# Pengenalan Pengembangan Aplikasi Berbasis Perangkat Bergerak

**Roberto Kaban**
email: **roberto.kaban@yahoo.com**
blog: **https://kaban.my.id**
hp/wa: **081260329842**

## Pendahuluan

Aplikasi perangkat bergerak merupakan aplikasi yang dimaksudkan untuk di-install dan dijalankan pada perangkat bergerak (mobile phone/gadget). Saat ini, pasar mobile phone terbagi menjadi dua kelompok besar yakni Android dan iOS (iPhone). Pangsa pasar untuk Android saat ini menempati bagian yang lebih besar daripada iOS. Statistik pada https://gs.statcounter.com/os-market-share/mobile/worldwide menunjuk- kan bahwa pangsa pasar pada bulan November 2020 adalah 71,18% untuk Android, 28,19% untuk iOS, dan sisanya untuk perangkat bergerak lainnya.

Sebagai suatu perangkat bergerak dengan pangsa pasar terbesar, Android mempunyai berbagai perangkat pengembangan yang dapat digunakan. Dari berbagai perangkat pengembangan tersebut, Google sebagai perusahaan yang memproduksi Android OS mendukung dua bahasa pemrograman resmi yang, yakni **Java** dan **Kotlin**. Pengembangan menggunakan Java dan Kotlin dilakukan dengan menggunakan *software* Android Studio yang dapat diperoleh secara bebas. Meskipun demikian, banyak komunitas maupun perusahaan-perusahaan lain yang membuat perangkat pengembangan mereka sendiri. Dari berbagai piranti pengembangan tersebut, kita dapat membaginya ke dalam 2 kategori besar yaitu ***native*** dan ***hybrid***.

Perangkat pengembangan *native* langsung dapat dijalankan oleh *platform* Android, artinya dikembangkan dan dikompilasi ke *bytecode* JVM (*Java Virtual Machine*) karena Android *runtime* dikembangkan menggunakan Java. Java dan Kotlin merupakan bagian dari peranti pengembangan *native* karena hasil yang diperoleh merupakan *bytecode* JVM. Perangkat pengembangan *hybrid* dijalankan dengan menggunakan fitur ***WebView*** dan dengan pustaka (*library code*) tertentu dibuat hingga dapat mengakses sisi *native* dari sistem Android. Perangkat pengembangan ini menggunakan berbagai sarana spesifikasi *Web* (terutama HTML dan CSS) untuk membentuk antarmuka (*interface*), dan memerlukan pustaka (*library*) lain jika ingin mengakses sisi *native* (seperti mengaktifkan dan menggunakan kamera, dan sejenisnya).

Untuk dapat/mampu membangun aplikasi dengan menggunakan pendekatan *hybrid* ini, seorang pemrogram harus memahami pembuatan aplikasi di sisi *Web* terutama menggunakan HTML, CSS, serta *JavaScript* dan *framework* antarmuka (*interface*) *Web* tertentu.

## Lingkungan Pengembangan Aplikasi Perangkat Bergerak

Pada umumnya, aplikasi yang berjalan pada perangkat bergerak Android dapat dikembangkan secara *native* maupun *hybrid*. Aplikasi yang dikembangkan secara *native* memungkinkan dapat langsung dijalankan oleh *runtime* dari Android; sedangkan aplikasi *hybrid* merupakan aplikasi yang harus dijalankan di atas *platform Web* – pada sistem operasi Android, aplikasi tersebut dijalankan dengan menggunakan fitur *WebView*. Salah satu *framework* yang dapat digunakan untuk mengembangkan aplikasi *hybrid* ini adalah **Ionic**.

**Platform Pengembangan Aplikasi Perangkat Bergerak**

Aplikasi *mobile* dikembangkan dengan *platform mobile* tertentu. Saat ini terdapat 2 *platform* utama untuk pengembangan aplikasi *mobile* yaitu iOS (untuk perangkat iPhone) dan Android. Pada sistem Android, sistem operasi yang digunakan di perangkat *mobile* tersebut adalah sistem operasi Linux yang sudah dimodifikasi agar dapat digunakan dengan perangkat layar sentuh *mobile*. Sistem operasi untuk Android tersebut merupakan sistem operasi bebas (*free*) dengan kode terbuka (*open source*), itulah sebabnya kemudian banyak muncul berbagai varian dari Android yang dibuat oleh *vendor mobile phone*, misalnya ColorOS - untuk *mobile phone* Oppo.

