BAB 2
LANDASAN TEORI

2.1 Makanan

Makanan adalah segala bahan yang kita makan atau masuk ke dalam tubuh yang membentuk atau menggantikan jaringan tubuh, memberikan tenaga, atau mengatur semua proses dalam tubuh.[1]

Pada umumnya bahan makanan mengandung beberapa unsur atau senyawa seperti air, karbohidrat, protein, lemak, vitamin, enzim, pigmen dan lain lain. Setiap makanan memiliki kandungan gizi yang berbeda.

Setiap jenis gizi mempunyai fungsi yang berbeda. Karbohidrat merupakan sumber tenaga sehari hari. Protein digunakan oleh tubuh untuk membantu pertumbuhan, baik otak maupun tubuh. Lemak digunakan oleh tubuh sebagai cadangan makanan dan sebagai cadangan energi. Lemak akan digunakan saat tubuh kekurangan karbohidrat dan lemak akan memecah mejadi glukosa yang sangat berguna bagi tubuh saat membutuhkan energi.

2.2 Resep Makanan

Resep makanan adalah suatu susunan instruksi atau algoritma yang menunjukan cara membuat suatu masakan.

Dalam resep makanan terdapat beberapa komponen yang sebagian harus ditulis dan ada juga yang tidak harus ditulis. Komponen resep yang biasanya ada antara laim:

- Nama masakan
- Peruntukan jenis masakan, seperti : menu diet atau menu untuk penderita diabetes
- Komposisi atau bahan dengan kuantitasnya
- Alat alat masak yang digunakan
- Cara memasak
- Lama waktu memasak
- Jumlah sajian

- Perkiraan jumlah kalori
- Kadar gizi

2.3 Tingkat Metabolisme Basal

Tingkat metabolisme basal (basal metabolic rate / BMR) adalah tingkat minimal pengeluaran energi per satuan waktu oleh hewan endotermis saat istirahat. Hal ini dilaporkan dalam satuan energi per satuan waktu mulai dari watt (joule / second) hingga ml O2 / min atau joule per jam per kg massa tubuh J / (h · kg). Pengukuran yang tepat membutuhkan kriteria ketat yang harus dipenuhi. Kriteria ini termasuk berada dalam keadaan fisik dan psikologis yang tidak terganggu, di lingkungan yang netral secara termal, sementara di negara pasca penyerapan (yaitu, tidak secara aktif mencerna makanan). Pada hewan bradimetabolik, seperti ikan dan reptil, istilah setara standar tingkat metabolisme (SMR) digunakan. Ini mengikuti kriteria yang sama dengan BMR, namun memerlukan dokumentasi suhu di mana tingkat metabolisme diukur. Hal ini membuat BMR merupakan varian dari pengukuran tingkat metabolisme standar yang mengecualikan data suhu, sebuah praktik yang menyebabkan masalah dalam menentukan tingkat metabolisme "standar" bagi banyak mamalia. [2]

Metabolisme terdiri dari proses yang dibutuhkan tubuh. Tingkat metabolisme basal adalah jumlah energi yang terekspresikan dalam kalori yang dibutuhkan seseorang untuk menjaga fungsi tubuh tetap tenang. Beberapa dari proses tersebut adalah pernapasan, peredaran darah, pengendalian suhu tubuh, pertumbuhan sel, fungsi otak dan saraf, dan kontraksi otot. Tingkat metabolisme basal (basal metabolic rate / BMR) mempengaruhi tingkat bahwa seseorang membakar kalori dan akhirnya apakah individu mempertahankan, memperoleh, atau kehilangan berat badan. Tingkat metabolisme basal menyumbang sekitar 60 sampai 75% dari pengeluaran kalori harian oleh individu. Hal ini dipengaruhi oleh beberapa faktor. BMR biasanya menurun 1-2% per dekade setelah usia 20 tahun, terutama karena hilangnya massa bebas lemak, [2] meskipun variabilitas antara individu tinggi.

Ada tiga mekanisme metabolisme yang terjadi di tubuh kita: *basal metabolic rate* (BMR), metabolisme energi untuk aktivitas, dan metabolisme energi untuk mencerna makanan. BMR memiliki proporsi terbesar di tubuh kita, yaitu sebesar 50-80%, karena mekanisme ini berperan untuk menjaga berbagai fungsi organ dan keseimbangan antara lemak dan otot. Pada umumnya, metabolisme yang lambat terjadi karena perlambatan pembakaran energi untuk aktivitas, dan mekanisme BMR.

Fungsi dari proses metabolisme adalah untuk menyediakan energi, dan mengganti sel tubuh yang rusak dengan cara memecah nutrisi dan cadangan makanan atau lemak. Saat metabolisme menjadi lambat, tubuh akan lebih sulit memproses makanan untuk menghasilkan energi. Akibatnya, terjadi penurunan massa otot dan tubuh menyimpan lapisan lemak lebih banyak.

	Accurate Estimates	Under-estimation	Over-estimation
Harris-Benedict			
Nonobese (using BMI standard)	45%–81%	23%	42%
Obese	36%–64%	35%	43%
Mifflin-St. Jeor			
Nonobese (using BMI standard)	82%	18%	15%
Obese	70%	20%	15%

Source: Frankenfeld, Roth-Yousey & Compher 2005.

Gambar 2.1 Perbandingan Formula BMR

2.3.1 Kalkulator Kalori

Kalkulator kalori ini didasarkan pada persamaan Mifflin-St Jeor yang menghitung tingkat metabolisme basal (basal metabolic rate / BMR), dan hasilnya didasarkan pada perkiraan rata-rata. Tingkat metabolisme basal adalah jumlah energi yang dikeluarkan per hari saat istirahat. Persamaan Mifflin-St Jeor dianggap sebagai persamaan paling akurat untuk menghitung BMR. Persamaan Mifflin-St Jeor adalah sebagai berikut:

Untuk pria: BMR = 10 × berat (kg) + 6,25 × tinggi (cm) - 5 × usia (y) + 5
Untuk wanita: BMR = 10 × berat (kg) + 6,25 × tinggi (cm) - 5 × usia (y) - 161

Nilai yang diperoleh dari persamaan ini adalah perkiraan jumlah kalori yang dapat dikonsumsi seseorang dalam sehari untuk mempertahankan berat badan mereka dengan asumsi mereka tetap beristirahat. Nilai ini dikalikan dengan faktor aktivitas (umumnya 1,2-1,95) bergantung pada tingkat latihan khas seseorang untuk memperoleh nilai yang lebih realistis untuk mempertahankan berat badan (karena orang cenderung tidak beristirahat selama perjalanan tubuh. sepanjang hari). 1 pon berat badan atau sekitar 0,45 kg setara dengan sekitar 3.500 kalori. Dengan demikian, untuk menurunkan 1 pound per minggu, disarankan agar 500 kalori dicukur dari perkiraan kalori yang diperlukan untuk pemeliharaan berat badan per hari. Misalnya, jika seseorang memiliki perkiraan penjatahan 2.500 kalori per hari untuk mempertahankan berat badan, mengkonsumsi 2.000 kalori per hari selama satu minggu secara teoritis akan menghasilkan 3.500 kalori (atau 1 pon) yang hilang selama periode tersebut.[3]

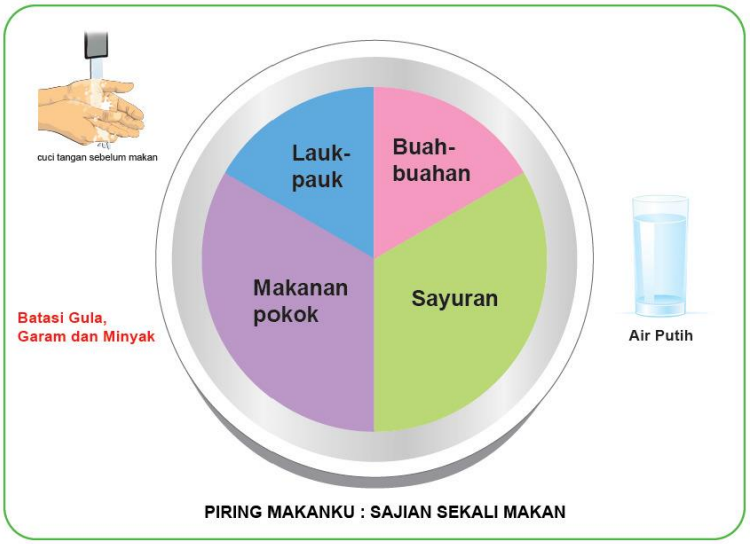
Tubuh tidak memerlukan banyak kalori untuk bisa bertahan. Namun, mengkonsumsi terlalu sedikit kalori menyebabkan tubuh berfungsi dengan buruk, karena hanya akan menggunakan kalori untuk fungsi yang penting untuk bertahan hidup, dan mengabaikan hal-hal yang diperlukan untuk kesehatan dan kesejahteraan umum. Dengan demikian, sangat disarankan agar seseorang yang mencoba menurunkan berat badan memonitor kebutuhan kalori tubuh mereka dan menyesuaikannya seperlunya untuk mempertahankan kebutuhan nutrisinya.

2.3.2 Pedoman Gizi Seimbang

Kualitas atau mutu gizi dan kelengkapan zat gizi dipengaruhi oleh keragaman jenis pangan yang dikonsumsi. Semakin beragam jenis pangan yang dikonsumsi semakin mudah untuk memenuhi kebutuhan gizi. Bahkan semakin beragam pangan yang dikonsumsi semakin mudah tubuh memperoleh berbagai zat lainnya yang bermanfaat bagi kesehatan. Oleh karena itu konsumsi anekaragam pangan merupakan salah satu anjuran dalam mewujudkan gizi seimbang.

Selain memperhatikan keanekaragaman makanan dan minuman juga perlu memperhatikan dari segi keamanannya yang berarti makanan dan minuman itu harus bebas dari kuman penyakit atau bahan berbahaya.

Cara menerapkan pesan ini adalah dengan mengkonsumsi lima kelompok pangan setiap hari atau setiap kali makan. Lima kelompok pangan tersebut adalah makanan pokok, lauk-pauk, sayuran, buah-buahan dan minuman. Mengkonsumsi lebih dari satu jenis untuk setiap kelompok makanan (makanan pokok, lauk-pauk, sayuran dan buah-buahan) setiap kali makan akan lebih baik.[10]



Gambar 2.2 Sajian Sekali Makan

2.4 Android

Android adalah sistem operasi berbasis *Linux* yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finasial dari *Google* yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didrikannya *Open Handset Alliance,* konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak dan telekomunikasi yang bertujuan untuk memajukan standard terbuka perangkat seluler. Ponsel pertama android mulai dijual pada bulan oktober 2008

Android kini menguasai ratusan juta perangkat mobile di lebih dari 190 negara di seluruh dunia. Ini adalah *Installed base* terbesar dari setiap platform seluler dan berkembang dengan cepat, setiap hari satu juta pengguna lainnya menyalakan perangkat android mereka untuk pertama kalinya dan memulai mencari aplikasi, game dan konten digital lainnya.[4]

2.4.1 Google Play

Google Play adalah pasar utama untuk menjual dan mendistribusikan aplikasi Android. Saat Anda mempublikasikan sebuah aplikasi di Google Play, Anda mencapai basis Android yang besar.

Sebagai *Open Source*, Google Play memberi Anda kendali atas bagaimana Anda menjual produk Anda. Anda dapat mempublikasikan kapan pun Anda mau, sesering yang Anda mau, dan kepada pelanggan yang Anda inginkan. Anda dapat mendistribusikan secara luas ke semua pasar dan perangkat atau fokus pada segmen, perangkat, atau rentang kemampuan perangkat keras tertentu.

Anda dapat memonetisasi cara yang paling sesuai untuk harga bisnis atau gratis, dengan produk atau langganan dalam aplikasi - untuk keterlibatan dan pendapatan tertinggi. Anda juga memiliki kontrol penuh atas harga untuk aplikasi dan produk

dalam aplikasi Anda dan dapat menetapkan atau mengubah harga dalam mata uang yang didukung setiap saat.

Selain mengembangkan basis pelanggan, Google Play membantu Anda membangun visibilitas dan keterlibatan di seluruh aplikasi dan merek Anda. Seiring bertambahnya popularitas aplikasi Anda, 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 Anda.[4]

2.4.2 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 seluler pintar 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 telepon pintar untuk menyaingin *Symbian* dan *Windows Mobile* (Iphone Apple belum dirilis pada saat itu). Meskipun para pengambang sedang menciptakan sebuah perangkat lunak yang diperuntukan bagi telepon seluler. Masih pada tahun yang sama, Rubin kehabisan uang. Steve Perlman, seorang teman dekat Rubin, meminjamkan $10.000 tunai dan menolak tawaran saham di perusahaan.

Google mengakuisisi Android Inc pada tanggal 17 agustus 2005, menjadikannya sebagai anak perusahaan yang sepenuhnya 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 pernagkat 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.[4]

2.4.3 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 2.3 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.

2. 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.

3. 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 kerusakn 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

4. 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.

5. 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

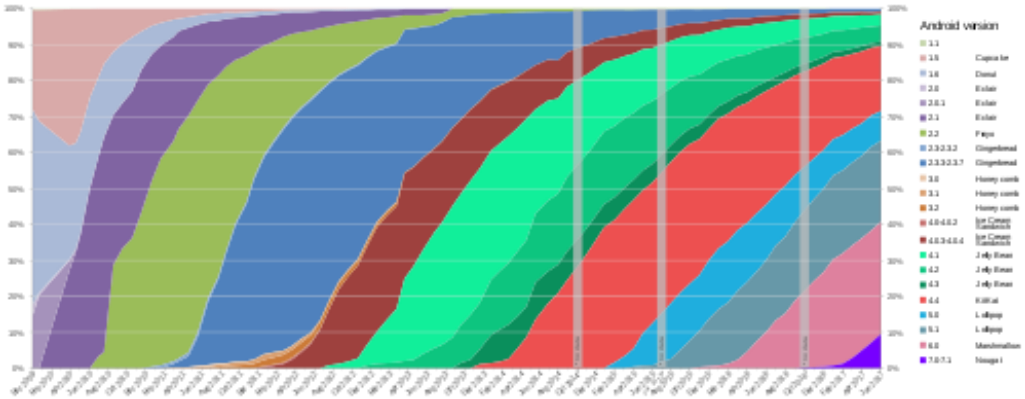
Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.
Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android

6. 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. [5]

2.4.4 Versi Android



Gambar 2.4 Distribusi Versi Android

Versi Android diawali dengan dirilisnya Android beta pada bulan November 2007. Versi komersial pertama, Android 1.0, dirilis pada September 2008. Android dikembangkan secara berkelanjutan oleh Google dan *Open Handset Alliance* (OHA), yang telah merilis sejumlah pembaruan sistem operasi ini sejak dirilisnya versi awal.

