BAB II
LANDASAN TEORI

2.1 Landasan Teori

Landasan teori menjelaskan teori dasar yang berhubungan dengan aplikasi yang akan di bangun. Landasan teori yang digunakan dalam penyusunan aplikasi ini meliputi pengertian teori dasar gizi, aplikasi, Android, API, bahasa pemrograman Java, PHP, JSON, MySQL, dan UML.

2.2 Kebutuhan Gizi

Kebutuhan gizi adalah zat gizi yang minimal yang diperlukan untuk diubah menjadi energi agar tubuh mendapatkan energi. Energi yang dibutuhkan tubuh manusia adalah agarnapat menjalankan fungsi-fungsinya dikeseharian dan membantu proses tumbuh kembang khususnya masih dalam masa pertumbuhan.

Terdapat 3 zat gizi dasar yang dapat di olah tubuh untuk diubah menajadi energi yang biasa disebut energi makro, antara lain Karbohidrat, Lemak dan Protein.

2.2.1 Karbohidrat

Karbohidrat memiliki peran penting dalam memberikan sumber energi bagi manusia, juga mudah didapatkan dan memiliki harga yang relatif murah. Tubuh membutuhkan sekitar 60-70% Karbohidrat dari energi total, atau sisa dari kebutuhan energi yang telah dikurangi dengan energi yang berasal dari protein dan lemak. Untuk mendapatkan Angka Kecukupan Karbohidrat (AKK) bisa didapat dengan rumus :

$$AKK = (65\% \times AKE) / 4 \quad . . (II.1)$$

2.2.2 Protein

Protein adalah zat yang paling utama karena sangat dibutuhkan saat masa pertumbuhan dan untuk memperbaiki jaringan tubuh, dan juga sebagai bahan pembetuk ezim-ezim yang diperlukan oleh tubuh, manusia memerlukan 10-15% dari kebutuhan energi total.

$$AKP = (15\% \times AKE) / 4 \; . \; . \; (II.2)$$

2.2.3 Lemak

Lemak memiliki salahsatu fungsi seperti bahan baku hormone, sebagai pelapis organ dan melindungi organ bagian dalam, menghemat serta sebagai insulin terhadap terjadinya perubahan suhu. Tubuh memerlukan lemak sebanyak 10-25% dari kebutuhan energi total. Untuk mendapatkan angka kebutuhan lemah bisa menggunakan rumus

$$AKL = (20\% \times AKE) / 9 \; . \; . \; (II.3).$$

2.3 Rekomendasi

Rekomendasi merupakan saran terhadap apa yang berhubungan dengan keinginan atau kebutuhan dan bersifat menganjurkan atau membenarkan seseorang. Rekomendasi sangat penting untuk meyakinkan orang lain terhadap sesuatu yang tepat dan layak. Ada banyak penerapan rekomendasi pada kehidupan kita sehari-hari. Rekomendasi juga bisa diterapkan pada aplikasi atau sistem yang bisa di kelola secara otomatis. Berikut contoh penerapan rekomendasi pada aplikasi:

1. Rekomendasi barang pada toko *online* (*e-commerce*) yang diberikan berdasarkan barang yang sering dicari oleh pengguna,
2. Rekomendasi tempat wisata berdasarkan tempat terdekat atau tempat wisata yang sedang populer,
3. Rekomendasi makanan berdasarkan makanan kesukaan atau tempat makan yang sedang digemari banyak orang, dan lain-lain.

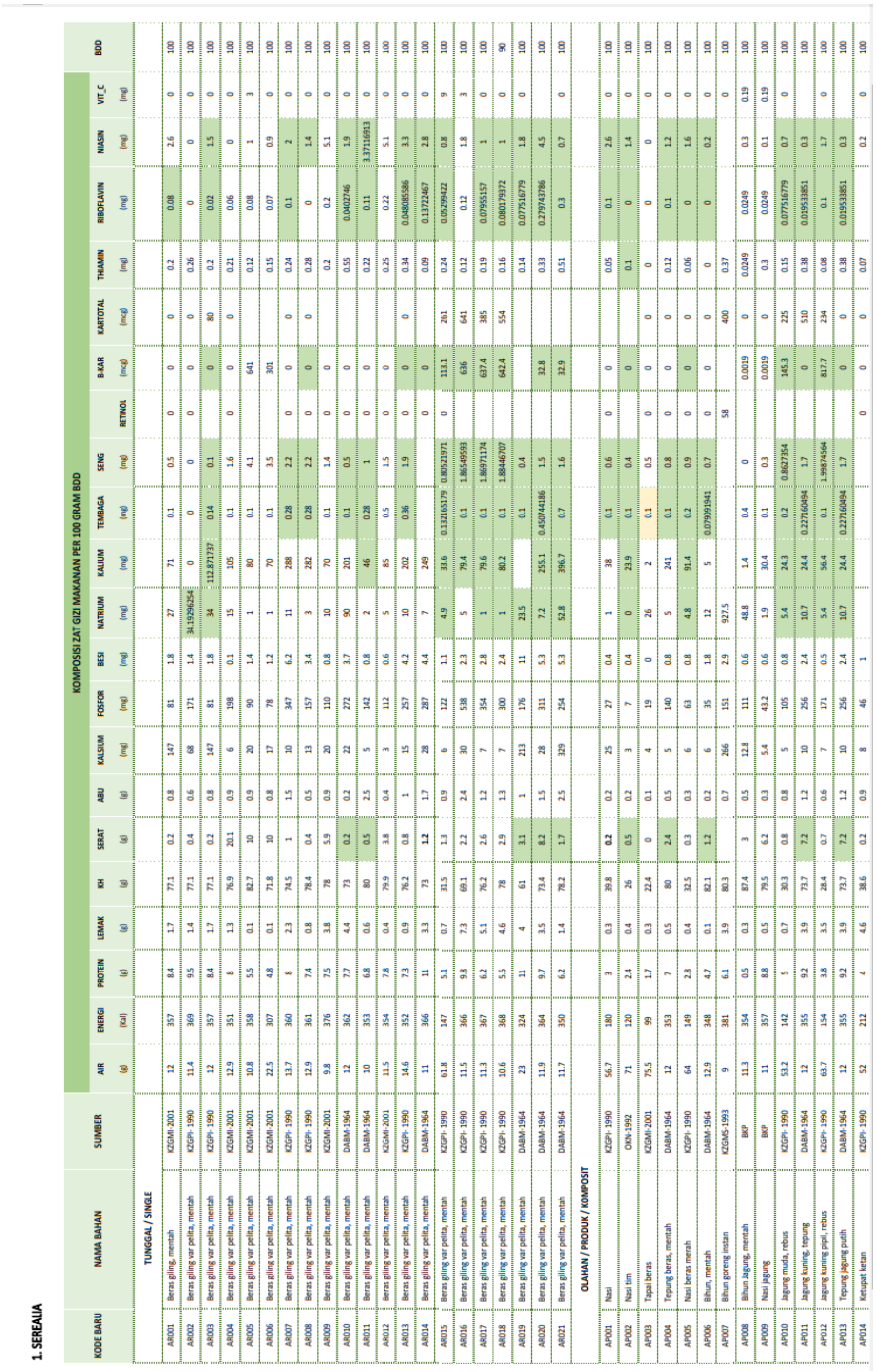
Rekomendasi juga pada sebuah sistem atau aplikasi untuk membantu kebutuhan sesorang dalam melakukan kegiatan sehari-hari. Sistem rekomendasi dapat didefinisikan sebagai program yang mencoba untuk merekomendasikan *item* yang cocok (baik produk atau jasa) untuk pengguna tertentu (individu atau usaha) dengan memprediksi minat pengguna di *item* berdasarkan informasi terkait[5].

