BAB II
LANDASAN TEORI

2.2. Landasan Teori

Landasan teori menjelaskan beberapa teori-teori dan penjelasan yang berkaitan dengan aplikasi mobile android yang akan dibangun. Teori – teori yang terkait dengan pembangunan aplikasi akan di jelasakan pada bab ini.

2.2.5. Hewan Peliharaan

Hewan peliharaan adalah hewan yang hidup secara dalam lingkungan tertentu, tidak bebas, mereka tumbuh, bergerak, mencari makan dan berkembang biak dengan bantuan manusia secara langsung maupun tidak langsung. Hewan peliharaan adalah hewan yang kehidupannya untuk sebagian atau seluruhnya bergantung pada manusia untuk maksud tertentu.Yang artinya adalah manusia selaku pemilik atau pengelola hewan pada prinsipnya bertanggungjawab penuh atas kesejahteraan hewan tersebut.

Hewan peliharaan sering di anggap sebagai teman atau sahabat oleh pemiliknya, oleh karena itu pemilik hewan selalu memberikan kasih sayang, merawat, dan menjaga hewan peliharaannya. Pemilik hewan di kota Bandung untuk saat ini berjumlah 1.268 anggota sesuai dengan jumlah anggota pada Komunitas Pecinta Kucing Bandung dan 310 anggota pada Komunitas Pecinta Anjing Ras Bandung Selatan.

2.2.6. Pet Shop

Pet Shop atau toko hewan peliharaan adalah bisnis ritel yang menjual berbagai macam kebutuhan hewan peliharaan, mulai dari menjual makanan hewan, kebutuhan perawatan, dan aksesoris hewan peliharaan. Beberapa toko hewan peliharaan juga menjual berbagai jenis hewan peliharaan bagi yang ingin memelihara hewan peliharaan.

2.2.7. Pemacakan Hewan

Pemancakan atau dapat disebut dengan mengawinkan adalah suatu proses pembiakan hewan guna memperoleh keturunan yang diinginkan. Pemacakan dari hewan ras yang memiliki bukti dari silsilah ras murni memberikan suatu jaminan kepada para pemacak bahwa keturunan dari hewan peliharaan yang mereka kawinkan nanti jelas apa rasnya. Pedigree juga menjelaskan bahwa hewan tersebut memiliki kesehatan yang terjamin karena setiap perubahan fisik ataupun kesehatan akan dicatatkan juga.

2.2.8. Konsultasi Medis

Konsultasi medis adalah pertemuan di mana seorang dokter berbicara dengan seseorang tentang masalah, dan pertanyaan mengenai suatu penyakit. Tujuan dari konsultasi ini adalah menjalankan tindakan pencegahan untuk menghentikan berkembangnya berbagai macam penyakit bagi pasien yang memiliki faktor resiko. Hal ini dapat dilakukan dengan berbagai cara. Antara lain, memperoleh diagnosis bagi gejala-gejala yang dialami pasien atau jika pasien rutin melakukan pemeriksaan kesehatan tahunan, dokter dapat meninjau kembali kemungkinan pasien mengidap suatu penyakit.

2.2.9. Rekomendasi

Rekomendasi adalah saran yang sifatnya menganjurkan, membenarkan, atau menguatkan mengenai sesuatu atau seseorang. Rekomendasi sangat penting artinya untuk meyakinkan orang lain bahwa sesuatu bahwa seseorang tepat dan layak. Misalnya ketika seseorang akan menggunakan jasa online shop biasanya akan terlihat sebuah testimoni dari orang – orang yang sudah pernah bertransaksi sebelumnya, apakah banyak yang merekomendasikan atau tidak. Jika banyak testimoni positif maka akan menambah keyakinan seseorang untuk bertransaksi.

Rekomendasi biasanya dibuat dalam bentuk tertulis seperti review produk dan testimoni dari internet atau dalam bentuk yang lebih formal dalam bentuk surat rekomendasi.

2.2.10. Vaksin

Tujuan vaksinasi adalah untuk memberikan kekebalan (antibodi) pada hewan sehingga dapat melawan antigen atau mikroorganisme penyebab penyakit. Pemberian kekebalan tubuh dengan vaksin adalah bentuk perlindungan yang sebaik-baiknya untuk hewan.

2.3. Konsep Pemrograman Mobile

2.3.5. Aplikasi Mobile

Menurut Irwansyah & V.Moniaga dalam bukunya yang berjudul "Pengantar Teknologi Informasi", pengertian dari mobile applications adalah aplikasi perangkat lunak yang dibuat khusus untuk dijalankan di dalam tablet dan juga smartphone. Umumnya, developer mobile apps memerlukan IDE atau Intergrated Development Environments dan juga SDK untuk pengembangan dari mobile apps itu sendiri. Pada saat ini, pada smartphone dan juga tablet, ada satu aplikasi yang berguna untuk menyediakan berbagai macam aplikasi yang dapat dijalankan di device tersebut. Aplikasi ini sering disebut store. Contoh store yaitu apple apps store, samsung apps, amazon kindlefire, windows store dan google playstore [8].

Gambar 3.9 Store yang tersedia saat ini
(sumber:http://appshopper.com/http://www.brighthub.com/http://cliveric
h.com/http://www.geekwire.com/

Gambar 2.1 *Store Mobile*

Jika membahas tentang Mobile Apps, umumnya kita tidak bisa tidak menyinggung soal Developer. Developer ialan badan usaha yang membuat aplikasi yang nantinya akan dijalankan dalam device. Pada dasarnya, aplikasi akan berjalan menggunakan tenaga baterai dan juga mendapat dukungan dari prosesor yang

ukurannya lebih kecil dibanding dengan prosesor komputer. Sebelum dilempar ke pasar, umumnya mobile apps akan dites terlebih dahulu menggunakan emulator. Emulator merupakan salah satu cara untuk menghemat biaya yang dikeluarkan oleh developer untuk membuat mobile apps(Irwansyah & V.Moniaga:61-62) [8].

2.3.6. Teknologi GPS

Menurut Antonius Aditya Hartanto dalam bukunya yang berjudul "Mengenal Aspek Teknik dan Bisnis LBS" Bahwa Global Positioning System atau sering disingkat dengan GPS adalah sistem navigasi yang menggunakan satelit yang didesain agar dapat menyediakan posisi secara instan, kecepatan dan informasi waktu di hampir semua tempat dimuka bumi, setiap saat dan dalam kondisi cuaca apapun. Pada dasaranya, GPS merupakan aplikasi yang harus menunggu terlebih dahulu permintaan dari pengguna. Aplikasi ini menyediakan akurasi positioning atau penentuan posisi yang berkisar antara 100 meter(95% dari waktu), hingga 5 sampai 10 meter, juga sampai akurasi relatif pada submeter, dan bahkan tingkat subcentimeter. Secara umum, semakin tinggi akurasi yang dihasilkan akan memerlukan infrastruktur yang lebih canggih dan tentunya berhubungan dengan biaya yang harus dikeluarkan.