Sejak April 2009, versi Android dikembangkan dengan nama kode yang dinamai berdasarkan makanan pencuci mulut dan penganan manis. Masing-masing versi dirilis sesuai urutan alfabet, yakni *Cupcake* (1.5), *Donut* (1.6), *Eclair* (2.0–2.1), *Froyo* (2.2–2.2.3), *Gingerbread* (2.3–2.3.7), *Honeycomb* (3.0–3.2.6), *Ice Cream Sandwich* (4.0–4.0.4), *Jelly Bean* (4.1–4.3), *KitKat* (4.4+), *Lollipop* (5.0+), *Marshmallow* (6.0+), hingga yang terbaru adalah *Nougat* (7.0+) dan selanjutnya versi android terbaru yang ditunggu-tunggu adalah Android O (8.0+).

Pada tanggal 3 September 2013, Google mengumumkan bahwa sekitar 1 miliar perangkat seluler aktif di seluruh dunia menggunakan OS Android.
Berikut adalah versi android berdasarkan tanggal rilis:

1. **Android 1.0 (API level 1)**

Android 1.0, versi komersial pertama Android, dirilis pada 23 September 2008. Perangkat Android pertama yang tersedia secara komersial adalah *HTC Dream*.

2. **Android 1.2 (API level 2)**

Pada 9 Februari 2009, pemutakhiran Android 1.1 dirilis, awalnya hanya untuk HTC Dream. Android 1.1 juga dikenal dengan "Petit Four", meskipun nama ini tidak digunakan secara resmi. Versi ini memperbaiki beberapa bug, mengubah API Android, dan menambahkan beberapa fitur.

3. **Android 1.5 Cupcake (API level 3)**

Pada 27 April 2009, Android 1.5 dirilis, menggunakan kernel Linux 2.6.27. Versi ini adalah rilis pertama yang secara resmi menggunakan nama kode berdasarkan nama-

nama makanan pencuci mulut ("*Cupcake*"), nama yang kemudian digunakan untuk semua versi rilis selanjutnya. Pembaruan pada versi ini termasuk beberapa fitur baru dan perubahan UI.

4. Android 1.6 Donut (API level 4)

Pada 15 September 2009, SDK Android 1.6 – dinamai Donut – dirilis, berdasarkan kernel Linux 2.6.29. versi ini hadir untuk menutupi bug pada versi sebelumnya, sekaligus unutk penambahan beberapa fitur seperti misalnya dukungan untuk perangkat dengan ukuran layar yang lebih besar.

5. Android 2.0-2.1 Eclair (API level 5-7)

Pada 26 Oktober 2009, SDK Android 2.0 – dinamai *Eclair* – dirilis, berbasis kernel Linux 2.6.29. Versi 2.0.1 rilis tanggal 3 desember 2009 dan versi 2.1 rilis tanggal 12 januari 2010. Beberapa penambahan fitur pada versi ini seperti *Bluethooth*, *flash* pada kamera, *digital zoom* pada kamera, *multi touch, live wallpaper* dan lainnya.

6. Android 2.2-2.2.3 Froyo(API level 8)

Pada 20 Mei 2010, SDK Android 2.2 (Froyo, singkatan untuk *frozen yogurt*) dirilis, yang berbasis kernel Linux 2.6.32.

7. Android 2.3-2.3.7 Gingerbread (API level 9-10)

Pada tanggal 6 Desember 2010, SDK Android 2.3 (Gingerbread) dirilis, berbasis kernel Linux 2.6.35. Versi 2.3.1 yang rilis desember 2010 dan versi 2.3.2 yang rilis januari 2011 berisi perbaikan bug pada Google Nexus S. Versi 2.3.3 sampai versi 2.3.7 rilis antara bulan februari 2011 sampai bulan september 2011.

Dirilis pada tanggal 6 Desember 2010 bersamaan dengan dihadirkannya Nexus S yang merupakan perangkat smartphone seri Nexus yang diproduksi oleh Samsung. Versi OS ini juga mengawali kesuksesan Android di jagad smartphone meski masih kalah populer dengan BlackBerry OS. Beberapa vendor mulai serius untuk menggarap

perangkat dengan OS Android.Pada saat itu, Samsung dengan Galaxy series nya berperan besar dalam kesuksesan Android. Promosi yang luar biasa gencarnya membuat orang awam mulai mengenal sistem operasi Android. Bahkan saat itu sebagian besar orang beranggapan bahwa OS Android adalah milik Samsung karena kuatnya branding yang dilakukan oleh Samsung. Ini juga menjadi awal mula kedigdayaan Samsung di jagad smartphone.

8. Android 3.0-3.2 Honeycomb (API level 11-13)

Pada 22 Februari 2011, SDK Android 3.0 (Honeycomb) – pembaruan pertama Android yang ditujukan hanya untuk komputer tablet – dirilis, berdasarkan kernel Linux 2.6.36. Perangkat pertama yang menggunakan versi ini adalah tablet Motorola Xoom, yang dirilis pada 24 Februari 2011. Versi 3.1 rilis tanggal 10 mei 2011. Versi 3.2 rilis tanggal 15 juli 2011. Google TV generasi pertama dan kedua menggunakan Honeycomb.

9. Android 4.0-4.0.4 Ice Cream Sandwitch (API level 14-15)

SDK Android 4.0.1 (Ice Cream Sandwich), berdasarkan kernel Linux 3.0.1, dirilis pada 19 Oktober 2011. Petinggi Google, Gabe Cohen, menyatakan bahwa Android 4.0 "secara teoretis kompatibel" dengan perangkat Android 2.3x yang diproduksi pada saat itu. Kode sumber untuk Android 4.0 tersedia pada tanggal 14 November 2011. Pembaruan pada versi ini antara lain:

Ice Cream Sandwitch adalah versi terakhir yang mendukung *Flash player Adobe System.* Versi 4.0.3 rilis tanggal 16 desember 2011. Versi 4.0.4 rilis tanggal 29 maret 2012.

10. Android 4.1-4.3 Jelly Bean (API level 16-18)

Google mengumumkan Android 4.1 (Jelly Bean) dalam konferensi Google I/O pada tanggal 27 Juni 2012. Berdasarkan kernel Linux 3.0.31, Jelly Bean adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuka

pengguna (UI). Pembaruan ini diwujudkan dalam "Proyek Butter", perbaikan ini termasuk antisipasi sentuh, triple buffering, perpanjangan waktu vsync, dan peningkatan frame rate hingga 60 fps untuk menciptakan UI yang lebih halus. Android 4.1 Jelly Bean dirilis untuk Android Open Source Project pada tanggal 9 Juli 2012. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

Google berencana merilis Jelly Bean 4.2 pada sebuah acara di New York City pada 29 Oktober 2012, tapi dibatalkan karena Badai Sandy. Jelly Bean 4.2 didasarkan pada kernel Linux 3.4.0, dan pertama kali digunakan pada Nexus 4 LG dan Nexus 10 Samsung, yang dirilis pada 13 November 2012.

Google merilis Jelly Bean 4.3 pada 24 Juli 2013 di San Francisco. Kebanyakan perangkat Nexus menerima pembaruan dengan segera. Nexus 7 generasi kedua adalah perangkat pertama yang menggunakan sistem operasi ini. Sebuah pembaruan minor dirilis pada tanggal 22 Agustus 2013.

11. Android 4.4 Kitkat (API level 19)

Google mengumumkan Android 4.4 KitKat (dinamai dengan izin dari Nestlé dan Hershey) pada 3 September 2013, dengan tanggal rilis 31 Oktober 2013. Sebelumnya, rilis berikutnya setelah Jelly Bean diperkirakan akan diberi nomor 5.0 dan dinamai 'Key Lime Pie'.

12. Android 5.0 Lillipop (API level 21)

Android 5.0 pertama kali diperkenalkan di bawah codename "Android L" pada 25 Juni 2014 selama presentasi keynote pada konferensi pengembang Google I / O. Di samping Lollipop, presentasi difokuskan pada sejumlah platform Android yang berorientasi dan teknologi baru, termasuk Android TV, pada platform Android Auto, dpt dipakai pada platform komputasi Android Wear, dan platform pelacakan kesehatan Google Fit.

Bagian dari presentasi didedikasikan untuk bahasa desain cross-platform baru yang disebut sebagai "material design". Memperluas pada "kartu" motif pertama kali terlihat di Google Now, adalah desain dengan peningkatan penggunaan layout berbasis grid, animasi dan transisi responsif, padding, dan efek kedalaman seperti pencahayaan dan bayangan.

- Desain antarmuka (tampilan) yang dinamakan "*Material Design*".
- 64-bit ART compiler
- Project volta, yang berguna untuk meningkatkan daya hidup baterai 30% lebih tahan lama.
- '*factory reset protection'*. Fitur ini berguna ketika smartphone hilang, ia tidak bisa direset ulang tanpa memasukkan id google dan kata sandi (password).

13. Android 6.0 Marshmallow (API level 23)

Android Marshmallow memperkenalkan model izin yang didesain ulang: sekarang ada hanya delapan kategori izin, dan aplikasi yang tidak lagi secara otomatis diberikan semua hak akses mereka ditentukan pada waktu instalasi. Sebuah sistem opt-in sekarang digunakan, di mana pengguna akan diminta untuk memberikan atau menolak izin individu (seperti kemampuan untuk mengakses kamera atau mikrofon) untuk aplikasi ketika mereka dibutuhkan. Aplikasi mengingat hibah izin mereka, dan mereka dapat disesuaikan oleh pengguna setiap saat. Model izin baru akan digunakan hanya oleh aplikasi yang dikompilasi untuk Marshmallow menggunakan kit pengembangan perangkat lunak (SDK) tersebut, sementara semua aplikasi lainnya akan terus menggunakan model izin sebelumnya.

Marshmallow juga memiliki skema manajemen daya baru bernama Doze yang mengurangi tingkat aktivitas aplikasi latar belakang saat perangkat menentukan bahwa itu tidak sedang aktif ditangani oleh pengguna, yang, menurut Google, menggandakan pemakaian baterai perangkat. Hal ini juga memperkenalkan pilihan untuk mengatur

ulang semua pengaturan jaringan, tersedia untuk pertama kalinya pada Android, yang membersihkan pengaturan terkait jaringan untuk WI-FI, Bluetooth dan koneksi seluler.

Android Marshmallow memberikan dukungan asli untuk pengenalan sidik jari, memungkinkan penggunaan sidik jari untuk membuka perangkat dan otentikasi Play Store dan pembelian Android Pay; API standar juga tersedia untuk melaksanakan otentikasi berbasis sidik jari dalam aplikasi lain. Android Marshmallow mendukung USB Type-C, termasuk kemampuan untuk menginstruksikan perangkat untuk mengisi daya perangkat lain melalui USB. Marshmallow juga memperkenalkan "pranala yang diverifikasi" yang dapat dikonfigurasi untuk membuka langsung dalam aplikasi tertentu mereka tanpa petunjuk pengguna lanjut.

Versi API Android yang disediakan oleh Marshmallow adalah 23. Alat pengembang Android Marshmallow tersedia di Pengelola SDK di bawah tingkat API "MNC".

14. Android 7.0-7.1 Nougat (API level 24-25)

Android "Nougat" (kode nama N dalam pengembangan) adalah rilis 7.0 besar dari sistem operasi Android. Ini pertama kali dirilis sebagai pratinjau pengembang pada tanggal 9 Maret 2016, dengan gambar pabrik untuk perangkat Nexus saat ini, serta dengan "Program Beta Beta" baru yang memungkinkan perangkat yang didukung ditingkatkan versinya ke versi Android Nougat melalui over-the -air update. Rilis terakhir adalah pada tanggal 22 Agustus 2016. Pratinjau akhir pembuatannya dirilis pada tanggal 18 Juli 2016, dengan nomor bangunan NPD90G.

Pada tanggal 19 Oktober 2016, Google merilis Android 7.1.1 sebagai pratinjau pengembang untuk Nexus 5X, Nexus 6P dan Pixel C. Pratinjau kedua mulai tersedia pada 22 November 2016, sebelum versi final diluncurkan ke publik pada bulan Desember. 5, 2016.

15. Android 8.0 Oreo (API level 26)

Android Oreo adalah rilis utama ke 8 dari sistem operasi Android. Ini pertama kali dirilis sebagai preview pengembang pada tanggal 21 Maret 2017, dengan gambar

pabrik untuk perangkat Nexus dan Pixel saat ini. Pratinjau pengembang terakhir dirilis pada tanggal 24 Juli 2017, dengan rilis stabil yang diharapkan pada bulan Agustus atau September 2017.

2.4.5 Siklus Hidup Android

Saat pengguna menjelajahi, keluar, dan kembali ke aplikasi Anda, contoh Aktivitas dalam transisi aplikasi Anda melalui berbagai *state* dalam siklus hidup mereka. Kelas Aktivitas menyediakan sejumlah callback yang memungkinkan aktivitas mengetahui bahwa sebuah *state* telah berubah: bahwa sistem sedang menciptakan, menghentikan, atau melanjutkan suatu aktivitas, atau menghancurkan proses di mana aktivitas berada.

Dalam metode callback siklus hidup, Anda dapat menyatakan bagaimana aktivitas Anda berperilaku saat pengguna pergi dan kembali memasuki aktivitas. Misalnya, jika Anda membangun pemutar video streaming, Anda dapat menjeda sementara video dan menghentikan sambungan jaringan saat pengguna beralih ke aplikasi lain. Saat pengguna kembali, Anda dapat menyambung kembali ke jaringan dan mengizinkan pengguna untuk melanjutkan video dari tempat yang sama. Dengan kata lain, setiap panggilan balik memungkinkan Anda melakukan pekerjaan spesifik yang sesuai dengan perubahan keadaan yang diberikan. Melakukan pekerjaan yang benar pada saat yang tepat dan menangani transisi dengan benar membuat aplikasi Anda lebih kuat dan performant. Misalnya, penerapan callback yang bagus dapat membantu memastikan aplikasi Anda dihindari:

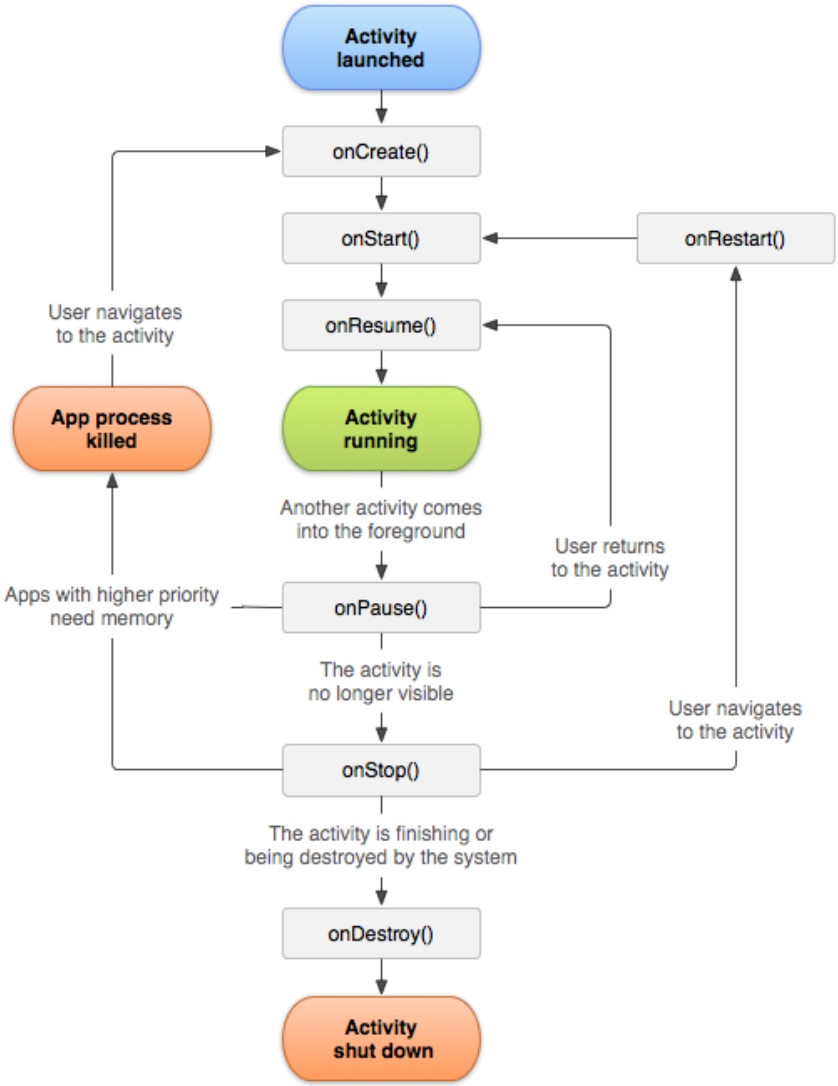
- *Crashing* jika pengguna menerima panggilan telepon atau beralih ke aplikasi lain saat menggunakan aplikasi Anda.
- Mengkonsumsi sumber daya sistem yang berharga saat pengguna tidak aktif menggunakannya.
- Kehilangan kemajuan pengguna jika mereka meninggalkan aplikasi Anda dan kembali lagi nanti.

