BAB 1 | ACTIVITY

1.1 Android (*open source*, Apache)

Android merupakan sebuah sistem operasi yang dikeluarkan oleh Google di bawah lisensi Apache. Android dibuat khusus untuk *smartphone* dan tablet. Salah satu alasan mengapa Android adalah sistem operasi *smartphone* dan tablet terpopuler antara lain karena OS Android merupakan sistem operasi yang *open source*, sehingga mendukung aplikasi pihak ketiga. Itu sebabnya saat ini ada ribuan aplikasi di Playstore yang bisa dipasang di Android, mulai aplikasi berita, aplikasi perpesanan, aplikasi *meeting online* seperti Zoom, hingga aplikasi saham.

Usai versi Android Alpha dan Beta dirilis, penamaan pada setiap tingkatan versi Android berikutnya adalah dengan menambahkan nama camilan, sehingga logo dan nama versi OS Android mulai Android C atau Android 1.5 identik dengan camilan, yaitu *Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, Kitkat, Lollipop, Marshmallow, Nougat, Oreo,* dan *Pie*. Namun sejak versi Android 10 pemberian nama camilan tidak digunakan lagi. Hingga sekarang pemberian nama setiap versi Android terbaru hanya berupa abjad dan angka saja.

Bahasa pemrograman yang sering digunakan untuk mengembangkan aplikasi Android adalah Java. Namun, ada beberapa bahasa lainnya yang dapat digunakan, seperti C++ dan Go. Pada IO 2017, Google juga menetapkan Kotlin sebagai tambahan bahasa resmi.

1.2 Android Studio

Android Studio adalah Integrated Development Environment (IDE) resmi untuk pengembangan aplikasi Android. IDE ini dikembangkan oleh Google dan berbasis pada IntelliJ IDEA, sebuah IDE populer untuk pengembangan aplikasi Java. Dengan menggunakan Android Studio, para developer dapat membuat aplikasi dari nol hingga dipublikasikan ke dalam *store*. Selain merupakan editor *code*

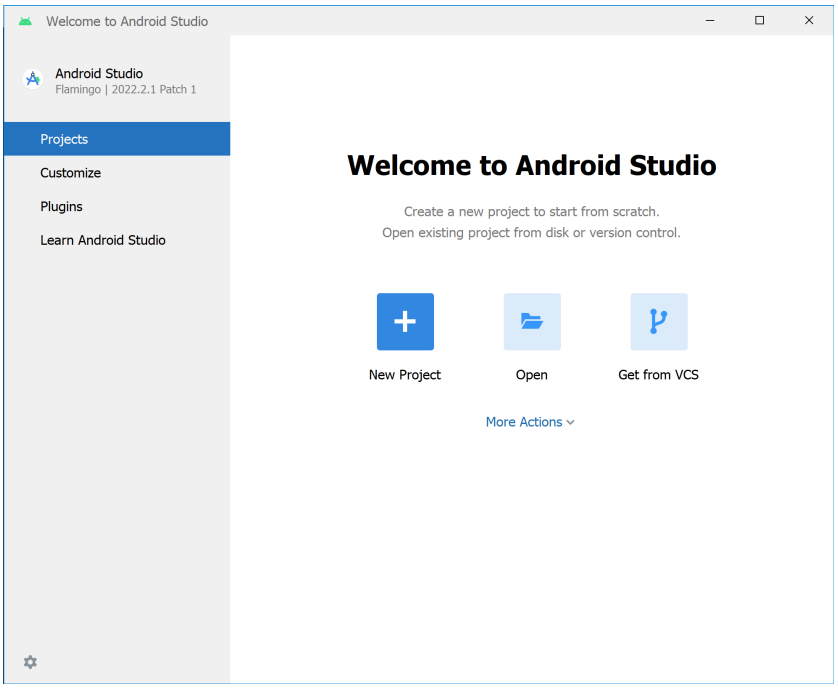
IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih demi meningkatkan produktifitas saat membuat aplikasi Android, misalnya:

- **Template**: template memulai project maupun Activity tanpa harus membuatnya dari nol.
- **Intelligent code editor**: *code completion* yang memudahkan untuk menulis kode dengan cepat tanpa harus menuliskan secara lengkap. Selain itu, juga ada warning apabila terdapat kesalahan penulisan kode.
- **Design tool**: digunakan untuk mendesain aplikasi serta melihat *preview* secara langsung sebelum dijalankan.
- **Flexible build system**: Android Studio menggunakan Gradle yang fleksibel untuk menciptakan build *variant* yang berbeda untuk berbagai *device*. Anda juga dapat menganalisa prosesnya secara mendetail.
- **Emulator**: menjalankan aplikasi tanpa harus menggunakan *device* Android.
- **Debugging**: memudahkan untuk mencari tahu masalah.
- **Testing**: menjalankan pengujian untuk memastikan semua kode aman sebelum rilis.
- **Publish**: membuat berkas AAB/APK dan menganalisanya guna dibagikan dan di-*publish* ke PlayStore. Dilengkapi dengan *Instant Run* untuk melihat perubahan tanpa harus *build project* dari awal.
- **Integrasi**: Terhubung dengan berbagai layanan yang memudahkan untuk mengembangkan aplikasi, seperti Github, Firebase, dan Google Cloud.

Cara install Android Studio:

1. Buka web browser dan masuk ke halaman https://developer.android.com/studio. Tautan tersebut akan otomatis mengarahkan ke versi software yang sesuai dengan OS Anda. Anda juga dapat melihat fitur fitur baru dan fitur utama pada halaman tersebut.
2. Klik tombol **Download Android Studio**.
3. Baca license agreement dan beri tanda centang pada **I have read and agree with the above terms and conditions** yang ada di bagian bawah.
4. Klik **Download Android Studio** dan pilih lokasi penyimpanan dan tunggu proses mengunduh sampai selesai.
5. Instal Android Studio dan ikuti petunjuknya sampai selesai.
6. Saat proses instalasi, Android Studio juga akan menginstal beberapa komponen lain supaya Android Studio dapat berjalan lancar. Contohnya adalah ADB dan SDK. Android SDK adalah sekumpulan tool yang digunakan untuk mengembangkan aplikasi Android. Untuk itu, silakan izinkan atau tekan "Yes" jika ada pop up atau alert yang muncul dan pastikan PC sudah terhubung ke internet saat proses instalasi tersebut.

7. Setelah selesai melakukan instalasi Android Studio, akan muncul seperti gambar di bawah ini.



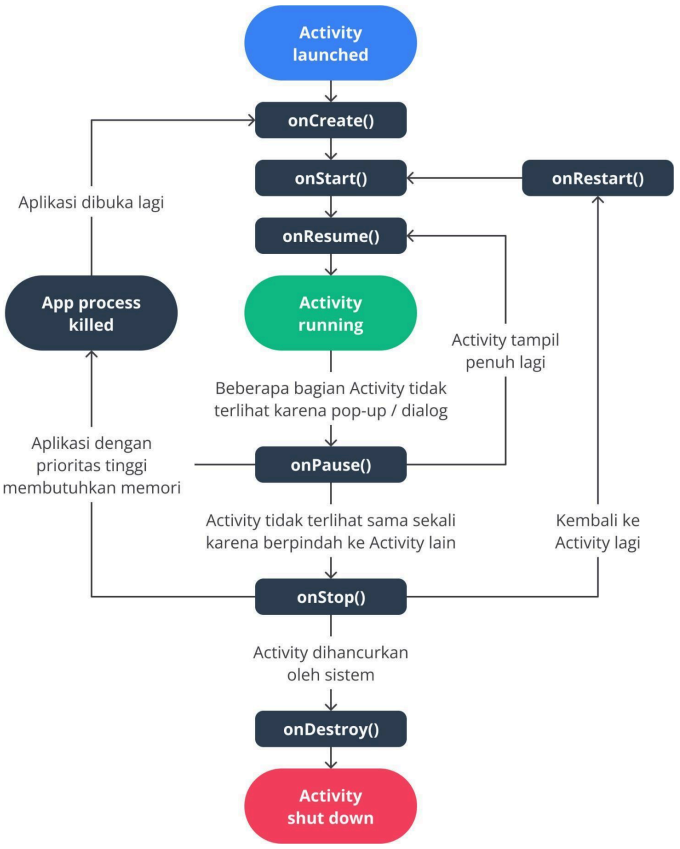
1.3 Pengertian Activity

Activity merupakan *public class* pada aplikasi Android, dimana Activity merupakan bagian terpenting dari sebuah siklus aplikasi. Sebagai salah satu komponen inti dalam platform Android, Activity bertanggung jawab untuk menampilkan antarmuka pengguna (UI) kepada pengguna dan mengelola interaksi yang terjadi di dalamnya.