Pengguna GPS untuk penentuan posisi saat ini diantaranya adalah navigasi untuk kegiatan pribadi (hiking, pelayaramn, berburu, petunjuk ketika mengemudi dan lain sebagainya), navigasi di pesawat, survei di lepas pantai dan navigasi kapal, fleet tracking, pengendalian mesin, teknik sipil, survey daratan, GIS dan pemetaan, analisis deformasi dan sebagainya.

Program GPS dan operasionalnya saat ini sebagian besar bersumber pada Department of Defense (DoD). Amerika serikat yang dapat dikatakan merupakan pembuat sistem ini. Manajmennya sendiri dilaksanakan oleh US Air Force dengan panduan dari komite eksekutif DoD Positioning/Navigation. Komite ini menerima masukan dari komite yang sama dari Department of Transportation (DoT) yang bertindak sebagai suara sipil untuk urusan atauran GPS(NAPA, 1995). GPS asli hasil desain oleh US Department of Defense (DoD) terdiri atas tiga komponen utama, yaitu control segment, space segment, dan user segment.

1) GPS Control Segment

Control segment terdiri atas lima stasiun yang terletak di pangkalan Falcon Air Force, Colorado Springs, Hawaii, Ascension Island, Diego Garcia dan Kwajalein. Stasiun-stasiun ini adalah mata dan telinga bagi GPS, bertugas memonitor satelit-satelit tersebut sebagaimana mereka mengirimkan data overhead dengan mengukr jarak antarsatelit setiap 1.5 detik (Hofmann-Wellenhof, 1992). Data ini kemudian diperhalus dengan menggunakan informasi ionospheric dan meteorologi dan dikirikan ke Master Control Station yang berada di fasilitas US Air Force Space Command yang ada di Colorado Spring. Disinilah parameter-parameternya baru dapat menggambarkan perkiraaan dari orbit satelit dan clock performance. Informasi ini selanjutnya akan dikembalikan ketiga buah uplink station (ko-lokasi di stasiun pemonitor Ascension Island, Diego Garcia dan Kwajalein) yang akan mengirimkan informasi tersebut ke satelit. Dengan mengacu pada luasnya daerah penyebaran dari control station, semua satelit GPS di-tracking selama 92% dari waktu.

2) GPS Space Segment

Space segment terdiri atas sebuah jaringan satelit dalam orbit lingkaran yang terdekat dengan tinggi nominar sekitar 20,183 km di atas permukaan bumi serta dengan periode selama 12 jam. Konstelasi yang sesungguhnya adalah untuk 24 satelit, dalam 3 ruang orbit dan condong ke equator(Spilker, 1980). Akan tetapi, skenario ini telah mulai berubah dan satelit-satelit saat ini ditempatkan dalam enam ruang orbit yang berbeda, dengan emapt satelit di masing-masing ruang.

3) GPS User Segment

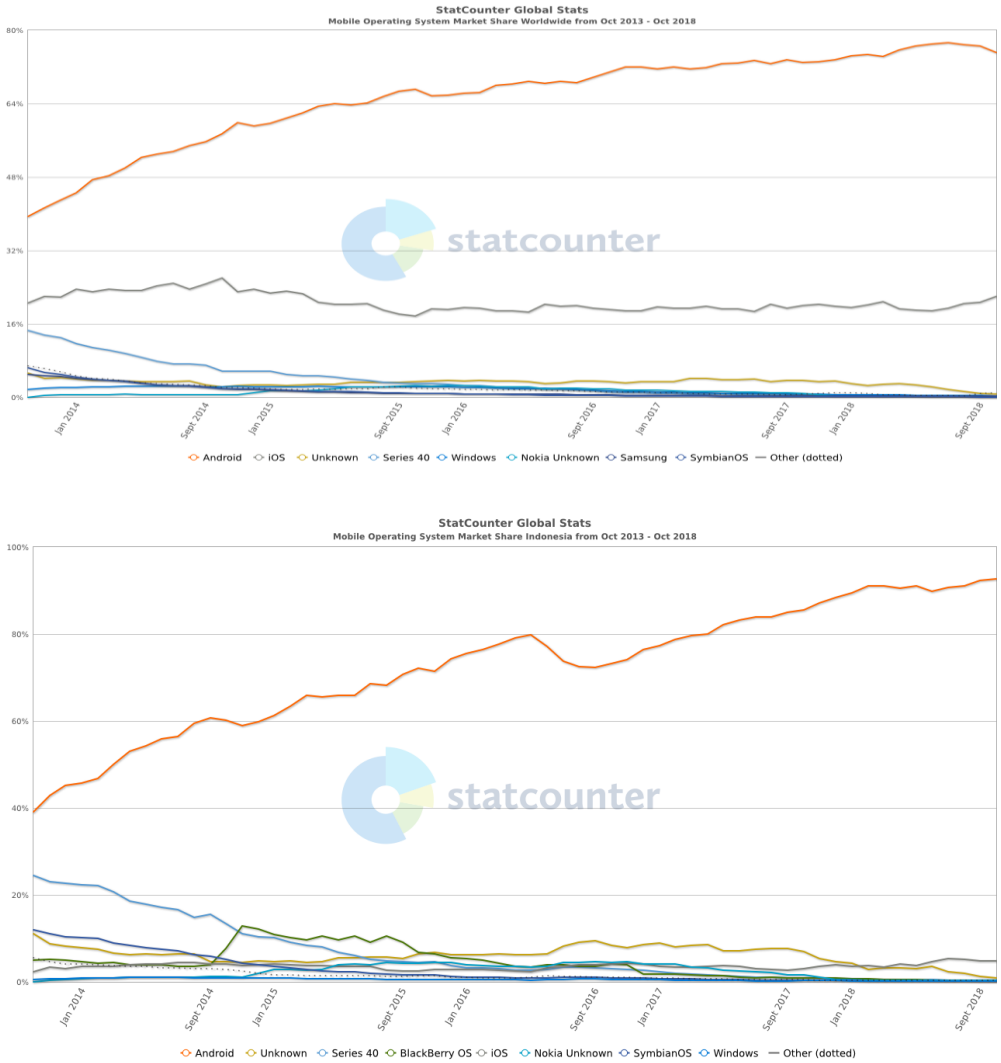
USER segement teridiri atas antenna dan prosessor receiver yang menyediakan positiong, kecepatan dan ketepatan waktu ke pengguna. Bagian ini menerima data dari satelit-satelit melalui sinyal radio yang dikirim-kan setelah mengalami koreksi oleh stasiun pengendali di daratan.

2.3 Android

Gambar 2.2 Logo Android

Menurut Fachrul K & Gianto W dalam bukunya berjudul "Cepat Menguasi Pemrograman Android" bahwa, Android adalah salah satu sistem operasi yang pada awalnya, kemudian berkembang menjadi bahasa pemrograman yang banyak dicari dan digunakan oleh programmer. Pada dasarnya android adalah sistem operasi yang berbasis linux. Pengguna android pada awalna hanya digunakan untuk melengkapi sistem operasi pada *gadget-gadget* seluler seperti smartphone yang menggunakan layar sentuh. Tetapi karena sistem yang dikembangkan open source, mau tidak mau perkembangan dan penerimaan didunia industri IT menjadi lebih cepat juga [9].

