BAB 2
LANDASAN TEORI

2.1 Pengertian Batik

Menurut Ari Wulandari (Batik Nusantara, 2011) batik merupakan wujud hasil cipta seni yang adiluhung, dieksperikan pada motif kain untuk pakaian, sarung, kain panjang, dan kain dekoratif lainnya. Dalam Bahasa Jawa, batik ditulis dengan bathik mengacuk pada huruf Jawa “tha” yang menunjukan bahwa batik adalah rangkaian dari titik - titik yang membentuk gambaran tertentu. [1]

Batik telah menjadi bagian dari budaya Indonesia yang mempunyai nilai seni yang tinggi sekaligus sarat makna. Batik bukan hanya sekedar seni dan keterampilan menuliskan malam atau lilin untuk membentuk motif atau lukisan diatas selembar kain dengan menggunakan canting. Banyak nilai dan makna yang terpancar dari sehelai kain yang bernama batik. Sebab, perpaduan antara warna dengan motif batik yang ditorehkan pada selembar kain selalu mengandung makna yang mendalam.

Pada sehelai kain batik dengan motif dan warna tertentu, dapat ditemukan makna yang mengungkap hubungan harmonis anatara manusia dengan sang pencipta, manusia dengan alam, dan juga antara sesama manusia. Tidak hanya itu, motif yang tergurat pada batik juga menyimpan pesan moral dan kearifan lokal suatu masyarakat. [7]

2.1.1 Motif Ceplok-Grompol

Berdasarkan corak dan motifnya, Grompol dapat dikategorikan ke dalam kelompok batik motif ceplok. Kata *grompol* berasal dari kata dompol yang berarti sekumpulan barang-barang atau nama bentuk cincin permata yang indah. Batik motif grompol memiliki makna filosofi berkumpul, Bersatu, atau berkelompok. Motif ini diibaratkan sebuah pohon yang penuh dengan bunga dan buah. Batik motif Ceplok-Grompol mengandung makna harapan kepada tuhan agar senantiasa melimpahkan rahmat dan anugerah berupa kehidupan yang rukun, tentram, sejahtera, serta banyak anak dan rezeki.

Sesuai dengan makna filosofi yang terkandung di dalamnya, batik motif Ceplok-Grompol banyak digunakan dalam upacara perkawinan, yaitu pada saat acara siraman atau kirab pengantin. Dengan memakai batik motif Ceplok-Grompol diharapkan berkumpulnya atau bersatunya segala sesuatu yang baik seperti rezeki, keturunan, kerukunan, dan kebahagiaa dalam hidup berkeluarga. Adapun contoh gambar dari salah satu motif Ceplok-Grompol adalah sebagai berikut. [7]

Gambar 2.1 Motif Ceplok-Grompol

2.1.2 Motif Kawung

Batik motif Kawung memiliki pola bulatan-bulatan seperti buah kawung , yaitu sejenis kepala atau buah kolang-kaling yang tersusun secara geometris dan rapi. Batik motif Kawung sering kali juga digambarkan sebagai bunga teratai dengan empat helai daun dimana terdapat bunga yang merekah dibagian tengahnya. Bunga teratai sendiri secara filosofi melambangkan kesucian dan panjang umur.

Pada zaman dahulu, batik motif Kawung hanya boleh dikenakan oleh raja dan keluarga kerajaan. Batik motif kawung melambangkan kekuasaan dan keadilan. Seiring dengan perkembangan zaman, batik motif Kawung bisa dikenakan oleh masyarakat. Batik motif Kawung kini lebih diartikan sebagai keinginan dan kerja keras yang akan selalu membuahkan hasil, walaupun membutuhkan perjuangan dan pengorbanan namun demikian sikap itu sebaiknya juga dibarengi dengan tindakan hemat dan cermat. Adapun contoh gambar dari salah satu motif Kawung adalah sebagai berikut. [7]

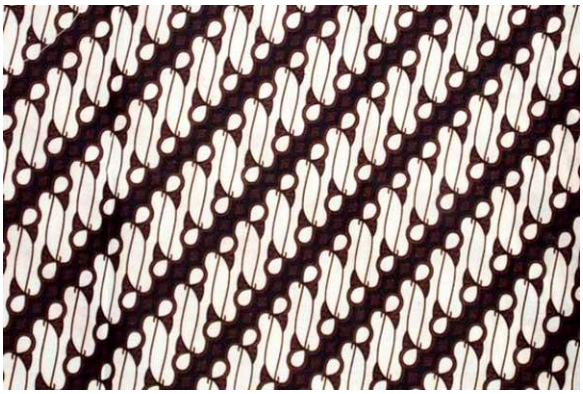
Gambar 2.2 Motif Kawung

2.1.3 Motif Parang

Batik motif Parang juga merupakan salah satu motif batik yang kuno. Kata parang berasal dari Bahasa Jawa *pereng* yang digambarkan berupa garis lengkung-lengkung menyerupai ombak laut. Batik motif Parang dikenal juga sebagai batik dengan pola pedang atau keris.

Simbol mata parang yang berbentuk seperti huruf"S" melambangkan kekuasaan, kekuatan, dan semangat yang tidak pernah padam bak lidah api yang menyala-nyala. Itulah sebabnya, maka pada masa kerajaan Mataram di Kartasura, batik motif Parang hanya boleh dikenakan oleh raja serta penguasa dan kesatria seperti pangeran dan senopati. Oleh sebab itu, batik motif Parang disebut juga batik Larangan karena tidak boleh dipakai oleh rakyat biasa.

Dalam perkembangan selanjutnya, muncul berbagai batik motif Parang yang lain seperti motif Parang Rusak, Parang Barong, Parang Klitik, Parang Slobog, dan Parang Kusumo. Masing-masing motif batik itu mempunyai perbedaan yang cukup mencolok namun tetap bercirikan motif garis lengkung seperti gelombang ombak. Adapun contoh gambar dari salah satu motif Parang adalah sebagai berikut. [7]



Gambar 2.3 Motif Parang

2.1.4 Motif Lereng

Batik motif Lereng mempunyai pola baris diagonal diantara motif Parang. Selain itu, batik motif ini juga dapat ditemukan dalam bentuk pola sederhana, yaitu hanya berupa deretan garis diagonal yang sempit penuh dengan ornament-ornamen kecil. Ciri-ciri khusus batik motif Lereng, antara lain pola baris diagonal miring 45 derajat. Ragam hiasnya hanya dibatasi garis lurus, kadang-kadang diselingi dengan motif Parang atau Lung-lungan.

Batik motif Lereng termasuk motif batik kuno. Pada zaman dahulu. Batik motif ini hanya boleh dikenakan untuk keluarga kerajaan. Beberapa batik motif Lereng Glebangan, Lereng Tahthit, dan Lereng Sobrah. Batik motif Lereng mengandung makna filosofi sebagai perlambangan kesucian dan kekuatan. Adapun contoh gambar dari salah satu motif Lereng adalah sebagai berikut. [7]

Gambar 2.4 Motif Lereng

2.1.5 Motif Nitik

Batik motif Nitik sebenarnya tercipta akibat pengaruh dari budaya luar yang berkembang di pantai utara laut Jawa. Batik motif Nitik akhirnya berkembang hingga di pedalaman dan menjadi motif batik yang sangat indah.

Setiap jenis batik motif Nitik mempunyai makna filosofi yang terkandung di dalamnya. Batik motif Nitik Karawitan mempunyai makna kebijaksanaan, sehingga para pemakainya diharapkan akan menjadi orang yang bijaksana. Itulah sebabnya, orang-orang yang dituakan pada suatu lingkungan banyak mengenakan batik motif ini.

Batik motif Nitik Cakar banyak digunakan pada upacara adat perkawinan. Diberi nama demikian karena dibagian motifnya terdapat ornament yang berbentuk seperti cakar. Cakar yang dimaksud dimkasudkan adalah cakar ayam. Batik motif Nitik dimaksudkan agar pasangan yang menikah dapat mencari nafkah dengan halal sepandai ayam mencari makan dengan cakar. Adapun contoh gambar dari salah satu motif Nitik adalah sebagai berikut. [7]

Gambar 2.5 Motif Nitik

2.1.6 Motif Semen

Dalam Bahasa Jawa, semen berasal dari kata *semi* yang berarti tumbuh atau berkembang. Secara khusus *semi* berati tunas atau tumbuhnyabagian dari tanaman. Batik motif Semen bukan termasuk batik motif larangan, sehingga bisa dikenakan oleh masyarakat biasa secara bebas pada berbagai acara dan keperluan sehari-hari. Sesuai dengan artinya, batik motif Semen mempunyai makna filosofi kehidupan yang bersemi, bertumbuh, atau berkembang.

Pada umumnya, batik motif Semen mempunyai ragam hias atau ornament utama yang secara keseluruhan menjadi satu kesatuan untuh membentuk makna tertentu. Ragam hias ornament itu, antara lain Meru, Lidah Api, Baito, Burung, Garuda, atau Rajawali, Pusaka atau Rembulan, Dampar atau Takhta, Bintang, dan Pohon Hayat atau Bumi. Adapun contoh gambar dari salah satu motif Semen adalah sebagai berikut. [7]

Gambar 2.6 Motif Semen

2.1.7 Motif Cuwiri

Dalam Bahasa Jawa, Kata *cuwiri* berarti kecil-kecil. Dengan demikian, batik motif Cuwiri merupakan motif batik dengan pola atau ragam hias kecil-kecil. Ciri khas yang lain dari batik motif Cuwiri adalah penggunaan zat pewarna soga alam dalam pembuatanya.

Batik motif Cuwiri mengandung makna pantas dan dihormati. Pemakai batik motif Cuwiri diharapkan terlihat pantas dan dihormati oleh masyarakat. Sejak kecil masyarakat Jawa sudah memiliki tradisi dan falsafah hidup agar kelak di kemudian hari mendapatkan kemakmuran dan kebaikan. Adapun contoh gambar dari salah satu motif Cuwiri adalah sebagai berikut. [7]

Gambar 2.7 Motif Cuwiri

2.1.8 Motif Tambal

Batik motif Tambak banyak digunakan sebagai kain panjang untuk selimut orang sakit. Konon, kain batik dengan motif Tambal dipercaya dapat mempercepat proses penyembuhan orang sakit. Orang yang tengah sakit jika mengenakan kain batik dengan motif Tambal sebagai selimut, dipercaya akan segera sembuh. Batik dengan motif Tambal dipercaya berfungsi sebagai tolak bala, yaitu dapat menghalangi berbagai pengaruh jahat.

Secara filosofi, batik dengan motif Tambal juga mempunyai arti menambah semangat. Dengan semangat itu, diharapkan muncul harapan baru, sehingga kesembuhan akan cepat terjadi.

Batik motif Tambal juga mengajarkan agar kita sebagai manusia hendaknya selalu berbenah diri dengan semangat memperbaiki diri ini, diharapkan kita dapat menuju kehidupan yang lebih baik lahir maupun batin, sebab, perjalanan hidup manusia tidak luput dari dosa dan kesalahan yang mengharuskan kita untuk selalu berbenah diri. Adapun contoh gambar dari salah satu motif Tambal adalah sebagai berikut. [7]

Gambar 2.8 Motif Tambal

2.1.9 Motif Sekar Jagad

Batik motif Sekar Jagad merupakan batik Petani yang tumbuh dan berkembang di Yogyakarta dan Surakarta sejak abad ke-18. Latar belakang kain yang berawarna putih melambangkan hamparan peta dunia. Keberagaman dunia itu diwujudkan dalam bentuk motif bercorak geometris yang berulang dengan cara ceplok atau dipasangkan secara bersisian.

Makna utama yang terpancar dari batik motif Sekar Jagad adalah keanekaragaman dan keluhuran kehidupan di seluruh dunia, serta kecantikan dan keindahan. Dengan mengenakan batik motif ini diharapkan akan mempesona siapa pun yang melihatnya. Oleh karena itu, batik motif Sekar Jagad banyak digunakan dalam acara pesta pernikahan. Adapun contoh gambar dari salah satu motif Sekar Jagad adalah sebagai berikut. [7]

Gambar 2.9 Motif Sekar Jagad

2.1.10 Motif Sido Mukti

Dalam Bahasa Jawa, *sido mukti* berasal dari kata *sido* yang berarti jadi, atau terus-menerus, serta *mukti* yang berarti mulia dan sejahtera. Dengan demikian batik motif Sido Mukti berarti terus-menerus menjadi mulia dan sejahtera atau hidup berkecukupan.

Batik motif Sido Mukti termasuk jenis batik klasik khas busana keraton. Batik motif ini menggunakan zat pewarna sogan alam. Batik motif Sido Mukti sebenarnya merupakan perkembangan dari batik Sido Mulya dengan latar belakang putih yang berasal dari zaman Kerajaan Mataram di Kartasura.