Dalam sebuah aplikasi, umumnya terdapat lebih dari satu Activity yang saling terhubung dengan tugas yang berbeda-beda. Apabila sebuah aplikasi android memiliki beberapa halaman UI yang saling berinteraksi, berarti aplikasi tersebut memiliki beberapa Activity yang saling berinteraksi. Hal yang perlu diperhatikan adalah setiap Activity harus terdaftar di **AndroidManifest.xml** supaya tidak terjadi error. Secara *default*, ketika kita membuat Activity baru, ia akan terdaftar secara otomatis di **AndroidManifest.xml**. Cara membuatnya yaitu klik kanan pada nama package → New → Activity → pilih template Activity yang tersedia.

1.4 Activity Lifecycle

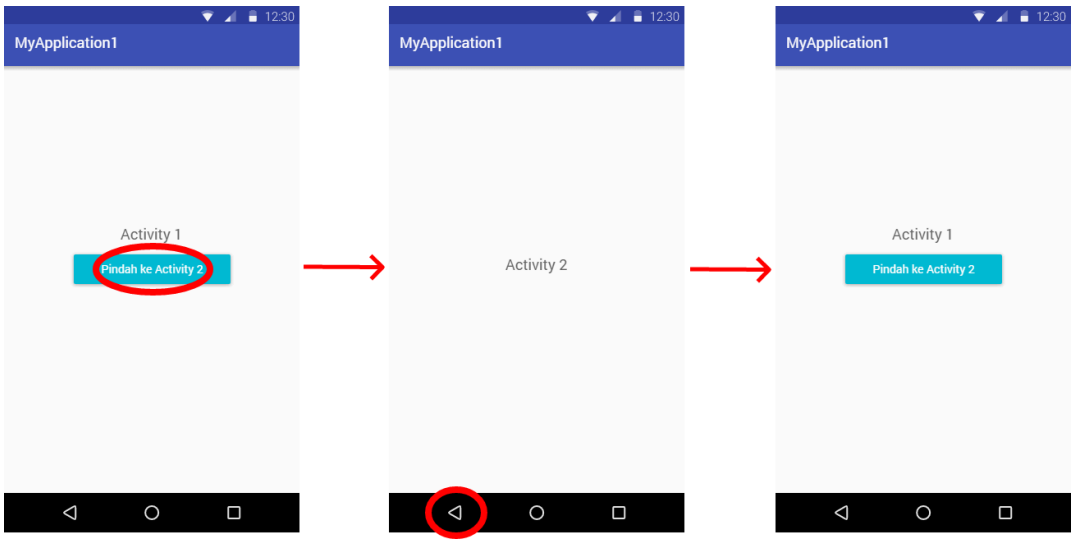
Lifecycle merupakan urutan *state* yang menunjukkan posisi proses aplikasi mulai dari Activity diciptakan sampai dihancurkan. Diagram berikut ini menampilkan status Activity dengan *method* yang akan dipanggil sebelum memasuki masing-masing status. Tiap kotak menampilkan *method* yang dipanggil.



- **onCreate()** dipanggil ketika Activity baru dibuat oleh sistem, biasanya dilakukan inisialisasi pada tahapan ini.
- **onStart()** dipanggil ketika Activity sedang dipersiapkan untuk ditampilkan kepada pengguna, tetapi Activity tersebut belum terlihat di layar.
- **onResume()** dipanggil ketika Activity sudah aktif dan sepenuhnya terlihat oleh pengguna, baik setelah Activity pertama kali dibuat atau setelah Activity kembali dari kondisi "*pause*" atau setelah **onPause()**.
- **onPause()** dipanggil ketika Activity kehilangan fokus tetapi masih terlihat oleh pengguna.
- **onStop()** dipanggil ketika Activity sedang tidak lagi terlihat oleh pengguna, tetapi masih ada dalam tumpukan aktivitas *(activity stack)* dan bisa dipulihkan.
- **onRestart()** dipanggil ketika Activity sedang dalam proses memulai kembali setelah sebelumnya berada dalam kondisi "*stop*".
- **onDestroy()** dipanggil ketika Activity sedang dihancurkan oleh sistem.

1.5 Last In, First Out (LIFO)

Last In, First Out (LIFO) adalah prinsip atau konsep dalam pemrograman dan pengelolaan data di mana elemen terakhir yang dimasukkan ke dalam suatu struktur data akan menjadi yang pertama kali diambil atau dihapus. Ini berarti elemen yang terakhir dimasukkan adalah yang pertama kali keluar atau diproses.