Perusahaan android dibawah bendera android.inc, adalah pengembang pertama kali sistem operasi ini. Android.inc, pertama kali berdiri dikota Alto, salah satu kota erkenal di California Amerika Serikat, tepatnya pada bulan Oktober tahun 2003. Pendirinya terdiri dari tiga orang yang ahli dalam bidangn pengembangan aplikasi, mereka adalah Andy Rubin, Rich Miner, dan Chris White. Kemudian sejak tahun 2005, Google membeli dan selanjutnya di kembangkan oleh sumber daya Google sendiri sehingga mudah digunakan oleh khayalayak ramai. Dan sejat tahun 2007 secara resmi Google meluncurkan android sebagai sebuah sistem operasi baru khususnya untuk digunakan pada *smartphone* atau *gadget*, dan sistem operasi ini bersifat *open source* alias tidak diperjualbelikan. Sejak saat itu android bisa berkembang sedemikian rupa dan bisa menghasilkan berbagai macam aplikasi yang dibutuhkan oleh masyarakat untuk membantu kehidupan sehari-hari [9].



. **Gambar 2.3 Statistik Pengguna OS Mobile di Indonesia**

2.3.1 Android SDK (Software Development Kit)

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java.

2.3.2 ADT (Adnroid Development Tools)

Android Develpoment Tools adalah plugin yang di desain untuk IDE Eclipse yang memberikan kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse akan memudahkan dalam membuat aplikasi *project* Android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya.

2.3.3 Android Life Cycle

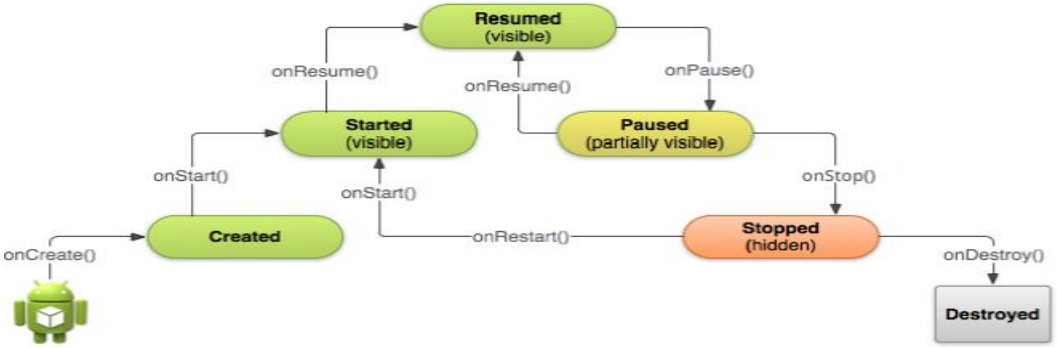
Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada , klasifikasi tersebut berdasarkan kelas- kelas dasar java yang digunakan.

Tabel 2.1 Komponen Android

Functionality	Java Base Class	Examples
Focused thing a user can do	Activity	Edit a note, play a game
Background process	Service	Play music, update weather icon
Receive messages	BroadcastReceiver	Trigger alarm upon event
Store and retrieve data	ContentProvider	Open a phone contact

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan specified permission sebelum digunakan seperti komponen Service, BroadcastReceiver. ContentProvider.

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem android menjalankan kode dalam method Activity dengan menerapkan metode callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web. Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna.



Gambar 2.4 Siklus Hidup Android

Terdapat beberapa state dalam siklus hidup android yang terjadi seperti diilustrasikan pada Gambar 2.4 Siklus Hidup Android ,akan tetapi hanya beberapa dari state tersebut yang menjadi statis diantaranya :

1. Resumed

   Resumed terjadi ketika aplikasi berjalan setelah state paused . State ini akan menjalankan perintah program yang ditulis pada method onResume() (Google Inc,2014).

2. Paused

   Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini (Google Inc, 2014).

3. Stopped

   Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihatoleh pengguna tetapi masih meninggalkan service dibackground (Google Inc, 2014). State lain seperti Created dan Started bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode life cycle callback berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), krmudian diikuti oleh onResume() (Schwarz, Dutson, Stelee, & To, 2013).

2.3.4 Arsitektur Android

Secara garis besar Arsitektur Android dapat di jelaskan dan di gambarkan sebagai berikut :

1) *Application* dan *Widgets*

Application dan *Widgets* adalah *layer* dimana *user* berhubungan dengan aplikasi saja, dimana biasanya user men-*download* aplikasi, melakukan instalasi dan menjalankan aplikasi.

2) *Application Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif.

3) *Libraries*

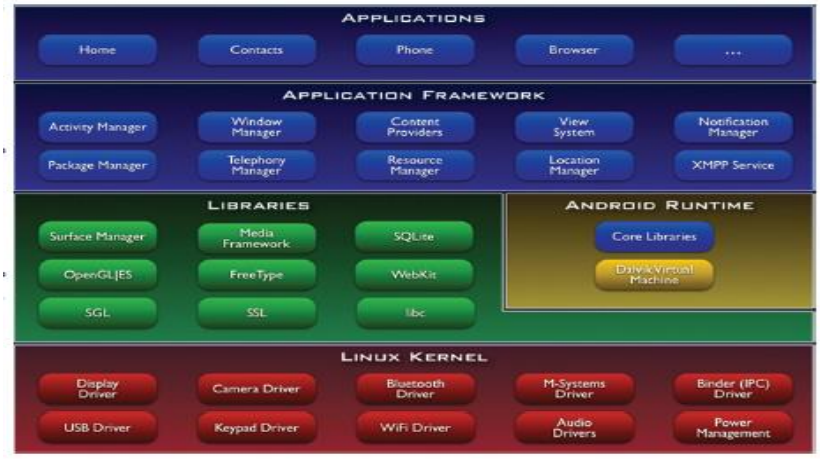
Libraries adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

4) Android *Runtime*

Layer yang membuat aplikasi Android dapat djalankan dimana dalam prosesnya menggunakan Implementasi Linux.

5) Linux Kernel

Linux Kernel adalah *layer* dimana inti dari *operating system* dari Android itu berada.



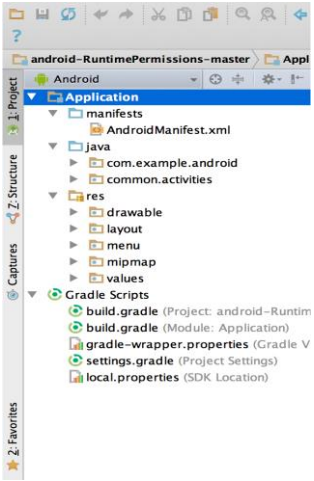
Gambar 2.5 Arsitektur Android

2.4 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. **Stuktur Proyek pada Android Studio**



Gambar 2.6 Tampilan File Struktur Android Studio

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

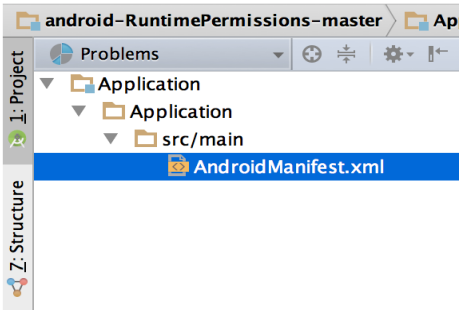
Secara default, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 1. Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke file sumber utama proyek Anda.

