## Pengantar Advanced UI

Seperti yang kita tahu, sebuah aplikasi yang baik tak akan lepas dengan tampilan yang ciamik. Membangun tampilan menggunakan berkas layouting adalah hal dasar yang perlu dikuasai oleh seorang Android Developer. Berbagai komponen View dan ViewGroup yang disediakan Android benar-benar memudahkan kita untuk menyusun tampilan sesuai keinginan.

[A person's hand pointing at a white board

Description automatically generated](https://www.dicoding.com/academies/352/tutorials/21852)

Dalam beberapa skenario, seorang Android Developer akan menjumpai permasalahan menyusun tampilan yang lebih rumit atau biasa kita sebut dengan Advance UI. Maksudnya adalah tidak semua tampilan yang Anda inginkan bisa diselesaikan melalui proses layouting menggunakan berkas XML. Lalu bagaimana cara mengatasinya? Tenang, ada banyak cara untuk mengatasi tampilan yang lebih rumit. Berikut beberapa di antaranya.

### Custom View

Seperti yang kita tahu, Android menawarkan berbagai komponen UI yang canggih dan kuat untuk membangun tampilan aplikasi, yakni View dan ViewGroup. Sebuah View atau ViewGroup bisa berupa pewarisan dari kelas View atau ViewGroup lain. Contoh sederhananya adalah kelas Button.[A black and white text

Description automatically generated with medium confidence](https://www.dicoding.com/academies/352/tutorials/21852)

Sumber : [Dokumentasi Button Android](https://developer.android.com/reference/android/widget/Button)

Dapat Anda lihat bahwa kelas Button merupakan sebuah kelas turunan dari TextView. Sehingga, segala sifat yang dimiliki TextView secara tidak langsung dimiliki juga oleh Button. Artinya, secara tidak langsung Android menyontohkan kepada kita untuk membuat pewarisan dari kelas lain jika fiturnya terbatas. Inilah yang dinamakan dengan view khusus atau **CustomView**.

Mari kita cari kasus yang mungkin akan Anda jumpai ketika bekerja dalam sebuah tim. Katakanlah desainer meminta Anda untuk membuatkan sebuah halaman untuk aplikasi catatan. Bayangkan jika Anda hanya memanfaatkan view yang sudah ada, seperti EditText, mungkin akan kesulitan untuk memenuhi kebutuhan aplikasi. Namun, dengan Anda membuat kelas khusus dari EditText, memungkinkan Anda untuk mengubah bentuk dari komponen EditText ketika tampil di layar. Misalnya, Anda mengganti tampilan EditText yang awalnya hanya satu baris menjadi sebuah tampilan layaknya aplikasi notepad. Tentu tidak terbatas pada segi tampilan saja, Anda bisa memodifikasi fungsionalitasnya pula. Menarik, kan?

[A black cell phone with a notepad

Description automatically generated](https://www.dicoding.com/academies/352/tutorials/21852)

Selain melakukan pewarisan dari kelas View atau ViewGroup lain, Anda juga dapat membuat Custom View dari awal (from scratch). Berbeda dengan membangun tampilan dari View yang sudah ada, kini Anda bisa membuat tampilan View murni dari awal, mulai dari mengatur bentuk tampilannya hingga fungsionalitasnya di dalam berkas Kotlin atau Java. Penasaran? Tenang, kita akan membahasnya pada materi CustomView.

### Canvas

Selain membangun tampilan dengan pewarisan kelas View dan ViewGroup, Anda juga bisa menggambar sebuah objek 2D ke dalam layar menggunakan Canvas. **Canvas** merupakan object atau bisa dikatakan sebagai gambar objek 2D yang menyediakan metode untuk menggambar ke dalam **Bitmap**. Contohnya menggambar lingkaran, persegi, garis, atau object lainnya. Setelah canvas dibungkus dalam bentuk Bitmap, Anda bisa menampilkannya ke layar dengan menetapkannya ke sebuah ImageView atau kelas View. Menarik, kan?

Ketika menggambar sebuah objek 2D menggunakan Canvas, Anda dapat menambahkan properties menggunakan **Paint**. Contohnya mengatur warna, style, ukuran teks, dan instruksi lainnya. Jadi, bisa dikatakan kelas Paint merupakan “how to draw” atau bagaimana cara kita menggambar sebuah Object 2D ke dalam Canvas.