Tujuan dari pengembangan sistem rekomendasi adalah untuk mengurangi informasi yang berlebihan dengan mengambil informasi dan layanan yang paling relevan dari sejumlah besar data, sehingga memberikan pelayanan pribadi. Fitur yang paling penting dari sebuah sistem rekomendasi adalah kemampuannya untuk menebak kepentingan dari pengguna dengan menganalisis tinjauan pengguna.

2.4 Tabel Komposisi Pangan Indonesia

Tabel Komposisi Pangan Indonesia atau biasa di singkat TKPI merupakan tabel dari nilai gizi suatu makanan, baik bahan mentah atau yang telah diolah menjadi makanan dengan bumbu. Tabel komposisi pangan merupakan salah satu alat ukur yang sangat penting untuk menyusun menu dan menilai kecukupan asupan konsumsi pangan individu dan kelompok dalam satu wilayah maupun negara baik pada kondisi sehat maupun sakit. Dengan adanya data yang terstandar, masyarakat dapat memiliki cukup informasi dalam memilih dan mengkombinasikan pangan sehat menurut kandungan gizinya.

Hingga saat ini data komposisi pangan di Indonesia masih terus dikembangkan. Pada akhir tahun 2017 telah dihasilkan Tabel Komposisi Pangan Indonesia (TKPI) yang merupakan pengembangan dari TKPI tahun 2009, dengan melengkapi nilai gizi dari TKPI 2009 yang belum memiliki nilai atau masih kosong, menggunakan metode *imputed values* dan *borrowed values* yang berasal dari tabel komposisi bahan pangan negara lain yang serupa. Contoh tabel data komposisi bahan makanan bisa dilihat pada Gambar 2.1. Untuk data lengkap berada pada lampiran D.



