



**MODUL PEMROGRAMAN MOBILE  
CIM 430**

**MODUL PERTEMUAN 3  
PERANCANGAN MODEL LAYOUT PADA APLIKASI ANDROID**

**DISUSUN OLEH  
7174 – SAWALI WAHYU, S.KOM, M.KOM**

Universitas  
**Esa Unggul**

**UNIVERSITAS ESA UNGGUL  
FAKULTAS ILMU KOMPUTER  
TAHUN 2021**

## PERANCANGAN MODUL LAYOUT PADA APLIKASI ANDROID

### A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

- 1) Mahasiswa mampu melakukan *refreshment* mengenai sesi sebelumnya yaitu sesi 2, dimana konsep Views sudah mulai dikenalkan .
- 2) Mahasiswa mampu memahami tipe-tipe layout yang disediakan oleh Android, yaitu layout *linear*, relatif dan *constraint*.
- 3) Mahasiswa mampu menerapkan layout-layout tersebut menjadi aplikasi sederhana

### B. Uraian dan Contoh

A) Pendahuluan

B) Konsep Layout

C) Layout Linear

D) Layout Relatif

E) Layout Constraint



## ***DESIGN LAYOUT MODEL IN ANDROID APPLICATION***

### **1. Pendahuluan**

Pada sesi sebelumnya, anda telah mengenali dasar-dasar lingkungan Android Studio dan sistem operasi Android. Selain itu anda juga telah diperkenalkan beberapa *skills* berikut :

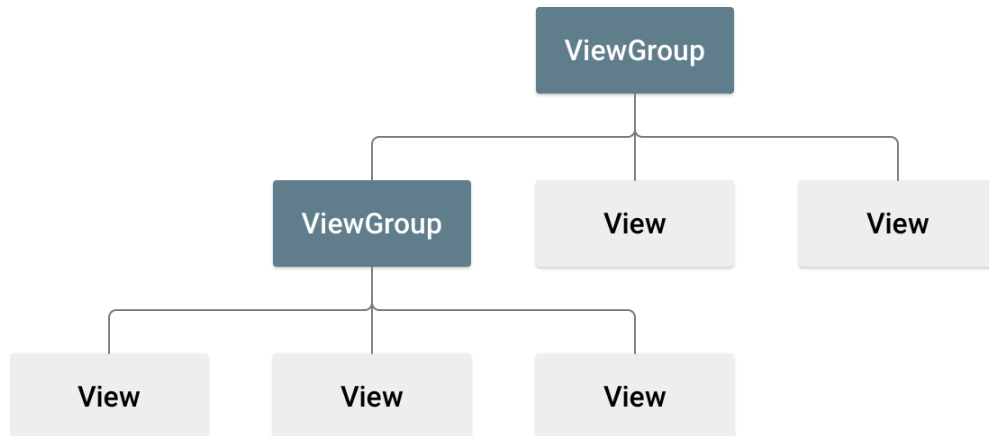
- 1) Membuat aplikasi *Hello World* dengan Android Studio
- 2) Menjalankan aplikasi di *smartphone* atau di *emulator*
- 3) Menerapkan *TextView* (lihat lagi kode pada file *activity-main.xml*) dalam layout aplikasi *Hello World*.

Adapun pada sesi ini, anda kan berlatih membuat beberapa tata letak yang disediakan oleh editor layout didalam Android Studio. *Layout* yang akan dibahas adalah layout *Linear*, layout *Relatif*, dan layout *Constraint*.

### **2. Konsep Layout**

Seperti yang telah dijelaskan sebelumnya, terdapat sekitar 3 juta *apps* yang tersedia di *Google Play Store*. Untuk memenangkan kompetisi agar calon *user* tertarik untuk men-*download* dan menggunakan aplikasi anda, maka aplikasi harus memiliki layout yang menarik dan intuitif.

Layout mendefinisikan bagaimana pengguna akan berinteraksi dengan aplikasi anda. Semua elemen aplikasi dibangun berdasarkan hierarki *View* ataupun *ViewGroup*. *View* merupakan setiap elemen aplikasi dalam layout aplikasi Android. Sedangkan *ViewGroup* adalah wadah penampung *View* yang fungsinya untuk mendefinisikan struktur tata letak setiap elemen aplikasi (lihat gambar dibawah ini).



