalah **kelas dasar untuk semua komponen UI di android.**

Misalnya, class **EditText** digunakan untuk menerima input dari user di aplikasi Android, yang merupakan subclass dari **View**.



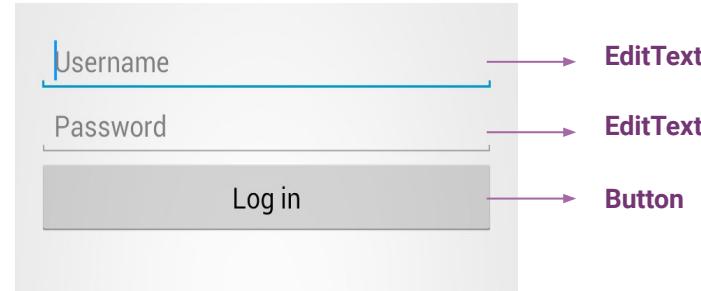


Gambar di atas ini menggambarkan hirarki pewarisan antar tampilan di Android. Walaupun keliatan bervariasi, tapi kita cukup mempelajari beberapa View yang sering digunakan dalam programming Android aja.



Berikut ini adalah beberapa **subclass View** umum yang akan digunakan dalam aplikasi android:

1. TextView
2. EditText
3. Button
4. CheckBox
5. RadioButton
6. ImageButton
7. ProgressBar
8. Spinner



Yuk kita bahas satu-satu~



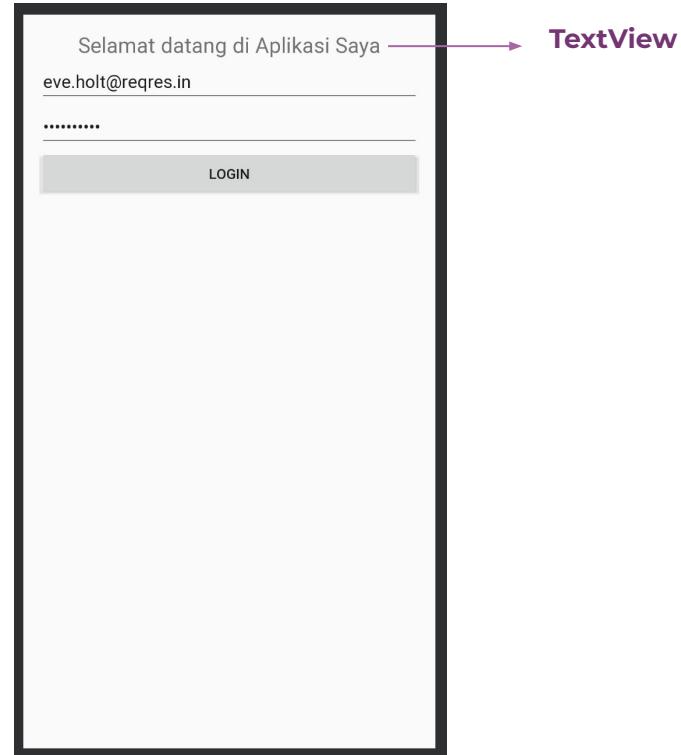
### View - TextView

**TextView** adalah komponen yang digunakan untuk **mengatur dan menampilkan teks ke layar**.

Itu akan bertindak seperti komponen label. Walaupun begitu, user tidak bisa mengubah atau mengedit text yang ada di *TextView*.

Pada Android, kita dapat membuat komponen *TextView* dengan dua cara:

- menulisnya dalam file *layout XML*, atau
- membuatnya dalam *file* secara terprogram.

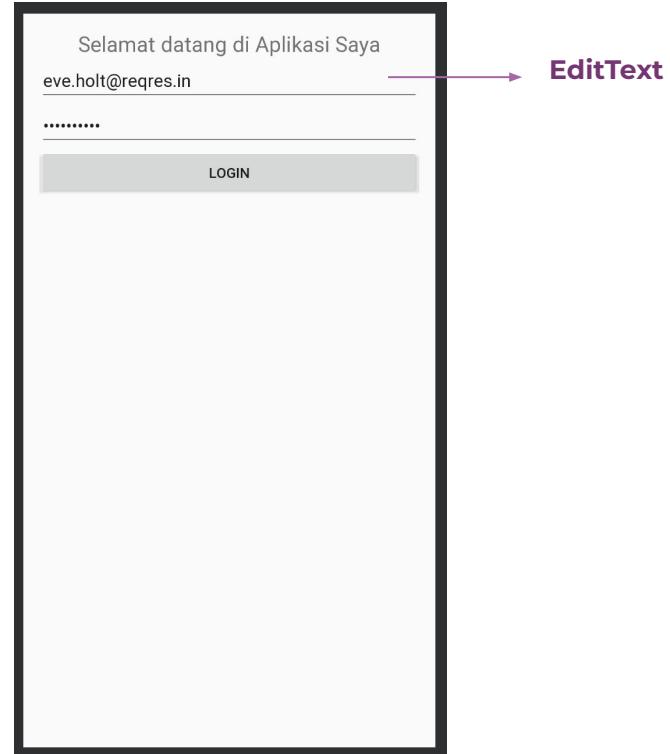


## View - EditText

**EditText** adalah komponen yang digunakan untuk **menginput berupa teks**.

EditText bertindak sebagai penampung teks yang kita ketik. EditText bisa menangani berbagai macam input, seperti :

- input untuk e-mail,
- password,
- nomor telefon,
- nomor pin, dll.





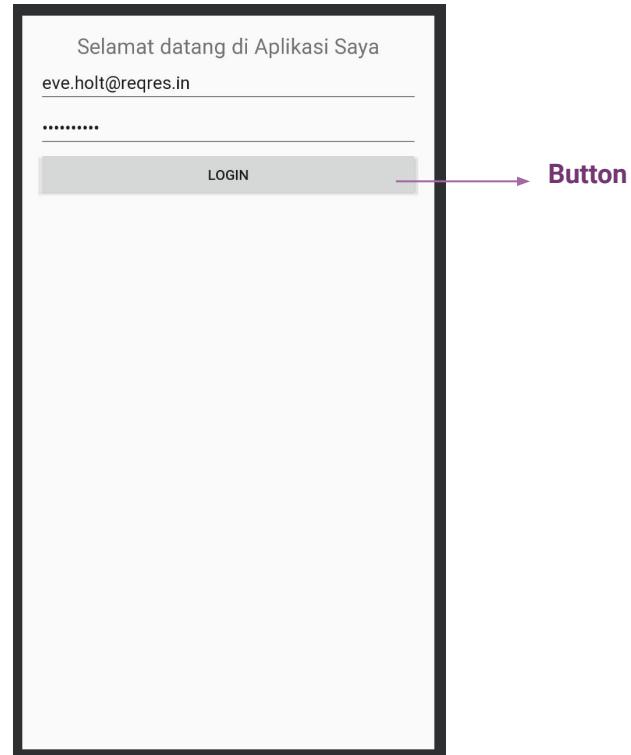
### View - Button

Button adalah **komponen yang digunakan untuk melakukan action setiap kali user mengetuk atau menahannya.**

Secara umum, Button akan berisi teks atau ikon atau keduanya dan melakukan suatu action ketika user menyentuhnya.

Kita dapat menggunakan berbagai jenis *button* yang tersedia untuk digunakan, yaitu :

- *ImageButton*,
- *ToggleButton*, dan
- *RadioButton*.





### View - CheckBox

Kita bisa menggunakan CheckBox untuk **menandai sebuah atau beberapa pilihan.**

Biasanya kita menemui penggunaan CheckBox ketika awal registrasi yang menandai kalau kita setuju dengan syarat dan ketentuan penggunaan aplikasi.

Saya seorang :

Laki-laki  
 Perempuan

Saya setuju dengan syarat dan ketentuan

**DAFTAR**

CheckBox ←

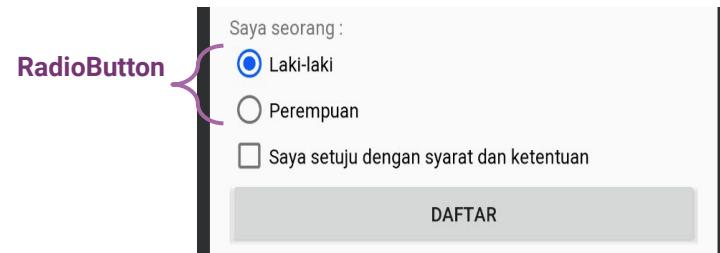


### View - RadioButton

Kita dapat menggunakan komponen **RadioButton** di aplikasi Android seperti **CheckBox**, namun user hanya bisa memilih satu opsi saja.

Untuk menggabungkan beberapa **RadioButton** menjadi satu grup, kita bisa menggunakan **RadioGroup**. Itu akan memastikan bahwa user hanya dapat memilih satu opsi dari grup beberapa opsi.

Contohnya seperti pertanyaan jenis kelamin di dalam survei.



The screenshot shows a user interface element for a survey. On the left, the text "RadioButton" is written vertically above a bracket that points to a "RadioGroup" containing two radio buttons: "Laki-laki" (selected) and "Perempuan". To the right of the group is a question "Saya seorang :". Below the RadioGroup is a checkbox labeled "Saya setuju dengan syarat dan ketentuan". At the bottom right is a large grey button labeled "DAFTAR".

Saya seorang :

Laki-laki  
 Perempuan

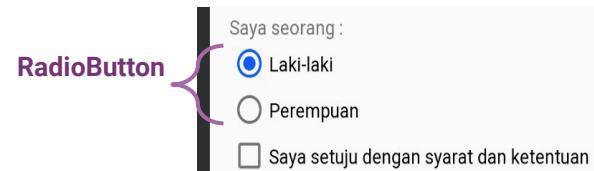
Saya setuju dengan syarat dan ketentuan

DAFTAR



Secara default, *RadioButton* akan muncul dalam keadaan OFF (tidak dipilih). Kita dapat mengubah status tersebut di *RadioButton* dengan menggunakan attribute **android:checked**.

Dalam hal ini, jika kita ingin mengubah status *RadioButton* ke ON (dipilih), maka kita perlu mengatur **android:checked="true"** di file layout XML.



The screenshot shows a user interface element for gender selection. It consists of a vertical list of options under the heading "Saya seorang:". The first two items are radio buttons: "Laki-laki" (selected, indicated by a blue outline) and "Perempuan". The third item is a checkbox labeled "Saya setuju dengan syarat dan ketentuan". A purple bracket on the left side of the image groups the first two items together, pointing towards the text "RadioButton" written vertically above them.

Saya seorang :

Laki-laki

Perempuan

Saya setuju dengan syarat dan ketentuan

DAFTAR

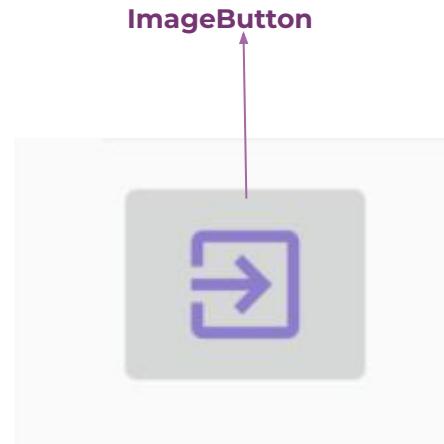


### View - ImageButton

Secara default, **ImageButton** terlihat sama dengan *button* lain, ia akan melakukan *action* ketika *user* mengetuk atau menyentuhnya. Tetapi, kita dapat menambahkan gambar khusus sebagai ganti teks.