- *Crashing* atau kehilangan kemajuan pengguna saat layar berputar antara orientasi landscape dan portrait.

Dokumen ini menjelaskan siklus hidup kegiatan secara rinci. Dokumen ini dimulai dengan menggambarkan paradigma siklus hidup. Selanjutnya, ini menjelaskan masing-masing callback: apa yang terjadi secara internal saat mereka melaksanakannya, dan apa yang harus Anda terapkan selama mereka melakukannya. Kemudian secara singkat mengenalkan hubungan antara keadaan aktivitas dan kerentanan proses untuk dibunuh oleh sistem. Terakhir, ini membahas beberapa topik yang berkaitan dengan transisi antara aktivitas *State*.

2. Konsep Siklus Hidup Aktivitas

Untuk menavigasi transisi antara tahapan siklus hidup aktivitas, kelas Aktivitas menyediakan satu set inti dari enam callback: onCreate (), onStart (), onResume (), onPause (), onStop (), dan onDestroy (). Sistem memanggil masing-masing callback ini sebagai sebuah aktivitas memasuki sebuah negara baru.



Gambar 2.5 Sebuah ilustrasi sederhana dari siklus hidup aktivitas

Saat pengguna mulai meninggalkan aktivitas, sistem memanggil metode untuk membongkar aktivitas. Dalam beberapa kasus, pembongkaran ini hanya bersifat parsial; 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 akan dilanjutkan dari tempat pengguna berhenti. Kemungkinan sistem membunuh proses tertentu - bersamaan dengan aktivitas di dalamnya -

tergantung pada keadaan aktivitas pada saat itu. Status aktivitas dan ejeksi dari memori memberikan lebih banyak informasi tentang hubungan antara negara dan kerentanan terhadap ejeksi.

Bergantung pada kompleksitas aktivitas Anda, Anda mungkin tidak perlu menerapkan semua metode siklus hidup. Namun, penting bagi Anda untuk memahami masing-masing dan menerapkannya yang memastikan aplikasi Anda sesuai dengan harapan pengguna.

Bagian selanjutnya dari dokumen ini memberikan rincian tentang callback yang Anda gunakan untuk menangani transisi antar negara

2. Siklus Hidup Callback

onCreate()

Anda harus menerapkan callback ini, yang menyala saat sistem pertama kali menciptakan aktivitas. Pada kegiatan penciptaan, aktivitas memasuki *state* yang Dibuat. Dalam metode onCreate (), Anda menjalankan logika startup aplikasi dasar yang seharusnya terjadi hanya sekali seumur hidup aktivitas. Misalnya, penerapan onCreate () Anda mungkin mengikat data ke daftar, menginisialisasi benang latar belakang, dan memberi contoh beberapa variabel kelas. Metode ini menerima parameter savedInstanceState, yang merupakan objek Bundle yang berisi status aktivitas yang tersimpan sebelumnya. Jika aktivitas tidak pernah ada sebelumnya, nilai dari objek *Bundle* adalah *null*.

Contoh berikut tentang metode onCreate () menunjukkan penyiapan mendasar untuk aktivitas tersebut, seperti menyatakan antarmuka pengguna (didefinisikan dalam file tata letak XML), menentukan variabel anggota, dan mengkonfigurasi beberapa UI. Dalam contoh ini, file tata letak XML ditentukan dengan mengirimkan ID sumber file R.layout.main_activity to setContentView()

```
TextView mTextView;

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

@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) {
        mGameState = 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
    mTextView = (TextView) findViewById(R.id.text_view);
}

// This callback is called only when there is a saved instance previously saved 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) {
    mTextView.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, mGameState);
    outState.putString(TEXT_VIEW_KEY, mTextView.getText());

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

Gambar 2.6 *Metode onCreate*()

Sebagai alternatif untuk mendefinisikan file XML dan meneruskannya ke setContentView (), Anda dapat 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 akar ViewGroup ke setContentView (). Untuk informasi lebih lanjut tentang membuat antarmuka pengguna, lihat dokumentasi *User Interface*.

Aktivitas Anda tidak berada di negara yang Dibuat. Setelah metode onCreate () menyelesaikan eksekusi, aktivitas memasuki keadaan Started, dan sistem memanggil metode onStart () dan onResume () secara berurutan. Bagian selanjutnya menjelaskan onStart () callback.

onStart()

Saat aktivitas memasuki keadaan Started, sistem memanggil callback ini. Panggilan onStart () membuat aktivitas terlihat oleh pengguna, karena aplikasi mempersiapkan aktivitas untuk masuk ke latar depan dan menjadi interaktif. Misalnya, metode ini adalah tempat aplikasi menginisialisasi kode yang mempertahankan UI. Mungkin juga mendaftarkan BroadcastReceiver yang memantau perubahan yang tercermin di UI.

Metode onStart () selesai dengan sangat cepat dan, seperti pada negara yang Dibuat, aktivitas tersebut tidak tinggal di negara bagian Started. Setelah panggilan balik ini selesai, aktivitas memasuki status yang dilanjutkan, dan sistem memanggil metode onResume ().

onResume()

Ketika aktivitas memasuki keadaan yang dilanjutkan, ia datang ke latar depan, dan kemudian sistem memanggil onResume () callback. Ini adalah status di mana aplikasi berinteraksi dengan pengguna. Aplikasi tetap berada di negara ini sampai terjadi sesuatu yang mengambil fokus dari aplikasi. Peristiwa semacam itu mungkin, misalnya, menerima panggilan telepon, pengguna menavigasi ke aktivitas lain, atau mematikan layar perangkat.

Saat terjadi peristiwa interupsi, aktivitas memasuki keadaan Dijeda, dan sistem memanggil onpause () panggil balik.

Jika aktivitas kembali ke status Kembali dari keadaan Dijeda, sistem sekali lagi memanggil metodeResume (). Untuk alasan ini, Anda harus menerapkan onResume () untuk menginisialisasi komponen yang Anda rilis saat onPause (). Misalnya, Anda dapat menginisialisasi kamera sebagai berikut:

```
@Override
public void onResume() {
    super.onResume();  // Always call the superclass method first


    // Get the Camera instance as the activity achieves full user focus
    if (mCamera == null) {
        initializeCamera(); // Local method to handle camera init
    }
}
```

Gambar 2.7 Contoh *onResume()*

Sadarilah bahwa sistem memanggil metode ini setiap kali aktivitas Anda masuk ke latar depan, termasuk saat dibuat untuk pertama kalinya. Dengan demikian, Anda harus menerapkan onResume () untuk menginisialisasi komponen yang Anda rilis saat onPause (), dan melakukan inisialisasi lainnya yang harus dilakukan setiap kali aktivitas memasuki status yang dilanjutkan. Misalnya, Anda harus memulai animasi dan menginisialisasi komponen yang aktivitasnya hanya digunakan bila memiliki fokus pada pengguna.

onPause()

Sistem memanggil metode ini sebagai indikasi pertama bahwa pengguna meninggalkan aktivitas Anda (meski tidak selalu berarti aktivitas tersebut dihancurkan). Gunakan metode onPause () untuk menjeda operasi seperti animasi dan pemutaran musik yang tidak boleh berlanjut sementara Aktivitas berada dalam keadaan Dijeda, dan Anda akan segera melanjutkan. Ada beberapa alasan mengapa suatu aktivitas dapat memasuki keadaan ini. Sebagai contoh:

- Beberapa acara mengganggu eksekusi aplikasi, seperti yang dijelaskan di bagian onResume (). Ini adalah kasus yang paling umum.
- Di Android 7.0 (tingkat API 24) atau lebih tinggi, beberapa aplikasi berjalan dalam mode multi-jendela. Karena hanya satu aplikasi (windows) yang fokus setiap saat, sistem akan menghentikan semua aplikasi lainnya.
- Aktivitas baru semi-transparan (seperti dialog) akan terbuka. Selama aktivitas tersebut masih terlihat sebagian tapi tidak fokus, tetap saja berhenti sejenak.

Anda dapat menggunakan metode onPause () untuk melepaskan sumber daya sistem, seperti receiver siaran, menangani sensor (seperti GPS), atau sumber daya yang dapat mempengaruhi masa pakai baterai sementara aktivitas Anda dihentikan sementara dan pengguna tidak memerlukannya.

Misalnya, jika aplikasi Anda menggunakan Camera, metode onPause () adalah tempat yang baik untuk melepaskannya. Contoh berikut onPause () adalah pendamping untuk onResume () contoh di atas, melepaskan kamera yang diindeks onResume () diinisialisasi.

```
@Override
public void onPause() {
    super.onPause();  // Always call the superclass method first