Batik motif Sido Mukti biasanya dikenakan sebagai busana dalam upacara adat perkawinan pada saat acara resepsi atau *pahargyan*. Makna filosofi yang terkandung dari batik motif ini adalah harapan untuk mendapatkan kemuliaan dan kesejahteraan lahir dan batin. Batik motif ini dikenakan oleh pengantin pria maupun wanita yang dinakaman juga sebagai sawitan (sepasang). Dengan memakai batik motif Sido Mukti diharapkan kedua mempelai akan memperoleh kebahagiaan selama hidupnya. Adapun contoh gambar dari salah satu motif Sido Mukti adalah sebagai berikut. [7]

Gambar 2.10 Motif Sido Mukti

2.1.11 Motif Mega Mendung

Batik motif Mega Mendung telah menjadi salah satu ikon penting Kota Cirebon. Di bandingkan dengan motif batik dari daerah lain, batik motif Mega

Mendung mempunyai keunikan dan ciri khasnya sendiri. Bentuk motif batik khas Kota Udang ini menyerupai kumpulan atau lapisan-lapisan awan. Batik motif Mega Mendung ini terbentuk dari lengkungan-lengkungan kecil yang bergerak membersar keluar, dan akhirnya kembali lagi menjadi lengkungan-lengkungan kecil secara tidak terputus.

Banyak orang berpendapat, batik motif Mega Mendung tercipta akibat pengaruh kebudayaan Tiongkok. Bangsa Tiongkok yang ppada masa lalu seing singgah dan menetap di pelabuhan Muara Jati, Cirebon diduga membawa paham Taoisme. Dalam paham ini, bentuk awan melambangkan dunia atas atau luas dan bebas yang mempunyai makna ketuhanan.

Makna dan filosofi yang terkandung dalam motif Mega Mendung dapat dilihat baik dari pilihan warna maupun bentuk motifnya. Warna biru diselingi merah pada batik motif mega mendung melambangkan karakter masyarakat pesisir yang terbuka dan egaliter namun lugas. Disamping itu, warna biru juga melambangkan warna langit yang luas dan tenang sebagai harapan akan datangnya hujan yang membawa kesuburan dan kehidupan. Adapun contoh gambar dari salah satu motif Mega Mendung adalah sebagai berikut. [7]

Gambar 2.11 Motif Mega Mendung

2.2 Android

Android merupakan sistem operasi berbasis *Linux* yang dirancang untuk perangkat layar sentuh seperti *smartphone* dan *tablet*. Android awalnya dikembangkan oleh Android, Inc., dengan didukung finasial oleh *Google* yang kemudian dibeli pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun

2007, bersamaan dengan didirikannya *Open Handset Alliance*, dari perusahaan-perusahaan perangkat keras, perangkat lunak dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel android pertama kali mulai dijual pada bulan oktober 2008. [8]

2.2.1 Google Play

Google Play merupakan pasar utama untuk menjual dan mendistribusikan aplikasi Android. Sebagai *Open Source*, Google Play memberikan kendali bagaimana untuk menjual produk aplikasi. Karena dapat mendistribusikan langsung kepada pengguna Google play. Serta dapat mendistribuskan secara luas ke semua pasar dan perangkat atau fokus pada segmen perangkat, atau rentang kemampuan perangkat keras tertentu. juga dapat memonetisasi dengan cara yang paling sesuai untuk harga bisnis atau gratis, dengan produk atau langganan dalam aplikasi untuk keterlibatan dan pendapatan tertinggi. Serta memiliki kontrol penuh atas harga untuk aplikasi dan produk dalam aplikasi juga dapat menentukan atau mengubah harga dalam mata uang yang didukung setiap saat. Selain mengembangkan basis untuk pengguna Google play, juga dapat membantu membangun visibilitas dan keterlibatan di seluruh aplikasi dan merek. Seiring bertambahnya popularitas aplikasi, Google Play memberi mereka penempatan yang lebih tinggi dalam grafik dan peringkat "top" mingguan, dan untuk slot promosi aplikasi terbaik di koleksi yang dikuratori. Terinstal di miliaran perangkat Android di seluruh dunia, Google Play dapat menjadi mesin pertumbuhan untuk bisnis. [8]

2.2.2 Sejarah Android

Android, Inc. Didirikan di Palo Alto, California, pada bulan oktober 2003 oleh Andy Rubin (Pendiri Danger), Rich Milner (pendiri Wildfire Cimmunication, Inc.), Nick Sears (mantan Vp T-Mobile) dan Chris White (kepala desain dan pengembangan antarmuka WebTV) untuk mengembangkan “Perangkat *smartphone* yang lebih sadar akan lokasi dan referensi penggunanya”. Tujuan awal pengembangan Android adalah untuk pengembangan sebuah sistem operasi canggih yang diperuntukan bagi kamera digital, namun kemudian disadari bahwa pasar untuk perangkat tersebut tidak cukup besar dan pengembangan Android lalu

dialihkan bagi pasar *smartphone* untuk menyaingi Symbian dan Windows Mobile (Iphone Apple belum dirilis pada saat itu).

Google mengakuisisi Android Inc pada tanggal 17 agustus 2005, menjadikannya sebagai anak perusahaan yang dimiliki oleh Google. Pendiri Android Inc seperti Rubin, Miner dan White tetap bekerja di perusahaan setelah diakuisisi oleh Google. Setelah itu tidak banyak yang diketahui tentang perkembangan Android Inc., namun banyak anggapan yang menyatakan bahwa Google berencana untuk memasuki pasar telepon seluler dengan tindakannya ini. Di Google, tim yang dipimpin oleh Rubin mulai mengembangkan platform perangkat seluler dengan menggunakan *kernel linux*. Google memasarkan platform tersebut kepada produsen perangkat seluler dan operator nirkabel, dengan janji bahwa mereka menyediakan sistem fleksibel dan bisa diperbaharui. Google telah memilih beberapa mitra perusahaan perangkat lunak dan perangkat keras, serta mengisyaratkan kepada operator seluler bahwa kerja sama ini terbuka bagi siapapun yang ingin berpartisipasi.

Spekulasi tentang niat Google untuk memasuki pasar komunikasi seluler terus berkembang hingga bulan desember 2006. BBC dan Wall Street Journal melaporkan bahwa Google sedang bekerja keras untuk menyertakan aplikasi dan mesin pencariannya di perangkat seluler. Berbagai media cetak dan media daring mengabarkan bahwa Google sedang mengembangkan perangkat seluler dengan merek Google. Beberapa di antaranya berspekulasi bahwa Google telah menentukan spesifikasi teknisnya, termasuk produsen telepon seluler dan operator jaringan. Pada bulan Desember 2007, InformationWeek melaporkan bahwa Google telah mengajukan beberapa aplikasi paten di bidang telepon seluler.

Pada tanggal 5 November 2007, Open Handset Alliance (OHA) didirikan. OHA adalah konsorsium dari perusahaan-perusahaan teknologi seperti Google, produsen perangkat seluler seperti HTC, Sony dan Samsung, operator nirkabel seperti Sprint Nextel dan T-Mobile, serta produsen chipset seperti Qualcomm dan Texas Instruments. OHA sendiri bertujuan untuk mengembangkan standar terbuka bagi perangkat seluler. Saat itu, Android diresmikan sebagai produk pertamanya; sebuah platform perangkat seluler yang menggunakan kernel Linux versi 2.6.

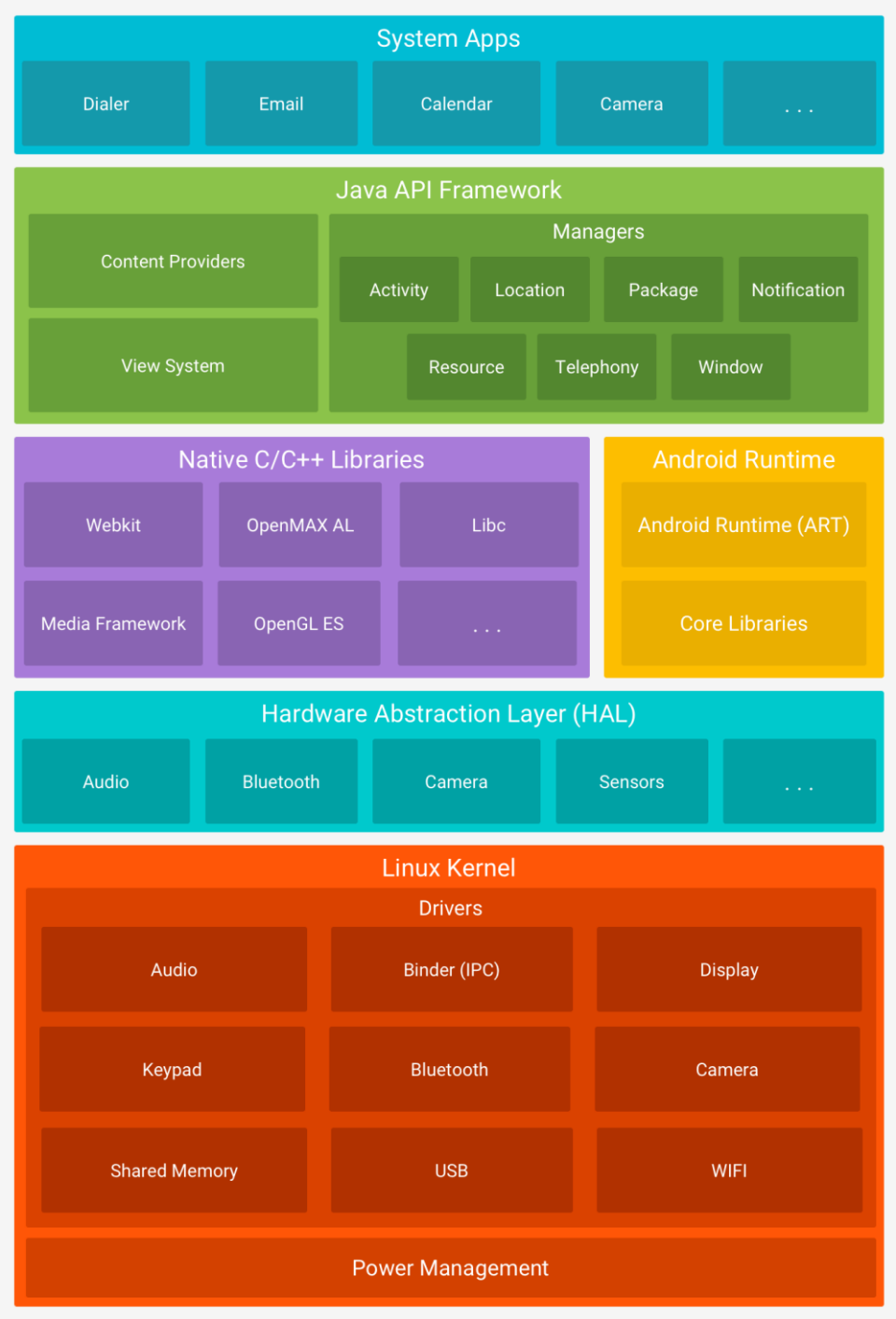
Telepon seluler komersial pertama yang menggunakan sistem operasi Android adalah HTC Dream, yang diluncurkan pada 22 Oktober 2008.

Pada tahun 2010, Google merilis seri Nexus; perangkat telepon pintar dan tablet dengan sistem operasi Android yang diproduksi oleh mitra produsen telepon seluler seperti HTC, LG, dan Samsung. HTC bekerja sama dengan Google dalam merilis produk telepon pintar Nexus pertama, yakni Nexus One. Seri ini telah diperbarui dengan perangkat yang lebih baru, misalnya telepon pintar Nexus 4 dan tablet Nexus 10 yang diproduksi oleh LG dan Samsung. Pada 15 Oktober 2014, Google mengumumkan Nexus 6 dan Nexus 9 yang diproduksi oleh Motorola dan HTC. Pada 13 Maret 2013, Larry Page mengumumkan dalam postingan blognya bahwa Andy Rubin telah pindah dari divisi Android untuk mengerjakan proyek-proyek baru di Google. Ia digantikan oleh Sundar Pichai, yang sebelumnya menjabat sebagai kepala divisi Google Chrome, yang mengembangkan Chrome OS.

Sejak tahun 2008, Android secara bertahap telah melakukan sejumlah pembaruan untuk meningkatkan kinerja sistem operasi, menambahkan fitur baru, dan memperbaiki bug yang terdapat pada versi sebelumnya. Setiap versi utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut atau camilan bergula; misalnya, versi 1.5 bernama Cupcake, yang kemudian diikuti oleh versi 1.6 Donut. Versi terbaru adalah 5.0 Lollipop, yang dirilis pada 15 Oktober 2014. [8]

2.2.3 Arsitektur Android

Android adalah tumpukan perangkat lunak berbasis Linux sumber terbuka yang dibuat untuk berbagai perangkat dan faktor bentuk. Diagram berikut menunjukkan komponen besar dari platform Android.



Gambar 2.12 Arsitektur Android

1. **Linux Kernel**

Fondasi platform Android adalah kernel Linux. Sebagai contoh, *Android Runtime* (ART) bergantung pada kernel Linux untuk fungsionalitas dasar seperti *threading* dan manajemen memori tingkat rendah.

Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

1. **Hardware Abstraction Layer (Hal)**

   Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

2. **Android Runtime**

   Untuk perangkat yang menjalankan Android versi 5.0 (API level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap *Android Runtime* (ART). ART ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file DEX, format bytecode yang didesain khusus untuk Android yang dioptimalkan untuk footprint memori minimal. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke *bytecode* DEX, yang dapat berjalan pada platform Android. Beberapa fitur utama ART mencakup:

   - Kompilasi mendahului waktu (AOT) dan tepat waktu (JIT)
   - Pengumpulan sampah (GC) yang dioptimalkan
   - Dukungan debug yang lebih baik, mencakup profiler sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

   Sebelum ke Android versi 5.0 (API level 21), Dalvik adalah waktu proses Android. Jika aplikasi Anda berjalan baik pada ART, semestinya berfungsi baik juga pada Dalvik, tetapi mungkin tidak sebaliknya.

Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8, yang digunakan kerangka kerja Java API.

3. **Pustaka C/C++ Asli**

Banyak komponen dan layanan sistem Android inti seperti ART dan HAL dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. Platform Android memungkinkan kerangka kerja Java API mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses OpenGL ES melalui kerangka kerja Java OpenGL API Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda. Jika Anda mengembangkan aplikasi yang memerlukan kode C atau C++, Anda bisa menggunakan Android NDK untuk mengakses beberapa pustaka platform asli langsung dari kode asli.

4. **Kerangka Kerja Java API**

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

- Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan.
- Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout.
- Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status.
- Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum.

- Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android.

5. **Aplikasi Sistem**

Android dilengkapi dengan serangkaian aplikasi inti untuk email, perpesanan SMS, kalender, menjelajahi internet, kontak, dll. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi Settings sistem). Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh developer dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan SMS, Anda tidak perlu membangun fungsionalitas tersebut sendiri—sebagai gantinya Anda bisa menjalankan aplikasi SMS mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan. [9]

2.2.4 Siklus Hidup Android

Saat pengguna menavigasi, keluar, dan kembali ke aplikasi, Aktivitas akan muncul dalam transisi di aplikasi melalui berbagai status dalam siklus hidupnya. Kelas *Activity* menyediakan sejumlah panggilan balik yang memungkinkan aktivitas mengetahui bahwa suatu keadaan telah berubah bahwa sistem membuat, menghentikan, atau melanjutkan suatu aktivitas, atau menghancurkan proses di mana aktivitas berada.

Dalam metode callback siklus hidup, dapat mendeklarasikan bagaimana aktivitas berperilaku ketika pengguna meninggalkan dan masuk kembali ke aktivitas. Misalnya, jika membuat pemutar video streaming, dapat menjeda video dan memutuskan koneksi jaringan ketika pengguna beralih ke aplikasi lain. Ketika pengguna kembali, dapat menyambung kembali ke jaringan dan memungkinkan

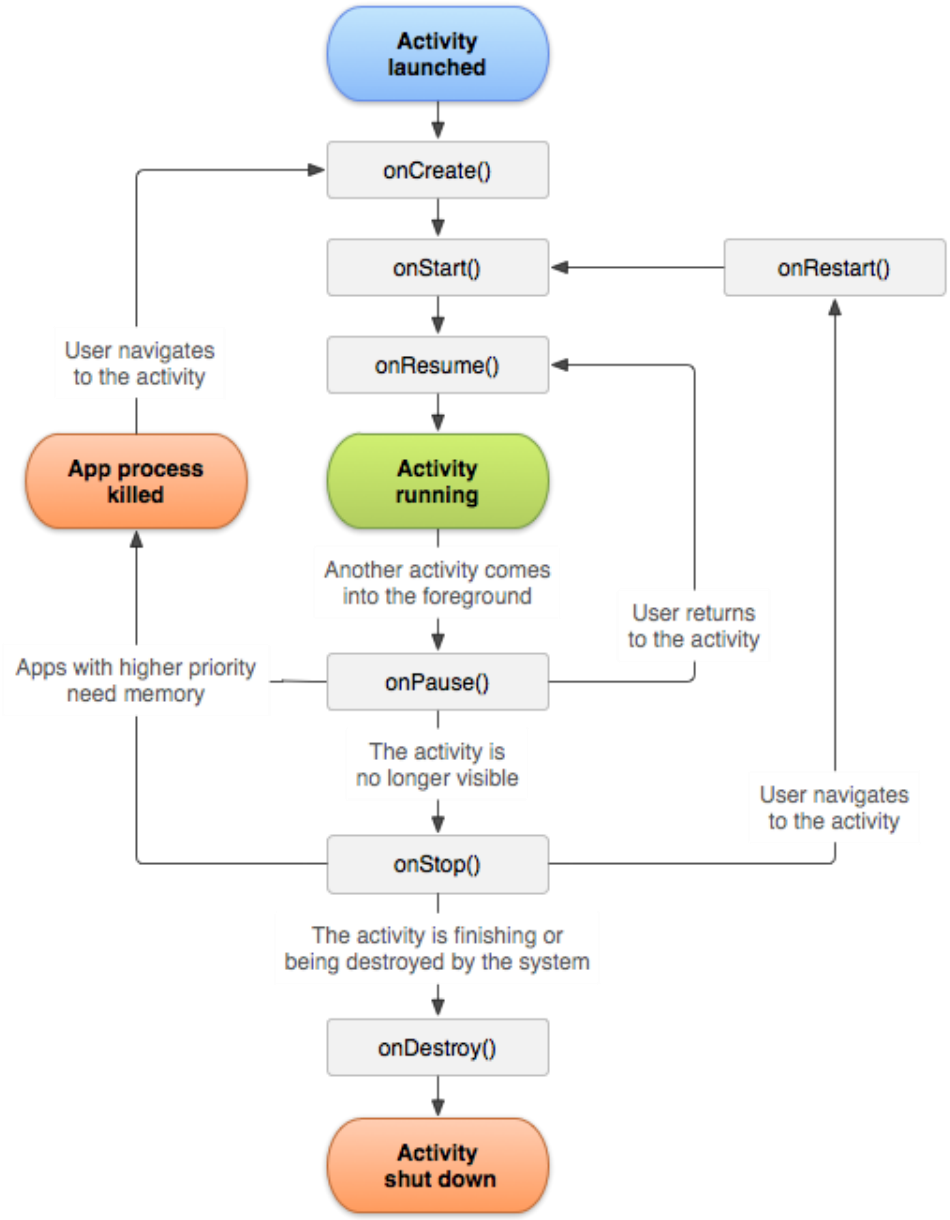
pengguna untuk melanjutkan video dari tempat yang sama. Dengan kata lain, setiap panggilan balik memungkinkan melakukan pekerjaan tertentu yang sesuai dengan perubahan kondisi yang diberikan. Melakukan pekerjaan yang tepat di waktu yang tepat dan menangani transisi dengan benar membuat aplikasi lebih kuat dan berkinerja. Misalnya, penerapan panggilan balik siklus hidup yang baik dapat membantu memastikan bahwa aplikasi menghindari: callback yang bagus dapat membantu memastikan aplikasi dihindari:

- *Crashing* jika pengguna menerima panggilan telepon atau beralih ke aplikasi lain saat menggunakan aplikasi.
- *Consuming* sumber daya sistem yang berharga saat pengguna tidak aktif menggunakannya
- *Losing* kemajuan pengguna jika mereka meninggalkan aplikasi dan kembali lagi nanti.
- *Crashing* atau kehilangan kemajuan pengguna saat layar berputar antara orientasi landscape dan portrait.

Dokumen ini menjelaskan siklus hidup aktivitas secara terperinci. Dokumen dimulai dengan menjelaskan paradigma siklus hidup. Selanjutnya, ini menjelaskan masing-masing panggilan balik: apa yang terjadi secara internal saat mereka mengeksekusi, dan apa yang harus Anda terapkan selama mereka. Ini kemudian secara singkat memperkenalkan hubungan antara keadaan aktivitas dan kerentanan suatu proses untuk dibunuh oleh sistem. Terakhir, ini membahas beberapa topik yang terkait dengan transisi antara status aktivitas. [9]

2.2.5 Konsep Siklus Hidup Aktivitas

Untuk menavigasi transisi antara tahap siklus hidup aktivitas, kelas Activity menyediakan set inti enam panggilan balik yaitu onCreate (), onStart (), onResume (), onPause (), onStop (), dan onDestroy (). Sistem memanggil masing-masing panggilan balik ini saat aktivitas memasuki keadaan baru.



Gambar 2.13 Sebuah ilustrasi sederhana dari siklus hidup aktivitas

Ketika pengguna mulai meninggalkan aktivitas, sistem memanggil metode untuk membongkar aktivitas. Dalam beberapa kasus, pembongkaran ini hanya sebagian; aktivitas masih berada dalam memori (seperti ketika pengguna beralih ke aplikasi lain), dan masih dapat kembali ke latar depan. Jika pengguna kembali ke aktivitas itu, aktivitas dilanjutkan dari tempat pengguna tadi pergi. Kemungkinan sistem untuk membunuh proses tertentu — bersama dengan aktivitas di dalamnya — tergantung pada keadaan aktivitas pada saat itu. Status aktivitas dan ejeksi dari

memori memberikan lebih banyak informasi tentang hubungan antara status dan kerentanan terhadap ejeksi.

Bergantung pada kerumitan aktivitas, mungkin tidak perlu menerapkan semua metode siklus hidup. Namun, penting untuk memahami masing-masing dan menerapkannya yang memastikan aplikasi dapat berperilaku seperti yang diharapkan pengguna.

Bagian selanjutnya dari dokumen ini memberikan detail tentang panggilan balik yang Anda gunakan untuk menangani transisi antar negara.

2.2.6 Siklus Hidup Callback

Bagian ini memberikan informasi konseptual dan implementasi tentang metode panggilan balik yang digunakan selama siklus hidup aktivitas. Beberapa tindakan, seperti memanggil setContentView (), termasuk dalam metode siklus hidup aktivitas itu sendiri. Namun, kode yang menerapkan tindakan komponen dependen harus ditempatkan dalam komponen itu sendiri. Untuk mencapai ini, Anda harus membuat komponen yang sadar siklus hidup. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen dependen Anda sadar siklus

onCreate()

Anda harus menerapkan panggilan balik ini, yang menyala saat sistem pertama kali membuat aktivitas. Pada pembuatan aktivitas, aktivitas memasuki kondisi Dibuat. Dalam metode onCreate(), Anda melakukan logika *startup* aplikasi dasar yang harus terjadi hanya sekali selama seumur hidup aktivitas. Misalnya, implementasi onCreate() mungkin mengikat data ke daftar, mengaitkan aktivitas dengan *ViewModel*, dan membuat instance beberapa variabel lingkup kelas. Metode ini menerima parameter SavedInstanceState, yang merupakan objek bundel yang berisi keadaan aktivitas yang disimpan sebelumnya. Jika aktivitas belum pernah ada sebelumnya, nilai objek Bundle adalah nol.

Contoh berikut dari metode onCreate() menunjukkan pengaturan mendasar untuk aktivitas, seperti mendeklarasikan antarmuka pengguna (didefinisikan dalam file tata letak XML), mendefinisikan variabel anggota, dan mengkonfigurasi

beberapa UI. Dalam contoh ini, file tata letak XML ditentukan dengan meneruskan ID sumber daya file R.layout.main_activity ke setContentView ().

```
TextView textView;

// some transient state for the activity instance
String gameState;

@Override
public void onCreate(Bundle savedInstanceState) {
    // call the super class onCreate to complete the creation of activity like
    // the view hierarchy
    super.onCreate(savedInstanceState);

    // recovering the instance state
    if (savedInstanceState != null) {
        gameState = savedInstanceState.getString(GAME_STATE_KEY);
    }

    // set the user interface layout for this activity
    // the layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity);

    // initialize member TextView so we can manipulate it later
    textView = (TextView) findViewById(R.id.text_view);
}

// This callback is called only when there is a saved instance that is previously saved by using
// onSaveInstanceState(). We restore some state in onCreate(), while we can optionally restore
// other state here, possibly usable after onStart() has completed.
// The savedInstanceState Bundle is same as the one used in onCreate().
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    textView.setText(savedInstanceState.getString(TEXT_VIEW_KEY));
}

// invoked when the activity may be temporarily destroyed, save the instance state here
@Override
public void onSaveInstanceState(Bundle outState) {
    outState.putString(GAME_STATE_KEY, gameState);
    outState.putString(TEXT_VIEW_KEY, textView.getText());

    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState);
}
```

Gambar 2.14 Metode *onCreate()*

Sebagai alternatif untuk mendefinisikan file XML dan meneruskannya ke setContentView (), Anda bisa membuat objek View baru dalam kode aktivitas Anda dan membangun hierarki tampilan dengan memasukkan Tampilan baru ke dalam ViewGroup. Anda kemudian menggunakan tata letak itu dengan melewatkan root ViewGroup ke setContentView(). Untuk informasi lebih lanjut tentang membuat antarmuka pengguna, lihat dokumentasi Antarmuka Pengguna. Aktivitas Anda tidak berada di negara Dibuat. Setelah metode onCreate () menyelesaikan eksekusi, aktivitas memasuki keadaan awal, dan sistem memanggil metode onStart () dan onResume () secara berurutan. Bagian selanjutnya menjelaskan panggilan balik onStart ().

onStart()

Ketika aktivitas memasuki kondisi Mulai, sistem memanggil panggilan balik ini. Panggilan onStart() membuat aktivitas terlihat oleh pengguna, saat aplikasi mempersiapkan aktivitas untuk memasuki latar depan dan menjadi interaktif. Misalnya, metode ini adalah tempat aplikasi menginisialisasi kode yang mengelola UI. Metode onStart() selesai dengan sangat cepat dan, seperti pada kondisi Dibuat, aktivitas tidak tetap berada dalam status Mulai. Setelah panggilan balik ini selesai, aktivitas memasuki status Dilanjutkan, dan sistem memanggil metode onResume().

onResume()