Ini seperti **banner ads** yang kamu temui di website-website pada umumnya.

Kita dapat menambahkan gambar ke *button* dengan menggunakan attribute **<ImageButton>** dan menambahkan attribute android:src dalam file layout XML atau dengan menggunakan method **setImageResource()**.





### View - ProgressBar

ProgressBar adalah **komponen yang digunakan untuk menunjukkan progress dari action tertentu**. Misalnya, proses download atau upload file.

Secara default, **ProgressBar** akan ditampilkan sebagai roda yang berputar.

Jika kita ingin menampilkannya seperti *horizontal bar*, maka kita perlu mengubah properti *style* menjadi horizontal seperti `style="?Android:attr/progressBarStyleHorizontal"`.



ProgressBar



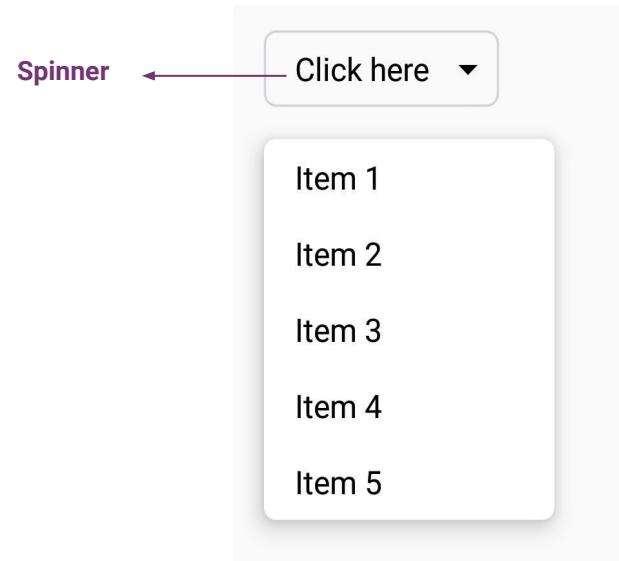
ProgressBar  
Style  
Horizontal



## View - Spinner

**Spinner** adalah komponen yang memungkinkan *user* memilih satu pilihan dari daftar pilihan. Spinner di Android akan sama dengan *dropdown menu* di web pada umumnya.

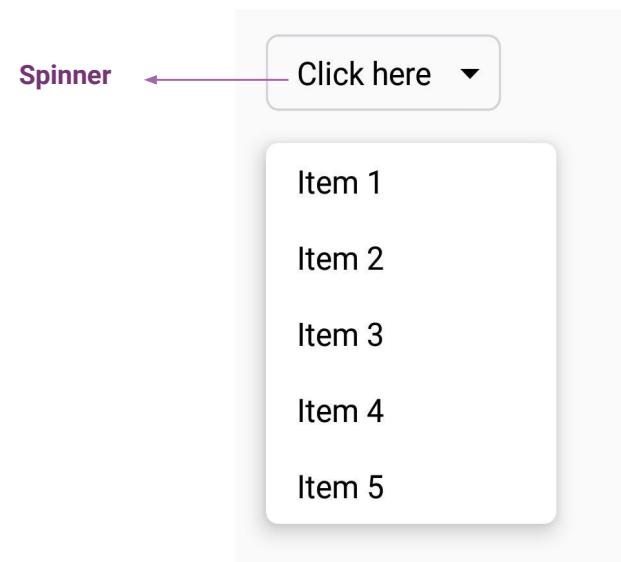
Secara default, *spinner* akan menunjukkan opsi yang dipilih saat ini dan dengan menggunakan **Adapter**.





Nah, kita dapat mengisi komponen *spinner* pada daftar pilihan dengan mendefinisikan **ArrayAdapter** di file *activity* kita.

*Adapter* memuat data dari *source* seperti *array* atau *database* dan mengubah setiap item menjadi tampilan hasil dan yang ditempatkan ke dalam daftar.





Sipp markisip banget deh! Kita udah mempelajari beberapa **View** Android.

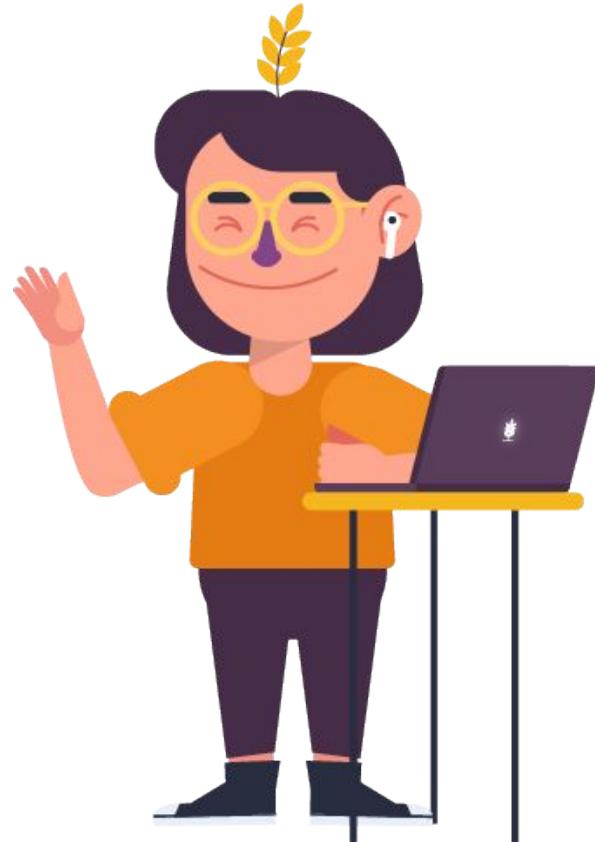
Sekarang kita pelajari ‘*Tempat*’ dari **View bertengger**, yaitu **ViewGroup**!



### Apa sih, ViewGroup itu?

Gampangnya, **ViewGroup** adalah *subclass* dari *view*.

*ViewGroup* akan menyediakan wadah yang tidak terlihat untuk menampung *View* dan *ViewGroup* lainnya.

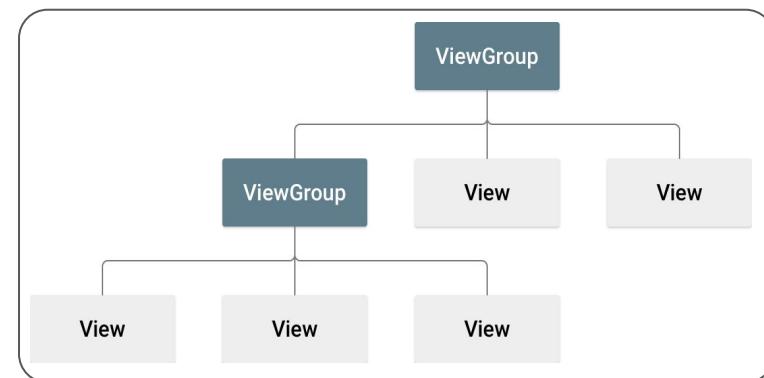




Dalam hierarki komponen View dan ViewGroup dapat juga digambarkan kayak gambar disamping.

Jika kita artikan, berarti dalam sebuah ViewGroup nantinya dapat menampung dua buah komponen View. Dimana satu komponen ViewGroup terdiri dari 3 buah komponen View.

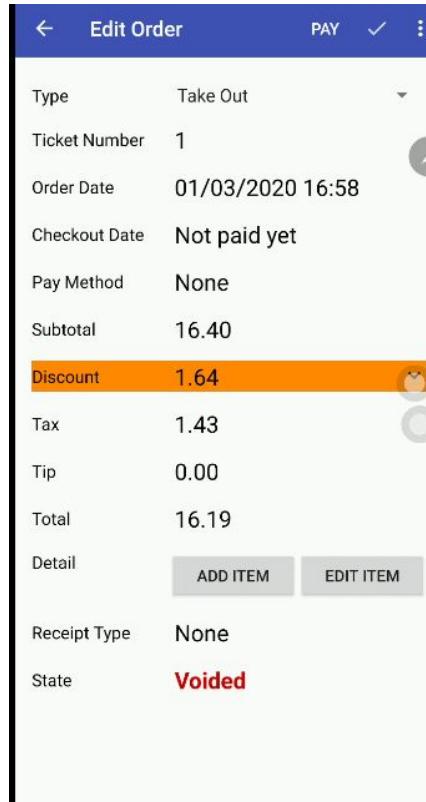
Lalu, di dalam ViewGroup tersebut, bisa terdapat View ataupun ViewGroup lagi. Hal ini disebut **Nested ViewGroup**.

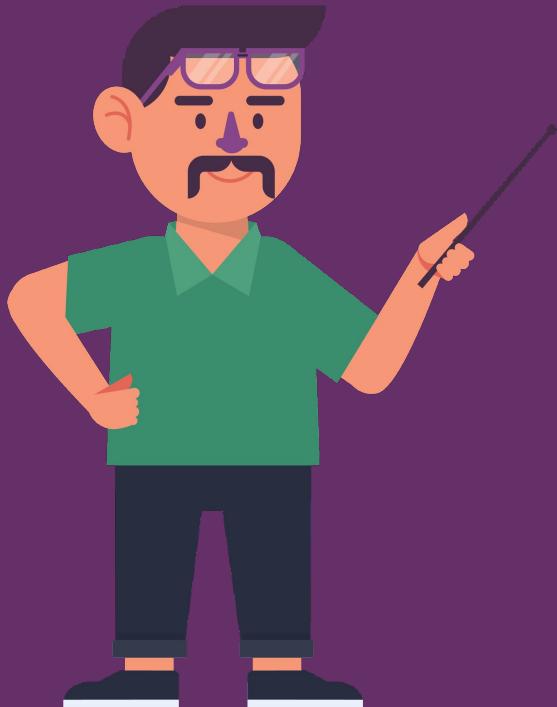




Coba deh kamu lihat gambar disamping. Nested Viewgroup berjalan kayak gitu.

Kalau kamu klik data pada “Discount”, akan muncul data-data lainnya. Grouping semacam ini lah yang disebut dengan **Nested ViewGroup**





## <ViewGroup/>

Kita udah belajar **ViewGroup** secara umum.

Kali ini kita kerucutin pembelajaran kita menuju ke komponen-komponen **Viewgroup** yang ada pada Android!



### Coba Recall dulu, ViewGroup itu apa sih?

ViewGroup itu yang **mengatur semua komponen itu tertata rapi**, sehingga kita bisa membuat tampilan yang enak dipandang.