Gambar 1	Gambar 2	Gambar 3
Aktif: Activity 1 **onCreate() → onStart() → onResume()**	Aktif: Activity 2 Stack append: Activity 2 [**onResume()**]	Activity 1 **onStop() → onRestart() → onStart() → onResume()**
Aksi: Klik Button 1 (pindah)	Aksi: Klik Hardware Back Button	Aktif: Activity 1
Stack append: Activity 1 [**onStop()**]	Activity 2 [**finish()**] Stack pop: Activity 2 [**onDestroy()**]	

Singkatnya, ketika sebuah Activity baru dimulai, ia ditambahkan ke atas tumpukan Activity. Ketika pengguna menekan tombol "*Back*" atau Activity selesai dan dihapus, Activity yang terakhir dimulai (terbaru) akan menjadi yang pertama kali dihapus dari tumpukan Activity, sesuai dengan prinsip LIFO.

1.6 Tampilan Aplikasi (Layouting)

1.6.1 Layout XML

Layout merupakan *user interface* dari suatu activity. Layout dituliskan dalam format xml (extensible markup language). Perhatikan bahwa dalam penulisan kode XML, ada dua cara dalam penulisan tag seperti gambar di bawah ini.

- **Self-closing tag**: tag diawali dengan tanda “<”, diikuti oleh nama elemen, dan diakhiri dengan “/>”. Self-closing tag biasanya digunakan untuk View tanpa isi atau yang hanya memiliki atribut.
- **Opening dan closing tag**: Opening tag diawali dengan tanda “<” dan diakhiri dengan “>”, sedangkan closing tag diawali dengan tanda “< /”, diikuti oleh nama View, dan diakhiri dengan “>”. Opening dan closing tag biasanya digunakan untuk layout yang menampung View lain di dalamnya.

1.6.2 View dan ViewGroup

Layout Android terdiri dari dua tipe komponen *user interface*, yaitu View dan ViewGroup.

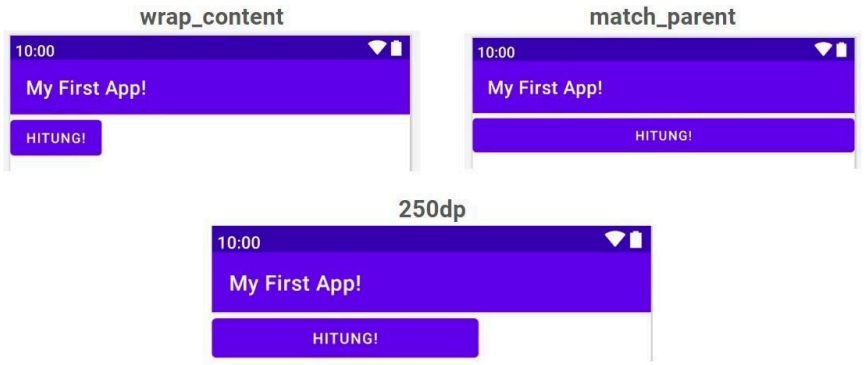
1. **View**
    - View adalah elemen dasar yang digunakan untuk membangun UI di Android. Ini termasuk elemen-elemen seperti **TextView** (untuk menampilkan teks), **ImageView** (untuk menampilkan gambar), **Button** (untuk menampilkan tombol), dan banyak lagi.
    - Dalam sebuah tag View, kita dapat mengubah *attribute* dengan beberapa format berikut.
        - **android:<property_name>="@+id/view_id"** untuk penulisan id.
        - **android:<property_name>="<property_value>"** untuk attribute biasa.
        - **android:<property_name>="@<resource_type>/resource_id"** untuk *attribute* yang memanggil value dari folder res, seperti string, *color*, dan *dimens*. yang dapat diatur melalui XML atau kode Java.

2. **ViewGroup**
    - ViewGroup adalah kontainer yang menampung satu atau lebih View maupun ViewGroup itu sendiri.
    - ViewGroup digunakan untuk mengatur posisi dan tata letak dari elemen-elemen di dalamnya sehingga membentuk tampilan aplikasi yang utuh.ViewGroup dapat mengatur posisi, ukuran, dan interaksi antara elemen-elemen UI di dalamnya, serta dapat memiliki properti tata letak seperti margin dan padding.
    - Beberapa jenis ViewGroup yang umum digunakan sebagai berikut:
        - **LinearLayout,** menyusun elemen secara tersusun secara linear baik vertikal maupun horizontal. Penggunaannya sederhana dan mudah digunakan.
        - **RelativeLayout,** menyusun elemen berdasarkan hubungan relatif antara satu sama lain.
        - **ConstraintLayout,** menyusun tiap elemen berdasarkan constraint. Penerapannya sangat fleksibel dan pilihan terbaik untuk membuat tampilan UI yang responsif.