Ketika aktivitas memasuki keadaan dilanjutkan, ia datang ke latar depan, dan kemudian sistem memanggil panggilan balik onResume (). Ini adalah keadaan di mana aplikasi berinteraksi dengan pengguna. Aplikasi tetap dalam kondisi ini sampai terjadi sesuatu untuk mengambil fokus dari aplikasi. Peristiwa semacam itu mungkin, misalnya, menerima panggilan telepon, pengguna bernavigasi ke aktivitas lain, atau layar perangkat mati. Saat aktivitas beralih ke status yang dilanjutkan, komponen sadar siklus apa pun yang terkait dengan siklus hidup aktivitas akan menerima acara ON_RESUME. Di sinilah komponen siklus hidup dapat mengaktifkan fungsionalitas apa pun yang perlu dijalankan saat komponen terlihat dan di latar depan, seperti memulai pratinjau kamera. Ketika suatu peristiwa interupsi terjadi, aktivitas memasuki keadaan dijeda, dan sistem memanggil panggilan balik onPause (). Jika aktivitas kembali ke status Lanjutkan dari kondisi Jeda, sistem sekali lagi memanggil metode onResume (). Karena alasan ini, Anda harus menerapkan onResume () untuk menginisialisasi komponen yang Anda lepaskan selama onPause (), dan melakukan inisialisasi lainnya yang harus terjadi setiap kali aktivitas memasuki keadaan dilanjutkan.:

```
public class CameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    public void initializeCamera() {
        if (camera == null) {
            getCamera();
        }
    }

    ...
}
```

Gambar 2.15 Metode *onResume()*

Kode di atas menempatkan kode inisialisasi kamera dalam komponen yang menyadari siklus hidup. Anda bisa menempatkan kode ini langsung ke *callback* daur hidup aktivitas seperti onStart() dan onStop() tetapi ini tidak dianjurkan. Menambahkan logika ini ke dalam komponen yang mandiri dan sadar siklus memungkinkan Anda untuk menggunakan kembali komponen tersebut di berbagai aktivitas tanpa harus menduplikasi kode. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen yang sadar siklus..

onPause()

Sistem memanggil metode ini sebagai indikasi pertama bahwa pengguna meninggalkan aktivitas Anda (meskipun itu tidak selalu berarti aktivitas sedang dihancurkan); itu menunjukkan bahwa aktivitas tidak lagi di latar depan (meskipun mungkin masih terlihat jika pengguna berada dalam mode multi-jendela). Gunakan metode onPause () untuk menjeda atau menyesuaikan operasi yang tidak boleh dilanjutkan (atau harus dilanjutkan dalam jumlah sedang) saat Aktivitas dalam keadaan Jeda, dan Anda berharap untuk melanjutkan segera. Ada beberapa alasan mengapa suatu kegiatan dapat memasuki kondisi ini. Sebagai contoh:

- Beberapa acara mengganggu eksekusi aplikasi, seperti yang dijelaskan di bagian onResume (). Ini adalah kasus yang paling umum.
- Di Android 7.0 (API level 24) atau lebih tinggi, beberapa aplikasi berjalan dalam mode multi-jendela. Karena hanya satu aplikasi (windows) yang memiliki fokus kapan saja, sistem menjeda semua aplikasi lain.

- Aktivitas semi-transparan baru (seperti dialog) terbuka. Selama aktivitas masih terlihat sebagian tetapi tidak dalam fokus, itu tetap dijeda.

Anda juga dapat menggunakan metode onPause () untuk melepaskan sumber daya sistem, menangani sensor (seperti GPS), atau sumber daya apa pun yang dapat memengaruhi masa pakai baterai saat aktivitas Anda dijeda dan pengguna tidak membutuhkannya. Namun, seperti yang disebutkan di atas di bagian onResume(), aktivitas yang Dijeda mungkin masih sepenuhnya terlihat jika dalam mode multi-jendela. Karena itu, Anda harus mempertimbangkan menggunakan onStop() daripada onPause() untuk sepenuhnya melepaskan atau menyesuaikan sumber daya dan operasi terkait UI untuk lebih mendukung mode multi-jendela.

```
public class JavaCameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
    public void releaseCamera() {
        if (camera != null) {
            camera.release();
            camera = null;
        }
    }

    ...
}
```

Gambar 2.16 Metode *onPause()*

Penyelesaian metode onPause () tidak berarti bahwa aktivitas meninggalkan status Jeda. Sebaliknya, aktivitas tetap dalam kondisi ini sampai aktivitas dilanjutkan atau menjadi sama sekali tidak terlihat oleh pengguna. Jika aktivitas dilanjutkan, sistem sekali lagi memanggil panggilan balik onResume (). Jika aktivitas kembali dari status Jeda ke status Dilanjutkan, sistem menyimpan instance instance tetap dalam memori, mengingat instance tersebut ketika sistem memanggil onResume (). Dalam skenario ini, Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Jika aktivitas menjadi benar-benar tidak terlihat, sistem memanggil onStop (). Bagian selanjutnya membahas panggilan balik onStop ().

onStop()

Ketika aktivitas Anda tidak lagi terlihat oleh pengguna, itu telah memasuki status Berhenti, dan sistem memanggil panggilan balik onStop (). Ini dapat terjadi, misalnya, ketika aktivitas yang baru diluncurkan mencakup seluruh layar. Sistem juga dapat memanggil onStop () ketika aktivitas telah selesai berjalan, dan akan dihentikan.

Dalam metode onStop (), aplikasi harus melepaskan atau menyesuaikan sumber daya yang tidak diperlukan saat aplikasi tidak terlihat oleh pengguna. Misalnya, aplikasi Anda dapat menjeda animasi atau beralih dari pembaruan lokasi berbutir halus ke berbutir kasar. Menggunakan onStop () dan bukan onPause () memastikan bahwa pekerjaan terkait UI berlanjut, bahkan ketika pengguna melihat aktivitas Anda dalam mode multi-jendela.

Anda juga harus menggunakan onStop () untuk melakukan operasi shutdown yang relatif intensif CPU. Misalnya, jika Anda tidak dapat menemukan waktu yang lebih tepat untuk menyimpan informasi ke database, Anda mungkin melakukannya selama onStop (). Contoh berikut menunjukkan implementasi onStop () yang menyimpan konten catatan konsep ke penyimpanan persisten:

```
@Override
protected void onStop() {
    // call the superclass method first
    super.onStop();

    // save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    // do this update in background on an AsyncQueryHandler or equivalent
    asyncQueryHandler.startUpdate (
            mToken,  // int token to correlate calls
            null,    // cookie, not used here
            uri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
    );
}
```

Gambar 2.17 Metode *onStop()*

Ketika aktivitas Anda memasuki status Berhenti, objek Aktivitas disimpan di dalam memori: Objek menyimpan semua informasi status dan anggota, tetapi tidak

dilampirkan ke manajer jendela. Ketika aktivitas dilanjutkan, aktivitas mengingat informasi ini. Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Sistem juga melacak keadaan saat ini untuk setiap objek Lihat dalam tata letak, jadi jika pengguna memasukkan teks ke dalam widget EditText, konten tersebut dipertahankan sehingga Anda tidak perlu menyimpan dan mengembalikannya.

onDestroy()

onDestroy () dipanggil sebelum aktivitas dihancurkan. Sistem memanggil panggilan balik ini karena:

- aktivitas sedang selesai (karena pengguna benar-benar mengabaikan aktivitas atau karena selesai () dipanggil pada aktivitas).
- sistem sementara menghancurkan aktivitas karena perubahan konfigurasi (seperti rotasi perangkat atau mode multi-jendela).

Jika aktivitas selesai, onDestroy() adalah panggilan balik siklus hidup terakhir yang diterima aktivitas. Jika onDestroy() dipanggil sebagai hasil dari perubahan konfigurasi, sistem segera membuat instance aktivitas baru dan kemudian memanggil onCreate() pada instance baru dalam konfigurasi baru. Callback onDestroy() harus melepaskan semua sumber daya yang belum dirilis oleh callback sebelumnya seperti onStop(). [9]

2.2.7 Android Studio

Android Studio adalah IDE resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain kemampuan yang Anda harapkan dari IntelliJ, Android Studio menawarkan:

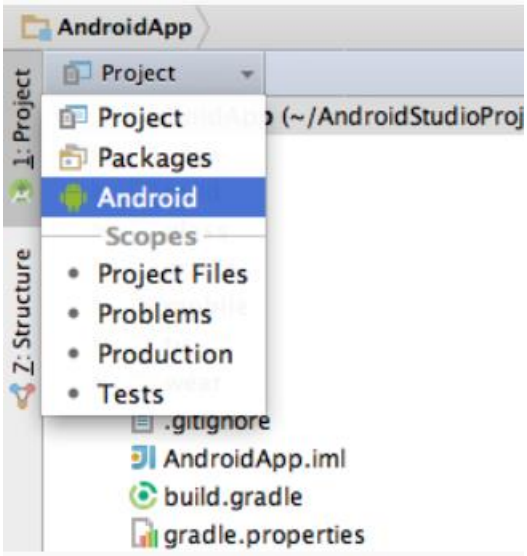
- Sistem pembangunan berbasis Gradle fleksibel.
- Buat varian dan beberapa pembuatan file apk.
- Templat kode untuk membantu Anda membangun fitur aplikasi umum.
- Editor tata letak kaya dengan dukungan untuk mengedit tema drag and drop.
- Alat serat untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya.
- Kemampuan ProGuard dan penandatanganan aplikasi.

- Dukungan bawaan untuk Google Cloud Platform, membuatnya mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine.

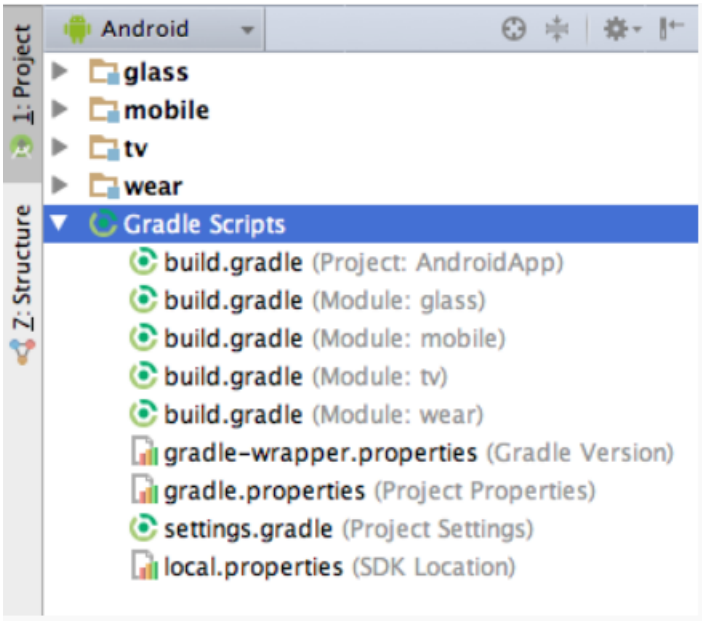
1. **Struktur Proyek**

Secara default, Android Studio menampilkan file profil Anda dalam tampilan proyek Android. Tampilan ini menunjukkan versi rata dari struktur proyek Anda yang menyediakan akses cepat ke file sumber utama proyek Android dan membantu Anda bekerja dengan sistem build berbasis Gradle yang baru. Tampilan proyek Android:

- Grup file build untuk semua modul di tingkat atas hirarki proyek.
- Memperlihatkan direktori sumber paling penting di tingkat atas hirarki modul.
- Grup semua file manifes untuk setiap modul.
- Memperlihatkan file sumber daya dari semua set sumber Gradle.
- Grup file sumber daya untuk berbagai lokal, orientasi, dan jenis layar dalam satu grup per jenis sumber daya



Gambar 2.18 File proyek di tampilan Android



Gambar 2.19 File pembuatan proyek Android

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

- Modul aplikasi Android
- Modul Pustaka
- Modul Google *App Engine*

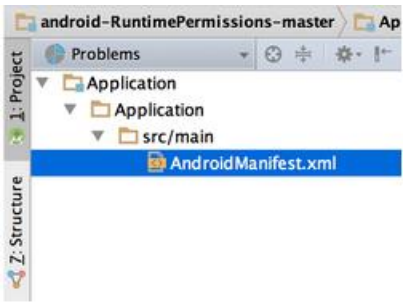
Secara default, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 1. Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke file sumber utama proyek Anda.

Semua file versi terlihat di bagian atas di bawah **Gradle Scripts** dan masing-masing modul aplikasi berisi folder berikut:

- manifests: Berisi file *AndroidManifest.xml.*
- java: Berisi file kode sumber Java, termasuk kode pengujian JUnit.
- res: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.

Sruktur proyek Android pada disk berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih **Project** dari menu tarik turun **Project** (dalam gambar 1, struktur ditampilkan sebagai **Android**).