1. SEREALIA

KODE BARU	NAMA BAHAN	SUMBER	KOMPOSISI ZAT GIZI MAKANAN PER 100 GRAM BDD																							
			AIR (g)	ENERGI (Kal)	PROTEIN (g)	LEMAK (g)	KH (g)	SERAT (g)	ABU (g)	KALSIUM (mg)	FOSFOR (mg)	BESI (mg)	NATRIUM (mg)	KALIUM (mg)	TEMBAGA (mg)	SENG (mg)	RETINOL (mcg)	B-KAR (mcg)	KAROTOTAL (mcg)	THIAMIN (mg)	RIBOFLAVIN (mg)	NIASIN (mg)	VIT_C (mg)	BDD
	TUNGGAL / SINGLE																									
A0001	Beras giling, mentah	KEMKI-2011	12	357	8.4	1.7	77.1	0.2	0.8	147	81	1.8	27	71	0.1	0.5	0	0	0	0.2	0.08	2.6	0	100
A0002	Beras giling var pelita, mentah	KEGPI- 1990	11.4	369	9.5	1.4	77.1	0.4	0.6	68	171	1.4	34-19296254	0	0	0	0	0	0	0.26	0	0	0	100
A0003	Beras giling var pelita, mentah	KEGPI- 1990	12	357	8.4	1.7	77.1	0.2	0.8	147	81	1.8	34	113.977137	0.14	0.1	0	0	80	0.2	0.02	1.5	0	100
A0004	Beras giling var pelita, mentah	KEMKI-2011	12.9	351	8	1.3	76.9	20.1	0.9	6	198	0.1	15	105	0.1	1.6	0	0	0	0.21	0.06	0	0	100
A0005	Beras giling var pelita, mentah	KEMKI-2011	10.8	358	5.5	0.1	82.7	10	0.9	20	90	1.4	1	80	0.1	4.1		641		0.12	0.08	1	3	100
A0006	Beras giling var pelita, mentah	KEMKI-2011	22.5	307	4.8	0.1	71.8	10	0.8	17	78	1.2	1	70	0.1	3.5		301		0.15	0.07	0.9	0	100
A0007	Beras giling var pelita, mentah	KEGPI- 1990	13.7	360	8	2.3	74.5	1	1.5	10	347	6.2	11	288	0.28	2.2	0	0	0	0.24	0.1	2	0	100
A0008	Beras giling var pelita, mentah	KEGPI- 1990	12.9	361	7.4	0.8	78.4	0.4	0.5	13	157	3.4	3	282	0.28	2.2	0	0	0	0.28	0	1.4	0	100
A0009	Beras giling var pelita, mentah	KEMKI-2011	9.8	376	7.5	3.8	78	5.9	0.9	20	110	0.8	10	70	0.1	1.4	0	0	0	0.2	0.2	5.1	0	100
A0010	Beras giling var pelita, mentah	DABM-1964	12	362	7.7	4.4	73	0.2	0.2	22	272	3.7	90	201	0.1	0.5	0	0	0	0.55	0.0402746	1.9	0	100
A0011	Beras giling var pelita, mentah	DABM-1964	10	353	6.8	0.6	80	0.5	2.5	5	142	0.8	2	46	0.28	1	0	0	0	0.22	0.11	3.3711693	0	100
A0012	Beras giling var pelita, mentah	KEGMI-2011	11.5	354	7.8	0.4	79.9	3.8	0.4	3	112	0.6	5	85	0.5	1.5	0	0	0	0.25	0.22	5.1	0	100
A0013	Beras giling var pelita, mentah	KEGPI- 1990	14.6	352	7.3	0.9	76.2	0.8	1	15	257	4.2	10	202	0.36	1.9	0	0	0	0.34	0.0488586	3.3	0	100
A0014	Beras giling var pelita, mentah	DABM-1964	11	366	11	3.3	73	1.2	1.7	28	287	4.4	7	249			0	0		0.09	0.13722467	2.8	0	100
A0015	Beras giling var pelita, mentah	KEGPI- 1990	61.8	147	5.1	0.7	31.5	1.3	0.9	6	122	1.1	4.9	33.6	0.132165179	0.870127971	0	1111	261	0.24	0.05299422	0.8	9	100
A0016	Beras giling var pelita, mentah	KEGPI- 1990	11.5	366	9.8	7.3	69.1	2.2	2.4	30	538	2.3	5	79.4	0.1	1.86549593		636	641	0.12	0.12	1.8	3	100
A0017	Beras giling var pelita, mentah	KEGPI- 1990	11.3	367	6.2	5.1	76.2	2.6	1.2	7	354	2.8	1	79.6	0.1	1.86971174		637.4	385	0.19	0.07955157	1	0	100
A0018	Beras giling var pelita, mentah	KEGPI- 1990	10.6	368	5.5	4.6	78	2.9	1.3	7	300	2.4	1	80.2	0.1	1.88446707		642.4	554	0.16	0.08017372	1	0	90
A0019	Beras giling var pelita, mentah	DABM-1964	23	324	11	4	61	3.1	1	213	176	11	23.5		0.1	0.4				0.14	0.07516779	1.8	0	100
A0020	Beras giling var pelita, mentah	DABM-1964	11.9	364	9.7	3.5	73.4	8.2	1.5	28	311	5.3	7.2	255.1	0.450744186	1.5		32.8		0.33	0.27974786	4.5		100
A0021	Beras giling var pelita, mentah	DABM-1964	11.7	350	6.2	1.4	78.2	1.7	2.5	329	254	5.3	52.8	396.7	0.7	1.6		32.9		0.51	0.3	0.7	0	100
	OLAHAN / PRODUK / KOMPOSIT																							
AP001	Nasi	KEGPI- 1990	56.7	180	3	0.3	39.8	0.2	0.2	25	27	0.4	1	38	0.1	0.6	0	0		0.05	0.1	2.6	0	100
AP002	Nasi tim	CIKN-1992	71	120	2.4	0.4	26	0.5	0.2	3	7	0.4	0	23.9	0.1	0.4		0		0.1	0	1.4	0	100
AP003	Tepak beras	KEGMI-2001	75.5	99	1.7	0.3	22.4	0	0.1	4	19	0	26	2	0.1	0.5	0	0	0	0	0	0	0	100
AP004	Tepung beras, mentah	DABM-1964	12	353	7	0.5	80	2.4	0.5	5	140	0.8	5	241	0.1	0.8	0	0	0	0.12	0.1	1.2	0	100
AP005	Nasi beras merah	KEGPI- 1990	64	149	2.8	0.4	32.5	0.3	0.3	6	63	0.8	4.8	91.4	0.2	0.9	0	0	0	0.06	0	1.6	0	100
AP006	Bihun, mentah	DABM-1964	12.9	348	4.7	0.1	82.1	1.2	0.2	6	35	1.8	12	5	0.079091941	0.7	0	0	0	0	0	0.2	0	100
AP007	Bihun goreng instan	KEMKI-1993	9	381	6.1	3.9	80.3		0.7	266	151	2.9	927.5				58		400	0.37			0	100
AP008	Bihun jagung, mentah	BKP	11.3	354	0.5	0.3	87.4	3	0.5	12.8	111	0.6	48.8	1.4	0.4	0		0.0019	0	0.0249	0.0249	0.3	0.19	100
AP009	Nasi jagung	BKP	11	357	8.8	0.5	79.5	6.2	0.3	5.4	43.2	0.6	1.9	30.4	0.1	0.3		0.0019	0	0.3	0.0249	0.1	0.19	100
AP010	Jagung muda, rebus	KEGPI- 1990	53.2	142	5	0.7	30.3	0.8	0.8	5	105	0.8	5.4	24.3	0.2	0.8627354		145.3	225	0.15	0.07516779	0.7	0	100
AP011	Jagung kuning, tepung	DABM-1964	12	355	9.2	3.9	73.7	7.2	1.2	10	256	2.4	10.7	24.4	0.227160494	1.7		0	510	0.38	0.03513851	0.3		100
AP012	Jagung kuning pipil, rebus	KEGPI- 1990	63.7	154	3.8	3.5	28.4	0.7	0.6	7	171	0.5	5.4	56.4	0.1	1.99874564		817.7	234	0.08	0.1	1.7	0	100
AP013	Tepung jagung putih	DABM-1964	12	355	9.2	3.9	73.7	7.2	1.2	10	256	2.4	10.7	24.4	0.227160494	1.7		0	0	0.38	0.03513851	0.3		100
AP014	Ketupat ketan	KEGPI- 1990	52	212	4	4.6	38.6	0.2	0.9	8	46	1					0	0	0	0.07			0	100

Gambar 2.1 Tabel Komposisi Pangan

Berikut penjelasan dari data yang ada pada tabel TKPI di atas:

1. Kolom pertama terdapat kode bahan makanan

2. Kolom kedua menampilkan nama bahan makanan tunggal atau yang sudah diolah
3. Data komposisi makanan yang disediakan dengan berat 100gram
4. Terdapat 21 kolom mengandung nilai gizi yang terdapat pada makanan tersebut
5. Kolom BDD (berat dapat dimakan) merupakan persentase dari 100gram makanan