Gambar 1. Layout View

Android menyediakan berbagai varian View seperti berikut :

- 1) TextView – untuk menampilkan label teks
- 2) ImageView - untuk menampilkan gambar
- 3) Button – tombol yang dapat diklik memicu aktifitas tertentu
- 4) ImageButton - untuk menampilkan gambar yang dapat diklik
- 5) EditText - bidang teks yang dapat diedit oleh user
- 6) ListView – untuk menampilkan daftar item yang dapat di-scroll

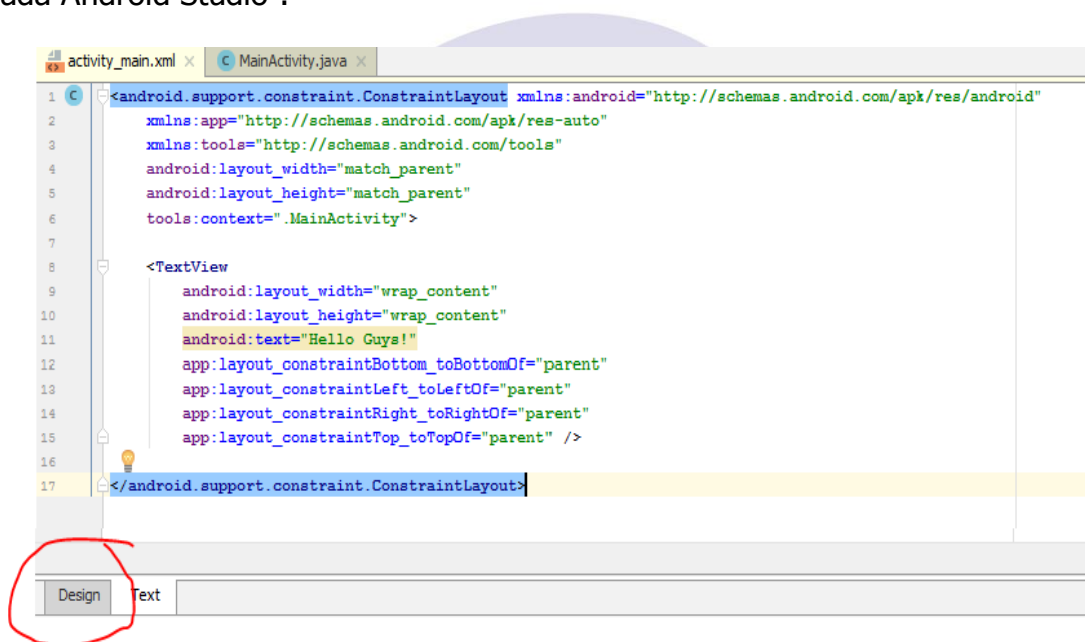
Views yang disediakan Android perlu diatur tata letaknya agar menghasilkan layout yang memudahkan user dalam menggunakan aplikasi. Untuk membuat layout, Android Studio menyediakan beberapa opsi

- 1) Secara grafis menggunakan Editor Layout
- 2) Menggunakan kode yaitu XML
- 3) Secara Programmatically

Mari mulai membuat layout, pertama-tama buatlah proyek baru yaitu Empty Activity seperti yang telah dijelaskan pada modul sesi 2.

