BAB 2
LANDASAN TEORI

2.1 Tinjauan Tempat Penelitian

Dinas Pangan Kabupaten Majalengka merupakan organisasi perangkat daerah (OPD) di Kabupaten Majalengka yang bertanggung jawab dalam rangka penyelenggaraan pemerintahan di daerah. Dinas Pangan Kabupaten Majalengka dibentuk pada tanggal 30 Desember 2009 berdasarkan Peraturan Daerah Kabupaten Majalengka No. 10 Tahun 2009 tentang Organisasi Perangkat Daerah Kabupaten Majalengka, yang sebelumnya bernama Badan Pelaksana Penyuluhan Pertanian, Perikanan dan Kehutanan (BP4K).

Pada tahun 2016, Pemerintah Kabupaten Majalengka melakukan restrukturisasi pembentukan susunan organisasi perangkat daerah (OPD) yang baru, Badan Pelaksana Penyuluhan Pertanian, Perikanan dan Kehutanan (BP4K) namanya berubah menjadi Dinas Pangan. Pembentukan susunan perangkat daerah yang baru dilakukan guna menyesuaikan kewenangan bagi pemerintah daerah sebagaimana diamanatkan Undang-undang Pemerintah Daerah Nomor 23 Tahun 2014.

Dinas Pangan Kabupaten Majalengka melaksanakan satu urusan wajib yaitu Ketahanan Pangan dan satu urusan lainnya yaitu Penyuluhan. Oleh karena itu, dalam kegiatannya menggabungkan beberapa sektor pembangunan, yaitu ketahanan pangan, pertanian, perikanan, dan kehutanan.

2.1.1 Visi dan Misi

Sebagai organisasi perangkat daerah (OPD), Dinas Pangan Kabupaten Majalengka memiliki visi dan misi untuk meningkatkan kinerja dan dapat mencapai tujuan yang diinginkan. Adapun visi dan misi pada Dinas Pangan Kabupaten Majalengka, yaitu :

2.1.1.1 Visi

Visi Dinas Pangan Kabupaten Majalengka adalah "**Terwujudnya profesionalisme penyuluh, pelaku utama dan pelaku usaha yang mandiri, kelembagaan ekonomi petani yang maju serta ketahanan pangan mantap menuju Majalengka Makmur**". Visi ini dapat diurai menjadi beberapa kata kunci, yaitu :

1. *Profesionalisme Penyuluh*, yang dimaksud dengan profesionalisme penyuluh adalah suatu komitmen terhadap keahliannya dengan cara selalu meningkatkan kemampuan profesionalnya dan terus menerus mengembangkan strategi-strategi yang digunakan dalam melakukan pekerjaan.
2. *Pelaku Utama dan Pelaku Usaha yang Mandiri*, yang dimaksud dengan mandiri adalah mampu mengelola usaha (baik perorangan maupun kelompok) yang efisien dan menguntungkan secara berkelanjutan serta ramah lingkungan. Ciri dari kemandirian adalah mempunyai otoritas dan keswadayaan dalam menentukan jenis, volume dan sistem usaha taninya serta kegiatan-kegiatan dalam kelompok yang dibentuknya.
3. *Kelembagaan Ekonomi Petani yang Maju*, yang dimaksud dengan hal ini adalah kelembagaan ekonomi petani yang telah memiliki kemitraan dengan pelaku usaha secara kontinu, mampu merencanakan kegiatan usaha tani tahunan secara baik, dan adanya kemampuan untuk memupuk modal.
4. *Ketahanan Pangan Mantap,* yang dimaksud hal ini adalah ketika semua orang, setiap saat, memiliki akses fisik dan ekonomis pada pangan yang cukup, aman dan bergizi untuk memenuhi kebutuhan makanan dan seleranya dalam rangka kehidupan yang sehat dan aktif.
5. *Majalengka Makmur*, yang dimaksud dengan hal ini adalah terwujudnya suatu tatanan masyarakat, pemerintahan dan pembangunan yang maju, aman, kondusif, mandiri, unggul dan religius.

2.1.1.2 Misi

Dari visi di atas, kemudian diuraikan misi yang harus dilakukan dalam rangka pencapaian visi tersebut. Misi yang diemban adalah :

1. Meningkatkan standar kompetensi penyuluh
2. Meningkatkan kualitas dan kuantitas sarana penyuluhan
3. Mendorong peningkatan skala usaha kelembagaan petani menjadi Lembaga Ekonomi Petani yang mandiri
4. Meningkatkan kerja sama pelaku utama dan pelaku usaha agar tercipta sinergi
5. Meningkatkan capaian indikator Standar Pelayanan Minimal Bidang Ketahanan Pangan

2.1.2 Tujuan

Perumusan tujuan dilakukan dengan memperhatikan faktor-faktor penentu keberhasilan yang diperoleh melalui analisis SWOT. Berdasarkan analisis SWOT faktor-faktor yang dianggap paling berpengaruh dalam menentukan keberhasilan dan kegagalan baik internal maupun eksternal adalah sebagai berikut :

1. Lingkungan Internal terdiri atas :

a. Kekuatan (*Strenghts*)
   i. Peraturan Daerah (Perda) Nomor 10 Tahun 2009 tentang Pembentukan Organisasi Perangkat Daerah Kabupaten Majalengka.
   ii. Sumber daya manusia (SDM) Penyuluh Pertanian, Perikanan dan Kehutanan yang cukup tersedia.
   iii. Dukungan dan komitmen dari atasan (eksekutif) dan lembaga legislatif cukup baik.
   iv. Kendaraan untuk kegiatan operasional cukup tersedia.
   v. Institusi pendukung program kegiatan seperti Balai Penyuluhan tersedia.

b. Kelemahan (*Weaknesses*)
   i. Kualitas SDM masih belum merata.
   ii. Dana penunjang program kegiatan kurang memadai.
   iii. Tenaga administrasi dan pelaksana sangat kurang.

2. Lingkungan Eksternal terdiri atas :

a. Peluang (*Opportunity*)

i. Petani senantiasa masih memerlukan pembinaan.

ii. Ilmu pengetahuan dan teknologi selalu berkembang.

iii. Sumber Daya Alam (SDA) cukup tersedia.

iv. Sektor pertanian, perikanan dan kehutanan tangguh dalam menghadapi krisis ekonomi.

v. Perekonomian semakin membaik.

b. Tantangan (*Threats*)

i. Era perdagangan bebas.

ii. Kemampuan petani dalam mengadopsi teknologi masih rendah.

iii. Pengkaderan di sektor pertanian, perikanan dan kehutanan lemah.

iv. Ketahanan pangan masyarakat belum mantap.

v. Kepemilikan lahan usaha tani oleh petani relatif sempit.

Berdasarkan kepada faktor-faktor kunci keberhasilan sebagaimana diuraikan sebelumnya, maka ditetapkan tujuan yang dapat dilihat pada Tabel 2.1. berikut :

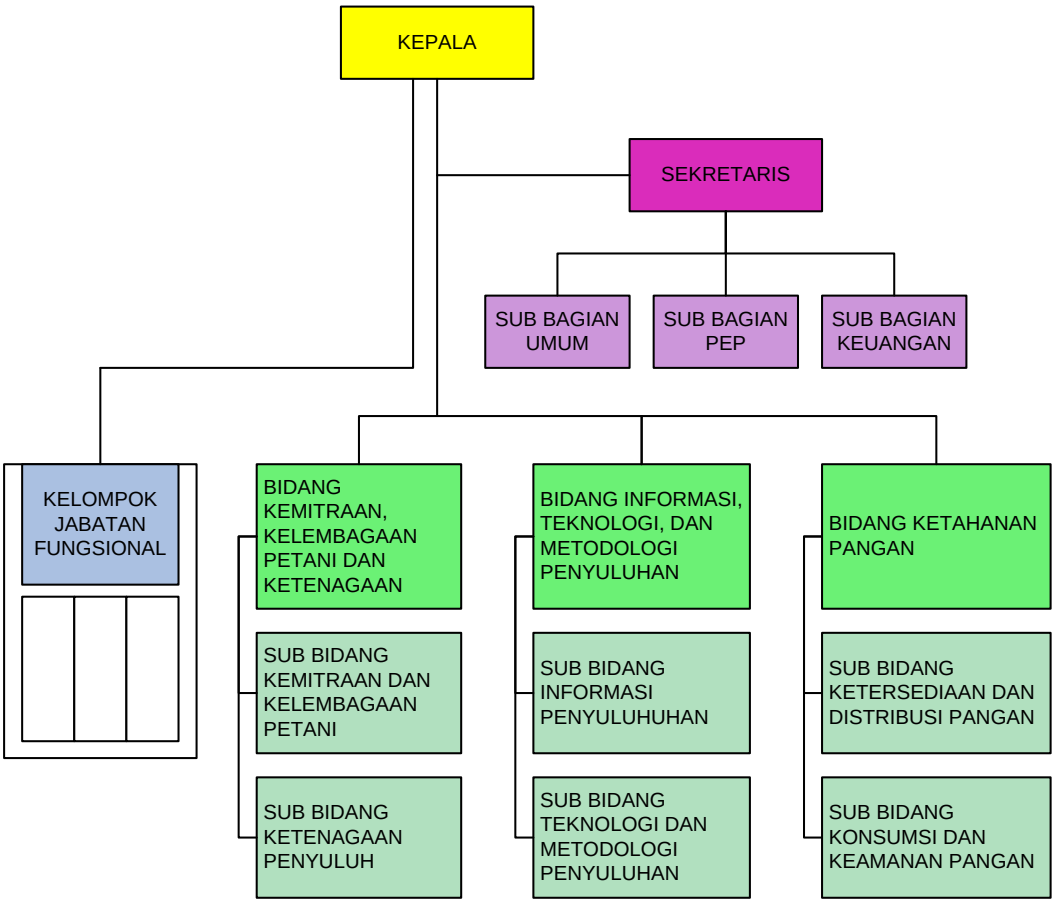
Tabel 2.1. Tujuan Dinas Pangan

Tujuan	Sasaran
Meningkatkan kompetensi SDM Penyuluhan sehingga memiliki integritas yang tinggi serta memiliki wawasan global	Terfasilitasnya penyelenggaraan pendidikan dan pelatihan bagi penyuluh dengan berpedoman kepada standar, akreditasi serta pola pendidikan dan pelatihan penyuluh yang sesuai dengan peraturan
	Peningkatan frekuensi kunjungan dan penyuluhan kepada kelompok tani
Meningkatkan mobilitas penyuluh dan aksesibilitas terhadap informasi dan teknologi melalui penyempurnaan sarana dan prasarana penyuluhan	Tersedianya sarana penyuluhan yang diperlukan oleh penyuluh sesuai dengan kebutuhan setempat, spesifik dan tepat sasaran

Tujuan	Sasaran
	Tersedianya informasi terkini dalam hal inovasi teknologi, kebijakan serta perubahan kecenderungan global, sehingga dapat cepat beradaptasi dan mengambil keputusan yang tepat
Mengembangkan dan meningkatkan kualitas kelembagaan penyuluhan supaya andal dengan menerapkan metodologi yang tepat	Terwujudnya Dinas Pangan sebagai penentu kebijakan dan program penyuluhan di tingkat kabupaten
	Terwujudnya Dinas Pangan sebagai pelaksana penyuluhan yang kompeten, multi-sektoral, dan dipercaya oleh pelaku utama dan pelaku usaha
Meningkatkan kelas kelompok tani agar terbentuk usaha tani yang maju	Meningkatnya kemampuan pengelolaan kelompok tani
Terciptanya jalinan usaha yang kokoh dan mutualisme antara pelaku utama dan pelaku usaha	Terbentuknya Lembaga Ekonomi Petani sebagai basis pengembangan agribisnis pedesaan
Mewujudkan ketahanan pangan melalui pemantauan ketersediaan, distribusi, konsumsi, dan keamanan pangan dengan memberdayakan kelembagaan masyarakat	Meningkatnya kualitas pemantauan dan pengkajian ketahanan pangan melalui koordinasi yang sinergis dengan instansi terkait, peningkatan kemampuan aparat dan pelibatan lembaga masyarakat dalam pengelolaan ketahanan pangan

2.1.3 Struktur Organisasi

Dinas Pangan Kabupaten Majalengka dibentuk melalui Peraturan Daerah Nomor 10 Tahun 2009 tentang Pembentukan Perangkat Daerah Kabupaten Majalengka, maka telah ditetapkan Struktur Organisasi seperti tertera pada Gambar 2.1 berikut :



Gambar 2.1. Struktur Organisasi

2.1.4 Tugas dan Fungsi

Adapun tugas pokok dan fungsi masing-masing jabatan adalah sebagai berikut :

1. Kepala Dinas

Kepala Dinas yang berada di bawah dan bertanggung jawab kepada bupati yang mempunyai tugas pokok merumuskan, menyelenggarakan, membina, dan mengevaluasi penyusunan dan pelaksanaan kebijakan daerah dalam bidang ketahanan pangan, pertanian, perikanan, dan kehutanan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Kepala Dinas mempunyai fungsi:

a. Perumusan kebijakan teknis bidang ketahanan pangan, pertanian, perikanan, dan kehutanan;
b. Pemberian dukungan atas penyelenggaraan bidang ketahanan pangan, pertanian, perikanan, dan kehutanan;
c. Pembinaan, pelaksanaan tugas, dan evaluasi bidang ketahanan pangan, pertanian, perikanan, dan kehutanan;
d. Penyelenggaraan administrasi kesekretariatan; dan
e. Pelaksanaan tugas lain yang diberikan oleh Bupati sesuai dengan tugas fungsinya.

2. Sekretaris

Sekretaris yang berada di bawah dan bertanggung jawab kepada kepala dinas yang mempunyai tugas pokok merencanakan operasional, mengelola, mengoordinasikan, mengendalikan, mengevaluasi dan melaporkan urusan umum, keuangan perencanaan, evaluasi dan pelaporan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sekretaris Dinas mempunyai fungsi:

a. Perencanaan operasional urusan umum, keuangan serta pengelolaan perencanaan, evaluasi dan pelaporan;
b. Pengelolaan urusan umum, keuangan, perencanaan, evaluasi dan pelaporan;
c. Pengendalian, evaluasi dan pelaporan urusan umum, keuangan serta pengelolaan perencanaan, evaluasi dan pelaporan;
d. Pengoordinasian urusan umum, keuangan, perencanaan, evaluasi dan pelaporan; dan
e. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan tugas dan fungsinya.

Dalam melaksanakan tugasnya, Sekretaris dibantu oleh:

a. Sub Bagian Umum;
b. Sub Bagian Keuangan;
c. Sub Bagian Perencanaan, Evaluasi dan Pelaporan.

3. Sub Bagian Umum

Sub Bagian Umum dipimpin oleh seorang kepala sub bagian yang berada di bawah dan bertanggung jawab kepada sekretaris dinas yang mempunyai tugas pokok merencanakan kegiatan, melaksanakan, membagi tugas, dan mengontrol urusan tata warkat, kepegawaian, kehumasan dan dokumentasi, pelengkapan, pembekalan dan keperluan tulis, dan ruang perkantoran.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bagian Umum mempunyai fungsi:

a. Perencana kegiatan urusan tata warkat, kepegawaian, kehumasan dan dokumentasi, perlengkapan, perbekalan dan keperluan tulis, dan ruang perkantoran;
b. Pelaksanaan urusan tata warkat, kepegawaian, kehumasan dan dokumentasi, perlengkapan, perbekalan dan keperluan tulis, dan ruang perkantoran;
c. Pembagian pelaksanaan tugas urusan tata warkat, kepegawaian, kehumasan dan dokumentasi, perlengkapan, dan keperluan tulis, dan ruang perkantoran; dan
d. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan tugas dan fungsinya.

4. Sub Bagian Keuangan

Sub Bagian Keuangan dipimpin oleh seorang kepala sub bagian yang berada di bawah dan bertanggung jawab kepada sekretaris dinas yang mempunyai tugas pokok merencanakan kegiatan, melaksanakan, membagi tugas, dan mengontrol urusan administrasi keuangan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bagian Keuangan mempunyai fungsi:

a. Perencanaan kegiatan administrasi keuangan;
b. Pelaksanaan administrasi keuangan;
c. Pembagian pelaksanaan tugas administrasi keuangan; dan
d. Pelaksanaan tugas lain yang di berikan oleh atasan sesuai dengan tugas dan fungsinya.

5. Sub Bagian Perencanaan, Evaluasi dan Pelaporan

Sub Bagian Perencanaan, Evaluasi dan Pelaporan dipimpin oleh seorang kepala sub bagian yang berada di bawah dan bertanggung jawab kepada sekretaris dinas yang mempunyai tugas pokok merencanakan kegiatan, melaksanakan, membagi tugas, dan mengontrol urusan perencanaan, evaluasi dan pelaporan yang meliputi penghimpunan rencana program/kegiatan, evaluasi dan laporan dari masing-masing bidang.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bagian Perencanaan Evaluasi, dan Pelaporan mempunyai fungsi:

a. Perencanaan kegiatan perencanaan, evaluasi dan pelaporan yang meliputi penghimpunan rencana program atau kegiatan, evaluasi dan laporan dari masing-masing bidang;
b. Pelaksanaan perencanaan evaluasi dan pelaporan yang meliputi penghimpunan rencana program/kegiatan, evaluasi dan laporan dari masing-masing bidang;
c. Pembagian pelaksanaan tugas perencanaan, evaluasi dan pelaporan yang meliputi penghimpunan rencana program/kegiatan, evaluasi dan laporan dari masing-masing bidang; dan
d. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan tugas dan fungsinya.

6. Bidang Kemitraan, Kelembagaan Petani dan Ketenagaan Penyuluhan

Bidang Kemitraan, Kelembagaan Petani dan Ketenagaan Penyuluhan dipimpin oleh seorang kepala bidang yang berada di bawah dan bertanggung jawab kepada kepala dinas yang mempunyai tugas pokok melaksanakan sebagian fungsi di bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Bidang Kemitraan, Kelembagaan Petani dan Ketenagaan Penyuluhan mempunyai fungsi:

a. Penyusunan rencana dan program kerja bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan;
b. Pembinaan perencanaan dan penyusunan programa;
c. Pelaksanaan koordinasi dan kerja sama di bidang penyuluhan dan ketahanan pangan;
d. Pembinaan *monitoring* dan evaluasi kegiatan bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan;
e. Pelaksanaan penyusunan laporan bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan;
f. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan;
g. Pelaporan hasil pelaksanaan tugas di bidang kemitraan, kelembagaan petani dan ketenagaan penyuluhan; dan
h. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

Dalam melaksanakan tugasnya, Kepala Bidang Kemitraan, Kelembagaan Petani dan Ketenagaan Penyuluhan dibantu oleh:

a. Sub Bidang Kemitraan dan Kelembagaan Petani; dan
b. Sub Bidang Ketenagaan Penyuluhan.