2.5 Aplikasi

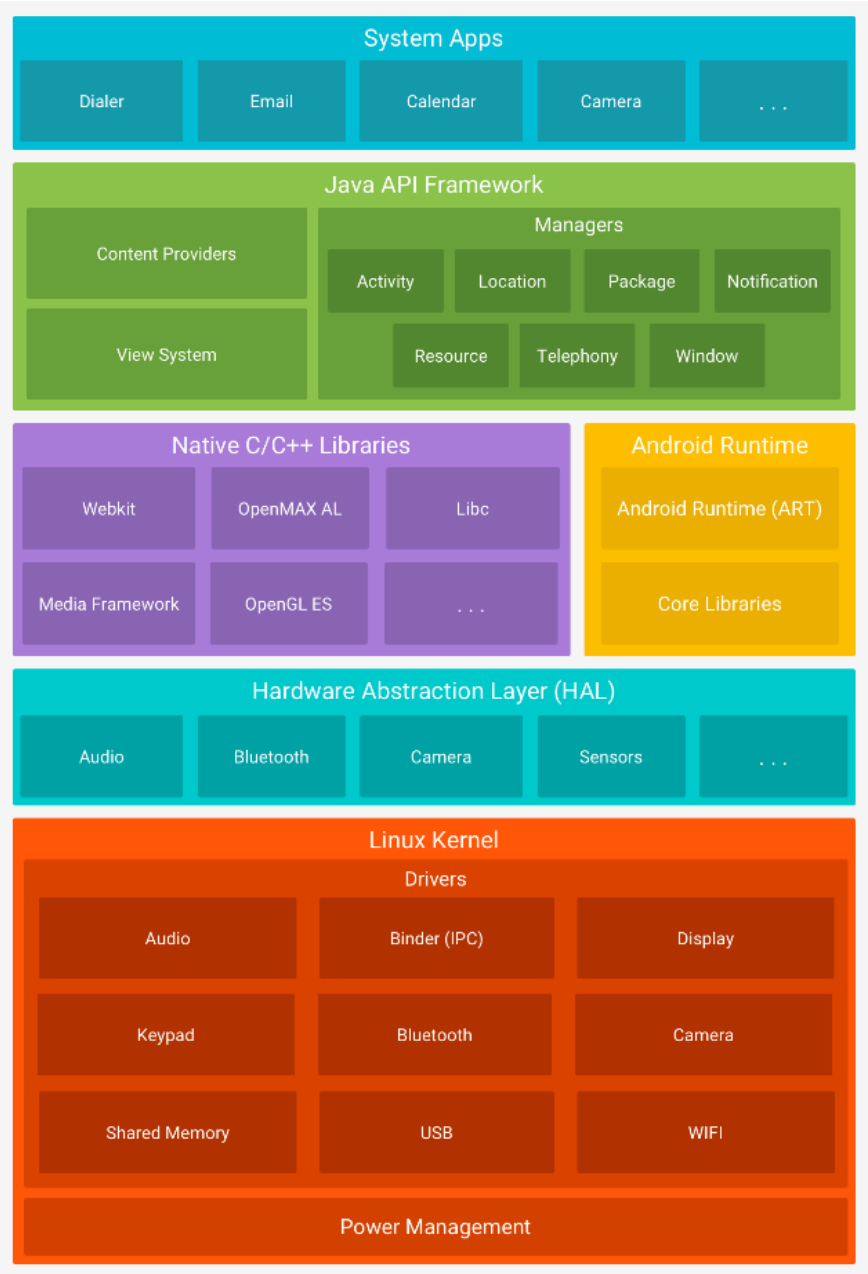
Aplikasi adalah progam yang berisikan perintah-perintah untuk mengelolah kumpulan data agar lebih efektif dan efisien[6]. Pada dunia teknologi informasi, aplikasi sebagai alat mempermudah dan membantu perkerjaaan manusia dari cara manual menjadi optimal dengan memanfaatkan komputer. Aplikasi dapat dibangun ataupun dikembangkan dengan maksud membatu tugas yang bersifat umum atau khusus.

Pada dunia teknologi aplikasi adalah sebuah perangkat lunak dibangun untuk membantu atau mempermudah pekerjaan manusia. Aplikasi juga merupakan suatu program perangkat lunak yang berjalan pada suatu sistem tertentu. Sebuah aplikasi dapat dibuat dan dikembangkan dengan tujuan untuk melakukan tugas yang bersifat umum atau juga dapat dikembangkan untuk melakukan tugas yang bersifat spesifik/ khusus.

2.5.1 Android

Secara istilah aplikasi adalah sistem operasi untuk telepon seluler yang berbasis Linux[7]. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Dengan begini pengembangan maupun pembuatan Android bisa dipermudah oleh setiap *developer*.

Android memiliki komponen utama yang dapat membuat berjalan dengan baik dan disebut dengan Arsitektur *Platform* Android. Berikut diagram komponen dari Android :



Gambar 2.2 Arsitektur *Platform* Android

2.5.1.1 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 dan perubahan fitur berdasarkan tanggal rilis :

1. Android 1.0 (API level 1)

   Android 1.0 merupakan versi komersial pertama Android yang dirilis pada 23 September 2008. Awalnya versi 1.0 ini akan dinamai "*Astro*", tapi dikarenakan ada permasalahan hak cipta dan *trademark* nama "*Astro*" maka penamaan pada versi pertama tersebut dibatalkan oleh Google.

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 "*Bender*", meskipun nama ini tidak digunakan secara resmi dikarenakan masalah yang sama dengan versi pertamanya.

3. Android 1.5 *Cupcake* (API level 3)
   Versi ini dirilis pada tanggal 27 April 2009, menggunakan Kernel Linux 2.6.27. Mulai dari versi ini, android melakukan penamaan berdasarkan nama pencuci mulut. Karena ini merupakan versi ketiga yang dirilis, sehingga penamaan diawali dari huruf "C", yaitu "*Cupcake*". Pembaruan pada versi ini mulai terlihat dari fitur baru dan perubahan UI.

4. Android 1.6 *Donut* (API level 4)
   Pada 15 September 2009, Android 1.6 dirilis. Versi ini dinamai "*Donut*", dan menggunakan Kernel Linux 2.6.29. Android 2.0 – 2.1 *Eclair* (API level 5-7).

5. Android 2.2 – 2.2.3 *Froyo* (API level 8)
   Pada 20 Mei 2010, SDK Android 2.2 dirilis dan diberi nama *Froyo* (singkatan dari *Froze Yoghurt*) menggunakan Kernel Linux 2.6.32.

6. 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.

7. 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..

8. Android 4.0 – 4.0.4 *Ice Cream Sandwich* (API level 14 – 15)
   SDK Android 4.0.1 (*Ice Cream Sandwich*), berdasarkan Kernel Linux 3.0.1, dirilis pada 19 Oktober 2011.

9. 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).

10. 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..

11. Android 5.0 *Lollipop* (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, dapat dipakai pada *platform* komputasi Android *Wear*, dan platform pelacakan kesehatan *Google Fit*.

12. Android 6.0 *Marshmallow* (API level 23)

    Android 6.0 ini dirilis pada tanggal 25 Juni 2014 dan diberi kode nama *Marshmallow* pada saat acara Google I/O. 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.

13. 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.*

14. 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.