    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release();
        mCamera = null;
    }
}
```

Gambar 2.8 Contoh *onPause()*

onPause () eksekusi sangat singkat, dan tidak harus cukup waktu untuk melakukan save operations. Untuk alasan ini, Anda tidak boleh menggunakan onPause () untuk menyimpan aplikasi atau data pengguna, melakukan panggilan jaringan, atau melakukan transaksi database; Pekerjaan semacam itu mungkin tidak lengkap sebelum metode selesai. Sebagai gantinya, Anda harus melakukan operasi shutdown beban berat selama onStop (). Untuk informasi lebih lanjut tentang pengoperasian yang sesuai untuk dilakukan selama onStop (), lihat onStop (). Untuk informasi lebih lanjut tentang menyimpan data, lihat Menyimpan dan mengembalikan status aktivitas.

Penyelesaian metode onPause () tidak berarti bahwa aktivitas tersebut meninggalkan keadaan Dijeda. Sebaliknya, aktivitas tetap berada dalam keadaan ini sampai aktivitas tersebut dilanjutkan atau menjadi tidak terlihat oleh pengguna. Jika aktivitas dilanjutkan, sistem sekali lagi memanggil onResume () callback. Jika aktivitas

kembali dari keadaan Dijeda ke keadaan Kembali, sistem menyimpan instance Aktivitas yang tinggal di memori, mengingat contoh itu saat sistem memanggil onResume (). Dalam skenario ini, Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan ulang yang mengarah ke status Kembali. Jika aktivitas menjadi sama sekali tidak terlihat, sistem akan memanggil ONStop (). Bagian selanjutnya membahas callback onStop ().

onStop()

Bila aktivitas Anda tidak lagi terlihat oleh pengguna, akun tersebut telah memasuki status Berhenti, dan sistem memanggil onback on call (). Hal ini dapat terjadi, misalnya, saat aktivitas yang baru diluncurkan mencakup keseluruhan layar. Sistem juga dapat menghubungi onStop () saat aktivitas selesai berjalan, dan akan segera dihentikan.

Pada metode onStop (), aplikasi harus melepaskan hampir semua sumber yang tidak diperlukan saat pengguna tidak menggunakannya. Misalnya, jika Anda mendaftarkan BroadcastReceiver di onStart () untuk mendengarkan perubahan yang mungkin memengaruhi UI Anda, Anda dapat membatalkan registrasi penerima siaran di onStop (), karena pengguna tidak dapat lagi melihat UI. Hal ini juga penting bahwa Anda menggunakan onStop () untuk melepaskan sumber daya yang mungkin menghabiskan memori, karena memungkinkan sistem membunuh proses yang menghosting aktivitas Anda tanpa menghubungi callback onDestroy () aktivitas akhir.

Anda juga harus menggunakan onStop () untuk melakukan operasi shutdown CPU yang intensif. Misalnya, jika Anda tidak dapat menemukan waktu yang lebih tepat untuk menyimpan informasi ke database, Anda mungkin melakukannya saat onStop (). Contoh berikut menunjukkan penerapan onStop () yang menyimpan isi draf catatan 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
    mAsyncQueryHandler.startUpdate (
            mToken,  // int token to correlate calls
            null,    // cookie, not used here
            mUri,    // 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 9 Contoh *onStop()*

Ketika aktivitas Anda memasuki keadaan Berhenti, objek Aktivitas disimpan di memori: ia memelihara semua informasi keadaan dan anggota, namun tidak melekat pada window manager. Saat aktivitas dilanjutkan, kegiatan mengingat informasi ini. Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan ulang yang mengarah ke status Kembali. Sistem juga melacak keadaan saat ini untuk setiap objek View dalam tata letak, jadi jika pengguna memasukkan teks ke dalam widget EditText, konten itu akan tersimpan sehingga Anda tidak perlu menyimpan dan mengembalikannya.

Catatan: Setelah aktivitas Anda dihentikan, sistem mungkin akan menghancurkan proses yang berisi aktivitas jika sistem perlu memulihkan memori. Bahkan jika sistem menghancurkan proses sementara aktivitas dihentikan, sistem tetap mempertahankan keadaan objek View (seperti teks dalam widget EditText) dalam Bundle (segumpal pasangan kunci-nilai) dan mengembalikannya jika pengguna menavigasi kembali ke aktivitas. Untuk informasi lebih lanjut tentang mengembalikan aktivitas ke pengguna yang kembali, lihat Menyimpan dan memulihkan status aktivitas.

Dari keadaan Berhenti, aktivitas tersebut kembali berinteraksi dengan pengguna, atau aktivitas selesai berjalan dan hilang. Jika aktivitasnya kembali, sistem memanggil onRestart (). Jika Aktivitas selesai berjalan, sistem akan memanggil onDestroy (). Bagian selanjutnya menjelaskan onDestroy () callback.

onDestroy()

Disebut sebelum aktivitas hancur. Ini adalah panggilan terakhir yang aktivitasnya diterima. Sistem ini memanggil callback ini karena aktivitasnya selesai karena pemanggilan seseorang selesai (), atau karena sistem tersebut untuk sementara menghancurkan proses yang berisi aktivitas untuk menghemat ruang. Anda dapat membedakan kedua skenario dengan metode isFinishing (). Sistem juga dapat memanggil metode ini saat terjadi perubahan orientasi, dan kemudian segera menghubungi onCreate () untuk menciptakan kembali proses (dan komponen yang dikandungnya) dalam orientasi baru.

OnDestroy () callback merilis semua sumber yang belum dirilis oleh callback sebelumnya seperti onStop ()

2.4.6 Fitur Android

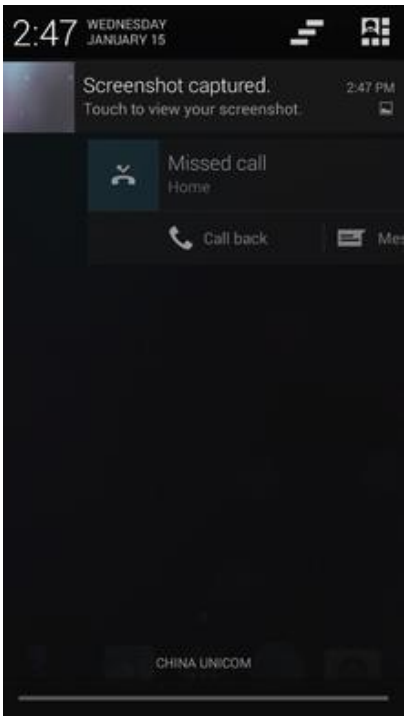
1. Antarmuka

Antarmuka pengguna pada Android didasarkan pada manipulasi langsung, menggunakan masukan sentuh yang serupa dengan tindakan di dunia nyata, misalnya menggesek (*swiping*), mengetuk (*tapping*), dan mencubit (*pinching*), untuk memanipulasi objek di layar. Masukan pengguna direspon dengan cepat dan juga tersedia antarmuka sentuh layaknya permukaan air, seringkali menggunakan kemampuan getaran perangkat untuk memberikan umpan balik haptik kepada pengguna. Perangkat keras internal seperti akselerometer, giroskop, dan sensor proksimitas digunakan oleh beberapa aplikasi untuk merespon tindakan pengguna, misalnya untuk menyesuaikan posisi layar dari potret ke lanskap, tergantung pada

bagaimana perangkat diposisikan, atau memungkinkan pengguna untuk mengarahkan kendaraan saat bermain balapan dengan memutar perangkat sebagai simulasi kendali setir.

Ketika dihidupkan, perangkat Android akan memuat pada layar depan (*homescreen*), yakni navigasi utama dan pusat informasi pada perangkat, serupa dengan desktop pada komputer pribadi. Layar depan Android biasanya terdiri dari ikon aplikasi dan widget; ikon aplikasi berfungsi untuk menjalankan aplikasi terkait, sedangkan widget menampilkan konten secara langsung dan terbarui otomatis, misalnya prakiraan cuaca, kotak masuk surel pengguna, atau menampilkan tiker berita secara langsung dari layar depan. Layar depan bisa terdiri dari beberapa halaman, pengguna dapat menggeser bolak balik antara satu halaman ke halaman lainnya, yang memungkinkan pengguna Android untuk mengatur tampilan perangkat sesuai dengan selera mereka. Beberapa aplikasi pihak ketiga yang tersedia di Google Play dan di toko aplikasi lainnya secara ekstensif mampu mengatur kembali tema layar depan Android, dan bahkan bisa meniru tampilan sistem operasi lain, misalnya Windows Phone. Kebanyakan produsen telepon seluler dan operator nirkabel menyesuaikan tampilan perangkat Android buatan mereka untuk membedakannya dari pesaing mereka.

Di bagian atas layar terdapat status bar, yang menampilkan informasi tentang perangkat dan konektivitasnya. Status bar ini bisa "ditarik" ke bawah untuk membuka layar notifikasi yang menampilkan informasi penting atau pembaruan aplikasi, misalnya surel diterima atau SMS masuk, dengan cara tidak mengganggu kegiatan pengguna pada perangkat. Pada versi awal Android, layar notifikasi ini bisa digunakan untuk membuka aplikasi yang relevan, namun setelah diperbarui, fungsi ini semakin disempurnakan, misalnya kemampuan untuk memanggil kembali nomor telepon dari notifikasi panggilan tak terjawab tanpa harus membuka aplikasi utama. Notifikasi ini akan tetap ada sampai pengguna melihatnya, atau dihapus dan di nonaktifkan oleh pengguna.



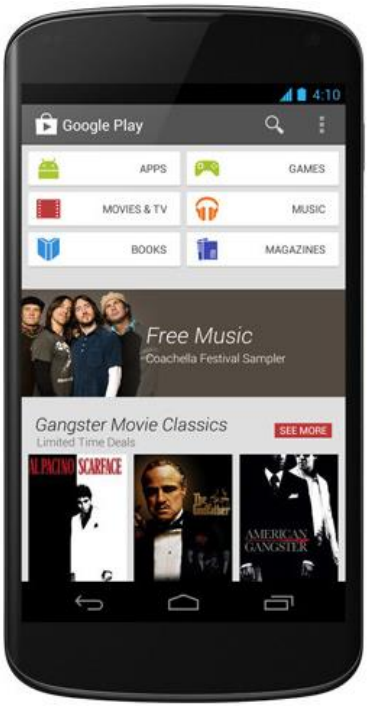
Gambar 2.10 Antarmuka

2. Aplikasi

Android memungkinkan penggunanya untuk memasang aplikasi pihak ketiga, baik yang diperoleh dari toko aplikasi seperti Google Play, Amazon Appstore, ataupun dengan mengunduh dan memasang berkas APK dari situs pihak ketiga. Di Google Play, pengguna bisa menjelajah, mengunduh, dan memperbarui aplikasi yang diterbitkan oleh Google dan pengembang pihak ketiga, sesuai dengan persyaratan kompatibilitas Google. Google Play akan menyaring daftar aplikasi yang tersedia berdasarkan kompatibilitasnya dengan perangkat pengguna, dan pengembang dapat membatasi aplikasi ciptaan mereka bagi operator atau negara tertentu untuk alasan bisnis. Pembelian aplikasi yang tidak sesuai dengan keinginan pengguna dapat dikembalikan dalam waktu 15 menit setelah pengunduhan. Beberapa operator seluler juga menawarkan tagihan langsung untuk pembelian aplikasi di Google Play dengan cara menambahkan harga pembelian aplikasi pada tagihan bulanan pengguna. Pada

bulan September 2012, ada lebih dari 675.000 aplikasi yang tersedia untuk Android, dan perkiraan jumlah aplikasi yang diunduh dari Play Store adalah 25 miliar.

Aplikasi Android dikembangkan dalam bahasa pemrograman Java dengan menggunakan kit pengembangan perangkat lunak Android (SDK). SDK ini terdiri dari seperangkat perkakas pengembangan, termasuk debugger, perpustakaan perangkat lunak, emulator handset yang berbasis QEMU, dokumentasi, kode sampel, dan tutorial. Didukung secara resmi oleh lingkungan pengembangan terpadu (IDE) Eclipse, yang menggunakan plugin Android Development Tools (ADT). Perkakas pengembangan lain yang tersedia di antaranya adalah Native Development Kit untuk aplikasi atau ekstensi dalam C atau C++, Google App Inventor, lingkungan visual untuk pemrogram pemula, dan berbagai kerangka kerja aplikasi web seluler lintas platform.