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.

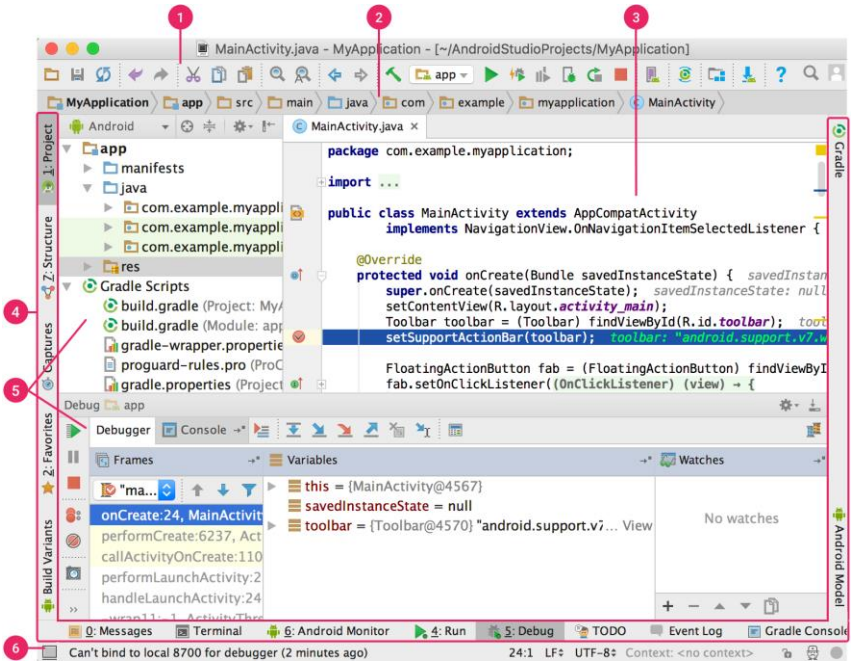
Struktur proyek Android pada disk berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih Project dari menu tarik turun Project (dalam gambar 1, struktur ditampilkan sebagai Android).

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 2.7 Tampilan File Proyek Android Studio

2. **Antarmuka Pengguna pada Android Studio**



(sumber : https://www. developer.android.com/studio/intro/ 05-Oktober-2018).

Gambar 2.8 Tampilan 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.

2.4.1 Global Positioning System (GPS)

GPS merupakan sebuah infrastruktur satelit yang melayani penempatan posisi dari berbagai macam objek(Speikerman 2004). GPS pertama kali digunakan untuk kepentingan militer, tetapi pada tahun 1980-an pemerintah Amerikat Serikat memutuskan untuk membuat sistem *positioning* secara bebas dan dan tersedia untuk berbagai macam industry di dunia. Menurut (Roth 2004) sistem GPS terdiri atas 3 segmen yaitu [10].

1. ***User Segment*** yang terdiri atas perangkat bergerak dari pengguna (GPS *receivers).*
2. ***Space Segment*** yang terdiri atas satelit. Setiap satelit mempunyai berat anatra 1.5 sampai 2 ton dan mempunyai energy yang swatantra yang disuplai oleh sel matahari.
3. ***Control segment*** administrasi yang dibutuhkan oleh satelit sebagai koreksi dari internet data satelit (sistem waktu dan orbit).

GPS dapat melakukan perhitungan dan menentukan posisi user dan menampilkan dalam maps. Jika sudah dapat menyimpan posisi user selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah tuju, rute, tujuan perjalanan serta jarak tujuan. GPS ini juga dimanfaatkan untuk membangun sebuah aplikasi pencarian pet shop atau klinik hewan dimana si pengguna menetukan lokasi untuk mengetahui posisi pengguna dan lokasi pet shop atau klinik hewan.

2.4.1.1 Akurasi GPS

Posisi yang ditunjukkan oleh suatu GPS mempunyai faktor kesalahan atau juga disebut tingkat akurasi. Sebagai contoh suatu alat GPS menunjukkan titik koordinat dengan tingkat akurasi 5 meter, itu berarti posisi pengguna bisa berada dalam range radius 5 meter dari titik yang ditunjukkan tersebut. Mengapa tingkat akurasi yang terlihat bisa berubah-ubah? Kadang terlihat 10 meter, 15 meter, atau 5 meter. Ada beberapa hal yang mempengaruhi tingkat akurasi tersebut, antara lain:

1. Kesalahan Ephemeris. Terjadi jika satelit tidak dapat mentransmisikan posisinya di orbit dengan tepat.
2. Keadaan Ionosphere. Ionosphere berada pada jarak sekitar 43-50 mil di atas permukaan bumi.Walaupun GPS receiver berusaha untuk

mengkoreksi/memperbaiki faktor keterlambatan yang terjadi tetap saja aktivitas tertentu dari plasma bisa menyebabkan kesalahan perhitungan.

3. Keadaan Troposphere. Troposphere adalah bagian terendah dari atmosfer sampai dengan ketinggian sekitar 11 mil dari permukaan tanah. Variasi pada temperatur, tekanan, dan kelembaban bisa menyebabkan perbedaan kecepatan penerimaan gelombang radio.
4. Kesalahan Waktu. Karena penempatan jam atom pada setiap GPS receiver tidak berjalan sebagaimana mestinya. Kesalahan waktu dari GPS receiver yang tidak presisi dapat menimbulkan ketidakakurasian.
5. Kesalahan Multipath. Terjadi karena sinyal satelit membentur permukaan keras (seperti bangunan atau tebing) sebelum mencapai GPS receiver. Hal tersebut bisa menyebabkan terjadinya delay sehingga perhitungan jarak menjadi tidak akurat.
6. Buruknya Sinyal Satelit. Keadaan langit yang terhalang akan menyebabkan GPS sulit menerima data satelit. Sebuah sinyal satelit yang pada hari tertentu diterima dengan sangat bagus belum tentu pada hari lain bisa diterima dengan kualitas yang sama walaupun user berdiri pada tempat yang sama.

