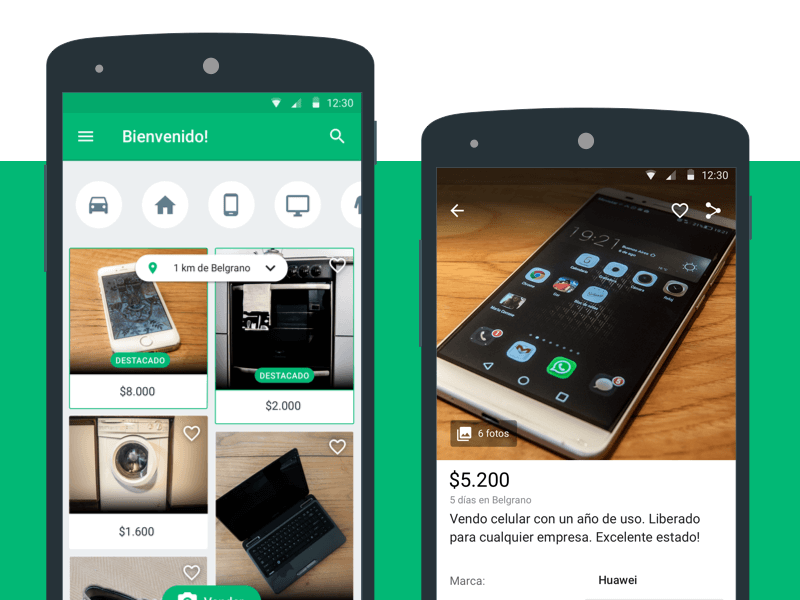
**Views dan ViewGroup**

**Teori**

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



Keren bukan? Baik, kita mulai saja pembahasannya.

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

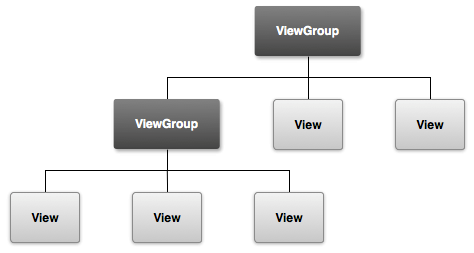
Sebuah view adalah obyek yang menggambar komponen tampilan ke layar yang mana pengguna dapat melihat dan berinteraksi langsung. Contoh komponen turunan dari view seperti :

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

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

* **LinearLayout**
* **FrameLayout**
* **RelativeLayout**
* **TableLayout**

Hierarki komponen view dan viewgroup dapat digambarkan dengan diagram berikut:



Jika diterjemahkan di dalam sebuah viewgroup akan ditampung dua buah komponen view dan satu komponen viewgroup yang terdiri dari 3 buah komponen view.

Anda dapat membaca tautan berikut untuk membaca lebih lanjut mengenai materi di atas: [User Interface Overview](https://developer.android.com/guide/topics/ui/overview.html)

Salah satu contoh dari tampilan dalam file layout xml untuk merepresentasikan kolaborasi view dan viewgroup seperti ini :

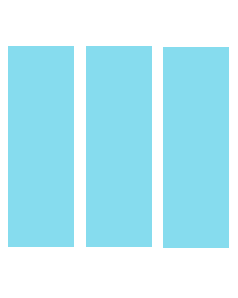
|  |
| --- |
| 1. <?xml version="1.0" encoding="utf-8"?> 2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" 3. android:layout\_width="match\_parent" 4. android:layout\_height="match\_parent" 5. android:orientation="vertical" > 6. <TextView android:id="@+id/text" 7. android:layout\_width="wrap\_content" 8. android:layout\_height="wrap\_content" 9. android:text="I am a TextView" /> 10. <Button android:id="@+id/button" 11. android:layout\_width="wrap\_content" 12. android:layout\_height="wrap\_content" 13. android:text="I am a Button" /> 14. </LinearLayout> |

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

Berikut adalah definisi singkat dan inti dari komponen-komponen di atas terhadap penempatan komponen view (child) di dalamnya:

#### **LinearLayout**

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

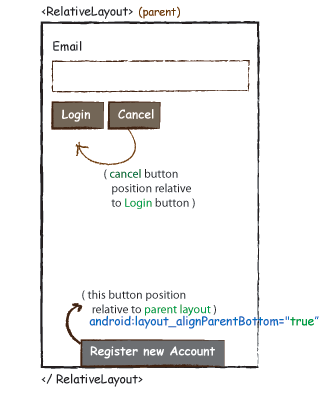


            android:orientation=”vertical”                      android:orientation=”horizontal”