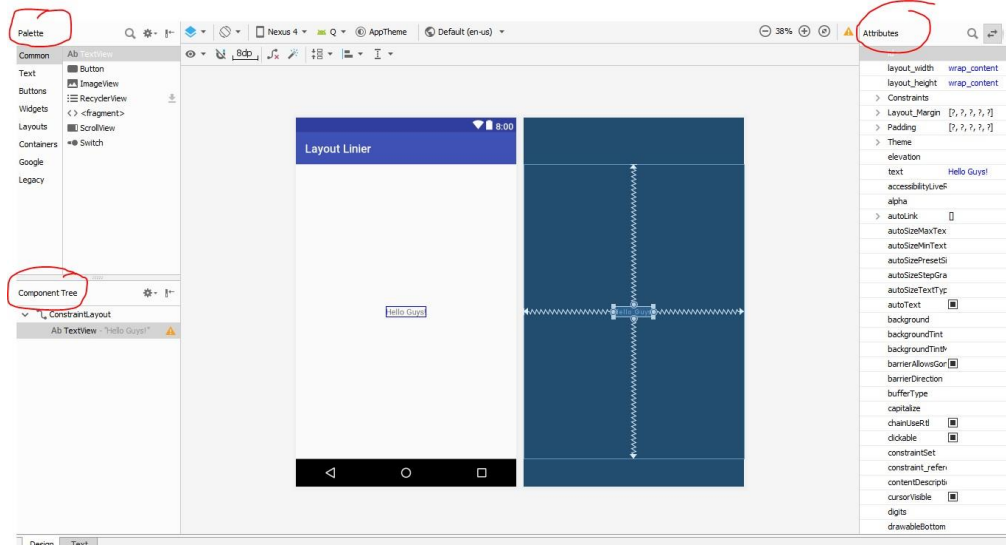
## Membuat Layout secara grafis dengan Editor Layout

Pilihan pertama dalam membuat layout adalah dengan menggunakan editor layout. Disini anda dapat mengedit dan preview (melihat) aplikasi anda secara langsung tanpa harus melakukan kompilasi kode menjadi APK dan menjalankannya di smartphone/emulator. Editor Layout akan muncul secara otomatis ketika anda membuka file XML. Berikut tampilan dari editor layout pada Android Studio :



Gambar 2. Membuat Editor Layout

- Untuk memunculkan editor layout, klik tab *Design* (lihat tab kiri-bawah).  
Note: pada versi yang terbaru, terdapat sedikit perbedaan untuk menampilkan editor layout.
- Untuk menambahkan *Views* pada editor layout, pilih elemen yang tersedia pada *Palette* (klik *Palette* jika belum terbuka, lihat gambar dibawah) dengan cara menariknya dari *palette* kepada layer aplikasi.



Gambar 3. Memetakan Layout

Alat bantu visual (seperti pada desain *Blueprint* sebelah kanan) membantu anda untuk memposisikan elemen secara relatif satu sama lain. Anda dapat mengedit atribut elemen disebelah kanan editor layout, ataupun memindahkan posisi relatif menggunakan *ComponentTree* disebelah kiri-bawah. Selain itu, anda juga dapat mengubah ukuran elemen dengan menarik bagian tepinya. Namun sebagian developers merasa pembuatan layout akan lebih terkontrol jika menggunakan kode XML mentahnya.

## Membuat Layout dengan kode XML



Gambar 4. Layout pada XML

Untuk membuat layout menggunakan kode XML, maka anda harus berpindah dari mode *Design* ke mode *Text*, tekan tab *Text* pada bagian kiri-bawah.

Pada file tersebut terdapat beberapa elemen yang perlu kita pelajari, yaitu :

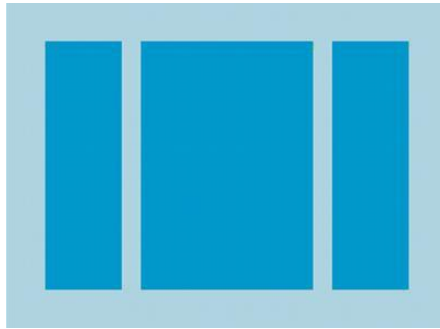
- A) Element pertama (baris pertama) adalah deklarasi XML dan encoding yang digunakan, yaitu `<?xml version="1.0" encoding="utf-8"?>`. Note: elemen ini hanya untuk memastikan, sekiranya dihilangkan aplikasi masih tetap berjalan.
- B) Elemen kedua terdiri (baris kedua) dari deklarasi tipe layout dan *namespace* Android. Pada contoh diatas, tipe layout adalah layout *constraint*.
- C) Elemen berikutnya adalah *View*. Pada contoh diatas elemen *View*-nya adalah *TextView*.
- D) Setiap *View* memiliki atribut-atribut khusus, misalnya, *TextView* memiliki atribut *textSize* untuk mengatur ukuran *text*. Adapun atribut umum yang dimiliki semua *View* adalah atribut *id* untuk memberikan *identifier* (referensi nama) dan atribut tata letak yang menggambarkan orientasi ruang.