2.4.1.2 Android dan GPS

Dewasa ini, teknologi berkembang dengan pesat. Dulu ponsel hanya sekedar digunakan untuk menelpon dan SMS saja. Sekarang ponsel sudah menjelma menjadi kotak kecil ajaib yang serba bisa. Salah satunya adalah ponsel dengan sistem operasi Android. Dengan standarisasi fitur dan hardware yang dimiliki, menjadikan ponsel Android ponsel canggih yang disukai banyak orang. Tidak lagi canggih karena adanya fitur MMS, radio, atau internet berkecepatan tinggi tapi juga karena ditanamkannya fitur teknologi satelit di dalamnya. Ya, perangkat GPS *receiver* yang dulu besar dan eksklusif, sekarang sudah bisa dimiliki dengan “hanya” membeli sebuah ponsel.

Dengan ponsel berteknologi satelit (GPS), banyak hal yang bisa dilakukan. Ingin melihat di mana posisi user sekarang dalam sebuah peta? Mengambil foto/video yang sudah dilengkapi dengan data koordinat? Ingin tahu jalur olahraga bersepeda yang sudah pernah user lalui? Tidak hanya itu, user juga dapat pergi ke

tempat wisata tertentu dengan dipandu gambar dan suara dari sebuah ponsel! Bahkan lebih jauh lagi, GPS dapat digunakan untuk membantu memberikan peringatan dini terhadap terjadinya bencana alam. Sekarang ini banyak sekali pengembang - pengembang aplikasi untuk sistem operasi Android termasuk aplikasi-aplikasi GPS. Yang menyenangkan aplikasi-aplikasi tersebut jenisnya beragam dan jumlahnya pun banyak.

2.4.2 Location Basesd Services

LBS adalah layanan yang menyediakan informasi mengenai suatu tempat, dapat diakses dengan perangkat *mobile* melalui jaringan selular dan memiliki kemampuan untuk menggunakan posisi geografis dari perangkat *mobile* [11].

Location Based Service (LBS) atau Layanan Berbasis lokasi adalah layanan informasi yang dapat diakses melalui mobile device dengan menggunakan mobile network yang dilengkapi kemampuan untuk memanfaatkan lokasi dari mobile device tersebut. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Oleh karena itu pengguna memberitahu penyedia layanan untuk memberi informasi,dengan referensi posisi pengguna tersebut. Layanan berbasis lokasi dapat digambarkan sebagaisuatu layanan yang berada pada pertemuan tiga teknologi yaitu: Geographic Information System, Internet Service, dan Mobile Devices, hal ini dapat dilihat pada gambar LBS adalah pertemuan dari tiga teknologi. Secara Garis besar jenis Layanan Berbasis Lokasi juga dapat dibagi menjadi dua, yaitu:

1. ***Pull Service***: Layanan diberikan berdasarkan permintaan dari pelanggan akan kebutuhan suatu informasi. Jenis layanan ini dapat dianalogikan seperti menggakses suatu web pada jaringan internet.
2. ***Push Service***: Layanan ini diberikan langsung oleh sevice provider tanpa menunggu permintaan dari pelanggan, tentu saja informasi yang diberikan tetap berkaitan dengan kebutuhan pelanggan.

   Teknologi ini dipakai dalam pembangunan aplikasi.

2.4.3 Firebase Cloud Messaging

Gambar 2.9 Logo Firebase

Firebase Cloud Messaging (FCM) adalah solusi perpesanan lintas-platform yang memungkinkan Anda mengirimkan pesan dan pemberitahuan dengan terpercaya tanpa biaya. Dengan menggunakan FCM, Pengguna bisa memberi tahu aplikasi klien bahwa email baru atau data lainnya tersedia untuk disinkronkan. Pengguna mengirim pemberitahuan untuk mendorong pelibatan kembali dan retensi pengguna. Untuk kasus penggunaan seperti perpesanan instan, pesan dapat mentransfer payload hingga 4 KB ke aplikasi klien. Selain itu, FCM menyertakan konsol Notifications, yang dapat Anda gunakan untuk mengirim pemberitahuan ke aplikasi klien.

Firebase Notifications dibuat pada *Firebase Cloud Messaging* dan memiliki FCM SDK yang sama untuk pengembangan klien. Untuk uji coba atau mengirim pesan pemasaran atau keterlibatan dengan penargetan bawaan dan analitik yang andal, Pengguna bisa menggunakan Notifications.

2.4.3.1 Fungsi Utama Firebase

Tabel 2.2 Fungsi Utama Firebase

Penargetan pesan serba guna	Distribusikan pesan kepada aplikasi klien Anda dengan salah satu dari tiga cara — ke satu perangkat, ke grup perangkat, atau ke perangkat yang berlangganan topik.
Dukungan pesan data dan pemberitahuan	Mengirim pemberitahuan hingga 2 KB, *payload data* hingga 4KB, dan mengirim pesan dengan pemberitahuan maupun *payload data*
Perpesanan upstream dari aplikasi klien	Kirimlah kembali pemberitahuan, chat, dan pesan lain dari perangkat ke server Anda melalui saluran koneksi FCM yang bisa diandalkan dan hemat baterai.

2.4.3.2 Firebase Notification

Firebase Notifications adalah layanan gratis yang memungkinkan pemberitahuan pengguna yang ditargetkan untuk pengembang aplikasi seluler. Dibangun di atas *Firebase Cloud Messaging* dan FCM SDK, *Firebase Notifications* (*Notifications*) menyediakan opsi bagi pengembang dan organisasi yang mencari platform pemberitahuan fleksibel yang mengharuskan upaya pengkodean minimal untuk memulai, dan konsol grafis untuk mengirim pesan. Dengan GUI konsol *Notifications,* Anda dapat kembali menumbuhkan interaksi dan mempertahankan basis pengguna, membantu perkembangan aplikasi, dan mendukung kampanye pemasaran.

2.4.3.3 Cara Kerja Firebase Notification

Mengirim pemberitahuan gunakan GUI konsol *Notifications* untuk menulis dan mengirim pemberitahuan ke semua sasaran pesan yang didukung. *Firebase Cloud Messaging* menangani rute dan penyerahan ini ke perangkat yang ditargetkan.

2.4.4 Firebase Realtime Database

Firebase Realtime Database adalah database NoSQL yang di-host di cloud dan dapat digunakan untuk menyimpan dan menyinkronkan data antarpengguna secara real-time. Realtime Database dikirimkan dengan SDK seluler dan web sehingga dapat membuat aplikasi tanpa memerlukan server. Realtime Database juga dapat menjalankan kode backend yang merespons peristiwa yang dipicu oleh database menggunakan Cloud Function for Firebase.

Firebase Realtime Database memungkinkan developer untuk membuat aplikasi kolaboratif dan kaya fitur dengan menyediakan akses yang aman ke database, langsung dari kode sisi klien. Data disimpan di drive lokal. Bahkan saat offline sekalipun, peristiwa realtime terus berlangsung, sehingga pengguna akhir akan merasakan pengalaman yang responsif. Ketika koneksi perangkat pulih kembali, Realtime Database akan menyinkronkan perubahan data lokal dengan update jarak jauh yang terjadi selama klien offline, sehingga setiap perbedaan akan otomatis digabungkan.