Contohnya, kayak gambar aplikasi disamping ini~





### Apa aja tuh komponen-komponen ViewGroup?

Ada banyak komponen dalam ViewGroup yang bisa kita gunakan untuk mendesain. Selanjutnya, kita akan coba bahas beberapa komponennya satu per satu :

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Constraint Layout
- WebView
- ListView
- GridView
- Scroll

### Contoh ViewGroup



LinearLayout

RelativeLayout

GridLayout

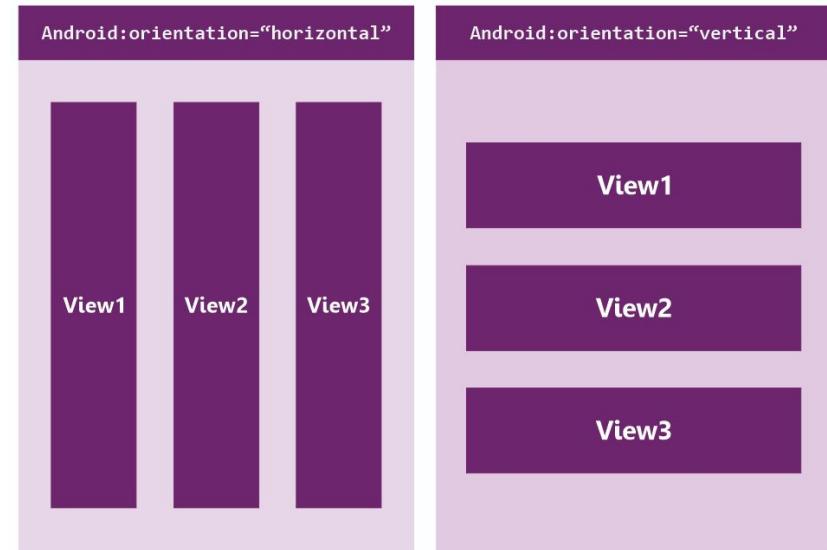


## LinearLayout

LinearLayout adalah *layout* yang akan menempatkan komponen-komponen di dalamnya secara berkelompok.

Nah, kita dapat memilih apakah layout akan dimuat secara horizontal maupun vertikal.

Kita dapat menentukan arah LinearLayout menggunakan *attribute android:orientation*.



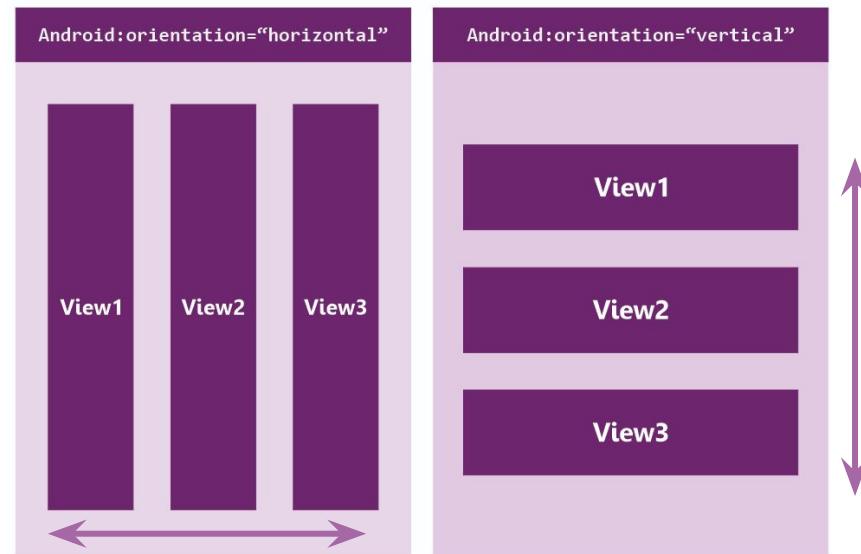
Jika kita ingin menampilkan arahnya menjadi vertikal, maka kita harus mendeklarasikan *attribute*-nya menjadi :

**android:orientation="vertical".**

Kalau mau menampilkan arahnya jadi horizontal, tinggal deklarasikan atributnya jadi :

**android:orientation="horizontal"**

Pada LinearLayout juga terdapat **attribute weight** yang berfungsi untuk membagi masing-masing porsi ukuran View, menyesuaikan space yang tersedia dalam layout.



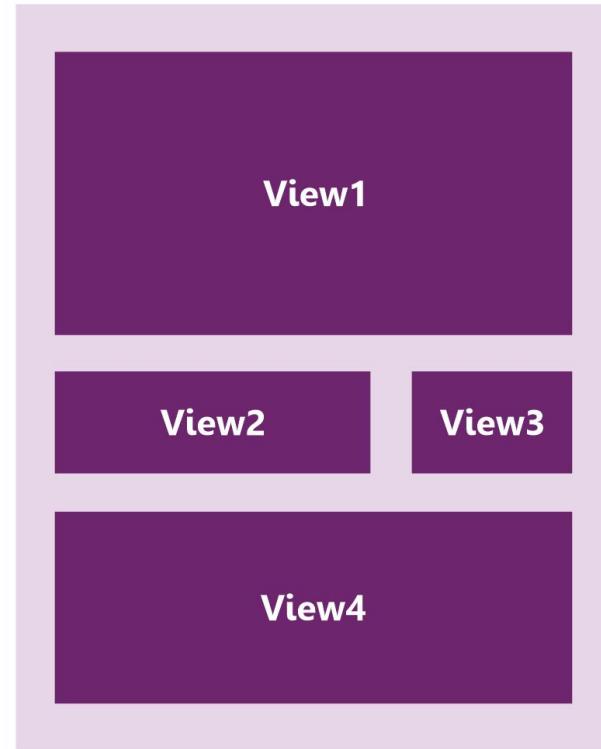


## RelativeLayout

[RelativeLayout](#) digunakan untuk **menentukan posisi setiap komponen secara relatif terhadap komponen yang lain.**

Posisi setiap View dapat ditentukan sebagai relatif terhadap komponen lain (seperti di sebelah kirinya atau di bawahnya tampilan lain).

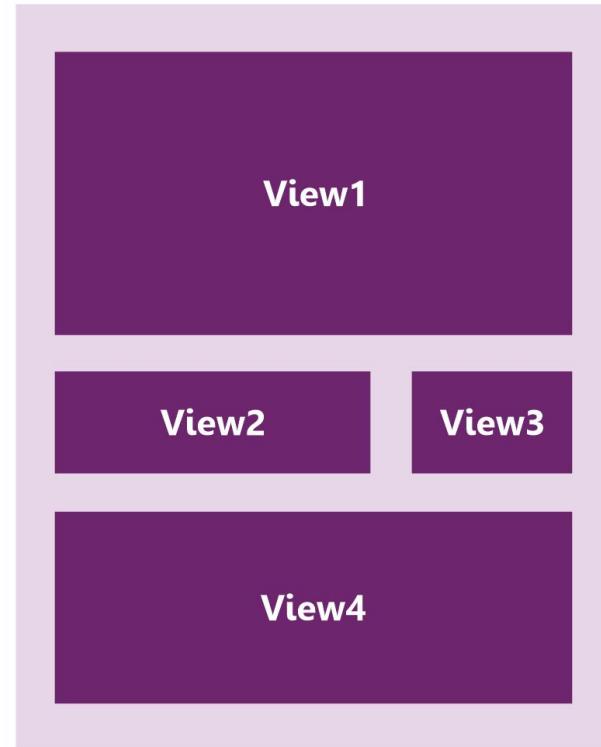
Atau, ia juga bisa berada di posisi yang relatif terhadap RelativeLayout itu (seperti disejajarkan dengan bagian bawah, kiri, atau tengah).





Dengan menggunakan RelativeLayout, kita dapat **menghilangkan Nested ViewGroup seperti pada LinearLayout.**

Selain itu, kita juga bisa menjaga hierarki *layout* tetap flat, yang dapat meningkatkan kinerja aplikasi.



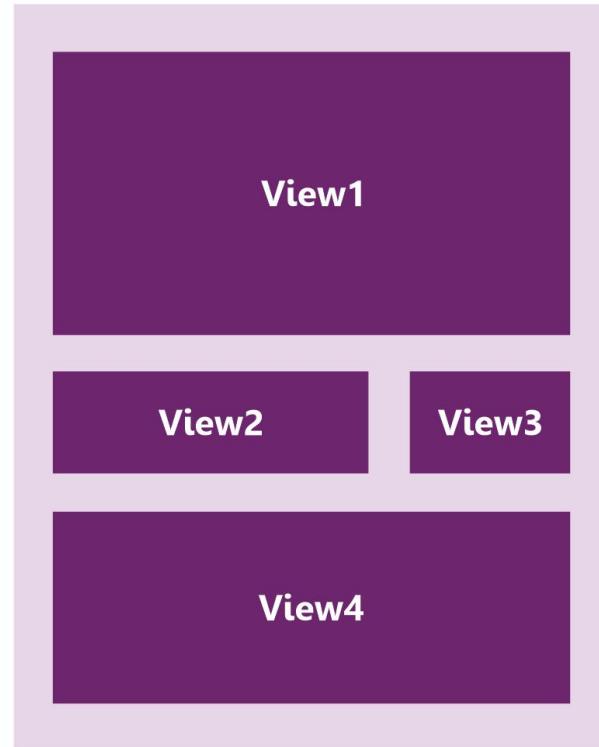


### RelativeLayout Positioning Views

Tambahan untuk RelativeLayout, sangat memungkinkan child View menentukan posisi mereka relatif terhadap *parent View* atau sama lain (ditentukan oleh ID).

Sehingga kita bisa menyelaraskan tuh dua elemen dengan batas kanan, atau membuat satu di bawah yang lain, berpusat di layar, berpusat di kiri, dan sebagainya.

Secara default, semua child View digambar di kiri atas layout, jadi kita harus menentukan posisi setiap View menggunakan berbagai properti layout yang tersedia dari RelativeLayout.





Beberapa layout properties yang ada pada views dalam relative Layout :

- **android:layout\_alignParentTop**  
Jika "true", jadikan tepi atas view nempel dengan tepi atas parent.
- **android:layout\_centerVertical**  
Jika "true", pusatkan view secara vertikal di dalam parent.
- **android:layout\_below**  
Tepi atas View ini nempel dibawah View yang ditentukan dengan ID View.
- **android:layout\_toRightOf**  
Tepi kiri View ini nempel di sebelah kanan View yang ditentukan dengan ID resource.

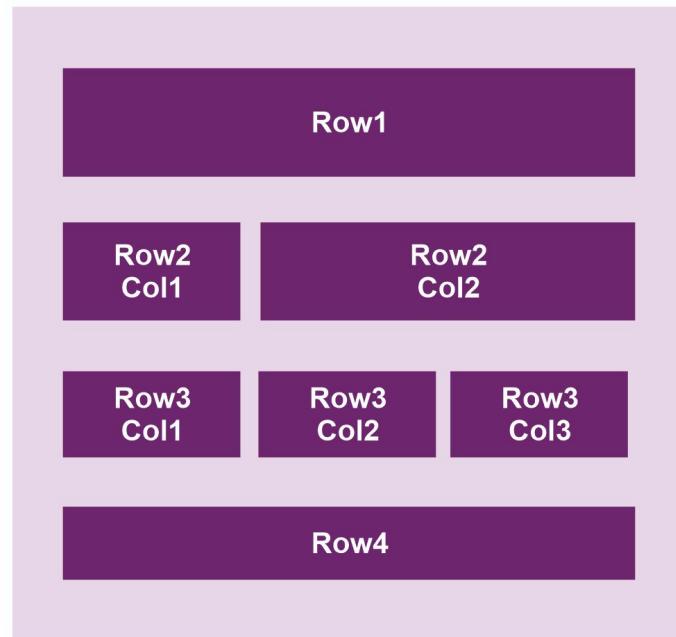
Ini hanya beberapa contoh. Semua attribute Layout didokumentasikan di `RelativeLayout.LayoutParams`.