1.6.3 Ukuran View

Sebuah view harus memiliki ukuran, yakni lebar dan tinggi. Dalam menentukan tinggi dan lebar suatu View, terdapat beberapa alternatif *value* yang bisa digunakan, seperti:

- **wrap_content**: menyesuaikan dengan ukuran konten di dalamnya;
- **match_parent**: menyesuaikan dengan ukuran parent (View induknya). Apabila di paling luar, berarti mengikuti ukuran layar device-nya; serta
- **fixed size**: menentukan ukuran dengan nilai tetap dengan satuan dp. Berikut adalah contoh ilustrasinya.



1.6.4 Padding dan Margin

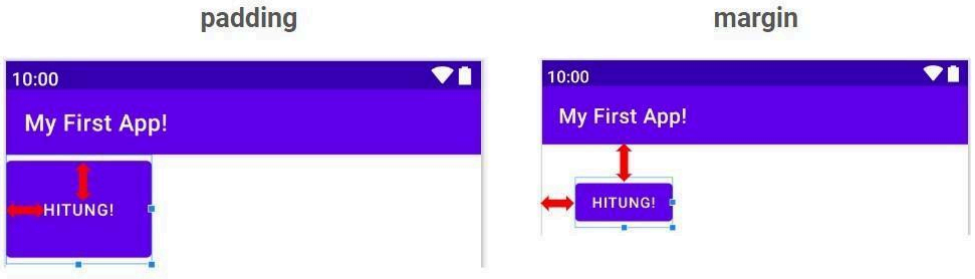
Ada beberapa cara untuk memberikan jarak antara View dalam tata letak (layout) Android:

- Padding

  Padding adalah jarak antara tepi dalam View dan kontennya. Ini berpengaruh pada ruang antara batas View dan konten di dalamnya. Kita dapat menentukan padding menggunakan atribut **android:padding** dalam XML layout file.
- Margin

  Margin adalah jarak antara tepi luar View dan tepi kontainer (biasanya ViewGroup) di sekitarnya. Kita dapat menentukan margin menggunakan atribut **android:layout_margin** dalam XML layout file.

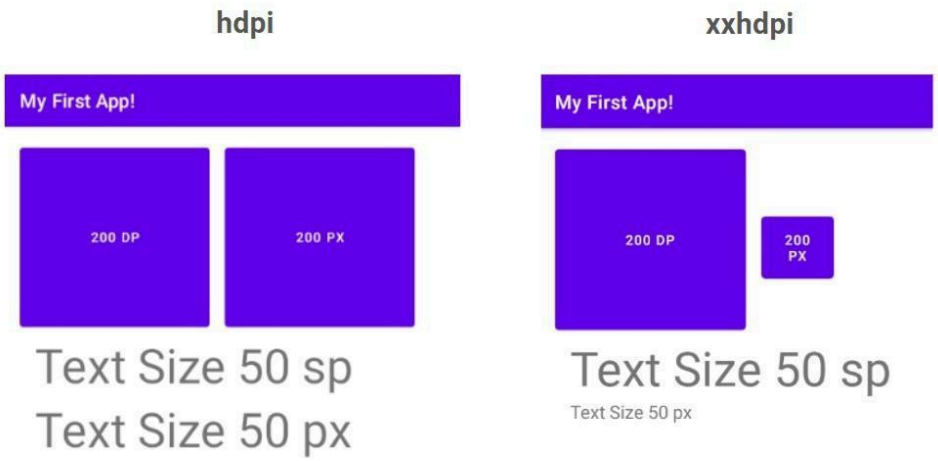
Untuk mendapatkan gambaran terkait kedua attribute tersebut, perhatikan ilustrasi berikut.