**Note:**

Anda juga dapat membuat layout saat *runtime* saat aplikasi berjalan. Jika anda tertarik silahkan pelajari lagi atau coba lihat tutorial di <https://www.android-examples.com/android-create-vertical-linearlayout-programmatically/>.

### 3. Layout Linear (Layout yang biasa digunakan *Developers*)

*LinearLayout* adalah wadah untuk menyelaraskan beberapa *Views* dalam arah yang sama, baik arah vertikal (tegak) atau horizontal (mendatar). Arah layout *linear* ini dapat diatur dengan mengubah nilai pada atribut `android:orientation`. Ilustrasi layout linear secara horizontal diberikan sebagai berikut :



Semua *View* didalam *LinearLayout* diatur letaknya satu demi satu, sehingga jika kita pilih mode horizontal maka views akan menjadi satu baris. *LinearLayout* menyediakan atribut `android:layout_weight` untuk penetapan *weight* (bobot) relatif bagi setiap *View*. Atribut ini mengatur berapa ruang yang harus ditempati dilayar. Nilai *weight* yang lebih besar memungkinkannya sebuah *View* untuk mengisi ruang yang tersisa di tampilan wadahnya. Secara *default*, nilai *weight* adalah nol.

### Tata Letak yang Rata

Untuk menghasilkan tata letak *linier* yang rata, dimana setiap *View* menempati ruang yang sama besarnya, maka lakukan setingan berikut ini

- 1) Set nilai `android:layout_height` setiap *View* menjadi "0dp" (untuk tata letak vertikal)
- 2) Atau set `android:layout_width` setiap *View* menjadi "0dp" ( untuk tata letak horizontal).
- 3) Kemudian atur `android:layout_weight` dari setiap *View* menjadi "1".

Note:

Anda juga dapat membuat tata letak linier yang tidak rata. Misalkan ada tiga *View* dan bobot dari dua *View* yang pertama diset 1 dan view ketiga tidak diset (*default* bobot adalah 0). Setingan ini menjadikan *View* ketiga hanya menempati area yang diperlukan oleh kontennya. Sedangkan, dua *View* awal akan mengisi seluruh ruang lainnya.



Mari menerapkan konsep layout *linear* ini. Pertama-tama, ubahlah bagian deklarasi layout *constraint* pada baris kedua file *activity\_main.xml* dari proyek anda, yaitu `</android.support.constraint.ConstraintLayout>`, menjadi deklarasi layout *linear*.

Note : jangan mengubah bagian deklarasi *namespace*-nya.

Kemudian hilangkan *TextView* dan ganti dengan 4 *Views* sesuai kode berikut 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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Gambar 5. Layout Text

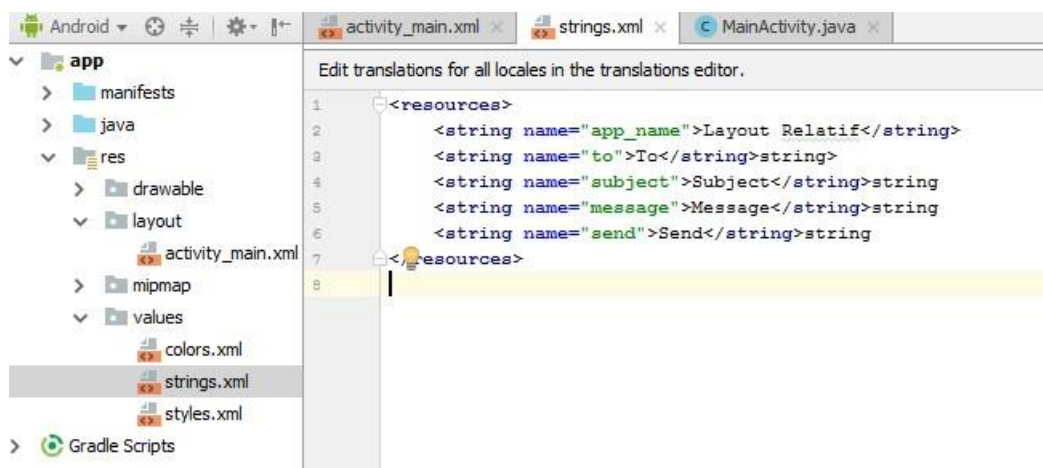
Kode diatas menunjukkan bagaimana weight bekerja pada sebuah aktivitas "kirim pesan" yang View-nya diatur secara vertikal. Terdapat tiga View yaitu bidang To, baris Subjek, dan tombol Send, masing-masing hanya mengambil ketinggian yang mereka butuhkan.