**Gambaran Umum**

Android OS dikembangkan berdasar pada kernel Linux yang telah dimodifikasi. Kode sumber untuk Android OS ini dapat diperoleh di https://source.android.com. Android OS merupakan sistem operasi yang tersedia bebas. Secara umum, komponen dari Android OS ini dapat dilihat pada Gambar 1.1.


Gambar 1.1 Arsitektur Android OS

Agar dapat memahami cara membangun suatu aplikasi Android, tidak terlalu diperlukan pengetahuan mendalam sampai di level paling bawah (kernel), melainkan cukup dengan memahami arsitektur Android OS secara umum.

Aplikasi Android berada di level paling atas dan akan dijalankan oleh suatu *runtime*. Pada Android di bawah versi 4.4 (**KitKat**) *runtime* yang digunakan adalah **Dalvik**. Pada versi 4.4 sudah menggunakan versi ART (*Android Run Time*) sebagai pengganti dari Dalvik, namun masih menyertakan Dalvik. Mulai versi 5.0 (**Lolipop**) hingga versi pada saat ini (Android v.11/**Red Velvet Cake** ), Android sudah menggunakan ART secara penuh dan menghilangkan Dalvik. *Runtime* tersebut akan menjadi jembatan ke akses rendah/mesin melalui HAL (*Hadware Abstraction Layer*) yang didefinisikan menggunakan HIDL (*HAL Interface Definition Language*). *Kernel* merupakan *software*

yang mengelola mesin secara langsung dan menyediakan antarmuka (*interface*) bagi ART.

**Perbedaan Aplikasi *Native* dan *Hybrid***

Aplikasi *native* merupakan aplikasi yang dijalankan langsung oleh ART (pada Dalvik di versi 4.4/KitKat ke bawah). Aplikasi *native* dibuat dengan menggunakan bahasa pemrograman Java/Kotlin atau semua bahasa yang menghasilkan *bytecode* atau C++ dengan akses langsung ke fungsi *native* dan kemudian menggunakan JNI (*Java Native Interface*) yang dipanggil oleh Java/Kotlin. Beberapa aplikasi dibangun dengan SDK (*Software Development Kit*) lain seperti Flutter yang menggunakan C/C++ di NDK (*Native Development Kit*) untuk membangun *engine*.

Pada teknologi lainnya, suatu aplikasi *hybrid* memerlukan suatu *software* lain yang akan dimasukkan ke dalam suatu aplikasi yaitu teknologi *browser*. *Browser* akan dimasukkan sebagai *engine* dari suatu aplikasi dan memungkinkan aplikasi yang dibangun dapat menggunakan standar *Web* (HTML, CSS, JavaScript) sebagai sarana antarmukanya. Secara *default*, seperti halnya aplikasi *Web* pada perangkat bukan *mobile*; suatu aplikasi *Web* tidak memungkinkan/mengizinkan aplikasi untuk dapat mengakses fasilitas level rendah/mesin secara langsung. Untuk keperluan tersebut, digunakan perangkat *software* lain yang berfungsi untuk menyediakan jembatan bagi aplikasi *Web* dengan akses level rendah.

Aplikasi *native* mempunyai kemampuan yang baik untuk mengakses semua fasilitas standar dari *mobile phone*. Sementara itu, untuk *hybrid* diperlukan jembatan dan *wrapper* terhadap kemampuan akses level rendah. Hal ini menyebabkan kode sumber (*source code*) untuk aplikasi *native* relatif dapat langsung mengakses level rendah tanpa bantuan dari berbagai perangkat pihak ketiga.

Dari sisi pengembangan, aplikasi *native* bersifat relatif lebih khusus/ spesifik karena kode sumber dibuat sesuai dengan perangkat yang dituju/digunakan. Hal ini berbeda dengan *hybrid* yang menggunakan spesifikasi terbuka – *Web*; maka sekali dibangun untuk satu *platform*, biasanya dapat digunakan/dijalankan pada *platform* yang lain (Android, iOS, maupun *Web desktop*/Internet). Tugas terbesar pada pengembangan versi *hybrid* dilakukan oleh pihak ketiga yang menyediakan akses ke level rendah/mesin/ *native*. Pada pengembangan aplikasi berbasis Ionic, tugas ini dikerjakan oleh *Capacitor* dan/atau *Apache Cordova.* Subyek/topik ini akan disajikan pada saat membahas *API Plugins*.