Selain itu, Canvas juga menyediakan metode untuk memotong (clipping) sebuah tampilan. **Clipping** merupakan tindakan mendefinisikan secara geometris bagian canvas mana yang dapat dilihat pengguna dalam tampilan. Bagian yang terlihat tersebut dalam terminologi grafis dinamakan dengan viewport.

Anda bisa melihat semua komponen yang diperlukan untuk menggambar ke dalam Canvas pada gambar berikut:

[A diagram of a product

Description automatically generated with medium confidence](https://www.dicoding.com/academies/352/tutorials/21852)

Jadi, Anda memerlukan 4 komponen dasar untuk menggambar object 2D di dalam aplikasi Android, yakni View (dalam hal ini kelas ImageView atau View), Bitmap, Canvas, serta Paint. Menarik bukan? Tenang, kita akan bahas lebih detailnya di materi **Canvas** ya.

### Android Widget

Tak hanya menyusun berbagai tampilan beserta fiturnya dalam aplikasi Android, terkadang seorang Android Developer juga dituntut agar bisa menyusun tampilan teaser atau sebagian fitur di dalam halaman home screen pengguna. Tunggu, menyusun tampilan di halaman home screen? Ya, mungkin terdengar asing di telinga. Tak perlu khawatir karena kita bisa memanfaatkan Android Widget untuk mengatasinya. Apa itu Android Widget?

Android Widget merupakan tambahan komponen yang ada pada launcher sistem operasi Android. Fungsi utama-nya adalah sebagai teaser atau bagian kecil dari sebuah aplikasi yang ada di dalam peranti Android kita. Sehingga, Anda dapat berinteraksi secara langsung tanpa harus membukanya.

[A screen shot of a cell phone

Description automatically generated](https://www.dicoding.com/academies/352/tutorials/21852)

Banyak aplikasi yang telah mengimplementasikan Android Widget sebagai salah satu jalan pintas sehingga pengguna dapat mengakses fitur tertentu cukup di halaman home screen. Contoh penggunaan widget antara lain:

* mengetahui proses pengiriman barang dari aplikasi e-commerce,
* mengetahui waktu sholat,
* mengetahui cuaca hari ini,
* mengetahui janji yang telah ditentukan dalam kalender,
* dan sebagainya.

Menarik kan? Tak perlu terburu-buru, Anda akan mempelajarinya secara lebih dalam pada materi Android Widget.

### WebView

Komponen yang bisa Anda gunakan adalah WebView. WebView merupakan turunan dari kelas Android View, yang memungkinkan Anda untuk menampilkan halaman website sebagai bagian dari Activity. Namun, bukan berarti semua fitur yang dikembangkan di pengembangan web bisa ditampilkan, seperti kontrol navigasi atau address bar.

[A screenshot of a cell phone

Description automatically generated](https://www.dicoding.com/academies/352/tutorials/21852)

Mengapa harus menggunakan WebView? Mengapa tidak meminta pengguna untuk mengakses langsung melalui web browser? Mungkin itu menjadi pertanyaan yang muncul dalam benak Anda, bukan? Namun, tidak mengapa. Kita akan memahami kapan waktu yang tepat menggunakan WebView, melalui skenario berikut.

Oke, katakanlah Anda mempunyai website e-commerce yang sudah berjalan dan cukup populer. Karena ingin meraih pengguna lebih banyak, Anda berinisiatif untuk membuat versi mobile-nya, khususnya Android. Namun, waktu yang Anda miliki ternyata terbatas. Sehingga pilihannya yang tersedia adalah menunda waktu perilisan aplikasi atau justru tidak menyediakan semua fitur yang ada di website Anda.

Nah, Anda tak perlu risau. Di sinilah WebView hadir untuk menyelesaikan masalah tersebut. Anda bisa mempertahankan fitur-fitur yang ada dalam website dengan menampilkannya dalam WebView, seperti FAQ, informasi promo, mengakses dokumen yang secara online, dan fitur-fitur lainnya. Dengan WebView, penyampaian informasi akan lebih mudah dan cepat daripada melakukan request API, parsing data, dan me-render-nya dalam tampilan Android. Sehingga, waktu yang digunakan tak sebanyak ketika Anda membangunnya secara native. Pembahasan lebih lengkapnya dapat Anda temui dalam teori WebView di materi berikutnya.

Itulah gambaran singkat materi yang akan Anda pelajari dalam materi Advanced UI. Yuk, kita lanjut untuk mempelajari berbagai materi tersebut. Selamat belajar!