Pada kode tersebut terdapat 4 Views, yaitu 3 EditText dan 1 Button. Pada EditText ketiga, nilai atribut `android:layout_weight` diset menjadi "1" (dengan demikian bobot relatifnya menjadi lebih besar dari Views lainnya yang secara default nilai 0). Dengan cara ini, EditText ketiga akan mengisi seluruh ruang yang tersisa pada layout tersebut. Namun untuk lebih memastikan hal ini, maka atribut `android:layout_height` diset menjadi "0dp" agar tidak crash (bentrok) dengan atribut `weight`. Konfigurasi ini memungkinkan EditText ketiga yaitu bagian pesannya (Message) untuk mengambil sisa ruang yang ada.

Satu atribut lagi yang perlu diperhatikan pada EditText ketiga adalah `android:layout_gravity`. Atribut ini mengatur bagaimana view diletakkan terhadap wadahnya. Untuk memastikannya berada tepat setelah EditText kedua (baris Subjek) kedua, maka set atribut `android:layout_gravity` menjadi "top". Silahkan pahami setingan dari Views lainnya.

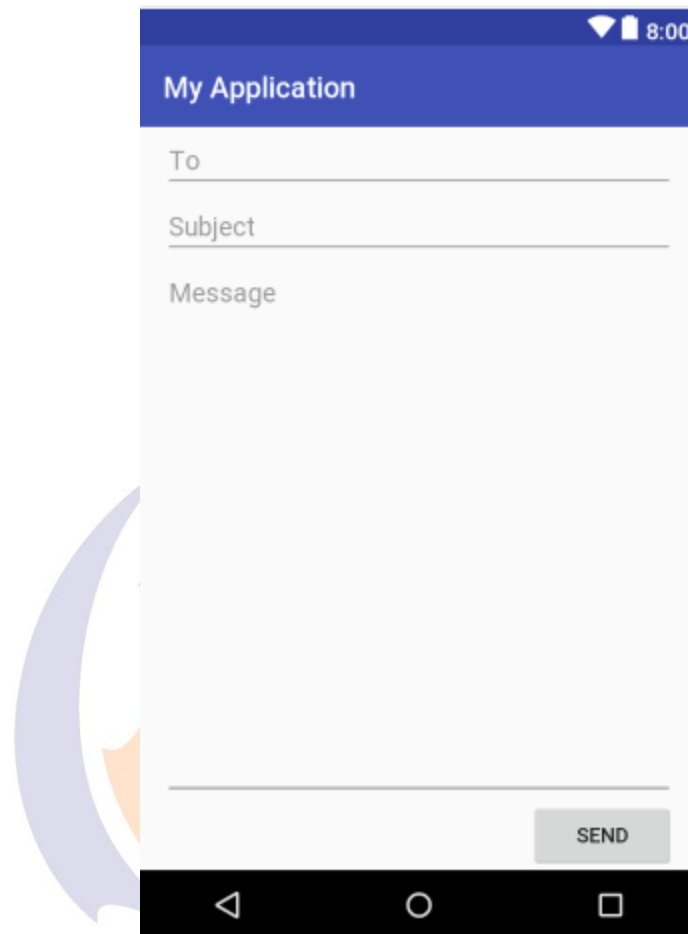
## Penggunaan String Global

Saat ini jika aplikasi dijalankan, maka IDE akan menampilkan *error* karena nilai atribut `android:hint` dan `android:text` belum terdefinisi. Oleh karena itu, bukalah file *strings.xml* pada folder *app/res/values*. Definisikan setiap variabel string sebagai berikut :



