Jetpack Compose adalah toolkit UI modern yang dirancang untuk menyederhanakan pengembangan UI di Android. Ia terdiri dari model pemrograman reaktif dengan keringkasan dan kemudahan bahasa pemrograman Kotlin. Jetpack Compose sepenuhnya deklaratif sehingga Anda dapat mendeskripsikan UI Anda dengan memanggil serangkaian fungsi yang akan mengubah data Anda menjadi hierarki UI. Ketika data berubah atau diperbarui, framework secara otomatis memanggil kembali fungsi-fungsi ini dan memperbarui tampilan untuk Anda. Dalam artikel ini, kita akan melihat beberapa topik dasar yang bermanfaat sebelum memulai dengan Jetpack Compose.

- Apa itu Jetpack Compose?
- Apa saja manfaatnya?
- Tantangan apa saja yang dapat kita atasi dengan menggunakan Jetpack Compose?
- Beberapa fungsi dasar Jetpack Compose.

Apa itu Jetpack Compose?

Jetpack Compose adalah toolkit UI modern yang baru-baru ini diluncurkan oleh Google yang digunakan untuk membangun UI Android native. Jetpack Compose menyederhanakan dan mempercepat pengembangan UI dengan kode yang lebih sedikit, API Kotlin, dan alat-alat yang canggih.

Apa saja manfaat menggunakan Jetpack Compose?

- **Deklaratif:** Ini sepenuhnya deklaratif sehingga Anda dapat mendeskripsikan komponen UI Anda dengan memanggil beberapa fungsi yang telah ditentukan sebelumnya.
- **Kompatibel:** Sangat mudah kompatibel dengan tampilan yang sudah ada di Android.
- **Meningkatkan kecepatan pengembangan:** Sebelumnya, pengembang harus mengerjakan file XML dan file Kotlin. Namun dengan bantuan Jetpack Compose, hal ini menjadi mudah dan pengembang hanya perlu mengerjakan file Kotlin, sehingga akan membantu pengembang dalam meningkatkan kecepatan pengembangan.
- **Kotlin yang Ringkas dan Sesuai dengan Gaya Bahasa Kotlin:** Jetpack Compose membangun UI dengan memanfaatkan keunggulan yang ditawarkan Kotlin.
- **Mudah dipelihara:** Karena kode sumber aplikasi apa pun terdapat dalam satu file, maka pengelolaan dan penanganan kode sumber aplikasi menjadi mudah.
- **Ditulis dalam Kotlin:** Aplikasi yang ditulis menggunakan Jetpack Compose menggunakan 100% bahasa pemrograman Kotlin.

Beberapa Fungsi Dasar Jetpack Compose

1. **Fungsi Composable:** Fungsi Composable direpresentasikan dalam kode dengan menggunakan anotasi @Composable pada nama fungsi. Fungsi ini memungkinkan Anda mendefinisikan UI aplikasi Anda secara terprogram dengan mendeskripsikan bentuk dan ketergantungan datanya, alih-alih berfokus pada proses konstruksi UI.

2. **Fungsi Preview:** Nama fungsi itu sendiri memberi tahu kita bahwa fungsi tersebut digunakan untuk menghasilkan pratinjau fungsi yang dapat disusun. Fungsi ini digunakan untuk menampilkan pratinjau fungsi yang dapat disusun di dalam IDE kita, bukan saat menginstal APK kita di emulator atau perangkat virtual. Fungsi pratinjau diberi anotasi @Preview.

3. **Fungsi Kolom:** Fungsi kolom digunakan untuk menumpuk elemen UI secara vertikal. Fungsi ini akan menumpuk semua elemen anak secara langsung satu demi satu secara vertikal tanpa spasi di antaranya. Fungsi ini diberi anotasi dengan **Column()** .

4. **Fungsi Row:** Fungsi row digunakan untuk menumpuk elemen UI secara horizontal. Fungsi ini akan menumpuk semua elemen anak satu per satu secara horizontal tanpa jarak di antaranya. Fungsi ini diberi anotasi dengan **Row()** .

5. **Box:** Sebuah widget yang digunakan untuk memposisikan elemen satu di atas yang lain. Widget ini akan memposisikan elemen anaknya relatif terhadap tepinya. Stack digunakan untuk menggambar elemen anak yang akan tumpang tindih sesuai urutan yang ditentukan. Widget ini diberi anotasi dengan **Box()** .

6. **Spacer:** Digunakan untuk memberikan jarak antara dua tampilan. Kita dapat menentukan tinggi dan lebar kotak. Ini adalah kotak kosong yang digunakan untuk memberikan jarak antara tampilan. Kotak ini diberi anotasi dengan **Spacer()** .