7. Sub Bidang Kemitraan dan Kelembagaan Petani

Sub Bidang Kemitraan dan Kelembagaan Petani dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Kemitraan dan Kelembagaan Petani dalam hal program dan kerja sama.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Kemitraan dan Kelembagaan Petani mempunyai fungsi:

a. Penyusunan rencana dan program kerja sub bidang kemitraan dan kelembagaan petani;
b. Pengumpulan, pengolahan, dan penyusunan data sebagai bahan bidang kemitraan dan kelembagaan petani;
c. Pelaksanaan koordinasi penyusunan bidang kemitraan dan kelembagaan petani;
d. Pelaksanaan bidang kemitraan dan kelembagaan petani;
e. Pelaksanaan bidang kemitraan dan kelembagaan petani hasil pelaksanaan tugas dalam hal kemitraan dan kelembagaan petani;
f. Pelaporan hasil pelaksanaan tugas dalam hal kemitraan dan kelembagaan petani; dan
g. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

8. Sub Bidang Ketenagaan Penyuluhan

Sub Bidang Ketenagaan Penyuluhan dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Ketenagaan Penyuluhan dalam hal ketenagaan dan penyuluhan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Ketenagaan Penyuluhan mempunyai fungsi:

a. Penyusunan rencana dan program kerja dalam hal ketenagaan penyuluhan;

b. Pengumpulan, pengolahan, dan penyusunan data sebagai bahan ketenagaan penyuluhan;
c. Pelaksanaan pengembangan ketenagaan penyuluhan;
d. Pelaksanaan koordinasi dan kerja sama dalam hal ketenagaan penyuluhan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
e. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas dalam hal ketenagaan penyuluhan;
f. Pelaporan hasil pelaksanaan tugas dalam hal ketenagaan penyuluhan; dan
g. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

9. Bidang Informasi, Teknologi, dan Metodologi Penyuluhan

Bidang Informasi, Teknologi, dan Metodologi Penyuluhan dipimpin oleh seorang kepala bidang yang berada di bawah dan bertanggung jawab kepada kepala dinas yang mempunyai tugas pokok melaksanakan sebagian fungsi di bidang informasi, teknologi , dan Metodologi Penyuluhan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Bidang Informasi, Teknologi, dan Metodologi Penyuluhan mempunyai fungsi:

a. Penyusunan program dan rencana kerja di bidang informasi, teknologi, dan metodologi penyuluhan;
b. Pembinaan dan pengembangan informasi, teknologi, dan metodologi penyuluhan;
c. Pelaksanaan koordinasi dan kerja sama di bidang informasi, teknologi, dan metodologi penyuluhan dengan unit kerja lain di lingkungan dinas dan di luar dinas;

d. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas di bidang informasi, teknologi, dan metodologi penyuluhan;
e. Pelaporan hasil pelaksanaan tugas di bidang informasi, teknologi, dan metodologi penyuluhan; dan
f. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

Dalam melaksanakan tugasnya, Kepala Bidang Informasi, Teknologi dan Metodologi Penyuluhan dibantu oleh:

a. Sub Bidang Informasi Penyuluhan; dan
b. Sub Bidang Teknologi dan Metodologi Penyuluhan.

10. Sub Bidang Informasi Penyuluhan

Sub Bidang Informasi Penyuluhan dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Informasi, Teknologi, dan Metodologi Penyuluhan dalam hal informasi penyuluhan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Informasi Penyuluhan mempunyai fungsi:

a. Penyusunan program dan rencana kerja dalam hal informasi penyuluhan;
b. Pelaksanaan koordinasi dan kerja sama dalam hal informasi penyuluhan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
c. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas dalam hal informasi penyuluhan;
d. Pelaporan hasil pelaksanaan tugas dalam hal informasi penyuluhan; dan
e. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

11. Sub Bidang Teknologi dan Metodologi Penyuluhan

Sub Bidang Teknologi dan Metodologi Penyuluhan dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Informasi, Teknologi, dan Metodologi Penyuluhan dalam hal teknologi dan metodologi penyuluhan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Teknologi dan Metodologi Penyuluhan mempunyai fungsi:

a. Penyusunan program dan rencana kerja dalam hal teknologi dan metodologi penyuluhan;
b. Pengembangan teknologi dan metodologi penyuluhan;
c. Pelaksanaan koordinasi dan kerja sama dalam hal teknologi dan metodologi penyuluhan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
d. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas dalam hal teknologi dan metodologi penyuluhan;
e. Pelaporan hasil pelaksanaan tugas dalam hal teknologi dan metodologi penyuluhan; dan
f. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

12. Bidang Ketahanan Pangan

Bidang Ketahanan Pangan dipimpin oleh seorang kepala bidang yang berada di bawah dan bertanggung jawab kepada kepala dinas yang mempunyai tugas pokok melaksanakan sebagian fungsi di Bidang Ketahanan Pangan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Bidang Ketahanan Pangan mempunyai fungsi:

a. Penyusunan rencana dan program kerja di Bidang Ketahanan Pangan;
b. Pembinaan dan pengembangan Bidang Ketahanan Pangan;

c. Pelaksanaan koordinasi dan kerja sama di Bidang Ketahanan Pangan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
d. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas di Bidang Ketahanan Pangan;
e. Pelaporan hasil pelaksanaan tugas di Bidang Ketahanan Pangan; dan
f. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

Dalam melaksanakan tugasnya, Kepala Bidang Ketahanan Pangan dibantu oleh:

a. Sub Bidang Ketersediaan dan Distribusi Pangan; dan
b. Sub Bidang Konsumsi dan Keamanan Pangan.

13. Sub Bidang Ketersediaan dan Distribusi Pangan

Sub Bidang Ketersediaan dan Distribusi Pangan dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Ketahanan Pangan dalam hal ketersediaan dan distribusi pangan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Ketersediaan dan Distribusi Pangan mempunyai fungsi:

a. Penyusunan rencana dan program kerja dalam hal ketersediaan dan distribusi pangan;
b. Pengumpulan, pengolahan, dan penyusunan data sebagai bahan pengelolaan ketersediaan dan distribusi pangan;
c. Pelaksanaan pengelolaan ketersediaan dan distribusi pangan;
d. Pelaksanaan koordinasi dan kerja sama dalam hal ketersediaan dan distribusi pangan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
e. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas dalam hal ketersediaan dan distribusi pangan;

f. Pelaporan hasil pelaksanaan tugas dalam hal ketersediaan dan distribusi pangan; dan

g. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

14. Sub Bidang Konsumsi dan Keamanan Pangan

Sub Bidang Konsumsi dan Keamanan Pangan dipimpin oleh seorang kepala sub bidang yang berada di bawah dan bertanggung jawab kepada kepala bidang yang mempunyai tugas pokok melaksanakan sebagian fungsi Bidang Ketahanan Pangan dalam hal konsumsi dan keamanan pangan.

Dalam menyelenggarakan tugas pokok sebagaimana dimaksud, Sub Bidang Konsumsi dan Keamanan Pangan mempunyai fungsi:

a. Penyusunan rencana dan program kerja dalam hal konsumsi dan keamanan pangan;
b. Pengumpulan, pengolahan, dan penyusunan data sebagai bahan pengembangan konsumsi dan keamanan pangan;
c. Pelaksanaan pengembangan konsumsi dan keamanan pangan;
d. Pelaksanaan sosialisasi konsumsi dan keamanan pangan;
e. Pelaksanaan koordinasi dan kerja sama dalam hal konsumsi dan keamanan pangan dengan unit kerja lain di lingkungan dinas dan di luar dinas;
f. Pelaksanaan *monitoring* dan evaluasi hasil pelaksanaan tugas dalam hal konsumsi dan keamanan pangan;
g. Pelaporan hasil pelaksanaan tugas dalam hal konsumsi dan keamanan pangan; dan
h. Pelaksanaan tugas lain yang diberikan oleh atasan sesuai dengan bidang tugasnya.

2.1.5 Logo

Logo Dinas Pangan merupakan logo dari Pemerintah Kabupaten Majalengka yang dapat dilihat pada Gambar 2.2 berikut.



Gambar 2.2. Logo Dinas Pangan Kabupaten Majalengka

Bentuk logo berupa sebuah Perisai bersudut lima bersisi hijau muda, dasar hijau muda yang di tengah-tengahnya terdapat lukisan yang terdiri dari 9 (sembilan) macam wujud benda yaitu :

1. Batang tanpa dahan berwarna hitam putih.
2. Selendang berwarna biru muda bersisi putih bertuliskan Sindangkasih Sugih Mukti warna putih.
3. Air/sungai berwarna puti dan biru muda.
4. Bangunan 3 (tiga) suhunan berwarna kuning tua bergaris sisi hitam dan putih sejajar.
5. Gunung berwarna biru.
6. Padi berwarna kuning bergaris sisi hitam.
7. Kapas berwarna putih kuning bergaris sisi hitam.
8. Kompas/mata angin berwarna hitam kuning.
9. Pita merah putih yang mengelilingi 9 (sembilan) wujud benda.

2.2 Landasan Teori

Landasan teori memberikan gambaran dan teori yang terkait dengan pembangunan sistem. Untuk membangun sebuah sistem , maka perlu dikemukakan hal-hal atau teori yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai landasan dalam pembuatan laporan tugas akhir ini.

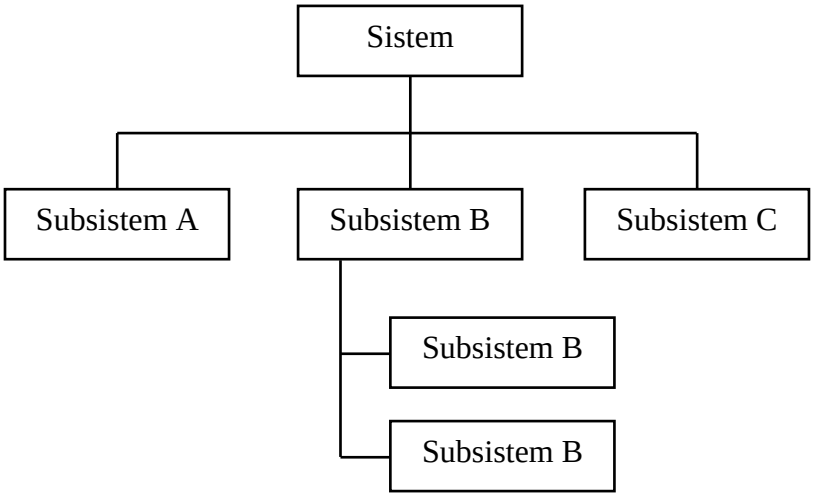
2.2.1 Sistem

Definisi dari sistem dapat dilakukan dengan 2 pendekatan, yaitu pendekatan prosedur dan pendekatan komponen. Dengan pendekatan prosedur sistem didefinisikan sebagai kumpulan dari beberapa prosedur yang mempunyai tujuan tertentu. Sebagai contoh sistem informasi penjualan barang. Sistem ini terdiri dari beberapa prosedur, misalnya prosedur persediaan barang, prosedur pemesanan, prosedur pelayanan konsumen dan prosedur pemesanan barang. Sedangkan dengan pendekatan komponen, sistem merupakan kumpulan dari komponen-komponen yang saling berkaitan untuk mencapai tujuan tertentu. Contoh dari definisi ini adalah sistem komputer. Sistem komputer merupakan gabungan dari beberapa komponen perangkat keras, perangkat lunak dan sistem operasi. Kedua definisi ini dapat digabungkan untuk memperoleh pengertian yang mendalam mengenai arti sesungguhnya dari sistem [10].

Suatu sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan [11]. Subsistem sebenarnya merupakan sebuah sistem yang berada di dalam sistem. Contohnya adalah sistem yang membangun sistem mobil atau otomotif. Mobil yang digambarkan sebagai suatu sistem, terdiri dari subsistem mesin, subsistem kelistrikannya, subsistem rangka atau badan mobil. Subsistem sistem mesin juga terdiri dari gabungan beberapa subsistem karburator, bahan bakar, dan lain-lain. Gambar 2.3. menunjukkan hubungan antara sistem dan subsistem.

Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yaitu mempunyai komponen-komponen (*componen*), batas sistem (*boundary*),

lingkungan luar sistem (*environments*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*), dan sasaran (*objectives*) atau tujuan (*goal*).

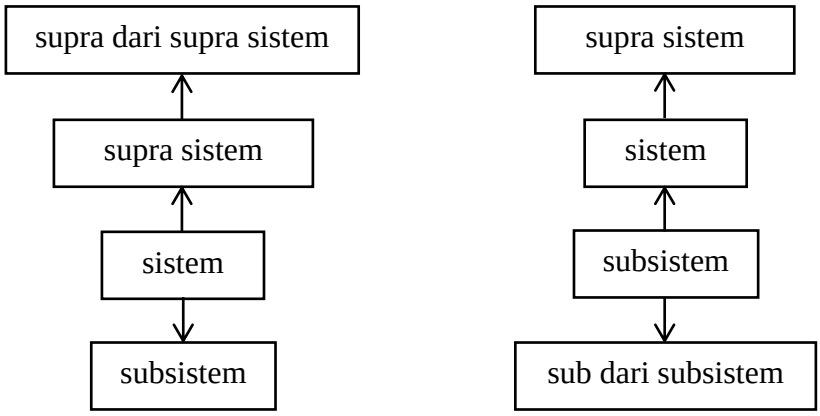


Sumber gambar : Sistem Informasi [10]

Gambar 2.3. Subsistem pada Sistem

1. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap subsistem mempunyai sifat-sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu yang lebih besar yang disebut dengan *supra system*. Berikut ini Gambar 2.4. merupakan susunan dari sistem.



Sumber gambar : Pengenalan Komputer **[11]**

Gambar 2.4. Subsistem, sistem, dan supra sistem

2. Batas Sistem

Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

Lingkungan luar dari suatu sistem adalah apa pun di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak maka akan dapat mengganggu kelangsungan hidup dari sistem.

4. Penghubung Sistem

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya. Keluaran (*output*) dari satu sistem akan menjadi masukan (*input*) untuk subsistem yang lainnya dengan melalui

penghubung. Dengan penghubung satu subsistem dapat berinteraksi dengan subsistem yang lainnya yang membentuk satu kesatuan.

5. Masukan Sistem

Masukan adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh di dalam sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan data adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem

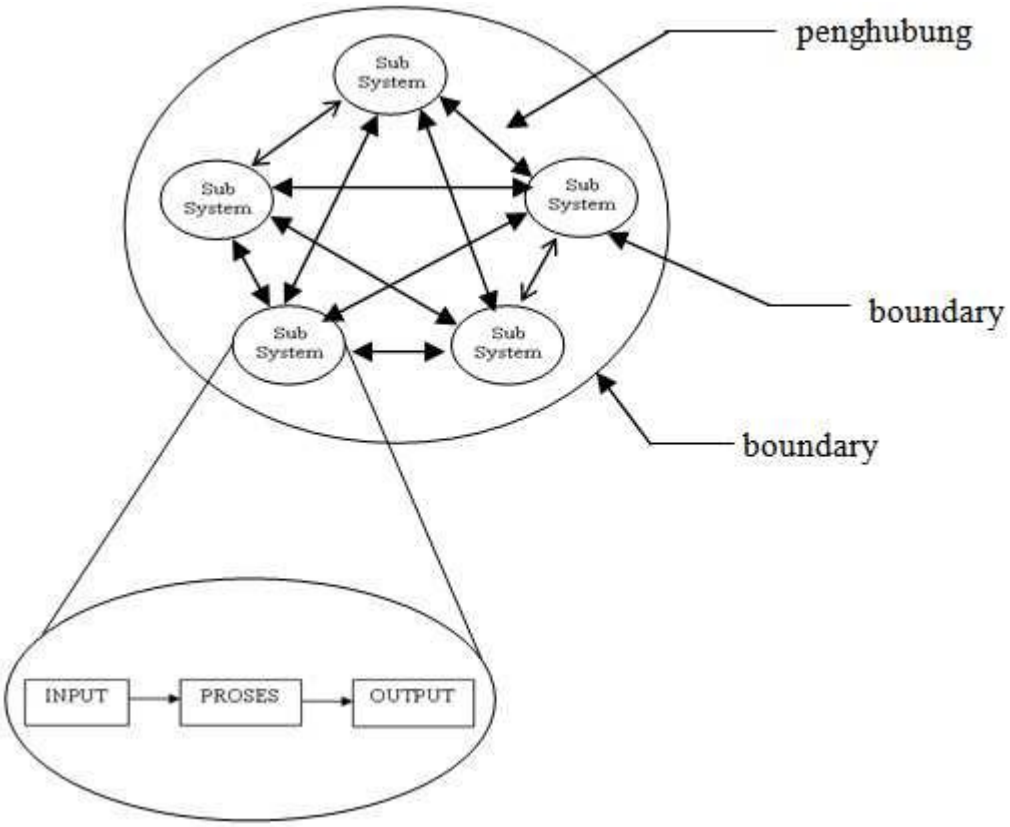
Keluaran adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukan untuk subsistem yang lain atau kepada supra sistem. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna merupakan hasil sisa pembuangan, sedangkan informasi adalah keluaran yang dibutuhkan.

7. Pengolah Sistem

Suatu sistem dapat mempunyai suatu bagian pengolahan atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan mengubah masukan menjadi keluaran. Suatu sistem produksi akan mengolah masukan berupa bahan baku dan bahan-bahan yang lain akan menjadi keluaran berupa barang jadi. Contoh, sistem akuntansi akan mengolah data-data transaksi menjadi laporan-laporan keuangan dan laporan-laporan lain yang dibutuhkan oleh manajemen.

8. Sasaran Sistem

Suatu sistem pasti mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.



Sumber gambar : Pengenalan Komputer [11]

Gambar 2.5. Karakteristik Suatu Sistem

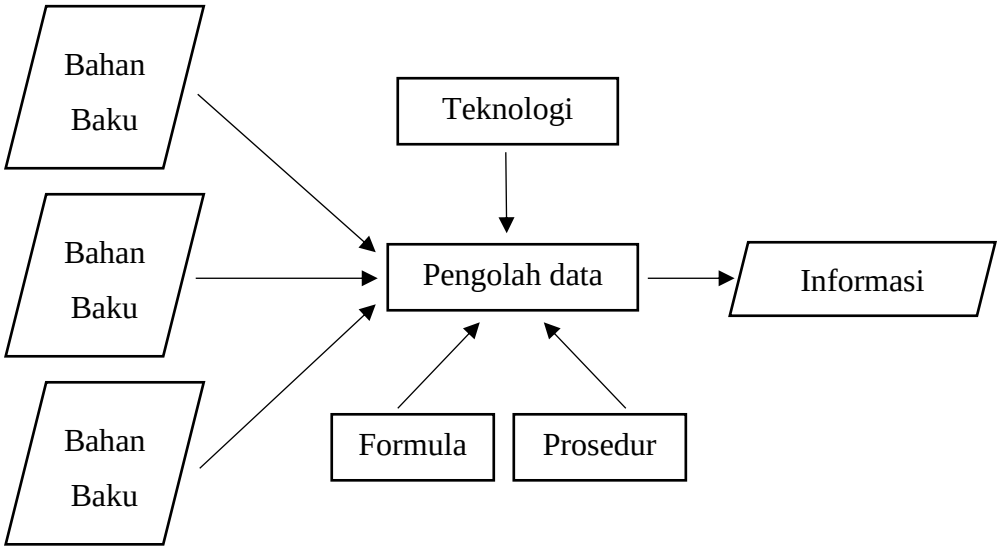
Sistem yang dibangun dalam penelitian ini terdiri dari dua subsistem, yaitu subsistem pada platform *website* yang digunakan oleh administrator untuk mengelola data dokumentasi kegiatan kunjungan lapangan dan data pengguna sistem dan subsistem pada platform *mobile* yang digunakan oleh petugas lapangan untuk melakukan proses dokumentasi kegiatan kunjungan lapangan dan melakukan proses analisis kebutuhan nitrogen pada taman padi.