Anda dapat membaca lebih lanjut mengenai linearlayout pada tautan berikut: [Linear Layout](https://developer.android.com/guide/topics/ui/layout/linear.html)

#### **RelativeLayout**

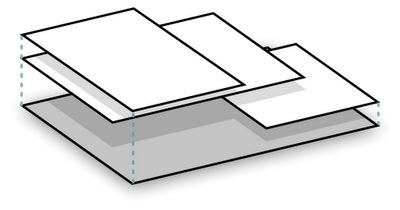
Layout yang paling fleksible dikarenakan posisi dari masing-masing komponen di dalamnya dapat mengacu secara relatif pada komponen yang lainnya dan juga dapat mengacu secara relatif ke batas layar.



Anda dapat membaca lebih lanjut mengenairelativelayoutpada tautan berikut: [Relative Layout](https://developer.android.com/guide/topics/ui/layout/relative.html)

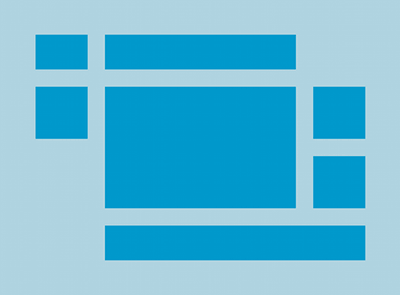
#### **FrameLayout**

Layout ini adalah layout yang paling sederhana. Layout ini akan membuat komponen yang ada di dalamnya menjadi menumpuk atau saling menutupi satu dengan yang lainnya.  
Komponen yang paling 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. Berikut ilustrasi penggunaan framelayout terhadap child view yang dimilikinya:



Anda dapat membaca lebih lanjut mengenaiframelayoutpada tautan berikut: [Frame Layout](http://developer.android.com/reference/android/widget/FrameLayout.html)

#### **TableLayout**



Susunan komponen di dalam tablelayout akan berada dalam baris dan kolom. Namun layout jenis ini tidak akan menampilkan garis pembatas untuk baris, kolom atau cell-nya. Anda dapat membaca lebih lanjut mengenaitablelayoutpada tautan berikut: [TableLayout](https://developer.android.com/guide/topics/ui/layout/grid.html)

Salah satu perbedaan mendasar dari keempat jenis layout di atas adalah bagaimana komponen view di dalamnya tersusun.

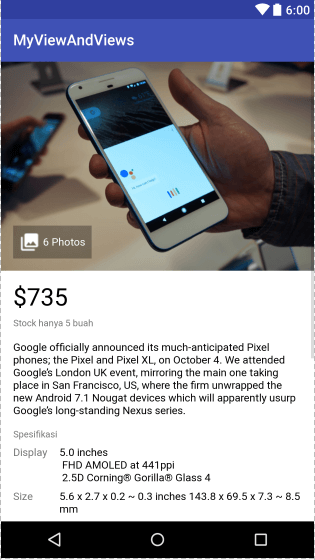
Kapan saatnya menggunakan linearlayout, relativelayout, framelayout dan tablelayout? Jawabannya tergantung pada kebutuhan dan performa aplikasi yang dibuat.

Salah satu tips performa untuk aplikasi Android adalah dengan menyusun hierarki layout dalam sebuah file layout xml secara sederhana dan flat. Artinya, kita tidak membuat layout dengan jumlah kedalaman yang tinggi. Selain itu, penting juga bagi kita untuk memahami dasar pembangunan antar muka, pengalaman pengguna ketika menggunakannya, dan selalu mencari pendekatan terbaik dalam penggunaan komponen antar muka.

Semua tergantung latihan dan seberapa sering kita mengubah mockup menjadi tampilan sebenarnya, terutama dengan memaksimalkan penulisan kode antara muka di berkas xml.

Untuk memahami lebih lanjut mengenai layout, Anda dapat membacanya pada tautan berikut: [Declaring Layout](https://developer.android.com/guide/topics/ui/declaring-layout.html)

**Contoh Design Layout**



**Contoh Codingan Layout**



Ketika aplikasi memuat informasi yang banyak dan melebihi ukuran layar, maka Anda membutuhkan tampilan yang memungkinkan pengguna untuk membaca informasi dengan lengkap. Kita membutuhkan komponen scrollview untuk mengatasi masalah ini.