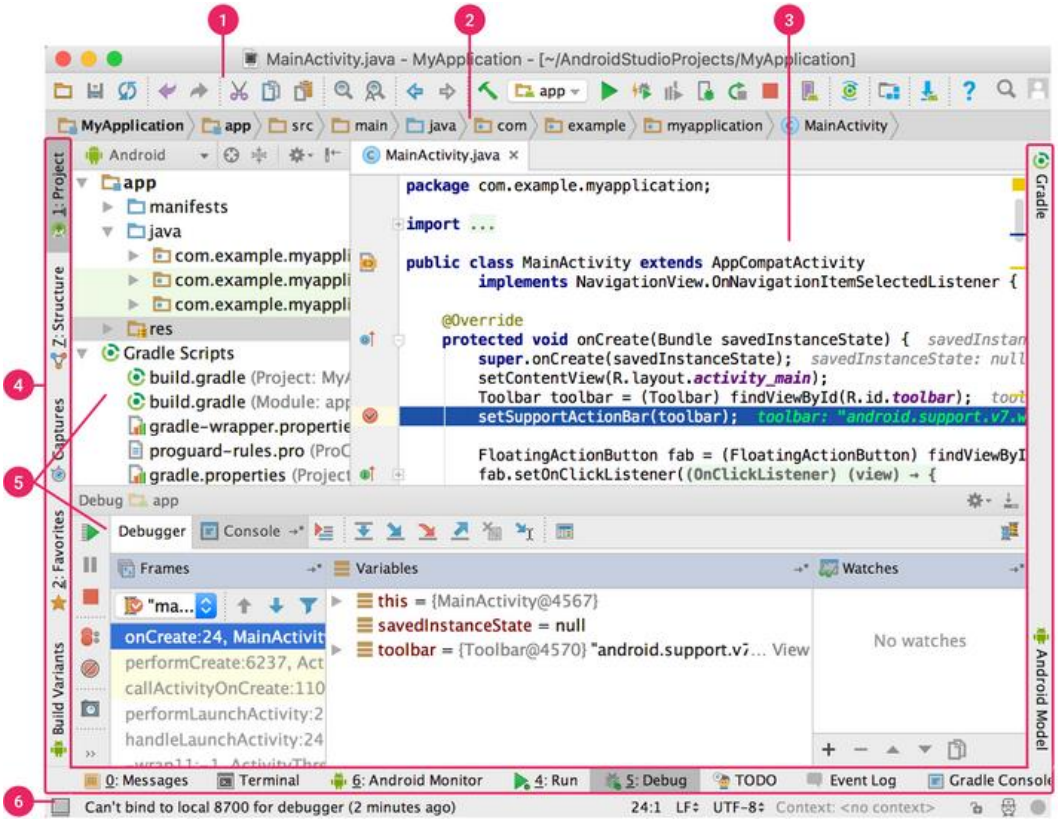
Anda juga bisa menyesuaikan tampilan file proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan **Problems** dari tampilan proyek Anda akan menampilkan tautan ke file sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen XML tidak ada dalam file tata letak.



Gambar 2.20 File proyek di tampilan *Problem*

2. **Antarmuka Pengguna**

Jendela utama Android Studio terdiri dari beberapa bidang logika yang diidentifikasi dalam gambar berikut.



Gambar 2.21 Jendela utama Android Studio

1) Bilah alat memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.
2) Bilah navigasi membantu Anda bernavigasi di antara proyek dan membuka file untuk diedit. Bilah ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela Project.
3) Jendela editor adalah tempat Anda membuat dan memodifikasi kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor menampilkan Layout Editor.
4) Bilah jendela alat muncul di luar jendela IDE dan berisi tombol yang memungkinkan Anda meluaskan atau menciutkan jendela alat individual.
5) Jendela alat memberi Anda akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Anda bisa meluaskan dan juga menciutkannya.
6) Bilah status menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.

Anda bisa menata jendela utama untuk memberi Anda ruang layar yang lebih luas dengan menyembunyikan atau memindahkan bilah alat dan jendela alat. Anda juga bisa menggunakan pintasan keyboard untuk mengakses sebagian besar fitur IDE.

Anda dapat menelusuri seluruh kode sumber, basis data, tindakan, elemen antarmuka pengguna, dan seterusnya setiap saat dengan menekan tombol Shift dua kali, atau mengeklik kaca pembesar di sudut kanan atas dari jendela Android Studio. Ini akan sangat berguna misalnya saat Anda mencoba menemukan tindakan IDE tertentu yang Anda lupakan cara memicunya. [10]

Jendela Alat

Dari pada menggunakan perspektif yang sudah diatur sebelumnya, Android Studio mengikuti konteks Anda dan secara otomatis memunculkan jendela alat yang relevan saat Anda bekerja. Secara default, alat yang tersering dipakai akan disematkan ke bilah jendela alat di tepi jendela aplikasi.

- Untuk meluaskan atau menciutkan jendela alat, klik nama alat di bilah jendela alat. Anda juga bisa menyeret, menyematkan, melampirkan, dan melepaskan jendela alat.
- Untuk kembali ke tata letak jendela alat default saat ini, klik *Window* > *Restore Default Layout* atau sesuaikan tata letak default Anda dengan mengeklik *Window* > *Store* Current Layout as Default.
- Untuk kembali ke tata letak jendela alat default saat ini, klik **Window > Restore Default Layout** atau sesuaikan tata letak default Anda dengan mengeklik **Window > Store Current Layout as Default**.
- Untuk menampilkan atau menyembunyikan bilah jendela alat, klik ikon jendela di sudut kiri bawah jendela Android Studio
- Untuk menemukan jendela alat tertentu, arahkan ke atas ikon jendela dan pilih jendela alat tersebut dari menu.

Anda juga bisa menggunakan pintasan keyboard untuk membuka jendela alat. Tabel 2.1 mencantumkan pintasan jendela paling umum.

Tabel 2.1 Keyboard shortcut ke beberapa jendela alat yang penting

Jendela Alat	Windows dan Linux	Mac
Proyek	Alt+1	Command+1
Kontrol Versi	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Android Monitor	Alt+6	Command+6
Kembali ke Editor	Esc	Esc
Menyembunyikan Semua Jendela Alat	Control+Shift+F12	Esc

Jika Anda ingin menyembunyikan semua bilah alat, jendela alat, dan tab editor, klik *View* > *Enter Distraction Free Mode*. Ini akan mengaktifkan *Distraction Free*

Mode. Untuk keluar dari *Distraction Free Mode*, klik *View* > *Exit Distraction Free Mode*.

Anda bisa menggunakan *Speed Search* untuk menelusuri dan memfilter di dalam sebagian besar jendela alat dalam Android Studio. Untuk menggunakan *Speed Search*, pilih jendela alat lalu ketik kueri penelusuran Anda.

Pelengkapan Kode

Android Studio memiliki tiga jenis pelengkapan kode, yang bisa anda akses memakai pintasan keyboard.

Tabel 2.2 Keyboard shortcut untuk pelengkapan code

Tipe	Keterangan	Windows dan Linux	Mac
Pelengkapan Dasar	Menampilkan saran dasar untuk variabel, tipe, metode, ekspresi, dan seterusnya. Jika Anda memanggil pelengkapan dasar dua kali secara berturutturut, Anda melihat lebih banyak hasil, termasuk anggota pribadi dan anggota statis yang tidak diimpor.	Control+Space	Control+Space
Pelengkapan Cerdas	Menampilkan opsi relevan berdasarkan konteks. Pelengkapan cerdas mengetahui tipe yang diharapkan dan alur data. Jika Anda memanggil Pelengkapan Cerdas dua kali berturut-	Control+Shift+Space	Control+Shift+Space

	turut, Anda akan melihat lebih banyak hasil, termasuk rantai.		
Pelengkapan Pernyataan	Membantu Anda melengkapi pernyataan saat ini, menambahkan tanda kurung, tanda kurung siku, tanda kurung kurawal, pemformatan, dsb.	Control+Shift+Enter	Shift+Command+Enter

Anda juga bisa melakukan perbaikan cepat dan menunjukkan tindakan maksud Anda dengan menekan Alt+Enter.

Navigasi

Berikut beberapa tip untuk membantu Anda menjelajahi di dalam Android Studio.

- Beralih antar file yang baru saja diakses menggunakan tindakan *Recent Files*. Tekan *Control+E* (*Command+E* pada Mac) untuk memunculkan tindakan *Recent Files*. Secara default, akses yang terakhir dipilih. Anda juga bisa mengakses jendela alat mana saja melalui kolom kiri dalam tindakan ini.
- Tampilkan struktur file saat ini menggunakan tindakan *File Structure*. Munculkan tindakan *File Structure* dengan menekan *Control+F12* (*Command+F12* pada Mac). Menggunakan tindakan ini, Anda bisa menavigasi dengan cepat ke bagian mana pun dari file Anda saat ini.
- Telusuri dan masuk ke kelas tertentu di proyek menggunakan *tindakan Navigate to Class*. Munculkan tindakan dengan menekan *Control+N* (*Command+O* pada Mac). Navigasikan ke Kelas yang mendukung ekspresi canggih, termasuk *CamelHumps*, jalur, baris menavigasi ke, nama tengah pencocokan, dan banyak lagi. Jika Anda memanggilnya dua kali berturut-turut, hasil dari kelas proyek akan ditampilkan.

- Masuk ke file atau folder menggunakan tindakan "*Navigate to File**. Munculkan tindakan *Navigate to File* dengan menekan *Control+Shift+N* (*Command+Shift+O* pada Mac). Untuk menelusuri folder dan bukan file, tambahkan / di akhir ekspresi Anda.
- Masuk ke metode atau bidang menurut nama menggunakan tindakan *Navigate to Symbol*. Munculkan tindakan *Navigate to Symbol* dengan menekan *Control+Shift+Alt+N* (*Command+Shift+Alt+O* pada Mac).
- Temukan semua bagian kode yang merujuk kelas, metode, bidang, parameter, atau pernyataan di posisi kursor saat ini dengan menekan Alt+F7

Gaya dan Pemformatan

Saat Anda mengedit, Android Studio otomatis menerapkan pemformatan dan gaya seperti yang ditetapkan dalam setelan gaya kode. Anda dapat menyesuaikan setelan gaya kode dengan bahasa pemrograman, termasuk menetapkan konvensi untuk tab dan inden, spasi, pembungkusan dan tanda kurung kurawal, dan baris kosong. Untuk menyesuaikan setelan gaya kode, *klik File > Settings > Editor > Code Style (Android Studio > Preferences > Editor > Code Style* pada Mac.) Meski IDE otomatis menerapkan pemformatan saat Anda bekerja, Anda juga dapat secara eksplisit memanggil tindakan Reformat Code dengan menekan *Control+Alt+L* (*Opt+Command+L* pada Mac), atau inden otomatis semua baris dengan menekan *Control+Alt+I* (*Alt+Option+I* pada Mac).

Dasar-dasar Kontrol Versi

Android Studio mendukung berbagai versi sistem kontrol, termasuk *Git, GitHub, CVS, Mercurial, Subversion*, dan Penyimpanan *Google Cloud Source*.

Setelah mengimpor aplikasi Anda ke dalam Android Studio, gunakan opsi menu Android Studio VCS untuk mengaktifkan dukungan VCS bagi sistem kontrol versi yang diinginkan, membuat penyimpanan, mengimpor file baru ke dalam kontrol versi, dan melakukan pengoperasian kontrol versi lainnya:

- Dari menu Android Studio VCS, klik *Enable Version Control Integration*.
- Dari menu tarik-turun, pilih sistem kontrol versi yang terkait dengan akar proyek, lalu klik OK.

- Menu VCS sekarang menunjukkan sejumlah opsi kontrol versi berdasarkan sistem yang Anda pilih.

2.2.8 Sistem Versi Gradle

Android Studio menggunakan Gradle sebagai dasar sistem versi, dengan kemampuan khusus Android yang disediakan oleh Plugin Android untuk Gradle. Sistem ini bisa dijalankan sebagai alat terpadu dari menu Android Studio dan secara independen dari baris perintah. Anda bisa menggunakan fitur-fitur sistem versi untuk melakukan yang berikut:

- Menyesuaikan, mengonfigurasi, dan memperluas proses pembangunan.
- Membuat beberapa APK untuk aplikasi Android Anda, dengan aneka fitur menggunakan proyek dan modul yang sama. Menggunakan kembali kode dan sumber daya pada seluruh set sumber

Dengan menerapkan fleksibilitas Gradle, Anda dapat mencapai semua ini tanpa mengubah file sumber inti aplikasi. File versi Android Studio diberi nama build.gradle. File ini adalah teks biasa yang menggunakan Groovy mengonfigurasi versi dengan elemen yang disediakan oleh plugin Android untuk Gradle. Masing-masing proyek memiliki file versi level atas untuk seluruh proyek dan file versi level modul terpisah untuk setiap modul. Saat Anda mengimpor proyek saat ini, Android Studio otomatis menghasilkan file versi yang diperlukan.

Varian Versi

Sistem versi dapat membantu Anda membuat versi berbeda dari aplikasi yang sama dari satu proyek. Ini berguna ketika Anda sama-sama memiliki versi gratis dan versi berbayar dari aplikasi, atau jika Anda ingin mendistribusikan beberapa APK untuk perangkat berbeda di Google Play.

Pemisahan APK

Pemisahan APK memungkinkan Anda untuk membuat beberapa APK berdasarkan kepadatan layar atau ABI. Misalnya, pemisahan APK memungkinkan Anda membuat versi hdpi dan mdpi terpisah dari aplikasi sembari masih mempertimbangkannya sebagai satu varian dan memungkinkannya untuk berbagi setelan aplikasi pengujian, javac, dx, dan ProGuard.

Penyusutan Sumber Daya

Penyusutan sumber daya di Android Studio secara otomatis membuang sumber daya yang tidak terpakai dari aplikasi terkemas dan dependensi perpustakaan. Misalnya, jika aplikasi Anda menggunakan layanan Google Play untuk mengakses fungsi Google Drive, dan saat ini Anda tidak memakai Google Sign-In, maka penyusutan sumber daya dapat membuang berbagai aset yang dapat digambar untuk tombol SignIn Button.