2.2.2 Informasi

Informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian (*event*) yang nyata (*fact*) yang digunakan untuk pengambilan keputusan.

Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal atau data-idem. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah sesuatu yang terjadi pada saat yang tertentu. Kesatuan nyata (*fact*) adalah berupa suatu obyek nyata seperti tempat, benda dan orang yang betul-betul ada dan terjadi [11].

Data dapat berupa angka-angka atau kode-kode tertentu. Data berupa angka atau wujud yang lain belum mempunyai arti bagi penggunanya. Untuk mempunyai arti maka data tersebut diolah dengan menggunakan sebuah pengolah data. Data yang masih berupa bahan mentah (*raw*) ini diolah dengan menggunakan prosedur-prosedur tertentu akan menjadi sebuah informasi bagi penggunanya. Singkatnya, informasi adalah data yang telah diolah dan mempunyai arti bagi pengguna [10].



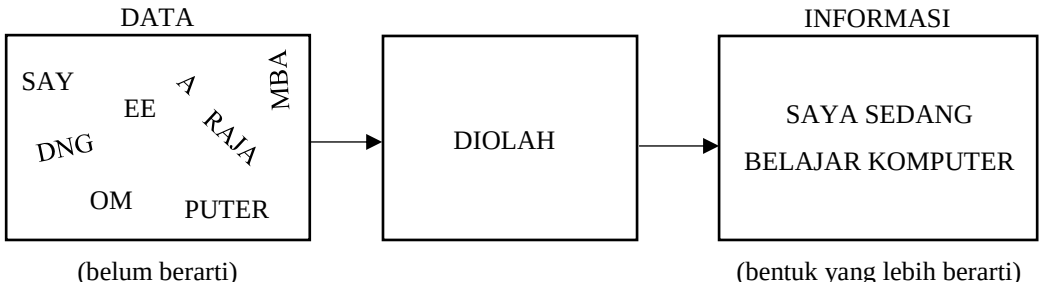
Sumber gambar: Sistem Informasi **[10]**

Gambar 2.6. Pengolahan Data

2.2.2.1 Siklus Informasi

Data merupakan bentuk yang masih mentah, belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model untuk dihasilkan informasi. Data dapat berbentuk simbol-simbol semacam huruf-huruf

atau alfabet, angka-angka, bentuk-bentuk suara, sinyal-sinyal, gambar-gambar dan sebagainya.



Sumber gambar : Pengenalan Komputer [11]

Gambar 2.7. Data Yang Diolah Menjadi Informasi

2.2.2.2 Kualitas Informasi

Kualitas dari suatu informasi tergantung dari tiga hal, yaitu informasi harus akurat, tepat pada waktunya dan relevan [11].

Akurat, berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan bagi orang yang menerima informasi tersebut. Selain itu juga berarti informasi harus jelas mencerminkan maksudnya.

Tepat pada waktunya, informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi. Karena informasi merupakan landasan di dalam pengambilan keputusan. Saat ini mahalnya nilai informasi disebabkan harus cepatnya informasi tersebut didapat, sehingga diperlukan teknologi-teknologi mutakhir untuk mendapatkan, mengolah dan menerimanya.

Relevan, berarti informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda. Misalnya informasi mengenai sebab-musabab kerusakan mesin produksi kepada akuntan perusahaan adalah kurang relevan dan akan lebih relevan apabila ditujukan kepada ahli teknik perusahaan.

2.2.3 Aplikasi

Aplikasi pada dasarnya berasal dari bahasa inggris yaitu dari kata *application* yang berarti penerapan ataupun penggunaan. Jika di tinjau secara istilah, aplikasi merupakan suatu program yang secara sengaja di buat dan telah siap untuk dipakai *user* untuk melakukan suatu fungsi atau tugas guna mencapai sebuah sasaran yang di tuju.

Adapun beberapa pengertian menurut para ahli di antaranya [12] :

1. Menurut Jogiyanto, aplikasi merupakan penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output*.
2. Menurut Rachmad Hakim S, aplikasi adalah perangkat lunak yang digunakan untuk tujuan tertentu, seperti mengolah dokumen, mengatur Windows, permainan (*game*) dan sebagainya.
3. Menurut Harip Santoso, aplikasi adalah suatu kelompok file (*form*, *class*, *report*) yang bertujuan untuk melakukan aktivitas tertentu yang saling terkait, misalnya aplikasi *payroll*, aplikasi *fixed asset*, dll.

2.2.4 Aplikasi Mobile

Aplikasi Mobile adalah sebuah aplikasi yang memungkinkan penggunanya melakukan mobilitas dengan menggunakan perlengkapan seperti PDA, telepon seluler atau *Handphone*. Dengan menggunakan aplikasi *mobile*, pengguna dapat dengan mudah melakukan berbagai macam aktivitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya.

Pemanfaatan aplikasi *mobile* untuk hiburan paling banyak digemari oleh hampir 70% pengguna telepon seluler, karena dengan memanfaatkan adanya fitur *game*, *music player*, sampai *video player* membuat kita menjadi semakin mudah menikmati hiburan kapan saja dan di mana pun [13].

Menurut Buyens [14] aplikasi *mobile* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, dan penggunaan. Secara

istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju. Sedangkan *mobile* dapat diartikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Sehingga dapat diartikan bahwa aplikasi *mobile* merupakan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah-pindah dari satu tempat ke tempat yang lain pada suatu perangkat yang berukuran cukup kecil. Aplikasi *mobile* ini dapat diakses melalui perangkat PDA, telepon seluler, *smartphone*, Tablet PC, dan perangkat sejenisnya.

2.2.5 Jenis dan Kategori Aplikasi Mobile

Untuk keberhasilan dalam membangun aplikasi *mobile* (*mobile application development*), sangat penting untuk mengetahui dan paham tentang apa aplikasi *mobile* sebenarnya dan bagaimana perbedaan interaksi yang pengguna lakukan dengan aplikasi *dekstop*. Tabel 2.2. dan Tabel 2.3. Kategori Aplikasi Mobile di bawah ini menjelaskan berbagai jenis dan kategori yang terlibat dalam pembangunan aplikasi *mobile* [15].

Tabel 2.2. Jenis Aplikasi Mobile

No.	Jenis	Deskripsi
1	Aplikasi Akses Browser	Aplikasi ini tidak pasang pada perangkat dan dapat diakses melalui browser dengan memasukkan URL dari web. Ukuran memori perangkat tidak penting karena data aplikasi tidak disimpan dalam perangkat. Ini tergantung pada kualitas browser. Misalnya, m.yahoo.com, www.google.com.
2	Aplikasi *Native*	Aplikasi yang diinstal pada perangkat. Mereka tidak memerlukan transfer data ke server dan bekerja di perangkat tanpa jaringan karena data tentang aplikasi disimpan dalam perangkat itu sendiri. Misalnya, Catatan dan *Reminder* di iPhone.

No.	Jenis	Deskripsi
3	Aplikasi *Hybrid* (Web)	Aplikasi yang dipasang di perangkat dan selalu membutuhkan koneksi internet untuk berjalan dan berfungsi. Misalnya, Aplikasi Jaringan Sosial (*Facebook, Twitter*), *Instant Messenger* (*Skype*), *eCommerce* (*Flipkart*), Testing Internet Kecepatan (*Speedtest*).
4	Aplikasi *Hybrid* (*Mixed*)	Aplikasi yang dipasang di perangkat yang mungkin memerlukan atau mungkin tidak memerlukan koneksi internet untuk berjalan dan berfungsi. Misalnya, aplikasi medis dan beberapa permainan yang bisa dimainkan sendiri, *offline* dan juga *online* untuk bermain dengan beberapa pemain.

Tabel 2.3. Kategori Aplikasi Mobile

No.	Kategori	Deskripsi
1	Komunikasi	E-Mail, IM, jaringan sosial, *mobile* / internet browser, Berita / Informasi, portal pada perangkat (portal Java).
2	Permainan	Teka-teki / Strategi, Kartu / Kasino, Aksi / Petualangan, Olahraga, Hiburan Olahraga.
3	Multimedia	Grafis / Penampil gambar, Penampil presentasi, Pemutar video, Pemutar audio, Pemutar *streaming*(Audio / Video).
4	Produktivitas	Kalender, Kalkulator, *Diary*, Notepad / Memo / Pemroses Kata, *Spreadsheet*, *Directory Services*, Perbankan dan keuangan, Rekaman panggilan, Pemantauan kesehatan *mobile*, *Mobile advertising*.
5	Perjalanan	Panduan kota, Konverter mata uang, GPS (layanan berbasis lokasi), *Maps*, Penerjemah, Rencana Perjalanan, Jadwal, Peramal cuaca.
6	Utilitas	*Profile Manager, Idle screens, Screensavers, Address book, Task manager, Call manager, File manager, Mobile search 7*
7	Pendidikan	Alfabet, Numerik

2.2.6 Karakteristik Perangkat Mobile

Berdasarkan survei yang dilakukan [15], mayoritas responden survei berpendapat bahwa aplikasi *mobile* secara signifikan berbeda dari aplikasi desktop mereka dalam beberapa hal walaupun memiliki kesamaan mencolok. Banyak responden menyatakan bahwa ponsel berkinerja sebagai komputer pribadi dan berisi informasi pribadi dari pemilik yang digunakan secara eksklusif oleh pemilik, tidak seperti PC yang mungkin berada di sebuah ruangan dan dapat digunakan bersama oleh berbagai individu. Selain itu, responden berpendapat bahwa ponsel juga bertindak sebagai agen, mengambil pesan atau memberitahukan pemilik pada peristiwa penting. Sejak ponsel adalah entitas pribadi dan selalu dengan pengguna, aplikasi yang dikembangkan harus sedemikian rupa sehingga selalu dapat berinteraksi, dan menyesuaikan dengan pengguna.

Berikut ini adalah karakteristik dari perangkat *mobile*.

1. Berukuran Kecil

Perangkat *mobile* memiliki ukuran yang kecil karenanya hanya membutuhkan ruang yang sedikit. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

2. Mekanisme Masukan

Mekanisme *input* manusia di aplikasi *mobile* berbeda dari aplikasi desktop karena fitur seperti sentuhan, cubitan, dan *swipe*. Disarankan bahwa aplikasi *mobile* harus *user friendly* dan *touch friendly* dengan opsi lebih banyak pilihan bersama dengan entri *keyboard* yang minimal.

3. Ukuran Layar

Karena ukuran layar ponsel kecil dan bervariasi dari satu perangkat ke perangkat lainnya, disarankan bahwa pengembang harus memilih sejumlah fitur yang mereka ingin perkenalkan ketika merancang aplikasi ponsel pintar.

4. Memori Terbatas

Perangkat *mobile* juga memiliki memori yang kecil, pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari *memory*, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang mahal ini.

5. Daya Proses Terbatas

Sistem *mobile* tidaklah setangguh sistem desktop. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini.

2.2.7 Android

Android adalah sistem operasi *mobile* yang didasarkan pada versi dimodifikasi dari *Linux*. Android pada awalnya dikembangkan oleh *startup* dengan nama yang sama, Android, Inc. Pada tahun 2005, sebagai bagian dari strategi untuk memasuki ruang *mobile*, Google membeli Android dan mengambil alih pekerjaan pembangunan (beserta tim pengembangan).

Google ingin Android menjadi terbuka dan bebas; oleh karena itu, sebagian besar kode Android dirilis di bawah *open source Apache License*, yang berarti bahwa siapa pun yang ingin menggunakan Android dapat melakukannya dengan mengunduh penuh kode sumber Android. Selain itu, vendor (biasanya produsen *hardware*) dapat menambahkan ekstensi milik mereka sendiri untuk Android dan menyesuaikan Android untuk membedakan produk mereka dari orang lain. Model pembangunan yang sederhana ini membuat Android sangat menarik dan telah mengusik minat banyak vendor.

Keuntungan utama dari mengadopsi Android adalah menawarkan pendekatan terpadu untuk pengembangan aplikasi. Pengembang hanya perlu mengembangkan untuk Android, dan aplikasi mereka dapat dijalankan pada berbagai perangkat yang berbeda, asalkan perangkat yang didukung menggunakan Android [16].

2.2.7.1 Arsitektur Android

Untuk memahami bagaimana kerja Android, dapat dilihat pada Gambar 2.8. yang menunjukkan berbagai lapisan yang membentuk sistem operasi Android. Sistem operasi Android ini dibagi menjadi lima bagian dalam empat lapisan utama, yaitu:

1. *Linux kernel*

   Bagian ini adalah lapisan paling bawah, yang merupakan kernel yang menjadi basis dari Android. Lapisan ini berisi semua *driver* perangkat pada tingkat rendah untuk berbagai komponen *hardware* dari perangkat Android.

2. *Libraries*

   Bagian ini berisi semua kode yang menyediakan fitur-fitur utama dari OS Android. Misalnya, *library SQLite* memberikan dukungan *database* sehingga aplikasi dapat menggunakannya untuk penyimpanan data. *Library WebKit* menyediakan fungsionalitas untuk *web browsing*.

3. *Android runtime*

   Berada pada lapisan yang sama dengan *libraries*, *Runtime Android* menyediakan satu set inti *libraries* yang memungkinkan pengembang untuk menulis aplikasi Android menggunakan bahasa pemrograman Java. *Runtime Android* juga termasuk mesin virtual Dalvik, yang memungkinkan setiap aplikasi Android untuk berjalan dalam prosesnya sendiri, dengan contoh sendiri dari mesin virtual Dalvik (aplikasi Android dikompilasi ke dalam *executable Dalvik*). Mesin virtual Dalvik mengeksekusi file dengan format *Dalvik Executable* (.dex) yang dirancang untuk meminimalkan *memory footprint*. Mesin virtual Dalvik berbasis register, dan dapat menjalankan kelas-kelas yang dikompilasi dengan bahasa pemrograman Java dan ditransformasikan menjadi format .dex. Dalvik VM sendiri bergantung pada Kernel Linux untuk fungsi dasarnya seperti *threading* dan manajemen memori secara *low-level*. Dalvik adalah mesin virtual khusus yang dirancang secara khusus untuk Android dan dioptimalkan untuk perangkat *mobile* bertenaga baterai dengan memori dan CPU terbatas.

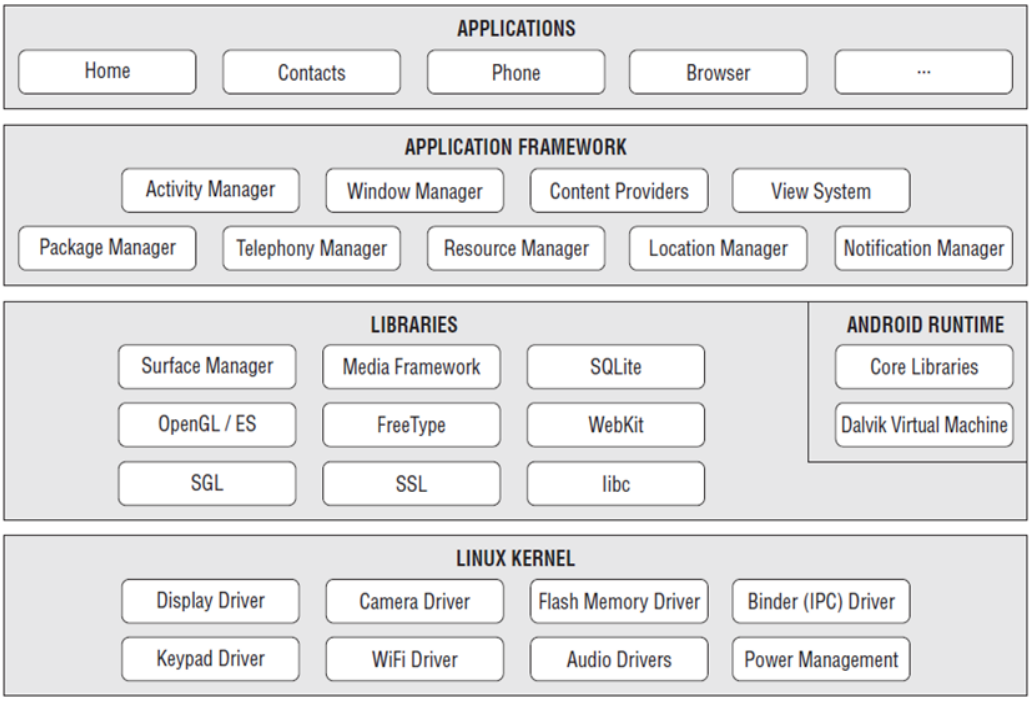
4. *Application framework*

Lapisan ini dirancang untuk memudahkan penggunaan komponen dari Android sendiri. Lapisan ini mengekspos berbagai kemampuan dari OS Android untuk pengembang aplikasi sehingga mereka dapat memanfaatkannya dalam aplikasi mereka.

5. *Applications*

Applications merupakan program yang langsung berhubungan dengan pengguna. Pada lapisan atas ini, Anda akan menemukan aplikasi bawaan dari perangkat Android (seperti Telepon, Kontak, Browser, dll.), serta aplikasi yang Anda unduh dan pasang dari *Android Market*. Setiap aplikasi yang Anda buat juga berada pada lapisan ini.

Berikut adalah gambar arsitektur pembentuk sistem operasi Android.



Sumber Gambar : Beginning Android™ 4 Application Development [16].

Gambar 2.8. Arsitektur Android

2.2.7.2 Versi Android

Telah banyak *smartphone* dan *Tablet PC* yang menggunakan sistem operasi Android dengan versi yang berbeda. Semakin tinggi versi, maka fiturnya pun semakin canggih dan banyak. Ponsel pertama yang memakai sistem operasi Android adalah T-Mobile G1 atau nama pasarnya adalah *HTC Dream* yang dirilis pada tanggal 22 Oktober 2008 [17].

Setiap versi utama dari Android memiliki julukan berbasis hidangan penutup (*dessert*), dan semua dalam urutan abjad. Android telah melalui cukup banyak pembaharuan sejak rilis pertama. Tabel 2.4. berikut menunjukkan berbagai versi Android beserta kode namanya [16] [18].

Tabel 2.4. Versi Android

Versi	Rilis	Kode Nama
1.1	9 Februari 2009	-
1.5	30 April 2009	*Cupcake*
1.6	15 September 2009	*Donut*
2.0/2.1	26 Oktober 2009	*Eclair*
2.2	20 Mei 2010	*Froyo*
2.3	6 Desember 2010	*Gingerbread*
3.0/3.1/3.2	22 Februari 2011	*Honeycomb*
4.0	19 Oktober 2011	*Ice Cream Sandwich*
4.1/4.2/4.3	9 Juli 2012	*Jelly Bean*
4.4	31 Oktober 2013	*KitKat*
5.0	12 November 2014	*Lollipop*
6.0	5 Oktober 2015	*Marshmallow*

Berikut penjelasan dari masing-masing versi yang tertera pada Tabel 2.4. di atas.

1. Android Versi 1.1

Pada 9 Maret 2009, *Google* merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan *Gmail*, dan pemberitahuan Email.

2. Android Versi 1.5 (*Cupcake*)

Cupcake adalah perbaikan besar pertama dari OS Android. Android 1.5 SDK (*Software Development Kit*) dirilis pada April 2009 dan membawa banyak perubahan UI (*User Interface*), yang terbesar adalah memungkinkan dukungan untuk *widget* dan folder pada *home screens*.