Note: anda bisa saja tidak menggunakan string global seperti langsung memberi nilai string pada atribut yang ada didalam file *activity\_main.xml*. Namun, cara ini dapat mengakibatkan nilai atribut string tidak konsisten ketika

layout anda semakin bertambah. Sekarang, jika anda run aplikasi layout linear ini maka, akan didapatkan UI layout aplikasi sebagai berikut :



### Kelebihan Layout Linear

Seringkali, *developers* memerlukan tampilan yang memiliki varian *Views* yang mendatar maupun yang horizontal. Pada layout linear hal ini dapat dicapai dengan membuat layout linear di dalam layout linear. Misalkan seperti pada layout diatas utamanya menggunakan layout vertikal. Jika kita ingin menambahkan tombol di sebelah tombol *Send* maka kita dapat mendeklarasikan layout dengan orientasi horizontal setelah *EditText* ketiga yaitu bagian *Message*. Pada layout horizontal ini, kita dapat memasukkan 2 *Button* yang akan tersusun secara mendatar. Namun demikian cara ini mengakibatkan waktu loading tampilan menjadi lebih lambat.

#### 4. Layout Relatif

Berbeda halnya dengan LinearLayout, Layout Relatif (RelativeLayout) adalah wadah layout yang mengatur Views secara relatif satu sama lain. Posisi setiap View dapat ditentukan sebagai relatif terhadap View saudaranya (seperti di sebelah kiri atau di bawahnya). Atau pun pada posisi yang relatif terhadap area wadah induknya (seperti disejajarkan dengan paling bagian bawah, kiri atau tengah). Ilustrasi layout relatif diberikan sebagai berikut :



Dengan RelativeLayout kita tidak memerlukan deklarasi layout didalam layout seperti pada LinearLayout. Hal ini juga tentunya akan memudahkan developer karena struktur kode menjadi lebih mudah dipahami.

Beberapa dari atribut di RelativeLayout dengan posisi relatif terhadap wadahnya adalah :

- a) Atribut `android:layout_alignParentTop`. Jika bernilai "true" maka posisi View akan berada pada bagian paling atas sesuai wadahnya.
- b) Atribut `android:layout_centerVertical` Jika bernilai "true" maka posisi View akan berada ditengah wadah secara vertikal.
- c) Atribut `android:layout_below` akan memposisikan View dibawah View lain berdasarkan ID-nya.
- d) Atribut `android:layout_toRightOf` memposisikan View disebelah kanan View lain berdasarkan ID-nya.

Sebagai contoh, anda dapat menyatakan bahwa sebuah View diposisikan di sebelah kiri View lainnya. Contoh di bawah ini menunjukkan skenario tersebut.

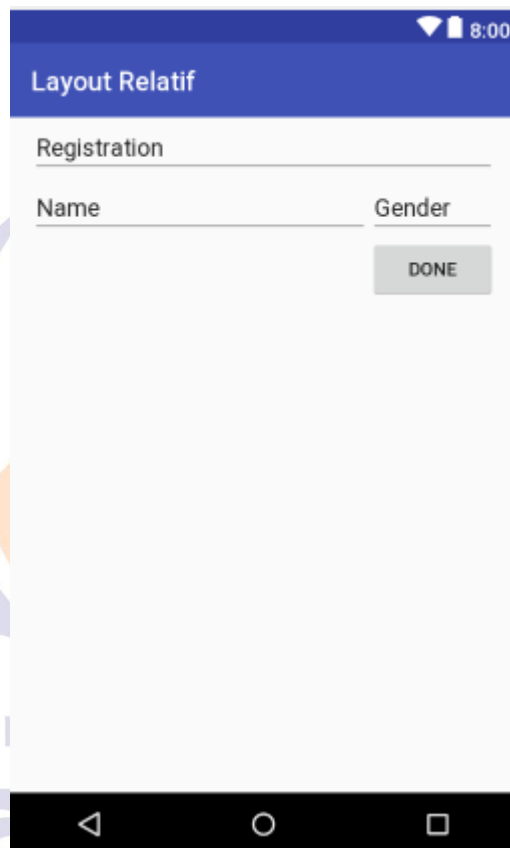
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/register" />
    <EditText
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/register"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/gender"
        android:text="@string/name" />
    <EditText
        android:id="@id/gender"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/register"
        android:layout_alignParentRight="true"
        android:text="@string/gender" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/gender"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