7. **Vertikal Scroll:** Jika komponen UI di dalam aplikasi tidak sesuai dengan tinggi layar, maka kita harus menggunakan tipe tampilan pengguliran. Dengan bantuan penggulir vertikal, kita dapat memberikan perilaku pengguliran vertikal pada tampilan kita. Konten di dalam penggulir vertikal akan dipotong sesuai dengan batas penggulir vertikal. Ini diberi anotasi dengan **VerticalScroll()** .

8. **Padding:** Fungsi padding digunakan untuk memberikan ruang kosong tambahan sesuai kebutuhan di sekitar tampilan tertentu. Fungsi ini cukup diberi anotasi dengan **Padding()** .

9. **Lazy List:** Composable ini setara dengan recycler view di sistem view Android. Ia diberi anotasi LazyColumn().

Arsitektur Android

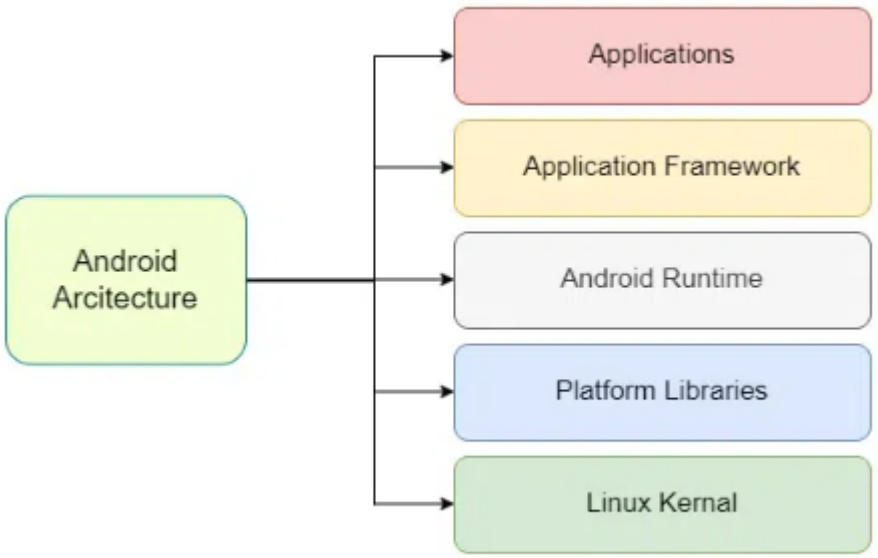
Arsitektur Android berisi sejumlah komponen berbeda untuk mendukung kebutuhan setiap perangkat Android. Perangkat lunak Android berisi Kernel Linux sumber terbuka yang memiliki kumpulan sejumlah pustaka C/C++ yang diekspos melalui layanan kerangka kerja aplikasi. Di antara semua komponen tersebut, Kernel Linux menyediakan fungsi utama sistem operasi untuk ponsel pintar, dan Dalvik Virtual Machine (DVM) menyediakan platform untuk menjalankan aplikasi Android.

Komponen Arsitektur Android

Komponen utama arsitektur Android adalah sebagai berikut:

- Aplikasi
- Kerangka Kerja Aplikasi
- Android Runtime
- Pustaka Platform
- Kernel Linux

Representasi bergambar arsitektur Android dengan beberapa komponen utama dan sub-komponennya.



https://www.geeksforgeeks.org/android/android-architecture/

Arsitektur Lain yang Umum Digunakan di Android

Terdapat beberapa arsitektur Android yang umum digunakan, yang disebutkan di bawah ini:

1. MVC (Model View Controller)

MVC atau Model View Controller membagi model menjadi tiga komponen utama: **Model** yang menyimpan data aplikasi, **View** (lapisan UI) yang menampilkan komponen di layar, dan **Controller** yang membangun hubungan antara Model dan View.

2. MVP (Model View Presenter)

Untuk menghindari kompleksitas seperti pemeliharaan, keterbacaan, skalabilitas, dan refactoring aplikasi, kami menggunakan model MVP . Cara kerja dasar model ini bergantung pada poin-poin yang disebutkan di bawah ini:

- Komunikasi antara View-Presenter dan Presenter-Model terjadi dengan bantuan Interface (juga disebut Contract).
- Terdapat hubungan satu lawan satu antara Presenter dan View, satu kelas Presenter hanya mengelola satu View pada satu waktu.
- Model dan View tidak memiliki pengetahuan satu sama lain.