Gambar 2.11 Tampilan *Google Play*

3. Pengelolaan Memori

Karena perangkat Android umumnya bertenaga baterai, Android dirancang untuk mengelola memori (RAM) guna menjaga konsumsi daya minimal, berbeda dengan sistem operasi desktop yang bisa terhubung pada sumber daya listrik tak terbatas. Ketika sebuah aplikasi Android tidak lagi digunakan, sistem secara otomatis akan menangguhkannya (suspend) dalam memori – secara teknis aplikasi tersebut masih "terbuka", namun dengan ditangguhkan, aplikasi tidak akan mengkonsumsi sumber daya (misalnya daya baterai atau daya pemrosesan), dan akan "diam" di latar belakang hingga aplikasi tersebut digunakan kembali. Cara ini memiliki manfaat ganda, tidak hanya meningkatkan respon perangkat Android karena aplikasi tidak perlu ditutup dan dibuka kembali dari awal setiap saat, tetapi juga memastikan bahwa aplikasi yang berjalan di latar belakang tidak menghabiskan daya secara sia-sia.

Android mengelola aplikasi yang tersimpan di memori secara otomatis: ketika memori lemah, sistem akan menonaktifkan aplikasi dan proses yang tidak aktif untuk sementara waktu, aplikasi akan dinonaktifkan dalam urutan terbalik, dimulai dari yang terakhir digunakan. Proses ini tidak terlihat oleh pengguna, jadi pengguna tidak perlu mengelola memori atau menonaktifkan aplikasi secara manual. Namun, kebingungan pengguna atas pengelolaan memori pada Android telah menyebabkan munculnya beberapa aplikasi task killer pihak ketiga yang populer di Google Play.[5]

2.4.7 Prinsip Desain

Prinsip desain ini dikembangkan oleh dan untuk tim *Android User Experience* agar selalu mempertimbangkan kepentingan pengguna. Untuk pengembangan dan desainer Android, mereka terus meletakkan dasar pedoman desain yang lebih detail untuk beragam tipe perangkat.

1. **Enchant Me**

- *Delight me in surprising ways*

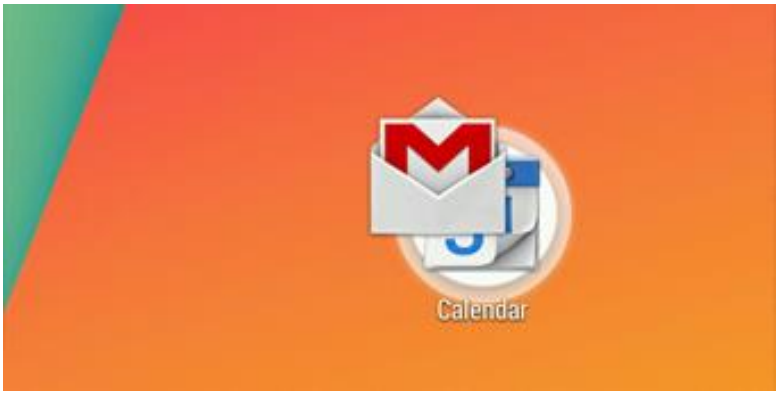
Permukaan yang cantik, animasi yang ditempatkan dengan hati-hati, atau efek suara di saat yang tepat sungguh menyenangkan untuk dinikmati. Efek yang lembut menimbulkan perasaan serba mudah dan kesadaran bahwa kekuatan yang bisa diandalkan ada dalam genggaman



Gambar 2.12 ***Delight me in surprising ways***

- *Real Object are more fun than buttons and menus*

Biarkan orang langsung menyentuh dan memanipulasi objek dalam aplikasi Anda. Ini mengurangi upaya kognitif yang diperlukan untuk menjalankan tugas sekaligus membuatnya lebih memuaskan secara emosional



Gambar 2.13 ***Real object are more fun than buttons and menus***

- *Let me make it mine*

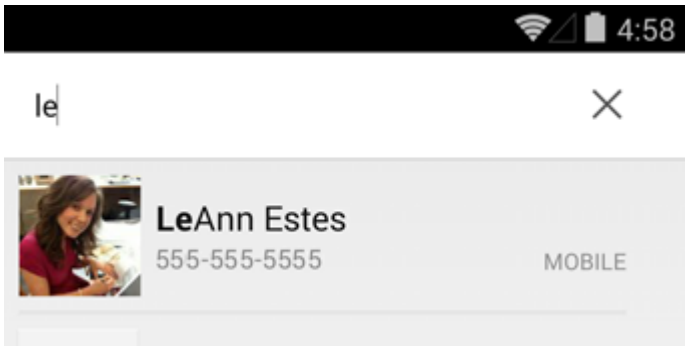
Orang suka menambahkan sentuhan pribadi karena membantu mereka merasa betah dan memegang kendali. Memberikan default yang pantas dan indah, tetapi juga mempertimbangkan penyesuaian opsional yang menyenangkan, yang tidak mengganggu tugas utama.



Gambar 2.14 *Let make it mine*

- *Get to know me*

Pelajari preferensi orang dari waktu ke waktu. Daripada meminta mereka untuk membuat pilihan yang sama berulang-ulang, tempatkan pilihan sebelumnya agar mudah dijangkau.

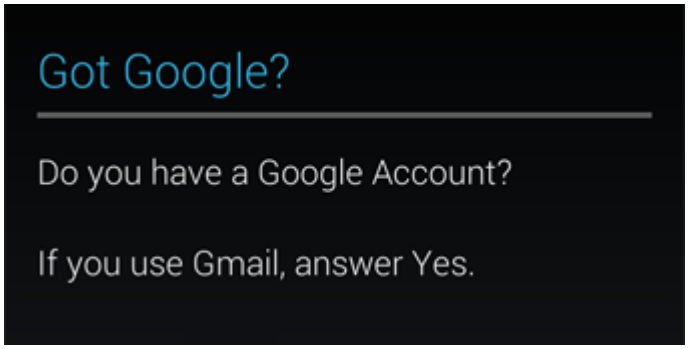


Gambar 2.15 *Get to know me*

2. Simplify My Life

- *Keep it brief*

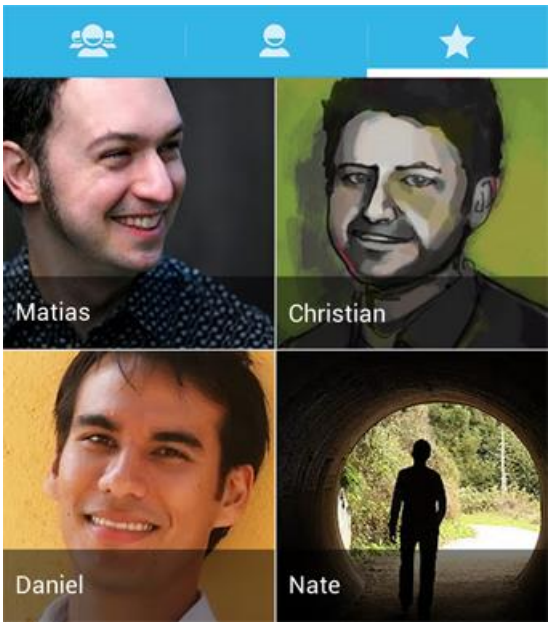
Gunakan frasa pendek dengan kata-kata sederhana. Orang cenderung melewatkan kalimat-kalimat panjang.



Gambar 2.16 **Keep it brief**

- *Picture are faster than words*

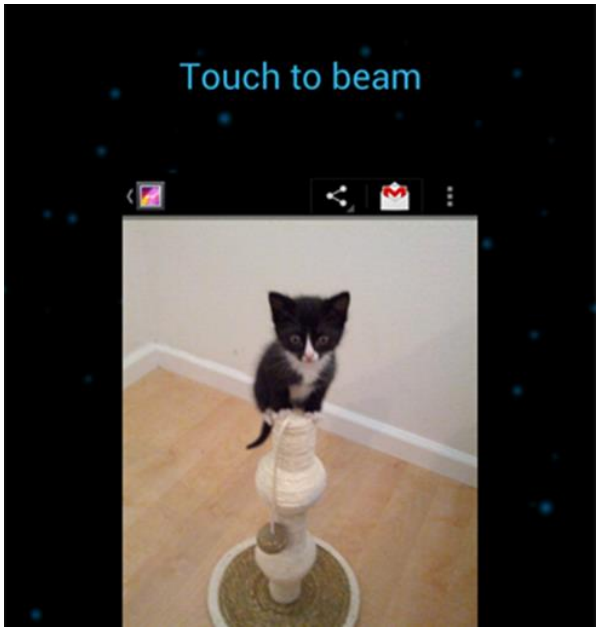
Pertimbangkan menggunakan gambar untuk menjelaskan gagasan. Gambar menarik perhatian orang dan bisa jauh lebih efisien dibanding kata-kata.



Gambar 2.17 ***Picture are faster than words***

- *Decide for me but let me have the final say*

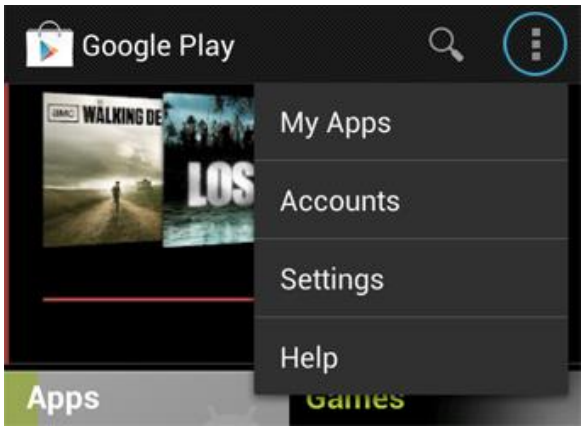
Gunakan tebakan terbaik Anda dan bertindaklah daripada meminta terlebih dahulu. Terlalu banyak pilihan dan keputusan membuat orang tidak suka. Untuk berjaga-jaga jika Anda salah, izinkan 'pembatalan'.



Gambar 2.18 ***Decide for me but let me have the final say***

- *Only show what I need when I need it*

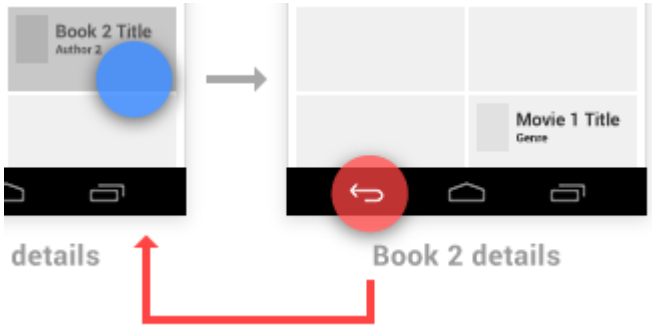
Orang merasa kewalahan ketika melihat terlalu banyak hal sekaligus. Uraikan tugas dan informasi menjadi potongan-potongan kecil yang mudah dicerna. Sembunyikan opsi yang tidak perlu pada saat ini, dan ajari orang sambil jalan.



Gambar 2.19 ***Only show what I need when I need it***

- *I should always know where I am*

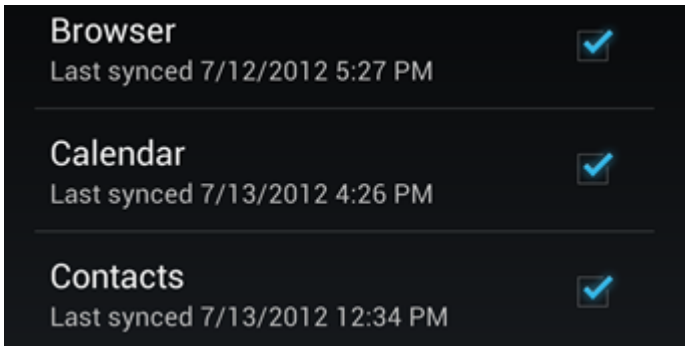
Beri orang kepercayaan diri bahwa mereka tahu di mana berada. Buat agar tempat-tempat dalam aplikasi Anda terlihat berbeda dan gunakan transisi untuk menunjukkan hubungan antar layar. Berikan umpan balik tentang tugas yang sedang berlangsung.



Gambar 2.20 ***I should always know where I am***

- *Never lose my stuff*

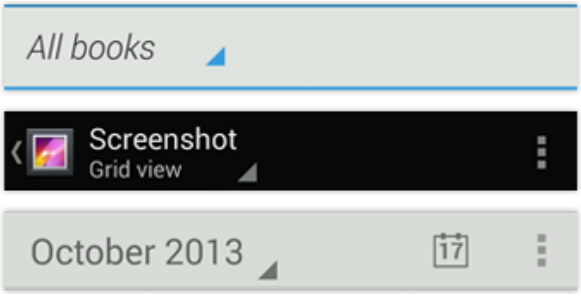
Simpan apa yang telah susah-payah dibuat orang dan biarkan mereka mengaksesnya dari mana saja. Ingat pengaturan, sentuhan pribadi, dan kreasi lintas ponsel, tablet, dan komputer. Itu membuat pemutakhiran menjadi hal termudah di dunia.



Gambar 2.21 ***Never lose my stuff***

- *If it looks the same, it should act the same*

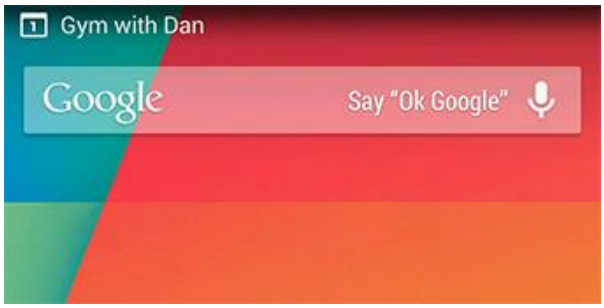
Bantu orang merasakan perbedaan fungsional dengan membuat mereka terlihat berbeda daripada mirip. Hindari mode, yaitu tempat yang terlihat mirip tetapi berbeda fungsinya pada input yang sama.



Gambar 2.22 ***If it looks the same, it should act the same***

- *Only interrupt me if its important*

Layaknya asisten pribadi yang baik, lindungi orang dari detail yang tidak penting. Orang ingin tetap fokus, dan kecuali jika memang penting dan sensitif waktu, interupsi bisa melelahkan dan menjengkelkan.

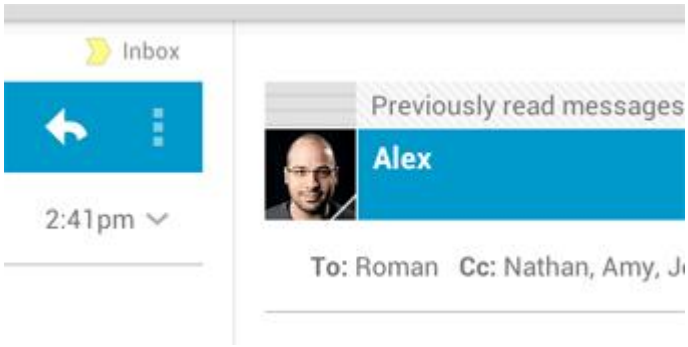


Gambar 2.23 ***Only interrupt me if its important***

3. Make Me Amazing

- *Give me trick that work everywhere*

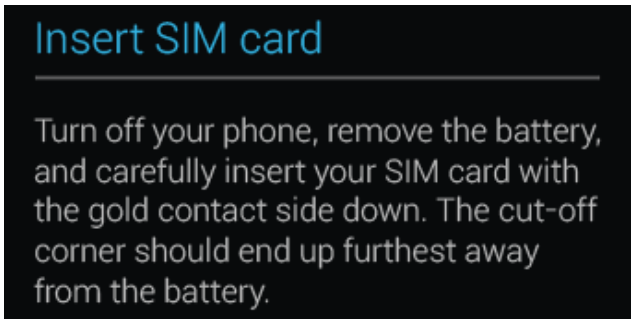
Orang merasa senang ketika mereka memahami sendiri sesuatu. Jadikan aplikasi Anda lebih mudah dipelajari dengan memanfaatkan pola visual dan memori otot dari aplikasi Android lainnya. Misalnya, gerakan menggeser dapat menjadi pintasan navigasi yang bagus.



Gambar 2.24 ***Give me trick that work everywhere***

- *Its not my fault*

Bersikap ramahlah dalam meminta orang untuk melakukan koreksi. Mereka ingin merasa pintar ketika menggunakan aplikasi Anda. Jika terjadi kesalahan, berikan petunjuk perbaikan yang jelas tetapi lepaskan mereka dari detail teknis. Jika Anda dapat memperbaikinya secara diam-diam, tentu lebih baik.



Gambar 2.25 ***Its not my fault***

- *Springkle encouragement*

Uraikan tugas-tugas rumit menjadi langkah-langkah kecil yang dapat dilakukan dengan mudah. Beri umpan balik tentang tindakan, meskipun hanya sesuatu yang sederhana.



Gambar 2.26 ***Springkle encouragement***

- *Do the heavy lifting for me*

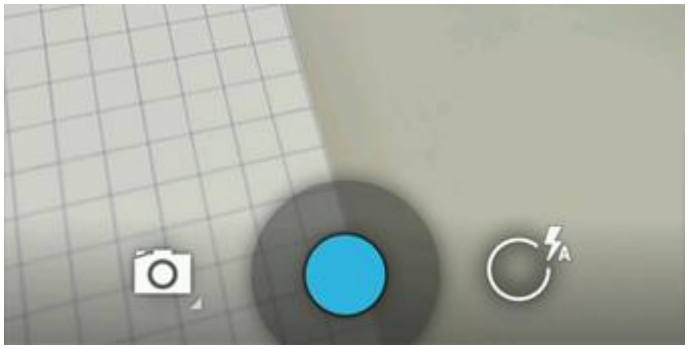
Buatlah pemula merasa seperti ahli dengan memungkinkan mereka untuk melakukan hal-hal yang mereka pikir tidak akan bisa. Misalnya, pintasan yang menggabungkan beberapa efek foto dapat membuat foto amatir terlihat mengagumkan hanya dalam beberapa langkah.



Gambar 2.27 ***Do the heavy lifting for me***

- *Make important things fast*

Tidak semua tindakan itu sama. Putuskan apa yang terpenting dalam aplikasi Anda dan permudah untuk menemukannya serta cepat untuk digunakan, seperti tombol rana pada kamera, atau tombol jeda pada pemutar musik.



Gambar 2.28 ***Make important things fast***

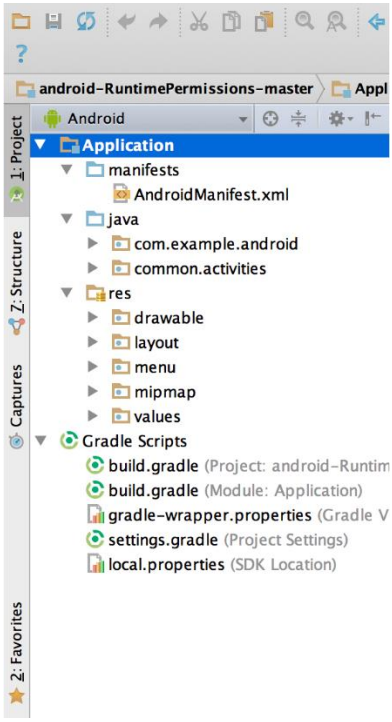
2.4.8 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan *IntelliJ* IDEA . Selain merupakan editor kode *IntelliJ* dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya:

- Sistem versi berbasis Gradle yang fleksibel

- Emulator yang cepat dan kaya fitur
- Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
- Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
- Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
- Alat pengujian dan kerangka kerja yang ekstensif
- Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
- Dukungan C++ dan NDK
- Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*

1. Struktur Proyek



Gambar 2.29 *File proyek di tampilan 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*

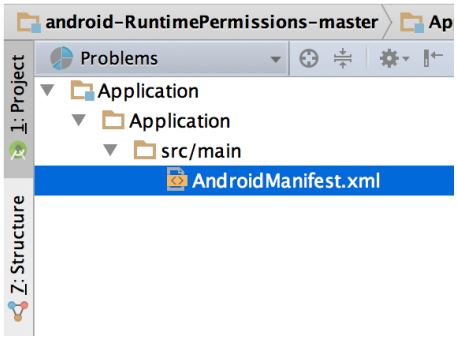
Secara default, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 1. Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke file sumber utama proyek Anda.

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.

Struktur 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).

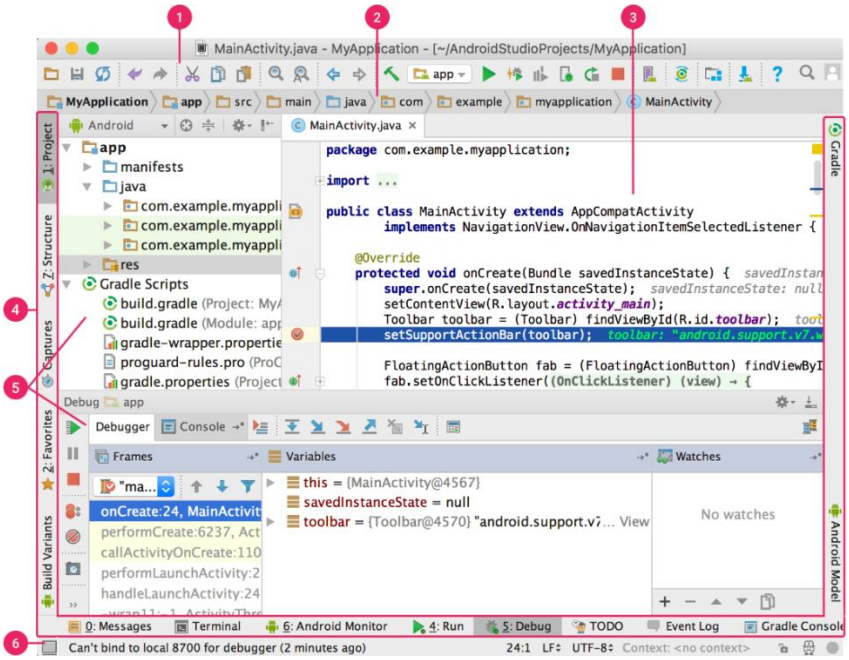
Anda juga bisa menyesuaikan tampilan file proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan Problems dari tampilan proyek Anda akan menampilkan tautan ke file sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen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



Gambar 2.30 ***File proyek dalam tampilan Problems***

2. Antarmuka Pengguna

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



Gambar 2.31 ***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.

Jendela Alat

Daripada 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 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.

Table 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**	**Command+Shift+F12**

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*.

Table 2.2 Keyboard shortcut untuk pelengkapan kode

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 berturut-turut, 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	**Control+Shift+Space**	**Control+Shift+Space**

	memanggil Pelengkapan Cerdas dua kali berturut-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).

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
```