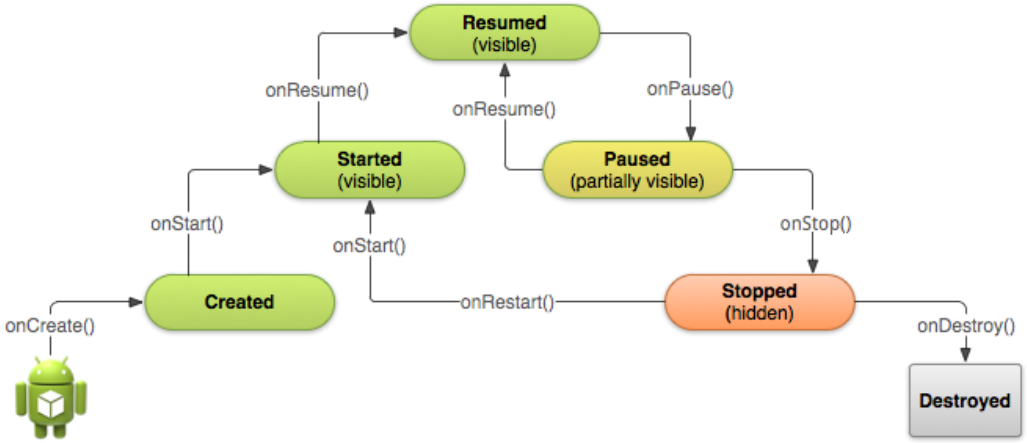
15. Android 9.0 *Pie* (API level 27)

    Android 9.0 resmi dirilis pada Agustus 2018 dengan menggunakan nama kode "*Pie*". Pada versi ini Google mengoptimalkan penggunaan baterai, sehingga menjaga ketahanan perangkat untuk bisa digunakan dalam jangka waktu lama.

2.5.1.2 Siklus Hidup Android

Di dalam sistem operasi Android terdapat siklus hidup dimana pada siklus ini untuk menavigasi antara tahap *Activity Life-Cycle*, Android itu sendiri dengan menyediakan 6 method inti *callback* yaitu *onCreate*(), *onStart*(), *onResume*(), *onPause*(), *onStop*(), dan *onDestroy*(). Dalam siklus hidup *method callback*, kita dapat mendeklarasikan cara perilaku *activity* saat pengguna meninggalkan dan memasuki kembali ke sebuah *activity* itu. Disitulah peranan *method callback* dalam

sebuah aplikasi pada *platform* Android . Berikut pada Gambar 2.3 merupakan *Activity Life-Cycle* pada *platform* Android.



Gambar 2.3 Siklus Hidup Android

Dari siklus tersebut dapat diketahui bahwa sebuah kelas *Activity* menyediakan 6 *method callback*, dan berikut penjelasannya:

1. *onCreate*()

   Method ini adalah *method* utama dari setiap *Activity*. *Method* ini akan dipanggil pertama kali ketika menjalankan sebuah sistem. Pengembang harus mengimplementasikan metode *onCreate*() untuk menjalankan logika memulai aplikasi dasar yang hanya boleh terjadi satu kali selama hidup aktivitas. Misalnya, implementasi *onCreate*() Anda harus mendefinisikan antarmuka pengguna dan mungkin membuat *instance* beberapa variabel dalam cakupan kelas.

2. *onStart*()

   Method ini dipanggil ketika *method onCreate*() telah dipanggil. *onStart*() dipanggil ketika terlihat oleh *user*. *Method* ini selesai dengan cepat dan dilanjutkan dengan *method* setelahnya yaitu *onResume*().

3. *onResume*()

Method ini dipanggil ketika *method onStart*() selesai dipanggil. *Method* ini adalah keadaan dimana pengguna berinteraksi dengan aplikasi. Aplikasi akan tetap dalam keadaan ini sampai terjadi suatu statement dari aplikasi semisal menerima panggilan telepon atau mematikan layar *smartphone*.

4. *onPause*()
   Method ini dipanggil ketika pengguna meninggalkan *activity* (meskipun tidak selalu berarti *activity* dihancurkan). *Method* ini berguna untuk menghentikan sementara operasi yang sedang berjalan semisal menjeda pemutaran musik dan lain-lain.

5. *onStop*()
   Method ini dipanggil ketika *activity* tidak terlihat lagi oleh pengguna, dengan kata lain *activity* berhenti dijalankan. Hal ini dapat terjadi semisal ada aktivitas baru dijalankan meliputi seluruh layar. Sistem juga dapat menghubungi *method* ini ketika *activity* selesai berjalan, dan akan segera dihentikan.

6. *onDestroy*()
   Method ini adalah *method callback* ketika *activity* telah selesai dijalankan dan kemudian memanggil *method finish*() atau karena sistem untuk sementara menghancurkan proses yang berisi *activity* tersebut untuk menghemat ruang memori.

2.5.2 API

API adalah *software interface* yang memiliki intruksi dan disimpan dalam bentuk *library* dan dapat terhubung atau berinteraksi dengan *software* lain[8]. Secara struktural, API merupakan spesifikasi dari suatu struktur data, *object*, *function*, beserta parameter-parameter yang diperlukan untuk mengakses *resource*

dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu *interface* yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain.

Dengan API, panggilan-panggilan yang bolak-balik antar aplikasi diatur melalui *web service*. *Web service* adalah kumpulan standar teknis dan protokol, termasuk XML (*Extensible Markup Language*), bahasa umum yang digunakan oleh aplikasi-aplikasi tersebut selama berkomunikasi di internet. API dan *web service* sepenuhnya bekerja di belakang layar. Dengan demikian, API menjadi data terbuka milik perusahaan *software* atau perusahaan lainnya yang bisa digunakan untuk pembuatan aplikasi dari layanan yang telah diberikan.

2.5.3 *Web Service*

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. *Web service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu *website* untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan *web service*. *Web service* menyimpan data informasi dalam *format* XML, sehingga data ini dapat diakases oleh sistem lain walaupu berbeda *platform*, sistem operasi, maupun bahasa *compiler*.

Web service bertujuan untuk meningkatkan kolaborasi antar pemrograman dan perusahaan, yang memunkinkan sebuah fungsi di dalam. *Web service* dapat di pinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya. Beberapa alasan mengapa digunakannya *web service* adalah sebagai berikut :

1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa bisnis *logic* atau *class* dan objek yang terpisah dalam satu

ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.

2. *Web service* memiliki kemudahan dalam proses *deployment*, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web service* cukup di-*upload* ke *web server* dan siap diakses oleh pihak-pihak yang telah di berikan otoritas.
3. *Web service* berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian *web service* tidak memerlukan konfigurasi khusus di sisi *firewall*.

2.5.4 *Firebase*

Firebase menyediakan *database realtime* dan *backend* sebagai layanan (*Backend as a Service*). Layanan ini menyediakan pengembang aplikasi API yang memungkinkan aplikasi data yang akan disinkronisasi pada klien dan di simpan di *cloud firebase*. *Firebase* menyediakan *library* untuk berbagai *client platform* yang memungkinkan integrasi dengan Android, iOS, JavaScript, Java, Objective-C, dan NodeJs juga disebut sebagai layanan DbaaS (*Database as A Service*) dengan konsep *realtime*.