3. Android Versi 1.6 (*Donut*)

Donut, dirilis pada September 2009, dengan memperluas fitur yang datang pada Android 1.5. Di balik layar, *Donut* membawa dukungan untuk layar sentuh resolusi tinggi, kamera jauh lebih baik dan dukungan galeri.

4. Android Versi 2.0/2.1 (*Eclair*)

Eclair adalah langkah yang cukup besar di atas pendahulunya. Diperkenalkan pada akhir tahun 2009, Android 2.0 pertama kali muncul pada *Motorola Droid*, yang membawa perbaikan dalam browser, Google *Maps*, dan antarmuka pengguna baru.

5. Android Versi 2.2 (*Froyo : Frozen Yoghurt*)

Android 2.2 *Froyo* telah di umumkan pada bulan Mei 2010 pada saat konferensi Google IO di San Francisco. Perubahan tunggal terbesar adalah pengenalan *Just-In-Time Compiler* (JIT), yang secara signifikan mempercepat daya pengolahan telepon. Seiring dengan JIT, Android 2.2 juga membawa dukungan untuk *Adobe Flash* 10.1. yang berarti dapat memainkan permainan berbasis *Flash* di web browser Android ini.

Froyo juga membawa dukungan untuk *tethering*, yang berarti bisa menggunakan koneksi data *smartphone* Android untuk memberikan Internet (nirkabel atau dengan kabel USB) untuk hampir semua perangkat yang diinginkan.

6. Android Versi 2.3 (*Gingerbread*)

Android 2.3 *Gingerbread* dirilis pada bulan Desember 2010, dengan membawa beberapa peningkatan UI untuk Android, hal-hal seperti merasa lebih konsisten di menu dan dialog, dan sebuah bar notifikasi hitam yang baru, tapi masih terlihat dan terasa seperti Android yang biasa digunakan.

Gingerbread juga membawa dukungan untuk teknologi baru yaitu, NFC (*Near Field Communication*), SIP (panggilan Internet), efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*) dan dukungan format video VP8 dan WebM.

7. Android Versi 3.0 (*Honeycomb*)

Android 3.0 *Honeycomb* keluar pada bulan Februari 2011 dengan *Motorola Xoom*. Android *Honeycomb* dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi-prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis.

8. Android Versi 4.0 (*Ice Cream Sandwich*)

Tindak lanjut untuk *Honeycomb* diumumkan di Google IO Mei 2011 dan dirilis pada bulan Oktober 2011. Dijuluki *Ice Cream Sandwich* (ICS) dan akhirnya ditunjuk Android 4.0, ICS membawa banyak elemen desain *Honeycomb* untuk *smartphone*, dan juga menyempurnakan pengalaman pada *Honeycomb*. Perangkat pertama yang diluncurkan dengan ICS adalah *Samsung Galaxy Nexus*.

9. Android Versi 4.1 (*Jelly Bean*)

Pada bulan Juni 2012 Google mengumumkan Android versi 4.1 (*Jelly Bean*) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. *Jelly Bean* adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuka pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, *triple buffering*, perpanjangan waktu *vsync* dan peningkatan *frame rate* hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet *Nexus* 7, yang dirilis pada 13 Juli 2012.

10. Android Versi 4.4 (*KitKat*)

Google pada bulan September 2013 mengumumkan bahwa versi baru untuk Android akan diberi nama *KitKat*. Android versi 4.4 (*KitKat*) dirilis pada

bulan Oktober 2013. *KitKat* dibawa lebih ringan, datar dan tampilan yang jauh lebih berwarna untuk Android, tetapi lebih banyak perubahan di bagian lain. Seperti peluncur *Google Now*, dan integrasi SMS dengan *Hangouts*.

11. Android Versi 5.0 (*Lollipop*)

Google merilis Android 5.0 *Lollipop* dengan *Nexus* 6 dan *Nexus* 9, dan diantar dalam bahasa desain baru dan dukungan untuk perangkat 64-bit. Kebanyakan perubahan pada API baru selain dari fitur pada antar muka yang baru juga. Google sendiri telah memperbarui *Nexus* 5, *Nexus* 4 dan *Nexus* 7 untuk *Lollipop*, dan perusahaan lain seperti Motorola, Samsung, HTC dan LG telah relatif cepat mengikuti.

12. Android Versi 6.0 (*Marshmallow*)

Google merilis Android 6.0 *Marshmallow* dengan *Nexus* 6P dan *Nexus* 5X. Dalam *Marshmallow* terdapat beberapa perubahan visual, seperti peluncur baru dan panel notifikasi disempurnakan. *Marshmallow* memberikan kontrol yang lebih baik dalam hal perizinan, yang memungkinkan pengguna untuk mengontrol data bagian mana yang dapat di akses oleh aplikasi.

2.2.7.3 Komponen Aplikasi Android

Komponen aplikasi adalah blok pembangun penting dari aplikasi Android. Setiap komponen merupakan titik berbeda yang digunakan sistem untuk memasuki aplikasi yang dibuat. Masing-masing komponen ada sebagai kesatuan sendiri dan memainkan peran tertentu, yang masing-masing merupakan blok pembangun unik yang mendefinisikan perilaku aplikasi secara keseluruhan.

Ada empat macam tipe komponen aplikasi. Setiap tipe memiliki kegunaan tersendiri dan daur hidupnya sendiri yang mendefinisikan cara komponen dibuat dan dimusnahkan. Berikut ini empat tipe komponen aplikasi [19]:

1. *Activities*

Suatu *activity* adalah titik masukan untuk berinteraksi dengan pengguna. *Activity* mewakili satu layar dengan satu antarmuka pengguna. Misalnya, aplikasi email mungkin memiliki satu *activity* yang menampilkan daftar email baru, *activity*

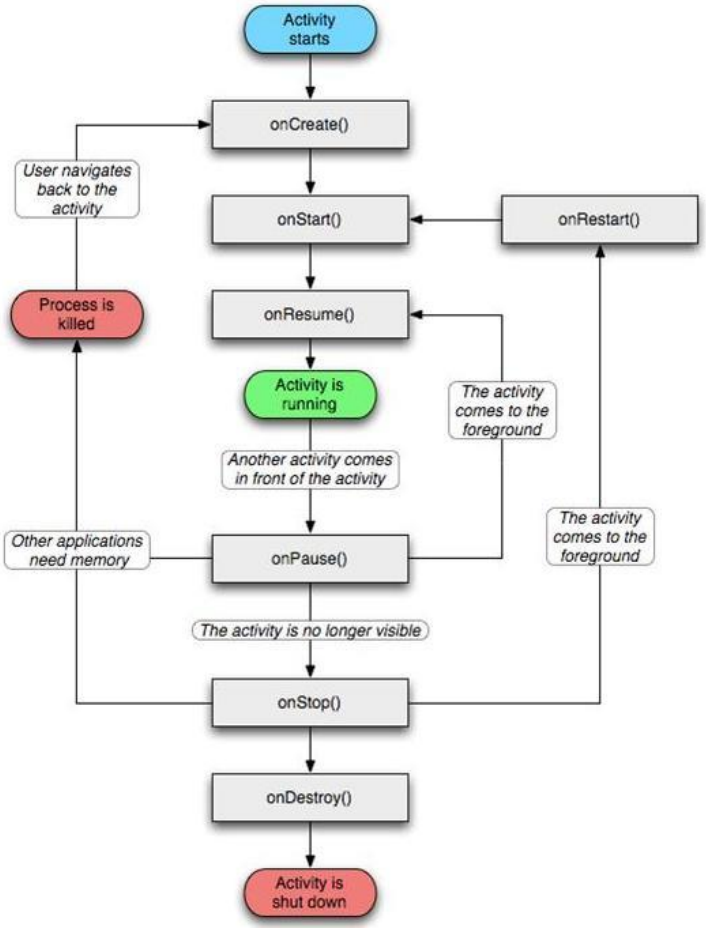
lain untuk menulis email, dan *activity* satunya lagi untuk membaca email. Walaupun semua *activity* bekerja sama untuk membentuk pengalaman pengguna yang kohesif dalam aplikasi email, masing-masing tidak saling bergantung. Karenanya, aplikasi berbeda bisa memulai salah satu *activity* ini (jika aplikasi email mengizinkannya). Misalnya, aplikasi kamera bisa memulai *activity* dalam aplikasi email yang membuat email baru agar pengguna bisa berbagi gambar [19].

Kelas dasar *activity* mendefinisikan serangkaian peristiwa yang mengatur siklus hidup dari suatu kegiatan. Kelas *activity* mendefinisikan peristiwa berikut [16]:

a. ***onCreate***, dipanggil ketika *activity* tersebut pertama dibuat. Gunakan metode *onCreate()* untuk membuat dan menginstansiasi objek yang akan digunakan dalam aplikasi.
b. ***onStart***, dipanggil ketika *activity* menjadi terlihat untuk pengguna.
c. ***onResume***, dipanggil ketika *activity* mulai berinteraksi dengan pengguna. Gunakan metode *onResume()* untuk memulai segala layanan atau kode yang perlu untuk dijalankan saat *activity* di latar depan.
d. ***onPause***, dipanggil ketika *activity* saat sedang berhenti dan *activity* sebelumnya sedang dilanjutkan. Gunakan metode *onPause()* untuk menghentikan layanan atau kode yang tidak perlu dijalankan saat *activity* tidak di latar depan.
e. ***onStop***, dipanggil ketika *activity* tidak lagi diperlihatkan di sisi pengguna.
f. ***onDestroy***, dipanggil sebelum *activity* dihancurkan oleh sistem (baik secara manual atau dengan sistem untuk menghemat memori).
g. ***onRestart***, dipanggil ketika aktivitas tersebut telah berhenti dan *restart* lagi.

Secara *default*, ketika *activity* diciptakan berisi *onCreate() event*. Dalam *event* ini adalah kode yang membantu untuk menampilkan elemen UI layar.

Berikut ini Gambar 2.9. menunjukkan siklus hidup dari suatu *activity* dan berbagai tahap yang dilalui, dari ketika aktivitas dimulai sampai berakhir.



Sumber gambar : Beginning Android™ 4 Application Development [16].

Gambar 2.9. Activity Lifecycle Android

2. *Services*

Service adalah komponen yang berjalan di latar belakang untuk melakukan operasi yang berjalan lama atau untuk melakukan pekerjaan bagi proses jarak jauh (*remote*). *Service* tidak menyediakan antarmuka pengguna. Misalnya, sebuah *service* bisa memutar musik di latar belakang sementara pengguna berada dalam aplikasi lain, atau *service* bisa menarik data lewat jaringan tanpa memblokir interaksi pengguna dengan *activity*.

3. *Content Providers*

Suatu *content provider* mengelola seperangkat data aplikasi bersama yang dapat disimpan dalam sistem file, *database SQLite*, di web, atau lokasi penyimpanan permanen lainnya yang bisa diakses aplikasi. Melalui *content provider*, aplikasi lain bisa melakukan *query* atau bahkan memodifikasi data (jika *content provider* mengizinkannya). Misalnya, sistem Android menyediakan *content provider* yang mengelola informasi kontak pengguna. Karenanya, setiap aplikasi dengan izin yang sesuai bisa melakukan *query* mengenai bagian dari *content provider* untuk membaca dan menulis informasi tentang orang tertentu.

Content provider diimplementasikan sebagai subkelas *ContentProvider* dan harus mengimplementasikan seperangkat standar API (*Application Programming Interface*) yang memungkinkan aplikasi lain melakukan transaksi.

4. *Broadcast Receivers*

Broadcast receiver adalah komponen yang merespons pemberitahuan dalam lingkup sistem. Banyak *broadcast* yang berasal dari sistem misalnya, *broadcast* yang memberitahukan bahwa layar telah dinonaktifkan, baterai lemah, atau gambar telah direkam. Aplikasi juga bisa memulai *broadcast* misalnya untuk menginformasikan ke aplikasi lain bahwa sebagian data telah diunduh ke perangkat dan bisa digunakan aplikasi lain tersebut. *Broadcast receiver* diimplementasikan sebagai subkelas *BroadcastReceiver* dan setiap siaran dikirim sebagai objek *Intent*.

Karena sistem menjalankan setiap aplikasi dalam proses terpisah dengan izin file yang membatasi akses ke aplikasi lain, aplikasi pengembang tidak bisa langsung mengaktifkan komponen dari aplikasi lain. Akan tetapi, sistem Android bisa melakukannya. Jadi, untuk mengaktifkan komponen dalam aplikasi lain, pengembang harus mengirim pesan ke sistem yang menetapkan maksud pengembang untuk memulai komponen tertentu. Selanjutnya sistem akan mengaktifkan komponen untuk aplikasi pengembang [19].

2.2.7.4 Tipe Aplikasi Android

Terdapat tiga kategori aplikasi pada android dapat dilihat pada penjelasan sebagai berikut [20]:

1. *Foreground Applications*

   Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2. *Background Applications*

   Aplikasi yang memiliki interaksi terbatas dengan *user*, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tampak pada layar. Aplikasi ini sering mendengarkan pesan atau tindakan yang disebabkan oleh *hardware*, sistem, atau aplikasi lain, daripada mengandalkan interaksi pengguna.

3. *Intermittent Applications*

   Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. *Chatting* dan aplikasi *e-mail* adalah contoh yang khas.

   Aplikasi ini memungkinkan memperbarui UI *Activity* ketika itu terlihat dan mengirim pemberitahuan untuk memberitahu pengguna tentang pembaharuan ketika itu di latar belakang.

2.2.7.5 Kelebihan Android

Sudah banyak *platform* untuk perangkat seluler saat ini, termasuk di dalamnya *iPhone*, *Windows Mobile*, *BlackBerry*, dan sebagainya. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, Android adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan *Linux* dan *open source*. Pembuat perangkat menyukai hal

ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar *royality*. Sementara pengembang *software* menyukai karena Android dapat digunakan di perangkat mana pun dan tanpa terikat oleh *vendor* mana pun.

2. Arsitektur komponen dasar android terinspirasi dari teknologi *internet Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.

3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database* SQL, *browser* dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.

4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil. Pengguna tak perlu khawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.

5. Dukungan grafis dan suara terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh *Flash* menyatu dalam 3D menggunakan *OpenGL* memungkinkan membuat aplikasi maupun *game* yang berbeda.

6. *Portabilitas* aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin *virtual Dalvik*, sehingga kode program *portable* antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.2.8 Global Positioning System (GPS)

Global Positioning System (GPS) merupakan sistem untuk menentukan posisi dan navigasi secara global dengan menggunakan satelit. Sistem yang pertama kali dikembangkan oleh Departemen Pertahanan Amerika ini digunakan untuk kepentingan militer maupun sipil (survei dan pemetaan).

Sistem GPS, yang nama aslinya adalah NAVSTAR GPS (*Navigation Satellite Timing and Ranging Global Positioning System*), mempunyai tiga segmen yaitu [21]:

1. Satelit - bertugas untuk menerima dan menyimpan data yang ditransmisikan oleh stasiun-stasiun pengontrol, menyimpan dan menjaga informasi waktu dengan ketelitian tinggi (ditentukan dengan jam *atomic* di satelit), dan memancarkan sinyal dan informasi secara kontinu ke pesawat penerima (*receiver*) dari pengguna.
2. Pengontrol - bertugas untuk mengendalikan dan mengontrol satelit dari bumi baik untuk mengecek kesehatan satelit, penentuan dan prediksi orbit dan waktu, sinkronisasi waktu antar satelit, dan mengirim data ke satelit.
3. Penerima - bertugas menerima data dari satelit dan memprosesnya untuk menentukan posisi (posisi tiga dimensi yaitu koordinat di bumi plus ketinggian), arah, jarak dan waktu yang diperlukan oleh pengguna. Ada dua macam tipe penerima yaitu tipe NAVIGASI dan tipe GEODETIC. Yang termasuk *receiver* tipe NAVIGASI antara lain: *Trimble Ensign, Trimble Pathfinder, Garmin, Sony* dan lain sebagainya. Sedangkan tipe GEODETIC antara lain: *Topcon, Leica, Astech, Trimble* seri 4000 dan lain-lain.

Satelit GPS yang mengorbit bumi, dengan orbit dan kedudukan yang tetap (koordinatnya pasti), seluruhnya berjumlah 24 buah di mana 21 buah aktif bekerja dan 3 buah sisanya adalah cadangan. 24 satelit GPS tersebut berada sekitar 12.000 mil di atas bumi bergerak mengelilingi bumi 12 jam dengan kecepatan 7.000 mil per jam. Satelit GPS berkekuatan energi sinar matahari, memiliki baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak

ada energi matahari dan memiliki roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya.

2.2.8.1 Sistem Koordinat

Pengenalan tentang sistem koordinat sangat penting agar dapat menggunakan GPS secara optimum. Salah satu koordinat yang dipakai oleh GPS maupun dalam pemetaan yaitu: sistem koordinat global yang biasa disebut sebagai koordinat GEOGRAFI [21].

Koordinat GEOGRAFI diukur dalam lintang dan bujur dalam besaran derajat desimal, derajat menit desimal, atau derajat menit detik. Lintang diukur terhadap ekuator sebagai titik NOL (0^o sampai 90^o positif ke arah utara dan 0^o sampai 90^o negatif ke arah selatan). Bujur diukur berdasarkan titik NOL di *Greenwich* → 0^o sampai 180^o ke arah timur dan 0^o sampai 180^o ke arah barat.

Dalam penelitian ini, gps digunakan dalam aplikasi untuk mendapatkan titik koordinat di mana perangkat pengguna berada ketika melakukan proses dokumentasi, yang nantinya koordinat tersebut dibutuhkan untuk proses geotagging maupun untuk proses pada pengambilan suhu berdasarkan koordinat.

2.2.9 Google Maps

Google Maps adalah layanan gratis yang diberikan oleh Google dan sangat popular yang merupakan suatu peta dunia yang dapat kita gunakan untuk melihat suatu daerah. Layanan ini juga menyediakan API (*Application Programming Interface*) yang memungkinkan pengembang lain untuk memanfaatkan aplikasi ini di aplikasi buatannya.

Pengguna memiliki akses penuh ke peta, yang memungkinkannya untuk mengontrol tampilan, mengubah tingkat *zoom*, dan menggeser ke lokasi yang berbeda. Dengan menggunakan *Overlay*, pengembang dapat membubuhi keterangan peta dan menangani *input* pengguna [20].

Dengan menggunakan *GoogleMaps* API, pengembang dapat menghemat waktu dan biaya untuk membangun aplikasi peta digital yang andal, sehingga

pengembang dapat fokus hanya pada data-data yang akan ditampilkan. Dengan kata lain, pengembang hanya membuat suatu data sedangkan peta yang akan ditampilkan adalah milik Google sehingga pengembang tidak dipusingkan dengan membuat peta suatu lokasi, bahkan dunia.

Untuk menggunakan *Google Maps* dalam aplikasi Android, pengembang perlu memastikan bahwa mereka memeriksa API Google sebagai target pembangunannya. *Google Maps* bukan merupakan bagian dari SDK Android standar, sehingga pengembang perlu mencarinya di *Google API add-on* [16].

Google Maps digunakan dalam subsistem pada platform *website* yang dibangun pada penelitian ini untuk menampilkan lokasi dalam bentuk map di mana pengambilan foto dokumentasi kunjungan dilakukan.

2.2.10 Open Weather Map

OpenWeatherMap adalah layanan *online* yang menyediakan data cuaca terkini, termasuk data prakiraan dan data historis terkini untuk para pengembang layanan web dan aplikasi *mobile*. Untuk sumber data, *OpenWeatherMap* menggunakan layanan siaran meteorologi, data mentah dari stasiun cuaca bandara, data mentah dari stasiun radar, dan data mentah dari stasiun cuaca resmi lainnya.