Gambar 2.32 Kode sebelum pemformatan

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);

    // Get reference to the drawer layout and set event listener
```

Formatted 7 lines
Show reformat dialog: ⌥⇧⌘L

Gambar 2.33 Kode setelah pemformatan

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

3. 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 tombolSignInButton

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).

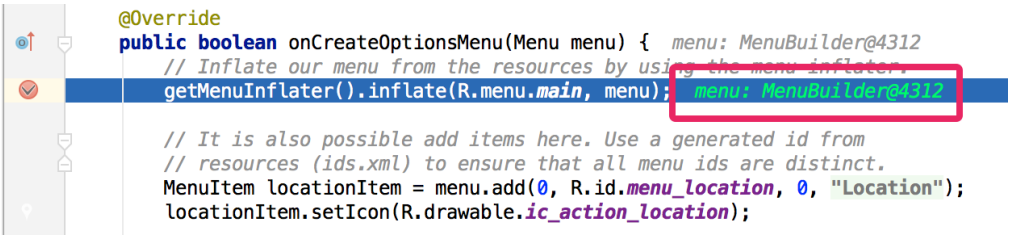
4. Alat Profil dan Debug

Android Studio membantu Anda dalam melakukan debug dan meningkatkan kinerja kode Anda, termasuk alat debug inline dan analisis kinerja.

Debug Inline

Gunakan debug inline untuk meningkatkan langkah-langkah kode Anda dalam tampilan debugger dengan verifikasi inline dari nilai referensi, ekspresi, dan variabel. Informasi debug inline meliputi:

- Nilai variabel inline
- Objek perujuk yang merujuk objek terpilih
- Nilai kembalian metode
- Ekspresi operator dan Lambda
- Nilai keterangan alat



Gambar 2.34 Nilai *variable inline*

Untuk mengaktifkan debug inline, di jendela Debug, klik Settings dan pilih kotak centang untuk *Show Values Inline*.

Pemantauan kinerja

Android Studio menyediakan pemantauan kinerja agar Anda dapat lebih mudah melacak penggunaan memori dan CPU aplikasi Anda, menemukan objek yang dibatalkan alokasinya, menemukan kebocoran memori, mengoptimalkan performa grafis, dan menganalisis permintaan jaringan. Dengan aplikasi yang berjalan pada perangkat atau emulator, buka jendela alat Android Monitor, lalu klik tab Monitors.

Heap dump

Saat Anda memantau penggunaan memori di Android Studio, Anda bisa secara bersamaan memulai pengumpulan sampah dan membuang heap Java ke cuplikan heap dalam file format biner HPROF khusus-Android. Penampil HPROF menampilkan kelas, instance setiap kelas, dan pohon referensi untuk membantu Anda melacak penggunaan memori dan menemukan kebocoran memori.

Pelacak alokasi

Android Studio memungkinkan Anda melacak alokasi memori saat memantau penggunaan memori. Melacak alokasi memori memungkinkan Anda untuk memantau tempat objek dialokasikan saat Anda melakukan tindakan tertentu. Mengetahui alokasi ini memungkinkan Anda untuk mengoptimalkan kinerja aplikasi dan penggunaan

memori dengan menyesuaikan metode panggilan yang terkait dengan tindakan tersebut.

Akses file data

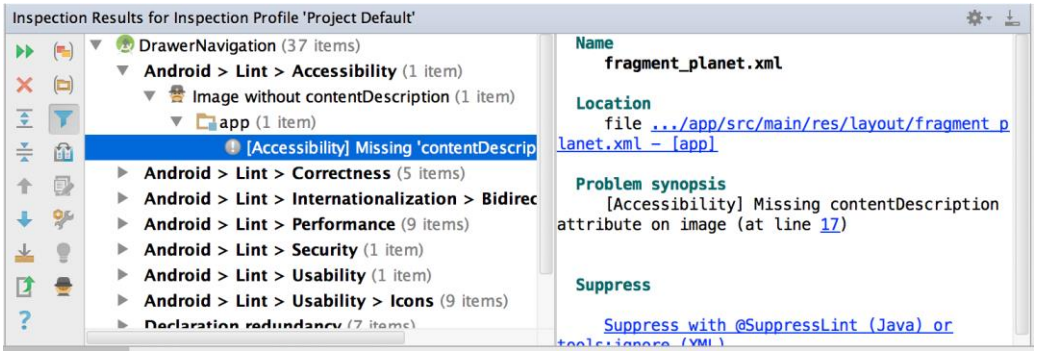
Alat Android SDK, seperti *Systrace*, *logcat*, dan *Traceview*, menghasilkan data kinerja dan debug untuk analisis aplikasi secara detail.

Untuk melihat ketersediaan file daya yang dihasilkan, buka jendela alat Captures. Dalam daftar file yang dihasilkan, klik ganda file tersebut untuk melihat datanya. Klik kanan file .hprof untuk mengonversinya ke format file standar .hprof.

Pemeriksaan kode

Kapan pun Anda mengompilasi program, Android Studio secara otomatis akan menjalankan Lint yang telah dikonfigurasi dan Pemeriksaan IDE untuk membantu Anda mengidentifikasi dan memperbaiki masalah secara mudah dengan kualitas struktur kode Anda.

Alat Lint memeriksa file sumber proyek Android Anda dari kemungkinan bug dan perbaikan optimalisasi untuk keakuratan, keamanan, kinerja, kegunaan, aksesibilitas, dan internasionalisasi.



Gambar 2.35 Hasil pemeriksaan Lint di Android Studio

Selain pemeriksaan Lint, Android Studio juga dan melakukan pemeriksaan kode IntelliJ dan memvalidasi anotasi untuk merampingkan alur kerja pengkodean Anda.

Anotasi di Android Studio

Android Studio mendukung anotasi untuk variabel, parameter, dan mengembalikan nilai untuk membantu Anda menangkap bug, misalnya pengecualian pointer nol dan konflik tipe sumber daya. Android SDK Manager memaketkan perpustakaan di Android Support Repository untuk digunakan dengan Android Studio. Android Studio akan memvalidasi anotasi yang dikonfigurasi selama pemeriksaan kode.

Pesan log

Saat membuat dan menjalankan aplikasi dengan Android Studio, Anda bisa melihat pesan output dan log perangkat dan adb (logcat) dengan mengeklik Android Monitor di bagian bawah jendela.

Jika Anda ingin men-debug aplikasi dengan Pemantauan Perangkat Android, Anda bisa membuka Pemantauan Perangkat dengan mengeklik *Tools* > *Android* > *Android Device Monitor*. Dalam Device Monitor Anda bisa menemukan seperangkat alat lengkap DDMS untuk mengidentifikasi aplikasi, mengontrol perilaku perangkat, dan sebagainya. Juga disertakan alat *Hierarchy Viewer* untuk membantu mengoptimalkan tata letak Anda.

2.5 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh *James Gosling* saat masih bergabung di *Sun Microsystems* saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code (bytecode)* dan dapat dijalankan pada berbagai Mesin *Virtual Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*),

dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi

2.5.1 Sejarah Java

Bahasa pemrograman Java terlahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh *Patrick Naughton, Mike Sheridan*, dan *James Gosling*, beserta sembilan pemrogram lainnya dari *Sun Microsystems.* Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh *Joe Palrang*.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 *Hamilton Avenue, Palo Alto*.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browse*r) *Mosaic* sebagai landasan awal untuk membuat perambah Java pertama yang dinamai *Web Runner*, terinsipirasi dari film 1980-an, *Blade Runner*. Pada perkembangan rilis pertama, *Web Runner* berganti nama menjadi *Hot Java*.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, *Eric Schmidt* dan *George Paolini* dari *Sun Microsystems* bersama *Marc Andreessen*, membentuk *Netscape*.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", *James Gosling*. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

2.5.2 Kelebihan

Multiplatform. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / bytecode) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan di atas *operating system* Linux tetapi dijalankan dengan baik di atas *Microsoft Windows*. Platform yang didukung sampai saat ini adalah *Microsoft Windows, Linux, Mac OS* dan *Sun Solaris*.

Penyebabnya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan bytecode tersebut.

OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek)

Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.

Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.5.3 Kekurangan

Tulis sekali, jalankan di mana saja - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.

Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform.

Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/*direverse-engineer*.

Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.6 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.

2.7 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.[6]



Gambar 2.36 *Clarifai API*

2.7.1 Clarifai Models

Model adalah blok pemrosesan yang mengubah masukan (misalnya gambar) ke prediksi (misalnya konsep, kotak pembatas, dll.). Ketika Anda menjalankan sebuah input melalui sebuah model, akan mengembalikan prediksi yang dilihat model pada input.

Setiap model dilatih untuk mengenali rangkaian output yang unik, seperti konsep, wajah, dan warna. Misalnya, jika Anda menjalankan masukan Anda melalui model 'makanan', ia akan mengembalikan konsep yang diketahui oleh 'makanan',

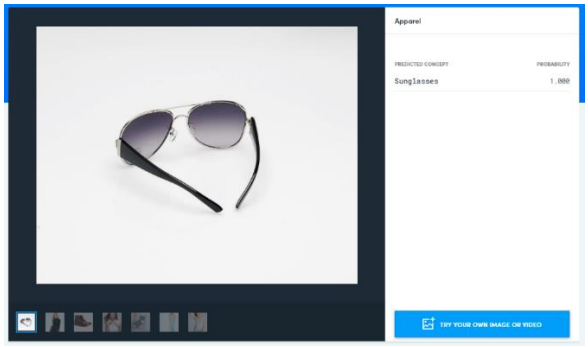
seperti nama hidangan tertentu. Jika Anda menjalankan input yang sama melalui model 'warna', itu akan mengembalikan prediksi tentang warna dominan pada gambar Anda.