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.

2.5.4.1 *Firebase Cloud Messaging*

FCM adalah sebuah layanan yang digunakan untuk melakukan pemberitahuan (*notifications*) pada aplikasi berbasis Android, iOS maupun aplikasi *web*. Dahulunya *Firebase Cloud Messaging* ini bernama *Google Cloud Messaging*

atau GCM, namun sekarang sudah berubah dan menjadi lebih besar di *Firebase*. Langkah utama untuk mengimplementasikan FCM di Android adalah membuat *project* di *Firebase* dan mengintegrasikannya dengan aplikasi Android[9]. Langkah-langkah yang diperlukan adalah :

1. Membuat akun atau *project console* di *Firebase Console*, lalu *Create New Project* atau buatlah *project baru*, beri nama sesuai keperluan anda.
2. Setelah masuk *dashboard*, lalu carilah tombol *Add Firebase to your Android app* dan ikuti saja langkahnya (masukan nama *namespace* dari aplikasi anda, lalu *generate* dan *download file configgnya* (google-services.json).
3. Letakan *file* google-services.json tersebut di *folder* app/ dari *project* anda.
4. Jangan lupa tambahkan dependensi pada gradle, lalu *sync project* anda.

Sejauh ini ada dua metode cara kirim notifikasi. Metode pertama adalah paling *simple*, mengirim melalui halaman *console firebase*. Secara sederhana, kita *login* ke *Console Firebase*, lalu kita mengirimkan pesan notif melalui fitur yang sudah tersedia disana. Metode kedua adalah dengan dengan membuat *server* sebagai pengirim pesan, bahasa pemrogramannya bisa menggunakan php, go lang, phyton, java ataupun bahasa alien lainnya.

2.5.5 JAVA

Java adalah salah satu bahasa pemrograman yang paling populer. Java dapat digunakan untuk berbagai hal, termasuk pengembangan perangkat lunak, aplikasi *mobile* dan pengembangan sistem yang besar. Inilah yang membuat bahasa pemrograman Java sangat terkenal di lingkungan pengembangan perangakat lunak.

Bahasa pemrograman java banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana. Java merupakan bahasa pemrograman yang bersifat umum/ non-spesifik (*general*

purpose), dan secara khusus didesain 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. Berikut contoh dari bahasa pemrograman java :

```
Public class HelloWorld (){
       Public static void main (String[] args){
              System.out.println("Hello World");
       }
}
```

2.5.6 PHP

PHP (*HyperText Preprocessor*) adalah sebuah bahasa utama *script server-side* yang disisipkan pada HTML yang dijalankan di *server*, dan juga bisa digunakan untuk membuat aplikasi *dekstop*. PHP bersifat *open-source* sehingga dapat dipakai secara cuma-cuma dan mampu lintas *platform*, dalam artian bisa digunakan pada sistem operasi Windows ataupun Linux[10].

Tentunya bahasa pemerograman PHP berbeda dengan HTML, pada PHP *script*/ kode yang dibuat tidak dapat ditampilkan pada halaman/ muka *website* begitu saja, tapi harus diproses terlebih dahulu oleh *web server* lalu di tampilkan dalam bentuk halaman *website* di *web browser*, script PHP juga dapat di sisipkan pada HTML dan script PHP selalu diawali dengan <?php dan di akhiri dengan ?>. Berikut contoh penggunaan PHP :

```
<?php
      echo "Hello World…!!!";
      echo "<br />";
      print "Hello Again World…!!!";
      print "<br /><br />";
?>
```

2.5.7 JSON

JSON (*JavaScript Obejct Notation*) merupakan format pertukaran data yang ringan, mudah dibaca, dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat oleh komputer. Pertukaran data dengan menggunakan *format* JSON sangat ideal karena *format* JSON berbasis teks dan bisa terbaca dengan mudah oleh manusia, serta digunakan untuk merepresentasikan struktur data sederhana dan larik assosiatif.

Sebuah objek JSON adalah *format data key-value* yang biasanya di *render* di dalam kurung kurawal. Saat kita bekerja dengan JSON, kita akan sering melihat objek JSON disimpan di dalam sebuah *file* .json, tapi mereka juga dapat disimpan sebagai objek JSON atau *string* di dalam sebuah program. Penggunaan JSON dapat dilihat pada contoh berikut :

```
{
      "data_diri": {
            "nama_depan" : "Sammy",
            "nama_belakang" : "Ducky",
            "alamat" : "Sekeloa",
            "pekerjaan" : "Mahasiswa"
      }
}
```

Pasangan *key-value* memiliki tanda titik dua diantara mereka "*key*" : "*value*". Setiap *key-value* dipisahkan oleh sebuah koma, sehingga ditengah isi sebuah JSON terlihat seperti in: "*key*" : "*value*", "*key*" : "*value*", "*key*": "*value*". Pada contoh di atas, nilai pertama pasangan *key-value* kita adalah "data_diri" : "(data berupa nama, alamat, dan pekerjaan)". Lalu dalam *value* data_diri terdapat key baru yaitu “nama_depan” dan memiliki value : “Sammy”.

2.6 MySQL

MySQL merupakan *software database* yang termasuk paling populer di lingkungan Linux, kepopuleran ini karena ditunjang oleh performansi *query* dari *database*nya yang saat itu bisa dikatakan paling cepat dan jarang bermasalah. Dalam penggunaan *database* MySQL, setiap perintah yang diketikkan disebut *query*. Perintah pada MySQL dapat dikelompokkan menjadi 3 bagian, antara lain:

2.6.1 DDL (*Data Definition Language*)

Perintah dalam SQL yang pertama adalah perintah DDL. DDL dapat diartikan sebagai perintah yang berhubungan dengan pendefinisian dari suatu struktur *database*. Terdapat beberapa perintah DDL pada MySQL sebagai berikut:

1. *CREATE*, berfungsi untuk membuat *database* baru, *table* baru, *view* baru dan kolom baru.
2. *ALTER*, berfungsi untuk mengubah struktur tabel. Seperti mengubah kolom, menambah kolom, mengganti nama tabel, memberikan atribut pada kolom maupun menghapus kolom.
3. *DROP*, berfungsi untuk menghapus *database* dan tabel yang telah dibuat.
4. *TRUNCATE*, berfungsi untuk menghapus semua catatan dari tabel.
5. *COMMENT*, berfungsi untuk menambahkan komentar untuk sebagai penjelasan pada data.
6. *RENAME*, berfungsi untuk mengubah nama obyek.