Mengelola Dependensi

Dependensi untuk proyek Anda ditetapkan oleh nama dalam file build.gradle. Gradle menangani penemuan dependensi Anda dan menyediakannya di versi Anda. Anda bisa mendeklarasikan dependensi modul, dependensi biner jarak jauh, dan dependensi biner setempat dalam file build.gradle Anda. Android Studio mengonfigurasi proyek untuk menggunakan Penyimpanan Pusat Maven secara default. (Konfigurasi ini disertakan dalam file versi tingkat atas untuk proyek tersebut). [10]

2.3 Java

Java merupakan Bahasa pemrograman tingkat tinggi yang dipelopori oleh James Gosling yang merupaka enginer di Sun Microsystem. Java mulai dibangun pada tahun 1991. Versi alpha dan beta dari java dirilis pada tahun 1995, 4 tahun setelah projek Java diinisiasi. Pada tahun 2010, Sun Microsystem diakusisi oleg Oracle dan menjadikan Java dikembangkan di bawah kuasa Oracle.

Per Januari 2018, versi stabil terkahir dari Java yaitu versi Java SE 9. Sebelumnya Java SE 9, terdapat beberapa versi dari Java yang telah dirilis. Versi – versi dari java yang pernah dirilis yaitu :

- JDK Aplha dan Beta (1995)
- JDK 1.0 (23 Januari 1996)
- JDK 1.1 (19 Februari 1997)
- J2SE 1.2 (8 Desember 1998)
- J2SE 1.3 (8 Mei 200)
- J2SE 1.4 (6 Februari 2002)
- J2SE 5.0 (30 September 2004
- Java SE 6 (11 Desember 2006)

- Java SE 7 (18 Juli 2011)
- Java SE 8 (18 Maret 2014)
- Java SE 9 (21 September 2017)

Java merupakan salah satu Bahasa yang popular saat ini. Saking populernya, beberapa edisi untuk Java pu diciptakan, contohnya yaitu J2EE untuk Aplikasi Enterprise dan J2ME untuk Aplikasi Mobile. Salah satu alasan lain mengapa Java merupakan bahasa yang popular yaitu dikarenakan Java dapat berjalan di bebagai platfrom system operasi. Java pun dikenal dengan *Write Once, Run Anywhere* karena kompatibilitasnya tersebut. Menuru Sun yang merupakan pelopor dari Java, Java itu:

Simpel

Java merupakan Bahasa yang simple sehingga mudah dipahami. Bermodalkan pengetahuan programming dasar, fundamental dari pemrograman java akan cepat dipahamai

Berorientasi Objek

Java menggunakan konsep pemrograman berorientasi objek sehingga pembuat aplikasi bisa dilakukan lebih modular.

Kuat

Java didesain untuk membuat aplikasi yang memiliki reliabilitas tinggi. Salah satu cara Java untuk membuat program yang memiliki reliabilitas tinggi yaitu dengan mengeliminasi situasi error dengan memeriksanya pada compile time dan runtime.

Aman

Java didesain untuk digunakan pada lingkungan yang terdistribusi. Keamanan merupakan hal yang penting dalam lingkungan terdistribusi. Fitur – fitur keamanan yang dimiliki Java membuat perangkat lunak yang dibuat tidak bisa diserang dari luar atau disisipi virus.

Arsitektur Netral

Aplikasi yang dibuat menggunakan Java merupakan aplikasi yang platfrom independent. Aplikasi hanya perlu satu buah versi yang bisa dijalankan pada platfrom sistem operasi yang berbeda.

Portable

Java dengan karakteristik arsitektur netralnya membuat Java tidak bergantung pada mesin tertentu atau dengan kata lain Java sangatlah portable.

Perfoma Tinggi

Java sangat memperhatikan perfoma. Dengan pengenalan *Just in Time compilation*, proses kompilasi Java menjadi lebih cepat.

Multithreaded

Java memungkinkan pembuatan aplikasi yang bisa melakukan beberapa pekerjaan secara bersamaan.

Dinamis

Java lebih dinamis dibandingan C atau C++ karena didesain untuk dijalankan pada langkungan yang dinamis. [11]

2.3.1 Sejarah Java

Sejarah Java dimulai pada tahun 1991. Sebuah tim kecil yang berisikan engineer dari Sun Microsystem mempelopori proyek Java ini. Tim ini disebut Green Team dan dipimpin oleh James Gosling. Saat awal pengembangag,, bahasa yang dikembangkan oleh Green Team disebut nama “Greentalk”. Setelah itu, mereka mengganti Namanya menjadi “Oak” ketika pohon oak yang tiba-tiba muncul di luar kantornya. Pada tahun 1995, Green Team menggnati Namanya menjadi “Java” karena Oak sudah menjadi trademark dari Oak Technologies. Pada tahun itu, Sun Microsystem lalu merilis Java untuk pertama kali.Pada 13 November 2006, Sun Microsystem merilis Java dengan gratis dan open source dengan lisensi GNU General Public License (GPL). Tetapi pada tahun 2010, Sun Microsystem dibeli oleh Oracle dan menjadikan Java dikembangkan dan dipelihara di bawah kendali Oracle. [11]

2.3.2 Spesifikasi Java

Spesifikasi minimum untuk komputer menjalankan Java yaitu Pentium 2 266 MHz dengan RAM 128 MB. Selain itu, kita juga perlu menyiapkan software berikut: [11]

- Sistem operasi Windows XP/7/8/10 atau Ubuntu Linux minimal versi 12.04 atau Mac OS X minimal versi 10.8.3.

- Java SE 8
- IDE atau text editor seperti Intellij IDEA, Netbeans, Eclipse, Visual Studio Code, Sublime, Atom, dan sebagainya.

2.4 API

Dalam pemrograman komputer, sebuah antarmuka pemrograman aplikasi (API) adalah seperangkat definisi subrutin, protokol, dan alat untuk membangun perangkat lunak aplikasi. Secara umum, ini adalah seperangkat metode komunikasi yang jelas antara berbagai komponen perangkat lunak. API yang bagus memudahkan pengembangan program komputer dengan menyediakan semua blok bangunan, yang kemudian disatukan oleh pemrogram. API mungkin untuk sistem berbasis web, sistem operasi, sistem basis data, perangkat keras komputer atau perpustakaan perangkat lunak. Spesifikasi API dapat mengambil banyak bentuk, namun sering kali mencakup spesifikasi untuk rutinitas, struktur data, kelas objek, variabel atau panggilan jarak jauh. *POSIX, Microsoft Windows API, C ++ Standard Template Library dan Java API* adalah contoh dari berbagai bentuk API. Dokumentasi untuk API biasanya disediakan untuk memudahkan penggunaan.

2.4.1 Clarifai API

API Clarifai menawarkan pengenalan gambar dan video sebagai layanan. Apakah Anda memiliki satu gambar atau miliaran, Anda hanya beberapa langkah dari penggunaan kecerdasan buatan untuk mengenali konten visual Anda.

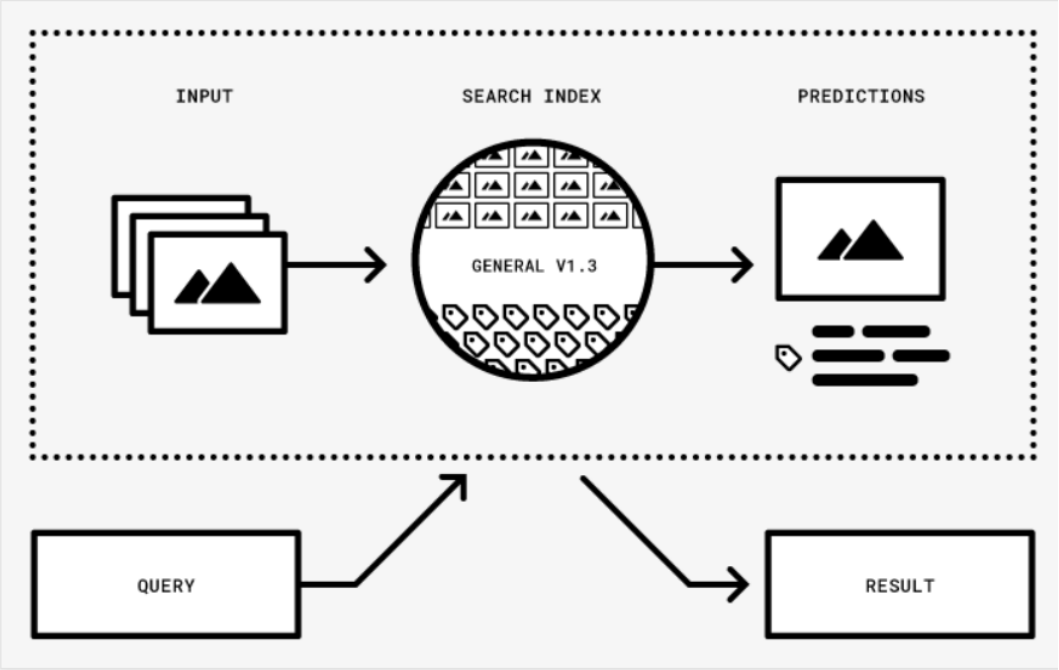
API dibangun di seputar ide sederhana. Anda mengirim masukan (gambar atau video) ke layanan dan mengembalikan prediksi.

Jenis prediksi didasarkan pada model apa yang Anda gunakan untuk memasukkan input. Misalnya, jika Anda menjalankan masukan Anda melalui model 'makanan', prediksi yang akan dihasilkannya akan mengandung konsep yang diketahui oleh model 'makanan'. Jika Anda menjalankan input Anda melalui model 'warna', itu akan mengembalikan prediksi tentang warna dominan pada gambar Anda. [11]

Clarifai Search

API Pencarian memungkinkan Anda untuk mengirim gambar (url atau byte) ke layanan dan membuat mereka diindeks oleh konsep model 'umum' dan

representasi visualnya. Setelah diindeks, Anda dapat mencari gambar berdasarkan konsep atau gambar.



Gambar 2.22 *Clarifai Search API*

Anda juga dapat mencari gambar menggunakan gambar lain. Dalam hal ini, Anda memberikan gambar (url atau byte) dan hasilnya akan mengembalikan semua gambar dalam indeks pencarian Anda yang secara visual mirip dengan yang disediakan. [12]

2.4.2 Bukalapak API

Bukalapak merupakan salah satu *online marketplace* terkemuka di Indonesia yang menyediakan sarana jual–beli dari pelapak ke konsumen. Semua orang dapat membuka toko online di Bukalapak dan melayani pembeli dari seluruh Indonesia untuk transaksi satuan maupun banyak.

Bukalapak memiliki slogan jual-beli *online* mudah dan terpercaya karena Bukalapak memberikan jaminan 100% uang kembali kepada pembeli jika barang tidak dikirimkan oleh pelapak.

Visi Bukalapak: Menjadi online marketplace nomor 1 di Indonesia

Misi Bukalapak: Memberdayakan UKM yang ada di seluruh penjuru Indonesia

Aturan Penggunaan Bukalapak.com

PT Bukalapak.com (selanjutnya disebut "**Bukalapak**") adalah suatu perseroan terbatas yang salah satu jenis usahanya bergerak di bidang jasa portal web. Bukalapak dalam hal ini menyediakan Platform perdagangan elektronik (*e-commerce*) di mana Pengguna dapat melakukan transaksi jual-beli barang dan menggunakan berbagai fitur serta layanan yang tersedia. Setiap pihak yang berada di wilayah Negara Kesatuan Republik Indonesia dapat mengakses Platform Bukalapak untuk kemudian membuka lapak, menjual barang, membeli barang, menggunakan fitur/layanan, atau hanya sekadar mengakses/mengunjungi Platform Bukalapak. Sebagai penunjang bisnis dan penyedia Platform perdagangan elektronik, Bukalapak menjamin keamanan dan kenyamanan bagi para Pengguna.

Aturan Penggunaan ini mengatur penggunaan seluruh layanan yang terdapat pada Platform Bukalapak yang berlaku terhadap seluruh Pengguna dan terhadap setiap Pihak yang menyampaikan permintaan atau informasi kepada Bukalapak. Dengan mendaftar akun Bukalapak dan/atau menggunakan Platform Bukalapak, maka Pengguna dianggap telah membaca, mengerti, memahami dan menyetujui seluruh isi dalam Aturan Penggunaan. [13]

DEFINISI

Dalam Aturan Penggunaan ini, sepanjang tidak ditentukan lain, istilah-istilah di bawah ini mempunyai arti sebagai berikut:

1. **Pengguna** adalah pihak (terdaftar/tidak terdaftar) yang mengakses Platform Bukalapak, termasuk namun tidak terbatas pada Pembeli dan Penjual (Pelapak) serta wajib mematuhi Aturan Penggunaan Bukalapak beserta ketentuan-ketentuan lain termasuk namun tidak terbatas pada Kebijakan Privasi Bukalapak.
2. **Akun** adalah data tentang Pengguna, minimum terdiri dari *username*, *password*, nomor telepon, dan email yang wajib diisi oleh Pengguna Terdaftar.
3. **Platform Bukalapak** adalah situs resmi www.bukalapak.com dan seluruh *microsite* resmi beserta aplikasi resmi Bukalapak (berbasis Android dan iOS) yang dapat diakses melalui perangkat komputer dan/atau perangkat seluler Pengguna.