Pada kode tersebut terdapat 4 Views, yaitu 3 EditText dan 1 Button. Perhatikan setiap EditText ini diberikan ID (sebagai penanda/identifier), misal pada yang pertama yaitu `android:id="@+id/register"`, agar view lain dapat mengacu kepada View ini. EditText kedua memiliki 3 atribut relatif. Atribut `android:layout_below="@id/register"` menyatakan bahwa View ini berada dibawah View `id/register`. Atribut `android:layout_alignParentLeft="true"` menyatakan View ini akan berada pada bagian kiri wadah. Adapun atribut `android:layout_toLeftOf="@+id/gender"` menyatakan View ini berada pada

sebelah kiri View id/gender. Demikian pula halnya dengan EditText ketiga dan Button. Mereka masing-masing memiliki posisi relatif terhadap View lainnya.

**Note :**

jangan lupa mendefinisikan setiap string pada file strings.xml di folder app/res/values. Sebagai latihan, atur nilai-nilai string tersebut agar menghasilkan UI layout aplikasi sebagai berikut :

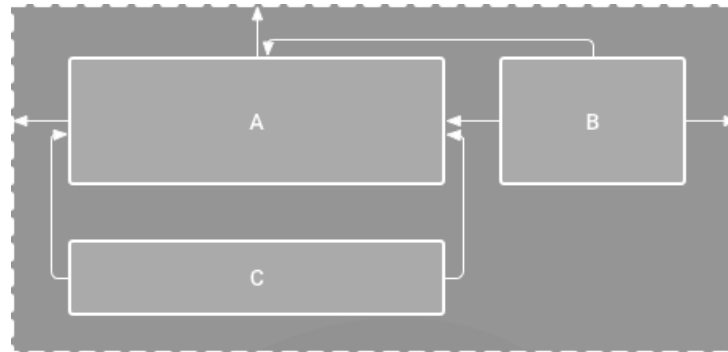


## 5. Layout Constraint

Sebagian developers mencukupkan diri mereka dengan Layout Relatif. Namun, sebenarnya Google telah menyediakan layout constraint, yang jauh lebih fleksibel dan juga mudah digunakan pada Editor Layout.

Untuk menentukan posisi tampilan di ConstraintLayout, anda harus menambahkan setidaknya satu acuan dari View lain ataupun dari wadah

induknya. Layout constraint menjadikan setiap View memiliki constraint (seperti tali) yang bisa diatur elastisitas dan marginnya. Ilustrasi layout constraint diberikan sebagai berikut :



Silahkan anda pelajari dan coba layout constraint sendiri. Info lebih lengkap bisa didapatkan di <https://developer.android.com/training/constraint-layout/index.html>.

#### **Note :**

Selain layout *constraint*, Android juga menyediakan layout lainnya seperti *GridView*, *FrameLayout*, dan *TableLayout*.

#### **KESIMPULAN**

Sesi ini telah menjelaskan beberapa teknik *layouting* yang biasa digunakan untuk mengatur tata letak pada sebuah aplikasi Android, yaitu layout *linier*, layout relatif, dan layout *constraint*. Selain itu, sesi ini telah menerapkan beberapa *Views* yaitu *TextView*, *EditText* dan *Button* yang sering digunakan pada aplikasi Android. Pada sesi berikutnya, anda akan mempelajari bagaimana melakukan perpindahan antara 2 aktifitas, dimana masing-masing aktifitas memiliki layout yang berbeda.