3. MVVM (Model View ViewModel):

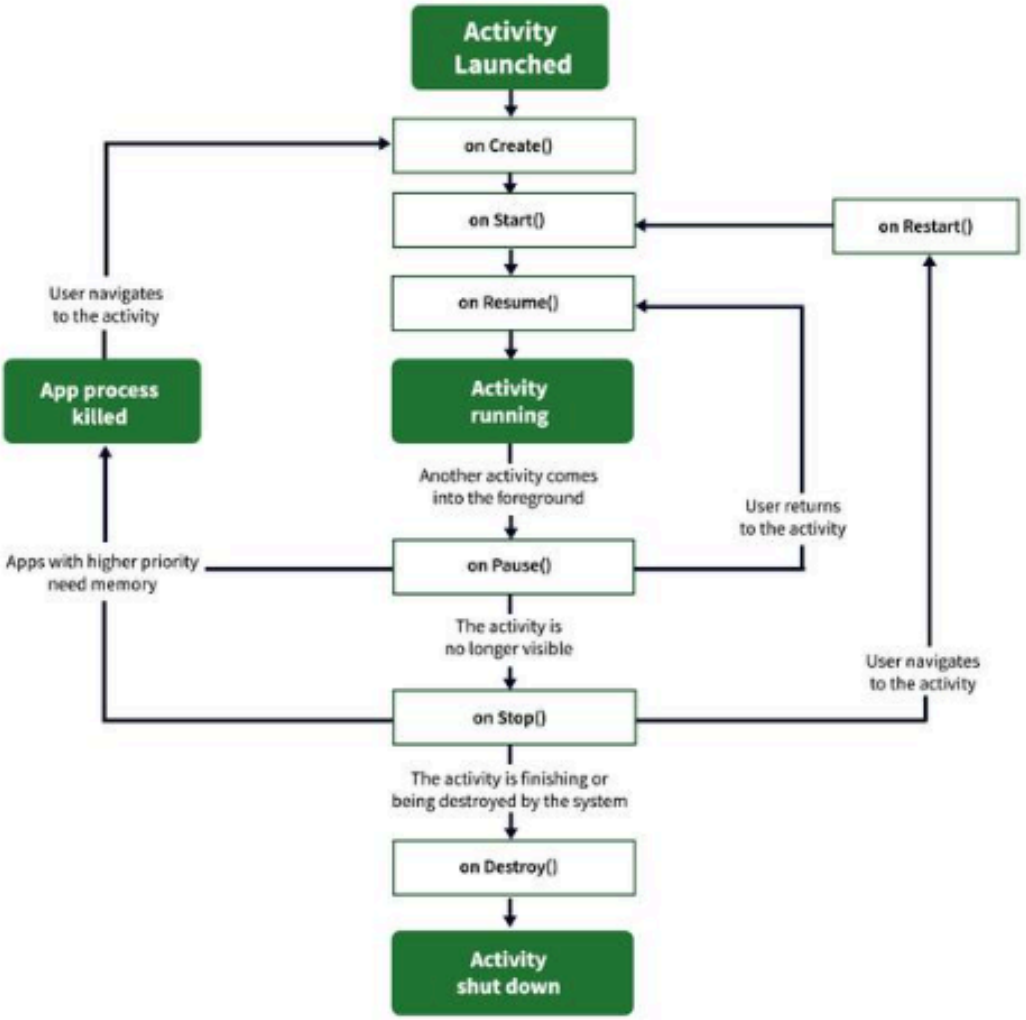
MVVM atau Model View ViewModel, seperti namanya, mirip dengan model MVC dan juga terdiri dari tiga komponen: Model, View, dan ViewModel. Fitur-fitur model MVVM disebutkan di bawah ini:

- ViewModel tidak menyimpan referensi apa pun ke View.
- Terdapat banyak relasi satu-ke-satu antara View dan ViewModel.
- Tidak ada metode pemicu untuk memperbarui Tampilan.

Dan kita dapat mencapai hal ini menggunakan 2 metode:

- Menggunakan pustaka DataBinding dari Google.
- Menggunakan alat seperti RxJava untuk pengikatan data.

Lifecycle Aplikasi



Activity Lifecycle in Android

https://www.geeksforgeeks.org/blogs/android-app-development-fundamentals-for-beginners/

https://www.geeksforgeeks.org/android/activity-lifecycle-in-android-with-demo-app/

https://www.geeksforgeeks.org/android/android-tutorial/

https://www.geeksforgeeks.org/kotlin/kotlin-android-tutorial/

https://www.geeksforgeeks.org/android/android-studio-tutorial/

https://www.geeksforgeeks.org/android/android-projects-from-basic-to-advanced-level/

Tahapan Siklus Hidup Android:

1. **OnCreate:** Fungsi ini dipanggil saat aktivitas pertama kali dibuat.
2. **OnStart:** Fungsi ini dipanggil ketika aktivitas tersebut terlihat oleh pengguna.

3. **OnResume:** Fungsi ini dipanggil ketika aktivitas mulai berinteraksi dengan pengguna.
4. **OnPause:** Fungsi ini dipanggil ketika aktivitas tidak terlihat oleh pengguna.
5. **OnStop:** Fungsi ini dipanggil ketika aktivitas tidak lagi terlihat.
6. **OnRestart:** Fungsi ini dipanggil ketika aktivitas dihentikan, dan kemudian dimulai kembali.
7. **OnDestroy:** Metode ini dipanggil ketika aktivitas akan ditutup atau dihancurkan.