## TableLayout

TableLayout merupakan layout yang digunakan untuk menampilkan komponen dalam baris dan kolom, seperti membuat tabel. Nah, kita bisa menggunakan komponen **<TableRow>** untuk membuat baris di tabel.

Setiap baris punya nol atau lebih cell, dan setiap cell dapat menampung satu object View. TableLayout tidak akan menampilkan garis pembatas ketika membuat baris, kolom, atau cellnya.



## FrameLayout

FrameLayout akan membuat komponen yang ada di dalamnya **menjadi menumpuk atau saling menutupi satu sama lain**. Komponen yang pertama pada layout ini akan menjadi alas bagi komponen-komponen di atasnya.

**FrameLayout** memiliki kemampuan untuk menjadi kontainer untuk fragment-fragment di dalam sebuah activity

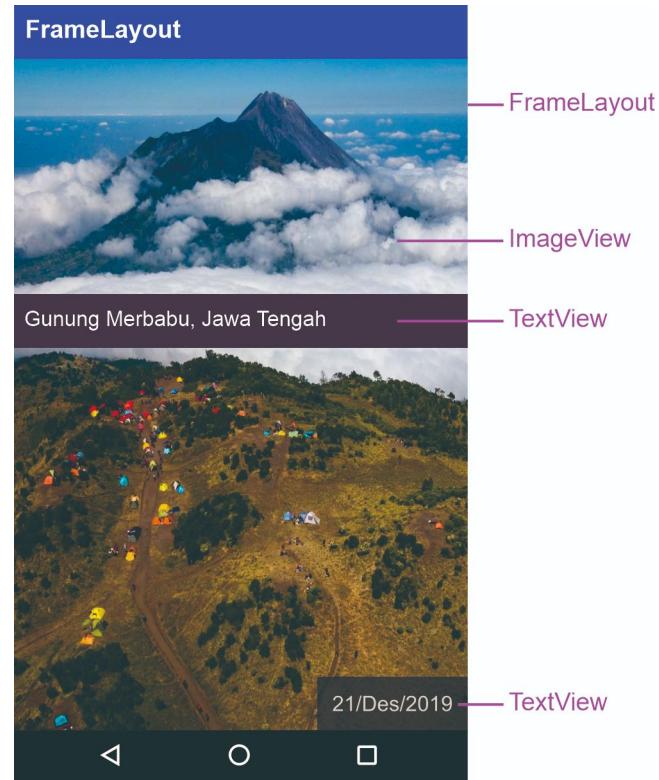




Jika kita ingin menerapkannya langsung pada Android, maka contohnya seperti pada gambar.

Pada contoh gambar, kita menggunakan FrameLayout sebagai ViewGroup teratas. Di dalam FrameLayout ini terdapat ImageView dan beberapa TextView.

Posisi ImageView dan TextView dalam FrameLayout dapat kita atur dengan menggunakan atribut **android:layout\_gravity**.



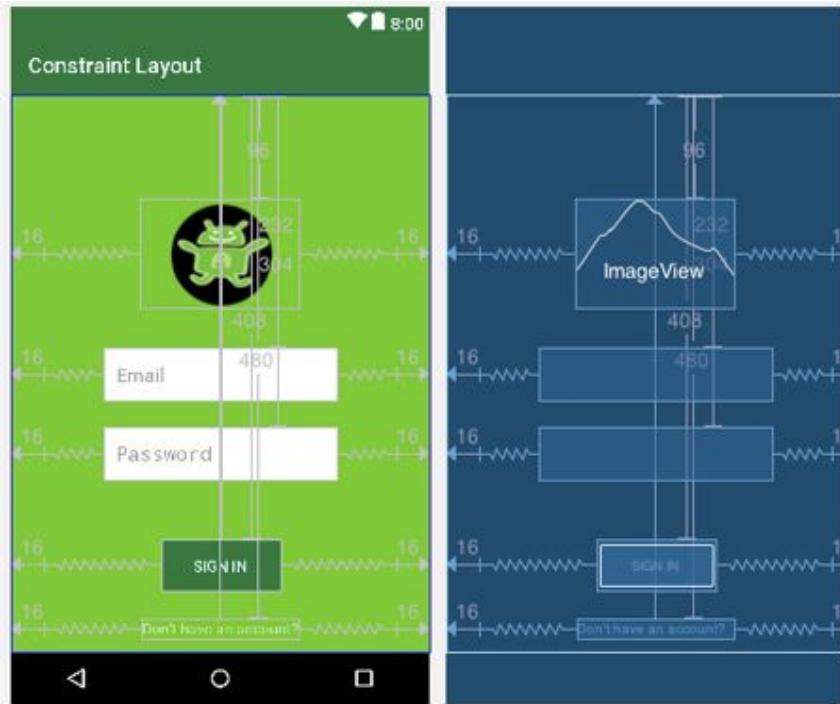


## ConstraintLayout

ConstraintLayout memungkinkan kita **membuat layout yang kompleks dengan hierarki flat (tanpa Nested ViewGroup)**.

Mirip dengan dengan RelativeLayout bukan? dimana semua View ditata sesuai dengan keterkaitannya dengan View, ViewGroup, ataupun pada parentnya.

Tapi funfact-nya, ConstraintLayout lebih fleksibel daripada RelativeLayout.



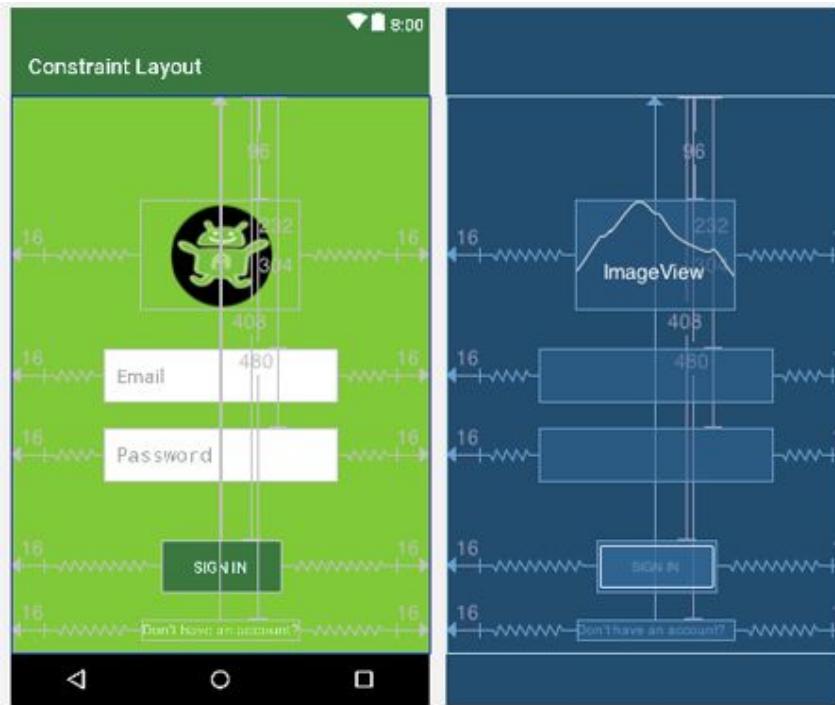


ConstraintLayout mempermudah kita untuk mendesain layout dengan cara **drag-and-drop** yang sudah terintegrasi dengan Android Studio.

ConstraintLayout tersedia dengan API library yang compatible di Android 2.3 (API level 9) dan seterusnya.

Jika kita membuat sebuah project baru dengan menggunakan Android Studio versi 3.4 ke atas, maka ConstraintLayout akan dijadikan default pada layout main.

Lain halnya ketika kita sudah mengganti default tersebut.



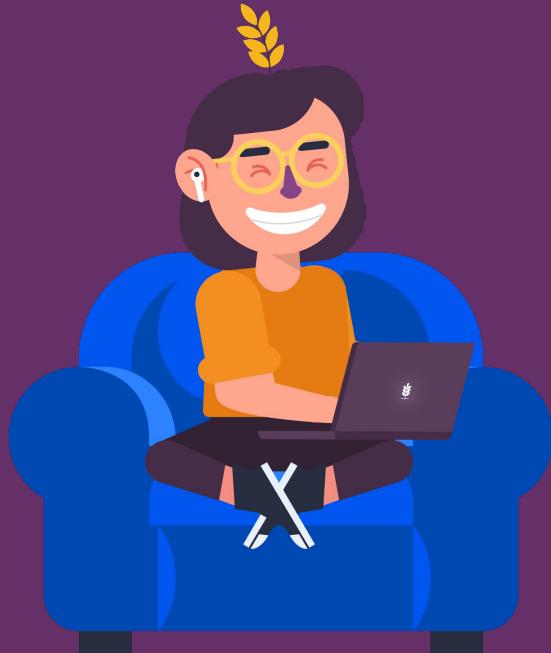


Jika kamu lebih nyaman mendesain **ConstraintLayout** dengan tampilan Code XML, berikut ini adalah atribut-atribut yang bisa kamu manfaatkan :

Atribut	Fungsi
layout_constraintTop_toTopOf	Mensejajarkan bagian atas View yang diinginkan ke atas yang lain.
layout_constraintTop_toBottomOf	Mensejajarkan bagian atas View yang diinginkan ke bagian bawah yang lain.
layout_constraintBottom_toTopOf	Mensejajarkan bagian bawah tampilan yang diinginkan ke atas yang lain.
layout_constraintBottom	Mensejajarkan bagian bawah tampilan yang diinginkan ke bawah yang lain. (Cont.)



Atribut	Fungsi
layout_constraintRight_toTopOf	Mensejajarkan sebelah kanan tampilan yang diinginkan ke atas yang lain.
layout_constraintRight_toBottomOf	Mensejajarkan sebelah kanan tampilan yang diinginkan ke bawah yang lain.
layout_constraintRight_toLeftOf	Mensejajarkan kanan tampilan yang diinginkan ke kiri yang lain.
layout_constraintRight_toRightOf	Mensejajarkan kanan tampilan yang diinginkan ke kanan yang lain.



Nah, beda lagi nih kalo kamu lebih nyaman desain **ConstraintLayout** menggunakan mode tampilan Design.

Kita perlu tahu **komponen-komponen ConstraintLayout** supaya membantu kita design dengan mode Design!