Dari sisi ukuran dan kecepatan aplikasi, aplikasi *native* mempunyai ukuran yang lebih kecil serta kecepatan yang lebih baik dibandingkan aplikasi *hybrid*. Hal ini disebabkan karena aplikasi *hybrid* menyertakan fasilitas *browser engine* sebagai perangkat untuk menampilkan antarmuka serta menyediakan kemungkinan akses ke level rendah melalui jembatan penghubung; Sedangkan pada aplikasi *native* akan disertakan hasil kompilasi berupa *bytecode* yang siap untuk langsung dijalankan oleh ART.

**Perangkat Pengembangan dan *Framework* untuk Aplikasi *Mobile***

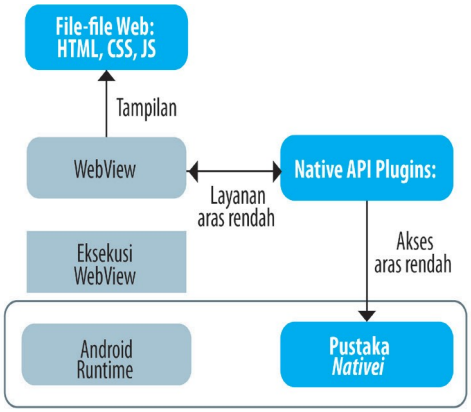
Pada dasarnya, tersedia banyak perangkat lunak yang dapat digunakan untuk membangun suatu aplikasi *mobile*. Perangkat lunak tersebut terbagi dalam kategori *native* serta *hybrid*. Beberapa perangkat lunak tersebut dapat dilihat pada Tabel 1.1.

Tabel 1.1
Daftar Framework untuk Mengembangkan Aplikasi Android

| Nama | Bahasa Pemrograman | Teknologi | Cross Platform Android - iOS |
|---|---|---|---|
| Java | Java | Native | Tidak |
| Kotlin | Kotlin | Native | Tidak |
| KMM (Kotlin Multiplatform Mobile) | Kotlin | Native | Ya |
| Flutter | Dart | Native | Ya |
| NativeScript | JavaScript /TypeScript | Native | Ya |
| Quasar Framework | JavaScript /TypeScript | Hybrid | Ya |
| Framework7 | JavaScript /TypeScript | Hybrid | Ya |
| Xamarin | C# | Native | Ya |
| ReactNative | JavaScript | Native | Ya |
| Ionic Framework | JavaScript /TypeScript | Hybrid | Ya |

**APLIKASI *HYBRID***

Aplikasi *hybrid* menyertakan *browser engine* sebagai komponen untuk membangun antarmuka serta akses ke level rendah/mesin. Untuk memahami lebih lanjut tentang Ionic, diperlukan pemahaman tentang arsitektur aplikasi *hybrid* seperti pada Gambar 1.2.




Gambar 1.2 Arsitektur Aplikasi *Hybrid*

Terdapat 2 komponen utama dari suatu aplikasi *hybrid*, yaitu komponen untuk tampilan dan komponen untuk akses pustaka *native*. Jika aplikasi tidak menggunakan fasilitas *native*, maka tidak diperlukan *native API plugins*. Namun apabila aplikasi memerlukan akses ke level rendah - misalnya untuk mengaktifkan kamera, *geolocation*, dan lain-lain, maka diperlukan *native API plugins* untuk fungsi tersebut. *WebView* dijalankan oleh **ART** dan kemudian menghasilkan suatu tampilan. *WebView* juga menyediakan *engine* JavaScript yang memungkinkan penggunaan JavaScript sebagai bahasa pemrograman untuk mengimplementasikan *business logic* dari aplikasi tersebut.

## IONIC FRAMEWORK

### Gambaran Umum

Ionic merupakan salah satu *framework* yang dapat digunakan untuk membangun aplikasi *mobile* dengan menggunakan teknologi *hybrid*. Ionic dibuat oleh perusahaan yang bernama **Drifty Co.**, pertama kali dirilis pada tahun 2013. Pada saat ini, Ionic sudah mencapai versi 7. Ionic memungkinkan pembuatan aplikasi yang bersifat *cross platform* dan dapat dijalankan pada berbagai perangkat *mobile* (Android, iOS), aplikasi *desktop*, serta aplikasi PWA (*Progressive Web Apps*).