Realtime Database menyediakan bahasa aturan berbasis ekspresi yang fleksibel, atau disebut juga Aturan Keamanan Firebase Realtime Database, untuk menentukan metode strukturisasi data dan kapan data dapat dibaca atau ditulis. Ketika diintegrasikan dengan Firebase Authentication, developer dapat menentukan siapa yang memiliki akses ke data tertentu dan bagaimana mereka dapat mengaksesnya.

Realtime Database adalah database NoSQL, sehingga memiliki pengoptimalan dan fungsionalitas yang berbeda dengan database terkait. API Realtime Database dirancang agar hanya mengizinkan operasi yang dapat dijalankan dengan cepat. Hal ini memungkinkan Anda untuk membangun pengalaman realtime yang luar biasa dan dapat melayani jutaan pengguna tanpa mengorbankan kemampuan respons. Oleh karena itu, perlu dipikirkan bagaimana pengguna mengakses data, kemudian buat struktur data sesuai dengan kebutuhan tersebut.

2.4.5 Firebase Authentication

Firebase Authentication merupakan layanan sistem otentikasi yang menerapkan kode client-side, sehingga pengguna dapat mendaftar dan login ke aplikasi Facebook, GitHub, Twitter dan Google (Google Play Games). Selain itu, Firebase termasuk sistem manajemen pengguna dimana pengembang dapat mengaktifkan otentikasi pengguna dengan login email dan kata sandiyang disimpan dengan Firebase. Sebagian besar aplikasi perlu mengetahui identitas pengguna. Dengan mengetahui identitas pengguna, aplikasi dapat menyimpan data pengguna secara aman di cloud dan memberikan pengalaman personal yang sama di setiap perangkat pengguna. Firebase Authentication mendukung otentikasi menggunakan sandi, nomor telepon, penyedia identitas gabungan yang populer, seperti Google, Facebook, dan Twitter, dan lain-lain.

2.4.6 Firebase Storage

Firebase Storage dirancang untuk pengembang aplikasi yang perlu menyimpan dan menampilkan konten buatan pengguna, seperti foto atau video dan menambahkan keamanan Google pada unggah dan unduh berkas untuk aplikasi

Firebase, bagaimana pun kualitas jaringannya. Pengembang dapat menggunakannya untuk menyimpan gambar, audio, video, atau konten lain yang dibuat pengguna secara langsung dari Firebase SDK Klien. Firebase Storage didukung oleh Google Cloud Storage.

2.4.7 API (*Application Programing Interfaces*)

API merupakan software interface yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk library dan menjelaskan bagaimana agar suatu software dapat berinteraksi dengan software lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan software yang akan dibuat, dan kontraktor merupakan API yang mengerjakan bagian tertentu dari software tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut. Interface pada software merupakan suatu entry points yang digunakan untuk mengakses seluruh resources yang terdapat di dalam software tersebut. Dengan adanya API, maka terdapat aturan bagaimana software dapat berinteraksi dengan software lain untuk mengakses resources melalui interface yang telah tersedia.

2.4. Flutter

Gambar 2.10 Logo Flutter

Flutter adalah SDK untuk pengembangan aplikasi mobile yang dikembangkan oleh Google. Sama seperti *react native*, *framework* ini dapat digunakan untuk

membuat atau mengembangkan aplikasi *mobile* yang dapat berjalan pada device iOS dan Android. Dibuat menggunakan bahasa C, C++, Dart dan Skia. pada *framework* ini semua kodenya di *compile* dalam kode native (Android NDK, LLVM, AOT-compiled) tanpa ada *intrepeter* pada prosesnya sehingga proses *compile*-nya menjadi lebih cepat. Dari segi penulisan kodenya, Flutter sangat berbeda dari *react native* dan lebih cenderung mendekati Java Android.

Versi pertama Flutter dikenal sebagai "*Sky*" dan berjalan pada sistem operasi Android. Diresmikan pada perhelatan *Dart developer summit* tahun 2015, dengan tujuan untuk mampu merender grafis secara konsisten pada 120 bingkai per detik. Komponen utama Flutter termasuk :

1. ***Flutter engine***

   Flutter engine, ditulis terutama dengan bahasa pemrograman C++, memberikan dukungan rendering tingkat rendah menggunakan *library* grafik Skia milik Google. Selain itu, *flutter engine* juga berinteraksi dengan perkakas pengembangan perangkat lunak (*SDK*) spesifik-serambi (*flatform-specific*) seperti yang disediakan oleh Android dan iOS.

2. ***Foundation library***

   Foundation library, ditulis dengan bahasa pemrograman Dart, menyediakan fungsi dan *class-class* dasar yang digunakan untuk membangun aplikasi menggunakan Flutter, seperti *API* untuk berkomunikasi dengan *engine*.

3. **Widget spesifik desain**

   Framework Flutter berisi dua set widget yang disesuaikan dengan bahasa desain tertenu. Widget *Material Design* menerapkan bahasa desain Google dengan nama yang sama, sedangkan widget 'Cupertino' meniru desain iOS milik Apple.

   (sumber : https://id.wikipedia.org/wiki/Flutter_(perangkat_lunak)

2.5. Dart

Bahasa pemrograman Dart merupakan bahasa pemrograman *general-purpose* yang dirancang oleh Lars Bak dan Kasper Lund. Bahasa pemrograman ini dikembangkan sebagai bahasa pemrograman aplikasi yang dapat dengan mudah

untuk dipelajari dan disebarkan. Bahasa pemrograman besutan Google ini dapat digunakan untuk mengembangkan berbagai macam platform termasuk di dalamnya adalah *web*, aplikasi *mobile*, *server*, dan perangkat yang mengusung teknologi Internet of Things. Bahasa pemrograman tersebut dapat digunakan untuk mengembangkan aplikasi untuk dijalankan pada berbagai macam peramban modern. Dart juga dapat digunakan untuk mengembangkan aplikasi dari *codebase* tunggal menjadi aplikasi Android maupun iOS. Bahasa pemrograman Dart dapat digunakan secara bebas oleh para developer, karena bahasa ini dirilis secara *open-source* oleh Google di bawah lisensi BSD. Bahasa pemrograman Dart merupakan bahasa pemrograman berbasis *class* dan berorientasi terhadap objek dengan menggunakan sitaks bahasa pemrograman C.

Bahasa ini dikenalkan oleh Google sebagai pengganti bahasa pemrograman JavaScript, akan tetapi secara opsional bahasa ini dapat dikompilasi ke dalam JavaScript dengan menggunakan dart2js *compiler*. Sedikit berbeda dengan bahasa pemrograman JavaScript yang bertipe statis, bahasa pemrograman Dart merupakan bahasa pemrograman bertipe dinamis.

Adapun kelebihan bahasa pemrograman Dart antara lain adalah sebagai berikut :

1. Fleksibel

Seperti yang dijelaskan sebelumnya, salah satu kelebihan bahasa pemrograman Dart adalah bahasa pemrograman tersebut termasuk ke dalam bahasa pemrograman bertipe dinamis. Bahasa pemrograman ini dapat dikompilasi ke dalam bahasa pemrograman JavaScript dengan *compiler* yang sudah disertakan di dalamnya.