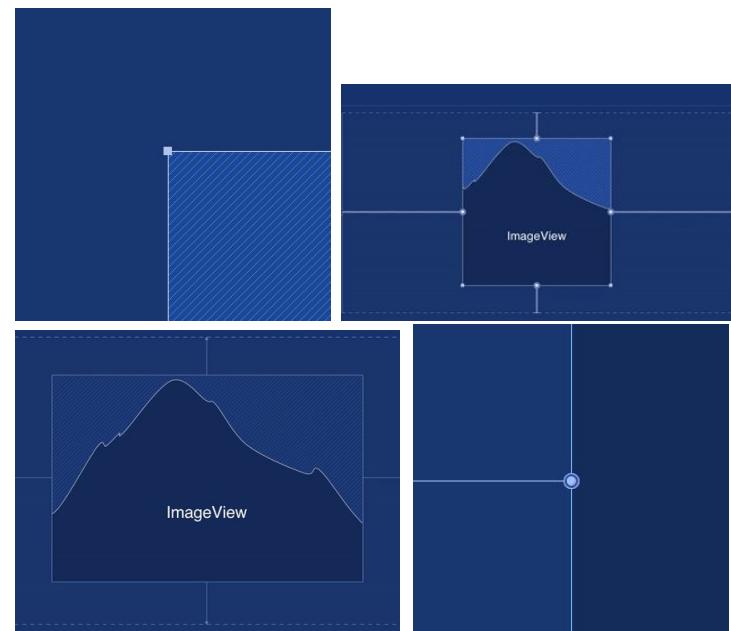


### Apa aja tuh Komponen-Komponen ConstraintLayout?

Ada 4 Komponen dalam ConstraintLayout yang bisa kita gunakan untuk mendesain, kayak gambar di samping :

- **Resize Handle**
- **Resize Anchor**
- **Side Constraint Handle**
- **Horizontal Bias**

Yuk kita bahas satu persatu~



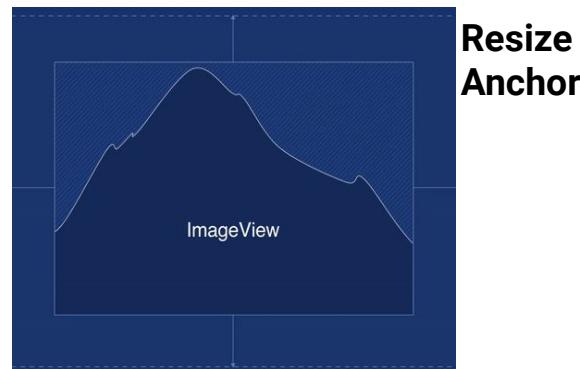
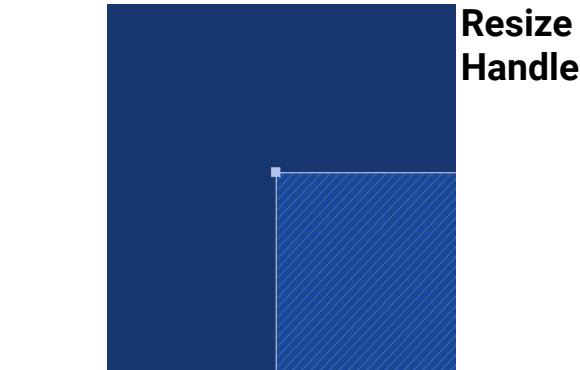


### Resize Handle

Kita bisa mengubah ukuran View yang dipilih dan menetapkan Constraint untuk ukuran baru.

### Resize Anchor

Mengubah ukuran View menggunakan resize anchor dalam layout editor akan secara otomatis menghitung ulang Constraint yang telah ditetapkan sebelumnya pada View tersebut.





## Side Constraint Handle

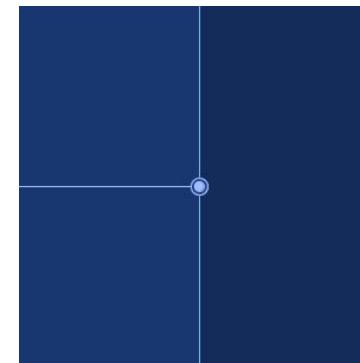
Digunakan untuk menentukan lokasi View dalam layout. Misalnya, anchor ini dapat digunakan untuk menentukan bahwa View yang dipilih selalu ditampilkan di sebelah kiri View lain, dengan ukuran margin yang ditentukan.

## Horizontal Bias

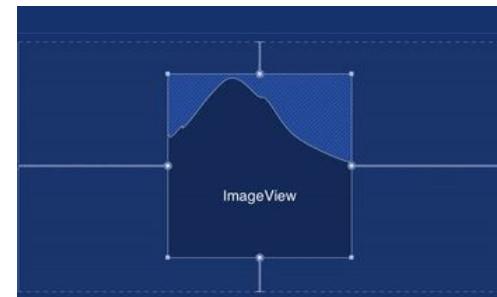
Ini memungkinkan kita untuk memposisikan View di sepanjang sumbu horizontal menggunakan nilai bias, posisi view akan relatif tergantung ukuran lebar layar.

Kita dapat mengatur horizontal bias menggunakan slider di layout editor atau menggunakan attribute di XML kita seperti:

```
app:layout_constraintHorizontal_bias="0,5"
```



**Side  
Constraint  
Handle**



**Horizontal  
Bias**



Gimana menurut kamu bahasan komponen **ConstraintLayout**-nya?

**Daebak? Amazing? Sasageyo?**

Mudah-mudahan belajar kita sangat ramah bintang lima yaa. Setelah ini kita menuju ke **pengaturan ukuran ConstraintLayout** pada mode Design. Mari~



### Gimana tuh pengaturan ukuran pada ConstraintLayout?

Pengaturan ukuran ConstraintLayout, atau kita sebut **View Sizing**, dilakukan dengan beberapa cara :

- AnySize
- Warp Content
- Auto-connect
- Manual Connect

Yuk kita bahas satu-satu~



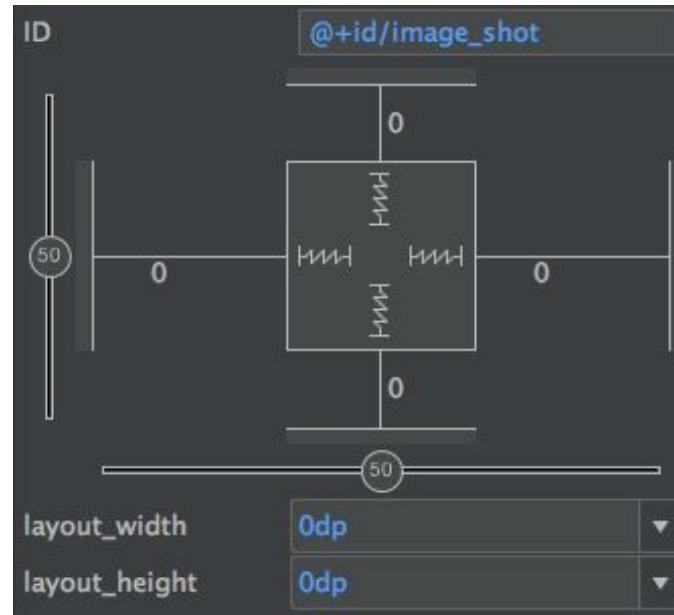


### Any Size

Di dalam kotak kita dapat melihat garis berlekuk-lekuk yang menandakan bagaimana View akan mengubah ukuran dalam layout.

Baris-baris tersebut menyatakan bahwa ukuran View akan mematuhi Constraint, yang berarti akan menjadi **match\_parent** dan menggunakan ruang yang tersedia.

Mengubah layout\_width dan layout\_height secara manual dengan nilai 0dp sama saja dengan menerapkan Any Size.

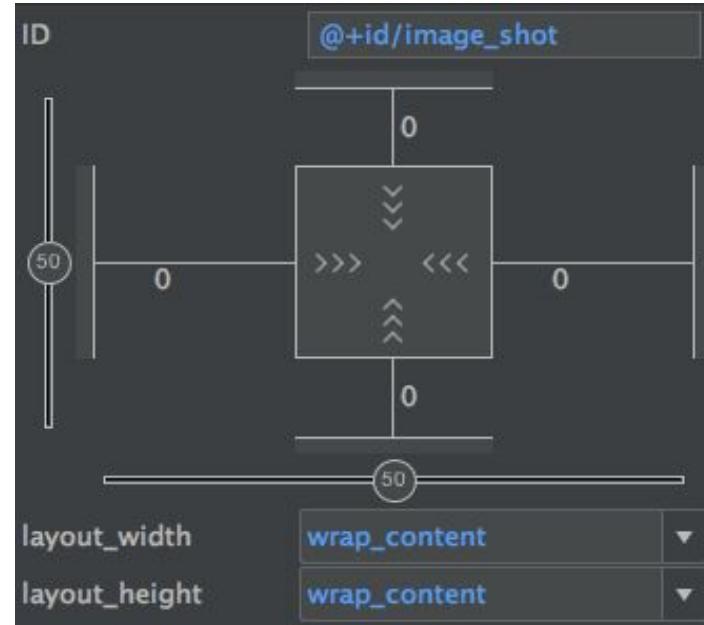


Any Size



## Wrap Content

Garis-garis ini sekarang menyatakan bahwa tinggi dan lebar View akan membungkus konten, yang berarti hanya akan sebesar yang diperlukan. Misalnya nih kalau image maka akan sesuai dengan ukuran gambar aslinya.



Wrap Content



### Auto-connect

Auto-connect memungkinkan kita untuk menempatkan View di dalam layout editor dan membuatnya secara otomatis menghitung dan mengatur Constraint untuk kita.

### Manual Connect

Manual connect mirip kayak menonaktifkan Auto-connect. Manual connect dilakukan jika kita ingin secara manual mengatur Constraint dalam layout editor.

Psst! Dengan menggunakan ini, kita bisa lebih leluasa mengatur Constraint dan menghemat waktu kita menghapus Constraint otomatis yang tidak kita butuhkan!

Cara Penggunaan :



Make sure this guy is enabled!

Pastikan icon Magnet-nya sudah diaktifkan ya!



Now auto-connect is disabled

Untuk manual connect, cukup dengan non-aktifkan icon Magnet yang tadi!



## WebView

WebView pada Android merupakan pengembangan dari kelas ViewGroup dan digunakan untuk **menampilkan konten halaman web HTML statis atau konten halaman web dengan URL** di aplikasi Android sebagai bagian dari layout activity.

Pada umumnya, WebView akan bertindak sebagai browser yang disisipkan untuk menampilkan konten halaman web dalam layout activity kita.

Namun fiturnya nggak terlalu lengkap kayak browser normal. Misalnya, nggak adanya address bar, navigation control, dll.

```
<html>
<body>

<!-- Web Page Content -->

</body>
</html>
```



WebView berguna untuk memasukkan dan menampilkan konten halaman web lain atau konten aplikasi di halaman yang diperlukan, seperti end-user agreements, dll.

Kita dapat mengimplementasikan WebView pada Android dengan menambahkan kode kayak gambar disamping pada XML:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

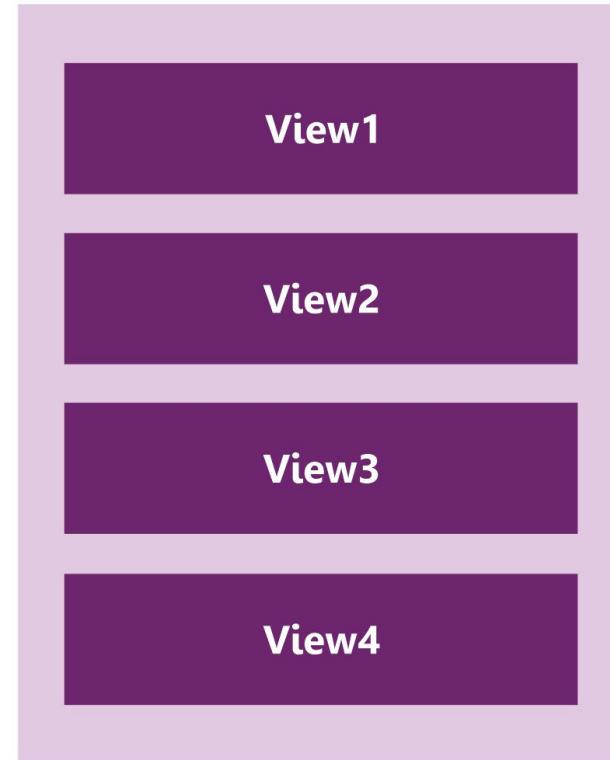


### ListView

ListView adalah ViewGroup yang berfungsi untuk **mengelompokkan beberapa item dan menampilkannya dalam list yang dapat di scroll secara vertikal.**

Item list secara otomatis dimasukkan ke list menggunakan Adapter yang menyediakan konten dari sumber data seperti array atau database.