Semua data diproses oleh *OpenWeatherMap* dengan cara mencoba menyediakan data perkiraan cuaca dan peta cuaca *online* yang akurat, seperti awan atau curah hujan. Di luar itu, layanan difokuskan pada aspek sosial dengan melibatkan pemilik stasiun cuaca dalam menghubungkan layanan dan dengan demikian meningkatkan akurasi data cuaca. Ideologi ini terinspirasi oleh *OpenStreetMap* dan *Wikipedia* yang membuat informasi gratis dan tersedia untuk semua orang. *OpenWeatherMap* menggunakan *OpenStreetMap* untuk menampilkan peta cuacanya.

OpenWeaterMap dalam penelitian ini digunakan dalam subsistem pada platform *mobile* untuk mendapatkan suhu dari lokasi berdasarkan koordinat gps di mana perangkat pengguna berada.

2.2.11 Geotagging

Geotagging adalah proses menambahkan informasi geografis ke berbagai media dalam bentuk *metadata*. Data biasanya terdiri dari koordinat seperti lintang dan bujur, tapi bahkan mungkin termasuk ketinggian, jarak dan nama tempat. *Geotagging* yang diterapkan pada gambar atau foto juga dikenal sebagai GPS *photo tagging*, yang merupakan proses penambahan informasi posisi data GPS (*Latitude*, *Longitude, Altitude*) ke dalam *metadata* sebuah foto digital.

Ponsel-ponsel berkamera yang memiliki GPS *receiver* internal umumnya memiliki fitur ini. Mekanisme GPS *photo tagging* adalah pada saat sebuah foto diambil menggunakan kamera (digital atau ponsel) yang memiliki fitur *geotagging*, kamera atau ponsel tersebut mencatat lebih banyak informasi/data yaitu berupa tambahan informasi geografis ke dalam foto tersebut, dibandingkan dengan sebuah foto yang diambil dengan kamera biasa.

Foto dalam perangkat digital menyimpan *metadata* di EXIF (*Exchangeable Image File Format*), termasuk waktu, orientasi dari kamera, apakah menggunakan lampu *flash* dan detail kamera lainnya yang digunakan seperti *aperture*, *exposure*, dan *focal length*, dan sebagainya. Data EXIF memungkinkan kita untuk mencari dan mengelola foto secara efektif dengan berbagai kondisi melalui beberapa atribut [22].

Pada penelitian ini *Geotagging* digunakan dalam subsistem pada platform *mobile* dalam proses dokumentasi kegiatan kunjungan lapangan.

2.2.12 Reverse Geocoding

Istilah *geocoding* umumnya mengacu pada penerjemahan alamat yang bisa dibaca orang menjadi sebuah lokasi pada peta. Proses melakukan kebalikannya, yaitu menerjemahkan lokasi pada peta menjadi alamat yang bisa dibaca orang, dikenal sebagai *Reverse Geocoding*. *Reverse Geocoding* menampilkan alamat jalan dari lokasi fisik yang ditentukan oleh lintang dan bujur [20]. Parameter yang diperlukan dalam *Reverse Geocoding* adalah *latlng* atau *placeId*.

Latlng merupakan nilai garis lintang dan garis bujur yang menetapkan lokasi yang diinginkan dalam memperoleh alamat terdekat yang bisa dibaca orang. Sedangkan *placeId* adalah ID tempat dari lokasi yang diinginkan dalam memperoleh alamat yang bisa dibaca orang. ID tempat adalah sebuah *identifier* unik yang bisa digunakan bersama Google API lainnya. Misalnya, pengembang bisa menggunakan *placeId* yang dikembalikan oleh *Google Maps Roads API* untuk mendapatkan alamat titik yang telah diidentifikasi. Nilai ini hanya bisa ditetapkan jika permintaan menyertakan kunci API atau ID klien *Google Maps API* [23].

Dalam penelitian ini, *reverse geocoding* digunakan dalam subsistem pada platform *mobile* untuk menerjemahkan koordinat dari gps menjadi alamat yang nantinya mudah untuk dibaca dan digunakan dalam sistem.

2.2.13 Object Oriented Programming

Object Oriented Programming (OOP) atau Pemrograman berorientasi objek adalah salah satu ide mutakhir berkaitan dengan metodologi pemrograman komputer. Metode ini bertujuan untuk memudahkan dan menyistematiskan pekerjaan programmer. Pada pemrograman berorientasi objek, perangkat lunak secara keseluruhan adalah sebuah sistem yang dibuat dari berbagai komponen atau objek, masing-masing dengan fungsi khusus dan berinteraksi satu dengan yang lainnya dengan cara yang telah ditetapkan [24].

Menurut Booch [25], pemrograman berorientasi objek adalah metode implementasi di mana program diorganisasikan sebagai kumpulan objek yang bekerja sama, masing-masing objek merepresentasikan instan dari kelas, dan kelas-kelas itu anggota suatu hierarki kelas-kelas yang disatukan lewat keterhubungan pewarisan.

Pemrograman berorientasi objek menggunakan metode pembentukan sistem di mana komponen sistem (objek) sering kali terbentuk dari objek-objek lain yang lebih kecil. Dengan demikian, pada pemrograman berorientasi objek, ketergabungan objek-objek merupakan salah satu aspek yang paling dasar.

Bahasa pemrograman berorientasi objek dapat meningkatkan cara ekspresi program secara besar-besaran, tapi juga dapat meningkatkan peluang kekeliruan pemrogram yang sembrono. Menjadi pemrogram yang baik memerlukan disiplin tinggi dalam menjalankan prinsip-prinsip dan menjalankan gaya pemrograman yang baik. Program yang mengikuti petunjuk-petunjuk perancangan yang baik cenderung lebih mudah diguna ulang, mudah diperluas dan *debugging* dapat dilakukan lebih mudah.

2.2.14 Unified Modeling Language

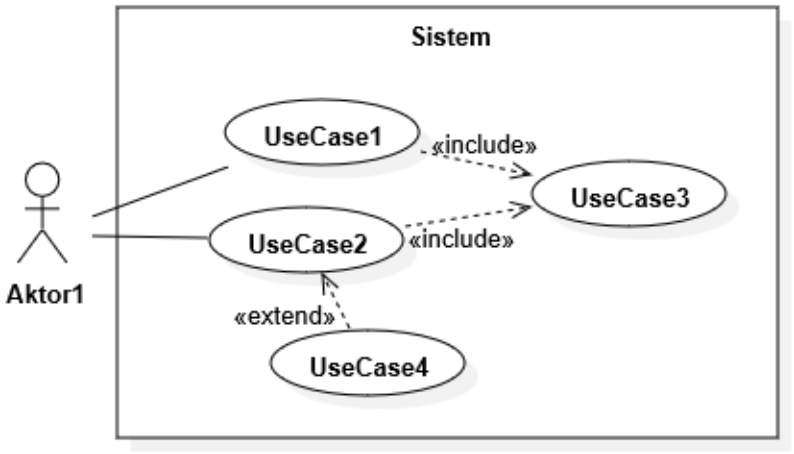
Unified Modeling Language (UML) adalah bahasa standar untuk menulis cetak biru perangkat lunak. UML dapat digunakan untuk memvisualisasikan, menentukan, membangun, dan mendokumentasikan artefak dari sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung pada pengembangan, tidak bergantung bahasa dan teknologi, pemanduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh gaya-gaya yang diintegrasikan lewat XML. Standar UML dikelola oleh OMG (*Object Management Group*) [25].

Pengembangan UML dimulai dari kerja sama Grady Booch dan James Rumbaugh pada 1994 untuk mengombinasikan dua metodologi terkenal yaitu Booch dan OMT. Kemudian Ivan Jacobson, pencipta metode OOSE (*Object Oriented Software Engineering*) bergabung. Usulan UML diberikan kepada OMG (*Object Management Group*) sebagai konsorsium standardisasi teknologi objek agar UML dijadikan bahasa dan notasi pemodelan dilakukan pada tahun 1997. OMG menerima UML, UML telah menjadi standar *de-facto* karena pencipta-penciptanya sangat popular. Banyak pengembang perangkat lunak yang mengadopsi UML [25].

2.2.14.1 Use-Case Diagram

Diagram *use-case* (*use-case diagram*) merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram *use-case* menunjukkan sekumpulan *use-case, actor* dan hubungannya. Diagram *use-case*

digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Sebuah *use-case* menggambarkan bagaimana pengguna berinteraksi dengan sistem dengan mendefinisikan langkah yang diperlukan untuk mencapai tujuan tertentu. Dalam diagram *use-case*, tindakan yang dapat dilakukan ditampilkan dalam bentuk oval. Para aktor dihubungkan dengan garis pada tindakan yang dapat mereka lakukan [26]. Contoh *use-case* dapat dilihat pada Gambar 2.10. berikut.



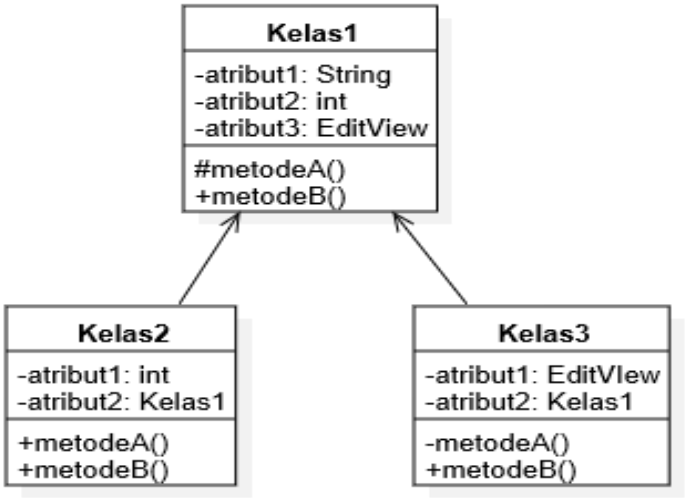
Gambar 2.10. Contoh Use-Case Diagram

2.2.14.2 Class Diagram

Class Diagram atau Diagram kelas merupakan diagram paling umum dipakai di semua pemodelan berorientasi objek. Pemodelan kelas merupakan pemodelan paling utama di pendekatan berorientasi objek. Diagram kelas menunjukkan aspek statis sistem terutama untuk mendukung kebutuhan fungsional sistem. Kebutuhan fungsional berarti layanan-layanan yang harus disediakan sistem ke pemakai. Salah satu maksud diagram kelas adalah untuk mendefinisikan fondasi bagi diagram-diagram lain di mana aspek-aspek lain dari sistem ditunjukkan [25].

Sebuah diagram kelas memberikan pandangan statis atau struktural dari sistem. Diagram kelas tidak menunjukkan sifat dinamis dari komunikasi antara objek dari kelas dalam diagram. Unsur-unsur utama dari diagram kelas adalah kotak, yang merupakan ikon yang digunakan untuk mewakili kelas dan *interface*. Setiap kotak dibagi menjadi bagian-bagian horizontal. Bagian atas berisi nama kelas. Bagian tengah berisi daftar atribut dari kelas. Atribut dapat berupa nilai yang

di dapat dari variabel instan atau nilai di dapat dari objek lain yang tersusun. Bagian ketiga dari diagram kelas berisi operasi atau perilaku kelas. Operasi mengacu pada apa yang objek dari kelas dapat lakukan, biasanya diimplementasikan sebagai metode kelas [26]. Contoh *class diagram* dapat dilihat pada Gambar 2.11. berikut.



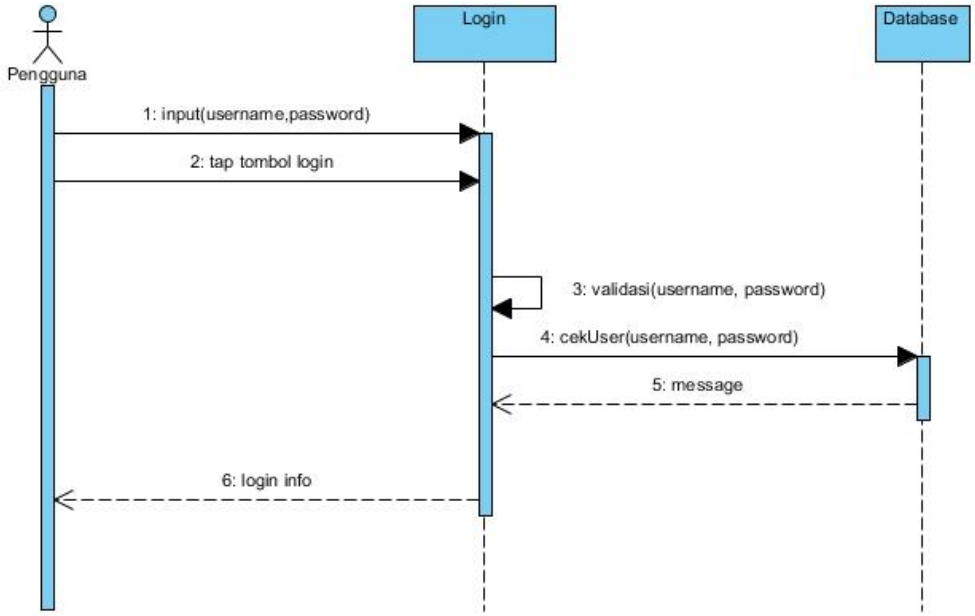
Gambar 2.11. Contoh Class Diagram

2.2.14.3 Sequence Diagram

Urutan kejadian digambarkan dengan diagram sekuen (*sequence diagram*). Diagram sekuen mendeskripsikan komunikasi di antara objek-objek, meliputi pesan-pesan yang ada dan urutan pesan tersebut muncul. Diagram sekuen digunakan untuk memodelkan skenario penggunaan. Skenario penggunaan adalah barisan kejadian yang terjadi selama satu eksekusi sistem. Cakupan skenario dapat beragam, dari mulai semua kejadian di sistem atau hanya kejadian di objek-objek tertentu.

Diagram sekuen menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horizontal dari objek pengiriman ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukkan barisan kejadian, bukan pewaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan kejadian [25].

Sebagian mungkin menggunakan diagram urutan untuk menunjukkan interaksi dalam satu *use-case* atau dalam satu skenario dari sistem perangkat lunak [26]. Contoh *sequence diagram* dapat dilihat pada Gambar 2.12. berikut.



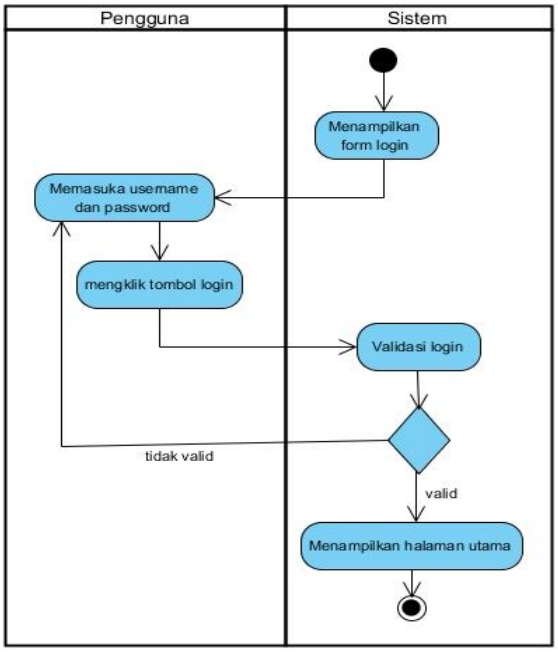
Gambar 2.12. Contoh Sequence Diagram

2.2.14.4 Activity Diagram

Pada dasarnya, *activity diagram* atau diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas berupa operasi-operasi dan aktivitas-aktivitas di *use-case*.

Sebuah diagram aktivitas menggambarkan perilaku dinamis dari sistem atau bagian dari sistem melalui aliran kontrol antara tindakan yang sistem lakukan. Komponen utama dari sebuah diagram aktivitas adalah simpul tindakan, ditunjukkan oleh persegi panjang bulat, yang sesuai dengan tugas yang dilakukan oleh sistem perangkat lunak. Panah dari satu tindakan *node* ke yang lain menunjukkan aliran kontrol. Artinya, panah antara dua tindakan simpul berarti bahwa setelah aksi pertama selesai, aksi yang kedua dimulai. Sebuah titik hitam solid membentuk simpul awal yang menunjukkan titik awal dari aktivitas tersebut. Sebuah titik hitam yang dikelilingi oleh lingkaran hitam adalah simpul akhir

menunjukkan akhir aktivitas [26]. Contoh *activity diagram* dapat dilihat pada Gambar 2.13. berikut.



Gambar 2.13. Contoh Activity Diagram

2.2.15 Skala Likert

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena. Dan pada evaluasi, skala likert digunakan untuk menilai keberhasilan suatu kebijakan atau program, menilai manfaat pelaksanaan suatu kebijakan atau program, dan mengetahui kepuasan *stakeholder* terhadap pelaksanaan suatu kebijakan atau program.

Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert. Skala likert hanya menggunakan item yang secara pasti baik dan secara pasti buruk. Skala likert menggunakan ukuran ordinal, karenanya hanya dapat membuat rangking,

tetapi tidak dapat diketahui berapa kali satu responden lebih baik atau lebih buruk dari responden lainnya dalam skala [9].

2.2.15.1 Prosedur Pembuatan Skala Likert

Prosedur dalam pembuatan skala likert adalah sebagai berikut.

1. Peneliti mengumpulkan *item-item* yang relevan dengan masalah yang sedang diteliti, dan terdiri atas *item* yang cukup jelas disukai dan tidak disukai.
2. Kemudian *item-item* tersebut dicoba kepada sekelompok responden yang cukup representatif dari populasi yang diteliti.
3. Responden di atas diminta untuk mengecek tiap *item*, apakah ia menyenangi atau tidak menyukainya. Responsi tersebut dikumpulkan dan jawaban yang memberikan indikasi menyenangi diberikan skor tertinggi. Tidak ada masalah untuk memberikan angka 5 untuk yang tertinggi dan skor 1 untuk yang terendah atau sebaliknya. Yang penting adalah konsistensi dari arah sikap yang diperlihatkan. Demikian juga, apakah jawaban "setuju" atau "tidak setuju" disebut yang disenangi, tergantung dari isi pertanyaan dan isi dari *item-item* yang disusun.
4. Total skor dari masing-masing individu adalah penjumlahan dari skor masing-masing *item* dari individu tersebut.
5. Respons dianalisis untuk mengetahui *item-item* mana yang sangat nyata batasan antara skor tinggi dan skor rendah dalam skala total. Misalnya, respons responden pada *upper* 25% dan *lower* 25% dianalisis untuk melihat sampai berapa jauh tiap item dalam kelompok ini berbeda. *Item-item* yang tidak menunjukkan beda yang nyata, apakah masuk ke dalam skor tinggi atau rendah juga dibuang untuk mempertahankan konsistensi internal dari pernyataan.

2.2.15.2 Keuntungan Skala Likert

Berikut adalah keuntungan dari skala likert.

1. Dalam menyusun skala, item-item yang tidak jelas menunjukkan hubungan dengan sikap yang sedang diteliti masih dapat dimasukkan dalam skala.