4. **Barang** adalah suatu objek yang memiliki wujud dan nilai jual beli yang memenuhi kriteria pengiriman pihak penyelenggara jasa ekspedisi pengiriman Barang.
5. **Layanan** adalah secara kolektif: (i) Platform Bukalapak; (ii) Konten, fitur, layanan, dan fungsi apa pun yang tersedia di atau melalui Platform oleh atau atas nama Bukalapak, termasuk Layanan Partner; dan pemberitahuan email, tombol, widget, dan iklan.
6. **Aturan Penggunaan** adalah perjanjian antara Pengguna dan Bukalapak yang berisi seperangkat peraturan yang mengatur hak, kewajiban, tanggung jawab Pengguna dan Bukalapak, serta tata cara penggunaan sistem layanan Bukalapak
7. **Pelapak (Penjual)** adalah Pengguna terdaftar yang melakukan penjualan dan/atau penawaran Barang kepada para Pengguna melalui lapak di Platform Bukalapak serta wajib mematuhi Aturan Penggunaan Bukalapak beserta ketentuan-ketentuan lain termasuk namun tidak terbatas pada Kebijakan Privasi Bukalapak.
8. **Pembeli** adalah Pengguna terdaftar atau tidak terdaftar yang melakukan pembelian Barang yang dijual oleh Penjual (Pelapak) di Platform Bukalapak serta wajib mematuhi Aturan Penggunaan Bukalapak beserta ketentuan-ketentuan lain termasuk namun tidak terbatas pada Kebijakan Privasi Bukalapak.
9. ***Bukalapak Payment System*** adalah sistem pembayaran yang difasilitasi oleh Bukalapak untuk para Pengguna melakukan transaksi jual-beli di Platform Bukalapak.
10. **Otorisasi Akun (TFA)** adalah fitur keamanan untuk akun pada saat login. Pengguna akan menerima kode otentikasi (OTP) yang harus dimasukkan ketika login menggunakan *device* yang belum tervalidasi. Tujuan dari TFA ini adalah untuk menjaga keamanan akun dari pembajakan dan penyalahgunaan akun oleh pihak lain.
11. **Saldo** adalah jumlah dana pada akun Pengguna setelah semua transaksi dan operasi perdagangan dihitung, termasuk untuk operasi nonperdagangan,

yakni Penambahan Dana dan Penarikan dana. Saldo terdiri dari BukaDompet dan Dana.

12. **BukaDompet** adalah suatu bentuk dompet virtual yang dimiliki setiap Pengguna Terdaftar yang berfungsi untuk menyimpan dana hasil penjualan (remit) dan dana hasil pengembalian transaksi (refund).
13. **Lapak** adalah tempat dimana Penjual (Pelapak) dapat menawarkan dan/atau menjual barang dagangannya yang terdapat pada Platform Bukalapak.
14. **Barang Terlarang** adalah adalah barang yang dilarang diperdagangkan oleh Penjual (Pelapak) di Platform Bukalapak berdasarkan peraturan perundang-undangan yang berlaku di Indonesia dan kebijakan internal Bukalapak.
15. **BukaBantuan** merupakan layanan yang disediakan oleh Bukalapak untuk memfasilitasi penyelesaian masalah termasuk namun tidak terbatas pada masalah transaksi Pembeli dan Penjual, Hak Kekayaan Intelektual, pelaporan pelanggaran produk, dan lain-lain.

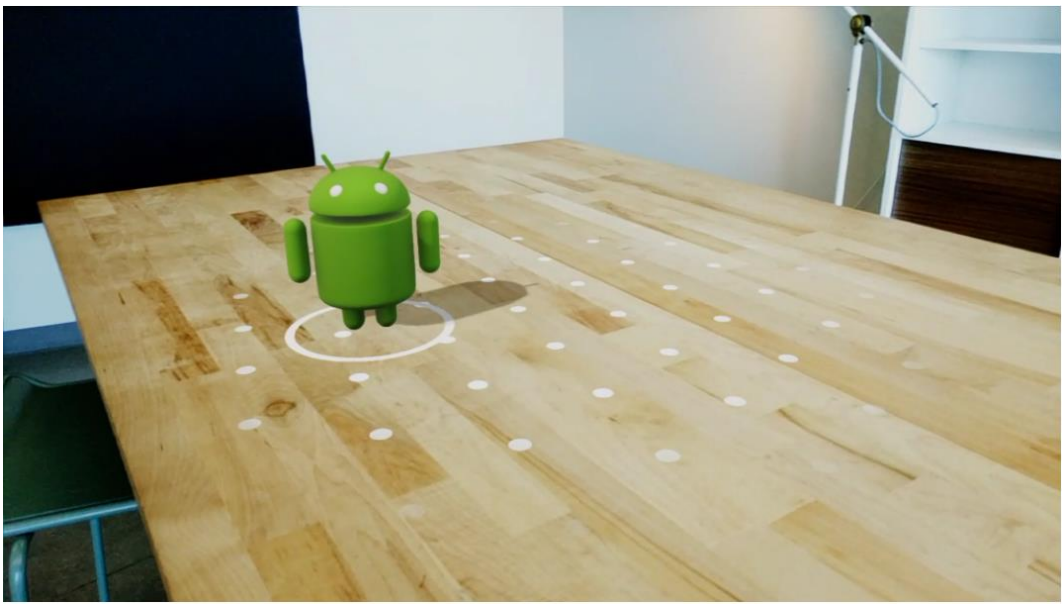
2.4.3 ARCore API

ARCore merupakan Software Development Kit (SDK) untuk Android dan iOS yang pertama rilis Maret 2018 (versi stable September 2018). Sebelum merilis SDK ini, Google sempat merilis Tango pada 2014, yang diharapkan akan se-booming GPS, mampu memetakan dunia berbekal sensor dan kamera motion-tracking. Namun, Google ‘berubah pikiran’ setelah ARKit milik Apple rilis.

Google mengembangkan SDK berbasis AR yang beroperasi melalui 2 cara: *tracking* dan *building environment*. Posisi perangkat *mobile* pengguna selagi bergerak. Lalu, sistem akan menerjemahkan lingkungan hidup pengguna agar tampak serealistis mungkin. Pada dasarnya, teknologi motion tracking menggunakan kamera handphone untuk mengidentifikasi poin-poin menarik alias ‘fitur’ kemudian melacak pergerakannya seiring waktu. Dengan mengkombinasikan pergerakan poin dan membaca sensor inersia, ditentukan posisi dan orientasi handphone selagi pengguna bergerak berpindah tempat.

ARCore mampu mendeteksi permukaan datar, seperti meja dan lantai, sekaligus memperkirakan pencahayaan rata-rata di area sekitar titik utama. Detail

tentang ini akan kami berikan di bawah. Seketika memahami lingkungan pengguna, Anda bebas meletakkan objek visual, membaca anotasi, atau informasi lain seakan-akan terjadi langsung di dunia nyata. ARCore terbentuk menggunakan beberapa set API yang berbeda-beda, framework, dan tools lainnya. Beberapa API itu sudah bisa diterapkan menggunakan Android dan iOS tertentu. [13]



Gambar 2.23 Arcore API

ARCore adalah platform Google untuk membangun pengalaman augmented reality. Menggunakan API yang berbeda, ARCore memungkinkan ponsel Anda merasakan lingkungannya, memahami dunia, dan berinteraksi dengan informasi. Beberapa API tersedia di Android dan iOS untuk memungkinkan pengalaman AR yang dibagikan.

ARCore menggunakan tiga kemampuan utama untuk mengintegrasikan konten virtual dengan dunia nyata seperti yang terlihat melalui kamera ponsel Anda: [14]

- Motion tracking memungkinkan ponsel untuk memahami dan melacak posisinya relatif terhadap dunia.
- Environmental understanding memungkinkan telepon mendeteksi ukuran dan lokasi semua jenis permukaan: permukaan horizontal, vertikal dan miring seperti tanah, meja kopi atau dinding.

- Light estimation memungkinkan ponsel untuk memperkirakan kondisi pencahayaan saat ini di lingkungan.

Bagaimana cara kerja Arcore

Jika Anda baru menggunakan Arcore, akan sangat membantu untuk meninjau world *space dan poses*.

- *World space*
- Koordinasikan ruang di mana kamera dan objek diposisikan
- Posisi kamera dan objek diperbarui di ruang dunia dari bingkai ke bingkai
- *Poses*
- Merupakan posisi dan orientasi objek dalam ruang dunia

Saat Anda membuat Arcore, Anda menggunakan pose yang menggambarkan posisi dan orientasi relatif terhadap perkiraan world space untuk *frame* saat ini.

Anda melampirkan satu atau lebih objek ke Arcore ini. Arcore dan benda-benda yang melekat padanya tampaknya tetap berada di tempat mereka ditempatkan di world. Karena pose Arcore beradaptasi dengan pembaruan *world space* di setiap frame, Arcore memperbarui pose objek sesuai.

Anda dapat melampirkan banyak objek ke Arcore yang sama untuk memastikan bahwa objek-objek ini mempertahankan posisi dan orientasi relatifnya, bahkan ketika pose jangkar menyesuaikan. [14]

2.5 Basis Data

Menurut Fathansyah, basis data adalah suatu kelompok data yang saling berhubungan dan diorganisasikan agar dapat dimanfaatkan kembali dengan cepat dan mudah. Kumpulan data yang saling berhubungan disimpan secara bersama tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan. [15]

2.5.1 Firebase

2.5.1.1 Firebase Realtime Database

Firebase Realtime Database adalah database yang di-host di cloud. Data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung. Ketika Anda membuat aplikasi lintas-platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah instance Realtime Database dan menerima update data terbaru secara otomatis. [15]

Kemampuan utama:

- **Realtime**

  Sebagai ganti permintaan HTTP biasa, Firebase Realtime Database menggunakan sinkronisasi data—setiap kali data berubah, semua perangkat yang terhubung akan menerima update dalam waktu milidetik. Memberikan pengalaman yang kolaboratif dan imersif tanpa perlu memikirkan kode jaringan.

- **Offline**

  Aplikasi Firebase tetap responsif bahkan saat offline karena SDK Firebase Realtime Database menyimpan data ke disk. Setelah konektivitas pulih, perangkat klien akan menerima setiap perubahan yang terlewat dan melakukan sinkronisasi dengan status server saat ini.

- **Dapat Diakses dari Perangkat Klien**

  Firebase Realtime Database dapat diakses secara langsung dari perangkat seluler atau browser web; server aplikasi tidak diperlukan. Keamanan dan validasi data dapat diakses melalui Aturan Keamanan Firebase Realtime Database yang merupakan kumpulan aturan berbasis ekspresi dan dijalankan ketika data dibaca atau ditulis.

- **Menskalakan di beberapa database**

  Dengan Firebase Realtime Database pada paket harga Blaze, Anda dapat mendukung kebutuhan data aplikasi Anda pada skala tertentu dengan membagi data Anda di beberapa instance database di project Firebase yang sama. Menyederhanakan autentikasi dengan Firebase Authentication pada project Anda dan mengautentikasi pengguna di instance database Anda. Mengontrol akses ke data di tiap database dengan Aturan Firebase Realtime Database khusus untuk tiap instance database.

2.5.1.2 Firebase Authentication

Sebagian besar aplikasi perlu mengetahui identitas pengguna. Dengan mengetahui identitas pengguna, aplikasi dapat menyimpan data pengguna secara aman di cloud dan memberikan pengalaman personal yang sama di setiap perangkat pengguna.

Firebase Authentication menyediakan layanan backend, SDK yang mudah digunakan, dan library UI yang siap pakai untuk mengautentikasi pengguna ke aplikasi Anda. Firebase Authentication mendukung autentikasi menggunakan sandi, nomor telepon, penyedia identitas gabungan yang populer, seperti Google, Facebook, dan Twitter, dan lain-lain.

Firebase Authentication terintegrasi erat dengan layanan Firebase lainnya dan sistem ini memanfaatkan berbagai standar industri, seperti OAuth 2.0 dan OpenID Connect, sehingga dapat dengan mudah diintegrasikan dengan backend kustom Anda.

Kemampuan utama membuat pengguna login ke aplikasi Firebase dengan menggunakan FirebaseUI sebagai solusi autentikasi drop-in lengkap atau dengan menggunakan Firebase Authentication SDK untuk mengintegrasikan salah satu atau beberapa metode login ke aplikasi secara manual. [15]

- **Solusi autentikasi drop-in**

  FirebaseUI menyediakan solusi autentikasi drop-in yang menangani alur UI untuk membuat pengguna login dengan alamat email dan sandi, nomor telepon, dan penyedia identitas gabungan yang populer, seperti Login dengan Google dan Login dengan Facebook. Komponen FirebaseUI Auth mengimplementasikan praktik terbaik untuk proses autentikasi di perangkat seluler dan situs, yang akan memaksimalkan konversi login dan pendaftaran untuk aplikasi Anda. FirebaseUI Auth juga menangani kasus khusus, seperti pemulihan akun dan penyambungan akun, yang dapat dianggap sensitif bagi keamanan dan rawan kesalahan untuk ditangani dengan benar. FirebaseUI dapat disesuaikan dengan mudah agar cocok dengan gaya visual aplikasi Anda. FirebaseUI juga open source, sehingga Anda tidak dibatasi dalam mewujudkan pengalaman pengguna yang Anda inginkan. [15]

2.5.1.3 Firebase Storage

Cloud Storage for Firebase adalah layanan penyimpanan objek yang andal, sederhana, dan hemat biaya yang dibuat untuk skala Google. Firebase SDK untuk Cloud Storage menambahkan keamanan Google pada upload dan download file untuk aplikasi Firebase Anda, bagaimanapun kualitas jaringannya. Anda dapat

menggunakan SDK kami untuk menyimpan gambar, audio, video, atau konten buatan pengguna lainnya. Di server, Anda dapat menggunakan Google Cloud Storage untuk mengakses file yang sama. [15]

Kemampuan utama :

- **Operasi yang stabil**

  Firebase SDK untuk Cloud Storage melakukan upload dan download, bagaimanapun kualitas jaringannya. Upload dan download bersifat stabil, artinya proses ini akan dilanjutkan dari posisi berhenti, sehingga menghemat waktu dan bandwidth pengguna.