Agar ListView dapat menampilkan daftar data, diperlukan sebuah **Adapter**. Adapter menjembatani antara UI Component dan sumber data yang mengisi data ke UI Component.





Secara umum, ada **berbagai jenis adapter** yang tersedia di Android untuk mengambil data dari data sources yang berbeda. Sehingga memungkinkan kita untuk mengisi data ke dalam tampilan adapter.

Adapter	Deskripsi
ArrayAdapter	ArrayAdapter menggunakan Array atau List sebagai input.
CursorAdapter	menerima instance kursor sebagai input.
SimpleAdapter	menerima data statis yang ditentukan dalam resources.
BaseAdapter	implementasi umum untuk ketiga jenis adapter dan dapat digunakan untuk ListView, Gridview atau Spinners berdasarkan persyaratan tertentu



## GridView

GridView adalah ViewGroup yang digunakan untuk menampilkan item dalam bentuk kotak-kotak dua dimensi (baris dan kolom), kotak yang dapat di scroll dan item grid secara otomatis dimasukkan ke gridview layout menggunakan ListAdapter.

ListView dan GridView adalah subclass dari AdapterView dan dapat diisi dengan Adapter, yang mengambil data dari source eksternal dan membuat View yang mewakili setiap entri data.





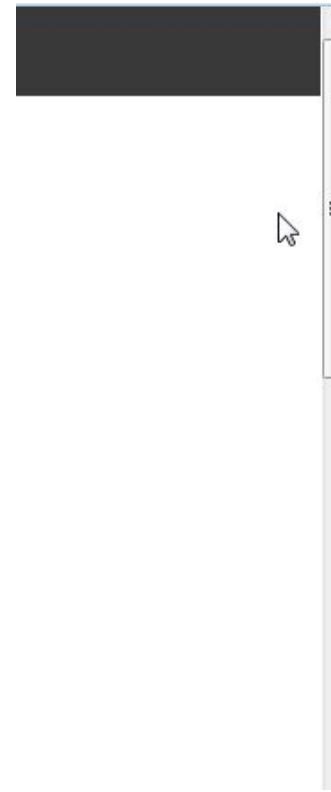
## ScrollView

ScrollView adalah sejenis layout yang berguna untuk menambahkan scroll bar vertikal atau horizontal ke konten yang lebih besar dari ukuran sebenarnya seperti LinearLayout, RelativeLayout, FrameLayout, dll.

Secara umum, ScrollView berguna ketika kita memiliki konten yang nggak muat dengan layar aplikasi android kita.

ScrollView akan memungkinkan pengguna untuk melihat konten lengkapnya dengan cara *scrolling*.

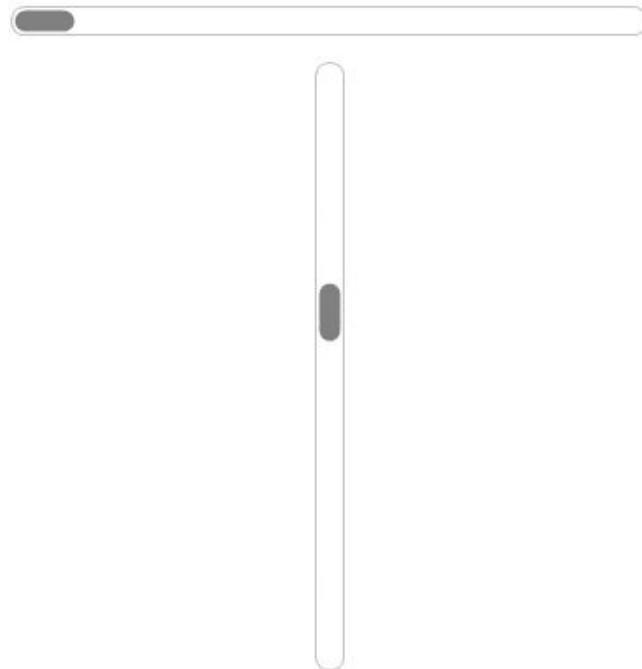




Android ScrollView **hanya dapat menampung satu ViewGroup.**

Jika kita ingin menambahkan beberapa View dalam ScrollView, maka kita perlu memasukkannya dalam ViewGroup seperti LinearLayout, RelativeLayout, FrameLayout, dll.

Untuk mengaktifkan scrolling pada aplikasi android kita, ScrollView adalah pilihan terbaik tetapi kita tidak boleh menggunakan ScrollView bersama dengan ListView atau GridView karena keduanya akan melakukan scrolling vertikal dengan caranya masing-masing.



ScrollView hanya mendukung scrolling vertikal saja. Jika kita ingin menerapkan scrolling horizontal, maka kita perlu menggunakan komponen **HorizontalScrollView**.

Android ScrollView memiliki properti bernama **android:fillViewport**, yang digunakan untuk menentukan apakah ScrollView harus merentangkan kontennya untuk mengisi viewport atau tidak.



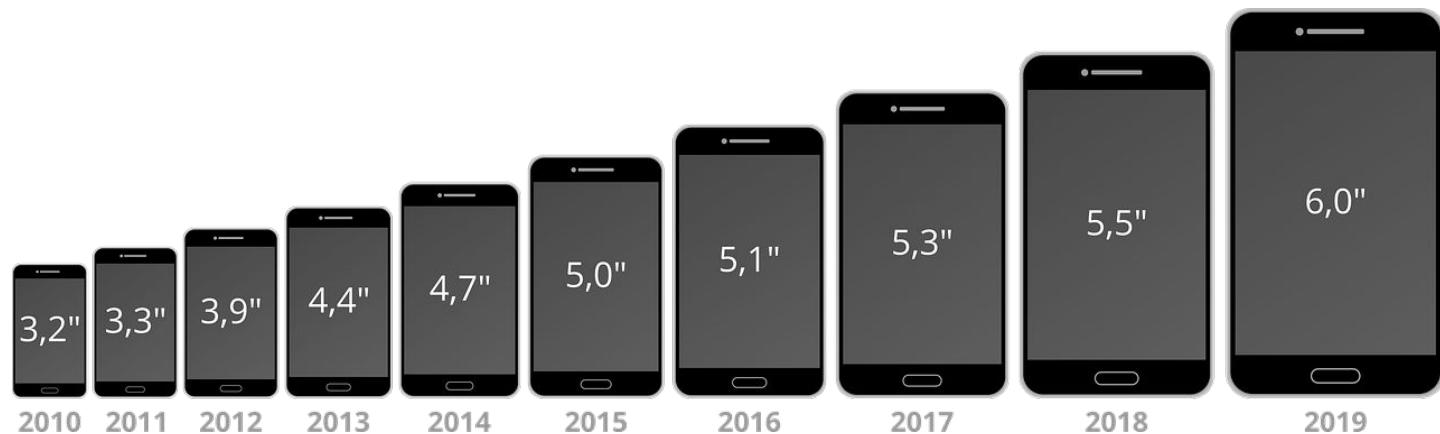
**Ngomongin View dan ViewGroup yang udah kita bahas tadi, ada loh satuan dimensi untuk menentukan ukuran keduanya.**



### Dimensi pada Android

Device Android terkenal dengan keberagaman ukuran perangkat, layar, kepadatan pixelnya, hingga Operating Sistemnya (OS). Karena hal tersebut, kita perlu mempertimbangkan mengenai penataan layout yang baik dan benar supaya aplikasi dapat berjalan dan tampil secara maksimal.

Kabar baiknya, developer Android sudah memperkirakan hal itu. Sehingga Android memiliki satuan unit dimensinya sendiri untuk membantu kita menentukan ukuran tinggi dan lebar dari sebuah komponen View atau ViewGroup.





## Dimensi pada Android

- **px (pixels)** - Jumlah piksel di layar. Tidak sering digunakan karena layar memiliki kerapatan piksel yang berbeda.
- **in (inci)** - Ukuran fisik layar. Tidak sering digunakan karena ukuran layar yang berbeda memiliki resolusi yang berbeda. 1 Inch = 2.54 sentimeter.
- **mm (milimeter)** - Ukuran fisik layar, mirip dengan cara kerja inci.

Dimension	Description	Units / Physical Inch	Density Independent	Same Physical Size on Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density independent pixels	~160	Yes	No
sp	Scale independent pixels	~160	Yes	No



- **pt** [1/72 inci (2,54 cm)] - berdasarkan ukuran fisik layar.
- **dp or dip** (Density-independent Pixels) - unit abstrak yang didasarkan pada kepadatan fisik layar.

Unit-unit ini relatif terhadap layar 160 dpi, jadi satu dp adalah satu piksel pada layar 160 dpi.

Ini umumnya paling sering digunakan ketika mengatur tata letak atau ukuran apa pun pada aplikasi Android.

Dimension	Description	Units / Physical Inch	Density Independent	Same Physical Size on Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density independent pixels	~160	Yes	No
sp	Scale independent pixels	~160	Yes	No



- **sp** (Scale-independent Pixels) - ini seperti unit dp, tetapi juga diskalakan oleh preferensi ukuran font pengguna.

Biasanya digunakan ketika mengatur ukuran font di dalam Aplikasi Android.

Dimension	Description	Units / Physical Inch	Density Independent	Same Physical Size on Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density independent pixels	~160	Yes	No
sp	Scale independent pixels	~160	Yes	No



### Jika kita simpulkan, satuan Dimensi di Android itu ...

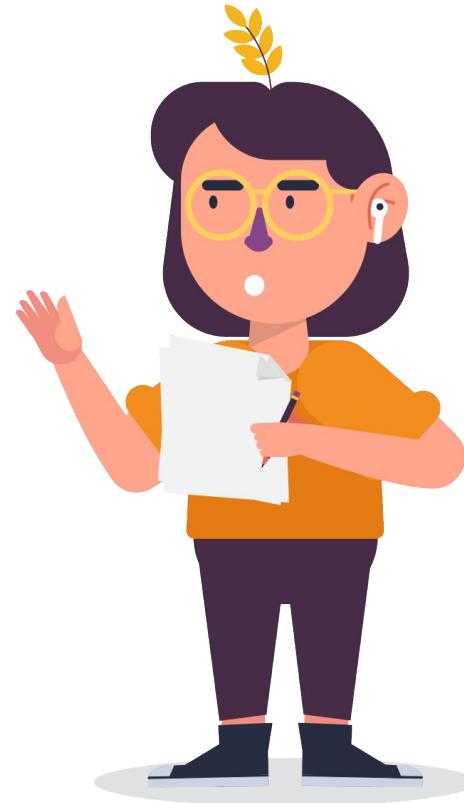
Bermanfaat untuk menghasilkan tampilan yang konsisten  
di perangkat Android.

Tapi kamu tahu nggak sih? ada 2 jenis satuan dalam  
dimensi Android, yaitu **dip/dp** dan **sp**.