Bahasa pemrograman ini dikembangkan untuk mudah digunakan dalam pengembangan, sesuai dengan pengembangan aplikasi modern, dan memiliki implementasi berkinerja tinggi. Bahkan, bahasa pemrograman ini dapat digunakan juga sebelum dikompilasi.

Dart VM menawarkan kemampuan untuk menjalankan secara langsung kode sumber tanpa perlu dikompilasi terlebih dulu. Bahasa pemrograman ini juga dapat langsung digunakan pada peramban Chrome tanpa perlu dikompilasi.

Bahasa pemrograman Dart mendukung banyak arsitektur, termasuk di dalamnya IA-32, X64, MIPS, ARMv5TE, ARMv6, ARMv7, dan arsitektur ARM64. Bahasa

pemrograman ini mendukung secara *native* pengembangan aplikasi *mobile* untuk ke dua platform Android dan iOS.

2. Berdiri Sendiri

Kelebihan bahasa pemrograman Dart lainnya adalah ketersediaan SDK yang dilengkapi dengan berbagai macam *tools* pengembangan. Salah satu *tool*-nya adalah Dart VM, dimana *tool* tersebut akan membantu para developer untuk menjalankan kode dalam lingkungan tampilan *command line*.

Selain itu, dalam SDK tersebut juga terdapat dart2js *compiler* yang dapat digunakan untuk mengkompilasi Dart ke daalam bahasa pemrograman JavaScript. SDK tersebut juga dilengkapi dengan manajer paket yang disebut dengan pup, yang dapat digunakan untuk menggunakan kode pihak ketiga atau berbagi kodingan.

3. Concurency

Bahasa pemrograman Dart memiliki kelebihan dengan adanya konstruksi nyata dari *concurrency* dan paralelisme. Kelebihan bahasa pemrograman Dart ini ditawarkan dengan bentuk Dart Isolates. Dengan adanya Dart Isolates, program-program akan terisolasi untuk bekerja secara independen tanpa adanya pembagian memori, akan tetapi tetap terdapat komunikasi diantaranya. Setiap program Dart menggunaka setidaknya satu buah isolasi.

2.5 Java Script Object Notation (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.JSON terbuat dari dua struktur:

1. 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.

2. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence).

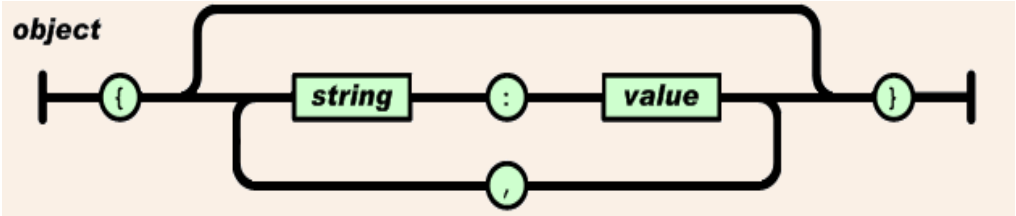
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga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

2.5.1 Bentuk JSON

JSON menggunakan bentuk sebagai berikut:

2. Objek

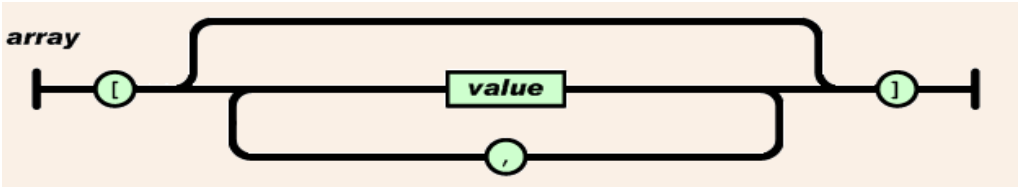
Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurang kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



Gambar 2.11 Objek JSON

3. Larik

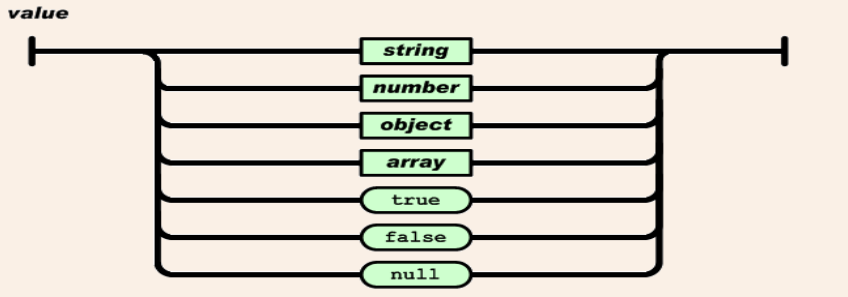
Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



Gambar 2.12 Array JSON

4. Value

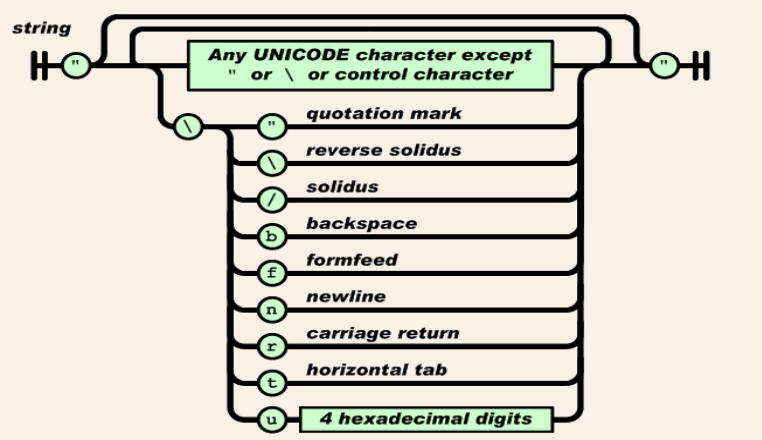
Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur- struktur tersebut dapat disusun bertingkat.



Gambar 2.13 Value JSON

5. String

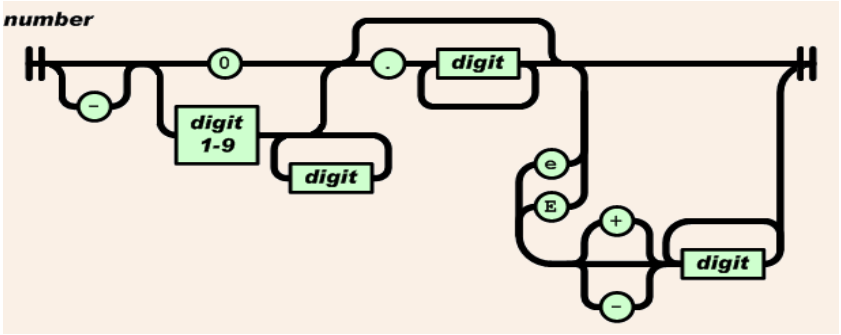
String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2.14 String JSON

6. Angka

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



Gambar 2.15 Number JSON

7. Number

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

2.6 Teori Pemodelan dan UML

Menurut Rosa A & Shalahuddin di dalam bukunya bahwa pemodelan adalah gambaran dari realita yang simpel dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Salah satu pemodelan yang saat ini paling banyak digunakan adalah UML. Unified Modeling Language (UML) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [13].