Clarifai menyediakan beragam model pra-terlatih yang bisa Anda gunakan untuk membuat prediksi. Model ini dibangun di rumah dan telah diuji secara menyeluruh. Sebelum melatih model Anda sendiri, sebaiknya coba ini untuk melihat apakah sesuai dengan kebutuhan Anda

1. **Apparel Model (Beta)**

Model 'Apparel' mengenali lebih dari 100 konsep *fashion-related* termasuk pakaian dan aksesoris. Model ini sangat bagus untuk siapa saja yang membangun aplikasi yang mirip mode seperti platform e-niaga atau situs kurasi mode.

Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala

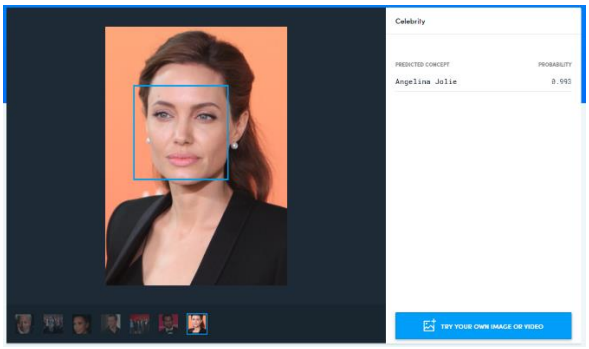


Gambar 2.37 ***Clarifai Apparel Model***

2. **Celebrity Model (Beta)**

Model 'Celebrity' menganalisis gambar dan mengembalikan nilai probabilitas pada kemungkinan media mengandung lebih dari 10.000 selebriti yang dikenali. Model ini sangat bagus bagi siapa saja yang membangun aplikasi yang mengandalkan perbandingan selebriti.

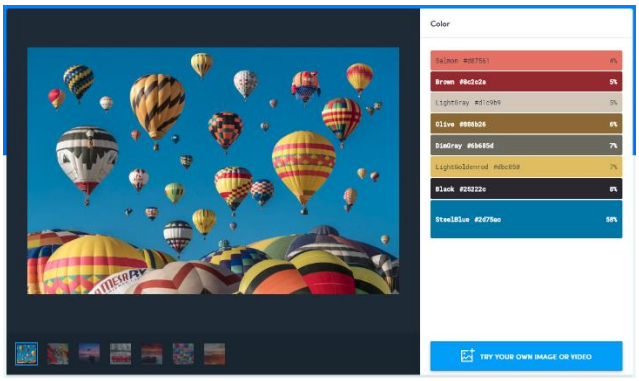
Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.



Gambar 2.38 *Clarifai Celebrity Model*

3. Color Model

Model 'Color' mengembalikan nilai kerapatan untuk warna dominan yang ada dalam gambar. Prediksi warna dikembalikan dalam format hex dan dipetakan ke rekan W3C terdekat mereka. Model ini sangat bagus untuk siapa pun yang membangun aplikasi yang warnanya penting untuk dhilangkan.

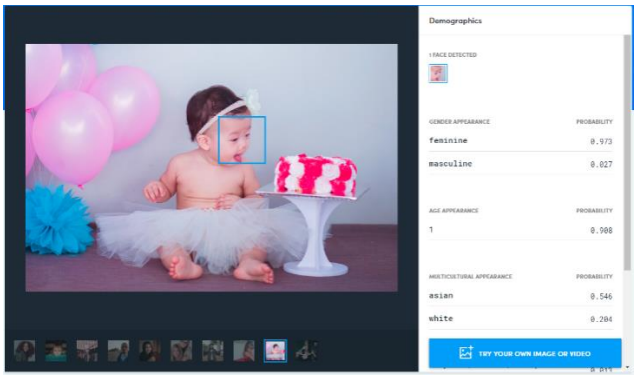


Gambar 2.39 *Clarifai Color Model*

4. Demographics Model (Beta)

Model 'Demografi' menganalisis gambar dan mengembalikan informasi tentang usia, jenis kelamin, dan penampilan multikultural untuk setiap wajah yang terdeteksi berdasarkan karakteristik wajah. Model ini sangat bagus bagi siapa pun yang membuat aplikasi yang mengandalkan informasi demografis untuk memahami pengguna.

Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.

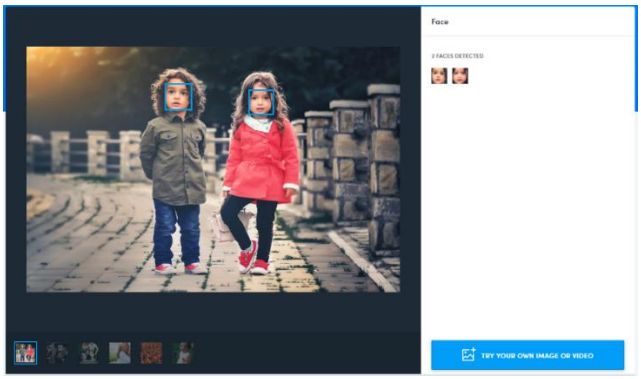


Gambar 2.40 ***Clarifai Demografi Model***

5. Face Detection Model (Beta)

Model 'Face Detection' mengembalikan nilai probabilitas pada kemungkinan gambar tersebut berisi wajah manusia dan mengkoordinasikan lokasi di mana wajah tersebut muncul dengan kotak pembatas. Model ini sangat bagus untuk siapa pun yang membangun aplikasi yang memantau atau mendeteksi aktivitas manusia.

Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.



Gambar 2.41 *Clarifai Face Model*

6. Face Emdedding Model (Beta)

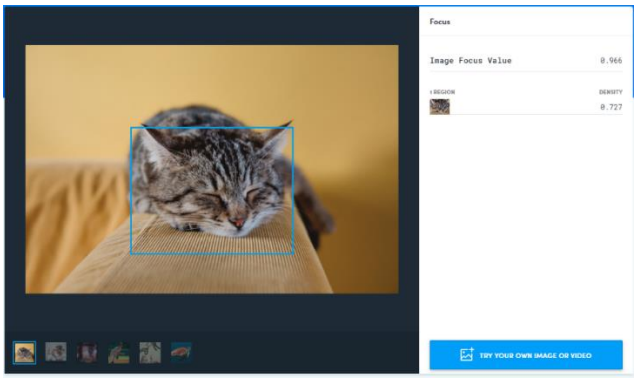
Model 'Face Embedding' menganalisis gambar dan mengembalikan vektor numerik yang mewakili setiap wajah yang terdeteksi pada gambar di ruang berdimensi 1024. Representasi vektor dihitung dengan menggunakan model 'Face Detection' Clarifai. Para vektor wajah yang serupa secara visual akan saling berdekatan di ruang berdimensi 1024. Model 'Face Embedding' dapat digunakan untuk mengatur, menyaring, dan merinci gambar sesuai dengan kemiripan visual.

Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.

7. Focus Model (Beta)

Model 'Fokus' menganalisis citra dan hasil kembali 1) nilai fokus keseluruhan (0-1) yang mewakili probabilitas bahwa ada daerah fokus dalam gambar, dan 2) kotak pembatas dan kepadatan fokus untuk setiap in- fokus dalam gambar. Model ini sangat bagus untuk siapa pun yang membangun aplikasi yang mengandalkan penyaringan gambar buram.

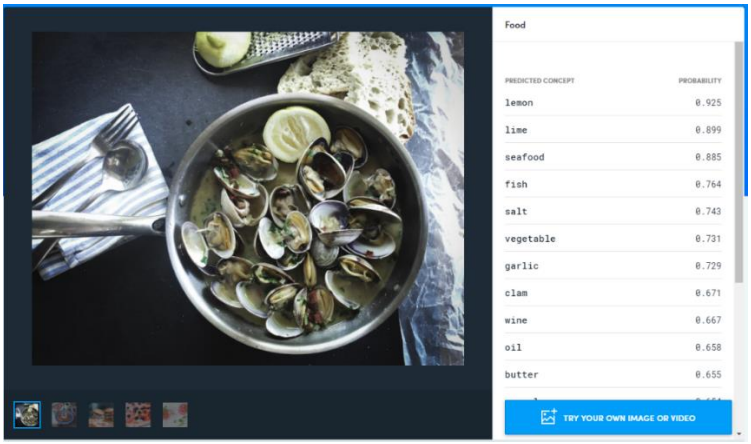
Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.



Gambar 2.42 *Clarifai Focus Model*

8. Food Model

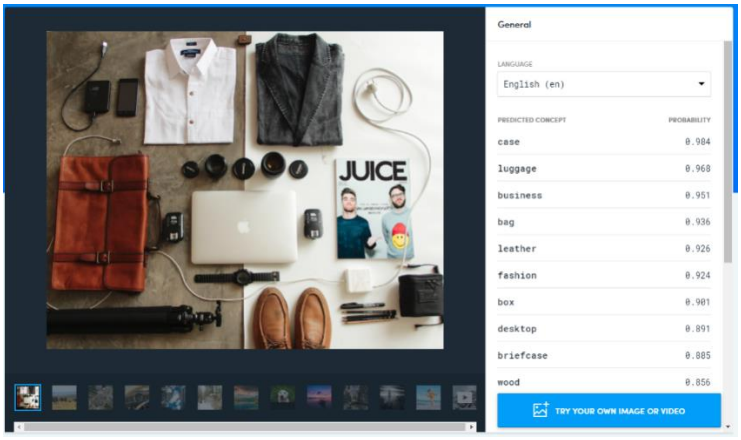
Model 'Makanan' mengenali lebih dari 1.000 item makanan dalam gambar sampai ke tingkat bahan. Model ini sangat bagus untuk membangun aplikasi kesehatan dan kesehatan.



Gambar 2.43 *Clarifai Food Model*

9. General Model

Model 'Umum' mengenali lebih dari 11.000 konsep yang berbeda termasuk objek, tema, suasana hati, dan banyak lagi. Model ini adalah solusi semua tujuan yang hebat untuk kebutuhan pengenalan visual yang paling banyak.



Gambar 2.44 *Clarifai General Model*

10. General Embedding Model (Beta)

Model 'General Embedding' menganalisis gambar dan mengembalikan vektor numerik yang mewakili gambar masukan dalam ruang berdimensi 1024. Representasi vektor dihitung dengan menggunakan model 'Umum' Clarifai. Vektor gambar yang serupa secara visual akan saling berdekatan di ruang berdimensi 1024. Model 'General Embedding' dapat digunakan untuk menyaring, mengindeks, memeringkat, dan mengatur gambar sesuai dengan kesamaan visual.

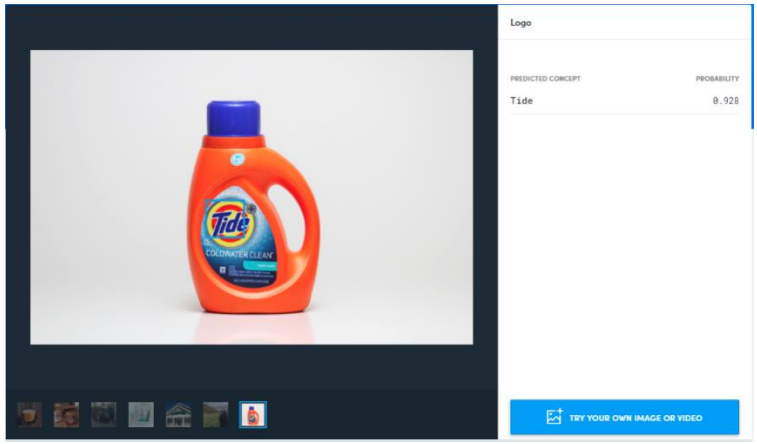
Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.

11. Logo Model (Beta)

Model 'Logo' menganalisis gambar dan mengembalikan nilai probabilitas pada kemungkinan media berisi lebih dari 500 nama merek yang diakui. Model ini sangat

bagus untuk siapa pun yang membangun aplikasi yang mengandalkan deteksi logo merek pada gambar.

Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala.

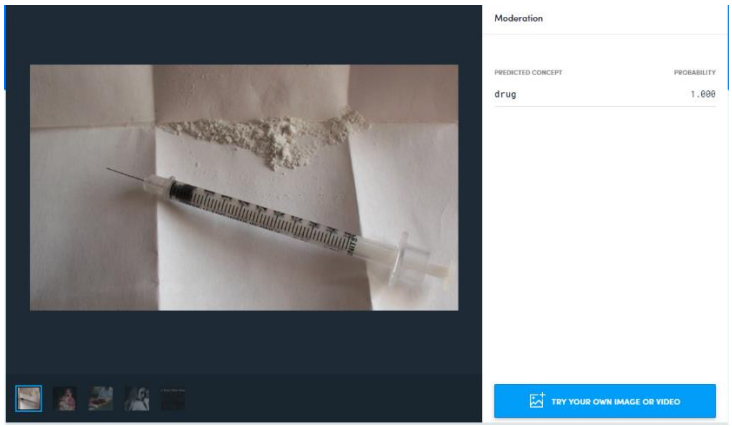


Gambar 2.45 ***Clarifai Logo Model***

12. Moderation Model (Beta)

Model ini mengembalikan nilai probabilitas pada kemungkinan bahwa gambar mengandung konsep seperti menanduk, obat-obatan, ketelanjangan eksplisit, atau ketelanjangan sugestif. Ini juga mengembalikan probabilitas bahwa gambar itu "aman", yang berarti tidak mengandung empat kategori moderasi tersebut di atas. Model ini sering digunakan untuk melindungi bisnis online dan komunitas dari risiko kepercayaan dan keamanan yang terkait dengan konten buatan pengguna.

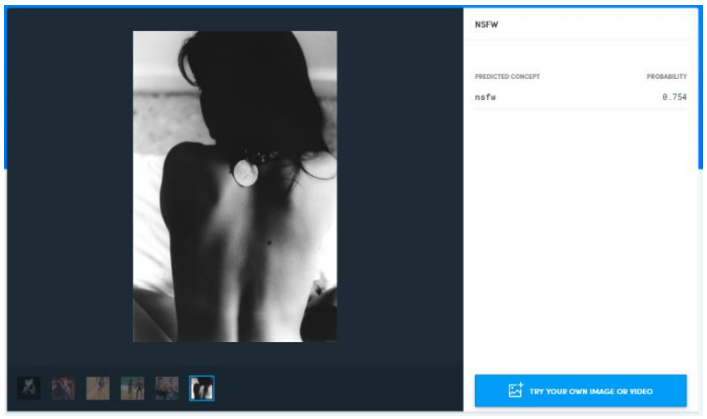
Harap dicatat bahwa model ini saat ini dalam pengujian BETA dan dapat diperbarui secara berkala



Gambar 2.46 *Clarifai Moderation Model*

13. NSFW Model

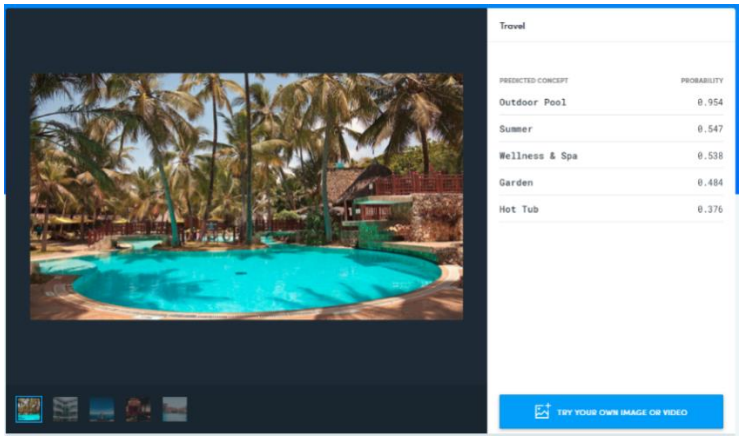
Model 'NSFW' (Tidak Aman untuk Pekerjaan) mengembalikan nilai probabilitas pada kemungkinan gambar itu berisi ketelanjangan. Model ini sangat bagus untuk siapa saja yang mencoba memoderasi atau menyaring konten yang menyinggung dari platform mereka secara moderat.



Gambar 2.47 *Clarifai NSFW Model*

14. Travel Model

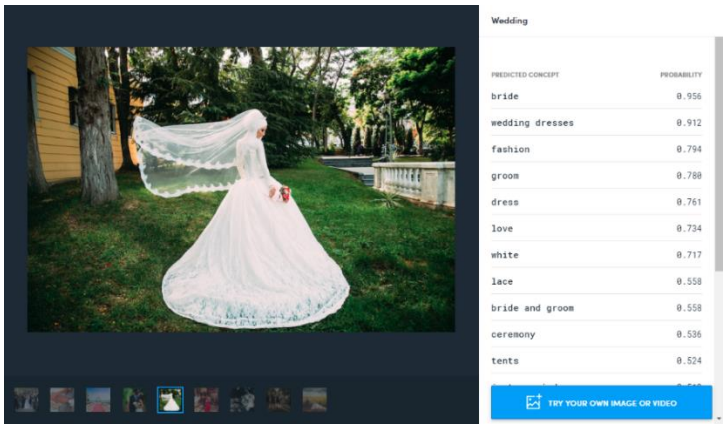
Model 'Travel' mengenali fitur spesifik properti residensial, hotel, dan perjalanan. Model ini sangat bagus untuk siapa pun yang membuat aplikasi perjalanan dan aplikasi perhotelan.



Gambar 2.48 ***Clarifai Travel Model***

15. Wedding Model

Model 'Wedding' mengenali lebih dari 400 konsep yang berkaitan dengan pernikahan termasuk pengantin, pengantin pria, pengantin pria, dan banyak lagi. Model ini sangat bagus untuk siapa pun yang membangun platform pernikahan atau mengurasi konten yang berkaitan dengan pernikahan.



Gambar 2.49 *Clarifai Wedding Model*

2.8 Database

Databse dapat dijelaskan sebagai kumpulan koleksi data yang terintegrasi. Data adalah sebuah representasi dari beberapa objek fisik. Database berisikan penjelasan tentang struktur yang memilikinya database saling terintegrasi dan memiliki hubungan antar data didalamnya (Berg, Seymour & Goel, 2013).

2.8.1 NoSQL Database Model

NoSQL (Not Only SQL) merupakan salah satu jenis dari database management system yang menggunakan kelompok data yang tidak bersifat non-relational dimana database tidak dibangun kedalam tabel-tabel dan tidak menggunakan SQL dalam melakukan manipulasi data. NOSQL database management system sangat berguna ketika bekerja dengan jumlah data yang sangat besar dan data-data didalamnya tidak memerlukan model relational (Moniruzzaman & Hossain,2013).

2.8.2 Klasifikasi NoSQL Database

Menurut (Arora & Aggarwal, 2013), Database NoSQL model dibagi menjadi 4 jenis yaitu :