#### **ScrollView**

#### Adalah layout yang memungkinkan komponen di dalamnya digeser (scroll) secara vertikal dan horizontal. Komponen di dalam scrollview hanya diperbolehkan memiliki 1 *parent* utama dari linearlayout, relativelayout, framelayout, atau tablelayout. Anda dapat membaca lebih lanjut mengenaiScrollViewpada tautan berikut [ScrollView](https://developer.android.com/reference/android/widget/ScrollView.html)

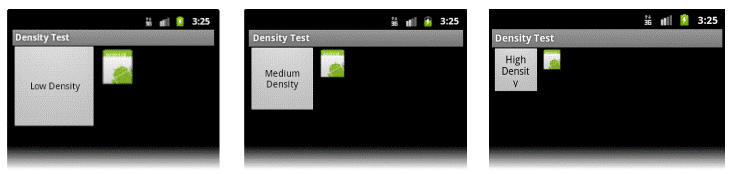
#### **Satuan Dimensi di Android**

Platform Android dikenal karena keberagamannya. Mulai ukuran perangkatnya, layar, spesifikasi, hingga level operating system-nya. Karena keberagaman tersebut, tak heran jika dibutuhkan tampilan yang konsisten agar aplikasi kita bisa berjalan dan tampil maksimal.

Android sendiri memiliki satuan unit dimensi untuk ukuran tinggi dan lebar sebuah komponen view atau viewgroup. Berikut adalah esensi dari satuan dimensi unit di android.

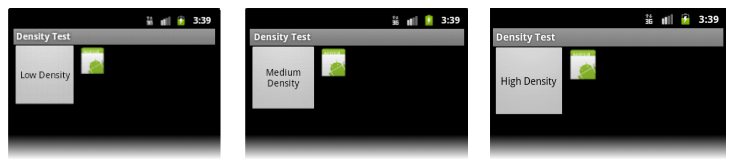
1. Ekosistem Android dikenal dengan fragmentasi spesifikasi perangkat yang sangat bervariasi. Beragam perangkat Android memiliki perbedaan dimensi layar dan kerapatan pixel (density).
2. Untuk tampilan yang konsisten di perangkat Android, terdapat 2 jenis satuan, yaitu dip/dp (density-independent pixel) dan sp (scale-independent pixels).
3. Satuandp/dip digunakan untuk satuan dari nilai dimensi misal width (attribut : layout\_width) dan height (attribut : layout\_height) dari sebuah komponen view atau viewgroup.
4. 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 peranti (yang biasa dapat diakses melalui menu settings)

**Contoh 1 :**



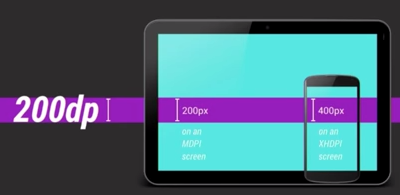
Misalkan ada dua tablet 7-inch (ukuran diagonal layar). Tablet pertama (A) memiliki resolusi layar 1200x1920px 320dpi. Tablet kedua (B) memiliki resolusi 2048x1536px 326dpi.

Sebuah tombol berukuran 300x300px akan tampak normal pada tablet A. Tapi tombol tersebut akan terlihat terlalu kecil pada tablet B.



Tapi akan berbeda jika kita tentukan ukuran button-nya dengan ukuran yang bergantung pada density layar. Bila kita menggunakan ukuran 300x300dp, maka tombol tersebut akan terlihat baik pada beragam perangkat dengan ukuran yang berbeda.

**Contoh 2 :**



Pada gambar di atas ukuran 200dp akan dikonversi pada device mdpi (device dengan density 160dpi/dots per inch) menjadi 200px dan menjadi 400px pada device xhdpi (density 420dpi) misal pada nexus 4. Sehingga ukuran tersebut tampak sama dan konsisten secara fisik untuk beragam peranti dengan ukuran layar yang berbeda.

Beberapa tautan berikut akan membantu untuk lebih memahami bagaimana aplikasi dapat mendukung beragam ukuran perangkat Android:

* [Multiple Screen Support](https://developer.android.com/guide/practices/screens_support.html)
* [Screen Density](http://developer.android.com/training/multiscreen/screendensities.html)
* [DPI Screen](http://dpi.lv/)
* [DP/PX Converter](https://pixplicity.com/dp-px-converter/)
* [Video Density-independent Pixels](https://www.youtube.com/watch?v=zhszwkcay2A)
* [Density Pixal dan Scale Pixel](https://jampasir.wordpress.com/2015/07/15/android-unit-px-pixel-dpdip-density-independent-pixel-dan-sp-scale-independent-pixels/)

Pada modul berikutnya, kita akan menerapkan teori di atas. Sehingga Anda dapat memiliki pemahaman yang lebih baik mengenai teori tersebut.