Berikut contoh penggunaan DDL dalam MySQL:

```
CREATE DATABASE mahasiswa;
ALTER TABLE mata_kuliah ADD (thn_ajaran CHAR(9));
DROP DATABASE ukm;
TRUNCATE TABLE kendaraan;
RENAME nilai TO nilai_akhir;
```

2.6.2 DML (*Data Manipulation Language*)

Data Manipulation Language (DML) ialah perintah yang digunakan untuk mengelola/ memanipulasi data dalam *database*. Terdapat beberapa perintah DML pada MySQL sebagai berikut :

1. *SELECT* berfungsi untuk mengambil/ menampilkan data dari *database*.
2. *INSERT* berfungsi untuk memasukkan data ke dalam tabel.
3. *UPDATE* berfungsi untuk memperbarui data dalam tabel.
4. *DELETE* berfungsi untuk menghapus data dari tabel.
5. *CALL* berfunsi untuk memanggil subprogram PL / SQL atau Java.
6. *EXPLAIN PLAN* berfungsi untuk menjelaskan jalur akses ke data.
7. *LOCK TABLE* berfungsi untuk mengunci tabel.

Berikut contoh penggunaan DML pada MySQL :

```
SELECT nim FROM mahasiswa;
INSERT INTO mahasiswa VALUES ("10119001","Sammy
Ducky");
UPDATE mahasiswa SET nama = "Michael Asep" WHERE
nim = "10119001";
DELETE FROM mahasiswa WHERE nim = "10119001";
```

2.6.3 DCL (*Data Control Language*)

Data Control Language (DCL) ialah perintah yang digunakan untuk melakukan pengontrolan data dan *server database*nya. Terdapat beberapa perintah DCL pada MySQL sebagai berikut :

1. GRANT berfungsi untuk memberikan hak akses pengguna ke database.
2. REVOKE berfungsi untuk menghilangkan hak akses yang telah diberikan dengan perintah GRANT.

```
GRANT all privileges on * to nm_user@localhost
identified by 'nm_passwd' with grand option;

REVOKE all on nm_db.nm_tbl from nm_user@localhost
identified by 'nm_passwd';
```

2.7 UML (Unified Modelling Language)

UML (*Unified Modelling Language*) adalah sebuah bahasa yang telah menjadi standar dalam industri visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML dapat dibuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML menggunakan *class* dan *operation* dalam konsep dasarnya, maka lebih cocok untuk penulisan perangkat lunak dalam bahasa berorientasi objek seperti C++, Java, atau VB. NET[11].

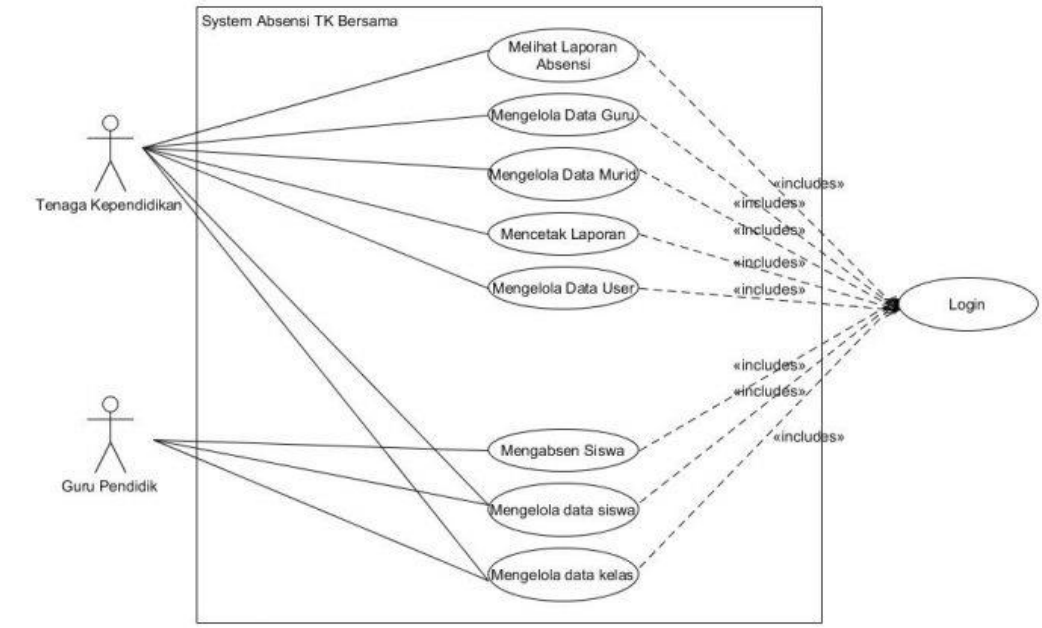
Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/ semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. 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*).

2.7.1 *Use Case Diagram*

Use case diagram merupakan bagian tertinggi dari fungsionalitas yang dimiliki sistem yang menggambarkan bagaimana seseorang atau aktor dalam menggunakan dan memanfaatkan sistem. Contoh *use case* dapat dilihat pada Gambar 2.4. *Use case* terdiri dari tiga bagian yaitu identifikasi aktor, identifikasi *use case*, dan skenario *use case*. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut *use case* dan aktor.

Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

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.



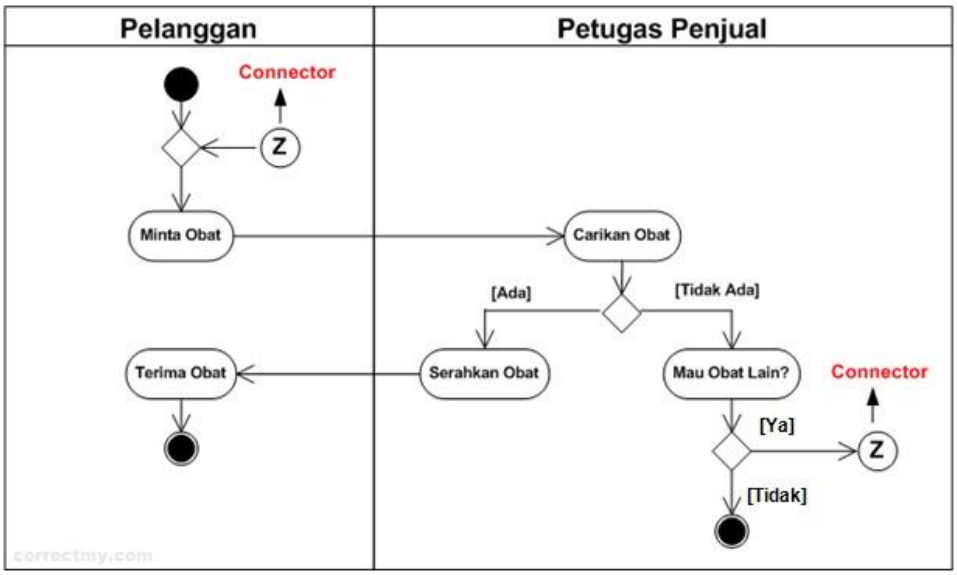
Gambar 2.4 Contoh *Use Case Diagram*

2.7.2 *Activity Diagram*

Diagram aktivitas suatu *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. Contoh *activity diagram* dapat dilihat pada Gambar 2.5. Diagram aktifitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktifitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/ *user interface* dimana setiap aktifitas dianggap memiliki sebuah rancangan antarmuka tampilan