- Satuan **sp** digunakan untuk **ukuran teks**. Perbedaannya dengan dp/dip adalah satuan sp android akan men-scale ukuran teks sesuai dengan setting ukuran teks di device masing-masing (yang biasa dapat diakses melalui menu settings)
- Satuan **dp/dip** digunakan untuk **satuan dari nilai dimensi**. Misalnya, width (**android:layout\_width**) dan height (**android:layout\_height**) dari sebuah komponen view atau viewgroup.





### Contoh Penggunaan Dimensi dalam Android

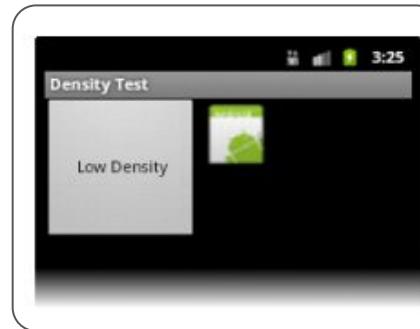
Semisal terdapat tiga buah tablet dengan resolusi yang tinggi.

- Tablet A memiliki resolusi layar 1920X2160px,
- Tablet B memiliki resolusi 2560X1440px, dan
- Tablet C memiliki resolusi 3840X2160.

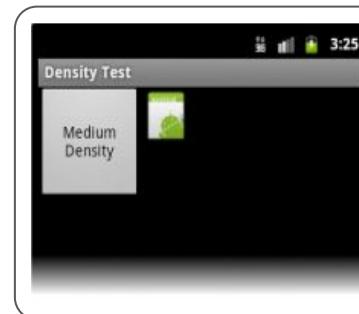
Kita masukkan sebuah button berukuran 500x500px dan terlihat normal pada tablet A.

Tetapi jika diperhatikan pada tablet B maupun C, maka button akan terlihat kecil bahkan terlalu kecil.

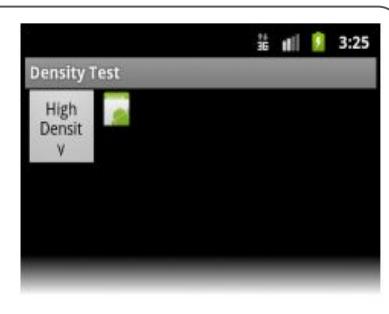
Button 500px X 500px



Tablet A



Tablet B



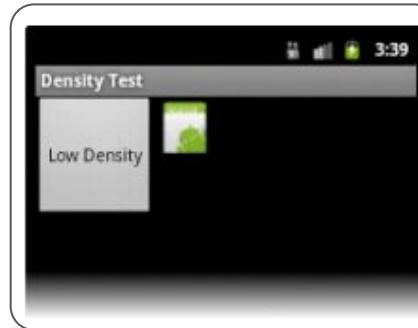
Tablet C



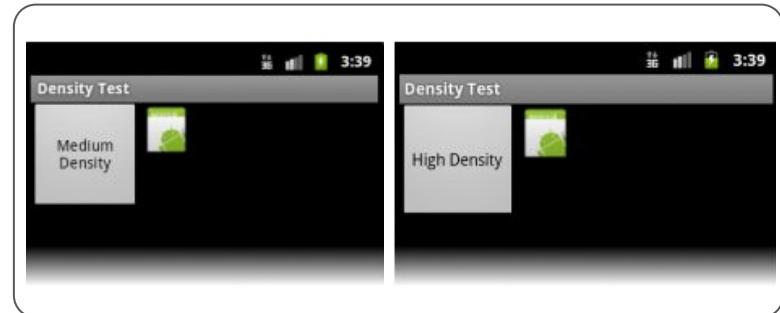
Beda lagi kalau kita menentukan ukuran button-nya menggunakan dp.

Bila kita menggunakan 500x500dp, maka button tersebut akan terlihat lebih baik pada beberapa device dengan ukuran yang berbeda.

**Button 500dp X 500dp**



**Tablet A**



**Tablet B**

**Tablet C**



### Contoh lainnya ...

Gambar ini menjelaskan bahwa ukuran 200dp akan dikonversi pada device dengan density mdpi (device dengan density 160dpi/dots per inch) menjadi 200px dan menjadi 400px pada device dengan density xhdpi (device dengan density 420dpi).



Density Bucket	Screen Density	Physical Size	Pixel Size
ldpi	120 dpi	0.5 x 0.5 in	0.5 in * 120 dpi = 60 x 60 px
mdpi	160 dpi	0.5 x 0.5 in	0.5 in * 160 dpi = 80 x 80 px
hdpi	240 dpi	0.5 x 0.5 in	0.5 in * 240 dpi = 120 x 120 px
xhdpi	320 dpi	0.5 x 0.5 in	0.5 in * 320 dpi = 160 x 160 px
xxhdpi	480 dpi	0.5 x 0.5 in	0.5 in * 480 dpi = 240 x 240 px



Dengan hal tersebut, ukuran tampak sama dan konsisten secara fisik, untuk berbagai device dengan ukuran layar yang berbeda.

**Catatan: Disarankan menggunakan berbagai macam ukuran image dengan beragam density. Hal ini akan mempermudah konversi image jika digunakan pada layar yang berbeda.**



Density Bucket	Screen Density	Physical Size	Pixel Size
ldpi	120 dpi	0.5 x 0.5 in	0.5 in * 120 dpi = 60 x 60 px
mdpi	160 dpi	0.5 x 0.5 in	0.5 in * 160 dpi = 80 x 80 px
hdpi	240 dpi	0.5 x 0.5 in	0.5 in * 240 dpi = 120 x 120 px
xhdpi	320 dpi	0.5 x 0.5 in	0.5 in * 320 dpi = 160 x 160 px
xxhdpi	480 dpi	0.5 x 0.5 in	0.5 in * 480 dpi = 240 x 240 px



### Sehingga, kesimpulannya ...

Disarankan menggunakan berbagai macam ukuran image dengan beragam density. Hal ini akan mempermudah konversi image jika digunakan pada layar yang berbeda.





Upcoming trip: Seattle

Jun 12-19 • 2 guests



Check In  
12:00 PM, Jun 12

Check Out  
11:00 AM, Jun 19

---

Weekly Mix

20 songs, 1hr 52 min



Autumn Fever  
Bascilica

Urban  
We Summit

Flicker  
Print Works

◀ ▶ ▶▶

Heart icons for favoriting each song entry.

## Terakhir nih, Kita Bahas Slicing ...

Dalam *User Interface (UI) Design*, Slicing adalah proses memotong-motong komposisi *layout* dalam *interface 2D* (*comp*) menjadi beberapa *file gambar* (aset digital).

Ini merupakan bagian dari proses pengembangan *client side* untuk halaman dan/atau situs web.

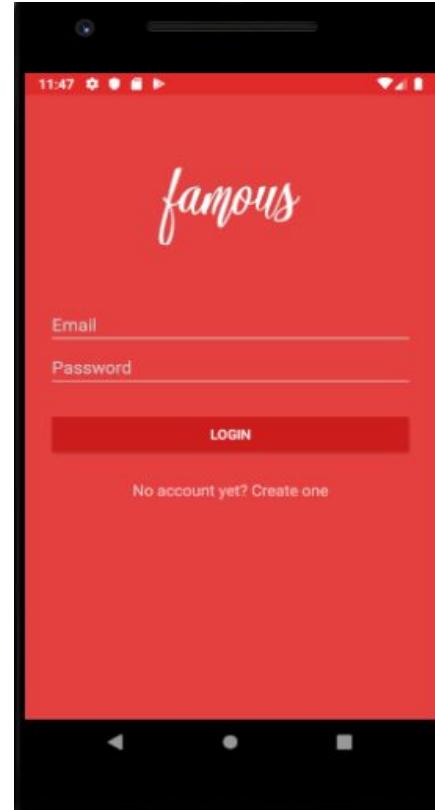
Meski begitu, proses Slicing juga digunakan dalam proses desain *user interface (UI)* dalam pengembangan software dan game.



**Udah banyak banget teknik layouting  
yang sudah kita bahas!**

**Sekarang, saat nya kita prak to the tik  
alias Praktik 🔥**

**Yuk, kita coba buat layout~**



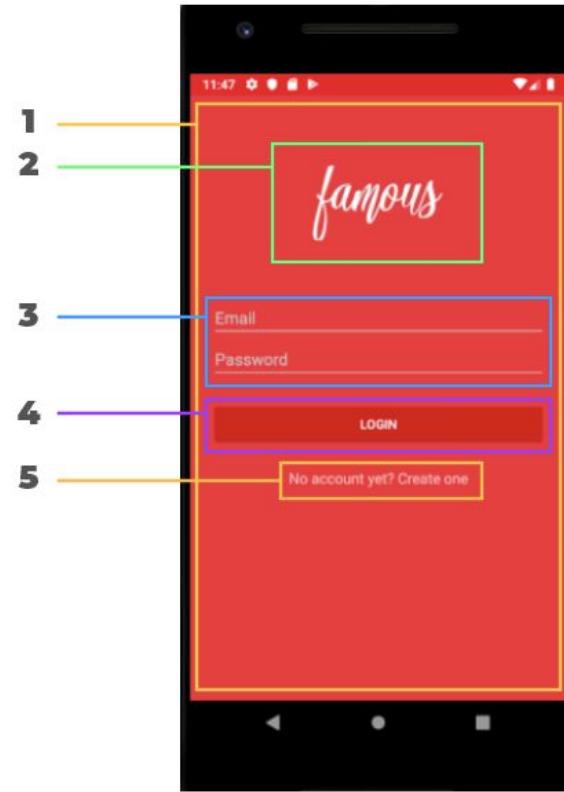
Sebelum membuat layout, kita membutuhkan **desain layout**.

Contohnya kita akan menggunakan desain layout berikut



Setelah mendapatkan desainya, kita harus melakukan *slicing* ke xml untuk screen dari androidnya. Kita akan menggunakan beberapa **View**, antara lain:

1. **LinearLayout** untuk membuat semua *View* di dalam *ViewGroup Horizontal*
2. **ImageView** untuk logo
3. **EditText** untuk input email dan password
4. **Button** untuk tombol Login
5. **TextView** untuk menampilkan label yang jika dipencet akan navigasi ke Signup





Setelah kita menentukan setiap komponen *View* dan *ViewGroup* yang akan digunakan dalam desain, kita bisa menuliskannya di **XML Layout**.



## Menerapkan Desain pada Layout

```
● ● ●

<resources>

    <!-- Base application theme. -->
    <style name="AppTheme">
        <parent>Theme.AppCompat.Light.DarkActionBar</parent>
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
        <item name="colorAccent">@color/accent</item>

        <item name="android:windowBackground">@color/primary</item>

        <item name="colorControlNormal">@color/iron</item>
        <item name="colorControlActivated">@color/white</item>
        <item name="colorControlHighlight">@color/white</item>
        <item name="android:textColorHint">@color/iron</item>

        <item name="colorButtonNormal">@color/primary_darker</item>
        <item name="android:colorButtonNormal">@color/primary_darker</item>
    </style>

    <style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
        <item name="colorAccent">@color/white</item>
        <item name="android:textColorPrimary">@color/iron</item>
        <item name="android:background">@color/primary</item>
    </style>

</resources>
```

```
<!-- Password Label -->
<EditText android:id="@+id/input_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Password"/>

<Button
    android:id="@+id/btn_login"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginBottom="24dp"
    android:padding="12dp"
    android:text="Login"/>

<TextView android:id="@+id/link_signup"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="No account yet? Create one"
    android:gravity="center"
    android:textSize="16dp"/>

</LinearLayout>
</ScrollView>
```

Sekarang, kita desain Layout XML sedemikian rupa sehingga sangat mirip dengan desain yang kita dapatkan.