### Komponen Ionic *Framework*

Ionic terdiri atas 2 (dua) komponen, yakni (a) komponen untuk antarmuka dan (b) komponen untuk akses ke pustaka *native* guna memenuhi kebutuhan akses level rendah. Ionic menggunakan Apache Cordova serta Capacitor untuk memfasilitasi akses ke level rendah/mesin - khususnya untuk memberi kemampuan menggunakan sarana yang tersedia pada *mobile phone* seperti kamera, lampu (*flashlight)*, *geolocation*, dan lain sebagainya. Pada sisi pembuatan antarmuka, Ionic memungkinkan penggunaan berbagai *framework* yang banyak digunakan di industri, yaitu Angular, React, atau Vue.

Bahasa pemrograman yang digunakan secara *default* adalah JavaScript. Meskipun demikian, TypeScript merupakan bahasa pemrograman yang makin banyak digunakan dan dapat digunakan untuk ketiga *framework* antarmuka tersebut. Vue pada versi terakhir (versi 3) dikembangkan dengan menggunakan TypeScript.

### Ekosistem Ionic *Framework*

Beberapa *software* atau komponen *software* mempunyai keterkaitan dengan Ionic.

1. **Node.js**: JavaScript *runtime* yang dibuat dari *V8* (*JavaScript engine* dari Google Chrome).
2. **TypeScript**: bahasa pemrograman bertipe *static typing* yang merupakan *superset* dari JavaScript. TypeScript ini banyak digunakan di berbagai *framework* untuk tampilan antarmuka. TypeScript dikompilasi menjadi JavaScript dan dijalankan dengan menggunakan Node.js.
3. ***Framework* untuk antarmuka**: Ionic mendukung setidaknya 3 *framework*, yaitu Angular, React, dan Vue.
4. Pustaka / paket JavaScript dan TypeScript di **npm (**suatu *software* yang digunakan untuk mengelola paket serta proyek Node.js)
5. ***Web Components***: komponen siap pakai untuk tampilan antarmuka dengan

menggunakan Ionic.

6. **Stencil**: kompilator untuk membuat *web components*.
7. **Apache Cordova**: perangkat penyedia *native API plugins* untuk memungkinkan aplikasi yang dibangun; dengan menggunakan sarana ini, Ionic dapat mengakses perangkat sistem di level rendah.
8. **Capacitor**: mempunyai fungsi yang sama dengan Apache Cordova, tetapi dibuat oleh Ionic.
9. **Android SDK** (*Software Development Kit*): perangkat pengembangan yang digunakan untuk membangun aplikasi Android.
10. **IDE** (*Integrated Development Environment*): perangkat *software* yang menyediakan lingkungan pengembangan terpadu untuk membangun aplikasi. Bagi Ionic, meskipun dapat menggunakan IDE/editor teks apa saja, tetapi disarankan menggunakan Visual Studio Code. Visual Studio Code menyediakan dukungan yang komprehensif untuk pengembangan berbasis JavaScript / TypeScript.
11. Informasi tentang integrasi Ionic dengan berbagai komponen lainnya, dapat diakses di https://ionicframework.com/integrations.

**Rangkuman**

Saat ini tersedia 2 platform besar untuk perangkat *mobile* yaitu Android serta iOS. Android menempati pangsa pasar tertinggi di seluruh dunia. Aplikasi untuk Android dikembangkan dengan menggunakan 2 teknik, yaitu *native* dan *hybrid*.

Aplikasi *native* adalah aplikasi yang dikembangkan secara langsung untuk dijalankan secara langsung dengan menggunakan ART (*Android Runtime*).

Aplikasi *hybrid* menyertakan *browser engine* di dalam aplikasi. *Browser engine* tersebut berfungsi untuk menghasilkan tampilan berbasis spesifikasi teknologi *Web* (HTML, CSS, JavaScript). Guna mendapatkan akses aras/level rendah/sistem, aplikasi *hybrid* menggunakan perangkat pembantu yang disebut *Native API plugins*.

Salah satu *framework* yang dapat digunakan untuk mengembangkan aplikasi *hybrid* adalah Ionic. Ionic memungkinkan seseorang pengembang aplikasi untuk dapat membangun aplikasi dengan menggunakan spesifikasi teknologi *Web* serta memakai *framework* antarmuka Angular, React, atau Vue; dengan menggunakan JavaScript dan/ atau TypeScript sebagai bahasa pemrograman untuk Ionic.