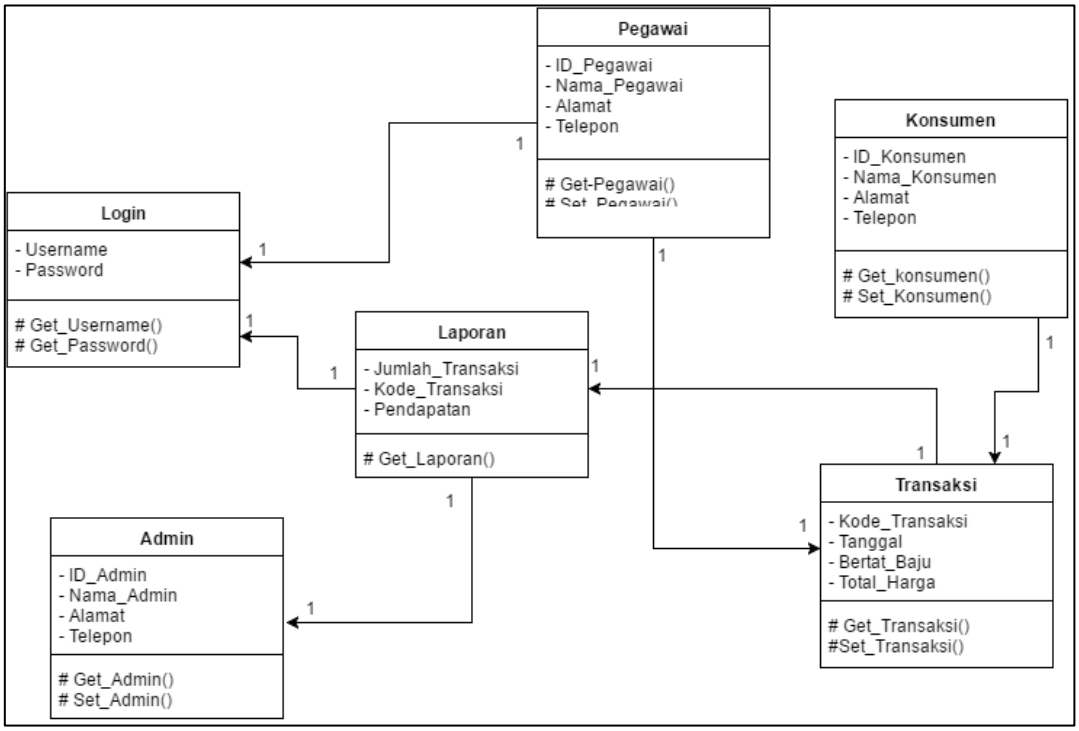
3. Rancangan pengujian dimana setiap aktifitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4. Rancangan menu yang ditampilkan pada diagram aktifitas



Gambar 2.5 Contoh *Activity Diagram*

2.7.3 *Class Diagram*

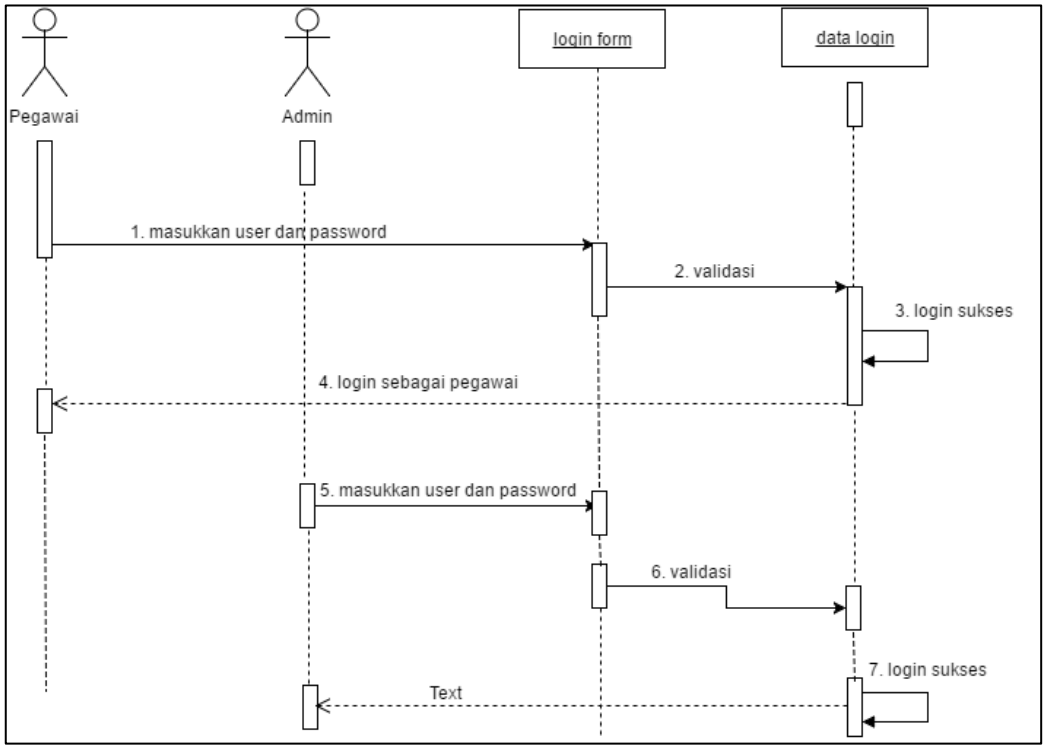
Class diagram membantu dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (dalam *logical view*) dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Contoh *class diagram* dapat dilihat pada Gambar 2.6.



Gambar 2.6 Contoh *Class Diagram*

2.7.4 *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 pada Gambar 2.7 contoh dari *sequence diagram*.



Gambar 2.7 Contoh *Sequence Diagram*