2. Skala likert lebih mudah dalam pembuatannya dan lebih mudah dalam penafsirannya.
3. Skala likert mempunyai reliabilitas yang relatif tinggi untuk jumlah item yang sama. Skala likert dapat memperlihatkan respons yang dinyatakan dalam beberapa respons alternatif (sangat setuju, setuju, bimbang, tidak setuju, sangat tidak setuju) tentang senang tidak senang terhadap suatu item.
4. Jangka responsi yang lebih besar membuat skala likert dapat memberikan keterangan yang lebih nyata dan jelas tentang pendapatan atau sikap responden tentang isu yang dipertanyakan.

2.2.15.3 Kelemahan Skala Likert

Skala likert mempunyai beberapa kelemahan, antara lain sebagai berikut.

1. Ukuran yang digunakan adalah ukuran ordinal, skala likert hanya dapa mengurutkan individu dalam skala, tetapi tidak dapat membandingkan berapa kali satu individu lebih baik dari individu lain.
2. Kadang kala total skor dari individu tidak memberikan arti yang jelas, karena banyak pola respons terhadap beberapa item akan memberikan skor yang sama.

2.2.16 K-Nearest Neighbor

K-nearest neighbor adalah algoritma *supervised learning* di mana hasil dari *instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori K-tetangga terdekat. Tujuan dari algoritma ini adalah untuk mengklasifikasikan obyek baru berdasarkan atribut dan sampel-sampel dari data *training*. Algoritma *K Nearest neighbor* menggunakan *neighborhood classification* sebagai nilai prediksi dari nilai *instance* yang baru [27].

K-nearest neighbor bekerja berdasarkan jarak minimum (termirip) dari data baru ke data *training* untuk menentukan K tetangga terdekat. Setelah itu, dapatkan nilai mayoritas sebagai hasil prediksi dari data yang baru tersebut. Data berisi banyak atribut (X_1, X_2, …, Xn) digunakan untuk mengklasifikasikan atribut target (Y_1, Y_2, ..., Yn).

Ada banyak cara untuk mengukur jarak minimum antara data baru dengan data *training*, di antaranya *euclidean distance* dan *manhattan distance* (*city block distance*), yang paling sering digunakan adalah *euclidean distance* [28], yaitu :

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ ... \ + \ (a_n - b_n)^2}$$

Berikut ini langkah-langkah dari algoritma *K-Nearest Neighbor* (KNN):

1. Tentukan parameter K = jumlah banyaknya tetangga terdekat
2. Hitung jarak antara data baru dan semua data yang ada di data *training*.
3. Urutkan jarak tersebut dan tentukan tetangga mana yang terdekat berdasarkan jarak minimum ke-K.
4. Tentukan kategori dari tetangga terdekat.
5. Gunakan kategori mayoritas yang sederhana dari tetangga yang terdekat tersebut sebagai nilai prediksi dari data yang baru.

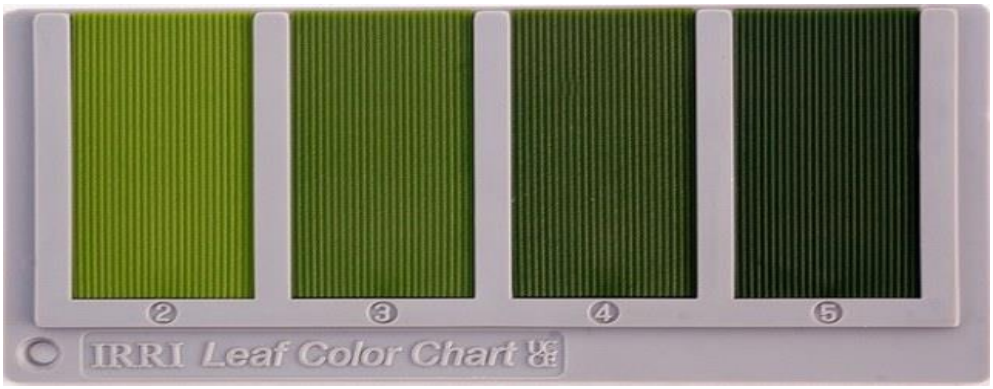
Dalam penelitian ini, KNN digunakan dalam subsistem pada platform *mobile* untuk pengklasifikasian citra digital berdasarkan nilai rata-rata RGB pada saat proses analisis nitrogen menggunakan perangkat *mobile* pengguna.

2.2.17 Bagan Warna Daun

Bagan Warna Daun (BWD) adalah salah satu alat yang digunakan oleh untuk mengevaluasi keadaan tanaman padi seperti kadar pupuk nitrogen (N) yang dibutuhkan oleh tanaman padi. BWD dikembangkan oleh *International Rice Research Institute* (IRRI) dengan tujuan memantau pertumbuhan tanaman padi. BWD awalnya dikembangkan di Jepang untuk membantu petani menentukan intensitas warna daun yang berhubungan langsung dengan kandungan klorofil dan status N dalam daun. Peneliti Tiongkok dari *Zhejiang Agricultural University* kemudian mengembangkan BWD dan mengalibrasinya untuk padi indica, japonica, dan padi hibrida [3].

Skala warna, yang tersusun dari suatu seri warna hijau, dari hijau kekuningan sampai hijau tua, sesuai dengan warna-warna daun di lapang, dapat digunakan untuk mengukur warna daun. Bila suatu nilai warna daun lebih rendah

dari batas kritis tertentu, maka tanaman memerlukan pupuk N tambahan. Warna daun adalah suatu indikator yang berguna bagi kebutuhan pupuk N tanaman padi. Daun yang berwarna pucat atau hijau kekuningan menunjukkan bahwa tanaman kekurangan N. Berikut ini Gambar 2.14. merupakan gambar alat bagan warna daun skala 4.



Sumber gambar : http://rkb.irri.org/images/

Gambar 2.14. Alat Bagan Warna Daun

2.2.17.1 Waktu Penggunaan

Waktu penggunaan BWD dapat melalui dua pendekatan, yaitu [3]:

1. Berdasarkan kebutuhan tanaman, dengan membandingkan warna daun padi secara berkala dengan skala warna pada BWD setiap 7 sampai 10 hari sejak tanaman berumur 21-45 HST. Tanaman segera diberi pupuk N bila warna daun berada di bawah skala 4 BWD. Dengan cara ini maka perlu sering ke sawah untuk membandingkan warna daun padi dengan BWD.
2. Berdasarkan waktu yang telah ditetapkan, yaitu pada saat pembentukan anakan aktif (21-28 HST) dan primordia (35-40 HST). Dengan cara ini hanya diperlukan dua kali pengukuran warna daun padi dengan BWD.

2.2.17.2 Cara Penggunaan

Untuk menggunakan BWD, dipilih secara acak 10 rumpun tanaman sehat pada hamparan yang seragam, kemudian dipilih daun teratas yang telah membuka penuh pada satu rumpun. Bagian tengah daun diletakkan di atas BWD dan dibandingkan antara warna daun dan skala warna pada BWD. Pada waktu mengukur warna daun dengan BWD, perlu dihindari menghadap sinar matahari karena pantulan sinar matahari dari daun padi dapat memengaruhi pengukuran warna daun.

Waktu pembacaan adalah pagi atau sore hari. Sebaiknya pengukuran dilakukan pada waktu yang sama dan oleh orang yang sama. Jika 6 atau lebih dari 10 daun yang diamati warnanya berada dalam batas kritis, yaitu di bawah skala 4, maka tanaman perlu segera diberi pupuk N susulan sesuai dengan tingkat hasil di tempat bersangkutan.

Nilai warna daun dengan BWD	Hasil (t/ha GKG)			
	~ 5,0	~ 6,0	~ 7,0	~ 8,0
	Takaran pupuk urea yang digunakan (kg/ha), berdasarkan waktu yang telah ditetapkan			
2 – 3	7 5	100	125	150
Antara 3 dan 4	5 0	7 5	100	125
4 – 5	0	0–50	5 0	5 0
	Takaran pupuk urea yang digunakan (kg/ha), berdasarkan kebutuhan riil tanaman			
Di bawah 4	5 0	7 5	100	125

Sumber gambar : Jurnal Litbang Pertanian [3].

Gambar 2.15. Takaran pupuk urea sesuai dengan nilai warna daun pada BWD berdasarkan waktu yang telah ditetapkan dan kebutuhan riil tanaman

2.2.18 Kunjungan Lapangan

Kunjungan lapangan adalah salah satu kegiatan yang terencana yang dilakukan dengan cara melihat kondisi lapangan yang terkait dengan suatu kegiatan tertentu. Suatu kunjungan lapangan biasanya dilakukan oleh seseorang atau sekelompok orang berkenaan dengan kegiatan berkunjung ke tempat khusus untuk tujuan khusus. Tujuan tersebut mungkin untuk mengamati situasi, mengamati kegiatan atau praktik, atau membawa menemui seseorang atau objek. Keberhasilan suatu kegiatan kunjungan lapangan tergantung pada seberapa baiknya perencanaan yang dibuat.

2.2.19 Nitrogen

Nitrogen (N) merupakan gas tidak berwarna, tidak berbau, tidak berasa, dan sebagian besar merupakan gas diatomik. Nitrogen merupakan elemen penting bagi kehidupan karena merupakan salah satu penyusun DNA, dan dengan demikian merupakan bagian dari kode genetik. Dalam air dan tanah, nitrogen ditemukan pada senyawa nitrat dan nitrit. Nitrogen dihasilkan oleh berbagai industri sehingga meningkatkan kadar nitrat dan nitrit dalam tanah dan air. Penggunaan komersial terbesar nitrogen adalah sebagai komponen dalam pembuatan amonia yang kemudian digunakan sebagai pupuk dan untuk memproduksi asam nitrat.

Nitrogen (N) merupakan salah satu unsur hara utama dalam tanah yang sangat berperan dalam merangsang pertumbuhan dan memberi warna hijau pada daun. Kekurangan nitrogen dalam tanah menyebabkan pertumbuhan dan perkembangan tanaman terganggu dan hasil tanaman menurun karena pembentukan klorofil yang sangat penting untuk proses fotosintetis terganggu.

Adapun peranan nitrogen bagi tanaman adalah berperan dalam pertumbuhan vegetatif tanaman, memberikan warna pada tanaman, panjang umur tanaman, penggunaan karbohidrat, dan lain-lain. Kekurangan salah satu atau beberapa unsur hara akan mengakibatkan pertumbuhan tanaman tidak sebagaimana mestinya yaitu ada kelainan atau penyimpangan-penyimpangan dan banyak pula tanaman yang mati muda yang sebelumnya tampak layu dan mengering.

Adapun gejala yang ditimbulkan akibat dari kekurangan dan kelebihan unsur N bagi tanaman adalah sebagai berikut [29] :

1. Efek kekurangan unsur nitrogen bagi tanaman dapat menyebabkan pertumbuhan tanaman menjadi kerdil, warna daun menguning, produksi menurun, fase pertumbuhan terhenti dan kematian.
2. Efek dari kelebihan unsur nitrogen bagi tanaman dapat menyebabkan kualitas buah menurun, menyebabkan rasa pahit (seperti pada buah timun), produksi menurun, daun lebat dan pertumbuhan vegetatif yang cepat dan dapat menyebabkan keracunan pada tanaman.

2.2.20 Tanaman Padi

Padi adalah salah satu tanaman budidaya terpenting dalam peradaban manusia dan merupakan sumber karbohidrat utama bagi mayoritas penduduk dunia. Padi merupakan tanaman yang termasuk genus *Orzya L*. yang meliputi kurang lebih 25 spesies, tersebar di daerah tropis dan daerah subtropis, seperti Asia, Afrika, Amerika dan Australia. Padi yang ada sekarang merupakan persilangan antara *Oryza officianalis* dan *Oryza sativa F. Spontane* [30].

O. sativa terdiri dari dua sub spesies, yaitu: *indica* dan *japonica*. *O. sativa* sub spesies *indica* dibudidayakan di daerah selatan Pegunungan Himalaya dan *O. sativa* sub spesies *japonica* didomestikasi di bagian selatan China. *O. sativa* sub spesies *japonica* memiliki ciri berdaun sempit dan berwarna hijau tua, bentuk biji membulat, lebar dan tebal memiliki bulu yang panjang atau ada juga yang tidak berbulu rambut pada glume tebal dan panjang [31].



Sumber gambar : http://lapaknasa.com

Gambar 2.16. Tanaman Padi

2.2.20.1 Morfologi Tanaman Padi

Morfologi suatu tanaman sangat berpengaruh terhadap produktivitas misal, efektivitas menangkap radiasi surya, suhu mikro tajuk tanaman dan ketersediaan air bagi tanaman akibat perakaran yang berbeda dalam penyebarannya. Pemahaman tentang bentuk dan fungsi dari organ-organ tanaman padi diperlukan antara lain untuk merancang tipe tanaman ideal.

1. Akar

Akar tanaman padi berfungsi sebagai penguat/penunjang tanaman untuk dapat tumbuh tegak, menyerap hara dan air dari dalam tanah untuk selanjutnya diteruskan ke organ lainnya di atas tanah. Akar tanaman padi termasuk golongan akar serabut, akar primer (radikula) yang tumbuh sewaktu berkecambah bersama akar-akar lain yang muncul dari janin dekat bagian buku skutellum disebut akar seminal yang jumlahnya antara 1-7.

2. Batang

Batang padi berfungsi sebagai penopang tanaman, penyalur senyawa-senyawa kimia dan air dalam tanah serta sebagai cadangan makanan. Hasil tanaman

yang tinggi harus didukung dengan batang padi yang kokoh, bila tidak maka tanaman padi akan rebah terutama didaerah yang sering dilanda angin kencang.

Batang terdiri atas beberapa ruas yang dibatasi oleh buku, daun dan tunas (anakan) tumbuh pada buku. Pada permukaan stadia tumbuh batang yang terdiri atas pelepah-pelepah daun ruas-ruas yang tertumpuk padat, ruas-ruas tersebut kemudian memanjang dan berongga setelah tanaman memasuki stadia reproduktif.

3. Daun

Daun merupakan bagian dari tanaman yang berwarna hijau karena mengandung klorofil (zat hijau daun) adanya klorofil ini menyebabkan daun tanaman dapat mengolah sinar radiasi surya menjadi karbohidrat/energi untuk tumbuh kembangnya organ-organ tanaman lainnya atau disebut sebagai *sources*. Daun tanaman padi tumbuh pada batang dalam susunan yang berselang-selang satu daun pada tiap buku. Tiap daun terdiri atas (1) helai daun, (2) pelepah daun yang membungkus ruas, (3) telinga daun (*auricle*), (4) lidah daun (*ligule*). Daun teratas disebut daun bendera yang posisi dan ukurannya tampak berbeda dari daun yang lain. Panjang daun lebih beragam dibandingkan dengan lebarnya dan dikaitkan dengan sudut daun.

4. Bunga dan Malai

Bunga padi secara keseluruhan disebut malai, tiap unit bunga pada malai dinamakan spikelet yang pada hakikatnya adalah bunga yang terdiri atas tangkai, bakal buah, lemma, palea, putik dan benang sari serta beberapa organ lainnya yang bersifat inferior. Tiap unit bunga pada malai terletak pada cabang-cabang bulir yang terdiri atas cabang primer dan sekunder. Tiap unit bunga padi pada hakikatnya adalah *floret* yang hanya terdiri dari atas satu bunga. Pada jenis rumput-rumputan lainnya, *floret* tersebut dapat terdiri atas beberapa bunga yang merupakan ciri bentuk-bentuk primitif.

5. Gabah

Gabah terdiri atas biji yang terbungkus oleh sekam, biji yang sehari-hari dikenal dengan nama beras pecah kulit adalah karyopsis yang terdiri atas janin

(embrio) dan endosperma yang diselimuti oleh lapisan aleuron, kemudian tegmen dan lapisan terluar disebut perikarp. Dalam jenis-jenis japonika, sekam terdiri atas gulma rudimenter dan bagian dari tangkai gabah (pedicel), sedangkan pada jenis-jenis indika, sekam dibentuk oleh palea, lemma mandul dan rakhila.

2.3 Perangkat Lunak Pendukung

Dalam pembangunan sistem ini digunakan beberapa perangkat lunak pendukung atau *software* yang dapat mendukung pada pembuatan sistem aplikasi ini. Adapun perangkat lunak yang dibutuhkan yaitu:

2.3.1 Hypertext Prepocessor

Hypertext Prepocessor (PHP) adalah salah satu bahasa pemrograman yang berjalan dalam sebuah web server dan berfungsi sebagai pengolah data pada sebuah server. Untuk membuat *website* yang dinamis dan mudah diunggah setiap saat dari browser, dibutuhkan sebuah program yang mampu mengolah data dari komputer *client* atau dari komputer server itu sendiri sehingga mudah dan nyaman untuk disajikan di browser. Salah satu program yang dapat dijalankan di server dan cukup andal adalah PHP. Dengan menggunakan program PHP, sebuah *website* akan lebih interaktif dan dinamis [32].

Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (situs personal) dan PHP itu sendiri pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995, dan pada saat PHP masih bernama FI (*Form Interpreter*), yang wujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data *form* dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum. PHP adalah sebuah bahasa *scripting* yang terpasang pada HTML. Sebagian besar sintaknya mirip dengan bahasa pemrograman C, Java, ASP dan Perl, ditambah beberapa fungsi PHP yang spesifik dan mudah dimengerti [33].

PHP itu sendiri sudah dapat berinteraksi dengan beberapa *database* walaupun dengan kelengkapan yang berbeda, yaitu seperti: *DBM, FilePro (Personic, Inc), Informix, Ingres, InterBase, Microsoft Access, MSSQL, MySQL, Oracle, PostgrSQL*, dan *Sybase* [33].

Cara kerja PHP yaitu: pertama *client* web browser atau pengguna memakai komputer kemudian pengguna tersebut menjalankan file PHP itu di web browser, kemudian file PHP itu dikirim ke web server. Web server mengirimkan lagi *Engine PHP* atau mesin PHP dan di dalam mesin PHP itu diproses, dan setelah diproses oleh mesin PHP maka akan berbentuk file HTML, dan file HTML ini akan dikirim ke web server dan web server akan memberikannya ke pengguna.

Bahasa pemrograman PHP dalam penelitian ini digunakan untuk membangun subsistem pada platform *website* yang nantinya dapat diakses melalui jaringan internet dengan bantuan web browser pada perangkat desktop pengguna.

2.3.2 MySQL

MySQL adalah sebuah program *database* server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multi-user* serta menggunakan perintah standar SQL (*Structured Query Language*). *MySQL* memiliki dua bentuk lisensi, yaitu *FreeSoftware* dan *Shareware. MySQL* yang biasa digunakan adalah *MySQL FreeSoftware* yang berada di bawah Lisensi GNU/GPL (*General Public License*).

MySQL merupakan sebuah *database* server yang *free*, artinya pengguna bebas menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. *MySQL* pertama kali dirilis oleh seorang programmer *database* bernama Michael Widenius. Selain sebagai *database* server, *MySQL* juga merupakan program yang dapat mengakses suatu *database MySQL* yang berposisi sebagai server. Pada saat itu berarti program pengguna berposisi sebagai *Client*. Jadi *MySQL* adalah sebuah *database* yang dapat digunakan baik sebagai *Client* maupun *Server*.