Secara fisik, UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG (*Object Management Group*).UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Specification,* UML Infrastructure, UML Superstructure, dan Object Constraint Language (OCL). Seluruh spesifikasi tersebut dapat diakses di website http://www.omg.org [13].

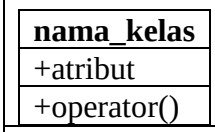
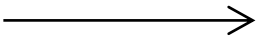
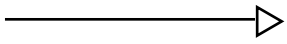
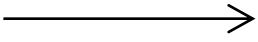
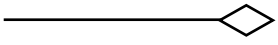
2.6.1 Diagram Kelas

Menurut Rosa A & Shalahuddin bahwa 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 opeasi [13].

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

Berikut adalah simbol-simbol yang ada padad diagram kelas :

Tabel 2.3 Simbol-Simbol Pada Diagram Kelas

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

2.6.2 Usecase Diagram

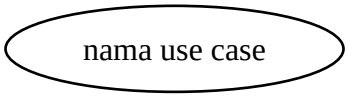
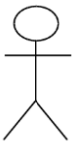
Use Cace atau diagram use case merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu(Rosa A & Shalahuddin ,2018 : 155) [13].

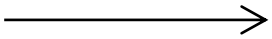
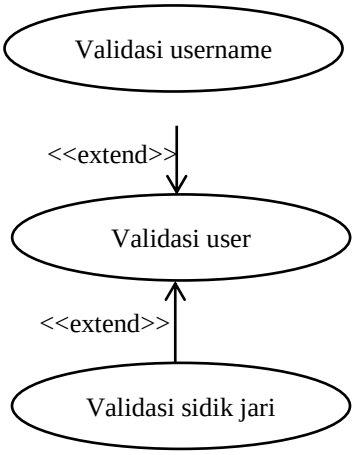
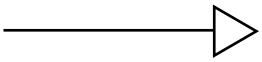
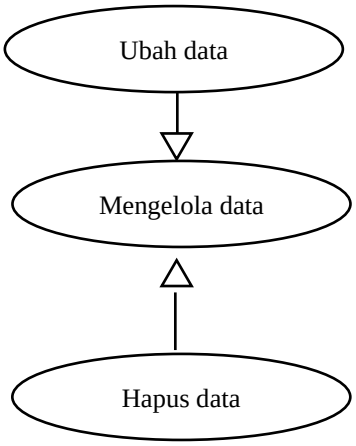
Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

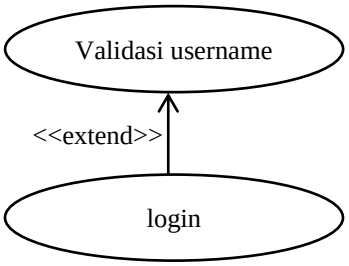
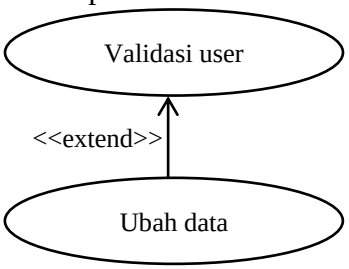
- Aktor merupakan orang, prosesm atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
- Use case merupakan fungsionalitas yang disediakan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

Berikut adalah simbol-simbol yang ada pada diagram use case :

Tabel 2.4 Simbol-Simbol Pada Diagram Use Case

Simbol	Deskripsi
use case 	Fungisonalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nam use case
Aktor / actor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor

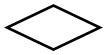
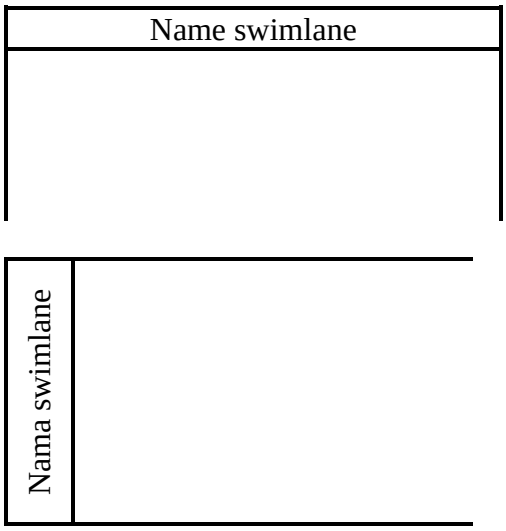
Ekstensi / extend <<extend>> 	Relasi use case tambahan kesebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip *intheritance* pada pemrograman berorientasi objek;biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal  Arah panah mengarah pada use case yang ditambahkan;biasanya use case yang menjadi extend-nya merupakan jenis yang sama dengan use case yang menjadi induknya.
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalanya :  Arah panah mengarah pada use case yang menjadi generalisasinya (umum)

Menggunakan / include / uses <<include>> <<uses>>	Relasi use case tambahan kesebuah use case dimana use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini Adadua sudut pandang yang cukup besar mengenai include di use case: 1. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :  2. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut :  Kedua interpretasi diatas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.6.3 Diagram Aktifitas

Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem(Rosa A & Shalahuddin ,2018 : 161) [13].

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / decision 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
penggabungan / join 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

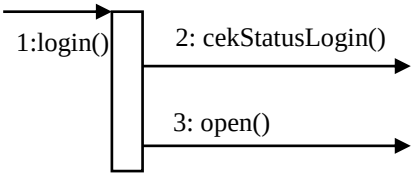
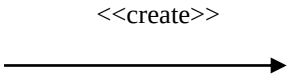
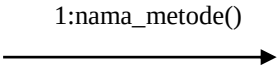
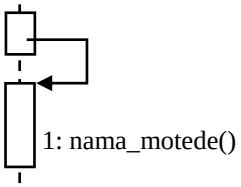
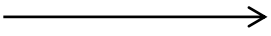
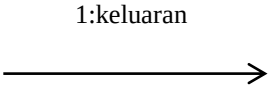
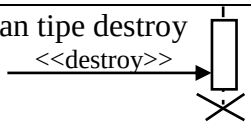
2.6.4 Diagram Sekuen

Diagram sekuen menggambarkan kelakukan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan 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. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case(Rosa A & Shalahuddin ,2018 : 165) [13].

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak [13].

Berikut adala simbol-simbol yang ada pada diagram sekuen :

Simbol	Deskripsi
Aktor  nama aktor Atau nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informais yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup / lifeline 	Menyatakan kehidupan satu objek
Objek nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini

	adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka **cekStatusLogin()** dan **open()** dilakukan didalam metode **login()** Aktor tidak memiliki waktu aktif
Pesan tipe create  	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call  	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek laina atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe send  	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
Pesan tipe return  	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy  	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaikya jika ada create maka ada destroy