Karena komponen View dalam LinearLayout itu cukup banyak, kita bisa menggunakan ScrollView sebagai Root.

Dengan begitu, pengguna bisa melakukan scroll untuk melihat semua komponen meskipun ukuran layar *smartphone* mereka kecil.

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark"
parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
        <item name="colorAccent">@color/accent</item>

        <item
name="android:windowBackground">@color/primary</item>

        <item name="colorControlNormal">@color/iron</item>
        <item name="colorControlActivated">@color/white</item>
        <item name="colorControlHighlight">@color/white</item>
        <item name="android:textColorHint">@color/iron</item>

        <item
name="colorButtonNormal">@color/primary_darker</item>
        <item
name="android:colorButtonNormal">@color/primary_darker</item>
    </style>

    <style name="AppTheme.Dark.Dialog"
parent="Theme.AppCompat.Dialog">
        <item name="colorAccent">@color/white</item>
        <item name="android:textColorPrimary">@color/iron</item>
        <item name="android:background">@color/primary</item>
    </style>

</resources>
```

```
<!-- Password Label -->
<EditText android:id="@+id/input_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Password"/>

<Button
    android:id="@+id/btn_login"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="24dp"
    android:layout_marginBottom="24dp"
    android:padding="12dp"
    android:text="Login"/>

<TextView android:id="@+id/link_signup"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="No account yet? Create one"
    android:gravity="center"
    android:textSize="16dp"/>

</LinearLayout>
</ScrollView>
```



Untuk membuat *Theme* yang berisikan variabel warna dalam app, kita mendefinisikannya pada file `/res/values/styles.xml`



Nah, sekarang kita siapkan style yang akan kita terapkan.

Kita menyiapkan style di styles.xml. Pada styles.xml ini, kita bisa mendefinisikan style tampilan untuk aplikasi kita secara keseluruhan, maupun untuk komponen-komponen tertentu.

Contohnya, kita bisa customize tampilan Dialog yang diadaptasi dari tema bawaan android. Coba perhatikan tag style yang terakhir!.

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
        <item name="colorAccent">@color/secondary</item>

        <item name="android:windowBackground">@color/primary</item>

        <item name="colorControlNormal">@color/iron</item>
        <item name="colorControlActivated">@color/white</item>
        <item name="colorControlHighlight">@color/white</item>
        <item name="android:textColorHint">@color/iron</item>

        <item name="colorButtonNormal">@color/primary_darker</item>
        <item name="android:colorButtonNormal">@color/primary_darker</item>
    </style>

    <style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
        <item name="colorAccent">@color/white</item>
        <item name="android:textColorPrimary">@color/iron</item>
        <item name="android:background">@color/primary</item>
    </style>

</resources>
```



Supaya makin omezing, kita bedah yuk code di styles.xml!

- **Atribut name:**

Digunakan untuk penamaan style yang kita buat. Nama setiap style nggak boleh sama yaa!

- **Atribut Parent:**

Digunakan untuk ‘menurunkan’ style dari style yang sudah ada / disiapkan sebelumnya.

- **Tag Item:**

Digunakan untuk mengubah style bawaan dari style parent. Atribut name dalam tag item ini bisa berbeda-beda ya tergantung parent yang digunakan.

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>

    <style name="AppTheme.Dark" parent="Theme.AppCompat.NoActionBar">
        <item name="colorPrimary">@color/primary</item>
        <item name="colorPrimaryDark">@color/primary_dark</item>
        <item name="colorAccent">@color/accent</item>
        <item name="android:windowBackground">@color/primary</item>
        <item name="colorControlNormal">@color/iron</item>
    </style>

    <style name="AppTheme.Dark.Dialog" parent="Theme.AppCompat.Dialog">
        ...
    </style>
</resources>
```

Nama Style

Nama Parent.  
Parent bisa dari Tema  
bawaan dari Android.

Nama Style  
Dengan  
Sub nama

Atribut untuk  
meng-custom  
elemen



Untuk menggunakan theme dari nilai *values* terhadap sebuah *screen* (yaitu *main activity*), kita dapat memanggil dari file **AndroidManifest.xml**

```
● ● ●

<application
    ...
    <activity android:name=".SplashScreen">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity android:name=".HomeActivity" />

    <!-- Begini cara memasangnya -->
    <activity android:name=".MainActivity"
        android:theme="@style/AppTheme.Dark">
    </activity>
</application>
```

# Saatnya kita Quiz!





**1. Suatu tampilan tata letak di Android yang berguna untuk mengatur penempatan teks, gambar, ataupun komponen lainnya sehingga tampilan pada aplikasi yang kita buat terlihat rapi, dan cocok untuk dilihat oleh kita maupun user**  
...

- A. Layout
- B. XML
- C. UI/UX



- 1. Suatu tampilan tata letak di Android yang berguna untuk mengatur penempatan teks, gambar, ataupun komponen lainnya sehingga tampilan pada aplikasi yang kita buat terlihat rapi, dan cocok untuk dilihat oleh kita maupun user ...**
- A. Layout
  - B. XML
  - C. UI/UX

**Yap betul!. Pilihan yang tepat adalah Layout!.**

**Layout adalah suatu tampilan tata letak di Android yang berguna untuk mengatur penempatan teks, gambar, ataupun komponen lainnya sehingga tampilan pada aplikasi yang kita buat terlihat rapi, dan cocok untuk dilihat oleh kita maupun user.**



## 2. Berikut ini, pernyataan yang benar terkait ViewGroup adalah...

- A. Sebuah tempat untuk membuat layout.
- B. Menyediakan wadah yang tidak terlihat untuk menampung View lain maupun ViewGroup itu sendiri.
- C. Tempat yang mengelompokkan View.



## 2. Berikut ini, pernyataan yang benar terkait **viewGroup** adalah...

- A. Sebuah tempat untuk membuat layout.
- B. Menyediakan wadah yang tidak terlihat untuk menampung View lain maupun ViewGroup itu sendiri.
- C. Tempat yang mengelompokkan View.

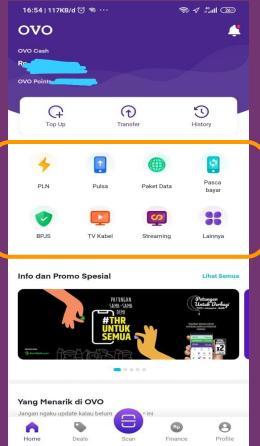
Yap benar banget!.

**ViewGroup** akan menyediakan wadah yang tidak terlihat untuk menampung View lain atau ViewGroup dan untuk menentukan layout properties. Misalnya, **LinearLayout** adalah bagian dari ViewGroup yang berisi kontrol UI seperti **Button**, **TextView**, dll. Dan layout lainnya juga.



**3. ViewGroup apa yang paling cocok untuk membuat bagian yang ditandai?**

- A. LinearLayout
- B. SquareLayout
- C. GridLayout



### 3. ViewGroup apa yang paling cocok untuk membuat bagian yang ditandai?

- A. LinearLayout
- B. SquareLayout
- C. GridLayout

Yap betul! Seperti yang dapat kita lihat, bahwa bagian yang ditandai itu berbentuk Grid / Kotak-kotak, sehingga paling cocok adalah menggunakan GridLayout. Sedangkan untuk LinearLayout, bisa juga membentuk Grid, namun pasti membuat NestedLayout yang tidak direkomendasikan karena bisa membuat lag tampilan.



**4. Dimensi yang diskalakan oleh preferensi ukuran font pengguna. Biasanya digunakan ketika mengatur ukuran font di dalam Aplikasi Android ...**

- A. sp
- B. dp
- C. cm

**4. Dimensi yang diskalakan oleh preferensi ukuran font pengguna. Biasanya digunakan ketika mengatur ukuran font di dalam Aplikasi Android ...**

- A. sp
- B. dp
- C. cm

Yap, benar banget!. Jawabannya sp adalah dimensi yang diskalakan oleh preferensi ukuran font pengguna. Biasanya digunakan ketika mengatur ukuran font di dalam Aplikasi Android. Berbeda dengan dp yang menyesuaikan ukuran dengan kerapatan layar.



**5. ViewGroup yang digunakan untuk menampilkan item dalam bentuk kotak-kotak dua dimensi (baris & kolom) ...**

- A. ListView
- B. SquareView
- C. GridView



5. ViewGroup yang digunakan untuk menampilkan item dalam bentuk kotak-kotak dua dimensi (baris & kolom) ...

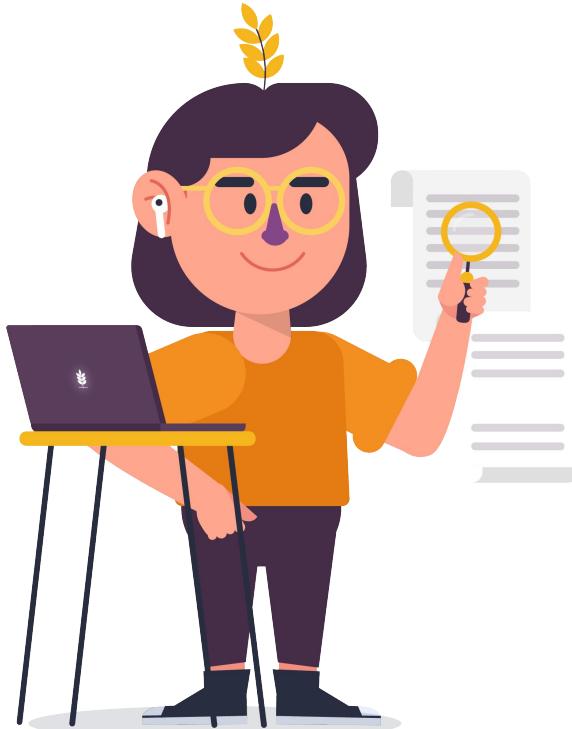
- A. ListView
- B. SquareView
- C. GridView

Yuhuu~! Jawabannya adalah GridView.

GridView dapat menampilkan daftar item dengan tata letak kotak-kota dua dimensi yes~

## Referensi dan bacaan lebih lanjut~

1. [Android UI Layouts Tutorial with Examples](#)
2. [Build a Responsive UI with ConstraintLayout | Android Developers](#)
3. [Android Unit | Muslim Developer](#)
4. [Tata letak | Developer Android](#)
5. [Relative Layout | Android Developers](#)
6. [Table | Android Developers](#)
7. [FrameLayout | Android Developers](#)
8. [Linear Layout | Android Developers](#)
9. [ScrollView | Android Developers](#)
10. [Slices | Android Developers](#)





**Nah, selesai sudah pembahasan kita di Chapter 1 Topic 4 ini.**

**Selanjutnya, kita bakal belajar **Version Control Git**.**

**Penasaran kayak gimana? Cus langsung ke topik selanjutnya~**



# Terima Kasih!



Next Topic

loading...