- **Keamanan yang kuat**

  Firebase SDK untuk Cloud Storage terintegrasi dengan Firebase Authentication untuk menyediakan autentikasi yang mudah dan intuitif bagi developer. Anda dapat menggunakan model keamanan deklaratif kami untuk mengizinkan akses berdasarkan nama file, ukuran, jenis konten, dan metadata lainnya.

- **Skalabilitas tinggi**

  Cloud Storage for Firebase dibuat untuk skala exabyte ketika aplikasi Anda menjadi viral. Berkembang dari prototipe menjadi produksi dengan mudah menggunakan infrastruktur yang juga memberdayakan Spotify dan Google Foto.

2.6 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. [16]

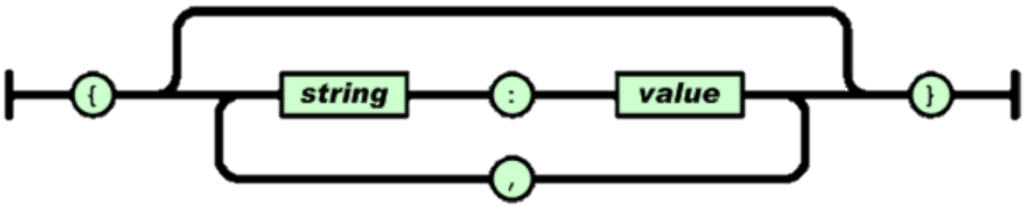
JSON terbuat dari dua struktur:

- Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
- Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

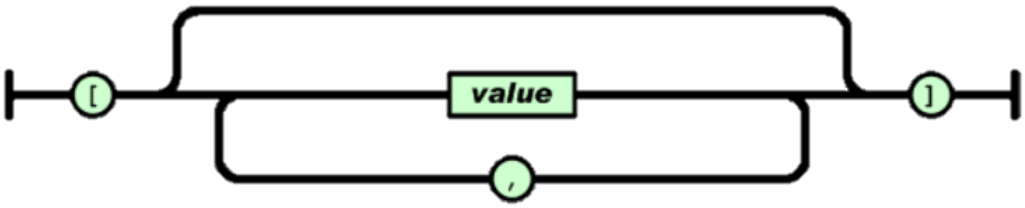
JSON menggunakan bentuk sebagai berikut:

1. **Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan {(kurung kurawal buka) dan diakhiri dengan} (kurung kurawal tutup). Setiap nama diikuti dengan: (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh, (koma).



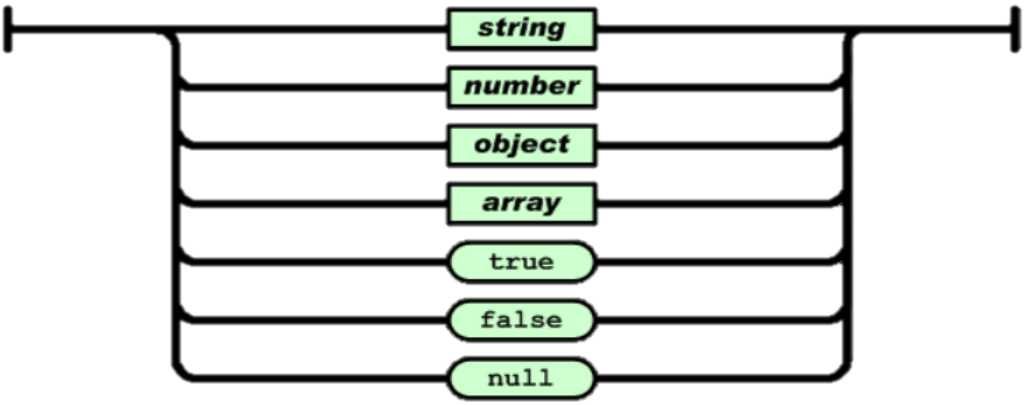
Gambar 2.24 *Objek JSON*

2. **Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh, (koma).



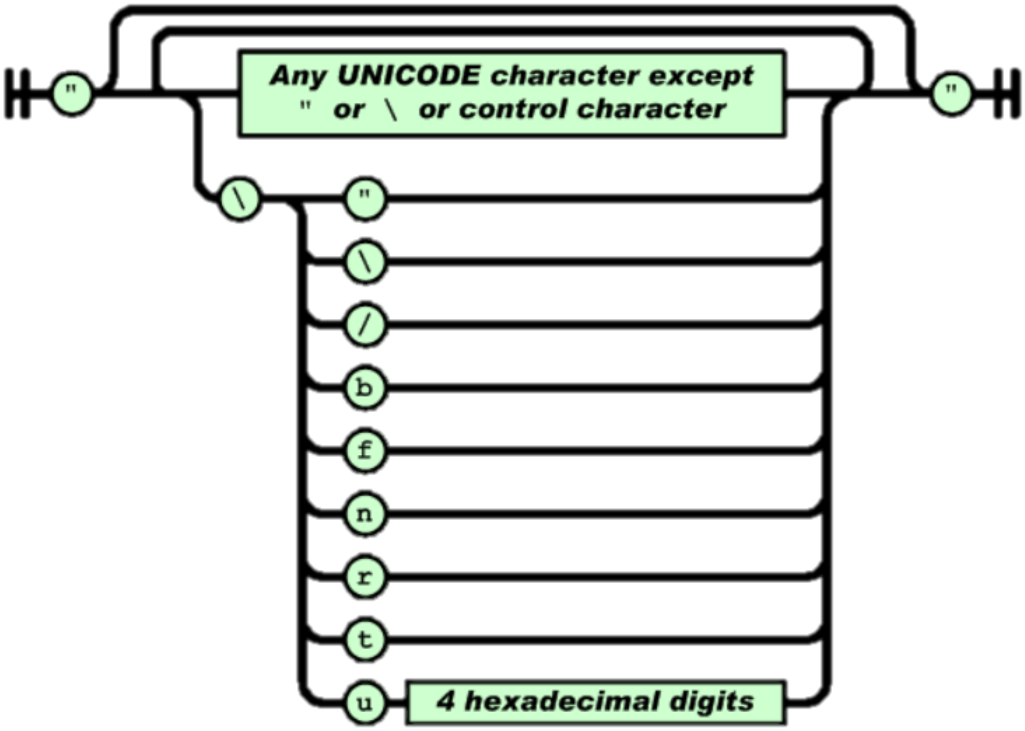
Gambar 2.25 ***Array JSON***

3. **Nilai** (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau trueatau false atau null, atau sebuah objek atau sebuah larik. Strukturstruktur tersebut dapat disusun bertingkat.



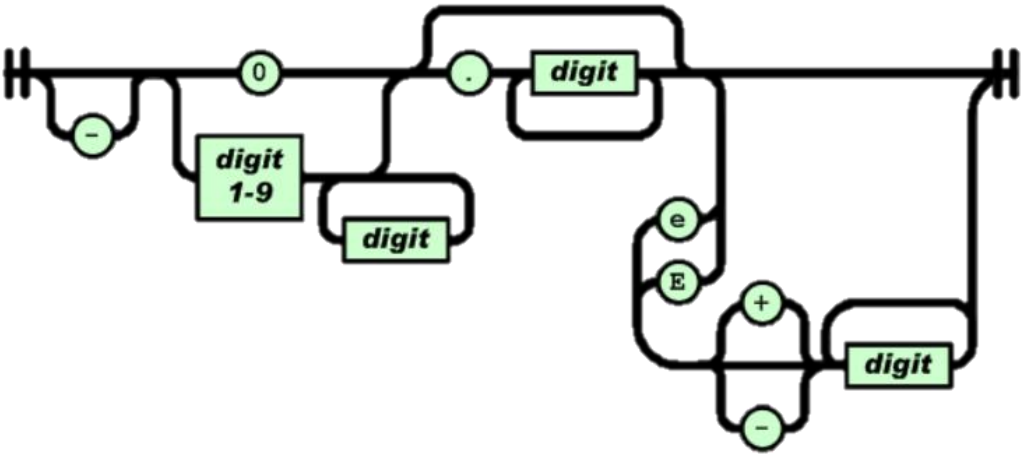
Gambar 2.26 ***Value JSON***

4. **String** adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2.27 ***String JSON***

5. **Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2.28 ***Number JSON***

2.7 Objek Oriented Analysis Design

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement (syarat/keperluan) yang

harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. [17]

2.8 Unified Modelling Language

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

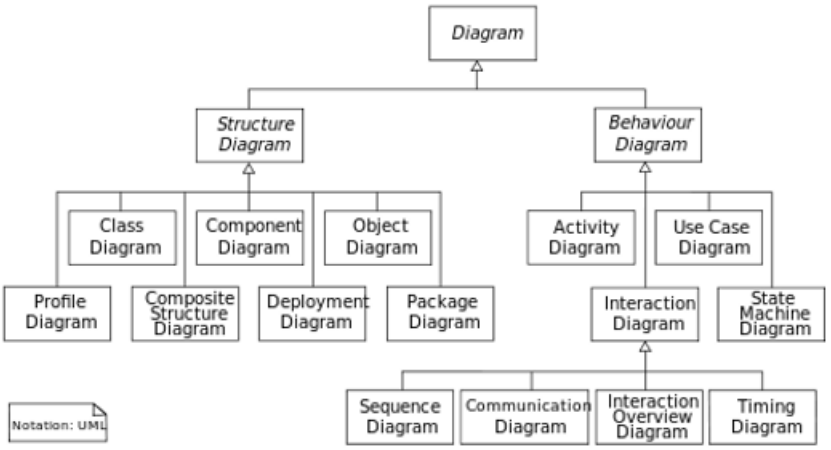
Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri,

yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.

2.8.1 Diagram UML

Berikut penjelasan singkat dari pembagian kategori tersebut adalah:



Gambar 2.29 Diagram UML

- *Structure* Diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari suatu sistem yang dimodelkan
- Behavior Diagram yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
- Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interkasi antar subsistem pada suatu sistem.

2.8.2 Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas - kelas di dalam program perangkat lunak sesuai dengan

perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas sebagai berikut:

- Kelas main yaitu kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
- Kelas yang menangani tampilan sistem (*view*) yaitu kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
- Controller yaitu kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasa disebut kelas proses.
- Model yaitu kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

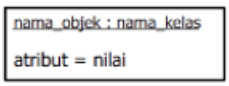
Berikut adalah simbol-simbol yang ada pada diagram kelas :

Simbol	Deksripsi
Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur sistem.
Antarmuka / *Interface* nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
Asosiasi / *Association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *Directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum – khusus)
Kebergantungan / *Dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.
Agregasi / *Aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

Gambar 2.30 Simbol *Class Diagram*

2.8.3 Objek Diagram

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Berikut adalah simbol-simbol yang ada pada diagram objek: [17]

Simbol	Deskripsi
Objek 	objek dari kelas yang berjalan saat sistem dijalankan
Link	relasi antar objek

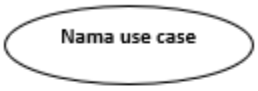
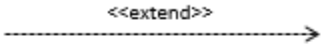
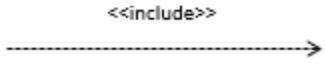
Gambar 2.31 Simbol Diagram Objek

2.8.4 Use Case Diagram

Diagram use case merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fingsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case. [17]

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu orang.
- Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Simbol	Deksripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor, biasanya dinyatakan dengan menggunakan kata kerja di awal di frase nama usecase
Aktor / Actor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase aktor.
Asosiasi / Association 	Komunikasi antar aktor dan use case yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / Extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi / Generalization 	Hubungan generalisasi dan spesialisasi (umum- khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan / Include / Uses 	Relasi *use case* tamabahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Gambar 2.32 Simbol *Use Case Diagram*

2.8.5 Activity Diagram

Diagram aktivitas asatu activity diagram menggambarkan workflow (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktifitas

menggambarkan aktifitas bukan apa yang dilakukan aktor. [17] Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hak berikut:

- Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
- Urutan atau pengelompokan tampilan dari sistem / user interface dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan
- Rancangan pengujian dimana setiap aktifitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
- Rancangan menu yang ditampilkan pada diagram aktifitas

Berikut adalah simbol-simbol yang ada pada diagram aktifitas:

Gambar 2.33 Simbol *Acttivity Diagram*

2.8.6 Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirim dan diterima antar

objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objekobjek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. [17] Berikut adalah simbol-simbol yang ada pada sequence diagram:

Gambar 2.34 Simbol *Sequence Diagram*