1.6.5 Satuan Ukuran

Pada platform Android, terdapat beberapa satuan ukuran yang umum digunakan untuk menentukan dimensi dalam tata letak (layout) dan *styling* elemen UI (antarmuka pengguna). Berikut adalah beberapa satuan ukuran yang sering digunakan:

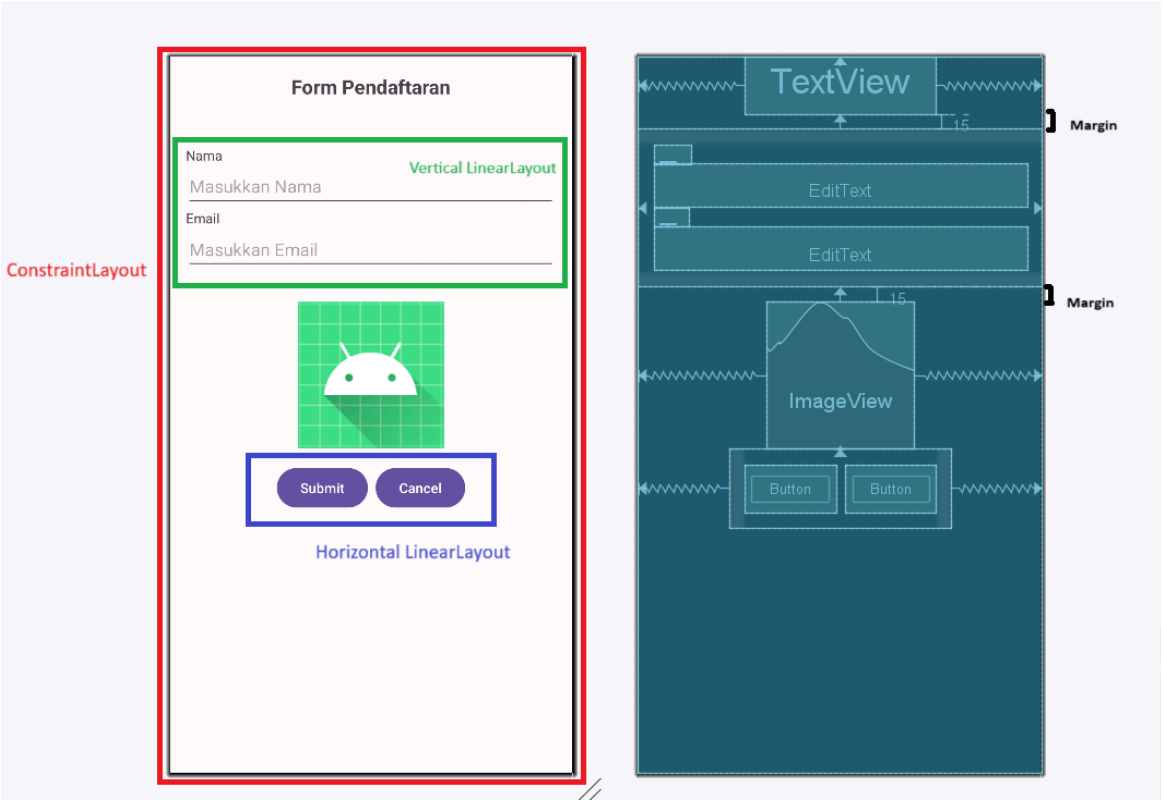
- sp (Scale-independent Pixel)
  Digunakan untuk ukuran teks, tetapi diskalakan berdasarkan preferensi ukuran font pengguna.
- dp (Density-independent Pixel)
  Digunakan untuk semuanya selain ukuran teks.
- px (Pixel)
  Sesuai dengan piksel sebenarnya di layar. Penggunaannya tidak disarankan karena dapat menghasilkan ukuran yang berbeda. Berikut adalah ilustrasi perbedaan hasil penggunaan px pada layar hdpi (ponsel) dan layar xxhdpi (tablet).



Jadi, penggunaan px dapat menghasilkan ukuran berbeda pada layar yang berbeda, tidak seperti pada penggunaan sp dan dp.

1.6.6 Studi Kasus Layouting

Pada bagian ini, kita akan mencoba mengimplementasikan materi yang telah dipelajari sebelumnya ke dalam sebuah contoh tampilan form. Simak baik-baik bagaimana komponen *user interface* (view dan viewgroup) digunakan pada tampilan form berikut:



Gambar di atas menunjukkan penggunaan ConstraintLayout sebagai layout utama dari activity. Di dalamnya, terdapat viewgroup berupa LinearLayout dengan orientasi vertikal untuk menampilkan komponen TextView dan EditText tersusun dari atas ke bawah, serta LinearLayout dengan orientasi horizontal untuk menampilkan dua buah Button secara sejajar. Margin juga digunakan untuk memberikan jarak antar beberapa komponen.