Database MySQL memiliki beberapa kelebihan dibanding *database* lain, di antaranya adalah [34]:

1. *MySQL* sebagai *Database Management System* (DBMS)
2. *MySQL* sebagai *Relation Database Management System* (RDBMS)

3. *MySQL* adalah sebuah *Software database* yang *OpenSource*, yang artinya program ini bersifat *free* atau bebas digunakan oleh siapa saja tanpa harus membeli dan membayar lisensi kepada pembuatnya.
4. *MySQL* merupakan sebuah *database* server, jadi dengan menggunakan *database* ini pengguna dapat menghubungkannya ke media internet sehingga dapat diakses dari jauh.
5. *MySQL* merupakan sebuah *database client*. Selain menjadi server yang melayani permintaan, *MySQL* juga dapat melakukan *query* yang mengakses *database* pada server. Jadi *MySQL* dapat juga berperan sebagai *client*.
6. *MySQL* mampu menerima *query* yang bertumpuk dalam satu permintaan atau yang disebut *Multi-Threading*.
7. *MySQL* merupakan sebuah *database* yang mampu menyimpan data berkapasitas sangat besar hingga berukuran Gigabyte sekalipun.
8. *MySQL* didukung oleh *driver* ODBC, artinya *database MySQL* dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti *Delphi* maupun *Visual Basic*.
9. *MySQL* merupakan Server *database* yang *multi-user*, artinya *database* ini tidak hanya digunakan oleh sepihak orang, akan tetapi merupakan *database* yang dapat digunakan oleh banyak pengguna.
10. *MySQL* didukung oleh sebuah komponen C dan Perl API, sehingga *database MySQL* dapat diakses melalui sebuah program aplikasi yang berada di bawah *protocol* internet berupa Web. Biasanya aplikasi yang sering digunakan adalah PHP dan Perl.

Dengan beberapa kelebihan yang dimiliki di atas, *MySQL* menjadi sebuah program *database* yang sangat popular digunakan. Pada umumnya *MySQL* digunakan sebagai *database* yang diakses melalui web.

Dalam *MySQL* ada beberapa operasi dasar yang lebih dikenal dengan operasi CRUD sebagai berikut:

1. *Create*

Create adalah operasi penambahan data baru ke dalam tabel. *Query*-nya adalah sebagai berikut:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. *Read*

Read adalah operasi untuk menampilkan semua atau sebagian data yang berada di dalam *database*. *Query*-nya adalah sebagai berikut:

```
SELECT column_name, column_name
FROM table_name;
```

3. *Update*

Update adalah operasi untuk mengubah data yang ada di dalam *database*. *Query*-nya adalah sebagai berikut:

```
UPDATE table_name SET column1=value1, column2 = value2, ...
WHERE some_column=some_value;
```

4. *Delete*

Delete adalah operasi untuk menghapus data yang ada di dalam tabel. *Query*-nya adalah sebagai berikut:

```
DELETE FROM table_name
WHERE some_column = some_value;
```

MySQL pada penelitian ini digunakan dalam subsistem pada platform *website* yang disimpan dalam server sebagai pusat penyimpanan data dari sistem yang dibangun.

2.3.3 Java

Sejarah bahasa pemrograman Java dimulai pada tahun 1990, James Gosling, programmer yang bekerja pada Sun Microsystems mendapat tugas untuk menciptakan program untuk mengendalikan perangkat elektronika rumah tangga. Gosling dan kawan-kawan memulai mendesain program tersebut dengan C++, bahasa program kelas satu yang menjadi unggulan para programmer profesional saat itu karena sifatnya yang berorientasi objek itu. Saat mulai menggunakan C++, Gosling dan timnya mulai mendapat masalah karena berbagai kerumitan yang dimiliki oleh C++, misalnya masalah pewarisan majemuk (*multiple inheritance*) dan masalah *bug* program, misalnya kebocoran memori (*memory leaks*). Gosling segera memutuskan untuk merancang sendiri bahasa pemrograman komputer yang sederhana dan mampu menghindarkannya dari masalah-masalah yang ia temui ketika menggunakan bahasa pemrograman C++ [24].

Meskipun Gosling tidak menyukai kerumitan yang dimiliki oleh C++, tetapi ia menyukai sintak dasar dan sifat berorientasi objek yang dimiliki oleh C++. Oleh karena itu, ketika mulai merancang bahasa pemrogramannya sendiri, ia menggunakan C++ sebagai modelnya dengan membuang fitur-fitur yang membuatnya susah. Rancangan bahasa pemrograman yang telah selesai ia buat, diberi nama Oak (konon diilhami oleh pohon Oak yang tumbuh di depan jendela kantornya).

Oak dipakai dalam proyek *video-on-demand* (VOD), yakni sebagai perangkat lunak yang mengendalikan sistem televisi interaktif. Meskipun proyek Star7 dan VOD tidak pernah menjadi produk yang dipasarkan, Sun Microsystems memberikan kesempatan agar bahasa pemrograman Oak berkembang dan matang. Pada saat itulah Sun mengganti nama bahasa pemrograman Oak yang dibuat timnya menjadi bahasa pemrograman Java (konon diilhami saat minum kopi salah seorang timnya mengatakan Java, maksudnya asal bijih kopi yang sedang diminumnya) pada tanggal 23 Mei 1995.

Java dirancang agar sederhana namun andal. Lebih penting lagi, Java adalah bahasa pemrograman yang *multi-platform*, yang berarti bahwa Java dapat

berjalan pada sistem operasi mana pun serta semua program yang dibuat dengan Java dapat dijalankan pada sistem komputer mana pun. Sifat *multi-platform* ini didapatkan dengan menggunakan format khusus untuk mengompilasi program-program Java. Format file yang digunakan adalah *byte-code*, ini dapat dibaca dan dieksekusi oleh sembarang sistem operasi yang memiliki *Interpreter Java*.

Sun Sendiri mendeskripsikan Java sebagai bahasa pemrograman yang sederhana, berorientasi-objek, bertipe *static*, terkompilasi, *multi-threaded, garbage collected*, netral arsitektur, *robust*, aman, dan dapat diperluas. Rincian sifat-sifat Java yang dideskripsikan dapat dipaparkan sebagai berikut [24]:

1. Sederhana

Pengembangan Java banyak membuang fitur-fitur yang tidak diperlukan seperti yang dimiliki bahasa pemrograman tingkat tinggi lainnya, sebagai contoh java menghilangkan *pointer mathematic*, *casting type implicit*, struktur atau *unions*, *operator overloading*, *template*, *header filer*, maupun *multiple inheritance*.

2. Berorientasi Objek

Seperti halnya C++, Java menggunakan kelas-kelas untuk mengelola kode ke dalam modul-modul logika. Pada saat *runtime* program menciptakan obyek dari kelas-kelas yang ada. Kelas-kelas Java dapat merupakan warisan dari kelas lain, tetapi pewarisan majemuk di mana suatu kelas mewariskan metode-metode dan *field* yang berasal lebih dari satu kelas tidak diperkenankan.

3. Bertipe Statik

Semua objek yang digunakan dalam program harus dideklarasikan sebelum digunakan. Cara ini memungkinkan kompilator Java melaporkan dan melokalisir terjadinya konflik-konflik tipe.

4. Terkompilasi

Sebelum dapat menjalankan program yang ditulis dengan bahasa pemrograman Java, program harus dikompilasi oleh kompilator Java. Hasil kompilasinya berupa file kode *byte* (*byte-code*), yang mirip dengan file kode mesin,

yang dapat dieksekusi di bawah sistem operasi apa pun asal memiliki *interpreter java.*

5. *Multi-threaded*

Program Java dapat berisi eksekusi *multiple threads*, yang memungkinkan program mengulas citra pada layar pada satu *thread*, sementara tetap menerima masukan dari *keyboard* pada *thread* utamanya.

6. *Garbage Collected*

Program Java menyingkirkan sendiri "sampah-sampah" tidak berguna, artinya program tidak perlu menghapus objek-objek yang dialokasikan di memori. Ini menjadikan program bebas dari masalah-masalah pengelolaan memori.

7. Bebas arsitektur

Pada dasarnya Java tidak dirancang untuk prosesor ataupun sistem operasi tertentu. Ia dirancang untuk bekerja pada berbagai arsitektur prosesor dan berbagai sistem operasi.

8. Tangguh (*Robust*)

Karena interpreter Java memeriksa seluruh akses sistem yang dilakukan program, maka program Java tidak akan membuat sistem menjadi *crash*. Bila terjadi masalah serius, program Java membuat perkecualian (*exception*). *Exception* ini dapat ditangani dan dikelola oleh program tanpa berisiko memacetkan sistem.

9. Aman

Sistem Java tidak hanya memverifikasi seluruh akses ke memori, tetapi juga menjamin bahwa tidak ada virus yang "membonceng" pada *applet* yang sedang berjalan. Karena Java tidak mendukung *pointer*, program tidak dapat memperoleh akses ke area sistem di mana ia tidak otorisasi.

10. Dapat diperluas

Program Java mendukung metode *native* yakni fungsi-fungsi yang ditulis dalam bahasa lain, biasanya C++. Dukungan terhadap metode *native* ini

memungkinkan programmer menulis fungsi yang bisa dieksekusi secara lebih cepat daripada fungsi yang sama yang ditulis dalam Java.

Pada penelitian ini, bahasa pemrogrman Java digunakan untuk membangun aplikasi atau subsistem pada platform *mobile* dengan sistem operasi Android.

2.3.4 JavaScript Object Notation

JavaScript Object Notation (JSON) 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 Pemrograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apa pun 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 [35].

JSON terbuat dari dua struktur, yaitu:

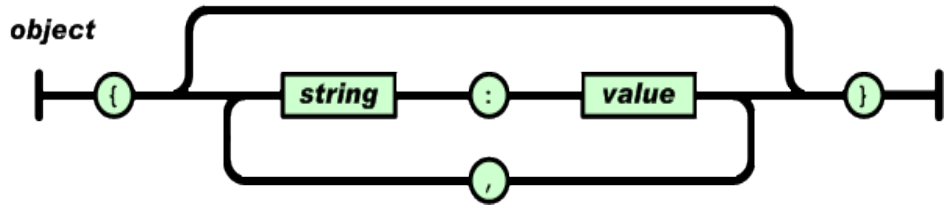
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). 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 pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan.

Berikut ini merupakan bentuk yang digunakan oleh JSON.

1. Objek (*Object*) adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung

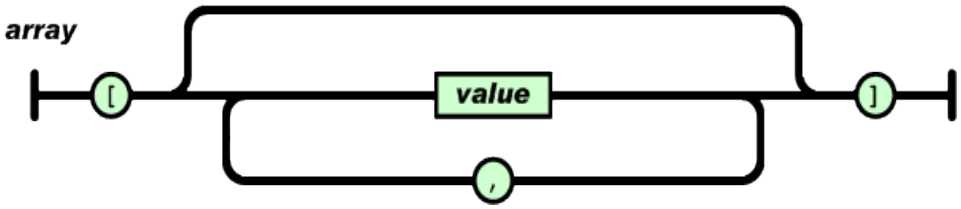
kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



Sumber gambar : http://www.json.org/

Gambar 2.17. Bentuk Objek pada JSON

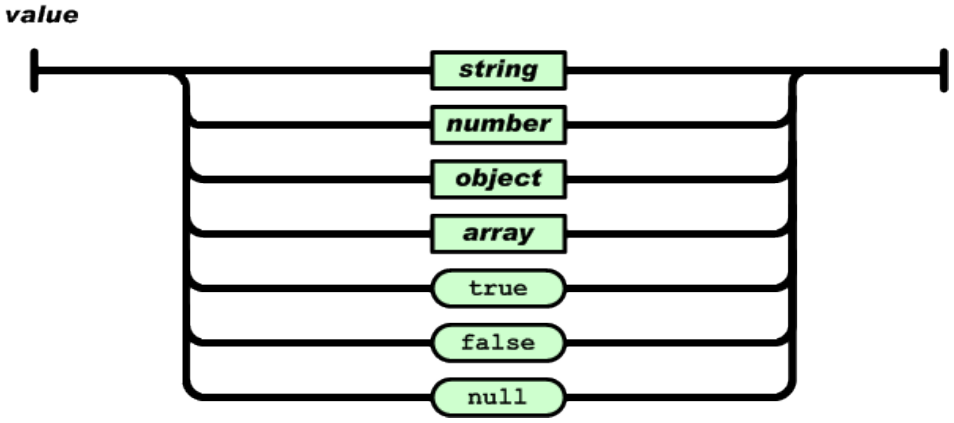
2. Larik (*Array*) adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



Sumber gambar : http://www.json.org/

Gambar 2.18. Bentuk Larik pada JSON

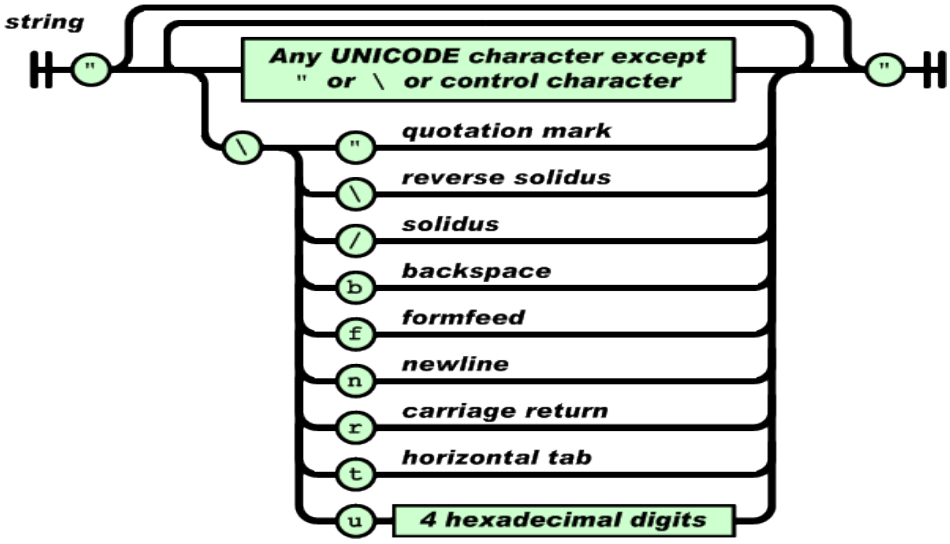
3. Nilai (*value*) dapat berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.



Sumber gambar : http://www.json.org/

Gambar 2.19. Bentuk Nilai pada JSON

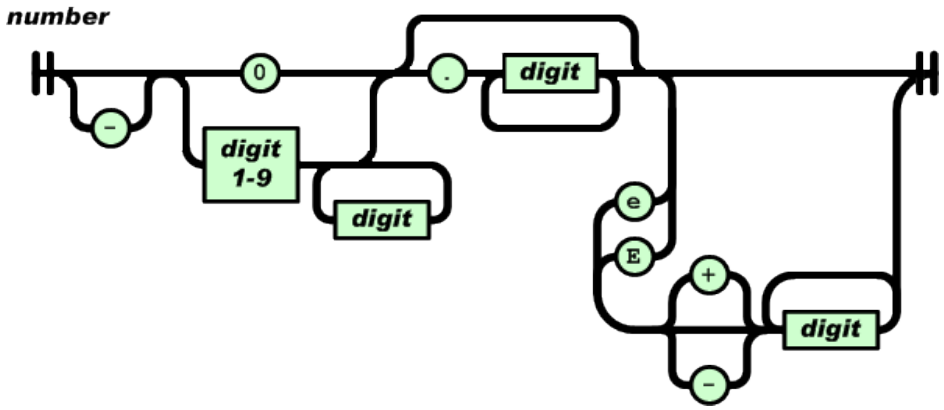
4. *String* adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan *string* C atau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



Sumber gambar : http://www.json.org/

Gambar 2.20. Bentuk String pada JSON

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



Sumber gambar : http://www.json.org/

Gambar 2.21. Bentuk Angka pada JSON

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

JavaScript Object Notation (JSON) pada penelitian ini digunakan sebagai media pertukaran data antara *server* sebagai penyedia data dengan *client* yang berupa perangkat *mobile* yang dibuat dalam *Application Programming Interface* (API).

2.3.5 Atom Text Editor

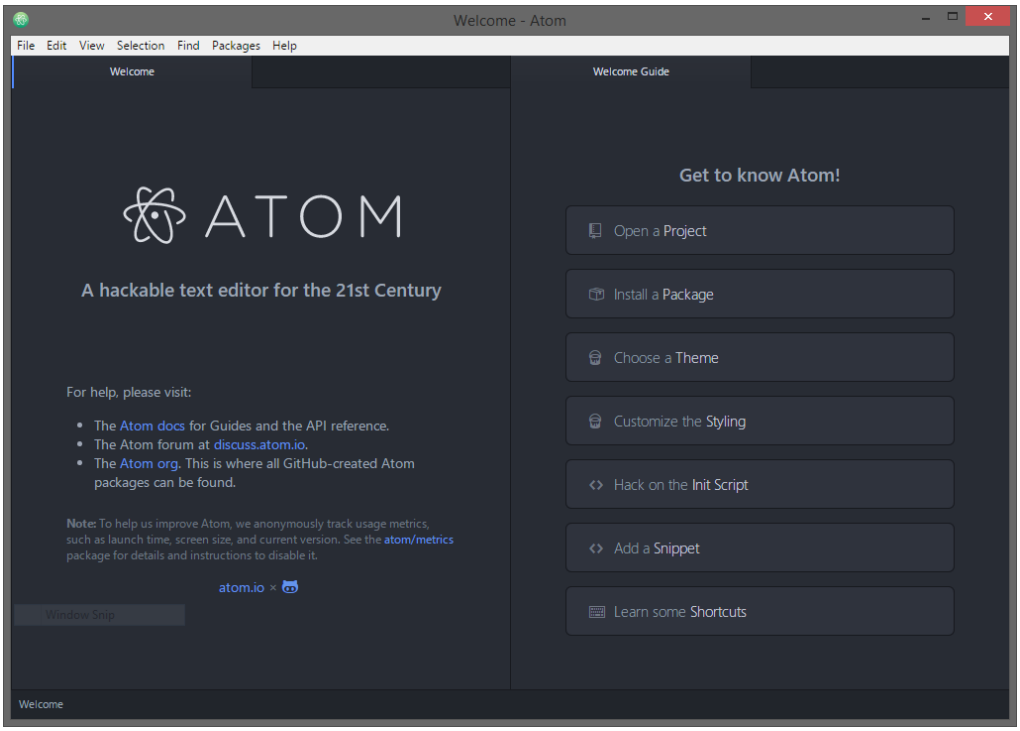
Sejak menjadi *official project* pada tahun 2011, Atom yang pada awalnya bernama *Atomicity* terus menerus dikembangkan demi memenuhi kebutuhan penggunanya. Atom merupakan *text editor* modern yang dibuat oleh GitHub dan dirilis pertama kali pada tahun 2014. Atom memiliki *tagline* "*A hackable text editor for the 21st Century*" yang dengan tegas menerangkan bahwa text editor ini sangat mudah untuk dikustomisasi sesuai kebutuhan pengguna. Pengguna tetap dapat melakukan kustomisasi untuk menyesuaikannya sesuai kebutuhan tanpa harus menyentuh file konfigurasinya. Tercatat 660 tema, 2.090 *packages/plugins* termasuk *plugin* penting seperti *linter*, *autocomplete-plus*, dan *minimap* [36].

Berikut adalah fitur-fitur yang dimiliki Atom *text editor*.