- Key-Value Databse

  Penyimpanan data menggunakan *key value* dilakukan dalam sebuah form yang berisi key dan value
- Column Oriented Database

  Penyimpanan data berbasis kolom dilakukan dengan meyimpan data dalam sekumpulan kolom, kolom tersebut disimpan dalam sebuah file yang sama yang disebut dengan column families. Database jenis ini biasa dipakai untuk aplikasi yang secara intensif dipakai oleh user.
- Document Database

  Penyimpanan data dalam dokumen ini hampir mirip dengan key-value databse, namun data yang disimpan dalam dokumen database ini disimpan dalam format JSON atau XML. Database jenis ini digunakan untuk aplikasi yang data-data didalamnya sering mengalami perubahan seperti aplikasi Customer Relationship Management.
- Graph Database

  Database grafik menggunakan struktur grafik untuk menyimpan data. Databsae jenis ini cocok digunakan oleh aplikasi yang memiliki banyak interkoneksi antar data seperti media sosial.

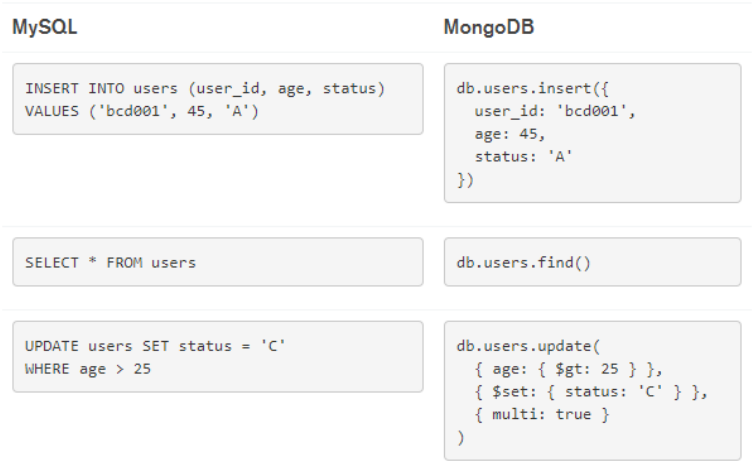
2.8.3 NoSQL Database Query

Penggunaan query adalah salah satu bagian yang dijabarkan tidak secara rinci pada database NoSQL. Salah satu kemungkinan dari proses query pada database NoSQL adalah dengan menggunakan perintah SQL yang sangat terbatas. Database NoSQL menggunakan operasi semacam get (key) untuk mendapatkan data dari key yang dipanggil, put (key,value) untuk melakukan proses insert atau update serta perintah delete (key) untuk menghapus key yang ada di database. Beberapa database NoSQL memerlukan perintah tambahan execute (key) untuk mejalankan perintah-perintah diatas.

Distribusi data secara horizontal pada database NoSQL menyebabkan tidak didukungnya perintah join atau order by seperti yang ada pada database relational. Apabila diperlukan, maka operasi join maupun order by dapat dilakukan dari sisi client. Operasi seleksi pada database NoSQL sering dijelaskan pada level API (Pokorny,2013).

2.8.4 MongoDB

MongoDB adalah salah satu jenis database dokumen yang memberikan kinerja dan ketersediaan tinggi serta skalabilitas yang mudah. Dalam penggunaanya, MonggoDB menggunakan beberpa database. Satu database memegang sekumpulan data yang disebut sebagai collections. Sebuah collection memegang sekumpulan dokumen. Sebuah dokumen berisikan kumpulan pasang key dan value. Dokumen memiliki skema yang dinamis yang berarti bahwa dokumen-dokumen yang berada pada collections yang sama tidak perlu memiliki kolom dan struktur yang sama. Kolom data dalam setiap dokumen yang ada pada collections bisa memiliki tipe data berbeda (MongoDB).[7]



Gambar 2.50 *Query MongoDB*

2.9 Web Service

Web service menurut W3.org mendefinisikan web service sebagai "sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefiniskan, dideskripsikan, dan dimengerti oleh XML dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML melalui protokol internet".

Web service adalah sebuah sofware aplikasi yang tidak terpengaruh oleh platform, ia akan menyediakan method-method yang dapat diakses oleh network. Ia juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entities bisnis yang berbeda.

Selain itu ada definisi lain tentang Web Service, Web service adalah sistem software yang dirancang untuk mendukung interopabilitas mesin-ke-mesin yang dapat berinteraksi melalui jaringan. Web service memiliki antarmuka yang dijelaskan dalam format mesin-processable (khusus WSDL). Sistem lain berinteraksi dengan Web service dalam cara ditentukan oleh deskripsi dengan menggunakan pesan SOAP, biasanya disampaikan menggunakan HTTP dengan serialisasi XML dalam hubungannya dengan Web lainnya yang terkait standar.

Dalam pengertian yang sederhana , XML Web Services dapat di definisikan sebagai aplikasi yang diakses oleh aplikasi yang lain. Mungkin orang berpendapat itu semacam web site, tetapi itu bukan demikian. Ada perbedaan – perbedaan yang membedakan dengan web site.

Perbedaan tersebut dapat dilihat dibawah ini :

Website

1) Memiliki web interface
2) Dibuat untuk ber interaksi langsung dengan user
3) Dibuat untuk bekerja pada web browser.

Web Services

1) Tidak memiliki interface yang bagus
2) Dibuat untuk ber interaksi langsung dengan applikasi yang lain baik beda OS / Konsep sekalipun.
3) Dibuat untuk bekerja pada semua tipe client applikasi / perangkat device

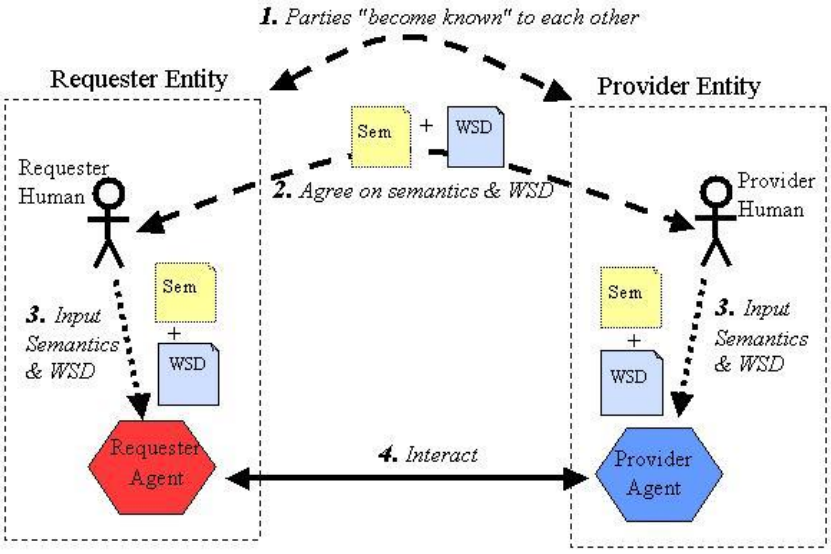
Beberapa karakteristik dari Web Service adalah :

– Message-based

– Standards-based

– Programming language independent

– Platform-neutral



Gambar 2.51 Proses umum *Web Service*

2.10 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.[8]

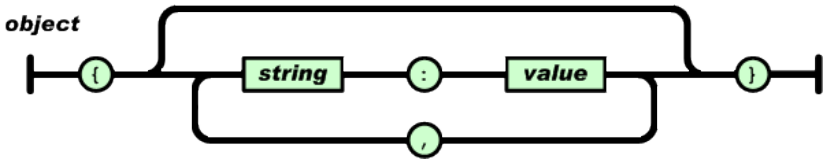
JSON terbuat dari dua struktur:

- 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.

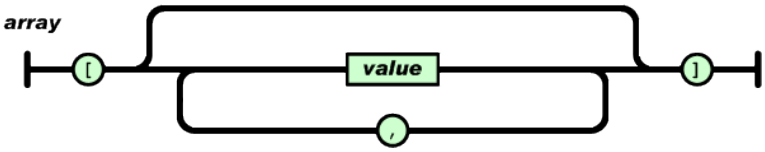
JSON menggunakan bentuk sebagai berikut:

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).



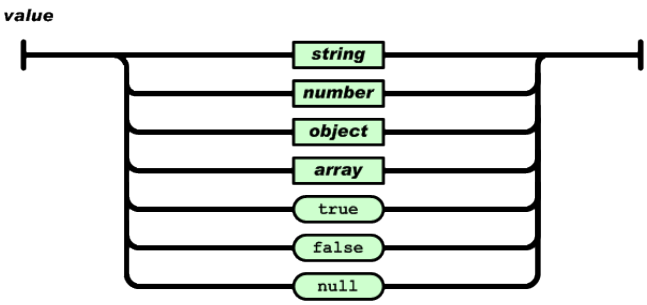
Gambar 2.52 ***Objek JSON***

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



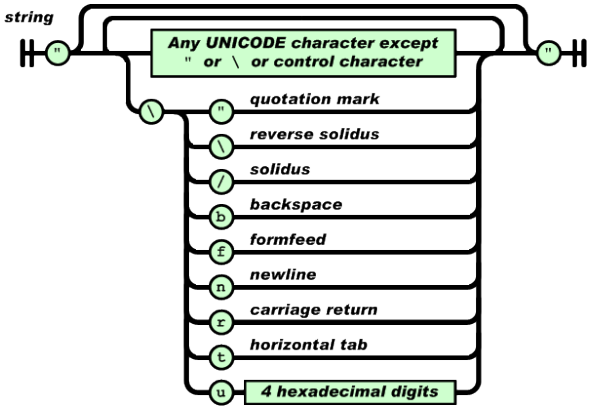
Gambar 2.53 ***Array JSON***

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



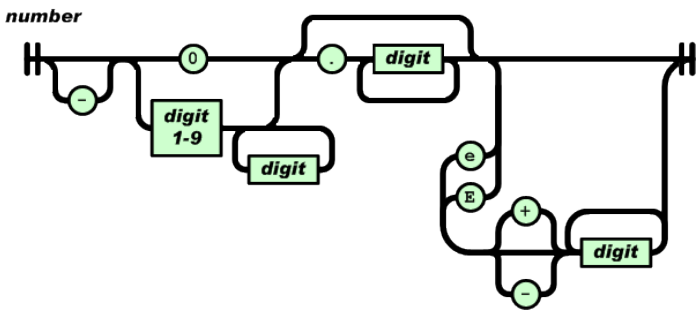
Gambar 2.54 ***Value JSON***

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 2.55 ***String JSON***

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



Gambar 2.56 ***Number JSON***

Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan. [17]

2.11 Object 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.

2.12 Unified Modelling Language

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

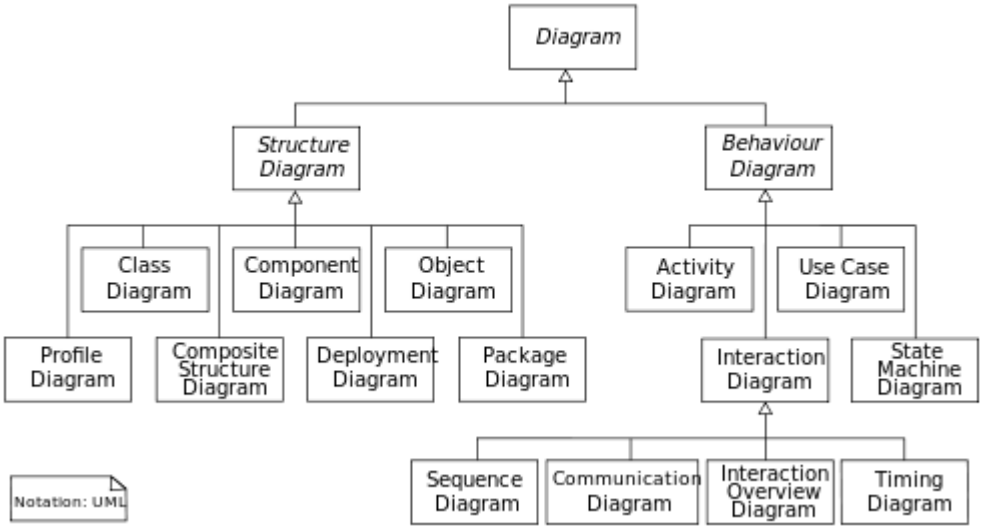
Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi

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.

2.12.1 Diagram UML



Gambar 2.57 Diagram UML

Berikut penjelasan singkat dari pembagian kategori tersebut adalah :

- 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.

2.12.2 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.[20]

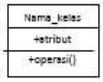
- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

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

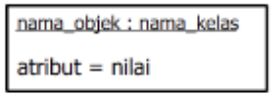
Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Antar muka / *interface* 	Sama dengan konsep *interface* dalam pemprograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*
Asosiaso berarah/ *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai *multiplicity*
generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan/ *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/ *aggregation* 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

Gambar 2.58 Simbol *Class Diagram*

2.12.3 Object 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:

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

Gambar 2.59 Simbol Diagram Objek

2.12.4 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.[20]

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

- 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.

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Gambar 2.60 Simbol *Use Case Diagram*

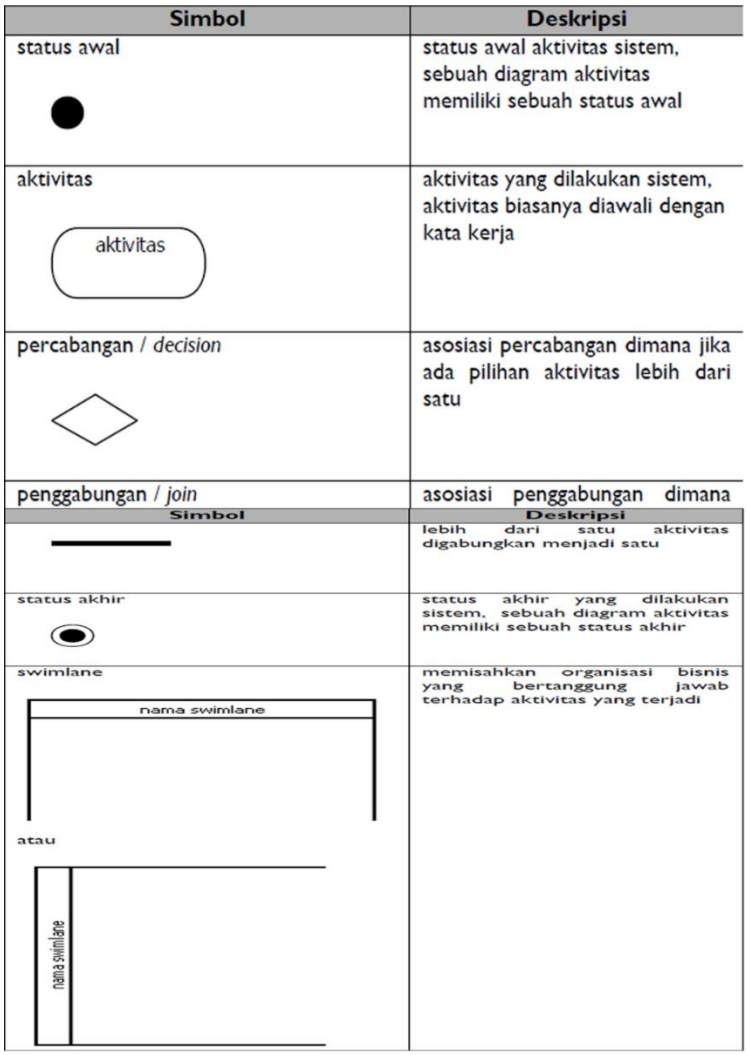
2.12.5 Activity Diagram

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.

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 2.61 Simbol Diagram Aktifitas

2.12.6 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 objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Berikut adalah simbol-simbol yang ada pada sequence diagram:



Gambar 2.62 Simbol *Sequence Diagram*