1. *Cross-platform editing* - Atom dapat berjalan pada berbagai sistem operasi, meliputi Windows, Mac OSX, serta Linux.
2. *Built-in Package Manager* - Pengguna dapat mencari dan menginstall berbagai *package* untuk mempermudah kegiatan *coding*-nya, atau bahkan membuat *package* itu sendiri langsung melalui Atom.
3. *Smart Autocompletion* - Dengan Atom pengguna dapat menulis kode dengan lebih cepat dengan fitur *autocomplete* yang pintar dan *flexible*.
4. *File System Browser* - Pengguna dapat dengan mudah menelusuri, membuka suatu file, keseluruhan projek, atau beberapa projek dalam satu window.
5. *Multiple Panes* - Pengguna dapat membagi tampilan Atom menjadi beberapa panel, untuk membandingkan serta mengedit beberapa file kode sekaligus.
6. *Find and Replaces* - Temukan dan ganti teks yang diketikkan pada sebuah file, atau pada keseluruhan projek sekaligus.
7. *Packages* – Di mana pengguna bisa memilih ribuan paket *open source* yang dapat menambahkan berbagai fitur dan fungsi baru pada Atom *text editor* sesuai kebutuhannya, atau membuat paket sendiri dan membagikannya untuk digunakan oleh orang lain.
8. *Themes* - Atom *text editor* hadir dengan empat tema UI, dan delapan tema warna sintak bawaan baik dalam warna gelap ataupun cerah, serta masih tersedia lebih banyak tema yang bisa digunakan melalui Atom *community* yang bisa diinstall langsung melalui Atom *Text Editor*, atau pengguna bisa membuatnya sendiri.
9. *Customization* - Pengguna dapat mengkustomisasi Atom *text editor* dengan mudah. Pengguna dapat menggunakan *CSS/Less* untuk mengubah tampilannya, atau menggunakan *HTML* dan *JavaScript* untuk menambahkan berbagai macam fitur.

10. *Under the hood* - Atom *text editor* merupakan aplikasi desktop yang dibangun menggunakan integrasi antara *HTML*, *JavaScript*, *CSS*, dan *Node.js*. Atom berjalan pada *Electron*, sebuah *framework* untuk membuat aplikasi desktop *cross-platform* menggunakan teknologi web.

Berikut ini merupakan tampilan awal dari *Atom Text Editor*.



Sumber gambar : https://sourceforge.net/projects/atomportable

Gambar 2.22. Atom Text Editor

Pada penelitian ini, Atom *Text Editor* digunakan untuk melakukan pemrograman atau pengkodean dalam bahasa PHP untuk membangun subsistem pada platform *website*.

2.3.6 Web Server

Web server adalah suatu program komputer yang mempunyai tanggung jawab atau tugas menerima permintaan HTTP dari komputer klien, yang dikenal dengan nama web browser, dan melayani mereka dengan menyediakan respons

HTTP berupa konten data, biasanya berupa halaman web yang terdiri dari dokumen HTML, objek terkait seperti gambar, dan lain-lain [33].

Untuk dapat menjalankan PHP yang disertai *database MySQL* dapat digunakan dua jenis web server yaitu *Online mode* dan *Offline mode*. Pada *Online mode,* selain komputer kita harus mempersiapkan domain dan *hosting* serta koneksi internet yang memadai untuk mengelolanya sehingga harus keluar biaya ekstra terlebih dahulu. Sedangkan pada *Offline mode* yang diperlukan cukup komputer dan beberapa *software* untuk membuat web server lokal atau *localhost.*

Ada beberapa jenis *software* untuk membangun web server lokal atau *localhost* yang *support* sistem operasi Windows di antaranya adalah:

1. WampServer
2. XAMPP
3. AppServ
4. PHP Triad
5. Vertigo

Software di atas merupakan gabungan dari *PHP*, *MySQL database* dan *Apache*. Walaupun masing-masing program web server secara detail berbeda tetapi pada umumnya program web server memiliki fitur-fitur dasar yang sama seperti berikut:

1. HTPP : Setiap program web server bekerja dengan menerima permintaan HTTP dari klien, dan memberikan respons HTTP kepada klien tersebut. Respons HTTP biasanya mengandung dokumen HTML tetapi dapat juga berupa berkas RAW, gambar, dan berbagai jenis dokumen lainnya.

2. *Logging* : Umumnya setiap web server mempunyai kemampuan untuk melakukan pencatatan atau *logging* terhadap informasi detail mengenai permintaan klien dan respons dari web server dan disimpan dalam berkas log, dengan adanya berkas log ini maka akan memudahkan web master untuk mendapat statistik dengan menggunakan *tool log analyzer*.

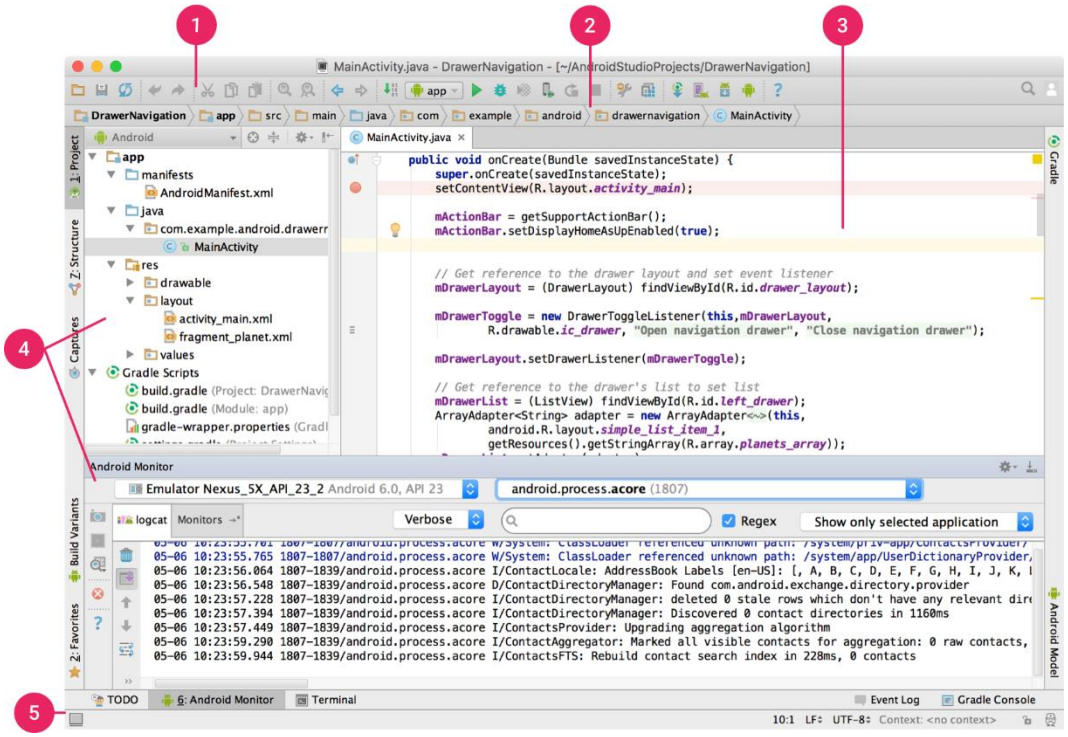
2.3.7 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu atau *Integrated Development Environment* (IDE) resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Tujuannya dibuat untuk Android adalah untuk mempercepat pengembangan dan membantu pengguna membuat aplikasi berkualitas tinggi untuk setiap perangkat Android. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas pengguna saat membuat aplikasi Android, misalnya [37]:

1. Sistem pembuatan berbasis Gradle yang fleksibel;
2. Emulator yang cepat dan kaya fitur;
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android;
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru;
5. *Template* kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh;
6. Alat penguji dan kerangka kerja yang ekstensif;
7. Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain;
8. Dukungan C++ dan NDK; dan
9. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine.*

2.3.7.1 Antarmuka Pengguna

Jendela utama Android Studio terdiri dari beberapa bidang logika yang diidentifikasi dalam Gambar 2.23. berikut ini.



Sumber gambar : https://developer.android.com/studio

Gambar 2.23. Antarmuka Pengguna Android Studio

1. Bilah Alat - memungkinkan pengguna untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.

2. Bilah Navigasi - akan membantu pengguna menavigasi di antara proyek dan file yang dibuka untuk pengeditan. Di sini tampilan struktur yang terlihat tampak lebih ringkas daripada jendela *Project*.

3. Jendela Editor - merupakan tempat untuk membuat dan mengubah kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor akan menampilkan *Layout* Editor.

4. Jendela Alat - memberi pengguna akses ke tugas-tugas spesifik seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Pengguna bisa meluaskan dan juga menciutkannya.

5. Bilah Status - menampilkan status proyek dan IDE itu sendiri, serta setiap peringatan atau pesan.

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

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

2.3.7.2 Gaya dan Pemformatan

Saat pengguna melakukan pengeditan, Android Studio otomatis menerapkan pemformatan dan gaya seperti yang ditetapkan dalam setelan gaya kode. Pengguna dapat menyesuaikan setelan gaya kode dengan bahasa pemrograman, termasuk menetapkan konvensi untuk tab dan inden, spasi, pembungkusan dan tanda kurung kurawal, dan baris kosong. Untuk menyesuaikan setelan gaya kode, klik **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** pada Mac).

Meski IDE otomatis menerapkan pemformatan saat pengguna bekerja, pengguna juga dapat secara eksplisit memanggil tindakan *Reformat Code* dengan menekan **Control+Alt+L** (**Opt+Command+L** pada Mac), atau inden otomatis semua baris dengan menekan **Control+Alt+I** (**Alt+Option+I** pada Mac).

Berikut adalah Gambar 2.24. yang menunjukkan kode sebelum dilakukan pemformatan dan Gambar 2.25. yang menunjukkan kode setelah dilakukan pemformatan.

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

Sumber gambar : https://developer.android.com/studio

Gambar 2.24. Kode Sebelum Pemformatan

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

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

Formatted 7 lines
Show reformat dialog: ⌥⇧⌘L

Sumber gambar : https://developer.android.com/studio

Gambar 2.25. Kode Setelah Pemformatan

2.3.7.3 Fitur-Fitur

Berikut ini adalah fitur-fitur Android Studio yang diberikan kepada pengguna untuk meningkatkan produktivitas pengguna saat membuat aplikasi Android [37].

1. *Instant Run*

Saat pengguna mengeklik *Run* atau *Debug*, fitur *Instant Run* Android Studio akan mendorong perubahan kode dan sumber daya ke aplikasi pengguna yang sedang berjalan. Secara cerdas memahami perubahan dan sering kali menyelesaikannya tanpa harus memulai ulang aplikasi atau membuat kembali APK sehingga pengguna bisa melihat dampaknya seketika itu juga.

2. Editor kode cerdas

Editor kode membantu pengguna menulis kode yang lebih baik, bekerja lebih cepat, dan lebih produktif dengan menawarkan pelengkapan kode canggih, pemfaktoran ulang, dan analisis kode. Saat pengguna mengetik, Android Studio

menyediakan saran dalam daftar tarik turun. Cukup tekan Tab untuk memasukkan kode ini.

3. Emulator yang cepat dan kaya fitur

Android Emulator memasang dan memulai aplikasi pengguna lebih cepat daripada perangkat sebenarnya dan memungkinkan pengguna untuk membuat purwarupa dan menguji aplikasi di semua konfigurasi perangkat Android: perangkat ponsel, tablet, *Android Wear*, dan Android TV. Pengguna juga bisa menyimulasikan berbagai fitur perangkat keras seperti lokasi GPS, latensi jaringan, dan *input* multi-sentuh.

4. Sistem versi yang kuat dan fleksibel

Android Studio menawarkan otomatisasi versi, manajemen dependensi, dan konfigurasi versi yang bisa disesuaikan. Pengguna bisa mengonfigurasi proyek untuk menyertakan perpustakaan lokal dan di-host, serta mendefinisikan varian versi yang menyertakan kode dan sumber daya berbeda, juga menerapkan penciutan kode dan konfigurasi penandatanganan aplikasi.

5. Didesain untuk tim

Android Studio mengintegrasikan alat bantu kontrol versi, seperti GitHub dan *Subversion*, agar pengguna dapat membuat tim tetap selaras dengan proyek dan perubahan versi. Sistem versi Gradle sumber terbuka memungkinkan pengguna membuat versi yang sesuai dengan lingkungannya dan berjalan pada server integrasi berkesinambungan seperti *Jenkins*.

6. Dikembangkan untuk semua perangkat Android

Android Studio menyediakan lingkungan yang menyatu tempat pengguna dapat membuat aplikasi untuk ponsel dan tablet Android, *Android Wear*, Android TV, dan Android Auto. Modul kode terstruktur memungkinkan pengguna untuk membagi proyek menjadi unit-unit fungsi yang bisa pengguna buat, uji, dan debug secara independen.

7. Template kode dan aplikasi contoh

Android Studio menyertakan proyek dan *template* kode yang memudahkan penambahan pola-pola yang sudah mapan seperti panel samping navigasi dan pager tampilan. Pengguna juga bisa mengimpor aplikasi yang berfungsi penuh dari GitHub, langsung dari layar *Create Project*. Pengguna bisa bereksperimen dan menggunakan kembali aplikasi-aplikasi ini yang dibuat oleh Google dan pihak-pihak lain.

8. *Lintelligence*

Android Studio menyediakan kerangka kerja analisis statis yang kuat dan menyertakan lebih dari 280 pemeriksaan lint berlainan di seluruh aplikasi. Selain itu, Android Studio juga menyediakan perbaikan cepat yang membantu pengguna mengatasi berbagai masalah di kategori berbeda, seperti kinerja, keamanan, dan keakuratannya, dengan sekali klik.

9. Alat bantu & kerangka kerja pengujian

Android Studio menyediakan alat bantu dan kerangka kerja yang beragam untuk membantu menguji aplikasi Android pengguna. Dengan cepat membuat dan menjalankan kode pengujian untuk aplikasi pengguna, termasuk *JUnit 4* dan pengujian UI fungsional. Pengguna bisa menjalankan pengujian pada perangkat, emulator, lingkungan integrasi berkesinambungan, atau dalam *Firebase Test Lab*.

10. Dukungan C++ dan NDK

Android Studio memungkinkan pengguna menggunakan C++ dan Android NDK bersama-sama kode Java. Menyediakan penyorotan sintak dan pemfaktoran ulang untuk C++ dan *debugger* berbasis *lidb* sehingga pengguna bisa men-*debug* Java dan C++ secara bersamaan.

11. Integrasi *Cloud*

Alat bantu bawaan untuk *Google Cloud Platform* membantu pengguna membuat dan menerapkan *backend* untuk aplikasi Androidnya seperti penggunaan layanan *Google Cloud Endpoint* dan *Firebase Cloud Messaging*.

12. *Vector Asset Studio*

Android Studio memudahkan pembuatan aset gambar yang baru untuk setiap ukuran kepadatan. Dengan *Vector Asset Studio*, pengguna bisa memilih dari ikon desain material yang disediakan Google atau memuat file SVG sendiri. *Vector Asset Studio* selanjutnya menghasilkan file bitmap untuk setiap kepadatan gambar untuk mendukung versi Android lama yang tidak mendukung format vektor Android yang dapat digambar.

13. Editor Terjemahan

Editor Terjemahan memberi pengguna tampilan tunggal dari semua sumber daya yang telah pengguna terjemahkan, sehingga mengubah atau menambah terjemahan serta menemukan terjemahan yang tidak ada tanpa harus membuka setiap versi file strings.xml. Editor bahkan menyediakan tautan untuk memesan layanan terjemahan.

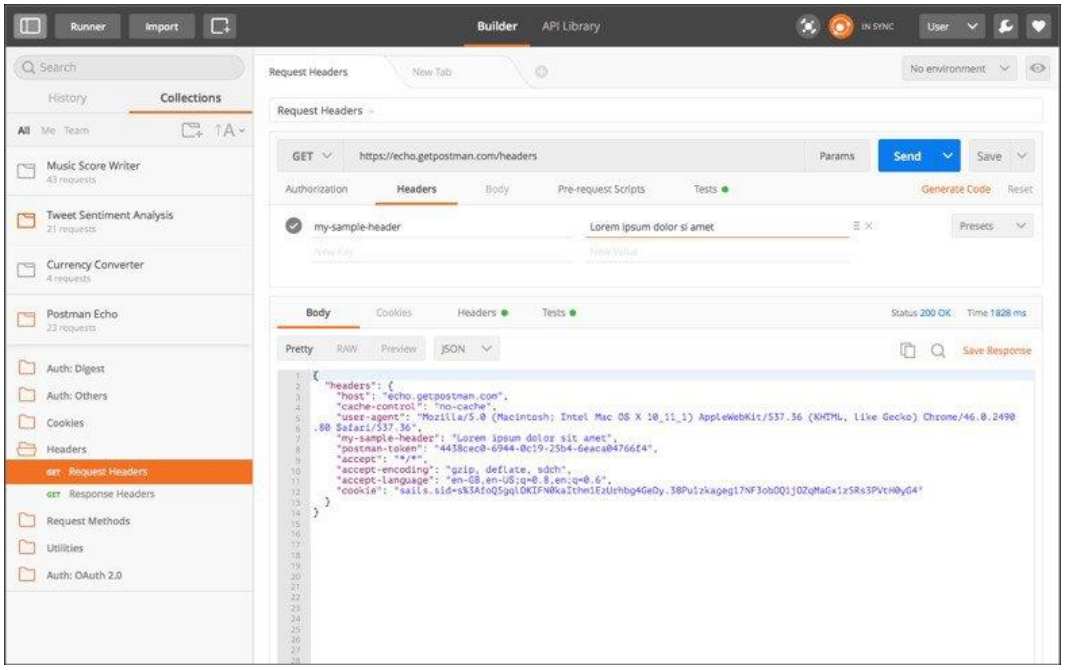
Aplikasi Android Studio pada penelitian ini digunakan untuk melakukan pemrograman atau pengkodean menggunakan bahasa pemrograman Jawa untuk membangun subsistem pada platform *mobile* dengan sistem operasi android.

2.3.8 Postman

Postman adalah *toolchain* penting bagi pengembang API untuk berbagi, uji coba, dokumentasi dan memantau API. Lebih dari 3 juta insinyur dan pengembang di seluruh dunia menggunakan *Postman* untuk membangun perangkat lunak yang terhubung melalui API-cepat, mudah dan akurat [38].

Ide untuk *Postman* muncul ketika para pendiri bekerja bersama-sama, dan frustrasi dengan alat yang ada untuk pengujian API. Mereka merasa harus ada bahasa yang lebih baik bagi para pengembang untuk berkomunikasi tentang API. Hal ini menyebabkan penciptaan *Postman*. *Postman* membantu pengembang membangun, menguji, dan mendokumentasikan lebih cepat [38].

Gambar 2.26. berikut menunjukkan tampilan dari aplikasi *Postman*.



Sumber gambar : https://www.getpostman.com

Gambar 2.26. Tampilan Aplikasi Postman

Pada penelitian ini, aplikasi Postman digunakan untuk melakukan uji coba terhadap API yang dibangun untuk mengetahui keberhasilan atau kesalahan yang terjadi pada API tersebut.

2.3.9 XAMPP

XAMPP adalah sebuah *software* web server *Apache* yang di dalamnya sudah tersedia *database* server *MySQL* dan *support PHP programming*. XAMPP berfungsi sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apa pun), *Apache*, *MySQL*, PHP dan Perl. Program ini berada dalam GNU *General Public License* dan bebas [39].

XAMPP dalam penelitian ini digunakan untuk menjalankan subsistem pada platform website secara *offline* dan melakukan *priview* sehingga dapat memodifikasi *website* tanpa harus *online* atau terakses dengan internet.